



โปรแกรมช่วยในการวิเคราะห์ระบบควบคุม

COMPUTER-AIDED CONTROL SYSTEM ANALYSIS

โดย

นายพัฒน์พงษ์ เหล่าสุวรรณ

นายมนี อยู่ประพัฒน์

นายสุบิน เลี้ยงพันธุ์สกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมระบบควบคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

032629



ปริญญานิพนธ์ปีการศึกษา 2535 ภาควิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมช่วยในการวิเคราะห์ระบบควบคุม

ผู้จัดทำ

1. นายพัฒน์พงษ์ เหล่าสุวรรณ รหัส 32-1209
2. นายมนิ อยู่ประพัฒน์ รหัส 32-1243
3. นายสุบิน เลี้ยงพันธุ์สกุล รหัส 32-1381

ผศ.ดร.จกกล งามวิวิทย์ อาจารย์ที่ปรึกษา

(.....)

รศ.วิพันธ์ ปรีชาพานิช อาจารย์ที่ปรึกษา

(.....)

โปรแกรมช่วยในการวิเคราะห์ระบบควบคุม

นายพัฒน์พงษ์ เหล่าสุวรรณ รหัส 32-1209

นายมนี อยู่ประพัฒน์ รหัส 32-1243

นายสุบิน เลี้ยงพันธุ์สกุล รหัส 32-1381

อาจารย์ที่ปรึกษา ผศ.ดร. จงกล งามวิวิทย์

อาจารย์ที่ปรึกษา รศ.วิพันธ์ ปรีชาพานิช

ระดับการศึกษา ปริญญาตรี ภาควิชา วิศวกรรมระบบควบคุม ปีการศึกษา 2535

บทคัดย่อ

โดยทั่วไปการศึกษาทางทฤษฎีวิศวกรรมระบบควบคุมจะเน้นการศึกษาเกี่ยวกับระบบเชิงเส้นไม่แปรตามเวลา แต่ทางปฏิบัติระบบทางกายภาพโดยทั่วไปจะมีลักษณะที่ไม่เป็นเชิงเส้น การทำการวิเคราะห์ระบบที่เป็นเชิงเส้นเป็นการพิจารณาเพื่อให้่ายและสะดวกในการวิเคราะห์เท่านั้น ยิ่งกว่านั้นระบบทางกายภาพโดยทั่วไปมักจะประกอบด้วยอุปกรณ์ที่มีลักษณะดริฟต์ หรือ แปรเปลี่ยนตามเวลา การทำการวิเคราะห์และออกแบบตัวควบคุมสำหรับระบบที่ไม่เป็นเชิงเส้นหรือระบบที่แปรค่าตามเวลาจะมีลักษณะที่ซับซ้อนกว่าระบบที่เป็นเชิงเส้น เป็นอย่างมาก

ในปัจจุบันเรามีโปรแกรมที่ใช้สำหรับการวิเคราะห์ และ ออกแบบระบบควบคุมอยู่หลายโปรแกรมแต่โปรแกรมเหล่านี้มักมีขีดจำกัดในการใช้งาน กล่าวคือสามารถทำการวิเคราะห์ระบบที่มีรูปแบบค่อนข้างตายตัวหรือสามารถทำการวิเคราะห์ระบบที่มีลักษณะเป็นเชิงเส้นเป็นต้น ทางกลุ่มผู้จัดทำได้ตระหนักถึงขีดจำกัดเหล่านี้เป็นอย่างดี จึงได้ทำการพัฒนาและออกแบบโปรแกรมใช้สำหรับวิเคราะห์ระบบขึ้นโดยยึดหลักเพื่อให้โปรแกรมนี้นี้มีความยืดหยุ่นในการทำงานสูงสุด กล่าวคือสามารถทำการวิเคราะห์ระบบที่มีรูปแบบเป็นโครงข่ายเช่นไรก็ได้ ตลอดจนทำการวิเคราะห์ระบบที่ไม่เป็นเชิงเส้นและแปรค่าตามเวลาได้ เป็นต้น นอกจากนี้เรายังสามารถใช้โปรแกรมนี้ช่วยในการแก้ไขปัญหาต่างๆ ทางวิศวกรรมระบบควบคุมได้เป็นอย่างดี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ABSTRACT

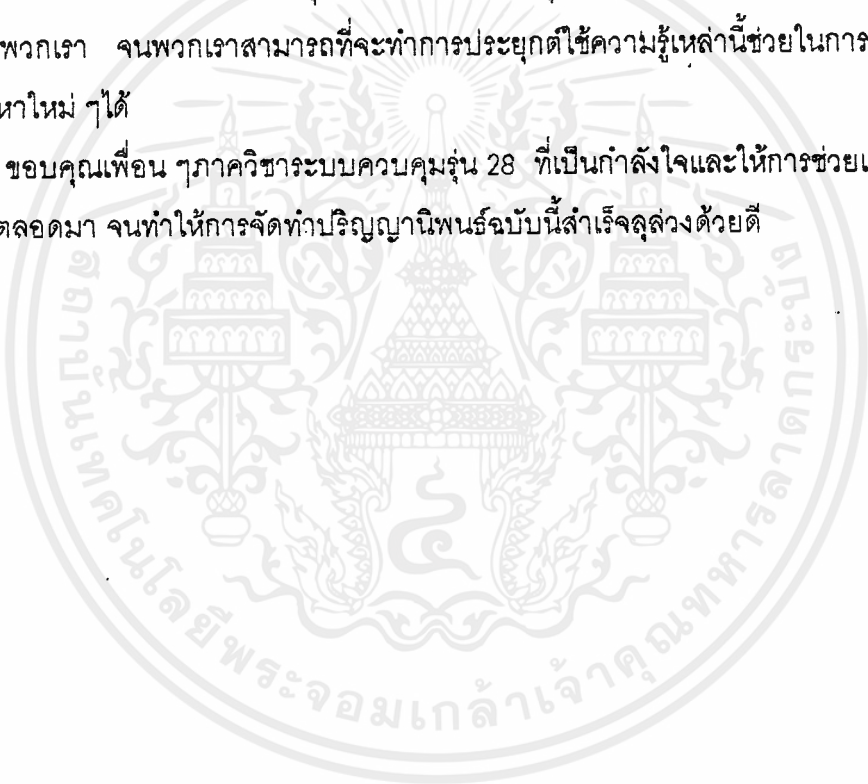
Most of control theory that we have studied nowadays emphasizes on linear-time invariant systems. But linear-time invariant systems do not exist in practice, since all physical systems are non-linear to some extent. Linear feedback control systems are idealized models made by the analyst for the simplicity of analysis and design. Moreover, most physical systems contain elements which drift or vary with time. We have found that the analysis and design of either non-linear system or time-varying system is usually much more complicated than that of the linear-time invariant system. Although nowadays there are many available control system softwares, almost all of them are not elaborated and powerful enough for us to use them for our tasks.

Our group is so concerned about these problems, so we are very enthusiastic to design and develop a software which is capable of analyzing complex non-linear time-varying systems with reasonable speed. This software is also designed to enhance the ability of solving many of the control engineering problems.

กิตติกรรมประกาศ

หากปริญญานิพนธ์ฉบับนี้เป็นประโยชน์ต่อวิทยาการทางวิศวกรรมไฟฟ้าระบบควบคุมแล้ว
ทางกลุ่มผู้จัดทำขออน้อมจิตระลึกถึงตลอดจนขอขอบพระคุณ ท่านอาจารย์ จงกล งามวิวิทย์ และท่าน
อาจารย์ วิพันธ์ ปรีชาพานิช ที่คอยสอนเนื้อหา และให้คำปรึกษาตลอดจนเป็นกำลังใจให้กับกลุ่มผู้
จัดทำจนทำให้ การจัดทำปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี ขอขอบพระคุณท่านอาจารย์
ของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ได้สอนเนื้อหาความรู้พื้นฐาน
ต่าง ๆ ให้กับพวกเรา จนพวกเราสามารถที่จะทำการประยุกต์ใช้ความรู้เหล่านี้ช่วยในการศึกษาและ
เรียนรู้ในเนื้อหาใหม่ ๆ ได้

ขอบคุณเพื่อน ๆ ภาควิชาระบบควบคุมรุ่น 28 ที่เป็นกำลังใจและให้การช่วยเหลือในสิ่ง
ต่าง ๆ ด้วยดีตลอดมา จนทำให้การจัดทำปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี



สารบัญ

	หน้า
บทนำ	11
บทที่ 1 การวิเคราะห์แบบเฟสเพลน	
1.1 บทนำ	31
1.2 การแปลงสมการอนุพันธ์อันดับสองด้วยสมการอันดับหนึ่ง	51
1.3 คำนิยาม	51
1.4 แนวคิด เรื่องโมดกับการวิเคราะห์เฟสเพลน	61
1.5 การพิจารณา phase portrait สำหรับระบบโดยทั่วไป	121
1.6 การพิจารณาค่าเวลาบนเฟส portrait	131
บทที่ 2 วิธีการวิเคราะห์แบบเดสไคร์มิงฟังก์ชัน	
2.1 บทนำ	181
2.2 นิยามของเดสไคร์มิงฟังก์ชัน	181
2.3 เดสไคร์มิงฟังก์ชันสำหรับระบบไม่เป็นเชิงเส้นที่ นำสนใจ	191
2.4 การประยุกต์ใช้เดสไคร์มิงฟังก์ชันในการวิเคราะห์ เสถียรภาพ	211
บทที่ 3 การวิเคราะห์เสถียรภาพโดยใช้ทฤษฎีลีอาปูนอฟ	
3.1 บทนำ	271
3.2 แนวความคิดเกี่ยวกับเสถียรภาพของลีอาปูนอฟ	271
3.3 การหาเงื่อนไขเสถียรภาพของลิมิตไซเคิล	35
3.4 ฟังก์ชันลีอาปูนอฟสำหรับระบบต่อเนื่องเชิงเส้น ไม่ขึ้นกับเวลา	36
3.5 วิธีการของคราซอฟสกี	361
บทที่ 4 การควบคุมระบบที่ไม่เป็นเชิงเส้น	
4.1 บทนำ	371
4.2 ทฤษฎีที่ใช้มากสำหรับการออกแบบระบบควบคุม	41

4.3 ระบบควบคุมแบบมีโมเดลอ้างอิงสำหรับระบบอันดับหนึ่ง	42
4.4. ระบบควบคุมแบบมีโมเดลอ้างอิงสำหรับระบบไม่เป็นเชิงเส้นอันดับ n โดย	45
4.5 เทคนิคที่ใช้ในการประมาณค่าพารามิเตอร์ของระบบ	47
4.5.1 โมเดลการประมาณค่าพารามิเตอร์ให้เป็นเชิงเส้น	48
4.5.2 การทำนายค่าผิดพลาดโดยใช้วิธีการประมาณ	48
4.6 การควบคุมการปรับตัวแบบคอมโพสิต	51
บทที่ 5 การคำนวณและการสร้าง	
5.1 การสร้างอัลกอริทึมสำหรับการวิเคราะห์เฟสเพลน	53
5.2 วิธีการเชิงเลขสำหรับการหาคำตอบของสมการอนุพันธ์	55
5.3 การแปลงดิสครีตฟูเรียร์	56
บทที่ 6 การใช้งานโปรแกรม Simulation	58
บทที่ 7 ตัวอย่างโปรแกรมและผลการทดลอง	
7.1 บทนำ	61
7.2 ตัวอย่างการใช้โปรแกรมวิเคราะห์ระบบต่างๆ	61
บทที่ 8 บทสรุปและวิจารณ์	127
ภาคผนวก	
หนังสืออ้างอิง	

บทนำ

ในการวิเคราะห์และออกแบบระบบควบคุม สิ่งสำคัญสิ่งหนึ่งที่เราจำเป็นต้องทราบคือคุณสมบัติของระบบที่เราจะทำการควบคุมว่ามีลักษณะเป็นเช่นไร โดยทั่วไปเราสามารถแบ่งระบบออกได้หลายประเภทขึ้นกับเกณฑ์ที่เราใช้ในการพิจารณาวิธีหนึ่งที่เรานิยมใช้กันคือ การพิจารณาถึงความ เป็นเชิงเส้นของระบบเมื่อใช้เกณฑ์นี้ในการพิจารณาเราสามารถจะแบ่งระบบออกเป็น 2 ประเภทคือ

- 1.ระบบที่เป็นเชิงเส้น (linear system) เป็นระบบที่สอดคล้องกับทฤษฎีซูเปอร์โพสิชัน
- 2.ระบบที่ไม่เป็นเชิงเส้น (nonlinear system) เป็นระบบที่ไม่สอดคล้องกับทฤษฎีซูเปอร์โพสิชัน

ระบบที่เป็นเชิงเส้นจะมีวิธีในการวิเคราะห์และออกแบบค่อนข้างแน่นอนกล่าวคือ

หากเป็นระบบต่อเนื่องอินพุตเดียวเอาต์พุตเดียว เราจะใช้ลาปลาซทรานสฟอร์มในการวิเคราะห์ หากเป็นระบบไม่ต่อเนื่องอินพุตเดียวเอาต์พุตเดียว เราจะใช้แซดทรานสฟอร์มในการวิเคราะห์และหากระบบนั้นมีอินพุตเอาต์พุตหลายตัวเราจะใช้วิธีการสเตทสเปซในการวิเคราะห์ เมื่อเราพิจารณาในแง่ของเสถียรภาพเราสามารถตรวจสอบโดยใช้วิธีการต่างๆ เช่น Routh criterion, Root locus, Nyquist plot หรือ Bode plot เป็นต้น ระบบที่ไม่เป็นเชิงเส้นจะมีวิธีในการวิเคราะห์มากมายขึ้นอยู่กับลักษณะของระบบ วิธีหนึ่งอาจใช้วิเคราะห์ระบบประเภทหนึ่ง แต่อาจไม่เหมาะสมสำหรับระบบประเภทอื่นๆ ซึ่งเห็นได้ชัดว่าการวิเคราะห์และออกแบบระบบจะซับซ้อนกว่าระบบที่เป็นเชิงเส้นอย่างมาก

สำหรับปริญญาโทฉบับนี้เป็นการวิเคราะห์ และออกแบบระบบที่ไม่เป็นเชิงเส้นโดยจะประกอบด้วยทฤษฎีในการวิเคราะห์ระบบสามวิธีและวิธีการออกแบบตัวควบคุมโดยใช้ตัวควบคุมที่มีลักษณะแบบปรับตัวเอง นอกจากนี้จะมีโปรแกรมวิเคราะห์ระบบสำหรับการช่วยในการศึกษาระบบเป็นไปอย่างมีประสิทธิภาพ รายละเอียดของส่วนต่าง ๆ สามารถสรุปได้ดังนี้

1. การวิเคราะห์แบบเฟสเพลน (PHASE PLANES) วิธีนี้จะใช้ในการวิเคราะห์ระบบที่เป็นอันดับสองและเราสามารถพิจารณาถึงเสถียรภาพของระบบได้โดยคร่าว ๆ
2. วิธีการวิเคราะห์แบบเดสไคริบิงฟังก์ชัน (DESCRIBING FUNCTION) วิธีนี้จะใช้ในการวิเคราะห์ระบบที่เกี่ยวข้องกับส่วนที่ไม่เป็นเชิงเส้น ซึ่งเกิดขึ้นจากตัวพลานท์ คอนโทรลเลอร์ หรือตัวเซนส์เซอร์ วิธีนี้จะประมาณระบบที่ไม่เป็นเชิงเส้นให้เป็นระบบที่เป็นเชิงเส้นภายใต้ข้อสมมติฐานต่างๆ
3. การวิเคราะห์เสถียรภาพโดยวิธีลียาปูนอฟ จะประกอบด้วยวิธีการวิเคราะห์สองวิธีกล่าวคือวิธีการประมาณเชิงเส้นและวิธีการโดยตรง

4. วิธีการออกแบบตัวควบคุม จะเป็นตัวควบคุมที่มีลักษณะปรับตัวเองได้ โดยจะประกอบ ด้วยตัวควบคุมที่เรียกว่า MRAC และตัวควบคุมที่เราเรียกว่า COMPOSITE ADAPTIVE

5. โปรแกรมวิเคราะห์ระบบเป็นโปรแกรมที่เขียนขึ้นด้วยภาษาซี โดยมีจุดมุ่งหมายเพื่อช่วย ในการวิเคราะห์และออกแบบระบบควบคุมเป็นไปด้วยความคล่องตัวและมีประสิทธิภาพ อาทิเช่น

- ผู้ใช้สามารถกำหนดให้ระบบควบคุมแบบป้อนกลับที่มีลักษณะเป็นโครงข่ายเช่นใดก็ได้ กล่าวคือ สามารถกำหนดลักษณะของแต่ละบล็อกไดโอดแกรมในรูปของตัวแปรสเกลเช่นใดก็ได้ สามารถกำหนดส มการของสัญญาณอินพุตในรูปฟังก์ชันของเวลา ซึ่งมีความสัมพันธ์ทางคณิตศาสตร์เป็นเช่นไรก็ได้ เป็นต้น

- ผู้ใช้สามารถทำการวิเคราะห์ระบบ โดยที่องค์ประกอบต่างๆ ของระบบเป็นไปดังที่ได้ก่ าหนดไว้แล้วข้างต้น การวิเคราะห์จะประกอบด้วยการวิเคราะห์ผลตอบสนองทางเวลาการวิเคราะห์ผล ตอบสนองทางความถี่ อาทิเช่น สามารถพิจารณาลักษณะของทางเดินของตัวแปรสเกล(กรณีทีระบบ เป็นอันดับสอง),สามารถพิจารณาผลตอบสนองทางเวลาของสัญญาณต่าง ๆ ในระบบ กล่าวคือเรา สามารถที่จะเลือกทำการพิจารณาผลตอบสนองของเอาต์พุต หรือผลตอบสนองของตัวแปรสภาวะ ต่าง ๆ ภายในระบบ ตลอดจนสามารถพิจารณาผลตอบสนองที่เป็นความสัมพันธ์ทางคณิตศาสตร์ ของผลตอบสนองแต่ละปริมาณก็ได้ และสามารถทำการวิเคราะห์เกี่ยวกับสเปกตรัมของสัญญาณ ต่างๆที่เราสนใจ โดยอาศัยอาศัยการแปลงฟูริเยร์ เป็นต้น

บทที่ 1

การวิเคราะห์เฟสเพลน(PHASE PLANES ANALYSIS)

1.1. บทนำ

ระบบโดยทั่วไปที่สามารถจะแสดงได้ด้วยสมการดิฟเฟอเรนเชียลอันดับ n เมื่อเราวิเคราะห์ระบบโดยใช้สเตทสเปซ จะสามารถแสดงระบบด้วยสมการดิฟเฟอเรนเชียลอันดับ 1 ซึ่งเขียนในรูปของเมตริกซ์ โดยสเตทเวกเตอร์จะมีองค์ประกอบ n ส่วน การจะแสดงค่าของตัวแปรสเตทจะต้องอาศัยมิติที่มีขนาด n มิติ สำหรับระบบที่เราสามารถเขียนเป็นสมการดิฟเฟอเรนเชียลอันดับ 2 เมื่อวิเคราะห์ในรูปสมการสเตทจะมีตัวแปรสเตท 2 ตัว การจะแสดงค่าของตัวแปรสเตทต้องอาศัยมิติที่มีขนาด 2 มิตินั้นคือเราสามารถจะแสดงลักษณะของตัวแปรสเตทลงบนระนาบและเราจะเรียกระนาบนี้ว่าเฟสเพลน (PHASE PLANE)

พิจารณาระบบเชิงเส้นอันดับสองซึ่งมีสมการเป็น

$$\frac{d^2\theta}{dt^2} + 2\xi\omega_n\frac{d\theta}{dt} + \omega_n^2\theta(t) = 0$$

เพื่อความสะดวกในการพิจารณาเราจะแบ่งการพิจารณาออกเป็น 2 กรณีคือ

(1) กรณี undamped ($\xi=0$) นั่นคือจะได้สมการเป็น

$$\frac{d^2\theta}{dt^2} + \omega_n^2\theta(t) = 0$$

จากความรู้ทางแคลคูลัสเราสามารถคำนวณคำตอบของสมการข้างต้นได้โดยการสมมติให้

$$\theta(t) = R \sin(\omega_n t + \Phi)$$

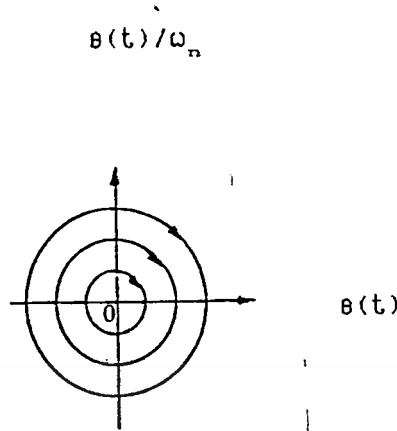
ฉะนั้น

$$\frac{d\theta}{dt} = R\omega_n \cos(\omega_n t + \Phi)$$

กำจัดตัวแปรเวลาออกจากสมการทั้งสองข้างต้น จะได้สมการเป็น

$$\left[\frac{1}{\omega_n} \frac{d\theta}{dt}\right]^2 + \theta(t)^2 = R^2$$

เมื่อเรานำค่าของ $\theta(t)$ และ $\theta(t)/\omega_n$ ไปวาดกราฟในระนาบจะได้กราฟเป็นวงกลมซึ่งมีรัศมีต่าง ๆ กันดังรูป



รูปที่ 1

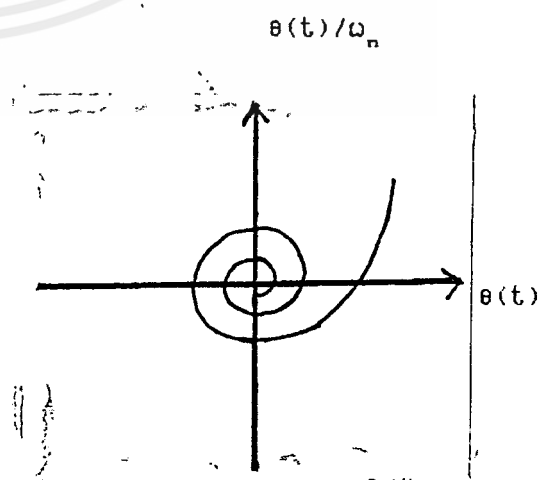
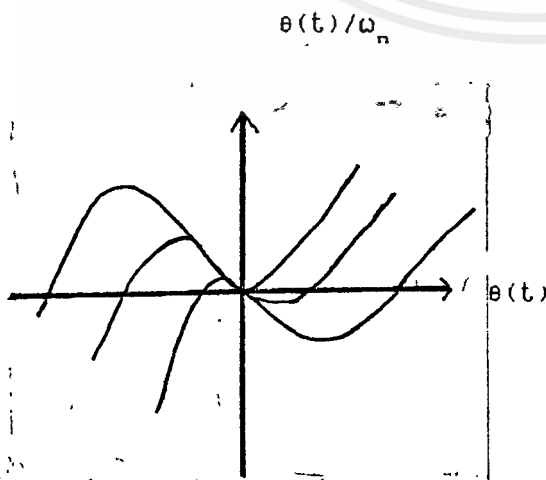
จะเห็นได้ว่าเมื่อเราพิจารณาเงื่อนไขเริ่มต้นที่แตกต่างกันไป จะได้กลุ่มของกราฟที่แสดงถึงการเปลี่ยนแปลงของตัวแปรสเตตซึ่งเราจะเรียกว่า phase-portrait

ข้อสังเกต เราสามารถพิจารณาถึงเสถียรภาพโดยการพิจารณาจาก phase-portrait ได้ นั่นคือหากทิศทางการเคลื่อนที่ของกราฟมีทิศทางไปยังอนันต์แสดงว่าระบบนั้นไม่เสถียร ในทางตรงกันข้ามหากทิศทางการเคลื่อนที่ของกราฟมีทิศมุ่งไปยังจุดใดจุดหนึ่ง จะแสดงว่าระบบนั้นเสถียรนั่นเอง

2) กรณี เป็นค่าใดๆ เราจะได้สมการเป็น

$$\frac{d^2\theta}{dt^2} + 2\zeta\omega_n\dot{\theta} + \omega_n^2\theta = 0$$

เห็นได้ว่าคำตอบกรณีนี้จะอยู่ในรูปของเวลาโดยที่เราไม่สามารถกำหนดเวลาออกจากสมการทั้งสองได้ การพล็อต phase-portrait กรณีนี้ทำได้โดยการแก้สมการหา $\theta(t)$ และนำ $\theta(t)$ มาหาค่า $\dot{\theta}(t)$ ซึ่งค่า $\theta(t)$ และ $\dot{\theta}(t)$ ที่เวลาหนึ่งๆจะเป็นจุดหนึ่งบน phase-portrait จากนั้นทำการเปลี่ยนแปลงค่าเวลา แลทำในทำนองเดียวกันไปเรื่อย ๆ ผลลัพธ์ที่ได้จะเป็น phase-portrait แสดงดังรูปที่ 2



ข้อสังเกต 1. phase-portrait จะเป็นกราฟซึ่งเป็นผลรวมของกราฟย่อย ๆ ที่ไม่มีการตัดกัน โดยกราฟย่อยแต่ละเส้นนี้จะขึ้นอยู่กับ initial condition ที่แตกต่างกันนั่นเอง

2. สำหรับระบบที่ยกตัวอย่างนี้ จะเห็นว่าลักษณะกราฟลู่เข้าหา origin แสดงว่าระบบนี้เป็นระบบที่เสถียรนั่นเอง จะเห็นได้ว่าโดยทั่ว ๆ ไปการที่จะพล็อต phase-portrait จากการหาคำตอบ ของสมการดิฟเฟอเรนเชียลโดยตรงไม่ใช่เป็นเรื่องที่ยากด่ายเลย แม้ว่าเราจะหาคำตอบของสมการนั้นได้ ยิ่งกว่านั้นหากเราไม่สามารถหาคำตอบของสมการนั้นในรูป close-form ได้ การพล็อต phase-portrait จะแทบเป็นไปไม่ได้เลย ฉะนั้นเราจำเป็นต้องมีวิธีในการกำจัดตัวแปร t ออกจากสมการ ดังจะกล่าวในหัวข้อต่อไป

1.2 การแปลงสมการอนุพันธ์อันดับสองไปเป็นสมการอันดับหนึ่ง

พิจารณาระบบสมการอนุพันธ์อันดับสอง ที่มีรูปทั่วไปเป็น

$$\ddot{x} = f(x, \dot{x}) \quad \text{----- (3)}$$

กำหนดให้ $\dot{x} = y$

$$\dot{y} = \ddot{x}$$

จะได้ว่า $\frac{\dot{y}}{\dot{x}} = \frac{f(x,y)}{y} \quad \text{----- (4)}$

คำตอบของสมการ (4) จะเป็นความสัมพันธ์ของตัวแปร x และ y ซึ่งใช้ในการวาด phase-portrait นั่นเอง

1.3 คำนิยาม

ในการศึกษาระบบที่เป็นระบบ NON LINEAR เราควรจะทราบนิยามและความหมายของคำเฉพาะต่าง ๆ เพื่อให้การศึกษาง่ายยิ่งขึ้น ฉะนั้นในหัวข้อนี้จะเป็นการสรุปนิยาม ของความหมาย คำต่าง ๆ ซึ่งเรามักพบเสมอในการศึกษาระบบที่เป็น NON LINEAR ดังต่อไปนี้

- AUTONOMOUS EQUATION คือสมการที่ค่าของ x จะไม่เป็นความสัมพันธ์กับเวลา นั่นคือสามารถเขียนได้เป็น

$$\ddot{x} = f(x, \dot{x}) \quad \text{----- (5)}$$

- NON AUTONOMOUS EQUATION คือสมการที่ค่าของ x จะเป็นความสัมพันธ์กับเวลา นั่นคือสามารถเขียนได้เป็น

$$\ddot{x} = f(x, \dot{x}, t) \quad \text{----- (6)}$$

- EQUILIBRIUM ระบบจะกล่าวว่ายู่ในภาวะ equilibrium ถ้าหากตัวแปรสเทตของระบบนั้นหยุดนิ่งไม่เปลี่ยนแปลงไปกับเวลา นั่นคือถ้าหากระบบมีลักษณะเป็น

$$\dot{x} = f(x, \dot{x}, t) \quad \text{----- (7)}$$

เราจะกล่าวได้ว่า equilibrium state จะสอดคล้องกับคำตอบที่เป็นค่าคงที่ของสมการ (7) นั่นคือจะสอดคล้องกับสมการ

$$f(x, 0, t) \quad \text{----- (8)}$$

- SINGULAR POINT เป็นจุดที่อยู่ใน PHASE PLANE โดยที่ระบบจะอยู่ในสภาวะ equilibrium ณ จุดนี้ทั้งความเร็วและความเร่งของระบบจะมีค่าเป็นศูนย์ นั่นคือจุด singular point จะเป็นจุดที่เป็น equilibrium state นั่นเอง สำหรับระบบที่เป็น linear จุด singular-point จะมีเพียงจุดเดียวคือจุด origin นั่นเอง

1.4 แนวคิดเรื่องโมดกับการวิเคราะห์ PHASE PLANE

โดยทั่วไประบบ line-time invariant ที่มีรูปแบบทั่วไปเป็น

$$\text{state eqn : } \dot{x} = Ax + Bu \quad \text{----- (9)}$$

$$\text{output eqn: } y = Cx + Du \quad \text{----- (10)}$$

กรณีระบบเป็น second order และเป็น autonomous type เราจะได้สมการมีลักษณะเป็น

$$\begin{aligned} \dot{x}_1 &= Ax_1 + Bx_2 \\ \dot{x}_2 &= Cx_1 + Dx_2 \\ y &= C_0x_1 + C_1x_2 \end{aligned} \quad \text{----- (11)}$$

ค่า eigenvalue ของระบบนี้จะสอดคล้องกับสมการ

$$\lambda^2 - (a + d)\lambda + (ad - bc) = 0$$

สำหรับค่า eigenvalue แต่ละตัวจะมี eigenvector ที่สอดคล้องกับมัน เนื่องจาก eigenvector แต่ละตัวมีลักษณะเป็นอิสระเชิงเส้น ฉะนั้นเราจึงสามารถนำเซตของ eigenvector ทั้งหมดมาเป็น basis สำหรับ space n มิติได้ สำหรับระบบอันดับสองก็เช่นเดียวกันเราสามารถเขียน $x(t)$ และ $Bu(t)$ ในรูป linear combination ของ basis ได้เป็น

$$x(t) = z_1(t)m_1 + z_2(t)m_2 \quad \text{----- (12)}$$

$$Bu(t) = w_1(t)m_1 + w_2(t)m_2 \quad \text{----- (13)}$$

โดย m_1 และ m_2 เป็น eigenvector ที่สอดคล้องกับ eigenvalue ตามลำดับนำสมการ (11) และ (12) แทนลงในสมการ (9) จะได้เป็น

$$\dot{z}(t)_1 m_1 + \dot{z}(t)_2 m_2 = A z_1(t) m_1 + A z_2(t) m_2 + W_1(t) m_1 + W_2(t) m_2$$

$$[\dot{z}_1(t) m_1 - A z_1(t) m_1 - W_1(t) m_1] + [\dot{z}_2(t) m_2 - A z_2(t) m_2 + W_2(t) m_2] = 0$$

เราจะพิจารณาเป็น 2 กรณีคือ

กรณีที่ 1 ค่า eigenvalue มีค่าต่างกัน จากคุณสมบัติของ eigenvalue เราได้ว่า

$$A m_i = \lambda_i m_i$$

ฉะนั้นเราจะได้ว่า

$$(z_1(t) - \lambda_1 z_1 - W_1) m_1 + (z_2(t) - \lambda_2 z_2 - W_2) m_2 = 0$$

เนื่องจากคุณสมบัติของ eigenvector m_1, m_2 ที่เป็นอิสระต่อกัน เราจะได้เป็น

$$\dot{z}_1(t) = z_1 \lambda_1 + W_1$$

$$\dot{z}_2(t) = z_2 \lambda_2 + W_2$$

นั่นคือเราจะได้ว่า

$$z_i(t) = e^{\lambda_i t} z_i(0) + \int e^{\lambda_i t} w_i(\tau) d\tau$$

นำไปแทนในสมการ (12) จะได้ว่า

$$x(t) = [e^{\lambda_1 t} z_1(0) + \int e^{\lambda_1 t} w_1(\tau) d\tau] m_1 + [e^{\lambda_2 t} z_2(0) + \int e^{\lambda_2 t} w_2(\tau) d\tau] m_2$$

โดย $z_i(0)$ เป็นองค์ประกอบของ $x(0)$ ในทิศทางของ m_i นั้น

$w_i(t)$ เป็นองค์ประกอบของ $Bu(t)$ ในทิศทางของ m_i นั้น

นั่นคือเราสรุปได้ว่า $x(t)$ จะมีองค์ประกอบ ส่วนซึ่งเป็นอิสระต่อกันโดยองค์ประกอบแต่ละส่วนจะขึ้นกับค่า eigenvalue แต่ละค่ามันเอง สำหรับระบบเชิงเส้นที่ไม่มีอินพุตและมี equilibrium point อยู่ที่จุดออร์จิน $x(t)$ สามารถลดรูปเหลือเพียง

$$x(t) = e^{\lambda_1 t} z_1(0) m_1 + e^{\lambda_2 t} z_2(0) m_2 \quad \text{----- (15)}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (7) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กรณีที่ 2 ค่า eigenvalue มีค่าเหมือนกัน จากคุณสมบัติของ eigenvalue เราได้ว่า

$$Am_1 = \lambda_1 m_1$$

$$Am_2 = \lambda_2 m_2$$

เพื่อความสะดวก เราจะให้ระบบที่กำลังพิจารณากลายเป็นระบบเชิงเส้นที่ไม่มีอินพุต ฉะนั้นจะได้ว่า

$$\dot{z}_1(t) = z_1 \lambda_1 + z_2$$

$$\dot{z}_2(t) = z_2 \lambda_1$$

นั่นคือจะได้ว่า

$$z_2(t) = z_2(0)e^{\lambda_1 t}$$

$$z_1(t) = z_1(0)e^{\lambda_1 t} + z_2(0)te^{\lambda_1 t}$$

ดังนั้นเราจะได้ว่า

$$x(t) = (z_1(0)e^{\lambda_1 t} + z_2(0)te^{\lambda_1 t})m_1 + z_2(0)e^{\lambda_1 t})m_2 \quad \text{---- (16)}$$

เพื่อความสะดวกในการวิเคราะห์ เราจะแยกการพิจารณาดังนี้

1. eigenvalue เป็นค่าจริงต่างกัน ($(a+d)^2 > 4(ad-bc)$)

เราพิจารณากเป็นกรณีต่าง ๆ ได้ดังนี้

กรณีที่ 1 ค่า eigenvalue มีค่าเป็นบวกทั้งคู่ เราทำการวิเคราะห์ได้ดังนี้

- พิจารณาจากสมการ (15) เห็นได้ว่าเมื่อเวลาที่มีค่าเข้าใกล้อนันต์ จะได้ว่าองค์ประกอบของ $x(t)$ ในทิศทาง m_1 และ m_2 จะเข้าใกล้อนันต์ด้วย นั่นคือจะได้ว่าทิศทางของลูกศรจะพุ่งออกจากจุด singular point

- พิจารณาจากสมการ (15) เห็นได้ว่าเมื่อเวลาเข้าใกล้อนันต์ ค่าเอกซ์โปเนนเชียลของ λ ที่มีค่ามากจะมีอัตราการเพิ่มที่เร็วกว่า นั่นคือทิศทางของ $x(t)$ จะอยู่ในแนวเดียวกับ eigenvector ที่สอดคล้องกับ λ ที่มีค่ามากเมื่อเวลา t มีค่ามาก ๆ

- จุด singular point กรณีนี้ เราเรียกว่า unstable node

- กรณีจุดเริ่มต้นอยู่ในทิศเดียวกับ eigenvector ตัวใดตัวหนึ่ง จะได้ว่าองค์ประกอบของจุดเริ่มต้นนั้นในทิศของ eigenvector อีกตัวนั้นจะเป็นศูนย์ (พิจารณาจาก (15)) ดังนั้นทิศทางการเคลื่อนที่ของกราฟจะอยู่เฉพาะในแนวเดียวกับ eigenvector ตัวนั้นเท่านั้น นั่นคือกราฟจะอยู่ในแนว



เส้นตรง

กรณีที่ 2 ค่า eigenvector เป็นค่าลบทั้งคู่ เราทำการวิเคราะห์ได้ดังนี้

-พิจารณาจากสมการ (15) เห็นได้ว่าเมื่อเวลา t มีค่าเข้าใกล้อนันต์ จะได้ว่าองค์ประกอบของ $x(t)$ ในทิศทาง m_1 และ m_2 จะเข้าใกล้ศูนย์ด้วย นั่นคือจะได้ว่าทิศทางของลูกศรจะพุ่งเข้าหาจุด singular point

-พิจารณาจากสมการ (15) เห็นได้ว่าเมื่อเวลา t เข้าใกล้อนันต์ ค่าเอกซ์โปเนนเชียลของที่มีค่าติดลบน้อย(ขนาดน้อยกว่า)จะมีอัตราการลดที่ช้ากว่า นั่นคือทิศทางของ $x(t)$ จะอยู่ในแนวเดียวกับ eigenvector ที่สอดคล้องกับ ที่มีค่าติดลบน้อยนั่นเอง

-กรณีจุดเริ่มต้นอยู่ในทิศเดียวกับ eigenvector ตัวใดตัวหนึ่ง จะได้ว่าองค์ประกอบของจุดเริ่มต้นนั้นในทิศของ eigenvector อีกตัวหนึ่งจะเป็นศูนย์ (พิจารณาจาก (12)) ดังนั้นทิศทางการเคลื่อนที่ของกราฟจะอยู่เฉพาะในแนวเดียวกับ eigenvector ตัวนั้นเท่านั้น

-จุด singular point กรณีนี้ เราเรียกว่า stable node

กรณีที่ 3 ค่า eigenvalue มีเครื่องหมายต่างกัน เราทำการวิเคราะห์ได้ดังนี้

-กรณีจุดเริ่มต้นอยู่ในทิศเดียวกับ eigenvector ตัวที่สอดคล้องกับ eigenvalue ที่เป็นบวก ทิศทางของ $x(t)$ จะอยู่ในแนวเดียวกับ eigenvector ตัวนี้ โดยมีทิศทางพุ่งออกจากจุด singular point

-กรณีจุดเริ่มต้นอยู่ในทิศเดียวกับ eigenvector ตัวที่สอดคล้องกับ eigenvalue ที่มีค่าเป็นลบ ทิศทางของ $x(t)$ จะอยู่ในแนวเดียวกับ eigenvector ตัวนี้ โดยมีทิศทางพุ่งเข้าหาจุด singular point

-กรณีจุดเริ่มต้นอยู่ใกล้กันกับ eigenvector ตัวที่สอดคล้องกับ eigenvalue ที่มีค่าเป็นลบ เมื่อพิจารณาจากสมการ (12) และ (15) จะได้ว่า

เมื่อเวลา t มีค่าน้อย : eigenvector ตัวที่สอดคล้องกับ eigenvalue ที่มีค่าเป็นลบจะมีอิทธิพลต่อ $x(t)$ มากกว่า

เมื่อเวลา t มีค่ามาก : eigenvector ตัวที่สอดคล้องกับ eigenvalue ที่มีค่าเป็นบวกจะมีอิทธิพลต่อ $x(t)$ มากกว่า

-จุด singular point กรณีนี้ เราเรียกว่า saddle point

กรณีที่ 4 กรณีที่มีค่า eigenvalue ค่าหนึ่งเป็นศูนย์ (สมมติให้ $\lambda = 0$)

กรณีจุดเริ่มต้นอยู่ในทิศเดียวกับ eigenvector ตัวที่สอดคล้องกับ eigenvalue ที่มีค่าเป็นศูนย์ กรณีนี้ $x(t)$ จะเป็นค่าคงที่อธิบายได้โดยสมการ (15)

-กรณีจุดเริ่มต้นอยู่ในทิศทางอื่น พิจารณาจากสมการ (15) จะได้ว่าทิศทางของกราฟจะเคลื่อน

ที่ขนานกับ eigenvector m_2 เพราะองค์ประกอบในทิศ m_2 เปลี่ยนแปลง แต่องค์ประกอบใน ทิศ m_1 คงที่

และเมื่อเวลาเข้าใกล้อนันต์ จะได้ว่า $x(t)$ จะเป็นค่าคงที่ซึ่งเกิดจากองค์ประกอบในทิศ m_1 โดยมีค่าเป็น

$$x(t) = z_1(0)m_1$$

2. กรณีค่า eigenvalue ซ้ำกัน $[(a+d)^2 = 4(ad-bc)]$

กรณีที่ 1 กรณีที่มีค่า eigenvalue เป็นบวก

- พิจารณาจากสมการ (16) จะเห็นได้ว่ากรณีนี้ทิศทางของลูกศรจะพุ่งออกจากจุด singular point และเมื่อเวลาเข้าใกล้อนันต์ทิศทางการเคลื่อนที่ของกราฟจะค่อนไปทางทิศทางของ eigenvector m_2

- จุด singular point กรณีนี้เราเรียกว่า node

กรณีที่ 2 กรณีที่ค่า eigenvector เป็นลบ

- พิจารณาจากสมการ (16) จะได้ว่ากรณีนี้ทิศทางของลูกศรจะพุ่งออกจาก singular point

- จุด singular point กรณีนี้เราเรียกว่า node

หมายเหตุ: พิจารณากรณี decouple (diagnol) นั่นคือ

$$A = \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} \quad \text{--- (17)}$$

เราพิจารณาโดยเขียนสมการ (17) ใหม่เป็น

$$\dot{x}_1 = ax_1$$

$$\dot{x}_2 = ax_2$$

จะได้เป็น

$$\frac{dx_1}{dx_2} = \frac{x_1}{x_2}$$

แก้สมการดิฟเฟอเรนเชียลได้เป็น

$$x_1 = cx_2$$

โดยค่าคงที่ c หาได้จาก $c = x_2(0)/x_1(0)$

เราจะได้ลักษณะกราฟเป็นเส้นตรงที่มีความชันต่าง ๆ โดยที่

(10)

ถ้า $a < 0$: ทิศทางของกราฟจะพุ่งเข้าหาจุด singular point

ถ้า $a > 0$: ทิศทางของกราฟจะพุ่งออกจากจุด singular point

3. กรณีค่า eigenvalue เป็นจำนวนเชิงซ้อน $[(a + d)^2 < 4(ad - bc)]$

สมมติให้ค่า eigenvalue มีค่าเป็น $-\alpha + j\beta$

กำหนดให้ transform เมตริกซ์เป็น

$$T = \langle \mu, v \rangle$$

โดย μ และ v เป็นส่วนจริงและส่วนจินตภาพของ eigenvector ที่สอดคล้องกับค่า eigenvalue แต่ละค่าตามลำดับ ภายใต้การ transform จะได้ระบบอยู่ในรูป

$$\dot{\mathbf{x}} = \begin{bmatrix} -\alpha & \beta \\ -\beta & -\alpha \end{bmatrix} \mathbf{x}$$

พิจารณาแยกทีละสเกล

$$\dot{x}_1 = -\alpha x_1 + \beta x_2$$

$$\dot{x}_2 = -\beta x_1 - \alpha x_2$$

เพื่อความสะดวกในการพิจารณาลักษณะของ phase-portrait เราจะพิจารณาในระนาบเชิงซ้อน โดยกำหนดให้ตัวแปร

$$z = x_1 + jx_2 \quad \text{--- (18)}$$

$$\text{นั่นคือ} \quad \dot{z} = (-\alpha x_1 + \beta x_2) + j(-\beta x_1 - \alpha x_2) \quad \text{--- (19)}$$

ทำการจัดรูปใหม่เป็น

$$\begin{aligned} \dot{z} &= (-\alpha x_1 - j\alpha x_2) + (\beta x_2 - j\beta x_1) \\ &= -\alpha(x_1 + jx_2) - j\beta(x_1 + jx_2) \\ &= (-\alpha - j\beta)z \end{aligned}$$

พิจารณาในพิกัดขั้วโคออร์ดิเนตโดยกำหนดให้

$$z = re^{j\theta}$$

แทนค่าลงใน (20) จะได้เป็น

$$re^{j\theta} + jr\dot{\theta}e^{j\theta} = (-\alpha - j\beta)re^{j\theta} = -\alpha re^{j\theta} - j\beta re^{j\theta}$$

เทียบสัมประสิทธิ์จะได้สมการเป็น

$$\begin{aligned} \dot{r} &= -\alpha r \longrightarrow \frac{dr}{dt} = -\alpha r \\ r\dot{\theta} &= -\beta r \longrightarrow \frac{d\theta}{dt} = -\beta \end{aligned} \quad \text{---- (20)}$$

จากระบบสมการ (20) เราจะได้เป็น

$$\frac{dr}{d\theta} = (\alpha/\beta)r$$

แก้สมการดิฟเฟอเรนเชียลจะได้สมการแสดงความสัมพันธ์ในพิกัดโพลาร์โคออร์ดิเนตเป็น

$$r = r_0 e^{(\theta - \theta_0)\alpha/\beta} \quad \text{---- (21)}$$

เราแยกการพิจารณาเป็นกรณีได้ดังต่อไปนี้

กรณีที่ 1 ค่าส่วนจริงของ eigenvalue เป็นศูนย์

- พิจารณาจากสมการ(21) จะเห็นได้ว่าลักษณะของ phase-portrait จะเป็นวงกลม โดยมีรัศมีต่าง ๆ กันขึ้นกับเงื่อนไขเริ่มต้น ดังสมการ

$$r = r_0$$

- ค่าส่วนจริงของ eigenvalue มีค่าเป็นบวก เราสังเกตจากสมการ (20) จะเห็นได้ว่าขนาดของโลกลัสจะเพิ่มขึ้นด้วยอัตราแบบเอกซ์โปเนนเชียล ส่วนขนาดของมุมจะเพิ่มขึ้นเรื่อย ๆ นั่นคือโลกลัสที่ได้จะมีลักษณะเป็น spiral ที่มีทิศพุ่งออกจากจุด singular point

จุด singular point กรณีนี้เราเรียกว่า spiral (unstable focus)

- ค่าส่วนจริงของ eigenvalue มีค่าเป็นลบ เราสังเกตจากสมการ (20) จะเห็นได้ว่าขนาดของโลกลัสจะลดลงด้วยอัตราแบบเอกซ์โปเนนเชียล ส่วนขนาดของมุมจะเพิ่มขึ้นเรื่อย ๆ นั่นคือโลกลัสที่ได้จะมีลักษณะเป็น spiral ที่มีทิศพุ่งเข้าหาจุด singular point

จุด singular point กรณีนี้เราเรียกว่า spiral (stable focus)

1.5 การพิจารณา phase-portrait สำหรับระบบโดยทั่วไป

เราสามารถพิจารณาลักษณะของ phase-portrait อย่างคร่าว ๆ โดยการหาจุด singular point ของระบบก่อน จากนั้นทำการ linearize ณ จุด singular point โดยการกระจายอนุกรมเทย์เลอร์ เมื่อได้สมการที่ linearize แล้วเราสามารถนำหลักการที่ได้กล่าวในหัวข้อที่ผ่านมาทำการวิเคราะห์ลักษณะของ phase-portrait ณ จุด singular point ได้ โดยจุด singular point กรณีนี้จะสมมูลกับจุด

ออริจินของระบบเดิมนั่นเอง

หมายเหตุ การ linearized โดยใช้การกระจายของอนุกรมของเทย์เลอร์รอบจุด (x_0, y_0) สามารถคำนวณจากสมการ

$$f(x, y) = c_1 + c_2(x - x_0) + c_3(y - y_0) \quad \text{----- (22)}$$

$$\begin{aligned} \text{โดย } c_1 &= f(x, y) \quad x = x_0, y = y_0 \\ c_2 &= f_x(x, y) \quad x = x_0, y = y_0 \\ c_3 &= f_y(x, y) \quad x = x_0, y = y_0 \end{aligned}$$

1.6 การพิจารณาค่าเวลาบน phase-portrait

ในที่นี้จะพิจารณาค่าเวลาบน phase-portrait สองวิธีคือ

1) การคำนวณจาก $\Delta t = \Delta x / \dot{x}$

เมื่อพิจารณาในช่วงระยะเวลาสั้น ๆ เราจะได้สมการเป็น

$$\Delta x = \dot{x} \Delta t \quad \text{----- (23)}$$

นั่นคือกรณีที่เรากำลังหาช่วงเวลาระหว่างจุดสองจุดบน phase-portrait เราสามารถทำได้โดยการแบ่งช่วงที่ต้องการพิจารณาออกเป็นช่วงสั้น ๆ จากนั้นคำนวณหาช่วง ที่สอดคล้องกับแต่ละช่วง ผลรวมของ ในแต่ละช่วงที่ได้จะเป็นช่วงเวลาทั้งหมดที่เราต้องการนั่นเอง

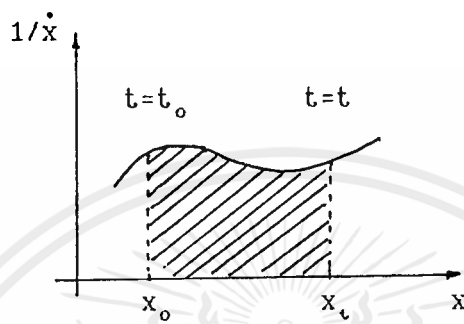
2) การคำนวณจาก $t = \int (1/\dot{x}) dx$

พิจารณาจากสมการ $\dot{x} = dx/dt$

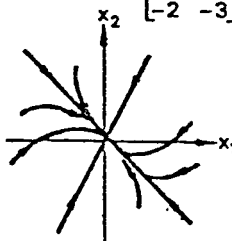
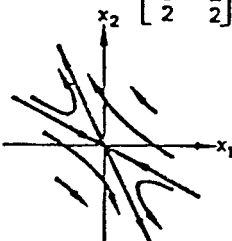
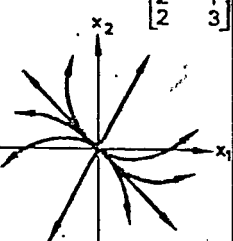
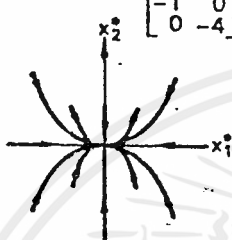
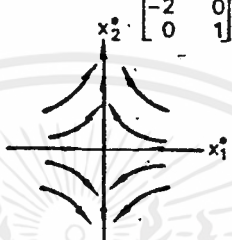
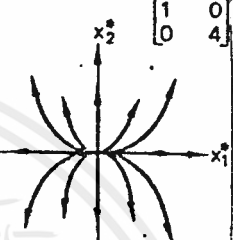
ทำการอินทิเกรตจาก $t=t_0$ ถึง $t=t$ เราจะได้ความสัมพันธ์เป็น

$$t - t_0 = \int_{x(t_0)}^{x(t)} (1/\dot{x}) dx$$

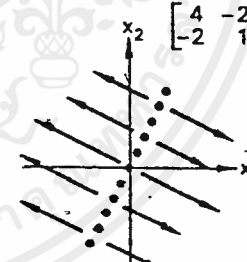
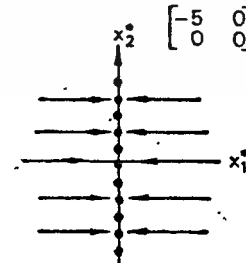
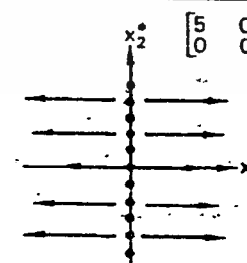
เราสามารถคำนวณค่าช่วงเวลาระหว่างจุดสองจุดบน phase-portrait จากสมการ (24) โดยการหาส่วนกลับของกราฟในช่วงที่เราต้องการจะทราบ (หากกราฟ $1/\dot{x}$ โดยเขียนบน แกน x) จากนั้นคำนวณหาพื้นที่ใต้กราฟของกราฟส่วนกลับนั้น พื้นที่ใต้กราฟที่ได้คือค่าของเวลาที่ต้องการนั่นเอง



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (14) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

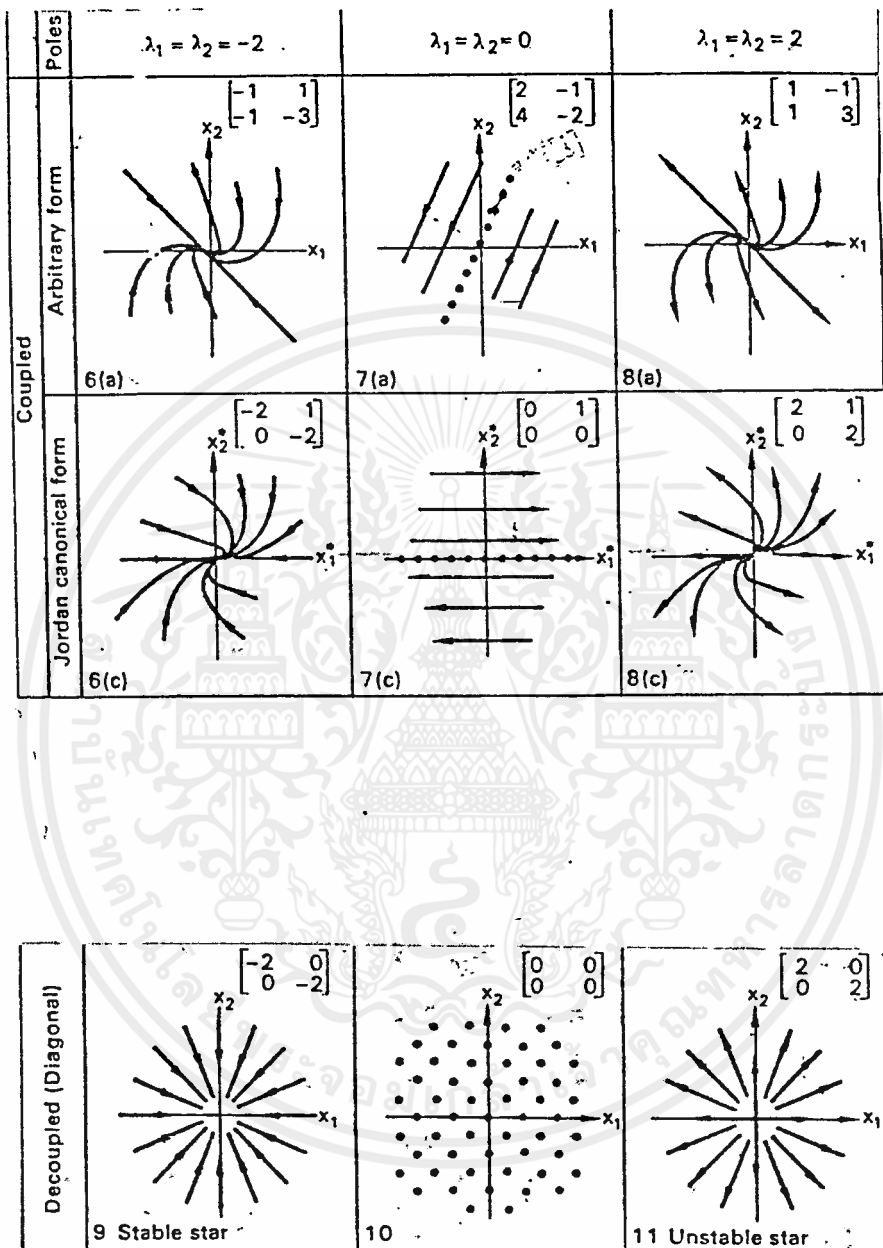
Poles			
	$\lambda_1 = -1, \lambda_2 = -4$	$\lambda_1 = -2, \lambda_2 = 1$	$\lambda_1 = 1, \lambda_2 = 4$
Arbitrary form	$\begin{bmatrix} -2 & -1 \\ -2 & -3 \end{bmatrix}$ 	$\begin{bmatrix} -3 & -2 \\ 2 & 2 \end{bmatrix}$ 	$\begin{bmatrix} 2 & 1 \\ 2 & 3 \end{bmatrix}$ 
	1(a) Stable node	2(a) Saddle	3(a) Unstable node
Both poles not zero	$\begin{bmatrix} -1 & 0 \\ 0 & -4 \end{bmatrix}$ 	$\begin{bmatrix} -2 & 0 \\ 0 & 1 \end{bmatrix}$ 	$\begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$ 
	1(c)	2(c)	3(c)

รูปที่ 3 แสดงลักษณะ phase-portrait กรณีรากเป็นจำนวนจริงค่าต่างกัน

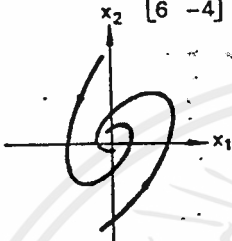
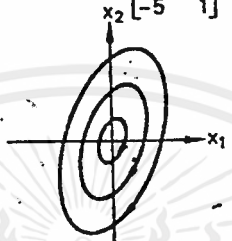
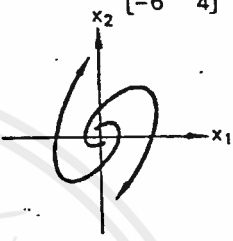
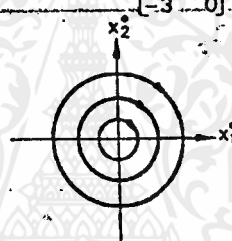
Poles			
	$\lambda_1 = -5, \lambda_2 = 0$		$\lambda_1 = 5, \lambda_2 = 0$
Arbitrary form	$\begin{bmatrix} -4 & 2 \\ 2 & -1 \end{bmatrix}$ 		$\begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix}$ 
	4(a)		5(a)
One pole zero	$\begin{bmatrix} -5 & 0 \\ 0 & 0 \end{bmatrix}$ 		$\begin{bmatrix} 5 & 0 \\ 0 & 0 \end{bmatrix}$ 
	4(c)		5(c)

แสดงลักษณะ phase-portrait กรณีรากเป็นจำนวนจริงค่าต่างกัน

(มีรากค่าหนึ่งเป็นศูนย์)



รูปที่ 4 แสดงลักษณะ phase-portrait กรณีรากเป็นจำนวนจริงค่าเท่ากัน

Poles	$\lambda_1, \lambda_2 = -1 \pm 3j$	$\lambda_1, \lambda_2 = \pm 3j$	$\lambda_1, \lambda_2 = 1 \pm 3j$
Arbitrary form	$\begin{bmatrix} 2 & -3 \\ 6 & -4 \end{bmatrix}$  12(a) Stable focus	$\begin{bmatrix} -1 & 2 \\ -5 & 1 \end{bmatrix}$  13(a) Centre	$\begin{bmatrix} -2 & 3 \\ -6 & 4 \end{bmatrix}$  14(a) Unstable focus
Modified canonical form	$\begin{bmatrix} -1 & 3 \\ -3 & -1 \end{bmatrix}$  12(c)	$\begin{bmatrix} 0 & 3 \\ -3 & 0 \end{bmatrix}$  13(c)	$\begin{bmatrix} 1 & -3 \\ 3 & 1 \end{bmatrix}$  14(c)

รูปที่ 5 แสดงลักษณะ phase-portrait กรณีรากเป็นจำนวนเชิงซ้อน

บทที่ 2

วิธีการวิเคราะห์แบบเดสไคร์บิงฟังก์ชัน (Describing Function Method)

2.1 บทนำ

ในการวิเคราะห์ระบบที่เป็น NON LINEAR จะมีวิธีการในการวิเคราะห์หลายแบบแตกต่างกันไปขึ้นกับลักษณะของระบบ วิธีการวิเคราะห์แบบ phase-plane เป็นวิธีที่เหมาะสมสำหรับระบบที่เป็นอันดับสอง สำหรับระบบที่มีอันดับมากกว่าสอง เราจำเป็นต้องหาวิธีการวิเคราะห์แบบอื่นส่วนวิธีการวิเคราะห์แบบเดสไคร์บิงฟังก์ชัน (describing function method) จะใช้ในการวิเคราะห์ระบบที่เกี่ยวข้องกับ nonlinear element ซึ่งเกิดขึ้นจากตัวพลานท์ คอนโทรลเลอร์ หรือตัวเซนส์-เซอร์ ในการทำการวิเคราะห์แบบนี้จะกระทำภายใต้สมมติฐานต่าง ๆ คือ

- 1) อินพุตของ non linear element เป็น sinusoidal
- 2) เอาท์พุตของ nonlinear element เป็นสัญญาณที่เป็นคาบและมี fundamental frequency เช่นเดียวกับสัญญาณอินพุต การวิเคราะห์พฤติกรรมของ nonlinear element จะกระทำในรูป frequency response

- 3) ส่วนสำคัญที่เราใช้ในการพิจารณาเอาท์พุตคือส่วนที่เป็น fundamental

ภายใต้เงื่อนไขเหล่านี้เราจะได้ระบบที่เปรียบเสมือนแบบ linear ทำให้เราสามารถทำการวิเคราะห์โดยใช้หลักเดียวกันกับการวิเคราะห์ระบบที่เป็น linear นั่นเอง

ข้อควรสังเกต : หากระบบของเรามีคุณสมบัติเป็น low pass filter มากเพียงพอ การวิเคราะห์แบบเดสไคร์บิงฟังก์ชันจะให้ผลลัพธ์ที่ถูกต้องมากขึ้นเท่านั้น เนื่องจากการวิเคราะห์แบบนี้เราสามารถใช้ได้ดีกับสัญญาณที่มีสเปกตรัมสูง ๆ ที่ความถี่หนึ่งและช่วงของแบนด์วิธสัญญาณแคบ ๆ

2.2 นิยามของเดสไคร์บิงฟังก์ชัน

จากการวิเคราะห์แบบคอนเวนชัน เราทราบว่าทรานส์เฟอ์ฟังก์ชันของระบบที่เป็นลิเนียร์จะเป็นอัตราส่วนของเอาท์พุตต่ออินพุตเมื่อสัญญาณอินพุตเป็นไซน์ กำหนดให้

$$\text{สัญญาณอินพุต : } x = X \sin \omega t \quad \text{---- (1)}$$

$$\text{สัญญาณเอาท์พุต : } y = Y \sin(\omega t + \Phi) \quad \text{---- (2)}$$

เราจะได้ว่าอัตราส่วนของ Y/X และมุมเฟส จะเป็นฟังก์ชันกับความถี่ของส่วนอินพุตเท่านั้น (ไม่ขึ้นกับขนาดของอินพุต)

เมื่อเราพิจารณาระบบที่เป็น NON LINEAR จากเงื่อนไขเอาท์พุตของ nonlinear element เป็นสัญญาณที่เป็นคาบและมี fundamental frequency เช่นเดียวกับสัญญาณอินพุต เราจะได้สมการ

เป็น

$$y(t+T) = y(t) \quad \text{โดย} \quad T = 2\pi/\omega.$$

เราสามารถหาส่วนที่เป็น fundamental ของเอาต์พุตโดยการกระจาย fourier series ซึ่งเป็นไป

ตามสมการ

$$y(t) = A_0 + \sum_{n=1}^{\infty} (A_n \cos n\omega t + B_n \sin n\omega t) \quad \text{-----}(3)$$

$$= A_0 + \sum_{n=1}^{\infty} Y_n \sin(n\omega t + \Phi_n)$$

$$\text{โดยที่} \quad A_n = \frac{1}{\pi} \int_0^{2\pi} y(t) \cos n\omega t d(\omega t).$$

$$B_n = \frac{1}{\pi} \int_0^{2\pi} y(t) \sin n\omega t d(\omega t)$$

$$Y_n = \sqrt{A_n^2 + B_n^2}$$

$$\Phi_n = \arctan(A_n/B_n)$$

จากเงื่อนไขส่วนสำคัญที่เราใช้ในการพิจารณาเอาต์พุตคือส่วนที่เป็น fundamental เราจะได้สมการเป็น

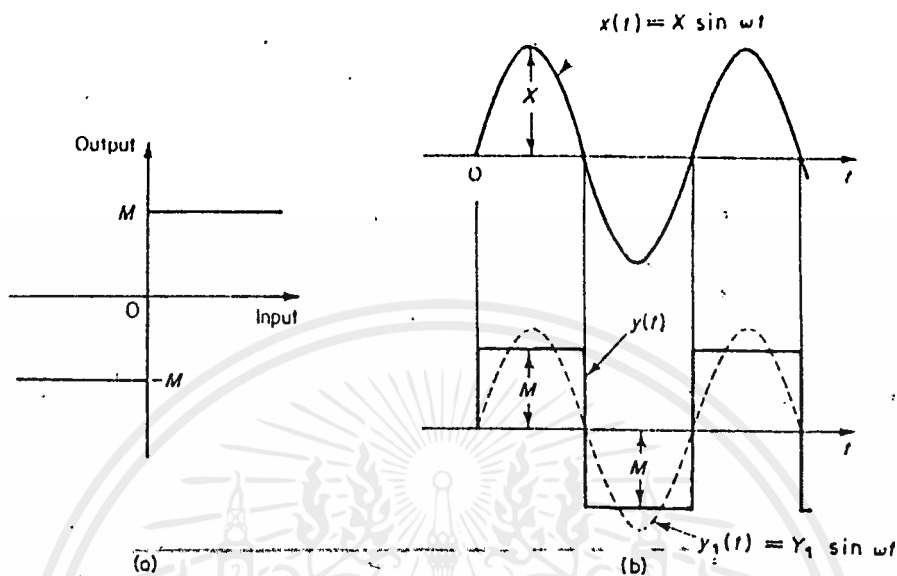
$$\begin{aligned} y_1(t) &= A_1 \cos \omega t + B_1 \sin \omega t \\ &= Y_1 \sin(\omega t + \Phi_1) \end{aligned} \quad \text{-----} (5)$$

เราจะได้เดสไคร์บิงฟังก์ชันเป็นไปตามสมการ

$$\begin{aligned} N &= (Y_1/x) \angle \Phi \\ &= \frac{\sqrt{A_1^2 + B_1^2}}{x} \angle \arctan(A_1/B_1) \end{aligned}$$

2.3 เดสไคร์บิงฟังก์ชันสำหรับ NON LINEAR ที่น่าสนใจ

สมมติให้ non linear element มีลักษณะเป็น on-off ดังแสดงในรูป



รูปที่ 1 ลักษณะของ on-off

กรณีนี้ $y(t)$ เป็นฟังก์ชันคี่ ฉะนั้น A_n ($n=0,1,2,\dots$) มีค่าเป็นศูนย์ เราจะได้ว่า

$$y_1(t) = B_1 \sin \omega t \quad \text{--- (6)}$$

โดยที่

$$\begin{aligned} y_1(t) &= \frac{1}{\pi} \int_0^{2\pi} y(t) \sin \omega t d(\omega t) \\ &= \frac{2}{\pi} \int_0^{2\pi} y(t) \sin \omega t d(\omega t) \end{aligned}$$

$$= 4m/\pi$$

แทนค่าลงใน (6) จะได้เป็น

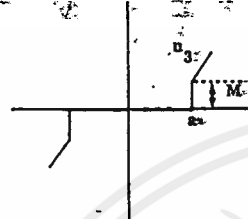
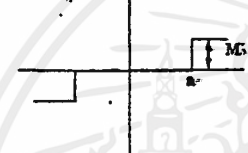
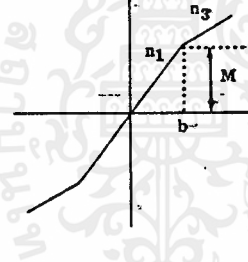
$$y_1(t) = (4m/\pi) \sin(\omega t)$$

ดังนั้นเดสไครบิงฟังก์ชันสำหรับกรณีนี้คือ

$$= (4m/\pi)x$$

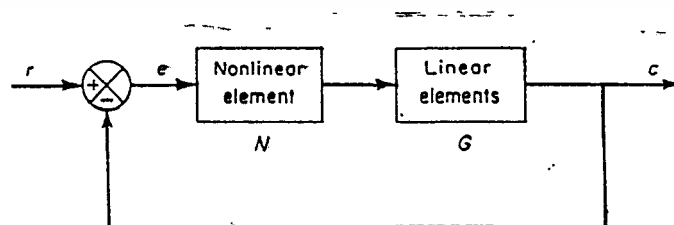
กรณี non linear element อื่น ๆ เราสามารถทำการคำนวณได้ในทำนองเดียวกัน สำหรับ non linear element อื่น ๆ ที่เราพบบ่อย ๆ ในการใช้งานได้สรุปเดสไครบิ๊งฟังก์ชันสำหรับแต่ละ element ดังตารางที่ (1)

ตารางที่ 1 แสดงเดสไครบิ๊งฟังก์ชันของ nonlinear element ต่าง ๆ ที่น่าสนใจ

	$g(R_o) = \frac{n_3}{\pi R_o} \left[R_o (\pi - 2\theta_1) + R_o \sin 2\theta_1 - 4k_3 \cos \theta_1 \right]$ $b(R_o) = 0$
	$g(R_o) = \frac{4M}{\pi R_o} \cos \theta_1$ $b(R_o) = -0$
	$g(R_o) = \frac{n_1}{\pi} (2\theta_2 - \sin 2\theta_2) + \frac{n_3}{\pi R_o} \left[R_o (\pi - 2\theta_2) + R_o \sin 2\theta_2 - 4k_3 \cos \theta_2 \right]$ $b(R_o) = 0$

2.4 การประยุกต์ใช้เดสไครบิ๊งฟังก์ชันในการวิเคราะห์เสถียรภาพ

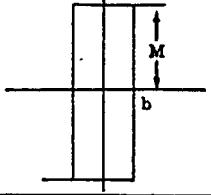
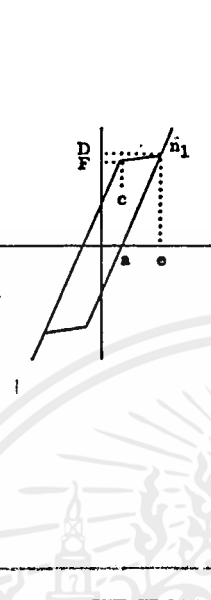
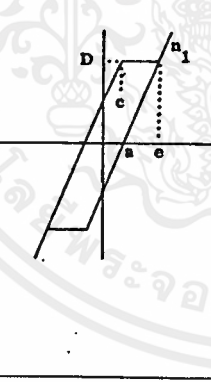
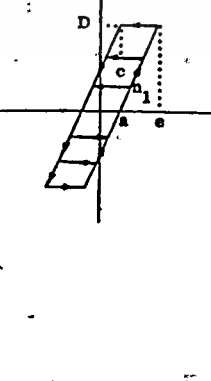
พิจารณาระบบป้อนกลับแบบวงปิดดังรูปที่ 1



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (21) ขาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	$g(R_o) = \frac{n_1}{\pi} (2\theta_2 - \sin 2\theta_2) + \frac{4M}{\pi R_o} \cos \theta_2$ $b(R_o) = 0$
	$g(R_o) = \frac{n_3}{\pi R_o} \left[R_o (\pi - 2\theta_2) + R_o \sin 2\theta_2 - 4b \cos \theta_2 \right]$ $b(R_o) = 0$
	$g(R_o) = n_3 + \frac{4M}{\pi R_o}$ $b(R_o) = 0$
	$g(R_o) = \frac{4M}{\pi R_o}$ $b(R_o) = 0$

	$g(R_o) = \frac{n_1}{\pi R_o} \left[2R_o (-\theta_1 + \theta_2) + R_o (\sin 2\theta_1 - \sin 2\theta_2) + 4a (\cos \theta_2 - \cos \theta_1) \right]$ $+ \frac{n_3}{\pi R_o} \left[R_o (\pi - 2\theta_2) + R_o \sin 2\theta_2 - 4k_3 \cos \theta_2 \right]$ $b(R_o) = 0$
	$g(R_o) = \frac{n_1}{\pi R_o} \left[2R_o (-\theta_1 + \theta_2) + R_o (\sin 2\theta_1 - \sin 2\theta_2) + 4a (\cos \theta_2 - \cos \theta_1) \right]$ $+ \frac{4M}{\pi R_o} \cos \theta_2$ $b(R_o) = 0$

	$g(R_o) = \frac{4M}{\pi R_o} \cos \theta_2$ $b(R_o) = \frac{-4M}{\pi R_o} \sin \theta_2$
	$g(R_o) = \frac{n_1}{\pi R_o} \left[R_o (\pi + \theta_2 - \theta_5) + \frac{R_o}{2} (-\sin 2 \theta_3 + \sin 2 \theta_5) - 2a (\cos \theta_3 + \cos \theta_5) \right]$ $+ \frac{n_2}{\pi R_o} \left[R_o (\theta_5 - \theta_3) + \frac{R_o}{2} (\sin 2 \theta_3 - \sin 2 \theta_5) + 2k_2 (\cos \theta_5 - \cos \theta_3) \right]$ $b(R_o) = \frac{n_1}{\pi R_o} \left[\frac{R_o}{2} (\cos 2 \theta_3 - \cos 2 \theta_5) - 2a (\sin \theta_3 + \sin \theta_5) \right]$ $+ \frac{n_2}{\pi R_o} \left[\frac{R_o}{2} (-\cos 2 \theta_3 + \cos 2 \theta_5) + 2k_2 (\sin \theta_5 - \sin \theta_3) \right]$
	$g(R_o) = \frac{n_1}{\pi R_o} \left[R_o (\pi + \theta_3 - \theta_5) + \frac{R_o}{2} (-\sin 2 \theta_3 + \sin 2 \theta_5) - 2a (\cos \theta_3 + \cos \theta_5) \right]$ $+ \frac{2D}{\pi R_o} (\cos \theta_3 - \cos \theta_5)$ $b(R_o) = \frac{n_1}{\pi R_o} \left[\frac{R_o}{2} (\cos 2 \theta_3 - \cos 2 \theta_5) - 2a (\sin \theta_3 + \sin \theta_5) \right]$ $+ \frac{2D}{\pi R_o} (\sin \theta_3 - \sin \theta_5)$
	$g(R_o) = \frac{2n_1}{\pi} \left(\frac{\pi}{4} + \frac{\theta_3}{2} + \frac{\sin 2 \theta_3}{4} \right)$ $b(R_o) = \frac{-2n_1}{\pi} \cos^2 \theta_3$

จากรูปจะได้ความสัมพันธ์เป็น

$$\frac{C(j\omega)}{R(j\omega)} = \frac{NG(j\omega)}{1+NG(j\omega)} \quad \text{----- (7)}$$

และจะได้สมการคุณลักษณะเป็น

$$1 + NG(j\omega) = 0 \quad \text{----- (8)}$$

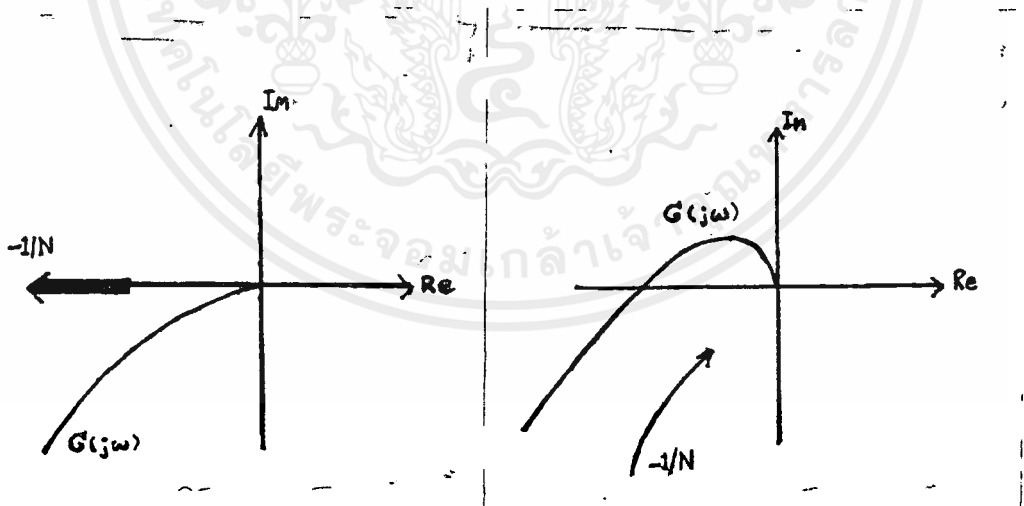
$$G(j\omega) = -1/N$$

หากระบบสอดคล้องกับสมการที่ (7) เราจะได้ว่าผลตอบสนองของเอาต์พุตจะมีลักษณะเป็น limit cycle (เมื่อพิจารณาเปรียบเทียบกับกรณี linear ทางเดินของ $-1/N$ จะสอดคล้องกับจุด $-1+j0$ ในกรณีการวิเคราะห์แบบ Nyquist นั่นเอง)

ในการวิเคราะห์เสถียรภาพ กรณีนี้สมมติให้ระบบเป็น minimum phase system (ระบบที่มีโพลของวงเปิดอยู่ด้านซ้ายของระนาบ(s) เราทำการพล็อตโลโก้ของ $-1/N$ และของ $G(j\omega)$ จากทฤษฎีของ Nuquist จะได้ว่า

กรณีที่ 1 หากโลโก้ของ $-1/N$ ไม่ถูก enclosed ด้วยโลโก้ของ $G(j)$ จะสรุปได้ว่าระบบเสถียรภาพดังรูปที่ (2)

กรณีที่ 2 หากโลโก้ของ $-1/N$ ถูก enclosed ด้วยโลโก้ของ $G(j)$ ทั้งหมด จะสรุปได้ว่าระบบเสถียรภาพ ดังรูปที่ (3)

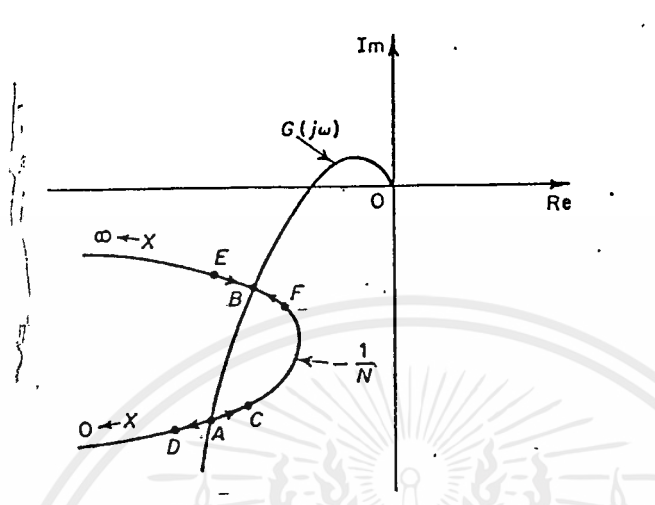


รูปที่ 2

รูปที่ 3

กรณีที่ 3 หากโลโก้ของ $-1/N$ และ $G(j\omega)$ มีการตัดกัน ผลตอบสนองของระบบจะมีพฤติกรรม

ที่เรียกว่า sustained oscillator การวิเคราะห์เสถียรภาพของ limit cycle ทำได้ โดยการพิจารณาจากรูป (4) เราจะวิเคราะห์พฤติกรรมของระบบที่จุด A และ B



รูปที่ (4)

1) สมมติให้ระบบมีจุดเริ่มต้นที่จุด A เมื่อมีสัญญาณรบกวนจากภายนอกทำให้ขนาดของอินพุตที่มีต่อ nonlinear element เปลี่ยนแปลงไป สามารถพิจารณาเป็น 2 กรณีคือ

1.1 มีผลให้ขนาดของอินพุตสูงขึ้น จะทำให้จุดทำงานเลื่อนจากจุด A ไปจุด C กรณีนี้ทำให้โลกซ์ของ $G(j)$ ล้อมรอบจุดทำงาน นั่นคือระบบจะไม่เสถียรภาพ ทำให้ขนาดของเอาต์พุตเพิ่มขึ้น เอาต์พุตนี้จะถูกป้อนกลับเป็นอินพุตของ nonlinear element เช่นนี้ไปเรื่อย ๆ ทำให้จุดทำงานของระบบเคลื่อนที่ไปอยู่จุด B

1.2 มีผลให้ขนาดของอินพุตน้อยลง จะทำให้จุดทำงานเลื่อนจากจุด A ไปจุด D กรณีนี้ทำให้โลกซ์ของ $G(j)$ ไม่ล้อมรอบจุดทำงาน นั่นคือระบบจะเสถียรภาพ ทำให้ขนาดของเอาต์พุตลดลง เอาต์พุตนี้จะถูกป้อนกลับเป็นอินพุตของ nonlinear element เช่นนี้ไปเรื่อย ๆ ทำให้จุดทำงานของระบบเคลื่อนที่ล่อออกไปจากจุด A

จะเห็นได้ว่าพฤติกรรมของระบบที่จุด A จะมีลักษณะล่อออกจากจุด (divergent) เราจะเรียกเรียกพฤติกรรมนี้ว่า unstable limit circle

2) สมมติให้ระบบมีจุดเริ่มต้นที่จุด B เมื่อมีสัญญาณรบกวนจากภายนอกทำให้ขนาดของอินพุตที่มีต่อ nonlinear element เปลี่ยนแปลงไป สามารถพิจารณาเป็น 2 กรณีคือ

1.1 มีผลให้ขนาดของอินพุตสูงขึ้น จะทำให้จุดทำงานเลื่อนจากจุด B ไปจุด E กรณีนี้ทำให้โลกัสของ $G(j)$ ไม่ล้อมรอบจุดทำงาน นั่นคือระบบจะเสถียรภาพ ทำให้ขนาดของเอาต์พุตลดลง เอาต์พุตนี้จะถูกป้อนกลับไปเป็นอินพุตของ nonlinear element เช่นนี้ไปเรื่อย ๆ ทำให้จุดทำงานของระบบเคลื่อนที่ไปสู่เข้าหาจุด B

1.2 มีผลให้ขนาดของอินพุตน้อยลง จะทำให้จุดทำงานเลื่อนจากจุด B ไปจุด E กรณีนี้ทำให้โลกัสของ $G(j)$ ล้อมรอบจุดทำงาน นั่นคือระบบจะไม่เสถียรภาพ ทำให้ขนาดของเอาต์พุตเพิ่มขึ้น เอาต์พุตนี้จะถูกป้อนกลับไปเป็นอินพุตของ nonlinear element เช่นนี้ไปเรื่อย ๆ ทำให้จุดทำงานของระบบเคลื่อนที่สู่เข้าหา A จะเห็นได้ว่าพฤติกรรมของระบบที่จุด B จะมีลักษณะลู่เข้าหาจุด (convergent) เรา จะเรียกพฤติกรรมนี้ว่า stable limit circle



บทที่ 3

การวิเคราะห์เสถียรภาพโดยใช้ทฤษฎีลียาปูนอฟ (Lyapunov Theory)

3.1 บทนำ

สำหรับระบบต่อเนื่องเชิงเส้นไม่แปรตามเวลา ถ้าเราเปลี่ยนสมการไดนามิกส์เป็นทราנסเฟอรัฟังกชันได้เราก็สามารถตรวจสอบเสถียรภาพโดยใช้วิธีต่างๆ เช่น วิธีของราท์(Routh criterion), วิธีตรวจสอบจากเส้นทางเดินของราก(Root locus), วิธีพล็อตไนควิสต์ (Nyquist plot) และวิธีพล็อตโบด(Bode plot) เมื่อระบบไม่เป็นเชิงเส้น (nonlinear) และขึ้นกับเวลา (Time-variant) วิธีหาเสถียรภาพดังกล่าวไม่สามารถนำมาใช้ได้ โดยเฉพาะอย่างยิ่งระบบที่ขึ้นกับเวลา ซึ่งถ้าระบบมีเสถียรภาพที่เวลาหนึ่งไม่ได้หมายความว่าระบบจะมีเสถียรภาพที่เวลาอื่น ๆ สำหรับวิธีที่จะใช้ศึกษาเสถียรภาพของระบบที่ไม่ใช่เชิงเส้นมีอยู่หลายวิธี วิธีหนึ่งได้แก่วิธีของลียาปูนอฟ ซึ่งประกอบด้วย วิธีวิเคราะห์เสถียรภาพ 2 วิธี คือ วิธีการประมาณเชิงเส้น (Linearization method) และ วิธีการโดยตรง (Direct method) ก่อนที่เราจะกล่าวถึง 2 วิธีนี้ ควรจะมาทำความเข้าใจถึง นิยามของระบบบางอย่างก่อน

3.1.1 ระบบออโตโนมัส (Autonomous) และ ระบบนอนออโตโนมัส (Non-autonomous System)

นิยามที่ 1 สำหรับระบบที่ไม่เป็นเชิงเส้น $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ เรียกว่าเป็นระบบ autonomous ถ้า t ไม่ขึ้นกับเวลาโดยสิ้นเชิง สามารถเขียนได้ว่า

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$$

นอกนั้นเป็นระบบ ระบบนอนออโตโนมัสทั้งสิ้น ถ้าเทียบกับระบบที่เป็นเชิงเส้นไม่แปรค่ากับเวลาก็คือระบบออโตโนมัส และระบบที่เป็นเชิงเส้นแปรค่ากับเวลาเทียบได้กับระบบนอนออโตโนมัส

3.2 แนวความคิดเกี่ยวกับเสถียรภาพของลียาปูนอฟ

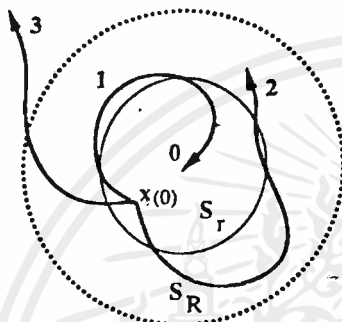
ก่อนอื่นเรากำหนดให้ B_R แทนขอบเขตที่กำหนดโดย $\|\mathbf{x}\| < R$ ในสเปซเวกเตอร์ และ S_R กำหนดโดย $\|\mathbf{x}\| = R$

นิยามที่ 2 ที่สถานะสมดุล (Equilibrium State) $\mathbf{x}=0$ จะเรียกว่ามีเสถียรภาพ(Stable)สำหรับทุกๆ ค่า $R>0$ และ $r>0$ บางตัว ถ้า $\|\mathbf{x}(0)\| < r$ แล้ว $\|\mathbf{x}(t)\| < R$ ทุกๆ เวลาที่ $t > 0$ มิฉะนั้นจะเรียกว่า ไม่มีเสถียรภาพ (unstable)

เสถียรภาพนี้อาจเรียกว่า เสถียรภาพในแง่ของ ลียาปูนอฟ (Stability in sense of Lyapunov) หมายความว่า เส้นทางของระบบสามารถรักษาให้เข้าสู่สถานะสมดุล(Equilibrium state) โดยมีค่าเริ่ม

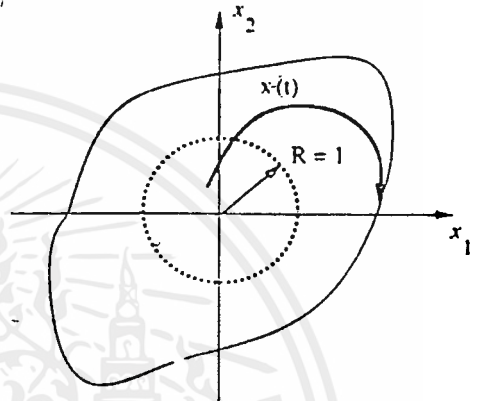
ต้นที่เพียงพอที่จะรู้ได้ว่าระบบที่เสถียร เมื่อระบบเริ่มต้นใน S_r ที่ $t=0$ จะไม่ออกนอก S_R เลย เมื่อเวลา t เพิ่มขึ้น จากรูปที่ 1 ทางเดินของระบบ 2 ไม่ได้ลู่เข้าสู่ออริจินแต่ก็ยังคงเสถียรเพราะยังอยู่ใน S_R เรียกเสถียรภาพแบบนี้ว่าเสถียรแบบมาจิ้น(Marginally Stable) ส่วนทางเดินของระบบ 3 ไม่เสถียรเพราะลู่ออกนอก S_R

แต่การลู่ออกจากออริจินก็ไม่ได้หมายความว่าระบบไม่เสถียร ถ้าเป็นการลู่เข้าสู่ limit cycle กรณีนี้เรียกว่า ไม่เสถียรที่ออริจิน จากรูปที่ 2



รูปที่ 1

แนวคิดเกี่ยวกับเสถียรภาพ



รูปที่ 2

การไม่เสถียรที่ออริจิน

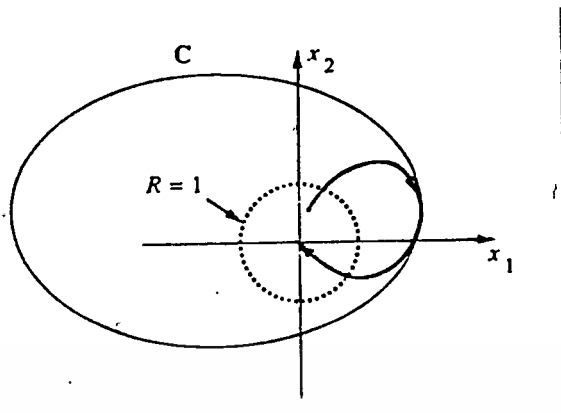
ของแวนเดอร์ โพลอสซิลเลเตอร์ (Van der Pol Oscillator)

3.2.1 เสถียรภาพเชิงอะซิมโทติก และ เชิงเอกซ์โปเนนเชียล (Asymptotically and

Exponentially Stable)

นิยามที่ 3 ที่จุดสมดุล (Equilibrium point) 0 เรียกว่า มีเสถียรภาพเชิงอะซิมโทติก (Asymptotically Stable) ถ้าระบบมีเสถียรภาพ และ ถ้าสำหรับ r บางตัวซึ่ง $r > 0$ โดยที่ $\|x(0)\| < r$ แล้ว $x(t)$ จะมีค่าเข้าใกล้ 0 เมื่อ t เข้าใกล้ ∞

เราจึงเรียก B_r ว่าขอบเขตของการลู่เข้า(Domain of attraction) ของจุดสมดุล เนื่องจากทางเดินของระบบจะลู่เข้าสู่ขอบเขตนี้เสมอ ซึ่งถ้าเป็นเสถียรภาพเชิงอะซิมโทติกจะเสถียรในแง่ของลีโอพูนอฟด้วย และขนาดของ B_r ที่ใหญ่ที่สุดไม่ว่าระบบจะเริ่มต้นจากสเกลใดใน B_r ผลเฉลยจะลู่เข้าสู่จุดสมดุลที่ ออริจินเสมอ จากรูปที่ 1 เส้นทางเดินของระบบหมายเลข 1 เสถียรเชิงอะซิมโทติก แต่การลู่เข้าของสเกลไม่ได้หมายความว่าเสถียร เพราะต้องพิจารณาว่า ทางเดินของระบบเกินขอบเขตที่จะศึกษาหรือไม่ ถ้าเกินขอบเขตถึงแม้ว่าจะลู่เข้าก็ตาม ย่อมถือว่าไม่เสถียร จากรูปข้างล่าง



รูปที่ 3 การลู่เข้าซึ่งไม่ถือว่าเสถียร

อย่างไรก็ตามย่อมไม่เป็นการเพียงพอ ถ้าเราทราบเพียงการลู่เข้าของระบบเพียงอย่างเดียว แต่ควรทราบด้วยว่า ระบบเข้าสู่จุดสมดุลเร็วเท่าไร

นิยามที่ 4 ที่จุดสมดุล 0 เรียกว่าเสถียรเชิงเอกซ์โปเนนเชียล(Exponentially stable)ถ้ามีจำนวนที่เป็นบวก 2 ตัว คือ α และ γ โดย

$$\text{ทุก } \gamma \text{ ค่า } t > 0 \quad \|x(t)\| \leq \alpha \|x(0)\| e^{-\gamma t}$$

ใน B_r รอบ ๆ จุด origin

หมายความว่าสเตทเวกเตอร์ของระบบที่เสถียรเชิงเอกซ์โปเนนเชียล จะลู่เข้าสู่อริจิน เร็วกว่าฟังก์ชันเอกซ์โปเนนเชียล ความเร็วในการลู่เข้าจึงขึ้นอยู่กับ γ เรียกว่าอัตราการลู่เข้าแบบเอกซ์โปเนนเชียล ดังนั้นการที่ระบบเสถียรเชิงเอกซ์โปเนนเชียล จึงเสถียรเชิงอะซิมโทติกด้วย แต่ระบบที่เสถียรเชิงอะซิมโทติกอาจไม่ใช่เสถียรเชิงเอกซ์โปเนนเชียลเสมอไป

3.2.2 วิธีการประมาณเชิงเส้น(Linearization method)

วิธีการประมาณเชิงเส้นของลีอาฟูนอฟ จะเกี่ยวข้องกับเสถียรภาพในขอบเขตแคบ (Local stability) ของระบบไม่ใช่เชิงเส้น จะพิจารณาพฤติกรรมของระบบเพียงขอบเขตแคบๆ เท่านั้นซึ่งระบบมีคุณลักษณะใกล้เคียงกับความเป็นเชิงเส้น จึงใช้วิธีการประมาณให้เป็นเชิงเส้นแล้วใช้วิธีการหาเสถียรภาพตามแบบของระบบเชิงเส้น ลองพิจารณาระบบต่อเนื่องไม่แปรค่ากับเวลาที่มีสมการไดนามิกส์ดังนี้

$$\dot{\mathbf{x}} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)_{\mathbf{x}=0} \mathbf{x} + \mathbf{f}_{h.o.t}(\mathbf{x})$$

$\mathbf{f}_{h.o.t}$ เป็นส่วนของเลขยกกำลังอันดับสูงในเทอมของ $\mathbf{x}$

ถ้าให้ A เป็นเมตริกซ์ค่าคงที่แทนจาโคเบียนเมตริกซ์(Jacobian matrix) ของ $\mathbf{f}$

$$A = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)_{\mathbf{x}=0}$$

ดังนั้นสมการของระบบจะเป็น $\dot{\mathbf{x}} = A\mathbf{x}$

เป็นการทำการประมาณเชิงเส้นของระบบไม่เชิงเส้นที่จุดสมดุล 0 โดยตัดเทอมที่มีอันดับสูงกว่า 1 ออก

สรุปเป็นเงื่อนไขเสถียรภาพได้ว่า

1. ถ้าการประมาณเป็นเชิงเส้นทำให้เกิดเสถียรภาพ (ค่ารากของสมการอยู่ทางซ้ายมือของ s-plane) ดังนั้นที่จุดสมดุลจะเรียกว่าเสถียรเชิงอะซิมโทติก

2. ถ้าการประมาณเชิงเส้นทำให้ไม่เสถียร (ค่ารากของสมการอยู่ทางขวามือของ s-plane) ดังนั้นจุดสมดุลจะไม่เสถียรด้วย

3. ถ้าระบบที่ถูกประมาณเป็นเชิงเส้นเป็นการเสถียรแบบมาจินคือมีค่ารากอย่างน้อยหนึ่งค่าอยู่บนแกน $j\omega$ ทำให้ไม่สามารถบอกได้ว่ามีเสถียรภาพหรือไม่

จากข้อที่ 3 ต้องเอาเทอมที่มีอันดับสูงๆ มาคิดด้วย จึงจะสามารถตัดสินว่าระบบเสถียรหรือไม่เสถียร วิธีที่จะกล่าวต่อไปเป็นอีกวิธีหนึ่งของลีอาปูนอฟซึ่งมีข้อดีกว่าคือไม่ต้องหาผลเฉลยของสมการเหมือนวิธีการประมาณเชิงเส้น แต่ใช้แนวความคิดของพลังงานแทน

3.2.3 วิธีการโดยตรง(Direct method)

เป็นวิธีที่ใช้แนวความคิดเกี่ยวกับการเปลี่ยนแปลงพลังงานของระบบทางฟิสิกส์ ในการ

พิจารณาเสถียรภาพเช่น สมมติมีระบบอันดับสอง

$$\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}})$$

โดยที่ $\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) = -\mathbf{h}(\mathbf{x}, \dot{\mathbf{x}}) - \mathbf{g}(\mathbf{x})$

จะได้ว่า $\ddot{\mathbf{x}} + \mathbf{h}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{g}(\mathbf{x}) = 0$

เทียบได้กับสมการของระบบทางฟิสิกส์ซึ่งประกอบด้วย $\mathbf{g}(\mathbf{x})$ แทนส่วนที่เก็บพลังงาน และ

$\mathbf{h}(\mathbf{x}, \dot{\mathbf{x}})$ แทนส่วนที่ใช้พลังงาน

ถ้าให้พลังงานจลน์ $T = \frac{1}{2} \dot{\mathbf{x}}^2$ และเรากำหนดพลังงานศักย์โดยให้

$$V(\mathbf{x}) = \int \mathbf{g}(\mathbf{x}) d\mathbf{x} \quad \text{และ} \quad \mathbf{g}(\mathbf{x}) = \frac{d}{dx} V(\mathbf{x})$$

พลังงานรวมของระบบ E เป็นผลรวมของพลังงานจลน์กับพลังงานศักย์

$$E = T + V = \frac{1}{2}\dot{\mathbf{x}}^2 + \int \mathbf{g}(\mathbf{x})d\mathbf{x}$$

พลังงานของระบบจะมีการเปลี่ยนแปลงตามทางเดินเฟส (phase path)

$$\begin{aligned}\frac{dE}{dt} &= \frac{dT}{dt} + \frac{dV}{dt} = \dot{\mathbf{x}}\ddot{\mathbf{x}} + \frac{d}{dt}V(\mathbf{x})\dot{\mathbf{x}} \\ &= \dot{\mathbf{x}}\left[\ddot{\mathbf{x}} + \frac{d}{dt}V(\mathbf{x})\right] \\ &= -\dot{\mathbf{x}}\mathbf{h}(\mathbf{x}, \dot{\mathbf{x}})\end{aligned}$$

เมื่อทำการอินทิเกรตจากเวลา t_0 ถึง t

$$E(t) - E(t_0) = -\int_{t_0}^t \dot{\mathbf{x}}\mathbf{h}(\mathbf{x}, \dot{\mathbf{x}})dt$$

ซึ่งทางเดินของเฟสนี้จะอยู่บนเฟสเพลนโดยแยกพิจารณาเป็น 2 กรณี

1 $\dot{\mathbf{x}}\mathbf{h}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{y}\mathbf{h}(\mathbf{x}, \mathbf{y}) > 0$ กรณีนี้พลังงานจะลดลงและ $\mathbf{h}$ มีพฤติกรรมเป็น

positive damping มีขนาดลดลง

2. $\dot{\mathbf{x}}\mathbf{h}(\mathbf{x}, \dot{\mathbf{x}}) = \mathbf{y}\mathbf{h}(\mathbf{x}, \mathbf{y}) < 0$ กรณีนี้พลังงานจะเพิ่มขึ้นโดยรับเข้ามาจากภายนอก

ระบบ ระบบหนึ่งอาจมีทั้ง 2 ลักษณะตัวอย่างเช่นนาฬิกาแบบลูกตุ้มที่มีลูกน้ำหนักเป็นตัวชดเชยพลังงานเป็นต้น

นิยามที่ 5 สำหรับฟังก์ชันสเกลาร์ต่อเนื่อง $V(\mathbf{x})$ เรียกว่า Locally positive definite ถ้า $V(0)=0$ และในขอบเขต BR_0

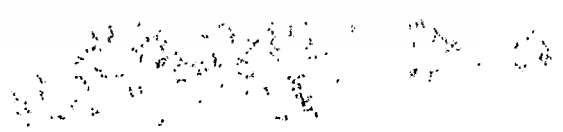
$$\text{ถ้า } \mathbf{x} = 0 \text{ แล้ว } V(\mathbf{x}) > 0 \quad \text{ถ้า } V(0)=0$$

และสามารถรักษาสถานะดังกล่าวข้างต้นทั้งสเปซเฟส เรียก $V(\mathbf{x})$ นี้ว่า Globally positive definite

นอกจากนี้ $V(\mathbf{x})$ เรียก Negative definite ถ้า $-V(\mathbf{x})$ เป็น Positive definite

$V(\mathbf{x})$ เรียก Positive semi-definite ถ้า $V(0)=0$ และ $V(\mathbf{x}) \geq 0$ สำหรับ $\mathbf{x}=0$

$V(\mathbf{x})$ เรียก Negative semi-definite ถ้า $-V(\mathbf{x})$ เป็น Positive semi-definite



นิยามที่ 6 ถ้าในบริเวณ BR_0 ฟังก์ชัน $V(\mathbf{x})$ เป็น positive definite และมี partial derivative ที่ต่อเนื่องและถ้า time derivative ตามเส้นทางเดินของระบบ $\mathbf{x}=\mathbf{f}(\mathbf{x})$ เป็น Negative semi-definite

$$\dot{V}(\mathbf{x}) < 0$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (31) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แล้วเรียก $V(x)$ ว่า Lyapunov function

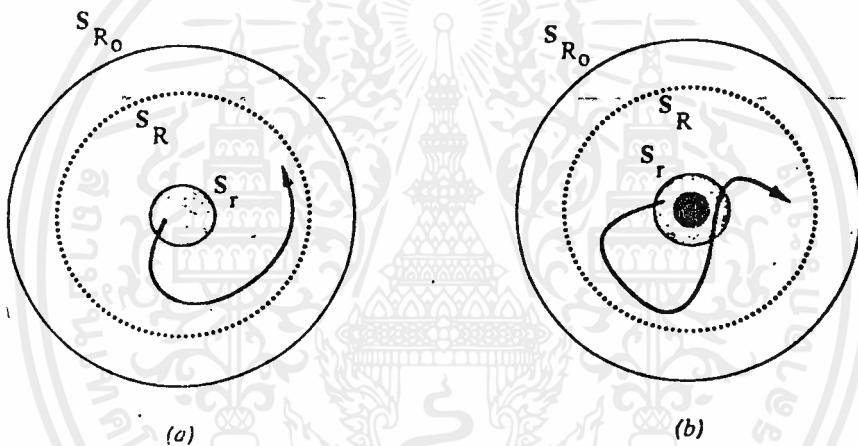
3.2.4 การพิจารณาเสถียรภาพรอบจุดสมดุล

Local and Global Stability

Local Stability ถ้าในบริเวณ B_{R0} มีสเกลาร์ฟังก์ชัน $V(x)$ มีความต่อเนื่องสำหรับ first partial derivative โดยที่

- $V(x)$ เป็น positive definite
- $\dot{V}(x)$ เป็น negative semi-definite

ดังนั้น จุดสมดุลที่จุด 0 จะเสถียร และถ้า $V(x)$ เป็น locally negative definite ใน B_{R0} สรุปได้ว่าเสถียรเชิงอะซิมโทติก



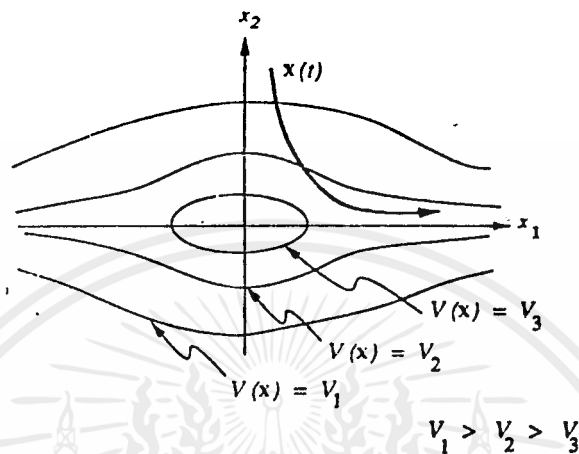
รูปที่ 5 แสดง Local Stability

Global stability ถ้ามีสเกลาร์ฟังก์ชัน $V(x)$ มีความต่อเนื่องสำหรับ first partial derivative โดยที่

- $V(x)$ เป็น positive definite
- $\dot{V}(x)$ เป็น negative definite
- $V(x) \rightarrow \infty$ เมื่อ $\|x\| \rightarrow \infty$

ดังนั้น จุดสมดุลที่จุด 0 จะเสถียรแบบ globally asymptotically stable มีการเพิ่มเงื่อนไข $V(x) \rightarrow \infty$ เมื่อ $\|x\| \rightarrow \infty$ เข้าไป หมายความว่า $V(x)$ ต้อง radially unbound นั่นคือ เมื่อ x มีค่าเข้าใกล้อนันต์ ค่า $V(x)$ จะเพิ่มค่าเข้าใกล้อนันต์ในทุกทิศทางเป็นการขยายขอบเขต

ของ B_{R0} ให้ครอบคลุมทั้งสเตตสเปซ จากรูปที่ 6 จะเห็นว่า ถ้า สเตตเริ่มต้นภายใน V_3 จะลู่เข้าสู่ออริจิน แต่ถ้าเริ่มต้นนอก V_3 แล้ว เมื่อสเตต x ค่ามากขึ้น ค่า $V(x)$ ก็จะมีค่ามากขึ้นจนถึงอนันต์. ถึงแม้ว่า $V(x)$ จะลู่ออกแต่อัตราการลู่ออก $\dot{V}(x)$ จะลดลงเรื่อยๆ (เป็น Negative definite)



รูปที่ 6 เงื่อนไขของ radial unboundedness

3.2.5 ทฤษฎีอินวาเรียนท์เซต (Invariant Set)

เพื่อขยายขอบเขตของ Lyapunov Theorem จะใช้ทฤษฎีอินวาเรียนท์เซตเข้าช่วยอธิบายการเสถียรเชิงอะซิมโทติกในแง่ของการลู่เข้า (Convergence) มากกว่าในแง่ของสภาวะสมดุลและ จะพิจารณาเงื่อนไข Negative semi-definite ของ $V(x)$ ในการตรวจสอบเสถียรภาพ

นิยามที่ 7 เซต G เป็น อินวาเรียนท์เซตสำหรับระบบไดนามิกส์ ถ้าทุกๆ เส้นทางเดินของระบบเริ่มจากจุดใดจุดหนึ่งใน G จะคงอยู่ใน G ตลอดเวลา

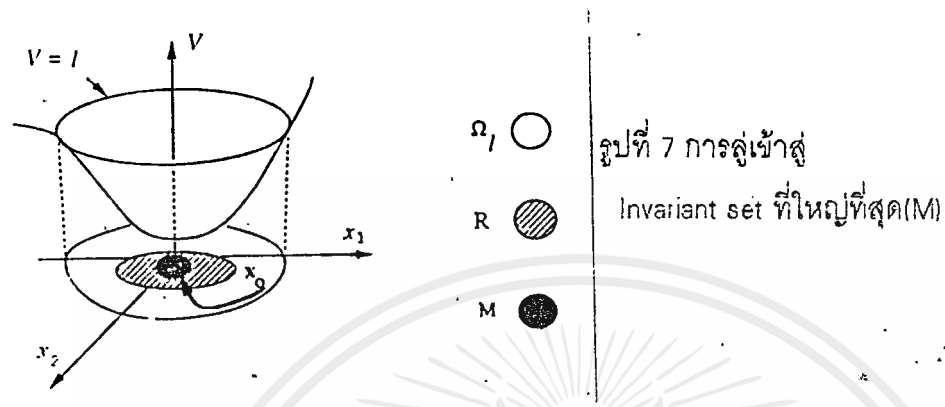
ตัวอย่างของอินวาเรียนท์เซตเช่น จุดสมดุล, domain of attraction รวมทั้งลิมิตไซเคิลด้วย

ทฤษฎีอินวาเรียนท์เซตในบริเวณแคบๆ (Local invariant set Theorem)

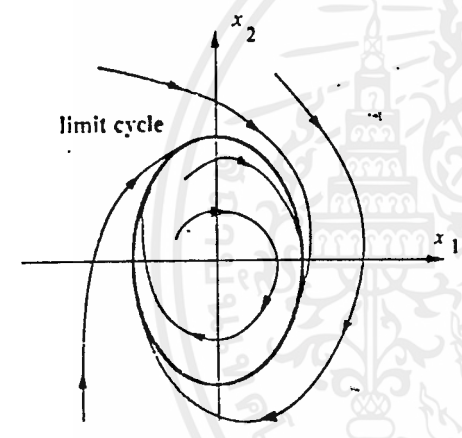
พิจารณาระบบ $\dot{x} = f(x)$ โดย f ต่อเนื่องและให้ $V(x)$ เป็นสเกลาร์ฟังก์ชันที่ต่อเนื่องสำหรับ first partial derivative สมมติให้

- o สำหรับ $\epsilon > 0$ บริเวณ U_ϵ กำหนดโดย $V(x) < \epsilon$ มีขอบเขตจำกัด (bounded)
- o $\dot{V}(x) < 0$ สำหรับทุกค่า x ใน U_ϵ ให้ R เป็นเซตของจุดทุกจุดใน U_ϵ

ที่ทำให้ $V(x)=0$ และ M เป็น อินวาเรียนท์เซต ที่ใหญ่ที่สุดใน R ดังนั้นทุกๆ ผลเฉลย $x(t)$ ที่กำเนิดใน U_1 จะลู่เข้าสู่ M เมื่อ $t \rightarrow \infty$



- Ω_1 รูปที่ 7 การลู่เข้าสู่
- R Invariant set ที่ใหญ่ที่สุด(M)
- M



จากรูปที่ 8 limit cycle
จะเห็นการลู่เข้าสู่ของทางเดินของระบบสู่ limit cycle จึงไม่มีการเปลี่ยนแปลงของพลังงาน $V(x)=0$ แต่ค่าของพลังงานจะไม่เท่ากับศูนย์ อาจเท่ากับค่าคงที่ค่าหนึ่งซึ่ง limit cycle นี้ก็คือ อินวาเรียนท์เซต M นั่นเองระบบจึงไม่เสถียรที่ จุดออร์จิน

รูปที่ 8 การลู่เข้าสู่ limit cycle

เราอาจจะเรียกการลู่เข้าสู่อินวาเรียนท์เซตว่าเสถียรเชิงอะซิมโทติกได้ ถึงแม้ว่าจะเป็น การลู่เข้าสู่ ลิมิตไซเคิลก็ตามเพราะอินวาเรียนท์เซตมีความหมายรวมถึงลิมิตไซเคิลด้วยจึงเป็น การขยายแนว ความคิดของทฤษฎีลียาปูนอฟ ซึ่งพิจารณาเพียงการลู่เข้าสู่จุดสมดุลอย่างเดียว

ทฤษฎีอินวาเรียนท์เซตในบริเวณกว้าง(Global invariant set Theorem)

พิจารณาระบบ $\dot{x} = f(x)$ โดย f ต่อเนื่องและให้ $V(x)$ เป็นสเกลาร์ฟังก์ชันที่ต่อเนื่องสำหรับ first partial derivative สมมติให้

- o สำหรับ $\dot{V}(x) < 0$ สำหรับทั้งสเทสเปซ
- o $V(x) \rightarrow \infty$ เมื่อ $\|x\| \rightarrow \infty$

ให้ R เป็นเซตของจุดทุกจุดที่ $V(x)=0$ และ M เป็นอินวาเรียนท์เซตที่ใหญ่ที่สุดใน R ดังนั้น
 ผลเฉลยทั้งหมดจะลู่เข้าแบบ globally asymptotically สู่ M เมื่อ $t \rightarrow \infty$

เป็นการเพิ่มเงื่อนไข radially unbounded เข้าไปเช่นเดียวกับ Lyapunov's Global stability
 และขยายขอบเขตของ R ออกเต็มสเปซ

3.3 การหาเงื่อนไขเสถียรภาพของลิมิตไซเคิล

เนื่องจากลิมิตไซเคิล(limit cycle) เป็นผลเฉลยที่เป็นคาบ(periodic solution)ของระบบอยู่ด้วย
 เงื่อนไขของการที่อัตราการเปลี่ยนแปลงพลังงานเป็นศูนย์

พิจารณาสมการทั่วไปของระบบ

$$\ddot{x} + \epsilon h(x, \dot{x}) + x = 0$$

เขียนเป็นสมการสเตท

$$\dot{x} = y, \quad \dot{y} = -\epsilon h(x, y) - x$$

โดยที่ขนาดของ ϵ มีค่าน้อยกว่า 1 มากๆ ดังนั้นเทอม $\epsilon h(x, y)$ ที่ไม่เป็นเชิงเส้นจึงอาจ
 ตัดทิ้งได้เหลือเพียง

$$\ddot{x} + x = 0$$

จะมีจุดสมดุลอยู่ที่ออริจิน ซึ่งเป็นไปได้ที่มีอย่างน้อยหนึ่งผลเฉลยที่เป็นคาบที่มีทางเดินของ
 เฟสล้อมรอบออริจินผลเฉลยของสมการที่ตัดเทอม $\epsilon h(x, y)$ ออกคือ

$$x(t) = a \cos(t + \alpha)$$

เป็นวงกลมรัศมี a มีคาบ 2π เขียนเป็นสมการวงกลมได้ดังนี้

$$x^2 + y^2 = a^2$$

จากสมการการเปลี่ยนแปลงพลังงานที่หาไว้แล้วข้างต้น

$$E(t) - E(t_0) = -\int_{t_0}^t \dot{x} \epsilon h(x, y) dt$$

แทนค่าผลเฉลย $x(t) = a \cos t$ (ให้ $\alpha = 0$) และ $\dot{x}(t) = -a \sin(t)$ ลงไป

$$E(2\pi) - E(0) = \epsilon a \int_0^{2\pi} h(a \cos t, -a \sin t) \sin t dt = g(a)$$

ถ้าให้ a_0 เป็นรัศมีของลิมิตไซเคิลดังนั้น $g(a_0) = 0$ เพราะไม่มีการเปลี่ยนแปลงของ
 พลังงานบน ลิมิตไซเคิลแสดงว่าที่รัศมีน้อยกว่า a_0 พลังงานจะมีอัตราเพิ่มขึ้นและที่รัศมีมากกว่า
 a_0 พลังงานจะมีอัตราลดลง

$$g(a) > 0 \text{ เมื่อ } a_0 - \beta < a < a_0, \quad \beta > 0$$

$$g(a) < 0 \text{ เมื่อ } a_0 < a < a_0 + \beta$$

อาจกล่าวได้ว่าเงื่อนไข

$$g(a_0) < 0$$

เพียงพอที่จะสรุปว่า ลิมิตไซเคิลนี้มีเสถียรภาพ

3.4 ฟังก์ชันลียาปูนอฟสำหรับระบบต่อเนื่องเชิงเส้นไม่ขึ้นกับเวลา

กำหนดให้ระบบต่อเนื่องเชิงเส้นไม่ขึ้นกับเวลา $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$ มี Quadratic Lyapunov function คือ

$$V(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x}$$

เมตริกซ์ $\mathbf{P}$ เป็น Symmetric positive definite matrix ถ้าเราหาอนุพันธ์เทียบกับเวลาของ $V(\mathbf{x})$

$$\begin{aligned} \dot{V}(\mathbf{x}) &= \mathbf{x}^T [\mathbf{A}^T \mathbf{P} + \mathbf{P} \mathbf{A}] \mathbf{x} \\ &= -\mathbf{x}^T \mathbf{Q} \mathbf{x} \end{aligned}$$

โดยที่ $\mathbf{Q} = -(\mathbf{A}^T \mathbf{P} + \mathbf{P} \mathbf{A})$ ระบบจะเสถียรเชิงอะซิมโทติกถ้า $-\mathbf{Q}$ เป็น negative definite

แต่สำหรับระบบไม่เชิงเส้นเมื่อหาเมตริกซ์ $\mathbf{A}$ โดยทำการตัดเทอมที่มี order เกิน 1 ออกไป เขามาแต่เพียงสัมประสิทธิ์ของเทอมที่เป็นกำลัง 1 เมื่อหา $\mathbf{P}$ ออกมาแล้วทดสอบว่าเป็น positive definite สรุปได้ว่าเสถียรเชิงอะซิมโทติกแต่ไม่อาจสรุปได้ว่าเสถียรในขอบเขตกว้างได้ เพียงแต่แสดงให้เห็นว่าเสถียรรอบๆจุดที่พิจารณาเท่านั้น

3.5 วิธีของคราซอฟสกี (Krasovski's Method)

เป็นวิธีหนึ่งที่ใช้หา ฟังก์ชันลียาปูนอฟ สำหรับระบบไม่เชิงเส้น ซึ่งทฤษฎีของคราซอฟสกี จะพิจารณาระบบ $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ มีจุดสมดุลที่สนใจอยู่ที่จุดกำเนิด

ให้ $\mathbf{A}(\mathbf{x})$ เป็น Jacobian matrix ของระบบ

$$\mathbf{A}(\mathbf{x}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}$$

ถ้าเมตริกซ์ $\mathbf{F} = \mathbf{A} + \mathbf{A}^T$ เป็น negative definite ในบริเวณ U ดังนั้น จุดสมดุล ที่จุดกำเนิดจะเสถียรเชิงอะซิมโทติก Lyapunov function สำหรับระบบคือ

$$V(\mathbf{x}) = \mathbf{f}^T(\mathbf{x}) \mathbf{f}(\mathbf{x})$$

ถ้า U แทนทั้งสเปซและ $V(\mathbf{x}) \rightarrow \infty$ เมื่อ $\|\mathbf{x}\| \rightarrow \infty$ ดังนั้นจุดสมดุลจะเสถียรเชิงอะซิมโทติกในขอบเขตกว้าง

บทที่ 4

การควบคุมระบบที่ไม่เป็นเชิงเส้น

4.1. บทนำ

สำหรับบทที่ 1, 2 และ 3 เราทำการศึกษาวิธีต่างๆที่ใช้ในการวิเคราะห์ระบบไม่เป็นเชิงเส้นโดยการสมมติให้ระบบควบคุมนั้น ๆ ถูกออกแบบเป็นที่เรียบร้อยแล้ว สำหรับบทนี้เราจะศึกษาถึงการออกแบบระบบควบคุมสำหรับระบบไม่เป็นเชิงเส้น โดยจะเลือกศึกษาตัวควบคุมที่มีลักษณะแบบปรับตัว(Adaptive controller)ได้

4.1.1 แนวความคิดพื้นฐานในการออกแบบระบบควบคุม

โดยทั่วไปที่เราเคยศึกษามา อันดับแรกเราจะทำการเลือกโครงสร้างของตัวควบคุมที่เราต้องการ(เช่นตัวควบคุมแบบพีไอดี(PID) หรือตัวควบคุมแบบกำหนดตำแหน่งโพล(pole-placement) เป็นต้น)

จากนั้นเราก็คำนวณหาค่าพารามิเตอร์ของตัวควบคุมโดยพิจารณาจากพลานท์ที่เรารู้ค่าพารามิเตอร์ต่างๆ อย่างแน่นอน ตัวควบคุมลักษณะนี้เราเรียกว่า ตัวควบคุมที่มีค่าคงที่ (fixed controller) ซึ่งในทางปฏิบัติค่าพารามิเตอร์ที่แน่นอนของพลานท์บางครั้งเราไม่สามารถทราบค่าที่แน่นอนได้ตลอดจนคุณสมบัติของพลานท์ที่มีลักษณะไม่เป็นเชิงเส้นเราจึงจำเป็นต้องมีตัวควบคุมแบบปรับตัวเองตัวควบคุมแบบนี้ ค่าพารามิเตอร์ของตัวควบคุมมีค่าที่ไม่ตายตัว แต่จะมีการเปลี่ยนแปลง ซึ่งการควบคุมการเปลี่ยนแปลงพารามิเตอร์ของตัวควบคุมนี้จะถูกจัดการโดยกฎการปรับตัว(Adaptation law) ซึ่งเราจะได้กล่าวถึงในหัวข้อถัดไป

ปัญหาเกี่ยวกับการควบคุมระบบที่ไม่เป็นเชิงเส้นเราแยกการพิจารณาออกเป็น 2 ประเภทคือ

ปัญหาเกี่ยวกับเสถียรภาพ(Stability problem)

กรณีนี้ตัวควบคุมของเราจะเรียกว่าตัวรักษาเสถียรภาพ(stabilizer หรือ regulator) ของระบบ ซึ่งลักษณะการออกแบบ จะเป็นไปในแนวทางเพื่อที่จะทำให้ตัวแปรสเตทของระบบมีลักษณะเสถียรรอบจุดสมดุล(equilibrium point) นั่นคือเราอาจกล่าวได้ว่าสำหรับระบบที่มีสมการเป็น

$$\dot{x} = f(x, u, t)$$

เราจะหาตัวควบคุม u เพื่อให้ตัวแปรสเตท x ที่มีจุดเริ่มต้นที่จุดใดๆในบริเวณหนึ่ง มีลักษณะที่ลู่เข้าสู่ศูนย์ เมื่อเวลาเป็นอนันต์

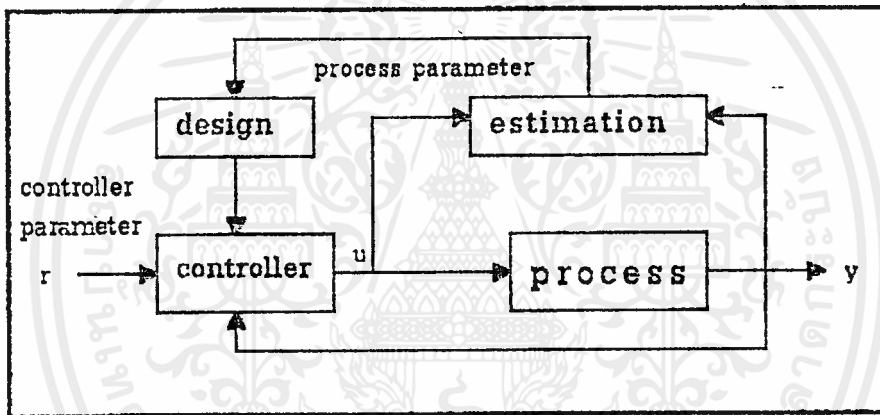
ปัญหาเกี่ยวกับการติดตาม (tracking problem)

กรณีนี้ตัวควบคุมของเราเรียกว่าตัวติดตาม(tracker) มีลักษณะการออกแบบจะเป็นไปในแนวทางเพื่อทำให้เอาต์พุตของระบบสามารถมีแนวทางการเคลื่อนที่ตามที่เราต้องการ นั่นคือเราอาจกล่าวได้ว่า

$$\text{สำหรับระบบที่มีสมการเป็น} \quad \dot{x} = f(x, u, t) \\ y = h(x)$$

และแนวทางการเคลื่อนที่ที่เราต้องการคือ $y_d(t)$ เราจะหาตัวควบคุม u โดยที่สำหรับตัวแปรสถานะ x ซึ่งมีจุดเริ่มต้นที่จุดใด ๆ ในบริเวณหนึ่งแล้ว ข้อผิดพลาดของการติดตาม $y(t) - y_d(t)$ มีค่าลู่เข้าหาศูนย์เมื่อเวลาเข้าใกล้อนันต์ โดยที่ตัวแปรสถานะของระบบถูกจำกัดค่าอยู่ในพื้นที่บริเวณหนึ่ง

โครงสร้างของระบบควบคุมแบบปรับตัวโดยทั่วไปมีลักษณะดังรูป



ระบบควบคุมแบบปรับตัวประกอบด้วยส่วนสำคัญ 2 ส่วนคือ

1)ระบบป้อนกลับหลัก

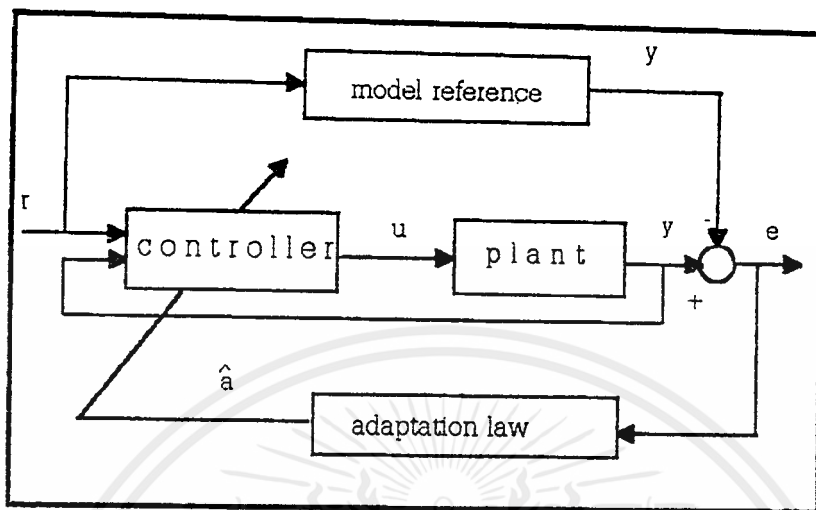
เป็นส่วนที่พิจารณาถึงการเปลี่ยนแปลงสัญญาณของกระบวนการ

2)ระบบป้อนกลับรอง

เป็นส่วนที่พิจารณาถึงการเปลี่ยนแปลงพารามิเตอร์ของกระบวนการนั่นคือส่วนป้อนกลับหลัก จะถูกใช้ดังเช่นตัวควบคุมที่มีค่าพารามิเตอร์คงที่ ในขณะที่ส่วนป้อนกลับรองจะเป็นส่วนที่ทำให้ตัวควบคุมมีลักษณะเป็นแบบปรับตัวเอง ระบบควบคุมแบบปรับตัวเองที่ใช้กันโดยทั่วไปแบ่งออกเป็น 2 ลักษณะคือ

การควบคุมแบบมีโมเดลอ้างอิง(MODEL REFERENCE ADAPTIVE CONTROL :MRAC)

โครงสร้างของระบบควบคุมแบบนี้โดยทั่วไปมีลักษณะดังรูป



จะเห็นได้ว่าระบบควบคุมที่มีโมเดลอ้างอิง(MRAC) จะประกอบด้วยส่วนสำคัญ 4 ส่วนคือ ฟลანท์ ในที่นี้สมมติว่าเรารู้โครงสร้างของฟลานท์แน่นอน โดยที่เราไม่ทราบค่าพารามิเตอร์ต่างๆ ของฟลานท์ กล่าวคือ

- กรณีระบบเป็นเชิงเส้น เรารู้จำนวนของโพล(pole) และซีโร่(zeros) โดยไม่รู้ตำแหน่งที่แน่นอน
- กรณีระบบไม่เป็นเชิงเส้น เรารู้โครงสร้างทางไดนามิกของระบบ แต่ไม่รู้ค่าพารามิเตอร์ต่าง ๆ ที่แน่นอน

โมเดลอ้างอิง เป็นตัวกำหนดผลตอบสนองทางอุดมคติของระบบจะต้องอยู่ภายใต้ขอบเขตบางอย่างคือ ทำให้ผลตอบสนองของโมเดลนี้มีพฤติกรรมของผลตอบสนองชั่วคราวและผลตอบสนองทางความถี่เป็นไปตามที่เราต้องการ

ตัวควบคุม เราแยกการพิจารณาออกเป็น 2 กรณีคือ

- 1) หากเรารู้ค่าพารามิเตอร์ของระบบโดยแน่นอน

ตัวควบคุมจะต้องทำให้สัญญาณเอาต์พุต ของฟลานท์มีลักษณะเหมือนกับสัญญาณเอาต์พุตของโมเดลอ้างอิง

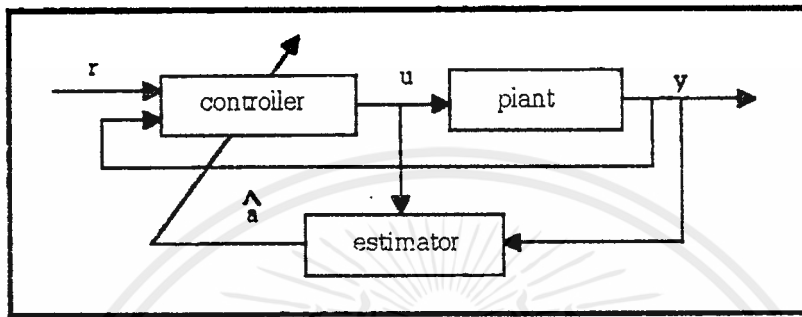
- 2) หากเราไม่ทราบค่าพารามิเตอร์ที่แท้จริงของระบบ

กรณีนี้กฎการควบคุมจะต้องปรับค่าพารามิเตอร์ของตัวควบคุมเพื่อให้การติดตาม (tracking) มีลักษณะเป็นแอสซิมโทติก(asymptotic)นั่นเอง

4.1.2 กฎการควบคุม(Control law)

จะคำนวณค่าพารามิเตอร์เพื่อให้ผลตอบสนองของพลานท์ภายใต้การควบคุมแบบปรับตัวเอง มีลักษณะเหมือนกับผลตอบสนองของโมเดลอ้างอิง

การควบคุมแบบปรับจูนตัวเอง(SELF-TUNING CONTROL)



ในเวลานั้นๆ ตัวประมาณค่า(estimator) จะส่งเซตของค่าประมาณของพารามิเตอร์ ของพลานท์ไปยังตัวควบคุม(สัญญาณ $\hat{a}$) โดยตัวประมาณค่าจะคำนวณสัญญาณ $\hat{a}$ จากค่าสัญญาณอินพุต u และสัญญาณเอาต์พุต y ของพลานท์ในอดีต จากนั้นตัวควบคุมจะคำนวณค่าพารามิเตอร์ที่เหมาะสมของตัวควบคุมเองและส่งสัญญาณควบคุม u ซึ่งขึ้นอยู่กับพารามิเตอร์ของตัวควบคุมออกไปยังกระบวนการ ซึ่งสัญญาณ u นี้จะทำให้ค่าเอาต์พุตของพลานท์เปลี่ยนไปเป็นค่าใหม่ ตัวประมาณค่าก็จะประมาณค่าพารามิเตอร์ของพลานท์ใหม่

และระบบจะมีพฤติกรรมเช่นนี้เรื่อยไป

ส่วนเทคนิคในการประมาณค่าพารามิเตอร์

ของพลานท์นี้สำหรับระบบเชิงเส้นวิธีที่นิยมใช้กันคือ วิธี

กำลังสองน้อยที่สุด(least square techniques) ส่วนวิธีอื่นๆ ที่มีการใช้กันได้แก่ การกำหนดตำแหน่งโพล(pole-placement), วิธีพีไอดี(PID),วิธีควอดราติกเชิงเส้น(linear quadratic control:LQR), วิธีหาโดยใช้หลักการแปรผันน้อยที่สุด(Minimum variance control)หรือ H_∞ เป็นต้น

เราจะเห็นได้ว่าระบบควบคุมที่มีโมเดลอ้างอิงค่าพารามิเตอร์จะถูกปรับเพื่อให้ผลต่างระหว่างผลตอบสนองของพลานท์ภายใต้การควบคุมแบบปรับตัวเองและผลตอบสนองของโมเดลอ้างอิงมีการลู่เข้าสู่ศูนย์เมื่อเวลามีค่าเพิ่มขึ้นในขณะที่ระบบปรับจูนตัวเองค่าพารามิเตอร์จะถูกปรับเพื่อให้เกิดการสอดคล้องกัน ระหว่างสัญญาณอินพุตและสัญญาณเอาต์พุตของพลานท์

4.2 ทฤษฎีที่ใช้มากสำหรับการออกแบบระบบควบคุม

ในหัวข้อนี้จะกล่าวถึงทฤษฎีบทต่างๆซึ่งจะเป็นประโยชน์ต่อการศึกษาระบบการควบคุมแบบปรับตัว

จากทฤษฎี เราจะกล่าวว่าทรานสเฟอร์ฟังก์ชัน $H(s)$ มีลักษณะเป็น จำนวนจริงบวกเสมอ Strickly Positive Real:SFR) ก็ต่อเมื่อ

- $H(s)$ เป็นทรานสเฟอร์ฟังก์ชันที่เสถียร
- ส่วนจริงของ $H(s)$ มีค่าเป็นบวกเมื่อเทียบกับแกน $j\omega$ นั่นคือ

จากทฤษฎีนี้เราสามารถทราบถึงเงื่อนไขที่จำเป็นของ $H(s)$ เพื่อให้มีลักษณะเป็น SFR ได้แก่

- 1) $H(s)$ มีดีกรีสัมพัทธ์ (Relative degree) เป็น 0 หรือ 1 (ดีกรีสัมพัทธ์เป็นผลต่างของดีกรีระหว่างเศษและส่วนของโพลีโนเมียล)
- 2) $H(s)$ มีลักษณะเป็นทรานสเฟอร์ฟังก์ชันที่มีเฟสน้อยที่สุด (Minimum phase transfer function) กล่าวคือมีขั้วโรยู่ทางด้านซ้ายของระนาบ s ทฤษฎีพิจารณาสัญญาณค่าผิดพลาดซึ่งมีความสัมพันธ์เป็นดังสมการ

$$e(t) = H(s)[k\phi^T(t)\mathbf{v}(t)]$$

เมื่อ $e(t)$ เป็นสัญญาณเอาต์พุต(สเกลลาร์)

$\phi(t)$ เป็น $m \times 1$ เวกเตอร์

$\mathbf{v}(t)$ เป็น $m \times 1$ เวกเตอร์ของสัญญาณที่วัดได้

$H(s)$ เป็น ทรานสเฟอร์ฟังก์ชันที่เป็นจำนวนจริงบวก

k เป็นค่าคงที่ที่ไม่ทราบค่าแต่ทราบเครื่องหมาย

$$\text{ถ้า } \dot{\phi}(t) = -\text{sgn}(k)\gamma e\mathbf{v}(t)$$

โดย γ เป็นค่าคงที่ค่าบวก

แล้วเราสามารถสรุปได้ว่า $e(t)$ และ $\phi(t)$ จะเป็นสัญญาณที่มีขอบเขต

หมายเหตุ: กรณีที่ $\mathbf{v}(t)$ เป็นสัญญาณที่มีขอบเขตแล้วเราจะได้ $e(t)$ จะเข้าใกล้ศูนย์เมื่อเวลาเข้าใกล้อนันต์ (Globally stabilization)

4.3 ระบบควบคุมแบบมีโมเดลอ้างอิง(MRAC)สำหรับอันดับหนึ่ง (First order system)

4.3.1 กรณีระบบเชิงเส้น

สำหรับหัวข้อนี้เราจะพิจารณาการควบคุมแบบ MRAC สำหรับระบบที่เป็นอันดับหนึ่งเพื่อให้ทราบแนวความคิดทั่วไปเกี่ยวกับการควบคุมแบบ MRAC โดยจะทำการพิจารณาสำหรับระบบที่เป็นเชิงเส้น และไม่เชิงเส้น สำหรับกรณีของระบบที่เป็นอันดับใด ๆ จะกล่าวถึงในหัวข้อถัดไป

สำหรับระบบเชิงเส้นอันดับหนึ่ง

$$\text{ส่วนที่เป็นพลานท์:} \quad \dot{y} = -a_p y + b_p u$$

โดยที่ y เป็นเอาต์พุตของพลานท์

u เป็นอินพุตของพลานท์

a_p, b_p เป็นค่าคงที่ที่ไม่ทราบค่า(แต่รู้เครื่องหมาย)ที่เป็นพารามิเตอร์ของพลานท์

$$\text{ส่วนที่เป็นโมเดล:} \quad \dot{y}_m = -a_m y_m + b_m r(t)$$

โดยที่ a_m, b_m เป็นค่าคงที่ที่มีค่าเป็นบวก

$r(t)$ เป็นสัญญาณอ้างอิงภายนอกที่มีขอบเขตจำกัด

$$\text{นั่นคือ } Y(s) = M(s)R(s) \text{ เมื่อ } M(s) = \frac{b_m}{s+a_m}$$

สังเกต $M(s)$ มีส่วนจริงเป็นบวก(strictly positive real transfer function)

$$\text{กฎการควบคุม(control law) เลือก } u = \hat{a}_r r(t) + \hat{a}_y y(t)$$

$$\text{เราจะได้สมการเป็น } \dot{y} = -(a_p - \hat{a}_y b_p)y + \hat{a}_r b_p r(t)$$

สังเกตหากเราทราบค่าของ a_p และ b_p ของพลานท์และเราเลือกให้

$$\hat{a}_r = a_r^* = \frac{b_m}{b_p}$$
$$\hat{a}_y = a_y^* = \frac{a_p - a_m}{b_p}$$

$$\text{เราจะได้สมการเป็น } \dot{y} = -a_m y + b_m u$$

ซึ่งเหมือนกับโมเดลอ้างอิงนั่นเอง แต่ในความเป็นจริงเราจะกำหนดให้ $\hat{a}_r$ มีค่าเป็น a_r^* และ $\hat{a}_y$ มีค่าเป็น a_y^* โดยตรงไม่ได้เราจึงจำ

เป็นต้องคำนวณหากฎการควบคุมสำหรับควบคุมการเปลี่ยนแปลงค่าพารามิเตอร์ $\hat{a}_r$ และ $\hat{a}_y$ นั่นเอง

ต่อไปหา กฎการปรับตัว(adaption laws)

กำหนด $e = y - y_m$ (ค่าผิดพลาดของการ tracking)

$$\tilde{a}_r = \hat{a}_r - a_r^* \text{ (ค่าผิดพลาดของพารามิเตอร์)}$$

$$\tilde{a}_y = \hat{a}_y - a_r^*$$

เราจะได้ความสัมพันธ์เป็นดังสมการ

$$e = y - y_m = -(a_p - \hat{a}_r b_p)y + \hat{a}_r b_p r(t) + a_m y_m - b_m r(t)$$

$$\begin{aligned} \dot{e} &= -a_m(y - y_m) + (a_m - a_p + b_p \hat{a}_y)y + (b_p \hat{a}_r - b_m)r(t) \\ &= -a_m e + b_p(\tilde{a}_r r + \tilde{a}_y y) \end{aligned}$$

$$\text{นั่นคือ } e = \frac{b_p}{s+a_m}(\tilde{a}_r r + \tilde{a}_y y) = \frac{1}{a_r^*} M(s)[(\tilde{a}_r r + \tilde{a}_y y)]$$

ความสัมพันธ์ระหว่างค่าผิดพลาดของการติดตาม(tracking) และค่าผิดพลาดของพารามิเตอร์

มีลักษณะเช่นเดียวกับทฤษฎีที่ได้กล่าวมาแล้ว

นั่นคือเราจะได้กฎการควบคุมเป็นดังสมการ

$$\dot{\hat{a}}_r^* = -\text{sgn}(b_p) \gamma e r$$

$$\dot{\hat{a}}_y^* = -\text{sgn}(b_p) \gamma e y \quad \gamma \text{ เป็นค่าคงที่ที่เป็นบวก}$$

ข้อสังเกต 1) เราสามารถแสดงโดยใช้ทฤษฎีของลีอาปูนอฟ ว่าสัญญาณ e จะมีลักษณะลู่เข้าหาศูนย์

2) การประมาณค่าพารามิเตอร์ของระบบจะถูกต้องแค่ไหนจะขึ้นกับลักษณะของสัญญาณ

อ้างอิงด้วยกล่าวคือ สัญญาณต้องมีการเปลี่ยนแปลงอย่างต่อเนื่อง(persistent excitation) กำหนดให้

$\mathbf{V} = \begin{bmatrix} r & y \end{bmatrix}^T$ เราจะได้ว่าการประมาณค่าพารามิเตอร์จะถูกต้อง เมื่อทุกๆค่า t ที่เป็นบวก $\mathbf{V}$ สอดคล้องกับความสัมพันธ์

$$\int_t^{t+T} \mathbf{V}^T \mathbf{V} dr \geq \alpha_1 \mathbf{I}$$

สำหรับค่าคงที่ที่เป็นบวกบางค่า α_1 และ $\mathbf{I}$

เมื่อพิจารณาในเชิงกายภาพเราสามารถกล่าวได้ว่า การที่จะประมาณค่าพารามิเตอร์ให้มีค่าถูกต้องเป็นจำนวน m ค่า สัญญาณอ้างอิงควรประกอบด้วยองค์ประกอบที่เป็น sine อย่างน้อย $\frac{m}{2}$ ลูก

4.3.2 กรณีระบบไม่เป็นเชิงเส้น

$$\text{ส่วนที่เป็นพลานท์: } \dot{y} = -a_p y - c_p f(y) + b_p u$$

โดยที่ y เป็นเอาต์พุตของพลานท์

u เป็นอินพุตของพลานท์

a_p, b_p เป็นค่าคงที่ไม่ทราบค่า(แต่รู้เครื่องหมาย)

ที่เป็นพารามิเตอร์ของพลานท์

$f(y)$ เป็นฟังก์ชันไม่เป็นเชิงเส้นที่เรา

ส่วนที่เป็นโมเดล : $\dot{y}_m = -a_m y_m + b_m r(t)$

โดยที่ a_m, b_m เป็นค่าคงที่ที่มีค่าเป็นบวก

$r(t)$ เป็นสัญญาณอ้างอิงภายนอกที่มีขอบเขตจำกัด

กฎการควบคุม เลือก $u = \hat{a}_r r(t) + \hat{a}_y y + \hat{a}_f f(y)$

เราจะได้สมการเป็น

$$\dot{y} = -(a_p - \hat{a}_y b_p)y + \hat{a}_r b_p r(t) + (c_p + \hat{a}_f b_p)f(y)$$

สังเกต หากเราทราบค่าของ a_p, b_p และ c_p ของพลานท์และเราเลือก

$$\hat{a}_r = a_r^* = \frac{b_m}{b_p}$$

$$\hat{a}_y = a_y^* = \frac{a_p - a_m}{b_p}$$

$$\hat{a}_f = a_f^* = \frac{c_p}{b_p}$$

เราจะได้สมการเป็น

$$\dot{y} = -a_m y + b_m u$$

ซึ่งเหมือนกับโมเดล อ้างอิงนั่นเองแต่ในความเป็นจริงเราจะกำหนดให้ $\hat{a}_r$ มีค่าเป็น a_r^* ส่วน $\hat{a}_y$ มีค่าเป็น a_y^* และ $\hat{a}_f$ มีค่าเป็น a_f^* โดยตรงไม่ได้ เราจึงจำเป็นต้องคำนวณหากฎการควบคุมสำหรับควบคุมการเปลี่ยนแปลงของค่าพารามิเตอร์ $\hat{a}_r, \hat{a}_y$ และ $\hat{a}_f$ นั้นเอง

กฎการปรับตัว

กำหนด $e = y - y_m$ (ข้อผิดพลาดของการ tracking)

$$\tilde{a}_r = \hat{a}_r - a_r^* \quad (\text{ข้อผิดพลาดของพารามิเตอร์})$$

$$\tilde{a}_y = \hat{a}_y - a_y^*$$

$$\tilde{a}_f = \hat{a}_f - a_f^*$$

เราจะได้ความสัมพันธ์เป็นดังสมการ

$$\begin{aligned} y - y_m &= -(a_p - \hat{a}_y b_p)y + \hat{a}_r b_p r(t) + (-c_p + b_p \hat{a}_f)f(y) \\ &\quad + a_m y_m - b_m r(t) \end{aligned}$$

$$\text{นั่นคือ } e = \frac{1}{a_r^*} M(s) [\tilde{a}_r r + \tilde{a}_y y + \tilde{a}_f f(y)]$$

ความสัมพันธ์ระหว่างค่าผิดพลาดของการติดตาม(tracking)และค่าผิดพลาดของพารามิเตอร์มีลักษณะเช่นเดียวกับทฤษฎีดังกล่าวมาแล้ว

$$\begin{aligned}\dot{\hat{a}}_y^* &= -\text{sgn}(b_p)\gamma e y \\ \dot{\hat{a}}_r^* &= -\text{sgn}(b_p)\gamma e r \\ \dot{\hat{a}}_f^* &= -\text{sgn}(b_p)\gamma e f(y)\end{aligned}$$

จากหลักการของตัวควบคุมที่กล่าวมาข้างต้นนี้ เราสามารถแสดงให้เห็นจริงจากโปรแกรมจำลองระบบดังผลการทดลอง

ข้อสังเกต 1) เราสามารถแสดงโดยใช้ทฤษฎีของLyapunov ว่าสัญญาณจะมีลักษณะลู่เข้าหาศูนย์

2) การประมาณค่าพารามิเตอร์ของระบบจะถูกต้องแค่นั้นจะขึ้นกับลักษณะของสัญญาณ

อ้างอิงด้วยกล่าวคือ กำหนดให้ $\mathbf{v} = [r(t) \ y(t) \ f(y)]^T$

เราจะได้ว่าการประมาณค่าพารามิเตอร์จะถูกต้อง เมื่อสำหรับทุกค่า t ที่เป็นบวก $\mathbf{v}$ สอดคล้องกับความสัมพันธ์

$$\int_t^{t+T} \mathbf{v}^T \mathbf{v} dr = \alpha_1 \mathbf{I}$$

สำหรับค่าคงที่ที่เป็นบวกบางค่า α_1 และ $\mathbf{I}$

4.4 MRAC สำหรับระบบไม่เป็นเชิงเส้นอันดับ n ใดๆ

ส่วนที่เป็นพลาเน็ต กำหนดให้เราสามารถทำการวัดตัวแปรสแตตทุกตัวของระบบ

พลาเน็ตจะอยู่

ในรูป

$$y^{(n)} + \sum_{i=1}^n \alpha_i f_i(\mathbf{x}, t) = bu$$

เมื่อ $\mathbf{x}$ เป็นสแตตเวกเตอร์ $= [y \ \dot{y} \ \dots \ y^{(n-1)}]^T$

f_i เป็นฟังก์ชันที่ไม่เป็นเชิงเส้นของสแตตและเวลา

i เป็นค่าคงที่ที่ไม่ทราบค่า

b เป็นค่าคงที่ที่ไม่ทราบค่า(แต่รู้เครื่องหมาย)

จัดรูปใหม่เป็น

$$hy^{(n)} + \sum_{i=1}^n a_i f_i(\mathbf{x}, t) = u \quad \text{--- (1)}$$

โดย $h = \frac{1}{b}$, $a_i = \frac{\alpha_i}{b}$

กฎควบคุม

กำหนดให้ค่าผิดพลาดรวมของทุกสเตจ(combine error)เป็นดังสมการ

$$S = e^{(n-1)} + \lambda_{n-2}e^{(n-2)} + \dots + \lambda_0 e = \Delta(s)e$$

ซึ่ง $\Delta(s) = s^{(n-1)} + \lambda_{n-2}s^{(n-2)} + \dots + \lambda_0$

เป็นสมการโพลิโนเมียลที่เสถียรในเทอมของตัวแปร ลาปลาซ(Laplace variable) S

โดย e เป็นค่าผิดพลาดของการติดตาม(tracking)

จัดรูปใหม่เป็น $S = y^{(n-1)} - y_r^{(n-1)}$

เมื่อ $y_r^{(n-1)} = y_d^{(n-1)} - \lambda_{n-2}e^{(n-2)} - \dots - \lambda_0 e$

สังเกตว่า $y^{(n-1)} - y_d^{(n-1)} = e^{(n-1)}$

พิจารณากฎการควบคุม

$$u = hy^{(n)} - kS + \sum_{i=1}^n a_i f_i(\mathbf{x}, t)$$

กรณีที่เรารู้ค่าพารามิเตอร์ของพลานท์ทั้งหมด

เงื่อนไขที่ทำให้พลานท์จริงสอดคล้องกับโมเดลอ้างอิงคือ $hS + kS = 0$ ซึ่งจากการแก้สมการดิฟเฟอเรนเชียลเราจะได้ว่า s จะมีการลู่เข้าแบบเอกซ์โปเนนเชียล ซึ่งนั่นแสดงว่า e มีการลู่เข้าเช่นเดียวกันนั่นเอง

กฎการปรับตัว พิจารณากฎการควบคุม

$$u = \hat{h}y_r^{(n)} - kS + \sum_{i=1}^n \hat{a}_i f_i(\mathbf{x}, t) \quad \text{---- (2)}$$

นำ (1)-(2) เราจะได้สมการเป็น

$$hy^{(n)} + \sum_{i=1}^n a_i f_i(\mathbf{x}, t) = \hat{h} y_r^{(n)} - kS + \sum_{i=1}^n \hat{a}_i f_i(\mathbf{x}, t) \quad \text{-----}(3)$$

จาก $\tilde{h} = \hat{h} - h$ และ $\tilde{a}_i = \hat{a}_i - a$ แทนค่าลงใน (3) จะได้

$$hS + kS = \tilde{h} y_r^{(n)} + \sum_{i=1}^n \tilde{a}_i f_i(\mathbf{x}, t)$$

แก้สมการจะได้

$$S = \frac{1/h}{s+(k/h)} [\tilde{h} y_r^{(n)} + \sum_{i=1}^n \tilde{a}_i f_i(\mathbf{x}, t)]$$

ความสัมพันธ์ของพารามิเตอร์มีลักษณะเช่นเดียวกับทฤษฎีที่ได้กล่าวมาแล้วในบทหน้านั้นคือ เราจะกฎการปรับตัวมีสมการเป็น

$$\begin{aligned} \dot{\hat{h}} &= -\text{sgn}(h) \gamma S y_r^{(n)} \\ \dot{\hat{a}}_i &= -\text{sgn}(h) \gamma f_i \end{aligned}$$

4.5 เทคนิคที่ใช้ในการประมาณค่าพารามิเตอร์ของระบบ

สำหรับหัวข้อนี้จะเป็นการกล่าวถึงเทคนิค ที่เราใช้ในการประมาณค่าพารามิเตอร์ต่างๆ ของระบบที่เราไม่ทราบค่าจากข้อมูลต่าง ๆ ที่เราทราบ ซึ่งเมื่อเราใช้เทคนิคการประมาณค่าต่าง ๆ เหล่านี้ร่วมกับตัวควบคุมชนิดต่าง ๆ เราก็จะได้ระบบควบคุมแบบปรับจูนตัวเอง(self-tuning) นั้นเอง โดยทั่วไปเทคนิคที่ใช้ในการประมาณค่าพารามิเตอร์ของระบบจะพิจารณาเป็น 2 ประเภทคือ

OFF-LINE มักใช้ในกรณีที่พารามิเตอร์ของระบบเป็นค่าคงที่และเรามีเวลาเพียงพอในการประมาณค่าพารามิเตอร์ก่อนที่จะทำการควบคุม

ON-LINE จะใช้สำหรับกรณีที่พารามิเตอร์ของระบบที่เปลี่ยนแปลงไปตามเวลา (แม้ว่าการเปลี่ยนแปลงจะช้าก็ตาม)

4.5.1 โมเดลของการประมาณค่าพารามิเตอร์ให้เป็นเชิงเส้น

(LINEAR PARAMETERIZATION MODEL)

เราสามารถกล่าวได้ว่า สำหรับระบบใด ๆ สามารถจะพิจารณาให้อยู่ในรูป

$$\mathbf{y}(t) = \mathbf{W}(t)\mathbf{a}$$

โดย $\mathbf{y}(t)$ เป็นเอาต์พุตของระบบ เป็น $n \times 1$ เวกเตอร์

$\mathbf{W}(t)$ เป็นเมตริกซ์สัญญาณ เป็น $n \times m$ เมตริกซ์

$\mathbf{a}$ เป็นตัวแปรที่เราไม่ทราบค่า เป็น $m \times 1$ เวกเตอร์

นั่นคือเราสามารถคำนวณค่าพารามิเตอร์ที่ไม่ทราบค่าของระบบจากข้อมูลที่เราทราบ ซึ่งในที่นี้ได้แก่ $\mathbf{y}$ และ $\mathbf{W}(t)$ นั่นเอง กรณีที่เป็น off-line เราทำการเก็บรวบรวมข้อมูลในช่วงเวลาหนึ่งและทำการคำนวณค่าพารามิเตอร์ที่ต้องการครั้งเดียวและใช้ค่านี้ตลอดไป ส่วนกรณีที่เป็น on-line เราจะมี การคำนวณค่าพารามิเตอร์ใหม่ทุกครั้งที่เกิดของข้อมูล $\mathbf{y}$ และ $\mathbf{W}(t)$ มีการหาค่าใหม่

สำหรับกรณีที่ระบบเป็น SISO เราสามารถทำการ linear parameterization ได้ดัง ต่อไปนี้

พิจารณา $A(s)y = B(s)u$

โดย $A(s) = a_0 + a_1s + \dots + a_{n-1}s^{n-1} + s^n$

$$B(s) = b_0 + b_1s + \dots + b_{n-1}s^{n-1}$$

ทำการหารสมการข้างต้นด้วย monic polynomial ที่เราทราบค่าจะได้เป็น

$$y = \frac{A_o(s)-A(s)}{A_o(s)}y + \frac{B(p)}{A_o(s)}u$$

โดย $A_o = \alpha_0 + \alpha_1s + \dots + \alpha_{n-1}s^{n-1} + s^n$ นั่นคือเราจะได้ความสัมพันธ์เป็น

$$y = \theta^T \mathbf{W}(t)$$

เมื่อ $\theta = [(\alpha_0 - a_0) (\alpha_1 - a_1) \dots (\alpha_{n-1} - a_{n-1}) b_0 \dots b_{n-1}]^T$

$$\mathbf{W} = \left[\begin{array}{cccc} \frac{y}{A_o} & \frac{sy}{A_o} & \dots & \frac{s^{n-1}y}{A_o} & \frac{u}{A_o} & \dots & \frac{s^{n-1}u}{A_o} \end{array} \right]^T$$

4.5.2 การทำนายค่าผิดพลาดโดยใช้วิธีการประมาณ

(PREDICTION-ERROR BASED ESTIMATION METHOD)

กำหนดให้พารามิเตอร์ที่เราประมาณค่าเป็น $\hat{\mathbf{a}}(t)$ กรณีนี้เราจะได้อาต์พุตที่ได้สำหรับพารามิเตอร์ที่เราประมาณเป็น

$$\hat{\mathbf{y}}(t) = \mathbf{W}(t)\hat{\mathbf{a}}(t)$$

กำหนดให้ $\mathbf{e}_1$ เป็นผลแตกต่างระหว่างเอาต์พุตจริงกับเอาต์พุตที่ได้จากการประมาณค่า

(prediction error) นั่นคือเราจะได้ความสัมพันธ์เป็น

$$\begin{aligned}\mathbf{e}_1(t) &= \hat{\mathbf{y}}(t) - \mathbf{y}(t) \\ &= \mathbf{W}\hat{\mathbf{a}} - \mathbf{W}\mathbf{a}; = ; \mathbf{W}\tilde{\mathbf{a}}\end{aligned}$$

$$\text{โดยที่ } \tilde{\mathbf{a}} = \hat{\mathbf{a}} - \mathbf{a}$$

ในการวิเคราะห์ต่อไปนี้จะเราให้ค่าพารามิเตอร์ที่เราทำการพิจารณา ($\mathbf{a}$) มีค่าคงที่สำหรับเทคนิคการประมาณค่าต่างๆที่เป็นที่นิยมใช้กันได้แก่

วิธีที่ 1 การประมาณโดยใช้กราเดียนท์ (The Gradient Estimator)

หลักการของวิธีนี้คือการปรับค่าพารามิเตอร์จะเป็นไปในแนวทางเพื่อจะลดค่าของ prediction error นั้นเอง โดยสมการแสดงความสัมพันธ์ของการปรับค่าพารามิเตอร์จะเป็นดังสมการ

$$\dot{\hat{\mathbf{a}}} = -p_o \frac{\partial [\mathbf{e}_1^T \mathbf{e}_1]}{\partial \hat{\mathbf{a}}}$$

โดยที่ p_o เป็นค่าบวกเรียกว่าค่าเกนของการประมาณ (estimator gain)

$$\dot{\hat{\mathbf{a}}} = -p_o \mathbf{W} \mathbf{e}_1$$

$$\dot{\tilde{\mathbf{a}}} = -p_o \mathbf{W}^T \mathbf{W} \tilde{\mathbf{a}}$$

การพิจารณาคุณสมบัติเกี่ยวกับการลู่เข้าเราอาศัยทฤษฎีของลียาปูนอฟ โดยที่ทำการเลือกฟังก์ชันลียาปูนอฟให้มีลักษณะเป็น

$$V = \tilde{\mathbf{a}}^T \tilde{\mathbf{a}}$$

ทำการหาอนุพันธ์ของ Lyapunov function เราจะได้ว่า

$$\dot{V} = -2p_o \tilde{\mathbf{a}}^T \mathbf{W}^T \mathbf{W} \tilde{\mathbf{a}} \leq 0$$

จากทฤษฎีของลียาปูนอฟและลักษณะของลียาปูนอฟฟังก์ชันที่เราเลือกเราสามารถสรุปได้ว่า

- 1) ค่า $\tilde{\mathbf{a}}$ จะมีลักษณะลู่เข้า
- 2) ค่า $\tilde{\mathbf{a}}$ จะมีลักษณะลดลงเรื่อย ๆ

แต่จากความสัมพันธ์ไม่ได้บอกให้เราทราบถึงขอบเขตล่างของค่า $\tilde{\mathbf{a}}$

วิธีที่ 2 การประมาณด้วยวิธีกำลังสองน้อยที่สุด (The standard least square estimator)

หลักการคือการปรับค่าพารามิเตอร์จะเป็นไปในแนวทางเพื่อที่จะทำให้

$$J = \int_0^t \|y(r) - W(r)\hat{a}(t)\|^2 dr$$

มีค่าน้อยที่สุด จากการคำนวณทางคณิตศาสตร์เราสามารถพิจารณาหาเงื่อนไขที่ทำให้

J น้อยที่สุดคือ $\frac{\partial J}{\partial \hat{a}} = 0$ จะได้

$$\left[\int_0^t W^T W dr \right] \hat{a}(t) = \int_0^t W^T y dr$$

กำหนด $P(t) = \left[\int_0^t W^T(r) W(r) dr \right]^{-1}$

และ $\frac{d}{dt}[P^{-1}(t)] = W^T(t)W(t)$

ทำการหาอนุพันธ์และจัดรูปจะได้

$$\dot{\hat{a}} = -P(t)W^T e_1 \quad \text{----- (2)}$$

โดย $P(t)$ เป็นเมตริกซ์เกนของการประมาณ(estimator gain matrix)

พิจารณาความสัมพันธ์

$$\frac{d}{dt}[PP^{-1}] = \dot{P}P^{-1} + P\frac{d}{dt}[P^{-1}] = 0$$

จัดรูปจะได้ว่า

$$\dot{P} = -PW^TWP \quad \text{----- (3)}$$

เมื่อพิจารณาการสู่เข้า

$$P^{-1}(t) = P^{-1}(0) + \int_0^t W^T(r)W(r)dr$$

$$\frac{d}{dt}[P^{-1}(t)\tilde{a}(t)] = 0$$

ดังนั้น

$$\tilde{a}(t) = P(t)P^{-1}(0)\tilde{a}(0) \quad \text{----- (4)}$$

หมายเหตุ จากสมการที่ (4) เราจะเห็นผลของเงื่อนไขเริ่มต้น นั่นคือ

หาก $\tilde{a}(0)$ มีค่าน้อย จะทำให้ $\tilde{a}(t)$ มีค่าน้อย

หาก $\tilde{a}(0)$ มีค่ามาก จะทำให้ $\tilde{a}(t)$ มีค่ามาก

4.6 การควบคุมการปรับตัวแบบคอมโพสิท (COMPOSITE ADAPTIVE CONTROL)

จากหัวข้อข้างต้นเราจะเห็นได้ว่ากรณีใช้ตัวควบคุมแบบโมเดลอ้างอิงนั้น เราจะหากฎการปรับตัวโดยพิจารณาจาก ความผิดพลาดในการติดตามระบบ ส่วนกรณีของ การประมาณค่าพารามิเตอร์ เราประมาณค่าพารามิเตอร์จากค่าผิดพลาดจากการทำนาย ทำให้เกิดแนวคิดในการหากฎการปรับตัว โดยอาศัยข้อมูล 2 ชนิดคือค่าผิดพลาดจากการติดตามระบบ (tracking error) และจากค่าผิดพลาดจากการทำนาย (prediction error) ซึ่งตัวควบคุมที่อาศัยหลักการเช่นนี้เราเรียกว่าตัวควบคุมการปรับตัวแบบคอมโพสิท (composite adaptation controller)

สำหรับหัวข้อนี้เราจะศึกษากรณีที่ระบบของเราเป็นอันดับหนึ่งโดยมีความสัมพันธ์ของระบบเป็น

$$\dot{y} = -a_p y + b_p u$$

พิจารณาส่วนที่ใช้หลักการของ MRAC จากหัวข้อข้างต้นเราสามารถสรุปได้ว่า

ตัวควบคุม $u = \mathbf{v}^T \mathbf{a}$

กฎการปรับตัว $\dot{\hat{\mathbf{a}}}^* = -\text{sgn}(b_p) \gamma \mathbf{v} e$

พิจารณาส่วนของ ค่าผิดพลาดจากการทำนายในที่นี้ เราจะใช้เทคนิควิธีการประมาณแบบกราเดียนท์ ในการประมาณค่าของ a_r และ a_y ก่อนอื่นเราต้องทำการจัดพารามิเตอร์ของ พลานท์ ของเรา ก่อน โดยพิจารณาจาก

$$\dot{y} + a_m y = -(a_p - a_m)y + b_p u$$

$$\text{ดังนั้น } u = \frac{1}{b_p}(s + a_m)y + \frac{a_p - a_m}{b_p}y$$

$$= a_r \frac{s + a_m}{b_m}y + a_y y$$

คูณทั้งสองข้างของสมการด้วย $\frac{1}{s + a_m}$ เราจะได้สมการเป็น

$$\frac{u}{s + a_m} = a_r \frac{y}{b_m} + a_y \frac{y}{s + a_m}$$

กำหนดให้ $u_1 = \mathbf{w} \mathbf{a}$

โดยที่ $u_1 = \frac{u}{s + a_m}, \quad \mathbf{a} = [a_r \ a_y]^T, \quad \mathbf{w} = [\frac{y}{b_m} \ \frac{y}{s + a_m}]^T$

จากอัลกอริทึมของการประมาณแบบ กราเดียนท์เราจะได้ว่า

$$\dot{\hat{\mathbf{a}}}^* = -\mathbf{P} \mathbf{w} e_1$$

โดย e_1 เป็นค่าผิดพลาดจากการทำนายค่าผิดพลาดจากการทำนาย ของ u_1 สำหรับกฎการปรับตัวแบบคอมโพสิท (composite adaptation laws) จะมีองค์ประกอบต่าง ๆ เป็นดังนี้

กฎการเปลี่ยนแปลง $\dot{\hat{\mathbf{a}}}^* = -\mathbf{P}[\text{sgn}(b_p)\mathbf{v}e + \alpha(t)\mathbf{w}e_1]$

ตัวควบคุม $u = \mathbf{v}^T \mathbf{a}$

ตัวควบคุมแบบคอมโพสิตนี้ จะทำให้ผลตอบสนองของการควบคุม มีการปรับปรุงขึ้นจากตัวควบคุมเดิมที่ใช้ลักษณะการควบคุมแบบ MRAC เพียงอย่างเดียว

ส่วนวิธีการประมาณค่าพารามิเตอร์อาจจะใช้วิธีการประมาณค่าด้วยวิธีกำลังสองน้อยที่สุดซึ่งจะเพิ่มเติมในส่วนของการปรับค่าเกณฑ์ให้ดียิ่งขึ้น การเลือกจะใช้วิธีใดขึ้นอยู่กับดุลพินิจของผู้ออกแบบจะต้องพิจารณาตามความเหมาะสมกับระบบนั้นๆ



บทที่ 5

การคำนวณและการสร้าง

5.1 การสร้างแอลกอริทึมสำหรับการวิเคราะห์เฟสเพลน

เทคนิคการวิเคราะห์แบบระบบในกรณีนี้ จะใช้กับระบบอันดับสอง ที่มีตัวแปรสเกลเพียง 2 ตัว(ดังที่เคยอธิบายไว้ในบทที่ 1)

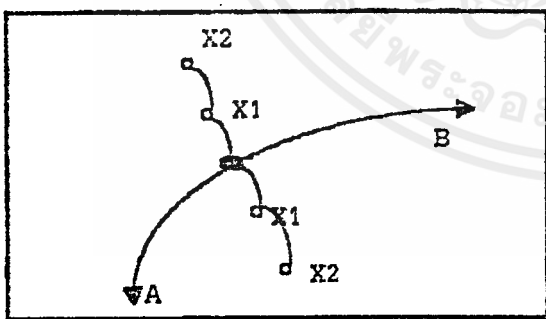
เมื่อเราพิจารณาสมการอนุพันธ์ 2 สมการ

$$\dot{x} = f(x, y) \quad \text{----- (1)}$$

$$\dot{y} = g(x, y) \quad \text{----- (2)}$$

เมื่อเรากำหนดค่าเงื่อนไขเริ่มต้นให้กับสมการทั้งสอง แล้วแทนค่าลงในสมการทั้งสอง เราจะได้ค่า $\dot{x}$ และ $\dot{y}$ ซึ่งค่าทั้งสองนี้จะสัมพันธ์กับความชันของกราฟ คือค่าของ $dy/dx = \dot{y}/\dot{x}$ ค่าความชันนี้จะใช้ในการลากเส้นกราฟไปที่จุด x และ y จุดต่อไป แล้วใช้ค่า x และ y ที่ได้หาค่าความชันอีกครั้ง แล้วทำการลากเส้นกราฟต่อไปเรื่อย ๆ

สำหรับการเขียนโปรแกรมกำหนดค่าเริ่มต้นขึ้นมาก่อนหนึ่งจุดแล้วลากกราฟออกไปสองทิศทาง จนจนขอบจอจนได้กราฟคำตอบของสมการหนึ่งเส้น เปลี่ยนค่าเงื่อนไขเริ่มต้นนี้ไปที่จุดต่อไป แต่การขยับจุด ๆ นี้จะต้องลากในทิศทางที่ตั้งฉากกับกราฟ เพราะจะทำให้ระยะระหว่างเส้นกราฟเกิดความสมดุล การขยับจุดเงื่อนไขเริ่มต้นนี้จะทำทั้งสองทิศทางดังรูปเฟสเพลนนี้จะต้องลากเส้นกราฟให้เต็ม



ระนาบเกี่ยวกับตรงจุดนี้ก็ใช้เทคนิคบางอย่างกล่าวคือจากรูป แนวกราฟ A,B เกิดจากการลากกราฟคำตอบของสมการ โดยมีจุด (x_1, y_1) เป็นจุดเริ่มต้นจุดเริ่มต้นนี้ จะถูกปรับไปอยู่ที่จุด (x_2, y_2) เพื่อทำการลาก กราฟ แล้วรับต่อไปอีกไปอยู่ที่จุด (x_3, y_3)

แล้วทำเช่นนี้ไปเรื่อย ๆ (ในโปรแกรม จำนวนครั้งในการขยับจุดนี้สามารถเช็คค่าได้โดยเลือกเมนู num line) จุดเงื่อนไขเริ่มต้น (x_1, y_1) เมื่อขยับไปทางด้านหนึ่งจนครบแล้วก็จะขยับไปทางอีกด้านหนึ่ง

สำหรับโปรแกรมนี้จะมีการเช็คค่า (x_1, y_1) ไว้ 4 ตำแหน่งในระนาบคือจุดกึ่งกลางในแต่ละ

ควอดแรนท์ เพื่อให้แนวกราฟสมมาตรกันทั้งระนาบ

การเซตพารามิเตอร์สำหรับโปรแกรมเฟสเพลนมีดังนี้คือ

1) numline หมายถึงจำนวนครั้งที่จะลากเส้นกราฟ ถ้าค่า numline มากขึ้น จะมีกราฟจำนวนมากขึ้นที่อยู่ในระนาบ การเซตค่านี้ขึ้นอยู่กับ ความเหมาะสมของลักษณะเฟสเพลน หรือรายละเอียดตามที่เราต้องการ

2) numpoint เป็นตัวกำหนดความยาวของกราฟแต่ละเส้น ถ้าค่าของ numpoint น้อยจนเกินไป จะทำให้เส้นกราฟสั้นลง แต่ถ้าถูกกำหนดมากเกินไป ตัวโปรแกรมก็จะลากจุดไปเรื่อย ๆ ที่ละจุด จนครบจำนวน numpoint แต่เมื่อลากจนจนจบแล้วก็จะหยุดลากโดยอัตโนมัติ เพื่อรักษาเวลาในการทำงานในกรณีผู้ใช้เซตค่านี้มากเกินไป

3) range x และ range y เป็นตัวกำหนดสเกลของจอภาพโดยที่จะต้องใส่ค่าที่เป็นบวกเท่านั้น เช่นกำหนดให้

$$\text{range } x = 5$$

$$\text{และ } \text{range } y = 3$$

หมายถึงกรอบของเฟสเพลนจะอยู่ในช่วง $x = -5$ จนถึง $x = 5$

$$y = -3 \text{ จนถึง } y = 3$$

4) การกำหนดสมการของระบบให้อ้างอิงถึงตัวแปรสเตตเพียงสองตัวคือ x และ y เท่านั้น

หมายเหตุ เนื่องจากกราฟที่ได้จากการพล็อตของโปรแกรมนี้ไม่ได้คำนวณบน time domain เราจึงไม่สามารถดูทิศทางของกราฟที่สัมพันธ์กับเวลาได้ เหตุผลไม่คำนวณเวลาดังนั้นก็เพราะใช้เวลาในการคำนวณมากขึ้น และพฤติกรรมของระบบที่แสดงโดยเฟสเพลนก็เป็นเพียงพฤติกรรมหยาบ ๆ ไม่ได้ต้องการความละเอียดมากนักเกินไป

5.2 วิธีการเชิงเลขสำหรับการหาคำตอบของสมการอนุพันธ์

หลักการเชิงตัวเลขที่ใช้หาคำตอบของสมการอนุพันธ์ที่ใช้คือวิธี fourth order runge gutta method ซึ่งรูปแบบของสมการที่ใช้ในการคำนวณ(สำหรับระบบอันดับสาม) เป็นดังนี้

สมมติ

$$\dot{x} = f(x, y, z)$$

$$\dot{y} = g(x, y, z)$$

$$\dot{z} = h(x, y, z)$$

เมื่อเราทราบจุดเริ่มต้น(x,y,z) เราต้องการหาค่า $\Delta x, \Delta y$ และ Δz ซึ่งเป็นค่าที่เลื่อนสเตปไปยังจุดต่อไป เราจะสมมติให้ $\Delta x, \Delta y$ และ Δz คือ L,M และ N ตามลำดับ สำหรับวิธี fourth order runge gutta การหาค่า K,L และ M จะทำการหาสี่ครั้ง แล้วนำค่าที่ได้ทั้งสี่มารวมกันตามอัตราส่วนต่อไปนี้

$$k_1 = \Delta t * f(T, x_1, x_2, x_3)$$

$$l_1 = \Delta t * g(T, x_1, x_2, x_3)$$

$$m_1 = \Delta t * h(T, x_1, x_2, x_3)$$

$$k_2 = \Delta t * f(T + 0.5\Delta t, x_1 + 0.5k_1, x_2 + 0.5l_1, x_3 + 0.5m_1)$$

$$l_2 = \Delta t * g(T + 0.5\Delta t, x_1 + 0.5k_1, x_2 + 0.5l_1, x_3 + 0.5m_1)$$

$$m_2 = \Delta t * h(T + 0.5\Delta t, x_1 + 0.5k_1, x_2 + 0.5l_1, x_3 + 0.5m_1)$$

$$k_3 = \Delta t * f(T + 0.5\Delta t, x_1 + 0.5k_2, x_2 + 0.5l_2, x_3 + 0.5m_2)$$

$$l_3 = \Delta t * g(T + 0.5\Delta t, x_1 + 0.5k_2, x_2 + 0.5l_2, x_3 + 0.5m_2)$$

$$m_3 = \Delta t * h(T + 0.5\Delta t, x_1 + 0.5k_2, x_2 + 0.5l_2, x_3 + 0.5m_2)$$

$$k_4 = \Delta t * f(T + \Delta t, x_1 + k_3, x_2 + l_3, x_3 + m_3)$$

$$l_4 = \Delta t * g(T + \Delta t, x_1 + k_3, x_2 + l_3, x_3 + m_3)$$

$$m_4 = \Delta t * h(T + \Delta t, x_1 + k_3, x_2 + l_3, x_3 + m_3)$$

อัตราส่วนของน้ำหนักที่ใช้ในการคำนวณค่าของสเตปถัดไปเป็นดังสมการ

$$x_1 = x_1 + (k_1 + 2 * k_2 + 2 * k_3 + k_4)/6$$

$$x_2 = x_2 + (l_1 + 2 * l_2 + 2 * l_3 + l_4)/6$$

$$x_3 = x_3 + (m_1 + 2 * m_2 + 2 * m_3 + m_4)/6$$

5.3 การแปลงดิสครีตฟูเรียร์(Discrete Fourier Transform)

เนื่องจากการแปลงฟูเรียร์เป็นเครื่องมือที่มีความสำคัญมากในการวิเคราะห์ระบบต่าง ๆ ฉะนั้นการมีวิธีการแปลงที่มีประสิทธิภาพและให้ความถูกต้องสูงเป็นสิ่งที่มีความจำเป็น โดยเฉพาะอย่างยิ่งเนื่องจากนิยามของการแปลงจะประกอบด้วยเทอมที่เป็นการอินทิเกรต ซึ่งจะทำให้เกิดปัญหามากเมื่อเราประยุกต์ใช้กับคอมพิวเตอร์ สำหรับวิธีการแปลงที่ใช้ในปริยญาพจน์นี้เป็นวิธีที่เรียกว่า DISCRETE FOURIER TRANSFORM

เพื่อให้การอธิบายเข้าใจง่าย โดยผู้อ่านไม่จำเป็นต้องมีความรู้ทางด้าน Digital Signal Processing มากเกินไป การอธิบายหลักการจะหลีกเลี่ยงการคำนวณทางคณิตศาสตร์ที่ซับซ้อน แต่จะกล่าวพอให้เห็นแนวทางในการคำนวณ ดังนี้

ให้ฟังก์ชันของสัญญาณต่อเนื่องมีฮาร์โมนิกเป็นดังนี้

$$f(t) = A_0 + \sum_{n=1}^{\infty} (A_n \cos(n\omega_0 t) + B_n \sin(n\omega_0 t))$$

$$A_0 = \frac{1}{T} \int_0^T f(t) dt \quad \text{-----(1)}$$

$$A_n = \frac{2}{T} \int_0^T f(t) \cos(n\omega_0 t) dt \quad \text{-----(2)}$$

$$B_n = \frac{2}{T} \int_0^T f(t) \sin(n\omega_0 t) dt \quad \text{-----(3)}$$

พิจารณาฟังก์ชันต่อเนื่องและเป็นคาบ $x(t)$ จากการหาค่าสเปกตรัมของสัญญาณนี้ ผลลัพธ์ที่ได้จะมีลักษณะที่เป็นดิสครีต หากเราทำการสุ่มสัญญาณต่อเนื่องที่เป็นคาบนี้และทำการหาสเปกตรัม เราจะได้ว่าสเปกตรัมของมันจะมีคุณสมบัติที่เป็นดิสครีต (มาจากสมมติฐานที่เราวิเคราะห์สัญญาณที่เป็นคาบ) นอกจากนี้สเปกตรัมที่ได้จะมีลักษณะที่เป็นคาบด้วย(มาจากทฤษฎีการสุ่มตัวอย่าง) กำหนดให้ลำดับของข้อมูลที่เราสุ่มมาเมื่ออยู่ N ลำดับช่วงเวลาที่ใช้ในคือ T ดังนั้นเราจะรู้ค่าความถี่ของการสุ่มคือ

$$\omega_s = \frac{N}{T} \quad \text{----- (4)}$$

$$\text{และความถี่ของสัญญาณ } \omega_0 = \frac{1}{T}$$

จากทฤษฎีการสุ่มสัญญาณ ทำให้เราสามารถจะรู้ค่าของความถี่ที่เกิดสเปกตรัมได้จากข้อมูลต่าง ๆ ที่อยู่ในโดเมนของเวลา คือข้อมูลจากสมการ (4) นั้นเอง

ทำการแปลงสมการ (1)-(3) เป็นดิสครีต

$$A(k) = \frac{2}{N} \sum_{n=0}^{N-1} f(n) \Delta T \cos(k\omega_0 t) \quad , t = n\Delta T \quad , \Delta T = \frac{T}{N}$$

$$= \frac{2}{N} \sum_{n=0}^{N-1} f(n) \cos\left(\frac{kn}{N}\right)$$

$$B(k) = \frac{2}{N} \sum_{n=0}^{N-1} f(n) \sin\left(\frac{nk}{N}\right)$$

จะเห็นได้ว่าการเขียนโปรแกรมคอมพิวเตอร์ เราจะสามารถรู้ค่าต่าง ๆ ได้อย่างถูกต้อง
นั่นเอง

กล่าวโดยสรุป จากการที่ทำการสุ่มฟังก์ชันเป็นคาบและต่อเนื่องออกเป็นฟังก์ชันดิสครีต ในที่
นี้สมมติให้มีสัญญาณที่ได้จากการสุ่มจำนวน n ลำดับ เราจะพบว่าสเปกตรัมของสัญญาณต่อเนื่อง
กับสเปกตรัมของสัญญาณที่ถูกสุ่มตัวอย่าง ในช่วงที่ $0 < k < n-1$ จะเหมือนกันทุกประการแต่ผลจากการ
สุ่มตัวอย่างทำให้เกิดข้อแตกต่างระหว่างสเปกตรัมของสัญญาณทั้งสอง กล่าวคือ สเปกตรัมของสัญญาณ
ดิสครีตจะเป็นคาบด้วยลักษณะเช่นนี้เหมาะสมมากสำหรับการคำนวณโดยคอมพิวเตอร์ เนื่องจาก
จากเราสามารถรู้เวลาที่แน่นอนได้นั่นเอง

บทที่ 6

การใช้งานโปรแกรม simulation

โปรแกรม "MAINP.C" นี้เป็นโปรแกรมที่ได้เขียนขึ้นเพื่อเป็นเครื่องมือที่ใช้ช่วยในการวิเคราะห์ระบบที่มีพฤติกรรมเป็น time-variant nonlinear system (ส่วนในกรณีเฉพาะคือ time-invariant หรือ linear system ก็สามารถใช้ได้เช่นกัน) ในการใช้งานจะมีวิธีใช้ดังต่อไปนี้

- 1) การเริ่มต้นโปรแกรม เราจะใช้คำสั่ง `simulate`
- 2) การกำหนดเงื่อนไขและค่าเริ่มต้นต่าง ๆ ในการคำนวณ เรามีคำสั่งต่าง ๆ ดังต่อไปนี้

`width` กำหนดจำนวนจุดของข้อมูลที่จะใช้ในการพิจารณา

`time` กำหนดขอบเขตของเวลาที่เรากำลังพิจารณา

`precision` กำหนดความละเอียดที่ใช้ในการคำนวณ

`initial` กำหนดค่าเริ่มต้นของตัวแปรที่เราใช้ในการคำนวณ

- 3) การกำหนดตัวแปรที่ใช้ในการคำนวณ ใช้คำสั่ง

`variable ucee`

คำสั่งนี้จะบอกตัวโปรแกรมว่าตัวแปร `u` และ `c` มีชนิดของตัวแปรเป็น `expression`

`variable xsys`

คำสั่งนี้จะบอกตัวโปรแกรมว่า ตัวแปร `x` และ `y` มีชนิดของตัวแปรเป็น `state`

หมายเหตุ ชนิดของตัวแปรนี้จะบอกให้เห็นถึงความแตกต่างระหว่างสมการที่ใช้ในการกำหนดค่าและสมการอนุพันธ์ โดยชนิดของตัวแปร `s` และ `e` นี้จะบอกให้ทราบว่าตัวแปรนั้น ๆ ถูกกำหนดค่าโดยสมการในรูปแบบใด

- 4) การกำหนดตัวแปรที่ต้องการบันทึกค่า ณ เวลาต่างๆ

`record` ใช้กำหนดตัวแปรที่เราต้องการจะบันทึกค่าที่ได้จากการประมวลผล ณ เวลา

ต่างๆ (เป็นการกำหนดเนื้อที่ในหน่วยความจำให้กับตัวแปรนั้น ๆ)

- 5) การกำหนดค่าคงที่ ใช้คำสั่ง

`define` ใช้กำหนดค่าคงที่ให้กับตัวแปรต่าง ๆ

- 6) การกำหนดสมการของระบบ

สมการของระบบแบ่งออกเป็น 2 รูปแบบคือ

6.1 สมการที่กำหนดค่าตัวแปรที่เป็น expression เช่น กำหนดให้สัญญาณอินพุต u อยู่ในรูปฟังก์ชันของเวลาดังนี้

$$u = 1 + 2 \sin(3 * t + 6)$$

หรือกำหนดให้สัญญาณควบคุม c สัมพันธ์กับสัญญาณ error e ดังนี้

$$c = \text{sign}(1 - \text{abs}(e)) * e + \text{sign}(\text{abs}(e) - 1)$$

ตัวแปรที่ถูกกำหนดค่าลักษณะนี้จะต้องทำการประกาศให้มีชนิดของข้อมูล เป็น 'e'

6.2 สมการแสดงความสัมพันธ์ของตัวแปรเป็นสมการดิฟเฟอเรนเชียลเช่น

$$\dot{x} = 3x + 4y + u$$

$$\dot{y} = \sin(x) + y$$

สำหรับสมการอนุพันธ์จะเขียนในรูปเดียวกับสมการที่ใช้ในการกำหนดค่า(กล่าวคือไม่ต้องใช้เครื่องหมายอนุพันธ์) แต่ตัวแปรของระบบที่ถูกกำหนดค่าโดยสมการอนุพันธ์จะถือว่าเป็นตัวแปร internal state ตัวแปรเหล่านี้จะต้องถูกประกาศให้มีชนิดของข้อมูลเป็น s

7) คำสั่งเกี่ยวกับการประมวลผล

run ทำการประมวลผลระบบที่เราได้ทำการป้อนไปแล้วดังข้อ (1)-(6)

plot ทำการพล็อตกราฟค่าของตัวแปรที่เราต้องการจะพิจารณาเทียบกับเวลาโดยประ

กอบด้วย plot จะทำการพล็อตตัวแปรทุกตัวที่เรากำหนดโดยใช้คำสั่ง record

plot ชื่อไฟล์ จะทำการพล็อตตัวแปรที่อ่านมาจากชื่อไฟล์ที่กำหนด

plot ##ชื่อตัวแปร จะสามารถเลือกพล็อตเฉพาะตัวแปรที่เราต้องการได้

8) คำสั่งแสดงผลการคำนวณ

showp จะแสดงค่าของพารามิเตอร์ต่าง ๆ ที่ผู้ใช้ทำการป้อนในตอนต้น

showd จะแสดงค่าของข้อมูลที่ได้จากการคำนวณ ณ เวลาต่างๆ

showt จะแสดงค่าของเวลาที่ใช้ในการประมวลผล

9) คำสั่งที่ใช้ในการกำหนดค่าของตัวแปรจะมีคำสั่งต่าง ๆ ดังต่อไปนี้

assign เป็นการกำหนดค่าของตัวแปรที่อยู่ทางด้านขวาของเครื่องหมายเท่ากับให้ กับ

ตัวแปรที่อยู่ทางซ้ายมือ assign equ เป็นการกำหนดค่าของความสัมพันธ์

ทางคณิตศาสตร์ระหว่างตัวแปรต่าง ๆ ให้กับตัวแปร

10) คำสั่งที่เป็นการคำนวณทางคณิตศาสตร์

FFT จะทำการหาฟูเรียร์ทรานสฟอร์มของสัญญาณต่าง ๆ ที่เราต้องการ

IFFT จะทำการหาอินเวอร์สฟูเรียร์ทรานสฟอร์มของสัญญาณต่าง ๆ ที่เราต้องการ



บทที่ 7

ตัวอย่างโปรแกรมและผลการทดลอง

7.1 บทนำ

สำหรับบทนี้จะเป็นการแสดงการประยุกต์ใช้โปรแกรมนี้เพื่อทำการวิเคราะห์ปัญหาต่าง ๆ ทางวิศวกรรม เช่น ประยุกต์ใช้ในงานวิเคราะห์และออกแบบระบบควบคุม, ประยุกต์ใช้ในการวิเคราะห์วงจรไฟฟ้า, ประยุกต์ใช้ในทางไฟฟ้ากำลัง ตลอดจนประยุกต์ในทางวิศวกรรมสื่อสาร เป็นต้น

ลำดับของการเสนอจะประกอบไปด้วย

- 1) ส่วนนำเกี่ยวกับโปรแกรม ว่าโปรแกรมนี้เขียนขึ้นเพื่อวัตถุประสงค์อะไร
- 2) ส่วนตัวอย่างโปรแกรมที่เขียนขึ้น
- 3) ผลการรันโปรแกรม(เป็นรูปภาพ)

โดยทั่วไป หากเราทราบความสัมพันธ์ของตัวแปรที่เป็นฟังก์ชันของเวลาเราสามารถจะทำการพล็อตกราฟของตัวแปรนั้น โดยเราทำการกำหนดตัวแปรนั้นเป็น record ซึ่งโดยวิธีนี้เราสามารถทำการพล็อตค่าของตัวแปรหลายตัวบนระนาบเดียวกันได้

หากกำหนดสัญญาณที่มีรูปแบบต่างๆ มาให้และเราต้องการจะเขียนสัญญาณนี้ในรูปแบบสมการทางคณิตศาสตร์ เราสามารถที่จะตรวจสอบโดยการใช้โปรแกรมพล็อตกราฟนี้ได้ว่าสมการทางคณิตศาสตร์ที่เราคิดนั้นถูกต้องหรือไม่ โดยวิธีนี้เราสามารถนำไปประยุกต์ใช้งานได้หลายด้าน เช่น หากเราต้องการวิเคราะห์วงจรไฟฟ้าวงจรหนึ่ง โดยเราต้องการพิจารณาผลตอบสนองของวงจรนี้ที่มีต่อสัญญาณอินพุตรูปแบบต่างๆ ในการคำนวณทางทฤษฎีจะต้องมีการแก้สมการดิฟเฟอเรนเชียล(ถ้าระบบมีอันดับเกินหนึ่ง) ซึ่งหากสัญญาณที่เป็นอินพุตซับซ้อนขึ้นการคำนวณด้วยมือจะไม่มีทางเป็นไปได้เลยในทางปฏิบัติ เป็นต้น

นอกจากนี้เรายังสามารถบันทึกผลการคำนวณลงไฟล์ได้ด้วย (โดยปกติจะทำการบันทึกลงในไฟล์ตามชื่อที่ตั้ง default ไว้ ถ้าผู้ใช้ไม่ได้ระบุชื่อไฟล์ที่จะบันทึก) ตัวอย่างโปรแกรมการบันทึกลงไฟล์ซึ่งใช้คำสั่ง SAVE DATA และรูปแบบการบันทึกข้อมูลลงไฟล์มีในหน้าถัดไป)

7.2 ตัวอย่างการใช้โปรแกรมในการวิเคราะห์ระบบต่างๆ

```
SIMULATE
#
# File Name : File.1
# Function : Test File Writing
#
PATH B:
FILE DATA.XXX
WIDTH 50
PRECISION 50
TIME 6.28
RECORD abcd
EQUATION
A=SIN(1*t)
B=SIN(2*t)
C=SIN(3*t)
D=SIN(4*t)
ENDEQUATION
RUN
SAVE DATA
SHOWD
ENDPROGRAM
```



FILE NAME : B:data.xxx.
 EXP. DATE : 21:3:1993
 EXP. TIME : 23:19:4
 Time-Range: 6.3
 Precision : 50
 Initial : All states are initialized by zero

System var: abcd

Equation type

Expression: abcd

Diff. equ.:

System Eq.:

(1) : a=sin(1*t)

(2) : b=sin(2*t)

(3) : c=sin(3*t)

(4) : d=sin(4*t)

Record var: abcd

VARIABLE : a

SIZE : 50

aDATA

0	0	0.1252700	0.2485665	0.3679469	0.4815304
0.5875275	0.6842684	0.7702289	0.8440547	0.9045827	0.9508594
0.9821557	0.9979784	0.9980784	0.9824540	0.9513514	0.9052606
0.8449077	0.7712436	0.6854288	0.5888153	0.4829252	0.3694268
0.2501082	0.1268493	0.0015918	-0.1236907	-0.2470245	-0.3664665
-0.4801350	-0.5862390	-0.6831072	-0.7692130	-0.8432002	-0.9039031
-0.9503654	-0.9818550	-0.9978760	-0.9981758	-0.9827497	-0.9518408
-0.9059361	-0.8457587	-0.7722567	-0.6865880	-0.5901026	-0.4843202
-0.3709076	-0.2516515				

VARIABLE : b

SIZE : 50

bDATA

0	0	0.2485665	0.4815304	0.6842685	0.8440548
0.9508595	0.9979784	0.9824540	0.9052606	0.7712438	0.5888157
0.3694275	0.1268502	-0.1236895	-0.3664652	-0.5862377	-0.7692121
-0.9039027	-0.9818550	-0.9981758	-0.9518407	-0.8457581	-0.6865870
-0.4843185	-0.2516491	-0.0031837	0.2454817	0.4787380	0.6819438
0.8423436	0.9498692	0.9977711	0.9830426	0.9066083	0.7732661
0.5913858	0.3723841	0.1300077	-0.1205292	-0.3635005	-0.5836548
-0.7671729	-0.9025356	-0.9812458	-0.9983631	-0.9528130	-0.8474545
-0.6889013	-0.4871056				

VARIABLE : c

SIZE : 50

cDATA

0	0	0.3679469	0.6842685	0.9045828	0.9979784
0.9513513	0.7712438	0.4829257	0.1268500	-0.2470233	-0.5862377
-0.8431994	-0.9818549	-0.9827499	-0.8457589	-0.5901026	-0.2516505
0.1221093	0.4787363	0.7681940	0.9498688	0.9982706	0.9066087
0.6877435	0.3723832	0.0047760	-0.3635022	-0.6807786	-0.9025372
-0.9976637	-0.9528115	-0.7742735	-0.4871006	-0.1315845	0.2423937
0.5823626	0.8406225	0.9809380	0.9836220	0.8482978	0.5939524
0.2562718	-0.1173653	-0.4745361	-0.7651251	-0.9483629	-0.9985403
-0.9086157	-0.6912076				

VARIABLE : d

SIZE : 50

dDATA

0	0	0.4815304	0.8440548	0.9979784	0.9052605
0.5888155	0.1268500	-0.3664654	-0.7692121	-0.9818549	-0.9518410
-0.6865880	-0.2516510	0.2454794	0.6819417	0.9498682	0.9830431
0.7732673	0.3723849	-0.1205292	-0.5836555	-0.9025363	-0.9983630
-0.8474525	-0.4871014	-0.0063673	0.4759405	0.8406236	0.9975536
0.9079468	0.5939485	0.1331609	-0.3605354	-0.7651288	-0.9806277
-0.9537737	-0.6912035	-0.2578087	0.2393010	0.6772697	0.9478576
0.9841913	0.7772914	0.3782912	-0.1142002	-0.5784680	-0.8997729
-0.9987073	-0.8508206				

ตัวอย่างการใช้งานพื้นฐาน

- การเขียนโปรแกรมเพื่อพล็อตกราฟรูปแบบต่างๆ
- การเขียนโปรแกรมเพื่อทำอินทีเกรต
- การเขียนโปรแกรมแปลงฟูเรียร์

ตัวอย่างการประยุกต์ใช้งานในด้านระบบควบคุม

- การเขียนโปรแกรมเปรียบเทียบกับกราฟในบทที่ 4 ในหนังสือ
• MODERN CONTROL ENGINEERING ของ OKATA
- การเขียนโปรแกรมวิเคราะห์ระบบป้อนกลับ
- การเขียนโปรแกรมเพื่อช่วยวิเคราะห์การออกแบบตัวควบคุม
Adaptive control สำหรับระบบที่ไม่เป็นเชิงเส้นโดยใช้

ตัวอย่างการประยุกต์ใช้งานในด้านอื่นๆ

- การใช้งานด้านการวิเคราะห์วงจรไฟฟ้า
 1. วงจรโจเซฟสันที่ปราศจากแหล่งจ่าย (A source free Josephson junction circuit)
 2. แวนเดอร์โพลอสซิลเลเตอร์ (Van der pole Oscillator)
- การประยุกต์ใช้งานในด้านอิเล็กทรอนิกส์กำลัง
 1. อินเวอร์เตอร์หนึ่งเฟส
 2. อินเวอร์เตอร์สามเฟส
- การประยุกต์ใช้งานในการสื่อสาร
การมอดูเลตแบบต่างๆ
 1. Amplitude Modulation
 2. Frequency Modulation
 3. Frequency Shift Keying
 4. Phase Shift Keying

ตัวอย่างการเขียนโปรแกรมเพื่อพล็อตกราฟรูปแบบต่างๆ

เป็นตัวอย่างการใช้ฟังก์ชันทางคณิตศาสตร์และรูปแบบการเขียนโปรแกรม
ซึ่งแต่ละกราฟจะพล็อตที่ระดับต่างกันโดยเรียกใช้ฟังก์ชัน Shift



```

simulate
// This Reserved Word Will Always be Used
// in The First Line of Program.

# Program for Plot Graph Example.
# This Text File Will be Executed On Our Simulation Program.
# The Syntact Description Are Included in Our Project (Appendix).
# If Syntact Error, This File Cannot be Run in our Project.
# Please Check All Commands Allowed in Writing The Program.
# We Write Many Demonstration Programs To Guide The Application of
# The Simulation Program.
# File Name : GRAPH.1
# Date      : 18 March 1993
# REMARK    : "#" Used For Comment The Whole Line.
#           : "/" Used For Comment The Whole Line or Part of Line.
# The Program Syntact Not Depends on Character Size.

1 :freemem // Free Memory That Allocated Before
           // And Set All Parameter To Default Value.
2 :showt 2 // Reset Clock To Check Time Consuming.
           // (Start Timing)
3 :width 600 // Set Numbar of Data to 600.
           // "width"=600 are Full Screen Display.
           // If "width" is Changed From Previous Value,
           // The Memory Allocated For Record variables Will
           // be Modified each Whose Sizes Equal to The New
           // "width".
4 :time 6.28 // Set Time Range of Data That be Simulated.
           // 6.28=2*PHI
5 :precision 1 // This Precision Value Cannot Effect The System
           // Equation That Don't Have Diffential Equation.
           // This Command Line Can Be Removed.
6 :record abcdefgh // Allcate Memory For Record Variables.
           // If Declare Much More Variables, Some Variable
           // May not Succeed in Allocation.

define Shift=3 // Warning : The Word "Shift" MUST Not Exist in
           // Other Places Except Equation, Because "IT"
           // Will Be Replaced by 3 Immediately.
           // User Will Define The contant Carefully.

7 :equation // The Variable Allowed, Are a,b,c,...,z Only.
8 :a=sin(t)+Shift // Use "Shift" to Change The Level of Graph.
9 :b=cos(2*t)+Shift
10:c=exp(-t*t)+Shift
11:d=Shift
12:e=sin(4*t+1)-Shift
13:f=ln(t+2)-Shift
14:g=log10(5*t+2)-Shift
15:h=-Shift
16:endequation

17:run // Calculate All Equations Consecutively From
       // time=0 to time=Time Range.

18:plot // If no Arguments, The Program Will Plot All
       // Record Variables That The User Declares
       // Which are "a" to "h" ( 8 Variables).
       // There Are Only 11 Different Colors That
       // Support Graph Plotting, The Program Will Plot
       // Only 11 Variable If There Are More Variables.
       // For This Case : User Can Separate Variables
       // in Groups, And Plot Each Group.
19:plot ##acgh // "#" Preceeding, Tell The Program To Plot Some
       // Variables Only.
20:return // When Reach "return" for First Time , Nothing
       // Happen Because The Program Will Skip This Line.
       // However, If The Program Jumps From "goto"
       // Command, The "return" Will Force The Program
       // To Jump Back To The Next Command Beyond "goto".

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

21:width 1200          // Set Number of Data to 1200.
                        // "width"=1200 are 2 Full Screen Display.
                        // See That,"width" is Changed From Previous Value
                        // So, The Memory Allocated For Record variables
                        // Will be Modified each Whose Sizes Equal to The
                        // New "width".
22:time 10             // Set New Time Range
                        // The Previous Value Will Be Substituted by 10
                        // Automatically.
23:record ab           // Allcate Memory For Record Variables.
                        // If Size of Record Variables Are So Much, Some
                        // Variables May not Succeed in Allocation.

24:equation            // If The New Equation Systems Are Defined, They
                        // Will Replaced The Previous Ones Implicitly.
25:a=(t+1)*(t-1)*(t-4)*(t-6)*(t-9)*(t-15)
                        // Equation Must not be Longer Than 80 Characters
26:b=1000*sin(t)/(t+0.000001)
                        // 0.000001 Prevent "Zero Deviding" (Overflow).
                        // Because The Equations are Evaluated From
                        // Time=0.
27:endequation

28:run
29:plot                // This Line Will Plot Only "a" And "b".
29:showt 1             // Show The Time Consumption of Program.
                        // End Program.
                        // This Reserved Word Will Always Used
                        // in The Last Line of Program.

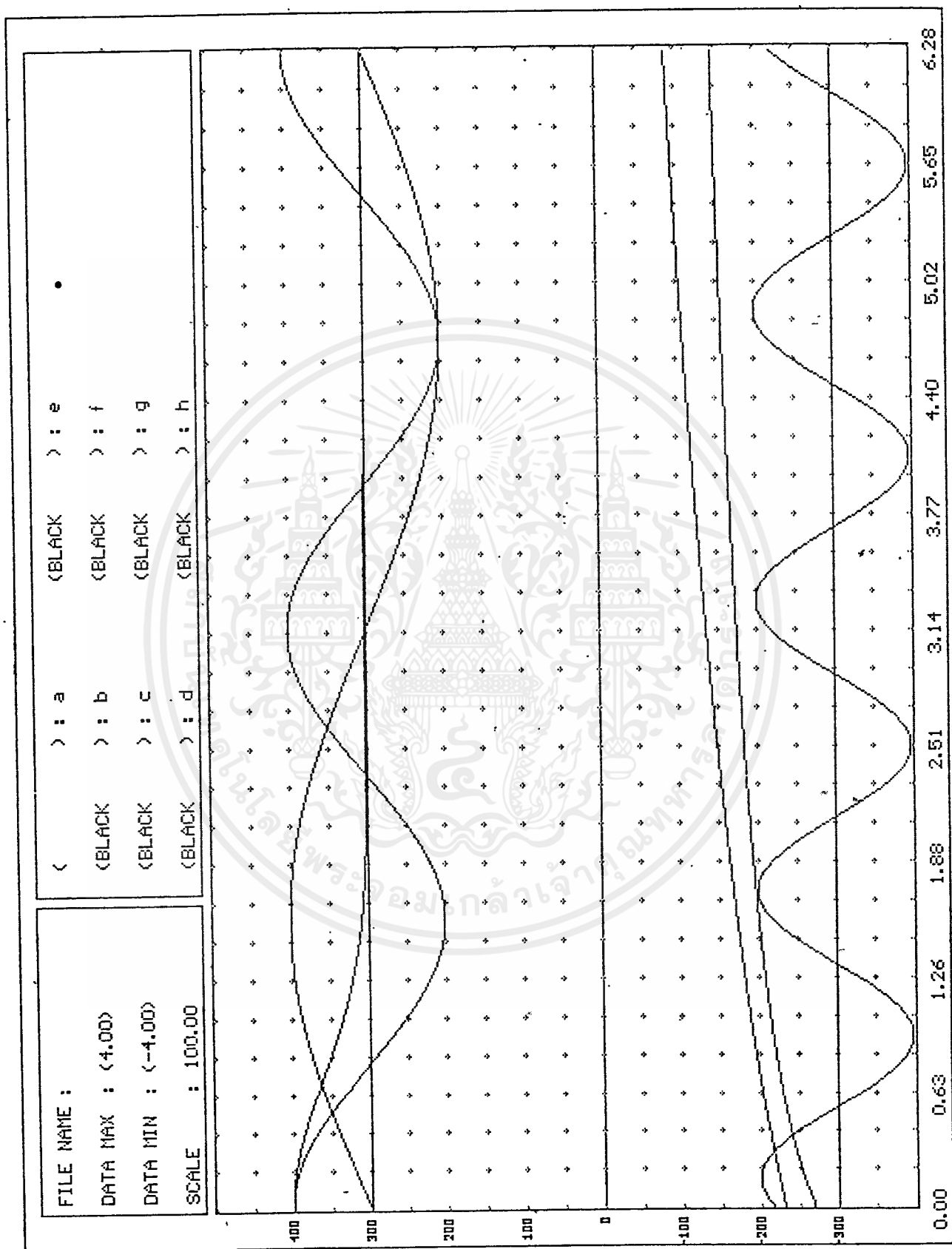
endprogram

```

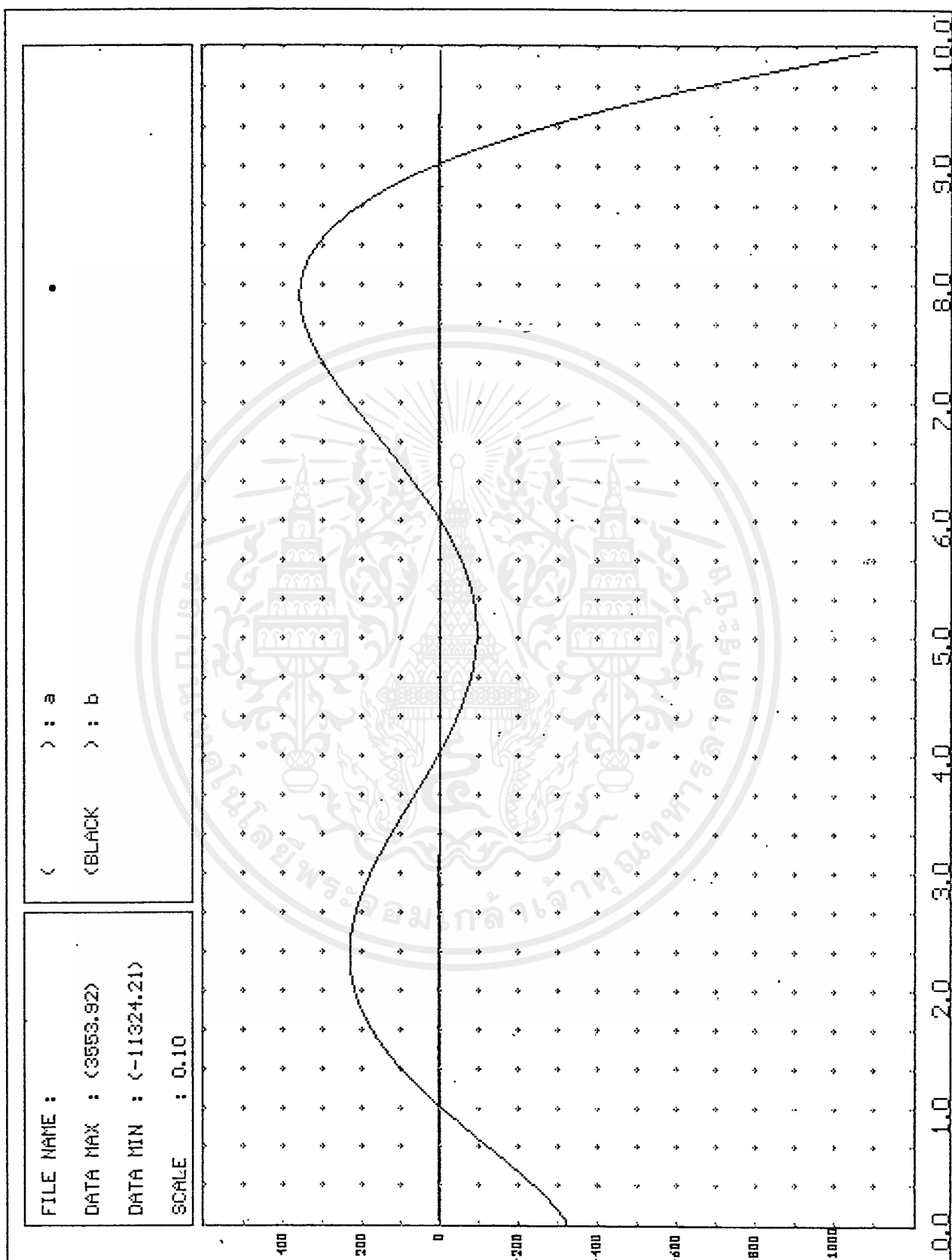
```

/***** Executable Program *****/
/***** Compile & Run : OK *****/
simulate
1 :freemem
2 :showt 2
3 :width 600
4 :time 6.28
5 :precision 1
6 :record abcdefgh
define Shift=3
7 :equation
8 :a=sin(t)+Shift
9 :b=cos(2*t)+Shift
10:c=exp(-t*t)+Shift
11:d=Shift
12:e=sin(4*t+1)-Shift
13:f=ln(t+2)-Shift
14:g=log10(5*t+2)-Shift
15:h=-Shift
16:endequation
17:run
18:plot
19:plot ##acgh
20:return
21:width 1200
22:time 10
23:record ab
24:equation
25:a=(t+1)*(t-1)*(t-4)*(t-6)*(t-9)*(t-15)
26:b=1000*sin(t)/(t+0.000001)
27:endequation
28:run
29:plot
29:showt 1
endprogram

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (68) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (69) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่างโปรแกรมแสดงการประยุกต์ใช้โปรแกรมเพื่อทำการอินทิเกรต

โปรแกรมนี้แสดงการทำการอินทิเกรตฟังก์ชันที่เป็นความสัมพันธ์กับเวลาโดยลักษณะของการอินทิเกรตจะเป็นแบบมีขอบเขตจำกัด กล่าวคือ จะมีการกำหนดขอบเขตล่างและขอบเขตบนของการอินทิเกรต ตัวอย่างโปรแกรมนี้นี้จะไม่มีผลการรัน เนื่องจากการจะไม่มีการพล็อตกราฟแต่ผลลัพธ์จะเป็นการแสดงค่าที่ได้จากการอินทิเกรตออกหน้าจอและเนื่องจากการทำการอินทิเกรตเราจะใช้หลักการคำนวณเชิงตัวเลขและเพื่อให้ข้อผิดพลาดจากการคำนวณเชิงตัวเลขมีค่าน้อยที่สุด การจะทำการอินทิเกรตฟังก์ชันนั้น ๆ จึงมีการใช้เนื้อที่ในหน่วยความจำมาก จึงทำให้เกิดข้อจำกัดต่างๆในการเขียนโปรแกรม กล่าวคือ

1) ค่าพารามิเตอร์ของคำสั่ง width ต้องมีค่าน้อยกว่า 32767

2) กรณีที่เราต้องการทำการอินทิเกรต เราควรจะหลีกเลี่ยงการจองตัวแปรแบบ record เพื่อทำการพล็อตกราฟ เนื่องจากเหตุผลดังกล่าวแล้วข้างต้น กล่าวคือจะเกิดปัญหาเกี่ยวกับเนื้อที่ในหน่วยความจำจะไม่พอเพียงสำหรับการประมวลผล

```

simulate          // This Reserved Word Will Always be Used
                  // in The First Line of Program.

# Program for Evaluate of, Integration
# File Name : INTEGRAT.1
# Date      : 18 March 1993

1 :freemem        // Free Memory That Allocated Before
                  // And Set All Parameter To Default Value.
2 :showt  2       // Reset Clock To Check Time Consuming.
                  // (Start Timing)
3 :width  20000   // The "width" Can not Set Greater Than 32767
                  // 32767= 215-1
                  // In Case of Integrating, Graph is not Displayed
                  // And Record Variables Are Not Necessary to
                  // Declared.
                  // Warning : User MUST NOT Declare Record
                  // Variables Because Memory May not Sufficient.
                  // Command "record" is not Necessary in This
                  // Program Because We Want to Integrate Graph
                  // Not Plot Graph.
4 :time  50       // Set Time Range of Integration.
                  // This Command Refer The Same Variable "time"
                  // Existing In Other Programs.
5 :precision 1    // This Precision Value Cannot Effect The System
                  // Equation That Don't Have Diffential Equation.
                  // So, This Command Line Can Be Removed.

8 :integrate exp(-t) // The Parameter of This Function is "INTEGRAND"
9 :integrate exp(-t*t) // And is Function of Time.
10:integrate t^2+t-1

29:showt 1        // Show The Time Consumption of Program.
30:endprogram      // End Program.
                  // This Reserved Word Will Always Used
                  // in The Last Line of Program.

```

ตัวอย่างการแปลงฟูเรียร์และการแปลงกลับสู่สัญญาณเดิม

สัญญาณที่ใช้เป็นรูปฟันเลื่อยกับรูปสี่เหลี่ยม จะทำการพล็อตบนแกนเวลาของสัญญาณเดิมแต่ค่าที่อ่านได้นั้นจะอ่านเป็นค่าของความถี่โดยจะเรียงตามลำดับของฮาร์โมนิกส์

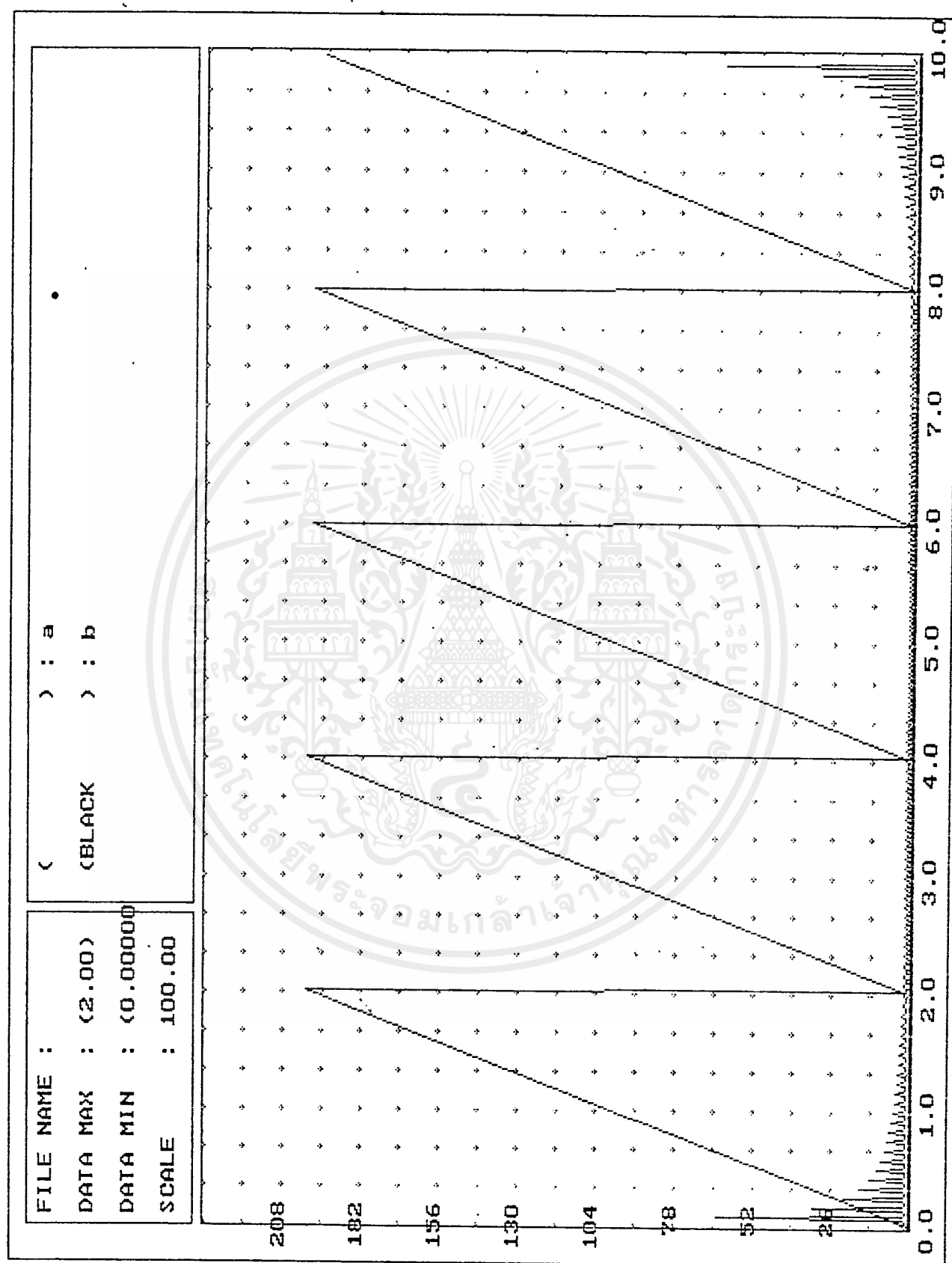
เมื่อทำการแปลงกลับสัญญาณก็จะได้สัญญาณที่มีค่าใกล้เคียงกับสัญญาณเดิม



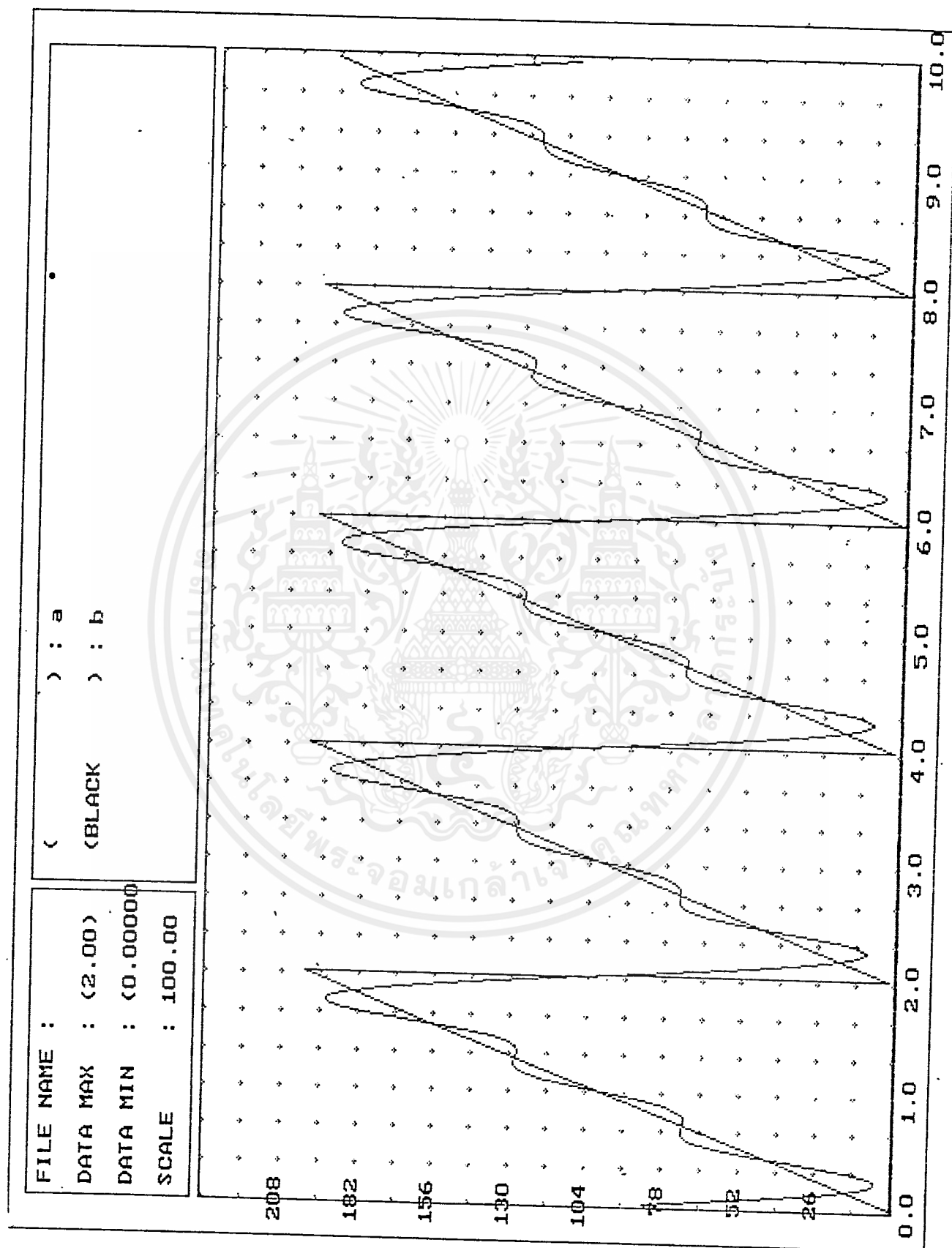
```
# Program : Discrete Fourier Transform (And Inverse DFT)
# File : FOURIER.1
# Require : Memory (Sufficient)
```

```
simulate
1 :freemem
2 :showt 2
3 :width 600
4 :time 10
5 :record abxy
6 :equation
7 :a=t%2
8 :b=sign(1-t%2)
9 :endequation
10:run
11:fft axy
12:ifft cxy 0 0.05
13:fft bxy
14:ifft cxy 0 0.05
endprogram
```

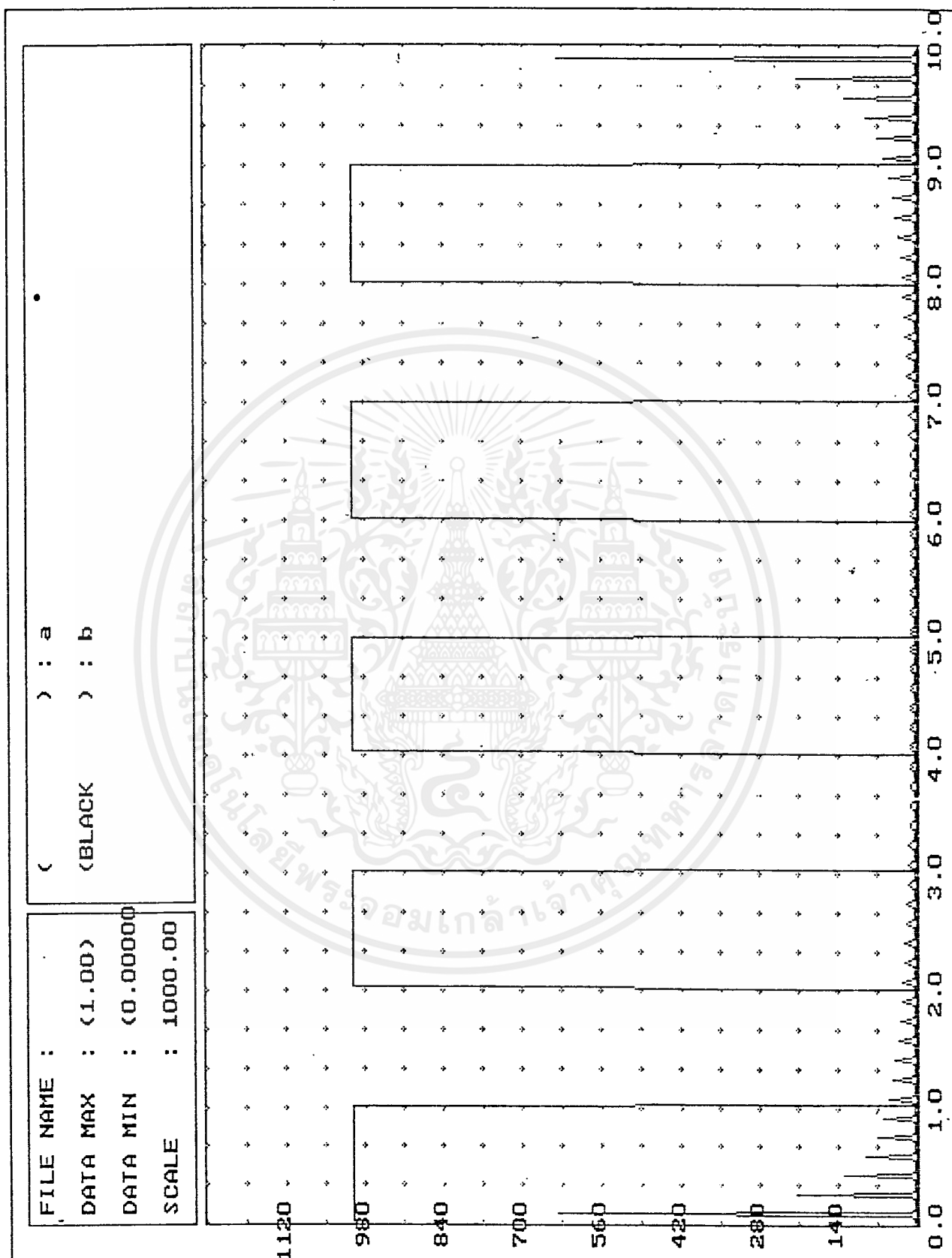




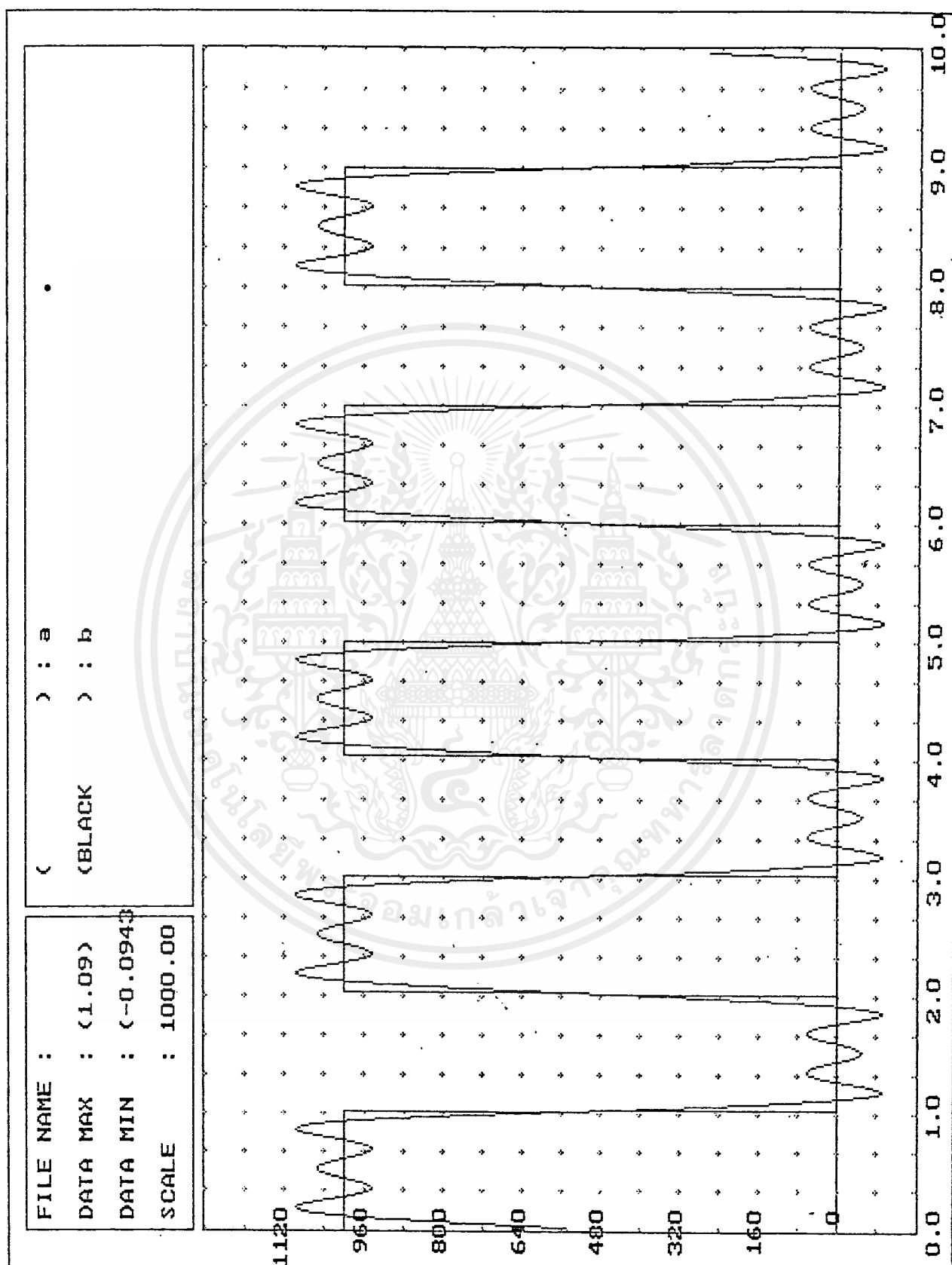
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (75) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (76) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

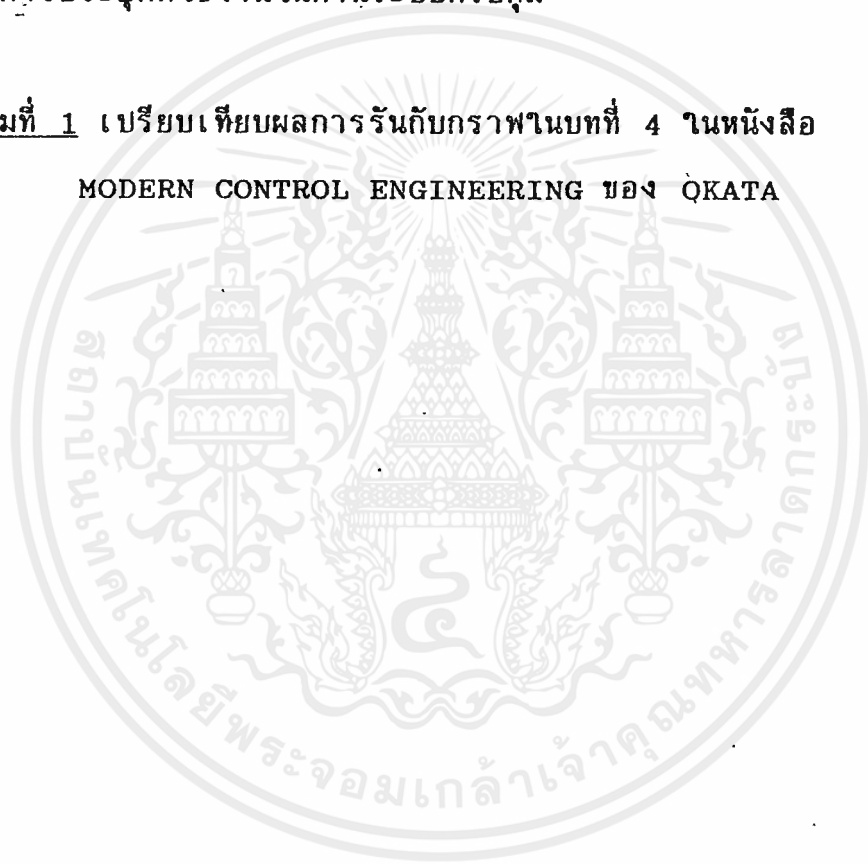


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (77) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่างการประยุกต์ใช้งานในด้านระบบควบคุม

โปรแกรมที่ 1 เปรียบเทียบผลการรันกับกราฟในบทที่ 4 ในหนังสือ

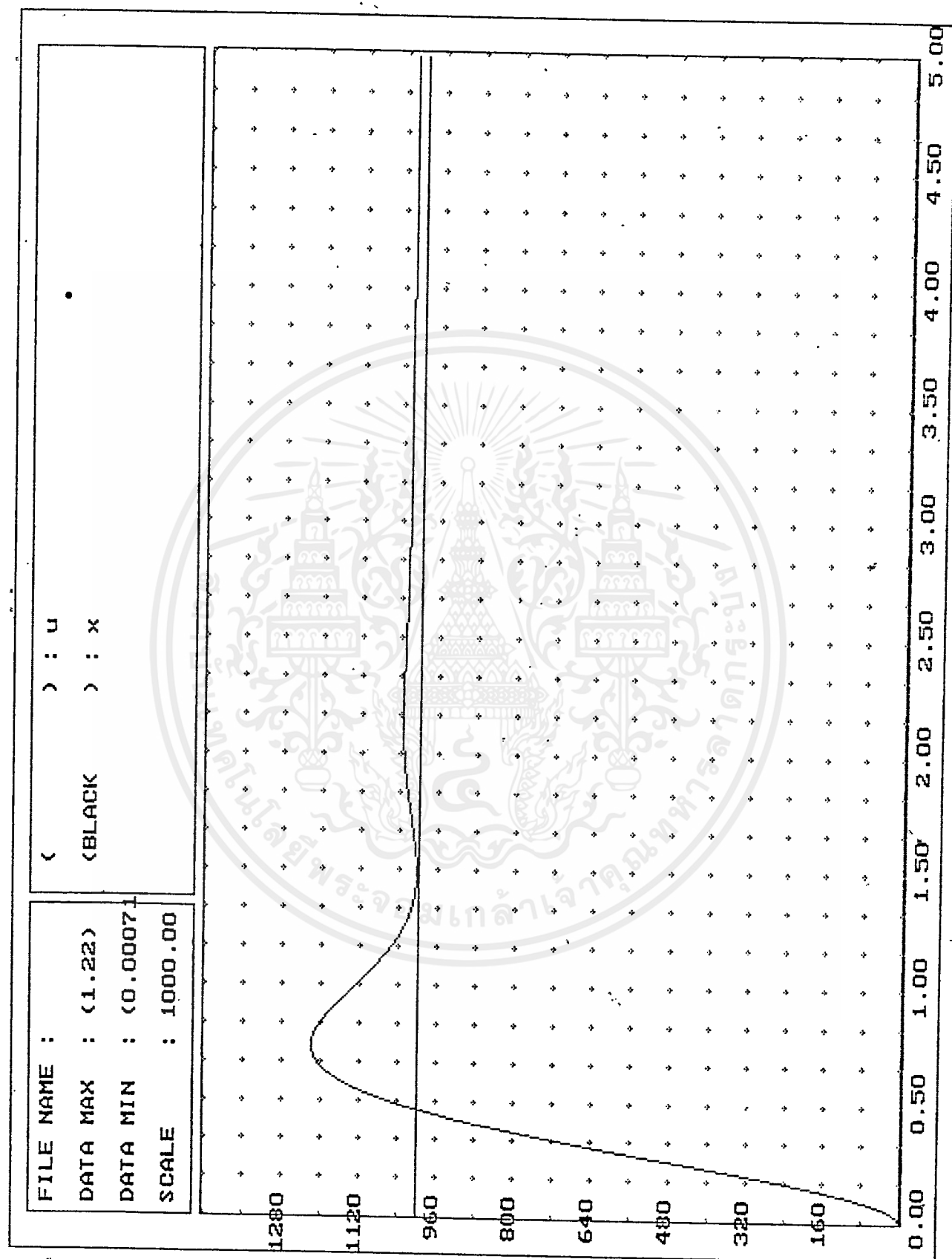
MODERN CONTROL ENGINEERING ของ OKATA



```

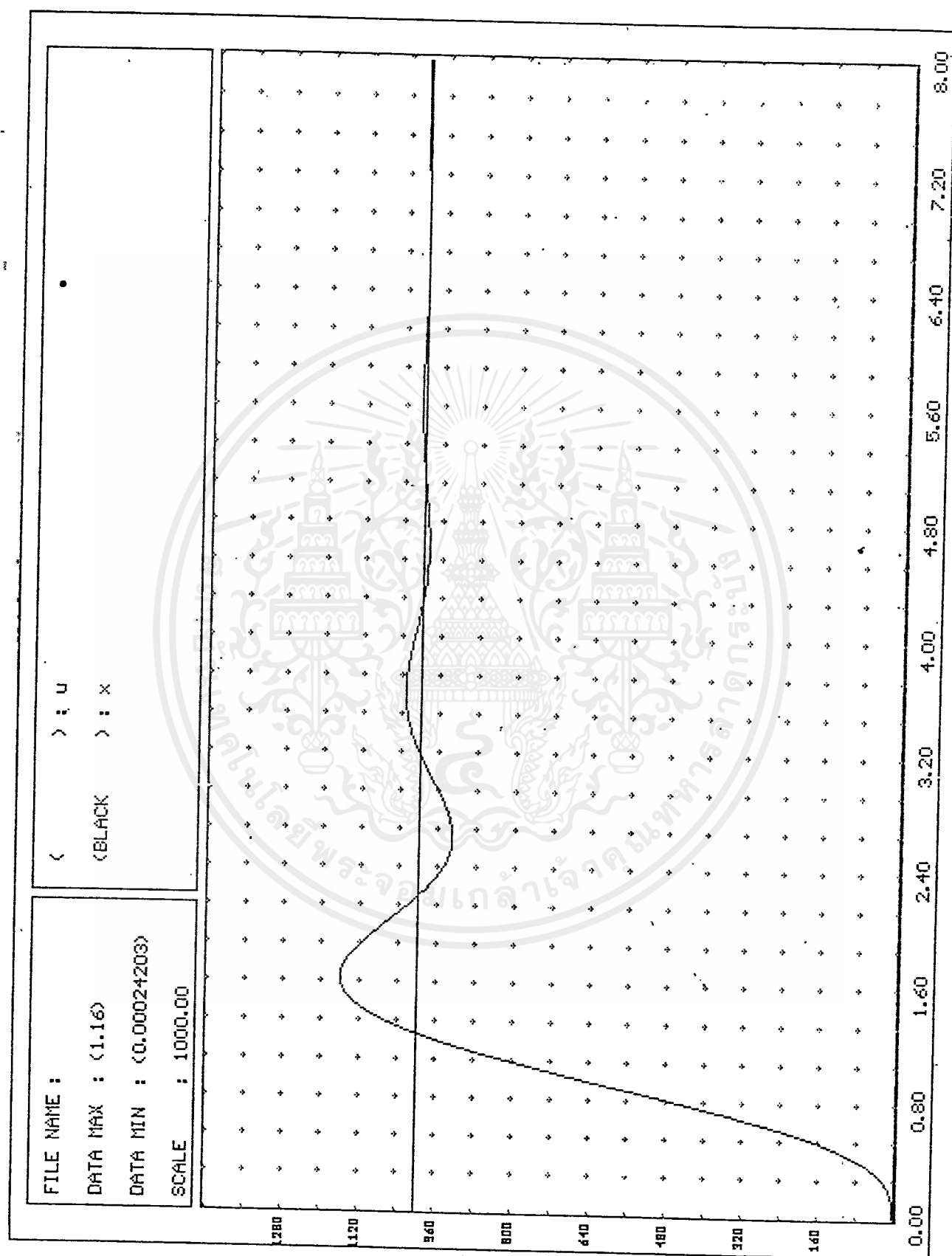
simulate
#
# Example of Unit 4 in "OGATA" (Figure 4-43, p. 316)
#
showt 0
precision 10
width 600
time 5.0
variable xsyszs
record ux
define INPUT=1
equation
u=INPUT
x=y
y=z+25.04*u
z=-5.008*x-25.1026*y-5.032478*z-121.005*u
endequation
run
plot
#
# Example of Unit 4 in "OGATA" (Ex.A-4-24)
#
freemem
precision 10
width 600
time 8.0
variable asxsyszs
record ux
equation
u=INPUT
x=y
y=z
z=a+5*u
a=-100*x-80*y-32*z-8*a+60*u
endequation
run
plot
endprogram

```

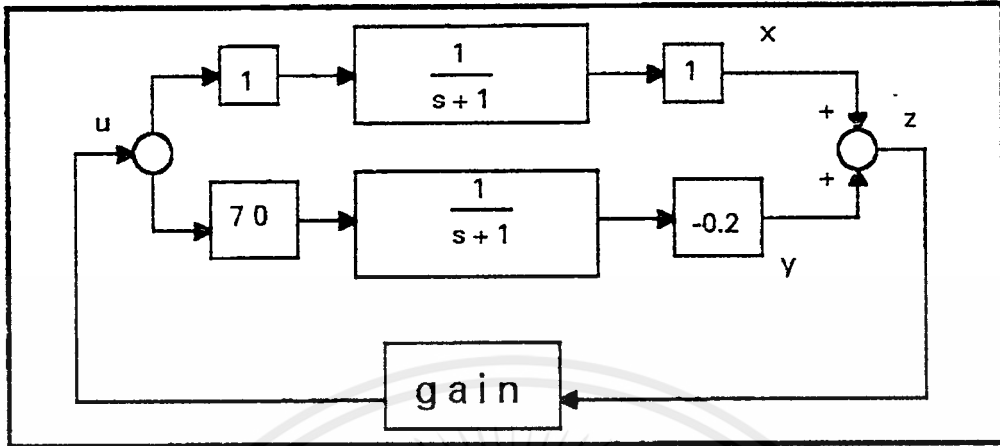


(80)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



โปรแกรมที่ 2 เป็นการวิเคราะห์ระบบวงปิดที่มีลักษณะดังรูป



พิจารณาความสัมพันธ์ของระบบ จะได้ว่า

$$\frac{x}{u} = \frac{1}{s+1}$$

$$\dot{x} = u - x$$

$$\frac{y}{u} = \frac{-14}{s+10}$$

$$\dot{y} = -14u - 10y$$

โดย $u = \text{gain} * (x+y)$ แทนค่าจะได้

$$\dot{x} = \text{gain} * (x+y) - x$$

$$\dot{y} = -14 * \text{gain} * (x+y) - 10y$$

พิจารณาจากสมการของระบบ เราจะได้ว่า system matrix จะมีค่าเป็น

$$\begin{bmatrix} k-1 & k \\ -14k & -14k-10 \end{bmatrix}$$

ทำการหาตำแหน่งของ pole โดยพิจารณาจากสมการ

$$(\lambda - k + 1)(\lambda + 14k + 10) + 14k^2 = 0$$

$$\lambda^2 + (13k + 1)\lambda + 10 = 0$$

$$\lambda = [-(13k + 1) \pm \sqrt{(13k + 1)^2 - 40^2}] / 2$$

ระบบจะเสถียรเมื่อส่วนจริงของโพลอยู่ด้านซ้ายของระนาบ ฉะนั้นจะได้

$$13k + 11 > 0$$

นั่นคือเราจะได้ว่าระบบจะเสถียรเมื่อ $k > \frac{-11}{13}$

กราฟรูปแรก ทำการกำหนดค่า gain เป็น -1 นั่นคือ ระบบมีลักษณะไม่เสถียร จากกราฟเราจะเห็นได้ว่าเมื่อเวลามีค่าเพิ่มขึ้น ลักษณะของสัญญาณ u, x และ y มีลักษณะที่ลู่ออก

หมายเหตุ ค่าเริ่มต้นของตัวแปรสเตทในที่นี้เรากำหนดให้

$$x(t=0) = y(t=0) = 10$$

กราฟรูปสอง ทำการกำหนดค่า gain เป็น -11/3 นั่นคือระบบมีลักษณะออสซิลเลทจากกราฟเราจะเห็นได้ว่าเมื่อเวลามีค่าเพิ่มขึ้น ลักษณะของสัญญาณ u, x และ y มีลักษณะที่แกว่งนั่นเอง

หมายเหตุ จากกราฟสาเหตุที่การแกว่งไม่มีลักษณะเป็นไซน์โดยสมบูรณ์เนื่องจากเวลาที่เรากำหนดในการเขียนโปรแกรม(กำหนดโดยคำสั่ง time)มีค่าน้อยเกินไปนั่นเอง

กราฟรูปสาม ทำการกำหนดค่า gain เป็น 2 นั่นคือ ระบบมีลักษณะเสถียรจากกราฟเราจะเห็นได้ว่าเมื่อเวลามีค่าเพิ่มขึ้น ลักษณะของสัญญาณ u, x และ y มีลักษณะที่ลู่เข้านั่นเอง

หมายเหตุ จากกราฟสาเหตุที่การแกว่งไม่มีลักษณะเป็นไซน์โดยสมบูรณ์เนื่องจากเวลาที่เรากำหนดในการเขียนโปรแกรม(กำหนดโดยคำสั่ง time)มีค่าน้อยเกินไปนั่นเอง

```

simulate
// This Reserved Word Will Always be Used
// in The First Line of Program.

# Program simulation for Plot Graph Example.
# File Name : FEEDBACK.1
# Date : 18 March 1993
# REMARK : "#" Used For Comment Line.
# "://" Used For Comment Line or Part of Line.

1 :freemem // Free Memory That Allocated Before
// And Set All Parameter To Default Value.
2 :showt 2 // Reset Clock To Check Time Consuming.
// (Start Timing)
3 :width 600 // Set Numbar of Data to 600.
4 :time 10.0 // Set Time Range of Time Domain.
5 :precision 20 // The Precision Value Will Effect The System-
// That Have Diffential Equations.
// Warning : If The "precision" is Not Large
// Enough, Calculation May be Overflowed Becuase
// of More Iteration Time.
6 :variable xsys // See That Variable x,y are State Variable.
// They are Defined by Differential Equations.
// We Declare Them to Have Variable Type "s".
// Comma May be Used for Separating Them in
// Pairs. (Other Variables That are Not Declared
// Will be Defaulted to Have Variable Type "e".
7 :initial xlyl // Set Initial Value to x And y
// This Initial Value Will Be Poped Up(Restored)
// When The Program Complete Execution "run"
// Command.
8 :record uxy // Allcate Memoty For Record Variables.

define Gain=-1 // Warning : The Word "Gain" MUST Not Exist in
// Other Places Except Equation, Because "IT"
// Will Be Replaced by -2 Immediatedly.
// User Will Define The contant Carefully.

9 :equation
10 :z=x+y // This Equation Should Preceed "u=Gain*z".
// It Changes Very Little When This Equation
// is Removed To Last Equation.
11 :u=Gain*z // "Gain" is -2 when Reaching The First Time
// and "Gain" is -11/13 for The Second Time
// and "Gain" is 2 for The Third Time

12 :x=u-x
13 :y=-14*u-10*y
14 :endequation

15 :run // Calculate All Equations Consecutively From
// time=0 to time=Time Range.
// Remark: The System Equation Will be Orderd
// Correctly. (Depend on Order in Calculation)
16 :plot // If no Arguments, The Program Will Plot All
// Record Variables That The User Declares
// Which are "u", "x", "y" ( 3 Variables).
17 :return // When Reach "return" for First Time , Nothing
// Happen Because The Program Will Skip This Line.
// However, If The Program Jumps From "goto"
// Command, The "return" Will Force The Program
// To Jump Back To The Next Command Beyond "goto".

# Execute The System Response (Second Time)
# But Change The Feedback Gain to -11/13
define Gain=-11/13 // This Word Can be Redefined,but The Value -11/13
// Will Be Assigned To "Gain" When The Program
// Reach This Command.
// The New Value Will Replace The Previous Value.
18 :goto 9 // Goto "equation" Command. (in Line 9)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (84) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

// The Line No. not Embrace "define", Comment Line
// and "Simulate". (These Lines Have No Effect in
// Counting Line No. of The Program)
// Notice That "endequation" Will be Counted ,Too.

# Execute The System Response (Second Time)
# But Change The Feedback Gain to 2
define Gain=2          // The Value 2 Will be Assigned to "Gain" When The
                        // Program Reach This Command.
                        // 2 Will Replace -11/13
19:goto 9               // Goto "equation" Command. (in Line 9)
20:showt 1              // Show The Time Consumption of Program.
                        // End Program.
                        // This Reserved Word Will Always Used
                        // in The Last Line of Program.

endprogram

```

```

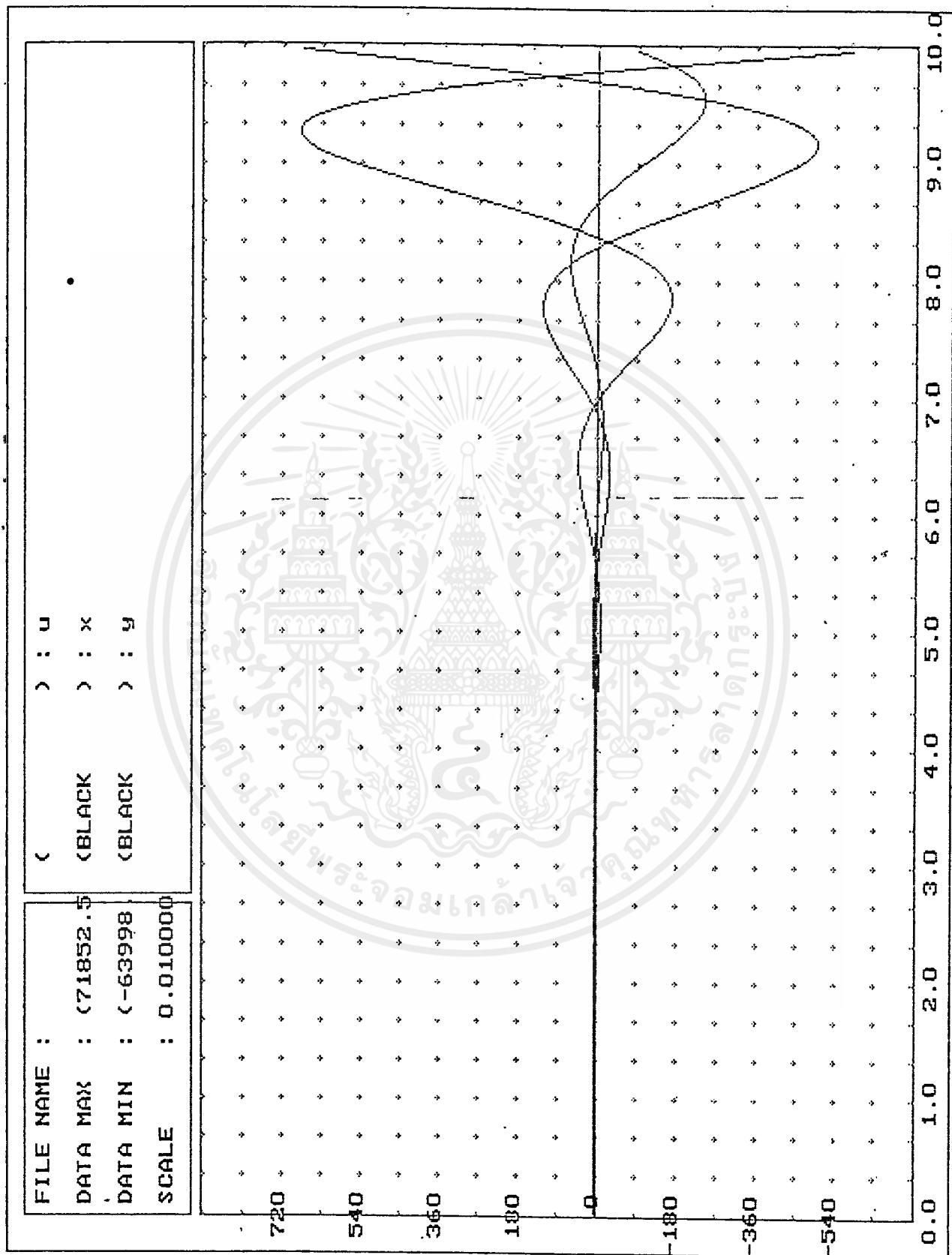
#/****** Executable Program *****/
#/****** Compile : OK *****/
#/****** Run : OK *****/

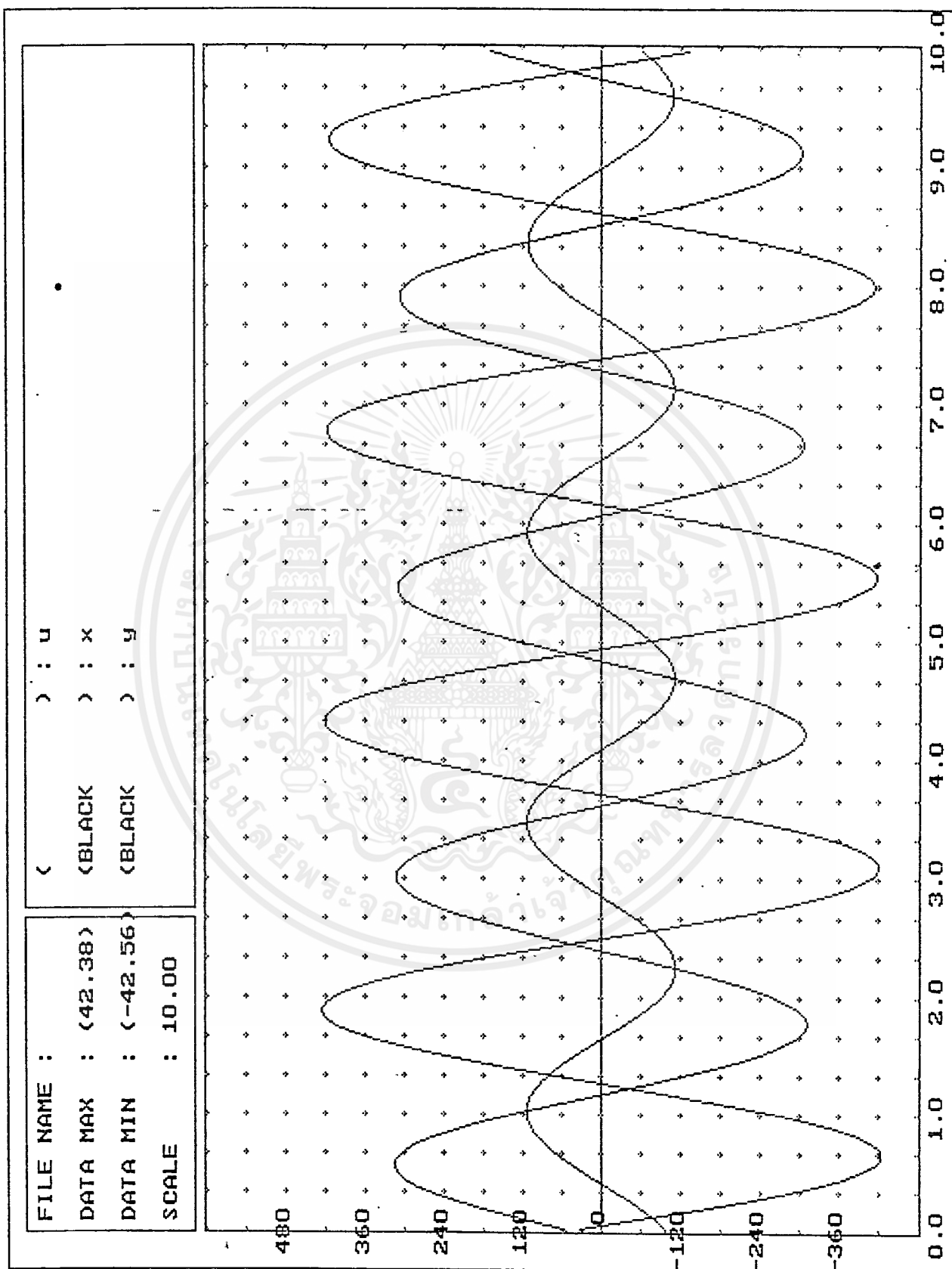
```

```

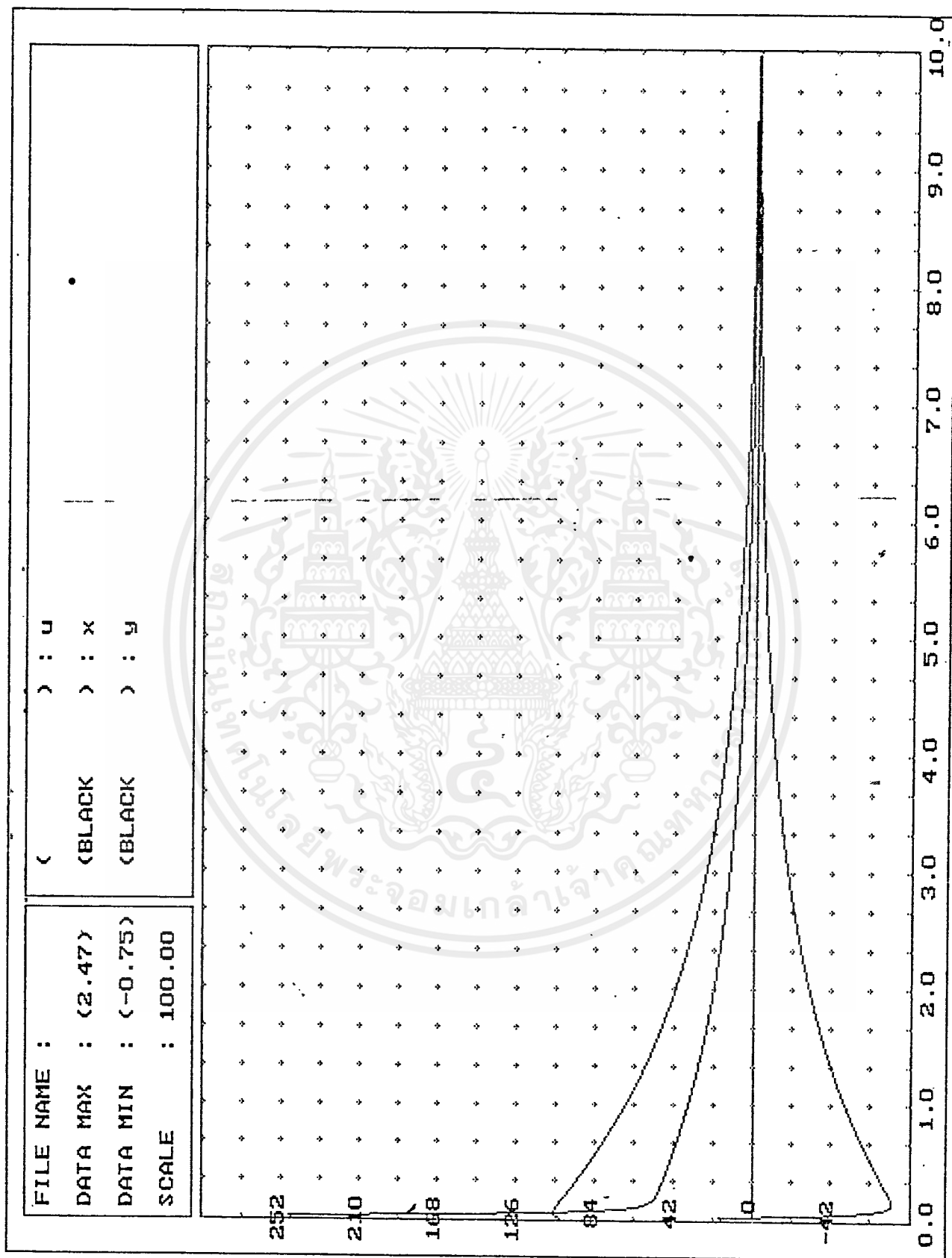
simulate
freemem
showt 2
width 600
time 10.0
precision 20
variable xsys
#
#UNSTABLE
initial xly1
define Gain=-1
record uxy
equation
z=x+y
u=-1.0*z
x=u-x
y=-14.0*u-10.0*y
endequation
run
plot
return
#
# STABLE
define Gain=2
initial xly1
equation
z=x+y
u=2*z
x=u-x
y=-14*u-10*y
endequation
run
plot
#
# OSCILLATE
initial x-10y5
define Gain=-11/13
equation
z=x+y
u=-11/13*z
x=u-x
y=-14*u-10*y
endequation
run
plot
showt 1
endprogram

```





เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (87) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมที่ 3 เป็นการออกแบบตัวควบคุมสำหรับระบบที่ไม่เป็นเชิงเส้น

สำหรับตัวอย่างนี้เป็นการประยุกต์ใช้โปรแกรม ในการออกแบบตัวควบคุมสำหรับควบคุมระบบที่ไม่เป็นเชิงเส้น โดยจะใช้ทฤษฎีที่ได้กล่าวไว้ในบทที่ 4 ซึ่งจะพิจารณาลักษณะการควบคุมทั้งแบบ MRAC และแบบคอมโพสิต

ในตัวอย่างนี้ สมการของพลานท์เป็นดังสมการ

$$\dot{y}_p = -a_p y_p - c_p y_p^2 + b_p u$$

โดยมีเทอม $c_p y_p^2$ ที่แสดงลักษณะของการไม่เป็นเชิงเส้นซึ่งจะต้องทำให้เทอมนี้มีค่าเข้าใกล้ 0 ที่สุดจึงเลือก

สมการของโมเดลเป็นดังสมการ

$$\dot{y}_m = -a_m y_m + b_m r(t) \quad \text{-----}(1)$$

หลักการควบคุมแบบมีโมเดลอ้างอิง

จากบทที่ 4 หัวข้อ 4.3.2 แทน $f(y)$ ด้วย y_p^2 ดังนั้น จะได้

$$\text{กฎควบคุมคือ } u = \hat{a}_r r(t) + \hat{a}_y y_p + \hat{a}_f y_p^2$$

แทนลงในสมการของพลานท์จะได้ว่า

$$\dot{y}_p = -(a_p - \hat{a}_y b_p) y_p - (c_p - \hat{a}_f b_p) y_p^2 + \hat{a}_r b_p r(t) \quad \text{-----}(2)$$

$$\text{กฎการปรับตัวคือ } \dot{\hat{a}}_r = -p_o \operatorname{sgn}(b_p) e r$$

$$\dot{\hat{a}}_y = -p_o \operatorname{sgn}(b_p) e y_p$$

$$\dot{\hat{a}}_f = -p_o \operatorname{sgn}(b_p) e y_p^2$$

p_o เป็นเกนในการปรับตัว

$$\text{ทดสอบโดยแทนค่าพารามิเตอร์ของโมเดล } a_m = b_m = 2$$

$$\text{และของพลานท์ } a_p = b_p = 1, c_p = -1$$

จะได้สัญญาณของผลตอบสนองดังรูปหน้าถัดไป สังเกตว่าการปรับค่าพารามิเตอร์จะทำทุกครั้งที่มีความแตกต่างระหว่างสัญญาณเอาต์พุต(output signal) ของโมเดลกับพลานท์ แต่ถ้าเมื่อใดที่สัญญาณทั้งสองเท่ากันก็就不用มีการปรับค่าพารามิเตอร์ จนกว่าจะมีความแตกต่างของสัญญาณอีก

ดังนั้นเพื่อให้การปรับค่าพารามิเตอร์ดีขึ้นจึงได้เพิ่ม ส่วนการทำนายค่าผิดพลาดซึ่งไม่ได้สนใจค่าสัญญาณแตกต่างของโมเดลกับพลานท์ แต่จะใช้หลักการทำนายสัญญาณแตกต่างระหว่างค่าเอาต์พุตของพลานท์ในปัจจุบันกับค่าในอนาคต แล้วจะใช้กฎการปรับค่าพารามิเตอร์เพื่อปรับให้มีค่า

น้อยที่สุด

เลือกใช้วิธีการเดียวกันในการประมาณค่าพารามิเตอร์ และจัดรูปสมการตามหัวข้อ 4.6 ได้ว่า

$$\frac{u}{s+a_m} = a_r \frac{y_p}{b_m} + a_y \frac{y_p}{s+a_m} + a_f \frac{y_p^2}{s+a_m}$$
$$u_1 = \mathbf{W} \mathbf{a}$$

เลือกกฎการปรับค่าพารามิเตอร์ $\hat{\mathbf{a}}^* = -\mathbf{P}_o \mathbf{W} \mathbf{e}_1$

เมื่อรวมกับ MRAC จะได้กฎการปรับค่าพารามิเตอร์แบบคอมโพสิตทั้งหมดดังนี้

$$\dot{\hat{a}}_r^* = -p_o [\text{sgn}(b_p) e r + w_1^2 (\hat{a}_r - a_r)] \quad \text{-----}(3)$$

$$\dot{\hat{a}}_y^* = -p_o [\text{sgn}(b_p) e y_p + w_2^2 (\hat{a}_y - a_y)] \quad \text{-----}(4)$$

$$\dot{\hat{a}}_f^* = -p_o [\text{sgn}(b_p) e y_p^2 + w_3^2 (\hat{a}_f - a_f)] \quad \text{-----}(5)$$

และสมการของสัญญาณ w

$$w_1 = \frac{y_p}{b_m} \quad \text{-----}(6)$$

$$\dot{w}_2 = -a_m w_2 + y_p \quad \text{-----}(7)$$

$$\dot{w}_3 = -a_m w_3 + y_p^2 \quad \text{-----}(8)$$

จากนั้นก็นำสมการ (1)-(8) ไปเขียนโปรแกรมพล็อตกราฟ

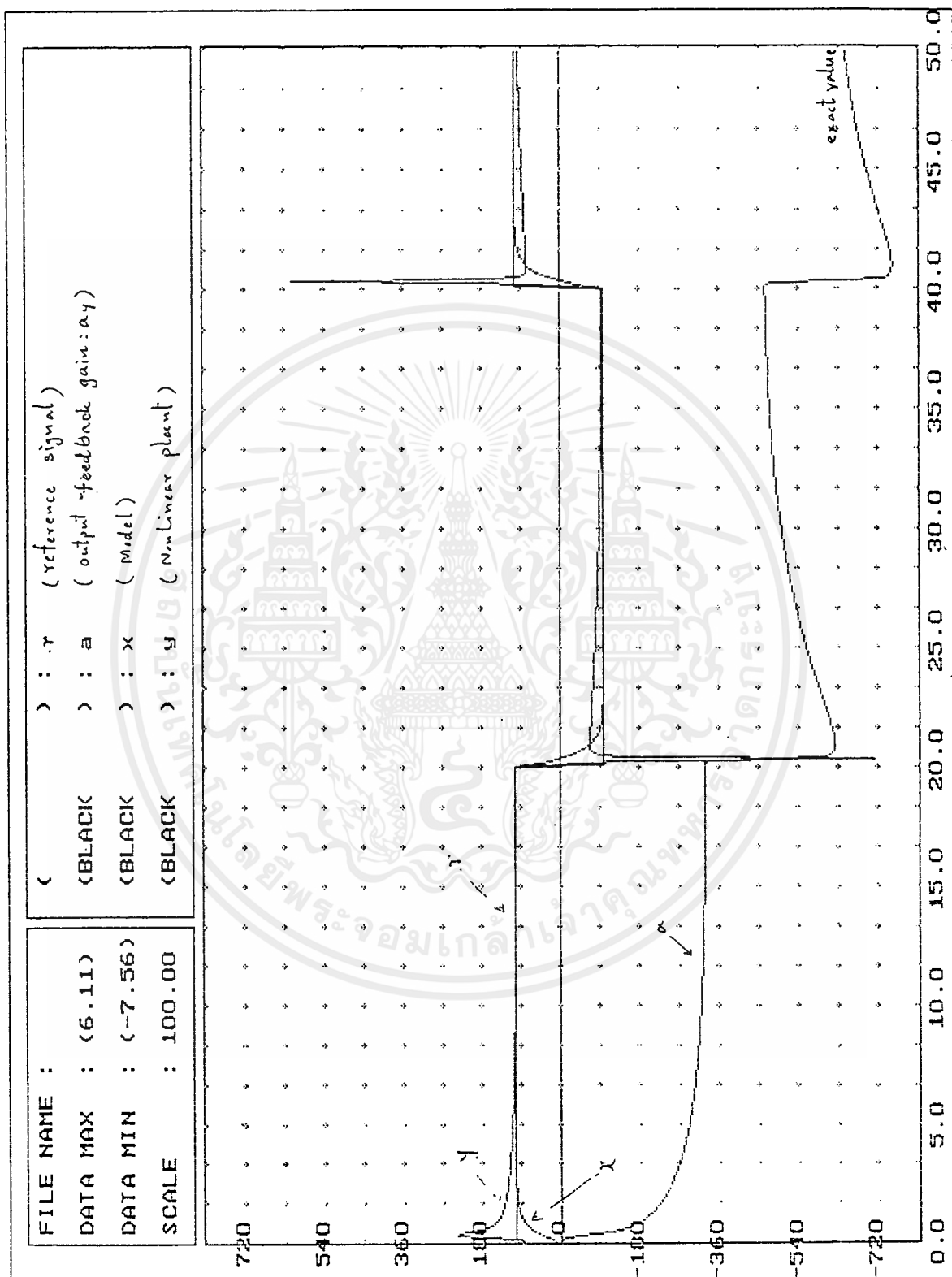
สำหรับสัญญาณอ้างอิงในที่นี้เป็นสัญญาณ clock และช่วงเวลาที่เราพิจารณาผลตอบสนองเป็น 10 วินาที จากกราฟสัญญาณ x เป็นผลตอบสนองของส่วนที่เป็นโมเดล ส่วนสัญญาณ y เป็นผลตอบสนองของส่วนที่เป็นพลานท์ เห็นได้ว่าผลตอบสนองของพลานท์จะปรับเข้าหาโมเดลโดยใช้อินพุตเพียงครั้งคาบเท่านั้นซึ่งให้ผลตอบสนองที่เร็วกว่าการใช้หลัก MRAC เพียงอย่างเดียว

```

simulate
# Model reference Adaptive Control
precision 20
width 600
time 50
record raxy
variable rexsyzsasbs
initial z10
equation
r=(sign(20-t%40)-0.5)*2           // Clock input
a=-(sign(1)*(y-x)*y)              // Adaption law (ay)
b=-(sign(1)*(y-x)*(y^2))          // Adaption law (af)
z=-(sign(1)*(y-x)*r)              // Adaption law (ar)
x=-2*x+2*r                        // Model
y=(-2-a*1)*y-(1-b*1)*(y^2)+z*1*r  // Nonlinear plant
endequation
run
plot
endprogram

```





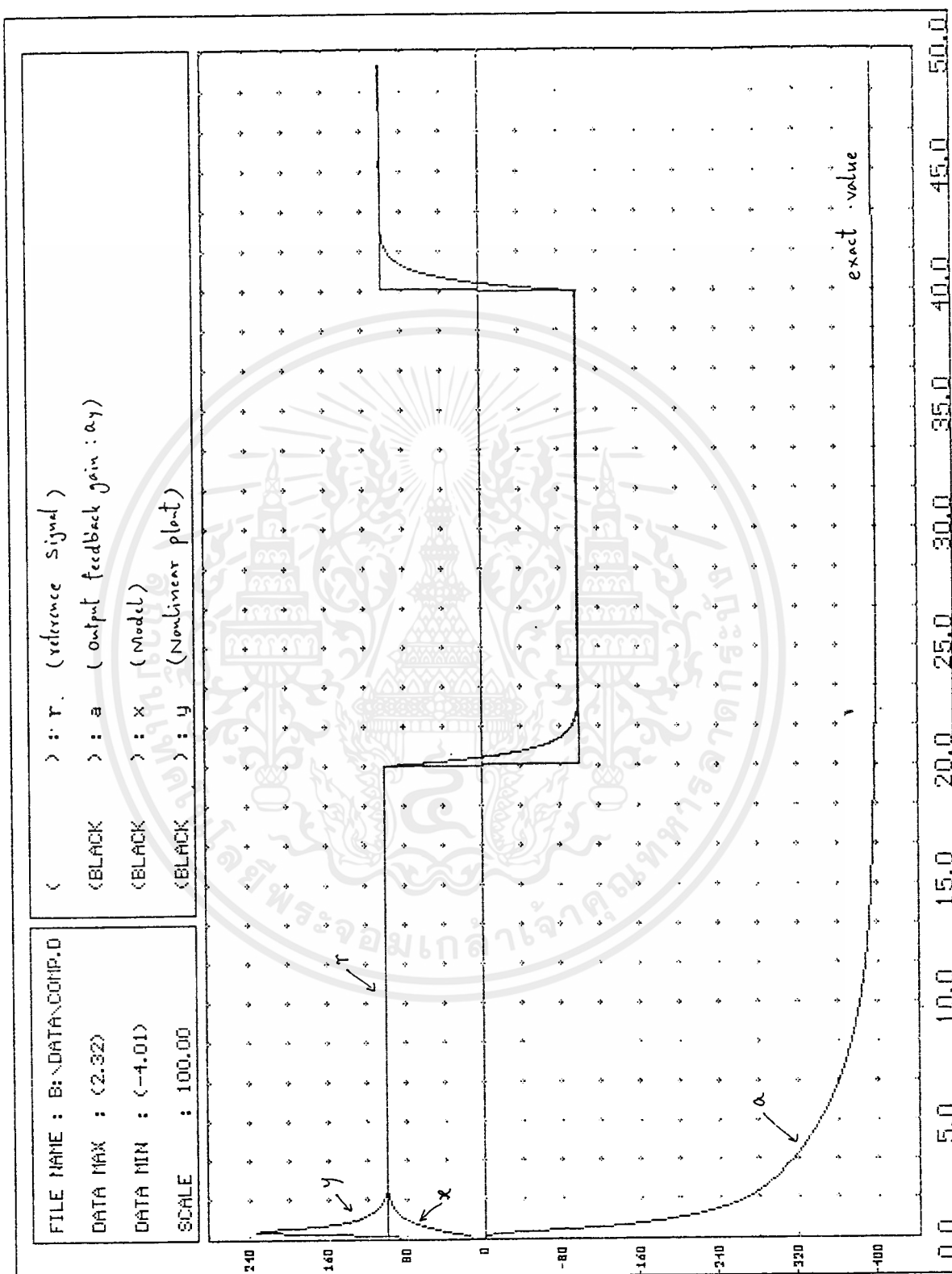
```

simulate
# Composite Adaptive Control
precision 2
width 600
time 10.0
record raxy
variable rexsyzsasbscsds
initial z10
equation
r=(sign(1-t%2)-0.5)*2
c=-2*c+y
d=-2*d+x^2
a=-((sign(1)*(y-x)*y+(c^2)*(a+4))
b=-((sign(1)*(y-x)*(y^2)+(d^2)*(b-1))
z=-((sign(1)*(y-x)*u+((y/2)^2)*(z-2))
x=-2*x+2*r
y=-((-2-a*1)*y-(1-b*1)*(y^2)+z*1*r
endequation
run
plot
endprogram

```

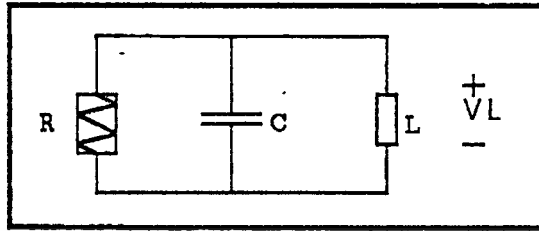
// w2
// w3
// Adaption law (ay)
// Adaption law (af)
// Adaption law (ar)
// Model
// Nonlinear plant





ตัวอย่างเกี่ยวกับวงจรไฟฟ้า

พิจารณาวงจรไฟฟ้าซึ่งมีลักษณะดังรูป



∴ source free parallel
junction circuit

เราได้สมการของวงจรนี้เป็น

$$\dot{v}_c = \frac{1}{c}[-Gv_c - i_o \sin(k * \Phi_L)] \quad , \quad i_L = I_o \sin(k \phi_L)$$

$$\Phi_L = v_c$$

เราสามารถจัดเป็นสมการสแตตเป็น

$$\dot{x} = y$$

$$\dot{y} = \frac{1}{c}[-Gy - i_o \sin(k * x)]$$

กำหนดค่าให้ $c = 1$, $i = 1$, $k = 1$ และ $G = 0.25$

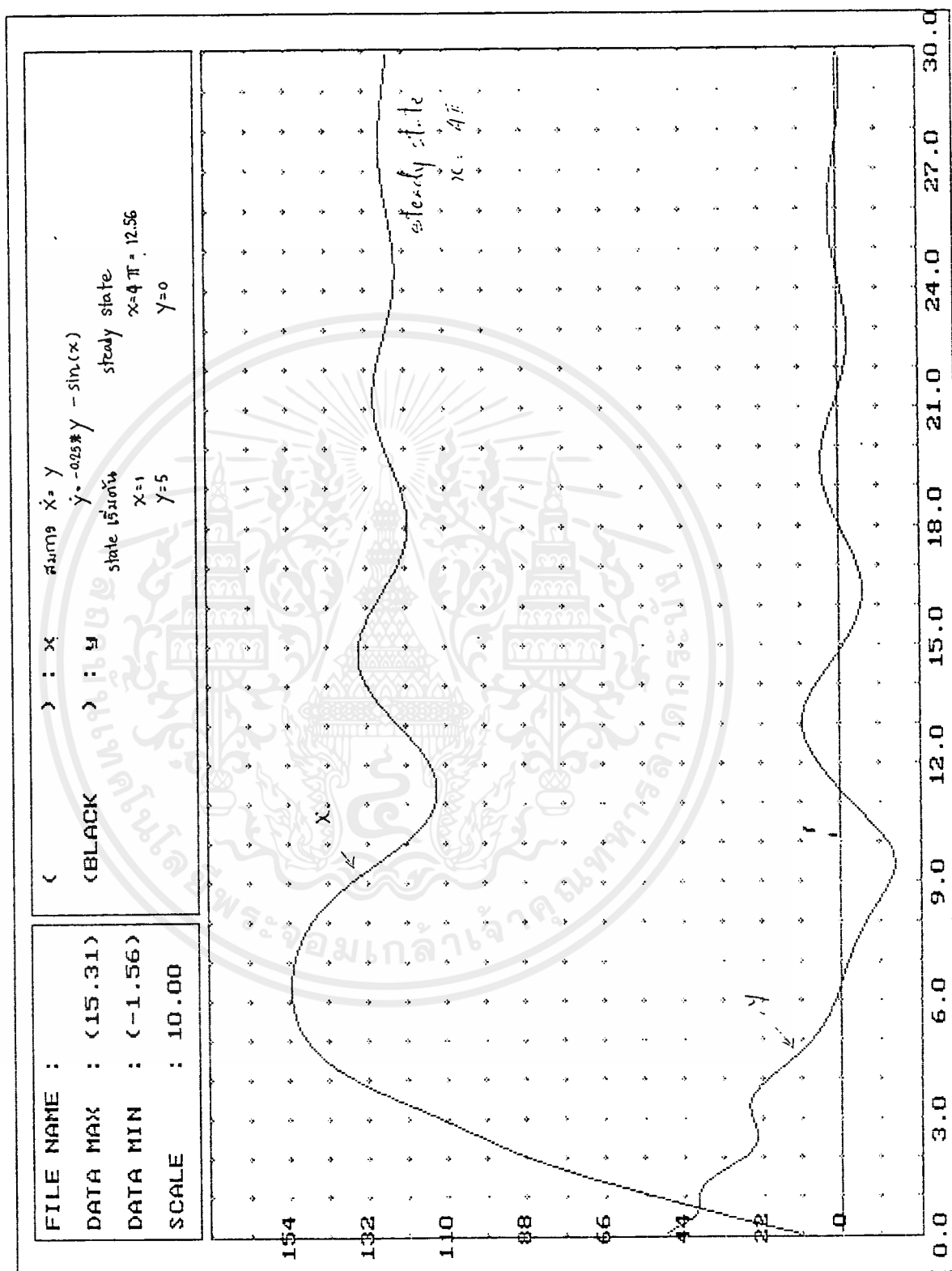
เราจะได้กราฟซึ่งมีลักษณะดังรูป(เปรียบเทียบกับตัวอย่าง)

```

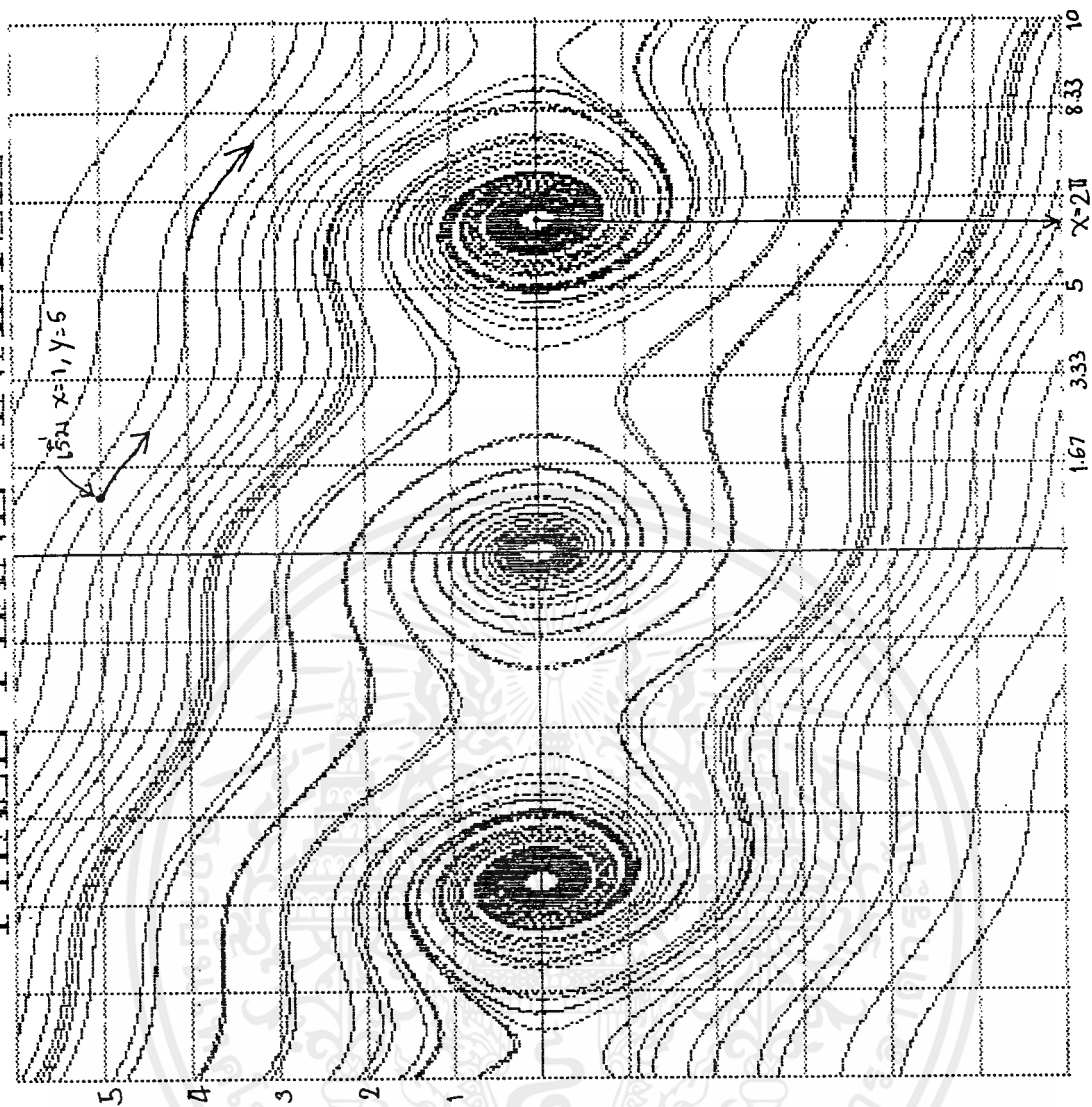
simulate
#
# Ex. for Josephson Junction Circuit
#
width 600
precision 30
time 30
record xy
initial xly5
variable xsys
equation
x=y
y=-0.25*y-sin(x)
endequation
run
plot
endprogram

```





PHASE PLANE ANALYSIS



EQUATION 1 :

$$\dot{x} = y$$

EQUATION 2 :

$$\dot{y} = -0.25*y - \sin(x)$$

RANGE OF X-AXIS :

$(-10.00, 10.00)$

RANGE OF Y-AXIS :

$(-6.00, 6.00)$

ตัวอย่าง VANDER POL OSCILLATOR

สมการของระบบนี้มีลักษณะเป็น

$$\dot{x} = -y/\varepsilon$$

$$\dot{y} = \varepsilon[x - (y^3/3 - y)]$$

ในที่นี้จะทำการทดลองค่า ε จำนวน 3 ค่า คือ 0.05, 0.5 และ 30 ตามลำดับ

สำหรับค่า $\varepsilon = 0.05$ ระบบเกิดการแกว่งด้วยแอมพลิจูดค่อนข้างคงที่(ลดลงด้วยอัตราน้อยมาก)

สำหรับค่า $\varepsilon = 0.5$ ระบบเกิดพฤติกรรมของลิมิตไซเคิล(limit cycle)ซึ่งกราฟปิดเส้นนี้จะมีผลเฉลยที่เป็นคาบ(periodic solution)ของระบบ กล่าวคือเมื่อเงื่อนไขเริ่มต้นของระบบอยู่บนกราฟเส้นนี้จะทำให้เกิดการแกว่งด้วยขนาดคงที่ ส่วนค่าเงื่อนไขเริ่มต้นที่อยู่ด้านในและด้านนอกของกราฟปิดนี้จะมีพฤติกรรมลู่เข้าหาผลเฉลยที่เป็นคาบนี้

หมายเหตุ สำหรับสมการนี้จะมีผลการรันบนระนาบของเวลาด้วย ขอให้ผู้อ่านทำการเปรียบเทียบผลทั้งสองนี้ด้วย

สำหรับค่า $\varepsilon = 30$ จะเกิดคำตอบที่เกือบเป็นเส้นตรง ลักษณะกรณีนี้สามารถพิจารณาราวๆ ดังนี้

พิจารณาสมการ

$$\dot{x} = -\frac{1}{\varepsilon}y$$

$$\dot{y} = \varepsilon[x - (\frac{1}{3}y^3 - y)]$$

สังเกตเมื่อค่า ε มีค่ามากจะทำให้ $\dot{x}$

มีค่าน้อยลงจึงทำให้เส้นกราฟมีการเปลี่ยนแปลงในทิศทาง x น้อยมาก เส้นกราฟจึงเคลื่อนที่ในแนวตั้ง แต่เมื่อค่า y เปลี่ยนแปลงจนทำให้สมการ

$$x - (\frac{1}{3}y^3 - y) = 0$$

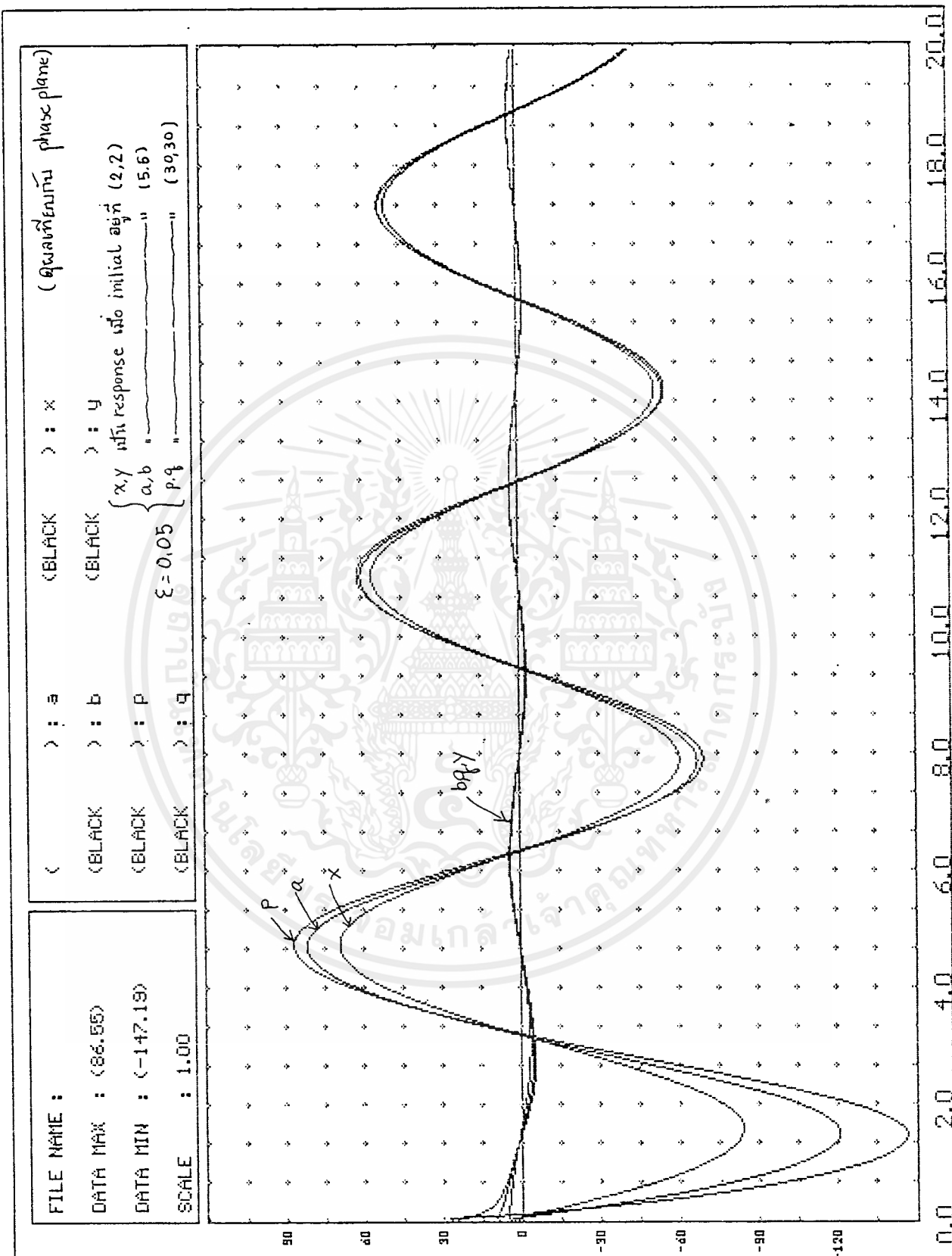
จะทำให้ค่า y มีการเปลี่ยนแปลงน้อยลงนั่นคือหากระบบมีจุดเริ่มต้นบนกราฟเส้นนี้จะทำให้พฤติกรรมของระบบวิ่งตามกราฟเส้นนี้เข้าหาจุดเริ่มต้นนั่นเอง

```

simulate
#
# Van der pol Oscillator
#
# .
#  $x = -(1/e)*y$ 
# .
#  $y = e*(x-(y^3/3-y))$ 
#
#
width 600
time 20.0
precision 20
variable asbsxsyspsqs
record abpqxy
#
# case 1=>0.05 // Colon is the Reserved Character for Line index
initial a5b5p10q10x2y2
equation
a=-1/0.05*b
b=0.05*(a-(b^3/3-b))
x=-1/0.05*y
y=0.05*(x-(y^3/3-y))
p=-1/0.05*q
q=0.05*(p-(q^3/3-q))
endequation
run
plot
#
# case 2=>0.5
initial a3b0p20q20x-0.1y-0.1
equation
a=-1/0.5*b
b=0.5*(a-(b^3/3-b))
x=-1/0.5*y
y=0.5*(x-(y^3/3-y))
p=-1/0.5*q
q=0.5*(p-(q^3/3-q))
endequation
run
plot
#
# case 3=> 30
initial a0.5b0p0q-0.5x2y-4
time 100
equation
a=-1/30*b
b=30*(a-(b^3/3-b))
x=-1/30*y
y=30*(x-(y^3/3-y))
p=-1/30*q
q=30*(p-(q^3/3-q))
endequation
run
plot
endprogram

```

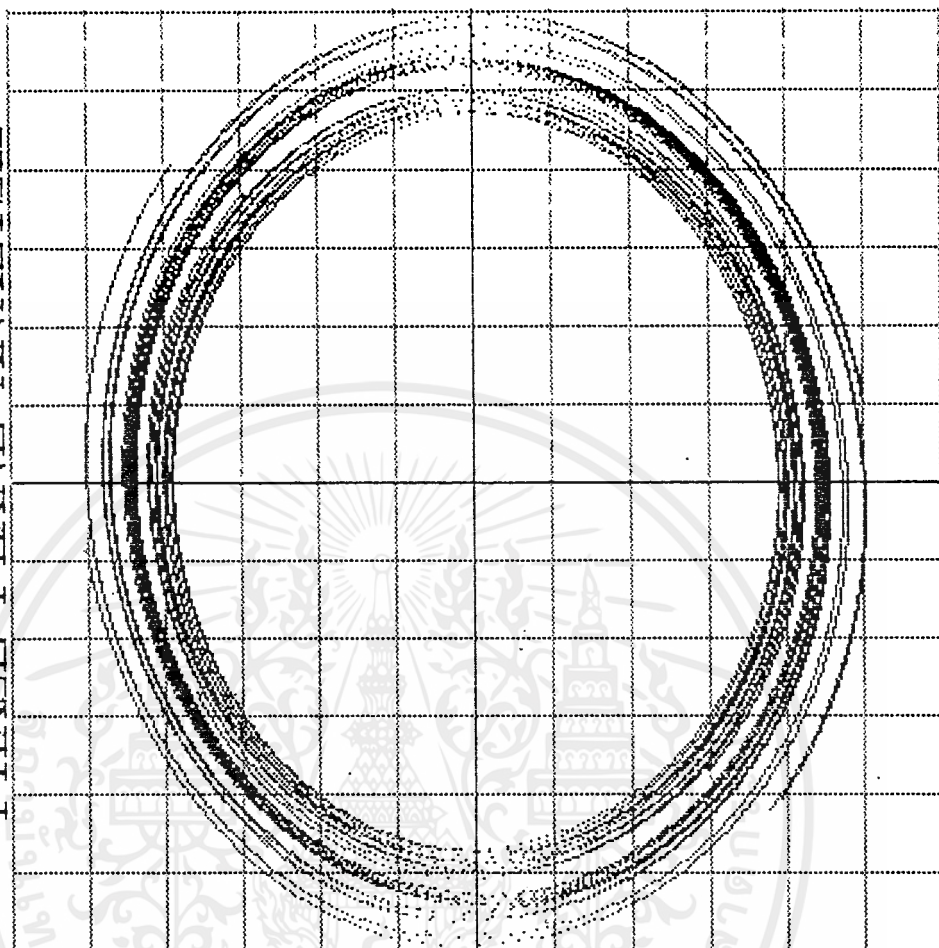
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



(101)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

PHASE PLANE ANALYSIS



EQUATION 1 :

$$\dot{x} = -1/0.05 * y$$

EQUATION 2 :

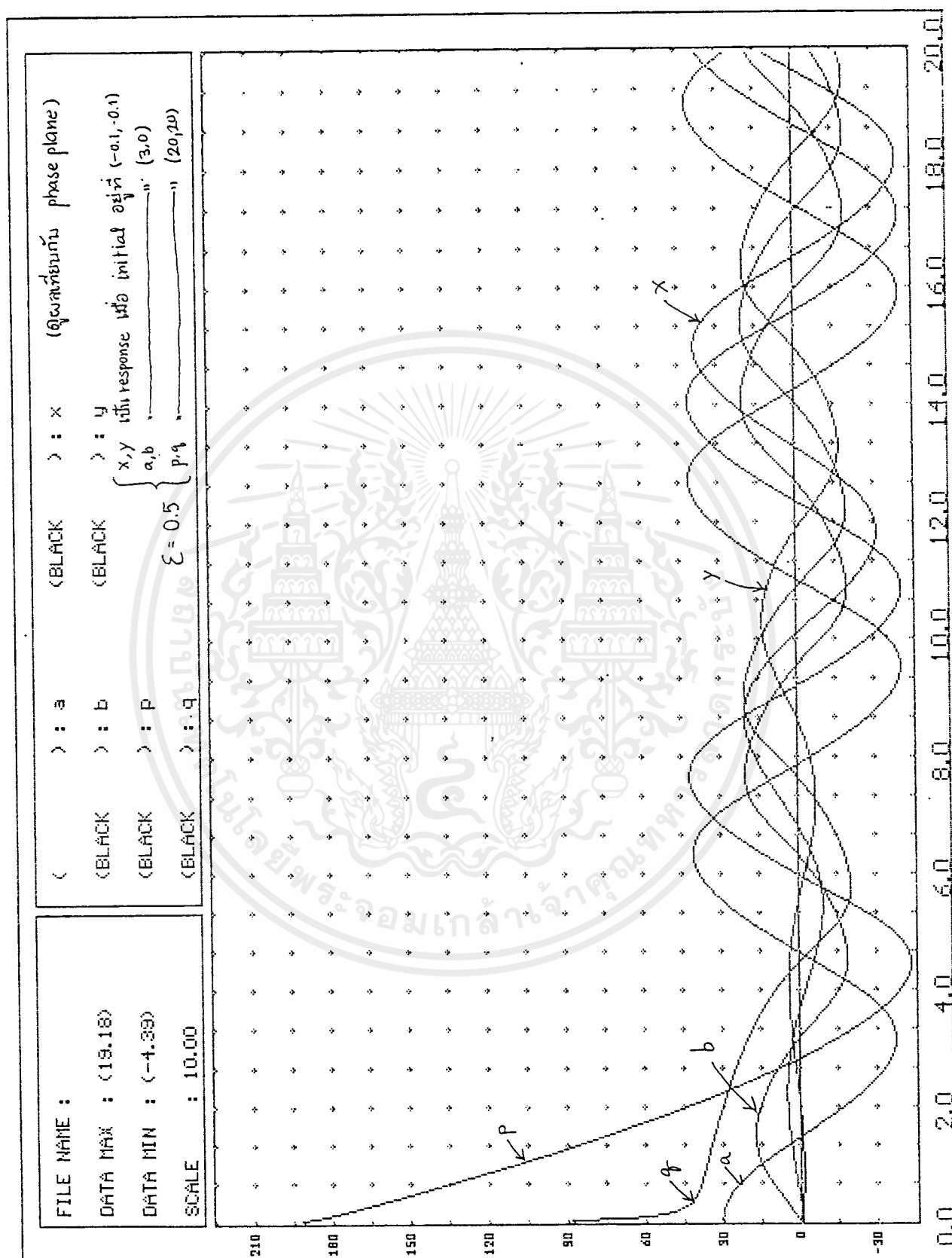
$$\dot{y} = 0.05 * (x - y^{3/3+y})$$

RANGE OF X-AXIS :

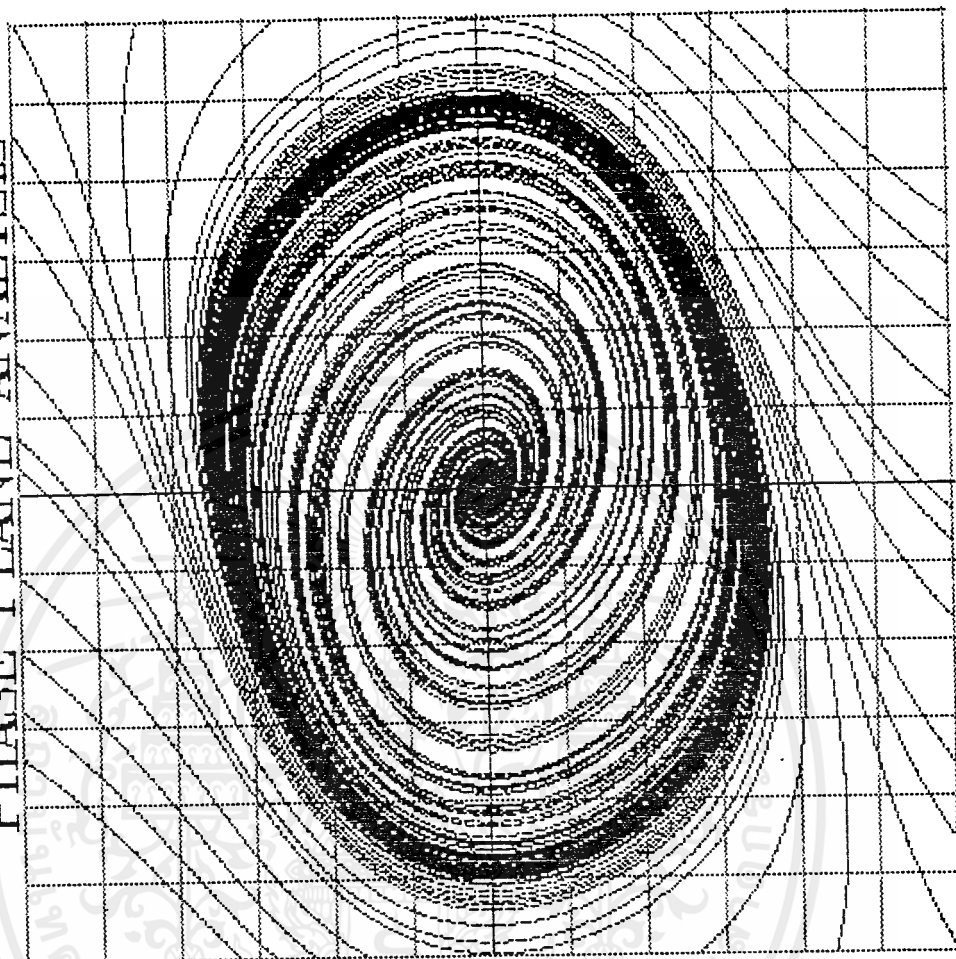
$\langle -50.00, 50.00 \rangle$

RANGE OF Y-AXIS :

$\langle -3.00, 3.00 \rangle$



PHASE PLANE ANALYSIS



EQUATION 1 :

$$\dot{x} = -1/0.5*y$$

EQUATION 2 :

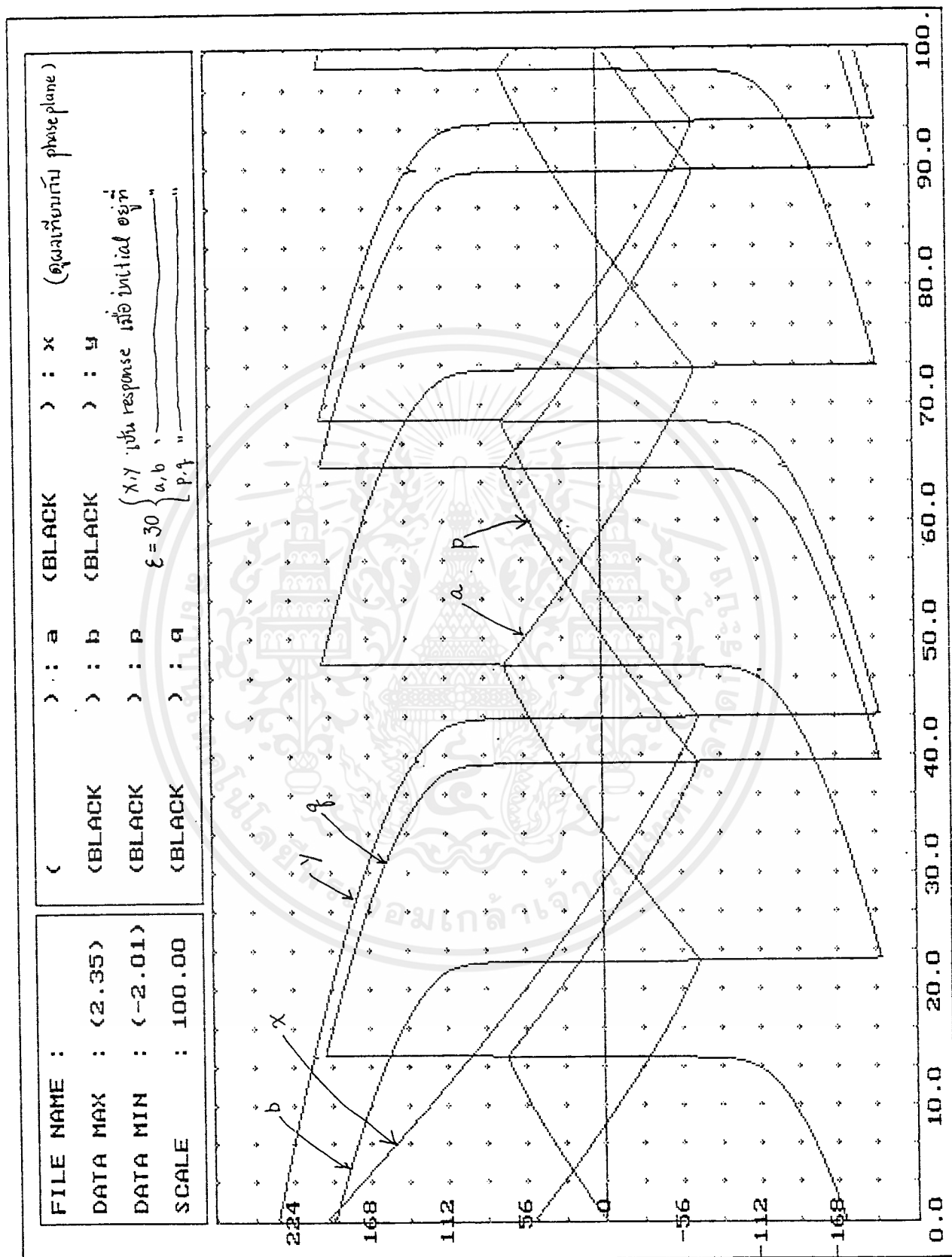
$$\dot{y} = 0.5*(x-y^3/3+y)$$

RANGE OF X-AXIS :

$(-5.00, 5.00)$

RANGE OF Y-AXIS :

$(-3.50, 3.50)$



PHASE PLANE ANALYSIS



EQUATION 1 :

$$\dot{x} = -1/30 * y$$

EQUATION 2 :

$$\dot{y} = 30 * (x - y^3/3 + y)$$

RANGE OF X-AXIS :

$(-5.00, 5.00)$

RANGE OF Y-AXIS :

$(-3.50, 3.50)$

ตัวอย่างโปรแกรมแสดงการประยุกต์ใช้ทางด้านอิเล็กทรอนิกส์กำลัง(Power electronics)

ในที่นี้จะทำการวิเคราะห์การแปลงไฟฟ้ากระแสตรงเป็นไฟฟ้ากระแสสลับหรือที่เรานิยมเรียกว่าวงจรอินเวอร์เตอร์ซึ่งสามารถเปลี่ยนแปลงหรือควบคุมระดับแรงดันไฟฟ้าและความถี่ของไฟฟ้ากระแสสลับได้ โดยทั่วไปเราสามารถแบ่งวงจรอินเวอร์เตอร์ออกเป็น 2 ชนิดคือ

- 1) อินเวอร์เตอร์ชนิดหนึ่งเฟส
- 2) อินเวอร์เตอร์ชนิดสามเฟส

การควบคุมแรงดันเอาต์พุตและปรับปรุงให้แรงดันเอาต์พุตมีฮาร์โมนิกส์น้อยมากจะมีวิธีต่างๆ หลายวิธี วิธีหนึ่งที่นิยมใช้กันคือ Pulse-Width Modulation (PWM) โดยวิธีการนี้จะเป็นการควบคุมให้อุปกรณ์ทางด้าอิเล็กทรอนิกส์กำลังมีการนำกระแสและหยุดนำกระแส เพื่อให้ได้รูปคลื่นของแรงดันเอาต์พุตเปลี่ยนแปลงตามความกว้างของพัลส์ วิธีการทำให้รูปคลื่นของแรงดันเอาต์พุตเป็นพัลส์ขนาดที่ถูกต้อง เราจะใช้คลื่นรูปไซน์ (sine wave) เป็นรูปคลื่นอ้างอิงที่มีความถี่ที่ต้องการ เปรียบเทียบกับรูปคลื่นสามเหลี่ยม (triangle wave) โดยจุดที่เป็นการตัดกันของสองลูกคลื่นจะเป็นสถานะที่มีการสร้างสัญญาณไปทริกอุปกรณ์ทางอิเล็กทรอนิกส์กำลัง

รูปคลื่น PWM ที่ถูกนำมาประยุกต์ใช้โดยทั่วไปได้แก่

- 1) รูปแบบที่เป็นไบโพลาร์ (bi-polar) จะเป็นรูปคลื่นที่มีการสวิตช์ระหว่างแรงดันระดับอ้างอิง 2 ระดับ คือ $+E$ และ $-E$
- 2) รูปแบบที่เป็นยูนิโพลาร์ (uni-polar) จะเป็นรูปคลื่นที่มีการสวิตช์ระหว่างแรงดันระดับอ้างอิง 3 ระดับ คือ $+E$, 0 และ $-E$

กราฟรูปแรก เป็นผลการรันของวงจรอินเวอร์เตอร์เฟสเดียว (Single phase inverter) โดยกรณีนี้มีลักษณะของการเปรียบเทียบเป็นแบบยูนิโพลาร์และไบโพลาร์ ดังที่เห็นได้จากรูปคลื่น PWM สองเส้นจากกราฟ

เนื่องจากรูปคลื่น PWM นอกจากจะประกอบไปด้วยความถี่หลักมูลแล้วยังประกอบไปด้วยองค์ประกอบทางฮาร์โมนิคต่าง ๆ ที่มีความถี่สูง ดังนั้นจึงจำเป็นต้องทำการวิเคราะห์องค์ประกอบฮาร์โมนิคของรูปคลื่น PWM ที่ได้สร้างขึ้น เพื่อประโยชน์ในการออกแบบวงจรกรองความถี่ (filter) เพื่อขจัดฮาร์โมนิคต่าง ๆ ออกไปเหลือแต่รูปไซน์ของความถี่หลักมูลเพียงความถี่เดียวเท่านั้น ซึ่งเราสามารถวิเคราะห์ถึงกรณีดังกล่าวมานี้ได้โดยการใช้คำสั่ง FFT ในขั้นตอนของการป้อนโปรแกรม ผลการรัน

แสดงได้ดังกราฟรูปถัดไป(รูปนี้เป็นกราฟวิเคราะห์สเปกตรัมของสัญญาณ PWM สำหรับกรณีของ ยูนิโพลาร์)

นอกจากนี้การวิเคราะห์สเปกตรัมของสัญญาณ PWM ยังมีประโยชน์ในด้านอื่น ๆ อีกกล่าวคือ จะเป็นข้อมูลสำหรับช่วยในการการออกแบบวงจรอินเวอร์เตอร์ว่าควรจะใช้ค่าอัตราส่วนความถี่เท่าใด จึงจะเหมาะสมและสามารถกำจัดฮาร์โมนิคต่ำ ๆ ออกไปได้

กราฟถัดไปเป็นผลการรันของวงจรอินเวอร์เตอร์สามเฟส(Three phase inverter) สำหรับ สัญญาณ PWM ในกรณีนี้จะเป็นแบบยูนิโพลาร์โดยตัวแปรต่าง ๆ จะแทนสัญญาณต่าง ๆ ดังนี้

- r เป็นสัญญาณฟันเลื่อย
- a,b,c เป็นสัญญาณของไฟ 3 เฟส
- x,y,z เป็นสัญญาณ pwm ของไฟ 3 เฟส
- s เป็น V line-line ในที่นี้เป็นศักดากระหว่าง x และ y

การวิเคราะห์ต่าง ๆ จะมีลักษณะการวิเคราะห์เช่นเดียวกับกรณีอินเวอร์เตอร์เฟสเดียวดังกล่าวข้างต้น

```

simulate                                // This Reserved Word Will Always Used
                                        // in The First Line of Program.

# Program simulation for Pulse Width Modulation ( 1 Phase )
# The PWM Signal Will be Transformed by FFT,to Check its Spectrum.
# File Name : PWM.1
# Date      : 18 March 1993

1 :freemem                             // Free Memory That Allocated Before
                                        // And Set All Parameter To Default Value.
2 :showt 2                             // Reset Clock To Check Time Consuming.
3 :width 600                           // Set Number of Data to 600.
4 :time 10.0                           // Set Time Range of Time Domain.
                                        // The Precision Value Cannot Effect The System-
                                        // That Don't Have Diffential Equations.
5 :record amklz                        // Allcate Memoty For Record Variables.

define Ma=1.0                          // Amplitude Modulation Ratio.
define Mf=20                           // Frequency Modulation Ratio.
define AxisShift=5                     // This Constant Used for Shifting Up The Graph.

6 :equation                            // "time" is Previously Defined by User to be 10.
7 :p=time/Mf
8 :g=t/p/4.0
9 :b=(g%p)*sign(p/2.0-g%p)
10:a=10*2/p*(b+(p-g%p)*sign(g%p-p/2)-p/4)+AxisShift
11:m=Ma*5*sin(2*3.14159*t/10)+AxisShift
12:k=sign(m-a)-0.5
13:l=k+sign(m)-0.5
14:endequation
15:run                                 // Calculate All Equations Consecutively From
                                        // time=0 to time=Time Range.
16:plot                               // If no Arguments, The Program Will Plot All
                                        // Record Variables That The User Declares
                                        // Which are "a", "m", "l", "k" ( 4 Variables).
17:fft kxy                            // Take FFT of Signal "k" and Store The Component
                                        // of Spectrum in x and y ( x Store Real Part And
                                        // y Store Imagine Part.
18:assign z=sqrt(x^2+y^2)             // Assign Tha Modulus of Spectrum to Variable "z".
19:plot ##kz                          // Plot Graph for Only Variable "k" And "z".
20:ifft zxy 0 0.1                    // Reconstruction The Signal From The Spectrum x,y
                                        // in The Range of 0-10% of Frequency Axis.
21:return                             // When Reach "return" for First Time , Nothing
                                        // Happen Because The Program Will Skip This Line.

define Ma=0.6                          // Redefine Amplitude Modulation Ratio.

22:goto 6                             // Goto "equation" Command. (in Line 6)
                                        // The Line No. not Embrace "define", Comment Line
                                        // and "Simulate". (These Lines Have No Effect to
                                        // Count Line No.)
                                        // Notice That "endequation" Will be Counted ,Too.
23:showt 1                             // Show The Time Consumption of Program.
24:endprogram                         // End Program.
                                        // This Reserved Word Will Always Used
                                        // in The Last Line of Program.

;

/***** Executable Program *****/
/***** Compile & Run : OK *****/

simulate
width 600
time 10.0
precision 1
record amklz

```

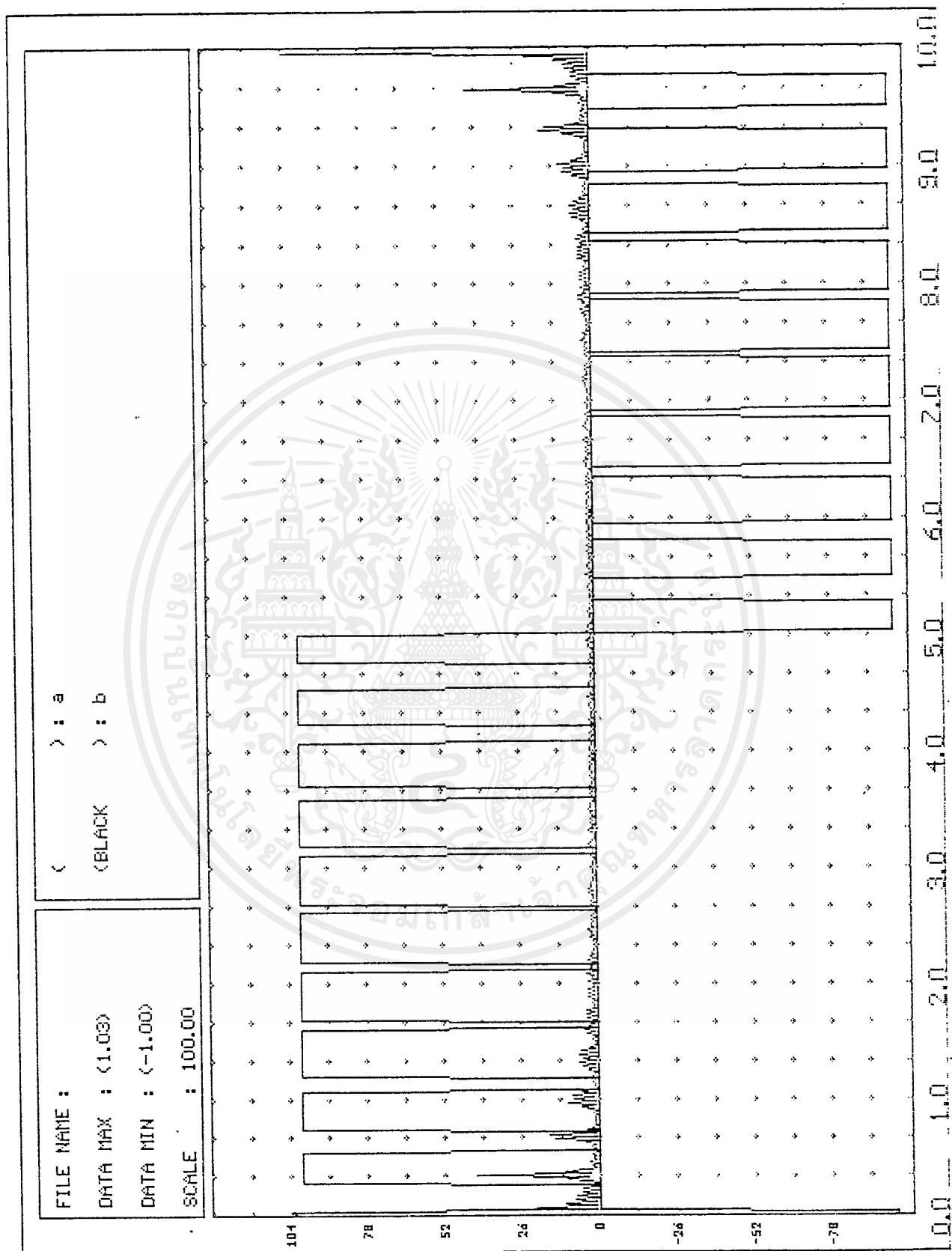
```

equation
p=0.5001
g=t/p/4.0
b=(g%p)*sign(p/2.0-g%p)
a=10*2/p*(b+(p-g%p)*sign(g%p-p/2)-0.25*p)
m=0.8*5*sin(2*3.14159*t/10)
k=sign(m-a)-0.5
l=k+sign(m)-0.5
endequation
run
plot
fft lxy
assign z=sqrt(x^2+y^2)
plot ##lz
ifft zxy 0.0 0.1
plot ##lz
endprogram

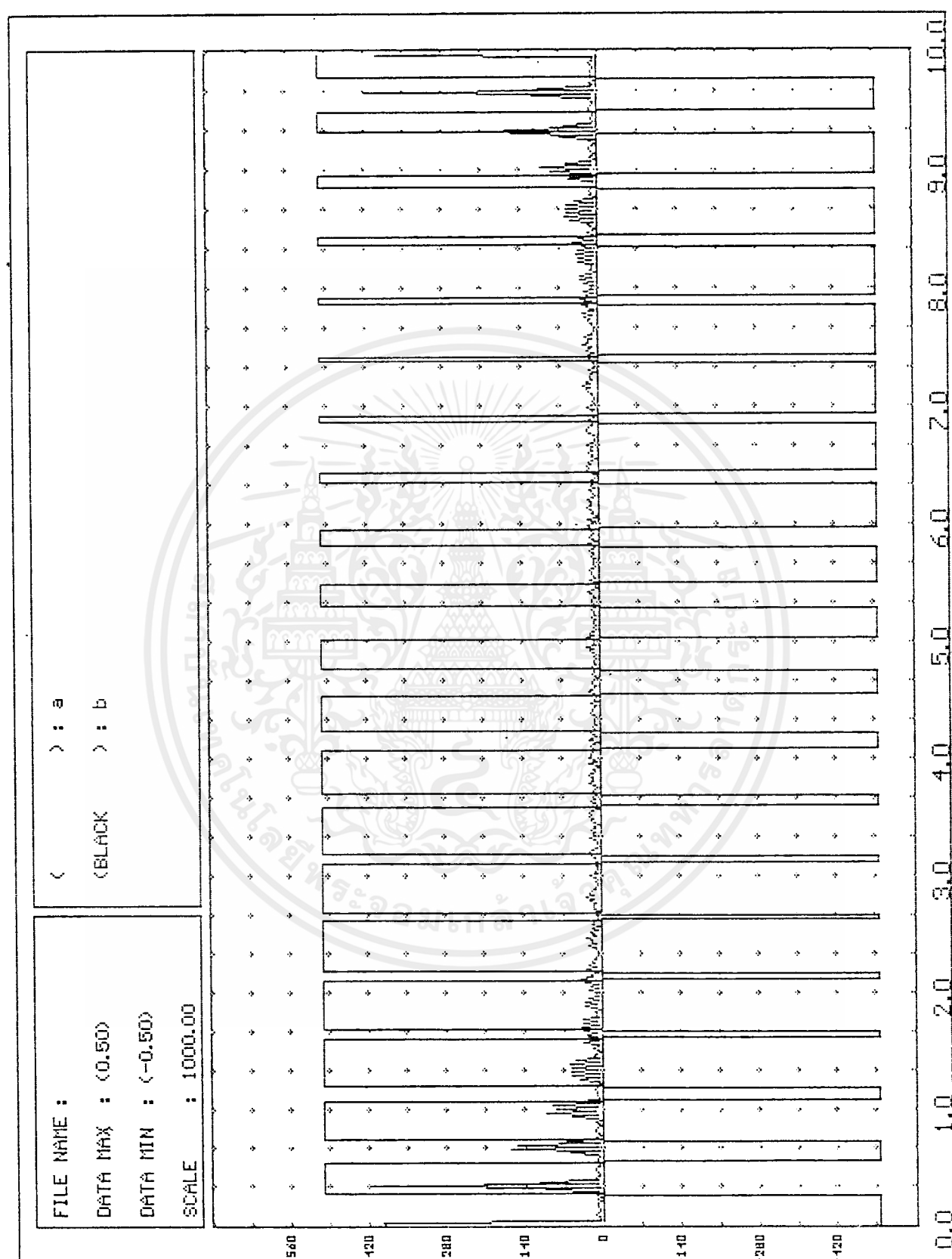
```



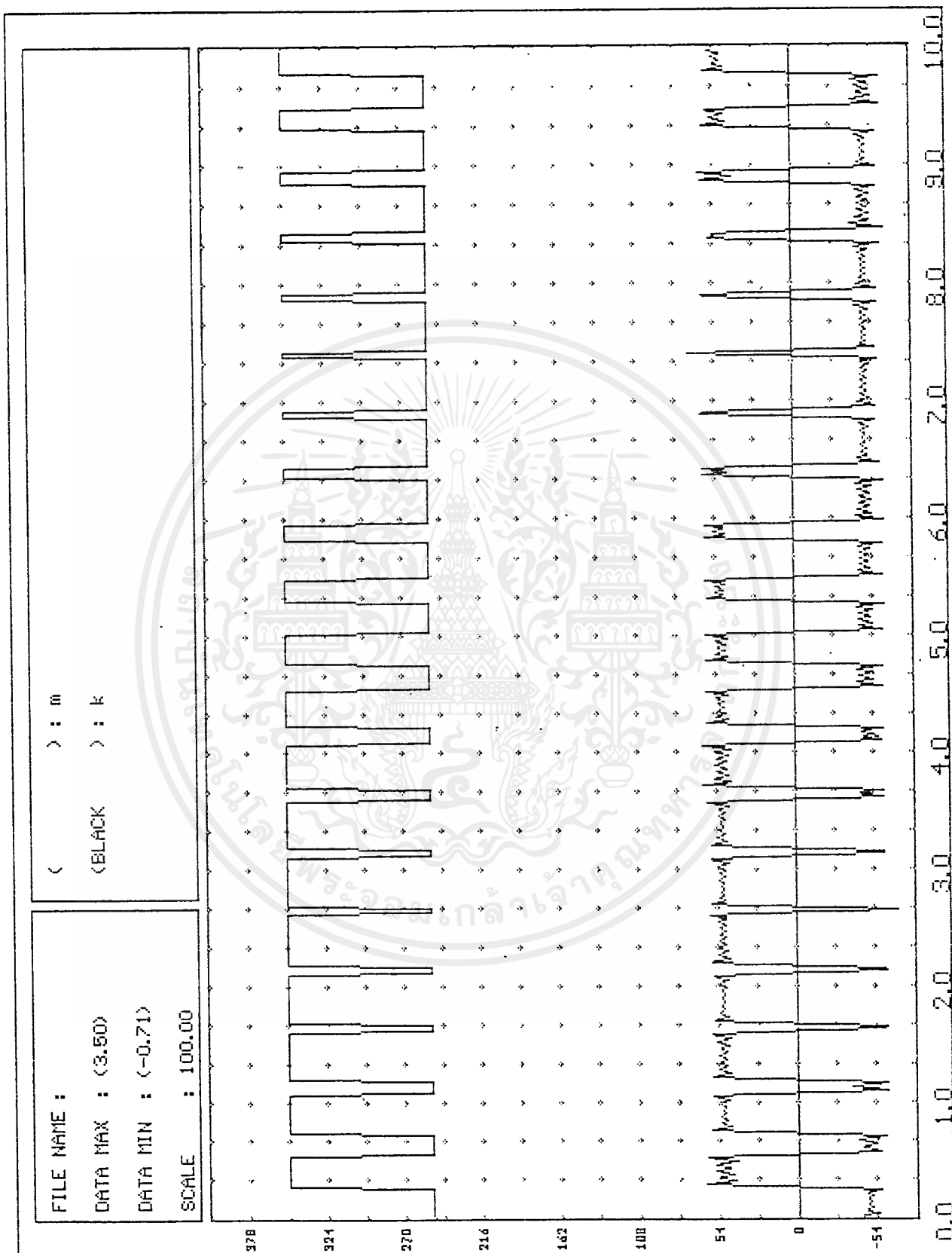
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (110) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (lii) เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



```

simulate                                     // This Reserved Word Will Always Used
                                           // in The First Line of Program.

# Program simulation for Pulse Width Modulation ( 3 Phase )
# The PWM Signal Will be Transformed by FFT,to Check its Spectrum.
# File Name : PWM_3P.1
# Date      : 18 March 1993

1 :freemem                                // Free Memory That Allocated Before
                                           // And Set All Parameter To Default Value.
2 :showt 2                                // Reset Clock To Check Time Consuming.
3 :width 600                              // Set Number of Data to 600.
4 :time 6.28                              // Set Time Range of Time Domain.
                                           // The Precision Value Cannot Effect The System-
                                           // That Don't Have Differential Equations.
5 :record rabcxys                         // Allcate Memoty For Record Variables.

define Ma=0.8                             // Amplitude Modulation Ratio.
define Mf=20                              // Frequeuncy Modulation Ratio.
define AxisShift=2                        // This Constant Used for Shifting Up The Graph.

6 :equation                               // "time" is Previously Defined by User to be 10.
7 :p=time/Mf
8 :g=t/p/4.0
9 :q=(g%p)*sign(p/2.0-g%p)
10:r=2*2/p*(q+(p-g%p)*sign(g%p-p/2)-p/4)+5*AxisShift
11:a=Ma*sin(t)+5*AxisShift
12:b=Ma*sin(t-2*3.1415/3)+5*AxisShift
13:c=Ma*sin(t-4*3.1415/3)+5*AxisShift
14:x=sign(r-a)-0.5+3*AxisShift
15:y=sign(r-b)-0.5+2*AxisShift
16:z=sign(r-c)-0.5+1*AxisShift
17:s=x-y-AxisShift
18:endequation
19:run                                     // Calculate All Equations Consecutively From
                                           // time=0 to time=Time Range.
20:plot                                   // If no Arguments, The Program Will Plot All
                                           // Record Variables That The User Declares.
21:fft sxy                               // Take FFT of Signal "s" and Store The Component
                                           // of Spectrum in x and y ( x Store Real Part And
                                           // y Store Imagine Part.)
22:assign z=sqrt(x^2+y^2)                // Assign Tha Modulus of Spectrum to Variable "z".
23:plot ##sz                             // Plot Graph for Only Variable "s" And "z".
                                           // "s" is Signal And "z" is its Spectrum(Modulus).
24:ifft zxy 0 0.1                        // Reconstruction The Signal From The Spectrum x,y
                                           // in The Range of 0-10% of Frequency Axis.
                                           // The Reconstructed Signal Will be Stored in "z".
25:plot ##sz                             // "s" is Signal And "z" is Reconstructed Signal
                                           // From 10% of Spectrum Range of "s"
26:return                                // When Reach "return" for First Time , Nothing
                                           // Happen Because The Program Will Skip This Line.

define Ma=0.6                             // Redefine Amplitude Modulation Ratio.
27:goto 6                                // Goto "equation" Command. (in Line 6)
                                           // The Line No. not Embrace "define", Comment Line
                                           // and "Simulate". (Because These Lines Have No
                                           // Effect in Counting Line No.)
                                           // Notice That "endequation" Will be Counted.
28:showt 1                                // Show The Time Comsumption of Program.
29:endprogram                             // End Program.
                                           // This Reserved Word Will Always Used
                                           // in The Last Line of Program.

```

```

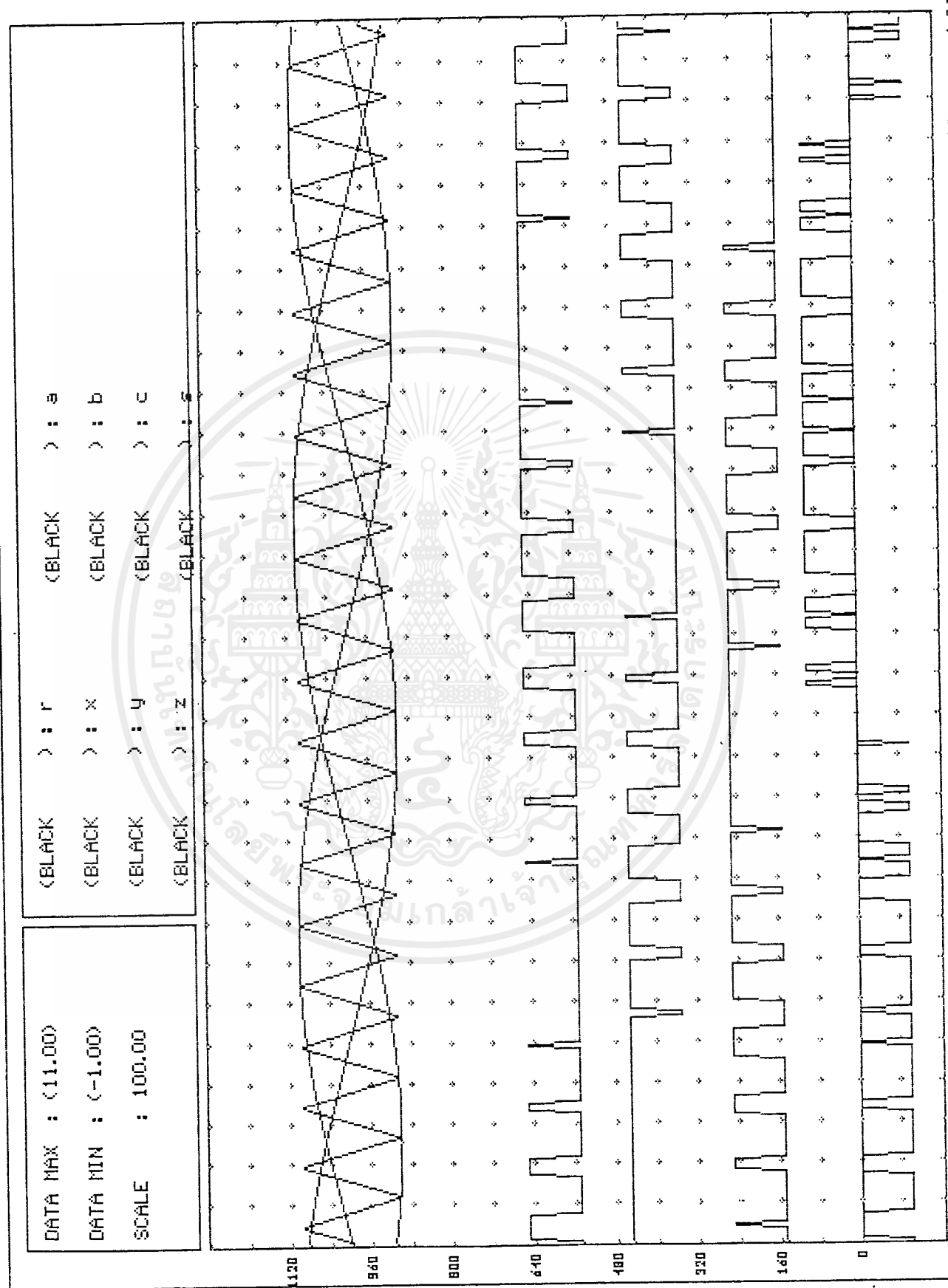
#/****** Executable Program *****/
#/****** Compile : OK *****/
#/****** Run : OK *****/

```

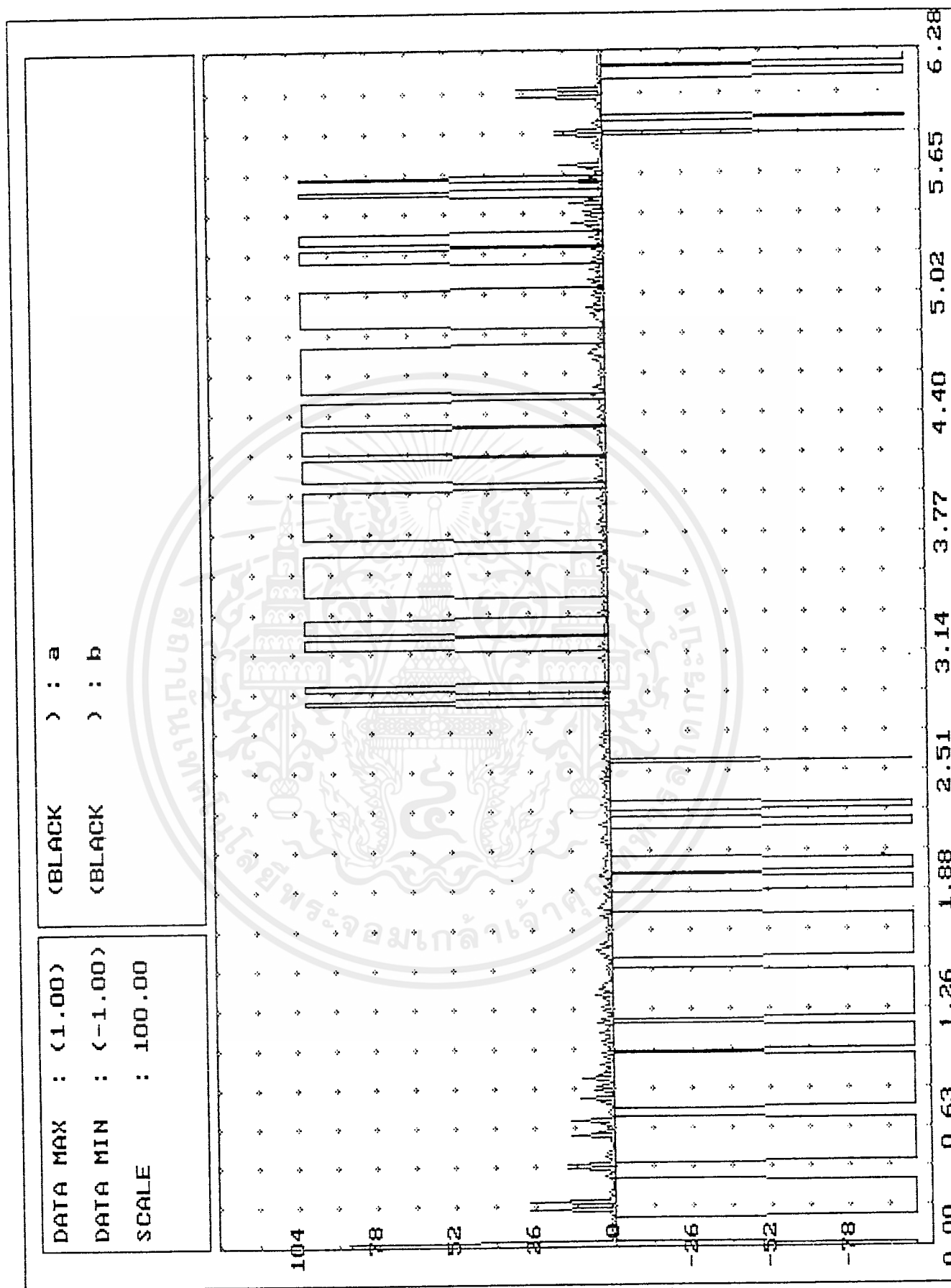
```

simulate
1 :freemem
2 :showt 2
3 :width 600
4 :time 6.28
5 :record rxyzabcs
define Ma=0.8
define Mf=20
define AxisShift=2
6 :equation
7 :p=6.28/Mf
8 :g=t+p/4.0
9 :q=(g*p)*sign(p/2.0-g*p)
10:r=2*2/p*(q+(p-g*p)*sign(g*p-p/2)-p/4)+5*AxisShift
11:a=Ma*sin(t)+5*AxisShift
12:b=Ma*sin(t-2*3.1415/3)+5*AxisShift
13:c=Ma*sin(t-4*3.1415/3)+5*AxisShift
14:x=sign(r-a)-0.5+3*AxisShift
15:y=sign(r-b)-0.5+2*AxisShift
16:z=sign(r-c)-0.5+1*AxisShift
17:s=x-y-AxisShift
18:endequation
19:run
20:plot
21:fft s
22:showt 1
endprogram

```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษานี้เท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ตัวอย่างโปรแกรมแสดงการประยุกต์ใช้ในทางด้านไฟฟ้าสื่อสาร

การติดต่อสื่อสาร หากเราส่งสัญญาณต่าง ๆ ออกไปโดยตรงจะเกิดปัญหาต่างๆ หลายประการ เช่น สัญญาณอาจมีกำลังไม่สูงพอที่จะเดินทางได้ไกลๆ ตลอดจนอาจเกิดปัญหาทางด้านสัญญาณรบกวน เป็นต้น ปัญหาเหล่านี้เราสามารถแก้ไขได้โดยการใช้สัญญาณความถี่สูงเป็นพาหะเพื่อนำสัญญาณที่เราจะส่งไปด้วย วิธีการนี้เราเรียกว่า การมอดูเลชัน (modulation)

ในขบวนการมอดูเลตโดยทั่วไป เราใช้คลื่นรูปไซน์ที่มีความถี่สูงเป็นพาหะ แล้วทำการเปลี่ยนแปลงคุณสมบัติของสัญญาณนี้ด้วยสัญญาณที่เราต้องการจะส่ง

โปรแกรมแรก เป็นตัวอย่างของการมอดูเลตทางแอมพลิจูด (amplitude modulation) กรณีนี้การทำการมอดูเลตจะมีลักษณะที่ตรงไปตรงมากล่าวคือ จะเป็นการคูณสัญญาณโมดูเลตกับคลื่นพาหะโดยตรง สำหรับตัวอย่างนี้กำหนดให้

ข้อมูลเป็น	$\sin t$
สัญญาณพาหะเป็น	$\sin(30t)$
สัญญาณโมดูเลตที่ได้คือ	$a = \sin(t) \sin(30t)$

จากความรู้ทางคณิตศาสตร์เราทราบว่าผลคูณของสัญญาณที่เป็นไซน์จะเป็นผลรวมของสัญญาณที่เป็นโคไซน์สองลูกคลื่น ฉะนั้นเมื่อเราวิเคราะห์สเปกตรัมของสัญญาณนี้จึงเห็นแกนแนวดิ่งที่เป็นขนาดของแอมพลิจูด ณ ความถี่สองแห่งนั่นเอง

โปรแกรมที่สอง เป็นตัวอย่างของการมอดูเลตทางความถี่ กรณีนี้สัญญาณที่เป็นข้อมูลจะเข้าไปมอดูเลตลงบนสัญญาณพาหะสัญญาณที่ได้จากการมอดูเลต จะมีลักษณะแสดงดังกราฟ กรณีนี้หากสัญญาณที่เป็นข้อมูลมีค่าของขนาดสูงสุด สัญญาณที่ได้จากการมอดูเลตจะมีความถี่สูงสุด หากสัญญาณที่เป็นข้อมูลมีค่าของขนาดต่ำสุด สัญญาณที่ได้จากการมอดูเลตจะมีความถี่ต่ำสุด นั่นคือเราสามารถสรุปลักษณะของการมอดูเลตทางความถี่ได้ว่า สัญญาณที่ได้จะมีขนาดคงที่ตลอด แต่ความถี่จะเปลี่ยนแปลงตามสัญญาณของข้อมูล

ในปัจจุบัน วิทยาการทางด้านดิจิตอลอิเล็กทรอนิกส์ได้มีความสำคัญและพัฒนาขึ้นอย่างมาก ฉะนั้นการส่งข้อมูลที่เป็นดิจิตอลจึงมีความสำคัญเพิ่มขึ้นเป็นลำดับ นอกจากนี้การส่งข้อมูลที่เป็นดิจิตอลจะมีข้อดีเมื่อเปรียบเทียบกับ การส่งข้อมูลที่เป็นอนาล็อก กล่าวคือกรณีการส่งสัญญาณอนาล็อกทางภาครับสัญญาณ จะต้องทำการแยกสัญญาณข้อมูล ออกจากสัญญาณรบกวน โดย

สัญญาณที่ได้จะต้องมีทั้งขนาดและความถี่เหมือนกับสัญญาณข้อมูลทุกประการ การส่งสัญญาณจึงถูกต้อง ในขณะที่การส่งสัญญาณแบบดิจิตอลจะทำการตรวจจับเฉพาะระดับของสัญญาณเท่านั้น โอกาสที่จะเกิดการผิดพลาดจึงน้อยกว่า กล่าวคือจะเกิดการผิดพลาดก็ต่อเมื่อสัญญาณรบกวนมีผลทำให้ระดับสัญญาณเปลี่ยนแปลงจาก 0 ไปเป็น 1 หรือเปลี่ยนแปลงจาก 1 ไปเป็น 0 นั่นเอง

โปรแกรมที่สาม เป็นเทคนิคที่ใช้ในการมอดูเลตสัญญาณที่เป็นดิจิตอล ที่เรียกว่า Frequency Shift Keying (FSK) กรณีนี้การเปลี่ยนความถี่ของสัญญาณที่เป็นพาหะจะขึ้นกับสัญญาณข้อมูลที่เป็นดิจิตอล เช่น ช่วงความถี่สูงแทนข้อมูลดิจิตอลที่เป็น 1 ช่วงความถี่ต่ำแทนข้อมูลดิจิตอลที่เป็น 0 ซึ่งจากการวิเคราะห์สเปกตรัมเราจะเห็นกลุ่มของสเปกตรัมเป็น 2 ชุดอย่างชัดเจน

โปรแกรมที่สี่ เป็นเทคนิคที่ใช้ในการมอดูเลตสัญญาณที่เป็นดิจิตอลที่เรียกว่า Phase Shift Keying (PSK) กรณีนี้การเปลี่ยนเฟสของสัญญาณที่เป็นพาหะจะขึ้นกับสัญญาณข้อมูลที่เป็นดิจิตอล เช่น เฟสเป็น 0 แทนข้อมูลดิจิตอลที่เป็น 0 เฟสเป็น 180 แทนข้อมูลดิจิตอลที่เป็น 1 เป็นต้น



```

simulate
// This Reserved Word Will Always Used in The
// in The First Line of Program.

# Program simulation for Examining The Spectrum of Modulated Signal
# In Communication System. Including,
# (1) : Amplitude Modulation
# (2) : Frequency Modulation
# (3) : Frequency Shift Keying
# (4) : Phase Shift Keying (4PSK for this case)
# File Name : MODULATE.1
# Date : 18 March 1993

#REMARK: Line Number is Not Necessary in The File
#Because Command "goto" is not Used.
1 :freemem // Free Memory That Allocated Before
// And Set All Parameter To Default Value.
2 :showt 2 // Reset Clock To Check Time Consuming.
// (Start Timing)
3 :width 600 // Set Number of Data to 600.
4 :time 6.283 // Set Time Range of Time Domain.
// 6.283 = 2*PHI
// The Precision Value Cannot Effect The System-
// That Don't Have Differential Equations.
5 :record abcd // Allcate Memoty For Record Variables.

6 :equation
7 :a=sin(30*t)*sin(t) // Amplitude Modulation.
// sin(t) is DATA and sin(30*t) is CARRIER.
8 :b=sin(20*sin(t))+2 // Frequency Modulation.
9 :c=sin(20*t+40*sign(1-t%2)*t)+4 // Frequency Shift Keying.
// The Carrier Will Oscillate With 2 Different
// Frequency. ( 20Hz. And 60 Hz.)
10:d=sin(4*t+1.57*(sign(t-1.571)+sign(t-3.1416)+sign(t-4.712)))+6
// 1.57=PHI/2
11:endequation

12:run // Calculate All Equations Consecutively From
// time=0 to time=Time Range.
// Remark: The System Equation Will be Orderd
// Correctly. (Depend on Order in Calculation)
13:plot // If no Arguments, The Program Will Plot All
// Record Variables That The User Declares
// Which are "a", "b", "c", "d" ( 4 Variables).

# Amplitude Modulation
14:fft axy // Take FFT of Signal "a" and Store The Component
// of Spectrum in x and y (x Stores Real Part And
// y Stores Imagine Part).
15:assign z=sqrt(x^2+y^2) // Assign Tha Modulus of Spectrum to Variable "z".
16:plot ##az // Plot Graph for Only Variable "k" And "z".

# Frequency Modulation
17:fft bxy
18:assign z=sqrt(x^2+y^2)
19:plot ##bz

# Frequency Shift Keying
20:fft cxy
21:assign z=sqrt(x^2+y^2)
22:plot ##cz

#Phase Shift Keying
23:fft dxy
24:assign z=sqrt(x^2+y^2)
25:plot ##dz

26:showt 1 // Show The Time Consumption of Program.ใช้ประโยชน์ด้านการค้า
// End Program.
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้าม // This Reserved Word Will Always Used ครั้งที่มีการนำไปใช้

```

// in The Last Line of Program.

endprogram

```
#/***** Executable Program *****/
#/**/ Compile&Run :OK *****/
```

simulate

1 :freemem

2 :showt 2

3 :width 600

4 :time 6.283

5 :record abcd

6 :equation

7 :a=sin(30*t)*sin(t)

8 :b=sin(20*sin(t))+2

9 :c=sin(20*t+40*sign(1-t%2)*t)+4

10:d=sin(4*t+1.57*(sign(t-1.571)+sign(t-3.1416)+sign(t-4.712)))+6

11:endequation

12:run

13:plot

#

fft axy

assign z=sqrt(x^2+y^2)

plot ##az

#

fft bxy

assign z=sqrt(x^2+y^2)

plot ##bz

#

fft cxy

assign z=sqrt(x^2+y^2)

plot ##cz

#

fft dxy

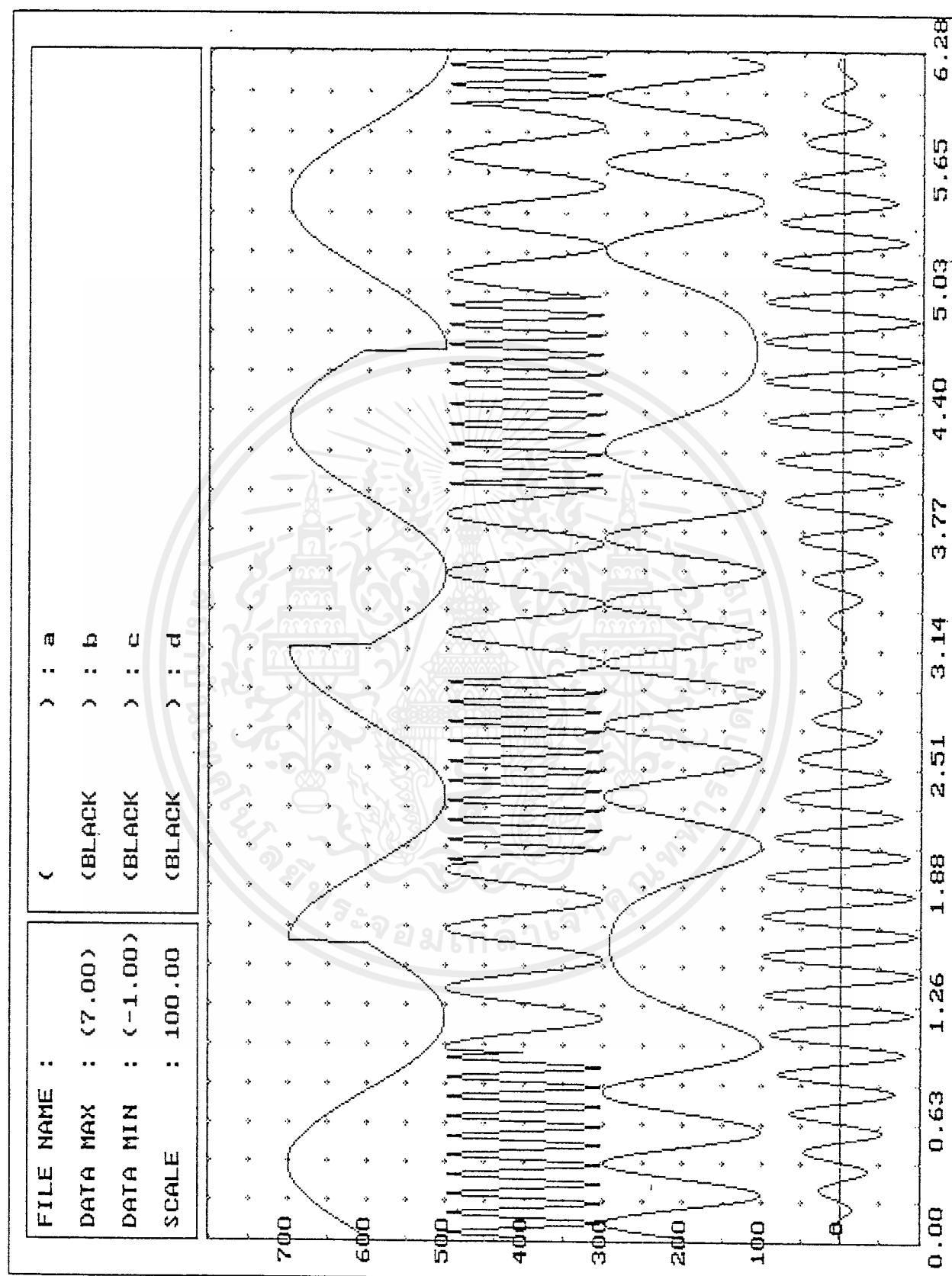
assign z=sqrt(x^2+y^2)

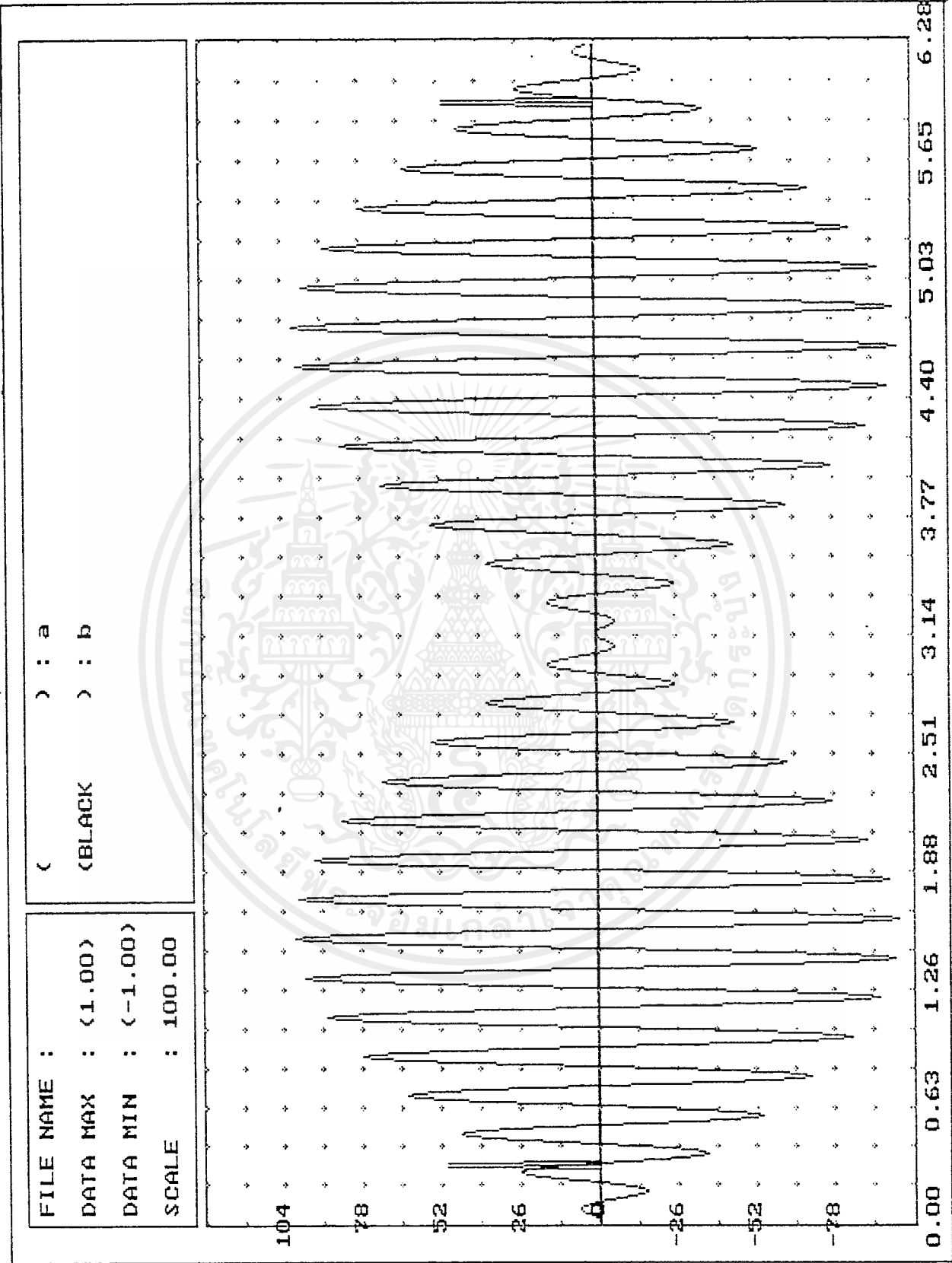
plot ##dz

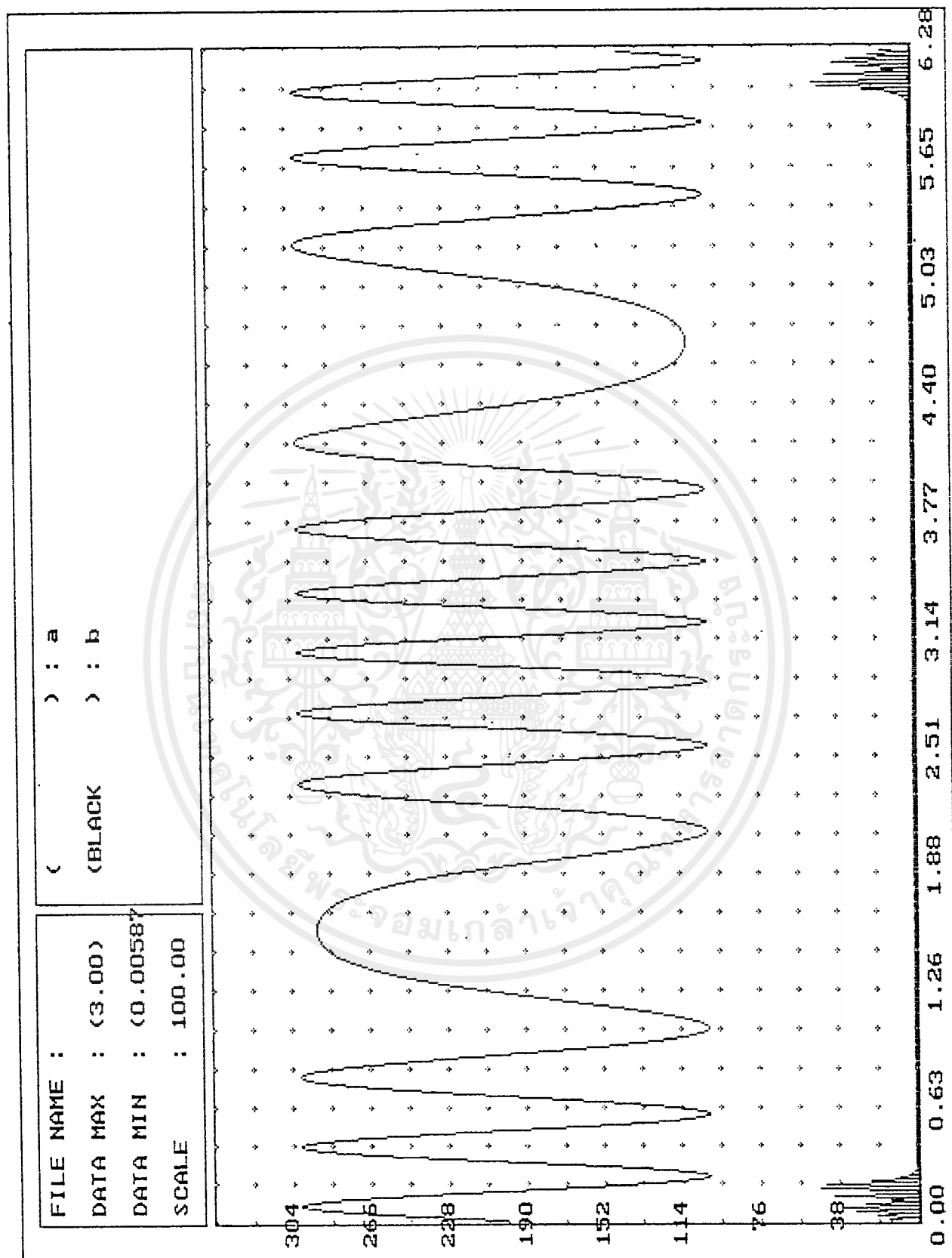
showt 1

endprogram

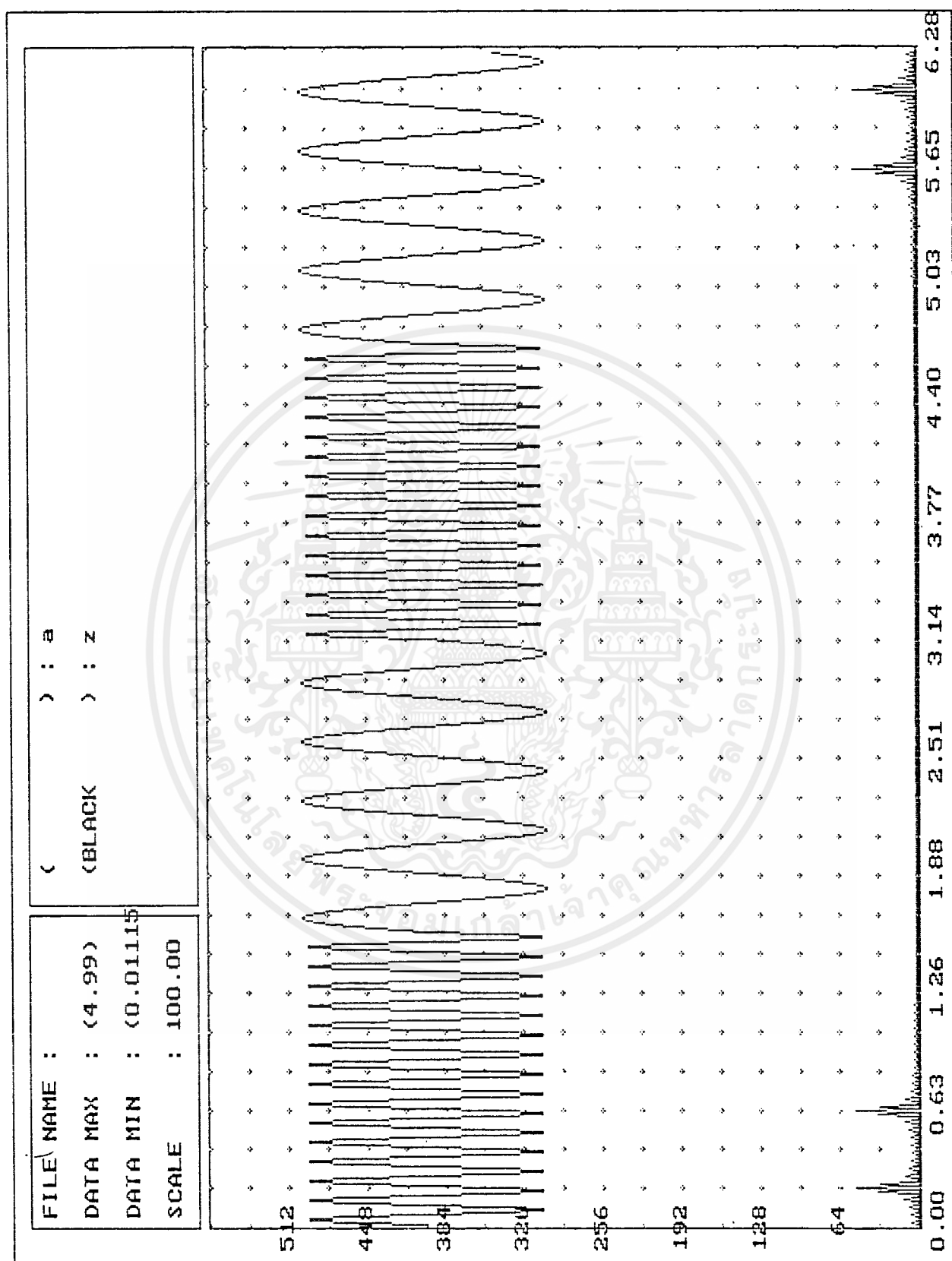
1



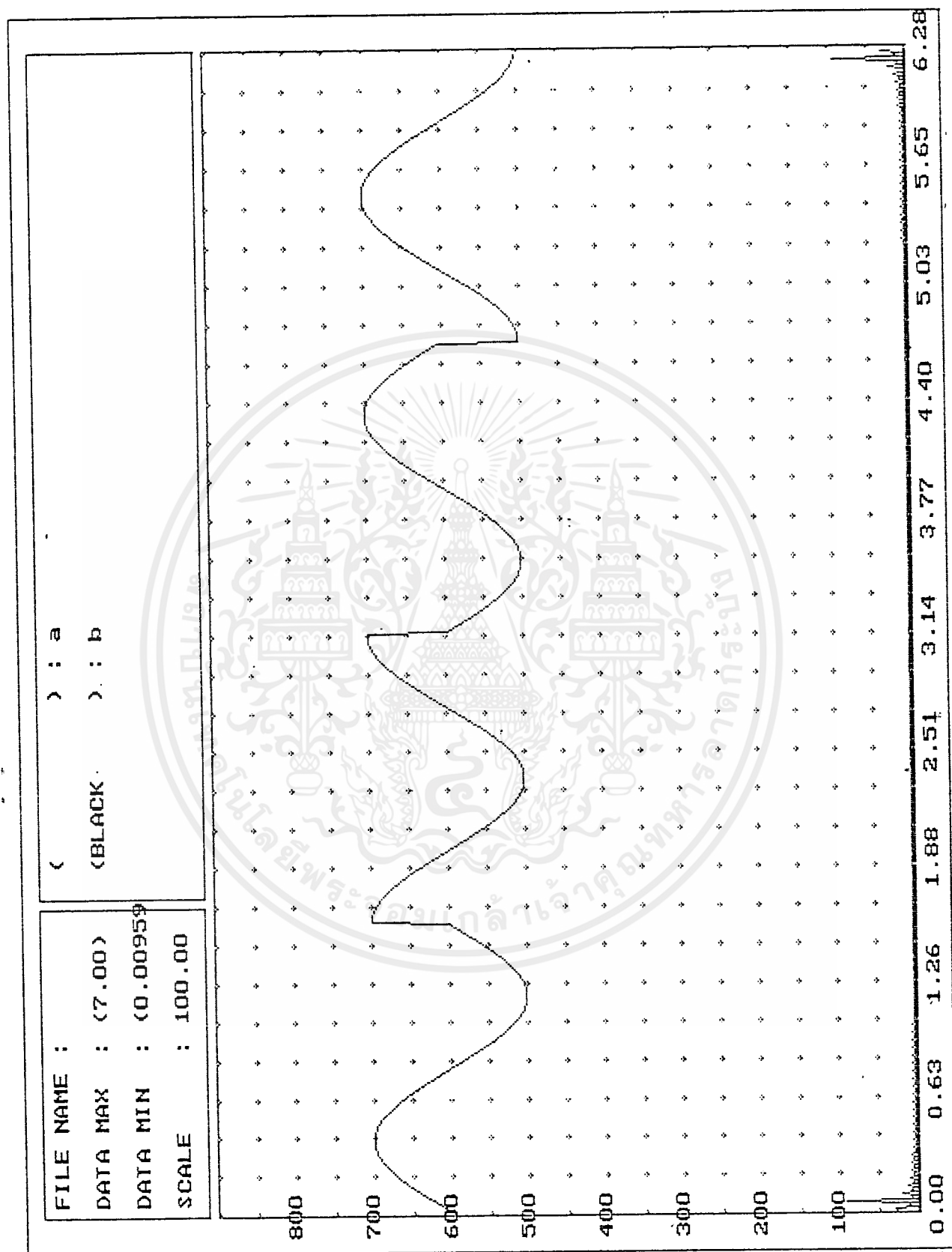




เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อ (125)
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



บทที่ 8

บทสรุปและวิจารณ์

1) การวิเคราะห์แบบเฟสเพลนวิธีนี้จะใช้ในการวิเคราะห์ระบบที่เป็นอันดับสอง ซึ่งเราสามารถพิจารณาถึงเสถียรภาพของระบบได้โดยคร่าว ๆ ซึ่งลักษณะเช่นนี้มักเป็นการวิเคราะห์ระบบที่ไม่มีอินพุต (free motion หรือ natural response หรือ homogenous solution)

2) วิธีการวิเคราะห์แบบเดสไคริ빙ฟังก์ชัน วิธีนี้จะใช้ในการวิเคราะห์ระบบที่เกี่ยวข้องกับอุปกรณ์ที่ไม่เป็นเชิงเส้น ซึ่งเกิดขึ้นจากตัวพลานท์ คอนโทรลเลอร์ หรือ ตัวเซนส์เซอร์วิธีนี้จะประมาณระบบที่ไม่เป็นเชิงเส้นให้เป็นระบบที่เป็นเชิงเส้นภายใต้ข้อสมมติฐานต่าง ๆ

3) การวิเคราะห์เสถียรภาพโดยวิธีลียาปูนอฟ จะประกอบด้วยวิธีการวิเคราะห์ สองวิธีคือ
- วิธีการประมาณเชิงเส้น (linearization method) วิธีนี้เราจะใช้ได้เฉพาะกรณีที่เมตริกซ์ระบบไม่ได้แสดงพฤติกรรมแบบ saddle point (ค่า eigenvalue มีเครื่องหมายต่างกัน)

- วิธีการโดยตรง (direct method) วิธีนี้จะทำการเลือก ฟังก์ชันลียาปูนอฟ (Lyapunov function) ที่เหมาะสมที่สามารถให้ข้อมูลเกี่ยวกับพฤติกรรมของระบบได้มากที่สุด กล่าวคือสามารถบอกพฤติกรรมครอบคลุมช่วงการทำงานของระบบที่เราสนใจ สำหรับเทคนิคการเลือกฟังก์ชันลียาปูนอฟ ขึ้นกับความซับซ้อนและความยุ่งยากของระบบนั้นๆ เช่น

ระบบที่มีคุณสมบัติเป็นเชิงเส้น (จะแปรตามเวลาหรือไม่ก็ได้) จะมีวิธีเลือกโดยการพิจารณาจากการแก้สมการลียาปูนอฟ (Lyapunov equation) ระบบที่มีคุณสมบัติไม่เป็นเชิงเส้น (จะแปรตามเวลาหรือไม่ก็ได้) จะมีเทคนิคการเลือกฟังก์ชันลียาปูนอฟ หลายวิธีซึ่งจะขึ้นกับลักษณะของสมการของระบบ ส่วนวิธีการเลือกจะไม่มีเทคนิคที่ตายตัวเหมือนกับระบบที่มีคุณสมบัติเป็นเชิงเส้น ซึ่งขึ้นกับความชำนาญของผู้ทำการวิเคราะห์

4) การเลือกวิธีควบคุมระบบจะต้องทำการวิเคราะห์อย่างละเอียดเพราะหลักการต่างๆ ที่ใช้ในปัจจุบันยังไม่อาจใช้ได้ครอบคลุมกับทุกระบบ ในส่วนของปริญาณิพนธ์ฉบับนี้ได้เลือกเอาหลักการควบคุมแบบปรับตัวโดยวิธีคอมโพสิตซึ่งรวมหลักการควบคุมแบบมีโมเดลอ้างอิง กับหลักการประมาณค่าพารามิเตอร์เข้าด้วยกันซึ่งทำให้ระบบมีผลตอบสนองเป็นไปตามโมเดลที่เราต้องการได้ดียิ่งขึ้น เพราะกฎการปรับตัวที่ไม่ได้รวมการประมาณค่าพารามิเตอร์ จะทำให้มีการปรับค่าพารามิเตอร์เฉพาะในช่วงที่มีการเปลี่ยนแปลงสัญญาณอินพุตเท่านั้นและจะทำการปรับค่าพารามิเตอร์ไปจนกว่าเอาต์พุตของพลานท์จะเท่ากับของโมเดลอ้างอิง และจะหยุดปรับเมื่อเข้าสู่สถานะคงตัว (steady state) จนกว่า

จะมีการเปลี่ยนแปลงสัญญาณอินพุตอีก

จะเห็นได้ว่าระบบจะปรับค่าพารามิเตอร์ในช่วงผลตอบสนองชั่วขณะเท่านั้น ทำให้ต้องใช้สัญญาณอินพุตที่มีการเปลี่ยนแปลงอย่างต่อเนื่อง (persistent excitation) หลายคาบ แต่ถ้าใช้หลักการประมาณค่าพารามิเตอร์รวมเข้าด้วย จะช่วยปรับค่าพารามิเตอร์ของพลานท์ให้เข้าใกล้โมเดลตลอดเวลาเพราะจะทำนายค่าพารามิเตอร์จากเอาต์พุตของพลานท์ โดยไม่ได้พิจารณาเพียงแค่ค่าผิดพลาดระหว่างพลานท์กับโมเดลแต่เพียงอย่างเดียว

5) สำหรับซอฟต์แวร์ที่ได้เขียนขึ้นนี้ จะช่วยให้การวิเคราะห์และออกแบบระบบควบคุมเป็นไปอย่างมีประสิทธิภาพและมีความยืดหยุ่นในการทำงานสูงสุดโดยยึดแนวความคิดที่ว่า การออกแบบระบบควบคุมในกระบวนการจริง ๆ นั้น เราไม่สามารถออกแบบตามทฤษฎีเพียงอย่างเดียวเท่านั้น เนื่องจากระบบโดยทั่วไปจะมีขีดจำกัดบางประการ รวมทั้งในบางครั้งเราไม่จำเป็นต้องทราบทฤษฎีที่ลึกซึ้งมากจนเกินไป แต่อาจอาศัยจากการสังเกตพฤติกรรมของระบบและทำการวิเคราะห์อย่างมีเหตุผลเราก็สามารถที่จะควบคุมระบบนั้นๆ ได้ ฉะนั้นหากมีซอฟต์แวร์ที่สามารถทำการวิเคราะห์ระบบอย่างมีประสิทธิภาพแล้ว เราก็สามารถทำให้งานซึ่งมีความซับซ้อนมากในทางทฤษฎีกลายเป็นเรื่องที่มีความซับซ้อนน้อยลงในทางปฏิบัติ ได้นั่นเอง



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
1 : #include <alloc.h>
2 : #include <conio.h>
3 : #include <ctype.h>
4 : #include <dos.h>
5 : #include <dir.h>
6 : #include <graphics.h>
7 : #include <math.h>
8 : #include <mem.h>
9 : #include <stdlib.h>
10 : #include <stdio.h>
11 : #include <string.h>
12 : #include <stdarg.h>
13 : #include <time.h>
14 :
15 : #define TRUE      1
16 : #define FALSE     0
17 :
18 : #define UP        72
19 : #define DOWN      80
20 : #define LEFT      75
21 : #define RIGHT     77
22 : #define PgUp      73
23 : #define PgDn      81
24 : #define CR        13
25 : #define ESC       27
26 : #define EQ        '='
27 : #define SPACE     ' '
28 : #define BSP       '\b'
29 : #define TAB       '\t'
30 : #define ALT_L     38
31 : #define ALT_S     31
32 : #define ALT_N     49
33 : #define ALT_Y     21
34 : #define CTRL_Y    '\x19'
35 : #define CTRL_N    '\xE'
36 : #define DEL       83
37 : #define INS       82
38 : #define F1        59
39 : #define F2        60
40 : #define F3        61
41 : #define F10       68
42 : #define HOME      71
43 : #define END       79
44 :
45 : #define MAXCOL    76
46 : #define MAXROW    14
47 : #define MINCOL    4
48 : #define MINROW    4
49 : #define TAB_SIZE  4
50 : #define NORM      0x1E
51 : #define REV       0x71
52 : #define CUR       0x4F
53 : #define MSG_ATR   0x6F†
```

```

1 : /*****
2 : /***** DEFINE : support simulate.c *****/
3 : /*****
4 : #define MAX_DATA      1000
5 : #define DEFAULT_DRIVE 1 /* 1=B: ;default A: */
6 : #define CARRAGE_RETURN 13
7 : #define MAX_EQUATION 26
8 : #define BACK_SPACE 8
9 : #define ASC_a 97
10 :
11 : /*****
12 : /***** DEFINE : support 2runge.c *****/
13 : /*****
14 : #define MAXTIMEDOMAIN 600
15 : #define MAXORDER 26 /* linear differential order */
16 : #define MAXDEGREE 3 /* nonlinear order */
17 :
18 : /**
19 : #define maxx getmaxx()+1
20 : #define maxy getmaxy()+1
21 : #define x_center maxx/2
22 : #define y_center maxy/2
23 : **/
24 :
25 : #define EXPO 2.718282
26 : #define PHI 3.141593
27 :
28 :
29 : /*****
30 : /***** DEFINE : support 2polish.c *****/
31 : /*****
32 : #define MAXPRIORITY 6
33 : #define MAXSTACK 50
34 : #define MAXSTRING 80
35 : #define MAXTOKEN 50
36 : #define MAXNAME 7
37 : #define MAXDEGREE 3
38 : #define MAXVARIABLE 26
39 :
40 :
41 : typedef float Value_type;
42 : typedef struct var_token{int name_index,token_index;} Var_type;
43 :
44 : typedef enum kind_tag{UNARYOP,BINARYOP,OPERAND,LEFTPAREN,
45 : RIGHTPAREN,ENDEXPR} Kind_type;
46 :
47 : union info_tag {
48 : int pri;
49 : Value_type val;
50 : };
51 :
52 : typedef struct token_tag {
53 : int index;
54 : char name[MAXNAME];
55 : int kind;
56 : union info_tag info;
57 : }Token_type;
58 :
59 : typedef struct com_fmt {
60 : char com[15];
61 : char fmt[5];
62 : }Fmt_type;
63 :
64 :
65 :
66 : /*****
67 : /***** DEFINE : support 1file.c,simulate.c *****/
68 : /*****
69 : #define EXECUTEF_COM 7
70 : #define SET_EQU_COM 12
71 : #define PLOT_COM 14
72 : #define FFT_COM 18
73 : #define IFFT_COM 19
74 : #define MAX_COMMAND 23
75 : #define MAX_FORMAT MAX_COMMAND+5
76 : char command_table[MAX_COMMAND][3][25] =
77 : {{"RUN", "Execute Process.", "-"},
78 : {"QUIT", "Terminate Program.", "-"},
79 : {"HELP", "Show Command Help.", "-"},
80 : {"SHOWP", "Show Parameter.", "-"},

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :      {"FILE",      "Set File Name.",      "FILE NAME"},
82 :      {"PATH",      "Set Current Path.",    "FILE PATH"},
83 :      {"SHOWD",      "Show Data of File.",   "FILE NAME"},
84 :      {"EXECUTEF",    "Execute File.",        "-/FILE NAME"},
85 :      {"TIME",        "Input Range of Time.", "NUMBER"},
86 :      {"WIDTH",        "Graph Width.",        "NUMBER"},
87 :      {"PRECISION",    "Numerical Precision.", "NUMBER"},
88 :      {"INITIAL",      "Set Initial Condition.", "VAR/NUM,..."},
89 :      {"EQUATION",     "Input Sys. Equations.", "-"},
90 :      {"INTEGRATE",    "Integrate f(t) or Record", "f(t)/f(record)"},
91 :      {"PLOT",         "Plot rec-var from File.", "STRING(var)"},
92 :      {"VARIABLE",     "Declare Variable.",    "VAR/TYPE,..."},
93 :      {"RECORD",       "Record some Variables.", "VAR,VAR,..."},
94 :      {"SAVEDATA",     "Save Rec_Var to File.", "-"},
95 :      {"FFT",          "Fast Fourier Transform.", "Var(only one)"},
96 :      {"IFFT",         "Take FFT and then IFFT.", "Var & NUMBER"},
97 :      {"SHOWT",        "Tell Time or Consuming T", "-"},
98 :      {"FREEMEM",      "Free Mem,Default Para.", "-"},
99 :      {"ASSIGN",       "Assign Var.from Equation", "VAR=EQUATION"}
100:
101:   };
102:
103:
104:   Fmt_type command_fmt[MAX_FORMAT]=
105:   {{"RUN",""}, {"QUIT",""}, {"HELP",""}, {"SHOWP",""},
106:    {"FILE","f"}, {"PATH","p"}, {"SHOWD","f"}, {"EXECUTEF","f"},
107:    {"TIME","n"}, {"WIDTH","n"}, {"PRECISION","n"}, {"INITIAL","c\nr"},
108:    {"EQUATION",""}, {"INTEGRATE","e"}, {"PLOT","s"}, {"VARIABLE","ccr"},
109:    {"RECORD","s"}, {"SAVEDATA","f"}, {"FFT","s"}, {"IFFT","snn"},
110:    {"SHOWT","n"}, {"FREEMEM",""}, {"ASSIGN","c=e"},
111:
112:    {"GOTO","n"}, {"RETURN",""}, {"ENDEQUATION",""},
113:    {"ENDPROGRAM",""}, {"DEFINE","s=e"}
114:   };
115: /*****
116: ↑

```

```
1 : #define CLS_ATR 0x71
2 : #define CLS_CH '\xB0'
3 : #define BLOCK 10
4 : #define M_BAR 0x3F
5 : #define M_BKG 0x70
6 : #define S_BAR 0x3F
7 : #define S_BKG 0x70
8 : #define WINL 3
9 : #define WINT 18
10 : #define WINR 78
11 : #define WINB 23
12 :
13 : /*****/
14 : #include "i:\thesis3\key.h"
15 : #include "i:\thesis3\util.c"
16 : #include "i:\thesis3\mem_mng.c"
17 : #include "i:\thesis3\menu.c"
18 : /*#include "i:\thesis3\dir.c"*/
19 : #include "i:\thesis3\simulate.c"
20 : /*#include "i:\thesis3\edit.c"*/
21 : #include "i:\thesis3\linear.c"
22 : #include "i:\thesis3\phasepln.c"
23 : /*****/
24 :
25 :
26 : int Mainmenu(int i);
27 : void Sub_Option(void);
28 : void Sub_Run(void);
29 : void Sub_Plot(void);
30 : void Sub_Show(void);
31 : int Linear();
32 : void MainPhasePlane();
33 : void Sub_Math(void);
34 : void Help_Demo(void);
35 : void SetEquMenu(int *num_equation,int choice);
36 : int GetEquMenu(int num_equation,int choice);
37 :
38 : /***** EXTERNAL VARIABLE *****/
39 : int color_mode=0;
40 : int ready_run,width,iteration_scale,num_equation;
41 : char FILE_NAMES[50],Drive[50];
42 : Value_type time_range,iteration_time,var_val[26];
43 : char rec_var_str[27],equ_typ[27],var_str[MAX_EQUATION];
44 :
45 : /***** GLOBAL VARIABLE *****/
46 : HEAD far *Phead,*Vhead;
47 : int MEMFULL=0;
48 : int DIRECT=0,CR_P=0; /* check key LEFT RIGHT in main menu */
49 :
50 :
51 : void main(void)
52 : {
53 :     int i=1;
54 :     char str[80];
55 :
56 :     install_handler();
57 :     strcpy(Drive,"A:");
58 :     if(DEFAULT_DRIVE)
59 :         strcpy(Drive,"B:");
60 :     set_parameter();
61 :     clrscr();
62 :     Print_menu(1);
63 :
64 :     for(;;){
65 :         cursorShape(CURSOROFF);
66 :         i=Mainmenu(i);
67 :         switch(i){
68 :             case 1: /* Edit(); Print_menu(1); */
69 :                 break;
70 :             case 2: Sub_Option();
71 :                 break;
72 :             case 3: Sub_Run();
73 :                 break;
74 :             case 4: Sub_Plot();
75 :                 break;
76 :             case 5: Sub_Show();
77 :                 break;
78 :             case 6: Sub_Math();
79 :                 break;
80 :             case 7: Help_Demo();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :             break;
82 :         case 8: strcpy(str,Show_Get_message(2*BLOCK+5,i+3,0x0A,
83 :             "Press Y to confirm,N to cancel",2));
84 :             if(str[0]=='Y'){
85 :                 free_memory(); /* TERMINATE THE PROGRAM */
86 :                 Cls(1,1,80,25,0x07,SPACE);
87 :                 exit(0);
88 :             }
89 :             break;
90 :         }
91 :     }
92 : }
93 :
94 : int Mainmenu(int choice)
95 : {
96 :     char key;
97 :
98 :     for(;;){
99 :         Setattr((choice-1)*BLOCK+1,1,BLOCK,M_BAR);
100 :        if(DIRECT!=0 || (CR_P!=0 && choice>1 && choice<8)){
101 :            Setattr((choice-1)*BLOCK+1,1,BLOCK,M_BKG);
102 :            choice+=DIRECT;
103 :            Setattr((choice-1)*BLOCK+1,1,BLOCK,M_BAR);
104 :            DIRECT=0;
105 :            if(choice!=1) return(choice);
106 :        }
107 :        key=getch();
108 :        if(key==CR){
109 :            CR_P=1; break;
110 :        }
111 :        if(key==ESC) CR_P=0;
112 :        Setattr((choice-1)*BLOCK+1,1,BLOCK,M_BKG);
113 :        if(key==0){
114 :            key=getch();
115 :            switch(key){
116 :                case LEFT:if(choice>1) choice--;
117 :                           else choice=MAXC;
118 :                           break;
119 :                case RIGHT:if(choice<MAXC) choice++;
120 :                           else choice=1;
121 :                           break;
122 :            }
123 :        }
124 :    }
125 :    return(choice);
126 : }
127 :
128 : void Sub_Option(void)
129 : #define LINE1 9
130 : #define LENG1 18
131 : {
132 :     char *pt,str[MAXCOL],st[LINE1][LENG1]={"Set path","Set file",
133 :         "Set time range","Number of data","Set precision",
134 :         "State initialize","Declare variable","Set sys. equation",
135 :         "Show parameter"};
136 :     static int choice=1;
137 :     int txt[500],i;
138 :
139 :     i=choice;
140 :     Fastgettext(BLOCK+1,2,BLOCK+LENG1+8,LINE1+4,txt);
141 :     do{
142 :         i=Menu(BLOCK+1,2,S_BKG,S_BKG,S_BAR,&st,LINE1,LENG1,i);
143 :         choice=abs(i);
144 :         switch(i){
145 :             case 1: strcpy(str,Show_Get_message(BLOCK+5,i+3,MESG_ATR,
146 :                 "Drive ",1));
147 :                     set_drive(str,Drive);
148 :                     break;
149 :             case 2: strcpy(str,Show_Get_message(BLOCK+5,i+3,MESG_ATR,
150 :                 "File name",1 ));
151 :                     set_file(str,FILE_NAMES,Drive);
152 :                     break;
153 :             case 3: strcpy(str,Show_Get_message(BLOCK+5,i+3,MESG_ATR,
154 :                 "Time(sec)",1 ));
155 :                     set_time_range(str,&time_range);
156 :                     break;
157 :             case 4: strcpy(str,Show_Get_message(BLOCK+5,i+3,MESG_ATR,
158 :                 "Data number[500]",1 ));
159 :                     set_graph_width(str,&width);
160 :                     break;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
161:         case 5: strcpy(str,Show_Get_message(BLOCK+5,i+3,MESG_ATR,
162:         "Precision[10]",1) );
163:             set_precision_multiply(str,&iteration_scale);
164:             break;
165:         case 6: strcpy(str,Show_Get_message(BLOCK+5,i+3,MESG_ATR,
166:         "Initial",1) );
167:             if( str[0]!='\0' )
168:                 set_initial_state(str,var_val);
169:             break;
170:         case 7: strcpy(str,Show_Get_message(BLOCK+5,i+3,MESG_ATR,
171:         "Variable(string)",1) );
172:             if( str[0]!='\0' )
173:                 var_declaration(str);
174:             break;
175:         case 8: SetEquMenu(&num_equation,i);
176:             break;
177:         case 9: makewin(1,2,80,25,0x7F,0x31,0);
178:             Fastputs( (80-strlen("PARAMETER"))/2,2,0x70,"PARAMETER");
179:             window(3,3,78,24);
180:             textattr(0x31);
181:             show_all_parameter(); getch();
182:             Print_menu(2);
183:             break;
184:     }
185: }while(i>0);
186: Fastputtext(BLOCK+1,2,BLOCK+LENG1+8,LINE1+4,txt);
187: }
188:
189: void Sub_Run(void)
190: #define LINE2 6
191: #define LENG2 22
192: {
193:     char str[MAXCOL],st[LINE2][LENG2]={ "Set Record variable",
194:     "Run from parameter","Run from file", "Run command line",
195:     "Save data","Show Data"};
196:     int txt[500],i;
197:     static int choice=1;
198:
199:     i=choice;
200:     Fastgettext(2*BLOCK+1,2,2*BLOCK+LENG2+8,LINE2+4,txt);
201:     do{
202:         i=Menu(2*BLOCK+1,2,S_BKG,S_BKG,S_BAR,&st,LINE2,LENG2,i);
203:         choice=abs(i);
204:         switch(i){
205:             case 1: strcpy(str,Show_Get_message(2*BLOCK+5,i+3,MESG_ATR,
206:             "String(var)",1) );
207:                 if( str[0]!='\0' )
208:                     strcpy(rec_var_str,record(str) );
209:                 break;
210:             case 2: run_parameter();
211:                 break;
212:             case 3: strcpy(str,Show_Get_message(2*BLOCK+5,
213:             i+3,MESG_ATR, "File name",1));
214:                 if(strchr(str,'*'))
215:                     Directory(str);*/
216:                 if(str[0]!='\0'){
217:                     Fastputtext(2*BLOCK+1,2,2*BLOCK+LENG2+8,LINE2+4,txt);
218:                     ExecuteFile(str);
219:                     Print_menu(3);
220:                     cursorShape(CURSROFF);
221:                 }
222:                 break;
223:             case 4: Setattr(1,25,80,0x71);
224:                 ExecuteCommand();
225:                 Print_menu(3);
226:                 break;
227:             case 5: if(ready_run==0){
228:                 Show_Get_message(2*BLOCK+5,i+3,0x0B,
229:                 "Process is not executed yet.",0);
230:                 break;
231:             }
232:                 strcpy(str,Show_Get_message(2*BLOCK+5,i+3,MESG_ATR,
233:                 "File Destination",1));
234:                 save_data(str);
235:                 break;
236:             case 6: strcpy(str,Show_Get_message(2*BLOCK+5,i+3,MESG_ATR,
237:             "Source File",1));
238:                 makewin(1,2,80,25,0x7F,0x31,0);
239:
240:
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 :      Fastputs( (80-strlen("Show Data"))/2,2,0x70,"Show data");
2 :      window(3,3,78,24);
3 :      textattr(0x31);
4 :      show_data(str,FILE_NAMES); getch();
5 :      Print_menu(3);
6 :  }
7 :  }while(i>0);
8 :  Fastputtext(2*BLOCK+1,2,2*BLOCK+LENG2+8,LINE2+4,txt);
9 : }
10 :
11 : void Sub_Plot(void)
12 : #define LINE3 4
13 : #define LENG3 14
14 : {
15 :     char str[MAXCOL],st[LINE3][LENG3]={"From memory","From file",
16 :                                           "Color","Black/White"};
17 :     int txt[300],i;
18 :     static int choice=1;
19 :
20 :     i=choice;
21 :     Fastgettext(3*BLOCK+1,2,3*BLOCK+LENG3+8,LINE3+4,txt);
22 :     do{
23 :         i=Menu(3*BLOCK+1,2,S_BKG,S_BKG,S_BAR,&st,LINE3,LENG3,i);
24 :         choice=abs(i);
25 :         switch(i){
26 :             case 1: plot_graph("MEMORY");
27 :                     Print_menu(4);
28 :                     break;
29 :             case 2: strcpy( str,Show_Get_message(3*BLOCK-1,i+3,MESG_ATR,
30 : "File name",1) );
31 : /* if(strchr(str,'*'))
32 :             Directory(str);*/
33 :             if( str[0]!='\0' ){
34 :                 plot_graph(str);
35 :                 Print_menu(4);
36 :             }
37 :             break;
38 :             case 3: color_mode=0; /* Default Color Mode */
39 :                     cprintf("\n\rGraph are plotted in colors.");
40 :                     break;
41 :             case 4: color_mode=1;
42 :                     cprintf("\n\rGraph are plotted in Black/White colors.");
43 :                     break;
44 :         }
45 :     }while(i>0);
46 :     Fastputtext(3*BLOCK+1,2,3*BLOCK+LENG3+8,LINE3+4,txt);
47 : }
48 :
49 : void Sub_Show(void)
50 : #define LINE4 2
51 : #define LENG4 15
52 : {
53 :     char str[MAXCOL],st[LINE4][LENG4]={"Phase plane","Linear System"};
54 :     int txt[200],i;
55 :     static int choice=1;
56 :
57 :     i=choice;
58 :     Fastgettext(4*BLOCK+1,2,4*BLOCK+LENG4+8,LINE4+4,txt);
59 :     do{
60 :         i=Menu(4*BLOCK+1,2,S_BKG,S_BKG,S_BAR,&st,LINE4,LENG4,i);
61 :         choice=abs(i);
62 :         switch(i){
63 :             case 1: MainPhasePlane();
64 :                     Print_menu(5);
65 :                     break;
66 :             case 2: Linear();
67 :                     Print_menu(5);
68 :                     break;
69 :         }
70 :     }while(i>0);
71 :     Fastputtext(4*BLOCK+1,2,4*BLOCK+LENG4+8,LINE4+4,txt);
72 : }
73 :
74 : void Sub_Math(void)
75 : #define LINE5 6
76 : #define LENG5 16
77 : {
78 :     char str[MAXCOL],st[LINE5][LENG5]={"Fast Fourier","Inverse FFT",
79 : "Assign Record","Integrate","Time Range","Number of Data"};
80 :     int txt[300],i;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 : static int choice=1;
82 :
83 : i=choice;
84 : Fastgettext(5*BLOCK+1,2,5*BLOCK+LENG5+8,LINE5+4,txt);
85 : do{
86 :     i=Menu(5*BLOCK+1,2,S_BKG,S_BKG,S_BAR,&st,LINE5,LENG5,i);
87 :     choice=abs(i);
88 :     switch(i){
89 :     case 1: strcpy(str,Show_Get_message(3*BLOCK,i+3,MESG_ATR,
90 :         "Signal,destination1(or 1,2)",1) );
91 :         if( str[0]!='\0' ){
92 :             FFT(str);
93 :             Print_menu(6);
94 :         }
95 :         break;
96 :     case 2: strcpy(str,Show_Get_message(3*BLOCK,i+3,MESG_ATR,
97 :         "Destination,An,Bn",1) );
98 :         if( str[0]!='\0' ){
99 :             IFFT(str);
100 :            Print_menu(6);
101 :        }
102 :        break;
103 :    case 3: strcpy(str,Show_Get_message(3*BLOCK,i+3,MESG_ATR,
104 :        "Format : Var= Equation",1) );
105 :        if( str[0]!='\0' )
106 :            Assign(str);
107 :        break;
108 :    case 4: strcpy(str,Show_Get_message(3*BLOCK,i+3,MESG_ATR,
109 :        "Function",1) );
110 :        if( str[0]!='\0' )
111 :            Integrate(str);
112 :        break;
113 :    case 5: strcpy(str,Show_Get_message(3*BLOCK,i+3,MESG_ATR,
114 :        "Time(sec)",1) );
115 :        set_time_range(str,&time_range);
116 :        break;
117 :    case 6: strcpy(str,Show_Get_message(3*BLOCK,i+3,MESG_ATR,
118 :        "Data number",1) );
119 :        set_graph_width(str,&width);
120 :        break;
121 :    }
122 : }while(i>0);
123 : Fastputtext(5*BLOCK+1,2,5*BLOCK+LENG5+8,LINE5+4,txt);
124 : }
125 :
126 : void Help_Demo(void)
127 : #define LINE6 2
128 : #define LENG6 12
129 : {
130 :     char str[MAXCOL],st[LINE6][LENG6]={"Help Table","File Demo"};
131 :     int txt[200],i;
132 :     static int choice=1;
133 :     i=choice;
134 :     Fastgettext(6*BLOCK+1,2,6*BLOCK+LENG6+8,LINE6+4,txt);
135 :     do{
136 :         i=Menu(6*BLOCK+1,2,S_BKG,S_BKG,S_BAR,&st,LINE6,LENG6,i);
137 :         choice=abs(i);
138 :         switch(i){
139 :         case 1: makewin(1,2,80,25,0x7F,0x31,0);
140 :             Fastputs( (80-strlen("HELP"))/2,2,0x70,"HELP");
141 :             window(3,3,78,24);
142 :             textattr(0x31);
143 :             help(); getch();
144 :             Print_menu(7);
145 :             break;
146 :         case 2: /* Demo() */
147 :             Print_menu(7);
148 :             break;
149 :         }
150 :     }while(i>0);
151 :     Fastputtext(6*BLOCK+1,2,6*BLOCK+LENG6+8,LINE6+4,txt);
152 : }
153 :
154 : void SetEquMenu(int *num_equation,int choice)
155 : { ready_run=0;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
161: free_equ_alloc();
162: free_var_alloc();
163: var_str[1]='\0';
164: *num_equation=0;
165: while( !GetEquMenu(*num_equation,choice) )
166:     (*num_equation)++;
167: var_str[( *num_equation)+1]='\0';
168: AdjEquTyp(equ_typ,var_str);
169: }
170:
171:
172: int GetEquMenu(int num_equation,int choice) /* choice of sub_menu */
173: { int equation_index;
174:   char str[MAXCOL];
175:   equation_index=num_equation+1;
176:   itoa(equation_index,str,10);
177:   strcat(str," ");
178:   strcpy(str,Show_Get_message(BLOCK+5,choice+3,0x6F,
179:                               str,1));
180:   return( set_equ_from_str(equation_index,str) );
181: }
182: ↑
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : /*****
2 : FILE NAME : SIMULATE.C
3 : SUPPORT   : MAINP.C (only)
4 : REQUIRE   : I_DEFINE.H
5 :           1STRING.C,1FILE.C,1SETPARA.C,1PLOTGPH.C
6 :           2POLISH.C,2RUNGE.C,2MANYPLT.C
7 :           3COMPILE.C,3FOURIER.C
8 : *****/
9 :
10 : #include "i:\thesis3\i_define.h"
11 : #include "i:\thesis3\1string.c"
12 : #include "i:\thesis3\1file.c"
13 : #include "i:\thesis3\1setpara.c"
14 : #include "i:\thesis3\1plotgph.c"
15 :
16 : #include "i:\thesis3\2polish.c"
17 : #include "i:\thesis3\2runge.c"
18 : #include "i:\thesis3\2manyp1t.c"
19 : #include "i:\thesis3\3compile.c"
20 : #include "i:\thesis3\3fourier.c"
21 :
22 : /***** EXTENAL VARIABLE FROM MAINP.C *****/
23 : extern int      ready_run;
24 : extern int      width,iteration_scale,num_equation;
25 : extern char     FILE_NAMES[50],Drive[50];
26 : extern Value_type time_range,iteration_time;
27 : extern Value_type var_val[26]; /* [26] mean a,b,...,z = 0,1,...,25*/
28 : extern char     rec_var_str[27];
29 : /* sequence of Recorded Variables in string */
30 : extern char     var_str[MAX_EQUATION];
31 : /* tell:which equation belongs to which variable */
32 : /* store variable name of each equation */
33 : extern char     equ_typ[27];
34 : /* E,S= Expression, State */
35 : /* which variable is what type of an Equation */
36 :
37 : /***** GLOBAL FOR SIMULATE.C *****/
38 :
39 : Value_type      far *rec_var[26];
40 : /* point to allocated array of record-variable */
41 : Var_type        far *var_tok[26];
42 : /* store variable position in token */
43 : Token_type      far *pol_tok[26];
44 : /* store TOKEN of polished string */
45 :
46 : char            far *lex_str[MAX_EQUATION];
47 : /* lex_str,equation_str allocated simutaneously*/
48 : /* store string of equation to polish */
49 : /* index = the No. or Order of Equation */
50 : char            far *equation_str[MAX_EQUATION];
51 : /* whole string of equation , "a=a+b+c" */
52 : char            rec_alloc[27];
53 : /* tell allocation of recorded-var array */
54 : /* *rec_var */
55 : char            var_alloc[27];
56 : /* tell allocation of variable detial */
57 : /* *lex_str, *var_tok, *pol_tok */
58 : char            equ_alloc[27];
59 : /* tell allocation of equation-string */
60 : /* *equation_str */
61 : FILE *fp;
62 :
63 :
64 : /***** PROTOTYPE : DO_COMMAND *****/
65 : void print_output_window(int command_index);
66 : void do_command(char *parameter,int command_index);
67 : void set_parameter(void);
68 : void ExecuteCommand(void);
69 : void run_parameter(void); /*0*/
70 :
71 : void free_memory(void);
72 : void free_equ_alloc(void);
73 : void free_var_alloc(void);
74 : void free_rec_alloc(void);
75 : void help(void); /*2*/
76 : void show_all_parameter(void); /*3*/
77 : void show_initial(Value_type *var_val);
78 : void show_equation(char **equation_str);
79 :

```

60 :เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 : int ExecuteFile(char *file_name);          /*7*/
82 : int RunList(char *file_name);
83 : int InitFile(char *file_name);
84 : int CompileProgram(void);
85 : int CheckProgSyntetic(int *line_err_ptr,char *error_mes);
86 : int GetComLine(char *command,char *parameter,int line_no);
87 : int SetEquFile(int line_start,int *num_equation);
88 : void AdjEquTyp(char *equ_typ,char *var_str);
89 :
    para.c */
90 : void set_graph_width(char *parameter,int *width);          /*9*/
91 : void set_sys_equations(int *num_equation);          /*12*/
92 : int get_equation(int i);
93 :     int set_equ_from_str(int equation_index,char *str);
94 :     int assign_var_str(int equa_index,char var_ch);
95 :     int assign_equation_str(int equa_index,char *str);
96 :     void replace_equ_str(char *str);
97 :     int assign_var_detail(int equa_index,char var_ch);
98 :     int allocate_equa(char **equation1,char **equation2,char var_ch);
99 : int Integrate(char *equation_str);          /*13*/
100 : void plot_graph(char *plot_vars);          /*14*/
101 :     int PlotVariable(Value_type **rec_var,char *rec_var_str,char *var_str,
102 :         int width,Value_type time_range,Value_type init_time);
103 : int var_declaration(char *parameter);          /*15*/
104 : int allocate_var(char chl);
105 : char *record(char *parameter);          /*16*/
106 : void clear_memory(Value_type *pointer,int width);
107 : int alloc_rec(char ch,int width);
108 : int save_data(char *parameter);
109 : int FFT(char *var_ch);          /*18*/
110 :     int AssignValue(char var_dest,Value_type *source,int width);
111 :     void CopyMemory(Value_type *dest,Value_type *source,int width);
112 : int IFFT(char *parameter);          /*19*/
113 : int TellTime(int mode);          /*20*/
114 : void FreeAndSet(void);          /*21*/
115 : int Assign(char *parameter);          /*22*/
116 :     int SetValEqu(char *equation_str, Value_type *DATA, int width);
117 :     int RecExistString(Var_type *equ_vartype,char *var_exist,
118 :         char *rec_var_str);
119 :
120 :
121 :
122 : /*****
123 :
124 : #define OUTL 1
125 : #define OUTT 1
126 : #define OUTR 80
127 : #define OUTB 24
128 :
129 : void ExecuteCommand(void)
130 : { char command[50],parameter[50],string_com[80];
131 :   int command_index,cury;
132 :   struct text_info tx;
133 :
134 :   cursorShape(CURSORMON);
135 :   gettextinfo(&tx);
136 :   cury=1;
137 :   Cls(1,1,80,25,0x07,SPACE);
138 :   makewin(OUTL,OUTT,OUTR,OUTB,0x7F,0x31,0);
139 :   Fastputs( (OUTR-strlen("Output"))/2,OUTT,0x70,"Output");
140 :   for(;;){
141 :       window(1,1,80,25);
142 :       Cls(12,25,80,25,0x71,SPACE);
143 :       textattr(0x07);
144 :       Fastputs(1,25,0x71," COMMAND: ");
145 :       gotoxy(13,25);
146 :       getstr(string_com,66,0x71);
147 :       window(OUTL+2,OUTT+1,OUTR-2,OUTB-1);
148 :       textattr(0x31);
149 :       gotoxy(1,cury);
150 :
151 :                               /*string_com[0] = '\0';
152 :                               get_string_from_keyboard(string_com);*/
153 :       parse(command,parameter,string_com);
154 :       command_index=table_compare(command);
155 :       if(command_index==1)
156 :           break; /* 1 mean "quit" command */
157 :       do_command(parameter,command_index);
158 :       print_output_window(command_index);
159 :       cursorShape(CURSORMON);
160 :       cury=wherey();
161 :   }

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161:
162:     window(1,1,80,25);
163:     textattr(tx.attribute);
164:     cursorShape(CURSORMOFF);
165: }
166:
167: void print_output_window(int command_index)
168: {
169:     switch(command_index){
170:         case -1 : /* Special case */
171:             case EXECUTEF_COM :
172:             case PLOT_COM :
173:             case FFT_COM :
174:             case IFFT_COM :
175:                 makewin(OUTL,OUTT,OUTR,OUTB,0x7F,0x31,0);
176:                 Fastputs( (OUTR-strlen("Output"))/2,OUTT,0x70,"Output");
177:                 break;
178:     }
179: }
180:
181:
182: void do_command(char *parameter,int command_index)
183: {
184:     strlwr(parameter);
185:     switch(command_index){
186:         case 0 : run_parameter(); break;
187:         case 1 : break; /*quit command used for terminating ExecutCommand*/
188:         case 2 : help(); break;
189:         case 3 : show_all_parameter(); break;
190:         case 4 : set_file(parameter,FILE_NAMES,Drive); break;
191:         case 5 : set_drive(parameter,Drive); break;
192:         case 6 : show_data(parameter,FILE_NAMES); break;
193:         case 7 : ExecuteFile(parameter); break;
194:         case 8 : set_time_range(parameter,&time_range); break;
195:         case 9 : set_graph_width(parameter,&width); break;
196:         case 10 : set_precision_multiply(parameter,&iteration_scale); break;
197:         case 11 : set_initial_state(parameter,var_val); break;
198:         case 12 : set_sys_equations(&num_equation); break;
199:         case 13 : Integrate(parameter); break;
200:         case 14 : plot_graph(parameter); break;
201:         case 15 : var_declaration(parameter); break;
202:         case 16 : strcpy(rec_var_str,record(parameter)); break;
203:         case 17 : save_data(parameter); break;
204:         case 18 : FFT(parameter); break;
205:         case 19 : IFFT(parameter); break;
206:         case 20 : TellTime(atoi(parameter)); break;
207:         case 21 : FreeAndSet(); break;
208:         case 22 : Assign(parameter); break;
209:         default : ErrorMessage(1,"Command error!: Press Help for instructions");
210:     };
211: }
212:
213: void set_parameter(void)
214: {
215:     int ch_index;
216:     set_f_name(FILE_NAMES,Drive);
217:     ready_run=0;
218:     num_equation=0;
219:     width=MAXTIMEDOMAIN;
220:     time_range=10;
221:     iteration_scale=10;
222:     iteration_time=time_range/(width*iteration_scale);
223:     for(ch_index=0; ch_index<=25; ch_index++){
224:         var_val[ch_index]=0; /* initialize all states to zero */
225:         equ_typ[ch_index]='_';
226:         var_alloc[ch_index]='_'; /*** if allocated then store 'Y' ***/
227:         equ_alloc[ch_index]='_';
228:         rec_alloc[ch_index]='_';
229:     }
230:     equ_typ[26]=var_alloc[26]=equ_alloc[26]=rec_alloc[26]='\0';
231:     var_str[1]='\0';
232: }
233:
234: /***** 0 *****/
235: void run_parameter(void)
236: {
237:     if(num_equation>0)
238:         if(ExecuteEquation(num_equation)==0)
239:             ready_run=1;
240: }

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
241:
242: /***** free all memory that previously allocated *****/
243: void free_memory(void)
244: { free_equ_alloc();
245:   free_var_alloc();
246:   free_rec_alloc();
247: }
248:
249: void free_equ_alloc(void)
250: { int equation_index;
251:   char var_ch;
252:   for(equation_index=1; equation_index<=num_equation; equation_index++){
253:     var_ch=var_str[equation_index];
254:     if( equ_alloc[var_ch-ASC_a]=='Y' ){
255:       equ_alloc[var_ch-ASC_a]='_';
256:       farfree( equation_str[equation_index] );
257:       farfree( lex_str[equation_index] );
258:     }
259:   }
260: }
261:
262: void free_var_alloc(void)
263: { int equation_index,var_index;
264:   for(var_index=0; var_index<=25; var_index++){
265:     if( var_alloc[var_index]=='Y' ){
266:       var_alloc[var_index]='_';
267:       farfree( var_tok[var_index] );
268:       farfree( pol_tok[var_index] );
269:     }
270:   }
271: }
272:
273: void free_rec_alloc(void)
274: { int equation_index,var_index;
275:   for(equation_index=1; equation_index<=num_equation; equation_index++){
276:     var_index=var_str[equation_index]-ASC_a;
277:     if( rec_alloc[var_index]=='Y' ){
278:       rec_alloc[var_index]='_';
279:       farfree( rec_var[var_index] );
280:     }
281:   }
282:   rec_var_str[0]='\0';
283: }
284:
285:
286: /***** 2 *****/
287: void help(void)
288: {int i;
289:   Cprint("\n\n----- HELP -----");
290:   Cprint("\n\n      COMMAND      PARAMETER      PURPOSE");
291:   for(i=0;i<MAX_COMMAND;){
292:     Cprint("\n\n%-12s ",command_table[i][0]);
293:     Cprint("%-20s ",command_table[i][2]);
294:     Cprint("%-40s",command_table[i][1]);
295:     if( (++i)%20==0){
296:       Cprint("\n\n-----PRESS ANY KEY TO CONTINUE-----");
297:       getch();
298:     };
299:   }
300: }
301:
302: /***** 3 *****/
303: void show_all_parameter(void)
304: { int i,check;
305:   Cprint("\n\n***** SHOW ALL PARAMETER *****");
306:   Cprint("\n\n DRIVE      : %s",Drive);
307:   Cprint("\n\n FILE_NAMES : %s",FILE_NAMES);
308:   Cprint("\n\n time_range : %-6.1f",time_range);
309:   Cprint("\n\n graph_width : %-6d",width);
310:   Cprint("\n\n precision_scale: %-6d",iteration_scale);
311:   Cprint("\n\n equation type : ");
312:   Cprint("\n\n (1)expression : ");
313:   for(i=0; i<=25; i++){
314:     if(equ_typ[i]=='E') Cprint("%c",'a'+i);
315:   }
316:   Cprint("\n\n (2)diff. equ. : ");
317:   for(i=0; i<=25; i++){
318:     if(equ_typ[i]=='S') Cprint("%c",'a'+i);
319:   }
320:   Cprint("\n\n recorded var. : ");
321:   for(i=0,check=0; i<=25; i++){
322:     if(rec_alloc[i]=='Y'){
323:       Cprint("%c",'a'+i);
324:       check++;
325:     }
326:   }
327:   if(check==0) Cprint("\n\n no recorded variable");
328: }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้คัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

321:         Cprint("%c",'a'+i);
322:         check=1;
323:     }
324:     if(check==0)
325:         Cprint("there are no variables allocated for recording.");
326:     Cprint("\n\r system variable: ");
327:     for(i=0; i<=25; i++){
328:         if(equ_alloc[i]=='Y') Cprint("%c",'a'+i);
329:         show_initial(var_val);
330:     } show_equation(equation_str);
331: }
332:
333:
334: void show_initial(Value_type *var_val)
335: {Value_type *val_ptr;
336:  int i,zero_init=1;
337:  val_ptr=var_val;
338:  Cprint("\n\r initial state : ");
339:  for(i=0; i<=25; i++){
340:      if( (*val_ptr)!=0 ){
341:          Cprint("%c%-3.1f/",i+ASC_a,*val_ptr);
342:          zero_init=0;
343:      }
344:      val_ptr++;
345:  }
346:  if(zero_init==1)
347:      Cprint("all states is initialized by zero.");
348: }
349:
350: void show_equation(char **equation_str)
351: { int i;
352:  char variable;
353:  Cprint("\n\r _____%-2d SYSTEM EQUATIONS_____",num_equation);
354:  for(i=1;i<=num_equation;i++){
355:      variable=equation_str[i][0];
356:      Cprint("\n\r (%d)  %c : %s",i,variable,equation_str[i]);
357:  }
358: }
359:
360:
361:
362: /***** case 6 are in 1file.c *****/
363:
364: /***** case 4,5,8,9,10,11,13 in 1setpara.c *****/
365:
366: /***** 7 *****/
367: #define MAX_RESERVE 4 /* SIZE of array "line_table" */
368: char line_table[MAX_RESERVE][15]={"GOTO","RETURN","ENDEQUATION",
369:                                     "ENDPROGRAM"};
370:
371: int ExecuteFile(char *file_name)
372: { struct text_info tx;
373:
374:     gettextinfo(&tx);
375:     cursorShape(CURSOROFF);
376:     if(!InitFile(file_name)) return(1);
377:     print_output_window(-1);
378:     window(OUTL+2,OUTT+1,OUTR-2,OUTB-1);
379:     RunList(file_name);
380:     window(1,1,80,25);
381:     textattr(tx.attribute);
382: }
383:
384:
385: int RunList(char *file_name)
386: { char parameter[80],command[80],command_line[80];
387:  int  com_index;
388:  int  line_no;
389:  int  next_no=0;
390:
391:  set_parameter();
392:  line_no=1;
393:  com_index=GetComLine(command,parameter,line_no);
394:  while( com_index!=-MAX_RESERVE ){
395:      switch(com_index){
396:          case SET_EQU_COM : line_no=SetEquFile(line_no,&num_equation);
397:                           break;
398:          case -1 : next_no=line_no+1;
399:                  line_no=atoi(parameter);
400:                  break;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
401:     case -2 : if(next_no!=0){ /* RETURN */
402:                 line_no=next_no;
403:                 next_no=0;
404:             }
405:             else /* For the first "RETURN" in program, We skip. */
406:                 line_no++;
407:             break;
408:     case -3 : ErrorMessage(0,"[Executefile]: ENDEQUATION MISPLACED");
409:             return(1);
410:     default : do_command(parameter,com_index);
411:             if(com_index==EXECUTEF_COM){
412:                 Print_menu(3);
413:                 InitFile(file_name);
414:             }
415:             else{
416:                 print_output_window(com_index);
417:                 window(OUTL+2,OUTT+1,OUTR-2,OUTB-1);
418:             }
419:             line_no++;
420:         }
421:         com_index=GetComLine(command,parameter,line_no);
422:     }
423:     return(0);
424: }
425:
426:
427: int InitFile(char *file_name)
428: { if(!Loadlist(file_name)) return(1);
429:   if(Check_program())>0 return(1);
430:   return(CompileProgram());
431: }
432: }
433:
434: int CompileProgram(void)
435: {int line_error;
436:  char mes[80],error_mes[80];
437:  if(CheckProgSyntetic(&line_error,error_mes)!=0){
438:      sprintf(mes,"LINE %d :%s",line_error,error_mes);
439:      ErrorMessage(0,mes);
440:      return(1);
441:  }
442:  else{
443:      ErrorMessage(0,"Compile Complete");
444:      return(0);
445:  }
446: }
447:
448: int CheckProgSyntetic(int *line_err_ptr,char *error_mes)
449: { int line_no=1;
450:   int node_start=0, node_next=0;
451:   char command_line[80];
452:   *line_err_ptr=0;
453:   do{
454:       strcpy(command_line,Getcommand(line_no));
455:
456:       if( node_start==2)
457:           replace_equ_str(command_line);
458:
459:       node_next=NextNode(node_start,command_line,error_mes);
460:       if(node_next!= -2)
461:           node_start=node_next;
462:       else
463:           *line_err_ptr=line_no;
464:       line_no++;
465:   }while((strcmp(strupr(command_line),"ENDPROGRAM")!=0) &&
466:          (node_start!=-1) && (node_next!=-2) );
467:   if( node_start== -1)
468:       return(0); /* no error */
469:   else
470:       return(1); /* error */
471: }
472:
473: int GetComLine(char *command,char *parameter,int line_no)
474: { int com_index,i;
475:   char command_line[80];
476:   strcpy(command_line,Getcommand(line_no));
477:   parse(command,parameter,command_line);
478:   com_index=table_compare(command);
479:   if(com_index!=MAX_COMMAND)
480:       return(com_index);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

481:     else{
482:         for(i=1; i<=MAX_RESERVE; i++)
483:             if(strcmp(command,line_table[i-1])==0)
484:                 return(-i);
485:         return(-MAX_RESERVE-1);
486:     }
487: }
488:
489: int SetEquFile(int line_start,int *num_equation)
490: {char equ_string[80],parameter[5];
491:  int com_index;
492:  int line_no;
493:
494:  ready_run=0;
495:  free_var_alloc();
496:  free_equ_alloc();
497:  *num_equation=1;
498:  var_str[1]='\0';
499:  line_no=line_start+1;
500:
501:  com_index=GetComLine(equ_string,parameter,line_no);
502:  if(com_index==MAX_RESERVE)
503:      ErrorMessage(0,"ENDEQUATION NOT EXIST");
504:  while( com_index!=-3 ){ /* -3 mean ENDEQUATION */
505:      if( set_equ_from_str(*num_equation,equ_string)==1 )
506:          ErrorMessage(0,"[SetEquFile]: equation format error!!!");
507:      else{
508:          var_str[*num_equation]=equ_string[0];
509:          (*num_equation)++;
510:          var_str[( *num_equation )]='\0';
511:      }
512:      line_no++;
513:      com_index=GetComLine(equ_string,parameter,line_no);
514:      if(com_index==MAX_RESERVE)
515:          ErrorMessage(0,"ENDEQUATION NOT EXIST");
516:  }
517:  line_no++;
518:  (*num_equation)--;
519:  AdjEquTyp(equ_typ,var_str);
520:  return(line_no);
521: }
522:
523: void AdjEquTyp(char *equ_typ,char *var_str)
524: { int i,j;
525:   for(i=0; i<=25; i++)
526:       if(equ_typ[i]=='S'){
527:           equ_typ[i]='E';
528:           for(j=1; j<=num_equation; j++)
529:               if(var_str[j]==(i+ASC_a))
530:                   equ_typ[i]='S';
531:       }
532: }
533:
534: /***** 9 *****/
535: void set_graph_width(char *parameter,int *width)
536: {int value;
537:   if( parameter[0]=='\0' )
538:       Cprint("\n\rGraph Width : %-4d", *width);
539:   else{
540:       if( (value=atoi(parameter))>0 ){
541:           *width=value;
542:           ready_run=0;
543:           strcpy(rec_var_str,record(rec_var_str));
544:           Cprint("\n\rREMARK: Memory for Record Variable are reallocated");
545:       }
546:       Cprint("\n\rGraph Width : %-4d ...received !!",*width);
547:   }
548: }
549:
550:
551: /***** 12 *****/
552: void set_sys_equations(int *num_equation)
553: { ready_run=0;
554:   Cprint("\n\r----- SET SYSTEM EQUATION -----");
555:   free_var_alloc();
556:   free_equ_alloc();
557:   var_str[1]='\0';
558:   *num_equation=0;
559:   while( !get_equation(*num_equation) )
560:       (*num_equation)++;

```

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
561: Cprint("\n\r");
562: var_str[(num_equation)+1]='\0';
563: AdjEquTyp(equ_typ,var_str);
564: }
565:
566: int get_equation(int num_equation)
567: /* return 0: no error, 1:error */
568: { int equation_index;
569:   char str[MAXSTRING];
570:   equation_index=num_equation+1;
571:   Cprint("\n\r%d : ",equation_index);
572:   getstr(str,MAXSTRING,0x31);
573:   return( set_equ_from_str(equation_index,str) );
574: }
575:
576: int set_equ_from_str(int equation_index,char *str)
577: { char var_ch,diff_ch;
578:   if( strlen(str) > MAXSTRING ){
579:     ErrorMessage(0,"String Too Long");
580:     return(1);
581:   }
582:   rid_a_space(str);
583:   strlwr(str);
584:   var_ch=*str;
585:   if( (var_ch>='a') && (var_ch<='z') ){
586:     if(var_ch=='t'){
587:       Cprint("\n\r");
588:       ErrorMessage(0,"Cannot define time-variable");
589:       return(1);
590:     }
591:     if( assign_var_str(equation_index,var_ch) ==0){ /* No repetition */
592:       if(assign_equation_str(equation_index,str) ){
593:         assign_var_detail(equation_index,var_ch) ){
594:           ErrorMessage(0,"EQUATION FORMAT ERROR");
595:           return(1); /* synthetic error */
596:         }
597:         if( equ_typ[var_ch-ASC_a]=='E')
598:           diff_ch='';
599:         else
600:           diff_ch='^';
601:         Cprint("\n\rEqu. (%d): %c%c=%s ..received!!",
602:               equation_index,var_ch,diff_ch,lex_str[equation_index]);
603:         return(0); /* no ERROR */
604:       }
605:     }
606:     return(1); /* ERROR !! */
607: }
608:
609: int assign_var_str(int equa_index,char var_ch)
610: { int i,repeat=0; /** "repeat" : a repetition in define same variable.*/
611:   for(i=1; i<=num_equation; i++)
612:     if(var_str[i]==var_ch){
613:       repeat=1;
614:       Cprint("\n\r");
615:       ErrorMessage(0,"[assign_var_string]: repeat variable define");
616:     };
617:   if(repeat==0) {
618:     var_str[equa_index]=var_ch;
619:     if( equ_typ[var_ch-ASC_a]!='S' )
620:       equ_typ[var_ch-ASC_a]='E';
621:   }
622:   return(repeat);
623: }
624:
625: int assign_equation_str(int equa_index,char *str)
626: { char var_ch=*str,equ_format[4]="c=e";
627:
628:   if( strchr(str,'=') ==NULL )
629:     return(1); /* = not exist */
630:   replace_equ_str(str);
631:   if(CheckSyntetic(equ_format,str)==1)
632:     return(1);
633:   if(allocate_equa(&equation_str[equa_index],&lex_str[equa_index],var_ch))
634:     return(1);
635:   strcpy(equation_str[equa_index],str); /*should check succeed of alloc*/
636:   strcpy(lex_str[equa_index], get_str(1,'=','\0',str) );
637:   return(0);
638: }
639:
640: void replace_equa_str(char *str)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

641: {char value_str[80];
642:   if( strstr(str,"time") !=NULL ){
643:     sprintf(value_str,"%3.2f",time_range);
644:     replace_string(str,"time",value_str);
645:   }
646:   if( strstr(str,"precision") !=NULL ){
647:     sprintf(value_str,"%d",iteration_scale);
648:     replace_string(str,"precision",value_str);
649:   }
650:   if( strstr(str,"width") !=NULL ){
651:     sprintf(value_str,"%d",width);
652:     replace_string(str,"width",value_str);
653:   }
654: }
655:
656: int allocate_equa(char **equation1,char **equation2,char var_ch)
657: { int var_index;
658:   var_index=var_ch-ASC_a;
659:   if( toupper( equ_alloc[var_index])!='Y' ){
660:     *equation1= (char far *) farmalloc(MAXSTRING*sizeof(char));
661:     *equation2= (char far *) farmalloc(MAXSTRING*sizeof(char));
662:     if( *equation2==NULL){
663:       ErrorMessage(0,"[allocate_equa]: Cannot allocate lex_str.");
664:       equ_alloc[var_index]='.';
665:       return(1); /** ERROR **/
666:     }
667:     else{
668:       equ_alloc[var_index]='Y';
669:       return(0);
670:     }
671:   }
672: }
673:
674: int assign_var_detail(int equa_index,char var_ch)
675: { if( var_alloc[var_ch-ASC_a] !='Y' )
676:   if( allocate_var( var_ch ) ){
677:     ErrorMessage(0,"[assign_var_detail]: allocate Error");
678:     return(1);
679:   }
680:   PolishToken(pol_tok[var_ch-ASC_a],var_tok[var_ch-ASC_a],
681:               lex_str[equa_index]);
682:   return(0);
683: }
684:
685:
686: /***** 13 *****/
687: int Integrate(char *equation_str)
688: { Value_type var_val[26],integrate_value,time,time_step;
689:   Token_type equ_token[MAXTOKEN];
690:   Var_type equ_vartype[MAXTOKEN];
691:   char var_exist[27],*var_ptr,record_str[26],value_str[80];
692:   int index,var_index,var_ind,var_len;
693:
694:   PolishToken(equ_token,equ_vartype,equation_str);
695:   sprintf(record_str,"%s%c",rec_var_str,'t');
696:   if(RecExistString(equ_vartype,var_exist,record_str))
697:     return(1);
698:   integrate_value=0;
699:   time=0;
700:   time_step=time_range/width;
701:   var_len=strlen(var_exist);
702:   for(index=0; index<width; index++){
703:     for(var_index=0; var_index<var_len; var_index++){
704:       var_ind=var_exist[var_index];
705:       if( var_ind!='t' ){
706:         var_ind-=ASC_a;
707:         var_val[var_ind]= *( rec_var[var_ind] +index );
708:       }
709:       else
710:         var_val['t'-ASC_a]=time;
711:     }
712:     ReadVariables(var_val,equ_token,equ_vartype);
713:     integrate_value+=EvaluatePosfix(equ_token);
714:     time+=time_step;
715:   }
716:
717:   integrate_value*=time_step;
718:   Cprint("\n\r %3.2f",time_range);
719:   Cprint("\n\r <");
720:   Cprint("\n\r < (%s) dt = %3.2f",equation_str,integrate_value);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

721:  Cprint("\n\r0");
722:  return(0);
723: }
724:
725:
726: /***** 14 *****/
727: void plot_graph(char *parameter)
728: { FILE *fp;
729:   char *var_ptr,variable_str[26];
730:
731:   if( strcmp(strupr(parameter),"FILE")==0){
732:     ManyVarPlot(FILE_NAMES);
733:   }
734:   else{
735:     if( (fp=fopen(parameter,"rt"))!=NULL ){
736:       fclose(fp);
737:       ManyVarPlot(parameter);
738:     }
739:     else if (ready_run==1){
740:       if( (parameter[0]=='#') && (parameter[1]=='#') ){
741:         var_ptr=parameter+2;
742:         PlotVariable(rec_var,var_ptr,var_ptr,width,
743:           time_range,0/*init_time*/);
744:         /* plot all variable beyond "##" */
745:       }
746:       else{
747:         if( strcmp(strupr(parameter),"RECORD")==0)
748:           strcpy(variable_str,rec_var_str);
749:         else
750:           strcpy(variable_str,var_str+1);
751:         PlotVariable(rec_var,rec_var_str,variable_str,width,
752:           time_range,0/*init_time*/);
753:       }
754:     }
755:   }
756: }
757:
758:
759: int PlotVariable(Value_type **rec_var,char *rec_var_str,char *var_str,
760:   int width,Value_type time_range,Value_type init_time)
761: { int max_width,i;
762:   char plot_str[27];
763:   Value_type *DATA[26],time_period;
764:
765:   if( rec_var_str[0]=='\0' || var_str[0]=='\0' )
766:     return(1);
767:   for(i=0; i<26; i++){
768:     DATA[i]=rec_var[i];
769:   }
770:   max_width=width;
771:   strcpy(plot_str,rec_var_str);
772:   intersect_str(plot_str,var_str);
773:   if( (time_period=MAXTIMEDOMAIN*time_range/width) >time_range)
774:     time_period=time_range;
775:   while(max_width>0){
776:     if( max_width>600 )
777:       width=600;
778:     else{
779:       width=max_width;
780:       time_period*=max_width/MAXTIMEDOMAIN;
781:     }
782:     PlotVarMem(DATA,plot_str,plot_str,width,time_period,init_time);
783:     for(i=0; i<26; i++){
784:       DATA[i]+=width;
785:       max_width-=width;
786:       init_time+=time_period;
787:     }
788:   }
789: }
790: /***** 15 *****/
791: int var_declaration(char *parameter)
792: { char string[80],*str_p,ch1,ch2;
793:   Cprint("\n\rVariable Declaration.....");
794:   strcpy(string,get_only_char(parameter) );
795:   str_p=string;
796:   while( (ch1=toupper(*str_p)) !='\0' ){
797:     if(ch1!='t'){
798:       if( allocate_var(ch1) )
799:         return(1); /* mean -cannot allocate memory */
800:       ch2=toupper(*(str_p+1));

```

เอกสารนี้เป็นเอกสารสงวนลิขสิทธิ์ของสถาบันเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

801:         if( (ch2=='E') || (ch2=='S') ){
802:             equ_typ[ch1-ASC_a]=ch2;
803:             Cprint("%c%c/",ch1,ch2);
804:             str_p+=2;
805:         }
806:     else{
807:         ErrorMessage(0,"[var_declaration] Data Type Error");
808:         return(1); /* mean error */
809:     }
810: }
811: else
812:     str_p+=2;
813: }
814:
815: Cprint(".....received!!");
816: return(0); /* no error */
817: }
818:
819:
820:
821: int allocate_var(char ch1)
822: { if( var_alloc[ch1-ASC_a] != 'Y' ){
823:     var_tok[ch1-ASC_a] = (Var_type far *) farmalloc(MAXSTRING*sizeof(Var_type));
824:     pol_tok[ch1-ASC_a] = (Token_type far *) farmalloc(MAXTOKEN*sizeof(Token_type));
825:
826:     if( (pol_tok[ch1-ASC_a] == NULL) || (var_tok[ch1-ASC_a] == NULL)){
827:         ErrorMessage(0,"[allocate_var] : Cannot allocate pol_tok[.].");
828:         return(1); /* ERROR! */
829:     }
830:     else{
831:         var_alloc[ch1-ASC_a]='Y';
832:         return(0); /* no ERROR */
833:     }
834: }
835: else
836:     return(0);
837: }
838:
839: /***** 16 *****/
840: char *record(char *parameter)
841: /* if parameter[0] has been initiated by '##'
842: then this function will not free_rec_alloc() */
843: { char new_rec[80],new_rec_dest[80],message[40],*ch;
844:   int i;
845:   if(width==0) return(NULL);
846:   strlwr(parameter);
847:   if( (parameter[0]=='#') && (parameter[1]=='#') ){
848:       parameter+=2; /* Skip over "##" */
849:       sprintf(new_rec,"%s%s",rec_var_str,parameter);
850:   }
851:   else{
852:       strcpy(new_rec,strlwr(get_only_char(parameter)) );
853:       free_rec_alloc();
854:   }
855:   new_rec_dest[0]='\0';
856:   for(i=0,ch=new_rec; *ch!='\0'; ch++)
857:       switch( alloc_rec(*ch,width) ) {
858:       case 1: sprintf(message,"[record] : Variable (%c) cannot allocate",
859:                               new_rec[i-1]);
860:               ErrorMessage(0,message);
861:               new_rec[i]='\0';
862:               ch=&new_rec[i]; /* Assign for breaking the loop */
863:               break;
864:       case 2: /* case 2 means the allocation had been made before */
865:               if( strchr(new_rec_dest,*ch) ==NULL ){
866:                   new_rec_dest[i]=*ch; /* Appen *ch into new_rec_dest */
867:                   i++;
868:               }
869:               break;
870:       case 0: new_rec_dest[i]=*ch;
871:               i++; /* Allocation complete */
872:               break;
873:       }
874:   new_rec_dest[i]='\0';
875:   Cprint("\n\rRecord Variable : %s....received!!",new_rec_dest);
876:   return(new_rec_dest);
877: }
878:
879:
880: int alloc_rec(char ch,int width)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 ในวารณใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
881: { if( toupper(rec_alloc[ch-ASC_a])!='Y' ){
882:     rec_var[ch-ASC_a]=(Value_type far *) farmalloc(width*sizeof(Value_type));
883:     if(rec_var[ch-ASC_a]==NULL){ /** Cannot Allocate **/
884:         ErrorMessage(0, "[alloc_rec] : Cannot Allocate Memory");
885:         return(1); /* ERROR */
886:     }
887:     else{
888:         rec_alloc[ch-ASC_a]='Y';
889:         clear_memory(rec_var[ch-ASC_a],width);
890:         return(0); /* no ERROR */
891:     }
892: }
893: return(2);
894: }
895:
896: void clear_memory(Value_type *pointer,int width)
897: { int i;
898:   for(i=0; i<width; i++,pointer++)
899:     *pointer=0;
900: }
901:
902:
903: /***** 17 *****/
904: int save_data(char *file_name)
905: { char f_name[50],var_save[30],*ptr;
906:   if(ready_run==1){
907:     if(rec_var_str[0]=='\0') return(1);
908:     if( (ptr=strstr(file_name,"##")) != NULL ){
909:       *ptr='\0'; ptr+=2; /* skip over "##" */
910:       strcpy(var_save,file_name);
911:       intersect_str(var_save,rec_var_str);
912:       strcpy(f_name,ptr);
913:     }
914:     else{
915:       strcpy(f_name,file_name);
916:       strcpy(var_save,rec_var_str);
917:     }
918:     if( f_name[0]=='\0' ){
919:       if( strchr(FILE_NAMES,':') !=NULL)
920:         strcpy(f_name,FILE_NAMES);
921:       else
922:         sprintf(f_name,"%s%s",Drive,FILE_NAMES);
923:     }
924:     WriteHeaderFile(f_name);
925:     WriteRecVarToFile(f_name,width,var_save);
926:   }
927:   else
928:     ErrorMessage(0, "[save_data]: System is not processed,yet.");
929: }
930:
931:
932: /***** 18 *****/
933: int FFT(char *var_ch)
934: { int i,j,return_val=0;
935:   Value_type *DATA_PNTR[2],*spectrum,*An,*Bn;
936:   char rec_para[30];
937:   if(var_ch[0]=='\0') return(1);
938:   if(ready_run==0){
939:     ErrorMessage(0, "[FFT]: Process not executed.");
940:     return(1);
941:   }
942:   strlwr(var_ch);
943:   get_only_char(var_ch);
944:   if( strlen(var_ch)>3 ){
945:     ErrorMessage(0, "[FFT]: Number of Variables Exceeds 3.");
946:     /* var_ch[0] => Signal
947:        var_ch[1] => An / spectrum
948:        var_ch[2] => Bn
949:        SO, The Number of Variables Must not Exceeds 3. */
950:     return(1);
951:   }
952:   sprintf(rec_para,"##%s",var_ch);
953:   strcpy( rec_var_str,record(rec_para) );
954:   if(var_ch[0]=='\0'){
955:     ErrorMessage(0, "[FFT]: Variable not recorded.");
956:     return(1);
957:   }
958:   An=(Value_type *)malloc( width*sizeof(Value_type) );
959:   Bn=(Value_type *)malloc( width*sizeof(Value_type) );
```

เอกสารนี้เป็นเอกสารสงวนไว้สำหรับการใช้ภายในหน่วยงานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งยังมีเหตุเปลี่ยนแปลงสิ่งนี้ และต้องอยู่ใต้อำนาจของเอกสารทุกครั้งที่มีการนำไปใช้

```

961: spectrum=(Value_type *)malloc( width*sizeof(Value_type) );
962: if(spectrum==NULL){
963:     if(An==NULL) free(An);
964:     if(Bn==NULL) free(Bn);
965:     ErrorMessage(0,"[FFT]: Cannot Allocate Memory.");
966:     return(1);
967: }
968: DATA_PNTR[0]=rec_var[tolower(*var_ch)-'a'];
969: DATA_PNTR[1]=spectrum;
970:
971: i=DETECT;
972: Fourier(rec_var[tolower(*var_ch)-'a'],width,An,Bn);
973: Modulus(spectrum,width,An,Bn);
974: initgraph(&i,&j,"");
975: PlotVarMem(DATA_PNTR,"ab","ab",width, time_range, 0);
976: closegraph();
977: if( strlen(var_ch)>1 ){
978:     if(strlen(var_ch)==2){
979:         AssignValue(var_ch[1],spectrum,width);
980:         return_val=1;
981:     }
982:     else{
983:         AssignValue(var_ch[1],An,width);
984:         AssignValue(var_ch[2],Bn,width);
985:     }
986: }
987: free(An);
988: free(Bn);
989: free(spectrum);
990: return(return_val);
991: }
992:
993: int AssignValue(char var_dest,Value_type *source,int width)
994: /* var_dest points to name of variable that will be destination */
995: {
996:     if(strchr(rec_var_str,var_dest)==NULL){
997:         ErrorMessage(0,"[AssignValue]: Dest not Allocated");
998:         return(1);
999:     }
1000:     else
1001:         CopyMemory(rec_var[var_dest-ASC_a],source,width);
1002:     return(0);
1003: }
1004:
1005: void CopyMemory(Value_type *dest,Value_type *source,int width)
1006: {
1007:     memcpy(dest,source,sizeof(Value_type)*width);
1008: }
1009:
1010: /****** 19 *****/
1011: int IFFT(char *parameter)
1012: /* take Fast Fourier Transform and then take inverse FFT */
1013: { Value_type filt1,filt2;
1014:   char var_para[80],rec_para[80],filter_para[80],
1015:         filter_para1[80],filter_para2[80];
1016:   int i,j,high_cut,low_cut;
1017:
1018:   if(ready_run==1){
1019:       parse(var_para,filter_para,parameter);
1020:       get_only_char(var_para);
1021:       strlwr(var_para);
1022:       if(strlen(var_para)<3){
1023:           ErrorMessage(0,"[IFFT]:IFFT fails,Parameter incomplete.");
1024:           return(1);
1025:       }
1026:       if( rec_var[var_para[1]-ASC_a]==NULL || rec_var[var_para[2]-ASC_a]==NULL ){
1027:           ErrorMessage(0,"[IFFT]: Variable not recorded");
1028:           return(1);
1029:       }
1030:       parse(filter_para1,filter_para2,filter_para);
1031:       filt1=atof(filter_para1);
1032:       filt2=atof(filter_para2);
1033:       if((filt1>=0)&&(filt1<=1) && (filt2>=0)&&(filt2<=1) &&
1034:          (filt2>filt1)){
1035:           low_cut=(int) (width*filt1);
1036:           high_cut=(int) (width*filt2);
1037:       }
1038:       else{
1039:           ErrorMessage(0,"[IFFT]:Cut-off freq,not Conform Format.");
1040:           return(1);

```

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1041:     }
1042:     filter(width,low_cut,high_cut,rec_var[*var_para+1]-ASC_a,
1043:           rec_var[*var_para+2]-ASC_a]);
1044:     InvFourier(rec_var[*var_para-ASC_a],width,
1045:           rec_var[*var_para+1]-ASC_a,rec_var[*var_para+2]-ASC_a]);
1046:     /* var_para=> reconstructed signal
1047:        var_para+1=> An
1048:        var_para+2=> Bn */
1049:     plot_graph(rec_para);
1050:     return(0);
1051: }
1052:     ErrorMessage(0,"[IFFT]:Process not exeuted,yet.");
1053:     return(1);
1054: }
1055:
1056: /***** 20 *****/
1057: int TellTime(int mode)
1058: /* mode 2: for first start
1059:    mode 1: show the elapsed(?) time
1060:    mode 0: tell real time */
1061: { struct time tm;
1062:   struct date dt;
1063:   static long time_set=0;
1064:   long time_set1;
1065:   int text[500];
1066:
1067:   gettimeofday(&tm);
1068:   getdate(&dt);
1069:   if(time_set==0)
1070:     time_set=24*3600*dt.da_day+3600*tm.ti_hour+60*tm.ti_min+tm.ti_sec;
1071:   if(mode==2){
1072:     time_set=24*3600*dt.da_day+3600*tm.ti_hour+60*tm.ti_min+tm.ti_sec;
1073:     return(0);
1074:   }
1075:   Fastgettext(27,7,49,12,text);
1076:   makewin(27,7,49,12,0x2E,0x20,1);
1077:
1078:   switch(mode){
1079:     case 0: Fastputs(31,8,0x20,"      TIME      ");
1080:             Fastprintf(33,10,0x21,"%d : %d : %d",tm.ti_hour,tm.ti_min,
1081:                       tm.ti_sec);
1082:             Fastputs(31,11,0x20,"      hr.      ");
1083:             break;
1084:     case 1: Fastputs(31,8,0x21,"    PROCESSED    ");
1085:             Fastputs(31,9,0x20,"      TIME      ");
1086:             time_set1=3600*24*dt.da_day+3600*tm.ti_hour+
1087:                     60*tm.ti_min+tm.ti_sec;
1088:             time_set1-=time_set;
1089:             Fastprintf(33,11,0x21,"%1ld Seconds",time_set1);
1090:             break;
1091:     case 3: ErrorMessage(0,"Press Any Key to Continue...");
1092:             break;
1093:     case 4: ErrorMessage(1,"Press Any Key to Continue...");
1094:             break;
1095:   }
1096:   getch();
1097:   Fastputtext(27,7,49,12,text);
1098: }
1099:
1100: /***** 21 *****/
1101: void FreeAndSet(void)
1102: { free_memory();
1103:   set_parameter();
1104: }
1105:
1106:
1107: /***** 22 *****/
1108: int Assign(char *parameter)
1109: { char var_ch,equ_str[80],rec_para[4];
1110:   if( parameter[0]!='\0' ) return(0);
1111:   rid_a_space(parameter);
1112:   var_ch=*parameter;
1113:   strcpy( equ_str,get_str(1,',','\0',parameter) );
1114:   if( rec_var[var_ch-ASC_a]==NULL ){
1115:     sprintf(rec_para,"##%c",var_ch);
1116:     strcpy( rec_var_str,record(rec_para) );
1117:   }
1118:   SetValEqu(equ_str, rec_var[var_ch-ASC_a] , width);
1119: }
1120:

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
1121:
1122: int SetValEqu(char *equation_str, Value_type *DATA, int width)
1123: /* set DATA from function of record variable */
1124: {
1125:     Value_type var_val[26],*data_pntr;
1126:     Token_type equ_token[MAXTOKEN];
1127:     Var_type    equ_vartype[MAXTOKEN];
1128:     char        var_exist[27],*var_pntr;
1129:     int         index,var_index,var_ind,var_len;
1130:
1131:     if(PolishToken(equ_token,equ_vartype,equation_str))
1132:         return(1);
1133:     if(RecExistString(equ_vartype,var_exist,rec_var_str))
1134:         return(1);
1135:     var_len=strlen(var_exist);
1136:     data_pntr=DATA;
1137:     for(index=0; index<width; index++){
1138:         for(var_index=0; var_index<var_len; var_index++){
1139:             var_ind=var_exist[var_index]-ASC_a;
1140:             var_val[var_ind]= *( rec_var[var_ind] +index );
1141:         }
1142:         ReadVariables(var_val,equ_token,equ_vartype);
1143:         (*data_pntr)=EvaluatePosfix(equ_token);
1144:         data_pntr++;
1145:     }
1146:     return(0);
1147: }
1148:
1149: int RecExistString(Var_type *equ_vartype,char *var_exist,char *rec_var_str)
1150: /* return 1 : error mean there exist a variable in equation_str that not
1151:    exist in rec_var_str
1152:    var_exist : is passed by reference and store all variable that exist
1153:    in equation_str */
1154: { char *var_pntr;
1155:   int  var_index,var_len;
1156:
1157:   var_pntr=var_exist;
1158:   var_index=0;
1159:   while( equ_vartype[var_index].name_index != -1 ){
1160:       *var_pntr=equ_vartype[var_index].name_index-FIRSTOPERAND+ASC_a;
1161:       var_index++;
1162:       var_pntr++;
1163:   }
1164:   *var_pntr='\0';
1165:   var_pntr=var_exist;
1166:   var_len=strlen(var_exist);
1167:   for(var_index=0; var_index<var_len; var_index++){
1168:       if( strchr(rec_var_str,var_exist[var_index])!=NULL ){
1169:           /* There are some variable that have no record */
1170:           ErrorMessage(0,"[SetValEqu]: Some Var not Assigned,yet.");
1171:           return(1); /* error case */
1172:       }
1173:   }
1174:   return(0); /* no error */
1175: }
1176: +
```

```

1 : /*****
2 : FILE NAME : 1STRING.C
3 : LEVEL : 1
4 : SUPPORT : SIMULATE.C
5 : REQUIRE : I_DEFINE.H
6 : *****/
7 :
8 : void ErrorMessage(int mode,char *message);
9 : int ParseManyString(char *source,char **dest);
10 : void parse(char *command,char *parameter, char *string_com);
11 : void ParseTwoString(char *source1,char *dest1,char *dest2);
12 : void replace_string(char *string,char *sub_string,char *replace_str);
13 : int InsertStr(char *format);
14 : int InsertChar(char *start_pos);
15 : void get_string_from_keyboard(char *string_com);
16 : int table_compare(char *command);
17 : int table_index(char *string,char **str_table,int max_index);
18 : char *intersect_str(char *domain,char *boundary);
19 : void rid_a_space(char *string);
20 : void del_ch_from_str(char *source,char ch);
21 : char *get_only_char(char *str);
22 : int CheckType(char ch);
23 :
24 :
25 : void ErrorMessage(int mode,char *message)
26 : #define MAX_MESSAGE 40
27 : { switch(mode){
28 :     case 0 : if( strlen(message)>=MAX_MESSAGE )
29 :                 message[MAX_MESSAGE-1]='\0';
30 :                 Show_Get_message(15,15,0x4E,message,0);
31 :                 break;
32 :     case 1 : printf("\n\r%s",message);
33 :                 getch();
34 :                 break;
35 : }
36 : }
37 :
38 : int ParseManyString(char *source,char **dest)
39 : /* return number of string that can be parsed */
40 : /* dest will be array of pointer to string : *dest[] */
41 : { char dest_temp[80];
42 :   char source1[80];
43 :   int num_string=0;
44 :   strcpy(source1,source);
45 :   if(*source=='\0')
46 :       return(0);
47 :   do{
48 :       ParseTwoString(source1,*dest,dest_temp);
49 :       dest++;
50 :       num_string++;
51 :       strcpy(source1,dest_temp);
52 :   }while( *dest_temp!='\0' );
53 :   return(num_string);
54 : }
55 :
56 : void parse(char *command,char *parameter,char *string_com)
57 : {
58 :     ParseTwoString(string_com,command,parameter);
59 : }
60 :
61 :
62 : void ParseTwoString(char *source1,char *dest1,char *dest2)
63 : /* format of source1: SPACE(n)dest1(1)SPACE(1)dest2(1) */
64 : { int i,k;
65 :   char *source;
66 :   source=source1;
67 :   while((*source)==' ')
68 :       source++;
69 :   dest1[0]='\0';
70 :   dest2[0]='\0';
71 :   i=0;
72 :   while( (source[i]!=' ') && (source[i]!='\0') ){
73 :       dest1[i]=source[i];
74 :       i++;
75 :   }
76 :   dest1[i]='\0';
77 :   while( source[i+1]==' ' )
78 :       i++;
79 :   if(source[i]!='\0'){
80 :       i++;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :     k=0;
82 :     while(source[i] != '\0'){
83 :         dest2[k] = source[i];
84 :         i++;
85 :         k++;
86 :     }
87 : }
88 :     dest2[k]='\0';
89 :     strupr(dest1);
90 :     strupr(dest2);
91 : }
92 :
93 :
94 : int InsertStr(char *format)
95 : /* all character that preceed a number will be replaced by the character
96 :  in the number of times */
97 : /* return the length of format that already processed */
98 : { char *fmt_ptr;
99 :   if(*format=='\0')
100 :     return(0);
101 :   fmt_ptr=format+1;
102 :   while( (*fmt_ptr)!='\0' ){
103 :     if( CheckType(*fmt_ptr)==2)
104 :       if( CheckType(*(fmt_ptr-1))==1 )
105 :         InsertChar(fmt_ptr-1);
106 :     fmt_ptr++;
107 :   }
108 :   return(strlen(format));
109 : }
110 :
111 : int InsertChar(char *start_pos)
112 : /* insert the character pointed by start_pos in the given times */
113 : /* number of times specified by *(start_pos+1) */
114 : /* return number of times or *(start_pos+1) */
115 : {
116 :   char insert_char,*str_ptr,*last_pos;
117 :   int time_number,i,str_length;
118 :
119 :   insert_char=*start_pos;
120 :   time_number=*(start_pos+1)-'0';
121 :   if(time_number==0)
122 :     return(0);
123 :   str_length=strlen(start_pos+2);
124 :   str_ptr=start_pos;
125 :   while( (*str_ptr)!='\0')
126 :     str_ptr++;
127 :   last_pos=str_ptr-2+time_number;
128 :   for(i=0; i<str_length; i++){
129 :     *last_pos=*str_ptr;
130 :     last_pos--;
131 :     str_ptr--;
132 :   }
133 :   for(i=0; i<time_number; i++){
134 :     *start_pos=insert_char;
135 :     start_pos++;
136 :   }
137 :   return(time_number);
138 : }
139 :
140 :
141 :
142 : void get_string_from_keyboard(char *string_com)
143 : { int j=0;
144 :   string_com[j] = toupper(getch());
145 :   while(string_com[j] == BACK_SPACE)
146 :     string_com[j]=toupper(getch());
147 :   if(string_com[j] != CARRAGE_RETURN){
148 :     putchar(string_com[j++]);
149 :     string_com[j] = toupper(getch());
150 :   }
151 :
152 :   while( string_com[j] != CARRAGE_RETURN){
153 :     if( string_com[j] == BACK_SPACE ){
154 :       putchar(BACK_SPACE);
155 :       putchar(' ');
156 :       putchar(BACK_SPACE);
157 :       j-=2;
158 :     }
159 :     else
160 :       putchar(string_com[j]);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161:     j++;
162:     string_com[j]=toupper(getch());
163: };
164: printf("\n");
165: string_com[j] = '\0';
166: }
167:
168: int table_compare(char *command)
169: {   int i=0,equal=1;
170:     /***** can call table_index() *****/
171:     while( (i<MAX_COMMAND) && (equal!=0) )
172:     {   equal=strcmp(command,command_table[i][0]);
173:         i++;
174:     };
175:     if(equal==0) return(i-1);
176:     else return(MAX_COMMAND);
177:     /*****/
178: }
179:
180: int table_index(char *string,char **str_table,int max_index)
181: {   int i=0,equal=1;
182:     while( (i<max_index) && (equal!=0) )
183:     {equal=strcmp(string,*str_table);
184:       (*str_table)++;
185:       i++;
186:     };
187:     if(equal==0) return(i-1);
188:     else return(max_index);
189: }
190:
191:
192: void replace_string(char *string,char *sub_string,char *replace_str)
193: { char *ptr,str[80],strings[80];
194:   int sub_str_len;
195:
196:   sub_str_len=strlen(sub_string);
197:   while( (ptr=strstr(string,sub_string)) != NULL){
198:     strcpy(str,ptr+sub_str_len);
199:     *ptr='\0';
200:     sprintf(strings,"%s%s",string,replace_str,str);
201:     }
202:   strcpy(string,strings);
203: }
204:
205:
206: char *intersect_str(char *domain,char *boundary)
207: /* return "domain" that overlaps "boundary" */
208: /* return the intersection between two strings */
209: { char *var_pntr;
210:   if( *domain=='\0') return(NULL);
211:   if(*boundary=='\0') return(NULL);
212:   strlwr(domain);
213:   strlwr(boundary);
214:   var_pntr=domain;
215:   while( (*var_pntr)!='\0' ){
216:     if( strcmp(strchr(boundary,*var_pntr),NULL)==0 ){
217:       del_ch_from_str(domain,*var_pntr);
218:       var_pntr--;
219:     }
220:     var_pntr++;
221:   }
222:   return(domain);
223: }
224:
225: void rid_a_space(char *string)
226: { char *c;
227:   int i=0;
228:   c=string;
229:   while( (*c)!='\0' ){
230:     if( *c!=' ' ) string[i++]=*c;
231:     c++;
232:   }
233:   string[i]='\0';
234: }
235:
236: void del_ch_from_str(char *source,char ch)
237: /* all ch will be remove */
238: { char *dest;
239:   dest=source;
240:   while((*source)!='\0'){

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
241:     if( (*source)!=ch ){
242:         *dest=*source;
243:         dest++;
244:     }
245:     source++;
246: }
247: *dest='\0';
248: }
249:
250: char *get_only_char(char *str)
251: { char ch1,string[80];
252:   int i=0,j=0;
253:   while(str[i]!='\0'){
254:       if( ((ch1=tolower(str[i])) >='a') && (ch1<='z') )
255:           string[j++]=ch1;
256:       i++;
257:   }
258:   string[j]='\0';
259:   return(string);
260: }
261:
262:
263: int CheckType(char ch)
264: /* check : which type is ch? */
265: /* return 1 if ch is a..z or A..Z ,
266:    return 2 if ch is number 0..9
267:    otherwise 0 */
268: { ch=tolower(ch);
269:   if( (ch>='a') && (ch<='z') )
270:       return(1);
271:   else if( (ch>='0') && (ch<='9') )
272:       return(2);
273:   else
274:       return(0);
275: }
276: }
277: ↑
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : /*****
2 :   FILE NAME : 1FILE.C
3 :   SUPPORT   : 1PLOTGPH.C
4 :               2MANYPLT.C
5 :               SIMULATE.C
6 :   REQUIRE   : ErrorMessage();
7 : *****/
8 :
9 : #define DATA_LENGTH 12
10 : #define MAX_COLUMN 80
11 : #define MAX_DATA_P_LINE (int)(MAX_COLUMN/DATA_LENGTH)
12 :
13 : extern Value_type time_range,var_val[26];
14 : extern int iteration_scale,num_equation;
15 : extern char far *equation_str[MAX_EQUATION];
16 : extern char equ_typ[27];
17 : extern char equ_alloc[27],rec_alloc[27];
18 :
19 :
20 : Value_type GetTimeFile(char *FILE_NAMES);
21 : int GetRecFile(char *FILE_NAMES,char *rec_var_str);
22 : int GetParameterFile(FILE *fp,char *parameter,char *find_word);
23 : void SkipSpaces(char *dest,char *source);
24 : char *FileSearch(FILE* fp,char *find_word);
25 : char *get_str(int begin_pos,char begin_ch,char final_ch, char *string);
26 : int GetDataFromFile(char *FILE_NAMES,char rec_var_name,Value_type *DATA);
27 : int WriteHeaderFile(char *FILE_NAMES);
28 : int WriteArrayToFile(char *FILE_NAMES,Value_type *DATA,int num_data,
29 :                      char var_name);
30 : void set_f_name(char *FILE_NAMES,char *Drive);
31 : void show_data(char *parameter,char *FILE_NAMES);
32 :
33 : FILE *fp;
34 :
35 :
36 : Value_type GetTimeFile(char *FILE_NAMES)
37 : { char string[80];
38 :   FILE *fp;
39 :   fp=fopen(FILE_NAMES,"rt");
40 :   if(fp==NULL) return(0);
41 :   if(GetParameterFile(fp,string,"Time")==1){ /* not found */
42 :     fclose(fp);
43 :     return(0);
44 :   }
45 :   fclose(fp);
46 :   return(atof(string));
47 : }
48 :
49 : Value_type GetSizeFile(char *FILE_NAMES)
50 : { char string[80];
51 :   FILE *fp;
52 :   fp=fopen(FILE_NAMES,"rt");
53 :   if(fp==NULL) return(0);
54 :   if(GetParameterFile(fp,string,"SIZE")==1){ /* not found */
55 :     fclose(fp);
56 :     return(0);
57 :   }
58 :   fclose(fp);
59 :   return(atof(string));
60 : }
61 :
62 : int GetRecFile(char *FILE_NAMES,char *rec_var_str)
63 : { FILE *fp;
64 :   char str[80],string[80],*var_ptr,*temp_ptr;
65 :
66 :   fp=fopen(FILE_NAMES,"rt");
67 :   if(fp==NULL){
68 :     rec_var_str[0]='\0';
69 :     return(0);
70 :   }
71 :
72 :   var_ptr=rec_var_str;
73 :   rewind(fp);
74 :   temp_ptr=FileSearch(fp,"DATA");
75 :   while(temp_ptr!=NULL){
76 :     strcpy(string,temp_ptr);
77 :     SkipSpaces(string);
78 :     *var_ptr=string[0];
79 :     temp_ptr=FileSearch(fp,"DATA");
80 :     var_ptr++;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :     }
82 :     *var_pntr='\0';
83 :
84 :     fclose(fp);
85 :     return(strlen(rec_var_str));
86 : }
87 :
88 : int GetParameterFile(FILE *fp,char *parameter,char *find_word)
89 : { char string[80],str[80],*ptr;
90 :   if( (ptr=FileSearch(fp,find_word))==NULL)
91 :     return(1);
92 :   memcpy(string,ptr,80);
93 :   strcpy(str,get_str(1,':','\0',string) );
94 :   SkipSpaces(parameter,str);
95 :   return(0);
96 : }
97 :
98 :
99 : void SkipSpaces(char *dest,char *source)
100 : { while((*source!='\0')){ /* skip over the space */
101 :   if(!isspace(*source)){
102 :     *dest=*source;
103 :     dest++;
104 :   }
105 :   source++;
106 : }
107 : *dest='\0';
108 : }
109 :
110 :
111 : char *FileSearch(FILE* fp,char *find_word)
112 : { char *ptr,string[80];
113 :   int find_data_position;
114 :
115 :   if(find_word[0]!='\0') return(NULL);
116 :   ptr=find_word; /* set initial for condition "ptr!=NULL" */
117 :   find_data_position=0;
118 :   while( (find_data_position==0) && (ptr!=NULL) ){
119 :     ptr=fgets(string,80,fp);
120 :     if( strstr(string,find_word)!=NULL)
121 :       find_data_position=1;
122 :   };
123 :   if(find_data_position==0)
124 :     return(NULL);
125 :   else
126 :     return(string);
127 : }
128 :
129 : char *get_str(int begin_pos,char begin_ch,char final_ch, char *string)
130 : { int i=0;
131 :   char *ch,str_gotten[80];
132 :   ch=string+begin_pos-1;
133 :   while(*ch != begin_ch)
134 :     ch++;
135 :   ch++;
136 :   while( (*ch != final_ch) && (*ch != '\0') ){
137 :     str_gotten[i++]=*ch;
138 :     ch++;
139 :   }
140 :   str_gotten[i]='\0';
141 :   return(str_gotten);
142 : }
143 :
144 : int GetDataFromFile(char *FILE_NAMES,char rec_var_name,Value_type *DATA)
145 : /* RETURN : number of data in file */
146 : { char string[80],*ptr,data_pntr[10],find_word[10];
147 :   int i,index;
148 :
149 :   if( (fp=fopen(FILE_NAMES,"rt"))==NULL ) return(0);
150 :   if(rec_var_name!='\0')
151 :     sprintf(find_word,"%cDATA\n",rec_var_name);
152 :   else
153 :     sprintf(find_word,"DATA\n");
154 :   ptr=FileSearch(fp,find_word);
155 :   if(ptr==NULL)
156 :     return(0); /* There are no data. */
157 :
158 :   index=0;
159 :   while( strlen( fgets(string,MAX_COLUMN,fp) ) >1){
160 :     ptr=string;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161:     for(i=0; (i<MAX_DATA_P_LINE) && (*ptr!=EOF); i++){
162:         memcpy(data_pntr, ptr, DATA_LENGTH);
163:         *DATA=atof(data_pntr);
164:         DATA++;
165:         index++;
166:         ptr+=DATA_LENGTH;
167:     }
168: }
169: fclose(fp);
170: return(index-MAX_DATA_P_LINE);
171: }
172:
173:
174: int WriteHeaderFile(char *FILE_NAMES)
175: { struct time tm;
176:   struct date ds;
177:   int i,check;
178:
179:   if( (fp=fopen(FILE_NAMES,"wt")) == NULL ){
180:       ErrorMessage(0,"[WriteHeaderFile]: Cannot open file!!");
181:       return(1);
182:   }
183:   getdate(&ds);
184:   gettime(&tm);
185:   fprintf(fp,"FILE NAME : %s\n",FILE_NAMES);
186:   fprintf(fp,"EXP. DATE : %d:%d:%d\n",ds.da_day,ds.da_mon,
187:                                                  ds.da_year);
188:   fprintf(fp,"EXP. TIME : %d:%d:%d\n",tm.ti_hour,tm.ti_min,
189:                                                  tm.ti_sec);
190:   fprintf(fp,"Time-Range: %e\n",time_range);
191:   fprintf(fp,"Precision : %e\n",iteration_scale);
192:   fprintf(fp,"Initial : ");
193:   for(i=0,check=0; i<26; i++)
194:       if(var_val[i]!=0){
195:           fprintf(fp,"%c=%-3.1f, ", 'a'+i);
196:           check=1;
197:       }
198:   if(check==0)
199:       fprintf(fp,"All states are initialized by zero\n");
200:   fprintf(fp,"System var: ");
201:   for(i=0; i<=25; i++)
202:       if(equ_alloc[i]=='Y') fprintf(fp,"%c", 'a'+i);
203:   fprintf(fp,"Equation type: ");
204:   fprintf(fp,"Expression: ");
205:   for(i=0; i<=25; i++)
206:       if(equ_typ[i]=='E') fprintf(fp,"%c", 'a'+i);
207:   fprintf(fp,"Diff. equ.: ");
208:   for(i=0; i<=25; i++)
209:       if(equ_typ[i]=='S') fprintf(fp,"%c", 'a'+i);
210:   fprintf(fp,"System Eq.: ");
211:   for(i=1; i<=num_equation; i++)
212:       fprintf(fp," (%d) : %s\n", i, equation_str[i]);
213:   fprintf(fp,"Record var: ");
214:   for(i=0; i<=25; i++)
215:       if(rec_alloc[i]=='Y')
216:           fprintf(fp,"%c", 'a'+i);
217:   fprintf(fp,"");
218:   fclose(fp);
219:   return(0);
220: }
221:
222:
223: int WriteArrayToFile(char *FILE_NAMES, Value_type *DATA, int num_data,
224:                     char var_name)
225: { int index, digit;
226:   Value_type datum;
227:   if( (fp=fopen(FILE_NAMES,"at")) == NULL )
228:       if( (fp=fopen(FILE_NAMES,"wt")) == NULL ){
229:           ErrorMessage(0,"[WriteArrayToFile]: Cannot open file, memory full!!");
230:           return(1);
231:       }
232:   if(num_data==0) return(1);
233:   fprintf(fp,"VARIABLE : %c\n", var_name);
234:   fprintf(fp,"SIZE : %d\n", num_data);
235:   fprintf(fp,"%cDATA\n", var_name);
236:   for(index=0; index<num_data; index++){
237:       datum=DATA[index];
238:       if(datum!=0)
239:           digit=(int)ceil( log10(fabs(datum)) );
240:       else digit=5; /*** %9.1f : 1 point ***/

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

241:     if(digit>2){
242:         if(digit>(DATA_LENGTH-2)){
243:             ErrorMessage(0,["WriteArrayToFile]: Overflow data\n");
244:             return(1);
245:         }
246:         if( (index+1)%MAX_DATA_P_LINE !=0 )
247:             fprintf(fp,"%12.0f",datum); /* 12 means DATA_LENGTH */
248:         else
249:             fprintf(fp,"%12.0f\n",datum); /* 12 means DATA_LENGTH */
250:     }
251:     else{
252:         if( (index+1)%MAX_DATA_P_LINE !=0 )
253:             fprintf(fp,"%12.7f",datum); /* 12 means DATA_LENGTH */
254:         else
255:             fprintf(fp,"%12.7f\n",datum); /* 12 means DATA_LENGTH */
256:     }
257: }
258: fprintf(fp,"\n\n");
259: fclose(fp);
260: return(0);
261: }
262:
263: void set_f_name(char *FILE_NAMES,char *Drive)
264: { char DATAFILE[50],SUR_NAME[4];
265:   int SUR_NUMBER;
266:   sprintf(DATAFILE,"%sDATA.000",Drive);
267:   if( (fp=fopen(DATAFILE,"rt"))==NULL ){
268:       fp=fopen(DATAFILE,"wt");
269:       fprintf(fp,"002");
270:       fclose(fp);
271:       sprintf(SUR_NAME,"001");
272:   }
273:   else{
274:       fread(SUR_NAME,3,1,fp);
275:       fclose(fp);
276:       fp=fopen(DATAFILE,"wt");
277:       SUR_NUMBER=atoi(SUR_NAME)+1;
278:       SUR_NUMBER=SUR_NUMBER%1000;
279:       fprintf(fp,"%03d",SUR_NUMBER);
280:       fclose(fp);
281:   };
282:   SUR_NAME[3]='\0';
283:   sprintf(FILE_NAMES,"%sDATA.%s",Drive,SUR_NAME);
284: }
285:
286: void show_data(char *parameter,char *FILE_NAMES)
287: {int c,line_count=1,LINE=18;
288:  char ch,f_name[30],str[80];
289:
290:  for(c=0; c<70; c++) str[c]='_';
291:  str[70]='\0';
292:  if(parameter[0]=='\0')
293:      strcpy(f_name,FILE_NAMES);
294:  else
295:      strcpy(f_name,parameter);
296:  if( (fp=fopen(f_name,"rt")) != NULL){
297:      cprintf("\n\r\n\r%s",str);
298:      cprintf("\n\r FILE NAME : %s ",f_name);
299:      cprintf("\n\r%s\n\r",str);
300:      while( (c=getc(fp)) != EOF ){
301:          if(c!='\t'){
302:              if(c=='\n'){
303:                  putchar('\r');
304:                  line_count++;
305:              }
306:              putchar(c);
307:          }
308:          else cprintf(" ");
309:
310:          if(line_count%LINE==0){
311:              cprintf("\n\r( Press Any Key to Continue....)");
312:              ch=getch();
313:              if( ch!=CR) /* CR=carrage return */
314:                  if( (ch=getch()) ==ESC){
315:                      fclose(fp);
316:                      break;
317:                  }
318:              cprintf("\n\r");
319:              LINE=21;
320:              line_count=1;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้คัดลอกเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
321:         }
322:     }
323:     fclose(fp);
324: }
325: else{
326:     cprintf("\n\r    Cannot find FILE NAME : %s ",f_name);
327:     cprintf("\n\r    Press Any Key....");
328: }
329: }
330: ↑
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : /*****
2 :   FILE NAME : 1SETPARA.C
3 :   SUPPORT   : SIMULATE.C,MAINP.C
4 :   REQUIRE   : (ErrorMessage NOT exist)
5 : *****/
6 :
7 : void set_file(char *parameter,char *FILE_NAMES,char *Drive);           /*4*/
8 : void set_drive(char *parameter,char *Drive);                           /*5*/
9 : void set_time_range(char *parameter,Value_type *time_range);           /*8*/
10 : void set_precision_multiply(char *parameter,int *iteration_scale);       /*10*/
11 : void set_initial_state(char *parameter,Value_type *var_val);           /*11*/
12 :
13 :
14 : /***** 4 *****/
15 : void set_file(char *parameter,char *FILE_NAMES,char *Drive)
16 : { char F_NAMES[80];
17 :   if(parameter[0]!='\0')
18 :     strcpy(parameter,FILE_NAMES);
19 :   if( ( strchr(parameter,':') ==NULL) && ( strchr(parameter,'\\') ==NULL))
20 :     sprintf(F_NAMES,"%s%s",Drive,parameter);
21 :   else
22 :     strcpy(F_NAMES,parameter);
23 :
24 :   cprintf("\n\rFILE NAME : %s received!!!",F_NAMES);
25 :   strcpy(FILE_NAMES,F_NAMES);
26 : }
27 :
28 : /***** 5 *****/
29 : void set_drive(char *parameter,char *Drive)
30 : {int len;
31 :   if(parameter[0] == '\0')
32 :     cprintf("\n\rCURRENT DRIVE : %s",Drive);
33 :   else{
34 :     strcpy(Drive,parameter);
35 :     len=strlen(Drive);
36 :     if(Drive[len-1]!='\\'){
37 :       Drive[len]='\\';
38 :       Drive[len+1]='\0';
39 :     }
40 :     cprintf("\n\rCURRENT DRIVE : %s recieved!!!",Drive);
41 :   }
42 : }
43 :
44 :
45 :
46 : /***** 8 *****/
47 : void set_time_range(char *parameter,Value_type *time_range)
48 : { Value_type value;
49 :   if( parameter[0]!='\0' )
50 :     cprintf("\n\rTime-range : %6f",*time_range);
51 :   else{
52 :     if( (value=atof(parameter)) >0 )
53 :       *time_range=value;
54 :     cprintf("\n\rTime-range : %6f ...received !!!",*time_range);
55 :   }
56 : }
57 :
58 : /***** 10 *****/
59 : void set_precision_multiply(char *parameter,int *iteration_scale)
60 : {int value;
61 :   if( parameter[0]!='\0' )
62 :     cprintf("\n\rPrecision multiply: %3d", *iteration_scale);
63 :   else{
64 :     if( (value=atoi(parameter)) <=0 )
65 :       cprintf("\n\rPrecision scale: %3d",*iteration_scale);
66 :     else{
67 :       *iteration_scale=value;
68 :       cprintf("\n\rPrecision scale: %3d ...received!!!",*iteration_scale);
69 :     }
70 :   }
71 : }
72 :
73 : /***** 11 *****/
74 : void set_initial_state(char *parameter,Value_type *var_val)
75 : { char *str_p,ch1,val_str[MAXSTRING];
76 :   int i;
77 :   cprintf("\n\rInitial : ");
78 :   str_p=parameter;
79 :   ch1=tolower(*str_p);
80 :   if( (ch1<='z') && (ch1>='a' ) )

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
81 :     while(ch1!='\0'){
82 :         str_p++; i=0;
83 :         while( ch1>'z' || ch1<'a' ){
84 :             ch1=tolower(*str_p);
85 :             str_p++;
86 :         }
87 :         cprintf("%c=",ch1);
88 :         while(( *str_p>='0') && (*str_p<='9') || (*str_p=='-') || (*str_p=='+') ||
89 :             (*str_p=='.')) {
90 :             val_str[i++]=*str_p;
91 :             str_p++;
92 :         }
93 :         val_str[i]='\0';
94 :         var_val[ch1-ASC_a]=atof(val_str);
95 :         cprintf("%-6.1f /",var_val[ch1-ASC_a]);
96 :         ch1=tolower(*str_p);
97 :     }
98 :     if( wherex()>65)
99 :         cprintf("\n\r....received!!");
100 :     else
101 :         cprintf("....received!!");
102 : }
103 : ↑
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : /*****
2 :     FILE NAME : 1PLOTGPH.C
3 :     SUPPORT   : 2MANYPLT.C
4 :     REQUIRE   : (ErrorMessage DOES NOT exist)
5 : *****/
6 :
7 : #define MAX_WIDTH_Y 360
8 : #define MAX_WIDTH_X 600
9 : #define XX 20
10 : #define YY 460
11 : #define RangeY      MAX_WIDTH_Y-40
12 : #define PixelWidth  20
13 :
14 : #define MAX_VAR_PLOT 8
15 : #define NUM_COLOR    11
16 : #define COLOR_LENGTH 15
17 :
18 : extern int color_mode;
19 :
20 : int color_table[NUM_COLOR];
21 : char color_str[NUM_COLOR][COLOR_LENGTH];
22 : int default_color[NUM_COLOR]={ GREEN,YELLOW,LIGHTCYAN,LIGHTGREEN,WHITE,
23 :                                LIGHTMAGENTA,LIGHTBLUE,CYAN,RED,BROWN,BLUE};
24 : char default_c_str[NUM_COLOR][COLOR_LENGTH]={
25 :     "GREEN","YELLOW","LIGHTCYAN",
26 :     "LIGHTGREEN","WHITE","LIGHTMAGENTA","LIGHTBLUE",
27 :     "CYAN","RED","BROWN","BLUE"};
28 :
29 : int WINDOW_COLOR;
30 : int BACK_GROUND_COLOR;
31 : int HEAD_COLOR;
32 : int DOT_COLOR;
33 : int EDGE_OUT_COLOR;
34 : int EDGE_IN_COLOR;
35 : int NUMBER_COLOR;
36 :
37 :
38 : /***** prototype *****/
39 : void WriteHeader(char *FILE_NAMES,Value_type max,Value_type min,
40 :     Value_type multiply);
41 : void PlotGraph(Value_type *DATA,int size,Value_type y_middle,
42 :     Value_type scale_y,Value_type multiply,int color);
43 : void PlotMaxMinAxis(Value_type max,Value_type min,Value_type sampling,
44 :     Value_type init_time);
45 : void SetColorMode(int color_mode);
46 : void FillBackGround(int x1,int y1,int x2,int y2,int back_g_color,int edge);
47 : Value_type FindScale(Value_type max,Value_type min,int range,int pixel_width,
48 :     Value_type *multiply);
49 : Value_type RoundScale(Value_type scale,Value_type multiply);
50 : int FindMidY(Value_type max,Value_type min,int range,int pixel_width);
51 : int gprintf( int *xloc, int *yloc, char *fmt, ... );
52 :
53 :
54 : /*****/
55 : void SetColorMode(int color_mode)
56 : { int i;
57 :   if(color_mode==0){
58 :     for(i=0; i<NUM_COLOR; i++)
59 :       color_table[i]=default_color[i];
60 :     for(i=1; i<NUM_COLOR; i++)
61 :       strcpy(color_str[i],default_c_str[i]);
62 :     WINDOW_COLOR      =LIGHTGRAY;
63 :     BACK_GROUND_COLOR =DARKGRAY;
64 :     HEAD_COLOR        =LIGHTCYAN;
65 :     EDGE_OUT_COLOR     =LIGHTRED;
66 :     DOT_COLOR          =LIGHTGRAY;
67 :     EDGE_IN_COLOR      =YELLOW;
68 :     NUMBER_COLOR       =LIGHTMAGENTA;
69 :   }
70 :   else{
71 :     for(i=0; i<NUM_COLOR; i++)
72 :       color_table[i]=WHITE;
73 :     for(i=1; i<NUM_COLOR; i++)
74 :       strcpy(color_str[i],"BLACK");
75 :     WINDOW_COLOR      =BLACK;
76 :     BACK_GROUND_COLOR =BLACK;
77 :     HEAD_COLOR        =WHITE;
78 :     DOT_COLOR          =DARKGRAY;
79 :     EDGE_OUT_COLOR     =WHITE;
80 :     EDGE_IN_COLOR      =WHITE;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :     NUMBER_COLOR      =WHITE;
82 : }
83 : }
84 :
85 :
86 :
87 : void WriteColorHeader(plot_vars,num_var)
88 : int num_var;
89 : char *plot_vars;
90 : { int x=200,y=15,var_index;
91 :   FillBackGround(XX+165,9,XX+MAX_WIDTH_X+1,YY-MAX_WIDTH_Y-6,BLACK,
92 :                 EDGE_OUT_COLOR);
93 :   settextstyle(2,0,4);
94 :   for(var_index=0; var_index<num_var; var_index++){
95 :     setcolor(color_table[var_index]);
96 :     fprintf(&x,&y,"(%-10s) : %c", color_str[var_index],
97 :           *(plot_vars+var_index) );
98 :     y+=11;
99 :     if((var_index+1)%4==0){
100 :       x+=150;
101 :       y=15;
102 :     }
103 :   }
104 : }
105 :
106 :
107 : void WriteHeader(char *FILE_NAMES,Value_type max,Value_type min,
108 : Value_type multiply)
109 : { int x=25,y=15;
110 :   char F_NAME[50];
111 :   FillBackGround(XX-1,9,XX+160,YY-MAX_WIDTH_Y-6,BLACK,EDGE_OUT_COLOR);
112 :   settextstyle(2,0,4);
113 :   setcolor(HEAD_COLOR);
114 :   if( strcmp(FILE_NAMES,NULL)!=0 ){
115 :     strcpy(F_NAME,FILE_NAMES);
116 :     if(strlen(FILE_NAMES)>14) F_NAME[14]='\0';
117 :     fprintf(&x,&y,"FILE NAME : %-15s",F_NAME);
118 :     y+=11;
119 :   }
120 :   if( fabs(max) > 0.1)
121 :     fprintf(&x,&y,"DATA MAX : (%-3.2f)",max);
122 :   else
123 :     fprintf(&x,&y,"DATA MAX : (%-3.8f)",max);
124 :   y+=11;
125 :   if( fabs(min) > 0.1)
126 :     fprintf(&x,&y,"DATA MIN : (%-3.2f)",min);
127 :   else
128 :     fprintf(&x,&y,"DATA MIN : (%-3.8f)",min);
129 :   y+=11;
130 :   if( fabs(multiply) > 0.1)
131 :     fprintf(&x,&y,"SCALE : %-3.2f",multiply);
132 :   else
133 :     fprintf(&x,&y,"SCALE : %-3.8f",multiply);
134 : }
135 :
136 :
137 : void PlotGraph(Value_type *DATA,int size,Value_type y_middle,
138 : Value_type scale_y,Value_type multiply,int color)
139 : { int i;
140 :   Value_type *data_pntr;
141 :   data_pntr=DATA;
142 :   setcolor(color);
143 :   moveto(XX,y_middle-(int)((data_pntr+1)*multiply*PixelWidth/scale_y));
144 :   /*PixelWidth=20*/
145 :   for(i=1;i<size;i++){
146 :     data_pntr++;
147 :     lineto(XX+i,y_middle-(*data_pntr)*multiply*PixelWidth/scale_y);
148 :   }
149 : }
150 :
151 : void PlotMaxMinAxis(Value_type max,Value_type min,Value_type sampling,
152 : Value_type init_time)
153 : {int x1,y1,x2,y2,y,l,k;
154 :   Value_type scale_y,y_middle,multiply;
155 :   Value_type time,step_time,f;
156 :
157 :   cleardevice();
158 :   SetColorMode(color_mode);
159 :   FillBackGround(0,0,639,479,WINDOW_COLOR,EDGE_OUT_COLOR);
160 :   FillBackGround(XX-1,YY-MAX_WIDTH_Y-1,XX+MAX_WIDTH_X+1,YY+1,

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161:                                     BACK_GROUND_COLOR, EDGE_IN_COLOR);
162: scale_y=FindScale(max,min,RangeY,PixelWidth,&multiply);
163: y_middle=FindMidY(max,min,RangeY,PixelWidth);
164: line(XX,y_middle,XX+MAX_WIDTH_X,y_middle);
165: setcolor(DOT_COLOR);
166: for(l=0; l<=MAX_WIDTH_Y; l+=PixelWidth)
167:   for(k=PixelWidth; k<=MAX_WIDTH_X; k+=PixelWidth)
168:     line(XX+k,YY-l+1,XX+k,YY-l-1);
169: for(l=0; l<=MAX_WIDTH_X; l+=PixelWidth)
170:   for(k=PixelWidth; k<=MAX_WIDTH_Y; k+=PixelWidth)
171:     line(XX+l+1,YY-k,XX+l-1,YY-k);
172:
173: setcolor(NUMBER_COLOR);
174: y=PixelWidth;
175: f=((y_middle-(YY-y))/PixelWidth)*scale_y;
176: if( fmod(f/scale_y,2)!=0 ){
177:   y=2*PixelWidth;
178:   f=((y_middle-(YY-y))/PixelWidth)*scale_y;
179: }
180: settextstyle(2,0,2);
181: for( ; y<MAX_WIDTH_Y; f+=2*scale_y, y+=2*PixelWidth){
182:   x1=XX-20;
183:   y1=YY-y-5;
184:   gprintf(&x1,&y1,"%4.0f",f);
185: };
186: step_time=sampling*60;
187: if( (step_time<1) || (step_time>9999) )
188:   settextstyle(2,0,4);
189: else
190:   settextstyle(2,0,5);
191: for(k=0,time=0; k<=MAX_WIDTH_X; k+=60, time+=step_time)
192:   { x2=XX+k-12;
193:     y2=YY+6;
194:     if(step_time<1)
195:       gprintf(&x2,&y2,"%-7.2f",time+init_time);
196:     else
197:       gprintf(&x2,&y2,"%-7.1f",time+init_time);
198:   };
199: }
200:
201:
202:
203: void FillBackGround(int x1,int y1,int x2,int y2,int back_g_color,int edge)
204: {
205:   setfillstyle(SOLID_FILL,back_g_color);
206:   bar(x1,y1,x2,y2);
207:   setcolor(edge);
208:   line(x1,y1,x2,y1);
209:   line(x1,y1,x1,y2);
210:   line(x2,y1,x2,y2);
211:   line(x1,y2,x2,y2);
212: }
213:
214: Value_type FindScale(Value_type max,Value_type min,int range,int pixel_width,
215:                     Value_type *multiply)
216: { Value_type delta,scale;
217:   int digit;
218:   delta=max-min;
219:   if(delta==0) return(1);
220:   scale=pixel_width*delta/range;
221:   digit=(int)floor(log10(scale));
222:   *multiply=pow(10,-digit+1);
223:   scale=RoundScale(scale,*multiply);
224:   return(scale);
225: }
226:
227:
228: Value_type RoundScale(Value_type scale,Value_type multiply)
229: { Value_type new_scale,residue_check;
230:   int digit;
231:   new_scale=scale*multiply;
232:   digit=(int)floor(log10(new_scale));
233:   residue_check=fmod(new_scale,pow(10,digit));
234:   if(digit==1){
235:     if( floor(new_scale/pow(10,digit))<=3 ){
236:       residue_check=fmod(new_scale,pow(10,digit-1));
237:       new_scale=new_scale-residue_check+pow(10,digit-1);
238:     }
239:   }
240:   else
     new_scale=new_scale-residue_check+pow(10,digit);

```

```
241:     return(new_scale);
242: }
243: if( residue_check>(5*pow(10,digit-1)) )
244:     new_scale=new_scale-residue_check+pow(10,digit);
245: else
246:     new_scale=new_scale-residue_check;
247: return(new_scale);
248: }
249:
250: int FindMidY(Value_type max,Value_type min,int range,int pixel_width)
251: { Value_type scale,multiply;
252:   int y_middle;
253:   scale=FindScale(max,min,range,pixel_width,&multiply);
254:   y_middle=YY+20*(int)floor(min*multiply/scale);
255:   /*This formular is painstakingly scrutinized,so it is certainly valid*/
256:   return(y_middle);
257: }
258:
259:
260: int gprintf( int *xloc, int *yloc, char *fmt, ... )
261: {
262:   va_list argptr;
263:   char str[140];
264:   int cnt;
265:   va_start( argptr, fmt );
266:   cnt = vsprintf( str, fmt, argptr );
267:   outtextxy( *xloc, *yloc, str );
268:   *yloc += textheight( "H" ) + 2;
269:   va_end( argptr );
270:   return( cnt );
271: }
272: ↑
```

```

1 : /*****
2 : FILE NAME : 2POLISH.C
3 : PROGRAMMER: PATPONG LAOSUWAN
4 : REQUIRE   : I_DEFINE.H
5 :             1STRING.C
6 :             (ErrorMessage exist)
7 : SUPPORT   : 3SET_FN.C
8 :             SIMULATE.C
9 :             MODELROD.C
10 :            PHASEPLN.C
11 : COMMENT   : 1. Check syntetic of expresion
12 :             2. Covert expression to postorder form
13 :             3. Evaluate value of the postorder expression
14 : UPDATE    : March 14, 1993.
15 : *****/
16 :
17 :
18 : /***** PROTOTYPE *****/
19 : void      ReadVar(Value_type *ff,Token_type *lex[26],Var_type *var);
20 : void      ReadVariables(Value_type *ff, Token_type *lex,
21 :                        Var_type *var);
22 : Value_type EvaluatePosfix(Token_type *token_pntr);
23 :
24 : int PolishToken(Token_type *TOKEN,Var_type *var_pntr,char *equa_str);
25 : int TranslateExpression(Token_type *lex, char *equa_str);
26 : int FindNumber(char *str,int pos,Token_type *lex);
27 : int FindWord(char *str,int pos,Token_type *lex);
28 : int ExtractWord(char *str,int pos,char *word);
29 : int Hash(char *word);
30 : int FindSymbol(char *str,int pos,Token_type *lex);
31 : int Syntetic(Token_type *);
32 : int count(Token_type *,char *);
33 : int leading1(Token_type *);
34 : int leading2(Token_type *);
35 : int lagging1(Token_type *);
36 : int lagging2(Token_type *);
37 : int Polish1(Token_type *lex, Token_type *translate);
38 : void FindVariable(Token_type *lex,Var_type *index_pntr);
39 :
40 : Value_type DoUnary(Token_type *,Value_type);
41 : Value_type DoBinary(Token_type *,Value_type,Value_type);
42 :
43 :
44 : #define FIRSTUNARY 3 /* ~ abs sqrt exp ln log10 sin cos sign */
45 : #define LASTUNARY FIRSTUNARY+8 /* 11 */
46 : #define FIRST_U_STR FIRSTUNARY+1 /* abs sqrt exp ln log10 sin cos sign */
47 : #define LAST_U_STR LASTUNARY
48 : #define FIRSTBINARY LASTUNARY+1 /* + - * / % ^ */
49 : #define LASTBINARY LASTUNARY+6 /* 11+6 , 6 binary operators */
50 :
51 : #define FIRSTOPERAND 18 /* 18 = LASTBINARY+1 */
52 : #define LASTOPERAND LASTBINARY+MAXVARIABLE /* 18+26 */
53 :
54 :
55 :
56 :
57 :
58 : /***** VARIABLE TABLE *****/
59 : Token_type Token_table[LASTOPERAND+1]={ {0,"end",ENDEXPR,0},
60 : {1,"(",LEFTPAREN,0}, {2,")",RIGHTPAREN,0},
61 : {3,"~",UNARYOP,6}, {4,"abs",UNARYOP,6},
62 : {5,"sqrt",UNARYOP,6}, {6,"exp",UNARYOP,6},
63 : {7,"ln",UNARYOP,6}, {8,"log10",UNARYOP,6},
64 : {9,"sin",UNARYOP,6}, {10,"cos",UNARYOP,6},
65 : {11,"sign",UNARYOP,6}, {12,"+",BINARYOP,4},
66 : {13,"-",BINARYOP,4}, {14,"*",BINARYOP,5},
67 : {15,"/",BINARYOP,5}, {16,"%",BINARYOP,5},
68 : {17,"^",BINARYOP,6},
69 : {18,"a",OPERAND,0}, {19,"b",OPERAND,0},
70 : {20,"c",OPERAND,0}, {21,"d",OPERAND,0},
71 : {22,"e",OPERAND,0}, {23,"f",OPERAND,0},
72 : {24,"g",OPERAND,0}, {25,"h",OPERAND,0},
73 : {26,"i",OPERAND,0}, {27,"j",OPERAND,0},
74 : {28,"k",OPERAND,0}, {29,"l",OPERAND,0},
75 : {30,"m",OPERAND,0}, {31,"n",OPERAND,0},
76 : {32,"o",OPERAND,0}, {33,"p",OPERAND,0},
77 : {34,"q",OPERAND,0}, {35,"r",OPERAND,0},
78 : {36,"s",OPERAND,0}, {37,"t",OPERAND,0},
79 : {38,"u",OPERAND,0}, {39,"v",OPERAND,0},
80 : {40,"w",OPERAND,0}, {41,"x",OPERAND,0},

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :             {42,"y",OPERAND,0},    {43,"z",OPERAND,0}
82 :             };
83 :
84 : /***** START PROGRAM *****/
85 :
86 : /** send (var_val,pol_tok['a'-ASC_a],var_tok['a'-ASC_a]) **/
87 : void ReadVariables(ff,lex,var)
88 : Value_type *ff;
89 : Token_type *lex;
90 : Var_type *var;
91 : /* lex points to TOKEN or the beginnig of polishing string */
92 : { int i,v;
93 :   Token_type *lex0;
94 :   lex0=lex;
95 :   for( ; (v=(*var).name_index)!=-1 ; var++){
96 :     i=(*var).token_index;
97 :     lex+=i;
98 :     (*lex).info.val=(ff+v-FIRSTOPERAND); /*FIRSTOPERAND=18*/
99 :     lex=lex0;
100 :   }
101 : }
102 :
103 : Value_type EvaluatePosfix(Token_type *token_pntr)
104 : /* stack points array of Stack_type */
105 : {
106 :   Kind_type type;
107 :   Value_type a,b;
108 :   Value_type stacks[MAXSTACK],*stack;
109 :
110 :   stack=stacks;
111 :   do{
112 :     switch(type=(*token_pntr).kind ){
113 :       case OPERAND: *stack=(*token_pntr).info.val;
114 :                     stack++;
115 :                     break;
116 :       case UNARYOP: a=(--stack);
117 :                     *stack=DoUnary(token_pntr,a);
118 :                     stack++;
119 :                     break;
120 :       case BINARYOP: b=(--stack);
121 :                      a=(--stack);
122 :                      *stack=DoBinary(token_pntr,a,b);
123 :                      stack++;
124 :                      break;
125 :       case ENDEXPR: a=(--stack);
126 :                     if(stack != &stacks[0])
127 :                       ErrorMessage(0,"[EvaluatePosfix]: Invalid expression");
128 :                     break;
129 :     }; /* end of switch */
130 :
131 :     token_pntr++;
132 :   }while(type!=ENDEXPR);
133 :
134 :   return(a);
135 : } /* end of function */
136 :
137 :
138 :
139 : int PolishToken(Token_type *TOKEN,Var_type *var_pntr,char *equa_str)
140 : /* PolishToken() check the syntetic of expression, polish expression to
141 : lexicons and store lexicons in TOKEN */
142 : /* return 0=no error; otherwise error */
143 : /* return negative value to indicate the error case */
144 : /* -1: ErrorMessage("[FindWord]: Variable/Operator not defined."); */
145 : /* -2: ErrorMessage("[ExtractWord]: Word is truncated."); */
146 : /* -3: ErrorMessage("[FindSymbol]: Syntax/Symbol error."); */
147 : /* -4: ErrorMessage("[Syntetic]: expression is incomplete."); */
148 :
149 : { Token_type *T_pntr;
150 :   int error_index;
151 :
152 :   T_pntr=TOKEN; /* T_pntr points to destination, which is TOKEN. */
153 :   if( (error_index=TranslateExpression(TOKEN,equa_str)) !=0 ){
154 :     /*TranslateExpression() return nonzero to indicate index of error case*/
155 :     return(error_index);
156 :   }
157 :
158 :   Polish1(T_pntr,TOKEN);
159 :   /* Polish TOKEN in post order. */
160 :   /* value return from Polish1() has no special meaning. */

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161: FindVariable(TOKEN,var_pntr);
162: return(0);
163: }
164:
165:
166: /* ***** TranslateExpression ***** */
167:
168: int TranslateExpression(Token_type *lexicon, char *equ_str)
169: /* Translate the string "equ_str" and store in lexicon, not polish yet */
170: /* return 0=no error ; otherwise =error */
171: /* return negative mean index of error case */
172: { int len,pos,i;
173:   char *instring,c;
174:
175:   Token_type *lex;
176:   rid_a_space(equ_str);
177:   instring=equ_str;
178:   if(instring[0]=='~') instring[0]='';
179:   len=strlen(instring);
180:   for(i=1; i<len; i++)
181:     if( (instring[i-1]=='(') && (instring[i]=='-') ) instring[i]='~';
182:   lex=lexicon;
183:
184:   for(pos=0; pos<len; ){
185:     c=instring[pos];
186:     if( ( c >='0' && c <='9' ) || c =='.') {
187:       pos=FindNumber(instring,pos,lex); /* return negative mean error */
188:       lex++;
189:     }
190:     else
191:       if( (c=tolower(instring[pos])) >='a' && c<='z' ){
192:         pos=FindWord(instring,pos,lex); /* return negative mean error */
193:         lex++;
194:       }
195:       else{
196:         pos=FindSymbol(instring,pos,lex); /* return negative mean error */
197:         lex++;
198:       }
199:       if( pos<0 ) /* pos<0 mean error */
200:         return(pos); /* return negative to show index of error case */
201:     };
202:     sprintf( (*lex).name , "end");
203:     (*lex).kind=ENDEXPR;
204:     if(Syntetic(lexicon)) /* Syntetic()==1 mean correct syntetic */
205:       return (0); /* 0 mean no error */
206:     else
207:       return 1; /* 1 mean error in TranslateExpresion() */
208:   }
209:
210:
211: /* FindNumber : find number is in str[] staring at position. */
212: int FindNumber(char *str,int pos,Token_type *lex)
213: { int c;
214:   (*lex).kind=OPERAND;
215:   (*lex).info.val=atof(&str[pos]);
216:   strcpy((*lex).name,"number");
217:   for(c=str[pos] ; (c>='0' && c<='9') || c=='.' ; )
218:     c=str[++pos];
219:   return pos;
220: }
221:
222:
223:
224:
225: /* FindWord : find word in str[] starting at pos */
226: int FindWord(char *str,int pos,Token_type *lex)
227: { int h;
228:   char word[MAXNAME];
229:
230:   if( (pos=ExtractWord(str,pos,word)) <0 )
231:     return(pos);
232:   h=Hash(word);
233:   if(h!=-1){ /* token is already defined */
234:     strcpy( (*lex).name,Token_table[h].name);
235:     (*lex).index=Token_table[h].index;
236:     (*lex).kind=Token_table[h].kind;
237:     (*lex).info.pri=Token_table[h].info.pri;
238:     return pos;
239:   }
240:   else

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

241:         return(-1); /* index of error case */
242: }
243:
244: /* ExtractWord : extract word from str[] start at pos */
245:
246: int ExtractWord(char *str,int pos,char *word)
247: { int i;
248:   char c;
249:
250:   for(i=pos,c=tolower(str[i]); (c!='a' && c!='z') || (c!='0' && c!='9');
251:     c=tolower(str[++i]) )
252:     *word++=c;
253:   *word='\0';
254:   if(i-pos > MAXNAME){
255:     return(-2); /* index of error case */
256:                 /* not truncate but inform error */
257:   };
258:   return(i);
259: }
260:
261: int Hash(char *word)
262: /* return -1 mean word not found */
263: { int i;
264:   for(i=FIRST_U_STR ; i<=LAST_U_STR ; i++)
265:     if( strcmp(word,Token_table[i].name) == 0 ) return(i);
266:   for(i=FIRSTOPERAND ; i<=LASTOPERAND ; i++)
267:     if( strcmp(word,Token_table[i].name) == 0 ) return(i);
268:   return (-1);
269: }
270:
271: int FindSymbol(char *str,int pos,Token_type *lex)
272: { int i,j,find=0;
273:   char c;
274:
275:   c=*(str+pos);
276:   for(i=1;i<=FIRST_U_STR && !find; i++)
277:     if( c==(Token_table[i].name[0]) { find=1; j=i;}
278:   for(i=FIRSTBINARAY ; i<=LASTBINARAY && !find ;i++)
279:     if( c==(Token_table[i].name[0]) { find=1; j=i;}
280:   if(find) {
281:     strcpy( (*lex).name,Token_table[j].name);
282:     (*lex).index=Token_table[j].index;
283:     (*lex).kind=Token_table[j].kind;
284:     (*lex).info.pri=Token_table[j].info.pri;
285:     return pos+1;
286:   }
287:   else
288:     return(-3);
289: }
290:
291:
292: int Syntetic(Token_type *lex)
293: /* return 1 mean correct syntetic
294:    return 0 mean incorrect syntetic */
295: { int i,j;
296:
297:   i=count(lex,"(");
298:   j=count(lex,")");
299:   if(i!=j) return 0; /* 0 mean incorrect */
300:                                     /* this test may not be neccessary */
301:   if( !lagging1(lex) )
302:     return (0);
303:   for( lex++ ; (*lex).kind!=ENDEXPR; lex++){
304:     if(leading1(lex)){ /****** already revised */
305:       if(lagging1(lex+1))
306:         /* Do nothing ,correct syntax */ ;
307:       else
308:         return (0);
309:     }
310:     else{
311:       if( leading2(lex)){
312:         if(lagging2(lex+1))
313:           /* do nothing */ ;
314:         else
315:           return (0);
316:       }
317:       else
318:         return (0);
319:     }
320:   } /* end of for */

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้คัดลอกเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

321: return (1); /* return 1 imply syntetic correct */
322: }
323:
324: int count(Token_type *lex,char *str)
325: { int i=0;
326:   for( ; (*lex).kind !=ENDEXPR ; lex++)
327:     if( strcmp( (*lex).name,str ) ==0) i++;
328:   return i;
329: }
330:
331: int leading1(token_pntr)
332: Token_type *token_pntr;
333: { int k;
334:   return((k=(*token_pntr).kind) == LEFTPAREN || k==UNARYOP || k==BINARYOP );
335: }
336:
337: int leading2(token_pntr)
338: Token_type *token_pntr;
339: { int k;
340:   return( (k=(*token_pntr).kind) == OPERAND || k==RIGHTPAREN );
341: }
342:
343: int lagging1(token_pntr)
344: Token_type *token_pntr;
345: { int k;
346:   return((k=(*token_pntr).kind) == LEFTPAREN || k==UNARYOP || k==OPERAND );
347: }
348:
349: int lagging2(token_pntr)
350: Token_type *token_pntr;
351: { int k;
352:   return((k=(*token_pntr).kind) ==RIGHTPAREN || k==BINARYOP || k==ENDEXPR);
353: }
354:
355:
356: #define TRUE 1
357: #define FALSE 0
358: int Polish1(Token_type *lex, Token_type *translate)
359: /* You will call PolishToken() before.*/
360: /* PolishToken() check the syntetic of expression */
361: /* always return(0) because the syntetic of *translate will be scrutinized
362: in TranslateExpression() */
363: { Token_type stacks[MAXTOKEN],*stack;
364:   Token_type *trans,*token_pntr;
365:   Kind_type type;
366:   int stack_index=0,endright;
367:
368:   trans=translate;
369:   stack=stacks;
370:   do{
371:     token_pntr=lex++;
372:     switch(type = (*token_pntr).kind ) {
373:       case OPERAND :
374:         *(trans++)=*token_pntr;
375:         /* push token into translation stack */
376:         break;
377:       case LEFTPAREN :
378:         *(stack++)=*token_pntr; /* push into stack */
379:         stack_index++;
380:         break;
381:       case RIGHTPAREN :
382:         for(*token_pntr=*(--stack),stack_index--;
383:            (*token_pntr).kind!=LEFTPAREN;
384:            (*token_pntr)=*(--stack),stack_index--)
385:           *(trans++)=*token_pntr;
386:         break;
387:       case UNARYOP:
388:       case BINARYOP :
389:         do{
390:           if(stack_index<=0) /* empty stack */
391:             endright=TRUE;
392:           else if( (*stack-1).kind == LEFTPAREN )
393:             endright=TRUE;
394:           else if( (*stack-1).info.pri < (*token_pntr).info.pri)
395:             endright=TRUE;
396:           else if( (*stack).info.pri == (*token_pntr).info.pri
397:                  && (*token_pntr).info.pri == MAXPRIORITY )
398:             endright=TRUE;
399:           else{
400:             endright=FALSE;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

401:         *(trans++)=*(--stack);
402:         stack_index--;
403:     }
404:     }while(!endright);
405:     *(stack++)=*token_pntr;
406:     stack_index++;
407:     break;
408: case ENDEXPR :
409:     while(stack_index > 0){
410:         *token_pntr=*(--stack);
411:         stack_index--;
412:         *(trans++)=*token_pntr;
413:     };
414:     break;
415: } /* end of case */
416:
417: }while(type!=ENDEXPR);
418: (*trans).kind = ENDEXPR;
419:
420: sprintf((*trans).name,"end");
421: return (0);
422: } /* end of translate */
423:
424:
425: /** send(pol_tok['a'-ASC_a], var_tok['a'-ASC_a] ) **/
426: void FindVariable(Token_type *lex,Var_type *index_pntr)
427: { int i,j;
428:   char c;
429:   for (i=0 ; (*lex).kind!=ENDEXPR ; lex++,i++){
430:       if( strlen((*lex).name) ==1){
431:           c=tolower((*lex).name[0]);
432:           if( c>='a' && c<='z'){
433:               (*index_pntr).token_index=i;
434:               for(j=FIRSTOPERAND; j<=LASTOPERAND; j++){
435:                   if( c==(*Token_table[j]).name[0] )
436:                       (*index_pntr).name_index=j;
437:               }
438:               index_pntr++;
439:           } /* end of if */
440:       } (*index_pntr).name_index=-1;
441:   }
442: }
443:
444: /* DoUnary: apply unary operator on x ; return the result. */
445: Value_type DoUnary(Token_type *lex,Value_type x)
446: { /* lex points to TOKEN of UNARY OPERATOR */
447:   switch( (*lex).index ) {
448:       case 3: return -x;
449:       case 4: return abs(x);
450:       case 5: return sqrt(x);
451:       case 6: return exp(x);
452:       case 7: return log10(x)/log10(EXPO);
453:       case 8: return log10(x);
454:       case 9: return sin(x);
455:       case 10: return cos(x);
456:       case 11: if (x>0) return 1; else return 0;
457:       default: ErrorMessage(0,"[DoUnary]: Invalid Unary token."); break;
458:   } /* end of switch */
459: }
460:
461: /* doBinary: apply binary operator on x and y; return the result. */
462: Value_type DoBinary(Token_type *lex,Value_type x,Value_type y)
463: { /* lex points to TOKEN of BINARY OPERATOR */
464:   switch((*lex).index) {
465:       case 12: return x+y; /* binary plus */
466:       case 13: return x-y; /* binary minus */
467:       case 14: return x*y; /* times */
468:       case 15: if (y != (Value_type) 0) /* divide */
469:           return x/y;
470:       else
471:           ErrorMessage(0,"[DoBinary]: Zero divisor");
472:       case 16: if (y != (Value_type) 0) /* modulus operator */
473:           return fmod(x,y);
474:       else
475:           ErrorMessage(0,"[DoBinary]: Zero divisor");
476:       case 17: return pow(x,y); /* x to power y */
477:       default: ErrorMessage(0,"[DoBinary]: Invalid Binary token.");
478:           break;
479:   }
480: }

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : /*****
2 : FILE NAME : 2RUNGE.C
3 : PROGRAMMER: PATPONG LAOSUWAN
4 : REQUIRE   : I_DEFINE.H
5 :             1STRING.C
6 :             2POLISH.C
7 :             (ErrorMessage exist)
8 : SUPPORT    : 3SET_FN.C
9 :             SIMULATE.C
10 :            MODELROD.C
11 :            PHASEPLN.C
12 : FUNCTION   : 1. Check syntetic of expresion
13 :              2. Covert expression to postorder form
14 :              3. Evaluate value of the postorder expression
15 : UPDATE     : March 14, 1993.
16 : *****/
17 :
18 :
19 : /*****/
20 : /***** external variable from Simulate.c & Mainp.c *****/
21 : /*****/
22 : extern int      width,iteration_scale,num_equation;
23 : extern char     FILE_NAMES[50];
24 : extern Value_type time_range,iteration_time;
25 : extern Value_type var_val[26]; /* [26] mean a,b,...,z = 0,1,...,25 */
26 : extern char     var_str[MAX_EQUATION];
27 :
28 : /* tell:which equation belongs to which variable*/
29 : /* store variable name of each equation */
30 : extern char     equ_typ[27];
31 :
32 : /* E,S= Expression, State */
33 : /* which variable is what type of an Equation */
34 : extern char     rec_var_str[27];
35 :
36 : /*****/
37 : extern Value_type far *rec_var[26];
38 : /* point to allocated array of record-variable */
39 : extern Var_type   far *var_tok[26];
40 : /* store variable position in token */
41 : /* array index is equation_index */
42 : /* store TOKEN of polished string */
43 : /*****/
44 : int ExecuteEquation(int num_equ);
45 : int GetOrder(int equ_index);
46 : void FindNextState(Value_type *var_val,int order,int equ_index);
47 : Value_type EvaluateExpression(Value_type *variable,int equ_index);
48 : void set_gobal(void);
49 : void RecordVarVal(Value_type far **rec_var, Value_type *var_val,
50 :                  int time_index);
51 : void WriteRecVarToFile(char *FILE_NAMES,int width,char *rec_var_str);
52 :
53 :
54 : /*****/
55 : int ExecuteEquation(int num_equ)
56 : { int order,time_index,equ_index,var_t_index,index,iter_scale;
57 :   int iterate,i,text[300];
58 :   Value_type var_val_prev[26],*t_var_ptr;
59 :
60 :   if(num_equ==0){
61 :     ErrorMessage(0,"[ExecuteEquation]: Equation not exist");
62 :     return(1);
63 :   }
64 :   for(i=0;i<26;i++) var_val_prev[i]=var_val[i];
65 :   for(i=1,iter_scale=1;i<=num_equ;i++)
66 :     if( equ_typ[i]=='S' ) iter_scale=iteration_scale;
67 :   /* if there is no diff. equ. then iteration_scale should be 1.*/
68 :   iteration_time=time_range/(width*iter_scale);
69 :   var_t_index='t'-'a';
70 :   t_var_ptr=var_val+var_t_index;
71 :
72 :   Fastgettext(27,7,49,12,text);
73 :   makewin(27,7,49,12,0x2E,0x20,1);
74 :   Fastputs(31,8,0x21,"PROCESS RUNNING");
75 :   Fastputs(31,9,0x20,"LOOP COUNT DOWN");
76 :
77 :   for(time_index=1; time_index<=width; time_index++){
78 :     for(iterate=1; iterate<=iter_scale; iterate++){
79 :       equ_index=1;
80 :       do{ index=var_str[equ_index]-'a';

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :         switch(equ_typ[index]){
82 :             case 'E': var_val[index]=EvaluateExpression(var_val,equ_index);
83 :                 equ_index++;
84 :                 break;
85 :             case 'S': order=GetOrder(equ_index);
86 :                 FindNextState(var_val,order,equ_index);
87 :                 equ_index+=order;
88 :                 break;
89 :             default : ErrorMessage(0,
90 :                 "[ExecuteEquation]: equ_typ error!!\n");
91 :                 Fastputtext(27,7,49,12,text);
92 :                 return(1);
93 :         }
94 :         }while(equ_index<=num_equ);
95 :         (*t_var_ptr)+=iteration_time;
96 :     }
97 :     if( rec_var_str[0]!='\0')
98 :         RecordVarVal(rec_var,var_val,time_index);
99 :     Fastprintf(38,11,0x21,"%-5d",width-time_index);
100 : }
101 : Fastputtext(27,7,49,12,text);
102 : for(i=0; i<26; i++) var_val[i]=var_val_prev[i];
103 : return(0);
104 : }
105 :
106 : void WriteRecVarToFile(char *FILE_NAMES,int width,char *rec_var_str)
107 : { char *rec_pntr,var_ch;
108 :   int rec_index;
109 :   int check_equ_exist[26],ch_index,equ_index;
110 :   for(ch_index=0; ch_index<26; ch_index++)
111 :       check_equ_exist[ch_index]=0;
112 :   equ_index=1;
113 :   var_ch=var_str[equ_index];
114 :   while(var_ch!='\0'){
115 :       check_equ_exist[var_ch-'a']=1;
116 :       equ_index++;
117 :       var_ch=var_str[equ_index];
118 :   }
119 :   WriteHeaderFile(FILE_NAMES);
120 :   rec_pntr=rec_var_str;
121 :   while( (*rec_pntr)!='\0' ){
122 :       rec_index=(*rec_pntr)-'a';
123 :       if( check_equ_exist[rec_index]==1 )
124 :           WriteArrayToFile(FILE_NAMES,rec_var[rec_index],width,*rec_pntr);
125 :       rec_pntr++;
126 :   }
127 : }
128 :
129 :
130 : /** send (rec_var, var_val, rec_var_str, time_index) **/
131 : void RecordVarVal(rec_var,var_val,time_index)
132 : Value_type far **rec_var,*var_val;
133 : int time_index;
134 : { int var_index;
135 :   char *rec_pntr;
136 :   Value_type *header_pntr;
137 :   if( time_index<width ){
138 :       rec_pntr=rec_var_str;
139 :       while( (*rec_pntr)!='\0' ){
140 :           var_index=(*rec_pntr)-'a';
141 :           header_pntr=*(rec_var+var_index);
142 :           header_pntr+=time_index;
143 :           *header_pntr=*( var_val+var_index );
144 :           rec_pntr++;
145 :       }
146 :   }
147 : }
148 :
149 : int GetOrder(int equ_index)
150 : { int index,order;
151 :   order=0;
152 :   index=var_str[equ_index]-'a';
153 :   while(equ_typ[index]=='S'){
154 :       order++;
155 :       equ_index++;
156 :       index=var_str[equ_index]-'a';
157 :   }
158 :   if(order>MAXORDER)
159 :       ErrorMessage(0,"[GetOrder]: order overflow !!");
160 :   else return(order);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161: }
162:
163: void FindNextState(Value_type *var_val,int order,int equ_index)
164: { Value_type variable[26],Kn[5][MAXORDER],temp_state[MAXORDER],factor;
165:   register int KUTTA,i,j,index;
166:
167:   for(i=0;i<26;i++) variable[i]=var_val[i]; /***** copy *****/
168:   for(i=0;i<order;i++){
169:     Kn[0][i]=0;
170:     temp_state[i]=var_val[var_str[equ_index+i]-'a'];
171:     /* store state variable in temp_state */
172:   }
173:   for(KUTTA=1;KUTTA<=4;KUTTA++){
174:     if( (KUTTA==4) ||(KUTTA==4) ) factor=1;
175:     else factor=0.5;
176:     for(i=0;i<order;i++){
177:       for(j=0;j<order;j++){ /**** Evaluate middle state ****/
178:         index=var_str[equ_index+i]-'a';
179:         variable[index]=var_val[index]+factor*Kn[KUTTA-1][j];
180:       }
181:       Kn[KUTTA][i]=iteration_time*EvaluateExpression(variable,equ_index+i);
182:     }
183:   }
184:   for(i=0;i<order;i++){
185:     index=var_str[equ_index+i]-'a';
186:     var_val[index]=temp_state[i];
187:     /* return temp_state back to state variable */
188:     var_val[index]+=(Kn[2][i]+Kn[3][i])/3;
189:     var_val[index]+=(Kn[1][i]+Kn[4][i])/6;
190:   }
191: }
192:
193: Value_type EvaluateExpression(Value_type *variable,int equ_index)
194: { int index;
195:   index=var_str[equ_index]-'a';
196:   ReadVariables(variable,pol_tok[index],var_tok[index]);
197:   return( EvaluatePosfix(pol_tok[index]) );
198: }
199:
200:
201: ↑

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : /*****
2 : FILE NAME : 2MANYPLT.C
3 : LEVEL : 2
4 : REQUIRE : i_define.h
5 :          lplotgph.c
6 :          lstring.c
7 :          lfile.c
8 :          (ErrorMessage exist)
9 : SUPPORT : SIMULATE.C
10 : /*****/
11 :
12 : /***** typedef float Value_type; *****/
13 : void AskFileName(char *FILE_NAMES);
14 : int WriteFileTest(char *FILE_NAMES, Value_type *DATA);
15 : void WriteDataToArray(Value_type *DATA, int num_data);
16 : void WriteHeader(char *FILE_NAMES, Value_type max, Value_type min,
17 : Value_type multiply);
18 : void WriteColorHeader(char *plot_vars, int num_var);
19 : int PlotVarMem(Value_type far **rec_var, char *rec_var_str, char *variable_str,
20 : int width, Value_type time_range, Value_type init_time);
21 : int ManyVarPlot(char *FILE_NAMES);
22 : int DataAlloc(Value_type **DATA, int num_data, int num_var);
23 : int FindMaxInt(int *data_start, int *data_stop);
24 : void FindManyMaxMin(Value_type **DATA, int index_array, int *size,
25 : int *data_size, Value_type *max, Value_type *min, int num_var);
26 : int GetDataFromFile(char *FILE_NAMES, char rec_var_name,
27 : Value_type *DATA);
28 : int FindMaxMin(Value_type *data_start, Value_type *data_stop,
29 : Value_type *max, Value_type *min);
30 :
31 : FILE *fp;
32 :
33 :
34 : int ManyVarPlot(char *FILE_NAMES)
35 : { Value_type init_time, time_period, max1, min1, bin1;
36 : Value_type y_middle1, scale_y1, multiply1;
37 : Value_type max[MAX_VAR_PLOT], min[MAX_VAR_PLOT],
38 : *DATA[MAX_VAR_PLOT], *DATA_pntr;
39 : Value_type time_range;
40 : char plot_vars[27];
41 : int size[MAX_VAR_PLOT], data_size[MAX_VAR_PLOT];
42 : int index_array, size_max, var_index, num_var, num_alloc, num_data;
43 : int i=DETECT, j;
44 :
45 : init_time=0;
46 :
47 : time_range=GetTimeFile(FILE_NAMES); /* set local variable "time_range" */
48 : num_var=GetRecFile(FILE_NAMES, plot_vars);
49 : num_data=GetSizeFile(FILE_NAMES);
50 : if(num_var>MAX_VAR_PLOT){
51 : num_var=MAX_VAR_PLOT;
52 : *(plot_vars+MAX_VAR_PLOT)='\0';
53 : }
54 : num_alloc=num_var=DataAlloc(DATA, num_data, num_var);
55 : if(num_var==0) return(1);
56 : *(plot_vars+num_var)='\0';
57 :
58 : var_index=0;
59 : while( var_index<num_var ){
60 : data_size[var_index]=GetDataFromFile(FILE_NAMES, *(plot_vars+var_index),
61 : DATA[var_index]);
62 : if( data_size[var_index]==0 ){
63 : del_ch_from_str(plot_vars, *(plot_vars+var_index));
64 : num_var--;
65 : var_index--;
66 : }
67 : var_index++;
68 : }
69 : size_max=FindMaxInt(data_size, data_size+num_var);
70 : if(size_max==0) return(1);
71 : time_period=MAX_WIDTH_X*time_range/num_data;
72 :
73 : initgraph(&i, &j, "");
74 : for(index_array=0; index_array<size_max; index_array+=MAX_WIDTH_X,
75 : init_time+=time_period){
76 : FindManyMaxMin(DATA, index_array, size, data_size, max, min, num_var);
77 : FindMaxMin(max, max+num_var, &max1, &bin1);
78 : FindMaxMin(min, min+num_var, &min1, &min1);
79 : scale_y1=FindScale(max1, min1, RangeY, PixelWidth, &multiply1);
80 : y_middle1=FindMidY(max1, min1, RangeY, PixelWidth);

```

```

81 :         PlotMaxMinAxis(max1,min1,time_range/num_data,init_time);
82 :
83 :         WriteHeader(FILE_NAMES,max1,min1,multiply1);
84 :         WriteColorHeader(plot_vars,num_var);
85 :         for(var_index=0; var_index<num_var; var_index++){
86 :             DATA_pntr=DATA[var_index];
87 :             DATA_pntr+=index_array;
88 :             PlotGraph( DATA_pntr, size[var_index], y_middle1, scale_y1,
89 :                 multiply1, color_table[var_index]);
90 :         }
91 :         getch();
92 :     };
93 :     for(i=0; i<num_alloc; i++)
94 :         free(DATA[i]);
95 :     closegraph();
96 :     return(0);
97 : }
98 :
99 :
100: int PlotVarMem(Value_type far **rec_var,char *rec_var_str,char *variable_str,
101:     int width, Value_type time_range,Value_type init_time)
102: /* rec_var = pointer of array a[],b[],...,z[]
103:    rec_var_str = variable needed to plot
104:    variable_str = variable that have been processed
105:    ( in general case, set variable_str=rec_var_str ) */
106: { Value_type max1,min1,bin1;
107:   Value_type y_middle1,scale_y1,multiply1;
108:   Value_type max[26],min[26],*DATA[26],*DATA_pntr;
109:   int size[26],data_size[26],num_var,index,var_index,width1;
110:   char record_var_str[27],*var_pntr;
111:   int i=DETECT,j;
112:
113:   if(width<=1)
114:       return(1);
115:   else{ if(width>MAX_WIDTH_X)
116:       width1=MAX_WIDTH_X;
117:       else width1=width;
118:   }
119:   index=0;
120:   strcpy(record_var_str,strlwr(rec_var_str));
121:   var_pntr=record_var_str;
122:   while( (*var_pntr)!='\0' ){
123:       if( strcmp(strchr(variable_str,*var_pntr),NULL)!=0 ){
124:           DATA[index]=rec_var[(*var_pntr)-'a'];
125:           data_size[index++]=width1;
126:       }
127:       else{
128:           del_ch_from_str(record_var_str,*var_pntr);
129:           var_pntr--;
130:       }
131:       var_pntr++;
132:   }
133:
134:   num_var=strlen(record_var_str);
135:   if(num_var==0) return(1);
136:
137:   initgraph(&i,&j,"");
138:   FindManyMaxMin(DATA,1,size,data_size,max,min,num_var);
139:   FindMaxMin(max,max+num_var,&max1,&bin1);
140:   FindMaxMin(min,min+num_var,&min1,&bin1);
141:   scale_y1=FindScale(max1,min1,RangeY,PixelWidth,&multiply1);
142:   y_middle1=FindMidY(max1,min1,RangeY,PixelWidth);
143:   PlotMaxMinAxis(max1,min1,time_range/width,init_time);
144:   /* Last parameter is "sampling time". */
145:   WriteHeader("",max1,min1,multiply1);
146:   WriteColorHeader(rec_var_str,num_var);
147:   for(var_index=0; var_index<num_var; var_index++){
148:       DATA_pntr=DATA[var_index];
149:       PlotGraph( DATA_pntr, size[var_index], y_middle1, scale_y1,
150:           multiply1, color_table[var_index]);
151:   }
152:   getch();
153:   closegraph();
154:   return(0);
155: }
156:
157: int DataAlloc(Value_type **DATA,int num_data,int num_var)
158: { int i=0;
159:   if(num_var==0) return(0);
160:   DATA[i++]= (Value_type *) malloc( num_data*sizeof(Value_type) );

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
161: while( (i<num_var) && (DATA[i-1]!=NULL) )
162:     DATA[i++] = (Value_type *) malloc( num_data*sizeof(Value_type) );
163: if(DATA[i-1]!=NULL)
164:     return(i);
165: else
166:     return(i-1);
167: }
168:
169: int FindMaxInt(int *data_start,int *data_stop)
170: { int *pntr,max;
171:   max=*data_start;
172:   for(pntr=data_start; pntr!=data_stop; pntr++)
173:       if( (*pntr)>max )
174:           max=*pntr;
175:   return(max);
176: }
177:
178:
179: void FindManyMaxMin(DATA,index_array,size,data_size,max,min,num_var)
180: Value_type **DATA,*max,*min;
181: int index_array,*size,*data_size,num_var;
182: {Value_type *head0_ptr,*head1_ptr;
183:  int var_index;
184:  for(var_index=0; var_index<num_var; var_index++)
185:      if(index_array>data_size[var_index])
186:          size[var_index]=0;
187:      else{
188:          head0_ptr=head1_ptr=(Value_type *) DATA[var_index];
189:          head0_ptr+=index_array;
190:          if(index_array>(data_size[var_index]-MAX_WIDTH_X)){
191:              head1_ptr+=data_size[var_index];
192:              FindMaxMin(head0_ptr,head1_ptr,max+var_index, min+var_index);
193:              size[var_index]=data_size[var_index]-index_array;
194:          }
195:          else{
196:              head1_ptr+=index_array+MAX_WIDTH_X;
197:              FindMaxMin(head0_ptr,head1_ptr,max+var_index, min+var_index);
198:              size[var_index]=MAX_WIDTH_X;
199:          }
200:      }
201: }
202:
203: int FindMaxMin(Value_type *data_start,Value_type *data_stop,
204:               Value_type *max,Value_type *min)
205: { Value_type *pntr;
206:   if(data_start>data_stop){
207:       ErrorMessage(0,"[FindMaxMin]: Pointer error!");
208:       return(1);
209:   }
210:   pntr=data_start;
211:   *min=*max=*pntr;
212:   for(; pntr!=data_stop; pntr++){
213:       if( (*pntr)>(*max) )
214:           *max=*pntr;
215:       else if( *pntr<*min )
216:           *min=*pntr;
217:   }
218: }
219:
220: †
```

```

1 : /*****
2 : FILE NAME: 2frame.c (level 2 function)
3 : REQUIRE : i_define.h
4 :          1plotgph.c
5 : SUPPORT : phasepln.c (only)
6 : *****/
7 :
8 : #define X_MARGIN      30
9 : #define Y_MARGIN      50
10 :
11 : void draw_frame(int xi,int yi,int xf,int yf,int mode);
12 : void draw_plane(int xi,int yi,int xf,int yf);
13 :
14 :
15 : int THICK_EDGE,B_OUT_GROUND,B_IN_GROUND,AXIS_COLOR,MESH_COLOR;
16 : /*****/
17 : void draw_frame(int xi,int yi,int xf,int yf,int color_mode)
18 : /*** xf-xi-10-X_MARGIN, yf-yi-10-Y_MARGIN must be divisible by 60 ***/
19 : /*** Or the mesh is not balanced. ***/
20 : {
21 :     if(color_mode==0){
22 :         THICK_EDGE= YELLOW;
23 :         B_IN_GROUND=DARKGRAY;
24 :         B_OUT_GROUND=BLACK;
25 :         AXIS_COLOR=WHITE;
26 :         MESH_COLOR= LIGHTGREEN;
27 :     }
28 :     else{
29 :         THICK_EDGE= WHITE;
30 :         B_IN_GROUND=BLACK;
31 :         B_OUT_GROUND=BLACK;
32 :         AXIS_COLOR=WHITE;
33 :         MESH_COLOR= GREEN;
34 :     }
35 :     setfillstyle(1,THICK_EDGE);
36 :     bar(xi,yi,xf,yf);
37 :     setfillstyle(1,B_OUT_GROUND);
38 :     bar(xi+5,yi+5,xf-5,yf-5);
39 :     draw_plane(xi+5+X_MARGIN*0.5-1,yi+5+Y_MARGIN*0.5-1,
40 :                xf-5-X_MARGIN*0.5+1,yf-5-Y_MARGIN*0.5+1);
41 : }
42 :
43 :
44 : void draw_plane(int xi,int yi,int xf,int yf)
45 : {
46 :     int xxx,yyy,x,y,dex,dely;
47 :
48 :     setfillstyle(1,B_IN_GROUND);
49 :     bar(xi,yi,xf,yf);
50 :     xxx=(xi+xf)*0.5;
51 :     yyy=(yi+yf)*0.5;
52 :     dex=(xxx-xi)%30;
53 :     dely=(yyy-yi)%30;
54 :     xi+=dex; yi+=dely;
55 :     xf-=dex; yf-=dely;
56 :     setcolor(AXIS_COLOR);
57 :     line(xi,yyy,xf,yyy); /* x-axis */
58 :     line(xxx,yi,xxx,yf); /* y-axis */
59 :     setcolor(MESH_COLOR);
60 :     for(x=xxx+30;x<=xf;x+=30)
61 :         line(x,yi,x,yf);
62 :     for(x=xxx-30;x>=xi;x-=30)
63 :         line(x,yi,x,yf);
64 :     for(y=yyy+30;y<=yf;y+=30)
65 :         line(xi,y,xf,y);
66 :     for(y=yyy-30;y>=yi;y-=30)
67 :         line(xi,y,xf,y);
68 : }
69 : +

```

```

1 : /*****
2 : FILE NAME : 3COMPILE.C
3 : REQUIRE : I_DEFINE.H
4 :           1STRING.C
5 :           2POLISH.C
6 : SUPPORT : SIMULATE.C
7 : COMMENT : Used for checking the syntetic of program that
8 :           will be executed by function ExecuteFile(); in
9 :           SIMULATE.c
10 : *****/
11 :
12 : /*****
13 : #include "i:\thesis3\i_define.h"
14 : #include "i:\thesis3\1string.c"
15 : #include "i:\thesis3\2polish.c"
16 : *****/
17 :
18 : #define GOTO_C MAX_COMMAND /* MAX_COMMAND is defined in i_define.h */
19 : #define RETURN_C GOTO_C+1
20 : #define ENDEQUOT_C RETURN_C+1
21 : #define ENDPROG_C ENDEQUOT_C+1
22 : #define DEFINE_C ENDPROG_C+1
23 :
24 : typedef struct diagram_struct{
25 :     int def;
26 :     int com;
27 :     int set;
28 :     int equ;
29 :     int eqt;
30 :     int epc;
31 :     int end;
32 :     }Diagram_Type;
33 :
34 : char *error_table[40]=
35 : {
36 :     "CAN NOT USE BEFORE EQUATION-COMMAND", "COMMAND-LINE MUST EXIST",
37 :     "CAN NOT DEFINE IN EQUATION LOOP", "THIS COMMAND-LINE MUST BE REMOVED",
38 :     "EQUATION-COMMAND IS REPEATED", "ENDEQUATION NOT EXIST",
39 :     "EQUATION FORMAT ERROR", "COMMAND-LINE ERROR",
40 :     "ENDPROGRAM MISPLACED" };
41 : Diagram_Type error_index[3]=
42 : {{0,0,0,1,1,2,9},{0,0,0,1,1,0,0},{3,4,5,0,0,6,6}};
43 :
44 : Diagram_Type next_node[3]=
45 : {{0,1,2,-2,-2,-2,-2},{1,1,2,-2,-2,0,-1},{-2,-2,-2,2,1,-2,-2}};
46 :
47 : char equ_fmt[5]="c=e";
48 :
49 : int NextNode(int node,char *com_line,char *error_mes);
50 : int CheckEquation(char *equ_str);
51 : char *ErrMesCompile(int node,int com_index);
52 : int com_fmt_table(char *command);
53 : int CheckSyntetic(char *format,char *check_string);
54 : char *SkipPointer(char data_type,char *start_pntr);
55 : int CheckType(char ch);
56 :
57 : /*****
58 : main()
59 : { char command[80],format[20],err_mes[80];
60 :   static int node_start=0,node_next=0;
61 :   do{
62 :     printf("input: ");
63 :     gets(command);
64 :     node_next=NextNode(node_start,command,err_mes);
65 :     if(node_next!= -2)
66 :       node_start=node_next;
67 :     printf(" start node =%d : ",node_start);
68 :   }while(node_start!=-1);
69 : }
70 : *****/
71 :
72 :
73 : int NextNode(int node,char *com_line,char *error_mes)
74 : /* 0,1,2=node number; -1=terminate; -2=error */
75 : { char command[80],parameter[80];
76 :   int com_index,next;
77 :   /* next=-2; */
78 :   next=node;
79 :   ParseTwoString(com_line,command,parameter);
80 :   com_index=com_fmt_table(command);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :   if(com_index<MAX_FORMAT){ /* command conforms the table */
82 :       if(node!=2){
83 :           if( CheckSyntetic( &(amp;command_fmt[com_index].fmt), parameter) ==0)
84 :               switch( com_index ){ /* no error case */
85 :                   case GOTO_C      : next=next_node[node].com ; break;
86 :                   case RETURN_C    : next=next_node[node].epc ; break;
87 :                   case ENDEQUOT_C  : next=next_node[node].eqt ; break;
88 :                   case ENDPROG_C   : next=next_node[node].end ; break;
89 :                   case DEFINE_C    : next=next_node[node].def ; break;
90 :                   case SET_EQU_COM : next=next_node[node].set ; break;
91 :                   default :       next=next_node[node].com ;
92 :               }
93 :           }
94 :       else /* node==2 */
95 :           if(com_index==ENDEQUOT_C)
96 :               next=next_node[node].eqt;
97 :           else
98 :               next=-2; /* endequation must exist */
99 :       }
100 :   else /* command does not conform the table */
101 :       /* command must be in equation form : format="c=e" */
102 :       if( (CheckSyntetic(equ_fmt, command)==0)&&(node==2) )
103 :           next=next_node[node].equ ; /* 0=no error */
104 :       else
105 :           next=-2; /* equation format error */
106 :   if(next==-2)
107 :       strcpy(error_mes,ErrMesCompile(node,com_index) );
108 :   return(next);
109 : }
110 :
111 :
112 : int CheckEquation(char *equ_str)
113 : /* return 0=no error ; return 1=error */
114 : { Token_type TOKEN[MAXTOKEN];
115 :   if(equ_str[0]=='\0')
116 :       return(1);
117 :   if(TranslateExpression(TOKEN,equ_str)!=0 )
118 :       return(1);
119 :   return(0);
120 :   /* the values returned from TranslateExpression() and CheckEquation()
121 :      have the same meaning */
122 : }
123 :
124 :
125 : char *ErrMesCompile(int node,int com_index)
126 : /* node must be 0,1,2; node=-1 mean terminate */
127 : { int error_ind;
128 :   switch( com_index ){
129 :       case SET_EQU_COM: error_ind=error_index[node].set ; break;
130 :       case GOTO_C     : error_ind=error_index[node].com ; break;
131 :       case RETURN_C   : error_ind=error_index[node].epc ; break;
132 :       case ENDEQUOT_C : error_ind=error_index[node].eqt ; break;
133 :       case ENDPROG_C  : error_ind=error_index[node].end ; break;
134 :       case DEFINE_C   : error_ind=error_index[node].def ; break;
135 :       case MAX_FORMAT : if(node==2)
136 :                               error_ind=7;
137 :                       else
138 :                               error_ind=8;
139 :                       break;
140 :       default :       error_ind=error_index[node].com ;
141 :   }
142 :   return(error_table[error_ind]);
143 : }
144 :
145 :
146 : int com_fmt_table(char *command)
147 : {
148 :     /* table_index(command,command_fmt,MAX_FORMAT); */
149 :     /*******/
150 :     int i=0,equal=1;
151 :     while( (i<MAX_FORMAT) && (equal!=0) )
152 :     { equal=strcmp(command,command_fmt[i].com);
153 :       i++;
154 :     };
155 :     if(equal==0) return(i-1);
156 :     else return(MAX_FORMAT);
157 :     /*******/
158 : }
159 :
160 :

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161: int CheckSyntetic(char *format,char *check_string)
162: /* return 0=no error; return 1=error */
163: { int error=0;
164:   char *fmt_pntr,*check_pntr,*skip_pntr;
165:   InsertStr(format); /* this function is in lstring.c */
166:   fmt_pntr=format;
167:   check_pntr=check_string;
168:   while( (*check_pntr)==' ' )
169:     check_pntr++;
170:   while( (*fmt_pntr)==' ' )
171:     fmt_pntr++;
172:   while( ((*fmt_pntr)!='\0') && (error==0) )
173:     if( (skip_pntr=SkipPointer(*fmt_pntr,check_pntr))==check_pntr )
174:       error=1;
175:   else{
176:     check_pntr=skip_pntr;
177:     while( (*check_pntr)==' ' )
178:       check_pntr++;
179:     while( (*fmt_pntr)==' ' )
180:       fmt_pntr++;
181:     fmt_pntr++;
182:     if( *fmt_pntr=='r' ){
183:       if( *check_pntr!='\0' )
184:         fmt_pntr=format; /* return to the start of format */
185:       else
186:         fmt_pntr++; /* skip over 'r' */
187:     }
188:   }
189:   if((*check_pntr)!='\0')
190:     return(1); /** error !! */
191:   return(error);
192: }
193:
194:
195:
196: char *SkipPointer(char data_type,char *start_pntr)
197: /* data_type : c=char, s=string, n=number */
198: /* start_pntr will be automatically changed to the next word */
199: /* return : start_pntr */
200: {
201:   data_type=tolower(data_type);
202:   switch(data_type){
203:     case 'c': if( CheckType(*start_pntr)==1 )
204:               start_pntr++;
205:               return(start_pntr);
206:     /* break; is not necessary */
207:     case 's': if( CheckType(*start_pntr)==1 )
208:               start_pntr++;
209:               else
210:                 return(start_pntr); /* the first char will be exist */
211:               while((CheckType(*start_pntr)==1)|| (CheckType(*start_pntr)==2))
212:                 start_pntr++;
213:               return(start_pntr);
214:     case 'n': if( *start_pntr=='-' )
215:               start_pntr++;
216:               while( CheckType(*start_pntr)==2 )
217:                 start_pntr++;
218:               if( *start_pntr=='.' ){
219:                 start_pntr++;
220:                 while( CheckType(*start_pntr)==2 )
221:                   start_pntr++;
222:               }
223:               return(start_pntr);
224:     case 'e': if( CheckEquation(start_pntr)==0 )
225:               while( *start_pntr !='\0' )
226:                 start_pntr++;
227:               return(start_pntr);
228:     case 'p': /* case 'p' and 'f' are not checked */
229:     case 'f': while( *start_pntr !='\0' )
230:               start_pntr++;
231:               return(start_pntr);
232:     default : if((*start_pntr)==data_type)
233:               start_pntr++;
234:               return(start_pntr);
235:   };
236: }
237: ↑

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : /*****
2 : FILE NAME : JSET_FN.C
3 : LEVEL : 3
4 : REQUIRE : 1DEFINE.H
5 :           1PLOTGH.C
6 :           1STRING.C
7 :           2POLISH.C
8 :           (ErrorMessage not exist)
9 : SUPPORT : MODELRODT.C
10 : DESCRIPTION: EVALUATE A TIME FUNCTION BY CONVERTING TO ARRAY OF DATA
11 : ARGUMENT : DATA= pointer to the first position of data storage
12 :           width= number of array of data
13 :           time_step= time step for interating the time fuction
14 :           init_time= initial time
15 : *****/
16 :
17 : void SetFunction(Value_type *DATA,int width,Value_type time_step,
18 :                 Value_type init_time);
19 : void SetStrFunction(char *equation_str, Value_type *DATA, int width,
20 :                    Value_type time_step, Value_type init_time);
21 : char *EvaluateStr(char *expression_str);
22 :
23 :
24 : void SetFunction(Value_type *DATA,int width,Value_type time_step,
25 :                 Value_type init_time)
26 : { char equation_str[MAXSTRING];
27 :   Token_type equ_token[MAXTOKEN];
28 :   Var_type equ_vartype[MAXTOKEN];
29 :   int error=0;
30 :   do{
31 :     printf("\nINPUT EQUATION (time domain) : f(t)\n");
32 :     printf("EQUATION: f(t) = ");
33 :     scanf("%s",equation_str);
34 :     if( (error=PolishToken(equ_token,equ_vartype,equation_str)) !=0){
35 :       printf("EQUATION FORMAT ERROR!!");
36 :       /* PolishToken() return nonzero imply error */
37 :       /* return zero imply correct expression */
38 :       /* return negative value to indicate index of error case */
39 :       /* -1:ErrorMessage("[FindWord]: Variable/Operator not defined.");*/
40 :       /* -2:ErrorMessage("[ExtractWord]: Word is truncated."); */
41 :       /* -3:ErrorMessage("[FindSymbol]: Syntax/Symbol error."); */
42 :       /* -4:ErrorMessage("[Syntetic]: expression is incomplete."); */
43 :     }
44 :   }while(error!=0);
45 :   SetStrFunction(equation_str, DATA, width, time_step, init_time);
46 : }
47 :
48 :
49 : void SetStrFunction(char *equation_str, Value_type *DATA, int width,
50 :                    Value_type time_step, Value_type init_time)
51 : {
52 :   Value_type var_val[26],*var_t_pntr,*data_pntr;
53 :   Token_type equ_token[MAXTOKEN];
54 :   Var_type equ_vartype[MAXTOKEN];
55 :   int index,var_t_index;
56 :
57 :   PolishToken(equ_token,equ_vartype,equation_str);
58 :   data_pntr=DATA;
59 :   var_t_pntr=&var_val['t'-'a'];
60 :   (*var_t_pntr)=0;
61 :   for(index=0; index<width; index++){
62 :     ReadVariables(var_val,equ_token,equ_vartype);
63 :     (*data_pntr)=EvaluatePosfix(equ_token);
64 :     data_pntr++;
65 :     (*var_t_pntr)+= time_step;
66 :   }
67 : }
68 :
69 : char *EvaluateStr(char *expression_str)
70 : {
71 :   Value_type data;
72 :   Token_type equ_token[MAXTOKEN];
73 :   Var_type equ_vartype[MAXTOKEN];
74 :
75 :   if( PolishToken(equ_token,equ_vartype,expression_str)!= 0)
76 :     return(NULL); /* syntetic error */
77 :   if(equ_vartype[0].name_index!=-1) /** VARIABLE MUST NOT EXIST ***/
78 :     return(NULL);
79 :   data=EvaluatePosfix(equ_token);
80 :   return( itoa(data,expression_str,10) );

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
81 :  
82 :  /***** SHOULD USE ecvt() *****/  
83 :  
84 : }  
85 :  
86 : ↑
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : /*****
2 : FILE NAME : 3FOURIER.C
3 : LEVEL : 3
4 : PROGRAMMER: PATPONG LAOSUWAN
5 : SUPPORT : SIMULATE.C
6 : REQUIRE : KEY.H
7 :
8 :             I_DEFINE.H
9 :             1PLOTGPH.C,1STRING.C,1FILE.C
10 :            2MANYPLT.C
11 :            MEM_MNG.C
12 :            (ErrorMessage not exist)
13 : *****/
14 : #define PHI 3.141593
15 :
16 : void filter(int width,int low_cutoff,int high_cutoff,Value_type *An,
17 :            Value_type *Bn);
18 : void Modulus(Value_type *spectrum,int width,Value_type *An,Value_type *Bn);
19 : void Fourier(Value_type *signal,int width,Value_type *An,Value_type *Bn);
20 : void InvFourier(Value_type *signal,int width,Value_type *An,Value_type *Bn);
21 :
22 :
23 : void filter(int width,int low_cutoff,int high_cutoff,Value_type *An,
24 :            Value_type *Bn)
25 : /* "WINDOW" Filter */
26 : /* Remove the spectrum that are out of (low_cutoff,high_cutoff) */
27 : { int i;
28 :
29 :     if(low_cutoff<width)
30 :         for(i=0; i<low_cutoff; i++)
31 :             An[i]=Bn[i]=0;
32 :     if(high_cutoff>low_cutoff)
33 :         for(i=high_cutoff; i<width; i++)
34 :             An[i]=Bn[i]=0;
35 : }
36 :
37 :
38 : void Modulus(Value_type *spectrum,int width,Value_type *An,Value_type *Bn)
39 : { int k;
40 :   for(k=0; k<width; k++)
41 :       spectrum[k]=sqrt( An[k]*An[k]+Bn[k]*Bn[k] );
42 : }
43 :
44 :
45 : void Fourier(Value_type *signal,int width,Value_type *An,Value_type *Bn)
46 : /* return the modulus of cos&sin component through "An & Bn" */
47 : { Value_type radian,factor_kn,factor,factor1;
48 :   int n,k,range_spectrum;
49 :   int text[500];
50 :
51 :   range_spectrum=ceil( width/2 );
52 :   /* range_spectrum=width; */
53 :
54 :   for(n=0; n<width; n++)
55 :       An[n]=Bn[n]=0; /* initialize An[],Bn[] */
56 :
57 :   for(n=0; n<width; n++)
58 :       An[0]+=signal[n]; /* Evaluate An[0] */
59 :   An[0]/=width;
60 :
61 :   Fastgettext(20,7,50,12,text);
62 :   makewin(20,7,50,12,0x2E,0x20,1);
63 :   Fastputs(24,8,0x21,"FAST FOURIER TRANSFORM");
64 :   Fastputs(27,9,0x20,"LOOP COUNT DOWN");
65 :
66 :   radian=2*PHI/width;
67 :   factor1=2.0/width;
68 :   for(k=1; k<range_spectrum; k++){
69 :       factor_kn=0;
70 :       for(n=0; n<width; n++){
71 :           factor=factor_kn*radian; /* factor=k*2*PI*n/N */
72 :           An[k]+=signal[n]*cos(factor);
73 :           Bn[k]+=signal[n]*sin(factor);
74 :           factor_kn+=k; /* factor_kn=k*n */
75 :       }
76 :       An[k]*=factor1; /* factor1= 2/width */
77 :       Bn[k]*=factor1;
78 :       Fastprintf(32,11,0x21,"%5d",range_spectrum-k);
79 :   }
80 :   for(k=0; k<range_spectrum; k++){

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
81 :         An[width-k-1]=An[k];
82 :         Bn[width-k-1]=Bn[k];
83 :     }
84 :     Fastputtext(20,7,50,12,text);
85 : }
86 :
87 :
88 : void InvFourier(Value_type *signal,int width,Value_type *An,Value_type *Bn)
89 : /* return the reconstructed signal through "signal" */
90 : { Value_type radian,factor_kn,factor;
91 :   int n,k,range_spectrum;
92 :   int text[500];
93 :
94 :   range_spectrum=width;
95 :
96 :   for(n=0; n<width; n++)
97 :       signal[n]=An[0];
98 :
99 :   Fastgettext(20,7,50,12,text);
100 :   makewin(20,7,50,12,0x2E,0x20,1);
101 :   Fastputs(29,8,0x21,"INVERSE FFT");
102 :   Fastputs(27,9,0x20,"LOOP COUNT DOWN");
103 :
104 :   radian=2*PI/width;
105 :   for(n=0; n<width; n++){
106 :       factor_kn=0;
107 :       for(k=1; k<range_spectrum; k++){
108 :           factor_kn+=n; /* factor_kn=k*n */
109 :           factor=factor_kn*radian; /* factor=k*2*PI*n/N */
110 :           signal[n]+=An[k]*cos(factor);
111 :           signal[n]+=Bn[k]*sin(factor);
112 :       }
113 :       Fastprintf(32,11,0x21,"%5d",width-n);
114 :   }
115 :   Fastputtext(20,7,50,12,text);
116 : }
117 :
118 : ↑
```

```

1 : /*****
2 : FILE NAME : PHASEPLN.C
3 : REQUIRE   : KEY.H
4 :
5 :             I_DEFINE.H
6 :             ISTRING.C
7 :             1PLOTGPH.C
8 :             2FRAME.C      (independent from simulate.c)
9 :             2POLISH.C
10 :            UTIL.C
11 : SUPPORT    : MAINP.C (directly)
12 : COMMENT    : It can be executed independently from (include file must exist)
13 : *****/
14 : #define X_MAX    getmaxx()+1
15 : #define Y_MAX    getmaxy()+1
16 : #define X_CENTER X_MAX*0.5
17 : #define Y_CENTER Y_MAX*0.5
18 : #define SHRINK_X  4
19 : #define SHRINK_Y  1
20 : #define SHIFT_X   SHRINK_X*60/2
21 : #define SHIFT_Y   SHRINK_Y*60/2
22 : #define NONL_DEGREE  2
23 : #define PRECISION_SCALE 200
24 : #define LL_SCALE     15
25 :
26 : #include "i:\thesis3\2frame.c"
27 :
28 : /***** data type definition *****/
29 : typedef struct field {float x ; float y;}FIELD;
30 :
31 : void MainPhasePlane(void);
32 : void PhasePlaneAnalysis(void);
33 : int  get_input_equation(void);
34 : void string_header(int xi,int yi,int xf,int yf,char *string,int str_length,
35 :                   int num_str);
36 : void draw_block(int xi,int yi,int xf,int yf,int color);
37 : void draw_block(int xi,int yi,int xf,int yf,int color);
38 : int  inside(FIELD xy);
39 : void set_begin_state(float *x,float *y,FIELD *pntr);
40 : void plot(FIELD xy);
41 : void perpendicular (FIELD *pointxy,FIELD *fieldxy_90);
42 : void evaluate_direction(FIELD *direction,FIELD *at_point);
43 :
44 : /***** gobal *****/
45 : Value_type variable_value[26];
46 : Token_type lexicon[NONL_DEGREE][MAXTOKEN];
47 : Var_type   var_p[NONL_DEGREE][MAXTOKEN];
48 : char       equ_string[NONL_DEGREE][MAXSTRING];
49 :
50 :
51 : FIELD begin_point[4]=
52 :     {{0.5,0.5},{-0.5,-0.5},{0.5,-0.5},{-0.5,0.5}};
53 : /* {{0,0.7},{0,-0.7},{0.7,0},{-0.7,0}}; */
54 :
55 : extern int color_mode;
56 :
57 : float field_scale,scale_x,scale_y,step_r,iteration_time;
58 : int   x_width,y_width;
59 : int   PLOT_COLOR,TOPIC_COLOR;
60 :
61 : static int num_line=5,num_point=500;
62 : static float range_x=5,range_y=3.5;
63 :
64 : void MainPhasePlane(void)
65 : #define LINE 6
66 : #define LENG 18
67 : {
68 :     register int i=1;
69 :     char str[MAXCOL],st[LINE][LENG]="Equation","Line number",
70 :                                     "Point number","X range","Y range","Plot";
71 :     float value;
72 :     int txt[500],ready_run,error;
73 :
74 :     ready_run=0;
75 :     Fastgettext(4*BLOCK+5,4,4*BLOCK+LENG+10,LINE+5,txt);
76 :     do{
77 :         i=Menu(4*BLOCK+5,4,0x36,0x30,0x4F,&st,LINE,
78 :                LENG,i);
79 :         if(i<0) break;
80 :         i=abs(i);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :         switch(i){
82 :             case 1: error=get_input_equation();
83 :                 switch(error){
84 :                     case 1: ErrorMessage(0,"EQUATION FORMAT ERROR");
85 :                     case -1: ready_run=0; /* equation format error */
86 :                         break;
87 :                     case 0 : ready_run=1;
88 :                         break;
89 :                 }
90 :             break;
91 :             case 2: strcpy(str,Show_Get_message(3*BLOCK,i+5,0x6F,
92 :                 "<5>",1) );
93 :                 if( (value=atoi(str)) >0)
94 :                     num_line=(int) value;
95 :                 break;
96 :             case 3: strcpy(str,Show_Get_message(3*BLOCK,i+5,0x6F,
97 :                 "<500>",1) );
98 :                 if( (value=atoi(str)) >0)
99 :                     num_point=(int) value;
100 :                 break;
101 :             case 4: strcpy(str,Show_Get_message(3*BLOCK,i+5,0x6F,
102 :                 "<5.0>",1) );
103 :                 if( (value=atof(str)) >0)
104 :                     range_x=value;
105 :                 break;
106 :             case 5: strcpy(str,Show_Get_message(3*BLOCK,i+5,0x6F,
107 :                 "<3.5>",1) );
108 :                 if( str[0]!='\0' )
109 :                     range_y=atof(str);
110 :                 break;
111 :             case 6: if(ready_run==1)
112 :                 PhasePlaneAnalysis();
113 :             else
114 :                 ErrorMessage(0,"EQUATION NOT SET");
115 :             break;
116 :         }
117 :     }while(i!=0);
118 :     Fastputtext(4*BLOCK+5,4,4*BLOCK+LENG+10,LINE+5,tst);
119 :
120 :     /** x_f-x_i-10-X_MARGIN, y_f-y_i-10-Y_MARGIN must be divisible by 60 ***/
121 :     /** X_MARGIN=30, Y_MARGIN=50 ***/
122 :     /** x_width=x_f-x_i; y_width=y_f-y_i; ***/
123 :     x_width=X_MAX-60*SHRINK_X; /* 60 must not */
124 :     y_width=Y_MAX-60*SHRINK_Y; /* 60 must not */
125 :
126 :     scale_x=(X_CENTER-5-0.5*X_MARGIN-0.5*(X_MAX-x_width))/(range_x);
127 :     scale_y=(Y_CENTER-5-0.5*Y_MARGIN-0.5*(Y_MAX-y_width))/(range_y);
128 :     /** 5=edge_thick **/
129 :     field_scale=range_x/LL_SCALE;
130 :     step_r=range_x/PRECISION_SCALE;
131 :
132 :     if(color_mode==0){
133 :         PLOT_COLOR= WHITE;
134 :         TOPIC_COLOR= WHITE;
135 :     }
136 :     else{
137 :         PLOT_COLOR= LIGHTMAGENTA;
138 :         TOPIC_COLOR= LIGHTMAGENTA;
139 :     }
140 : }
141 : }
142 :
143 :
144 :
145 : void PhasePlaneAnalysis(void)
146 : {
147 :     FIELD point,direction,track1,track2;
148 :     FIELD xy1,xy2,xy0;
149 :
150 :     FIELD *begin_pntr;
151 :     float r1,r2,x_begin,y_begin;
152 :     int i,j,sign,begin_index;
153 :     int x1,y1,x2,y2;
154 :
155 :     int a=DETECT,b;
156 :
157 :     initgraph(&a,&b,"");
158 :     x1=(X_MAX-x_width)*0.5+SHIFT_X;
159 :     y1=(Y_MAX-y_width)*0.5+SHIFT_Y;
160 :     x2=(X_MAX+x_width)*0.5+SHIFT_X;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161:  y2=(Y_MAX+y_width)*0.5+SHIFT_Y;
162:  string_header(x1-2*SHIFT_X,y1,x1-5,y2,&equ_string[0][0],MAXSTRING,2);
163:  draw_frame(x1,y1,x2,y2,color_mode);
164:
165:  begin_pntr=begin_point;
166:  direction.x=0; direction.y=0;
167:  for(begin_index=1;begin_index<=4;begin_index++){
168:      set_begin_state(&x_begin,&y_begin,begin_pntr++);
169:      for(sign=-1;sign<2;sign+=2){
170:          xy0.x=xy1.x=xy2.x=point.x=(x_begin-sign*direction.x);
171:          xy0.y=xy1.y=xy2.y=point.y=(y_begin-sign*direction.y);
172:
173:          for(i=0;i<num_line;i++){
174:              { point.x+=sign*direction.x;
175:                  point.y+=sign*direction.y;
176:                  xy1.x=xy2.x=point.x;
177:                  xy1.y=xy2.y=point.y;
178:                  j=0;
179:                  do{
180:                      if( inside(xy1) ){
181:                          plot(xy1);
182:                          evaluate_direction(&track1,&xy1);
183:                          r1=sqrt(track1.x*track1.x+track1.y*track1.y)/step_r;
184:                          xy1.x+=track1.x/r1; xy1.y+=track1.y/r1;
185:                      }
186:
187:                      if( inside(xy2) ){
188:                          plot(xy2);
189:                          evaluate_direction(&track2,&xy2);
190:                          r2=sqrt(track2.x*track2.x+track2.y*track2.y)/step_r;
191:                          xy2.x-=track2.x/r2; xy2.y-=track2.y/r2;
192:                      }
193:
194:                      j++;
195:                  }while( (j<num_point) );
196:                  perpendicular (&xy0,&direction);
197:              }
198:          }
199:      }
200:  setcolor(TOPIC_COLOR);
201:  settextstyle(1,0,3);
202:  outtextxy(190+SHIFT_X,3+SHIFT_Y+0.5*(Y_MAX-y_width),
203:      "PHASE PLANE ANALYSIS");
204:  getch();
205:  closegraph();
206:  clrscr();
207: }
208:
209: int get_input_equation(void)
210: { Token_type TOKEN[MAXTOKEN];
211:   int index,txt[600],error=0;
212:   struct text_info tx;
213:
214:   gettextinfo(&tx);
215:   Fastgettext(3*BLOCK,4,3*BLOCK+47,11,txt);
216:   makewin(3*BLOCK,4,3*BLOCK+47,11,0x2F,0x2F,1);
217:   window(1,1,80,25);
218:   Fastputs(3*BLOCK+3,5,0x21,"");
219:   Fastputs(3*BLOCK+3,6,0x21,"INPUT EQUATION: x = f(x,y) and y = g(x,y)");
220:   cursorShape(CURSORMON);
221:   for(index=0;index<NONL_DEGREE;index++){
222:       Fastputs(3*BLOCK+3,7+2*index,0x20,"");
223:       Fastprintf(3*BLOCK+3,8+2*index,0x20,"EQUATION: %c = ','x'+index);
224:       gotoxy(3*BLOCK+19,8+2*index);
225:       getstr(equ_string[index],20,0x20);
226:       if(equ_string[0]!='\0'){
227:           if( PolishToken lexicon[index],var_p[index],equ_string[index])!=0){
228:               error=1; break;
229:           }
230:       }
231:       else{
232:           error=-1; break;
233:       }
234:   }
235:   cursorShape(CURSROFF);
236:   Fastputtext(3*BLOCK,4,3*BLOCK+47,11,txt);
237:   window(tx.winleft,tx.wintop,tx.winright,tx.winbottom);
238:   return(error);
239: }
240:

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

241: void string_header(int xi,int yi,int xf,int yf,char *string,int str_length,
242:                    int num_str)
243: {
244:     int i,x,y,line_space;
245:     line_space=(yf-yi)/(2*num_str+4+5);
246:     setttextstyle(0,0,1);
247:     draw_block(xi,yi,xf,yf,WHITE);
248:     x=xi+5;
249:     y=yi+line_space;
250:     setcolor(WHITE);
251:     for(i=0; i<num_str; i++){
252:         gprintf(&x,&y," EQUATION %d :",i+1);
253:         y+=line_space;
254:         outtextxy(x+7,y-7,".");
255:         gprintf(&x,&y," %c = %s", 'x'+i, string+i*str_length);
256:         y+=line_space;
257:     }
258:     y+=line_space;
259:     gprintf(&x,&y," RANGE OF X-AXIS :"); y+=line_space;
260:     gprintf(&x,&y,"      (-%3.2f,%3.2f)",range_x,range_x);
261:     y+=line_space;
262:     gprintf(&x,&y," RANGE OF Y-AXIS :"); y+=line_space;
263:     gprintf(&x,&y,"      (-%3.2f,%3.2f)",range_y,range_y);
264: }
265:
266: void draw_block(int xi,int yi,int xf,int yf,int color)
267: {
268:     setcolor(color);
269:     line(xi,yi,xf,yi);
270:     line(xi,yi,xi,yf);
271:     line(xf,yf,xf,yi);
272:     line(xf,yf,xi,yf);
273: }
274:
275: int inside(FIELD xy)
276: {
277:     return( ( fabs(xy.x)<range_x ) && ( fabs(xy.y)<range_y ) );
278: }
279:
280: void set_begin_state(float *x,float *y,FIELD *pntr)
281: {
282:     *x= range_x* (*pntr).x;
283:     *y= range_y* (*pntr).y;
284: }
285:
286: void plot(FIELD xy)
287: {
288:     putpixel( SHIFT_X+X_CENTER+xy.x*scale_x, SHIFT_Y+Y_CENTER-xy.y*scale_y,
289:              PLOT_COLOR);
290: }
291:
292: void perpendicular (FIELD *pointxy,FIELD *fieldxy_90)
293: {
294:     FIELD fieldxy;
295:     float r,c;
296:     evaluate_direction(&fieldxy,pointxy);
297:     if( (fieldxy).y !=0 ){
298:         c=(fieldxy).x/(fieldxy).y;
299:         r=sqrt(1+c*c);
300:         (*fieldxy_90).x=field_scale/r;
301:         (*fieldxy_90).y=-c*field_scale/r;
302:     }
303:     else if( (fieldxy).x !=0 ){
304:         (*fieldxy_90).x=0;
305:         (*fieldxy_90).y=field_scale;
306:     }
307:     else{
308:         (*fieldxy_90).x=field_scale;
309:         (*fieldxy_90).y=field_scale;
310:     }
311: }
312:
313: void evaluate_direction(FIELD *direction,FIELD *at_point)
314: {
315:     Value_type value[NONL_DEGREE]; /* x,y */
316:     int index;
317:     variable_value[23]=(*at_point).x;  /** 23='x'-'a'; ASC_x=120 **/
318:     variable_value[24]=(*at_point).y;  /** 24='y'-'a'; ASC_y=121 **/
319:     for(index=0;index<NONL_DEGREE;index++){

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
321:   ReadVariables(variable_value,lexicon[index],var_p[index]);
322:   value[index]=EvaluatePosfix(lexicon[index]); /* invariant time */
323: }
324: (*direction).x=value[0];
325: (*direction).y=value[1];
326: }
327:
328: ↑
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : /*****
2 : FILE NAME : LINEAR.C
3 : REQUIRE   : I_DEFINE.H
4 :           : 1STRING.C
5 :           : 1FILE.C
6 :           : 1PLOTGPH.C
7 :           : 2MANYPLT.C
8 :           : 2POLISH.C
9 :           : 3SET_FN.C      (this file support linear.c only)
10 :          : (ErrorMessage not exist)
11 : SUPPORT   : MAINP.C (directly)
12 : COMMENT   : It can be executed independently from (include file must exist)
13 : *****/
14 :
15 :
16 : #include "i:\thesis3\3set_fn.c"
17 :
18 : #define MAX_LIN_ORDER 5
19 : #define MAXSTRING 80
20 : #define maxx gettmaxx()+1
21 : #define maxy getmaxy()+1
22 :
23 : #define xx 50
24 : #define yy 250
25 : #define yy1 440
26 :
27 : int Linear();
28 : int allocate_io(int width);
29 : int allocatemem(float *value_ptr,int width);
30 : void calculate_next_state(float *X,float *Bn,int order,int time_index);
31 : int get_parameter(int *order);
32 : void set_gobal_parameter();
33 : void find_Bn(float *Bn,float *ii,float *oo,int order);
34 : void set_system_cof(float *ii,float *oo,int order_input,int choice);
35 :
36 :
37 : /***** gobal *****/
38 : int state_output,MAX_TIMEDOMAIN;
39 : float time_range, iteration_time, iterate_scale;
40 : float *INPUT,*OUTPUT,X[MAX_LIN_ORDER],Bn[MAX_LIN_ORDER],
41 : output_cof[MAX_LIN_ORDER],input_cof[MAX_LIN_ORDER];
42 :
43 :
44 : int Linear()
45 : {
46 :     int order,time_index;
47 :     int i,j;
48 :     float *DATA[3],time;
49 :
50 :     MAX_TIMEDOMAIN=600;
51 :     if(allocate_io(MAX_TIMEDOMAIN+1))
52 :         return(1); /* cannot allocate */
53 :     while( get_parameter(&order)==1 )
54 :         /*error case*/;
55 :     find_Bn(Bn,input_cof,output_cof,order);
56 :
57 :     for(i=0;i<MAX_LIN_ORDER;i++) X[i]=0;
58 :     for(i=1;i<=MAX_TIMEDOMAIN;i++) OUTPUT[i]=0;
59 :     time=0;
60 :     for(j=time_index=1; j<=MAX_TIMEDOMAIN;j++){
61 :         for(i=1;i<=iterate_scale;i++,time_index++,time+=iteration_time){
62 :             calculate_next_state(X,Bn,order,j);
63 :         }
64 :         OUTPUT[j]=X[state_output];
65 :     }
66 :
67 :     DATA[0]=INPUT;
68 :     DATA[1]=OUTPUT;
69 :     PlotVarMem(DATA,"ab","ab",MAX_TIMEDOMAIN,time_range,0);
70 : }
71 :
72 : int allocate_io(int width)
73 : {
74 :     if(allocatemem(INPUT,width))
75 :         return(1); /* cannot allocate */
76 :     if(allocatemem(OUTPUT,width))
77 :         return(1); /* cannot allocate */
78 :     return(0); /* allocation Succeed */
79 : }
80 :

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
81 : int allocatemem(float *Value_ptr,int width)
82 : {
83 :     Value_ptr=(float *)malloc(width*sizeof(float));
84 :     if(Value_ptr==NULL){
85 :         cprintf("\n\r Cannot Allocate MEMORY");
86 :         return(1); /* error case */
87 :     }
88 :     return(0);
89 : }
90 :
91 : void calculate_next_state(float *X,float *Bn,int order,int time_index)
92 : {
93 :     int KUTTA,i,n;
94 :     float Kn[5][MAX_LIN_ORDER],value,factor;
95 :
96 :     for(i=1;i<=order;i++) Kn[0][i]=0;
97 :     for(KUTTA=1;KUTTA<=4;KUTTA++){
98 :         if( (KUTTA==1)||!(KUTTA==4)) factor=1;
99 :         else factor=0.5;
100 :         for(i=1;i<=order;i++){
101 :             value=0;
102 :             if(i<order){
103 :                 value+=(X[i+1]);
104 :                 value+=factor*Kn[KUTTA-1][i+1];
105 :             }
106 :             else
107 :                 for(n=1;n<=order;n++){
108 :                     value-=output_cof[n-1]*(X[n]+factor*Kn[KUTTA-1][n]);
109 :                 }
110 :             value+=(Bn[i])*INPUT[time_index];
111 :             Kn[KUTTA][i]=iteration_time*value;
112 :         }
113 :     }
114 :     for(n=1;n<=order;n++){
115 :         for(KUTTA=1;KUTTA<=4;KUTTA++){
116 :             if(fabs(KUTTA-2.5)<1)
117 :                 X[n]+=Kn[KUTTA][n]/3;
118 :             else
119 :                 X[n]+=Kn[KUTTA][n]/6;
120 :         }
121 :     }
122 : }
123 : void find_Bn(float *Bn,float *ii,float *oo,int order)
124 : {
125 :     int n,nn,i;
126 :     *Bn=0;
127 :     for(i=1;i<=order;i++){
128 :         {ii[i]/=oo[order];
129 :         oo[i]/=oo[order];
130 :         }
131 :     for(n=1;n<=order;n++){
132 :         { *(Bn+n)=ii[order-n];
133 :         for(nn=1;nn<=n;nn++)
134 :             *(Bn+n)-=oo[order-nn]*Bn[n-nn];
135 :         }
136 :     }
137 : }
138 : int get_parameter(int *order)
139 : #define LINE_L 7
140 : #define LENG_L 16
141 : {int txt[500];
142 : register int i=1,value;
143 : char str[80],st[LINE_L][LENG_L]={"Input function","Time range",
144 : "Graph width","System order","System cof.",
145 : "Precision","Run"};
146 :
147 : set_gobal_parameter();
148 : Fastgettext(4*BLOCK+5,4*BLOCK+LENG_L+10,LINE_L+5,txt);
149 : do{
150 :     i=Menu(4*BLOCK+5,4,0x36,0x30,0x4F,&st,LINE_L,
151 :     LENG_L,i);
152 :     if(i<0) break;
153 :     i=abs(i);
154 :     switch(i){
155 :     case 1: strcpy(str,Show_Get_message(3*BLOCK,i+5,0x6F,
156 : "Input F(t)",1));
157 :         if(str[0]!='\0')
158 :             SetStrFunction(str,INPUT, MAX_TIMEDOMAIN,
159 :             iteration_time*iterate_scale, 0 );
160 :         break;
161 :     case 2: strcpy(str,Show_Get_message(3*BLOCK,i+5,0x6F,
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161:         "Time range<Sec>",1) );
162:         if( str[0]!='\0' )
163:             time_range=atof(str);
164:         cprintf("\n\r(Local) Time range= ..received!!",time);
165:         break;
166:     case 3: strcpy(str,Show_Get_message(3*BLOCK,i+5,0x6F,
167:         "Graph width",1).);
168:         if( (value=atoi(str)) >0 ){
169:             MAX_TIMEDOMAIN=value;
170:             allocate_io(MAX_TIMEDOMAIN+1);
171:         }
172:         cprintf("\n\r(Local) width= ..received!!",MAX_TIMEDOMAIN);
173:         break;
174:     case 4: strcpy(str,Show_Get_message(3*BLOCK,i+5,0x6F,
175:         "System order",1).);
176:         if( str[0]!='\0' )
177:             *order=atoi(str);
178:         cprintf("\n\r(Local) system order= ..received!!",*order);
179:         break;
180:     case 5: if( *order!=0 )
181:         set_system_cof(input_cof,output_cof,*order,i);
182:         /*i=choice*/
183:         break;
184:     case 6: strcpy(str,Show_Get_message(3*BLOCK,i+5,0x6F,
185:         "Precision",1).);
186:         if( str[0]!='\0' )
187:             iteration_scale=atoi(str);
188:         cprintf("\n\r(Local) Precision= ..received!!",
189:             iteration_scale);
190:         break;
191:     case 7: i=0; /* to exit loop while */
192:         break;
193:     }
194: }while(i!=0);
195: Fastputtext(4*BLOCK+5,4,4*BLOCK+LENG_L+10,LINE_L+5,tst);
196: iteration_time=time_range/(iterate_scale*MAX_TIMEDOMAIN);
197: return(0); /* return means "ok" : succeed */
198: }
199:
200: void set_gobal_parameter()
201: { state_output=1;
202:   time_range=10;
203:   iterate_scale=10;
204:   iteration_time=time_range/(iterate_scale*MAX_TIMEDOMAIN);
205: }
206:
207: void set_system_cof(float *ii,float *oo,int order_input,int choice)
208: { int i;
209:   char str[40];
210:
211:   /* makewin(WINL-2,WINT-1,WINR+2,WINB+1,0x7F,0x31,0); */
212:   cprintf("\n\rINPUT TRANSFER FUNCTION :");
213:   cprintf("\n\r  A0 + A1*s + A2*s^2 + A3*s^3 + ... + A(n-1)*s^(n-1)");
214:   cprintf("\n\r  B0 + B1*s + B2*s^2 + B3*s^3 + ... + B(n)*s^n");
215:   cprintf("\n\rINPUT A0...TO...A(%d) : \n ",order_input-1);
216:   for(i=0;i<order_input;i++){
217:       sprintf(str,"INPUT A(%d) : ",i);
218:       strcpy(str,Show_Get_message(3*BLOCK,choice+3,0x6F,
219:           str,1));
220:       ii[i]=atof(str);
221:   }
222:   ii[order_input]=0;
223:   cprintf("\n\rINPUT B0...TO...B(%d) :  B(%d)!=0 ",order_input,
224:       order_input);
225:   i=0;
226:   while( i<=order_input ){
227:       sprintf(str,"INPUT B(%d) : ",i);
228:       strcpy(str,Show_Get_message(3*BLOCK,choice+3,0x6F,
229:           str,1));
230:       oo[i]=atof(str);
231:       if( (i==order_input)&&(oo[i]==0) ){
232:           i--;
233:           cprintf("\n\rB(%d) must not be zero.",order_input);
234:       }
235:       i++;
236:   }
237: }
238: }
239:
240: +

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : void Edit(void);
2 : void Title(void);
3 :
4 : int Newline(DNODE far *cur,DNODE far *new,char *str,int *x);
5 : void Show1_page(DNODE far *new);
6 : void ClipSpace(char *str);
7 : void AddSpace(char *str);
8 : void Showlist(DNODE far *node,int numlist,int y);
9 :
10 : extern int MEMFULL;
11 :
12 :
13 : void Edit(void)
14 : #define EDIT_CHOICE 6
15 : {
16 :     DNODE far *new=NULL;
17 :     char str[MAXCOL+MINCOL+10],str1[MAXCOL+MINCOL+10],*pt,*pt1,key;
18 :     char file_name[MAXPATH];
19 :     char editch[EDIT_CHOICE][14]={ "New", "Save", "Load", "Compile",
20 :                                     "Run", "Quit"};
21 :     int df,Line,Insert,choice, txt[400];
22 :     static int x,y;
23 :     register int i;
24 :     struct text_info tf;
25 :
26 :     gettextinfo(&tf);
27 :     window(1,1,80,25);
28 :
29 :     x=MINCOL; y=MINROW; Line=1; Insert=1; choice=1;
30 :     file_name[0]='\0';
31 :     Title();
32 :     Fastputs(MINCOL+(MAXCOL-MINCOL)/2-9,MINROW-1,0x12,"Insert");
33 :     Clearlist(Phead);
34 :     Phead=create_head();
35 :     memset(str,SPACE,MAXCOL);
36 :     str[MAXCOL]='\0';
37 :     Appendlist(Phead,str);
38 :     pt=str;
39 :     new=Phead->first;
40 :     cursorShape(CURSORON);
41 :     for(;;){
42 :         Fastprintf(MINCOL+5,MINROW-1,0x12,
43 :             "Line %-5d Col %-5d ",Line,x-MINCOL+1);
44 :         gotoxy(x,y); Setattr(x,y,1,CUR);
45 :         key=getch(); Setattr(x,y,1,NORM);
46 :         if(key!=0){
47 :             switch(key){
48 :                 case CTRL_N:if(!MEMFULL){
49 :                     ClipSpace(str);
50 :                     strcpy(new->string,str);
51 :                     new=Insertlist(Phead,new);
52 :                     strcpy(str,new->string);
53 :                     pt=str;
54 :                     movetext(MINCOL,y,MAXCOL,MAXROW-1,
55 :                         MINCOL,y+1);
56 :                     Cls(MINCOL,y,MAXCOL,y,NORM,SPACE);
57 :                 }
58 :                 else Memory_state();
59 :                 break;
60 :                 case CTRL_Y: new=Dellist(Phead,new);
61 :                     movetext(MINCOL,y+1,MAXCOL,MAXROW,
62 :                         MINCOL,y);
63 :                     Cls(MINCOL,MAXROW,MAXCOL,MAXROW,
64 :                         NORM,SPACE);
65 :                     strcpy(str,new->string);
66 :                     pt=str;
67 :                     pt+=Newline(new,new,str,&x);
68 :                     if(new->next){
69 :                         df=MAXROW-y;
70 :                         for(i=0;i<=df && new->next;i++){
71 :                             new=new->next;
72 :                             Showlist(new->prev,1,y+i-1);
73 :                             for(df=i-1,i=0;i<=df;i++){
74 :                                 new=new->prev;
75 :                             }
76 :                             break;
77 :                         }
78 :                     case BSP:if(x>MINCOL){
79 :                         if(*--pt==TAB){
80 :                             movetext(x,y,MAXCOL,y,

```

```

81 :             x-=TAB_SIZE;
82 :         }
83 :         else{
84 :             movetext(x,y,MAXCOL,y,x-1,y);
85 :             x--;
86 :         }
87 :         for(pt1=pt,i=strlen(pt)-1;i>0;i--){
88 :             *pt1=*(pt1+1);
89 :             pt1++;
90 :         }
91 :     }
92 :     break;
93 : case CR:if(!MEMFULL){
94 :     if(Insert){
95 :         ClipSpace(str);
96 :         strcpy(new->string,str);
97 :         new=Insertlist(Phead,new);
98 :         Cls(x,y,MAXCOL,y,NORM,SPACE);
99 :         x=MINCOL;
100 :         if(y<MAXROW-1){
101 :             y++;
102 :             movetext(MINCOL,y,MAXCOL,
103 :                 MAXROW-1,MINCOL,y+1);
104 :         }
105 :         else
106 :             movetext(MINCOL,MINROW+1,
107 :                 MAXCOL,y,MINCOL,MINROW);
108 :         Cls(MINCOL,y,MAXCOL,y,
109 :             NORM,SPACE);
110 :         strcpy(str,new->string);
111 :         pt=str;
112 :         AddSpace(str);
113 :         Showlist(new,1,y);
114 :     }
115 :     else{
116 :         pt=str;
117 :         pt+=Newline(new,new->next,str,&x);
118 :         new=new->next;
119 :         if(y<MAXROW-1) y++;
120 :         else{
121 :             movetext(MINCOL,MINROW+1,
122 :                 MAXCOL,MAXROW,
123 :                 MINCOL,MINROW);
124 :             Cls(MINCOL,y+1,MAXCOL,y+1,
125 :                 NORM,SPACE);
126 :             Showlist(new->next,1,y+1);
127 :         }
128 :     }
129 :     for(x=MINCOL,pt=str;x<MAXCOL;pt++){
130 :         if(*pt==SPACE) x++;
131 :         else if(*pt==TAB) x+=TAB_SIZE;
132 :         else break;
133 :     }
134 :     if(x>=MAXCOL-MINCOL-1){
135 :         x=MINCOL; pt=str;
136 :     }
137 :     Line++;
138 : }
139 : else Memory_state();
140 : break;
141 : case TAB:if(x+TAB_SIZE<MAXCOL){
142 :     if(Insert){
143 :         pt1=str;
144 :         pt1+=strlen(str);
145 :         for(pt1-=2;pt1!=pt;pt1--){
146 :             *pt1=*(pt1-1);
147 :             movetext(x,y,MAXCOL-TAB_SIZE-1,y,
148 :                 x+TAB_SIZE,y);
149 :         }
150 :     }
151 :     else
152 :         movetext(x+1,y,MAXCOL-TAB_SIZE-1,y,
153 :             x+TAB_SIZE,y);
154 :     for(i=0;i<TAB_SIZE;i++,x++){
155 :         putchar(' ');
156 :         *pt+=TAB;
157 :     }
158 :     break;
159 : default:if(x<MAXCOL-1 && key!=ESC){
160 :     if(Insert){
161 :         pt1=str;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161:                                     pt1+=strlen(str);
162:                                     for(pt1-=2;pt1!=pt;pt1--)
163:                                         *pt1=*(pt1-1);
164:                                     movetext(x,y,MAXCOL-1,y,x+1,y);
165:                                     }
166:                                     else{
167:                                         if(*pt==TAB)
168:                                             movetext(x+TAB_SIZE,y,MAXCOL,
169:                                                 y,x+1,y);
170:                                     }
171:                                     *pt+=key;
172:                                     putchar(key);
173:                                     x++;
174:                                     }
175:                                     break;
176:                                     }
177:                                     }
178:                                     else{
179:                                         key=getch();
180:                                         switch(key){
181:                                             case UP :if(new->prev){
182:                                                 pt=str;
183:                                                 pt+=Newline(new,new->prev,str,&x);
184:                                                 new=new->prev;
185:                                                 if(y>MINROW) y--;
186:                                                 else{
187:                                                     movetext(MINCOL,MINROW,MAXCOL,
188:                                                         MAXROW-1,MINCOL,MINROW+1);
189:                                                     Cls(MINCOL,y,MAXCOL,y,NORM,SPACE);
190:                                                     Showlist(new,1,y);
191:                                                 }
192:                                                 Line--;
193:                                             }
194:                                             break;
195:                                             case DOWN:if(new->next){
196:                                                 pt=str;
197:                                                 pt+=Newline(new,new->next,str,&x);
198:                                                 new=new->next;
199:                                                 if(y<MAXROW-1) y++;
200:                                                 else {
201:                                                     movetext(MINCOL,MINROW+1,MAXCOL,
202:                                                         MAXROW,MINCOL,MINROW);
203:                                                     Cls(MINCOL,y+1,MAXCOL,y+1,NORM,SPACE);
204:                                                     if(new->next)
205:                                                         Showlist(new->next,1,y+1);
206:                                                 }
207:                                                 Line++;
208:                                             }
209:                                             break;
210:                                             case LEFT :if(x>MINCOL){
211:                                                 if(*(pt-1)==TAB) x-=TAB_SIZE;
212:                                                 else
213:                                                     x--;
214:                                                 pt--;
215:                                             }
216:                                             break;
217:                                             case RIGHT:if(x<MAXCOL){
218:                                                 if(*pt==TAB) x+=TAB_SIZE;
219:                                                 else
220:                                                     x++;
221:                                                 pt++;
222:                                             }
223:                                             break;
224:                                             case PgUp :if(new->prev){
225:                                                 ClipSpace(str);
226:                                                 strcpy(new->string,str);
227:                                                 df=MAXROW-2*MINROW+y;
228:                                                 for(i=0;i<=df;i++)
229:                                                     if(new->prev) new=new->prev;
230:                                                 else {
231:                                                     y=MINROW;
232:                                                     Line=MAXROW-MINROW+2;
233:                                                     break;
234:                                                 }
235:                                                 Showl_page(new);
236:                                                 for(i=0;i<y-MINROW;i++)
237:                                                     if(new->next) new=new->next;
238:                                                 strcpy(str,new->string);
239:                                                 pt=str;
240:                                                 pt+=Newline(new,new,str,&x);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
241:                                     Line-=MAXROW-MINROW+1;
242:                                     }
243:                                     break;
244: case PgDn: if(new->next &&
245:               (Phead->length)-Line>MAXROW-MINROW+1){
246:               ClipSpace(str);
247:               strcpy(new->string,str);
248:               df=MAXROW-y;
249:               for(i=0;i<=df && new->next;i++,Line++)
250:                 new=new->next;
251:               Show1_page(new);
252:               for(i=0;i<y-MINROW && new->next;i++,Line++)
253:                 new=new->next;
254:               strcpy(str,new->string);
255:               pt=str;
256:               pt+=Newline(new,new,str,&x);
257:               }
258:               break;
259: case DEL: if(*pt==TAB)
260:               movetext(x+TAB_SIZE,y,MAXCOL,y,x,y);
261:               else
262:                 movetext(x+1,y,MAXCOL,y,x,y);
263:               for(pt1=pt,i=strlen(pt)-1;i>0;pt1++,i--)
264:                 *pt1=*(pt1+1);
265:               Fastputs(MAXCOL,y,NORM," ");
266:               break;
267: case INS: Insert=(Insert==1)? 0:1;
268:               if(!Insert)
269:                 Fastputs(MINCOL+(MAXCOL-MINCOL)/2-9,
270:                           MINROW-1,0x70,"");
271:               else
272:                 Fastputs(MINCOL+(MAXCOL-MINCOL)/2-9,
273:                           MINROW-1,0x70,"Insert");
274:               break;
275: case HOME: x=MINCOL; pt=str;
276:               break;
277: case END :pt=str; x=MINCOL;
278:               while(x<MAXCOL){
279:                 if(*pt++==TAB) x+=TAB_SIZE;
280:                 else x++;
281:               }
282:               while(*--pt==SPACE) x--;
283:               pt++;
284:               break;
285: case F10: cursorShape(CURSOROFF);
286:               Fastgettext(MINCOL,MINROW-1,MINCOL+20,
287:                           MINROW+EDIT_CHOICE,txt);
288:               choice=Menu(MINCOL,MINROW-1,0x36,0x30,
289:                           0x0F,editch,EDIT_CHOICE,14,choice);
290:               switch(choice){
291: case 1: Fastputtext(MINCOL,MINROW-1,
292:                     MINCOL+20,MINROW+5,txt);
293:               Clearlist(Phead);
294:               Phead=create_head();
295:               memset(str,SPACE,MAXCOL);
296:               str[MAXCOL]='\0';
297:               Appendlist(Phead,str);
298:               pt=str;
299:               new=Phead->first;
300:               Show1_page(new);
301:               Fastgettext(MINCOL,MINROW-1,
302:                           MINCOL+20,MINROW+5,txt);
303:               Line=1; x=MINCOL;
304:               y=MINROW; pt=str;
305:               file_name[0]='\0';
306:               Cls(MAXCOL-15,MINROW-1,MAXCOL,
307:                   MINROW-1,0x17,SPACE);
308:               break;
309: case 2: ClipSpace(str);
310:               strcpy(new->string,str);
311:               strcpy(str1,Show_Get_message(MINCOL+5,
312:                                               MINROW+choice,MESG_ATR,
313:                                               "File name",1));
314:               Storelist(str1);
315:               Fastprintf(MAXCOL-15,MINROW-1,
316:                           0x17,"%s",str1);
317:               strcpy(file_name,str1);
318:               break;
319: case 3: strcpy(str,Show_Get_message(MINCOL+5,
320:                                       MINROW+choice,MESG_ATR,
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

321:         "File name",1));
322:     if(strchr(str,'*'))
323:         strcpy(str1,Directory(str));
324:     else{
325:         i=strlen(str);
326:         pt1=str; pt1+=i; pt1--;
327:         while(*pt1!=':' && *pt1!='\\' && i>0){
328:             pt1--; i--;
329:         }
330:         pt1++; strcpy(str1,pt1);
331:     }
332:     strcpy(file_name,str);
333:     if(str1[0]!='\0' && Loadlist(file_name)){
334:         Fastprintf(MAXCOL-15,MINROW-1,
335:             0x17,"%c:%s",str[0],str1);
336:         new=Phead->first;
337:         Fastputtext(MINCOL,MINROW-1,
338:             MINCOL+20,MINROW+5,txt);
339:         Show1_page(new);
340:         strcpy(str,new->string);
341:         Line=1; x=MINCOL;
342:         y=MINROW; pt=str;
343:         Fastgettext(MINCOL,MINROW-1,
344:             MINCOL+20,MINROW+EDIT_CHOICE,txt);
345:     }
346:     break;
347: case 4: ClipSpace(str);
348:         strcpy(new->string,str);
349:         CompileProgram();
350:         break;
351: case 5: ClipSpace(str);
352:         strcpy(new->string,str);
353:
354:         CompileProgram();
355:         textattr(tf.attribute);
356:         window(tf.winleft,tf.wintop,tf.winright,
357:             tf.winbottom);
358:         RunList(file_name);
359:
360:         cursorShape(CURSORON);
361:         Print_menu(1);
362:         Title();
363:         window(1,1,80,25);
364:         gotoxy(MINCOL,MINROW);
365:
366:         Loadlist(file_name);
367:         Fastprintf(MAXCOL-15,MINROW-1,
368:             0x17,"%c:%s",str[0],str1);
369:         new=Phead->first;
370:         Fastputtext(MINCOL,MINROW-1,
371:             MINCOL+20,MINROW+5,txt);
372:         Show1_page(new);
373:         strcpy(str,new->string);
374:         Line=1; x=MINCOL;
375:         y=MINROW; pt=str;
376:         Fastgettext(MINCOL,MINROW-1,
377:             MINCOL+20,MINROW+EDIT_CHOICE,txt);
378:         textattr(NORM);
379:     }
380:     Fastputtext(MINCOL,MINROW-1,MINCOL+20,
381:         MINROW+EDIT_CHOICE,txt);
382:     if(choice==EDIT_CHOICE){
383:         Clearlist(Phead);
384:         break;
385:     }
386: }
387: }
388: cursorShape(CURSORON);
389: if(choice==EDIT_CHOICE) break;
390: choice=abs(choice);
391: }
392: }
393: }
394:
395: void Title(void)
396: #define ATKEY 0x74
397: #define BORD 0x12
398: {
399:     makewin(MINCOL-3,MINROW-2,MAXCOL+3,MAXROW+1,BORD,NORM,0);
400:     Fastputs(MINCOL+(MAXCOL-MINCOL)/2-4,MINROW-2,0x20," EDIT ");

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
401: Fastputs(MINCOL,MINROW-2,0x1D,"F10");
402: Fastputs(MINCOL+3,MINROW-2,0x17,"-File[\xF0]");
403: Setattr(MINCOL,MINROW-1,MAXCOL-MINCOL+1,0x12);
404: textattr(NORM);
405: }
406:
407: void Show1_page(DNODE far *new)
408: {
409:   Cls(MINCOL,MINROW,MAXCOL,MAXROW,NORM,SPACE);
410:   Showlist(new,MAXROW-MINROW+1,MINROW);
411: }
412:
413: int Newline(DNODE far *cur,DNODE far *new,char *str,int *x)
414: {
415:   char *pt;
416:   register int newx=MINCOL,step=0;
417:
418:   ClipSpace(str);
419:   strcpy(cur->string,str);
420:   cur=new;
421:   strcpy(str,cur->string);
422:   AddSpace(str);
423:   pt=str;
424:   while(newx<*x){
425:     if(*pt++==TAB) newx+=TAB_SIZE;
426:     else newx++;
427:     step++;
428:   }
429:   *x=newx;
430:   return(step);
431: }
432:
433: void ClipSpace(char *str)
434: {
435:   char *pt;
436:
437:   pt=str;
438:   pt+=strlen(str); pt--;
439:   while(isspace(*pt)) pt--;
440:   while(isspace(*pt)) pt--;
441:   *++pt='\0';
442: }
443:
444: void AddSpace(char *str)
445: {
446:   int x,i;
447:   char *pt;
448:
449:   pt=str;
450:   x=strlen(str);
451:   pt+=x;
452:   for(i=MAXCOL+MINCOL+10-x;i>0;i--)
453:     *pt++=' ';
454:   *--pt='\0';
455: }
456:
457: }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : typedef struct dnode{
2 :     char string[MAXCOL+MINCOL+10];
3 :     struct dnode far *prev,*next;
4 : }DNODE;
5 :
6 : typedef struct dhead{
7 :     int length;
8 :     DNODE far *first,*last;
9 : }HEAD;
10 :
11 : char *Getcommand(int Line);
12 : char *Replace_Var(char *st);
13 : int Search_Replace_Var(char *data);
14 : void SaveVar(DNODE far *begin);
15 : int Check_program(void);
16 : char *SkipSpace(char *data);
17 :
18 : HEAD far *create_head(void);
19 : DNODE far *Getdata(char *data,DNODE far *prev,DNODE far *next);
20 : DNODE far *Insertlist(HEAD far *head,DNODE far *node);
21 : DNODE far *Dellist(HEAD far *head,DNODE far *node);
22 :
23 : int Appendlist(HEAD far *head,char *data);
24 : void Storelist(char *fname);
25 : int Loadlist(char *fname);
26 : void Clearlist(HEAD far *head);
27 : void Memory_state(void);
28 : void Showlist(DNODE far *node,int numlist,int y);
29 :
30 : extern HEAD far *Phead,*Vhead;
31 : extern int MEMFULL;
32 :
33 : char *Getcommand(int Line)
34 : {
35 :     DNODE far *var,*node;
36 :     register int i;
37 :     char *pt,data[MAXCOL+MINCOL+10],*keyw[]={ "DEFINE","RETURN",
38 :                                                "ENDPROGRAM"};
39 :
40 :     node=Phead->first;
41 :     node=node->next;
42 :     if(node){
43 :         for(i=0;i<Line;i++){
44 :             strcpy(data,node->string);
45 :             strupr(data);
46 :             pt=SkipSpace(data);
47 :             while(!strncmp(pt,keyw[0],strlen(keyw[0]))){
48 :                 *pt=='\0' || *pt=='#'{
49 :                     node=node->next; /* skip across keyword */
50 :                     strcpy(data,node->string);
51 :                     strupr(data);
52 :                     pt=SkipSpace(pt);
53 :                 }
54 :                 node=node->next;
55 :             }
56 :             node=node->prev;
57 :
58 :             /*----- if "ENDPROCESS" set new Variable area -----*/
59 :             if(!strncmp(data,keyw[1],strlen(keyw[1]))){
60 :                 node=node->next;
61 :                 SaveVar(node);
62 :             }
63 :             if(strcmp(data,keyw[2]))
64 :                 memcpy(data,node->string,strlen(node->string));
65 :             pt=SkipSpace(data);
66 :             strcpy(data,pt);
67 :             strcpy(data,Replace_Var(data));
68 :             return(data);
69 :         }
70 :         return(NULL);
71 :     }
72 :
73 : char *Replace_Var(char *st)
74 : {
75 :     DNODE far *var;
76 :     char str[MAXCOL+MINCOL+10],data[MAXCOL+MINCOL+10],
77 :         dat[MAXCOL+MINCOL+10],*pt,*pt1,*pt2,*pt3;
78 :     register int ENDLINE=0;
79 :
80 :     strcpy(str,st);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :   pt=strchr(str,EQ);
82 :   if(pt){
83 :       while(isspace(++pt));
84 :       var=Vhead->first;
85 :       while(var->string){
86 :           /* while(*pt!='\0'){*/
87 :               strcpy(data,var->string);
88 :               strcpy(dat,data);
89 :               pt1=strchr(data,EQ);
90 :               while(isspace(*--pt1)); /* data store variable name */
91 :               *++pt1='\0';
92 :               pt2=strchr(dat,EQ);
93 :               while(isspace(++pt2)); /* pt2 store variable value */
94 :               while((pt3=strstr(pt,data))!=NULL){
95 :                   *pt3='\0';
96 :                   pt3+=strlen(data);
97 :                   if(*pt3=='\0')
98 :                       strcat(pt,pt2);
99 :                   else
100 :                       sprintf(pt,"%s%s%s",pt,pt2,pt3);
101 :               }
102 :
103 : /*
104 :   if(pt1){ /*-- if variable match --
105 :       strcpy(dat,var->string);
106 :       pt2=strchr(dat,EQ);
107 :       while(isspace(++pt2));
108 :       pt+=strlen(pt2)-1;
109 :       strcpy(buf,pt);
110 :       if(pt=strstr(pt1,data)!=NULL){
111 :           pt+=strlen(data);
112 :           *pt='\0';
113 :           sprintf();
114 :       }
115 :       if(*pt=='\0') ENLINE=1;
116 :       if(strlen(str)+strlen(pt2)<MAXCOL+MINCOL+10)
117 :           strcat(str,pt2);
118 :       if(!ENLINE)
119 :           if(strlen(str)+strlen(pt)<MAXCOL+MINCOL+10)
120 :               strcat(str,buf);
121 :       ENLINE=0;
122 :   }
123 :   else{
124 :       pt=str;
125 :       pt+=strlen(str);
126 :   }
127 :   pt=strchr(str,EQ);
128 :   while(isspace(++pt)); */
129 :   var=var->next;
130 : }
131 : }
132 : return(str);
133 : }
134 :
135 : int Search_Replace_Var(char *data)
136 : {
137 :     DNODE far *var;
138 :     char str[MAXCOL+MINCOL+10],dat[MAXCOL+MINCOL+10],*pt;
139 :
140 :     strcpy(str,data);
141 :     pt=strchr(str,EQ);
142 :     while(isspace(*--pt));
143 :     *++pt='\0';
144 :     var=Vhead->first;
145 :     while(var){
146 :         strcpy(dat,var->string);
147 :         if(!strcmp(dat,str,strlen(str))){
148 :             strcpy(var->string,data);
149 :             return(TRUE);
150 :         }
151 :         var=var->next;
152 :     }
153 :     return(FALSE);
154 : }
155 :
156 : void SaveVar(DNODE far *begin)
157 : {
158 :     char data[MAXCOL+MINCOL+10],*pt,*keyw[]="RETURN", "DEFINE";
159 :
160 :     while(begin){

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161:         strcpy(data,begin->string);
162:         pt=SkipSpace(data);
163:         strupr(pt);
164:         if(!strcmp(pt,keyword[0],strlen(keyword[0])))
165:             break;
166:         if(!strcmp(pt,keyword[1],strlen(keyword[1]))){
167:             strcpy(data,begin->string);
168:             pt=SkipSpace(data);
169:             pt=strchr(pt,SPACE);
170:             pt=SkipSpace(pt);
171:             while(isspace(*pt)) pt++;
172:             if(!Search_Replace_Var(pt)){
173:                 Appendlist(Vhead,Replace_Var(pt));
174:             }
175:         }
176:         begin=begin->next;
177:     }
178: }
179:
180: int Check_program(void)
181: {
182:     DNODE far *node;
183:     char *pt,data[MAXCOL+MINCOL+10];
184:     char *keyword={"SIMULATE","ENDPROGRAM"};
185:
186:     node=Phead->first;
187:     if(!node){
188:         Show_Get_message(15,15,0x4E,
189:             "Not found program...press any key",0);
190:         return(1); /* not found program */
191:     }
192:     strcpy(data,node->string);
193:     strupr(data);
194:     pt=SkipSpace(data);
195:
196:     if(strcmp(pt,keyword[0],strlen(keyword[0]))!=0){
197:         Show_Get_message(15,15,0x4E,
198:             "Not found \"simulate\"...press any key",0);
199:         return(2); /* not found "SIMULATE" */
200:     }
201:     node=node->next;
202:
203:     Vhead=create_head(); /* Create variable area */
204:     SaveVar(node);
205:
206:     while(node){
207:         strcpy(data,node->string);
208:         strupr(data);
209:         pt=SkipSpace(data);
210:         if(!strcmp(pt,keyword[1],strlen(keyword[1]))){
211:             return(0);
212:             node=node->next;
213:         }
214:         Show_Get_message(15,15,0x4E,
215:             "Not found \"enprogram\"...press any key",0);
216:         return(3); /* not found "ENDPROGRAM" */
217:     }
218:
219: char *SkipSpace(char *data)
220: {
221:     char *pt;
222:
223:     if(pt=strstr(data,"//")!=NULL) *pt='\0';
224:     if((pt=strchr(data,':'))!=NULL) pt++;
225:     else pt=data;
226:     while(isspace(*pt)) pt++;
227:     return(pt);
228: }
229: /*----- Link list -----*/
230: HEAD far *create_head()
231: {
232:     HEAD far *head;
233:
234:     head=farmalloc(sizeof(HEAD));
235:     if(head){
236:         head->length=0;
237:         head->first=head->last=NULL;
238:     }
239:     return(head);
240: }

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
241:
242: DNODE far *Getdata(char *data,DNODE far *prev,DNODE far *next)
243: {
244:     DNODE far *new=NULL;
245:
246:     new=(DNODE *)farmalloc(sizeof(DNODE));
247:     if(new){
248:         strcpy(new->string,data);
249:         new->prev=prev;
250:         new->next=next;
251:         return(new);
252:     }
253:     else{
254:         Memory_state();
255:         return(NULL);
256:     }
257: }
258:
259: DNODE far *Insertlist(HEAD far *head,DNODE far *node)
260: {
261:     DNODE far *new;
262:     char str[MAXCOL+MINCOL+10],*pt;
263:     int newx=MINCOL;
264:
265:     pt=node->string;
266:     while(newx<wherex()){
267:         if(*pt++==TAB) newx+=TAB_SIZE;
268:         else newx++;
269:     }
270:     strcpy(str,pt);
271:     *pt='\0';
272:     if(node->next){
273:         new=Getdata(str,node,node->next);
274:         if(new){
275:             node->next=node->next->prev=new;
276:             head->length++;
277:         }
278:     }
279:     else{
280:         Appendlist(head,str);
281:         new=head->last;
282:     }
283:     return(new);
284: }
285:
286: DNODE far *Dellist(HEAD far *head,DNODE far *node)
287: {
288:     DNODE far *new;
289:     char str[MAXCOL+MINCOL+10];
290:
291:     if(node->next){
292:         if(!node->prev){
293:             new=node->next;
294:             head->first=new;
295:             new->prev=NULL;
296:         }
297:         else{
298:             new=node->next;
299:             node->prev->next=new;
300:             new->prev=node->prev;
301:         }
302:         head->length--;
303:         far free(node);
304:         if(MEMFULL) MEMFULL=0;
305:     }
306:     else{
307:         strcpy(node->string,"");
308:         new=node;
309:     }
310:     return(new);
311: }
312:
313: int Appendlist(HEAD far *head,char *data)
314: {
315:     DNODE far *new;
316:
317:     new=Getdata(data,head->last,NULL);
318:     if(new){
319:         if(head->length)
320:             head->last->next=new;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
321:         else head->first=new;
322:         head->last=new;
323:         head->length++;
324:         return(TRUE);
325:     }
326:     else return(FALSE);
327: }
328:
329: void Storelist(char *fname)
330: {
331:     FILE *f;
332:     DNODE far *this;
333:
334:     if(!strcmp(fname,"")) return;
335:     if(Phead!=NULL){
336:         if((f=fopen(fname,"w"))!=NULL){
337:             this=Phead->first;
338:             do{
339:                 fputs(this->string,f);
340:                 fputc('\n',f);
341:                 this=this->next;
342:             }while(this!=NULL);
343:             fclose(f);
344:         }
345:         else
346:             Show_Get_message(15,15,0x4E,"Cannot open...press anykey",0);
347:     }
348: }
349:
350: int Loadlist(char *fname)
351: {
352:     FILE *f;
353:     DNODE far *new;
354:     char data[MAXCOL+MINCOL+10],*pt;
355:     int i;
356:
357:     strupr(fname);
358:     if(!strcmp(fname,"")) return(FALSE);
359:     if((f=fopen(fname,"r"))!=NULL){
360:         Clearlist(Phead);
361:         Clearlist(Vhead);
362:         Phead=create_head();
363:         while(!feof(f)){
364:             pt=data;
365:             i=MINCOL;
366:             do{
367:                 *pt=fgetc(f);
368:                 i++;
369:             }while(*pt++!='\n' && !feof(f) && i<MAXCOL+MINCOL+8);
370:             *--pt='\0';
371:             if(!Appendlist(Phead,data)) break;
372:         }
373:         fclose(f);
374:         return(TRUE);
375:     }
376:     else{
377:         Show_Get_message(15,15,0x4E,"Cannot open...press anykey",0);
378:         return(FALSE);
379:     }
380: }
381:
382: void Clearlist(HEAD far *head)
383: {
384:     DNODE far *del;
385:
386:     if(head && head->length){
387:         while(head->length){
388:             del=head->first;
389:             head->first=del->next;
390:             head->length--;
391:             farfree(del);
392:         }
393:         head->first=head->last=NULL;
394:         far free(head);
395:         head=NULL;
396:     }
397: }
398:
399: void Memory_state(void)
400: {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
401:  sound(800); delay(100); nosound();
402:  Show_Get_message(10,5,0x4F,"Not enough memory...",0);
403:  MEMFULL=1;
404: }
405:
406: void Showlist(DNODE far *node,int numlist,int y)
407: {
408:  char *pt,dat[MAXCOL+MINCOL+10];
409:  register int x;
410:
411:  if(node!=NULL){
412:      do{
413:          strcpy(dat,node->string);
414:          pt=dat;
415:          x=MINCOL;
416:          while(*pt!='\0'){
417:              if(*pt==TAB){
418:                  if(x+TAB_SIZE<MAXCOL){
419:                      x+=TAB_SIZE;
420:                      pt++;
421:                  }
422:                  else break;
423:              }
424:              else{
425:                  if(x<MAXCOL)
426:                      Fastprintf(x++,y,NORM,
427:                                  "%c",*pt++);
428:                  else break;
429:              }
430:          }
431:          y++;
432:          node=node->next;
433:      }while(node!=NULL && --numlist);
434:  }
435: }
436:
437: ↑
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : #define IGNORE 0
2 : #define RETRY 1
3 : #define ABORT 2
4 : #define CURSOROFF 0x2000 /* Cursor OFF */
5 : #define CURSORON 0x0607 /* Cursor ON */
6 :
7 : char *Show_Get_message(int x1,int y1,int attr,char *mes,int ans);
8 : void getstr(char *str,int length,int attr);
9 : int Menu(int x,int y,int borattr,int fillattr,int barattr,
10 : char *str,int numline,int maxlength,int _choice);
11 : void makewin(int x1,int y1,int x2,int y2,int bordattr,
12 : int fillattr,int style);
13 : void Cls(int x1,int y1,int x2,int y2,int attr,char ch);
14 : void Setattr(int x,int y,int length,int attr);
15 : void Fastprintf(int x,int y,int attr,char *msg, ...);
16 : void Fastputs(int x,int y,int attr,char *str);
17 : void Fastgettext(int x1,int y1,int x2,int y2,int *store);
18 : void Fastputtext(int x1,int y1,int x2,int y2,int *store);
19 : void cursorShape(unsigned int status);
20 : int handler(int errval,int ax,int bp,int si);
21 : int error_win(char *msg);
22 : void install_handler(void);
23 : int Cprint(char *msg,...);
24 :
25 : extern int DIRECT,CR_P;
26 :
27 : char *Show_Get_message(int x1,int y1,int attr,char *mes,int ans)
28 : #define LEN 48
29 : {
30 :     int tx[800],x2,y2;
31 :     char *answer,an[80],ch;
32 :     struct text_info tf;
33 :
34 :     if(ans==2){
35 :         y2=y1+8; x2=x1+LEN-12;
36 :     }
37 :     else{
38 :         y2=y1+2; x2=x1+LEN;
39 :     }
40 :     if(ans==1) cursorShape(CURSORON);
41 :     else cursorShape(CURSOROFF);
42 :     gettextinfo(&tf);
43 :     Fastgettext(x1,y1,x2,y2,tx);
44 :     textattr(attr);
45 :     if(ans==2)
46 :         makewin(x1,y1,x2,y2,0x7F,0x7F,1);
47 :     else
48 :         makewin(x1,y1,x2,y2,attr,attr,0);
49 :     window(1,1,80,25);
50 :     switch(ans){
51 :     case 0:window(x1+(LEN-strlen(mes))/2,y1+1,x2-2,y1+1);
52 :             cprintf("%s",mes);
53 :             getch(); /* Not echo */
54 :             break;
55 :     case 1: sprintf(an,"[ %s ]",mes);
56 :             gotoxy(x1+(LEN-strlen(an))/2,y1);
57 :             cprintf("%s",an);
58 :             gotoxy(x1+3,y1+1); /* Echo string */
59 :             getstr(an,LEN-5,attr);
60 :             break;
61 :     case 2: Fastputs(x1+(x2-x1-8)/2,y1,0x7E," Message ");
62 :             Fastputs(x1+(x2-x1)/5,y2-(y2-y1)/3,0x20,
63 :                     " Yes ");
64 :             Fastputs(x2-(x2-x1)/4-7,y2-(y2-y1)/3,0x20,
65 :                     " No ");
66 :             Fastputs(x1+(x2-x1)/5+10,y2-(y2-y1)/3,0x70,"\\xDC");
67 :             Fastputs(x2-(x2-x1)/4+3,y2-(y2-y1)/3,0x70,"\\xDC");
68 :             Fastputs(x1+(x2-x1)/5+1,y2-(y2-y1)/3+1,0x70,
69 :                     "#####");
70 :             Fastputs(x2-(x2-x1)/4-6,y2-(y2-y1)/3+1,0x70,
71 :                     "#####");
72 :             Setattr(x1+(x2-x1)/5+4,y2-(y2-y1)/3,1,0x2E);
73 :             Setattr(x2-(x2-x1)/4-3,y2-(y2-y1)/3,1,0x2E);
74 :             Cls(x1+3,y1+1,x2-3,y2-2*(y2-y1)/3,attr,SPACE);
75 :             window(x1+3,y1+1,x2-3,y2-2*(y2-y1)/3);
76 :             cputs(mes);
77 :             do{
78 :                 ch=toupper(getch());
79 :             }while(ch!='Y' && ch!='N');
80 :             sprintf(an,"%c",ch);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :         break;
82 :     }
83 :     Fastputtext(x1,y1,x2,y2,tx);
84 :     textattr(tf.attribute);
85 :     window(tf.winleft,tf.wintop,tf.winright,tf.winbottom);
86 :     gotoxy(tf.curx,tf.cury);
87 :     cursorShape(CURSROFF);
88 :
89 :     answer=an;
90 :     if(ans)
91 :         return(answer);
92 :     else
93 :         return(NULL);
94 : }
95 : void getstr(char *str,int length,int attr)
96 : {
97 :     char ch,*pt,*pt1;
98 :     int count=0,x,y;
99 :     struct text_info tx;
100 :
101 :     gettextinfo(&tx);
102 :     x=tx.curx;
103 :     y=tx.cury;
104 :     textattr(attr);
105 :     memset(str,'\0',MAXCOL);
106 :     pt=str;
107 :     for(;;){
108 :         ch=getch();
109 :         if(ch!=0){
110 :             if(ch==ESC){
111 :                 *str='\0';         break;
112 :             }
113 :             if(ch==CR) break;
114 :             if(ch==BSP){
115 :                 if(pt>str){
116 :                     if(count>length-1){
117 :                         movetext(tx.curx,y,x-2,y,tx.curx+1,y);
118 :                         gotoxy(tx.curx,y);
119 :                         putch(*(pt-x+tx.curx-1));
120 :                         if(count==MAXCOL){
121 :                             gotoxy(tx.curx-1,y); putch(' ');
122 :                             count--;
123 :                         }
124 :                         gotoxy(x,y);
125 :                     }
126 :                     else{
127 :                         x--;
128 :                         movetext(x+1,y,tx.curx+length-1,y,x,y);
129 :                         gotoxy(x,y);
130 :                     }
131 :                 }
132 :                 pt--; pt1=pt;
133 :                 while(*pt1!='\0'){
134 :                     *pt1=*(pt1+1);
135 :                     pt1++;
136 :                 }
137 :                 count--;
138 :             }
139 :         }
140 :         else if(count<MAXCOL-1){
141 :             *pt+=ch;
142 :             count++;
143 :             if(x>=tx.curx+length-1){
144 :                 movetext(tx.curx+1,y,x,y,tx.curx,y);
145 :                 gotoxy(x-1,y); putch(ch);
146 :             }
147 :             else{
148 :                 gotoxy(x,y); putch(ch);
149 :                 x++;
150 :             }
151 :         }
152 :         if(count==MAXCOL-1){
153 :             gotoxy(tx.curx-1,y); putch('\x11');
154 :             gotoxy(x+1,y);
155 :             count++;
156 :         }
157 :     }
158 :     else{
159 :         ch=getch();
160 :         switch(ch){

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161:         case LEFT:if(pt>str){
162:             pt--;
163:             if(x>tx.curx){
164:                 gotoxy(--x,y);
165:             }
166:             else{
167:                 movetext(x,y,x+length-2,y,x+1,y);
168:                 gotoxy(x-1,y);
169:                 cprintf("\x11%c\b",*pt);
170:             }
171:             if(pt==str){
172:                 gotoxy(x-1,y); putchar(' ');
173:             }
174:         }
175:         break;
176:         case RIGHT:if(*pt!='\0'){
177:             pt++;
178:             if(x<tx.curx+length-1){
179:                 gotoxy(++x,y);
180:             }
181:             else{
182:                 movetext(tx.curx+1,y,x,y,tx.curx,y);
183:                 gotoxy(x,y); cprintf("%c\x10",*pt);
184:             }
185:             if(*pt=='\0'){
186:                 gotoxy(x+1,y); putchar(' ');
187:             }
188:         }
189:         break;
190:     }
191: }
192: textattr(tx.attribute);
193: }
194: }
195:
196: void Window_message(int x1,int y1,int borattr,int fillattr,
197: char *str,int numline,int maxlength)
198: {
199:     register int i;
200:     int text[1000],x2,y2;
201:
202:     x2=x1+maxlength+5;
203:     y2=y1+numline+1;
204:     Fastgettext(x1,y1,x2,y2+1,text);
205:     makewin(x1,y1,x2,y2+1,borattr,fillattr,1);
206:     for(i=0;i<numline;i++,str+=maxlength)
207:         Fastputs(x1+1,y1+i+1,fillattr,str);
208:     Fastputs(x1+1,y1+i+1,fillattr,"Press any key...");
209:     getch();
210:     Fastputtext(x1,y1,x2,y2+1,text);
211: }
212:
213: int Menu(int x,int y,int borattr,int fillattr,int barattr,
214: char *str,int numline,int maxlength,int _choice)
215: {
216:     register int i,x1,y1;
217:     int x2,y2;
218:     char key;
219:
220:     x1=x; y1=y;
221:     x2=x1+maxlength+5;
222:     y2=y1+numline+1;
223:     makewin(x1,y1,x2,y2,borattr,fillattr,0);
224:     for(i=0;i<numline;i++,str+=maxlength)
225:         Fastputs(x1+3,y1+i+1,fillattr,str);
226:     do{
227:         Setattr(x1+2,y1+_choice,maxlength+2,barattr);
228:         key=getch();
229:         if(key==CR) return(_choice);
230:         Setattr(x1+2,y1+_choice,maxlength+2,fillattr);
231:         if(key==0){
232:             key=getch();
233:             switch(key){
234:                 case UP:if(_choice>1) _choice--;
235:                     else _choice=numline;
236:                     break;
237:                 case DOWN:if(_choice<numline) _choice++;
238:                     else _choice=1;
239:                     break;
240:                 case LEFT:DIRECT=-1; key=ESC;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

241:                                     break;
242:                                     case RIGHT:DIRECT=1; key=ESC;
243:                                     }
244:                                     }
245:                                     if(key==ESC){
246:                                         _choice--2*_choice;
247:                                         if(DIRECT!=0) break;
248:                                         CR_P=0;
249:                                     }
250:                                     }while(key!=ESC && key!=CR);
251:                                     return(_choice);
252: }
253:
254: void makewin(int x1,int y1,int x2,int y2,int bordattr,
255: int fillattr,int style)
256: {
257:     register int i,j,hori = x2-x1+1;
258:     char upper[81];
259:     char bord[][6] = {
260:         {'\xC4','\xB3','\xDA','\xBF','\xD9','\xC0'},
261:         {'\xCD','\xBA','\xC9','\xBB','\xBC','\xC8'}
262:     };
263:     Cls(x1,y1,x2,y2,fillattr,SPACE);
264:     memset(upper,NULL,sizeof(upper));
265:     memset(upper,bord[style][0],hori);
266:     Fastputs(x1,y1,bordattr,upper);
267:     Fastputs(x1,y2,bordattr,upper);
268:     j = y2-y1;
269:     for (i = 1; i < j; i++) {
270:         Fastprintf(x1,y1+i,bordattr," %c",bord[style][1]);
271:         Fastprintf(x2-1,y1+i,bordattr,"%c ",bord[style][1]);
272:         Setattr(x2+1,y1+i,1,0x08);
273:     }
274:     Setattr(x2+1,y1+i,1,0x08);
275:     Setattr(x1+1,y2+1,hori,0x08);
276:     Fastprintf(x1,y1,bordattr," %c",bord[style][2]);
277:     Fastprintf(x2-1,y1,bordattr,"%c ",bord[style][3]);
278:     Fastprintf(x2-1,y2,bordattr,"%c ",bord[style][4]);
279:     Fastprintf(x1,y2,bordattr," %c",bord[style][5]);
280: }
281:
282: void Cls(int x1,int y1,int x2,int y2,int attr,char ch)
283: {
284:     register int i;
285:     char buf[81];
286:
287:     memset(buf,'\0',81);
288:     memset(buf,ch,x2-x1+1);
289:     for(i=y1;i<=y2;i++)
290:         Fastputs(x1,i,attr,buf);
291: }
292:
293: void Setattr(int x,int y,int length,int attr)
294: {
295:     register int Addr,i,pos=2*(x-1)+160*(y-1)+1;
296:
297:     _AH=15;
298:     geninterrupt(0x10);
299:     Addr=(_AL=='\x3')?0xB800:0xB000;
300:     for(i=0;i<length;i++,pos+=2)
301:         pokeb(Addr,pos,(char)attr);
302: }
303:
304: void Fastputs(int x,int y,int attr,char *str)
305: {
306:     register int Addr,pos=2*(x-1)+160*(y-1);
307:
308:     _AH=15;
309:     geninterrupt(0x10);
310:     Addr=(_AL=='\x3')?0xB800:0xB000;
311:
312:     while(*str!='\0'){
313:         pokeb(Addr,pos++,*str++);
314:         pokeb(Addr,pos++,(char)attr);
315:     }
316: }
317:
318: void Fastprintf(int x,int y,int attr,char *msg, ...)
319: {
320:     char str[256];

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
321: va_list ap;
322:
323: va_start(ap,msg);
324: vsprintf(str,msg,ap);
325: va_end(ap);
326: Fastputs(x,y,attr,str);
327: }
328:
329: void Fastgettext(int x1,int y1,int x2,int y2,int *store)
330: {
331:     register int Addr,i,j,pos=2*(x1-2)+160*(y1-1);
332:
333:     _AH=15;
334:     geninterrupt(0x10);
335:     Addr=( _AL=='\x3')?0xB800:0xB000;
336:     for(j=y1-1;j<y2+1;j++,pos=2*(x1-2)+160*j)
337:         for(i=x1-1;i<x2+1;i++)
338:             *store++=peek(Addr,pos+=2);
339: }
340:
341: void Fastputtext(int x1,int y1,int x2,int y2,int *store)
342: {
343:     register int Addr,i,j,pos=2*(x1-2)+160*(y1-1);
344:
345:     _AH=15;
346:     geninterrupt(0x10);
347:     Addr=( _AL=='\x3')?0xB800:0xB000;
348:     for(j=y1-1;j<y2+1;j++,pos=2*(x1-2)+160*j)
349:         for(i=x1-1;i<x2+1;i++)
350:             poke(Addr,pos+=2,*store++);
351: }
352:
353: void cursorShape(unsigned int status)
354: {
355:     _AH = 1;
356:     _CX = status;
357:     geninterrupt(0x10);
358: }
359: /*
360: This function will trap disk errors and prompt the user for action.
361: */
362:
363: /* define the error messages for trapping disk problems */
364: static char *err_msg[] = { "write protect","unknown unit","drive not ready",
365:     "unknown command","data error (CRC)","bad request","seek error",
366:     "unknown media type","sector not found","printer out of paper",
367:     "write fault","read fault","general failure","reserved",
368:     "reserved","invalid disk change"
369: };
370:
371: int error_win(char *msg)
372: {
373:     int retval,x,y,attr;
374:
375:     x=25; y=16; attr=0x4E;
376:     makewin(x-4,y-1,x+40,y+2,attr,attr,1);
377:     while(*msg!='\0'){
378:         if(*msg!='\n') Fastprintf(x++,y,attr,"%c",*msg++);
379:         else{
380:             msg++; x=25; y++; attr++;
381:         }
382:     }
383:     gotoxy(x,y);
384:
385:     /*prompt for user to press a key to abort, retry, ignore*/
386:     while(1)
387:     {
388:         retval= tolower(getch());
389:         if (retval == 'r' )
390:         {
391:             retval = RETRY;
392:             break;
393:         }
394:         if (retval == 'i' )
395:         {
396:             retval = IGNORE;
397:             break;
398:         }
399:     }
400: }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
401:    return(retval);
402: }
403:
404: /*
405: pragma warn -par~ reduces warnings which occur
406: due to the non use of the parameters errval,
407: bp and si to the handler.
408: */
409: #pragma warn -par
410:
411: int handler(int errval,int ax,int bp,int si)
412: {
413:     static char msg[80];
414:     unsigned di;
415:     int drive;
416:     int errorno;
417:
418:     di= _DI;
419: /*
420: if this is not a disk error then it was
421: another device having trouble
422: */
423:
424:     if (ax < 0)
425:     {
426:         /* report the error */
427:         error_win("Device error");
428:         /* and return to the program directly requesting abort */
429:         hardretn(ABORT);
430:     }
431: /* otherwise it was a disk error */
432:     drive =,ax & 0x00FF;
433:     errorno = di & 0x00FF;
434: /* report which error it was */
435:     sprintf(msg, "Error: %s on drive %c\nRetry, Ignore: ",
436:         err_msg[errorno], 'A' + drive);
437: /*
438: return to the program via dos interrupt 0x23 with abort, retry,
439: or ignore as input by the user.
440: */
441:
442:     hardresume(error_win(msg));
443:     return ABORT;
444: }
445: #pragma warn +par
446:
447: void install_handler(void)
448: {
449: /* install our handler on the hardware problem interrupt */
450:     harderr(handler);
451: }
452:
453: int Cprint(char *msg,...)
454: {
455:     char str[256];
456:     va_list ap;
457:
458:     va_start(ap,msg);
459:     vsprintf(str,msg,ap);
460:     va_end(ap);
461:     cprintf("%s",str);
462:     clreol();
463: }†
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
1 : void Print_menu(int choice);
2 :
3 : void Print_menu(int choice)
4 : #define MAXC 8
5 : {
6 :     register int i;
7 :     char *str[]={ "Edit", "Option", "Run", "Plot", "Show",
8 :                  "Math", "Help", "Quit" };
9 :
10 :    clrscr();
11 :    Cls(1,2,80,25,CLS_ATR,CLS_CH);
12 :    for(i=0;i<MAXC;i++)
13 :        Fastputs(1+BLOCK*i+(BLOCK-strlen(str[i]))/2,1,NORM,str[i]);
14 :    Setattr(1,1,80,M_BKG);
15 :    Setattr((choice-1)*BLOCK+1,1,BLOCK,M_BAR);
16 :    makewin(WINL-2,WINT-1,WINR+2,WINB+1,0x7F,0x31,0);
17 :    Fastputs(WINL+(WINR-WINL-7)/2,WINT-1,0x70,"Message");
18 :    Setattr(1,25,80,M_BKG);
19 :    window(WINL,WINT,WINR,WINB);
20 :    textattr(0x31);
21 : }
22 : †
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

1 : #define MINR 8
2 : #define MAXR 13
3 : #define MINC 13
4 : #define FLEN 14
5 : #define MAXF 3
6 : #define MAXCO 67-FLEN*(4-MAXF)
7 :
8 : char *Directory(char *curpath);
9 : void Searchfile(char *curpath,char *findf,char *fname1);
10 : void Getpath(char *curpath,char *fname);
11 : void Print_one_line(char far *fp,int y,int h,int result);
12 :
13 : char *Directory(char *curpath)
14 : {
15 :     char far *oldDTA;
16 :     char *oldpath,fname[FLEN],fname1[FLEN],*pt;
17 :     int i,scrn[700];
18 :
19 :     pt=curpath;
20 :     if(*pt=='\0') return(NULL);
21 :     i=strlen(curpath);
22 :     pt+=i;
23 :     while(*pt!='\\' && *pt!=':' && i--!=1)
24 :         pt--;
25 :
26 :     strcpy(fname,pt); *pt='\0';
27 :
28 :     Fastgettext(MINC-5,MINR-1,MAXCO+5,MAXR+3,scrn);
29 :     oldDTA=getdta();
30 :     oldpath=malloc(MAXDIR);
31 :     getcurdir(0,oldpath);
32 :     i=getdisk();
33 :
34 :     for(;;){
35 :         Getpath(curpath,fname);
36 :         Searchfile(curpath,fname,fname1);
37 :         strcpy(curpath,"");
38 :         if(fname1[0]=='\\'){
39 :             pt=fname1;
40 :             while(*pt++!=SPACE); *--pt='\0';
41 :             pt=fname1; pt++;
42 :             chdir(pt);
43 :         }
44 :         else break;
45 :     }
46 :     pt=fname1; pt=strchr(fname1,' '); *pt='\0';
47 :     Getpath(curpath,fname1);
48 :     setdta(oldDTA);
49 :     setdisk(i);
50 :     chdir(oldpath);
51 :     free(oldpath);
52 :     Fastputtext(MINC-5,MINR-1,MAXCO+5,MAXR+3,scrn);
53 :     return(fname1);
54 : }
55 :
56 : void Searchfile(char *curpath,char *findf,char *fname1)
57 : {
58 :     int result,attrib,h,x,y,i=0;
59 :     unsigned hour,min,year,month,date;
60 :     struct ffbk *file;
61 :     char buf[MAXCOL],fname[FLEN],*pt,key;
62 :     char far *fp,*fp1;
63 :
64 :     file=malloc(sizeof(struct ffbk));
65 :     if((result=findfirst(findf,file,0xFF))==0){
66 :         while(result==0){
67 :             attrib=_chmod(file->ff_name,0);
68 :             if((attrib & FA_DIREC || attrib & FA_ARCH)
69 :                && attrib!=-1)
70 :                 i++;
71 :             result=findnext(file);
72 :         }
73 :         fp=farcalloc(i,sizeof(buf));
74 :         if(!fp) return;
75 :         fp1=fp;
76 :     }
77 :     if((result=findfirst(findf,file,0xFF))==0){
78 :         while(result==0){
79 :             attrib=_chmod(file->ff_name,0);
80 :             if(attrib & FA_DIREC)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

81 :                                     strcpy( fname,"\\");
82 :                                     else
83 :                                         strcpy( fname,"");
84 :                                     strcat( fname,file->ff_name);
85 :                                     if( fname[1]!='.') i--;
86 :                                     if((attrib & FA_DIRC || attrib & FA_ARCH)
87 :                                         && fname[1]!='.' && attrib!=-1){
88 :                                         year=(file->ff_fdate>>9)+1980;
89 :                                         month=(file->ff_fdate>>5) & 0x0F;
90 :                                         date=file->ff_fdate & 0x1F;
91 :                                         hour=file->ff_ftime>>11;
92 :                                         min=(file->ff_ftime>>5) & 0x3F;
93 :                                         h=hour-12;
94 :                                         sprintf(buf,"%-13s %10lu  %.2d/%.2d/%.2d %.2d:%.2d%s",
95 :                                             fname,file->ff_fsize,date,month,year,
96 :                                             (h>0)?h:hour,min,(h>0)?"pm":"am");
97 :                                         strncpy( fp,buf,sizeof(buf));
98 :                                         fp+=sizeof(buf);
99 :                                     }
100 :                                     result=findnext(file);
101 :                                 }
102 :                                 for(pt=curpath,result=0;*pt!='\0' && result<2;pt++)
103 :                                     if(*pt=='\\') result++;
104 :                                 if(result>1){
105 :                                     sprintf(buf,"%-13s %-30s","\\..","Directory");
106 :                                     strncpy( fp,buf,sizeof(buf));
107 :                                     i++;
108 :                                 }
109 :                             }
110 :                             free(file);
111 :                             if(i==0){
112 :                                 Fastputs(MINC,MINR,0x30,"Not found...");
113 :                                 getch();
114 :                                 fname[0]='\0';
115 :                                 return; -
116 :                             }
117 :                             result=i; fp=fp1;
118 :                             for(i=0,x=MINC,y=MINR;i<result;i++,fp+=sizeof(buf)){
119 :                                 strncpy( fname,fp,FLEN); fname[FLEN]='\0';
120 :                                 Fastputs(x,y,0x30,fname);
121 :                                 if(x<MAXCO-FLEN) x+=FLEN;
122 :                                 else{
123 :                                     x=MINC; y++;
124 :                                 }
125 :                                 if(y>MAXR) break;
126 :                             }
127 :                             fp=fp1;
128 :                             for(x=MINC,y=MINR,h=0;;){
129 :                                 Setattr(x-1,y,FLEN,0x3E);
130 :                                 strncpy(buf,fp,sizeof(buf)); buf[MAXCOL]='\0';
131 :                                 Fastputs(MINC-2,MAXR+2,0x13,buf);
132 :                                 key=getch();
133 :                                 Setattr(x-1,y,FLEN,0x30);
134 :                                 if(key==0){
135 :                                     key=getch();
136 :                                     switch(key){
137 :                                         case UP:if(h>=3){
138 :                                             fp-=MAXF*sizeof(buf); h-=3;
139 :                                             if(y==MINR)
140 :                                                 Print_one_line(fp,y,h,result);
141 :                                             else y--;
142 :                                         }
143 :                                         break;
144 :                                         case DOWN:if(h+MAXF<result){
145 :                                             fp+=MAXF*sizeof(buf); h+=3;
146 :                                             if(y==MAXR)
147 :                                                 Print_one_line(fp,y,h,result);
148 :                                             else y++;
149 :                                         }
150 :                                         break;
151 :                                         case LEFT:if(h>0){
152 :                                             fp-=sizeof(buf); h--;
153 :                                             if(x<=MINC){
154 :                                                 x=MAXCO-FLEN+2;
155 :                                                 if(y==MINR)
156 :                                                     Print_one_line(fp,y,h,result);
157 :                                                 else y--;
158 :                                             }
159 :                                             else x-=FLEN;
160 :                                         }

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

161:         break;
162:     case RIGHT:if(h<result-1){
163:         fp+=sizeof(buf); h++;
164:         if(x>=MAXCO-FLEN){
165:             x=MINC;
166:             if(y==MAXR)
167:                 Print_one_line(fp,y,h,result);
168:             else y++;
169:         }
170:         else x+=FLEN;
171:     }
172:     break;
173: }
174: }
175: if(key==CR || key==ESC) break;
176: }
177: strncpy(fname1,fp,13); fname1[13]='\0';
178: if(key==ESC) fname1[0]='\0';
179: farfree(fp);
180: }
181:
182: void Getpath(char *curpath,char *fname)
183: {
184:     char path[MAXPATH],dr,*pt;
185:     int drive;
186:
187:     makewin(MINC-5,MINR-1,MAXCO+5,MAXR+3,0x7F,0x71,1);
188:     Cls(MINC-1,MINR,MAXCO+1,MAXR,0x30,SPACE);
189:     Fastputs(MINC-2,MAXR+1,0x7F,"File status");
190:     if(!*curpath){
191:         drive=getdisk();
192:         getcurdir(0,path);
193:         if(*path)
194:             strcat(path,"\\");
195:         dr='A'+drive;
196:         sprintf(curpath,"%c:\\%s",dr,path,fname);
197:     }
198:     else{
199:         drive=toupper(*curpath)-'A';
200:         setdisk(drive);
201:         pt=curpath; pt+=strlen(curpath);
202:         if(*--pt!='\\') strcat(curpath,"\\");
203:         chdir(curpath);
204:         strcat(curpath,fname);
205:     }
206:    strupr(curpath);
207:     Fastprintf(MINC-2+(MAXCO-MINC-strlen(curpath))/2,MINR-1,0x76,
208:         "< %s >",curpath);
209: }
210:
211: void Print_one_line(char far *fp,int y,int h,int result)
212: {
213:     register int i,j;
214:     char fname[13];
215:
216:     for(i=0,j=h*MAXF;i<j;i++,h--) fp-=MAXCOL;
217:     if(y==MAXR)
218:         movetext(MINC,MINR+1,MAXCO,MAXR,MINC,MINR);
219:     else movetext(MINC,MINR,MAXCO,MAXR-1,MINC,MINR+1);
220:     Cls(MINC,y,MAXCO,y,0x30,SPACE);
221:
222:     for(i=0;i<MAXF && h<result;i++,h++){
223:         strncpy(fname,fp,13); fname[13]='\0';
224:         Fastputs(MINC+i*FLEN,y,0x30,fname);
225:         fp+=MAXCOL;
226:     }
227: }

```

```
1 : /*
2 : This program will trap disk errors and prompt the user for action.
3 : */
4 :
5 : #define IGNORE 0
6 : #define RETRY 1
7 : #define ABORT 2
8 :
9 : /* define the error messages for trapping disk problems */
10 : static char *err_msg[] = { "write protect", "unknown unit", "drive not ready",
11 :     "unknown command", "data error (CRC)", "bad request", "seek error",
12 :     "unknown media type", "sector not found", "printer out of paper",
13 :     "write fault", "read fault", "general failure", "reserved",
14 :     "reserved", "invalid disk change"
15 : };
16 :
17 : error_win(char *msg)
18 : {
19 :     int retval, x, y, attr;
20 :
21 :     x=25; y=16; attr=0x4E;
22 :     makewin(x-4, y-1, x+40, y+2, attr, attr, 1);
23 :     while(*msg!='\0'){
24 :         if(*msg!='\n') Fastprintf(x++, y, attr, "%c", *msg++);
25 :         else{
26 :             msg++; x=25; y++; attr++;
27 :         }
28 :     }
29 :     gotoxy(x, y);
30 :
31 : /*
32 : prompt for user to press a key to abort, retry, ignore
33 : */
34 : while(1)
35 : {
36 :     retval= tolower(getch());
37 :     if (retval == 'a')
38 :     {
39 :         retval = ABORT;
40 :         break;
41 :     }
42 :     if (retval == 'r')
43 :     {
44 :         retval = RETRY;
45 :         break;
46 :     }
47 :     if (retval == 'i')
48 :     {
49 :         retval = IGNORE;
50 :         break;
51 :     }
52 : }
53 : return(retval);
54 : }
55 :
56 : /*
57 : #pragma warn -par reduces warnings which occur
58 : due to the non use of the parameters errval,
59 : bp and si to the handler.
60 : */
61 : #pragma warn -par
62 :
63 : int handler(int errval, int ax, int bp, int si)
64 : {
65 :     static char msg[80];
66 :     unsigned di;
67 :     int drive;
68 :     int errorno;
69 :
70 :     di= _DI;
71 :
72 : /*
73 : if this is not a disk error then it was
74 : another device having trouble
75 : */
76 :
77 : if (ax < 0)
78 : {
79 :     /* report the error */
80 :     error_win("Device error");
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
81 :      /* and return to the program directly requesting abort */
82 :      hardretn(ABORT);
83 :  }
84 :  /* otherwise it was a disk error */
85 :  drive = ax & 0x00FF;
86 :  errorno = di & 0x00FF;
87 :  /* report which error it was */
88 :  sprintf(msg, "Error: %s on drive %c\nAbort, Retry, Ignore: ",
89 :      err_msg[errorno], 'A' + drive);
90 :  /*
91 :  return to the program via dos interrupt 0x23 with abort, retry,
92 :  or ignore as input by the user.
93 :  */
94 :
95 :      hardresume(error_win(msg));
96 :      return ABORT;
97 :  }
98 :  #pragma warn +par
99 :
100: void install_handler(void)
101: {
102: /* install our handler on the hardware problem interrupt */
103:     harderr(handler);
104: }
105: †
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

1. D.W.Jordan and P Smith, "Nonlinear ordinary Differential Equations", 1986
2. Jean Jacques E. Slotine and Weiping, " Appied Nonlinear Control ",1991
3. Katsuhiko Okata , " Modern Control Engineering " ,1990
4. Arithur L.Greensile ,"Elements Of Modern Control Theory",1970
5. K.J.Astrom and B.Wittenmark: "Adaptive Control ",1989

