การสร้างโปรแกรมเข้ารหัสลับโดยใช้พหุหลุมวนดึงดูด

Chaotic Encryption Software using Multi-chaotic Attractors

โดย

นาย สุธี รุ่งเรืองทวีพงศ์

นาย แสงชัย หาญกล้า

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2550

Chaotic Encryption Software using Multi-chaotic Attractors

By

Mr. Suthee Rungruangthaweephong

Mr. Sangchai Hankla

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

BACHELOR IN DEPARTMENT OF INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2007

ใบเสนอปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	การสร้างโปรแกรมเข้ารหัสลับโดยใช้พหุหลุมวนดึงดูด
ชื่อนักศึกษา	นาย สุธี รุ่งเรืองทวีพงศ์ รหัส 47010867
	นาย แสงชัย หาญกล้า รหัส 47010913
อาจารย์ที่ปรึกษา	รศ.ดร.ปิติเขต สู้รักษา
	ผศ.กฤดากร กล่อมการ
ระดับการศึกษา	ปริญญาตรี วิศวกรรมศาสตรบัณฑิต
	สาขาวิศวกรรมสารสนเทศ
	ภาควิชา วิศวกรรมสารสนเทศ
ปีการศึกษา	2550

ปริญญานิพนธ์ฉบับนี้ได้รับการเห็นชอบจากอาจารย์ที่ปรึกษาเป็นที่เรียบร้อยแล้ว

.....

(รศ.ดร.ปิติเขต สู้รักษา)

อาจารย์ผู้ควบคุมวิทยานิพนธ์

.....

(ผศ.กฤดากร กล่อมการ)

อาจารย์ผู้ควบคุมวิทยานิพนธ์

หัวข้อปริญญานิพนธ์ การสร้างโปรแกรมเข้ารหัสลับโดยใช้พหุหลุมวนดึงดูด
ชื่อนักศึกษา นายสุธี รุ่งเรืองทวีพงศ์ รหัส 47010867
นายแสงชัย หาญกล้า รหัส 47010913
อาจารย์ที่ปรึกษา รศ.ดร.ปิติเขต สู้รักษา
ผศ.กฤดากร กล่อมการ
ระดับการศึกษา ปริญญาตรี วิศวกรรมศาสตรบัณฑิต
สาขาวิศวกรรมสารสนเทศ
ภาควิชา วิศวกรรมสารสนเทศ
ปีการศึกษา 2550

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เสนอการสร้างโปรแกรมเข้ารหัสลับข้อมูลเพื่อเพิ่มความปลอดภัยและมั่นคงของข้อมูล โดยใช้ทฤษฎี เคออส(Chaos) เข้ามาใช้ในการสร้าง กุญแจรหัส ของการเข้ารหัสลับ และ ปริญญานิพนธ์นี้ เสนอระเบียบวิธีทางคณิตศาสตร์แบบต่างๆ ในการแก้สมการเชิงอนุพันธ์ ซึ่งเป็นสมการจากทฤษฎี เคออส โดยได้เปรียบเทียบผล และความเร็วในการสร้างกุญแจรหัส เพื่อนำมาประยุกต์ในการเขียน โปรแกรม แล้วจึงนำโปรแกรมนี้ไปทำงานบนอุปกรณ์ไมโครคอนโทรเลอร์ ARM7 เพื่อใช้เป็นอุปกรณ์เข้ารหัสข้อความอักษร (Text)

Thesis Title	Chaotic Encryption software using Multi-chaotic Attractors	
Student	Mr Suthee Rungruangthaweephong	ID. 47010867
	Mr. Sangchai Hankla	ID. 47010913
Advisor	Assoc.Prof. D.r. Pitikhate Sooraksa	
	Asst.Prof. Kiddakron Klomkarn	
Graduate Level	Bachelor Degree of Information Engineering	
Department	Information Engineering	
Academic Year	2007	

ABSTRACT

This thesis presents of encryption program for information security by using chaos theory for key generation. The thesis presents the numerical analysis for solving ordinary differential equations. The encryption program is written based on comparing speed of key generation by an ARM-7Microcontroller.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ ทางผู้จัดทำได้รับความอนุเคราะห์ทางด้านต่าง ๆ ทั้งคำปรึกษาทางด้านวิชาการ และคำแนะนำในการปฏิบัติงาน รวมถึงความช่วยเหลือทางด้านเครื่องมือ และอุปกรณ์ต่าง ๆ จากอาจารย์ ปิติเขต สู้รักษา อาจารย์กฤดากร กล่อมการ และพี่แมว จนกระทั่งสำเร็จเป็นปริญญานิพนธ์ฉบับนี้

สุดท้ายนี้ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ ที่คอยห่วงใยและให้การสนับสนุนด้านการศึกษาเป็นอย่างดีมาโดยตลอด รวมทั้งขอบคุณเพื่อน ๆ นักศึกษาทุกคนที่ช่วยเหลือให้คำแนะนำทางด้านวิชาการ และการดำเนินงานในหลาย ๆ ด้าน

คุณประโยชน์ที่เกิดจากปริญญานิพนธ์ฉบับนี้ ผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่าน

นาย สุธี รุ่งเรืองทวีพงศ์
นาย แสงชัย หาญกล้า
ผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญรูปภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 แนวความคิดและที่มา

ในปัจจุบันความปลอดภัยของ ข้อมูลมีความสำคัญอย่างยิ่ง ข้อมูลมากมายมหาศาลเพียงใดหากไม่มีความปลอดภัยของข้อมูลแล้ว ข้อมูลนั้นก็จะไม่มีค่าเลย ซึ่งข้อมูลเหล่านั้นจำเป็นจะต้องถูกป้องกันจากการ ตรวจจับหรือการอ่านจากบุคคลภายนอก ซึ่งนี่เป็นที่มาของโครงงานนี้ จะทำการเข้ารหัสข้อมูล โดยใช้ทฤษฎีของปรากฏการเคออสติก (Chaotic) นำมาประยุกต์ใช้ในการเข้ารหัสลับข้อมูล

1.2 จุดประสงค์

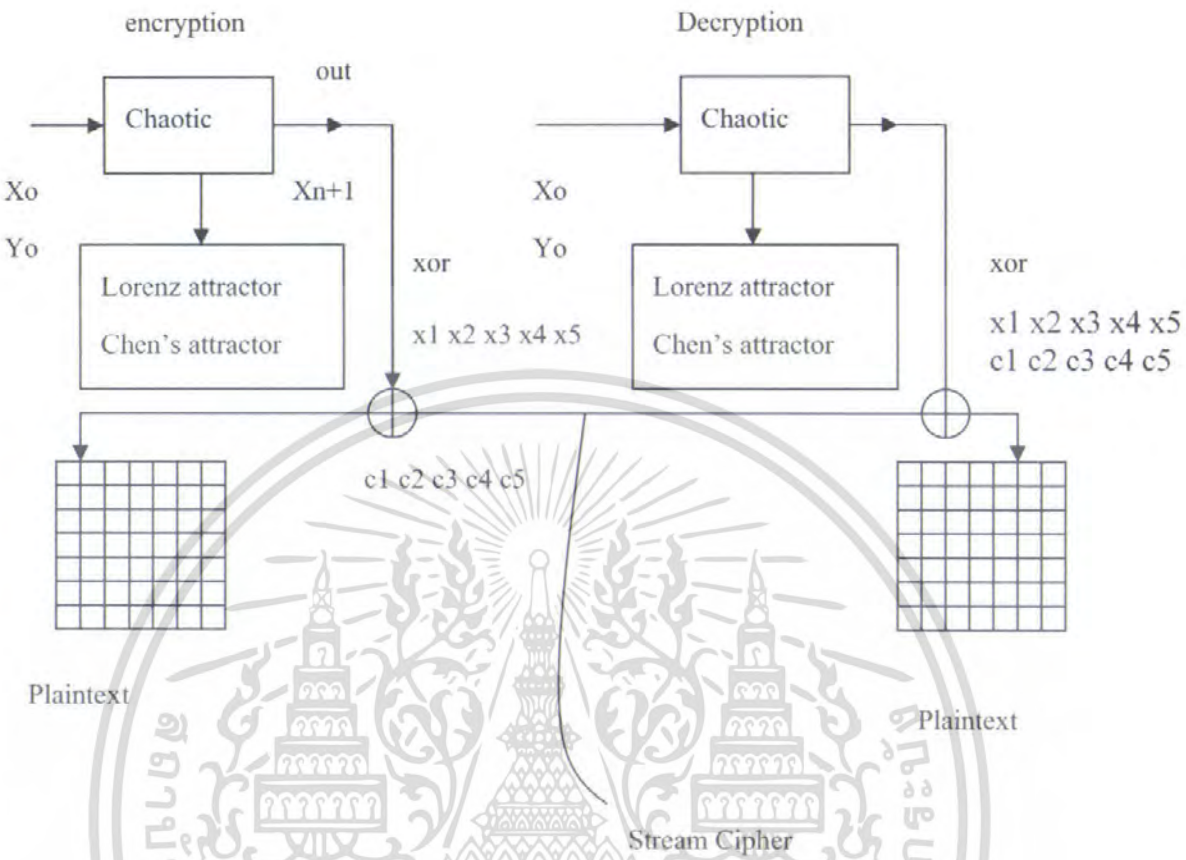
1. เพื่อศึกษาปรากฏการณ์เคออสติกต่างๆ จาก Multi chaotic map
2. ศึกษาการแก้สมการเชิงอนุพันธ์ เพื่อนำมาแก้ระบบสมการ
3. สามารถนำค่าที่ได้จาก สมการมาเป็นตัวเข้ารหัสลับได้
4. นำ ไมโคร คอนโทรเลอร์ อาร์มเซเว่น (Microcontroller ARM-7) มาสร้างรหัสลับข้อมูล และถอดรหัสลับข้อมูลจากคอมพิวเตอร์ได้

1.3 ขอบเขตของโครงงาน

1. สร้างโปรแกรมจำลองการทำงานของสมการ Chaotic ต่างๆ โดยใช้ภาษาซี
2. เปรียบเทียบความสามารถและความถูกต้องของการหาค่าสมการ Chaotic ต่างๆ โดยใช้ระเบียบวิธีทางคณิตศาสตร์หลายๆวิธี
3. นำโปรแกรมที่ได้มาทำงานบน ไมโคร คอนโทรเลอร์ อาร์มเซเว่น เพื่อเป็นอุปกรณ์เข้ารหัสและถอดรหัสลับ
4. สร้างโปรแกรมเข้ารหัสที่ทำงานบนเครื่องคอมพิวเตอร์และทำการเข้าและถอดรหัสลับบนการเชื่อมต่อข้อมูลผ่านทาง ซีเรียล พอร์ต (Serial Bus Port)

1.4 ผลที่คิดว่าจะได้รับ

1. โปรแกรมเข้ารหัสโดยใช้ เคออสติก ฟังก์ชัน หลายๆวิธี
2. สามารถสร้างข้อมูลเมื่อถูกเข้ารหัสลับแล้วยากต่อการถูกเข้าถึงโดยผู้ที่ไม่ได้รับอนุญาต
3. ระบบการเข้ารหัสที่มีรูปแบบการเข้ารหัสลับโดยใช้เคออสติกซิสเต็ม (Chaotic system)



รูปที่ 1-1 การเข้ารหัสแบบ เคออสติก (Chaotic based cryptography)

สมการอธิบายหลักการเข้ารหัสโดยสามารถอธิบายได้ในสมการ

Ex-or

0	0	0
1	0	1
0	1	1
1	1	0

$$c1 = x1 + p1$$

$$c1+x1 = x1 + p1 + x1$$

$$c1+x1 = p1 + 0 = P1$$

โดย c คือ ข้อมูลที่เข้ารหัสลับแล้วหรือ (Cipher)

x คือ รหัสที่ถูก สร้างโดย ระบบเคออสติก

p คือ ข้อมูล (plaintext)

บทที่ 2
ทฤษฎีและหลักการที่ใช้ในโครงงาน

2.1 ทฤษฎี Chaos

ทฤษฎีความอลวน (Chaos theory)

ทฤษฎีความอลวน (chaos theory)[1] เป็นทฤษฎีที่อธิบายถึง ลักษณะพฤติกรรมของระบบพลวัต (คือ ระบบที่มีการเปลี่ยนแปลง เช่น เปลี่ยนแปลงตามเวลาที่เปลี่ยนไป) โดยลักษณะการเปลี่ยนแปลงของระบบที่ เรียกว่าเคออสนี้ จะมีลักษณะที่ปั่นป่วนจนดูคล้ายว่า การเปลี่ยนแปลงนั้นเป็นแบบสุ่มหรือไร้ระเบียบ (random/stochastic) แต่จริง ๆ แล้ว ระบบเคออสนี้เป็นระบบแบบไม่สุ่ม หรือระบบที่มีระเบียบ (deterministic) ในทางคณิตศาสตร์และฟิสิกส์ คำจำกัดความของระบบเคออส คือ ระบบแบบไม่เป็นเชิงเส้น (nonlinear system) ประเภทหนึ่ง ที่มีความไวต่อสภาวะเริ่มต้น

กล่าวอีกนัยหนึ่งคือ ถ้าระบบ 2 ระบบนั้นเริ่มต้น จากสภาวะที่แตกต่างกันเพียงเล็กน้อย คือเกือบจะเหมือนกันทุกประการ เมื่อระบบได้มีการเปลี่ยนไปสักระยะหนึ่ง สภาวะของระบบทั้งสองที่เราสังเกตได้เมื่อเวลาผ่านไปจะแตกต่างกันอย่างสังเกตเห็นได้ชัด

เรามักจะได้ยินคำพูดที่เป็นที่นิยมพูดกันอย่างกว้างขวางที่ว่า "เด็ดดอกไม้สะเทือนถึงดวงดาว" หรือ "ผีเสื้อขยับปีกทำให้เกิดพายุ" (จาก "butterfly effect") ซึ่งมีคนจำนวนไม่น้อยที่ตีความในลักษณะของ ขนาดความรุนแรงของผลลัพธ์เท่านั้น ระบบเคออสนั้นไม่จำเป็นจะต้องแตกต่างกันในแง่ของขนาด ของผลลัพธ์เสมอไป แต่อาจแตกต่างกันในแง่ของ พฤติกรรม การเปลี่ยนแปลงก็ได้ จากตัวอย่างข้างต้น การ เปลี่ยนแปลงของระบบทั้งสองนั้นจะมีลักษณะที่คล้ายคลึงกันมากในขณะเริ่มต้น เมื่อเวลาผ่านไป การเปลี่ยนแปลงนั้นแทบจะเรียกได้ว่าไม่มีอะไร

จุดเริ่มต้นของทฤษฎีเคออสนี้ สามารถสืบย้อนกลับไปได้ถึงในช่วงปี พ.ศ. 2443 (ค.ศ. 1900) จากการศึกษาปัญหาวงโคจรของวัตถุสามชิ้นในสนามแรงดึงดูดระหว่างกัน ซึ่งมีชื่อเรียกเป็นทางการว่า ปัญหาสามวัตถุ (three-body problem) โดย ออองรี ปวังกาเร ซึ่งได้ค้นพบว่า วงโคจรที่ศึกษานั้นอาจจะมีลักษณะที่ไม่ได้เป็นวงรอบ (periodic) คือไม่ได้มีทางวิ่งซ้ำเป็นวงรอบ ยิ่งไปกว่านั้น วงโคจรนั้นก็ไม่ได้ขยายวงออกไปเรื่อย ๆ หรือมีลักษณะที่ลู่เข้าหาจุดใด ๆ ต่อมาได้มีการศึกษาถึงปัญหาสมการเชิงอนุพันธ์ไม่เป็นเชิงเส้นที่เกี่ยวข้อง โดยที่ เบอร์คอฟ (G.D. Birkhoff) นั้นศึกษาปัญหาสามวัตถุ คอลโมโกรอฟ ศึกษาปัญหาความปั่นป่วน (หรือ เทอร์บิวเลนซ์) และปัญหาเกี่ยวกับดาราศาสตร์. ส่วน คาร์ทไรท์ (M.L. Cartwright) และ ลิตเติลวูด (J.E. Littlewood) นั้นศึกษาปัญหาทางวิศวกรรมการสื่อสารด้วยคลื่นวิทยุ สเมล (Stephen Smale) นั้นอาจเป็นนักคณิตศาสตร์คนแรกที่ทำการศึกษาถึงปัญหาทางด้านพลศาสตร์ของระบบไม่เป็นเชิงเส้น. ถึงแม้ว่าความอลวนของเส้นทางโคจรของดาว นั้นยังไม่ได้มีการทำการสังเกตบันทึกแต่อย่างใด แต่ก็ได้มีการสังเกตพบพฤติกรรมความอลวนในความปั่นป่วนของการเคลื่อนที่ของของไหล และ ในการออสซิลเลท แบบ

ไม่เป็นวงรอบของวงจรวิทยุ ซึ่งไม่มีทฤษฎีใดในขณะนั้นสามารถอธิบายพฤติกรรมเหล่านี้ได้

ความตื่นตัวในการพัฒนาทฤษฎีความอลวนนี้ เกิดขึ้นในช่วงกลางของศตวรรษที่ 20 เมื่อเป็นที่ประจักษ์ว่า ทฤษฎีของระบบเชิงเส้นนั้นไม่สามารถใช้อธิบายพฤติกรรมบางอย่าง แม้กระทั่งพฤติกรรมของระบบที่ไม่ซับซ้อนอย่าง แมพลอจิสติก (Logistic map) อีกปัจจัยหนึ่งที่ส่งผลให้พัฒนาการของทฤษฎีความอลวนเป็นไปอย่างรวดเร็วก็คือ คอมพิวเตอร์ การคำนวณในทฤษฎีความอลวนนั้น โดยส่วนใหญ่จะมีลักษณะที่เป็นการคำนวณค่าแบบซ้ำ ๆ จากสูตรคณิตศาตร์ และสามารถใช้คอมพิวเตอร์ช่วยในการคำนวณได้อย่างมีประสิทธิภาพ

2.2 ทฤษฎีของ Lorenz

จากความก้าวหน้าของการประดิษฐ์เครื่องคอมพิวเตอร์ทำให้เกิดเหตุการณ์ที่สำคัญคือ การทดลองคำนวณดินฟ้าอากาศระยะยาวของ Edward Lorenz [2] [3][4] แห่งสถาบัน MIT เมื่อกลางทศวรรษที่ 60 ศาสตราจารย์ด้านอุตุนิยมวิทยาผู้นี้พยายามสร้างโมเดลการคำนวณในการพยากรณ์อากาศโดยใช้สมการง่าย ๆ แสดงการปฏิสัมพันธ์ระหว่างอุณหภูมิกับกระแสลม โดยศาสตราจารย์ Lorenz ได้ป้อนข้อมูลที่มีจุดทศนิยมขนาด 6 หลัก คือ 0.506127 เข้าเครื่องคอมพิวเตอร์ซึ่งพิมพ์ผลออกมาทุก ๆ นาที แต่ด้วยความที่ไม่ต้องการนั่งคอยผลลัพธ์นาน ๆ เนื่องจากคอมพิวเตอร์ยุคนั้นทำงานช้า Lorenzจึงตัดตัวเลขหลังจุดทศนิยมออกไปให้เหลือ 3 หลัก
คือ 0.506 และได้เอาผลลัพธ์จากระยะหนึ่งมาเป็นจุดเริ่มต้นของการคำนวณแล้วเริ่มคำนวณใหม่ผลของการคำนวณในระยะแรกเหมือนกันการทดลองเก่าๆ ที่เคยทำครั้งแล้วครั้งเล่า แต่หลังจากที่ผ่านไปซักพักหนึ่งแล้วกลับมาดูตัวเลขใหม่ ปรากฏว่าผลลัพธ์ต่างกัน โดยสิ้นเชิง โมเดลของดินฟ้าอากาศไปกันคนละทิศทาง เขาได้ทำการทดลองซ้ำอีกเพื่อยืนยันความถูกต้อง

ในที่สุดเขาก็สรุปว่าการลดตัวเลขขนาด 3 หลักหลังจุดทศนิยมซึ่งเป็นเงื่อนไขเบื้องต้นในการคำนวณทำให้เกิดผลลัพธ์ที่แตกต่างกันไปจากจุดเริ่มต้นจำนวนมหาศาลอย่างไม่น่าเชื่อ โดยสมการของ Lorenz เป็นดังนี้

$$\begin{aligned} \frac{dx}{dt} &= a(y - x) \\ \frac{dy}{dt} &= x(b - z) - y \\ \frac{dz}{dt} &= xy - cz \end{aligned} \qquad (2.1)$$

โดยที่ $a = 10, b = 28, c = \frac{8}{3}$

2.2.1 ตัวดึงดูดของ Lorenz

ตัวดึงดูด Lorenz คิดขึ้นโดย Edward Lorenz ในปี ค.ศ. 1963 เป็นระบบพลวัตไม่เป็นเชิงเส้น 3 มิติ โดยเป็นแบบจำลองในรูปอย่างง่ายของ การพัดพาความร้อนในบรรยากาศ ระบบนี้จะแสดงพฤติกรรมความอลวนที่ค่าพารามิเตอร์บางค่า รวมถึงลักษณะของระบบที่เรียกว่า ตัวดึงดูดประหลาด (strange attractor) ซึ่งพิสูจน์โดย ทัคเกอร์ (W. Tucker) ในปี ค.ศ. 2001 ตัวดึงดูดประหลาดในที่นี้เป็น เฟร็กทัล ที่มีค่ามิติ Hausdorff อยู่ระหว่าง 2 ถึง 3 กราสเบอร์เกอร์ (Grassberger) ได้ประมาณค่ามิติฮอสดอร์ฟว่ามีค่าประมาณ 2.06 ± 0.01 และ ค่ามิติโครีเลชัน (correlation dimension) ประมาณ 2.05 ± 0.01

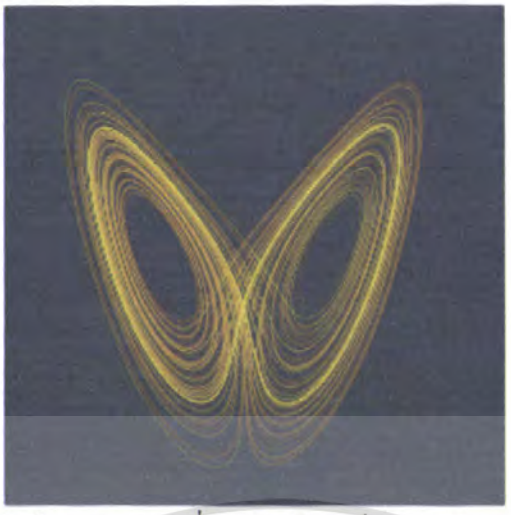
ระบบที่มีปรากฏพฤติกรรมตามแบบจำลอง Lorenzนี้คือ เลเซอร์, ไดนาโม และกังหันน้ำบางชนิด

ตัวดึงดูด Lorenz เขียนในรูปสมการทางคณิตศาสตร์ได้ดังต่อไปนี้:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dy} &= x(r - z) - y \\ \frac{dz}{dy} &= xy - bz \end{aligned} \qquad (2.2)$$

โดยที่ σ เรียกว่า ตัวเลข Prandtl และ r เรียกว่า ตัวเลข Reynolds $\sigma, r, b > 0$ แต่ปกติแล้วจะมีค่า $\sigma = 10$, $b = 8/3$ และ r เป็นค่าที่ปรับได้ ระบบจะแสดงพฤติกรรมความอลวนที่ค่า $r = 28$ แต่แสดงพฤติกรรมโคจรพันกันเป็นวงรอบ ที่ค่า r อื่นๆ ตัวอย่างเช่น ที่ค่า $r = 99.96$ วงโคจรจะเป็นรูป $T(3,2)$ เงื่อนทอรัส(torus knot)

รูปร่างของตัวดึงดูด Lorenzที่คล้ายผีเสื้อนี้ เป็นส่วนหนึ่งของจุดกำเนิดของคำ butterfly effect ในทฤษฎีความอลวน



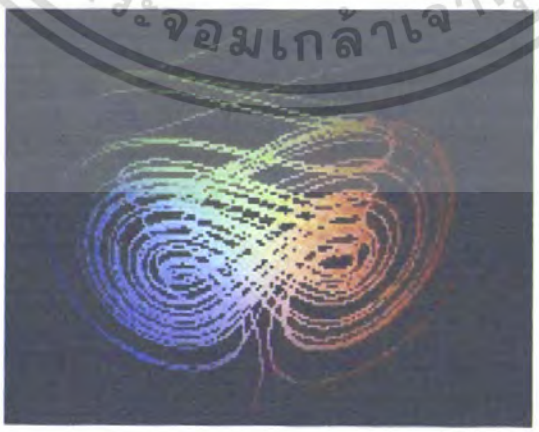
รูปที่ 2-1 ตัวดึงดูด Lorenz ที่ค่าพารามิเตอร์ r=28, σ = 10, b = 8/3

2.3 สมการของ Chen

ต่อมาได้มีผู้ที่ต้องการจะควบคุมการเกิดสัญญาณเคออสติก (Chaotic Signal) ที่เกิดจากสมการของ Lorenz ดังนั้นจึงได้มีการคิดค้น โดยใช้สมการของ Lorenz เป็นพื้นฐาน ซึ่งเรียกว่าสมการของ Chen และผู้ค้นคือ Tetsushi Ueta และ Guanrong Chen [5] [6] [7] ซึ่งมีรูปแบบสมการดังนี้

$$\begin{aligned} \frac{dx}{dt} &= a(y - x) \\ \frac{dy}{dt} &= (c - a)x - xz + cy \\ \frac{dz}{dt} &= xy - bz \end{aligned} \qquad (2.3)$$

โดย a = 35 , b= 3 , c= 28



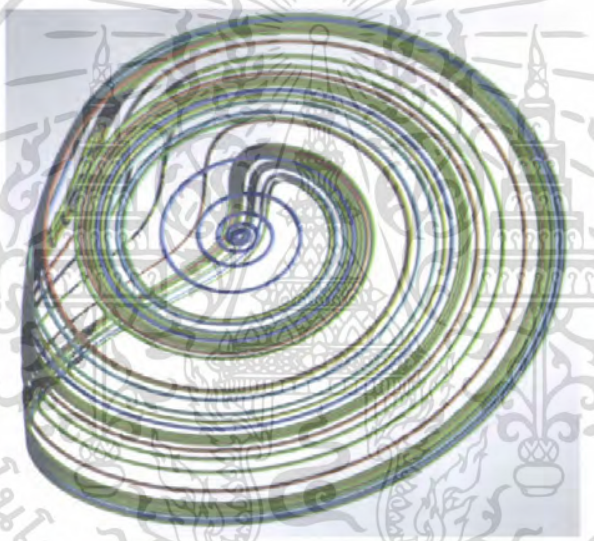
รูปที่ 2-2 ตัวดึงของ Chen

2.4 สมการของ Rossler

สมการ Rossler[8][9][10] จะมีรูปแบบ ตัวดึงดูดอีกรูปแบบหนึ่งซึ่งจะอธิบายปรากฏการณ์เคออส โดยมีสมการดังนี้

$$\begin{aligned}\frac{dx}{dt} &= -y - z \\ \frac{dy}{dt} &= x + ay \\ \frac{dz}{dt} &= b + z(x - c)\end{aligned} \tag{2.4}$$

โดย a = 0.2, b = 0.2, c = 5.7



รูปที่ 2-3 ตัวดึงดูดของ Rossler

2.5 Logistic map

Logistic map เป็น แม็ปฟังก์ชัน ที่มี โดเมน และ เรนจ์ อยู่ในปริภูมิเดียวกัน) พหุนาม นิยมใช้เป็นตัวอย่างของระบบพลวัตไม่เป็นเชิงเส้นอย่างง่ายที่สามารถแสดงพฤติกรรมความอลวนได้ Logistic map นี้เริ่มเป็นที่รู้จักกว้างขวางจากผลงานตีพิมพ์ของนักชีววิทยา Robert May แรกเริ่มนั้น Logistic map นี้ถูกสร้างขึ้นโดย Pierre François Verhulst เพื่อเป็นแบบจำลองการกระจายปริมาณประชากรมนุษย์ ต่อมาถูกนำไปใช้สำหรับ การเพิ่มปริมาณประชากรของสปีชีส์อื่นๆ ภายใต้สภาวะแวดล้อมจำกัด เช่น อาหาร, โรค, และ อื่นๆ ซึ่งแบบจำลองจะมีพฤติกรรมจากผลของ

1. การสืบพันธุ์ คือ จำนวนประชากร จะเพิ่มขึ้นด้วยอัตราที่แปรผันตามจำนวนประชากรในขณะนั้น

2. การขาดอาหาร คือ จำนวนประชากรจะลดลงด้วยอัตรา ที่แปรผันตาม จำนวนประชากรที่สภาพแวดล้อมนั้นสามารถรองรับได้ในทางทฤษฎี ลบออกด้วยค่าจำนวนประชากรในขณะนั้น

ซึ่งสามารถเขียนในรูปสมการทางคณิตศาสตร์ดังนี้

$$x_{n+1} = rx_n(1 - x_n) \qquad (2.5)$$

โดยที่

$x_n \in [0,1]$, ใช้หมายถึงปริมาณประชากร ที่ปี n, และ x_0 หมายถึงปริมาณประชากรเริ่มต้น (ที่ปี 0)

r เป็นจำนวนบวก ใช้หมายถึง ค่าผลรวมของอัตรา การสืบพันธุ์ และ การขาดอาหาร

รูปที่ 2-4 Logistic map ที่ r=1, 2, 4

2.5.1 พฤติกรรมตามค่า *r*

พฤติกรรมของระบบ ที่ค่าพารามิเตอร์ *R* ต่างๆ

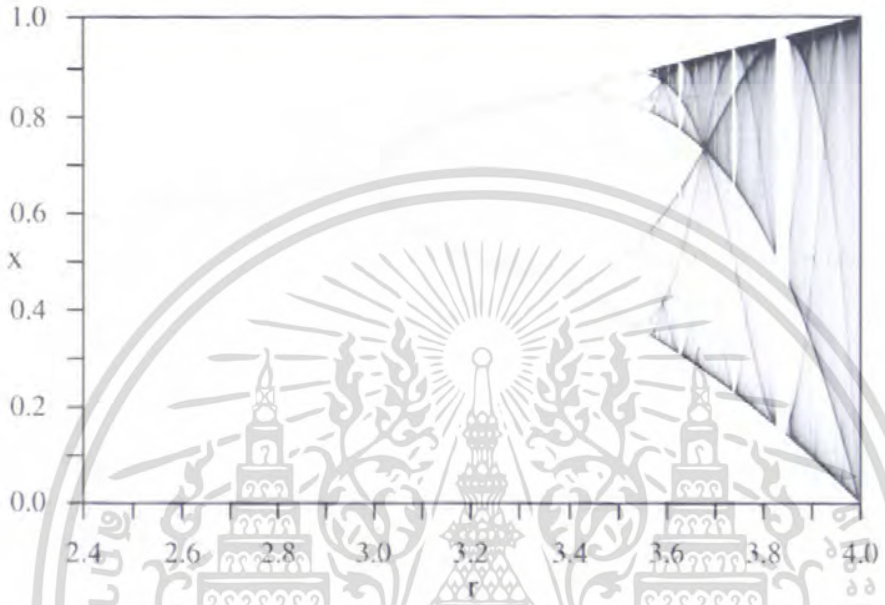
- $0 \le r \le 1$, ประชากรจะตายไปจนหมดโดยไม่ขึ้นกับค่าเริ่มต้น โดยระบบมีจุดตายตัว (fixed point) เพียงจุดเดียวที่ x = 0 ซึ่งเป็นจุดตายตัวแบบดึงดูด(attracting fixed point) หรือเรียกว่า "จุดดูดซับ" (sink) และดึงดูดค่าเริ่มต้นทุกค่าใน [0,1]
- $1 \le r \le 3$, ระบบมีจุดตายตัว 2 จุดคือที่ x = 0 และ x = (r −1) / r โดยที่ x = 0 เป็นจุดตายตัวแบบผลักออก(repelling fixed point) หรือ เรียกว่า "จุดกำเนิด" (source) และ x = (r − 1) / r เป็น จุดตายตัวแบบดึงดูด

  -ที่ค่า r อยู่ระหว่าง 1 ถึง 2 จำนวนประชากรจะลู่เข้าหาค่า (r -1) / r และคงตัวอย่างรวดเร็ว

  - ที่ค่า r อยู่ระหว่าง 2 และ 3 จำนวนประชากรจะเริ่มแกว่งก่อนลู่เข้าหาจุดดูดซับ โดยมีอัตราการลู่เข้าเป็นเชิงเส้น

  - ที่ค่า *r* เท่ากับ 3 อัตราการลู่เข้าจะช้ากว่าอัตราที่เป็นเชิงเส้น
- $3 \le r \le 1+\sqrt{6}$ (ประมาณ 3.45) จำนวนประชากรจะมีค่าแกว่งสลับระหว่างค่า 2 ค่า ซึ่ง 2 ค่านี้ขึ้นกับค่า *r* แต่ไม่ขึ้นกับค่าเริ่มต้น ซึ่งก็คือ ระบบมีจุดวงรอบคาบ 2 แบบดึงดูด หรือ จุดดูดซับแบบวงรอบคาบ2
- $1+\sqrt{6} \le r \le 3.54$ (โดยประมาณ) จำนวนประชากรจะมีค่าแกว่งสลับระหว่างค่า 4 ค่า ไม่ขึ้นกับคู่เริ่มต้น ซึ่งก็คือ ระบบมีจุดดูดซับแบบวงรอบคาบ 4
- เมื่อค่า *r* มีค่าเพิ่มมากกว่า 3.54 จำนวนประชากรจะมีค่าแกว่งสลับ เป็นวงรอบด้วยคาบ 8,16,32 และเพิ่มขึ้นเรื่อยๆ ช่วงการเพิ่มของค่า *r* ที่ส่งผลให้คาบวงรอบการแกว่งเพิ่มขึ้นจะลดลงอย่างรวดเร็ว สัดส่วนของระยะของค่าพารามิเตอร์ที่ทำให้มีการเพิ่มคาบ (หรือเรียกช่วงระยะไบเฟอร์เคชัน) ที่อยู่ถัดกัน จะลู่เข้าหา ค่าคงที่ Feigenbaum constant δ = 4.669 ซึ่งพฤติกรรมดังกล่าวนี้จะไม่ขึ้นกับค่าเริ่มต้นเลยอย่างใด
- ที่ค่า *r*= 3.57 (โดยประมาณ) เป็นจุดที่ระบบเริ่มมีพฤติกรรมความอลวน ระบบจะไม่มีพฤติกรรมการแกว่งเป็นวงรอบดังค่า *r* ที่ผ่านมา แต่ระบบจะมีพฤติกรรมที่ไวต่อค่าเริ่มต้น ซึ่งเป็นคุณลักษณะของความอลวน ความแตกต่างเพียงเล็กน้อยของค่าจำนวนประชากรเริ่มต้น จะมีผลต่อการเปลี่ยนแปลงของค่าประชากรในระยะยาว
- ค่า *r* ระหว่าง 3.57 และ 4 มีหลายค่าที่ระบบมีพฤติกรรมความอลวน แต่ก็ยังมีค่า *r* บางค่าที่ระบบไม่แสดงพฤติกรรมความอลวน ตัวอย่างเช่น ที่ *r* ประมาณ 3.82 จะมีบางช่วงที่ระบบมีพฤติกรรมแกว่งเป็นวงรอบคาบ 3 และที่ค่า *r* สูงขึ้นเล็กน้อยจะแกว่งเป็นวงรอบคาบ 6, 12 และเพิ่มขึ้นตามลำดับ และจะมีบางช่วงที่มีการแกว่งเป็นคาบ 5 และอื่นๆ ซึ่งพฤติกรรมทั้งหมดนี้จะมีการแกว่งเป็นวงรอบ และไม่ขึ้นกับค่าเริ่มต้น
- ที่ค่า $r = 4$ และมากกว่านั้น ค่าของระบบจะลู่ออก สำหรับทุกค่าเริ่มต้นใน [0,1]

Bifurcation diagram ดังรูป แสดงให้เห็นถึงพฤติกรรมดังกล่าวข้างต้นนี้ โดยที่แกนนอนของแผนผังเป็น ค่า r และ แกนตั้งเป็นค่าจำนวนประชากร หรือค่าของระบบในระยะยาว
แผนผังไบเฟอร์เคชันนี้เป็น แฟร็กทัล ถ้าเราพิจารณาที่ค่า $r = 3.82$ ที่ระบบมีพฤติกรรมแกว่งเป็นวงรอบคาบ 3 เมื่อเราเลือกกิ่งหนึ่งใน 3 และขยายที่กิ่งนั้นเราจะเห็นภาพที่มีลักษณะเหมือนภาพเดิม

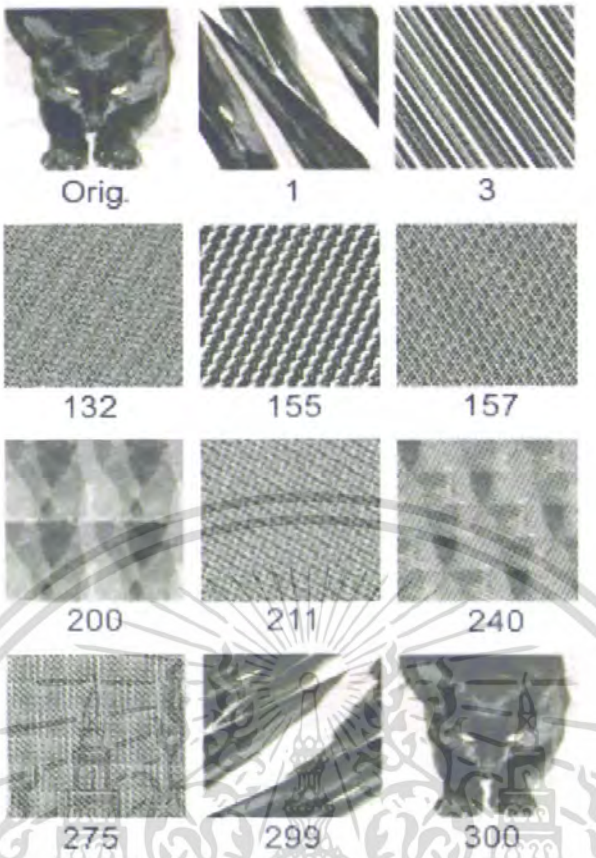


รูปที่ 2-5 Bifurcation diagram ของ Logistic map

2.6 Cat map

ในทางคณิตศาสตร์ Cat map [5] (Arnold's cat map) เป็น เคออสติกแม็ป (chaotic map) แบบหนึ่ง ซึ่งชื่อนี้ตั้งตาม Vladimir Arnold ผู้ซึ่งพิสูจน์ผลกระทบของ Cat map ในปี ค.ศ.1960 โดยใช้รูปของแมว

หนึ่งในลักษณะของ Cat map คือ ภาพจะดูเหมือนว่ามีเปลี่ยนแปลงรูปร่างแบบสุ่ม แต่สามารถกลับสู่ภาพเดิมได้ภายหลัง ดังในรูป 2-6 ภาพแมวดั้งเดิมถูกตัด จากนั้นถูกห่อรวมกัน ในการทำซ้ำครั้งแรกของการเปลี่ยนแปลงรูปร่าง หลังจากนั้นการทำซ้ำครั้งอื่น จะมีผลคือรูปที่ปรากฏจะดูเหมือนสุ่มหรือไม่มีลำดับขั้นตอน และท้ายที่สุดการทำซ้ำจะวนภาพจนกลับมาเป็นรูปแมวดังเดิม



รูปที่ 2-6 การแปลงค่าวิธี Cat map

สมการ Cat map

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \pmod{1} \tag{2.6}$$

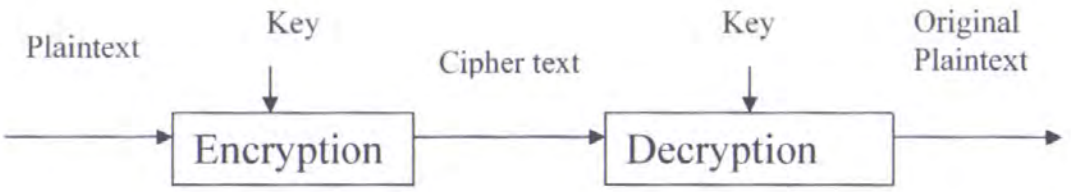
2.7 การเข้ารหัสข้อมูล [1](Cryptography)

การเข้ารหัสข้อมูลโดยพื้นฐานแล้วจะเกี่ยวข้องกับวิธีการทางคณิตศาสตร์เพื่อใช้ในการป้องกันข้อมูลหรือข้อความตั้งต้นที่ต้องการส่งไปถึงผู้รับ ข้อมูลตั้งต้นจะถูกแปรเปลี่ยนไปสู่ข้อมูลหรือข้อความอีกรูปแบบหนึ่งที่ไม่สามารถอ่านเข้าใจได้โดยใครก็ตามที่ไม่มีกุญแจสำหรับเปิดดูข้อมูลนั้น เราเรียกกระบวนการในการแปรรูปของข้อมูลตั้งต้นว่า "การเข้ารหัสลับข้อมูล" (Encryption) และกระบวนการในการแปลงข้อความที่ไม่สามารถอ่าน และทำความเข้าใจให้กลับไปสู่ข้อความตั้งเดิม ว่าการถอดรหัสลับข้อมูล (Decryption) [11][12]

---

1 จากที่มา http://www.thaicert.nectec.or.th/paper/encryption/intro_crypt.php

เพื่อให้เกิดความสะดวกสำหรับผู้อ่านในการทบทวนพื้นฐานจึงอ้างถึง บทความต้นฉบับดังกล่าว ณ ที่นี้อีกครั้ง



รูปที่ 2-7 การเข้ารหัสลับและถอดรหัสลับข้อมูล

2.7.1 อัลกอริทึมในการเข้ารหัสข้อมูล

อัลกอริทึมในการเข้ารหัสข้อมูลมี 2 ประเภทหลัก คือ

2.7.1.1 อัลกอริทึมแบบสมมาตร (Symmetric key algorithms)

อัลกอริทึมแบบนี้จะใช้กุญแจที่เรียกว่า กุญแจลับ (Secret key) ซึ่งมีเพียงหนึ่งเดียวเพื่อใช้ในการเข้าและถอดรหัสข้อความที่ส่งไป อัลกอริทึมยังสามารถแบ่งย่อยออกเป็น 2 ประเภท ได้แก่ แบบบล็อค (Block Algorithms) ซึ่งจะทำการเข้ารหัสทีละบล็อค (1 บล็อคประกอบด้วยหลายไบต์ เช่น 64 ไบต์ เป็นต้น) และแบบสตรีม (Stream Algorithms) ซึ่งจะทำการเข้ารหัสทีละไบต์อัลกอริทึมแบบนี้จะใช้กุญแจที่เรียกว่า กุญแจลับ (Secret key) ซึ่งมีเพียงหนึ่งเดียวเพื่อใช้ในการเข้าและถอดรหัสข้อความที่ส่งไป อัลกอริทึมยังสามารถแบ่งย่อยออกเป็น 2 ประเภท ได้แก่ แบบบล็อค (Block Algorithms) ซึ่งจะทำการเข้ารหัสทีละบล็อค (1 บล็อคประกอบด้วยหลายไบต์ เช่น 64 ไบต์ เป็นต้น) และแบบสตรีม (Stream Algorithms) ซึ่งจะทำการเข้ารหัสทีละไบต์

การเข้ารหัสข้อมูลและการถอดรหัสข้อมูล ด้วยอัลกอริทึมแบบสมมาตร สามารถแสดงด้วยสมการ

$$\mathbf{Ek(M) = C}$$

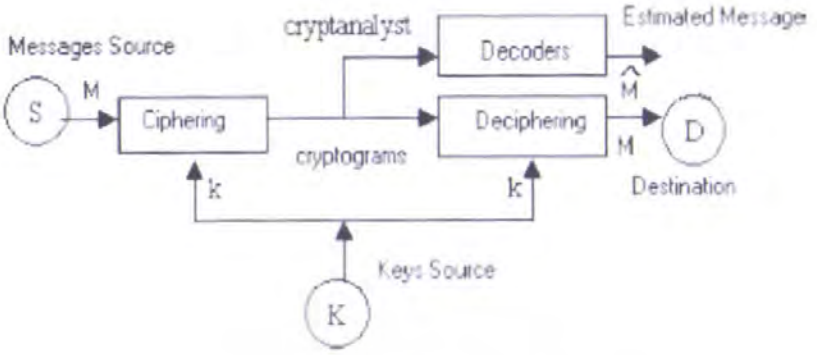
$$\mathbf{Dk(C) = M}$$

โดยที่ M คือ ข้อมูลต้นแบบ (Plaintext)

C คือ ข้อมูลที่ถูกเข้ารหัสแล้ว (Cipher text)

Ek คือการเข้ารหัสข้อมูล

Dk คือการถอดรหัสข้อมูล



รูปที่ 2-8 ระบบของการเข้ารหัสข้อมูล (Ciphering system)

2.7.1.2 อัลกอริทึมแบบอสมมาตร (Asymmetric key algorithms)

อัลกอริทึมนี้จะใช้กุญแจสองตัวเพื่อทำงาน ตัวหนึ่งใช้ในการเข้ารหัสและอีกตัวหนึ่งใช้ในการถอดรหัสข้อมูลที่เข้ารหัสมาโดยกุญแจตัวแรก อัลกอริทึมกลุ่มสำคัญในแบบอสมมาตรนี้คือ อัลกอริทึมแบบกุญแจสาธารณะ (Public keys Algorithms) ซึ่งใช้กุญแจที่เรียกกันว่า กุญแจสาธารณะ (Public keys) ในการเข้ารหัสและใช้กุญแจที่เรียกกันว่า กุญแจส่วนตัว (Private keys) ในการถอดรหัสข้อมูลนั้น กุญแจสาธารณะนี้สามารถส่งมอบให้กับผู้อื่นได้ เช่น เพื่อนร่วมงานที่เราต้องการติดต่อด้วย หรือแม้กระทั่งวางไว้บนเว็บไซต์เพื่อให้ผู้อื่นสามารถดาวน์โหลดไปใช้งานได้ สำหรับกุญแจส่วนตัวนั้นต้องเก็บไว้กับผู้เป็นเจ้าของกุญแจส่วนตัวเท่านั้นและห้ามเปิดเผยให้ผู้อื่นทราบโดยเด็ดขาด

อัลกอริทึมแบบกุญแจสาธารณะยังสามารถประยุกต์ใช้ได้กับการลงลายมือชื่ออิเล็กทรอนิกส์ (ซึ่งเปรียบเสมือนการลงลายมือชื่อของเราที่ใช้กับเอกสารสำนักงานทั่วไป) การลงลายมือชื่อนี้จะเป็นการพิสูจน์ความเป็นเจ้าของและสามารถใช้ได้กับการทำธุรกรรมต่างๆ บนอินเทอร์เน็ต เช่น การซื้อสินค้า เป็นต้น วิธีการใช้งานคือ ผู้เป็นเจ้าของกุญแจส่วนตัวลงลายมือชื่อของตนกับข้อความที่ต้องการส่งไปด้วยกุญแจส่วนตัว แล้วจึงส่งข้อความนั้นไปให้กับผู้รับ เมื่อได้รับข้อความที่ลงลายมือชื่อมา ผู้รับสามารถใช้กุญแจสาธารณะ (ที่เป็นคู่ของกุญแจส่วนตัวนั้น) เพื่อตรวจสอบว่าเป็นข้อความที่มาจากผู้ส่งนั้นหรือไม่

2.7.2 ปัญหาของอัลกอริทึมแบบสมมาตร

อัลกอริทึมแบบสมมาตรมีความสำคัญไม่ด้อยไปกว่าอัลกอริทึมแบบอสมมาตร ทั้งนี้เนื่องจากอัลกอริทึมแบบแรกทำงานได้รวดเร็วกว่าและง่ายต่อการใช้งานกว่าแบบหลัง อย่างไรก็

ตามอัลกอริทึมแบบสมมาตรยังมีปัญหาที่สำคัญ 3 ประการ ซึ่งเป็นข้อจำกัดในการใช้งานอัลกอริทึมนี้

1. ในการใช้งานอัลกอริทึมนี้ สองกลุ่มที่ต้องการแลกเปลี่ยนข้อมูลกัน (เช่น องค์กร ก และ ข) จำเป็นต้องแลกเปลี่ยนกุญแจลับกันก่อน (ซึ่งอาจหมายถึงส่งมอบกุญแจลับให้กับอีกกลุ่มหนึ่ง) การแลกเปลี่ยนกุญแจนั้นอาจทำได้อย่างยุ่งยากและไม่สะดวก
2. ทั้งสองกลุ่มต้องรักษากุญแจลับนั้นไว้เป็นอย่างดี ห้ามเปิดเผยให้ผู้อื่นล่วงรู้โดยเด็ดขาด การที่กุญแจถูกเปิดเผยออกไปสู่ผู้อื่น (จะโดยกลุ่มใดกลุ่มหนึ่งก็ตาม) และอีกกลุ่มหนึ่งไม่ได้รับทราบปัญหานี้ อาจก่อให้เกิดปัญหากับกลุ่มที่ไม่ทราบนี้ได้ เช่น กลุ่มนี้อาจส่งข้อความที่เป็นความลับไปให้กับอีกกลุ่มหนึ่ง แต่ข้อความนี้อาจถูกเปิดเผยได้โดยใช้กุญแจลับที่ล่วงรู้โดยผู้อื่น
3. สำหรับสองกลุ่มที่ต้องการติดต่อกัน จำเป็นต้องใช้กุญแจลับเป็นจำนวน 1 กุญแจเพื่อติดต่อกัน สมมติว่ามีผู้ที่ต้องติดต่อกันเป็นจำนวน n กลุ่ม จำนวนกุญแจลับทั้งหมดที่ต้องแลกเปลี่ยนกันคิดเป็นจำนวนทั้งหมด C_2^n หรือเท่ากับ $n(n-1)/2$ กุญแจ ซึ่งจะเห็นได้ว่าจำนวนกุญแจมีมากมายเกินไป ซึ่งอาจก่อให้เกิดปัญหาด้านการรักษาความปลอดภัยให้กับกุญแจเหล่านี้

อัลกอริทึมแบบกุญแจสาธารณะ (ซึ่งเป็นแบบอสมมาตร) ช่วยแก้ปัญหาเหล่านี้ได้ทั้งหมด ผู้ใช้ที่ถือกุญแจส่วนตัวและต้องการให้บุคคลอื่นที่ตนติดต่อด้วยส่งเอกสารหรือข้อความที่เข้ารหัสมาหาตน สามารถเผยแพร่กุญแจสาธารณะของตนไว้บนเว็บไซต์หรือในที่สาธารณะซึ่งผู้อื่นสามารถเข้ามาดาวน์โหลดไปใช้งานได้ วิธีการใช้งานคือให้บุคคลอื่นที่มาดาวน์โหลดกุญแจไปนั้นทำการเข้ารหัสข้อความที่ต้องการส่งด้วยกุญแจสาธารณะ แล้วจึงส่งข้อความที่เข้ารหัสไปให้กับผู้เป็นเจ้าของกุญแจสาธารณะ โดยวิธีนี้จะไม่มีผู้อื่นสามารถเปิดดูข้อความที่เข้ารหัสนั้นได้ยกเว้นผู้ที่ถือกุญแจส่วนตัว (ที่เป็นคู่ของกุญแจสาธารณะนั้น) จึงจะสามารถเปิดข้อความนี้ดูได้

การเผยแพร่กุญแจสาธารณะในสถานที่ต่างๆ ได้ทำให้ลดความยุ่งยากในการแลกเปลี่ยนกุญแจกัน ซึ่งเป็นปัญหาข้อแรกของการเข้ารหัสแบบสมมาตร สำหรับปัญหาที่ว่าทั้งสองกลุ่มจะต้องรักษากุญแจลับไว้เป็นอย่างดีนั้น วิธีการของกุญแจสาธารณะจะทำให้ผู้ที่ต้องรับผิดชอบเหลือเพียงผู้เดียว กล่าวคือ ผู้ถือกุญแจส่วนตัว ซึ่งห้ามให้ผู้อื่นล่วงรู้โดยเด็ดขาด

สำหรับปัญหาที่สามที่ว่าจำนวนกุญแจลับที่จำเป็นต้องใช้มีมากมายเกินไป วิธีการของกุญแจสาธารณะจะใช้จำนวนกุญแจที่ประหยัดกว่า เนื่องจากกุญแจสาธารณะ 1 กุญแจของกลุ่มๆ หนึ่งจะสามารถเผยแพร่ให้กับกี่กลุ่มก็ได้ที่เราต้องการติดต่อด้วย (แทนที่จะเป็น 1 กุญแจลับต่อสองกลุ่มที่ต้องการติดต่อกัน) ดังนั้นถ้ามีกลุ่มที่ต้องติดต่อกันจำนวน n กลุ่ม จำนวนกุญแจส่วนตัวที่ต้องระวังรักษาก็คือ n กุญแจ ซึ่งจะเห็นได้ว่าลดลงไปได้เป็นจำนวนมาก

ข้อเสียที่สำคัญของระบบกุญแจสาธารณะที่สำคัญคือ ต้องใช้เวลาในการคำนวณการเข้าและถอดรหัส เมื่อเทียบกับระบบกุญแจสมมาตร และอาจใช้เวลาเป็นพันเท่าของเวลาที่ใช้โดยระบบกุญแจสมมาตร

2.7.3ความแข็งแกร่งของอัลกอริทึมสำหรับการเข้ารหัส

ความแข็งแกร่งของอัลกอริทึมหมายถึงความยากในการที่ผู้บุกรุกจะสามารถถอดรหัสข้อมูลได้โดยปราศจากกุญแจที่ใช้ในการเข้ารหัส ซึ่งจะขึ้นอยู่กับปัจจัยดังนี้

1. การเก็บกุญแจเข้ารหัสไว้อย่างเป็นความลับ ผู้เป็นเจ้าของกุญแจลับหรือส่วนตัวต้องระมัดระวังไม่ให้กุญแจสูญหายหรือล่วงรู้โดยผู้อื่น
2. ความยาวของกุญแจเข้ารหัส ปกติกุญแจเข้ารหัสจะมีความยาวเป็นบิต ยิ่งจำนวนบิตของกุญแจยิ่งมาก ยิ่งทำให้การเดาเพื่อสุ่มหากุญแจที่ถูกต้องเป็นไปได้ยากยิ่งขึ้น (เช่น กุญแจขนาด 1 บิต จะสามารถแทนตัวเลขได้ 2 ค่าคือ 0 กับ 1 กุญแจขนาด 2 บิต จะเป็นไปได้ 4 ค่าคือ 0, 1, 2, 3 เป็นต้น)
3. ความไม่เกรงกลัวต่อการศึกษาหรือดูอัลกอริทึมเพื่อหารูปแบบของการเข้ารหัส อัลกอริทึมที่ดีต้องเปิดให้ผู้รู้ทำการศึกษาในรายละเอียดได้โดยไม่เกรงว่าผู้ศึกษาจะสามารถจับรูปแบบของการเข้ารหัสได้
4. การมีประตูลับในอัลกอริทึม อัลกอริทึมที่ดีต้องไม่แฝงไว้ด้วยประตูลับที่สามารถใช้เป็นทางเข้าไปสู่อัลกอริทึม แล้วอาจใช้เพื่อทำการถอดรหัสข้อมูลได้ ประตูลับนี้ทำให้ไม่จำเป็นต้องใช้กุญแจในการถอดรหัส
5. ความไม่เกรงกลัวต่อปัญหาการหาความสัมพันธ์ในข้อมูลที่ได้รับ กล่าวคือเมื่อผู้บุกรุกทราบข้อมูลบางอย่างที่เป็นข้อมูลตั้งต้นซึ่งยังไม่ได้เข้ารหัส รวมทั้งมีข้อมูลที่เข้ารหัสแล้ว (ของข้อมูลตั้งต้นนั้น) ผู้บุกรุกอาจจะสามารถหาความสัมพันธ์ระหว่างข้อความทั้งสองนั้นได้ ซึ่งจะเป็นวิธีการในการถอดรหัสข้อมูลได้ ปัญหานี้เรียกกันว่า Known plaintext attack (คำว่า plaintext หมายถึงข้อความตั้งต้นที่ยังไม่ได้ผ่านการเข้ารหัส)
6. คุณสมบัติของข้อความตั้งต้น คุณสมบัตินี้อาจใช้เป็นช่องทางในการถอดรหัสข้อมูลได้ อัลกอริทึมที่ดีต้องไม่ใช้คุณสมบัติของข้อความเป็นกลไกในการเข้ารหัสข้อมูล

คำแนะนำในการเลือกใช้อัลกอริทึมคือให้ใช้อัลกอริทึมที่ได้มีการใช้งานมาเป็นระยะเวลานานแล้ว ทั้งนี้เนื่องจากหากปัญหาของอัลกอริทึมนี้มีจริง ก็คงเกิดขึ้นมานานแล้วและก็คงเป็นที่ทราบกันแล้ว นั่นคืออย่างน้อยที่สุดจวบจนกระทั่งถึงปัจจุบัน ก็ยังไม่มีการบุกรุกที่ทำให้อัลกอริทึมนั้นไม่สามารถใช้งานได้อย่างปลอดภัยเป็นที่ประจักษ์ ดังนั้นจึงไม่ควรใช้อัลกอริทึมใหม่ๆ ที่เพิ่งได้มีการนำเสนอกันสู่สาธารณะ เพราะอาจมีช่องโหว่แฝงอยู่และยังไม่เป็นที่ทราบในขณะนี้

2.7.4ความยาวของกุญแจที่ใช้ในการเข้ารหัส

ความยาวของกุญแจเข้ารหัสมีหน่วยนับเป็นบิต หนึ่งบิตในคอมพิวเตอร์เป็นตัวเลขฐานสองที่ประกอบด้วยค่า 0 และ 1 กุญแจที่มีความยาว 1 บิต ตัวเลขที่เป็นไปได้เพื่อแทนกุญแจนั้น จึงอาจมีค่าเป็น 0 หรือ 1 กุญแจที่มีความยาว 2 บิต ตัวเลขที่เป็นไปได้จึงเป็น 0, 1, 2 และ 3 ตามลำดับ กุญแจที่มีความยาว 3 บิต ตัวเลขที่เป็นไปได้จะอยู่ระหว่าง 0 ถึง 7 ดังนั้นเมื่อเพิ่มความยาวของกุญแจทุกๆ 1 บิต ค่าที่เป็นไปได้ของกุญแจจะเพิ่มขึ้นเป็นสองเท่าตัว หรือจำนวนกุญแจที่เป็นไปได้จะเพิ่มขึ้นเป็น 2 เท่าตัวนั่นเอง

ฉะนั้นจะเห็นได้ว่ากุญแจยิ่งมีความยาวมาก โอกาสที่ผู้บุกรุกจะสามารถคาดเดากุญแจที่ตรงกับหมายเลขที่ถูกต้องของกุญแจจะยิ่งยากมากขึ้นตามลำดับ ในการที่ผู้บุกรุกลองผิดลองถูกกับกุญแจโดยใช้กุญแจที่มีหมายเลขต่างๆ กัน เพื่อหวังที่จะพบกุญแจที่ถูกต้องและสามารถใช้ถอดรหัสข้อมูลได้ การลองผิดลองถูกนี้เราเรียกกันว่า Key search หรือการค้นหากุญแจนั่นเอง ทฤษฎีได้กล่าวไว้ว่าการลองผิดลองถูกนี้โดยเฉลี่ยจะต้องทดลองกับกุญแจเป็นจำนวนครึ่งหนึ่งของกุญแจทั้งหมดก่อนที่จะพบกุญแจที่ถูกต้อง

ความยาวของกุญแจที่มีขนาดเหมาะสมจึงขึ้นอยู่กับความเร็วในการค้นหากุญแจของผู้บุกรุกและระยะเวลาที่ต้องการให้ข้อมูลมีความปลอดภัย ตัวอย่างเช่น ถ้าผู้บุกรุกสามารถลองผิดลองถูกกับกุญแจเป็นจำนวน 10 กุญแจภายในหนึ่งวินาทีแล้ว กุญแจที่มีความยาว 40 บิต จะสามารถป้องกันข้อมูลไว้ได้ 3.484 ปี ถ้าผู้บุกรุกสามารถลองได้เป็นจำนวน 1 ล้านกุญแจในหนึ่งวินาที (เทคโนโลยีปัจจุบันสามารถทำได้) กุญแจที่มีความยาว 40 บิตจะสามารถป้องกันข้อมูลไว้ได้เพียง 13 วันเท่านั้น (ซึ่งอาจไม่เพียงพอสำหรับในบางลักษณะงาน) ด้วยเทคโนโลยีในปัจจุบันหากผู้บุกรุกสามารถทดลองได้เป็นจำนวน 1,000 ล้านกุญแจในหนึ่งวินาที กุญแจขนาด 128 บิตจะสามารถป้องกันข้อมูลไว้ได้ 1022 ปี ดังนั้นด้วยลักษณะงานทั่วไปกุญแจขนาด 128 บิตจะพอเพียงต่อการรักษาความลับของข้อมูลเอาไว้ได้

2.7.5อัลกอริทึมสำหรับการเข้ารหัสแบบสมมาตร

2.7.5.1 อัลกอริทึม DES

DES ย่อมาจาก Data Encryption Standard

อัลกอริทึมนี้ ได้รับการรับรองโดยรัฐบาลสหรัฐอเมริกาในปี ค.ศ. 1977 ให้เป็นมาตรฐานการเข้ารหัสข้อมูลสำหรับหน่วยงานของรัฐทั้ง หมด ในปี 1981อัลกอริทึมยังได้รับการกำหนดให้เป็นมาตรฐานการเข้ารหัสข้อมูลในระดับนานาชาติตามมาตรฐาน ANSI (American National Standards) อีกด้วย DES เป็นอัลกอริทึมแบบบล็อกซึ่ง ใช้กุญแจที่มีขนาดความยาว 56 บิตและเป็นอัลกอริทึมที่มีความแข็งแกร่ง แต่เนื่องด้วยขนาดความยาวของกุญแจที่มีขนาดเพียง 56 บิต ซึ่งในปัจจุบันถือได้ว่าสั้น เกินไปผู้บุกรุกอาจใช้วิธีการลองผิดลองถูกเพื่อค้นหากุญแจที่ถูกต้องสำหรับการถอดรหัสได้

ในปี 1998 ได้มีการสร้างเครื่องคอมพิวเตอร์พิเศษขึ้น มาซึ่ง มีมูลค่า 250,000 เหรียญสหรัฐ เพื่อใช้ในการค้นหากุญแจที่ถูกต้องของการเข้ารหัสข้อมูลหนึ่งๆ ด้วยDESและพบว่าเครื่องคอมพิวเตอร์นี้สามารถค้นหากุญแจที่ถูกต้องได้ภายในระยะเวลาไม่ถึงหนึ่งวัน

2.7.5.2 อัลกอริทึม Triple-DES

Triple-DES เป็นอัลกอริทึมที่เสริมความปลอดภัยของ DESให้มีความแข็งแกร่งมากขึ้น โดยใช้อัลกอริทึม DESเป็นจำนวนสามครั้ง เพื่อทำการเข้ารหัส แต่ละครั้ง จะใช้กุญแจในการเข้ารหัสที่แตกต่างกันดังนั้น จึงเปรียบเสมือน การใช้กุญแจเข้ารหัสที่มีความยาวเท่ากับ 56*3 = 168 บิต Triple-DES ได้ถูก ใช้งานกับสถาบันทางการเงินอย่างแพร่หลาย รวมทั้ง ใช้งานกับโปรแกรม Secure Shell (ssh) ด้วยการใช้อัลกอริทึม DES เพื่อเข้ารหัสเป็นจำนวนสองครั้ง ด้วยกุญแจสองตัว (56*2=112 บิต) ยังถือได้ว่าไม่ปลอดภัยอย่างพอเพียง

2.7.5.3 อัลกอริทึม Blowfish

Blowfish เป็นอัลกอริทึมที่มีความรวดเร็วในการทำงาน มีขนาดเล็กกะทัดรัด และใช้การเข้ารหัสแบบบล็อค ผู้พัฒนาคือ Bruce Schneier อัลกอริทึมสามารถใช้กุญแจที่มีขนาดความยาวตั้งแต่ไม่มากนักไปจนถึงขนาด 448 บิต ซึ่ง ทำให้เกิดความยืดหยุ่นสูงในการเลือกใช้กุญแจรวมทั้งอัลกอริทึมยังได้รับการออกแบบมาให้ทำงานอย่างเหมาะสมกับหน่วยประมวลผลขนาด 32 หรือ 64 บิต Blowfish ได้เปิดเผยสู่สาธารณะและไม่ได้มีการจดสิทธิบัตรใดๆ นอกจากนั้น ยังใช้ในโปรแกรม SSH และอื่นๆ

2.7.5.3 อัลกอริทึม IDEA

IDEA ย่อมาจาก International Data Encryption Algorithm อัลกอริทึมนี้ได้รับการพัฒนาในประเทศสวิสเซอร์แลนด์ที่เมือง Zarich โดย James L.Massey และ Xuejia Lai และได้รับการตีพิมพ์เผยแพร่ในปี ค.ศ. 1990 อัลกอริทึมใช้กุญแจที่มีขนาด 128 บิต และได้รับการใช้งานกับโปรแกรมยอดฮิตสำหรับการเข้ารหัสและลงลายมือชื่ออิเล็กทรอนิกส์ ในระบบอีเมล์ที่มีชื่อว่า PGP ต่อมา IDEA ได้รับการจดสิทธิบัตรทางด้านซอฟต์แวร์โดยบริษัท Ascom-Tech AG ในประเทศสวิสเซอร์แลนด์ ซึ่ง ทำให้การนำไปใช้ในงานต่างๆ เริ่มลดลง ทั้ง นี้เนื่องจากติดปัญหาเรื่องลิขสิทธินั่นเอง

2.7.5.4 อัลกอริทึม RC4

อัลกอริทึมนี้เป็นอัลกอริทึมแบบสตรีม (ทำงานกับข้อมูลทีละไบต์) ซึ่ง ได้รับการพัฒนาขึ้นมาโดย Ronald Riverst และถูกเก็บเป็นความลับทางการค้าโดยบริษัท RSA Data Security

ในภายหลังอัลกอริทึมนี้ได้รับการเปิดเผยใน Usenet เมื่อปี ค.ศ. 1994 และเป็นที่ทราบกันว่าเป็นอัลกอริทึมที่มีความแข็งแกร่งโดยสามารถใช้ขนาดความยาวของกุญแจที่มีขนาดตั้ง แต่ 1 บิตไปจนกระทั่งถึงขนาด 2048 บิต

2.7.5.5 อัลกอริทึม Rijndael (หรืออัลกอริทึม AES)

อัลกอริทึมนี้ได้รับการพัฒนาโดย Joan Daemen และ Vincent Rijmen ในปี 2000อัลกอริทึมได้รับการคัดเลือกโดยหน่วยงาน National Institute of Standard and Technology (NIST) ของสหรัฐอเมริกาให้เป็นมาตรฐานในการเข้ารหัสชั้น สูงของประเทศ อัลกอริทึมมีความเร็วสูงและมีขนาดกะทัดรัดโดยสามารถใช้กุญแจที่มีความยาวขนาด 128, 192 และ 256 บิต

2.7.5.6 อัลกอริทึม One-time Pads

อัลกอริทึมนี้ได้รับการยอมรับว่าเป็นอัลกอริทึมที่ไม่มีใครสามารถเจาะความแข็งแกร่งของอัลกอริทึมได้ อัลกอริทึมใช้กุญแจที่มีขนาดความยาว ซึ่ง อาจจะมากกว่าขนาดความยาวของข้อความที่ต้องการเข้ารหัส กุญแจจะถูกสร้างออกมาแบบสุ่ม และโดยปกติจะถูกใช้งานแค่เพียงครั้งเดียว แล้วทิ้ง ไป แต่ละไบต์ ของข้อความที่ต้องการส่งไป จะถูกเข้าและถอดรหัส โดยหนึ่ง ไบต์ (ชนิดไบต์ต่อไบต์) ของกุญแจที่ถูกสร้างขึ้น นำใช้งาน เนื่องจากกุญแจที่ถูกใช้งานแต่ละครั้ง จะไม่ซ้ำกัน และถูกสร้างขึ้น มาแบบสุ่ม จึงเป็นการยากที่จะค้นหากุญแจที่ถูกต้องได้ ข้อจำกัดของอัลกอริทึมนี้ คือขนาดของกุญแจที่อาจมีขนาดยาวกว่าข้อความที่ต้องการส่ง ซึ่งส่งผลให้การส่งมอบกุญแจที่มีขนาดใหญ่ทำได้ไม่สะดวกนัก รวมทั้ง การสร้างกุญแจให้มีความสุ่มสูงไม่ใช่เป็นสิ่งที่ทำได้ง่ายนักอย่างไรก็ตามอัลกอริทึมนี้ก็ยังมีการใช้งานในระบบเครือข่ายที่ต้องการความปลอดภัยสูง

2.8 การแก้สมการเชิงอนุพันธ์สามัญ[2]

สมการเชิงอนุพันธ์สามัญ[14] หมายถึงสมการที่เกี่ยวข้องกับอนุพันธ์เทียบกับตัวแปรตัวเดียว โดยทั่วไปคือ เวลา ถึงแม้การแก้สมการเชิงอนุพันธ์สามัญเชิงเส้นจะได้คำตอบเป็นสมการที่เรียกว่า คำตอบเชิงวิเคราะห์ (analytical solution) ทำให้สามารถเปรียบเทียบระหว่างคำตอบเชิงตัวเลข (numerical solution) กับคำตอบเชิงวิเคราะห์ และไม่ต้องกำหนดข้อจำกัดที่ของปัญหานี้

โดยระเบียบวิธีการแก้สมการเชิงอนุพันธ์สามัญมีหลายๆ รูปแบบ ในที่นี้ใช้ 4 วิธี คือ

2.8.1 ระเบียบวิธีของ Euler

ระเบียบวิธีของ Euler จัดได้ว่าเป็นระเบียบวิธีที่ง่ายแก่การเข้าใจมากที่สุด สำหรับการแก้สมการเชิง

---

[2] จากที่มา ปราโมทย์ เดชะอำไพ. 2544. ระเบียบวิธีเชิงตัวเลข ในงานวิศวกรรม
เพื่อให้เกิดความสะดวกสำหรับผู้อ่านในการทบทวนพื้นฐานจึงอ้างถึง บทความต้นฉบับดังกล่าว ณ ที่นี้อีกครั้ง

อนุพันธ์สามัญ ซึ่งเราสามารถทำการคำนวณโดยเริ่มจากเงื่อนไขเริ่มต้นของ y_i ที่ x_i และสามารถคำนวณค่า y_{i+1} ใหม่จากความกว้างช่วง h ที่กำหนดให้ ซึ่งจะได้สมการคือ

$$y_{i+1} = y_i + f(x_i + y_i)h \tag{2.7}$$

2.8.2 ระเบียบวิธีของ Huan

ระเบียบวิธีของ Huan เป็นระเบียบวิธีที่ดัดแปลงมาจากระเบียบวิธีของออยเลอร์เพื่อก่อให้เกิดผลลัพธ์จากการแก้สมการเชิงอนุพันธ์สามัญที่มีความเที่ยงตรงมากยิ่งขึ้น โดยระเบียบวิธีของ Huan เป็นระเบียบวิธีการที่ประกอบด้วยการคำนวณตัวทำนายและตัวแก้ (predictor-corrector method) ที่มีขั้นตอนดังนี้

$$\text{ตัวทำนาย } y^0_{i+1} = y_i + f(x_i + y_i)h \tag{2.8}$$

$$\text{ตัวแก้ } y_{i+1} = y_i + \frac{f(x_i + y_i) + f(x_{i+1} + y^0_{i+1})}{2}h \tag{2.9}$$

2.8.3 ระเบียบวิธีของ Euler Modifier

ระเบียบวิธีของ Euler Modifier เป็นระเบียบวิธีที่เกิดจากความเข้าใจในทำนองเดียวกันกับความเข้าใจในระเบียบวิธีของ Huan กล่าวคือ ผลลัพธ์ที่ได้จะมีความเที่ยงตรงมากขึ้นหากเราสามารถหาค่าความชันที่จะนำมาใช้ในการคำนวณได้แม่นยำมากขึ้น โดยที่ Euler Modifier แล้วมีสมการดังนี้

$$y_{i+1} = y_i + f(x_{i+1/2} + y_{i+1/2})h \tag{2.10}$$

2.8.4 ระเบียบวิธีของ Runge-Kutta

ระเบียบวิธีของ Runge-Kutta จัดได้ว่าเป็นระเบียบวิธีที่ได้รับความนิยมและใช้กันอย่างกว้างขวาง โดยเฉพาะในการคำนวณที่ต้องการผลลัพธ์ที่มีความเที่ยงตรงสูง แนวความคิดที่ใช้ในการประดิษฐ์ระเบียบวิธี Runge-Kutta นี้คือ การหาค่าความชันที่มีความเที่ยงตรงสูงเพื่อก่อให้เกิดผลลัพธ์ที่มีความเที่ยงตรงสูงตามมา สมการหลักที่ใช้ในการคำนวณผลลัพธ์ในระเบียบวิธีของ Runge-Kutta นั้นอยู่ในรูปแบบทำนองเดียวกันกับระเบียบวิธีต่างๆที่เราได้ศึกษามาก่อนหน้านี้นั่นคือ

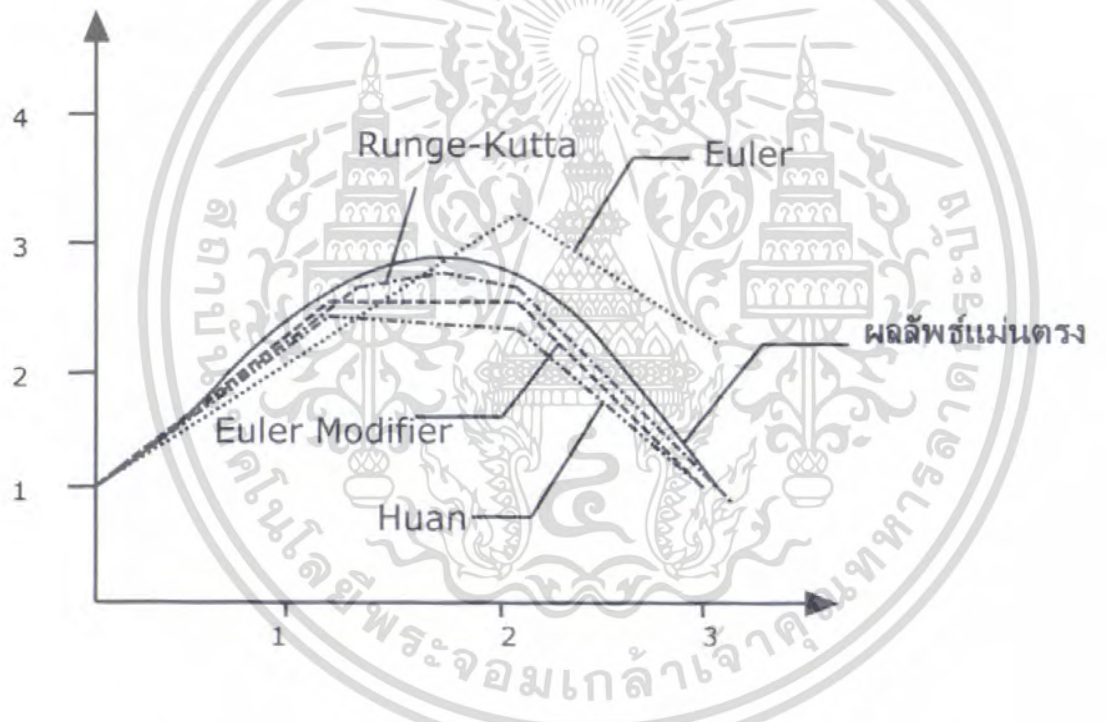
$$y_{i+1} = y_i + \varphi(x_i, y_i, h)h \tag{2.11}$$

โดย $\varphi(x_i, y_i, h)$ เรียกว่าฟังก์ชันส่วนเพิ่ม ซึ่งมีความหมายของความชันเฉลี่ยตลอดขนาดช่วงความกว้าง h ที่จะนำไปใช้ในการคำนวณหาส่วนที่เพิ่มขึ้นจากผลลัพธ์เดิม ฟังก์ชันส่วนเพิ่มนี้จะเป็นตัวบ่งบองถึงอันดับของระเบียบวิธี Runge-Kutta เช่น

$$\text{อันดับสอง } y_{i+1} = y_i + (a_1k_1 + a_2k_2)h \tag{2.12}$$

$$\text{อันดับสาม } y_{i+1} = y_i + \left[\frac{1}{6}(k_1 + 4k_2 + k_3)\right]h \tag{2.13}$$

$$\text{อันดับสี่ } y_{i+1} = y_i + \left[\frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\right]h \tag{2.14}$$



รูปที่ 2-9 การเปรียบเทียบผลลัพธ์ที่ได้จากระเบียบวิธีต่างๆใช้สมการ $\frac{dy}{dx} = y\cos x$

2.9 สถาปัตยกรรมของซีพียู ARM 7[3]

2.9.1 ประวัติความเป็นมาของซีพียู ARM

บริษัท Acorn Computer Limited ในเมืองแคมบริดจ์ ประเทศอังกฤษเป็นบริษัทแรกที่ออกแบบซีพียู ARM ในช่วงเดือนตุลาคม ค.ศ. 1983 ถึงเมษายน ค.ศ.1985 โดยสร้างต้นแบบของซีพียู ARM ในวันที่ 26 เมษายน ค.ศ.1985 จากนั้นได้ถูกส่งไปผลิตที่ บริษัท VLSI Technology Inc ประเทศสหรัฐอเมริกา[16]

บริษัท Acorn เป็นบริษัทชั้นนำผู้ผลิตเครื่องไมโครคอมพิวเตอร์ BBC microcomputer ที่ได้รับความนิยมแพร่หลายอย่างมากในประเทศอังกฤษตั้งแต่เริ่มวางจำหน่ายในเดือน มกราคม ค.ศ. 1982 เครื่อง BBC microcomputer เป็นเครื่องที่ทำงานโดยใช้ซีพียู 8 บิต เบอร์ 6502

หลังจากที่ประสบความสำเร็จกับเครื่อง BBC microcomputer แล้วทางบริษัท Acorn ได้คิดพัฒนาซีพียูสำหรับเครื่องไมโครคอมพิวเตอร์ยุคถัดไป โดยช่วงที่กำลังพัฒนาในปี ค.ศ. 1983 นั้น ซีพียู 16 บิตที่มีใช้งานแบบ CISC (Complex Instruction Set Computer) ที่มีความซับซ้อนมาก ทำงานช้ากว่าหน่วยความจำที่มีในยุคนั้น ทางวิศวกรของบริษัทจึงคิดออกแบบซีพียูเพื่อใช้งานทางการค้าเอง ซึ่งถ้าออกแบบทั้งหมดจะต้องใช้เวลาและแรงงานมาก ทางวิศวกรผู้ออกแบบจึงได้นำมาซีพียู Berkeley RISC I ซึ่งเป็นซีพียูแบบ RISC (Reduce Instruction Set Computer) ขนาด 32 บิต ที่ถูกออกแบบโดยนักศึกษาปริญญาโทของมหาวิทยาลัย Berkeley ที่เป็นซีพียูแบบง่ายๆ ไม่ซับซ้อน ใช้งานด้านการศึกษามาพัฒนาต่อให้เป็นซีพียู 32 บิตอเนกประสงค์เพื่อใช้งานทางการค้า ซีพียู ARM มีข้อดีในเรื่องสถาปัตยกรรมที่ไม่ซับซ้อน ประหยัดพื้นที่ในการผลิตชิปซีพียู กินพลังงานน้อย โดยที่ยังคงมีสมรรถนะที่สูง

หลังจากที่ออกแบบ และทดสอบจนพอใจแล้ว ได้มีการก่อตั้งบริษัท Advanced RISC Machines Ltd. ขึ้นในเดือนพฤศจิกายนปีค.ศ.1990 ในประเทศอังกฤษ โดยเป็นบริษัทร่วมทุนระหว่างบริษัท Apple Computer, Acorn Computer Group และ VLSI Technology โดยบริษัท Apple และ VLSI Technology เป็นผู้ลงทุน โดยบริษัท Acorn เป็นผู้อนุญาตให้ใช้เทคโนโลยีและโอนทีมวิศวกรผู้ออกแบบจำนวน 12 คน

ซีพียู ARM รุ่นแรกที่ออกจำหน่ายในปี 1991 เป็นซีพียูตระกูล ARM6 จัดว่าเป็นซีพียู RISC 32 บิตแบบฝังตัว (embedded RISC) ตัวแรกของโลก นับเป็นการสร้างมาตรฐานใหม่ให้กับวงการซีพียู 32 บิต หลังจากนั้นก็มีบริษัทผู้ผลิตซีพียูไมโครโปรเซสเซอร์ และไมโครคอนโทรลเลอร์ทั่วโลกอื่นๆ เข้าร่วมผลิตซีพียูตระกูล ARM ในปัจจุบันมีซีพียู ARM ต่างๆดังนี้ ARM7, ARM9, ARM9E, ARM10 และ ARM11

---

[3] จากที่มา http://www.ett.co.th/product/ARM/CP_JR_ARM7_USB_LPC2148.htm
เพื่อให้เกิดความสะดวกสำหรับผู้อ่านในการทบทวนพื้นฐานจึงอ้างถึง บทความต้นฉบับดังกล่าว ณ ที่นี้อีกครั้ง

ในยุคแรกซีพียู ARM เป็นซีพียู RISC 32 บิต การทำงานจำเป็นต้องต่อกับหน่วยความจำและอุปกรณ์ภายนอก เมื่อมีบริษัทผู้ผลิตไมโครคอนโทรลเลอร์จำนวนมากได้นำลิขสิทธิ์ของซีพียู ARM ไปพัฒนาต่อ ได้มีการเพิ่มหน่วยความจำภายในทั้ง ROM และ RAM และเพิ่มโมดูลอุปกรณ์เสริมต่างๆ เช่น วงจรสื่อสารแบบอนุกรม วงจรแปลงดิจิตอลเป็นแอนะลอก ฯลฯ ทำให้กลายเป็นไมโครคอนโทรลเลอร์แบบ 32 บิตที่กินพลังงานต่ำ สามารถทำงานได้โดยใช้ชิปไอซีเพียงตัวเดียวโดยไม่ต้องต่ออุปกรณ์เพิ่มเติมภายนอก ทำให้มีการนำไปใช้ในงานคอมพิวเตอร์ฝังตัว (embedded computer) มากขึ้น

2.9.2สถาปัตยกรรมซีพียู ARM7

สถาปัตยกรรม ARM7 เป็นซีพียูแบบ RISC ขนาด 32 บิต ภายในมีบัสขนาด 32 บิตตัวเดียวที่ใช้สำหรับรับส่งข้อมูล และคำสั่ง ชุดคำสั่งจะมีขนาด 32 บิตคงที่ ในขณะที่ข้อมูลสามารถเลือกได้ว่าจะมีขนาด 8, 16 หรือ 32 บิต โดยแสดงแกนกลาง (core) ของซีพียู ARM7

โครงสร้างของ ARM7 จะเป็นแบบที่เรียบง่าย มีชุดคำสั่งไม่มากนัก ประหยัดพื้นที่สารกึ่งตัวนำที่ใช้สร้าง ประหยัดพลังงาน

สถาปัตยกรรมของ ARM7 จะเป็นแบบ load-and-store ในการประมวลผลข้อมูลใดๆต้องกระทำผ่านทางรีจิสเตอร์ เริ่มต้นด้วยการโหลดค่าจากหน่วยความจำเก็บในรีจิสเตอร์นำค่ามาประมวลผล เสร็จแล้วจะเขียนค่าเก็บในหน่วยความจำดังเดิม

รีจิสเตอร์ของ ARM7 ที่ใช้งานได้สำหรับผู้ใช้มีทั้งหมด 16 ตัวคือ R0-R15 โดยทุกตัวมีขนาด 32 บิต โดย R0-R12 เป็น รีจิสเตอร์ทั่วไปที่ไม่ได้กำหนดหน้าที่การทำงานพิเศษ ส่วน R12 ทำหน้าที่เป็น stack pointer (sp) R14 ทำหน้าที่เป็น link register (LR) และ R15 ทำหน้าที่เป็น Program Counter (PC)

2.9.3 ระบบการทำงานของ ARM7

ขั้นตอนการออกแบบชุดคำสั่งของ ARM7 นั้นเป็นชนิด RISC (reduce instruction set computer) ที่มีการออกแบบให้มีคำสั่งขนาดเล็ก ทำให้สามารถประมวลผลได้รวดเร็ว โดยเฉพาะใน ARM7 นี้คำสั่งที่ออกแบบนั้นสามารถทำงานได้เสร็จเพียงใน 1 รอบการทำงาน (single cycle) เท่านั้น

2.9.3.1 ไปป์ไลน์

ไปป์ไลน์(pipeline)หรือการทำงานแบบสายท่อของ ARM7 นั้นมีการออกแบบไปป์ไลน์ของชุดคำสั่งเอาไว้ 3 สเตจ (stage) คือ ระยะของการอ่านชุดคำสั่ง(fetch)ระยะการถอดรหัสของชุดคำสั่ง (decode)และระยะของการทำงานตามชุดคำสั่ง(execute)ซึ่งการออกแบบด้วยวิธีการนี้มีข้อดีคือทำให้สามารถเรียกคำสั่งได้หลายคำสั่งมาซ้อนกันได้ ดังรูปที่ 2-9 จะเห็นว่าในช่วงแรกของการทำงานรอบที่ 1 นั้นจะมีการอ่านชุดคำสั่งที่ 1 เข้าไปสู่ไปป์ไลน์ หลังจากนั้นเมื่อทำการถอดรหัสชุดคำสั่งที่ 1 ระยะการอ่านจะว่างจึงทำการอ่านชุดคำสั่งที่ 2 เมื่อทำการทำตามชุดคำสั่ง

ที่ 1 อยู่นั้นชุดคำสั่งที่ 2 ก็จะถูกถอดรหัสและชุดคำสั่งที่ 3 จะถูกอ่านเข้ามา และเมื่อทำไปเรื่อยๆจะเห็นว่าจะสามารถทำงานได้ 4 คำสั่งโดยใช้การทำงานเพียง 2 รอบเท่านั้น

	ชุดคำสั่งที่ 1	ชุดคำสั่งที่ 2	ชุดคำสั่งที่ 3	ชุดคำสั่งที่ 4
การทำงานรอบที่ 1	Fetch	รอ	รอ	รอ
	Decode	Fetch	รอ	รอ
	Execute	Decode	Fetch	รอ
การทำงานรอบที่ 2	ทำงานเสร็จ	Execute	Decode	Fetch
	ทำงานเสร็จ	ทำงานเสร็จ	Execute	Decode
	ทำงานเสร็จ	ทำงานเสร็จ	ทำงานเสร็จ	Execute
	ทำงานเสร็จ	ทำงานเสร็จ	ทำงานเสร็จ	ทำงานเสร็จ

รูปที่ 2-10 ไปป์ไลน์ของ ARM7 ทั้ง 3 ขั้นตอน

2.9.3.2 รีจิสเตอร์

รีจิสเตอร์ (Register) เป็นหน่วยความจำที่อยู่ในหน่วยประมวลผล และตัว ARM7 มีสถาปัตยกรรมการทำงานแบบ โหลดและสโตร์ (load and store) ซึ่งหมายความว่า ข้อมูลที่จะใช้ในการประมวลผลนั้นจะต้องถูกนำเข้ามาเก็บในรีจิสเตอร์แล้วจึงทำการประมวลผลและเมื่อได้ทำการประมวลผลเสร็จ ผลลัพธ์ที่ได้นั้นจะถูกนำไปเก็บเอาไว้ในรีจิสเตอร์ ซึ่งสรุปการทำงานได้ 3 ขั้นตอนดังนี้

ขั้นตอนที่ 1 ทำการโหลดข้อมูลมาเก็บในรีจิสเตอร์

ขั้นตอนที่ 2 ทำการประมวลผลจากข้อมูลในรีจิสเตอร์

ขั้นตอนที่ 3 นำผลลัพธ์ที่ได้เก็บในรีจิสเตอร์ปลายทาง

เช่น ถ้าต้องการหาผลบวกของ M1 กับ M2 แล้วนำผลลัพธ์มาเก็บใน M3 สามารถนำมาเขียนเป็น 3 ขั้นตอนได้ดังนี้

ขั้นตอนที่ 1 ทำการโหลด M1 ไปเก็บในรีจิสเตอร์ (สมมุติว่าเป็น R1)

ทำการโหลด M2 ไปเก็บในรีจิสเตอร์ (สมมุติว่าเป็น R2)

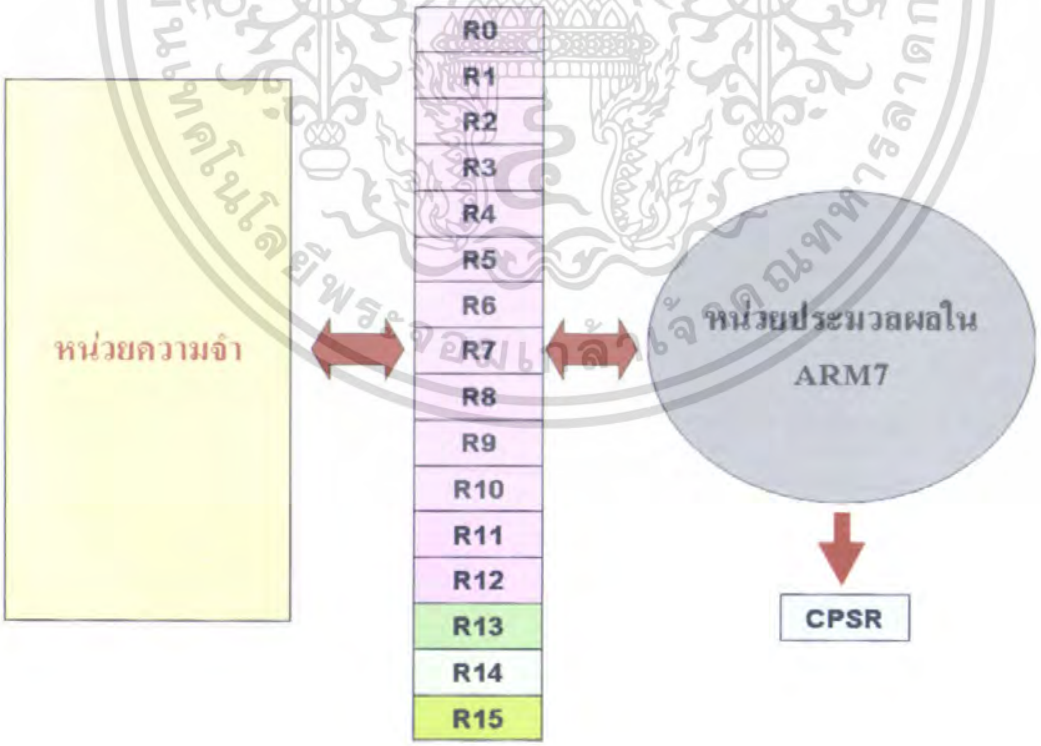
ขั้นตอนที่ 2 เรียกคำสั่งบวก นั้นคือ ADD R3, R1, R2 (สมมุติว่าเก็บผลลัพธ์ ใน R3) คำสั่งนี้มีความหมายว่า R3 = R1 + R2

ขั้นตอนที่ 3 โอนข้อมูลจากรีจิสเตอร์ (R3) มาเก็บใน M3

รีจิสเตอร์สำหรับผู้ใช้ของหน่วยประมวลผล ARM7 นั้นจะมีขนาด 32 บิต ซึ่งมีทั้งหมด 17 ตัวคือ R0 ถึง R15 และ CPSR (current program status register) โดยจำแนกกลุ่มทำงานได้ดังนี้

- R0 ถึง R12 สามารถใช้งานได้ตามที่ผู้ใช้หรือเขียนโปรแกรมต้องการ
- R13 ถูกใช้เป็นที่เก็บค่าตัวชี้ไปยังหน่วยความจำสแตก (stack pointer)
- R14 หรือเรียกว่า LR (link register) ถูกใช้เป็นที่เก็บตำแหน่งของชุดคำสั่งถัดไปที่จะต้องประมวลผลก่อนที่จะเกิดการเรียกโปรแกรมย่อย (call) ซึ่งเป็นชุดคำสั่งที่จะต้องนำมาทำงานหลังจากที่โปรแกรมย่อยนั้นทำงานเสร็จแล้ว
- R15 หรือเรียกว่า PC (program counter) ถูกใช้เป็นที่เก็บตำแหน่งของคำสั่งถัดไปที่จะถูกนำมาประมวลผล
- CPSR ใช้เป็นที่เก็บสถานการณ์ทำงานของคำสั่งที่ถูกประมวลผลไปหลังสุด ซึ่งนิยมเรียกรีจิสเตอร์นี้กันว่า เรจิสเตอร์สถานะ หรือ รีจิสเตอร์แฟล็ก (flag register)

จากรีจิสเตอร์ที่กล่าวมานี้ สามารถนำมาเขียนความสัมพันธ์ระหว่างกันได้ดังรูปที่ 2-11 ซึ่งจะเห็นว่าหน่วยความจำนั้นจะต้องถ่ายโอน/แลกเปลี่ยนข้อมูลกับรีจิสเตอร์ โดยการประมวลผลนั้น ตัวประมวลผลจะใช้ข้อมูลจากรีจิสเตอร์ประกอบการทำงาน และเมื่อทำงานเสร็จจะบันทึกสถานะของการทำงานจากคำสั่งปัจจุบันเก็บเอาไว้ใน CPSR



รูปที่ 2-11 ความสัมพันธ์ระหว่างหน่วยความจำ รีจิสเตอร์และหน่วยประมวลผลของ ARM7

2.9.3.3 รีจิสเตอร์เก็บสถานการณ์ทำงาน

รีจิสเตอร์เก็บสถานการณ์ทำงานของ ARM7 จะมีขนาด 32 บิตมีหน้าที่รายงานและใช้ควบคุมการทำงานของหน่วยประมวล ARM7 นั้นคือ เมื่อหน่วยประมวลผลทำงานเสร็จจะเก็บสถานการณ์ทำงานใน CPSR แล้วผู้เขียนโปรแกรมสามารถนำค่าเหล่านี้มาใช้เป็นการควบคุมโปรแกรมที่เขียนเพื่อสั่งงานหน่วยประมวลผล หรืออาจจะกำหนดสถานะบางอย่างเพื่อให้หน่วยประมวลผลทำงานภายใต้ภาวะ (mode) ที่ผู้เขียนโปรแกรมกำหนด

โครงสร้างของรีจิสเตอร์ตัวนี้เป็นดังรูปที่ 2-11 แบ่งได้ 2 กลุ่ม คือ

1. กลุ่มรายงานสถานการณ์ทำงาน ได้แก่

- N (negative) เป็น 1 เมื่อผลการทำงานก่อให้เกิดค่าลบ
- Z (zero) เป็น 1 เมื่อผลการทำงานทำให้เกิดค่าศูนย์
- C (carry) เป็น 1 เมื่อผลการทำงานทำให้เกิดการทดค่า
- V (overflow) เป็น 1 เมื่อผลการทำงานเกิดล้นของข้อมูล

2. กลุ่มกำหนดการทำงานของหน่วยประมวลผล ได้แก่

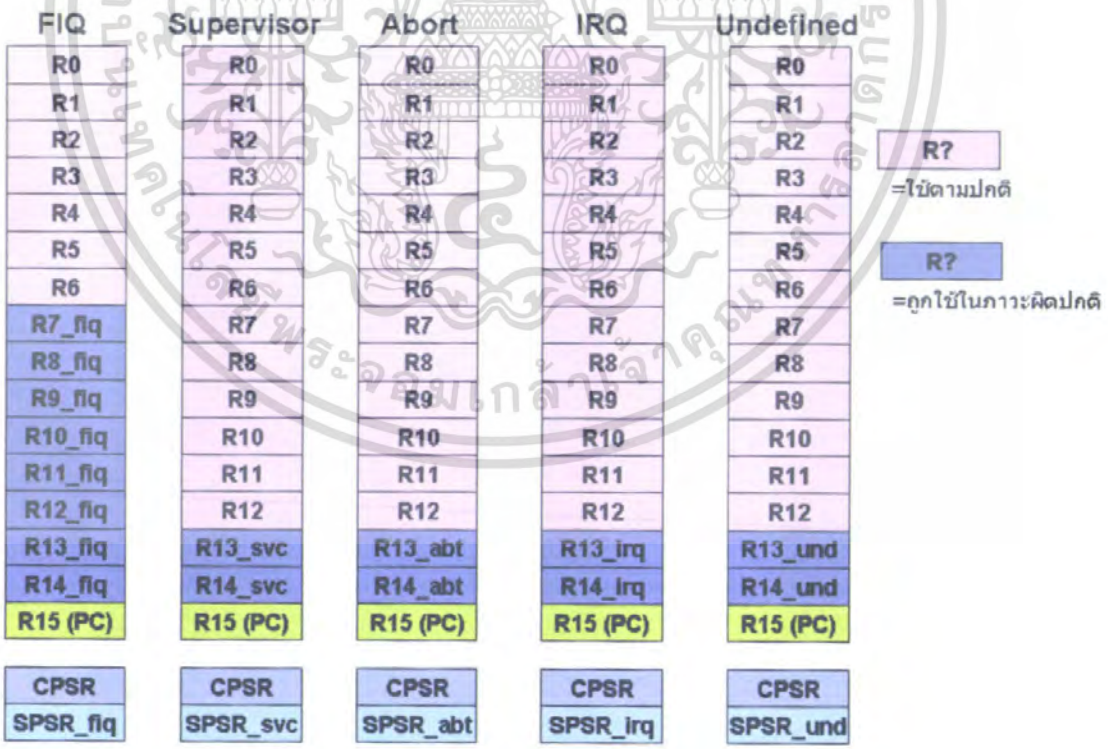
- I (interrupt enable) กำหนดให้เป็น 1 เมื่อต้องการดักสัญญาณขัดจังหวะแบบ IRQ
- F (fast interrupt enable) กำหนดให้เป็น 1 เมื่อต้องการดักสัญญาณขัดจังหวะแบบ FIQ
- T (thumb instruction set) กำหนดเป็น 1 เมื่อต้องการให้ทำงานชุดคำสั่งแบบ 16 บิต ซึ่งจะทำให้ ARM7 สามารถประมวลผลคำสั่งได้ 2 คำสั่งในการทำงานเพียง 1 รอบการทำงาน
- M4, M3, M2, M1, M0 ใช้ในการกำหนดภาวะการณ์ทำงาน ซึ่งมีทั้งหมด 7 โหมด (mode) ที่มีผลต่อการใช้งาน รีจิสเตอร์แตกต่างกันไป แต่อย่างไรก็ดี ในการทำงานแบบโหมดผู้ใช้ (user mode) จะมีการทำงานเหมือนกับที่ได้อธิบายเอาไว้ในหัวข้อรีจิสเตอร์

รูปที่ 2-12 ความหมายของบิตในรีจิสเตอร์สถานการณ์ทำงาน

2.9.3.4 ภาวะผิดปกติ

ภาวะผิดปกติ (Exception modes) เป็นการเกิดสิ่งที่ผิดปกติ จากการทำงานทั่วไป และเมื่อเกิดสิ่งผิดปกตินี้หน่วยประมวลผลจะเปลี่ยนภาวการณ์ทำงานของตัวเองพร้อมทั้งเปลี่ยนแปลงค่าของ รีจิสเตอร์ PC เพื่อกระโดดไปทำงาน ณ ตำแหน่งที่กำหนดเอาไว้ตารางที่ด้านล่างเพื่อตอบสนองต่อการทำงานตามชนิดของภาวะผิดปกติที่เกิดขึ้น โดยแต่ละภาวะนั้นจะมีการใช้รีจิสเตอร์แตกต่างกันออกไปดังรูป 2-12

สิ่งผิดปกติ	ภาวะการทำงาน	ตำแหน่ง
Reset	Supervisor	0x00000000
Undefined instruction	Undefined	0x00000004
Software interrupt (SWI)	Supervisor	0x00000008
Prefetch Abort (fetch memory abort)	Abort	0x0000000C
Data Abort (data access memory abort)	Abort	0x00000010
IRQ (interrupt)	IRQ	0x00000018
FIQ (fast insterrupt)	FIQ	0x0000001C



รูปที่ 2-13 การใช้รีจิสเตอร์ในภาวะผิดปกติ

เนื่องจากมีภาวะผิดปกติได้หลายประเภท ATM7 จึงมีการจัดลำดับความสำคัญเรียงจากมากไปน้อยไว้ดังนี้

1. Reset
2. Data Abort
3. FIQ
4. IRQ
5. Prefetch Abort
6. Undefined instruction และ SWI

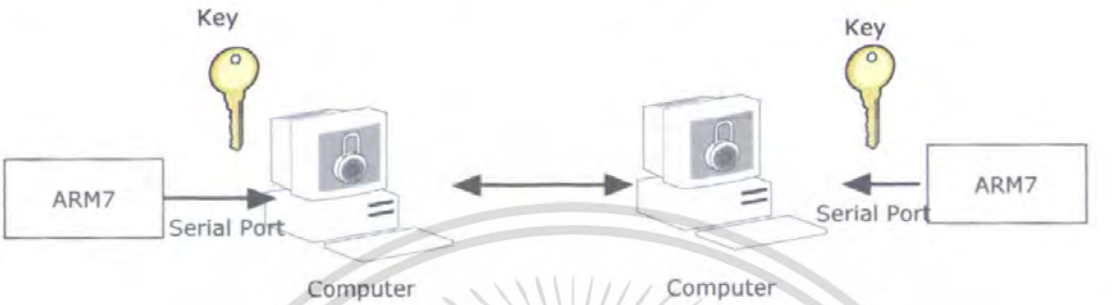
ขออธิบายลำดับการทำงานเมื่อเกิดสิ่งผิดปกติเป็นขั้นตอนดังนี้

1. อ่านตำแหน่งถัดไปที่จะเรียกขึ้นมาทำงานไปเก็บใน LR
2. คัดลอกสถานะใน CPSR ไปเก็บใน SPSR
3. นำค่าข้อมูลจากตำแหน่งที่ใช้ตอบสนองกับภาวะที่ผิดปกติไปเก็บใน PC ซึ่งหมายความว่าค่าที่เก็บในหน่วยความจำที่กำหนดตามตารางที่ 1 นั้นจะเป็นค่าตำแหน่งของชุดคำสั่งที่จะถูกเรียกใช้งานเมื่อเกิดสิ่งผิดปกติ
4. เริ่มทำงาน ณ ตำแหน่งที่ระบุใน PC
5. เมื่อทำงานชุดคำสั่งตอบสนองสิ่งผิดปกติเสร็จแล้วจะนำค่าที่เก็บใน LR มาเก็บใน PC เพื่อเตรียมทำงานชุดคำสั่งปกติ
6. นำค่าจาก SPSR มาเก็บใน CPSR เพื่อนำสถานการณ์ทำงานปกติกลับมา
7. เรียกคำสั่ง ณ ตำแหน่ง ณ ตำแหน่งที่เก็บใน PC มาทำงานแล้วเลื่อน PC เป็นตำแหน่งของชุดคำสั่งถัดไป

บทที่ 3

การออกแบบและการทดลอง

ในการออกแบบการทดลองจะทำตาม ขั้นตอน ดังรูป



รูปที่ 3-1 ขั้นตอนการออกแบบและการเข้ารหัสลับ

3.1 การออกแบบโปรแกรมหาค่า กุญแจ

ในด้าน โปรแกรมได้นำทฤษฎี เคออสติก ที่ได้ศึกษามาทำการแก้สมการหา กุญแจ ที่จะนำไป เข้ารหัสข้อมูล โดยได้เลือก รูปแบบต่างๆไว้ดังนี้

ทฤษฎี เคออสติก แบบต่อเนื่อง(Continuous)

- สมการของ Lorenz
- สมการของ Chen
- สมการของ Rossler

ทฤษฎี เคออสติก แบบไม่ต่อเนื่อง(Discrete)

- Logistic map
- Cat map

ทฤษฎีการแก้สมการเชิงอนุพันธ์

- ระเบียบวิธีของ Euler
- ระเบียบวิธีของ Huan
- ระเบียบวิธีของ Euler Modifier
- ระเบียบวิธีของ Runge-Kutta อันดับสาม
- ระเบียบวิธีของ Runge-Kutta อันดับสี่

โดยได้ทำการทดลอง เขียนโปรแกรมด้วยภาษาซี(C language) เพื่อให้คอมพิวเตอร์สามารถประมวลผลค่าผลลัพธ์ออกมาได้และนำโปรแกรมที่ได้เขียนไปประมวลผลบนMicro controller ARM7

3.1.1 การทดลองการประมวลผลสมการ chaotic โดยใช้ สมการ Lorenz

3.1.1.1 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Euler

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.01 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
float akx,aky,akz,x,y,z,h;
int i,n,a,b;
double c;
float fx(float x,float y,float z)
        {return a*(y-x);}
float fy(float x,float y,float z)
        { return x*(b - z) - y ;}
float fz(float x,float y,float z)
        {  return x*y-c*z;}
void main()
{       x=0.1; y=0; z=0; n=10000; h=0.01; a=10; b=28; c=8/3;
        for (i=1; i <= n ;i++ )
        {
         akx=fx(x,y,z);
         aky=fy(x,y,z);
         akz=fz(x,y,z);
         x=x+(akx*h);
         y=y+(aky*h);
        z=z+(akz*h);
        printf("%.8f      %.8f      %.8f  \n  " ,x,y,z);
        }
}
```

3.1.1.2 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Huan

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.01 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
float akx,akx1,akxx,aky,aky1,akyy,akz,akz1,akzz,x,y,z,h,zz,yy,xx;
int i,n,a,b;
double c;
float fx(float x,float y,float z)
        {return a*(y-x);}
float fy(float x,float y,float z)
        {return x*(b - z) - y ;}
float fz(float x,float y,float z)
        { return x*y-c*z;}
void main()
{       x=0.1; y=0; z=0; n=10000; h=0.01; a=10; b=28; c=8/3;
        for (i=1; i <= n ;i++ )
        {akx=fx(x,y,z);
         aky=fy(x,y,z);
          akz=fz(x,y,z);
          xx=x+akx*h;
          yy=y+aky*h;
          zz=z+akz*h;
          akxx=fx(xx,yy,zz);
           akyy=fy(xx,yy,zz);
          akzz=fz(xx,yy,zz);
          akx1=(akx+akxx)/2;
            aky1=(aky+akyy)/2;
          akz1=(akz+akzz)/2;
          x=x+akx1*h;
         y=y+aky1*h;
          z=z+akz1*h;
        printf("%.8f        %.8f        %.8f  \n  " ,x,y,z);
        }
}
```

3.1.1.3 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Euler

Modifier

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.01 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
float akx,aky,akz,akx1,aky1,akz1,x,y,z,h,xx,yy,zz;
int i,n,a,b;
double c;
float fx(float x,float y,float z)
        {return a*(y-x);}
float fy(float x,float y,float z)
        { return x*(b - z) - y ;}
float fz(float x,float y,float z)
        {return x*y-c*z;}
void main()
{       x=0.1; y=0; z=0; n=10000; h=0.01; a=10; b=28; c=8/3;
        for (i=1; i <= n ;i++ )
        {
         akx=fx(x,y,z);
         aky=fy(x,y,z);
         akz=fz(x,y,z);
         xx=x+(akx*h)/2;
         yy=y+(aky*h)/2;
         zz=z+(akz*h)/2;
         akx1=fx(xx,yy,zz);
         aky1=fy(xx,yy,zz);
         akz1=fz(xx,yy,zz);
         x=x+(akx1*h);
         y=y+(aky1*h);
        z=z+(akz1*h);
        printf("%.8f      %.8f      %.8f  \n  " ,x,y,z);
        }
}
```

3.1.1.4 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Runge-Kutta'3rd order

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.01 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
float akx1,akx2,akx3,aky1,aky2,aky3,akz1,akz2,akz3,x,xx,y,yy,z,zz,h;
int i,n,a,b;
double c;
float fx(float x,float y,float z)
        {return a*(y-x);}
float fy(float x,float y,float z)
        { return x*(b - z) - y ;}
float fz(float x,float y,float z)
        {return x*y-c*z;}
void main()
{       x=0.1; y=0; z=0; n=10000; h=0.01; a=10; b=28; c=8/3;
        for (i=1; i <= n ;i++ )
        {
                akx1=fx(x,y,z);
                aky1=fy(x,y,z);
                akz1=fz(x,y,z);
                xx=x+(h*akx1)/2;
                yy=y+(h*aky1)/2;
                zz=z+(h*akz1)/2;
                akx2=fx(xx,yy,zz);
                aky2=fy(xx,yy,zz);
                akz2=fz(xx,yy,zz);
                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);
                akx3=fx(xx,yy,zz);
                aky3=fy(xx,yy,zz);
                akz3=fz(xx,yy,zz);
                x=x+(akx1+4*akx2+akx3)*(h/6);
                y=y+(aky1+4*aky2+aky3)*(h/6);
                z=z+(akz1+4*akz2+akz3)*(h/6);
                printf("%.8f        %.8f        %.8f  \n  " ,x,y,z);
        }
}
```

3.1.2 การทดลองการประมวลผลสมการ chaotic โดยใช้ สมการ Chen

3.1.2.1 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Euler

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.001 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
float akx,aky,akz,x,y,z,h;
int i,n,a,b,c;
float fx(float x,float y,float z)
        {return a*(y-x);}
float fy(float x,float y,float z)
        { return (c-a)*x-(x*z)+(c*y);}
float fz(float x,float y,float z)
        {return x*y-b*z;}
void main()
{       x=0.1; y=0; z=0; n=10000; h=0.001; a=35; b=3; c=28;
        for (i=1; i <= n ;i++ )
        {
          akx=fx(x,y,z);
          aky=fy(x,y,z);
          akz=fz(x,y,z);
          x=x+(akx*h);
          y=y+(aky*h);
          z=z+(akz*h);
printf("%.8f      %.8f      %.8f  \n ",x,y,z);
        }
}
```

3.1.2.2 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Huan

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.001 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
float akx,akxx,aky,akyy,akz,akzz,x,y,z,h;
int i,n,a,b,c;
float fx(float x,float y,float z)
        {return a*(y-x);}
float fy(float x,float y,float z)
        { return (c-a)*x-(x*z)+(c*y);}
float fz(float x,float y,float z)
```

```
        {return x*y-b*z;}
void main()
{       x=0.1; y=0; z=0; n=10000; h=0.001; a=35; b=3; c=28;
        for (i=1; i <= n ;i++ )
        { akx=fx(x,y,z);
          aky=fy(x,y,z);
          akz=fz(x,y,z);
          x=x+akx*h;
          y=y+aky*h;
          z=z+akz*h;
          akxx=fx(x,y,z);
          akyy=fy(x,y,z);
          akzz=fz(x,y,z);
          akx=(akx+akxx)/2;
          aky=(aky+akyy)/2;
          akz=(akz+akzz)/2;
          x=x+akx*h;
          y=y+aky*h;
          z=z+akz*h;
          printf("%.8f        %.8f        %.8f  \n  ",x,y,z);
        }
}
```

3.1.2.3 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Euler Modifier

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.001 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
float akx,aky,akz,akx1,aky1,akz1,x,y,z,h,xx,yy,zz;
int i,n,a,b,c;
float fx(float x,float y,float z)
        {return a*(y-x);}
float fy(float x,float y,float z)
        { return (c-a)*x-(x*z)+(c*y);}
float fz(float x,float y,float z)
        { return x*y-b*z;}

void main()
{       x=0.1; y=0; z=0; n=10000; h=0.001; a=35; b=3; c=28;
        for (i=1; i <= n ;i++ )
        {
          akx=fx(x,y,z);
          aky=fy(x,y,z);
          akz=fz(x,y,z);
```

```
        xx=x+(akx*h)/2;
        yy=y+(aky*h)/2;
        zz=z+(akz*h)/2;
        akx1=fx(xx,yy,zz);
        aky1=fy(xx,yy,zz);
        akz1=fz(xx,yy,zz);
        x=x+(akx1*h);
        y=y+(aky1*h);
        z=z+(akz1*h);
      printf("%.8f      %.8f      %.8f  \n  " ,x,y,z);
      }
}
```

3.1.2.4 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Runge-Kutta'3rd

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.001 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
float akx1,akx2,akx3,aky1,aky2,aky3,akz1,akz2,akz3,x,xx,y,yy,z,zz,h;
int i,n,a,b,c;
float fx(float x,float y,float z)
      {return a*(y-x);}
float fy(float x,float y,float z)
      { return (c-a)*x-(x*z)+(c*y);}

float fz(float x,float y,float z)
      { return x*y-b*z;}
void main()
{     x=0.1; y=0; z=0; n=10000; h=0.001; a=35; b=3; c=28;
      for (i=1; i <= n ;i++ )
      {
            akx1=fx(x,y,z);
            aky1=fy(x,y,z);
            akz1=fz(x,y,z);
            xx=x+(h*akx1)/2;
            yy=y+(h*aky1)/2;
            zz=z+(h*akz1)/2;
            akx2=fx(xx,yy,zz);
            aky2=fy(xx,yy,zz);
            akz2=fz(xx,yy,zz);
            xx=x-(h*akx1)+(2*h*akx2);
            yy=y-(h*aky1)+(2*h*aky2);
            zz=z-(h*akz2)+(2*h*akz2);
            akx3=fx(xx,yy,zz);
            aky3=fy(xx,yy,zz);
```

```
            akz3=fz(xx,yy,zz);
            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
             printf("%.8f        %.8f        %.8f  \n  " ,x,y,z);
        }
}
```

3.1.2.5 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Runge-Kutta'4th

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.001 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
float
akx1,akx2,akx3,akx4,aky1,aky2,aky3,aky4,akz1,akz2,akz3,akz4,x,xx,y,yy,z,zz,h;
int i,n,a,b,c;
float fx(float x,float y,float z)
        {return a*(y-x);}
float fy(float x,float y,float z)
        {return (c-a)*x-(x*z)+(c*y);}
float fz(float x,float y,float z)
        {  return x*y-b*z;}
void main()
{       x=0.1;y=0;z=0;n=10000;h=0.001;a=35;b=3;c=28;
        for (i=1; i <= n ;i++ )
        {       akx1=fx(x,y,z);
                aky1=fy(x,y,z);
               akz1=fz(x,y,z);
               xx=x+(h*akx1)/2;
               yy=y+(h*aky1)/2;
               zz=z+(h*akz1)/2;
               akx2=fx(xx,yy,zz);
               aky2=fy(xx,yy,zz);
               akz2=fz(xx,yy,zz);
               xx=x+(h*akx2)/2;
               yy=y+(h*aky2)/2;
               zz=z+(h*akz2)/2;
               akx3=fx(xx,yy,zz);
               aky3=fy(xx,yy,zz);
               akz3=fz(xx,yy,zz);
               xx=x+h*akx3;
               yy=y+h*aky3;
               zz=z+h*akz3;
               akx4=fx(xx,yy,zz);
               aky4=fy(xx,yy,yy);
```

```
            akz4=fz(xx,yy,zz);
            x=x+(akx1+2*akx2+2*akx3+akx4)*(h/6);
            y=y+(aky1+2*aky2+2*aky3+aky4)*(h/6);
            z=z+(akz1+2*akz2+2*akz3+akz4)*(h/6);
             printf("%.8f        %.8f        %.8f  \n  " ,x,y,z);
        }
}
```

3.1.3 การทดลองการประมวลผลสมการ chaotic โดยใช้ สมการ Rossler

3.1.3.1 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Euler

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.01 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h
float akx,aky,akz,x,y,z,h;
int i,n;
double a,b,c;
float fx(float x,float y,float z)
        {return - y - z;}
float fy(float x,float y,float z)
        {return x + a*y;}
float fz(float x,float y,float z)
        {return b + z*( x - c );}
void main()
{       x=0.1; y=0; z=0; n=10000; h=0.01; a=0.2; b=0.2; c=5.7;
for (i=1; i <= n ;i++ )
        {
         akx=fx(x,y,z);
        aky=fy(x,y,z);
         akz=fz(x,y,z);
         x=x+(akx*h);
        y=y+(aky*h);
        z=z+(akz*h);
printf("%.8f        %.8f        %.8f  \n  " ,x,y,z);
        }
}
```

3.1.3.2 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Huan

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.01 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
float akx,akx1,akxx,aky,aky1,akyy,akz,akz1,akzz,x,y,z,h,zz,yy,xx;
int i,n;
double c,b,a;
float fx(float x,float y,float z)
        {return - y - z;}
float fy(float x,float y,float z)
        {return x + a*y;}
float fz(float x,float y,float z)
        {  return b + z*( x - c );}
void main()
{       x=0.1; y=0; z=0; n=10000; h=0.01; a=0.2; b=0.2; c=5.7;
        for (i=1; i <= n ;i++ )
        {
          akx=fx(x,y,z);
          aky=fy(x,y,z);
          akz=fz(x,y,z);
          xx=x+akx*h;
          yy=y+aky*h;
          zz=z+akz*h;
          akxx=fx(xx,yy,zz);
          akyy=fy(xx,yy,zz);
          akzz=fz(xx,yy,zz);
          akx1=(akx+akxx)/2;
          aky1=(aky+akyy)/2;
          akz1=(akz+akzz)/2;
          x=x+akx1*h;
          y=y+aky1*h;
          z=z+akz1*h;
        printf("%.8f        %.8f        %.8f  \n  ",x,y,z);
        }
}
```

3.1.3.3 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Euler

Modifier

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.01 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
float akx,aky,akz,akx1,aky1,akz1,x,y,z,h,xx,yy,zz;
int i,n;
double c,b,a;
float fx(float x,float y,float z)
        {return - y - z;}
float fy(float x,float y,float z)
        { return x + a*y;}
float fz(float x,float y,float z)
        {  return b + z*( x - c );}
void main()
{       x=0.1; y=0; z=0; n=10000; h=0.01; a=0.2; b=0.2; c=5.7;
        for (i=1; i <= n ;i++ )
        {
         akx=fx(x,y,z);
         aky=fy(x,y,z);
         akz=fz(x,y,z);

         xx=x+(akx*h)/2;
         yy=y+(aky*h)/2;
         zz=z+(akz*h)/2;
         akx1=fx(xx,yy,zz);
         aky1=fy(xx,yy,zz);
         akz1=fz(xx,yy,zz);
         x=x+(akx1*h);
        y=y+(aky1*h);
        z=z+(akz1*h);
        printf("%.8f        %.8f        %.8f  \n  " ,x,y,z);
        }
}
```

3.1.3.4 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Runge-Kutta'3rd

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.01 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
float akx1,akx2,akx3,aky1,aky2,aky3,akz1,akz2,akz3,x,xx,y,yy,z,zz,h;
int i,n;
double c,b,a;
float fx(float x,float y,float z)
        {return - y - z;};
float fy(float x,float y,float z)
        {return x + a*y;}
float fz(float x,float y,float z)
        { return b + z*( x - c );}
void main()
{       x=0.1; y=0; z=0; n=10000; h=0.01; a=0.2; b=0.2; c=5.7;
        for (i=1; i <= n ;i++ )
        {       akx1=fx(x,y,z);
                aky1=fy(x,y,z);
                akz1=fz(x,y,z);
                xx=x+(h*akx1)/2;
                yy=y+(h*aky1)/2;
                zz=z+(h*akz1)/2;
                akx2=fx(xx,yy,zz);
                aky2=fy(xx,yy,zz);
                akz2=fz(xx,yy,zz);
                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);
                akx3=fx(xx,yy,zz);
                aky3=fy(xx,yy,zz);
                akz3=fz(xx,yy,zz);
                x=x+(akx1+4*akx2+akx3)*(h/6);
                y=y+(aky1+4*aky2+aky3)*(h/6);
                z=z+(akz1+4*akz2+akz3)*(h/6);
        printf("%.8f        %.8f        %.8f  \n  " ,x,y,z);
        }
}
```

3.1.3.5 ทำการเขียนภาษาซีในการแก้สมการเชิงอนุพันธ์โดยใช้ระเบียบวิธีของ Runge-Kutta'4th

โดยกำหนดค่าเริ่มต้น

ดังนี้ x = 0.1, y = 0, z = 0 และ h = 0.01 โดยให้ผลลัพธ์ออกมา N= 10,000ค่า

```
#include <stdio.h>
#include <math.h>
float
akx1,akx2,akx3,akx4,aky1,aky2,aky3,aky4,akz1,akz2,akz3,akz4,x,xx,y,yy,z,zz,h;
int i,n;
double c,b,a;
float fx(float x,float y,float z)
        {return - y - z;}
float fy(float x,float y,float z)
        {return x + a*y;}
float fz(float x,float y,float z)
        { return b + z*( x - c );}
void main()
{       x=0.1; y=0; z=0; n=10000; h=0.01; a=0.2; b=0.2; c=5.7;
        for (i=1; i <= n ;i++ )
        {       akx1=fx(x,y,z);
                aky1=fy(x,y,z);
                akz1=fz(x,y,z);
                xx=x+(h*akx1)/2;
                yy=y+(h*aky1)/2;
                zz=z+(h*akz1)/2;
                akx2=fx(xx,yy,zz);
                aky2=fy(xx,yy,zz);
                akz2=fz(xx,yy,zz);
                xx=x+(h*akx2)/2;
                yy=y+(h*aky2)/2;
                zz=z+(h*akz2)/2;
                akx3=fx(xx,yy,zz);
                aky3=fy(xx,yy,zz);
                akz3=fz(xx,yy,zz);
                xx=x+h*akx3;
                yy=y+h*aky3;
                zz=z+h*akz3;
                akx4=fx(xx,yy,zz);
                aky4=fy(xx,yy,yy);
                akz4=fz(xx,yy,zz);
                x=x+(akx1+2*akx2+2*akx3+akx4)*(h/6);
                y=y+(aky1+2*aky2+2*aky3+aky4)*(h/6);
                z=z+(akz1+2*akz2+2*akz3+akz4)*(h/6);
                printf("%.8f        %.8f        %.8f  \n  " ,x,y,z);
        }
}
```

3.1.4 การทดลองการประมวลผลสมการ chaotic โดยใช้ Map Logistic

ทำการเขียนโปรแกรมภาษาซีโดยใช้สมการ แม็ปลอจิสติก

โดยกำหนดค่าเริ่มต้น x =0.1 และ r = 4

```
#include "stdio.h"
#include "stdlib.h"
#include "math.h"
#define r 4
int main()
{       int i;
        float x[255];
        x[0]=0.1;
        for (i=0;i<255;i++ )
        {       printf(" %f\n",x[i]);
                x[i+1]=r*x[i]*(1-x[i]); }
}
```

3.1.5 การทดลองการประมวลผลสมการ chaotic โดยใช้ Cat Map

```
#include "stdio.h"
#include "stdlib.h"
#include "math.h"
int main()
{               int i; int x[255]; int y[255]; int z[255];
                x[0]=1; y[0]=2; z[0]=1;
                for (i=0;i<255;i++ )
                {printf("%d \n",z[i]);
                  x[i+1]=(x[i]+y[i]+z[i])%255;
                  y[i+1]=(x[i]+2*y[i]+2*z[i])%255;
                  z[i+1]=(x[i]+2*y[i]+3*z[i])%255;
                }
}
```

3.1.6 การทดลอง Multi-Chaotic แบบ สลับตัวดึงดูด

3.1.6.1 โดยสลับตัวดึงดูดทุกๆ 333 ค่า ซึ่งใช้ 3 สมการ 2 ระเบียบวิธี ดังนี้

- ใช้ สมการ Chen โดยใช้ระเบียบวิธี Euler หาค่า เริ่มต้น
- ใช้ สมการ Chen โดยใช้ระเบียบวิธี Runge-Kutta3rd หาค่ากุญแจ
- ใช้ สมการ Lorenz โดยใช้ระเบียบวิธี Runge-Kutta3rd หาค่ากุญแจ
- ใช้ สมการ Rossler โดยใช้ระเบียบวิธี Runge-Kutta3rd หาค่ากุญแจ

โดยกำหนดค่าเริ่มต้น

a=10,b=2,c=2ให้กับสมการ Chen Euler ซึ่งหาค่าที่ m=1000 จะได้ค่าเริ่มต้นx,y,z ในสมการของ Chen Runge-Kutta3rd h=0.01 โดย หากุญแจ n=10000 ค่า

```
#include <stdio.h>
#include <stdlib.h>
float buffer,akx1,akx2,akx3,aky1,aky2,aky3,akz1,akz2,akz3,x,xx,y,yy,z,zz,h,a,b,c;
float kx,ky,kz;
int  m;
unsigned long data,i,n,length;

//Chen Key
float fx(float a,float b,float c)
        {return 35*(b-a);}
float fy(float a,float b,float c)
        { return (-7)*a-(a*c)+(28*b);}
float fz(float a,float b,float c)
        { return a*b-3*c;}

//Chen Runge3rd
float Chenfx(float x,float y,float z)
        {return 35*(y-x);}
float Chenfy(float x,float y,float z)
        { return (-7)*x-(x*z)+(28*y);}
float Chenfz(float x,float y,float z)
        { return x*y-3*z;}
//Lorenz Runge3rd
float Lorenzfx(float x,float y,float z)
        {return 10*(y-x);}
float Lorenzfy(float x,float y,float z)
        {return x*(28 - z) - y ;}
float Lorenzfz(float x,float y,float z)
        {  return x*y-(8/3)*z;}
//Rossler Runge3rd
float Rosslerfx(float x,float y,float z)
        {return - y - z;}
float Rosslerfy(float x,float y,float z)
        { return x + (0.2)*y;}
float Rosslerfz(float x,float y,float z)
        { return (0.2) + z*( x - (5.7) );}
```

```
void main()
{
        data=10;
         a=data;
         b=2;
         c=2;
         m=1000;
         h=0.001;
         for (i=1; i <= m ;i++ )
         {
           kx=fx(a,b,c);
           ky=fy(a,b,c);
           kz=fz(a,b,c);
           a=a+(kx*h);
           b=b+(ky*h);
           c=c+(kz*h);
         }
         x=a;
         y=b;
         z=c;
         length=1000;
         n=length;
         buffer=0;
         h=0.01;
         for (i=1; i <= n ;i++ )
         {
               if (i % 1000 < 333)//Chen
               {
                  akx1=Chenfx(x,y,z);
                  aky1=Chenfy(x,y,z);
                  akz1=Chenfz(x,y,z);

                  xx=x+(h*akx1)/2;
                  yy=y+(h*aky1)/2;
                  zz=z+(h*akz1)/2;

                  akx2=Chenfx(xx,yy,zz);
                  aky2=Chenfy(xx,yy,zz);
                  akz2=Chenfz(xx,yy,zz);

                  xx=x-(h*akx1)+(2*h*akx2);
                  yy=y-(h*aky1)+(2*h*aky2);
                  zz=z-(h*akz2)+(2*h*akz2);

                  akx3=Chenfx(xx,yy,zz);
                  aky3=Chenfy(xx,yy,zz);
                  akz3=Chenfz(xx,yy,zz);
```

```
    x=x+(akx1+4*akx2+akx3)*(h/6);
    y=y+(aky1+4*aky2+aky3)*(h/6);
    z=z+(akz1+4*akz2+akz3)*(h/6);

  }
  if (( i % 1000 >= 333)&&(i % 1000 <666))//Lorenz
  {
    akx1=Lorenzfx(x,y,z);
    aky1=Lorenzfy(x,y,z);
    akz1=Lorenzfz(x,y,z);

    xx=x+(h*akx1)/2;
    yy=y+(h*aky1)/2;
    zz=z+(h*akz1)/2;

    akx2=Lorenzfx(xx,yy,zz);
    aky2=Lorenzfy(xx,yy,zz);
    akz2=Lorenzfz(xx,yy,zz);

    xx=x-(h*akx1)+(2*h*akx2);
    yy=y-(h*aky1)+(2*h*aky2);
    zz=z-(h*akz2)+(2*h*akz2);

    akx3=Lorenzfx(xx,yy,zz);
    aky3=Lorenzfy(xx,yy,zz);
    akz3=Lorenzfz(xx,yy,zz);

    x=x+(akx1+4*akx2+akx3)*(h/6);
    y=y+(aky1+4*aky2+aky3)*(h/6);
    z=z+(akz1+4*akz2+akz3)*(h/6);
  }
  if (i % 1000 >=666)//Rossler
  {
    akx1=Rosslerfx(x,y,z);
     aky1=Rosslerfy(x,y,z);
    akz1=Rosslerfz(x,y,z);

    xx=x+(h*akx1)/2;
    yy=y+(h*aky1)/2;
    zz=z+(h*akz1)/2;

    akx2=Rosslerfx(xx,yy,zz);
    aky2=Rosslerfy(xx,yy,zz);
    akz2=Rosslerfz(xx,yy,zz);

    xx=x-(h*akx1)+(2*h*akx2);
    yy=y-(h*aky1)+(2*h*aky2);
    zz=z-(h*akz2)+(2*h*akz2);
```

```
            akx3=Rosslerfx(xx,yy,zz);
            aky3=Rosslerfy(xx,yy,zz);
            akz3=Rosslerfz(xx,yy,zz);


            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
          }


          printf(".8%f\n",x);

  }
}
```

ความพิเศษของโปรแกรมข้างต้นคือ รับค่าเริ่มต้นมาเข้าสมการ Chen ซึ่งแก้ด้วยระเบียบวิธี Euler ซึ่งนำมาเป็นค่าเริ่มต้นใหม่ (x, y, z) ให้กับ Multi-Chaotic Attractor โดยMulti-Chaotic Attractor นี้

ถ้า I mod 1000 <332 ค่าจะเข้า สมการ Chen แก้โดยระเบียบวิธี Runge-Kutta3rd order

ถ้า 333<=I mod 1000 <666 ค่าจะเข้า สมการ Lorenz แก้โดยระเบียบวิธี Runge-Kutta3rd order

ถ้า I mod 1000 >=666 ค่าจะเข้า สมการ Rossler แก้โดยระเบียบวิธี Runge-Kutta3rd oeder

3.1.6.2 การทดลอง Multi-Chaotic แบบเปลี่ยน ตัวดึงดูด ทุกๆ 1 ค่า

จากโปรแกรมข้างต้นทำให้เกิดข้อสงสัยว่า ทำไมไม่ให้เปลี่ยน Attractor ทุกๆ 1 ค่า จึงเกิดการทดลองโปรแกรมดังนี้

โดยกำหนดค่าเริ่มต้น

a=10, b=2, c=2ให้กับสมการ Chen Euler ซึ่งหาค่าที่ m=1000 จะได้ค่าเริ่มต้นx,y,z ในสมการของ Chen Runge-Kutta3rd order h=0.01 โดย หากุญแจ n=10000 ค่า

โดย ถ้า I mod 3 <1 ค่าจะเข้า สมการ Chen แก้โดยระเบียบวิธี Runge-Kutta3rdorder

ถ้า 1<=I mod 3 <2 ค่าจะเข้า สมการ Lorenz แก้โดยระเบียบวิธี Runge-Kutta3rdorder

ถ้า I mod 3 >=2 ค่าจะเข้า สมการ Rossler แก้โดยระเบียบวิธี Runge-Kutta3rdorder

มีโปรแกรมดังนี้

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

float buffer,akx1,akx2,akx3,aky1,aky2,aky3,akz1,akz2,akz3,x,xx,y,yy,z,zz,h,a,b,c;
float kx,ky,kz;
int   ss,n;
unsigned long data,i,m,length;
//Chen Key
float fx(float a,float b,float c)
      {return 35*(b-a);}
float fy(float a,float b,float c)
      { return (-7)*a-(a*c)+(28*b);}
float fz(float a,float b,float c)
      { return a*b-3*c;}
//Chen Runge3rd
float Chenfx(float x,float y,float z)
      {return 35*(y-x);}
float Chenfy(float x,float y,float z)
      { return (-7)*x-(x*z)+(28*y);}
float Chenfz(float x,float y,float z)
      {return x*y-3*z;}
//Lorenz Runge3rd
float Lorenzfx(float x,float y,float z)
      {return 10*(y-x);}
float Lorenzfy(float x,float y,float z)
      { return x*(28 - z) - y ;}
float Lorenzfz(float x,float y,float z)
      { return x*y-(8/3)*z;}
//Rossler Runge3rd
float Rosslerfx(float x,float y,float z)
      {return - y - z;}
float Rosslerfy(float x,float y,float z)
      {return x + (0.2)*y;}
float Rosslerfz(float x,float y,float z)
      {  return (0.2) + z*( x - (5.7) );}
void main()
{
   data=10;
      a=data;
      b=2;
      c=2;
      m=1000;
      h=0.001;
      for (i=1; i <= m ;i++ )
      {
        kx=fx(a,b,c);
        ky=fy(a,b,c);
        kz=fz(a,b,c);
        a=a+(kx*h);
        b=b+(ky*h);
        c=c+(kz*h);
```

```
}
x=a;
y=b;
z=c;
length=10000;
n=length;
buffer=0;
h=0.01;
for (i=1; i <= n ;i++ )
{
        if (i % 3 < 1)//Chen
        {
          akx1=Chenfx(x,y,z);
          aky1=Chenfy(x,y,z);
          akz1=Chenfz(x,y,z);

          xx=x+(h*akx1)/2;
          yy=y+(h*aky1)/2;
          zz=z+(h*akz1)/2;

          akx2=Chenfx(xx,yy,zz);
          aky2=Chenfy(xx,yy,zz);
          akz2=Chenfz(xx,yy,zz);

          xx=x-(h*akx1)+(2*h*akx2);
          yy=y-(h*aky1)+(2*h*aky2);
          zz=z-(h*akz2)+(2*h*akz2);

          akx3=Chenfx(xx,yy,zz);
          aky3=Chenfy(xx,yy,zz);
          akz3=Chenfz(xx,yy,zz);

          x=x+(akx1+4*akx2+akx3)*(h/6);
          y=y+(aky1+4*aky2+aky3)*(h/6);
          z=z+(akz1+4*akz2+akz3)*(h/6);

        }
        if (( i % 3 >= 1)&&(i % 3 <2))//Lorenz
        {
          akx1=Lorenzfx(x,y,z);
          aky1=Lorenzfy(x,y,z);
          akz1=Lorenzfz(x,y,z);

          xx=x+(h*akx1)/2;
          yy=y+(h*aky1)/2;
          zz=z+(h*akz1)/2;

          akx2=Lorenzfx(xx,yy,zz);
          aky2=Lorenzfy(xx,yy,zz);
          akz2=Lorenzfz(xx,yy,zz);
```

```
            xx=x-(h*akx1)+(2*h*akx2);
            yy=y-(h*aky1)+(2*h*aky2);
            zz=z-(h*akz2)+(2*h*akz2);

            akx3=Lorenzfx(xx,yy,zz);
            aky3=Lorenzfy(xx,yy,zz);
            akz3=Lorenzfz(xx,yy,zz);

            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
          }
          if (i % 3 >=2)//Rossler
          {
            akx1=Rosslerfx(x,y,z);
             aky1=Rosslerfy(x,y,z);
            akz1=Rosslerfz(x,y,z);

            xx=x+(h*akx1)/2;
            yy=y+(h*aky1)/2;
            zz=z+(h*akz1)/2;

            akx2=Rosslerfx(xx,yy,zz);
            aky2=Rosslerfy(xx,yy,zz);
            akz2=Rosslerfz(xx,yy,zz);

            xx=x-(h*akx1)+(2*h*akx2);
            yy=y-(h*aky1)+(2*h*aky2);
            zz=z-(h*akz2)+(2*h*akz2);

            akx3=Rosslerfx(xx,yy,zz);
            aky3=Rosslerfy(xx,yy,zz);
            akz3=Rosslerfz(xx,yy,zz);

            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
          }

          printf("%.8f\n",z);
    }

}
```

3.1.7 การทดลอง Multi-Chaotic แบบ ขึ้นอยู่กับค่าเริ่มต้น ซึ่งใช้ 3 สมการ 2 ระเบียบวิธี

- ใช้ สมการ Chen โดยใช้ระเบียบวิธี Euler หาค่า เริ่มต้น
- ใช้ สมการ Chen โดยใช้ระเบียบวิธี Runge-Kutta3rd หาค่ากุญแจ
- ใช้ สมการ Lorenz โดยใช้ระเบียบวิธี Runge-Kutta3rd หาค่ากุญแจ
- ใช้ สมการ Rossler โดยใช้ระเบียบวิธี Runge-Kutta3rd หาค่ากุญแจ

โดยกำหนดค่าเริ่มต้น

a=10, b=2, c=2ให้กับสมการ Chen Euler ซึ่งหาค่าที่ m=1000 จะได้ค่าเริ่มต้นx, y, z ในสมการเริ่มต้นตามค่า เริ่มต้นที่นำมาหารเอาเศษ และค่า h=0.01 โดย หากุญแจ n=10000 ค่า

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

float buffer,akx1,akx2,akx3,aky1,aky2,aky3,akz1,akz2,akz3,x,xx,y,yy,z,zz,h,a,b,c;
float kx,ky,kz;
int  ss,n;
unsigned long data,i,m,length;
//Chen Key
float fx(float a,float b,float c)
      {return 35*(b-a);}
float fy(float a,float b,float c)
      {return (-7)*a-(a*c)+(28*b);}
float fz(float a,float b,float c)
      {return a*b-3*c;}
//Chen Runge3rd
float Chenfx(float x,float y,float z)
      {return 35*(y-x);}
float Chenfy(float x,float y,float z)
      {return (-7)*x-(x*z)+(28*y);}
float Chenfz(float x,float y,float z)
      { return x*y-3*z;}
//Lorenz Runge3rd
float Lorenzfx(float x,float y,float z)
      {return 10*(y-x);}
float Lorenzfy(float x,float y,float z)
      { return x*(28 - z) - y ;}
float Lorenzfz(float x,float y,float z)
      { return x*y-(8/3)*z;}
//Rossler Runge3rd
float Rosslerfx(float x,float y,float z)
      {return - y - z;}
float Rosslerfy(float x,float y,float z)
```

```
        { return x + (0.2)*y;}
float Rosslerfz(float x,float y,float z)
        { return (0.2) + z*( x - (5.7) );}
void main()
{
        data=10;
        a=data;
        b=2;
        c=2;
        m=1000;
        h=0.001;
        for (i=1; i <= m ;i++ )
        {
          kx=fx(a,b,c);
          ky=fy(a,b,c);
          kz=fz(a,b,c);
          a=a+(kx*h);
         b=b+(ky*h);
         c=c+(kz*h);
        }
        x=a;
        y=b;
        z=c;
        length=10000;
        n=length;
        buffer=0;
        h=0.01;
        switch (data % 6)
{
        case 0://Chen->Lorenz->Rossler
                for (i=1; i <= n ;i++ )
                  {
                  if (i % 1000 < 333) Chen
                          {
                          akx1=Chenfx(x,y,z);
                          aky1=Chenfy(x,y,z);
                          akz1=Chenfz(x,y,z);

                          xx=x+(h*akx1)/2;
                          yy=y+(h*aky1)/2;
                          zz=z+(h*akz1)/2;

                          akx2=Chenfx(xx,yy,zz);
                          aky2=Chenfy(xx,yy,zz);
                          akz2=Chenfz(xx,yy,zz);
```

```
            xx=x-(h*akx1)+(2*h*akx2);
            yy=y-(h*aky1)+(2*h*aky2);
            zz=z-(h*akz2)+(2*h*akz2);

            akx3=Chenfx(xx,yy,zz);
            aky3=Chenfy(xx,yy,zz);
            akz3=Chenfz(xx,yy,zz);

            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
}
if (( i % 1000 >= 333 )&&(i % 1000 <666)) Lorenz
{
            akx1=Lorenzfx(x,y,z);
            aky1=Lorenzfy(x,y,z);
            akz1=Lorenzfz(x,y,z);

            xx=x+(h*akx1)/2;
            yy=y+(h*aky1)/2;
            zz=z+(h*akz1)/2;

            akx2=Lorenzfx(xx,yy,zz);
            aky2=Lorenzfy(xx,yy,zz);
            akz2=Lorenzfz(xx,yy,zz);

            xx=x-(h*akx1)+(2*h*akx2);
            yy=y-(h*aky1)+(2*h*aky2);
            zz=z-(h*akz2)+(2*h*akz2);

            akx3=Lorenzfx(xx,yy,zz);
            aky3=Lorenzfy(xx,yy,zz);
            akz3=Lorenzfz(xx,yy,zz);

            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
}
if (i % 1000 >=666) Rossler
{
            akx1=Rosslerfx(x,y,z);
            aky1=Rosslerfy(x,y,z);
```

```
                akz1=Rosslerfz(x,y,z);

                xx=x+(h*akx1)/2;
                yy=y+(h*aky1)/2;
                zz=z+(h*akz1)/2;

                akx2=Rosslerfx(xx,yy,zz);
                aky2=Rosslerfy(xx,yy,zz);
                akz2=Rosslerfz(xx,yy,zz);

                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);

                akx3=Rosslerfx(xx,yy,zz);
                aky3=Rosslerfy(xx,yy,zz);
                akz3=Rosslerfz(xx,yy,zz);

                x=x+(akx1+4*akx2+akx3)*(h/6);
                y=y+(aky1+4*aky2+aky3)*(h/6);
                z=z+(akz1+4*akz2+akz3)*(h/6);
        }
        printf("%.8f\n",x);
    }
        break;
case 1: //Rossler->Chen->Lorenz
        for (i=1; i <= n ;i++ )
{
        if (i % 1000 < 333)//Rossler
        {
                akx1=Rosslerfx(x,y,z);
                aky1=Rosslerfy(x,y,z);
                akz1=Rosslerfz(x,y,z);

                xx=x+(h*akx1)/2;
                yy=y+(h*aky1)/2;
                zz=z+(h*akz1)/2;

                akx2=Rosslerfx(xx,yy,zz);
                aky2=Rosslerfy(xx,yy,zz);
                akz2=Rosslerfz(xx,yy,zz);

                xx=x-(h*akx1)+(2*h*akx2);
```

```
            yy=y-(h*aky1)+(2*h*aky2);
            zz=z-(h*akz2)+(2*h*akz2);

            akx3=Rosslerfx(xx,yy,zz);
            aky3=Rosslerfy(xx,yy,zz);
            akz3=Rosslerfz(xx,yy,zz);

            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
}
if (( i % 1000 >= 333 )&&(i % 1000 <666)) Chen
{
            akx1=Chenfx(x,y,z);
            aky1=Chenfy(x,y,z);
            akz1=Chenfz(x,y,z);

            xx=x+(h*akx1)/2;
            yy=y+(h*aky1)/2;
            zz=z+(h*akz1)/2;

            akx2=Chenfx(xx,yy,zz);
            aky2=Chenfy(xx,yy,zz);
            akz2=Chenfz(xx,yy,zz);

            xx=x-(h*akx1)+(2*h*akx2);
            yy=y-(h*aky1)+(2*h*aky2);
            zz=z-(h*akz2)+(2*h*akz2);

            akx3=Chenfx(xx,yy,zz);
            aky3=Chenfy(xx,yy,zz);
            akz3=Chenfz(xx,yy,zz);

            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
}
if (i % 1000 >=666) Lorenz
{
            akx1=Lorenzfx(x,y,z);
            aky1=Lorenzfy(x,y,z);
            akz1=Lorenzfz(x,y,z);
```

```
                xx=x+(h*akx1)/2;
                yy=y+(h*aky1)/2;
                zz=z+(h*akz1)/2;

                akx2=Lorenzfx(xx,yy,zz);
                aky2=Lorenzfy(xx,yy,zz);
                akz2=Lorenzfz(xx,yy,zz);

                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);

                akx3=Lorenzfx(xx,yy,zz);
                aky3=Lorenzfy(xx,yy,zz);
                akz3=Lorenzfz(xx,yy,zz);

                x=x+(akx1+4*akx2+akx3)*(h/6);
                y=y+(aky1+4*aky2+aky3)*(h/6);
                z=z+(akz1+4*akz2+akz3)*(h/6);
        }
        printf("%.8f\n",x);
}
        break;
case 2: Lorenz -> Chen -> Rossler
        for (i=1; i <= n ;i++ )
{
        if (i % 1000 < 383) Lorenz
        {
                akx1=Lorenzfx(x,y,z);
                aky1=Lorenzfy(x,y,z);
                akz1=Lorenzfz(x,y,z);

                xx=x+(h*akx1)/2;
                yy=y+(h*aky1)/2;
                zz=z+(h*akz1)/2;

                akx2=Lorenzfx(xx,yy,zz);
                aky2=Lorenzfy(xx,yy,zz);
                akz2=Lorenzfz(xx,yy,zz);

                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);
```

```
            akx3=Lorenzfx(xx,yy,zz);
            aky3=Lorenzfy(xx,yy,zz);
            akz3=Lorenzfz(xx,yy,zz);

            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
    }
if (( i % 1000 >= 333 )&&(i % 1000 <666)) Chen
{
            akx1=Chenfx(x,y,z);
            aky1=Chenfy(x,y,z);
            akz1=Chenfz(x,y,z);

            xx=x+(h*akx1)/2;
            yy=y+(h*aky1)/2;
            zz=z+(h*akz1)/2;

            akx2=Chenfx(xx,yy,zz);
            aky2=Chenfy(xx,yy,zz);
            akz2=Chenfz(xx,yy,zz);

            xx=x-(h*akx1)+(2*h*akx2);
            yy=y-(h*aky1)+(2*h*aky2);
            zz=z-(h*akz2)+(2*h*akz2);

            akx3=Chenfx(xx,yy,zz);
            aky3=Chenfy(xx,yy,zz);
            akz3=Chenfz(xx,yy,zz);

            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
}
if (i % 1000 >=666) Rossler
{
            akx1=Rosslerfx(x,y,z);
            aky1=Rosslerfy(x,y,z);
            akz1=Rosslerfz(x,y,z);

            xx=x+(h*akx1)/2;
            yy=y+(h*aky1)/2;
            zz=z+(h*akz1)/2;
```

```
                akx2=Rosslerfx(xx,yy,zz);
                aky2=Rosslerfy(xx,yy,zz);
                akz2=Rosslerfz(xx,yy,zz);

                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);

                akx3=Rosslerfx(xx,yy,zz);
                aky3=Rosslerfy(xx,yy,zz);
                akz3=Rosslerfz(xx,yy,zz);

                x=x+(akx1+4*akx2+akx3)*(h/6);
                y=y+(aky1+4*aky2+aky3)*(h/6);
                z=z+(akz1+4*akz2+akz3)*(h/6);
        }
        printf("%.8f\n",x);
}
        break;
case 3: Lorenz->Rossler->Chen
        for (i=1; i <= n ;i++ )
{
        if (i % 1000 < 333) Lorenz
        {
                akx1=Lorenzfx(x,y,z);
                aky1=Lorenzfy(x,y,z);
                akz1=Lorenzfz(x,y,z);

                xx=x+(h*akx1)/2;
                yy=y+(h*aky1)/2;
                zz=z+(h*akz1)/2;

                akx2=Lorenzfx(xx,yy,zz);
                aky2=Lorenzfy(xx,yy,zz);
                akz2=Lorenzfz(xx,yy,zz);

                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);

                akx3=Lorenzfx(xx,yy,zz);
                aky3=Lorenzfy(xx,yy,zz);
                akz3=Lorenzfz(xx,yy,zz);
```

```
            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
}
if (( i % 1000 >= 333 )&&(i % 1000 <666)) //Rossler
{
            akx1=Rosslerfx(x,y,z);
            aky1=Rosslerfy(x,y,z);
            akz1=Rosslerfz(x,y,z);

            xx=x+(h*akx1)/2;
            yy=y+(h*aky1)/2;
            zz=z+(h*akz1)/2;

            akx2=Rosslerfx(xx,yy,zz);
            aky2=Rosslerfy(xx,yy,zz);
            akz2=Rosslerfz(xx,yy,zz);

            xx=x-(h*akx1)+(2*h*akx2);
            yy=y-(h*aky1)+(2*h*aky2);
            zz=z-(h*akz2)+(2*h*akz2);

            akx3=Rosslerfx(xx,yy,zz);
            aky3=Rosslerfy(xx,yy,zz);
            akz3=Rosslerfz(xx,yy,zz);

            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
  }
if (i % 1000 >=666) //Chen
{
            akx1=Chenfx(x,y,z);
            aky1=Chenfy(x,y,z);
            akz1=Chenfz(x,y,z);

            xx=x+(h*akx1)/2;
            yy=y+(h*aky1)/2;
            zz=z+(h*akz1)/2;

            akx2=Chenfx(xx,yy,zz);
            aky2=Chenfy(xx,yy,zz);
```

```
                akz2=Chenfz(xx,yy,zz);

                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);

                akx3=Chenfx(xx,yy,zz);
                aky3=Chenfy(xx,yy,zz);
                akz3=Chenfz(xx,yy,zz);

                x=x+(akx1+4*akx2+akx3)*(h/6);
                y=y+(aky1+4*aky2+aky3)*(h/6);
                z=z+(akz1+4*akz2+akz3)*(h/6);
        }
          printf("%.8f\n",x);
}
        break;
case 4: Rossler->Lorenz->Chen
        for (i=1; i <= n ;i++ )
{
        if (i % 1000 < 333) //Rossler
        {
                akx1=Rosslerfx(x,y,z);
                aky1=Rosslerfy(x,y,z);
                akz1=Rosslerfz(x,y,z);

                xx=x+(h*akx1)/2;
                yy=y+(h*aky1)/2;
                zz=z+(h*akz1)/2;

                akx2=Rosslerfx(xx,yy,zz);
                aky2=Rosslerfy(xx,yy,zz);
                akz2=Rosslerfz(xx,yy,zz);

                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);

                akx3=Rosslerfx(xx,yy,zz);
                aky3=Rosslerfy(xx,yy,zz);
                akz3=Rosslerfz(xx,yy,zz);

                x=x+(akx1+4*akx2+akx3)*(h/6);
```

```
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
}
if (( i % 1000 >= 333 )&&(i % 1000 <666)) //Lorenz
{
            akx1=Lorenzfx(x,y,z);
            aky1=Lorenzfy(x,y,z);
            akz1=Lorenzfz(x,y,z);

            xx=x+(h*akx1)/2;
            yy=y+(h*aky1)/2;
            zz=z+(h*akz1)/2;

            akx2=Lorenzfx(xx,yy,zz);
            aky2=Lorenzfy(xx,yy,zz);
            akz2=Lorenzfz(xx,yy,zz);

            xx=x-(h*akx1)+(2*h*akx2);
            yy=y-(h*aky1)+(2*h*aky2);
            zz=z-(h*akz2)+(2*h*akz2);

            akx3=Lorenzfx(xx,yy,zz);
            aky3=Lorenzfy(xx,yy,zz);
            akz3=Lorenzfz(xx,yy,zz);

            x=x+(akx1+4*akx2+akx3)*(h/6);
            y=y+(aky1+4*aky2+aky3)*(h/6);
            z=z+(akz1+4*akz2+akz3)*(h/6);
}
if (i % 1000 >=666) //Chen
{
            akx1=Chenfx(x,y,z);
            aky1=Chenfy(x,y,z);
            akz1=Chenfz(x,y,z);

            xx=x+(h*akx1)/2;
            yy=y+(h*aky1)/2;
            zz=z+(h*akz1)/2;

            akx2=Chenfx(xx,yy,zz);
            aky2=Chenfy(xx,yy,zz);
            akz2=Chenfz(xx,yy,zz);
```

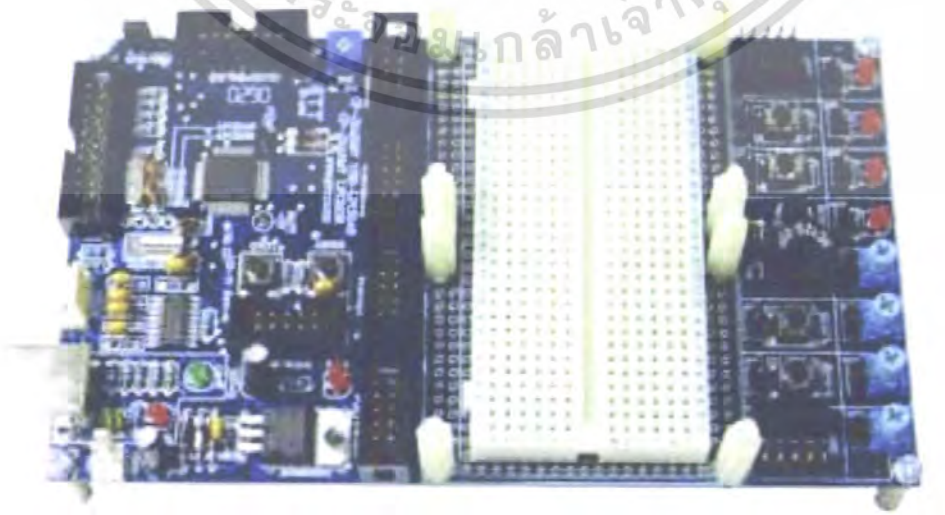
```
                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);

                akx3=Chenfx(xx,yy,zz);
                aky3=Chenfy(xx,yy,zz);
                akz3=Chenfz(xx,yy,zz);

                x=x+(akx1+4*akx2+akx3)*(h/6);
                y=y+(aky1+4*aky2+aky3)*(h/6);
                z=z+(akz1+4*akz2+akz3)*(h/6);
        }
        printf("%.8f\n",x);
}
        break;
case 5: //Chen-->Rossler-->Lorenz
        for (i=1; i <= n ;i++ )
{
        if (i % 1000 < 333)  //Chen
        {
                akx1=Chenfx(x,y,z);
                aky1=Chenfy(x,y,z);
                akz1=Chenfz(x,y,z);

                xx=x+(h*akx1)/2;
                yy=y+(h*aky1)/2;
                zz=z+(h*akz1)/2;

                akx2=Chenfx(xx,yy,zz);
                aky2=Chenfy(xx,yy,zz);
                akz2=Chenfz(xx,yy,zz);

                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);

                akx3=Chenfx(xx,yy,zz);
                aky3=Chenfy(xx,yy,zz);
                akz3=Chenfz(xx,yy,zz);

                x=x+(akx1+4*akx2+akx3)*(h/6);
                y=y+(aky1+4*aky2+aky3)*(h/6);
                z=z+(akz1+4*akz2+akz3)*(h/6);
```

```
}
if (( i % 1000 >= 333 )&&(i % 1000 <666)) //Rossler
{
        akx1=Rosslerfx(x,y,z);
        aky1=Rosslerfy(x,y,z);
        akz1=Rosslerfz(x,y,z);

        xx=x+(h*akx1)/2;
        yy=y+(h*aky1)/2;
        zz=z+(h*akz1)/2;

        akx2=Rosslerfx(xx,yy,zz);
        aky2=Rosslerfy(xx,yy,zz);
        akz2=Rosslerfz(xx,yy,zz);

        xx=x-(h*akx1)+(2*h*akx2);
        yy=y-(h*aky1)+(2*h*aky2);
        zz=z-(h*akz2)+(2*h*akz2);

        akx3=Rosslerfx(xx,yy,zz);
        aky3=Rosslerfy(xx,yy,zz);
        akz3=Rosslerfz(xx,yy,zz);

        x=x+(akx1+4*akx2+akx3)*(h/6);
        y=y+(aky1+4*aky2+aky3)*(h/6);
        z=z+(akz1+4*akz2+akz3)*(h/6);
}
if (i % 1000 >=666)//Lorenz
{
        akx1=Lorenzfx(x,y,z);
        aky1=Lorenzfy(x,y,z);
        akz1=Lorenzfz(x,y,z);

        xx=x+(h*akx1)/2;
        yy=y+(h*aky1)/2;
        zz=z+(h*akz1)/2;

        akx2=Lorenzfx(xx,yy,zz);
        aky2=Lorenzfy(xx,yy,zz);
        akz2=Lorenzfz(xx,yy,zz);

        xx=x-(h*akx1)+(2*h*akx2);
```

```
                    yy=y-(h*aky1)+(2*h*aky2);
                    zz=z-(h*akz2)+(2*h*akz2);

                    akx3=Lorenzfx(xx,yy,zz);
                    aky3=Lorenzfy(xx,yy,zz);
                    akz3=Lorenzfz(xx,yy,zz);

                    x=x+(akx1+4*akx2+akx3)*(h/6);
                    y=y+(aky1+4*aky2+aky3)*(h/6);
                    z=z+(akz1+4*akz2+akz3)*(h/6);
            }
            printf("%.8f\n",x);
        }
            break;
    }
}
```

3.2 ออกแบบโปรแกรมในส่วนของ ARM7 โดยนำโปรแกรมหาค่าคุณแจที่ได้มาประยุกต์ใช้ร่วมกัน

โดยเลือกใช้ Microcontroller ARM7 LPC 2148 ของ บริษัทอีทีที ซึ่งมีคุณสมบัติและรูปแบบดังนี้สามารถ RUN ได้ทั้งในแบบ 16 BIT และ 32 BIT ออกแบบบอร์ดให้ใช้ในงานควบคุมต่างๆ สามารถ IN-CIRCUIT DOWNLOAD โปรแกรมเข้าหน่วยความจำภายในได้โดยตรง ทาง PORT RS232

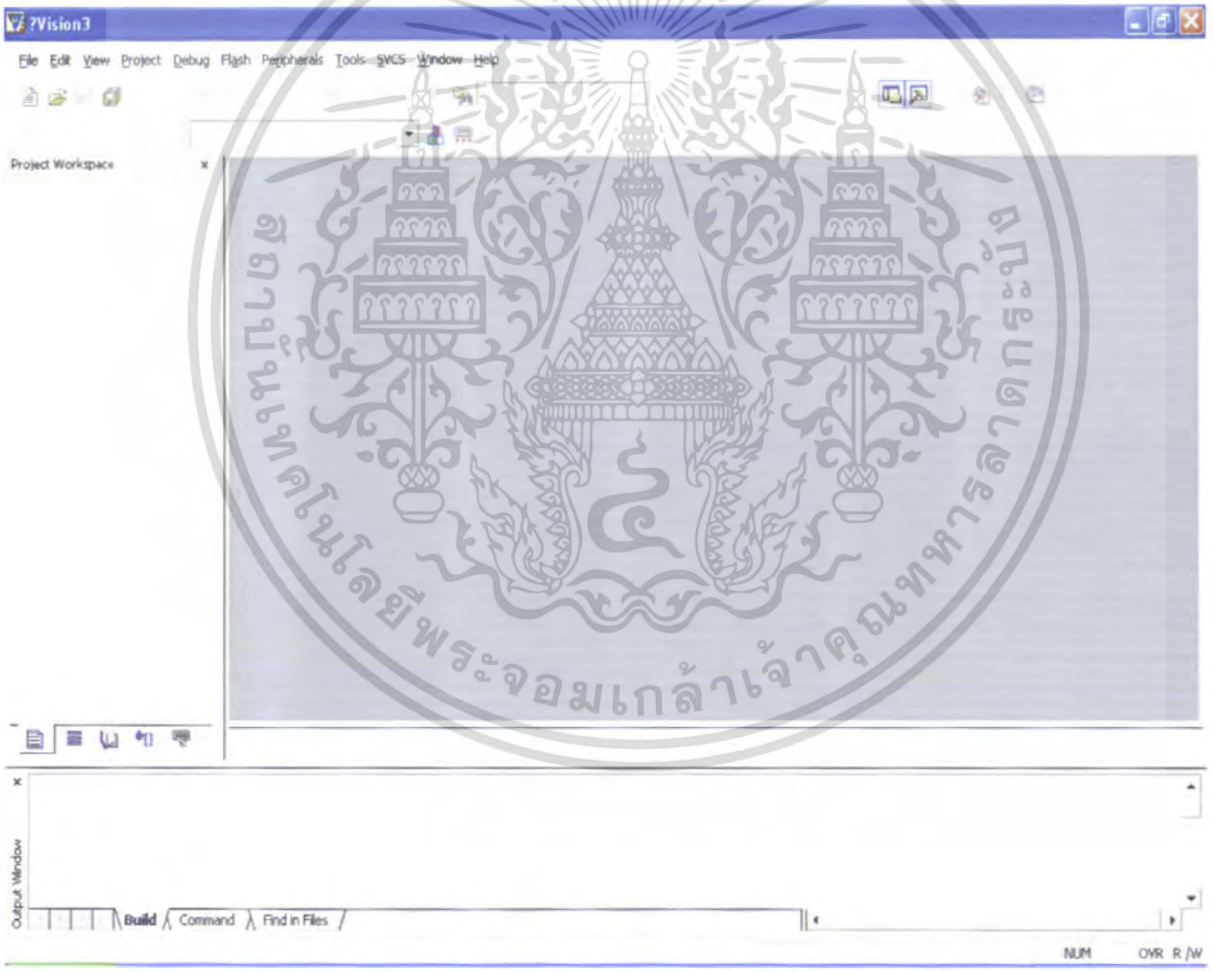


รูปที่ 3-2 รูป Microcontroller ARM7 LPC 2148

3.2.1 เลือกใช้โปรแกรม Keil uVision3

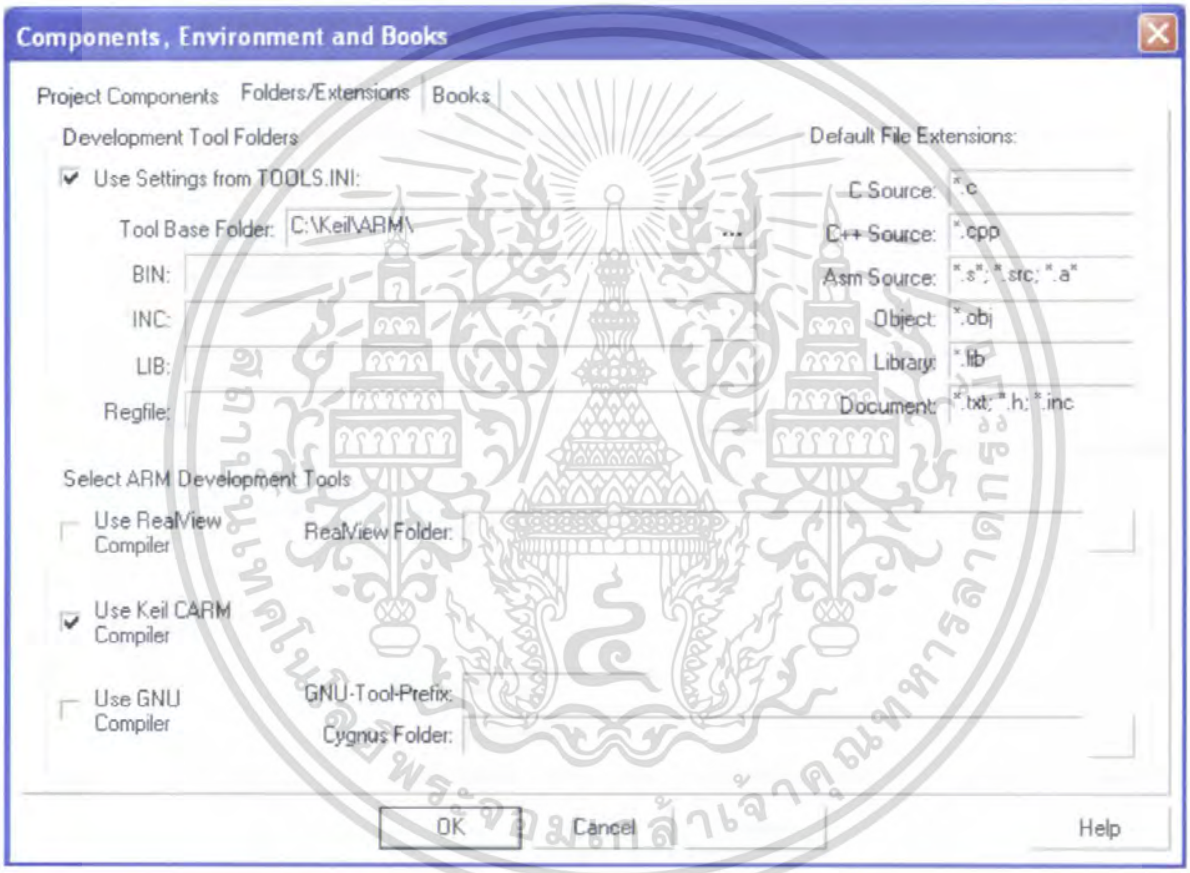
ตัวอย่างการใช้ Keil uVision3 ในการสร้าง Project File ของ Keil ARM ในที่นี้จะขอแสดงแนวทางการเขียนโปรแกรมภาษาซี โดยใช้ Keil-CARM ในการแปลคำสั่ง ภายใต้โปรแกรม Text Editor ของ Keil (Keil uVision3) โดยจะขออธิบายถึงเฉพาะวิธีการกำหนดค่า Option สำหรับ เชื่อมโยงคำสั่งในการสั่งแปลโปรแกรมด้วย Keil-CARM ผ่านทาง Keil uVision3 เท่านั้น ส่วนรายละเอียดคำสั่งและ การใช้งานฟังก์ชั่นต่างๆในการเขียนโปรแกรมของ Keil-CARM นั้นขอให้ผู้ใช้ศึกษาจากคู่มือคำสั่งของ Keil-C ARM เอง โดยวิธีการกำหนดค่าตัวเลือกของ Keil uVision3 ให้ใช้งานกับ Keil-CARM นั้นมีขั้นตอนพอสังเขปดังนี้คือ

1. เปิดโปรแกรม Keil uVision3 ซึ่งเป็นโปรแกรม Text Editor ของ Keil-CARM ใช้สำหรับใช้ในการเขียนโปรแกรมที่เป็น Source Code ภาษาซี โดยจะมีลักษณะดังรูป



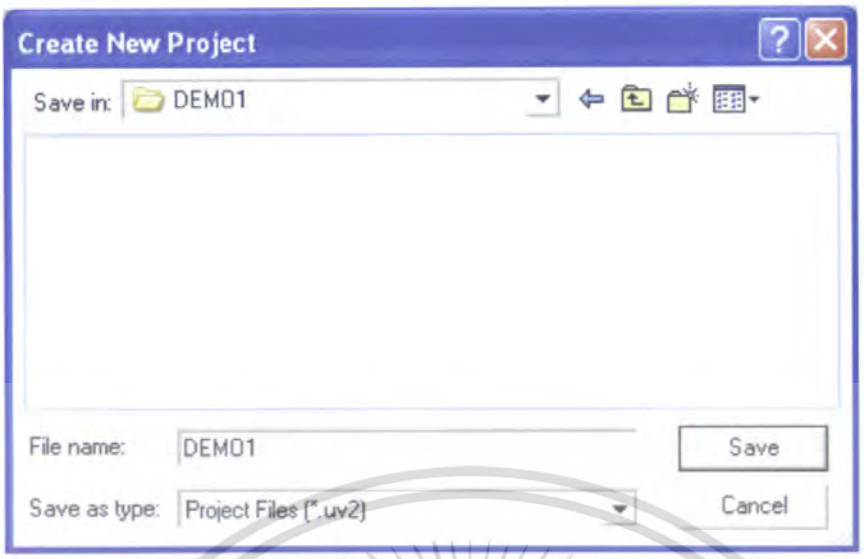
รูปที่ 3-3 รูปแบบโปรแกรม Keil uVision3

2. ทำการกำหนดค่าตัวเลือกในการแปลคำสั่งของ uVision3 ให้ใช้งานกับโปรแกรม Keil uVision3 และ Keil-CARM โดยให้เลือกคลิกเมาส์ที่เมนูคำสั่ง Project ⟶ Components, Environment, Books… จากนั้นให้เลือกค่าตัวเลือก สำหรับกำหนดการใช้งาน Complier จากหัวข้อ Select ARM Development Tools ซึ่งจะมีค่าตัวเลือกอยู่ 3 แบบ
คือ Use Keil-CARM Tools ,Use GNU Tools และ Use ARM Tools โดยให้เลือกเป็น "Use Keil ARM Tools" จากนั้นให้ทำการกำหนดตำแหน่ง Folder สำหรับเก็บค่าตัวเลือกการทำงานของโปรแกรม Keil ARM ซึ่งตามปกติ แล้วจะเป็น "C:\Keil\ARM\" แต่ถ้าติดตั้ง Keil ไว้ที่อื่นก็ต้องปรับเปลี่ยนให้ถูกต้องตามความเป็นจริงด้วยดังรูป



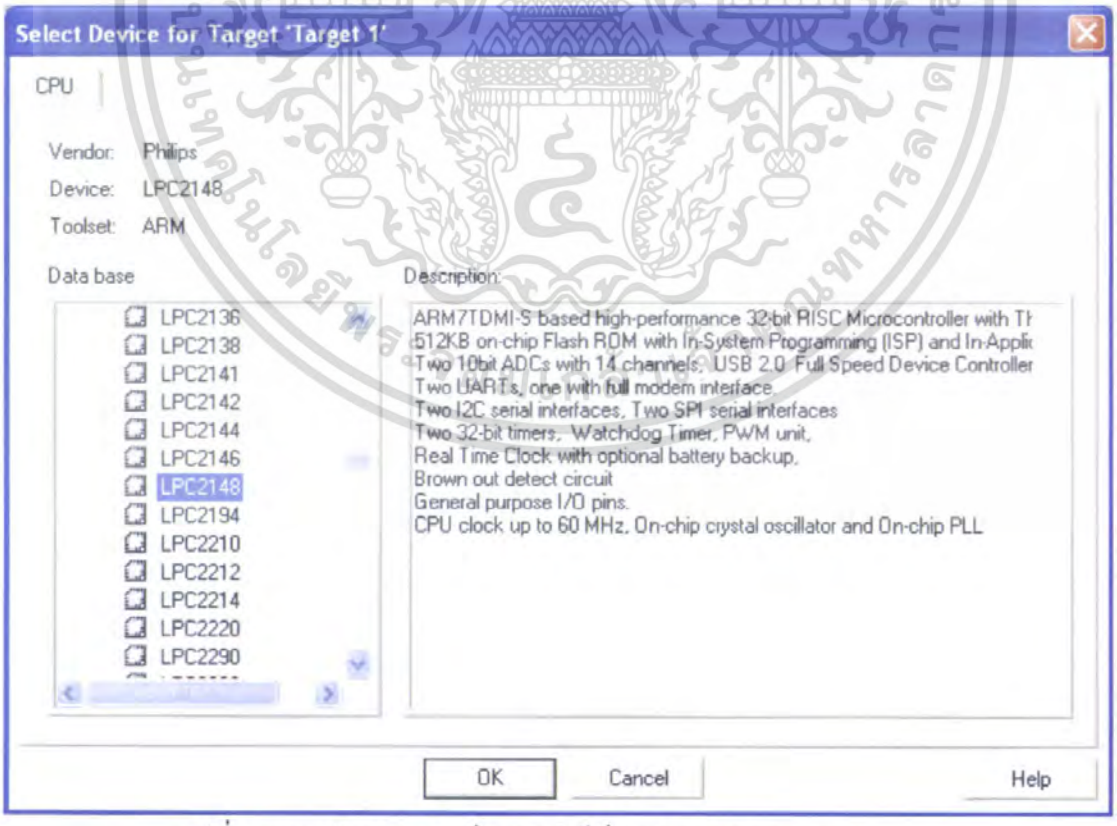
รูปที่ 3-4 การกำหนดค่าตัวเลือกการแปลคำสั่งของโปรแกรม Keil uVision3

3. ทำการสร้าง Project File ขึ้นมาใหม่ โดยเรียกเมนูคำสั่ง Project ⟶ New Project จากนั้นให้เลือกกำหนดหรือ สร้างตำแหน่ง Folder ที่จะบันทึก Project File พร้อมกับกำหนดชื่อ Project File ตามต้องการ เช่น ถ้าต้องการสร้าง Project File ชื่อ DEMO1 โดยเก็บไว้ใน Folder ชื่อ DEMO1 ก็สามารถกำหนดตำแหน่ง Folder และชื่อ Project File ได้เอง โดยเมื่อกำหนดชื่อในช่อง File name แล้วให้เลือก Save เพื่อบันทึก Project File ไว้ ดังรูป



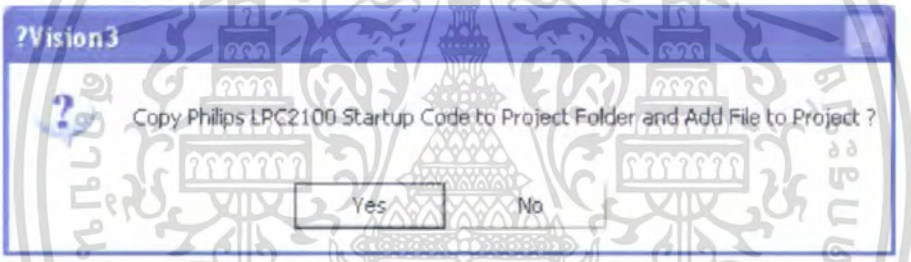
รูปที่ 3-5 การสร้างโปรเจคใหม่

หลังจากกำหนดชื่อและสั่ง Save Project File แล้ว โปรแกรมจะรอให้ผู้ใช้ทำการกำหนด เบอร์ MCU ที่จะใช้งานใน Project ที่สั่ง Save นั้น ซึ่งในกรณีที่ใช้งานกับบอร์ด CP-JR ARM7 USB-LPC2148 นั้น ให้เลือกกำหนดเบอร์ของ MCU เป็น LPC2148 ของ Philips แล้วเลือก OK ดังรูป



รูปที่ 3-6 การกำหนด เบอร์ MCU ให้เป็น LPC2148 ของ Philips

หลังจากเลือกกำหนดเบอร์ของ MCU เป็นที่เรียบร้อยแล้ว ในขั้นตอนนี้โปรแกรมจะรอให้ผู้ใช้ยืนยันว่า ต้องการจะทำการ Copy ไฟล์ Startup ของ Keil เพื่อใช้งานกับ MCU ของ Philips มาใช้ใน Project ด้วยหรือไม่ โดยStartup ไฟล์จะเป็นส่วนของการกำหนดค่าเริ่มต้นการทำงานให้กับ MCU เช่น การกำหนดค่า Stack และการ กำหนดค่าการทำงานให้กับ Phase-Lock-Loop ต่างๆ ก่อนที่จะเริ่มต้นทำงานตามโปรแกรมที่เราเขียนขึ้น ไม่เช่นนั้นแล้วโปรแกรมที่เราเขียนขึ้นมานั้นจะต้องเพิ่มคำสั่งในการเตรียมการทำงานส่วนเหล่านี้ให้ MCU เองทั้งหมดแต่เนื่องจากไฟล์ Startup ของ Keil-ARM นั้น เป็นไฟล์ภาษาแอสแซมบลี้ ซึ่งกำหนดค่าการทำงานไว้กับชุดพัฒนาของ Keil เอง ดังนั้นข้อกำหนดและการกำหนดค่าบางอย่างจะมีความแตกต่างกันอยู่กับค่าที่ต้องการสำหรับบอร์ด "CP-JR ARM7 USB-LPC2148" ไม่สามารถใช้งานไฟล์ Startup ได้ทันที ต้องมีการแก้ไขค่าตัวเลือกใหม่ดังนั้นก่อนที่จะใช้โปรแกรม Keil-CARM ในการแปลคำสั่งให้นั้น ผู้ใช้จะต้องเข้าไปแก้ไขไฟล์ Startup ใหม่โดยต้องกำหนดรูปแบบให้ถูกต้องตรงกับความต้องการของบอร์ดด้วย ดังนั้นในที่นี้ขอแนะนำให้เลือก "No" เพื่อไม่ให้ Keil uVision3 ทำการ Copy ไฟล์ Startup ของ Keil-CARM มาใช้ใน Project นี้ด้วย

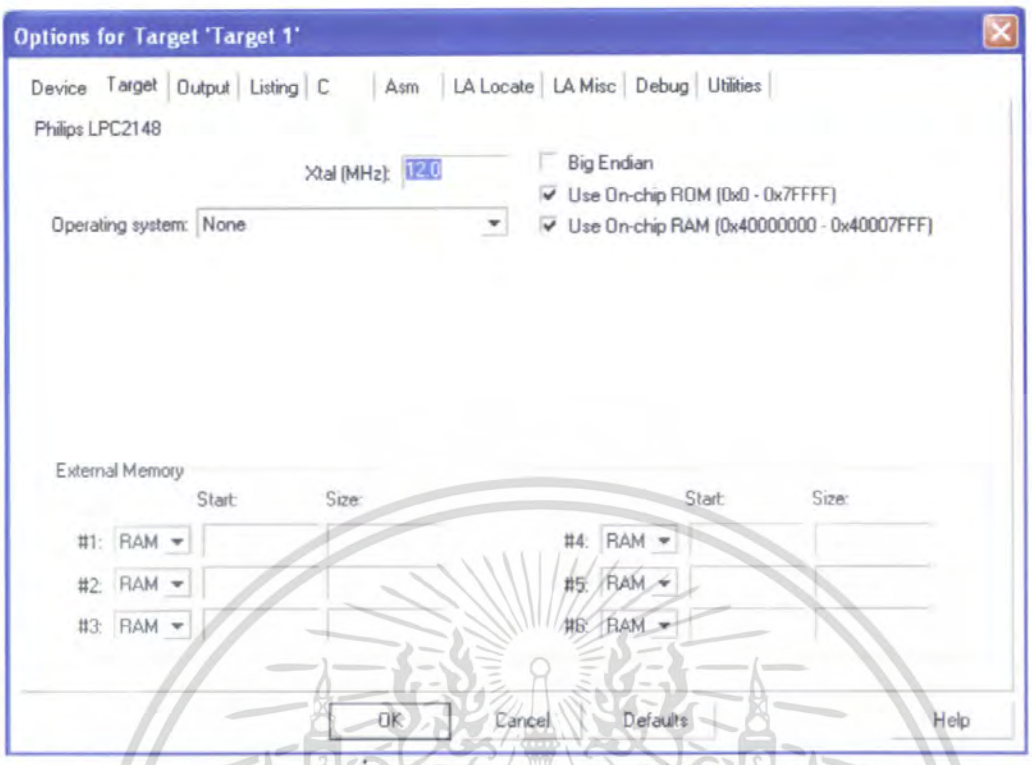


รูปที่ 3-7 ขั้นตอนการก๊อปไฟล์ Startup ของ Keil

4. ให้ทำการ Copy File ชื่อ "Startup.s" ที่ทาง อีทีที จัดเตรียมไว้ใน CD-ROM ซึ่งเก็บไว้ใน Example ชื่อ "Startup.s" มาไว้ในตำแหน่ง Folder เดียวกันกับ Project File ที่สร้างขึ้นมาใหม่นี้ โดยไฟล์ "Startup.s" จะเป็นไฟล์ซึ่งบรรจุคำสั่งภาษาแอสแซมบลี้ของ ARM7 สำหรับทำหน้าที่กำหนดค่า เริ่มต้นการทำงานที่จำเป็นให้กับ MCU ซึ่งได้แก่การ กำหนดค่า Stack ให้กับ MCU การ Initial Phase-Lock-Loop การกำหนดค่าให้กับ MAM Function และการกำหนดตำแหน่ง Vector ต่างๆของ MCU สำหรับใช้งานร่วมกับบอร์ด "CP-JR ARM7 USB-LPC2148" ซึ่งถ้าสั่ง Add ไฟล์ "Startup.s" จาก Keil หรือ Copy ไฟล์ดังกล่าวมาจากแหล่ง อื่นๆ อาจมีการทำงานของโปรแกรมใน Startup ไม่เหมือนกัน ซึ่งจะส่งผลต่อการทำงานของโปรแกรมด้วย

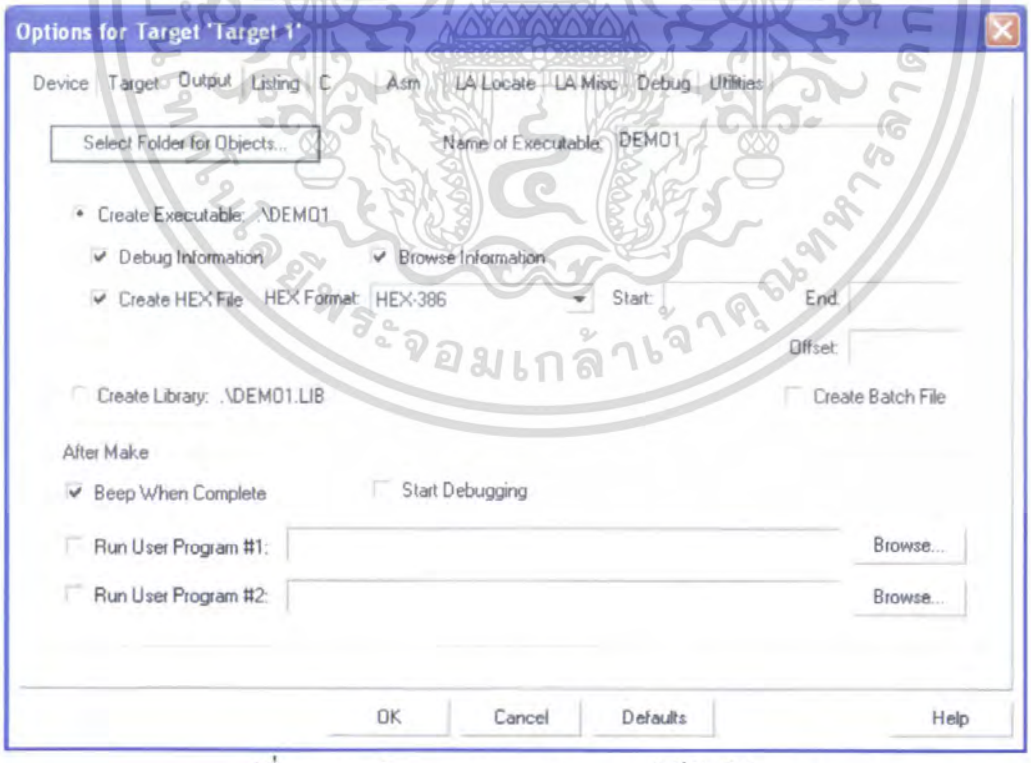
5. ให้ทำการกำหนดค่า Option ของ Project File โดยเลือกเมนูคำสั่ง Project ⟶ Option for Target 'Target 1' จากนั้นเลือกที่ Tab ของ Target เพื่อกำหนดค่าของ MCU Target โดยให้กำหนดดังนี้

5.1 X-TAL ให้กำหนดเป็น 12 MHz พร้อมกับเลือกกำหนดให้ใช้หน่วยความจำที่มีอยู่ใน MCU เป็นเงื่อนไขในการแปลโปรแกรมของ Keil-CARM ด้วย ดังรูป



รูปที่ 3-8 กำหนดค่า X-TAL ให้กับโปรแกรม Keil

5.2 Output ให้เลือกคลิกเมาส์ที่ค่าตัวเลือก Create HEX File พร้อมกับเลือกกำหนดรูปแบบของ Hex ให้เป็นแบบ HEX-386 แล้วเลือก OK ดังรูป



รูปที่ 3-9 การกำหนด Create Hex File ให้กับโปรแกรม Keil

เท่านี้ก็จะสามารถเริ่มต้นเขียนโปรแกรม ใน ARM7 ได้

3.2.2 การเขียนโปรแกรมติดต่อคอมพิวเตอร์ผ่านทาง RS232 โดยใช้โปรแกรม Keil uVision3 โดยสร้างโปรแกรมหาค่า กุญแจแบบ Multi Chaotic (3.1.6)

```
#include <LPC214x.h>
#include <stdio.h>
#include <stdlib.h>

float buffer,akx1,akx2,akx3,aky1,aky2,aky3,akz1,akz2,akz3,x,xx,y,yy,z,zz,h,a,b,c;
float kx,ky,kz;
int  ss,n;
unsigned long data,getnum(),i,m,length;
char scanfa();
        //Chen Key
        float fx(float a,float b,float c)
              {return 35*(b-a);}
        float fy(float a,float b,float c)
              { return (-7)*a-(a*c)+(28*b);}
        float fz(float a,float b,float c)
              {  return a*b-3*c;}

        //Chen Runge3rd
        float Chenfx(float x,float y,float z)
              {return 35*(y-x);}
        float Chenfy(float x,float y,float z)
              { return (-7)*x-(x*z)+(28*y);}
        float Chenfz(float x,float y,float z)
              { return x*y-3*z;}

        //Lorenz Runge3rd
        float Lorenzfx(float x,float y,float z)
              {return 10*(y-x);}
        float Lorenzfy(float x,float y,float z)
              {return x*(28 - z) - y ;}
        float Lorenzfz(float x,float y,float z)
              { return x*y-(8/3)*z;}

        //Rossler Runge3rd
        float Rosslerfx(float x,float y,float z)
              {return - y - z;}
        float Rosslerfy(float x,float y,float z)
              { return x + (0.2)*y;}
        float Rosslerfz(float x,float y,float z)
              {return (0.2) + z*( x - (5.7) );}
```

```
void uart0_init (void)
{
  PINSEL0 &= 0xFFFFFFF0;          // Reset P0.0,P0.1 Pin Config
  PINSEL0 |= 0x00000001;          // Select P0.0 = TxD(UART0)
  PINSEL0 |= 0x00000004;          // Select P0.1 = RxD(UART0)

  U0LCR &= 0xFC;                  // Reset Word Select(1:0)
  U0LCR |= 0x03;                  // Data Bit = 8 Bit
  U0LCR &= 0xFB;                  // Stop Bit = 1 Bit
  U0LCR &= 0xF7;                  // Parity = Disable
  U0LCR &= 0xBF;                  // Disable Break Control
  U0LCR |= 0x80;            // Enable Programming of Divisor Latches

  // U0DLM:U0DLL = 60.00 MHz / [16 x Baud]
  //          = 60.00 MHz / [16 x 9600]
  //          = 390.6 = 391 = 0187H
  U0DLM = 0x00;             // Program Divisor Latch(391) for 9600 Baud
  U0DLL = 0x21;

  U0LCR &= 0x7F;            // Disable Programming of Divisor Latches

  U0FCR |= 0x01;                  // FIFO Enable
  U0FCR |= 0x02;                  // RX FIFO Reset
  U0FCR |= 0x04;                  // TX FIFO Reset
  U0FCR &= 0x3F;
}
int main(void)
{
  uart0_init();                   // Initial UART0 = 9600,N,8,1

  while(1)
  {
    data=getnum();                //wait for Initial
        a=data;
        b=2;
        c=2;
        m=1000;
        h=0.001;
        for (i=1; i <= m ;i++ )
        {
          kx=fx(a,b,c);
          ky=fy(a,b,c);
          kz=fz(a,b,c);
          a=a+(kx*h);
          b=b+(ky*h);
          c=c+(kz*h);
        }
        x=a;
        y=b;
        z=c;
        length=getnum();          //wait for length
```

```
n=length;
buffer=0;
h=0.01;
for (i=1; i <= n ;i++ )
{
  if (i % 1000 < 333)//Chen
  {
        akx1=Chenfx(x,y,z);
        aky1=Chenfy(x,y,z);
        akz1=Chenfz(x,y,z);

        xx=x+(h*akx1)/2;
        yy=y+(h*aky1)/2;
        zz=z+(h*akz1)/2;

        akx2=Chenfx(xx,yy,zz);
        aky2=Chenfy(xx,yy,zz);
        akz2=Chenfz(xx,yy,zz);

        xx=x-(h*akx1)+(2*h*akx2);
        yy=y-(h*aky1)+(2*h*aky2);
        zz=z-(h*akz2)+(2*h*akz2);

        akx3=Chenfx(xx,yy,zz);
        aky3=Chenfy(xx,yy,zz);
        akz3=Chenfz(xx,yy,zz);

        x=x+(akx1+4*akx2+akx3)*(h/6);
        y=y+(aky1+4*aky2+aky3)*(h/6);
        z=z+(akz1+4*akz2+akz3)*(h/6);
  }
  if (( i % 1000 >= 333)&&(i % 1000 <666))//Lorenz
  {
        akx1=Lorenzfx(x,y,z);
        aky1=Lorenzfy(x,y,z);
        akz1=Lorenzfz(x,y,z);

        xx=x+(h*akx1)/2;
        yy=y+(h*aky1)/2;
        zz=z+(h*akz1)/2;

        akx2=Lorenzfx(xx,yy,zz);
        aky2=Lorenzfy(xx,yy,zz);
        akz2=Lorenzfz(xx,yy,zz);

        xx=x-(h*akx1)+(2*h*akx2);
        yy=y-(h*aky1)+(2*h*aky2);
        zz=z-(h*akz2)+(2*h*akz2);
```

```
                akx3=Lorenzfx(xx,yy,zz);
                aky3=Lorenzfy(xx,yy,zz);
                akz3=Lorenzfz(xx,yy,zz);

                x=x+(akx1+4*akx2+akx3)*(h/6);
                y=y+(aky1+4*aky2+aky3)*(h/6);
                z=z+(akz1+4*akz2+akz3)*(h/6);
        }
        if (i % 1000 >=666)//Rossler
        {
                akx1=Rosslerfx(x,y,z);
                aky1=Rosslerfy(x,y,z);
                akz1=Rosslerfz(x,y,z);

                xx=x+(h*akx1)/2;
                yy=y+(h*aky1)/2;
                zz=z+(h*akz1)/2;

                akx2=Rosslerfx(xx,yy,zz);
                aky2=Rosslerfy(xx,yy,zz);
                akz2=Rosslerfz(xx,yy,zz);

                xx=x-(h*akx1)+(2*h*akx2);
                yy=y-(h*aky1)+(2*h*aky2);
                zz=z-(h*akz2)+(2*h*akz2);

                akx3=Rosslerfx(xx,yy,zz);
                aky3=Rosslerfy(xx,yy,zz);
                akz3=Rosslerfz(xx,yy,zz);

                x=x+(akx1+4*akx2+akx3)*(h/6);
                y=y+(aky1+4*aky2+aky3)*(h/6);
                z=z+(akz1+4*akz2+akz3)*(h/6);
        }
       buffer=x;

        if(x<0)
        {
         buffer=buffer*(-1);
        }
        buffer=buffer*(1000);
        ss=buffer;
        printf("%c",ss);

  }
 }
}
```

```
/*****************************/
/* Write Character To UART0 */
/*****************************/
int putchar (int ch)
{
 if (ch == '\n')
 {
  while (!(U0LSR & 0x20));          // Wait TXD Buffer Empty
  U0THR = CR;                       // Write CR
 }
 while (!(U0LSR & 0x20));
      // Wait TXD Buffer Empty
 return (U0THR = ch);
        // Write Character
}
/*****************************/
/* Read Character From UART0 */
/*****************************/
char scanfa(void)
{
 while (!(U0LSR & 0x01));           // Wait RXD Receive Data Ready
 return (U0RBR);                    // Get Receive Data & Return
}
unsigned long getnum(void)
{   unsigned long sum=0,x=0;
       int i=0;
       do{
        x=scanfa()-'0';
        if(x!=49)
        {
                    sum=(sum*10)+x;
        }
       }while( x!=49);
       return(sum);                 // Get Receive Data & Return
}
```

3.3 เขียนโปรแกรม ภาษา C# ในส่วนการติดต่อผ่าน RS232 และการเข้ารหัสลับข้อมูล โดยใช้โปรแกรม Visual Studio 2005

3.3.1 หน้าจอของโปรแกรมดังรูป

รูปที่ 3-10 แสดงหน้าต่าง โปรแกรมเข้ารหัสลับและถอดรหัสลับ

3.3.2 การเขียนโปรแกรม C# เพื่อการเข้ารหัสลับและถอดรหัสลับข้อมูลซึ่งมีโค้ดดังนี้

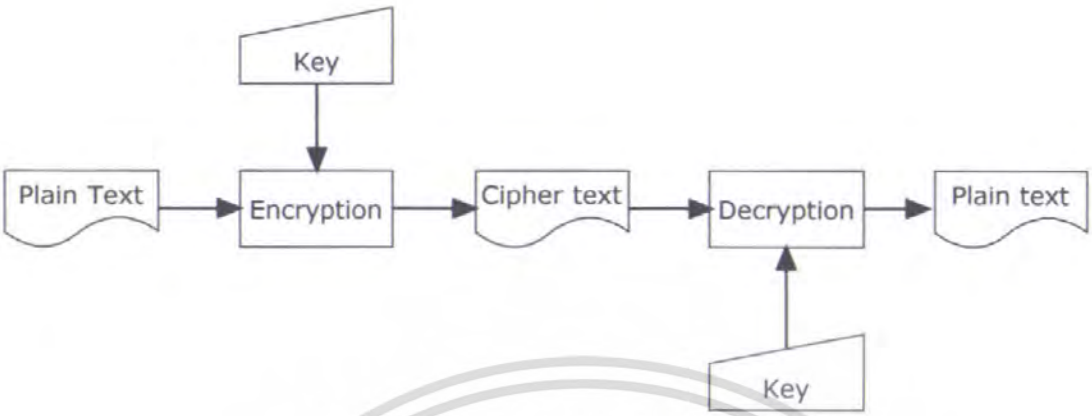
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.IO;

namespace Encryption
{

    public partial class Form1 : Form
    {
        private UInt32 dlen;
        private string slen = "";
        private string ss="";
        private string ky = "";
        private string cipher = "";
        private string plain = "";

        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            setPort();              //set parameter of serial port
        }
        private void bro_Click(object sender, EventArgs e)
        {
            openFile();             //open plaint text for encrypt or
                                    cipher for decrypt
        }
```

```
        private void enc_Click(object sender, EventArgs e)
    {
        if (lb.Text == "") MessageBox.Show("กรุณาเลือกไฟล์ที่ต้องการ !!!",
"ข้อผิดพลาด", MessageBoxButtons.OK, MessageBoxIcon.Information);
                                    //check choose file
        else if (textBox1.Text == "") MessageBox.Show("กรุณาป้อนตัวเลขสอง
หลัก !!!", "ข้อผิดพลาด", MessageBoxButtons.OK, MessageBoxIcon.Information);
                                    //check key value define
        else
        {
        ss = "";
        sendKey(kv);                //send a key value from user
        delay();                    //delay time for arm7
        sendKey(slen);              //send length of plain text in
        wait();                     //wait for recieve key stream
        encr();                     //encrypt or decrypt
        saveFile();                 //save cipher or de cipher text
        }
    }

    private void rs232_DataReceived(object sender,
SerialDataReceivedEventArgs e)
    {
        ss += rs232.ReadExisting();//get stream of key
    }

    private void setPort()
    {
        try
        {
            string[] s = SerialPort.GetPortNames();
                                    //create object for get port name
            int i = 0;
            comboBox1.Items.Clear();//clear item in combo box
            foreach (string port in s)
            {
                comboBox1.Items.Add(s[i]);//loop to get port name
                                          and add in to combo box
                i++;
            }
            comboBox1.SelectedIndex = 0;//select default
            if (comboBox1.Items.Count > 0)//check serial
                                          port choose
            {
                if (rs232.IsOpen) rs232.Close();//check status of
                                                serial port
                rs232.PortName = comboBox1.Text;//set port name
                rs232.Open();               //open serial port
                rs232.DiscardInBuffer();    //clear input buffer
                rs232.DiscardOutBuffer();   //clear output buffer
            }
        }
        catch (InvalidOperationException e)
        {
            MessageBox.Show("กรุณาลองใหม่อีกครั้ง"+e, "Please try again
!!!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
        catch (Exception e)
        {
```

```
            MessageBox.Show("กรุณาลองใหม่อีกครั้ง"+e, "Please try again !!!",
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }

        private void sendKey(string slen)
        {
            char[] clen = new char[slen.Length];//create char array
                                                  for send to ARM7
            for (int i = 0; i < slen.Length; i++)
            {
                clen[i] = slen[i];            //convert string to array
                rs232.Write(clen, i, 1);      //send one char per loop
            }
            rs232.Write("a");                 //send "a" for finally
        }

        private void encr()
        {
            cipher = "";
            char[] cip= new char[plain.Length]; //create cipher text
                                                  in char array
            int[] cp = new int[plain.Length];   //creatr cipher text
                                                  in integer array
            for (int k = 0; k < plain.Length; k++)
            {
                cp[k] = plain[k] ^ ss[k];       //XOR between plain
                                                  text and key, cipher in integer
                cip[k] = Convert.ToChar(cp[k]); //convert cipher in
                                                  integer to char
                cipher += cip[k].ToString();    //get array of cipher
                                                  to string cipher
            }
        }

        private void delay()
        {
            for (int i = 0; i <= 100; i++){ }   //delay loop
        }

        private void wait()
        {
            while ((ulong)ss.Length < dlen) { MessageBox.Show("กำลังรับ
กุญแจลับ", "Loading... Please wait !!!", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation); }
                        //wait time until length of key equal plain
        }

        private void openFile()
        {
            plain = "";
            OpenFileDialog ofDlg = new OpenFileDialog();
                            //create object to open file dialog
            ofDlg.Title = "เลือกไฟล์ที่ต้องการเข้ารหัสลับ ";     //set title of dialog
            ofDlg.Filter = "All Files (*.*)|*.*";
                                                  //set type of file to open
            ofDlg.RestoreDirectory = true;        //set restrore
            if (ofDlg.ShowDialog() == DialogResult.OK)
            {
```

```
            string FilePath = "";
            FilePath = ofDlg.FileName;      //get file path
            FileInfo fif = new FileInfo(FilePath);
                                            //create object file info
            lb.Text = fif.FullName;  //show name of file in label
            dlen = (ulong)fif.Length;    //get length of file
            slen = dlen.ToString();      //convert length of file
                                            to string type
            StreamReader sr = new StreamReader
    (FilePath,Encoding.GetEncoding(874),true);
                        //create object straem reader
            plain = sr.ReadToEnd();         //read from stream
                                             to plain text
            sr.Close();                     //close stream
        }
    }

    private void savePile()
    {
        string strPath = "";
        SaveFileDialog sfDlg = new SaveFileDialog();//create
                                                    object to save file dialog
        sfDlg.Title = "บันทึกข้อมูลเข้ารหัสลับ ";     //set title of dialog
        sfDlg.Filter = "All Files (*.*)|*.*";  //set type of
                                                file to save
        sfDlg.RestoreDirectory = true;          //set restore
        if (sfDlg.ShowDialog() == DialogResult.OK)
        {
            strPath = sfDlg.FileName;           //get file path
            sw = new StreamWriter(strPath, false,
Encoding.GetEncoding(874));         //create object straem writer
            sw.Write(cipher);           //write cipher to stream
            sw.Flush();                 //clear stream buffer
            sw.Close();                 //close stream
            MessageBox.Show("บันทึกเรียบร้อยแล้ว!!", "ผลการทำงาน");
        }
    }

    private void textBox1_KeyPress(object sender,
KeyPressEventArgs e)
    {
if ((e.KeyChar < 48 || e.KeyChar > 57) && (e.KeyChar != 8))
                        //check value is not character and space
        {
            e.Handled = true;
        }
        kv = textBox1.Text;     //get key value
    }
}
}
```

3.4 System Flow Diagram ของโปรแกรมเข้ารหัส

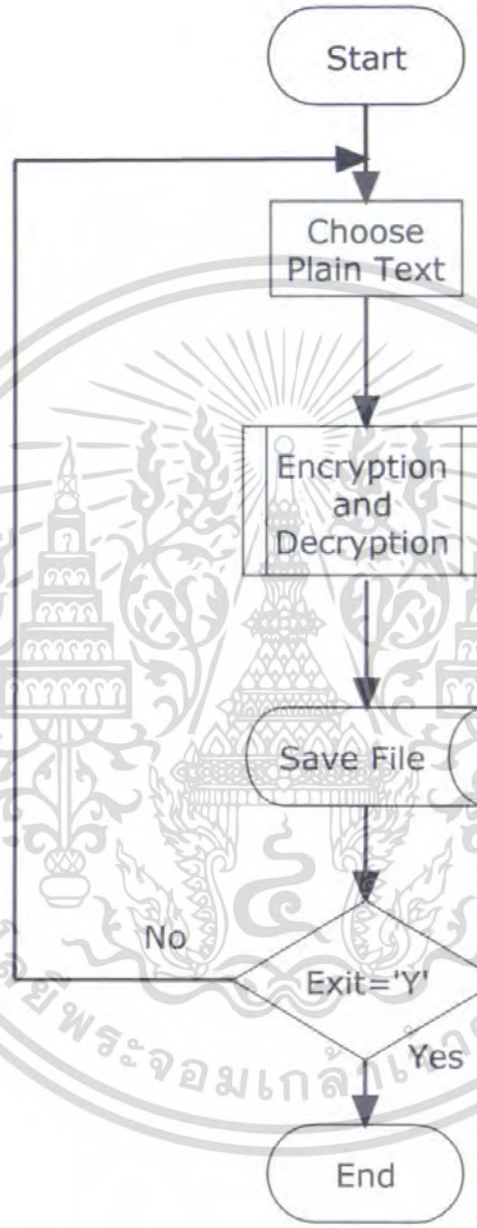


รูปที่ 3-11 System Flow Diagram ของโปรแกรม

จาก System Flow Diagram สามารถอธิบายได้ดังนี้คือ

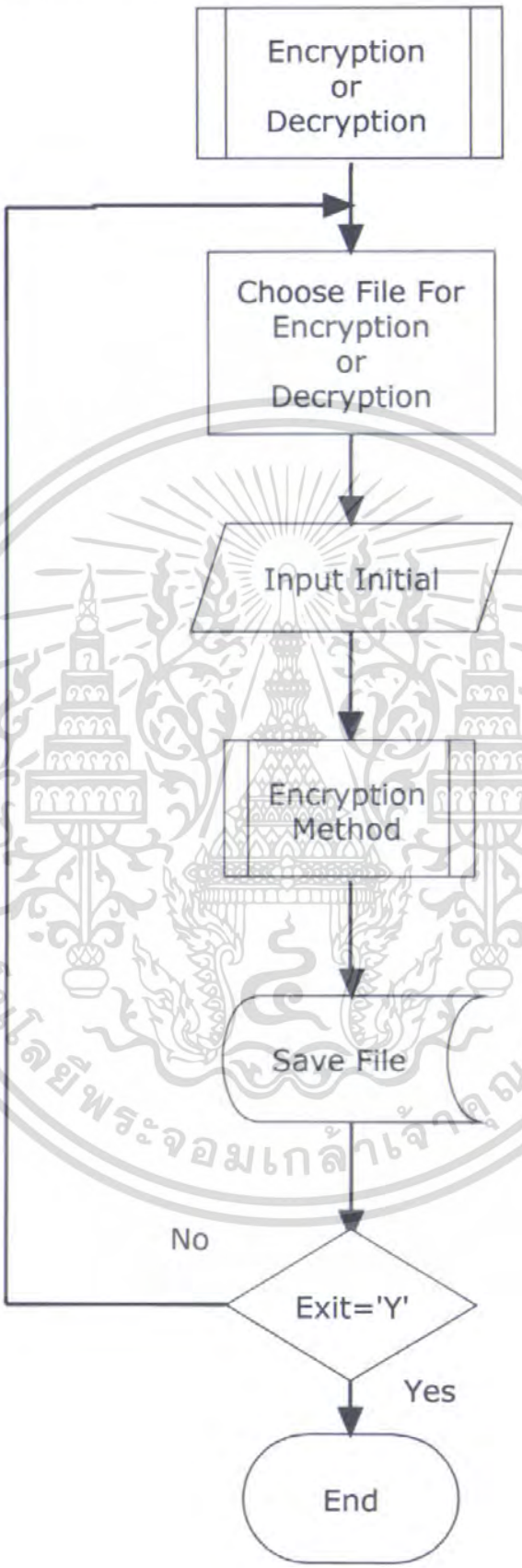
1. ผู้ใช้กำหนดไฟล์ข้อมูล (Plain text) และ กำหนดค่าเริ่มต้นที่ใช้ในการเข้ารหัส
2. เข้ารหัสไฟล์โดยใช้ กุญแจจาก ไมโครคอนโทรเลอร์ อาร์มเซเว่น จะได้ผลลัพธ์ (Cipher text) ออกมา จากนั้นจึงบันทึกเป็นไฟล์ที่เข้ารหัส ซึ่งจะใช้ในการส่งต่อไป
3. ถอดรหัสไฟล์โดยใช้ กุญแจจาก ไมโครคอนโทรเลอร์ อาร์มเซเว่น และค่าเริ่มต้นเดียวกัน
4. จะได้ไฟล์ข้อมูลเดิม (Plain text) กลับมา

3.4.1 Flow Chart



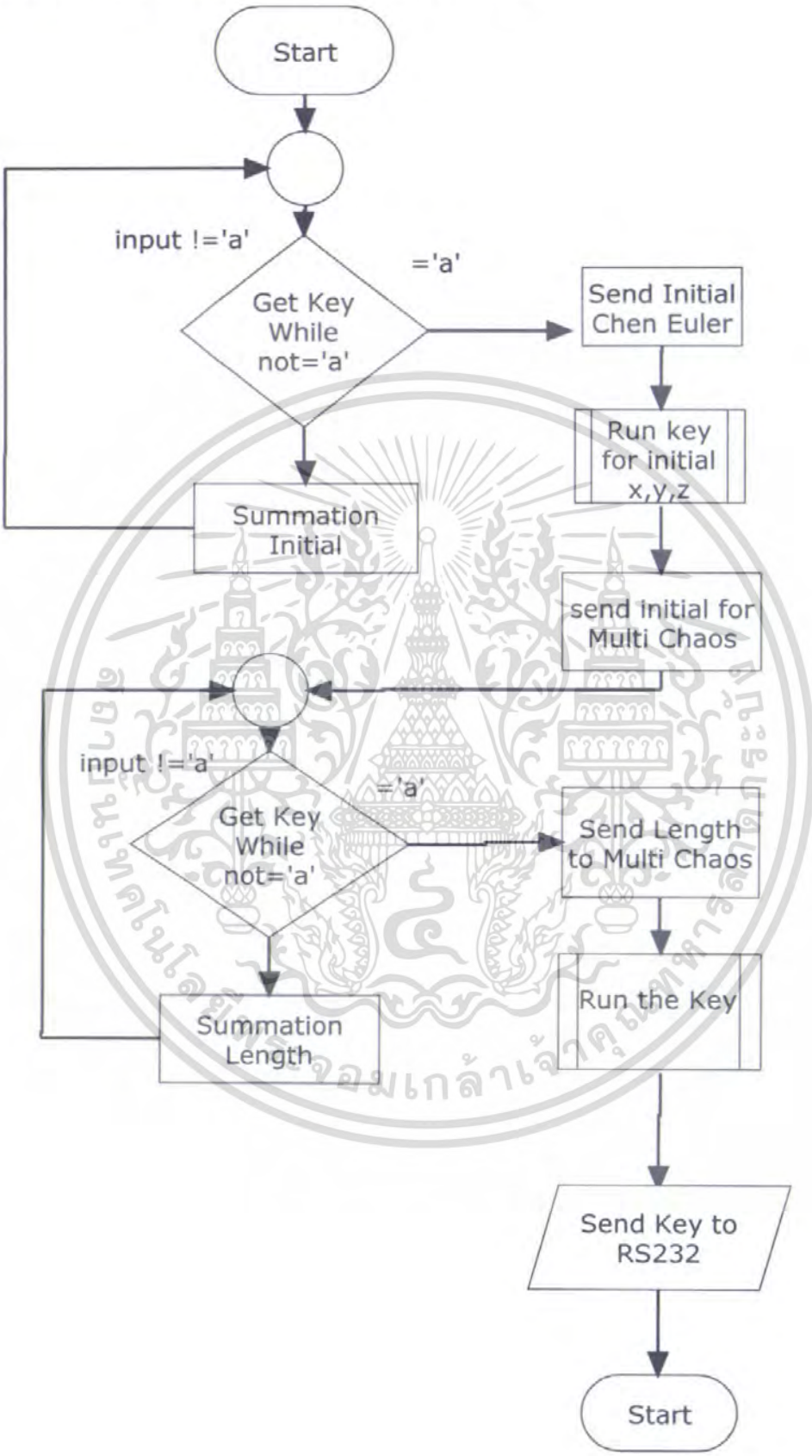
รูปที่ 3-12 Flow Chart ของโปรแกรม

ส่วนการทำงานของ ส่วนของโปรแกรม C#



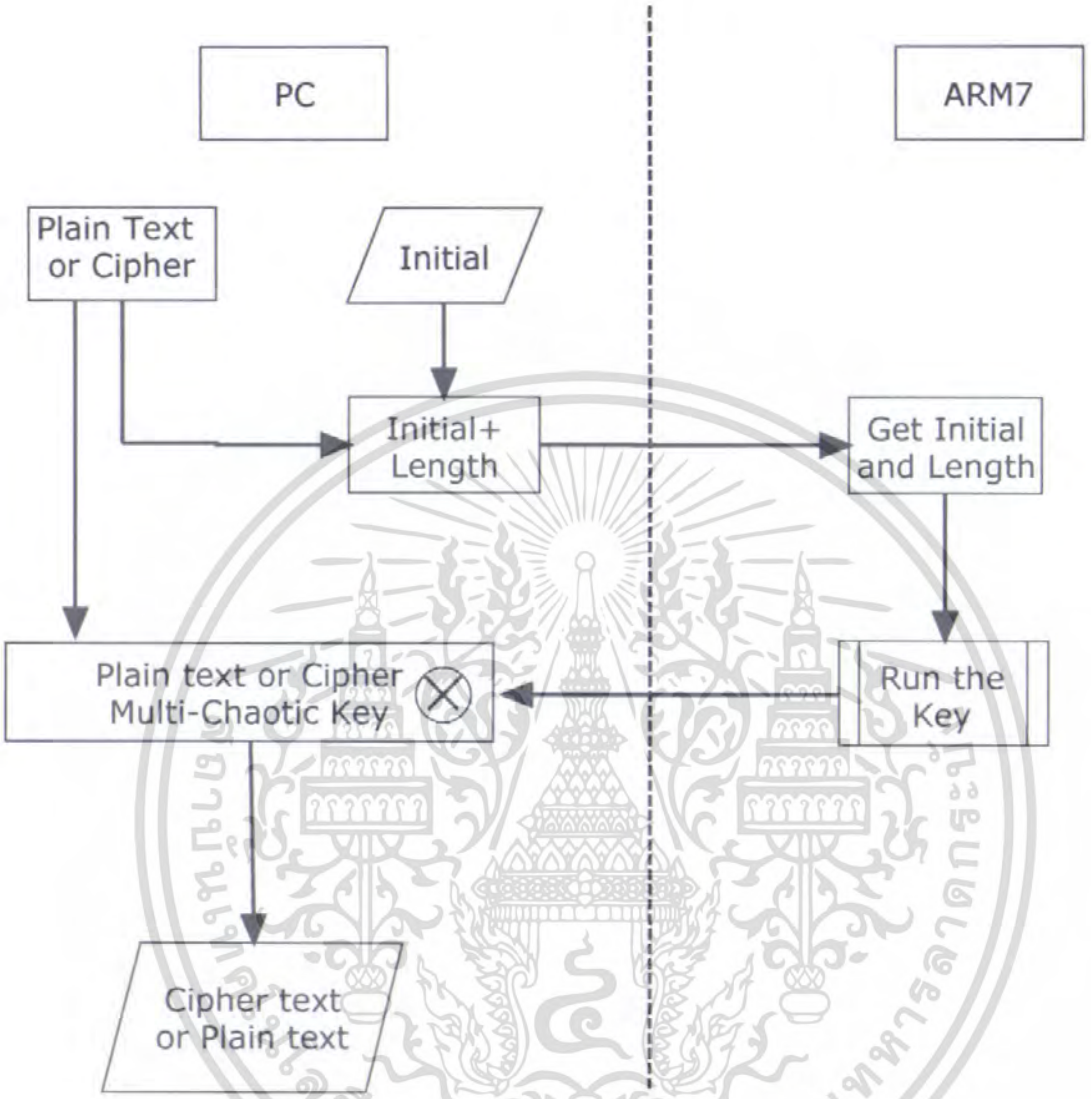
รูปที่ 3-13 Flow Chart ของการเข้ารหัสลับและการถอดรหัสลับ

การทำงานโปรแกรม ส่วนของ ไมโครคอนโทลเลอร์ อาร์มเซเว่น



รูป 3-14 Flow Chart ของไมโครคอนโทรเลอร์ ARM-7

การทำงานระหว่าง PC และ ARM7



รูปที่ 3-15 Timing Diagram ของการทำงานระหว่าง PC และ ARM7

บทที่ 4

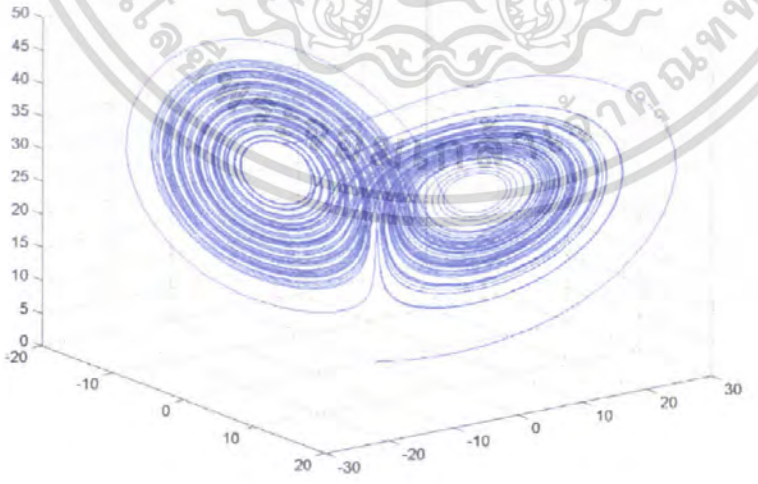
ผลการทดลอง

4.1 ผลที่ได้จากการออกแบบโปรแกรม

4.1.1 ผลการทดลองการประมวลผลสมการ Chaotic โดยใช้ สมการ Lorenz

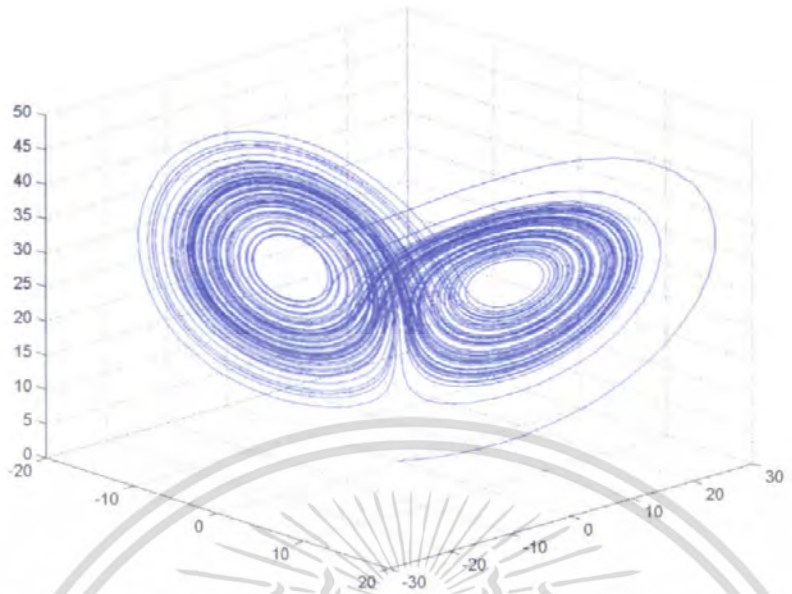
x=0.1, y = 0, z = 0, h = 0.01, n=10000

รูปที่ 4-1 ผลลัพธ์ที่ได้จากการแก้สมการของ Lorenz โดยใช้ระเบียบวิธี Euler



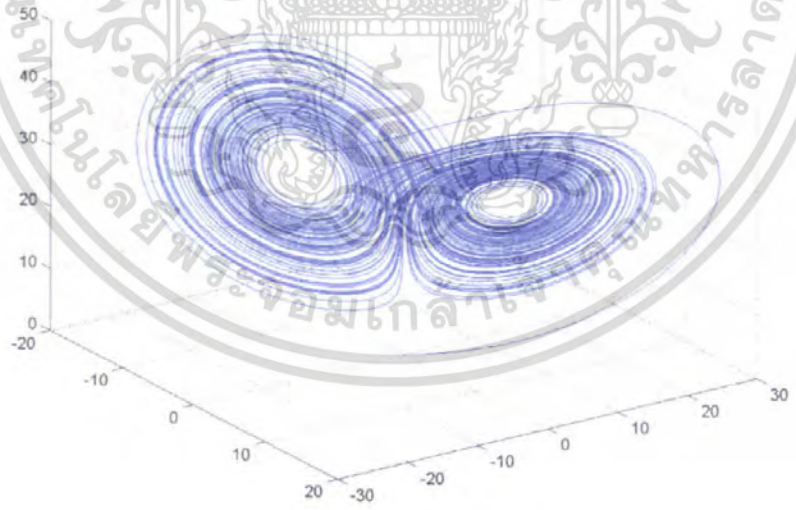
x=0.1, y = 0, z = 0, h = 0.01, n=10000

รูปที่ 4-2 ผลลัพธ์ที่ได้จากการแก้สมการของ Lorenz โดยใช้ระเบียบวิธี Huan



x=0.1, y = 0, z = 0, h = 0.01, n=10000

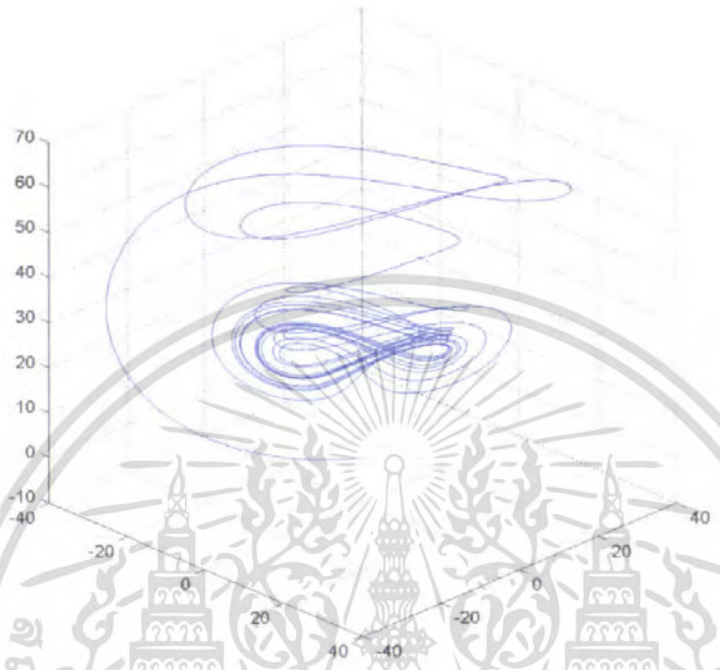
รูปที่ 4-3 ผลลัพธ์ที่ได้จากการแก้สมการของ Lorenz โดยใช้ระเบียบวิธี Euler Modified



x=0.1, y = 0, z = 0, h = 0.01, n=10000

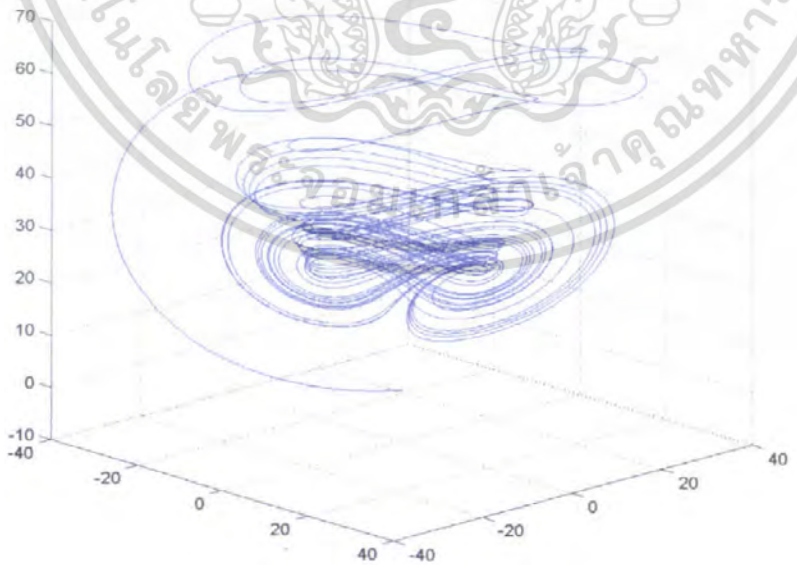
รูปที่ 4-4 ผลลัพธ์ที่ได้จากการแก้สมการของ Lorenz โดยใช้ระเบียบวิธี Runge Kutta'3rdorder

1.1.2 ผลการทดลองการประมวลผลสมการ Chaotic โดยใช้ สมการ Chen



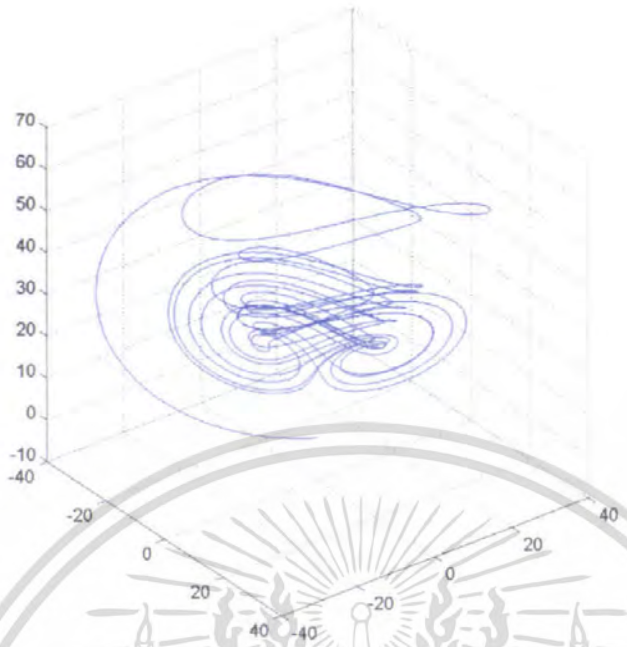
x=0.1, y = 0, z = 0, h = 0.01, n=10000

รูปที่ 4-5 ผลลัพธ์ที่ได้จากการแก้สมการของ Chen โดยใช้ระเบียบวิธี Euler



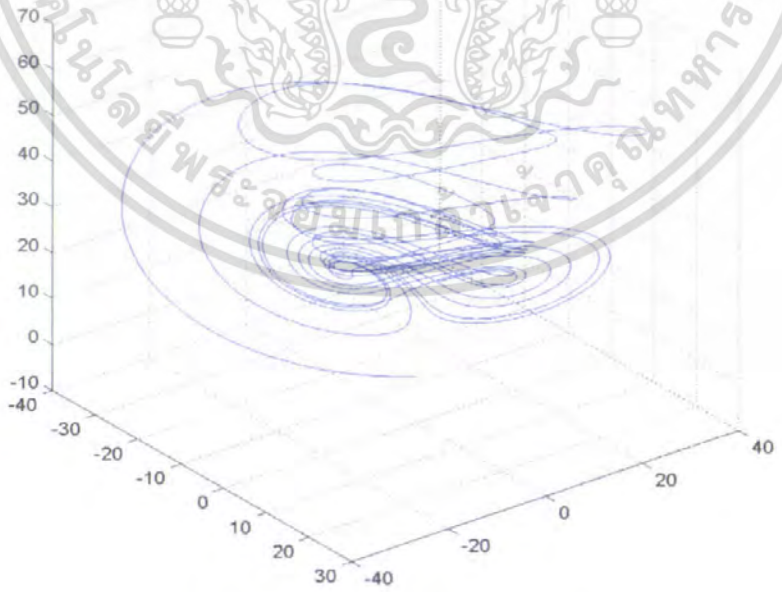
x=0.1, y = 0, z = 0, h = 0.01, n=10000

รูปที่ 4-6 ผลลัพธ์ที่ได้จากการแก้สมการของ Chen โดยใช้ระเบียบวิธี Huan



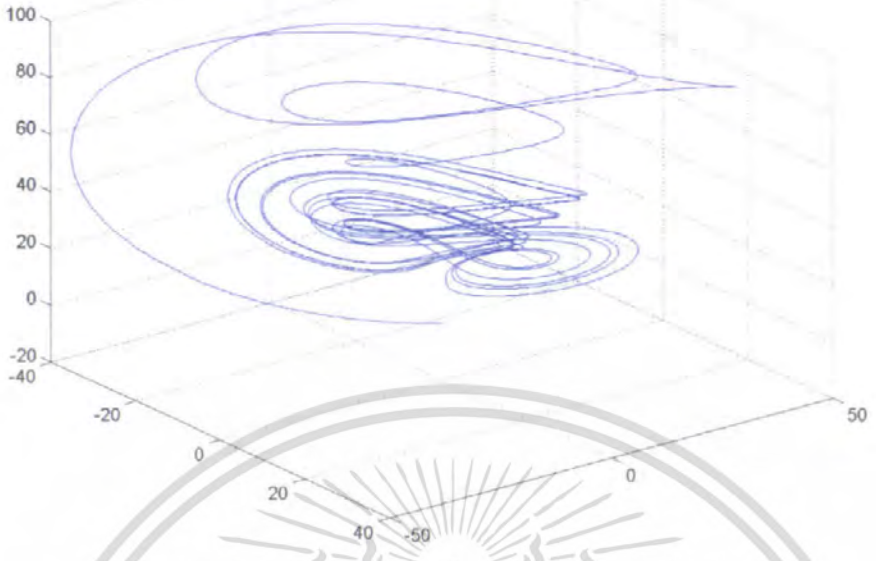
x=0.1, y = 0, z = 0, h = 0.01, n=10000

รูปที่ 4-7 ผลลัพธ์ที่ได้จากการแก้สมการของ Chen โดยใช้ระเบียบวิธี Euler Modified



x=0.1, y = 0, z = 0, h = 0.01, n=10000

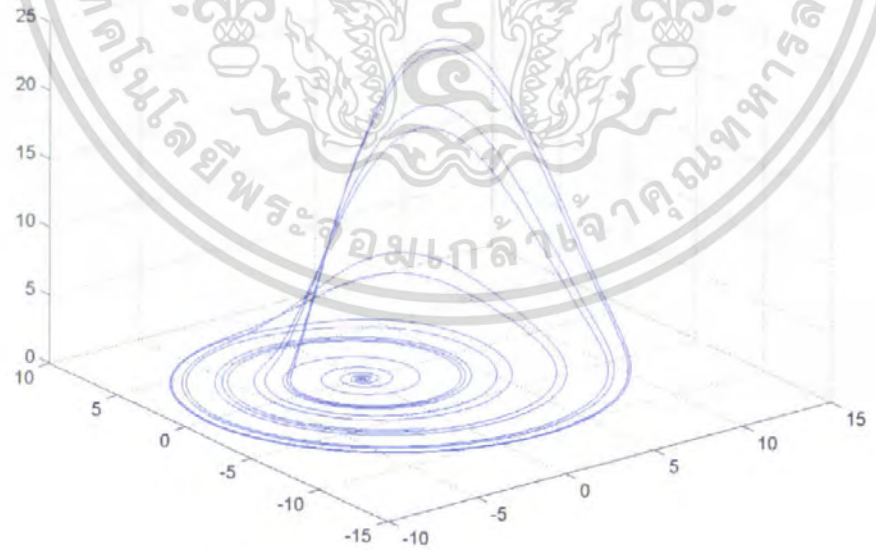
รูปที่ 4-8 ผลลัพธ์ที่ได้จากการแก้สมการของ Chen โดยใช้ระเบียบวิธี Runge Kutta'3rdorder



x=0.1, y = 0, z = 0, h = 0.01, n=10000

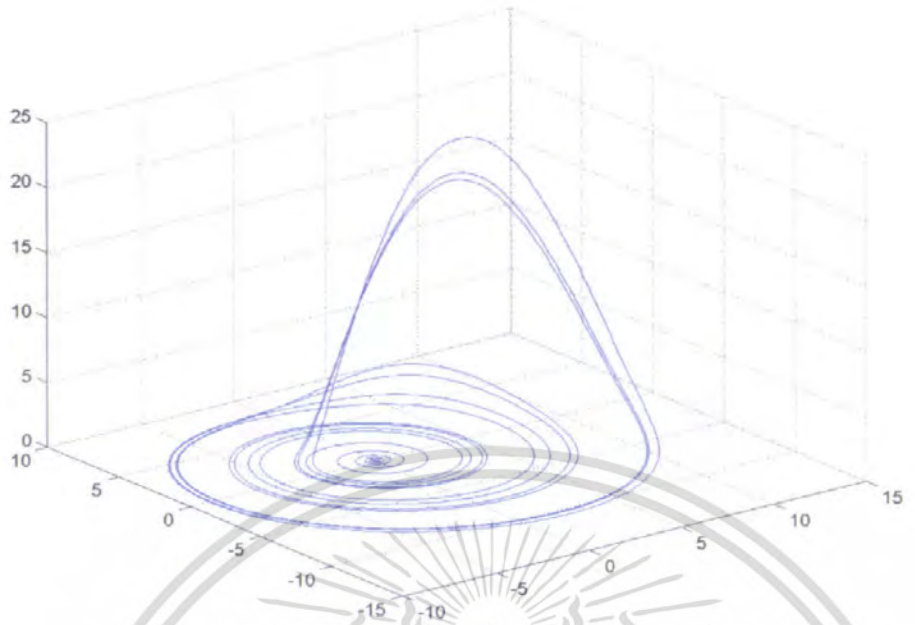
รูปที่ 4-9 ผลลัพธ์ที่ได้จากการแก้สมการของ Chen โดยใช้ระเบียบวิธี Runge Kutta'4th order

4.1.3 ผลการทดลองการประมวลผลสมการ chaotic โดยใช้ สมการ Rossler



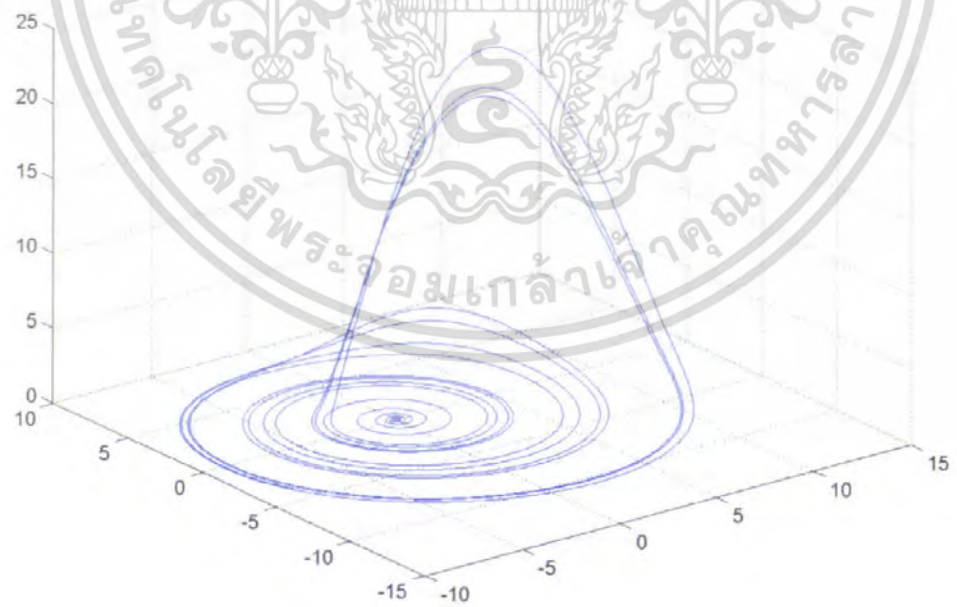
x=0.1, y = 0, z = 0, h = 0.01, n=10000

รูปที่ 4-10 ผลลัพธ์ที่ได้จากการแก้สมการของ Rossler โดยใช้ระเบียบวิธี Euler



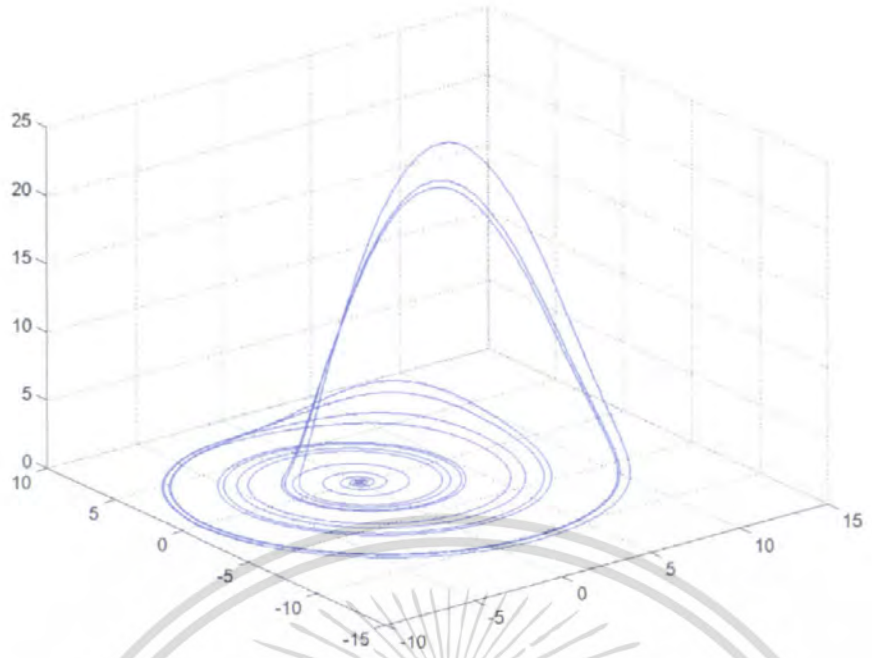
x=0.1, y = 0, z = 0, h = 0.01, n=10000

รูปที่ 4-11 ผลลัพธ์ที่ได้จากการแก้สมการของ Rossler โดยใช้ระเบียบวิธี Huan



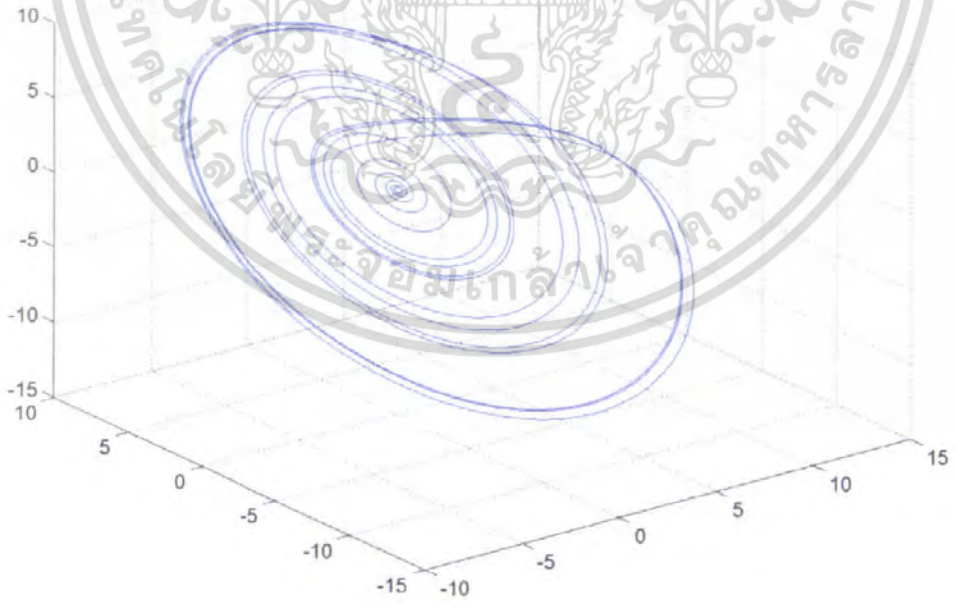
x=0.1, y = 0, z = 0, h = 0.01, n=10000

รูปที่ 4-12 ผลลัพธ์ที่ได้จากการแก้สมการของ Rossler โดยใช้ระเบียบวิธี Euler Modified



x=0.1, y = 0, z = 0, h = 0.01, n=10000

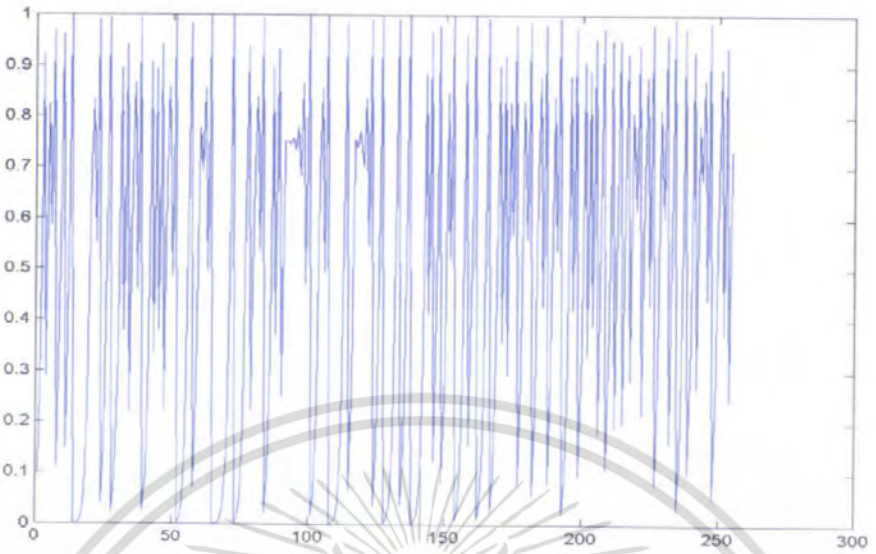
รูปที่ 4-13 ผลลัพธ์ที่ได้จากการแก้สมการของ Rossler โดยใช้ระเบียบวิธี Runge Kutta'3rd order



x=0.1, y = 0, z = 0, h = 0.01, n=10000

รูปที่ 4-14 ผลลัพธ์ที่ได้จากการแก้สมการของ Rossler โดยใช้ระเบียบวิธี Runge Kutta'4th order

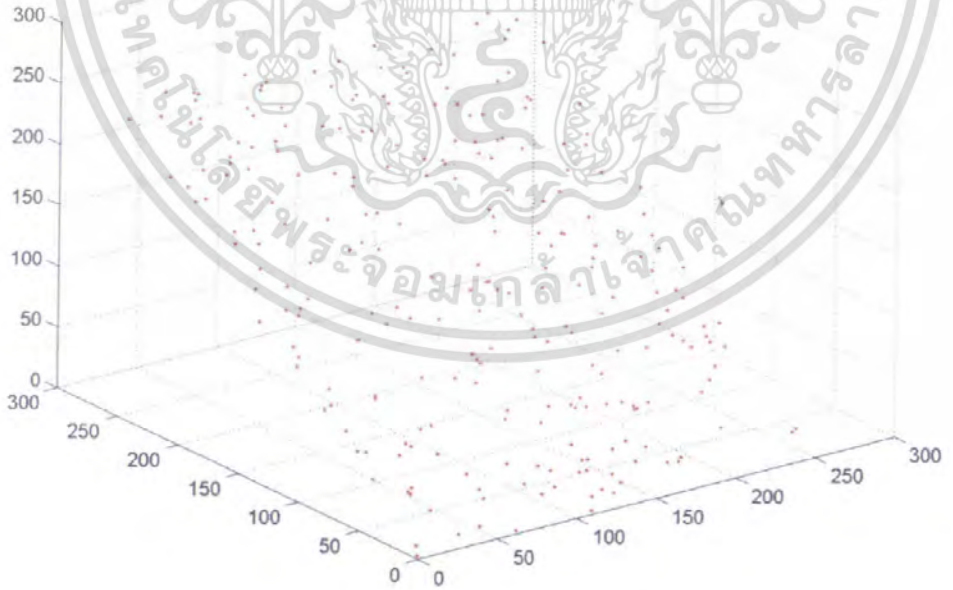
4.1.4 ผลการทดลองการประมวลผล โดยใช้ Logistic Map



x =0.1 และ r = 4

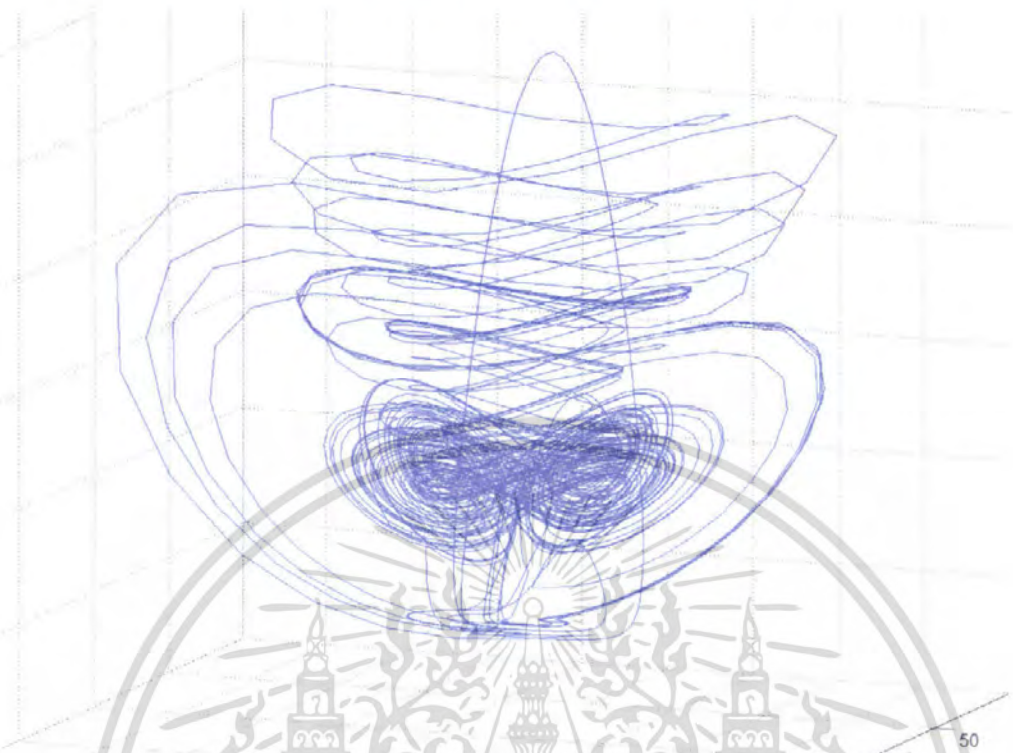
รูปที่ 4-15 ผลลัพธ์ที่ได้จากการประมวลสมการ Logistic map

4.1.5 ผลการทดลองการประมวลผล โดยใช้ Cat map



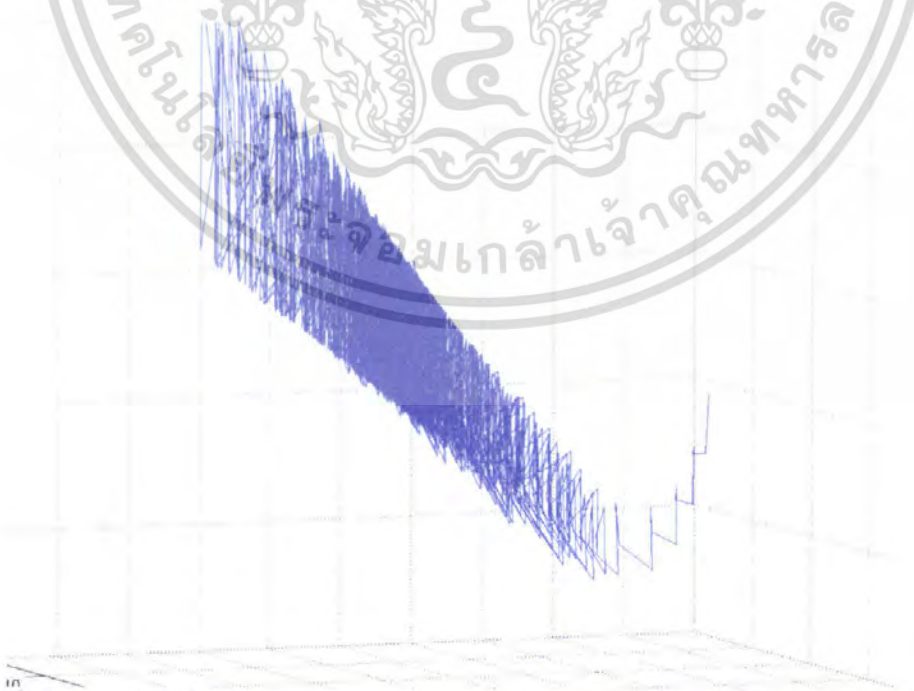
รูปที่ 4-16 ผลลัพธ์ที่ได้จากการประมวลสมการ Cat map

4.1.6.1 ผลการทดลอง Multi-chaotic แบบสลับตัวดึงดูด ทุกๆ 333 ค่า



รูปที่ 4-17 ผลลัพธ์ที่ได้จากการทดลอง Multi-Chaotic แบบปกติ

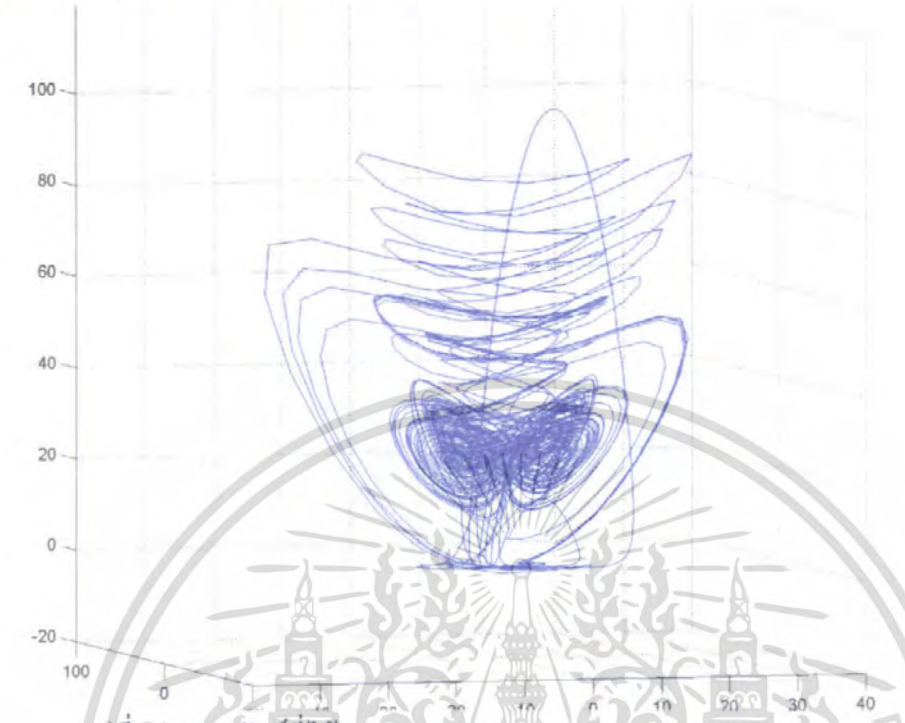
4.1.6.2 การทดลอง Multi-Chaotic แบบสลับตัวดึงดูด ทุกๆ 1 ค่า



รูปที่ 4-18 ผลลัพธ์ที่ได้จากการทดลอง Multi-Chaotic แบบเปลี่ยน ตัวดึงดูด ทุกๆ 1 ค่า

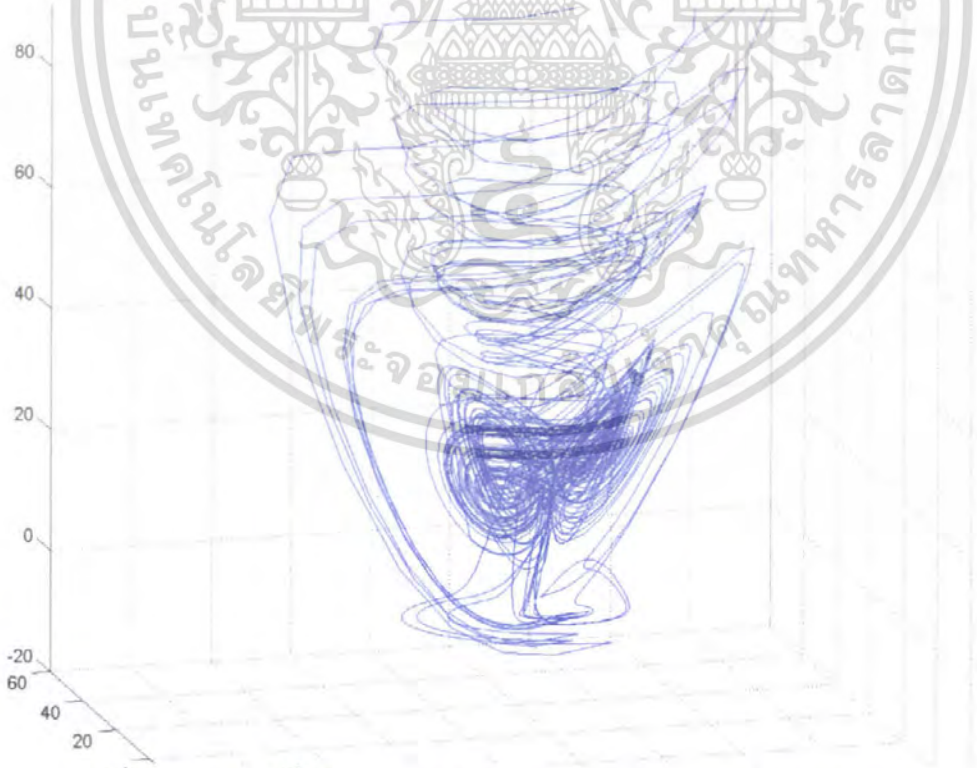
4.1.7 การทดลอง Multi-Chaotic แบบ ขึ้นอยู่กับค่าเริ่มต้น

CASE 0 Chen-->Lorenz-->Rossler



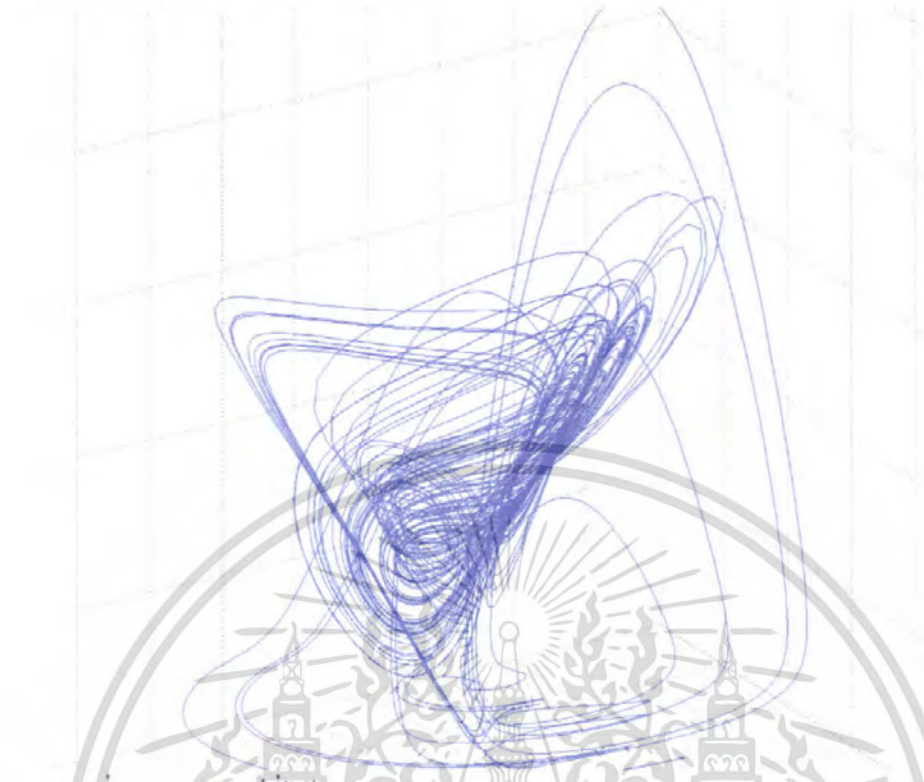
รูปที่ 4-19 ผลลัพธ์ที่ได้จากการทดลอง Case '0' Chen-->Lorenz-->Rossler

CASE 1 Rossler-->Chen-->Lorenz



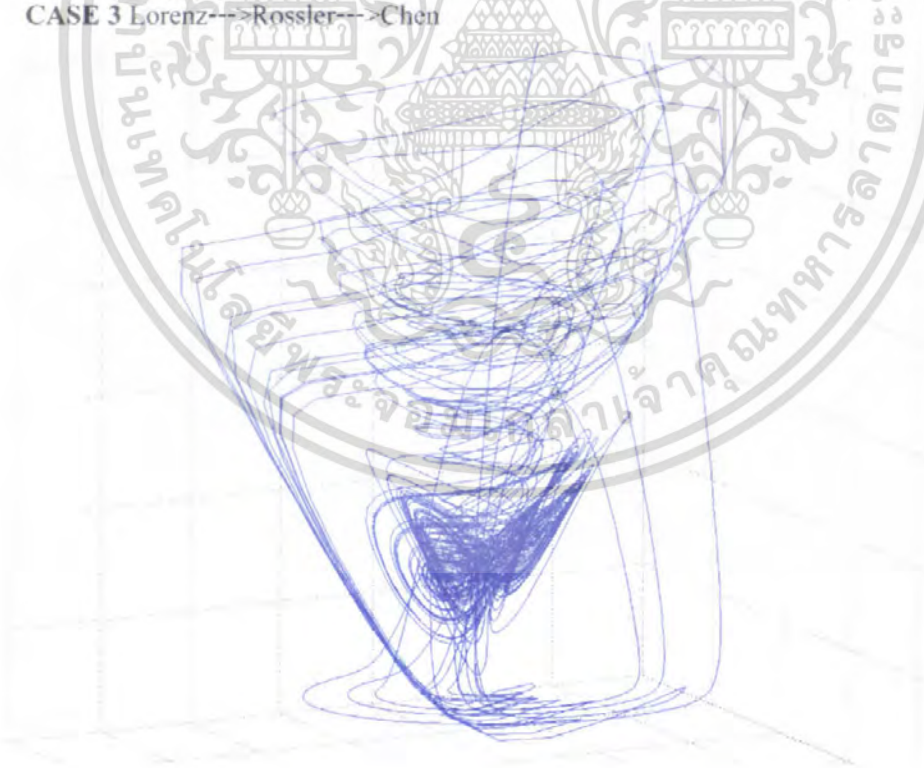
รูปที่ 4-20 ผลลัพธ์ที่ได้จากการทดลอง Case '1' Rossler-->Chen-->Lorenz

CASE 2 Lorenz--->Chen--->Rossler



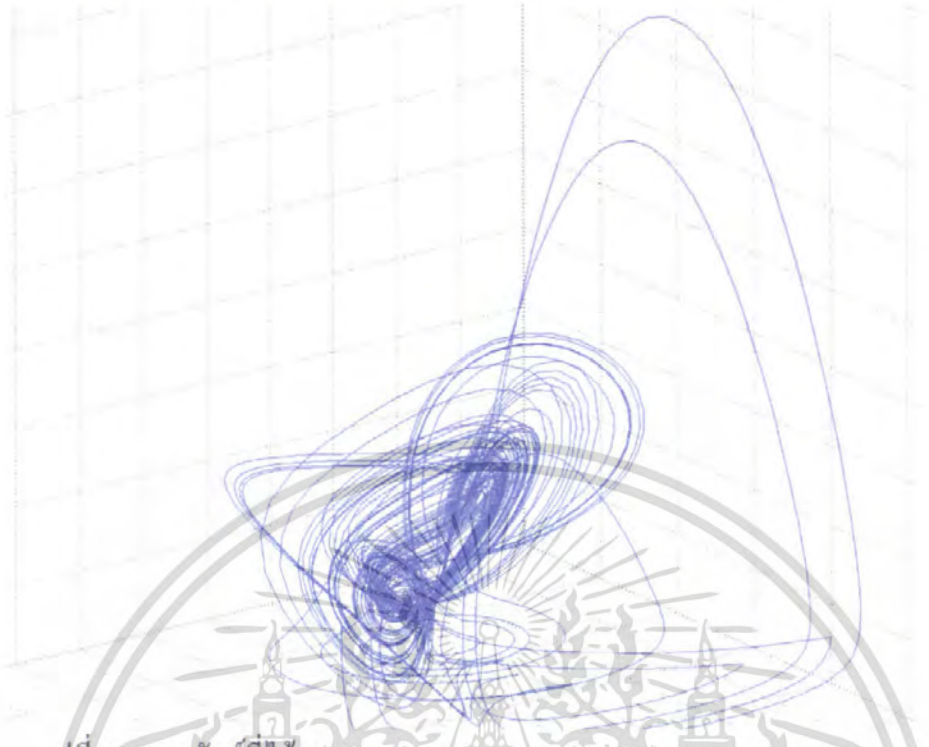
รูปที่ 4-21 ผลลัพธ์ที่ได้จากการทดลอง Case '2' Lorenz--->Chen--->Rossler

CASE 3 Lorenz--->Rossler--->Chen



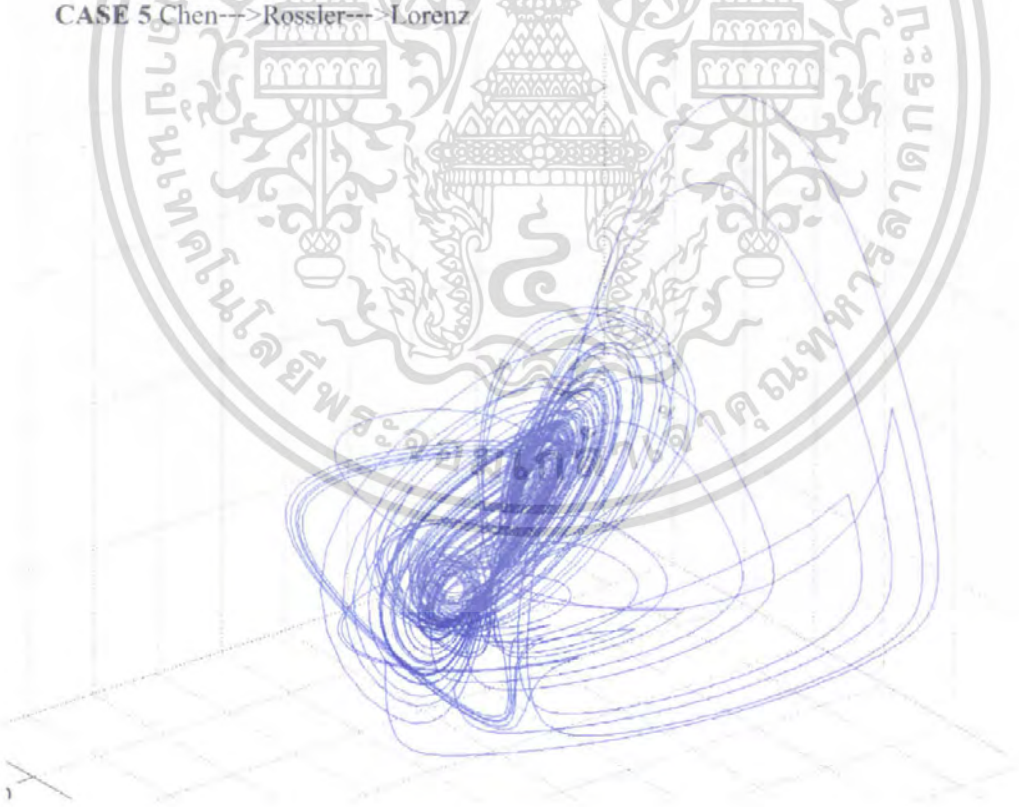
รูปที่ 4-22 ผลลัพธ์ที่ได้จากการทดลอง Case '3' Lorenz--->Rossler--->Chen

CASE 4 Rossler--->Lorenz--->Chen



รูปที่ 4-23 ผลลัพธ์ที่ได้จากการทดลอง Case '4' Rossler--->Lorenz--->Chen

CASE 5 Chen--->Rossler--->Lorenz



รูปที่ 4-23 ผลลัพธ์ที่ได้จากการทดลอง Case '5' Chen--->Rossler--->Lorenz

4.2 การทดลองเข้ารหัสลับ ไฟล์

4.2.1 การเข้ารหัสลับไฟล์

ขั้นตอนที่ 1ทำการเปิดโปรแกรมเข้ารหัส

Encryption.exe

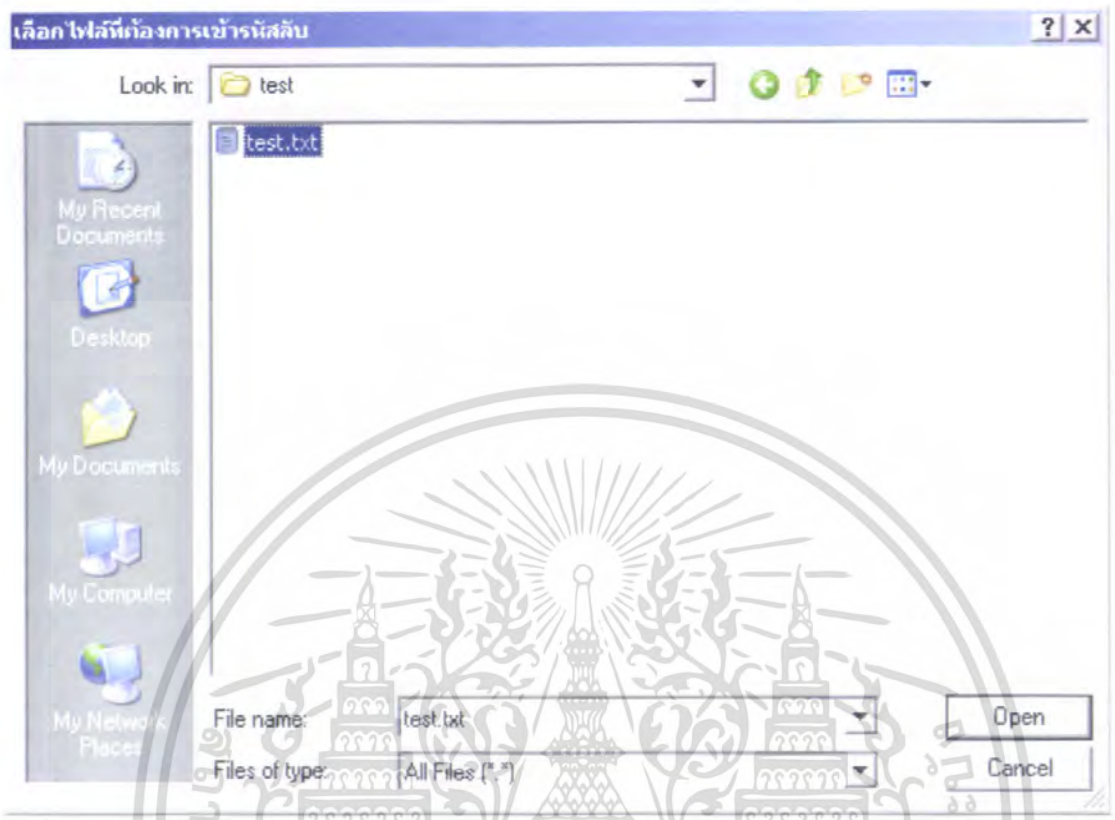
รูปที่ 4-24 รูปไอคอนโปรแกรมเข้ารหัสลับและถอดรหัสลับ

รูปที่ 4-25 รูปหน้าตาโปรแกรม

ขั้นตอนที่ 2 ทำการสร้าง ไฟล์ต้นฉบับเพื่อทดสอบการเข้ารหัสลับ

รูปที่ 4-26 ถึงไฟล์ต้นฉบับ

ขั้นตอนที่ 3 เลือกไฟล์ที่ต้องการเข้าหรัสลับ



รูปที่ 4-26 การเปิดไฟล์ ต้นฉบับ (Plain Text)

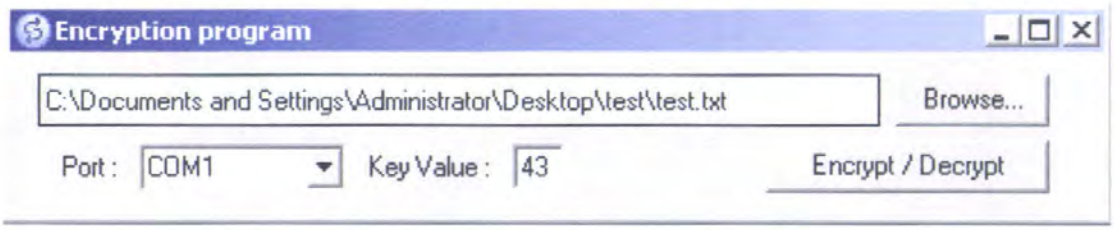
เมื่อได้เลือกไฟล์ที่ต้องการเข้ารหัสแล้วแล้ว

รูปที่ 4-27 ตำแหน่งและชื่อของไฟล์ที่ต้องการเข้ารหัส

ขั้นตอนที่ 4 ทำการกำหนด COM PORT ที่เราเสียบกับ ARM7

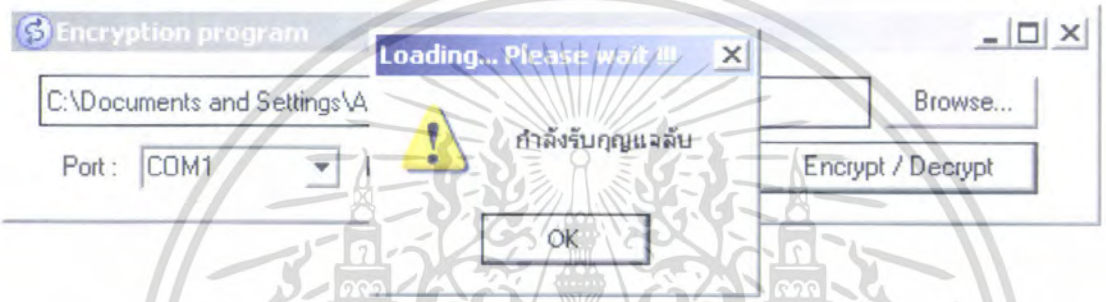
รูปที่ 4-28 การกำหนด COM PORT ให้กับโปรแกรม

ขั้นตอนที่ 5 ทำการกำหนด COM PORT ที่เราเสียบกับ ARM7



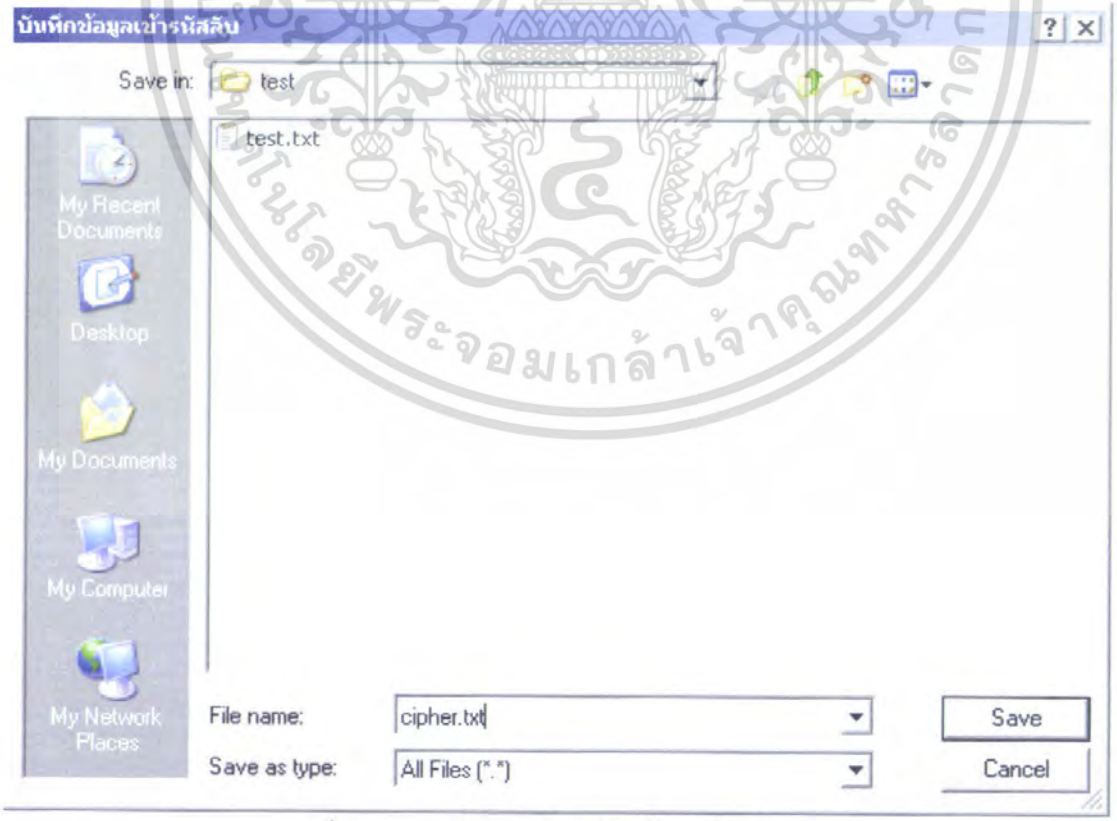
รูปที่ 4-29 กำหนดค่าเริ่มต้นให้กับไฟล์ที่ต้องการเข้ารหัสลับ

ขั้นตอนที่ 6 ทำการ Encryption ไฟล์ และรอ

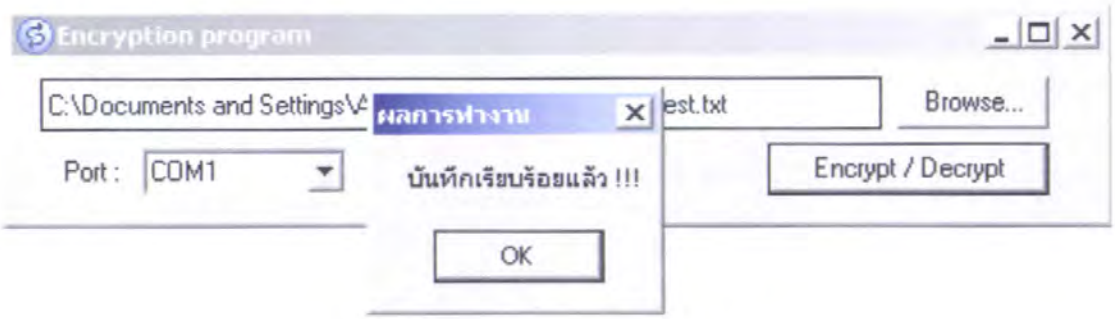


รูปที่ 4-30 หน้าต่างขณะที่ทำการเข้ารหัส

ขั้นตอนที่ 7 ทำการเลือกบันทึกไฟล์ ที่ถูกเข้ารหัสแล้ว (Cipher)



รูปที่ 4-31 การเลือกบันทึกไฟล์ ที่ถูกเข้ารหัสแล้ว (Cipher)



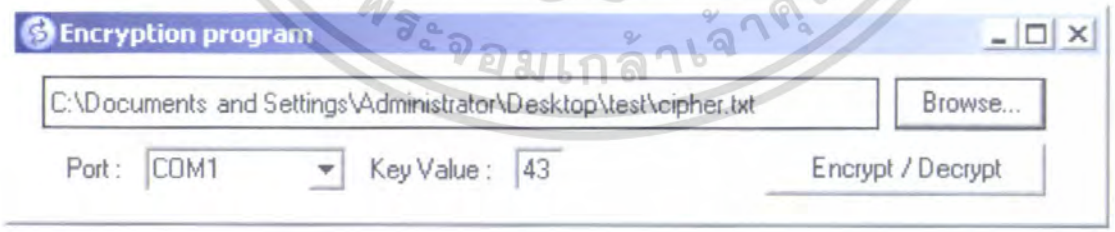
รูปที่ 4-32 การเลือกบันทึกไฟล์ ที่เสร็จสิ้นแล้ว

4.2.2 การถอดรหัสลับไฟล์

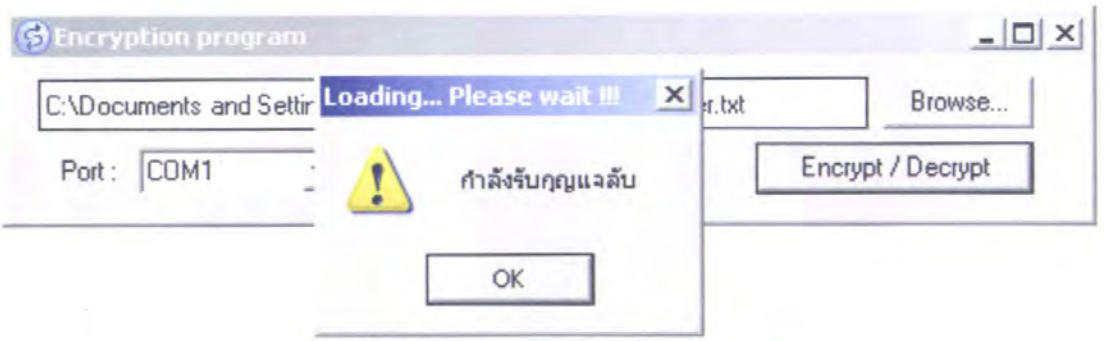
ขั้นตอนที่ 1 ลักษณะไฟล์ที่ถูกเข้ารหัสลับแล้ว

รูปที่ 4-33 ไฟล์ที่ถูกเข้ารหัสแล้ว

ขั้นตอนที่ 2 เลือกไฟล์ที่เข้ารหัสแล้วมาทำการ ถอดรหัสลับและกำหนดค่าเดียวกับตอนเข้ารหัสลับ

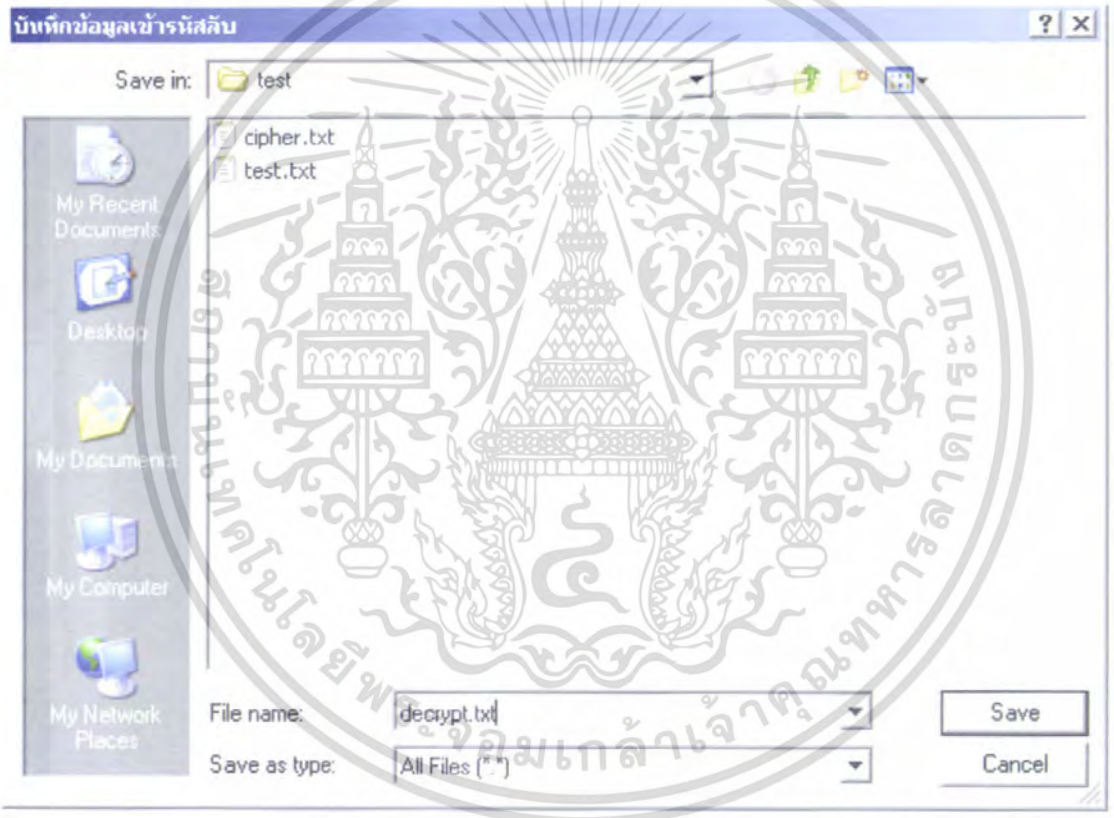


รูปที่ 4-34 การเลือกไฟล์ที่ถูกเข้ารหัสลับแล้วและกำหนดค่า

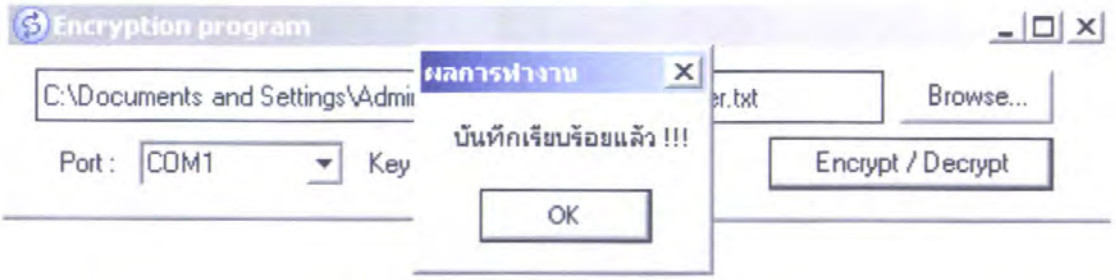


รูปที่ 4-35 การรอขณะทำการถอดรหัสลับ

ขั้นตอนที่ 3 พิมพ์ ชื่อไฟล์ที่ต้องการบันทึกไฟล์ที่ถูกถอดรหัสแล้ว

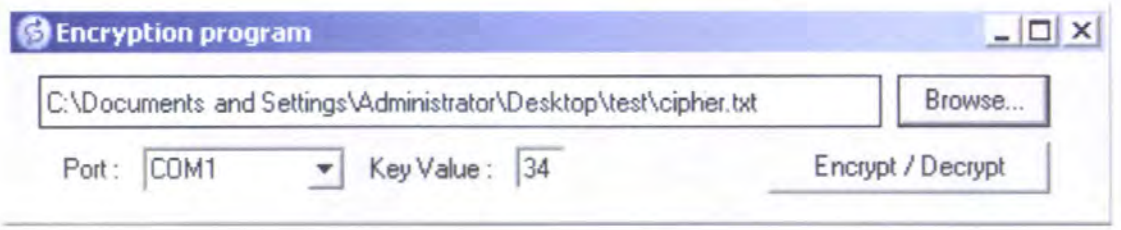


รูปที่ 4-36 การบันทึกไฟล์ที่ถูกถอดรหัสแล้ว



รูปที่ 4-35 หน้าต่างการบันทึกไฟล์เรียบร้อย

4.2.3 การทดลองการถอดรหัสลับไฟล์โดยใส่ค่าเริ่มต้นคนละค่า

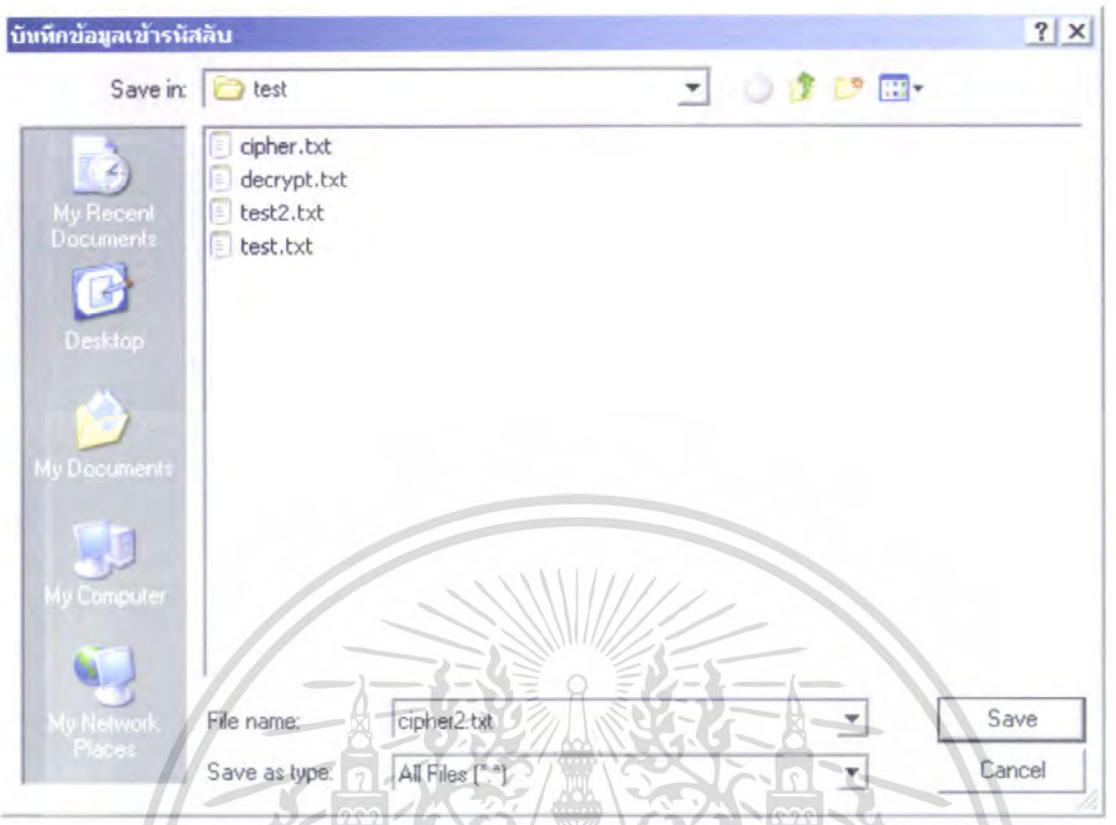


รูปที่ 4-36 การถอดรหัสโดยใส่ค่า เริ่มต้นคนละค่า

รูปที่ 4-37 ไฟล์เมื่อถูกถอดรหัสด้วยค่าเริ่มต้นที่ไม่เหมือนกัน

4.2.4 ทำการเปลี่ยนไฟล์ต้นฉบับโดยใส่ค่าเริ่มต้นเดียวกันแล้วไฟล์ที่ถูกเข้ารหัสมาเปรียบเทียบ

รูปที่ 4-38 การเข้ารหัสไฟล์ใหม่โดยใช้ค่าเริ่มต้นเดียวกัน



รูปที่ 4-39 การบันทึกไฟล์ที่ถูกเข้ารหัสลับแล้ว

รูปที่ 4-40 การเปรียบเทียบระหว่าง ค่าที่ถูกเข้ารหัสโดยใช้ค่าเริ่มต้นเดียวกันแต่ไฟล์ต้นฉบับคนละไฟล์

4.3 สรุปผลการทดลอง

จากกราฟซึ่งได้จากการนำค่าที่ได้จากโปรแกรมซึ่งจะเป็นกุญแจในการเข้ารหัสข้อมูล มาเปรียบเทียบกันในแต่ละระเบียบวิธีทางคณิตศาสตร์ พบว่าแต่ละกราฟมีความแตกต่างกันพอสมควร ถึงแม้จะใช้ค่าเริ่มต้นเดียวกัน

สาเหตุของผลเช่นนี้หนึ่งเกิดจากลักษณะการคำนวณของแต่ละระเบียบวิธี และอีกสาเหตุหนึ่งก็คือความสามารถในการรับค่าของโปรแกรม ซึ่งหากค่าที่รับมานั้นคลาดเคลื่อนเพียงนิดเดียว หรือเพียงแค่การปัดจุดค่าทศนิยม จะทำให้ผลลัพธ์ที่ออกมาเปลี่ยนแปลงโดยสิ้นเชิง ซึ่งลักษณะเหล่านี้เกิดจากคุณสมบัติของสมการเคออสนั่นเอง

สรุป

1.เวลา จากการศึกษาและทดลองพบว่าในแต่ละระเบียบวิธีทางคณิตศาสตร์ในการนำไปแก้ไขสมการเชิงอนุพันธ์สามัญของสมการเคออสต่างๆ เวลาของการทำงานของโปรแกรมบนเครื่องคอมพิวเตอร์นั้นมีผลแทบไม่ต่างกันเลยในหน่วยเวลาเป็นวินาที ซึ่งผลเหล่านี้อาจจะเปลี่ยนแปลงเมื่อนำโปรแกรมนี้ไปทำงานบนไมโครคอนโทรลเลอร์อาร์ม 7

แต่เมื่อนำมาใช้กับไมโครคอนโทลเลอร์ อาร์มเซเว่นแล้วเวลาจะขึ้นอยู่กับขนาดของไฟล์ ซึ่งถ้าไฟล์ใหญ่มากก็จะกินเวลานาน

2. การทดลอง Multi-Chaotic

- จากการทดลอง Multi-chaotic แบบสลับตัวดึงดูด ทุกๆ 333 ค่า จะเห็นได้ว่ากราฟที่ได้ออกมานั้นเป็นการผสมระหว่าง ตัวดึงดูด Chen กับ Lorenz และ Rossler โดยมีจุดตรึง ต่างกันซึ่งจะยังคงความเป็น Chaos อยู่และค่า กุญแจที่ได้ก็จะยิ่ง สุ่มมากขึ้น
- Multi-Chaotic แบบสลับตัวดึงดูด ทุกๆ 1 ค่า จะเห็นได้ว่ากราฟที่ได้ออกมานั้น มีความเป็น Chaos อยู่ก็จริงแต่ค่าที่ได้นั้นมีค่าใกล้เคียงกันมากจึงไม่เหมาะที่จะนำมาใช้กับการเข้ารหัสลับ
- การทดลอง Multi-Chaotic แบบ ขึ้นอยู่กับค่าเริ่มต้น จะเห็นได้ว่า Multi-Chaotic แบบนี้ให้ความสุ่มมาก ซึ่งขึ้นอยู่กับค่า เริ่มต้นที่ได้มาซึ่งจะเลือกรูปแบบของตัวดึงดูดที่ไม่เหมือนกัน ซึ่งทำให้ค่ากุญแจนั้นมีความสุ่มมาก ถึงมากที่สุดจึงจำเหมาะสำหรับ การเข้ารหัสลับข้อมูลที่ต้องการปลอดภัยสูง

3. โปรแกรมเข้ารหัส

จากโปรแกรม เราจะสามารถเข้ารหัสไฟล์ข้อความได้ทั้งภาษาไทยและภาษาอังกฤษ ได้และมีความปลอดภัยสูงแต่ยังมีข้อเสียซึ่งถ้าไฟล์ที่มีขนาดใหญ่มากๆ จะทำให้ใช้เวลาในการเข้ารหัสลับนาน แต่ก็เหมาะสมกับการเก็บให้เป็นความลับซึ่งอัลกอริทึมเราจะสามารถเปลี่ยนรูปแบบได้อีกมากมายซึ่งไม่ได้เปิดเผยในขณะนี้

บทที่ 5
สรุป

5.1 สรุปการพัฒนาโครงงาน

จากโปรแกรมที่สร้างขึ้นเราจะได้ชุดข้อมูลหนึ่ง โดยที่เราเรียกชุดข้อมูลเหล่านี้ว่า กุญแจ ซึ่งกุญแจนี้มีรูปแบบดูเหมือนว่าเป็นค่าที่สุ่ม แต่ในความเป็นจริงค่าเหล่านี้มีรูปแบบที่แน่นอน ซึ่งลักษณะเหล่านี้เป็นไปตามทฤษฎีเคออส เราสามารถนำกุญแจที่สร้างขึ้นนี้ไปรวมกับข้อมูลที่ต้องการทำเป็นความลับ หรือเข้ารหัส ซึ่งเราจะได้ข้อมูลชุดใหม่ ที่มีความปลอดภัยและยากต่อการเข้าใจสำหรับผู้ที่ไม่ได้รับอนุญาตหรือไม่มีกุญแจที่สร้างจากโปรแกรมนี้

กุญแจนี้สามารถปรับแต่งตามอุปกรณ์ฮาร์ดแวร์ เพื่อนำไปใช้เป็นอุปกรณ์ในการเข้ารหัสข้อมูล ซึ่งในโครงงานนี้ได้ใช้ไมโครคอนโทรลเลอร์อาร์ม7 เพื่อสร้างกุญแจให้กับเครื่องคอมพิวเตอร์ จากนั้นคอมพิวเตอร์จะนำกุญแจนี้ไปรวมกับข้อมูลที่ต้องการเข้ารหัสข้อมูล เพื่อสร้างข้อมูลที่มีความปลอดภัย เมื่อได้ข้อมูลที่เข้ารหัสลับแล้วจึงนำข้อมูลนี้ส่งไปให้อุปกรณ์ตัวอื่นหรือคอมพิวเตอร์ที่มีโปรแกรมเข้ารหัสลับนี้เหมือนกันและต้องมีไมโครคอนโทรเลอร์ ARM-7 เหมือนกัน เพื่อจะสามารถถอดรหัสข้อมูลแล้วนำข้อมูลจริงไปใช้

5.2 ปัญหาที่เกิดขึ้นในด้านเทคนิค

ปัญหาในด้านเทคนิคของการสร้างโปรแกรมนี้ส่วนใหญ่เกิดจากข้อจำกัดในความสามารถของภาษาซี เช่น การที่ค่ามีการเพิ่มเกินที่ตัวแปรรับได้ หรือการปัดจุดทศนิยม ซึ่งสิ่งเหล่านี้จะมีผลอย่างมาก เนื่องจากสมการที่ใช้เป็นสมการเคออสซึ่งมีความไวต่อการเปลี่ยนแปลงค่ามาก ปัญหาอีกข้อนึงคือการจับเวลาในการทำงานเนื่องจากโปรแกรมเหล่านี้ทำงานบนคอมพิวเตอร์ซึ่งสภาวะแต่ละครั้งในการทำงานอาจไม่เหมือนกันเนื่องจากระบบปฏิบัติการที่ทำงานมีโปรแกรมย่อยต่างๆ ทำงานอยู่ด้วย ทำให้เวลาที่จับได้นั้นอาจมีความผิดพลาด อีกทั้งหน่วยที่จับเวลายังทำได้แค่เพียงระดับวินาที ซึ่งความละเอียดอาจไม่พอสำหรับในการทำงานจริง

ปัญหาในด้านการเข้ารหัสลับในการใช้ การเข้ารหัสลับแบบนี้จะต้องใช้ กุญแจจำนวนเท่ากับ ความยาวของไฟล์ที่ใช้ในการเข้ารหัสลับไฟล์ ซึ่งทำให้ขนาดของไฟล์ที่ถูกเข้ารหัสและความเร็วในการเข้ารหัสขึ้นอยู่กับขนาดของไฟล์

5.3 แนวทางการพัฒนาต่อ

นำโปรแกรมเข้ารหัสไปประยุกต์ใช้ในการยืนยันความถูกต้องของการเปิดใช้ โปรแกรมอื่นๆ โดยมีการเชื่อมต่อจากอุปกรณ์ ฮาร์ดแวร์ ภายนอก

ตัวอย่างเช่น เมื่อผู้ใช้ต้องการเปิดโปรแกรมหนึ่งจะต้องนำ อุปกรณ์นี้เข้ามาใช้ร่วมในการเปิดโปรแกรมและใช้ฮาดแวร์กับโปรแกรมนั้นตลอดเวลา

บรรณานุกรม

[1] http://hpcmath.kmutt.ac.th/moodle/mod/forum/discuss.php?d=318

[2] Lorenz, E.N (1963).Deterministic nonperiodic flow. *Journal of Atmospheric Scienes* **20**, 130-41.

[3] Lorenz, E.N (1984a). The local structure of a chaotic attractor in four dimensions. *Physica D* **13**, 90-104.

[4] Lorenz, E.N. (1984b). Irregularity: A Fundamental property of the atmosphere. *Tellus* **36A**, 98-110.

[5] Chen, G. and Ueta, T. (1999). Yet another chaotic attractor. *International Journal of Bifurcation and Chaos* **9**, 1465-6.

[6] Chen, G. and Dong, X. (1993). From chaos to order: perspectives and methodologies in Controlling chaotic nonlinear dynamical systems. *International Journal of Bifurcation and Chaos* **3**, 1363-1409.

[7] Chen, G.,Miola ,J.L., and Wang, H. O.(2000). Bifurcation control: theories, methods, and Applications. *International Journal of Bifurcation and Chaos* **10**,511-48.

[8] Rossler, O.E. (1976).An equation for continuous chaos.*Physics Letter A* **57**,397-8.

[9] Rossler, O.E. (1979a).Continuous chaos-four prototype equations. *Annals of the New York Academy Of Sciences* **316**,376-92.

[10] Rossler, O.E. (1979b).An equation for hyperchaos. *Physics Letter A* **71**,155-7.

[11] H.S. Kwok,Wallace K.S(2007). Tang. A fast image encryption system based on chaotic maps with Finite precision representation: Chaos, Solutions & Fractals **32**, 1518-1529 .

[12] http://www.thaicert.nectec.or.th/paper/encryption/intro_crypt.php

[13] Bruce Schneier. 1996. Applied Cryptography. Canada: John Wiley and Sons.

[14] ปราโมทย์ เดชะอำไพ. 2544. ระเบียบวิธีเชิงตัวเลข ในงานวิศวกรรม. พิมพ์ครั้งที่ 3 .กรุงเทพ : โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย

[15] Steve furber .2000 .ARM system-on-chip architecture. Second edition. *British: Addision-Wesley.*

[16] http://www.ett.co.th/product/ARM/CP_JR_ARM7_USB_LPC2148.htm