44

รายงานผลโครงการวิจัย

พัฒนาคลาสไลบรารีเกี่ยวกับการคำนวณเชิงตัวเลขสำหรับสร้างบทเรียนทางฟิสิกส์ที่เรียนรู้ผ่านเครือข่ายอินเทอร์เน็ต

Developing Class Library Focus on Numerical Computing for Physics E-learning

โดย

รองศาสตราจารย์วัชระ รอดสัมฤทธิ์

ผู้ช่วยศาสตราจารย์จรัส บุณยธรรมา

นายศราวุธ ใจเย็น

นางสาวสุกัญญา นิลม่วง

สาขาวิชาฟิสิกส์ คณะวิทยาศาสตร์และเทคโนโลยี

มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี

(ได้รับเงินงบประมาณประจำปี 2550 หมวดเงินอุดหนุน)

บทคัดย่อ

งานวิจัยสิ่งประดิษฐ์นี้เป็นการสร้างคลังโปรแกรม จัดเก็บอยู่ในรูป class file มีความสามารถในการคำนวณเชิงตัวเลขในหัวข้อต่อไปนี้ : หารากสมการแบบไม่เป็นเชิงเส้น ผลเฉลยของระบบสมการเชิงเส้น ประมาณค่าด้วยวิธีกำลังสองน้อยที่สุด อนุพันธ์และปริพันธ์เบื้องต้น คำตอบสมการเชิงอนุพันธ์แบบสามัญ และคำนวณค่าทางสถิติเบื้องต้น สามารถทำงานได้ภายใต้ระบบปฏิบัติการใด ๆ ของคอมพิวเตอร์ทุกแบบ และผ่านเครือข่ายอินเทอร์เน็ต

การนำไปใช้ประกอบกับการพัฒนาบทเรียน ผู้ใช้เพียงแค่สร้างส่วนที่เป็นเนื้อหาของบทเรียนซึ่งอาจอยู่ในรูปเอกสาร HTML หรือ เป็นกราฟิกเชื่อมต่อและปฏิสัมพันธ์กับผู้เรียน ในส่วนที่ต้องเกี่ยวข้องกับการคำนวณเชิงตัวเลข สามารถนำไฟล์คลาสเหล่านี้ฝังลงในเอกสารบทเรียนแล้วเรียกใช้งานเพื่อการคำนวณได้ทันที

Abstract

This project presents the program library files in class type, which their numerical computing capability cover the following topics: Finding the root of non-linear equation, the solution of system of linear equation, method of least square, basic derivative and integration, the solution of ordinary differential equation and simple statistics. This collection of class files work well under any platform of computer and operating system and via internet network.

To use them in any E-learning lesson, just creating the content document in HTML format or writing an interactive graphic user interface then embeds these class files in documents and invokes them when numerical computing is needed.

สารบัญ

บทที่ หน้า

บทที่ หน้า

สามารถ Download โปรแกรมต้นฉบับที่ปรับปรุงล่าสุดได้ที่ http://203.158.100.140/jmt

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

การจัดทำบทเรียนเพื่อเผยแพร่ความรู้เกี่ยวกับฟิสิกส์ เพื่อให้นักศึกษาหรือผู้สนใจได้มีโอกาสเรียนรู้ด้วยตนเองตามอัธยาศัย โดยผ่านทางเว็บไซต์หรือทำเป็นโปรแกรมสำเร็จรูปไว้ในแผ่น CD จะเป็นที่น่าสนใจและน่าติดตามเพียงใดนั้น ส่วนหนึ่งขึ้นอยู่กับการออกแบบบทเรียนให้มีลักษณะเคลื่อนไหวและสามารถโต้ตอบกับผู้เรียน หรือที่เรียกว่ามีลักษณะพลวัต (dynamic) บทเรียนในลักษณะนี้จะช่วยให้การเรียนรู้เป็นไปอย่างสนุกสนานและไม่น่าเบื่อ ทำให้เกิดความรู้สึกสมจริง สามารถสร้างความเข้าใจได้ง่ายกว่าบทเรียนที่มีเฉพาะตัวหนังสือหรือภาพนิ่ง ถ้าภาพเคลื่อนไหวนั้นสามารถกำหนดมุมมอง สามารถย่นระยะเวลาของเหตุการณ์ให้ช้าหรือเร็ว หรือผู้เรียนสามารถกำหนดเงื่อนไขได้ตามความคิดของผู้เรียนได้ ยิ่งทำให้บทเรียนนั้นน่าสนใจและทำให้เกิดจินตนาการได้อย่างไม่มีขีดจำกัด

เบื้องหลังการสร้างเนื้อหาบทเรียนทางฟิสิกส์ให้มีลักษณะพลวัตนั้น จะต้องอาศัยความรู้ความสามารถทางด้านกราฟิกและการคำนวณเชิงตัวเลข (numerical computation) เป็นอย่างมาก การประมวลผลเชิงตัวเลขของคอมพิวเตอร์นั้นจะใช้เลขฐานสองในการคำนวณ ต้องใช้ระเบียบวิธีการแก้ปัญหาทางคณิตศาสตร์ ซึ่งต่างไปจากวิธีวิเคราะห์ที่ใช้ในชีวิตประจำวัน หรือจากที่ได้เรียนรู้จากห้องเรียน ภาระงานในการสร้างบทเรียน ส่วนหนึ่งจะต้องเสียเวลาไปไม่น้อยกับการเขียนโปรแกรมให้คอมพิวเตอร์คำนวณเชิงตัวเลข ไม่ว่าจะเป็นการหารากสมการแบบไม่เป็นเชิงเส้น การหาผลเฉลยของระบบสมการเชิงเส้น การหาค่าอนุพันธ์และปริพันธ์ การแก้สมการเชิงอนุพันธ์ การหาค่าต่างๆ ทางสถิติ นอกจากนี้ยังต้องคำนึงถึงความถูกต้อง ความแม่นยำในการคำนวณ มีความคลาดเคลื่อนจากการคำนวณน้อยที่สุด ถ้ามีการจัดทำระเบียบวิธีการแก้ปัญหาทางคณิตศาสตร์เหล่านี้มีการเขียนไว้เป็น "คลังโปรแกรม" หรือ Program Library งานสร้างบทเรียนที่มีเนื้อหาทางฟิสิกส์จะลดภาระ และประหยัดเวลาไปได้มาก ผู้สร้างบทเรียนจะได้ทุ่มเทไปทางด้านการจัดทำเนื้อหา และกราฟิกที่จะใช้โต้ตอบกับผู้เรียน (Graphic User Interface, GUI) ส่วนที่เกี่ยวข้องกับการคำนวณในบทเรียน สามารถนำโปรแกรมที่จัดเก็บไว้ในคลังโปรแกรมประกอบใส่ในบทเรียน ทำให้บทเรียนสามารถแสดงการแก้ปัญหาหรือแสดงวิธีคำนวณได้ทันที

1.2 ความมุ่งหมายของการวิจัย

1. เพื่อออกแบบและพัฒนา class library ด้วยภาษา java ที่มีความสามารถในการคำนวณเชิงตัวเลข โดยคัดเลือกจากหัวข้อทางคณิตศาสตร์ที่จะต้องนำไปใช้ในการเรียนการสอนวิชาฟิสิกส์ขั้นพื้นฐาน (กลศาสตร์ ความร้อน คลื่นกล เสียง ไฟฟ้าสถิต ไฟฟ้ากระแส สนามแม่เหล็กไฟฟ้า แสง ทฤษฎีอะตอม และฟิสิกส์นิวเคลียร์เบื้องต้น) และคณิตศาสตร์ที่ต้องใช้ในการเรียนปฏิบัติการฟิสิกส์

2. เพื่อนำ class library ที่พัฒนาได้ไปใช้เป็นส่วนประกอบในบทเรียนทางฟิสิกส์สามารถประมวลผลบนเว็บเพ็จได้

1.3 ความสำคัญของการวิจัย

1. ได้ class library ที่มีความสามารถในการคำนวณปัญหาทางคณิตศาสตร์ ต่าง ๆ เช่น การแก้สมการแบบไม่เป็นเชิงเส้น การแก้สมการระบบสมการเชิงเส้น การหาอนุพันธ์และปริพันธ์ เบื้องต้น การหาคำตอบสมการเชิงอนุพันธ์แบบสามัญ การประมาณค่าโดยวิธีกำลังสองน้อยที่สุด การหาค่าทางสถิติเบื้องต้น ซึ่งเป็นคณิตศาสตร์ที่จำเป็นสำหรับการเรียนวิชาฟิสิกส์พื้นฐาน

2. ผู้สร้างบทเรียนทางฟิสิกส์ (รวมทั้งวิทยาศาสตร์สาขาอื่นๆ) สามารถนำ class library ไปเป็นส่วนประกอบของโปรแกรมหรือบทเรียนได้ทันที ไม่ต้องเสียเวลาสร้างโปรแกรมตรงส่วนที่ต้องใช้ในการคำนวณอีก

1.4 ขอบเขตของการวิจัย

เป็นการพัฒนา class library โดยใช้ภาษา java ในการพัฒนา ขอบเขตความสามารถในการคำนวณเชิงตัวเลขจะครอบคลุมในหัวข้อต่อไปนี้

1. การหารากสมการแบบไม่เป็นเชิงเส้น
2. การหาผลเฉลยของระบบสมการเชิงเส้น
3. การประมาณค่าโดยวิธีกำลังสองน้อยที่สุด
4. การหาอนุพันธ์และปริพันธ์เบื้องต้น
5. การหาคำตอบสมการเชิงอนุพันธ์แบบสามัญ
6. การหาค่าทางสถิติเบื้องต้น

ในการวิจัยนี้จะได้โปรแกรมต้นฉบับ (source code) และไฟล์สกุล .class ซึ่งเป็นไฟล์ที่ได้จากการคอมไพล์ให้อยู่ในรูป byte code ซึ่งพร้อมที่นำไปใช้ประกอบในบทเรียนได้ทันที

1.5 นิยามศัพท์เฉพาะ

คลาสไลบรารี (class library) หมายถึง ไฟล์ที่มีสกุล .class จำนวนหลาย ๆ ไฟล์ ไฟล์เหล่านี้เกิดจากการคอมไพล์โปรแกรมต้นฉบับซึ่งเขียนด้วยภาษาจาวา แต่ละไฟล์มีหน้าที่ในการประมวลผลต่าง ๆ กัน นำมาจัดเก็บไว้ในแหล่งเดียวกันเป็น package เพื่อสะดวกในการอ้างถึงและใช้งาน ไฟล์เหล่านี้สามารถนำไปเป็นส่วนประกอบของโปรแกรมที่ทำงานบน desktop หรือโปรแกรมที่ทำงานผ่านเครือข่ายอินเทอร์เน็ต

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 Java และ Java Applet

จาวาเป็นทั้งภาษาคอมพิวเตอร์และสภาพแวดล้อมที่ทำให้โปรแกรมที่ถูกคอมไพล์แล้วทำงานได้โดยลำพัง (run-time environment) ออกแบบและสร้างขึ้นโดยบริษัทซัน ไมโครซิสเท็ม (Sun Micro system) เมื่อปี ค.ศ. 1996 ปัจจุบัน (ค.ศ. 2007) พัฒนามาถึง version 6 ภาษาจาวาจัดเป็นภาษาเชิงวัตถุ (Object-oriented language) คล้ายกับภาษา C++ แต่ออกแบบให้มีความง่าย และลดช่องทางที่จะทำให้เกิดความผิดพลาดได้ดีกว่า คอมไพเลอร์หรือตัวแปลภาษาจาวาจะแปลงชุดคำสั่งให้อยู่ในรูป byte codes ไม่ว่าคอมพิวเตอร์นั้นจะเป็น CPU ชนิดใด หรือทำงานภายใต้ระบบปฏิบัติการใด ขอเพียงแต่ให้คอมพิวเตอร์เครื่องนั้นมีการติดตั้ง Java Virtual Machine (JVM) อยู่ภายในเครื่องนั้น ก็สามารถทำให้ byte codes นั้นทำงานได้ทันที โดยไม่ต้องติดตั้งคอมไพเลอร์ให้ยุ่งยาก คุณลักษณะของจาวาเช่นนี้ทำให้การเขียนภาษาจาวา สามารถนำโปรแกรมไปใช้งานได้ในทุก platform โดยไม่ต้องคอมไพล์ใหม่หรือแก้ไขโปรแกรมเลย ภาษาจาวาได้จัดเตรียมคลาสต่าง ๆ รวบรวมไว้เรียกว่า Java API (Java Application Program Interface) เป็นประโยชน์สำหรับผู้เขียนโปรแกรมนำไปใช้งานได้ทันที เช่น คลาสที่ประกอบด้วยเมธอดการคำนวณฟังก์ชันตรีโกณมิติ คลาสที่ใช้ในการเขียนและอ่านไฟล์ คลาสที่ใช้ในการเข้าถึงข้อมูลในฐานข้อมูล คลาสที่ใช้แสดงผลทางด้านกราฟิกบนจอภาพ เป็นต้น รายละเอียดของคลาสต่าง ๆ สามารถดูได้จาก Java API documentation ผ่านทางเว็บไซต์ http:// java.sun.com/docs.

ภาษาจาวาจะมีการตรวจสอบขอบเขตของตัวแปรประเภทอะเรย์(array)ในระหว่างที่โปรแกรมกำลังทำงาน เพื่อป้องกันมิให้มีการใช้หน่วยความจำบริเวณอื่นนอกขอบเขตของตัวแปร ซึ่งอาจทำให้โปรแกรมหยุดทำงานกลางคันได้ นอกจากนี้ยังมีการจัดการหน่วยความจำให้สามารถใช้งานได้อย่างมีประสิทธิภาพอย่างอัตโนมัติที่เรียกว่า garbage collector จะเรียกคืนหน่วยความจำจากตัวแปรที่ไม่ได้ถูกใช้งานแล้ว ภาษาจาวามีวิธีจัดการกับความผิดพลาดโดยที่ไม่ทำให้โปรแกรมหยุดการทำงาน กลไกการตรวจความผิดพลาดนี้เรียกว่า exception handling โดยที่ผู้เขียนโปรแกรมภาษาจาวาไม่ต้องเสียเวลาและไม่ต้องคอยระมัดระวังหรือเขียนส่วนใดส่วนหนึ่งของโปรแกรมเพื่อดักจับความผิดพลาดเหล่านี้เลย ดังนั้น ซอฟต์แวร์ที่เขียนด้วยภาษาจาวาจึงมีความน่าเชื่อถือมากกว่าเขียนด้วยภาษาคอมพิวเตอร์อื่น

ข้อแตกต่างของภาษาจาวาที่เห็นได้ชัดเจนเมื่อเทียบกับภาษาอื่นคือ ภาษาจาวาเกิดขึ้นมาพร้อม ๆ กับการใช้งานอินเทอร์เน็ตที่เริ่มแพร่หลาย จาวาจึงถูกออกแบบมาให้สามารถทำงานภายใต้บราวเซอร์ มีลักษณะที่เรียกว่า applet แอพเพล็ตเป็นโปรแกรมเล็ก ๆ ถูกคอมไพล์ให้อยู่ในรูป class file เก็บไว้ที่ฝั่งแม่ข่าย เมื่อผู้ใช้เรียกใช้งานเอกสาร html ผ่านทางอินเทอร์เน็ตซึ่งในเอกสารนั้นมีคำสั่งเรียกใช้งานแอพเพล็ต แอพเพล็ตจะถูก download ไปทำงานในบราวเซอร์ของเครื่องคอมพิวเตอร์ของผู้ใช้นั้น บราวเซอร์ของเครื่องลูกข่ายจะทำงานร่วมกับ JVM ทำให้แอพเพล็ตสามารถทำงานได้ นอกจากนี้ภาษาจาวายังออกแบบ servlet ซึ่งเป็นโปรแกรมที่ทำงานที่ฝั่งแม่ข่าย แล้วแสดงผลลัพธ์ที่ได้เป็นเอกสาร html ส่งไปยังลูกข่ายได้อีกด้วย

เพราะแอพเพล็ตจะต้องถูกแจกจ่ายไปตามเครือข่าย จึงต้องมีข้อจำกัดในการสร้างแอพเพล็ตเพื่อมิให้มีผู้นำแอพเพล็ตไปใช้ในเชิงไม่สร้างสรรค์ เช่นลักลอบดู หรือทำลายข้อมูลหรือลบแฟ้มข้อมูลของคอมพิวเตอร์ที่เชื่อมต่ออยู่ในระบบเครือข่าย การสร้างแอพเพล็ตจึงมีข้อห้ามหลายอย่างเพื่อความปลอดภัย เช่น ห้ามเขียน อ่าน เปลี่ยนชื่อ เปลี่ยนวันเวลา หรือลบไฟล์ ห้ามตรวจสอบไดเรกทอรีหรือสร้างหรือดูไดเรกทอรี ห้ามใช้ฟังก์ชันที่เกี่ยวข้องกับระบบปฏิบัติการ เช่น System.exit(), Runtime.exec ( ) ไม่สามารถติดต่อกับคอมพิวเตอร์เครื่องอื่น ยกเว้นเครื่องที่ส่ง applet นั้นมา และมีข้อจำกัดในการใช้ package ที่เกี่ยวกับการรักษาความปลอดภัย

2.2 การนำจาวาไปใช้ในการคำนวณเชิงตัวเลขและบทเรียนทางฟิสิกส์

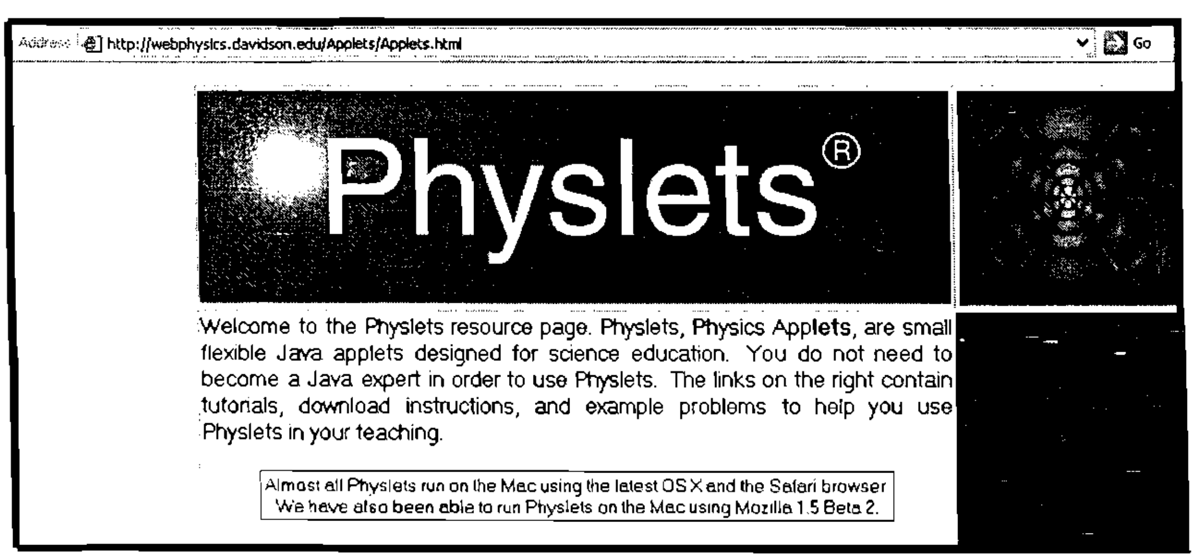
Luján, Mikel. Freeman, T. L., Gurd, John R. ได้ออกแบบโปรแกรมสำหรับการคำนวณพีชคณิตเชิงเส้น เรียกโปรแกรมชุดนี้ว่า OoLALA : an Object oriented analysis and design of numerical linear algebra. ได้เสนอการออกแบบที่เน้นไปทางวิธีปฏิบัติการและการจัดการเมตริกซ์ พบว่าสามารถขจัดข้อบกพร่องซึ่งเคยมีจากโปรแกรมที่มีผู้ทำไว้ในรุ่นก่อนๆ สามารถคำนวณได้ถูกต้องและสอดคล้องกับวิธีที่ได้จากการวิเคราะห์ ผู้วิจัยได้แสดงสมรรถนะการคำนวณเมื่อนำไปใช้กับภาษาจาวาให้เห็นด้วยว่าสามารถทำงานได้อย่างมีประสิทธิภาพ

Moreiera, Jose E. ได้ศึกษาการใช้ array แบบหลายมิติในภาษาจาวา โดยเปรียบเทียบการใช้อะเรย์แบบหลายมิติจาก class library พบว่าผลลัพธ์ของการคำนวณให้ความแม่นยำและมีประสิทธิภาพ แต่ไม่สะดวกในการเข้าถึงสมาชิกอะเรย์แบบหลายมิติเหล่านั้น เว้นแต่จะเพิ่มไวยากรณ์เข้าไปในตัวภาษาเพื่อให้ใช้งานอะเรย์แบบหลายมิติสะดวกยิ่งขึ้น ผู้วิจัยยังศึกษาการใช้อะเรย์โดยอาศัยคอมไพเลอร์ JVM (Java Virtual Machine) โดยให้คอมไพเลอร์มองอะเรย์หลายมิติในลักษณะเป็น array of array พบว่าให้ผลลัพธ์จากการคำนวณที่ถูกต้อง แต่ก็ไม่ทำให้วิธีการเขียนโปรแกรมเชิงคำนวณกะทัดรัดหรือสะดวกกว่าเดิมแต่อย่างใด ผู้วิจัยได้ทดลองเพิ่ม

ภาษาจาวา โดยสร้างคำสั่งที่เกี่ยวข้องกับ array หลายมิติให้เป็น byte code ซึ่งจะทำให้การเรียกใช้งานได้สะดวกกว่า โดยไม่ต้องไปเรียกใช้ class library งานวิจัยนี้ยังได้ศึกษาผลกระทบการใช้งานอะเรย์ในรูปแบบต่าง ๆ ขณะที่มีการประมวลผล ข้อกำหนดของภาษาและ ตัว JVM การนำไปใช้งาน และสมรรถนะของการคำนวณที่ได้จากแต่ละวิธี

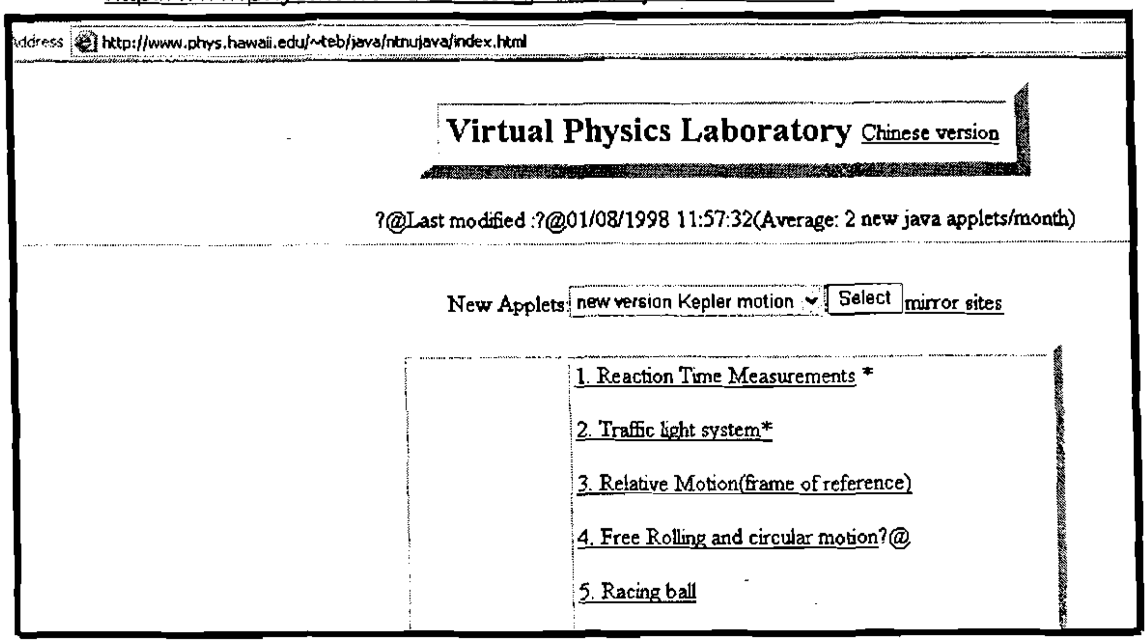
Bishop, Judith และ Bishop, Nigel ได้ศึกษาเปรียบเทียบการเขียนโปรแกรมสำหรับงานทางวิทยาศาสตร์ด้วยภาษาจาวา กับภาษาคอมพิวเตอร์ที่นิยมใช้ดั้งเดิม ได้แก่ ภาษาฟอร์แทรน ซี หรือ ปาสคาล ซึ่งใช้แนวคิดของ procedural language เมื่อนักวิทยาศาสตร์และวิศวกรได้เรียนรู้แนวคิดแบบ object oriented พบว่าสามารถเขียนโปรแกรมให้ใช้งานได้เหมือนกับภาษาคอมพิวเตอร์ที่ใช้อยู่เดิม และมีข้อได้เปรียบคือสามารถเขียนโปรแกรมคำนวณผ่านระบบเครือข่าย หรือโปรแกรมคำนวณคำสั่งในลักษณะคู่ขนานกันได้โดยไม่ต้องรอการประมวลผลแบบเรียงลำดับ ผู้ศึกษาได้รวบรวมปัญหาที่พบในทางวิทยาศาสตร์ 50 ปัญหา เพื่อใช้ทดสอบโปรแกรมที่เขียนด้วยภาษาจาวา และรวบรวมคลาสที่จำเป็นสำหรับการเรียนการสอนภาษาจาวาและงานคำนวณทางวิทยาศาสตร์

Christian, W. และ Belloni, M. ได้จัดทำแอพเพล็ตสำหรับใช้ประกอบกับตำราทางด้านฟิสิกส์ ที่เขาเขียนขึ้น โดยบรรจุไว้ในแผ่นCD ซึ่งผู้เรียนสามารถใช้งานแอพเพล็ตได้โดยไม่ต้องเชื่อมต่ออินเทอร์เน็ต เรียก applet เหล่านี้ว่า Physlet (Physics applets) ใช้ประกอบกับบทเรียนเรื่องกลศาสตร์ ของไหล คลื่น เทอร์โมไดนามิกส์ แม่เหล็กไฟฟ้า ไฟฟ้ากระแส และแสง สามารถดูตัวอย่าง Physlet ของเขาได้ที่เว็บไซต์ http://webphysics.davidson.edu/Applets/Applets.html

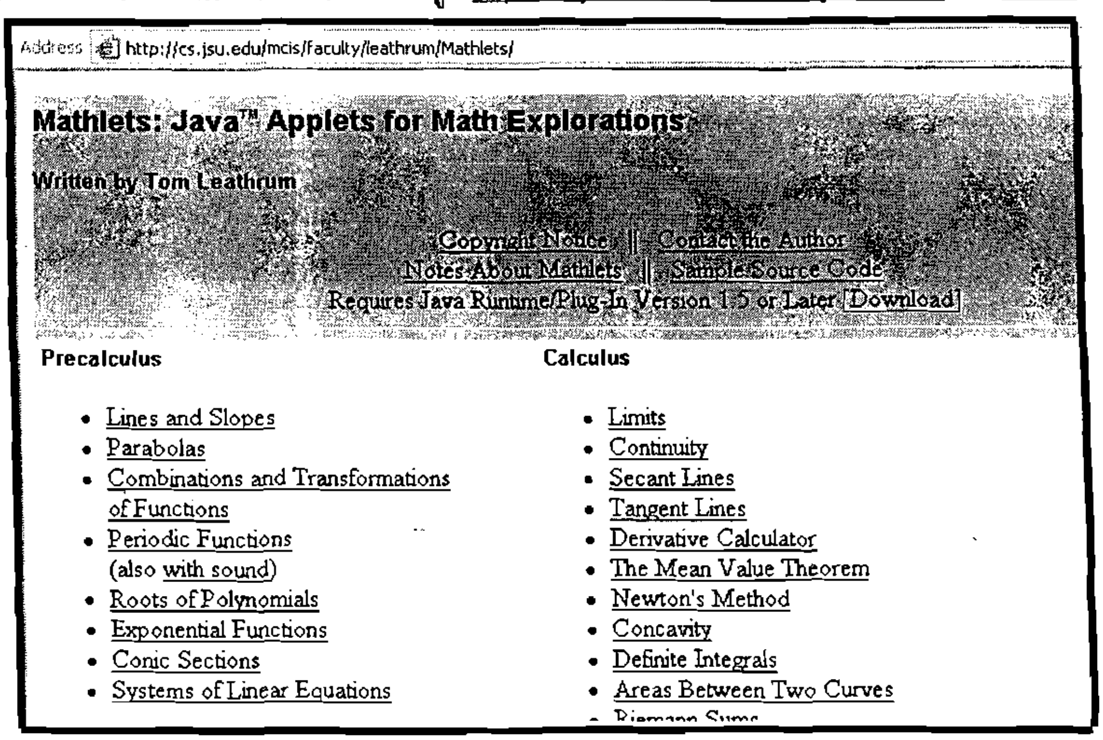


Fu-Kwun-Hwang นักฟิสิกส์ชาวไต้หวัน สร้างแอพเพล็ตเพื่อจำลองสถานการณ์และปรากฏการณ์ต่าง ๆ ทางฟิสิกส์ ได้แก่ จลศาสตร์ พลศาสตร์ คลื่น ความร้อน แม่เหล็กไฟฟ้า และแสง เปิดโอกาสให้ผู้เรียนสามารถ download แอพเพล็ตไปใช้แบบ offline ได้แต่ต้องลงทะเบียนผ่านเว็บไซต์ก่อน สามารถดูผลงานของเขาได้ที่

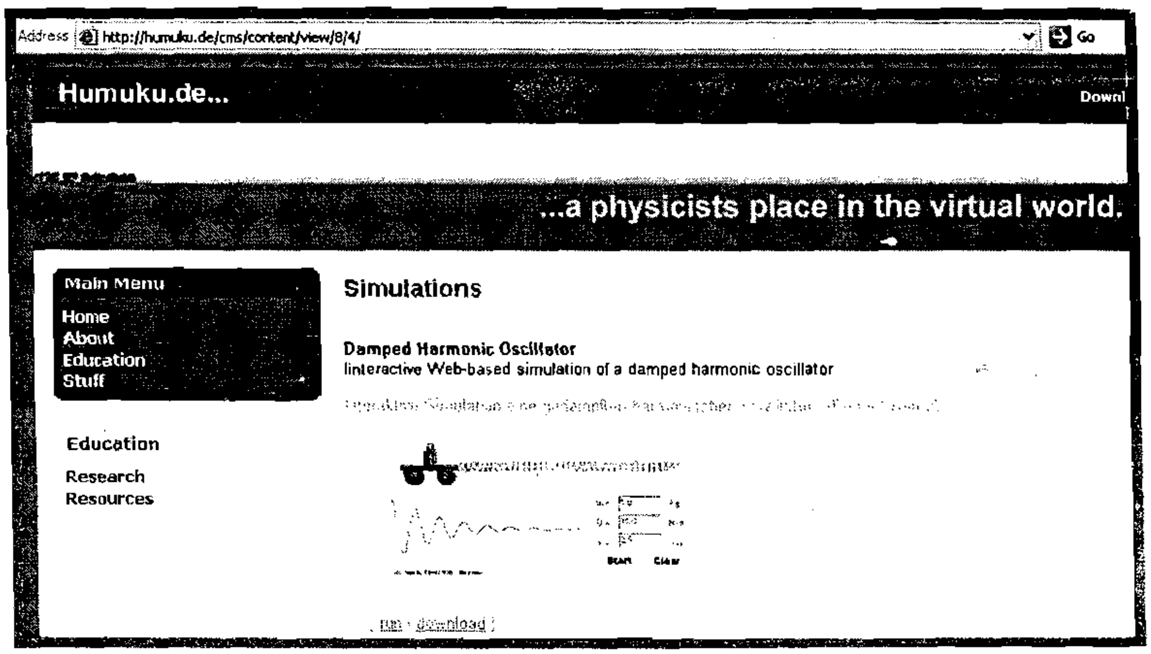
http://www.phys.hawaii.edu/~teb/java/ntnujava/index.html



Leathrum, T. ศาสตราจารย์ในมหาวิทยาลัย Jacksonwile State ได้เขียนแอพเพล็ต เพื่อการเรียนรู้ทางคณิตศาสตร์ เกี่ยวกับเรขาคณิตวิเคราะห์ สามารถนำสมการเขียนเป็นกราฟ โดยแสดงผลผ่านทางอินเทอร์เน็ต เว็บไซต์อยู่ที่ http://cs.jsu.edu/mcis/faculty/leathrum/Mathlets/



เว็บไซต์ www.humuku.de/htm/deucation/software.html ได้พัฒนาการจำลองสถานการณ์ทางฟิสิกส์ผ่านเว็บ (web base physics simulations (java-applet)) ในหัวข้อ การสั่นแบบมีการหน่วง ปรากฏการณ์ดอปเปลอร์ การเคลื่อนที่ของอนุภาค 4 ชิ้นโดยใช้กฎของเคปเลอร์ แสงเชิงเรขาคณิต การเคลื่อนที่วิถีโค้ง และแบบจำลองการคลื่นน้ำ แอพเพล็ตเหล่านี้ผู้เรียนสามารถ download ไปใช้ได้ในระยะเวลาจำกัด ถ้าต้องการใช้งานได้โดยไม่หมดอายุจะต้องซื้อแอพเพล็ตเหล่านี้



ต่อไปนี้เป็นเว็บไซต์ของสถาบันการศึกษาต่าง ๆ ที่มีการแทรกแอพเพล็ตในบทความหรือบทเรียนทางฟิสิกส์บ้างประปราย

http://jersey.Uoregon.edu/vlab เป็นเว็บที่รวมแอพเพล็ตเกี่ยวกับกลศาสตร์ เทอร์โมไดนามิกส์ ดาราศาสตร์ และวิทยาศาสตร์สิ่งแวดล้อม ออกแบบเว็บไซต์ได้สวยงามและแปลกตากว่าเว็บไซต์อื่น ๆ ในแต่ละแอพเพล็ตที่ปรากฏในแต่ละเว็บเพ็จ จะมีนาฬิกาจับเวลาและเครื่องคิดเลขอำนวยความสะดวกแก่ผู้เรียนด้วย

http://ist-socrates.berkeley.edu/~cywon สร้างแอพเพล็ตแสดงการเคลื่อนที่ของคลื่น การสั่นของโครงผลึก แบบจำลองอะตอมของไฮโดรเจน การเลี้ยวเบนของแสงแบบต่าง ๆ ผู้เรียนสามารถโต้ตอบกับบทเรียนโดยใช้เมาส์ลาก scroll bar หรือคลิกที่ปุ่มต่าง ๆ

http://www.surendranath.org/applet.html พัฒนาแอพเพล็ตทางคณิตศาสตร์และฟิสิกส์ ได้แก่ จลศาสตร์ พลศาสตร์ การสั่นแกว่ง คลื่น ความร้อน และ แสง

http://www.falstad.com/mathphysics.html เป็นเว็บไซต์ที่นำ java applet ประกอบกับบทความทางฟิสิกส์ ในหัวข้อ คลื่นและการสั่นสะเทือน เสียง การประมวลผลสัญญาณ พลศาสตร์ของแม่เหล็กไฟฟ้า กลศาสตร์ควอนตัม และvector calculus

http://www.physics.uoguelph.ca/www_physics/ เป็นเว็บไซต์ของภาควิชาฟิสิกส์ มหาวิทยาลัย Guelph ประเทศแคนาดา ได้พัฒนาแอพเพล็ตเกี่ยวกับกลศาสตร์ โดยเฉพาะเรื่องการเคลื่อนที่วัตถุ ผู้เรียนสามารถ download โปรแกรมต้นฉบับ (source code) มาศึกษาหรือดัดแปลงได้

การที่จะนำ applet ที่มีผู้พัฒนาไว้แล้ว มาใช้ประกอบในบทเรียนที่จัดทำนั้นย่อมทำได้ ซึ่ง applet ที่ทำการเผยแพร่แจกจ่ายจะอยู่ในรูปของคลาสไฟล์ ผู้ใช้จะต้องรู้ข้อกำหนดและเงื่อนไขการใช้งานตามที่ผู้เขียนโปรแกรมกำหนดไว้ ข้อดีคือช่วยลดเวลาการพัฒนาไม่ต้องเขียนโปรแกรมการคำนวณเชิงตัวเลขในส่วนนี้ อย่างไรก็ตามคลาสไฟล์ที่มีผู้เขียนแจกจ่ายไว้ ส่วนใหญ่จะไม่เปิดเผยโปรแกรมต้นฉบับ หรือ source code หรือถ้าจะเปิดเผย ก็มักจะเปิดเผยเพียงบางส่วน หรือถ้าต้องการต้นฉบับครบถ้วนต้องจ่ายเงินซื้อ ทำให้ผู้นำมาใช้งานขาดความยืดหยุ่น ไม่สามารถนำโปรแกรมมาดัดแปลงแก้ไขให้เหมาะกับบทเรียนของผู้ใช้งานได้ คลาสไฟล์ที่มีการเผยแพร่อาจมีเพียงการแก้ปัญหาคณิตศาสตร์บางหัวข้อ ไม่ครอบคลุมเนื้อหาหรือปัญหาที่เราต้องการหาคำตอบ การพัฒนาคลาสไฟล์ขึ้นใช้เอง ทำให้ได้ระเบียบวิธีการคำนวณเชิงตัวเลขตามหัวข้อที่เราต้องการ สามารถแก้ไข เปลี่ยนแปลงได้ถึงระดับโปรแกรมต้นฉบับ ทำให้โปรแกรมสามารถคำนวณและแสดงผลตามรูปแบบของบทเรียนที่ต้องการให้เป็น โปรแกรมการคำนวณเชิงตัวเลขที่พัฒนาขึ้นมาเหล่านี้สามารถนำไปขยายผล ประกอบกับโปรแกรมส่วนอื่น ๆ เพื่อสร้างเป็นซอฟต์แวร์ใหม่ขึ้นมาใช้งานด้านอื่น ๆ ที่เกี่ยวข้องกับการคำนวณได้อีกด้วย

บทที่ 3
วิธีการดำเนินการวิจัย

การพัฒนาคลาสไลบรารี เกี่ยวกับการคำนวณเชิงตัวเลขสำหรับสร้างบทเรียนทางฟิสิกส์ที่เรียนรู้ผ่านเครือข่ายอินเทอร์เน็ต เป็นการเขียนโปรแกรมโดยใช้ภาษาจาวา นำโปรแกรมต้นฉบับ (Source code) ที่พัฒนาได้ไปทำการเปลี่ยนให้อยู่ในรูป byte codes และจัดเก็บไว้ใน class file โดยใช้คอมไพเลอร์ของภาษาจาวา

เครื่องมือที่ใช้ในการทำวิจัย

1. เครื่องคอมพิวเตอร์สำหรับใช้เขียนโปรแกรม และทดสอบโปรแกรม
2. ซอฟต์แวร์ Java SDK 1.6 (java Standard Development Kit) สามารถ download ได้โดยไม่เสียค่าใช้จ่ายจาก http:/java./sun.com
3. ซอฟต์แวร์อื่น ๆ ที่เป็นเครื่องมือในการพัฒนาโปรแกรม ส่วนใหญ่จะเป็น Open source สามารถ download มาใช้ได้โดยไม่เสียค่าใช้จ่ายเช่นเดียวกัน ได้แก่ Eclipse, Jcreator light หรือ Notepad ที่ติดมาพร้อมกับระบบปฏิบัติการวินโดว์
4. เครื่องคอมพิวเตอร์ทำหน้าที่เป็น server หมายเลข IP คือ 203.158.100.100 ใช้โปรแกรม IIS 6 (Internet Information System) เป็น web server ภายใต้ระบบปฏิบัติการวินโดว์ 2003 สำหรับทดสอบการทำงานของโปรแกรมผ่านเครือข่ายอินเทอร์เน็ต

การวิจัยนี้ได้รับงบประมาณแผ่นดินประจำปี. 2550 การใช้เงินงบประมาณจะเริ่มตั้งแต่ตุลาคม 2549 สิ้นสุด กันยายน 2550 ผู้วิจัยได้วางแผนการพัฒนา โดยมีแผนงานเป็นลำดับดังนี้

ขั้นตอนที่ 1 ศึกษาค้นคว้าระเบียบวิธีการคำนวณเชิงตัวเลขตามหัวข้อที่กำหนดไว้ในขอบเขตการวิจัย ดังนี้

1. การหารากสมการแบบไม่เป็นเชิงเส้น
2. การหาผลเฉลยของระบบสมการเชิงเส้น
3. การประมาณค่าโดยวิธีกำลังสองน้อยที่สุด
4. การหาอนุพันธ์และปริพันธ์เบื้องต้น
5. การหาคำตอบสมการเชิงอนุพันธ์แบบสามัญ
6. การหาค่าทางสถิติเบื้องต้น

โดยศึกษาจากหนังสือ บทความ งานวิจัย ที่เขียนไว้เป็น flow chart diagram หรือบางเล่มเป็น pseudo code หรือ source code อยู่ในรูปโปรแกรมคอมพิวเตอร์ภาษาต่าง ๆ เช่น ภาษาเบสิก ภาษาฟอร์แทรน ภาษาปาสคาล ภาษา C และภาษา C++ เนื่องจากภาษาคอมพิวเตอร์เหล่านี้เป็นภาษาแบบเชิงโครงสร้าง procedural language (ยกเว้นภาษา C++ ซึ่งจัดเป็นภาษาเชิงวัตถุ) ไม่สามารถแปลงโปรแกรมต้นฉบับเหล่านี้ให้เป็นภาษาจาวาโดยตรง ผู้วิจัยจึงได้แต่เพียงศึกษาข้อดีข้อเสียของระเบียบวิธีการคำนวณแบบต่าง ๆ ทดลองให้โปรแกรมเหล่านี้ทำงานแล้วดูความคลาดเคลื่อน ความแม่นยำ และความเที่ยงตรงของโปรแกรมที่ได้

ในการแก้ปัญหาทางคณิตศาสตร์เรื่องเดียวกัน สามารถมีระเบียบวิธีการแก้ปัญหาได้หลายหลายวิธี เช่น การหารากสมการแบบไม่เป็นเชิงเส้น สามารถใช้วิธีแบ่งครึ่ง (bisection) หรือวิธีวางตำแหน่งคำตอบผิดที่ (method of false position) หรือวิธีนิวตัน- ราฟสัน (Newton-Raphson method หรือวิธีเซแคนต์ (Secant method) หรือวิธีมุลเลอร์ (Muller method) หรือวิธีลาแกร์ (Laguerre method) ผู้วิจัยจะต้องคัดเลือกระเบียบวิธีที่ใช้งานได้ไม่ยุ่งยากซับซ้อน ใช้ชุดคำสั่งน้อยที่สุด ใช้เวลาประมวลผลน้อย และสิ้นเปลืองทรัพยากร (ได้แก่ หน่วยความจำ หรือพื้นที่เก็บโปรแกรม) น้อยที่สุด แต่ให้คำตอบถูกต้องแม่นยำ และมีความคลาดเคลื่อนเกิดขึ้นน้อยที่สุด

นอกจากศึกษาข้อดีข้อเสียแล้ว ยังต้องคำนึงถึงเงื่อนไขการใช้งาน เช่นวิธี bisection หารากสมการได้เฉพาะที่เป็นจำนวนจริง ไม่สามารถหารากสมการที่เป็นจำนวนจินตภาพได้ วิธีนิวตัน-ราฟสันต้องป้อนข้อมูลเริ่มต้น และต้องใช้อนุพันธ์ของฟังก์ชันมาใช้ในการคำนวณด้วยระเบียบวิธีการแก้ปัญหาบางวิธีเหมาะกับปัญหาบางกรณี เช่นการหาผลเฉลยของระบบสมการเชิงเส้นโดยวิธี Gauss elimination ถ้าใช้กับสมการที่สัมประสิทธิ์ของตัวแปรมีขนาดต่างกันมาก ๆ จะเกิดความคลาดเคลื่อนสะสมจำนวนมากในระหว่างการคำนวณ ผลก็คือจะได้คำตอบสุดท้ายห่างไกลจากค่าแท้จริง ผู้วิจัยจำเป็นต้องศึกษาจุดอ่อนจุดแข็งของแต่ละวิธีให้ถ่องแท้ ถ้าจำเป็นต้องใช้วิธีการดังกล่าวจำเป็นต้องหาทางแก้ไขไม่ให้ความคลาดเคลื่อนสะสมเกิดขึ้นขณะประมวลผล

ขั้นตอนที่ 2 ออกแบบระเบียบวิธีการคำนวณ โดยคัดเลือกจากระเบียบวิธีการแก้ปัญหาบางวิธีจากขั้นตอนที่ 1 ในการแก้ปัญหาคณิตศาสตร์หัวข้อเดียวกัน แต่มีระเบียบวิธีแก้ปัญหาหลายวิธี ผู้วิจัยต้องศึกษาขั้นตอนการแก้ปัญหาของแต่ละวิธีอย่างละเอียด ศึกษาข้อมูลที่ต้องป้อนเข้าทาง input และผลลัพธ์ที่จะแสดงออกมาทาง output ตัวแปรต่าง ๆ ที่ต้องใช้ระหว่างการทำงานของโปรแกรม จำนวนเมธอดและลักษณะการทำงานของแต่ละเมธอด จากนั้นจึงออกแบบคลาสแม่ (super class หรือ base class) ซึ่งประกอบด้วยตัวแปรคลาส (instance variable) และเมธอดที่ใช้ได้ครอบคลุมกับทุก ๆ ระเบียบวิธี คลาสที่ทำหน้าที่สำหรับแก้ปัญหาของแต่ละระเบียบวิธีจะเป็นคลาสลูกที่สืบทอดคุณสมบัติต่าง ๆ ครบถ้วนมาจากคลาสแม่ จากนั้น

เพิ่ม instance variable และเมธอดที่ใช้เฉพาะกับระเบียบวิธีนั้น ๆ ลงไปในคลาสลูกนี้ การเขียนโปรแกรมในลักษณะ Object oriented นี้จะช่วยประหยัดเวลาการพัฒนาโปรแกรม เพราะจะใช้คุณสมบัติการสืบทอดของคลาสแม่ส่งต่อมายังคลาสลูก ไม่ต้องเขียนโปรแกรมซ้ำในส่วนนี้อีก เมธอดชื่อเดียวกันที่สืบทอดมายังคลาสลูก สามารถเพิ่มเติมหรือเปลี่ยนแปลงให้แตกต่างไปจากคลาสแม่ได้ โดยไม่กระทบกระเทือนการทำงานของเมธอดนั้นที่อยู่ในคลาสแม่เลย เราเรียกลักษณะการเขียนโปรแกรมในลักษณะนี้ว่าเป็น polymorphism

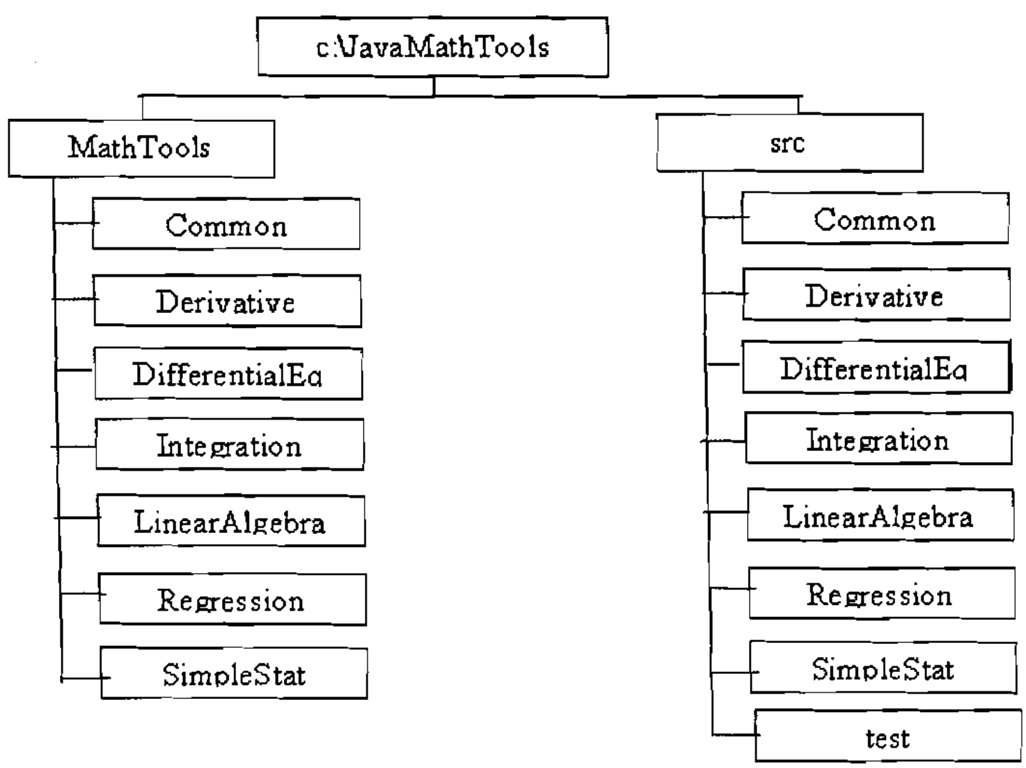
สิ่งที่สำคัญอย่างยิ่งอีกอย่างหนึ่งในการออกแบบก็คือผู้วิจัยจะต้องคาดการณ์ถึงความผิดพลาดของโปรแกรม(exception) ที่อาจเกิดขึ้นได้ในระหว่างการทำงาน โปรแกรมที่ดีจะต้องมีการดักจับความผิดพลาดที่อาจจะเกิดขึ้นนี้ ไม่ให้โปรแกรมเกิดการ crash ในระหว่างการทำงาน ซึ่งอาจทำให้ผู้ใช้โปรแกรมเกิดความงุนงง ถ้าความผิดพลาดของโปรแกรมเกิดจากการใช้งานอย่างไม่ถูกต้องของผู้ใช้โปรแกรม เช่น ป้อนตัวอักษรเข้าไปในโปรแกรมแทนที่จะเป็นตัวเลข ต้องมีการแจ้งข้อความให้ผู้ใช้ทราบถึงสาเหตุที่เกิดข้อผิดพลาด

ตัวอย่างการออกแบบ การหารากสมการแบบไม่เป็นเชิงเส้น ซึ่งผู้วิจัยได้คัดเลือกระเบียบวิธีแก้ปัญหาที่เหมาะสมไว้ถึง 4 แบบคือ แบบแบ่งครึ่งช่วง แบบวางตำแหน่งผิดที่ แบบนิวตัน-ราฟสัน และแบบเซแคนต์ ทั้ง 4 แบบนี้มีขั้นตอนการแก้ปัญหาที่คล้ายกันคือ ต้องกำหนดฟังก์ชันที่จะใช้หารากสมการ ต้องกำหนดค่าเริ่มต้นของรากสมการ แล้วนำค่าเริ่มต้นนั้นไปวนรอบเพื่อหาค่ารากสมการที่ให้ค่าเท่ากับหรือใกล้ค่าแท้จริงมากที่สุด ถ้าค่าที่ได้ในการวนรอบครั้งแรกยังไม่อยู่ในขอบเขตของความคลาดเคลื่อนที่ยอมรับได้ ก็จะต้องกำหนดค่าประมาณของรากสมการครั้งถัดไป แล้ววนรอบเช่นนี้ซ้ำแล้วซ้ำเล่า จนได้ค่าที่ได้อยู่ในช่วงที่ยอมรับได้ ผู้วิจัยจึงกำหนดตัวแปรและเมธอดที่มีลักษณะเหมือนกันเหล่านี้ไว้ในคลาสแม่ที่ชื่อว่า RootUtils ในคลาสนี้จะมี instance variable เป็นฟังก์ชันที่จะใช้หารากสมการ ตัวแปรสำหรับเก็บจำนวนครั้งการวนรอบ ตัวแปรสำหรับเก็บคำตอบของรากสมการ จากนั้นจึงออกแบบเมธอดที่ใช้ได้กับทุก ๆ ระเบียบวิธี ได้แก่ เมธอด testInterval () (ตรวจสอบดูว่าภายในช่วงดังกล่าวมีรากสมการปรากฏอยู่หรือไม่) เมธอด testLoopCount() (ใช้หาค่าจำนวนรอบที่วนผ่านไปแล้ว เพื่อป้องกันการวนรอบแบบไม่รู้จบ) และมีเมธอด getRoot() (หาค่ารากสมการ) findNextPosition() (หาค่ารากสมการถัดไปที่ให้ค่าใกล้เคียงค่าแท้จริงกว่าค่าก่อนหน้า) และจะต้องมีเมธอด doIteration() (ทำการวนรอบจนกว่าจะได้ค่ารากสมการที่เท่ากับหรือใกล้ค่าแท้จริง และเมธอด IsSuccessed () ตรวจสอบว่าการหารากสมการสิ้นสุดได้หรือยัง) เมธอด 3 เมธอดหลังนี้จะต้องเขียนคำสั่งไว้ในแต่ละคลาสลูก เพราะแต่ละระเบียบวิธีจะมีลักษณะการทำงานแตกต่างกัน ในการออกแบบการหาค่ารากสมการแบบไม่เป็นเชิงเส้นนี้ ได้กำหนดให้คลาส BadIntervalException และ ExceedLoopException เป็นคลาสสำหรับดักจับความผิดพลาดที่เกิดขึ้นระหว่างที่โปรแกรมทำงาน

ขั้นตอนที่ 3 เขียนคำสั่งโปรแกรมเป็นภาษาจาวา ทำการคอมไพล์ให้เป็น byte code อยู่ในรูปของไฟล์ class คำสั่งต่าง ๆ ที่เขียนในโปรแกรมต้นฉบับงานวิจัยนี้จะยึดมาตรฐานของจาวารุ่นที่ 6.0 หรือทันสมัยกว่าเป็นหลัก นั่นหมายถึงการนำโปรแกรมต้นฉบับไปคอมไพล์โดยใช้จาวาที่มีรุ่นต่ำกว่า 6.0 อาจพบข้อความผิดพลาดขณะทำการคอมไพล์ อาจจะต้องตัดแปลงหรือแก้ไขคำสั่งบางคำสั่งให้สอดคล้องกับจาวารุ่นที่เลขเวอร์ชันต่ำกว่า 6.0

โปรแกรม text editor ที่ใช้เขียน source code สามารถใช้ editor ตัวก็ได้ เช่น Notepad , Editplus, WinEditor หรือจะพัฒนาใน IDE (Integrate Developing Environment) เช่น Eclipses หรือ Netbean หรือ Jcreator ก็ได้เช่นเดียวกัน

เพื่อความเป็นระเบียบและเป็นระบบในการเขียนโปรแกรม และเพื่อความสะดวกในการนำไปใช้งาน ผู้วิจัยจึงได้จัดสร้างโฟลเดอร์ มีลักษณะเป็นลำดับชั้นดังนี้



ในโฟลเดอร์ MathTools จะเก็บ class file ที่ผ่านการคอมไพล์ แบ่งเป็นโฟลเดอร์ย่อยตามหัวข้อเรื่อง โฟลเดอร์ src ใช้เก็บ source code ที่เขียนด้วยภาษาจาวา เก็บเป็นโฟลเดอร์ย่อยเรียงตามหัวข้อเรื่องเช่นเดียวกัน

โฟลเดอร์ Common ใช้เก็บค่าคงที่ทั่วไป ค่าคงที่ทางคณิตศาสตร์และฟิสิกส์ ฟังก์ชันที่จะต้องใช้ร่วมกันทั้งproject

โฟลเดอร์ Derivative เก็บคลาสและโปรแกรมที่ใช้แก้ปัญหาการหาอนุพันธ์

โฟลเดอร์ DifferentialEq เก็บคลาสและโปรแกรมการแก้สมการอนุพันธ์แบบสามัญ

โฟลเดอร์ Integration เก็บคลาสและโปรแกรมการหาปริพันธ์เบื้องต้น

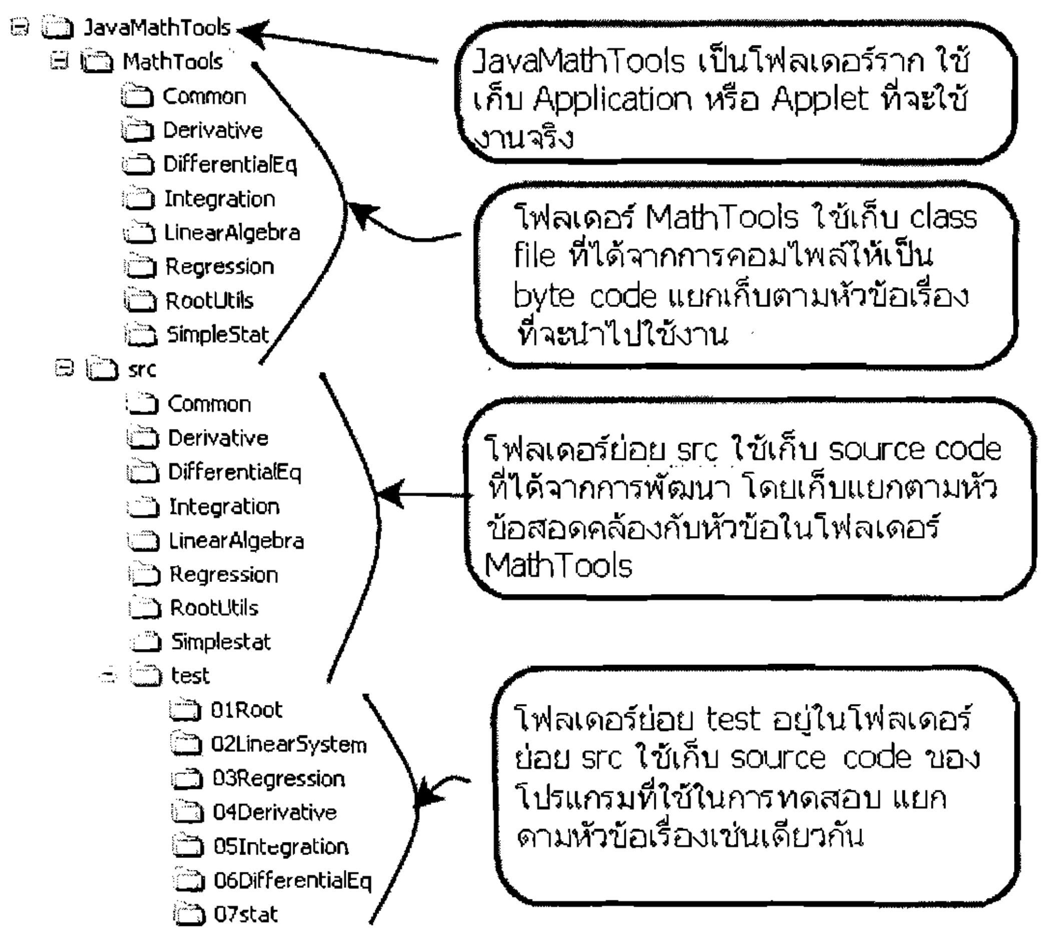
โฟลเดอร์ LinearAlgebra เก็บคลาสและโปรแกรมการคำนวณเมตริกซ์ และดีเทอร์มิแนนท์ การแก้สมการระบบสมการเชิงเส้น

โฟลเดอร์ Regression เก็บคลาสและโปรแกรมที่ใช้การประมาณค่าโดยวิธีกำลังสองน้อยที่สุด ได้แก่ Linear regression และ Polynomial regression.

โฟลเดอร์ SimpleStat เก็บคลาสและโปรแกรมที่ใช้ในการคำนวณทางสถิติเบื้องต้น

โฟลเดอร์ test มีเฉพาะในโฟลเดอร์ src ใช้เก็บโปรแกรมที่ใช้ในการทดสอบ class file ของแต่ละหัวข้อ

หรือดูจากโฟลเดอร์ที่เก็บในฮาร์ดดิสก์ จะเป็นดังนี้



ขั้นตอนที่ 4 ทดสอบการทำงานของโปรแกรม โดยนำปัญหาจากแบบฝึกหัดหรือจากตัวอย่างต่าง ๆ ในวิชาฟิสิกส์พื้นฐาน หรือจากผลการทดลองในห้องปฏิบัติการฟิสิกส์ ให้โปรแกรมคำนวณหาคำตอบแล้วนำไปเปรียบเทียบกับค่าที่แท้จริง พร้อมทั้งจับเวลาที่ใช้ในการคำนวณ

กรณีที่ปัญหาบางข้อไม่ทราบค่าที่แท้จริง สามารถหาคำตอบเพื่อเปรียบเทียบค่าได้จาก การใช้โปรแกรมสำเร็จรูปทางคณิตศาสตร์ ในที่นี้คณะผู้วิจัย ใช้ Mathematica หาผลลัพธ์เพื่อนำมาเปรียบเทียบ

โปรแกรมที่ใช้ทดสอบการทำงาน หรือ test drive จะถูกเก็บไว้ในโฟลเดอร์ย่อยชื่อ test ซึ่งอยู่ภายใต้โฟลเดอร์ src อีกทีหนึ่ง โฟลเดอร์ test นี้ยังเก็บโปรแกรมต้นฉบับแยกย่อยไปตามหัวข้อเรื่องที่ใช้ในการทดสอบ โปรแกรมที่ใช้ในการทดสอบเมื่อผ่านการคอมไพล์แล้ว จะอยู่ในรูป class file ภายใต้โฟลเดอร์ราก JavaMathTools สามารถใช้โปรแกรม Interpreter ประมวลผลคลาสไฟล์เหล่านี้เพื่อดูผลลัพธ์ได้เลย

เพื่อความสะดวกในการทดสอบ โปรแกรมที่ใช้ในการทดสอบจะเขียนอยู่ในรูป Application ที่ทำงานในเครื่องคอมพิวเตอร์ที่กำลังทดสอบ การทดสอบในรูป applet ซึ่งจะต้องนำไปเก็บไว้ที่เครื่องแม่ข่ายที่ให้บริการอินเทอร์เน็ต จะกระทำเมื่อโปรแกรมต่าง ๆ ในแต่ละหัวข้อไม่มีข้อผิดพลาดที่จะต้องแก้ไขแล้ว

ขั้นตอนที่ 5 ปรับปรุงหรือแก้ไขชุดคำสั่ง ในกรณีที่ประมวลผลแล้วได้ค่าคลาดเคลื่อนหรือมีการใช้เวลาในการประมวลผลนานผิดปกติ

ถ้าพบว่ามีการประมวลผลแล้วได้คำตอบคลาดเคลื่อนไปจากค่าแท้จริงมาก ผู้วิจัยต้องเริ่มตรวจสอบตั้งแต่ชุดคำสั่งที่ถอดมาจากระเบียบวิธีการคำนวณที่ออกแบบไว้ ถ้าชุดคำสั่งไม่มีข้อผิดพลาด ต้องย้อนกลับไปดูระเบียบวิธีการคำนวณแต่ละขั้นตอนว่าขั้นตอนใดทำให้เกิดข้อผิดพลาดบ้าง อาจต้องออกแบบหรือแก้ไขปรับปรุงระเบียบวิธีการคำนวณใหม่ เพื่อให้ได้คำตอบที่ใกล้ค่าแท้จริง เกิดความคลาดเคลื่อนน้อยที่สุด

การทดสอบดูการทำงานของโปรแกรมนอกจากจะใช้วิธีให้ตัวโปรแกรมพิมพ์ค่าที่ได้ในการคำนวณแสดงผลทางจอภาพในแต่ละขั้นตอนแล้ว สามารถใช้ Eclipses เป็น Debugger เพื่อหาตำแหน่งหรือชุดคำสั่ง ที่ทำให้ผลการคำนวณคลาดเคลื่อนได้อีกด้วย

ขั้นตอนที่ 6 นำไปทดลองใช้ร่วมกับบทเรียนทางฟิสิกส์สำหรับการเรียนรู้ด้วยตนเองผ่านทางอินเทอร์เน็ต ในการทดลองนี้นำบทเรียนทางฟิสิกส์ที่ต้องใช้การคำนวณเป็นบทนำร่อง นำ class file ที่จัดสร้างไว้เรียบร้อยแล้ว มาใช้ร่วมกับเอกสาร HTML โดยจัดทำในรูป applet

ขั้นตอนที่ 7 เผยแพร่ class file และ source code ที่พัฒนาได้ ผู้สนใจที่จะนำไปใช้ในการเรียนการสอน หรือนำไปใช้เป็นส่วนประกอบของเว็บเพ็จสามารถนำไปใช้ได้ทันที โดยไม่ต้องเสียค่าลิขสิทธิ์หรือค่าใช้จ่ายใด ๆ โดยจะติดตั้งและเผยแพร่ผ่านเว็บไซต์ของภาควิชาฟิสิกส์ www.rmutphics.com หรือเข้าถึงโดยตรงที่ http://203.158.100.140/jmt

บทที่ 4
ผลการวิจัย

ในการสร้าง class library ผู้วิจัยได้พัฒนาและจัดเก็บไว้ในโฟลเดอร์หลักชื่อว่า Math Tools และแยกเก็บเป็นโฟลเดอร์ย่อย ๆ ภายใต้โฟลเดอร์หลัก ตามหัวข้อดังนี้

- Common เป็นโฟลเดอร์ใช้เก็บค่าคงที่ทางคณิตศาสตร์และฟิสิกส์ รวมทั้งค่าคงที่ที่เกี่ยวข้องกับโปรแกรมที่ทุก ๆ คลาสจำเป็นที่จะต้องใช้
- RootUtils เป็นโฟลเดอร์เก็บคลาสที่ทำหน้าที่หารากสมการแบบไม่เป็นเชิงเส้น ประกอบด้วยคลาส Bisection, คลาส False Position คลาส Newton Raphson และคลาส Secant
- LinearAlgebra เป็นโฟลเดอร์ที่เก็บคลาสที่ใช้หาคำตอบของระบบสมการเชิงเส้น คลาสที่เกี่ยวข้องกับปฏิบัติการเมตริกซ์ เช่น การบวก ลบ และคูณเมตริกซ์ การหา determinant การหาเมตริกซ์ผกผัน คลาสที่ใช้หาคำตอบของระบบสมการเชิงเส้น ได้แก่ คลาส Cramer คลาส Gauss Elimination คลาส LU Decompositon
- Regression เป็นโฟลเดอร์เก็บคลาสที่ใช้คำนวณสมการการถดถอยเชิงเส้นตรงและแบบพหุนาม คลาสที่เกี่ยวข้องได้แก่ คลาส Linear Regression คลาส Polynomial Regression
- Derivative เป็นโฟลเดอร์ที่ใช้เก็บคลาสที่ใช้หาอนุพันธ์อันดับหนึ่ง และอันดับสอง ประกอบด้วยคลาส First Derivative และ Second Derivative
- Integration เป็นโฟลเดอร์ที่ใช้กับคลาสที่ใช้คำนวณค่าปริพันธ์ คลาสที่เกี่ยวข้องได้แก่ คลาส trapezoidal คลาส Simpson1-3, คลาส Simpson 3-8 และคลาส Gauss Quadrature
- DifferentialEq เป็นโฟลเดอร์ที่ใช้เก็บคลาสที่ใช้แก้สมการอนุพันธ์อันดับหนึ่ง ประกอบด้วยคลาส Modified Euler คลาส RungKutta และคลาส AdamsMulton
- SimpleStat เป็นโฟลเดอร์ที่ใช้เก็บคลาสที่ใช้คำนวณค่าทางสถิติเบื้องต้นทั้งหมด

ในการนำ class library ไปใช้งานสามารถอ้างถึงโดยใช้คำสั่ง import ตามด้วยโฟลเดอร์หลักและโฟลเดอร์ย่อยที่คลาสนั้นอยู่ เช่น ต้องการใช้คลาส Newton Raphson ไปหารากสมการแบบไม่เป็นชิงเส้นทำได้ดังนี้

```
Import MathTool.RootUtils.NewtonRaphson;
```

ต้องการคำนวณเพื่อหา slope และจุดตัดแกน y ของสมการเชิงเส้น ต้องเรียกใช้คลาส Linear Regression ดังนี้

Import MathTools.Regression LinearRegression;

หรือใช้สัญลักษณ์ * แทนชื่อคลาสทุกคลาสที่อยู่ภายใต้โฟลเดอร์นั้นก็ได้

Import MathTools.Regression .* ;

คำสั่งนี้สามารถเรียกใช้ class ทุกคลาสที่อยู่ภายใต้โฟลเดอร์ MathTools/Regression

4.1 การหารากสมการแบบไม่เป็นเชิงเส้น

สมการแบบไม่เป็นเชิงเส้นที่จะหารากสมากรนี้จะต้องเป็นสมการที่มีตัวแปรเพียงตัวเดียว เขียนในรูปทั่วไปได้เป็น

$$f(x) = 0$$

รากสมการที่หาได้จะเป็นจำนวนจริง

ตัวอย่างสมการแบบไม่เป็นเชิงเส้น

$$x^3+4x^2-x+3 = 0$$

$$\sin x + 2 - xe^{-3x} = 0$$

$$\sqrt{x^2+1}-5 = 0$$

ผู้วิจัยได้เลือกวิธีการหารากสมการไว้ 4 วิธี เพื่อพัฒนาเป็น class library คือ

1. วิธีแบ่งครึ่งช่วง (Bisection)
2. วิธีวางตำแหน่งผิดที่ (Method of False Position)
3. วิธีนิวตัน-ราฟสัน (Newton-Raphson Method)
4. วิธีเซแคนต์ (Secant Method)

ทั้ง 4 วิธีนี้เป็นการหาค่ารากสมการโดยประมาณ ทุกวิธีจะต้องกำหนดค่าเริ่มต้นที่คิดว่ามีค่าใกล้เคียงกับรากสมการเพื่อใช้ประมาณค่ารากสมการเสียก่อน จากนั้นจะนำค่านี้ไปวนรอบเพื่อหาค่าที่ทำให้ f(x) = 0 หรือมีค่าอยู่ในช่วงที่ความคลาดเคลื่อนเป็นที่ยอมรับได้

ในการออกแบบโปรแกรมจึงกำหนดให้คลาส RootUtisls เป็นคลาสหลักและเป็นคลาสนามธรรมของทุก ๆ คลาส เมธอด (method) ต่าง ๆ ที่ต้องใช้ร่วมกันจะถูกสร้างเก็บไว้ในคลาสนี้

คลาสแม่หรือคลาสหลัก RootUtils.java มีรายละเอียด

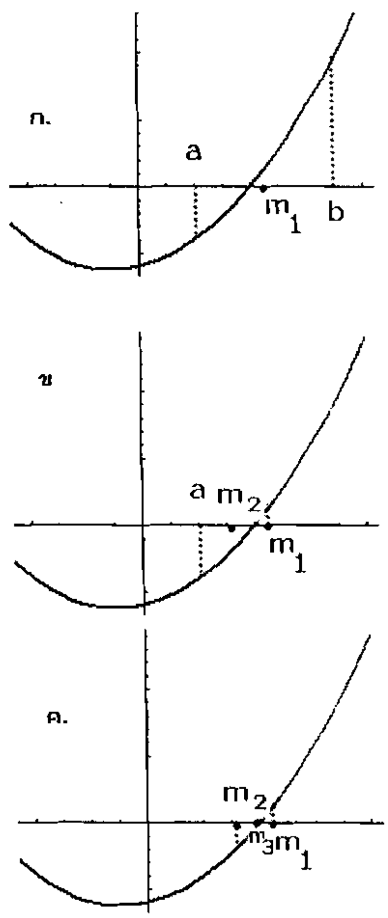
```java
/* File : RootUtils.java */

package MathTools.RootUtils;

import static MathTools.Common.CommonConstants.*;
import MathTools.Common.Function;

public abstract class RootUtils {
      Function function;
      private int counter;
      private int max_loop;

RootUtils ( Function f, int max_loop){
      this.function = f;
      this.max_loop = max_loop;
}
public int getLoopCount() { return counter;}
abstract double getRoot();

void resetCounter() {  counter = 0; }

public void testInterval (double x1, double x2) throws
BadIntervalException {
      double y1 = function.Of(x1);
      double y2 = function.Of(x2);
      if (y1*y2 > 0 || (x1 > x2)){
      throw new BadIntervalException();
      }
}
public void testLoopCount() throws ExceedLoopException {
      counter = counter+1;
      if (counter > max_loop){
            throw new ExceedLoopException ();
      }
}
abstract void findNextPosition();
abstract void doIteration(int count);
abstract boolean IsSuccessed ();
public boolean DoProcess() throws ExceedLoopException {
      testLoopCount();
      doIteration(counter);

      findNextPosition();
      return IsSuccessed();
}
}
```

ในการออกแบบนี้ได้สร้าง class สำหรับดักจับความผิดพลาดระหว่างที่โปรแกรมทำงาน 2 คลาสคือ BadInteval Exception ความผิดพลาดนี้เกิดขึ้นจากค่าเริ่มต้นที่กำหนดให้นั้นไม่สามารถทำให้โปรแกรมหาค่ารากสมการได้ ผู้ใช้ต้องกำหนดการประมาณค่าเริ่มต้นใหม่หรือเกิดจากสมการนั้นมีรากเป็นจำนวนจินตภาพ และคลาส ExceedLoopException เป็นคลาสดัก

การวนรอบ ป้องกันมิให้เกิดการวนรอบแบบไม่รู้จบ ทั้งสี่วิธีได้กำหนดเป็นค่าปริยาย (default) ไม่เกิน 50 รอบ จำนวนครั้งการวนรอบผู้ใช้สามารถกำหนดเองได้ตามใจชอบ

4.1.1 **วิธีแบ่งครึ่งช่วง** (Bisection Method)

เป็นวิธีที่ง่ายที่สุด เกิดความผิดพลาดที่คาดไม่ถึงน้อยที่สุดเมื่อทำเป็นโปรแกรมคอมพิวเตอร์ ใช้หารากสมการได้ทั้งชนิดที่เป็นพหุนามและเชิงอดิศัย เริ่มต้น ด้วยการระบุช่วงตัวเลขที่ต้องการหารากสมการขึ้นมาก่อน สมมติว่าอยู่ในช่วง a ถึง b โดยที่ค่า b มีค่ามากกว่า a เขียนเป็นสัญลักษณ์ได้เป็น [a,b] a คือ ขีดจำกัดล่าง b คือ ขีดจำกัดบน ค่าฟังก์ชัน f(x) ในช่วง [a,b] นี้ต้องมีค่าต่อเนื่อง เมื่อนำ a และ b แทนค่าลงในฟังก์ชันแล้ว ผลคูณของค่าฟังก์ชันจะต้องมีค่าเป็นลบ (f(a).f(b) < 0) ขั้นตอนการหารากสมการมีดังนี้



รูป 4.1 แสดงวิธีการหารากสมการโดยวิธีแบ่งครึ่งช่วง

ขั้นที่ 1 เลือกค่า a และ b โดยการสุ่มแต่ต้องทำให้ f(a).f(b) น้อยกว่า ศูนย์

ขั้นที่ 2 หาค่ารากสมการโดยประมาณโดยแบ่งครึ่งช่วงระหว่าง a กับ b ให้จุดที่แบ่งครึ่ง (ครั้งที่ 1) นี้คือ m_1 (m คือ midpoint)

m_1 = (a + b)/2 ดังรูป ก.

ขั้นที่ 3 ขณะนี้ช่วง a และ b จะถูกแบ่งเป็น 2 ช่วง โดยมี m_1 เป็นจุดแบ่งครึ่ง ตรวจสอบดูว่ารากสมการที่แท้จริงอยู่ช่วงใด

ก. ถ้า f(a).f(m_1) < 0 แสดงว่ารากสมการอยู่ในช่วง [a, m_1]

ข. ถ้า f(m_1).f(b) < 0 แสดงว่ารากสมการอยู่ในช่วง [$_1$, b]

ค. ถ้า f(a).f(m_1)= 0 หรือน้อยกว่าความคลาดเคลื่อนที่ยอมรับให้มีได้แสดงว่า m_1 คือรากสมการและสิ้นสุดการคำนวณ

ขั้นที่ 4 ถ้า m_1 ยังไม่ใช่รากสมการที่ต้องการ ให้แบ่งครึ่งช่วงอีก (ย้อนกลับไปขั้นที่ 2) จากรูป (ข)

m_2 = (a + m_1)/2

กระทำซ้ำเช่นนี้เรื่อย ๆ จนถึง k ครั้ง เมื่อได้ f(m_{k-1}).f(m_k) เท่ากับศูนย์หรือน้อยกว่าความคลาดเคลื่อนที่กำหนดแล้ว ค่า m_k นี้คือรากสมการที่ต้องการ

การแบ่งครึ่งช่วงแต่ละครั้ง จะทำให้ความกว้างของช่วงตัวเลขแคบลงเรื่อย ๆ แต่ละครั้งจุด m_k จะเป็นค่าประมาณของรากสมการ การกระทำซ้ำจะสิ้นสุดลงเมื่อ m_k อยู่ในช่วงความคลาดเคลื่อนที่ยอมรับได้ (given tolerance)

เพื่อป้องกันความผิดพลาดที่ไม่คาดคิดเกิดขึ้น ทำให้โปรแกรมทำงานวนรอบไม่สิ้นสุด จึงกำหนดจำนวนรอบสูงสุดของการกระทำซ้ำไว้ด้วย

ในการพัฒนา ผู้วิจัยได้สร้างคลาส Bisection ซึ่งสืบทอดคุณสมบัติมาจากคลาสแม่ คือ RootUtils ได้สร้างเมธอด doIteration(int) บรรทัดที่ 56-65, findNextPosition() บรรทัดที่ 66-70, IsSuccessed() บรรทัดที่ 71-73 การหารากสมการจะสำเร็จเมื่อ นำค่าประมาณที่ได้ในแต่ละการวนรอบไปแทนลงในฟังก์ชันแล้วฟังก์ชันนั้นมีค่าเป็นศูนย์หรือมีค่าน้อยกว่า ZERO_APPROACH (กำหนดค่าไว้ที่ 10^{-16})

```
 1: /* File : Bisection.java*/
 2: package MathTools.RootUtils;
 3:
 4: import static java.lang.Math.*;
 5:
 6: import MathTools.RootUtils.*;
 7: import MathTools.Common.Function;
 8: import static MathTools.Common.CommonConstants.*;
 9:
10: public class Bisection extends RootUtils{
11:       private double xLeft;
12:       private double xRight;
13:       private double xMid;
14:       private double f_xLeft;
15:       private double f_xRight;
16:       private double f_xMid;
17:
18:       /** constructor    */
19: public Bisection(Function function, double xMin, double xMax)
20:             throws BadIntervalException {
21:             super(function, MAX_LOOP);
22:             testInterval(xMin, xMax);
23:
24:             double yMin = function.Of(xMin);
25:             double yMax = function.Of(xMax);
26:
27:             if (yMin < 0){
28:                   xLeft = xMin;
29:                   xRight = xMax;
30:                   f_xLeft = yMin;
31:                   f_xRight = yMax;
32:             } else {
33:                   xLeft = xMax;
34:                   xRight = xMin;
35:                   f_xLeft = yMax;
36:                   f_xRight = yMin;
37:             }
38:             findNextPosition();
39:                   try{
40:                   boolean IsOK;
41:                   do{
```

```
                        IsOK = DoProcess();
                    }while (!IsOK);
                }// try
                catch (Exception e)      { System.out.println( e);}
    }

    public double get_xLeft () { return xLeft;}
    public double get_xMid () { return xMid;}
    public double get_xRight () { return xRight;}
    public double get_f_xLeft () { return f_xLeft;}
    public double get_f_xRight () { return f_xRight;}
    public double get_f_xMid () { return f_xMid;}
    public double getRoot() { return get_xMid();}

    void doIteration( int count) {
          if ((f_xLeft*f_xMid) < 0 ){
                xRight = xMid;
                f_xRight = f_xMid;
          } else {
                xLeft = xMid;
                f_xLeft = f_xMid;
          }

    }
    void findNextPosition() {
          xMid = (xLeft + xRight)/2.0;
          f_xMid = function.Of(xMid);
    }

    boolean IsSuccessed() {
          return abs(f_xMid) <= ZERO_APPROACH;
    }

    }
```

4.1.2 **วิธีวางตำแหน่งคำตอบผิดที่** (Method of False Position)

วิธีนี้คล้ายกับวิธีแบ่งครึ่งช่วง สิ่งที่ต่างกันคือแทนที่จะประมาณค่ารากสมการเริ่มต้นด้วยจุดกึ่งกลาง กลับใช้วิธีลากเส้นตรงผ่านจุด f(a) และ f(b) เส้นตรงนี้ตัดผ่านแกน x ที่จุดใดจะใช้ค่านี้เป็นค่าประมาณเริ่มต้น

เงื่อนไขที่จะใช้วิธีนี้ได้ คือ f(x) จะต้องเป็นฟังก์ชันที่มีค่าต่อเนื่องในช่วง [a,b] ผลคูณของ f(a) และ f(b) มีค่าเป็นลบ เมื่อลากเส้นตรงจากจุด (a,f(a)) ไปยังจุด (b,f(b)) ผ่านแกน x ที่จุด (x_1, 0) x_1 จะเป็นค่าประมาณค่าแรก (ดูรูป 4.2)

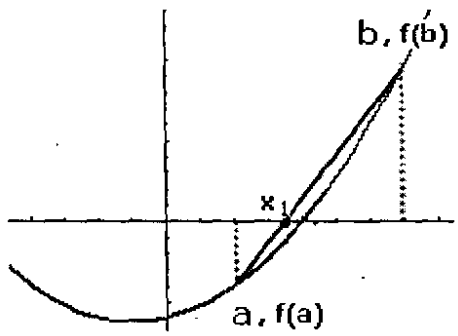
สมการของเส้นตรงนี้คือ

$$y - f(a) = \frac{f(b) - f(a)}{(b - a)}(x - a)$$

ที่ $x = x_1$ $y = 0$

$$-f(a) = \frac{f(b) - f(a)}{(b - a)}(x_1 - a)$$

$$x_1 = \frac{a\,f(b) - b\,f(a)}{f(b) - f(a)}$$



รูป 4.2 แสดงจุดที่วางตำแหน่งผิดที่

จัดรูปสมการใหม่ โดยแยกออกเป็นเศษส่วนย่อย

$$x_1 = \frac{af(b)}{f(b) - f(a)} - \frac{bf(a)}{f(b) - f(a)}$$

นำ a และ –a ใส่เพิ่มเข้าไปทางซ้ายมือ

$$x_1 = a + [\frac{af(b)}{f(b) - f(a)} - a] - \frac{bf(a)}{f(b) - f(a)}$$

$$x_1 = a + \frac{f(a)(a - b)}{f(b) - f(a)}$$

$$x_1 = a - \frac{f(a)(b - a)}{f(b) - f(a)} \qquad \ldots\ldots(4.1)$$

จากนั้นตรวจสอบต่อไปว่าระหว่าง $[a,x_1]$ และ $[x_1,b]$ ค่ารากสมการที่แท้จริงจะตกอยู่ช่วงใด จะเห็นได้ชัดว่า ถ้า f(a) มีค่าน้อยกว่า f(b) ค่า x_1 จะเยื้องมาทางปลายจุด a มากกว่าจุด b การตรวจสอบทำได้โดยดูเครื่องหมายของผลคูณระหว่าง $f(a)f(x_1)$ และ $f(x_1)f(b)$ เปลี่ยนค่าช่วง a และ b แล้วหาค่า x_2 โดยวิธีเดียวกัน กระทำซ้ำเช่นนี้เพื่อหาค่า x_3, x_4, ... จนกระทั่งได้ค่า x_k ซึ่งมีค่าน้อยกว่าค่าความคลาดเคลื่อนที่ยอมรับได้ จึงหยุดการทำงาน

ผู้วิจัยได้สร้างคลาส FalsePosition ซึ่งสืบทอดคุณสมบัติมาจากคลาสแม่คือ RootUtils ได้สร้างเมธอด doIteration(int) บรรทัดที่ 57-66, findNextPosition()บรรทัดที่ 67-70, IsSuccessed() บรรทัดที่ 71-73 การหารากสมการจะสำเร็จเมื่อ นำค่าประมาณที่ได้ในแต่ละการวนรอบไปแทนลงในฟังก์ชันแล้วฟังก์ชันนั้นมีค่าเป็นศูนย์ หรือมีค่าน้อยกว่า ZERO_APPROACH (กำหนดค่าไว้ที่ 10^{-16}) จะเห็นได้เมธอดทั้งสามเมธอดนั้นมีชุดคำสั่งไม่เหมือนกับของคลาส Bisection เรียกวิธีการเขียนเมธอดเช่นนี้ว่าเป็นการ override เมธอดต่าง ๆ จากคลาสแม่ของมัน

```
/* file : FalsePosition.java */
package MathTools.RootUtils;

import static java.lang.Math.*;

import MathTools.RootUtils.*;
import MathTools.Common.Function;
import static MathTools.Common.CommonConstants.*;

public class FalsePosition extends RootUtils{
      private double xLeft;
      private double xRight;
      private double xFalse ;
      private double f_xLeft;
      private double f_xRight;
      private double f_xFalse;

      /** constructor   */
 public FalsePosition(Function function,double xMin,double xMax)
            throws BadIntervalException {
            super(function, MAX_LOOP);
            testInterval(xMin, xMax);

            double yMin = function.Of(xMin);
            double yMax = function.Of(xMax);

            if (yMin < 0){
                  xLeft = xMin;
                  xRight = xMax;
                  f_xLeft = yMin;
                  f_xRight = yMax;
            } else {
                  xLeft = xMax;
                  xRight = xMin;
                  f_xLeft = yMax;
                  f_xRight = yMin;
            }
            findNextPosition();
            try{
                  boolean IsOK;
                  do{
                        IsOK = DoProcess();
                  }while (!IsOK);
            }// try
            catch (Exception e)     { System.out.println( e);}

      }

  public double get_xLeft () { return xLeft;}
  public double get_xFalse () { return xFalse;}
  public double get_xRight () { return xRight;}
  public double get_f_xLeft () { return f_xLeft;}
  public double get_f_xRight () { return f_xRight;}
  public double get_f_xFalse () { return f_xFalse;}
  public double getRoot() { return xFalse;}


  void doIteration(int count ) {
        if (count==1) return;
        if (f_xFalse < 0 ){
              xLeft = xFalse;
```

```
                f_xLeft = f_xFalse;
        } else {
                xRight = xFalse;
                f_xRight = f_xFalse;
        }
}
void findNextPosition() {
xFalse = xLeft - f_xLeft*(xRight - xLeft)/(f_xRight - f_xLeft);
        f_xFalse = function.Of(xFalse);
}

boolean IsSuccessed() {
        return abs(f_xFalse) < ZERO_APPROACH;
}
}
```

4.1.3 **วิธีนิวตัน-ราฟสัน** (Newton - Raphson Method)

ถ้ากำหนด x_0 เป็นค่าแรกของการประมาณค่ารากสมการ ค่ารากสมการที่แท้จริงคือ α h เป็นค่าความคลาดเคลื่อน รากสมการของฟังก์ชัน f(x) = 0 คือ $\alpha = x_0 + h$

กระจายฟังก์ชัน f(x) เป็นอนุกรมเทเลอร์ รอบจุด x_0 จะได้

$$f(x) = f(x_0 + h) = f(x_0) + hf'(x_0) + h^2 \frac{f''(x_0)}{2!} + \ldots$$

ความคลาดเคลื่อน h มีค่าน้อย ๆ จนสามารถตัด $h^2, h^3 \ldots$ ทิ้งได้ จะเหลือ

$$f(x_0) + h f'(x_0) = 0$$

$$h \approx -\frac{f(x_0)}{f'(x_0)}$$

แทนค่า h จะได้ค่ารากสมการที่ใกล้เคียงค่าจริงมากกว่า x_0 คือ

$$x_1 = x_0 + h = x_0 - \frac{f(x_0)}{f'(x_0)}$$

กระทำซ้ำ $x_2, x_3 \ldots$ จนได้ค่าที่มีความคลาดเคลื่อนน้อยลงจนถึงครั้งที่ x_{i+1} สามารถเขียนเป็นสูตรทั่วไปได้ดังนี้

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad \ldots\ldots(4.2)$$

เมื่อ i = 0,1,2,3......

สมการนี้ เป็นสมการของนิวตัน-ราฟสัน ใช้สำหรับหารากสมการของฟังก์ชัน f(x)

การใช้งานคลาส NewtonRaphson จะต้องกำหนดค่าเริ่มต้นซึ่งเป็นค่าประมาณของรากสมการ และอนุพันธ์อันดับหนึ่งของฟังก์ชันที่ต้องการหารากสมการ ในการวนรอบเพื่อหาค่ารากสมการโดยประมาณแต่ละครั้ง(เมธอด findNextPostion() บรรทัดที่ 48 ถึง 51) โปรแกรมจะสร้างเส้นสัมผัสขึ้นมา ตำแหน่งที่เส้นสัมผัสไปตัดกับแกน x คือค่ารากสมการโดยประมาณในครั้งถัดไป

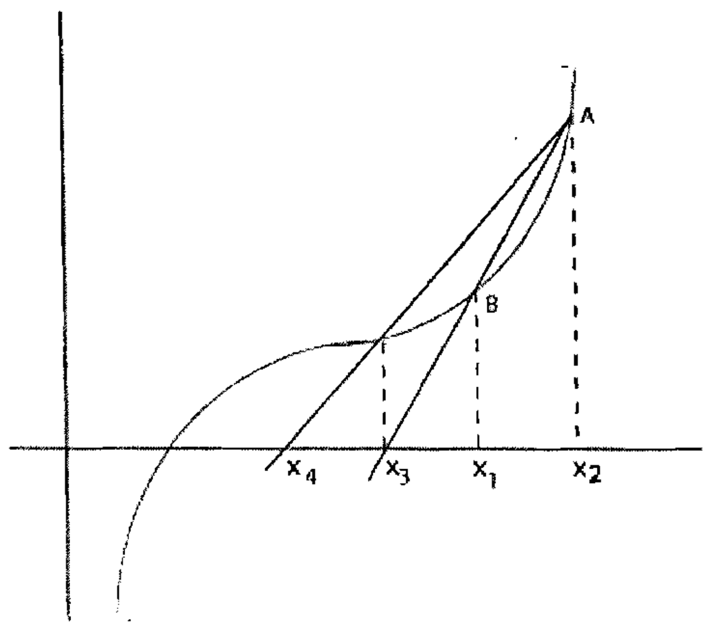
โปรแกรมจะทำงานสิ้นสุดเมื่อค่ารากสมการที่ได้ในการวนรอบปัจจุบันต่างกับค่ารากสมการที่ได้จากการวนรอบที่แล้วน้อยกว่าค่าความคลาดเคลื่อนที่ยอมรับได้ (โปรแกรมได้ตั้งค่า DEFAULT_TOLERANCE ไว้ที่ 10^{-7})

```
  /* file : NewtonRaphson.java */
package MathTools.RootUtils;
import static java.lang.Math.*;

import MathTools.RootUtils.*;
import MathTools.Common.Function;
import static MathTools.Common.CommonConstants.*;

public class NewtonRaphson extends RootUtils{
        private double xn;
        private double xn_next;
        private double f_xn;
        private double f_xn_next;
        private double diff_f_xn;

        /** constructor   */
        public NewtonRaphson(Function function, double guess)
                throws BadIntervalException {
                super(function, MAX_LOOP);

                xn = guess;
                f_xn = function.Of(xn);
                diff_f_xn = function.DerivativeOf(xn);
                xn_next = xn - f_xn/diff_f_xn;
                        boolean IsOK;
                        do{
                                IsOK = DoProcess();
                        }while (!IsOK);
                }
        catch(Exception e){ System.out.println(" Error(s):"+ e);}

        }

 public double get_xn () { return xn;}
 public double get_xn_next () { return xn_next;}
 public double get_f_xn () { return f_xn;}
 public double get_f_xn_next () { return f_xn_next;}
 public double get_diff_f_xn() { return diff_f_xn; }
 public double getRoot() { return get_xn_next();}


 void doIteration(int count ) {
        xn = xn_next;
        f_xn = function.Of(xn);
        diff_f_xn =function.DerivativeOf(xn);

 }
 void findNextPosition() {
                xn_next = xn - f_xn/diff_f_xn;

 }

 boolean IsSuccessed() {
        return abs(xn-xn_next) < DEFAULT_TOLERANCE;
```

```
    }

}
```

4.1.4 วิธีเซแคนต์ (Secant Method)

กรณีที่ฟังก์ชันมีค่าซับซ้อน การอนุพันธ์ของฟังก์ชันทำได้ไม่สะดวก ใช้เวลาในการคำนวณมาก อาจเลี่ยงไปใช้วิธีเซแคนต์ วิธีนี้เริ่มต้นด้วยการประมาณค่ารากสมการเริ่มต้น 2 ค่า และจะใช้ค่ารากทั้งสองนี้คำนวณหาค่าฟังก์ชันและค่ารากครั้งต่อ ๆ ไป โดยไม่จำเป็นต้องหาค่าอนุพันธ์ของฟังก์ชัน ไม่ต้องทดสอบว่าค่าฟังก์ชันของค่าประมาณทั้งสองมีค่าติดลบหรือไม่



เริ่มต้นด้วยการประมาณค่ารากสมการให้มีค่าเป็น x_1 และ x_2 ลากเส้นตรงเชื่อมระหว่างจุด $(x_1, f(x_1))$ และ $(x_2, f(x_2))$ เส้นตรงนี้ตัดแกน x ที่จุด x_3 ความชันของเส้นตรง AB หาได้จาก

$$\frac{f(x_2)-f(x_1)}{x_2-x_1} = \frac{f(x_1)-0}{x_1-x_3}$$

รูป 4.3 แสดงการหารากสมการโดยวิธีเซแคนต์

$$\text{จะได้} \quad x_3 = \frac{x_1 f(x_2) - x_2 f(x_1)}{f(x_2) - f(x_1)}$$

จากนั้นลากเส้นตรงเชื่อมระหว่างจุด $(x_3, f(x_3))$ และ $(x_2, f(x_2))$ เส้นตรงนี้ไปตัดแกน x ที่จุด x_4 โดยที่

$$x_4 = \frac{x_2 f(x_3) - x_3 f(x_2)}{f(x_3) - f(x_2)}$$

ในทำนองเดียวกันกับ $x_5, x_6, \ldots, x_n$ สามารถเขียนเป็นสูตรทั่วไปได้ดังนี้

$$x_n = \frac{x_{n-2} f(x_{n-1}) - x_{n-1} f(x_{n-2})}{f(x_{n-1}) - f(x_{n-2})}$$

เราสามารถหาสูตรการประมาณค่า x_n ใด ๆ ได้อีกวิธีหนึ่งดังนี้

สมการที่ใช้แทนเส้นตรง AB คือ

$$y - f(x_1) = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

ที่จุด x_3 จะได้ค่า $y = 0$

$$x_3 = x_2 - \frac{(x_2 - x_1)f(x_2)}{f(x_2) - f(x_1)}$$

เขียนให้อยู่ในรูปทั่วไปได้ดังนี้

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{\frac{f(x_{n-1}) - f(x_{n-2})}{(x_{n-1} - x_{n-2})}} \qquad \text{...............(4.3)}$$

สมการ (4.3) มีส่วนคล้ายกับสูตรของนิวตัน-ราฟสันในสมการ (4.2) แตกต่างกันตรงส่วน แทนที่จะเป็นค่าอนุพันธ์ สูตรของวิธีเซแคนต์จะเป็นการหาความชันของเส้นตรงที่ลากผ่านจุด x_n ในเมธอด findNextPosition() จะใช้สมการ (4.2) ประมาณค่ารากสมการในการวนรอบครั้งต่อไป

ข้อควรระวังของการใช้วิธีนี้คือ ถ้าค่า x_{n-1} และ x_n มีค่าใกล้กันมาก นั่นคือ y_{n-1} และ y_n จะมีค่าเข้าใกล้กันมากตามไปด้วย ผลต่างของ y_{n-1} และ y_n จะมีค่าน้อย จะเกิดความคลาดเคลื่อนเนื่องจากการปัดเศษ ยิ่งกระทำซ้ำหลายครั้ง ความคลาดเคลื่อนนี้จะถูกสะสมเพิ่มขึ้นเรื่อย ๆ จนได้ค่ารากสมการที่ผิดไปจากค่าที่แท้จริงมาก การกำหนดค่าเริ่มต้นที่ไม่เหมาะสม อาจทำให้เกิดการลู่เข้าสู่ค่ารากที่ไม่ต้องการ หรือไม่ลู่เข้าเลยเช่นเดียวกันกับวิธีของนิวตัน-ราฟสัน

```
/*file: Secant.java */

package MathTools.RootUtils;
import static java.lang.Math.*;
import MathTools.RootUtils.*;
import MathTools.Common.Function;
import static MathTools.Common.CommonConstants.*;

public class Secant extends RootUtils{
        private double xn_minus1;
        private double xn;
        private double xn_plus1;
        private double f_xn_minus1;
        private double f_xn;
        private double f_xn_plus1;

        /** constructor   */
 public Secant(Function function, double guess1, double guess2)
                throws BadIntervalException {
        super(function, MAX_LOOP);

        xn_minus1 = guess1;
```

```
            f_xn_minus1 = function.Of(guess1);
            xn          = guess2;
            f_xn        = function.Of(guess2);
            findNextPosition();
            try{
                  boolean IsOK;
                  do{
                        IsOK = DoProcess();
                  }while (!IsOK);

            }
            catch (Exception e)
            { System.out.println(" Error(s) : "+ e);
            }
            }
      public double get_xn_minus1 () { return xn_minus1;}
      public double get_xn () { return xn;}
      public double get_xn_plus1 () { return xn_plus1;}
      public double get_f_xn_minus1 () { return f_xn_minus1;}
      public double get_f_xn () { return f_xn;}
      public double get_f_xn_plus1 () { return f_xn_plus1;}
      public double getRoot() { return xn_plus1;}

      void doIteration(int count ) {
            xn_minus1 = xn;
            xn = xn_plus1;
            f_xn_minus1 = f_xn;

            f_xn = f_xn_plus1;
      }
      void findNextPosition() {
            xn_plus1 = xn -f_xn*(xn - xn_minus1)/(f_xn- f_xn_minus1);
            f_xn_plus1 = function.Of(xn_plus1);
      }
      boolean IsSuccessed() {
            return abs(f_xn_plus1) < ZERO_APPROACH;
      }
      }
```

4.1.5 การทดสอบการหารากสมการแบบไม่เป็นเชิงเส้น

การหารากสมการแบบไม่เป็นเชิงเส้นทั้ง 4 วิธี ได้นำมาทดสอบกับโจทย์คณิตศาสตร์และปัญหาในวิชาฟิสิกส์ดังตัวอย่างต่อไปนี้

<u>**ตัวอย่าง 4.1**</u> จงแก้สมการหาค่า x ต่อไปนี้

$$\cos x - x = 0 \quad \text{(ค่าแท้จริงของคำตอบนี้คือ } x = 0.739085\text{)}$$

<u>**วิธีทำ**</u> listing ต่อไปนี้เป็นโปรแกรมที่ใช้หาค่ารากสมการแบบไม่เป็นเชิงเส้น ในการทดสอบโปรแกรม ให้นำฟังก์ชันที่ต้องการหารากสมการ ในที่นี้คือ $\cos x - x$ พิมพ์ลงไปตรงส่วนที่ใช้แสดงค่าฟังก์ชันของโปรแกรม (บรรทัดที่ 12) และพิมพ์ฟังก์ชันอนุพันธ์อันดับหนึ่ง $-\sin x - 1$ ลงไปตรงบรรทัดที่ 13

กำหนดค่าเริ่มต้นในการหารากสมการคือ 0 และค่าสุดท้ายที่ใช้ในการประมาณค่าคือ 100 (ยกเว้นวิธีนิวตัน-ราฟสัน ประมาณค่าเริ่มต้นไว้เท่ากับ 0 เพียงค่าเดียว)

```
/* File : AllRootTest.java  */

import static java.lang.Math.*;
import MathTools.RootUtils.*;
import MathTools.Common.Function;

class  AllRootTest {

public static void main(String[] args) {
       long usedTime,start;
       Function func = new Function() {
       public double Of(double x) {return(cos(x)-x );  }
       public double DerivativeOf(double x){ return -sin(x)-1;}
 };
       try{
              start = System.nanoTime() ;
              Bisection bs = new Bisection( func,0,100);
              System.out.printf("\n  Root  of  your  equation
              (Bisection Algorithm) = %f\n",bs.getRoot());
              usedTime = System.nanoTime() - start;
       System.out.println("\nExecuted time = "+usedTime+" ns");
                     //========================================
               start = System.nanoTime() ;
               FalsePosition fp  = new FalsePosition
               (   func,0,100);
              System.out.printf("\n Root of your equation (False
              Position Method) = %f\n",fp.getRoot());
              usedTime = System.nanoTime() - start;
       System.out.println("\nExecuted time = "+usedTime+" ns");
                     //========================================
              start = System.nanoTime() ;
              NewtonRaphson nr  = new NewtonRaphson( func,0);
              System.out.printf("\n  Root  of  your  equation  (
              Newton Raphson method) = %f\n",nr.getRoot());
              usedTime = System.nanoTime() - start;
       System.out.println("\nExecuted time = "+usedTime+" ns");
                     //========================================
              start = System.nanoTime() ;
              Secant sc = new Secant( func,0, 100);
              System.out.printf("\n  Root  of  your  equation  (
              Secant method) = %f\n",sc.getRoot());
              usedTime = System.nanoTime() - start;
       System.out.println("\nExecuted time = "+usedTime+" ns");

              }
              catch (Exception e)
              { System.out.println(" Error(s) : "+ e);
              }
       }
 }
```

ผลลัพธ์ของโปรแกรม พบว่าการหารากสมการแบบไม่เป็นเชิงเส้นทั้ง 4 วิธี จะให้คำตอบออกมาเท่ากัน แต่วิธีแบ่งครึ่งช่วง จะมีข้อความแจ้งว่า มีการวนรอบเกินกว่าที่กำหนดไว้คือ 50 รอบ

เวลาที่ใช้ในการประมวลผลหาคำตอบสำหรับตัวอย่างนี้ วิธีเซแคนต์ จะใช้เวลาน้อยที่สุด ขณะที่วิธีแบ่งครึ่งช่วงจะใช้เวลามากที่สุด

```
MathTools.RootUtils.ExceedLoopException: The iterations more than the limit.
 Root of your equation (Bisection Algorithm) = 0.739085
        Executed time = 39668729 ns
 Root of your equation (False Position Method) = 0.739085
        Executed time = 1105448 ns
 Root of your equation ( Newton Raphson method) = 0.739085
        Executed time = 980851 ns
 Root of your equation ( Secant method) = 0.739085
        Executed time = 904305 ns
```

<u>ตัวอย่าง 4.2</u> อัดแก๊สไฮโดรเจนจำนวน n กิโลโมล ที่อุณหภูมิคงที่ที่ 300 เคลวิน ปริมาตรของแก๊สเปลี่ยนจาก V_1 = 10 ลูกบาศก์เมตร เป็น V_2 = 0.1 ลูกบาศก์เมตร ให้แก๊สนี้มีคุณสมบัติเป็นไปตามกฎสถานะแก๊สของแวน เดอร์ วาลล์ พบว่างานที่ใช้ในการอัดแก๊สนี้เท่ากับ 1153 กิโลจูล จงหาปริมาณของแก๊สไฮโดรเจน (n)

<u>วิธีทำ</u> จากกฎสถานะแก๊สของแวนเดอร์ วาลล์ จะได้สมการความดันของแก๊สดังนี้

$$P = \frac{nRT}{V - nb} - \frac{an^2}{V^2}$$

เมื่อ R คือค่าคงที่ของแก๊ส = 8.314 kJ/kmol.K

a, b คือค่าคงที่ของแก๊สขึ้นอยู่กับชนิดของแก๊ส สำหรับแก๊สไฮโดรเจน

a = 24.8 kPa.m^6/$kmol^2$ b = 0.0266 m^3/kmol

T คือ อุณหภูมิสมบูรณ์

งานที่ให้แก่ระบบจนปริมาตรแก๊สเปลี่ยนจาก V_1 ไปเป็น V_2 คือ

$$W = \int_{V_1}^{V_2} PdV = nRT\ln(\frac{V_2 - nb}{V_1 - nb}) + an^2(\frac{V_1 - V_2}{V_1V_2})$$

แทนค่าต่าง ๆ ลงในสมการจะได้

$$9.9 \times 24.8n^2 + 2494.2n\ln(\frac{10 - 0.0266n}{0.1 - 0.0266n}) - 1153 = 0$$

ต้องการหารากสมการ n

การหารากสมการนี้จะใช้ 3 วิธี คือ วิธีแบ่งครึ่งช่วง วิธีวางตำแหน่งผิดที่ และวิธีเซแคนต์ (ส่วนวิธีของนิวตัน-ราฟสัน ไม่ใช้ในที่นี้เพราะการหาอนุพันธ์อันดับที่1 ของฟังก์ชันนี้ค่อนข้างซับซ้อน) กำหนดค่าเริ่มต้นของรากสมการไว้ที่ 0.01 ถึง 1 และเปลี่ยนแปลงค่าฟังก์ชันดังนี้

```
Function func = new Function() {
        public double Of(double x)
        {return(9.9*24.8*x*x+ 2494.2*x*log((10-0.0266*x)/(0.1-
0.0266*x))-1153 );      }
};
```

ผลการทำงานของโปรแกรมจะได้ผลลัพธ์ซึ่งเป็นค่ารากสมการดังนี้

```
Root of your equation (Bisection Algorithm) = 0.099594

Root of your equation ( False Position Method) = 0.099594

Root of your equation ( Secant method) = 0.099594
```

ทั้งสามวิธีจะให้คำตอบเท่ากัน

4.2 การหาผลเฉลยของระบบสมการเชิงเส้น

รูปทั่วไปของระบบสมการเชิงเส้น หาสมการมีตัวไม่รู้ค่า n ตัว เขียนได้ดังนี้

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n = b_2 \\ \vdots \\ \vdots \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots a_{mn}x_n = b_m \end{array}$$

เขียนให้อยู่ในรูปเมตริกซ์ โดยกำหนดให้

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

จะได้

$$Ax = b \qquad \text{..........(4.4)}$$

A เป็นเมตริกซ์ของสัมประสิทธิ์ x เป็นเวกเตอร์ของตัวแปร b เป็นเวกเตอร์ของค่าคงที่

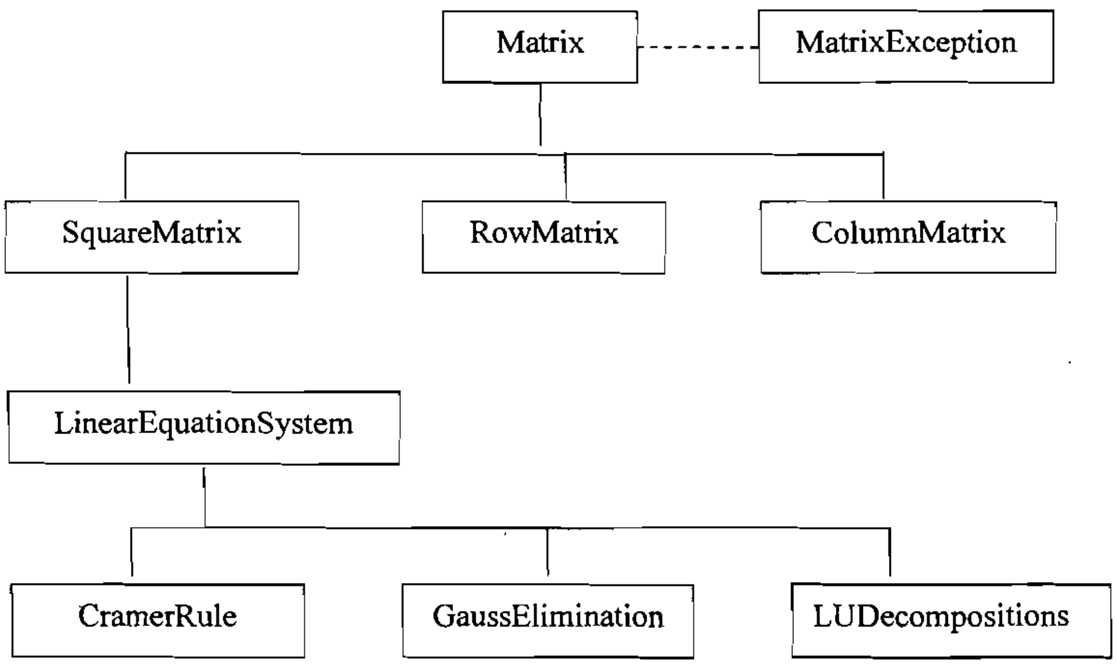
ถ้าค่าคงที่ทางด้านขวามือเป็นศูนย์หมดทุกตัว b = 0 จะเรียกสมการแบบนี้ว่าเป็นสมการเอกพันธ์ (homogeneous equation) จะได้ผลเฉลยที่เห็นได้ชัดคือ ตัวแปรทุกตัว (x) มีค่าเท่ากับศูนย์ จัดเป็นผลเฉลยที่มีความสำคัญน้อย (trivial solution)

ถ้าจำนวนสมการไม่เท่ากับจำนวนตัวแปร $m \neq n$ ผลเฉลยสำหรับระบบสมการชุดนี้เป็นไปได้หลายกรณีคือ อาจจะมีคำตอบเพียง 1 ชุด (unique solution) หรืออาจไม่มีคำตอบที่สอดคล้องกับสมการชุดนี้เลย หรือสมการหนึ่งสมการใดหรือหลาย ๆ สมการอาจมีผลเฉลยหลายค่า (redundant) หรือผลเฉลยของสมการหนึ่งขึ้นอยู่กับผลเฉลยของอีกสมการหนึ่ง

ในที่นี้ ผู้วิจัยจะพัฒนาคลาสไลบรารี การหาผลเฉลยในกรณีที่มีจำนวนสมการเท่ากับจำนวนตัวแปรเท่านั้น และเป็นระบบสมการไม่เอกพันธ์ (Inhomogeneous equation)

เนื่องจากการหาผลเฉลยของระบบสมการเชิงเส้นต้องไปเกี่ยวข้องกับเมตริกซ์ ผู้วิจัยจึงออกแบบคลาส Matrix ให้เป็นคลาสแม่ มีเมธอดที่ทำหน้าที่ต่าง ๆ เกี่ยวกับปฏิบัติการเมตริกซ์ และมีคลาส MatrixException คอยดักจับความผิดพลาด คลาสที่สืบทอดมาจากคลาส Matrix รองลงมาคือคลาส SquareMatrix สำหรับจัดการเมตริกซ์ที่มีจำนวนแถวและจำนวนสดมภ์เท่ากัน คลาส RowMatrix และคลาส ColumnMatrix สำหรับจัดการเมตริกซ์ที่เป็น row vector และ column vector

คลาส LinearEquationSystem เป็นคลาสที่มีบทบาทหลักในการหาผลเฉลยระบบสมการเชิงเส้นสืบทอดคุณสมบัติมาจากคลาส SquareMatrix อีกทอดหนึ่ง คลาส Linear EquationSystem ยังแตกหน่อออกไปเป็นคลาส CramerRule, คลาส GaussElimination และคลาส LUDecomposition ซึ่งเป็นวิธีเฉพาะสำหรับการหาผลเฉลยของระบบสมการเชิงเส้น สามารถเขียนแผนผังแสดงการสืบทอดของคลาสได้ดังนี้



คลาส Matrix มีตัวแปรคลาสคือ rows, cols และ elements ซึ่งใช้เก็บจำนวนแถวและจำนวนสดมภ์ของเมตริกซ์ และ ค่าของสมาชิกเมตริกซ์แต่ละตัวตามลำดับ มีเมธอดที่เกี่ยวข้องกับการกระทำกับเมตริกซ์ทั้งปวง เช่น การเรียกดูและกำหนดค่าต่าง ๆ ของสมาชิกเมตริกซ์ การทำสำเนาเมตริกซ์ การกำหนดค่าเมตริกซ์เป็นแถว และเป็นสดมภ์ การบวก ลบ คูณ และ transpose

```
/* File: Matrix.java */
package MathTools.LinearAlgebra;

import static java.lang.Math.*;
import MathTools.LinearAlgebra.MatrixException;
import static MathTools.Common.CommonConstants.*;

public class Matrix {
/** rows of matrix */   int rows;
/** column of matrix */ int cols;
/** values of element of matrix */  double elements[][];

/**
    * Construct an m x n constant matrix.
    * @param m    Number of rows.
    * @param n    Number of colums.
  */
public Matrix(int m, int n) {
        if (m >0 ) rows = m; else rows =1;
        if (n > 0 )cols = n; else cols =1;
        elements = new double[rows][cols];
}
/**
    * Construct an m x n constant matrix.
    * @param m    Number of rows.
    * @param n    Number of colums.
```

```
    * @param c     scalar value was filled in the matrix.
    */
   public Matrix(int m, int n, double c) {
      rows = m;
      cols = n;
      elements = new double[m][n];
      for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
          elements[i][j] = c;
   }

  /**
  * Constructor : create a matrix from a 2 dimensional array.
  * @param val- values of array of matrix
  * @thrown MatrixException if some rows have different length
  */
  public Matrix (double[][] val) {
        rows = val.length;
        cols = val[0].length;
        elements = val;
  }
  /** default Constructor
  */
  public Matrix() { }
  /**
  * Get the nubmer of row of this matrix
  * @return number of row count
  */
  public int getRowCount() { return rows;}
  /**
  * Get the nubmer of column of this matrix
  * @return number of column count
  */
  public int getColumnCount() { return cols;}
  /**
  * Get the value of an element (i,j) in matrix
  *@param i the row index
  *@param j the column index
  *@return the value of an element at (i,j)
  *@thrown Matrix.MatrixException for index error
  */
  public double getValueAt(int i, int j) throws MatrixException {
        if ((i < 0 || i >= rows)||(j < 0 || j >= cols)){
        throw new MatrixException(MatrixException.BAD_INDEX);
        }
        return elements[i][j];
  }
  /**
  * Set the value of an element (i,j) in matrix
  *@param i the row index
  *@param j the column index
  *@param value the value
  *@thrown Matrix.MatrixException for index error
  */
  public void setValueAt(int i,int j,double value)throws
                                  MatrixException {
        if ((i < 0 || i >= rows)||(j < 0 || j >= cols)){
        throw new MatrixException(MatrixException.BAD_INDEX);
        }
         elements[i][j]=value;
  }
```

```


/** Get the values of all elements of this matrix
* @return the values of all elements of this matrix
*/
public double[][] getValueOfElements( ) {
      return elements;
}
/** Set the values of all elements of this matrix
* @return the values of all elements of this matrix are set.
*/

public void setValueOfElements(double[][] values){
      elements = values;
}
/**Set the values at indexed row of this matrix with row matrix
* @param rowIndex the index row ( a row to be set.)
* @ param rowmatrix the row vector
* *@thrown Matrix.MatrixException for index and dimesion errors
*/

public void setValueAtRow(int rowIndex,RowMatrix rowmatrix)
                    throws MatrixException {
      if (cols != rowmatrix.cols)
      throw new atrixException(MatrixException.BAD_DIMENSIONS);
      if (rowIndex < 0 || rowIndex >= rows)
      throw new MatrixException(MatrixException.BAD_INDEX);
      for (int j=0;j < cols ;j++ ){
      this.elements[rowIndex][j] = rowmatrix.elements[0][j];
      }
}
/**Set the values at indexed column of this matrix with column
 vector.
* @param rowIndex the index column
* @ param colmatrix the column vector
* @thrown Matrix.MatrixException for index  and dimesion errors
*/
public void setValueAtColumn(int colIndex,ColumnMatrix
             colmatrix) throws MatrixException {
      if (rows != colmatrix.rows)
      throw new atrixException(MatrixException.BAD_DIMENSIONS);
      if (colIndex < 0 || colIndex >= cols)
      throw new MatrixException(MatrixException.BAD_INDEX);
      for (int i=0;i < cols ;i++ ){
        this.elements[i][colIndex] = colmatrix.elements[i][0];
      }
}

/** Copy the values of all elements into  2 dimension array
* @ return the values of all elements in 2 dimension array.
*/
public double[][] copyValueOfElements() {
      double[][] values = new double [rows][cols];

      for(int i=0; i < rows; i++) {
            for(int j=0; j < cols; j++) {
                  values[i][j] = elements[i][j];
            }
      }
      return values;
}
```

```
//--------------------------------
//    A Collection of Matrix Operation
//---------------------------------

/** Add another matrix to this matrix
* @param matrix an another matrix to be added
* @return the sum matrix
* @throws MatrixException for Bad dimension of matrix
*/
public Matrix add(Matrix matrix) throws MatrixException {

      if ((rows == matrix.rows)&& (cols == matrix.cols)) {
                  double temp[][] = new double[rows][cols];
            for(int i = 0 ; i < rows; i++)
            for(int j =0; j < cols; j++)
            temp[i][j] = elements[i][j]+ matrix.elements[i][j];
            return new Matrix(temp);
      } else {
      throw new atrixException(MatrixException.BAD_DIMENSIONS);
      }//else
}
/** Substract another matrix to this matrix
* @param matrix an another matrix to be substracted
* @return the difference matrix
* @throws MatrixException for Bad dimension of matrix
*/
public Matrix substract(Matrix matrix) throws MatrixException {

      if ((rows == matrix.rows)&& (cols == matrix.cols)) {
                  double temp[][] = new double[rows][cols];
            for(int i = 0 ; i < rows; i++)
            for(int j =0; j < cols; j++)
            temp[i][j]= elements[i][j]- matrix.elements[i][j];
            return new Matrix(temp);
      } else {
   throw new MatrixException(MatrixException.BAD_DIMENSIONS);
      }//else
}
/** Transpose of this matrix
* @return the transposed matrix
*/
public Matrix transpose(){
      double temp[][] = new double[cols][rows];
      for (int i=0;i < rows ;i++ )
            for (int j=0;j < cols ;j++ )
                  temp[j][i] = elements[i][j];
      return new Matrix(temp);
}
/** Multiply this matrix by another matrix
* @param matrix multiplier
* @return a product matrix
* @throws MatrixException for Invalid dimension of matrix.
*/
public Matrix multiply(Matrix matrix) throws MatrixException {

      if (cols == matrix.rows) {
      double product[][] = new double[rows][matrix.cols];
            for(int i = 0 ; i < rows; i++)
            for(int j =0; j < matrix.cols; j++) {
              double temp =0;
              for(int k = 0; k < cols; k++)
```

```
                temp = temp +elements[i][k]*matrix.elements[k][j];
                     product[i][j] = temp;
               } // for j
               return new Matrix(product);

         } else {
   throw new MatrixException(MatrixException.INVALID_MULTIPLY);
   }//else
   }

   /** Multiply this matrix by  scalar quantity.
   * @param c (a constant)
   * @return a product matrix
   */
   public Matrix multiply(double c)  {

         double product[][] = new double[rows][cols];
               for(int i = 0 ; i < rows; i++)
               for(int j =0; j < cols; j++)
                     product[i][j] = c * elements[i][j];
               return new Matrix(product);
   }
   /** Print  a matrix for test  */
   public void printMatrix( ) {
         for (int i = 0;i < rows ;i++ ){
               for (int j =0;j < cols ;j++ ) {
                     System.out.print( elements[i][j]+ " ");
               }
               System.out.println();
         }
   }
   }
```

คลาส SquareMatrix สืบทอดมาจากคลาส Matrix เพิ่มเมธอดต่าง ๆ ที่เกี่ยวข้องกับเมตริกซ์จัตุรัส เช่น การหาตัวกำหนด (determinant) การทำเมตริกซ์ผกผัน(inverse) ส่วนปฏิบัติการเมตริกซ์บวก ลบ คูณ และ transpose ไม่จำเป็นต้องเขียนชุดคำสั่งใหม่

```
/* File: SquareMatrix.java*/
package MathTools.LinearAlgebra;

import static java.lang.Math.*;
import static MathTools.Common.CommonConstants.*;
import MathTools.LinearAlgebra.MatrixException;
public class SquareMatrix extends Matrix {
      /** Constructor.
      * @param n number of rows and columns
      */
      public SquareMatrix(int n){
            super(n,n);
      }
      /** Constructor.
      * @param val the value  of elements
      */
```

```
        public SquareMatrix(double[][] val){
              setValue(val);

        }
        /** Constructor.
        * @param matrix the matrix n row  1 columms.
        */
        public SquareMatrix(Matrix matrix){
        this.rows = this.cols= min(matrix.rows, matrix.cols);
        this.elements= matrix.elements;
        }
        /** Constructor.
        * defalut Constructor.
        */

        public SquareMatrix() {
              super();
        }
        /** Get the number of elements ( Or row) of this matrix.
        * @return the size of this column vector
        */
        public int getSquareMatrixSize() {
              return rows;
        }
/**
* get the value of an element (i,j) in matrix
*@param i the row index
*@param j the column index
*
*/
public double getValueAt(int i, int j) throws MatrixException {
      return(super.getValueAt(i,j));
}


/* set this matrix from  a two dimension array of value.
* @param val a two dimension array of value.
*/
void setValue(double values[][]) {

      rows = values.length;
      cols = values[0].length;
      this.elements = values;
      rows = cols = min(rows,cols);
}
/**
* Set the value of an element (i,j) in matrix
*@param i the row index
*@param j the column index
*@param value the value
*@thrown Matrix.MatrixException for index error
*/
public void setValueAt(int i, int j, double value)
                        throws MatrixException {
      super.setValueAt(i,j,value);
}
 /**
* Generate identity matrix
* @param n    Number of rows, and columns
* @return A square matrix with 1 on diagonal and 0 elsewhere.
   */
```

```
  public static SquareMatrix identity(int n) {
    double[][] temp = new double[n][n];
    for (int i = 0; i < n; i++)
            for(int j =0; j < n ; j++)
                  temp[i][j] = (i == j ? 1.0 : 0.0);
     return new SquareMatrix(temp);
  }
      // ********** Square Matrix Operation *************
/** Add another square matrix to this square matrix
* @param sq_matrix an another column vector to be added
* @return the sum column vector
* @throws MatrixException for Bad dimension of row vector
*/
public SquareMatrix add(SquareMatrix sq_matrix)
                              throws MatrixException {
      return new SquareMatrix(super.add(sq_matrix));
}
/** Substract another square matrix to square matrix
* @param sq_matrix an another square matrix to be substracted
* @return the difference square matrix
* @throws MatrixException for Bad dimension of matrix
*/
public SquareMatrix substract(SquareMatrix sq_matrix)
                  throws MatrixException {
      return new SquareMatrix(super.substract(sq_matrix));
}
/** Multiply another square matrix to square matrix
* @param sq_matrix an another square matrix to be multiplied
* @return the difference square matrix
* @throws MatrixException for Bad dimension of matrix.
*/
public SquareMatrix multiply(SquareMatrix sq_matrix)
                          throws MatrixException {
      return new SquareMatrix(super.multiply(sq_matrix));
}
/** Transpose  square matrix
* @return the transposed square matrix
*/
public SquareMatrix transpose() {
      return new SquareMatrix(super.transpose());
}

/** Multiply this matrix by  scalar quantity.
* @param c (a constant)
* @return a product matrix
*/
public SquareMatrix multiply(double c)  {
      return new SquareMatrix(super.multiply(c));
}
/** Evaluate the determinant of this  square matrix.
*   (no pivot version)
* @return the determinant
*
*/
public double determinant() {
double tmp=0;
double multiplier=0;
double det = 1.0;
double[][] values = new double[rows][cols];

      /* copy value of this matrix */
```

```
                values = this.copyValueOfElements();
                values = super.makeUpperTriangular();

        for (int i = 0; i < rows; i++ )
          det *= values[i][i];
//      if (pivotChecked) det = -det;
        return det;
}


/** Evaluate the inverse of this  square matrix
* @return the inverse matrix
* throws MatrixException if error occurred
*/
public SquareMatrix inverse() throws MatrixException{
int pivot;
double temp=0,multiplier=0;
double[][] values = new double[rows][cols];
double[][] dummy = new double[rows][2*rows];
      // copy values of this matrix into dummy matrix
      for(int i=0; i < rows; i++)
            for(int j=0; j < rows; j++)
                  dummy[i][j] = elements[i][j];
      // append dummy matrix with identity matrix
      for( int i= 0; i < rows; i++) {
            for (int j=rows; j < 2*rows; j++){
                if ( (j - i) == rows) dummy[i][j]    = 1.0;
                else    dummy[i][j]= 0.;
            }
      }
 for (int i=0; i < (rows-1); i++) {
        pivot = i;
        /* Check the maximum element in the same column */
        for (int j = i+1; j < rows; j++)
       if (abs(dummy[pivot][i]) < abs(dummy[j][i])) pivot=j;
        /* Pivoting */
        if (pivot != i) {
             for (int j =0;j < (2*rows); j++) {
               temp = dummy[i][j];
               dummy[i][j] = dummy[pivot][j];
               dummy[pivot][j] = temp;
             }
        }
        if (dummy[i][i] == 0) {
 throw new MatrixException(MatrixException.SINGULAR_MATRIX); }
 // make an upper triangular matrix
       for (int r = i+1;r < rows; r++) {
            if(dummy[r][i] != 0 ) {
                  multiplier= dummy[r][i]/dummy[i][i];
              for(int c = 0;c < (2*rows);c++) {
            dummy[r][c]= dummy[r][c]- multiplier*dummy[i][c];
                  if (abs(dummy[r][c]) <= ZERO_APPROACH)
                              dummy[r][c]= 0;
              }// for (int c
            } /* if */
       } // for ( r....)
       } //for (i..)
 // Make a Matrix lower triangular
     for (int i = rows-1; i > 0; i--) {
        for ( int r = i-1;r >= 0; r--) {
            if(dummy[r][i] != 0 ) {
```

```
                    multiplier= dummy[r][i]/dummy[i][i];
                      for(int c = i;c < 2*rows;c++) {
                  dummy[r][c] = dummy[r][c]-multiplier*dummy[i][c];
                          if (abs(dummy[r][c]) <= ZERO_APPROACH)
                                      dummy[r][c] = 0;
                    }
                 } /* if */
            } /* for ( r....) */
       } /* for(i.. */
  // Create Identity Matrix
      for ( int i= rows-1 ; i >=0; i--) {
              temp = dummy[i][i];
        for (int j = i; j < 2*rows; j++) {
              dummy[i][j] /= temp;
              }
        }
  /* copy to square matrix */
      for ( int i= 0 ; i < rows; i++) {
              int k =0;
              for (int j = rows; j < 2*rows; j++) {
              values[i][k] = dummy[i][j];
              k = k+1;
              }
        }
  return(new SquareMatrix(values));
   }
  }
```

คลาส LinearEquationSystem สืบทอดมาจากคลาส SquareMatrix อีกทอดหนึ่ง เมธอดต่าง ๆ จากคลาส SquareMatrix และ คลาส Matrix ยังคงเรียกใช้ได้โดยไม่ต้องเขียนเมธอดขึ้นมารองรับใหม่ เมธอด determinant (บรรทัดที่ 99 – 116)ได้ปรับปรุงแก้ไขใหม่โดยให้มีการสลับที่แถวสมาชิกที่มีค่ามากกับแถวที่สมาชิกมีค่าน้อย(pivoting) เพื่อกระจัดความคลาดเคลื่อนในการคำนวณ การเขียนเมธอด determinant ในคลาส LinearEquationSystem นี้เรียกว่าเป็นการ override เมธอดของคลาสแม่ ให้เหมาะสมและสอดคล้องกับการใช้งานในคลาสลูกมากยิ่งขึ้น นอกจากนี้ยังมีการ overloading เมธอด determinant (บรรทัดที่ 117-141) มีการแทนค่าสมาชิกในแนวสดมภ์ของ determinant ตามที่กำหนดไว้ใน argument เพื่อเตรียมไว้ใช้กับกฎของคราเมอร์อีกด้วย เมธอดที่เพิ่มขึ้นมาเพื่อใช้สำหรับหาผลเฉลยได้แก่ forwardElimination() (บรรทัดที่ 49-94) และ backSubstitution( ) (บรรทัด 146-158)

```
 1:  /* File: LinearEquationSystem.java */
 2:  package MathTools.LinearAlgebra;
 3:
 4:  import static java.lang.Math.*;
 5:  import static MathTools.Common.CommonConstants.*;
 6:  import MathTools.LinearAlgebra.MatrixException;
 7:  //import MathTools.LinearAlgebra.ColumnMatrix;
 8:  public class LinearEquationSystem extends SquareMatrix {
 9:         /** count the number of pivoting */ int pivotCount=0 ;
10:         /**  right- hand constants */    ColumnMatrix rhConstant;
```

```
        /** dummy of two array  for temporaly value */
                              SquareMatrix dummy;
        /** dummy of an array  for temporaly value */
                              ColumnMatrix b;
        /** Constructor.
        * @param n number of linear equation
        */
        public LinearEquationSystem (int n) {
              super(n);

              rhConstant = new ColumnMatrix(n);
        }
  /** Constructor.
  * @param augment the matrix of coefficient and their constants.
  */
        public LinearEquationSystem (double[][] coefficient) {
              super(coefficient);
              rhConstant = new ColumnMatrix();
        }
  /** Constructor.
  * @param matrix the matrix n row n columms of coefficient.
  */
        public LinearEquationSystem (SquareMatrix matrix){
              super(matrix);
              rhConstant = new ColumnMatrix(rows);
        }
  /** Constructor.
  * @param matrix the matrix n row n columms of coefficient.
  * @param bmatrix the right hand constant vector
  */
        public LinearEquationSystem (SquareMatrix matrix,
                              ColumnMatrix bmatrix){
              super(matrix);
              rhConstant = new ColumnMatrix(bmatrix);

  }
  protected void setDummyArray(){
        dummy = new SquareMatrix(this.copyValueOfElements());
  if(rhConstant != null)
  b = new ColumnMatrix( rhConstant.copyValueOfColumnMatrix());
  }
  protected void forwardElimination(boolean
                        with_rhConstant)throws MatrixException{
  int pivot;
  double multiplier;
  //Pivot loop start here ============>
        for (int rowPivot =0; rowPivot < rows-1; rowPivot++) {
                    pivot = rowPivot;
              // Check the maximum element in the same column.
                    for(int r= rowPivot+1; r < rows; r++)
                    if(abs(dummy.getValueAt(pivot,rowPivot)) <
                          abs(dummy.getValueAt(r,rowPivot)))
                                pivot = r;
              // pivoting rows for get the best element.
              if(pivot != rowPivot) {
                    for(int c = 0; c < cols; c++) {
        double temp = dummy.getValueAt(rowPivot,c);
        dummy.setValueAt(rowPivot,c,dummy.getValueAt(pivot,c));
        dummy.setValueAt(pivot,c, temp);
                    }
                    if (with_rhConstant){
```

```
                                double temp = b.getValueAt(rowPivot);
                        b.setValueAt(rowPivot,b.getValueAt(pivot));
                        b.setValueAt(pivot,temp);
                        }
                                pivotCount += 1;
                        }
                        if(dummy.getValueAt(rowPivot,rowPivot)==0)
throw new MatrixException(MatrixException.SINGULAR_MATRIX);
        // make an upper triangular matrix
                        for(int r = rowPivot+1; r < rows; r++){
                        if(dummy.getValueAt(r,rowPivot) != 0){
                        multiplier = dummy.getValueAt(r,rowPivot)
                                /dummy.getValueAt(rowPivot,rowPivot);
                        for(int c = 0; c < cols;c++){
                                double temp = dummy.getValueAt(r,c)-
                        multiplier*dummy.getValueAt(rowPivot,c);
                        if(abs(temp) <= ZERO_APPROACH) temp = 0;
                                dummy.setValueAt(r,c, temp);
                        }// for int c
                        if (with_rhConstant){
                        double temp = b.getValueAt(r)-
                                multiplier*b.getValueAt(rowPivot);
                        b.setValueAt(r,temp);
                        }
                        }//if (dummy....
                        }//for r
                }//for rowPivot
        // check the lower most of element of matrix is zero ?
        //      if(dummy.getValueAt(rows-1,rows-1)==0)
//throw new MatrixException(MatrixException.SINGULAR_MATRIX);
}
/** Evaluate the determinant of this  coefficient matrix.
*  with partially row pivot.
* @return the determinant
*/
public double determinant() {
double det = 1.0;
                pivotCount = 0; // initialize pivot count
                setDummyArray();
                try{
                        forwardElimination(false);
                }catch ( MatrixException msg){
                        System.out.println(msg);
                }
                try {
                for (int r = 0; r < rows; r++ )
                        det *= dummy.getValueAt(r,r);
                }catch (MatrixException msg) {
                        System.out.println(msg);
                }
                if(pivotCount%2 != 0) det = -det;
                return det;
}
/** Evaluate the determinant of this  coefficient matrix.
*  but replace colIndex column with right-hand constants.
* @return the determinant
*/
public double determinant(int colIndex) {
double det = 1.0;
                pivotCount = 0; // initialize pivot count
                setDummyArray();
```

```
            try{
            dummy.setValueAtColumn(colIndex, b);
            }catch( MatrixException msg) { }
            try{
                  forwardElimination(false);
            }catch ( MatrixException msg){
                  System.out.println(msg);
            }
            try {
            for (int r = 0; r < rows; r++ )
                  det *= dummy.getValueAt(r,r);
            }catch (MatrixException msg) {
                  System.out.println(msg);
            }
            if(pivotCount%2 != 0) det = -det;
            return det;
}
/** Solve Ax = b for x by back substitution
*  @param -- Column vector b (right hand constants)
* @ return -- the solution of column matrix
*/
protected ColumnMatrix backSubstitution(ColumnMatrix b)
                     throws MatrixException{
      ColumnMatrix x = new ColumnMatrix(rows);
        /* Backward Substitution */

     for (int r = rows-1; r >=0; r--) {
       double temp = b.getValueAt(r);
       for (int c = r; c < cols-1; c++) {
       temp = temp - dummy.getValueAt(r,c+1)*x.getValueAt(c+1);
     }
       x.setValueAt(r, temp/dummy.getValueAt(r,r));
}
   return x;
}
}
```

4.2.1 กฎของคราเมอร์ (Cramer's Rule)

การหาผลเฉลยของระบบสมการเชิงเส้น ซึ่งมี 3 ตัวแปร เขียนสมการได้ดังนี้

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

เมื่อใช้กฎของคราเมอร์จะได้ผลเฉลยของตัวแปรแต่ละตัวดังนี้

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}}, \quad x_2 = \frac{\begin{vmatrix} a_{11} & b_1 & a_{13} \\ a_{21} & b_2 & a_{23} \\ a_{31} & b_3 & a_{33} \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}}, \quad x_3 = \frac{\begin{vmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ a_{31} & a_{32} & b_3 \end{vmatrix}}{\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}}$$

เขียนเป็นสมการทั่วไปได้ดังนี้

$$x_{ji} = \frac{|A_j|}{|A|}$$

เมื่อ A เป็นเมตริกซ์ของสัมประสิทธิ์ของตัวแปร โดยที่ $|A|$ เท่ากับศูนย์

A_j เป็นเมตริกซ์ขนาด $n \times n$ เมื่อ n เป็นจำนวนสมการ โดยแทนที่แถวตั้ง (column) ที่ j ด้วยค่าคงที่ b

$$A_j = \begin{vmatrix} a_{11} & a_{12} & \cdots & b_1 & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & b_2 & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \cdots & b_n & \cdots & a_{3n} \end{vmatrix}$$

แทนค่า แถวตั้งที่ j ด้วยค่าคงที่

คลาส CramerRule จะเป็นคลาสที่มีชุดคำสั่งสั้น ๆ เพราะสืบทอดมาจากคลาส Linear Equation system จึงสามารถใช้เมธอดต่าง ๆ ที่อยู่ในคลาสของบรรพบุรุษที่สืบทอดมาได้เลย (ได้แก่ เมธอดการหาตัวกำหนด (determinant) เป็นต้น) รายละเอียดของคลาส Cramer Rule มีดังนี้คลาส CramerRule จะเป็นคลาสที่มีชุดคำสั่งสั้น ๆ เพราะสืบทอดมาจากคลาส LinearEquationSystem จึงสามารถใช้เมธอดต่าง ๆ ที่อยู่ในคลาสของบรรพบุรุษที่สืบทอดมาได้เลย (ได้แก่เมธอดการหาตัวกำหนด(determinant) เป็นต้น) รายละเอียดของคลาสCramerRule มีดังนี้

```
/* File: CramerRule.java */
package MathTools.LinearAlgebra;

import static java.lang.Math.*;
```

```
import static MathTools.Common.CommonConstants.*;
import MathTools.LinearAlgebra.MatrixException;
public class CramerRule extends LinearEquationSystem {
      ColumnMatrix x = new ColumnMatrix(rows);
      double delta = 0;
/** Constructor.
* @param matrix the coefficient matrix (n x n size.)
*@param bconstant the right hand side column vector
*/
public CramerRule(SquareMatrix matrix, ColumnMatrix bconstant)
                  throws MatrixException{
      super(matrix,bconstant);
      if (rows > ARRAY_SIZE || cols > ARRAY_SIZE)
throw new MatrixException(MatrixException.ARRAY_SIZE_EXCEED);
      }

      public ColumnMatrix solve() throws MatrixException{
            setDummyArray();
            delta = determinant();
            for(int i=0; i < rows ; i++){
                  double temp = 0;
                  temp = determinant(i);
                  x.setValueAt(i,temp/delta);
                  dummy.elements = this.copyValueOfElements();
            }
            return x;
   }
}
```

4.2.2 **การลดทอนแบบเกาส์** (Gauss Elimination)

ระเบียบวิธี(Algorithm) การลดทอนของเกาส์สามารถอธิบายได้โดยประกอบกับตัวอย่างดังต่อไปนี้

<u>ตัวอย่าง</u> <u>4.2.1</u> จงหาผลเฉลยของระบบสมการเชิงเส้นโดยใช้การลดทอนของเกาส์

$$2x - 2y + z = 1$$

$$4x - 2y + z = 3$$

$$x + y - z = 0$$

<u>วิธีทำ</u>

ขั้นที่ 1 นำเมตริกซ์ของสัมประสิทธิ์และเวกเตอร์ค่าคงที่มาเขียนให้อยู่ในรูปของเมตริกซ์แต่งเติมแล้ว (augment matrix) $(A|B)$

$$A = \begin{bmatrix} 2 & -2 & 1 \\ 4 & -2 & 1 \\ 1 & 1 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}$$

$$(A|B) = \left[\begin{array}{ccc|c} 2 & -2 & 1 & 1 \\ 4 & -2 & 1 & 3 \\ 1 & 1 & -1 & 0 \end{array}\right]$$

ขั้นที่ 2 เริ่มลดทอนตัวแปรโดยวิธีของเกาส์ ในที่นี้จะใช้วิธีลดทอนไปข้างหน้า

- ใช้แถวนอนแถวที่ 1 เป็นหลักนำ 2 คูณสมาชิกทุกตัวของแถวนอนที่ 1 แล้วนำไปลบกับแถวที่ 2 (เขียนเป็นสัญลักษณ์ได้เป็น $R_2 \leftarrow R_2 - 2 \times R_1$)

$$\left[\begin{array}{ccc|c} 2 & -2 & 1 & +1 \\ 0 & 2 & -1 & +1 \\ 1 & 1 & -1 & 0 \end{array}\right]$$

- นำ 0.5 คูณสมาชิกทุกตัวของแถวนอนแถวแรก แล้วไปลบแถวนอนที่ 3 ($R_3 \leftarrow R_3 - 0.5 \times R_1$)

$$\left[\begin{array}{ccc|c} 2 & -2 & -1 & 1 \\ 0 & 2 & -1 & 1 \\ 0 & 2 & -1.5 & -0.5 \end{array}\right]$$

ขั้นที่ 3 ใช้แถวนอนที่ 2 เป็นหลัก

- นำแถวนอนที่ 3 ตั้งลบด้วยแถวนอนที่ 2

$$\left[\begin{array}{ccc|c} 2 & -2 & -1 & 1 \\ 0 & 2 & -1 & 1 \\ 0 & 0 & -0.5 & -1.5 \end{array}\right]$$

ขั้นที่ 4 ใช้วิธีแทนค่าป้อนกลับ จะได้

$$-0.5z = -1.5$$

$$z = 3$$

แทนค่า Z ในแถวนอนที่ 2 จะได้ค่า y

$$2y - 1 \times 3 = 1$$

$$y = 2$$

แทนค่า y, z ลงในแถวนอนที่ 1 จะได้

$$2x - 2 \times 2 - 1 \times 3 = 1$$

$$x = 1$$

ผลเฉลยของสมการชุดนี้คือ $x = 1, y = 2, z = 3$

ผู้วิจัยได้ออกแบบคลาส GaussElimination โดยให้สืบทอดมาจาก คลาส LinearEquationSystem เรียกใช้เมธอด forwardElimination () และ backSubstitution( ) จากคลาสแม่มาใช้แก้สมการหาผลเฉลย รายละเอียดของคลาส GaussElimination มีดังนี้

```
/* File: GaussElimination.java  */
package MathTools.LinearAlgebra;

import static java.lang.Math.*;
import static MathTools.Common.CommonConstants.*;
import MathTools.LinearAlgebra.MatrixException;
public class GaussElimination extends LinearEquationSystem {
      ColumnMatrix x = new ColumnMatrix();
/** Constructor.
* @param n number of linear equation.
*/
      public GaussElimination(int n)throws MatrixException{
      super(n);
      if (rows > ARRAY_SIZE || cols > ARRAY_SIZE)
throw new MatrixException(MatrixException.ARRAY_SIZE_EXCEED);
      }
/** Constructor.
* @param augment the matrix of coefficient and their constants.
*/
      public GaussElimination(double[][] coefficient)
            throws MatrixException {
            super(coefficient);
      if (rows > ARRAY_SIZE || cols > ARRAY_SIZE)
throw new MatrixException(MatrixException.ARRAY_SIZE_EXCEED);
      }
      /** Constructor.
      * @param matrix the matrix n x n size.
      */
      public GaussElimination(SquareMatrix matrix)
                  throws MatrixException{
            super(matrix);
```

```

        if (rows > ARRAY_SIZE || cols > ARRAY_SIZE)
 throw new MatrixException(MatrixException.ARRAY_SIZE_EXCEED);
        }
        /** Constructor.
        * @param matrix the coefficient matrix (n x n size.)
        *@param bconstant the right hand side column vector
        */
 public GaussElimination(SquareMatrix matrix,
         ColumnMatrix bconstant)throws MatrixException{
              super(matrix,bconstant);
              if (rows > ARRAY_SIZE || cols > ARRAY_SIZE)
 throw new MatrixException(MatrixException.ARRAY_SIZE_EXCEED);
        }

        public ColumnMatrix solve() throws MatrixException{
              setDummyArray();
              try{
                    forwardElimination(true);
              }catch ( MatrixException msg){
                    System.out.println(msg);
              }
              x = backSubstitution(b);
              return x;
    }
 }
```

4.2.3 วิธีแยกเป็นเมตริกซ์สามเหลี่ยมล่างและบน (LU Decomposition)

เมตริกซ์ของสัมประสิทธิ์ (A) จะถูกแยกออกเป็นสามเหลี่ยมล่าง (L, Lower triangular matrix) และเมตริกซ์สามเหลี่ยมบน (U- Upper triangular matrix) โดยที่

$$A = LU$$

การแยกเมตริกซ์ A ให้อยู่ในรูป L และ U สามารถทำได้เสมอ ถ้าเมตริกซ์นั้นไม่เป็นเมตริกซ์เอกฐาน (singular matrix) นั่นคือขนาดของตัวกำหนด (determinant) ไม่เป็นศูนย์ สามารถทำให้เข้าใจได้ง่าย โดยจะยกตัวอย่างเมตริกซ์ A ซึ่งมีขนาด 3 x3

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

เมตริกซ์ L จะมีสมาชิกในแนวเส้นทแยงมุม มีค่าเป็น 1 ทั้งหมด จะหาค่า l_{ij} และ u_{ij} โดยการคูณเมตริกซ์ L และ U เข้าด้วยกัน

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ \ell_{21}u_{21} & \ell_{21}u_{21}+u_{22} & \ell_{22}u_{22} \\ \ell_{31}u_{31} & \ell_{31}u_{31}+u_{32} & \ell_{32}u_{32} \end{bmatrix}$$

เปรียบเทียบกันระหว่างเทอมต่อเทอมที่สมนัยกัน

$$u_{11} = a_{11}, \qquad u_{12} = a_{12}, \qquad u_{13} = a_{13}$$

หรือ $u_{ij} = a_{ij}$ เมื่อ $j = 1$ ถึง 3

$$a_{21} = l_{21}u_{11} \quad \text{และ} \quad a_{31} = l_{31}u_{11}$$

นั่นคือ $l_{21} = \dfrac{a_{21}}{u_{11}}$ และ $l_{31} = \dfrac{a_{31}}{u_{11}}$

$$a_{22} = l_{21}u_{12} + u_{22}, \quad a_{23} = l_{21}u_{13} + u_{23}$$

จะได้ $u_{22} = a_{22} - l_{21}u_{12}$, $u_{23} = a_{23} - l_{21}u_{13}$

$$a_{32} = l_{31}u_{12} + l_{32}u_{22}$$

หรือ $l_{32} = \dfrac{(a_{32} - l_{31}u_{12})}{u_{22}}$

$$a_{33} = l_{31}u_{13} + l_{32}u_{23} + u_{33}$$

หรือ $u_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23}$

เมื่อขยายผลไปจนถึงลำดับที่ N

1. แถวนอนแถวแรกของ U คือ $u_{1,j}$ เมื่อ $j = 1$ ถึง N หาได้จาก

$$u_{1j} = a_{1j} \qquad j = 1 \text{ ถึง } N$$

2. แถวตั้งแถวแรกของ L คือ $l_{i,1}$ เมื่อ $i = 2$ ถึง N หาได้จาก

$$l_{i,1} = \frac{a_{i1}}{u_{11}} \qquad \text{เมื่อ } i = 2 \text{ ถึง } N$$

3. แถวนอนที่ 2 ของ U หาได้จาก

$$u_{2j} = a_{2j} - l_{21}u_{1j} \quad \text{เมื่อ } j = 2 \text{ ถึง } N$$

4. แถวตั้งที่ 2 ของ l คือ $l_{i,2}$ หาได้จาก

$$l_{i,2} = \frac{(a_{i2} - l_{i1}u_{12})}{u_{22}} \qquad i = 3 \text{ ถึง } N$$

5. แถวนอนที่ n ของ U หาได้จาก

$$u_{nj} = a_{nj} - \sum_{k=1}^{n-1} l_{nk}u_{kj} \qquad j = n \text{ ถึง } N$$

6. แถวตั้งที่ n ของ L หาได้จาก

$$l_{in} = \frac{(a_{in} - \sum_{k=1}^{n-1} l_{ik}u_{kn})}{u_{nn}} \qquad \text{เมื่อ } i = n+1 \text{ ถึง } N$$

เมื่อแยกเมตริกซ์ A ออกเป็นเมตริกซ์ L และ U แล้ว ขั้นตอนวิธีหาผลเฉลยต่อไปทำได้ดังนี้

จากสมการ $Ax = b$

แทนค่า $A = LU$ จะได้

$LUx = b$

กำหนดให้ $Ux = w$ สมการจะกลายเป็น

$Lw = b$

เนื่องจากสมาชิกในแนวเส้นทแยงมุมของ L มีค่าเป็น 1 ทุกตัว จึงใช้วิธีแทนค่าไปข้างหน้า หาค่า w ได้

ในกรณีที่เมตริกซ์ของสัมประสิทธิ์มีขนาด 3 x 3 ขั้นตอนวิธีหาผลเฉลยจะเป็นดังต่อไปนี้

$$\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

ใช้วิธีแทนค่าไปข้างหน้า

$$w_1 = b_1$$

$$w_2 = b_2 - w_1 l_{21}$$

$$w_3 = b_3 - w_1 l_{31} - w_2 l_{32}$$

แทนค่า w ลงในสมการ Ux = w ก็จะสามารถหาค่า x จะได้

การออกแบบคลาส LUDecomposition ได้ออกแบบให้สืบทอดมาจากคลาส LinearEquationSystem ทำให้สามารถใช้ตัวแปรคลาสได้ร่วมกัน และได้ override เมธอด backSubstitution() ให้เหมาะสมกับรูปแบบของเมตริกซ์ที่เปลี่ยนไปเป็นเมตริกซ์สามเหลี่ยมล่างและบน เมธอดที่เพิ่มเติมได้แก่ decompose() เป็นการแยก augment matrix ให้อยู่ในรูปผลคูณของเมตริกซ์สามเหลี่ยมล่างและสามเหลี่ยมบน forwardSubstitution( ) เป็นการแทนค่าไปข้างหน้าซึ่งมีใช้ในคลาสนี้เป็นครั้งแรก

รายละเอียดของคลาส LUDecomposition มีดังนี้

```
/* File: LUDecomposition.java */
package MathTools.LinearAlgebra;

import static java.lang.Math.*;
import static MathTools.Common.CommonConstants.*;
import MathTools.LinearAlgebra.MatrixException;
public class LUDecomposition extends LinearEquationSystem {
/** keeping solution in here */
            ColumnMatrix x = new ColumnMatrix(rows);
/** LU Lower and Upper triangular matrix in commpact */
            SquareMatrix LU = new SquareMatrix(rows);
      /** Constructor.
      * @param matrix the coefficient matrix (n x n size.)
      *@param bconstant the right hand side column vector
      */
      public LUDecomposition(SquareMatrix matrix,
             ColumnMatrix bconstant)throws MatrixException{
            super(matrix,bconstant);
            if (rows > ARRAY_SIZE || cols > ARRAY_SIZE)
 throw new MatrixException(MatrixException.ARRAY_SIZE_EXCEED);
      }

 public ColumnMatrix solve() throws MatrixException{
      ColumnMatrix y = new ColumnMatrix(rows);
            setDummyArray();
            decompose();
            y = forwardSubstitution(); // evaluate Ly = b
            x = backSubstitution(y); // evaluate Ux = y
      return x;
   }

 private void decompose() throws MatrixException {
      try{
            partialPivot(1);
      }
      catch (MatrixException err){ System.out.println( err);}
            // first row of U matrix
      for(int j = 0 ; j < cols ; j++)
                  LU.setValueAt(0,j,dummy.getValueAt(0,j));
      // first column of L matrix
      for(int i = 1; i < rows; i++)
                  LU.setValueAt(i,0,(dummy.getValueAt(i,0)
                       /LU.getValueAt(0,0)));

```

```
        for(int i =1; i < rows; i++){
        try{
              partialPivot(i);
        }catch (MatrixException err){ System.out.println (err);}
        // for n th row
              for(int j = i; j < cols ; j++) {
                    double temp = dummy.getValueAt(i,j);
                    for(int k =0; k <= i-1; k++){
                    temp = temp -
                          LU.getValueAt(i,k)*LU.getValueAt(k,j);
              }
              LU.setValueAt(i,j,temp);
              }
        // for n th column
        for(int m = i+1; m < rows ; m++){
              double      temp = dummy.getValueAt(m,i);
        for(int k = 0 ; k <= i-1; k++){
              temp = temp -
                    LU.getValueAt(m,k)*LU.getValueAt(k,i);
              if (abs(LU.getValueAt(i,i)) < ZERO_APPROACH)
  throw new MatrixException(MatrixException.SINGULAR_MATRIX);
              else LU.setValueAt(m,i, temp/LU.getValueAt(i,i));
        }// for(int k...
        } //for (int m...

        }//for (int i..
  }
  /** pivot the elements of matrix by exchange rows.
  * @param start_row the starting row of matrix to be pivoted.
  * @param values the elements of matrix to be partial pivoted.
  * @return true if some rows of matrix has been pivoted.
  */
  private void partialPivot(int rowPivot)throws MatrixException{
        /* Check the maximum element in the same column */
        int pivot = rowPivot;
         for(int r= rowPivot+1; r < rows; r++)
         if(abs(dummy.getValueAt(pivot,rowPivot)) <
                    abs(dummy.getValueAt(r,rowPivot)))
               pivot = r;
              // pivoting rows for get the best element.
              if(pivot != rowPivot) {
                    for(int c = 0; c < cols; c++) {
                    double temp = dummy.getValueAt(rowPivot,c);
                    dummy.setValueAt(rowPivot,c,
                          dummy.getValueAt(pivot,c));
                    dummy.setValueAt(pivot,c, temp);
                    temp = LU.getValueAt(rowPivot,c);
                    LU.setValueAt(rowPivot,c
                                ,LU.getValueAt(pivot,c));
                    LU.setValueAt(pivot,c, temp);
              }
         double temp = b.getValueAt(rowPivot);
         b.setValueAt(rowPivot,b.getValueAt(pivot));
         b.setValueAt(pivot,temp);
         }
  }
  /** Solve Ly = b by forward substitution.
  * @return the column matrix y
  * @throws LinearAlgebra.MatrixException if an error occured.
  */
  protected ColumnMatrix forwardSubstitution()
```

```
                    throws MatrixException {
         ColumnMatrix y = new ColumnMatrix(rows);
         for(int i = 0; i < rows; i++){
           double temp = b.getValueAt(i);
           for(int j = 0; j < i; j++){
                temp = temp - LU.getValueAt(i,j)*y.getValueAt(j);
                }
                y.setValueAt(i,temp);
         }
         return y;
}
/** Solve Ux = y by backward substitution
* @return the column matrix y
* @throws LinearAlgebra.MatrixException if an error occured.
*/
protected ColumnMatrix backSubstitution(ColumnMatrix y)
                    throws MatrixException{
         ColumnMatrix x = new ColumnMatrix(rows);
           /* Back Substitution */
       for (int i = rows-1; i >=0; i--) {
           double temp = y.getValueAt(i);
           for (int j = i; j < cols-1; j++) {
             temp = temp - LU.getValueAt(i,j+1)*x.getValueAt(j+1);
           }
             x.setValueAt(i, temp/LU.getValueAt(i,i));
            }
     return x;
}
}
```

4.2.5 การทดสอบหาผลเฉลยของระบบสมการเชิงเส้น

จากการพัฒนาวิธีหาผลเฉลยของระบบสมการเชิงเส้นทั้ง 3 วิธี คือ การใช้กฎคราเมอร์ การลดทอนแบบเกาส์ และ วิธีแยกเป็นเมตริกซ์สามเหลี่ยมล่างและบน ได้นำวิธีเหล่านี้ไปทดสอบกับปัญหาทางคณิตศาสตร์และทางฟิสิกส์

__ตัวอย่าง 4.2.1__ จงหาผลเฉลยของระบบสมการเชิงเส้นต่อไปนี้

$$3x - 2y + 2z + w = 5$$

$$2x + 4y - z - 2w = 3$$

$$3x + 7y - z + 3w = 23$$

$$x - 3y + 2z - 3w = -12$$

ผลเฉลยแท้จริงของระบบสมการชุดนี้คือ $x = 2, y = 1, z = -1, w = 3$

<u>**วิธีทำ**</u> เขียนอยู่ในรูปเมตริกซ์จะได้ดังนี้

$$\begin{bmatrix} 3 & -2 & 2 & 1 \\ 2 & 4 & -1 & -2 \\ 3 & 7 & -1 & 3 \\ 1 & -3 & 2 & -3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \\ 23 \\ -12 \end{bmatrix}$$

นำค่าต่าง ๆ ของ augment matrix และ ค่าคงที่ด้านขวามือ ไปใส่ลงในโปรแกรม LESTestAll.java ดังรายละเอียดโปรแกรมต่อไปนี้

```
/* File: LESTestAll.java (LinearEquationSystemTestAll.java)

*/

import static java.lang.Math.*;
import MathTools.LinearAlgebra.*;

class  LESTestAll
{

public static void main(String[] args) {
// ============To change Augument Matrix here ==========
double[][] problem01 = {{3.0,-2.0,2.0,1.},
        {2.,4.0,-1.0,-2.0},{3.,7.,-1.,3.},{1.0,-3.,2.,-3.}};

// Right hand side constants
        double[] constant01={5.,3.,23.,-12.0};

// ==========================================================
        ColumnMatrix solution = new ColumnMatrix();

// example 1
        SquareMatrix A = new SquareMatrix(problem01);
        ColumnMatrix b = new ColumnMatrix(constant01);
        System.out.println(" Square Matrix of Problem 1.-->");
        A.printMatrix();
        System.out.println(" The constants of Problem1.-->");
        b.printMatrix();
        System.out.println(" ============================");

        try{
        // cramer's rule
        CramerRule cr = new CramerRule(A,b);

                solution = cr.solve();
 System.out.println("\n\nThe Solution Using cramer's rule ...");

                solution.printMatrix();
                }catch ( MatrixException msg){
                System.out.println(msg);
                }
        //end of cramer's rule

        //-----------------------------------------------
        try{
        // GaussElimination method

```

```
        GaussElimination ge = new GaussElimination( A,b);

              solution = ge.solve();
  System.out.println("\n\n Using Gauss Elimination ..");
              solution.printMatrix();
              }catch ( MatrixException msg){
              System.out.println(msg);
              }
        //end of Gauss elimination method
        //--------------------------------------------------
        // LU (lower and upper triangular decomposition.
        try{
        LUDecomposition lu= new LUDecomposition( A,b);

              solution = lu.solve();
        System.out.println("\n\nUsing LU decomposition ...");
              solution.printMatrix();
              }catch ( MatrixException msg){
                    System.out.println(msg);
              }
        //end of LU decomposition method
        }
  }
```

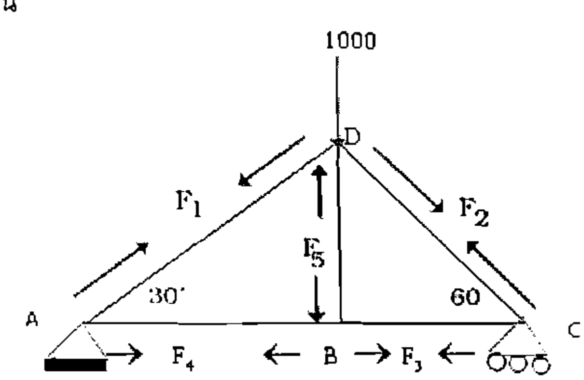
ผลจากการให้โปรแกรม LESTestAll.java ทำงานจะได้ผลลัพธ์ดังนี้

```
Square Matrix of Problem 1.-->
3.0 -2.0 2.0 1.0
2.0 4.0 -1.0 -2.0
3.0 7.0 -1.0 3.0
1.0 -3.0 2.0 -3.0


 The constant Column Matrix of Problem 1.-->
5.0
3.0
23.0
-12.0
 ===========================
```

```
The Solution of Problem by Using Cramer's rule ...
1.9999999999999993
1.0000000000000007
-0.999999999999997
2.9999999999999996


The Solution of Problem by Using Gauss Elimination ..
2.0
1.0000000000000004
-0.9999999999999996
2.9999999999999996


The Solution of LU decomposition ...
2.0
1.0000000000000004
-0.9999999999999996
2.9999999999999996
```

ผลที่ได้ของแต่ละวิธีเป็นค่าประมาณที่ใกล้เคียงค่าแท้จริงเท่านั้น

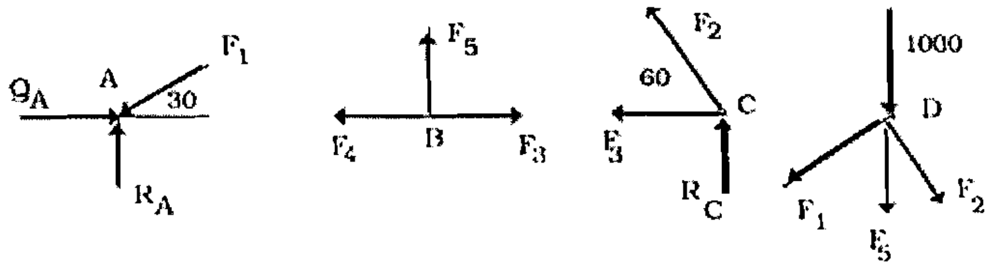
__ตัวอย่าง 4.2.2__ โครงข่ายอยู่ในสภาพสมดุล จงหาแรงที่เกิดขึ้นในโครงข่าย จุด c เป็นล้อเลื่อน ไม่คิดแรงเสียดทานที่พื้น



รูป 4.4 แสดงแรงที่กระทำบนโครงข่าย

__วิธีทำ__ ให้ F, F_2 , F_3 , F_4 , F_5 เป็นแรงที่เกิดขึ้นในโครงข่าย Q_A , R_A และ R_3 เป็นแรงปฏิกิริยาภายนอก

เขียน Free body diagram ที่ node ต่างของโครงข่ายได้ดังรูป



รูป 4.5 แสดง Free body diagram ที่จุดต่าง ๆ

ที่ node A $\quad \Sigma F_x = 0 \quad : \quad R_A + F_1 \sin 30 = 0$

$\quad \Sigma F_y = 0 \quad : \quad Q_A + F_4 = 0$

ที่ node B $\quad \Sigma F_x = 0 \quad : \quad F_5 = 0$

$\quad \Sigma F_y = 0 \quad : \quad F_3 - F4 = 0$

ที่ node C $\quad \Sigma F_x = 0 \quad : \quad R_C + F_2 \sin 60 = 0$

$\quad \Sigma F_y = 0 \quad : \quad F_3 + F_2 \cos 60 = 0$

ที่ node D $\quad \Sigma F_x = 0 \quad : \quad -F_5 + F_1 \cos 60 - F_2 \cos 30 = 1000$

$\quad \Sigma F_y = 0 \quad : \quad -F_1 \sin 60 + F_2 \sin 30 = 0$

มีสมการทั้งหมด 8 สมการ (8 ตัวแปร) เขียนระบบสมการเชิงเส้นให้อยู่ในรูปเมตริกซ์จะได้

$$\begin{bmatrix} 1 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0.866 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 1 & 0 & 0 \\ 0 & 0 & 0 & -0.5 & -0.866 & 0 & 0 & -1 \\ 0 & 0 & 0 & -0.866 & 0.5 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} R_A \\ Q_A \\ R_c \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1000 \\ 0 \end{bmatrix}$$

แก้ไขโปรแกรม LESTestAll.java เฉพาะตรงส่วนที่เป็น augment matrix และค่าคงที่ให้เป็นดังต่อไปนี้

```
// ===================To change Augment Matrix here ==========
double[][] problem01 = {{1.,0,0,0.5,0,0,0,0}, {0,1.,0,0,0,0,1.,0},

            {0,0,0,0,0,0,0,1.},{0,0,0,0,0,1.,-1.,0},

            {0,0,1.0,0,0.866,0,0,0}, {0,0,0,0,0.5,1.0,0,0},

            {0,0,0,-0.5,-0.866,0,0,-1.0}, {0,0,0,-0.866,0.5,0,0,0}};


            // Right hand side constants
      double[] constant01={0,0,0,0,0,0,1000,0};

 // ======================================================
```

ผลลัพธ์การทำงานของโปรแกรมจะได้ดังนี้

```
The Solution of Problem by Using Cramer's rule ...
250.01100048402128
-433.0190528383249
749.9889995159788
-500.0220009680426
-866.0381056766498
433.0190528383249
433.0190528383249
0.0
        Executed time = 3161016 ns

The Solution of Problem by Using Gauss Elimination ..
250.0110004840213
-433.0190528383249
749.9889995159788
-500.0220009680426
-866.0381056766498
433.0190528383249
433.0190528383249
0.0
        Executed time = 1345702 ns

The Solution of LU decomposition ...
250.0110004840213
-433.0190528383249
749.9889995159788
-500.0220009680426
-866.0381056766498
433.0190528383249
433.0190528383249
0.0
        Executed time = 1747987 ns
```

ทั้งสามวิธีจะได้ผลเฉลยของระบบสมการซึ่งเป็นค่าประมาณได้ใกล้เคียงกัน ค่าที่ออกมาติดลบ ได้แก่ Q_A , F_1 และ F_2 แสดงว่าทิศทางของแรงตามรูปภาพจะต้องกลับทิศเสียใหม่ เวลาที่ใช้ในการคำนวณ พบว่า วิธีลดทอนของเกาส์ใช้เวลาน้อยที่สุด กฎของคราเมอร์ใช้เวลามากที่สุด

<u>**ตัวอย่าง 4.2.3**</u> ทดสอบกับปัญหาที่มีจำนวนสมการเชิงเส้น 50 สมการ ได้ผลลัพธ์ออกมาดังตารางหน้า 61 ส่วนโจทย์ของระบบสมการเชิงเส้น 50 สมการได้แสดงไว้ในรูปเมตริกซ์แต่งเติมแล้วในหน้า 62 -63

<u>**วิธีทำ**</u> เวลาที่ใช้ในการประมวลผล พบว่า วิธีของคราเมอร์ใช้เวลา 327,279,584 ns การลดทอนของเกาส์ใช้เวลา 8,320,331 ns วิธี LUDecomposition ใช้เวลา 16,039,469 ns.

ตัวแปร	ค่าแท้จริง	วิธีCramer	การลดทอนของเกาส์	LU Decomposition
x1	0	0.000000000000000	-0.000000000000014	0.000000000000559
x2	1	0.999999999999979	1.000000000000020	0.999999999999071
x3	2	1.999999999999970	1.999999999999980	1.999999999999680
x4	3	3.000000000000000	2.999999999999970	3.000000000000650
x5	4	3.999999999999960	3.999999999999990	3.999999999999740
x6	0	0.000000000000000	0.000000000000017	-0.000000000000316
x7	-1	-0.999999999999980	-0.999999999999961	-1.000000000000400
x8	-2	-1.999999999999990	-2.000000000000020	-1.999999999998910
x9	-3	-2.999999999999940	-2.999999999999970	-3.000000000000990
x10	-4	-4.000000000000030	-3.999999999999990	-3.999999999999720
x11	0	0.000000000000000	0.000000000000023	-0.000000000000475
x12	0.5	0.500000000000045	0.500000000000097	0.499999999999506
x13	0.4	0.399999999999989	0.399999999999999	0.400000000000573
x14	0.3	0.299999999999943	0.299999999999982	0.300000000000145
x15	0.2	0.200000000000020	0.200000000000002	0.200000000000856
x16	0.1	0.100000000000016	0.100000000000024	0.099999999999852
x17	0	0.000000000000000	-0.000000000000028	-0.000000000000070
x18	-0.1	-0.099999999999978	-0.099999999999995	-0.099999999999942
x19	-0.2	-0.200000000000000	-0.199999999999995	-0.200000000000129
x20	-0.3	-0.299999999999996	-0.300000000000034	-0.299999999999687
x21	0	0.000000000000000	0.000000000000010	-0.000000000000050
x22	1	1.000000000000000	1.000000000000030	0.999999999999524
x23	2	1.999999999999880	1.999999999999920	2.000000000000900
x24	3	2.999999999999980	3.000000000000020	2.999999999999880
x25	4	3.999999999999940	3.999999999999950	4.000000000000940
x26	5	5.000000000000040	5.000000000000050	4.999999999999520
x27	4	4.000000000000040	4.000000000000020	3.999999999999990
x28	3	2.999999999999960	2.999999999999970	3.000000000000510
x29	2	2.000000000000010	2.000000000000030	1.999999999999780
x30	1	1.000000000000020	1.000000000000030	0.999999999999689
x31	-5	-5.000000000000010	-5.000000000000010	-5.000000000000100
x32	-4	-3.999999999999960	-3.999999999999920	-4.000000000000820
x33	-3	-3.000000000000000	-3.000000000000010	-2.999999999999700
x34	-2	-1.999999999999970	-1.999999999999970	-1.999999999999930
x35	-1	-1.000000000000020	-1.000000000000020	-0.999999999999779
x36	0	0.000000000000000	-0.000000000000014	0.000000000000207
x37	1	1.000000000000070	1.000000000000060	0.999999999999138
x38	2	2.000000000000000	2.000000000000010	1.999999999999920
x39	3	3.000000000000020	3.000000000000030	2.999999999999460
x40	4	3.999999999999990	4.000000000000000	4.000000000000090
x41	1	0.999999999999970	0.999999999999956	1.000000000000440
x42	2	1.999999999999970	1.999999999999980	2.000000000000090
x43	3	3.000000000000020	3.000000000000030	2.999999999999840
x44	5	5.000000000000040	5.000000000000050	4.999999999999650
x45	7	6.999999999999980	6.999999999999970	7.000000000000310
x46	9	9.000000000000030	9.000000000000030	8.999999999999430
x47	11	11.000000000000000	11.000000000000000	10.999999999999400
x48	12	11.999999999999900	11.999999999999900	12.000000000000500
x49	13	13.000000000000000	13.000000000000000	12.999999999999700
x50	15	15.000000000000000	15.000000000000000	14.999999999999200

-6	-9	-3	-7	-8	9	-5	5	-4	-5	-6	-7	-3	9	1	-6	-9	-9	7	7	9	-6	-5	-8	-9	10	0	-9	5	3
2	4	-8	0	-1	7	-3	3	9	8	7	-7	-1	-1	1	-9	-9	4	6	-9	9	-6	0	8	8	-9	0	-3	-4	6
6	0	-1	-9	7	-1	-5	-10	0	-10	1	0	1	-6	1	9	-3	1	2	2	8	-5	-8	-6	7	6	2	1	-9	-9
-7	-5	8	-1	4	7	8	-1	7	7	5	5	-6	9	3	-6	-9	-6	-10	3	-2	-4	7	-1	7	-10	0	0	-1	-2
-5	8	2	3	7	0	6	7	4	-2	7	-8	-1	1	-9	-5	-5	5	-6	-1	6	0	0	7	-8	-2	8	3	-8	-10
-5	-3	-3	-5	-9	9	-2	5	7	-2	4	0	-9	2	-2	5	-3	-3	-8	10	10	1	8	8	-1	-5	-4	-1	1	6
3	4	3	-3	1	-1	5	-7	7	2	-2	8	10	0	-1	-9	-2	8	6	1	-5	-2	1	0	9	8	-2	7	-9	7
-2	2	-9	7	5	1	9	4	0	-3	5	-9	9	-5	1	-5	-6	6	4	-3	4	7	9	-6	0	4	2	5	0	-2
-5	-6	-6	-6	-3	9	-3	4	-4	-7	4	-2	1	-4	-7	-4	1	3	-3	-9	7	-6	9	6	-6	2	-2	9	-4	-5
6	6	5	2	9	2	10	8	8	-3	6	-2	6	4	2	2	-3	5	-10	-3	1	-1	-6	6	1	-4	3	0	-1	-5
9	7	-8	-2	-5	-4	-7	-3	-8	-9	7	2	2	1	6	-9	5	1	2	-4	-7	8	-2	9	-2	1	6	-6	-3	-3
-2	6	3	5	6	-6	3	-5	3	9	-10	-5	5	1	6	7	-1	6	3	1	-3	7	10	-7	-9	2	4	1	5	7
4	-9	7	4	0	-9	-7	9	-1	0	9	9	4	8	6	-6	-4	1	1	-5	3	7	-4	-10	2	3	-9	-8	-7	-3
3	-10	9	2	-6	4	-6	-7	1	2	-3	5	-3	1	0	-3	1	-2	-10	-3	5	-8	-2	9	-8	-5	-5	-2	-7	6
3	-1	-1	0	7	1	-7	-7	1	-6	-2	0	-8	4	-3	8	4	0	2	3	3	0	6	7	-7	-4	1	-7	5	-1
-9	-2	3	-8	0	-1	8	-5	-6	7	-4	-5	-2	-9	8	9	9	10	-2	-6	7	-8	8	7	-1	0	-8	-2	3	-4
9	-5	-1	-8	-1	-4	-2	-9	6	9	3	-1	7	4	-2	-6	-3	9	2	3	6	3	8	-9	6	3	-6	0	1	1
6	9	4	-5	8	-5	6	10	3	6	8	-4	7	9	-2	-9	-3	-3	-8	2	-2	-5	-6	-6	3	-5	9	7	10	-5
3	3	7	0	7	-4	1	-5	2	-6	5	-1	-1	4	7	6	0	-4	-8	0	-3	0	-9	6	-1	-7	-2	5	-10	5
-2	1	-6	-3	5	4	3	1	7	4	-10	-3	3	3	-3	-3	-3	1	5	3	-7	6	6	2	-5	9	1	7	-7	2
-9	-5	-4	6	2	-9	-4	2	10	5	-7	-6	-5	7	-6	5	-3	0	4	2	-1	6	3	5	3	-3	-8	-4	3	-1
-3	8	0	4	-9	-6	-7	0	1	-3	6	2	-4	-4	-9	1	2	-3	-4	-3	-6	3	6	-5	7	-4	1	-9	9	5
-3	3	-8	2	9	-4	-9	4	5	-4	2	-7	5	-5	-1	3	8	-4	10	-9	0	-5	10	5	-10	2	-3	5	-4	10
-4	-6	8	6	-10	-1	-9	-9	6	-3	-4	1	-2	-1	4	-1	-6	9	9	7	6	5	-7	4	3	-10	9	-6	-1	4
-5	10	-6	9	6	-4	-7	-9	-5	5	-1	9	-4	10	7	2	-8	2	6	-4	7	-5	7	1	-1	-2	-9	0	-5	4
-10	-7	-2	-4	-3	6	-4	-5	4	2	-6	5	-4	-2	10	1	5	-10	9	3	-1	-3	0	2	-3	-10	-6	-3	-4	-7
-6	-9	3	-3	-8	-10	-9	-1	4	3	1	2	-2	4	-6	5	-1	0	5	3	-8	-8	3	5	9	5	-3	-2	2	5
5	-3	-2	2	-1	-5	3	-2	7	3	5	7	1	1	-9	0	-1	6	7	-4	7	9	9	-7	10	-4	2	4	-4	-7
-3	5	9	6	1	3	6	5	2	-9	-1	-1	-5	5	-3	-4	-8	2	6	-5	-1	-5	-5	-4	-5	-1	0	4	-3	2
-5	2	-2	1	2	6	5	5	-2	-10	-1	-6	0	7	4	6	-5	-9	2	-1	2	0	-6	1	-4	9	3	-2	9	-5
-2	-1	-5	-7	-8	-1	7	3	6	3	-7	-5	6	-5	7	-4	4	9	2	-3	1	-5	9	6	1	5	-4	7	7	-7
6	-1	-4	-7	8	-4	4	2	-10	-2	5	7	6	4	-7	9	3	-9	0	10	-9	5	0	6	-3	2	3	8	-7	3
4	2	5	-9	10	4	3	9	-1	1	3	-7	-6	7	-4	2	7	0	-7	-1	1	10	-2	2	4	-4	-7	2	-3	2
-8	1	8	3	-5	7	6	2	-9	-9	3	-2	-6	-7	9	-3	-6	-5	6	-3	-9	4	5	7	9	-1	6	3	1	-5
9	3	-9	1	7	8	-9	4	1	4	1	6	5	4	-9	-4	5	-9	6	-3	-9	-9	-2	6	-2	-10	-9	-4	2	-4
0	-6	3	-6	-10	-5	-4	3	3	-7	-2	5	-1	2	-6	-6	6	-6	-6	4	-5	9	8	4	6	3	6	0	8	-1
0	10	2	-10	8	6	9	9	-3	8	-6	-2	7	8	-2	3	-8	-3	5	10	2	1	0	-6	-1	1	-5	7	3	9
-1	1	-4	2	6	-2	-6	1	-3	4	9	-9	-9	-1	-7	-1	5	7	10	-10	-1	3	-2	-6	-9	4	6	-4	3	1
9	2	-6	-1	-8	-3	1	6	-7	1	6	9	-9	3	9	-1	9	-9	-9	3	-5	0	2	6	-3	-6	3	-2	10	-1
5	-6	0	-4	2	-1	7	3	7	-5	-3	7	-2	1	-4	-9	-3	-1	-1	-9	-9	3	1	-2	-4	5	9	7	-1	-4
6	-4	-6	6	-4	6	6	-2	-3	4	6	1	-4	2	2	-1	-3	-4	-7	7	3	-5	8	3	-6	-4	-4	7	-4	-6
1	3	10	0	6	-9	0	5	3	1	-5	0	3	-2	-3	-2	-4	-1	-6	-10	7	-6	-4	1	1	9	2	5	-1	-9
-4	-2	7	0	-4	5	-4	7	1	-3	-8	3	4	3	9	1	6	4	-5	-3	5	10	-3	0	7	-3	-5	4	6	5
6	1	3	-1	-1	9	7	2	1	-5	9	-3	-7	3	-6	2	-7	-9	5	6	3	3	0	-2	-1	4	-9	-7	4	2
-5	7	-6	6	-10	-4	7	3	-2	6	4	-1	4	5	5	6	9	-6	7	1	-7	4	6	4	7	2	4	-9	-1	-5
-3	-10	5	10	8	7	1	-8	-9	0	-5	-7	9	-3	-1	-2	-4	3	3	5	9	-3	-5	7	3	-2	5	-2	8	-6
-7	0	6	-3	1	-3	-3	-6	3	-5	0	1	-3	5	-4	2	-6	-6	-6	-7	-9	3	2	2	4	-2	-1	-7	2	-2
-7	7	0	-7	-4	7	-1	-8	8	-3	9	-9	2	-1	2	0	-4	-5	-9	-6	-1	-9	-3	5	-2	-5	0	2	-1	-6
-5	5	0	-6	0	-3	5	-1	-4	3	-7	-10	-2	-2	10	-3	-8	-1	6	5	-1	1	6	6	-7	1	5	-2	-4	1
-1	6	3	4	-8	2	6	7	-5	-1	-5	-5	-4	9	-1	7	-7	-8	5	3	-4	3	-1	7	2	-3	-1	-6	7	3

-2	-5	1	2	9	6	-1	9	9	2	5	-1	-3	3	-2	10	-8	1	-7	9	x1		58.7
0	-6	-3	7	-8	-2	10	-6	-7	-3	6	8	-4	-1	-5	-5	8	-7	-8	0	x2		-216.2
0	-6	5	7	-3	-5	0	-5	4	-1	-9	6	-5	-9	-4	6	-8	2	-6	9	x3		6.6
-7	-10	6	6	-5	-9	-2	3	-9	6	-4	-3	-1	10	-6	-3	7	6	6	2	x4		251.5
-5	-1	0	2	-6	1	-2	10	0	0	8	-6	7	-2	-3	5	4	2	1	5	x5		249.6
-2	2	-3	4	-7	3	-4	9	8	4	10	-3	7	-4	6	-6	-4	0	8	5	x6		79
-2	-2	8	-2	2	7	2	3	0	-6	7	9	2	1	-1	-9	8	6	9	1	x7		256
-6	6	-7	-7	8	-10	-5	-2	9	-2	-6	2	5	3	1	-6	-4	-6	8	-5	x8		-1.5
-7	-5	6	3	-2	-2	2	5	0	1	10	-5	1	9	-10	0	2	-2	7	5	x9		233.4
-9	3	-5	0	8	-8	-2	-9	1	-5	-3	-10	-10	2	9	6	-2	8	2	-5	x10		72.4
-2	-1	0	10	3	-1	-6	7	7	2	-8	-7	-7	-3	-5	5	6	6	-4	-6	x11		78.2
1	5	3	2	-5	-8	8	3	-8	4	8	-4	-2	-2	9	-2	7	1	-5	-2	x12		0.8
-1	4	-3	-5	-10	3	1	-10	-5	-7	2	7	2	-5	4	-7	7	-5	6	0	x13		-111.4
3	5	7	2	6	2	-6	3	2	-10	4	3	-4	-7	9	4	-2	-8	0	2	x14		-255.6
-5	7	-3	-9	-9	8	-5	4	-8	2	-9	-3	-8	2	6	4	7	7	9	-2	x15		356.9
2	-2	-8	3	1	-9	8	6	-7	7	0	-10	-5	-3	0	3	-2	2	10	-1	x16		104.9
8	-1	5	0	4	-6	7	1	3	9	-9	5	-4	3	10	7	-8	6	4	7	x17		228.8
-10	2	-2	6	4	-1	4	1	7	3	-7	7	-6	-5	4	7	6	4	7	1	x18		338.5
-4	3	-4	-8	-4	3	-6	-8	9	-3	-6	4	5	-2	0	-10	-5	-3	7	-6	x19		-137.7
-4	-6	-8	2	0	-5	-5	2	-6	6	-5	-6	2	5	-6	-3	3	-4	-3	0	x20		-5.7
-5	-4	-10	7	-10	6	1	7	-9	-9	-8	0	7	6	3	2	-7	-6	-1	2	x21		-58.2
-6	1	2	3	-9	-7	-2	-3	3	-6	8	-1	6	7	2	7	8	10	-6	-3	x22		250.8
-5	-3	-5	9	-5	3	8	-3	-7	-9	-7	7	1	6	6	2	9	-6	-4	2	x23		148.2
6	2	4	-5	-5	9	-3	-2	-1	-3	8	2	-2	5	-4	-9	9	-9	2	-4	x24		-213.7
-3	0	-4	-2	7	-3	5	2	-5	6	4	2	9	-5	-5	5	9	7	-2	2	x25	=	246.8
0	2	-5	6	-5	9	1	1	9	4	5	-6	-4	2	8	-7	3	-6	2	-9	x26		-312.1
2	0	-4	-4	9	5	9	6	6	10	-3	-2	1	-4	-3	-3	-2	-2	-10	0	x27		-144.2
-7	2	7	1	5	7	6	10	-7	2	-3	-9	0	-3	-5	-7	7	2	-4	3	x28		-40.4
-8	-5	8	4	4	-5	4	-7	-2	-4	-6	0	-9	-5	4	-2	2	-1	9	-5	x29		-21.1
9	5	-4	6	3	-3	-4	-4	-3	4	7	0	-2	-1	-9	-3	7	7	-2	-1	x30		-34.7
-8	3	-2	-2	-6	-7	-3	-3	0	3	4	-6	9	2	-5	-3	7	7	-4	6	x31		206.8
-1	1	-7	-7	2	2	-9	-2	-6	5	-5	-3	-5	-9	1	-4	-7	-5	7	3	x32		66.2
-4	-2	6	3	-3	4	8	4	-10	-5	3	2	4	6	-10	-1	8	-8	-4	3	x33		-126.7
3	-4	0	3	7	9	-1	10	-2	-6	-9	9	2	-1	-7	-4	-2	-9	7	5	x34		126
1	9	-9	8	8	-5	-4	-3	4	-1	2	-3	1	9	-4	-6	3	5	-6	-1	x35		-198.9
10	6	3	5	10	3	-5	5	-10	-10	3	-2	6	3	0	3	4	0	-7	1	x36		11.9
6	-1	3	-6	-6	-5	-5	6	10	-3	5	-2	1	-5	-2	7	-6	5	5	3	x37		57.9
-6	0	-3	0	-8	3	-2	3	-1	-4	2	5	-1	0	3	3	2	6	-7	3	x38		252.4
-4	-5	0	-10	8	-5	-4	10	-10	5	-5	-6	7	3	6	3	-2	-5	4	3	x39		219.9
-6	-5	-1	3	-10	6	-8	-6	-5	4	6	5	4	-2	-2	0	0	-10	1	-3	x40		-116.7
0	-4	-6	3	-5	-5	9	-1	7	-3	3	-4	3	-3	5	-2	-10	-5	6	1	x41		-125.5
8	-2	-1	5	-3	-5	4	-8	-4	-9	10	4	1	1	-6	-10	-8	-6	2	1	x42		-214.5
-6	1	-6	-3	-5	-6	-2	7	3	6	9	4	4	-8	-5	-6	-7	4	5	5	x43		215.9
3	2	5	7	6	-1	-6	-9	10	3	10	-5	1	-5	1	-3	9	-9	-6	-4	x44		-242.8
-2	-9	-1	-2	4	5	-4	-3	8	5	-6	-9	-9	2	-9	7	7	-1	6	1	x45		160.4
5	2	6	-3	-7	-6	-3	4	-4	-7	-4	-3	5	-6	-4	2	8	-1	2	4	x46		214.2
0	-4	3	-4	4	7	8	1	3	-6	9	7	4	-7	-3	2	6	1	-4	10	x47		191.8
-4	4	5	-7	4	8	8	-1	-10	-1	-5	-5	5	-3	-3	5	0	1	6	-5	x48		-40.5
-5	5	-8	7	10	-5	4	-3	-10	-3	-3	7	-5	5	1	-6	2	9	1	9	x49		227.8
9	3	6	2	4	7	0	7	2	7	7	-3	7	-7	-5	-3	-3	9	1	3	x50		10

4.3 **การประมาณค่าโดยวิธีกำลังสองน้อยที่สุด** (Method of Least Square)

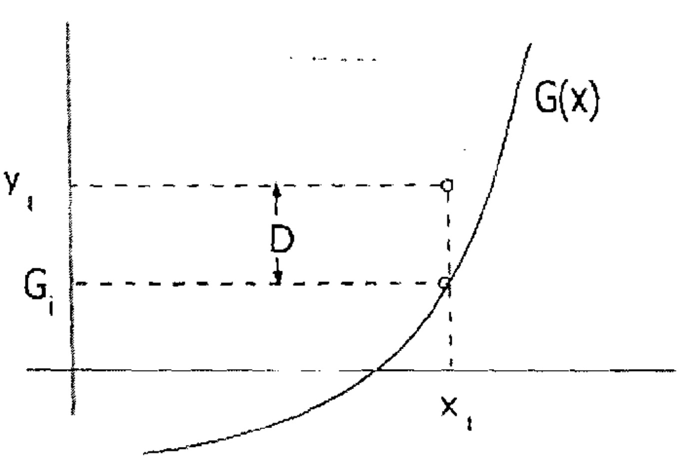
การสร้างแบบจำลองทางคณิตศาสตร์เพื่ออธิบายปรากฏการณ์ทางวิทยาศาสตร์วิธีหนึ่งก็คือ นำข้อมูลจากการทดลองมาหาความสัมพันธ์ระหว่างตัวแปรที่ได้จากการทดลอง แล้วเขียนเป็นฟังก์ชันหรือสมการทางคณิตศาสตร์ เพื่อใช้ในการอธิบายปรากฏการณ์นั้น

การประมาณค่าฟังก์ชันโดยวิธีกำลังสองน้อยที่สุดจะได้ฟังก์ชันที่เป็นตัวแทนที่ดีที่สุดของข้อมูล เพราะได้จากการเกลี่ยค่าความคลาดเคลื่อนของข้อมูลให้เหลือน้อยที่สุด ดังนั้นเมื่อเขียนกราฟของฟังก์ชัน เส้นกราฟจะผ่านไปในบริเวณจุดต่าง ๆ ของข้อมูล โดยจะตัดผ่านจุดของข้อมูลบางจุด

หลักการของวิธีกำลังสองน้อยที่สุด มีดังนี้ ถ้ามีข้อมูล x, y ทั้งสิ้น n ชุด ให้ฟังก์ชันที่ประมาณค่าข้อมูลชุดนี้เป็น G(x) โดยที่ G(x) อยู่ในรูป

$$G(x) = a_1g_1(x) + a_2g_2(x) + \dots a_mg_m(x) \qquad \dots\dots\dots\dots (4.5)$$

โดยที่ $m \leq n$ $g_1(x), \dots g_m(n)$ เป็นฟังก์ชันซึ่งขึ้นอยู่กับค่า x อาจจะอยู่ในรูปพหุนาม (polynomial) รูปล็อกการิทึม หรือเอ็กซโพเนนเชียล สมการ (4.5) จะสมบูรณ์ได้ก็ต่อเมื่อทราบค่า $a, a_2, \dots a_m$ โดยหาค่าสัมประสิทธิ์เหล่านี้ได้จากการทำให้ค่าเบี่ยงเบนของข้อมูลกับค่าประมาณที่ได้จากฟังก์ชัน G(x) มีค่าน้อยที่สุด



รูป 4.6 แสดงการหาค่าเบี่ยงเบนของวิธีกำลังสองน้อยที่สุด

จากรูป ค่าแตกต่างของข้อมูลชุดที่ i คือ

$y_i - G(x_i)$ เมื่อหาค่าแตกต่างของข้อมูลทุกชุดแล้ว นำค่าแตกต่างเหล่านี้มารวมกัน แล้วยกกำลังสอง เพื่อขจัดเครื่องหมายลบ จะได้

$$D = \sum_{i=1}^{n}\left[y_i - G(x_i)\right]^2 \qquad \dots\dots\dots\dots (4.6)$$

ค่าสัมประสิทธิ์ $a_1, a_2, \ldots a_m$ จะเป็นตัวแปร เพราะเมื่อค่าเหล่านี้มีค่าต่าง ๆ กัน ฟังก์ชัน G(x) จะเป็นฟังก์ชันที่แตกต่างกันออกไป แต่ต้องการหาค่า $a_1, a_2, \ldots a_m$ ที่มีเงื่อนไขทำให้เกิดค่า D มีค่าน้อยที่สุด คืออนุพันธ์ของ D เมื่อเทียบกับ $a_1, a_2, \ldots a_m$ จะมีค่าเป็นศูนย์

$$\left.\begin{aligned} \frac{\partial D}{\partial a_1} &= 0 \\ \frac{\partial D}{\partial a_2} &= 0 \\ &\vdots \\ \frac{\partial D}{\partial a_m} &= 0 \end{aligned}\right\} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (4.7)$$

จะได้สมการออกมา m ชุด สามารถหาค่า $a_1, \ldots a_m$ ได้ โดยใช้ระบบสมการเชิงเส้น

การหาฟังก์ชันเพื่อประมาณค่าชุดข้อมูลโดยวิธีกำลังสองน้อยที่สุดนี้เรียกชื่ออีกอย่างหนึ่งว่า การถดถอย (regression)

4.3.1 **ข้อมูลมีความสัมพันธ์แบบเชิงเส้น** (Linear Regression)

ถ้าข้อมูลทั้ง n ชุด มีความสัมพันธ์กันในลักษณะเชิงเส้นตรง สมการเส้นตรงที่หาได้จากวิธีนี้เรียกว่า การถดถอยแบบเชิงเส้น ให้ฟังก์ชันที่จะใช้เป็นตัวแทนข้อมูลชุดนี้มีรูปสมการเป็น

$$G(y) = a_0 + a_1 x \qquad \ldots\ldots\ldots\ldots\ldots\ldots (4.8)$$

เมื่อเปรียบเทียบกับสมการ (4.5) นั่นคือ $g_1(x) = 1$, $g_2(x) = x$ เทอมต่อ ๆ ไปเป็นศูนย์ทั้งหมด

ผลรวมกำลังสองของค่าเบี่ยงเบนจากสมการ (2) คือ

$$D = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2$$

a_0, a_1 คือค่าคงที่ที่ต้องการหา โดยทำให้ D มีค่าน้อยที่สุด นั่นคือ

$$\frac{\partial D}{\partial a_0} = -2\sum_{i=1}^{n} (y_i - a_0 - a_1 x_i) = 0$$

$$\frac{\partial D}{\partial a_1} = -2\sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)(x_i) = 0$$

จัดรูปใหม่จะได้

$$na_0 + a_1\sum x_i = \sum y_i$$

$$a_0\sum x_i + a_1\sum x_i^2 = \sum x_i y_i$$

แก้สมการหาค่า a_0, a_1 จะได้

$$a_0 = \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

$$a_1 = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

.... (4.9)

a_0 คือจุดตัดแกน y และ a_1 คือความชันของเส้นตรงนั่นเอง

4.3.2 **ข้อมูลมีความสัมพันธ์แบบพหุนาม** (Polynomial Regression)

ข้อมูล n ชุด มีความสัมพันธ์กันแบบพหุนามอันดับ m เขียนเป็นสมการได้ดังนี้

$$G(x) = a_0 + a_1x + a_2x^2 + ... a_mx^m$$

ผลรวมกำลังสองของความเบี่ยงเบนของข้อมูลกับ G(x) คือ

$$D = \sum\left(y_i - G(x)\right)^2$$

หาค่า a_0, a_1, a_2, ... a_m โดยใช้เงื่อนไข D ต้องมีค่าน้อยที่สุด

$$\frac{\partial D}{\partial a_i} = 0 \quad i = 0, 1, ... m$$

จะได้เป็นระบบสมการเชิงเส้น m+1 สมการ ได้คำตอบของสมการ

ให้ $\sum_{i=1}^{n} x_i = \sum x$ ละเว้นพิสัยของการหาผลรวมเพื่อให้ดูง่าย

$$\begin{bmatrix} n & \sum x & \sum x^2 & \cdots & \sum x^m \\ \sum x & \sum x^2 & \sum x^3 & \cdots & \sum x^{m+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \sum x^m & \sum x^{m+1} & \sum x^{m+2} & \cdots & \sum x^{2m} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} \sum y \\ \sum xy \\ \vdots \\ \sum x^m y \end{bmatrix}$$

.... (4.10)

จะเห็นว่าสมการ (4.9) เป็นส่วนหนึ่งของสมการ (4.10) เมื่อ m = 1 นั่นเอง

ผู้วิจัยได้ออกแบบให้คลาส PolynomialRegression เป็นคลาสแม่ โดยกำหนดให้พหุนามที่จะใช้ในการประมาณฟังก์ชันข้อมูลนี้มีกำลังสูงสุดของตัวแปรอิสระ มีค่าไม่เกิน 10 (x มีกำลังไม่เกิน 10 , x^{10}) ข้อมูลที่ใช้ในการหาค่าฟังก์ชันต้องมีจำนวนข้อมูลตั้งแต่ 2 ชุดขึ้นไป สำหรับสมการเชิงเส้น หรือจำนวนข้อมูลจะต้องมีมากกว่า กำลังของตัวแปรอิสระบวกด้วยหนึ่ง (m+1) โดยให้คลาส RegressionException ดักจับความผิดพลาดเหล่านี้ เมธอดที่สำคัญในคลาส PolynomialRegression คือ computeCoefficients() จะคำนวณสัมประสิทธิ์ของตัวแปร a_0, a_1, a_2, ... a_m โดยใช้สมการที่ 4.10 รายละเอียดของคลาส PolynomialRegression มีดังนี้

```
/* File: PolynomialRegression.java */
package MathTools.Regression;

import static java.lang.Math.*;
import  MathTools.Common.*;
import MathTools.LinearAlgebra.*;
/**
 * Finding coefficients of polynomial regression.
 */
public class PolynomialRegression {

/** number of data  */ private int     dataCount=0;
/** maximum number of data */ private int maxPoints;
/** true if those coefficients are calculated */
      private boolean CalculatedChecking;
/** degree of the polynomial , max value = 10 */
      private int degree;
/** data in (x, y) pairs for calculating coefficient */
      private DataItem data[];

 /** coefficient square matrix */ private SquareMatrix coeff;
 /** coefficient of x */ private ColumnMatrix a;
 /** right hand side constants */ private ColumnMatrix c;
  /**
     * Constructor.
     */
    public PolynomialRegression() {}

 /**
 * Constructor.
 * @param degree the degree of polynomial
 * @param data the array of xy data.
 */
     public PolynomialRegression(int degree,DataItem data[]) {
         this.data = data;
             this.dataCount = data.length;
             this.degree= degree;
     }
 /**
       * Constructor.
         * @param degree the degree of polynomial
       * @param maxPoints the maximum number of data.
 */
```

```

      public PolynomialRegression(int degree,int maxPoints) {
        this.maxPoints = maxPoints;
            this.degree= degree;
            this.data = new DataItem[maxPoints];
     }
 /**
      * Constructor.
        * @param degree the degree of polynomial
      * @param x  independent variable x
        * @param y  dependent variable y
 */

 public PolynomialRegression(int degree,double[] x,double[] y) {
             this.degree= degree;
             this.dataCount = min(x.length,y.length);
             this.data = new DataItem[dataCount];
             for(int i = 0; i < dataCount; i++)
                   data[i] = new DataItem(x[i],y[i]);
     }
     /**
      * Return the current number of data .
      * @return the number of data.
      */
 public int getNumberOfData() { return dataCount; }
     /**
      * Return the data items
      * @return DataItem
      */
 public DataItem[] getDataItem() { return data; }

     /**
      * Return the coefficients matrix
      * @return coefficient square matrix
     */
 public SquareMatrix getCoeffMatrix() { return coeff; }
     /**
      * Return the coefficients of Polynomial of x
      * @return coefficient column matrix
     */
 public ColumnMatrix getCoeffOfPolynomial() {
             if(!CalculatedChecking)
                try{
                   computeCoefficients();
             } catch(RegressionException msg){
                   System.out.println(msg);
             }
             return a;
      }
     /**
      * Add a new data point
      * @param dataItem the new data point
      */
 public void addData(DataItem dataItem) {
         data[dataCount] = dataItem;
          dataCount +=1;
          CalculatedChecking = false;
     }
     /**
      * Return the value of the polynomial regression  at x.
      * @param x the value of x
```

```
     * @return the value of polynomail at x
     */
public double getValueAt(double x){
            double y=0;
            double xPower=1;
        if(!CalculatedChecking) {
              try{
                  computeCoefficients();
            } catch(RegressionException msg){
                  System.out.println(msg);
            }
            }
            for(int i =0; i <= degree; i++){
                  y += a.getValueAt(i)*xPower;
                  xPower = xPower*x;
            }
            return y;
    }
    /**
     * Reset all instance variable in object.
     */
public void resetAll()
    {
        dataCount = 0;
        CalculatedChecking = false;
    }
      /** Calculate the sum of x for each power of x
      *   @return sum
      */
private double sumXPower(int power){
          double sum=0;
          for(int i=0; i< dataCount; i++){
                sum += xPower(data[i].x, power);
          }
          return sum;
   }
/**  find the sum of x for each power of x and multiplied by y
*   @return sum
*/
private double sumXPowerY(int power){
          double sum=0;
          for(int i=0; i< dataCount; i++){
                sum += data[i].y*xPower(data[i].x, power);
          }
          return sum;
}
/** Compute the power of x using my special algorithm.
* @param x  the value of x
* @param n  the integer power of x
* @ return the resulte of the nth power of x
*/
private double xPower(double x, int n){
        if (n < 0) return 1/xPower(x, -n);
          double power = 1;
          // Loop to compute x^n.
          while (n > 0) {
              // Is the rightmost exponent bit a 1?
              if ((n & 1) == 1) power *= x;
              // Square x
              x *= x;
                    //shift the exponent 1 bit to the right.
```

```
                n >>= 1;
            }
            return power;
    }
    /** Compute the coefficients of polynomial.
    * @exception MatrixException
    */

    private void computeCoefficients() throws RegressionException{
         if (CalculatedChecking) return;
               coeff = new SquareMatrix(degree+1);
               c        = new ColumnMatrix(degree+1);
   // compute each coefficient of upper left half (by row)
   //          a00   a01   a02   a03   ...   a0m ( m = degree)
   //          a10   a11         ...   a1(m-1)
   //          ...   ...         a2(m-2)
   //          am0
   //
   if (degree > 10 || degree < 1) throw new
   RegressionException(RegressionException.DEGREE_INVALID);
   if (dataCount < degree+1) throw new
         RegressionException(RegressionException.LESS_DATA);

               for(int i=0; i <=degree; i++){
                     double sumX = sumXPower(i);
                     int j =0;
                     // set the value which along the diagonal
                     for(int k = i; k >=0; k--){
                     try{
                           coeff.setValueAt(k,j,sumX);
                           j = j+1;
                     }catch (MatrixException er) {
                     System.out.println(er.getMessage());
                     }
                     }
                     c.setValueAt(i,sumXPowerY(i));
               }// for i
    // compute each coefficient of lower right half (by column)
               for(int j=1; j <= degree; j++){
                     double sumX = sumXPower(j+degree);
                     int i = degree;
                     for(int k = j; k <=degree; k++){
                           try{
                           coeff.setValueAt(i,k,sumX);
                           i = i-1;
                           } catch(MatrixException er)
                     { System.out.println(er.getMessage());}
                     }
               }
               try{
         GaussElimination ge = new GaussElimination( coeff,c);
         a = ge.solve();
               }catch ( MatrixException msg){
                     System.out.println(msg);
               }
    }
 }
```

ส่วนคลาส LinearRegression จะสืบทอดมาจากคลาส PolynomialRegression และใช้ตัวแปรคลาสและเมธอดต่าง ๆ ที่มีอยู่ทั้งหมด ไม่ต้องเขียนชุดคำสั่งขึ้นมาใหม่ สิ่งที่ต่างไปจากเดิมคือกำหนดค่า degree ให้มีค่าเท่ากับ 1 เท่านั้น

รายละเอียดของคลาส LinearRegression มีดังนี้

```
/* File: LinearRegression.java */
package MathTools.Regression;

import static java.lang.Math.*;
import  MathTools.Common.*;
import MathTools.LinearAlgebra.*;
/**
 * Finding coefficients of Linear  regression.
 */
public class LinearRegression extends PolynomialRegression {
   /**
      * Constructor.
      */
     public LinearRegression() {}
/**
      * Constructor.
        *  @param data the array of xy data.
      */
      public LinearRegression (DataItem data[ ]){
             super(1,data);
      }
/**
      * Constructor.
      * @param maxPoints the maximum number of data.
      */
       public LinearRegression(int maxPoints) {
             super(1,maxPoints);
      }
/**
      * Constructor.
      * @param x  independent variable x
        * @param y  dependent variable y
      */
       public LinearRegression(double[ ] x, double[ ] y) {
             super(1,x,y);
      }
}
```

4.3.3 การตรวจสอบความเหมาะสมของฟังก์ชัน

ปัญหาที่สำคัญประการหนึ่งคือจะรู้ได้อย่างไรว่าข้อมูลที่ได้มานั้นสอดคล้องกับสมการเส้นตรงหรือสมการพหุนาม หรือสมการเอ็กซโพเนนเชียล วิธีตรวจสอบที่ง่ายคือการพล็อตกราฟสมการนั้นเปรียบเทียบกับกราฟที่ได้จากข้อมูล แต่วิธีนี้ก็จะทำได้ลำบากถ้ามีตัวแปรในสมการมากกว่า 2 ตัวขึ้นไป ด้วยเหตุนี้ จึงมีการคำนวณทางสถิติ เช่น สัมประสิทธิ์ของการกำหนด (Coefficient of determination) จะใช้วัดความสัมพันธ์กันหรือสหสัมพันธ์ระหว่างตัวแปรอิสระทั้งหลายกับตัวแปรตาม สามารถบอกได้ว่าสอดคล้องกับสมการที่ได้หรือไม่

หลักการเบื้องต้นของสัมประสิทธิ์การกำหนดสามารถอธิบายได้ดังนี้ ข้อมูลที่ได้จากการทดลอง (x_i, y_i) และค่าที่ได้จากการคำนวณของ $G(x_i)$ ความแตกต่างที่ได้คือ $y_i - G(x_i)$ เมื่อหาความแตกต่างของทุก ๆ จุด ยกกำลังสองทุกค่าเพื่อขจัดค่าที่เป็นลบ ผลรวมกำลังสองของความแตกต่างของ y_i (จากข้อมูล) กับ $G(x_i)$ (ที่ได้จากการคำนวณ) เรียกว่า ผลรวมกำลังสองของเศษเหลือ (The sum of the square of the residuals)

$$SSE = \sum_{i=1}^{n}\left(y_i - G(x_i)\right)^2 \qquad \text{............ (4.11)}$$

ถ้านำค่า y_i จากข้อมูลที่ได้ลบออกจากค่าเฉลี่ยของ y $(\bar{y})$ ผลรวมกำลังสองของความแตกต่าง y_i และ $\bar{y}$ ทุกค่าเรียกว่า ผลรวมกำลังสองของการถดถอย (The sum of the square of the regression)

$$SSR = \sum_{i=1}^{n}(y_i - \bar{y})^2 \qquad \text{............ (4.12)}$$

สัมประสิทธิ์การกำหนดจะบอกว่าข้อมูลกับฟังก์ชันมีความเหมาะสมสอดคล้องกันดีหรือไม่ หาได้จาก

$$R^2 = 1 - \frac{SSE}{SSR} \qquad \text{............. (4.13)}$$

ในกรณีที่เป็นระบบสมการเชิงเส้น สมการ R^2 สามารถเขียนได้เป็น

$$R^2 = \frac{n\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} y_i)^2}{(\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2)^2 (n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2)} \qquad \text{............ (4.14)}$$

ค่า R^2 จะมีค่าอยู่ระหว่าง 0 ถึง 1 ถ้าค่า y_i และ $y(x_i)$ มีความแตกต่างกันน้อยหรือไม่มีความแตกต่าง จะได้ค่า R^2 ใกล้เคียง 1 เส้นกราฟที่ได้จะสอดคล้องและเหมาะสมกับข้อมูล ควรจะได้ค่า R^2 มากกว่า 0.9

ค่าทางสถิติอีกค่าหนึ่งคือ ค่าความคลาดเคลื่อนมาตรฐานของการประมาณค่า y (Standard error of the estimate) จะเป็นการหาค่าความคลาดเคลื่อนของค่า y ที่ได้จากการคำนวณ และค่า y ของข้อมูล ซึ่งหาได้จากสูตรต่อไปนี้

$$S_{yx} = \sqrt{\frac{\sum\left(y_i - G(x_i)\right)^2}{(n-m)}} \qquad \text{................ (4.15)}$$

เมื่อ n คือจำนวนข้อมูล และ m คือจำนวนสัมประสิทธิ์ในสมการถดถอย เรียก (n - m) ว่าเป็นองศาของความเป็นอิสระ (number of degree of freedom) ถ้าฟังก์ชันที่ใช้ในการประมาณเป็นสมการเส้นตรง องศาแห่งความเป็นอิสระ คือ n - 2

ค่า S_{yx} ยิ่งน้อย และ R^2 ใกล้เคียง 1 มากเท่าใด แสดงว่าฟังก์ชันที่ได้สอดคล้องและเหมาะสมกับข้อมูลที่มีอยู่มากขึ้นเพียงนั้น

ในคลาส PolynomialRegression มีเมธอด `testDeterminationAndError()` ใช้สำหรับหาค่าสัมประสิทธิ์ของการกำหนดและความคลาดเคลื่อนมาตรฐานของการประมาณค่า y โดยอาศัยสมการที่ 4.14 และ 4.15 โปรแกรมต้นฉบับของเมธอดดังกล่าวมีดังนี้

```
/** compute the coefficient of the determination and standard error
 *  of estimate  for y (from data) and  predicted polynomial
    function
 */
private void  testDeterminationAndError() {
    double SSofResidual=0;
    double SSofRegress = 0;
    int DegreeofFreedom;
    double y_estimate,temp;
        if (dataCount <= degree+1) return;

    DegreeofFreedom = dataCount - (degree+1);
    SSofResidual = 0;
    for (int i = 0; i< dataCount; i++) {
         y_estimate = a.getValueAt(0);
       for (int j = 1; j <= degree; j++) {
              temp = 1;
              for (int k=1; k<= j; k++) {
            temp = temp*data[i].x;
         }
      y_estimate += a.getValueAt(j)*temp;
      }
      SSofResidual = SSofResidual + (data[i].y -
              y_estimate)*(data[i].y - y_estimate);
     }
      std_err = sqrt(SSofResidual/DegreeofFreedom);
```

```
     /* Find average value of y */
   double  sumy = 0;
   double ybar = 0;
      for ( int i = 0; i < dataCount; i++) {
    sumy = sumy + data[i].y;
     }
      ybar = sumy/dataCount;

      SSofRegress = 0;
      for (int i = 0; i< dataCount; i++ ) {
    SSofRegress = SSofRegress + (data[i].y - ybar)*
        (data[i].y -ybar);
      }
     R2 = 1 - SSofResidual/SSofRegress;
  }
```

4.3.4 การทดสอบการประมาณค่าโดยวิธีกำลังสองน้อยที่สุด

การทดสอบคลาส LinearRegression และคลาส PolynomialRegression ทำได้ง่าย โดยการกำหนดฟังก์ชันขึ้นมาฟังก์ชันหนึ่ง สร้างข้อมูลจากฟังก์ชันนี้ นำข้อมูลไปทดสอบให้แต่ละคลาสทดลองคำนวณดูว่าสามารถให้ค่าฟังก์ชันตรงกับที่กำหนดไว้หรือไม่

<u>**ตัวอย่าง 4.3.1**</u> ข้อมูลต่อไปนี้ได้มาจากสมการเส้นตรง $y = 3 + 2x$

x	0	1	2	3	4	5
y	3	5	7	9	11	13

ทดลองนำข้อมูลเหล่านี้ให้คลาส LinearRegression คำนวณหาค่าความชัน และ จุดตัดแกน y จะได้เท่ากับค่าที่ได้จากฟังก์ชันต้นกำเนิดหรือไม่

<u>**วิธีทำ**</u> โปรแกรมที่ใช้ทดสอบการประมาณค่าโดยวิธีกำลังสองน้อยที่สุดสำหรับสมการเส้นตรง มีดังนี้

```
/* File: LinearTest.java */

import static java.lang.Math.*;
import MathTools.Regression.*;
import MathTools.LinearAlgebra.*;
import MathTools.Common.*;

class  LinearTest {

public static void main(String[] args){
      ColumnMatrix a = new ColumnMatrix();
      double[] x = {0,1.,2.,3.,4.,5.};
      double[] y = {3., 5., 7., 9., 11., 13.};
      try{
```

```
        LinearRegression lr = new LinearRegression( x, y);
        System.out.println("Number of data = " +
                    lr.getNumberOfData());
        a = lr.getCoeffOfPolynomial();
        //print polynomial equation
        System.out.print("y = " + a.getValueAt(0) + "  + " +
                    a.getValueAt(1)+ "x ");
        }     catch(Exception msg) {
                    System.out.println(msg);
        }
    }
}
```

ผลการทดสอบจะได้คำตอบดังนี้

```
Number of data = 6
y = 3 + 2 x
```

ตัวอย่าง 4.3.2 ก้อนหินเคลื่อนที่แบบวิถีโค้ง ความสัมพันธ์ระหว่างความสูงซึ่งวัดจากพื้นดิน (y) และระยะทางที่เคลื่อนที่ได้ในแนวราบ (x) มีดังนี้

x	0	1	2	4	5	6	7	8
y	0	0.5121	0.8936	1.2648	1.2545	1.1136	0.8421	0.4400

ข้อมูลชุดนี้ได้จากโจทย์กล่าวถึงการขว้างก้อนหินเป็นมุม 30° กับพื้นโลกด้วยความเร็วต้น 10 เมตรต่อวินาที เมื่อไม่คิดแรงต้านของอากาศ จะได้สมการแสดงตำแหน่งต่าง ๆ ของก้อนหินเป็น

$$y = 0.5774x - 0.0653x^2$$

จงคำนวณหา สัมประสิทธิ์ของ x ซึ่งอยู่ในรูปของพหุนามกำลัง 2

วิธีทำ ใช้ข้อมูล x, y ป้อนให้โปรแกรม PolyRegressTest.java ซึ่งมีรายละเอียดดังนี้

```
/* File: PolyRegressTest.java*/

import static java.lang.Math.*;
import MathTools.Regression.*;
import MathTools.LinearAlgebra.*;
import MathTools.Common.*;

class  PolyRegressTest {

public static void main(String[] args){
int degree = 2;
ColumnMatrix a = new ColumnMatrix();
double[] x = { 0, 1., 2., 4., 5., 6., 7., 8.};
```

```
double[] y = {0, 0.5121, 0.8936, 1.2648, 1.2545, 1.1136,
                    0.8421, 0.44};

PolynomialRegression pr = new PolynomialRegression
                                   (degree,x, y);
      System.out.println("Number of data = " +
                    pr.getNumberOfData());
      a = pr.getCoeffOfPolynomial();
      //print polynomial equation
      System.out.print("y = " + a.getValueAt(0) + "  + " +
                    a.getValueAt(1)+ "x ");
      for(int i = 2; i <= degree; i++)
      System.out.print( " + "+ a.getValueAt(i)+ " x^"+ i);

}
}
```

ผลลัพธ์จากการคำนวณจะได้ดังนี้

```
Number of data = 8
y = 2.9150471210670776E-15  + 0.5773999999999974x  + -0.06529999999999968 x^2
```

จะเห็นว่าใกล้เคียงกับค่าแท้จริง เทอมแรกมีค่าน้อยมาก 10^{-15} สามารถปัดให้เป็นศูนย์ได้

4.4 การหาอนุพันธ์และปริพันธ์เบื้องต้น

การพัฒนาคลาสไลบรารีสำหรับการหาอนุพันธ์เป็นการหาอนุพันธ์ของฟังก์ชันที่มีตัวแปรอิสระเพียงตัวเดียว และเป็นฟังก์ชันที่ให้ค่าเป็นจำนวนจริง ผู้ใช้จะต้องกำหนดฟังก์ชันที่จะหาอนุพันธ์และกำหนดจุดที่ต้องการหาอนุพันธ์นั้น สามารถหาได้ทั้งอนุพันธ์อันดับที่ 1 และอันดับที่ 2

4.4.1 การหาอนุพันธ์อันดับหนึ่งของฟังก์ชันที่กำหนดให้

ให้ f(x) เป็นฟังก์ชันที่ต้องการหาค่าอนุพันธ์และ x คือจุดที่ต้องการหาค่าอนุพันธ์ สามารถประมาณค่าอนุพันธ์อันดับหนึ่งที่จุด x ด้วยสูตรการใช้จุด 3 จุดหาอนุพันธ์ (three point formula) ในเบื้องต้นก่อน จากนั้นปรับแต่งค่าประมาณที่ได้นี้ให้ใกล้เคียงค่าแท้จริงด้วยการประมาณค่าแบบริชาร์ดสัน (Richardson extrapolation)

การประมาณค่าการหาอนุพันธ์อันดับ 1 ที่จุด x โดยใช้สูตรการใช้จุด 3 จุดหาอนุพันธ์มีดังนี้

$$f'(x) = \frac{f(x+h)-f(x-h)}{2h}$$

หลักการประมาณค่าแบบริชาร์ดสัน มีดังนี้

ถ้า f(x) เป็นฟังก์ชันที่มีค่าอนุพันธ์ต่อเนื่องถึงอันดับ 5 กระจาย f(x) เป็นอนุกรม โดยอาศัยทฤษฎีบทของเทเลอร์ จะได้

$$f(x_0+h) = f(x_0) + f'(x_0)h + f^{(2)}(x_0)\frac{h^2}{2} + f^{(3)}(x_0)\frac{h^3}{2} + f^{(4)}(x_0)\frac{h^4}{24} + f^{(5)}(x_0)\frac{h^5}{120} + O(h^6) \quad \text{.................}(4.16)$$

$$f(x_0+h) = f(x_0) - f'(x_0)h + f^{(2)}(x_0)\frac{h^2}{2} - f^{(3)}(x_0)\frac{h^3}{2} + f^{(4)}(x_0)\frac{h^4}{24} - f^{(5)}(x_0)\frac{h^5}{120} + O(h^6) \quad \text{..................}(4.17)$$

นำสมการ (4.16) ลบกับสมการ (4.17) เทอมที่ประกอบด้วย $f^{(2)}$, $f^{(4)}$ จะหายไป จัดรูปสมการใหม่ จะได้เป็น

$$y' = f'(x) = \frac{1}{2h}\Big(f(x_0+h) - f(x_0-h)\Big) - f^{(3)}(x_0)\frac{h^2}{6} - f^{(5)}(x_0)\frac{h^4}{120} \quad \text{.................}(4.18)$$

เมื่อ $O(h^6)$ คือ เทอม ๆ หลังจาก $f^{(5)}$ โดยปกติจะตัดทิ้งไป ทำให้เกิดความคลาดเคลื่อนเนื่องจากการตัดปลาย

เขียนสมการ (4.18) ให้ง่าย เพื่อที่จะนำไปใช้กับการประมาณค่าแบบริชาร์ดสัน

$$y' = A_0(h) + a_2h^2 + a_4h^4 + O(h^6) \quad \text{.................}(4.19)$$

$$A_0(h) = \frac{1}{2h}\Big(f(x_0+h) - f(x_0-h)\Big)$$

a_2 คือ $-f^{(3)}(x_0)/6$ $\quad a_4$ คือ $f^{(5)}(x_0)/120$

เนื่องจากการประมาณค่าต้องกระทำซ้ำกันหลายครั้ง หรือหลายระดับ จึงใส่ตัวเลขห้อยที่ตัวอักษร A ไว้ เพื่อมิให้สับสนก่อนการประมาณค่า $y' = A_0(h)$

เริ่มประมาณค่าครั้งที่ 1 โดยให้เปลี่ยนค่า h เป็น h/2

จากสมการ (4.19) จะได้

$$y' = A_0(\frac{h}{2}) + a_2\frac{h^2}{4} + a_4\frac{h^4}{16} + \ldots \quad \text{................}(4.20)$$

นำ 4 คูณสมการ (4.20) แล้วนำสมการ (4.19) มาลบออก เทอม h^2 จะหมดไป ตัดเทอม h^4 ทิ้ง จะได้เป็นค่าประมาณค่าใหม่ คือ

$$y' = \frac{4A_0(h/2) - A_0(h)}{3} + \frac{3}{4}a_4h^4 + \ldots \qquad \text{............... (4.21)}$$

$$y' = \frac{4A_0(h/2) - A_0(h)}{3} = A_1(h)$$

ค่า y' จะเป็นค่าใหม่ เนื่องจากเป็นการประมาณค่าครั้งที่ 1 และ $y' = A^0(h)$ ค่าเดิม จึงให้ค่า y' ใหม่นี้เป็น A_1 (h)

หาผลต่างระหว่าง $A_1(h)$ กับ $A_0(h)$ ถ้ายังมากกว่าค่าความคลาดเคลื่อนที่ยอมรับได้ ให้ประมาณค่าครั้งที่ 2 ต่อไป

จากสมการ (4.21)

$$y' = A_1(h) - \frac{3}{4}a_4h^4 + O(h^6) \qquad \text{................ (4.22)}$$

แทนค่า h ด้วย $\frac{h}{2}$ แล้วแทนสัมประสิทธิ์ของ h^4 ด้วย b_1

$$y' = A_1(h/2) + b_1h^4 + O(h^6) \qquad \text{................ (4.23)}$$

แทนค่า h ด้วย h/2 ลงในสมการ (4.23)

$$y' = A_1(\frac{h}{4}) + b_1\frac{h^4}{16} + O(h^6) \qquad \text{................. (4.24)}$$

นำสมการ (4.24) × 16 แล้วนำสมการ (4.23) มาลบออก ตัดเทอม $O(h^6)$ ทิ้ง

$$y' = \frac{16\,A_1\left(\frac{h}{2}\right) - A_1\left(\frac{h}{2}\right)}{15} \qquad \text{............... (4.25)}$$

$$= A_2(h)$$

จะได้ $A_2(h)$ เป็นค่าประมาณค่าใหม่ของ y' ถ้าผลต่างของ $A_2(h)$ และ $A_1(h)$ ยังมากกว่าค่าความคลาดเคลื่อนที่ยอมรับได้ ให้ประมาณค่าครั้งที่ 3 ต่อไป ทำเช่นนี้เรื่อย ๆ จนผลต่างของค่าประมาณน้อยกว่าค่าความคลาดเคลื่อนที่ยอมรับได้

ถ้าการประมาณค่าทำไปเรื่อย ๆ จนถึงครั้งที่ n จะได้สูตรการประมาณค่าทั่วไปดังนี้

$$A_n(h) = \frac{4^n A_{n-1}\left(\frac{h}{2}\right) - A_{n-1}(h)}{4^n - 1} \qquad \text{.................. (4.26)}$$

เพื่อให้ดูง่าย จึงใส่ค่าต่าง ๆ เป็นตารางดังนี้

ระดับ \ อันดับที่คำนวณ	$O(h^2)$	$O(h^4)$	$O(h^6)$	$O(h^8)$
0	$A_0(h)$			
1	$A_0(h/2)$	$A_1(h)$		
2	$A_0(h/4)$	$A_1(h/2)$	$A_2(h)$	
3	$A_0(h/8)$	$A_1(h/4)$	$A_2(h/2)$	$A_3(h)$
4	$A_0(h/16)$	...	...	...

ผู้วิจัยได้ออกแบบคลาสDerivative ให้เป็นคลาสแม่และเป็นคลาสนามธรรม (abstract class) มีเมธอด computeFirstApproximate () สำหรับค่าประมาณเบื้องต้นของอนุพันธ์ที่ตำแหน่ง x โดยใช้สูตรการใช้จุด 3 จุดหาอนุพันธ์ เมธอด RichardsonExtrapolation() จะนำค่าที่ได้จากการประมาณครั้งแรกมาปรับแต่งแก้ไขให้ได้ใกล้ค่าแท้จริงหรือเท่ากับค่าแท้จริง ค่าความคลาดเคลื่อนที่ยอมรับได้ที่ตั้งค่าไว้โดยปริยายคือ DEFAULT_TOLERANCE เท่ากับ 10^{-7} สามารถเปลี่ยนแปลงค่านี้ได้ตามความเหมาะสม

การดักจับความผิดพลาดที่อาจเกิดขึ้นได้ในขณะที่โปรแกรมทำงาน จะใช้คลาส DerivativeException ทำหน้าที่นี้ มีการกำหนดการประมาณค่าแบบริชาร์ดสันไว้ไม่เกิน 50 ระดับ เพื่อป้องกันการวนรอบแบบไม่รู้จบ

คลาส FirstDerivative จะสืบทอดมาจากคลาส Derivative อีกทอดหนึ่ง จะมีการ override เมธอด computeFirstApproximate() โดยการหาค่าประมาณครั้งแรกโดยใช้จุด 3 จุดหาอนุพันธ์ในบรรทัดที่ 43-45

รายละเอียดของคลาส Derivative และ FirstDerivative มีดังนี้

```
/* File : Derivative.java */

package MathTools.Derivative;

import static java.lang.Math.*;
import static MathTools.Common.CommonConstants.*;
import MathTools.Common.Function;

/** Abstract class for Derivative */

public abstract class Derivative {
/** Function to integrate */  Function function;
/** default tolerance */double tolerance = DEFAULT_TOLERANCE;
/**  value x at which to differentiate */
                              double xDerive;
      /** first derivative at xDerive point */
                              double yDerive;

      /** the spacing of x value */   double  h;

/** Constructor */
Derivative() { }

/** Constructor
* @param f function to differentiate
* @param x  data point at which to differentiate
*/
Derivative( Function f, double x){
      this.function = f;
      xDerive = x;
      initialize_h();
}
/** Constructor
* @param f function to differentiate
* @param x  data point at which to differentiate
* @param tol tolerance assigned by user
*/
Derivative( Function f, double x, double tol){
      this.function = f;
      xDerive = x;
if(tol <= 0) tol = DEFAULT_TOLERANCE; else tolerance = tol;
      initialize_h();
}

private void initialize_h(){
      if(abs(xDerive) < ZERO_APPROACH)
            h = sqrt(tolerance);
      else h = abs(xDerive*sqrt(tolerance));
}

/** Evaluate the first approximated derivative of the fucntion
*  at the point x by using three point formula
*  @return the value of derivative
*/
abstract double computeFirstApproximate(double deltaX);

/** to improve the current approximation
```

```
* uses Richardson extrapolation
*/

protected void RichardsonExtrapolate() throws
                    DerivativeException{
double[][] result = new double[ARRAY_SIZE][ARRAY_SIZE];
int row = 0, col = 0;
double delta, number4;

      delta = h;
      result[0][0] = computeFirstApproximate(delta);
      /* Richardson extrapolate start here */

      do {
            row +=1;
            if(row > ARRAY_SIZE) throw new
 DerivativeException(DerivativeException.ARRAY_SIZE_EXCEED);
            delta /=2;
            result[row][col] = computeFirstApproximate(delta);
            number4 = 1.0;
            for(int j = 1; j <= row ; j++){
                  number4  *=4;
                  result[row][j] = (number4*result[row][j-1] -
                        result[row-1][j-1])/(number4-1);
            } //for j
      }while ((abs(delta)> ZERO_APPROACH) &&
 (abs(result[row][row]-result[row][row-1]) > tolerance));
      yDerive = result[row][row];

/* Richardson extrapolate result print out for checking */
            System.out.println(" value of h = " + h);
          for(int k = 0; k <=row; k++){
            for (int j = 0; j<=k; j++){
                  System.out.print(result[k][j]+ "\t");
            }
            System.out.println();
          }

}/** getting the Derivative result */
public double getDerivativeResult(){ return yDerive;}

}
```

ต่อไปนี้เป็นรายละเอียดของคลาส FirstDerivative ซึ่งสืบทอดมาจากคลาส Derivative อีกทีหนึ่ง

```
/* File : FirstDerivative.java */

package MathTools.Derivative;

import static java.lang.Math.*;
import static MathTools.Common.CommonConstants.*;
import MathTools.Common.Function;
import MathTools.Derivative.*;

/** Abstract class for Derivative */

public  class FirstDerivative extends Derivative {

/** Constructor */
```

```
15:  public FirstDerivative() { }
16:
17:  /** Constructor
18:  * @param f function to differentiate
19:  * @param x  data point at which to differentiate
20:  */
21:  public FirstDerivative( Function f, double x){
22:        super(f,x);
23:        try{
24:        RichardsonExtrapolate();
25:        } catch (DerivativeException msg){
     System.out.println(msg);}
26:  }
27:
28:  /** Constructor
29:  * @param f function to differentiate
30:  * @param x  data point at which to differentiate
31:  * @param tol tolerance assigned by user
32:  */
33:  public FirstDerivative( Function f, double x, double tol){
34:        super(f,x,tol);
35:        try{
36:        RichardsonExtrapolate();
37:        } catch (DerivativeException msg){
     System.out.println(msg);}
38:  }
39:  /** Evaluate the first approximated derivative of the fucntion
40:  *  at the point x by using three point formula
41:  *  @return the value of derivative
42:  */
43:  protected double computeFirstApproximate(double deltaX){
44:        return(function.Of(xDerive + deltaX) -
               function.Of(xDerive - deltaX))/(2*deltaX);
45:  }
46:  }
```

4.4.2 การหาอนุพันธ์อันดับสองของฟังก์ชันที่กำหนดให้

ให้ f(x) เป็นฟังก์ชันที่ต้องการหาอนุพันธ์อันดับสอง และ x คือจุดที่ต้องการหาค่าอนุพันธ์อันดับสองนั้น สามารถประมาณค่าเบื้องต้นได้โดยใช้สูตรการใช้จุด 3 จุดหาอนุพันธ์อันดับสอง ดังนี้

$$f''(x) = \frac{f(x+h)-2f(x)+f(x-h)}{h^2}$$

แล้วใช้การประมาณค่าแบบริชาร์ดสัน หาค่าที่แท้จริงหรือใกล้เคียงแท้จริงอีกครั้งหนึ่ง

คลาส SecondDerivative ที่ใช้หาอนุพันธ์อับดับสองนี้สืบทอดมาจากคลาส Derivative มีการ override เมธอด ComputerFirstApproximate() โดยใช้สูตรใช้จุด 3 จุดหาอนุพันธ์อันดับสอง ซึ่งสูตรจะแตกต่างไปจากการหาอนุพันธ์อันดับหนึ่ง (บรรทัดที่ 47-49) ส่วนการประมาณค่าแบบริชาร์ดสันนั้น ระเบียบวิธีการคำนวณยังคงเดิมทุกประการ

รายละเอียดของคลาส Sceond Derivative มีดังนี้รายละเอียดของคลาส SecondDerivative มีดังนี้

```java
/* File : SecondDerivative.java */

package MathTools.Derivative;

import static java.lang.Math.*;
import static MathTools.Common.CommonConstants.*;
import MathTools.Common.Function;
import MathTools.Derivative.*;

/** Abstract class for Derivative */

public  class SecondDerivative extends Derivative {

/** Constructor */
public SecondDerivative() { }

/** Constructor
* @param f function to find the second derivative
* @param x  data point at which to differentiate
*/
public SecondDerivative( Function f, double x){
       super(f,x);
       try{
       RichardsonExtrapolate();
       } catch (DerivativeException msg)
             { System.out.println(msg);}

}

/** Constructor
* @param f function to differentiate
* @param x  data point at which to differentiate
* @param tol tolerance assigned by user
*/
public SecondDerivative( Function f, double x, double tol){
       super(f,x,tol);
       try{
       RichardsonExtrapolate();
       } catch (DerivativeException msg)
             { System.out.println(msg);}

}
/** Evaluate the second approximated derivative of the fucntion

*  at the point x by using three point formula
*  @return the value of derivative
*/
protected double computeFirstApproximate(double deltaX){
       return(function.Of(xDerive + deltaX) -
             2*function.Of(xDerive) + function.Of(xDerive -
                   deltaX))/(deltaX*deltaX);
}

public double getSecondDerivativeResult(){ return yDerive;}
}
```

4.4.3 การหาปริพันธ์เบื้องต้น

การหาปริพันธ์ของฟังก์ชัน f(x) ที่มีค่าต่อเนื่องในช่วง a, b โดยที่ $a \leq x \leq b$ เขียนเป็นสัญลักษณ์ $\int_a^b f(x)dx$ มีค่าเท่ากับพื้นที่ใต้เส้นโค้งของ f(x) ในช่วงดังกล่าว

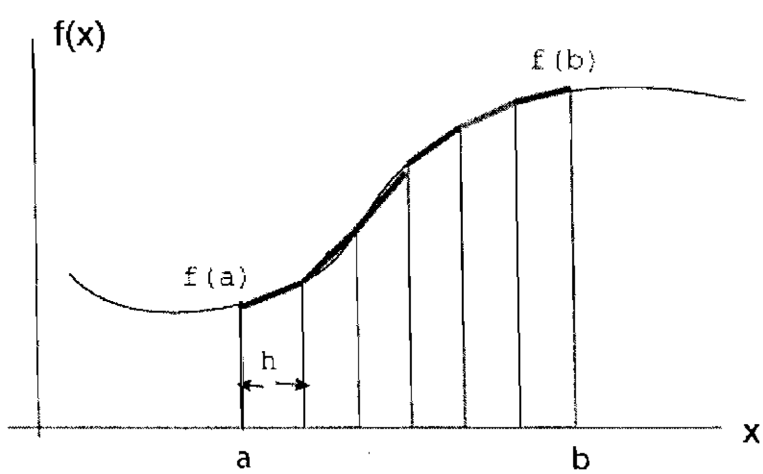
การหาพื้นที่ใต้เส้นโค้ง ทำได้โดยแบ่งพื้นที่ออกเป็นรูปเปลี่ยมเล็ก ๆ จำนวนมาก (เรียกรูปเหลี่ยมเล็ก ๆ นี้ว่า quadrilaterals หรือ numerical quadrature) จากนั้นรวมพื้นที่เล็ก ๆ เหล่านี้เข้าด้วยกัน กลายเป็นพื้นที่ส่วนที่แรเงาทั้งหมด

เขียนเป็นสมการได้ดังนี้

$$\int_a^b f(x)dx = \sum_{i=1}^{N} f(x_i)\,\omega_i$$

เมื่อ N คือจำนวนเส้นที่แบ่งพื้นที่ใต้โค้งจากรูป N = 9 จะแบ่งพื้นที่ออกเป็น 8 ส่วนเล็ก ๆ ω_i คือค่าน้ำหนัก (weight) ของ f(x) ของแต่ละค่า i

ก. **กฎของสี่เหลี่ยมคางหมู** (Trapezoid rule)



รูป 4.6 แสดงการแบ่งพื้นที่ในช่วง x =a ถึง x = b เป็นสี่เหลี่ยมคางหมูเล็ก ๆ

แบ่งพื้นที่ใต้โค้งในช่วง a ถึง b ออกเป็นช่องเล็ก (interval) กว้างช่องละ h แต่ละช่องลากเส้นตรงเชื่อมจุดบนเส้นโค้งจะเกิดเป็นสี่เหลี่ยมคางหมูเป็นจำนวนมาก รวมสี่เหลี่ยมคางหมูเล็ก ๆ เหล่านี้ทั้งหมดจะได้เป็นค่าประมาณของการหาปริพันธ์ของ f(x) ในช่วง a ถึง b ยิ่งแบ่งช่วง

ระยะ h ให้มีค่าน้อยเท่าใดเส้นตรงที่ลากเชื่อมจุดต่าง ๆ บนเส้นโค้งจะทับกับเส้นกราฟของฟังก์ชันที่แท้จริงมากขึ้นเพียงนั้น

ที่จุด x_i ใด ๆ สามารถหาค่าโดยวัดจาก a เป็นหลัก

$$x_i \quad = \quad a + (i-1)h \quad \text{เมื่อ } 1 = 1, 2, \ldots N$$

เมื่อแทน $x_i = b$ โดยที่ $i = N$ จะได้

$$H \quad = \quad \frac{b-a}{N-1}$$

พื้นที่สี่เหลี่ยมคางหมูแต่ละส่วนเล็ก ๆ

$$\int_{x_i}^{x_{i+h}} f(x)dx \quad \cong \quad \frac{1}{2} h\,(f(x_i) + f(x_{i+1}))$$

$$= \quad \frac{1}{2} h\,(f(x_i) + \frac{1}{2} h\, f(x_{i+1})$$

พื้นที่ทั้งหมดใต้เส้นโค้งในช่วง [a, b] คือ

$$\int_a^b f(x)dx \quad \cong \quad \frac{h}{2} f(x_1) + hf(x_2) + hf(x_3) + \ldots + hf(x_{N-1}) + \frac{h}{2} f(x_N) \qquad \ldots\ldots\ldots(4.27)$$

จุด x_i ที่ไม่ใช่เป็นจุดปลายจะถูกนับ 2 ครั้ง จากสมการจะเห็นว่าค่าน้ำหนัก (w_i) ที่ x_i ใด ๆ คือ

$$w_i \quad = \quad \left\{\frac{h}{2}, h, \ldots, h, \frac{h}{2}\right\}, \; i = 1, N$$

ข. กฎของซิมป์สัน $\frac{1}{3}$ (Simpons's $\frac{1}{3}$ rule)

ถ้าเส้นที่ลากเชื่อมจุดต่าง ๆ บนเส้นกราฟเป็นพาราโบลา (ไม่ใช่เส้นตรงดังกฎสี่เหลี่ยมคางหมู) ซึ่งมีสมการเป็น $f(x) = A + Bx + Cx^2$ ต้องใช้ค่า x ถึง 3 จุดเพื่อที่จะหาค่า A, B, C ดังนั้น จึงลากคลุมส่วนแบ่ง 2 ช่อง จำนวนช่องหรือ interval จึงต้องเป็นเลขคู่ หรือ N (จำนวนจุดที่ใช้หาปริพันธ์) ต้องเป็นเลขคี่

เพื่อที่จะได้หาค่า A, B และ C ได้ง่ายขึ้นพิจารณาช่อง 2 ช่องที่อยู่ติดกัน จุดที่หาปริพันธ์ทั้ง 3 จุด คือ x_{i-} , x_i และ x_{i+1} แต่ละจุดมีระยะห่างกับ h แทนค่าจุด $x_{i=1}$, x_i และ x_{i+1} ลงใน $f(x) = A + Bx + Cx^2$

$$\int(x_{i-1}) \quad = \quad A + B(0) + C(0) \qquad \text{..........(1)}$$

$$\int(x_i) \quad = \quad A + Bh + Ch^2 \qquad \text{..........(2)}$$

$$\int(x_{i+1}) \quad = \quad A + 2Bh + 4Ch^2 \qquad \text{..........(3)}$$

จากสมการ (1) (2) (3) หาค่า A, B, C จะได้

$$H^2C \quad = \quad \frac{f(x_{i+1}) + f(x_{i-1}) - f(x_i)}{2}$$

$$hB \quad = \quad 2f(x_i) - \left(\frac{f(x_{i+1} + 3f(x_{i-1})}{2}\right)$$

$$A \quad = \quad f(x_{i-1})$$

พื้นที่ของช่องเล็ก ๆ 2 ช่องที่อยู่ติดกันคือ

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx \quad = \quad \int_{x_{i-1}}^{x_{i+1}} (A + Bx + Cx^2)\, dx$$

$$= \quad Ax + \frac{Bx^2}{2} + \frac{Cx^3}{3}\bigg|_{x_{i-1}}^{x_{i+1}}$$

แทนค่า x ด้วยขีดจำกัดบนและล่าง แทนค่า A,B, C จะได้

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx \quad = \quad \frac{h}{3} f(x_{i-1}) + \frac{4}{3} h\, f(x_i) + \frac{h}{3} f(x_{i+1})$$

หาพื้นที่ทั้งหมด โดยรวมพื้นที่ส่วนเล็ก ๆ ทั้งหมด

$$\int_a^b f(x)dx \quad \cong \quad \frac{h}{3} f(x_1) + \frac{4h}{3} f(x_2) + \frac{2h}{3} f(x_3) + \frac{4h}{3} f(x_4) + \ldots + \frac{4h}{3} f(x_{N-1}) + \frac{h}{3} f(x_N) \qquad \text{........(4.28)}$$

ค่าน้ำหนัก (w_i) ที่จุด x_i ใด ๆ คือ

$$w_i \quad = \quad \left\{\frac{h}{3}, \frac{4h}{3}, \frac{2h}{3}, \frac{4h}{3}, \ldots \frac{4h}{3}, \frac{h}{3}\right\}, \; i=1, \ldots N$$

จะเห็นว่าค่าน้ำหนักจะซ้ำ ๆ กัน ระหว่าง $\frac{4h}{3}$ และ $\frac{2h}{3}$ ที่จุดปลายทั้ง 2 ข้างค่าน้ำหนักจะเป็น $\frac{h}{3}$

ค. **กฎของซิมป์สัน** $\frac{3}{8}$ (Simpson's $\frac{3}{8}$ rule)

ถ้าเส้นที่ลากเชื่อมจุดต่าง ๆ บนเส้นกราฟเป็นพหุนามมีองศาเท่ากับ 3 $f(x) = A+Bx+Cx^2+Dx^3$ ต้องใช้ค่า x ถึง 4 จุดเพื่อที่จะหาค่า A, B, C และ D จึงต้องใช้ช่องเล็ก ๆ 3 ช่อง ในการหาพื้นที่เล็ก ๆ 1 ครั้ง จำนวนช่องหรือ interval จึงเป็นเลขที่จำนวน 3 หารได้ลงตัว

พื้นที่ใต้เส้นโค้ง 3 ช่องเล็ก ๆ 1 ครั้ง ตั้งแต่ $x_{i-1}, x_i, x_{i+1}, x_{i+2}$

$$\int_{x_{i-1}}^{x_{i+2}} f(x)dx = \frac{3}{8}h\, f(x_{i-1}) + \frac{9}{8}h\, f(x_i) + \frac{9}{8}h\, f(x_{i+1}) + \frac{3}{8}h\, f(x_{i+2})$$

พื้นที่ทั้งหมดตั้งแต่ x = a ถึง x = b คือ

$$\int_a^b f(x)dx = \frac{3}{8}h[(f(x_1)+f(x_N))+3(f(x_2)+f(x_3)+f(x_5)+f(x_6) + \ldots f(x_{N-1}) + 2(f(x_4)+f(x_7) + f(x_{10}) + \ldots f(x_{N-2}))] \qquad \ldots\ldots\ldots..(4.29)$$

ค่าน้ำหนัก (w_i) ที่จุด x_i ใด ๆ คือ

$$w = \left\{\frac{3h}{8}, \frac{9h}{8}, \frac{9h}{8}, \frac{6h}{8}, \frac{9h}{8}, \frac{9h}{8}, \frac{6h}{8}, \ldots, \frac{9h}{8}, \frac{3h}{8}\right\} \text{ เมื่อ } i=1\ N$$

ง. **วิธีของเกาส์-เลอจองด์** (Legendre-Gauss Quadrature)

วิธีนี้ไม่ต้องแบ่งพื้นที่ใต้เส้นโค้งออกเป็นส่วนเล็ก ๆ ที่มีขนาดเท่ากัน แต่ใช้วิธีแบ่งช่วง a ถึง b ออกเป็นจุด ๆ ที่เรียกว่า node แล้วใช้พหุนามเลอจองด์(Legendre polynomial)ในการประมาณค่าการหาปริพันธ์ โดยทั่วไปยิ่งใช้พหุนามที่มีองศาสูงมากขึ้นเท่าใด ผลลัพธ์ที่ได้ก็จะใกล้ค่าแท้จริงมากขึ้นเพียงนั้น แต่คอมพิวเตอร์จะต้องใช้เวลาในการคำนวณมาก

กำหนดให้ผลลัพธ์ของการอินทิเกรทได้ค่าประมาณอยู่ในรูปของผลบวกของฟังก์ชัน

$$\int_a^b f(x)dx = I = w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3) + \ldots + w_n f(x_N)$$

ให้ N เป็นจำนวนเทอมของฟังก์ชัน ค่าน้ำหนัก(w_i)และตำแหน่ง(node)ที่จะใช้แทนค่าฟังก์ชัน เพื่อจะประมาณค่าการหาอนุพันธ์ จะเป็นดังนี้

N	ตำแหน่งของnode	w_i
2	0.5773502691	1
3	0	0.8888888888
	0.7745966692	0.5555555555
4	0.3399810435	0.6521451548
	0.8611363115	0.3478548451
5	0	0.5688888888
	0.5384693101	0.4786286704
	0.9061798459	0.2369268850
6	0.2386191860	0.4679139345
	0.6612093864	0.3607615730
	0.9324695142	0.1713244923
8	0.183434642	0.362683783
	0.525532410	0.313706646
	0.796666478	0.222381034
	0.960289857	0.101228536

ผู้วิจัยได้พัฒนาคลาสใช้สำหรับคำนวณการหาปริพันธ์ไว้ 4 แบบ คือ คลาส TrapezoidalIntegration คลาส Simpson1_3Integration คลาส Simpson3_8Integration และคลาส GaussQuadratureIntegration คลาสทั้งหมดใช้หาปริพันธ์ของฟังก์ชันที่มีค่าต่อเนื่องในช่วง a ถึง b และเป็นการหาปริพันธ์แบบชั้นเดียว คลาสทั้งสี่สืบทอดคุณสมบัติมาจากคลาสแม่ซึ่งเป็นคลาสนามธรรมชื่อ Integrator ซึ่งมีเมธอดที่จะต้องนำไป override ในคลาสลูกได้แก่ find AreaUnderCurve() และ Integrate()

ความผิดพลาดที่อาจเกิดขึ้นและคลาส IntegrationException ทำหน้าที่ดักจับความผิดพลาดเหล่านี้ ได้แก่ ขีดจำกัดล่าง (a) และขีดจำกัดบน (b) ของการหาปริพันธ์ต้องไม่เท่ากัน, ขนาดของช่องเล็ก ๆ (h) จะมีค่าเป็นลบไม่ได้ และจำนวน node ที่ใช้ในการหาปริพันธ์โดยวิธีเกาส์-เลอจองด์อยู่ระหว่าง 2 ถึง 8

รายละเอียดของคลาสที่ใช้ในการคำนวณหาปริพันธ์มีดังนี้

คลาส Integrator ซึ่งเป็นคลาสแม่หรือคลาสหลักของทุก ๆ คลาส

```
/* File : Integrator.java */

package MathTools.Integration;

import static MathTools.Common.CommonConstants.*;
import MathTools.Common.Function;

/** Abstract class for integration */

public abstract class Integrator {
      /** Function to integrate */
                  Function function;
      /** the number of equal-width intervals */
                  int intervals;
      /** the lower limit of integration */
                  double lower;
      /** the upper limit of integration */
                  double upper;
      /** Integration results */
                  double IntegrationResult;

 Integrator() { }

 Integrator( Function f, double a, double b, int n){
       this.function = f;
       this.lower = a;
       this.upper = b;
       this.intervals = n;
 }
 /** getting the Integration result */
 public double getIntegrationResult()
 { return IntegrationResult;}
 /** Compute the area at nth   region */
 abstract double integrate()throws IntegrationException;
 abstract double findAreaUnderCurve(double x_left, double h);
 }
```

รายละเอียดของคลาส TrapezoidalIntegration สำหรับใช้หาค่าปริพันธ์โดยแบ่งพื้นที่ใต้โค้งเป็นรูปสี่เหลี่ยมคางหมู

```
/* File :TrapezoidalIntegration.java */
package MathTools.Integration;

import static java.lang.Math.*;

import MathTools.Integration.*;
import MathTools.Common.Function;
import static MathTools.Common.CommonConstants.*;

/** Integration of given function from lower limit
      to upper limit using the trapezoidal rule.
*/
public class TrapezoidalIntegration extends Integrator{

/** Constructor.
* @param function the function to integration.
```

```
 * @param lower the lower limit of integration.
 * @param upper the upper limit of integration.
 * @param intervals the number of equal-width intervals
 */
public TrapezoidalIntegration (Function function,
        double lower, double upper, int intervals){
            super(function, lower, upper,intervals);
            try {
                IntegrationResult = integrate();
            } catch (IntegrationException msg) {
                System.out.println( "Error : "+ msg);
            }
}
double integrate() throws IntegrationException {
        if(intervals <=0) throw new
 IntegrationException(IntegrationException.INVALID_INTERVALS);
        double h = (upper - lower)/intervals;
        double area = 0;
        for(int i = 0; i < intervals; i++){
            double x_left = lower + i*h;
            area += findAreaUnderCurve(x_left,h);
        }
        return area;
}
double findAreaUnderCurve(double x_left, double h){
        double x_right = x_left + h;
        double y_left = function.Of(x_left);
        double y_right = function.Of(x_right);
        return (h*(y_left + y_right)/2);  // area of trapezoidal
}
}

```

รายละเอียดของคลาส Simpson1_3Integration สำหรับใช้หาค่าปริพันธ์โดยแบ่งพื้นที่ใต้โค้งโดยมีเส้นเชื่อมจุดต่าง ๆ บนกราฟเป็นแบบพาราโบลา

```
/* File :Simpson1_3Integration.java */
package MathTools.Integration;

import static java.lang.Math.*;

import MathTools.Integration.*;
import MathTools.Common.Function;
import static MathTools.Common.CommonConstants.*;

/** Integration of given function from lower limit
        to upper limit using the Simpson 1/3 method.
*/
public class Simpson1_3Integration extends Integrator{

/** Constructor.
 * @param function the function to integration.
 * @param lower the lower limit of integration.
 * @param upper the upper limit of integration.
 * @param intervals the number of equal-width intervals
 */
public Simpson1_3Integration (Function function,
        double lower, double upper, int intervals){
            super(function, lower, upper,intervals);
            try {
```

```
                IntegrationResult = integrate();
            } catch (IntegrationException msg) {
                System.out.println( "Error : "+ msg);
            }
}
double integrate() throws IntegrationException {
        if(intervals <=0) throw new
IntegrationException(IntegrationException.INVALID_INTERVALS);
        double h = (upper - lower)/(2*intervals);
        double area = 0;
        for(int i = 0; i < intervals; i++){
                double x1 = lower + 2*i*h;
                area += findAreaUnderCurve(x1,h);
        }
        return area;
}
/** Evaluate the area under parabolic curve
* @param x1 the left bound of the region
* @param h the interval width
*/
double findAreaUnderCurve(double x1, double h){
        // area of the parabolic region = h/3(f(-h) +4f(0) +f(h))

        double x2 = x1 + h; // middle point
        double x3 = x2 + h; // rightmost point
        double y1 = function.Of(x1);
        double y2 = function.Of(x2);
        double y3 = function.Of(x3);

        return (h*(y1 + 4*y2 +y3)/3);
}
}
```

รายละเอียดของคลาส Simpson3_8Integration สำหรับใช้หาค่าปริพันธ์โดยแบ่งพื้นที่ใต้โค้งโดยมีเส้นเชื่อมจุดต่าง ๆ บนกราฟเป็นแบบพหุนามองศาสาม

```
/* File :Simpson3_8Integration.java */
package MathTools.Integration;

import static java.lang.Math.*;

import MathTools.Integration.*;
import MathTools.Common.Function;
import static MathTools.Common.CommonConstants.*;

/** Integration of given function from lower limit
        to upper limit using the Simpson 3/8 method.
*/
public class Simpson3_8Integration extends Integrator{

/** Constructor.
* @param function the function to integration.
* @param lower the lower limit of integration.
* @param upper the upper limit of integration.
* @param intervals the number of equal-width intervals
*/
public Simpson3_8Integration (Function function,
        double lower, double upper, int intervals){
```

```
                super(function, lower, upper,intervals);
                try {
                      IntegrationResult = integrate();
                } catch (IntegrationException msg) {
                      System.out.println( "Error : "+ msg);
                }
}
double integrate() throws IntegrationException {
      if(intervals <=0) throw new
IntegrationException(IntegrationException.INVALID_INTERVALS);
      double h = (upper - lower)/(3*intervals);
      double area = 0;
      for(int i = 0; i < intervals; i++){
            double x1 = lower + 3*i*h;
            area += findAreaUnderCurve(x1,h);
      }
      return area;
}
/** Evaluate the area under parabolic curve
* @param x1 the left bound of the region

* @param h the interval width
*/
double findAreaUnderCurve(double x1, double h){
// area of the 3rd degree region = 3h(y0 + 3y1 + 3y2 +y3)/8
      double x2 = x1 + h;
      double x3 = x2 + h;
      double x4  = x3+h;

      double y1 = function.Of(x1);
      double y2 = function.Of(x2);
      double y3 = function.Of(x3);
      double y4 = function.Of(x4);
      return (3*h*(y1 + 3*y2 + 3*y3 + y4)/8);
}
}
```

รายละเอียดของคลาส GaussQuadratureIntegration สำหรับใช้หาค่าปริพันธ์โดยกำหนดจำนวน node และน้ำหนักของฟังก์ชันโดยอิงอาศัยพหุนามเลอจองด์

```
/* File :GaussQuadratureIntegration.java */
package MathTools.Integration;
import static java.lang.Math.*;
import MathTools.Integration.*;
import MathTools.Common.Function;
import static MathTools.Common.CommonConstants.*;
/** Integration of given function from lower limit
      to upper limit using the Gauss Quadrature method.
*/
public class GaussQuadratureIntegration extends Integrator{
/** the number of node for integration */ int node;
/** Constructor.
* @param function the function to integration.
* @param lower the lower limit of integration.
```

```
* @param upper the upper limit of integration.
* @param node the number of node for integration from 2 to 6
*/
public GaussQuadratureIntegration (Function function,
        double lower, double upper, int node){
            this.function = function;
            this.lower = lower;
            this.upper = upper;
            this.node = node;

            try {
                IntegrationResult = integrate();
            } catch (IntegrationException msg) {
                System.out.println( "Error : "+ msg);
            }
}
double integrate() throws IntegrationException {
double[] weight = new double[9]; // weight of this point
double[] u = new double[9]; // node point of function.
double[] x = new double[9];
double area =0;
        if(node < 2 || node >8 || (upper == lower)) throw new
        IntegrationException(IntegrationException.INVALID_NODE);
            switch (node) {
            case 2 : { u[1]= -1/sqrt(3.);
                            u[2] = -u[1];
                            weight[1] = weight[2] = 1;
                            break;
            }
            case 3 :{ u[1] = -sqrt(15.)/5.;
                            u[2] = 0;
                            u[3] = -u[1];
                            weight[1] = 5.0/9.0;
                            weight[2] = 8.0/9.0 ;
                            weight[3] = weight[1];
                            break;
            }
            case 4 :  { u[1] =  -sqrt(525+70*sqrt(30))/35;
                        u[2] =  -sqrt(525-70*sqrt(30))/35;
                        u[3] =  -u[2];
                        u[4] =  -u[1];
                        weight[1] = (18. - sqrt(30))/36.0;
                        weight[2] = (18. + sqrt(30))/36.0;
                        weight[3] = weight[2];
                        weight[4] = weight[1];
                          break;
            }
            case 5 :  { u[1] = -sqrt(245.+14*sqrt(70.))/21.0;
                        u[2] = -sqrt(245.-14*sqrt(70.))/21.0;
                        u[3] = 0;
                        u[4] = -u[2]; //u[2] = -u[4]
                        u[5] = -u[1]; // 0.9061798459
            weight[1] =  weight[5] = (322-13*sqrt(70))/900.0;
            weight[2] =  weight[4] =(322+13*sqrt(70))/900.0;
                        weight[3] = 128./224. ;
                         break;
            }
            case 6 : {  u[1] = -0.9324695142;
                              u[2] = -0.6612093864;
                              u[3] = -0.2386191860;
                              u[4] =  0.2386191860;
```

```
                                u[5] = 0.6612093864;
                                u[6] = 0.9324695142;

                           weight[1] = weight[6] = 0.1713244923;
                           weight[2] = weight[5] = 0.3607615730;
                           weight[3] = weight[4] = 0.4679139345;
                           break;

                 }
                 case 7 : {  u[1] = -0.94910791;
                                u[2] = -0.74153119;
                                u[3] = -0.40584515;
                                u[4] =  0;
                                u[5] = -u[3];
                                u[6] = -u[2];
                                u[7] = -u[1];

                            weight[1] = weight[7] = 0.12948497;
                            weight[2] = weight[6] = 0.27970539;
                            weight[3] = weight[5] = 0.38183005;
                            weight[4] = 0.41795918;
                         break;
                 }
                 case 8 : {  u[1] = -0.96028986;
                                u[2] = -0.79666648;
                                u[3] = - 0.52553241;
                                u[4] =  - 0.18343464;
                                u[5] = 0.18343464;
                                u[6] = 0.52553241;
                                u[7] = 0.79666648;
                                u[8] = 0.96028986;


                            weight[1] = weight[8] = 0.10122854;
                            weight[2] = weight[7] =0.22238103;
                            weight[3] = weight[6] = 0.31370665;
                            weight[4]  = weight[5] = 0.36268378 ;
                         break;

                 }
                 }// switch(node)
                 for(int i = 1; i <= node; i++) {
                 x[i] = 0.5*((upper+lower)+(upper - lower)*u[i]);
                 area += findAreaUnderCurve(x[i], weight[i]);
                 }
                 return      area*(upper - lower)/2;

}
/** Evaluate the area under  curve
* @param point the node value.
* @param w the weight of function at this node.
*/
double findAreaUnderCurve(double point, double w){
       return function.Of(point)*w;
}
}
```

4.4.4 การทดสอบการหาอนุพันธ์และปริพันธ์

ตัวอย่าง 4.4.1 จงหาอนุพันธ์อันดับหนึ่งของ $y = x^2 \cos(x)$ ที่จุด $x = \pi / 2$

วิธีทำ โปรแกรมที่ใช้ทดสอบการหาอนุพันธ์อันดับหนึ่งของ $y = x^2 \cos(x)$ มีดังนี้

```
/* File : FirstDerivativeTest.java */
import static MathTools.Common.CommonConstants.*;
import MathTools.Common.Function;
import MathTools.Derivative.*;

import static java.lang.Math.*;

public class FirstDerivativeTest {

public static void main(String[ ] args) {
       double x=PI/2.0;
       // insert  function in here
       Function func = new Function() {

              public double Of(double x) {
                     return x*x*cos(x);
              }

       };
 // FirstDerivative(fucntion,xDerive,tolerance)

 FirstDerivative fd = new FirstDerivative(func,x);
              System.out.println("\n The result of Derivative at
              x = "+x+ "  is " + fd.getDerivativeResult());
 }
 }
```

เมื่อให้โปรแกรมทำงาน ผลลัพธ์ที่ได้คือ

```
The result of Derivative at x = 1.5707963267948966  is -2.467401100272417
```

ค่าอนุพันธ์ที่แท้จริงคือ $\dfrac{dy}{dt} = 2x\cos x - x^2 \sin x$

แทนค่า $x = \pi / 2$ จะได้ $\dfrac{dy}{dt} = -\dfrac{\pi^2}{4} = -2.46740110027234$

ตัวอย่าง 4.4.2 จงหาอนุพันธ์อันดับสองของ $y = x^2 \cos(x)$ ที่จุด $x = \pi / 2$

วิธีทำ โปรแกรมที่ใช้ทดสอบการหาอนุพันธ์อันดับสองของ $y = x^2 \cos(x)$ มีดังนี้

```
/* File : SecondDerivativeTest.java */
import static MathTools.Common.CommonConstants.*;
import MathTools.Common.Function;
import MathTools.Derivative.*;

import static java.lang.Math.*;

public class SecondDerivativeTest {

public static void main(String[] args) {
        double x=PI/2.0;
        // insert  function in here
        Function func = new Function() {

                public double Of(double x) {
                        return x*x*cos(x);
                }

        };
        SecondDerivative sd = new SecondDerivative(func,x);
                System.out.println("\n The result of second
                derivative at x = "+x+ "  is " +
                sd.getSecondDerivativeResult());
 }
 }
```

ผลลัพธ์ที่ได้จากการให้โปรแกรมทำงานมีดังนี้

```
The result of second derivative at x = 1.5707963267948966  is -6.28318530717974
```

อนุพันธ์อันดับสองของ $y = x^2 \cos(x)$ คือ

$$\frac{d^2y}{dt^2} = 2\cos x - 4x\sin x - x^2\cos x$$

แทนค่า $x = \pi / 2$ จะได้

$$\frac{d^2y}{dt^2} = -2\pi = -6.28318530718$$

ตัวอย่าง 4.4.3 งานที่ใช้ในการเคลื่อนวัตถุจากตำแหน่ง x = a ถึง x = 2a หาได้จาก

$$W = \int_{a}^{2a} \frac{kdx}{(x^2+a^2)^{\frac{3}{2}}}$$

จงหาค่าของงานนี้ เมื่อ a = 1, k = 0.5

วิธีทำ โปรแกรมที่ใช้คำนวณหาค่าปริพันธ์ของโจทย์ในตัวอย่างนี้มีดังนี้

```
/* File : IntegrationTest.java */
import static MathTools.Common.CommonConstants.*;
import MathTools.Common.Function;
import MathTools.Integration.*;
import static java.lang.Math.*;

public class IntegrationTest {

public static void main(String[] args) {
long usedTime,start;

      // insert integrand function in here
      Function func = new Function() {
            public double Of(double x) {
                  return (0.5/pow((x*x+ 1),1.5));
      }
      };
start = System.nanoTime() ;
   TrapezoidalIntegration tp =
      new TrapezoidalIntegration(func,1,2,1000);
System.out.println("\n The result of Integration using
 Trapezoidal rule = " + tp.getIntegrationResult());
usedTime = System.nanoTime() - start;
System.out.println("\nExecuted time = "+usedTime+" ns");
//====================================
start = System.nanoTime() ;
Simpson1_3Integration ss1 =
      new   Simpson1_3Integration(func,1,2,1000);
System.out.println("\n The result of Integration using Simpson
      1/3 rule = " + ss1.getIntegrationResult());
usedTime = System.nanoTime() - start;
System.out.println("\nExecuted time = "+usedTime+" ns");
//=========================================
start = System.nanoTime() ;
Simpson3_8Integration ss2 =
       new  Simpson3_8Integration(func,1,2,1000);
System.out.println("\n The result of Integration using
      Simpson 3/8 rule = " + ss2.getIntegrationResult());
usedTime = System.nanoTime() - start;
System.out.println("\nExecuted time = "+usedTime+" ns");
//==========================================
start = System.nanoTime() ;
GaussQuadratureIntegration gq4 = new
      GaussQuadratureIntegration(func,1,2,8);
System.out.println("\n The result of Integration using
GaussQuadrature  8 point = " + gq4.getIntegrationResult());
usedTime = System.nanoTime() - start;
```

```
System.out.println("\nExecuted time = "+usedTime+" ns");
}
}
```

ผลลัพธ์ที่ได้จากการทดสอบให้โปรแกรมทำงาน ทุกวิธีให้ผลลัพธ์ใกล้เคียงค่าแท้จริง วิธีของเกาส์เลอจองด์ ใช้เวลาในการคำนวณน้อยที่สุด ขณะที่กฎสี่เหลี่ยมคางหมูใช้เวลามากที่สุด

The result of Integration using Trapezoidal rule = 0.09366022253163486

Executed time = 7971125 ns

The result of Integration using Simpson 1/3 rule = 0.09366020490668424

Executed time = 3000940 ns

The result of Integration using Simpson 3/8 rule = 0.0936602049066842

Executed time = 4742223 ns

The result of Integration using GaussQuadrature 8 point = 0.09366020498197819

Executed time = 971632 ns

ค่าที่แท้จริงจะได้

$$W = \frac{k}{a^2} \sin\left[\tan^{-1} \frac{x}{9}\right]$$

เมื่อแทนค่าขีดจำกัดบนและล่าง แทนค่า k และ a แล้ว จะได้

$$W = 0.5\left(\frac{2}{\sqrt{5}} - \frac{1}{\sqrt{2}}\right)$$

$$= 0.093660$$

4.5 การหาคำตอบของสมการอนุพันธ์แบบสามัญ

การพัฒนาคลาสไลบรารีเพื่อใช้คำนวณหาคำตอบของสมการเชิงอนุพันธ์มีข้อจำกัดดังนี้ คือใช้กับสมการเชิงอนุพันธ์แบบสามัญ (Ordinary differential equation) เป็นสมการอนุพันธ์ที่มีตัวแปรอิสระเพียงตัวเดียว มีลักษณะเป็นเชิงเส้น ค่าอนุพันธ์ที่ปรากฏในสมการเป็นค่าที่หาเทียบกับตัวแปรเพียงตัวเดียว ไม่ใช่เป็นแบบ partial derivative เขียนให้อยู่ในรูปทั่วไปได้ดังนี้

$$y' = (x, y) \qquad \text{.................... (4.30)}$$

โดยบอกเงื่อนไขที่จุดเริ่มต้น (initial condition) คือ $y(x_0) = y_0$

ระเบียบวิธีที่ใช้หาคำตอบของสมการอนุพันธ์ในที่นี้จะใช้อยู่ 2 วิธี ได้แก่ วิธีแบบขั้นเดียว (one step techniques) โดยเริ่มประมาณค่าตัวแปรอิสระ (x) ตามเงื่อนไขเริ่มต้น และนำค่าที่หาได้นี้ไปคำนวณหาค่าจุดต่อไปบนโค้งของ f(x) ได้แก่ วิธีของออยเลอร์ที่ปรับปรุงแล้ว (modified Euler's method) วิธีของ รังเง-คุตตา (Runge-Kutta method) อีกวิธีหนึ่งคือวิธีแบบหลายขั้น (multi step techniques) เป็นวิธีที่หาจุดถัดไปบนเส้นโค้ง f(x) โดยใช้ข้อมูลจากการคำนวณที่ผ่านมาหลายจุด จะมีการวนรอบจนกระทั่งได้ค่าความคลาดเคลื่อนไม่เกินที่กำหนดไว้ ได้แก่ วิธีของ อาดัม-มูลตัน (Adam-Moulton's method)

4.5.1 วิธีของออยเลอร์ (Forward Euler and Modified Euler Method)

เป็นวิธีที่ง่ายและธรรมดาที่สุด สะดวกที่จะใช้เขียนโปรแกรมให้คอมพิวเตอร์ทำงาน และพื้นฐานสำคัญในการนำไปหาคำตอบสมการอนุพันธ์ย่อย (partial differential equation)

การประมาณค่าการหาอนุพันธ์ สามารถนิยามได้ดังนี้

$$y' \cong \frac{y_{n+1} - y_n}{h}$$

หรือ
$$y_{n+1} \cong y_n + hy'$$

แทนค่า y' จากสมการ (4.30) จะได้

$$y_{n+1} \cong y_n + hf(x, y) \qquad \text{............. (.4.31)}$$

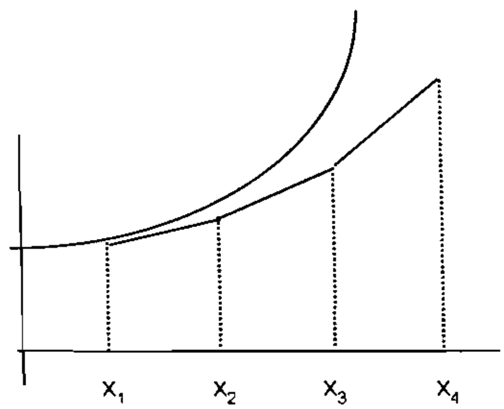
เมื่อกำหนดเงื่อนไขเริ่มต้นมาให้ $y(x_0) = y_0$ เราสามารถนำเงื่อนไขนี้มาหาค่า y ครั้งต่อ ๆ ไปได้

$$y_1 = y_0 + hf(x_0, y_0)$$

$$y_2 = y_1 + hf\,(x_1,\, y_1)$$

$$\vdots \qquad\qquad \vdots$$

$$y_n = y_{n+1} + hf\,(x_{n-1},\, y_{n-1})$$

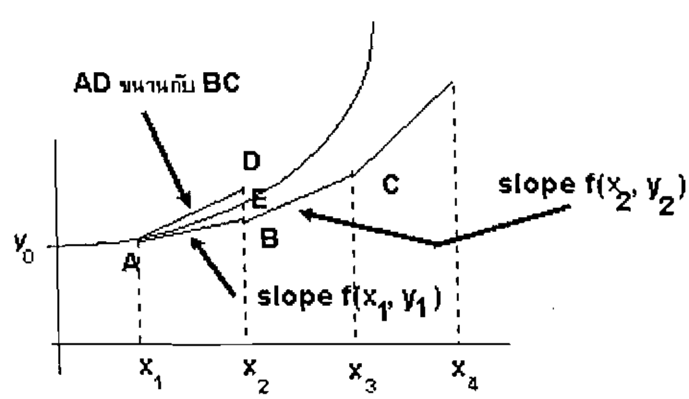


รูป 4.7. เงื่อนไขเริ่มต้น x_0 นำไปหาค่า y ที่จุด x_1,x_2,.... ได้

ข้อบกพร่องของวิธีนี้เห็นได้ชัดว่าที่จุด x ที่อยู่ห่างจากค่าเริ่มต้นมาก ๆ ความคลาดเคลื่อนที่เกิดเมื่อคำนวณจะมากตามระยะที่ห่างไปด้วย ด้วยเหตุนี้จึงจำเป็นต้องปรับปรุงขั้นตอนวิธีของออยเลอร์ เพื่อให้ได้ผลลัพธ์ที่ใกล้เคียงความจริงมากขึ้น

วิธีออยเลอร์แบบปรับปรุงแล้ว (Modified Euler's Method) เป็นการนำเอากฎสี่เหลี่ยมคางหมูมาช่วยหาคำตอบสมการดิฟเฟอเรนเชียล สมการ (4.31) จะกลายเป็น

$$y_{n+1} = y_n + \frac{h}{2}\left[f\,(x_{n+1},\, y_{n+1})\; f\,(x_n,\, y_n)\right] \qquad .... \qquad (4.32)$$



รูป 4.8 แสดงวิธีของออยเลอร์แบบปรับปรุงแล้ว

จากรูปจะเห็นว่า y_0+ hf (x_0, y_0) คือเส้น AB ซึ่งจะได้ต่ำกว่าค่าที่เป็นจริง (ต่ำกว่าจุด E) และ y_0 + hf (x_1, y_1) คือเส้น AD มีค่าสูงกว่าค่าจริง เมื่อใช้ค่าเฉลี่ยของค่าทั้งสองก็จะได้ค่าประมาณที่ใกล้ค่าจริงคือ AE จึงเป็นที่มาของสมการ (4.32)

ปัญหาอีกประการหนึ่งของสมการ (4.32) คือ y_{n+1} เป็นตัวไม่ทราบค่า ซึ่งจะพบว่ามีอยู่ทางด้านขวามือของสมการ (4.32) ด้วย การนำสมการ (4.32) ไปใช้จึงต้องประมาณค่า y_{n+1} ทางด้านขวามือก่อน เรียกว่าเป็นตัวทำนาย (predict) แล้วจึงนำไปหาค่า y_{n+1} ที่ถูกต้องมากยิ่งขึ้นในสมการ (4.32) (correct) การพยากรณ์ค่า y_{n+1} หาได้จากสมการ (4.31)

4.5.2 **วิธีของรังเง-คุตตา** (Runge - Kutta method)

ถึงแม้ว่าวิธีของออยเลอร์ที่ถูกปรับปรุงแล้วจะให้ผลลัพธ์ใกล้เคียงกับค่าที่แท้จริงมากขึ้น แต่ความคลาดเคลื่อนเมื่อหาคำตอบของสมการอนุพันธ์ที่จุดต่างๆ ซึ่งห่างจากจุดเริ่มต้นมาก ๆ ยังคงปรากฏให้เห็นอย่างชัดเจน นักคณิตศาสตร์ชาวเยอรมัน 2 คน คือ รังเง (Runge) และคุตตา (Kutta) ได้เสนอวิธีหาคำตอบสมการอนุพันธ์ที่มีประสิทธิภาพให้ความเที่ยงตรงมากกว่า ใช้เวลาในการคำนวณน้อยกว่า

หลักการของรังเง - คุตตา อาศัยการเพิ่มจำนวนเทอมซึ่งปกติจะถูกตัดทิ้ง เป็นการหาค่าประมาณของอนุพันธ์อันดับสองในเทอมของอนุพันธ์อันดับที่ 1 ตรงจุด x_i และ x_{i+1} ซึ่งจำเป็นจะต้องหาค่าฟังก์ชัน ณ ตรงจุด $x_i + \alpha h$ โดยที่ α เป็นค่าใด ๆ ถ้าเราเพิ่มจำนวนเทอมซึ่งเป็นอนุพันธ์อันดับสูงกว่าสอง ซึ่งก็ต้องหาค่าการเปลี่ยนแปลงของฟังก์ชันที่ค่า α ต่าง ๆ กัน เราจะได้สูตรสำหรับวิธีของรังเง - คุตตา มากมาย ขึ้นอยู่ว่าเราต้องการความละเอียดและตัดเทอมต่าง ๆ ทิ้งตรงตำแหน่งใด

โดยทั่วไปแล้วจะนิยมใช้วิธีรังเง-คุตตา อันดับ 4 ซึ่งได้มาจากการประมาณค่าถึงเทอม h^4 จะได้ตัวแปร 13 ตัวแปร จำนวนสมการ 11 สมการ มีตัวแปร 2 ตัว ซึ่งต้องถูกกำหนดค่าได้ตามใจชอบ สมการที่ใช้กันอย่างแพร่หลายสำหรับวิธีรังเง-คุตตาอันดับ 4 คือ

$$\left.\begin{aligned}
y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
k_1 &= h\,f(x_n, y_n) \\
k_2 &= h\,f(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\
k_3 &= h\,f(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\
k_4 &= h\,f(x_n + h, y_n + k_3)
\end{aligned}\right\} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (4.33)$$

4.5.3 วิธีของอาดามส์-มูลตัน

การหาคำตอบค่าถัดไปของสมการอนุพันธ์โดยวิธีรังเง-คุตตาจะใช้คำตอบจากค่าที่คำนวณได้ไว้แล้วก่อนหน้านั้นเพียง 1 ค่า เรียกวิธีแบบนี้ว่าเป็นวิธีคำนวณขั้นตอนเดียว (one step method) วิธีของอาดามส์-มูลตัน เป็นวิธีที่หาคำตอบโดยใช้ 2 สูตรควบคู่กัน สูตรแรกจะใช้เป็นตัวทำนาย (predict) คำตอบที่ได้ แล้วจึงใช้สูตรที่ 2 ปรับค่า (correct) ที่ได้จากสูตรที่ 1 จนได้ค่าความคลาดเคลื่อนน้อยกว่าที่กำหนดไว้ ทั้งสูตรที่ 1 และสูตรที่ 2 ต้องใช้ค่าที่ได้จากขั้นตอนก่อนหน้านี้ถึง 4 จุดมาช่วยในการคำนวณ เรียกวิธีนี้ว่าเป็นวิธีคำนวณหลายขั้นตอน (multi step method) หรือวิธีใช้ตัวทำนาย-ตัวแก้ไข (predictor - corrector method)

วิธีของอาดามส์-มูลตัน ใช้กับสมการอนุพันธ์อันดับหนึ่ง ซึ่งมีรูปทั่วไปดังสมการ (4.30) โดยทั่วไปฟังก์ชัน f (x, y) สามารถแทนได้ด้วยพหุนาม พหุนามนี้จะแทนค่า f(x,y) ได้ใกล้เคียงเมื่อกระจายให้ x มีอันดับสูงเท่าที่จะทำได้

เมื่อประมาณค่าฟังก์ชันเป็นพหุนามถึง x^3 โดยใช้จุด 4 จุดในการพิจารณา จะได้สมการสำหรับเป็นตัวทำนาย (Predictor) ดังนี้

$$y_{i+1} = y_i + \frac{h}{24}(55f_i + 59f_{i-1} + 37f_{i-2} - 9f_{i-3}) \qquad \text{..................(4.34)}$$

เพื่อที่จะได้ค่าที่ใกล้เคียงค่าแท้จริง จึงหาสูตรสำหรับใช้ปรับค่าสมการ (4.34) อีกสูตรหนึ่ง สูตรนี้ได้จากการหาปริพันธ์โดยอาศัยจุด 4 จุดของฟังก์ชันเช่นเดียวกับที่ได้ในสมการ (4.34) แต่เลื่อนล้ำไปข้างหน้าอยู่หนึ่งจุด จะได้สมการที่ใช้แก้ไขค่าทำนายดังนี้

$$y_{i+1} = y_i + \frac{h}{24}(f_{i-2} - 5f_{i-1} + 19f_i + 9f_{i+1}) \qquad \text{................. (4.35)}$$

ในการออกแบบได้สร้างคลาสDifferentialEquation ซึ่งสืบทอดมาจากคลาส Function สำหรับเก็บค่า f(x,y) และเงื่อนไขเริ่มต้น คลาสDifferentialEquationSolver เป็นคลาสแม่ มีตัวแปรคลาสและเมธอดที่ทุก ๆ วิธีจะต้องใช้ร่วมกัน คลาส ModifiedEuler คลาส RungeKutta และคลาส AdamsMoulton ต่างก็สืบทอดมาจากคลาสแม่นี้ทั้งสิ้น

รายละเอียดของคลาส DifferentialEquationSolver มีดังนี้

```
/* File : DifferentialEqSolutionFinder.java */

package MathTools.DifferentialEq;
import static MathTools.Common.CommonConstants.*;
import MathTools.Common.*;
import MathTools.DifferentialEq.*;

/** Abstract base class for differential equation solving */
```

```
public abstract class DifferentialEquationSolver  {
  /** the differential equation to be solved */
      DifferentialEquation  differentialEquation;
      /** initial condition */       DataItem initialValue;
      /** current x */              double current_x;
      /** current y */              double current_y;
      /** spacing of each data point */
                           double h = DEFAULT_INTERVAL;
      /** counter for check the number of  nextStep */
                                 int counter;
      /** the solution of differential equation at x */
                                 public  double solution;

/** Constructor:
* @param diffEquation the differential equation to be solved.
*/
public DifferentialEquationSolver
      (DifferentialEquation differentialEquation ){
this.differentialEquation = differentialEquation;
this.initialValue = differentialEquation.getInitialCondition();
      current_x = initialValue.x;
      current_y = initialValue.y;
}
/** Constructor:
* @param diffEquation the differential equation to be solved.
* @param h  an interval of data point change
*/
public DifferentialEquationSolver
(DifferentialEquation differentialEquation,double interval ){
this.differentialEquation = differentialEquation;
this.initialValue = differentialEquation.getInitialCondition();
      current_x = initialValue.x;
      current_y = initialValue.y;
      if (interval > 0) this.h = interval;
}

/** The next result of computing  */
public abstract DataItem nextStep();

/** get the solution of Differential equation at x
* @return the solution.
*/

public double getSolution(){ return solution;}

/** the number of counting how many time nexStep() does loops.
* @return counter the number of counting
*/

public int getCounter() { return counter;}

}
```

ต่อไปนี้รายละเอียดของคลาส ModifiedEuler ในคลาสนี้ได้ override เมธอด NextStep() ตามวิธีการประมาณคำตอบของออยเลอร์ที่ได้ปรับปรุงวิธีการคำนวณแล้ว

```
/* File : ModifiedEuler.java  */

package MathTools.DifferentialEq;

import MathTools.Common.*;
import MathTools.DifferentialEq.*;
/** Solving the 1st order differential equation by using
modified Euler's method
*/

public class ModifiedEuler extends DifferentialEquationSolver {

/** the value of x for solving differential equation*/
public double target_x;
/** Constructor:
* @param DifferentialEquation differential eqn to be solved.
*/
public ModifiedEuler (DifferentialEquation
        differentialEquation, double h, double target_x){
                super(differentialEquation,h);
                this.target_x = target_x;
                solution = solver();
}

private double solver() {
        DataItem nextData = null;
        counter = 0;
        while( current_x < target_x) {
                counter += 1;
                nextData = nextStep();
        }
        return nextData.y;
}

/** The next result of computing
* return the next data item of the solution approximately.
* @param h  the width of the interval.
*/
public  DataItem nextStep(){
        double prev_x = current_x;  //  previous x
        double prev_y = current_y;  // previous y

current_y +=  h*differentialEquation.Of(current_x,current_y);
        current_x +=  h;
        // modified value of y
current_y = prev_y + h*(differentialEquation.Of(prev_x, prev_y)

+differentialEquation.Of(current_x,current_y))/2.0;
        return new DataItem(current_x, current_y);
}
}
```

รายละเอียดของคลาส RungeKutta สังเกตที่เมธอด nextStep( ) ซึ่งจะถูกเขียนขึ้นมาใหม่ตามระเบียบวิธีของ Runge – Kutta

```
/* File : RungeKutta.java */

package MathTools.DifferentialEq;
import MathTools.Common.*;
import MathTools.DifferentialEq.*;
/** Solving the 1st order differential equation
*    by using fourth-order Runge-Kutta method.
*/

public  class RungeKutta extends  DifferentialEquationSolver {
/** the value of x for solving differential equation*/
 public double target_x;
/** Constructor:
* @param DifferentialEquation  differential eqn to be solved.
*/
public RungeKutta (DifferentialEquation
        differentialEquation, double h, double target_x){
              super(differentialEquation,h);
              this.target_x = target_x;
              solution = solver();
 }
 private double solver() {
        DataItem nextData = new DataItem(0,0);
        counter = 0;
        while( current_x < target_x) {
              counter += 1;
              nextData = nextStep();
        }
        return nextData.y;
 }

 /** The next result of computing
 * return the next data item of the solution approximately.
 * @param h  the width of the interval.
 */
 public  DataItem nextStep(){
 double k1 = h*differentialEquation.Of(current_x, current_y);
 double k2 = h*differentialEquation.Of(current_x+h/2,current_y +
             k1/2);
 double k3 = h*differentialEquation.Of(current_x+h/2,current_y +
             k2/2);
 double k4 = h*differentialEquation.Of(current_x+h,current_y +
             k3);
 current_y = current_y + (k1 + 2*k2 + 2*k3 + k4)/6;
 current_x += h;
 return new DataItem(current_x, current_y);
 }
 }
```

รายละเอียดของคลาส AdamsMoulton มีเมธอด nextStep( ) เป็นการกำหนดค่าประมาณที่เป็นค่าทำนาย และนำตัวแก้ไขมาคำนวณเพื่อให้ได้คำตอบเข้าใกล้ค่าแท้จริงยิ่งขึ้น

```
/* File : AdamsMoulton.java */

package MathTools.DifferentialEq;
import MathTools.Common.*;
import MathTools.DifferentialEq.*;
/** Solving the 1st order differential equation
*     by using predictor-corrector method.
*/
public  class AdamsMoulton extends DifferentialEquationSolver {

/** the value of x for solving differential equation*/
            public double target_x;
/** dirived value */ private double F[] = new double [4];
/** number of interval */ private int interval;


/** Constructor:
* @param DifferentialEquation  differential eqn to be solved.
*/
public AdamsMoulton (DifferentialEquation
      differentialEquation, double h, double target_x){
            super(differentialEquation,h);
            this.target_x = target_x;
            interval = (int)((target_x - initialValue.x)/h );
            if (interval < 4) h = (target_x -initialValue.x)/4;
            solution = solver();
}

private double solver() {
      DataItem nextData = null;
      counter = 0;
      getFirstThreeValues();
      for(int i = 4; i<= interval; i++){
            counter += 1;
            nextData = nextStep();
      }
      return nextData.y;
}

/** Using the Runge-Kutta one step method to solve for
* first three of y and the derivative
* at 3 points are calculated */
private void getFirstThreeValues() {

      F[0] = differentialEquation.Of(current_x, current_y);
            for(int i =1; i < 4; i++) {
      double k1 = differentialEquation.Of(current_x,current_y);
      double k2 = differentialEquation.Of(current_x+h/2,
                  current_y + k1*h/2);
      double k3 = differentialEquation.Of(current_x+h/2,
                  current_y + k2*h/2);
      double k4 = differentialEquation.Of(current_x+h,
                  current_y + h*k3);
      current_y = current_y + (k1 + 2*k2 + 2*k3 + k4)*h/6;
```

```
        current_x += h;
        F[i] = differentialEquation.Of(current_x,current_y);
}
}
/** The next result of computing
* return the next data item of the solution approximately.
* @param h  the width of the interval.
*/
public  DataItem nextStep(){
        double prev_y = current_y;  // previous y
        // Apply predictor
current_y =prev_y + h/24*(55*F[3] - 59*F[2] + 37*F[1]- 9*F[0]);
        current_x = current_x + h;
        F[0] = F[1];
        F[1] = F[2];
        F[2] = F[3];
        F[3] = differentialEquation.Of(current_x, current_y);
        // Apply corrector
current_y = prev_y + h/24*(9*F[3]+ 19*F[2] - 5*F[1] + F[0]);
        return new DataItem(current_x, current_y);
}
}
```

4.5.4 การทดสอบการหาคำตอบของสมการอนุพันธ์แบบสามัญ

__ตัวอย่าง 4.5.1__ จงหาคำตอบของสมการอนุพันธ์ต่อไปนี้ ที่ x = 1

$$\frac{dy}{dt} = 2xe^{2x} + y$$

โดยที่ y(0) = 1

__วิธีทำ__ สร้างคลาสสำหรบทดสอบสมการอนุพันธ์ รายละเอียดของโปรแกรมมีดังนี้

```
/* File : DETestAll.java */
import static java.lang.Math.*;
import MathTools.Common.*;
import MathTools.DifferentialEq.*;
class  DETestAll {
public static void main(String[] args) {
long executedTime,start;
DifferentialEquation equation = new
DifferentialEquation(new DataItem(0d,1.0)) {
        public double Of(double x) { return 0;}
        public double Of(double x, double y)
          {return 2*x*exp(2*x)+y;}
        public double solutionOf(double x) {
                return 3*exp(x) - 2*exp(2*x) + 2*x*exp(2*x);
                }
```

```
16:         };
17: System.out.println("Differential Eqn. : y' = 2xe^2x +y ");
18: System.out.println("                h. = 0.001");
19: System.out.println("---------------------------------");
20:
21:         start = System.nanoTime() ;
22: ModifiedEuler me = new ModifiedEuler( equation, 0.001,1.0);
23:         System.out.println("Loop count : "+ me.getCounter());
24:         System.out.println("by using Modified Euler Method...");
25: System.out.println("\n The solution of this D.E.at x=
                "+me.target_x + " is "+ me.getSolution());
26:         executedTime = System.nanoTime() - start;
27: System.out.println("Executed time = "+executedTime+" ns");
28:         System.out.println("-----------------------------");
29:
30:
31:         start = System.nanoTime() ;
32:         RungeKutta rk = new RungeKutta( equation, 0.001,1.0);
33:         System.out.println("Loop count : "+ rk.getCounter());
34:         System.out.println("by using Runge- Kutta Method...");
35: System.out.println("\nThe solution of this D.E.at x= "+
                rk.target_x + " is "+ rk.getSolution());
36:         executedTime = System.nanoTime() - start;
37: System.out.println("Executed time = "+executedTime+" ns");
38:         System.out.println("------------------------------");
39:
40:
41:         start = System.nanoTime() ;
42:         AdamsMoulton am = new AdamsMoulton( equation, 0.001,1.0);
43:         System.out.println("Loop count : "+ am.getCounter());
44:         System.out.println("by using Adams - Moulton Method...");
45: System.out.println("\n The solution of this D.E.at x= "+
                am.target_x + " is "+ am.getSolution());
46:         executedTime = System.nanoTime() - start;
47: System.out.println("Executed time = "+executedTime+" ns");
48:
49:         }
50: }
51:
```

ใส่ค่าฟังก์ชัน $2xe^{2x} + y$ ลงไปตรงบรรทัดที่ **12** ใส่เงื่อนไขเริ่มต้น (y(0) = 1) ลงในบรรทัดที่ 10 หลังจากให้คลาส ModifiedEuler, **คลาส** RungeKutta, คลาส AdamsMoulton ทำงานผลลัพธ์ที่ได้จากการคำนวณมีดังนี้

```
Differential Eqn. : y' = 2xe^2x +y
                      h. = 0.001
---------------------------------------------------------------------
Loop count : 1000
by using Modified Euler Method...

 The solution of this D.E.at x= 1.0 is 8.154845431632598
Executed time = 5993778 ns
---------------------------------------------------------------------
Loop count : 1000
by using fourth order Runge- Kutta Method...

The solution of this D.E.at x= 1.0 is 8.154845485376939
Executed time = 4263670 ns
---------------------------------------------------------------------
Loop count : 997
by using Adams - Moulton Method...

 The solution of this D.E.at x= 1.0 is 8.1548454853935
Executed time = 2702019 ns
```

คำตอบของสมการอนุพันธ์ในตัวอย่าง 4.5.1 คือ $y = 3e^{x} - 2e^{2x} + 2xe^{2x}$ เมื่อแทนค่า x = 1 จะได้ค่า y = 8.154845

เมื่อพิจารณาผลลัพธ์ที่ได้จากการทำงาน ทุกวิธีจะให้คำตอบใกล้เคียงกับค่าแท้จริง เมื่อมองถึงจำนวนการวนรอบเพื่อหาคำตอบและเวลาที่ใช้ในการคำนวณ ในตัวอย่างนี้วิธีของ Adams-Moulton จะใช้เวลาน้อยที่สุด และการวนรอบเพื่อหาคำตอบดีกว่าวิธีอื่น รองลงมาคือวิธี Runge-Kutta และ ModifiedEuler ตามลำดับ

<u>ตัวอย่าง 4.5.2</u> จงหาคำตอบสมการอนุพันธ์ต่อไปนี้

$$\frac{dy}{dx} = \frac{y}{x} + x - 1$$

กำหนดเงื่อนไขคือ y(1) = 1

<u>วิธีทำ</u>

ใช้โปรแกรมเดียวกับตัวอย่าง 4.5.1 เพียงแต่เปลี่ยนค่าฟังก์ชันให้เป็นดังนี้

```
DifferentialEquation equation = new DifferentialEquation
      (new DataItem(1,1))
      {
      public double Of(double x)
       { return 0}
```

```
public double Of(double x, double y)
 {return y/x+x-1);}
public double solutionOf(double x) {
return 0;
      }
};
```

ผลลัพธ์การทำงานของโปรแกรมจะได้คำตอบดังนี้

```
Differential Eqn. : y' =  y/x + x -1
                          h. = 0.001
-------------------------------------------------------------------
using Modified Euler Method...
 The solution of this D.E.at x= 2.0 is 2.6160129916787485
Executed time = 4989181 ns
-------------------------------------------------------------------
using fourth order Runge- Kutta Method...
The solution of this D.E.at x= 2.0 is 2.6160132417411783
Executed time = 3666108 ns
-------------------------------------------------------------------
using Adams - Moulton Method...
 The solution of this D.E.at x= 2.0 is 2.613705638880162
Executed time = 3196775 ns
```

คำตอบของสมการอนุพันธ์ในตัวอย่างนี้คือ

$$y = x^2 - x\ln(x)$$

เมื่อ x = 2 จะได้ y = 2.6137056

วิธีของ Adams-Moulton จะใช้เวลาน้อยที่สุดและให้คำตอบใกล้เคียงค่าแท้จริงมากที่สุด รองลงมาคือวิธี Runge- Kutta และ ModifiedEuler ตามลำดับ

4.6 การคำนวณค่าสถิติเบื้องต้น

ข้อมูลทางวิทยาศาสตร์ส่วนใหญ่จะเป็นตัวเลขหรือเชิงปริมาณ เช่น ใช้ไมโครมิเตอร์วัดเส้นผ่านศูนย์กลางของเส้นลวดเส้นหนึ่งหลาย ๆ ครั้ง หรือวัดค่าความเร่งเนื่องจากแรงโน้มถ่วงของโลก (g) โดยให้วัตถุตกจากตำแหน่งที่สูงจากพื้นต่าง ๆ กัน แล้วจับเวลาการเคลื่อนที่ จะได้เส้นผ่านศูนย์กลางของเส้นลวด หรือค่า g หลาย ๆ ค่า จากนั้นนำข้อมูลที่รวบรวมได้เหล่านี้มาสรุปผลการทดลอง โดยใช้สถิติเชิงพรรณนา (Descriptive Statistic) หาความถี่ ค่าเฉลี่ย ค่ามัธยฐาน ค่าฐานนิยม พิสัย ส่วนเบี่ยงเบนมาตรฐาน

ในคลาสไลบรารีชื่อ SimpleStat ประกอบด้วยเมธอดที่ทำหน้าที่ต่อไปนี้

4.6.1 **การแจกแจงความถี่** (Frequency) เป็นการหาความถี่และความถี่สะสมของข้อมูล โดยจะเรียงลำดับข้อมูลจากค่ามากไปหาค่าน้อย เมธอดที่ทำหน้าที่นี้คือ findFrequency() และ findCumulativeFrequency() การเรียงลำดับข้อมูลจะใช้ระเบียบวิธี Quick sort

4.6.2 **การวัดแนวโน้มเข้าสู่ส่วนกลาง** (Central Tendency) เป็นการหาค่ากลางของข้อมูลเพื่อใช้เป็นตัวแทนของข้อมูลทั้งหมด เป็นการหาแบบพิจารณาข้อมูลทั้งหมด (ไม่แบ่งข้อมูลเป็นช่วงชั้น)

- ค่าเฉลี่ยหรือค่ามัชฌิมเลขคณิต ($\bar{x}$) (Mean)

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

เมื่อ n คือจำนวนข้อมูลทั้งหมด

x_i คือข้อมูลตัวที่ i เมื่อ i = 1, 2, ... n

เมธอดที่ทำหน้าที่หาค่าเฉลี่ยคือ findMean( )

- ค่ามัธยฐาน (Median) คือเมื่อนำข้อมูลมาเรียงลำดับ (จากค่าน้อยไปสู่ค่ามากหรือจากค่ามากไปน้อยก็ได้) ค่าของข้อมูลที่อยู่ตรงกลางของข้อมูลทั้งหมดคือค่ามัธยฐาน ถ้าข้อมูลทั้งหมดมีเป็นจำนวนคู่ ค่าที่อยู่ตรงกลางจะมี 2 ค่า ต้องหาค่าเฉลี่ยของค่าทั้งสองเสียก่อน เมธอดที่ทำหน้าที่หาค่ามัธยฐานคือ findMedian( )

- ค่าฐานนิยม (Mode) คือค่าของข้อมูลที่มีความถี่มากที่สุดหรือปรากฏซ้ำกันมากที่สุดในข้อมูลชุดนั้น ๆ ข้อมูลบางชุดอาจมีค่าฐานนิยมมากกว่า 1 ค่าก็ได้ เมธอด findMode() ใช้หาค่าฐานนิยม ซึ่งออกแบบไว้ให้มีการตรวจสอบในกรณีที่ฐานนิยมมีลักษณะเป็น multimodal ด้วย

4.6.3 **การวัดการกระจายของข้อมูล** (Measure of variation) เป็นการอธิบายว่าข้อมูลแต่ละค่านั้นมีค่าห่างกันมากน้อยเพียงใด

- ค่าพิสัย (Range) คือค่าผลต่างของข้อมูลตัวที่มากที่สุด กับตัวที่มีค่าน้อยที่สุด เมธอด findRange() ทำหน้าที่หาค่าพิสัยนี้

- ค่าส่วนเบี่ยงเบนมาตรฐาน (Standard deviation, S.D.) คือค่ารากที่สองของผลรวมของความแตกต่างระหว่างข้อมูลกับค่าเฉลี่ยของข้อมูลชุดนั้น หารด้วยจำนวนข้อมูลทั้งหมด ในที่นี้จะใช้สูตรหาค่าส่วนเบี่ยงเบนมาตรฐานจากกลุ่มตัวอย่าง โดยใช้ข้อมูลดิบทั้งหมด (ไม่ได้ทำเป็นตารางแจกแจงความถี่เป็นช่วงชั้น)

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

s คือ ค่าส่วนเบี่ยงเบนมาตรฐาน

n คือจำนวนข้อมูลทั้งหมด

x_i คือข้อมูลตัวที่ i เมื่อ i = 1, 2, ... n

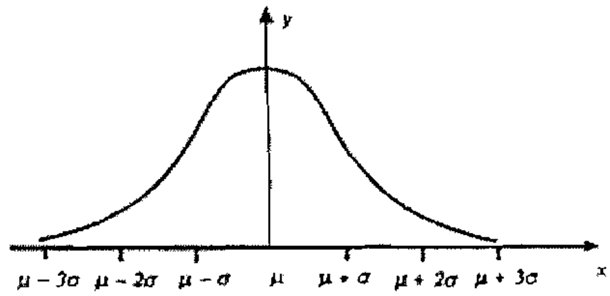
เมธอด findSD() จะใช้ในการหาค่าส่วนเบี่ยงเบนมาตรฐาน

4.6.4 **การกระจายแบบโค้งปกติ** (Normal Distribution or Gaussian Distribution)

หัวข้อนี้จะก้าวล้ำไปสู่สถิติอนุมานหรือสถิติอ้างอิง (Inference statistics) เป็นการนำข้อมูลที่เก็บได้จากกลุ่มตัวอย่าง (sample) ไปใช้อธิบายหรืออ้างอิงถึงกลุ่มประชากร(population) ทั้งหมด โดยใช้หลักความน่าจะเป็น (probability) ตรวจสอบสมมติฐาน

ข้อมูลทางวิทยาศาสตร์ที่เก็บได้จากกลุ่มตัวอย่างที่มีจำนวนมาก เช่นวัดความสูงหรือน้ำหนักของคนโดยสุ่มจากกลุ่มคนจำนวนมาก ๆ การกระจายของความสูงและน้ำหนักจะมีลักษณะเป็นรูประฆังคว่ำหรือโค้งปกติ ชิ้นงานที่ถูกออกแบบให้มีขนาดคงที่ค่าหนึ่ง เมื่อผลิตจากเครื่องจักรพบว่าเกิดความคลาดเคลื่อนไปจากที่แบบที่ออกไว้ ความคลาดเคลื่อนขนาดชิ้นส่วนเมื่อวัดแบบสุ่มจะมีลักษณะการกระจายเป็นแบบโค้งปกติ เช่นกัน

ลักษณะของโค้งปกติ จะมีลักษณะดังรูป



รูป 4.9 รูปโค้งปกติ

เมื่อสุ่มตัวอย่างจากประชากรใด ๆ ซึ่งมีค่าเฉลี่ยเป็น μ และความเบี่ยงเบนมาตรฐานเป็น σ สามารถประมาณการกระจายได้ด้วยการกระจายแบบปกติ โดยกลุ่มที่สุ่มตัวอย่างมีขนาดใหญ่ เราสามารถเขียนเป็นฟังก์ชันที่บอกถึงความน่าจะเป็นของตัวแปรสุ่ม (x) ในรูปของ Probability Density Function ได้ดังนี้

$$y \quad = \quad \frac{1}{\sqrt{2\pi}} e^{\left[\frac{x-\mu}{\sigma}\right]^2} \qquad \text{................. (4.36)}$$

x จะมีค่าอยู่ระหว่าง $-\infty$ และ $+\infty$ จากรูปจะเห็นว่าเส้นโค้งจะมีลักษณะสมมาตรที่จุด $x = \mu$ และจะลดค่าอย่างรวดเร็วเมื่อ x เพิ่มหรือลดค่าไปด้านซ้ายมือหรือขวามือ

การกระจายแบบโค้งปกติที่มีค่าเฉลี่ย μ ความแปรปรวนเป็น σ^2 สามารถเขียนให้อยู่ในรูปสั้น ๆ ได้ดังนี้

$$x \quad \sim \quad N(\mu, \sigma) \qquad \text{................ (4.37)}$$

เมื่อกำหนดให้ $z = (x-\mu)/\sigma$ แล้วนำไปแทนค่าลงในสมการ (1) เพื่อให้อยู่ในรูปมาตรฐาน (standard normal distribution) นั่นคือ ค่าเฉลี่ยจะมีค่าเป็นศูนย์ และความแปรปรวนมีค่าเท่ากับ 1

$$x \quad \sim \quad N(0, 1) \qquad \text{............... (4.38)}$$

สมการ (4.36) จะเปลี่ยนรูปเป็น

$$y \quad = \quad \frac{1}{\sqrt{2\pi}} e^{-z^2/2} \qquad \text{............. (4.39)}$$

พื้นที่ใต้เส้นโค้งของสมการ (4.39) จะมีค่าเท่ากับ 1 หรือความน่าจะเป็นของ z ในช่วง $-\infty$ ถึง ∞ มีค่าเป็น 100% ความน่าจะเป็นของ z ในช่วง a ถึง b คือพื้นที่ใต้เส้นโค้งซึ่งอยู่ระหว่าง a และ b นั้น และจะเห็นว่า y มีค่าเป็นบวกเสมอ

การหาความน่าจะเป็นเมื่อกำหนดค่า z ในช่วงใด ๆ สามารถทำได้จากการเปิดตาราง ค่าสถิติ z จากหนังสือสถิติทั่วไป หรืออาจเขียนโปรแกรมให้คอมพิวเตอร์หาพื้นที่ใต้เส้นโค้งในช่วง z_1 ถึง z_2 ได้ดังนี้

$$\text{พื้นที่ใต้เส้นโค้ง} \quad = \quad \frac{1}{\sqrt{2\pi}} \int_{z_1}^{z_2} e^{-z^2/2} dz$$

ความน่าจะเป็นของตัวแปรสุ่มที่อยู่ระหว่างช่วง ($\mu - \sigma$, $\mu + \sigma$) มีค่าประมาณ 68% ความน่าจะเป็นระหว่าง (μ - 2σ, μ + 2σ) ประมาณ 95% และอยู่ระหว่าง (μ + 3σ) ประมาณ 99.7%

เมธอด findZ() ใช้หาค่า z โดยคำนวณจากสูตร $z = (x-\mu)/\sigma$

เมธอด findAreaUnderNormalCurve(z1, z2) ใช้หาพื้นที่ใต้เส้นโค้ง(หรือความน่าจะเป็น)ที่อยู่ระหว่างค่า z ที่กำหนดมาให้ การหาพื้นที่ใต้เส้นโค้งอาศัยหลักการการหาปริพันธ์โดยวิธีของซิมพ์สัน 1/3 โดยกำหนดช่วงที่จะหาปริพันธ์ไว้ =1000 อาจจะกำหนดช่วงให้มากกว่านี้ เพื่อที่จะได้ค่าพื้นที่ใต้โค้งใกล้เคียงกับค่าแท้จริง แต่จะใช้เวลาในการประมวลผลมากขึ้น

ในทางกลับกัน เมธอด findZAtKnownArea (area) จะทำหน้าที่หาค่า z เมื่อกำหนดความน่าจะเป็นหรือพื้นที่ใต้เส้นโค้งมาให้ เมื่อกำหนดพื้นที่ใต้โค้งมาให้ โดยเริ่มจากด้านซ้ายมือ (z = $-\infty$) จนถึงตำแหน่งที่ต้องการหา z ในการออกแบบจะให้โปรแกรมคำนวณโดยเริ่มจากตรวจสอบดูว่าพื้นที่ให้มานั้นมีค่ามากกว่า 0.5 หรือไม่ ถ้ามีค่ามากกว่า 0.5 จะนำค่าพื้นที่นั้นเป็นตัวตั้ง นำ 0.5 มาหักออก พื้นที่ที่เหลือจะเป็นพื้นที่ที่เริ่มจาก z = 0 ถึง z ที่ต้องการจะหาค่า แต่ถ้าพื้นที่น้อยกว่า 0.5 จะนำพื้นที่ไปหักออกจาก 0.5 นั้น ผลลัพธ์จะเป็นพื้นที่ตั้งแต่ z = 0 ถึง z ที่จะหาค่าเช่นกัน แต่ z ที่ได้ในกรณีนี้จะเป็นค่าลบ การค้นหาค่า z ที่ทำให้ได้พื้นที่เท่ากับค่าที่กำหนดทำได้โดยเพิ่มค่า z ครั้งละ 0.01 โดยเริ่มจาก z เท่ากับศูนย์ คำนวณหาพื้นที่เพิ่มขึ้นมานี้ รวมพื้นที่ที่เพิ่มขึ้นมากับพื้นที่เดิม เปรียบเทียบกับพื้นที่ที่กำหนดให้ ถ้ายังน้อยกว่าให้เพิ่มค่า z (ครั้งละ 0.01) ไปเรื่อย ๆ จนได้ค่าพื้นที่เท่ากับพื้นที่ที่กำหนดให้ z ตรวจจุดนี้คือ ค่า z ที่ต้องการ

ต่อไปนี้เป็นรายละเอียดของคลาส SimpleStat ที่ประกอบด้วยเมธอดตามที่กล่าวมาข้างต้น

```
/* File : Simplestat.java
* Project: Math Tools in Java
* Purpose: Compute basic statistics
*      frequency, central tendency, measure of variation.
*      normal curve distribution.
* Author : Wachara R.
* First Released: Th 3 May 2007
* Last Updated : Mon 10 May 2007
*/

package MathTools.SimpleStat;

import static java.lang.Math.*;
import static java.lang.Double.*;
import MathTools.SimpleStat.SimpleStatException;
import static MathTools.Common.CommonConstants.*;
import MathTools.Common.Function;
import MathTools.Integration.*;

/** class for basic statistics algorithm:
*       frequency, measure of variation.
*      some inference statistics*/

public  class SimpleStat {

/** data for computing the basic statistics */ double[ ] data;
/** number of data */  int  numData;
/** number of data after finding frequency */
      int numDataAtLast;
/** basic statistic quantities */
/** mean value */  double mean;
/** median value */ double median;
/** range of data */ double range;
/** standard deviation */ double sd;
/** managed data */  double [] mData;
/** keeping repeated mode data */ double[] modeData;
/** mode of data */ int mode;
/** frequency of data set */ int [] frequency;
/** cumulated frequency */ int [] cFrequency;
/** checking data have been sorted */ boolean isSorted = false;
/** checking data have been classified with frequencies */
                              boolean isClassified = false;
/** Constructor
* @param dat data for computing basic statistics
*/
public SimpleStat( double [ ] dat ){
      data = dat;
      numData = dat.length;
      findMean();
            findSD();
      quickSort( data, 0,numData-1,false);
      findMedian();
       findFrequency();
      findCumulativeFrequency();
      findMode();
```

```
        findRange();
 }
 /** copy data from current array to another array */
 private double[ ] copyData(double[] itemToBeCopied,int nItem) {
        double[ ] dummy  = new double[ nItem];
        for(int i = 0 ; i < nItem; i++)
              dummy[i] = itemToBeCopied[i];
        return dummy;
        }

        private int[ ] copyData(int[] itemToBeCopied,int nItem) {
        int[ ] dummy  = new int[ nItem];
        for(int i = 0 ; i < nItem; i++)
              dummy[i] = itemToBeCopied[i];
        return dummy;
        }


        // ========= Frequency ==========
        /** finding frequency of each data */
        private void  findFrequency(  ){
        int n_freq = 0;
        int tempfrequency[ ]  = new int[numData];
        double temp  [ ]  = new double[numData];
        double dummy[] = new double [numData];
 if (isSorted == false)  quickSort( data, 0,numData-1,false);
        temp = copyData(data, numData);
        for (int i = 0; i < numData;i++){
         if (temp[i] != MAX_VALUE) {
              tempfrequency[n_freq]=1;
              dummy[n_freq] = temp[i];
              int j = i;
              while ( (j < numData-1) && (temp[j+1] == temp[i])){
                          tempfrequency[n_freq] +=1;
                                temp[j+1] = MAX_VALUE;
                                j = j+1;
              }

              n_freq +=1;
   }

        }
        mData= new double[n_freq];
        mData = copyData ( dummy, n_freq);
        frequency = new int[n_freq];
        frequency = copyData(tempfrequency, n_freq);
        isClassified = true;
        numDataAtLast = n_freq;
 }
 /** finding cumulative frequency */
 private void findCumulativeFrequency(){
        // if data are not sorted, sort them decendingly.
 if (isSorted == false)  quickSort( data, 0,numData-1,true);
        cFrequency = new int[numDataAtLast];
        cFrequency[ numDataAtLast-1] =frequency[numDataAtLast-1];
              for(int i = numDataAtLast -2; i >=0; i--){
              cFrequency[i] = frequency[i] + cFrequency[i+1];
              }

        }
  /** quick sort: sort  all data in ascending or decending
```

```
 * param item  data to be sorted
 * param left  the first item of data
 * param right the last item of data
 * param ascending if true,data will be sort in ascending
 */
void quickSort(double[]item,int left,int right,
        Boolean ascending){
        int i,j;
        double comparand, temp;
             i = left;
             j = right;
             comparand = item[(left+right)/2];
             do { if (ascending) {
                   while( item[i] < comparand && i < right) i++;
                   while (comparand < item[j] && j > left) j--;
                   }else {
                   while( item[i] > comparand && i < right) i++;
                   while (comparand > item[j] && j > left) j--;
                   }
                   if ( i <= j) {
                         temp = item[i];
                         item[i] = item[j];
                         item[j] = temp;
                         i++;
                         j--;
                   }
       }while ( i <= j );
             if (left < j) quickSort(item, left, j, ascending);
             if(i < right ) quickSort(item, i,right, ascending);
                   isSorted = true;
             }
       // ============ Central tendecy =============
 /** find mean or average of data */
 private void findMean( ) {
       double sumData=0;
       for(int i = 0 ; i < numData; i++ )
             sumData +=data[i];

       mean = sumData/(double)numData;
 }
 /** find median  of data */
 private void findMedian( ) {
       if (isSorted == false)quickSort(data, 0,numData-1,true);
       if (numData%2 == 0)
             // number of data is even, find the average
             median = (data[numData/2] + data[(numData/2)-
                   1])/2.0 ;
       else
             median = data[numData/2];
       }

 /** find mode  of data */
 private void findMode( ) {
       int multiMode =1;

       if (isClassified == false)    findFrequency( );
       if(numData == numDataAtLast)  {
             modeData = new double[1];
             modeData[0] =0;
             mode = 0;
             return;
```

```
        }
// walk through classified data looking for multimode
              for (int i = 0; i < numDataAtLast; i++) {
              if(frequency[i] >= mode) {
                    if (frequency[i] == mode)
                         multiMode  +=1;
                    else {
                    mode = frequency[i];
                    multiMode = 1;
                    }
              }
        }
        if (multiMode ==1 ) {
              modeData = new double[1];
        } else {
                    modeData = new double[multiMode];
      }
        int j =0;
        for (int i = 0 ; i < numDataAtLast ; i++){
                    if (frequency[i] == mode) {
                    modeData[j] = mData[i];
                    j = j+1;
              }
        }
        }


        // ===========Measure of variation =====
        /** find range of data */
        private void findRange(){
              if (isClassified == false)findFrequency( );
              range = data[0] - data[data.length-1];
        }

   /** find standard deviation of data */
   private void findSD() {
         double sumDifference=0;
         for(int i = 0; i < numData; i++){
               sumDifference += (data[i] - mean)*(data[i]-mean);
         }
         sd = sqrt(sumDifference/(double)(numData-1));
   }


  // ========== Some parameter inference statistics ==========

        /** Compute the area between 2 statistic z value
        * @param lower_z lower limit of area
        * @param upper_z upper limit of area
        * @ return area under normal curve between 2 value of z
        */

  public double findAreaUnderNormalCurve(double lower_z,
         double upper_z) {

        Function normalFunction = new Function() {
  public double Of(double x) {
                    return (1/sqrt(2.0*PI)*exp(-(x*x)/2.0));}
  };
   Simpson1_3Integration si = new Simpson1_3Integration
         (normalFunction,lower_z, upper_z,1000);
```

```
            return (si.getIntegrationResult());
}

/** finding statistic Z of any data x
* @param x   any data x
* @return z value statistic Z
*/
public double findZ(double x){
      return ( x - mean)/sd;
}

/** finding statistic Z value at known area under normal curve
* @param area the area under normal curve should be between 0 -
1
* @return z value at known area
*/
public double findZAtKnownArea (double area) throws
         SimpleStatException {
      boolean isAreaGreaterThanHalf = false;
      double lowerLimit=0;
      double newArea=0;
      double z=0 , sumArea=0;
      double dA;
 if ( area < 0 || area >1 ) throw
      new SimpleStatException(SimpleStatException.BAD_AREA);
            if (area > 0.5){
                   newArea = area - 0.5;
                   isAreaGreaterThanHalf = true;
             } else {
                   newArea = 0.5 - area;
            }
            if (newArea >= 0.3412944) {
                  lowerLimit = 1;
                  sumArea = 0.3412944;
            }
            if (newArea >= 0.4771599) {
                  lowerLimit = 2;
                  sumArea = 0.4771599;
            }
            if (newArea >= 0.4986253) {
                  lowerLimit = 3;
                  sumArea = 0.4986253;
            }
            z = lowerLimit;

            while( newArea - sumArea >0.001) {
            dA = (1/sqrt(2*PI))*0.5*0.01*
      (exp(-0.5*z*z)+exp(-0.5*(z+0.01)*(z+0.01)));
                  sumArea += dA;
                  z += 0.01;
            }
            if (isAreaGreaterThanHalf)  return z;
            else return -z;
      }

      // ========= Getter =============
public double getMean(){ return mean;}
public double getMedian() { return median;}
public double getStandardDeviation( ){ return sd;}
public double getRange() { return range;}
public int getFrequencyOfMode() { return mode;}
```

```
public double[] getModeData(){ return modeData;}
public double[] getSortedData( ) { return data;}
public double[]getManagedData() { return mData;}
public int[ ] getFrequency() { return frequency;}
public int[] getCumulativeFrequency() { return cFrequency;};
}

```

4.6.5 การทดสอบการหาค่าสถิติเบื้องต้น

ได้นำคลาสไลบรารีชื่อ SimpleStat มาทดสอบหาค่าสถิติ ดังตัวอย่างต่อไปนี้

<u>ตัวอย่าง 4.6.1</u> จากการวัดเส้นผ่านศูนย์กลางของลูกปืน(Ball bearing) ที่ผลิตจากเครื่องจักร จำนวน 110 ลูก วัดได้ละเอียดถึงทศนิยมตำแหน่งที่ 3 ในหน่วยมิลลิเมตร ต้องการหาความถี่, ค่าเฉลี่ย และความเบี่ยงเบนมาตรฐานของข้อมูลที่ได้

3.510 3.515 3.519 3.517 3.517 3.518 3.522 3.521 3.520 3.524

3.523 3.522 3.522 3.519 3.526 3.521 3.526 3.522 3.523 3.520

3.522 3.509 3.524 3.525 3.510 3.515 3.521 3.522 3.519 3.516

3.521 3.522 3.519 3.520 3.522 3.518 3.519 3.519 3.521 3.515

3.516 3.518 3.521 3.521 3.519 3.517 3.523 3.525 3.522 3.522

3.521 3.520 3.516 3.522 3.521 3.525 3.520 3.521 3.520 3.524

3.523 3.520 3.519 3.525 3.525 3.527 3.528 3.519 3.525 3.520

3.517 3.516 3.518 3.520 3.520 3.516 3.521 3.520 3.519 3.521

3.517 3.523 3.524 3.517 3.518 3.518 3.520 3.520 3.521 3.524

3.520 3.523 3.519 3.518 3.517 3.517 3.524 3.523 3.519 3.518

3.523 3.524 3.523 3.524 3.519 3.518 3.519 3.518 3.520 3.520

<u>วิธีทำ</u>

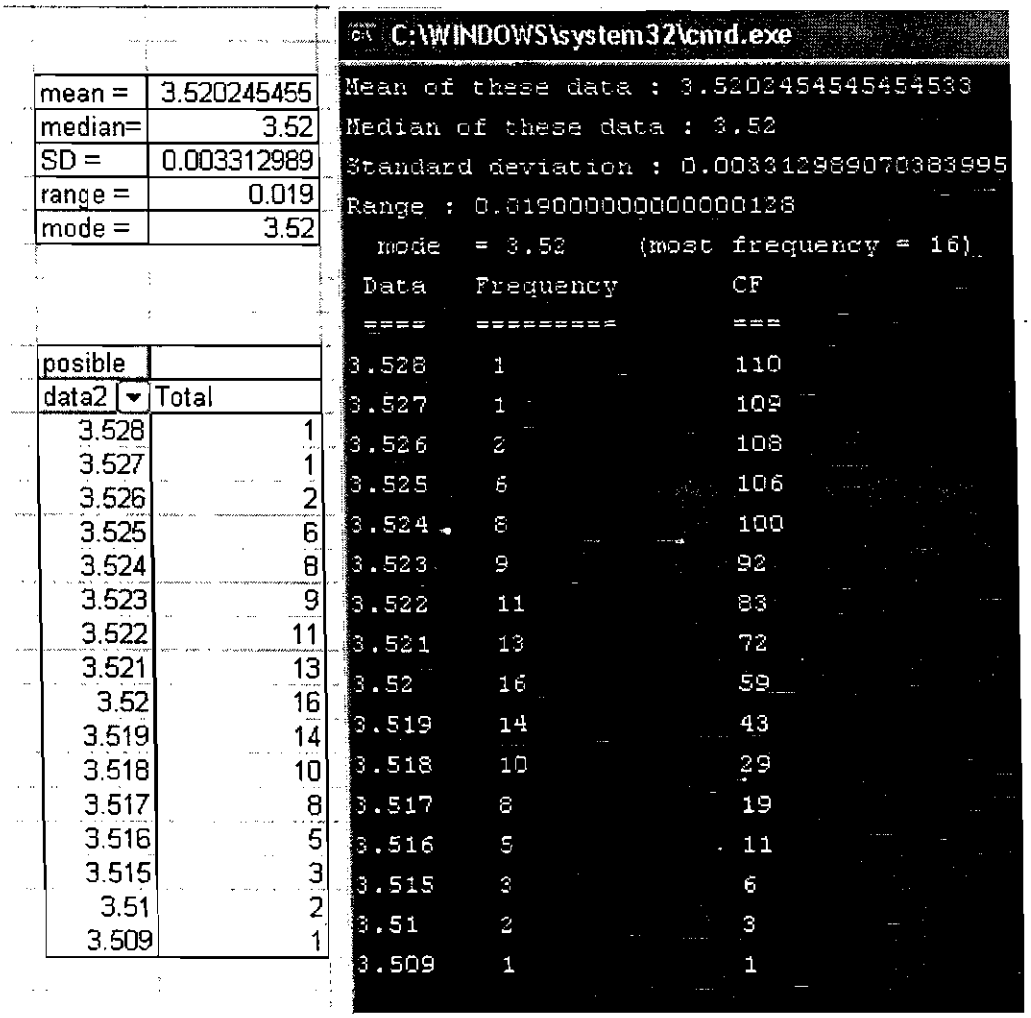
ขั้นตอนการทดสอบทำได้ดังนี้

1. นำข้อมูลทั้งหมดจัดเก็บไว้ใน text file ชื่อว่า ball.txt
2. สร้าง application โปรแกรมสำหรับทดสอบค่าสถิติ เรียกใช้คลาส SimpleStat เพื่อคำนวณ รายละเอียดของโปรแกรมมีดังนี้

```
/* File :StatTest.java
* Project: Math Tools in Java
* Purpose: Test class for finding simple statistic formula
*           and Reading data from text file..
* Author : Wachara R.
* First Released: Sun 6 May 2007
* Last Updated :  Wed 9 May 2007
*/

import static java.lang.Math.*;
import java.io.*;
import java.util.*;
import MathTools.Common.*;
import MathTools.SimpleStat.SimpleStat;

class  StatTest {

      public static void main(String[] args) {
      // this means you must know the number of data
      double data[ ] = new double[110];
      double dataAtMode[] = new double[110];



      // read data from text file
      try{
      File file = new File ("ball.txt");

      FileReader filereader = new FileReader(file);

BufferedReader reader = new BufferedReader(filereader);
            String line = null;
            StringTokenizer  token;
            int i=0;
            while ((line = reader.readLine()) != null){
                  token = new StringTokenizer(line);
                  while(token.hasMoreTokens()){
data[i] = Double.parseDouble(token.nextToken());
                              i++;
                        }
            }
            reader.close();
      } catch(EOFException err) {
            System.out.println("end of stream");
      } catch(IOException err) {
            System.out.println(err.getMessage());
      }
      catch(NumberFormatException err) {
            System.out.println(err.getMessage());
      }

      double  classifiedData[ ] = new double[data.length];

      SimpleStat st = new SimpleStat(data);
      System.out.println("Mean  : "+ st.getMean());
      System.out.println("Median : "+ st.getMedian());
System.out.println("Standard deviation :"+
      st.getStandardDeviation());
      System.out.println("Range : "+ st.getRange());
```

```
        System.out.print("mode   = ");
        dataAtMode = st.getModeData();
        for(int i = 0; i < dataAtMode.length; i++){
              System.out.print (  dataAtMode[i]+ "     " );
        }
System.out.print(" (most frequency = " +
        st.getFrequencyOfMode()+") ");
        System.out.println();
        classifiedData = st.getManagedData();
        int []freq = new int[data.length];
        int[] cf = new int[data.length];
        freq= st.getFrequency();
        cf   = st.getCumulativeFrequency();

        System.out.println(" Data\tFrequency\tCF " );
        System.out.println(" ====\t=========\t=== " );

        for(int i = 0; i < classifiedData.length; i++){
        System.out.println ( classifiedData[i] + "\t " +
         freq[i]+"\t\t"+ cf[i]);
        }
        }
}
```

3. คอมไพล์และให้โปรแกรมทำงาน ผลการทดสอบจะได้ผลลัพธ์ดังภาพ เปรียบเทียบผลลัพธ์ที่คำนวณได้กับผลจากการทดสอบจากโปรแกรมสำเร็จรูป Excel โดยใช้ข้อมูลชุดเดียวกันนี้ จะเห็นว่าได้ผลลัพธ์ที่ไม่ต่างกัน (ในหน้าต่างสีดำ คือผลที่ได้จากโปรแกรม StatTest.java ส่วนที่เป็นสีขาวคือผลที่ได้จากการคำนวณของ Excel)



mean =	3.520245455
median=	3.52
SD =	0.003312989
range =	0.019
mode =	3.52

posible	
data2	Total
3.528	1
3.527	1
3.526	2
3.525	6
3.524	8
3.523	9
3.522	11
3.521	13
3.52	16
3.519	14
3.518	10
3.517	8
3.516	5
3.515	3
3.51	2
3.509	1

```
C:\WINDOWS\system32\cmd.exe
Mean of these data : 3.5202454545454533
Median of these data : 3.52
Standard deviation : 0.0033129890703839995
Range : 0.019000000000000128
  mode  = 3.52     (most frequency = 16)
 Data   Frequency        CF
 ====   =========        ===
3.528    1               110
3.527    1               109
3.526    2               108
3.525    6               106
3.524    8               100
3.523    9               92
3.522    11              83
3.521    13              72
3.52     16              59
3.519    14              43
3.518    10              29
3.517    8               19
3.516    5               11
3.515    3               6
3.51     2               3
3.509    1               1
```

รูป 4.10 ผลลัพธ์ที่ได้จากโปรแกรม StatTest.java

<u>ตัวอย่าง 4.6.2</u> ทดสอบการคำนวณหาพื้นที่ใต้เส้นโค้งปกติเมื่อกำหนดค่า z มาให้ต่าง ๆ กันเพื่อเปรียบเทียบกับตารางความน่าจะเป็นสะสมของการแจกแจงปกติมาตรฐาน(z) ในหนังสือสถิติว่าได้ค่าต่างกันหรือไม่อย่างไร (คิดที่ทศนิยม 4 ตำแหน่ง)

<u>วิธีทำ</u> เพื่อไม่ให้ล้นหน้ากระดาษ จะแสดงผลการคำนวณหาค่า z หาพื้นที่ใต้โค้งเป็น 2 ตาราง คือ ตารางที่ z มีค่าเป็นบวก (ตั้งแต่ 0 ถึง 1.99) และและตารางพื้นที่ใต้โค้งที่ z มีค่าเป็นลบ (ตั้งแต่ -1.99 ถึง 0) โดยอาศัยคลาส Ztable ซึ่งมีรายละเอียดของโปรแกรมดังนี้

```
/* File :Ztable.java
* Project: Math Tools in Java
* Purpose: Test class for finding Z table
* Author : Wachara R.
* First Released: Th 10 May 2007
* Last Updated :
*/

import static java.lang.Math.*;
import java.io.*;
import java.util.*;
import MathTools.Common.*;
import MathTools.SimpleStat.SimpleStat;

public class Ztable {

public static void main(String[] args) {
double z,area, step;
double tempz;
        SimpleStat s = new SimpleStat();
              step = 0.01;
        System.out.println("\n\t  Find the Area under Normal
        curve  between 0 to 1.99 step 0.01.\n\n");
        System.out.println("        0       1       2       3
    4       5       6       7       8       9");
        i = 0;
        area = 0;
        for( z = 0; z<= 2; z = z+step) {
              if ( i % 10 ==0)  {
                    System.out.printf("\n%2.1f   ",z);
                     i =0;
              }
              i +=1;
              area = s.findAreaUnderNormalCurve(0,z);
              System.out.printf(" %4.4f",area+0.5);
        }
        System.out.println("\n\t  Find the Area under Normal
        curve between 0 to -1.99 step 0.01.\n\n");
        System.out.println("        0       1       2       3
    4       5       6       7       8       9");
        area = 0;
        step = 0.1;
        for( z = -1.9; z <= 0; z = z+step) {
              System.out.printf("\n%2.1f   ",z);
               tempz = z;
              for(int k = 0; k < 10 ; k++) {
                    area = s.findAreaUnderNormalCurve(z,0);
                    System.out.printf(" %4.4f", 0.5 - area);
                    tempz = tempz + 0.01;
              }

        }

 }
 }
```

ผลการทำงานของโปรแกรมจะเป็นดังนี้ ตารางแสดงพื้นที่ใต้เส้นโค้งปกติของค่า Z ตั้งแต่ 0 ถึง 1.99 เพิ่มค่าครั้งละ 0.01

	0	1	2	3	4	5	6	7	8	9
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767

ตารางต่อไปนี้แสดงพื้นที่ใต้เส้นโค้งปกติ ตั้งแต่ -1.99 ถึง 0

	0	1	2	3	4	5	6	7	8	9
-1.9	0.0287	0.0281	0.0274	0.0268	0.0262	0.0256	0.0250	0.0244	0.0239	0.0233
-1.8	0.0359	0.0351	0.0344	0.0336	0.0329	0.0322	0.0314	0.0307	0.0301	0.0294
-1.7	0.0446	0.0436	0.0427	0.0418	0.0409	0.0401	0.0392	0.0384	0.0375	0.0367
-1.6	0.0548	0.0537	0.0526	0.0516	0.0505	0.0495	0.0485	0.0475	0.0465	0.0455
-1.5	0.0668	0.0655	0.0643	0.0630	0.0618	0.0606	0.0594	0.0582	0.0571	0.0559
-1.4	0.0808	0.0793	0.0778	0.0764	0.0749	0.0735	0.0721	0.0708	0.0694	0.0681
-1.3	0.0968	0.0951	0.0934	0.0918	0.0901	0.0885	0.0869	0.0853	0.0838	0.0823
-1.2	0.1151	0.1131	0.1112	0.1093	0.1075	0.1056	0.1038	0.1020	0.1003	0.0985
-1.1	0.1357	0.1335	0.1314	0.1292	0.1271	0.1251	0.1230	0.1210	0.1190	0.1170
-1.0	0.1587	0.1562	0.1539	0.1515	0.1492	0.1469	0.1446	0.1423	0.1401	0.1379
-0.9	0.1841	0.1814	0.1788	0.1762	0.1736	0.1711	0.1685	0.1660	0.1635	0.1611
-0.8	0.2119	0.2090	0.2061	0.2033	0.2005	0.1977	0.1949	0.1922	0.1894	0.1867
-0.7	0.2420	0.2389	0.2358	0.2327	0.2296	0.2266	0.2236	0.2206	0.2177	0.2148
-0.6	0.2743	0.2709	0.2676	0.2643	0.2611	0.2578	0.2546	0.2514	0.2483	0.2451
-0.5	0.3085	0.3050	0.3015	0.2981	0.2946	0.2912	0.2877	0.2843	0.2810	0.2776
-0.4	0.3446	0.3409	0.3372	0.3336	0.3300	0.3264	0.3228	0.3192	0.3156	0.3121
-0.3	0.3821	0.3783	0.3745	0.3707	0.3669	0.3632	0.3594	0.3557	0.3520	0.3483
-0.2	0.4207	0.4168	0.4129	0.4090	0.4052	0.4013	0.3974	0.3936	0.3897	0.3859
-0.1	0.4602	0.4562	0.4522	0.4483	0.4443	0.4404	0.4364	0.4325	0.4286	0.4247

เมื่อเปรียบเทียบกับตารางหาค่าสถิติ z จากหนังสือ การวิจัยและวิเคราะห์ข้อมูลทางสถิติด้วย SPSS. [3] พบว่าได้ตรงกันทุกค่า (คิดที่ทศนิยม 4 ตำแหน่ง)

<u>ตัวอย่าง 4.6.3</u> โดยใช้ข้อมูลจากตัวอย่าง 4.6.1 จะเห็นว่าค่าเฉลี่ยของเส้นผ่านศูนย์กลางของลูกปืนคือ 3.520 มิลลิเมตร ส่วนเบี่ยงเบนมาตรฐานเท่ากับ 0.003 มิลลิเมตร สุ่มตัวอย่างการวัดพบว่ามีลักษณะการกระจายของข้อมูลเป็นแบบโค้งปกติ

a. จงหาความน่าจะเป็นที่จะพบลูกปืนที่มีเส้นผ่านศูนย์กลางระหว่าง 3.518 ถึง 3.523 ม.ม.

b. ลูกปืนที่มีเส้นผ่านศูนย์กลางน้อยกว่า 3.515 ม.ม. และมากกว่า 3.525 ม.ม. จะต้องถูกคัดทิ้ง จำนวนลูกปืนที่มีโอกาสจะถูกคัดทิ้งมีจำนวนกี่เปอร์เซ็นต์

<u>วิธีทำ</u>

แสดงวิธีทำโดยการหาค่าจากการเปิดตารางเพื่อเปรียบเทียบกับค่าที่ได้จากโปรแกรม

a. หาความน่าจะเป็นที่จะพบลูกปืนที่มีเส้นผ่านศูนย์กลางระหว่าง 3.518 ถึง 3.523 ม.ม. ทำได้โดยหาค่า z ของข้อมูล 3.518 และ 3.523 จะได้ค่า z ดังนี้

$$z1 = \frac{3.518-3.52}{0.003} = -0.6667 = -0.67$$

$$z2 = \frac{3.523-3.52}{0.003} = +1.0$$

เปิดตารางหาพื้นที่ใต้เส้นโค้งปกติตรงตำแหน่ง z1 และ z2 จะได้ 0.2514 และ 0.8413 ตามลำดับ

ความน่าจะเป็นที่จะพบลูกปืน 0.8413 – 0.2514 = 0.5899

b. หาจำนวนลูกปืนที่มีโอกาสจะถูกคัดทิ้ง ค่าสถิติ z ของข้อมูล 3.515 และ 3.525 หาได้ดังนี้

$$z3 = \frac{3.515-3.52}{0.003} = -1.6667 = -1.67$$

$$z4 = \frac{3.525-3.52}{0.003} = +1.6667 = 1.67$$

เปิดตารางหาพื้นที่ใต้เส้นโค้งปกติตรงตำแหน่ง z3 และ z4 จะได้ 0.0475 และ 0.9525 ตามลำดับ ความน่าจะเป็นของลูกปืนที่ผ่านการคัดเลือก = 0.9525 – 0.0475 = 0.9050

ความน่าจะเป็นที่ลูกปืนจะถูกคัดทิ้งเพราะมีขนาดใหญ่หรือน้อยกว่าที่กำหนดมีค่าเท่ากับ (1 – 0.9050) x 100 = 9.5 %

โปรแกรมที่ใช้ทดสอบการหาพื้นที่ใต้เส้นโค้งปกติในตัวอย่าง 4.6.3 มีดังนี้

```
/* File :AreaTest.java
* Project: Math Tools in Java
* Purpose: Test class for finding area under normal curve
*                                 between lower z and upper z
* Author : Wachara R.
* First Released: Sun 6 May 2007
* Last Updated :
*/

import static java.lang.Math.*;
import MathTools.Common.*;
import MathTools.SimpleStat.*;

class  AreaTest {

       public static void main(String[] args) {
              double area;
              double z1,z2,z3,z4;

              SimpleStat s= new SimpleStat();
              z1=  -0.67;
              z2 = 1.0;
              area = s.findAreaUnderNormalCurve(z1,z2);
              System.out.printf(" a)  Area between %1.4f   to
       %1.4f    =  %1.4f  \n",z1, z2, area);
              System.out.println( "----------------------");
              z3 = -1.67;
              z4 = 1.67;
              area = s.findAreaUnderNormalCurve(z3,z4);
              System.out.printf(" b)  Area between %1.4f   to
        %1.4f    =  %1.4f  \n",z3, z4, area);
              System.out.printf("Ball bearing must be split
       out     =  %1.4f  percents\n",(1.0- area)*100);
       }
 }
```

จะเห็นว่า ได้ใส่ค่า z1 = -0.67 z2 =1.0 z3 = -1.67 และ z4 = 1.67 ลงไปในโปรแกรม โดยที่ไม่ได้ให้โปรแกรมคำนวณหาค่า z ในตัวมันเอง เพราะต้องการขจัดความคลาดเคลื่อนเนื่องจากตำแหน่งทศนิยมไม่เท่ากัน ในการเปิดตารางหาค่าพื้นที่ใต้เส้นโค้งปกตินั้นจะใช้ค่า z ที่มีทศนิยมไม่เกิน 2 ตำแหน่ง

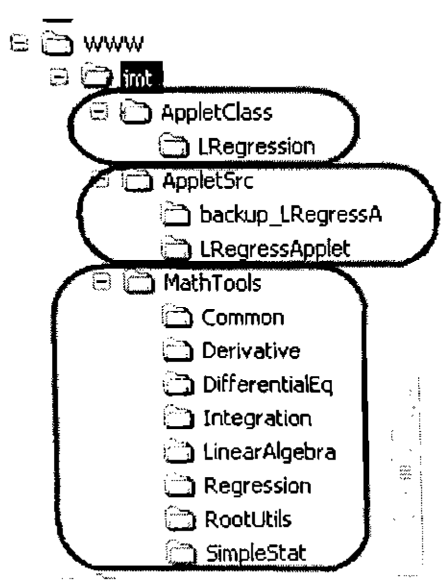
ผลลัพธ์ที่ได้จากการทำงานของโปรแกรมมีดังนี้

```
a)   Area between -0.6700   to    1.0000    =  0.5899
-------------------------------------------------------
b)   Area between -1.6700   to    1.6700    =  0.9051
     ball bearing must be splitted out      =  9.4949    percents
```

จะเห็นว่าผลลัพธ์ที่ได้จากการคำนวณของโปรแกรม และจากวิธีการเปิดตารางมีค่าไม่ต่างกัน

4.7 ทดสอบการทำงานของคลาสไลบรารี ผ่าน web server

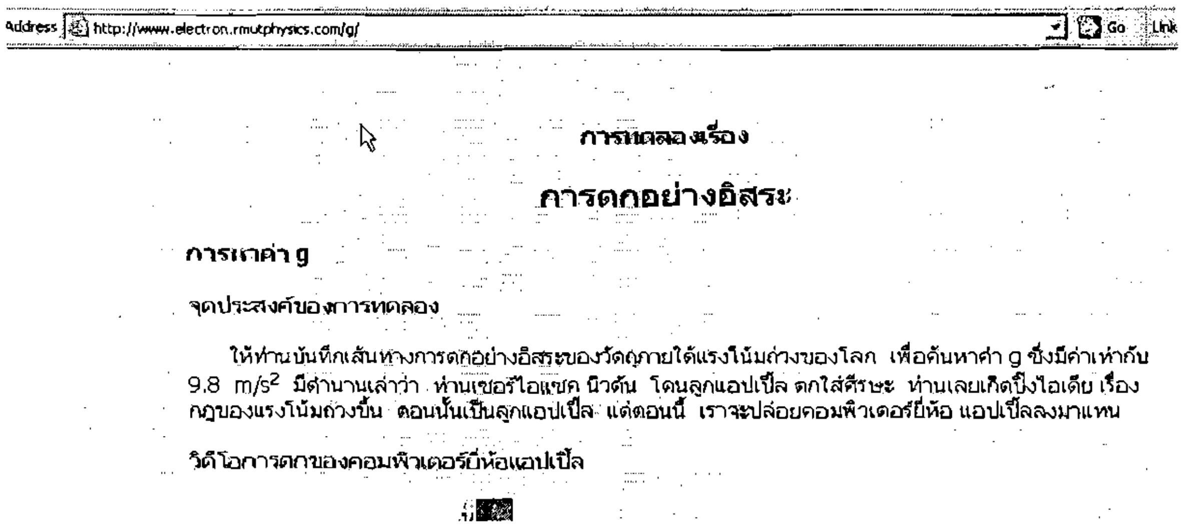
ผู้วิจัยได้ทดสอบคลาสไฟล์โดยนำไปติดตั้งไว้ใน web server ของภาควิชาฟิสิกส์ (http://203.158.100.140/jmt) โดยจัดเก็บไว้ในโฟลเดอร์ c:\www ลักษณะของโฟลเดอร์จะมีการจัดเก็บดังนี้



รูป 4.11 โครงสร้างของโฟลเดอร์ที่ใช้จัดเก็บคลาสไฟล์แต่ละหัวข้อ

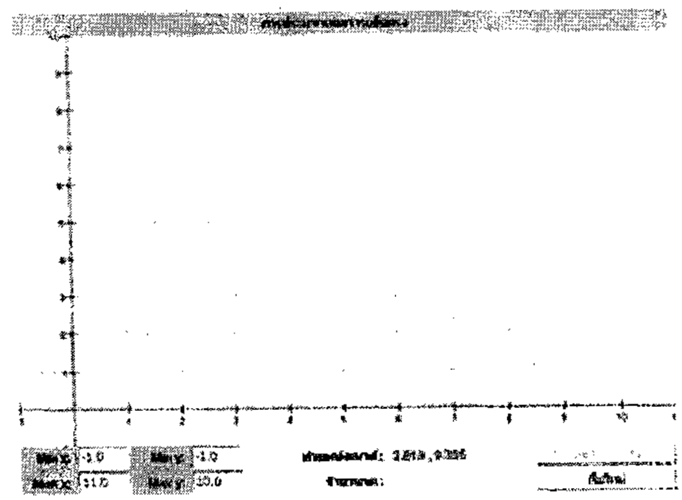
ในการทดสอบการทำงานผ่าน web server ผู้ทดสอบไม่จำเป็นต้องไปหาเครื่องแม่ข่ายเพื่อจัดเก็บไฟล์ต่าง ๆ ผู้ทดสอบสามารถจำลองเครื่องคอมพิวเตอร์ที่ใช้งานอยู่ให้ทำหน้าที่เป็นเสมือน web server ได้เช่นกัน (ดูภาคผนวก 2)

ตัวอย่างต่อไปนี้ผลที่ได้จากการทดสอบโปรแกรมผ่าน web server เป็นการนำคลาส Regression ไปใช้ในการประมาณค่าสมการเส้นตรง โดยอาศัยข้อมูลที่ได้จากการทดลองเรื่องการตกอย่างอิสระ



รูป 4.12 การนำคลาส LinearRegression ไปใช้ในบทเรียนทางฟิสิกส์

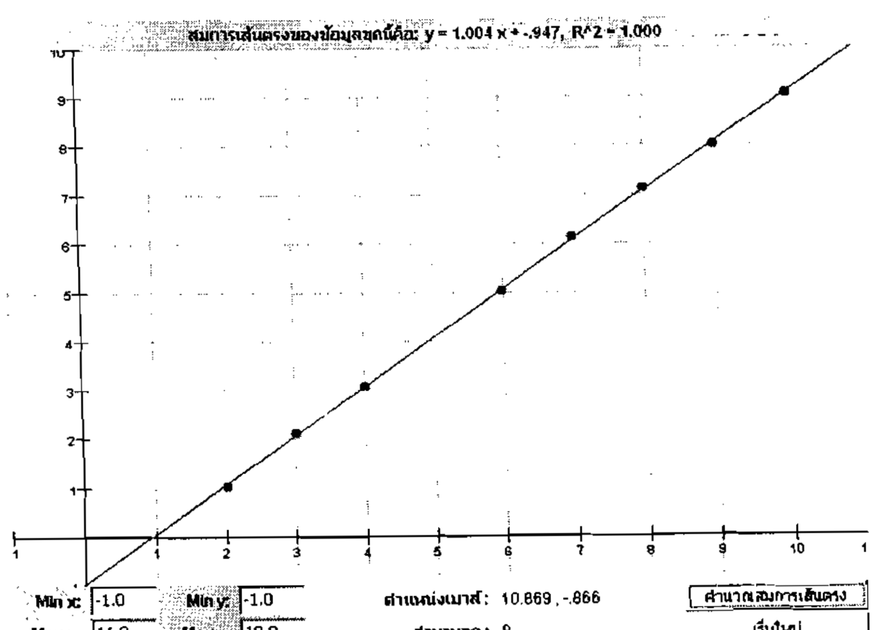
ในการทดสอบนี้ ผู้วิจัยได้เขียนโปรแกรมตรงส่วนที่ติดต่อกับผู้ใช้ (Graphic user interface) เพื่อให้ผู้ใช้ป้อนข้อมูลโดยการคลิกเมาส์บนกระดาษกราฟบนจอภาพ โปรแกรมจะเก็บข้อมูลที่ได้ตามตำแหน่ง (x, y) ณ ตรงจุดที่เมาส์คลิก แล้วนำไปประมวลผลโดยใช้คลาส LinearRegression จะได้สมการเส้นตรงที่เป็นตัวแทนของข้อมูล รูปร่างของกระดาษกราฟจะมีลักษณะดังภาพต่อไปนี้



การวาดกราฟระหว่าง y กับ t^2 ให้ใช้ Applet ดังรูป ทำแทนกราฟในหนังสือก็ได้
และให้นำความชันที่ได้ไปคำนวณหาค่าความเร่งโน้มถ่วง

รูป 4.13 กระดาษกราฟที่ผู้เรียนสามารถใช้เมาส์คลิกตำแหน่งข้อมูล

เมื่อคลิกเมาส์ตามตำแหน่งต่าง ๆ บนกระดาษกราฟแล้ว จากนั้นคลิกปุ่ม "คำนวณสมการเส้นตรง" โปรแกรมจะสร้างเส้นตรง โดยอาศัยการคำนวณของคลาส LinearRegression ดังภาพ



รูป 4.14 กราฟเส้นตรงซึ่งคำนวณโดยอาศัยคลาส LinearRegression

บทที่ 5
สรุปวิจารณ์และข้อเสนอแนะ

จากการทดสอบใช้งานคลาสไลบรารีที่พัฒนามีข้อสรุป ข้อสังเกต ข้อพึงระวัง และข้อเสนอแนะในการใช้งานและการปรับปรุงในการพัฒนาครั้งต่อ ๆ ไปดังนี้

5.1 การหารากสมการแบบไม่เป็นเชิงเส้น

ผู้วิจัยได้สร้างคลาสไลบรารีสำหรับหารากสมการแบบไม่เป็นเชิงเส้นไว้ 4 วิธี วิธีนิวตันราฟสัน และวิธีเซแคนต์ จะใช้จำนวนครั้งในการวนรอบเพื่อหาค่ารากสมการน้อยกว่าวิธีแบ่งครึ่งช่วงและวิธีวางตำแหน่งผิดที่

ระเบียบวิธี (algorithm) การหารากสมการของวิธีแบ่งครึ่งช่วงและวิธีวางตำแหน่งผิดที่มีลักษณะที่ทำความเข้าใจได้ง่ายกว่า เขียนชุดคำสั่งได้ไม่ซับซ้อน โอกาสผิดพลาดในการเขียนโปรแกรมน้อย แต่ใช้จำนวนครั้งในการวนรอบค่อนข้างมาก ถ้ากำหนดค่าเริ่มต้นไม่เหมาะสมจำนวนครั้งการวนรอบอาจเกิน 50 รอบ

วิธีนิวตันราฟสัน เป็นวิธีที่กำหนดค่าเริ่มต้นของรากสมการเพียงค่าเดียว ไม่ต้องกำหนดค่าสุดท้าย แต่ต้องหาอนุพันธ์อันดับ 1 ของฟังก์ชันที่จะหารากสมการนั้นด้วย ถ้าฟังก์ชันมีลักษณะซับซ้อน การหาอนุพันธ์อันดับ 1 ทำได้ยาก วิธีนิวตันราฟสันจะไม่เหมาะกับฟังก์ชันประเภทนี้ ควรใช้วิธีเซแคนต์แทน แต่ต้องกำหนดช่วงของรากสมการทั้งค่าเริ่มต้นและค่าสุดท้าย ข้อเสียประการหนึ่งของวิธีนิวตันราฟสันและวิธีเซแคนต์คือ ถ้าเลือกจุดเริ่มต้นไม่เหมาะสมอาจเกิดการลู่เข้าสู่อนันต์ หรือลู่เข้าสู่คำตอบอื่นที่ไม่ต้องการ

ข้อจำกัดของระเบียบวิธีที่เสนอไว้ทั้ง 4 วิธีนี้ คือใช้หารากสมการของฟังก์ชันที่มีตัวแปรเพียง 1 ตัว แต่ละฟังก์ชันมีค่าต่อเนื่องในช่วงที่จะหารากสมการ รากสมการที่ได้จะเป็นค่าจริงเท่านั้น ไม่สามารถหารากสมการที่มีค่าจินตภาพได้

ข้อเสนอแนะ

- ขยายขีดความสามารถพัฒนาระเบียบวิธีที่หารากสมการที่เป็นจำนวนจินตภาพโดยใช้วิธีของมูลเลอร์ (Muller) หรือวิธีของลาแกร์ (Laguerre's method)
- ควรมีการตรวจสอบหรือดักจับความผิดพลาดถ้าฟังก์ชันนั้นมีค่าไม่ต่อเนื่องตรงบริเวณช่วงที่หารากสมการ

- ควรมีการแจ้งเตือนกรณีที่รากสมการนั้นมีค่าซ้ำกัน เช่น $x^2 -4x + 4 = 0$ มีค่ารากสมการซ้ำกัน คือ $x = 2$ ทั้ง 2 ค่า

- การกำหนดค่าเริ่มต้นและค่าสุดท้ายสำหรับการประมาณค่ารากสมการ ควรกำหนดให้ใกล้ค่ารากสมการที่แท้จริง ในกรณีที่ผู้ใช้ไม่สามารถคาดเดาได้ว่ารากสมการควรอยู่ในช่วงใด ควรมีโปรแกรมที่ช่วยแนะนำช่วงที่มีรากสมการปรากฏอยู่

5.2 การหาผลเฉลยของระบบสมการเชิงเส้น

ผู้วิจัยได้พัฒนาคลาสไลบรารี สำหรับหาผลเฉลยของระบบสมการเชิงเส้น 3 วิธีคือ กฎของคราเมอร์ (Cramer's rule) การลดทอนแบบเกาส์ (Gauss Elimination) และวิธีแยกเป็นเมตริกซ์สามเหลี่ยมล่างและบน (LU Decomposition)

ได้ทดสอบการหาผลเฉลยของทั้งสามวิธีกับปัญหาเดียวกันพบว่าจะให้ผลเฉลยที่ได้ค่าใกล้เคียงกัน ดังนั้น การเลือกวิธีใดไปใช้งานจึงสามารถเลือกวิธีใดก็ได้ให้ผลลัพธ์ใกล้เคียงกัน

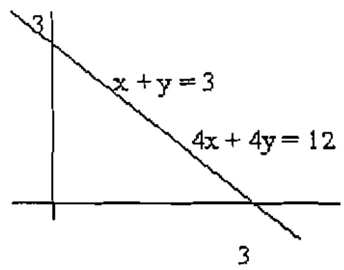
ข้อจำกัดของการหาผลเฉลยของทั้งสามวิธีคือ ผู้วิจัยได้จำกัดจำนวนตัวแปรไว้ไม่เกิน 50 ตัวแปร ต้องการขยายความสามารถของโปรแกรมให้ได้มากกว่า 50 ตัวแปร ทำได้โดยแก้ไขค่า ARRAY_SIZE ในไฟล์/common/CommonConstants.java ให้เป็นจำนวนที่ต้องการ แต่ทั้งนี้ผู้วิจัยได้ทดสอบวิธีทั้งสามนี้กับตัวแปรไม่เกิน 50 ตัวแปรเท่านั้น จำนวนตัวแปรและจำนวนสมการต้องเท่ากันเสมอ

การหาผลเฉลยโดยวิธีลดทอนของเกาส์ ผู้วิจัยได้ลดความคลาดเคลื่อนโดยใช้วิธีสลับแถว (pivoting) มีหลักการดังนี้ ถ้าสมาชิกของเมตริกซ์สัมประสิทธิ์ในแนวเส้นทแยงมุมตัวใดตัวหนึ่งมีค่าเป็นศูนย์หรือมีค่าเข้าใกล้ศูนย์มาก ๆ หรือมีค่าน้อยมากเมื่อเทียบกับสมาชิกตัวอื่น ๆ ในแถวตั้งเดียวกัน การจะนำค่าในแนวเส้นทแยงมุมนี้ไปเป็นตัวหาร ผลลัพธ์ที่ได้จากการหาร (ซึ่งตัวหารมีค่าน้อย ๆ หรือเข้าใกล้ศูนย์) จะมีค่ามหาศาลจนเกินขีดจำกัดของคอมพิวเตอร์ที่จะเก็บไว้ได้ หรือทำให้เกิดความคลาดเคลื่อนเนื่องจากการปัดเศษ การแก้ไขคือสลับแถวนอน เมื่อสลับแล้วทำให้สมาชิกของเมตริกซ์สัมประสิทธิ์ที่มีค่ามากที่สุดอยู่ในแนวเส้นทแยงมุม (อาจสลับแถวตั้งด้วยก็ได้ แต่ในที่นี้จะใช้สลับแถวนอนเพียงแบบเดียว)

ข้อที่ควรคำนึงอีกประการหนึ่งคือ ผลเฉลยของระบบสมการเชิงเส้นที่ไม่เป็นเอกพันธ์ (non-homogenous equation – ระบบสมการเชิงเส้นที่ค่าคงที่ด้านขวามือไม่เป็นศูนย์ทุกตัว) อาจได้ผลเฉลยหลายชุดหรือไม่สามารถหาผลเฉลยได้เลย ตัวอย่างระบบสมการเชิงเส้นที่เกิดปัญหาในการหาผลเฉลย

1. เมื่อมีบางสมการในระบบสมการนั้นมีลักษณะอิงอยู่กับอีกสมการหนึ่งในระบบเดียวกัน ซึ่งเรียกว่า Linearly dependent สมการหนึ่งอาจเกิดจากการนำค่าคงที่คูณอีกสมการหนึ่ง หรือเกิดจากการบวกลบกับอีกสมการหนึ่ง จำนวนผลเฉลยที่ได้จะมีมากมายไม่สิ้นสุด ดังตัวอย่าง

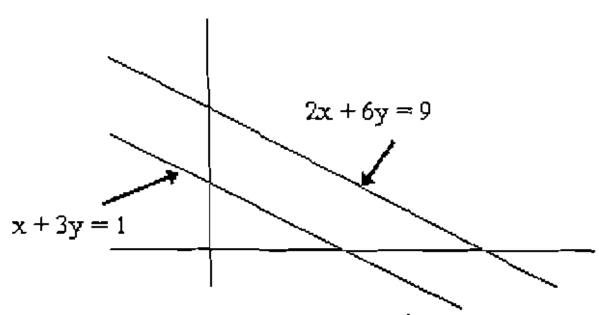
$$x + y = 3$$
$$4x + 4y = 12$$



รูป 5.1 แสดงกราฟเส้นตรงที่เกิดจากสมการทั้งสอง

สมการที่ 2 เกิดจากการนำ 4 คูณกับสมการที่ 1 อันที่จริงแล้วสมการทั้งสองคือสมการเดียวกันนั่นเอง ผลเฉลยที่ได้มีมากมายไม่สิ้นสุด เช่น $x = 0, y = 3$; $x = 1, y = 2$ หรือ $x = 1.5, y = 1.5$, ฯลฯ

2. เมื่อสมการของระบบเป็นเส้นขนานไม่มีจุดตัดกัน ผลเฉลยของระบบสมการนี้จะไม่มี เรียกลักษณะนี้ว่าเป็นระบบขัดแย้ง (inconsistent system) สามารถทำให้สมาชิกของเมตริกซ์สัมประสิทธิ์อย่างน้อย 1 แถว เป็นศูนย์ทั้งหมด โดยการบวกลบกับแถวอื่น ๆ โดยที่ตัวทางขวามือไม่เป็นศูนย์ ตัวอย่างเช่น



$$2x + 6Y = 9$$
$$x + 3y = 1$$

รูป 5.2 กราฟเส้นตรง 2 เส้นที่มีความชันเท่ากัน

เมตริกซ์แต่งเติมแล้วจะได้เป็น $\left[\begin{array}{cc|c} 1 & 3 & 1 \\ 2 & 6 & 9 \end{array}\right]$

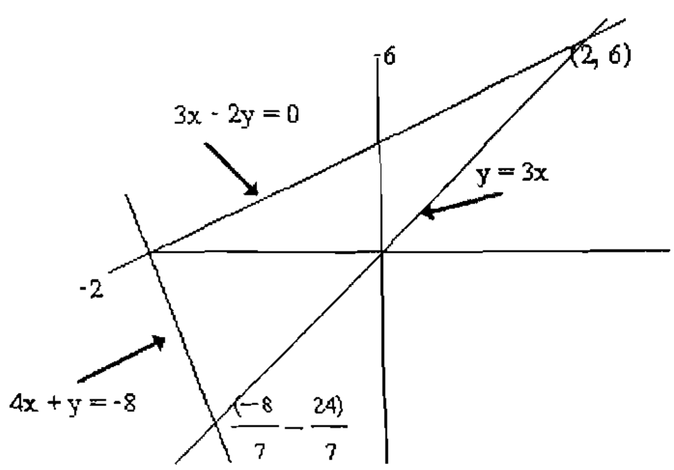
นำ 2 คูณสมาชิกทุกตัวในแถวนอนที่ 1 แล้วไปหักออกจากแถวนอนที่ 2

$$\left[\begin{array}{cc|c} 1 & 3 & 1 \\ 0 & 0 & 7 \end{array}\right]$$

ขนาดเมตริกซ์สัมประสิทธิ์มีค่าเป็นศูนย์ จึงไม่มีผลเฉลยสำหรับระบบสมการชุดนี้

3. ถ้าจำนวนสมการมากกว่าจำนวนตัวแปร สมการแต่ละสมการเป็นอิสระแก่กัน จะไม่สามารถหาคำตอบที่สอดคล้องกับสมการทุกสมการในระบบสมการชุดนี้ได้เลย ตัวอย่างเช่น

$$3x - y = 0$$
$$3x - 2y = -6$$
$$4x + y = -8$$



รูป 5.3 กราฟเส้นตรงแทนสมการทั้งสามสมการ

ผู้วิจัยได้สร้างคลาส MatrixException เป็นคลาสที่ใช้ดักจับความผิดพลาดที่เกิดจากการทำงานของโปรแกรม รวมทั้งกรณีที่ระบบสมการเชิงเส้นเป็นระบบที่อิงกัน (Linearly dependent) หรือเป็นระบบที่ขัดแย้ง (inconsistent system) และตรวจสอบจำนวนตัวแปรและจำนวนสมการต้องเท่ากันเสมอ

ผลลัพธ์ที่ปรากฏในตัวอย่างนี้ ผู้วิจัยให้โปรแกรมแสดงผลออกมาโดยไม่มีการปัดจุด การนำไปใช้งานจริงผู้พัฒนาอาจจะต้องปัดจุดทศนิยมให้เหมาะสมกับงานที่นำไปใช้ด้วย

5.3 **การประมาณค่าโดยวิธีกำลังสองน้อยที่สุด** (Method of Least Square)

การพัฒนาคลาสไลบรารีประมาณค่าโดยวิธีกำลังสองน้อยที่สุด เพื่อสร้างสมการพหุนามจากข้อมูลที่กำหนดให้หรือข้อมูลที่ได้จากการทดลอง จากการทดสอบพบว่าคลาส Polynomial Regression ให้สมการพหุนามตรงกับฟังก์ชันที่ให้กำเนิดข้อมูลชุดนั้น คลาสนี้ยังครอบคลุมถึงสมการเชิงเส้นตรงด้วย ในการใช้งานนั้นมีข้อจำกัดที่ต้องบอกกำลังสูงสุดของตัวแปรอิสระ (x) ให้โปรแกรมและพหุนามที่สามารถหาได้นั้นต้องมีกำลังสูงสุดไม่เกิน 10

ถ้าข้อมูลที่ได้มีความสัมพันธ์แบบไม่เป็นพหุนามอาจอยู่ในรูปล็อกการิธึม, เอ็กซโพเนนเชียล ต้องใช้เทคนิคดัดแปลงรูปสมการโดยการเปลี่ยนตัวแปรเสียก่อน เช่น

$$y = a_0+a_1\ln x$$

กำหนดให้ x = ln x สมการจะกลายเป็น

$$y = a_0+a_1x$$

ซึ่งเป็นสมการเส้นตรงสามารถหาค่า a_0 และ a_1 ได้

ข้อที่ควรพัฒนาเพิ่มเติมในเรื่องการประมาณค่าโดยวิธีกำลังสองน้อยที่สุด มีดังนี้

1. ให้สามารถประมาณค่าได้กับข้อมูลที่มีตัวแปรอิสระหลายตัวแปร ข้อมูลที่ได้ในทางวิทยาศาสตร์อาจขึ้นกับตัวแปรหลาย ๆ ตัวแปร เช่น ความดันของอากาศในกระบอกสูบขึ้นอยู่กับอุณหภูมิ ปริมาตรของกระบอกสูบที่เปลี่ยนไป กระแสไฟฟ้าในวงจร RC ถ้าตัวต้านทานมีการเปลี่ยนค่าได้ กระแสไฟฟ้าจะขึ้นอยู่กับเวลาและความต้านทานที่เปลี่ยนไป เรียกความสัมพันธ์ของตัวแปรตามที่ขึ้นอยู่กับตัวแปรอิสระหลาย ๆ ตัวว่า การถดถอยแบบหลายตัว (Multiple regression) เขียนเป็นสมการได้ดังนี้

$$y = f(x_1, x_2, x_3, \ldots x_j)$$

เมื่อ j คือจำนวนตัวแปรอิสระทั้งหมด

2. ควรมีการตรวจสอบความเหมาะสมของฟังก์ชัน การตรวจสอบว่าข้อมูลที่ได้มานั้นสอดคล้องกับสมการเส้นตรงหรือสมการพหุนาม หรือสมการเอ็กซโพเนนเชียลทำได้โดยใช้วิธีการคำนวณทางสถิติ ที่เรียกว่าหาค่าสัมประสิทธิ์ของการกำหนด (Coefficient of determination, R^2) เป็นการวัดความสัมพันธ์กันหรือสหสัมพันธ์ระหว่างตัวแปรอิสระทั้งหลายกับตัวแปรตาม ค่า R^2 จะ

มีค่าอยู่ระหว่าง 0 -ถึง 1 ถ้าค่า y_i และ $y(x_i)$ มีค่าแตกต่างกันน้อยหรือไม่มีความแตกต่าง จะได้ค่า R^2 ใกล้เคียง 1 ฟังก์ชันที่ได้จะสอดคล้องและเหมาะสมกับข้อมูล ควรได้ค่า R^2 มากกว่า 0.9

ค่าสถิติอีกค่าหนึ่งที่ใช้วัดความเหมาะสมของข้อมูลคือความคลาดเคลื่อนมาตรฐานของการประมาณค่า y (Standard error of the estimate, S_{yx} ได้ดังนี้

$$S_{yx} = \sqrt{\frac{\sum (y_i - G(x_i))^2}{(n-m)}}$$

ค่า S_{yx} ยิ่งน้อยและ R^2 ใกล้เคียง 1 มากเท่าใด แสดงว่าข้อมูลและฟังก์ชันที่คำนวณได้มีความสอดคล้องและเหมาะสมอย่างยิ่ง ในกรณีที่เป็นงานวิจัยจะต้องนำค่า R^2 นี้ไปทดสอบหาความเชื่อมั่นด้วยค่าสถิติ F (F-test) เสียก่อนเพื่อให้มั่นใจว่าค่าที่ได้มานี้มิใช่มาจากความบังเอิญของข้อมูลมาสอดคล้องกับฟังก์ชันที่คำนวณได้พอดี

5.4 การหาค่าอนุพันธ์และปริพันธ์

5.4.1 การหาค่าอนุพันธ์อันดับหนึ่งและสอง

เมื่อกำหนดฟังก์ชัน f(x) โดยที่ฟังก์ชันนี้มีค่าต่อเนื่อง และอนุพันธ์อันดับ 1 และ 2 มีค่าต่อเนื่อง ณ จุดที่ต้องการหาค่าอนุพันธ์นั้น การหาอนุพันธ์อันดับหนึ่งและสองทำได้โดยประมาณค่าโดยใช้สูตร three point formula ในตอนแรกแล้วปรับค่าให้ได้ใกล้ค่าแท้จริงโดยใช้การประมาณแบบริชาร์ดสัน ผลลัพธ์ที่ได้จากการทดสอบการหาอนุพันธ์อันดับหนึ่งและสองให้ผลลัพธ์ถูกต้องเมื่อเทียบกับค่าแท้จริง (ตัวอย่าง 4.4.1 , ตัวอย่าง 4.4.2) ถึงทศนิยมตำแหน่งที่ 10 ความคลาดเคลื่อนที่ยอมรับได้ (DEFAULT_TOLERANCE) ตั้งไว้ที่ 10^{-7}

สิ่งที่ยังขาดหายไปในการพัฒนาครั้งนี้คือ ไม่มีการตรวจสอบความต่อเนื่องของฟังก์ชัน f(x) ณ จุด x ใด ๆ ที่ต้องการหาค่าอนุพันธ์ และกรณีที่ได้ข้อมูลที่ได้มาเป็นตาราง (ไม่ใช่ในรูปฟังก์ชัน) ไม่สามารถนำคลาสไลบรารีทั้งสองนี้ไปหาค่าอนุพันธ์ข้อมูลที่บอกมาในรูปตารางได้

5.4.2 การหาค่าปริพันธ์

ผู้วิจัยได้พัฒนาคลาสไลบรารีสำหรับการหาปริพันธ์ไว้ 4 แบบ คือ คลาส Trapezoidal Integration, คลาส Simpson1_3Integration, คลาส Simpson3_8Integration และคลาส GaussQuadratureIntegration ทั้ง 4 แบบให้ผลลัพธ์การหาค่าปริพันธ์ได้ใกล้เคียงค่าแท้จริง (ตัวอย่าง 4.4.3)

สิ่งที่ควรปรับปรุงในการพัฒนาครั้งต่อ ๆ ไปคือ

1. ควรมีการตรวจสอบความต่อเนื่องของฟังก์ชันตรงบริเวณที่จะหาค่าปริพันธ์

2. การหาค่าปริพันธ์นี้ใช้ได้กับปริพันธ์ชั้นเดียว ควรเพิ่มคลาสที่สามารถหาปริพันธ์แบบ 2 ชั้น และ 3 ชั้น

3. คลาส GaussQuadratureIntegration เป็นวิธีที่ใช้พหุนามเลอจองด์ มาช่วยหาค่าปริพันธ์จำกัดไว้ที่จำนวน node ไม่เกิน 8 จุด สามารถขยายให้มากกว่านี้ได้

4. ระเบียบวิธี (Algorithm) การหาปริพันธ์ยังมีวิธีอื่น ๆ อีกหลายแบบที่ไม่ได้นำเสนอไว้ในการพัฒนาครั้งนี้ เช่นวิธีของ Romberg วิธี Monte Calo เป็นต้น

5.5 การหาคำตอบของสมการอนุพันธ์แบบสามัญ

ขอบเขตการใช้งานคลาสไลบรารีของหัวข้อนี้คือ ใช้หาคำตอบของสมการอนุพันธ์แบบสามัญที่มีตัวแปรอิสระ 1 ตัว และมีลักษณะเชิงเส้นโดยกำหนดเงื่อนไขเริ่มต้น คลาสไลบรารีแสดงคำตอบที่ได้อยู่ในรูปเชิงตัวเลข (ไม่ใช่สัญลักษณ์)

คลาสที่ออกแบบไว้สำหรับแก้ปัญหาสมการอนุพันธ์แบบสามัญมีจำนวน 3 คลาส คือ คลาส ModifiedEuler คลาส RungeKutta และคลาส AdamsMoulton ผลจากการทดสอบการทำงานของคลาสไลบรารีเหล่านี้พบว่า คลาส AdamsMoulton จะให้คำตอบใกล้ค่าแท้จริงมากที่สุด และใช้เวลาในการประมวลผลน้อยที่สุด แต่ระเบียบวิธีของคลาสนี้ค่อนข้างซับซ้อนยากต่อการทำความเข้าใจ คลาส RungeKutta ให้ผลลัพธ์ใกล้ค่าแท้จริงและใช้เวลาในการประมวลผลเป็นอันดับสอง คลาส ModifiedEuler ให้ผลลัพธ์ได้มีความเที่ยงตรงค่อนข้างต่ำเมื่อเทียบกับวิธีอื่น และใช้เวลาในการประมวลผลมากที่สุด แต่ระเบียบวิธีของคลาสนี้เป็นวิธีแบบขั้นตอนเดียว จึงทำความเข้าใจและเขียนโปรแกรมได้ง่าย

ข้อที่ควรพัฒนาเพิ่มเติม

1. ควรขยายขีดความสามารถให้หาคำตอบของสมการอนุพันธ์แบบสามัญได้เมื่อกำหนดเงื่อนไขขอบเขต (Boundary condition)

2. ควรขยายขีดความสามารถให้เพิ่มขึ้น สามารถหาคำตอบสมการอนุพันธ์ที่มีอันดับมากกว่าหนึ่ง โดยกำหนดเงื่อนไขเริ่มต้นหรือกำหนดเงื่อนไขขอบเขตมาให้

ในการหาคำตอบสมการอนุพันธ์อันดับสองนั้น สามารถใช้เทคนิคการเปลี่ยนแปลงรูปสมการให้เป็นสมการอนุพันธ์อันดับหนึ่ง แล้วใช้คลาสใดคลาสหนึ่งหาคำตอบได้เช่นกัน ตัวอย่างเช่น

$$L\frac{d^2 I}{dt^2} + R\frac{dI}{dt} \quad = \quad 0$$

ให้ $\frac{dI}{dt} = z$ สมการอนุพันธ์จะเปลี่ยนรูปเป็น

$$L\frac{dz}{dt} + Rz = 0$$

$$\frac{dz}{dt} = -\frac{Rz}{L} = f(z, t)$$

ซึ่งจะเป็นสมการอนุพันธ์อันดับหนึ่ง

5.6 การคำนวณค่าสถิติเบื้องต้น

คลาส Simple Stat ได้ออกแบบไว้สำหรับหาค่าสถิติเบื้องต้นของข้อมูล ค่าสถิติที่คำนวณ ได้แก่ ความถี่ ความถี่สะสม ค่าเฉลี่ย ค่ามัธยฐาน ค่าฐานนิยม พิสัย ค่าส่วนเบี่ยงเบนมาตรฐาน ในการคำนวณนั้นได้ใช้ข้อมูลดิบทั้งหมดไม่แจกแจงเป็นตารางแบ่งข้อมูลเป็นช่วงชั้น (grouped data)

ผลลัพธ์ที่ได้จากการคำนวณ เมื่อเปรียบกับค่าที่คำนวณได้จากโปรแกรมตารางคำนวณ excel พบว่าได้ผลลัพธ์ตรงกัน

การแจกแจงของข้อมูลที่ได้ตั้งอยู่ในสมมติฐานที่ว่าการกระจายของข้อมูลเป็นรูปโค้งปกติ ภายในคลาส SimpleStat ได้มีการคำนวณค่าสถิติ z และพื้นที่ใต้โค้งระหว่างค่า z ใด ๆ ซึ่งบอกถึงความน่าจะเป็นของข้อมูลที่อยู่ระหว่างค่า z นั้น ๆ ซึ่งสามารถคำนวณได้โดยไม่ต้องอาศัยการเปิดตารางหาค่าสถิติ z

ได้ตรวจสอบผลลัพธ์การหาพื้นที่ใต้โค้งสะสมตั้งแต่ค่า z =-1.99 ถึง z = 1.99 ความละเอียดของทศนิยม 4 ตำแหน่ง พบว่าได้ค่าตรงกับตารางสถิติ z ในหนังสือสถิติทุกประการ [2] [3]

ค่าสถิติเบื้องต้นบางปริมาณที่ไม่ได้ใส่ไว้ในคลาส SimpleStat ได้แก่ ค่าส่วนเบี่ยงเบนควอไทล์ (Quartile deviation) การหาเปอร์เซนไทล์ (percentile) ความเบ้ (skewness) และความโด่ง (kurutis) ของการกระจายของข้อมูล

การหาค่าสถิติ z เพื่อหาความน่าจะเป็นของข้อมูลนิยมใช้กับข้อมูลที่มีจำนวนมาก ๆ (มากกว่า 30) ถ้าข้อมูลมีเป็นจำนวนน้อยควรใช้ค่าสถิติ t แทน ซึ่งในคลาส SimpleStat ไม่ได้มีการตรวจสอบจำนวนข้อมูลและไม่มีเมธอดหาค่าสถิติ t เตรียมไว้ให้

ในการเรียงลำดับข้อมูล สามารถกำหนดให้เรียงลำดับจากค่าน้อยไปสู่ค่ามากหรือจากค่ามากไปสู่ค่าน้อย ขั้นตอนวิธีการเรียงลำดับได้เลือกวิธี Quick sort เพียงวิธีเดียวไม่มีวิธีอื่นให้เลือก ยังไม่ได้ทำการทดสอบว่าถ้าเปลี่ยนวิธีเรียงลำดับไปเป็นแบบอื่น โปรแกรมจะประมวลผลเร็วขึ้นและมีประสิทธิภาพมากขึ้นหรือไม่

คลาส SimpleStat1 ไม่ได้รวมการนำเสนอข้อมูลไว้ให้จึงไม่สามารถแสดงผลเป็นกราฟเส้น กราฟแท่ง หรือวงกลมได้ และไม่สามารถนำเสนอข้อมูลเป็นรูปภาพ (pictograph) ได้

ในกรณีที่มีข้อมูล 2 ชุด ชุดที่ 1 มีค่าเป็น $x_1, x_2, x_3, \ldots x_n$ ชุดที่ 2 มีค่าเป็น $y_1, y_2, y_3, \ldots y_n$ การหาค่าสถิติเบื้องต้นของข้อมูลแต่ละชุดจะต้องสร้าง object ของคลาส Simple Stat ขึ้นมา 2 object เพื่อประมวลผลแยกกัน การหาความสัมพันธ์ระหว่างข้อมูลทั้งสองชุดต้องนำคลาส Polynomial Regression (ในหัวข้อ 4.3) มาใช้งานอีกทอดหนึ่ง ซึ่งทำให้ดูยุ่งยากพอสมควร

บรรณานุกรม

1. ศิริพงษ์ ศรีพิพัฒน์. **คณิตศาสตร์เชิงตัวเลข.** สงขลา : มหาวิทยาลัยสงขลานครินทร์, 2530.
2. วีนัส พีชวณิชย์ และคณะ. **สถิติพื้นฐานสำหรับนักสังคมศาสตร์** กรุงเทพฯ : มหาวิทยาลัยธรรมศาสตร์, 2547.
3. ธานินทร์ ศิลป์จารุ. **การวิจัยและวิเคราะห์ข้อมูลทางสถิติด้วย SPSS.** กรุงเทพฯ : วี. อินเตอร์ พริ้นท์, 2550.
4. Lowe, Doug. **Java for Dummies (All in One Desk Reference).** Indiana: Wiley Publishing, 2005.
5. Mak, Ronald. **Java Number Cruncher.** New Jersey: Pearson Education, Inc., 2003.
6. Brian, cole. Eckstein, Robert., **Java Swing** , O'Reilly., 2002.
7. Luján, Mikel. Freeman, T. L., Gurd, John R., **OoLALA: an object oriented analysis and design of numerical linear algebra.** Proceedings of the 15th ACM SIGPLAN conference. , 2000.
8. Moreiera, Jose E, **A comparison of three approaches to language, compiler, and library support for multidimensional arrays in Java,** Proceedings of the 2001 joint ACM-ISCOPE conference, 2000.
9. Sedgewick, Robert. **Algorithms in Java, part 5 Graph Algorithm.** Boston: Pearson Education, Inc., 2004.
10. O'Hagen, Don. **Core Java 2, vol II.** New York: Sun Micro System, Inc., 2005.
11. Richardson, W. Clay. **Professional Java 6.** Indiana: Wiley Publishing, 2007.
12. Palmer, Grunt. Technical Java: **Developing Scientific and Engineering Application.** New Jersey : Pearson Education, Inc., 2003.
13. Al-Khafaji Amir W. and Tooley Hohn R. **Numerical Methods in Engineering Practice.** New York : CBS Collage, 1986.
14. Bajpai, A.C. and others. **Engineering Mathematics. 2nd edition.** Singapore: John Willey & Sons, Inc., 1990.
15. Burden, Richard L. and Faires, Douglas J. **Numerical Analysis.** 5 th edition. Massachusetts : PWS Publishing., 1993.

16. Dejong, Marvin L. **Introduction to Computational Physics**. New York: Addison - Wesley., 1991.

17. Henrici,P. **Elements of Numerical Analysis**. New York: John Willey & Sons, Inc., 1964.

18. Kreyszig, Erwin **Advance Engineering Mathematics**. New York: John Willey & Sons,Inc., 1988.

19. Nakamura, S. **Applied Numerical Methods in C**. New York: Prentice-Hall, Inc., 1993.

20. Scheid, Francis. **Numerical Analysis, Schaum's Outline Series**. Singapore: McGraw - Hill., 1983.

21. Staff of Research and Education Association. The **Numerical Analysis Problem** Solver. New York: Research and Education Association, 1984.

22. Walker, Robert D. **Numerical Methods for Engineers and Scientists**. New York: Tab Book Inc., 1987.

ภาคผนวก 1

วิธีการทดสอบ Java Math Tools Class Library.

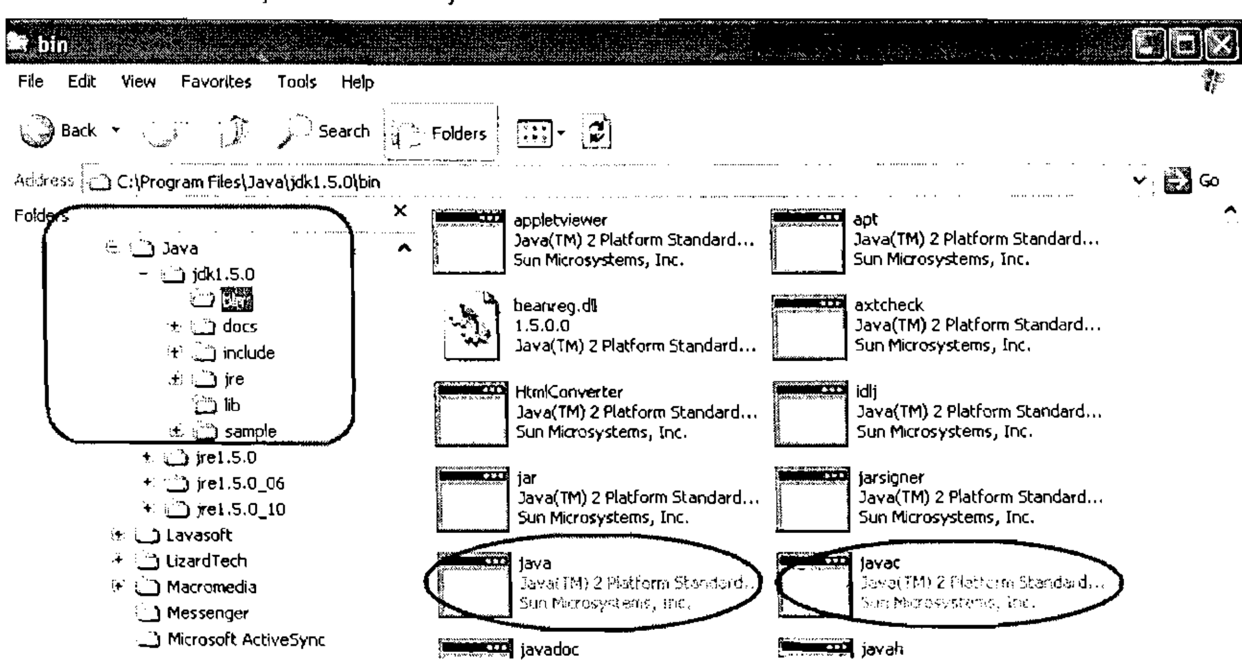
หลังจากได้พัฒนาคลาสเกี่ยวกับคณิตศาสตร์เรียบร้อยแล้ว ขั้นตอนที่สำคัญขั้นตอนหนึ่งคือทดสอบการทำงานของคลาสที่สร้างขึ้นมา เพื่อหาข้อผิดพลาดและข้อบกพร่องอันอาจจะพึงมี เพื่อจะได้นำมาแก้ไขให้คลาสที่สร้างขึ้นมีความสมบูรณ์ที่สุดเท่าที่จะทำได้ วิธีการตามขั้นตอนต่อไปนี้ได้เขียนไว้ให้ผู้ร่วมพัฒนาใช้เป็นคู่มือในการทดสอบ มีภาพประกอบการทดสอบเพื่อให้ทำตามและเข้าใจได้ง่าย

1. โปรแกรมที่ใช้ในการทดสอบ

1. JDK 1.6 (Java Development Kit รุ่น 1.6) ต้องใช้รุ่น 1.6 หรือสูงกว่าเท่านั้น ให้ทำการติดตั้งลงไปในฮาร์ดดิสก์

2. Edit plus รุ่น 2.1 หรือ โปรแกรม text editor อื่น ๆ

เมื่อติดตั้งเสร็จแล้วทั้งสองรายการ จะเห็นว่าในโฟลเดอร์ที่ติดตั้ง Java จะมีโปรแกรมอยู่มากมายหลายโปรแกรม และมีโฟลเดอร์ jre เวอร์ชันต่าง ๆ โปรแกรมที่จะใช้ในการทดสอบครั้งนี้มีเพียง 2 โปรแกรมคือ javac.exe และ java.exe



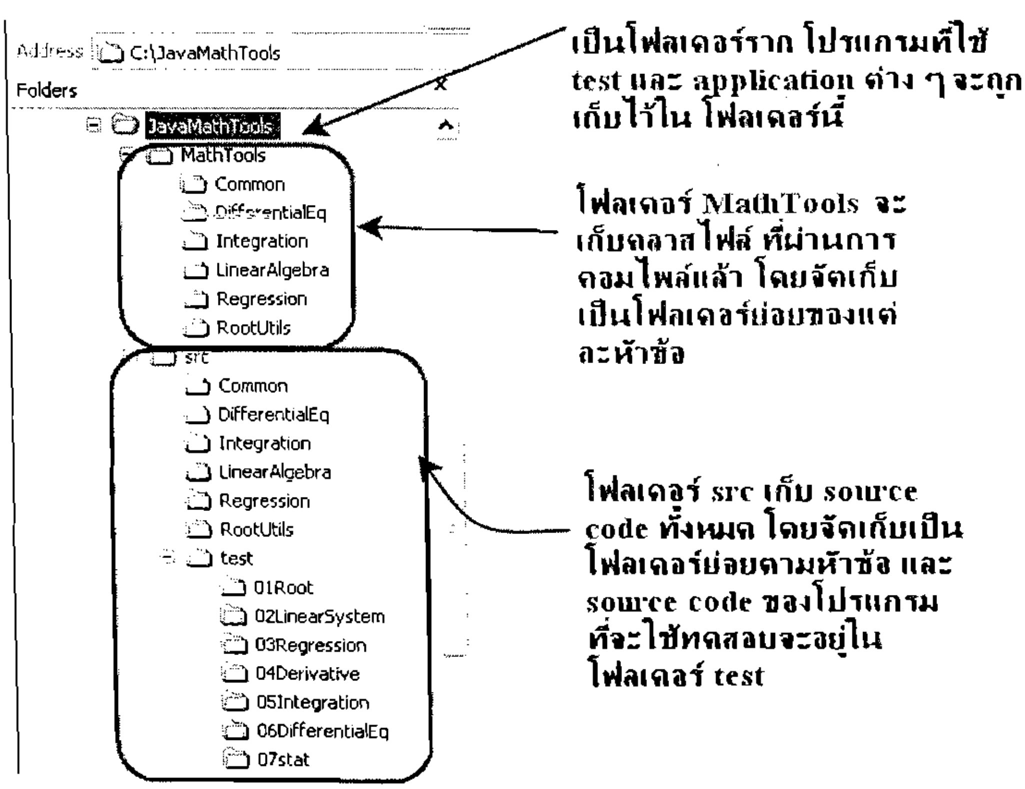
javac.exe ย่อมาจาก java compiler ใช้ในการแปลภาษาจาวา ที่เราเขียนเป็น source code ด้วย edit plus ให้เป็น byte code เพื่อทำให้ Interpreter แปลคำสั่งเหล่านี้เป็นภาษาเครื่องหรือภาษาที่เป็นเลขฐานสอง ที่คอมพิวเตอร์สามารถนำไปใช้งานหรือปฏิบัติได้

java.exe คือตัว Interpreter นำ byte code ที่ได้จากการคอมไพล์ด้วย javac.exe มาแปลงเป็นภาษาเครื่องหรือเลขฐานสอง ซึ่งเป็นภาษาที่ฮาร์ดแวร์ของคอมพิวเตอร์รู้จัก แล้วเชื่อมโยง (link)กับคลาสต่าง ๆ ที่เกี่ยวข้องกับโปรแกรม เพื่อให้คอมพิวเตอร์ประมวลผล

โฟลเดอร์ jre (Java Runtime Environment)คือที่เก็บ class library ต่าง ๆ ที่มาพร้อมกับ jdk โดยจะเก็บคลาสเหล่านี้อยู่ในรูป zip file ซึ่งในภาษาจาวาจะเรียกไฟล์เหล่านี้ ว่า jar file

2. เตรียมการทดสอบ

ให้ทำการ copy โฟลเดอร์ JavaMathTools ทั้งโพลเดอร์ ไปไว้ที่ไดรว์ c:\ (ในกรณีที่ copy จาก CD จะต้องเปลี่ยน attribute ของไฟล์ จาก read only ให้สามารถเขียนทับได้เสียก่อน) ลักษณะของโฟลเดอร์จะเป็นดังนี้



3. **ดำเนินการทดสอบ** สิ่งที่จำเป็นต้องรู้ก่อนดำเนินการทดสอบ

- ใช้ editplus คล่อง สามารถเรียกโปรแกรมมาแก้ไข edit และ save file ได้
- สามารถเขียน mathematic expression ให้เป็นภาษาจาวาได้
- สามารถ compile โปรแกรม source code และ run ดูผลของโปรแกรมได้

Mathematic expression ในภาษาจาวา จะมีรูปแบบเหมือนของภาษา C และ C++

ฟังก์ชันคณิตศาสตร์ทั่ว ๆ ไป

Method	Function	Argument Type	Result Type
sqrt(arg)	Calculates the square root of the argument.	double	double
cbrt(arg)	Calculates the cube root of the argument	double	double
pow(arg1, arg2)	Calculates the first argument raised to the power of the second argument , $arg1^{arg2}$	Both double	double
hypot (arg1, arg2)	Calculates the square root of ($arg1^2$+ $arg2^2$)	Both double	double
exp (arg)	Calculates e raised to the power of the argument, e^{arg}	double	double
expm1(arg)	Calculates e raised to the power of the argument munus 1, e^{arg} - 1		
log (arg)	Calculates the natural logarithm (base e) of the argument	double	double
log1p(arg)	Calculates the natural logarithm (base e) of arg+1	double	double
log10 (arg)	Calculates the base 10 logarithm of the argument	double	double
random()	returns a pseudo-random number greater than or equal to 0.0 and less than 1.0	None	double

Method	Function	Argument Type	Result Type
abs (arg)	Calculates the absolute value of the argument	int, long, float or double	The same type as the argument
max (argl, arg2)	Returns the larger of the two arguments, both of the same type	int, long, float, Or double	The same type as the argument
min (arg1, arg2)	Returns the smaller of the two arguments, both of the same type	Int, long, float, or double	The same type As the argument
ceil (arg)	Returns the smallest integer that is greater than or equal to the argument	double	double
floor (arg)	Returns the largest integer that is less than or equal to the argument	double	double
round (arg)	Calculates the nearest integer to the argument value	float or double	of type int for a float argument, of type long for a double argument
rint (arg)	Calculates the nearest integer to the argument value	float or double	Of type int for A float Argument, of Type long for A double argument
IEEEremainder (arg1, arg2)	Calculates the remainder when arg1 is divided by arg2	both of type double	Of type double

ฟังก์ชันตรีโกณมิติ Built in มาพร้อมกับ JDK 1.6 มีดังนี้

Method	Function	Argument Type	Result Type
sin(arg)	sine of the argument	double in radians	double
cos(arg)	cosine of the argument	double in radians	double
tan(arg)	tangent of the argument	double in radians	double
asin(arg)	$\sin^{-1}$(arc sine) of the argument	double	double in radi-ans, with values from $-\pi/2$ to $\pi/2$
acos(arg)	Cos^{-1}(arc cosine) of the argument	double	double in radi-Ans, with values from 0.0 to π
atan(arg)	$\tan^{-1}$(arc tangent) of the argument	double	double in radi-ans, with values from $-\pi/2$to$\pi/2$
atan2(arg1,arg2)	$\tan^{-1}$(arc tangent) of arg1/arg2	Both double	Double in radians, with $-\pi$ toπ

ฟังก์ชันไฮเปอร์บอลิก ที่ built in ได้แก่

Method	Function	Argument Type	Result Type
sinh(arg)	Hyperbolic sine of the argument, which is:$(e^{arg}-e^{-arg})/2$	double	double
cosh(arg)	Hyperbolic cosine of the argument, which is:$(e^{arg}-e^{-arg})/2$	double	double
tanh(arg)	Hyperbolic tangent of the argument, which is:$(e^{arg}-e^{-arg})/(e^{arg}+e^{-arg})$	double	double

ตัวอย่างการเขียนฟังก์ชันทางคณิตศาสตร์

<u>**ตัวอย่าง**</u> ปริมาตรของทรงกระบอก

volume $= \pi r^2 h$ ในภาษาจาวา เขียนเป็น volume = PI * r * r *h;

<u>**ตัวอย่าง**</u> รากของสมการ quadratic equation

$$x = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$ เขียนได้เป็น x = (-b + sqrt(b*b – 4*a*c))/(2*a);

<u>**ตัวอย่าง**</u> การเคลื่อนที่วิถีโค้งแบบ 2 มิติ

$x = (u \cos 60^\circ)t$ เขียนได้เป็น x = u* cos (60*PI/180)*t;

$y = (u \sin 60^\circ)\, t - 0.5gt^2$ เขียนได้เป็น y = u*sin(60*PI/180)*t -0.5*g*t*t;

<u>**ตัวอย่าง**</u> สมการประจุไฟฟ้าในวงจร RLC

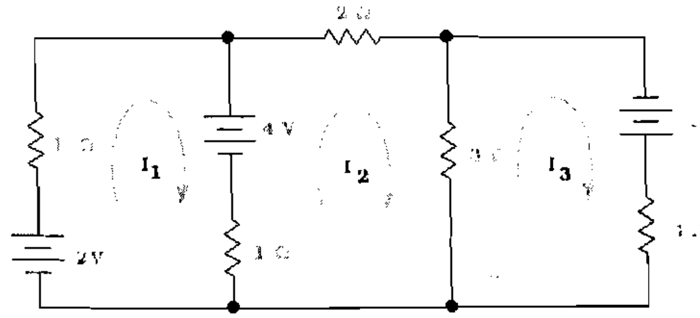
$$q = q_0 e^{-Rt/2L} \cos\left(t\sqrt{\frac{R^2}{4L} - \frac{1}{LC}}\right)$$ เขียนได้เป็น

q = q0 * exp(-R*t/(2*L))*cos(t*sqrt(pow(R,2)/(4*L) –(1/(L*C))));

<u>**การคอมไพล์โปรแกรมต้นฉบับ และ ให้โปรแกรมทำงานเพื่อดูผลลัพธ์**</u>

<u>**ตัวอย่าง**</u> แสดงวิธีการทดสอบ ระบบสมการเชิงเส้น

มีโจทย์ปัญหาดังนี้ จงหากระแสไฟฟ้า I_1, I_2 และ I_3



วิธีทำ เขียนสมการของกระแส I_1, I_2 และ I_3 เป็นระบบสมการเชิงเส้นดังนี้

$$\begin{aligned} -2I_1 + I_2 &= 2 \\ I_1 - 6I_2 + 3I_3 &= -4 \\ 3I_2 - 7I_3 &= -2 \end{aligned}$$

แก้สมการหาค่ากระแสไฟฟ้า I_1 , I_2 และ I_3

$$I_1 = \frac{\begin{vmatrix} +2 & +1 & 0 \\ -4 & -6 & +3 \\ -2 & +3 & -7 \end{vmatrix}}{\begin{vmatrix} -2 & +1 & 0 \\ +1 & -6 & +3 \\ 0 & +3 & -7 \end{vmatrix}} = -\frac{32}{59} = -0.5424 \quad \text{A}$$

$$I_2 = \frac{\begin{vmatrix} -2 & +2 & 0 \\ +1 & -4 & +3 \\ 0 & -2 & -7 \end{vmatrix}}{\begin{vmatrix} -2 & +1 & 0 \\ +1 & -6 & +3 \\ 0 & +3 & -7 \end{vmatrix}} = \frac{54}{59} = 0.9153 \quad \text{A}$$

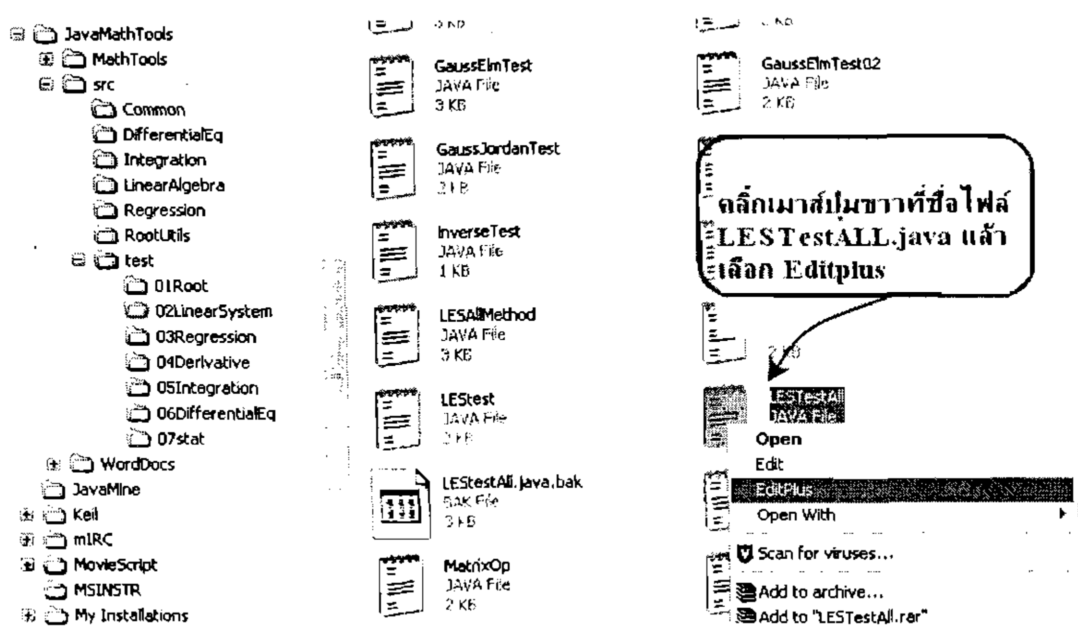
$$I_3 = \frac{\begin{vmatrix} -2 & +1 & +2 \\ +1 & -6 & -4 \\ 0 & -3 & -2 \end{vmatrix}}{\begin{vmatrix} -2 & +1 & 0 \\ +1 & -6 & +3 \\ 0 & +3 & -7 \end{vmatrix}} = \frac{40}{59} = 0.6780 \quad \text{A}$$

จัดรูปสมการใหม่ให้อยู่ในรูปเมทริกซ์ AX = b จะได้

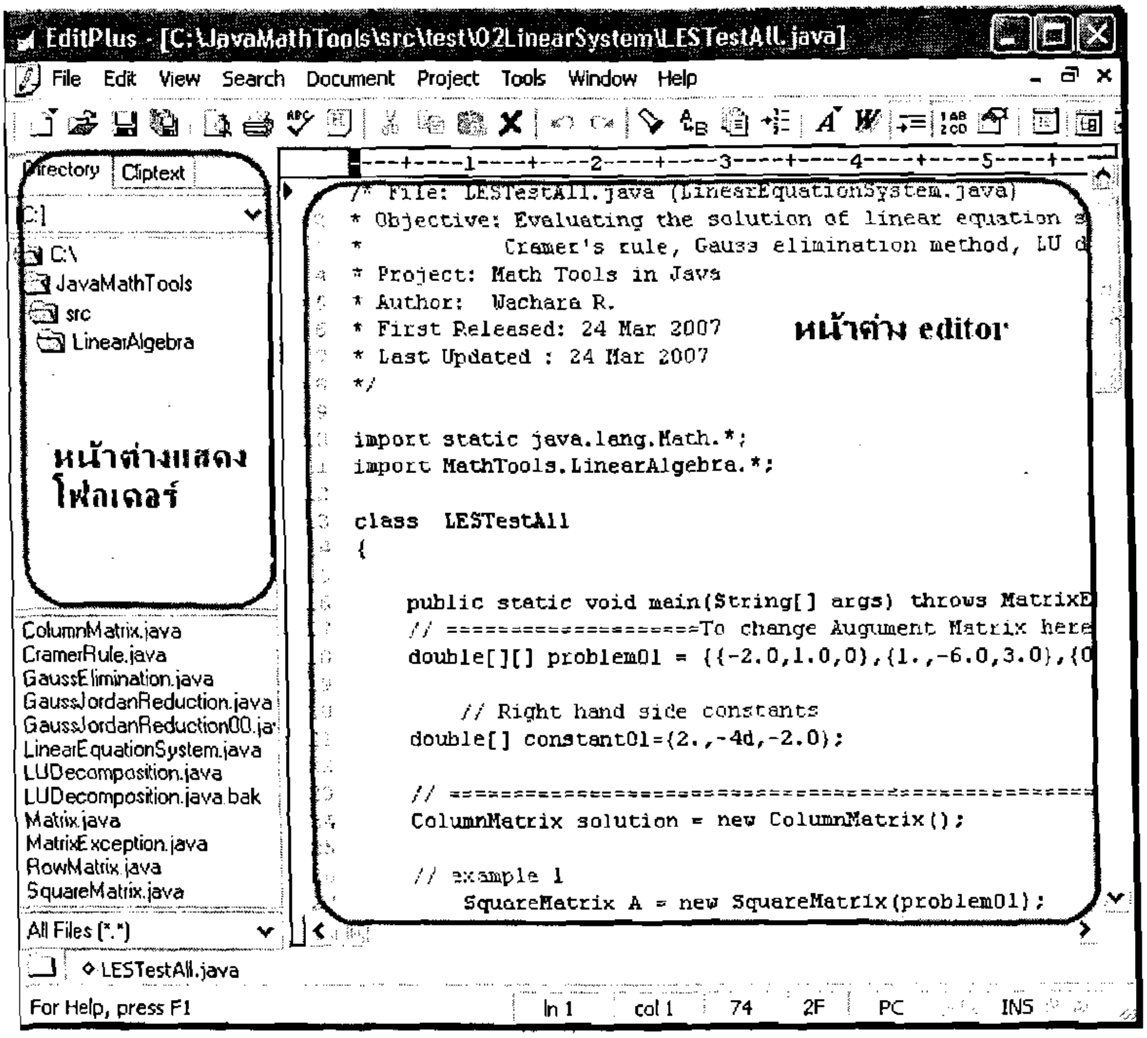
$$\begin{vmatrix} -2 & 1 & 0 \\ 1 & -6 & 3 \\ 0 & 3 & -7 \end{vmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} 2 \\ -4 \\ -2 \end{bmatrix}$$

ขั้นตอนการทดสอบระบบสมการเชิงเส้น

1. โปรแกรมที่จะใช้ test อยู่ในโฟลเดอร์ c:\JavaMathTools\src\test\02LinearSystem ชื่อของไฟล์โปรแกรมที่จะนำมาทดสอบคือ LESTestAll.java (ชื่อไฟล์ย่อมาจาก Linear Equation System Test All) ให้คลิกเมาส์ปุ่มขวา เพื่อเปิดทำการแก้ไขด้วย Editplus



2. Source code ของโปรแกรม LESTestAll.java จะปรากฏอยู่ในหน้าต่าง editor ของโปรแกรม Editplus ดังรูป



3. ทำการแก้ไขโปรแกรม โดยเติมค่าสมาชิกของเมตริกซ์ A และ ค่าคงที่ด้านขวามือ column matrix b ดังรูป

พิมพ์ augument matrix (เมตริกซ์ A) และ ค่าคงที่ลงตรงนี้

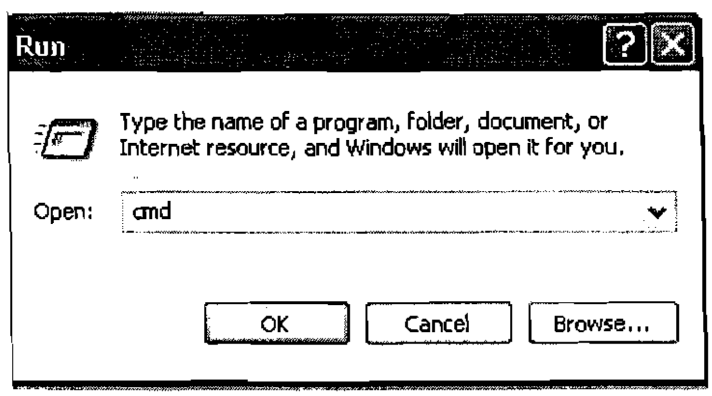
```
import static java.lang.Math.*;
import MathTools.LinearAlgebra.*;

class  LESTestAll
{

    public static void main(String[] args) throws MatrixException
    // ===================To change Augument Matrix here ==========
    double[][] problem01 = {{-2.0,1.0,0},{1.,-6.0,3.0},{0.,3.,-7.}};

        // Right hand side constants
    double[] constant01={2.,-4d,-2.0};

    // ==================================================================
    ColumnMatrix solution = new ColumnMatrix();

    // example 1
        SquareMatrix A = new SquareMatrix(problem01);
        ColumnMatrix b = new ColumnMatrix(constant01);
        System.out.println(" Square Matrix of Problem 1.-->");
        A.printMatrix();
```

เมตริกซ์ A

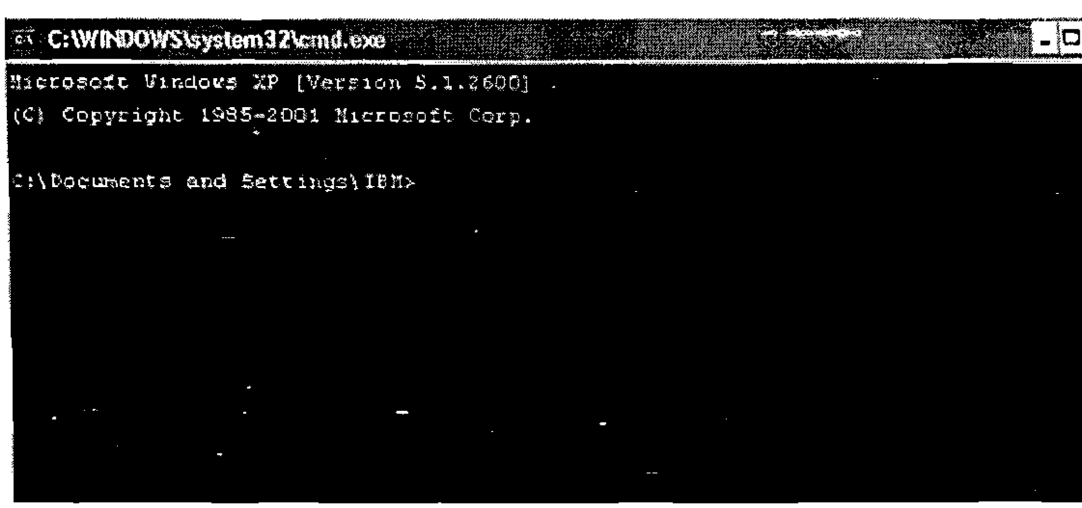
ค่าคงที่ด้านขวามือ

4. save โปรแกรมเพื่อบันทึกการแก้ไข โปรแกรมนี้จะนำค่าเมตริกซ์ A และ เมตริกซ์ b ไปแก้หาคำตอบของระบบสมการเชิงเส้น 3 แบบด้วยกัน แบบแรกใช้วิธี Cramer's rule แบบที่สองใช้วิธี Gauss elimination และแบบที่ 3 ใช้วิธี LU decomposition (วิธีการแตกเมตริกซ์ A ให้อยู่ในรูปผลคูณของ Lower matrix (L) และ Upper matrix (U))

5. เริ่มขั้นตอนการคอมไพล์ โดยคลิกที่ปุ่ม start >> run >> cmd เป็นการคอมไพล์โดยเราจะเป็นผู้พิมพ์คำสั่งใน command console



6. เมื่อคลิกปุ่ม OK จะปรากฏ command console ดังรูป



7. พิมพ์คำสั่ง cd \ (cd ตามด้วยเครื่องหมาย back slash) เป็นคำสั่งที่ให้กลับไปที่โฟลเดอร์ราก คือ c:\

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\IBM>cd\

C:\>_
```

8. เข้าไปที่โฟลเดอร์ src ซึ่งเป็นโฟลเดอร์ที่เก็บ source code ของเรา โดยใช้คำสั่งดังนี้

```
C:\>cd JavaMathTools/src

C:\JavaMathTools\src>_
```

9. คอมไพล์โปรแกรม LESTestAll.java โดยใช้คำสั่งดังนี้

```
C:\JavaMathTools\src>javac -d .. -cp .. ./test/02LinearSystem/LESTestAll.java

C:\JavaMathTools\src>
```

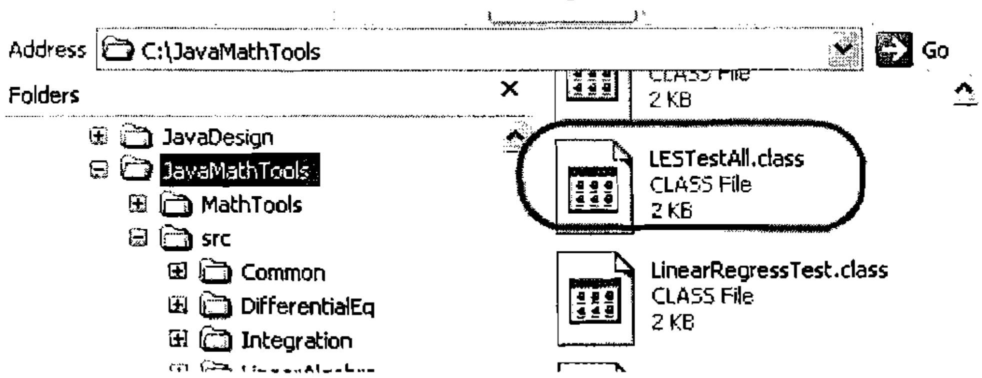
ถ้า source code ไม่มีข้อบกพร่อง หน้าจอจะไม่แสดงข้อผิดพลาดใด ๆ

Option ที่ปรากฏท้ายคำสั่ง javac คือ –d หมายถึง โฟลเดอร์ที่จะเก็บโปรแกรมซึ่งมีนำสกุลเป็น class ในที่นี้เป็น จุด 2 จุด หมายถึงให้เก็บ class file ที่ได้จากการคอมไพล์นี้ไว้ในโฟลเดอร์ที่อยู่เหนือ src ขึ้นไป 1 ลำดับชั้น

-cp นั้นย่อมาจากคำว่า class path เป็นการระบุว่า class fileที่เป็น library นั้น ตัวคอมไพเลอร์สามารถไปค้นหาโดยเริ่มต้นจากโฟลเดอร์ที่อยู่เหนือ src ไป 1 ลำดับชั้นเช่นเดียวกัน

./test/02LinearSystem/LESTestAll.java เป็นการระบุชื่อไฟล์ที่จะทำการคอมไพล์

เมื่อคอมไพล์เสร็จแล้วจะได้ LESTestAll.class อยู่ในโฟลเดอร์ c:\JavaMathTools



10. ทดลอง run โปรแกรมเพื่อทดสอบผลลัพธ์ที่ได้ โดยใช้คำสั่ง java ดังนี้

```
C:\JavaMathTools\src>cd ..

C:\JavaMathTools>java LESTestAll
```

จะได้ผลลัพธ์จากการรันโปรแกรมดังรูป

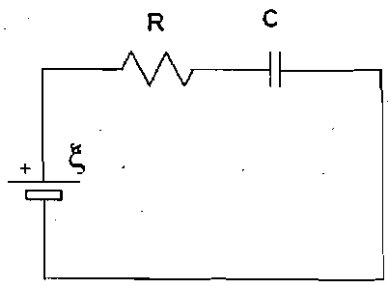
```
C:\JavaMathTools>java LESTestAll
 Square Matrix of Problem 1.-->
-2.0 1.0 0.0
1.0 -6.0 3.0
0.0 3.0 -7.0
 The constant Column Matrix of Problem 1.-->
2.0
-4.0
-2.0
 ============================
delta = -59.0
Determinant 0 = 31.999999999999993
Determinant 1 = -54.0
Determinant 2 = -40.0


The Solution of Problem by Using cramer'rule ...
-0.5423728813559321
0.9152542372881356
0.6779661016949152
```

นำไปตรวจสอบเพื่อเปรียบเทียบกับค่าแท้จริงที่หาได้จากวิธีอื่น ๆ

<u>ตัวอย่าง</u> ทดสอบโปรแกรมหารากสมการของ non-linear equation โดยวิธี bisection

วงจรไฟฟ้าประกอบด้วยตัวเก็บประจุ C ตัวต้านทาน R และแหล่งจ่ายไฟตรง ξ โวลต์ ถ้า C มีขนาด 10^{-5} ฟารัด (10 μF) แหล่งจ่ายไฟตรงมีแรงเคลื่อนไฟฟ้า 12 โวลต์ Rจะต้องมีค่าเท่าไร ตัวเก็บประจุจึงจะสามารถเก็บประจุได้ 20 %ของค่าสูงสุดในเวลา 0.01 วินาที

<u>วิธีทำ</u>



สมการแรงดันไฟฟ้าของวงจรเขียนได้ดังนี้

$$\frac{q}{C} + IR = \xi$$

แทนค่า $I = dq/dt$

$$\frac{dq}{dt} + \frac{q}{RC} = \frac{\xi}{R}$$

แก้สมการเชิงอนุพันธ์ และใช้เงื่อนไข

เมื่อ t = 0 q = 0 จะได้คำตอบของสมการนี้ คือ

$$q = \xi C(1 - e^{-t/RC})$$

กำหนดให้ $q_0 = \varepsilon C$ คือประจุค่าสูงสุดที่ตัวเก็บประจุสามารถเก็บไว้ได้ แทนค่าต่าง ๆ ลงในสมการ

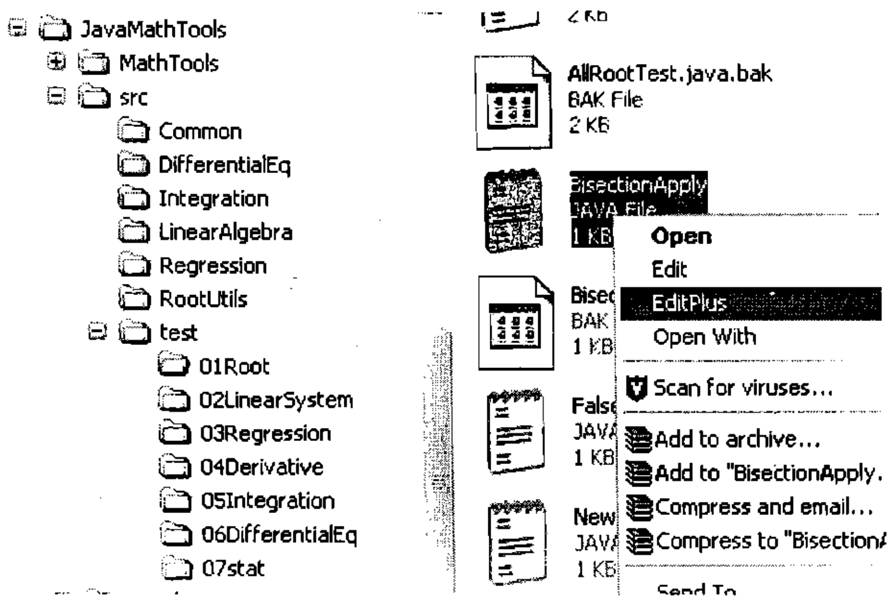
$$\frac{q}{q_0} = 0.02 = (1 - e^{-0.01/10^{-5}R})$$

จัดรูปสมการให้อยู่ในรูป f(R) = 0

$$0.98 - e^{-\frac{1000}{R}} = 0$$

วิธีทดสอบ ทำตามขั้นตอนดังนี้

1. ไปที่โฟลเดอร์ c:\JavaMathTools\src\test\01Root เปิดไฟล์ BisectionApply.java ด้วย Editplus



2. แก้ไขโปรแกรม โดยพิมพ์ฟังก์ชันที่จะหาค่ารากสมการ และใส่ค่าเริ่มต้น และค่าสิ้นสุดของรากสมการ (ค่าของรากสมการจะอยู่ระหว่างค่าทั้งสองนี้)

```
class  BisectionApply {
     public static void main(String[] args) {
          Function func = new Function() {  พิมพ์ฟังก์ชันที่นี่
          //  public double Of(double x) {r
          public double Of(double x) {return 0.98 - exp(-1000.0/x); }
          //  public double DerivativeOf(double x){ return 2*x;}
          };
          try{                                   พิมพ์ค่าเริ่มต้น และค่า
               //Bisection bs = new Bisectio สุดท้ายที่ใช้ประมาณ   on("x
               //Bisection bs = new Bisectio ค่า root              on("e
               //Bisection bs = new Bisection( TargetFunction.Function("e
               Bisection bs = new Bisection( func,1,100000);
               System.out.printf("\n Root of your equation (Bisection Tes
          }
          catch (Exception e)
          { System.out.println(" Error(s) : "+ e);
```

3. save โปรแกรม

4. ใช้คำสั่ง start >> run >> cmd

5. คอมไพล์และรันโปรแกรม ดังนี้

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\IBM>cd\

C:\>
C:\>cd JavaMathTools/src

C:\JavaMathTools\src>javac -d .. -cp .. ./test/01Root/BisectionApply.java

C:\JavaMathTools\src>cd ..

C:\JavaMathTools>java BisectionApply
```

จะได้ผลลัพธ์ดังภาพ

```
Root of your equation (Bisection Test) = 49498.490860

C:\JavaMathTools>
```

เมื่อเปรียบเทียบผลลัพธ์ที่ได้กับ Mathematica 5.1 จะได้ดังนี้

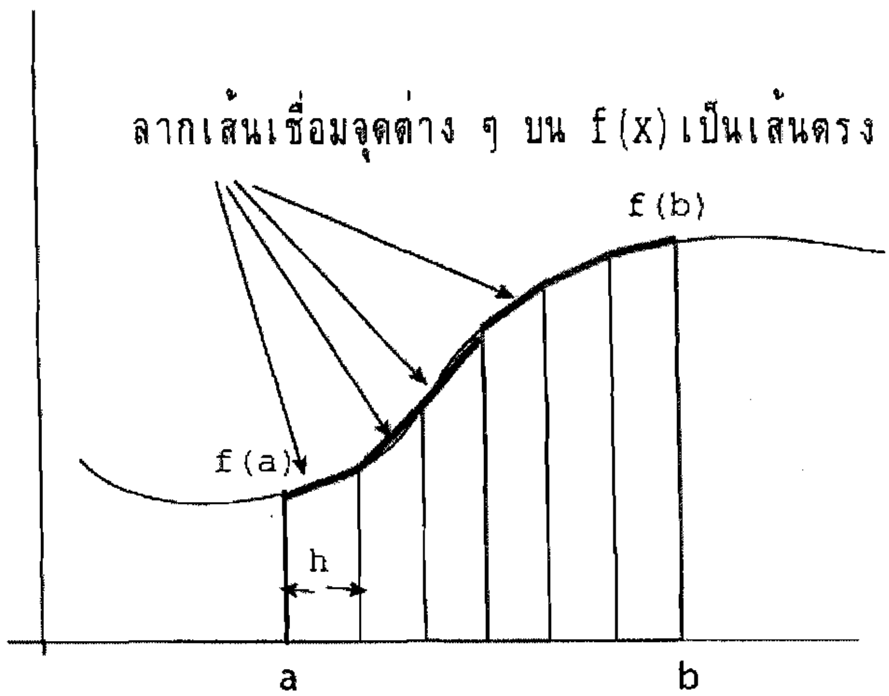
```
Untitled-1 *
In[1]:= FindRoot[Exp[-1000 / x] == 0.98, {x, 40000}]
Out[1]= {x → 49498.3}
```

การทดสอบการหาปริพันธ์ (Integration)

เป็นการหาค่า $\int_a^b f(x)dx = ?$ ต้องกำหนดค่า a (lower limit) และ b(upper limit) ค่าที่ได้จะเป็นตัวเลข โปรแกรมนี้ไม่สามารถหาค่าปริพันธ์ที่ได้ผลลัพธ์ออกมาเป็นสัญลักษณ์ได้ (เช่น $\int xdx = \frac{1}{2}x^2 + c$ เป็นต้น)

ในการพัฒนา Class library นี้จะใช้ algorithm 3 แบบ ในการหาค่าปริพันธ์ ทุกวิธีจะแบ่งพื้นที่ใต้โค้ง f(x) ให้เป็นพื้นที่เล็ก ๆ แล้วรวมพื้นที่เล็ก ๆ นั้นตั้งแต่ x =a จนถึง x = b ก็จะได้เป็นผลลัพธ์ของปริพันธ์นั้นทั้งหมด

วิธีที่ 1 เรียกว่า Trapezoid rule (กฎสี่เหลี่ยมคางหมู) เป็นการลากเส้นตรงเชื่อมพื้นที่เล็ก ๆ บนเส้นโค้ง f(x) จะได้พื้นที่สี่เหลี่ยมคางหมูเล็ก ๆ เป็นจำนวนมาก รวมพื้นที่สี่เหลี่ยมคางหมูเล็ก ๆ ทั้งหมดจะได้ผลลัพธ์ของการหาปริพันธ์



วิธีที่ 2 Simpson 1/3 rule เป็นการเปลี่ยนจากการลากเส้นตรงบนเส้นโค้ง f(x) เป็นการลากเส้นโค้งพาราโบลา ซึ่งลากเป็นเส้นโค้งนี้น่าจะได้เส้นที่ใกล้เคียงกับเส้นกราฟ f(x) มากกว่าการลากเส้นตรง แล้วรวมพื้นที่เล็ก ๆ ใต้เส้นโค้ง f(x) ทั้งหมด

วิธีที่ 3 Simpson 3/8 rule เป็นการลากเส้นโค้ง x^3 บนเส้นโค้ง f(x) แล้วรวมพื้นที่เล็ก ๆ ใต้เส้นโค้ง f(x) ทั้งหมด

วิธีที่ 4 Gauss quadrature เป็นวิธีแบ่งช่วง a, b ออกเป็นจุด โดยแต่ละจุดมีระยะห่างไม่เท่ากันเรียกว่า node จากนั้นใช้ Legendre polynomial ช่วยในการประมาณค่า

ต่อไปนี้จะเป็นการนำคลาสไลบรารีทั้งสี่วิธีคำนวณหาค่าปริพันธ์

ตัวอย่าง รถยนต์มวล 5,400 กิโลกรัม วิ่งด้วยความเร็ว 30 ม./วินาที เริ่มจับเวลาเมื่อดับเครื่องยนต์ ปล่อยให้รถเคลื่อนที่อย่างอิสระ สมการการเคลื่อนที่ของรถยนต์ หลังจากดับเครื่องแล้ว คือ

$$m\frac{vdv}{dx} = -cv^2 - R$$

เมื่อ	m	คือ มวลของรถ
	v	คือ ความเร็วที่เวลาใดๆ
	R	คือ แรงต้านทานที่มีต่อล้อขณะหมุน = 2,000 N

cv^2 คือแรงต้านทานอากาศ = 8.276 N

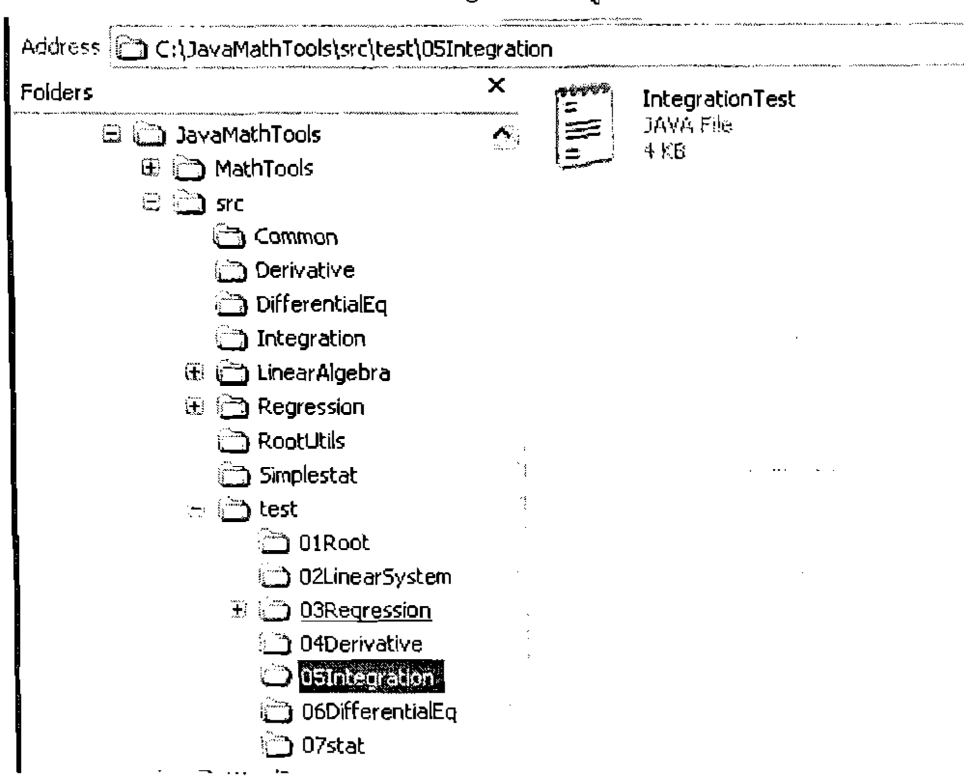
จงหาระยะทางที่รถเคลื่อนที่ได้ เมื่อความเร็วลดเหลือ 15 ม./วินาที

วิธีทำ เมื่อจัดรูปสมการใหม่จะได้

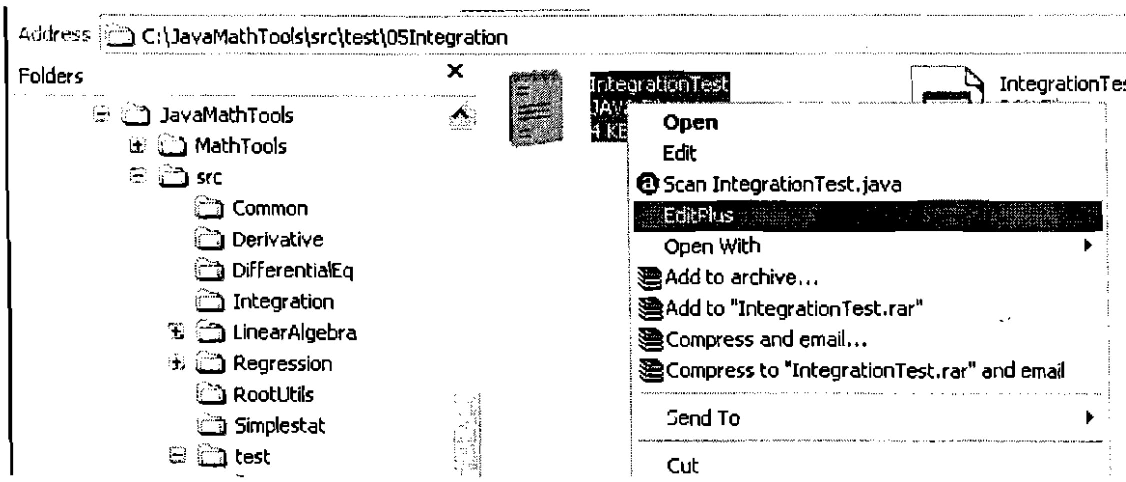
$$x = -\int_{30}^{15} \frac{5400vdv}{8.276v^2 + 2000}$$

ในที่นี้ f(v) คือ $-\dfrac{5400v}{8.276v^2 + 2000}$, lower limit (a) คือ 30 , upper limit(b) คือ 15

1. เปิดโปรแกรม IntegrationTest.java ซึ่งอยู่ในโฟลเดอร์ C:\JavaMathTools\src\test\05Integration ดังรูป



ใช้เมาส์คลิกปุ่มขวามือ เลือก editplus เพื่อทำการแก้ไขโปรแกรม



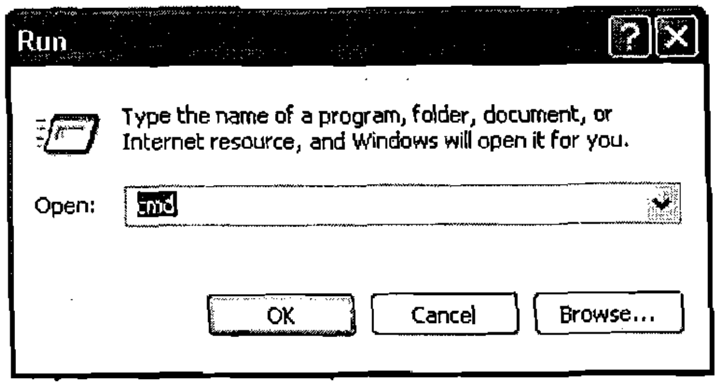
2. แก้ไขโปรแกรม IntegrationTest.java โดยการเปลี่ยนค่าฟังก์ชันให้เป็นไปตามโจทย์ในตัวอย่างที่ 1 เปลี่ยนค่า lower limit และ upper limit เป็น 30 และ 15 ตามลำดับ

```
public class IntegrationTest {

public static void main(String[] args) {
    long usedTime,start;

    // insert integrand function in here
Function func = new Function() {
public double Of(double x) {
        return (-5400.*x/(8.276*x*x+2000d));
};
    // XXXX__Integration(intergrand, lower limit,upp
    start = System.nanoTime() ;
    TrapezoidalIntegration tp = new TrapezoidalIntegration(func,30,15,1000);
        System.out.println("\n The result of Integration using Trapezoidal rule =
        usedTime = System.nanoTime() - start;
        System.out.println("\nUsed time = "+usedTime+" ns");
    //=========================================
start = System.nanoTime() ;
    Simpson1_3Integration ss1 = new Simpson1_3Integration(func,30,15,1000);
        System.out.println("\n The result of Integration using Simpson 1/3 rule =
            usedTime = System.nanoTime() - start;
        System.out.println("\nUsed time = "+usedTime+" ns");
    //=============================================
start = System.nanoTime() ;
    Simpson3_8Integration ss2 = new Simpson3_8Integration(func,30,15,1000);
        System.out.println("\n The result of Integration using Simpson 3/8 rule =
    usedTime = System.nanoTime() - start;
        System.out.println("\nUsed time = "+usedTime+" ns");
        //===============================================
    start = System.nanoTime() ;
    GaussQuadratureIntegration gq4 = new GaussQuadratureIntegration(func,30,15,8);
        System.out.println("\n The result of Integration using GaussQuadrature 8
    usedTime = System.nanoTime() - start;
    System.out.println("\nUsed time = "+usedTime+" ns");
}
}
/*
```

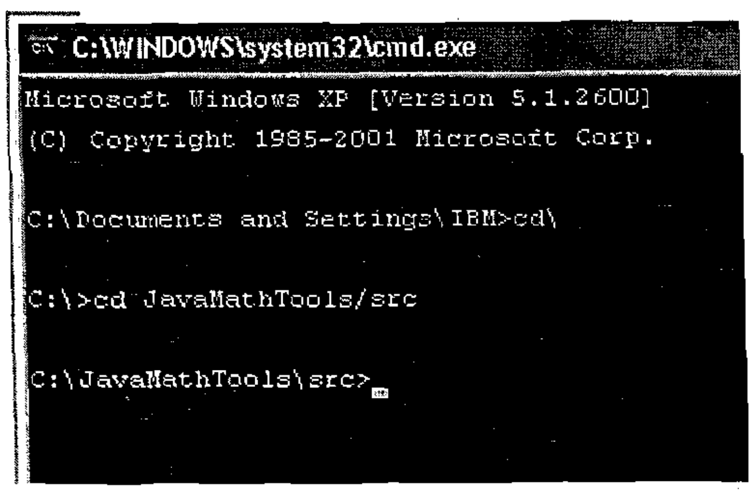
ใส่ฟังก์ชันที่ต้องการอินทิเกรตที่นี่

ใส่ lower limit และ upper limit

3. save โปรแกรมแล้วใช้คำสั่ง command เพื่อออกไปสู่ console mode



4. เปลี่ยนโฟลเดอร์ให้ไปอยู่ที่ c:\JavaMathTools/src



5. คอมไพล์โปรแกรม IntegrationTest.java โดยเรียกใช้คำสั่งดังภาพ

```
C:\JavaMathTools\src>javac -d .. -cp .. ./test/05Integration/IntegrationTest.jav
a

C:\JavaMathTools\src>
```

6. run โปรแกรม โดยใช้คำสั่งดังนี้

```
C:\JavaMathTools\src>cd ..

C:\JavaMathTools>java IntegrationTest
```

ผลลัพธ์จากการทำงานของโปรแกรมมีดังนี้

```
C:\JavaMathTools>java IntegrationTest

 The result of Integration using Trapezoidal rule =291.86958116739027

Used time = 17824332 ns

 The result of Integration using Simpson 1/3 rule =291.8695882828839

Used time = 1668089 ns

 The result of Integration using Simpson 3/8 rule =291.869588262884

Used time = 2442489 ns

 The result of Integration using GaussQuadrature  8 point =291.8695882555317

Used time = 1238147 ns

C:\JavaMathTools>
```

เปรียบเทียบกับค่าที่แท้จริง คือ

$$x = -\frac{2700}{8.276} ln(8.276v^2 + 2000)\Big|_{30}^{15}$$

$$= 291.8695883 \text{ ม.}$$

การทดสอบการหาคำตอบของสมการอนุพันธ์แบบสามัญ

สมการอนุพันธ์ที่สามารถหาคำตอบได้จะอยู่ในรูปต่อไปนี้

$$\frac{dy}{dx} = f(x, y)$$

โดยกำหนดเงื่อนไขเริ่มต้น(Initial condition) ให้ สำหรับโจทย์ที่กำหนดเงื่อนไขขอบเขต (boundary condition) ไม่สามารถนำมาหาคำตอบได้

Algorithm ที่ใช้หาคำตอบอนุพันธ์มีอยู่ 3 วิธีดังนี้

1. Modified Euler method วิธีของออยเลอร์ที่ปรับแต่งแล้ว

2. Runge – Kutta method วิธีของ รังเง คุตตา

3. Adams – Moulton method วิธีของอาดัมส์ และ มูลตัน

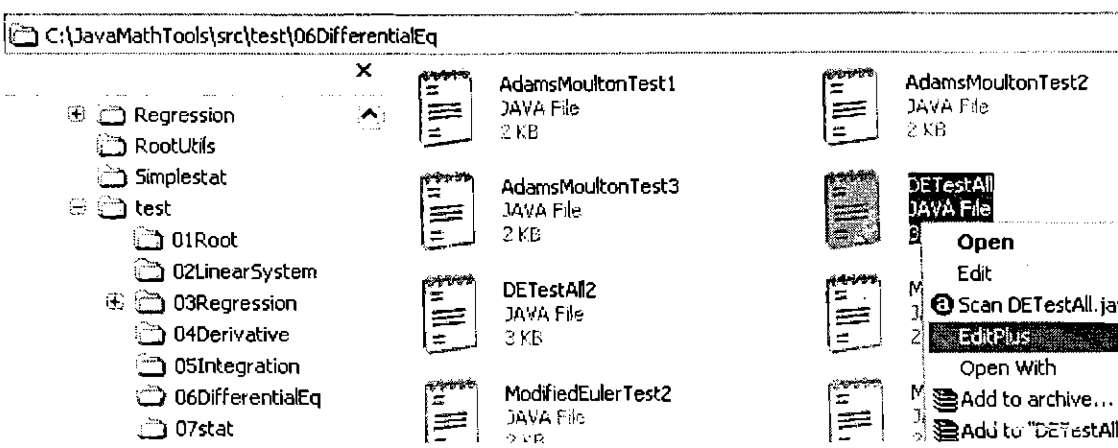
<u>ตัวอย่าง</u> จงหาคำตอบของสมการอนุพันธ์แบบ Bernoulli ที่ x = 10

$$y' = \frac{4xy + 4x\sqrt{y}}{1 + x^2}$$

มีเงื่อนไขเริ่มต้น x = 0, y = 1

<u>วิธีทำ</u>

1. เปิดไฟล์ c:\JavaMathTools\src\test\06DifferentialEq\DETestAll.java โดยใช้โปรแกรม Editplus เป็น editor



2. แก้ไขโปรแกรม โดยเปลี่ยนฟังก์ชัน ให้เป็นไปตามฟังก์ชันในตัวอย่างที่ 1 พร้อมใส่เงื่อนไขเริ่มต้น

ใส่เงื่อนไขเริ่มต้น x = 0, y = 1

```
public static void main(String[] args) {
    long executedTime,start;
    DifferentialEquation equation = new DifferentialEquation(new DataItem(0d,1.0)) {
        public double Of(double x) { return 0;}
        public double Of(double x, double y) {return (4*x*(y +  sqrt (y))/(1+x*x));}
        public double solutionOf(double x) {
            return 0;
        }
    };
```

ใส่ฟังก์ชัน f(x,y) ที่นี่

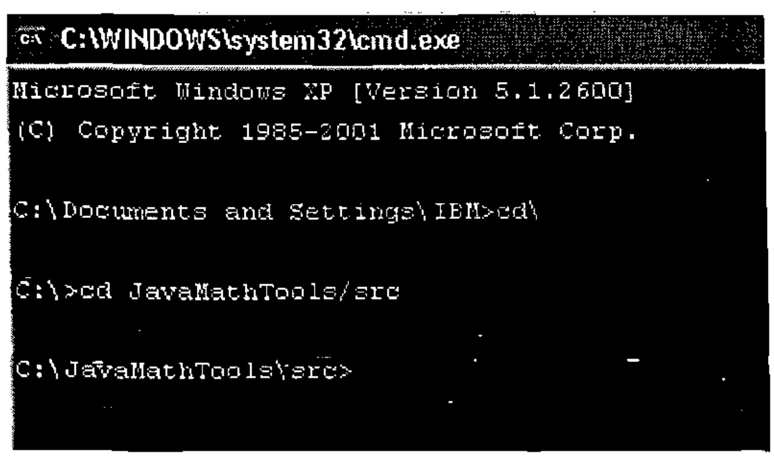
3. ต้องการหาคำตอบของสมการอนุพันธ์ ณ ตำแหน่ง x = 10

```
    ModifiedEuler me = new ModifiedEuler( equation, 0.001, 10.0);
        System.out.println("Loop count : "+ me.getCounter());
    System.out.println("by using Modified Euler Method...");
    System.out.println("\n The solution of this D.E.at x= "+me
executedTime = System.nanoTime() - start;
    System.out.println("Executed time = "+executedTime+" ns");
    System.out.println("-------------------------------------

//RungeKutta (DifferentialEquation differentialEquation, doubl
start = System.nanoTime() ;
    RungeKutta rk = new RungeKutta( equation, 0.001,10.0);
    System.out.println("Loop count : "+ rk.getCounter());
    System.out.println("by using fourth order Runge- Kutta Met
    System.out.println("\nThe solution of this D.E.at x= "+ rk.target_x + " :
    executedTime = System.nanoTime() - start;
    System.out.println("Executed time = "+executedTime+" ns");
    System.out.println("--------------------------------------------------------

//  AdamsMoulton (DifferentialEquation differentialEquation, double h, doubl
    start = System.nanoTime() ;
    AdamsMoulton am = new AdamsMoulton( equation, 0.001,10.0);
    System.out.println("Loop count : "+ am.getCounter());
    System.out.println("by using Adams - Moulton Method...");
    System.out.println("\n The solution of this D.E.at x= "+ am.target_x + "
       executedTime = System.nanoTime() - start;
    System.out.println("Executed time = "+executedTime+" ns");

}
```

ต้องการหาค่าตอบที่ x = 10

4. save โปรแกรม แล้วทำการcompile โปรแกรม โดยออกไปที่ console โดยใช้คำสั่ง start >> run >> cmd

5. ไปที่โฟลเดอร์ C:\JavaMathTools\src



6. คอมไพล์โปรแกรม โดยใช้คำสั่งดังนี้

```
C:\JavaMathTools\src>javac -d .. -cp .. ./test/06DifferentialEq/DETestAll.java

C:\JavaMathTools\src>_
```

7. run โปรแกรม โดยพิมพ์คำสั่งดังนี้

```
C:\JavaMathTools\src>
C:\JavaMathTools\src>java DETestAll
Exception in thread "main" java.lang.NoClassDefFoundError: DETestAll

C:\JavaMathTools\src>cd ..

C:\JavaMathTools>java DETestAll
```

ผลลัพธ์ที่ได้มีดังนี้

```
Loop count : 10001
by using Modified Euler Method...

 The solution of this D.E.at x= 10.0 is 40416.931335408735
Executed time = 10743544 ns
------------------------------------------------------------
Loop count : 10001
by using fourth order Runge- Kutta Method...

The solution of this D.E.at x= 10.0 is 40417.08240413235
Executed time = 9935621 ns
------------------------------------------------------------
Loop count : 9997
by using Adams - Moulton Method...

 The solution of this D.E.at x= 10.0 is 40401.00000000089
Executed time = 72846409 ns

C:\JavaMathTools>
```

เปรียบเทียบกับค่าแท้จริง

คำตอบของสมการอนุพันธ์ คือ $y = (2x^2 + 1)^2$

เมื่อ $x = 10$ จะได้ $y = 40401$

<u>ตัวอย่าง</u> สมการการเคลื่อนที่ของนักกระโดดร่มมวล m กก. มีดังนี้

$$\frac{dv}{dt} = g - \frac{c}{m}v$$

เมื่อ c คือสัมประสิทธิ์แรงต้านอากาศ = 12.5 กก/วินาที

t คือเวลาเป็นวินาที

g = 9.81 ม./วินาที^{2}

v คือ ความเร็วของนักกระโดดร่มที่เวลา t ใด ๆ มีหน่วยเป็น เมตร./วินาที

ถ้านักกระโดดร่มกระโดดจากบอลลูนที่ลอยอยู่นิ่ง จงหาความเร็วของนักกระโดดร่ม ซึ่งมีมวล(คน+ร่ม) เท่ากับ 75 กก. เมื่อเวลาผ่านไป 20 วินาที

<u>วิธีทำ</u> สมการอนุพันธ์ของโจทย์ในตัวอย่างนี้คือ $\frac{dv}{dt} = 9.81 - \frac{12.5}{75}v$

มีเงื่อนไขเริ่มต้น คือ t = 0 , v = 0

1. แก้ไขฟังก์ชันของสมการอนุพันธ์ ในที่นี้ y = v และ x = t

เงื่อนไขเริ่มต้น x หรือ t = 0 ,
y หรือ v = 0

```
public static void main(String[] a
    long executedTime,start;
    DifferentialEquation equation = new DifferentialEquation(new DataItem(0,0)) {
       public double Of(double x) { return 0 ;}
        public double Of(double x, double y) {return 9.81-12.5*y/75.0 ;}
        public double solutionOf(double x) {
            return 0;
        }
    };
```

ฟังก์ชันของสมการอนุพันธ์

2 แก้ไขตำแหน่งที่ต้องการหาคำตอบอนุพันธ์ ในที่นี้ x หรือ t = 20 วินาที

```
    ModifiedEuler me = new ModifiedEuler( equation, 0.001, 20.0);
        System.out.println("Loop count : "+ me.getCounter());
    System.out.println("by using Modified Euler Method...");
    System.out.println("\n The solution of this D.E.at x= "+me.targe
executedTime = System.nanoTime() - start;
    System.out.println("Executed time = "+executedTime+" ns");
    System.out.println("-----------------------------------------

//RungeKutta (DifferentialEquation differentialEquation, double h, d
start = System.nanoTime() ;
    RungeKutta rk = new RungeKutta( equation, 0.001, 20.0);
    System.out.println("Loop count : "+ rk.getCounter());
    System.out.println("by using fourth order Runge- Kutta Method...
    System.out.println("\nThe solution of this D.E.at x= "+ rk.targe
    executedTime = System.nanoTime() - start;
    System.out.println("Executed time = "+executedTime+" ns");
    System.out.println("-----------------------------------------

//  AdamsMoulton (DifferentialEquation differentialEquation, double
    start = System.nanoTime() ;
    AdamsMoulton am = new AdamsMoulton( equation, 0.001, 20.0);
```

3. compile และ run โปรแกรมเพื่อหาผลลัพธ์

```
C:\JavaMathTools\src>javac -d .. -cp .. ./test/06DifferentialEq/DETestAll.java

C:\JavaMathTools\src>cd..

C:\JavaMathTools>java DETestAll
```

ผลลัพธ์ที่ได้

```
--------------------------------------------------------------
Loop count : 20000
by using Modified Euler Method...

 The solution of this D.E.at x= 20.0 is 56.76022871917295
Executed time = 12923710 ns
--------------------------------------------------------------
Loop count : 20000
by using fourth order Runge- Kutta Method...

The solution of this D.E.at x= 20.0 is 56.76022875158083
Executed time = 14386186 ns
--------------------------------------------------------------
Loop count : 19997
by using Adams - Moulton Method...

 The solution of this D.E.at x= 20.0 is 56.76022875158083
Executed time = 9499532 ns
```

ตรวจสอบคำตอบด้วยวิธีวิเคราะห์ สามารถหาความเร็วของนักกระโดดร่มโดยหาคำตอบของสมการอนุพันธ์โดยวิธีแยกตัวแปร

$$\int \frac{dv}{g - \frac{c}{m}v} = \int dt$$

จากเงื่อนไข $t = 0$ $v = 0$ จะได้

$$v = gm\left(1 - e^{-(c/m)t}\right)/c$$

แทนค่า $t = 20$ วินาที จะได้ $v = 56.70348$ เมตร /วินาที

การทดสอบโปรแกรม Regression

จะแบ่งการทดสอบออกเป็น 2 ส่วน ส่วนที่ 1 จะเป็น Linear Regression ส่วนที่ 2 จะเป็น Polynomial Regression

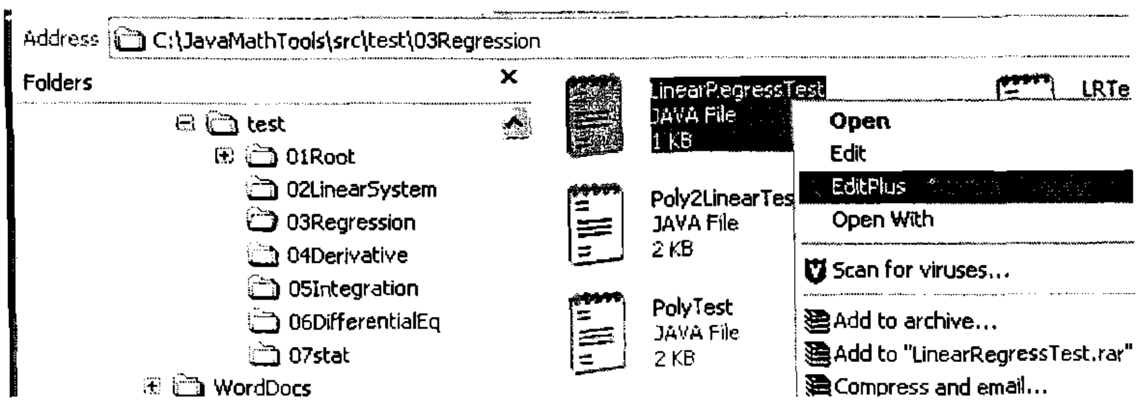
<u>Linear Regression</u> : เรามีข้อมูลจากการทดลองเขียนเป็นตารางได้ดังนี้

x	0	1	2	3	4	5
y	3	5	7	9	11	13

ต้องการจะหาความสัมพันธ์เชิงเส้นของตัวแปร x และ y ในรูป y = mx + c เมื่อ m คือ slope และ c คือ y-intercept

ขั้นตอนการทดสอบ

1.ไปที่โฟลเดอร์ C:\JavaMathTools\src\test\03Regression คลิกขวาที่ไฟล์ LinearRegressTest.java เลือกที่เมนู editplus



1. เพิ่มข้อมูล x และ y ดังรูป

```
class  LinearRegressTest
{

    public static void main(String[] args)
    {
//      double x1[] = {2.};
//      double y1[] = {5.};                       ข้อมูล x
    double[] x1 = {0,1.,2.,3.,4.,5.};
    double[] y1 = {3., 5., 7., 9., 11., 13.};          ข้อมูล y

    double m ;  // y = mx + c
    double c;
    try{
    LinearRegression lr = new LinearRegression(x1, y1);
                m       = lr.getSlope();
                c       = lr.getYIntercept();
                System.out.println( "The line equation of this data is  Y  = "+m+" X + "+c);
    }catch ( RegressionException msg){ System.out.println(msg);}
 }

 }
```

3. save program จากนั้นทำการcompile

```
C:\JavaMathTools\src>javac -d .. -cp .. ./test/03Regression/LinearRegressTest.ja
va

C:\JavaMathTools\src>
```

4. run โปรแกรม จะได้ผลลัพธ์ดังนี้

```
C:\JavaMathTools\src>cd ..

C:\JavaMathTools>java LinearRegressTest
The line equation of this data is  Y  = 2.0 X + 3.0
```

Polynomial Regression เป็นการหาความสัมพันธ์ระหว่างข้อมูล x, y ที่อยู่ในรูปโพลีโนเมียล หรือเลขยกกำลังของตัวแปร x มีค่ามากกว่า 1

ข้อมูล n ชุด มีความสัมพันธ์กันแบบพหุนามอันดับ m เขียนเป็นสมการได้ดังนี้

$$y = a_0 + a_1x + a_2x^2 + \ldots\ldots a_mx^m$$

ตัวอย่าง ก้อนหินเคลื่อนที่แบบโปรเจกไตล์ ความสัมพันธ์ระหว่างความสูงซึ่งวัดจากพื้นดิน (y) และระยะทางที่เคลื่อนที่ได้ในแนวราบ (x) มีดังนี้

x	0	1	2	4	5	6	7	8
y	0	0.5121	0.8936	1.2648	1.2545	1.1136	0.8421	0.4400

ข้อมูลชุดนี้ได้จากโจทย์กล่าวถึงการขว้างก้อนหินเป็นมุม 30° กับพื้นโลกด้วยความเร็วต้น 10 ม./วินาที ไม่คิดแรงต้านของอากาศ จะได้สมการแสดงตำแหน่งต่าง ๆ ของก้อนหินเป็น

$$y = 0.5774x - 0.0653x^2$$

จะใช้ข้อมูล x, y ป้อนให้โปรแกรม PolyRegressTest.java คำนวณหา สัมประสิทธิ์ของ x ว่าจะได้เท่ากับค่าสัมประสิทธิ์ที่ได้จากค่าทฤษฎีหรือไม่

ขั้นตอนการทดสอบ Polynomial Regression

1.ไปที่โฟลเดอร์ C:\JavaMathTools\src\test\03Regression คลิกขวาที่ไฟล์ PolyRegressTest.java เลือกที่เมนู editplus

2. แก้ไขโปรแกรม กำหนดค่า degree = 2 (เพราะโพลีโนเมียลที่เราจะหาความสัมพันธ์ x ยกกำลังไม่เกิน 2 และเพิ่มข้อมูล x และ y ตรงบริเวณลูกศรชี้

กำหนด degree = 2

```
class  PolyRegressTest {

   public static void main(String[] args){
       int degree = 2;
       ColumnMatrix a = new ColumnMatrix();
/*
// first style: this style works well.

        PolynomialRegression pr = new PolynomialRegression(degree,8);
        pr.addData(new DataItem(0., 0.));
        pr.addData(new DataItem(1., 0.5121));
        pr.addData(new DataItem(2., 0.8936));
        pr.addData(new DataItem(4., 1.2648));
        pr.addData(new Dat
        pr.addData(new Dat
        pr.addData(new Dat
        pr.addData(new Dat
```

ใส่ข้อมูล x และ y ที่นี่

```
/* second style also works well */
    double[] x = { 0, 1., 2., 4., 5., 6., 7., 8.};
    double[] y = {0, 0.5121, 0.8936, 1.2648, 1.2545, 1.1136, 0.8421, 0.44};

        PolynomialRegression pr = new PolynomialRegression(degree , x, y);
```

3. save โปรแกรม และทำการคอมไพล์

```
C:\JavaMathTools>
C:\JavaMathTools>cd src

C:\JavaMathTools\src>javac -d .. -cp .. ./test/03Regression/PolyRegressTest.java
```

4. ทดสอบให้โปรแกรมทำงาน

```
C:\JavaMathTools\src>cd..

C:\JavaMathTools>java PolyRegressTest
Number of data = 8
y = 2.9150471210670776E-15  + 0.5773999999999974x  + -0.06529999999999968 x^2
C:\JavaMathTools>
```

จะเห็นว่าใกล้เคียงกับค่าแท้จริง เทอมแรกมีค่าน้อยมาก 10^{-15} สามารถปัดให้เป็นศูนย์ได้

ถ้าเราใช้ข้อมูลจากตัวอย่างในเรื่อง Linear regression มาป้อนให้โปรแกรม Polynomial Regression โดยเปลี่ยน degree = 1 จะพบว่าจะได้คำตอบเหมือนกัน ดังนั้น Polynomial Regression จึงครอบคลุม Linear Regression ด้วย

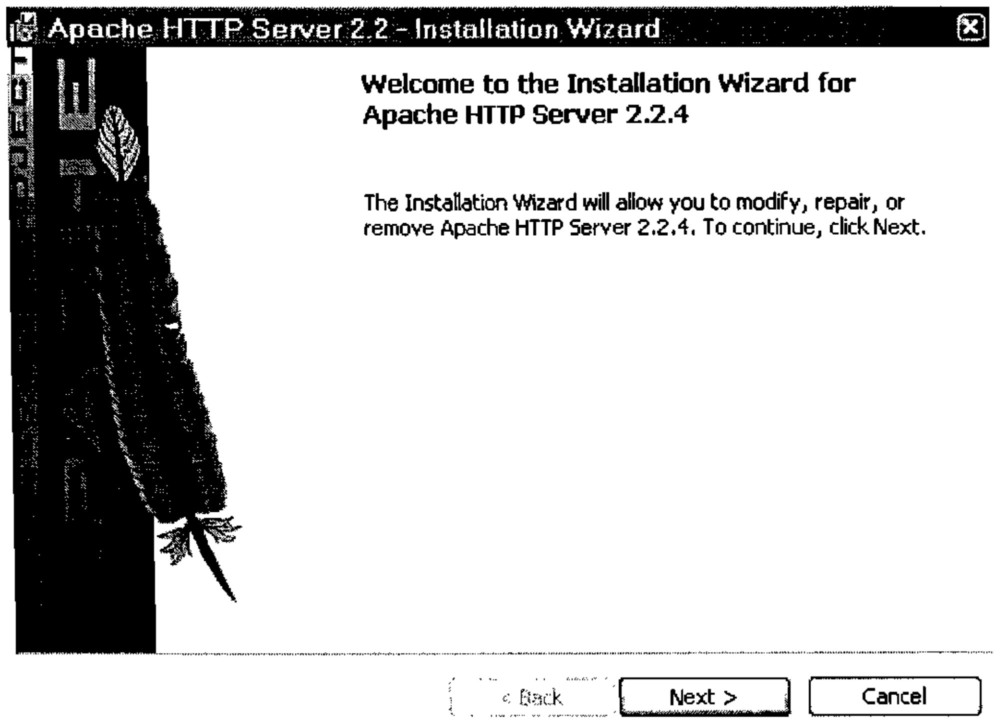
ภาคผนวก 2
การติดตั้ง Web server เพื่อทดสอบการทำงานของคลาสไลบรารี

การทดสอบการทำงานของคลาสไลบรารีทางคณิตศาสตร์ที่สร้างขึ้นมาผ่านทางอินเทอร์เน็ต ทำได้โดยจัดทำเป็น applet ฝังไว้ในเอกสาร Html แล้วจัดเก็บไว้ใน web server ผู้ใช้งานสามารถเรียกใช้ applet นั้นผ่านทางบราวเซอร์และ applet จะแสดงผลภายในพื้นที่ของบราวเซอร์ ในการทดสอบนั้นในเบื้องต้น เราสามารถจำลองคอมพิวเตอร์ที่เราใช้งานอยู่ทำหน้าที่เป็น web server ได้ในชื่อของ Localhost ซึ่งมีเลข IP เป็น 127.0.0.1 ก่อนอื่นจะต้องติดตั้งโปรแกรมที่ทำหน้าที่เป็น web server เสียก่อน ในที่นี้จะใช้ Apache version 2.2

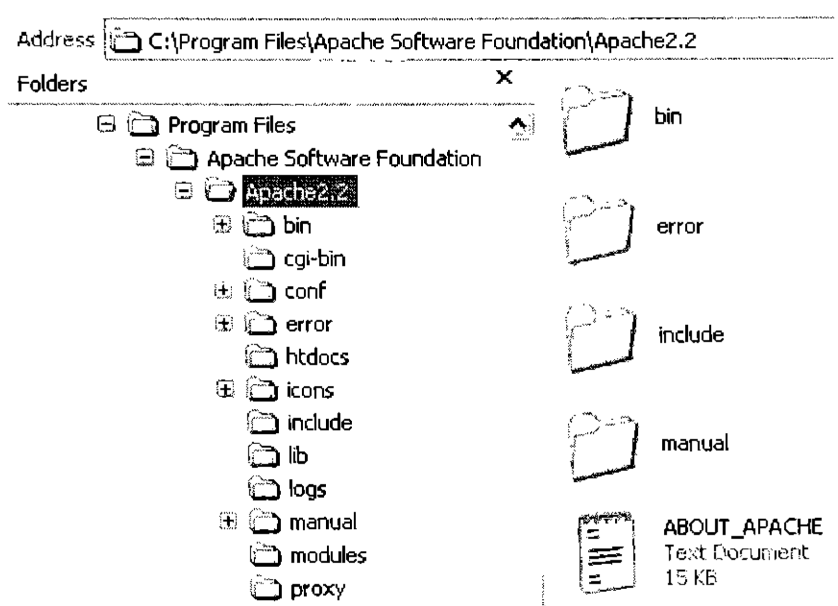
ขั้นที่ 1 ติดตั้ง Web Server

ชื่อเต็มของโปรแกรมคือ Apache HTTP Server 2.2 สามารถดาวน์โหลด ได้จาก http://apache.org ขนาดของไฟล์ประมาณ 4.3 Mb

ให้ดำเนินการติดตั้งไปตามลำดับเหมือนกับโปรแกรมที่ใช้งานในวินโดว์ ทั่วไป ๆ

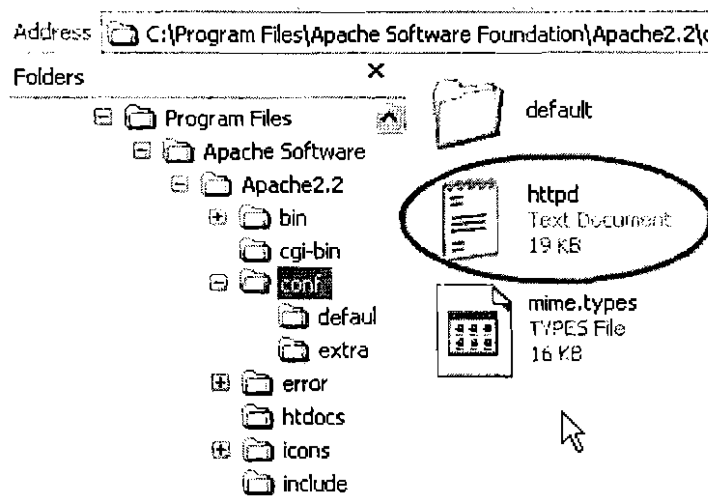


เมื่อติดตั้งเสร็จแล้ว จะได้โฟลเดอร์ที่จัดเก็บโปรแกรม Apache web server ดังรูป



ก่อนใช้งาน Apache เป็นเว็บ เซิร์ฟเวอร์ ต้องแก้ไขไฟล์ configuration ให้สอดคล้องกับงานทดลองเสียก่อน ในการทดสอบนี้จะเก็บเอกสาร html และ คลาสไฟล์ทั้งหลาย ไว้ภายใต้โฟลเดอร์ C:\www\jmt การแก้ไขทำได้ดังนี้

ไปที่โฟลเดอร์ conf ซึ่งเป็นโฟลเดอร์ย่อยภายใต้ โฟลเดอร์ Apache2.2 อีกที่หนึ่ง



แก้ไขไฟล์ httpd.conf โดยใช้เอดิเตอร์ (อาจเป็น notepad) แก้ไขคำสั่งในไฟล์บางบรรทัดดังนี้

1. แก้ไข SeverName ให้เป็น 127.0.0.1:80 เลข 80 ที่ต่อท้ายหมายถึงติดต่อโดยผ่านพอร์ตหมายเลข 80

```
# If your host doesn't have a registe
#
ServerName 127.0.0.1:80

#
```

2. แก้ไขโฟลเดอร์ที่จะใช้เก็บเอกสาร html และ ไฟล์คลาสต่าง ๆ

```
#DocumentRoot "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs"
DocumentRoot "C:/WWW"
#
```

3. แก้ไขเงื่อนไขต่าง ๆ ให้ตรงกับโฟลเดอร์ WWW ที่เป็นโฟลเดอร์เก็บเอกสาร html

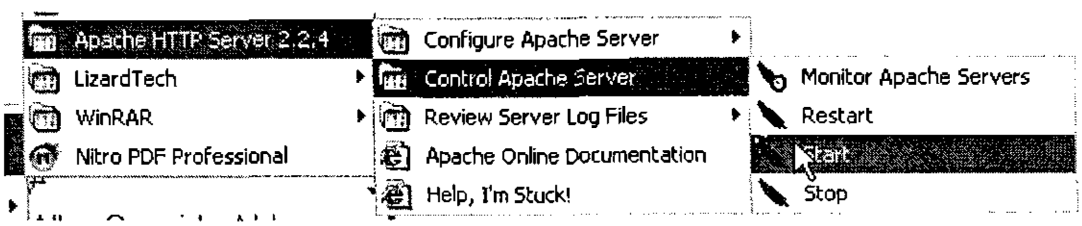
```
#<Directory "C:/Program Files/Apache Softwar
<Directory "C:/WWW">
 #
```

และแก้ไขบรรทัดนี้ด้วย

```
#
# AllowOverride controls what directives may be place
# It can be "All", "None", or any combination of the ke
#   Options FileInfo AuthConfig Limit
#
AllowOverride ALL
#
```

จากนั้น save ไฟล์ที่แก้ไขแล้วนี้ทับของเดิม

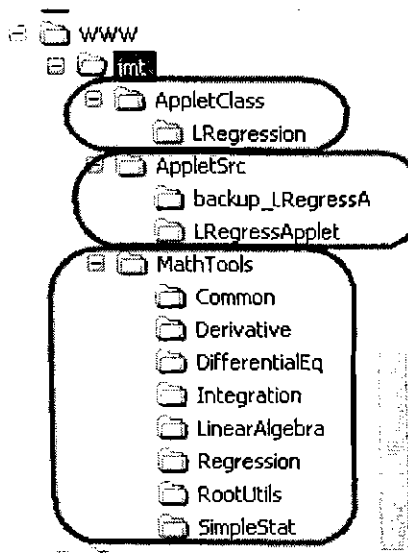
4. start ให้โปรแกรม apache ทำงาน



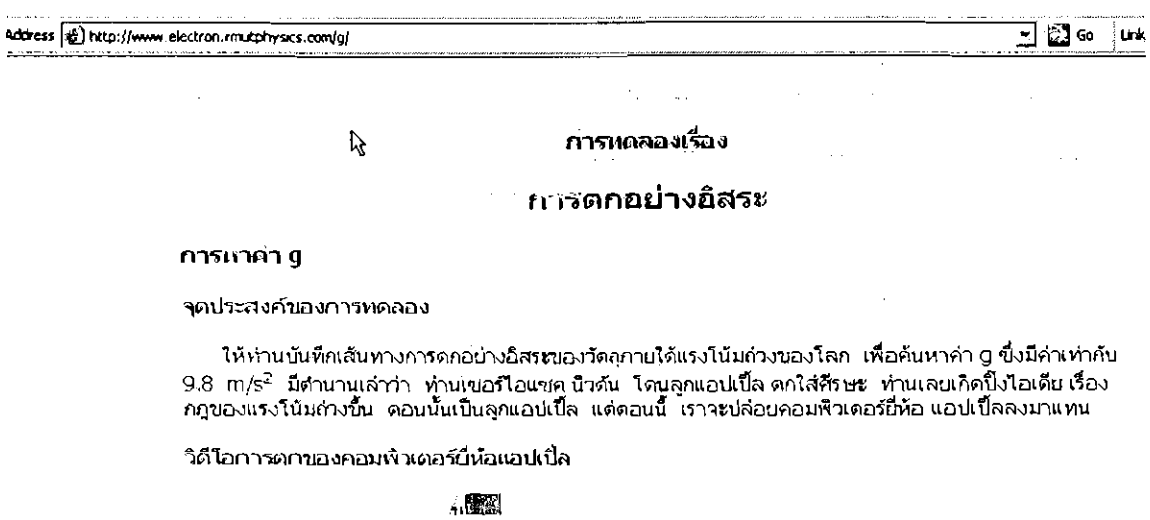
5. ถ้าไม่มีข้อผิดพลาดอันใด เราจะเห็น icon ของโปรแกรม Apache ฝังอยู่ที่ แถบมุมขวาล่างของจอภาพ ตรงกึ่งกลางวงกลมสีขาว จะเห็นมีสีเขียวอยู่ตรงจุดกึ่งกลาง แสดงว่า Apache ทำงานได้อย่างถูกต้อง

ขั้นที่ 2 นำผลงานไปติดตั้งไว้ที่ Web server

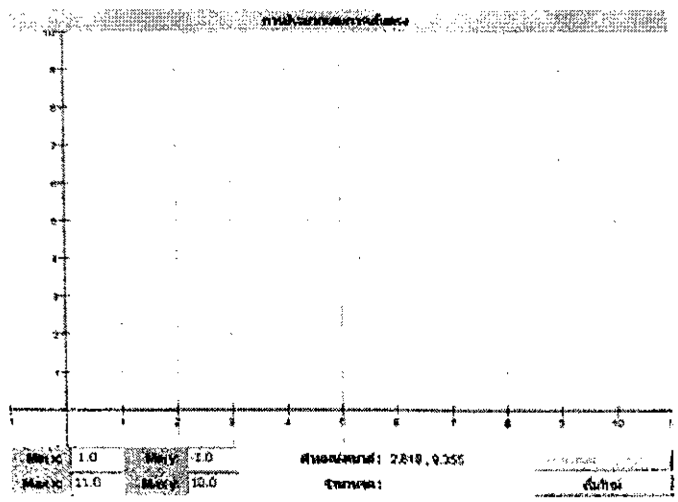
นำผลงานวิจัยที่ได้ทั้งหมดที่อยู่ในรูปคลาสไฟล์ และ เอกสาร Html ไปเก็บไว้ในโฟลเดอร์ c:\www ลักษณะของโฟลเดอร์จะมีการจัดเก็บดังนี้



ตัวอย่างต่อไปนี้ผลที่ได้จากการทดสอบโปรแกรมผ่าน web server เป็นการนำคลาส Regression ไปใช้ในการประมาณค่าสมการเส้นตรง โดยอาศัยข้อมูลที่ได้จากการทดลองเรื่องการตกอย่างอิสระ

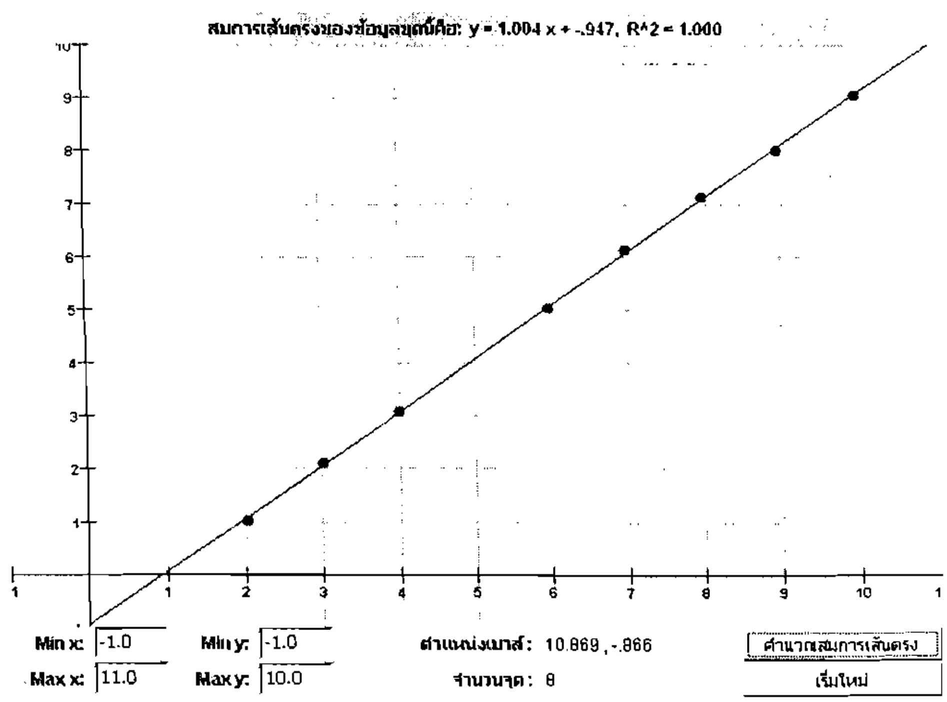


เขียนโปรแกรมตรงส่วนที่ติดต่อกับผู้ใช้ (Graphic user interface) เพื่อให้ผู้ใช้ป้อนข้อมูลโดยการคลิกเมาส์บนกระดาษกราฟบนจอภาพ โปรแกรมจะเก็บข้อมูลที่ได้ตามตำแหน่ง (x, y) ณ ตรงจุดที่เมาส์คลิก แล้วนำไปประมวลผลโดยใช้คลาส LinearRegression จะได้สมการเส้นตรงที่เป็นตัวแทนของข้อมูล รูปร่างของกระดาษกราฟจะมีลักษณะดังภาพต่อไปนี้



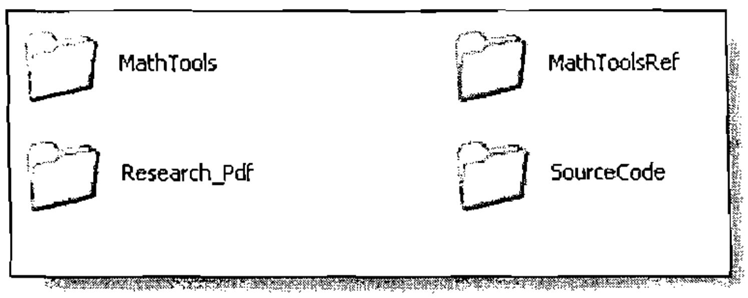
การวาดกราฟระหว่าง y กับ t^2 ให้ใช้ Applet ดังรูป ทำแทนกราฟในหนังสือก็ได้ และให้นำความชันที่ได้ไปคำนวณหาค่าความเร่งโน้มถ่วง

เมื่อคลิ้กเมาส์ตามตำแหน่งต่าง ๆ บนกระดาษกราฟแล้ว จากนั้นคลิกปุ่ม "คำนวณสมการเส้นตรง" โปรแกรมจะสร้างเส้นตรง โดยอาศัยการคำนวณของคลาส LinearRegression ดังภาพ



ภาคผนวก 3
เอกสารในแผ่น CD

เมื่อเปิดแผ่น CD ที่แนบมากับรายงานผลโครงการวิจัยสิ่งประดิษฐ์ฉบับนี้ จะพบโฟลเดอร์ดังต่อไปนี้



1. โฟลเดอร์ MathTools จัดเก็บ class file ที่ได้จากการคอมไพล์โปรแกรมต้นฉบับภาษาจาวาทั้งหมดของโครงการวิจัยนี้ สามารถนำไปใช้ประกอบกับโปรแกรมบทเรียนต่าง ๆ ได้ทันที เพียงแค่นำโฟลเดอร์นี้ไปเก็บไว้ในเครื่องแม่ข่ายที่ให้บริการอินเทอร์เน็ตเท่านั้น

2. โฟลเดอร์ MathToolsRef เป็น Math Tool Quick Reference ที่สร้างจากโปรแกรม Javadoc เอกสารเหล่านี้สามารถเปิดใช้งานผ่าน Browser โดยคลิกที่ index.html จะมี link เชื่อมต่อให้สืบค้นหรืออ้างอิงคลาสต่าง ๆ ที่สร้างขึ้นมาในโครงการวิจัยนี้ ในเอกสารนี้จะแสดงคลาสต่าง ๆ ในลักษณะกิ่งไม้ (Hierarchy diagram) โดยจะบอกคุณลักษณะของคลาสและ interface หน้าที่ของคลาส package ต่าง ๆ ที่คลาสมีการเรียกใช้ การดักจับข้อผิดพลาด (Exception) ของคลาส ตัวแปรคลาส และเมธอดทั้ง public protected และ private ที่ปรากฏในคลาสนั้น การส่งผ่านค่าต่าง ๆ ผ่าน parameter ในเมธอด การส่งค่าคืนกลับ (return) ของเมธอด รวมทั้ง inner class ที่มีอยู่ในคลาสเหล่านั้น

3. โฟลเดอร์ Research_pdf ในโฟลเดอร์นี้เป็นรายงานผลโครงการวิจัย "พัฒนาคลาสไลบรารี (Class library) เกี่ยวกับการคำนวณเชิงตัวเลขสำหรับสร้างบทเรียนทางฟิสิกส์ที่เรียนรู้ผ่านเครือข่ายอินเทอร์เน็ต" ฉบับที่ท่านกำลังถืออยู่นี้ทั้งเล่ม จัดเก็บในรูป pdf สามารถเปิดอ่านได้โดยอาศัยโปรแกรม Acrobat reader *ฉบับที่บรรจุใน CD จะมีภาคผนวก 4 ซึ่งเป็นการรวบรวมตัวอย่างปัญหาทางคณิตศาสตร์และฟิสิกส์ที่ใช้ทดสอบการทำงานของโปรแกรม จำนวน 40 หน้า* รวมเข้าไป

ด้วย ภาคผนวก 4 จะไม่ปรากฏในเล่มที่เป็นรายงานผลโครงการวิจัยที่เป็นเล่ม ทั้งนี้เพื่อไม่ให้รูปเล่มของรายงานมีขนาดหนาเกินไป

4. โฟลเดอร์ SourceCode จะเก็บโปรแกรมต้นฉบับทั้งหมดที่พัฒนาขึ้นมาในโครงการวิจัยนี้ โปรแกรมต้นฉบับที่ปรากฏอยู่ในรายงานผลการวิจัยนี้ได้หยิบมาเฉพาะบางส่วนที่เกี่ยวข้องกับทฤษฎีที่กำลังอธิบายอยู่เท่านั้น ไม่ได้นำมาหมดทั้งตัวโปรแกรม ผู้สนใจสามารถดูโปรแกรมฉบับเต็มทุกโปรแกรมรวมทั้งโปรแกรมตรงส่วนที่เป็น Graphic User Interface ได้จากโฟลเดอร์นี้

ต้องการข้อมูลเพิ่มเติมหรือต้องการโปรแกรมที่มีการปรับปรุงแก้ไขล่าสุด สามารถ สืบค้น และ download ได้จาก เว็บไซต์ http://204.158.100.140/jmt รวมทั้งข้อคิดเห็น ข้อเสนอแนะต่าง ๆ คณะผู้วิจัยยินดีที่จะรับฟังด้วยความเต็มใจยิ่ง

INDEX