บทที่ 2

หลักการทำงาน และโครงสร้างข้อมูล

ภายในบทนี้ เป็นการกล่าวถึงหลักการทำงาน โครงสร้างของวัตถุ และลำดับขั้นตอนต่างๆ ของโปรแกรม

2.1 หลักการทำงาน

เนื่องจากการพัฒนาโปรแกรมอิงอยู่บนการใช้ภาษา C++ ซึ่งเป็นภาษาสำหรับการโปรแกรมเชิงวัตถุ ซึ่งอาศัยการพิจารณาโครงสร้างข้อมูลและหน้าที่การทำงานต่างๆ จากวัตถุประสงค์และข้อกำหนด จากนั้นจึงใช้ความสัมพันธ์ระหว่างข้อมูลและหน้าที่การทำงานเพื่อเปลี่ยนโครงสร้างข้อมูลและฟังก์ชันในการทำงานออกเป็นวัตถุ (Object) ต่าง ๆ ในหลายๆ ระดับชั้น ในแต่ละวัตถุมีสมาชิกภายในอยู่จำนวนหนึ่ง ทั้งที่เป็นข้อมูล และฟังก์ชันการทำงาน ในต้นฉบับโปรแกรมที่พัฒนาในแบบเชิงวัตถุ จึงเป็นการเรียกใช้วัตถุแบบต่างๆ ที่กำหนดขึ้น เพื่อให้ได้ขั้นตอนการทำงานตามที่ต้องการ กลไกในภาษา C++ ที่ใช้ในการนำเสนอระบบวัตถุ คือ การใช้คลาส (Class) ซึ่งเป็นสิ่งที่พัฒนาต่อมาจากโครงสร้างข้อมูลแบบสตรักเจอร์ (Structure) ในภาษา C สตรักเจอร์ในภาษา C เป็นการรวมข้อมูลที่เกี่ยวข้องกันไว้ภายใต้ชื่อเดียวกัน ส่วนคลาสในภาษา C++ เป็นการเพิ่มความสามารถของสตรักเจอร์ในการรวมข้อมูลและฟังก์ชันไว้ด้วยกัน พร้อมทั้งเพิ่มเติมความสามารถในด้าน Inheritance และ Polymorphism การพัฒนาโปรแกรมด้วยภาษา C++ จึงเป็นการสร้างคลาสแบบต่างๆ ขึ้นมา โดยแต่ละคลาสมีหน้าที่เก็บข้อมูลและจัดการงานที่แตกต่างกันไป สำหรับรายละเอียดของการใช้ภาษา C++ สามารถศึกษาได้จาก (Berry, 1992; Ladd, 1992; Skinner, 1992; Schildt, 1994)

1. การพิจารณาหน้าที่ต่างๆ ของโปรแกรม เมื่อพิจารณาวัตถุประสงค์ ขอบเขต และข้อกำหนดของโปรแกรม สามารถสรุปเป็นหน้าที่การทำงานได้ดังนี้

    1.1 การรับข้อมูลของระบบ

        1.1.1 การรับข้อมูลทั่วไปของระบบ

        1.1.2 การรับข้อมูลแบบจำลองทางคณิตศาสตร์ของระบบ

        1.1.3 การรับข้อมูลในการทำงานของฟังก์ชันต่างๆในโปรแกรม

    1.2 การวิเคราะห์ข้อมูลของระบบที่ป้อนเข้ามา

1.2.1 การวิเคราะห์ตามทฤษฎีควบคุมแบบคลาสสิก

1.2.2 การวิเคราะห์ตามทฤษฎีควบคุมแบบสมัยใหม่

1.2.3 การวิเคราะห์ตามทฤษฎีควบคุมแบบดิจิตัล

1.3 การออกแบบระบบควบคุม ตามข้อกำหนดที่ป้อนให้

1.3.1 การออกแบบตามทฤษฎีควบคุมแบบคลาสสิก

1.3.2 การออกแบบตามทฤษฎีควบคุมแบบสมัยใหม่

1.3.3 การออกแบบตามทฤษฎีควบคุมแบบดิจิตัล

1.4 การจำลองผลตอบ

1.4.1 การจำลองผลตอบเชิงเวลา

1.4.2 การจำลองผลตอบเชิงความถี่

1.4.3 การจำลองผลในรูป Root Locus

1.5 การแสดงผลลัพธ์จากการวิเคราะห์และออกแบบ

1.5.1 การแสดงผลลัพธ์ที่เป็นข้อมูลแบบตัวเลข

1.5.2 การแสดงผลลัพธ์ที่เป็นรูปกราฟ

1.6 การจัดเก็บและดึงข้อมูลจากดิสค์

1.6.1 การจัดเก็บข้อมูลแบบจำลองทางคณิตศาสตร์ของระบบ

1.6.2 การจัดเก็บข้อมูลที่เป็นตัวเลขแบบต่างๆ

1.7 การพิมพ์ผลออกทางเครื่องพิมพ์

1.7.1 การพิมพ์ผลซึ่งเป็นข้อมูลแบบตัวเลข

1.7.2 การพิมพ์รูปกราฟ

เมื่อนำหน้าที่การทำงานต่างๆ ข้างต้นไปจัดเป็นแผนผังความสัมพันธ์ของฟังก์ชันของโปรแกรม จะได้เป็นแผนผังในรูป 2-1

2. การพิจารณาโครงสร้างข้อมูลของโปรแกรม เมื่อคำนึงถึงรายละเอียดของทฤษฎีควบคุม ความแตกต่างในด้านการคำนวณ โครงสร้างการทำงานของ Windows และฟังก์ชันของโปรแกรม สามารถสรุปชนิดของข้อมูลที่จำเป็นในโปรแกรม ดังนี้

2.1 ข้อมูลเกี่ยวกับการจัดการแสดงผลตามรูปแบบของ Windows ได้แก่

2.1.1 ข้อมูลเกี่ยวกับสถานะของโปรแกรมประยุกต์

2.1.2 ข้อมูลเกี่ยวกับการควบคุม Window ที่แสดงบนหน้าจอ

2.1.3 ข้อมูลเกี่ยวกับ Resource ที่ใช้ในโปรแกรม

2.1.4 ข้อมูลเกี่ยวกับระบบแสดงผลแบบกราฟฟิกของ Windows

2.1.5 ข้อมูลเกี่ยวกับการจัดการหน่วยความจำใน Windows

2.2 ข้อมูลพื้นฐานสำหรับการคำนวณทางคณิตศาสตร์ ได้แก่

2.2.1 ข้อมูลประเภทเมตริกซ์

2.2.2 ข้อมูลประเภทเวกเตอร์

2.2.3 ข้อมูลประเภทโพลิโนเมียล

2.3 ข้อมูลแบบจำลองทางคณิตศาสตร์ของระบบ ได้แก่

2.3.1 ข้อมูลในรูปฟังก์ชันถ่ายโอน (Transfer Function)

2.3.2 ข้อมูลในรูปสมการสถานะ (State Equation)

2.3.3 ข้อมูลในรูปฟังก์ชันถ่ายโอนแบบไม่ต่อเนื่อง (Puise Transfer Function)

2.3.4 ข้อมูลในรูปสมการสถานะแบบไม่ต่อเนื่อง (Discrete-time State Equation)

2.4 ข้อมูลที่ใช้ในกระบวนการทำงานต่างๆ ของโปรแกรม ได้แก่

2.4.1 ข้อมูลที่ใช้ในเก็บสถานะทั่วไปของข้อมูลระบบ

2.4.2 ข้อมูลที่ใช้ในการควบคุมระบบติดต่อผู้ใช้ของโปรแกรม ได้แก่

2.4 2.1 ข้อมูลที่ใช้ควบคุมการเลือกฟังก์ชันของโปรแกรม

2.4.2.2 ข้อมูลที่ใช้ในระบบรับและแสดงข้อมูลตัวเลข

2.4.2.3 ข้อมูลที่ใช้ในระบบแสดงผลแบบรูปกราฟ

2.4.3 ข้อมูลที่เก็บพารามิเตอร์ที่ใช้ในการทำงานของโปรแกรม

2.5 ข้อมูลที่ใช้ในการคำนวณต่างๆ ภายในโปรแกรม

2.5.1 ข้อมูลที่ใช้ในขั้นตอนการวิเคราะห์และออกแบบ

2.5.2 ข้อมูลที่ใช้ในการสร้างกราฟแบบต่างๆ

2.5.3 ข้อมูลที่เก็บพารามิเตอร์ที่ใช้ในการคำนวณ

สำหรับความสัมพันธ์ระหว่างโครงสร้างข้อมูลภายในโปรแกรมประเภทต่างๆ สามารถสรุปได้ในรูป 2-2

3. การวิเคราะห์การทำงานของโปรแกรม เมื่อพิจารณาหน้าที่การทำงาน และโครงสร้างข้อมูลของโปรแกรม พบว่าหากโปรแกรมมีหน้าที่การทำงานที่จำเพาะเจาะจงตามชนิดของข้อมูลระบบ โครงสร้างของโปรแกรมจะมีขนาดใหญ่และซับซ้อนมาก เนื่องจากโปรแกรมต้องรองรับข้อมูลแบบจำลองทางคณิตศาสตร์ถึง 4 รูปแบบ และในการประมวลผลข้อมูลแต่ละรูปแบบต้องมีการรับข้อมูล การวิเคราะห์ การออกแบบ และการแสดงผลลัพธ์ซึ่งต่างกันออกไป นอกจากนี้ การแปลงรูประหว่างข้อมูลระบบในรูปต่างๆกันยังทำได้ยาก เพราะมีข้อมูลจำนวนมากที่ต้องคำนึงถึง เพื่อให้โปรแกรมที่พัฒนาขึ้นมีความเรียบง่ายในการใช้งาน แต่ยังคงความสามารถในการประมวลผลต่างๆ จึงต้องมีการศึกษาเพื่อจัดโครงสร้างของข้อมูลใน

โปรแกรมให้เหมาะสม จากการศึกษา พบว่ามีข้อสังเกตุเกี่ยวกับหน้าที่การทำงานและชนิดของข้อมูลที่น่าสนใจ ดังนี้

3.1 แบบจำลองทางคณิตศาสตร์ของระบบ มีการนำเสนอใน 2 รูปแบบหลัก คือ ฟังก์ชันถ่ายโอน และสมการสถานะ โดยข้อมูลของระบบแบบไม่ต่อเนื่อง (Discrete System) ตามทฤษฎีควบคุมแบบดิจิตัล แตกต่างจากระบบแบบต่อเนื่อง (Continuous System) ในทฤษฎีควบคุมแบบคลาสสิกและแบบสมัยใหม่ในเรื่องอัตราการสุ่มสัญญาณเท่านั้น

3.2 การคำนวณที่ใช้ในด้านการวิเคราะห์และออกแบบระบบควบคุม มีความแตกต่างกันตามแต่รูปแบบของข้อมูลที่ใช้ แต่เทคนิคที่ใช้ยังอิงบนหลักการเดียวกัน เมื่อรูปแบบของข้อมูลเหมือนกัน

3.3 ข้อมูลที่ใช้ในแบบจำลองทางคณิตศาสตร์ของระบบ ข้อมูลที่ใช้ในการคำนวณ และข้อมูลที่ได้จากการวิเคราะห์และออกแบบ สามารถจัดแบ่งตามรูปแบบของข้อมูลออกเป็น 5 ประเภทใหญ่ๆ คือ

3.3.1 ค่าตัวเลข

3.3.2 ฟังก์ชันถ่ายโอน

3.3.3 เมตริกซ์

3.3.4 เวกเตอร์

3.3.5 พารามิเตอร์ของอุปกรณ์ในระบบควบคุม

3.4 ระบบรับข้อมูลอย่างน้อยต้องเตรียมไว้สำหรับรูปแบบของข้อมูล ดังนี้

3.4.1 แบบจำลองทางคณิตศาสตร์ทั้ง 4 แบบ

3.4.2 เมตริกซ์

3.4.3 ฟังก์ชันถ่ายโอน

3.4.4 พารามิเตอร์ของตัวควบคุม PID และตัวชดเชย

3.4.5 ข้อกำหนดต่างๆ ในการวิเคราะห์และออกแบบ ซึ่งมีความหลากหลายแล้วแต่การคำนวณที่เลือกใช้ในขณะนั้น โดยสามารถจัดกลุ่มของประเภทข้อกำหนดเป็น 4 กลุ่มดังนี้

3.4.5.1 ข้อมูลเป็นตัวเลขบอกปริมาณ

3.4.5.2 ข้อมูลเป็นเวกเตอร์

3.4.5.3 ข้อมูลเป็นเมตริกซ์

3.4.5.4 ข้อมูลเป็นพารามิเตอร์ของตัวควบคุมแบบ PID หรือตัวชดเชยล้ำหน้า ล้าหลัง

3.5 การป้อนข้อมูล ควรมีระบบที่อำนวยความสะดวกในการแก้ไข ง่ายต่อการเรียนรู้ รวมทั้งควรสามารถนำข้อมูลที่ป้อนแล้วกลับมาใช้ใหม่ได้ เพราะในการออกแบบอาจจำเป็นต้องมีการใช้ชุดของข้อมูลหรือข้อกำหนดที่มีความคล้ายคลึงกันบ่อย

3.6 การคำนวณที่ใช้ในการวิเคราะห์ และออกแบบระบบควบคุม มีบางส่วนที่สามารถใช้อัลกอริทึมซ้ำกันได้

3.7 อัลกอริทึมที่ใช้ในการวิเคราะห์และออกแบบสามารถเลือกใช้ตามความเหมาะสมในแต่ละกรณี

3.8 การจัดการระบบ Windows เป็นหน้าที่ซึ่งแยกจากส่วนของการคำนวณในทฤษฎีควบคุม การทำงานในส่วนนี้มีความเกี่ยวข้องกับการคำนวณในด้านการเรียกใช้ฟังก์ชันในการวิเคราะห์และออกแบบต่างๆ ตามการสั่งงานของผู้ใช้

4. โครงสร้างการทำงาน จากการศึกษาเงื่อนไขต่างๆ ของโปรแกรม จึงได้เข้าสู่การกำหนดโครงสร้างการทำงานของโปรแกรม โดยโครงสร้างของโปรแกรมสามารถสรุปอย่างคร่าวๆ ได้ดังนี้

4.1 ระบบเมนูของโปรแกรม รูปแบบการใช้งานโปรแกรม ใช้ระบบสั่งงานแบบเมนู เพราะเป็นระบบที่เรียนรู้ได้ง่าย รวมทั้งเป็นระบบที่ Windows ให้การสนับสนุนอย่างเต็มที่ เนื่องจากฟังก์ชันในการวิเคราะห์และออกแบบระบบควบคุมมีความแตกต่างกันค่อนข้างเด่นชัดดังนั้นระบบเมนูของโปรแกรม จึงต้องมีความสามารถในการปรับให้เข้ากับชนิดของข้อมูลระบบในขณะนั้นได้ นอกจากนี้การเรียกใช้งานฟังก์ชันต่างๆ ในการวิเคราะห์และออกแบบ ยังต้องอิงอยู่บนรูปแบบมาตรฐานของโปรแกรมประยุกต์ที่ทำงานบน Windows เพื่อให้เกิดความคุ้นเคย เพื่อให้สามารถเรียนรู้การใช้งานโปรแกรมได้อย่างรวดเร็ว สำหรับรายละเอียดของระบบเมนู มีดังนี้

4.1.1 เมนู File เป็นเมนูที่ใช้ในการควบคุมการทำงานทั่วไปของโปรแกรม ได้แก่ การติดต่อดิสค์ การจัดการเครื่องพิมพ์ และการจบโปรแกรม สำหรับเมนูย่อยในหัวข้อเมนู File ได้แก่

4.1.1.1 เมนู New ใช้สำหรับการสร้างข้อมูลใหม่

4.1.1.2 เมนู Open ใช้อ่านข้อมูลจากดิสค์

4.1.1.3 เมนู Save ใช้จัดเก็บข้อมูลลงดิสค์

4.1.1.4 เมนู Save As ใช้จัดเก็บข้อมูลลงดิสค์ โดยมีการเปลี่ยนแปลงชื่อของไฟล์ใหม่

4.1.1.5 เมนู Print ใช้สั่งพิมพ์ข้อมูลออกทางเครื่องพิมพ์

4.1.1.6 เมนู Printer Setup ใช้ควบคุมสถานะเครื่องพิมพ์

4.1.1.7 เมนู Exit ใช้จบการทำงานของโปรแกรม

4.1.2 เมนู Edit ใช้สำหรับควบคุมการแก้ไขข้อมูล รวมทั้งการติดต่อกับ Clipboard การแสดงเมนูย่อยในหัวข้อนี้ มีการปรับเปลี่ยนเพื่อให้เหมาะสมกับชนิดของ Window ที่แสดงอยู่ในขณะนั้นได้ สำหรับเมนูย่อยในหัวข้อนี้ ได้แก่

4.1.2.1 เมนู Cut ใช้สำหรับตัดข้อมูลแล้วส่งให้ Clipboard

4.1.2.2 เมนู Copy ใช้ทำสำเนาข้อมูลให้ Clipboard

4.1.2.3 เมนู Paste นำข้อมูลจาก Clipboard มาใช้

4.1.3 เมนู Analysis เป็นเมนูที่ใช้สำหรับวิเคราะห์ข้อมูลของระบบควบคุม หัวข้อการวิเคราะห์ในเมนูนี้ มีการเปลี่ยนแปลงตามชนิดของข้อมูลในขณะนั้นโดยอัตโนมัติ สำหรับหัวข้อของการวิเคราะห์ โดยสรุปมีดังนี้

4.1.3.1 Pole&zero Location ใช้หาตำแหน่งเสาและศูนย์ของระบบ

4.1.3.2 Routh Table ใช้สร้าง Routh Table สำหรับการวิเคราะห์เสถียรภาพของระบบต่อเนื่อง

4.1.3.3 Steady-state Error คำนวณค่าความผิดพลาดในสภาวะอยู่ตัวของระบบ

4.1.3.4 Gain & Phase Margin ใช้คำนวณหาค่า Gain Margin และ Phase Margin

4.1.3.5 Eigenvalue ใช้หาค่า Eigenvalue ของระบบ

4.1.3.6 Controllability ใช้ในการวิเคราะห์คุณสมบัติ Controllability ของระบบ

4.1.3.7 Observability ใช้ในการวิเคราะห์คุณสมบัติ Observability ของระบบ

4.1.3.8 Jury Table ใช้สำหรับการสร้าง Jury Table สำหรับการพิจารณาเสถียรภาพของระบบไม่ต่อเนื่อง

สำหรับรายการวิเคราะห์ในเมนู Analysis จะปรากฏให้เลือกแล้วแต่ข้อมูลซึ่งกำลังใช้งานในขณะนั้น กรณีกำลังเลือกใช้ข้อมูลของระบบในรูปฟังก์ชันถ่ายโอน จะสามารถเลือกใช้ฟังก์ชัน 4.1.3.1 ถึง 4.1.3.4 แต่ถ้าระบบเป็นแบบไม่ต่อเนื่อง จะแทนฟังก์ชัน 4.1.3.2 ด้วย 4.1.3.8 กรณีข้อมูลระบบเป็นสมการสถานะ จะสามารถเลือกใช้ฟังก์ชัน 4.1.3.5 ถึง 4.1.3.7

4.1.4 เมนู Design เป็นเมนูที่ใช้สำหรับการออกแบบระบบควบคุม โดยหัวข้อการออกแบบในเมนูนี้ มีการเปลี่ยนแปลงตามชนิดของข้อมูลในขณะนั้นอย่างอัตโนมัติ สำหรับหัวข้อของการออกแบบ โดยสรุปมีดังนี้

4.1.4.1 PID Tuning ใช้สำหรับหาพารามิเตอร์ของตัวควบคุมแบบ PID

4.1.4.2 Compensator Design ใช้สำหรับออกแบบตัวชดเชยแบบล้ำหน้า ล้าหลัง และแบบล้ำหน้า-ล้าหลัง โดยอาศัยเทคนิคการออกแบบเชิงความถี่

4.1.4.3 Pole Placement ใช้สำหรับออกแบบระบบควบคุมแบบป้อนกลับด้วยการกำหนดตำแหน่งเสา

4.1.4.4 State Observer ใช้ออกแบบ Observer ด้วยเทคนิคการกำหนดตำแหน่ง Pole

4.1.4.5 Optimum Feedback ใช้สำหรับออกแบบระบบป้อนกลับ ด้วยเทคนิคเล็งเลิศ (Optimization Technique)

4.1.4.6 Kalman Filter ใช้ในการออกแบบ Kalman Filter ซึ่งเป็น Filter ที่ออกแบบด้วยหลักการเล็งเลิศ

ในกรณีข้อมูลระบบเป็นฟังก์ชันถ่ายโอน จะสามารถใช้ฟังก์ชันในหัวข้อ 4.1.4.1 และ 4.1.4.2 สำหรับกรณีข้อมูลระบบอยู่ในรูปสมการสถานะ จะสามารถเลือกใช้ฟังก์ชัน 4.1.4.3 ถึง 4.1.4.6

4.1.5 เมนู Simulation เป็นเมนูที่ใช้สำหรับการจำลองผลตอบ โดยหัวข้อในการจำลองผลสามารถปรับตามชนิดของข้อมูลโดยอัตโนมัติ สำหรับหัวข้อของการจำลองผล ได้แก่

4.1.5.1 Time Response เป็นการจำลองผลตอบเชิงเวลาของระบบ

4.1.5.2 Nyquist Plot ใช้จำลองผลตอบเชิงความถี่ โดยแสดงผลในรูปกราฟแบบ Nyquist Plot

4.1.5.3 Bode Plot ใช้จำลองผลตอบเชิงความถี่ โดยแสดงผลในรูปกราฟแบบ Bode Plot

4.1.5.4 Nichols Chart ใช้จำลองผลตอบเชิงความถี่ โดยแสดงผลในรูปกราฟแบบ Nichols Chart

4.1.5.5 Root Locus เป็นการสร้างกราฟแนวทางเดินของตำแน่งเสาระบบวงปิด หรือ Root Loci โดยอาศัยเทคนิค Root Locus

กรณีข้อมูลระบบเป็นฟังก์ชันถ่ายโอนจะสามารถเลือกใช้ได้ทุกฟังก์ชัน โดยถ้าระบบเป็นแบบไม่ต่อเนื่อง จะอาศัยการทำ Bilinear Transformation กลับมาเพื่อหาผลตอบเชิงความถี่ แต่สำหรับข้อมูลระบบที่เป็นสมการสถานะจะไม่มีรายการย่อยให้เลือก กล่าวคือสามารถทำการจำลองผลตอบในเชิงเวลาเท่านั้น

4.1.6 เมนู Option ใช้ตั้งสถานะต่างๆของโปรแกรม ประกอบด้วยรายการย่อย ดังนี้

4.1.6.1 Computation ใช้สำหรับตั้งค่าคงที่ต่างๆ ที่ใช้ในการคำนวณ

4.1.6.2 Environment ใช้สำหรับการตั้งเงื่อนไขและสถานะต่างๆ ของโปรแกรม

4.1.7 เมนู Window ใช้ควบคุมการแสดงผลของ Window ย่อย ที่แสดงผลภายใน Window หลัก ตามมาตรฐานรูปแบบการใช้งานแบบ Multiple Document Interface ของ Windows สำหรับเมนูย่อยในหัวข้อเมนูนี้ ได้แก่

4.1.7.1 เมนู Tile จัด Window ให้แสดงผลแบบตาราง

4.1.7.2 เมนู Cascade จัด Window ให้แสดงผลในแบบทับซ้อนกัน

4.1.7.3 เมนู Arrange Icons ใช้สำหรับสั่งให้ Window ย่อยที่เป็น Icon เรียงตัวอย่างเป็นระเบียบ

4.1.7.4 เมนู Close All ใช้สำหรับปิดการแสดงผลของทุก Window ย่อย

สำหรับแผงผังแสดงฟังก์ชันภายในเมนูของโปรแกรม ภายใต้สภาวะการทำงานแบบต่างๆ แสดงไว้ในรูป 2-3

4.2 ระบบรับและแสดงผลข้อมูลของโปรแกรมไม่ได้ใช้ข้อมูลแบบจำลองทางคณิตศาสตร์ของระบบโดยตรง เพราะทำให้การทำงานของโปรแกรมซับซ้อนมากเกินไป ระบบรับและแสดงผลข้อมูลของโปรแกรมนำเสนอโดยใช้รูปแบบที่เรียกว่า Desktop คือ ผู้ใช้จะทำการป้อนข้อมูลให้โปรแกรมในรูปของเมตริกซ์ ฟังก์ชันถ่ายโอน หรือพารามิเตอร์ของอุปกรณ์ในระบบควบคุมเท่านั้น จากนั้นเมื่อผู้ใช้ต้องการวิเคราะห์หรือออกแบบระบบควบคุม สามารถเลือกใช้ฟังก์ชันของโปรแกรมในเมนู Analysis Design และ Simulation สำหรับการคำนวณได้ การใช้ระบบ Desktop นี้มีจุดเด่นในหลายด้าน ดังนี้

4.2.1 ลดเวลาในการป้อนข้อมูล ในกรณีที่เป็นระบบที่มีข้อมูลบางส่วนเหมือนกัน

4.2.2 สามารถทำการออกแบบระบบควบคุมพร้อมกันหลายระบบ ในการเรียกใช้โปรแกรมเพียงครั้งเดียวได้

4.2.3 สามารถทำการต่อเชื่อมระบบในรูปต่างๆ กัน เพื่อให้ได้ข้อมูลของระบบที่มีความซับซ้อนมากขึ้นได้

4.3 สำหรับการพัฒนาระบบคลาสในโปรแกรม ได้ใช้โครงสร้างของคลาสแบบที่เรียกว่า Envelope-Letter ซึ่งนำเสนอโดย Coplien (1992) โครงสร้างของคลาสแบบนี้ เป็นการขยายความสามารถของกลไก Inheritance และ Polymorphism ให้มากกว่าสิ่งที่

ภาษา C++ มีไว้ เนื่องจากภาษา C++ แม้ว่าได้เพิ่มความสามารถในการโปรแกรมเชิงวัตถุแล้ว แต่ยังมีจุดอ่อนในบางด้านอยู่ ได้แก่

1. ในภาษา C++ เมื่อต้องการให้โปรแกรมสามารถเลือกการทำงานให้เหมาะสมกับวัตถุในขณะนั้น ต้องอาศัยการใช้พอยน์เตอร์ (Pointer) ของคลาสต้นแบบ (Base Class) ซึ่งชี้ไปยังคลาสประยุกต์ (Derived Class) ใดๆ ที่สืบทอดคุณสมบัติจากคลาสต้นแบบ เมื่อใช้คู่กับฟังก์ชันเสมือน (Virtual Function) จะทำให้โปรแกรมสามารถทำงานตามชนิดของคลาสประยุกต์ที่กำลังชี้อยู่ได้โดยอัตโนมัติ แม้ว่าพอยน์เตอร์นั้นยังคงเป็นของคลาสต้นแบบ จุดอ่อนของการใช้กลไกแบบพื้นฐานของภาษา C++ คือ การใช้พอยน์เตอร์แทนตัวข้อมูลจริง โดยในกรณีซึ่งเกิดการแปลงชนิดของพอยน์เตอร์ขึ้น อาจทำให้โปรแกรมเกิดอาการหลงชนิดของคลาสขึ้น ทำให้ระบบ Inheritance และ Polymorphism ไม่ทำงานได้

2. โครงสร้างแบบคลาสทั่วไปในภาษา C++ ยังไม่สามารถรองรับการเปลี่ยนแปลงในระบบโครงสร้างของคลาส เนื่องจากหากมีการเปลี่ยนแปลงโครงสร้างข้อมูลในคลาสต้นแบบ อาจทำให้ต้องมีการเปลี่ยนแปลงตลอดลำดับชั้นการสืบทอดความสามารถได้

3. การใช้คลาสในภาษา C++ ยังไม่สนับสนุนความหมายในการโปรแกรมเชิงวัตถุได้อย่างเต็มที่ ในหลักการเชิงวัตถุ เมื่อกำหนดให้ชื่อ A หมายถึงวัตถุหนึ่ง และมีการกำหนดว่าชื่อ B เหมือนชื่อ A ทั้ง A และ B ควรอ้างถึงวัตถุเดียวกัน แต่เพื่อให้เกิดความเข้ากันได้กับภาษา C ทำให้ภาษา C++ ใช้การทำสำเนาข้อมูลทั้งหมดของวัตถุซึ่งชื่อ A อ้างถึงไปให้วัตถุชื่อ B ลักษณะเช่นนี้อาจไม่มีผลมากนักสำหรับข้อมูลทั่วไป แต่ในโปรแกรมบน Windows ซึ่งมีการใช้ Handle ในการอ้างถึงสิ่งต่างๆ สิ่งนี้อาจทำให้เกิดการทำงานที่ไม่เหมาะสมได้ ตัวอย่างเช่น เมื่อกำหนดให้ A เป็นตัวแปรที่แทน Window ซึ่งปรากฏบนหน้าจอ การกำหนดให้ B = A ควรมีความหมายอย่างไร ? B คือ Window เดียวกับ A หรือ B คือ Window ซึ่งมีทุกอย่างเหมือน A

4. ระบบอ้างถึงวัตถุในภาษา C++ ทำให้เกิดการซ้ำซ้อนของหน่วยความจำ โดยไม่จำเป็น เมื่อกำหนดให้ตัวแปร B เท่ากับ A กลไกของภาษา C++ จะทำการทำสำเนาข้อมูลทั้งหมดในตัวแปร A ไปให้ตัวแปร B ดังนั้นการกำหนดค่าให้ตัวแปร จึงใช้ขนาดของหน่วยความจำ เพิ่มขึ้นเท่ากับขนาดของชนิดของวัตถุนั้นทุกครั้ง

การใช้โครงสร้างแบบ Envelope-Letter เป็นการเข้ามาแก้ไขจุดอ่อนเหล่านี้ โดยหลักการของ Envelope-Letter คือ การสร้างคลาสเป็น 2 ระดับชั้น ได้แก่

1. ระดับ Envelope ทำหน้าที่ติดต่อกับการเรียกใช้งานจากภายนอก ภายในคลาสแบบ Envelope มีเพียงข้อมูลของพอยน์เตอร์ที่ชี้ไปยังคลาสแบบ Letter เท่านั้น โดยคลาสแบบ Envelope มีหน้าที่เกี่ยวกับการสร้าง และการอ้างอิงถึงวัตถุในขณะทำงานเท่านั้น ส่วนการทำงานอื่นๆนั้น คลาสแบบ Envelope จะส่งผ่านไปให้คลาสแบบ Letter ทั้งหมด

2. ระดับ Letter ทำหน้าที่ประมวลผลการเรียกใช้ฟังก์ชันสมาชิกทุกอย่าง โดยคลาสแบบ Letter มีความคล้ายคลึงกับโครงสร้างคลาสแบบทั่วไป เพียงแต่ในการเรียกใช้งานนั้น ไม่มีการเรียกใช้งานคลาสแบบ Letter โดยการเรียกใช้งานทุกอย่างต้องผ่านมาจากคลาสแบบ Envelope ซึ่งห่อหุ้มคลาสแบบ Letter นั้น

การใช้โครงสร้างแบบ Envelope-Letter มีข้อดีหลายอย่าง เนื่องจากการแบ่งโครงสร้างออกเป็น 2 ระดับชั้น ทำให้แต่ละระดับชั้นต่างช่วยส่งเสริมการทำงานซึ่งกันและกัน โดยคลาสแบบ Envelope ทำให้การใช้คลาสมีลักษณะของวัตถุที่แท้จริงขึ้นมา รวมทั้งสามารถป้องกันพอยน์เตอร์จากการแปลงชนิดโดยอุบัติเหตุได้ ส่วนคลาสแบบ Letter ก็สามารถทำงานได้อย่างเต็มที่ โดยไม่ต้องพะวงถึงปัญหาความผิดพลาดในการใช้งาน นอกจากนี้โครงสร้างแบบ Envelope-Letter ยังสนับสนุนการสร้างวัตถุหลายๆ ชนิดในขณะทำงานได้ โดยไม่ต้องคำนึงถึงตั้งแต่ขั้นตอนการโปรแกรม เพราะคลาสแบบ Envelope จะรับหน้าที่ทั้งหมดในการจัดสร้างวัตถุในขณะโปรแกรมทำงานให้เอง สำหรับตัวอย่างของโครงสร้างแบบ Envelope-Letter อยู่ในรูป 2-4

4.4 ระบบคลาสที่ใช้สำหรับแทนข้อมูลในรูปคณิตศาสตร์ มีอยู่ 3 ประเภท คือ เวกเตอร์ เมตริกซ์ และฟังก์ชันถ่ายโอน เนื่องจากข้อมูลในรูปเวกเตอร์และเมตริกซ์ ต้องสามารถรับข้อมูลที่เป็นทั้งค่าจำนวนจริงและเชิงซ้อน อีกทั้งระบบอ้างอิงสมาชิกต้องสามารถปรับตัวเองให้เหมาะสมกับข้อมูลภายในเวกเตอร์หรือเมตริกซ์ขณะนั้นได้ ส่วนข้อมูลในรูปฟังก์ชันถ่ายโอนมีทั้งเป็นรูปโพลิโนเมียลและรูปเทอมของเสาและศูนย์ ทำให้ข้อมูลเกี่ยวกับคณิตศาสตร์มีความซับซ้อนมาก การใช้โครงสร้างของระบบคลาสแบบ Envelope-Letter ไม่สามารถรับกับความซับซ้อนแบบนี้ได้ ซึ่งในวิทยานิพนธ์ฉบับนี้ได้นำเสนอการแก้ปัญหาทั้งสองกรณีนี้ไว้ดังนี้

4.4.1 ข้อมูลแบบเวกเตอร์และเมตริกซ์ ข้อมูลทั้งสองรูปแบบนี้ มีคุณสมบัติที่เหมือนกัน คือ เป็นข้อมูลที่เก็บค่าสมาชิกได้ทั้งค่าจำนวนจริงและค่าจำนวนเชิงซ้อน โดยสามารถนำข้อมูลมาผ่านตัวกระทำทางคณิตศาสตร์ได้ ไม่ว่าข้อมูลภายในเป็นชนิดใด โดยชนิดของผลลัพธ์จากการคำนวณขึ้นอยู่กับชนิดของข้อมูลที่นำมาประมวลผล จากคุณสมบัติที่กล่าวมานี้ ทำให้โครงสร้างของคลาสที่ใช้กับข้อมูลแบบเวกเตอร์หรือเมตริกซ์ต้องมีความสามารถที่ปรับรูปแบบการคำนวณตามชนิดของข้อมูลที่นำมาร่วมใช้ในการประมวลผลได้โดยอัตโนมัติ จากการศึกษาพบว่า โครงสร้างของคลาสที่สามารถปรับตัวเองให้เหมาะสมกับชนิดของข้อมูลได้อย่างอัตโนมัติ และสามารถอ้างอิงถึงข้อมูลสมาชิกภายในคลาสได้ถูกต้อง ไม่สามารถทำได้ด้วยความสามารถของภาษา C++ ในขณะนี้ ดังนั้นเราจึงสร้างคลาสเฉพาะข้อมูลแบบเมตริกซ์ คือ matrix ซึ่งมีค่าสมาชิกเป็นจำนวนจริงเท่านั้น โดยข้อมูลแบบเวกเตอร์จะนำมาอิงอยู่บนการใช้คลาสของข้อมูลแบบเมตริกซ์อีกทีหนึ่ง การสร้างเฉพาะคลาสของเมตริกซ์ที่มีค่าสมาชิกเป็นจำนวนจริงเท่านั้น ไม่มีผลต่อการคำนวณที่ใช้ในการวิเคราะห์และออกแบบเพราะอัลกอริทึมทั้งหมดที่มีการคำนวณเมตริกซ์ในวิทยานิพนธ์ฉบับนี้ ใช้การคำนวณแบบค่าจำนวนจริงเท่านั้น

กรณีข้อกำหนดในรูปของเวกเตอร์ที่มีค่าสมาชิกเป็นจำนวนเชิงซ้อน ได้ใช้การสร้างคลาสสำหรับเก็บชุดข้อมูลขึ้นมาแทน โดยคลาสนี้คือ cplx_array ซึ่งมีฟังก์ชันสำหรับการเก็บข้อมูลและการอ้างอิงถึงข้อมูลได้ แต่ไม่มีฟังก์ชันสำหรับการประมวลผลทางคณิตศาสตร์

4.4.2 ข้อมูลแบบฟังก์ชันถ่ายโอน ฟังก์ชันถ่ายโอนสามารถจัดแบ่งตามรูปแบบการนำเสนอทางคณิตศาสตร์ออกเป็น 2 รูป คือ

4.4.2.1 อัตราส่วนของโพลิโนเมียล ซึ่งมีการนำเสนอเป็นอัตราส่วนระหว่างโพลิโนเมียลของตัวแปรเชิงซ้อน s จำนวน 2 เทอม โดยข้อมูลของฟังก์ชันถ่ายโอนในกรณีนี้ อยู่ในรูปของค่าสัมประสิทธิ์ของโพลิโนเมียลแต่ละเทอม ซึ่งเป็นค่าจำนวนจริง

4.4.2.2 อัตราส่วนระหว่างเทอมของเสาและศูนย์ ซึ่งนำเสนอในรูปของอัตราส่วนระหว่างเทอมของเสาและเทอมของศูนย์ สำหรับข้อมูลของฟังก์ชันถ่ายโอนในกรณีนี้ เป็นค่าของเสาและศูนย์ ซึ่งต้องเก็บอยู่ในรูปจำนวนเชิงซ้อน

เนื่องจากในทฤษฎีระบบควบคุมจำเป็นต้องมีการคำนวณเกี่ยวกับฟังก์ชันถ่ายโอนในหลายรูปแบบ ทั้งการคำนวณทางด้านคณิตศาสตร์ การคำนวณในการต่อเชื่อมระบบ และการวิเคราะห์และออกแบบระบบควบคุม ดังนั้นคลาสที่ใช้แทนฟังก์ชันถ่ายโอนจึงต้องมีความสามารถในการประมวลผลข้อมูลที่เก็บอยู่ภายในซึ่งมีความแตกต่างกัน โดยไม่สามารถสังเกตุได้จากการเรียกใช้งาน สำหรับโครงสร้างของคลาสที่ใช้กับข้อมูลในรูปของฟังก์ชันถ่ายโอน คือ โครงสร้างแบบ Envelope-Letter ที่มีการใช้เทคนิคการประมวลผลแบบ Multi-Methods หลักการของเทคนิคแบบ Multi-Methods คือ การเพิ่มฟังก์ชันเสมือน (Virtual Function) จำนวน 2 ฟังก์ชัน คือ IsA() และ Promote() เข้าไปในฟังก์ชันสมาชิกของคลาสแบบ Envelope-Letter โดยการประกาศของฟังก์ชันทั้งสองคือ

```
BOOL        IsA(string &TypeName);
transfn &Promote(transfn &Source);
```

สำหรับฟังก์ชันสมาชิก IsA() ทำหน้าที่เป็นฟังก์ชันที่ตรวจสอบชนิดของข้อมูลภายในคลาส ซึ่งจะคืนค่าเป็นจริง เมื่อ TypeName ตรงกับชื่อของชนิดข้อมูลในคลาส ส่วนฟังก์ชันสมาชิก Promote() ทำหน้าที่สร้างวัตถุใหม่ในคลาสนั้น แต่มีชนิดของข้อมูลภายในตรงกับชนิดข้อมูลของ Source ดังนั้นเมื่อมีการคำนวณใดๆ ซึ่งกระทำต่อวัตถุ 2 วัตถุในคลาสของ transfn เช่น การบวก การลบ ฯลฯ ภายในฟังก์ชันจะใช้การตรวจสอบชนิดของข้อมูลด้วย IsA() แล้วใช้ฟังก์ชัน Promote() ในการสร้างข้อมูลที่ตรงกันให้

4.5 เนื่องจากอัลกอริทึมที่ใช้ในการคำนวณในการวิเคราะห์และออกแบบด้านเดียวกัน มีความแตกต่างกันตามชนิดของข้อมูล รวมทั้งเพื่อให้สามารถจัดกลุ่มของหน้าที่การทำงานให้กลายเป็นรูปของวัตถุ ดังนั้นในโปรแกรมจึงได้มีการนำเสนอโครงสร้างของ

คลาสแบบ Functor สำหรับใช้ในฟังก์ชันการทำงานที่มีความหมายเหมือนกัน เช่น คลาส time_simulator ซึ่งทำหน้าที่คำนวณหาผลตอบเชิงเวลาจากข้อมูลของระบบ โดยคลาสนี้สามารถเลือกใช้เทคนิคการจำลองผลตอบได้หลายแบบ ขึ้นอยู่กับรูปแบบของข้อมูลระบบ

หลักการของคลาสแบบ Functor คือ การรวมฟังก์ชันการทำงานที่มีความหมายเหมือนกันไว้ภายใต้คลาสเดียวกัน โดยชนิดของข้อมูลและขั้นตอนการคำนวณต่างกัน สำหรับโครงสร้างของคลาสแบบ Functor อิงจากรูป Envelope-Letter โดยส่วนของคอนสตรักเดอร์ (Constructor) ในคลาส Envelope ทำหน้าที่สร้างพอยน์เตอร์ไปยัง Letter Class ที่เหมาะสมให้ตามชนิดของข้อมูลที่ระบุให้ สำหรับภายใน Letter Class เป็นส่วนที่ทำการรวมขั้นตอนในการประมวลผลข้อมูล โดยชนิดของ Letter Class มีจำนวนขึ้นอยู่กับชนิดของข้อมูล จุดเด่นของการใช้โครงสร้างแบบ Functor คือ การสนับสนุนการโปรแกรมตามหน้าที่ (Applicative Programming) รวมทั้งทำให้เกิดคลาสที่มีลักษณะคล้ายกล่องดำ ซึ่งให้ผลลัพธ์ออกมาตามชนิดของข้อมูลที่ป้อนเข้าไปได้ รูป 2-5 แสดงถึงการประยุกต์ใช้คลาส time_simulator ในการจำลองผลตอบ ในกรณีข้อมูลระบบมีความแตกต่างกัน

2.2 โครงสร้างของวัตถุ

การพัฒนาโปรแกรมในวิทยานิพนธ์ฉบับนี้ เป็นการออกแบบและจัดสร้างคลาสต่างๆ ขึ้น เพื่อคลอบคลุมการทำงานที่ต้องการ จากรายละเอียดของการวิเคราะห์และโครงสร้างการทำงานต่างๆ จึงสามารถสร้างคลาสขึ้นมาสำหรับใช้งานในโปรแกรม โดยคลาสที่สร้างขึ้นมา สามารถแบ่งได้ออกเป็น 3 ส่วนหลัก คือ

1. คลาสจัดการระบบใช้งาน Windows เป็นคลาสที่สร้างขึ้นมาสำหรับจัดการและติดต่อกับฟังก์ชันต่างๆของ Windows โดยทำหน้าที่เป็นเสมือนเปลือก ในการเรียกใช้งานฟังก์ชันภายใน Windows โดยรายละเอียดเกี่ยวกับฟังก์ชันต่างๆใน Windows สามารถศึกษาได้จาก (Couger, 1992) สำหรับคลาสในกลุ่มนี้ใช้การเรียกจากคลาสที่มีการนิยามไว้แล้วใน Object Window Library ของบริษัท Borland

2. คลาสของข้อมูลทางคณิตศาสตร์ และการคำนวณแบบต่างๆในทฤษฎีควบคุม เป็นคลาสที่พัฒนาขึ้น เพื่อเป็นโครงสร้างข้อมูลที่สนับสนุนการคำนวณทางด้านคณิตศาสตร์แบบต่างๆ รวมทั้งเป็นการรวมฟังก์ชันในการวิเคราห์และออกแบบตามทฤษฎีควบคุมไว้ในรูปของคลาสด้วย สำหรับคลาสในกลุ่มนี้สามารถแบ่งออกตามลักษณะของข้อมูลและฟังก์ชันในการทำงาน ออกเป็น 3 กลุ่มคือ

2.1 คลาสที่เป็นข้อมูลและการคำนวณพื้นฐานทางคณิตศาสตร์ เป็นคลาสที่ทำหน้าที่เพิ่มประเภทของข้อมูลในด้านคณิตศาสตร์ให้กับโปรแกรม รวมทั้งสนับสนุนการ

คำนวณเชิงเลขพื้นฐาน สำหรับการประกาศและส่วนโค้ดของคลาสในกลุ่มนี้รวมไว้ในไฟล์ matrix.cpp poly.cpp และ model.cpp ตามลำดับ ส่วนโครงสร้างของการสืบทอดของคลาสในกลุ่มนี้อยู่ในรูป 2-6

2.1.1 matrix ทำหน้าที่เป็นข้อมูลแบบเมตริกซ์

2.1.2 polynomial ทำหน้าที่เป็นข้อมูลแบบ polynomial

2.1.3 transfn ทำหน้าที่เป็นข้อมูลแบบฟังก์ชันถ่ายโอน

2.1.4 state_eq ทำหน้าที่เป็นข้อมูลแบบสมการสถานะ

2.2 คลาสที่ทำหน้าที่สนับสนุนกระบวนการวิเคราะห์และออกแบบตามทฤษฎีควบคุม ซึ่งคลอบคลุมงานของการเตรียมข้อมูล การแปลงรูปข้อมูล ให้เหมาะสมกับการวิเคราะห์และออกแบบ ส่วนโครงสร้างของการสืบทอดของคลาสในกลุ่มนี้อยู่ในรูป 2-7

2.2.1 decomposer คลาสนี้ใช้สำหรับการจัดรูปค่าสมาชิกในเมตริกซ์ให้ตรงตามรูปแบบเฉพาะ ซึ่งเป็นขั้นตอนการเตรียมข้อมูลก่อนการนำข้อมูลไปวิเคราะห์หรือออกแบบด้วยอัลกอริทึมต่างๆ

2.2.2 discretizer ใช้ในการแปลงรูปของข้อมูลระหว่างระนาบของสัญญาณแบบต่อเนื่องและไม่ต่อเนื่อง

2.2.3 transformer ใช้ในการแปลงรูปแบบจำลองของระบบระหว่างฟังก์ชันถ่ายโอนและสมการสถานะ

2.3 คลาสที่ทำหน้าที่รวมฟังก์ชันการวิเคราะห์และออกแบบระบบควบคุมเชิงเส้นตามทฤษฎีควบคุมทั้ง 3 แบบ สำหรับส่วนโค้ดของคลาสในกลุ่มนี้ รวมไว้ในไฟล์ analysis.cpp และ design.cpp ตามลำดับ ส่วนโครงสร้างของการสืบทอดคลาสอยู่ในรูป 2-7

2.3.1 analyzer เป็นคลาสที่รวมฟังก์ชันในการวิเคราะห์

2.3.2 designer เป็นคลาสที่รวมฟังก์ชันในการออกแบบ

2.4 คลาสที่ทำหน้าที่เป็นฟังก์ชันการจำลองผลตามทฤษฎีควบคุม เป็นคลาสที่ทำหน้าที่เป็นส่วนจำลองผลตอบของระบบในรูปแบบต่างๆกัน สำหรับการประกาศและส่วนโค้ดของคลาสในกลุ่มนี้ รวมอยู่ในไฟล์ simulate.cpp ส่วนโครงสร้างของการสืบทอดคลาสอยู่ในรูป 2-7

2.2.1 time_simulator ทำหน้าที่เป็นคลาสสำหรับการจำลองผลตอบในเชิงเวลา

2.2.2 freq_simulator ทำหน้าที่เป็นคลาสสำหรับการจำลองผลตอบเชิงความถี่

2.3.3 loci_simulator เป็นคลาสที่คำนวณหาแนวทางการเปลี่ยนแปลงค่า Pole ของระบบวงปิด เมื่อมีการเปลี่ยนแปลงพารามิเตอร์บางตัวของระบบ

3. คลาสจัดการรูปแบบการใช้งาน เป็นคลาสที่พัฒนาขึ้นสำหรับทำหน้าที่เป็นระบบติดต่อผู้ใช้ให้กับโปรแกรม โดยควบคุมและจัดการการเรียกใช้งานฟังก์ชันใน 2 กลุ่มแรก เพื่อให้สามารถใช้งานโปรแกรมได้อย่างง่ายและสะดวก สำหรับคลาสในกลุ่มนี้ แบ่งออกตามหน้าที่รับผิดชอบได้เป็น กลุ่มคือ

3.1 คลาสที่จัดการการรับข้อมูลต่างๆจากผู้ใช้ เป็นคลาสที่ควบคุมระบบจัดการเกี่ยวกับการรับข้อมูลต่างๆจากผู้ใช้ รวมไปถึงการแสดงผลที่ได้จากการวิเคราะห์และออกแบบตามทฤษฎีระบบควบคุม ซึ่งอยู่ในรูปข้อมูลแบบตัวเลข

3.2 คลาสที่จัดการการแสดงผลการทำงาน เป็นคลาสสำหรับควบคุมการแสดงผลตอบที่จำลองได้ในรูปกราฟแบบต่างๆ โดยโครงสร้างการสืบทอดความสามารถของคลาสในกลุ่มนี้อยู่ในรูป 2-8

3.2.1 time_response เป็นคลาสที่ทำหน้าที่ควบคุมการแสดงผลตอบเชิงเวลาของระบบออกทาง Window

3.2.2 nyquist_plot เป็นคลาสที่ทำหน้าที่ควบคุมการแสดงผลตอบเชิงความถี่ในรูปของ Nyquist Plot

3.2.3 bode_plot เป็นคลาสที่ทำหน้าที่ควบคุมการแสดงผลตอบเชิงความถี่ในรูปของ Bode Plot

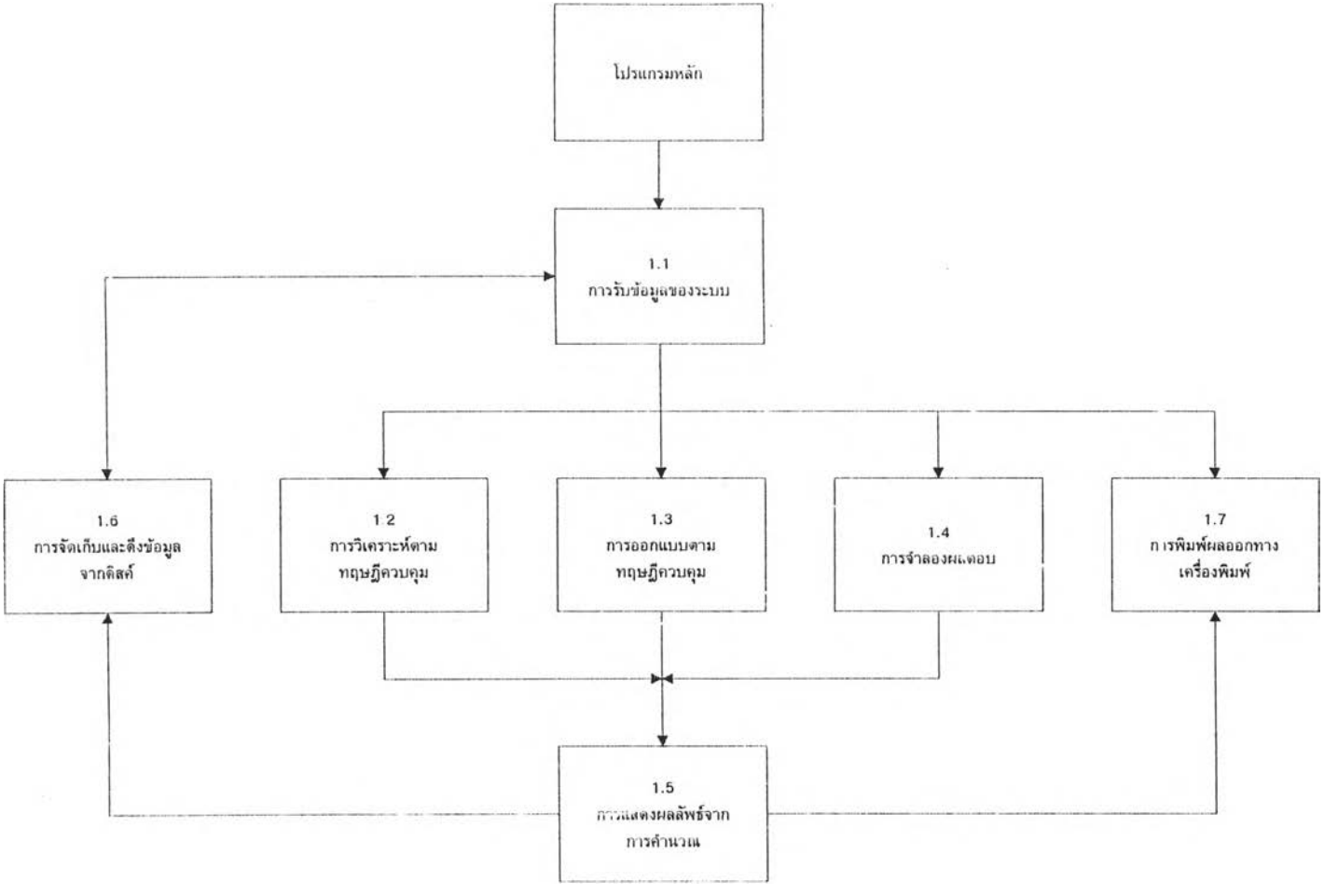
3.2.4 nichols_chart เป็นคลาสซึ่งควบคุมการแสดงผลตอบเชิงความถี่ในรูปของ Nichols Chart

3.2.5 loci_plot เป็นคลาสที่ทำหน้าที่ควบคุมการแสดงแนวทางเดินของตำแหน่งเสาของระบบวงปิด หรือ Root Loci

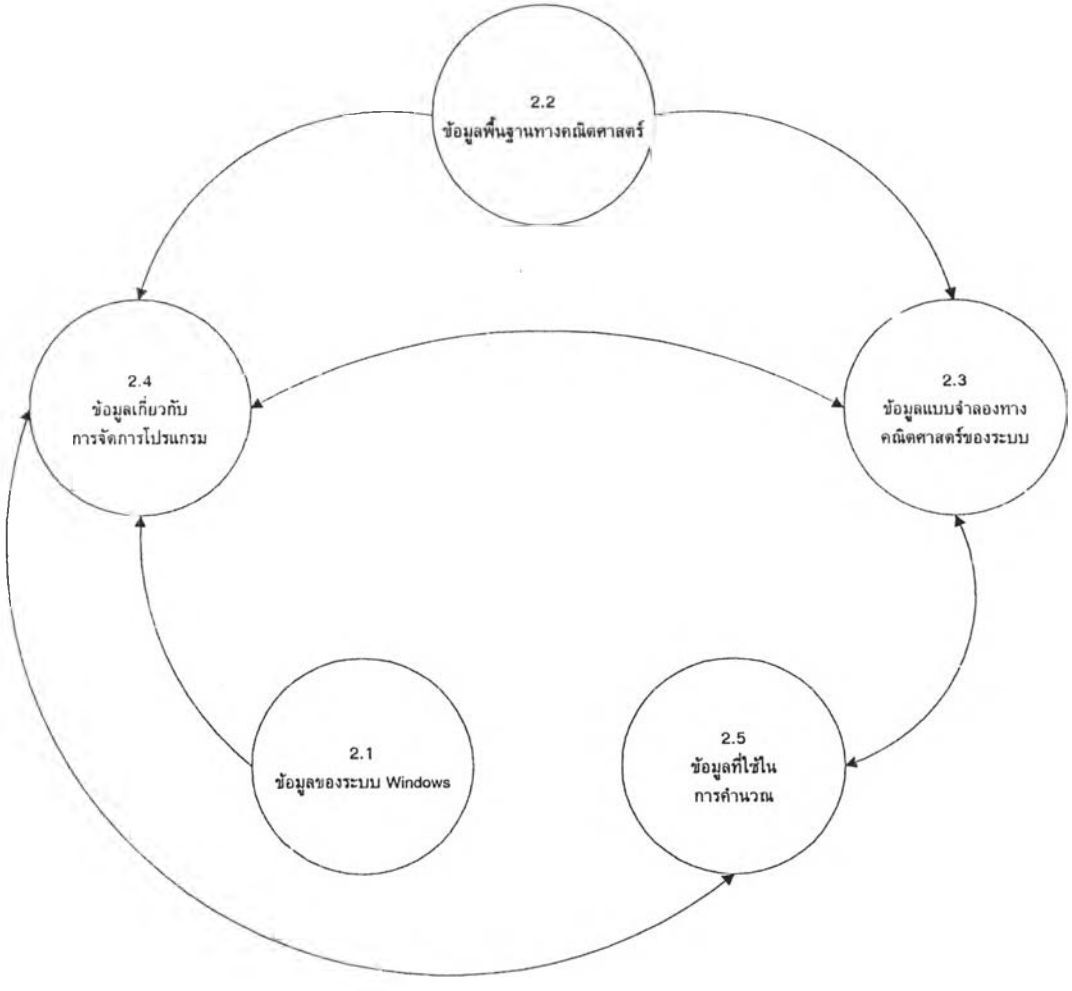
2.3 ลำดับขั้นตอน

เนื่องจากข้อมูลและฟังก์ชันทำงานของโปรแกรมที่พัฒนาขึ้น ถูกรวมไว้อยู่ในคลาสเกือบทั้งหมดแล้ว ในส่วนของโปรแกรมจึงเหลือเพียงการเรียกใช้คลาสเท่านั้น สำหรับในส่วนของโปรแกรมหลัก จึงมีเพียงแค่การสร้างวัตถุประเภทคลาส TModule ซึ่ง TModule จะทำการสร้างส่วนประกอบเริ่มต้นที่จำเป็นให้กับโปรแกรม เช่น Main Window เป็นต้น โดยอัตโนมัติ ซึ่งกลไกภายในคลาสของส่วนประกอบที่สร้างขึ้น จะไปทำการส่วนประกอบที่จำเป็นสำหรับการทำงานของส่วนนั้นเองจนครบในที่สุด จากนั้นระบบ Message Loop ของโปรแกรมเริ่มทำงาน เมื่อ Message Loop เริ่มทำงานแล้ว จะมีการรับ Message ที่ส่งมาจาก Windows แล้วแจกจ่ายไปตาม Window ต่างๆ ภายในแต่ละ Window จะมีฟังก์ชันการประมวลผลที่เตรียมไว้แล้วซึ่งจะทำงานตาม Message ที่เข้ามาโดยอัตโนมัติ โปรแกรมจึงสามารถทำงานต่อไปได้ จนกว่าผู้

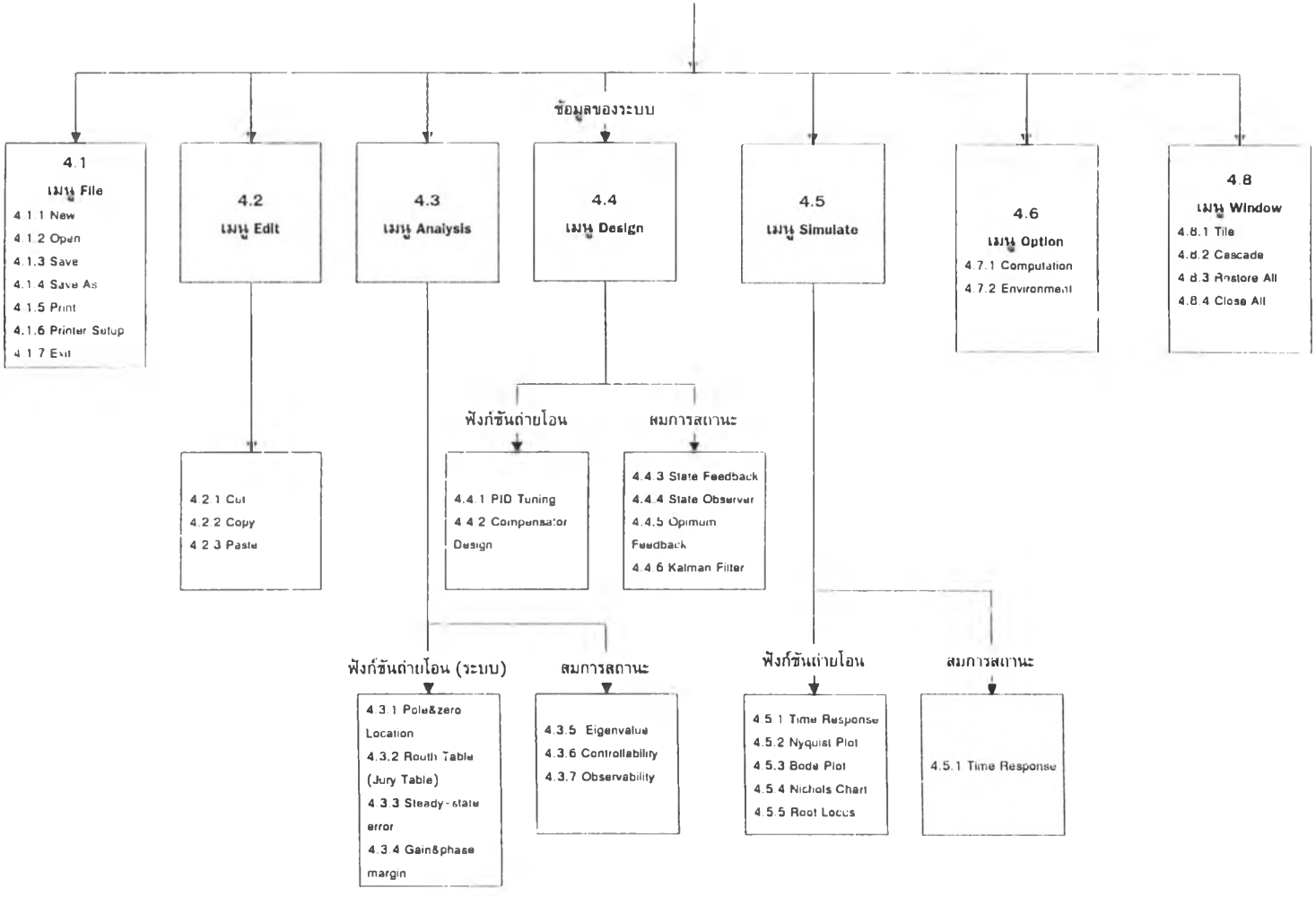
ใช้จะมีคำสั่งให้จบการทำงานของโปรแกรม สำหรับขั้นตอนการทำงานของโปรแกรมแสดงไว้ใน รูป 2-9



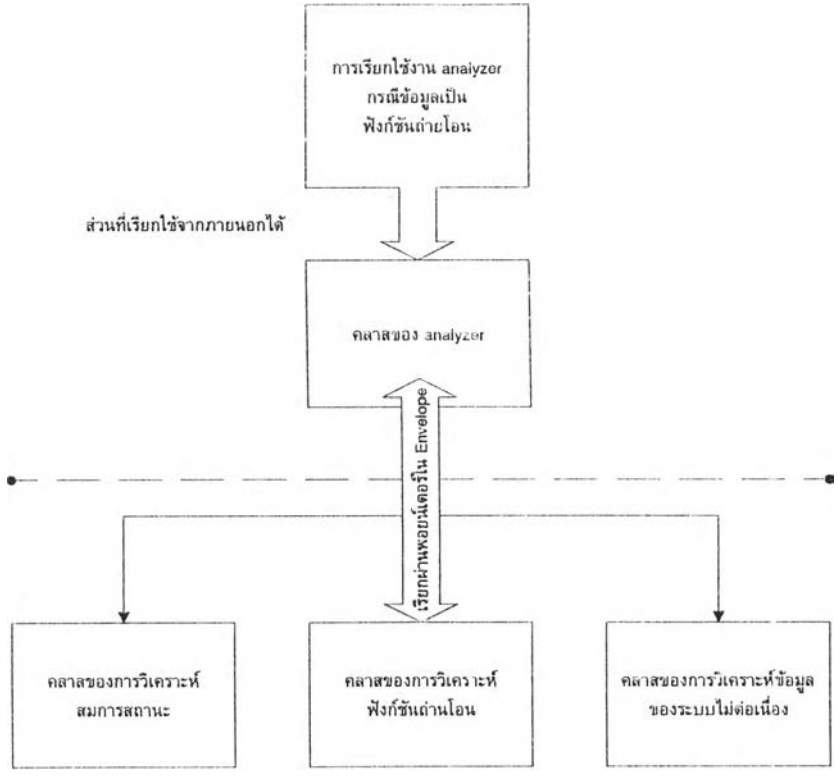
รูป 2-1 หน้าที่การทำงานภายในโปรแกรม



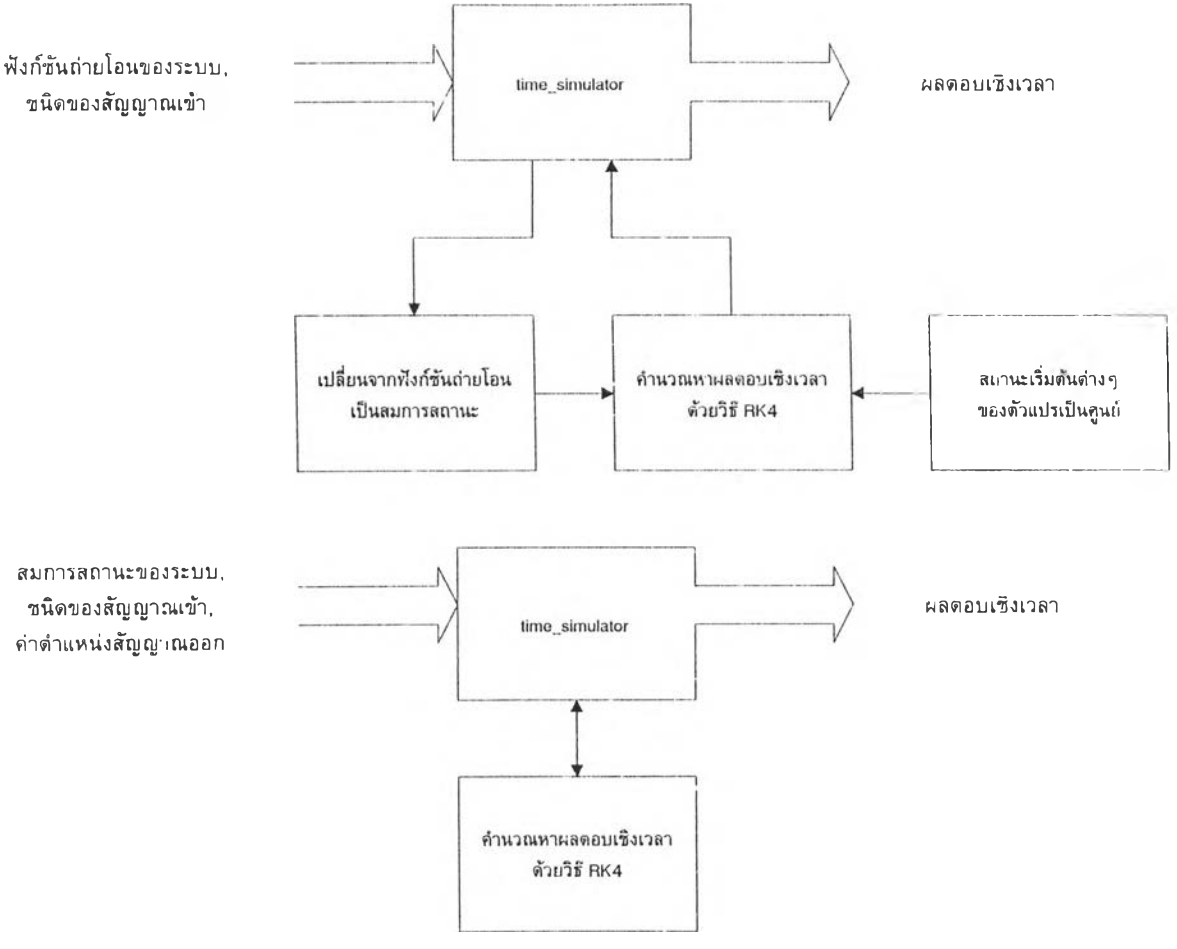
รูป 2-2 ความสัมพันธ์ระหว่างข้อมูลภายในโปรแกรม



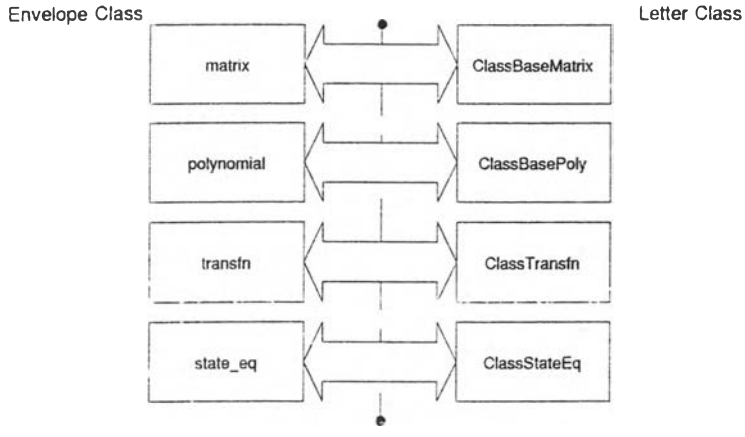
รูป 2-3 โครงสร้างระบบเมนู



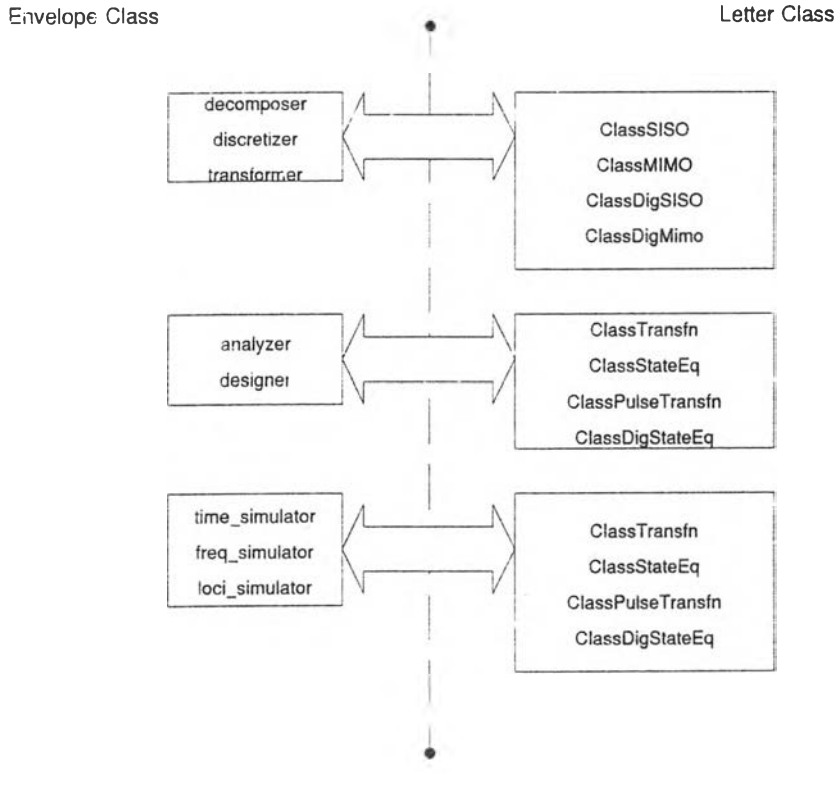
รูป 2-4 โครงสร้างของคลาสแบบ Envelope-Letter



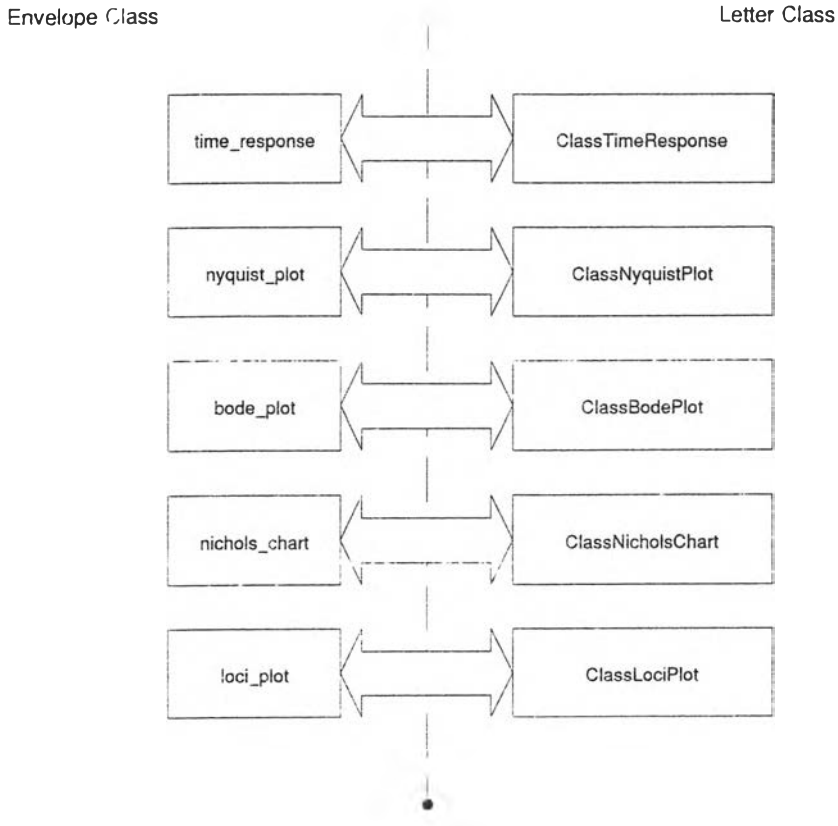
รูป 2-5 การใช้โครงสร้างของคลาสแบบ Functor



รูป 2-6 โครงสร้างของคลาสข้อมูลทางด้านคณิตศาสตร์



รูป 2-7 โครงสร้างของคลาสสำหรับการวิเคราะห์และออกแบบตามทฤษฎีควบคุม



รูป 2-8 โครงสร้างของคลาสสำหรับการแสดงกราฟ

รูป 2-9 การทำงานภายในส่วนโปรแกรมหลัก