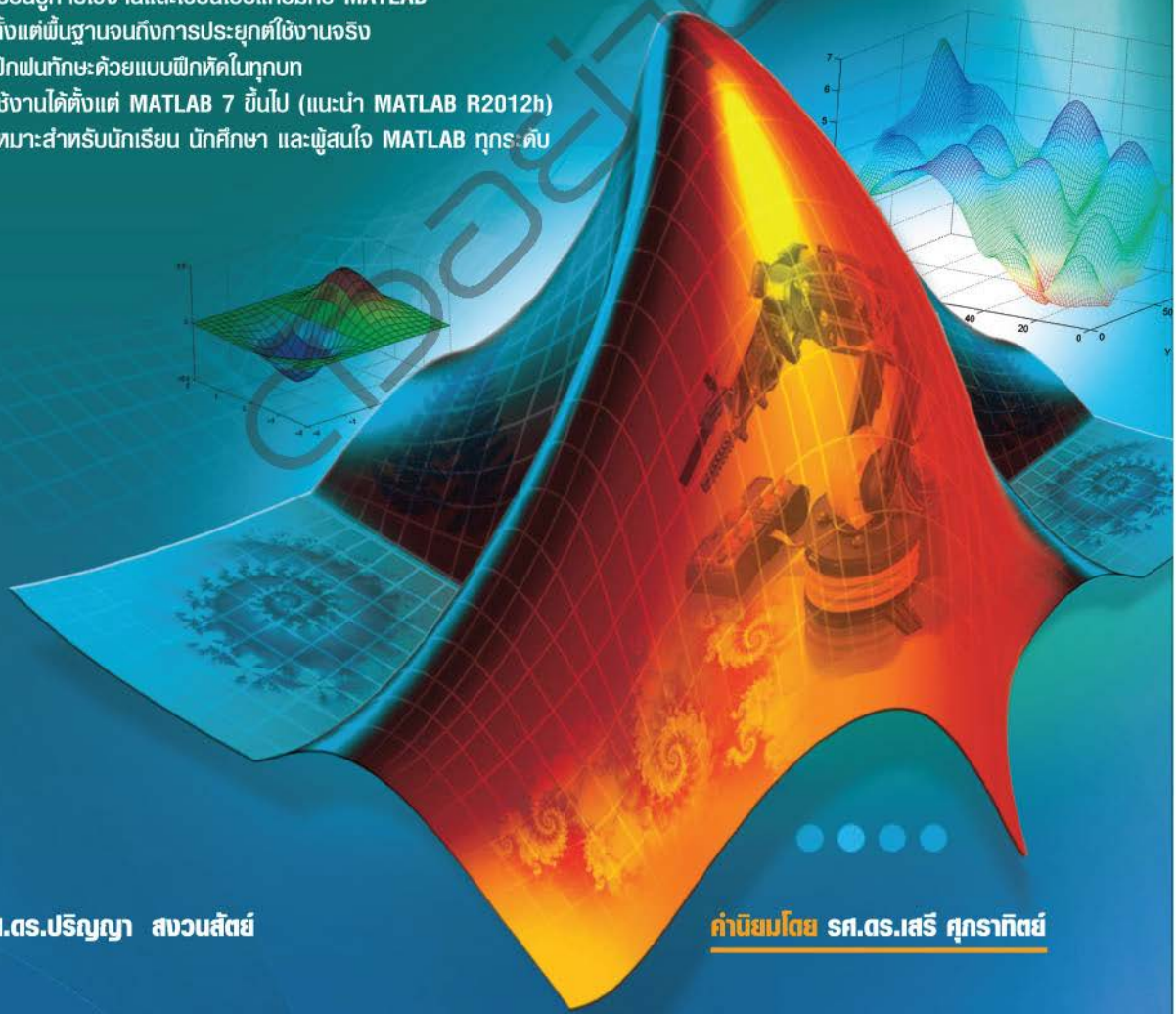
INFOPRESS GROUP

หนังสือ MATLAB ที่มีผู้อ่านมากที่สุด

DEV BOOK

คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์

- เรียนรู้การใช้งานและเขียนโปรแกรมกับ MATLAB ตั้งแต่พื้นฐานจนถึงการประยุกต์ใช้งานจริง
- ฝึกฝนทักษะด้วยแบบฝึกหัดในทุกบท
- ใช้งานได้ตั้งแต่ MATLAB 7 ขึ้นไป (แนะนำ MATLAB R2012b)
- เหมาะสำหรับนักเรียน นักศึกษา และผู้สนใจ MATLAB ทุกระดับ



พศ.ดร.ปริญญา สงวนสัตย์

คำนิยมโดย รศ.ดร.เสรี ศุภราทิตย์

คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์

ผู้แต่ง	ผศ.ดร.ปริญญา สงวนสัตย์ http://researchers.in.th/blog/transfinity
บรรณาธิการ	สัจจะ จรัสรุ่งรวีวร sajja@infopress.co.th
ออกแบบปก	อนัน วาโซะ
ออกแบบและจัดรูปเล่ม	วุฒิพันธ์ สมพระเมฆ, สิริชัย บำรุงสุข
พิสูจน์อักษร	มานิดา ณ บางช้าง
ประสานงานการผลิต	สุพัตรา อาจปรุ, ฉัตรชนก แก้วจันทร์

บริษัท ไอดีซี พรีเมียร์ จำกัด จัดตั้งขึ้นเพื่อเผยแพร่ความรู้เทคโนโลยีสารสนเทศทั้งในระดับพื้นฐานและระดับสูง เรายินดีรับงานเขียนของนักวิชาการและนักเขียนทุกท่าน โดยเฉพาะงานที่เกี่ยวข้องกับสารสนเทศ ท่านผู้สนใจกรุณาติดต่อผ่านทางอีเมลที่ editor@infopress.co.th หรือ โทรศัพท์หมายเลข 0-2962-1081 (อัตโนมัติ 10 คู่สาย) โทรสาร 0-2962-1084

ข้อมูลทางบรรณานุกรม

ผศ.ดร.ปริญญา สงวนสัตย์
คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์. นนทบุรี : ไอดีซีฯ, 2556
504 หน้า
1. แมทแล็บ (โปรแกรมคอมพิวเตอร์) 2. การเขียนโปรแกรม (คอมพิวเตอร์) I ชื่อเรื่อง
005.133
ISBN (E-Book) 885-916-100-333-5

DEV BOOK
Infopress Developer Book

พิมพ์ครั้งที่ 1 มีนาคม 2556

จัดพิมพ์และจัดจำหน่ายโดย

IDC PREMIER

บริษัท ไอดีซี พรีเมียร์ จำกัด
200 หมู่ 4 ชั้น 19 ห้อง 1901 อาคารจัสมินอินเตอร์เนชั่นแนล
ถ.แจ้งวัฒนะ ต.ปากเกร็ด อ.ปากเกร็ด จ.นนทบุรี 11120
โทรศัพท์ 0-2962-1081 (อัตโนมัติ 10 คู่สาย) โทรสาร 0-2962-1084

สำหรับร้านค้าและตัวแทนจำหน่าย
โทรศัพท์ 0-2962-1081-3 ต่อ 112-114
โทรสาร 0-2962-1084

ลูกค้าสัมพันธ์
โทรศัพท์ 0-2962-1081-3 ต่อ 601
โทรสาร 0-2962-1084

ราคา 350 บาท

Contents

บทที่ 1 รู้จักกับ MATLAB

บทที่ 2 เทนเซอร์และอาร์เรย์

Chapter 01

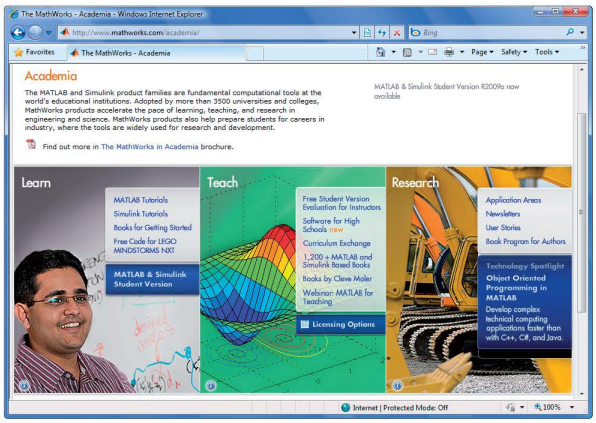
รู้จักกับ MATLAB

MATLAB คืออะไร

MATLAB คือ โปรแกรมการคำนวณเชิงตัวเลขที่มีสิ่งแวดล้อมในการคำนวณของตัวเอง (Numerical Computing Environment) และมีภาษาเฉพาะตัวในการเขียนโปรแกรมได้ โดย MATLAB มาจากคำ 2 คำรวมกัน คือ Matrix และ Laboratory ซึ่งหมายถึงห้องปฏิบัติการเมทริกซ์

MATLAB มีจุดกำเนิดในช่วงปี ค.ศ. 1970 ซึ่งในยุคเริ่มต้นนั้น MATLAB เป็นเพียงส่วนติดต่อของภาษา Fortran เพื่อให้ใช้งานกับ LINPACK (ไลบรารีที่ใช้ในการคำนวณพีชคณิตเชิงเส้น) และ EISPACK (ไลบรารีที่ใช้ในการคำนวณค่าลักษณะเฉพาะ (Eigen Value) และเวกเตอร์ลักษณะเฉพาะ (Eigen vector)) เพื่อให้ผู้ใช้ไม่จำเป็นต้องเรียนรู้การใช้งานภาษา Fortran

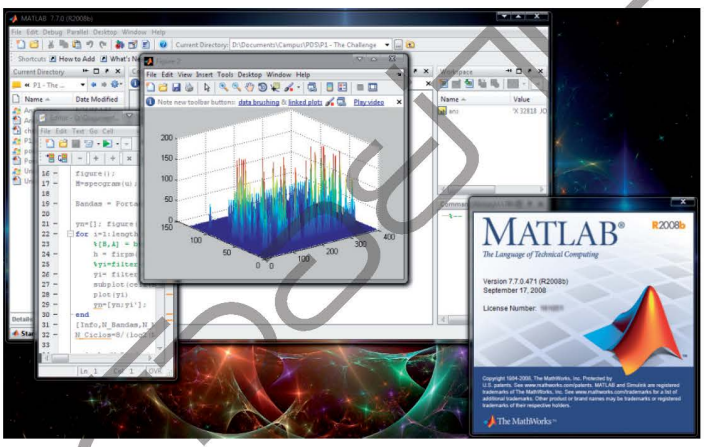
หลังจากนั้นในช่วงปี ค.ศ. 1984 บริษัท MathWorks ถูกก่อตั้งขึ้นเพื่อพัฒนา MATLAB และ MATLAB ถูกเขียนขึ้นใหม่ด้วยภาษา C พร้อมไลบรารี JACKPAC จากนั้น MATLAB ถูกพัฒนามาอย่างต่อเนื่องจนปัจจุบัน MATLAB มี GUI พัฒนาโดยภาษา Java และ Simulink ถูกผนวกเข้ากับ MATLAB



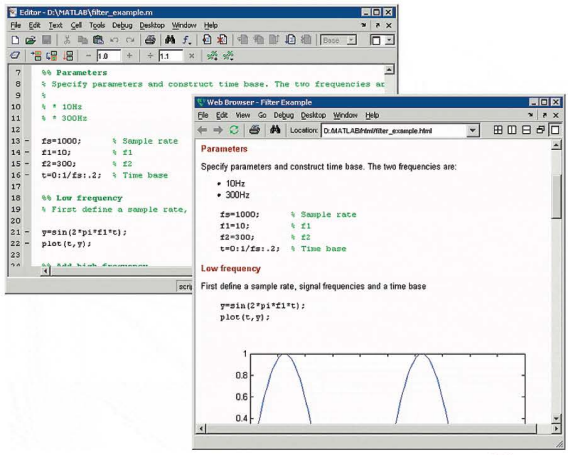
ประโยชน์ของ MATLAB

การพัฒนาโปรแกรมด้วย MATLAB มีความง่ายและเร็วกว่าภาษาอื่นๆ เพราะมีไลบรารีจำนวนมากรองรับ และด้วยตัวลักษณะการทำงานเชิงเมทริกซ์ ทำให้เราสามารถจัดการกับอาร์เรย์ได้ง่ายดาย โค้ดโปรแกรมสั้นกะทัดรัด เหมาะกับการสร้างและทดสอบระเบียบวิธีใหม่ๆ รองรับการทำงานกับกราฟิก รวมถึง GUI ทำให้สะดวกในการป้อนค่าและแสดงผล นอกจากนั้น ยังสามารถติดต่อกับฮาร์ดแวร์และโปรแกรมภาษาอื่นๆ ได้ โดยเราสามารถแบ่งประโยชน์ของ MATLAB ออกเป็น 3 ประเภท ได้แก่

1. MATLAB เป็นโปรแกรมคำนวณ ที่รองรับทั้ง
   a. **เชิงตัวเลข (Numeric)** เราสามารถใช้เป็นเครื่องคำนวณธรรมดา หรือใช้งานฟังก์ชันทางคณิตศาสตร์ขั้นสูงได้
   b. **เชิงสัญลักษณ์ (Symbolic)** เราสามารถคำนวณในเชิงตัวแปรได้ เช่น การอินทิเกรต หรือการแก้สมการต่างๆ แบบติดตัวแปร



2. MATLAB สามารถเขียนเป็นโปรแกรมได้
   a. สามารถเขียนได้ทั้งแบบ Script ซึ่งทำงานในลักษณะชุดคำสั่งต่อเนื่อง หรือเขียนเป็น Function เพื่อใช้งานก็ได้
   b. สามารถใช้งานได้ทั้งแบบ Interpret หรือ Compile โดยเราสามารถ Compile โปรแกรม MATLAB ออกมาได้หลายชนิดทั้งแบบ Standalone หรือ Library เช่น .exe หรือ .dll เป็นต้น
   c. มี GUI รองรับ โดยสามารถเขียนได้ทั้งแบบใช้ GUIDE (คล้าย Visual Basic) หรือแบบไม่ใช้ก็ได้
   d. รองรับการเขียนโปรแกรมเชิงวัตถุทั้งคลาสของ MATLAB เองหรือคลาสของภาษาอื่น เช่น Java หรือ .NET
   e. สามารถ Debug โปรแกรมได้ และในส่วนการติดต่อภาษาอื่นๆ สามารถ Compile ไปเพื่อทำการ Debug ในโปรแกรมอื่น เช่น Visual Studio ได้ด้วย



3. MATLAB สามารถติดต่อหรือใช้งานร่วมกับโปรแกรม ภาษา ฮาร์ดแวร์ หรือแฟ้มข้อมูลรูปแบบต่างๆ ได้
   - a. สามารถเชื่อมต่อกับภาษาหรือโปรแกรมอื่นๆ ได้ เช่น Java, C/C++, .NET, MS Excel โดยเราอาจให้โปรแกรมหลักเขียนโดย MATLAB แล้วเรียกใช้งานภาษาอื่น หรือให้ภาษาอื่นเป็นโปรแกรมหลักแล้วทำการเรียกใช้งาน MATLAB ก็ได้
   - b. สามารถอ่านหรือเขียนแฟ้มข้อมูลสื่อสารแบบมาตรฐานได้ เช่น ข้อความ รูปภาพ เสียง วิดีโอ เป็นต้น
   - c. สามารถติดต่อกับฮาร์ดแวร์ได้ กล้องวิดีโอ บอร์ด DSP เป็นต้น

การติดตั้งโปรแกรม MATLAB

ก่อนที่เราจะติดตั้ง MATLAB เราต้องเลือกรุ่นที่ตรงตามความต้องการของเราก่อน โดยอันดับแรกคือระบบปฏิบัติการ ซึ่ง MATLAB นั้นรองรับทั้ง Unix, Linux, MacOS และ MS Windows

สำหรับระบบปฏิบัติการ Unix หรือ Linux เราสามารถใช้งานแบบ Command line ผ่าน Console ปกติ หรือใช้งาน GUI ผ่าน X-window ก็ได้

หลังจากเลือกระบบปฏิบัติการแล้วให้เรามาเลือกว่า ต้องการใช้ MATLAB แบบ 32 บิต หรือ 64 บิต ทั้งนี้ขึ้นอยู่กับฮาร์ดแวร์ของเราด้วย

ความแตกต่างระหว่างรุ่น x86 และ x64

ใน MATLAB R2012b หรือรุ่น 8.0.0.783 นอกจากจะมีให้เลือกใช้บนระบบปฏิบัติการหลายหลากแล้ว MATLAB ยังมีให้เราเลือกใช้อีก 2 รุ่น ได้แก่ รุ่น x86 กับ x64

สำหรับรุ่น x86 หรือแบบ 32 บิต สามารถติดตั้งได้ทั้งระบบปฏิบัติการแบบ 32 และ 64 บิต ส่วนรุ่น x64 ซึ่งต้องติดตั้งบนระบบปฏิบัติการแบบ 64 บิตเท่านั้น

สำหรับความแตกต่างของระบบทั้งสองแบบนี้ จากประสบการณ์ของผู้เขียนพบว่า สิ่งที่เห็นได้ชัดเจนมีทั้งหมด 3 เรื่องคือ

1. **เวลาในการประมวลผล** ซึ่งจากการทดสอบของผู้เขียนพบว่า รุ่น x86 ทำงานได้เร็วกว่ารุ่น x64 เป็นส่วนใหญ่
2. **หน่วยความจำที่สามารถใช้งานได้** รุ่น x64 สามารถใช้งานหน่วยความจำได้มากกว่ารุ่น x86 อย่างเห็นได้ชัด ซึ่งดูเหมือนจะเป็นเหตุผลหลักที่ทำให้เราต้องใช้รุ่น x64 สำหรับงานที่ต้องการใช้หน่วยความจำในการประมวลผลมากๆ เช่น งานด้านที่เกี่ยวข้องกับสื่อประสม เป็นต้น
3. **จำนวน Toolbox หรือ Blockset ที่แตกต่างกัน** โดย รุ่น x86 จะมี Toolbox หรือ Blockset ครบ แต่ใน รุ่น x64 จะมีน้อยกว่าอยู่จำนวน 1 Blockset ดังตาราง

Toolbox และ Blockset ภายในรุ่น R2012b	x86 Version	x64 Version
Aerospace Blockset	3.10	3.10
Aerospace Toolbox	2.10	2.10
Bioinformatics Toolbox	4.2	4.2
Communications System Toolbox	5.3	5.3
Computer Vision System Toolbox	5.1	5.1
Control System Toolbox	9.4	9.4
Curve Fitting Toolbox	3.3	3.3
DO Qualication Kit	2.0	2.0
DSP System Toolbox	8.3	8.3
Data Acquisition Toolbox	3.2	3.2
Database Toolbox	4.0	4.0

Toolbox และ Blockset ภายในรุ่น R2012b	x86 Version	x64 Version
Datafeed Toolbox	4.4	4.4
Econometrics Toolbox	2.2	2.2
Embedded Coder	6.3	6.3
Filter Design HDL Coder	2.9.2	2.9.2
Financial Instruments Toolbox	1.0	1.0
Financial Toolbox	5.0	5.0
Fixed-Point Toolbox	3.6	3.6
Fuzzy Logic Toolbox	2.2.16	2.2.16
Gauges Blockset	2.0.6	-
Global Optimization Toolbox	3.2.2	3.2.2
HDL Coder	3.1	3.1
HDL Verier	4.1	4.1
IEC Certication Kit	3.0	3.0
Image Acquisition Toolbox	4.4	4.4
Image Processing Toolbox	8.1	8.1
Instrument Control Toolbox	3.2	3.2
MATLAB	8.0	8.0
MATLAB Builder EX	2.3	2.3
MATLAB Builder JA	2.2.5	2.2.5
MATLAB Builder NE	4.1.2	4.1.2
MATLAB Coder	2.3	2.3
MATLAB Compiler	4.18	4.18
MATLAB Distributed Computing Server	6.1	6.1
MATLAB Report Generator	3.13	3.13
Mapping Toolbox	3.6	3.6
Model Predictive Control Toolbox	4.1.1	4.1.1
Model-Based Calibration Toolbox	4.5	4.5
Neural Network Toolbox	8.0	8.0
OPC Toolbox	3.1.2	3.1.2
Optimization Toolbox	6.2.1	6.2.1
Parallel Computing Toolbox	6.1	6.1
Partial Differential Equation Toolbox	1.1	1.1
Phased Array System Toolbox	1.3	1.3

Toolbox และ Blockset ภายในรุ่น R2012b	x86 Version	x64 Version
RF Toolbox	2.11	2.11
Real-Time Windows Target	4.1	4.1
Robust Control Toolbox	4.2	4.2
Signal Processing Toolbox	6.18	6.18
SimBiology	4.2	4.2
SimDriveline	2.3	2.3
SimElectronics	2.2	2.2
SimEvents	4.2	4.2
SimHydraulics	1.11	1.11
SimMechanics	4.1	4.1
SimPowerSystems	5.7	5.7
SimRF	3.3	3.3
Simscape	3.8	3.8
Simulink	8.0	8.0
Simulink 3D Animation	6.2	6.2
Simulink Code Inspector	1.2	1.2
Simulink Coder	8.3	8.3
Simulink Control Design	3.6	3.6
Simulink Design Optimization	2.2	2.2
Simulink Design Verier	2.3	2.3
Simulink Fixed Point	7.2	7.2
Simulink PLC Coder	1.4	1.4
Simulink Report Generator	3.13	3.13
Simulink Verication and Validation	3.4	3.4
Spreadsheet Link EX	3.1.6	3.1.6
Stateow	8.0	8.0
Statistics Toolbox	8.1	8.1
Symbolic Math Toolbox	5.9	5.9
System Identication Toolbox	8.1	8.1
SystemTest	2.6.4	2.6.4
Vehicle Network Toolbox	1.7	1.7
Wavelet Toolbox	4.10	4.10
xPC Target	5.3	5.3
xPC Target Embedded Option	5.3	5.3

NOTE

คำว่า Toolbox หมายถึง ชุดเครื่องมือที่ใช้งานกับ MATLAB ส่วนคำว่า Blockset หมายถึง ชุดเครื่องมือที่ใช้งานกับ Simulink

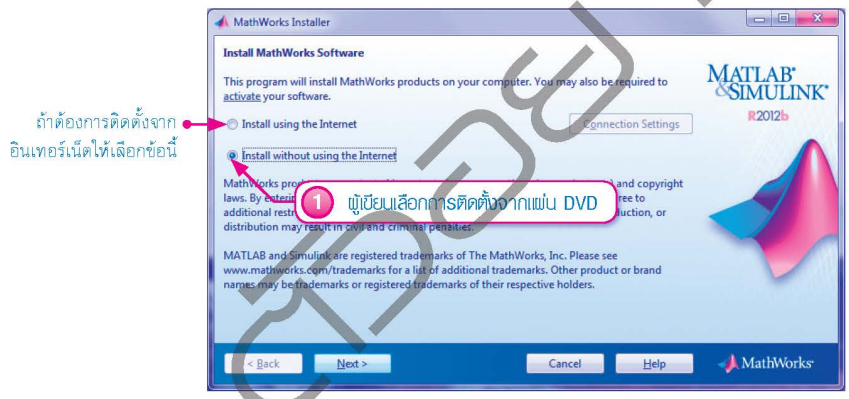
ดังนั้น ขอสรุปหลักการเลือกใช้ระหว่างรุ่น x86 และ x64 ดังนี้

- ตรวจสอบ Toolbox หรือ Blockset ที่เราต้องการใช้งานก่อนว่ารองรับหรือไม่
- ถ้าเป็นงานที่ต้องติดต่อกับฮาร์ดแวร์ทั่วไปควรใช้รุ่น x86
- ถ้าต้องการใช้หน่วยความจำมาก เช่น ต้องสร้างอาร์เรย์ขนาด 10000 x 10000 ควรใช้รุ่น x64
- ถ้าต้องการความเร็วในการประมวลผลสูงควรใช้ x86

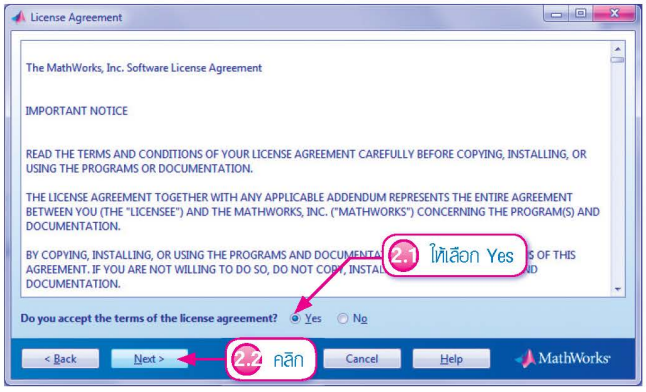
ขั้นตอนการติดตั้งโปรแกรม MATLAB

ภาพประกอบที่ใช้ในที่นี้เป็นการติดตั้ง MATLAB R2012b แบบ x64 บน Microsoft Windows 7 โดยมีขั้นตอนดังต่อไปนี้

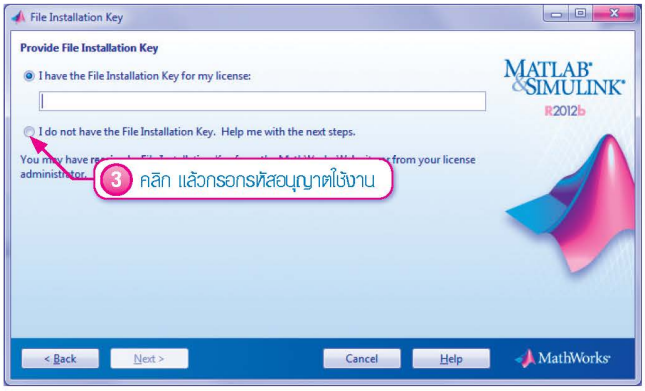
1. ใส่แผ่นติดตั้งในไดรฟ์ DVD แล้วรัน setup.exe จะได้หน้าต่างดังรูป



2. ในหน้าต่าง License Agreement เป็นเงื่อนไข และข้อตกลงในการใช้งาน

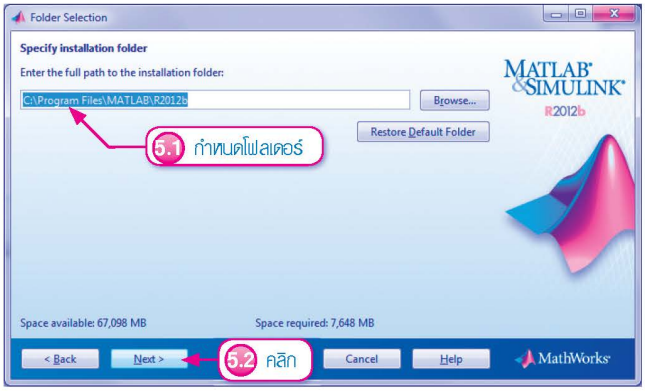


3. ในหน้าต่าง File Installation Key คลิกตัวเลือกแรก แล้วกรอกรหัสอนุญาตใช้งานของเรา

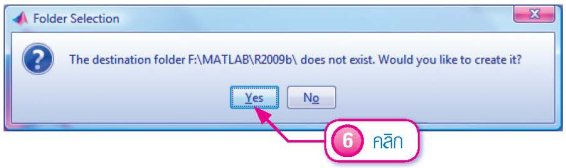


4. ในหน้าต่าง Installation Type ให้เลือกประเภทของการติดตั้ง ถ้าต้องการติดตั้งแบบปกติให้เลือก Typical ถ้าต้องการกำหนด Directory ที่ติดตั้งใหม่ให้เลือก Custom

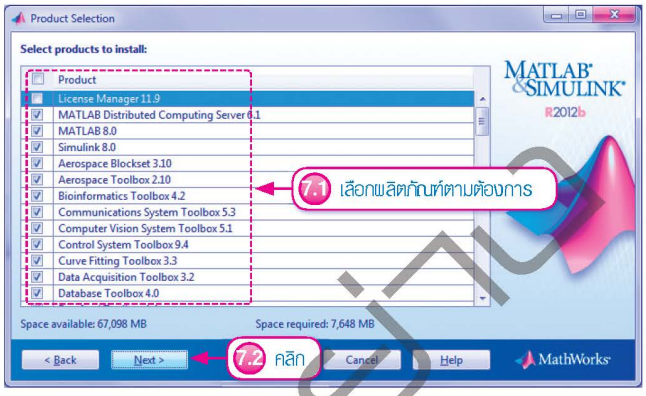
5. กำหนดโฟลเดอร์ที่ต้องการติดตั้ง MATLAB (เราสามารถใช้งาน MATLAB หลายรุ่นบนเครื่องเดียวกันได้โดยการกำหนดโฟลเดอร์ที่ใช้ติดตั้งให้ต่างกัน)



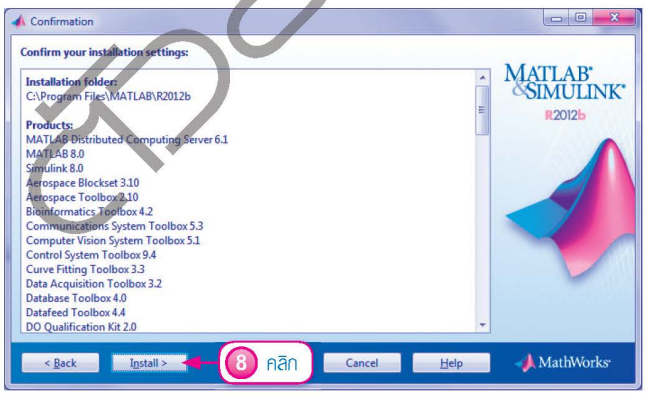
6. คลิก Yes เพื่อยืนยันให้สร้างโฟลเดอร์ตามที่เรากำหนด



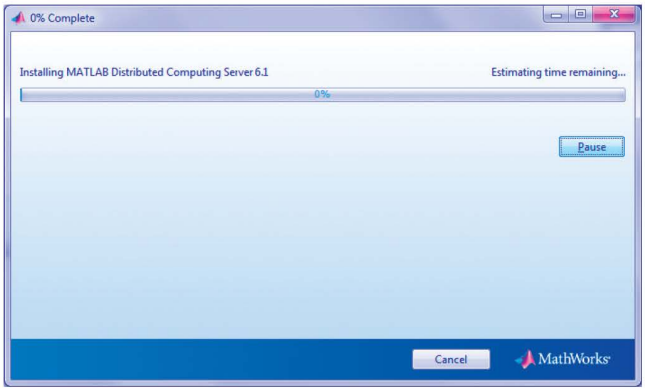
7. ในหน้าต่าง Product Selection ให้เราเลือก Toolbox หรือ Blockset ที่เราต้องการติดตั้ง



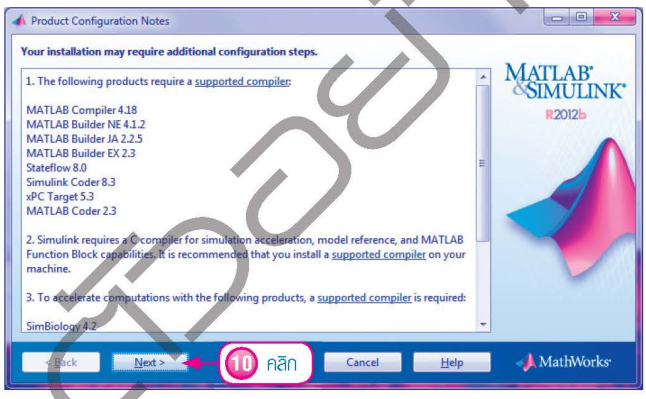
8. ในหน้าต่าง Conmation ให้เราตรวจสอบตัวเลือกในการติดตั้งอีกครั้ง



9. รอสักครู่ ชุดติดตั้งจะติดตั้งองค์ประกอบต่างๆ ตามที่เลือกไว้



10. เมื่อติดตั้งครบ 100% แล้วจะปรากฏหน้าต่าง Product Conguration Notes ซึ่งบอกข้อมูลการตั้งค่าครั้งสุดท้าย ในกรณีที่มีการใช้งานชุดเครื่องมือพิเศษบางตัว เช่น MATLAB Compiler หรือ Builder ต่างๆ



11. ในหน้าต่าง Installation Complete ให้เลือก Activate MATLAB

12. ในหน้าต่าง MathWorks Software Activation เพื่อให้เลือกวิธีการ Activate ซึ่งมี 2 วิธีคือ ผ่านอินเทอร์เน็ต และแบบไม่ผ่านอินเทอร์เน็ต

13. เมื่อการ Activate เสร็จสิ้นจะปรากฏหน้าต่าง Activation Complete ให้คลิกปุ่ม Finish

NOTE

MATLAB รุ่นใหม่จะไม่มีการให้เลือกแฟ้มข้อมูลสกุลต่างๆ เช่น .m .mat .fiffiig ฯลฯ จะถูกเรียกเปิดด้วย MATLAB แบบอัตโนมัติ และหากเราทำการเพิ่ม File Association เหล่านี้เอง จะทำให้การเปิดโปรแกรมและการแสดงผลรูป Icon ทำได้ไม่ถูกต้องนัก สำหรับวิธีการแก้ไขสามารถทำได้ดังนี้

1. หลังจากเปิด MATLAB มาแล้วให้ไปที่หน้าต่าง Command Window ของ MATLAB

2. หลังเครื่องหมาย >> ให้พิมพ์คำสั่งดังนี้

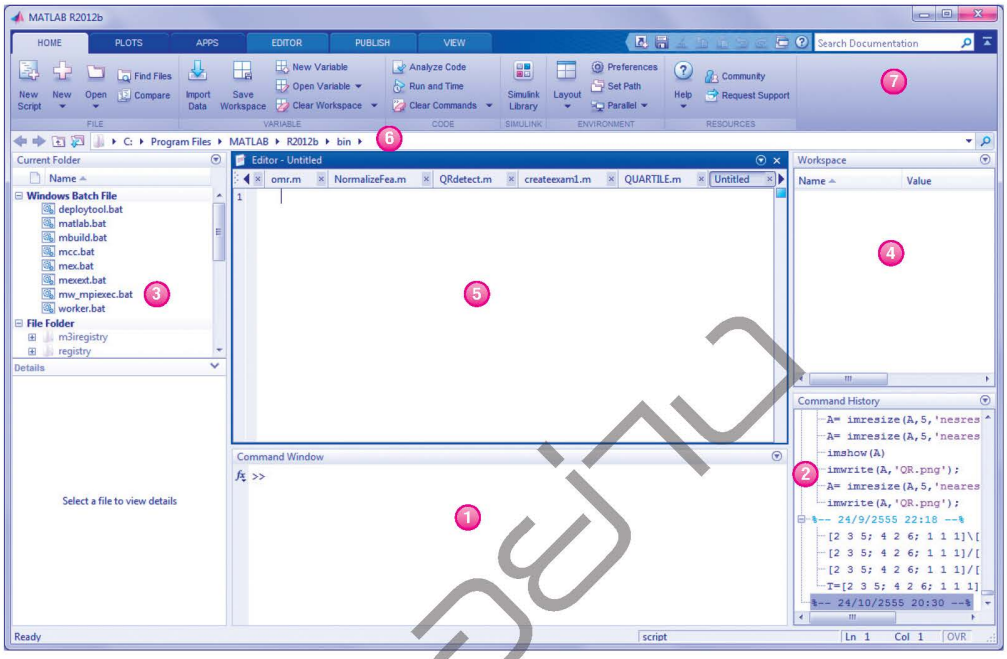
```
>> cwd=pwd; cd([matlabroot '\toolbox\matlab\winfun\private']);
fileassoc('add', {'.m','.mat','.mdl','.fig','.p','.mlprj','.mexw64'})
```

3. จากนั้นกดปุ่ม <Enter>

แฟ้มข้อมูลสกุลต่างๆ ของ MATLAB จะสามารถแสดง Icon และเปิดได้ถูกต้อง ซึ่งบางครั้งอาจต้องทำการ Restart เครื่องคอมพิวเตอร์ใหม่ก่อน

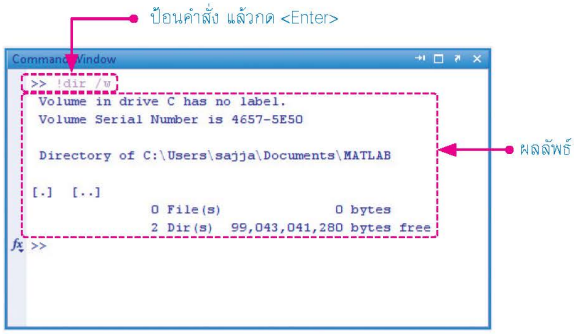
ส่วนติดต่อผู้ใช้ของ MATLAB

ในหน้าต่างหลักของ MATLAB มีหน้าต่างย่อยๆ ซึ่งทำหน้าที่รับค่าและแสดงผลที่แตกต่างกันดังรูป



หน้าต่างย่อยที่สำคัญ ประกอบด้วย

1. **Command Window** เป็นหน้าต่างหลักของ MATLAB เราสามารถเรียกใช้งานคำสั่งต่างๆ และแสดงผลเป็น Standard Input/Output ของ MATLAB โดยมีเครื่องหมาย >> เป็นตัวพร็อมท์ (Command Prompt) นอกจากนี้ เรายังสามารถใช้งานเป็น console ของระบบปฏิบัติการหลักของเราได้ด้วย โดยใช้เครื่องหมาย ! นำหน้าคำสั่ง (System Command) เช่น เราเรียกใช้คำสั่ง dir ของ MS Windows เราสามารถเรียกได้ดังนี้



เราสามารถเรียกใช้ได้โดยใช้ฟังก์ชันหรือ Script เรียกคำสั่ง

```
>> commandwindow
```

2. **Command History** เป็นบันทึกคำสั่งที่ถูกเรียกใช้ใน Command Window เทียบได้กับไดอารีของผู้ใช้ สามารถเรียกดูย้อนหลังได้ เราสามารถเรียกใช้ได้โดยการใช้คำสั่งใน Command Window ดังนี้

```
>> commandhistory
```

TIPS

ในหน้าต่าง Command Window เราสามารถใช้ปุ่มลูกศรขึ้นหรือลงในการเรียกใช้งานคำสั่งย้อนหลังทีละคำสั่งได้ โดยคำสั่งนั้นก็คือ คำสั่งที่เก็บอยู่ในหน้าต่าง Command History นั่นเอง

3. **Current Folder** เป็นหน้าต่างแสดงสารบบของ Current Directory หรือ path ที่เรากำลังทำงานอยู่ เราสามารถเรียกใช้ได้โดยการใช้คำสั่งใน Command Window ดังนี้

```
>> filebrowser
```

4. **Workspace** เป็นหน้าต่างแสดงผลตัวแปรทั้งหมดที่ใช้อยู่ในขณะนั้น เราสามารถเรียกใช้ได้โดยการใช้คำสั่งใน Command Window ดังนี้

```
>> workspace
```

5. **Editor** เป็นหน้าต่างหลักที่ใช้ในการเขียนโปรแกรม คล้าย Text Editor ทั่วไป แต่มีสีแสดง syntax ต่างๆ มี M-lint ช่วยวิเคราะห์โปรแกรม และมีส่วนควบคุมการทำงานของโปรแกรม เราสามารถเรียกใช้ได้โดยการใช้คำสั่งใน Command Window ดังนี้

```
>> workspace
```

6. **Current Directory** แสดงพาธ (path) หรือไดเร็คทอรี (Directory) ที่เรากำลังใช้งานอยู่
7. **Toolstrip** เป็นส่วนเสริมพิเศษ เริ่มมีใช้ใน R2012b เป็นครั้งแรก ซึ่งมีลักษณะคล้ายกับ Ribbon ใน Microsoft Ofce 2007 หรือ 2010 นั่นเอง ซึ่ง Toolstrip นี้ประกอบด้วย 3 เมนูหลักคือ HOME, PLOTS และ APPS โดย
   a. เมนู HOME จะคล้ายกับเมนูของ MATLAB ในรุ่นเก่า ซึ่งจะเกี่ยวข้องกับการเปิด บันทึก นำเข้า และการตั้งค่าต่างๆ
   b. เมนู PLOTS จะเกี่ยวข้องกับการวาดกราฟแบบต่างๆ โดยเมื่อเราคลิกเลือกตัวแปรใน Workspace เมนูนี้จะมีการเปลี่ยนแปลงตามลักษณะของตัวแปรนั้นๆ เช่น ถ้าตัวแปรเป็นอาร์เรย์ 1 มิติ ก็จะแสดงเครื่องมือกราฟในรูปแบบ 2 มิติ หรือถ้าตัวแปรมีลักษณะเป็น 2 มิติ ก็จะแสดงเครื่องมือกราฟในรูปแบบ 3 มิติ เป็นต้น
   c. เมนู APPS เป็นการรวบรวมบรรดา Application ย่อยใน MATLAB เช่น MuPAD (กล่าวถึงในบทที่ 10) มาอยู่ในลักษณะสมัยนิยมตามแบบ Iphone หรือ Android ซึ่งสามารถ Download มาเพิ่มเติมได้จาก MATLAB Exchange (http://www.mathworks.com/matlabcentral/leexchange/index?term=type%3Aapp)

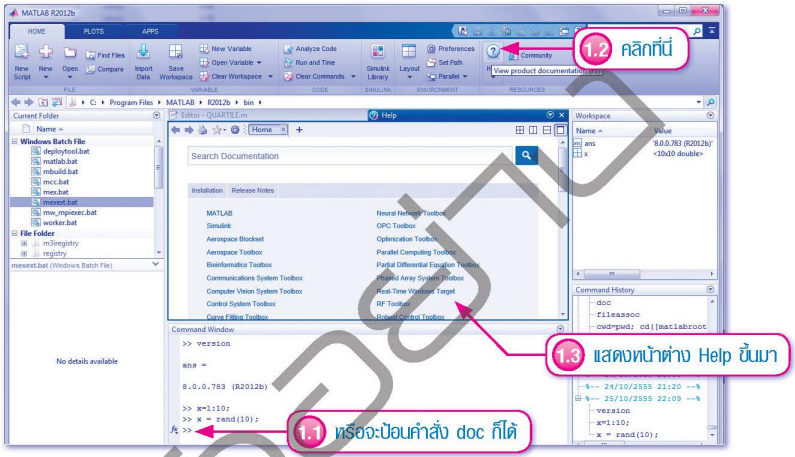
NOTE

หน้าต่างเหล่านี้สามารถเปิดหรือปิดได้ผ่าน Toolstrip ในแถบเมนู HOME หมวด ENVIRONMENT ตัวเลือก Layout แล้วคลิกเพื่อทำเครื่องหมาย ✓ หน้าชื่อหน้าต่างที่ต้องการใช้แสดง และคลิกเพื่อเอาเครื่องหมาย ✓ ออก ในกรณีที่ไม่ต้องการให้แสดง

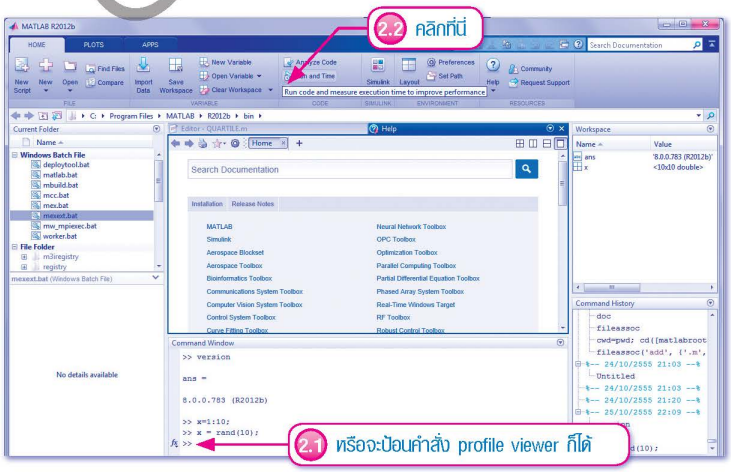
หน้าต่างอื่นๆ ที่ควรรู้จัก

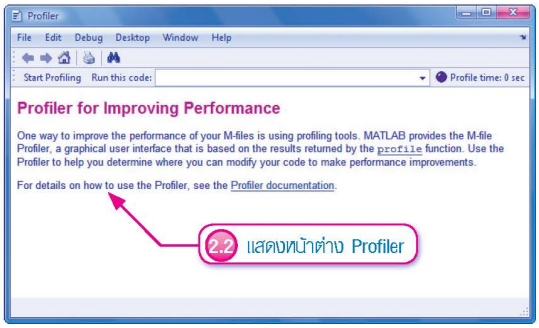
นอกจากนี้ ยังมีหน้าต่างอื่นๆ ที่ควรรู้จักอีก เช่น

1. **Help** เป็นหน้าต่างแสดงคู่มือ และการช่วยเหลือในการใช้งานคำสั่งต่างๆ ใน MATLAB ทั้งหมด เราสามารถเรียกใช้ได้โดยคลิกปุ่ม Help หรือใช้คำสั่ง doc ใน Command Window

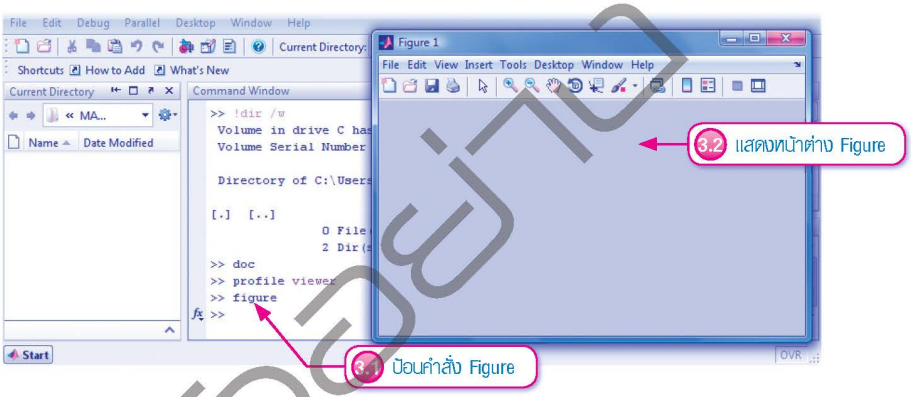


2. **Proler** เป็นหน้าต่างแสดงผลการวิเคราะห์โปรแกรม เช่น ช่วยให้เราทราบว่า โปรแกรมของเรา ควรปรับปรุงในจุดใดเพื่อให้ประมวลผลได้เร็วขึ้น เราสามารถเรียกใช้งานโดยคลิกปุ่ม Run and Time หรือใช้คำสั่ง proler viewer ใน Command Window

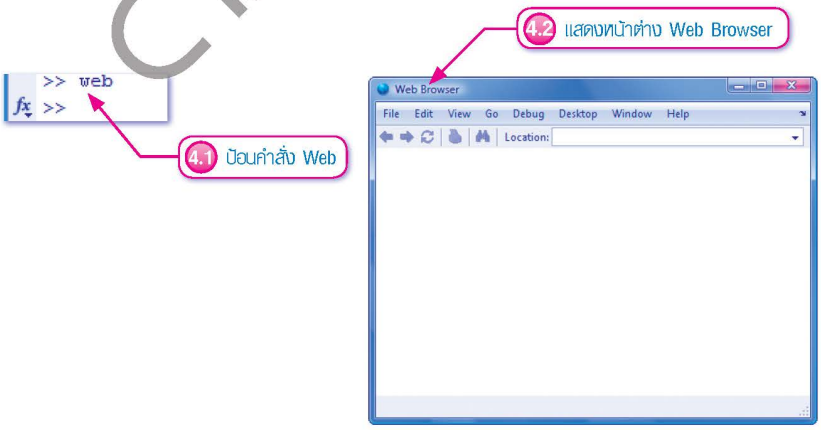




3. **Figures** เป็นหน้าต่างแสดงผลกราฟิกทั้งหมดใน MATLAB เช่น รูปภาพ กราฟ หรือ GUI ซึ่งหน้าต่างนี้จะปรากฏทุกครั้งเมื่อมีการทำงานกับกราฟิก หรือเราอาจเรียกใช้แสดงผลได้ดังนี้



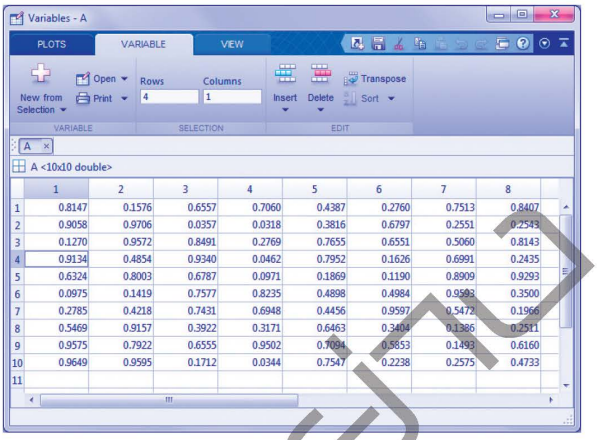
4. **Web Browser** เป็นโปรแกรมย่อยที่สามารถในงานอินเทอร์เน็ตได้เช่นเดียวกับ Internet Explorer เราสามารถเรียกใช้งานได้ดังนี้



5. Variable Editor เป็นโปรแกรมย่อยที่สามารถแสดงผลและแก้ไขชื่อหรือค่าของตัวแปรได้ คล้ายกับ MS Excel ซึ่งเราสามารถเรียกใช้งานผ่านการดับเบิลคลิกที่ item ตัวแปรที่ปรากฏอยู่ใน Workspace หรือผ่าน Command Window ด้วยคำสั่ง

```
>> openvar('A')
```

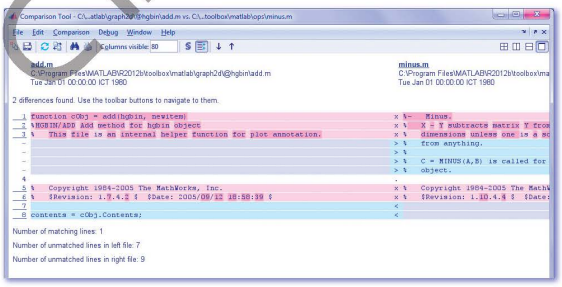
โดย A คือชื่อตัวแปรที่ต้องการแสดงผล



6. File and directory Comparisons เป็นหน้าต่างที่แสดงการเปรียบเทียบความแตกต่างของโค้ดโปรแกรมสองโค้ดเพื่อความสะดวกในการเปรียบเทียบและแก้ไข เราสามารถเรียกใช้ได้โดยการใช้คำสั่งใน Command Window ดังนี้

```
>> visdiff('add.m','minus.m')
```

โดยเราสามารถเปลี่ยน add.m และ minus.m เป็นแฟ้มข้อมูลใดๆ ที่เราต้องการเปรียบเทียบ



TIPS

ทุกหน้าต่างย่อยสามารถแยกออกมาเป็นหน้าต่างแบบอิสระที่ไม่ได้ฝังตัวอยู่ในหน้าต่างหลักของ MATLAB ได้โดยคลิกปุ่ม หรือ (Undock) ภายใต้เมนูย่อยของปุ่ม ที่มุมด้านขวาบนของแต่ละหน้าต่าง

แต่ถ้าเปลี่ยนใจจะนำเข้ามาฝังในหน้าต่างหลักของ MATLAB ก็ทำได้โดยคลิกปุ่ม (Dock) ภายใต้เมนูย่อยของปุ่ม ของแต่ละหน้าต่างเช่นเดียวกัน

การขอความช่วยเหลือ

การขอความช่วยเหลือจาก MATLAB ทำได้ 2 วิธีคือ แบบ GUI และ Command Window ซึ่งการใช้ GUI ดังที่กล่าวไปแล้วคือ เราสามารถเปิด GUI ได้โดยเรียกคำสั่ง

```
>> doc
```

ในกรณีที่เรารู้ชื่อฟังก์ชัน แต่ต้องการวิธีการใช้งาน ให้เราใช้ชื่อฟังก์ชันตามหลังได้เลย เช่น

```
>> doc imshow
```

อย่างไรก็ดีการเรียกใช้หน้าต่าง Help จำเป็นต้องทำงานในระบบปฏิบัติการที่รองรับ GUI แต่ในบางครั้ง เช่น ใน UNIX Linux หรือ การ Telnet เข้าไปรันงาน เราไม่สามารถใช้งานกราฟิกได้ MATLAB จึงมีคำสั่ง help เพื่อแสดงความช่วยเหลือแบบด่วนผ่านทาง Command Window

NOTE

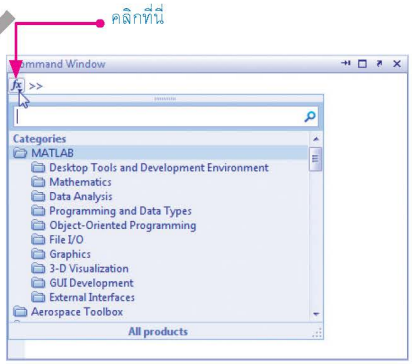
ในกรณีที่ระบบปฏิบัติการนั้นไม่รองรับการแสดงผลแบบกราฟิก เมื่อเราเรียกใช้งาน MATLAB ตัว Console จะทำหน้าที่เป็น Command Window ของ MATLAB เองโดยอัตโนมัติ เช่น การ Telnet เป็นต้น

การใช้ help ทำได้เช่นเดียวกับคำสั่ง doc คือ ตามด้วยชื่อฟังก์ชัน เช่น

```
>> help imshow
```

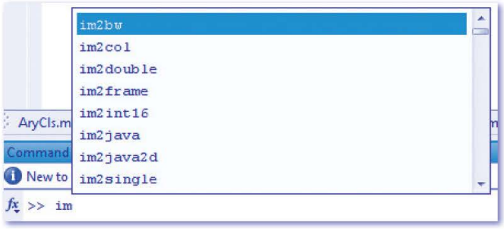
การเรียกใช้งานฟังก์ชันจากตัวช่วยเหลือ

ในกรณีที่ไม่ทราบชื่อฟังก์ชันต่างๆ ใน MATLAB เราสามารถที่จะขอความช่วยเหลือได้จาก GUI ซึ่งสามารถเข้าถึงได้โดยคีย์ลัด <Shift+F1> หรือจากปุ่มฟังก์ชันหน้าเครื่องหมาย (>>) ซึ่งอยู่ในหน้าต่าง Command Window ดังนี้



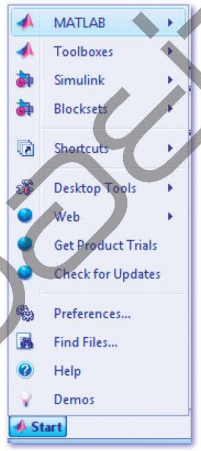
ซึ่งจะแบ่งออกเป็นหมวดหมู่ตามชุดเครื่องมือต่างๆ หรือสามารถทำการค้นหาได้ผ่านช่องข้อความด้านบน

ในกรณีที่ทราบอักษรต้นของชื่อฟังก์ชัน เราสามารถใช้ปุ่ม Tab เพื่อแสดงรายการชื่อฟังก์ชันได้เช่นเดียวกับใน Linux หรือ Visual Basic เช่น กรณีที่เราทราบว่า ฟังก์ชันนำหน้าด้วย im เราจะพิมพ์ im แล้วกด Tab จะปรากฏชื่อฟังก์ชันหรือตัวแปรทั้งหมดใน Workspace ขณะนั้นให้เราเลือกใช้ดังรูป



ปุ่มเริ่ม

ปุ่มเริ่ม (Start Button) มีลักษณะการใช้งานคล้ายปุ่มเริ่มของ MS Windows จะอยู่บริเวณมุมซ้ายล่างของหน้าต่างหลักของ MATLAB ดังรูป



ในปุ่ม Start จะมีรายการเครื่องมือต่างๆ ให้เลือกใช้ เช่น GUI ของโปรแกรมในแต่ละชุดเครื่องมือ Demo ของแต่ละชุดเครื่องมือ การตั้งค่า การใช้งานเว็บบราวเซอร์ ฯลฯ

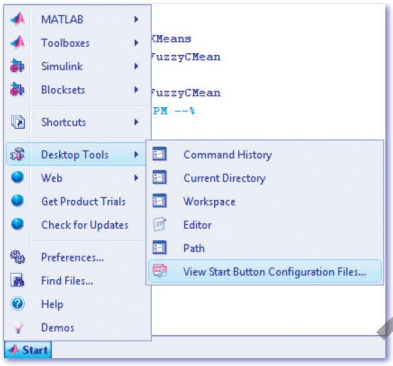
NOTE

ปุ่มเริ่มหรือปุ่ม Start จะมีใน MATLAB รุ่น R2012a ลงไปเท่านั้น สำหรับรุ่น R2012b จะไม่มีปุ่มนี้

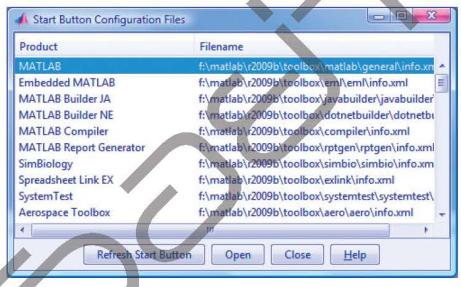
การเพิ่ม ลด หรือแก้ไขไอเท็มภายในเมนู

เราสามารถเพิ่ม ลด หรือแก้ไขไอเท็มภายในเมนูนี้ได้โดย

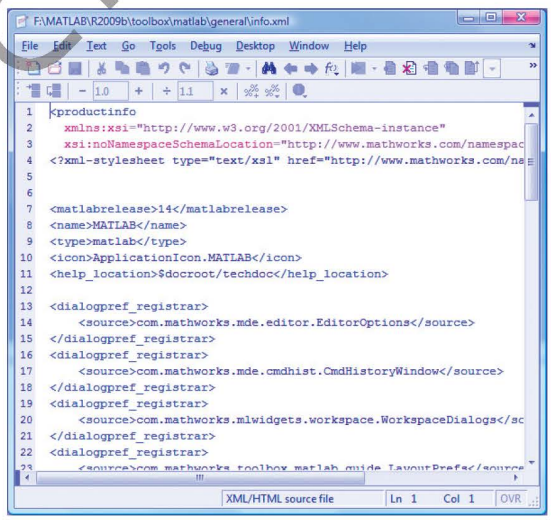
คลิกเลือก Start > Desktop Tools > View Start Button Conguration Files...



เลือกเปิด info.xml ของแต่ละไอเท็มที่ต้องการ โดยคลิกปุ่ม Open หรือดับเบิลคลิก info.xml



จะปรากฏขึ้นในหน้าต่าง Editor ดังนี้



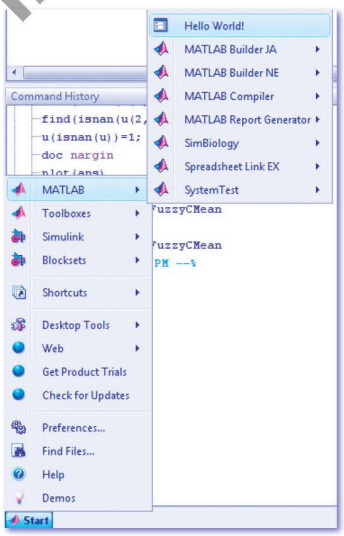
โดย Tag สำคัญต่างๆ มีความหมายดังนี้

XML Tag	ความหมาย
<matlabrelease>	ฉบับที่ของ MATLAB
<name>	ชื่อของชุดเครื่องมือ
<type>	ชนิดของผลิตภัณฑ์ เช่น MATLAB หรือ Simulink
<icon>	ที่เก็บแฟ้มข้อมูลภาพไอคอน
<help_location>	ที่เก็บแฟ้มข้อมูลคู่มือหรือความช่วยเหลือ
<list>	รายการ
<listitem>	ไอเท็ม
<label>	ชื่อไอเท็มที่ต้องการให้ปรากฏในปุ่มเริ่ม
<callback>	คำสั่งภาษา MATLAB ที่จะถูกเรียกให้เหมือนถูกกด

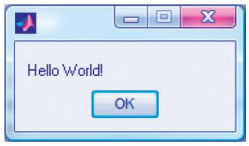
ตัวอย่างการเพิ่มไอเท็มส่วนตัวในปุ่มเริ่มของ MATLAB ภายใต้ Tag <list> ของแฟ้มข้อมูล info.xml ดังนี้

```
<listitem>
<label>Hello World!</label>
<callback>msgbox('Hello World!');</callback>
<icon>ApplicationIcon.GENERIC_GUI</icon>
</listitem>
```

จากนั้น Refresh Start Button จากหน้าต่าง Start Button Conguration Files หรือเปิดโปรแกรม MATLAB ใหม่อีกครั้ง จะปรากฏไอเท็มใหม่ภายในเมนู MATLAB ดังรูป



เมื่อคลิกปุ่ม Hello World! ซึ่งสร้างขึ้นมาใหม่ จะปรากฏกล่องข้อความ Hello World! ดังรูป



โดยผลที่ได้นี้เกิดจากการเรียกใช้งานฟังก์ชัน msgbox ใน <callback>

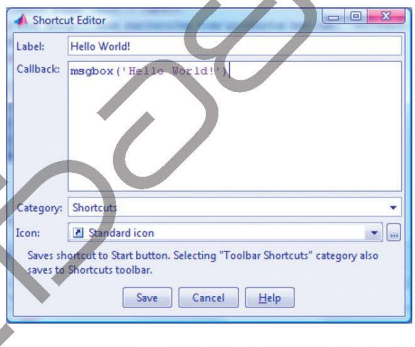
NOTE

msgbox เป็นฟังก์ชันในการแสดงสตริงในกล่องข้อความ คล้ายกับ alert ในภาษา Java หรือ MsgBox ใน Visual Basic หรือ MessageBox ใน Visual C++

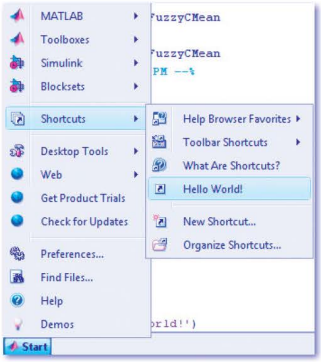
การสร้างทางลัด (Shortcut)

ทางลัดเป็นเหมือน Script ที่สามารถเริ่มใช้ได้โดยการคลิก โดยเราสามารถสร้างได้ 2 วิธี โดยแต่ละวิธีจะปรากฏขึ้นใน 2 บริเวณ ดังนี้

วิธีที่ 1 : ใช้ GUI ที่เปิดได้จากปุ่ม Start > Shortcuts > New Shortcut... จะปรากฏหน้าต่างดังรูป

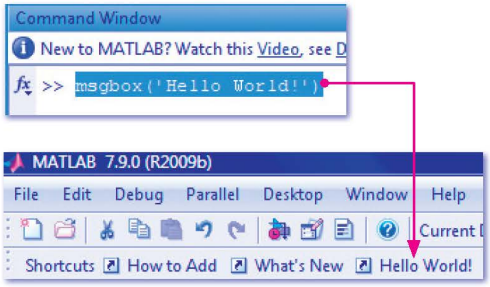


ทดลองเขียน Label และ Callback ดังในรูป แล้วทำการบันทึกด้วยการกดปุ่ม Save จะปรากฏทางลัดใหม่ในเมนูปุ่มเริ่มดังรูป



เมื่อคลิกก็จะปรากฏกล่องข้อความ Hello World! เช่นเดียวกับตัวอย่างข้างต้น

วิธีที่ 2 : คัดลอกคำสั่งโดยการคลิกลากและนำไปวางบริเวณบาร์ทางลัดด้านบนของหน้าต่าง MATLAB หลักดังรูป



จากนั้นให้ตั้งชื่อทางลัดในช่อง Label จะได้ทางลัดดังรูปด้านบน

หลักการใช้งาน MATLAB เบื้องต้น

ขอสรุปการใช้งาน MATLAB เบื้องต้นดังนี้

1. **การคำนวณ รันโปรแกรม เรียกฟังก์ชัน** ใช้กระทำภายในหน้าต่าง Command Window โดยผลที่เกิดจากการคำนวณ หรือผลที่ได้จากการทำงานโปรแกรมใดๆ จะถูกเก็บไว้ในตัวแปรปกติชื่อ ans เสมอ เช่น

2. ตัวแปรทั้งหมดในขณะนั้นจะปรากฏในหน้าต่าง Workspace ซึ่งเราสามารถดับเบิลคลิกเพื่อเรียกดู หรือแก้ไขผ่านหน้าต่าง Variable Editor ได้
3. การเขียนโปรแกรมให้ทำในหน้าต่าง Editor ซึ่งจะถูกบันทึกในนามสกุล .m
4. คำสั่งใดที่เคยใช้งานได้แล้วจะถูกบันทึกไว้ในหน้าต่าง Command History
5. **การเรียกดูคู่มือหรือการช่วยเหลือแบบสมบูรณ์**ให้ใช้คำสั่ง doc แล้วตามด้วยคำสั่งที่ต้องการความช่วยเหลือ
6. **คำสั่ง clear** ใช้ในการลบตัวแปรทั้งหมดใน Workspace
7. **คำสั่ง clc** ใช้ในการลบหน้าจอของหน้าต่าง Command Window

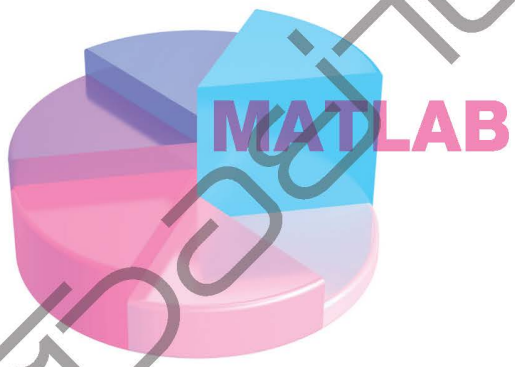
เอกสารอ้างอิง

(1) MathWorks 2012. MATLAB, MATLAB Version 8.0 (R2012b). User's Guide. MathWorks.

(2) MathWorks 2012. Simulink, Simulink Version 8.0 (R2012b). User's Guide. MathWorks.

คำถามท้ายบท

1. หากต้องการใช้งานกับข้อมูลขนาดใหญ่ (Big Data) เราควรเลือกใช้ MATLAB รุ่นใด
2. โดยปกติแล้ว MATLAB ไม่มีคำสั่ง Telnet (คำสั่งที่ใช้ในการเชื่อมต่อในระบบเครือข่าย) ท่านคิดว่าเราสามารถใช้งาน Telnet ผ่านหน้าต่าง Command Window ของ MATLAB ได้หรือไม่ อย่างไร
3. ในกรณีใดบ้างที่เราไม่สามารถใช้งานกราฟิกใน MATLAB ได้
4. หากต้องการทราบชื่อฟังก์ชันที่ใช้ในการคำนวณหาค่าเฉลี่ย เราสามารถทำอย่างไรได้บ้าง
5. หากต้องการทราบตัวอักษรขึ้นต้นของฟังก์ชันที่ต้องการใช้งาน เราจะสามารถทราบชื่อฟังก์ชันเต็มได้ด้วยวิธีการใดบ้าง
6. หากต้องการใช้คำสั่งซ้ำเดิมอีกครั้ง เราสามารถทำอย่างไรได้บ้าง จงบอกมาอย่างน้อย 2 วิธี


MATLAB

Chapter 02

เทนเซอร์และอาร์เรย์

เราอาจจะมองว่า MATLAB เป็นเครื่องคิดเลขที่ใช้ในการคำนวณทางคณิตศาสตร์ หรือเป็นภาษาที่ใช้ในการเขียนโปรแกรมหนึ่งก็ได้

ปริมาณหรือข้อมูลที่ใช้ทั้งการคำนวณทางคณิตศาสตร์ และในการเขียนโปรแกรมคอมพิวเตอร์ก็มีความคล้ายคลึงกัน ซึ่งในคณิตศาสตร์เราเรียกปริมาณหรือข้อมูลนั้นว่า เทนเซอร์ ส่วนในภาษาคอมพิวเตอร์เราเรียกว่า อาร์เรย์

ความเหมือนและแตกต่าง

เทนเซอร์ (Tensor) เป็นปริมาณที่เป็นรูปทั่วไปที่สุดในคณิตศาสตร์ โดยเทนเซอร์อันดับศูนย์คือ สเกลาร์ (Scalar) เทนเซอร์อันดับหนึ่งคือ เวกเตอร์ (Vector) และเทนเซอร์อันดับสองคือ เมทริกซ์ (Matrix) ส่วนเทนเซอร์อันดับสูงกว่าสองไม่มีชื่อเรียกพิเศษ

เราสามารถแทนค่าเทนเซอร์ในอันดับต่างๆ ได้ด้วยอาร์เรย์แบบหลายมิติ ซึ่งอันดับของเทนเซอร์มีค่าเท่ากับจำนวนมิติของอาร์เรย์ โดยสามารถเปรียบเทียบได้ ดังตาราง

อันดับของเทนเซอร์	มิติของอาร์เรย์	ตัวอย่างข้อมูล
0	0	อุณหภูมิ
1	1	ความเร็ว
2	2	ภาพระดับเทา
3	3	ภาพสี
4	4	วิดีโอสี

NOTE

อย่างไรก็ตามอาร์เรย์เป็นเพียงการแทนค่าของเทนเซอร์ แต่มีข้อควรระวังในการใช้งาน ตัวอย่างเช่น การคูณ โดยปกติแล้วการคูณของเมทริกซ์ในทางคณิตศาสตร์จะได้ผล ดังนี้

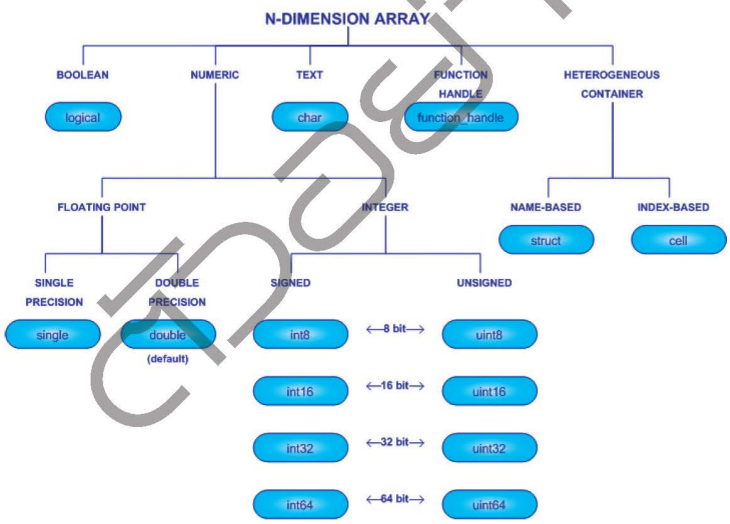
$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \times \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}1x5+2x7 & 1x6+2x8\\3x5+4x7 & 3x6+4x8\end{bmatrix} = \begin{bmatrix}19 & 22\\43 & 50\end{bmatrix}$$

แต่ในการคำนวณของอาร์เรย์เรามักจะคูณกันแบบตัวต่อตัวมากกว่า เช่น

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \times \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix} = \begin{bmatrix}1x5 & 2x6\\3x7 & 4x8\end{bmatrix} = \begin{bmatrix}5 & 12\\21 & 32\end{bmatrix}$$

ชนิดของข้อมูล

ในการใช้ MATLAB เพื่อการคำนวณ หรือเพื่อการเขียนโปรแกรม สิ่งแรกที่เราต้องรู้จักก็คือ ชนิดข้อมูล เราอาจจำแนกแยกแยะข้อมูลที่ใช้งานใน MATLAB ดังแผนภาพต่อไปนี้



เมื่อนำข้อมูลมาเขียนโปรแกรมกับ MATLAB นั้น เราจะใส่มันลงไปในตัวแปร แต่มีความพิเศษคือ MATLAB นั้นไม่จำเป็นต้องประกาศตัวแปร เพราะเมื่อใดก็ตามที่เรามีการเรียกใช้ตัวแปรชนิดตัวเลข **ตัวแปรจะเป็นชนิด double** ทันที

การที่มีชนิดตัวแปรเป็น double มีข้อดีคือ สามารถใช้งานร่วมกับฟังก์ชันของ MATLAB ได้เกือบทุกฟังก์ชัน แต่ก็มีข้อเสียคือ มีขนาดใหญ่มากถึง 64 บิต ในบางครั้งเราไม่ต้องการใช้ค่ามากมายขนาดนี้ ดังนั้นเราควรเลือกชนิดของตัวแปรให้เหมาะสมกับการใช้งาน ดังตาราง

ชนิดข้อมูล	ใช้กับข้อมูล	พิสัยข้อมูล
single	จำนวนจริง	-3.4×10^{38} ถึง 3.4×10^{38}
double	จำนวนจริง	-1.8×10^{308} ถึง 1.8×10^{308}
intn เมื่อ n = 8,16,32,64 {int8, int16, int32, int64}	จำนวนเต็ม	-2^{n-1} ถึง $2^{n-1}-1$
uintn เมื่อ n = 8,16,32,64 {uint8, uint16, uint32, uint64}	จำนวนเต็มศูนย์ และจำนวนเต็มบวก	0 ถึง $2^{n-1}-1$
char	ตัวอักษร หรือข้อความ	
logical	ข้อมูลตรรกะ	true (1) หรือ false (0)

อย่างไรก็ตามค่าพิสัยของข้อมูลเหล่านี้ เราสามารถเรียกดูได้โดยคำสั่ง realmax, realmin, intmax, intmin

NOTE

ใน MATLAB ชนิดของข้อมูล (Data Type) เรียกว่า คลาส (Class)

พิสัยของข้อมูล

บางครั้งในการกำหนดค่าเริ่มต้นในการวนซ้ำ เราอาจต้องการค่าที่สูงสุด หรือต่ำสุดเท่าที่เป็นไปได้ ซึ่งเราสามารถเรียกใช้ค่ามากที่สุด หรือค่าน้อยที่สุดของตัวแปรแต่ละชนิดได้ ดังนี้

ข้อมูลชนิด Double

ตัวอย่างต่อไปนี้เป็นการหาค่าสูงสุดของข้อมูลชนิด Double

```
>> realmax
ans =
   1.7977e+308
```

ส่วนการหาค่าต่ำสุด (ที่เป็นบวก) ทำได้ดังนี้

```
>> realmin
ans =
   2.2251e-308
```

ข้อมูลชนิด Single

ตัวอย่างต่อไปนี้เป็นการหาค่าสูงสุดของข้อมูลชนิด Single

```
>> realmax('single')
ans =
  3.4028e+038
```

ส่วนการหาค่าต่ำสุด (ที่เป็นบวก) ทำได้ดังนี้

```
>> realmin('single')
ans =
  1.1755e-038
```

NOTE

ค่าต่ำสุดที่ได้เป็นค่าต่ำสุดที่เป็นบวก แต่ไม่ใช่ค่าต่ำที่สุดจริงๆ ถ้าต้องการค่าต่ำที่สุดจริงๆ เราต้องเรียกใช้ -realmax (ยิ่งลบมากยิ่งน้อย)

ข้อมูลชนิด Integer

ตัวอย่างต่อไปนี้เป็นการหาค่าสูงสุดของข้อมูลชนิด Integer

```
>> intmax('int8')
ans =
   127
```

ส่วนการหาค่าต่ำสุด ทำได้ดังนี้

```
>> intmin('int8')
ans =
  -128
```

เนื่องจากจำนวนเต็มนั้นมีอยู่หลายรูปแบบ เราสามารถเปลี่ยนจาก int8 เป็น int8, int16, int32, int64, uint8, uint16, uint32 หรือ uint64 ก็ได้ขึ้นอยู่กับชนิดของจำนวนเต็มที่ต้องการ

การสร้าง และการเปลี่ยนชนิดของตัวแปร

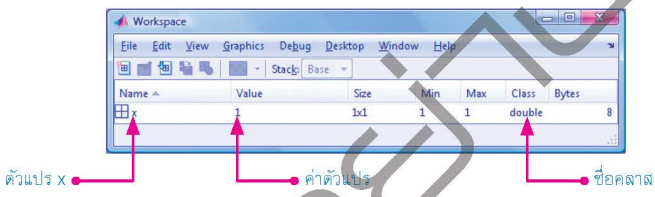
เมื่อจะเขียนโปรแกรม หรือคำนวณ เราต้องสร้างตัวแปรขึ้นมาก่อนเพื่อเก็บตัวเลขที่จะใช้ในการคำนวณ และผลลัพธ์การคำนวณ ซึ่งจะพบบ่อยๆ ว่าเราต้องเปลี่ยนแปลงชนิดของตัวแปรให้เหมาะสมกับข้อมูลที่จะใช้งานภายหลังจากการสร้าง

การสร้างตัวแปรใน MATLAB

เราสามารถสร้างตัวแปรได้โดยตรงโดยไม่ต้องประกาศเหมือนในภาษาอื่นๆ เช่น

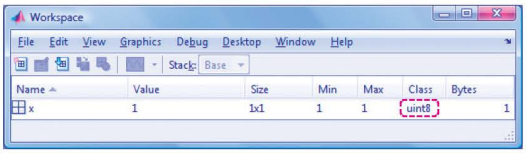
```
>> x=1

x =

     1
```

เมื่อเราไปดูที่หน้าต่าง Workspace เราจะพบว่าตัวแปร x มีค่าเป็น 1 และมีคลาสเป็น double



แต่ถ้าหากต้องการระบุชนิดของตัวแปร เราทำได้โดยใช้ชื่อฟังก์ชันเดียวกับชื่อชนิดของตัวแปรที่ต้องการ เช่น หากเราต้องการสร้าง x=1 เป็นแบบ uint8 เราทำได้โดย

```
>> x=uint8(1)

x =

     1
```



ยกเว้นตัวแปรชนิด logical เราสามารถใช้ฟังก์ชัน logical หรือการกำหนดค่าเป็น true หรือ false ก็ได้ เช่น

```
>> x=true

x =

     1
```

NOTE

- หากเราสร้างตัวแปรที่มีค่าเกินพิสัยของชนิดที่ระบุนั้น MATLAB จะเปลี่ยนค่าเป็นค่าที่ใกล้เคียงที่สุดให้เอง
- ยกเว้นชนิด logical จะเปลี่ยนค่า 0 เป็น false (0) และค่าที่ไม่เท่ากับ 0 ทั้งหมดไม่ว่าจะบวกหรือลบให้เป็น true (1)
- ส่วนชนิด double และ single จะถูกแทนด้วยอนันต์ Inf เมื่อมากเกินพิสัย หรือแทนด้วยลบอนันต์ -Inf เมื่อน้อยเกินพิสัย

ตัวแปรชนิด char (อักขระ 1 ตัว) เราจะสร้างได้โดยใช้เครื่องหมาย ' ' ดังตัวอย่าง

```
>> x = 'a'

x =

a
```

การเปลี่ยนชนิด

ในการเปลี่ยนชนิดของตัวแปร (หรือคลาสของตัวแปร) ทำได้โดยใช้ฟังก์ชันเช่นเดียวกับการสร้างตัวแปร แต่ใส่ Input และ Output เป็นตัวแปรที่เราต้องการเปลี่ยน เช่น หาก x เป็นตัวแปรแบบ char ที่มีอยู่ใน Workspace หากเราต้องการเปลี่ยน x ให้เป็นแบบ uint8 เราทำได้โดย

```
>> x = 'a'

x =

a

>> x = uint8(x)

x =

   97
```

การสร้างอาร์เรย์

หัวข้อที่ผ่านมาเราได้ศึกษาวิธีการสร้างตัวแปรแบบสเกลาร์ไปแล้ว ในหัวข้อนี้เราจะมาสร้างตัวแปรในรูปแบบอาร์เรย์หลายมิติกันดูบ้าง โดยเริ่มจาก

การสร้างอาร์เรย์ 1 มิติ (เวกเตอร์)

เวกเตอร์โดยทั่วไปจะหมายถึง เวกเตอร์หลัก (Column Vector) ซึ่งเราสามารถสร้างได้หลายวิธี เช่น

วิธีที่ 1

```
>> x = [1;2;3;4]

x =

     1
     2
     3
     4
```

วิธีที่ 2

```
>> x = [1 2 3 4]'

x =

     1
     2
     3
     4
```

วิธีที่ 3

```
>> x = [1,2,3,4]'

x =

     1
     2
     3
     4
```

- **เครื่องหมาย [ ]** ใช้บอกขอบเขตของอาร์เรย์
- **เครื่องหมาย ;** ใช้ในการขึ้นแถวใหม่
- **เครื่องหมาย ,** หรือ**เว้นวรรค** ใช้ในการขึ้นหลักใหม่
- **เครื่องหมาย '** ใช้ในการทรานสโพส (Transpose) การเปลี่ยนแถวเป็นหลัก หรือเปลี่ยนหลักเป็นแถว

NOTE

- จริงๆ แล้วเครื่องหมาย ' หมายถึง complex conjugate transpose แต่หากต้องการ transpose จริงๆ สำหรับจำนวนเชิงซ้อนต้องใช้เครื่องหมาย .'
- เวกเตอร์แถวสร้างได้เช่นเดียวกับวิธีที่ 2 และ 3 เพียงแต่ไม่ต้องใส่เครื่องหมาย '

การสร้างอาร์เรย์ 2 มิติ (เมทริกซ์)

เราสามารถสร้างเมทริกซ์ได้โดยการใช้เครื่องหมายที่กล่าวไปแล้วในการสร้างเวกเตอร์มาใช้ร่วมกัน เช่น

```
>> x = [1,2,3; 4,5,6]

x =

     1     2     3
     4     5     6
```

การสร้างเมทริกซ์มากเลขศูนย์ (Sparse Matrix)

เมทริกซ์มากเลขศูนย์ (Sparse Matrix) หมายถึง เมทริกซ์ที่มีเลขศูนย์จำนวนมาก โดยในการเก็บข้อมูลนั้นเราจะเก็บเฉพาะค่าที่ไม่ใช่ศูนย์ และดัชนีของค่าที่ไม่ใช่ศูนย์เท่านั้น ทำให้เหมาะสมในการเก็บเมทริกซ์ขนาดใหญ่ ดังตัวอย่างเช่น

```
>> A = [1 0 0 0 0; 2 1 0 0 0; 0 0 0 0 0; 0 0 1 0 0; 0 0 0 0 9]

A =

     1     0     0     0     0
     2     1     0     0     0
     0     0     0     0     0
     0     0     1     0     0
     0     0     0     0     9
```

เราสามารถสร้างเมทริกซ์มากเลขศูนย์ได้จากฟังก์ชัน sparse ดังนี้

```
B = sparse(A)
```

เช่น

```
>> B = sparse(A)

B =

   (1,1)         1
   (2,1)         2
   (2,2)         1
   (4,3)         1
   (5,5)         9

>> whos
  Name        Size                Bytes  Class     Attributes

  A           5x5                   200  double
  B           5x5                   128  double    sparse
```

จะเห็นได้ว่า ตัวแปร A เดิมมีขนาด 200 ไบต์ แต่ตัวแปร B ซึ่งมีค่าเดียวกับ A เหลือเพียง 128 ไบต์เท่านั้น

ทั้งนี้เราสามารถแปลงเมทริกซ์มากเลขศูนย์กลับไปเป็นเมทริกซ์แบบปกติได้ด้วยคำสั่ง full ดังนี้

```
B = full(A)
```

เช่น

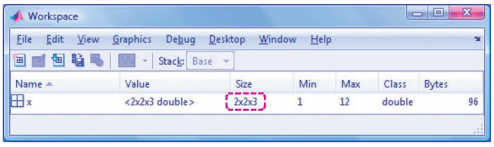
```
>> C = full(B)

C =

     1     0     0     0     0
     2     1     0     0     0
     0     0     0     0     0
     0     0     1     0     0
     0     0     0     0     9

>> whos
  Name        Size                Bytes  Class     Attributes
```

```
  A          5x5                200  double
  B          5x5                128  double    sparse
  C          5x5                200  double
```

การสร้างอาร์เรย์ 3 มิติ (เทนเซอร์อันดับ 3)

เราไม่สามารถสร้างอาร์เรย์อันดับ 3 แบบระบุค่าขึ้นมาได้โดยตรง แต่เราทำได้โดยการสร้างอาร์เรย์ 2 มิติขึ้นมาซ้อนกันทีละชั้น ดังเช่น หากเราต้องการสร้างตัวแปร x ขนาด 2x2x3 เราทำได้โดย

```
>> x(:,:,1) = [1,2; 3,4]

x =

     1     2
     3     4

>> x(:,:,2) = [5,6; 7,8]

x(:,:,1) =

     1     2
     3     4

x(:,:,2) =

     5     6
     7     8

>> x(:,:,3) = [9,10; 11,12]

x(:,:,1) =

     1     2
     3     4

x(:,:,2) =

     5     6
     7     8

x(:,:,3) =

     9    10
    11    12
```

NOTE

สำหรับอาร์เรย์อันดับสูงกว่านี้เราสามารถสร้างได้ในลักษณะเดียวกัน

บางครั้งเราต้องการสร้างอาร์เรย์ขนาดใหญ่หรือมีค่าซ้ำๆ กัน การสร้างอาร์เรย์แบบระบุค่าทุกตัวทำได้ยาก

การสร้างอาร์เรย์แบบลำดับเลขคณิต

อาร์เรย์แบบนี้จะมีการเพิ่มขึ้น หรือลดลงอย่างละเท่าๆ กันทุกช่วง เราจะใช้เครื่องหมาย : ในการสร้างโดยมีวิธีการใช้ ดังนี้

ค่าเริ่มต้น : [ผลต่างร่วม :] ขอบเขตบน

- **ค่าเริ่มต้น** หมายถึง ตัวเลขตัวแรกของลำดับ
- **ผลต่างร่วม** หมายถึง ผลลบระหว่างค่าที่ติดกันสองตัว โดยใช้ตัวหลังตั้งแล้วลบด้วยตัวหน้า ซึ่งถ้าไม่กำหนด MATLAB จะระบุค่าเป็น 1 โดยอัตโนมัติ
- **ขอบเขตบน** หมายถึง ค่าที่ตัวเลขตัวสุดท้ายจะ**ไม่เกินค่านี้**

ตัวอย่างเช่น

- x เป็นลำดับเลขคณิตที่เริ่มจาก 1 จนถึง 10 โดยเพิ่มขึ้นทีละ 1

```
>> x=1:10

x =

     1     2     3     4     5     6     7     8     9    10
```

- x เป็นลำดับเลขคณิตที่เริ่มจาก 1 จนถึง 10 โดยเพิ่มขึ้นทีละ 4

```
>> x=1:4:10

x =

     1     5     9
```

ค่าสุดท้ายไม่ใช่ 10 แต่เป็น 9 เพราะ 10 เป็นค่าขอบเขตบนไม่ใช่ค่าสุดท้ายของลำดับ

- x เป็นลำดับเลขคณิตที่เริ่มจาก 0 จนถึง -10 โดยลดลงทีละ 2

```
>> x=0:-2:-10

x =

     0    -2    -4    -6    -8   -10
```

NOTE

ผลต่างร่วมสามารถเป็นจำนวนเต็มหรือไม่ก็ได้ หากเรากำหนดค่าผลต่างร่วม และขอบเขตบนไม่สัมพันธ์กัน MATLAB จะคืนค่าว่างเปล่า (Empty)

การสร้างอาร์เรย์แบบลำดับแบบอื่น

การสร้างลำดับแบบอื่นๆ เราต้องอาศัยการสร้างอาร์เรย์แบบลำดับเลขคณิตผสมกับตัวดำเนินการทางคณิตศาสตร์ ตัวอย่างเช่น

- x เป็นลำดับ**เรขาคณิต**ที่เริ่มจาก 2 จนถึง 64 โดยเพิ่มขึ้นทีละ 2 เท่า

```
>> x=2.^(1:6)

x =

     2     4     8    16    32    64
```

การใช้ . ไข้เพื่อบอกให้ดำเนินการแบบตัวต่อตัว

- x เป็นลำดับ**ฮาร์โมนิค**ที่เริ่มจาก 1 จนถึง 1/10

```
>> x=rats(1./(1:10))

x =

       1            1/2           1/3           1/4           1/5
  1/6           1/7           1/8           1/9           1/10
```

ฟังก์ชัน rats เป็นการคงรูปเศษส่วนโดยไม่ทำการหารเป็นทศนิยม

การสร้างอาร์เรย์ที่มีค่าเดียวกันทั้งอาร์เรย์

บางครั้งในการกำหนดค่าเริ่มต้นให้ตัวแปร เรานิยมกำหนดค่าเป็น 0 หรือ 1 ซึ่งการจะสร้างอาร์เรย์ที่มี 0 หรือ 1 ทั้งหมด ทำได้ด้วยฟังก์ชัน ones และ zeros ตามลำดับ ดังนี้

```
ones(ขนาดมิติ, ['ชนิดตัวแปร'])
zeros(ขนาดมิติ, ['ชนิดตัวแปร'])
```

ตัวอย่างเช่น

```
>> x=zeros(2,3)

x =

     0     0     0
     0     0     0
```

```
>> x=ones(1,2,3,'uint8')

x(:,:,1) =

     1     1

x(:,:,2) =

     1     1

x(:,:,3) =

     1     1
```

ถ้าเราต้องการให้เป็นค่าอื่น ก็สามารถทำได้โดยการคูณระหว่างฟังก์ชัน ones กับตัวเลขที่ต้องการ เช่น

```
>> x=5*ones(3,2)

x =

     5     5
     5     5
     5     5
```

การสร้างอาร์เรย์ที่มีค่าเกิดจากการสุ่ม

การสุ่มใน MATLAB นั้นมี 2 แบบ ที่ต่างกันตามการกระจายของความน่าจะเป็นคือ การสุ่มแบบ Uniform และการสุ่มแบบ Normal

1. **การสุ่มแบบ Uniform** : คือ การสุ่มที่ทุกๆ ค่าตั้งแต่ 0 จนถึง 1 จะมีโอกาสออกมาเท่าเทียมกัน ทำได้โดยเรียกใช้ฟังก์ชัน rand

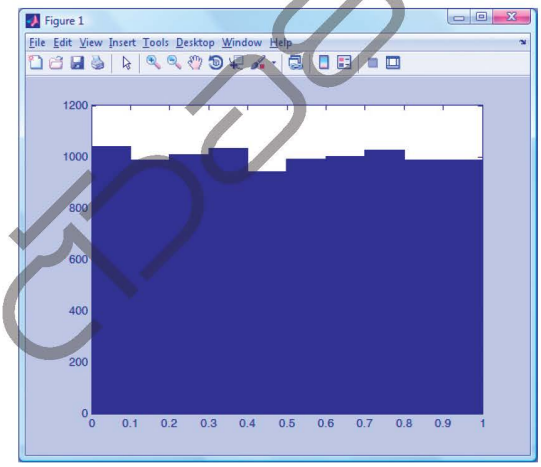
```
rand(ขนาดในมิติ 1, ขนาดในมิติ 2, ขนาดในมิติ 3, ...)
```

ตัวอย่างเช่น

```
>> x = rand(2,3)

x =

    0.8147    0.1270    0.6324
    0.9058    0.9134    0.0975
```

ซึ่งเมื่อเราสร้างฮีสโตแกรมของค่าที่สุ่มจำนวนมากๆ เช่น 10000 ตัว จะได้รูปการกระจายข้อมูลคล้ายรูปสี่เหลี่ยม ดังนี้

```
>> x=rand(10000,1);
>> hist(x)
```



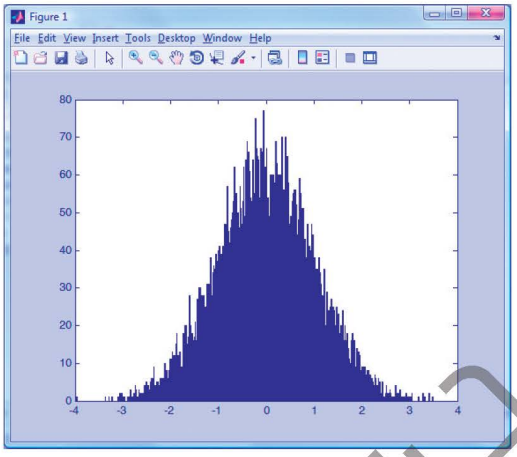
2. **การสุ่มแบบ Normal** : คือ การสุ่มที่ทุกๆ ค่าจะมีโอกาสออกมาไม่เท่าเทียมกัน ซึ่งจะเป็นไปตามเส้นโค้งปกติ โดยค่าที่สุ่มได้ส่วนใหญ่จะอยู่ระหว่าง -4 จนถึง 4 ทำได้โดยเรียกใช้ฟังก์ชัน randn

```
randn(ขนาดในมิติ 1, ขนาดในมิติ 2, ขนาดในมิติ 3, ...)
```

ตัวอย่างเช่น

```
>> x=randn(10000,1);
>> hist(x,500)
```

ซึ่งเมื่อเราสร้างฮีสโตแกรมของค่าที่สุ่มจำนวนมากๆ เช่น 10000 ตัว จะได้รูปการกระจายข้อมูลคล้ายรูประฆังคว่ำ ดังนี้



TIPS

ฟังก์ชัน hist เป็นการสร้างกราฟฮีสโตแกรมของอาร์เรย์ที่ป้อนเข้า โดย hist(x,500) หมายถึง ให้สร้างฮีสโตแกรมของ x จำนวน 500 แท่ง

การสุ่มค่าแบบจำนวนเต็ม

สำหรับการสุ่มค่าแบบจำนวนเต็ม เราอาจทำได้โดย

```
fix(rand*ค่าสูงสุด)+ค่าต่ำสุด
```

ตัวอย่างเช่น ถ้าเราต้องการสุ่มค่า 2 ค่า มาจากช่วงตั้งแต่ 1 ถึง 10 ทำได้โดย

```
>> x = fix(rand(2,1)*10)+1

x =

     5
     2
```

NOTE

ฟังก์ชัน fix ใช้ในการปัดเศษเข้าหาศูนย์ เช่น

fix(2.9) = 2

fix(-2.9) = -2

ข้อควรระวัง fix กับ floor ไม่เหมือนกัน โดยฟังก์ชัน floor เป็นการปัดเศษเข้าหาลบอนันต์ เช่น

floor(-2.9) = -3

การสุ่มตำแหน่ง/การเปลี่ยนลำดับ

การสุ่มในอีกลักษณะหนึ่งคือ การสุ่มตำแหน่ง หรือการเปลี่ยนลำดับ (Permutation) เช่น ถ้าเราต้องการเขียนเกมส์ไพ่ และเราต้องการสับไพ่ก่อน การสับไพ่ก็คือ การเปลี่ยนลำดับชนิดหนึ่ง ไพ่ 1 สำรับมี 52 ใบ ถ้าไพ่ทุกใบมีหมายเลขติดอยู่ตั้งแต่ 1 ถึง 52 เราสามารถสับไพ่สำรับนี้ได้โดยใช้ฟังก์ชัน randperm

randperm(จำนวนเต็มบวก)

```
>> randperm(52)

ans =

  Columns 1 through 16

    47    40    39     3    44    16    17    49     2    25     8    22    26    20    10    18

  Columns 17 through 32

    34    51    48     6    24    29    37    46    28     4    11    52    41    15    32    38

  Columns 33 through 48

    45    31     7    27     1    12     9    36    35    42    50    21    30    19    13    33

  Columns 49 through 52

    23    43    14     5
```

จะเห็นได้ว่า ผลลัพธ์ที่ได้จะมีหมายเลข 1 ถึง 52 แต่สลับตำแหน่งกันอย่างสุ่ม การสุ่มแบบนี้เหมาะกับงานที่ต้องการสุ่มเป็นจำนวนเต็มที่ไม่ต้องการสุ่มซ้ำค่าเก่า

NOTE

นอกจากนี้ ยังมีฟังก์ชันกลุ่มอื่นๆ ที่มากับ Toolbox อื่นๆ เช่น randi สำหรับการสุ่มค่าจำนวนเต็ม เป็นต้น

การอ้างถึงค่าภายในอาร์เรย์

การอ้างถึงค่าภายในอาร์เรย์ใน MATLAB ทำได้ 2 แบบคือ อ้างถึงค่าเดียว หรืออ้างถึงแบบช่วง

การอ้างถึงค่าภายในอาร์เรย์แบบค่าเดียว

เมื่อเราต้องการอ้างถึงค่าของอาร์เรย์ในตำแหน่งใด เราจะใช้เครื่องหมาย ( ) และระบุตำแหน่งโดยใช้เครื่องหมาย , เป็นตัวคั่น ผลลัพธ์ที่ได้คือ สเกลาร์ค่าเดียว ตัวอย่างเช่น

```
>> x = rand(2,3)

x =

    0.4610    0.7574    0.8857
    0.4098    0.0994    0.7772

>> x(2,3)
```

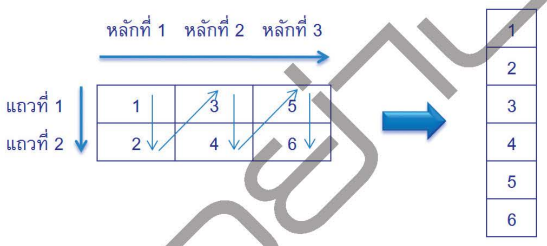
```
ans =

    0.7772
```

การระบุตำแหน่งแบบนี้เรียกว่า การระบุตำแหน่งโดยใช้ Subscripts โดยเพื่อให้มองภาพได้ง่าย จะยกตัวอย่างในลักษณะอาร์เรย์ 2 มิติ ดังภาพ

การเรียก x(2,3) หมายถึง ค่าของ x แถวที่ 2 หลักที่ 3

นอกจากการเรียกตำแหน่งโดยใช้ Subscripts แล้ว เรายังสามารถใช้ Linear Index แทนก็ได้

Linear Index หมายถึง ตำแหน่งของอาร์เรย์ที่ถูกยืดออกเป็นเวกเตอร์หลัก ดังรูป



ดังนั้น x(2,3) สามารถเรียกได้โดย x(6)

เราสามารถเปลี่ยน Subscripts เป็น Linear Index ได้ด้วยฟังก์ชัน sub2ind เช่น

```
>> sub2ind(size(x),2,3)

ans =

     6
```

ฟังก์ชัน size จะบอกจำนวนในแต่ละมิติของอาร์เรย์

และสามารถเปลี่ยน Linear Index เป็น Subscripts ได้ด้วยฟังก์ชัน ind2sub เช่น

```
>> [m n]=ind2sub(size(x),6)

m =

     2


n =

     3
```

ฟังก์ชันใน MATLAB ต่างจากภาษาอื่นคือ สามารถส่งค่าออกไปได้มากกว่า 1 ค่า

การอ้างถึงค่าภายในอาร์เรย์แบบช่วง

เราสามารถใช้การสร้างลำดับที่ได้กล่าวไปแล้วมาเป็นดัชนี เพื่อระบุถึงตำแหน่งของอาร์เรย์เป็นช่วงได้ โดยมีเครื่องหมายสำคัญ ดังนี้

- **เครื่องหมาย** : มี 2 ความหมาย โดยที่
  - ถ้าไม่มีเลขติดจะหมายถึง **เอาทั้งหมดทุกตัว**
  - ถ้ามีตัวเลขติดจะหมายถึง **จาก...ถึง**
- **เครื่องหมาย** , หมายถึง **และตามด้วย** โดยเมื่อต้องการใช้เครื่องหมายนี้ต้องใช้ [ ] ครอบก่อน

ตัวอย่างเช่น ถ้าเรามีตัวแปร x ที่มีค่าดังนี้

```
>> x=magic(4)

x =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

เราสามารถเลือก x ในแถวที่ 2 ได้โดย

```
>> x(2,:)

ans =

     5    11    10     8
```

ซึ่ง x(2,:) หมายถึง เอาค่า x แถวที่ 2 ทุกๆ หลัก

หรือเราสามารถเลือก x ในหลักที่ 3 ได้โดย

```
>> x(:,3)

ans =

     3
    10
     6
    15
```

ซึ่ง x(:,3) หมายถึง เอาค่า x ทุกๆ แถวเฉพาะในหลักที่ 3

NOTE

การใช้ x(:,:) มีค่าเหมือนกับการใช้ x เฉย ๆ ในกรณี x เป็นอาร์เรย์สองมิติ แต่ถ้าเราใช้ x(:) จะหมายถึงการทำ Column Stack Vectorization หรือการยืดออกให้เป็นเวกเตอร์หลัก

```
>> x(:)

ans =

    16
     5
     9
     4
     2
    11
     7
    14
     3
    10
     6
    15
    13
     8
    12
     1
```

หรือเราสามารถเลือก x ในหลักที่ 1 และ 3 ได้โดย

```
>> x(:,[1,3])

ans =

    16     3
     5    10
     9     6
     4    15
```

ซึ่ง x(:,[1,3]) หมายถึง เอาค่า x ทุกๆ แถว เฉพาะในหลักที่ 1 และ 3
ต่อไปนี้เป็นตัวอย่างการใช้งาน

$$\begin{bmatrix}16&2&3&13\\5&11&10&8\\9&7&6&12\\4&14&15&1\end{bmatrix} \Rightarrow x(2:3,2:3) \qquad \begin{bmatrix}16&2&3&13\\5&11&10&8\\9&7&6&12\\4&14&15&1\end{bmatrix} \Rightarrow x(3:4,3:4)$$

$$\begin{bmatrix}16&2&3&13\\5&11&10&8\\9&7&6&12\\4&14&15&1\end{bmatrix} \Rightarrow x(2:4,1:3) \qquad \begin{bmatrix}16&2&3&13\\5&11&10&8\\9&7&6&12\\4&14&15&1\end{bmatrix} \Rightarrow x([1,4],[1,3])$$

การแก้ไขค่าภายในอาร์เรย์

การแก้ไขค่าภายในอาร์เรย์ใน MATLAB ทำได้ 2 แบบ เช่นเดียวกับการอ้างถึงค่าภายในอาร์เรย์ที่ได้กล่าวไปแล้วคือ แก้ไขค่าเดียว หรือแก้ไขแบบช่วง

การแก้ไขค่าภายในอาร์เรย์แบบค่าเดียว

ทำได้เหมือนกับการอ้างถึงค่าภายในอาร์เรย์แบบค่าเดียว โดยกำหนดค่าด้วยเครื่องหมาย =
ตัวอย่างเช่น ถ้าเราต้องการเปลี่ยนค่า x ดังนี้

$$\begin{bmatrix}16&2&3&13\\5&11&10&8\\9&7&6&12\\4&14&15&1\end{bmatrix} \Rightarrow \begin{bmatrix}16&2&3&13\\5&11&0&8\\9&7&6&12\\4&14&15&1\end{bmatrix}$$

เราทำได้โดย

```
>> x(2,3)=0

x =

    16     2     3    13
     5    11     0     8
     9     7     6    12
     4    14    15     1
```

และเช่นเดียวกัน เราสามารถใช้การอ้างดัชนีแบบ Linear Index แทนได้ดังนี้

```
>> x(10)=0

x =

    16     2     3    13
     5    11     0     8
     9     7     6    12
     4    14    15     1
```

การแก้ไขค่าภายในอาร์เรย์แบบช่วง

ทำได้เหมือนกับการอ้างถึงค่าภายในอาร์เรย์แบบช่วง โดยการกำหนดค่าด้วยเครื่องหมาย = แต่ต้องระวังเรื่องขนาดมิติของค่าที่กำหนดให้ต้องมีขนาดมิติเท่ากัน

ตัวอย่างเช่น ถ้าเราต้องการเปลี่ยนค่า x ดังนี้

$$\begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 16 & 2 & 3 & 13 \\ 5 & 1 & 2 & 8 \\ 9 & 3 & 4 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}$$

เราทำได้โดย

```
>> x(2:3,2:3)=[1,2;3,4]

x =

    16     2     3    13
     5     1     2     8
     9     3     4    12
     4    14    15     1
```

การลบค่าภายในอาร์เรย์

การลบค่าภายในอาร์เรย์ทำได้เหมือนกับการแก้ไขค่าภายในอาร์เรย์ แต่เราจะกำหนดค่าให้เป็นว่างเปล่า โดยใช้เครื่องหมาย [ ] โดย MATLAB จะทำการเลื่อนแถว หรือหลักมาชิดกันเองโดยอัตโนมัติ เช่น

```
>> x(:,3) = [ ]

x =

    16     2    13
     5    11     8
     9     7    12
     4    14     1
```

โดยการสั่งแบบนี้เราต้องลบทั้งแถวหรือทั้งหลัก แต่ถ้าหากต้องการลบเพียงบางตัว เราสามารถทำได้โดยการใช้อ้างอิงแบบ Linear Index แต่ผลลัพธ์ที่ได้จะเป็นเวกเตอร์แถวเท่านั้น เช่น

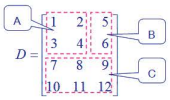
```
>> x(8) = [ ]

x =

    16     5     9     4     2    11     7     3    10     6    15    13     8    12     1
```

การรวมอาร์เรย์

การรวมหรือการต่ออาร์เรย์ทำได้เมื่อขนาดมิติสัมพันธ์กันเท่านั้น ซึ่งทำได้เช่นเดียวกับการสร้างอาร์เรย์ที่ได้กล่าวไปแล้ว ตัวอย่างเช่น

```
>> A=[1,2;3,4]

A =

     1     2
     3     4

>> B=[5;6]

B =

     5
     6

>> C=[7,8,9;10,11,12]

C =

     7     8     9
    10    11    12

>> D=[A,B;C]

D =

     1     2     5
     3     4     6
     7     8     9
    10    11    12
```

นั่นคือ D เกิดจากการรวม A B และ C เข้าด้วยกัน ดังรูป



ตัวอย่าง การสร้างภาพสีด้วยอาร์เรย์สามมิติ

ในตัวอย่างนี้เราจะลองสร้างรูปธงชาติไทย ทำได้ดังนี้

```
>> A=zeros(11,20,3)    % สร้างอาร์เรย์ 3 มิติขนาด 11x20x3
A(:,:,1) =

   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์

เรียนรู้การใช้งาน **MATLAB** สุดยอดโปรแกรมด้านการคำนวณที่ได้รับความนิยมสูงสุดในวงการการศึกษาและการวิจัย โดยเริ่มตั้งแต่พื้นฐานทางคณิตศาสตร์และการเขียนโปรแกรมที่ควรทราบ แล้วจึงก้าวสู่การประยุกต์ใช้งานในรูปแบบต่างๆ ทั้งการสร้าง **GUI**, การแสดงผลด้วยกราฟ, การเขียนไลบรารีเพื่อใช้งานกับภาษาโปรแกรมอื่นๆ และการประมวลผลขั้นสูง ทั้งการประมวลผลแบบกระจายและแบบขนาน

เนื้อหาอ่านง่าย สอดคล้องกับการศึกษาและงานวิจัย พร้อมกับแบบฝึกหัดที่มีให้ช้อมมือทุกบท

- การติดตั้ง **MATLAB**
- แนะนำเทนเซอร์, อาร์เรย์ และการใช้งานสตริงกับ **MATLAB**
- เริ่มต้นเขียนโปรแกรมกับ **MATLAB**
- การเขียนโปรแกรมเชิงวัตถุกับ **MATLAB**
- กราฟิกและการแสดงผลกราฟด้วย **MATLAB**
- เขียนโปรแกรมจัดการ **GUI**
- ใช้งาน **MATLAB** ร่วมกับ **Visual Studio**
- เขียนไลบรารี **MATLAB** เพื่อใช้งานกับภาษา **C/C++, Java**
- การแก้ปัญหาทางคณิตศาสตร์ด้วย **Symbolic Math Toolbox**
- การประมวลผลแบบกระจาย (**Distributed Computing**) และแบบขนาน (**Parallel Computing**)
- การประยุกต์ใช้ **MATLAB**

ผู้แต่ง : ผศ.ดร.ปริญญา สงวนสัตย์

สำเร็จการศึกษา

สาขาวิศวกรรมศาสตรดุษฎีบัณฑิต
(วิศวกรรมไฟฟ้า) จุฬาลงกรณ์มหาวิทยาลัย
ปัจจุบันเป็นหัวหน้าสาขาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์และเทคโนโลยี
สถาบันการจัดการปัญญาภิวัฒน์

งานวิจัยที่เกี่ยวข้อง

Digital Image Processing and Pattern Recognition : Character/Hand-written, face and automatic target recognitions, multimedia processing, image coding and processing, Machine Learning Information retrieval, text categorization.

ผศ.ดร.ปริญญา สงวนสัตย์
บรรณาธิการ **สัจจะ จรัสรุ่งรวีวร**

DEV BOOK

จัดจำหน่ายโดย IDC PREMIER
ISBN 885-916-100-333-5
8 859161 003335
350 บาท