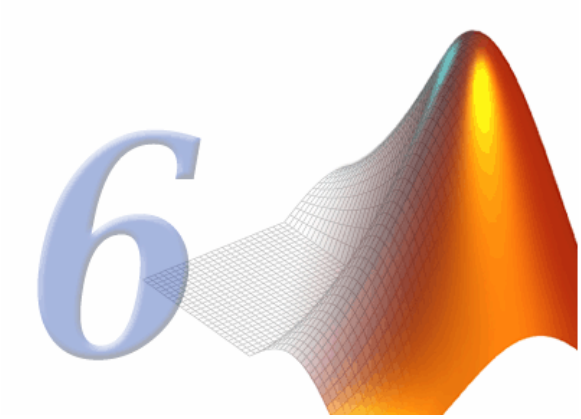
การใช้

MATLAB®

สำหรับงานทางวิศวกรรม



ฉบับปรับปรุงครั้งที่ 2

โศรฎา แข็งการ
กนต์ธร ชำนิประศาสน์

ข้าขอประณตน้อมสักการ
บูรพคณาจารย์
ผู้ก่อเกิดประโยชน์ศึกษา

คำนำ

เอกสารการใช้ MATLAB สำหรับงานทางวิศวกรรมเล่มนี้ นับเป็นการปรับปรุงใหญ่ครั้งแรก ตั้งแต่ฉบับแรกเมื่อประมาณ 3 ปีก่อน และเราได้ใช้ในการเรียนการสอนของนักศึกษาวิศวกรรมเครื่องกล และนักศึกษาอื่นๆ ที่สนใจ โดยในระหว่างการใช้เอกสารในชุดแรกเราก็ได้มีการปรับปรุงแก้ไขเนื้อหาต่างๆ มาตลอด และสิ่งที่เราพยายามเพิ่มเติมเข้าไปในเนื้อหาก็คือการเขียน GUI ซึ่งเป็นส่วนหนึ่งที่นักศึกษาได้ให้ความสนใจในการสร้างโปรแกรมประเภทนี้มากขึ้น เพราะผลงานที่ออกมาจะสามารถทำให้ผู้ใช้เกิดความสะดวกในการใช้งานมากกว่าโปรแกรมประเภท text ต่างๆ มาก และอีกส่วนหนึ่งก็คือมีการปรับปรุงครั้งใหญ่ของ MATLAB ไปแล้วถึงสองครั้ง นั่นคือปรับปรุง MATLAB 5.3 ซึ่งถือว่าเป็น Release 11 ของ MATLAB และ MATLAB 6.0 ซึ่งถือว่าเป็น Release 12 ของ MATLAB ดังนั้นการแก้ไขก็คงจะมีมากพอสมควร

ในความเป็นจริงเอกสารที่เราปรับปรุงครั้งแรกและส่วนที่มี GUI นั้นมีความแตกต่างจากเอกสารชุดนี้พอสมควร เพราะเมื่อ MATLAB 6.0 ออกมาสิ่งที่นับว่าเปลี่ยนแปลงมากที่สุดสิ่งหนึ่งก็คือ GUIDE ซึ่งเป็นการช่วยการสร้าง GUI โดยมีการเปลี่ยนแปลงรูปแบบและลักษณะการทำงานออกไป เรียกได้ว่าแทบจะทั้งหมด ดังนั้นเราจึงต้องจัดการปรับปรุงบทที่กล่าวถึง GUI และ GUIDE ใหม่ทำให้เอกสารชุดนี้ปรากฏออกมาล่าช้าไปบ้าง

สิ่งที่เปลี่ยนแปลงของเอกสารชุดนี้จากที่ออกมาฉบับแรกที่สำคัญก็คือเราได้ปรับปรุงและแก้ไขบทที่ 1 ค่อนข้างมาก เพื่อให้ผู้ใช้เข้าใจการทำงานของ MATLAB ได้ง่ายขึ้น อธิบายถึงหน้าต่างที่มีเพิ่มขึ้นจาก MATLAB รุ่นก่อนๆหรือที่เรียกว่า MALLAB Desktop ซึ่งทำให้ผู้ใช้สะดวกกับการใช้งานมากขึ้น เราได้ยุบบทที่กล่าวถึงประเภทของตัวแปร Cell Array และ Structure มารวมไว้ในบทที่ 2 เพราะเราถือว่าเป็นข้อมูลเบื้องต้นที่เราควรจะทราบ และเรื่องนี้ก็ไม่ใช่ของใหม่แล้วเพราะตัวแปรประเภทนี้ได้มีการนำเสนอครั้งแรกตั้งแต่ใน version 5.0

สำหรับบทที่เขียนขึ้นมาใหม่ทั้งหมดเลยก็คือบทที่ 12 ถึงบทที่ 15 ซึ่งเป็นบทที่มีความเกี่ยวข้องกับการเขียน GUI บน MATLAB หรือที่เรามักจะเรียกกันเล่นๆ ว่า Visual MATLAB เพราะการจะเขียนโปรแกรมประเภทนี้ได้ต้องเข้าใจถึง Graphic ของ MATLAB ดีพอสมควร โดยบทที่ 12 จะอธิบายถึงวิธีการกำหนดวัตถุหรือ object ใน MATLAB บทที่ 13 เรากล่าวถึงคุณสมบัติของ Object ประเภทต่างๆ บทที่ 14 เราแนะนำให้รู้จัก GUIDE เครื่องมือที่จะช่วยในการเขียน GUI ได้อย่างสะดวก และบทที่ 15 จะเป็นตัวอย่างการเขียน GUI เบื้องต้นเพื่อให้ผู้อ่านเกิดความเข้าใจ แล้วสามารถนำไปพัฒนาโปรแกรมที่ตนเองต้องการได้

เราขอขอบคุณ เพื่อน พี่และน้องในสาขาวิชาวิศวกรรมเครื่องกล มหาวิทยาลัยเทคโนโลยีสุรนารี ที่ได้ให้ความช่วยเหลือและให้คำปรึกษาที่เป็นประโยชน์ต่อการจัดทำเอกสารฉบับนี้ และท้ายที่สุดขอขอบพระคุณ รองศาสตราจารย์นาวาอากาศเอก ดร. วรพจน์ ขำพิศ ที่เป็นทั้ง ครู เพื่อนร่วมงาน และผู้บังคับบัญชาของเราทั้งสองคนที่คอยเอาใจใส่ ให้คำเสนอแนะและช่วยเหลือในการเรียบเรียงเอกสารฉบับนี้อย่างดียิ่งเสมอมา

โศรฎา แข็งการ
กนต์ธร ชำนิประศาสน์
มหาวิทยาลัยเทคโนโลยีสุรนารี

สารบัญ

บทที่ 1 ทำความรู้จักกับ MATLAB

1.1 กล่าวนำ

MATLAB เป็นโปรแกรมคอมพิวเตอร์สมรรถนะสูงเพื่อใช้ในการคำนวณทางเทคนิค MATLAB ได้รวมการคำนวณ การเขียนโปรแกรมและการแสดงผลรวมกันอยู่ในตัวโปรแกรมเดียวได้อย่างมีประสิทธิภาพ และอยู่ในลักษณะที่ง่ายต่อการใช้งาน นอกจากนี้ลักษณะของการเขียนสมการในโปรแกรมก็จะเหมือนการเขียนสมการคณิตศาสตร์ที่เราคุ้นเคยดีอยู่แล้ว งานที่ทั่วไปที่ใช้ MATLAB ก็เช่น การคำการคำนวณทั่วไป การสร้างแบบจำลองและการทดสอบแบบจำลอง การวิเคราะห์ข้อมูล การแสดงผลในรูปกราฟทั้งโดยทั่วไปและกราฟทางด้านทางวิทยาศาสตร์และวิศวกรรม สามารถสร้างโปรแกรมในลักษณะที่ติดต่อกับผู้ใช้ทางกราฟฟิกส์

การทำงานของ MATLAB จะสามารถทำงานได้ทั้งในลักษณะของการติดต่อโดยตรง(Interactive) คือการเขียนคำสั่งเข้าไปทีละคำสั่ง เพื่อให้ MATLAB ประมวลผลไปเรื่อยๆ หรือสามารถที่จะรวบรวมชุดคำสั่งเรานั้นเป็นโปรแกรมก็ได้ ข้อสำคัญอย่างหนึ่งของ MATLAB ก็คือข้อมูลทุกตัวจะถูกเก็บในลักษณะของ array คือในแต่ละตัวแปรจะได้รับการแบ่งเป็นส่วนย่อยเล็กๆขึ้น (หรือจะได้รับการแบ่งเป็น element นั่นเอง) ซึ่งการใช้ตัวแปรเป็น array ในMATLAB นี้เราไม่จำเป็นที่จะต้องจอง dimension เหมือนกับการเขียนโปรแกรมในภาษาขั้นต่ำตัวไป ซึ่งทำให้เราสามารถที่จะแก้ปัญหาของตัวแปรที่อยู่ในลักษณะของ matrix และ vector ได้โดยง่าย ซึ่งทำให้เราลดเวลาการทำงานลงได้อย่างมากเมื่อเทียบกับการเขียนโปรแกรมโดย C หรือ Fortran

MATLAB เป็นโปรแกรมสำเร็จรูปที่ใช้กันอย่างแพร่หลายในแวดวงของนักวิทยาศาสตร์และวิศวกรในปัจจุบัน ชื่อโปรแกรม MATLAB นั้นย่อมาจาก MATrix LABoratory โดย MATLAB นั้นได้เริ่มต้นขึ้นเพื่อต้องการให้เราสามารถแก้ปัญหาตัวแปรที่มีลักษณะเป็น Matrix ได้ง่ายขึ้น สำหรับ MATLAB ได้เริ่มพัฒนาครั้งแรกโดย Dr. Cleve Molor ซึ่งเขียนโปรแกรมนี้ขึ้นมาด้วยภาษา Fortran โดยโปรแกรมนี้ได้พัฒนาภายใต้โครงการ LINPACK และ EISPACK

สำหรับในปัจจุบันนี้ MATLAB ได้ถูกเขียนขึ้นโดยใช้ภาษา C โดยบริษัท MathWorks ภายใต้โครงการ LAPACK และ ARPACK ถ้าหากเราจะเริ่มนับจากโปรแกรมที่ออกเผยแพร่เป็นครั้งแรกที่มีผู้ร่วมเขียนโปรแกรมไม่กี่คน จนกระทั่งทุกวันนี้มีทีมงานขนาดใหญ่ที่ทำงานในการพัฒนาโปรแกรมให้มีประสิทธิภาพสูงขึ้น ซึ่งทำให้ทุกวันนี้ MATLAB เป็นโปรแกรมที่สุดยอดในการคำนวณที่คำนวณด้าน matrix สำหรับงานทางวิทยาศาสตร์และวิศวกรรมโปรแกรมหนึ่ง นับจากวันแรกที่ได้เริ่มโครงการขึ้นจนกระทั่งในไตรมาสสุดท้ายของปี ค.ศ.2000 ได้พัฒนาเป็น MATLAB 6.0 ซึ่งเป็นการปรับปรุงใหม่และ

ออกสู่ตลาดเป็นครั้งที่ 12 สำหรับในมุมมองของการศึกษานั้น MATLAB ถือได้ว่าเป็นเครื่องมือที่สำคัญอันหนึ่งสำหรับนักศึกษาทางด้านวิทยาศาสตร์และเทคโนโลยีที่จะใช้เป็นเครื่องมือในการคำนวณ และขณะนี้หลายๆ มหาวิทยาลัยได้ยกอันดับของ MATLAB ขึ้นจากโปรแกรมสำเร็จรูป ให้เป็นภาษาสำหรับการใช้งานทางด้านเทคโนโลยี นั่นคือมีระดับเป็นภาษาเหมือนกับภาษา C หรือ Fortran นั่นเอง นอกเหนือจากเพื่อการเรียนการสอนในสถาบันการศึกษาแล้ว MATLAB ยังเป็นเครื่องมือสำคัญที่ใช้ในงานวิจัย งานพัฒนาและการวิเคราะห์ของหน่วยงานต่างๆมากมาย

นอกเหนือจากตัวโปรแกรม MATLAB เองแล้ว บริษัท MathWorks ผู้ผลิต MATLAB ยังได้ผลิตเครื่องมือหรือที่เรียกว่า toolbox ซึ่งเป็นโปรแกรมที่เขียนขึ้นเพื่อประกอบกับการใช้ MATLAB สำหรับงานที่จำเพาะเจาะจงหลายประเภท Toolbox นั้นเป็นการนำเอาโปรแกรมที่เขียนขึ้นเป็นฟังก์ชันสำหรับ MATLAB เพื่อรวมเข้าเพื่อให้ผู้ใช้งานมีความสะดวกในการเรียกใช้มากขึ้น ทำให้ผู้ใช้ไม่จำเป็นที่จะต้องสร้างโปรแกรมขึ้นมาใช้งานเอง โดย toolbox ที่สร้างขึ้นจะครอบคลุมการทำงานด้านต่างๆมากมายเช่น signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation เป็นต้น

ถ้าหากจะสรุปโดยรวมแล้วความสามารถหลักของ MATLAB ที่ทำให้เป็นโปรแกรมที่เหมาะสมกับการทำงานทางด้านวิศวกรรมด้วยเหตุผลดังนี้

- MATLAB เป็นโปรแกรมเพื่อการคำนวณและแสดงผลได้ทั้งตัวเลขและรูปภาพซึ่งมีประสิทธิภาพสูง โดยทางบริษัท Math Works ผู้ผลิตได้ให้นิยามว่าเป็น High-Performance Numeric Computation and Visualization Software
- MATLAB จะควบคุมการทำงานด้วยชุดคำสั่งและยังสามารถรวบรวมชุดคำสั่งเป็นโปรแกรมได้อีกด้วย
- MATLAB มี function ที่เหมาะสมกับงานทางวิศวกรรมพื้นฐานมากมาย นอกจากนั้นผู้ใช้ยังสามารถเขียน function ขึ้นมาใหม่โดยสามารถใช้ประโยชน์จาก function ที่มีอยู่แล้วเพื่อให้เหมาะสมกับงานของผู้ใช้แต่ละกลุ่ม
- ลักษณะการเขียนโปรแกรมใน MATLAB จะใกล้เคียงการเขียนสมการทางคณิตศาสตร์ที่เราคุ้นเคยจึงง่ายกว่าการเขียนโปรแกรมโดยใช้ภาษาชั้นสูงเช่น C, FORTRAN หรืออื่นๆ
- MATLAB มีความสามารถในการเขียนกราฟและรูปภาพทั้ง 2 มิติและ 3 มิติได้อย่างมีประสิทธิภาพ
- MATLAB สามารถทำ Dynamic Link กับโปรแกรมอื่นๆได้ไม่ว่าจะเป็น Word, Excel หรืออื่นๆที่ร่วมทำงานอยู่บน windows
- MATLAB มี toolbox หรือชุด function พิเศษสำหรับผู้ใช้ที่ต้องการใช้งานเฉพาะทางหรืองานด้านวิศวกรรมขั้นสูงอื่นๆ

MATLAB มีโปรแกรมที่จำหน่ายแก่นักศึกษาโดยเฉพาะซึ่งจะมีราคาต่ำกว่าราคาปกติมากแต่มีประสิทธิภาพเท่าเทียมกัน แม้ว่าอาจจะมีการจำกัดขีดความสามารถของโปรแกรมบ้างแต่ก็เพียงพอ

สำหรับนักศึกษาที่จะใช้เพื่อการศึกษาในระดับอุดมศึกษา และ MATLAB ยังมีความสามารถและข้อดีอื่นๆอีกมากมายซึ่งจะได้กล่าวในรายละเอียดในภายหลัง

การหารายละเอียดเพิ่มเติม

สำหรับคู่มือฉบับนี้มีจุดประสงค์เขียนขึ้นเพื่อจะแนะนำให้นักศึกษาเข้าใจถึงพื้นฐานเบื้องต้นของการใช้โปรแกรม MATLAB เท่านั้น สำหรับรายละเอียดของการใช้ชุดคำสั่งของ MATLAB นั้นมีมากมายทำให้ไม่สามารถที่จะบรรจุในที่เอกสารนี้ได้ทั้งหมด อย่างไรก็ตามนักศึกษาสามารถศึกษาค้นคว้าได้จาก

- เอกสารประกอบการใช้งานหรือคู่มือของ MATLAB
- หารายละเอียดเพิ่มเติมได้จาก Help ของ MATLAB
- ค้นคว้ารายละเอียดเพิ่มเติมรวมทั้งสอบถามข้อสงสัยทาง Internet ของบริษัทผู้ผลิตได้ที่ **http://www.mathworks.com** นอกจากนี้ผู้ใช้ที่มีการลงทะเบียนอย่างถูกต้องกับทางผู้ผลิตจะได้รับโปรแกรมเพิ่มเติมหรือแก้ไขข้อผิดพลาดต่างๆ เพิ่มเติมจากทางบริษัทอีกด้วย

1.2 ข้อแนะนำเบื้องต้นเกี่ยวกับการทำงานของ MATLAB

❶ ระบบการทำงานของ MATLAB

ในการทำงานของ MATLAB เพื่อให้การทำงานเป็นไปตามจุดมุ่งหมาย MATLAB ได้แบ่งส่วนการทำงานของโปรแกรมออกเป็นส่วนหลักที่สำคัญ 5 ส่วนด้วยกัน

1. Development Environment.
2. The MATLAB Mathematical Function Library.
3. The MATLAB Language.
4. Handle Graphics
5. The MATLAB Application Program Interface (API)

ซึ่งแต่ละส่วนจะมีหน้าที่ควบคุมในการทำงานแบบหนึ่งๆ และประสานการทำงานระหว่างส่วนต่างๆ ไปพร้อมกันด้วย สำหรับรายละเอียดในการทำงานของส่วนต่างๆ มีดังนี้

☋ Development Environment

ในส่วนนี้จะเป็นชุดเครื่องมือที่ช่วยให้เราสามารถที่จะใช้ฟังก์ชันก์และไฟล์ต่างๆ โดยเครื่องมือหลายตัวในนี้จะมีลักษณะเป็น graphical user interface ซึ่งรวมถึง MATLAB Desktop และ Command Windows, command history และ browsers สำหรับเพื่อใช้ดู help, workspace, files และ search path ซึ่งทั้งหมดนี้จะได้กล่าวถึงในรายละเอียดต่อไป

☋ The MATLAB Mathematical Function Library

ในส่วนนี้จะเป็นที่รวบรวมส่วนของโปรแกรมที่ได้รวบรวมเป็นไฟล์ย่อยๆ ไว้ ไฟล์แต่ละไฟล์จะเป็นไฟล์มที่เขียนขึ้นมาเพื่อใช้กำหนดลักษณะในการคำนวณหรือ Algorithms แบบต่างๆ นับจากฟังก์ชันง่ายๆ เช่นการบวก ฟังก์ชันตรีโกณมิติพื้นฐาน เช่น sine, cosine ไปจนถึงฟังก์ชันที่มีความซับซ้อน มีขั้นตอนในการคำนวณมากมาย เช่นการหา inverse ของ matrix การหา eigenvalues และ eigenvector หรือ fast Fourier transforms เป็นต้น

☋ The MATLAB Language

ส่วนนี้จะเป็นภาษาระดับสูงที่ใช้ตัวแปรเป็น matrix หรือ array ซึ่งมีคำสั่งที่ใช้ในการควบคุมการทำงานของโปรแกรม การทำงานของฟังก์ชัน การกำหนดโครงสร้างของตัวแปรแบบต่างๆ กำหนด input และ output ของโปรแกรม ซึ่งทั้งหมดนี้จะช่วยทำให้ในการเขียนโปรแกรม Matlab แต่ละโปรแกรมจะเป็นโปรแกรมที่มีขนาดเล็กกว่าเมื่อเทียบกับโปรแกรมที่ใช้เพื่อวัตถุประสงค์เดียวกันแต่ผู้ใช้ต้องเขียนฟังก์ชันการทำงานทุกขั้นตอนขึ้นมาเอง

☋ Handle Graphics

ส่วนนี้จะเป็นส่วนที่ใช้แสดงกราฟฟิกส์และรูปภาพต่างๆ รวมถึงคำสั่งระดับสูงที่ใช้ในการแสดงผลในสองและสามมิติ การจัดรูปแบบในลักษณะ image processing การทำภาพเคลื่อนไหว นอกจากนี้ในส่วนนี้ยังได้รวมเอาภาษาในระดับต่ำไว้เพื่อให้เราสามารถปรับแก้รูปภาพต่างๆ ให้เป็นไปตามที่เราต้องการได้มากที่สุด รวมถึงการสร้าง Graphic User Interface ภายใต้การทำงานของ MATLAB ด้วย

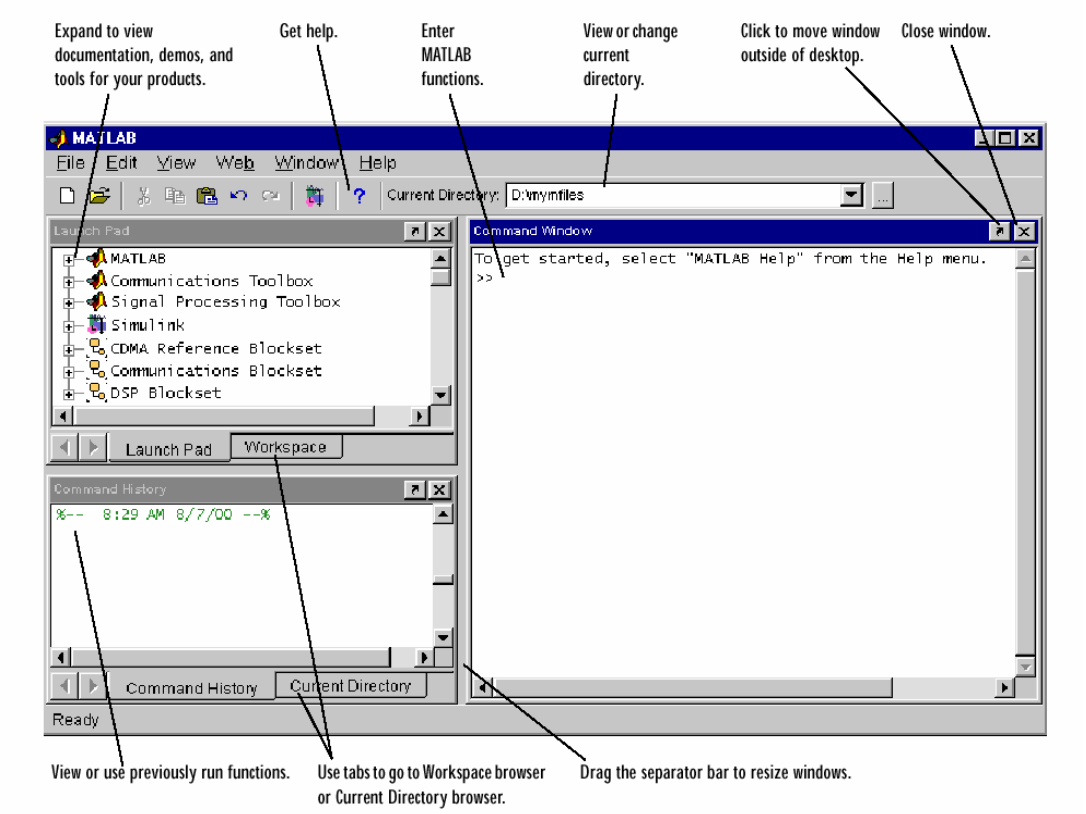
☋ The MATLAB Application Program Interface (API)

ส่วนนี้จะเป็น library ที่ให้เราสามารถที่จะเขียนโปรแกรมขึ้นในภาษา C หรือ Fortran แล้วมีการเชื่อมโยงการทำงานเข้ากับ MATLAB ซึ่งในส่วนนี้ยังได้รวมถึงการเขียนโปรแกรมขึ้นมาแล้วเรียกฟังก์ชันของ MATLAB ไปใช้งาน (dynamic linking), ซึ่งจะทำให้ MATLAB มีหน้าที่เสมือน engine ในการคำนวณ รวมถึงสามารถที่จะเขียนหรืออ่าน MAT-file ได้ด้วย

❷ เริ่มการทำงานกับ MATLAB

สำหรับการทำงานภายใต้ระบบปฏิบัติการ Windows เราสามารถที่จะเริ่มการทำงานของ MATLAB ได้โดยการใช้เมาส์กดที่ shortcut ของ MATLAB ซึ่งจะปรากฏอยู่บน desktop หลังจากที่เราได้ติดตั้งโปรแกรมนี้ลงไปเรียบร้อยแล้ว หรืออาจจะใช้เมาส์กดที่ปุ่ม Start แล้วเลือก MATLAB ภายใต้เมนู Programs เหมือนกับการเปิดโปรแกรมอื่นๆ ใน Windows ก็ได้

เมื่อเราเริ่มเปิดโปรแกรม MATLAB 6 สิ่งแรกที่เราจะพบในครั้งแรกก็คือ MATLAB desktop ที่ประกอบด้วยหน้าต่างย่อยๆ อีกหลายหน้าต่าง โดยหน้าต่างแต่ละอันนั้นจะทำหน้าที่เป็นเครื่องมือที่จะช่วยเราในการจัดการเกี่ยวกับไฟล์ ตัวแปร และอื่นๆเกี่ยวกับการทำงานของ MATLAB โดย MATLAB desktop จะมีลักษณะดังรูปต่อไปนี้



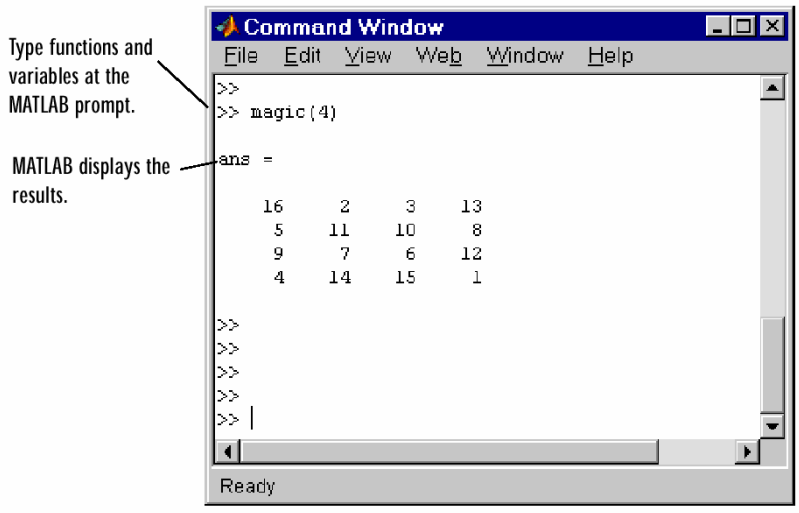
แม้ว่าในบางกรณี Launch Pad ของเราอาจมีลักษณะแตกต่างไปจากรูปข้างบนนี้บ้าง ตามแต่จำนวนโปรแกรมหรือ toolbox ที่เราได้บรรจุเข้าไปในการติดตั้ง MATLAB เราสามารถที่จะเปลี่ยนแปลงลักษณะของ desktop ได้ด้วยการเปิด ปิด เคลื่อนย้ายและปรับขนาดของเครื่องมือเหล่านี้ได้ นอกจากนั้นเรายังสามารถที่จะย้ายเครื่องมือเหล่านั้นออกไปนอก desktop หรือย้ายกลับเข้ามา (docking) วางไว้กับ desktop ได้ โดยเครื่องมือบน desktop เหล่านี้จะช่วยการทำงานในขั้นตอนที่ใช้บ่อยๆไม่ว่าจะเป็น shortcut หรือ context menus อีกทั้งเรายังสามารถที่จะกำหนดลักษณะพิเศษต่างๆของ desktop ให้เป็นไปตามที่เราต้องการได้ ด้วยการเลือก Preferences จากเมนู File เพื่อเปลี่ยนลักษณะของตัวหนังสือที่ใช้ใน Command Window สำหรับรายละเอียดในการปรับแก้ลักษณะของ desktop นี้ดูได้จากการกดปุ่ม Help ภายใต้หน้าต่าง Preferences

1.3 หน้าต่างที่มีใน MATLAB

ใน MATLAB 6.0 ซึ่งเป็น Version ใหม่ของ MATLAB จะประกอบด้วยหน้าต่างย่อยๆ หลายหน้าต่าง ในหัวข้อนี้จะเป็นการอธิบายถึงหน้าต่างที่อยู่ภายใต้ MATLAB desktop ซึ่งได้รับการพัฒนาขึ้นมาใช้ใน version 6 นี้ โดยแม้ว่าในความเป็นจริงเราสามารถที่จะใช้คำสั่งทั่วๆไปที่เป็นฟังก์ชันของ MATLAB พิมพ์เข้าไปเพื่อให้ได้ผลตามที่เราต้องการได้เช่นกัน แต่การใช้หน้าต่างและเครื่องมือใน MATLAB desktop ซึ่งถือว่าเป็นรูปแบบใหม่ของ MATLAB 6 นี้ จะช่วยให้การใช้คำสั่งทำได้สะดวกและรวดเร็วขึ้น โดยเราสามารถสั่งคำสั่งเหล่านั้นผ่านเมาส์ได้ และมีการปรับเปลี่ยนค่าต่างๆ ได้สะดวกและรวดเร็วขึ้นมาก สำหรับหน้าต่างที่สำคัญใน MATLAB Desktop จะมีอยู่ด้วยกัน 5 หน้าต่างคือ Command Windows, Command History Window, Current Directory Browser, Workspace Browser และ Launch Pad

❶ Command Window

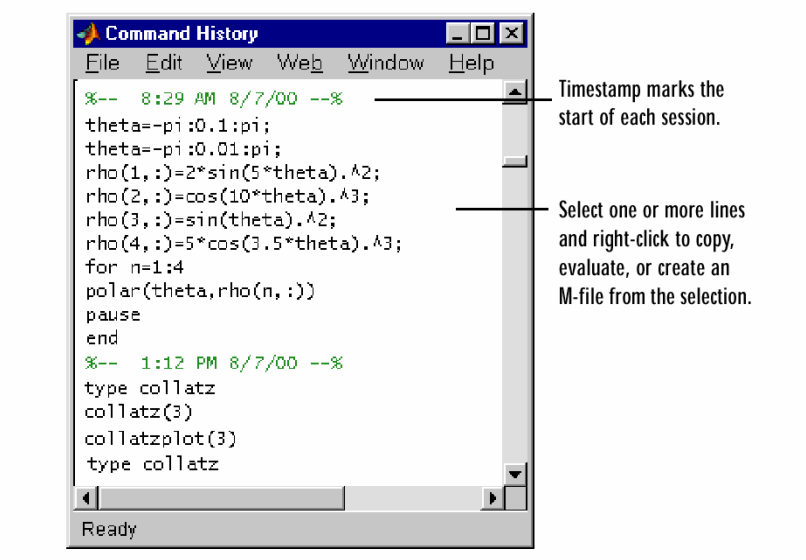
Command Window เป็นส่วนที่เราใช้ในการป้อนชุดคำสั่งเพื่อให้ MATLAB ทำงานตามคำสั่งนั้น และก็จะแสดงผลที่เป็นตัวหนังสือในหน้าต่างนี้ ซึ่งใน version ก่อนๆ ของ MATLAB ก็จะมีหน้าต่างนี้อยู่แล้ว ซึ่งเราสามารถที่จะกำหนดคำสั่งด้วยตัวอักษร เพื่อให้ MATLAB ทำงานตามที่เราต้องการได้นั่นเอง



การที่เราจะป้อนคำสั่งให้ ที่ MATLAB Command Window โดย MATLAB จะรับคำสั่งเกือบทั้งหมดทางหน้าต่างนี้ ซึ่งทุกครั้งที่ MATLAB พร้อมที่จะรับคำสั่ง MATLAB จะแสดงเครื่องหมาย MATLAB prompt ในลักษณะ » ขึ้นสำหรับ Professional Edition หรือจะเป็นลักษณะ **EDU»** สำหรับ Student Edition เมื่อปรากฏเครื่องหมายดังกล่าวนี้ขึ้นก็แสดงว่า MATLAB พร้อมที่จะรับคำสั่งต่อไป

แต่เพื่อความสะดวกสำหรับในเอกสารนี้เราจะไม่มีการแสดงเครื่องหมาย » หรือ **EDU»**หน้าชุดคำสั่งต่างๆ เพราะเราจะได้ทราบในภายหลังว่าการป้อนชุดคำสั่งเหล่านี้อาจจะกำหนดผ่านทาง file ที่เขียนขึ้นเป็นชุดคำสั่งให้กับ MATLAB หรือที่เรานิยมเรียกกันสั้นๆว่า M-file

หน้าต่าง Command History นี้มีไว้เพื่อให้เราทราบว่าเราได้ใช้คำสั่งอะไรไปแล้วบ้าง โดยข้อมูลการใช้คำสั่งจะได้รับการบันทึกไว้ทุกครั้งที่มีการเปิดโปรแกรม MATLAB ขึ้นมาใช้ นอกจากนั้นยังบอกวัน-เวลาที่เราได้เข้ามาใช้โปรแกรมนี้ในแต่ละครั้งด้วย ในหน้าต่างนี้เราสามารถที่จะเลือกใช้คำสั่งที่เคยใช้มาก่อนหน้านี้แล้วอีกก็ได้ โดยการกดเมาส์สองครั้งที่คำสั่งนั้น หรือเราอาจจะเลือกที่จะทำสำเนาคำสั่งนั้นก็ได้



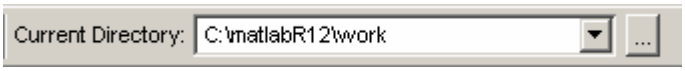
เราสามารถที่จะลบ ข้อมูลในหน้าต่างนี้ได้ โดยการกดเมาส์ปุ่มขวาแล้วเลือกว่าจะลบเฉพาะตัวที่เลือก (Delete Selection) ลบตั้งแต่ต้นจนกระทั่งถึงตัวที่เราเลือก (Delete to Selection) หรือลบทั้งหมดเลย (Delete Entire History) ก็ได้

นอกเหนือจากนั้นเรายังสามารถที่จะเลือกช่วงของคำสั่งที่เราใช้มาใน Command Windows นำมารวมกันแล้วสร้างเป็น M-file ได้อีกด้วย วิธีการก็คือเราเลือกช่วงคำสั่งที่เราต้องการขึ้นมาก่อนโดยการเลือกคำสั่งทีละคำสั่งแล้วกดปุ่ม Shift บนแป้นพิมพ์ค้างไว้ เมื่อได้คำสั่งครบตามต้องการแล้วให้กดเมาส์ปุ่มขวา แล้วเลือก Create M-File เราจะเข้าไปสู่ Editor เพื่อสร้าง M-file ต่อไป รายละเอียดในส่วนนี้จะกล่าวถึงอีกครั้งหนึ่งในการเขียน M-file

❷ Current Directory Browser

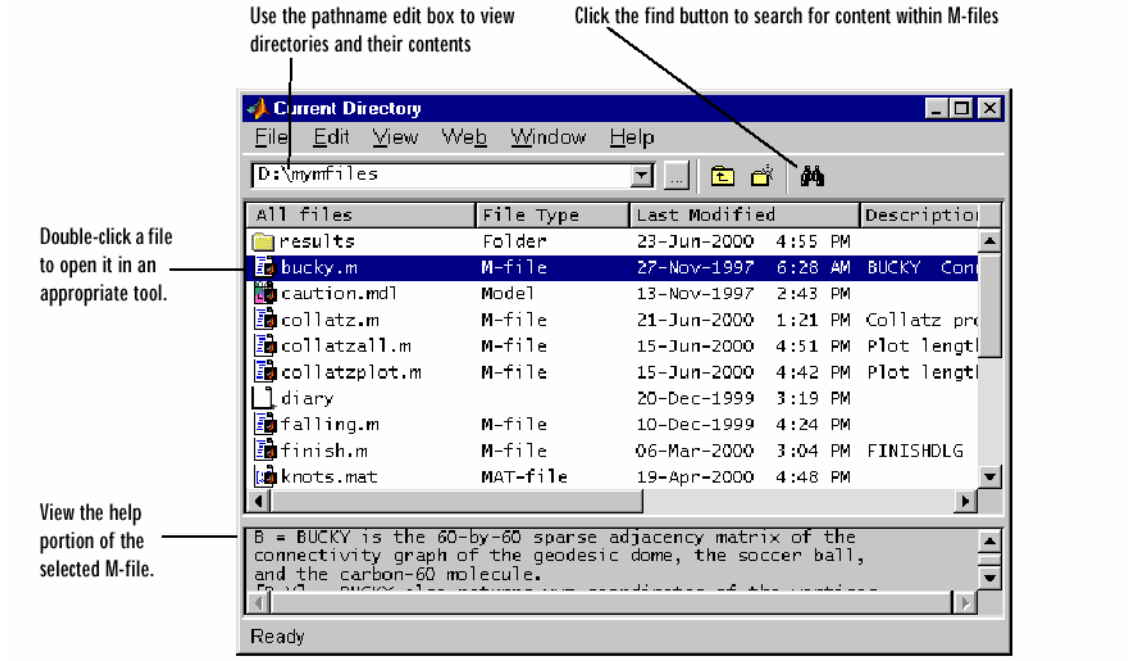
คำสั่งที่กำหนดให้ MATLAB ทำนั้น MATLAB จะใช้ Current Directory และ Search Path เป็นจุดเริ่มต้นของการทำงานและเป็นพื้นที่ในการค้นหาข้อมูลหรือคำสั่งต่างๆ ตามที่ได้รับคำสั่งมา โดยการค้นหาจะจำกัดวงอยู่เฉพาะในสองส่วนหลักนี้เท่านั้น MATLAB จะไม่มีการค้นหา file หรือคำสั่งต่างๆ นอกพื้นที่ดังกล่าว ดังนั้นคำสั่งหรือ M-file ต่างๆ ที่เราต้องการจะใช้งานนั้นจำเป็นอย่างยิ่งที่จะต้องอยู่ใน Current Directory หรือ Search Path

วิธีการที่จะดูว่าขณะนี้เราอยู่ใน Current Directory ใด ก็สามารถทำได้โดยดูที่แถบเครื่องมือ Current Directory ซึ่งอยู่ที่ Desktop Toolbar มีลักษณะตามที่แสดงในรูป นอกจากนั้นเรายังสามารถที่จะปรับเปลี่ยน Current Directory โดยการใช้แถบเครื่องมือนี้ได้อีกด้วย



โดยหากว่าเราต้องการจะปรับเปลี่ยนไปใช้ Current Directory ที่เราเคยใช้มาก่อนหน้านี้แล้ว เราสามารถกด เพื่อให้เมนูแสดง directory ที่เราเคยใช้เป็น Current Directory มาก่อน แต่ถ้าหากว่าเราต้องการที่จะเปลี่ยน Current Directory ไปอยู่ในdirectory ที่เราไม่เคยใช้มาก่อน เราจะต้องเลือกปุ่ม Browser ซึ่งจะเป็นการเปิดหน้าต่างใหม่เพื่อให้เราค้นหา directory ที่เราต้องการ เหมือนกับการค้นหา file ในระบบปฏิบัติการ Windows ทั่วๆไป

นอกเหนือจากนั้นสำหรับการค้นหา ดู หรือ เปิด file ที่เราต้องการ เราสามารถที่จะทำได้โดยใช้ MATLAB Current Directory Browser ซึ่งเป็น desktop tool ที่มีหน้าที่เพื่อการนี้โดยเฉพาะ ลักษณะของ current Directory Browser จะมีลักษณะดังรูป



เมื่อใช้ Current Directory Browser ตามรูปที่แสดงข้างบนนี้ เราสามารถที่จะค้นหา file โดยใช้ปุ่ม หรือเปิด file โดยการกดเมาส์สองครั้งที่ file ที่เราต้องการ นอกจากนั้นที่ส่วนล่างของหน้าต่างนี้ ยังแสดง help ของ M-file ที่เราได้เลือกในหน้าต่างส่วนบนด้วย สำหรับปุ่มและเมนูอื่นๆ ผู้ที่คุ้นเคยการทำงานกับ Windows คงจะทราบถึงความหมายของมันดีอยู่แล้ว เพราะจะมีลักษณะที่ใกล้เคียงกันนั่นเอง

Search Part

เพื่อที่จะให้ฟังก์ชันที่เราสั่งการจาก Command Window ทำงานนั้น Function นั้นรวมถึงฟังก์ชันก์ที่ถูกเรียกใช้จากในฟังก์ชันไฟล์นั้นอีกทีหนึ่งนั้นจะต้องอยู่ใน Search Path โดย Search Path นี้หมายถึงกลุ่มของ directory ที่เราได้รวบรวมและบอก MATLAB ว่าจะเป็นกลุ่มของ directory ที่ MATLAB จะต้องค้นหา ในการติดตั้ง MATLAB ไฟล์และ Toolbox ที่ได้รับการติดตั้งเข้าไปจะถูกรวมเข้าไปอยู่ใน Search Path นี้โดยอัตโนมัติอยู่แล้ว หากเราต้องการที่จะเพิ่ม Search Path เราสามารถที่จะทำได้โดย เลือกคำสั่ง Set Path ภายใต้ File Menu เพื่อเพิ่ม directory ที่เราต้องการเข้าไปอยู่ใน Search Path

หากเราต้องการที่จะดูว่ามี directory ใดบ้างที่อยู่ใน Search Path เราสามารถใช้คำสั่ง **path** ที่ MATLAB Command Prompt เช่น เมื่อเราป้อนคำสั่ง

```
» path
```

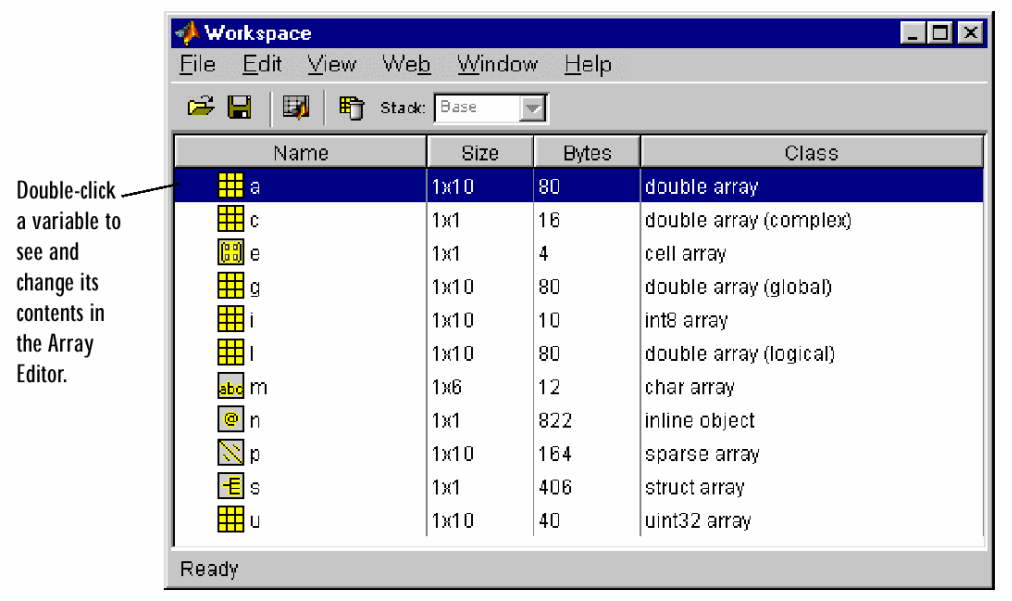
เราจะได้ผลเป็น

```
                MATLABPATH
C:\MATLABR12\toolbox\MATLAB\general
C:\MATLABR12\toolbox\MATLAB\ops
……….
C:\MATLABR12\toolbox\wavelet\wavedemo
C:\MATLABR12\toolbox\rtw\targets\xpc\target\kernel\embedded
C:\MATLABR12\work
```

ซึ่งเป็นการแสดง Search Part ทั้งหมดที่เราได้กำหนดให้ MATLAB ในขณะนั้น และเรายังสามารถใช้คำสั่ง **addpath** เพื่อเพิ่ม Search Path หรือใช้ **rmpath** เพื่อตัด directory นั้นออกจาก Search Path ได้อีกด้วย

❸ Workspace Browser

เมื่อเราได้มีการสร้างค่าตัวแปรหรือพารามิเตอร์ขึ้นใน MATLAB ค่าเหล่านั้นจะถูกเก็บไว้ในพื้นที่การทำงาน (Workspace) และหน่วยความจำของ MATLAB เราจะเพิ่มตัวแปรลงในพื้นที่ทำงานได้ด้วยการใช้คำสั่ง ให้ M-file ทำงานหรือ load ค่าที่บันทึกไว้เข้าสู่พื้นที่ทำงาน เพื่อที่จะดูว่าในขณะนั้นมีตัวแปรอะไรบ้างที่มีอยู่ในพื้นที่ทำงาน ใน MATLAB 6.0 นี้เราสามารถใช้ Workspace Browser ซึ่งเป็นหน้าต่างเครื่องมือหนึ่งใน Desktop Tool หรือในทุก version ของ MATLAB เราอาจใช้คำสั่ง who หรือ whos ที่ Command Windows ก็ได้ Workspace Browser จะมีลักษณะโดยทั่วๆ ตามรูปต่อไปนี้



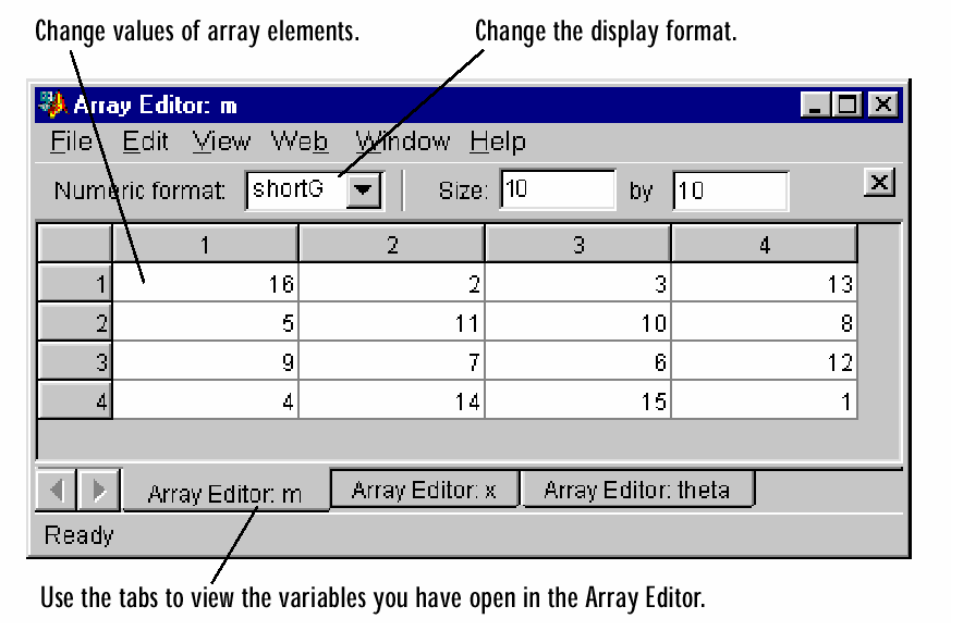
ที่ Workspace Browser นี้เราสามารถที่จะดูว่ามีตัวแปรหรือ array ตัวใดที่อยู่ใน workspace บ้าง นอกจากนี้สำหรับตัวแปรแต่ละตัวก็จะมีข้อมูลที่บอกว่าตัวแปรแต่ละตัวนั้นเป็นประเภทใด มีขนาดเท่าใด ใช้หน่วยความจำมากเท่าใดอีกด้วย สำหรับตัวแปรแต่ละตัวที่ปรากฏอยู่ในรายการภายใต้ workspace Browser นี้เราสามารถที่จะลบมันออกจาก Workspace ได้สองวิธีคือ

- ด้วยการเลือกที่ตัวแปรตัวนั้น กดเมาส์ปุ่มขวาแล้วเลือก Delete Selection
- ด้วยการเลือกตัวแปรตัวนั้นแล้วกดปุ่ม  ที่แถบเครื่องมือ เพื่อลบตัวแปรนี้ออกไป

เรายังสามารถที่จะแก้ไขข้อมูลของตัวแปรบางประเภทได้ เช่นตัวแปรที่เป็น array ซึ่งมีลักษณะเป็นเมตริกส์ (รายละเอียดของเรื่องตัวแปรประเภทต่างๆ เราจะได้กล่าวถึงในบทต่อไป) เราสามารถที่จะแก้ไขที่ cell ใด cell หนึ่งเป็นการเฉพาะได้ โดยการเปิดตัวแปรนั้นขึ้นมา ซึ่งการเปิดตัวแปรนั้นสามารถทำได้หลายวิธีคือ

- กดเมาส์สองครั้งที่ตัวแปรนั้น
- เลือกตัวแปรนั้นกดเมาส์ปุ่มขวาแล้วเลือก Open Selection
- เลือกตัวแปรนั้นแล้วกดปุ่ม  ที่แถบเครื่องมือ

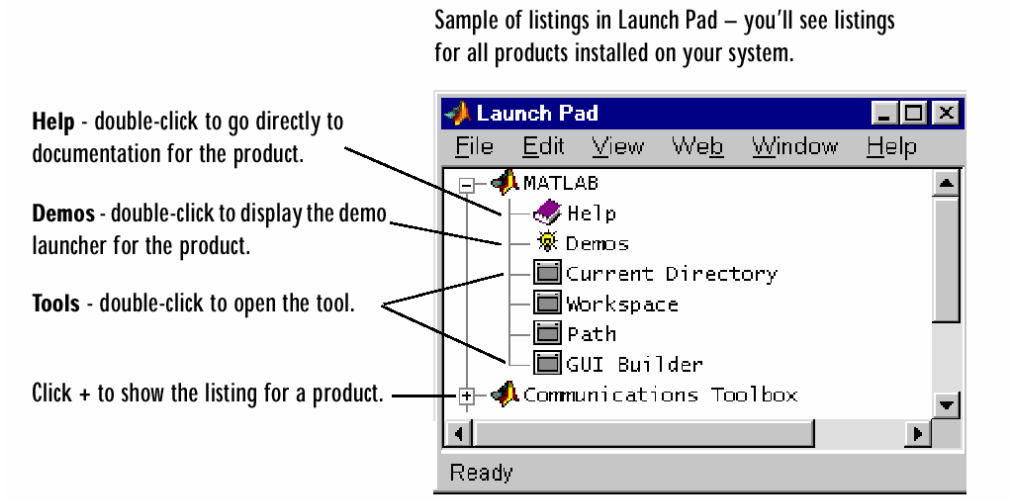
เมื่อเราทำการเปิดตัวแปรนั้นแล้ว เราจะได้หน้าต่างใหม่ขึ้นมา ซึ่งเราเรียกว่า Array Editor จะมีลักษณะคล้ายหน้าต่างของโปรแกรมประเภท space sheet คือมีลักษณะเป็น Matrix ซึ่งขนาดของ Matrix นั้นก็จะขึ้นอยู่กับขนาดของตัวแปร ลักษณะของ array editor จะมีลักษณะดังรูปต่อไปนี้



บน Array Editor เราจะทราบขนาดของตัวแปร รูปแบบการแสดงผล (จะกล่าวถึงในหัวข้อต่อไป) และเราสามารถที่จะปรับเปลี่ยนค่า cell แต่ละ cell ได้อย่างอิสระ และเราสามารถใช้ Array Editor นี้แสดงผลตัวแปรได้หลายๆตัวพร้อมกัน โดยเลือกจะให้แสดงผลตัวใดได้โดยใช้แถบเลือกข้างล่าง นอกจากนี้เรายังสามารถที่จะปรับเปลี่ยนขนาดของ array ได้ โดยการเปลี่ยนค่า size ให้เป็นไปตามต้องการ หากว่าเราทำการเพิ่มขนาด cell ที่เราไม่ได้กำหนดค่าจะมีค่าเท่ากับศูนย์ ข้อควรระวังคือการลดขนาดตัวแปรจะทำให้มีค่าของตัวแปรบางส่วนหายไป และเราไม่สามารถที่จะ undo เพื่อเรียกข้อมูลคืนมาได้

❹ Launch Pad

Launch Pad เป็นหน้าต่างที่แสดง toolbox ต่างๆที่เราได้ติดตั้งไว้ในเครื่องของเรา และทำให้เราสามารถที่จะเข้าสู่ เครื่องมือ ตัวอย่าง และเอกสารที่เกี่ยวข้องกับ MATLAB หรือ Toolbox ต่างๆ ได้โดยง่าย ลักษณะของ Launch Pad ก็จะเหมือนกับการแสดง file ใน Windows Explorer คือสามารถที่จะขยายหรือลดการแสดงรายละเอียดใน Toolbox ต่างๆ ได้ และเมื่อเรากดเมาส์สองครั้งในหัวข้อที่ต้องการเราก็จะได้เห็นตัวอย่าง หรือเอกสารที่เกี่ยวข้องกับหัวข้อนั้นได้ทันที โดยไม่ต้องเสียเวลาในการค้นหา ลักษณะของ Launch Pad จะเป็นตามรูปต่อไปนี้



สำหรับหน้าต่างทั้ง 5 ที่เปิดขึ้นมาพร้อมกับ MATLAB Desktop นี้เราสามารถที่จะนำออกมาจาก Desktop ได้หรือที่เรียกว่า undock โดยการกดที่ปุ่ม ที่อยู่ที่ขอบบนขวาของหน้าต่างนั้นๆได้ หรือเราอาจจะปิดหน้าต่างนั้นไปเลยก็ได้ ยกเว้น (Command Windows) และถ้าเราต้องการจะนำหน้าต่างนี้กลับมาอยู่ที่ Desktop เราก็สามารถที่จะเรียกมันกลับมาได้โดยการใช้คำสั่ง **Dock** ..... ภายใต้เมนู **View** ของหน้าต่างนั้นเมื่ออยู่นอก desktop

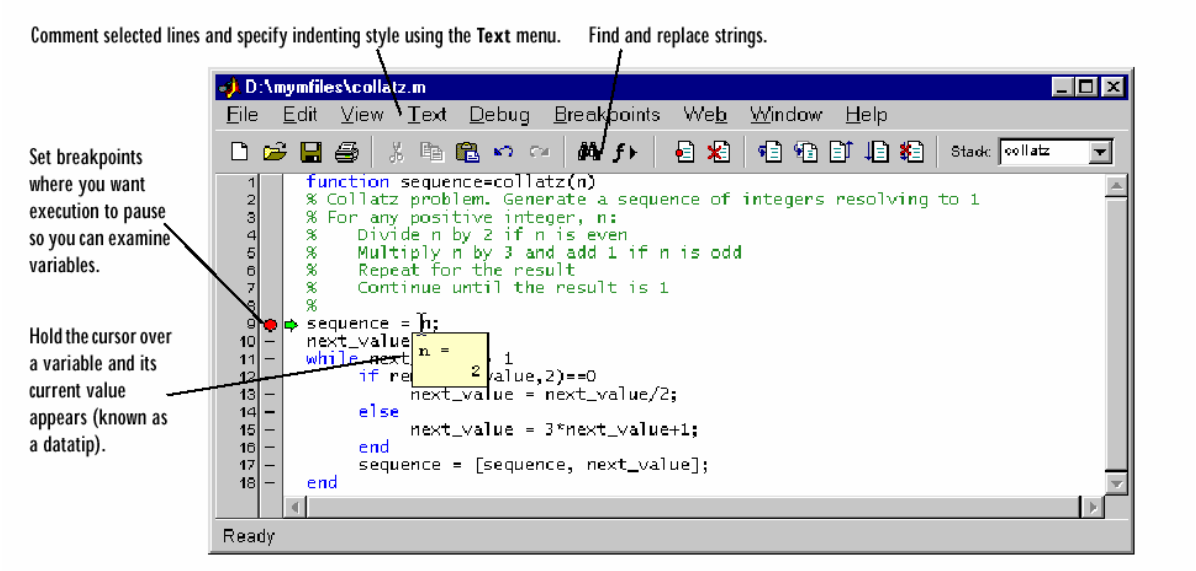
นอกจากนี้เรายังสามารถที่จะปรับเปลี่ยนรูปร่างของ Desktop ของเราได้โดยการใช้คำสั่งหลายคำสั่งที่อยู่ภายใต้เมนู **View** ซึ่งเราจะไม่ขอกล่าวในรายละเอียดในที่นี้ แต่หากได้มีการทดลองใช้คำสั่งต่างๆ ภายใต้เมนูนี้ เราเชื่อว่าผู้ใช้คงจะสามารถเข้าใจในคำสั่งต่างๆ ได้โดยไม่ยากนัก นอกเหนือจากนั้นเรายังสามารถที่จะตั้งค่าต่างๆ ของ Desktop นี้ได้โดยการใช้คำสั่ง **Preference** ภายใต้เมนู **File** ซึ่งจะสามารถปรับค่าต่างๆ เช่นแบบตัวหนังสือ สีที่แสดง และอื่นๆได้อีกมาก รายละเอียดเหล่านี้ บางส่วนเราจะกล่าวถึงในบทต่อๆไป

❺ หน้าต่าง Editor/Debugger

ในการเขียนโปรแกรมหรือที่เรียกว่า M-file จะเขียนด้วย Text Editor ธรรมดาเช่น Notepad ก็ได้ เพราะ M-file จะเป็นโปรแกรมที่ใช้ตัวอักษรในลักษณะ ASCII Code ธรรมดา และสำหรับ MATLAB version 5 เป็นต้นมานั้นจะมี editor มาพร้อมกับ MATLAB ด้วยเลยทำให้สะดวกในการใช้งานเป็นอย่างมากเพราะนอกจากจะเป็น editor แล้วยังมี debugger เพื่อช่วยในการแก้ไขโปรแกรมพร้อมอยู่ด้วย

เราสามารถที่จะใช้ Editor/Debugger เพื่อสร้างและแก้ไข M-files ซึ่งเป็นการเขียนโปรแกรมที่จะเรียกชุดคำสั่งหรือฟังก์ชันต่างๆของ MATLAB ขึ้นมาทำงาน Editor/Debugger นี้จะทำหน้าที่เป็นทั้ง text editor เพื่อใช้ในการเขียนโปรแกรม และทำหน้าที่เป็น Debugger คือมีเครื่องมือที่ช่วยในการแก้ไขโปรแกรมกรณีที่โปรแกรมเกิดความผิดพลาดขึ้น ฟังก์ชันหรือคำสั่งที่เราเขียนขึ้นเพื่อใช้กับ MATLAB

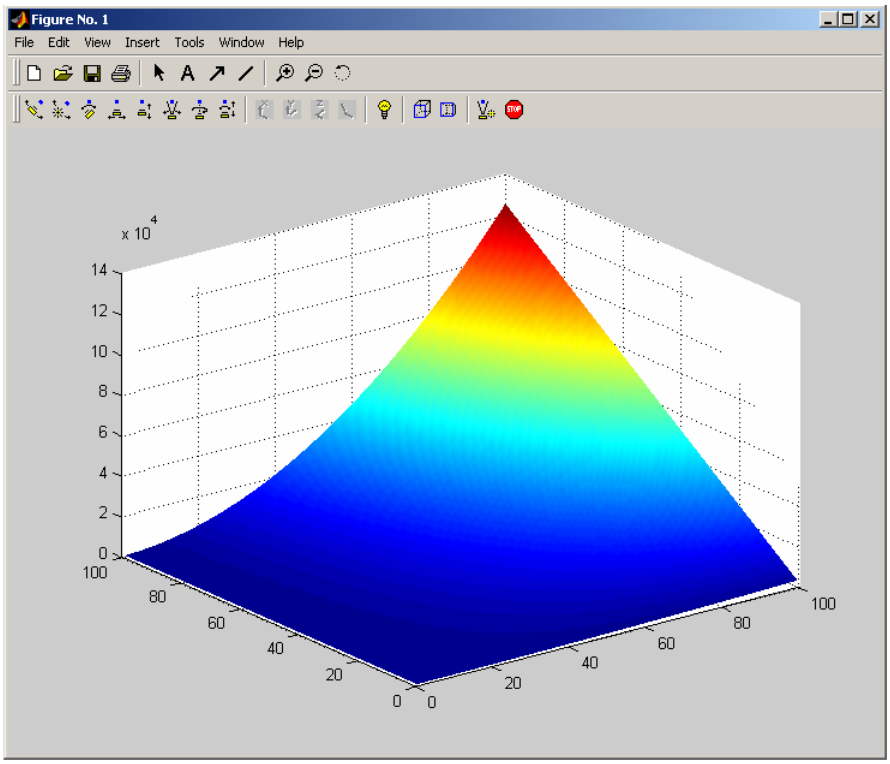
นั้นเราจะเรียกว่า MATLAB file หรือเพื่อความสะดวกเรานิยมที่จะย่อแล้วเรียกกันว่า M-file ลักษณะของ Editor/Debugger จะมีลักษณะตามรูปต่อไปนี้



ซึ่งบน Editor/Debugger นี้จะมีเครื่องมือหลายอย่างช่วยเหลือให้เราสามารถที่จะเขียนโปรแกรมได้สะดวกขึ้น อย่างไรก็ตามรายละเอียดในการใช้ Editor/Debugger จะกล่าวถึงในบทที่เกี่ยวข้องกับการเขียน M-file

❻ หน้าต่างแสดงรูปภาพ Graphic Windows

เมื่อได้รับคำสั่งให้เขียนกราฟ MATLAB จะแสดงผลบน Graphic Windows ซึ่งจะเรียกขึ้นใช้งานโดยอัตโนมัติ Graphic Window นี้อาจจะปรากฏขึ้นมากกว่าหนึ่งหน้าต่างในเวลาเดียวกันก็ได้แล้วแต่คำสั่งที่กำหนดให้กับ MATLAB ซึ่งบนหน้าต่างนี้จะมี Menu Bar และอื่นๆอยู่ด้วย ลักษณะของ Graphic Window จะมีลักษณะตามรูปต่อไปนี้



ซึ่งในหน้าต่างนี้นอกจากจะใช้แสดงผลรูปภาพแล้ว เรายังสามารถใช้สร้าง Graphical User Interface เพื่อใช้ทำโปรแกรมที่มีการติดต่อกับผู้ใช้โดยใช้ปุ่มต่างๆ เหมือนกับโปรแกรมที่ทำงานภายใต้ windows ทั่วไปได้อีกด้วย นอกเหนือจากนี้ Graphic Window ยังมีเครื่องมือที่ช่วยในการเขียนกราฟของเราสะดวกขึ้น โดยเฉพาะใน version หลังๆ ของ MATLAB นี้ เราสามารถที่จะแก้ไข เพิ่มเติมกราฟที่เราที่ Graphic Window ได้ ไม่ว่าจะเป็นการเพิ่มตัวหนังสือ การเพิ่มเส้น การเพิ่มชื่อแกนหรือชื่อกราฟ การปรับเปลี่ยนมุมมอง ปรับเปลี่ยนทิศทางของแสงที่ฉายมาที่รูป และอื่นๆอีกมาก และใน version 6 นี้เราสามารถที่จะทำการ fit curve ที่เราเขียนลงไปบน Graphic Window นี้ได้อีกด้วย สำหรับรายละเอียดในการใช้เครื่องมือต่างๆ บน Graphic Window เราจะกล่าวถึงในบทที่เกี่ยวข้องกับการใช้ Graphic ต่อไป

1.4 คำสั่งเบื้องต้น

ในบทต่อไปเราจะได้ทราบถึงคำสั่งต่างๆที่มีอยู่มากมายใน MATLAB อย่างไรก็ตามในส่วนนี้เราขอกล่าวถึงคำสั่งพื้นฐานเบื้องต้นและการใช้แป้นพิมพ์บางส่วนที่ใช้เป็นพื้นฐานในการทำงานภายใต้ MATLAB สำหรับ คำสั่งต่างๆที่เราจะกล่าวถึงต่อไปในที่นี้นั้นจะเป็นคำสั่งตัวหนังสือที่เราจะใช้กับ MATLAB โดยผ่าน Command Window ซึ่งคำสั่งพื้นฐานจะประกอบด้วยคำสั่งต่อไปนี้ แต่ก่อนอื่นเนื่องจากเอกสารนี้เขียนขึ้นโดยมุ่งหวังว่าผู้อ่านจะศึกษาการใช้งานขั้นพื้นฐานของ MATLAB ด้วยตัวเอง ดังนั้นจึงขอทำความเข้าใจถึงรูปแบบตัวอักษรและสัญญลักษณ์ต่างๆที่ใช้ในเอกสารนี้ก่อน

❶ รูปแบบและสัญลักษณ์ในเอกสาร

เนื่องจากในเอกสารชุดนี้จำเป็นต้องมีการยกตัวอย่างอยู่เสมอเพื่ออธิบายการใช้งาน MATLAB ดังนั้นเพื่อไม่ให้ผู้อ่านเกิดการสับสน ผู้เรียบเรียงได้ใช้รูปแบบของตัวหนังสือให้มีความแตกต่างกันโดยมีการใช้สัญลักษณ์ดังนี้

- ตัวหนังสือที่ใช้อธิบายปกติจะใช้ตัวหนังสือแบบ Angsana New Thai ทั้งภาษาไทยและภาษาอังกฤษ
- สำหรับคำสั่งที่เป็น input เพื่อให้ MATLAB คำนวณจะใช้ตัวหนังสือแบบ **Courier New** ตัวหนา เช่น

  Airtime = B*C(1,2)
- สำหรับ output ที่ได้จาก MATLAB จะใช้ตัวหนังสือแบบ *Courier New* แบบตัวเอียง

  Airtime = 20
- เครื่องหมาย % เป็นเครื่องหมาย comment ซึ่ง MATLAB จะไม่สนใจที่จะทำคำสั่งหรือข้อความต่างๆที่อยู่หลังเครื่องหมายนี้แต่มีประโยชน์ในการเขียนโปรแกรมเพราะจะสามารถบันทึกข้อความต่างๆลงไปได้
- เครื่องหมาย ℂ แสดงถึงว่าเป็นข้อควรระวังหรือควรระลึกถึงในการใช้ MATLAB
- เครื่องหมาย ↵ หมายถึงว่าให้กด Enter key หลังจากจบคำสั่ง (ในบางกรณีที่ต้องกด Enter key อย่างชัดเจนอยู่แล้วอาจยกเว้นไม่ใส่เครื่องหมายนี้)

❷ คำสั่งเบื้องต้นที่ควรทราบในการใช้ MATLAB

คำสั่งเบื้องต้นประกอบด้วยคำสั่งต่อไปนี้

- **quit** หรือ **exit** เลิกการทำงานของ MATLAB
- **clc** ลบข้อความที่บรรจุอยู่ใน Command Window แต่ไม่มีการลบค่าตัวแปรใดๆ
- **clf** ลบรูปภาพที่บรรจุอยู่ใน Graphic Window
- **clear** ลบตัวแปรทุกตัวออกจากหน่วยความจำ
- **save** เป็นการรวบรวมค่าตัวแปรทุกตัวที่มีอยู่ในขณะนั้นบันทึกลงบน disk
- หากต้องการยกเลิกการคำนวณในขณะที่ MATLAB ยังทำการคำนวณไม่เรียบร้อยให้กดแป้น Ctrl และ c พร้อมกัน

ℂ **สำหรับ MATLAB การใช้คำสั่งหรือ function ใดๆก็ตามจะต้องใช้อักษรตัวพิมพ์เล็กเสมอ ไม่เช่นนั้นท่านจะได้ข้อความแสดงความผิดพลาด**

❸ Function Keys พิเศษ

ใน MATLAB มีการสำรองแป้นไว้สำหรับช่วยในการใช้งานให้ง่ายขึ้น โดยจะประกอบด้วย keys พิเศษต่างๆต่อไปนี้

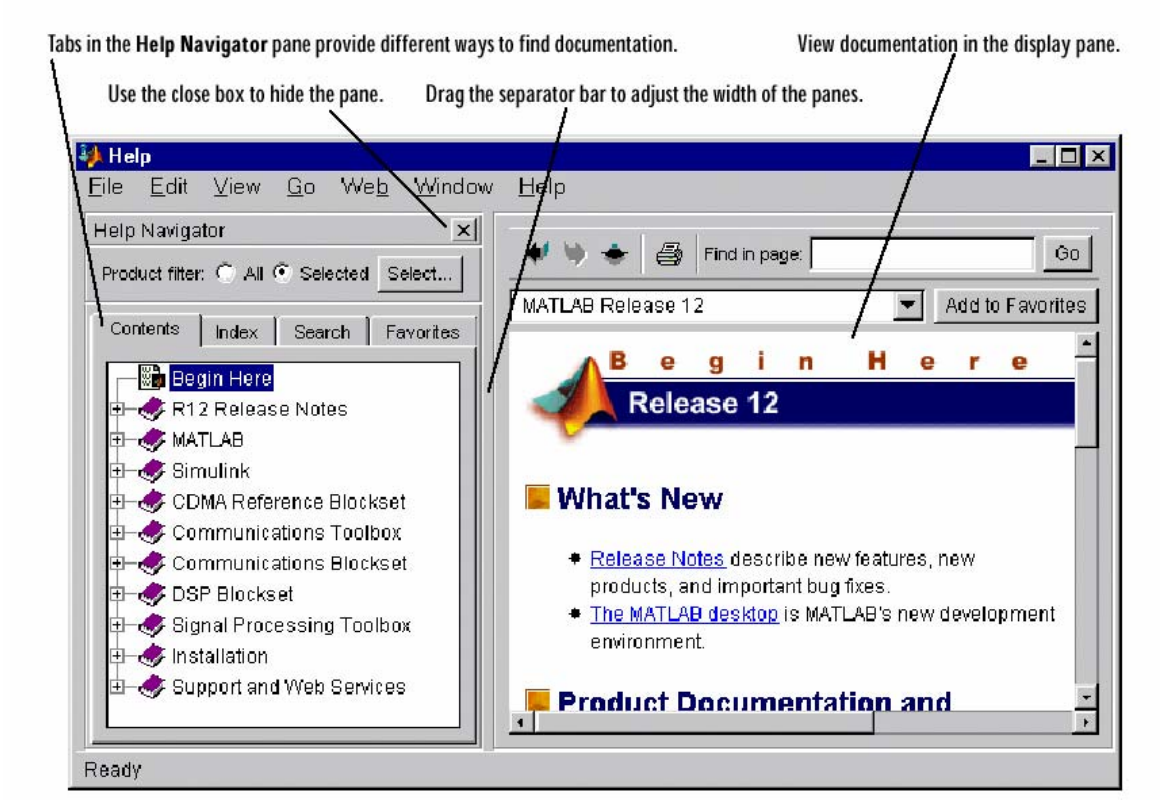
Ctrl-p หรือ ↑	ใช้เรียกคำสั่งที่ทำไปในครั้งที่ผ่านมา
Ctrl-n หรือ ↓	ใช้เรียกคำสั่งที่สั่งหลังจากคำสั่งที่กำลังสั่งอยู่
Ctrl-f หรือ →	เลื่อนไปทางขวา 1 ตัวอักษร
Ctrl-b หรือ ←	เลื่อนไปทางซ้าย 1 ตัวอักษร
Del หรือ **Backspace**	ลบตัวอักษรครั้งละ 1 ตัว
Ctrl-l หรือ **Ctrl**-←	เลื่อนไปทางซ้าย 1 คำ
Ctrl-r หรือ **Ctrl**-→	เลื่อนไปทางขวา 1คำ
Ctrl-a หรือแป้น **Home**	เลื่อนไปที่ตัวอักษรแรกของบรรทัด
แป้น **End**	เลื่อนไปที่ตัวอักษรสุดท้ายของบรรทัด
Ctrl-k	ลบทุกตัวอักษรจากจุดที่อยู่ไปถึงตัวสุดท้ายของบรรทัด

❹ การขอความช่วยเหลือในการใช้โปรแกรม

ถ้าในระหว่างการทำงานบน MATLAB แล้วมีปัญหาเกิดขึ้นในเรื่องของรูปแบบของการใช้คำสั่ง เราอาจใช้คำสั่ง help เพื่อช่วยการทำงานได้

- ใช้ **help** อย่างเดียวเพื่อให้ MATLAB แสดงชื่อ function ที่มีบรรจุอยู่ใน help
- ใช้ **help** แล้วตามด้วยชื่อ function เพื่อให้ MATLAB แสดงรายละเอียดของ function นั้น
- หากต้องการดูการสาธิตการทำงานของ MATLAB ให้ใช้คำสั่ง **demo**
- สำหรับการร้องขอ help อาจจะเรียกใช้จาก Menu bar ก็ได้
- Help Windows ซึ่งจะสะดวกในการค้นหามากขึ้น คำสั่งที่ใช้คือ **helpwin**

นอกเหนือจากนั้นใน version ใหม่นี้ยังได้มี Help Browser ที่สร้างขึ้นมาใน MATLAB เพื่อทำให้เราสามารถที่จะใช้เพื่อค้นหาและดูเอกสารที่เกี่ยวข้องกับ MATLAB ที่ผลิตขึ้นโดยบริษัท MathWorks ซึ่ง Help Browser นี้เป็นการนำ MATLAB Desktop นี้รวมเข้ากับ Web Browser เพื่อที่จะแสดงผลเอกสารประเภท HTML ซึ่งการทำงานของ Help Browser ก็จะเหมือนกับโปรแกรม Web Browser ทั่วๆไปนั่นเอง ลักษณะของ Help Browser จะมีลักษณะตามรูป



สำหรับรายละเอียดของการใช้ Help ภายใต้ Help Browser นั้นสามารถที่จะแยกเป็นส่วนการแสดงข้อมูลที่ช่วยในการใช้ MATLAB ออกเป็นส่วนๆได้หลายส่วน และเนื่องจาก Help นี้อยู่ในรูปแบบของ html ดังนั้นการใช้งานก็จะมีลักษณะเหมือนการใช้ Browser ทั่วๆไป เพียงแต่ข้อมูลทั้งหมดนั้นอยู่ภายในเครื่องของเรา (ถ้าเราเลือก install แบบนั้น) และเนื่องจากการจัดรูปของ Help อยู่ในลักษณะของ browser เราจึงสามารถที่จะลงลึกเข้าไปในข้อมูลได้เรื่อๆ อย่างไรก็ตามหัวข้อที่สำคัญภายใต้หน้า help หน้าแรกที่จะพาเราท่องไปที่ส่วนต่างๆ จะมีที่สำคัญอยู่ดังนี้

- ภายใต้ Using MATLAB จะมีข้อมูลที่จัดกลุ่มเป็นส่วนประกอบที่สำคัญต่างๆ ดังนี้
  - “Development Environment” เป็นข้อมูลทั้งหมดเกี่ยวกับการใช้ MATLAB desktop
  - “Mathematics” อธิบายถึงขีดความสามารถในการทำงานด้านคณิตศาสตร์และสถิติของ MATLAB
  - “Programming and Data Types” อธิบายถึงวิธีการสร้างฟังก์ชันและการใช้ภาษา MATLAB
  - “Graphics” อธิบายการเขียนกราฟและความสามารถของการทำงานด้านกราฟฟิกส์ของ MATLAB
  - “3-D Visualization” เป็นการแนะนำวิธีการใช้คำสั่ง view เพื่อกำหนดมุมมอง คำสั่ง lighting เพื่อกำหนดทิศทางและขนาดของแสงในการมองภาพ 3 มิติ และการกำหนดลักษณะภาพสามมิติอื่นๆ เพื่อให้ได้ภาพที่ซับซ้อนออกมาในลักษณะที่เราต้องการ

- “External Interfaces/API” อธิบายวิธีการติดต่อกับโปรแกรมที่เขียนขึ้นโดยภาษา C และ Fortran รวมถึงการติต่อกับภาษา Java, ไฟล์ข้อมูล, serial port I/O, ActiveX และ DDE เป็นจ้น
- “Creating Graphical User Interfaces” อธิบายถึงวิธีการใช้เครื่องมือและการสร้างโปรแกรม MATLAB ประเภท graphical user interface

ˆ ภายใต้ Reference ได้มีการจัดเอกสารต่างๆ ดังนี้

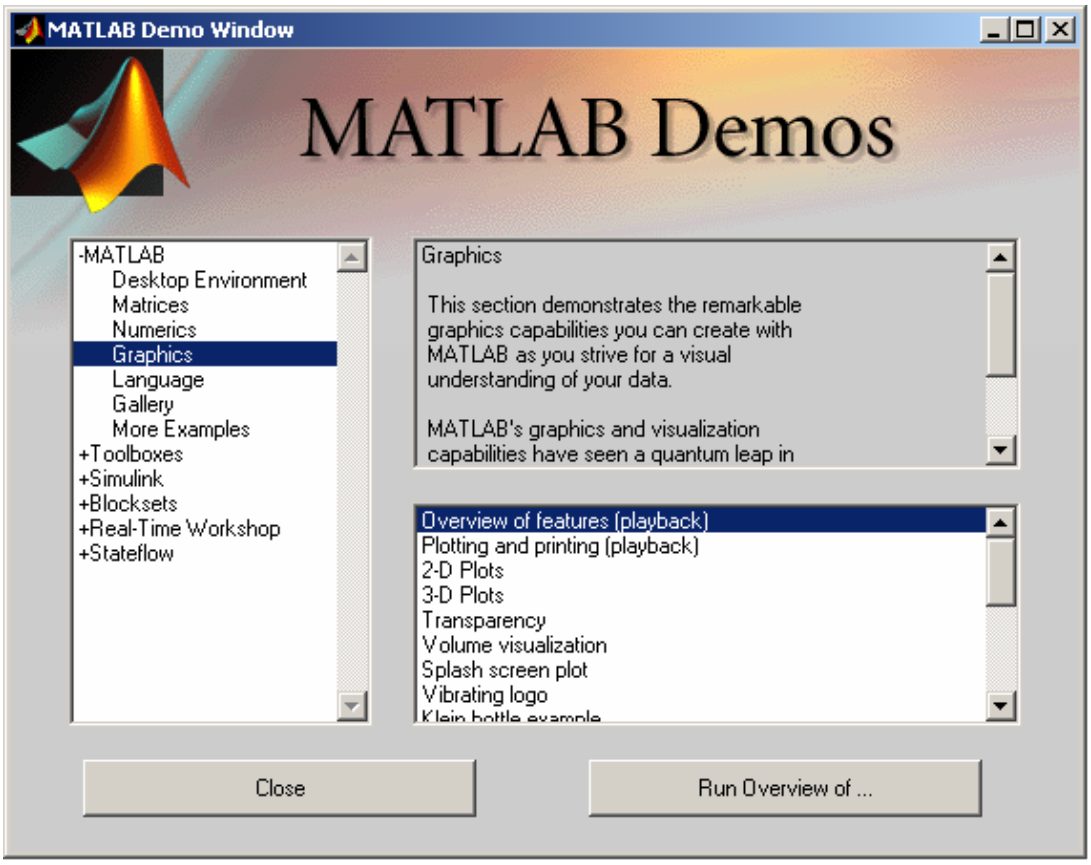
- “MATLAB Function Reference” อธิบายถึงฟังก์ชันที่สำคัญของ MATLAB ทั้งหมด โดยจะบอกถึงวิธีการป้อนข้อมูลให้กับฟังก์ชัน การกำหนด syntax และระเบียบวิธีการทางคณิตศาสตร์ที่ฟังก์ชันนั้นใช้ เราสามารถที่จะเลือกวิธีการนำเสนอได้ทั้งให้แสดงฟังก์ชันแบบ“Function By Category” คือเรียงตามหมวดหมู่ของฟังก์ชันนั้น หรือ “Alphabetical List of Functions” คือเรียงตามลำดับอักษร
- “External Interfaces/API Reference” อธิบายฟังก์ชันของ MATLAB ที่เกี่ยวข้องกับการเชื่อมต่อกับภายนอก วิธีการเรียกโปรแกรมต่างๆ ค่าที่จะได้จากการคำนวณและตัวอย่างต่างๆที่เกี่ยวข้อง
- นอกจากนี้ ยังรวม ซึ่งทำให้เราสามารถที่จะเข้าถึงคุณสมบัติของวัตถุที่เป็นรูปภาพต่างๆ ได้โดยง่าย รายละเอียดเกี่ยวกับกราฟฟิกส์ของ MATLAB นั้นมีค่อนข้างมาก ซึ่งเราอาจต้องทำความเข้าในในเบื้องต้นเสียก่อน จึงจะสามารถใช้งานในส่วนนี้ได้อย่างจริงจัง

˜ MATLAB Demos จะเป็นส่วนที่แสดงตัวอย่างทั้งหมดที่มากับโปรแกรม จำนวนตัวอย่างที่แสดงนั้นก็จะขึ้นอยู่กับจำนวนToolbox ที่เราได้ติดตั้งไว้บนเครื่องของเราด้วย ตัวอย่างเหล่านี้จะช่วยให้เราเข้าใจได้ว่าเราสามารถที่จะใช้ MATLAB ทำงานอะไรให้กับเราได้บ้าง

การเรียกดูตัวอย่างที่มีใน MATLAB สามารถทำได้โดยสั่ง

```
»demo
```

หลังจากนั้นก็จะปรากฏหน้าต่าง demo ขึ้นมีลักษณะดังรูปข้างล่างนี้ เราสามารถที่จะเลือก หัวข้อที่ต้องการจะดูตัวอย่างจากหน้าต่างย่อยทางซ้ายมือ จากนั้นจึงเลือกตัวอย่างทางหน้าต่างขวามืออีกครั้งหนึ่ง แล้วจึงกดปุ่มเลือก Run…… เพื่อดูตัวอย่างการทำงานของ MATLAB ต่อไป



1.5 สรุป

ที่เราได้กล่าวมาทั้งหมดในบทนี้เป็นการแนะนำส่วนประกอบพื้นฐานของ MATLAB ซึ่งเราจะต้องนำไปใช้ต่อไป สำหรับส่วนประกอบบางส่วนที่ไม่ได้กล่าวถึงในที่นี้เราจะกล่าวถึงเมื่อมีความจำเป็นต้องใช้

สำหรับใน 3 บทที่เราจะกล่าวถึงต่อไปนี้จะเป็นการแนะนำพื้นฐานของการใช้ MATLAB เพื่อการคำนวณพื้นฐาน ซึ่งคำสั่งต่างๆ นั้นบางคำสั่งคำสั่งพื้นฐานที่ไม่ได้มีการปรับเปลี่ยนไปตามการปรับโปรแกรม MATLAB แต่อาจจะมีบางส่วนที่สร้างขึ้นมาใหม่สำหรับใน MATLAB 6 พร้อมกับยกเลิกคำสั่งบางคำสั่งไปแล้ว ซึ่งเราจะพยายามกล่าวถึงคำสั่งเหล่านั้นเมื่อถึงเวลา

บทที่ 2 การใช้ MATLAB เบื้องต้น

2.1 แนะนำ Scalars, Vectors and Matrices

ในการแก้ปัญหาทางด้านวิศวกรรม เราพบว่ากลุ่มข้อมูลที่จะต้องพิจารณาอาจจะเป็นตัวเลขเพียงตัวเดียว หรือ scalar หรืออาจเป็นกลุ่มของข้อมูลที่จัดอยู่ในรูปของ vector หรือ matrix ก็ได้ สำหรับ MATLAB แล้วโครงสร้างข้อมูลทั้งหมดจะได้รับการจัดให้อยู่ในรูปของ matrix ทั้งหมดคือ

- จะถือว่า scalar คือ matrix ขนาด [1 x 1]
- ส่วน vector สามารถจะจัดอยู่ในรูป row matrix [n x 1] หรือ column matrix [1 x n] ซึ่งนิยมเรียกอีกอย่างหนึ่งว่า row vector และ column vector ตามลำดับ

การตั้งชื่อตัวแปรใน MATLAB มีเงื่อนไขต่อไปนี้

- ชื่อตัวแปรต้องขึ้นต้นด้วยตัวอักษร
- ชื่อตัวแปรจะบรรจุด้วยตัวอักษร ตัวเลข และ underscore (_) ได้
- ชื่อตัวแปรจะยาวเพียงใดก็ได้แต่ MATLAB จะจดจำชื่อตัวแปรและแยกตัวแปรสองตัวออกจากกันด้วยอักษรเพียง 19 ตัวแรกเท่านั้น
- MATLAB เป็น case-sensitive variable ดังนั้นการใช้ตัวอักษรตัวใหญ่และตัวเล็กจะเก็บไว้ในหน่วยความจำต่างกัน เช่นตัวแปร xA, xa, XA จะเป็นตัวแปรคนละตัวกัน

ควรตั้งชื่อตัวแปรให้เหมาะสมเพื่อช่วยให้จำได้ว่าตัวแปรเหล่านั้นแทนค่าของอะไร เพื่อไม่ให้เกิดการสับสนเมื่อต้องการที่จะแก้ไขโปรแกรม

❶ การกำหนดค่าตัวแปร

การกำหนดค่าตัวแปร หรือ assignment statement จะใช้เครื่องหมาย '=' ซึ่งมีส่วนประกอบดังนี้

ตัวแปร = ค่าของตัวแปร

- ถ้าค่าของตัวแปรเป็นตัวเลขตัวเดียว อาจกำหนดเป็นตัวเลขไปเลยก็ได้ เช่นถ้าต้องการให้ตัวแปร A มีค่า 1

```
A = 1
```

- ถ้าค่าของตัวแปรเป็น vector หรือ matrix ต้องกำหนดในเครื่องหมายวงเล็บใหญ่ [ ]
  - ค่าที่จะกำหนดในลักษณะของแถวนอน (row) ค่าแต่ละ elements ใน row เดียวกัน อาจจะแยกจากกันด้วยช่องว่าง (space) หรือ comma ก็ได้เช่น

```
A = [1, 2, 3]
```

หรือ

```
A = [1 2 3]
```

เมื่อมีการกำหนดค่า MATLAB จะตอบกลับมาซึ่งเหมือนกับการแสดงผลหรือการรับทราบค่า ซึ่งในกรณีนี้จะได้

```
A =

   1   2   3
```

ซึ่งจะให้ matrix A ขนาด 1 row x 3 column เหมือนกัน

- ใช้เครื่องหมาย sami-colon แยก row ออกจากกัน เช่นถ้าต้องการ matrix $a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ จะใช้

```
a = [1 2; 3 4]
a =
   1   2
   3   4
```

- การแยกแต่ละ row ออกจากกันสามารถกระทำได้อีกวิธีหนึ่งคือแยกบรรทัด ด้วยการใช้ enter เช่น

```
a =[1, 2
3, 4]
```

ก็จะให้ค่าเหมือนกับกรณีที่ผ่านมา

- ในกรณีที่ row ยาวมาก จนต้องเขียนต่อบรรทัดใหม่ให้ใช้สัญลักษณ์ ellipsis คือ comma แล้วตามด้วยจุด 3 จุด เช่น

```
a = [1, 2, 3,...
4 ,5, 6]
```

จะได้ a เป็น matrix ขนาด [1 x 6]

- MATLAB ยินยอมให้มีการกำหนดค่า matrix ใหม่โดยใช้ matrix เดิมเข้าช่วย เช่น

```
a = [1 2 3]
b = [0 a]
```

จะได้ b เป็น `b = [0 1 2 3]`

- โดยทั่วไป เมื่อมีการกำหนดค่าตัวแปร MATLAB จะมีการแสดงข้อมูลกลับ (echo) เพื่อให้ผู้ใช้รับทราบการยืนยันข้อมูลที่ส่งเข้าสู่หน่วยความจำ เช่น ถ้าพิมพ์

```
a = [1, 2, 3]
```

จะพบว่า MATLAB ตอบกลับว่า

```
a =
     1     2     3
```

ตามที่ได้ยกตัวอย่างมาแล้ว หากว่าไม่ต้องการให้ MATLAB ยืนยันข้อมูลกลับมา ให้ใช้ semi-colon ลงด้านท้ายของข้อมูล เช่น

```
a = [1, 2, 3];
```

ซึ่งในกรณีนี้จะไม่มีการ echo ของข้อมูล หากว่าไม่มีการกำหนดค่าตัวแปรที่จะเป็นคำตอบ MATLAB จะตั้งชื่อตัวแปรนั้นอัตโนมัติโดยใช้ตัวแปร *ans*

ในการหัดใช้ MATLAB ในครั้งแรกๆท่านอาจจะไม่ชำนาญจึงอาจทำให้การป้อนข้อมูลหรือคำสั่งผิดพลาดได้ ดังนั้นจึงควรให้เครื่อง echo ค่ากลับมาก่อน เมื่อท่านชำนาญแล้วจึงงดใช้ echo

- เราสามารถเปลี่ยนค่า element ของ matrix ได้โดยใช้ index กำหนดค่านั้น ๆ เช่นหากเรากำหนด

```
a = [1, 2; 3, 4]
```

ถ้าต้องการเปลี่ยนค่าที่ row 1 column 2 จาก 2 เป็น 5 สามารถทำได้โดย

```
a(1,2) = 5
```

จะได้ว่า matrix **a** เป็น

```
a =
   1   5
   3   4
```

- ถ้า matrix เป็น single row หรือ single column matrix อาจใช้ index ตัวเดียวได้ เช่นถ้า matrix

```
a = [1 2 3]
```

เราอาจกำหนด

```
a(3) = 5
```

เราจะได้ **a** มีค่าเป็น

```
a = [1 2 5]
```

- ถ้า **a** =[1 2 3] แล้วเรากำหนด **a**(7) = 5 เราจะได้ **a** มีค่าเป็น

```
a =[1 2 0 0 0 0 5]
```

ซึ่ง MATLAB จะเพิ่ม element จาก 3 เป็น 7 ทันที แล้วกำหนดค่า element ที่เพิ่มขึ้นโดยไม่มีการกำหนดค่าให้เป็นศูนย์โดยอัตโนมัติ

- การดูค่าตัวแปรที่บรรจุอยู่ในความจำของ MATLAB อาจใช้ คำสั่งได้ 3 คำสั่งคือ

  who จะบอกชื่อตัวแปร ทั้งหมดที่อยู่ในหน่วยความจำ

  whos จะบอกทั้งขนาด, จำนวน byte ที่ใช้ในหน่วยความจำ และ ชนิดของตัวแรปแต่ละตัวที่มีอยู่ทั้งหมดในขณะนั้น

  size จะบอกขนาดของตัวแปรที่เป็นทั้ง matrix และ array เช่นถ้า $a = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, จะได้

```
size(a)
ans =
     2     2
```

❷ Colon operator

Colon operator เป็น operator ที่มีประโยชน์ในการกำหนดค่า matrix มาก โดยมีหน้าที่สำคัญดังต่อไปนี้

- เราสามารถกำหนดค่า vector จาก matrix ที่มีการกำหนดค่าเรียบร้อยแล้ว เช่นหากเบื้องต้นกำหนดให้

```
x =[1 2 3; 4 5 6; 9 10 0];
```

ถ้าต้องการให้ตัวแปรใหม่ **a** มีค่าเท่ากับค่าทุกค่าใน column ที่ 1 ของ x สามารถใช้

```
a = x(:, 1)
```

นั่นคือ

```
a =
     1
     4
     9
```

หรือ **b = x(2,:)** หมายความว่า b มีค่าเท่ากับทุกค่าใน row ที่ 2 ของ x

นั่นคือ `b = [4 5 6]`

- Colon operator สามารถใช้สร้าง matrix ใหม่ขึ้นมา เช่น ถ้าใช้ colon แยกจำนวนเต็ม 2 จำนวน เช่น

k = 1:5 หมายถึงการสร้างให้ k มีค่าเป็น row vector มีค่าเป็นหรือเหมือนกับการใช้คำสั่ง

```
k = [1, 2, 3, 4, 5]
```

- ถ้าใช้ colon แยกจำนวน 3 จำนวนหมายถึงให้มีค่าเริ่มจากค่าแรกเพิ่มขึ้นครั้งละเท่ากับค่าที่ 2 และสิ้นสุดเมื่อค่าเท่ากับค่าที่ 3 เช่น

  k = 1:0.5:5 จะได้ row vector เหมือนกับการใช้คำสั่ง

  k = [1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]

- สามารถใช้ colon ในการนับถอยหลังได้เช่น

  k = 5:-0.5:1 จะได้ผลเหมือนกับ

  k = [5, 4.5, 4, 3.5, 3, 2.5, 2, 1.5, 1]

- สามารถใช้ colon operator แยก submatrix ได้ เช่น

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 9 & 7 & 8 \end{bmatrix}$$ ใช้คำสั่ง

```
a=[1 2 3;4 5 6;9 7 8]
a =
     1     2     3
     4     5     6
     9     7     8
```

หากต้องการหา minor ของ a(1, 1) หรือคือการตัด row 1 column 1 ออก อาจหาได้จาก

```
b = a(2:3,2:3);
```

คือให้ b มีค่าเท่ากับ row 2 ถึง row 3 และ column 2 ถึง column 3 ของ a นั่นคือ หรือตัวอย่างเช่น

```
c = a(1:2,1) จะได้
c =
     1
     4
```

```
d = a(:,2:3)      จะได้
d =
     2     3
     5     6
     7     8
```

- ถ้าใช้ a(:) จะได้ matrix column เดียวที่เริ่มต้นด้วย column ที่ 1 ของ a ต่อด้วย column ที่ 2, 3... ต่อไปเช่น

```
e = a(:)
e =
   1
   4
   9
   2
   5
   7
   3
```

6

8

- Matrix ว่าง คือ matrix ที่มีขนาด 0 x 0 เช่น **a = [];** หรือ **b = 10:1:0;** ซึ่ง matrix ว่างนี้แตกต่างจาก matrix ที่มี element ทุกค่าเป็นศูนย์

- Transpose Matrix คือการเปลี่ยน row เป็น column และ column เป็น row

เช่น $a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ ดังนั้น $a^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$

สำหรับ MATLAB เครื่องหมาย transpose คือเครื่องหมาย (') เช่น **a'** ตัวอย่างเช่น

```
a = [1 2 3;4 5 6]
a =
     1     2     3
     4     5     6

b = a'
b =
     1     4
     2     5
     3     6
```

❸ การเก็บค่าตัวแปร

ตัวแปร 1 ตัว จะสามารถเก็บค่าได้เพียง 1 ค่า และจะเก็บค่าสุดท้ายเท่านั้น เช่น ถ้ากำหนด

```
a = [1 2 3];
a = [3 4];
```

MATLAB จะใช้ค่า **a = [3 4]** เท่านั้น หรือ

```
a = [1 2 3];
a = [5 a];
```

MATLAB จะใช้ค่าเหมือนกับ

```
A = [5 1 2 3];
```

❹ ค่าพิเศษของ MATLAB

ค่าต่อไปนี้เป็นค่าที่กำหนดขึ้นพิเศษ เป็นค่าที่กำหนดไว้แต่ครั้งแรกเมื่อเรียก MATLAB ขึ้นมาใช้ และไม่ควรจะมีการใช้ชื่อตัวแปรเหล่านี้กำหนดเป็นค่าอื่นเพราะจะทำให้สับสนได้ ค่าพิเศษมีดังต่อไปนี้

pi ใช้แทนค่า π ซึ่งมีค่าประมาณ 3.14159

i และ **j** แทน imaginary valve ซึ่งมีค่า $\sqrt{-1}$

Inf แทน infinity ส่วนมากจะเกิดจากการหารจำนวนใดๆ ด้วยศูนย์

NaN แทน Not-a-Number แทนค่าที่ทางคณิตศาสตร์ไม่ได้นิยาม (undefined) เช่น $0 \div 0$

clock เวลาปัจจุบันที่มีการเรียกค่านี้ใช้ ซึ่งเป็นเวลาตามนาฬิกาของ Computer ที่กำลังทำงานอยู่

date วันใน format วัน-เดือน-ปี เช่น 1-jan-97

eps แทน floating-point precision ของเครื่อง computer ที่กำลังทำงานอยู่

ans แทนค่าที่คำนวณครั้งหลังสุดที่ไม่มีการกำหนดค่าอื่นใด

❺ Matrix พิเศษของ MATLAB

Matrix พิเศษที่สามารถให้ MATLAB สร้างขึ้น ประกอบด้วย matrix หลายแบบด้วยกันคือ

rand(m,n) เป็นการสร้าง matrix ขนาด [m x n] ที่แต่ละ element เป็นค่าสุ่มจากตัวเลขในช่วง 0 – 1

หรือใช้ **rand(m)** สำหรับ square matrix

hilb(N) เป็นการสร้าง Hilbert matrix คือ matrix ขนาด [N x N] ที่ element มีค่าเท่ากับ $\frac{1}{(i+j-1)}$

magic(N) เป็นการสร้าง square matrix ขนาด [N x N] จากจำนวนเต็ม 1 จนถึง N^2 โดยที่ผลรวมของ element ในแตละ row, column และใน diagonal จะมีค่าเท่ากัน สำหรับ N ที่มีค่าเท่ากับ 1, 3, 4, 5,…

zeros(m,n) เป็นการสร้าง matrix ขนาด [m x n] ที่มีทุก element เท่ากับศูนย์

ones(m,n) เป็นการสร้าง matrix ขนาด [m x n] ที่มีทุก element มีค่าเท่ากับ 1

eye(m,n) เป็นการสร้าง matrix ขนาด [m x n] ที่มี element ในแนว diagonal เท่ากับ 1 นอกนั้นเป็นศูนย์ ซึ่ง matrix นี้ ไม่จำเป็นต้องเป็น square matrix

linspace(a,b,n)

เป็นการสร้าง row vector ขนาด 1 x n โดยค่าแรกจะเท่ากับ a และค่าสุดท้ายจะเป็น b โดยค่อยๆ เพิ่มขึ้นเท่าๆ กัน จำนวน n ค่า ถ้าหากไม่กำหนดค่า n ไว้ MATLAB จะใช้ค่า n เท่ากับ 100

logspace(a,b,n)

เป็นการสร้าง row vector ขนาด 1 x n โดยค่าแรกจะเท่ากับ a และค่าสุดท้ายจะเป็น b โดยค่อยๆ เพิ่มขึ้นตาม logarithmic จำนวน n ค่า หากไม่กำหนดค่า n ไว้ MATLAB จะใช้ค่า n เท่ากับ 100

- ในกรณีที่จะสร้าง square matrix อาจไม่จำเป็นต้องใส่ค่า m และ n ทั้งสองค่าอาจใช้เพียงค่าเดียวก็ได้ เช่น

```
ones(2)
```
จะได้ matrix $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

ตัวอย่างอื่น ๆ เช่น

```
a=eye(3,4)
```
จะได้

```
a =
     1     0     0     0
     0     1     0     0
     0     0     1     0
```

```
linspace(1,3,10)
ans =
Columns 1 through 7
1.0000   1.2222  1.4444  1.6667   1.8889    2.1111    2.3333
Columns 8 through 10
2.5556    2.7778    3.0000
```

❻ String Variable

การกำหนดค่าตัวแปรนอกจะกำหนดเป็นตัวเลขแล้ว เรายังสามารถกำหนดเป็นตัวหนังสือได้อีกด้วย โดยการกำหนดค่าตัวแปรที่เป็นตัวหนังสือหรือที่เรียกว่า string จะต้องใส่ค่าที่เราต้องการกำหนดในเครื่องหมาย quotation "'" ดังตัวอย่างเช่น

```
Str1='Tom';
Str2='Pookie';
```

การกำหนดค่าตัวแปรเป็น string จะมีประโยชน์ในกรณีที่ต้องส่งค่าตัวแปรที่เป็นตัวหนังสือไปให้ function เพื่อใช้ในการประมวลผลหรือแสดงคำสั่งที่ต้องการ

สำหรับการกำหนดตัวแปรให้เป็น matrix โดยมีค่าแต่ละ element เป็น string นั้นมีข้อจำกัดอย่างหนึ่งคือทุกตัวแปรจะต้องมีความยาวเท่ากัน ซึ่งข้อจำกัดนี้มาจากว่า matrix ต้องมีจำนวน column ในแต่ละ row เท่ากัน ซึ่งจากข้อจำกัดนี้ทำให้เราอาจต้องเพิ่ม blank space มาขั้นในตัวแปรที่เป็น string แต่ละตัว

2.2 Matrix Operation

❶ Dot Product

Dot Product หรือ inner product เป็นการคำนวณหาค่า scalar value ของ vector ที่มีขนาดเท่ากัน ซึ่งถ้า A และ B เป็น vector ที่มีขนาดเท่ากันจะได้

$$A \bullet B = \sum_{i=1}^{N} a_i b_i$$ เมื่อ N เป็นจำนวน element ของ A และ B

ตัวอย่างเช่น A = [1 2 3] B = [4 5 6] จะได้ว่าถ้า C = A•B จะได้ (1)(4) + (2)(5) + (3)(6) = 4+10+18 = 32

MATLAB จะใช้คำสั่ง

```
C = dot(A,B)
```

dot(A,B) คำนวณหา dot product ของ A และ B ถ้า A และ B เป็น matrix dot product จะเป็น row vector ที่มี element ในแต่ละ column บรรจุค่า dot product ในแต่ละ column ของ A และ B

ตัวอย่างเช่น

```
A=[1 2 3];B=[4 5 6];
```

ดังนั้น

```
dot(A,B)
ans =
    32
```

❷ การคูณ Matrix

การที่ Matrix A และ B จะคูณกันได้ matrix ตัวแรกจะต้องมีจำนวน column เท่ากับจำนวน row ของตัวที่สอง และผลจะได้ matrix ที่มีขนาดเท่ากับ row ของตัวแรก และ column ของตัวหลัง เช่น A มีขนาด [m x n] และ B มีขนาด [p x q] AB จะมีค่าก็ต่อเมื่อ n = p และจะได้ matrix ขนาด m x q ดังนั้นโดยทั่วไป AB ≠ BA หรือไม่มีการสลับที่การคูณของ matrix สำหรับ MATLAB ถ้า C = AB จะใช้

```
C =A *B
```

ถ้าจำนวน column A ไม่เท่ากับจำนวน row ของ B MATLAB จะแสดงข้อความบอกความผิดพลาด

❸ การคูณ Matrix ด้วย Scalar

การคูณ matrix ด้วย scalar นั้นจะเป็นการคูณทุก element ของ matrix ด้วยค่าคงที่ เช่น A=[1 2]; จะได้

```
C =2*A
C =
     2     4
```

❹ การบวก-ลบ Matrix

การที่ matrix จะบวก/ลบกันได้ต้องมีขนาดเท่ากัน และการบวก/ลบจะกระทำโดยบวก/ลบแต่ละ element ของ matrix

```
A=[1 2 3];
B=[4 5 6];
C=A-B
C =
    -3    -3    -3
```

❺ การยกกำลัง Matrix

Matrix ที่จะยกกำลังได้ต้องเป็น square matrix เท่านั้น โดย

$A^2 = AA$

MATLAB จะเขียน A^2 เป็น **A^2** ดังนั้น

A^2 จะเท่ากับ **A*A**

A^3 จะเท่ากับ **A*A*A**

❻ Matrix Polynomial

สำหรับ polynomial function ของ x จะอยู่ในรูป

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots a_1 x + a_0$$

โดย $a_n, a_{n-1}, \ldots$ เป็นสัมประสิทธิของ polynomial ถ้า x เป็น matrix จะเรียกว่า matrix polynomial โดยกำลังของ matrix ก็จะใช้หลักการของการยกกำลัง matrix สำหรับ MATLAB สามารถจะเขียน polynomial ของ matrix โดยใช้ function polyvalm

`polyvalm (a,x)` เป็นการหาค่า polynomial ของ matrix x โดยใช้สัมประสิทธิ์ a ซึ่ง x เป็น square matrix และ a เป็น row vector

สำหรับ operation ของ Matrix ในทางคณิตศาสตร์เบื้องต้นจะมีประมาณเท่าที่กล่าวมาแล้ว แต่สำหรับ MATLAB จะมี operation ของ Matrix แบบอื่นๆ อีกมากมาย ทั้งที่มีนิยามตามคณิตศาสตร์และไม่มีในนิยาม แต่สร้างขึ้นมาเพื่อให้สะดวกแก่การใช้งาน โดยเฉพาะการหาร การทำ operation ของ elements ซึ่งจะกล่าวในหัวข้ออื่นต่อไป

2.3 Matrix Functions

❶ Matrix Inverse

ถ้า A เป็น square matrix จะได้ว่า Inverse Matrix ของ A นิยมเขียน A^{-1} จะนิยามโดย

$$AA^{-1} = A^{-1}A = I$$

เมื่อ I เป็น identity matrix สำหรับ matrix ที่ไม่สามารถหาค่า inverse ของ matrix ได้จะเรียกว่า singular matrix ส่วน matrix ที่หาค่า inverse ได้ จะเรียก nonsingular matrix การหาค่า inverse matrix ที่มีขนาดใหญ่เป็นเรื่องยากในทางปฏิบัติ โดยเฉพาะการคำนวณตามหลักของพีชคณิต ดังนั้นส่วนใหญ่ในโปรแกรมคอมพิวเตอร์นิยมใช้วิธีเชิงตัวเลขมากกว่า สำหรับ MATLAB ก็เช่นกันโดยคำสั่งที่ใช้จะใช้คำสั่ง inv

inv(a) เป็นการคำนวณ inverse ของ matrix a ถ้า a เป็น singular matrix หรือเป็น matrix ที่ไม่มี inverse MATLAB จะแสดงข้อความบอกความผิดพลาด

ตัวอย่างเช่น

```
A=[1 4;8 9];
inv(A)
ans =
   -0.3913    0.1739
    0.3478   -0.0435
```

❷ Rank of Matrix

ถ้าแต่ละ row ของ matrix แทนแต่ละสมการในพีชคณิตเชิงเส้น rank ของ matrix คือจำนวนสมการที่ไม่ขึ้นต่อกันของระบบสมการนั้น เช่น

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \end{bmatrix}$$

จะเห็นว่า row ที่ 2 เป็น 2 เท่าของ row แรก ดังนั้น row ทั้งสองจะขึ้นต่อกัน ทำให้ a นี้มี rank เท่ากับ 1 โดยทั่วไปแล้ว matrix ที่มี rank น้อยกว่าจำนวน row จะเป็น singular matrix สำหรับ MATLAB จะหา rank จากคำสั่ง rank

rank(a) การคำนวณหา rank ของ matrix a

ตัวอย่างเช่น

```
a=[1 2 3;2 4 6];
rank(a)
ans =
     1
```

❸ Determinant of Matrix

Determinant ของ square matrix เป็นการหาค่า scalar ของ matrix คำสั่งการหาค่า determinant ของ square matrix a ใน MATLAB คือ

det(a) หาค่า determinant ของ a

สำหรับ matrix มี determinate เท่ากับศูนย์ matrix นั้นจะเป็น singular matrix การหา determinant โดย MATLAB มีตัวอย่างเช่น

```
a=[1 2 3; 2 7 6; 5 4 8];
det(a)
ans =
   -21
```

❹ การหาร Matrix

ความจริงแล้ว matrix ไม่มีคุณสมบัติของการหาร แต่ MATLAB ได้สร้าง function นี้ขึ้นมาเพื่อสะดวกในการแก้ ระบบสมการ linear equations พิจารณา

$$Ax = B$$

ดังนั้น $x = A^{-1}B$

ซึ่ง MATLAB จะใช้ สัญลักษณ์ **x=A\B**

$$\therefore \text{A\B} = \text{A}^{-1}\text{B}$$

หากพิจารณา

$$xA = B$$

จะได้ $x = BA^{-1}$

ซึ่ง MATLAB จะใช้ **B/A**

ดังนั้น $B/A = BA^{-1}$

- สำหรับ เครื่องหมาย / ใน MATLAB จะเรียก left matrix division
- สำหรับ เครื่องหมาย \ ใน MATLAB จะเรียก right matrix division
- วิธีเชิงตัวเลขที่ใช้ในการคำนวณหาค่า x MATLAB จะใช้ Gauss elimination numerical Technique

สำหรับตัวอย่างในกรณีนี้จะอยู่ในหัวข้อการหาค่าคำตอบของระบบสมการเชิงเส้น ซึ่งจะกล่าวถึงวิธีการนี้อย่างละเอียด

❺ Matrix Decomposition

โดยทั่วไปแล้วเราสามารถจะแยก matrix A ออกเป็นผลคูณของ matrix 2 matrix ซึ่งเรียกว่า Decomposition หรือ Factorization ของ matrix ในปัญหาทางคณิตศาสตร์หลาย ๆ กรณี หากเราแยก matrix ออกแล้ว จะสามารถลดการคำนวณต่างๆ ให้มีขั้นตอนน้อยลงได้ การ decomposed matrix ที่นิยมทำกันมี 3 วิธีคือ

① **LU-decomposition** หรือบางครั้งเรียก Triangular Factorization คือการแยก matrix ออกเป็นผลคูณของสอง matrix โดยแยกเป็น lower triangular matrix และ upper triangular matrix

- Lower Triangular matrix, L, คือ matrix ที่มีค่าของ elements ที่อยู่ด้านบนของ main diagonal เป็นศูนย์ (มีค่าเฉพาะที่อยู่ต่ำกว่า main diagonal) ตัวอย่างเช่น

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 2 & 0 \\ 5 & 6 & 3 \end{bmatrix}$$

- Upper Triangular matrix, U คือ matrix ที่มีค่าของ elements ที่อยู่ด้านล่างของ main diagonal เป็นศูนย์ (มีค่าเฉพาะที่อยู่สูงกว่า main diagonal) ตัวอย่างเช่น

$$U = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{bmatrix}$$

ดังนั้น สำหรับการแยก A ออกเป็นผลคูณของสอง matrix หรือ

$$A = LU$$

สำหรับ MATLAB จะใช้

[L,U]= lu(A) คำนวณหา matrix ที่ได้จากการแยก A ออกเป็น Lower triangular L และ upper triangular U ซึ่งทำให้ LU = A และ Matrix A จะต้องเป็น Square Matrix เท่านั้น

ตัวอย่างเช่น

```
A=[1 3 2;-2 -6 1;2 5 7];
[L,U]=lu(A)
LD =
   -0.5000         0    1.0000
    1.0000         0         0
   -1.0000    1.0000         0

UD =
   -2.0000   -6.0000    1.0000
         0   -1.0000    8.0000
         0         0    2.5000
```

LU-decomposition อาจหาคำตอบได้ไม่เหมือนกับการแยกด้วยมือ แต่นั้นไม่สำคัญเนื่องจากคำตอบของการหา LU-decomposition ไม่ได้มีคำตอบเดียว และในการหาทั้งสองแบบนี้ได้รวม permutes lower triangular factor แล้ว

สำหรับในกรณีที่ต้องการทราบ permuted lower triangular factor ด้วยจะใช้คำสั่ง

```
[L,U,P]= lu(A)
L =
   1.0000    0         0
   -1.0000    1.0000         0
   -0.5000         0    1.0000
U =
   -2.0000   -6.0000    1.0000
         0   -1.0000    8.0000
         0         0    2.5000
P =
     0     1     0
     0     0     1
     1     0     0
```

และ $PA = LU$

② **QR-Decomposition** เป็นอีกวิธีหนึ่งในการแยก matrix ออกเป็น orthonormal matrix Q และ upper triangular matrix, R (ซึ่งบางครั้งเรียก right hand matrix เพราะมีค่าเฉพาะด้านขวามือของ main diagonal เท่านั้น) โดย orthogonal matrix คือ matrix ที่มีคุณสมบัติ

$$QQ^T = I$$

หรือ

$$Q^{-1} = Q^T$$

สำหรับ MATLAB ใช้

[Q,R]= qr(A) คำนวณหา matrix Q และ R ซึ่งทำให้ A = QR โดย Q เป็น orthonormal และ R เป็น upper triangular matrix โดยที่ A ไม่จำเป็นจะต้องเป็น Square Matrix

เช่น

```
A=[1 2 3; 5 8 7];
[Q,R]=qr(A)
Q =
   -0.1961   -0.9806
   -0.9806    0.1961

R =
   -5.0990   -8.2369   -7.4524
         0   -0.3922   -1.5689
```

③ **Singular Value decomposition (SVD)** เป็นการ decomposed matrix ออกเป็นผลคูณของ matrix 3 matrix คือ

$$A = QSV$$

โดย Q และ V เป็น orthonormal matrix และ S เป็น diagonal matrix สำหรับ MATLAB ใช้

[Q,S,V]=svd(A) decompose A ออกเป็น A = QSV เมื่อ Q และ V เป็น orthonormal และ s เป็น diagonal matrix

สำหรับการ decompose ทั้ง 3 กรณีนั้น ถ้า A มีขนาดเป็น [m x n] จะได้

- U, L และ R มีขนาดเดียวกันกับ A
- Q มีขนาดเป็น m x m
- V มีขนาดเป็น n x n
- S มีขนาดเป็น m x n

ตัวอย่างเช่น

```
A=[1 4;8 9];
[L,U]=lu(A)
L =
  0.1250    1.0000
  1.0000         0
U =
  8.0000    9.0000
         0  2.8750
[Q,R]=qr(A)
Q =
 -0.1240   -0.9923
 -0.9923    0.1240
R =
 -8.0623   -9.4266
       0   -2.8528
[Q,S,V] = svd(A)
Q =
    0.2966    0.9550
    0.9550   -0.2966
S =
   12.5963         0
         0    1.8259
V =
    0.6301   -0.7765
    0.7765    0.6301
```

❻ Array Operation

ในหลายๆ กรณีเรามีความจำเป็นที่จะคูณหรือหารเฉพาะ element ของ matrix เท่านั้น ซึ่งนิยมเรียก element - by - element operation หรือ array operation ตัวอย่างเช่นพิจารณา

```
A  = [1 2 3 4];
B  = [2 4 4 5];
```

จะพบว่า A * B จะไม่สามารถหาค่าได้ แต่หากเราต้องการ matrix C ซึ่งมีค่าเป็น $c_i = a_i b_i$ หรือ

$$c(1) = A(1) * B(1),\ c(2) = A(2) * B(2),\ldots$$

เราจะสามารถใช้คำสั่ง array operation ได้โดยใช้ “.” แล้วตามด้วย “ * ”

```
C  =  A.*B
```

ทำนองเดียวกัน หากต้องการหาค่า D ซึ่งมี element แต่ละ clement เท่ากับผลหารของแต่ละ element ของ A หารด้วย B ในตำแหน่งเดียวกัน จะใช้ “.” แล้วตามด้วย “/” เช่น

```
D  = A./B
```

ส่วนการยกกำลัง

```
E  = A.^B
```

จะได้ว่า E จะมี element แต่ละตัวเท่ากับ element นั้นของ A ยกกำลังด้วย element ที่ตำแหน่งเดียวกันของ B ตัวอย่างเช่น

```
A=[1 2 3 4 5];
B=[4 5 9 10 2];
A.*B
ans =
   4   10   27    40    10
B.^A
ans =
     4    25   729  10000    32
```

การใช้ Array operation สำหรับ matrix จะต้องใช้ได้กับ matrix ที่มีขนาดเท่ากันเท่านั้น

❼ ลำดับการคำนวณ

ขั้นตอนการคำนวณใน MATLAB ก็เช่นเดียวกันกับพีชคณิต ทั่วไปคือจะทำตามลำดับดังนี้คือ

(1) คำนวณในวงเล็บก่อน โดยเริ่มจากวงเล็บในสุดก่อน

(2) ยกกำลัง โดยเริ่มจากซ้ายไปขวา

(2) คูณหรือหาร โดยเริ่มจากซ้ายไปขวา

(3) บวก หรือ ลบ โดยเริ่มจากซ้ายไปขวา

เพื่อให้ลำดับการคำนวณเป็นไปตามที่ต้องการควรใช้วงเล็บเข้าช่วย เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นได้

❽ การจัดเรียง Matrix ใหม่

ในหลายกรณี เรามีความจำเป็นต้องจัด matrix ให้อยู่ในรูปอื่นๆ เพื่อความเหมาะสม เพื่อใช้ในการคำนวณที่เราต้องการ ซึ่ง MATLAB มี function ต่อไปนี้

rot90(A)	จะหมุน matrix A ไป 90° ทวนเข็มนาฬิกา
rot90(A,n)	จะหมุน matrix A ไป 90° เป็นจำนวน n ครั้ง ในทิศทวนเข็มนาฬิกา
fliplr(A)	พลิก Matrix A จากซ้ายไปขวาคือเอา column ซ้ายสุดไปไว้ขวาสุดแล้วสลับที่กันต่อ ๆ ไปจนครบ
flipud(A)	พลิก Matrix A จากบนลงล่างคือเอา row บนสุดไปไว้ล่างสุดแล้วสลับที่กันต่อๆ ไปจนครบ
reshape(A,m,n)	จัด matrix A ใหม่ให้มีขนาดเป็น [m x n] โดย element ของ matrix A เดิมต้องมีจำนวนเท่ากับ mn ด้วย และ MATLAB จะพิจารณาเพิ่มหรือลดขนาดไปทีละ column ของ A

diag(x)	ถ้า x เป็น matrix จะได้ผลเป็น vector ที่มี main diagonal ของ x เป็น element ถ้า x เป็น vector จะได้ผลเป็น matrix ที่มี vector A เป็น diagonal
triu(A)	สร้าง square matrix จาก matrix A โดยตัดส่วนที่อยู่ต่ำกว่า main diagonal ของ A ให้เป็นศูนย์
triu(A,k)	สร้าง square matrix จาก matrix A โดยตัดส่วนที่ต่ำกว่า diagonal ที่ k ของ A ให้เป็นศูนย์
tril(A)	สร้าง square matrix จาก matrix A โดยตัดส่วนที่อยู่สูงกว่า main diagonal ของ A ให้เป็นศูนย์
tril(A,k)	สร้าง square matrix จาก matrix A โดยตัดส่วนที่อยู่สูงกว่า diagonal ที่ k ของ A ให้เป็นศูนย์

ตัวอย่างเช่นกำหนดให้

```
A=[1 2 3;0 2 4;3 6 8]
A =
   1   2   3
   0   2   4
   3   6   8

rot90(A)
ans =
     3     4     8
     2     2     6
     1     0     3

tril(A)
ans =
     1     0     0
     0     2     0
     3     6     8

triu(A,2)
ans =
     0     0     3
     0     0     0
     0     0     0

diag(A)
ans =
     1
     2
     8
flipud(A)
ans =
     3     6     8
     0     2     4
     1     2     3
```

2.4 Multidimensional Arrays

ตามที่กล่าวมาแล้วว่าเดิมนั้น MATLAB จะจำกัดตัวแปรอยู่ในลักษณะ 2 มิติ แต่สำหรับ MATLAB version 5 ขึ้นไปจะสามารถจำกัดตัวแปรให้มีมิติมากกว่า 2 ได้ ยกตัวอย่างเช่นเรามีตัวแปรที่เป็น Matrix 2 มิติดังนี้

```
X=[1 3 5; 4 2 8]
X =
   1   3   5
   4   2   8

y=[7 1 4;2 0 1]
y =
     7     1     4
     2     0     1

z=[4 2 1;3 6 9]
z =
     4     2     1
     3     6     9
```

จากนั้นเราต้องการที่จะรวมตัวแปรทั้ง 3 นี้ให้เป็นตัวแปรเดียวกัน นั่นคือต้องมีการสร้างมิติที่ 3 ขึ้นมาสำหรับตัวแปรตัวนี้ ตัวแปรที่มี 3 มิตินี้สามารถจิตนาการได้ว่า เหมือนกับการที่เรานำตัวแปร x, y, z มาเขียนไว้บนกระดาษแต่ละหน้าจากนั้นเย็บกระดาษทั้งสามหน้าเข้าด้วยกัน การที่จะบอกตำแหน่งของค่าใดค่าหนึ่งจะต้องบอกให้ชัดเจนว่าอยู่ที่หน้าใด row ใด และ column ใด ซึ่งจะเป็นการบอกทั้งสามมิตินั่นเอง ส่วนตัวแปรที่มีมิติมากกว่านี้ก็สามารถที่จะสร้างได้เช่นกัน สำหรับการรวบรวมหรือเรียงต่อ (con**cat**enates) ตัวแปรทั้งสามที่กล่าวมาแล้วก็สามารถทำได้โดยใช้คำสั่ง **cat** ดังนี้

```
d=cat(3,x,y,z)
d(:,:,1)=
   1   3   5
   4   2   8
d(:,:,2)=
   7   1   4
   2   0   1
d(:,:,3)=
     4     2     1
     3     6     9

size(d)
ans =
     2     3     3
```

ซึ่งจากเดิมที่ x, y, z มีมิติเป็น 2 x 3 เราจะได้ d มีขนาดเป็น 2 x 3 x 3 สำหรับรายระเอียดของการใช้ **cat** มีดังนี้

```
Cat(dims,A1,A2,A3,...)
```

เป็นการเรียง matrix A1, A2, A3,... ซึ่งเป็น matrix ซึ่งมีขนาดเท่ากัน รวบรวมเข้าไว้ด้วยกันทั้งหมดจำนวน dims matrix

สำหรับคำสั่งที่ใช้กับ Multi-dimensional Array มีดังนี้

ndim(A)	จะให้คำตอบเป็นจำนวน dimension ของ A
B = permute(A,order)	การสลับจัดเรียงตัวแปร A ใหม่ให้เป็น B ซึ่งมี order ตามที่ต้องการ โดย order เป็น vector ที่กำหนดขนาดของ B และตัวแปรที่กำหนดใหม่จะต้องมีจำนวน element เท่ากับตัวแปรเดิม
ipermute(A,order)	ย้อนกลับคำสั่ง **permute(A, order)**
shiftdim(A,n)	ข้ามสลับ dimension ของ A ไปจำนวน n
B = squeeze(A)	เป็นการย้าย dimension ที่เท่ากับ 1 ออกไป ซึ่งจะได้ B ซึ่งมี element ทุกตัวเหมือน A แต่มิติใดที่เท่ากับ 1 จะถูกกำจัดออกไป สำหรับคำสั่งนี้จะไม่มีผลกระทบต่อ matrix ดังนั้นหากใช้คำสั่งนี้กับ vector ก็จะได้ผลออกมาเป็น vector เช่นเดิม

ตัวอย่างของคำสั่งเหล่านี้เช่น

```
x=[1 3 5;4 2 8];
y=[7 1 4;2 0 1];
z=[4 2 1;3 6 9];
d=cat(3,x,y,z);
e = permute(d,[3 1 2])
e(:,:,1) =
     1     4
     7     2
     4     3
e(:,:,2) =
     3     2
     1     0
     2     6
e(:,:,3) =
     5     8
     4     1
     1     9
    f = ipermute(e,[3 1 2])
f(:,:,1) =
     1     3     5
     4     2     8
f(:,:,2) =
     7     1     4
     2     0     1
f(:,:,3) =
     4     2     1
     3     6     9
g=shiftdim(d,2)

g(:,:,1) =
     1     4
     7     2
```

```
     4     3
g(:,:,2) =
     3     2
     1     0
     2     6
g(:,:,3) =
     5     8
     4     1
     1     9
p=rand(2,1,4,3);
size(p)
ans =
     2     1     4     3
p=squeeze(p);
size(p)
ans =
     2     4     3
```

สำหรับการที่จะกำหนดเลือกค่าตัวแปรที่ตำแหน่งใดก็สามารถจะกำหนดได้โดยบอกตำแหน่งที่เราต้องการ เช่นต้องการทราบค่าที่ตำแหน่ง 1-1-3 ของตัวแปร g ตามตัวอย่างข้างบนจะใช้

$g_{(1,1,3)}$

```
ans =
     5
```

2.5 Cell Arrays

สำหรับโครงสร้างตัวแปรใหม่อีกแบบหนึ่งของ MATLAB 5 ก็คือการที่มี cell array ซึ่งใน version ก่อนหน้านี้ แต่ละ element ของ matrix ต้องเป็นตัวเลขหรือตัวหนังสือเท่านั้น สำหรับความหมายของ cell array ก็คล้ายกับว่าเป็นการยอมให้ในแต่ละ element ของ matrix เป็น matrix ได้นั่นเอง

❶ การกำหนด Cell Arrays

การกำหนด cell arrays จะใช้วงเล็บปีกกา { } ซึ่งสามารถทำได้ 2 วิธีคือ

- กำหนดแบบ cell index

```
x(1,1)={[1 2;3 4]};
x(1,2)={5};
x(2,1)={linspace(0,10,10)};
x(2,2)={'Text Only'};
```

- กำหนดแบบ content addressing

```
x{1,1}=[1 2;3 4];
x{1,2}=5;
x{2,1}=linspace(0,10,10);
x{2,2}='Text Only';
```

ซึ่งทั้งสองแบบนี้จะให้ผลเหมือนกันและสามารถที่จะใช้แทนกันได้ จากตัวอย่างข้างบนเราได้สร้าง matrix x ซึ่งมีขนาด 2 x 2 โดยแต่ละ element จะมี cell ย่อยลงไปอีก เช่น x(1,1) จะเป็น matrix ขนาด 2 x 2 ซ้อนอยู่เป็นต้น หากเราต้องการทราบค่าและขนาดของ x ก็สามารถใช้คำสั่ง

```
x
x =
    [2x2  double]        [   5]
    [1x10 double]             'Text Only'

size(x)
ans =
     2     2
```

และชนิดของ x สามารถทราบได้จาก

```
class(x)
ans =
      cell
```

นั่นคือ x เป็น cell arrays หากต้องการทราบค่าทั้งหมดที่บรรจุใน x ก็สามารถใช้คำสั่ง **celldisp** ได้ดังนี้

```
celldisp(x)
x{1,1} =
        1     2
        3     4
x{2,1} =
Columns 1 through 7
0   1.1111    2.2222   3.3333   4.4444   5.5556   6.6667
Columns 8 through 10
7.7778    8.8889   10.0000
x{1,2} =
        5
x{2,2} =
        Text Only
```

หรือหากต้องการเจาะจง cell ก็สามารถกำหนดได้โดยใช้วงเล็บปีกกาเช่น

```
x{1,2}
ans =
     5
```

สำหรับการสร้าง cell array ตามแบบที่เคยสร้างใน matrix แบบเดิมก็ยังคงสามารถกระทำได้ โดยใช้เครื่องหมายปีกกาแทนวงเล็บใหญ่ส่วนการแบ่ง column ยังคงใช้ ; เหมือนเดิม เช่น

```
y={[1 3; 5 6], [2 3; 5 7]; ['TOM';'TON'],[2+i  5+4i]}
y =
  [2x2 double] [2x2 double]
  [2x3 char  ] [1x2 double]
```

ในการสร้าง cell array ที่ว่างเปล่าสามารถทำได้โดยใช้คำสั่ง cell เช่น

```
Z=cell(2,2)
z =
  [] []
  [] []
```

❷ การรวม เปลี่ยนรูปและกำหนดค่า Cell Array

การรวมหรือเปลี่ยนรูป cell array ก็สามารถกระทำได้เหมือนการรวม matrix ปกติที่เราทำผ่านมา เช่นจากตัวแปรที่กำหนดในตัวอย่างที่ผ่านมาคือ

```
x
x =
  [2x2  double]        [   5]
     [1x10 double]             'Text Only'

y
y =
    [2x2 double]   [2x2 double]
    [2x3 char  ]   [1x2 double]
```

หากเราต้องการรวมหรือเปลี่ยนรูปก็สามารถทำได้ดังนี้

```
a=[x y]
a =
[2x2  double]        [   5]   [2x2 double]    [2x2 double]
[1x10 double]    'Text Only'   [2x3 char  ]    [1x2 double]

b=[x;y]
b =
    [2x2  double]    [         5]
    [1x10 double]    'Text Only'
    [2x2  double]    [2x2 double]
    [2x3  char  ]    [1x2 double]
```

หากต้องการตัด matrix บางส่วนก็สามารถใช้ colon operator ได้เช่น

```
c=b(2:3,1)
c =
       [1x10 double]
       [2x2  double]
```

และหากต้องการเปลี่ยนรูปก็สามารถใช้ คำสั่ง reshape ได้ เช่น

```
c=reshape(b,2,4)
c =
[2x2  double] [2x2  double]   [ 5]  [2x2 double]
[1x10 double] [2x3  char  ]    'Text Only'    [1x2 double]
```

สำหรับการกำหนดค่า หรือเข้าเปลี่ยนแปลงค่าใน cell array เราก็สามารถกระทำได้ โดย

- ถ้า cell นั้นเป็นตัวแปรตัวเดียวหรือต้องการทราบค่าทั้งหมดใน cell นั้นก็สามารถกำหนดค่าโดยใช้เครื่องหมายปีกกาบอกตำแหน่งได้
- ถ้า cell นั้นเป็น matrix ซ้อนอยู่ให้ใช้เครื่องหมายปีกกากำหนดตำแหน่ง cell แล้วต่อด้วยวงเล็บกำหนดตำแหน่งใน matrix นั้นอีกครั้งหนึ่ง
- ถ้าหากว่าต้องการทราบหรือกำหนดค่าเป็นช่วงสามารถทำได้โดยใช้คำสั่ง **deal**

เช่นจากตัวอย่างข้างบนถ้า

```
c
c =
[2x2 double] [2x2 double]    [ 5]  [2x2 double]
[1x10 double]   [2x3 char ]    'Text Only'    [1x2 double]

c{1,4}
ans =
     2     3
     5     7

c{1,4}(2,1)
ans =
     5

[p q] = deal(c{1,2:3})
p =
     1     3
     5     6
q =
     5
```

❸ Cell Array และ Character String

สำหรับ character string เดิมนั้นหากจะนำมารวมอยู่ใน matrix แล้วข้อที่ควรระวังก็คือ ทุกแถวจะต้องมีจำนวน column เท่ากันซึ่งในทางปฏิบัติที่เกี่ยวข้องกับตัวหนังสือแล้วการทำเช่นนั้นค่อนข้างที่จะยุ่งยาก แต่ในกรณีของ MATLAB 5.x การใช้ cell arrays จะช่วยให้เราสามารถบรรจุ character string ที่มีความยาวต่างกันลงไปอยู่ใน column เดียวกันของ cell ได้ ตัวอย่างเช่น

```
x={'Tom';'Pookie';'Tata';'Nut';'Christina';'Mai'}
x =
    'Tom'
    'Pookie'
    'Tata'
    'Nut'
    'Christina'
    'Mai'

class(x)
ans =
cell

size(x)
ans =
     6     1
```

ซึ่งจะเห็นว่าตัวแปร x เป็น cell array ที่มี 6 row x 1 column และในแต่ละ row ไม่จำเป็นต้องมีความยาวเท่ากัน ซึ่งการปรับปรุงเช่นนี้จะทำให้มีความสะดวกในการกรอกข้อมูลมากขึ้นกว่าเดิมมาก

2.6 Structures

สำหรับโครงสร้างตัวแปรแบบที่ 3 ที่บรรจุอยู่ใน MATLAB 5.x คือโครงสร้างตัวแปรที่เรียกว่า structure ซึ่งทาง MATLAB จะกำหนดให้ structure นั้นเป็น object ในรูปแบบที่เป็นที่เก็บข้อมูล (Data Containers) หรือที่เรียกว่า fields สำหรับผู้ที่เคยผ่านงานการใช้โปรแกรมทางด้านฐานข้อมูล (database) คงคุ้นเคยกับศัพท์เหล่านี้เป็นอย่างดี ซึ่ง structure นั้นก็จะมีประโยชน์อย่างหนึ่งคือใช้ในการทำฐานข้อมูลเช่นกัน

❶ สร้างการ Structure

การสร้างโครงสร้างฐานข้อมูลใน MATLAB 5.x ตามความคิดของผู้เรียบเรียงแล้วคิดว่าค่อนข้างจะง่ายกว่าการสร้างฐานข้อมูลบนโปรแกรมฐานข้อมูลทั่วไป ลำดับการสร้างฐานข้อมูลมีดังนี้

- ลำดับแรกเราต้องกำหนดว่าเราจะใช้ฐานข้อมูลชื่ออะไร
- ในฐานข้อมูลนั้นจะต้องการเก็บรายระเอียดอะไรบ้าง ซึ่งแต่ละรายละเอียดก็คือ field นั่นเอง
- กำหนดชื่อ field ทั้งหมด
- เราสามารถเริ่มใส่ข้อมูลได้เลย

สำหรับตัวอย่างสมมุติว่าเราต้องการจะรวมข้อมูลของชิ้นส่วนเครื่องจักรเครื่องหนึ่งไว้ในฐานข้อมูลชื่อ part โดยรายละเอียดที่ต้องการรวบรวมคือ ชื่อ(name) ผู้ผลิตชิ้นส่วน (make) ราคาที่ซื้อ (price) และหมายเหตุ (remark) ถ้ามี ดังนั้นเมื่อรวมแล้วเราจะมี filed ทั้งหมด 4 field สมมุติว่าเราจะตั้งชื่อ field แต่ละ field ตามชื่อภาษาอังกฤษ จากนั้นเราก็เริ่มใส่ข้อมูลได้ ดังนี้

```
Parts.name ='Left motor';
Parts.make='GE';
Parts.price=3990;
Parts.remark='Re-build';
```

เราสามารถจะดูข้อมูลที่เราบรรจุไว้ได้เช่น

```
parts
parts =
      name: 'Left motor'
      make: 'GE'
     price: 3990
    remark: 'Re-build'
```

จากนั้นหากเราต้องการจะใส่ข้อมูลชุดที่ 2 และต่อๆ ไปเราสามารถทำได้ดังนี้

```
parts(2).name='Over head cam';
parts(2).make='ACME';
parts(2).price=420;
parts(2).remark='New';
```

จากนั้นหากต้องการจะดูค่าที่บันทึกไว้ ลองใช้คำสั่งปกติ

```
parts
```

```
parts =
1X2 struct array with fields:
    name
    make
    price
    remark
```

จะพบว่าเมื่อมีการบันทึกค่ามากกว่า 1 ชุดข้อมูล MATLAB จะแสดงเฉพาะโครงสร้างของตัวแปรให้เห็น หากต้องการจะเข้าถึงข้อมูลในกรณีเช่นนี้เราจะต้องมีการกำหนดชื่อ field ด้วยเช่น

```
parts.name
ans =
Left motor
ans =
Over head cam
```

หรืออาจเจาะจงลงไปก็ได้

```
Parts(2).price
ans =
     420
```

นอกจากนั้นถ้า field มีค่าเป็นตัวเลข เราก็สามารถที่จะนำไปใช้ในการคำนวณได้ด้วยเช่น

```
total = parts(1).price*35.4
total =
       141246
```

ซึ่งในการคำนวณอื่นๆ ก็สามารถทำได้เช่นกัน

❷ ลักษณะของ field

จากตัวอย่างที่ยกให้เห็นนั้นเป็นการยกตัวอย่างง่ายๆ เพื่อให้ผู้ที่ไม่เคยใช้ program ทางด้านฐานข้อมูลเข้าใจได้ สำหรับรายละเอียดที่มากกว่านั้นซึ่งเป็นคุณลักษณะของ MATLAB 5.x สามารถรวบรวมได้ดังนี้

- ค่าที่บรรทึกในแต่ละ field จะเป็น object ประเภท ตัวเลข, matrix, cell array หรือแม้แต่ Structure ก็ได้
- หากว่าต้องการจะดูชื่อ field ทั้งหมดเราสามารถทำได้โดยใช้คำสั่ง **fieldname** เช่น

```
fieldnames(parts)
ans =
    'make'
    'name'
    'price'
    'remark'
```

- การสร้าง structure อาจสร้างได้โดยใช้คำสั่ง **struct** เช่น

```
parts=struct('name',N,'make',M,'price',P)
parts =
1x2 struct array with fields:
    name
    make
```

```
price
```

ส่วนการเพิ่มข้อมูลก็อาจจะทำแบบที่ผ่านมาหรือจะใช้สร้าง matrix N, M ,P ใหม่ก็ได้

- ในแต่ละ field ที่กำหนดสามารถที่จะสร้างเป็น sub-field ต่อลงไปได้อีกเช่น

```
parts.make.FirstChoice='GE';
parts.make.SecondChoice='PW';
```

ซึ่งหากมีการนับ file แล้วจะมี field ชื่อ make เพียง field เดียว แต่จะมีลักษณะเป็น structure ซ้อนอยู่

```
parts
parts =
    make:[1x1 struct]
```

สำหรับการเข้าสู่ค่าในโครงสร้างอาจใช้คำสั่ง getfield ซึ่งมีรูปแบบเป็น

F=getfield(S,'field')

จะให้ค่า F เป็นค่าที่บรรจุอยู่ใน field ซึ่งคำสั่งนี้จะเทียบเท่ากับ F = S และ S ต้องเป็น 1-by-1 structure.

F=getfield(S,{i,j},'field',{k})

จะให้ค่าเทียบเท่ากับ F = S(i,j).field(k).

ซึ่งจากโครงสร้างของตัวแปรใหม่ใน MATLAB 5.x จะช่วยให้เราสามารถใช้ MATLAB ในงานต่างๆ ได้เพิ่มขึ้นอีกมากมาย รายละเอียดของการใช้ตัวแปรแบบต่างๆ นั้นสามารถศึกษาเพิ่มเติมได้จากคู่มือการใช้งาน

2.7 Output Options

มีวิธีการหลายวิธีที่จะแสดงค่าของตัวแปร หรือค่าคงที่ที่กำหนดลงไปในหน่วยความจำของ MATLAB วิธีที่ง่ายที่สุดที่พิมพ์ชื่อตัวแปรที่ Command Window แล้วกด Enter Key ซึ่ง MATLAB จะแสดงค่าตัวแปรนั้นออกมา ส่วนการแสดงผลนั้นมีวิธีกำหนดได้หลายวิธี ดังนี้

❶ Display Format

รูปแบบที่จะแสดงผลตัวเลขแบบต่างๆ สามารถกำหนดให้การแสดงค่าเป็นไปตามที่ต้องการได้ดังต่อไปนี้

คำสั่ง MATLAB	การแสดงผล	ตัวอย่าง
`format short`	แสดงทศนิยม 4 ตำแหน่ง	*22.1234*

format long	แสดงทศนิยม 15 ตำแหน่ง	*22.123456789012345*
format bank	แสดงทศนิยม 2 ตำแหน่ง	*22.12*
format short e	แสดงทศนิยม 4 ตำแหน่งพร้อมเลขยกกำลังของ 10	*2.123 e+01*
format long e	แสดงทศนิยม 15 ตำแหน่งพร้อมเลขยกกำลังของ 10	*2.12345678901234 e+01*
format +	แสดงค่า + , - หรือว่าง	+

สำหรับค่าที่ตั้งเบื้องต้นคือ format short ซึ่งหากไม่กำหนดค่าเป็นอย่างอื่น MATLAB จะแสดงค่านี้ไปตลอดการใช้งาน ตัวอย่างเช่นการแสดงค่า π สามารถแสดงได้หลายรูปแบบเช่น

```
format short
pi
ans =
  3.1416

format long
pi
ans =
  3.14159265358979

format short e
pi
ans =
  3.1416e+000

format long e
pi
ans =
  3.141592653589793e+000
```

❷ การแสดงผลตัวอักษรและค่าตัวแปร

นอกจาก MATLAB จะมี echo ซึ่งเป็นการแสดงค่าตัวแปรแล้ว ยังสามารถที่จะกำหนดรูปแบบให้ MATLAB แสดงผลตามที่ต้องการได้อีกด้วย ดังมีการใช้คำสั่งต่อไปนี้

- **การแสดงผลอาจใช้คำสั่ง disp**

เช่นถ้า A เป็นตัวแปร ซึ่งได้รับการกำหนดค่า **disp(A)** จะแสดงค่า A ส่วน **disp('Display')** MATLAB จะพิมพ์คำว่า Display ลงบนจอภาพ สำหรับตัวอักษร (string) นั้นจะต้องบรรจุอยู่ในเครื่องหมาย "' '"

ตัวอย่างเช่น

```
A=10;
disp(A) ; disp('newtons')
      10
```

```
newtons
```

➢ **การแสดงผลอย่างมีรูปแบบ**

หากต้องการแสดงผลอย่างมีรูปแบบ ควรจะใช้คำสั่ง **fprintf** เพราะจะสามารถควบคุม output ได้ดีกว่าคำสั่ง **disp** สำหรับคำสั่ง **fprintf** นั้นสามารถกำหนดรูปแบบของ output variable ได้โดยใช้ตัวกำหนด (specifier) ซึ่งประกอบด้วย %e, %f และ %g

%e จะเป็นการแสดงผลในลักษณะ exponential notation เช่น 2.3456 + 2

%f จะแสดงเป็นเลขทศนิยม

%g จะเลือกแสดง **%e** หรือ **%f** แล้วแต่ว่าแบบใดจะสั้นกว่า

ตัวอย่าง ถ้าใช้ specifier **%6.2f** จะเป็นการแสดงค่า 4 ตำแหน่ง ประกอบด้วยเลขหลังจุดทศนิยม 2 ตำแหน่ง 1 ตำแหน่งเป็นจุดทศนิยม ที่เหลือจะเป็นตัวเลขหน้าจุดทศนิยม

ตัวอย่าง การใช้ คำสั่ง **fprintf**

```
A=10;
fprintf ('Total force is %f  newtons \n',A)
Total force is 10.000000  newtons
```

คำสั่ง **\n** เป็นคำสั่งให้เริ่มพิมพ์ค่าต่อไปในบรรทัดใหม่ เช่น

```
fprintf ('Total force is \n  %f newtons '\n',A)
Total force is
 10.000000  newtons

fprintf ('Total force is %6.2f newtons \n',A)
Total force is 10.00 newtons
```

2.8 Simple Plot

สำหรับการแสดงผลที่สำคัญแบบหนึ่งของทางวิศวกรรมคือ การแสดงผลด้วยกราฟ ซึ่ง MATLAB เป็นโปรแกรมที่สามารถสร้างกราฟได้ดีมากโปรแกรมหนึ่ง โดยสามารถเขียนกราฟได้ทั้ง 2 มิติ และ 3 มิติ โดยเฉพาะกราฟ 3 มิตินั้นมีให้เลือกหลายแบบ สำหรับหัวข้อนี้จะเป็นการเสนอเฉพาะกราฟ x-y แบบง่ายๆ ที่ใช้ในการเขียนกราฟขั้นพื้นฐานทั่วไป ส่วนรายละเอียดในการเขียนกราฟในลักษณะอื่นจะกล่าวในบทต่อไป

สำหรับ คำสั่ง MATLAB ที่ใช้สั่งการ plot นั้นมีหลายคำสั่งโดยการกำหนดรายละเอียดของรูปแบบคำสั่งคล้าย ๆ กัน สำหรับ คำสั่ง **plot** แบบทั่ว ๆ ไปมีรูปแบบง่าย ๆ ดังนี้

plot(x,y) สร้าง Linear plot ของค่า x และ y โดยแกน x เป็นแกนนอน และ y เป็นแกนตั้ง

semilogx(x,y) สร้าง plot ของ x และ y โดยแกน x เป็นสเกล log และ y เป็นเชิงเส้น

semilogy(x,y) สร้าง plot ของ x และ y โดยแกน x เป็นเชิงเส้น และ y เป็น log

loglog(x,y) สร้าง plot บน log scale ของค่า x และ y

plotyy(x_1,y_1,x_2,y_2)

สร้าง Linear plot ของค่า x1-y1 และ x2-y2 แต่จะมีการวางแกน y ไว้สองข้างของกราฟ โดยวางแกน y1 ไว้ซ้ายมือ และวางแกน y2 ไว้ขวามือและใช้แกน x เป็นแกนนอนแกนเดียว

โดยทั่วไปเมื่อเราสั่ง plot แล้ว MATLAB จะสร้าง graphic window ขึ้นมาโดยอัตโนมัติและจะทำการ plot ค่าที่สั่งไป และทำการ plot ซ้ำบน graphic window เดิม หากมีการสั่งคำสั่ง plot ใหม่ ยกเว้นจะมีคำสั่งเป็นอย่างอื่น เช่น พิมพ์คำสั่ง **`figure`** ที่ Command Window หรือไปที่ File menu เลือก New Figure สำหรับชื่อของ graphic windows ถ้าไม่กำหนดเป็นอย่างอื่นจะเรียงเป็น figure 1, figure 2, .. ไปเรื่อย ๆ ตามลำดับ

❶ รูปแบบและรายละเอียดคำสั่ง

สำหรับรายละเอียดของการ plot มีดังนี้ (ในที่นี้ใช้คำสั่ง plot เป็นการยกตัวอย่าง ส่วนคำสั่งอื่น เช่น **`semilogx`**,... ก็จะมีรูปแบบเหมือนกัน)

`plot(x,y)` Plot vector x กับ vector y โดยมี coordinate (x_j , y_j)

`plot(y)` Plot coordinate (j, y_j) ของ vector y

`plot(z)` ในกรณี complex number z = x + iy จะเป็นการ plot ด้วย coordinate (Re(z), Im(z))

`plot(A)` ในกรณีที่ A เป็น matrix จะ plot coordinate (j,A_{1j}), (j,A_{2j}),.. เมื่อ j เป็น index ในแต่ละ row กล่าวอีกอย่างหนึ่งก็คือ จะ plot A ที่ละ column เทียบต่อ index ของมัน

`plot(x,A)` Plot matrix A ขนาด m x n เทียบกับ vector x ถ้า x เป็น vector ความยาว m จะเป็นการ plot แต่ละ column ของ A เทียบกับ x แต่ถ้า vector x มีความยาว n จะเป็นการ plot แต่ละ row ของ A เทียบกับ x

`plot(A,x)` plot vector x เทียบกับ Matrix A ถ้า A มีขนาด m x n และ vector x มีความยาว m จะ plot x เทียบกับแต่ละ column ของ A แต่ถ้า vector x มีความยาว n จะ plot x เทียบกับแต่ละ row ของ A

`plot(A,B)` plot column ของ B เทียบกับ column ของ A ดังนั้น ถ้าทั้ง A และ B มีขนาด m x n จะเป็นการ plot curve n เส้นแต่ละเส้นจะมี m coordinate

`plot(x,y,'str')`

เมื่อ 'str' คือ character string ซึ่งจะเป็นการกำหนดลักษณะของสีของเส้นกราฟ โดยมีการกำหนดดังนี้

สี	ลักษณะจุด	ลักษณะเส้น
y = yellow	. จุด	- เส้นทึบ ————
g = green	* เครื่องหมายดอกจันทร์	-- เส้นปะ .....
m = magenta	X เครื่องหมายกากบาท	-. เส้นปะ dash-dotted -.-.-.-.-

b = blue	O เครื่องหมายวงกลม	: เส้น dotted .....
c = cyan	+ เครื่องหมาย บวก	
w = white	S สี่เหลี่ยมจัตุรัส	
r = read	D รูปข้าวหลามตัด	
k = black	^ สามเหลี่ยมชี้ขึ้น	
	V สามเหลี่ยมชี้ลง	
	> สามเหลี่ยมชี้ข้างขวา	
	< สามเหลี่ยมชี้ข้างซ้าย	
	P ดาวห้าแฉก	
	H ดาวหกแฉก	

ถ้าไม่มีการเลือกตัวเลือก MATLAB จะใช้ " - " (เส้นทึบ) สำหรับสีถ้าหากไม่มีการเลือก MATLAB จะเรียงลำดับให้เป็น y, g, m, ... ตามลำดับของการ plot ในกราฟนั้น ตัวอย่าง เช่น ถ้า string เป็น 'r+' จะเป็นการ plot เป็นจุดใช้เครื่องหมาย + และมีสีแดง

plot(x,y,'str1',w,v,'str2',...)

จะเป็นการ plot curve มากกว่า 1 เส้นลงในระบบแกนเดียวกันโดยจะเป็นการ plot(x_i, y_i) โดยใช้คุณสมบัติ 'str1' แล้ว plot (w_i,v_i) โดยใช้คุณสมบัติ 'str2' ถ้าไม่มีการกำหนด str MATLAB จะเปลี่ยนสีให้เองโดยมีลำดับเป็น y, g, m ... และใช้เส้นทึบทุกเส้น

a =plot(x,y,'str$_1$',w,v,'str$_2$',...)

จะเป็นการ plot curve เหมือนในคำสั่งที่ผ่านมาเพียงแต่ MATLAB จะให้ค่า a ซึ่งจะเป็นค่าที่ใช้กำหนดเส้นกราฟที่สั่งด้วยคำสั่ง plot ในครั้งนี้ (MATLAB จะใช้ศัพท์ว่าเป็น handle ของกราฟ) ซึ่งกล่าวคร่าวๆว่าคล้ายกับเป็นชื่อที่ MATLAB ใช้เรียกกราฟนี้ เพียงแต่มีค่าเป็นตัวเลข

❷ การเขียนตัวหนังสือใน *Graphics Window*

สำหรับการเขียนตัวหนังสือลงบน Graphics window นั้นสามารถทำได้โดยตัวหนังสือที่จะเขียนลงไปนั้นจะต้องเป็น character string ซึ่งในคำสั่งต่อไปนี้เราสมมุติว่า **txt ='charactor string ที่ต้องการเขียน'**

title(txt) เขียน string txt เป็นชื่อของกราฟโดยเขียนที่ด้านบนของกราฟตรงกลาง

xlabel(txt) เขียน string txt เป็นชื่อแกน x คือเขียนด้านล่าง ตรงกลางของแกน x

ylabel(txt) เขียน string txt เป็นชื่อแกน y คือเขียนด้านข้างตรงกลางของแกน y และหมุนตัวหนังสือ ไปตามแกนด้วย

zlabel(txt) (สำหรับกรณี 3 มิติ) จะเป็นชื่อแกน z

text(x,y,txt) เขียน string txt ลงภายในส่วนที่มีการ plot โดยใช้ coordinate (x,y) ของกราฟที่กำลังแสดงอยู่ในกรณีที่ Text เป็น vector string ที่มีขนาดเท่ากับ vector x และ y จะเป็นการเขียน txt_i ที่ตำแหน่ง (x_i, y_i)

text(x,y,txt,'sc')

เขียน string txt ลงในส่วนที่มีการ plot โดยใช้ coordinate (0,0) ที่มุมซ้ายล่าง และ (1,1) ที่มุมขวาเป็นของกราฟที่กำลัง plot อยู่

legend(txt1,txt2,…)

เขียน legend ของ curve ที่กำลัง plot อยู่ โดย txt1 จะเป็นชื่อของcurve ที่ plot เส้นแรก, txt2 จะเป็นชื่อของ curve ที่ plot เส้นที่สองและต่อไปเรื่อยๆ

Legend(txt1,txt2,…,pos)

เขียน legend ของ curve ที่กำลัง plot อยู่ โดย txt1 จะเป็นชื่อของcurve ที่ plot เส้นแรก, txt2 จะเป็นชื่อของ curve ที่ plot เส้นที่สองและต่อไปเรื่อยๆ สำหรับ pos คือการกำหนดตำแหน่งโดยค่าของ pos จะกำหนดตำแหน่งดังนี้

0 = อัตโนมัติ จะรบกวนข้อมูลน้อยที่สุด

1 = มุมบนขวา (ค่าเบื้องต้น MATLAB จะใช้ค่านี้ถ้าไม่มีการกำหนด)

2 = มุมบนซ้าย

3 = มุมซ้ายล่าง

4 = มุมล่างขวา

-1 = ด้านขวาของกราฟ

legend off ลบ legend ออกจาก plot

legend (M,'txt1','txt2',...)

เมื่อ M คือค่า handle ของกราฟที่ต้องการกำหนด legend โดยค่าของ M ได้จากคำสั่ง **M = plot(x,y,...)** ใช้ **help legend** สำหรับข้อมูลเพิ่มเติม

❸ แกนและมาตราส่วนของแกน

สำหรับแกนของการ plot นั้นโดยทั่วไปแล้ว MATLAB จะกำหนดขนาดของแกน x, y ให้โดยอัตโนมัติเพื่อให้เหมาะสมกับค่าที่กำลัง plot อยู่ อย่างไรก็ตาม ผู้ใช้สามารถเลือกกำหนดค่าบนแกนแต่ละแกนได้ โดยใช้คำสั่ง **axis** ต่อไปนี้

axis([**Xmin, Xmax, Ymin, Ymax, Zmin, Zmax**])
: เป็นการกำหนดต่ำสุดและสูงสุดของแต่ละแกน (กำหนดค่า z สำหรับในกรณี 3 มิติ)

axis(**axis**) Lock scaling ป้องกันไม่ให้มีการเปลี่ยนแปลงค่าบนแกนในกรณีที่มีการ plot ต่อกันหลายๆ ค่า หรือ plot ค่าใหม่ลงในไปกราฟที่มีค่าของแกนเท่าเดิม ใช้ร่วมกับคำสั่ง hold

axis(**'string'**)
: เป็นการกำหนดค่า โดยให้ผลต่างกันตามแต่ 'string' ที่ให้โดย ถ้า 'string' เท่ากับ

'auto'	Reset ค่า scaling ให้ MATLAB กลับไปใช้ auto mode
'ij'	กำหนดให้แกน y นับค่าขึ้นเป็นลบ ลงเป็นบวก
'xy'	กำหนดให้แกน y นับค่าขึ้นเป็นบวก ลงเป็นลบ
'equal'	ให้ แกน x และ y มี scale เดียวกัน
'square'	ปรับค่าให้ระบบแกนมีลักษณะเป็นสี่เหลี่ยมจตุรัส
'normal'	ปรับค่าของระบบแกนให้กลับสู่มาตราฐาน
'off'	ให้ MATLAB ไม่ต้องแสดงแกนบนกราฟที่กำลัง plot
'on'	ให้ MATLAB แสดงแกนบนกราฟที่กำลัง plot

สำหรับคำสั่งเหล่านี้ อาจเขียนในลักษณะ **axis auto, axis on,...** โดยไม่ต้องมีวงเล็บก็ได้ นอกจากนี้ยังสามารถใช้คำสั่งให้แสดงลายเส้นบนเส้นกราฟด้วยคำสั่ง **grid** ดังต่อไปนี้

grid on	ให้แสดงลายเส้นบนกราฟ
grid off	ไม่ต้องแสดงลายเส้นบนกราฟ
grid	สลับจาก grid on เป็น grid off หรือ จาก grid off เป็น grid on

❹ การควบคุม Graphic

เราสามารถควบคุม graphic window ได้ดังต่อไปนี้

figure	สร้าง graphic window ใหม่ ซึ่งปกติ MATLAB จะสร้างเองอยู่แล้วหากมีการสั่ง plot
clf	ลบ graphic window
clc	ลบ command window

hold on ให้ยึดถือ graphics window ที่กำลัง plot อยู่เป็นหลักและการ plot ค่าต่อมาให้ plot ซ้ำในกราฟรูปเดิมนี้ต่อไปเลย ไม่ต้องมีการสร้างใหม่หรือลบกราฟเดิม

hold off ยกเลิก hold on

hold กลับค่าระหว่าง hold on และ hold off ไปมา

ishold เป็นคำสั่งที่ถามว่าในขณะนี้ MATLAB ได้ hold plot อยู่หรือไม่ ถ้า hold จะได้ 1 ถ้าไม่จะได้คำตอบ 0

❺ Subplot

สำหรับ graphic window โดยทั่วไปแล้วจะบรรจุกราฟอยู่เพียงรูปเดียว อย่างไรก็ตามเราสามารถกำหนดให้ MATLAB สร้าง กราฟมากกว่า 1 รูป ใน Graphic window เดียวได้ โดยกราฟย่อย ๆ แต่ละกราฟจะเรียกว่าเป็น sub - plot ของ graphic window สร้างขึ้นโดยใช้คำสั่ง **subplot** โดยมีรูปแบบดังนี้

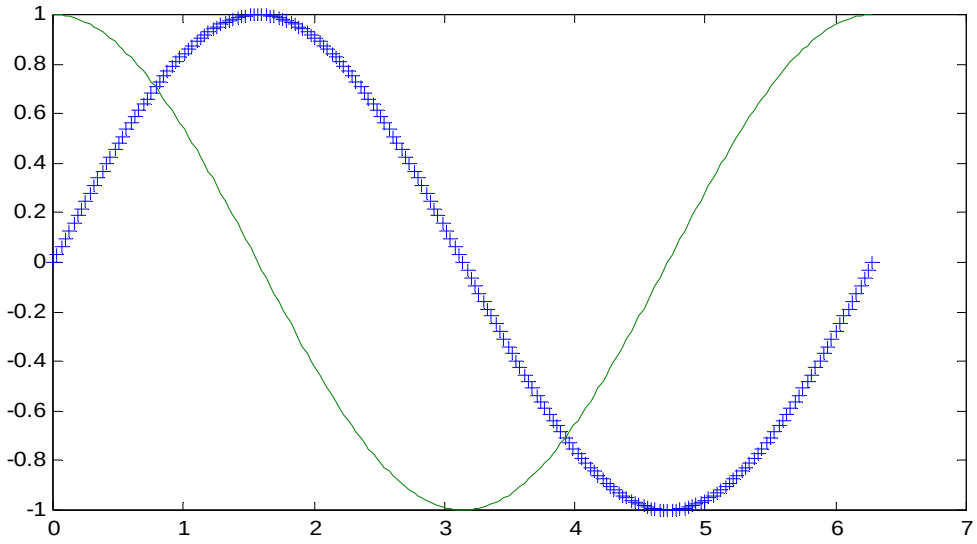
subplot(m,n,p)

สร้าง sub-plot บน graphic window โดยแบ่งออกเป็นกราฟย่อย วางอยู่ในรูป matrix ขนาด m x n และคำสั่งที่ตามมาจะให้ plot ลงบน sub-plot หมายเลข p เมื่อ p เริ่มจาก 1 โดยจาก sub-plot บนสุดและซ้ายสุด และเพิ่มขึ้นจากซ้ายมาขวา และบนลงล่าง

subplot คืนค่า subplot ให้มีเพียง window เดียว เหมือนกับคำสั่ง subplot (1,1,1)

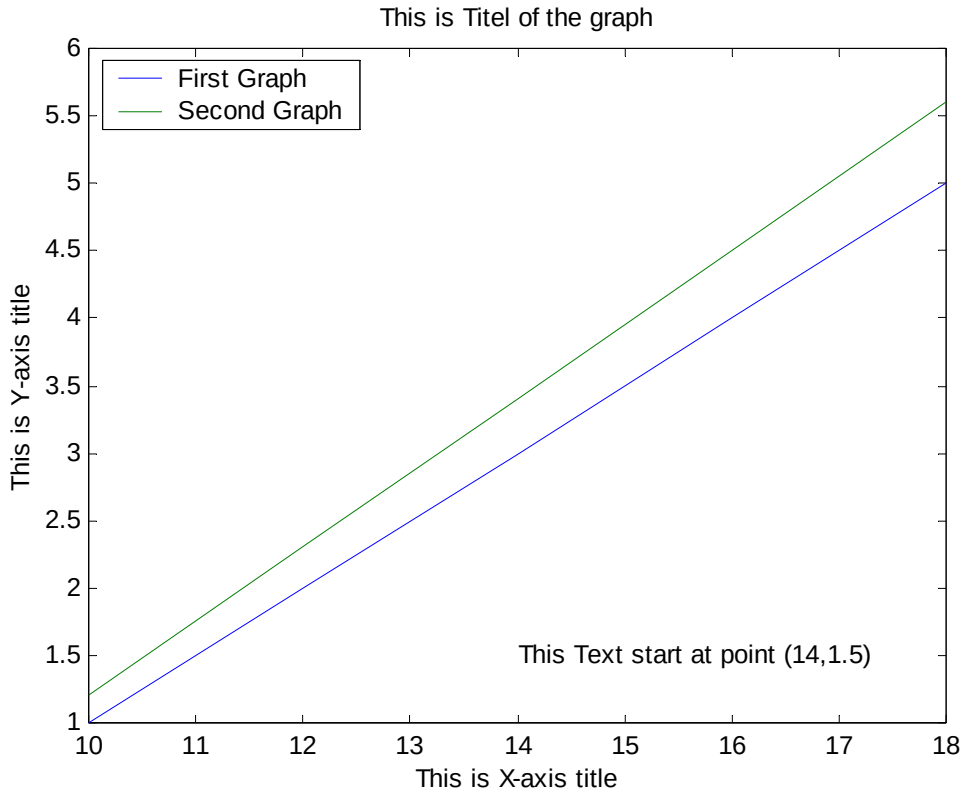
ตัวอย่างของการเขียนกราฟพื้นฐานสามารถดูได้จากตัวอย่างต่อไปนี้

```
x=0:pi/100:2*pi;
y=sin(x);
z=cos(x);
plot(x,y,'+',x,z,'-')
```



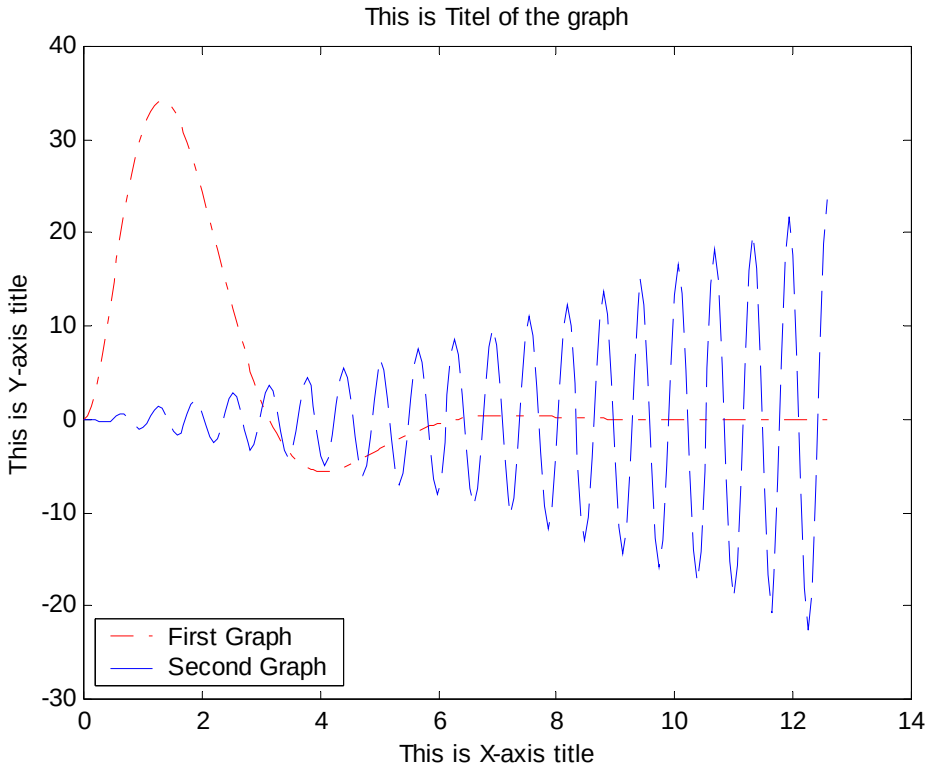
สำหรับตัวอย่างต่อไปนี้ เป็นการแสดงการเขียนกราฟพร้อมด้วยตัวหนังสือสำหรับแกนและหัวเรื่องต่างๆ

```
x=[10 12 14 16 18];
y=[1 2 3 4 5;1.2 2.3 3.4 4.5 5.6];
plot(x,y)
xlabel('This is X-axis title')
ylabel('This is Y-axis title')
title('This is Titel of the graph')
text(14,1.5,'This Text start at point (14,1.5)')
legend('First Graph','Second Graph',0)
```



ตัวอย่างต่อไปนี้เป็นการแสดงการใช้ตัวอย่าง legend ของ MATLAB และการกำหนด handel ของรูปภาพ

```
x=0:pi/50:4*pi;
y=x.*sin(x).*exp(-x);
z=x.*cos(x*10).*exp(x/20);
y=100*x.*sin(x).*exp(-x);
m=plot(x,y,'r-.',x,z,'b--');
xlabel('This is X-axis title')
ylabel('This is Y-axis title')
title('This is Titel of the graph')
legend(m,'First Graph','Second Graph',3)
```



ตัวอย่างต่อไปนี้เป็นการแสดงการ plot แบบต่างๆ โดยใช้คำสั่ง subplot

```
x=[10 12 14 16 18];
y=[1 2 3 4 5; 1.2 2.3 3.4 4.5 5.6];
subplot(2,2,1);
plot(x,y)
title('This is the graph at position 1 of 2x2')
subplot(2,2,2);
semilogx(x,y)
title('This is the graph at position 2 of 2x2')
grid on
subplot(2,2,3);
semilogy(x,y)
title('This is the graph at position 3 of 2x2')
subplot(2,2,4);
loglog(x,y)
title('This is the graph at position 4 of 2x2')
grid on
```

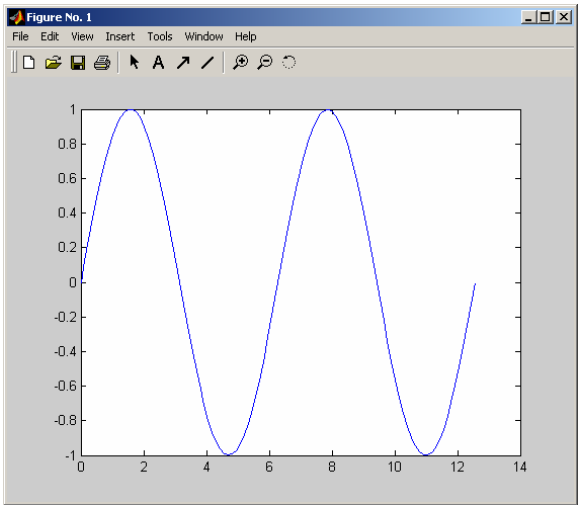
▸▸ การใช้คำสั่งบน Figure Windows เพื่อปรับปรุงลักษณะของกราฟ

การเขียนโปรแกรมเพื่อกำหนดรูปแบบต่างๆของกราฟอย่างง่ายเราอาจจะใช้คำสั่งที่กล่าวมาแล้วนี้ก็ได้ แต่หลังจาก MATLAB 5.3 เป็นต้นมาจนถึง MATLAB 6.0 ได้มีการปรับปรุงหน้าต่าง graphic windows โดยได้เพิ่ม toolbars และเมนูอื่นๆ เพิ่มเข้าไปในหน้าต่างมากขึ้น เพื่อที่จะให้เราสามารถที่จะปรับปรุง เปลี่ยนแปลงรูปแบบของกราฟได้มากขึ้น ยกตัวอย่างเช่นเราต้องการเขียน กราฟ sine ขึ้นมา ครั้งแรกใน MATLAB เราจะสั่ง

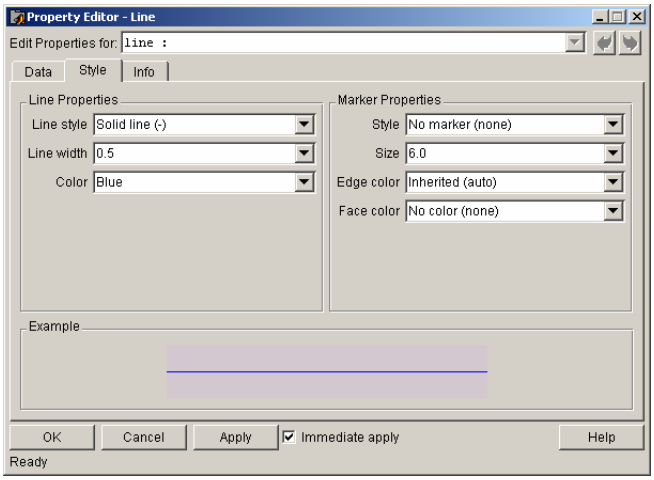
```
X=linspace(0,4*pi);
```

```
y=sin(x);
plot(x,y)
```

และเมื่อปรากฏ figure windows ขึ้นมาแล้ว เราสามารถที่จะเพิ่มชื่อแกน ชื่อกราฟ ลักษณะเส้น สีของเส้นกราฟ หรืออื่นๆ ได้อีก

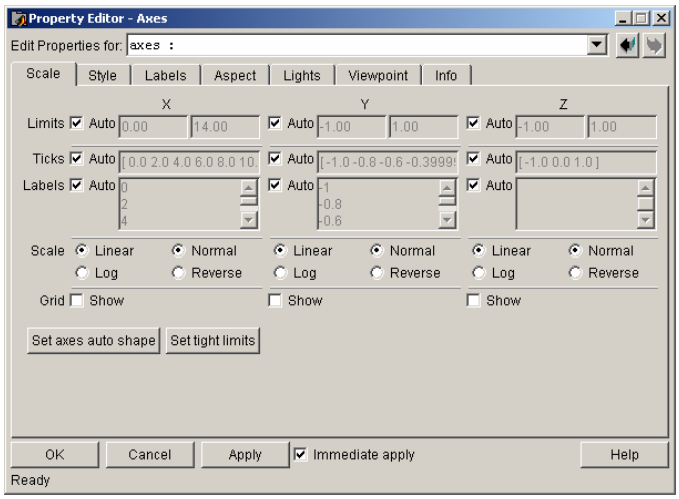


โดยเลือกที่เมนู Edit จากนั้นเลือก Current Object Properties ซึ่งจะทำให้เราได้หน้าต่าง Property Editor ซึ่งมีลักษณะดังต่อไปนี้



ซึ่งหน้าต่างนี้จะทำให้เราสามารถที่จะเลือกข้อมูลของกราฟ ว่าจะเลือกใช้เส้นแบบใด ความหนาของเส้นเป็นเท่าใด จุดบอกตำแหน่งข้อมูลเป็นเช่นไร และต่างๆ เหมือนกับการกำหนดค่าลงใน Plot เพียงแต่เราไม่จำเป็นที่จะต้องจำคำสั่งต่างๆ นั่นเอง

สำหรับหน้าต่างนี้จะเป็นหน้าต่างที่กำหนดลักษณะของกราฟ นอกเหนือจากนั้นเรายังสามารถกำหนดลักษณะของแกนของกราฟได้อีก โดยเราสามารถกำหนดได้โดยภายใต้เมนู Edit ให้เราเลือก Axes Properties ซึ่งเราจะได้หน้าต่าง Property Editor ของแกนขึ้นมา ซึ่งมีลักษณะดังนี้



หน้าต่างนี้เราสามารถที่จะปรับปรุงคุณสมบัติต่างๆ ของแกนได้ไม่ว่าจะเป็นการกำหนด ช่วง การแสดงค่าของแกน ลักษณะของแกน การกำหนดชื่อแกน และชื่อกราฟ กำหนดขนาด มุมของแสง และอื่นๆ อีกมาก อย่างไรก็ตามเราจะไม่ขอกล่าวถึงในรายละเอียดในที่นี้ เพราะการใช้งานหน้าต่างนี้ค่อนข้างง่าย เป็นไปตามตัวเลือกที่แสดงไว้ และในทำนองเดียวกันเราขอให้ลองใช้คำสั่งต่างๆที่มีอยู่บนเมนูของหน้าต่างนี้ คุณจะพบว่ามีหลายคำสั่งที่ช่วยในการทำงานให้รวดเร็วขึ้น และจำคำสั่งต่างๆ น้อยลง เพียงแต่บางคำสั่งอาจจะยังไม่เป็นที่เข้าใจในจุดนี้ เพราะอาจต้องศึกษาในบทที่กล่าวถึง Graphic ขั้นสูงเสียก่อนนั่นเอง

อย่างไรก็ตามการใช้หน้าต่างที่เรากล่าวถึงมาทั้งสองนี้ เหมาะสำหรับการที่เราทำงานแล้วต้องการปรับเปลี่ยนลักษณะของกราฟด้วยตนเอง แต่การเปลี่ยนแปลงด้วยวิธีนี้ไม่สามารถนำไปใช้ในกรณีที่เราเขียนโปรแกรมขึ้นเป็น M-file ได้ เรายังคงจะต้องใช้การสั่งการเป็นคำสั่งเหมือนเดิม ซึ่งทำให้เราต้องเข้าใจวิธีการสั่งเหมือนกับที่ผ่านมานั่นเอง ดังนั้นในตัวอย่างต่อไปนี้เราจะใช้คำสั่งที่กล่าวมาก่อนหน้านี้ในการปรับปรุงเส้นกราฟ หรือลักษณะแกน

การแก้ปัญหาทางวิศวกรรม

➢ การเขียนกราฟแสดงการไหลของ Potential Flow

สำหรับกราฟต่อไปนี้เป็นการแสดงการเขียนกราฟแสดง potential flow ผ่าน half-body (Uniform flow รวมกับ Source) ซึ่งมี stream function ของการไหลเป็น

$$\psi = Ur\sin\theta + \frac{q\theta}{2\pi}$$

เมื่อ U คือความเร็วของกระแสการไหลอิสระ

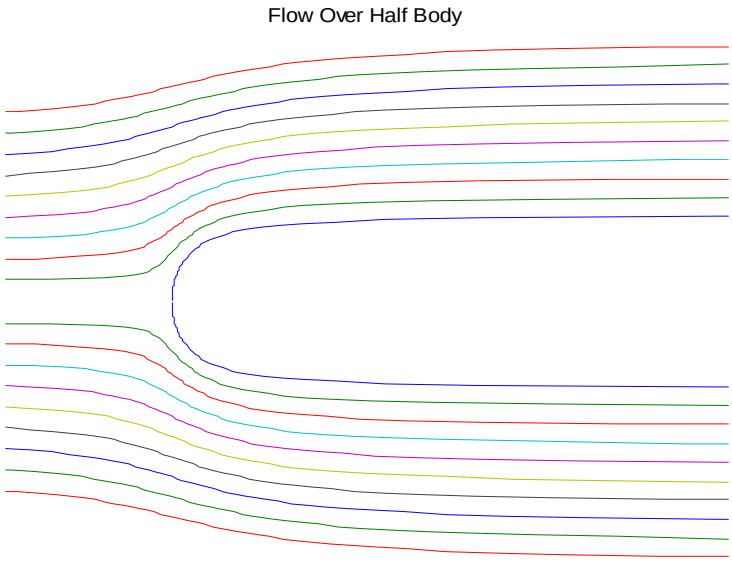
q เป็นค่าคงที่ค่า flow rate ของ source หรือเรียก source strength

ซึ่งเส้นที่ได้จากการ plot สมการ ψ = ค่าคงที่ จะเป็นเส้น stream line ดังนั้นในการ plot ขั้นแรกต้องมีการหาค่า $r = f(\theta)$ เสียก่อนนั่นคือ

$$r = \frac{\left(\psi - \frac{q\theta}{2\pi}\right)}{u\sin\theta}$$

จากนั้นสำหรับการ plot ในระบบแกน x-y ต้องมีการเปลี่ยนจาก polar coordinate โดยการใช้ความสัมพันธ์ $x = r\cos\theta$ และ $y = r\sin\theta$ ซึ่งสามารถใช้คำสั่งดังต่อไปนี้

```
zeta=linspace(pi/160,159*pi/160,50);
q=10;U=10;
phi=linspace(q/2,1.5*q,10);

for m=1:10        % ดูรายละเอียดของการทำ FOR-Loop ในบทที่กล่าวถึง M-FILES ค่า m ในที่นี้
                  % แสดงค่าของ stream function แต่ละค่า ซึ่งจะแทน stream linesแต่ละเส้น
   r(m,:)=(phi(m)-(q*zeta/(2*pi)))./(U.*sin(zeta));
end

r=abs(r');zeta=zeta';

for m=1:10
   x(:,m)=r(:,m).*cos(zeta);
   y(:,m)=r(:,m).*sin(zeta);
end

hold on
plot(x,y);y=-y;plot(x,y);
title('Flow Over Half Body')
axis([-2 6 -1.5 1.5])
axis off;hold off
```



สำหรับการไหลแบบนี้ ความจริงจะสะดวกกว่าหากว่าเราทำการ plot โดยใช้ Contour plot ซึ่งรายละเอียดจะอยู่ในส่วนที่กล่าวถึงการเขียนกราฟ 3 มิติ

➢ *การเขียนแสดงการสั่นของระบบ* Single Degree of Freedom

สำหรับการสั่นของระบบ Single Degree of Freedom Force Vibration จะมีรูปสมการการเคลื่อนไหวเป็น

$$m\ddot{x} + c\dot{x} + kx = F_0 \cos \omega t$$

เมื่อ m คือมวลของระบบ ที่มี stiffness k และ damping c ตกอยู่ภายใต้แรงกระทำแบบ harmonic F_0 ซึ่งมีความถี่ ω และ x เป็นการขจัดที่เกิดขึ้นกับมวลนี้ ซึ่งสมการนี้จะมีคำตอบของ x เป็น

$$x = Ae^{-\zeta\omega_n t} \sin(\omega_d t + \theta) + A_0 \cos(\omega t - \phi)$$

โดย parameter จะประกอบด้วย

$\omega_n = \sqrt{\frac{k}{m}}$ คือ natural frequency ของระบบ,

$\zeta = \frac{c}{2\sqrt{km}}$ คือ damping ratio ของระบบ

และ $\omega_d = \omega_n\sqrt{1-\zeta^2}$ คือ damping frequency ของระบบ

ซึ่งส่วนแรกของสมการจะเป็นลักษณะการสั่นชั่วครู่และจะหายไปเมื่อเวลาผ่านไปหรือเรียกว่า Transient Vibration สำหรับในส่วนหลังจะเป็นลักษณะการสั่นอย่างคงตัวหรือ Steady State ซึ่งขนาดของ

การสั่น A_0 เป็นสิ่งหนึ่งที่เราสนใจ เพราะจะเป็นเครื่องบ่งชี้ว่าระบบมีการสั่นมากน้อยเพียงใด ซึ่งสามารถพิจารณาได้ว่า

$$A_0 = \frac{f_0}{\sqrt{\left(\omega_n^2 - \omega^2\right)^2 + \left(2\zeta\omega_n\omega\right)^2}} = \frac{f_0}{\omega_n^2\sqrt{\left(1-r^2\right)^2 + \left(2\zeta r\right)^2}}$$

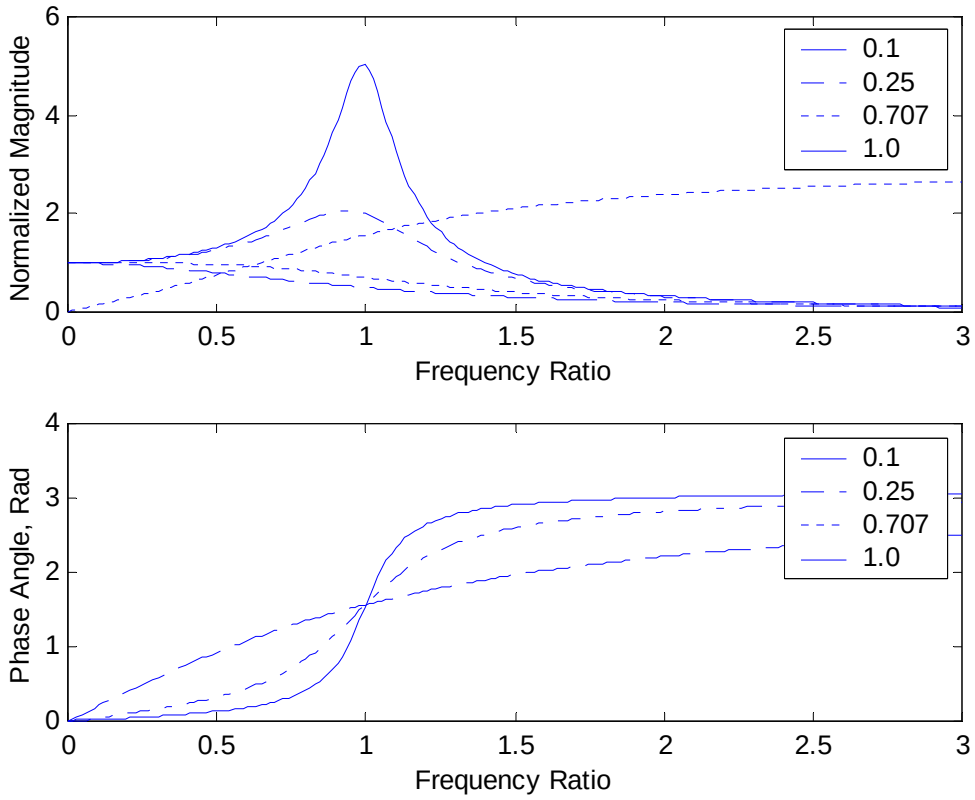
และ Phase angle จะเท่ากับ

$$\phi = \tan^{-1}\frac{2\zeta r}{1-r^2}$$

ซึ่งการเขียนกราฟผลตอบสนองของแรงกระทำที่มีต่อระบบนิยมเขียนในรูปของขนาดที่ไม่มีมิติ คือเขียน Normalized magnitude $\frac{A_0 k}{F_0}$ และ Phase angle, ϕ เทียบต่อ frequency ratio, r = $\frac{\omega}{\omega_n}$ เมื่อใช้ damping ratio ζ หลายๆ ค่า ซึ่งการเขียนกราฟเราสามารถทำได้ดังนี้สามารถทำได้ดังนี้

```
r=linspace(0,3,200);
r=r';
s=0.1;
x=1./sqrt((1-r.^2).^2+(2*s*r).^2);
phi=atan2(2*s*r,1-r.*r);
subplot(2,1,1)
a(1)=plot(r,x);
hold on
subplot(2,1,2)
b(1)=plot(r,phi);
hold on

s=0.25;
x=1./sqrt((1-r.^2).^2+(2*s*r).^2);
phi=atan2(2*s*r,1-r.*r);
subplot(2,1,1)
a(2)=plot(r,x,'b-.');
subplot(2,1,2)
b(2)=plot(r,phi,'b-.');

s=0.707;
x=1./sqrt((1-r.^2).^2+(2*s*r).^2);
phi=atan2(2*s*r,1-r.*r);
subplot(2,1,1)
a(3)=plot(r,x,'b:');
subplot(2,1,1)
b(3)=plot(r,phi,'b:');

s=1.0;
x=1./sqrt((1-r.^2).^2+(2*s*r).^2);
phi=atan2(2*s*r,1-r.*r);
subplot(2,1,1)
a(4)=plot(r,x,'b--');
xlabel('Frequency Ratio');
ylabel('Normalized Magnitude');
legend(a,'0.1','0.25','0.707','1.0')
subplot(2,1,2)
b(4)=plot(r,phi,'b--');
xlabel('Frequency Ratio');
ylabel('Phase Angle, Rad');
```

```
legend(b,'0.1','0.25','0.707','1.0')
```



บทที่ 3 MATLAB FUNCTIONS

เนื่องจาก MATLAB เป็นโปรแกรมสำเร็จรูป ดังนั้นในการเขียนชุดคำสั่งต่างๆ MATLAB ได้พยายามรวบรวม function ที่ใช้อยู่เป็นประจำเข้าไว้ด้วยกันเพื่อสะดวกในการใช้งาน function เหล่านี้จะอยู่ภายใน sub-directories ซึ่งมักจะเขียนเป็น file ที่เรียกว่า M-file การเขียน M-file นี้จะได้กล่าวถึงในบทต่อไป สำหรับ function ที่สำคัญได้รวบรวมเป็นหมวดหมู่เพื่อง่ายต่อการค้นหา

สำหรับการเรียกใช้ function นั้น ก็จะเรียกใช้ด้วยการกำหนดชื่อ function แล้วตามด้วย parameter ที่อยู่ในวงเล็บ ( ) หรือ

function(parameter$_1$,parameter$_2$,...)

เช่น

sin(x) คือการหาค่า sine ของมุม x

abs(x) คือการหาค่า absolute ของค่า parameter x

โดยทั่วไป x จะเป็น scalar, vector หรือ matrix ก็ได้

3.1 ฟังก์ชั่นทั่วไป

MATLAB มี function ทางคณิตศาสตร์พื้นฐาน เพื่อสะดวกแก่การใช้งานซึ่งได้แก่

abs(x) หาค่า absolute ของ x

sqrt(x) หาค่า รากที่ 2 ของ x

round(x) ทำให้ x เป็นจำนวนเต็ม โดยปัดค่าให้เป็นจำนวนเต็มที่ใกล้ x ที่สุด

fix(x) ทำให้ x เป็นจำนวนเต็ม โดยปัดค่า x ให้เป็นจำนวนเต็มที่ใกล้ศูนย์ที่สุด

floor(x) ทำให้ x เป็นจำนวนเต็ม โดยปัดค่า x ให้เป็นจำนวนเต็มที่ใกล้ x ไปทาง - infinity

ceil(x) ทำให้ x เป็นจำนวนเต็ม โดยปัดค่า x ให้เป็นจำนวนเต็มที่ใกล้ x ไปทาง + infinity

sign(x) บอกเครื่องหมายของ x โดยจะเป็น -1 ถ้า $x < 0$, เป็น 1 ถ้า $x > 0$ และเป็น 0 ถ้า $x = 0$

rem(x,y) หาเศษที่ได้จากการหาร x ด้วย y หรือ เศษของ x/y

exp(x) หาค่า e^x

log(x) หาค่า ln(x) หรือ natural logarithm ของ x

log10(x) หาค่า $\log_{10} x$ หรือ logarithm ฐาน 10 ของ x

3.2 ฟังก์ชันตรีโกณมิติ

สำหรับฟังก์ชันตรีโกณมิติ มุมที่ใช้ใน MATLAB จะเป็นมุม Radians ซึ่งมีความสัมพันธ์กับมุมองศา คือ $180° = \pi \ rad$

สำหรับฟังก์ชันตรีโกณมิติที่ใช้มีดังนี้

sin(x)	หาค่า sine ของมุม x radians
cos(x)	หาค่า cosine ของมุม x radians
tan(x)	หาค่า tangent ของมุม x radians
asin(x)	หาค่า arcsine หรือ inverse sine ของ x โดย $-1 \le \times \le 1$ และค่าที่ได้จะเป็นมุม radians มีค่าอยู่ระหว่าง $\frac{-\pi}{2}$ ถึง $\frac{\pi}{2}$
acos(x)	หาค่า arccosine หรือ inverse cosine ของ x โดย $-1 \le x \le 1$ และค่าที่ได้จะเป็นมุม radians มีค่าอยู่ระหว่าง 0 ถึง π
atan(x)	หาค่า arctangent หรือ inverse tangent x โดย x เป็นจำนวนจริงและค่าที่ได้จะอยู่ระหว่าง $\frac{-\pi}{2}$ ถึง $\frac{\pi}{2}$
atan2(x,y)	หาค่า arctangent หรือ inverse tangent ของ y/x โดย function จะให้ค่ามุม radians ซึ่งมีค่าอยู่ระหว่าง $-\pi$ ถึง π แล้วแต่เครื่องหมายของ x และ y

สำหรับ function ตรีโกณมิติอื่นแม้จะไม่มีอยู่ใน MATLAB แต่ก็สามารถหาได้ โดยใช้ความสัมพันธ์ทางตรีโกณมิติ คือ

$$\sec(x) = \frac{1}{\cos(x)}, \quad \cos ec(x) = \frac{1}{\sin(x)}, \quad \cot(x) = \frac{1}{\tan(x)}$$

$$arc\sec(x) = arc\cos\left(\frac{1}{x}\right) \quad \text{โดย} \quad |x| \ge 1$$

$$arc\cos(x) = arc\sin\left(\frac{1}{x}\right) \quad \text{โดย} \quad |x| \ge 1$$

$$arc\cot(x) = arc\cos\left(\frac{x}{\sqrt{1+x^2}}\right)$$

3.3 Hyperbolic Functions

Hyperbolic Functions เป็น function ของ natural exponential function, e^x ส่วน inverse ของ hyperbolic function นั่นจะเป็น function ของ natural logarithm functions, MATLAB ได้สร้าง function หลายค่าสำหรับการคำนวณหาค่า hyperbolic function ซึ่งประกอบด้วย

sinh(x)	หาค่า hyperbolic sine ของ x ซึ่งเท่ากับ $\frac{e^x - e^{-x}}{2}$
cosh(x)	หาค่า hyperbolic cosine ของ x ซึ่งเท่ากับ $\frac{e^x + e^{-x}}{2}$

tanh(x) หาค่า hyperbolic tangent ของ x ซึ่งเท่ากับ $\frac{\sinh(x)}{\cosh(x)}$

asinh(x) หาค่า inverse hyperbolic sine ของ x ซึ่งเท่ากับ $\ln\left(x+\sqrt{x^2+1}\right)$

acosh(x) หาค่า inverse hyperbolic cosine ของ x ซึ่งเท่ากับ $\ln\left(x+\sqrt{x^2-1}\right)$ โดย $x \geq 1$

atanh(x) หาค่า inverse hyperbolic tangent ของ x ซึ่งเท่ากับ $\ln\left(\sqrt{\frac{1+x}{1-x}}\right)$ โดย $|x| \leq 1$

สำหรับค่า hyperbolic function อื่นที่ไม่ได้บรรจุใน MATLAB function สามารถหาได้จากความสัมพันธ์

$$\coth(x) = \frac{\cosh(x)}{\sinh(x)} \quad \text{โดย} \quad x \neq 0, \qquad \operatorname{sech}(x) = \frac{1}{\cosh(x)}$$

$$\operatorname{csch}(x) = \frac{1}{\sinh(x)}, \qquad \operatorname{acoth}(x) = \ln\left(\frac{x+1}{x-1}\right) \quad \text{โดย} \quad |x \geq 1|$$

$$\operatorname{acsch}(x) = \ln\left(\frac{1}{x}+\frac{\sqrt{1+x^2}}{|x|}\right), \qquad \operatorname{asech}(x) = \ln\left(\frac{1+\sqrt{1-x^2}}{x}\right) \quad \text{โดย} \quad 0 \leq x \leq 1$$

3.4 Complex Number Functions

Complex number z นิยามด้วยจำนวนจริงสองจำนวนคือ

$$z = x + iy$$

เมื่อ x และ y เป็น real และ $i = \sqrt{-1}$

x เป็น real part ของ z ; x = Re(z)

y เป็น imaginary part ของ z ; y = Im (z)

สำหรับ MATLAB เมื่อเริ่มต้นจะกำหนดค่า i และ j ให้เท่ากับ $\sqrt{-1}$ ไว้โดยจะใช้ i หรือ j เป็น imaginary number ก็ได้ สำหรับ function ของ complex number ที่มีใน MATLAB โดย parameter z = x + iy มีดังนั้น

conj(z) หา complex conjugate ของ z คือ ถ้า z = x + iy, จะให้ค่า x - iy

real(z) หา Real part ของ z คือ ถ้า z = x + iy, จะเท่ากับ x

imag(z) หา imaginary part ของ z คือ ถ้า z = x + iy, จะเท่ากับ y

abs(z) หาค่า absolute หรือขนาดของ z จะให้ค่าผลลัพธ์เป็นค่าของ $\sqrt{x^2+y^2}$

angle(z) หาค่ามุมหรือ argument ของ z คือ angle (z) = $\tan^{-1}\frac{y}{x}$ โดยค่ามุมที่ได้จะอยู่ระหว่าง $-\pi$ ถึง π

3.5 Polynomial Functions

สำหรับ polynomial

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + ...+ a_2 x^2 + a_1 x + a_0$$

โดย $a_n, a_{n-1}, ...a_o$ เป็น coefficient ของ polynomial P(x) ซึ่งมีกำลังสูงสุด เท่ากับ n หรือ polynomial degree n สำหรับการสร้าง polynomial degree n ใน MATLAB จะทำได้โดยการสร้าง coefficient เป็น vector ที่มีจำนวนเท่ากับ n+1 หรือ

$$P = [a_n, a_{n-1}, ... a_2, a_1, a_0]$$

จากนั้นเราสามารถใช้ vector P นี้หาค่าต่างๆ ที่เกี่ยวข้องกับ polynomial ได้ สำหรับ function ที่ใช้หาค่าต่างๆ ของ polynomial นี้มีดังต่อไปนี้

polyval(P,x) หาค่า polynomial ที่มี coefficient P ที่ x ถ้า x เป็น scalar จะให้ค่า function ที่จุด x ถ้า x เป็น vector ผลที่ได้จะเป็น vector ที่มีขนาดเดียวกันกับ x และแต่ละ element จะเป็นค่าของ function ที่ element นั้นของ x

ตัวอย่างเช่น

```
P=[1, 2, 1]        % มีความหมายเหมือน f(x) = x^2 + 2x + 1
Polyval(P,1)%มีความหมายเหมือนต้องการหาค่า f(1) ซึ่งเท่ากับ 1
ans =
     1

X=[1 2 3 4];
polyval(P,X)
ans =
 4  9  16  25
```

❶ รากของ Polynomial

รากของ polynomial P(x) คือค่าของ x ที่ทำให้ P(x) = 0 โดยรากของสมการนี้จะมีจำนวนเท่ากับ order (เลขยกกำลัง) สูงสุดของpolynomial ซึ่งรากแต่ละรากที่ได้นั้นอาจจะเป็นจำนวนจริง หรือ complex number ก็ได้นอกจากนี้ยังมีโอกาสที่จะเป็นรากซ้ำได้อีกด้วย สำหรับ MATLAB สามารถหาค่ารากของ polynomial ได้โดยใช้คำสั่ง roots

roots(A) จะได้คำตอบเป็นค่าราก polynomial A

ตัวอย่างเช่นต้องการหาค่ารากของ $x^4 = 4$ หรือรากของ $x^4 - 4 = 0$

```
A=[1,0,0,0,-4];
B=roots(A)
B =
 -1.4142
  0.0000+ 1.4142i
```

```
0.0000- 1.4142i
1.4142
```

จะได้ว่ารากทั้ง 4 เป็น $\pm\sqrt{2},\pm\sqrt{2}i$ หรือตัวอย่างเช่น

```
D=[1,3,0,-1,0,2];
C=roots(D)
C =
  -2.9098
  -0.7186+ 0.6574i
  -0.7186- 0.6574i
   0.6735+ 0.5207i
   0.6735- 0.5207I
```

❷ การหา polynomial ที่มีรากตามที่กำหนด

เป็นการใช้คำสั่งให้ MATLAB คำนวณย้อนกลับจากคำสั่ง roots คือเป็นการหา polynomial หนึ่งซึ่งจะมีรากเท่ากับค่าที่กำหนดให้ โดยใช้คำสั่ง poly

poly(A) คำนวณหา polynomial ที่มีรากเท่ากับ vector A

ตัวอย่างเช่น

```
B=[1,-3,-1,3];
A=roots(B)
A =
      3.0000
      1.0000
     -1.0000
C=poly(A)
C =
      1.0000   -3.0000   -1.0000    3.0000
```

จะเห็นว่า C จะเท่ากับ B

❸ การคำนวณที่เกี่ยวข้องกับ polynomial

- การคูณ polynomial ด้วย scalar ก็คือการคูณสัมประสิทธิ์ทุกค่าของ polynomial นั้นด้วย scalar เช่น

```
P=[1, 2, 1]                % กำหนด P(x) = x^2 + 2x + 1
Q=2*P          %กำหนด Q(x) = 2(x^2 + 2x + 1) = 2x^2 + 4x + 2
polyval(Q,1)               % หาค่า Q(1)
ans =
      8
```

- การคูณ polynomial ด้วย polynomial ค่อนข้างจะยุ่งยากกว่าการคูณด้วยค่าคงที่ เพราะการคูณในลักษณะนี้จะเป็นการคูณทุกเทอม คำสั่งที่ใช้ใน MATLAB คือ **conv**

conv(P,Q) หาค่าของสัมประสิทธิ์ของ polynomial ที่ได้จากการคูณ polynomial P และ Q เข้าด้วยกัน

ตัวอย่างเช่น

```
P=[1 2 1];      % กำหนด P(x) = x^2 + 2x + 1
Q=2*P;          % กำหนด Q(x) = 2(x^2 + 2x + 1) = 2x^2 + 4x + 2
conv(P,Q)       % หาค่า P(x)*Q(x)
ans =
     2     8    12     8     2
```

- การหาร polynomial ด้วย polynomial เป็นวิธีการที่ค่อนข้างยุ่งยาก เราคงจะจำขั้นตอนที่ใช้ในการหารสมัยที่ยังเรียนอยู่ในชั้นมัธยมได้ สำหรับ MATLAB นั้นมีคำสั่งที่ใช้หาร polynomial เช่นกันโดยใช้คำสั่ง decond

`[A,B]=deconv(C,D)` ให้ MATLAB คำนวณหาผลหารของ polynomial C ด้วย polynomial D แล้วให้ค่าเป็นpolynomial A และมีค่าเศษที่เหลือเป็น polynomial B

ตัวอย่างเช่น

```
C=[3,4,-9,13,-1,1.5,-10.5,10,10];
D=[1,3,0,-1,0,2.5];
[A,B]=deconv(C,D)
A =
    3    -5     6    -2
B =
    0     0     0     0     0     0     0    -5    15
```

นั่นคือผลหารจะได้ polynomial $3x^3 - 5x^2 + 6x - 2$ และเหลือเศษเป็น polynomial $-5x + 15$

สำหรับค่า parameter ที่ใช้สำหรับ function ต่าง ๆ ที่เกี่ยวข้องกับ polynomial นั้นในตัวอย่างต่าง ๆ จะถือว่าเป็น vector แต่ถ้าหากว่าเรากำหนดค่า parameter เป็น matrix แล้ว MATLAB ก็จะทำการคำนวณเป็น row ทีละ 1 row แล้วค่าที่ได้ก็จะเป็น matrix ที่มีจำนวน row เท่ากับ row ของ parameter ของเรา

3.6 Statistical and Logical Functions

❶ Statistical Functions

สำหรับ function ทางสถิติที่ใช้ใน MATLAB จะประกอบด้วย function หลัก ๆ ดังต่อไปนี้

`max(x)` MATLAB จะให้ค่าที่มากที่สุดของ x ถ้า x เป็น matrix จะได้ผลเป็น row vector ที่บรรจุค่าสูงสุดในแต่ละ column ของ x ถ้า x เป็น complex number จะได้ผลเป็นค่าที่มีขนาดสูงสุด

`[y,ind]=max(x)` MATLAB จะให้ผลเป็น row vector y ที่บรรจุค่าที่สูงสุดในแต่ละ column ของ matrix x และ ind จะบรรจุตำแหน่งของแต่ละ element ในแต่ละ column ของ x

max(A,B) MATLAB จะให้ผลเป็น matrix มีขนาดเท่ากับ A และ B โดยแต่ละ element ที่ตำแหน่ง (i, j) จะเป็นค่าที่สูงสุดเมื่อเทียบระหว่าง a_{ij} และ b_{ij}

สำหรับการคำนวณเพื่อหาค่าน้อยที่สุด จะใช้ function **min** ซึ่งมีการใช้เช่นเดียวกับ function max

sum(x) ถ้า x เป็น vector จะได้ผลเป็นการรวมค่าทั้งหมดของ element ใน x แต่ถ้า x เป็น matrix จะได้ผลเป็น row vector ที่บรรจุผลบวกแต่ละ column ของ x ไว้

cumsum(x) ถ้า x เป็น vector จะได้ผลเป็นการรวมสะสม ของ x นั่นคือ element ที่ 2 เป็นผลรวมของ element 1 กับ 2 ของ x , element ที่ 3 จะเป็นการรวมของ element ที่ 1, 2 และ 3 ของ x ไปเรื่อย ๆ ถ้า x เป็น matrix MATLAB จะพิจารณาเช่นเดียวกันในแต่ละ column ของ x

prod(x) จะได้ผลคล้ายกับ **sum** แต่เปลี่ยนเป็นผลคูณของ element

cumprod(x) จะได้ผลคล้ายกับ **cumsum** แต่เปลี่ยนเป็นผลคูณของ element

mean(x) ถ้า x เป็น vector จะได้ผลเป็นค่ากลางของ element ของ x ถ้า x เป็น matrix จะได้ row vector ที่มีแต่ละ element เป็นค่ากลางของแต่ละ column ของ x

median(x) ถ้า x เป็น vector จะได้ผลเป็น median ของ element ของ x ถ้า x เป็น matrix จะได้ row vector ที่มีแต่ละ element เป็นค่า medien ของแต่ละ column ของ x

std(x) ถ้า x เป็น vector จะได้ผลเป็นส่วนเบี่ยงเบนมาตรฐาน (standard deviation) ของ element ทั้งหมดของ x ถ้า x เป็น matrix จะได้ผลเป็น row vector ที่แต่ละ element เป็นค่าเบี่ยงเบนมาตรฐานของแต่ละ column ของ x

cov(x) ถ้า x เป็น vector จะได้ผลเป็น Varian ของ x ถ้า x เป็น matrix จะได้ผลเป็น diagonal matrix ซึ่งแต่ละค่าจะเป็น Varian ของแต่ละ column ของ x

corrcoef(A) จะได้ผลเป็น correlation matrix ของ matrix A

sort(x) ถ้า x เป็น vector จะได้ผลเป็น vector ที่มีการเรียงลำดับของ element จากค่าน้อยไปมาก ถ้า x เป็น matrix จะได้ผลเป็น matrix ขนาดเท่ากันโดยแต่ละ column จะเป็นการเรียงลำดับจากน้อยไปหามากของแต่ละ column ของ x

[y,ind] = sort(x) จะได้ y เป็นผลของคำสั่ง sort (x) ส่วน ind คือผลที่ได้ในการเทียบ index ของ element ก่อนมีการเรียงลำดับของ x

ตัวอย่างต่อไปนี้กำหนดให้ column vector x, row vector y, matrix A มีค่าดังต่อไปนี้

```
x=[4;1;6]
x =
     4
     1
     6
y=[1 8 9]
y =
     1     8     9

A=[1 9 8;2 12 4;4 7 6]
A =
     1     9     8
     2    12     4
     4     7     6
```

จะได้ว่า

```
max(y)
ans =
     9

min(A)
ans =
     1     7     4

sort(x)
ans =
     1
     4
     6

mean(A)
ans =
    2.3333    9.3333    6.0000

sum(y)
ans =
    18

cumsum(x)
ans =
     4
     5
    11

cumprod(A)
ans =
     1     9     8
     2   108    32
     8   756   192
```

❷ Logical Function

ใน MATLAB จะมี function ทางตรรกศาสตร์หลาย function เพราะในบางกรณี เราจำเป็นต้องเปรียบเทียบ, ค้นหา element ที่อยู่ภายใน matrix ต่าง ๆ หรือตรวจสอบ element ภายใน matrix ซึ่ง function ทางด้านนี้ประกอบด้วย

find(X)	จะได้ผลเป็น vector ของตำแหน่ง ที่มี element ที่ไม่ใช่ศูนย์ใน x ถ้า x เป็น matrix ก็จะได้ผลเป็น column vector โดยจะนำผลของแต่ละ column ของ x มาเรียงต่อกันเป็น vector
any(X)	จะให้ค่า 1 ถ้ามี element ใดของ x ไม่เป็นศูนย์ แต่ถ้ามี element ใดของ x เป็นศูนย์จะให้ค่า 0 ในกรณีที่ x เป็น matrix จะได้ผลเป็น row vector ซึ่งผลแต่ละ column จะพิจารณาแต่ละ column ของ x
all(X)	จะให้ค่า 1 ถ้าทุก element ของ x ไม่เป็นศูนย์ แต่ถ้ามี element ใดของ x เป็นศูนย์จะให้ค่า 0 ในกรณีที่ x เป็น matrix จะให้ผลเป็น row vector ซึ่งผลแต่ละ column จะพิจารณาแต่ละ column ของ x
isnan(X)	จะให้ผลเป็น 1 ถ้าหากว่ามี element ของ x นั้นไม่ใช่ตัวเลข (Not-a-Number, NaN) และให้ผลเป็น 0 ถ้าไม่เป็นเช่นนั้น
finite(X)	จะให้ผลเป็น 1 หากว่ามี element ของ x มีค่าหรือ finite และเป็น 0 ถ้าหากมี element ของ x เป็น NaN หรือ infinity
isempty(X)	จะให้ผลเป็น 1 ถ้า x เป็น matrix ว่าง ไม่เช่นนั้นจะให้ผลเป็น 0

ตัวอย่างเช่น กำหนดให้

```
B=[1 0 4;3 6 0;4 -2 9]
B =
  1  0  4
  3  6  0
  4 -2  9
```

จะได้

```
any(B)
ans =
     1     1     1

find(B)
ans =
     1
     2
     3
     5
     6
     7
     9

all(B)
ans =
     1     0     0
```

```
finite(B)
ans =
     1     1     1
     1     1     1
     1     1     1
```

3.7 File Input/Output Functions

แม้ว่า MATLAB จะเป็นโปรแกรมที่มีความสามารถในการทำงานในการคำนวณได้มากมาย แต่ในหลายๆกรณีก็มีความจำเป็นที่จะต้องติดต่อ ส่งผ่านข้อมูลกับโปรแกรมอื่น หรือต้องมีการเก็บข้อมูลเพื่อที่จะนำไปใช้ต่อไป ดังนั้นจึงจำเป็นอย่างยิ่งที่จะต้องมีระบบการบันทึกและรับส่งข้อมูล สำหรับในหัวข้อนี้จะกล่าวถึงวิธีการเปิด-อ่าน-เขียน-ปิด file ของ MATLAB

สำหรับผู้ที่ยังไม่มีประสบการณ์มากนักกับการเปิด-ปิด file ขอให้พิจารณาหลักง่ายๆดังนี้ หากเราเดินเข้าไปที่ชั้นหนังสือ มองดูหนังสือที่มีอยู่มากมาย เรามีความต้องการที่จะทราบข้อมูลในหนังสือเล่มใดเล่มหนึ่งหรือหลายๆเล่มเราจะทำอย่างไร อันดับแรกก็คงจะเป็นการหยิบหนังสือที่มีชื่อที่ตรงกับที่เราสนใจขึ้นมา เปิดหนังสือไปที่หน้าที่ต้องการ อ่านข้อความเมื่อเสร็จสิ้นแล้วก็ปิดหนังสือเล่มนั้นเก็บเข้าที่เดิม ในการเปิด file ก็เช่นเดียวกันขั้นแรกก็จะต้องบอกเครื่องคอมพิวเตอร์ว่าในจำนวน file ที่มีมากมายใน disk นั้นเราต้องการจะเปิด file ใด จากนั้นก็อ่านข้อมูล หรือเขียนข้อมูลลงไป แล้วก็ทำการปิด file นั้นให้เรียบร้อย

❶ การเปิดและปิด file

คำสั่งแรกที่จะให้ MATLAB เปิด file ขึ้นมาก่อนก็คือคำสั่ง fopen

`fid=fopen(filename, permission)`

เป็นการเปิด file ตามชื่อ filename และกำหนดลักษณะการเปิด file ด้วย permission ถ้า file ที่เปิดขึ้นสำหรับที่จะอ่านค่าแล้วไม่ปรากฏใน directory ที่ MATLAB กำลังทำงานอยู่ MATLAB จะไปหาใน search path. สำหรับผลรับ fid ที่ได้จะเป็น file identifiers หรือก็คือหมายเลขกำหนด file ว่าขณะที่กำลังทำงานอยู่นั้น MATLAB จะจำ file นี้ด้วยหมายเลขอะไร สำหรับเงื่อนไขลักษณะการเปิดหรือ permission เป็น strings ตัวใดตัวหนึ่งต่อไปนี้

- **`'r'`** เปิดเพื่ออ่านอย่างเดียว (read)
- **`'w'`** เปิดขึ้นเพื่อเขียนหากไม่มี file ชื่อดังกล่าวก็จะสร้างขึ้นใหม่
- **`'a'`** เพิ่มเติม (append) สร้างใหม่ถ้าจำเป็น
- **`'r+'`** อ่านและเขียน ไม่มีการสร้างใหม่
- **`'w+'`** ตัดหรือสร้างใหม่สำหรับการอ่านและเขียน
- **`'a+'`** อ่านและเพิ่มเติม สร้างใหม่ถ้าจำเป็น

สำหรับในเบื้องต้นนั้น file ที่เปิดจะมี format เป็น binary code หากว่าต้องการจะเปิด file ที่เป็น text จะต้องเพิ่ม 't' เข้าไปใน string ตัวอย่างเช่น **'rt'** หรือ **'wt+'** อย่าลืมว่าบนเครื่อง UNIX และ Macintosh นั้น text file และ binary file จะเหมือนกันแต่สำหรับเครื่อง PC file ทั้งสองจะแตกต่างกันโดยสิ้นเชิง สำหรับในกรณีที่การเปิด file ไม่สำเร็จ MATLAB จะให้ค่า fid = -1 นอกจากนี้ fid =1, 2 และ 3 MATLAB จะสำรองไว้ใช้งานเอง
สำหรับชื่อ file นั้นจะต้องเขียนเป็น string และสำหรับเครื่อง PC การกำหนด directory จะทำตามมาตราฐานของ DOS

fids=fopen('all') จะให้ row vector ที่มี file identifiers สำหรับทุก file ที่กำลังเปิดทำงานอยู่ใน ขณะนั้น

ส่วนการปิด file จะใช้คำสั่ง fclose ดังมีรูปแบบต่อไปนี้

st=fclose(fid) ปิด file ตาม file identifier, fid, ที่กำหนด ซึ่งได้จากการใช้คำสั่ง fopen ก่อน หน้านี้ ถ้า st =0 แสดงว่าปิด file ได้เรียบร้อย และ st = -1 แสดงว่าไม่เรียบร้อย

st=fclose('all') ปิดทุก file ที่เปิดอยู่

❷ การเขียนและอ่าน Binary File

สำหรับการเขียนและอ่าน binary file สามารถใช้คำสั่ง **fread** และ **fwrite** ซึ่งมีรูปแบบของคำสั่งดังต่อไปนี้คือ สำหรับการอ่าน

[A,count]=fread(fid,size,precision)

อ่าน binary data จาก file ที่กำหนดโดย fid และกำหนดค่าที่ได้ให้เป็นค่าของ matrix A ส่วนตัวเลือกเพิ่มเติม count จะบอกว่าได้มีการอ่านข้อมูลสำเร็จไปเท่าใด นอกเหนือจากนั้นตัวกำหนดขนาด size จะเป็นตัวเลือกหากไม่มีการกำหนด MATLAB จะอ่านจนกระทั่งหมด file แต่ถ้าจะกำหนดจะสามารถกำหนดได้ดังนี้

N	อ่าน N element เข้าเป็น column vector
inf	อ่านจนจบ file (ค่าเบื้องต้น)
[M,N]	อ่านข้อมูลให้เข้าสู่ matrix ขนาด M x N เรียงตาม column

และสำหรับการเขียน

count=fwrite(fid,a,precision)

เขียน element ของ matrix A ลงไปใน file ที่กำหนดโดย fid โดย MATLAB จะเขียนตามลำดับ column ส่วนตัวเลือก count เป็นค่าของจำนวน elements ที่เขียนได้สำเร็จ

สำหรับ **precision** เป็นการกำหนดรูปแบบตัวแปรและความแม่นยำเป็นไปตามตารางต่อไปนี้

String	ความหมาย
'char'	ตัวอักษร - 8 bits
'uchar'	ตัวอักษรแบบ unsigned character - 8 bits
'schar'	ตัวอักษรแบบ signed character - 8 bits
'int8', 'int16','int32','int64'	จำนวนเต็ม - 8,16,32 และ 64 bits
'uint8', 'uint16', 'uint32', 'uint64'	จำนวนเต็มแบบ unsigned integer - 8,16,32 และ 64 bits
'float32'	floating point - 32 bits
'float64'	floating point - 64 bits

นอกเหนือจากนั้นยังสามารถใช้ค่าต่อไปนี้ได้ แต่อาจจะมีปัญหาหากใช้เครื่อง computer ที่มีรูปแบบการจัดโครงสร้างที่ต่างกัน

'short'	จำนวนเต็ม - 16 bits.
'int'	จำนวนเต็ม - 32 bits.
'long'	จำนวนเต็ม- 32 หรือ 64 bits.
'ushort'	unsigned integer - 16 bits.
'uint'	unsigned integer - 32 bits.
'ulong'	unsigned integer - 32 bits or 64 bits.
'float'	floating point - 32 bits
'double'	floating point - 64 bits.

ตัวอย่างการเปิดเพื่ออ่านหรือเขียน binary file เช่น

- ขั้นแรกเขียน file ใหม่

```
f1=fopen('c:\MATLABR11\work\myio1.dat','w');
A=linspace(0,2*pi,10);
st =fwrite(f1,A,'float32')
```

```
st =
    10
```

- การอ่าน file ให้จบในครั้งเดียว

```
f2=fopen('c:\MATLABR11\work\myio1.dat','r');
B=fread(f2,'float32');
fclose(f2);
```

- การอ่าน file แบบมีรูปแบบ

```
f3=fopen('c:\MATLABR11\work\myio1.dat','r');
C=fread(f2,[2 5],'float32');
fclose(f3);
```

- การอ่าน file แบบมีรูปแบบให้ตัวแปรหลายตัว

```
F4=fopen('c:\MATLABR11\work\myio1.dat','r');
D=fread(f2,[2,2],'float32');
E=fread(f2,[3,2],'float32');
fclose(f4);
```

- หากลองพิจารณาค่าตัวแปร B C D และ E จะได้

```
B =
         0
    0.6981
    1.3963
    2.0944
    2.7925
    3.4907
    4.1888
    4.8869
    5.5851
    6.2832

C =
         0    1.3963    2.7925    4.1888    5.5851
    0.6981    2.0944    3.4907    4.8869    6.2832

D =
         0    1.3963
    0.6981    2.0944

E =
    2.7925    4.8869
    3.4907    5.5851
    4.1888    6.2832
```

สำหรับการอ่าน binary file นี้เป็นไปตามมาตราฐาน IEEE ซึ่งสามารถใช้ได้ทั้งรูปแบบของ MATLAB ตามที่กล่าวมาแล้วหรืออาจใช้วิธีการกำหนดตามรูปแบบของ C หรือ FORTRAN ก็ได้ ดูรายละเอียดใน **help fread**

❸ การอ่านและเขียน Format File

สำหรับ text file หรือ file ที่มีรูปแบบเฉพาะจะมีขั้นตอนการอ่านและเขียนแตกต่างจาก binary file โดยคำสั่งที่ใช้จะเป็น fscanf และ fprintf ซึ่งมีรายละเอียดดังนี้ สำหรับการอ่าน

[a,count]=fscanf(fid,format,size)

อ่านค่าจาก format file ตาม fid ที่กำหนดจากนั้นเปลี่ยนให้มาเป็นรูปแบบที่กำหนดตามคำสั่ง format แล้วให้ค่าเป็น matrix a ส่วนตัวเลือก count เป็นค่าที่บอกว่าได้อ่านข้อมูลมาเท่าใดแล้ว ส่วน size เป็นตัวเลือกที่กำหนดว่าจะอ่านค่าครั้งละเท่าใด เหมือนที่ใช้กับคำสั่ง fread สำหรับ format string จะมีลักษณะเป็น เครื่องหมาย % ตามด้วยตัวอักษรต่อไปนี้คือ d, i, o, u, x, e, f, g, s, c, และ [. . .] (scanset). ซึ่งจะเป็นรูปแบบของภาษา C ที่ใช้กันทั่วไปก็เช่น %s เป็น character string, %f เป็น floating point และ %d เป็น decimal point integer เป็นต้น การอ่านข้อมูล MATLAB จะอ่านเรียงบรรทัดจากบนลงล่าง

สำหรับการเขียนจะใช้

count=fprintf(fid,format,a)

เขียน format data เฉพาะในส่วน real part ของ matrix a ภายใต้เงื่อนไขของ format string ลงบน file ที่เปิดขึ้นตาม fid ส่วน count เป็นตัวเลือกเพื่อบอกจำนวนข้อมูลที่เขียนลงไปบน file ได้สำเร็จ

ตัวอย่างเช่น ในการเขียน file เป็น text file ชื่อ iotest.txt จะใช้

```
x=linespace(0,5,10);
y=[x,cos(x)];
f1=fopen('c:\MATLABR11\work\iotest.txt','wt');
fprintf(f1,'%8.4f %8.4f\n',y);
fclose(f1);
```

และการอ่านสามารถทำได้โดย

```
f2=fopen('c:\temp\iotest.txt','rt');
A=fscanf(f2,'%f %f',[2 5]);
B=fscanf(f2,'%f');

A
A =
         0    1.1111    2.2222    3.3333    4.4444
    0.5556    1.6667    2.7778    3.8889    5.0000
```

```
B
B =
    1.0000
    0.8496
    0.4437
   -0.0957
   -0.6063
   -0.9345
   -0.9817
   -0.7335
   -0.2647
    0.2837
```

แม้ว่าในบทนี้จะกล่าวถึง function ของ MATLAB มาแล้วหลาย function ก็ตามแต่โดยความเป็นจริงแล้วยังคงมี function ของ MATLAB อีกมากที่ไม่ได้กล่าวถึงในที่นี้ ทั้งนี้หากผู้อ่านต้องการทราบรายละเอียดเพิ่มเติมประการใด ควรจะดูประกอบกับหนังสือคู่มือหรือ help files

การแก้ปัญหาทางวิศวกรรม

Stress-Strain Curve และค่าคุณสมบัติของวัสดุ

ในการศึกษาวิศวกรรมศาสตร์ทางด้านกลศาสตร์ของวัสดุ curve อย่างหนึ่งที่เราจะพบเห็นกันบ่อยๆ คือ stress-strain curve เพราะจะเป็น curve ที่บอกถึงค่าความยืดหยุ่นของวัสดุ บอกถึงความสามารถในการทนต่อความเค้นและความเครียดของวัสดุนั้น

สมมุติว่าจากการทดลองกับโลหะหนึ่งเราพบว่าความสัมพันธ์ระหว่าง stress และ strain เป็นไปตามตารางข้างล่างนี้ ขอให้เราหาสิ่งต่อไปนี้

- ค่า Young's Modulus ของวัสดุในหน่วย MPa
- ค่า Ultimate Strength ของวัสดุในหน่วย MPa
- ค่า Percent Elongation at Fracture ของโลหะ

Stress (kpsi)	Strain (in/in)	Stress (kpsi)	Strain (in/in)
0	0	76	0.08
30	0.001	75	0.10
55	0.002	73	0.12
60	0.005	69	0.14
68	0.01	65	0.16
72	0.02	56	0.18
74	0.04	51	0.19 (Fracture)
75	0.06		

โดยที่ค่าที่อยู่ในตารางนี้ได้เก็บไว้ใน file ที่ชื่อ tensile.dat โดยข้อมูลจะเก็บไว้เป็น 2 column สำหรับ stress และ strain เป็นแต่ละ column

ในการพิจารณาที่หน่วยของความเค้น จะพบว่าเราต้องการทำการเปลี่ยนหน่วยจาก kpsi มาเป็น MPa เสียก่อน โดยใช้ความสัมพันธ์ 1 kpsi = 6.895 MPa ส่วนความเครียดนั้นเป็นปริมาณที่ไม่มีมิติอยู่แล้ว จึงไม่มีความจำเป็นต้องเปลี่ยนหน่วย ลำดับแรกต้องมีการนำข้อมูลเข้าสู่ MATLAB ก่อน เราใช้คำสั่ง fopen

```
fid = fopen('c:\MATLABR11\work\tensile.dat','r');
A=fscanf(fid,'%f %f',[2 100]);
fclose(fid);
A=A';
```

จะเห็นว่าเนื่องจากเราไม่ทราบว่าจะมีการทดลองแล้วได้ข้อมูลมากี่ชุด เราจึงเผื่อว่าคงจะไม่เกิน 100 ชุดข้อมูล แต่เนื่องจากการอ่านค่าจาก file จะอ่านเป็นบรรทัด ซึ่งข้อมูลที่บรรจุไว้จะมีบรรทัดละ 2 ค่าคือของ stress และ strain อย่างละค่า ดังนั้นจึงต้องอ่านมาในลักษณะ 2 x 100 แล้วจึงมาใช้ transpose กลับมาให้แต่ละ column แทนค่าความเค้นและความเครียด ต่อจากนั้นจะแยกค่าความเค้นและความเครียดออกจากกันและมีการเปลี่ยนหน่วยในครั้งเดียว คือ

```
Stress  = A(:,1)*6.895;
Strain = A(:,2);
```

จากนั้นเป็นการคำนวณค่า สำหรับ Young Modulus ก็คือความชันของเส้นกราฟในช่วง linear elastic ดังนั้นเราจึงควรเลือกข้อมูลที่มีค่า strain ต่ำๆ ส่วน Ultimate Tensile Strength คือค่าความเค้นที่สูงที่สุด

```
Young = (Stress(3)-Stress(2))/(Strain(3)-Strain(2));
fprintf('Youngs Moduls = %7.3f MPa \n',Young)
Youngs Moduls = 172375.000 MPa

U_Strangth = max(Stress);
fprintf('Ultimate Tensile Stress = %7.3f MPa \n',U_Strangth)
Ultimate Teasel Stress = 524.020 MPa
```

ค่า Elongation at fracture หาจาก strain ได้เป็น

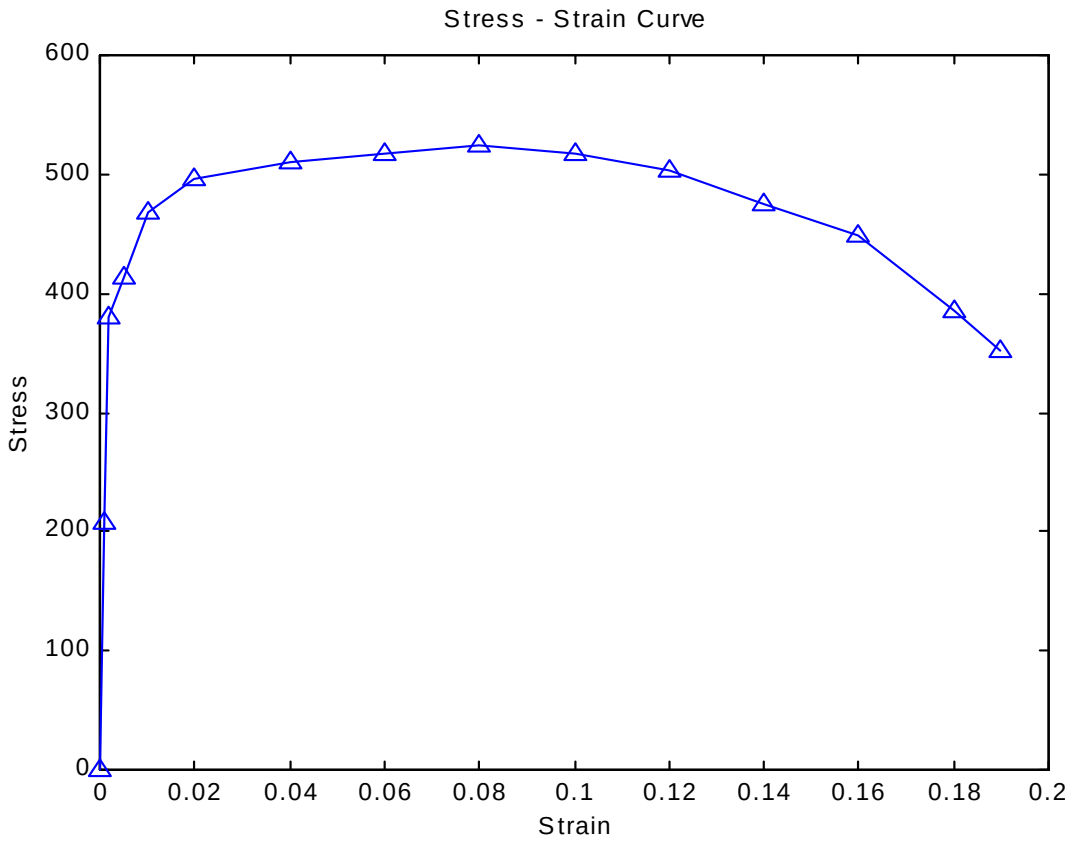
```
Elongation = (max(Strain)-min(Strain))*100;
fprintf('Percent Elongation at Fracture = %5.2f \n',Elongation)
Percent Elongation at fracture = 19.00
```

และสุดท้ายเป็นการเขียนกราฟ

```
plot(Strain,Stress,'^-');
xlabel('Strain')
ylabel('Stress')
title('Stress - Strain Curve')
```

ซึ่งก็จะได้กราฟตาม

รูป



บทที่ 4 การเขียน M-FILE

4.1 M-File

การเขียนโปรแกรมโดยใช้ MATLAB เป็นการช่วยให้การทำงานของวิศวกรง่ายขึ้นมากเมื่อเทียบกับการใช้ภาษาพื้นฐานเช่น C, FORTRAN, PASCAL,... ทั้งนี้เพราะว่า MATLAB มี function ทางคณิตศาสตร์และการเขียนกราฟรองรับไว้มากมายซึ่งช่วยลดเวลาการเขียนโปรแกรมลงไปได้อย่างมาก อย่างไรก็ตามในบางกรณีเรามีความจำเป็นต้องเขียน function ขึ้นมาใหม่เพื่อให้เหมาะสมกับงานของเรา

โปรแกรมที่เขียนโดย MATLAB จะ Save โดยใช้ extension เป็น " m " ซึ่งเรานิยมเรียกโปรแกรมที่เขียนโดยใช้ MATLAB ว่า M-file โดย M-file นี้จะแบ่งออกเป็น 2 ลักษณะคือเขียนในลักษณะของการบอกขั้นตอนหรือบอกบทการทำงานหรือที่นิยมเรียกว่า script file และอีกประเภทหนึ่งจะเขียนขึ้นในลักษณะของ function ซึ่งผู้ใช้สามารถรวบรวมเอา function ต่าง ๆ ของ MATLAB มารวมเข้าด้วยกันแล้วเขียนขึ้นเป็น function ใหม่ M-file ในลักษณะนี้เรียก function file

M-file ใน MATLAB จะเขียนเป็น plain text format ธรรมดา ดังนั้นเราอาจใช้ program เล็ก ๆ เขียนเช่น Notepad เขียนก็ได้ และการ save file จะ save เป็นชื่อ file ที่ต้องการโดยมี extension เป็น m สำหรับ MATLAB 5.x แล้ว จะมี MATLAB Editor/Debugger เพื่อใช้ในการเขียนโปรแกรมและแก้ไขโปรแกรม ส่วนการเรียกใช้ M-file นั้นก็เพียง พิมพ์ชื่อ file ที่ต้องการ โดยไม่ต้องมี extension เช่น ถ้าเราเขียน M-file ชื่อ **myfile.m** เวลาเรียกใช้ที่ command window ของ MATLAB ก็จะใช้คำสั่ง

```
» myfile
```

การเรียกใช้ M-file

การที่ MATLAB จะค้นหา M-file ที่สร้างขึ้นได้พบนั้น เราจะต้องกำหนดว่า directory ใด หรือ path ใดบ้าง ที่จะให้ MATLAB ค้นหาเพราะ MATLAB จะไม่ค้นหาทุก directory หรือ sub-directory ของทุก drive ที่มีในเครื่องขณะนั้น แต่ MATLAB จะค้นหาเฉพาะใน search path ที่กำหนดเท่านั้น ดังนั้นการเขียน M-file ขึ้นเองจะต้องเก็บ file นี้ให้อยู่ใน path ที่ MATLAB จะค้นหา สำหรับการปฏิบัติที่เหมาะสมเราควรจะเก็บ M-file ที่เราสร้างขึ้นไว้ใน directory ของเราต่างหากเพื่อไม่ให้ปะปนกับ M-file ของ MATLAB จากนั้นจึงเพิ่ม MATLAB search path ให้รวม directory ที่เราสร้างขึ้นใหม่เพื่อให้ MATLAB ทราบว่าต้องค้นหา program ที่เราเขียนขึ้นในที่ใด ขั้นตอนการเพิ่ม search path ทำได้หลายวิธีดังนี้

- การเพิ่ม search path ชั่วคราวสำหรับการทำงานแต่ละครั้งที่เรียกใช้ MATLAB ให้ใช้คำสั่ง **`addpath`** แล้วตามด้วยชื่อ directory ที่จะเพิ่มใน search path โดยการกำหนด directory ให้ใช้รูปแบบของ DOS
- การกำหนด search path อย่างถาวรใน **MATLAB 5.X** สามารถกระทำได้โดยง่ายโดยการ click ที่ File เลือก Set path... จากนั้นเพิ่ม path ที่ท่านต้องการ

4.2 Script Files & Function Files

ในการเขียน MATLAB เพื่อการคำนวณค่าต่างๆ นั้น เราสามารถป้อนค่าตัวแปรและคำสั่ง ซึ่งเป็น function ต่างๆ ที่สร้างขึ้นมาใน MATLAB เพื่อให้คำนวณและแสดงผลที่เราต้องการได้ โดยการป้อนคำสั่งนั้น อาจกระทำทีละบรรทัด ตามที่ตัวอย่างได้กล่าวมาแล้ว หรือ เราอาจจะเขียนเป็นชุดคำสั่งเหมือนกับการเขียน program computer ทั่ว ๆ ไป หรือที่เรียกกันว่าเขียน script files เพื่อสะดวกในการแก้ไขและมีขั้นตอนการคำนวณและแสดงผลเป็นไปตามที่เราต้องการ

นอกจาก MATLAB จะเขียนเป็นบท (script) แล้วยังสามารถจะสร้าง function file ที่ท่านต้องการเองได้อีกด้วย ข้อแตกต่างของ script file กับ function file คือ

- ใน Function file จะต้องมีการกำหนด input ที่จะเข้าสู่ file ที่แน่นอนและต้องมีการกำหนด output ที่ต้องการที่แน่นอน ส่วนใน script file ไม่จำเป็นต้องมี นั่นคือการเขียน function file จะต้องมี parameter
- ใน script file ค่าตัวแปรที่มีอยู่ในหน่วยความจำทุกค่าจะถูกส่งเข้าสู่ M-file และ MATLAB จะเก็บค่าทุกค่าที่เกิดขึ้นในระหว่างการคำนวณไว้ในหน่วยความจำแต่ function file จะใช้เฉพาะค่าที่กำหนดเข้าไปเท่านั้น และจะแสดงค่าเฉพาะค่าที่เราต้องการเป็น output ส่วนค่าอื่นที่เกิดขึ้นในระหว่างการหาค่า function นั้น MATLAB จะไม่เก็บไว้ในหน่วยความจำ กล่าวสั้นๆ ก็คือตัวแปรของ script file จะเป็น global variable ส่วนของ function file จะเป็น local variable

ซึ่งจะเห็นว่าการใช้ function file อาจจะดูยุ่งยากกว่า แต่มีข้อได้เปรียบหลายประการ คือ

- ประหยัดหน่วยความจำกว่า script file
- การตั้งชื่อ variable ใน M-file ไม่ต้องเกรงว่าจะเกิดการซ้ำซ้อนกับค่าที่ผู้ใช้กำลังใช้อยู่ เพราะ MATLAB จะเก็บในหน่วยความจำเฉพาะค่า input และ output
- ในหลายกรณี การใช้งาน function file จะสะดวกกว่า script file เพราะมีรูปแบบการกำหนดค่า parameter ต่าง ๆ ทั้งที่จะต้องให้เข้าไปและค่าที่ได้ออกมาเป็นรูปแบบที่แน่นอน

ไม่ว่าจะเป็น script file หรือ function file ก็ตาม การเขียนก็จะเขียนเป็น M-file เหมือนกัน และจะต้องบรรจุอยู่ใน search path ของ MATLAB เช่นเดียวกัน

การเขียน script file จะขึ้นต้นด้วยอะไรก็ได้ แต่สำหรับ function file จะต้องขึ้นต้นบรรทัดแรกด้วยรูปแบบนี้เสมอ

```
function  mypro(c)
```

โดยคำสั่ง function จะบอก MATLAB ว่า file นี้เป็น function file

c เป็น input หรือ parameter

mypro เป็นชื่อ function ของเรา **"และ file นี้จะต้อง save ในชื่อ mypro. m"** เท่านั้น

```
function  a =  myfunc(c,d)
```

โดยคำสั่ง function จะบอก MATLAB ว่า file นี้เป็น function file

a คือ ค่า output ที่ต้องการ อาจจะเป็น scalar หรือ matrix ก็ได้

c, d เป็น input หรือ parameter

myfunc เป็นชื่อ function ของเรา **"และ file นี้จะต้อง save ในชื่อ myfunc. m"** เท่านั้น

```
function[a, b, c]  = tiger(d, e, f, g)
```

จะเป็นการสร้าง function file ชื่อ tiger (Save ในชื่อ tiger.m) โดยมี input เป็น d, e, f และ g โดยมีค่าตัวแปร a, b และ c เป็น output

Function file จะต้องขึ้นบรรทัดแรกด้วยคำสั่ง function เสมอและชื่อของ M-file ต้อง Save ด้วยชื่อเดียวกับชื่อของ function นั้น

การใช้ Comment (%)

ในการเขียนโปรแกรมนั้นในหลาย ๆ กรณีจำเป็นต้องมีการเขียน comment หรือข้อช่วยบันทึกความจำลงไปด้วยเพื่อสะดวกในการแก้ไขหรือเป็นประโยชน์แก่ผู้ใช้ **เครื่องหมาย %** เป็นเครื่องหมาย comment ซึ่ง MATLAB จะไม่สนใจที่จะทำคำสั่งหรือข้อความต่าง ๆ ที่อยู่หลังเครื่องหมายนี้ ข้อสำคัญอีกประการหนึ่งสำหรับเครื่องหมาย % นี้ก็คือสามารถใช้เป็น **help** ของ M-file นั้นได้ด้วย นั่นก็คือ

- สำหรับ script file ข้อความที่อยู่หลัง % ทั้งหมดนับจากบรรทัดแรกจนกระทั่งถึงบรรทัดที่ไม่มีเครื่องหมายนี้จะปรากฏขึ้น ถ้าหากมีการพิมพ์ help แล้วต่อด้วยชื่อ file นั้น
- สำหรับ function file ข้อความที่อยู่หลัง % ทั้งหมดนับจากบรรทัดที่ต่อจากคำสั่ง function จนกระทั่งถึงบรรทัดที่ไม่มีเครื่องหมายนี้จะปรากฏขึ้นหากมีการพิมพ์ help แล้วต่อด้วยชื่อ file นั้น

ซึ่งรายละเอียดจะสามารถดูได้จากตัวอย่างต่อไปนี้

ตัวอย่างเช่นหากเราต้องการเขียน M-file เพื่อใช้หาค่าด้านตรงข้ามมุมฉากโดยกำหนดค่าด้านประชิดมุมสองด้านมาใช้

- ถ้าเขียน script file อาจเขียนเป็น

```
% 'Pythagoras' Theorem for Me
a     =     input ('Please input the first side');
b     =     input ('Please input the second side');
c     =     sqrt (a^2 + b^2)
```

หาก save file นี้ด้วยชื่อ pyta.m เมื่อเรียกใช้

```
pyta
Please input the first side  3
Please input the second side  4
c  =
      5

help pyta
      Pythagoras' Theorem for Me
```

- ถ้าเขียนเป็น function file อาจเขียนเป็น

```
function  c = pyta (a, b)
%  Pythagoras'  Theorem for Me
%  format  is c=pyta(a,b)
%  where a and b are sides of the rectangle
%  and c^2 =a^2+b^2
c = sqrt(a^2 + b^2)
```

และเมื่อ Save file ต้องใช้ชื่อ pyta.m ในกรณีเรียกใช้

```
c = pyta(3, 4)
c  =
      5
```

หรือ

```
Pyta (3,4)
ans  =
         5
```

หรือ

```
      x= Pyta (4, 3)
x  =
       5
```

และถ้าใช้

```
help pyta
```

จะได้

```
'Pythagoras' Theorem for Me
format  is c=pyta(a,b)
where a and b are sides of the rectangle
and c^2 =a^2+b^2
```

อย่างไรก็ดีสำหรับการจะเลือกใช้ M-file แบบใด ขึ้นอยู่กับความต้องการ และความเหมาะสมกับการทำงานของผู้ใช้

4.3 คำสั่งที่ควบคุมขั้นตอนการทำงานของ M-File

เช่นเดียวกับการเขียน program computer ทั่วไป MATLAB มีคำสั่งที่ใช้ในการควบคุมขั้นตอนการทำงานของ program เพื่อสะดวกในการทำงาน เช่นเดียวกับใน C, FORTRAN หรือ BASIC แต่ข้อแตกต่างจากการใช้ภาษาพื้นฐานเหล่านั้น คือ

- ใน MATLAB ไม่มี line number ดังนั้นการเขียนโปรแกรมหากต้องการข้ามชุดคำสั่งใดอาจต้องใช้ข้อความทางตรรกเข้าช่วยหรือใช้วิธีการใช้ subroutine หรือ function เข้าช่วย
- ใน MATLAB ไม่ต้องมีการบอกว่าตัวแปรใดเป็น string, integer หรือถ้าเป็นเลขทศนิยมก็ไม่จำเป็นต้องบอกว่าจะมี precision แบบใดยกเว้นว่าต้องการกำหนดเอง
- ไม่จำเป็นต้องจอง array หรือจอง dimension ของตัวแปร
- สามารถเรียกใช้ function ต่าง ๆ ที่มีอยู่ใน MATLAB หรือที่สร้างขึ้นเองได้ตลอดเวลา หรือ ไม่จำเป็นต้องเขียน subroutine ขึ้นมาใหม่
- MATLAB ทำงานได้โดยไม่ต้อง compile ก่อน ดังนั้น โดยทั่วไป MATLAB จะหยุดทำงานทันทีเมื่อพบ error

คำสั่งที่ใช้ควบคุมการทำงานหลัก ๆ มีดังนี้

- **For loop** Repetitive Control Structure
- **If - elseif - else - end** Conditional Control Structure
- **While loop** Repetitive Control Structure

❶ For Loop

สำหรับ For loop นั้นก็จะเหมือนกับ For loop (หรือ Do loop) ของภาษาอื่น ๆ โดยมีโครงสร้างเป็น

```
for   m = a:b
      ชุดคำสั่งหรือการคำนวณ
end
```

โดยคำสั่งนี้จะกำหนดให้ MATLAB ทำเป็นวงรอบ (loop) โดยเริ่มจากค่า m = a แล้วทำบรรทัดต่อมา คือ คำนวณชุดคำสั่งไม่ว่าจะมีชุดคำสั่งเท่าใดก็ตาม จากนั้นเมื่อพบคำสั่ง end แล้ว MATLAB ก็จะกลับไปเริ่มต้นที่คำสั่ง for อีก แล้วทำงานต่อโดยใช้ค่า m = a + 1 ไปเรื่อย และ จะเลิกทำงานเมื่อคำนวณค่าสุดท้ายที่ m = b ซึ่ง ในกรณีการใช้คำสั่งนี้ a และ b จะเป็นจำนวนเต็ม หรือหากใช้

`for m = a:b:c` จะเป็นการทำค่า 1 เริ่มจาก a ถึง c โดยเพิ่มขึ้นครั้งละ b

สำหรับผู้ที่ใช้ภาษา FORTRAN อยู่เป็นประจำอาจนิยมที่จะทำ loop ด้วยตัวแปร i หรือ j เพราะกำหนดเป็น integer ของ FORTRAN อย่างไรก็ตามใน MATLAB ไม่มีความจำเป็นและไม่แนะนำให้ใช้เพราะค่าทั้งสอง MATLAB ได้สำรองไว้สำหรับ imaginary number $\sqrt{-1}$

➢ ***การใช้*** For Loop ***ซ้อนกัน***

ในหลายกรณีเรามีความจำเป็นต้องใช้ for loop ซ้อนกัน ซึ่งก็จะสามารถกระทำได้ เช่นตัวอย่างต่อไปนี้

```
for   k =  1 : 5
   for i  =  1 : 5
         p(k,i)  =  k * i ;
   end
end
```

โปรแกรมนี้จะสร้าง matrix p มีขนาด 5x5 โดยแต่ละ element จะมีค่าเท่ากับผลคูณของตำแหน่ง row และ column จากโปรแกรมข้างบนนี้ จะเริ่มเมื่อ k = 1 จากนั้นจะเข้า loop ที่สองซึ่งจะได้ค่า i = 1 ดังนั้น P(1,1) = 1*1=1 และเมื่อพบคำสั่ง end เครื่องจะกลับไปที่จุดเริ่มต้นของ loop ที่ 2 ดังนั้น เมื่อถึงจุดนี้ k ยังคงเป็น 1 แต่ i = 2 ดังนั้น P(1,2) = 1*2=2 และเครื่องจะทำงานต่อไปจนกระทั่ง i = 5 ซึ่งครบคำสั่ง for ของ loop ที่สอง จากนั้น เมื่อพบคำสั่ง end ของ loop ที่สองเครื่องจะทำงานในบรรทัดต่อไป แต่เมื่อพบคำสั่ง end ของ loop แรกแล้ว เครื่องจะกลับไปเริ่มต้นใหม่ โดยใช้ค่า k = 2 แล้วจะวกกลับมาเข้า loop ที่สองอีกครั้งหนึ่ง ซึ่งวนค่า i = 1, 2, 3, 4, 5 ไปเรื่อย ๆ และ loop นี้จะหยุดการทำงานก็ต่อเมื่อจำนวนค่า k = 5 และ l = 5

ใน MATLAB 5.x เนื่องจากเราเขียน M-file ใน Editer ของ MATLAB เอง ดังนั้น เมื่อมี loop ซ้อนกันหลาย loop MATLAB จะทำการจัดย่อหน้าของคำสั่งให้เอง เพื่อความสะดวกในการแก้ไข โดยคำสั่งใน loop เดียวกันจะย่อหน้าเท่ากัน และคำสั่ง end จะตรงกับคำสั่ง for ของ loop นั้น ซึ่งจะเป็นเช่นเดียวกันกับคำสั่งในการวน loop อื่น ๆ

❷ If Statement

สำหรับการใช้ คำสั่ง If มีรูปแบบดังนี้

```
if     เงื่อนไข
       ชุดคำสั่ง
end
```

โดยถ้าเงื่อนไขเป็นจริง MATLAB จะคำนวณชุดคำสั่ง แต่ถ้าไม่เป็นจริง MATLAB จะข้ามไปทำคำสั่งในบรรทัดที่ต่อจาก end ต่อไป ในเงื่อนไขหรือ condition จะต้องมีค่าเป็นจริงหรือเท็จ เท่านั้น เครื่องหมายที่ใช้เปรียบเทียบ ส่วนใหญ่เป็นดังนี้

ความหมาย	สัญลักษณ์คณิตศาสตร์	MATLAB
เท่ากับ	=	= =
ไม่เท่ากับ	≠	~ =
ความหมาย	สัญลักษณ์คณิตศาสตร์	MATLAB
มากกว่า	>	>
น้อยกว่า	<	<
มากกว่าหรือเท่ากับ	≥	= >
น้อยกว่าหรือเท่ากับ	≤	= <
และ	AND	&
หรือ	OR	|

ตัวอย่างเช่น

```
if    x == 0
      a = 5
end
```

นั่นคือ ถ้า x = 0 แล้วจะได้ว่า a = 5 แต่ถ้า x ไม่เท่ากับศูนย์ จะไม่มีการกำหนดให้ค่า a = 5

➢ If-Elseif-Else

การใช้ else และ elseif นั้นมีรูปแบบดังนี้

```
if        เงื่อนไขที่ 1
          ชุดคำสั่งที่ 1
elseif    เงื่อนไขที่ 2
          ชุดคำสั่งที่ 2
elseif    เงื่อนไขที่ 3
          ชุดคำสั่งที่ 4
.....
else      เงื่อนไขที่ n
          ชุดคำสั่งที่ n
End
```

ในกรณีนี้ถ้าเงื่อนไขที่ 1 เป็นจริง MATLAB จะทำชุดคำสั่งที่ 1 แล้วมาที่ end แต่ถ้าไม่เป็นจริง MATLAB จะพิจารณาเงื่อนไขที่ 2 ถ้าเป็นจริง MATLAB จะทำชุดคำสั่งที่ 2 แล้วมาที่ end แต่ถ้ายังไม่เป็นจริงอีก MATLAB จะพิจารณาเงื่อนไขที่ 3 ถ้าเป็นจริง MATLAB จะทำชุดคำสั่งที่ 3 แล้วไปที่ end ทำเช่นนี้

ไปเรื่อยๆ จนกระทั่งถึงเงื่อนไขที่ n ถ้าไม่มีเงื่อนไขใดเลยที่เป็นจริง MATLAB ก็จะไม่ทำชุดคำสั่งใดแล้วมาที่ end เลย

❸ While - Loop

สำหรับ while loop นั้นก็จะคล้าย ๆ กับ for loop จะต่างกันที่ while loop นี้จะไม่กำหนดจำนวนรอบเหมือนกับ for loop แต่จะเป็นการวน loop ไปเรื่อย ๆ ตราบเท่าที่เงื่อนไขที่ให้ยังเป็นจริงอยู่

รูปแบบของ while-loop คือ

```
while   เงื่อนไข
        ชุดคำสั่ง
end
```

โดย loop นี้จะดำเนินไปเรื่อย ๆ จนกว่าเงื่อนไขนี้จะเป็นเท็จ เมื่อเงื่อนไขเป็นเท็จ MATLAB จะมาที่คำสั่ง end แล้วไปทำคำสั่งบรรทัดต่อไป

เช่น

```
a = 0
while a < 5
      a = a + 1
      b = 5 + a
end
```

เริ่มต้น a = 0 เมื่อเข้าสู่ loop while เงื่อนไข a < 5 จะเป็นจริง ดังนั้น MATLAB จะคำนวณค่า a = 0 + 1 = 1 และ b = 5 + 1 = 6 เมื่อถึง end แล้ว MATLAB จะกลับไปที่ while อีกครั้งหนึ่ง ในขณะนี้ a = 1 ดังนั้น เงื่อนไข a < 5 ยังคงเป็นจริงอยู่ ดังนั้น MATLAB ก็จะคำนวณชุดคำสั่งต่อไปอีก ซึ่งจะทำเป็นรอบไปเรื่อย ๆ และจะหยุดเมื่อ a มีค่ามากกว่า 5

ข้อควรระวัง ในการใช้คำสั่งเพื่อให้เกิด loop นั้นต้องตรวจสอบให้ดีไม่เช่นนั้น โปรแกรมอาจจะอยู่ในสภาพ "ติด loop" คือ วนทำงานอยู่ใน loop นั้นไม่สามารถออกมาทำงานในชุดคำสั่งอื่นได้ สภาพการเช่นนี้สังเกตได้จากเครื่องจะใช้เวลาในการคำนวณนานเกินความจำเป็นมาก และโปรแกรมไม่มีทีท่าว่าจะดำเนินต่อไปได้ การแก้ไขให้กด Ctrl-C เป็นยกเลิกชุดคำสั่ง MATLAB ทั้งหมด

❹ คำสั่ง ยกเลิก Loop

ใน program เอง หากว่า ต้องการยกเลิก loop ก็สามารถทำได้โดยใช้คำสั่ง

```
Break
```

เมื่อ MATLAB พบคำสั่งนี้จะออกจาก loop ที่กำลังทำอยู่ทันที แล้วดำเนินการคำนวณชุดคำสั่งหลังจาก end ต่อไปเช่น

```
for k =1:5
      y = k^2
      if  y = = 9
          break
      end
      x = 5/(y-9)
end
```

ในตัวอย่างนี้ MATLAB จะเริ่มเข้า loop และคำนวณหาค่า y ก่อน ถ้าหากว่า y ไม่เท่ากับ 9 MATLAB ก็จะทำงานต่อไป เพื่อหาค่า x แต่ว่าเมื่อใดก็ตาม y = 9 ทำให้เงื่อนไขเป็นจริง MATLAB จะทำงานในชุดคำสั่งของ If นั่นคือสั่งให้หยุด (break) ดังนั้นถึงจุดนี้ MATLAB จะข้ามมาทำบรรทัดต่อจาก end ของ loop for เลย ซึ่งเป็นการออกจาก loop มานั่นเอง

สำหรับคำสั่ง control flow ที่เพิ่มขึ้นของ MATLAB 5.X คือคำสั่ง **switch - case** ซึ่งเป็นคำสั่งที่ช่วยให้การใช้งานกรณีที่ตัวแปรมีโอกาสมีค่าได้หลายค่าและค่าแต่ละค่าจะมีคำสั่งให้ทำงานต่างๆกันออกไป โดยมีโครงสร้างต่อไปนี้

```
switch    ตัวแปร
case   ค่าตัวแปร กรณีที่ 1
        ชุดคำสั่งที่ 1
case   ค่าตัวแปร กรณีที่ 2
        ชุดคำสั่งที่ 2
case   ค่าตัวแปร กรณีที่ 3
        ชุดคำสั่งที่ 3
        ..............
otherwise
        ชุดคำสั่งที่ n
end
```

ลักษณะของการทำงานคือ เมื่อเรากำหนดให้พิจารณาตัวแปรตามคำสั่ง switch จากนั้น MATLAB จะพิจารณาว่าค่าของตัวแปรที่กำหนดนั้นเข้ากับกรณีใด ก็จะทำชุดคำสั่งของกรณีนั้น พิจารณาตัวอย่างต่อไปนี้

```
x=input('กรุณาเลือกคำตอบจากข้อ 1-4 \n');
switch x
case 1
   fprintf('ท่านได้เลือกข้อ 1 \n')
case 2
   fprintf('ท่านได้เลือกข้อ 2 \n')
case 3
   fprintf('ท่านได้เลือกข้อ 3 \n')
case 4
```

```
    fprintf('ท่านได้เลือกข้อ 4 \n')
otherwise
    fprintf('ท่านได้เลือกข้อที่ไม่มีกำหนดไว้\n')
end
```

จากการที่กำหนดค่า x แล้วเราสั่งให้พิจารณาค่า x จากนั้น MATLAB จะพิจารณาว่าค่า x ที่กำหนดให้จะเข้ากับกรณีใด ก็จะทำตามคำสั่งนั้น เช่นถ้า x = 1 เครื่องก็จะพิมพ์ว่า *ท่านได้เลือกข้อ 1* เป็นต้น

❺ คำสั่งควบคุมขั้นตอนอื่น ๆ

คำสั่งควบคุมขั้นตอนการทำงานอื่น ๆ ของ MATLAB มีดังนี้

pause	ให้หยุดการทำงานแล้วรอจนกระทั่งมีการกด keyboard อันใดอันหนึ่งจึงจะทำงานต่อ
pause(n)	หยุดการทำงานเป็นเวลา n วินาที
return	เลิกการทำงานของ M-file นั้นทั้งหมด แล้วกลับไปที่จุด ซึ่งเรียก M-file นี้มาใช้งาน

pause on หรือ **off** ให้รับหรือไม่รับคำสั่ง pause ที่จะมีตามมาใน M-file นี้

สำหรับการเขียน M-file นั้นอย่างที่กล่าวมาแล้วก็คือเหมือนกับการเขียนโปรแกรมภาษาอื่นๆ โดยทั่วไปเพียงแต่จะมีรูปแบบที่ง่ายกว่าและสามารถจะเรียกใช้ function ของ MATLAB ที่มีอยู่เดิมได้ตลอดเวลา

บทที่ 5 SOLUTION TO SYSTEM OF LINEAR EQUATIONS

5.1 ลักษณะของระบบสมการเชิงเส้น

สำหรับระบบสมการเชิงเส้นของค่าตัวแปร n ตัวซึ่งแต่ละตัวต้องมีกำลัง 1 และไม่อยู่ในรูปของ nonlinear function ใดๆ เช่น function sin, log เป็นต้น และการที่จะหาค่าได้นั้น อันดับแรกจะต้องมีสมการอิสระทั้งหมดเท่ากับจำนวนตัวแปร n ซึ่งระบบสมการเชิงเส้นสามารถเขียนในรูปทั่วไปได้เป็น

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ \ldots\ + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ \ldots\ + a_{2n}x_n &= b_2 \\ \ldots\ldots\ldots\ldots & \\ a_{n1}x_1 + a_{n2}x_2 + \ \ldots\ + a_{nn}x_n &= b_n \end{aligned}$$

หรือเขียนเป็นสมการ matrix ได้อยู่ในรูป

$$[A]\{x\} = \{b\}$$

โดย [A] เป็น [n x n] matrix ส่วน $\{x\}$ และ $\{b\}$ เป็น [n x 1] vector จากสมการนี้มีกรณีต่างๆ ให้พิจารณาดังนี้

- ถ้า $\{b\} \neq \{0\}$; $\{x\}$ จะหาค่าได้ ถ้า [A] มี inverse (หรือ $\det(A) \neq 0$)
- ถ้า $\{b\} = \{0\}$ จะหาค่า $\{x\}$ ที่เป็น non-trivial solution $(\{x\} \neq 0)$ ได้ก็ต่อเมื่อ $\det(A) = 0$ ซึ่งในกรณีนี้เรียก eigenvalue problems
- ถ้า $\{b\} \neq \{0\}$; และ [A] ไม่มี inverse (หรือ $\det(A) = 0$)จะไม่สามารถหาค่า $\{x\}$ ที่ unique ได้

สำหรับกรณีแรกถ้า $\{b\} \neq \{0\}$ และ[A] มี inverse จะได้

$$\begin{aligned} [A]^{-1}[A]\{x\} &= [A]^{-1}\{b\} \\ \{x\} &= [A]^{-1}\{b\} \end{aligned}$$

ใน MATLAB จะหาค่า variable x สำหรับ n-linear equations ได้โดยการกำหนดค่า [A] และ $\{b\}$ แล้วหาค่า x สำหรับคำตอบของระบบสมการเชิงเส้น สามารถหาค่าได้โดยไม่ยากนักเมื่อใช้ MATLAB ซึ่งแม้ว่าวิธีการแก้สมการประเภทนี้มีอยู่หลายวิธี แต่สำหรับ MATLAB แล้ววิธีที่นิยมใช้จะมีอยู่ 2 วิธีคือ

- การหา Inverse
- การใช้การหาร Matrix

ซึ่งการแก้สมการด้วยวิธีทั้งสองนั้น หากว่าระบบสมการเป็นระบบใหญ่การแก้สมการจะใช้ Numerical method ซึ่งในที่นี้จะไม่ขอกล่าวถึงรายละเอียดของวิธีการของการคำนวณแต่จะเป็นเพียงการแนะนำการใช้ MATLAB ช่วยหาคำตอบเท่านั้น

5.2 การหาคำตอบด้วยการหา Inverse

สำหรับการหาค่าคำตอบของสมการโดยวิธีการใช้ inverse นั้นดูเหมือนจะเป็นวิธีที่ยุ่งยากหากว่าจะต้องคำนวณด้วยตัวเอง แต่หากว่าใช้ MATLAB เป็นตัวช่วยในการคำนวณแล้วเราจะได้คำตอบที่รวดเร็ว สำหรับการหาค่า inverse ของ Matrix นั้น MATLAB เราจะใช้รูปแบบของการหาค่า x จากสมการ

$$[A]\{x\} = \{b\}$$

ด้วยคำสั่ง

```
x=inv(A)*b
```

ตัวอย่างเช่นต้องการหาค่า x จาก

$$\begin{aligned} 2x_1 + x_2 &= 1 \\ -x_1 + 3x_2 + 5x_3 &= 2 \\ 4x_1 - 6x_2 + 8x_3 &= 4 \end{aligned}$$

ดังนั้นใน MATLAB จะได้

```
A=[2,1,0;1,3,5;4,-6,8];
B=[1;2;4];
X=inv(A)*B
X =
   0.4833
   0.0333
   0.2833
```

และหากต้องการหาค่าความผิดพลาดของการคำนวณที่เกิดขึ้นอาจใช้

```
D=B-A*X
D =
  1.0e-015 *
   0.4441
        0
```

```
0
```

จะพบว่าความผิดพลาดเกิดขึ้นที่ค่า x_1 โดยจะมีค่าความผิดพลาด 0.4441×10^{-15}

ความจริงแล้วเราคงทราบกันดีว่าการหาค่า inverse ของ Matrix เป็นขั้นตอนที่ยุ่งยากและต้องใช้เวลามาก ไม่ว่าจะใช้วิธี analytical หรือ numerical ก็ตาม หากว่าเราต้องการจะหาค่าคำตอบของระบบสมการเชิงเส้นนี้เพียงอย่างเดียว โดยไม่ได้สนใจค่า inverse ของ Matrix อย่างจริงจังแล้ว MATLAB เองก็ไม่แนะนำให้ใช้วิธีนี้ แต่กลับแนะนำให้ใช้การหาร matrix หรือใช้ X = B/A หรืออาจใช้ X = A\B แทนเพราะจะให้ความถูกต้องและใช้เวลาในการคำนวณน้อยกว่ามากซึ่งรายละเอียดอยู่ในหัวข้อต่อไป

5.3 การหาคำตอบโดยการหาร Matrix

เป็นที่ทราบกันดีอยู่แล้วว่า matrix นั้นไม่มีคุณสมบัติของการหาร แต่ MATLAB ได้สร้างคุณสมบัตินี้ขึ้นมาเพื่อใช้หาคำตอบของสมการเชิงเส้นโดยเฉพาะ โดย MATLAB ใช้หลักการวิเคราะห์เชิงตัวเลขด้วยวิธี Gaussian Elimination ซึ่งจะเป็นการหาคำตอบโดยไม่จำเป็นต้องหา inverse ของ matrix ซึ่งเป็นวิธีการที่ประหยัดเวลาและได้ความถูกต้องมากกว่าการหา inverse ด้วยวิธีเชิงตัวเลขมาก

นอกเหนือจากนี้หากว่า matrix มีสภาพที่เรียกว่ามี ill-condition หรือสภาพที่ matrix เกือบเป็น singular matrix (คือ matrix ที่มี determinant ใกล้เคียงศูนย์มาก) จะทำให้การหาค่า inverse มีความผิดพลาดมากขึ้นไปอีก เพราะต้องคำนวณตัวเลขที่มี floating point สูงหลายครั้ง แต่สำหรับ Gaussian Elimination Method จะมีวิธีการที่จะตรวจสอบ ill-condition เหล่านี้ทำให้การหาค่าคำตอบมีความแม่นยำกว่ามาก

ดังนั้นจากระบบสมการเชิงเส้น

$$[A]\{x\} = \{b\}$$

หากเราต้องการหาคำตอบของสมการโดยใช้ MATLAB โดยวิธีการหาร matrix จะใช้

```
x=A\b
```

ตัวอย่างเช่นสมการที่ผ่านมาจะหาคำตอบได้โดย

```
A=[2,1,0;1,3,5;4,-6,8];
B=[1;2;4];
X=A\B
X =
   0.4833
   0.0333
   0.2833
```

และความผิดพลาด

```
D=B-A*X
D =
   0
```

0

0

จะเห็นว่าไม่เกิดความผิดพลาดขึ้นเลย

แม้ว่าตัวอย่างที่ยกมาโดยใช้การแก้ปัญหาด้วยวิธีทั้งสอง ปรากฏว่าไม่มีความแตกต่างกันมาก และเนื่องจากทุกวันนี้เครื่องคอมพิวเตอร์มีประสิทธิภาพสูงขึ้นมากจนทำให้อาจไม่เห็นถึงความแตกต่างของการคำนวณเวลามากนัก อย่างไรก็ตามการจะเลือกวิธีใดเพื่อไปใช้งานขึ้นกับความเหมาะสมกับงานนั้นๆเป็นหลัก

5.4 Eigenvalues และ Eigenvectors

พิจารณาสมการ

$$Ax = \lambda x$$

เมื่อ A เป็น square matrix ของค่าคงที่

x เป็น column vector

และ λ เป็น scalar

สมการนี้มี solution ของ x คือ $\{x\} = \{0\}$ คือทุก element ของ x มีค่าเท่ากับศูนย์ แต่ solution นี้เรียกว่า trivial solution และเป็นคำตอบที่นำไปใช้ประโยชน์อะไรไม่ได้ สำหรับค่า scalar λ ที่ทำให้เกิด non-trivial solution ของสมการจะเรียก eigenvalues ของ matrix A และค่า x ที่ได้จาก eigenvalue แต่ละค่าจะเรียกว่า eigenvector และปัญหานี้เรียก eigenvalue problem สำหรับสมการข้างบนนี้สามารถเขียนใหม่ได้เป็น

$$(A - \lambda I)x = \{0\}$$

จะเห็นว่าถ้า $\{x\} \neq \{0\}$ matrix $(A - \lambda I)$ จะต้องไม่มี inverse ตัวอย่างเช่นหาก matrix C และ d มีสมการเป็น

$$[C]\{d\} = \{0\}$$

ถ้า C มี inverse จะได้ $[C]^{-1}[C]\{d\} = [C]^{-1}\{0\}$

$$\{d\} = \{0\}$$

ดังนั้นถ้า $\{d\} \neq \{0\}$ **C** จะต้องไม่มี inverse

ดังนั้นในกรณีที่เรากำลังพิจารณาอยู่คือ $(A-\lambda I)$ จะต้องไม่มี inverse ข้อจำกัดที่จำเป็นและเพียงพอของ matrix $(A-\lambda I)$ นี้ก็คือ determinant ของ matrix นี้ต้องเท่ากับ 0 นั่นคือ

$$|A-\lambda I|=0$$

สมการนี้ เรียกว่า characteristic equation ของ matrix A เนื่องจากสมการนี้เป็นสมการที่ไม่ unique นั่นคือมีค่า x มากกว่า 1 ค่าที่จะทำให้สมการ $(A-\lambda I)x=\{0\}$ เป็นจริงได้ ดังนั้นการหาค่า eigenvector x จึงนิยมที่จะหาเป็น orthogonal matrix โดยจากนิยามว่าถ้า Q เป็น orthogonal matrix จะได้

$$QQ^T=I \quad \text{หรือ} \quad Q^T=Q^{-1}$$

สำหรับ eigenvector นอกจากจะเป็น orthogonal แล้ว ยังมีคุณสมบัติ normalized ด้วยคือ eigenvector แต่ละค่าของ eigenvalue เมื่อทำ dot product กันจะมีค่าเป็นศูนย์ ดังนั้นจึงนิยมเรียกว่า eigenvector ว่าเป็น orthonormal vector สำหรับ MATLAB การหา eigenvalue problem จะใช้ function ต่อไปนี้คือ

eig(A) คำนวณ column vector ที่บรรจุ eigenvalue ของ matrix A

[Q, b]= eig(A)

จะให้ค่า square matrix Q ซึ่งแต่ละ column จะแทน eigenvector ของ A และ square matrix b ที่บรรจุ eigenvalue ของ A อยู่ในแนว diagonal ส่วนค่าอื่นเป็น 0

ตัวอย่างเช่น

```
A=[1 4;8 9];
eig(A)
ans =
  -1.9282
  11.9282
```

หรือ

```
[B,C]=eig(A)
B =
   -0.8069   -0.3437
    0.5907   -0.9391
C =
   -1.9282  0
      0     11.9282
```

การแก้ปัญหาทางวิศวกรรม

การหา eigenvalue นี้เป็นเรื่องสำคัญในการคำนวณทางวิศวกรรมหลายประการเช่นการหา Principal Stress ของระบบที่อยู่ภายใต้แรงกระทำ การหา Natural Frequency ของระบบ N-DOF เมื่ออยู่ภายใต้สภาพการสั่นอย่างอิสระเป็นต้น

➢ Principal Stress

ตัวอย่างต่อไปนี้เป็นการหา Principal Stress และ Principal Direction ของระบบที่ตกอยู่ภายใต้ stress system ดังต่อไปนี้

$$\sigma_{xx} = 120, \quad \sigma_{yy} = 55, \quad \sigma_{zz} = -85, \quad \sigma_{xy} = -55, \quad \sigma_{yz} = 33, \quad \sigma_{zx} = -75 \quad \text{MPa}$$

ลำดับแรกจะให้ข้อมูลของ stress tensor ขนาด 3x3

```
S=[120 -55 -75;-55 55 33;-75 33 -85]
S =
  120   -55   -75
  -55    55    33
  -75    33   -85
```

จากนั้นหาค่า eigenvalue และ eigenvector

```
[Direc SP]=eig(S)
Direc =
  -0.4654   -0.8372    0.2872
  -0.8836    0.4587   -0.0944
    0.0527    0.2977    0.9532

SP =
   24.0644         0         0
         0  176.7995         0
         0         0 -110.8640
```

เนื่องจาก eigenvalue ยังคงเป็น matrix อยู่ เราอาจเปลี่ยนให้เป็น vector โดยใช้

```
S=diag(SP)
S =
   24.0644
  176.7995
 -110.8640
```

สำหรับ Maximum Shearing Stress อาจหาได้จาก

```
MaxShear=(max(S)-min(S))/2
MaxShear =
  143.8318
```

➢ Natural Frequency

ส่วนการแก้ปัญหาทางด้าน Vibrations สามารถใช้ Egienvalue Problem หาค่า Natural Frequency และ Mode Shapes ของระบบ สำหรับระบบที่มีการสั่นอย่างอิสระและไม่มี damp จะมีรูปสมการเป็น

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

เมื่อ x เป็น column vector ของ coordinate ที่ใช้กำหนดระบบ และ $\ddot{x} = \frac{d^2x}{dt^2}$ ส่วน M เป็น mass matrix และ K เป็น stiffness matrix สำหรับคำตอบของสมการอาจสมมุติให้อยู่ในรูปของ harmonic หรือ

$$x = Ae^{i\omega t}$$

เมื่อ A คือขนาดและ ω คือความถี่ของการสั่น ซึ่งเราจะได้ $\ddot{x} = -\omega^2 Ae^{i\omega t}$ แทนค่าลงในสมการการเคลื่อนไหวจะได้

$$\left[K - \omega^2 M\right]x = 0$$

หรือ

$$\left[AM - \lambda\right]x = 0$$

เมื่อ $A = K^{-1}$ influence coefficient matrix และ $\lambda = \frac{1}{\omega^2}$ ซึ่งเราจะพบว่าเราได้ลักษณะสมการเป็น eigenvalue problem ซึ่งลักษณะการเคลื่อนไหว x จะไม่สามารถหาค่าที่แน่นอนได้เพราะสมการนี้ไม่มี unique solution อย่างไรก็ตามเราสามารถหารูปแบบของการสั่นหรือ mode shape ได้ สิ่งที่เราสนใจจากปัญหานี้ก็คือ natural frequency และ mode shape ของการสั่นนั่นเอง ตัวอย่างต่อไปนี้สมมุติว่ามีระบบการสั่นอย่างอิสระ 3-DOF ซึ่งมี Mass และ Stiffness Matrix เป็นดังนี้

$$M = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \qquad K = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 & -k_3 \\ 0 & -k_3 & 0 \end{bmatrix}$$

เราสามารถใช้ MATLAB หา Natural Frequency และ Mode Shape ได้โดยการเขียน function file ขึ้นมาในที่นี้ให้ชื่อ threeDOF.m โดยเรากำหนดค่า $M = [m_1\ m_2\ m_3]$ และ $K = [k_1\ k_2\ k_3]$ เข้าไปเป็น vector เพื่อความสะดวก (ถ้าเป็นกรณีอื่นๆ เราอาจให้ค่า matrix M และ K เลยก็ได้)

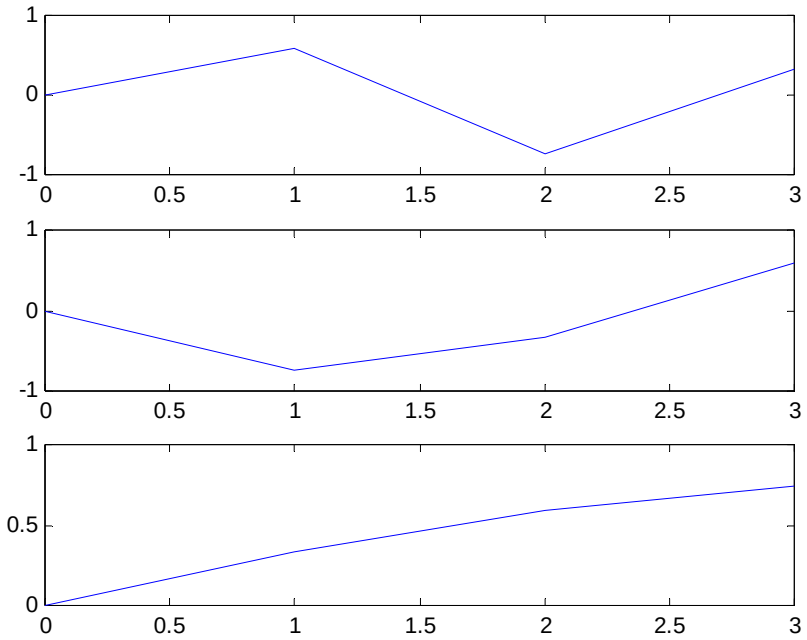
```
function [A,B]=threedof(M,K)
m=zeros(3,3);
k=zeros(3,3);
for j=1:3;
   m(j,j)=M(j);
end
k(1,1)=K(1)+K(2);
k(2,2)=K(2)+K(3);
```

```
k(3,3)=K(3);k(1,2)=-K(2);k(2,3)=-K(3);
k(2,1)=-K(2);k(3,2)=-K(3);

% หา A - Influence Coefficient Matrix โดย A=inv(K)
invk =inv(k);

% หา D - Dynamics Matrix โดย D = inv(K)*M
D = invk*m;
% หา  Natural Frequency และ Mode Shapes
[A B]=eig(D);
B=diag(B);
B=sqrt(abs(1./B)); % λ = 1/ω^2
% Plot Mode Shapes
X=[0 1 2 3]';
subplot(3,1,1);
   Y=[0;A(:,1)];
   plot(X,Y);
subplot(3,1,2);
   Y=[0;A(:,2)];
   plot(X,Y);
subplot(3,1,3);
   Y=[0;A(:,3)];
   plot(X,Y);
```

จากนั้นถ้าสมมุติให้ $m_1 = m_2 = m_3 = 1$ kg และ $k_1 = k_2 = k_3 = 10$ N/mm จะสามารถใช้ function threeDOF ได้โดย

```
M=[1 1 1];
K=[10 10 10];
[A B]=threedof(M,K)
A =
   0.5910   -0.7370    0.3280
  -0.7370   -0.3280    0.5910
   0.3280    0.5910    0.7370
B =
   5.6982
        3.9433
        1.4073
```

ซึ่งจะได้ fundamental frequency (ค่าความถี่ที่ต่ำสุด) เป็น 1.4073 rad/sec และได้ Mode Shape เป็น



นอกจากนี้ eigenvalue ยังมีบทบาทในการคำนวณทางด้านวิศวกรรมอีกหลายกรณี แต่ไม่ขอยกตัวอย่างในที่นี้

บทที่ 6 INTERPOLATION และ CURVE FITTING

6.1 แนะนำ Interpolation และ Curve Fitting

การทำงานทางด้านวิศวกรรม จำเป็นจะต้องใช้ข้อมูลที่ได้จากการทดลองเพื่อใช้ในการคำนวณต่าง ๆ แต่ค่าที่ได้จากการทดลองมักจะได้เป็นการกระจายของจุดข้อมูล ซึ่งอาจไม่เหมาะสมในการนำไปใช้ในการคำนวณ ดังนั้นจึงต้องมีวิธีการทางคณิตศาสตร์เพื่อเปลี่ยนข้อมูลที่มีลักษณะเป็นกลุ่มของจุดออกมาให้เป็น function ทางคณิตศาสตร์ที่มีความต่อเนื่อง การเปลี่ยนข้อมูลเป็น function สามารถกระทำได้โดยหลักใหญ่ๆ 2 กรณีคือ

1. Interpolation
2. Curve Fitting

ข้อแตกต่างของทั้ง 2 วิธีคือ สำหรับ interpolation นั้น function จะผ่านทุกจุดของข้อมูล แต่ curve fitting ไม่จำเป็นต้องเป็นเช่นนั้น นั่นคือค่า function อาจจะผ่านหรือไม่ผ่านจุดข้อมูลทุกจุดก็ได้แต่ควรจะเป็น function ที่ต่อเนื่องและมีค่าใกล้เคียงกับทุกจุดมากที่สุด หรืออาจเป็นเส้น curve ที่เหมาะสมที่สุดกับกลุ่มข้อมูลก็ได้ สำหรับวิธีการวิเคราะห์เชิงตัวเลขที่ใช้ทั้ง 2 กรณีนั้นมีหลายวิธี แต่ MATLAB ได้ใช้วิธีการต่อไปนี้

1. Interpolation วิธีที่ MATLAB ใช้เป็นหลักคือ

- Polynomial interpolation - คือการเชื่อมต่อข้อมูล 2 จุดที่อยู่ติดกันด้วยเส้นโค้งหรือเส้นตรงตามแต่ที่กำหนด ในกรณีที่กำหนดให้เป็นเส้นตรงจะเรียก linear interpolation
- Cubic-Spline interpolation - จะเป็น curve ที่ได้จาก polynomial degree 3 ที่เชื่อมต่อระหว่างจุด 2 จุด ที่อยู่ติดกันและนอกจากนั้นแล้วยังให้ slope ของเส้นโค้งสองเส้นที่บรรจบกันที่จุดข้อมูลมีค่าเท่ากันหรือกล่าวอีกอย่างหนึ่งก็คือ curve จะราบเรียบที่จุดต่อของ curve ทั้งสองเส้นนั่นเอง

สรุปทั้งสองกรณีหากมีข้อมูล n จุด ทั้งสองวิธีจะสร้างเส้น curve ขึ้น n-1 เส้น โดย linear interpolation นั้นจะให้ curve แต่ละเส้นเป็นเส้นตรง และ cubic spline จะให้ curve แต่ละเส้นเป็น polynomial degree 3

2. Curve fitting วิธีการที่ MATLAB ใช้คือ Least-Square Curve Fitting โดยจะเป็นวิธีที่เลือก curve ที่เป็น polynomial ซึ่งมี degree ตามที่ผู้ใช้ต้องการ โดย curve จะผ่านกลุ่มข้อมูลที่ผลรวมของระยะห่างจาก

ข้อมูลแต่ละจุดห่างจากเส้น curve นี้น้อยที่สุด สำหรับ curve fitting นี้ MATLAB จะคำนวณออกมาเป็น polynomial เพื่อให้ผู้ใช้สามารถนำไปใช้ต่อไปได้

สำหรับรายละเอียดขั้นตอนของการวิเคราะห์เชิงตัวเลขทั้งสองแบบจะไม่ขอกล่าวในเอกสารนี้ และผู้ที่สนใจสามารถศึกษาได้จากหนังสือที่เกี่ยวข้องโดยตรง เช่น หนังสือทางด้าน Numerical Analysis อย่างไรก็ตาม เนื่องจากผู้เรียบเรียงไม่ได้อธิบายข้อดีหรือข้อเสียของวิธีการเชิงตัวเลขที่ MATLAB นำมาใช้ ผู้เรียบเรียงจึงหวังว่าผู้อ่านคงจะค้นคว้าหาข้อมูลเพิ่มเติมให้เข้าใจถึงหลักการและขั้นตอนของวิธีการต่างๆ อย่างแท้จริง เพื่อจะได้เลือกใช้งานได้อย่างเหมาะสมที่สุด

6.2 Interpolation

❶ Linear Interpolation

สำหรับ linear interpolation จะเป็นการเชื่อมต่อจุดข้อมูลที่อยู่ติดกันด้วยเส้นตรง ดังนั้นการหาค่าของข้อมูลที่อยู่ระหว่างจุดทั้งสองก็จะใช้สมการเชิงเส้นของเส้นตรงที่เชื่อมจุดนั้นเป็นตัวกำหนด เหมือนกับที่เราใช้ในการเปิดตารางต่างๆ ทางวิศวกรรม เช่น ตารางเทอร์โมไดนามิกส์ เป็นต้น

สำหรับคำสั่งที่ใช้กับ MATLAB จะใช้คำสั่ง **`interp1`**

`Interp1(x,y,s)` เป็นการหา linear interpolation ของชุดข้อมูล x และ y เมื่อ y = y(x) โดยจะให้คำตอบเป็นค่าของ y(s)โดย x และ y เป็น vector ขนาดเท่ากัน ในกรณีที่ s เป็น vector จะให้ค่าเป็น vector ที่มีขนาดเท่ากับ s

สำหรับข้อมูลที่ใช้ใน `interp1` ต้องมีการเรียงค่าของ x และ MATLAB แนะนำให้เรียงค่า x จากน้อยไปหามาก

ตัวอย่างเช่นในการทดสอบระยะทางที่วัตถุเคลื่อนที่ ที่เปลี่ยนไปตามเวลา ได้ดังนี้

เวลา (วินาที)	ระยะทาง (เมตร)
0	0
1	200
2	600
3	680
4	800

ถ้าต้องการหาระยะทางที่วัตถุเคลื่อนที่ไปได้ เมื่อเวลา t = 2.5 วินาที โดยใช้ linear interpolation จะมีขั้นตอนดังนี้

```
x = [0 1 2 3 4];          % x ควรอยู่ในลักษณะเรียงจากน้อยไปหามาก
y = [0, 200, 600, 680, 800];
y1 = interp1(x, y, 2.5)
y1 =
  640
```

หากต้องการหาระยะที่วัตถุเคลื่อนที่ไปได้เมื่อเวลา t = 1.5 วินาที และ 35 วินาที โดยใช้ linear interpolation จะมีขั้นตอนดังนี้ (สมมุติว่าใส่ข้อมูล x และ y แล้วจากข้างบน)

```
s = [1.5 3.5];
y2 = interp1(x, y, s);
y2 =
  400  740
```

นั่นคือที่ t = 15 วินาที จะได้ระยะทาง 400 เมตร และที่ t = 3.5 วินาที จะได้ระยะทาง 740 เมตร

ในกรณีที่ x และ y เป็น matrix ที่มีขนาดเท่ากัน MATLAB จะจับคู่ของข้อมูลในลักษณะ column ต่อ column ดังนั้นในการใส่ข้อมูลจะต้องให้ความระมัดระวังด้วย

❷ Cubic Interpolation

สำหรับ function cubic interpolation ซึ่งจะคล้ายกับ spline แต่ที่จุดต่อของ curve ไม่จำเป็นต้องราบเรียบ รูปแบบจะเป็น

`interp1(x,y,s,'cubic')` โดย x, y, s จะเหมือนกับที่ผ่านมา ส่วน string 'cubic' จะบอกให้ MATLAB ทราบว่าต้องการ cubic interpolation

จากตัวอย่างที่ผ่านมาจะได้ว่า

```
x = [0, 1, 2, 3, 4];
y = [0, 200, 600, 680, 800];
y1 = interp1(x, y, 2.5,'cubic')
y1 =
  657.5000
```

และหากจะพิจารณาที่เวลา 1.5 วินาที และ 3.5 วินาที จะได้

```
s = [1.5 3.5];
y2 =  interp1(x, y, s,'cubic')
y2 =
  407.5000  735.0000
```

ซึ่งจะเห็นว่าได้ค่าแตกต่างจากที่ได้ในลักษณะของ linear spline สำหรับการพิจารณาความเหมาะสมว่าค่าใดถูกต้องมากกว่ากันก็จะขึ้นอยู่กับว่า function ที่แท้จริงนั้นมีลักษณะเช่นไร

❸ Spline Interpolation

สำหรับ spline interpolation จะมีรูปแบบ function เป็น

`interp1(x,y,s,'spline')` ค่า x, y, s จะเหมือนกับที่ผ่านมา ส่วน string 'spline' จะบอกให้ MATLAB ทราบว่าต้องการใช้วิธี cubic spline interpolation

จากตัวอย่างที่ผ่านมาหากต้องการหาค่าที่ 2.5 วินาที โดยใช้ cubic spline จะได้

```
x = [0, 1, 2, 3, 4];
y = [0, 200, 600, 680, 800];
y1 = interp1(x, y, 2.5,'cubic')
y1 =
   657.5000
```

และหากต้องการทราบระยะทางของวัตถุที่เวลา 1.5 วินาที และ 3.5 วินาที โดยใช้ cubic spline interpolation อาจทำได้ดังนี้

```
s=[1.5 3.5];
y1 = interp1(x, y, s,'spline')
y1 =
   421.2500   698.7500
```

ซึ่งเราสามารถเปรียบเทียบค่าทั้งสามวิธีว่าแตกต่างกันเพียงใด สำหรับการเรียกใช้ spline นี้อาจจะใช้ได้อีกรูปแบบหนึ่งคือใช้

`spline(x,y,s)` ซึ่งจะให้ผลเหมือน interp1 (x, y, s, 'spline') เพียงแต่คำสั่ง spline รับแต่เฉพาะ x, y ที่เป็น vector เท่านั้น

`spline(x,y)` จะให้ 'piecewise polynomial' ที่ได้จาก interpolate ชุดข้อมูล (x, y) โดย polynomial จะอยู่ในรูป y = y(x) ซึ่งเรียก polynomial ที่ได้ว่ามีรูปเป็น pp-form polynomial และหากต้องการหาค่า function ที่จุดใดอาจใช้คำสั่ง ppvalue

`ppval(pp,s)` pp คือ pp-form polynomial, s คือจุดที่ต้องการหาค่า ซึ่งให้ผลเหมือนกับการใช้คำสั่ง spline(x, y, s)

สำหรับตัวอย่างของคำสั่งเหล่านี้ หากใช้ข้อมูลจากตัวอย่างที่ผ่านมาจะได้

```
spline(x,y,s)
ans =
   421.2500  698.7500
```

หรือ

```
p=spline(x,y);
u=ppval(p,s)
u =
  421.2500  698.7500
```

❹ การเปรียบเทียบวิธีการ Interpolation

หากต้องการเปรียบเทียบความเหมาะสมของวิธีการทั้งสามสมมุติว่าเราทราบค่าฟังก์ชั่นที่แท้จริงคือ

$$y = x \cdot \sin(x)$$

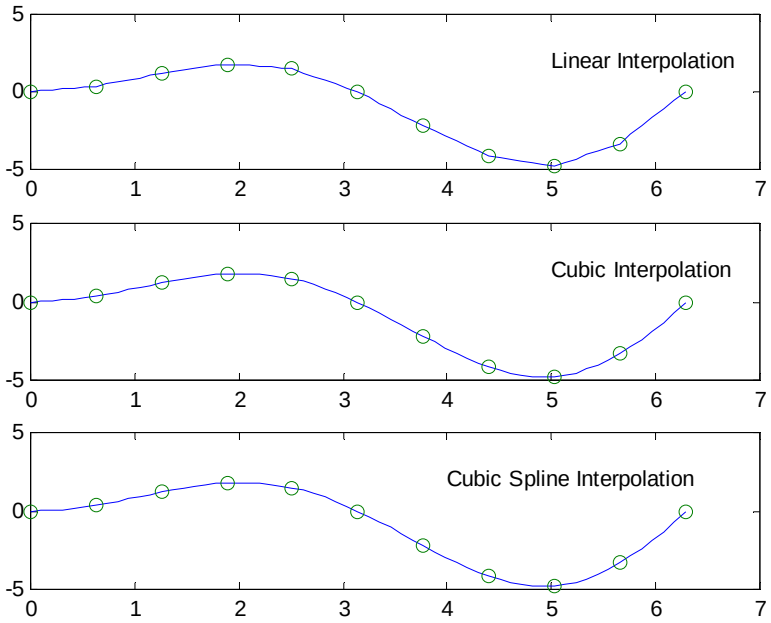
เราจะสร้างค่า x และ y มาเป็น vector ขนาด 1x 20 โดยมีระยะห่าง x เท่าๆกัน จากนั้นจะเปรียบเทียบความแม่นยำของวิธีการทั้งสาม โดยจะพิจารณาในช่วง 0 ถึง 2 π แล้วหาค่าจากการ interpolation ทั้งสามวิธีที x = 5 ว่าวิธีการใดจะได้ค่าใกล้เคียงที่สุด ซึ่งเมื่อเขียนเป็น M-file จะได้เป็น

```
%Interpolation Demo
x=0:pi/5:2*pi;
y=x.*sin(x);
s=0:pi/30:2*pi;    % Set interval for plot

% Linear
a=interp1(x,y,s);
a1=interp1(x,y,5);

% Cubic
b=interp1(x,y,s,'cubic');
b1=interp1(x,y,5,'cubic');

% Cubic Spline
c=interp1(x,y,s,'spline');
c1=interp1(x,y,5,'spline');

% Exact
d1=5*sin(5);
fprintf('Solution of x*sin(x) at x=5 by linear inter.= %f \n',a1)
fprintf('Solution of x*sin(x) at x=5 by cubic inter. = %f \n',b1)
fprintf('Solution of x*sin(x) at x=5 by spline inter. = %f \n',c1)
fprintf('Exact solution of function x*sin(x) at x=5 = %f \n',c1)

subplot(3,1,1)
plot(s,a,'-',x,y,'o')
text(5,2,'Linear Interpolation')
subplot(3,1,2)
plot(s,b,'-',x,y,'o')
text(5,2,'Cubic Interpolation')

subplot(3,1,3)
plot(s,c,'-',x,y,'o')
text(4,2,'Cubic Spline Interpolation')
```

ซึ่งผลได้เป็น

```
Solution of x*sin(x) at x=5 by linear inter. = -4.755283
Solution of x*sin(x) at x=5 by cubic inter. = -4.796262
Solution of x*sin(x) at x=5 by spline inter. = -4.795152
Exact solution of function x*sin(x) at x=5 = -4.795152
```

และกราฟที่ได้มีลักษณะเป็น



ซึ่งจากคำตอบและกราฟที่ได้คงจะพอมองเห็นว่าวิธีการวิธีใดให้ความแม่นยำในการคำนวณมากกว่ากัน อย่างไรก็ตามวิธีการใดจะเหมาะสมกับการใช้งานมากที่สุดก็ขึ้นอยู่กับว่าลักษณะ function ที่แท้จริงเป็นอย่างไรนั่นเอง

สำหรับ interpolation function ของ MATLAB นั้นมีอีกหลาย function รายละเอียดสามารถหาได้จาก หนังสือคู่มือ MATLAB หรือ ใช้ help ตามด้วยชื่อ function ซึ่งมีดังต่อไปนี้

interp2	สำหรับ 2-มิติ interpolation
interp#	สำหรับ interpolation # -มิติ ในขณะนี้ MATLAB ทำได้ถึง 6-มิติ
interpft	สำหรับ Fast Fourier Transform Interpolation Method
griddata	สำหรับ 2-มิติ interpolation

6.3 Polynomial Curve Fitting

Polynomial curve fitting คือการหาสมการ polynomial degree n ที่เหมาะสมกับชุดกลุ่มข้อมูลที่สุด แม้ว่ากราฟนี้จะไม่ผ่านทุกจุดของข้อมูลแต่กราฟจะเป็นกราฟเพียงเส้นเดียว นั่นคือมีสมการกราฟเพียงสมการเดียวตลอดช่วงที่กำหนดให้ ซึ่ง polynomial จะมีรูปแบบเป็น

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$$

โดยค่าสัมประสิทธิ์ a_i จะต้องเลือกให้เหมาะสมกับชุดข้อมูลที่สุด

สำหรับ Least square curve fitting เป็นการเลือก polynomial ใน degree ที่กำหนด เพื่อที่จะผ่านกลุ่มข้อมูลที่กำหนด สำหรับหลักการที่ใช้ในการเลือก a_i นั้นจะเลือกค่าที่ให้กราฟที่ได้มีลักษณะที่มีผลรวมกำลังสองของระยะทางจากจุดต่างๆ ห่างจากเส้นกราฟน้อยที่สุด สำหรับรายละเอียดของขั้นตอนการเลือกสัมประสิทธิ์นั้นขอละไว้ในที่นี้ ส่วนคำสั่งใน MATLAB จะมีรูปแบบคำสั่งเป็น

Polyfit(x,y,n) โดย x, y เป็นจุดข้อมูลที่ต้องการ fit curve ด้วย polynomial degree n MATLAB จะให้ สัมประสิทธิ์ของ polynomial y = f(x) คืนมาโดยเรียงจากสัมประสิทธิของกำลังมากไปหาน้อย

จากตัวอย่างที่ผ่านมาในเรื่องของ interpolation หากต้องการ fit curve ด้วย polynomial กำลัง 2 จะใช้

```
x = [0, 1, 2, 3, 4];
y = [0, 200, 600, 680, 800];
p=polyfit(x,y,2)
p =
  -34.2857  345.1429  -28.5714
```

หมายความว่า เราได้ curve $y = a_2x^2 + a_1x + a_0$ ซึ่งเป็น curve กำลัง 2 ที่ห่างจากกลุ่มข้อมูลนี้น้อยที่สุด โดยมีค่า a_2 = -34.2857, a_1= 345.1429 และ a_0 = - 28.5714 ตามลำดับและหากต้องการหาค่าที่ 2.5 วินาที ก็สามารถใช้คำสั่ง polyval ซึ่งได้ผลเป็น

```
y1=polyval(p,2.5)
y1 =
    620.0000
```

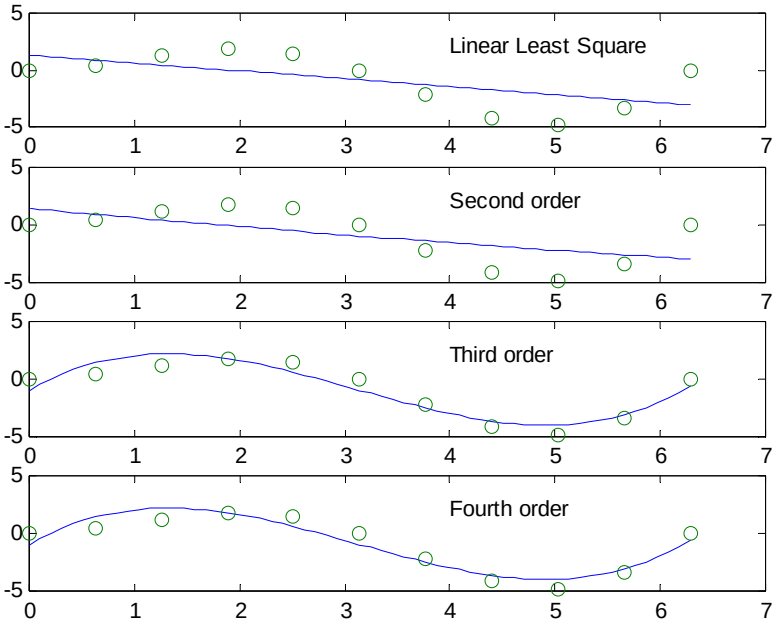
หรือถ้าต้องการหาค่าหลายจุดเช่นที่ 1.5 วินาที และ 3.5 วินาที ก็สามารถใช้

```
s=[1.5 3.5];
y1=polyval(p,s)
y1 =
  412.0000  759.4286
```

ซึ่งสามารถนำไปเปรียบเทียบกับกรณีของ interpolation ที่ผ่านมาได้ และเพื่อแสดงให้เห็นถึงความแตกต่างของ polynomial แต่ละ degree สามารถพิจารณาได้จาก M-file ต่อไปนี้

```
% Curve Fitting Demo
x=0:pi/5:2*pi;
y=x.*sin(x);
s=0:pi/30:2*pi;       % Set interval for plot
% Linear
a1=polyfit(x,y,1);
a=polyval(a1,s);
% Second order
b1=polyfit(x,y,2);
b=polyval(b1,s);
% Third order
c1=polyfit(x,y,3);
c=polyval(c1,s);
% Fourth order
d1=polyfit(x,y,4);
d=polyval(d1,s);
subplot(4,1,1)
plot(s,a,'-',x,y,'o')
text(4,2,'Linear Least Square')
subplot(4,1,2)
plot(s,b,'-',x,y,'o')
text(4,2,'Second order')
subplot(4,1,3)
plot(s,c,'-',x,y,'o')
text(4,2,'Third order')
subplot(4,1,4)
plot(s,c,'-',x,y,'o')
text(4,2,'Fourth order')
```

และจะได้ผลออกมาดังนี้



สำหรับในเอกสารนี้คงจะกล่าวถึงวิธีการของ curve fitting และ interpolation ไว้เพียงเท่านี้ อย่างไรก็ตามสำหรับผู้อ่านที่จะนำหัวข้อเหล่านี้ไปใช้งานขอให้ระลึกอยู่เสมอว่า

- ในกรณีที่ค่าชุดข้อมูลที่ได้นั้นได้มาเป็นค่าที่ถูกต้องไม่มีการผิดพลาดของข้อมูล เช่นได้จากสมการคณิตศาสตร์แต่เราไม่ทราบสมการที่แน่ชัด วิธีการที่เหมาะสมในการหาค่าที่อยู่ระหว่างจุดข้อมูลควรจะใช้ interpolation เพราะค่าที่ได้จะผ่านจุดข้อมูลทุกจุดซึ่งเป็นสิ่งที่ถูกต้อง แต่วิธีการนี้จะเป็นวิธีการที่ใช้เวลาในการคำนวณมากโดยเฉพาะอย่างยิ่งถ้าข้อมูลมีมาก
- ในกรณีที่ข้อมูลที่ได้อาจมีความผิดพลาดบ้าง เช่นข้อมูลที่ได้จากการทดลอง วิธีที่เหมาะสมน่าจะเป็นวิธีการใช้ Least Square Fitting เพราะข้อมูลที่ได้จากการทดลองก็ใช่ว่าจะไม่มีข้อผิดพลาดจากการวัดเลยดังนั้นกราฟที่ได้ก็ไม่จำเป็นต้องผ่านจุดข้อมูลทุกจุดแต่ควรจะเป็นเส้นที่ทำให้เกิดความผิดพลาดจากข้อมูลโดยรวมทั้งหมดน้อยที่สุดมากกว่า

การแก้ปัญหาทางวิศวกรรม

การเขียนกราฟแสดงผลการทดลองทางวิศวกรรม

สำหรับ Interpolation และ Curve Fitting นี้ถือว่าเป็นสิ่งสำคัญอย่างหนึ่งในการเขียนรายงานทางวิศวกรรมเช่นการเขียนกราฟเพื่อนำเสนอผลการทดลอง ตัวอย่างนี้จะสมมุติว่าเราทำการทดลองเรื่อง Impact of a Jet ซึ่งเป็นการหาแรงกระทำของกระแสของของไหลที่พุ่งเข้ากระทบวัตถุแบบต่างๆ ซึ่งมีรายละเอียดคร่าวๆดังนี้

การทดลองจะเป็นการปล่อยน้ำพุ่งเข้ากระทบเป้าหมายลักษณะต่างๆ จากนั้นจะเป็นการพิจารณาผลกระทบของลักษณะของเป้าแบบต่างๆ ที่มีผลต่อแรงกระทำที่เกิดขึ้นโดยแรงกระทำ F จะเท่ากับ

$$F = \dot{m}U(1-\cos\theta)$$

เมื่อ $\dot{m}$ คืออัตราการไหล U คือความเร็วที่กระแสน้ำพุ่งกระทบเป้าหมายและ θ เป็นมุมที่กระแสน้ำเปลี่ยนทิศทางไปเมื่อพุ่งเข้ากระทบเป้าหมาย นอกจากนั้นจุดประสงค์อีกอย่างหนึ่งของการทดลองคือการเปรียบเทียบลักษณะของกราฟที่ได้จากการทดลองเทียบต่อทางทฤษฎี

ในที่นี้สมมุติว่ากำหนดเป้าหมาย 2 แบบคือเป้ารูประนาบแบน (flat plat) ซึ่งมี θ เท่ากับ 90 องศา และรูปถ้วยครึ่งวงกลม (hemispherical cup) ซึ่งมี θ เท่ากับ 180 องศา ดังนั้นสำหรับระนาบตรงจะมีแรงกระทำเท่ากับ

$F = \dot{m}U$ ระนาบแบน

และสำหรับถ้วยครึ่งวงกลม

$F = 2\dot{m}U$ ครึ่งวงกลม

ดังนั้นหากว่าเราเขียนกราฟเทียบแรงกระทำที่เกิดขึ้นบนเป้าหมาย F กับโมเมนต์ตัมของของไหล $\dot{m}U$ จะได้กราฟที่มีความชันเท่ากับ 1 สำหรับระนาบแบนและเท่ากับ 2 ของรูปถ้วยครึ่งวงกลมตามลำดับ

แรงที่วัดจากการทดลองจะใช้เครื่องมือที่มีลักษณะเป็นคานที่วางแท่งน้ำหนักมวล 600 g เพื่อสร้างโมเมนต์ให้เท่ากับโมเมนต์ที่เกิดจากแรงปะทะของของไหล เมื่อคานอยู่ในสภาพสมดุลย์จะได้

$$F\,L = mgl$$

เมื่อ m คือมวลของแท่งน้ำหนัก g คือค่าความเร่งเนื่องจากแรงโน้มถ่วง L คือระยะจากจุดหมุนถึงตำแหน่งที่กระแสน้ำกระทำ ซึ่งเท่ากับ 150 mm และ l คือระยะที่วางน้ำหนักซึ่งได้จากการทดลอง ดังนั้นแรงกระทำจะเป็น

$$F = \frac{mgl}{L}$$

สำหรับในการทำการทดลองเราจะกระทำที่อัตราการไหลต่างๆ กัน 6 ค่าสำหรับวัตถุแต่ละชิ้น ส่วนค่าที่บันทึกในระหว่างการทดลองจะมีดังนี้

- ปริมาตรน้ำและเวลาที่ใช้เพื่อใช้ในการหาอัตราการไหล จะใช้ชื่อตัวแปร TF, VF, TH, LH โดย T คือเวลา V คือปริมาตร ส่วน F และ H แทนระนาบเรียบและถ้วยครึ่งวงกลมตามลำดับ
- ระยะทางที่วางน้ำหนักที่ใช้ถ่วง ซึ่งจะสามารถนำไปคำนวณหาแรงที่กระทำของของไหลได้ จะใช้ชื่อตัวแปร LF, LH โดย L แทนระยะทางส่วน F และ H เหมือนเดิม

สำหรับในขั้นแรกนี้เราเขียน M-file ขึ้นมา โดย file นี้ผู้เรียบเรียงเขียนขึ้นมาเพื่อจะใช้ใน M-book ซึ่งเป็นการทำงานร่วมกันระหว่าง MATLAB และ Microsoft Word for Windows ซึ่งรายละเอียดจะกล่าวถึงในบทของ M-book โดยเฉพาะ หากแต่ว่าในที่นี้ค่า **LF, TF, VF, LH, TH, VH** จะสมมุติว่ามีการกำหนดค่าที่ได้จากการทดลองเรียบร้อยแล้วและเป็นค่าที่ส่งมาจาก Microsoft Word ลักษณะของ M-file เป็นดังนี้

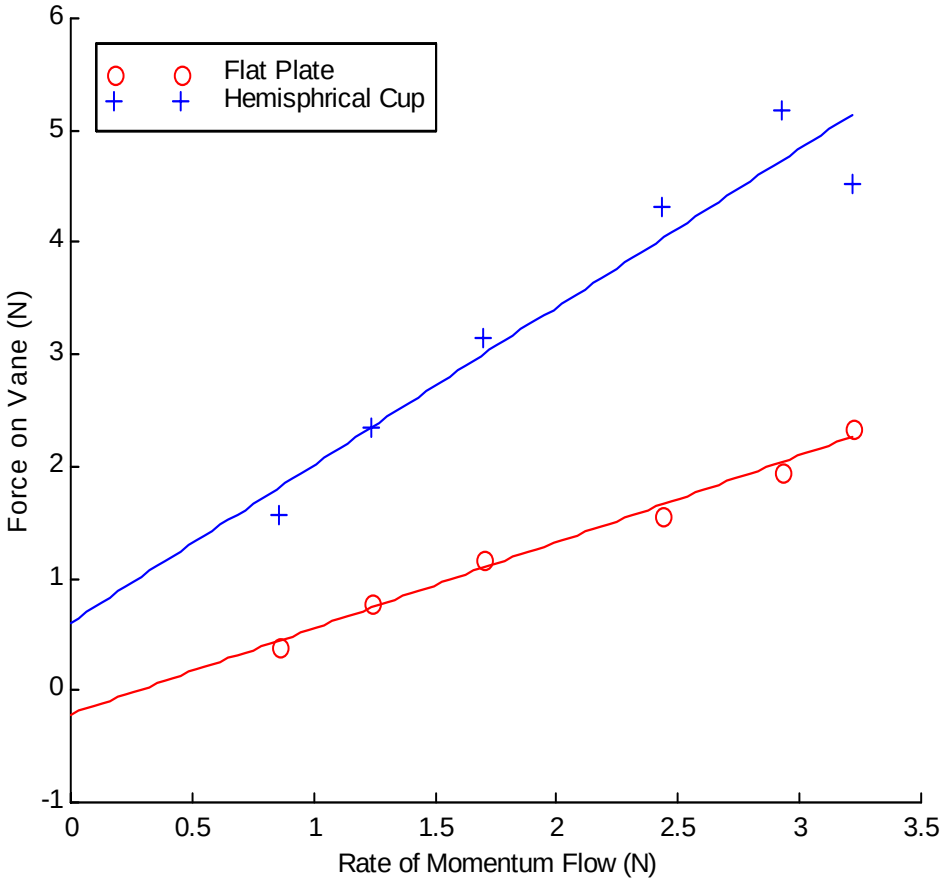
```
%  Impact of a Jet for M-book

%  Initialized Data
   LF=LF'; TF=TF'; VF=VF';LH=LH';TH=TH';VH=VH';
%  List of Constant
   grav = 9.81;          % [m/s2]
   NozzelDim = 10;       % [mm]
   High = 35   ;         % [mm]
   ArmLength = 150;      % [mm]
   mass = 0.6;           % [kg]
%  Initialize Figure and hold it
   figure('NumberTitle','off','color',[1 1 1])
   hold on
%  Calculation and Plot for Hemispherical Cub
   FlowRate = VH./TH ;                                  % [Liter/s]
```

```
NozzelVel = 4*FlowRate*(10^3)/(pi*NozzelDim^2);        %[m/s]
HitVel =sqrt(NozzelVel.^2-(2*grav*High/1000));         % [m/s]
ActualForce = LH * mass * grav /ArmLength;             % [Newton]
TheoryForce = HitVel.*FlowRate;                        % [Newton]
HH=plot(TheoryForce, ActualForce,'b+');    % เขียนจุดที่ได้จากการทดลอง
p =polyfit(TheoryForce,ActualForce,1);     % ใช้ Polyfit หาความชันเส้นกราฟของ
SlopHemi =p(1,1);                          % หาความชันเส้นกราฟจากค่า p
MaxOfx =max(TheoryForce);
MinOfx =min(TheoryForce);
xx =linspace(0,MaxOfx);
yy =polyval(p,xx);
plot(xx,yy,'b');                           % เขียนกราฟเส้นตรงความชัน p
%  Calculation and Plot for Flat Plate
FlowRate = VF./TF ;                                    % [Liter/s]
NozzelVel = 4*FlowRate*(10^3)/(pi*NozzelDim^2);        % [m/s]
HitVel =sqrt(NozzelVel.^2-(2*grav*High/1000));         % [m/s]
ActualForce = LF * mass * grav /ArmLength;             % [Newton]
TheoryForce = HitVel.*FlowRate;                        % [Newton]
HF = plot(TheoryForce, ActualForce,'ro');      % เขียนจุดที่ได้จากการทดลอง
p = polyfit(TheoryForce,ActualForce,1);        % ใช้ Polyfitหาความชันเส้นกราฟของ
SlopeFlate =p(1,1);
MaxOfx =max(TheoryForce);
MinOfx =min(TheoryForce);
xx =linspace(0,MaxOfx);
yy =polyval(p,xx);
plot(xx,yy,'r');                           % เขียนกราฟเส้นตรงความชัน p
H =[HF,HH];                                % เขียนกราฟ Legend ของกราฟทั้งสองเส้น
legend(H,'Flat Plate','Hemisphrical Cup',0)
xlabel('Rate of Momentum Flow (N)')
ylabel('Force on Vane (N)')
fprintf('ความชันเส้นกราฟของ Hemisphrical Cup เท่ากับ %5.2f \n',SlopHemi)
fprintf('ความชันเส้นกราฟของ Flat Plate เท่ากับ %5.2f \n',SlopeFlate)
hold off

% ************ END **************
```

ซึ่งหากมีการกำหนดค่า **LF, TF, VF, LH, TH, VH** จะได้รูปกราฟและผลเป็นดังนี้

ความชันเส้นกราฟของ Hemispherical Cup เท่ากับ 1.41

ความชันเส้นกราฟของ Flat Plate เท่ากับ 0.77



ซึ่งสามารถแสดงผลตามที่เราต้องการได้

บทที่ 7 NUMERICAL INTEGRATION AND DIFFERENTIATION

7.1 Numerical Integration

สำหรับการอินทิเกรดเชิงตัวเลข (Numerical Integration) นั้นเป็นวิธีการที่นิยมใช้อย่างแพร่หลายในสองกรณีใหญ่ๆคือ

1) ในกรณีที่เราได้ข้อมูลมาเป็นชุดของข้อมูล ซึ่งเราอาจจะต้องการ integrate โดยตรงโดยไม่ต้องการที่จะทำการ interpolation หรือ fitting ข้อมูลก่อนแล้วจึงนำมา integrate ด้วยวิธีปกติ
2) ในกรณีที่มี function ที่ยุ่งยากต่อการ integrate ด้วยวิธี analytical แบบธรรมดาที่ใช้กันทั่วไป

เราทราบดีแล้วว่าการ integrate จะยุ่งยากในการปฏิบัติหากว่าต้องทำด้วยตนเองหรือในบางกรณีอาจเป็นไปไม่ได้เลย แต่หากเราใช้วิธี numerical integration แล้วทำงานบนเครื่องคอมพิวเตอร์ความยุ่งยากเหล่านั้นจะลดน้อยลงไปมากเพราะการ integrate ก็คือการรวมขนาดของชิ้นส่วนเล็กๆเข้าด้วยกันนั่นเอง สำหรับการ integrate ใน MATLAB สามารถทำได้ทั้งสองกรณี เพราะว่าในกรณีที่ 2 ถ้าต้องการใช้ numerical integration ก็จะเหมือนกันกับวิธีแรกคือจะต้องคำนวณค่า function ออกมาเป็นจุดๆ ก่อน ซึ่งความละเอียดของผลที่ได้ก็ขึ้นอยู่กับความห่างของข้อมูลแต่ละจุด

อย่างไรก็ตามสำหรับ MATLAB แล้วหากเรามีข้อมูลแบบที่เป็นจุดซึ่งไม่ทราบค่า function เราจะสามารถใช้วิธี numerical integration แบบ Trapezoidal Numerical Integration ซึ่งถือว่าจะมีความละเอียดไม่มากนักแต่ว่าหากเราทราบ function ที่จะ integrate เราจะสามารถใช้วิธี numerical ที่ให้ความละเอียดกว่า เช่น Simpson's Rule แต่วิธีการใช้แบบหลังนี้อาจจะยุ่งยากกว่าบ้างเพราะต้องมีการเขียน function file เพิ่มเติมขึ้นมา

❶ Trapezoidal Numerical Integration

สำหรับ trapezoidal numerical integration นั้น MATLAB จะใช้ function **`trapz`** โดยการกำหนดค่า vector หรือ matrix x และ y เป็นการกำหนดข้อมูลแต่ละจุด ซึ่งมีรูปแบบดังนี้

`trapz(x,y)` จะเป็นการ integrate $\int y(x)dx$ โดย x และ y เป็น column vector ถ้า x และ y เป็น matrix จะคำนวณแต่ละ column ของ matrix นั้น ถ้า x เป็น column vector และ y เป็น matrix จะเทียบแต่ละ column ของ y กับ x

ลองพิจารณาจากตัวอย่างหากว่าเราต้องการหาค่า integrate ต่อไปนี้

$$\int_0^{\pi} x\sin(x)dx$$

ซึ่งหากว่าใช้วิธีการ integrate-by-path จะได้ค่าเท่ากับ π หากว่าเราใช้ function **trapz** ก็จะมีขั้นตอนดังนี้ อันดับแรกแบ่งค่า x ออกเป็นช่วงเท่าๆ กัน (ในความเป็นจริงหากเป็นข้อมูลที่ได้มานั้น ค่า x ไม่จำเป็นต้องมีช่วงห่างเท่ากันก็ได้) ซึ่งหากต้องการแบ่งช่วงออกเป็น 50 ช่วงจะได้

```
x=0:pi/50:pi;
```

จากนั้นคำนวณหาค่า y ทั้ง 50 จุดของ x จะได้

```
y=x.*sin(x);
```

และหาค่าได้เป็น

```
A=trapz(x,y)
A =
    3.1406
```

ซึ่งจะพบว่าค่ามีความผิดพลาดจากค่า π = 3.14159 ในทศนิยมตำแหน่งที่ 3

❷ Quadrature Integration

Quadrature คือวิธีการทาง numerical ที่ใช้หาพื้นที่ใต้กราฟของ function สำหรับ MATLAB จะมี function ให้เลือกได้ 2 function ซึ่งแต่ละ function จะใช้วิธีการคำนวณที่แตกต่างกัน function ที่กล่าวถึงนั้น คือ function **quad** และ **quadz** อย่างไรก็ตามการใช้ function ทั้งสองนี้ผู้ใช้จะต้องเขียน function file ขึ้นมา โดย function file นั้นจะต้องบรรจุ function y = f(x) ที่ต้องการ integrate ไว้แล้วใช้คำสั่ง **quad** หรือ **quadz** คำนวณหาค่า integrate โดยมีรูปแบบของคำสั่งทั้งสองเป็น

quad('function',a,b) หรือ

quad('function',a,b,tol,trace) หรือ
quad('function',a,b,tol,trace,p1,p2...)

โดย 'function' คือชื่อ function file ที่เขียนขึ้น (ซึ่งในตัวอย่างนี้จะมีชื่อเป็น function.m) a และ b เป็นค่าคงที่กำหนดจุดเริ่มการ integrate จาก a ถึง b ส่วนที่เหลือจากนี้คือ **tol, trace, p1**. จะมีหรือไม่ก็ได้สำหรับ **tol** เป็นค่าที่กำหนด relative error ของการ integrate ถ้าไม่กำหนด MATLAB จะตั้งค่านี้ไว้ที่ 0.001, **trace** คือ function ที่จะใช้ plot จุด และ $\mathbf{p_1}$, $\mathbf{p_2}$ คือค่า parameters อื่นๆ ที่จะส่งไปให้กับ MATLAB ซึ่งสามารถดูรายละเอียดได้จากคำสั่ง **help quad** สำหรับคำสั่งนี้ MATLAB จะใช้ Simpson's Rule ในการหาค่า integrate

Quad8('function',a,b) หรือ

Quad8('function',a,b,tol,trace) หรือ
Quad8('function',a,b,tol,trace,p1,p2...)

Parameter ต่างๆ จะเหมือน function **quad** เพียงแต่การคำนวณจะละเอียดมากกว่า โดยจะใช้ Newton-Cotes 8 panel rule ซึ่งจะเหมาะกว่า **quad** ในด้าน

ความละเอียดและหาค่า integral ของ function ที่มี singularity อยู่ในช่วงที่ต้องการ integrate

สำหรับตัวอย่างที่จะใช้ทั้งสองวิธีจะขอใช้ตัวอย่างเดิมที่กล่าวมาแล้วในวิธีก่อนหน้านี้ เพื่อใช้ในการเปรียบเทียบ แต่ถ้าต้องการใช้ function **quad** จะต้องสร้าง M-file เป็น function file ขึ้นมาก่อน ในที่นี้ให้สมมุติว่า save ชื่อเป็น myint.m

```
function y =myint(x)
y  = x.*sin(x);
```

จากนั้นใช้คำสั่ง

```
A=quad('myint',0,pi)
A =
    3.1416
```

ซึ่งเมื่อเทียบกับค่าจริงคือ $\pi = 3.14159$ แล้วจะพบว่าผิดพลาดที่ทศนิยมตำแหน่งที่ 4 หากต้องการเพิ่มความแม่นยำขึ้นก็สามารถใช้ function **quad8** โดยยังใช้ function myint เหมือนเดิม

```
format long
A=quad8('myint',0,pi)
A =
   3.14159265358962
```

สาเหตุที่ใช้คำสั่ง **format long** เพื่อให้เห็นว่าเมื่อเทียบกับค่าจริงคือ

```
pi
ans =
   3.14159265358979
```

จะได้เห็นว่าความผิดพลาดนั้นเกิดขึ้นที่ทศนิยมตำแหน่งที่ 13 ซึ่งนับว่าน้อยมาก

จากตัวอย่างที่ผ่านมาทั้งสามตัวอย่างท่านคงทราบถึงความแม่นยำของการคำนวณ numerical integration ด้วย function **trapz**, **quad** และ **quad8** อย่างหนึ่งที่ควรชี้ให้เห็นก็คือ ถ้าหากว่าต้องการเพิ่มความถูกต้องให้มากขึ้นสามารถทำได้ดังนี้

- ในการใช้ **trapz** ให้เพิ่มจุดของการ integrate ขึ้นหากทำได้
- ในการใช้ **quad** และ **quad8** สามารถทำได้โดยลดค่า **tol** ให้ต่ำลง โดยสำหรับ **quad** และ **quad8** กำหนดเบื้องต้นไว้ที่ 0.001

7.2 การ Integrate Double Integral

ที่ผ่านมาเป็นวิธีการ integrate เทียบต่อตัวแปรเพียงตัวเดียว แต่ในหลายกรณีเราจำเป็นต้อง integrate เทียบต่อ 2 ตัวแปรก็อาจทำได้โดยใช้ขั้นตอนดังนี้คือ ให้หาค่า function โดยให้ตัวแปรตัวหนึ่งคงที่ไว้ก่อนแล้วดำเนินการ integrate เทียบต่อตัวแปรตัวเดียว จากนั้นเมื่อได้ค่าแล้วจึง integrate เทียบต่อตัวแปรที่เหลือ เพื่อให้เข้าใจได้ง่ายขึ้นขอยกตัวอย่างการคำนวณหา double integral ต่อไปนี้

$$\int_0^1 \int_0^1 e^{-x^2-y^2} dydx$$

การใช้ MATLAB ขั้นแรก เราต้องสร้าง function file สมมุติชื่อ myint2.m ดังนี้

```
function f = myint2(x,y)
f = exp(-x.^2 - y.^2)
```

จากนั้นคำนวณหา integral โดยกำหนดค่า x ไว้ทีละค่า สมมุติเราต้องการแบ่ง x ออกเป็น 20 ช่วง จากนั้นใช้คำสั่ง (หรืออาจเขียน script file)

```
x = 0:1/19:1;
for   k = 1:20
      A2(i) = quad('myint2',0,1,[],[],x(i));
end
```

ในที่นี้เราต้องการส่งค่า integrate โดยใช้ **quad** function ในการ integrate จาก 0 ถึง 1 และใช้ **tol** และ **trace** ตามที่เครื่องกำหนด (ส่งเป็นค่าว่าง [ ]) และให้ส่งค่า x(i) เข้าไปสู่ M-file myint2 เพื่อคำนวณค่า x ดังนั้นเมื่อทำ loop ครบ 20 ครั้ง (k = 1 ถึง 20) เราจะได้ค่า A2 เป็น matrix ขนาด 1 x 20 จากนั้นเมื่อเราทราบค่า A2 แล้วเราสามารถ integrate function A2 ต่อไปได้โดยตรง จากนั้น fixed y แล้วเปลี่ยนค่า x บ้าง หรือก็คือการใช้คำสั่ง **trapz** นั่นเอง เช่น

```
A = trapz(x,A2)
A =
  0.55762
```

สำหรับ function ที่ใช้เพื่อการ integrate double integral เป็น function ที่ชื่อ **dblquad** ซึ่งมีรูปแบบของคำสั่งดังนี้

dblquad('function',inmin,inmax,outmin,outmax,tol,trace,'method')

เป็นการ integrate double integral โดยการใช้ function **quad** โดยที่ **inmin, inmax** คือค่า limit บนและล่างของ integrand ด้านใน ส่วน **outmin** และ **outmax** จะเป็นค่า limit บนและล่างของ integrate ด้านนอก function คือชื่อ function file ที่บรรจุ integrand ซึ่งจะต้องเป็น function ของสองตัวแปรโดย ส่วน option ที่เหลือคือ **tol** และ **trace** จะเหมือนกับการใช้ function **quad** หรือ **quad8** ปกติ ส่วน **'method'** นี้เป็น option ของวิธีการ integrate ว่าจะเป็น **'quad'** หรือ **'quad8'** ซึ่งหากไม่กำหนด MATLAB จะใช้ **'quad'** ส่วน function file ต้องอยู่ในรูป **function(inner, outer)** ซึ่ง **inner** และ **outer** คือตัวแปรที่ใช้กับ integrate ด้านในและด้านนอกตามลำดับ

สำหรับตัวอย่างการใช้ function นี้ ขอยกตัวอย่างเดิมที่ผ่านมาคือ

$$\int_0^1 \int_0^1 e^{-x^2-y^2}\, dydx$$

ขั้นแรกจะเป็นการสร้าง function file ที่ใช้จะเป็น integrand สมมุติใช้ชื่อ myint3.m

```
function f = myint3(x,y)
f = exp(-x.^2- y.^2);
```

และผลจะเป็น

```
dblquad('myint3',0,1,0,1)
ans =
    0.5577
```

จะพบว่าผลที่ได้ไม่แตกต่างจากวิธีก่อนหน้านี้มากนักแต่การใช้งานจะง่ายกว่ามาก

ที่ผ่านมาเป็นการใช้ MATLAB คำนวณหาค่า integration ทั้งในลักษณะแทนค่าได้โดยตรงกับแบบที่ต้องเขียน function ขึ้นมาเพิ่มเติมหรือที่เรียกใน MATLAB ว่าเป็น function ประเภท **function function** คือ การเรียกใช้ function หนึ่งที่จะต้องไปเรียกอีก function หนึ่งมาใช้งานด้วย นอกเหนือจากการ integrate ด้วยวิธีการเชิงตัวเลขแล้ว MATLAB ยังมี Symbolic Integration ซึ่งเป็นส่วนหนึ่งของ Symbolic Toolbox ซึ่งจะกล่าวถึงในภายหลัง

7.3 Numerical Differentiation

สำหรับ differentiation ของ y(x) เทียบต่อ x จะนิยมว่า

$$\frac{dy}{dx} = y'(x) = \lim_{\Delta x \to 0} \frac{y(x+\Delta x) - y(x)}{\Delta x}$$

สำหรับการทำ Numerical Differentiation นี้มักไม่นิยมที่จะใช้กับ Continuous function ที่ทราบค่าอยู่แล้ว เพราะการ differentiation function ด้วยวิธี analytical เป็นเรื่องที่ค่อนข้างง่าย เมื่อเทียบกับ integration อย่างไรก็ตามถ้าเราไม่ทราบค่า function ที่ต้องการหาอนุพันธ์ แต่เป็นเพียงการทราบข้อมูลแต่ละจุด (x,y) แล้วมีความจำเป็นต้องการหา $y'(x)$ ก็สามารถใช้การประมาณได้ว่า

$$y'(x_k) = \frac{y(x_k) - y(x_{k-1})}{x_k - x_{k-1}}$$

ซึ่งค่าจะถูกต้องมากขึ้นถ้า $\Delta x = x_k - x_{k-1}$ มีค่าเข้าใกล้ศูนย์มากขึ้น สำหรับ function ที่ใช้หา difference จะใช้

diff(x) หาค่า difference ของ element ต่าง ๆ ของ x ถ้า x เป็น vector 1 x n (หรือ n x 1) ให้ค่า vector ขนาด 1 x (n-1) [หรือ (n-1) x 1] ตามลำดับ ถ้า x เป็น matrix m x n แล้ว จะให้ค่า matrix ขนาด (m-1) x n

ในการหาค่า $\frac{dy}{dx}$ จาก set ของ coordinate (x_i, y_i) จะสามารถหาได้จาก

```
diff(y)./diff(x)
```

ตัวอย่างเช่นจากการทดลองได้ x = [1 2 3 4 5] และ y = [1 1.3 1.8 2.1 2.4] หากต้องการหาค่า $\frac{dy}{dx}$ สามารถทำได้ดังนี้

```
x = [1, 2, 3, 4, 5];
y = [1, 1.3, 1.8, 2.1, 2.4];
dydx = diff(y)./diff(x)
dydx =
    0.3000    0.5000    0.3000    0.3000
```

จากผลที่ได้หมายความว่าค่า $\frac{dy}{dx}$ ที่ x=1 มีค่า 0.3000, ค่า $\frac{dy}{dx}$ ที่ x=2 มีค่า 0.5000, ... ตามลำดับไปเรื่อยๆ

สำหรับการใช้ Numerical differential นั้นจะถูกต้องมากขึ้น ถ้าหากว่าค่า $\Delta x = x_i - x_{i-1}$ มีค่าน้อย แต่สำหรับในการทดลองทางวิศวกรรมนั้นอาจไม่สามารถทำให้ค่า Δx เล็กมากได้ เพราะเหตุผลหลายๆ ประการเช่น ถ้า Δx มีค่าน้อย จำนวนข้อมูลก็จะมาก เวลาในการเก็บข้อมูลนาน ค่าใช้จ่ายในการทดลองจะสูง ฯลฯ ดังนั้นในกรณีที่เรามีความจำเป็นที่จะต้องมีค่า Δx กว้างเราอาจต้องทำการ interpolate หรือ curve fitting ก่อนแล้วจึงนำมาหาค่า difference

การแก้ปัญหาทางวิศวกรรม

➢ การหากำลังของเครื่องยนต์สันดาปภายใน

การหากำลังของเครื่องยนต์สันดาปภายในมีหลายวิธี วิธีหนึ่งที่นิยมคือการวัดความดันในกระบอกสูบเทียบต่อตำแหน่งมุมของเพลาข้อเหวี่ยง จากนั้นจะสามารถนำมุมของเพลาข้อเหวี่ยงมาหาปริมาตรกระบอกสูบในขณะนั้น และนำมาเขียนกราฟแสดงความดันและปริมาตรกระบอกสูบซึ่งจะได้กราฟแสดงวัฏจักรแสดงการทำงานของเครื่องยนต์ และพื้นที่ใต้กราฟจะแทนงานของเครื่องยนต์ จากนั้นกำลังของเครื่องยนต์จะหาจากงานต่อด้วยเวลา กำลังที่ได้จากการหาด้วยวิธีนี้จะเรียกว่า indicated power และเครื่องมือวัดความดันในกระบอกสูบนั้นเรียกว่า indicator

สำหรับค่าที่เกี่ยวข้องกับเครื่องยนต์สันดาปภายในมีดังนี้

Indicated mean effective pressure, imep

$$imep = \frac{W}{V}$$

Indicated Power

$$P_i = \frac{imepV\ N}{n_r}$$

เมื่อ V คือปริมาตรกระบอกสูบ, N คือความเร็วรอบเครื่องยนต์ และ n_r จะมีค่าเท่ากับ 1 สำหรับเครื่องยนต์ 2 จังหวะและ 2 สำหรับเครื่องยนต์ 4 จังหวะ

สมมุติว่าการทดลองหนึ่งเราได้ค่าปริมาตรกระบอกสูบและความดันของเครื่องยนต์ 4 จังหวะ ดังนี้

ปริมาตรกระบอกสูบ (CC)	ความดันจังหวะอัด-ระเบิด (bar)	ความดันจังหวะดูด-คาย (bar)
141.0	16.0	15.0
164.25	47.0	11.0
186.50	42.0	9.5
207.55	37.0	8.0
230.00	33.0	7.0
324.13	21.0	4.5
410.40	14.5	3.0
500.00	11.0	2.0
590.04	8.5	1.5
680.00	7.0	1.0
770.00	5.0	1.5
815.00	4.5	1.5
860.00	3.5	3.5

อันดับแรกจะเป็นการ plot Indicator diagram ของความดัน-ปริมาตร สำหรับเครื่องยนต์ 4 จังหวะ ความดันจะต้องแบ่งออกเป็นสองส่วนคือ ในส่วนจังหวะอัดและส่วนจังหวะระเบิด หรือเป็นช่วงที่มีความดันสูงและช่วงความดันต่ำ ซึ่งจากข้อมูลจะได้กราฟลักษณะดังนี้

จากนั้นจะเป็นการหางานหรือพื้นที่ใต้กราฟ โดยการใช้ numerical integration โดยขั้นแรกจะต้องหาค่าความแตกต่างของความดันช่วงความดันสูงและความดันต่ำกันเสียก่อน สำหรับ function file ที่ใช้มีลักษณะดังนี้

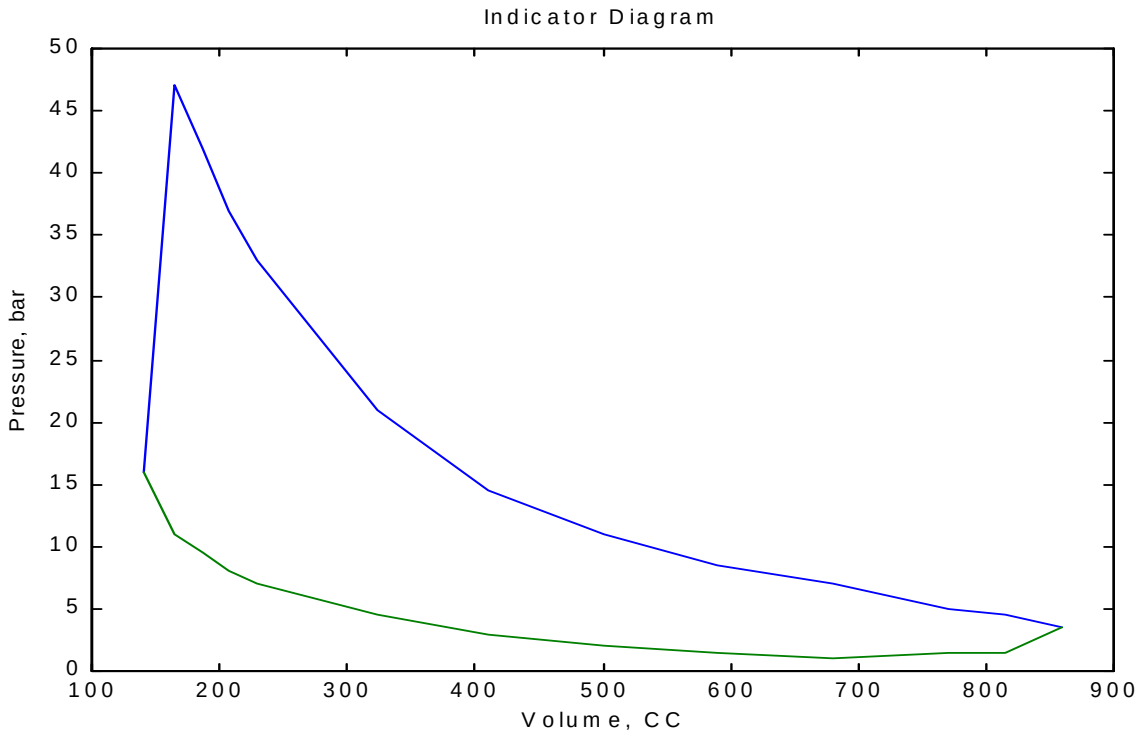
```
function [Power,imep]=ice_PV(V,PH,PL,N,nr)
plot(V,PH,V,PL)
ylabel('Pressure, bar');
xlabel('Volume, CC');
title('Indicator Diagram')

DP=PH-PL;
```

```
W=(trapz(V,DP))/10;                 % N-m
Vdis=max(V)-min(V);                 % Displace Volume
imep = (W/Vdis);                    % MPa
Power = W*N*2*pi/(nr*60000);        % kW

fprintf('Indicated Power = %5.2f kW \n' ,Power)
fprintf('Indicated mep = %5.2f MPa \n' ,imep)
```

และเมื่อกำหนดค่าตามตารางและให้ ความเร็วรอบเท่ากับ 1200 rpm สำหรับเครื่องยนต์สี่จังหวะ $n_r = 2$ 0 จะได้

```
N=1200;
nr=2;
[Power imep]=ice_pv(V,PH,PL,N,nr);

Indicated Power = 53.52 kW
Indicated mep = 1.18 MPa
```

ส่วนกราฟจะเป็นไปตามรูปด้านล่าง



บทที่ 8 SOLUTION OF ORDINARY DIFFERENTIIAL EQUATIONS

เป็นที่ยอมรับกันว่าปัญหาในทางวิศวกรรมคงจะหนีไม่พ้นเรื่องของการแก้สมการอนุพันธ์ ซึ่งถือเป็นพื้นฐานหลักของสมการต่างๆ ที่เรานำมาประยุกต์ใช้กัน แม้ว่าสมการท้ายสุดที่เรานำไปใช้นั้นไม่ได้อยู่ในรูปของสมการอนุพันธ์แล้วก็ตาม แต่สำหรับวิศวกรที่ต้องทำงานอยู่กับทฤษฎีต่างๆในงานวิจัยเพื่อปรับปรุงและค้นคว้าสิ่งใหม่ๆนั้นสมการเชิงอนุพันธ์จะเข้าไปมีบทบาทอย่างมากอยู่ตลอดเวลา ในหัวข้อนี้จะกล่าวถึงการแก้สมการ Ordinary Differential Equation (ODE) ด้วยการใช้ MATLAB สำหรับสมการเชิงอนุพันธ์ประเภท Partial Differential Equation (PDE) นั้น MATLAB จะมี toolbox ต่างหากซึ่งไม่ได้รวมอยู่กับโปรแกรมปกติ จึงไม่ขอกล่าวถึงในที่นี้

ในเอกสารชุดนี้จะกล่าวถึงการแก้สมการ ODE ทั้งแบบ first order และ order ที่สูงกว่าและนอกจากนั้นเนื่องจาก MATLAB จะมี function สำหรับ initial value problem เท่านั้น เอกสารชุดนี้จึงเสนอแนวทางที่จะใช้ function ที่มีอยู่ใช้แก้สมการประเภท boundary value problems ด้วย

8.1 First Order ODE

สำหรับ first order ODE จะมีรูปสมการเป็น

$$y' = \frac{dy}{dx} = g(x, y)$$

เมื่อ x เป็น independent variable และ y เป็น dependent variable ในรูป function ของ x และถ้าหากว่า g(x,y) เป็น linear function ของ y เราจะเรียก ODE นี้ว่าเป็น linear ODE ไม่เช่นนั้นจะเรียกว่าเป็น nonlinear ODE นอกเหนือจากนั้น ถ้าหากว่าทุกเทอมของ g(x,y) มี y ประกอบอยู่ด้วยเสมอจะเรียกสมการนั้นว่าเป็น homogenous ODE หากไม่เป็นเช่นนั้นจะเรียก nonhomogeneous ODE

การหาคำตอบของสมการ ODE ก็คือการหาค่า

$$y = y(x) \quad \text{ที่ทำให้} \quad y' = g(x, y)$$

ซึ่งปกติจะมี y(x) มากกว่า 1 function ที่ทำให้สมการนี้เป็นจริงแต่สมการเหล่านั้นจะแตกต่างกันก็แต่เพียงสภาพเริ่มต้น (initial condition) หรือสภาพที่ขอบเขต (boundary condition) เท่านั้น ดังนั้นเราจึงเรียก function เหล่านั้นว่า general solutions อย่างไรก็ตามหาก y(x) ได้มีการกำหนดค่า boundary หรือ initial

condition เป็นที่เรียบร้อยแล้วค่า y(x) ที่ได้ก็จะเป็นคำตอบเฉพาะสภาพนั้นๆ หรือเรียกว่าเป็น particular solution ของสภาพการกำหนดนั้น

การแก้สมการ ODE นั้น MATLAB ใช้ Runge-Kutta Method (R-K) ซึ่งเป็นวิธีการที่ใช้แก้ initial value problem ordinary differential equations และ MATLAB จะใช้ขั้นตอนแบบ variable step size โดยจะมีการเปลี่ยน step size อัตโนมัติหากว่า solution แต่ละ step ที่ได้จากการคำนวณมีการเปลี่ยนแปลงค่ามากหรือน้อยเกินไป นอกเหนือจากนั้น ODE function ของ MATLAB จะเป็นประเภท function-function คือจะต้องมีการเขียน function file ที่บรรจุสมการ ODE นี้แยกเป็น function file ต่างหากก่อน แล้วจึงให้ ODE function นี้เรียกใช้อีกครั้งหนึ่ง ซึ่งก็จะคล้ายกับการใช้ function - function เพื่อการ integrate ในบทที่ผ่านมา

สำหรับ order ของ Runge-Kutta Method นั้น MATLAB จะมีให้เลือก 2 แบบคือใช้ order 2-3 ซึ่งจะเป็น function **ode23** และจะมี order 4-5 ซึ่งจะใช้ function **ode45** โดย ODE functions มีรูปแบบเป็น

[x,y]=ode23('function',[a b],initial)

MATLAB จะใช้ Runge-Kutta Method 2nd- order และ 3rd - order ส่วน 'function' คือชื่อ function file [function.m สำหรับในตัวอย่างนี้] ที่กำหนด function ที่ต้องการจะแก้สมการ ส่วน a และ b เป็นช่วงที่ต้องการจะหาค่าคำตอบ initial เป็นค่าของ function ที่ a (initial condition) ส่วน x และ y เป็นคำตอบ โดย y = y(x)

[x,y] = ode45('function',[a b],intial)

ค่า parameter ต่างๆ จะเหมือนกับ **ode23** แต่จะใช้ R-K order 4 และ 5 เพื่อให้ได้ค่าที่มีความแม่นยำสูงขึ้นกว่า **ode23** โดยตามค่าปกติ **ode23** จะมี relative error 0.001 ส่วน **ode45** จะมี relative error 10^{-6}

ตัวอย่างการใช้ function ode เช่นหากต้องการแก้สมการ first order ODE

$$y' = \cos x - x\sin x$$

ในช่วง 0 ถึง π และเป็นไปตาม initial condition ที่ x = 0, y(0) = 0 ซึ่งสมการนี้มี exact solution เป็น $y = x\cos x$

ขั้นแรก เขียน function M-file ในที่นี้สมมุติชื่อ myode.m

```
function yprime  = myode(x, y)
yprime = cos(x) - x.*sin(x)
```

ขั้นที่ 2 เขียน script M-file (หรืออาจจะทำโดยไม่ต้องเขียน M-file ก็ได้)

```
[x, y] = ode23('myode',[0  pi],0)
```

ซึ่งผลที่ได้จะเป็น column vector x และ y แต่เนื่องจากว่า MATLAB ใช้ variable step side ดังนั้นเราจึงไม่ทราบขนาดของ x และ y ที่แน่นอนหากเราต้องการทราบค่าในระหว่างช่วง 0 - π เราอาจจำเป็นต้อง interpolate ค่า x และ y ก่อน สำหรับในตัวอย่างนี้ถ้าพิจารณาค่าสุดท้ายที่ x = π จะได้ค่าที่ถูกต้องเป็น

```
y = p cos(p) = - p =-3.14159
```

ส่วน **ode23** จะให้ค่าดังนี้

```
[a b]=size(y);
y(a)
ans =
   -3.1417
```

นั่นคือ y(π) = -3.1417 แต่ถ้าลองใช้ **ode45** จะได้

```
[x, y] = ode45('myode',[0  pi],0)
[a b]=size(y);
format long
y(a)
ans =
   -3.14159260200719
pi
ans =
    3.14159265358979
```

ซึ่งจะเห็นว่าได้ค่า y(π) = - 3.1415926 ที่ใกล้เคียงกับค่าคำตอบที่แท้จริงมากกว่า **ode23**

ในหลายๆ กรณีการแก้สมการ ODE จำเป็นจะต้องหาค่าของ function ที่จุดที่เป็น singularity หรือเกือบจะเป็น singularity ทำให้การแก้สมการนั้นจะ converge ยากมากขึ้น ปัญหาประเภทนี้นิยมเรียกว่าสมการมี stiff condition ซึ่งการแก้สมการประเภทนี้ต้องการความละเอียดและเสียเวลาค่อนข้างมาก สำหรับ MATLAB ก็จะมี function สำหรับ ODE พวกนี้ด้วย เช่น **ode15s, ode23s** รายละเอียดสามารถหาเพิ่มเติมได้จากคู่มือหรือดูจาก **help ode23s**

8.2 Higher Order ODE

สำหรับในกรณี n order ODE ซึ่งมีสมการอยู่ในรูป

$y^{(n)} = g(x, y, y', y'', \ldots y^{(n-1)})$ เมื่อ $y=y(x)$ และ $y^{(n)} = \frac{d^n y}{dx^n}$

โดยมี initial condition เป็น

$$y(x_0) = a_0$$
$$y'(x_0) = a_1$$
$$\ldots\ldots\ldots$$
$$y^{(n-1)}(x_0) = a_{n-1}$$

เมื่อ x เป็น independent variable และ y เป็น dependent variable ในรูป function ของ x และถ้าหากว่า g(x,y,...) เป็น linear function ของ y หรือ derivative ของ y เราจะเรียก ODE นี้ว่าเป็น linear ODE of order n ไม่เช่นนั้นจะเรียกว่าเป็น nonlinear ODE of order n นอกเหนือจากนั้นถ้าหากว่าทุกเทอมของ g(x,y,...) มี y หรือ derivative ของ y ประกอบอยู่ด้วยเสมอเราจะเรียกสมการนั้นว่าเป็น homogenous ODE of order n หากไม่เป็นเช่นนั้นจะเรียก nonhomogeneous ODE of order n

ขั้นตอนการแก้สมการ ODE ที่มี order สูงกว่า 1 ขั้นแรกจะต้องทำให้ ODE สมการนั้นเปลี่ยนเป็นระบบสมการ first order ODE โดยการสร้างตัวแปรขึ้นใหม่ เช่นจาก

$$y^{(n)} = g(x, y, y', \ldots y^{(n-1)})$$

กำหนดให้

$$v_1(x) = y^{(n-1)}$$
$$v_2(x) = y^{(n-2)}$$
$$\ldots\ldots\ldots$$
$$v_{n-2}(x) = y''$$
$$v_{n-1}(x) = y'$$
$$v_n = y$$

ดังนั้น

$$v_1'(x) = y^{(n)} = g(x, y, y', \ldots y^{(n-1)})$$
$$v_2'(x) = y^{(n-1)} = v_1(x)$$
$$v_3'(x) = y^{(n-2)} = v_2(x)$$
$$\ldots\ldots\ldots$$
$$v_{n-2}'(x) = y'' = v_{n-3}(x)$$
$$v_{n-1}'(x) = y'' = v_{n-2}(x)$$
$$v_n'(x) = y' = v_{n-1}(x)$$

ซึ่งสมการทั้งหมดจะได้สมการ First order จำนวน n สมการ จากนั้นเราก็จะแก้สมการทั้ง n สมการพร้อมกันสำหรับตัวอย่างในที่นี้ขอยกตัวอย่างสมการ nonlinear second order ODE

$$y'' = g(x, y, y') = y'(1 - y^2) - y^2$$

ขั้นแรกให้

$$v_1(x) = y' \Rightarrow v_1'(x) = y'' = y'(1 - y^2) - y^2$$
$$v_2(x) = y \Rightarrow v_2'(x) = y' = v_1$$

จากนั้น เนื่องจาก $v_2 = y, v_1 = y'$ แทนค่าลงใน v_1' จะได้

$$v_1' = v_1(1 - v_2^2) - v_2^2$$
$$v_2' = v_1$$

ตัวแปรที่ต้องการ คือ v_1 และ v_2 ในตัวอย่างนี้ เราจะเขียน function file สมมุติให้ชื่อ myode.m

```
function dv = myode2(x,v)
dv = zeros (2, 1)        % สร้าง vector ขนาด 2x1 ซึ่งเท่ากับจำนวนสมการ
dv (1) = v(1).*(1-(v(2).^2))-v(2).^2;
dv (2) = v(1) ;
```

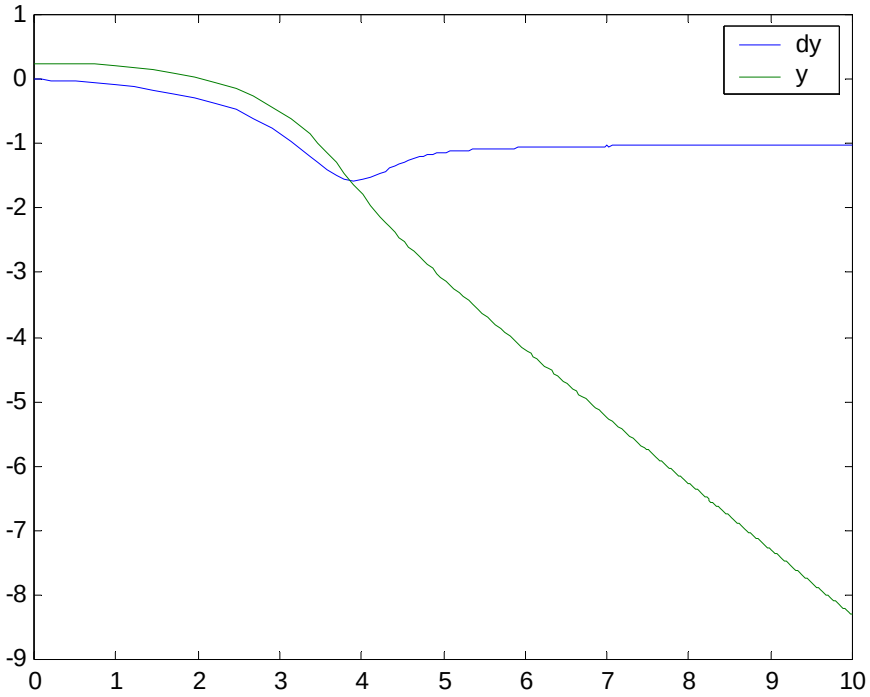
ถ้ากำหนดให้ initial condition เป็น $y'(0) = 0 = v_1(0)$

$$y(0) = 0.25 = v_2(0)$$

และต้องการหาค่าสมการในช่วง $0 \le x \le 10$ ใน MATLAB command จะใช้

```
t = [0 10];
initial = [0 0.25];
[x, y] = ode45('myode2',t,initial);
m = plot(x,y);
legend(m,'dy','y')
```

ผลที่ได้จาก **ode45** จะเป็น vector x ที่มีค่าเพิ่มขึ้นในช่วงจาก 0 ถึง 10 แต่เนื่องจากใช้วิธี R-K แบบ variable step size จึงไม่สามารถทราบขนาดของ vector ก่อนได้ ส่วน y จะเป็น matrix ที่มี 2 column ซึ่ง column แรกจะเป็นค่า v_1 หรือ y' ส่วน column ที่ 2 จะเป็นค่า v_2 หรือ y นั่นเอง กราฟที่ได้จากการแก้สมการเป็นดังนี้



จากตัวอย่างที่ผ่านมาจะเป็นการแก้สมการ second order อย่างไรก็ตาม สำหรับสมการที่มี order สูงกว่านี้ก็จะมีขั้นตอนเช่นเดียวกัน ซึ่งสามารถสรุปขั้นตอนได้ดังนี้

- เขียนสมการที่มี order สูงให้เป็นระบบสมการ first order
- เขียน function file ที่บรรจุระบบสมการ first order
- ใช้คำสั่ง **ode45** หรือ **ode23** ตามรูปแบบที่กำหนดให้

สำหรับในการใช้ ode function ถ้าไม่มีการกำหนด output ที่แน่ชัด เช่นสั่ง

```
ode23('myode',t,initial)
```

เนื่องจาก output ของ function **ode23** นั้นมีมากกว่า 1 ค่า คือมีทั้ง x และ y ดังนั้น MATLAB จึงไม่สามารถเก็บไว้ในตัวแปรชื่อ **ans** ได้ ทำให้ MATLAB แก้ปัญหาโดยการ plot ค่า (x, y) ให้ โดยอัตโนมัติ

ในเอกสารนี้ไม่ได้กล่าวถึง option อื่นๆ ของ function ode ดังนั้นสำหรับท่านที่ต้องการทราบ option อื่น ๆ ของ function ode สามารถศึกษาได้จากคู่มือหรือใช้ **`help ode23`**

8.3 Boundary Value Problems : Shooting Method

ในการแก้ปัญหา ODE ที่มี order สูงกว่า 1 จำเป็นจะต้องมีค่ากำหนดหรือ condition ให้เท่ากับจำนวน order ของสมการนั้น ที่ผ่านมาได้มีตัวอย่างที่มีค่า condition โดยเป็นการกำหนดค่าที่จุดจุดเดียวซึ่งเราสามารถใช้เป็นจุดเริ่มต้นในการหาคำตอบที่ต้องการหรือที่เรียกปัญหาในลักษณะนี้ว่า initial value

problem อย่างไรก็ตามในความเป็นจริงมีหลายกรณีที่จะมีการกำหนดสภาพที่ขอบเขตหรือ boundary condition มาให้ ซึ่งการพิจารณาในกรณีที่ใช้ analytical method ก็จะไม่แตกต่างกันมากนักไม่ว่าจะเป็น boundary หรือ initial value problems แต่สำหรับการใช้ Runge-Kutta numerical method แล้วลักษณะการแก้ปัญหาของทั้งสองกรณีจะแตกต่างกันอย่างมาก เพราะสำหรับ R-K จำเป็นอย่างยิ่งที่จะต้องกำหนดค่าเริ่มต้นเพื่อให้ R-K จะค่อย ๆ หา solution ไปทีละ step เริ่มจากจุดเริ่มต้น

สำหรับในกรณีของ boundary value problem นั้นสมการอาจจะกำหนดค่าของตัวแปรหรืออนุพันธ์ของมันไว้ที่จุดสองจุด ดังตัวอย่างเช่น

$$y'' = x + \left(1 - \frac{x}{5}\right)y \text{ และกำหนดให้ } y(1) = 2,\ y(3) = -1$$

ซึ่งจากสมการนี้จะเห็นว่ากำหนดค่า y มาให้ที่ x = 1 และ x = 3 ซึ่งเป็นการกำหนดค่าที่ของเขตมาให้ การแก้ปัญหาด้วย R-K สำหรับ ODE นี้นั้นไม่สามารถกระทำได้ เพราะเราไม่ทราบค่าตัวแปรและ derivative ของมันที่ขอบเขตใดเลย

การแก้ปัญหาในกรณีแบบนี้ วิธีหนึ่งคือเราจะเริ่มจากการ **เดา** ค่าเริ่มต้นของ condition ที่ยังขาดไป จากนั้นใช้ R-K แก้สมการในช่วงที่กำหนดแล้วเปรียบเทียบค่าตัวแปรหรืออนุพันธ์ที่กำหนดว่าเท่ากันหรือไม่ หากไม่เท่ากันก็ให้**เดาค่าเริ่มต้นใหม่**แล้วลองพิจารณาค่าเมื่อคำนวณเสร็จสิ้นอีกครั้งหนึ่ง จากการทำสองครั้งแรกเราก็คงจะพอนึกภาพออกว่าค่าที่ถูกต้องควรจะเป็นค่าใด และถ้าหากว่าสมการที่ต้องการหาคำตอบไม่ยุ่งยากมากนักเราก็คงจะได้คำตอบที่ถูกต้องในการลองทำครั้งที่สาม

วิธีการนี้เรียก **shooting method** ซึ่งชื่อของมันได้มาจากวิธีการยิงปืนของทหารปืนใหญ่เพื่อปรับตำแหน่งมุมปืนเพื่อให้ได้ตำแหน่งกระสุนตกตรงที่เป้าหมาย โดยขั้นแรกจะมีการยิงปืนออกไปครั้งแรก 1 นัด จากนั้นดูตำแหน่งกระสุนตกหากไม่เหมาะสมก็จะปรับมุมปืนใหม่แล้วยิงไปอีก 1 นัดแล้วพิจารณาตำแหน่งกระสุนตกอีก หากทำซ้ำเช่นนี้จำนวน 1 ชุดหรือประมาณ 3-5 นัดก็จะสามารถยิงปืนได้เข้าเป้าหมายตามต้องการ

จากสมการที่กำหนดให้หากจะใช้ MATLAB ช่วยในการแก้สมการนี้อาจทำได้ โดยขั้นแรกทำให้สมการ ODE order 2 นี้เป็นระบบสมการ ODE order 1 ก่อน ซึ่งจะได้

```
function dv = myode3(x,v)
dv=zeros(2,1);
dv(1)= x+v(2).*(1-x/5);
dv(2)= v(1);
```

จากนั้นลองใช้ค่า initial condition $y'(1) = -1$ (**เดา 100%**) นั่นคือกำหนดให้

```
init = [-1 2];
```

ส่วนช่วงของการหาค่าตามที่กำหนดคือ 1 ถึง 3

```
intv = [1 3];
```

จากนั้นเรียก **ode45** มาใช้จะได้

```
[x,y]=ode45('myode3',intv,init);
[a b]=size(y);
y(a,:)
ans =
    6.2568    6.2382
```

ซึ่งจะเห็นว่าได้ y(3)=6.2382 ซึ่งสูงกว่าค่าที่กำหนดให้คือ -1 หากมาพิจารณาแล้ว เมื่อเริ่มต้นเรากำหนดให้ slope ของกราฟนี้มีค่าเท่ากับ -1 ที่จุด x = 1 หรือ $y'(1) = -1$ นั่นเอง ซึ่งเมื่อได้ค่าที่สูงเกินไปก็แสดงว่า slop นั้นยังชันไม่พอ เราควรจะเพิ่มความชันเข้าไปอีก คราวนี้ลองสมมุติใช้ $y'(1) = -4$ ดังนั้นเปลี่ยนค่า init เป็น

```
init=[-4 2];
[x y]=ode45('myode3',intv,init);
[a b]=size(y);
y(a,:)
ans =
   -0.6125   -2.4651
```

ซึ่งได้ y(3) = - 2.4651 ซึ่งต่ำกว่าค่าที่กำหนดให้คือ -1 แต่จากสองครั้งที่ผ่านมาทำให้เราทราบว่า ค่าเริ่มต้นที่ควรกำหนดของ y' ควรจะอยู่ระหว่าง -1 และ -4 ในครั้งที่สามเราลองกำหนดโดยการใช้ interpolate ค่าเริ่มต้นและค่าสุดท้ายที่ได้จากการคำนวณทั้งสองครั้ง และค่าที่ต้องการจริงคือ -1 โดยใช้วิธี linear interpolation จะได้

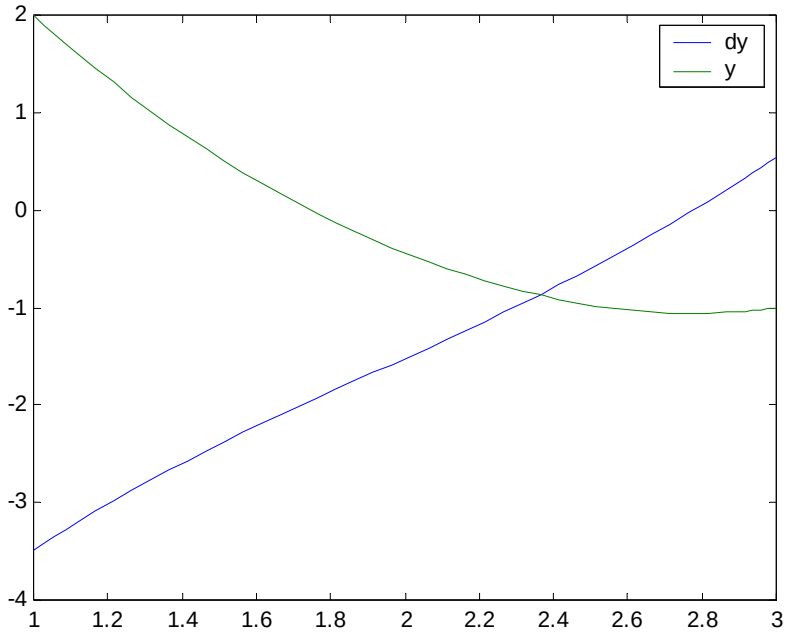
$$-1 + \frac{-4-(-1)}{-2.4651-6.2382}\left(-1-6.2382\right) = -3.495$$

ดังนั้นลองใช้ค่า $y'(1) = -3.495$ ซึ่งได้

```
init=[-3.495 2];
[x y]=ode45('myode3',intv,init);
[a b]=size(y);
y(a,:)
ans =
    0.5439   -1.0000
```

จะเห็นว่าเราได้ y(3) = 1 พอดี ซึ่งกราฟของคำตอบจะมีลักษณะดังนี้

```
m=plot(x,y);
legend(m,'dy','y')
```



จากตัวอย่างที่ผ่านมาเราได้ใช้วิธีการลองสุ่มค่าขึ้นมาสองครั้ง จากนั้นจึงเอาผลของทั้งสองครั้งมาทำ linear interpolation เพื่อหาค่าที่จะใช้เริ่มต้นเป็นครั้งที่ 3 จากนั้นปรากฏว่าในการทำครั้งที่ 3 เราจะได้คำตอบทันที บางท่านอาจคิดว่าที่ได้เป็นความบังเอิญ บางท่านอาจจะคิดว่าเป็นสิ่งที่ควรเป็น สำหรับคำตอบของทั้งสองแนวความคิดนี้คงจะตอบไม่ได้ว่าข้อใดผิดหรือถูกสำหรับ Boundary Value Problems ทั่วๆ ไป แต่ถ้าหากพิจารณาเฉพาะตัวอย่างนี้จะพบว่าไม่ว่าเราจะเริ่มต้นด้วยการเดาค่าเริ่มต้นสองค่าใดๆ ครั้งที่สามท่านจะได้ค่าคำตอบเสมอ สาเหตุก็เพราะ ODE ในตัวอย่างนี้เป็น second order linear ODE

การใช้ shooting method กับ boundary value problem นั้นหากว่า ODE เป็น linear จะสามารถพิสูจน์ได้ว่าผลที่ได้จากการทำ linear interpolation กับค่าเริ่มต้นที่เดาในสองครั้งแรก จะให้ผลเป็นค่าเริ่มต้นที่แท้จริงเสมอ อย่างไรก็ตามผู้เรียบเรียงขอละการพิสูจน์ไว้

❶ ขั้นตอนการใช้ Shooting Method สำหรับ Linear ODE

สำหรับการแก้ปัญหา Boundary Value Problem ของ linear ODE สามารถสรุปได้คร่าวๆ คือการเริ่มเดาค่าเริ่มต้น 2 ค่าเพื่อนำไปหาคำตอบจากนั้นค่าที่จะเลือกครั้งที่ 3 จะได้จากการทำ linear interpolation ของผลที่ได้โดยใช้ขั้นตอนต่อไปนี้

- ให้ G1 เป็นค่าที่เดาครั้งที่ 1
- ให้ G2 เป็นค่าที่เดาครั้งที่ 2
- ให้ R1 เป็นค่าที่ได้จากการเดาครั้งที่ 1 (ใช้ค่า G1)
- ให้ R2 เป็นค่าที่ได้จากการเดาครั้งที่ 2 (ใช้ค่า G2)
- ให้ D คือค่าที่ต้องการ
- ดังนั้นค่าที่ควรจะใช้ในครั้งที่ 3 G คือ

$$G = G1 + \frac{G2 - G1}{R2 - R1}(D - R1)$$

สำหรับ third order ODE ในกรณีของ boundary value problems จะมีด้านหนึ่งของขอบเขตที่กำหนดมาสองค่าและอีกด้านหนึ่งของขอบเขตกำหนดมาค่าเดียว (เพราะรวมกันต้องเท่ากับ order ของ ODE) ดังนั้นวิธีการก็ให้เริ่มจากด้านที่ทราบค่ามากที่สุดก่อน คือให้เริ่มจากด้านที่ทราบ 2 ค่าแล้วเพื่อที่จะเดาเพียงค่าเดียวแล้วใช้วิธีการเหมือนกัน

สำหรับ fourth order ODE ในกรณีที่กำหนดค่าด้านละ 2 ค่าอาจจำเป็นจะต้องใช้วิธีการเลือกว่าจะเริ่มจากด้านใดและต้องจับคู่ใดที่จะ interpolate โดยวิธีการอาจจะยุ่งยากขึ้นแต่การได้มาซึ่งคำตอบก็ยังเป็นหลักการเดียวกัน ส่วน ODE ที่มี order สูงกว่านี้ก็ใช้หลักการเช่นเดียวกัน

❷ Nonlinear ODE

สำหรับ nonlinear ODE แม้ว่าจะไม่สามารถรับประกันได้ว่าจะได้คำตอบภายใน 3 ครั้ง แต่วิธีการพิจารณาค่าคำตอบที่ได้ก็ยังคงสามารถนำไปปรับแต่งเพื่อให้ได้คำตอบตามที่ต้องการได้

ตัวอย่างเช่น ต้องการแก้สมการ

$$y'' = x + \left(1 - \frac{x}{5}\right)yy' \quad \text{โดยกำหนดให้ } y(1)=2 \text{ และ } y(3) = -1$$

สมการที่กำหนดให้เป็น nonlinear เพราะมีเทอมที่มี yy' อยู่ในสมการด้วย ในการแก้ปัญหาด้วย MATLAB เราก็สามารถทำได้เหมือนกับที่ผ่านมา เพียงแต่ในกรณีนี้ต้องทำซ้ำหลายๆ ครั้งดังนั้นอาจสะดวกกว่าที่จะเขียน script file ขึ้นมาเพื่อช่วยในการคำนวณ อย่างไรก็ตามลำดับแรกเราต้องเขียน function file สมมุติให้ชื่อ **myode4.m** ซึ่งมีลักษณะดังนี้

```
function dv = myode4(x,v)
dv = zeros(2,1);
dv(1) = x+ v(1).*v(2).*(1-x/5);
dv(2) = v(1);
```

จากนั้นเขียน script M-file ขึ้นมาเพื่อช่วยในการคำนวณ สมมุติว่าชื่อ **shooting.m** มีลักษณะดังนี้

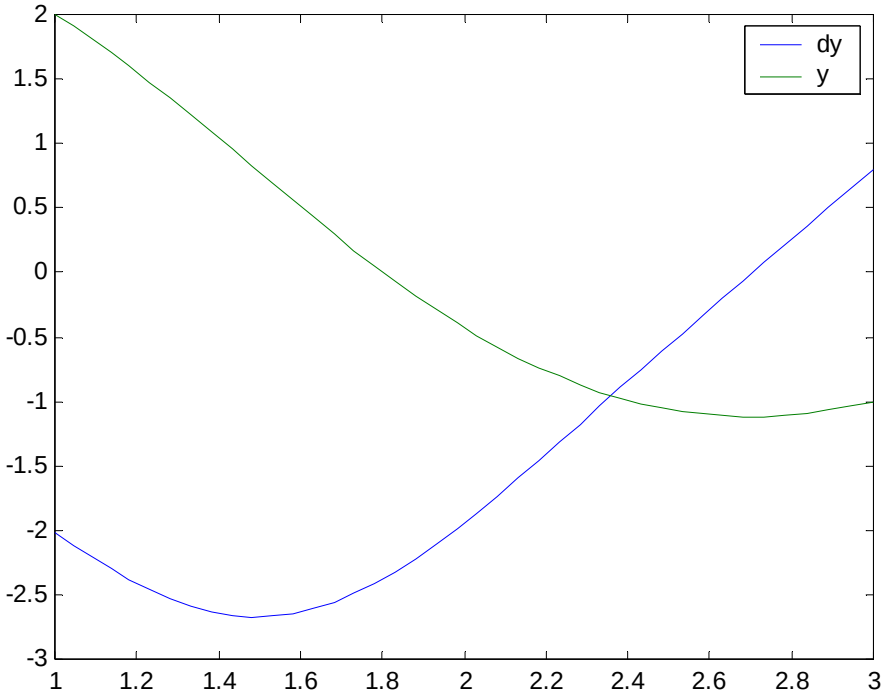
```
%Shooting Method for Nonlinear ODE
D = 1;
Intv = [1 3];

% First Try
G1 = -1;                                  % เดาครั้งที่ 1
init=[G1 2];
[x,y]=ode45('myode4',intv,init);
[a b]=size(y);
```

```
R1=y(a,2);                                  % ผลจากการเดาครั้งที่ 1

% Second Try
G2=-5;                                      % เดาครั้งที่ 2
init=[G2 2];
[x,y]=ode45('myode4',intv,init);
[a b]=size(y);
R2=y(a,2);                                  % ผลจากการเดาครั้งที่ 2

while abs(y(a,2)-D)>1e-6          %       ให้เครื่องทำไปเรื่อยๆจนกระทั่งผิดพลาดน้อยกว่า
1E-6
      G=G1+(G2-G1)*(D-R1)/(R2-R1);      % หาค่าประมาณครั้งต่อไป
      init=[G 2];
      G1=G2;                                % สลับค่า ให้การเดาเก่าลง 1 ครั้ง
      G2=G;                                 % ค่าที่เดาที่ใหม่สุด
    [x,y]=ode45('myode4',intv,init);
  [a b]=size(y);
      R1=R2;                                % สลับค่า ให้ค่าที่ได้เก่าลง 1 ครั้ง
      R2=y(a,2);                            %    ค่าที่ได้ที่ใหม่ที่สุด
      P=abs(y(a,2)-D);                      %  บอกผลให้เรารู้ว่าเครื่องติด loop หรือไม่
      fprintf('Now your error is %10.6f \n',P)
end                       % เมื่อออกจาก loop ได้แสดงว่าคำนวณได้เรียบร้อยแล้ว ให้ plot ค่าต่อไป
%
m=plot(x,y);
legend(m,'dy','y')

% ********************************************** END *******
```

และจากการ run file shooting จะได้

```
» shooting
Now your error is    1.207074
Now your error is    2.396240
Now your error is    0.568361
Now your error is    0.249510
Now your error is    0.045735
Now your error is    0.003186
Now your error is    0.000039
Now your error is    0.000000
```

เมื่อ MATLAB ทำงานจนกระทั่งค่า error ที่เกิดขึ้นระหว่างค่าที่กำหนดให้และค่าที่ได้จากการคำนวณมีค่าน้อยกว่า 10^{-6} ซึ่งเป็นค่าที่เรากำหนดขึ้น MATLAB จะหยุด และจะเขียนกราฟซึ่งจะได้กราฟมีลักษณะดังนี้



สำหรับวิธีการแก้ปัญหา Boundary Value Problems นอกจากจะใช้ Shooting Method แล้วยังมีวิธีการอื่นอีกหลายวิธี อย่างไรก็ตามวิธีการที่นิยมใช้อีกวิธีหนึ่งคือ Finite Difference Method ซึ่งเป็นวิธีที่มีประสิทธิภาพสูงอีกวิธีหนึ่งแต่เพราะวิธีนี้มีรายละเอียดที่หลากหลายมากจึงจะไม่ขอกล่าวไว้ในที่นี้ ผู้ที่สนใจสามารถที่จะศึกษาได้จากหนังสือที่เกี่ยวข้องกับ Numerical Method ได้

✍ *การแก้ปัญหาทางวิศวกรรม*

- ***การหาคำตอบที่แน่นอนของการไหลที่ชั้นชิดผิว*** (The Excact Solution of Boundary Layer Theory)

สำหรับปัญหาหนึ่งทางด้านกลศาสตร์ของไหลก็คือการหาความเร็วและลักษณะการไหลของของไหลบริเวณชั้นชิดผิว (Boundary Layer) ซึ่งการไหลในลักษณะดังกล่าวความหนืดจะมีผลต่อสภาพการไหลเป็นอย่างมาก ในปัจจุบันมีวิธีการหลายวิธีที่จะทำการหาคำตอบของการไหลของอากาศหรือของไหลอื่นผ่านรูปทรงต่างๆ มากมาย แต่ความสำคัญของทฤษฎีชั้นชิดผิวไม่ได้ลดลงไปเลยแม้ว่าชั้นชิดผิวนี้จะเป็นเพียงชั้นบางๆ ที่มีความสำคัญเฉพาะบริเวณใกล้กับผิวของวัตถุแต่ก็เป็นสิ่งที่ทำให้เกิดแรงเสียดทาน การแยกตัวของการไหลและอื่นๆ อีกมาก

การหาคำตอบที่แน่นอนของการไหลในชั้นชิดผิวที่สามารถหาคำตอบที่แน่นอนได้จะเป็นการไหลของของไหลผ่านระนาบเรียบซึ่งไม่มีความแตกต่างของความดัน สำหรับการหาคำตอบของการไหลในลักษณะนี้จะเริ่มจากสมการการไหลเต็มรูปคือ Navier-Stokes Equation จากนั้นจะเป็นการลดรูปสมการโดยการใช้วิธีการต่างๆ หลายประการ ท้ายที่สุดเราจะได้สมการการไหลอยู่ในรูป

$$2f''' + ff'' = 0$$

สมการนี้รู้จักกันดีในชื่อ Blasius' Equation ซึ่งตั้งตามชื่อของ H. Blasius ซึ่งเป็นผู้ที่หาคำตอบของการไหลในลักษณะนี้เป็นคนแรกในงานวิทยานิพนธ์ระดับดุษฎีบัณฑิตในปี 1908 โดยมีรากฐานมาจากอาจารย์ของเขาคือ Prandtl ผู้ที่กล่าวถึงการไหลในชั้นชิดผิวเป็นครั้งแรกในปี 1904 โดยค่าตัวแปรในสมการมีความสัมพันธ์กับความเร็วคือ

$$\frac{u}{U} = f'(\eta)$$

เมื่อ u คือความเร็วของการไหลในทิศทางของระนาบเรียบ

U คือความเร็วของของไหลอิสระเมื่ออยู่นอกขอบเขตของชั้นชิดผิวซึ่งมีค่าคงที่

และ $f' = \frac{df(\eta)}{d\eta}$ โดย

$$\eta = y\sqrt{\frac{U}{\nu x}}$$

เมื่อ ν คือ kinematic viscosity ของของไหลซึ่งสามารถหาค่าได้จากการทดลอง

x เป็นความยาวตามแนวระนาบ

y เป็นความยาวในแนวที่วัดตั้งฉากกับระนาบเรียบ

สิ่งที่เราสนใจมากที่สุดเกี่ยวกับการไหลในชั้นชิดผิวก็คือความหนาของชั้นชิดผิวว่าจะมีความหนามากน้อยเพียงใด เพราะเมื่ออยู่นอกความหนาของชั้นชิดผิวนี้ไปแล้วความหนืดจะไม่มีผลกระทบต่อการไหล ทำให้เราสามารถใช้ทฤษฎีของ potential flow แก้ปัญหาได้ ซึ่งมีความง่ายในการแก้ปัญหามากกว่า viscous flow มาก สำหรับ Blasius' Equation มี Boundary condition คือ

⇨ ความเร็วของของไหลที่ผิววัตถุมีค่าเป็นศูนย์ :

$$\eta = 0 \rightarrow f' = 0 \text{ และ } \eta = 0 \rightarrow f = 0$$

⇨ ที่ห่างจากวัตถุไปมากๆความเร็วของของไหลจะเท่ากับความเร็วของกระแสการไหลอิสระ :

$$\eta \rightarrow \infty,\ f' = 1$$

Blasius' Equation เป็น third order ordinary differential equation boundary value problem ซึ่งเราสามารถใช้ Runge - Kutta Method ร่วมกับ Shooting Method ที่กล่าวมาแล้วในบทนี้แก้ปัญหาได้

สำหรับ M-file ที่ใช้ใน shooting method มีลักษณะเหมือนกับที่ยกเป็นตัวอย่างที่ผ่านมา สำหรับ M-file ของ integrand จะมีลักษณะอยู่ในรูปของ system of first order ODE ดังนี้

```
function xdot = blas(t,x)
xdot = zeros(3,1);
xdot(1) = -x(3).*x(1)/2;
xdot(2) = x(1);
xdot(3) = x(2);
```

สำหรับค่าที่ต้องสมมุติในการใช้โปรแกรมนี้ก็คือ infinity จะอยู่ที่ระยะห่างเท่าใดจากผิววัตถุ ในโปรแกรมเราอาจเริ่มจากค่า infinity มากน้อยเท่าใดก็ได้ แต่เมื่อได้คำตอบเราอาจสามารถบอกได้ว่าค่า f' นี้เริ่มคงตัวเมื่อใด จุดที่ค่านี้เริ่มคงตัวเท่ากับ 1 ก็คือจุดที่พ้นชั้นชิดผิวออกไปแล้วนั่นเอง สำหรับ program ที่ใช้ใน shooting method และสมมุติว่า $\eta \to 6$, $f' = 1$ ในกรณีนี้ M-file จะมีลักษณะดังนี้คือ

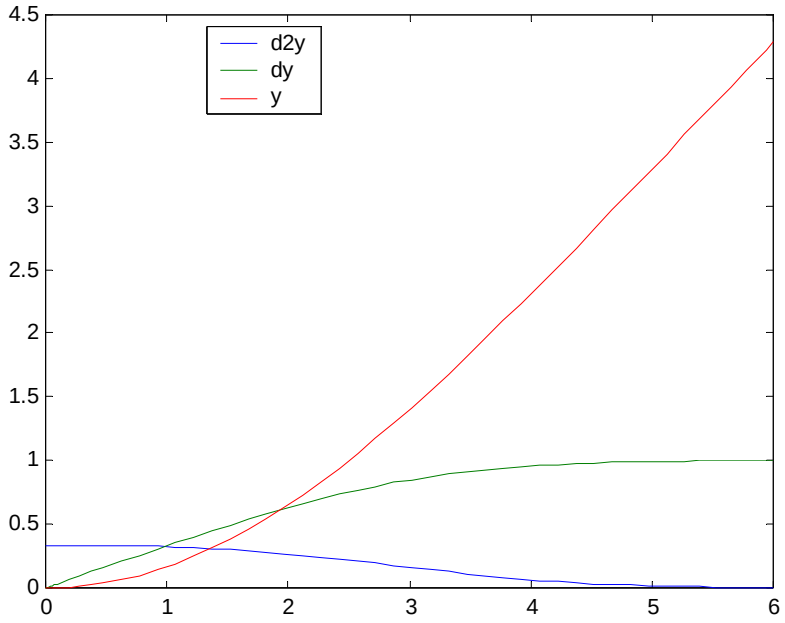
```
D=1;
Intv=[0 6];

% *************
G1=1;
init=[G1 0 0];
[x,y]=ode45('blas',intv,init);
[a,b]=size(y);
R1=y(a,2);
G2=2;
init=[G2 0 0];
[x,y]=ode45('blas',intv,init);
[a,b]=size(y);
R2=y(a,2);

% **************
while abs(y(a,2)-D)>1e-6
   G=G1+(G2-G1)*(D-R1)/(R2-R1);
   init=[G 0 0];
   G1=G2;
   G2=G;
   [x,y]=ode45('blas',intv,init);
   [a,b]=size(y);
   R1=R2;
   R2=y(a,2);
   P=abs(y(a,2)-D);
   fprintf('Now your error is %10.6f \n',P)
end
m=plot(x,y);
legend(m,'d2y','dy','y',0)
```

จะได้ผลเป็น

```
Now your error is    0.526004
Now your error is    0.249148
```

```
Now your error is    0.037440
Now your error is    0.002332
Now your error is    0.000023
Now your error is    0.000000
```

สำหรับกราฟที่ได้จะมีลักษณะดังนี้



สำหรับค่าความหนาของชั้นชิดผิวจะพิจารณาจากค่าความเร็วของการไหล โดยความหนาของชั้นชิดผิวจะเท่ากับจุดที่ซึ่งความเร็วของของไหลมีค่าเท่ากับ 99% ของความเร็วอิสระหรือ

$$\delta = y \ @ \ u = 0.99U$$

จากการพิจารณาค่าที่ได้จากการคำนวณ เนื่องจากที่ $\frac{u}{U} = f'(\eta)$ ดังนั้นจุดที่ $f' = 0.99$ จะหาค่า η ได้จากการทำ interpolation ของข้อมูลที่ได้ ดังนั้นจากการ run M-file shooting จะได้

```
shooting
Now your error is    0.514670
Now your error is    0.238315
Now your error is    0.033354
Now your error is    0.001918
Now your error is    0.000016
Now your error is    0.000000
```

หลังจากการแก้สมการ ODE ด้วย MATLAB แล้ว เราสามารถที่จะหาค่าความหนาชั้นชิดผิวได้ดังนี้

```
a=y(:,2);
b=interp1(a,x,0.991,'spline')
b =
    4.9654
```

นั่นคือ $$\delta = 4.9654\sqrt{\frac{U}{\nu x}}$$

หรือเขียนในรูปของ Reynold's Number, $\mathrm{Re}_x = \frac{\rho U x}{\mu} = \frac{Ux}{\nu}$ จะได้

$$\delta = \frac{4.965x}{\sqrt{\mathrm{Re}_x}}$$

ซึ่งก็จะใกล้เคียงกับการหาค่าความหนาของชั้นขอบผิวด้วยวิธีอื่นๆ

บทที่ 9 MATLAB GRAPHICS

ในบทที่ 2 เราได้ทราบถึงวิธีการ plot เบื้องต้นด้วย MATLAB ไปแล้วในบทนี้จะเป็นการกล่าวถึงการ plot ในลักษณะต่างๆ ที่นอกเหนือจากการ plot แบบ x, y ธรรมดาที่กล่าวถึงในบทที่ 2 และได้ใช้ในการแสดงผลประกอบการอธิบายในบทที่ผ่านมา

9.1 การ plot ใน 2 มิติ

การ plot ในสองมิติได้มีการกล่าวถึงไปบ้างแล้ว สำหรับ x, y plot แต่มีส่วนประกอบบางอย่าง ที่ยังไม่ได้กล่าวถึงในส่วนที่ผ่านมาซึ่งอาจมีประโยชน์ในการใช้งานบางประเภท ซึ่งได้แก่

❶ Error Bar

การใส่ error bar จะมีประโยชน์ในการวิเคราะห์ข้อมูลที่ได้จากการทดลอง ซึ่งมีคำสั่งต่อไปนี้

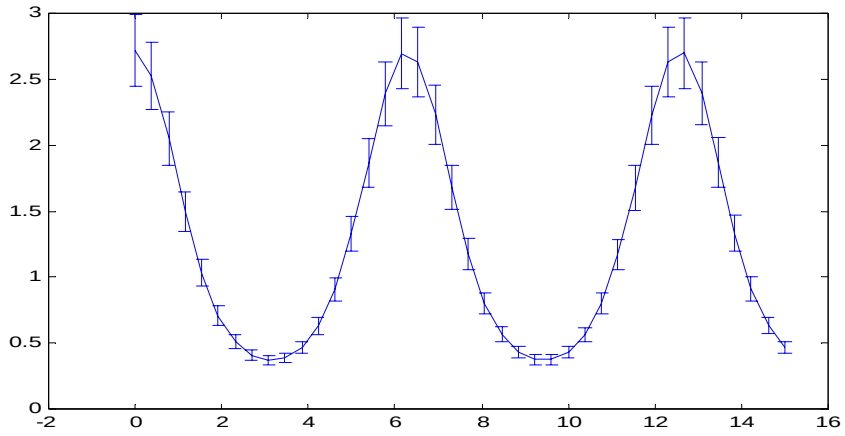
`errorbar(x,y,e,'str')`
เป็นคำสั่ง plot(x,y) แล้วให้มี error bar ขนาดเท่ากับ e วางสมมาตรอยู่ทุกจุด coordinate ที่มีการ plot x-y ส่วน 'str' เป็น option ซึ่งบอก character string ของสีและลักษณะเส้นกราฟ x,y ที่ใช้

`errorbar(x,y,l,u)` คล้ายกับคำสั่งข้างบน แต่จะวาง error bar ที่จุด (x_i, y_i) ที่อยู่สูงกว่า y_i เป็นระยะ l_i และต่ำกว่า y_i เป็นระยะ u_i

ตัวอย่างเช่น

```
x  = linspace (0,15,40);
y  = exp(cos(x)) ;
delta = 0.1*y ;
errorbar (x,y, delta)
```

ซึ่งจะได้ผลเป็นดังรูปข้างล่างนี้



❷ Graph Function

ถ้าหากว่าเรามี function file อยู่แล้วต้องการจะ plot function นั้นโดยตรง ไม่ว่าจะเป็น Function ที่มีอยู่ใน MATLAB หรือเขียนขึ้นเอง ซึ่งสามารถใช้คำสั่ง fplot ดังนี้

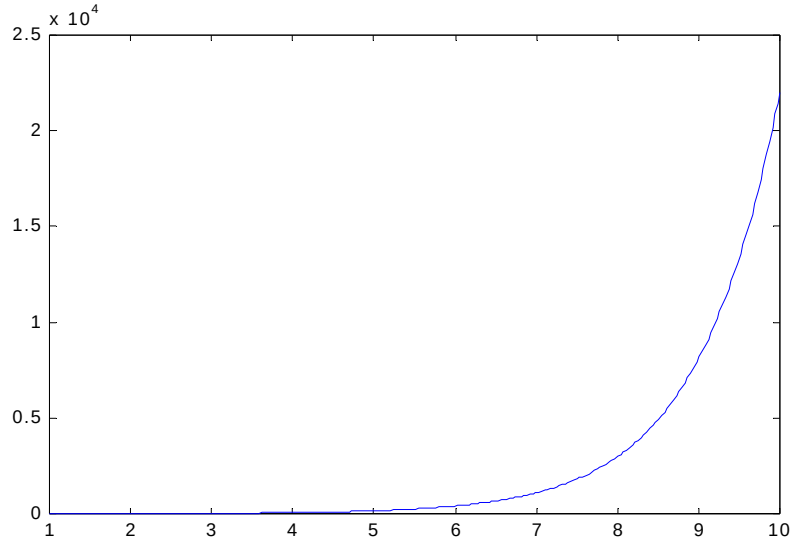
fplot ('function', lim, str, tot)

เป็นการ plot function ชื่อ function (อยู่ใน function file ชื่อ function.m) ส่วน lim เป็น vector ที่กำหนดค่า Limit ของ function ที่จะ plot แบ่งเป็น 2 แบบคือ lim = $[x_{min}\ x_{max}]$ เป็นการกำหนดค่า minimum และ maximum ของแกน x เท่านั้น lim = $[x_{min}\ x_{max}\ y_{min}\ y_{max}]$ กำหนดค่าทั้งสองแกน ส่วน option str เป็นการกำหนดลักษณะเส้น แต่ละค่า tot เป็นการกำหนด relative error ให้น้อยกว่าค่า tot

ตัวอย่างเช่น

```
lim =[1 0];
fplot('exp(x)',lim)
```

จะได้ผลเป็นดังรูป



❸ Polar Plot

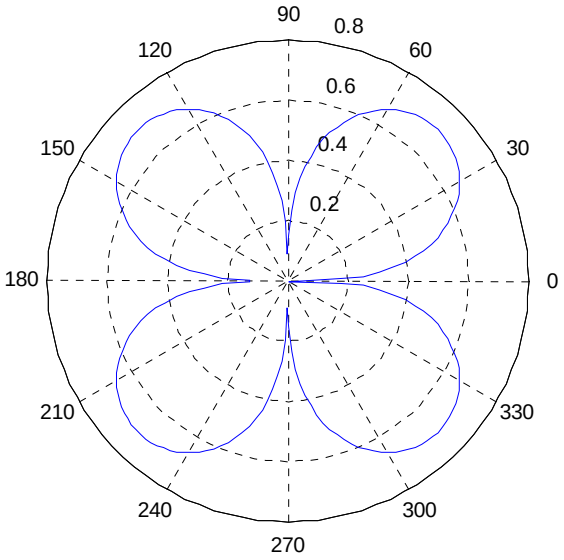
หากต้องการ plot กราฟในลักษณะของ polar coordinate ก็สามารถทำได้โดยใช้คำสั่ง polar

polar(q,r) จะเป็นการ plot ลงบน polar coordinate โดย q vector ของมุม q ในหน่วย radian เทียบกับรัศมีที่มีขนาดเท่ากับ vector r

ตัวอย่างเช่น

```
theta = linspace(0,2*pi,200);
r = sqrt(abs(sin(theta)).*abs(cos(theta)));
polar(theta,r)
```

ซึ่งจะได้ผลตามรูป ส่วน option string เช่นการกำหนดสีเส้น, ลักษณะของเส้นก็สามารถใช้ได้ตามปกติเหมือน plot ทั่วไป



❹ การ Plot บน Complex Plane

สำหรับ complex number z = x + iy สามารถ plot ลงบน complex plane ได้ด้วยคำสั่งต่อไปนี้ โดยกำหนดให้ X, Y, Z เป็น matrix และ x,y เป็น vector

quiver(X,Y)	เขียนลูกศรของแต่ละคู่ด้วย coordinate x_{ij}, y_{ij}
quiver(X,Y,dx,dy)	เขียนลูกศรที่ coordinate x_i, y_i ให้มี argument dx และ magnitude dy
quiver(X,Y,Dx,Dy)	เขียนลูกศรที่ coordinate x_{ij}, y_{ij} ให้มี argument และ magnitude เป็น dx_{ij}, dy_{ij} ตามลำดับ
feather(Z,str)	เขียนลูกศรแสดง magnitude และ argument ของแต่ละ element ของ complex matrix Z โดยมีจุดกำเนิดอยู่ที่กึ่งกลางแกน x ส่วน option string 'str' เป็น option ที่ใช้กำหนดสีและลักษณะเส้นตามที่เคยกล่าวไว้แล้ว
feather(x,y)	จะเหมือนกับการใช้ feather (x+y * i) โดย $i = \sqrt{-1}$
compass(z,str)	เขียนลูกศรจากจุดกำเนิด (0,0) แสดง magnitude และ argument ของ element ของ complex matrix Z ส่วน str เป็น option บอกลักษณะเส้นและสี
compass(x, y)	จะเหมือนกับการใช้ compass (x+y*i) โดย $i = \sqrt{-1}$
rose(v,n)	เขียน argument histogram แสดง frequency ของ arguments ของ vector v ส่วน n เป็น option ของจำนวนช่วงที่ใช้ ถ้าไม่กำหนด MATLAB จะใช้ n = 36

สำหรับตัวอย่างการ plot ใน complex plane กำหนดให้

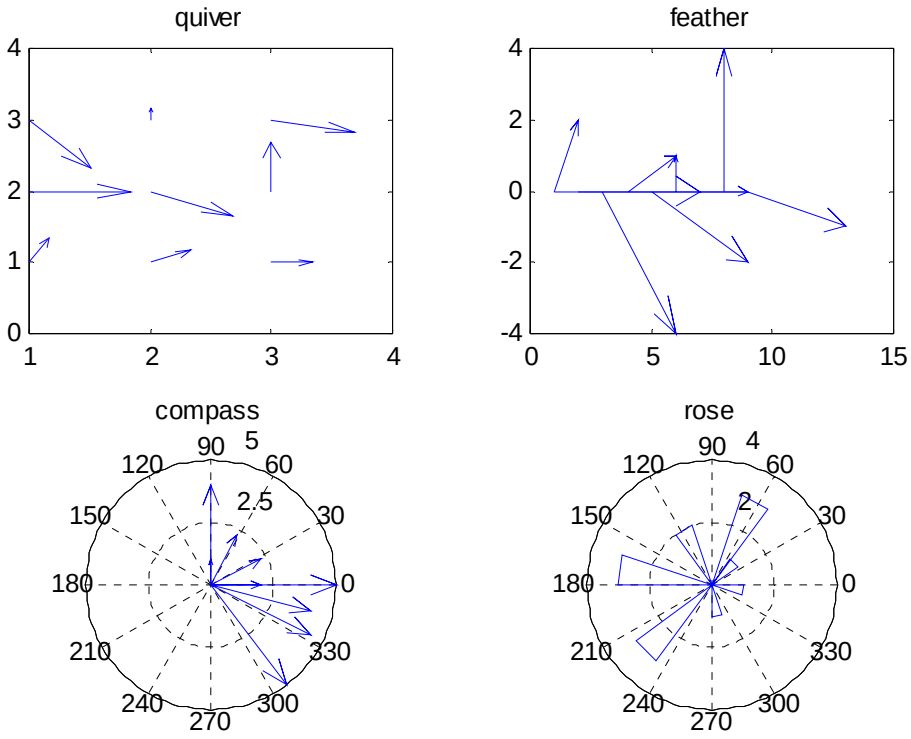
```
z =
  1.0000+2.0000i   2.0000+1.0000i   2.0000
  5.0000           4.0000-2.0000i   0+4.0000i
  3.0000-4.0000i   0+1.0000i        4.0000-1.0000i
```

และใช้คำสั่งต่อไปนี้

```
subplot(2,2,1)
quiver(real(z),imag(z))
title('quiver')

subplot(2,2,2)
feather(real(z),imag(z))
title('feather')

subplot(2,2,3)
compass(real(z),imag(z))
title('compass')
x=[1 2 1 3 4 5 3 2 3 4 4 1 6 7];
```

```
subplot(2,2,4)
rose(x)
title('rose')
```

ซึ่งตัวอย่างการ plot และผลเป็นไปตามรูปนี้



❺ Histogram และ Bar Graphs

Histogram เป็นกราฟที่นิยมใช้เพื่อพิจารณาการกระจายของข้อมูลในทางสถิติ โดยเป็นการนับจำนวนความถี่ของข้อมูลที่มีอยู่ คำสั่งของ MATLAB ที่ใช้ในการเขียน histogram มีดังนี้

his(x,n) จะเขียน histogram ของข้อมูลใน vector x ส่วน n เป็น option เป็นจำนวนกราฟที่ต้องการใช้แสดงข้อมูล ถ้าไม่มีการกำหนด MATLAB จะใช้ n = 10

สำหรับ bar graph เป็นกราฟที่นิยมใช้ในการแสดงผลเปรียบเทียบข้อมูล คำสั่งที่ใช้มีดังนี้

bar(x) เขียน bar graph ของ x

bar(x,z) เขียน bar graph ของ x กำหนดตำแหน่งโดย vector z ซึ่งมีการเรียงลำดับจากน้อยไปหามาก

stairs(x) เขียน stairs graph ของ x (stairs graph คือ bar graph ที่ไม่มีเส้นด้านข้าง)

stairs(x,z) เขียน stairs graph ของ x กำหนดตำแหน่งโดย vector z ซึ่งต้องมีการเรียงลำดับจากน้อยไปหามาก

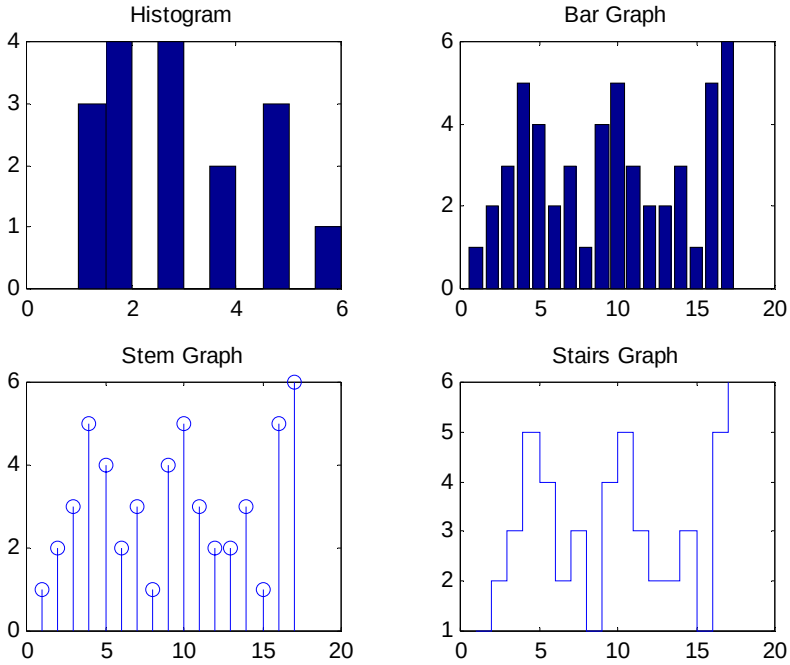
stem(x)	เขียน stem graph ของ x โดย stem graph เป็นกราฟที่คล้าย bar graph ที่ไม่มีความหนา จะเป็นเส้นตรงเส้นเดียวและที่จุดสิ้นสุดของเส้นจะมีวงกลมเขียนอยู่
stem(x,z)	เขียน stem graph ของ x โดยมี vector z เป็นตัวกำหนดตำแหน่งซึ่งจะเรียงจากน้อยไปหามาก

สำหรับตัวอย่างของการเขียน bar graph และ histogram เป็นดังนี้

```
x=[1 2 3 5 4 2 3 1 4 5 3 2 2 3 1 5 6];
```

จากค่าของ x ที่กำหนดให้นี้สามารถที่จะนำไปสร้างเป็นกราฟต่างๆ ได้ดังต่อไปนี้

```
subplot(2,2,1)
hist(x)
title('Histogram')
subplot(2,2,2)
bar(x)
title('Bar Graph')
subplot(2,2,3)
stem(x)
title('Stem Graph')
subplot(2,2,4)
stairs(x)
title('Stairs Graph')
```



❻ Comet Plot

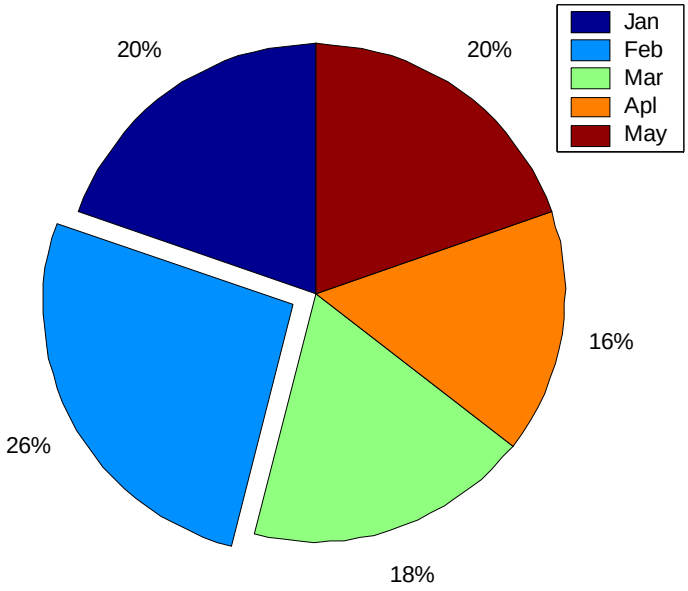
Comet plot จะให้ผลเหมือนกับการสั่ง Plot หากแต่ว่าการแสดงผลจะค่อย ๆ แสดงผลและมีจุดวงกลมค่อย ๆ เคลื่อนที่ไปตามแนวที่จะ plot โดยจะทิ้งเส้นที่วิ่งผ่านไปแล้วไว้ คล้าย ๆ กับหางของดาวหางนั่นเอง คำสั่งที่ใช้คือ

comet(x,y)	จะเขียนกราฟของ vector x และ y โดยวิธีค่อย ๆ ลากเส้น
pie(x)	จะเป็นการเขียนกราฟวงกลมของ vector x โดยจะแบ่งออกเป็นชิ้นส่วนย่อยตามร้อยละของแต่ละ element เมื่อเทียบกับผลรวมของทุก element ของ vector x
pie(x,y)	จะเป็นการเขียนกราฟวงกลม vector x โดยจะแบ่งออกเป็นชิ้นส่วนตามจำนวน element ของ x ส่วน vector y จะมีขนาดเท่ากับ x โดยจะมีค่าเป็น 0 หรือ 1 หาก element ใดของ y มีค่าเป็น 1 ใน element ตำแหน่งเดียวกันนั้นของ x เมื่อเขียนกราฟจะถูกแยกออกมา
pie(x,condition)	จะเหมือนข้างบนแต่จะเป็นการใช้ condition กำหนดว่าจะให้แยกชิ้นกราฟใดออกมา

สำหรับ comet plot นั้นถ้าหากเห็นภาพสุดท้ายแล้วจะพบว่าไม่ต่างจากการใช้คำสั่ง plot เพราะข้อแตกต่างจะอยู่ในขั้นตอนการนำเสนอรูปกราฟ จึงไม่ขอยกตัวอย่างในที่นี้ ส่วน pie3(x) นั้นจะมีความหมายเหมือนกับ pie(x) เพียงแต่เป็นการเขียนกราฟให้มีมุมมองเป็น 3 มิติเท่านั้น ตัวอย่างเช่น

```
x=[15 20 14 12 15]';
y=[0 1 0 0 0]';
txt=['Jan'; 'Feb';'Mar';'Apl';'May'];
pie(x,y);
legend(txt)
```

จะได้กราฟเป็น



9.2 การเขียนกราฟใน 3 มิติ

สิ่งหนึ่งที่ทำให้ MATLAB ได้รับความนิยมก็เพราะความสามารถในการเขียนกราฟใน 3 มิติ ของ MATLAB เพราะเราสามารถใช้คำสั่งง่ายๆ เพื่อที่จะสร้างภาพ 3 มิติได้หลายรูปแบบ ไม่ว่าจะเป็นลักษณะของลายเส้น (mesh) หรือจะเป็นรูปร่าง อีกทั้งยังสามารถกำหนดแสง สี และมุมมองภาพ 3 มิติ เหล่านั้นได้อย่างง่ายดาย การ plot ใน 3 มิติของ MATLAB มีคำสั่งที่แบ่งเป็นกลุ่ม ๆ ได้ต่อไปนี้

❶ การ Plot จุดใน 3 มิติ

หากต้องการจะ plot จุด coordinate (x,y,z) ลงในระบบแกน Cartesian Coordinate สามารถใช้คำสั่ง **plot3** โดยมีรูปแบบต่อไปนี้

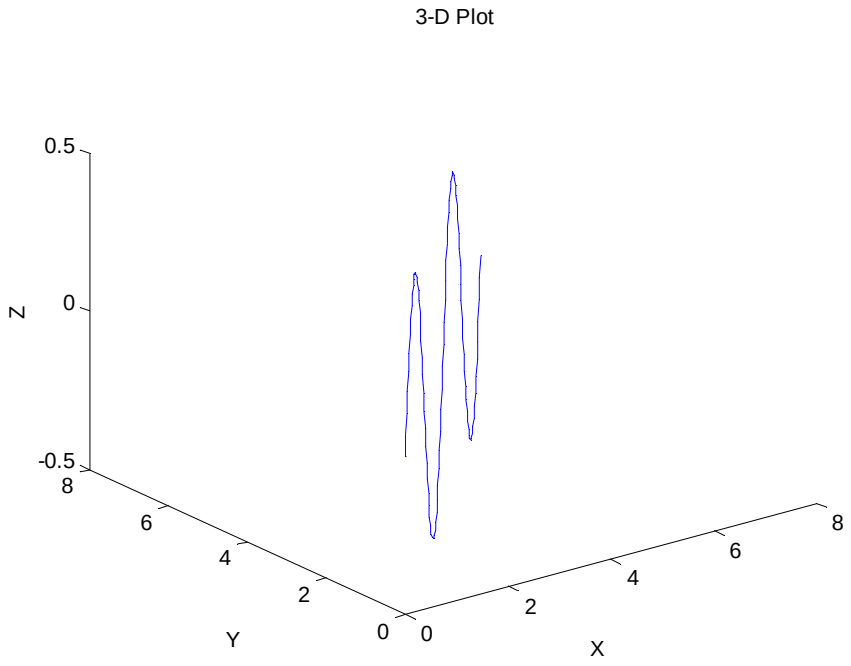
Plot3(x,y,z) โดย x, y ,z เป็น vector ที่มีขนาดเท่ากันจะเป็นการ plot coordinate (x_i, y_i, z_i) ลงบนแกน ถ้าหากว่า parameter x, y, z มีค่าใดเป็น Matrix แล้ว MATLAB จะ plot ตาม column ของ parameter นั้น ซ้ำไปเรื่อยๆ กับ vector ที่เหลือจนครบทุก column

Plot3(x,y,z,'str') จะเหมือนกับข้างบน หากแต่ 'str' จะเป็นเครื่องกำหนดลักษณะเส้นและสีที่ plot

นอกจากนี้คำสั่ง Title, xlabel, ylabel, zlabel, text ยังสามารถใช้ได้ตามปกติเหมือนในคำสั่ง plot ใน 2 มิติ สมมุติเราต้องการเขียนกราฟ z = sin (x) - cos (y) โดย x และ y มีค่าอยู่ระหว่าง 0 ถึง 2π จะสามารถเขียนได้ดังนี้

```
x  =  linspace (0, 2*pi, 100) ;
y  =  x ;
z  =  sin (x).* cos (y) ;

plot3(x,y,z)
xlabel('X')
ylabel('Y')
zlabel('Z')
title(' 3-D Plot')
```

ซึ่งจะได้ผลเป็นตามรูป



❷ การเขียนเส้นพื้นผิวใน 3 มิติ

การเขียนพื้นผิวใน 3 มิติ จะเป็นการเขียนกราฟของ

$$z = f(x,y)$$

ซึ่งจะต่างกับการ plot เป็นจุดที่ใช้ในคำสั่ง plot3 เพราะในการเขียนจุดเราจะกำหนดว่า $z_i = f(x_i, y_i)$ แต่ในการเขียนพื้นผิวเราจะกำหนดในลักษณะ

$$z_{ij} = f(x_i,y_j)$$

นั่นคือ ถ้า x เป็น vector ความยาว m และ y เป็น vector ความยาว n ในการเขียนพื้นผิวเราจะมีจุดที่ต้อง plot ทั้งหมด m x n แต่ใน plot3 เราจะได้ว่า m ต้องเท่ากับ n และจะ plot เป็นจำนวน m จุดเท่านั้น

ก่อนที่จะมีการเขียนพื้นผิวเราอาจจะต้องสร้าง grid ขึ้นมาก่อน โดย grid ที่สร้างขึ้นก็คือระบบแกนที่เราจะใช้ plot นั่นเอง คำสั่งสร้าง grid มีดังนี้

[u,v] = meshgrid(x,y) โดย x เป็น vector ความยาว m และ y เป็น vector ความยาว n ทั้งคู่เป็น vector ที่มีการเรียงค่าจากน้อยไปหามากและจะให้ผลเป็น matrix u และ v ซึ่งมีขนาดเป็น n x m ซึ่งเป็น matrix ที่กำหนด x และ y coordinate ใน rectangular domain โดย coordinate (u_{ij}, v_{ij}) คู่หนึ่งจะเป็นจุดที่ใช้หา $z_{ij} = f(u_{ij},v_{ij})$ นั่นเอง

[x,y,z] = cylinder(r,n) จะเป็นการสร้าง grid ใน cylindrical coordinate โดย vector r เป็นรัศมีแต่ละจุดที่มีระยะห่างเท่ากันตามระยะสูง นั่นคือ ถ้า r คงที่จะได้ coordinate ของรูปทรงกระบอก แต่ถ้า r เปลี่ยนไปตามระยะสูงจะได้รูปกรวย ส่วน n เป็นจุดที่แบ่งตามแนวเส้นรอบวง ถ้าไม่มีการกำหนด จะใช้ n = 20 และหากไม่กำหนดค่า r จะใช้ r = [1 1] เป็นค่าตั้งต้น

[x,y,z] = sphere(n) สร้าง spherical grid แล้วให้ค่า matrix x, y, z คืนมาโดย x, y และ z จะมีขนาด (n+1) * (n-1)

แม้ว่าการสร้าง grid อาจไม่จำเป็นในทุกกรณีการเขียนพื้นผิวโดยใช้ function ของ MATLAB แต่ที่แนะนำให้เห็นในที่นี้ก็คือจะเป็นการแสดงถึงลักษณะของ function ที่เราจะทำการเขียนพื้นผิว โดยลักษณะการสร้าง grid นั้นสามารถยกตัวอย่างง่าย ๆ ได้เช่น ถ้า x = [1 2 3] และ y = [4 5] หากเราจะ plot พื้นผิวเราจะต้องใช้ coordinate ของ z เป็น $z_{ij} = f(x_i,y_j)$ โดย (x_i,y_j) ในที่นี้จะเป็น (1,4), (2,4), (3,4), (1,5), (2,5), (3,5) ตามลำดับ ดังนั้นในการหา meshgrid โดยใช้คำสั่ง [u v] = meshgrid (x,y) เราจะได้

```
x=[1 2 3];
y=[4 5];
[u v]=meshgrid(x,y)
u =
     1     2     3
     1     2     3
v =
     4     4     4
     5     5     5
```

ซึ่งจะตรงกับค่า $z_{ij} = f(x_{ij},y_{ij})$

❸ การเขียนผิวลายเส้น

การเขียนผิวลายเส้นสามารถทำได้โดยใช้คำสั่ง mesh, meshc, meshz และ waterfall โดยมีลักษณะคำสั่งต่อไปนี้

mesh(X,Y,Z,C)	จะเป็นการ plot จุด (x_{ij}, y_{ij}, z_{ij}) ของ matrix X,Y,Z ที่มีขนาด m x n และเชื่อมต่อจุดที่อยู่ติดกันเข้าด้วยกัน ส่วน C เป็น matrix ขนาดเท่ากันซึ่งจะเป็นการกำหนดสีของแต่ละจุด โดย C เป็น option หากไม่ใช้ MATLAB จะกำหนดให้ C = Z
mesh(x,y,Z,C)	ถ้ากำหนด x เป็น vector ความยาว m และ y เป็น vector ความยาว n จะได้ว่า Z ต้องเป็น matrix ขนาด m x n โดย MATLAB จะทำการ plot (x_i, y_j, z_{ij}) นั่นคือ MATLAB จะสร้าง meshgrid ให้โดยอัตโนมัติ จากvector x และ y ส่วน C เป็น option เหมือนคำสั่งที่ผ่านมา
mesh(Z)	เหมือนคำสั่งข้างบน แต่เป็นการ plot (i, j, z_{ij}) โดย z ต้องเป็น matrix
meshc(...)	เหมือนคำสั่ง mesh (...) แต่จะมีการเขียน contour curve ใต้ภาพ
meshz(...)	เหมือนคำสั่ง mesh (...) แต่จะมีการเขียน grid ตามแกน x และ y ขนานแกน z
waterfall(...)	เหมือนคำสั่ง meshz(...) แต่จะมีการเขียน grid ตามแกนเดียว
hidden on	สั่งให้ MATLAB เขียน mesh โดยไม่ต้องเขียนเส้นที่ถูกบังไว้ด้านหลัง
hidden off	สั่งให้ MATLAB เขียน mesh โดยเขียนเส้นที่ถูกบังไว้ด้านหลังด้วย
hidden	สลับระหว่าง hidden on และ off โดยถ้าไม่มีคำสั่งมาก่อน MATLAB จะตั้งค่าไว้ที่ hidden on

สำหรับตัวอย่างการเขียน mesh เราจะทดลองเขียนผิวของ function เหมือนตัวอย่างที่ผ่านมาคือกำหนดให้

$$z = \sin(x)\cos(y)$$

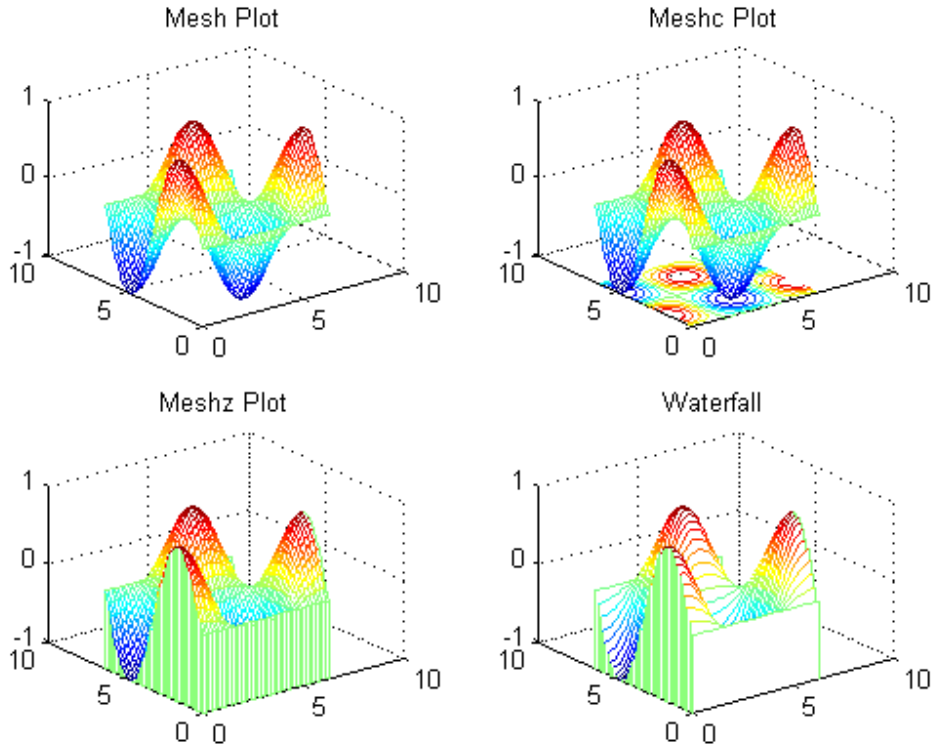
โดย x และ y มีค่าอยู่ระหว่าง 0 ถึง 2π

สำหรับ script file ต่อไปนี้จะเป็นการแสดงการเขียนพื้นผิวลายเส้นด้วยคำสั่งต่างๆ ดังนี้

```
x = linspace(0,2*pi,40);          % x เป็น vector ขนาด 1 x 100
y = x;                            % y เป็น vector ขนาด 1 x 100
z = sin(x)'*cos(y);       % สร้าง z ขนาด [100 x 1] x [1 x 100] = [100 x 1]

subplot(2, 2, 1);
mesh(x, y, z);
title('Mesh Plot')

subplot(2, 2, 2);
meshc(x, y, z) ;
title('Meshc Plot')
```

```
subplot(2, 2, 3)
meshz(x, y, z)
title('Meshz Plot')

subplot(2, 2, 4)
waterfall(x, y, z)
title('Waterfall')
```

ซึ่งจะได้ผลเป็นรูปต่อไปนี้



จะเห็นว่ารูปที่เขียนนั้น จะมีฉากหลัง เป็นระนาบ (x,y) , (y,z) และ (z, x) อยู่ 3 ระนาบ

ซึ่งหากเราต้องการฉากหลังแต่ไม่ต้องการเส้นประก็สามารถใช้คำสั่ง **`gird off`** ได้ หรือถ้าไม่ต้องการแทน x, y, z ก็อาจใช้คำสั่ง **`axis off`** ได้เหมือนในการ plot 2 มิติ

❹ การเขียนผิวทึบ หรือรูปทรง

การเขียนโครงข่าย หรือ mesh นั้นจะเป็นการลากเชื่อมจุดที่อยู่ติดกันด้วยเส้น แต่ถ้าหากเราต้องการเขียนรูปให้เป็นผิวทึบโดย MATLAB จะทำการเชื่อมต่อจุดที่ติดกันด้วยระนาบแทนที่จะเป็นเส้นเหมือนในคำสั่ง mesh ซึ่งสามารถทำได้โดยใช้คำสั่งต่อไปนี้

`surf(X,Y,Z,C)` กำหนด parameter X, Y, Z และ C เป็น matrix ขนาด m x n โดยจะเป็นการ plot จุด (x_{ij}, y_{ij}, z_{ij}) โดยใช้สี c_{ij} และจะมีการเชื่อมจุดที่อยู่ติดกันด้วยระนาบ หากไม่กำหนด C จะใช้ C = Z หากว่า x และ y เป็น vector จะเป็นการเขียนกราฟโดยใช้ plot (x_i , y_i , z_{ij})

surfc(x,y,z,c) จะเหมือน **surf(...)** เพียงแต่ MATLAB จะเขียน contour ใต้กราฟเพิ่มเติมเข้าไปด้วย

surfl(x,y,z,ls) จะเหมือน **surf(...)** เพียงแต่จะสามารถกำหนดทิศทางของแสงด้วย vector ls = [u h] เมื่อ u คือมุมบนระนาบ (x,y) ที่ทำกับแกน x วัดทวนเข็มนาฬิกา ส่วน h คือมุมที่ทำกับระนาบ (x,y) ค่าทั้งสองวัดเป็นมุมในหน่วยองศา

surfnorm(x,y,z) จะเหมือน **surf(...)** เพียงแต่จะเขียน normal vector ที่ผิวของกราฟด้วย ข้อควรระวังก็คือ x y และ z ต้องเป็น matrix ขนาดเท่ากัน เท่านั้นไม่สามารถใช้ vector x และ y ได้เหมือน **surf(...)**

[Nx,Ny,Nz] = surfnorm(x,y,z)
จะเหมือน surfnorm (x, y, z) ซึ่งไม่มีการเขียนกราฟ แต่จะให้ค่า normal vector ที่จุด (x_{ij}, y_{ij}, z_{ij}) ซึ่งมีค่า $(nx_{ij}, ny_{ij}, nz_{ij})$ แทน

สำหรับกราฟ 3 มิติที่ได้จากคำสั่ง surf นั้น จะยังคงมีลายเส้นของ mesh ปรากฏอยู่ ซึ่งเป็นการตั้งเบื้องต้นของ MATLAB ว่าถ้าเราไม่ต้องการกราฟในลักษณะนี้เราสามารถใช้คำสั่ง shading ได้ โดยมี 3 ลักษณะให้เลือกคือ

shading faceted เป็นค่าที่ตั้งเบื้องต้นของ MATLAB จะเป็นกำหนดให้ระนาบเล็กๆ แต่ละอันที่เชื่อมต่อจุดมีเพียงสีเดียว โดยจะเลือกค่าจากจุดที่มีค่าน้อยที่สุด และจะมีลายเส้นของ mesh แบ่งระนาบเล็ก ๆ เหล่านั้นด้วย

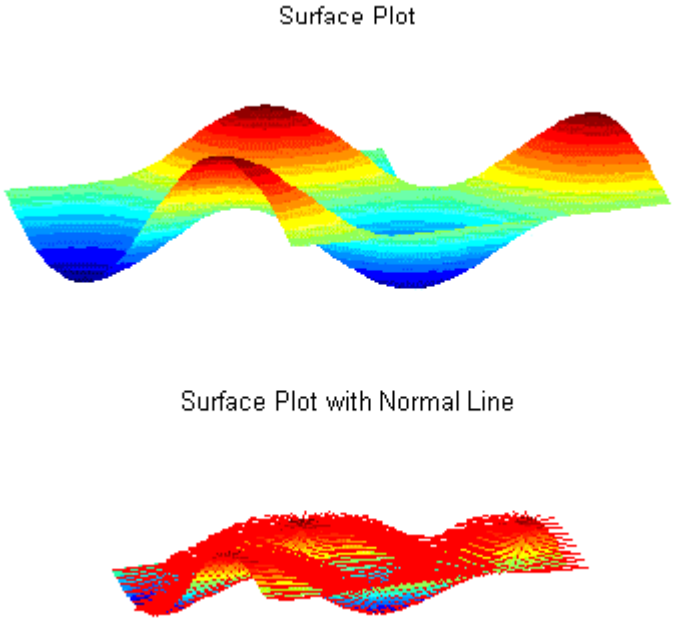
shading flat เหมือน faceted แต่จะไม่มีลายเส้น mesh

shading interp เป็นการ interpolated สีต่างๆ ให้กลมกลืนโดยจะไม่มีลายเส้นของ mesh ปรากฏอยู่และระนาบเล็กๆ แต่ละอันที่ประกอบเป็นผิวจะได้รับการ interpolate ให้มีสีที่ค่อยๆ เปลี่ยนไป

สำหรับตัวอย่างการเขียนกราฟด้วยคำสั่ง surf ต่างๆ เป็นดังนี้

```
x = linspace(0,2*pi,40); y = x; z = sin(x)'*cos(y);

subplot(2,1,1);
surf(x,y,z);shading interp
axis off
title('Surface Plot')

subplot(2,1,2)
[u v]=meshgrid(x,y);
surfnorm(u,v,z);shading flat
axis off
```

```
title('Surface Plot with Normal Line')
```

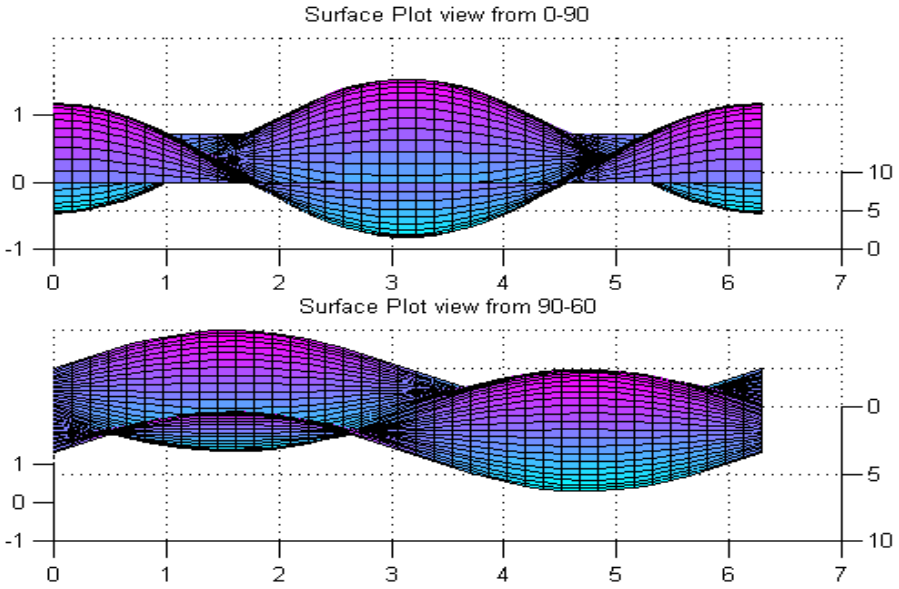


❺ การกำหนดมุมมอง

รูปทรง 3 มิตินั้นหากผู้มองมองจากมุมที่ต่างกันรูปที่ปรากฏก็อาจจะแตกต่างกันได้ การกำหนดตำแหน่งของมุมมอง สามารถกระทำได้ด้วยคำสั่ง view

view(az,el)	กำหนดมุมมองที่มุม azimuth, **az** และมุม elevation, **el** ทั้งคู่มีหน่วยเป็นองศา มุม azimuth คือมุมบนระนาบ (x,y) ที่กระทำกับแกน x วัดทวนเข็มนาฬิกา ส่วนมุม elevation คือมุมที่วัดขึ้นจากระนาบ (x,y) การกำหนดค่ามุม **az** หรือ **el** เป็นลบจะเป็นการวัดย้อนกลับจากทิศทางบวก
view(x,y,z)	กำหนดมุมมองเป็นมุมมองจากตำแหน่ง x, y, z ในระบบแกน Cartesian แต่ MATLAB จะไม่พิจารณาขนาดของ x, y, z กล่าวอีกอย่างหนึ่งก็คือเป็นการกำหนดมุมมองโดยอาศัยทิศทางของ vector นั่นเอง
view(2)	ตั้งค่ากลับไปที่มุมมองเบื้องต้นของ 2 มิติ (az = 0, el = 0)
view(3)	ตั้งค่ากลับไปที่มุมมองเบื้องต้นของ 3 มิติ (az = -37.5^0, el = 30)
[a,b]=view	จะได้ค่า a เป็นค่า az และ b เป็นค่า el ที่กำลังใช้อยู่ขณะนั้น

รูปต่อไปนี้แสดงกราฟซึ่งเขียนจากตัวอย่างที่ผ่านมาแต่ใช้มุมมองในการมองต่างกันออกไป ตามรูป



❻ การเขียน Contour Line

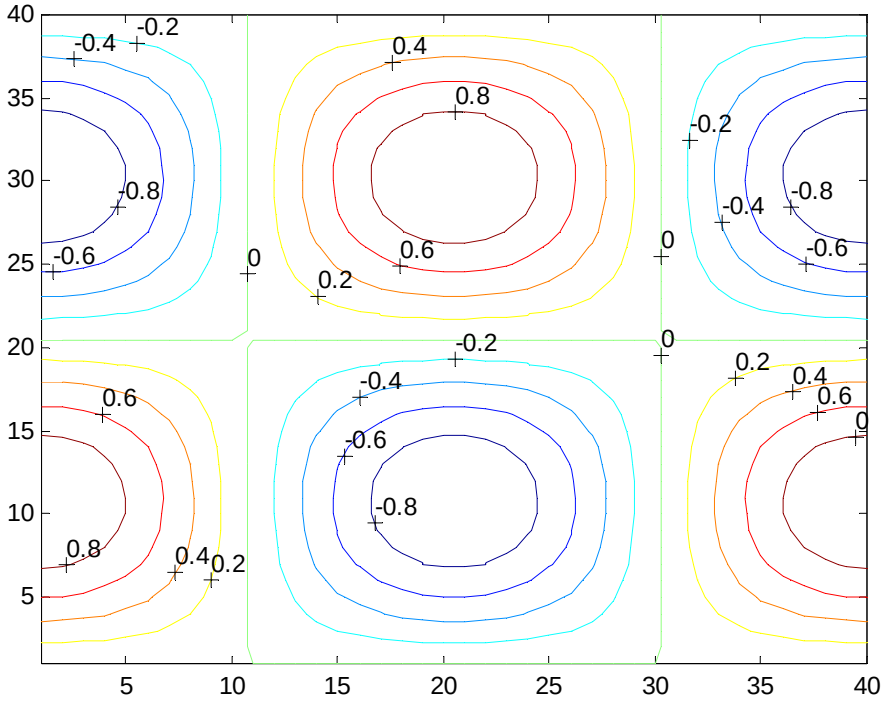
สำหรับ contour line จะเป็นการเขียนกราฟแสดงเส้นของเขตบอกความสูงเหมือนกับแผนที่บอกระดับที่มีเส้นบอกความสูง โดยการเขียน contour ก็เหมือนกับการนำระนาบมาตัดรูปกราฟ รอยตัดที่เกิดจากระนาบที่ระยะสูงต่างๆ กันเมื่อนำมาเขียนรวมบนระนาบเดียวก็จะได้ contour line คำสั่งที่ใช้มีดังนี้

contour(Z)	เขียนกราฟ contour ของ matrix Z โดยถ้า Z มีขนาด m x n, MATLAB จะแบ่ง scale ตามแกน x เป็น 1 ถึง m และ y เป็น 1 ถึง n
contour(Z,n)	เขียน contour line ของ matrix Z เป็นจำนวน n ระดับ ถ้าไม่กำหนดค่า n MATLAB จะใช้ค่าเบื้องต้นเป็น n = 10
contour(x,y,z)	เขียน contour line โดย x,y เป็นตัวกำหนดขนาด และ scale ของแกน x และแกน y ตามลำดับ ถ้า x,y เป็น rector จะเขียน (x_i,y_{ij},z_{ij})
contour(…,'str')	เขียน contour line โดยใช้ 'str' เป็น string กำหนดสีและเส้น
C = contour(…)	จะเป็นการคำนวณ matrix C เพื่อใช้ในการเขียน label ของเส้น contour
contourf(…)	จะเติมสีลงบน contour plot
clabel(C)	จะเขียน lable หรือค่าบอกขนาดของเส้นลงบนเส้น contour โดย **C** เป็นค่าที่คำนวณได้จากคำสั่ง **C = contour(…)** สำหรับตำแหน่ง MATLAB จะเลือกตำแหน่งที่เหมาะสมที่สุดเอง หากต้องการกำหนดเป็นอย่างอื่นสามารถอ่านข้อมูลเพิ่มเติมจาก **help clabel**
contour3(…)	เป็นการเขียน contour line ในระบบแกน 3 มิติ

ตัวอย่างของคำสั่ง contour และเป็นดังต่อไปนี้

```
x = linspace(0,2*pi,40); y = x; z = sin(x)'*cos(y);
C = contour(z);
clabel(C);
```

ซึ่งจะได้ผลเป็น



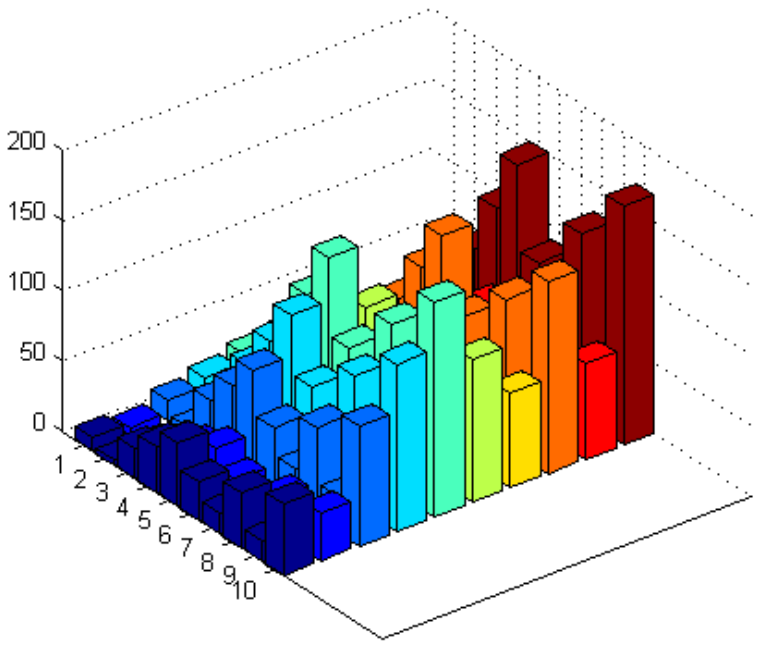
bar3(y,Z)	เขียน columns ของ m x n matrix Z เป็นแนวดิ่งใน bars กราฟ 3 มิติที่ตำแหน่งของ vector y ที่กำหนด โดย y จะต้องมีค่าของ element เพิ่มขึ้นหรือลดลงอย่างใดอย่างหนึ่งเท่านั้น
bar3(Z)	เหมือนคำสั่งด้านบนแต่จะกำหนดให้ y =1:m เมื่อ Z มีขนาด m x n
bar3(y,Z,w)	กำหนดความกว้างของแท่งกราฟ ซึ่งถ้า w > 1 จะทำให้รูปที่มีแท่งกราฟซ้อนกัน สำหรับค่าเบื้องต้นคือ w = 0.8
bar3(…,'detached')	จะเป็นกราฟตามค่าตั้งเบื้องต้น ซึ่งเป็นกราฟแท่งปกติ
bar3(…,'grouped')	จะเป็นกราฟแท่งที่เป็นกลุ่ม
bar3(…,'stacked')	จะเป็นกราฟแท่งที่ซ้อนกัน
bar3(…,linespec)	เป็นการใช้ string กำหนดลักษณะสีของแท่งกราฟ

และสำหรับกราฟ stem ใน 3 มิติจะมีลักษณะคำสั่งดังนี้

Stem3(z)	เขียนกราฟ discrete surface z เป็น stems จากระนาบ x-y และปลายจะจบด้วยวงกลมเหมือนใน 2 มิติ
Stem3(x,y,z)	เขียนกราฟ surface z ที่จุดที่กำหนดโดย x,y
Stem3(…,'filled')	เขียนกราฟ stem และระบายทึบวงกลมหรือเครื่องหมาย
Stem3(…,linespec)	ใช้กำหนดลักษณะของเส้นและเครื่องหมาย เหมือนคำสั่ง plot

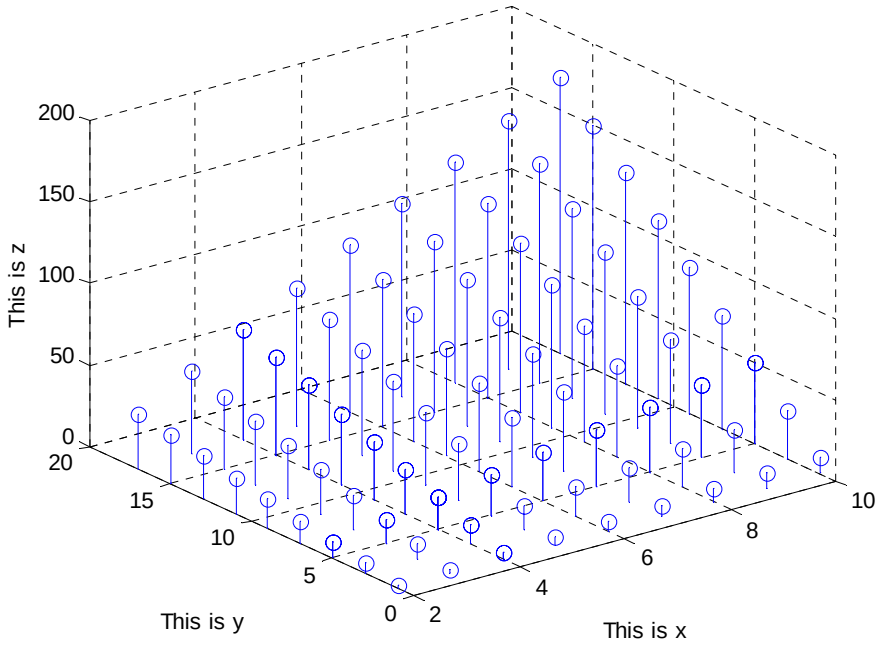
ตัวอย่างของกราฟแท่ง 3 มิติจะเป็นเช่น

```
x =3 2 5 7 9 6 4 8 4 10]';
y =2*x-3;
z =y*x';
bar3(z)
```



ตัวอย่างของ stem กราฟเช่น

```
x=[3 2 5 7 9 6 4 8 4 10]';
y=2*x-3;
z=y*x';
stem3(x,y,z)
xlabel('This is x')
ylabel('This is y')
zlabel('This is z')
```



9.3 การควบคุมสีของกราฟ

ในการใช้งานโดยปกติแล้ว MATLAB จะตั้งค่าต่างๆ มาให้ล่วงหน้าทำให้ผู้ใช้สามารถใช้ function ต่างๆ ในการเขียนกราฟได้ตามต้องการโดยไม่ต้องตั้งค่าใดๆ ก่อนก็ได้ อย่างไรก็ตามหากว่าผู้ใช้ต้องการที่จะสร้างกราฟฟิกส์ให้มีลักษณะและโทนสีที่แตกต่างออกไปก็สามารถทำได้โดยการกำหนดค่าตัวแปรที่เกี่ยวข้องกับการควบคุมสีให้เป็นไปตามความต้องการ

การควบคุมสีของ MATLAB ก็เหมือนการควบคุมสีของโทรทัศน์ซึ่งจะเป็นการผสมระหว่างแม่สี 3 สีคือ แดง-เขียว-น้ำเงิน หรือที่เรียกว่า Red-Green-Blue (RGB) Color โดยสีดำจะมี RGB color เป็น (0, 0, 0) ส่วนสีขาวจะมี RGB color เป็น (1 , 1, 1) ดังนั้นหากต้องการสร้างสีใหม่ก็เพียงแต่เลือกระดับสี RGB ให้มีสัดส่วนที่เหมาะสมตามความต้องการของเราเท่านั้น โดยค่าที่กำหนดของแต่ละสี RGB จะต้องอยู่ระหว่าง 0 ถึง 1

สำหรับการเขียนรูปพื้นผิว MATLAB จะใช้ **แผนที่สี** (color map) เป็นเครื่องควบคุมสีที่จะทำการ plot โดย color map เป็นจะ matrix ที่มีขนาด m x 3 โดยแต่ละ row ซึ่งมีตัวเลข 3 ตัวจะแทน RGB color แต่ละสี ดังนั้น matrix นี้จะกำหนดสีไว้จำนวน m สี เมื่อทราบแผนที่สีที่จะใช้อย่างแน่ชัดแล้วเมื่อมีการวาดพื้นผิวขึ้น MATLAB จะกำหนดสีของพื้นผิวด้วยดรรชนีของ color map ซึ่งโดยปกติเมื่อ MATLAB ได้รับคำสั่งให้เขียนผิว MATLAB จะนำค่าสูงสุดและต่ำสุดมาแบ่งออกเป็น m ช่วงตามจำนวนสีของ color map ที่กำหนดให้ จากนั้นจะกำหนดสีให้แต่ละค่าจากค่าสูงสุดถึงต่ำสุดของรูปพื้นผิวนั้น ส่วนค่าที่อยู่ในระหว่างนั้นจะใช้สีใดก็แล้วแต่การกำหนด shading เช่นการที่เราใช้คำสั่ง shading interp ก็เหมือนกับการ

interpolate color map matrix เพื่อกำหนดสีให้กับค่าระหว่างนั้นนั่นเอง คำสั่งที่เกี่ยวข้องกับการใช้ color map มีดังนี้

colormap(map)	ตั้งค่า color map ไปที่ค่า map โดย map นี้อาจจะใช้ที่ MATLAB กำหนดมาแล้วหรือผู้ใช้จะสร้างขึ้นเองก็ได้ map นี้เป็น M-file ที่ กำหนดค่า color map ที่จะใช้ในรูปภาพที่กำลังทำงานอยู่
colorbar	จะเขียนแถบสีในแนวดิ่งข้างรูปที่กำลัง plot อยู่
colorbar('horiz')	จะเขียนแถบสีในแนวนอนใต้รูปที่กำลัง plot อยู่

สำหรับ color map ที่ MATLAB กำหนดมาแล้วมีทั้งหมด 11 map ดังต่อไปนี้ (หากต้องการใช้ map ใดก็ใช้คำสั่ง) colormap(map) หรือ colormap map โดย map คือชื่อของแผนที่สีดังต่อไปนี้

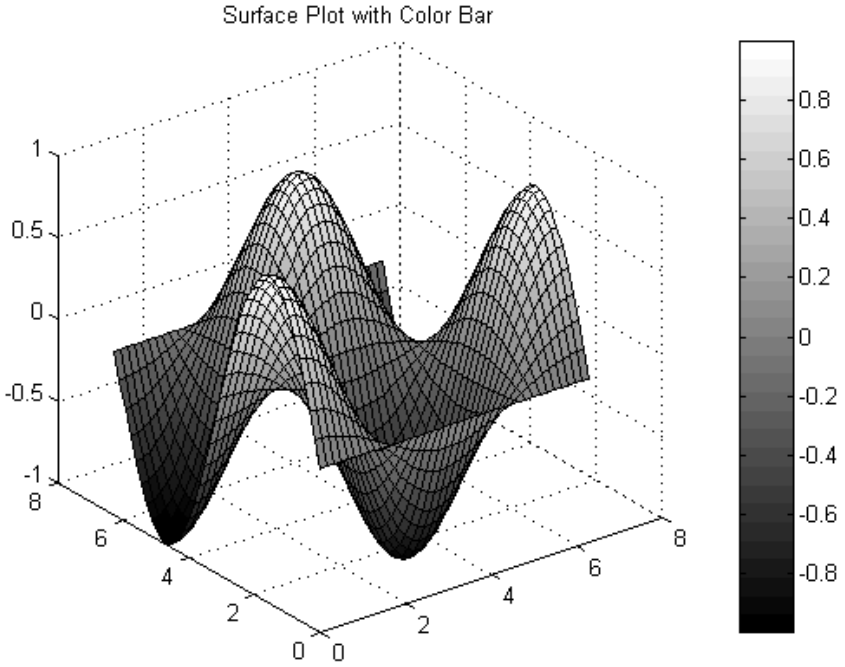
gray	สี gray linear scale จากดำไปขาว
hsv	เป็น Hsv-Saturation-Value color map เริ่มจากแดงไปน้ำเงินแล้วกลับมาแดงอีก
hot	เป็นสีโทนร้อนผสมระหว่างดำ-แดง-เหลือง-และขาว
cool	เป็นสีโทนเย็นออกทางสีฟ้า-ม่วง
bone	เป็นโทนสีฟ้าแก่ - สีเทา คล้ายกับการมองดูกระดูกบน X - ray
copper	เป็นสีในโทนสีของทองแดง
flag	เป็นสีแดง - ขาว - น้ำเงินแบบธงชาติสลับกันไป
pink	เป็นโทนสีทางสีชมพู
prism	เป็นสี แดง – ส้ม – เหลือง – เขียว – ฟ้า - ม่วง สลับกันไป
jet	เป็น hsv color map แต่ค่อย ๆ เปลี่ยนจากแดงเป็นน้ำเงิน
white	เป็นสีขาวทั้งหมด
lines	เป็น Color Map ที่ใช้ เขียนเส้นกราฟ
colorcube	เป็นสี color-cube
summer	โทนสีเขียวและเหลือง คล้ายฤดูร้อน
autumn	โทนสีแดงและเหลืองเหมือนฤดูใบไม้ร่วง
winter	โทนสีเขียวและน้ำเงิน จะดูทึมเหมือนฤดูหนาว
spring	โทนสีม่วงและเหลืองเหมือนฤดูใบไม้ผลิ

ซึ่งการจะดูรายละเอียดว่าสีใดมีลักษณะโทนสีเป็นอย่างไร ก็สามารถใช้คำสั่ง colormap (ชื่อของ map) ได้

การสั่ง map(m) โดย map เป็นชื่อของโทนสีข้างบนนี้จะเป็นแบ่ง map ของสีนั้นออกเป็น m ค่า หรือจะได้สีนั้นเป็น matrix ขนาด m x 3 เนื่องจากเอกสารที่ผลิตนี้พิมพ์ด้วยหมึกดำล้วนการยกตัวอย่างคงจะทำให้ท่านมองเห็นภาพได้ไม่ชัดเจนนัก แต่หากว่าท่านลองใช้คำสั่งต่อไปนี้กับเครื่องของท่านก็คงจะช่วยให้เข้าใจถึงการควบคุมสีต่าง ๆ ให้ดีขึ้น ตัวอย่างต่อไปนี้เป็นการเลือกใช้โทนสีเทา

```
x = linspace(0,2*pi,40);
y = x;
z = sin(x)'*cos(y);
surf(x,y,z);
colormap gray
colorbar
title('Surface Plot with Color Bar')
```

จะได้กราฟดังนี้



9.4 การสร้างภาพเคลื่อนไหว

MATLAB มี function ที่จะช่วยให้เราสามารถสร้างภาพเคลื่อนไหวได้ โดยหลักการแล้วภาพเคลื่อนไหวก็คือการเขียนภาพทีละภาพต่อๆ กันหลายๆ ภาพ หากว่าความเร็วในการเปลี่ยนภาพรวดเร็วพอเราก็จะเห็นว่าภาพนั้นเคลื่อนไหวได้เหมือนกับการเขียนภาพยนต์การ์ตูนนั่นเอง ดังนั้น MATLAB จึงใช้ชื่อ function เช่นเดียวกับภาพยนต์คือ แต่ละรูปจะเรียกว่า frame และเมื่อนำมาเขียนต่อเนื่องกันจะเรียก movie

ลำดับแรกในการสร้าง movie ก็จะต้องมีการสร้าง frame ขึ้นมาก่อน โดย frame ก็คือ figure window เฉพาะในส่วนที่กำลัง plot กราฟอยู่นั่นเอง เริ่มต้นก่อนจะทำการบันทึก frame เพื่อเป็นการกำหนดค่า

ความจำที่แน่นอนและช่วยประหยัดหน่วยความจำเราจะต้องกำหนดก่อนว่าภาพยนต์นี้จะมีกี่ frame และจะใช้ชื่อว่าอะไร สำหรับคำสั่งแรกของการสร้างภาพเคลื่อนไหวคือ

M = moviein(n) โดยภาพยนต์นี้จะชื่อ M ซึ่ง MATLAB จะบันทึกค่าของแต่ละ frame ไว้ในแต่ละ column ของ Matrix M และ n เป็นจำนวน frame ที่จะบันทึก

จากนั้นเมื่อต้องการบันทึก frame ใดก็จะใช้คำสั่ง

M(n,j) = getframe เป็นการบันทึก frame ที่ j ด้วยรูปภาพที่กำลังแสดงอยู่ใน figure window ขณะนั้นและเมื่อบันทึกครบทุก frame แล้ว ถ้าต้องการจะฉายภาพยนต์นั้นก็ใช้คำสั่ง

movie(M) จะเป็นการฉายภาพยนต์ ชื่อ M จำนวน 1 ครั้ง

movie(M,n) จะเป็นการฉายภาพยนต์ ชื่อ M จำนวน n ครั้ง

movie(M,n,FPS) จะเป็นการฉายภาพยนต์ ชื่อ M จำนวน n ครั้งด้วยความเร็ว FPS frame ต่อ วินาที ถ้าไม่กำหนดค่า FPS MATLAB จะตั้งค่าไว้ที่ 12 frame ต่อวินาที ข้อควรระวังก็คือการทำงานของ MATLAB จริงอาจไม่สามารถแสดงภาพตามค่าของ FPS ที่เรากำหนดได้ เนื่องจากมีตัวแปรหลายประการเช่นความเร็วของเครื่อง การทำงานของระบบแสดงภาพของเครื่อง เป็นต้น

เพื่อที่จะสามารถอธิบายรูปแบบทั่วไปในการสร้างภาพเคลื่อนไหวพิจารณาจากตัวอย่างต่อไปนี้

- สมมุติว่าเราต้องการสร้างภาพเคลื่อนไหวชื่อ ZORO ซึ่งจะมี frame ทั้งหมด 20 frame การเขียน M-file เพื่อจะสร้างภาพยนต์จะมีลำดับและรูปแบบเป็นดังนี้

```
n = 20;
ZORO  = moviein(n)
for j = 1:n
      ชุดคำสั่ง plot ภาพที่ต้องการบันทึก
      ZORO(:,j) = getframe
end
movie(ZORO)
```

อย่างไรก็ตามหลังจาก MATLAB version 5.3 จนถึง 6 เป็นต้นมา คำสั่งต่างๆ จะง่ายขึ้นเราไม่จำเป็นที่จะต้องใช้ฟังก์ชัน moviein(n) ก็ได้ เพียงแต่สั่ง getframe ลงในตัวแปรที่ต้องการเลย ทำให้เราสามารถสร้างภาพเคลื่อนไหวได้ง่ายขึ้น นั่นก็คือคำสั่งสำหรับ MATLAB version 6 จะมีลักษณะชุดคำสั่งเป็น

```
n = 20;
```

```
for j =1:n
      ชุดคำสั่ง plot ภาพที่ต้องการบันทึก
      ZORO(:,j)=getframe
end
movie(ZORO)
```

สำหรับ format นี้เป็น format มาตรฐานเบื้องต้นในการสร้างภาพเคลื่อนไหว สำหรับข้อควรปฏิบัติของการสร้างภาพเคลื่อนไหวมีดังนี้

- เนื่องจาก MATLAB มักจะปรับขนาดของแกนให้เหมาะสมกับการ plot แต่ละครั้งโดยอัตโนมัติ ทำให้ในการ plot หลายๆ ครั้งอาจมีการเปลี่ยนแปลงขนาดของแกน ดังนั้นเพื่อให้ภาพที่ได้ราบเรียบมากขึ้นควรจะมีการ fixed ค่าของแกนให้คงที่ไว้ โดยใช้คำสั่ง a = axis [x_{min},x_{max},...] หรือหลังจาก plot ครั้งแรกอาจจะใช้คำสั่ง axis(a), หรือ axis(axis) เพื่อให้ plot อยู่บน axis ขนาด a ขนาดเดียว
- การสร้างภาพเคลื่อนไหวอาจจะใช้หน่วยความจำมากดังนั้นระยะเคลื่อนไหวหรือการเปลี่ยนแปลงของแต่ละ frame ไม่ควรจะน้อยเกินไป อย่างไรก็ตามหากการเคลื่อนไหวของแต่ละ frame เปลี่ยนแปลงมากไป อาจทำให้มีความรู้สึกว่าภาพเคลื่อนไหวอย่างสดุดไปมาได้
- การทำงานของ Windows เป็นการทำงานในลักษณะ multi-tasking คือทำงานหลายๆ โปรแกรมในเวลาเดียวกัน ดังนั้นหากต้องการฉายภาพเคลื่อนไหวให้ได้ภาพที่สมบูรณ์อาจจะต้องปิดหรือเลิกการทำงานของโปรแกรมอื่นๆ ที่ทำงานอยู่ในขณะนั้น เพื่อที่ให้ cpu ใช้เวลาทำงานอยู่กับ MATLAB เพียงโปรแกรมเดียวอย่างเต็มที่
- การกำหนดค่า frame-per-second อาจช่วยให้ได้ภาพที่ดูเคลื่อนไหวสมจริงมากขึ้น

การแก้ปัญหาทางวิศวกรรม

❶ Flow Pass Rotating Cylinder

ในบทที่ 2 ได้มีการนำเสนอวิธีการเขียน stream line ของการไหล ซึ่งได้มาจากการเขียนกราฟของ stream function ซึ่งในทางปฏิบัติที่เหมาะสมแล้วควรจะเป็นการเขียน contour line มากกว่าเพราะว่าเส้น stream line ก็คือเส้นที่มีค่า stream function คงที่ ตัวอย่างต่อไปนี้เป็นการเขียนกราฟแสดงการไหลของของไหลอุดมคติที่ไม่อัดตัวหรือ potential flow ของการไหลแบบเอกรูป รวมกับ doublet ซึ่งทางกายภาพก็จะเหมือนกับการไหลของกระแสการไหลอิสระผ่านทรงกระบอก จากนั้นหากมีการรวม vortex เข้าไปอีกก็จะเป็นการไหลของกระแสอิสระผ่านทรงกระบอกที่หมุน ซึ่งในกรณีนี้จะทำให้เกิดแรงยกขึ้นและการศึกษาในเรื่องนี้จะเป็นพื้นฐานของการออกแบบแพนอากาศ สำหรับในเอกสารนี้ขอแสดงเฉพาะการเขียนกราฟของ stream function

$$\psi = U\left(r - \frac{a^2}{r}\right)\sin\theta + \frac{\Gamma}{2\pi}\ln(r)$$

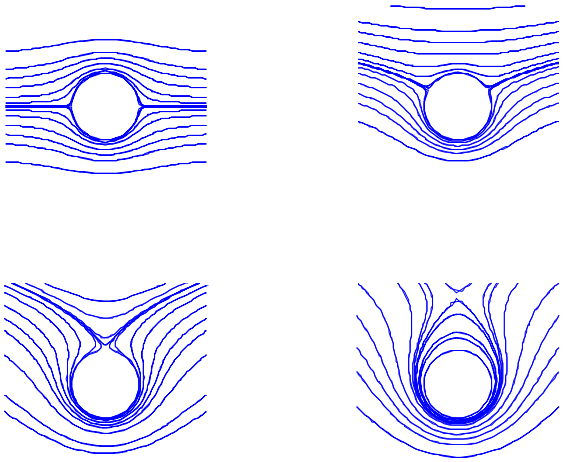
โดย a, K และ U เป็นค่าคงที่

สำหรับ M-file ต่อไปนี้เป็นการเขียนกราฟที่กำหนดให้ a =2 และมีค่า $\frac{\Gamma}{2\pi aU} = K$ ซึ่งหมายถึง circulation ต่างๆ กันเปลี่ยนไปเรื่อยๆ 4 ค่า โดยรูปแรกจะมี K= 0 ซึ่งก็หมายถึงการไหลที่ไหลผ่านทรงกระบอกที่หยุดนิ่ง จากนั้นรูปต่อๆ มาจะเป็นการเพิ่มค่า K ให้มากขึ้น ซึ่งก็เหมือนกับการเพิ่มความเร็วของการหมุนทรงกระบอกให้สูงขึ้นนั่นเอง M-file ที่ใช้เขียนรูปมีลักษณะดังนี้

```
%M-file for Potential Flow
% Flow Pass Rotating Cylindera=2;
r=linspace(a,20);
zeta=linspace(0,4*pi);
r=r';zeta=zeta';
[r zeta]=meshgrid(r,zeta);

K=0;
phi=sin(zeta)*(r./a - a./r)-K*log(r/a);
[x y]=pol2cart(zeta,r);
co= [-150 -100 -75 -50 -25 -10 -1 -0.1 0.1 1 10 25 50 75 100 150];
subplot(2,2,1)
contour(x,y,phi,co,'b');    % ในที่นี้จะเลือกสีน้ำเงินเท่านั้นเพราะจะแสดงภาพได้ชัดเจนเมื่อพิมพ์
axis equal
axis off
axis([-6 6 -6 6])

K=100;
phi=sin(zeta)*(r./a - a./r)-K*log(r/a);
[x y]=pol2cart(zeta,r);
co=[-150 -100 -75 -50 -25 -10 -1 -0.1 0.1 1 10 25 50 75 100 150];
subplot(2,2,2)
contour(x,y,phi,co,'b');
axis equal
axis off
axis([-6 6 -6 6])

K=200;
phi=sin(zeta)*(r./a - a./r)-K*log(r/a);
[x y]=pol2cart(zeta,r);
co= [-300 -250 -150 -100 -75 -50 -25 -10 -1 -0.1,...
      0.1 1 10 25 50 75 100 150];
subplot(2,2,3)
contour(x,y,phi,co,'b');
axis equal
axis off
axis([-6 6 -6 6])

K=300;
phi=sin(zeta)*(r./a - a./r)-K*log(r/a);
[x y]=pol2cart(zeta,r);
co= [-400 -300 -250 -200 -150 -100 -80 -75 -65 -60,...
      -40 -30 -1 5 10 25 35 50 60 75];
```

```
subplot(2,2,4)
contour(x,y,phi,co,'b');
axis equal
axis off
axis([-6 6 -6 6])
```

ซึ่งจะได้รูปกราฟออกมาดังนี้



❷ การ Plot Velocity Vector

สำหรับการไหลของ potential flow คงจะไม่สมบูรณ์หากไม่มีการเขียน velocity vector ในตัวอย่างนี้จะเป็นการพิจารณาการไหลผ่าน Rankine body ซึ่งมี stream function เป็น

$$\psi = Uy - \frac{q}{2\pi}\tan^{-1}\frac{2ay}{x^2 + y^2 - a^2}$$

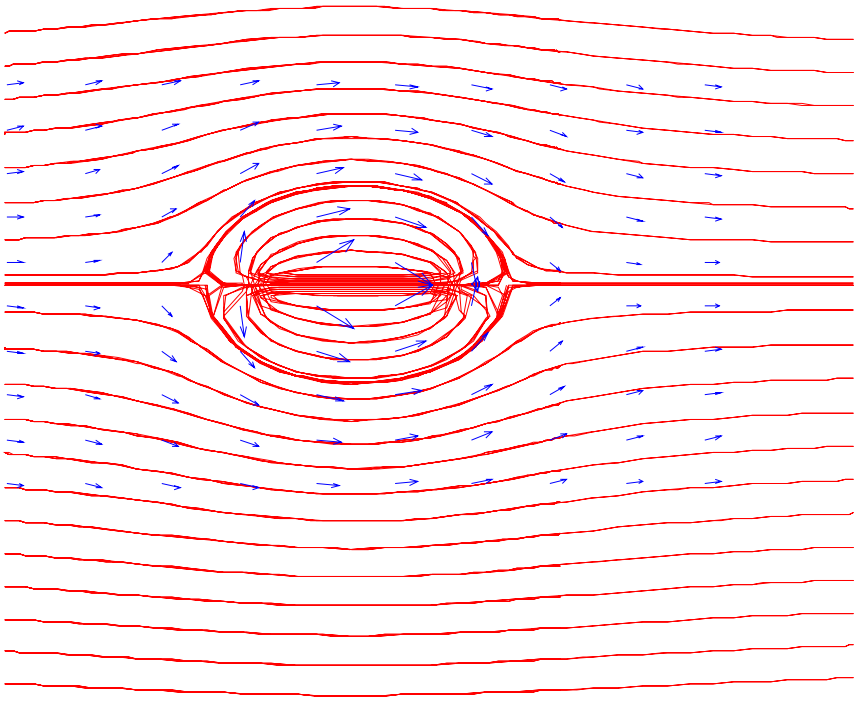
โดยความเร็วของการไหลนั้นจากทฤษฎีของ potential flow จะหาจาก $V_x = \frac{\partial \psi}{\partial y}$ และ $V_y = -\frac{\partial \psi}{\partial x}$ หรือการหาความเร็วก็จะสัมพันธ์กับการหา gradient ของ stream function นั่นเอง ในตัวอย่างต่อไปนี้จะเป็นการเขียนกราฟแสดง stream line พร้อมด้วย velocity vector ซึ่งมีขั้นตอนดังนี้

- ใช้ contour plot เขียน stream lines และหยุดกราฟไว้
- หาค่าความเร็วตามแนวแกน x และ y โดยใช้คำสั่ง **gradient**
- เขียน velocity vector โดยใช้คำสั่ง **quiver**

ซึ่งมีรายละเอียดของ M-file ดังต่อไปนี้

```
% Stream Line Plot With Velocity Vector
a=2;
```

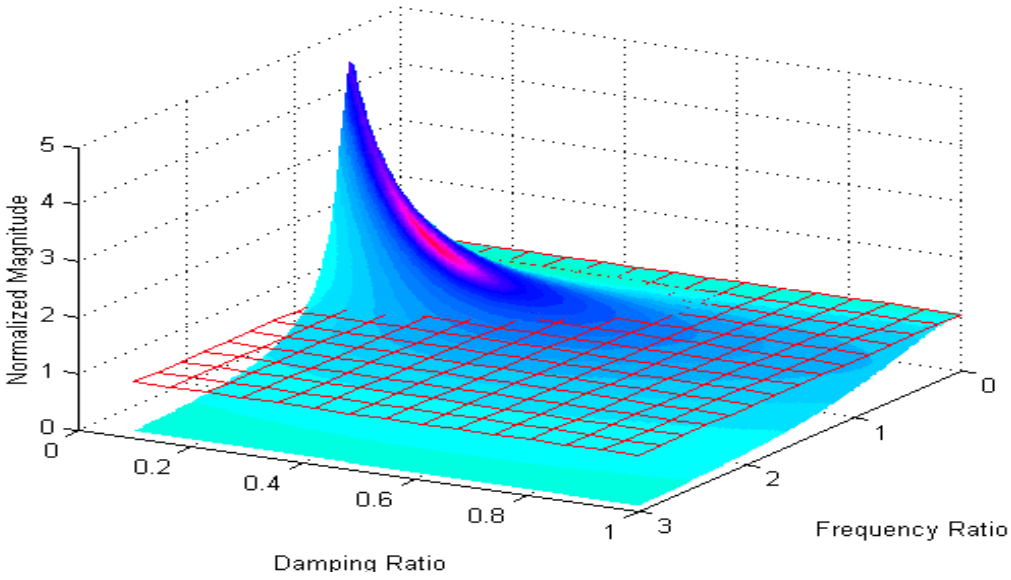
```
r=linspace(a^2,10);
q=linspace(0,2*pi);
r=r';q=q';
[x y]= pol2cart(q,r);
[x y]= meshgrid(x,y);
phi=0.8*y-(atan2(2*a*y,(x.^2+y.^2-a^2)));
contour(x,y,phi,20,'r');                %เขียนกราฟแสดง stream line
axis equal
hold on
grid off
phi0=[-0.01 0 0.01];
contour(x,y,phi,phi0,'r')               %เขียนกราฟแสดง stream line ของ body
x=linspace(-7,7,10);
y=linspace(-4,4,10);
x=x';y=y';
[x y]=meshgrid(x,y);
a=2;
phi=0.8*y-(atan2(2*a*y,(x.^2+y.^2-a^2)));
[v u]=gradient(phi,0.2,0.2);            %หาค่า velocity
v=-v;
u=abs(u);
quiver(x,y,u,v,0.6)                     %เขียนกราฟแสดง velocity vector
hold off
axis off
```



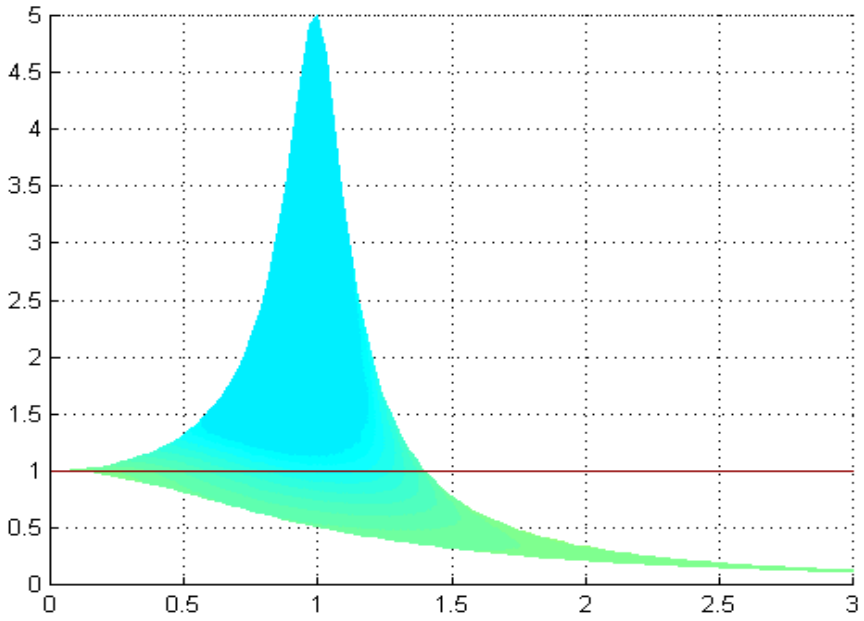
➢ Response Surface

สำหรับการสั่นทางวิศวกรรม ของ single DOF damped forced vibration ตามที่กล่าวมาในตัวอย่างของบทที่ 2 จะเป็น function ของ frequency ratio และ damping ratio ซึ่งสามารถนำมาเขียนเป็นรูปในสาม

มิติได้ ซึ่งรูปต่อไปนี้เป็นการเขียนผิวของการสั่นของระบบ ส่วนในรูปจะเห็นว่ามีระนาบตัดผ่านผิวอยู่ โดยระนาบมีตำแหน่งคงที่อยู่ที่ Normalize Magnitude เท่ากับ 1 ซึ่งระนาบนี้ทำให้เราทราบว่าค่าในช่วงใดที่มี normalized magnitude น้อยกว่า 1 หมายความว่าระบบจะเกิดการเคลื่อนไหวน้อยกว่ากรณีที่ระบบได้รับแรงคงที่ที่มีขนาดเท่ากัน



ส่วนรูปต่อไปนี้เป็นรูปที่ได้จากการเขียนกราฟเดียวกัน แต่ได้เปลี่ยนมุมมองโดยมองเข้ามาทางแกนของ damping ratio



บทที่ 10 SYMBOLIC MATHEMATICS

แม้ว่า MATLAB จะเป็นโปรแกรมที่มีวิธีการวิเคราะห์ค่าต่างๆ เป็นวิธีเชิงตัวเลขเกือบทั้งหมด แต่ MATLAB ก็มี toolbox ที่ใช้ในการคำนวณทางคณิตศาสตร์เชิงสัญลักษณ์หรือ Symbolic Mathematics ซึ่งมีความจำเป็นต้องใช้ในการคำนวณด้านวิศวกรรมในหลายๆ กรณี สำหรับ Symbolic Math Toolbox ของ MATLAB จะเป็น Toolbox มาตรฐาน ที่มากับ Student Edition ของ MATLAB ส่วนใน MATLAB 5.x ได้มีการยกเลิกคำสั่งที่เคยใช้ได้ใน version ที่ต่ำกว่าหลายๆ คำสั่งไป อย่างไรก็ตามในการยกตัวอย่างผู้เรียบเรียงจะยกตัวอย่างของ MATLAB 5.x ทั้งหมด และในตอนท้ายของบทนี้จะมีคำสั่งที่เกี่ยวข้องกับ Symbolic Toolbox ของ MATLAB 5.x ไว้เพิ่มเติมให้ด้วย

Symbolic Mathematics ก็คือคณิตศาสตร์ที่เราคุ้นเคยในการใช้งานมาตั้งแต่ชั้นมัธยมคือการใช้สัญลักษณ์เพื่อการคำนวณต่างๆ เช่นการกำหนดค่า sin(x) ที่ผ่านมาเราต้องกำหนดค่าตัวแปร x เป็นตัวเลขก่อน จึงจะหาค่า sin(x) ได้ แต่ในกรณีของ symbolic mathematic เราไม่จำเป็นต้องกำหนดค่า x เป็นตัวเลขแต่สามารถนำ function sin(x) นี้ไปคำนวณได้เลย เช่นการหาอนุพันธ์ของ sin(x) จะได้ $\frac{d\sin(x)}{dx} = \cos(x)$

สำหรับ Symbolic Math Toolbox ที่ใช้เป็น symbolic function ของ MATLAB นี้ มีพื้นฐานมาจากโปรแกรม Maple V. Software Package ซึ่งเป็น software ทางด้าน symbolic mathematic ที่มีชื่อเสียงและมีผู้นิยมใช้มากที่สุดโปรแกรมหนึ่งของโลก ซึ่งขั้นแรกได้มีการเริ่มทำโครงการวิจัยโปรแกรมนี้ขึ้นที่ University of Waterloo และในปัจจุบันได้มีการทำเป็นการค้าที่ผลิตแผยแพร่โดยบริษัท Watterloo Maple Software, Inc. ประเทศ Canada

10.1 Symbolic Algebra

❶ การกำหนด Symbolic Expression

สำหรับ Symbolic expression จะสามารถกำหนดได้โดยจะมี quote make ‘ ’ ดังเช่น

```
t = 'x^2+5*x-4'
```

จะเป็นการกำหนดค่าของ expression t(x) เป็นต้น โดยที่ expression เหล่านี้อาจบรรจุตัวแปรมากกว่า 1 ตัวแปร เช่น `'sin(y+x)'` จะมีทั้งตัวแปร x และ y ในการแก้สมการหรือกระทำกรรมวิธีใดๆ ทางคณิตศาสตร์ ซึ่งเราจำเป็นต้องทราบว่าตัวแปรตัวใดเป็น independent variable แต่หากไม่มีการกำหนด

independant variable ขึ้น MATLAB จะเลือกจาก expression นั้นมา 1 ตัว โดยจะเลือกจากตัวอักษรตัวเล็กเดี่ยว (ยกเว้น i และ j) ที่อยู่ใกล้กับ x มากที่สุด แต่ถ้าตัวอักษรยังห่างจาก x เท่ากัน MATLAB จะพิจารณาตัวที่ 2 ต่อไป (สำหรับกรณีที่ไม่เป็นอักษรเดี่ยว) ถ้าไม่มีตัวอักษรที่เข้าข่ายอยู่เลย MATLAB จะเลือก x (แม้ว่าไม่มีอักษร x อยู่ก็ตาม)เป็น independent variable ถ้าต้องการทราบว่า MATLAB จะเลือกตัวอักษรใดใน expression เป็น independent variable เราสามารถใช้ function symvar

`symvar('exp(x)')` MATLAB จะให้ independent variable สำหรับ symbolic expression **exp**

นอกจากนี้ เราจะสามารถกำหนด variable ได้มากกว่าหนึ่งตัว โดยเริ่มจากการกำหนดในตอนเริ่มเลยว่าตัวแปรตัวใดบ้างจะเป็น symbolic variable และ/หรือ symbolic expression แต่โดยทั่วไป MATLAB จะนิยมกำหนดตัวแปรให้เป็น symbolic variable มากกว่า สำหรับการกำหนดชนิดตัวแปรจะใช้คำสั่ง sym โดยมีรูปแบบคำสั่งต่อไปนี้

`S=sym('expression')` สร้าง symbolic variable S จาก expression โดยที่ expression อาจเป็นตัวแปรตัวเดียวหรือชุดสมการก็ได้ การกำหนดตัวแปรตัวใดตัวหนึ่งให้เป็นsymbolic variable จะช่วยลดภาระการกำหนด expression อื่นๆ อีกหลายกรณี

ตัวอย่างเช่น

```
x=sym('x')
```

เป็นการนิยามว่า x ต่อไปนี้จะเป็น symbolic variable และมีค่าเป็น x จากนั้นตัวแปรใดก็ตามที่นิยามโดยใช้ x เข้าไปเกี่ยวข้อง จะเป็น symbolic expression ทั้งนั้นเช่น

```
f=x^2+3*x-5;
g=2*x^2-5*x+2;
```

จะได้ว่าทั้ง f และ g เป็น symbolic expression

สำหรับอีกคำสั่งหนึ่งคือ คำสั่ง syms ซึ่งเป็นการนิยาม symbolic variable พร้อมกันหลายๆตัว

`syms 'ex1' 'ex2'…` สร้าง symbolic variable จาก expressions โดยที่ expression แต่ละค่าอาจเป็นตัวแปรตัวเดียวหรือชุดสมการก็ได้

ตัวอย่างเช่นถ้าเราต้องการให้ x, y และ z เป็น symbolic variable เราจะใช้

```
syms 'x' 'y' 'z'
```

จะมีผลเท่ากับการใช้

```
sym('x'); sym('y'); sym('z');
```

ส่วนการพิจารณาว่าค่าใดเป็นตัวแปรบ้างจะใช้คำสั่ง **findsym**

findsym(S) หาค่า symbolic variable ของ S โดย S ได้จากคำสั่ง sym

ดังตัวอย่างเช่น

```
s=sym('exp(bc)*sin(z)');
findsym(s)
ans =
bc, z
```

นั่นคือ bc และ z จะเป็น symbolic variables ของ expression นี้ หรือพิจารณาตัวอย่างต่อไปนี้

```
x=sym('x');
f=5*x^2-3*x-1;
findsym(f)
ans =
x
```

จะเห็นว่า f เป็น symbolic expression ที่มี symbolic variable เป็น x ส่วนการพิจารณาว่า expression ใดเป็น symbolic expression หรือไม่สามารถใช้คำสั่ง class ได้ โดยมีรูปแบบเป็น

class(OBJ) จะให้ชนิดของตัวทดสอบ OBJ ว่าเป็นตัวแปรชนิดใด ซึ่งคำตอบจะบอกชนิดของตัวแปรจะมีดังนี้

- **double** ตัวแปรเป็นตัวเลข double precision floating point number ซึ่งเป็นตัวเลขปกติที่ MATLAB ใช้
- **sparse** เป็น 2-D real (or complex) sparse matrix
- **struct** เป็น Structure array ดูรายละเอียดในบทที่ 12
- **cell** เป็น cell array ดูรายละเอียดในบทที่ 12
- **char** เป็น Character String
- **sym** เป็น symbolic variable

ตัวอย่างเช่น

```
x=sym('x');
class(x)
ans =
sym

f=5*x^2-3*x-1;
class(f)
ans =
sym
```

❷ การ plot ด้วย symbolic expression

จะใช้คำสั่ง **ezplot** ซึ่งเป็นการ plot อย่างง่าย ๆ โดย

1. plot 2 มิติ

ezplot(s) สร้าง plot ของ symbolic expression s โดยสมมุติว่าเป็น function ของตัวแปรเดี่ยวโดยจะ plot ในช่วง -2π ถึง 2π **ข้อควรระวัง การใช้ ezplot นี้จะต้องมี symbolic variable เพียงตัวเดียว**

ezplot(s,[min max]) สร้าง plot โดยใช้ช่วง independent variable ของ S ในช่วง min ถึง max

❸ การจัดรูป Expression ทางคณิตศาสตร์

การจัดรูป expression ทางคณิตศาสตร์ให้อยู่ในรูปอย่างง่ายมี function ที่ช่วยอยู่หลาย function ต่อไปนี้

collect(s) รวมสัมประสิทธิ์ของตัวแปรที่มีกำลังเท่ากันของ expression S เข้าด้วยกัน

collect(s,v) เหมือน **collect(s)** แต่กำหนดให้ v เป็น independer variable

expand(s) กระจาย expression **s**

factor(s) แยกตัวประกอบของ expression **s**

simple(s) จัดรูปให้ **s** อยู่ในรูปที่สั้นลงถ้าทำได้

simplify(s) จัดรูปให้ **s** อยู่ในรูปที่ง่ายขึ้นโดยใช้กฎของ Maple

ตัวอย่างการทำงานของ function ที่กล่าวมาโดยกำหนด expression ต่อไปนี้

```
x=sym('x')
s1= x^2-4*x+4;
s2=(x*cos(x))^2-x^2;
s3=(x-3)*(x-6)*(x+2)/(x^2-4*x-6);
```

จะได้

```
factor(s1)
ans =
(x-2)^2

expand(s2)
ans =
x^2*cos(x)^2-x^2

factor(s2)
ans =
x^2*(cos(x)-1)*(cos(x)+1)

collect(s3)
ans =
(x-3)*(x-6)*(x+2)/(x^2-4*x-6)

expand(s3)
ans =
1/(x^2-4*x-6)*x^3-7/(x^2-4*x-6)*x^2+36/(x^2-4*x-6)
```

```
simple(s3)
ans =
(x^3-7*x^2+36)/(x^2-4*x-6)
```

ในหลายๆ กรณีหากว่า symbolic expression ของเรามีความยุ่งยากมาก MATLAB ก็จะคืนค่าเดิมมาให้ เพราะไม่สามารถจะจัดให้อยู่ในรูปอย่างง่ายหรือรูปที่เราต้องการได้

❹ Operations on Symbolic expression

operation ปกติของคณิตศาสตร์สามารถใช้กับ symbolic expression ได้โดยใช้ symbolic function นอกจากนั้น symbolic function สามารถใช้เปลี่ยน symbolic expression จากรูปแบบหนึ่งไปอีกรูปแบบหนึ่งได้ด้วย function ต่าง ๆ มีดังนี้

numden(s)	ให้ expression 2 symbolic expression โดยเป็นเศษและส่วนของ s
numeric(s)	เปลี่ยน s เป็นรูปแบบเชิงตัวเลข เหมือนกับการใช้ **double(sym(s))**
poly2sym(c)	เปลี่ยน polynomial coefficient vector **c** เป็น symbolic expression
pretty(s)	พิมพ์ s ออกเป็น output
sym2poly(s)	เปลี่ยน s เป็น polynomial coefficient vector
symadd(s1,s2)	บวก symbolic expression, s1 + s2 หรือจะใช้ **sym(s1)+sym(s2)** หรือ **s1+s2** แทนก็ได้
symmul(s1,s2)	คูณ symbolic expression, s1 x s2 หรือจะใช้ **sym(s1)*sym(s2)** หรือ **s1*s2** แทนก็ได้
sympow(s1,s2)	ยกกำลัง symbolic expression, s1^s2 หรือจะใช้ **sym(s1)^sym(s2)** หรือ **s1^s2** แทนก็ได้
symsub(s1,s2)	ลบ symbolic expression, s1 - s2 หรือจะใช้ **sym(s1)-sym(s2)** หรือ **s1-s2** แทนก็ได้

เพื่อให้เข้าใจการทำงานของ function ที่กล่าวมา สมมุติให้ s1,s2 และ s3 เป็น symbolic expression ที่ยกตัวอย่างในตัวอย่างที่ผ่านมาคือ

```
x = sym('x')
s1= x^2-4*x+4');
s2= (x*cos(x))^2-x^2
s3= (x-3)*(x-6)*(x+2)/(x^2-4*x-6)
[s4 s5]=numden(s3)
s4 =
(x-3)*(x-6)*(x+2)
s5 =
x^2-4*x-6
sym2poly(s4)
ans =
     1     0     0    36

s4 + s5
```

```
ans =
(x-3)*(x-6)*(x+2)+x^2-4*x-6

factor(ans)
ans =
x^3-6*x^2+30-4*x
```

10.2 Equation Solving

❶ การแก้สมการระบบสมการเชิงเส้น

สำหรับระบบสมการเชิงเส้น MATLAB สามรถใช้ symbolic expression แก้สมการได้เช่นกันโดยมีรูปแบบดังนี้

solve(s) แก้สมการหารากของ expression s โดยการหารากของ variable ของ s ถ้า s เป็น symbolic equation จะเป็นการแก้สมการของ s

solve(s1,s2,…,sn) แก้สมการ n สมการ s1,s2 ถึง sn เพื่อหา independent variable ของแต่ละ symbolic equation s1, s2,...sn

ตัวอย่างการใช้งานคำสั่ง solve(s) เช่นกำหนดให้

```
syms 'x1' 'y1' 'z1'
s1= x1+y1+z1-5;
s2= 2*x1-y1+2*z1-4;
s3= x1-4*y1-z1+2;
a=solve(s1)
a =
-y1-z1+5
```

จะเห็นว่าในกรณีที่มี symbolic expression หลายตัว MATLAB จะเลือกตัวแปรที่มีลำดับอยู่ใกล้ตัวอักษร x มากที่สุด เป็น independent variable สำหรับตัวอย่างของการแก้ระบบสมการ จากค่า **s1, s2, s3** ที่กำหนดมาแล้วจะได้

```
[x y z]=solve(s1,s2,s3)
x =
9/2
y =
2
z =
-3/2
```

จากคำตอบที่ได้ จะพบว่า x = 9/2, y = 2 และ z = -3/2 ซึ่งมองเผินๆ จะเหมือนว่า x,y และ z ควรจะเป็นตัวเลข แต่ในความเป็นจริงแล้วเมื่อลองพิจารณา

```
class(x)
ans =
sym
```

จะเห็นว่า x นั้นเป็น symbolic variable ซึ่งไม่มีค่าทางคณิตศาสตร์ อย่างไรก็ตามเราสามารถเปลี่ยนค่านี้ไปเป็น ตัวเลขได้ โดยใช้คำสั่ง **double**

double(x) เปลี่ยน x ใน expression ใดๆ ให้มีค่าเป็นตัวเลขและมีลักษณะเป็น double precession number

เช่นจากตัวอย่างที่ผ่านมาหากเราใช้คำสั่งนี้กับค่า x จะได้

```
double(x)
ans =
  4.5000
```

ซึ่งค่าที่ได้นี้จะมีค่าเป็นค่าทางคณิตศาสตร์

สำหรับการแก้สมการที่ยุ่งยากมากๆ อาจไม่สามารถแก้ด้วยวิธี symbolic ได้ ในกรณีที่ MATLAB ไม่สามารถที่จะหา analytical solution ให้ได้ MATLAB จะหา numerical solution ให้แทน เช่น

```
S1=sym('x^3*exp(x)*sin(x))=2');
[b]=solve(S1)
b =
.25394136454664449139483333 -1.0402767432558262675794l0*i
```

❷ Solution to ODE

- การแก้ first-order-ODE ซึ่งมี สมการอยู่ในรูป

$$y' = \frac{dy}{dx} = g(x, y)$$

ซึ่งในบทที่ผ่านมาเราได้แก้สมการ ODE ด้วยวิธีเชิงตัวเลขมาแล้วสำหรับ symbolic toolbox ก็สามารถจะแก้ first-order ODE ซึ่งมีคำสั่งดังนี้

dsolve('equation','condition')

เป็นการแก้สมการ ODE ที่บรรจุอยู่ในสมการ symbolic **'equation'**และมี option เป็น**'condition'**ที่กำหนด initial condition นอกจากนี้สำหรับ independent variable ที่ MATLAB ใช้จะเป็น symbolic **'t'** ส่วน $\frac{dy}{dt}$ ใน MATLAB จะแทนด้วย Dy นั่นคือใช้ D แทนเครื่องหมาย differential ส่วนอักษรที่ตามมาก็จะเป็น dependent variable ตัวอย่างเช่น $\frac{dz}{dt} = \sin x$ หากเขียนใน MATLAB จะเขียนเป็น **'Dz=sin(x)'** ส่วน second derivative จะใช้ D2y และ third derivative ก็

จะเป็น D3y ต่อไปเรื่อยๆ สำหรับ condition หากไม่มีการกำหนดให้ MATLAB จะให้ค่า C1, C2,... ซึ่งถือว่าเป็นค่าคงที่ที่ได้จากการ integrate หรือแก้สมการคืนมาให้กับคำตอบ

- ตัวอย่างสำหรับการแก้สมการ first order 1 สมการ เช่น

```
dsolve('Dx = -a*x^2')
ans =
1/(a*t+C1)

x = dsolve('Dx = -a*x^2','x(0) = 1','s')
x =
1/(a*s+1)

x = dsolve('Dx = -a*x^2','x(0) = 1')
x =
1/(a*t+1)
```

- สำหรับการแก้ระบบสมการ first order จะใช้ลักษณะคำสั่งเป็น

```
dsolve('equation1', 'equation2',...,'condition')
```

หรือ

```
dsolve('equation1, equation2,...','condition')
```

ลองพิจารณาตัวอย่าง

```
S = dsolve('Df = f + g, Dg = -f + g','f(0) = 1','g(0) = 2')
S =
    f: [1x1 sym]
    g: [1x1 sym]
```

จะเห็นว่าเนื่องจากเรามีตัวแปร 2 ตัวที่ต้องการหาค่า แต่เรากำหนด output S เพียงค่าเดียว MATLAB จึงได้เก็บตัวแปรนี้ไว้ใน field S.f และ S.g ตามลำดับ (โครงสร้างตัวแปรแบบนี้เป็นโครงสร้างใหม่ที่ใช้ใน MATLAB 5 เท่านั้นดูรายละเอียดโครงสร้างตัวแปรแบบใหม่นี้ได้ในบทที่ 12) หากต้องการทราบค่าสามารถพิมพ์ชื่อ field ลงไปเช่น

```
S.f
ans =
exp(t)*cos(t)+2*exp(t)*sin(t)

S.g
ans =
-exp(t)*sin(t)+2*exp(t)*cos(t)
```

หรือลองพิจารณาสมการ first order 3 สมการดังนี้

```
[u1 v1 w1] = dsolve('Du=v, Dv=w, Dw=-u','u(0)=0, v(0)=0, w(0)=1')
u1 =
1/3*3^(1/2)*exp(1/2*t)*sin(1/2*t*3^(1/2))-
1/3*exp(1/2*t)*cos(1/2*t*3^(1/2))+1/3*exp(-t)
```

```
v1 =
1/3*3^(1/2)*exp(1/2*t)*sin(1/2*t*3^(1/2))+
1/3*exp(1/2*t)*cos(1/2*t*3^(1/2))-1/3*exp(-t)
w1 =
1/3*exp(-t)+2/3*exp(1/2*t)*cos(1/2*t*3^(1/2))
```

สำหรับสมการ order สูงๆขึ้นก็มีรูปแบบในทำนองเดียวกันเช่น

ตัวอย่างเช่นสมการ nonlinear second order ODE

$$\frac{d^2y}{dx^2} = y - \frac{dy}{dx} - 5$$

ถ้ากำหนดให้ initial condition เป็น $y'(0) = 0$

$$y(0) = 0.25$$

จะใช้คำสั่งเป็น

```
dsolve('D2y=y-Dy-5','Dy(0)=0,y(0)=0.25')
ans =
5+(-19/8-19/40*5^(1/2))*exp(1/2*(5^(1/2)-1)*t)-19/40*(5^(1/2)-
1)*5^(1/2)*exp(-1/2*(5^(1/2)+1)*t)
```

สำหรับในกรณีที่ไม่สามารถที่จะหา analytical solution ที่ใกล้เคียงได้ MATLAB จะให้คำเตือนออกมา

10.3 Differentiation และ Integration

สำหรับ differentiation และ Integration ของ symbolic math จะมี function ดังนี้

diff(s)	differentiation expression **s** เทียบต่อ independent variable ของมันเอง
diff(s,'t')	differentiation expression **s** เทียบต่อ independent variable **'t'**
diff(s,n)	differentiation expression **s** เทียบต่อ independent variable ของมันเอง เป็นจำนวน n ครั้ง
diff(s,'t',n)	differentiation expression **s** เทียบต่อ **t** เป็นจำนวน **n** ครั้ง

จะเห็นว่า symbolic toolbox นี้จะใช้ function diff เดียวกันนี้จะใช้ใน numerical diff function เช่นกัน ทั้งนี้ MATLAB จะคำนวณด้วยวิธีใดก็ขึ้นกับชนิดของตัวแปรว่าเป็น symbolic หรือ numeric variable

ส่วน Numeric Integration จะใช้ function ต่อไปนี้

int(s)	integrate symbolic expression **s** เทียบต่อ independent variable ของมันเอง
int(s)	integrate symbolic expression **s** เทียบต่อ independent variable '**t**'
int(s,a,b)	integrate **s** เทียบต่อ independent variable ของมันแล้วใช้ limit จาก **a** ถึง **b**

int(s,'t',a,b) integrate expressions เทียบต่อ **t** จาก **a** ถึง **b**

int(s,'m','n') integrate expression **s** เทียบต่อ independent variable ของมันเองโดยใช้ limit จาก '**m**' ถึง '**n**' โดยทั้งสองเป็น symbolic variable

ในหลายกรณี function ที่เราต้องการจะ integrate อาจจะมีความยุ่งยากซับซ้อนเกินกว่าที่จะสามารถ integrate ด้วยวิธี symbolic ได้ หากเป็นเช่นนั้น MATLAB จะตอบกลับว่าไม่สามารถทำได้ เราอาจต้องแก้ไขโดยการใช้ numerical integration หาคำตอบแทน

สำหรับตัวอย่างของ Symbolic differentiation และ integration สมมุติให้

```
x=sym('x');
s1=x^3*exp(x)*sin(x)-2;
s2=x^3+3*x^2+5*x-2;
s3=x*sin(x);
```

ซึ่งตัวอย่างการใช้คำสั่งมีดังต่อไปนี้

```
diff(s1)
ans =
3*x^2*exp(x)*sin(x)+x^3*exp(x)*sin(x)+x^3*exp(x)*cos(x)

diff(s1,'x')
ans =
3*x^2*exp(x)*sin(x)+x^3*exp(x)*sin(x)+x^3*exp(x)*cos(x)

diff(s1)
ans =
3*x^2*exp(x)*sin(x)+x^3*exp(x)*sin(x)+x^3*exp(x)*cos(x)

diff(s1,'t')
ans =
      0

diff(s2)
ans =
      3*x^2+6*x+5

diff(s3,3)
ans =
      -3*sin(x)-x*cos(x)
```

สำหรับการ integrate มีตัวอย่างดังต่อไปนี้

```
int('x^4')
ans =
1/5*x^5

int('x*cos(x)')
ans =
cos(x)+x*sin(x)
```

```
int('x^2*exp(x)')
ans =
x^2*exp(x)-2*x*exp(x)+2*exp(x)

int('x*sin(x)',0,pi)
ans =
pi
int('x^2/sin(x)')
ans =
int(x^2/sin(x),x)
```

จากตัวอย่างสุดท้ายจะพบว่า MATLAB ไม่สามารถที่จะ integrate ได้ จึงได้คืนค่าเดิมมาให้

10.4 Symbolic Transformation

การหา transformation ของ function เป็นส่วนหนึ่งที่สำคัญในการศึกษาทางวิศวกรรม โดยเฉพาะในเรื่องของการหา Fourier Transform และ Laplace Transform ซึ่ง Symbolic Toolbox ได้มี transformation ในส่วนนี้ไว้ดังมีรายละเอียดดังนี้

❶ Laplace Transform

Laplace Transform เป็นการเปลี่ยนค่า function f(t) ใน time domain ให้เป็น function L(s) ใน s-domain ตาม transformation

$$L(s) = \int_0^\infty f(t)e^{-st}dt$$

ซึ่ง Laplace Transform นี้จะมีประโยชน์ในการแก้สมการ differential equation และในด้านอื่นๆ เป็นอย่างมาก โดยเฉพาะการศึกษาในเรื่องของ Control System สำหรับ MATLAB จะใช้คำสั่ง **laplace** ซึ่งมีรูปแบบดังนี้

L = laplace(F,var1,var2)

เป็นการหา Laplace transform ของ symbolic expression F ใน var1-domain เพื่อเปลี่ยนเป็น function ใน var2-domain นั่นคือจะได้ผลเป็น L(var2) จากการ transform F(var1) ซึ่งถ้าหากไม่กำหนด option var1, var2 มาให้ MATLAB จะกำหนด var1ให้เป็น t และ var2 ให้เป็น s นอกจากนั้น F, var1 และ var2 จะต้องนิยามตามคำสั่ง **sym** หรือได้ค่าจากตัวแปรที่นิยามตามคำสั่ง **sym**

f = ilaplace(L,var1,var2)

เป็นการหา inverse Laplace transform ของ symbolic expression L โดย L โดย option var1 และ var2 จะมีค่าเป็น t และ s ตามลำดับ นั่นคือจะได้ f(var2) จากการ inverse transform L(var1) นอกจากนั้น F,

var1 และ var2 จะต้องนิยามตามคำสั่ง **sym** หรือได้ค่าจากตัวแปรที่นิยามตามคำสั่ง **sym**

ตัวอย่างเช่น ต้องการทำ Laplace transform ของ

$$f(t) = 5e^{-2at}\cos\omega t$$

ซึ่งจะสามารถทำโดย MATLAB ได้ดังนี้

```
syms a s t w
f=5*exp(-2*a*t)*cos(w*t);
L=laplace(f,t,s)
L =
5*(s+2*a)/((s+2*a)^2+w^2)
ilaplace(L,s,t)
ans =
5*exp(-2*a*t)*cos(w*t)
```

❷ Fourier Transform

สำหรับการเปลี่ยนค่า function ให้ออกมาให้อยู่ในรูป function ที่ซ้ำตัวเอง หรือการเปลี่ยนค่า function ใน time-domain ให้มาอยู่ใน frequency-domain จะมีประโยชน์มากโดยเฉพาะวิศวกรที่ทำงานเกี่ยวข้องกับการวิเคราะห์สัญญาณ ซึ่ง Fourier Transform มีรูปแบบดังนี้

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

ส่วนการ transform จาก frequency-domain กลับมาอยู่ใน time-domain จะใช้ Inverse Fourier Transform ซึ่งมีรูปแบบเป็น

$$f(t) = \int_{-\infty}^{\infty} F(\omega)e^{j\omega t}dt$$

สำหรับคำสั่งของ MATLAB สำหรับ symbolic Fourier transform จะมีดังนี้

F = fourier(f, var1, var2)

เป็นการหา Fourier transform ของ symbolic f ซึ่งมีค่ากำหนดเบื้องต้นให้มีค่า independent variable เป็น x และให้ค่าออกมาในรูป function ของ w. หากต้องการเป็นอย่างอื่นสามารถกำหนด independent variable เป็น var1 และให้ผลลัพธ์เป็น var2 ได้

f = ifourier(F,var1,var2)

เป็นการหา inverse Fourier transform ของ symbolic F ซึ่งมีค่ากำหนดเบื้องต้นให้มีค่า independent variable เป็น w และให้ค่าออกมาในรูป function ของ x หากต้องการเป็นอย่างอื่นสามารถกำหนด independent variable เป็น var1 และให้ผลลัพธ์เป็น var2 ได้

ตัวอย่างเช่น

```
syms x t
fourier(exp(-x^2),x,t)
ans =
pi^(1/2)*exp(-1/4*t^2)

f=ifourier(F,t,x)
f =
1/2*4^(1/2)*exp(-x^2)

simple(f)
ans =
exp(-x^2)
```

สำหรับการใช้ Fourier Transform นี้เราจะพบ function พิเศษ 2 function คือ Step function u(t) และ Impulse function δ(t) โดย Step function จะมีนิยามเป็น

$Ku(t-a) = 0$ เมื่อ $t < 0$ และ

$Ku(t-a) = a$ เมื่อ $t > 0$

ส่วน impulse function จะมีนิยามเป็น

$$K\delta(t-a) = 0 \qquad \text{for } t \neq a$$

$$\int_{-\infty}^{\infty} K\delta(t-a)dt = K \qquad \text{for } t = a$$

สำหรับใน MATLAB จะเรียกชื่อสอง function นี้ตามชื่อนักคณิตศาสตร์ที่ใช้ function ทั้งสองในงานของเขาเป็นอย่างมาก คือ จะเรียก Step function เป็น Heaviside(t) และ Impulse function เป็น Dirac(t) ดังนั้นในคำตอบหากเราพบว่ามีการแสดงค่าเหล่านี้ออกมาขอให้ทราบว่าเป็นการกล่าวถึง function ทั้งสองนี้

❸ Z-Transform

สำหรับ transform ที่ผ่านมาทั้งสองแบบนั้นเป็นการ transform ของ function ที่ต่อเนื่อง สำหรับระบบที่เป็น discrete-time domain จะใช้ Z-transform ซึ่งมีรูปแบบเป็น

$$F(z) = \sum_{n=0}^{\infty} f(n)z^{-n}$$

เมื่อ z เป็น complex สำหรับ MATLAB จะใช้ function ztrans และ iztran สำหรับ Z-transform และ inverse Z-transform โดยมีรูปแบบดังนี้

F=ztrans(f,var1,var2) เป็น Z-transform ของ symbolic function f โดยมีค่ากำหนดเบื้องต้นของ independent variable เป็น n และจะให้ค่า F เป็น function ของ z หากต้องการเปลี่ยนเป็นอย่างอื่นให้กำหนดค่า var1 และ var2 ตามลำดับ

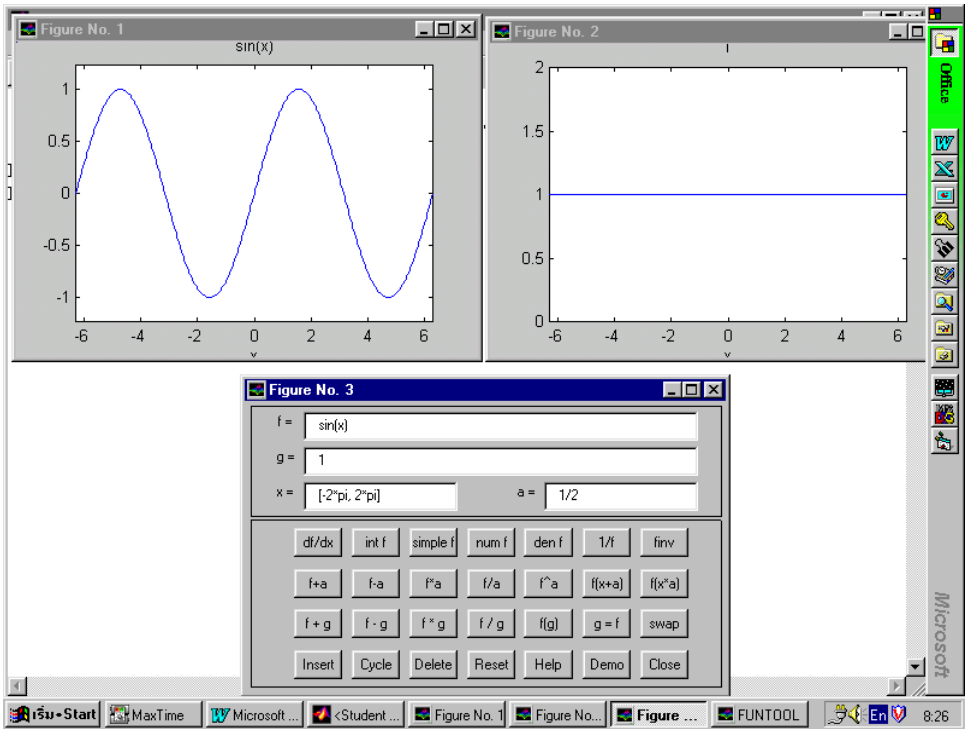
f=iztran(F,var1,var2) เป็น inverse Z-transform ของ symbolic function F โดยมีค่ากำหนดเบื้องต้นของ independent variable เป็น z และจะให้ค่า f เป็น function ของ n หากต้องการเปลี่ยนเป็นตัวแปรอื่นให้กำหนดค่า var1 และ var2 ตามลำดับ

ตัวอย่างเช่น

```
syms k n w z
ztrans(2^n)
ans =
z/(z-2)


iztrans(ans)
ans =
2^n

F=ztrans(cos(n*k),k,z)
F =
z*(-cos(n)+z)/(-2*z*cos(n)+z^2+1)

F=ztrans(cos(n*k),n,w)
F =
w*(-cos(k)+w)/(-2*w*cos(k)+w^2+1)

f=iztrans(exp(x/z),z,k)
f =
x^k/k!

F=ztrans(f,k,z)
F =
exp(x/z)
```

10.5 Function Calculator

สำหรับ symbolic toolbox จะมี M-file อยู่ file ที่มีประโยชน์มากในการคำนวณ function ต่างๆ ไม่ว่าจะเป็นการหาค่าอนุพันธ์ อินทิเกรท หรืออื่นๆที่มี function อยู่ใน symbolic toolbox และการคำนวณก็จะคำนวณในเชิงสัญลักษณ์ทั้งสิ้น สำหรับคำสั่งที่เรียก function calculator ขึ้นมาใช้งานก็คือ **funtool** เมื่อเรียก **funtool** แล้วจะปรากฏหน้าต่าง figure windows ขึ้น 3 หน้าต่าง ตามรูป



สำหรับหน้าต่างที่ 1 จะเป็นกราฟแสดง function f(x) หน้าต่างที่ 2 จะเป็นกราฟแสดง function g(x) และหน้าต่างที่ 3 จะมีลักษณะคล้ายเครื่องคำนวณทั่วไป ซึ่งเป็นที่ใช้กำหนด function f(x), g(x) และคำสั่งต่างๆ ที่เกี่ยวกับการคำนวณ function f(x) สำหรับหน้าต่างที่ 3 จะประกอบด้วยส่วนต่างๆดังนี้

❶ ส่วนที่ 1 ส่วนการกรอกข้อความ

ในส่วนที่ 1 ซึ่งจะอยู่ส่วนบนของหน้าต่างจะประกอบด้วยช่องกรอกข้อความต่อไปนี้

1. ช่องกรอกค่า function f(x) โดยการกรอกค่าก็จะต้องกรอกตาม MATLAB expression ที่กล่าวผ่านมาแล้วทุกประการ เมื่อกรอกค่าแล้วให้กด Enter จากนั้น MATLAB จะทำการเขียนกราฟ f(x) ในหน้าต่างที่ 1
2. ช่องกรอกค่า function g(x) โดยการกรอกค่าก็จะต้องกรอกตาม MATLAB expression ตามที่กล่าวผ่านมาแล้วทุกประการ เมื่อกรอกค่าแล้วให้กด Enter จากนั้น MATLAB จะทำการเขียนกราฟ g(x) ในหน้าต่างที่ 2
3. ช่องกรอกค่าช่วงของ x เป็นการกรอกค่าว่าจะให้ x มีค่าเริ่มจากค่าใดและสิ้นสุดที่ค่าใด โดยในช่องต้องกรอกในรูปแบบ $[x_{min}, x_{max}]$ เมื่อ x_{min} คือค่าน้อยที่สุดและ x_{max} คือค่ามากที่สุดของ x ตามลำดับ
4. ช่องกรอกค่า parameter a หากว่าเราต้องการพิจารณาผลกระทบของ function เมื่อมีการเปลี่ยนแปลงค่า parameter เราสามาถกรอกค่า f(x) และ g(x) ให้อยู่ในรูปที่มี parameter a รวมอยู่ด้วย และเมื่อเราเปลี่ยนค่า a ค่า f(x) และ g(x) ก็จะเปลี่ยนไปด้วย

❷ ส่วนที่ 2 ส่วนการคำนวณ

สำหรับส่วนที่ 2 จะเป็นส่วนที่ใช้ในการคำนวณ function ที่เรากำหนดค่าไว้ตามส่วนที่ 1 ซึ่งโดยส่วนใหญ่จะเป็นการคำนวณค่าของ function f(x) และค่าที่ได้จะเป็นค่าใหม่ของ f(x) ซึ่งเราสามารถแบ่งส่วนที่ 2 นี้ออกได้เป็น 4 แถวตามลักษณะของคำสั่งดังต่อไปนี้

- แถวที่ 1 แถวบนสุดนี้จะเป็นการคำนวณที่เกี่ยวข้องกับ f(x) เท่านั้นและค่าที่ได้จะเป็นการแทนค่า f(x) เดิมก่อนการคำนวณ โดยมีปุ่มต่างๆ ดังนี้

ปุ่ม	คำสั่ง
df/dx	หาอนุพันธ์ของ f(x) เชิงสัญลักษณ์
int f	หาอินทิเกรดของ f(x) เชิงสัญลักษณ์
simple f	ทำให้ f(x) อยู่ในรูปอย่างง่าย (ถ้าทำได้)
num f	แยกเฉพาะเศษของ f(x) ออกมาหาก f(x) อยู่ในรูปเศษส่วน
den f	แยกเฉพาะส่วนของ f(x) ออกมาหาก f(x) อยู่ในรูปเศษส่วน
1/f	แทน f(x) ด้วย 1/f(x)
finv	แทน f(x) ด้วย inverse function ของมัน

สำหรับการหาค่าอินทิเกรทและ inverse ของ function บาง function อาจไม่สามารถที่จะทำได้เนื่องจากเป็นการคำนวณเชิงสัญลักษณ์อาจจะไม่สามารถหาค่าคำตอบได้นั่นเอง

- แถวที่ 2 ในแถวนี้จะเป็นการพิจารณาการเปลี่ยนแปลงค่าของ function f(x) เนื่องจาก parameter a สำหรับปุ่มต่างๆ มีดังนี้

ปุ่ม	คำสั่ง
f +a	แทน f(x) ด้วย f(x) + a ด้วยวิธีเชิงสัญลักษณ์
f -a	แทน f(x) ด้วย f(x) - a ด้วยวิธีเชิงสัญลักษณ์
f *a	แทน f(x) ด้วย f(x) * a ด้วยวิธีเชิงสัญลักษณ์
f / a	แทน f(x) ด้วย f(x) / a ด้วยวิธีเชิงสัญลักษณ์
f^a	แทน f(x) ด้วย f(x) ^ a ด้วยวิธีเชิงสัญลักษณ์
f(x+a)	แทน f(x) ด้วย f(x + a) ด้วยวิธีเชิงสัญลักษณ์
f(x*a)	แทน f(x) ด้วย f(x * a) ด้วยวิธีเชิงสัญลักษณ์

- แถวที่ 3 เป็นแถวที่มีปุ่มที่เกี่ยวข้องกับ f(x) และ g(x) ซึ่งเป็นปุ่มที่เป็น function operation เสียเป็นส่วนใหญ่ โดยมีคำสั่งดังนี้

ปุ่ม	คำสั่ง
f +g	แทน f(x) ด้วย f(x) + g(x) ด้วยวิธีเชิงสัญลักษณ์
f – g	แทน f(x) ด้วย f(x) - g(x) ด้วยวิธีเชิงสัญลักษณ์
f *g	แทน f(x) ด้วย f(x) * g(x) ด้วยวิธีเชิงสัญลักษณ์
f /g	แทน f(x) ด้วย f(x) / g(x) ด้วยวิธีเชิงสัญลักษณ์
f(g)	แทน f(x) ด้วย f(g(x)) ด้วยวิธีเชิงสัญลักษณ์
g =f	แทน g(x) ด้วย f(x)
swap	สลับค่าระหว่าง f(x) และ g(x)

- แถวที่ 4 เป็นแถวที่เกี่ยวข้องกับการควบคุมการทำงานของ Function Calculator โดยสามปุ่มแรกจะเป็นส่วนที่ใช้จัดการกับ function f(x) ที่มีอยู่ใน list ที่ funtool ใช้อยู่ เราสามารถปรับเปลี่ยน สลับที่ หรือลบค่า f(x) ที่มีอยู่ในรายการได้ สำหรับ function ที่อยู่ในรายการจะบรรจุอยู่ใน file ชื่อ fxlist ซึ่งค่าเริ่มต้นของ fxlist จะมี function ที่น่าสนใจอยู่หลาย function เลยทีเดียว สำหรับการทำงานของสามปุ่มแรกเป็นดังนี้
  - **Insert** แทรกค่า f(x) ที่ปรากฏอยู่ในขณะนั้นลงไปใน list
  - **Cycle** หมุนค่า f(x) ไปเรื่อย ๆ ตามค่า f(x) ที่มีอยู่ใน list
  - **Delete** ลบค่า function f(x) ที่ปรากฏอยู่ออกจาก list (หากว่ามีอยู่ใน list)

สำหรับปุ่มที่เหลือมีหน้าที่ดังต่อไปนี้

- **Reset** ตั้งค่า f, g, x, a and fxlist ให้กลับมาอยู่ที่ค่าเริ่มต้น
- **Help** แสดงข้อความช่วยแนะนำการทำงานของ funtool
- **Demo** สาธิตการทำงานของ funtool
- **Close** เลิกการทำงานและปิดหน้าต่างทั้งสาม

การทำความเข้าใจกับการทำงานของ funtool ที่ดีที่สุดคงจะเป็นการทดลองทำกับเครื่องโดยตรง เพราะจะทำให้เราเข้าใจได้อย่างรวดเร็ว นอกจากนั้นจะพบว่าการใช้งานนั้นไม่ยากเลย สำหรับการทำงานของ funtool นอกจากจะเป็นการช่วยในการวิเคราะห์ function แล้ว ยังแสดงให้เห็นถึงความสามารถของ MATLAB ในการทำงานในลักษณะ Graphics User Interface (GUI) ซึ่งเป็นการทำงานโดยใช้การติดต่อคำสั่งส่วนใหญ่ผ่านทางรูปภาพและ mouse โดยความเป็นจริงแล้ว MATLAB มีเครื่องมือ

ที่จะช่วยท่านสร้าง GUI อยู่แต่เนื่องจากการสร้าง GUI เป็นเรื่องที่มีรายละเอียดมากผู้เขียนจึงจะไม่ขอกล่าวถึงในที่นี้ สำหรับผู้ที่สนใจสามารถอ่านได้ในส่วนที่ 2 ของเอกสารชุดนี้

10.6 คำสั่งของ Symbolic Math Toolbox : Student Edition

Symbolic Math Toolbox Version 2.0 นี้เป็น version สำหรับ Student Edition ซึ่งออกเผยแพร่ประมาณต้นปี ค.ศ. 1997 โดยได้มีการเปลี่ยนแปลงจาก version แรกเป็นอย่างมากผู้เขียนจึงอยากที่จะให้รายละเอียดเกี่ยวกับ symbolic toolbox นี้ให้มากที่สุด แต่อย่างไรก็ตามคงจะไม่สามารถบรรจุได้อย่างละเอียด ดังนั้นสำหรับคำสั่งอื่นๆ ผู้เขียนได้รวบรวมไว้เป็นหมวดๆ หากต้องการทราบรายละเอียดมากกว่านี้ก็สามารถที่จะใช้คำสั่ง help ช่วยได้ ซึ่งมีรายการดังต่อไปนี้

Calculus.

diff	Differentiate.
int	Integrate.
limit	Limit.
taylor	Taylor series.
jacobian	Jacobian matrix.
symsum	Summation of series.

Linear Algebra.

diag	Create or extract diagonals.
triu	Upper triangle.
tril	Lower triangle.
inv	Matrix inverse.
det	Determinant.
rank	Rank.
rref	Reduced row echelon form.
null	Basis for null space.
colspace	Basis for column space.
eig	Eigenvalues and eigenvectors.
svd	Singular values and singular vectors.
jordan	Jordan canonical (normal) form.
poly	Characteristic polynomial.
expm	Matrix exponential.

Simplification.

simplify	Simplify.
expand	Expand.
factor	Factor.
collect	Collect.
simple	Search for shortest form.
numden	Numerator and denominator.
horner	Nested polynomial representation.
subexpr	Rewrite in terms of subexpressions.
subs	Symbolic substitution.

Solution of Equations.

solve	Symbolic solution of algebraic equations.
dsolve	Symbolic solution of differential equations.
finverse	Functional inverse.
compose	Functional composition.

Variable Precision Arithmetic.

vpa	Variable precision arithmetic.
digits	Set variable precision accuracy.

Integral Transforms.

fourier	Fourier transform.
laplace	Laplace transform.
ztrans	Z transform.
ifourier	Inverse Fourier transform.
ilaplace	Inverse Laplace transform.
iztrans	Inverse Z transform.

Conversions.

double	Convert symbolic matrix to double.
poly2sym	Coefficient vector to symbolic polynomial.
sym2poly	Symbolic polynomial to coefficient vector.
char	Convert sym object to string.

Basic Operations.

sym	Create symbolic object.
syms	Short cut for constructing symbolic objects.
findsym	Determine symbolic variables.
pretty	Pretty print a symbolic expression.
latex	LaTeX represention of a symbolic expression.
ccode	C code represention of a symbolic expression.
fortran	Fortran represention of a symbolic expression.

Special Functions.

sinint	Sine integral.
cosint	Cosine integral.
zeta	Riemann zeta function.
lambertw	Lambert W function.

String handling utilities.

isvarname	Check for a valid variable name.
vectorize	Vectorize a symbolic expression.

Pedagogical and Graphical Applications.

rsums	Riemann sums.
ezplot	Easy to use function plotter.
funtool	Function calculator.

Demonstrations.

symintro	Introduction to the Symbolic Toolbox.
symcalcdemo	Calculus demonstration.
symlindemo	Demonstrate symbolic linear algebra.
symvpademo	Demonstrate variable precision arithmetic
symrotdemo	Study plane rotations.
symeqndemo	Demonstrate symbolic equation solving.

บทที่ 11 M-BOOK

11.1 แนะนำ M-book

การเขียนรายงานทางวิศวกรรมเรามักจะพบปัญหาประจำอย่างหนึ่งคือโปรแกรม word processing ที่เราใช้ไม่สามารถจะช่วยในการคำนวณหรือเขียนกราฟได้ตามที่เราต้องการ ดังนั้นวิธีการที่เราใช้ก็คือใช้โปรแกรมอีกโปรแกรมหนึ่งมาช่วยในการคำนวณ แสดงผล เขียนกราฟ หรืออื่น ๆ จากนั้นเราก็จะนำผลของโปรแกรมนั้นมาปะลงบน word processing ของเรา

การตัดปะนี้อาจจะเป็นการตัดปะลงบนกระดาษจริงๆ หรือการใช้คำสั่ง copy รูปภาพจากหน้าต่างหนึ่งมาลงบนอีกหน้าต่างหนึ่งและนำผลจากการคำนวณมาพิมพ์ลงไปใหม่ อย่างไรก็ตามการทำลักษณะอย่างนี้ทำให้มีปัญหาตามมาไม่น้อย เช่น เราต้องพิมพ์ข้อมูลทั้งสองครั้ง คือ ทั้งบน word processing และโปรแกรมที่ใช้คำนวณหรือถ้าว่าเราพบว่าข้อมูลผิดพลาดการแก้ไขก็จะลำบากหรืออาจจะเหมือนกับต้องเริ่มทำงานใหม่เกือบทั้งหมดก็ได้

ความจริงแล้วน่าจะเป็นการดีกว่าหากเราสามารถที่จะเขียนข้อความลงบน word processing แล้ว word processing นั้นสามารถคำนวณหาคำตอบและเขียนกราฟให้กับเราได้ตามที่เราต้องการก็คงจะเป็นการสะดวกไม่น้อยซึ่ง M-book ก็คือทางแก้ปัญหาเหล่านี้นั่นเอง

M-book เป็นการทำงานร่วมกันระหว่างโปรแกรม 2 โปรแกรม คือ Microsoft Word for Window ซึ่งจะทำหน้าที่เป็น program word processing ซึ่งก็เป็นที่ยอมรับกันว่าเป็น program ทางด้าน word processing ที่มีผู้ใช้มากที่สุดในโลกนำมาทำงานร่วมกับ MATLAB ซึ่งจะทำหน้าที่เป็นเครื่องจักรในการคำนวณ หาคำตอบ เขียนกราฟ โดยการรับคำสั่ง จะรับจาก Word โดยตรง และก็จะแสดงผลกลับไปที่ Word โดยตรง เช่นกันในทางปฏิบัติ MATLAB จะทำงานอยู่บน background และทำหน้าที่เป็น เครื่องจักรในการคำนวณให้กับ Word โดยผู้ใช้ไม่จำเป็นต้องสัมผัสกับ MATLAB เลย สำหรับเอกสารที่ได้จากการทำงานร่วมของ program ทั้งสองนี้เราเรียกว่า M-book แม้ว่าเราจะ Save file นี้อยู่ในรูปแบบของ Word document file (.doc) ก็ตาม

ระบบที่ต้องการเมื่อจะใช้ M-book มีดังนี้ (สำหรับ Window 95 system)

- MATLAB 4 หรือ MATLAB 5
- Microsoft Word for Window 6.0 หรือ 7.0

สำหรับ Word 97 นั้น ทางบริษัท Math Works ได้แจ้งว่าในเบื้องต้นจะไม่สามารถทำงานร่วมกับ MATLAB 5.0 ได้ แต่ผู้ใช้ที่ลงทะเบียนถูกต้องกับทางบริษัทสามารถ download file ใหม่ผ่านทาง Internet ที่ **www.mathworks.com** ได้ ส่วนผู้ใช้ MATLAB 5.1 จะสามารถใช้งานร่วมกับ Word 97 ได้เลย

- ในการติดตั้งโปรแกรมต้องติดตั้ง Word ก่อน แล้วจึงติดตั้ง MATLAB ต่อมาภายหลัง เพราะในการติดตั้ง MATLAB ครั้งแรก MATLAB จะถามถึง directory ของ Word เพื่อจะติดตั้ง template

M-book นั้นเป็น template หนึ่งที่ทำงานอยู่บน Word ผู้ที่คุ้นเคยกับการใช้ Word คงจะทราบถึงความหมายของ template ของโปรแกรมนี้เป็นอย่างดี ซึ่งสำหรับงานปกติ Word จะใช้ template ที่ชื่อ Normal.dot แต่สำหรับการทำงานร่วมกับ MATLAB แล้ว Word จะต้องใช้ template ที่ชื่อ M-book.dot ซึ่ง template นี้จะรวบรวมคำสั่งและวิธีการของการถ่ายโอนข้อมูลระหว่าง Word และ MATLAB ไว้เป็น macro

11.2 การทำงานเบื้องต้น

เมื่อเราติดตั้ง Word และ MATLAB พร้อมด้วย M-book เรียบร้อยแล้ว เรียก Word ขึ้นทำงานครั้งแรก Word จะใช้ template ปกติคือ Normal.dot หากว่าเราต้องการสร้างหรือแก้ไข document file ที่เป็น M-book เราก็จะสามารถทำได้โดย

- ถ้าเป็นการแก้ไขใช้คำสั่งการเปิด file ของ Word ปกติ
- ถ้าเป็นการสร้าง file ใหม่ให้เลือกคำสั่ง New จาก File Menu จากนั้นจะปรากฏ New dialog box ขึ้น เลือก M-book template จากนั้น Word ก็จะสร้าง document file ขึ้นใหม่ซึ่งพร้อมจะทำงานร่วมกันระหว่าง Word กับ MATLAB

✎ ไม่ว่าจะเป็นการสร้าง file ใหม่หรือแก้ไข file เดิมก็ตามทุกครั้งที่เราเรียก M-book มาใช้งาน จะเกิดสิ่งเหล่านี้ขึ้น

- MATLAB จะเริ่มทำงานโดยอัตโนมัติ (ถ้าหากขณะนั้นยังไม่เริ่มทำงาน) และจะมีการทำ Dynamics Data Exchange (DDE) ระหว่าง Word และ MATLAB
- จะมีการกำหนด macros ขึ้นเพื่อให้เกิดการรับ-ส่ง DDE ผ่านทาง Cells ซึ่งเป็นเส้นทางติดต่อระหว่าง MATLAB และ Word
- กำหนด styles หรือรูปแบบตัวหนังสือต่าง ๆ ที่จะใช้กับ text และ cells
- เพิ่ม menu Notebook ขึ้นบน Menu bar ของ Word

✎ หากว่าท่านกำลังทำงานอยู่บน MATLAB แล้วต้องการจะเรียก Word เพื่อสร้าง M-book ก็สามารถทำได้โดย ที่ prompt ของ MATLAB พิมพ์คำว่า

```
» notebook
```

Word จะทำงานโดยอัตโนมัติ พร้อมทั้งเรียก template M-book.dot ขึ้นมาพร้อมทำงาน

✎ หากว่าเรามี document file ธรรมดาอยู่แล้วต้องการจะเปลี่ยนเป็น M-book document ก็สามารถทำได้โดยเปิด document ใหม่ที่ใช้ template M-book ด้วยคำสั่ง New ตามที่กล่าวมาแล้ว เมื่อได้ M-book document ใหม่ที่ว่างเปล่าแล้วเราก็สามารถใช้คำสั่ง file ที่ Insert Menu จากนั้นเมื่อ insert file dialog box ปรากฏขึ้นเราก็จะสามารถเลือก file ที่ต้องการเปลี่ยนเป็น M-book ได้

11.3 Input & Output Cells

การส่งคำสั่ง MATLAB ผ่านมาจาก Word และการรับผลการคำนวณของ MATLAB ส่งกลับไปที่ Word จะเป็นการส่งผ่านส่วนที่เรียกว่า cell เนื่องจากการทำงานบน Word อาจจะมีข้อความตัวหนังสือหรือสมการอื่นๆ มากมาย แต่ MATLAB จะคำนวณค่าต่างๆ เฉพาะที่อยู่ใน input cell เท่านั้น และ MATLAB จะคำนวณเฉพาะเวลาที่เราสั่งให้มีการคำนวณเท่านั้นส่วนผลทุกรูปแบบที่ได้จะส่งกลับมาทาง output cell หากเรามีการเปลี่ยนแปลงค่าภายใน input cell ขึ้น MATLAB จะไม่มีการเปลี่ยนแปลงผลลัพธ์จนกว่าเราจะสั่งให้ MATLAB คำนวณอีกครั้งหนึ่ง

❶ การสร้าง Input cell

Input cell ที่เราสร้างขึ้นอาจมีลักษณะเป็น cell ที่รวมอยู่ใน text หรือเป็น cell ที่แยกออกมาจาก text ที่พิมพ์ปกติอีกบรรทัดหนึ่ง นอกจากนั้น input cell อาจจะเป็น คำสั่งคำสั่งเดียวหรือประกอบด้วยคำสั่งหลาย ๆ คำสั่งก็ได้ การสร้าง input cell ที่ง่ายที่สุดมีขั้นตอนดังนี้

- พิมพ์คำสั่งที่ต้องการจะส่งให้ MATLAB ซึ่งเป็นคำสั่งที่เราสั่งบน command window ให้อยู่ในรูป text ปกติ
- Highlight คำสั่ง หรือ ชุดคำสั่งทั้งหมด
- ที่ Notebook Menu เลือก Define Input Cell หรือ กด Alt-D

เมื่อทำตามขั้นตอนนี้แล้ว M-book จะทำการสร้าง input cell ขึ้นโดยจะมีการเปลี่ยนแปลงแบบตัวหนังสือให้อยู่ในรูปแบบของ input style (ซึ่งเราสามารถปรับเปลี่ยน style ให้เป็นไปตามต้องการได้ รายละเอียดจะกล่าวภายหลัง) นอกจากนี้ยังจะปรากฏ cell mark ขึ้น ลักษณะ cell mark จะเป็นวงเล็บใหญ่ [ ] เปิดและปิดวงเล็บของข้อความที่เรากำหนดให้เป็น input cell ขึ้น สำหรับ cell mark นี้จะปรากฏเฉพาะบนจอในการทำงานของ Word แต่เมื่อเราพิมพ์งานจะไม่ปรากฏ cell mark ขึ้นในเอกสารที่พิมพ์ออกมา นอกเหนือจากนั้นเรายังสามารถกำหนดว่าเราต้องการจะมองเห็น cell mark หรือไม่ในระหว่างการทำงานปกติได้อีกด้วย

หากไม่มีการเปลี่ยนแปลง style เบื้องต้นของ M-book ตัวอักษรของ input cell จะเป็นตัวอักษร Courier New ตัวหนาขนาด 10 points สีเขียวเข้ม

ตัวอย่างการสร้าง input cell เป็นดังนี้

พิมพ์คำสั่งปกติในลักษณะ normal text

```
A=10;
B=20;
C=A*B
```

highlight ตัวหนังสือที่ต้องการเปลี่ยนให้อยู่ใน input cell

```
A=10;
B=20;
C=A*B
```

ที่ Notebook Menu เลือก Define input cell หรือ กด Alt-D ผลที่ได้จะเป็น

```
[A=10;
B=20;
C=A*B]
```

❷ การรวม input cell เข้าด้วยกัน

ในการสร้าง input cell แต่ละ cell ขึ้นหากมีการสั่งคำสั่งไปสู่ MATLAB การส่งไปทีละ cell ซึ่งการส่งแต่ละครั้งเมื่อส่งไปแล้ว MATLAB จะทำการคำนวณ cell นั้นทันที และหากมีผลก็จะส่งผลกลับมาทันทีเช่นกัน ดังนั้น หากว่าคำสั่งของเราเป็นชุดคำสั่งคือต้องการผลการคำนวณหลังจากคำสั่งสุดท้ายทีเดียว สิ่งที่ควรทำก็คือพยายามเลือกชุดคำสั่งเหล่านั้นให้อยู่ใน input cell เดียวกัน หรือเราสามารถจะสร้างแต่ละคำสั่งให้อยู่ใน input cell ของมันเอง แล้วรวมกลุ่ม input cell เหล่านั้นเข้าด้วยกัน โดยขั้นตอนดังนี้

- Highlight input cell ทั้งหมดที่ต้องการรวมกลุ่มกัน
- ที่ Notebook Menu เลือก Group Cell หรือกด Alt-G M-book จะทำการรวมกลุ่ม input cell เหล่านั้นเข้าด้วยกัน สำหรับการเลือก input cell นั้นหากว่าในการเลือกมีการเลือก text ธรรมดาที่ไม่ใช่ input cell รวมอยู่ด้วย M-book จะทำการย้าย text ที่เลือกทั้งหมดมาไว้ข้างหลัง input cell Group นั้น

❸ การใช้ AutoInit Cell

สำหรับ input cell มีแบบพิเศษแบบหนึ่งคือ จะมีการคำนวณครั้งแรกโดยอัตโนมัติเมื่อเราเปิด file นั้นขึ้นมา (Automatic Initialize) ซึ่งเรียกว่า AutoInit Cell สำหรับการเลือก AutoInit Cell นั้นก็คล้ายกับการเลือก input cell ปกติคือ

- พิมพ์ข้อความที่ต้องการให้เป็น input ของ MATLAB
- Highlight คำสั่งหรือชุดคำสั่งนั้น
- ที่ Notebook menu เลือก Define AutoInit Cell (สำหรับคำสั่งนี้ไม่มี short cut key)

เมื่อใช้คำสั่งนี้แล้ว M-book จะเปลี่ยนข้อความที่เลือกเป็น AutoInit Cell โดยจะอยู่ใน Cell mark และจะมีค่าเริ่มต้นของลักษณะตัวหนังสือเป็น Courier New ตัวหนา ขนาด 10 points สีน้ำเงินเข้ม

❹ การส่งคำสั่งไปให้ MATLAB คำนวณ

สำหรับการส่งคำสั่งใน input cell ไปให้ MATLAB คำนวณนั้น ถ้าเป็น input cell ปกติจะส่งเฉพาะเวลาที่เราต้องการ แต่ถ้าเป็น AutoInit Cell จะส่งอัตโนมัติในตอนเปิด M-book ครั้งแรกและสามารถส่งได้อีกในเวลาที่เราต้องการ การส่ง input cell ใน M-book ไปให้ MATLAB คำนวณมีขั้นตอนดังนี้

- หากต้องการส่งเฉพาะ input cell ที่ต้องการให้นำ cursor ไปไว้ใน cell นั้น แล้วจาก Notebook Menu เลือก Evaluate Cell หรือกด Ctrl-Enter
- หากต้องการคำนวณทุก input cell ใน M-book นั้น จาก Notebook Menu เลือก Evaluate M-book หรือกด Alt-R Notebook จะทำการคำนวณจากต้น M-book จนจบ
- หากต้องการคำนวณ input cell บางชุดใน M-book ให้ highlight input cell ที่ต้องการ (อาจมี text ปกติรวมอยู่ด้วยก็ได้ เพราะ text ปกติเหล่านั้นจะไม่ถูกส่งไป MATLAB) จากนั้นที่ Notebook Menu เลือก Evaluate cell หรือ กด Ctrl-Enter แล้ว M-book จะทำการคำนวณจาก input cell แรกที่เลือกจนถึง cell สุดท้ายที่เลือก

❺ การแสดงผล

Notebook จะแสดงผลออกมาทาง output cell ซึ่ง output cell นี้ ก็จะมี ข้อความ,ตัวเลข หรือรูปภาพปกติเหมือนกับที่เราให้คำสั่งบน Command Window ของ MATLAB โดย Output cell ที่ได้จะออกมาปรากฏใน M-book ต่อจาก input cell ที่มี output นั้นเสมอ หากว่ามี text ปกติเชื่อมต่อจาก input cell อยู่ output cell นี้ก็จะเข้าไปแทรกอยู่ระหว่าง input cell กับ text นั้น ลักษณะทั่วไปของ input cell มีดังนี้

- Output cell จะบรรจุข้อความแสดงผลทุกอย่างเหมือนที่ปรากฏใน Command Windows ไม่ว่าจะเป็นผลลัพธ์, Echo, ข้อความแสดงความผิดพลาดหรือรูปภาพ
- ถ้าหากว่าไม่มีการแสดงผลใด ๆ จากคำสั่งของ input cell ก็จะไม่มี output cell เกิดขึ้น

- ถ้าหากว่ามีผลการคำนวณที่ต้องแสดงแต่ยังไม่มี output cell อยู่ Notebook จะสร้าง output cell ขึ้นใหม่ อย่างไรก็ตามหากว่า input cell นั้นเดิมมี output cell อยู่แล้ว Notebook จะแทน output cell เดิมด้วย output cell ใหม่
- ถ้า input cell เป็นชุดคำสั่งหรือเป็นกลุ่มของ input cell ที่มี output มากกว่า 1 คำสั่ง Notebook จะสร้าง output cell เท่ากับจำนวน output ที่ต้องการและ output จะเรียงลำดับว่าตัวเลขหรือตัวหนังสือที่เป็นผลก่อนผลที่เป็นรูปภาพ ไม่ว่าคำสั่งของ input cell จะมีลำดับเป็นอย่างไร
- ตัวหนังสือของ output จะเป็นแบบ Courier New ขนาด 10 points โดยผลลัพธ์จะเป็นตัวหนาสีน้ำเงิน ส่วนข้อความแสดงความผิดพลาดจะเป็นตัวหนังสือหนาสีแดง
- สำหรับรูปแบบการแสดงผลของตัวเลขและขนาดของรูปภาพสามารถกำหนดได้โดยไปที่ Notebook Menu แล้วเลือก option ซึ่งจะกล่าวถึงในภายหลัง
- เราสามารถเปลี่ยน input cell และ output cell เป็น text ธรรมดาได้โดย highlight cell ที่ต้องการเปลี่ยนแล้วที่ Notebook Menu เลือก Undefined cell หรือกด Alt-U
- หากต้องการลบ output cell ใด ให้ Highlight output cell นั้น แล้วที่ Notebook Menu เลือก Purge Output Cell หรือกด Alt-P

❻ ข้อแนะนำเกี่ยวกับ Input และ Output Cell

- ในกรณีที่ท่านใช้ M-book มากกว่า 1 file ทำงานในขณะเดียวกันแล้วมีการส่งตัวแปรไปคำนวณที่ MATLAB นั้นเนื่องจาก MATLAB จะทำงานอยู่เพียง “copy” เดียวและทุก M-book จะอาศัยที่ทำงานเดียวกันดังนั้นหากเราใช้ตัวแปรตัวเดียวกันแต่ค่าต่างกันในแต่ละ M-book การคำนวณอาจผิดพลาดขึ้นได้ ดังนั้นหากมีความจำเป็นต้องทำงานบน M-book หลาย file พร้อมกันต้องระวังในเรื่องการตั้งชื่อตัวแปรไม่ให้ซ้ำกันหรืออาจใช้คำสั่ง Clear เป็น input cell ที่ต้น M-book แต่ละอัน ยกเว้นเราตั้งใจที่จะส่งผ่านค่าเหล่านั้นข้าม M-book จริง ๆ
- การทำงานของ M-book จะมีการเปลี่ยนแปลง output เฉพาะเมื่อเราต้องการเท่านั้น การเปลี่ยนแปลง input cell จะไม่มีผลกระทบต่อ output เดิม จนกว่าจะสั่งให้มีการคำนวณใหม่ และการคำนวณ input cell ใด cell หนึ่งใหม่ จะไม่กระทบผลที่ได้จาก input cell อีก cell หนึ่ง แม้ว่าจะใช้ค่าต่อเนื่องกัน ยกเว้นว่าเราจะสั่งให้มีการคำนวณทุก input cell ที่เกี่ยวข้องกันใหม่ทั้งหมด ดังนั้นหากมีการเปลี่ยนแปลงค่า input cell ใดๆ หลังจากที่ได้ผลมาหลายๆ ค่าแล้ว จะเป็นการดีถ้าเราใช้คำสั่ง Evaluate M-book หรือ กด Alt-R เพื่อเป็นการ Update ค่าใหม่ทั้งหมด
- สำหรับในกรณีที่เกิด error ขึ้นในคำสั่งของ input cell ใด cell หนึ่ง เราอาจให้ MATLAB คำนวณ input cell ต่อไปหรือหยุดคำนวณทันทีก็ได้ โดยที่ Notebook Menu เลือก Options จะมี Options Window ปรากฏขึ้นใน Check box ของ Stop evaluating on error ถ้าเรา Check box นี้

MATLAB จะหยุดการคำนวณถ้าเกิด error ใน input cell ใด cell หนึ่ง แต่ถ้าเราปล่อยว่างไว้ เมื่อพบ error แล้ว MATLAB ยังจะคำนวณ input cell อื่น ๆ ต่อไป จนครบทุก input cell

❼ การแก้ไข Graphic Output

- สำหรับ Graphic output เราสามารถควบคุมให้แสดงใน output cell หรือสร้างใหม่ขึ้นบน Graphic window ซึ่งแยกออกจาก M-book ก็ได้ โดยไปที่ Notebook Menu เลือก option เมื่อ Options window ปรากฏขึ้น จะเห็น cheek box ของ Embed Figures in M-book ถ้าเราเลือก box นี้ รูปภาพจะแสดงใน output cell บน M-book แต่ถ้าเราไม่เลือกรูปภาพจะปรากฏขึ้นบน graphic window เหมือนในการสั่งบน command window ปกติ
- คำสั่ง Toggle Graph Output for Cell ใน Notebook Menu เป็นคำสั่งที่ใช้สลับว่าจะยอมหรือไม่ยอมให้มีการแสดงผลของ Graphic ลงใน Output cell โดยเมื่อทำงานครั้งแรก M-book จะยอมให้แสดงภาพใน Output cell (ถ้าเราเลือก Embed Figures in M-book) หากเราไม่ต้องการให้แสดงผลรูปภาพของ input cell ใด ก็ให้นำ cursor ไปนี้ cell นั้นแล้วเลือก Toggle Graph output for cell ใน Notebook Menu รูปภาพที่ปรากฏขึ้นก่อนหน้านี้จะหายไป เหลือแต่เพียง cell mark ถ้าหากว่าเราต้องการเห็นภาพผลอีกครั้งก็ให้ทำซ้ำเดิมประโยชน์ของคำสั่งนี้ก็เพื่อเพิ่มความรวดเร็วในการทำงาน เพราะกรณีที่มีรูปภาพมากๆ Word จะทำงานได้ช้าลง
- การเปลี่ยนแปลงขนาดของ Graphic Output สามารถทำได้โดยกำหนดความกว้างและความสูงของรูปภาพจาก คำสั่ง Option ใน Notebook Menu นั้นเพื่อใส่ค่าความกว้างและความยาวของรูปภาพที่ต้องการหรืออาจให้ Mouse click บริเวณรูปภาพ เราจะเห็น picture holder ปรากฏขึ้น เราสามารถใช้ mouse ปรับตำแหน่งให้เป็นไปตามต้องการได้
- ภาพที่ได้จาก MATLAB เมื่อปรากฏใน M-book จะมีขอบภาพเป็นสีเทาปรากฏอยู่ หากเราไม่ต้องการขอบสีเทา นั้นเราสามารถตัดออกได้ โดย click บริเวณรูปภาพ เมื่อ picture holder ปรากฏขึ้นให้กด shift แล้ว drag holder ของรูปไปในที่ที่ต้องการภาพที่แก้ไขจะมีขนาดเท่ากับ holder ใหม่ โดยไม่มีการเปลี่ยนขนาดของกราฟแต่อย่างใด
- รูปภาพที่ปรากฏขึ้น จะสามารถเลือกให้ใช้สี 16 สี หรือ 256 สีได้ โดยการเลือกจากคำสั่ง option ใน Notebook Menu

❽ Calculate Zone

calculate zone หรือ calc zone คือย่านที่ประกอบด้วย text, input cell, output cell และอื่นๆ เหมือนกับ M-book ทั่วไป เพียงแต่ในการทำ calc zone นี้ M-book จะแยก calc zone ออกเป็นอีก section หนึ่งต่างหาก โดยไม่มีการ brake หน้า แต่ brake section โดยจุดประสงค์ของ calc zone นี้ เพื่อที่จะให้เราสามารถจัดเตรียม input หรือ output ออกจากส่วนอื่นของ M-book การเลือก calc zone สามารถทำได้โดย highlight

ข้อความ, input, output หรืออื่น ๆ ที่ต้องการแยกออกเป็นอีก zone หนึ่ง จากนั้นที่ Notebook Menu เลือก Define calc zone MATLAB จะทำการแยก section ของ calc zone โดยอัตโนมัติ ส่วนการเลือก input cell หรืออื่น ๆ ก็ทำตามปกติ

- หากว่าเราต้องการจะคำนวณ input cell ทั้งหมดที่อยู่ใน zone นั้นให้ใช้คำสั่ง Evaluate calc zone จาก Notebook Menu
- ข้อควรระวังจากการใช้ calc zone ก็คือค่าตัวแปรของ calc zone หนึ่ง ๆ ไม่ได้เป็นค่าเฉพาะใน zone นั้น แต่เป็นค่าตัวแปรของทั้ง M-book

11.4 สรุปคำสั่งบน Notebook Menu

เมื่อเราเลือก M-book Template ขึ้นมาใช้งาน นอกจากจะมีการเปิด program MATLAB ให้ทำงาน (หากไม่ได้ทำงานอยู่)โดยอัตโนมัติและให้ทำงานอยู่บน background แล้ว บน Menu bar ของ Word จะมี Menu เพิ่มขึ้นอีก 1 Menu คือ Notebook Menu ซึ่งภายใต้ Menu นี้จะมีคำสั่งเรียงลำดับจากบนลงล่างดังนี้

ลำดับ	คำสั่ง	Short cut Key	จุดมุ่งหมาย และขั้นตอน
1	**Define input cell**	Alt +D	กำหนด input cell ก่อนเลือกคำสั่งนี้ต้อง highlight text ที่ต้องการกำหนดให้เป็นคำสั่งจะส่งให้ MATLAB ก่อน
2	**Define AutoInit cell**	-	กำหนด Automatic Initialized cell คือ input cell ที่จะมีการคำนวณครั้งแรกโดยอัตโนมัติเมื่อมีการเปิด M-book ขึ้นใช้ ก่อนใช้คำสั่งนี้ต้องมีการเลือก Text เช่นเดียวกับลำดับที่ 1
3	**Define Calc Zone**	-	กำหนด zone สำหรับการคำนวณ ก่อนใช้ คำสั่งต้องมีการเลือก zone ก่อน
4	**Undefined Cells**	Alt + U	ยกเลิก input หรือ output cell แล้วเปลี่ยนเป็น text ธรรมดา ก่อนใช้คำสั่งต้องมีการเลือก cell ที่ต้องการก่อน
5	**Purge output cell**	Alt + p	ลบ output cell ออก ก่อนใช้คำสั่งต้องเลือก cell ที่ต้องการลบก่อน
6	**Group Cell**	Alt + G	รวม Input cell หลาย ๆ Cell เข้าด้วยกันก่อนใช้ต้องมีการเลือก cell เหล่านั้น
7	**Ungroup cell**	-	ยกเลิกคำสั่ง Group cell ตามลำดับ 6
8	**Hide Cell Marks (Show Cell Marks)**	Alt + [	เป็นคำสั่งให้งดแสดง หรือให้แสดงเครื่องหมาย Cell Mark [...] ซึ่งจะเป็นการสั่งทั้ง M-book และคำสั่ง Hide หรือ show ที่จะขึ้นสลับกัน ขึ้นกับว่าขณะนั้นเราสั่งให้แสดง Cell Mark หรือไม่ **Note** Cell Mark ที่ปรากฏบนจอจะไม่ปรากฏในเอกสารเมื่อสั่งพิมพ์
9	**Toggle Graph Output to Cell**	-	เป็นคำสั่งสลับว่าจะให้แสดงรูปภาพบน M-book หรือไม่ ก่อนใช้คำสั่งต้องนำ Cursor ไปไว้ใน input cell ที่จะแสดงผลกราฟนั้น
10	**Evaluate Cell**	Ctrl - Enter	สั่งให้ คำนวณ input cell ที่เลือกอยู่ การเลือกต้อง highlight input cell ที่ต้องการคำนวณทั้งหมด หรือถ้าเป็น cell เดียวอาจใช้ cursor ไปอยู่ใน cell นั้นก็ได้
11	**Evaluate Calc zone**	Atl - Enter	สั่งให้คำนวณ calculate zone ก่อนใช้คำสั่งต้องเลือก calc zone ก่อน

ลำดับ	คำสั่ง	Short cut Key	จุดมุ่งหมาย และขั้นตอน
12	**Evaluate M-book**	Alt - R	สั่งให้ คำนวณ M-book ใหม่ทั้งหมดโดยเริ่มจาก input cell แรกถึงสุดท้าย
13	**Evaluate loop**	Alt - L	สั่งให้ M-book ทำงานเป็น loop หลาย ๆ ครั้ง โดยจะเริ่มจาก Cell แรกที่เลือก จนถึง Cell สุดท้ายที่เลือก ซึ่งเราสามารถจะกำหนดจำนวนครั้งของ loop ได้
14	**Bring MATLAB to Front**	Alt - M	ดึงให้ MATLAB window มาเป็น foreground ซึ่งโดยปกติ MATLAB command window จะทำงานอยู่บน background
15	**Notebook Options..**	-	เลือก option อื่น ๆ คือ 1. **Number format** คือเลือกลักษณะของรูปแบบตัวเลขที่จะให้แสดงผล ซึ่งจะเหมือนกับคำสั่ง format ของ MATLAB 2. **เลือกรูปภาพ**ที่แสดงผลคือ 2.1 เลือกว่าจะแสดงรูปบนM-bookหรือไม่โดยเลือกEmbed Figures in M-book 2.2 ถ้าเลือก 2.1 จะสามารถเลือกชนิดของสีว่าจะใช้ 16 สี หรือ 256 สีได้ นอกจากนั้นยังกำหนดขนาดของรูปภายใต้อีกด้วย 3. **เลือกจะให้ M-book หยุดคำนวณ**เมื่อเกิดความผิดพลาดที่ Input cell หนึ่ง หรือให้ทำต่อไปจนจบโดยการเลือก Stop Evaluation on Error จะทำให้หยุดเมื่อเกิดความผิดพลาด

11.5 การเปลี่ยนแปลงรูปแบบของ M-book Template

สำหรับ M-book Template ที่ MATLAB กำหนดมานั้นมีรูปแบบตัวหนังสือตามตรงนี้

Style	Font	ขนาด	น้ำหนัก	สี
Normal	Times New Roman	10 points	ปกติ	ดำ
AutoInit	**Courier New**	10 points	ตัวหนา	น้ำเงินเข้ม
Error	**Courier New**	10 points	ตัวหนา	แดง
Input	**Courier New**	10 points	ตัวหนา	เขียวเข้ม
Output	Courier New	10 points	ปกติ	น้ำเงิน

ซึ่งแบบตัวหนังสือเหล่านี้อาจไม่เหมาะสมกับงานของเรา หรือ การใช้ภาษาไทย สำหรับ word Thai Edition ซึ่งในส่วนนี้เราสามารถเปลี่ยนแปลงได้ โดยกำหนด style ของแบบตัวอักษรใหม่

ขั้นตอนการกำหนดรูปแบบตัวหนังสือใหม่สามารถกระทำได้ดังนี้

1. ที่ Format Menu เลือก Style

2. Style Window จะปรากฏขึ้น
3. ด้านซ้ายมือของ window จะเป็นชื่อของรูปแบบ (style) ด้านขวามือจะเป็นการแสดงตัวอย่าง
4. เลือก style ที่ต้องการแก้ไข โดย highlight ที่ style นั้น
5. Click ที่ปุ่ม Modify จะปรากฏ Menu bar ขึ้นว่าต้องการจะแก้ไขสิ่งใด
6. เลือก Fonts...
7. จะปรากฏ Fonts window ขึ้น เราสามารถเลือก ตัวหนังสือ , ขนาด น้ำหนักและสีของตัวหนังสือตามต้องการ
8. Click OK หน้าต่าง Fonts จะปิดแล้วกลับมาที่ style window
9. เลือก style อื่น ๆ ตามต้องการ
10. เมื่อเสร็จแล้ว click ที่ปุ่ม Apply

สำหรับรายละเอียดในส่วนนี้ดูได้จากคู่มือการใช้งานของ Word

บทที่ 12 Handle Graphic

ในขั้นต้นเราได้ศึกษาการเขียนกราฟ หรือรูปต่างๆโดยใช้คำสั่งของ MATLAB ไปแล้ว และเราได้พบว่าการใช้คำสั่งต่างๆ นั้นสามารถทำได้โดยง่าย อย่างไรก็ตามหากเราต้องการที่จะแก้ไขรูปแบบต่างๆ ของรูปภาพที่แสดง ส่วนนี้อาจเป็นส่วนที่ไม่ง่ายนัก เพราะคำสั่งที่เราใช้กำหนดลักษณะของรูปกราฟหรือรูปแบบต่างๆ โดยทั่วไปแล้วจะเป็นคำสั่งที่เรียกว่าคำสั่งขั้นต่ำ (Low Level Command) นอกจากนี้การที่เราจะสร้างโปรแกรมที่สามารถติดต่อกับผู้ใช้ทางกราฟฟิกส์ (Graphical User Interface, GUI) เราจำเป็นต้องเข้าใจถึงลักษณะพื้นฐานการเขียนรูปของ MATLAB เสียก่อน

สำหรับการเขียนกราฟขึ้นโดยใช้ MATLAB นั้นขั้นแรกขออธิบายถึงวิธีการ หรือขั้นตอนการเขียนรูปนี้ขึ้นมาก่อน อันดับแรกหากเราต้องการจะเขียนกราฟเส้นตรงขึ้นมาหนึ่งกราฟ เราต้องทำอะไรบ้าง ขั้นแรกเราคงจะต้องวางกระดาษไว้บนพื้นโต๊ะ จากนั้นเขียนแกนลงไปบนกระดาษ แล้วขั้นสุดท้ายจึงพลอตกราฟเส้นตรงนั้นลงไปบนแกน หลักการนี้ก็จะเป็นหลักการเดียวกันที่ MATLAB เขียนกราฟเส้นตรงเส้นนั้น ขั้นแรก MATLAB จะมีพื้นจอคอมพิวเตอร์ ซึ่งเปรียบเสมือนกับพื้นโต๊ะของเรา จากนั้นสร้าง Figure window ขึ้นเพื่อทำหน้าที่เป็นกระดาษ ขั้นต่อไป MATLAB ก็จะสร้าง Axes หรือแกนเพื่อเป็นเครื่องกำหนดขนาดและมิติของกราฟที่อยู่ในระบบแกนนั้น ขั้นตอนการวางรูปกราฟนี้จะสลับกันไม่ได้คือต้องเรียงลำดับจากพื้นจอขึ้นมาที่ Figure window – axes – และเส้นกราฟตามลำดับ โดยไม่สามารถที่จะสลับที่ได้ เพราะเราไม่สามารถที่จะเขียนแกนลงไปโดยไม่มีกระดาษได้ การเรียงลำดับเหล่านี้เราเรียกว่า Parent & Child หรือ Object Hierarchy ซึ่งหมายถึงการจัดลำดับขั้นของ object ต่างๆ ที่ถูกสร้างขึ้นโดย MATLAB นั่นเอง

ดังนั้นในบทนี้ เริ่มต้นเราจะกล่าวถึง Handle ของวัตถุ วิธีการสร้างวัตถุและหา Handle ของวัตถุที่เราสร้างขึ้น จากนั้นเรากล่าวถึงคุณสมบัติบางประการของวัตถุ เพื่อให้ผู้อ่านได้ทำความเข้าใจกับความหมายของ Handle ได้ดียิ่งขึ้น รวมถึงวิธีการตรวจสอบและปรับปรุงแก้ไขคุณสมบัติของวัตถุให้เป็นไปตามที่เราต้องการ

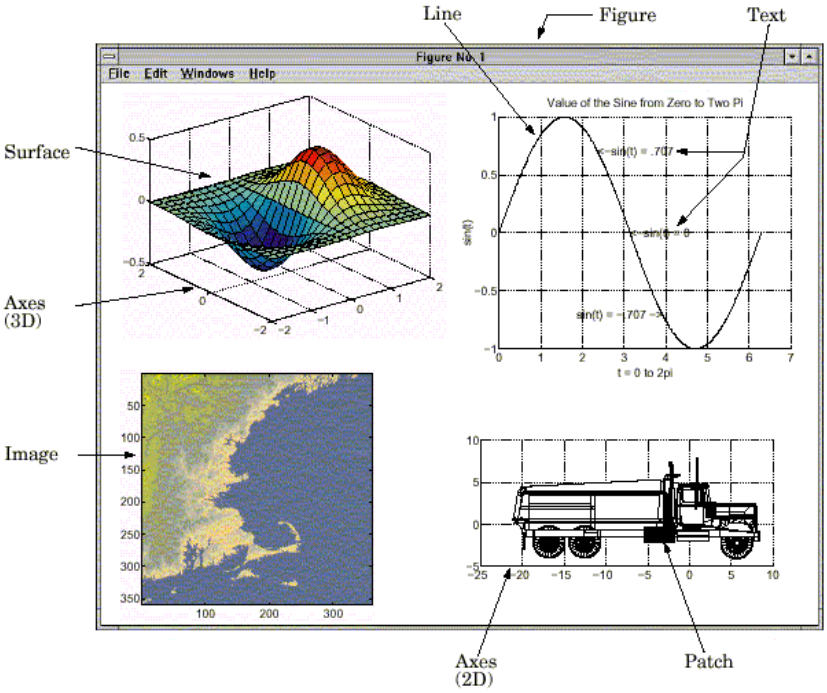
สำหรับในส่วนท้ายของบทนี้เราจะกล่าวถึงคำสั่งที่เกี่ยวข้องกับ Handle ของวัตถุต่างๆ รวมถึงคำสั่งที่เกี่ยวข้องกับการพิจารณาค่าของคุณสมบัติของวัตถุด้วย พร้อมกันนั้นเราจะยกตัวอย่างเพื่อให้เกิดความเข้าใจในเรื่องของ handle ของวัตถุและคุณสมบัติของวัตถุได้ดียิ่งขึ้น

12.1 MATLAB Graphic Object

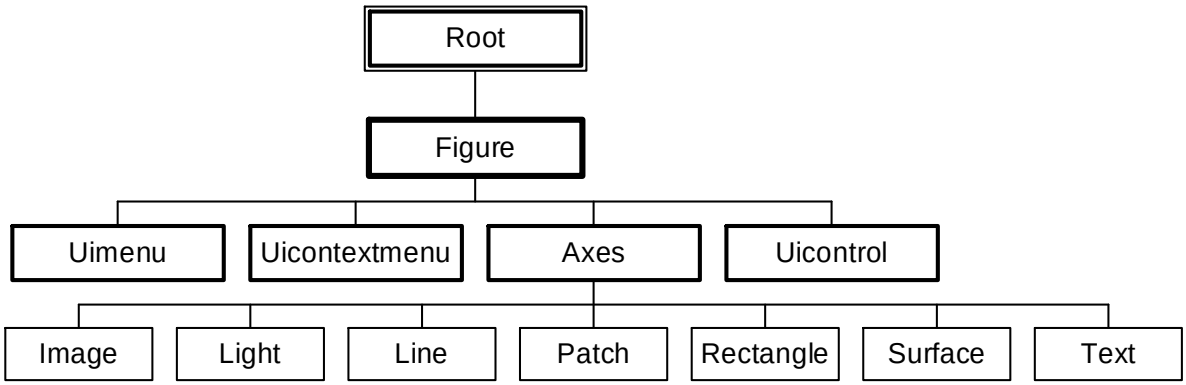
ในการสร้างรูปภาพรูปหนึ่งขึ้นมาโดย MATLAB แม้ว่าที่เราได้ศึกษามาก่อนหน้านี้นั้นจะเป็นการใช้คำสั่งสั้นๆ เพียงคำสั่งเดียว แต่ในความเป็นจริงแล้วจะเกิดการใช้คำสั่งเพื่อสร้างวัตถุขึ้นหลายวัตถุและหลายลำดับขั้นมาก อย่างไรก็ตามก่อนที่จะกล่าวถึงการจัดลำดับขั้นหรือการกำหนด handle ของ object ที่สร้างขึ้นโดย MATLAB ในขั้นแรกนี้เราขอแนะนำให้รู้จักกับวัตถุหรือ object ที่สามารถสร้างขึ้นมาโดย MATLAB ซึ่งมีวัตถุดังต่อไปนี้

Object	**ลักษณะ**
Root object	ความจริงแล้ว Root object นี้ไม่สามารถที่จะสร้างขึ้นได้โดยMATLAB เพราะวัตถุนี้เราหมายถึงจอคอมพิวเตอร์ของเรานั่นเอง ดังนั้นวัตถุนี้จึงถือว่าเป็นรากและจะอยู่ในระดับต่ำสุดซึ่งมีเพียงวัตถุเดียวเสมอ
Figure object	เป็นหน้าต่างที่ MATLAB สร้างขึ้นเพื่อที่จะเขียนสิ่งต่างๆลงไปในหน้าต่างนั้น
Axes object	เป็นพื้นที่สี่เหลี่ยมภายในหน้าต่างของรูปภาพที่สามารกำหนดพิกัดแบบต่างๆ สำหรับกราฟที่จะเขียนลงไป
Image object	รูปภาพต่างๆ ที่แสดงในกราฟ
Light object.	การกำหนดลักษณะของแสง ซึ่งจะมีผลกับ Patches และ Surfaces
Line object	วัตถุที่เป็นเส้นกราฟ
patch object	ส่วนรูปหลายเหลี่ยมที่สามารถนำหลายๆรูป มาต่อกันให้เป็นรูปทรงตามต้องการ
Rectangle	วัตถุซึ่งอาจเป็นได้ตั้งแต่สี่เหลี่ยมมุมฉากปกติ จนกระทั่งถึงวงรี และรูปที่สามารถบรรจุอยู่ในรูปสี่เหลี่ยมมุมฉาก
Surface object	วัตถุที่มีหน้าที่แสดงพื้นผิวในรูปกราฟ
text object	ตัวหนังสือที่เขียนขึ้นในรูปกราฟ
uicontextmenu object	Context menu ซึ่งหมายถึงเมนูที่จะปรากฏขึ้นเมื่อกดเมาส์ปุ่มขวาบริเวณวัตถุที่สร้างเมนูไว้
uicontrol object	เป็นส่วนที่สามารถให้ผู้ใช้ออกคำสั่งหรือโปรแกรมสู่ MATLAB แบบ GUI ซึ่งอาจเป็นปุ่มกด ที่ป้อนค่า เป็นต้น
uimenu object	แถบคำสั่งบน Menu bar ที่ปรากฏอยู่ด้านบนของ Figure window.

สำหรับตัวอย่างของ object ที่เราสามารถที่จะสร้างขึ้นใน MATLAB นั้นเราได้แสดงไว้ในรูปต่อไปนี้



เนื่องจากการวาดวัตถุแต่ละวัตถุจำเป็นจะต้องมีลำดับขั้นในการวาดเหมือนกับที่เรายกตัวอย่างการเขียนรูปลงไปบนกระดาษในตอนต้นแล้วนั้น สำหรับการลำดับขั้นของ Object ที่มีอยู่ใน MATLAB นั้นจะมีลำดับขั้นตามที่แสดงในรูปต่อไปนี้



จากรูปจะเห็นว่า Line จะเป็น Child ของ Axes นั่นคือเราไม่สามารถที่จะสร้าง Line ขึ้นมาได้หากเราไม่สร้าง Axes ขึ้นมาก่อน อย่างไรก็ตามเราคงสังเกตเห็นว่าเมื่อเราใช้คำสั่งขั้นสูงเช่น **plot** เราไม่จำเป็นต้องสร้าง Parent object ขึ้นมาก่อน นั่นคือไม่ต้องออกคำสั่งสร้าง Figure และ Axes ใน Figure นั้นขึ้นมา ทั้งนี้เพราะเหตุว่าคำสั่งชั้นสูงนั้นได้มีการกำหนดให้สร้าง Parent ของเส้นกราฟทั้งหมดขึ้นมาโดยอัตโนมัตินั่นเอง

ในการที่เราสร้างวัตถุขึ้นมาหลายๆอันพร้อมกัน เราจะจำได้อย่างไรว่ามีวัตถุใดบ้างและวัตถุแต่ละแบบจะแตกต่างจากวัตถุอื่นอย่างไร สำหรับ MATLAB ได้จัดการกับเรื่องเหล่านี้โดยมีการกำหนด Handle ของวัตถุแต่ละชิ้นที่ MATLAB สร้างขึ้น คำว่า Handle นี้ความหมายตรงตัวก็คือที่ซึ่งเราจะใช้จับ

วัตถุแต่ละชิ้น เช่นมือจับประตู มือจับกระเป๋าถือนั่นเอง โดยมือจับ (Handle) นี้ในความหมายของ MATLAB ก็คือการกำหนดค่าเป็นตัวเลขให้กับ Object ที่สร้างขึ้น โดย Object แต่ละตัวที่สร้างขึ้นจะมีหมายเลข Handle เป็นของตัวเองโดยเฉพาะ ไม่ซ้ำกับ Object อื่น ดังนั้นหากเราต้องการจะแก้ไขปรับปรุง Object ใด อันดับแรกเราจะต้องจับวัตถุนั้นไว้ก่อน นั่นก็คือต้องเลือกหรือหา Handle ที่ถูกต้องของวัตถุนั้นขึ้นมา

วัตถุต่างๆจะมีรูปร่าง สีหรือขนาดแตกต่างกันออกไป ถึงแม้ว่าจะเป็นวัตถุประเภทเดียวกัน สิ่งที่ทำให้วัตถุแต่ละแบบแตกต่างกันออกไปได้ก็คือ คุณสมบัติ (Property) ของวัตถุนั้น วัตถุแต่ละชิ้นจะมีคุณสมบัติเป็นของตัวเอง ไม่ว่าสี ขนาด รูปร่าง หรืออื่นๆ ดังนั้นหากว่าเราต้องการที่จะเปลี่ยนแปลงสิ่งที่ปรากฏของวัตถุเราจะต้องทำการเปลี่ยนแปลงคุณสมบัติของวัตถุนั้น

12.2 Create Object Function

วัตถุแต่ละแบบยกเว้น Root object จะสามารถสร้างขึ้นด้วยคำสั่ง ซึ่งถือว่าเป็นฟังก์ชันในการที่จะสร้างวัตถุนั้นๆ สำหรับในหัวข้อนี้เราจะกล่าวถึงฟังก์ชันที่ใช้ในการสร้างวัตถุแบบต่างๆ และวิธีการเรียก Handle ของวัตถุเพื่อใช้ในการพิจารณาหรือเพื่อปรับเปลี่ยนรูปแบบของวัตถุนั้นในภายหลังฟังก์ชันที่ใช้ในการสร้างวัตถุใน MATLAB จะมีดังต่อไปนี้

Function	Object Description
figure	คำสั่งที่ใช้สร้าง Figure object
axes	คำสั่งที่ใช้สร้าง Axes object
uicontrol	คำสั่งที่ใช้สร้าง Uicontrol object
uimenu	คำสั่งที่ใช้สร้าง Uimenu object
uicontextmenu	คำสั่งที่ใช้สร้าง Uicontextmenu object
image	คำสั่งที่ใช้สร้าง Image object
light	คำสั่งที่ใช้สร้าง Light object
line	คำสั่งที่ใช้สร้าง Line object
patch	คำสั่งที่ใช้สร้าง Patch object
rectangle	คำสั่งที่ใช้สร้าง Rectangle object
surface	คำสั่งที่ใช้สร้าง Surface object
text	คำสั่งที่ใช้สร้าง Text object

โดยในขั้นต้นนี้ถ้าเราไม่มีการกำหนดค่าคุณสมบัติอื่นใดวัตถุ วัตถุที่สร้างขึ้นใหม่จะใช้ค่าคุณสมบัติของวัตถุนั้นที่ MATLAB ตั้งเตรียมไว้ให้เป็นค่าเริ่มต้น (Default Value) ไว้ก่อนแล้ว หรือในบางกรณีถ้าเราไม่กำหนดค่าคุณสมบัติใดๆให้กับวัตถุนั้นเลยก็อาจจะเกิด Error ขึ้นมาได้เช่นกัน

สำหรับการใช้ฟังก์ชันเหล่านี้ก็จะมีรูปแบบที่คล้ายกัน นั่นคือใช้รูปแบบ

```
h =function('property name', property value,…)
```

ในการใช้คำสั่งเช่นนี้ MATLAB จะสร้างวัตถุ (ตามฟังก์ชันที่กำหนด เช่น line, axes เป็นต้น) โดยให้มีคุณสมบัติ **'property name'** เป็นไปตามค่าคุณสมบัติ **property value** และเมื่อสร้างวัตถุนี้เสร็จแล้ว MATLAB จะให้ Handle ของวัตถุกลับมาและเก็บไว้ในตัวแปร **h** ตามที่ผู้ใช้กำหนด

เราสามารถที่จะกำหนดค่าคุณสมบัติใดๆของวัตถุที่สร้างขึ้นนี้ลงไปได้ ยกเว้นในกรณีที่คุณสมบัตินั้นเป็นแบบ Read only สำหรับการกำหนดค่าคุณสมบัติจะมีลักษณะเป็นคู่ โดย Property Name หมายถึงชื่อของคุณสมบัตินั้นเช่น Tag, Visibility เป็นต้น จะต้องอยู่ภายใต้เครื่องหมาย **'…'** แล้วตามด้วย Property Value ซึ่งก็หมายถึงค่าของคุณสมบัตินั้นซึ่งอาจจะเป็น ตัวเลข ตัวอักษร หรืออื่นๆ แล้วแต่ชนิดของคุณสมบัตินั้นๆ จากตัวอย่างคำสั่งข้างบนนี้เราจะได้ Handle ของวัตถุที่เราสร้างขึ้นมาพร้อมด้วยโดยเรากำหนดให้เก็บ Handle ของวัตถุนั้นไว้ในตัวแปรชื่อ h

ตามที่เราได้กล่าวมาก่อนหน้านี้แล้วเราจะพบว่า Handle ของวัตถุที่สร้างขึ้นนี้จะมีประโยชน์เมื่อเราจำเป็นจะต้องมีการปรับปรุงคุณสมบัติของวัตถุเหล่านั้น เพราะ Handle จะเป็นตัวที่เราสามารถที่จะจับเอาวัตถุเหล่านั้นมาแก้ไขได้

อย่างไรก็ตามจุดประสงค์ที่เราทำความเข้าใจคุณสมบัติต่างๆในที่นี้ ก็เพื่อที่จะนำไปใช้ในการสร้าง GUI ซึ่งในความจริงแล้วเราจะสร้าง GUI ใน environment อีกรูปแบบหนึ่งซึ่งไม่มีความจำเป็นที่จะต้องใช้คำสั่งในการสร้างวัตถุเหล่านี้มากนัก แต่เราจำเป็นจะต้องเรียนรู้วิธีการแก้ไขคุณสมบัติของวัตถุ ดังนั้นในส่วนที่เหลือต่อไปนี้เราจะลองยกตัวอย่างในการแก้ไขคุณสมบัติบางประการของวัตถุ แต่ก่อนอื่นเราต้องเข้าใจรูปแบบและคำสั่งที่เกี่ยวข้องก่อนสักเล็กน้อย ในอันดับแรกเราต้องเข้าใจความหมายของคำสั่งแบบ High-Level และ Low-Level ก่อน

MATLAB มีคำสั่งที่เรียกว่า High-level graphics routines เช่นคำสั่ง **plot** หรือ **surf** เพื่อใช้เรียกมาสร้างวัตถุและวาดวัตถุเหล่านั้นเป็นการรวบรวมคำสั่งในระดับล่างหรือ Low-level graphics routines เข้าเป็นหมวดหมู่ เพื่อให้ผู้ใช้สามารถที่จะใช้งานในคำสั่งต่างๆ ใน High-level graphics routines ได้โดยสะดวก อย่างไรก็ตามการใช้คำสั่งใน High-level routines อาจจะทำให้เกิดการทำงานที่เราควบคุมไม่ได้ หรือเราไม่ต้องการให้เกิดขึ้น ทั้งนี้เพราะเมื่อมีการเขียน high-level graphics routines ผู้เขียนจะคำนึงถึงความสะดวกในการใช้คำสั่งนั้นๆ เป็นสำคัญ โดยหวังว่าผู้ใช้ส่วนใหญ่คงจะมีความสะดวกในการใช้คำสั่งนั้น

อย่างไรก็ตามในหลายๆครั้งเราจะรู้สึกไม่สะดวกกับการใช้ค่าที่ High-level graphics routines กำหนดมาให้ ดังนั้นเราจึงจำเป็นที่จะต้องใช้ Low-level graphics routines เพื่อที่จะสามารถปรับปรุงลักษณะของรูปให้เป็นไปตามที่เราต้องการได้

ในเบื้องต้นนี้เราจะขอยกตัวอย่างที่เกี่ยวกับการเขียนรูปใหม่ลงไปบนหน้าต่างรูปเดิม ซึ่งหากเราใช้ high-level graphics routines เช่นคำสั่ง plot จะทำให้ MATLAB ทำการลบแกนหรือสร้างรูปขึ้นมาใหม่

ทั้งนี้เพราะการเขียนคำสั่ง plot ใน high-level graphics routines นั้นผู้เขียนคาดว่าคงจะสะดวกกว่าถ้าหากว่าทำการวาดแกนใหม่ทุกครั้ง ดังนั้นจึงได้มีการกำหนดคุณสมบัติ NextPlot ของ axes และ figure ให้มีค่าเป็น replace คือแทนรูปเดิม

ในทางตรงกันข้าม การใช้ object creation functions ซึ่งถือว่าเป็น low-level graphics routines จะเป็นการสร้างวัตถุนั้นขึ้นและวางลงไปใน parent ปัจจุบัน โดยที่ไม่ได้สนใจว่าคุณสมบัติ NextPlot นี้จะมีค่าเป็นอย่างไร ยกตัวอย่างเช่นถ้าเราเรียกใช้ฟังก์ชัน line

```
Line('XData',x,'YData',y,'ZData',z,'Color','r')
```

คำสั่งข้างบนนี้จะทำให้ MATLAB สร้างเส้นสีแดงลงใน axes ปัจจุบัน โดยกำหนดข้อมูลตามแกน x หรือกำหนดคุณสมบัติ Xdata ให้มีค่าเท่ากับเวกเตอร์ x และทำนองเดียวกันกับแกน y และ z ตามลำดับ อย่างไรก็ตามหากไม่มี axes เกิดขึ้นก่อนที่จะใช้คำสั่งนี้ MATLAB ก็จะทำการสร้าง axes ขึ้นมาให้ ใหม่ และถ้าหากไม่ figure window ที่จะสร้าง axes ให้ MATLAB ก็จะสร้าง figure window ขึ้นให้ใหม่เช่นกัน ถ้าหากว่าเราเรียกใช้คำสั่งนี้อีกครั้งหนึ่ง MATLAB จะเขียนเส้นอีกเส้นหนึ่งเป็นเส้นที่สองในแกนปัจจุบัน โดยไม่มีการลบเส้นเดิมออกไป

ดังนั้นในขั้นต้นนี้เราจะทราบว่าพฤติกรรมของการใช้คำสั่งในระดับที่ต่างกัน เช่นการที่เราใช้ high-level functions เช่น plot จะให้ผลที่ต่างกัน นั่นคือ plot จะทำการลบ graphics objects และมีการตั้งขนาดแกนและปรับค่าคุณสมบัติต่างๆของ axes ใหม่ทั้งหมด (ยกเว้นคุณสมบัติ Position และ Units) ซึ่งตรงกันข้ามกับการใช้คำสั่ง low-level เช่นคำสั่ง line อย่างไรก็ตามเราสามารถที่จะเปลี่ยนแปลงพฤติกรรมของ high-level functions นี้ได้โดยการใช้คำสั่งในระดับสูงสั่งเพิ่มเติมเข้าไปได้อีก เช่นในกรณีรี้เราจะใช้คำสั่ง hold เพิ่มเติมเข้าไปตามที่เราได้เคยศึกษามาแล้ว เพื่อเปลี่ยนพฤติกรรมของคุณสมบัติ NextPlot และสามารถใช้เฉพาะคำสั่งในระดับสูงโดยไม่ต้องสนใจคำสั่งในระดับต่ำเลยก็ได้ แต่ตามที่เราได้กล่าวไว้ก่อนหน้านี้แล้วว่าการใช้คำสั่งในระดับต่ำจะทำให้เราสามารถควบคุมรูปร่างของวัตถุที่แสดงออกมาได้เป็นไปตามต้องการได้ดีกว่า

สำหรับรายละเอียดของคุณสมบัติ NextPlot ของวัตถุ Figure และ Axes รวมถึงคุณสมบัติอื่นๆ จะกล่าวถึงในบทต่อไป

12.3 Basic Graphic Object Properties

คุณสมบัติของวัตถุที่เป็นกราฟฟิกส์นี้จะควบคุมทั้งสภาพที่ปรากฏและพฤติกรรมของวัตถุนั้น คุณสมบัติจะประกอบด้วยค่าทั่วๆไป เช่นชนิดหรือแบบของวัตถุ เป็น parent ของวัตถุใด มีวัตถุใดบ้างที่เป็น children ผู้ใช้สมารถมองเห็นทางจอภาพได้หรือไม่ ตลอดจนกระทั่งคุณสมบัติที่เป็นค่าเฉพาะสำหรับวัตถุนั้น

วัตถุชนิดหนึ่งๆอาจจะคุณสมบัติไม่เท่ากันขึ้นอยู่กับชนิดของวัตถุนั้น เช่น axes object จะมีคุณสมบัติมากกว่า line object เป็นต้น คุณสมบัติของวัตถุแต่ละแบบ จะเป็นเครื่องกำหนดรูปร่างที่ปรากฏหรือการทำงานของวัตถุนั้น ตัวอย่างเช่นเส้นกราฟที่มีสีต่างกันก็เพราะมีคุณสมบัติทางด้านสีที่กำหนดต่างกันเป็นต้น ดังนั้นหากเราต้องการที่จะเขียน GUI ที่มีประสิทธิภาพ เราจำเป็นที่จะต้องเข้าใจถึงวิธีการเรียกใช้และเปลี่ยนแปลงคุณสมบัติต่างๆ ของวัตถุเหล่านั้น

MATLAB ได้จัดลำดับการเก็บข้อมูลของคุณสมบัติที่อยู่ในวัตถุต่างๆ ยกตัวอย่างเช่น root properties จะบรรจุข้อมูลที่เป็น handle ของรูปปัจจุบันอยู่ รวมถึงตำแหน่งปัจจุบันของ pointer หรือcursor จากนั้น figure properties จะมีรายการของวัตถุที่เป็น children และมีการเก็บปรากฏการณ์ต่างๆที่เกิดขึ้นในหน้าต่างนี้ ทำนองเดียวกัน axes properties ก็จะบรรจุข้อมูลที่เกี่ยวข้องกับ child ของมันทั้งหมดไว้ ด้วยเหตุนี้นั่นเองทำให้เมื่อเราเปลี่ยนคุณสมบัติของ parent สิ่งที่เกิดขึ้นหลังการเปลี่ยนแปลงก็อาจจะเกิดผลกระทบกับ child ของมันได้ แต่ถ้าหากว่าเราเปลี่ยนแปลงคุณสมบัติของ child จะไม่เกิดผลกระทบใดๆ กับ parent เลย

สำหรับในบทนี้เราจะอธิบายถึงคุณสมบัติของวัตถุแบบต่างๆ เพื่อให้เกิดความคุ้นเคย รวมถึงการกำหนดค่าให้กับคุณสมบัติเหล่านั้นด้วย เนื่องจากคุณสมบัติของวัตถุมีอยู่มากมาย ดังนั้นเราจะแบ่งออกเป็นส่วนๆ ดังนี้ ในส่วนแรกเราจะกล่าวถึงรายละเอียดของคุณสมบัติที่เป็นคุณสมบัติที่วัตถุทุกประเภทจะต้องมี ซึ่งทำให้เราสามารถอธิบายคุณสมบัติเหล่านี้ของวัตถุได้ทั้งหมดในครั้งเดียว ส่วนที่สองจะกล่าวถึงคุณสมบัติที่ใช้กันเป็นส่วนใหญ่ของวัตถุประเภทต่างๆ ซึ่งจะมีประโยชน์ในการที่จะนำไปใช้เขียน GUI ต่อไป และสุดท้ายจะเป็นตารางที่แสดงคุณสมบัติของวัตถุทุกแบบ เพื่อให้เราสามารถใช้เป็นที่อ้างอิง ในกรณีที่ต้องการศึกษาวัตถุนั้นๆ ในรายละเอียด

การเปลี่ยนแปลงค่าคุณสมบัติ สามารถกระทำได้โดยการเรียกค่าเหล่านั้นขึ้นมาเปลี่ยนแปลงทีละค่าอย่างไรก็ตามจะมีค่าคุณสมบัติบางค่าที่ไม่สามารถที่จะเปลี่ยนแปลงได้ หรือที่เรียกว่า read only นอกเหนือจากนี้คุณสมบัติหนึ่งๆของวัตถุจะเป็นอิสระต่อกัน แม้ว่าวัตถุชนิดเดียวกันจะมีคุณสมบัติเหมือนกัน การเปลี่ยนคุณสมบัติวัตถุหนึ่งจะไม่ทำให้คุณสมบัติของอีกวัตถุหนึ่งเปลี่ยนไปด้วย เนื่องจากก่อนจะทำการเปลี่ยนแปลงเราจะต้องจับวัตถุชิ้นนั้นๆไว้ก่อน โดยทั่วไปคุณสมบัติต่างๆ จะมีค่าที่ตั้งไว้ให้ก่อนแล้ว (Default Values) ดังนั้นเราไม่จำเป็นต้องกำหนดค่าคุณสมบัติทุกคุณสมบัติก่อนทำการสร้างวัตถุขึ้นมา

คุณสมบัติที่มีสำหรับวัตถุทุกชนิด จะเป็นไปตามตารางต่อไปนี้ โดยตารางนี้จะอธิบายถึงรายละเอียดของคุณสมบัติเหล่านี้ว่าเป็นอย่างไร มีผลอย่างไรต่อวัตถุนั้นๆ

คุณสมบัติ	รายละเอียดของคุณสมบัติ
BusyAction	เป็นการควบคุมวิธีการที่จะให้ MATLAB ปฏิบัติต่อคำสั่งใน callback (คำสั่งที่กระทำหลังจากที่มีการสั่งให้ object นั้นทำงาน) อย่างไร จะให้มีการขัดจังหวะคำสั่งที่กำลังทำอยู่ในขณะนั้นในขณะที่มีคำสั่ง ใหม่เข้ามาได้หรือไม่

คุณสมบัติ	รายละเอียดของคุณสมบัติ
ButtonDownFcn	เป็นคำสั่งที่จะให้ทำหากกดเมาส์ลงในบริเวณวัตถุนั้น
Children	แสดง Handles ของ object ที่เป็น children ของ object นี้ทั้งหมด
Clipping	ยอมให้หรือไม่ยอมให้เกิดการตัดส่วนที่เกินออก จะมีความหมายเฉพาะกับ children ของ axes เท่านั้น
CreateFcn	คำสั่งที่ให้ปฏิบัติเมื่อวัตถุนี้ได้สร้างขึ้น
DeleteFcn	คำสั่งที่ให้ปฏิบัติเมื่อได้รับคำสั่งให้ลบวัตถุนี้ออก
HandleVisibility	กำหนดให้ว่าเราจะสามารถมองเห็น handle ของวัตถุนี้ได้หรือเปล่า นิยมใช้ในกรณีที่ป้องกันผู้ใช้เปลี่ยนแปลงคุณสมบัติของวัตถุ
Interruptible	กำหนดว่าคำสั่งที่เกิดจากวัตถุนี้จะสามารถขัดจังหวะได้หรือไม่หากว่าเครื่องได้รับคำสั่งจากวัตถุอื่น
Parent	Parent ของวัตถุนี้
Selected	แสดงให้เห็นว่าขณะนี้วัตถุนี้ได้รับเลือกหรือไม่
SelectionHighlight	กำหนดลักษณะการแสดงผลให้ผู้ใช้ได้เห็นเมื่อวัตถุกำลังถูกเลือกใช้อยู่
Tag	ชื่อของวัตถุที่กำหนดโดยผู้ใช้
Type	ชนิดของ object เช่น figure, line, text เป็นต้น
UserData	กำหนดว่ามีข้อมูลใดบ้างที่ต้องการใช้ เมื่อเรียกวัตถุนี้
Visible	กำหนดว่าผู้ใช้จะสามารถมองเห็นวัตถุนี้ได้หรือไม่

สำหรับชื่อของคุณสมบัติต่างๆ มีข้อสังเกตที่ควรกล่าวถึงในที่นี้บางประการคือ โดยปกติแล้วในการกล่าวถึงคุณสมบัติต่างๆ MATLAB จะเขียนตัวพิมพ์ใหญ่ที่อักษรตัวแรกของคำที่นำมาประกอบเป็นชื่อคุณสมบัติ เช่น LineStyle หรือ XMinorTickMode การที่ทำเช่นนี้ก็เพื่อที่จะให้การอ่านชื่อคุณสมบัติง่ายขึ้นและเราเข้าใจถึงความหมายของคุณสมบัตินั้นๆ ได้ง่ายขึ้น อย่างไรก็ตาม MATLAB ไม่ได้มีการตรวจสอบอักษรตัวพิมพ์ใหญ่หรือตัวพิมพ์เล็กสำหรับการใช้สะกดชื่อคุณสมบัติ นอกเหนือจากนั้นในความเป็นจริงแล้วเราอาจใช้ตัวอักษรเพียง 2 ถึง 3 ตัวแรกที่สามารถแยกคุณสมบัติที่เราต้องการออกจากคุณสมบัติอื่น ก็สามารถที่จะบ่งถึงคุณสมบัตินั้นได้โดยไม่จำเป็นต้องพิมพ์ชื่อคุณสมบัติเหล่านั้นให้เต็มทั้งหมดก็ได้ อย่างไรก็ตามเราแนะนำว่าในการเขียน M-files เราแนะนำให้ใช้การสะกดชื่อให้เต็มเพราะอาจเกิดปัญหาอื่นตามมาเช่นหากว่ามีการนำไปใช้กับ MATLAB ใน version ที่ออกมาใหม่แล้วมีการเพิ่มคุณสมบัติเข้าไป อักษร 2 – 3 ตัวแรกที่เราใช้อาจไม่สามารถที่จะแยกคุณสมบัติตัวเดิมกับคุณสมบัติที่สร้างขึ้นมาใหม่ได้

12.4 GET and SET Function

คำสั่งที่สำคัญคำสั่งหนึ่งในการเรียกดูค่าคุณสมบัติและการปรับเปลี่ยนค่าคุณสมบัติก็คือคำสั่ง GET และ SET โดยคำสั่งทั้งสองนี้มีเป็นคำสั่งที่มีประโยชน์มากในการเรียกใช้หรือเปลี่ยนค่าคุณสมบัติ รายละเอียดของคำสั่งทั้งสองมีดังนี้

คำสั่ง GET

เป็นคำสั่งที่ใช้เรียกดูค่าคุณสมบัติต่างๆ ของ Graphic object โดยรูปแบบของคำสั่งมีดังนี้

```
get(object_handle)
```

หรือ

```
returned_value = get(object_handle,'PropertyName');
```

เมื่อ **object_handle** จะหมายถึงค่า handle ของวัตถุที่เราพิจารณา ส่วน **'PropertyName'** จะหมายถึงชื่อของคุณสมบัตินี้ ในการที่เราไม่กำหนดชื่อคุณสมบัติ ตามที่แสดงในตัวอย่างแรก จะทำให้เราได้คำตอบออกมาเป็นค่าคุณสมบัติทั้งหมดของวัตถุนั้น ตัวอย่างต่อไปนี้จะเป็นการแสดงการใช้คำสั่งนี้

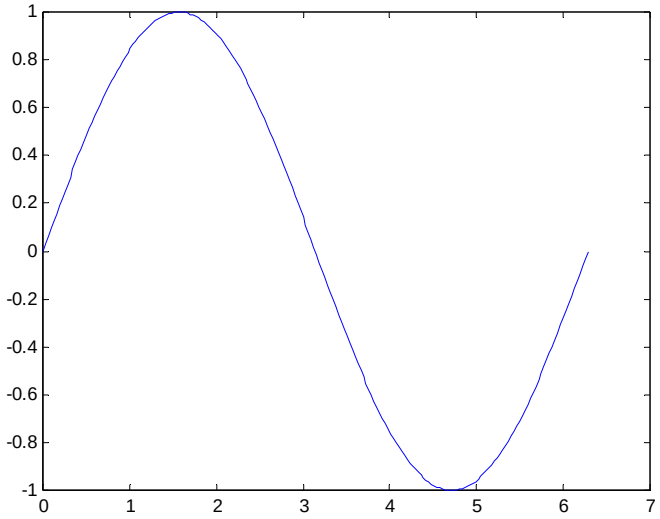
✍ ตัวอย่าง

ตัวอย่างต่อไปนี้มีจุดประสงค์เพื่อให้เราได้เข้าใจถึงวิธีการดูค่าคุณสมบัติของวัตถุ และปรับปรุงค่าคุณสมบัติของวัตถุ ซึ่งคำสั่งหลักที่เราจะใช้ในตัวอย่างนี้คือ set และ get สำหรับการทำตัวอย่างนี้เราหวังว่าคุณกำลังเปิดเครื่องคอมพิวเตอร์และทำตัวอย่างนี้ไปพร้อมๆ กับเรา

ขั้นแรกเราจะทำการสร้างกราฟขึ้นมาเส้นหนึ่งโดยใช้คำสั่งชั้นสูง ในที่นี้เราจะสร้างกราฟ sine ขึ้นมาหนึ่งคาบ โดยใช้คำสั่ง

```
» x=linspace(0,2*pi);
» y=sin(x);
» line_handle = plot(x,y);
```

ซึ่งเราจะได้รูปดังต่อไปนี้



ซึ่ง line_handle จะเป็น handle ของเส้นกราฟที่สร้างขึ้น อย่างไรก็ตามจะมี object อีก 2 object ที่สร้างขึ้นโดยอัตโนมัติ ภายใต้การทำงานของ MATLAB ดังนั้นหากเราต้องการทราบค่า handle ของรูปและแกน เราจำเป็นที่จะต้องใช้คำสั่งเพิ่มเติม สำหรับคำสั่งที่ใช้เพิ่มเติมในที่นี้คือ

gcf (**Get Current Figure**) เป็นคำสั่งที่ให้ MATLAB บอกค่า handle ของรูปปัจจุบัน จะกล่าวถึงในรายละเอียดในช่วงท้ายของบทนี้

gca (**Get Current Axes**) เป็นคำสั่งที่ให้ MATLAB บอกค่า handle ของแกนปัจจุบัน จะกล่าวถึงในรายละเอียดในช่วงท้ายของบทนี้

สำหรับความหมายของคำว่ารูปปัจจุบัน หรือแกนปัจจุบันคือรูปหรือแกนที่ได้มีการทำกิจกรรมใดๆ ขึ้นกับรูปหรือแกนนั้นเป็นรูปหรือแกนสุดท้าย และจะเป็นรูปหรือแกนซึ่งการทำกิจกรรมใหม่จะเกิดขึ้นใหม่ สำหรับในกรณีที่เรามีรูปและแกนเพียงหนึ่งอัน ก็คงไม่ยากที่จะชี้ไปที่รูปหรือแกนเหล่านั้น อย่างไรก็ตามในกรณีที่เราคิดว่าจะเปิดหน้าต่างรูปหลายหน้าต่างพร้อมกัน หรือในกรณีที่เราสร้างแกนขึ้นมาเขียนรูปพร้อมกัน การหา handle ของรูปหรือแกนที่เราต้องการอาจจะต้องเพิ่มขั้นตอนเข้าไปบ้าง ซึ่งเราจะกล่าวถึงในภายหลัง

ในตัวอย่างนี้เราสามารถได้ handle ของรูปและแกนจากคำสั่ง

```
» fig_handle =gcf

fig_handle =
     1
```

และ

```
» axes_handle =gca

axes_handle =
   73.0007
```

นั่นหมายความว่า MATLAB กำหนดให้ handle ของรูปนี้มีค่าเท่ากับ 1 และแกนที่เขียนเส้นนี้มี handle เท่ากับ 73.007 นั่นเอง

ขั้นต่อไปหลังจากที่เราได้ handle ของวัตถุทั้งสาม (Figure, Axes, Line) ที่เราได้สร้างขึ้นมาแล้ว หากเราต้องการจะดูว่าวัตถุที่เราสนใจนั้นมีคุณสมบัติอะไรบ้าง เราสามารถทำได้โดยใช้คำสั่ง get ยกตัวอย่างเช่นหากเราต้องการที่จะดูคุณสมบัติของ Figure ทั้งหมดที่ MATLAB ได้สร้างขึ้น เราสามารถใช้คำสั่ง

```
» get(fig_handle)
```

```
    BackingStore = on
    CloseRequestFcn = closereq
    Color = [0.8 0.8 0.8]
    Colormap = [(64 by 3) double array]
    CurrentAxes = [73.0007]
    CurrentCharacter =
    …
    …
    …
    WindowButtonUpFcn =
    WindowStyle = normal

    ButtonDownFcn =
    Children = [73.0007]
    …
    …
    …
    Tag =
    Type = figure
    UIContextMenu = []
    UserData = []
    Visible = on
```

สำหรับรายละเอียดของคุณสมบัติต่างๆ และลักษณะของค่าคุณสมบัติแต่ละชนิดว่าเป็นตัวเลข เมตริกซ์ หรืออื่นๆ เราจะกล่าวถึงในภายหลัง หรือในกรณีที่เราต้องการดูคุณสมบัติของเส้นที่สร้างขึ้น เราสามารถใช้คำสั่ง

```
» get(line_handle)
```

```
    Color = [0 0 1]
    EraseMode = normal
    LineStyle = -
    LineWidth = [0.5]
    Marker = none
    MarkerSize = [6]
    MarkerEdgeColor = auto
    MarkerFaceColor = none
    XData = [ (1 by 200) double array]
    YData = [ (1 by 200) double array]
    ZData = []
    ButtonDownFcn =
    Children = []
    Clipping = on
```

```
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = on
HitTest = on
Interruptible = on
Parent = [73.0007]
Selected = off
SelectionHighlight = on
Tag =
Type = line
UIContextMenu = []
UserData = []
Visible = on
```

ในกรณีที่เราต้องการจะเจาะจงทราบค่าคุณสมบัติตัวใดตัวหนึ่งโดยเฉพาะ ไม่ว่าเพื่อนำค่านี้ไปใช้ต่อไปหรือเพื่อทราบค่าปกติเราสามารถที่จะใช้คำสั่ง get แบบเจาะจงคุณสมบัติได้ เช่นหากเราต้องการที่จะทราบถึงสีของเส้นว่ามีสีใด เราสามารถใช้คำสั่ง

```
»l_color=get(line_handle,'Color')

l_color =
     0     0     1
```

ซึ่งคำสั่งนี้หมายความว่าให้เก็บค่าคุณสมบัติที่แสดงสีของเส้นลงในตัวแปรที่ชื่อ l_color และสีของเส้นนี้มีค่าเท่ากับ [0 0 1] ซึ่งหมายถึงสีน้ำเงิน ในระบบสีแบบ RGB ตามที่เราได้กล่าวถึงมาก่อนหน้านี้แล้ว

❶ Querying Individual Properties

การเลือกดูค่าคุณสมบัติทีละค่าอาจจะให้ค่าออกมาเป็นลักษณะของ structure ได้ เช่นเราใช้คำสั่ง

```
» get(gca,'ColorOrder')
ans =
                0          0    1.0000
                0    0.5000          0
    1.0000              0          0
         0          0.7500    0.7500
    0.7500          0    0.7500
    0.7500    0.7500          0
    0.2500    0.2500    0.2500
```

ซึ่งจะมีลักษณะของตัวแปรเป็นแบบ Structure

ถ้าเรากำหนดค่า output ของฟังก์ชัน get ให้กับตัวแปรหนึ่ง MATLAB จะสร้างตัวแปรนั้นให้เป็น structure array โดยมีชื่อ field แต่ละ field เป็นชื่อของคุณสมบัติต่างๆ และค่าของ field นั้นๆจะเป็นค่าของคุณสมบัติในขณะนั้นๆ ยกตัวอย่างเช่น

```
a = get(line_handle);
```

เราจะได้ตัวแปร a เป็น structure array และเราจะสามารถทราบค่าของตัวแปรในแต่ละ field นั้นด้วยการใช้ชื่อ filed ของแต่ละคุณสมบัติ เช่น

```
» a.Color
ans =
   0   0   1
```

❷ Querying Groups of Properties

เราสามารถที่จะใช้ cell array ของชื่อคุณสมบัติเพื่อใช้ดึงค่าคุณสมบัติหลายคุณสมบัติพร้อมกัน เพื่อความสะดวกในการใช้ค่าในกลุ่มนั้นๆ ยกตัวอย่างเช่นถ้าหากเราต้องการที่จะดึงค่าที่เกี่ยวข้องกับคุณสมบัติ “camera mode” ขั้นแรกเรากำหนด cell array ดังนี้

```
» camera_props(1)={'CameraPositionMode'};
» camera_props(2)={'CameraTargetMode'};
» camera_props(3)={'CameraUpVectorMode'};
» camera_props(4)={'CameraViewAngleMode'};
```

จากนั้นใช้ค่า cell array เป็นค่าที่ใช้เรียกดูค่าคุณสมบัติเหล่านั้น ดังนี้

```
» get(gca,camera_props)
ans =
'auto' 'auto' 'auto' 'auto'
```

✓ การใช้คำสั่ง set

ฟังก์ชัน set เป็นฟังก์ชันที่ใช้กำหนดค่าคุณสมบัติค่าใหม่ให้กับวัตถุที่เราต้องการ ซึ่งจะช่วยให้เราสามารถที่จะปรับเปลี่ยนคุณสมบัติต่างๆ ได้ รูปแบบการใช้คำสั่ง set จะมีลักษณะดังนี้

```
set(object_handle,'PropertyName','NewPropertyValue')
```

โดยที่

object_handle	คือ handle ของวัตถุที่เราต้องการเปลี่ยนคุณสมบัติ
PropertyName	คือคุณสมบัติที่เราต้องการเปลี่ยน
NewPropertyValue	คือค่าของคุณสมบัติใหม่ที่เราต้องการ โดยค่าคุณสมบัติที่กำหนดจะต้องเป็นค่าที่สอดคล้องกับคุณสมบัตินั้นๆ เช่นหากคุณสมบัตินั้นต้องการตัวอักษร ค่า **NewPropertyValue** นี้จะต้องเป็นตัวอักษรด้วย ไม่เช่นนั้นจะเกิดความผิดพลาดขึ้น สำหรับรายละเอียดของคุณสมบัติต่างๆ จะกล่าวถึงในภายหลัง

พิจารณากราฟที่เราสร้างขึ้นนี้หากว่าเราต้องการที่จะเปลี่ยนสีของเส้นกราฟเป็นสีแดง เราสามารถที่จะทำได้ โดยการกำหนดค่าคุณสมบัติสีของเส้นกราฟนี้ใหม่ สำหรับในระบบ RGB สีแดงจะมีค่าเป็น [1 0 0] ซึ่งการใช้คำสั่งจะเป็นดังนี้

```
» set(line_handle,'Color',[1 0 0])
```

เราจะพบว่าเส้นกราฟของเราได้เปลี่ยนเป็นสีแดง อย่าลืมว่าค่าคุณสมบัตินั้นเราจะต้องเขียนอยู่ภายใต้เครื่องหมาย quote เสมอเนื่องจากเป็นค่าตัวแปรแบบ string และในกรณีที่เราต้องการที่จะทราบว่าคุณสมบัติหนึ่งๆ ต้องการค่าคุณสมบัติแบบใด เราสามารถที่จะทำได้โดยการเปิดคู่มือของ MATLAB หรือเราสามารถที่จะทำได้โดยการใช้คำสั่ง set เพื่อให้แสดงค่าที่เหมาะสมกับคุณสมบัตินั้น วิธีการก็คือการใช้คำสั่ง

```
set(object_handle,'PropertyName')
```

เมื่อเราไม่ใส่ค่าคุณสมบัติใหม่ลงไป MATLAB จะแสดงค่าคุณสมบัติที่เป็นไปได้สำหรับคุณสมบัตินั้น ในกรณีที่คุณสมบัตินั้นมีค่าที่สามารถกำหนดได้เฉพาะกลุ่ม คือมีค่าเพียงกลุ่มเดียวไม่ใช่ค่าที่สามารถที่จะใส่ได้อย่างทั่วไป ยกตัวอย่างเช่นเราต้องการรู้ว่าลักษณะของเส้นที่เป็นไปได้ที่ MATLAB สามารถแสดงได้นั้นมีเส้นแบบใดบ้างเราสามารถที่จะใช้คำสั่ง

```
» set(line_handle,'LineStyle')
[ {-} | -- | : | -. | none ]
```

ความหมายก็คือค่าที่เป็นไปได้จะมี [{-} | -- | : | -. | none] ซึ่งค่าที่อยู่ในวงเล็บปีกกา { } คือค่าที่เป็นค่าในปัจจุบัน สำหรับในกรณีนี้ค่าปัจจุบันคือ – ซึ่งหมายถึงเส้นต่อเนื่อง ซึ่งในบทที่ผ่านมาเราได้กล่าวถึงลักษณะของเส้นไปแล้ว อีกตัวอย่างหนึ่งก็เช่นหากเราต้องการทราบค่าคุณสมบัติที่เป็นไปได้ของความหนาของเส้นกราฟเราสามารถที่จะใช้คำสั่ง

```
» set(line_handle,'LineWidth')
A line's "LineWidth" property does not have a fixed set of property values.
```

ซึ่งเราจะไม่ได้ค่าคุณสมบัติของความหนา เพราะความหนาของเส้นกราฟสามารถที่จะกำหนดได้หลากหลาย ไม่ได้เป็นกลุ่มเฉพาะกลุ่มเดียว

นอกจากนี้หากเราต้องการที่จะดูคุณสมบัติทั้งหมดที่มีค่าเป็นกลุ่มเฉพาะเท่านั้นว่ามีคุณสมบัติใดบ้างที่มีลักษระเช่นนั้น เราสามารถที่จะใช้คำสั่ง

```
Set(object_handle)
```

ยกตัวอย่างเช่น

```
» set(axes_handle)
AmbientLightColor
Box: [ on | {off} ]
```

```
CameraPosition
CameraPositionMode: [ {auto} | manual ]
CameraTarget
CameraTargetMode: [ {auto} | manual ]
CameraUpVector
CameraUpVectorMode: [ {auto} | manual ]
…
…
Tag
UIContextMenu
UserData
Visible: [ {on} | off ]
```

สำหรับคุณสมบัติที่ไม่มีค่าตามหมายความถือว่าคุณสมบัตินั้นไม่ได้มีค่าเป็น set เฉพาะนั่นเอง

เราสามารถที่จะกำหนดคุณสมบัติและค่าของคุณสมบัติโดยใช้ structure arrays หรือ cell arrays กรณีเช่นนี้จะมีประโยชน์ในการที่เราต้องการตั้งค่าคุณสมบัติเดียวกันนี้กับวัตถุหลายๆวัตถุให้มีค่าเหมือนกัน ยกตัวอย่างเช่นหากเราต้องการที่จะตั้งคุณสมบัติทางด้านการมองแกนสำหรับระบบแกนของกราฟรูปแบบหนึ่งเราสามารถที่จะใช้คำสั่งลักษณะดังต่อไปนี้

```
»view1.CameraViewAngleMode = 'manual';
»view1.DataAspectRatio = [1 1 1];
»view1.ProjectionType = 'Perspective';
```

และถ้าหากต้องการที่จะตั้งค่าเหล่านี้ลงบนแกนปัจจุบันเราสามารถที่จะใช้คำสั่ง

```
»set(gca,view1)
```

ถ้าหากว่าเรากำหนดให้มี output ของฟังก์ชัน set เป็นตัวแปรหนึ่ง MATLAB จะให้ค่ากลับมาเป็น structure array ตัวอย่างเช่น

```
» a =set(gca);
```

ซึ่งเราจะได้ชื่อ field ของค่า a เป็นชื่อของคุณสมบัติ และค่าของ field จะเป็นค่าที่เป็นไปได้สำหรับคุณสมบัตินั้นๆ ยกตัวอย่างเช่น

```
» a.GridLineStyle

ans =
    '-'
    '--'
    ':'
    '-.'
    'none'
```

ซึ่งเราจะได้ค่าที่เป็นไปได้ของคุณสมบัติ grid line styles ของระบบแกนนี้

อย่าลืมว่าแม้ชื่อของคุณสมบัตินั้นไม่เป็น case sensitive แต่อย่างไรก็ตามในกรณีที่เป็นชื่อของ structure field จะเป็น case sensitive ดังนั้นถ้าเราใช้คำสั่ง

```
»a.gridlinestyle
??? Reference to non-existent field 'gridlinestyle'.
```

เราจะได้ error เพราะเราต้องกำหนดรูปแบบตัวอักษรให้ถูกต้อง เนื่องจากตัวแปรใน MATLAB เป็น case sensitive

โดยทั่วไปแล้ว MATLAB จะตั้งค่าเริ่มต้นมาให้กับคุณสมบัติทุกคุณสมบัติ ซึ่งจะเป็นค่าที่ MATLAB ใช้ถ้าหากว่าผู้ใช้ไม่มีการกำหนดค่าคุณสมบัติอื่นลงไปอีก เราสามารถที่จะดูค่าที่ MATLAB ตั้งไว้ได้โดยการใช้คำสั่ง get เพื่อที่จะดูค่าคุณสมบัติทั้งหมดที่ตั้งไว้หรือที่นิยมเรียกว่า factory-defined ได้ตามคำสั่งนี้

```
» a = get(0,'Factory');
```

ในที่นี้ 0 หมายถึง handle ของวัตถุ root ซึ่งหมายถึงจอคอมพิวเตอร์นั่นเอง ค่าที่ได้จะเป็น structure array ซึ่งมี field เป็นชื่อของชนิดของวัตถุต่อด้วยชื่อของคุณสมบัตินั้น และค่าของ field จะเป็นค่าที่ตั้งหรือ factory value ของแต่ละวัตถุนั้นๆ ยกตัวอย่างเช่น field

```
UimenuSelectionHighlight: 'on'
```

หมายความว่าค่าเริ่มต้นของคุณสมบัติ SelectionHighlight ของวัตถุ uimenu จะมีค่าเป็น on และถ้าหากเราต้องการที่จะได้ค่าเริ่มต้นของคุณสมบัติหนึ่งเป็นการเฉพาะเราอาจใช้คำสั่ง

```
Get(0,'FactoryObjectTypePropertyName')
```

ยกตัวอย่างเช่น ถ้าเราสั่ง

```
» get(0,'FactoryTextFontName')
```

สำหรับระบบปฏิบัติการ Windows มาตราฐานเราจะได้คำตอบเป็น

```
ans =
Helvetica
```

นั่นคือจะมีชนิดของตัวหนังสือที่ใช้เป็นแบบ Helvetica

12.5 Related Object Function

คำสั่งต่อไปนี้เป็นคำสั่งที่เกี่ยวข้องกับการค้นหาหรือสร้างวัตถุ เราได้พยายามรวบรวมเฉพาะฟังก์ชันที่สำคัญและคาดว่าจะนำไปใช้งานในการสร้าง GUI ในบทต่อไป

✪ findobj

จุดมุ่งหมาย ใช้ในการค้นหาวัตถุ

รูปแบบ `h = findobj`

```
h = findobj('PropertyName',PropertyValue,…)
h = findobj(objhandles,…)
h = findobj(objhandles,'flat','PropertyName',PropertyValue,...)
```

รายละเอียด

คำสั่ง **findobj** จะค้นหาวัตถุที่เราต้องการและจะให้ค่า handles ของวัตถุนั้น เราสามารถที่จะค้นหาวัตถุที่เราต้องการได้โดยการให้ค่าคุณสมบัติที่เจาะจงของวัตถุนั้นๆ พร้อมกับการกำหนดลำดับขั้นของวัตถุนั้นด้วย สำหรับคำสั่ง

```
h = findobj
```

จะให้ handles ของ root object และวัตถุที่มีลำดับขั้นที่ต่ำกว่าทั้งหมด เช่น

```
» findobj
ans =
          0
    1.0000
   73.0007
    1.0010
```

ซึ่งจะเป็น handle ของ root, figure, axes และ line ที่ประกอบกันขึ้นมาเป็นกราฟตามลำดับ สำหรับคำสั่ง

```
h = findobj('PropertyName', PropertyValue,...)
```

จะให้ handle ของวัตถุทั้งหมดที่มีชื่อคุณสมบัติเป็น PropertyName และมีค่าของคุณสมบัตินั้นเท่ากับค่า PropertyValue เราสามารถที่จะจำกัดขอบเขตของการค้นหาวัตถุได้โดยการกำหนดคุณสมบัติและค่าคุณสมบัติที่ต้องการหลายๆค่าพร้อมกัน ซึ่งในกรณีนี้ **findobj** จะให้เฉพาะ handle ที่มีค่าคุณสมบัติเป็นไปตามที่กำหนดเท่านั้น ส่วนคำสั่ง

```
h = findobj(objhandles,…)
```

จะบังคับให้การค้นหานั้นอยู่ในวัตถุที่กำหนด handles และวัตถุที่มีลำดับต่ำกว่าเท่านั้น คำสั่ง

```
h = findobj(objhandles,'flat','PropertyName',PropertyValue,...)
```

เป็นการบังคับให้มองหาเฉพาะวัตถุที่มีค่าคุณสมบัติตามที่กำหนดโดยไม่มีการมองหาวัตถุที่อยู่ขั้นต่ำกว่า

หมายเหตุ

คำสั่ง **findobj** จะให้ error ถ้าหากว่าไม่มีวัตถุตาม handle ที่เรากำหนด นอกเหนือจากนั้นการกำหนดค่าคุณสมบัติจะใช้ค่าที่ถูกต้องในลักษณะใดลักษณะหนึ่งก็ได้เช่น

```
»findobj('Color','r')
```

จะเป็นการหาวัตถุที่มีคุณสมบัติ Color property เท่ากับ red หรือ r หรือ [1 0 0] เพราะทุกค่าคุณสมบัติจะหมายถึงสีแดงทั้งหมด นอกจากนั้นเมื่อวัตถุที่กำหนดมีวัตถุที่อยู่ต่ำกว่ามากกว่าหนึ่งวัตถุ MATLAB จะมองไปที่วัตถุนั้นๆ ในแต่ละครั้งที่คำสั่ง **findobj** พบวัตถุที่อยู่ต่ำกว่ามัน ดังนั้นอาจเป็นไปได้ที่จะมีการให้ค่า handle ของวัตถุหนึ่งๆ มากกว่าหนึ่งครั้ง

✪ copyobj

จุดมุ่งหมาย คัดลอกวัตถุและวัตถุที่อยู่ในขั้นที่ต่ำกว่า

รูปแบบ **new_handle = copyobj(h, p)**

คำอธิบาย

copyobj จะทำการคัดลอกวัตถุเดิมและสร้างวัตถุที่เหมือนกันขึ้นมาใหม่ ยกเว้นจะมีการกำหนดค่าของคุณสมบัติ Parent ให้แตกต่างไป และมีการกำหนด handle ขึ้นใหม่ โดย parent ใหม่นี้ต้องเป็นค่าที่เหมาะสมในการที่จะทำการคัดลอก เช่นการจะทำการคัดลอกวัตถุ line จะต้องเป็นการคัดลอกลงบนวัตถุ axes อื่นเท่านั้น คำสั่ง

```
new_handle =copyobj(h, p)
```

เป็นการคัดลอกวัตถุที่กำหนด handle ด้วย h และให้ค่า handle ของวัตถุที่คัดลอกขึ้นมาใหม่ และวัตถุที่สร้างขึ้นมาใหม่นี้จะเป็น children ของวัตถุที่มี handle ที่กำหนดด้วย p โดย h และ p สามารถเป็นได้ทั้งสเกลลาและเวกเตอร์ แต่ถ้าทั้งคู่เป็นเวกเตอร์ค่าทั้งสองจะต้องมีขนาดเท่ากันและค่าnew_handle ที่ได้จะเป็นเวกเตอร์ที่มีขนาดเท่ากัน ในกรณีนี้ค่า new_handle(i) จะเป็นการคัดลอก h(i)] ลงใน parent ที่กำหนดตาม p(i) ในกรณีที่ h เป็นสเกลลาและ p เป็นเวกเตอร์ จะทำการคัดลอกวัตถุ h แต่ละครั้งลงในparents ตาม p แต่ละค่า ค่าที่ได้ new_handle(i) จะเป็นสำเนาของ h ที่มี parent ตาม p(i) และค่าของ length(new_handle) จะเท่ากับ length(p).

เมื่อ h เป็นเวกเตอร์และ p เป็นสเกลลา ค่า new_handle(i) แต่ละค่าจะได้จากสำเนาของ h(i) ที่มี Parent เป็น p ความยาวของ new_handle จะเท่ากับ length(h) การสร้างวัตถุขึ้นมาใหม่จะสร้างวัตถุที่มีลำดับต่ำกว่าวัตถุนั้นลงไปด้วย

ตัวอย่างเช่นเราต้องการทำการคัดลอกวัตถุที่เป็น surface ลงในแกนใหม่ที่อยู่ใน figure อื่น เรามีขั้นตอนดังนี้ ขั้นแรกสร้าง surface หลักขึ้นก่อน

```
h =surf(peaks);
colormap hot
```

จากนั้นสร้างรูปและแกนขึ้นใหม่เพื่อจะได้คัดลอกกราฟเดิมลงไป

```
figure % สร้าง figure windows ขึ้นใหม่
axes % สร้างแกนขึ้นใหม่บน figure ที่สร้างขึ้น
```

ขั้นสุดท้ายเป็นการคัดลอกกราฟจากรูปแรกลงในรูปที่เราสร้างขึ้น

```
new_handle = copyobj(h,gca);
colormap hot
view(3)
grid on
```

ในขั้นตอนการคัดลอกเราจะเห็นว่ามีบางส่วนที่จะทำการคัดลอกไปกับวัตถุ เช่นรูปแบบหรือขนาดของ surface แต่ colormap ซึ่งเป็นคุฯสมบัติของ figure ตลอดจน view และ grid ซึ่งเป็นคุณสมบัติของ axes จะไม่ได้รับการคัดลอกไปกับผิว surface ด้วย

✪ gca

จุดมุ่งหมาย ดึงค่า handle ของ axes ปัจจุบัน (gca = Get Current Axes)

รูปแบบ `h = gca`

คำอธิบาย

คำสั่ง `h = gca` จะให้ handle ของ axes ปัจจุบันของ figure ปัจจุบัน ถ้าหากว่าขณะนั้นไม่มีแกนใดได้รับการสร้างขึ้นมาก่อนเลย MATLAB จะสร้างแกนขึ้นมาให้ใหม่และให้ handle ของแกนนั้น เราสามารถที่จะใช้คำสั่ง

```
get(gcf,'CurrentAxes')
```

ถ้าเราไม่ต้องการให้ MATLAB สร้างแกนขึ้นมาใหม่ ถ้าหากไม่มีแกนใดถูกสร้างขึ้นมาก่อน

แกนที่เป็น current axes หรือแกนที่เป็นแกนปัจจุบัน หมายถึงแกนที่ MATLAB จะทำการสร้างรูปภาพต่างๆลงไปบนแกนนั้นหากว่ามีการสั่งให้สร้างรูปภาพขึ้น คำสั่งสร้างรูปเช่น plot, text, หรือ surf จะเขียนผลของมันลงใน current axes การเปลี่ยน current figure หรือรูปปัจจุบันจะทำให้ current axes เปลี่ยนไปด้วย

✪ gcbo

จุดมุ่งหมาย เป็นคำสั่งให้แสดง handle ของวัตถุที่ callback ของมันกำลังทำงานอยู่

รูปแบบ `h = gcbo`

`[h, figure] = gcbo`

คำอธิบาย

คำสั่ง `h = gcbo` จะให้ handle ของวัตถุที่ callback ของมันกำลังทำงานอยู่ ส่วนคำสั่ง [h, figure] = gcbo จะให้ handle ของวัตถุที่ callback ของมันกำลังทำงานอยู่ และให้ handle ของ figure ที่บรรจุวัตถุนั้นอยู่ด้วย

หมายเหตุ

MATLAB จะเก็บ handle ของวัตถุที่ callback กำลังทำงานอยู่ไว้ในวัตถุ root ที่คุณสมบัติ CallbackObject ถ้าหากว่า callback หนึ่งได้รับการขัดจังหวะด้วย callback อีกคำสั่งหนึ่งMATLAB จะแทนค่า

คุณสมบัติ CallbackObject ด้วย handle ของวัตถุที่ callback นั้นเข้าไปขัดจังหวะ จนกระทั่ง callback ที่เข้าไปใหม่นี้ทำงานเสร็จสิ้น MATLAB จึงจะนำค่า handle ของวัตถุที่ถูกขัดจังหวะกลับมาแทนที่

ค่าคุณสมบัติ CallbackObject ของวัตถุ root จะเป็นคุณสมบัติแบบ read-only ดังนั้นค่านี้จะใช้ได้ตลอดเวลาที่ callback กำลังทำงานอยู่

สำหรับค่าคุณสมบัติ CurrentFigure ของวัตถุ root และค่าคุณสมบัติ CurrentAxes และ CurrentObject เป็นค่าที่ผู้ใช้สามารถกำหนดได้ ดังนั้นจึงสามารถที่จะมีการเปลี่ยนแปลงได้ในระหว่างที่ callback ทำงานอยู่ โดยเฉพาะในกรณีที่ callback หนึ่งถูกขัดจังหวะด้วย callback ของอีกวัตถุหนึ่ง ดังนั้นฟังก์ชันเหล่านี้จึงอาจให้ค่าที่ไม่แน่นอนว่า callback ที่กำลังทำงานอยู่นั้นเป็นของวัตถุใด

เมื่อเราเขียน callback สำหรับการใช้ CreateFcn และ DeleteFcn ของวัตถุใดๆ และการใช้ ResizeFcn ของ figure เราจะต้องใช้ **gcbo** เนื่องจาก callbacks เหล่านั้นจะไม่มีการเปลี่ยนแปลงค่าคุณสมบัติ CurrentFigure ของวัตถุ root หรือคุณสมบัติ CurrentObject หรือ CurrentAxes ของวัตถุ figure จะมีก็เพียงการเปลี่ยนแปลงค่า CallbackObject ของวัตถุ root

เมื่อไม่มี callbacks กำลังทำงานอยู่ในขณะนั้น **gcbo** จะให้ค่า [ ] หรือเมทริกซ์ว่าง

✪ *gcf*

จุดมุ่งหมาย เป็นคำสั่งให้ดึง handle ของรูปปัจจุบัน (gcf=Get current figure)

รูปแบบ **h = gcf**

คำอธิบาย

คำสั่ง **h = gcf** จะให้ handle ของ current figure หรือรูปภาพปัจจุบัน รูปภาพปัจจุบันนี้หมายถึง figure window ซึ่งคำสั่งในด้านกราฟฟิกส์ต่างเช่น plot, title หรือ surf จะเขียนผลของมันลงไปในรูปนี้ ถ้าไม่มีรูปภาพอยู่ในขณะนั้น MATLAB จะสร้างรูปขึ้นใหม่หนึ่งรูปแล้วให้ค่า handle ของรูปใหม่นั้น เราสามารถหลีกเลี่ยงการสร้างรูปใหม่ในการที่จะหา handle ของรูปภาพได้โดยการใช้คำสั่ง

```
get(0,'CurrentFigure')
```

✪ *gco*

จุดมุ่งหมาย ให้ handle ของวัตถุปัจจุบันหรือ current object (gco = get current object)

รูปแบบ **h = gco**

h = gco(figure_handle)

คำอธิบาย

คำสั่ง **h = gco** จะให้ handle ของ current object ส่วน **h = gco(figure_handle)** จะให้ค่า current object สำหรับ figure ที่กำหนดด้วนค่า figure_handle.

หมายเหตุ

วัตถุปัจจุบันหรือ current object คือวัตถุสุดท้ายที่มีการกดปุ่มเมาส์ลงบนวัตถุนั้น หรือวัตถุที่เป็น uimenus ซึ่งเกิดการทำงานขึ้นหลังสุด ถ้าหากว่าเรากดเมาส์ลงในรูป โดยไม่ได้กดลงไปบนวัตถุใดๆ ซึ่งเป็นลูกของรูปนั้น รูปนั้นจะกลายเป็น current object

MATLAB จะเก็บ handle ของ the current object ลงในค่าคุณสมบัติ CurrentObject ของวัตถุ figure ค่า CurrentObject ของ CurrentFigure ไม่ได้หมายความถึงวัตถุที่ callback ทำงานล่าสุดเสมอไป การขัดจังหวะของ callback ของวัตถุหนึ่งด้วย callback ของอีกวัตถุหนึ่ง สามารถที่จะเปลี่ยนแปลงค่าคุณสมบัติ CurrentObject หรือแม้แต่ค่า CurrentFigure สำหรับ callbacks บางแบบ เช่น CreateFcn, DeleteFcn และ uimenu Callback จะไม่มีการเปลี่ยนแปลงค่า CurrentFigure หรือ CurrentObject

หากต้องการทราบ handle ของวัตถุที่แน่นอนเราแนะนำให้ใช้คำสั่ง **gcbo** เพราะจะทำให้เรามั่นใจใน handle ที่ได้ตลอดเวลาไม่ว่า callback จะมีลักษณะคำสั่งเป็นอย่างไรและมีการขัดจังหวะในระหว่างที่ callback กำลังทำงานหรือไม่ก็ตาม

ตัวอย่าง คำสั่งต่อไปนี้จะให้ค่า handle ของ current object ใน figure window 2

```
» h = gco(2)

h =
   146.0001
```

ในบทนี้เป็นการแนะนำ handle ของวัตถุ และได้เสนอคุณสมบัติเบื้องต้นรวมถึงวิธีการแก้ไขคุณสมบัติเหล่านั้นของวัตถุ และท้ายที่สุดเราได้แนะนำให้รู้จักฟังก์ชันที่เกี่ยวข้องกับ handle ของวัตถุต่างๆ ซึ่งเราจะนำคำสั่งต่างๆนี้ไปใช้ร่วมกับในบทต่อไป ที่จะเป็นการกล่าวถึงคุณสมบัติของวัตถุต่างๆให้ละเอียดขึ้น

บทที่ 13 Object Property

ในบทนี้เราจะกล่าวถึงคุณสมบัติของวัตถุที่สร้างขึ้นเป็นรูปภาพ (Graphical Object) ซึ่งคุณสมบัติที่เราจะกล่าวถึงในบทนี้จะเป็นคุณสมบัติของวัตถุบางประเภทเท่านั้น เนื่องจากวัตถุที่สร้างขึ้นเป็นรูปภาพนั้นมีหลายแบบและแต่ละแบบก็จะมีเอกลักษณะของมันแตกต่างกันออกไป อย่างไรก็ดี คุณสมบัติที่ทำให้วัตถุเหล่านั้นเกิดความแตกต่างกันออกไปจะมีไม่มากนัก ดังนั้นเราจะกล่าวถึงเฉพาะคุณสมบัติของวัตถุที่เราใช้กันอยู่เป็นประจำ และที่สำคัญก็คือคุณสมบัติเหล่านั้นเป็นคุณสมบัติที่จะช่วยให้เราสามารถเขียน GUI ขั้นพื้นฐานได้

เนื่องจาก MATLAB เป็นโปรแกรมที่สามารถสร้างรูปกราฟที่ซับซ้อนและมีความอ่อนตัวในการสร้างรูปเหล่านั้นมาก ดังนั้นจึงไม่น่าแปลกใจเลยว่าวัตถุแต่ละวัตถุจะประกอบด้วยคุณสมบัติมากมาย เพราะคุณสมบัติแต่ละคุณสมบัติจะควบคุมพฤติกรรมหรือลักษณะของวัตถุนั้นเพียงอย่างเดียว ทำให้การที่จะควบคุมวัตถุต่างๆ ได้อย่างสมบูรณ์จำเป็นต้องใช้คุณสมบัติมากมาย

ในบทนี้เราจะกล่าวถึงวัตถุและคุณสมบัติของวัตถุต่อไปนี้คือ

1. **Figure Object**
2. **Axes Object**
3. **Uicontrol Object**
4. **Uimenu Object**
5. **UiContextMenu Object**

ซึ่งเราจะเห็นว่ารายการที่ 2 เป็นต้นมานั้นจะเป็น Children ของ Figure Object ทั้งนั้น ส่วนวัตถุหรือ object ที่เป็น children ของ axes นั้นเราจะไม่กล่าวถึงในที่นี้ ด้วยเหตุผลที่กล่าวมาแล้ว อย่างไรก็ตามสำหรับวัตถุเหล่านั้นเราจะมีคุณสมบัติที่สำคัญแสดงไว้ในภาคผนวกของเอกสารนี้ นอกจากนั้นแล้วการดูคุณสมบัติและค่าคุณสมบัติของวัตถุเหล่านั้นจะสามารถใช้คำสั่ง get และ set ตามที่เราได้ศึกษามาในบทที่ผ่านมาแล้ว นอกเหนือจากนั้นในบทที่เกี่ยวข้องกับตัวอย่างของการเขียน GUI เราจะมีการยกตัวอย่างถึงคุณสมบัติของวัตถุดังกล่าว รวมถึงการเปลี่ยนแปลงคุณสมบัติเหล่านั้นด้วย ซึ่งเราคิดว่าเพียงพอที่จะทำให้ผู้อ่านทำความเข้าใจกับคุณสมบัติที่เหลือเหล่านั้นได้

อย่างไรก็ตามถ้าหากว่าผู้อ่านต้องการจะทราบถึงการสร้างและคุณสมบัติวัตถุเหล่านั้นโดยละเอียดแล้ว เราแนะนำให้อ่านคู่มือที่มาพร้อมกับ MATLAB ไม่ว่าจะเป็น Reference Book I และ Reference Book II และอีกเล่มหนึ่งคือ Using Graphic Manual ซึ่งจะแสดงรายละเอียดของการสร้างรูปกราฟต่างๆ ไว้อย่างละเอียด นอกเหนือจากนั้นยังมีหนังสืออีกหลายเล่มที่เขียนเกี่ยวกับ GUI ใน MATLAB โดยเฉพาะหนังสือเหล่านั้นอาจจะสามารถสรุปรายละเอียดบางอย่างที่เราสนใจไว้

13.1 Figure

จุดมุ่งหมาย สร้าง figure graphics object

รูปแบบ

```
figure
figure('PropertyName',PropertyValue,...)
figure(h)
h = figure(...)
```

คำอธิบาย

คำสั่ง **figure** จะสร้างวัตถุที่เป็นกรอบรูปภาพ (figure graphics objects) ซึ่ง figure objects จะเป็นหน้าต่างต่างหากบนจอ ซึ่ง MATLAB ใช้แสดงผลรูปกราฟฟิคต่างๆ หน้าต่างรูปภาพที่สร้างขึ้นมาใหม่นี้จะใช้ค่าที่เป็นค่าเริ่มต้นในการสร้างวัตถุนี้ หากเราต้องการสร้างหน้าต่างรูปภาพที่มีคุณสมบัติต่างออกไป เราสามารถที่จะใช้คำสั่ง

```
figure('PropertyName', PropertyValue,...)
```

จะทำการสร้างหน้าต่างรูปภาพที่มีค่าคุณสมบัติตามที่เราต้องการ และ MATLAB จะใช้ค่าเริ่มต้นที่กำหนดมาสำหรับคุณสมบัติอื่นๆที่เราไม่ได้มีการกำหนดค่า

สำหรับคำสั่ง

```
figure(h)
```

จะทำสิ่งหนึ่งในสองสิ่งต่อไปนี้ขึ้นกับว่า handle h นั้นมีค่าขึ้นแล้วหรือยัง ถ้า h เป็น handle ของหน้าต่างรูปภาพที่มีอยู่แล้ว จะทำให้หน้าต่างที่มีรูปภาพนี้กลายเป็นรูปปัจจุบัน แต่ถ้า handle h ไม่มีอยู่ก่อน และ h เป็นเลขจำนวนเต็ม MATLAB จะสร้างรูปภาพขึ้นใหม่และกำหนด handle ให้เท่ากับ h อย่างไรก็ตามถ้าไม่มี handle ของรูปภาพที่มีค่าเป็น h อยู่ก่อนและ h ไม่เป็นเลขจำนวนเต็มเราจะได้ error เกิดขึ้นในรูปภาพ เพราะ handle ของรูปภาพต้องเป็นจำนวนเต็มเสมอ

ถ้าหากเราต้องการสร้างรูปภาพพร้อมทั้งให้ MATLAB บอก handle ของรูปภาพนั้นมาด้วย ให้เราใช้คำสั่ง

```
h = figure(...) returns the handle to the figure object
```

หมายเหตุ

ในการสร้างหน้าต่างรูปภาพ MATLAB จะสร้างหน้าต่างรูปภาพขึ้นใหม่ ซึ่งคุณสมบัติต่างๆ จะควบคุมด้วยคุณสมบัติเริ่มต้น ไม่ว่าจะเป็นการตั้งจากโรงงาน หรือผู้ใช้เป็นผู้กำหนด เราสามารถที่จะกำหนดคุณสมบัติและค่าของมันโดยการกำหนดคุณสมบัติและค่าคุณสมบัติเป็นคู่ๆ หรืออาจจะใช้ structure หรือ arrays หรือ cell arrays ก็ได้ ตามรูปแบบของคำสั่ง get และ set ส่วนคำสั่ง gcf จะเป็นการกำหนดให้แสดง handle ของรูปปัจจุบัน

ตัวอย่างต่อไปนี้จะเป็นการแสดงวิธีการสร้างหน้าต่างรูปที่มีขนาด หนึ่งในสี่ของหน้าจอคอมพิวเตอร์ และวางที่ตำแหน่งที่มุมบนซ้ายของจอคอมพิวเตอร์ คำสั่งที่เกี่ยวข้องก็จะเป็นการหา

ขนาดของวัตถุ root ซึ่งก็คือขนาดของจอคอมพิวเตอร์นั่นเอง ซึ่งคุณสมบัติที่บอกขนาดของจอก็คือคุณสมบัติ ScreenSize และคุณสมบัตินี้จะเป็นเวกเตอร์ของตัวเลข 4 ตัวที่แสดงตำแหน่ง [ซ้าย, ล่าง, กว้าง, สูง]: ตัวอย่างการสร้างรูปภาพเป็นดังนี้

```
» scrsz =get(0,'ScreenSize')
scrsz =
    1     1    1024     768

» figure('Position',[1 scrsz(4)/2 scrsz(3)/2 scrsz(4)/2])
```

การตั้งค่าเริ่มต้นให้กับหน้าต่างรูปภาพจะสามารถกระทำได้เมื่อเราอยู่ที่ระดับรากของคุณสมบัติ โดยใช้คำสั่ง

```
set(0,'DefaultFigureProperty',PropertyValue...)
```

เมื่อ Property เป็นชื่อของคุณสมบัติที่เราต้องการกำหนดค่าเริ่มต้นให้ และ PropertyValue คือค่าที่เราต้องการกำหนดให้คุณสมบัตินั้นๆ ลองใช้คำสั่ง set และ get ในการดูคุณสมบัติต่างๆ ของรูปภาพ

สำหรับคุณสมบัติของรูปภาพที่สำคัญจะมีตามตารางต่อไปนี้ โดยเราได้จัดเรียงไว้เป็นกลุ่มๆ ตามหน้าที่ของมัน และโดยทั่วไปชื่อคุณสมบัตินั้นจะบอกถึงว่ามันมีหน้าที่กำหนดคุณสมบัติในด้านใด

Property Name	Property Description	Property Value
การกำหนดตำแหน่งของรูป (Positioning the Figure)		
Position	ตำแหน่งและขนาดของหน้าต่างรูปภาพ	**ค่า**: เวกเตอร์แสดงตำแหน่ง [left, bottom, width, height] **ค่าเริ่ม**: ขึ้นกับลักษณะของจอภาพ
Units	หน่วยที่ใช้ในการแปรความขนาดและตำแหน่ง	**ค่า**: inches, centimeters, normalized, points, pixels, characters **ค่าเริ่ม**: pixels
การกำหนดรูปร่างที่ปรากฏ (Specifying Style and Appearance)		
Color	สีพื้นของรูป	**ค่า**: ColorSpec **ค่าเริ่ม**: ขึ้นกับชนิดของ color scheme
MenuBar	เลือกว่าจะให้รูปนั้นแสดงแถบเมนูบนหน้าต่างหรือไม่	**ค่า**: none, figure **ค่าเริ่ม**: figure
Name	ชื่อหน้าต่างรูปภาพ	**ค่า**: string **ค่าเริ่ม**: ' ' (string ว่าง)
NumberTitle	กำหนดให้แสดงหรือไม่แสดง คำว่า	**ค่า**: on, off

Property Name	Property Description	Property Value
	"Figure No. n", เมื่อ n คือหมายเลขของหน้าต่างรูป	**ค่าเริ่ม**: on
Resize	กำหนดว่าเราจะสามารถเปลี่ยนขนาดของหน้าต่างรูปภาพได้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
SelectionHighlight	เลือกว่าจะเน้นสีรูปเมื่อมีการเลือกหรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
Visible	ให้หน้าต่างรูปภาพนี้มองเห็นได้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
WindowStyle	เลือกรูปแบบของหน้าต่างแบบปกติหรือแบบไม่มีแถบเครื่องมือและแถบเมนู	**ค่า**: normal, modal **ค่าเริ่ม**: normal
การควบคุมสี (Controlling the Colormap)		
Colormap	colormap ของรูปภาพ	**ค่า**: เมทริกซ์ขนาด m x 3 ของค่าสี RGB **ค่าเริ่ม**: jet colormap
Dithermap	Colormap ที่ใช้สำหรับข้อมูล truecolor data บนการแสดงผลของสีแบบ pseudocolor	**ค่า**: เมทริกซ์ขนาด m x 3 ของค่าสี RGB **ค่าเริ่ม**: colormap ที่ให้สีได้เต็มที่ที่สุด
DithermapMode	ยอมให้ MATLABสร้างdithermap	**ค่า**: auto, manual **ค่าเริ่ม**: manual
FixedColors	สีที่ไม่ได้นำมาจาก colormap	**ค่า**: เมทริกซ์ขนาด m x 3 ของค่าสี RGB ค่านี้เป็น read only
MinColormap	จำนวนสีที่ต่ำที่สุดของระบบตารางสีที่จะใช้	**ค่า**: scalar **ค่าเริ่ม**: 64
ShareColors	ยอมให้ MATLAB ร่วมใช้ color table slots	**ค่า** on, off **ค่าเริ่ม**: on
การกำหนดวิธีการเขียนรูป (Specifying the Renderer)		
BackingStore	ยอม off screen pixel buffering	**ค่า**: on, off **ค่าเริ่ม**: on
DoubleBuffer	การสร้างภาพเคลื่อนไหวแบบง่าย โดยไม่มีการกระพริบภาพ	**ค่า**: on, off **ค่าเริ่ม**: off
Renderer	วิธีการเขียนภาพสีทับภาพเดิมและการพิมพ์	**ค่า**: painters, zbuffer, OpenGL **ค่าเริ่ม**: เลือกอัตโนมัติโดย MATLAB

Property Name	Property Description	Property Value
ข้อมูลทั่วไป (General Information About the Figure)		
Children	Handle ของ uicontrol, uimenu และ uicontextmenu ที่แสดงอยู่ในรูปนี้	**ค่า**: เวกเตอร์ของ handles
Parent	จะมีเฉพาะ root object เท่านั้นที่เป็น parent ของรูปภาพทั้งหมด ค่านี้จะเปลี่ยนไม่ได้	Value: always 0
Selected	แสดงให้ทราบว่าขณะนี้รูปภาพนี้ได้กำลังถูกเลือกจากผู้ใช้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
Tag	ชื่อที่ผู้ใช้กำหนดให้กับวัตถุนั้น	**ค่า**: string **ค่าเริ่ม**: string ว่าง
Type	ชนิดของวัตถุ (read only)	**ค่า**: string คำว่า 'figure'
UserData	ข้อมูลที่ผู้ใช้กำหนด	**ค่า**: เมทริกซ์ **ค่าเริ่ม**: [ ] (เมทริกซ์ว่าง)
RendererMode	กำหนดให้เลือกวิธีการวาดโดยอัตโนมัติหรือผู้ใช้กำหนด	**ค่า**: auto, manual **ค่าเริ่ม**: auto
ข้อมูลเกี่ยวกับสภาวะปัจจุบัน (Information About Current State)		
CurrentAxes	Handle ของแกนปัจจุบันในรูปภาพนี้	**ค่า**: handle ของ axes
CurrentCharacter	ตัวบนแป้นพิมพ์ที่กดตัวสุดท้ายในขณะที่รูปภาพนี้เป็นรูปปัจจุบัน	**ค่า**: character ตัวเดียว (read only)
CurrentObject	Handle ของวัตถุปัจจุบันในรูปภาพนี้	**ค่า**: graphics object handle
CurrentPoint	ตำแหน่งบนรูปภาพ ที่ผู้ใช้กดเมาส์ครั้งสุดท้ายในรูปภาพนี้	**ค่า**: เวกเตอร์ขนาด 2-element ที่กำหนด [x-coord, y-coord]
SelectionType	ชนิดการเลือกด้วยเมาส์ (read only)	**ค่า**: normal, extended, alt, open
Callback Routine Execution		
BusyAction	เป็นการกำหนดวิธีการที่จะใช้ในการเรียกใช้คำสั่งใหม่ในขณะที่ยังมีคำสั่งเก่าดำเนินการค้างอยู่	**ค่า**: cancel, queue **ค่าเริ่ม**: queue
ButtonDownFcn	กำหนดการเรียกใช้คำสั่งที่จะทำงาน หากมีการกดปุ่มเมาส์ลงในบริเวณที่ว่างในระบบแกน	**ค่า**: string **ค่าเริ่ม**: string ว่าง
CloseRequestFcn	กำหนดการเลือกใช้คำสั่งที่จะทำงาน เมื่อผู้ใช้ได้เลือกให้มีการใช้คำสั่ง close กับ	**ค่า**: string **ค่าเริ่ม**: string ว่าง

Property Name	Property Description	Property Value
	รูปภาพ	
CreateFcn	กำหนดการเลือกใช้คำสั่งที่จะทำงานก่อนที่จะมีการสร้างรูปภาพขึ้น	**ค่า**: string **ค่าเริ่ม**: string ว่าง
DeleteFcn	กำหนดการเลือกใช้คำสั่งที่จะทำงานก่อนที่จะมีการลบรูปภาพ	**ค่า**: string **ค่าเริ่ม**: string ว่าง
Interruptible	กำหนดว่าคำสั่งที่กำลังดำเนินการอยู่นั้นสามารถขัดจังหวะได้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on (สามารถขัดจังหวะได้)
KeyPressFcn	กำหนดการเลือกใช้คำสั่งที่จะทำงานเมื่อมีการกดแป้นในหน้าต่างรูปภาพ	**ค่า**: string **ค่าเริ่ม**: string ว่าง
ResizeFcn	กำหนดการเลือกใช้คำสั่งที่จะทำงานในกรณีที่ผู้ใช้มีการปรับขนาดของรูปภาพ	**ค่า**: string **ค่าเริ่ม**: string ว่าง
UIContextMenu	กำหนด context menu กับรูปภาพนี้	**ค่า**: handle ของ Uicontrextmenu
WindowButtonDownFcn	กำหนดคำสั่งที่จะทำงาน เมื่อมีการกดเมาส์ภายในหน้าต่างรูปภาพ	**ค่า**: string **ค่าเริ่ม**: string ว่าง
WindowButtonMotionFcn	กำหนดคำสั่งที่จะทำงานเมื่อมีการลากตัวชี้ของเมาส์เข้าไปภายในหน้าต่างรูปภาพ	**ค่า**: string **ค่าเริ่ม**: string ว่าง
WindowButtonUpFcn	กำหนดคำสั่งที่จะทำงาน เมื่อมีการปล่อยปุ่มเมาส์ภายในหน้าต่างรูปภาพ	**ค่า**: string **ค่าเริ่ม**: string ว่าง
การควบคุมการเข้าถึงวัตถุนี้ (Controlling Access to Objects)		
IntegerHandle	กำหนดให้ handle ของรูปภาพต้องเป็นจำนวนเต็มหรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on (handle เป็นจำนวนเต็ม)
HandleVisibility	กำหนดว่า handle ของหน้าต่างรูปนี้จะมองเห็นได้โดยผู้ใช้หรือไม่	**ค่า**: on, callback, off **ค่าเริ่ม**: on
HitTest	กำหนดว่าหน้าต่างรูปนี้จะกลายเป็นวัตถุปัจจุบันได้หรือไม่ (ดูคุณสมบัติ CurrentObject ประกอบ)	**ค่า**: on, off **ค่าเริ่ม**: on
NextPlot	กำหนดว่าจะเขียนรูปที่จะต้องสร้างต่อไปบนหน้าต่างรูปภาพนี้อย่างไร	**ค่า**: add, replace, replacechildren **ค่าเริ่ม**: add
การกำหนดตัวชี้ (Defining the Pointer)		
Pointer	กำหนดสัญลักษณ์ของตัวชี้ของเมาส์เมื่อปรากฏอยู่ในหน้าต่างรูปภาพ	**ค่า**: crosshair, arrow, watch, topl, topr, botl, botr, circle, cross, fleur, left, right, top, bottom,

Property Name	Property Description	Property Value
		fullcrosshair, ibeam, custom **ค่าเริ่ม**: arrow
PointerShapeCData	ข้อมูลที่ใช้สร้างสัญลักษณ์ของตัวชี้ของเมาส์	**ค่า**: 16-by-16 matrix **ค่าเริ่ม**: เลือกคุณสมบัติ Pointer ไปที่ custom แล้วดูค่าที่แสดง
PointerShapeHotSpot	กำหนดตำแหน่งของ active spot ของตัวชี้ของเมาส์	**ค่า**: เวกเตอร์ขนาด 2-element ที่บอกค่า [row, column] **ค่าเริ่ม**: [1,1]
คุณสมบัติที่มีผลต่อการพิมพ์ (Properties That Affect Printing)		
InvertHardcopy	เปลี่ยนสีของรูปเมื่อมีการพิมพ์	**ค่า**: on, off **ค่าเริ่ม**: on
PaperOrientation	กำหนดวิธีการหมุนกระดาษ	**ค่า**: portrait, landscape **ค่าเริ่ม**: portrait
PaperPosition	กำหนดตำแหน่งของรูปบนกระดาษเมื่อสั่งพิมพ์	**ค่า**: เวกเตอร์ขนาด 4-element ที่บอกค่า [left, bottom, width, height]
PaperPositionMode	กำหนดให้ระบบ WYSIWYG ทำงานหรือไม่	**ค่า**: auto, manual **ค่าเริ่ม**: manual
PaperSize	ขนาดของกระดาษ โดยมีหน่วยตามหน่วยที่กำหนดด้วยคุณสมบัติ PaperUnits และขนาดนี้จะเป็นตามคุณสมบัติ PaperType ด้วย ค่าคุณสมบัตินี้เป็นแบบ read only	**ค่า**: [width, height]
PaperType	เลือกขนาดของกระดาษตามระบบกระดาษมาตราฐาน	**ค่า**: see property description **ค่าเริ่ม**: usletter
PaperUnits	หน่วยที่ใช้ในการกำหนดขนาดและตำแหน่งของคุณสมบัติ PaperSize และ PaperPosition	**ค่า**: normalized, inches, centimeters, points **ค่าเริ่ม**: inches

การดูค่าและค่าเริ่มต้นของอุปกรณ์เหล่านั้นเราสามารถที่จะใช้คำสั่ง set และ get ตามที่ได้กล่าวถึงมาแล้วในบทที่ผ่านมา

13.2 Axes

จุดมุ่งหมาย สร้างวัตถุที่เป็นระบบแกน ความหมายของแกนในที่นี้หมายถึงพื้นที่สี่เหลี่ยมที่ปรากฏบนหน้าต่าง figure ซึ่งใช้ในการเขียนรูปกราฟ หรือจุดลักษณะต่างๆ ลงไป หรือกล่าวง่ายๆคือพื้นที่ที่ใช้แสดงผล

รูปแบบ

```
axes
axes('PropertyName',PropertyValue,...)
axes(h)
h = axes(...)
```

คำอธิบาย

ฟังก์ชัน **axes** นี้เป็นฟังก์ชันระดับต่ำหรืออยู่ในขั้นพื้นฐาน (low-level function) เพื่อใช้ในการเขียนรูปที่ต้องการลงไปในหน้าต่าง Figure ที่เป็นปัจจุบัน (current figure) ที่เรียกว่าฟังก์ชันนี้เป็นฟังก์ชันขั้นต่ำก็เพราะว่า โดยปกติหากเราต้องการที่จะสร้างกราฟ MATLAB จะทำการสร้างระบบแกนนี้ให้โดยอัตโนมัติอยู่แล้วเราจึงไม่จำเป็นที่จะต้องเรียกใช้คำสั่งนี้ในกรณีทั่วๆไป

สำหรับคุณสมบัติต่างๆของ axes object และรายละเอียดคร่าวๆของคุณสมบัติเหล่านั้นได้แสดงไว้ในตารางต่อไปนี้

Property Name	Property Description	Property Value
การควบคุมรูปแบบและรูปร่าง (Controlling Style and Appearance)		
Box	สลับค่าให้ axes plot box เป็น on หรือ off	**ค่า**: on, off **ค่าเริ่ม**: off
Clipping	คุณสมบัตินี้ไม่มีผลต่อ axes โดย axes จะทำการตัด (clip) ส่วนต่างๆให้พอดีกับหน้าต่างของ figure เสมอ	
GridLineStyle	ลักษณะของเส้นที่ใช้เป็นเส้น grid ในแกน	**ค่า**: -, --, :, -., none **ค่าเริ่ม**: :(เส้นประ)
Layer	ให้เขียนกราฟอยู่ข้างบนหรือข้างล่างแกน	**ค่า**: bottom, top **ค่าเริ่ม**: bottom
LineStyleOrder	ลำดับการใช้ลักษณะของเส้นกราฟที่จะเขียนลงบนแกน	**ค่า**: LineSpec **ค่าเริ่ม**: - (เส้นต่อ)
LineWidth	ความหนาของเส้นแกนในหน่วย points (1 point = 1/72")	**ค่า**: ค่าความหนาเป็น points **ค่าเริ่ม**: 0.5 points
SelectionHighlight	ให้เน้นสีของ axes เมื่อเลือกคุณสมบัตินี้ให้เป็น on	**ค่า**: on, off **ค่าเริ่ม**: on
TickDir	ทิศทางของขีดย่อยบนแกน (tick marks)	**ค่า**: in, out **ค่าเริ่ม**: in(2-D),out(3-D)
TickDirMode	ให้ MATLAB หรือผู้ใช้เป็นผู้กำหนดทิศทางของขีด	**ค่า**: auto, manual

Property Name	Property Description	Property Value
	ย่อยบนแกน	**ค่าเริ่ม**: auto
TickLength	ความยาวของขีดย่อยบนแกน เปรียบเทียบกับค่าบรรทัดฐานคือความยาวของแกน และกำหนดเป็น two-element vector	**ค่า**: [2-D 3-D] **ค่าเริ่ม**: [0.01 0.025]
Visible	กำหนดให้แกนนี้มองเห็นได้หรือมองเห็นไม่ได้	**ค่า**: on, off **ค่าเริ่ม**: on
XGrid, YGrid, ZGrid	สลับระหว่างให้เขียนหรือไม่ให้เขียนเส้น grid บนแกนที่กำหนด	**ค่า**: on, off **ค่าเริ่ม**: off
ข้อมูลทั่วไป (General Information About the Axes)		
Children	Handles ของ images, lights, lines, patches, surfaces และ text ที่แสดงอยู่บนแกนนี้	**ค่า**: เวกเตอร์ handles
CurrentPoint	ตำแหน่งที่กดเมาส์ครั้งสุดท้ายในพื้นที่ของแกน กำหนดด้วยตัวเลขในหน่วยที่กำลังใช้อยู่บนแกนนั้น	**ค่า**: เมทริกซ์ขนาด 2x3
HitTest	กำหนดให้แกนนี้สามารถกลายเป็น current object ได้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
Parent	Handle ของหน้าต่าง figure ที่บรรจุแกนนี้อยู่	**ค่า**: ค่าตัวเลขของ figure handle
Position	ตำแหน่งของแกน บนหน้าต่าง figure	**ค่า**: [left bottom width Height] **ค่าเริ่ม**: [0.1300 0.1100 0.7750 0.8150] ในหน่วยบรรทัดฐาน
Selected	กำหนดว่าขณะนี้แกนนี้อยู่สภาพที่ถูกเลือกอยู่หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
Tag	ชื่อของแกนนี้ กำหนดขึ้นโดยผู้ใช้	**ค่า**: any string **ค่าเริ่ม**: '' (string ว่าง)
Type	ชนิดของรูปที่ปรากฏอยู่ ค่านี้อ่านได้เท่านั้น เปลี่ยนแปลงไม่ได้	**ค่า**: string คำว่า 'axes'
Units	หน่วยที่ใช้ในการคำนวณระยะบนแกน ซึ่งจะนำไปใช้โดยคุณสมบัติ 'Position'	**ค่า**: inches, centimeters, Characters, normalized, Points, pixels **ค่าเริ่ม**: Normalized
UserData	ข้อมูลที่กำหนดโดยผู้ใช้	**ค่า**: any matrix **ค่าเริ่ม**: [ ] (เมทริกซ์ว่าง)
การเลือกแบบตัวหนังสือ (Selecting Fonts and Labels)		
FontAngle	เลือกแบบตัวหนังสือว่าเป็นตัวปกติหรือตัวเอียง	**ค่า**: normal, italic, Oblique

Property Name	Property Description	Property Value
		ค่าเริ่ม: normal
FontName	ชื่อของแบบตัวหนังสือที่ใช้ในแกน เช่น Helvetica หรือ Courier เป็นต้น	**ค่า**: ชื่อแบบตัวหนังสือที่ใช้ได้และยอมรับในระบบปฏิบัติการ **ค่าเริ่ม**: โดยทั่วไปHelvetica
FontSize	ขนาดของตัวหนังสือที่ใช้ในแกน	**ค่า**: เลขจำนวนเต็มในหน่วยที่ใช้กำหนดขนาดตัวหนังสือ **ค่าเริ่ม**: 10
FontUnits	หน่วยของตัวหนังสือที่ใช้ คุณสมบัตินี้จะนำไปใช้กับคุณสมบัติ FontSize	**ค่า**: points, normalized, inches, centimeters, pixels **ค่าเริ่ม**: points
FontWeight	เลือกว่าตัวหนังสือมีความหนาปกติหรือหนามาก	**ค่า**: normal, bold, light, demi **ค่าเริ่ม**: normal
Title	ค่า Handle ของตัวหนังสือที่เป็น title	**ค่า**: ค่า handle ของตัวหนังสือที่เป็น title
XLabel, YLabel, ZLabel	ค่า Handles ของตัวหนังสือที่เป็นชื่อแกนที่กำหนด	**ค่า**: ค่า Handles ของตัวหนังสือที่เป็นชื่อแกนที่กำหนด
XTickLabel, YtickLabel, ZtickLabel	กำหนดลักษณะของ tick mark labels สำหรับแกนที่กำหนด	**ค่า**: เมทริกซ์ของ strings **ค่าเริ่ม**: ค่าตัวเลขเรียกอัตโนมัติโดย MATLAB
XTickLabelMode, YTickLabelMode, ZtickLabelMode	เลือกว่าให้ MATLAB หรือผู้ใช้เป็นผู้กำหนดลักษณะของ tick mark labels	**ค่า**: auto, manual **ค่าเริ่ม**: auto
การควบคุมขนาดของแกน (Controlling Axis Scaling)		
XaxisLocation	กำหนดตำแหน่งของแกน x	**ค่า**: top, bottom **ค่าเริ่ม**: bottom
YaxisLocation	กำหนดตำแหน่งของแกน y	**ค่า**: right left **ค่าเริ่ม**: left
XDir, YDir, Zdir	กำหนดให้ค่าตามทิศทางของแกนให้มีค่าเพิ่มขึ้นหรือลดลงตามแกนที่กำหนด	**ค่า**: normal, reverse **ค่าเริ่ม**: normal
XLim, Ylim, Zlim	กำหนดค่าสูงสุดและต่ำสุดของแกนที่กำหนด	**ค่า**: [min max] **ค่าเริ่ม**: ค่าตัวเลขสูงสุดและต่ำสุดเลือกอัตโนมัติโดย MATLAB

Property Name	Property Description	Property Value
XlimMode, YlimMode, ZlimMode	กำหนดว่าให้ MATLAB หรือผู้ใช้เป็นผู้กำหนดค่าสูงสุดหรือต่ำสุดของแกนที่ต้องการ	**ค่า**: auto, manual **ค่าเริ่ม**: auto
XScale, YScale, Zscale	กำหนดว่าให้ใช้สเกลแบบ linear หรือ logarithmic ของแกนที่ต้องการ	**ค่า**: linear, log **ค่าเริ่ม**: linear (แต่อาจเปลี่ยนอัตโนมัติหากเรียกการ plot แบบ log มาใช้)
XTick, YTick, Ztick	กำหนดตำแหน่งของ ticks marks	**ค่า**: เวคเตอร์ของข้อมูลที่บอกตำแหน่งของเครื่องหมาย tick marks **ค่าเริ่ม**: ค่าจะคำนวณอัตโนมัติโดย MATLAB
XTickMode, YTickMode, ZtickMode	เลือกให้ MATLAB หรือผู้ใช้เป็นผู้กำหนดค่าของตำแหน่ง tick mark ในแกนที่ต้องการ	**ค่า**: auto, manual **ค่าเริ่ม**: auto
การควบคุมมุมมอง (Controlling the View)		
CameraPosition	กำหนดตำแหน่งของผู้มองว่าจะมองแกนนี้จากตำแหน่งใด	**ค่า**: [x,y,z] ของพิกัดในระบบแกน **ค่าเริ่ม**: ค่าคำนวณอัตโนมัติโดย MATLAB
CameraPositionMode	เลือกให้ MATLAB หรือผู้ใช้เป็นผู้เลือกตำแหน่งของการวางตำแหน่งผู้มองแกนนี้	**ค่า**: auto, manual **ค่าเริ่ม**: auto
CameraTarget	จุดกลางที่มองรูป เทียบกับตำแหน่งของผู้มอง	**ค่า**: [x,y,z] ของพิกัดในระบบแกน **ค่าเริ่ม**: คำนวณอัตโนมัติโดย MATLAB
CameraTargetMode	เลือกให้ MATLAB หรือผู้ใช้เป็นผู้เลือกตำแหน่งที่จะมองแกนนี้	**ค่า**: auto, manual **ค่าเริ่ม**: auto
CameraUpVector	ทิศทางแนวการมองแกนว่าเป็นมุมสูงเท่าใด	**ค่า**: [x, y, z] ในหน่วยของระบบแกน **ค่าเริ่ม**: คำนวณอัตโนมัติโดย MATLAB
CameraUpVectorMode	เลือกให้ MATLAB หรือผู้ใช้เป็นผู้เลือกตำแหน่งของผู้มองตามมุมสูงของการมองแกนนี้	**ค่า**: auto, manual **ค่าเริ่ม**: auto
CameraViewAngle	มุมกว้างของผู้มองว่าจะมองภาพกว้างเท่าใด	**ค่า**: อยู่ระหว่าง 0°—180°

Property Name	Property Description	Property Value
		ค่าเริ่ม: คำนวณอัตโนมัติโดย MATLAB
CameraViewAngleMode	เลือกให้ MATLAB หรือผู้ใช้เป็นผู้เลือกมุมกว้างของการมองแกนนี้	**ค่า**: auto, manual **ค่าเริ่ม**: auto
Projection	เลือกชนิดของ projection	**ค่า**: orthographic, perspective **ค่าเริ่ม**: orthographic
การควบคุมสัดส่วนของแกน (Controlling the Axes Aspect Ratio		
DataAspectRatio	กำหนดอัตราส่วนของสเกลตามแกนต่างๆ	**ค่า**: ค่าสัมพัทธ์สามค่าของ [dx dy dz] **ค่าเริ่ม**: คำนวณอัตโนมัติโดย MATLAB
DataAspectRatioMode	เลือกให้ MATLAB หรือผู้ใช้เป็นผู้เลือกการกำหนดอัตราส่วนสเกล	**ค่า**: auto, manual **ค่าเริ่ม**: auto
PlotBoxAspectRatio	ค่าสเกลสัมพัทธ์เทียบกับแกน	**ค่า**: ค่าสัมพัทธ์สามค่าของ [dx dy dz] **ค่าเริ่ม**: คำนวณอัตโนมัติโดย MATLAB
PlotBoxAspectRatioMode	เลือกให้ MATLAB หรือผู้ใช้เป็นผู้เลือกอัตราส่วนของเสกลตามแกนต่างๆ	**ค่า**: auto, manual **ค่าเริ่ม**: auto
การควบคุม Callback (Controlling Callback Routine Execution)		
BusyAction	เป็นการกำหนดวิธีการที่จะใช้ในการเรียกใช้คำสั่งใหม่ในขณะที่ยังมีคำสั่งเก่าดำเนินการค้างอยู่	**ค่า**: cancel, queue **ค่าเริ่ม**: queue
ButtonDownFcn	กำหนดการเรียกใช้คำสั่ง หากมีการกดปุ่มเมาส์ลงในบริเวณที่ว่างในระบบแกน	**ค่า**: string **ค่าเริ่ม**: string ว่าง
CreateFcn	กำหนดการเลือกใช้คำสั่งก่อนที่จะมีการสร้างระบบแกนขึ้น	**ค่า**: string **ค่าเริ่ม**: string ว่าง
DeleteFcn	กำหนดการเลือกใช้คำสั่งก่อนที่จะมีการลบระบบแกนออกไป	**ค่า**: string **ค่าเริ่ม**: String ว่าง
Interruptible	กำหนดว่าคำสั่งที่กำลังดำเนินการอยู่นั้นสามารถขัดจังหวะได้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
UIContextMenu	เป็นการกำหนดเพื่อใช้คู่กับคำสั่ง contextmenu	**ค่า**: ค่า handle ของ Uicontextmenu
การกำหนดการให้สีผิว (Specifying the Rendering Mode)		
DrawMode	กำหนดวิธีการให้สีผิว (rendering)	**ค่า**: normal, fast **ค่าเริ่ม**: normal

Property Name	Property Description	Property Value
การควบคุมการแสดงผล (Targeting Axes for Graphics Display)		
HandleVisibility	ควบคุมวิธีการเข้าถึงระบบแกนนั้นๆ	**ค่า**: on, callback, off **ค่าเริ่ม**: on
NextPlot	กำหนดวิธีการวาดกราฟใหม่ลงบนระบบแกนเดิมนี้	**ค่า**: add, replace, replacechildren **ค่าเริ่ม**: replace
คุณสมบัติที่ใช้กำหนดสี (Properties that Specify Color)		
AmbientLightColor	กำหนดสีของพื้นที่ผู้ใช้มองอยู่	**ค่า**: ColorSpec **ค่าเริ่ม**: [1 1 1]
CLim	กำหนดวิธีการคำนวณ Colormap	**ค่า**: [cmin cmax] **ค่าเริ่ม**: MATLAB ทำการคำนวณอัตโนมัติ
CLimMode	ให้ MATLAB หรือผู้ใช้กำหนดค่าในคำสั่ง Clim	**ค่า**: auto, manual **ค่าเริ่ม**: auto
Color	สีพื้นบนระบบแกน	**ค่า**: none, ColorSpec **ค่าเริ่ม**: none
ColorOrder	สีของเส้นที่ใช้ในกรณีที่เขียนกราฟหลายเส้น	**ค่า**: เมทริกซ์ขนาด m x 3 ของสี RGB **ค่าเริ่ม**: ขึ้นกับสีของระบบปฏิบัติการ
XColor, YColor, ZColor	สีของแกนและเครื่องหมาย tick marks	**ค่า**: ColorSpec **ค่าเริ่ม**: ขึ้นกับสีของระบบปฏิบัติการ

✪ ตัวอย่างที่เกี่ยวข้องกับคุณสมบัติของแกน

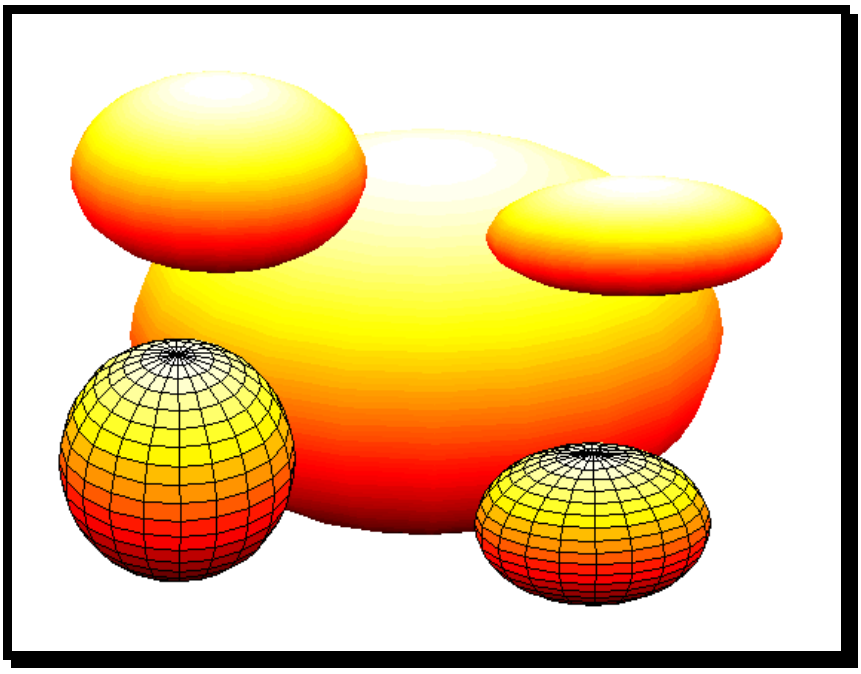
1. *ตัวอย่างการสร้างแกนหลายแกนและมีขนาดต่างกัน*

ตัวอย่างนี้จะแสดงให้เราเห็นว่า เราสามารถที่จะสร้างแกนหลายแกนเพื่อแสดงวัตถุแบบเดียวกันบนแต่ละแกน และแต่ละแกนจะมีขนาดแตกต่างกันออกไปได้ โดยไม่จำเป็นต้องเปลี่ยนแปลงข้อมูลที่จะสร้างวัตถุเหล่านั้นเหมือนกับที่เราต้องเปลี่ยนแปลงข้อมูลหากว่าเราจะแสดงวัตถุเหล่านั้นบนระบบแกนเดียวกัน

ตัวอย่างเช่นเราต้องการเขียนทรงกลมหลายๆขนาดขึ้นในรูปภพรูปภาพหนึ่ง โดยทรงกลมแต่ละอันจะมีขนาดที่ไม่เท่ากัน โดยเราสร้าง M-file ด้วยคำสั่งต่อไปนี้

```
h(1)= axes('Position',[0 0 1 1]);
sphere
shading interp
h(2)= axes('Position',[0 0 .4 .6]);
sphere

h(3)= axes('Position',[0 .5 .5 .5]);
sphere
shading interp

h(4)=axes('Position',[5 0 .4 .4]);
sphere

h(5)= axes('Position',[.5 .5 .5 .3]);
sphere
colormap(hot)
shading interp
set(h,'Visible','off')
```

ซึ่งเมื่อให้ m-file ทำงานเราจะให้รูปดังนี้



2. *ตัวอย่างการสร้างกราฟที่มีแกน X และ Y หลายค่า*

ในตัวอย่างต่อไปนี้เป็นการสร้างระบบแกนสองแกนลงไปในพื้นที่เดียวกัน ประโยชน์ของการทำเช่นนี้จะเป็นในกรณีที่เราต้องการสร้างกราฟที่มีแกน Y สองแกนซึ่งมีการแบ่งมาตรบนแกนทั้งสองแตกต่างกัน เพื่อเปรียบเทียบกับแกน X เพียงแกนเดียว หรือจะมีค่าตามแกน X สองมาตราก็ได้ ซึ่งจะทำให้ผู้อ่านกราฟสามารถที่จะเปรียบเทียบค่าและลักษณะการเปลี่ยนแปลงของเส้นกราฟทั้งสองได้พร้อมๆ กัน

แม้ว่าใน MATLAB 5.0 เป็นต้นมาจะมีคำสั่ง **plotyy** แต่เราพบว่าการใช้คำสั่งดังกล่าวมีข้อจำกัดมากและในความเป็นจริงแล้วไม่สะดวกในการไปใช้งานเพราะเราปรับเปลี่ยนค่าหรือกำหนดค่าคุณสมบัติต่างๆ ได้ค่อนข้างยาก ตัวอย่างนี้จึงแสดงวิธีการแก้ปัญหาของการใช้คำสั่งดังกล่าว แม้ว่าจะยุ่งยากกว่าแต่จะทำให้เรามีความอ่อนตัวกว่าในการปรับรูปแบบกราฟ

คุณสมบัติที่สำคัญในการสร้างกราฟที่มีแกน Y สองแกนเขียนลงไปร่วมกับแกน x เพียงแกนเดียวคือคุณสมบัติ XAxisLocation และ YAxisLocation ซึ่งเป็นคุณสมบัติที่ใช้ในกำหนดว่าจะวางแกน X และ Y ลงไปในด้านใด เราสามารถสร้างกราฟที่เราต้องการได้โดยการวางระบบแกนกราฟ 2 ระบบแกนไว้ในตำแหน่งที่กำหนด และใช้คุณสมบัติ XaxisLocation and YAxisLocation เพื่อกำหนดตำแหน่งของแกน X และ แกน Y ของแต่ละระบบแกน

ในตัวอย่างนี้เราจะสร้างกราฟขึ้นหนึ่งกราฟเพื่อใช้แสดงข้อมูลที่แตกต่างกัน 2 ชุด โดยใช้แกนด้านล่างและด้านซ้ายมือแทนระบบแกน x-y หนึ่ง และแกนด้านบนและด้านขวามือแทนระบบแกนอีกระบบแกนหนึ่ง

การใช้คำสั่งในระดับ low-level ของ line และ axes ช่วยให้เราสามารถที่จะวางวัตถุทับซ้อนกันลงไปได้โดยง่าย ขั้นตอนแรกจะเป็นการเขียนกราฟรูปแรกลงบนระบบแกนปกติ และหา Handle ของแกนที่เขียนลงไปเพื่อที่จะนำไปใช้ต่อไปในภายหลัง อันดับแรกเราสร้างค่า x1 และ x2 ซึ่งเป็นเวกเตอร์ของข้อมูล

```
» x1=linspace(0,30);
» y1=sin(x1).^exp(-x1/10);
» hl1= line(x1,y1,'color','r');
» ax1= gca;
» set(ax1,'Xcolor','r','Ycolor','r');
» grid on
```

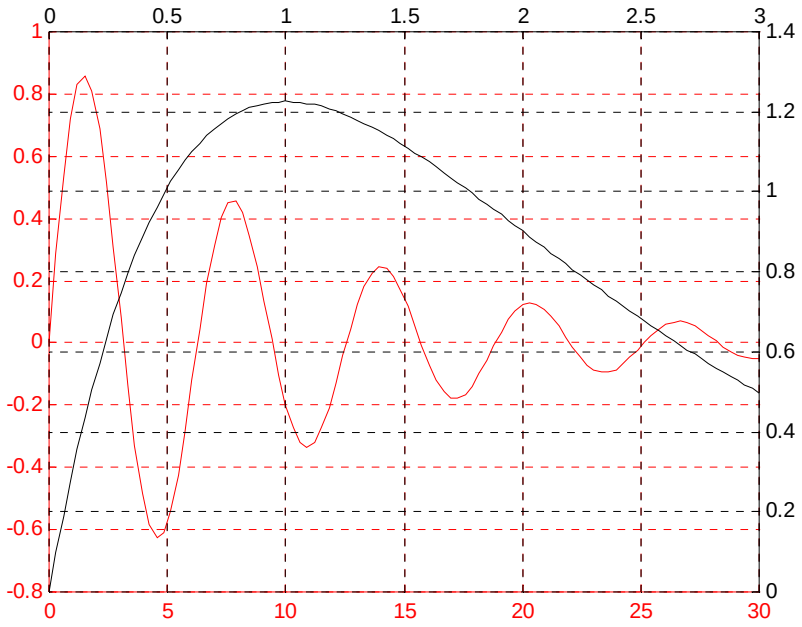
สำหรับคำสั่งสุดท้ายเป็นการสั่งให้สีของแกนกลายเป็นสีแดง ตามสีของเส้นกราฟเพื่อความสะดวกของผู้ใช้ในการอ่านค่า ขั้นต่อไปเราจะสร้างแกนอีกแกนหนึ่งลงในตำแหน่งเดียวกันกับแกนแรก แต่วางแกน x ไว้ด้านบนและแกน y ไว้ด้านขวามือ จากนั้นตั้งสีของระบบแกนเป็น none คือไม่มีสีเพื่อเราจะสามารถมองทะลุระบบแกนที่สร้างขึ้นภายหลังนี้ลงไปเห็นระบบแกนที่สร้างขึ้นก่อนได้ จากนั้นจะกำหนดสีของแกนให้เหมือนกับสีของกราฟเพื่อความสะดวกในการอ่าน ซึ่งตามที่กล่าวมาแล้วนั้นเราจะได้ชุดของคำสั่งดังต่อไปนี้

```
» ax2 = axes('Position',get(ax1,'Position'),...
'XAxisLocation','top',...
'YAxisLocation','right',...
'Color','none',...
'XColor','k','YColor','k');
```

จากนั้นเขียนข้อมูลชุดที่สองลงบนระบบแกนที่สอง โดยกำหนดสีให้เหมือนสีของแกนที่สอง ซึ่งมีคำสั่งเป็น

```
» x2=x1/10;
» y2=100*sin(x2/30).*exp(-x2);
» hl2 = line(x2,y2,'Color','k','Parent',ax2);
» grid on
```

ซึ่งเราจะได้กราฟตามที่แสดงในรูปต่อไปนี้



ขั้นต่อไปเราจะทำการเขียนเส้น grid ของระบบแกนที่เราสร้างขึ้นมา เพราะเมื่อเราสั่งให้ grid on ของทั้งสองระบบแกน เราจะพบว่าการแบ่งเส้น grid ของทั้งสองระบบแกนจะแบ่งไม่ตรงกันและจะทำให้เกิดความสับสนในการอ่านค่าเป็นอย่างมาก แม้ว่าเราจะกำหนดเส้น grid ให้มีสีที่แตกต่างกันก็ตาม ขั้นตอนต่อไปนี้จะแสดงวิธีการเขียน grid เพื่อให้เส้น grid ของทั้งสองระบบแกนทับกันสนิทเป็นเส้น grid ชุดเดียวกันพอดี

เราทราบอยู่แล้วว่าการเขียนเส้น grid จะเขียนที่ตำแหน่งของ tick mark ดังนั้นถ้าหากเรากำหนดตำแหน่งของ tick mark ขึ้นมาเอง เราก็จะสามารถเขียนเส้น grid ของทั้งสองระบบแกนให้ทับกันได้ ดังนั้นกุญแจสำคัญในการแก้ปัญหานี้ก็จะเป็นการกำหนด tick mark ของทั้งสองระบบแกนให้มีจำนวนเท่ากันและวางอยู่ในตำแหน่งที่ตรงกัน อันดับแรกเราจะต้องกำหนดจำนวน tick mark ที่เราต้องการให้ทั้งสองระบบแกนมีเสียก่อน คำนวณหาจำนวน tick mark ที่เราจะกำหนดก่อน สมมุติว่าเราจะกำหนดจำนวน tick mark ให้เท่ากับระบบแกนที่สองคือมี tick mark เท่ากับ 7 ตามแนวแกน x (อย่าลืมนับเส้นที่เป็นแกนและเส้นสุดท้ายด้วย) และทางแกน y จะมีจำนวน tick mark เท่ากับ 8 จากนั้นเราจะคำนวณขนาดและตำแหน่งและ tick mark ของระบบแกนแรก โดยใช้คำสั่งต่อไปนี้หาค่าขีดจำกัดบนและขีดจำกัดล่าง

```
» xlimits =get(ax1,'XLim');
» ylimits =get(ax1,'YLim');
```

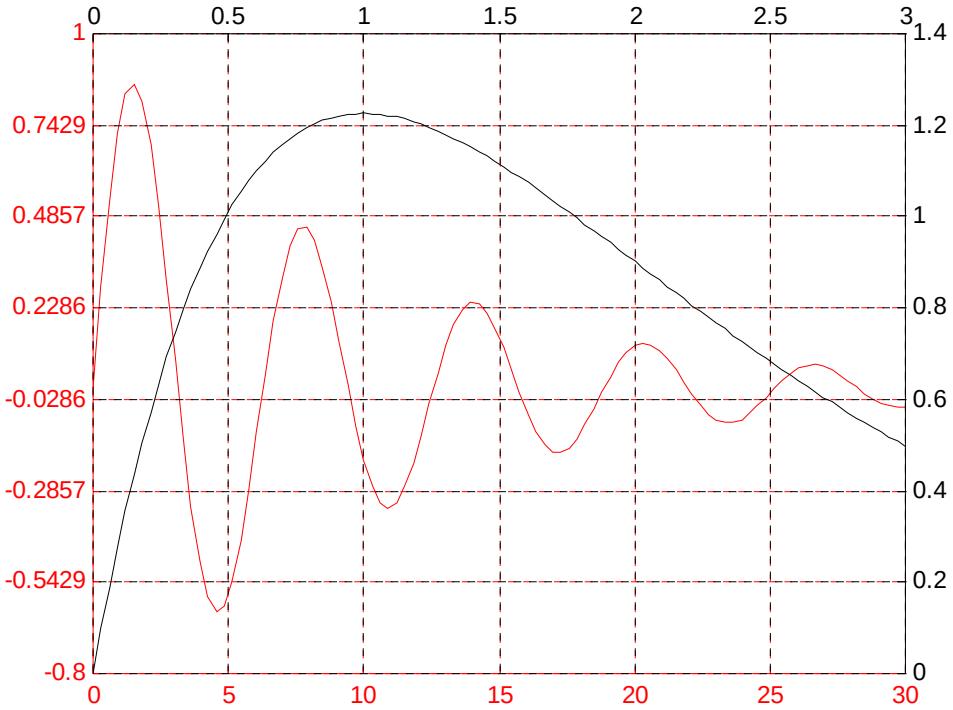
จากนั้นคำนวณหาระยะตามแนวแกน x และแกน y โดยใช้คำสั่ง

```
» xinc =(xlimits(2)-xlimits(1))/6;
» yinc =(ylimits(2)-ylimits(1))/7;
```

จากนั้นจึงกำหนดตำแหน่งของ tick mark โดยใช้คำสั่ง

```
» set(ax1,'XTick',[xlimits(1):xinc:xlimits(2)],...
'YTick',[ylimits(1):yinc:ylimits(2)])
```

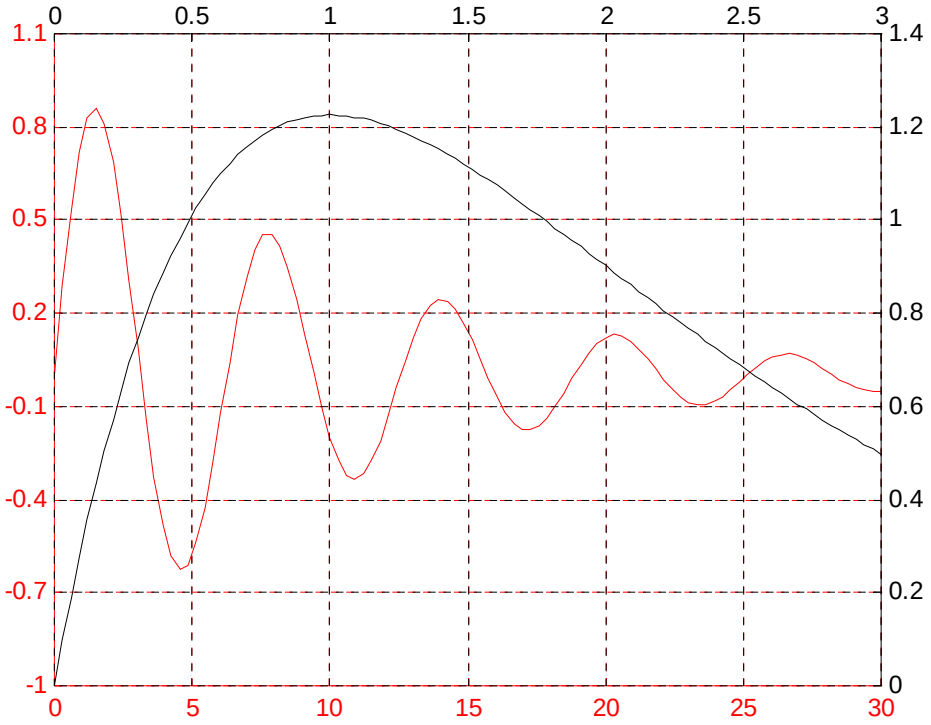
ผลที่ได้จะทำให้เราได้กราฟที่สามารถอ่านค่าได้ง่ายขึ้น แม้ว่าการแบ่งค่าตามแกน y ของกราฟรูปแรกจะอ่านค่าได้ลำบากไปบ้างเล็กน้อยเพราะตัวเลขเป็นเลขเป็นเลขทศนิยม



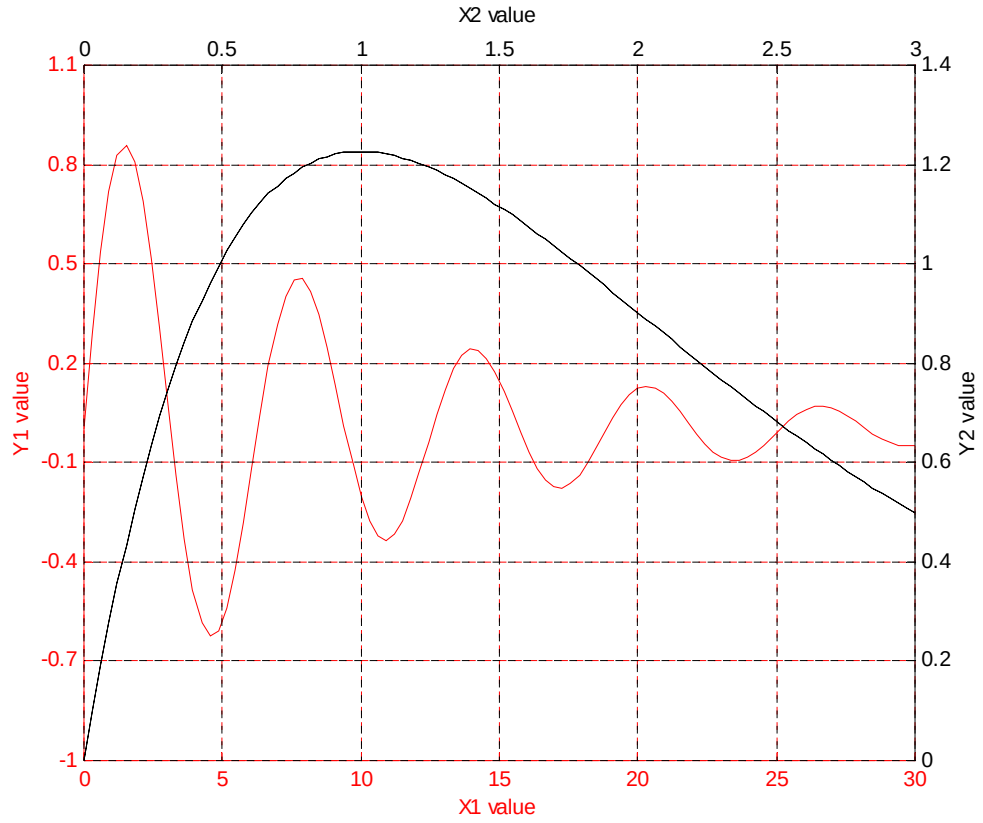
ถ้าหากเราต้องการให้ค่า trick mark ตามแกน y ของระบบแกนแรกมีค่าลงตัวเรา จำเป็นต้องกำหนดค่า limit ตามแกนy ขึ้นมาใหม่ให้หารด้วย 7 ลงตัวเนื่องจากตามแกน y มี 8 trick mark เราจึงต้องแบ่งช่วงออกเป็น 7 ช่วง ดังนั้น สมมุติว่าเรากำหนดช่วงเป็นจาก –1 ถึง 1.1 ซึ่งจะได้ความยาวช่วงเท่ากับ 2.1 และจะหารด้วย 7 ได้ความยาวช่วงละ 0.3 หน่วย โดยเราใช้คำสั่งต่อไปนี้

```
» ylimits =[-1.0 1.1];
» set(ax1,'YLim',ylimits);
» yinc = (ylimits(2)-ylimits(1))/7;
» set(ax1,'XTick',[xlimits(1):xinc:xlimits(2)],...
'YTick',[ylimits(1):yinc:ylimits(2)])
```

เราจะได้กราฟในรูปต่อไปนี้ ซึ่งจะสังเกตเห็นว่าเราสามารถอ่านค่าได้ง่ายขึ้น



ขั้นสุดท้ายเป็นการกำหนดชื่อแกนลงในแต่ละระบบแกน อย่างไรก็ตามเราขอทิ้งรายละเอียดให้คุณลองไปคิดดูนะครับว่าคุณจะใส่ชื่อของแกนแต่ละแกนลงไปได้อย่างไร



เราหวังเป็นอย่างยิ่งว่าตัวอย่างนี้จะช่วยทำให้คุณได้เข้าใจถึงการกำหนดค่าคุณสมบัติของแกนได้ดียิ่งขึ้น และสามารถปรับแก้ค่าคุณสมบัติต่างๆ ให้มีค่าตามที่คุณต้องการได้

13.3 Uicontrol

จุดมุ่งหมาย สร้าง user interface control object

รูปแบบ

```
handle =uicontrol(parent)
handle =uicontrol(…,'PropertyName',PropertyValue,…)
```

คำอธิบาย

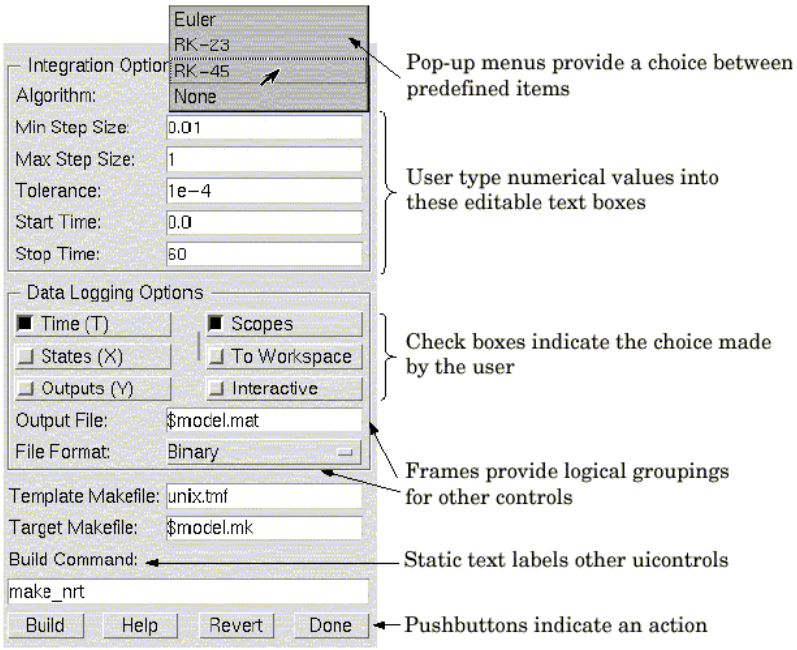
คำสั่ง **uicontrol** เป็นคำสั่งที่ใช้สร้างวัตถุประเภท uicontrol (user interface controls) เราสามารถที่จะสร้าง GUI ได้โดยการสร้าง uicontrols แบบต่างๆขึ้นมา โดยทั่วไปแล้วเมื่อผู้ใช้ได้เลือก **uicontrol** ไม่ว่าจะใช้เมาส์หรือแป้นพิมพ์ วัตถุนั้นก็จะทำงานตามที่ผู้เขียนโปรแกรมได้กำหนดขึ้น ใน MATLAB จะมี uicontrol อยู่มากมายหลายแบบ ซึ่งแต่ละแบบจะมีจุดมุ่งหมายที่จะให้ผู้ใช้ได้เลือกใช้ต่างๆ กัน uicontrol ที่มีใช้ใน MATLAB จะมีดังต่อไปนี้

1. **Check boxes**
2. **Editable text**
3. **Frames**
4. **List boxes**
5. **Pop-up menus**
6. **Push buttons**

7. **Radio buttons**
8. **Sliders**
9. **Static text**
10. **Toggle buttons**

โดยวัตถุที่ได้รับการสร้างขึ้นมาแต่ละแบบอาจมีคุณสมบัติบางประการแตกต่างกันออกไปบ้าง แต่โดยส่วนใหญ่แล้วคุณสมบัติของอุปกรณ์เหล่านี้จะคล้ายๆ กัน วัตถุเหล่านี้นับว่าเป็นวัตถุที่สำคัญมากในการสร้าง GUI เพราะเป็นวัตถุที่เราใช้ติดต่อกับผู้ใช้ เพื่อให้ผู้ใช้ใส่ค่าที่ต้องการ หรือเลือกที่จะให้โปรแกรมทำตามขั้นตอนใดๆตามที่ผู้ใช้ต้องการผ่านทางวัตถุเหล่านี้

สำหรับตัวอย่างบางประเภทของ uicontrol ทั้งหมดที่เรากล่าวถึงไว้ข้างบนนี้ได้แสดงเป็นตัวอย่างไว้ในรูปต่อไปนี้



และสำหรับรายละเอียดของ uicontrol แต่ละแบบมีดังต่อไปนี้

1. **Check boxes**

   จะมีการกำหนดค่าให้มีการทำงานเมื่อเราเลือกให้ค่าในกล่อง อุปกรณ์นี้มีประโยชน์เพื่อให้ผู้ใช้เลือกหัวข้อการทำงานของโปรแกรมแบบต่างๆได้อย่างอิสระ การที่จะให้ check box ทำงานเราจะใช้เมาส์กดไปที่บริเวณกล่องนั้น สภาวะการเลือกหรือไม่เลือกค่าตามที่กำหนดจะแสดงขึ้น

2. **Editable text boxes**

เป็น fields ที่ผู้ใช้สามารถที่จะแก้ไขตัวอักษรที่บรรจุอยู่ภายในกล่องนั้นได้ เราจะใช้ editable text เมื่อเราต้องการให้ผู้ใช้กำหนดค่าตัวอักษรเป็น input โดยในระบบปฏิบัติการ Microsoft Windows ถ้า editable text box เป็น focus แล้วเรากดเมาส์บน menu bar จะไม่ทำให้ callback ของ editable text ทำงาน ซึ่งจะตรงข้ามกับระบบปฏิบัติการUNIX ดังนั้นสำหรับระบบปฏิบัติการ Microsoft Windows เมื่อเราเปลี่ยนแปลงค่าในeditable text box แล้วใช้เมาส์กดบริเวณmenu bar ส่วนการใช้คำสั่ง **get(edit_handle,'String')** MATLAB จะไม่ให้ค่าของอักษรที่บรรจุอยู่ใน edit box เพราะ MATLAB จะต้องมีการสั่งให้ callback ทำงานเพื่อที่จะเปลี่ยนแปลงค่าคุณสมบัติ String แม้ว่าตัวอักษรที่ปรากฏอยู่บนจอภาพจะเปลี่ยนแปลงไปแล้วก็ตาม พฤติกรรมนี้เกิดขึ้นเนื่องจากความแตกต่างและข้อกำหนดของแต่ละระบบปฏิบัติการ

3. **Frames**

เป็นรูปแบบของพื้นที่สี่เหลี่ยม ซึ่งจะแบ่งหน้าต่างรูปภาพออกเป็นส่วนๆ Frames หรือกรอบจะทำให้ผู้ใช้สามารถแบ่งส่วนต่างๆของหน้าต่างออกได้ง่ายขึ้น ทำให้ผู้ใช้ไม่เกิดความสับสนในกรณีที่มี uicontrol ที่ต้องกำหนดค่าหลายๆ อันบนหน้าต่างเดียว เพราะเราสามารถนำ uicontrol ที่มีหน้าที่การทำงานในส่วนเดียวกันรวมกลุ่มไว้ด้วยกัน Frames จะไม่มี callback และจะมีเฉพาะ uicontrols เท่านั้นที่สามารถบรรจุอยู่ใน frames ได้ โดยทั่วไป Frames จะทึบแสง ไม่ใช่โปร่งใสดังนั้นในการเลือกใช้ frame กับ uicontrols เราจำเป็นต้องเรียงลำดับวัตถุเหล่านั้นให้ถูกต้อง และต้องพิจารณาว่า uicontrols นั้นบรรจุอยู่ใน frame หรือถูก frame ทับอยู่บางส่วน การเลือกใช้คำสั่ง Stacking order จะเป็นการคำนวณว่าลำดับการวาด uicontrols นั้นเป็นเช่นไร สำหรับวัตถุที่ได้รับการกำหนดขึ้นมาก่อนจะวางลงไปก่อน ส่วนวัตถุที่สร้างขึ้นที่หลังจะวางทับวัตถุที่สร้างขึ้นก่อนหน้านั้น ถ้าหากเราต้องการให้ frame บรรจุวัตถุ เราจะต้องสร้าง frame ขึ้นมาก่อนแล้วจึงวางวัตถุลงไป

4. **List boxes**

เป็นการแสดงรายการที่ผู้เขียนโปรแกรมกำหนดขึ้น ซึ่งจะกำหนดโดยคุณสมบัติ String และจะยอมให้ผู้ใช้สามารถที่จะเลือกรายการนั้นได้ที่ละหนึ่งรายการหรือมากกว่าได้ ค่าคุณสมบัติ Min และ Max จะเป็นตัวกำหนดจำนวนรายการที่เลือก ส่วนค่าคุณสมบัติ Value จะแสดงถึงหมายเลขลำดับของรายการที่ผู้ใช้เลือก ในกรณีที่ผู้ใช้เลือกรายการได้มากกว่าหนึ่งรายการค่าคุณสมบัติ Value จะเป็นเวกเตอร์ของหมายเลขลำดับรายการที่ผู้ใช้เลือก MATLAB จะสั่งให้คำสั่งตาม callback ของ list box ทำงานภายหลังจากได้มีการปล่อยปุ่มเมาส์ที่ทำให้เกิดการเปลี่ยนแปลงค่าของคุณสมบัติ Value ดังนั้นในกรณีที่เรากำหนดให้ผู้ใช้เลือกค่าได้หลายค่า

พร้อมกันเราอาจจำเป็นต้องเพิ่มปุ่มกด “Done” ลงใน GUI ของเราเพื่อให้การทำงานของ callback ช้าลงทำให้ผู้ใช้สามารถเลือกค่าที่ต้องการได้หมดเสียก่อน List boxes สามารถแยกความแตกต่างของการกดเมาส์หนึ่งครั้งและกดเมาส์ซ้อนกันสองครั้ง (double clicks) และสามารถที่จะเลือกกำหนดค่าคุณสมบัติ SelectionType ของวัตถุ figure ให้เป็น normal หรือ open ก่อนที่จะมีการกำหนดให้ callback ทำงาน

5. **Pop-up menus**

เป็นการเปิดรายการของตัวเลือกขึ้นหลังจากที่มีการกดเมาส์บริเวณเมนู เพื่อให้ผู้ใช้ได้เลือกรายการใดรายการหนึ่ง ซึ่งรายการของตัวเลือกจะกำหนดด้วยคุณสมบัติ String หากว่าเมนูนี้ไม่ได้เปิดขึ้น ค่าในเมนูจะแสดงค่าคุณสมบัติปัจจุบัน Pop-up menus นี้มีประโยชน์ในการที่จะให้ผู้ใช้ได้เลือกตัวเลือกตามที่ต้องการแต่ไม่ต้องการให้เปลืองเนื้อที่บนหน้าต่างรูปภาพ เราจำเป็นที่จะต้องกำหนดค่าคุณสมบัติ String

6. **Push buttons**

จะทำให้เกิดมีคำสั่งอื่นๆตามมาหลังจากที่ผู้ใช้เลือกกดปุ่มนี้ โดยทั่วไปผู้ใช้จะใช้เมาส์ในการกดปุ่ม Push buttons

7. **Radio buttons**

จะมีลักษณะการทำงานคล้ายกับ check boxes แต่ข้อแตกต่างสำคัญก็คือโดยทั่วไปแล้วการใช้ Radio button นี้เราจะจัดให้เป็นกลุ่มของตัวเลือก และในกลุ่มของ Radio button นั้นจะมีตัวเลือกที่สามารถถูกเลือกได้เพียงครั้งละตัวเดียวในกลุ่มนั้น ในการที่จะให้ radio button ทำงาน เราจะใช้เมาส์กดไปที่วัตถุนั้น จากนั้นสภาพของอุปกรณ์จะเปลี่ยนไปตามสภาพที่เราเลือก ผู้เขียนโปรแกรมจะต้องเป็นผู้กำหนดพฤติกรรมหลังจากที่มีการกดเมาส์ลงไปในวัตถุนี้

8. **Sliders**

เป็นการป้อนค่าตัวเลขโดยอาศัยแถบเลื่อนนี้ ค่าที่ป้อนจะถูกกำหนดโดยผู้เขียนโปรแกรมและผู้ใช้จะทำการกำหนดค่าโดยการใช้เมาส์กดแล้วเลื่อนแถบบน slider หรือใช้เมาส์กดบริเวณลูกศรเพื่อให้แถบค่อยๆเลื่อนไปเป็นลำดับ ตำแหน่งของตัวเลื่อนบนแท่งจะเป็นค่าตัวเลขตามสัดส่วนของระยะบนแท่งเลื่อนนี้ และค่าจะได้รับการคำนวณหลังจากที่เราปล่อยปุ่มเมาส์ เราสามารถที่จะตั้งค่าสูงสุด ต่ำสุด และค่าปัจจุบันของ slider ได้

9. **Static text boxes**

หมายถึงตัวหนังสือที่ผู้ใช้ไม่สามารถที่จะแก้ไขได้ แต่เราซึ่งเป็นผู้เขียนโปรแกรมอาจแก้ไขได้ ดังนั้นโดยทั่วไปเราจึงใช้ในการเขียนตัวหนังสือเพื่อบอกถึงชื่อของส่วนต่างๆ หรือ

อาจเป็นการบอกค่าของอุปกรณ์ควบคุมบางอย่าง หรือบอกถึงสภาวะการทำงานของโปรแกรม รวมถึงการบอกวิธีการใช้ GUI ของเราอย่างคร่าวๆ เป็นต้น และเมื่อผู้ใช้ไม่สามารถเปลี่ยนตัวอักษรใน static text ได้อย่าง interactive ดังนั้นเราจึงไม่สามารถที่จะเรียก callback ของ static text มาใช้ได้

10. **Toggle buttons**

เป็นปุ่มที่ทำหน้าที่เหมือนสวิชส์ไฟนั้นคือจะมีค่าเป็น on หรือ off และเมื่อผู้ใช้เปลี่ยนค่ามันโดยการกดเมาส์ลงไปในบริเวณของวัตถุนี้ จะเป็นการเรียก callback ให้ทำงานไปพร้อมกัน Toggle buttons มีประโยชน์ในการสร้าง toolbars

หมายเหตุ

ฟังก์ชัน **`uicontrol`** จะยอมรับชื่อคุณสมบัติและค่าของมันเป็นคู่ลำดับ หรือเป็นstructures และ cell arrays ที่เป็น arguments และจะมีตัวเลือกที่จะให้ handle ของวัตถุที่ถูกสร้างขึ้นและเราสามารถที่จะปรับเปลี่ยนหรือเลือกค่าเหล่านั้นได้ด้วยการใช้คำสั่ง **`set`** และ **`get`**

วัตถุประเภท Uicontrol จะเป็น children ของ figures ดังนั้นจึงไม่จำเป็นที่จะต้องสร้าง axes เพื่อที่จะวางวัตถุประเภทนี้

ตารางต่อไปนี้แสดงรายการคุณสมบัติต่างๆ ที่สำคัญของ uicontrol objects อย่าลืมว่าการตั้งชื่อคุณสมบัติเหล่านี้มักจะบอกถึงคุณลักษณะของคุณสมบัตินั้นๆด้วย

Property Name	Property Description	Property Value
Controlling Style and Appearance		
BackgroundColor	สี background ของวัตถุ	**ค่า**: ColorSpec **ค่าเริ่ม**: ขึ้นกับระบบปฏิบัติการ
Cdata	Truecolor image ที่แสดงบน control	**ค่า**: matrix
FegroundColoror	สีของตัวหนังสือ	**ค่า**: ColorSpec **ค่าเริ่ม**: [0 0 0]
SelectionHighlight	กำหนดให้วัตถุนั้นต้องเน้นสีเมื่อถูกเลือกหรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
String	ตัวอักษรที่แสดงบน Uicontrol และจะเป็นรายการที่แสดงสำหรับ list box และ pop-up menu	**ค่า**: string
Visible	กำหนดให้ผู้ใช้สามารถมองเห็น Uicontrol นี้ได้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
General Information About the Object		

Property Name	Property Description	Property Value
Children	Uicontrol objects ทุกแบบจะไม่มี children	
Enable	ให้ uicontrol นี้ใช้งานได้หรือไม่ (Enable หรือ disable)	**ค่า**: on, inactive, off **ค่าเริ่ม**: on
Parent	parent ของ Uicontrol	**ค่า**: scalar figure handle
Selected	แสดงให้ทราบว่าขณะนี้ว่าวัตถุนี้ถูกเลือกจากผู้ใช้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: off
SliderStep	ขนาดของการเลื่อนของ slider	**ค่า**: two-element vector **ค่าเริ่ม**: [0.01 0.1]
Style	ชนิดของ uicontrol object	**ค่า**: pushbutton, togglebutton, radiobutton, checkbox, edit, text, slider, frame, listbox, popupmenu **ค่าเริ่ม**: pushbutton
Tag	การกำหนดชื่อของวัตถุโดยผู้ใช้	**ค่า**: string
TooltipString	tooltip ที่จะแสดงเมื่อเราลากเมาส์ไปวางบริเวณ object	**ค่า**: string
Type	ชนิดของ graphics object	**ค่า**: string (read-only) **ค่าเริ่ม**: uicontrol
UserData	User-specified data (ข้อมูลที่กำหนดโดยผู้ใช้)	**ค่า**: matrix **ค่าเริ่ม**: [ ] เมทริกซ์ว่าง
Controlling the Object Position		
Position	ขนาดและตำแหน่งของวัตถุ uicontrol	**ค่า**: position rectangle **ค่าเริ่ม**: [20 20 60 20]
Units	หน่วยที่ใช้ในการแปรความถึงตำแหน่งและขนาด	**ค่า**: pixels, normalized, inches, centimeters, points, characters **ค่าเริ่ม**: pixels
Controlling Fonts and Labels		
FontAngle	มุมเอียงของตัวอักษร	ค่า: normal, italic, oblique ค่าเริ่ม: normal
FontName	Font family (ชนิดของตัวอักษร)	ค่า: string ค่าเริ่ม: ขึ้นกับระบบปฏิบัติการ
FontSize	ขนาดตัวอักษร	ค่า: size in FontUnits

Property Name	Property Description	Property Value
		ค่าเริ่ม: ขึ้นกับระบบปฏิบัติการ
FontUnits	หน่วยของขนาดตัวอักษร	**ค่า**: points, normalized, inches, centimeters, pixels **ค่าเริ่ม**: points
FontWeight	น้ำหนักของตัวอักษร	**ค่า**: light, normal, demi, bold **ค่าเริ่ม**: normal
HorizontalAlignment VerticalAlignment	ลักษณะการจัดวางตัวอักษร ในแนวระดับ และแนวดิ่ง	**ค่า**: left, center, right **ค่าเริ่ม**: ขึ้นกับ uicontrol object
String	ตัวอักษรที่แสดงบน Uicontrol และจะเป็นรายการที่แสดงสำหรับ list box และ pop-up menu	**ค่า**: string
Controlling Callback Routine Execution		
BusyAction	การขัดจังหวะของ Callback	**ค่า**: cancel, queue **ค่าเริ่ม**: queue
ButtonDownFcn	callback เมื่อมีการกดเมาส์ปุ่มซ้ายบริเวณวัตถุนั้น	**ค่า**: string
Callback	การควบคุมการทำงานหลังจากเลือกวัตถุนั้น	**ค่า**: string
CreateFcn	Callback ที่ทำงานในระหว่างการสร้างวัตถุ	**ค่า**: string
DeleteFcn	Callback ที่ทำงานในระหว่างการลบวัตถุ	**ค่า**: string
Interruptible	Mode ของการขัดจังหวะการทำงานของ Callback	**ค่า**: on, off **ค่าเริ่ม**: on
UIContextMenu	Uicontextmenu ที่เกี่ยวข้องกับวัตถุนี้	**ค่า**: handle
Information About the Current State		
ListboxTop	แหน่งที่แสดงตัวหนังสือสูงสุดบน List box	**ค่า**: scalar **ค่าเริ่ม**: [1]
Max	ค่าสูงสุด จะขึ้นอยู่กับชนิดของวัตถุ uicontrol	**ค่า**: scalar **ค่าเริ่ม**: ขึ้นกับชนิดของวัตถุ
Min	ค่าต่ำสุด จะขึ้นอยู่กับชนิดของวัตถุ uicontrol	**ค่า**: scalar **ค่าเริ่ม**: ขึ้นกับชนิดของวัตถุ
Value	ค่าขณะนั้นของ uicontrol	**ค่า**: scalar or vector **ค่าเริ่ม**: ขึ้นกับชนิดของวัตถุ
Controlling Access to Objects		

Property Name	Property Description	Property Value
HandleVisibility	กำหนดว่า handle ของวัตถุนี้จะเห็นได้ จากคำสั่งแบบcommand line และ GUIs หรือไม่	**ค่า**: on, callback, off **ค่าเริ่ม**: on
HitTest	สามารถจะเลือกได้โดยเมาส์หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on

ตัวอย่าง

ตัวอย่างต่อไปนี้จะสร้าง push button ซึ่งจะลบแกนปัจจุบันออกเมื่อเรากดปุ่มนั้น

```
h =uicontrol('Style', 'pushbutton', 'String', 'Clear',…
'Position', [20 150 100 70], 'Callback', 'cla');
```

เราสามารถที่จะสร้าง uicontrol ที่เปลี่ยน colormaps ของ figure ด้วยการกำหนดค่าแบบสีต่างๆลงใน pop-up menu และใช้ M-file ที่มีเป็นชื่อที่ใช้เรียกการทำงานใน Callback

```
» hpop = uicontrol('Style', 'popup',...
'String', 'hsv|hot|cool|gray',...
'Position', [20 320 100 50],...
'Callback', 'setmap');
```

สำหรับ uicontrol ที่เรียกมาใช้ตามคำสั่งข้างบนนี้จะมีการเลือกตัวเลือกได้สี่ตัวเลือก ที่จะปรากฏในเมนู นั่นคือ **hsv, hot**, **cool** และ **gray** เรากำหนดค่ารายการตัวเลือกนี้ได้โดยใช้ค่าคุณสมบัติ String และแยกตัวเลือกแต่ละรายการออกจากกันด้วย ตัวอักษร “ **|** ”

สำหรับ Callback ในกรณีนี้จะเป็นการเรียก M-file ที่ชื่อ setmap มาใช้งาน สำหรับในกรณีนี้เราเลือกที่จะเรียก M-file ที่เขียนไว้มาใช้งานเพราะว่าขั้นตอนการทำงานของ callback ค่อนข้างจะยาวและยุ่งยากมากกว่าที่จะใส่เป็นคำสั่งคำสั่งเดียวได้ ดังนั้นเพื่อความสะดวกและง่ายต่อการแก้ไขเราจะเขียนชุดคำสั่งเป็น M-file เพื่อให้คำสั่งใน callback เป็นคำสั่งเพียงคำสั่งเดียวคือ setmap แต่ขั้นตอนการทำงานเราจะแยกออกไปต่างหากใน M-file ที่เขียนขึ้นไว้ก่อนแล้ว สำหรับตัวอย่างนี้ M-file ชื่อ setmap จะมีคำสั่งดังต่อไปนี้

```
val =get(hpop,'Value');
if val ==1
colormap(hsv)
elseif val ==2
colormap(hot)
elseif val ==3
colormap(cool)
elseif val ==4
colormap(gray)
end
```

ค่าคุณสมบัติ Value จะมีสี่ค่าจาก 1 ถึง 4 ตามลำดับของรายการที่ผู้ใช้เลือก เช่นถ้าผู้ใช้เลือก **hsv** ค่าคุณสมบัติ value จะมีค่าเท่ากับ 1 เป็นต้น

อย่างไรก็ตามเราจะไม่กล่าวถึงการสร้าง uicontrol ด้วยวิธีการใช้ command line ตามตัวอย่างที่แสดงข้างบนนี้ในรายละเอียดเพราะในเอกสารนี้เราจะกล่าวถึงการสร้าง GUI ด้วยสภาพแวดล้อมอีกแบบหนึ่ง ซึ่งจะง่ายกว่าการใช้ command line มาก อย่างไรก็ตามที่ยกตัวอย่างข้างบนนี้ก็เพื่อให้เราได้เข้าใจถึงการกำหนดค่าคุณสมบัติต่างๆได้ดีขึ้นนั่นเอง

13.4 Uimenu

จุดประสงค์ สร้างเมนูบนหน้าต่างรูปภาพ

รูปแบบ

```
uimenu('PropertyName', PropertyValue,...)
uimenu(parent, 'PropertyName', PropertyValue,...)
handle = uimenu('PropertyName', PropertyValue,...)
handle = uimenu(parent,'PropertyName',PropertyValue,...)
```

คำอธิบาย

คำสั่ง **uimenu** จะเป็นการสร้างแถบคำสั่งหรือแถบเมนู รวมถึงเมนูย่อยที่อยู่ในหน้าต่างรูปภาพ นอกเหนือจากนั้นเรายังสามารถใช้คำสั่ง **uimenu** เพื่อสร้างเมนูสำหรับ context menu ได้อีกด้วย คำสั่ง

```
Handle =uimenu('PropertyName',PropertyValue,…)
```

จะสร้างเมนูในแถบเมนูของหน้าต่างรูปภาพปัจจุบัน โดยใช้ค่าคุณสมบัติที่กำหนด และให้ handle ของเมนูนั้นกลับมา ส่วนคำสั่ง

```
handle =uimenu(parent,'PropertyName',PropertyValue,...)
```

จะสร้างเมนูย่อย (submenu) สำหรับเมนูหลักที่กำหนด handle ด้วยค่า parent แต่อย่างไรก็ตามหากว่าถ้า parent เป็น handle ของ figure แทนที่จะเป็น uimenu object หรือ Uicontextmenu แล้ว MATLAB จะสร้างแถบเมนูใหม่ขึ้นบนรูปภาพที่กำหนด

หมายเหตุ

MATLAB จะเพิ่มเมนูเข้าไปต่อกับเมนูเดิมที่มีอยู่ และเมนูแต่ละเมนูสามารถที่จะสร้างเมนูย่อยเพิ่มเป็นลำดับขั้นต่อไปได้เรื่อยๆ โดยเมื่อเราเลือกเมนูนั้นแล้วก็จะแสดงเมนูย่อยของเมนูนั้นต่อๆไป

ในการกำหนดค่าคุณสมบัติให้กับ uimenu ก็จะเหมือนกับวัตถุอื่นๆ คือจะกำหนดเป็นคู่ของ property name และ property value ก็ได้ หรือเราอาจกำหนดเป็นตัวแปรแบบ structures และ cell arrays ของคุณสมบัติให้เป็น input arguments สำหรับคุณสมบัติ Callback ของ uimenu จะเป็นคุณสมบัติที่กำหนดว่าต้องการที่จะให้ MATLAB ทำงานอะไรภายหลังจากที่ผู้ใช้ได้เลือกเมนูนั้นๆ Uimenus จะปรากฏอยู่บนแถบเมนูในหน้าต่างรูปก็ต่อเมื่อคุณสมบัติ WindowStyle เลือกค่าเป็น normal ถ้าหากว่า figure object ที่บรรจุ uimenu ได้เปลี่ยนคุณสมบัติ WindowStyle เป็น modal เราจะไม่เห็นแถบเมนูปรากฏให้เห็นในหน้าต่างรูป

แม้ว่าค่าที่กำหนดให้จะมีอยู่ก็ตาม ส่วนค่าคุณสมบัติ MenuBar ของวัตถุ figure จะมีผลกระทบต่อตำแหน่งและจำนวนของรายการที่ปรากฏของแถบเมนู ในหน้าต่างรูป เมื่อคุณสมบัติ MenuBar เลือกเป็น figure จะทำให้มีรายการในเมนูที่สร้างขึ้นใหม่ปรากฏขึ้นพร้อมกับรายการในเมนูที่ MATLAB สร้างเป็นพื้นฐานไว้ แต่ถ้าคุณสมบัติ MenuBar เลือกไว้ที่ none จะปรากฏเฉพาะเมนูที่เราสร้างขึ้นปรากฏอยู่ในแถบเมนู

คุณสมบัติต่อไปนี้เป็นคุณสมบัติที่มีประโยชน์ หากเราจะเลือกใช้หรือเปลี่ยนแปลงคุณสมบัติของ uimenu เราจะเรียงลำดับตามลักษณะการทำงานของมัน ชื่อของคุณสมบัติมักจะถ่ายถอดถึงหน้าที่ของคุณสมบัตินั้น

Property Name	Property Description	Property Value
Controlling Style and Appearance		
Checked	แสดงให้เห็นว่าเลือกเมนูนั้นหรือไม่	**ค่า**: on, off **ค่าเริ่ม**: off
ForegroundColor	สีของตัวอักษร	**ค่า**: ColorSpec **ค่าเริ่ม**: [0 0 0]
Label	ชื่อของ Menu	**ค่า**: string
SelectionHighlight	จะให้วัตถุเน้นสีเมื่อเลือกหรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
Separator	แบ่งบรรทัดหรือไม่ Separator line mode	**ค่า**: on, off **ค่าเริ่ม**: off
Visible	กำหนดว่า Uimenu นี้มองเห็นได้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
General Information About the Object		
Accelerator	Keyboard equivalent	**ค่า**: character
Children	Handles ของเมนูย่อย	**ค่า**: vector of handles
Enable	กำหนดให้ uimenu นี้ทำงานได้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
Parent	แสดง handle ของ parent ของ Uimenu นี้	**ค่า**: handle
Tag	การกำหนดชื่อของวัตถุโดยผู้ใช้	**ค่า**: string
Type	ชนิดของ graphics object	**ค่า**: string (read-only) **ค่าเริ่ม**: uimenu
UserData	ข้อมูลที่ผู้ใช้กำหนด	**ค่า**: matrix
Controlling the Object Position		
Position	ตำแหน่งสัมพัทธ์ของ uimenu	**ค่า**: scalar **ค่าเริ่ม**: [1]
Controlling Callback Routine Execution		

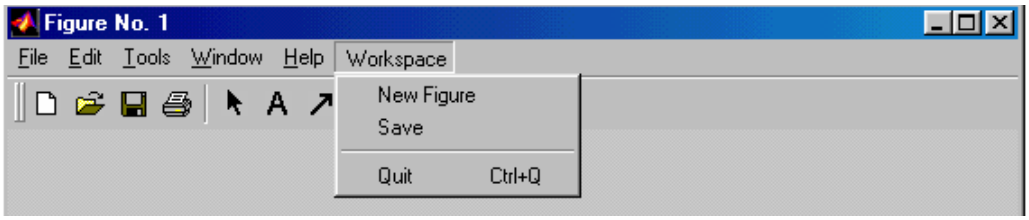
Property Name	Property Description	Property Value
BusyAction	การกำหนดว่า Callback นี้เมื่อมีการเรียกซ้อนจะทำอย่างไร	**ค่า**: cancel, queue **ค่าเริ่ม**: queue
ButtonDownFcn	callback ในกรณีที่มีการกดเมาส์บนวัตถุนั้น	**ค่า**: string
Callback	คำสั่งควบคุมการทำงาน เมื่อเลือกวัตถุนั้น	**ค่า**: string
CreateFcn	Callback ที่ทำงานเฉพาะในขณะที่กำลังสร้างวัตถุนั้น	**ค่า**: string
DeleteFcn	Callback ที่ทำงานเฉพาะในขณะที่กำลังลบวัตถุนั้น	**ค่า**: string
Interruptible	กำหนดว่า Callback นี้ขัดจังหวะได้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
Controlling Access to Objects		
HandleVisibility	กำหนดว่า handle นี้จะสามารถมองเห็นได้จากการใช้คำสั่งที่จะมองหาวัตถุนี้ ไม่ว่าจะจากcommand line หรือ GUIs	**ค่า**: on, callback, off **ค่าเริ่ม**: on
HitTest	กำหนดว่าสามารถที่จะเลือกวัตถุได้โดยการกดเมาส์หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on

✪ ตัวอย่าง

ตัวอย่างต่อไปนี้จะสร้างเมนู ที่ชื่อ Workspace ซึ่งจะให้ผู้ใช้สามารถที่จะ สร้าง figure window ใหม่ บันทึก (save) ตัวแปรที่อยู่ใน workspace และออกจากการทำงานของ MATLAB นอกเหนือจากนั้นยังนิยามการกดปุ่ม Ctrl พร้อมกับปุ่ม Q เพื่อให้มีผลเหมือนคำสั่ง Quit ซึ่งการใช้ปุ่มบนแป้นพิมพ์แบบนี้นิยมเรียกกันว่า accelerator key

```
» f =uimenu('Label','Workspace');
» uimenu (f,'Label','New Figure','Callback','figure');
» uimenu (f,'Label','Save','Callback','save');
» uimenu (f,'Label','Quit','Callback','exit',...
» 'Separator','on','Accelerator','Q');
```

ผลที่ได้จะเป็นแถบเมนูที่มีลักษณะดังรูปต่อไปนี้



13.5 Uicontextmenu

จุดประสงค์ เพื่อสร้าง context menu

รูปแบบ `handle = uicontextmenu('PropertyName',PropertyValue,...);`

คำอธิบาย

คำสั่ง **uicontextmenu** จะสร้าง context menu ซึ่งจะเป็นเมนูที่ปรากฏขึ้นเมื่อผู้ใช้กดเมาส์ปุ่มขวาในบริเวณวัตถุนั้น เราจะสร้างรายการใน context menu โดยการใช้ฟังก์ชัน uimenu เพื่อสร้างรายการที่จะให้ปรากฏเมื่อกดเมาส์ปุ่มขวาขึ้นบนวัตถุนั้น เราจะสามารถเชื่อมโยงวัตถุกับเมนูได้โดยใช้คุณสมบัติ UIContextMenu ของวัตถุนั้น และกำหนด handle ของ context menu เป็นค่าคุณสมบัติ UIContextMenu นั่นเอง

คุณสมบัติที่มีอยู่ในตารางนี้เป็นคุณสมบัติที่ใช้อยู่เป็นประจำของ uicontextmenu objects เราจะเรียงลำดับตามลักษณะการทำงานของมัน ชื่อของคุณสมบัติมักจะถ่ายทอดถึงหน้าที่ของคุณสมบัตินั้น

Property Name	Property Description	Property Value
Controlling Style and Appearance		
Visible	กำหนดว่า Uicontextmenu นี้มองเห็นได้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: off
Position	ตำแหน่งของ uicontextmenu เมื่อกำหนดให้ Visible เป็น on	**ค่า**: two-element vector **ค่าเริ่ม**: [0 0]
General Information About the Object		
Children	ให้ handle ของ uimenus ที่สร้าง uicontextmenu	**ค่า**: matrix
Parent	parent ของ Uicontextmenu	**ค่า**: scalar figure handle
Tag	การกำหนดชื่อของวัตถุโดยผู้ใช้	**ค่า**: string
Type	ชนิดของ graphics object	**ค่า**: string (read-only) **ค่าเริ่ม**: uicontrol
UserData	ข้อมูลที่ผู้ใช้กำหนด	**ค่า**: matrix
Controlling Callback Routine Execution		
BusyAction	การกำหนดว่า Callback นี้เมื่อมีการเรียกซ้อนจะทำอย่างไร	**ค่า**: cancel, queue **ค่าเริ่ม**: queue

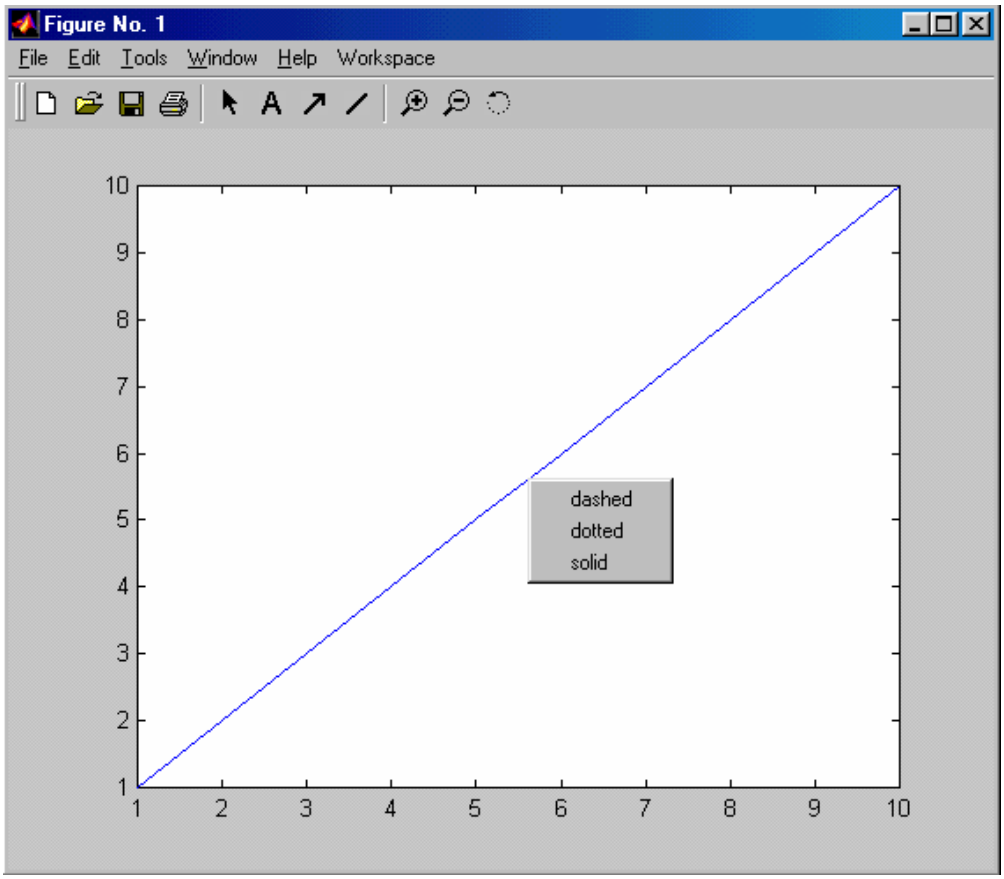
Property Name	Property Description	Property Value
Callback	คำสั่งควบคุมการทำงาน เมื่อเลือกวัตถุนั้น	**ค่า**: string
CreateFcn	Callback ที่ทำงานเฉพาะในขณะที่กำลังสร้างวัตถุนั้น	**ค่า**: string
DeleteFcn	Callback ที่ทำงานเฉพาะในขณะที่กำลังลบวัตถุนั้น	**ค่า**: string
Interruptible	กำหนดว่า Callback นี้ขัดจังหวะได้หรือไม่	**ค่า**: on, off **ค่าเริ่ม**: on
Controlling Access to Objects		
HandleVisibility	กำหนดว่า handle นี้จะสามารถมองเห็นได้จากการใช้คำสั่งที่จะมองหาวัตถุนี้ ไม่ว่าจะจากcommand line หรือ GUIs	**ค่า**: on, callback, off **ค่าเริ่ม**: on

✪ **ตัวอย่าง**

ตัวอย่างต่อไปนี้จะเป็นการสร้าง context menu เพื่อจะใช้กับวัตถุ line ซึ่งเขียนขึ้นเป็นกราฟรูปหนึ่ง เมื่อผู้ใช้กดเมาส์ปุ่มขวาไม่ว่าจะเป็นที่ใดบนเส้นนี้ จะปรากฏเมนูขึ้นและเมนูนี้จะยอมให้ผู้ใช้สามารถเลือกแบบของเส้นให้เป็นแบบต่างๆตามต้องการได้ ลักษณะของ M-file จะเป็นดังนี้

```
% สร้าง context menu
cmenu =uicontextmenu;
% สร้างเส้นเพื่อที่จะใช้กับ context menu
hline =plot(1:10,'UIContextMenu',cmenu);
% สร้าง callbacks สำหรับ context menu แต่ละรายการ
%ในที่นี้สมมุตว่ามี 3 รายการ
cb1= ['set(hline, ''LineStyle'', ''--'')'];
cb2 = ['set(hline, ''LineStyle'', '':'')'];
cb3 = ['set(hline, ''LineStyle'', ''-'')'];
% สร้างรายการใน context menu
item1 = uimenu(cmenu, 'Label', 'dashed', 'Callback', cb1);
item2 = uimenu(cmenu, 'Label', 'dotted', 'Callback', cb2);
item3 = uimenu(cmenu, 'Label', 'solid', 'Callback', cb3);
```

ซึ่งเมื่อเราให้ M-file นี้ทำงานเราจะได้กราฟดังรูป และเมื่อเราเอาเมาส์ปุ่มขวากดบริเวณเส้นกราฟจะเกิดเมนูขึ้น ดังที่แสดงในภาพต่อไปนี้



เราสามารถที่จะสร้าง context menu ให้กับวัตถุทุกชนิดบน MATLAB ได้เสมอ สำหรับในตัวอย่างนี้เราได้สร้างขึ้นกับเส้นหรือ line object ซึ่งการใช้ context menu นี้จะช่วยให้ผู้ใช้ GUI ได้รับความสะดวกขึ้นมาก

สำหรับคุณสมบัติของวัตถุที่เหลือ ซึ่งจะเป็นวัตถุที่เป็น children ของ Axes เราจะไม่ขอกล่าวถึงในรายละเอียดในที่นี้ แต่อย่างไรก็ตามเราได้แสดงรายการตารางของคุณสมบัติเหล่านั้นไว้ในภาคผนวกท้ายบท ด้วยเหตุผลสำคัญที่ว่าคุณสมบัติของวัตถุเหล่านั้นโดยมากจะเริ่มซ้ำกับคุณสมบัติที่เราได้กล่าวถึงไปก่อนหน้านี้แล้ว และถ้าหากเรากล่าวถึงวัตถุนั้นในรายเอียดก็จะเริ่มซ้ำกับวัตถุต่างๆ ที่กล่าวมาก่อนหน้านี้ อย่างไรก็ตามเราทราบดีอยู่แล้วว่าวัตถุทุกชนิดต้องมีคุณสมบัติพิเศษเป็นของตนเอง รายละเอียดในคุณสมบัตินั้นเราสามารถหาได้จากคู่มือที่เกี่ยวข้องของ MATLAB

บทที่ 14 ทำความรู้จักกับ GUIDE

Graphical User Interface (GUI) เป็น user interface ที่สร้างขึ้นด้วย graphical object แบบต่าง ๆ เช่น ปุ่ม เมนู slider โดยทั่วไป objects เหล่านี้ ผู้ใช้คอมพิวเตอร์ส่วนใหญ่เข้าใจถึงความหมายและวิธีการใช้ object เหล่านี้เป็นอย่างดีอยู่แล้ว สิ่งสำคัญที่เราจะกล่าวถึงคือ หลังจากผู้ใช้ได้มีการกดปุ่มเมาส์ เลื่อน slider หรือเลือกเมนู เราจะมีวิธีการกำหนดให้เกิดขั้นตอนต่อ ๆ ไป ตามที่เราต้องการได้อย่างไร

Application ต่าง ๆ ที่ออกแบบมาเป็น GUI จะพบว่าสามารถทำให้ผู้ใช้เข้าใจการใช้ application นั้นได้อย่างรวดเร็ว เพราะแทบจะไม่มีชุดคำสั่งใดให้จดจำ การทำงานของ application จะเกิดขึ้นทันทีที่ได้ input จากผู้ใช้

สำหรับบทนี้เราจะอธิบายรายละเอียดต่างๆ ที่เกี่ยวข้องกับ GUIDE ซึ่งเป็นเครื่องมือหลักที่เราจะใช้สร้าง GUI โดยในบทนี้จะเน้นในรายละเอียดว่า GUIDE นั้นมีเครื่องมืออะไรบ้าง และขั้นตอนในการทำงานของ GUIDE นั้นเป็นอย่างไร

ในบทนี้จะเป็นรายละเอียดค่อนข้างมาก โดยสำหรับตัวอย่างในการใช้ GUIDE ในการสร้าง GUI เราจะกล่าวถึงในบทต่อไป ดังนั้นสำหรับผู้ที่ต้องการเริ่มทดลองใช้งาน GUIDE เลยแล้วจะเรียนรู้จากตัวอย่างอาจข้ามบทนี้ไปก่อนก็ได้ และเมื่อใดที่มีข้อสงสัยในรายละเอียดบางเรื่องจึงกลับมาอ่านบทนี้เมื่อต้องการทราบรายละเอียดในการทำงาน

14.1 การสร้าง GUI ด้วย GUIDE

MATLAB จะสร้าง GUI อยู่บนหน้าต่างรูปภาพ (figure window) ซึ่งภายได้หน้าต่างนี้จะมีส่วนประกอบต่าง ๆ อยู่ได้ไม่ว่าจะเป็น axes, uicontrol หรือวัตถุอื่น ๆ ตามที่เราได้กล่าวถึงมาแล้วในบทก่อนหน้านี้ ใน MATLAB version ก่อนหน้านี้ เราสามารถที่จะสร้าง uicontrol , uimenu แบบต่าง ๆ ลงในหน้าต่างรูปภาพได้แต่เป็นไปด้วยความลำบากเพราะการสร้างเป็น text base ต่อมาร จนกระทั่ง version5 MATLAB ได้สร้าง Graphical User Interface Development Environment หรือ GUIDE ขึ้นเพื่อช่วยให้เราสร้างบันทึก และแก้ไป GUI ได้สะดวกขึ้น

การสร้าง GUI จะประกอบด้วยขั้นตอนสองขั้นตอน

- กำหนดและวางส่วนประกอบต่าง ๆ ลงบน GUI
- เขียนโปรแกรมเพื่อกำหนดการทำงานของส่วนประกอบต่าง ๆ ใน GUI

GUIDE นั้นโดยหลักใหญ่แล้วจะมีหน้าที่ในการวางส่วนประกอบที่เราต้องการให้มีลงใน GUI จากนั้น GUIDE จะสร้าง M-file ที่บรรจุ handle ของวัตถุหรือ object ทั้งหมดที่เราสร้างขึ้นรวมทั้งคำสั่งให้

GUI ทำงาน นอกเหนือจากนั้น M-file จะให้แนวทางในการเขียนฟังก์ชัน ที่ทำงานหลังจากผู้ใช้กดเมาส์ปุ่มซ้ายหรือปรับเปลี่ยนค่าของวัตถุนั้น ซึ่งเราเรียกว่า callback ของวัตถุนั้น

14.2 ส่วนประกอบของ GUI ใน MATLAB

ดังที่ได้กล่าวมาก่อนแล้วว่าเราสามารถสร้าง GUI ขึ้นมาได้โดยการเขียนเป็น M-file ขึ้นมาล้วน ๆ แต่การใช้ GUIDE จะทำให้การทำงานง่ายขึ้นมากเพราะจะช่วยให้เรากำหนดตำแหน่งของวัตถุต่าง ๆ ได้โดยง่าย หลังจากนั้น GUIDE จะสร้างไฟล์ขึ้งมา 2 ไฟล์เพื่อเก็บและนำ GUI ของเรามาใช้ต่อไปซึ่งจะประกอบด้วย

- FIG-file ซึ่งจะบรรจุรายละเอียดของวัตถุต่างที่เป็นองค์ประกอบอยู่ในหน้าต่างรูปภาพที่เป็น GUI ของเรา
- M-file ที่จะบรรจุฟังก์ชันที่กำหนดการทำงานของ GUI ของเรา รวมถึง callback ทั้งหมด ซึ่ง callback เหล่านี้จะบรรจุเป็น sub function อยู่ใน M-file และเราจะเรียก M-file ที่ควบคุมการทำงานของ GUI นี้ว่า Application M-file

ดังนั้น Application M-file จะไม่มีข้อมูลใด ๆ เกี่ยวกับรูปแบบของส่วนประกอบที่บรรจุอยู่ใน GUI เช่นสี ขนาด ตำแหน่ง หรือ อื่น ๆ เลย เพราะข้อมูลเหล่านั้นจะบรรจุอยู่ใน FIG-file

❶ ส่วนประกอบสำคัญของ Application M-file ที่สร้างโดย GUIDE

GUIDE จะรวบรวมองค์ประกอบต่าง ๆ ภายใน GUI แล้วสร้าง Application M-file โดยอัตโนมัติโดยมีรูปแบบของการสร้างที่ชัดเจน เพื่อให้เราได้โครงสร้างของ Application M-file จากนั้นเราสามารถนำโครงสร้างที่สร้างโดยอัตโนมัตินั้นมาปรับแก้ เพื่อให้เกิดการควบคุม GUI ตามที่เราต้องการ การกระทำดังกล่าวทำให้เราได้ข้อได้เปรียบหลายประการ เช่น

- M-file จะประกอบด้วยคำสั่งที่จำเป็นในการควบคุม GUI ครบถ้วน
- M-file จะทำให้เราส่งข้อมูลไปที่ส่วนต่าง ๆ ได้ง่าย สะดวก รวดเร็ว
- การใช้ M-file จะทำให้เราส่งข้อมูลไปที่ส่วนต่าง ๆ ภายใต้ MATLAB ได้ง่ายขึ้น
- Application M-file จะสร้าง Sub function สำหรับ uicontrols ทุกแบบที่มีใน GUI เพื่อทำให้เราเขียน callback ต่าง ๆ ได้สะดวกขึ้น

แม้ว่า GUIDE จะให้ทางเลือกกับเราว่าจะให้ GUIDE สร้างเฉพาะ fig-file เพื่อเก็บและใช้เป็นข้อมูลของ GUI ที่สร้างขึ้นเพียงอย่างเดียว แล้วเราเขียน M-file ขึ้นมาเอง แต่สำหรับผู้เริ่มเขียน GUI บน MATLAB เราคิดว่าการสร้าง GUI ด้วย GUIDE จะสะดวกกว่า หากเราให้ GUIDE สร้าง Application M-file ให้เราด้วย ดังนั้นในการสร้าง GUI ด้วย GUIDE ที่เรานำเสนอในเอกสารนี้จะมีการกำหนดขั้นตอนดังนี้

- เลือก GUIDE Application option แล้วเลือกให้ GUIDE สร้างทั้ง FIG-file และ M-file

- การใช้ Layout Editor เพื่อวางรูปแบบของ GUI
- เรียนรู้การสร้าง Application M-file จาก GUIDE และเข้าใจถึงวิธีการทำเพื่อจะนำไปใช้ต่อ
- ปรับแก้ Application M-file ให้ทำงานตามที่เรากำหนด

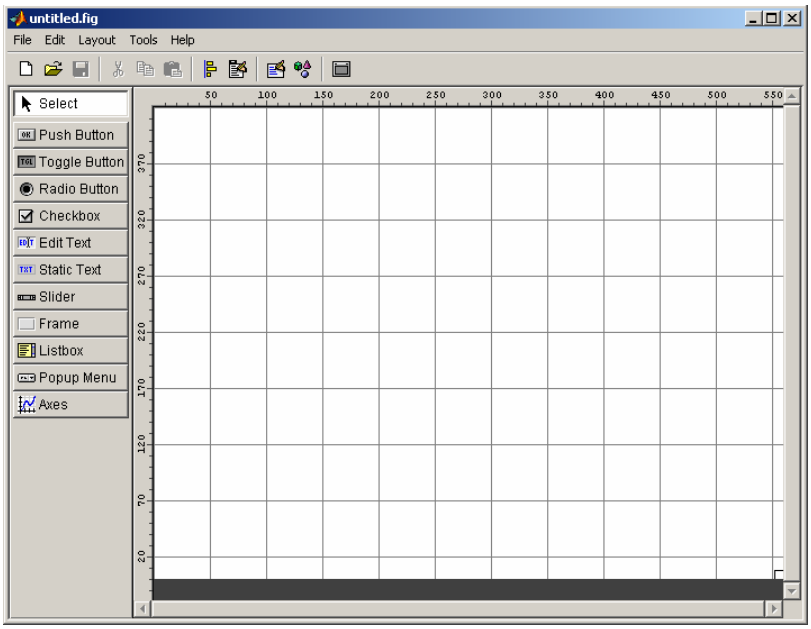
Note GUIDE จะสร้าง Application M-file มีชื่อเดียวกับ Fig-file ที่เรากำหนดจาก Layout Editor เมื่อเรากำหนดให้ GUI นั้น Active จาก Layout Editor, GUIDE พยายามที่จะให้ Application M-file นี้ทำงานและเปิดหน้าต่าง GUI นั้นขึ้นมา

❷ การเลือก GUIDE Application Options

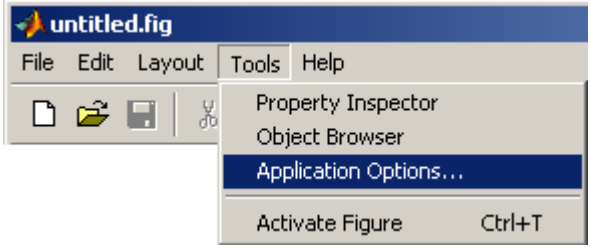
เมื่อเราต้องการจะใช้ GUIDE นั้น ครั้งแรก บน MATLAB COMMAND WONDOW ที่ prompt เราสั่ง

```
» guide
```

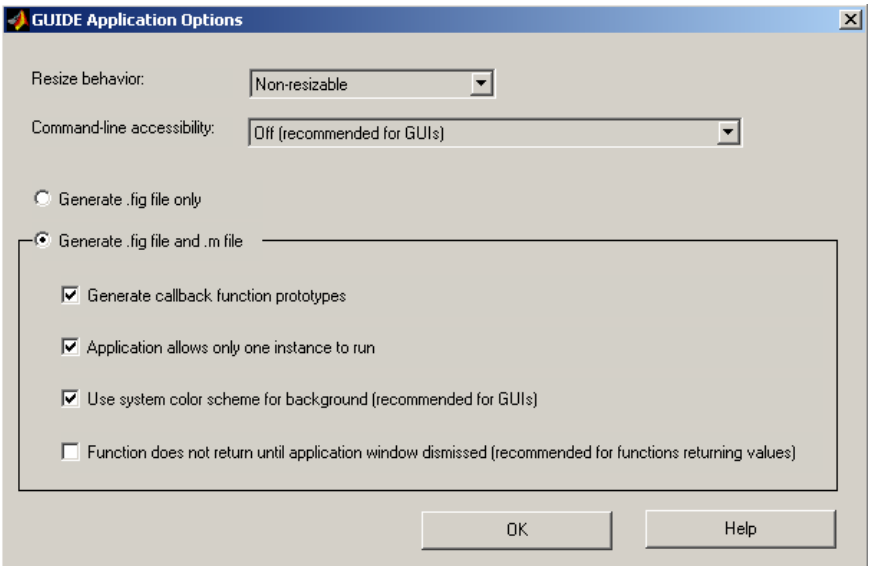
จากนั้น Layout Editor จะปรากฏขึ้น ซึ่งมีลักษณะดังรูป



ก่อนที่จะทำการเพิ่มส่วนประกอบต่าง ๆ ลงใน GUI เราควรกำหนดตัวเลือกต่าง ๆ ก่อน โดยภายใต้เมนู Tool เลือก Application Options



ซึ่งจะทำให้เราได้ หน้าต่าง GUIDE Application Options ซึ่งมีลักษณะตามรูป



นอกเหนือจากการเลือกให้ GUIDE จะสร้างเฉพาะ FIG-file หรือสร้างทั้ง Fig-file และ M-file แล้ว เรายังสามารถ กำหนดค่าต่าง ๆ ในหน้าต่างตัวเลือกนี้ได้ ซึ่งรายละเอียดมีดังนี้คือ

Resize Behavior

เป็นการกำหนดว่าผู้ใช้สามารถเปลี่ยนขนาดของหน้าต่าง GUI ที่สร้างขึ้นได้หรือไม่ และถ้าได้ จะให้ MATLAB ควบคุมการเปลี่ยนขนาดโดยผู้ใช้อย่างไร ซึ่ง GUIDE ให้ตัวเลือก 3 แบบคือ

- Non-Resizable ผู้ใช้ไม่สามารถเปลี่ยนขนาดของหน้าต่างได้ (default)
- Proportional ให้ผู้ใช้สามารถปรับขนาดของหน้าต่าง GUI ได้โดย MATLAB จะปรับขนาดขององค์ประกอบต่าง ๆ ใน GUI ให้มีสัดส่วนตามขนาดของหน้าต่าง GUI ที่เปลี่ยนไป
- User-Specifide มีการเขียนโปรแกรมกำหนดให้ GUI ปรับเปลี่ยนขนาดและตำแหน่งขององค์ประกอบต่าง ๆ ใน GUI ซึ่งการเลือกตัวเลือกนี้ผู้เขียน GUI ต้องเขียนคำสั่งเพื่อปรับขนาดและตำแหน่งขององค์ประกอบต่าง ๆ ใน GUI ให้ชัดเจน

สำหรับตัวเลือกตัวแรก เหมาะกับ GUI ที่ไม่ต้องการปรับปรุงขนาด ส่วนตัวเลือกที่สอง Proportional นั้นเหมาะสมกับการที่ให้ผู้ใช้ปรับปรุงขนาดหน้าต่างได้ แต่รายลเอียดต่าง ๆ ปล่อยให้เป็นไปโดยอัตโนมัติ และเราขอเตือนว่าหากปรับตำแหน่งด้วยตัวเลือก Proportional นี้ เมื่อปรับขนาดหน้าต่าง ขนาดตัวหนังสือใน GUI ไม่มีการปรับไปด้วย ดังนั้นหากเรากำหนดให้หน้าต่างมีขนาดเล็กเกินไป ตัวหนังสือต่าง ๆ อาจซ้อนทับกัน การปรับปรุงขนาดหน้าต่างด้วย User-Specify อาจจะเป็นการ

ปรับปรุงหน้าต่างขนาดต่าง ๆ ได้หลากหลายกว่า แต่ต้องมีการเขียนโปรแกรมควบคุม อย่างไรก็ตามเราจะไม่ขอกล่าวถึงรายละเอียดในที่นี้

Command-Line Accessibility

เมื่อ MATLAB สร้าง graph จะ figure และ axes ที่จะต้องเป็น parents ของรูปกราฟนั้น ซึ่ง MATLAB จะทำการมองหาก่อนว่ามี figure และ Axes เกิดขึ้นอยู่ในขณะนั้นหรือไม่ ถ้ามี MATLAB จะทำการเขียนกราฟลงใน figure และ Axes ที่มีอยู่ถ้าไม่ MATLAB จะทำการสร้าง figure และ Axes ขึ้นมาใหม่

ในการสร้าง GUI ของเรานั้น ส่วนมากแล้ว เราคงไม่ต้องการให้ผู้ใช้เขียนกราฟลงใน axes ที่ปรากฏอยู่ใน GUI ของเรา แต่บางกรณีเราอาจจะต้องการให้ผู้ใช้เขียนกราฟลงใน axes ที่ปรากฏอยู่ใน GUI ของเราก็ได้ ดังนั้น GUIDE จึงมีตัวเลือกให้เราเลือกสำหรับ Command-Line Accessibility ดังนี้

- Off - ป้องกันการสั่งเขียนกราฟผ่าน command-line บน GUI ของเรา (default)
- On – ให้มีการเขียนกราฟผ่าน command-line บน GUI ของเราได้
- User-Specified GUIDE จะให้ GUI ใช้ค่าที่กำหนดโดยคุณสมบัติ Handle Visibility และ Integer Handle ของ figure

Note คำสั่ง Findobj นี้จะใช้ไม่ได้กับวัตถุต่างที่อยู่ใน GUI Handle Visibility ของ Figure เป็น off เพราะเรากำหนดว่า handle ของ figure นี้จะมองไม่เห็นโดย MATLAB แม้ว่า figure นี้อาจจะปรากฏอยู่บนหน้าจอก็ตาม อย่างไรก็ตาม Application M-file จะมีข้อมูลเกี่ยวกับ handle ของวัตถุต่าง ๆที่เราสร้างใน GUI อยู่แล้ว ดังนั้นเราไม่จำเป็นต้องใช้คำสั่ง findobj เพื่อหา handle ของวัตถุต่างๆ สำหรับคุณสมบัติของ figure ที่เกี่ยวข้องกับการมองเห็น figure ที่สำคัญมี 2 ค่า คือ

- Handle Visibility ถ้าหากค่าคุณสมบัตินี้เป็น off ค่าของ handle ต่าง ๆ ที่เป็น children ของ figure นี้จะถูกลบออกจาก children ของ root object ทำให้ figure นี้ไม่เป็น current figure (current figure เป็นเป้าหมายในการสร้างกราฟของ MATLAB) อย่างไรก็ตาม handle เหล่านั้นยังคงใช้ได้ ดังนั้นคำสั่งต่าง ๆ ที่สั่งตรงถึง handle เหล่านั้นจึงเป็นไปได้
- Integer Handle ถ้าหากค่าคุณสมบัตินี้เป็น off แล้ว MATLAB จะกำหนดค่า handle ของวัตถุต่าง ๆ เป็นเลขจำนวนจริงที่จะไม่มีการกำหนดซ้ำอีกเช่น (68.0001224) แทนที่จะเป็นเลยจำนวนเต็ม ซึ่งจะเป็นการลดโอกาสที่จะมีวัตถุใน GUI อื่น ๆ ที่มี handle ซ้ำกับวัตถุใน GUI ของเรา

14.3 การสร้าง Application M-file ของ GUIDE

เมื่อเราสร้าง GUI โดย GUIDE และเลือกให้ GUIDE สร้าง FIG-file และ M-file เมื่อเราเลือกตัวเลือกนี้ จะมีตัวเลือกให้ผู้ใช้เลือกเพิ่มขึ้นได้อีก 4 ตัวเลือกเพื่อกำหนดลักษณะของ Application M-file ซึ่งตัวเลือกต่างๆ มีดังนี้

- Generate callback function prototypes
- Application allows only one instance to run
- Use system color scheme for background
- Function does not return until application window dismissed

โดยรายละเอียดของตัวเลือกต่าง ๆ มีดังนี้

❶ การสร้างต้นแบบของ Application M-file

เมื่อเราเลือกตัวเลือก Generate Callback Function Prototype ในการเลือกตัวของ GUIDE Application Option ก็จะทำให้ GUIDE เพิ่ม sub function ให้กับ application M-file สำหรับทุกวัตถุที่เราสร้างขึ้นใน GUI (ยกเว้น frame และ Static text) อย่างไรก็ตาม GUEDE จะสร้างเฉพาะ sub function เป็นต้นแบบไว้เท่านั้น ส่วนคำสั่งต่าง ๆนั้นเราต้องเป็นผู้เขียนใน sub function นั้นเอง นอกเหนือจากนั้น GUIDE ยังจะเพิ่ม sub function ทุกครั้งเมื่อเราแก้ไข callback จากการใช้เมาส์ปุ่มขวาในเมนู context

สำหรับการสร้างต้นแบบของ callback sub function นั้นจะสร้างขึ้นโดยมีลักษณะดังนี้

```
function object.Tag_Callback(h,eventdata,handles.varagin)
```

โดย arguments ต่าง ๆ จะเป็นดังนี้

h เป็น handle ของวัตถุที่เรียก callback นี้

eventdata ว่างสำรองเก็บไว้ใช้ในอนาคต

handles เป็นตัวแปรแบบ structure ที่บรรจุ handle ของทุกวัตถุที่อยู่ใน GUI โดยชื่อของ fileจะเป็นชื่อ tag ของวัตถุนั้น เราสามารถใช้ตัวแปรนี้ส่งข้อมูลเกี่ยวกับ handle ของวัตถุต่าง ๆ ใน GUI ไปที่ callback ตัวอื่นหรือโปรแกรมตัวอื่น ๆ ใน MATLAB ได้

varargin เป็น variable-length แสดงผลที่เราต้องการส่งผ่านไปที่ callback function ตัวอย่างเช่น ถ้าเราวาง push button ที่เรากำหนด Tag เป็น pushbutton1 จะทำให้ GUIDE สร้าง sub function ใน application ดังนี้

```
function pushbutton1_Callback(h,evendata,handles,varargin)
```

หลังจากนั้น GUIDE จะกำหนดคุณสมบัติ Callback ของ push button นี้เป็น

```
Mygui('pushbutton1_callback',gcbo,[],guidata(gcbo))
```

โดย

Mygui	เป็นชื่อของ FIG-file ที่เก็บ GUI นี้
Pushbutton1	Callback-เป็นชื่อของ callback sub funcion
gcbo	เป็นคำสั่งที่ให้ handle ของวัตถุนี้ ในที่นี้คือ push button
[ ]	เป็น เมตริกส์ว่างใช้เป็นที่เก็บ eventdata
guidata(gcbo)	เป็น handles structure ที่ได้จากข้อมูลที่บรรจุอยู่ใน figure ที่เป็น GUI

ถ้าหากเราต้องการจะส่งข้อมูลอื่น ๆ ผ่านต่อเข้าไปสู่ sub function นอกเหนือจากที่ GUIDE ได้สร้างเป็นต้นแบบ เราสามารถทำได้ดดยการเพิ่ม argument เข้าไปด้วยการแยกข้อมูลต่าง ๆ ด้วยเครื่องหมาย , ซึ่งการแก้ไขจะต้องทำทั้งใน (1) sub function ใน application M-file และ (2) ใน คุณสมบัติ callback ของวัตถุนั้น เช่นถ้าเราต้องการส่งผ่านตัวแปร gx1 และ gx2 ไปสู่ subfunction เพิ่มเติมให้เราแก้ไขคำสั่ง function ใน M-file เป็น

```
MYGUI('pushbutton1_callback',gcbo,[],guidata(gcbo),gx1,gx2)
```

กำหนดให้ GUI ทำงานที่ละครั้ง

ตัวเลือกนี้เป็นการกำหนดตัวเลือกว่าจะให้ GUI นั้นทำงานอย่างไร

- ยอมให้ MATLAB แสดง GUI นี้เพียงที่ละหนึ่งหน้าต่าง ในเวลาหนึ่ง ๆ
- ยอมให้ MATLAB แสดง GUI นี้ได้หลายหน้าต่างพร้อมกันในเวลาหนึ่ง ๆ

ถ้าเราเลือกให้ MATLAB ใช้ GUI นี้ได้เพียงหน้าต่างเดียวในแต่ละเวลาหนึ่ง ๆ จะทำให้ MATLAB เลือกใช้ GUI รูปเดิมขึ้นมาแสดงเมื่อมีการเรียกใช้ GUI นี้ซ้ำแทนที่จะสร้างขึ้นมาใหม่ แต่ถ้าเราไม่เลือกตัวเลือกนี้ MATLAB จะสร้าง GUI ตัวใหม่ขึ้นมาทุกครั้งที่มีการเรียกคำสั่งใช้ GUI นี้

สำหรับ code ใน application M-file ที่กำหนดตัวเลือกนี้จะอยู่ด้านบนของ application M-file โดยใช้คำสั่ง

```
fig = openfig(mfilename,'reuse')
```

กรณีที่เลือกตัวเลือกนี้และ

```
fig = openfig(mfilename,'new')
```

สำหรับกรณีที่เราไม่เลือกตัวเลือกนี้

Note ให้แน่ใจว่าเรามีการสั่ง openfig ใน application M-file รวมถึงใน command line เพียงครั้งเดียว

❷ ใช้สีพื้นที่กำหนดด้วย System ที่ MATLAB ทำงานอยู่

สีที่ใช้ในระบบและส่วนประกอบของ GUI จะเปลี่ยนไปตามระบบคอมพิวเตอร์ที่ใช้ตัวเลือกนี้ยอมให้เราใช้สีพื้นของ uicontrol เป็นสีเดียวกับสีพื้นของ figure ซึ่งจะทำให้ GUI ของเราดูมีความกลมกลืนเข้ากับสีพื้น แต่หากเราต้องการปรับเปลี่ยนสีพื้นของ uicontrol ที่ใช้ให้เป็นไปตามต้องการ เราก็ไม่ต้องเลือกตัวเลือกนี้

สำหรับใน application M-file จะมีคำสั่งในการเลือกตัวเลือกนี้คือ

```
%Use system color scheme for figure
set(fig,'color',get(0,'Default UIcontrol Background Color');
```

Note ขอให้แน่ใจว่ามีคำสั่งนี้ใน M-file รวมถึงใน command line เพียงครั้งเดียว

❸ การให้รอ input ของผู้ใช้

สำหรับตัวเลือกใน GUIDE application option

Function does not return until application window dismissed

เป็นการสร้าง application M-file ที่ออกแบบให้รอ input จากผู้ใช้ ซึ่งสามารถทำได้โดยใช้ฟังก์ชัน uiwait ซึ่งจะป้องกันการทำงานต่อไปของ M-file

ในขณะการทำงานนี้ถูกสั่งให้รอ MATLAB จะจัดคิวให้กับคำสั่งต่าง ๆ ที่เราสั่งเข้าไปไว้ตามลำดับแต่ยังไม่ให้เกิดการทำงานตามคำสั่งนั้น จนกว่าจะเกิดปรากฏการณ์ต่อไปนี้เกิดขึ้น

- รูป GUI ถูกลบ
- ใน GUI มี callback ที่ให้คำสั่ง uiresume

คำสั่งนี้มีประโยชน์ที่จะป้องกัน MATLAB ที่จะใช้คำสั่งจาก command line จนกว่ามีการตอบสนองต่อ dialog box แต่ในขณะเดียวกันยอมให้ callback ทำงานได้

ใน application M-file จะมีการสร้าง code ดังนี้

```
%wait for callbacks to run and window to be dismissed
uiwait(fig);
```

เมื่อ fig เป็น handle ของ figure นี้

Note ขอให้แน่ใจว่ามีคำสั่งนี้ใน M-file รวมถึง command line เพียงครั้งเดียว

❹ การตั้งชื่อไฟล์และ Tag

ในการกำหนดชื่อของไฟล์ หรือวัตถุต่าง ๆ ที่ใช้ใน GUI ซึ่งจะตั้งชื่อโดยคุณสมบัติ Tag สำหรับ GUIDE กำหนดค่าคุณสมบัติ Tag (หรือกำหนดชื่อของวัตถุนั้น) ให้กับวัตถุทุกแบบที่สร้างขึ้นโดยอัตโนมัติ เช่น pushbutton1 และให้ string นี้ จะนำไปใช้เป็นชื่อ callback sub function เช่น pushbutton1_callback อย่างไรก็ดี เพื่อให้ชื่อของวัตถุบ่งบอกถึงหน้าที่ของมันมากขึ้น เราอาจจะตั้งชื่อของวัตถุนั้นให้สื่อถึงหน้าที่ของมันมากขึ้น ดังนั้นเราแนะนำว่า หลังจากที่เราสร้างวัตถุนั้นขึ้นมาแบ้ว เราควรจะตั้งชื่อให้มันด้วย การตั้งชื่อของมันก็คือ การกำหนด คุณสมบัติ Tag ของมันนั่นเอง และเราควรจะทำก่อนที่จะ active หรือ save GUI นี้ด้วย

การใช้ save as จะทำให้ GUIDE ได้สร้าง application M-file ขึ้นมาใหม่ และปรับค่าคุณสมบัติ callback ให้เหมาะสมกับ callbacks ที่มีอยู่ด้วย

Note เนื่องจาก GUIDE ใช้คุณสมบัติ Tag เพื่อสร้างเป็น function และตัวแปรใน structure file ดังนั้น ชื่อ Tag ที่เราเลือกจะต้องเป็นตัวแปรที่ใช้ได้ตามข้อกำหนดของ MATLAB

การเปลี่ยนชื่อคุณสมบัติ Tag เราควรจะมีการปรับเปลี่ยนก่อนจะ Activate หรือ save รูป GUI และสร้าง Application M-file เพื่อป้องกันการสับสน อย่างไรก็ตามหากว่าเราทำการปรับเปลี่ยน Tag ของคุณสมบัติใด ๆ หลังจากเคยสร้าง application M-file ขึ้นมาแล้ว อาจจะมีปัญหาบางประการเกิดขึ้น เพราะมีบางส่วนที่ GUIDE จะไม่เปลี่ยนแปลงชื่อใน application M-file ให้โดยอัตโนมัติ ทำให้เราต้องตามเข้าไปแก้ไขใน application M-file เอง

ถ้าเราเปลี่ยน Tag หลังจากสร้าง application M-file GUIDE จะไม่สร้าง sub function ใหม่หใ อย่างไรก็ตาม เนื่องจาก handles นั้นจะสร้างขึ้นในเวลาที่ MATLAB ทำงาน ดังนั้น GUIDE จะใช้ชื่อ Tag ใหม่ในการสร้าง file ในตัวแปร handles ดังนั้นถ้าเติมใน application M-file ที่ใช้คำสั่ง

```
x = get(handles,listbox1,'string')
```

ถ้าเราเปลี่ยน Tag จาก Listbox1 เป็น graph เราจะต้องเปลี่ยนคำสั่งใหม่เป็น

```
x = get(handles,graph,'string')
```

เพราะในการทำงานใหม่ของ GUI จะไม่มี file ใน structure handles ที่ชื่อ listbox1 อีกต่อไปแล้ว และจะเกิด error ขึ้นเมื่อเราสั่ง MATLAB ทำงาน ถ้าไม่เปลี่ยนคำสั่งตามที่กล่าวไว้ ดังนั้นของแนะนำให้มีการเปลี่ยนชื่อที่สร้างโดยอัตโนมัติ หรือที่เราสร้างเป็นคำสั่งในภายหลังให้เหมาะสม

ข้อสำคัญอีกประการหนึ่งคือการสร้าง FIG-file และ M-file และ M-file ที่จะนำมาใช้งานร่วมกัน GUIDE ก็จะสร้างชื่อ file เหมือนกัน แต่มี extension ต่างกัน (fig และ M) แต่ถ้าเราเปลี่ยนชื่อ file ใด file หนึ่ง การทำงานจะ

GUI Building Tools (GUI Layout Tool)

MATLAB จะมีเครื่องมือในการช่วยสร้าง GUI อยู่หลายส่วนโดยจะเริ่มต้นจาก GUI Layout tools ซึ่งถือว่าเป็นขั้นตอนแรกในการสร้าง GUI เพราะจะเป็นการกำหนดว่าใน GUI นี้จะมี uicontrol และชื่อ axes อะไร อยู่ใน GUI อะไรบ้าง และแต่ละตัวจะมีตำแหน่งอยู่ที่ใด ขนาดเท่าไร รูปแบบ สี เป็นอย่างไร ซึ่งเครื่องมือนี้จะประกอบด้วย

- Layout Editor – เพิ่มและจัดวัตถุต่าง ๆ ใน GUI
- Alignment Tool – จัดวางวัตถุเทียบกับวัตถุอื่น ๆ ใน GUI ให้เป็นระเบียบมากยิ่งขึ้น
- Property Inspector – ตรวจสอบและตั้งค่าคุณสมบัติต่าง ๆ ของวัตถุแต่ละอัน
- Object Browser – ตรวจสอบและแสดงลำดับขั้นของวัตถุที่มี handle ใน MALAB ขณะนั้น
- Menu Editor – สร้างเมนูของหน้าต่าง และ context menu

เราจะเข้าสู่เครื่องมือต่าง ๆ นี้ได้โดยผ่านเข้าทาง GUIDE Layout Editor ในการเริ่มการทำงานของ Layout Editor ให้ใช้คำสั่ง

```
»guide
```

จากนั้น MATLAB จะแสดง GUI ใหม่ขึ้นมาโดยยังไม่มีวัตถุวางอยู่ใน GUI นั้น หรือถ้าเราสั่ง

```
»guide mygui.fig
```

(จะมี .fig หรือไม่ก็ได้) จะเป็นการเรียก GUI เก่าขึ้นมาเพื่อแก้ไข หรือถาต้องการแก้ไข GUI เก่าเราอาจใช้คำสั่ง open ภายใต้เมนู File ของ Layout Editor ก็ได้

❺ การวางส่วนประกอบต่าง ๆ ลงใน GUI โดย Layout Editor

การใช้ Layout Editor จะช่วยให้เราสามารถกำหนดส่วนประกอบต่าง ๆ ว่าจะมีอะไร และมีตำแหน่งอยู่ที่ใด การวางตำแหน่งวัตถุต่าง ๆ ซึ่งก็จะมี uicontrol และ axes ก็จะเหมือนการใช้โปรแกรมวาดรูปทั่ว ๆ ไป มีขั้นตอนดังนี้คือ

1. เลือก uicontrol หรือ axes ที่ต้องการจะเพิ่มไปใน GUI จาก component palette
2. เลื่อนเมาส์เข้ามาในบริเวณพื้นที่ของ GUI ลักษณะ cursor จะเปลี่ยนเป็นรูปกากบาทซึ่งเราจะสามารถใช้กำหนดตำแหน่ง มุมซ้ายบน ของวัตถุนั้นได้ โดยการกดเมาส์ปุ่มซ้ายที่ตำแหน่งที่เราต้องการ แล้วลากเมาส์ขณะกดปุ่มเมาส์ด้านซ้ายอยู่ เพื่อกำหนดตำแหน่งด้านขวาล่างของวัตถุเมื่อให้ตำแหน่งที่ต้องการให้เราปล่อยปุ่มเมาส์

3. เราสามารถปรับปรุงขนาดและเลื่อนตำแหน่งของวัตถุนั้นได้ โดยใช้เมาส์เลือกวัตถุนั้น แล้วเลื่อนหรือปรับขนาดได้ตามต้องการ

❻ Activating the Figure

เราสามารถสร้างการทำงานของ GUI ได้โดยสั่ง activate figure ที่เราได้ออกแบบมาแล้วด้วย Layout Editor เราสามารถ Activate รูปได้โดยเลือก activate figure ภายใต้เมนู Tool หรือโดยการกด Activator button บน Toolbar

เมื่อเราสั่ง activate figure สิ่งต่อไปนี้จะเกิดขึ้น

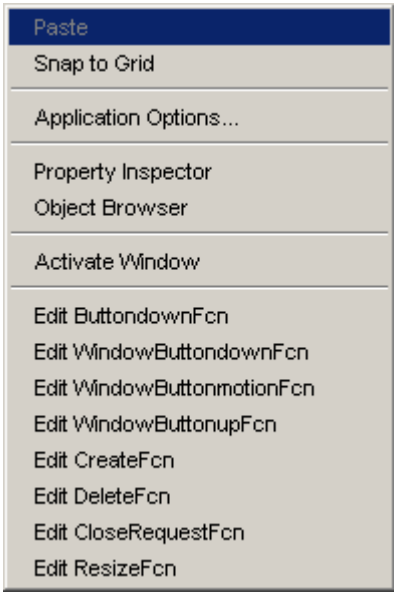
ก่อนอื่น GUIDE จะทำการ SAVE ไฟล์ทั้ง M-file และ Fig-file เป็นอันดับแรก ถ้า file ทั้งสองไม่เคยถูก SAVE มาก่อน ก็จะมี Dialog box SAVE AS เกิดขึ้น เพื่อถามชื่อ FILE ที่เราต้องการ SAVE ถ้าเราใช้ชื่อไฟล์ที่มีอยู่แล้ว MATLAB จะถามว่าเราต้องการเขียนทับ เขียนเพิ่ม (Append) หรือยกเลิกการใช้ชื่อนั้น

Note เพื่อให้เราสามารถสร้าง GUI ที่มีประสิทธิภาพและควบคุมการทำงานได้อย่างสะดวก เราควรตั้งชื่อ Tag ของวัตถุต่างๆ ใน GUI ให้เรียบร้อยก่อนจะมีการ Activate Figure เพราะการแก้ไขชื่อนี้ภายหลังจะมีปัญหายุ่งยากตามมาได้

Layout Editor Context Menu

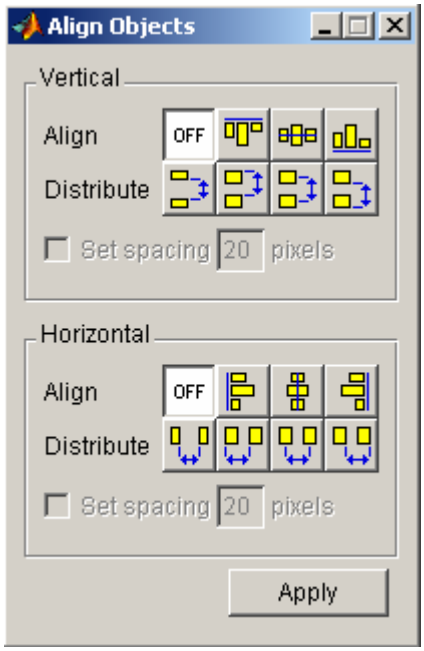
เมื่อเราทำงานภายใต้ Layout Editor เราสามารถเลือกวัตถุนั้นด้วยเมาส์ปุ่มซ้าย และเมื่อเรากดเมาส์ปุ่มขวาเหนือวัตถุนั้น ก็จะปรากฏ context menu ขึ้น นอกเหนือจากที่เราจะใช้เมนูที่ปรากฏอยู่ด้านบนของหน้าต่าง ซึ่งเราสามารถใช้ context menu นี้ สร้าง subfunction ให้กับ application M-file ของเราได้ สำหรับทุกวัตถุชนิดที่มี callback routine

ในรูปข้างล่างนี้แสดง เมนู Context ของ figure



❼ Aligning Component in The Layout Editor

ในการจัดเรียงส่วนประกอบต่างๆ ที่มีอยู่ แม้ว่าสามารถที่จะใช้เมาส์เลื่อนวัตถุต่างๆ ได้อยู่แล้ว แต่การจัดเรียงส่วนประกอต่างๆ ให้วางอยู่ในแนวเดียวกัน มีระยะห่างเท่าๆ กันนั้น จะมีความสะดวกขึ้น หากเราใช้ Alignment Tool เราสามารถเลือก Alignment Tool ได้โดยเลือก Alignment Tool จากปุ่มบนเมนูซึ่ง Alignment Tool จะมีลักษณะดังนี้



ก่อนที่เราจะใช้ Alignment Tool เราต้องเลือกกลุ่มวัตถุที่จะจัดเรียงเสียก่อน ซึ่งเราสามารถทำได้โดย

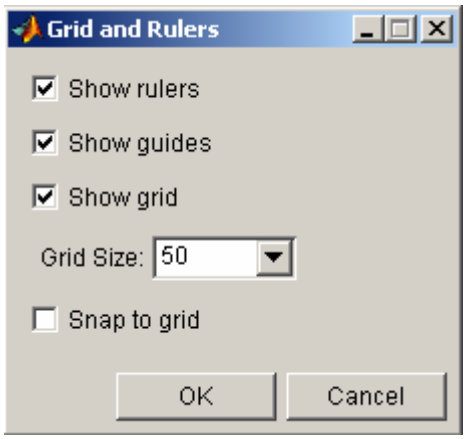
- เลือกลูกศร (select) จาก component palette แล้ว กำหนดพื้นที่กรอบสี่เหลี่ยมที่บรรจุ

วัตถุทั้งหมดที่ต้องการ Align เมื่อปล่อยเมาส์ วัตถุเหล่านั้นจะถูกเลือก

- เลือกวัตถุที่ละอัน โดยกดแป้น Shift บนแป้นพิมพ์ค้างไว้ แล้วเลือกวัตถุที่ต้องการไปเรื่อยๆ เมื่อเลือกกรอบแล้ว จึ้งปล่อยแป้นพิมพ์

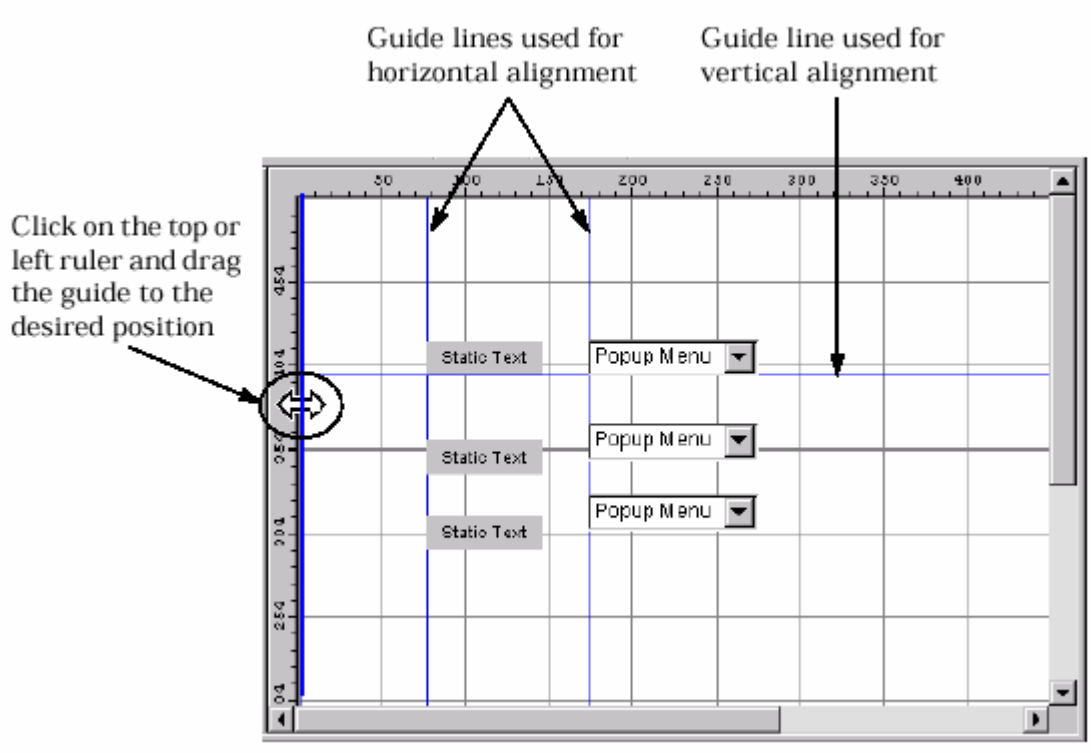
หลังจากที่เราเลือกวัตถุครบถ้วนแล้ว เราจึงเลือกวิธีการจัดเรียงวัตถุเหล่านั้นว่า เราต้องการให้จัดเรียงอย่างไร ทั้งในแนว Vertical และ Horizontal เมื่อเลือกลักษณะการจัดเรียงเรียบร้อยแล้วจึงกดปุ่ม Apply เพื่อจัดแนวและตั้งระยะห่างวัตถุทั้งในแนวตั้งและแนวนอนให้เป็นไปตามที่เราต้องการ

นอกเหนือจากการใช้ Alignment Tool เพื่อจัดเรียงวัตถุนั้นแล้วเรายังสามารถใช้ Grids และ Rulers เพื่อช่วยในการจัดเรียงโดย grid ที่สร้างขึ้นนั้นสามารถปรับปรุงได้โดยเรียก Grid and Rulers ภายใต้เมนู Layout



โดยเราสามารถกำหนด Grid size ได้ระหว่าง 10 57, 200 Pixel โดยค่า 50 เป็น default นอกจากนี้เรายังมีตัวเลือก Snap-to-grid เพื่อกำหนดให้วัตถุที่มีการเคลื่อนที่หรือปรับขนาดที่อยู่ในระยะ 9 pixels ของเส้น grid จะเคลื่อนที่เข้าหาเส้น grids การเลือก snap-to-grid นี้จะทำงานทั้งที่เราแสดงหรือไม่แสดงเส้น grid บน Layout Editor

นอกเหนือจากนั้นเรายังสามารถสร้าง guide line ขึ้นมาเพื่อสะดวกในการกำหนดตำแหน่ง การสร้าง guide line นี้ทำได้โดยใช้เมาส์ปุ่มซ้ายกดที่ ruler ด้านบนหรือด้านซ้ายมือ จากนั้นดึงเส้นเข้ามาภายในพื้นที่ของ gui จะมีเส้นตรงตามเมาส์เข้ามาได้ เมื่อเราปล่อยเมาส์ เส้นตรงใหม่ก็จะกลายเป็นเส้น grid เส้นใหม่แต่แสดงสีที่แตกต่างออกไป การสร้าง grid แสดงในรูป



และการจัดเรียงวัตถุใน GUI ที่จะกล่าวถึง เป็นแบบสุดท้ายในที่นี้คือการจัดเรียงลำดับการวางทับกันบน GUI ซึ่งปกติวัตถุที่สร้างที่หลังจะวางอยู่ด้านบนวัตถุที่สร้างก่อน แต่เราสามารถปรับลำดับได้โดยกดเมาส์ปุ่มขวาเมื่อเลือก context menu แล้วเลือก

Bring to Front, Send to back, Bring Forward หรือ **send Backward**

ตามต้องการ

14.4 การกำหนดค่าคุณสมบัติของส่วนประกอบต่างๆ

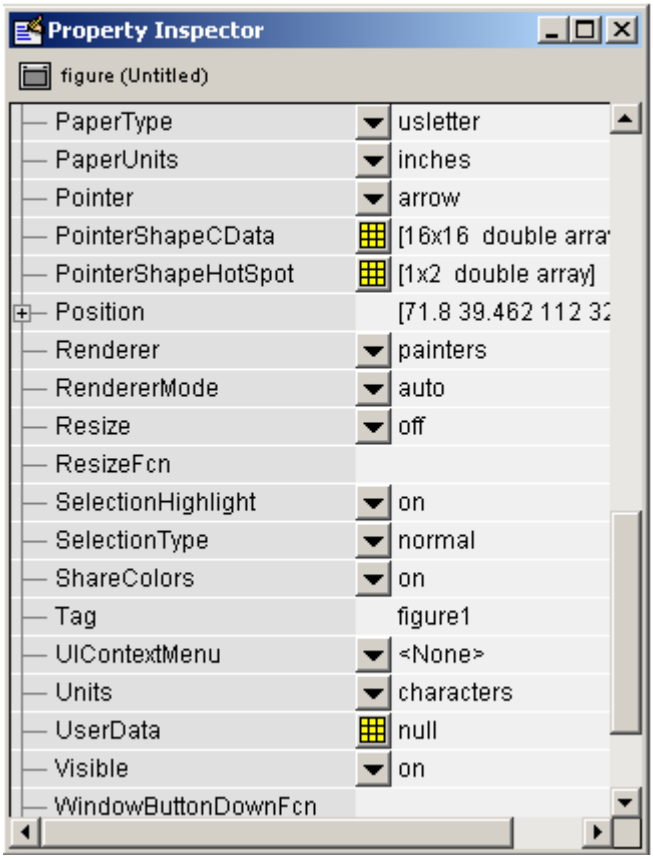
เราสามารถที่จะกำหนดค่าคุณสมบัติของส่วนประกอบต่างๆใน GUI ได้ด้วยการใช้ Property Inspector ซึ่งจะให้รายการคุณสมบัติทั้งหมดของวัตถุที่เราเลือกและแสดงค่าปัจจุบันของคุณสมบัติเหล่านั้น สำหรับคุณสมบัติแต่และตัวในรายการนั้น จะมีอุปกรณ์ที่ใช้ในการแก้ไขคุณสมบัติแต่ละตัวไว้ด้วย คุณสมบัติบางตัวซึ่งมีตัวเลือกอุปกรณ์แก้ไขก็จะแสดงตัวเลือกไว้ให้ ส่วนคุณสมบัติบางตัวต้องเป็นการกำหนดค่า ก็จะเป็นการกำหนดค่าลงไป

การที่เราจะให้ Property Inspector ปรากฏขึ้นเราสามารถทำได้หลายวิธีคือ

- กดเมาส์ปุ่มซ้ายสองครั้ง ส่วนประกอบที่ต้องการแสดงคุณสมบัติ
- เลือก Property Inspector ภายใต้เมนู Tools
- เลือก Inspect property ภายใต้เมนู Edit
- กดเมาส์ปุ่มขวาบนวัตถุนั้น แล้วเลือก Inspect Properties จากเมนู context

- กดเมาส์ปุ่มซ้ายที่ Property Inspector ที่ Toolbar

และ Property Inspector จะแสดงคุณสมบัติของวัตถุที่เราเลือกบน Layout Editor เมื่อเราเปลี่ยนวัตถุที่เลือกไป Property ที่แสดงก็จะเปลี่ยนไปตามวัตถุนั้นด้วย

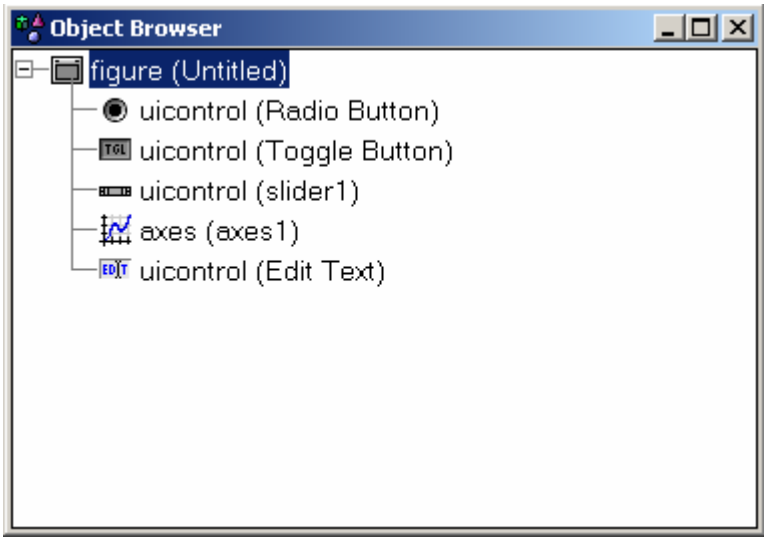


เมื่อเราตรวจคุณสมบัติเหล่านั้น เราก็สามารถจะปรับแก้คุณสมบัติต่างๆ ได้ตามต้องการสำหรับคุณสมบัติที่มีเครื่องหมาย อยู่ด้านหน้าชื่อคุณสมบัติ ⊞ หมายความว่าเราสามารถขยายคุณสมบัติเหล่านั้นได้ เพื่อปรับแก้คุณสมบัติย่อยแต่ละตัวอย่างอิสระ

ในกรณีที่เราเลือกวัตถุหลายวัตถุพร้อมกัน Property Inspector จะแสดงคุณสมบัติที่วัตถุนั้นมีร่วมกัน ส่วนค่าที่แสดงนั้นหากวัตถุแต่ละชิ้นมีค่าไม่เท่ากัน ค่าที่แสดงจะปรากฏเป็น **`Mixed`** ขึ้น หมายความว่าเป็นค่ารวมหลายๆ ค่าอยู่ โดยแต่ละวัตถุนี้มีคุณสมบัตินี้ไม่เท่ากัน ถ้าเราปรับเปลี่ยนค่าดังกล่าว คุณสมบัติของวัตถุทุกตัวที่เราเลือกก็จะเปลี่ยนไปมีค่าเท่ากัน ซึ่งจะมีประโยชน์ในการกำหนดขนาด สี แบบตัวอักษรของวัตถุหลายๆชนิด ที่เราต้องการให้มีคุณสมบัติบางอย่างเหมือนกัน ในการกำหนดครั้งเดียวแทนที่จะปรับแก้ทีละตัว

❶ The Object Browser

OBJECT BROWSER จะแสดงลำดับขั้น ของวัตถุต่างๆ ที่มีอยู่ในรูป ในรูปต่อไปนี้แสดงรายการแสดงของวัตถุที่มีอยู่โดยจะแสดง figure และ children ของ figure ทั้งหมดตามขั้นและลำดับการแสดง เราสามารถใช้ object Browser ในการเลือกวัตถุต่างๆ บน GUI ได้

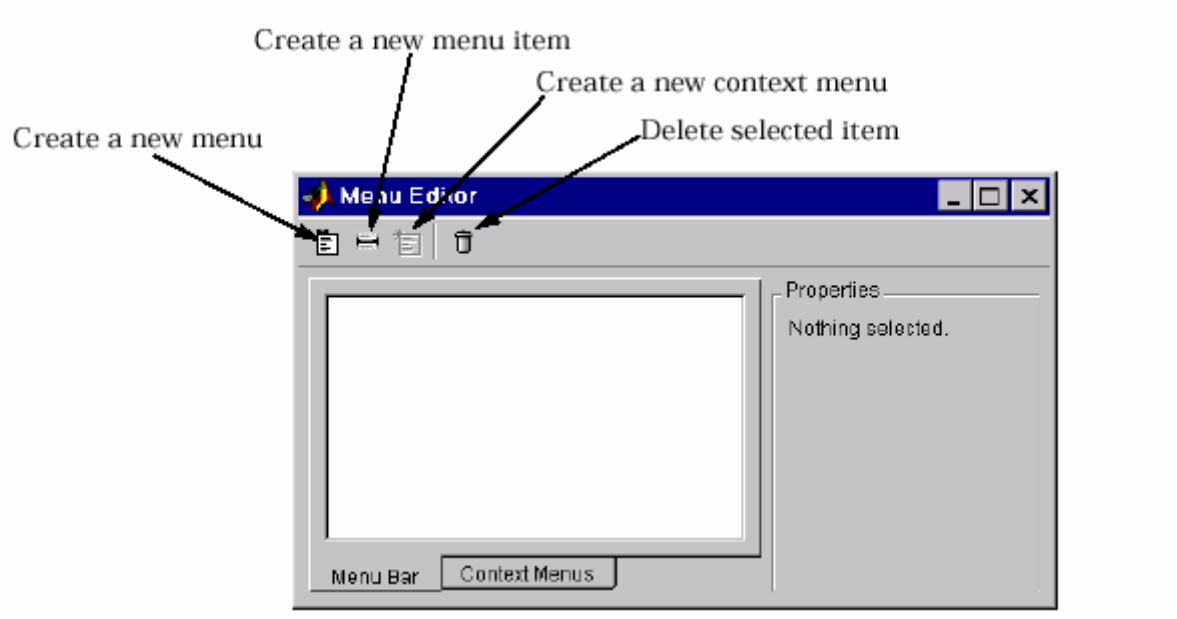


The Menu Editor

เราสามารถที่จะสร้างเมนูได้สองแบบใน MATLAB คือ

- Menu bar object - เป็นเมนูที่จะแสดงผลบน figure menu bar
- Context Menu - เป็นเมนูที่ปรากฏขึ้นเมื่อผู้ใช้กดเมาส์ปุ่มกดในวัตถุ

เราสามารถสร้างเมนูทั้งสองแบบโดยการใช้ Menu Editor ซึ่งเราสามารถเรียกใช้ได้โดยการกดเลือก Menu Editor บน Layout Editor Toolbar หรือ เรียก Edit Menu bar ภายใต้เมนู Layout ซึ่ง Menu Editor จะมีลักษณะดังรูป

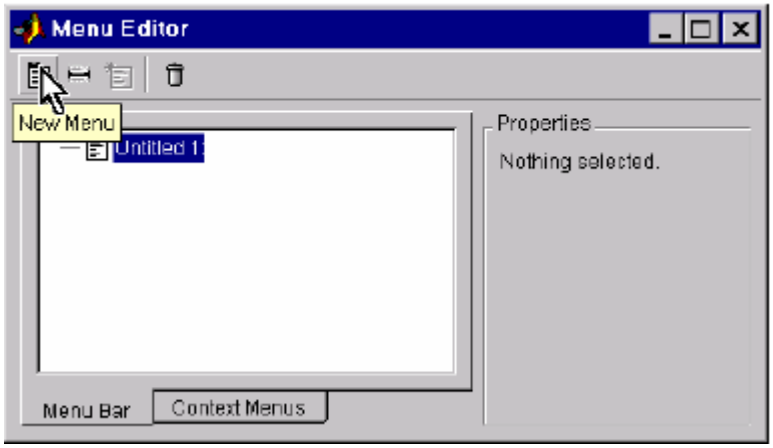


เมนูเหล่านี้จะทำงานภายใต้คำสั่ง UI menu และ UIcontext menu ของวัตถุนั้นๆ

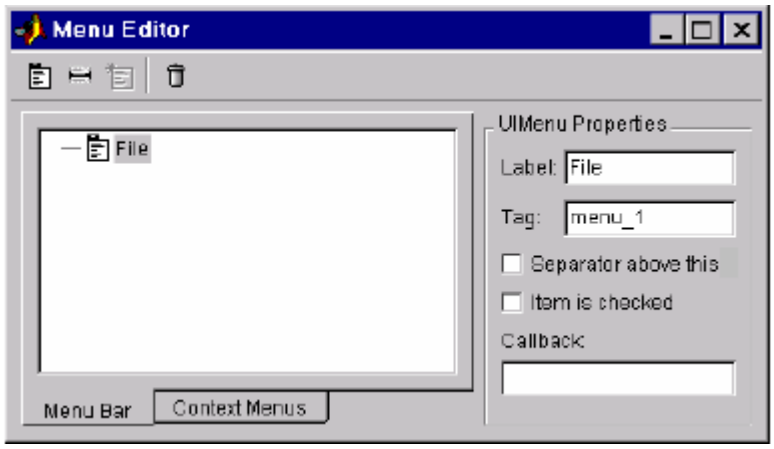
❷ การกำหนดเมนูบน Menubar

เมื่อเราสร้างเมนู MATLAB จะเพิ่มเมนูเหล่านั้น ลงใน figure Menu bar จากนั้นเราสามารถจะสร้างรายการให้แต่ละเมนูที่เราสร้างขึ้น และแต่ละรายการที่เราสร้างขึ้นก็สามารถมีรายการย่อย เพิ่มเข้าไปได้อีกย่อยลงไปเรื่อยๆ

ในการสร้างเมนูขั้นแรกกำหนดเลือก New Menu บน Toolbar ตามรูป



จากนั้นเมื่อเรากดเมาส์ไปที่เมนูที่กำหนดขึ้นใหม่จะมี UI MENU Property ปรากฏขึ้น เพื่อให้เรากำหนด Lable (ชื่อที่จะแสดง), Tag (ชื่อที่เราจะเรียก) Seperator (การแบ่งเมนูออกเป็นส่วนๆ) Checked Menu property (กำหนดว่าเป็นตัวที่ถูกเลือกไว้ก่อนหรือไม่) และสุดท้ายคือการกำหนด Callback ของ menu นั้น



หากเราต้องการสร้างรายการให้กับเมนูที่เราสร้างขึ้น ให้เราเลือก New Menu Item เพื่อกำหนดรายการสำหรับเมนูที่เราเพิ่งสร้างขึ้นนั้น ยกตัวอย่างคำสั่ง Print ภายใต้เมนู File

หากเราต้องการสร้างรายการเพิ่มให้กับเมนูเดิม ให้เราเลือกที่เมนูเดิมก่อนจากนั้น จึงเลือก New Menu Item แต่ถ้าเราอยู่ที่รายการภายใต้เมนูแล้วเราเลือก New Menu Item มันจะเป็นการสร้างรายการย่อยต่อลงจากรายการนั้น

❸ การกำหนด callback ของเมนู

คำสั่งในเมนูที่ปรากฏอยู่จะสั่งให้ callback ของมันทำงานเมื่อผู้ใช้เลือกรายการในเมนูนั้น เราสามารถเขียนคำสั่ง MATLAB ลงในช่องของ callback ของ Menu Editor ซึ่งปกติการกำหนดแบบนี้จะสะดวก ถ้าคำสั่งเป็นคำสั่งง่ายๆ เช่น print-dsp อย่างไรก็ตามโดยทั่วไปจะดีกว่าถ้าเราเพิ่ม Subfunction ลงไปใน Application- M – file เหมือนกับที่ GUIDE สร้างให้กับ UI control callback

Note GUIDE จะไม่สร้าง callback subfunction สำหรับเมนูใน application M-file โดยอัตโนมัติ เราจะต้องเพิ่ม subfunction ไปเอง

การกำหนด callback String นั้นก็ขึ้นกับว่าคำสั่งในเมนูของเรายุ่งยากเพียงใด หากเป็นคำสั่งง่ายๆ เราสามารถใช้ช่อง callback กำหนดคำสั่งได้เลย หากว่าคำสั่งค่อนข้างซับซ้อนเราจำเป็นต้องเขียน subfunction
จะเป็นดังนี้

```
My Gui ('Name_of_subfunction', gcbo, I J, guidata (gcbo))
```

เมื่อ **My Gui** คือซึ่งของ Applicaton M-file

Name_of_subfunction คือซึ่งของ subfuncton ที่จะใช้กับคำสั่ง หรือรายการในเมนู

gcbo เป็น handle ของรายการนี้

[] เป็นเมตริกส์ ว่าง สำรองไว้ใช้ภายหลัง

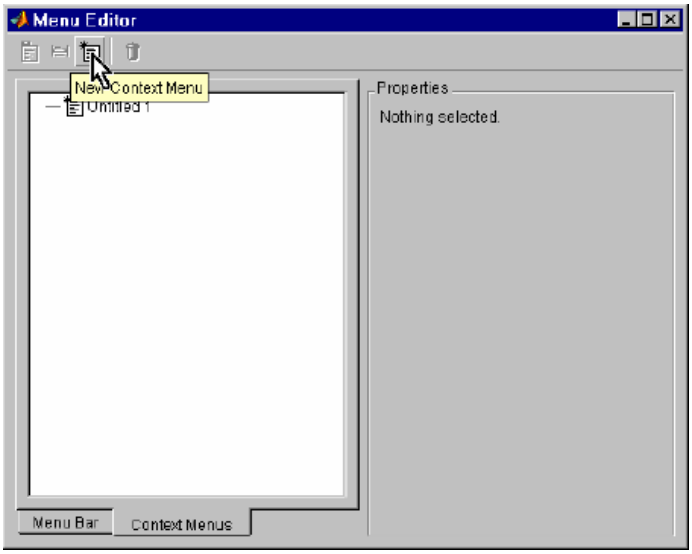
guidata (gcbo) เรียก **handles** ของวัตถุทั้งหมดที่มีอยู่ใน GUI นี้

ซึ่งคำสั่งนี้จะเขียนลงในช่อง callback ของ Menu Editor จากนั้นใน application M-file ให้เราเพิ่ม subfunction มีชื่อ Name_of_ subfunction ลงไป

❹ การสร้าง Context Menu

การสร้าง Context Menu เพื่อใช้กับวัตถุต่างๆ เมื่อเรากดเมาส์ปุ่มขวาลงบนวัตถุนั้น สามารถสร้างได้จาก Menu Editor ลำดับขั้นในการสร้าง context menu ดังนี้

1. สร้าง Paren Menu รายการต่างๆ ที่บรรจุอยู่ใน context menu จะเป็น children ของ context menu ซึ่งจะไม่แสดงบนเมนูของ figure ในการสร้าง paren menu อันดับแรกเลือก Context Menus tab ใต้ Menu Editor จากนั้นเลือก New Context menu



2. เลือก Tag สำหรับ context menu นั้น
3. กำหนดชื่อที่ปรากฏ และกำหนด callback String
4. เราสามารถสร้าง รายการในเมนูนี้ได้เรื่อยๆ ตามต้องการ
5. กำหนดว่า วัตถุใดจะใช้ Context menu นี้ ซึ่งทำได้โดยเลือกวัตถุนั้นบน Layout Editor จากนั้นภายใต้ Property Inspector ของวัตถุนั้น เลือกคุณสมบัติ UI context Menu ให้ตรงกับ Context Menu ที่เราต้องการ
6. สร้าง Callback subfunction ให้กับเมนูนั้นถ้าจำเป็น โดยมีขั้นตอนเหมือนกับการสร้างให้เมนูปกติ

14.5 User Interface Controls

สำหรับ User interface control นั้นจะประกอบด้วย 1) Check Boxes, 2) Editable Text, 3) Frames 4) List boxes, 5) Push Buttons, 6) Radio Buttons, 7) Sliders, 8) Static Text, 9) Toggle Buttons

ซึ่งรายละเอียดของคุณสมบัติเหล่านี้ เราได้กล่าวถึงไปแล้วในบทของ object Properties ที่ผ่านมา ดังนั้นเราจะไม่ขอกล่าวถึงคุณสมบัติ และวิธีการใช้ของมันอีกในบทนี้ เพียงแต่ใน GUIDE ของ MATLAB 6.0 นี้มีการสร้าง Callback subfunction ให้กับวัตถุต่างๆเหล่านี้ (ยกเว้น frame และ Static text) โดยอัตโนมัติ ซึ่งต้องใช้ชื่อ Tag เป็นองค์ประกอบในชื่อ subfunction นั้นด้วย ดังนั้นเราจะขอแสดงชื่อที่ DUIDE ตั้งให้ Control แต่ละตัว เป็น default ให้กับวัตถุเหล่านี้ อย่างไรก็ตามอย่าลืมว่าเราควรจะกำหนดชื่อ Tag เหล่านี้ใหม่ให้เหมาะสม กับการทำงานของมัน

UI Control	Default Tag
Check Boxes	checkbox1, checkbox2,…..
Editable Text	edittext1, …..
Frames	frame1, …..
List Boxes	listbox1, ….
Push Buttons	pushbutton1, ……
Radio Buttons	radiobutton1, …
Slider	slider1, …
Static Text	text1, …
Toggle Buttons	Togglebottom1,…

14.6 Understanding the Application M-File

Application M-file เป็นโครงร่างโปรแกรมที่ใช้ควบคุมการทำงานของ GUI ซึ่งจะถูกสร้างขึ้นโดยอัตโนมัติพร้อมกับ Fig-file เมื่อใช้ GUIDE ในการสร้าง GUI โดยที่ Application M-file จะช่วยให้ มีความสะดวก และรวดเร็วมากขึ้น ซึ่งโปรแกรมหรือ Code ทุกส่วนรวมถึง Subfunction จะรวมอยู่ใน Application M-file โดย Callback ทั้งหมดจะถูกเขียนเป็น Subfunction ซึ่งทำให้วิธีการเขียน Callback ง่ายขึ้นและทำให้สามารถปรับค่าเริ่มต้น หรือทำการ Initialize ให้กับ GUI ของเราได้ หัวข้อที่จะกล่าวต่อไปนี้จะเป็นประโยชน์กับผู้ที่จะทำการเขียน M-file ที่ใช้ควบคุม GUI ไม่ว่าจะสร้างขึ้นเอง หรือจะเป็น application M-file ที่สร้างขึ้นโดย GUIDE

อย่างไรก็ตาม ในเอกสารนี้จะเน้นถึงการพิจารณา Application M-file ที่สร้างขึ้นโดย GUIDE เป็นหลัก และเมื่อผู้อ่านเข้าใจถึงการเขียนคำสั่งต่างๆ ที่ถูกสร้างขึ้นมาแล้วก็จะสามารถสร้าง M-file ขึ้นมาควบคุมการทำงานของ GUI ได้ด้วยตนเอง ส่วนประกอบสำคัญ Application M-file มีดังต่อไปนี้

- การสร้างและกำหนดชื่อ Callback โดยอัตโนมัติ
- ขั้นตอนการทำงานของ Application M-file
- การกำหนดค่าเริ่มต้นใน Application M-file

❶ Automatic Naming of Callback Routines

GUIDE จะทำการสร้างและตั้งชื่อ Callback subfunction ขึ้นมาโดยอัตโนมัติ ให้กับวัตถุต่างๆ ที่สร้างขึ้นมาใน Layout Editor ยกเว้น Static Text และ Frame ดังนั้นเมื่อมีการเลือกวัตถุเหล่านั้นใน GUI จะทำให้ MATLAB มองไปที่ Callback ซึ่งจะชี้ไปที่ Callback subfunction ของแต่ละตัวใน application M-file แล้ว MATLAB จะทำงานตามที่กำหนดใน Subfunction ดังนั้นขั้นตอนการทำงานของ GUIDE และของ GUI ที่สร้างโดย GUIDE จะมีขั้นตอนโดยสรุปดังนี้

1. สร้างรูปแบบ GUI ใน GUIDE Layout Editor และปรับแก้คุณสมบัติต่างๆ ให้เป็นตามต้องการ
2. เมื่อทำการ Save GUI ใน Layout Editor แล้ว GUIE ก็จะทำขั้นตอนต่อไปนี้
   - สร้าง Application M-file และ Fig-file โดยใช้ชื่อเดียวกันกับชื่อ GUI
   - ทำการกำหนดค่า Callback ของวัตถุต่างๆ ให้เป็น
     `my_gui('ObjectTag_Callback',gcbo,[],guidata(gcbo))`

โดย

- **`my_gui`** จะเป็นชื่อของ application M-file ที่เรากำหนดในตอน save
- **`ObjectTag_Callback`** จะเป็นชื่อของ Subfunction ที่จะเกิดขึ้นใน Application M-file โดยชื่อ ObjectTag นั้นจะเปลี่ยนไปตามที่เรากำหนดคุณสมบัติ Tag ของวัตถุนั้นเช่น ถ้าเรากำหนดให้วัตถุนั้นมี Tag ชื่อ pushbutton1 ก็จะทำให้ชื่อของ subfunction นี้มีชื่อเป็น pushbutton1_Callback เป็นต้น ซึ่งจะเป็นการสะดวกที่เราจะทราบว่า subfunction ใดใน application M-file เป็นของวัตถุใดใน GUI นั่นเอง
- **`gcbo`** มาจาก get callback object นั่นคือจะให้ค่าเป็น handle ของวัตถุที่กำลังเรียก Callback จากผู้ใช้
- **`[ ]`** เป็นเมทริกส์ว่าง ที่เก็บสำรองไว้ ซึ่งอาจมีการใช้ในอนาคต
- **`guidata(gcbo)`** เป็นข้อมูลของ handle ของวัตถุทุกตัวที่อยู่ใน GUI ซึ่งจะทำให้เราเรียกหาวัตถุต่างๆที่มีอยู่ใน GUI นั้นได้สะดวกยิ่งขึ้น

เมื่อมีการสร้างวัตถุใดๆ ลงใน GUI แล้ว GUIDE ก็จะทำการสร้าง Subfunction ที่ชื่อ ObjectTag_Callback ขึ้นมาให้ด้วย ดังนั้นเมื่อมีการเรียกวัตถุนั้นใน GUI ก็จะทำให้ MATLAB ไปที่ Application M-file ตามที่เรากำหนด (ในตัวอย่างข้างบนก็จะไปที่ฟังก์ชันไฟล์ my_gui.m) และมองหา Subfunction ที่ชื่อ ObjectTag_CallBack โดยใน application M-file ก็จะปรากฏ Subfuction ในลักษณะต่อไปนี้ขึ้นมา

```
% --------------------------------------------------------------------
function varargout = ObjectTag_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.ObjectTag.
disp('ObjectTag Callback not implemented yet.')
```

ซึ่งสำหรับ input ที่กำหนดให้กับ Subfunction นี้จะมี

h เป็น handle ของวัตถุนี้

eventdata เป็นเมทริกส์ว่างซึ่งสำรองไว้ใช้ในอนาคต

handles จะเป็นตัวแปรประเภท structure โดยจะมี field ต่างๆ เป็น handle ของวัตถุที่มีใน GUI และชื่อของ field จะมีชื่อตาม Tag ของวัตถุนั้น ยกตัวอย่างเช่นถ้า GUI นั้นประกอบด้วยวัตถุอยู่ 3 วัตถุคือ

- หน้าต่างรูปภาพหรือ Figure ซึ่งมีชื่อ Tag เป็น figure1 และ handle มีค่าเป็น 1.00256
- ปุ่ม Push Buttom ซึ่งมีชื่อ Tag เป็น ObjectTag และ handle มีค่าเป็น 5.00123
- มี Slider ซึ่งมีชื่อ Tag เป็น slider1 และ handle มีค่าเป็น 6.00301

ก็จะทำให้ตัวแปร handles ซึ่งเป็นตัวแปรแบบ Structure นี้มี 3 field คือ

- handles.figure1 มีค่า 1.00256
- handles.ObjectTag มีค่า 5.00123
- handles.slider1 มีค่า 6.00301

ดังนั้นจะทำให้เราทราบ handle ของวัตถุทั้งหมดที่มีอยู่ใน GUI จึงทำให้ไม่จำเป็นต้องใช้คำสั่ง findobj เพื่อหาวัตถุเหล่านั้นเหมือนในการเขียน GUI ใน version ก่อนๆ ของ MATLAB

varargin เป็นตัวแปรที่ส่งผ่านเข้าไปสู่ฟังก์ชัน (variable argument input)

ส่วนคำสั่ง

```
disp('ObjectTag Callback not implemented yet.')
```

มีไว้เพื่อเมื่อเรากดที่วัตถุนั้นแล้ว บน Command Window จะแสดงข้อความว่าวัตถุนี้ยังไม่ได้มีการปรับเปลี่ยน Callback และจะทำงานยังไม่ได้ เมื่อเราเข้ามาที่ Subfuction นี้ ก็ให้เราลบบรรทัดนี้ออก แล้วเขียนชุดคำสั่งตามที่เราต้องการ

❷ ขั้นตอนการทำงานและกำหนดค่าเริ่มต้นของ Application –M file

ขั้นตอนการทำงานของ Application M-file จะเกิดขึ้นเมื่อผู้ใช้สั่งชื่อ M-file นี้จาก Command Windows ซึ่งผู้ใช้จะสั่งการได้สองแบบคือ

1. เรียก M-file โดยไม่มีข้อกำหนดใดเพิ่มเติม
2. เรียก M-file โดยมีข้อกำหนดเพิ่มเติม ซึ่งโดยส่วนใหญ่แล้วก็จะเป็นชื่อของ Subfunction และมักจะเป็น callback

ซึ่งทำให้ GUIDE ต้องทำส่วนที่เรียกว่า Switchyard Code ขึ้นมา เพื่อให้การทำงานของ GUI สามารถเกิดขึ้นได้ด้วยวิธีการเรียกใช้ทั้งสองแบบของการเรียกใช้ GUI

The Switchyard Code

เป็นส่วนของโปรแกรมที่เขียนขึ้นมาเพื่อตรวจสอบว่าผู้ใช้ได้เรียกใช้คำสั่งกับ GUI ของเราอย่างไร และจะกำหนดให้เกิดการทำงานตามที่ผู้ใช้ต้องการ ซึ่ง Switchyard Code นี้จะเป็นโปรแกรมที่ถูก GUIDE สร้างขึ้นอยู่ในส่วนต้น ของโปรแกรม และทำหน้าที่กำหนดสภาพเริ่มต้นการทำงานของโปรแกรมด้วยอีกหน้าที่หนึ่ง โดยมีลักษณะดังนี้

```
if nargin == 0 % LAUNCH GUI

      fig = openfig(mfilename,'reuse');

  % Use system color scheme for figure:
      set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));

  % Generate a structure of handles to pass to callbacks, and store it.
      handles = guihandles(fig);
      guidata(fig, handles);

      if nargout > 0
            varargout{1} = fig;
      end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

      try
            [varargout{1:nargout}]=feval(varargin{:});%FEVAL switchyard
      catch
            disp(lasterr);
      end
end
```

โดยเราจะขออธิบายถึงส่วนต่างๆ ใน code บางส่วนดังนี้

`if nargin == 0 % LAUNCH GUI` เริ่มจากคำสั่ง if ซึ่งขั้นแรกจะเป็นการตรวจสอบว่าคำสั่งที่มาจาก Command line นั้นมีข้อความต่อจากชื่อไฟล์ของ GUI ของเราหรือไม่ ถ้าไม่มีคือเท่ากับ 0 ก็จะทำให้ MATLAB ทำการเปิด GUI ขึ้นทันที โดยเมื่อเปิด GUI จะทำการต่อไปนี้

`fig = openfig(filename,'reuse');` เปิดหน้าต่างที่เป็น GUI ของเราขึ้นมา โดยจะเป็นการเปิด FIG-file ที่เก็บไว้นั่นเอง ส่วนข้อกำหนด **`reuse`** นั้นหมายความว่าจะยอมให้มี GUI นี้เกิดขึ้นได้เพียงหน้าต่างเดียวในเวลาหนึ่งๆ (ซึ่งเป็นไปตามข้อกำหนดที่เราเลือก ซึ่งได้กล่าวถึงไปก่อนหน้านี้แล้ว) แต่ถ้าหากว่าเราเลือกให้มี GUI นี้ได้พร้อมกันหลายหน้าต่าง ข้อกำหนดนี้จะเปลี่ยนเป็น **`new`**

`set(fig,'Color',get(0,'defaultUicontrolBackgroundColor'));` คำสั่งนี้จะทำให้ uicontrol หรือวัตถุต่างๆที่ปรากฏอยู่ใน GUI มีสีพื้นเหมือนกับสีพื้นของหน้าต่าง GUI ในส่วนนี้ก็มาจากข้อกำหนดที่เราเลือกในตอนสร้าง GUI ที่กล่าวมาก่อนหน้านี้นั่นเอง

`handles = guihandles(fig);` คำสั่ง **`guihandles`** จะสร้างตัวแปรแบบ structure จากรูปภาพที่สร้างขึ้น โดยให้ field ทั้งหมดในตัวแปรนั้นคือวัตถุต่างๆที่มีอยู่ในรูปภาพที่กำหนด ในตัวอย่างนี้จะสร้างตัวแปรที่มีชื่อว่า **`handles`** สำหรับรูปที่ handle มีค่าเท่ากับ **`fig`** และชื่อ field นี้ก็จะใช้ชื่อ Tag ของวัตถุต่างๆ ทำให้เราได้ตัวแปรโครงสร้าง handles มีลักษณะตามที่ได้กล่าวมาก่อนหน้านี้แล้ว

`guidata(fig, handles)` คำสั่งนี้เป็นคำสั่งที่ให้เก็บค่าตัวแปร **`handles`** เข้าไปเป็นส่วนหนึ่งของรูปที่สร้างขึ้น คำสั่ง **`guidata`** นี้เป็นคำสั่งที่ใช้เก็บข้อมูล หรือดึงข้อมูลที่มีอยู่จากรูปออกมา การเขียนคำสั่งในลักษณะเช่นนี้จะทำให้ MATLAB ทำการเก็บตัวแปรชื่อ **`handles`** นี้เข้าไว้ให้อยู่ในรูป ซึ่งเราสามารถที่จะเรียกข้อมูลจากรูปออกมาได้ในภายหลัง ส่วนสำคัญของคำสั่งนี้ก็คือหากในภายหลังเราได้มีการเพิ่มเติมข้อมูลสำคัญเข้าไปอยู่ใน GUI และต้องการส่งต่อหรือมีการปรับเปลี่ยนค่าตัวแปร **`handles`** นี้ไม่ว่าด้วยเหตุผลใด เราจะต้องใช้คำสั่งนี้ใหม่อีกครั้งหนึ่ง เพื่อให้เก็บข้อมูลที่ได้รับการปรับแก้แล้วเข้าไปอยู่ในรูป

ส่วนในชุดคำสั่งที่มี if หรือ elseif ที่อยู่ต่อมานั้นเป็นคำสั่งที่ในกรณีที่ผู้ใช้เรียกใช้ GUI โดยมีข้อกำหนดเพิ่มเติม ซึ่ง MATLAB ก็จะทำการตรวจสอบข้อกำหนดนั้นว่าเป็นอย่างไร และถ้าหากว่าข้อกำหนดนั้นเป็นชื่อของ Subfunction ที่อยู่ในโปรแกรมนี้ ก็จะสั่งให้ Subfunction นี้ทำงานก่อน ซึ่งเราจะไม่ขอกล่าวถึงรายละเอียดในที่นี้

ต่อไปเราจะลองทำตัวอย่างง่ายๆ GUIDE สร้าง GUI แต่ก่อนอื่น เราขอแนะนำถึงขั้นตอนในการออกแบบและสร้าง GUI ที่มีประสิทธิภาพ ซึ่งถือว่าเป็นแนวทางในการสร้าง GUI โดยทั่วไปก่อน เพื่อให้เกิดความเข้าใจในการสร้างมากยิ่งขึ้น

14.7 ข้อเสนอแนะในการออกแบบ GUI

ถ้าหากว่าท่านคิดที่จะสร้าง GUI ให้ทำงานบน MATLAB หรือโปรแกรมอื่นๆ ก็ตาม ข้อแนะนำประการแรกก็คืออย่าเริ่มทำอะไรกับ MATLAB หรือ GUIDE จนกระทั่งได้วางโครงร่าง ออกแบบ กำหนดการเขียน Callback และหน้าที่ต่างๆ ของ วัตถุที่ใช้ควบคุมทั้งหมดบน GUI เป็นที่เรียบร้อยแล้ว เราอาจจะต้องเสียเวลาในตอนเริ่มต้นสำหรับการเขียนแผนงานและการออกแบบบ้าง แต่หลังจากวางแผนการทำงานเป็นที่เรียบร้อยแล้ว จึงทำการสร้าง GUI ตามที่ออกแบบไว้ซึ่งการวางแผนจะช่วยให้เข้าใจการทำงาน แก้ไข และปรับปรุงการทำงานของ GUI ของเราได้โดยง่าย สำหรับหัวข้อนี้จะเป็นการเสนอแนะวิธีการเขียน GUI อย่างคร่าวๆ และเราจะถือแนวทางการแนะนำที่เป็นกลางที่สุด เพราะเราทราบดีว่าการออกแบบ GUI นั้นมีเรื่องของรสนิยมเข้ามาเกี่ยวข้องอยู่ด้วยเป็นอย่างมาก

ในการออกแบบ GUI โดยทั่วไปมีหลักที่ต้องคำนึงถึงอยู่หลายประการและมีหนังสือมากมายที่เขียนขึ้นสำหรับการออกแบบ GUI โดยเฉพาะ อย่างไรก็ตามเราคงจะไม่กล่าวถึงรายละเอียดและหลักการของการออกแบบ GUI ไว้ในที่นี้ เพียงแต่จะเป็นการให้ข้อเสนอแนะสำหรับผู้ที่จะเริ่มเขียน GUI และจากประสบการณ์ที่ผ่านมาทำให้เชื่อว่า GUI ที่ดีนั้นขึ้นอยู่กับกลุ่มผู้ใช้งานด้วย GUI ที่ดีสำหรับผู้ใช้กลุ่มหนึ่งอาจไม่ใช่ GUI ที่ดีสำหรับผู้ใช้อีกกลุ่มหนึ่งก็ได้ ดังนั้นเราจึงคิดว่าผู้ที่จะสามารถเขียน GUI คือผู้ที่มีประสบการณ์การใช้งาน GUI หลายๆแบบมาก่อนแล้วนำข้อดีและข้อเสียของ GUI แบบต่างๆ มาปรับปรุงให้เหมาะกับสภาพการทำงานของกลุ่มคนแต่ละกลุ่ม

สำหรับหลักการใหญ่ๆของการออกแบบ GUI เพื่อให้กลุ่มผู้ใช้ส่วนใหญ่มีความพึงพอใจ ผู้ออกแบบ GUI ควรจะคำนึงถึงหลักการต่อไปนี้

- ต้องออกแบบให้มีความง่ายในการใช้งาน
- ต้องออกแบบให้มีรูปแบบที่ชัดเจน
- ต้องออกแบบให้มีรูปแบบที่ผู้ใช้งานคุ้นเคย

ซึ่งถ้าหากผู้ใช้ได้ใช้ GUI ตามที่เราได้กล่าวมาแล้วข้างต้นนี้ จะช่วยให้ผู้ใช้สามารถที่จะใช้งาน GUI ได้อย่างรวดเร็วแม้ว่าจะเป็นการใช้งานในครั้งแรก ซึ่งแน่นอนที่สุดเมื่อผู้ใช้สามารถที่จะใช้งานมันได้อย่างรวดเร็ว ก็จะทำให้ผู้ใช้มีความรู้สึกว่า GUI ที่เขากำลังใช้อยู่นี้ได้รับการออกแบบที่ดี สำหรับหลักการทั้งสามมีรายละเอียดคร่าวๆ ดังต่อไปนี้

❶ การออกแบบให้ง่ายต่อการใช้งาน

ในการออกแบบ GUI นั้นเราจำเป็นอย่างยิ่งที่จะต้องคำถึงความง่ายในการใช้งาน เพราะการที่เราออกแบบ GUI ที่ซับซ้อนโดยไม่จำเป็นนั้นจะทำให้ผู้ใช้มีความรู้สึกว่า GUI นี้ยุ่งยากเกินกว่าที่จะสามารถนำมาใช้ได้อย่างมีประสิทธิภาพ และจะเลิกใช้ไปในที่สุด ตัวอย่างของการออกแบบให้ง่ายต่อการใช้งานก็เช่น

พยายามให้มีการ Interactive กับผู้ใช้ให้น้อยที่สุด การที่มีปฏิสัมพันธ์ กับผู้ใช้เกินความจำเป็นนั้นจะทำให้ผู้ใช้มีความรู้สึกว่าต้องตอบคำถามมากมายก่อนที่จะได้สิ่งที่ตนเองต้องการ ดังนั้นการเขียน GUI เราควรจะมีปุ่มต่างๆ ให้น้อยที่สุดและมีเฉพาะปุ่มที่จำเป็นที่จะต้องควบคุมโดยผู้ใช้เท่านั้น

ไม่ควรใช้หน้าต่างๆ หลายๆหน้าต่างในการแสดงผลและรับข้อมูล เพราะการที่มีหน้าต่างหลายหน้าต่าง จะทำให้ผู้ใช้สับสนได้ง่าย ดังนั้นถ้าเป็นไปได้เราควรจะเลือกออกแบบให้ GUI มีหน้าต่างเพียงหน้าต่างเดียว

ในหลายกรณีถ้าเป็นไปได้เราอาจจะเลือกใช้รูปภาพ หรือสัญลักษณ์แทนตัวอักษร ยกตัวอย่างเช่น การใช้ Slider แทนการใช้การป้อนข้อมูลด้วยตัวเลขเป็นต้น ทั้งนี้ต้องขึ้นกับความเหมาะสมของการใช้สัญลักษณ์ แบบนั้นๆ ด้วย

❷ การออกแบบให้มีรูปแบบที่ชัดเจน

การออกแบบให้มีรูปแบบที่จัดเจนหมายความว่าในกรณีที่เราสร้าง GUI ขึ้นมาใช้เป็นลำดับขั้นคือหลังจากการกรอกข้อมูลหน้าต่างแรกเสร็จ ก็จะปรากฏหน้าต่างที่สองขึ้นมา และอาจมีหน้าต่างอื่นๆตามขึ้นมาอีก เป็นต้น การออกแบบแต่ละหน้าต่างควรจะให้มีการกำหนดลักษณะของแถบเมนูที่ใกล้เคียงกัน GUI ควรจะมีสีพื้นที่เหมือนกัน คำอธิบายต่างๆก็ควรจะวางไว้ในตำแหน่งที่ใกล้เคียงกัน ในทุกหน้าต่าง การทำเช่นนี้จะไม่ทำให้ผู้ใช้มีความรู้สึกว่าจะต้องมีการเปลี่ยนหน้าต่างเพื่อใส่ข้อมูลอยู่ตลอดเวลาแม้ว่าในความเป็นจริงแล้วมีการเปลี่ยนหน้าต่างอยู่หลายหน้าต่าง

❸ การออกแบบเพื่อให้ผู้ใช้คุ้นเคย

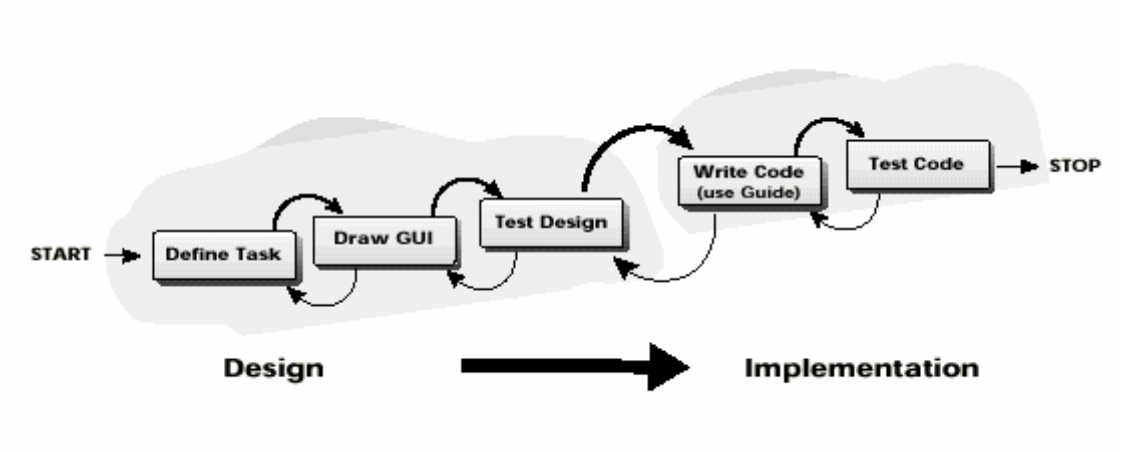
สำหรับผู้ที่ใช้โปรแกรมมานานพอสมควรคงสังเกตเห็นว่าการจัดเรียงรายการบนแถบเมนูของหลายๆโปรแกรมจะคล้ายๆกันแม้ว่าจะมาจากผู้ผลิตที่แตกต่างกัน ยกตัวอย่างเช่น File จะเป็นเมนูแรกและ help จะเป็นเมนูสุดท้ายเสมอ สาเหตุก็เพราะว่าผู้ใช้ส่วนใหญ่มีความคุ้นเคยในรูปแบบดังกล่าว ดังนั้นข้อคิดในที่นี้ก็คือเราอาจไม่มีความจำเป็นที่จะต้องออกแบบสิ่งที่ผู้ใช้คุ้นเคยอยู่แล้วนี้ขึ้นมาใหม่ เพราะผู้ใช้ทุกกลุ่มค้นเคยการใช้งานอย่างนี้อยู่แล้วนั่นเอง อย่างไรก็ตามเราไม่ได้เสนอให้ผู้ออกแบบ GUI จมปรักอยู่ในแนวทางการออกแบบเก่าๆ เสมอไป หากว่าเรามีเหตุผลที่พอเพียงและมีจุดมุ่งหมายที่ชัดเจน เราอาจออกแบบ GUI ที่แหวกแนวออกไปจากที่มีผู้อื่นใช้อยู่ก็ได้ และหลายๆครั้งเราจะเห็นนักออกแบบประเภทนี้ประสพผลสำเร็จได้รับการยอมรับจากผู้ใช้อย่างกว้างขวาง

อย่างไรก็ตามมีข้อสังเกตอยู่ประการหนึ่งซึ่งเราถือว่าเป็นหลักของการออกแบบทุกชนิดไม่ว่าจะเป็นด้านเครื่องจักร เครื่องใช้ไฟฟ้า วงจรการควบคุม หรืออื่นรวมถึงการออกแบบโปรแกรมคอมพิวเตอร์ หลักการนั้นก็คือ

"A simple design is a good design."

14.8 กระบวนการการออกแบบ GUI

ตามที่เราได้กล่าวไว้ก่อนหน้านี้แล้วว่าการออกแบบ GUI ที่ดีจำเป็นจะต้องมีการวางแผนการที่ดี หรือกล่าวอีกอย่างหนึ่งกระบวนการการออกแบบ GUI จะมีความสำคัญต่อ GUI ที่ได้เป็นอย่างมาก และอย่าลืมว่าอย่าเริ่มเขียน GUI จนกว่าคุณจะออกแบบมันเสร็จแล้ว ขั้นตอนการออกแบบ GUI เป็นขั้นตอนแสดงดังในรูปต่อไปนี้



เริ่มจากการวางโครงร่างด้วยแนวความคิดก่อนว่าเราจะทำอะไร

ขั้นแรกอาจเป็นขั้นที่ยากที่สุดของการออกแบบก็ว่าได้ เพราะเป็นขั้นที่จะต้องคิดว่าเราจะออกแบบอะไร และทำงานเพื่อให้สำเร็จจุดประสงค์ใด ในขั้นตอนนี้เราจะต้องวางโครงร่าง หลักการการทำงานและสิ่งต่างๆที่เกี่ยวข้องเพื่อจะประกอบเป็น GUI ของเรา

ลองเขียน GUI เหล่านี้บนกระดาษ

การเขียนแบบลงบนกระดาษ เป็นขั้นตอนที่สำคัญที่สุดที่จำเป็นในการออกแบบทุกชนิด เพราะเราสามารถที่จะปรับปรุงและแก้ไขปัญหาที่อาจเกิดขึ้นได้ในขั้นตอนแรก นอกเหนือจากนั้นทำให้เราเห็นภาพคร่าวๆ ว่าเมื่องานเสร็จสิ้นแล้ว GUI ของเราจะมีรูปร่างหน้าตาออกมาเป็นอย่างไรและขั้นนี้เราสามารถที่จะเริ่มถามความเห็นของผู้อื่นว่า มีความคิดเห็นเกี่ยวกับ GUI ของเราอย่างไรบ้าง ซึ่งสามารถทำให้เราปรับแก้สิ่งต่างๆ ได้อย่างง่ายดายในขั้นนี้

นอกจากรูปร่างที่ปรากฏภายนอกแล้วขั้นนี้เราสามารถที่จะเขียนหน้าที่การทำงานของอุปกรณ์ควบคุมแต่ละส่วนได้ หรือก็คือการเขียน callback ของส่วนควบคุมแต่ละส่วนนั่นเอง ดังนั้นในขั้นตอนนี้

เราจะทราบว่าส่วนควบคุมที่มีอยู่นี้มีอยู่อย่างเพียงพอหรือมากเกินไปหรือไม่ อีกทั้งหน้าที่ของแต่ละส่วนนั้นชัดเจนหรือยังนั่นเอง

การสร้างและทดสอบ

ขั้นต่อไปก็จะเป็นการสร้าง GUI และทดสอบ GUI ที่สร้างขึ้น ในส่วนนี้อาจจะเป็นเรื่องยากใน MATLAB รุ่นก่อนๆ แต่หลังจากที่มี GUIDE การทำเช่นนี้เป็นเรื่องที่ไม่ยากเลย เพราะการวางส่วนควบคุมต่างๆ เป็นเพียงการวางรูปลงไปบน figure windows เท่านั้น ส่วนการเขียน callback ก็จะมี Callback Editor เป็นอุปกรณ์ที่ช่วยในการเขียนคำสั่งเหล่านี้อยู่แล้ว

หลังจากที่เราได้เขียน GUI รวมถึงได้เขียนโปรแกรม calback และอื่นๆที่เกี่ยวข้องเรียบร้อยแล้ว เราจะต้องมีการทดสอบการทำงาน โดยการทดสอบอาจจะทำเอง หรืออาจจะให้เพื่อนๆเราช่วยใช้โปรแกรมเหล่านี้เพื่อจะได้ช่วยเรามองหาข้อผิดพลาดของโปรแกรม ซึ่งมักจะเกิดขึ้นเสมอ การให้ผู้อื่นช่วยทดสอบโปรแกรมจะทำให้เราสามารถมองหาจุดบกพร่องหรือข้อจำกัดของโปรแกรมได้ง่ายขึ้น เพราะผู้สร้างมักจะมีความเคยชินกับสิ่งที่ตัวเองสร้างขึ้นมาและมองข้ามรายละเอียดบางอย่างไปได้

การเผยแพร่

หลังจากที่เราทำการทดสอบส่วนต่างๆเป็นที่พอใจแล้วเราก็สามารถที่จะนำโปรแกรมของเราออกเผยแพร่เพื่อให้ผู้อื่นได้นำไปใช้งาน สำหรับ MATLAB นั้นเราสามารถที่จะเผยแพร่โปรแกรมผ่านส่วนต่างๆ ได้มากมาย เราสามารถที่จะส่ง GUI ที่เราสร้างขึ้นด้วย MATLAB ไปที่ web site เพื่อเผยแพร่ให้กับผู้ใช้ทั่วโลกได้นำไปใช้

14.9 ข้อควรระวัง

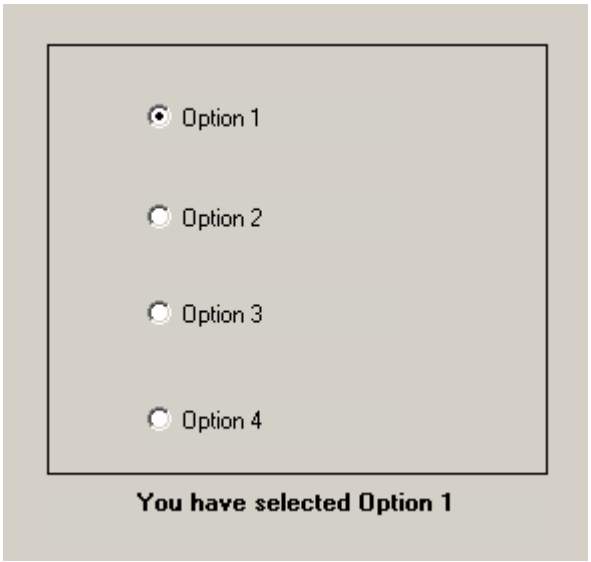
ในการเรียกค่าคุณสมบัติของวัตถุแต่ละชนิดมาใช้ในการควบคุม GUI หรือ นำมาใช้ในการคำนวณ จะต้องเลือกคุณสมบัติให้ถูกต้องและเหมาะสมสำหรับวัตถุนั้น ๆ เนื่องจากคุณสมบัติของวัตถุแต่ละชนิดที่มีชื่อเหมือนกัน แต่นำมาใช้ประโยชน์ต่างกัน เช่น Editable text object คือวัตถุที่ใช้สำหรับให้ผู้ใช้ป้อนค่าที่ต้องการได้และนำค่าที่ป้อนเข้าไปเก็บไว้ที่ คุณสมบัติที่ชื่อว่า String แต่สำหรับ Slider เมื่อผู้ใช้เลื่อนขีดของ Slider แล้วค่าที่ได้จะถูกนำไปเก็บไว้ที่คุณสมบัติที่ชื่อว่า Value ดังนั้นเมื่อเราต้องการนำค่าที่ผู้ใช้กำหนดไปใช้ในการคำนวณ เราจะต้องทราบก่อนว่าวัตถุแต่ละชนิดจะรับค่าไปเก็บไว้ที่คุณสมบัติใดแล้วเรียกค่ามาใช้ให้ถูกต้อง

บทที่ 15 การเริ่มใช้ GUIDE

ในบทนี้เราจะยกตัวอย่างหลายๆ ตัวอย่างที่เกี่ยวข้องกับการใช้งาน GUIDE ในบทที่ผ่านมาเราได้กล่าวถึงขั้นตอนและรายละเอียดการทำงานของ GUIDE ซึ่งบางท่านอาจจะอ่านมาบ้างแล้ว ในบทนี้เราจะเริ่มจากตัวอย่างง่ายๆ ก่อนเพื่อให้เริ่มเกิดความคุ้นเคยและสำหรับตัวอย่างแรกๆนั้นเราจะอธิบายค่อนข้างละเอียดเพื่อให้ผู้ใช้ได้เข้าใจขั้นตอนการทำงานอย่างแท้จริง จากนั้นการอธิบายรายละเอียดในตัวอย่างต่อๆมาก็จะเริ่มสั้นลงเพื่อให้เกิดความกระชับขึ้น เราคาดว่าหลังจากที่ได้อ่านตัวอย่างที่เราอธิบายในที่นี้แล้ว ผู้อ่านจะสามารถนำไปใช้ประโยชน์ในการเขียนโปรแกรม GUI บน MATLAB ได้

15.1 ตัวอย่างการใช้ Radio Button

ต่อไปเราจะลองสร้าง GUI แบบง่ายๆขึ้นโดยใช้ GUIDE เป็นเครื่องมือช่วย ซึ่งตัวอย่างหลายๆ ตัวอย่างต่อไปนี้จะช่วยให้เราเข้าใจการทำงานและวิธีการแก้ไข application M-file ได้ดียิ่งขึ้น ในตัวอย่างนี้จะเป็นการสร้าง GUI ที่ประกอบด้วย Radio Button หลายๆ ตัวเพื่อให้ผู้ใช้ได้ทำการเลือก จากนั้นเราจะแสดงผลการเลือกให้ผู้ใช้ได้ทราบ ลักษณะของ GUI จะมีลักษณะโดยโครงร่างดังนี้



นั่นคือเราจะมีตัวเลือก 4 ตัวเลือก ให้ผู้ใช้เลือก และเมื่อผู้ใช้เลือกแล้ว จะมีข้อความบอกว่าผู้ใช้เลือกตัวเลือกใด และตัวเลือกเหล่านี้จะเลือกได้เพียงตัวเลือกเดียว เพราะปกติการเลือกที่เลือกได้ตัวเลือก

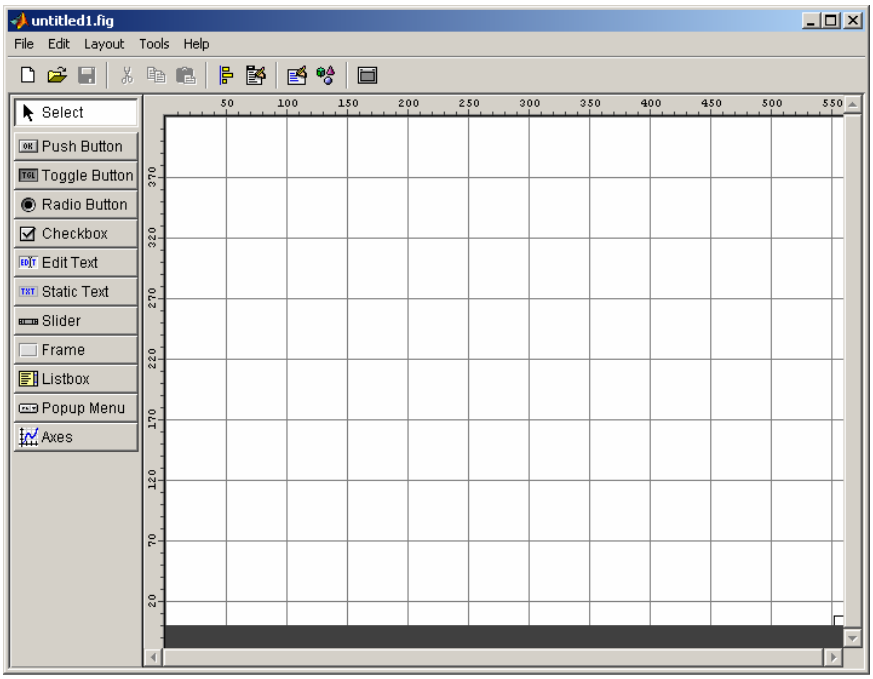
เดียวเราจะนิยมใช้ radio button แต่ถ้าต้องการให้สามารถเลือกได้หลายตัวเลือก เราจะนิยมใช้ check box ซึ่งเราก็คุ้นเคยกับตัวเลือกประเภทนี้ดีอยู่แล้ว ขั้นตอนการสร้าง GUI มีดังต่อไปนี้

วางรูปแบบใน GUIDE Layout Editor

ขั้นตอนแรกนี้เราจะเรียก GUIDE ขึ้นมาใช้งาน ใน Command Windows โดยใช้คำสั่ง

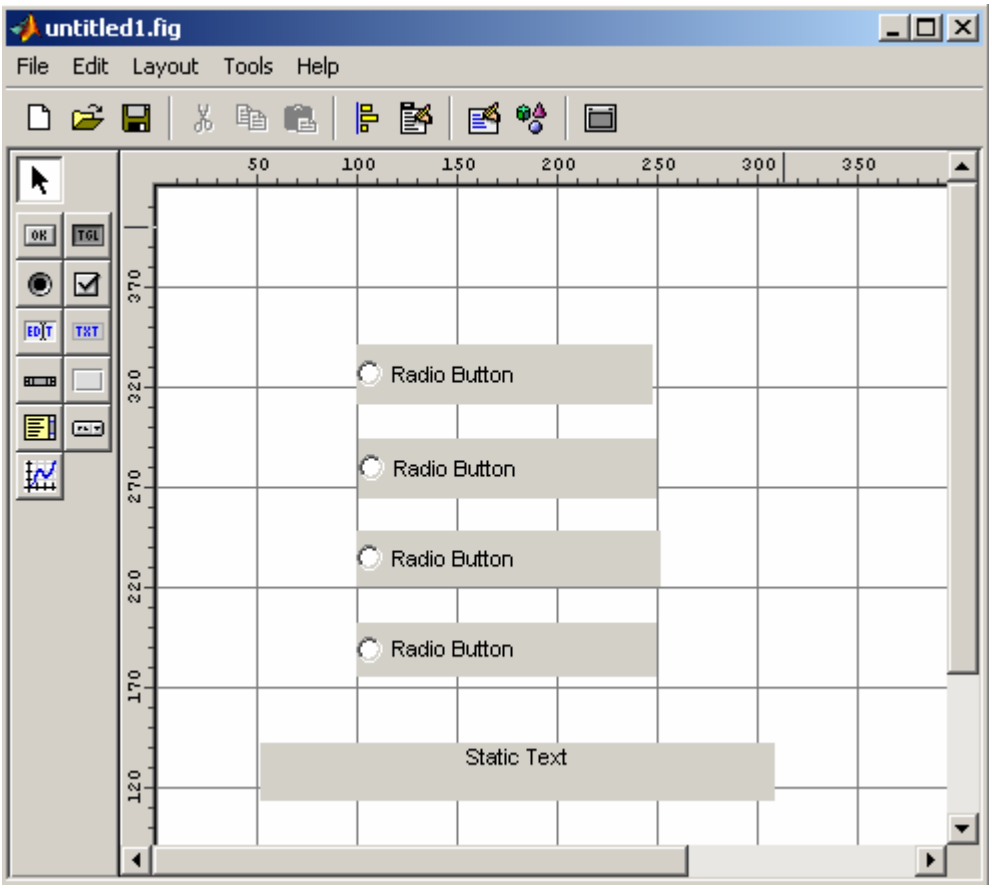
```
»GUIDE
```

ซึ่งเราก็จะได้หน้าต่างของ GUIDE Layout Editor ปรากฏขึ้นลักษณะดังรูป



ขั้นแรกเราควรจะกำหนดตัวเลือกต่างๆ ขึ้นมาก่อนโดยเลือก Allocation Option จากเมนู Tool แล้วเลือกตัวเลือกต่างๆ ตามที่เราต้องการ สำหรับในกรณีนี้เราจะไม่ปรับค่าใดๆ ที่ MATLAB ตั้งเป็นค่าเริ่มต้นมาให้

จากนั้นวางวัตถุ uicontrol ตามที่กำหนดลงไปใน GUI โดยเริ่มจากการเลือก Radio Button มาวางลงไป 4 อัน เลือก Radio Button ใน Tool Pallet ทางด้านซ้ายมือจากนั้นนำเมาส์เข้ามาในเนื้อที่หน้าต่างด้านขวามือ กดเมาส์ปุ่มซ้ายเพื่อกำหนดขอบบนซ้ายของวัตถุ กดเมาส์ค้างแล้วเลื่อนเมาส์ไปเพื่อเลือกขนาดของวัตถุ และปล่อยที่ตำแหน่งที่ต้องการให้เป็นมุมขวาล่างของวัตถุ สร้าง Radio Button ทีละอันจนครบทั้ง 4 อัน และจัดลำดับการวางโดยอันแรกให้อยู่ด้านบนสุด และอันต่อมาก็วางใต้อันแรกลดลั่นกันลงมา เพื่อความสะดวกในการกำหนดคุณสมบัติต่อไปสำหรับขนาด และการวางนั้นเรายังไม่ต้องสนใจมากในขณะนี้ก็ได้ เพราะเราสามารถปรับตำแหน่งและขนาดได้ในภายหลัง และทำนองเดียวกันกับ Static Text ซึ่งจะทำให้เราได้รูปแบบคร่าวๆของ GUI ในขณะนี้เป็น



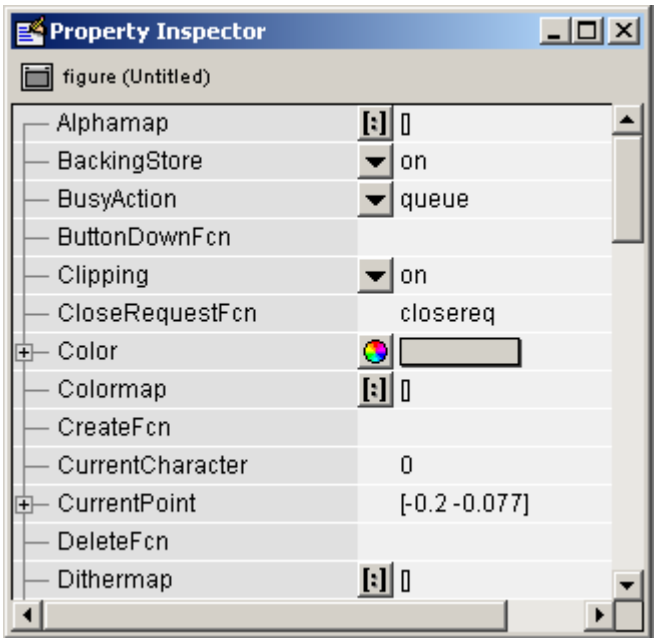
ขั้นต่อไปเพื่อความสวยงามเราอาจจะเพิ่ม Frame เพื่อจัดหมวดหมู่ใน GUI ของเรา เลือก Frame จาก Tool Pallet แล้วนำมาวางลงบน Radio Button ทั้งสี่ แต่เนื่องจากเราวาด Frame ขึ้นที่หลัง Radio Button จึงทำให้ Frame นั้นวางอยู่ข้างบนซึ่งทำให้มองไม่เห็น Radio Button ทั้งสี่

เราสามารถแก้ไขได้โดยใช้เมาส์ปุ่มขวากดลงไปที่ Frame จากนั้นจะปรากฏ Context Menu ขึ้น เราเลือก **Send to Back** เพื่อส่ง Frame นี้ไปอยู่ด้านหลังของ Radio Button ซึ่งเราก็จะได้รูปตามต้องการ

สุดท้ายของการจัดวัตถุบน GUI เราอาจจะใช้ Alignment Tool เพื่อช่วยในการจัดเรียงวัตถุเหล่านั้นให้มีระเบียบยิ่งขึ้นก็ได้

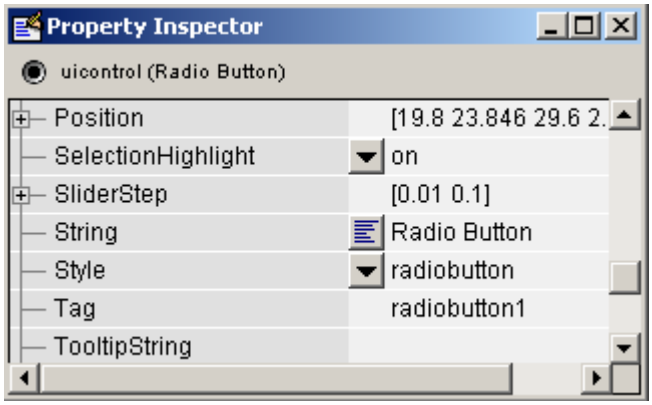
กำหนดชื่อวัตถุโดย Tag และข้อความที่จะปรากฏขึ้น

ต่อไปจะเป็นการกำหนดคุณสมบัติของวัตถุต่างๆ ขั้นแรกให้เราเปิด Property Inspector ขึ้นมาโดยการกดเมาส์ที่ปุ่ม  ซึ่งเป็นการเปิดหน้าต่าง Property Inspector ขึ้นมาซึ่งมีลักษณะดังรูป



ซึ่งเราจะเห็นจากรูปว่าในขณะนี้เป็นการแสดงคุณสมบัติของ Figure อยู่ และเมื่อเราเปลี่ยนตัวเลือก คือใช้เมาส์เลือกวัตถุใน Layout Editor หน้าต่าง Property Inspector ก็จะแสดงคุณสมบัติของวัตถุที่เราเลือก

อันดับแรกให้เราเลือกที่ Radio Button อันบนสุด ซึ่งเป็นอันแรกที่เรานำมาวางไว้ในรูป จากนั้นพิจารณาคุณสมบัติใน Property Inspector เราจะพบว่ามีคุณสมบัติ Tag และ String ดังรูป



ซึ่งคุณสมบัติ Tag จะมีค่าเป็น radiobutton1 และมีค่า String เป็น Radio Button ซึ่งเป็นค่า default ของวัตถุนี้ เราจะสามารถเปลี่ยนคุณสมบัติดังกล่าวได้ โดยการใช้เมาส์ไปกดลงที่ค่าคุณสมบัติเหล่านั้น ในกรณีนี้ให้เราเปลี่ยนคุณสมบัติ Tag ให้เป็น Option1 และเปลี่ยน String เป็น Option 1 ซึ่งเราจะสังเกตเห็นว่าเมื่อเราเปลี่ยนค่า String ไปแล้ว ข้อความที่ปรากฏบน Radio Button นี้จะเปลี่ยนไปเป็น Option 1

จากนั้นให้เราเปลี่ยนค่าคุณสมบัติ Tag และ String ของ Radio Button ที่เหลือให้เป็น Option2, Option3, Option4 และ Option 2, Option 3, Option 4 ตามลำดับ

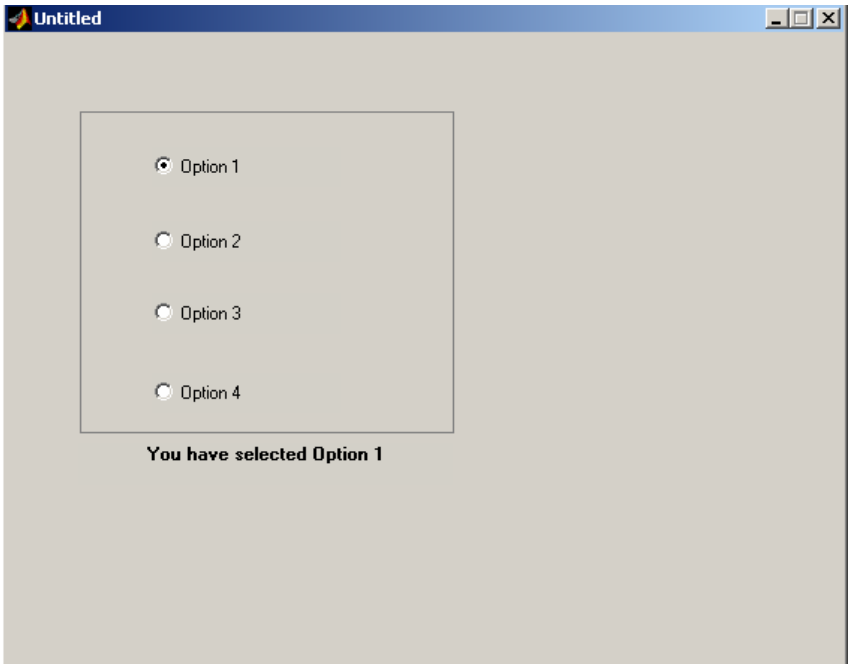
ขั้นต่อไปให้เราเปลี่ยนคุณสมบัติ Value ของ Radio Button Option 1 จากค่าเดิมที่เป็น 0.0 ให้เป็น 1.0 ในส่วนนี้ก็เพื่อกำหนดว่าตัวเลือกนี้เป็นตัวเลือก default นั่นเอง

จากนั้นให้เลือก Static Text แล้วดูคุณสมบัติ String ซึ่งจะมีค่า Default เป็น Static Text ให้เราเปลี่ยนค่านี้เป็น You have select Option 1 เพื่อให้เป็นไปตาม default ที่เราจะกำหนดให้ผู้ใช้

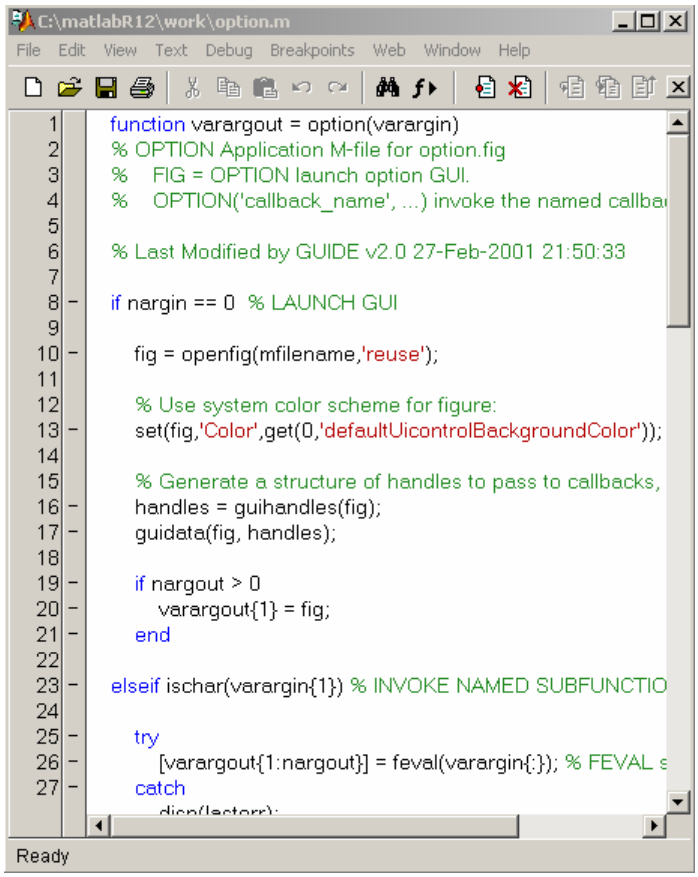
Save และ Activate GUI

เมื่อเราปรับปรุงคุณสมบัติของวัตถุต่างๆ ตามที่เราต้องการเรียบร้อยแล้วให้ทำการ Save GUI โดยเลือก Save จากเมนู File หรือใช้เมาส์กดที่ปุ่ม  ก็ได้ จากนั้น MATLAB จะถามชื่อไฟล์ที่ต้องการเก็บ ซึ่งในที่นี้จะตั้งชื่อว่า option แล้ว Click OK จากนั้น MATLAB ก็จะทำการเก็บไฟล์เป็นสองไฟล์คือ ไฟล์ข้อมูลที่เกี่ยวกับการวางรูปแบบ วัตถุและคุณสมบัติของวัตถุต่างๆ จะเก็บไว้ในไฟล์ที่ชื่อ option.fig และ Application M-file ที่ชื่อ option.m

จากนั้น MATLAB ก็จะเปิดหน้าต่างของ GUI ของเราขึ้นมา ดังรูป



และ Editor/Debugger ก็จะถูกเปิดขึ้นมาพร้อมกับเปิดไฟล์ option.m ดังแสดงในรูป



ตอนนี้ถ้าหากว่าเราลองเลือกปุ่ม Option อื่นๆ ใน GUI จะมีข้อความปรากฏที่ Command Windows ดังนี้

```
Option3 Callback not implemented yet.
```

สาเหตุก็เพราะเรายังไม่ได้ทำการปรับปรุง callback ซึ่งจะเป็น subfunction ที่อยู่ในไฟล์ option.m

ปรับปรุง และเพิ่มเติมชุดคำสั่งให้ GUI ทำงานตามที่ต้องการ

ในการปรับปรุงการทำงานของโปรแกรมนี้จะมีขั้นตอนหลายขั้นตอน แต่ในขั้นตอนแรกเราจะมองในภาพรวมของ Radio Button ทั้งสี่ก่อน เมื่อเราต้องการให้ผู้ใช้เลือกสามารถเลือกตัวเลือกได้เพียงครั้งละหนึ่งตัวเลือกเท่านั้น ดังนั้นหากผู้ใช้เลือก Option 2 ตัวเลือก Option 1 ที่เลือกไว้เดิมจะต้องถูกปิดโดยอัตโนมัติ และก็จะเป็นทำนองเดียวกันกับตัวเลือกอื่นๆ ดังนั้นเราควรจะเขียนฟังก์ชันเป็น subfunction เพิ่มเติมขึ้นมาสำหรับกรณีเช่นนี้ โดยเขียนไว้ที่ส่วนท้ายของโปรแกรม มีลักษณะดังนี้

```
% --------------------------------------------------------
function Turn_Off(off)
set(off,'Value',0);
% --------------------------------------------------------
```

Subfunction นี้ชื่อ Turn_Off จะรับข้อกำหนดเป็นค่า off จากการเรียกใช้คำสั่ง ซึ่งในที่นี้ off จะเป็น vector ของ handleของตัวเลือกต่างๆ ที่เราต้องการปิด จากนั้นทำการตั้งค่าคุณสมบัติ value ของ handle ต่างๆ ที่ส่งมากับข้อกำหนด off นี้ให้มีค่าเป็นศูนย์ ซึ่งก็หมายถึง radio button ตัวนั้นไม่ได้เลือกหรือถูกปิดนั่นเอง ในส่วนการเขียน subfunction เพื่อปิดตัวเลือกต่างๆ ก็จะเสร็จสิ้นลงเท่านี้

ขั้นต่อไปเราจะทำการเขียน subfunction ให้กับ callback ของ uicontrol ต่างๆ อันดับแรกให้ไปที่ subfunction ของตัวเลือก Option 1 นั่นคือ

```
function varargout=Option1_Callback(h,eventdata,handles,varargin)
% Stub for Callback of the uicontrol handles.Option1.
disp('Option1 Callback not implemented yet.')
```

ขั้นแรกให้ทำการลบบรรทัด **disp('...')** นี้ออกไปก่อน เราไม่จำเป็นต้องใช้อีกแล้ว จากนั้นพิมพ์ชุดคำสั่งต่อไปนี้ลงไปแทน

```
off =[handles.Option2 handles.Option3 handles.Option4];
turn_Off(off);
set(h,'Value',1);
set(handles.text1,'String','You have selected Option 1');
```

คำสั่งกำหนดค่า off ในบรรทัดแรก จะเป็นการสร้าง vector บรรจุค่าของ handle ของ Radio Button อีกสามอันที่เหลือที่ผู้ใช้ไม่ได้เลือก ในบรรทัดที่สองเราจะส่งค่า off นี้ไปให้ function Turn_Off ทำการเปลี่ยนค่า Value ของปุ่มทั้งสามนี้ให้เป็นศูนย์ สำหรับบรรทัดที่ 3 เรากำหนดให้ค่า Value ของวัตถุนี้ (h เป็นค่า handle ของวัตถุที่ถูกเรียก Callback) มีค่าเป็น 1 เพื่อป้องกันผู้ใช้กดปุ่มตัวเลือกที่เลือกอยู่แล้วซ้ำค่าจะได้ไม่กลับเป็นศูนย์ (การทำงานโดยปกติของ Radio Button นี้ถ้าผู้ใช้กดปุ่มเลือกขณะที่มันถูกเลือก (value = 1) อยู่เดิมแล้ว ค่าจะกลับเป็นไม่มีการเลือก(value = 0) และถ้ามีค่าเป็นไม่มีการเลือก (value = 0) อยู่ก่อนเมื่อผู้ใช้กดปุ่มเลือกก็จะเปลี่ยนเป็นมีการเลือก (value = 1)

ส่วนในบรรทัดสุดท้ายจะเป็นการกำหนดค่า String ของ Static Text ซึ่งมีค่า handles เป็น handles.text1 ให้มีค่าเป็น You have selected Option 1 ซึ่งเป็นการแสดงผลให้ผู้ใช้ทราบว่าเลือกตัวเลือกที่ 1

ในทำนองเดียวกันให้เราเปลี่ยน subfunction ที่เหลือให้มีลักษณะดังนี้

```
% --------------------------------------------------------------
function varargout=Option2_Callback(h,eventdata,handles,varargin)
% Stub for Callback of the uicontrol handles.Option2.
off =[handles.Option1 handles.Option3 handles.Option4];
Turn_Off(off);
set(h,'Value',1);
set(handles.text1,'String','You have selected Option 2');


% --------------------------------------------------------------
function varargout=Option3_Callback(h,eventdata,handles,varargin)
%  Stub for Callback of the uicontrol handles.Option3.
```

```
off =[handles.Option1 handles.Option2 handles.Option4];
Turn_Off(off);
set(h,'Value',1);
set(handles.text1,'String','You have selected Option 3');

% --------------------------------------------------------------------
function varargout=Option4_Callback(h,eventdata,handles,varargin)
% Stub for Callback of the uicontrol handles.Option4.
off =[handles.Option1 handles.Option2 handles.Option3];
Turn_Off(off);
set(h,'Value',1);
set(handles.text1,'String','You have selected Option 4');
```

ซึ่งเราจะเห็นว่ามีข้อแตกต่างในรายละเอียดของ Subfunction ต่างๆก็เพียงค่าตัวแปร off ที่จะทำการเปลี่ยนตัวเลือกอื่นให้มี Value = 0 และข้อความว่าขณะนี้ผู้ใช้ได้เลือกตัวเลือกใด เมื่อได้แก้ไขและเขียนคำสั่งให้กับ GUI เรียบร้อยแล้ว ให้ Save ไฟล์นี้ก่อนที่จะใช้งานมันต่อไป

ทดสอบการทำงานของ GUI

หลังจากที่เราเขียนคำสั่งขึ้นมาเรียบร้อยแล้ว ขั้นต่อไปจะเป็นการทดสอบ GUI ของเรา ถ้าหน้าต่าง GUI ปิดอยู่เราสามารถจะเปิดหน้าต่างขึ้นมาใหม่ได้ โดยที่ไปที่ Command Windows แล้วสั่ง

```
»option
```

จากนั้นลองทดสอบการทำงานของ GUI ของเราดู ซึ่งถ้าหากทำตามขั้นตอนที่กล่าวมาแล้วอย่างครบถ้วน ก็ไม่น่าจะมีปัญหาอะไร

ปรับปรุงข้อผิดพลาด

ในส่วนนี้เป็นขั้นตอนปกติของการเขียนโปรแกรม ซึ่งอาจมีข้อผิดพลาด หรือเราอาจจะยังไม่ใส่ใจในรายละเอียดในตอนแรก เมื่อได้ทดลองใช้แล้วอาจจะต้องมีการปรับปรุงการทำงานให้ดียิ่งขึ้น หรือรูปแบบของ GUI ให้มีความสวยงามและเหมาะสมยิ่งขึ้น

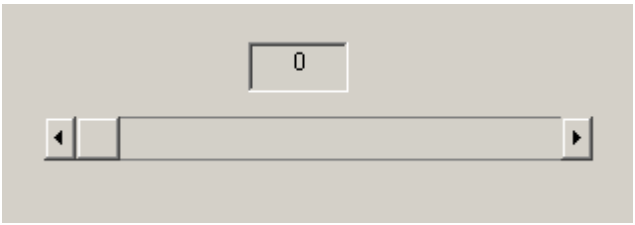
ถ้าเราดูที่หน้าต่าง GUI ของเราในตอนนี้เราจะพบว่าที่ title นั้นจะเป็นชื่อ Untitled อยู่เพราะยังไม่ได้กำหนดชื่อของ GUI ที่จะให้ปรากฏในระหว่างใช้งาน ซึ่งเราก็สามารถที่จะแก้ไขได้ โดยไปที่ Layout Editor เลือก figure คือกดเมาส์ลงในที่ว่างตรงใดก็ได้ จากนั้นไปที่ Property Inspector แล้วเปลี่ยนค่าคุณสมบัติ Name จาก Untitled ให้เป็น My First GUI: Option (หรือจะเป็นชื่ออื่นก็ได้ ตามที่เราต้องการ) จากนั้นสั่ง Save ที่ Layout Editor อย่างไรก็ดีถ้าเรามีหน้าต่างเก่าเราอยู่ เราจะเห็นว่าชื่อนั้นยังไม่เปลี่ยนแปลง สาเหตุเพราะเรายังไม่ได้มีการ update ข้อมูลส่งไปให้ GUI

เพื่อเป็นการ update ข้อมูลให้กับ GUI ของเราให้เราสั่ง Activate GUI ของเราอีกครั้งหนึ่งก่อน ซึ่งก็จะทำให้เราเห็นการเปลี่ยนแปลงที่เกิดขึ้นกับ GUI ของเรา

ตัวอย่างที่ผ่านมานี้ แสดงขั้นตอนการเขียน GUI ให้เราได้ทำความเข้าใจ แม้ว่า GUI ที่ยกตัวอย่างนี้จะเป็นตัวอย่างง่ายๆ แต่ขั้นตอนการเขียน GUI นี้สามารถนำไปใช้กับการเขียน GUI ได้แทบทุกระดับความยาก เพราะความยุ่งยากจริงๆ อยู่ที่การเขียน Subfunction ซึ่งเปรียบเสมือนเป็น Callback Function นั่นเอง ความยากง่ายก็มักจะขึ้นอยู่กับขั้นตอนการคำนวณ การกำหนดคุณสมบัติ การปรับปรุงค่าคุณสมบัติของวัตถุ และการส่งผ่านข้อมูลเหมือนกับการเขียน Function-file ทั่วๆไปนั่นเอง

15.2 ตัวอย่าง Slider และ Editable Text

ตัวอย่างนี้เราจะลองใช้ uicontrol ประเภทอื่นดูบ้าง ซึ่งเราจะสร้าง GUI ที่มีลักษณะดังรูป



คุณสมบัติคร่าวๆ ของ Slider ก็คือจะเป็นมีค่าคุณสมบัติ Value สัมพันธ์กับระยะเลื่อนของตัวเลื่อนบน Slider โดยถ้าตัวเลื่อนอยู่ที่ตำแหน่ง ซ้ายสุดก็จะให้ค่าต่ำสุดและเมื่อตัวเลื่อนอยู่ที่ตำแหน่งขวาสุดก็จะให้ค่าสูงสุด และค่าระหว่างนั้นจะเป็นการเปลี่ยนแปลงเชิงเส้นจากค่าน้อยสุดถึงมากที่สุดสัมพันธ์กับระยะเลื่อนของตัวเลื่อน ค่าต่ำสุดและสูงสุดจะถูกกำหนดโดยคุณสมบัติ Max และ Min ตามลำดับ ส่วนค่าที่ได้ในขณะนั้นจะแสดงทางคุณสมบัติ Value

สำหรับ Editable Text นั้นเป็นการแสดงผลที่เป็นตัวหนังสือที่ผู้ใช้สามารถกำหนดลงไปได้ ค่าที่แสดงผลนั้นเป็นไปตามคุณสมบัติ String ของ Editable Text ดังนั้นไม่ว่าตัวอักษรหรือตัวเลขที่แสดงอยู่นั้นจะเป็นตัวแปรประเภท String ไม่ใช่ตัวแปร Numeric

ในตัวอย่างนี้เราจะมีข้อกำหนดในการทำงาน GUI ดังนี้คือ

- ถ้าเราปรับค่าที่ Slider จะทำให้ค่าตัวเลขที่แสดงใน Editable Text นี้เปลี่ยนไปตามค่าของ slider ที่เปลี่ยนไป
- ถ้าเราปรับค่าตัวเลขที่ Editable Text จะทำให้ค่าที่แสดงใน Slider นี้เปลี่ยนไปตามค่าของที่ผู้ใช้ใส่ไป
- ค่าสูงสุดและต่ำสุดของ Slider นี้จะเป็น 10 และ 0 ตามลำดับ ถ้าหากว่าผู้ใช้กำหนดค่าทาง Editable Text ให้มีค่ามากกว่าหรือน้อยกว่าค่านี้ ค่าจะปรับเป็นค่าสูงสุดหรือต่ำสุดโดยอัตโนมัติ

วางรูปแบบใน GUIDE Layout Editor

ขั้นตอนแรกนี้เราจะเรียก GUIDE ขึ้นมาใช้งาน ดังนั้นใน Command Windows ให้เราสั่ง

```
»GUIDE
```

ซึ่งเราก็จะได้หน้าต่างของ Layout Editor ปรากฏขึ้น จากนั้นวาง uicontrol ทั้งสองตามรูปที่แสดงข้างบนคือ วาง Editable Text ไว้เหนือ Slider ขั้นตอนการวางจะเหมือนที่กล่าวมาแล้วในตัวอย่างที่ผ่านมา

กำหนดชื่อวัตถุโดย Tag และข้อความที่จะปรากฏขึ้น

ขั้นต่อไปจะเป็นการกำหนดคุณสมบัติของวัตถุต่างๆ ขั้นแรกให้เราเปิด Property Inspector ขึ้นมา โดยการกดเมาส์ที่ปุ่ม  ซึ่งเป็นการเปิดหน้าต่าง Property Inspector ขึ้นมา จากนั้นให้ดูคุณสมบัติ Tag ของวัตถุทั้งสอง โดย Editable Text จะมี Tag เป็น edit1 และ Slider จะมี Tag เป็น slider1 เราจะใช้ค่านี้ในการเขียนคำสั่งโดยไม่เปลี่ยนแปลง

คุณสมบัติที่เราจะต้องทำการเปลี่ยนแปลงในที่นี้อันแรกคือค่าต่ำสุดของ Slider เพราะโดย Default แล้วค่าต่ำสุดจะเป็น 0 ส่วนค่าสูงสุดจะเป็น 1 ดังนั้นในที่นี้เราจะเปลี่ยนค่าคุณสมบัติ Max ให้เป็น 10 และให้แน่ใจว่าค่า Value ของ Slider นั้นมีค่าเท่ากับ 0 ซึ่งเป็น default

คุณสมบัติของ Editable Text ที่เราจะเปลี่ยนก็คือค่า String ซึ่งเราจะเปลี่ยนจากค่าเดิมให้เป็นตัวเลข 0 ซึ่งต้องทำความเข้าใจก่อนว่าค่าคุณสมบัติตัวนี้จะเป็น string ดังนั้นจะไม่มีค่าในเชิงคณิตศาสตร์ นั่นคือเราจะนำไปกำหนดค่าตัวอื่นไม่ได้ ตัวเลข 0 ในที่นี้ไม่ได้มีผลทางคำนวณแตกต่างจากตัวอักษร a เลย

Save และ Activate GUI

ขั้นต่อไปให้เรา Save GUI ของเราโดยเลือก Save จากเมนู File หรือใช้เมาส์กดที่ปุ่ม  ก็ได้ จากนั้น MATLAB จะถามชื่อไฟล์ที่เราต้องการเก็บ ซึ่งในที่นี้เราจะตั้งชื่อว่า numerical ซึ่งเมื่อเราเลือก OK แล้ว MATLAB ก็จะทำการเก็บเป็นสองไฟล์คือ numerical.fig และ numerical.m

เมื่อทำการ Activate GUI จะทำให้ MATLAB เปิดหน้าต่างของ GUI ของเราขึ้นมา และ Editor/Debugger ก็จะถูกเปิดขึ้นมาพร้อมกับเปิดไฟล์ numerical.m ขึ้น

ปรับปรุง และเพิ่มเติมชุดคำสั่งให้ GUI ทำงานตามที่ต้องการ

เนื่องจากค่าที่ได้จาก Slider มีค่าเป็นตัวเลข ส่วนค่าที่ได้จาก Editable Text มีค่าเป็น String ทำให้เราไม่สามารถที่จะใช้ค่าของ Editable Text กับ Slider ได้โดยตรง จะต้องทำการเปลี่ยนคุณสมบัติก่อน ซึ่งจะทำได้โดยอาศัยฟังก์ชัน 2 ฟังก์ชันคือ

- **`str2num(X)`** ทำการเปลี่ยน String X ให้มีค่าเป็นตัวเลข ในกรณีที่ X ไม่ใช่ตัวอักษณที่เป็นตัวเลข เราจะได้ค่าว่าง [ ] ออกมา

- **num2str(X)** ทำการเปลี่ยนตัวเลข X ให้มีค่าเป็น String

ดังนั้นสำหรับ Slider ซึ่งมี callback ชื่อ slider1_Callback จะมีลักษณะของคำสั่งเป็น

```
nvalue = get(h,'Value');
svalue = num2str(nvalue);
set(handles.edit1,'String',svalue);
```

โดยบรรทัดแรกจะเป็นการอ่านค่าคุณสมบัติ Value ซึ่งได้ค่ามาเป็นตัวเลขเก็บไว้ในตัวแปรชื่อ nvalue ในบรรทัดที่สองจะเปลี่ยนค่า nvalue ที่เป็นตัวเลขนี้ให้มีค่าเป็น String และในบรรทัดสุดท้ายให้ปรับค่าคุณสมบัติ String ของ Editable Text ซึ่งมี Tag เป็น edit1 ให้มีค่าเท่ากับ svalue ในที่นี้เราไม่ต้องกังวลเรื่องค่าต่ำสุดหรือสูงสุดเพราะผู้ใช้ไม่สามารถกำหนดค่าทาง slider ให้สูงกว่าหรือต่ำกว่าค่าสูงสุดหรือต่ำสุดของ slider ได้อยู่แล้ว

สำหรับ Editable Text ซึ่งมี callback ชื่อ edit1_Callback จะมีลักษณะคำสั่งดังนี้

```
svalue = get(h,'String');
nvalue = str2num(svalue)
if nvalue > 10
    nvalue = 10;
    set(h,'String','10');
elseif nvalue < 0
    nvalue = 0
    set(h,'String','0');
elseif nvalue == [ ]
    nvalue = 0
    set(h,'String','0');
end

set(handles.slider1,'Value', nvalue);
```

ในบรรทัดแรกจะเป็นการอ่านค่าคุณสมบัติ String จาก Editable Text แล้วในบรรทัดที่สองก็จะเป็นการเปลี่ยนค่า String ที่กำหนดให้เป็นตัวเลขเก็บค่าไว้ในตัวแปร **nvalue** ส่วนในประโยค if ถัดมาเป็นการตรวจสอบว่า **nvalue** มากกว่าหรือน้อยกว่าขีดจำกัดหรือไม่ ถ้ามากกว่าก็จะปรับค่าให้เท่ากับค่าสูงสุดแล้วกำหนดคุณสมบัติ String ของ Editable Text นี้ให้เป็นค่าสูงสุดด้วย (ค่า '10' เป็นการกำหนด String) และในส่วนที่เหลือก็จะมีลักษณะในทำนองเดียวกันคือเป็นการปรับค่าให้อยู่ในช่วงที่กำหนด และในบรรทัดสุดท้ายก็จะเป็นการกำหนดค่า Value ให้กับ Slider ตามที่ผู้ใช้ป้อนผ่านทาง Editable Text

ทดสอบการทำงานของ GUI

ขั้นสุดท้ายเป็นการทดสอบการทำงานของ GUI ให้เราทำการ save file ที่ Editor/Debugger เสียก่อน จากนั้น Activate GUI ของเรา แล้วทดสอบการทำงาน ถ้าทำตามขั้นตอนที่กล่าวมาแล้ว GUI ไม่น่าจะมีปัญหาในการทำงาน

15.3 ตัวอย่าง Check Box, List Box และ Popup Menu

ตัวอย่างต่อไปเราจะใช้ Check Box, List Box และ Popup Menu ในการสร้าง GUI ที่ต้องการให้ผู้ใช้เลือกตัวเลือกผ่านอุปกรณ์เหล่านี้ อันดับแรกจะเป็นการอธิบายคุณสมบัติของ Check Box, List Box และ Popup Menu ดังนี้

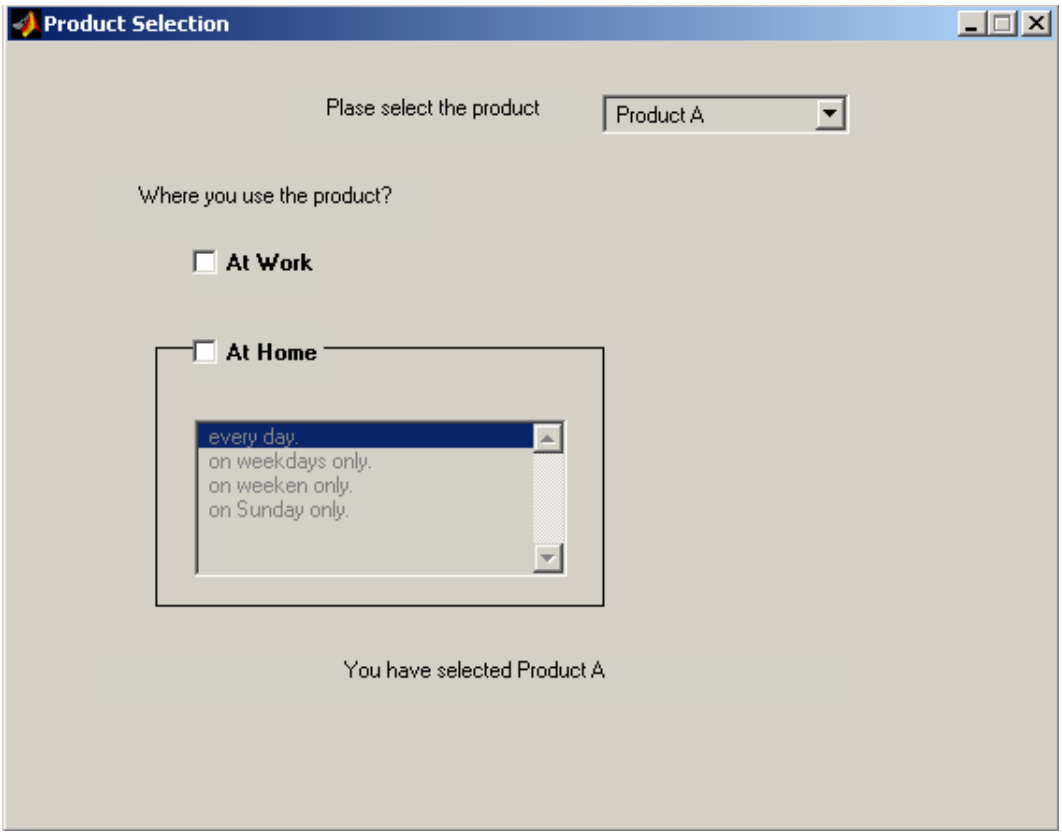
Check Box จะพิจารณาค่าคุณสมบัติ Value เพื่อแสดงว่าผู้ใช้เลือกหรือไม่ โดยเมื่อผู้ใช้เลือกจะมีค่า Value เป็น 1 และเมื่อไม่เลือกจะมีค่า Value เป็น 0

List Box จะพิจารณาค่าคุณสมบัติ Value ตามลำดับการเลือกตัวเลือกต่างๆ โดยถ้าผู้ใช้เลือกตัวเลือกที่ 1 ก็จะมีค่า Value เป็น 1 ถ้าผู้ใช้เลือกตัวเลือกที่ 2 ก็จะมีค่า Value เป็น 2 เป็นลำดับเช่นนี้เรื่อยไป การกำหนดตัวเลือกเหล่านั้นในคุณสมบัติ String ของ List Box นี้ จะทำได้โดยกำหนดค่าคุณสมบัติ String นี้เป็น Vector มีมิติเท่ากับจำนวนตัวเลือกที่ต้องการ

Popup Menu จะพิจารณาค่าคุณสมบัติ Value ตามลำดับการเลือกตัวเลือกต่างๆ เหมือนกับ List boxการกำหนดตัวเลือกเหล่านั้นให้กับคุณสมบัติ String ของ Popup Menu ก็จะกำหนดเป็น Vector มีมิติเท่ากับจำนวนตัวเลือกที่ต้องการ ด้วยเช่นกัน

ดังนั้นโดยการทำงานแล้ว List Box กับ Popup Menu นี้จะมีลักษณะเหมือนกัน เพียงแต่ว่าเรานิยมใช้ List Box เพื่อแสดงตัวเลือกทั้งหลายนี้พร้อมๆ กัน เพื่อให้ผู้ใช้ได้เห็นตลอดเวลา แต่มีข้อเสียที่เปลืองเนื้อที่บน GUI ซึ่ง Popup Menu จะใช้เนื้อที่บน GUI น้อยกว่า แต่ผู้ใช้จะเห็นตัวเลือกได้ค่าเดียว นอกจากจะกดเลือกเพื่อจะดูรายการที่มีอยู่

ลักษณะของ GUI ที่เราจะสร้างมีลักษณะดังรูปต่อไปนี้



ลักษณะการทำงานก็คือ

- ผู้ใช้จะเลือกรายการ Product จาก Popup Menu
- ผู้ใช้สามารถเลือกว่าใช้ผลิตภัณฑ์นั้นที่ใด คือที่ทำงาน และ/หรือ ที่บ้าน แต่ถ้าเลือกที่บ้านจะมีตัวเลือกให้เลือกว่าใช้ผลิตภัณฑ์นั้นบ่อยเพียงใด และถ้าหากไม่ได้เลือกตัวเลือกใช้ผลิตภัณฑ์นี้ที่บ้าน ให้ตัวเลือกใน List Box นั้นไม่สามารถเลือกได้
- แสดงผลด้วย Static Text ข้างล่าง

วางรูปแบบใน GUIDE Layout Editor

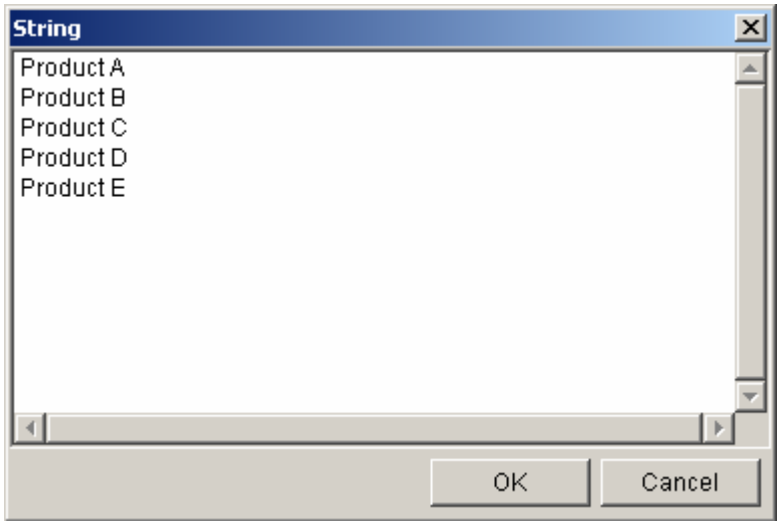
เริ่มด้วยการวัตถุตามรูปที่แสดงข้างบนนี้ ซึ่งเหมือนกับตัวอย่างที่ผ่านมาจึงไม่ขอกล่าวถึงในรายละเอียดในที่นี้อีก

กำหนดชื่อวัตถุโดย Tag และข้อความที่จะปรากฏขึ้น

จากรูปข้างบนนี้ให้กำหนดคุณสมบัติของวัตถุต่างๆ ต่อไปนี้ สำหรับที่ไม่ได้กล่าวถึงหมายความว่าไม่จำเป็นต้องมีการปรับเปลี่ยนคุณสมบัติ

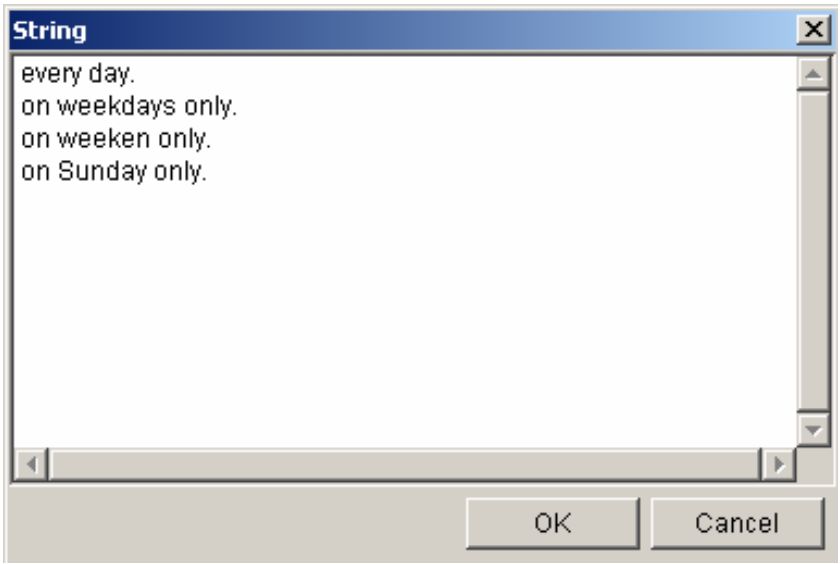
- String ของ Static Text ทั้งหมดดังรูป และให้แก้ไข Tag ของ Static Text ที่แสดงผล การเลือก Product ให้เป็น Result

- สำหรับ Check Box ที่มี String เป็น At Work ให้ใช้ Tag เป็น Work และให้ Value เป็น 0
- สำหรับ Check Box ที่มี String เป็น At Home ให้ใช้ Tag เป็น Home และให้ Value เป็น 0
- สำหรับ Popup Menu ให้กำหนด Tag เป็น Popup1 และให้กรอกคุณสมบัติ Sting โดยกดที่ปุ่ม  หลังคุณสมบัติ String แล้วจะปรากฏหน้าต่างดังรูป ให้กรอกข้อความตามรูป และขอให้เว้นวรรคหนึ่งวรรคก่อนจะเริ่มพิมพ์ เพื่อความสะดวกในการแสดงผล



เมื่อกรอกข้อความเสร็จแล้วให้กด OK เราจะเห็นว่าค่าที่แสดงใน Property Inspector จะแสดงเฉพาะค่าแรกเท่านั้น และให้กำหนดค่า Value ให้เท่ากับ 1

- สำหรับ List Box ให้กำหนด Tag เป็น HomeList และให้กรอกคุณสมบัติ Sting โดยกดที่ปุ่ม  หลังคุณสมบัติ String แล้วจะปรากฏหน้าต่างดังรูป ให้กรอกข้อความตามรูป และขอให้เว้นวรรคหนึ่งวรรคก่อนจะเริ่มพิมพ์ เพื่อความสะดวกในการแสดงผล



เมื่อกรอกข้อความเสร็จแล้วให้กด OK เราจะเห็นว่าค่าที่แสดงใน Property Inspector จะแสดงเฉพาะค่าแรกเท่านั้น กำหนดค่าคุณสมบัติ Value ให้เท่ากับ 1 และกำหนดค่าคุณสมบัติ Enable ให้เป็น off

- สำหรับ Figure นั้นให้กำหนดคุณสมบัติ Name เป็น Product Selection

ซึ่งเมื่อกำหนดค่าตามที่กล่าวแล้วก็ถือว่าเสร็จสิ้นการกำหนดคุณสมบัติ สำหรับคุณสมบัติอื่นๆ เช่น ลักษณะตัวอักษร ขนาดต่างๆ เราไม่ถือว่าเป็นสาระสำคัญในที่นี้

Save และ Activate GUI

ขั้นต่อไปให้เรา Save GUI ของเราโดยเลือก Save จากเมนู File หรือใช้เมาส์กดที่ปุ่ม  ก็ได้ จากนั้น MATLAB จะถามชื่อไฟล์ที่เราต้องการเก็บ ซึ่งในที่นี้เราจะตั้งชื่อว่า product ซึ่งเมื่อเราเลือก OK แล้ว MATLAB ก็จะทำการเก็บเป็นสองไฟล์คือ product.fig และ product.m

จากนั้นเมื่อเรา activate GUI จะทำให้ MATLAB เปิดหน้าต่างของ GUI ของเราขึ้นมา และ Editor/Debugger ก็จะถูกเปิดขึ้นมาพร้อมกับเปิดไฟล์ product.m ขึ้น

ปรับปรุง และเพิ่มเติมชุดคำสั่งให้ GUI ทำงานตามที่ต้องการ

ลักษณะของ GUI ที่เรากำลังจะทำนี้ จะมีลักษณะเหมือนกับการเก็บข้อมูลของวัตถุต่างๆที่อยู่ใน GUI แล้วนำมาแสดงผล ดังนั้นหากเราจะเขียนให้ทุก Callback Subfunction มีระบบการประเมินผลของตัวเองก็จะทำให้โปรแกรมมีขนาดใหญ่มากโดยไม่จำเป็น ดังนั้นเราจึงจะเขียน Subfunction ขึ้นมาอันหนึ่งเพื่อทำหน้าที่ประเมินผล ดังนั้นทุก Subfunction ที่เป็น Callback ของวัตถุต่างๆ ก็จะถูกเรียกมาที่ Subfunction นี้โดยตรง ยกเว้นว่ามีหน้าที่พิเศษที่ต้องดำเนินการก่อนนี้ เราจะเรียก Subfunction นี้ว่า manager ซึ่งให้เราเพิ่มเข้าไปในส่วนท้ายของ Application M-file product.m ดังนี้

```
% --------------------------------------------------------
function manager(handles)

nselect = get(handles.Popup1,'Value');
pop_string = get(handles.Popup1,'String');
ProSelect = pop_string(nselect);
string_display = strcat('You have selected',ProSelect);

if get(handles.Work,'Value') == 1;
    string_display =strcat(string_display,'for use it at work');
    if get(handles.Home,'Value') ==1;
        Nselect = get(handles.HomeList,'Value');
        list_string = get(handles.HomeList,'String');
        HomeSelect = list_string(Nselect);
        string_display=strcat(string_display,'and use it at home',
HomeSelect);
```

```
    end
elseif get(handles.Home,'Value') ==1;
        nselect = get(handles.HomeList,'Value');
        list_string = get(handles.HomeList,'String');
        HomeSelect = list_string(nselect);
        string_display=strcat(string_display,'and use it at home',
HomeSelect);
end

set(handles.Result,'String',string_display);
```

ขั้นตอนการทำงานเราจะกล่าวคร่าวๆ คือ subfunction นี้ต้องการ input ตัวเดียวคือ handles ที่จะให้ค่า handle ของวัตถุต่างๆ จากนั้นขั้นแรกก็จะเริ่มพิจารณาจาก Popup Menu ว่าผู้ใช้เลือกค่าใด หลังจากนั้นก็จะดึง String ที่เลือกมาเก็บไว้ แล้วนำมาต่อกับข้อความใหม่เพื่อสร้างเป็นประโยคขึ้น สำหรับคำสั่ง **strcat** เป็นคำสั่งที่ใช้รวมตัวแปรประเภท String ตั้งแต่สองตัวขึ้นไปเข้าด้วยกัน โดยจะตัดช่องว่างท้ายตัวแปร string นั้นออกไป ทำให้การกรอกคุณสมบัติของ Popup Menu และ List Box นั้นต้องเว้นวรรคไว้หนึ่งช่องก่อนแล้วจึงเริ่มพิมพ์ ซึ่งในกรณีอื่นอาจไม่จำเป็นต้องทำเช่นนี้

จากนั้นในส่วนของ if – elseif –end นั้นก็จะเป็น Logic ซึ่งสร้างขึ้นมาเพื่อตรวจสอบว่าผู้ใช้ได้เลือก check box ใดบ้างและจะมีการเพิ่มข้อความต่อไปให้เป็นประโยคได้อย่างไร โดยข้อความที่เพิ่มเติมเข้าไปนั้นส่วนหนึ่งก็จะเป็นการนำเอา string ของ list box มาเพิ่มเติมเข้าไปในประโยคด้วย เราจะไม่ขอกล่าวถึงรายละเอียดมากกว่านี้ สำหรับตัว Subfuction อื่นๆ ที่มีอยู่ใน Application M-file เราก็เพียงเพิ่มคำสั่ง **manager(handles)** เข้าไปเท่านั้น ยกเว้นใน Subfunction ของ Check Box ที่เรียก At Home เพราะเราต้องการเพิ่มข้อกำหนดว่าหากว่าไม่มีการเลือกใช้ให้เราทำให้ List Box ที่อยู่ข้างล่างนั้นใช้งานไม่ได้ ซึ่งจะมีคำสั่งในลักษณะต่อไปนี้

```
% --------------------------------------------------------------------
function varargout=Home_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.Home.
if get(h,'Value') == 1
   set(handles.HomeList,'Enable','on');
else
    set(handles.HomeList,'Enable','off');
end
manager(handles)
% --------------------------------------------------------------------
```

ซึ่งการที่จะกำหนดว่าวัตถุนั้นใช้ได้หรือไม่ได้ก็ให้เรากำหนดค่าคุณสมบัติ Enable เป็น on หรือ off นั่นเอง

ทดสอบการทำงานของ GUI

ขั้นสุดท้ายเป็นการทดสอบการทำงานของ GUI ให้เราทำการ save file ที่ Editor/Debugger เสียก่อน จากนั้น Activate GUI ของเรา แล้วทดสอบการทำงาน ถ้าทำตามขั้นตอนที่กล่าวมาแล้ว GUI ไม่น่าจะมีปัญหาในการทำงาน

15.4 การเพิ่ม Object Handle หลังจากที่มีการสร้าง GUI

เมื่อเราทำการสร้าง GUI ด้วย GUIDE เรียบร้อยแล้วก็จะมีการเก็บข้อมูลลงใน FIG-file และ M-file เราจะพบว่าได้มีการสร้างตัวแปร Structure ที่มีชื่อว่า handles โดยมีชื่อ field เป็นชื่อ Tag ของวัตถุนั้นและมีค่าของ field เป็นค่า handles ของวัตถุนั้น

อย่างไรก็ตาม การที่ MATLAB สร้างตัวแปร handles ขึ้นเป็นขั้นตอนเริ่มต้นการทำงาน ถ้าเราทำการเพิ่มวัตถุเข้าไปใน GUI เช่น ทำการเขียนกราฟขึ้น แล้วต้องการที่จะปรับปรุงคุณสมบัติของกราฟภายหลัง เราจำเป็นจะต้องทราบ handle ของเส้นกราฟนั้นเพื่อที่จะปรับปรุงคุณสมบัติ แต่ตัวแปร handles ที่เรามีอยู่นั้นไม่มีข้อมูลของเส้นกราฟเส้นนั้นอยู่ ทำให้เราจะต้องกำหนด ค่า handle ของวัตถุนั้นเพิ่มเข้าไป และอย่าลืมว่า ตัวแปร handles ที่ถูกสร้างขึ้นโดย GUIDE นั้นจะมีข้อมูลเฉพาะ uicontrol ที่มีอยู่ในตอนวาง layout GUI เท่านั้น จะไม่มีข้อมูลที่เกี่ยวข้องกับเส้นกราฟ หรือวัตถุอื่นๆที่สร้างขึ้นในภายหลัง ดังนั้นถ้าเราต้องการจะส่งผ่านข้อมูลอื่นๆ ที่เกิดขึ้นภายหลังทั้งที่เป็นวัตถุใน GUI หรือนำข้อมูลประเภทต่างๆไปใช้ใน Subfunction อื่นๆ เราจำเป็นที่จะต้องมีการแก้ไขและปรับปรุงตัวแปร handles แล้วนำไปเก็บไว้ในที่ซึ่งสามารถที่จะเรียกใช้ได้ในทุก Subfunction ที่ต้องการ

ตัวอย่างต่อไปนี้จะแสดงวิธีการแก้ไข เพิ่มเติม ปรับปรุง และบันทึกข้อมูลที่ต้องการส่งผ่านระหว่าง Subfunction ต่างๆ ที่มีอยู่ใน GUI เพื่อให้เรามีความคล่องตัวในการสร้าง GUI ของเรามากยิ่งขึ้น

แต่ก่อนที่จะกล่าวถึงตัวอย่างเราขออธิบายคำสั่ง สองคำสั่งที่สำคัญในการสร้าง เก็บและเรียกใช้ข้อมูลใน GUI คำสั่งดังกล่าวคือ **`guihandles`** และ **`guidata`** ซึ่งมีรายละเอียดดังนี้

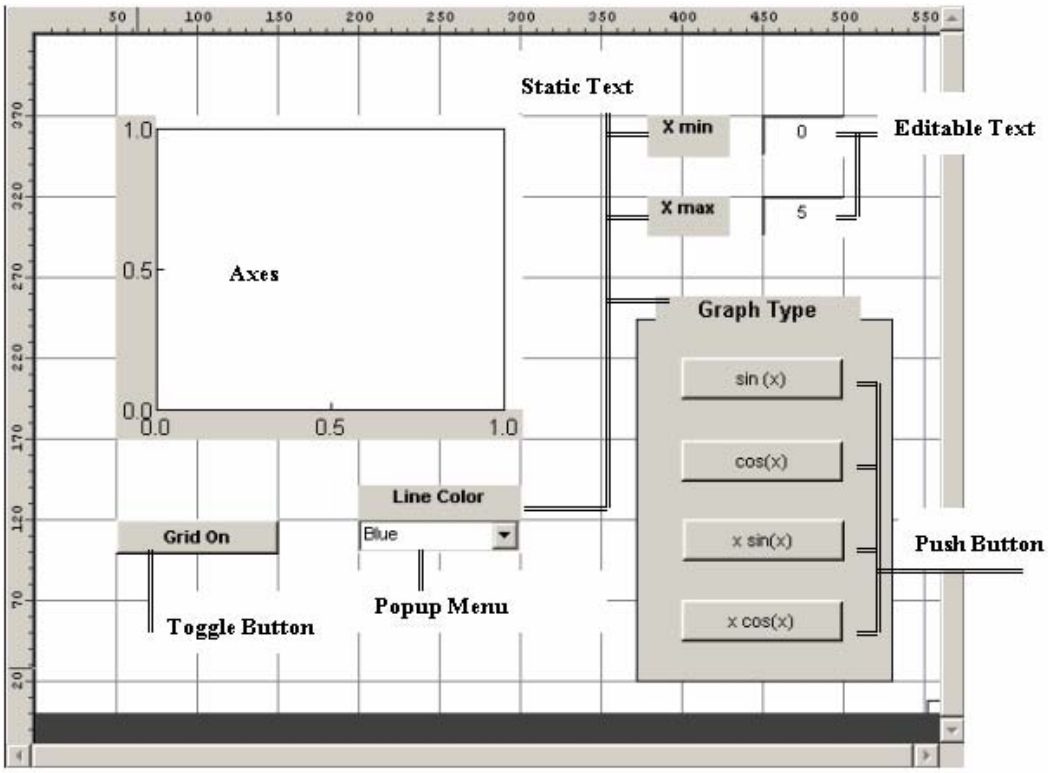
`handles = guihandles(fig);` คำสั่ง **`guihandles`** จะสร้างตัวแปรแบบ Structure จากรูปภาพที่มี handle เท่ากับ fig โดยให้ field ทั้งหมดในตัวแปรนั้นคือวัตถุต่างๆที่มีอยู่ในรูปภาพที่กำหนด ในตัวอย่างนี้ตัวแปรจะมีชื่อว่า **`handles`** สำหรับรูปที่ handle มีค่าเท่ากับ **`fig`** และชื่อ field นี้ก็จะใช้ชื่อ Tag ของวัตถุต่างๆ เหล่านั้น

`guidata(fig, handles)` คำสั่งนี้เป็นคำสั่งที่ให้เก็บค่าตัวแปร **`handles`** เข้าไปเป็นส่วนหนึ่งของรูปที่สร้างขึ้น คำสั่ง **`guidata`** นี้เป็นคำสั่งที่ใช้เก็บข้อมูล หรือดึงข้อมูลที่มีอยู่จากรูปออกมา การเขียนคำสั่งในลักษณะเช่นนี้จะทำให้ MATLAB ทำการเก็บตัวแปรชื่อ **`handles`** นี้เข้าไว้ให้อยู่ในรูป ซึ่งเราสามารถที่จะเรียกข้อมูลจากรูปออกมาได้ในภายหลัง ส่วนสำคัญของคำสั่งนี้ก็คือหากในภายหลังเราได้มีการเพิ่มเติมข้อมูลสำคัญเข้าไปอยู่ใน GUI และต้องการส่งต่อหรือมีการปรับเปลี่ยนค่าตัวแปร **`handles`** นี้ไม่

ว่าด้วยเหตุผลใด เราจะต้องใช้คำสั่งนี้ใหม่อีกครั้งหนึ่ง เพื่อให้เก็บข้อมูลที่ได้รับการปรับแก้แล้วเข้าไปอยู่ในรูป

ดังนั้นถ้าหากว่าเราต้องการที่จะเพิ่มข้อมูลให้กับรูป แล้วส่งผ่านไปยัง Subfunction อื่นๆต่อไป เราแนะนำให้คุณเก็บข้อมูลนั้นไว้ในตัวแปรที่ชื่อ handles นี้ แล้วใช้คำสั่ง **guidata** ทำการเก็บข้อมูลนี้เข้าไปไว้ในรูป เมื่อมีการเรียก Callback คำสั่งใน Callback จะกำหนดให้มีการดึงข้อมูลจากภายในรูปขึ้นมา แล้วส่งให้กับ Subfunction ใน Application M-file ดังนั้น Subfunction ใหม่ก็จะทราบค่าการเปลี่ยนแปลงตัวแปร handles นี้ได้

สำหรับตัวอย่างนี้เราจะสร้างกราฟขึ้นแล้วให้ผู้ใช้สามารถปรับปรุงคุณลักษณะของเส้นกราฟได้ GUI ของเราจะมีลักษณะดังนี้



วางรูปแบบใน GUIDE Layout Editor

สร้าง GUI ดังรูปข้างบน โดยใช้ Uicontrol และวัตถุอื่นๆ ตามแบบที่ได้กำหนดไว้ในรูป

กำหนดคุณสมบัติวัตถุ

กำหนดคุณสมบัติของวัตถุต่างๆ ที่มีอยู่ใน GUI ตามตารางต่อไปนี้

Uicontrol	Property name	Property value

Figure	Name	My Graph
Axes	Next Plot	Replacechildren
Editable Text (ตัวบน)	Tag	Xmin
	String	0
	Blackgroundcolor	เลือกให้เป็นสีขาว
Editable Text (ตัวล่าง)	Tag	Xmax
	String	5
	Blackgroundcolor	เลือกให้เป็นสีขาว
Push Button (sin(x))	Tag	Sine
	String	sin (x)
Uicontrol	**Property name**	**Property value**
Push Button (cos(x))	Tag	Cos
	String	cos(x)
Push Button (x sin(x))	Tag	XSine
	String	x sin (x)
Push Button (x cos(x))	Tag	XCos
	String	x cos (x)
Toggle Button	Tag	GridOn
	String	Grid On
	Value	0
Popup Menu	Tag	Popupmenu2
	String	Blue, Green, Red, Yellow, Black, Pink (โดยให้แต่ละตัวอยู่คนละบรรทัด เหมือนในตัวอย่างก่อนหน้านี้)
Static Text	String	ตามรูปข้างบน

เมื่อแก้ไขคุณสมบัติต่างๆ เรียบร้อยแล้ว ก็เป็นการเสร็จสิ้นในขั้นตอนนี้

Save และ Activate GUI

ขั้นต่อไปให้เรา Save GUI ของเราโดยเลือก Save จากเมนู File หรือใช้เมาส์กดที่ปุ่ม  ก็ได้ จากนั้น MATLAB จะถามชื่อไฟล์ที่เราต้องการเก็บ ซึ่งในที่นี้เราจะตั้งชื่อว่า My_graph ซึ่งเมื่อเราเลือก OK แล้ว MATLAB ก็จะทำการเก็บไฟล์เป็น My_graph.fig และ My_graph.m

จากนั้นเมื่อเรา activate GUI จะทำให้ MATLAB เปิดหน้าต่างของ GUI ของเราขึ้นมา และ Editor/Debugger ก็จะถูกเปิดขึ้นพร้อมกับเปิดไฟล์ My_graph.m ขึ้นมา

ปรับปรุง และเพิ่มเติมชุดคำสั่งให้ GUI ทำงานตามที่ต้องการ

ขั้นตอนต่อไปเป็นการปรับปรุงและแก้ไข subfunction ที่อยู่ภายใน application M-file ซึ่งมีสิ่งที่เราต้องทำเพิ่มเติมดังนี้

ขั้นตอนที่ 1 การเขียน subfunction ให้กับ push button ต่างๆ คือการสร้างเส้นกราฟลงไปในแกนที่เรากำหนดไว้ สำหรับปุ่ม Sine จะมีลักษณะของ code เป็นดังนี้

```
1    function varargout=Sine_Callback(h,eventdata,handles,varargin)
2    % Stub for Callback of the uicontrol handles.Sine.
3    set(handles.figure1,'HandleVisibility','on');
4    xmin = str2num(get(handles.Xmin,'String'));
5    xmax = str2num(get(handles.Xmax,'String'));
6    x=linspace(xmin,xmax,200);
7    y=sin(x);
8    LinePlot=plot(x,y);
9    handles.LineX = LinePlot;
10   guidata(gcbo, handles);
11   popupmenu2_Callback(h, eventdata, handles, varargin);
12   set(handles.figure1,'HandleVisibility','off');
```

หมายเลขข้างหน้าที่ใส่ไว้นั้นเพื่อความสะดวกในการอธิบาย เราไม่จำเป็นต้องใส่ลงไปใน Code สำหรับบรรทัดที่ 1 เป็น function, บรรทัดที่ 3 นั้นเราจะต้องทำให้รูป GUI ของเรานั้นสามารถที่จะให้ handle ของมันมองเห็นได้ เพราะก่อนหน้านี้เราสั่งให้ handle ของรูปภาพนี้มองไม่เห็นเพื่อไม่ให้สามารถที่จะ plot กราฟลงในหน้าต่าง GUI ของเราได้ ในที่นี้เราอาจจะไม่ set เป็น ON แต่ set เป็น Callback ก็ได้ คือให้มองเห็นได้จากคำสั่งที่มาจาก Callback แต่จะมองไม่เห็นจากคำสั่งที่มาจาก Command windows ถ้าเรา set ไว้ที่ ON นี้จะสามารถมองเห็นได้จากทุกที่ เราเลือก On เพราะเราคาดหวังว่าในระหว่างนี้คงไม่มีการสั่ง plot จาก Command windows

บรรทัดที่ 4 และ 5 เป็นการเรียกค่า string จาก edit text ทั้งสองแล้วนำมาเปลี่ยนเป็นตัวเลข เพื่ออ่านค่าว่าค่าสูงสุดและต่ำสุดเป็นเท่าใด บรรทัดที่ 6, 7 และ 8 เป็นการสร้างกราฟและสั่งเขียนกราฟตามปกติ และเก็บ handle ของกราฟนั้นไว้ในชื่อ LinePlot

สำหรับบรรทัดที่ 9 นี้เป็นการสร้าง field ใหม่เพิ่มเข้าไปในตัวแปร handles โดยใช้ชื่อ field ว่า LineX และให้มีค่าเท่ากับ handle ของเส้นกราฟที่เราเพิ่งจะเขียนขึ้น

จากนั้นในบรรทัดที่ 10 เราจะทำการเก็บค่าตัวแปรนี้ไว้ในรูปภาพ สำหรับกรณีนี้เราจะใช้ gcbo ซึ่งหมายถึง get callback object ซึ่งหมายถึง push button นี้ หรือจะใช้ handles.figure1 ซึ่งเป็น handle ของหน้าต่าง GUI ของเราก็ได้ เพราะอย่างไรค่าเหล่านั้นจะถูกเก็บไว้ที่หน้าต่าง GUI เช่นเดียวกัน การเลือกให้เก็บที่ gcbo นั้นโดยแท้จริง MATLAB จะนำไปเก็บไว้ที่ parent ที่เป็นรูปที่บรรจุของวัตถุนั้นไม่ใช่เก็บไว้

วัตถุนั้นแต่อย่างใด ทุกครั้งที่เราทำการเปลี่ยนแปลงค่า handles เราจะต้องทำการเก็บค่าที่เปลี่ยนแปลงเพื่อให้ Subfunction อื่นได้รับทราบการเปลี่ยนแปลงนี้ด้วย

ในบรรทัดที่ 11 จะเป็นการเรียก function ของ popup menu ที่ใช้ปรับสีของเส้นกราฟให้เป็นไปตามที่ผู้ใช้ต้องการ และในบรรทัด 12 เป็นการปิด handle ของรูปภาพไม่ให้ MATLAB สามารถมองเห็นได้

ส่วน Push Button อื่นๆ ก็จะมีลักษณะทำนองเดียวกันเพียงแต่ปรับเปลี่ยนคำสั่งที่ใช้ในการ plot เส้นกราฟให้มีลักษณะที่แตกต่างกันเท่านั้น ซึ่งมี Code ดังนี้คือ

```
function varargout=Cosine_Callback(h,eventdata,handles,varargin)
% Stub for Callback of the uicontrol handles.Cosine.
set(handles.figure1,'HandleVisibility','on');
xmin = str2num(get(handles.Xmin,'String'));
xmax = str2num(get(handles.Xmax,'String'));
x=linspace(xmin,xmax,200);
y=cos(x);
LinePlot=plot(x,y);
handles.LineX = LinePlot;
guidata(gcbo, handles);
popupmenu2_Callback(h, eventdata, handles, varargin);
set(handles.figure1,'HandleVisibility','off');

% --------------------------------------------------------------------
function varargout=Xsine_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.Xsine.
set(handles.figure1,'HandleVisibility','on');
xmin = str2num(get(handles.Xmin,'String'));
xmax = str2num(get(handles.Xmax,'String'));
x=linspace(xmin,xmax,200);
y=x.*sin(x);
LinePlot=plot(x,y);
handles.LineX = LinePlot;
guidata(gcbo, handles);
popupmenu2_Callback(h, eventdata, handles, varargin);
set(handles.figure1,'HandleVisibility','off');

% --------------------------------------------------------------------
function varargout=Xcosine_Callback(h,eventdata,handles,varargin)
% Stub for Callback of the uicontrol handles.Xcosine.
set(handles.figure1,'HandleVisibility','on');
xmin = str2num(get(handles.Xmin,'String'));
xmax = str2num(get(handles.Xmax,'String'));
x=linspace(xmin,xmax,200);
y=x.*cos(x);
LinePlot=plot(x,y);
handles.LineX = LinePlot;
guidata(gcbo, handles);
popupmenu2_Callback(h, eventdata, handles, varargin);
set(handles.figure1,'HandleVisibility','off');
% --------------------------------------------------------------------
```

สำหรับ Popup menu ที่ใช้เลือกสีกราฟนั้นจะมี Code ดังนี้

```
% --------------------------------------------------------------------
function varargout=popupmenu2_Callback(h,eventdata,handles,varargin)
% Stub for Callback of the uicontrol handles.popupmenu2.
```

```
col = get(handles.popupmenu2,'Value');
LinePlot=handles.LineX;
if col == 1
    set(LinePlot,'Color',[0 0 1]');
elseif col == 2
    set(LinePlot,'Color',[0 1 0]');
elseif col == 3
    set(LinePlot,'Color',[1 0 0]');
elseif col == 4
    set(LinePlot,'Color',[1 1 0]');
elseif col == 5
    set(LinePlot,'Color',[0 0 0]');
elseif col == 6
    set(LinePlot,'Color',[1 0 1]');
end
% --------------------------------------------------------------------
```

เป็นการสั่งให้เปลี่ยนสีเส้นกราฟ ที่บรรจุอยู่ใน handles.LineX ให้เป็นไปตามต้องการ โดยการกำหนดสีก็จะเป็นการผสมสี RGB นั่นเอง

อันดับต่อไป Callback ของ Toggle Button จะมี Code ดังนี้

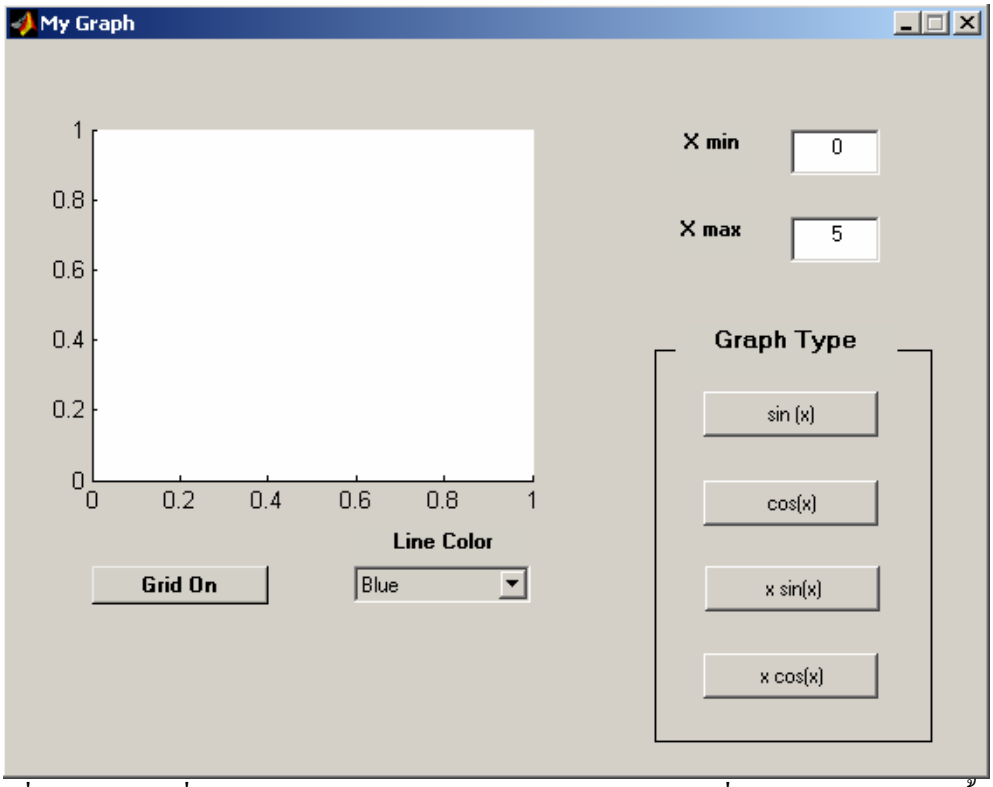
```
% --------------------------------------------------------------------
function varargout=GridOn_Callback(h,eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.GridOn.
set(handles.figure1,'HandleVisibility','on');
status = get(h,'Value');
if status == 0
    set(h,'String','Grid On');
    grid off
else
    set(h,'String','Grid Off');
    grid on
end
set(handles.figure1,'HandleVisibility','off');
% --------------------------------------------------------------------
```

ส่วนนี้ก็จะเป็นการกำหนดตัวหนังสือที่อยู่บน Toggle นี้ให้เปลี่ยนไป เมื่อมีการเลือกกดปุ่มนี้ และจะเห็นว่าตอนต้นและตอนท้าย code จะมีการเปิด และ ปิด handle ของรูปภาพนี้ด้วย ส่วนคำสั่ง **grid on** และ **grid off** นั้นเป็นคำสั่งตามปกติของกราฟทั่วๆไป

สำหรับ Subfucntion ที่เป็น Callback ของ Editable Text Max และ Min นั้นไม่ต้องเขียนเพิ่มเติมและเราก็จะไม่ใช้ในที่นี้ โดยเราจะลบออกทั้งหมด หรือจะลบออกเฉพาะในส่วนที่ให้แสดงข้อความว่ายังไม่มีการกำหนด Callback ให้กับวัตถุนี้ออกไปเท่านั้นก็ได้

ทดสอบการทำงานของ GUI

จากนั้นลอง Save Application M-file แล้วลอง activate GUI ของเราดู เราจะเห็นหน้าต่างต่อไปนี้ปรากฏขึ้น

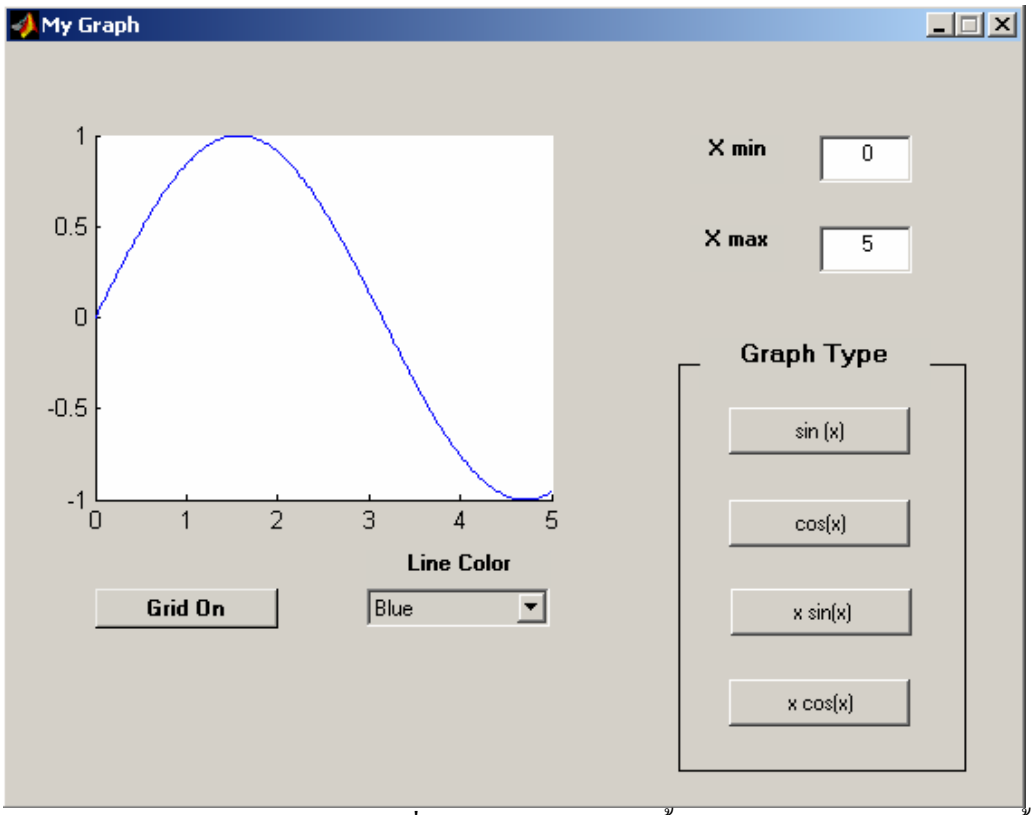


ซึ่งเราสามารถที่จะลองให้ GUI เขียนกราฟในลักษณะต่างๆ ที่เราต้องการได้ จากนั้นเราก็จะสามารถที่จะเปลี่ยนสีกราฟต่างๆ เหล่านั้นได้ ข้อเสียของ GUI นี้ก็คือถ้าให้มีการเปลี่ยนสีกราฟก่อนที่จะมีการเขียนกราฟขึ้นมันก็จะเกิดการ error ขึ้นทันที เพราะก่อนที่จะมีการสั่งให้ plot graph นั้นเรายังไม่มีการกำหนดตัวแปรชื่อ handles.LineX ลงไป ซึ่งเราอาจจะแก้ไขข้อเสียนี้ได้หลายวิธี ในที่นี้เราจะเสนอวิธีการเขียนกราฟลงไปในช่วงของการเปิด GUI นี้ขึ้นมา พร้อมทั้งมีการปรับแก้ตัวแปร handles นี้ไว้ก่อนที่ผู้ใช้จะเริ่มทำการใช้ GUI

ไปที่ Application M-file ในช่วงบน แล้วมองหาคำสั่ง guidata และหลังจากคำสั่ง guidata ที่เป็นคำสั่งเดิมให้เราสั่งคำสั่งเพื่อให้มีการสร้างเส้นกราฟขึ้นแล้วเก็บค่าใหม่ ด้วยคำสั่ง guidata อีกครั้งหนึ่ง ลักษณะคำสั่งเป็นดังนี้

```
set(fig,'HandleVisibility','on');
xmin = str2num(get(handles.Xmin,'String'));
xmax = str2num(get(handles.Xmax,'String'));
x=linspace(xmin,xmax,200);
y=sin(x);
LinePlot=plot(x,y);
handles.LineX = LinePlot;
guidata(fig, handles);
set(fig,'HandleVisibility','off');
```

ซึ่งคำสั่งนี้จะทำให้เกิดการสร้างกราฟขึ้นก่อน พร้อมกับปรับปรุงข้อมูลของ handles ใน GUI พร้อมกันไปด้วย จากนั้น Save Application M-file แล้วทำการ activate GUI ของเราอีกครั้งหนึ่ง เราจะได้หน้าต่างต่อไปนี้



จะเห็นว่าเราสามารถแก้ไขปัญหาที่เรากล่าวถึงก่อนหน้านั้นได้ จากการแก้ปัญหาในจุดนี้เราจะเห็นว่าเราสามารถที่จะเพิ่มเติมข้อมูลต่างๆ เพื่อจะส่งผ่านไปยัง Subfunction เมื่อเริ่มต้นการทำงานของ GUI ได้เช่นกัน

สำหรับการแก้ไข GUI นี้ในจุดอื่นๆ ก็สามารถทำได้เช่นกัน ทั้งนี้เป็นไปตามความต้องการของผู้ออกแบบ GUI นั่นเอง

ภาคผนวก สรุปคำสั่งของ MATLAB

จากที่เราได้ศึกษาการทำงานเบื้องต้นของ MATLAB จะพบว่า MATLAB มี function และคำสั่งมากมาย อย่างไรก็ตามที่ได้ศึกษาไปนั้นเป็นเพียงส่วนหนึ่งของ MATLAB เท่านั้นในภาคผนวกนี้ได้รวบรวมเอา function ต่างๆ ของ MATLAB ทั้งหมดมารวบรวมไว้เป็นหมวดหมู่ หากต้องการทราบรายละเอียดของ function ใด ให้ดูจาก help

General Purpose Commands

Managing Commands and Functions

addpath	Add directories to MATLAB's search path
doc	Load hypertext documentation
help	Online help for MATLAB functions and M-files
lasterr	Last error message
lookfor	Keyword search through all help entries
path	Control MATLAB's directory search path
profile	Measure and display M-file execution profiles
rmpath	Remove directories from MATLAB's search path
type	List file
version	MATLAB version number
what	Directory listing of M-files, MAT-files, and MEX-files
whatsnew	Display README files for MATLAB and toolboxes
which	Locate functions and files

Managing Variables and the Workspace

clear	Remove items from memory
disp	Display text or array
length	Length of vector
load	Retrieve variables from disk
pack	Consolidate workspace memory
save	Save workspace variables on disk
size	Array dimensions
who, whos	List directory of variables in memory

Controlling the Command Window

echo	Echo M-files during execution
format	Control the output display format
more	Control paged output for the command window

Working with Files and the Operating Environment

cd	Change working directory
delete	Delete files and graphics objects
diary	Save session in a disk file
dir	Directory listing
edit	Edit an M-file
fullfile	Build full filename from parts
inmem	Functions in memory
MATLABroot	Root directory of MATLAB installation

tempdir	Return the name of the system's temporary directory.
tempname	Unique name for temporary file
!	Execute operating system command

Starting and Quitting MATLAB

MATLABrc	MATLAB startup M-file
quit	Terminate MATLAB .
startup	MATLAB startup M-file

Operators and Special Characters

+	Plus
-	Minus
*	Matrix multiplication
.*	Array multiplication
^	Matrix power
.^	Array power
kron	Kronecker tensor product.
\	Backslash or left division.
/	Slash or right division
./ and .\	Array division, right and left
:	Colon
( )	Parentheses
[ ]	Brackets
{}	Curly braces
.	Decimal point
...	Continuation
,	Comma
;	Semicolon.
%	Comment
!	Exclamation point
'	Transpose and quote
.'	Nonconjugated transpose
=	Assignment.
==	Equality.
< >	Relational operators
&	Logical AND .
|	Logical OR.
~	Logical NOT
xor	Logical EXCLUSIVE OR

Logical Functions

all	Test to determine if all elements are nonzero
any	Test for any nonzeros
exist	Check if a variable or file exists
find	Find indices and values of nonzero elements
is*	Detect state.
*isa	Detect an object of a given class.
logical	Convert numeric values to logical

Language Constructs and Debugging

MATLAB as a Programming Language

builti	Execute builtin function from overloaded method
eval	Interpret strings containing MATLAB expressions
feval	Function evaluation
function	Function M-files
global	Define global variables
nargchk	Check number of input arguments

script	Script M-files

Control Flow

break	Break out of flow control structures
case	Case switch.
else	Conditionally execute statements
elseif	Conditionally execute statements
end	Terminate for, while, switch, and if statements or indicate last index
error	Display error messages
for	Repeat statements a specific number of times
if	Conditionally execute statements
otherwise	Default part of switch statement
return	Return to the invoking function
switch	Switch among several cases based on expression
warning	Display warning message
while	Repeat statements an indefinite number of times

Interactive Input

input	Request user input.
keyboard	Invoke the keyboard in an M-file.
menu	Generate a menu of choices for user input
pause	Halt execution temporarily

Object-Oriented Programming

class	Create object or return class of object
double	Convert to double precision
inferiorto	Inferior class relationship
inline	Construct an inline object.
isa	Detect an object of a given class
superiorto	Superior class relationship
uint8	Convert to unsigned 8-bit integer

Debugging

dbclear	Clear breakpoints
dbcont	Resume execution
dbdown	Change local workspace context
dbmex	Enable MEX-file debugging
dbquit	Quit debug mode
dbstack	Display function call stack
dbstatus	List all breakpoints.
dbstep	Execute one or more lines from a breakpoint.
dbstop	Set breakpoints in an M-file function
dbtype	List M-file with line numbers
dbup	Change local workspace context

Elementary Matrices and Matrix Manipulation

Elementary Matrices and Arrays

eye	Identity matrix
linspace	Generate linearly spaced vectors
logspace	Generate logarithmically spaced vectors
ones	Create an array of all ones
rand	Uniformly distributed random numbers and arrays
randn	Normally distributed random numbers and arrays
zeros	Create an array of all zeros
: (colon)	Regularly spaced vector

Special Variables and Constants

ans	The most recent answer
computer	Identify the computer on which MATLAB is running
eps	Floating-point relative accuracy
flops	Count floating-point operations
i	Imaginary unit.
Inf	Infinity
inputname	Input argument name
j	Imaginary unit.
NaN	Not-a-Number
nargin, nargout	Number of function arguments.
pi	Ratio of a circle's circumference to its diameter,π
realmax	Largest positive floating-point number
realmin	Smallest positive floating-point number
varargin, varargout	Pass or return variable numbers of arguments.

Time and Dates

calendar	Calendar.
clock	Current time as a date vector
cputime	Elapsed CPU time .
date	Current date string
datenum	Serial date number
datestr	Date string format
datevec	Date components
eomday	End of month.
etime	Elapsed time.
now	Current date and time
tic, toc	Stopwatch timer .
weekday	Day of the week .

Matrix Manipulation

cat	Concatenate arrays
diag	Diagonal matrices and diagonals of a matrix
fliplr	Flip matrices left-right
flipud	Flip matrices up-down
repmat	Replicate and tile an array.
reshape	Reshape array .
rot90	Rotate matrix 90 degrees.
tril	Lower triangular part of a matrix.
triu	Upper triangular part of a matrix .
: (colon)	Index into array, rearrange array .

Specialized Matrices

compan	Companion matrix .
gallery	Test matrices .
hadamard	Hadamard matrix . .
hankel	Hankel matrix .
hilb	Hilbert matrix .
invhilb	Inverse of the Hilbert matrix .
magic	Magic square
pascal	Pascal matrix .
toeplitz	Toeplitz matrix .
wilkinson	Wilkinson's eigenvalue test matrix

Elementary Math Functions

abs	Absolute value and complex magnitude
acos, acosh	Inverse cosine and inverse hyperbolic cosine
acot, acoth	Inverse cotangent and inverse hyperbolic cotangent

acsc, acsch	Inverse cosecant and inverse hyperbolic cosecant
angle	Phase angle
asec, asech	Inverse secant and inverse hyperbolic secant
asin, asinh	Inverse sine and inverse hyperbolic sine
atan, atanh	Inverse tangent and inverse hyperbolic tangent
atan2	Four-quadrant inverse tangent.
ceil	Round toward infinity.
conj	Complex conjugate
cos, cosh	Cosine and hyperbolic cosine.
cot, coth	Cotangent and hyperbolic cotangent.
csc, csch	Cosecant and hyperbolic cosecant
exp	Exponential
fix	Round towards zero
floor	Round towards minus infinity
gcd	Greatest common divisor
imag	Imaginary part of a complex number
lcm	Least common multiple
log	Natural logarithm
log2	Base 2 logarithm and dissect floating-point numbers into exponent and mantissa
log10	Common (base 10) logarithm
mod	Modulus (signed remainder after division)
real	Real part of complex number
rem	Remainder after division
round	Round to nearest integer
sec, sech	Secant and hyperbolic secant
sign	Signum function
sin, sinh	Sine and hyperbolic sine
sqrt	Square root
tan, tanh	Tangent and hyperbolic tangent

Specialized Math Functions

airy	Airy functions
besselh	Bessel functions of the third kind (Hankel functions)
besseli,	besselk
Modified	Bessel functions
besselj, bessely	Bessel functions
beta, betainc, betaln	Beta functions
ellipj	Jacobi elliptic functions
ellipke	Complete elliptic integrals of the first and second kind
erf, erfc, erfcx, erfinv	Error functions
expint	Exponential integral
gamma, gammainc, gammaln	Gamma functions
legendre	Associated Legendre functions.
pow2	Base 2 power and scale floating-point numbers
rat, rats	Rational fraction approximation

Coordinate System Conversion

cart2pol	Transform Cartesian coordinates to polar or cylindrical
cart2sph	Transform Cartesian coordinates to spherical
pol2cart	Transform polar or cylindrical coordinates to Cartesian
sph2cart	Transform spherical coordinates to Cartesian

Matrix Functions - Numerical Linear Algebra

Matrix Analysis

cond	Condition number with respect to inversion
condeig	Condition number with respect to eigenvalues
det	Matrix determinant
norm	Vector and matrix norms
null	Null space of a matrix
orth	Range space of a matrix
rank	Rank of a matrix
rcond	Matrix reciprocal condition number estimate
rref, rrefmovie	Reduced row echelon form.
subspace	Angle between two subspaces.
trace	Sum of diagonal elements.

Linear Equations

\ /	Linear equation solution.
chol	Cholesky factorization
inv	Matrix inverse
lscov	Least squares solution in the presence of known covariance
lu	LU matrix factorization.
nnls	Nonnegative least squares
pinv	Moore-Penrose pseudoinverse of a matrix
qr	Orthogonal-triangular decomposition

Eigenvalues and Singular Values

balance	Improve accuracy of computed eigenvalues
cdf2rdf	Convert complex diagonal form to real block diagonal form
eig	Eigenvalues and eigenvectors
hess	Hessenberg form of a matrix
poly	Polynomial with specified roots
qz	QZ factorization for generalized eigenvalues
rsf2csf	Convert real Schur form to complex Schur form.
schur	Schur decomposition
svd	Singular value decomposition

Matrix Functions

expm	Matrix exponential
funm	Evaluate functions of a matrix
logm	Matrix logarithm
sqrtm	Matrix square root

Low Level Functions

qrdelete	Delete column from QR factorization.
qrinsert	Insert column in QR factorization.

Data Analysis and Fourier Transform Functions

Basic Operations

convhull	Convex hull
cumprod	Cumulative product
cumsum	Cumulative sum
cumtrapz	Cumulative trapezoidal numerical integration
delaunay	Delaunay triangulation
dsearch	Search for nearest point
factor	Prime factors
inpolygon	Detect points inside a polygonal region
max	Maximum elements of an array
mean	Average or mean value of arrays

median	Median value of arrays
min	Minimum elements of an array
perms	All possible permutations
polyarea	Area of polygon
primes	Generate list of prime numbers
prod	Product of array elements
sort	Sort elements in ascending order.
sortrows	Sort rows in ascending order
std	Standard deviation
sum	Sum of array elements
trapz	Trapezoidal numerical integration
tsearch	Search for enclosing Delaunay triangle
voronoi	Voronoi diagram.

Finite Differences

del2	Discrete Laplacian
diff	Differences and approximate derivatives
gradient	Numerical gradient

Correlation

corrcoef	Correlation coefficients.
cov	Covariance matrix

Filtering and Convolution

conv	Convolution and polynomial multiplication
conv2	Two-dimensional convolution.
deconv	Deconvolution and polynomial division.
filter	Filter data with an infinite impulse response (IIR) or finite impulse response (FIR) filter
filter2	Two-dimensional digital filtering

Fourier Transforms

abs	Absolute value and complex magnitude
angle	Phase angle
cplxpair	Sort complex numbers into complex conjugate pairs
fft	One-dimensional fast Fourier transform
fft2	Two-dimensional fast Fourier transform
fftshift	Move zero'th lag to center of spectrum.
ifft	Inverse one-dimensional fast Fourier transform
ifft2	Inverse two-dimensional fast Fourier transform
nextpow2	Next power of two
unwrap	Correct phase angles

Vector Functions

cross	Vector cross product
intersect	Set intersection of two vectors
ismember	Detect members of a set
setdiff	Return the set difference of two vectors
setxor	Set exclusive-or of two vectors
union	Set union of two vectors
unique	Unique elements of a vector

Polynomial and Interpolation Functions

Polynomials

conv	Convolution and polynomial multiplication
deconv	Deconvolution and polynomial division
Poly	Polynomial with specified roots

polyder	Polynomial derivative
polyeig	Polynomial eigenvalue problem
polyfit	Polynomial curve fitting
polyval	Polynomial evaluation
polyvalm	Matrix polynomial evaluation
residue	Convert between partial fraction expansion and polynomial coefficients
roots	Polynomial roots

Data Interpolation

griddata	Data gridding.
interp1	One-dimensional data interpolation (table lookup)
interp2	Two-dimensional data interpolation (table lookup)
interp3	Three-dimensional data interpolation (table lookup)
interpft	One-dimensional interpolation using the FFT method
interpn	Multidimensional data interpolation (table lookup).
meshgrid	Generate X and Y matrices for three-dimensional plots
ndgrid	Generate arrays for multidimensional functions and interpolation
spline	Cubic spline interpolation

Function Functions – Nonlinear Numerical Methods

dblquad	Numerical double integration
fmin	Minimize a function of one variable
fmins	Minimize a function of several variables.
fzero	Zero of a function of one variable .
ode45, ode23, ode113, ode15s, ode23s	Solve differential equations
odefile	Define a differential equation problem for ODE solvers
odeget	Extract properties from options structure created with odeset
odeset	Create or alter options structure for input to ODE solvers
quad, quad8	Numerical evaluation of integrals

Sparse Matrix Functions

Elementary Sparse Matrices

spdiags	Extract and create sparse band and diagonal matrices
speye	Sparse identity matrix
sprand	Sparse uniformly distributed random matrix
sprandn	Sparse normally distributed random matrix.
sprandsym	Sparse symmetric random matrix

Full to Sparse Conversion

find	Find indices and values of nonzero elements
full	Convert sparse matrix to full matrix
sparse	Create sparse matrix
spconvert	Import matrix from sparse matrix external format

Working with Nonzero Entries of Sparse Matrices

nnz	Number of nonzero matrix elements
nonzeros	Nonzero matrix elements
nzmax	Amount of storage allocated for nonzero matrix elements
spalloc	Allocate space for sparse matrix
spfun	Apply function to nonzero sparse matrix elements
spones	Replace nonzero sparse matrix elements with ones

Visualizing Sparse Matrices

spy	Visualize sparsity pattern

Reordering Algorithms

colmmd	Sparse column minimum degree permutation.
colperm	Sparse column permutation based on nonzero count
dmperm	Dulmage-Mendelsohn decomposition
randperm	Random permutation .
symmmd	Sparse symmetric minimum degree ordering.
symrcm	Sparse reverse Cuthill-McKee ordering.

Norm, Condition Number, and Rank

condest 1	norm matrix condition number estimate
normest 2	norm estimate.

Sparse Systems of Linear Equations

bicg	BiConjugate Gradients method .
bicgstab	BiConjugate Gradients Stabilized method
cgs	Conjugate Gradients Squared method .
cholinc	Incomplete Cholesky factorizations .
gmres	Generalized Minimum Residual method (with restarts).
luinc	Incomplete LU matrix factorizations .
pcg	Preconditioned Conjugate Gradients method .
qmr	Quasi-Minimal Residual method .

Sparse Eigenvalues and Singular Values

eigs	Find a few eigenvalues and eigenvectors .
svds	A few singular values.

Miscellaneous

spparms	Set parameters for sparse matrix routines .

Sound Processing Functions

General Sound Functions

sound	Convert vector into sound

SPARCstation-specific Sound Functions

auread	Read NeXT/SUN (.au) sound file
auwrite	Write NeXT/SUN (.au) sound file

.WAV Sound Functions

wavread	Read Microsoft WAVE (.wav) sound file
wavwrite	Write Microsoft WAVE (.wav) sound file .

Character String Functions

General

abs	Absolute value and complex magnitude
eval	Interpret strings containing MATLAB expressions
strings	MATLAB string handling

String Manipulation

deblank	Strip trailing blanks from the end of a string
findstr	Find one string within another
lower	Convert string to lower case
strcat	String concatenation
strcmp	Compare strings.
strjust	Justify a character array.

strmatch	Find possible matches for a string
strncmp	Compare the first n characters of two strings
strrep	String search and replace
strtok	First token in string
strvcat	Vertical concatenation of strings
upper	Convert string to upper case

String to Number Conversion

char	Create character array (string)
int2str	Integer to string conversion
mat2str	Convert a matrix into a string
num2str	Number to string conversion
sprintf	Write formatted data to a string
sscanf	Read string under format control
str2num	String to number conversion

Radix Conversion

bin2dec	Binary to decimal number conversion
dec2bin	Decimal to binary number conversion
dec2hex	Decimal to hexadecimal number conversion
hex2dec	IEEE hexadecimal to decimal number conversion.
hex2num	Hexadecimal to double number conversion.

Low-Level File I/O Functions

File Opening and Closing

fclose	Close one or more open files
fopen	Open a file or obtain information about open files

Unformatted I/O

fread	Read binary data from file
fwrite	Write binary data to a file.

Formatted I/O

fgetl	Return the next line of a file as a string without line terminator(s)
fgets	Return the next line of a file as a string with line terminator(s)
fprintf	Write formatted data to file
fscanf	Read formatted data from file

File Positioning

feof	Test for end-of-file
ferror	Query MATLAB about errors in file input or output
frewind	Rewind an open file
fseek	Set file position indicator
ftell	Get file position indicator.

String Conversion

sprintf	Write formatted data to a string .
sscanf	Read string under format control

Specialized File I/O

dlmread	Read an ASCII delimited file into a matrix.
dlmwrite	Write a matrix to an ASCII delimited file
imfinfo	Return information about a graphics file
imread	Read image from graphics file
imwrite	Write an image to a graphics file
wk1read	Read a Lotus123 WK1 spreadsheet file into a matrix
wk1write	Write a matrix to a Lotus123 WK1 spreadsheet file.

Bitwise Functions

bitand	Bit-wise AND.
bitcmp	Complement bits
bitor	Bit-wise OR
bitmax	Maximum floating-point integer
bitset	Set bit
bitshift	Bit-wise shift.
bitget	Get bit
bitxor	Bit-wise XOR

Structure Functions

fieldnames	Field names of a structure
getfield	Get field of structure array
rmfield	Remove structure fields
setfield	Set field of structure array
struct	Create structure array
struct2cell	Structure to cell array conversion

Object Functions

class	Create object or return class of object
isa	Detect an object of a given class

Cell Array Functions

cell	Create cell array
cellstr	Create cell array of strings from character array
cell2struct	Cell array to structure array conversion
celldisp	Display cell array contents
cellplot	Graphically display the structure of cell arrays
num2cell	Convert a numeric array into a cell array

Multidimensional Array Functions

cat	Concatenate arrays
flipdim	Flip array along a specified dimension
ind2sub	Subscripts from linear index.
ipermute	Inverse permute the dimensions of a multidimensional array
ndgrid	Generate arrays for multidimensional functions and interpolation
ndims	Number of array dimensions
permute	Rearrange the dimensions of a multidimensional array
reshape	Reshape array
shiftdim	Shift dimensions
squeeze	Remove singleton dimensions
sub2ind	Single index from subscripts

เอกสารอ้างอิง

ในการเรียบเรียงเอกสารชุดนี้กระผมได้ใช้เอกสารอ้างอิงหลายเล่ม สำหรับรายการเอกสารและหนังสือที่ใช้อยู่เป็นประจำได้เรียบเรียงไว้เป็นหมวดหมู่ตามรายการข้างล่างนี้ โดยแบ่งเป็นส่วนๆดังนี้

เอกสารและหนังสือที่เกี่ยวข้องกับ MATLAB โดยตรง

The Student Edition of MATLAB Version 5 : User's Guide, Prentice Hall, 1997.

Biran, A. and Breiner, M., "*MATLAB for Engineers*," Addison-Wesley, 1995.

Enander, E. P., et al, "*The MATLAB Handbook*," Addison-Wesley, 1996.

Etter, D.M., "*Engineering Problem Solving with MATLAB*," 2nd ed., Prentice Hall, 1997.

MATLAB Language Reference Manual : Version 5, MathWorks Inc., 1996.

MATLAB Graphics Reference Manual : Version 5, MathWorks Inc., 1996.

MATLAB Notebook User's Guide : Version 5, MathWorks Inc., 1996.

เอกสารและหนังสือทางด้านคณิตศาสตร์

Gerald, C.F. and Wheatley, P.O., "*Applied Numerical Analysis*," 5th ed., Addison-Wesley, 1994.

Kreyszig, E., "*Advance Engineering Mathematics*," 7th ed., John Wiley, 1993.

เอกสารและหนังสือทางด้านวิศวกรรมศาสตร์

Inman, D.J., "*Engineering Vibration,*" Prentics Hall, 1994.

Thomson, W.T., "*Theory of Vibration with Applications*," 4th ed., Prentics Hall, 1993.

Rao, S. S., "*Mechanical Vibrations*," 3rd ed., Addison-Wesley, 1995.

Steidel, R. F., "*An Introduction to Mechanical Vibrations*," 3rd ed., John Wiley, 1989.

Kelly, S.G., "*Mechanical Vibrations*," McGraw-Hill, 1993.

Ferguson, C. R., "*Internal Combustion Engine*," John Wiley, 1986.

Heywood, J. B., "*Internal Combustion Engine Fundamentals*," McGraw-Hill, 1988.

Shames, I. V., "*Mechanics of Fluids*," 3rd ed., McGraw-Hill, 1992.

Janna, W. S., "*Introduction to Fluid Mechanics*," 3rd ed., PWS, 1993.

Anderson, J. D., "*Fundamental of Aerodynamics*," 2nd ed., McGraw-Hill, 1991.

Smith, W. F., "*Principles of Materials Science and Engineering*," 3rd ed., McGraw-Hill, 1996.

Shames, I. V., "*Introduction to Solid Mechanics*," 2nd ed., Prentics Hall, 1989.

Web Site: *www.mathworks.com*