ภาคผนวก ข
พื้นฐานการใช้งานโปรแกรม SCILAB

SCILAB[1] [4, 5] เป็นโปรแกรมภาษาขั้นสูงที่ถูกพัฒนาขึ้นโดยความร่วมมือกันระหว่างนักวิจัยจากสถาบัน Institut National De Recherche En Informatique Et En Automatique (INRIA) และ École nationale des ponts et chausses (ENPC) ประเทศฝรั่งเศส ตั้งแต่ปี ค.ศ. 1990 โดยมีจุดมุ่งหมายเพื่อใช้ในการคำนวณเชิงตัวเลขและแสดงผลกราฟริกที่ซับซ้อน นอกจากนี้โปรแกรม SCILAB ยังเป็นโปรแกรมที่ให้**ฟรี** (ไม่ต้องเสียเงินค่าลิขสิทธิ์ซอฟต์แวร์) ผู้อ่านสามารถดาวน์โหลดได้จาก http://www.scilab.org สำหรับในภาคผนวกนี้จะอธิบายพื้นฐานการใช้งานโปรแกรม SCILAB เพื่อให้ผู้อ่านสามารถทดลองใช้งานตัวอย่างโปรแกรม SCILAB ที่ปรากฏในแต่ละบทเรียนได้ เนื่องจากวิชา "สัญญาณและระบบ" มีสมการทางคณิตศาสตร์จำนวนมากซึ่งบางครั้งทำให้ยากแก่การทำความเข้าใจ ดังนั้นการใช้โปรแกรม SCILAB ควบคู่ไปกับการเรียนรู้ในแต่ละบทเรียนจะช่วยทำให้ผู้เรียนสามารถเข้าใจในบทเรียนได้มากยิ่งขึ้น

ข.1 การสร้างสเกล่าร์ เวกเตอร์ และเมทริกซ์

ค่าสเกลาร์ (scalar) สามารถที่จะถูกกำหนดลงในตัวแปรได้ทันที เช่น ถ้าต้องการกำหนดให้ $a = 2 + 3i$ และ $b = 5$ ก็ทำได้ดังนี้

```
-->a = 2 + 3*%i                    //กำหนดให้ a = 2 + 3i
```

---

[1] SCILAB เป็นโปรแกรมที่สามารถทำงานได้อย่างมีประสิทธิภาพใกล้เคียงกับโปรแกรม MATLAB [6] ซึ่งค่าลิขสิทธิ์ซอฟต์แวร์ของโปรแกรม MATLAB มีราคาแพงมาก แต่ SCILAB เป็นโปรแกรมที่ให้**ฟรี** (freeware) โดยสามารถดาวน์โหลดได้จาก http//www.scilab.org หรือ **http://home.npru.ac.th/piya/webscilab**

```
 a
    2. + 3.i
-->b = 5;                          //กำหนดให้ b = 5
-->
```

โดยที่ `%i` คือค่าคงที่พิเศษที่ใช้ในการแสดงตัวเลขเชิงซ้อน โดยจะมีค่าเท่ากับหน่วยจินตภาพ (imaginary unit) นั่นคือ $\texttt{i} = \sqrt{-1}$ ส่วนเครื่องหมายเซมิโคลอน “`;`” ที่ใช้ปิดท้ายคำสั่งที่สองเป็นการบอกให้โปรแกรมไม่ต้องแสดงผลลัพธ์ออกทางหน้าต่างคำสั่ง และเครื่องหมาย double slash “//” หรือเครื่องหมายคอมเมนต์ (comment) เป็นเครื่องหมายที่จะบอกให้โปรแกรม SCILAB ไม่ทำการประมวลผลต่อคำสั่งหรือข้อความที่อยู่หลังเครื่องหมายคอมเมนต์นี้

ในการใช้งานโปรแกรม SCILAB เครื่องหมายขึ้นบรรทัดใหม่ “`...`” ซึ่งมีลักษณะเป็นจุดที่เรียงต่อกันสามจุด จะมีประโยชน์มากในการเขียนโปรแกรมโดยเฉพาะอย่างยิ่งเมื่อคำสั่งที่ใช้มีความยาวมาก เครื่องหมายนี้เอาไว้ใช้ต่อท้ายคำสั่งเพื่อบอกว่าคำสั่งในบรรทัดนั้นยังไม่สิ้นสุด ดังนั้นถึงแม้ว่าจะกดปุ่ม Enter หลังเครื่องหมายจุดสามจุดนี้ โปรแกรม SCILAB ก็จะยังไม่นำคำสั่งนั้นไปประมวลผล แต่จะรอรับข้อมูลส่วนที่เหลือที่จะเขียนต่อไปในบรรทัดใหม่จนกระทั่งหมดคำสั่งแล้วกดปุ่ม Enter อีกครั้ง จากนั้นโปรแกรม SCILAB จึงจะเอาข้อความทั้งหมดมารวมกันเป็นประโยคคำสั่งเดียวแล้วค่อยนำเอาไปประมวลผล ตัวอย่างเช่น

```
-->x = 5;
-->y = 3;
-->z = x + y              //หาผลบวกของตัวแปร x กับตัวแปร y
 z  =                     //แล้วนำผลลัพธ์ที่ได้ไปบรรจุไว้ในตัวแปรใหม่ที่ชื่อตัวแปร z
    8.
-->z = x + ...            //หมายถึงยังไม่สิ้นสุดคำสั่ง โปรแกรม SCILAB จะยังไม่นำข้อมูลนี้ไปประมวลผล
-->y                      //เมื่อกดปุ่ม Enter ก็ถือว่าเป็นการสิ้นสุดคำสั่งที่ป้อนจากบรรทัดก่อนหน้านี้
 z  =                     //โปรแกรม SCILAB จะนำคำสั่งทั้งหมดคือ z = x + y ไปประมวลผล
    8.
```

จะเห็นได้ว่าผลลัพธ์ที่ได้มีค่าเท่ากัน

โปรแกรม SCILAB ได้เตรียมค่าคงที่พิเศษอื่นๆ เพื่อรองรับการคำนวณทางคณิตศาสตร์ เช่น

- `%pi` คือค่าอัตราส่วนระหว่างความยาวเส้นรอบวงกับเส้นผ่านศูนย์กลางของวงกลม มีค่าเท่ากับ $\pi = 3.1415927...$
- `%e` คือค่าคงที่ตรีโกณมิติ โดยมีค่าเท่ากับ $e = 2.7182818...$
- `%inf` มาจากคำว่า “infinity” คือค่าอนันต์ นั่นคือ `%inf` $= \infty$

- `%eps` มาจากคำว่า “epsilon” คือค่าหน่วยย่อยขนาดเล็กที่สุดที่โปรแกรม SCILAB สามารถรองรับได้ โดยมีค่าเท่ากับ $e \approx 2.22*10^{-16}$ ดังนั้นค่าจำนวนจริงที่มีค่าน้อยกว่าค่า `%eps` โปรแกรม SCILAB จะถือว่าเป็นค่าศูนย์
- `%nan` มาจากคำว่า “Not-A-Number” คือค่าที่ไม่สามารถแสดงให้อยู่ในรูปของตัวเลขได้
- `ans` คือตัวแปรชั่วคราวที่โปรแกรม SCILAB ใช้เก็บผลลัพธ์ที่ได้จากการคำนวณแต่ละคำสั่ง ในกรณีที่ไม่มีตัวแปรมารับค่าผลลัพธ์นั้น

เวกเตอร์ (vector) คือเมทริกซ์ (matrix) ขนาดหนึ่งแถว หรือเมทริกซ์ขนาดหนึ่งแนวตั้ง เวกเตอร์แถว (row vector) สามารถสร้างได้โดยการใช้เครื่องหมายคอมม่า (comma) “,” หรือช่องว่าง (space) เป็นตัวแยกสมาชิกแต่ละสมาชิกในเวกเตอร์แถว ตัวอย่างเช่น

```
-->v = [1, 2, -3]
 v  =
    1.    2.   - 3.
-->v = [1   2   -3]
 v  =
    1.    2.   - 3.
```

ถ้าต้องการทราบว่าเวกเตอร์ `v` มีความยาวเท่าใดหรือมีจำนวนสมาชิกทั้งหมดเท่าใด ก็ทำได้โดยการใช้คำสั่ง `length` ดังนี้

```
 -->length(v)
ans  =
    3.                          //หมายความว่าเวกเตอร์ v มีสมาชิกสามตัว
```

ในขณะที่เวกเตอร์แนวตั้ง (column vector) สามารถสร้างได้โดยการทรานส์โพส (transpose) เวกเตอร์แถวซึ่งจะใช้เครื่องหมาย single quote “ ' ” ตามหลังตัวแปรเวกเตอร์แถว หรือสามารถสร้างเวกเตอร์แนวตั้งขึ้นมาได้โดยตรงโดยการใช้เครื่องหมายเซมิโคล่อนเป็นตัวแยกสมาชิกแต่ละสมาชิกในเวกเตอร์แนวตั้ง เช่น

```
-->v = [1 2 3];             //สร้างเวกเตอร์แถว v
-->v'                       //ใช้ทรานส์โพสกับเวกเตอร์แถวเพื่อให้ได้เป็นเวกเตอร์แนวตั้ง
 ans  =
    1.
    2.
    3.
```

```
-->w = [1; 2; -3]                    //สร้างเวกเตอร์แนวตั้งขึ้นมาโดยใช้เครื่องหมายเซมิโคล่อน
 w  =
    1.
    2.
  - 3.
```

นอกจากการกำหนดค่าโดยตรงให้กับเวกเตอร์แล้ว ผู้ใช้ยังสามารถกำหนดค่าของเวกเตอร์ให้มีค่าเพิ่มขึ้นหรือลดลงแบบอัตโนมัติได้ โดยการใช้เครื่องหมายโคล่อน (colon) “ : ” ซึ่งมีรูปแบบการใช้งานดังนี้

ชื่อตัวแปร = ค่าเริ่มต้น : ค่าที่เพิ่มขึ้น (หรือค่าที่ลดลง) : ค่าสุดท้าย

ในกรณีที่ไม่มีการกำหนดค่าที่เพิ่มขึ้น (หรือค่าที่ลดลง) โปรแกรม SCILAB จะกำหนดให้เป็นค่าที่เพิ่มขึ้นเท่ากับ +1 โดยอัตโนมัติ (ค่าโดยปริยาย) ตัวอย่างเช่น

```
-->z = 1:2:10                        //เริ่มต้นที่ค่า 1 แล้วเพิ่มขึ้นทีละ +2 จนกระทั่งถึงค่าที่มากที่สุดที่ไม่เกิน 10
 z  =
    1.    3.    5.    7.    9.

-->z = 0:5                           //เริ่มต้นที่ค่า 0 แล้วค่าเพิ่มขึ้นทีละ +1 จนกระทั่งถึง 5
 z  =
    0.    1.    2.    3.    4.    5.

-->z = 5:0                           //ค่าเริ่มต้นที่ 5 ไม่สามารถเพิ่มขึ้นทีละ +1 จนถึง 0 ได้ ดังนั้นผลลัพธ์ที่ได้จึงเป็น
 z  =                                //เมทริกซ์ว่าง (empty matrix) นั่นคือมีจำนวนแถวกับจำนวนแนวตั้งเท่ากับศูนย์
     []
```

เมทริกซ์ขนาด m×n คือเมทริกซ์ที่มีจำนวนจำนวน m แถว และ n แนวตั้ง เช่น ถ้าต้องการสร้างเมทริกซ์ขนาด 2×3 สามารถสร้างได้ เช่น

```
-->A = [1 2 3; 4 5 6]
 A  =
    1.    2.    3.
    4.    5.    6.
```

และหากต้องการทราบว่าเมทริกซ์ A มีขนาดเท่าใดก็สามารถทำได้โดยใช้คำสั่ง size ดังนี้

```
-->size(A)
 ans  =
    2.    3.                       //บอกว่าเมทริกซ์ A มีขนาด 2×3 (หรือ 2 แถว และ 3 แนวตั้ง)
```

นอกจากนี้ผู้ใช้สามารถที่จะอ้างถึงสมาชิกแต่ละตัวในเมทริกซ์ได้โดยตรงตามรูปแบบการใช้งานดังนี้

```
-->b = A(2, 3)
 b  =
    6.
```

คำสั่งนี้เป็นการบอกโปรแกรม SCILAB ให้นำค่าของสมาชิกในแถวที่สองและแนวตั้งที่สามของเมทริกซ์ A ไปบรรจุไว้ในตัวแปร b ในทำนองเดียวกันผู้ใช้ยังสามารถที่จะกำหนดค่าให้แก่สมาชิกแต่ละตัวในเมทริกซ์ได้โดยตรง เช่น

```
-->A(2, 3) = 10
 A  =
    1.    2.    3.
    4.    5.    10.
```

ซึ่งเป็นการกำหนดให้ค่าของสมาชิกในแถวที่สองและแนวตั้งที่สามของเมทริกซ์ A มีค่าเป็นค่า 10 ดังนั้นเมทริกซ์ A จึงมีผลลัพธ์ตามที่แสดงไว้ข้างต้น

ข.1.1 การหาทรานส์โพส ดีเทอร์มิแนนต์ อินเวอร์สการคูณ ของเมทริกซ์

ทรานส์โพสเมทริกซ์ (matrix transpose) เป็นการเปลี่ยนแนวตั้งให้เป็นแถว และเปลี่ยนแถวให้เป็นแนวตั้ง โปรแกรม SCILAB สามารถทำการทรานส์โพสเมทริกซ์ได้ 2 รูปแบบ คือ

1) ทรานส์โพสแบบสังยุค (conjugate transpose) จะใช้เครื่องหมาย “ ' ” เป็นตัวดำเนินการ โดยทำหน้าที่สร้างทรานส์โพสเมทริกซ์ พร้อมทั้งทำการสังยุคของตัวเลขเชิงซ้อนด้วย
2) ทรานส์โพสแบบธรรมดา (transpose) จะใช้เครื่องหมาย “ .' ” เป็นตัวดำเนินการ โดยจะสร้างเฉพาะทรานส์โพสเมทริกซ์เท่านั้น

ตัวอย่างการใช้งานเช่น

```
-->A = [1+2*%i; 3; 2-%i];

-->A'                                   //ใช้ทรานส์โพสแบบสังยุค
 ans  =
    1. - 2.i    3.    2. + i            //ทำการสังยุคของตัวเลขเชิงซ้อนด้วย

-->A.'                                  //ใช้ทรานส์โพสแบบธรรมดา
 ans  =
    1. + 2.i    3.    2. - i
```

สำหรับค่าดีเทอร์มิแนนต์ (determinant) และการหาอินเวอร์สการคูณของเมทริกซ์ A สามารถทำได้โดยการใช้คำสั่ง det(A) และ inv(A) ตามลำดับ เช่น

```
-->A = [1 2; 3 4];                    //สร้างเมทริกซ์ A
-->det(A)                             //หาดีเทอร์มิแนนต์ ของเมทริกซ์ A
 ans  =
  - 2.
-->inv(A)                             //หาอินเวอร์สการคูณของเมทริกซ์ A
 ans  =
  - 2.       1.
    1.5    - 0.5
```

ข.2 พหุนาม

โปรแกรม SCILAB จะรองรับพหุนาม (polynomial) ที่มีรูปแบบดังนี้

$$y = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$$

เมื่อ x คือตัวแปรพหุนาม, a = [a_0 a_1 a_2 ... a_n] คือเวกเตอร์ที่มีสมาชิกแต่ละตัวเป็นค่าสัมประสิทธิ์ของพหุนาม, n คือดีกรี (degree) ของพหุนาม, และ y คือสมการพหุนาม ในโปรแกรม SCILAB สมการพหุนามสามารถสร้างได้โดยใช้คำสั่ง poly ดังนี้

```
y = poly(a, "x", [flag])
```

ซึ่งมีรูปแบบการใช้งานอยู่ 2 แบบ คือ

1) ถ้าพารามิเตอร์ a เป็นเวกเตอร์ ผลลัพธ์ที่ได้คือ สมการพหุนาม y ที่ถูกกำหนดโดยพารามิเตอร์ x และ flag เมื่อ x คือตัวแปรพหุนาม และ flag เป็นตัวเลือก (option) ที่มีการเรียกใช้งานดังนี้

   flag = "coeff" ให้สร้างสมการพหุนามจากค่าสัมประสิทธิ์ที่กำหนดโดยเวกเตอร์ a

   flag = "roots" (ค่าโดยปริยาย) ให้สร้างสมการพหุนามจากคำตอบของสมการพหุนามที่กำหนดโดยเวกเตอร์ a

   ตัวอย่างเช่น

```
-->q = poly([1 2 3], "x", "coeff") //สร้างสมการพหุนามจากค่าสัมประสิทธิ์
  q  =
```

```
                  2
      1 + 2x + 3x
```

```
-->p = poly([1  2], "s")          //สร้างสมการพหุนามจากคำตอบของสมการพหุนาม
  p  =
                2
      2 - 3s + s                  //นั่นคือ s = 1 และ s = 2 เป็นคำตอบของ s^2 -3s+2=0
```

```
-->roots(p)                       //คำสั่งที่ใช้หาคำตอบของสมการพหุนาม p
  ans  =
     1.
     2.
```

2) ถ้าพารามิเตอร์ a เป็นเมทริกซ์ ผลลัพธ์ที่ได้คือ สมการลักษณะเฉพาะ (characteristic equation) ของเมทริกซ์ a ซึ่งในทางคณิตศาสตร์สมการลักษณะเฉพาะของเมทริกซ์ A หาได้จากการการแก้สมการ $\det(A-\lambda I)=0$ โดยที่ λ คือค่าคงตัวใดๆ และ I คือเมทริกซ์เอกลักษณ์ที่มีขนาดกับเมทริกซ์ A เช่น ถ้ากำหนดให้เมทริกซ์ $A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$ จะได้ว่าสมการลักษณะเฉพาะของเมทริกซ์ A คือสมการ $\lambda^2-5\lambda-2=0$ โปรแกรม SCILAB สามารถหาสมการลักษณะเฉพาะของเมทริกซ์ A ได้โดยใช้คำสั่ง poly ดังนี้

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

```
-->y = poly(A, "x")
  y  =
                2
  - 2 - 5x + x                    //ผลลัพธ์เท่ากันกับสมการ λ^2 -5λ-2=0 เมื่อแทนค่า x=λ
```

ข.3 การดำเนินการทางคณิตศาสตร์

การคำนวณทางคณิตศาสตร์ระหว่างค่าสเกลาร์กับค่าสเกลาร์ เครื่องหมายที่ใช้ในการคำนวณกับเครื่องหมายที่ใช้ในโปรแกรม SCILAB จะต่างกันเล็กน้อย ดังที่แสดงในตารางที่ ข.1 ตัวอย่างเช่น

```
-->a = 3;
```

```
-->b = 2;
```

```
-->M = [a+b, a-b, a*b; a\b, a/b, a^b]
 M  =
    5.          1.      6.
    0.6666667   1.5     9.
```

ตารางที่ ข.1 การดำเนินการที่ใช้ในการคำนวณทางคณิตศาสตร์ของค่าสเกลาร์

การดำเนินการ	รูปแบบพีชคณิต	รูปแบบของ SCILAB
การบวก (addition)	$a + b$	`a + b`
การลบ (subtraction)	$a - b$	`a - b`
การคูณ (multiplication)	$a \times b$	`a * b`
การหารซ้าย (left division)	$\frac{b}{a}$	`a \ b`
การหารขวา (right division)	$\frac{a}{b}$	`a / b`
การยกกำลัง (exponentiation)	a^b	`a^b` หรือ `a**b`

ตารางที่ ข.2 ตัวดำเนินการที่ใช้ในการคำนวณทางคณิตศาสตร์ของเมทริกซ์

ตัวดำเนินการ	คำอธิบาย
+	การบวก (addition)
–	การลบ (subtraction)
*	การคูณ (multiplication)
.*	การคูณในระดับสมาชิก (element-wise multiplication)
.*.	การคูณแบบโครเนคเกอร์ (Kronecker product)
\	การหารซ้าย (left division)
.\	การหารซ้ายในระดับสมาชิก (element-wise left division)
.\.	การหารซ้ายแบบโครเนคเกอร์ (Kronecker left division)
/	การหารขวา (right division)
./	การหารขวาในระดับสมาชิก (element-wise right division)
./.	การหารขวาแบบโครเนคเกอร์ (Kronecker right division)
^ หรือ **	การยกกำลัง (exponentiation)
.^	การยกกำลังในระดับสมาชิก (element-wise exponentiation)

ในขณะที่ตัวดำเนินการที่ใช้ในการคำนวณทางคณิตศาสตร์สำหรับเมทริกซ์ แสดงในตารางที่ ข.2 ตัวอย่างเช่น

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

```
-->A + B
ans  =
    2.    3.    4.
    3.    4.    5.
-->A - B
 ans  =
    0.    1.    2.
    5.    6.    7.
-->A * B                          //ขนาดของเมทริกซ์ไม่สอดคล้องกับกฎการคูณกันของเมทริกซ์
      !--error 10
inconsistent multiplication
-->A * B'
 ans  =
    6.   - 6.
    15.  - 15.
-->A .* B                         //การคูณในระดับสมาชิก
 ans  =
    1.    2.    3.
  - 4.  - 5.  - 6.
-->A .\ B                         //การหารซ้ายในระดับสมาชิก
 ans  =
    1.       0.5    0.3333333
  - 0.25   - 0.2  - 0.1666667
-->A .^ B                         //การยกกำลังในระดับสมาชิก
 ans  =
    1.       2.     3.
    0.25     0.2    0.1666667
```

จะเห็นได้ว่าการดำเนินการต่างๆ ในระดับสมาชิกจะเกิดขึ้นได้ก็ต่อเมื่อเมทริกซ์ทั้งสองจะต้องมีขนาดเท่ากัน

ข.4 การแก้ระบบสมการเชิงเส้น

โปรแกรม SCILAB สามารถนำมาใช้แก้ไขปัญหาระบบสมการเชิงเส้น (linear equation system) ได้โดยง่าย ตัวอย่างเช่น ถ้าต้องการแก้สมการสองตัวแปรเพื่อหาค่าของตัวแปร x_1 และ x_2 จาก

$$2x_1 + x_2 = 3 \qquad (ก1)$$

$$x_1 - x_2 = 3 \qquad (ก2)$$

ถ้าใช้หลักการแก้สมการสองตัวแปรทั่วไปเพื่อแก้สมการที่ (ก1) และ (ก2) จะได้ผลลัพธ์คือ $x_1 = 2$ และ $x_2 = -1$ เช่นเดียวกันผู้ใช้สามารถแก้สมการทั้งสองนี้ได้โดยใช้หลักการของเมทริกซ์ดังนี้

สมการที่ (ก1) และ (ก2) สามารถเขียนสมการทั้งสองให้อยู่ในรูปของเมทริกซ์ได้คือ

$$\begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \quad \text{หรือ} \quad Ax = b \qquad (ก3)$$

โดยที่ $A = \begin{bmatrix} 2 & 1 \\ 1 & -1 \end{bmatrix}$ และ $b = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$ เนื่องจากคำตอบของสมการ $Ax = b$ คือ $x = A^{-1}b$ ดังนั้นคำตอบของสมการที่ (ก1) และ (ก2) สามารถหาได้จากการใช้ชุดคำสั่งดังนี้

```
-->A = [2 1; 1 -1];          //เมทริกซ์ขนาด 2×2
-->b = [3; 3];               //เวกเตอร์แนวตั้งขนาด 2×1
-->x = inv(A)*b              //คำตอบของสมการสองตัวแปรเป็นเวกเตอร์แนวตั้งขนาด 2×1
 x  =
    2.
  - 1.
```

ผลลัพธ์ที่ได้คือ $x_1 = 2$ และ $x_2 = -1$

ข.5 การดำเนินการทางคณิตศาสตร์

โปรแกรม SCILAB มีฟังก์ชันที่ใช้ในการคำนวณทางคณิตศาสตร์จำนวนมาก เช่น ฟังก์ชันพื้นฐานที่เกี่ยวกับตัวเลข, ฟังก์ชันตรีโกณมิติ, ฟังก์ชันพื้นฐานทางสถิติ เป็นต้น โดยมีรายละเอียดดังต่อไปนี้

ข.5.1 ฟังก์ชันพื้นฐานที่เกี่ยวกับตัวเลข

ตารางที่ ข.3 แสดงฟังก์ชันพื้นฐานที่เกี่ยวกับตัวเลข ตัวอย่างเช่น

```
-->abs([1, %i, -2, -2*%i, 3+4*%i])
 ans  =
    1.    1.    2.    2.    5.
-->sqrt([2, 4, -1, -4])
 ans  =
    1.4142136    2.    i      2.i
-->real([0.1, %i, -1.5+2*%i, 2-%i])
 ans  =
    0.1    0.  - 1.5    2.
```

ตารางที่ ข.3 ตัวอย่างฟังก์ชันพื้นฐานที่เกี่ยวกับตัวเลข

ฟังก์ชัน	คำอธิบาย
`abs(x)`	หาค่าสัมบูรณ์ (absolute value) ของตัวแปร `x`
`sqrt(x)`	หาค่ารากที่สอง (square root) ของตัวแปร `x`
`modulo(m,n)`	หาค่าเศษที่เหลือการหารตัวแปร `n` ด้วย `m`
`ceil(x)`	หาค่าจำนวนเต็มที่มีค่าใกล้กับค่า `x` ไปทางค่า ∞ มากที่สุด
`floor(x)`	หาค่าจำนวนเต็มที่มีค่าใกล้กับค่า `x` ไปทางค่า $-\infty$ มากที่สุด
`sign(x)`	หาค่าเครื่องหมายของตัวแปร `x`
`roots(p)`	หาค่ารากหรือคำตอบของสมการพหุนาม `p`
`real(x)`	หาค่าจำนวนจริงของตัวแปร `x`
`imag(x)`	หาค่าจำนวนจินตภาพของตัวแปร `x`
`conj(x)`	หาค่าสังยุคของจำนวนจำนวเชิงซ้อนของตัวแปร `x`
`exp(x)`	หาค่า e^x ของตัวแปร `x`
`log(x)`	หาค่า `log` ฐาน `e` ของตัวแปร `x`
`log2(x)`	หาค่า `log` ฐาน 2 ของตัวแปร `x`
`log10(x)`	หาค่า `log` ฐาน 10 ของตัวแปร `x`

```
-->imag([0.1, %i, -1.5+2*%i, 2-%i])
 ans  =
    0.    1.    2.   - 1.
-->log([1, %e, 10, 20, 100])
 ans  =
    0.    1.    2.3025851    2.9957323    4.6051702
-->log10([1, 2, 10, 20, 100])
 ans  =
    0.    0.30103    1.    1.30103    2.
```

ข.5.2 ฟังก์ชันตรีโกณมิติ

โปรแกรม SCILAB ได้เตรียมคำสั่งพื้นฐานสำหรับฟังก์ชันตรีโกณมิติและฟังก์ชันตรีโกณมิติผกผันไว้ ตามตารางที่ ข.4 โดยค่ามุมที่ใช้หรือที่ได้รับจากฟังก์ชันทางตรีโกณมิติจะต้องมีหน่วยเป็นเรเดียน (radian) ตัวอย่างเช่น

```
-->y = sin([0, 1, %pi/2, -%pi/2])
 y  =
    0.    0.8414710    1.   - 1.
```

ตารางที่ ข.4 ฟังก์ชันตรีโกณมิติและฟังก์ชันตรีโกณมิติผกผัน

ฟังก์ชัน	คำอธิบาย
`sin(x)`	หาค่า sine ของตัวแปร `x`
`cos(x)`	หาค่า cosine ของตัวแปร `x`
`tan(x)`	หาค่า tangent ของตัวแปร `x`
`asin(y)`	หาค่า sine inverse ของตัวแปร `y`
`acos(y)`	หาค่า cosine inverse ของตัวแปร `y`
`atan(y)`	หาค่า tangent inverse ของตัวแปร `y`

```
-->x = asin(y)
 x  =
    0.    1.    1.5707963   - 1.570796        //มีหน่วยเป็นเรเดียน

-->y = tan([0, %pi/6, %pi/4, %pi/3])
 y  =
    0.    0.5773503    1.    1.7320508

-->x = atan(y)
 x  =
    0.    0.5235988    0.7853982    1.0471976      //มีหน่วยเป็นเรเดียน
```

ข.5.3 ฟังก์ชันพื้นฐานทางสถิติ

โปรแกรม SCILAB ได้เตรียมฟังก์ชันสำหรับการใช้งานทางด้านสถิติ ตามที่แสดงในตารางที่ ข.6 ตัวอย่างเช่น

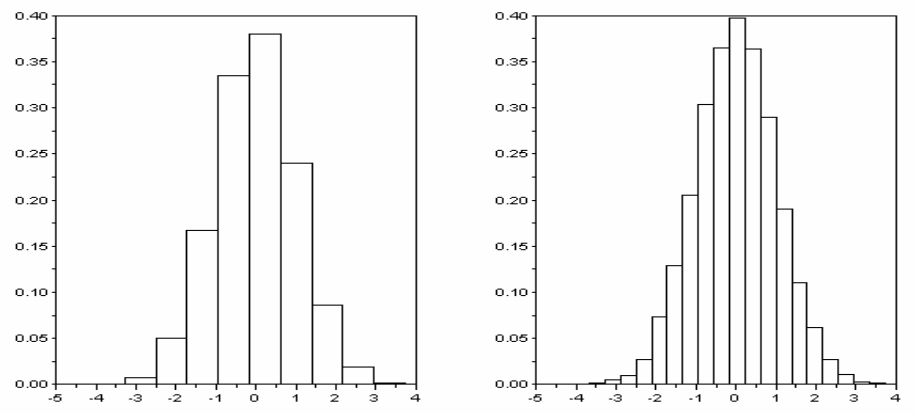
```
-->x = [3 1 4 9 2 5 7 8 6];
-->[min(x), max(x), mean(x), median(x), sum(x)]
 ans  =
    1.    9.    5.    5.    45.
-->sort(x)
 ans  =
    9.    8.    7.    6.    5.    4.    3.    2.    1.
```

ในการใช้งานทางด้านสถิติ คำสั่งที่ใช้บ่อยครั้ง คือ คำสั่ง `histplot(n,x)` ซึ่งเป็นคำสั่งที่ใช้ในการวาดรูปฮิสโตแกรม (histogram) ของค่าทั้งหมดในเวกเตอร์ `x` เป็นจำนวน `n` ช่วงระหว่างค่าต่ำสุดและค่าสูงสุดของเวกเตอร์ `x` ตัวอย่างเช่น

```
-->d = rand(1, 10000, 'normal');
```

ตารางที่ ข.6 ตัวอย่างฟังก์ชันพื้นฐานทางสถิติ

ฟังก์ชัน	คำอธิบาย
`min(x)`	หาค่าต่ำสุด (minimum) ของตัวเลขทั้งหมดในตัวแปร `x`
`max(x)`	หาค่าสูงสุด (maximum) ของตัวเลขทั้งหมดในตัวแปร `x`
`mean(x)`	หาค่าเฉลี่ย (mean) ของตัวเลขทั้งหมดในตัวแปร `x` (ถือเป็นค่าเฉลี่ยเลขคณิต)
`median(x)`	หาค่ามัธยฐาน (median) ของตัวเลขทั้งหมดในตัวแปร `x`
`sum(x)`	หาค่าผลบวกของตัวเลขทั้งหมดในตัวแปร `x`
`prod(x)`	หาค่าผลคูณของตัวเลขทั้งหมดในตัวแปร `x`
`sort(x)`	เรียงลำดับตัวเลขทั้งหมดในตัวแปร `x` จากค่ามากไปหาค่าน้อย
`histplot(n,x)`	วาดรูปฮิสโตแกรม (histogram) ของค่าทั้งหมดในเวกเตอร์ `x` เป็นจำนวน `n` ช่วงระหว่างค่าต่ำสุดและค่าสูงสุดของเวกเตอร์ `x`
`variance(x)`	หาค่าความแปรปรวน (variance) ของตัวเลขทั้งหมดในตัวแปร `x`
`geomean(x)`	หาค่าเฉลี่ยเรขา (geometric mean) ของตัวเลขทั้งหมดในตัวแปร `x`



รูปที่ ข.1 ตัวอย่างรูปฮิสโตแกรม

```
-->subplot(1,2,1);  histplot(10, d);          //รูปที่ ข.1 ด้านซ้าย
-->subplot(1,2,2);  histplot(20, d);          //รูปที่ ข.1 ด้านขวา
```

คำสั่งแรกจะทำการสร้างจำนวนสุ่ม (random number) จำนวน 10000 ตัว (บรรจุไว้ในเวกเตอร์ขนาด 1×10000) โดยมีลักษณะการแจกแจงปรกติ (normal distribution) หรือการแจกแจงแบบเกาส์เซียน (Gaussian) นั่นคือมีค่าเฉลี่ย (mean) เท่ากับค่า 0 และมีค่าความแปรปรวน (variance) เท่ากับค่า 1 จากนั้นก็ทำการวาดรูปฮิสโตแกรมของจำนวนสุ่มทั้งหมดโดยแบ่งข้อมูลเป็น 10 ช่วง (รูปที่ ข.1 ด้านซ้าย) และแบ่งข้อมูลเป็น 20 ช่วง (รูปที่ ข.1 ด้านขวา)

ตารางที่ ข.7 ตัวอย่างเมทริกซ์พิเศษในโปรแกรม SCILAB

คำสั่ง	คำอธิบาย
`eye`	เมทริกซ์เอกลักษณ์ (identity matrix)
`ones`	เมทริกซ์ค่าหนึ่ง (one matrix)
`zeros`	เมทริกซ์ค่าศูนย์ (zero matrix)
`rand`	เมทริกซ์สุ่ม (random matrix)
`diag`	เมทริกซ์ทแยงมุม (diagonal matrix)
`tril`	เมทริกซ์สามเหลี่ยมด้านล่าง (lower triangular matrix)
`triu`	เมทริกซ์สามเหลี่ยมด้านบน (upper triangular matrix)
`toeplitz`	เมทริกซ์ Toeplitz (toeplitz matrix)

ข.6 เมทริกซ์พิเศษ

ในการประยุกต์ใช้งานเมทริกซ์ บางครั้งมีความจำเป็นต้องสร้างเมทริกซ์ที่มีค่าเฉพาะหรือมีรูปแบบที่เป็นมาตรฐาน เช่น ต้องการสร้างเมทริกซ์ที่มีค่าเป็นหนึ่งทั้งหมดขนาด `m×n` โดยที่ `m` และ `n` มีค่ามาก ถ้าสร้างเมทริกซ์นี้โดยการพิมพ์ค่าแต่ละค่าเข้าไปอาจจะทำให้เสียเวลามากและอาจเกิดข้อผิดพลาดได้ง่าย ดังนั้นโปรแกรม SCILAB จึงได้เตรียมฟังก์ชันพื้นฐานสำหรับสร้างเมทริกซ์พิเศษหลายรูปแบบขึ้นมาไว้ใช้งานตามตารางที่ ข.7 ตัวอย่างเช่น

```
-->A = eye(3,3)                    //สร้างเมทริกซ์เอกลักษณ์ที่มีขนาด 3×3
 A  =
    1.    0.    0.
    0.    1.    0.
    0.    0.    1.
-->diag([1 2 3])                   //ค่า 1, 2, และ 3 อยู่ที่เส้นทแยงมุมหลัก
 ans  =
    1.    0.    0.
    0.    2.    0.
    0.    0.    3.
```

ข.6.1 เมทริกซ์สุ่ม

เมทริกซ์สุ่ม (random matrix) เป็นเมทริกซ์ที่มีสมาชิกเป็นจำนวนสุ่ม การสร้างเมทริกซ์สุ่มในโปรแกรม SCILAB สามารถทำได้โดยใช้คำสั่ง `rand` ซึ่งมีลักษณะการเรียกใช้งานดังนี้

```
rand(m1, m2, [key])
```

โดยที่

- `m1, m2` คือเลขจำนวนเต็มบวกที่ใช้กำหนดขนาดของเมทริกซ์สุ่มที่จะสร้างขึ้นมา เช่น `rand(m1, m2)` หมายถึงให้สร้างเมทริกซ์สุ่มขนาด `m1` แนวนอน และ `m2` แนวตั้ง
- `key` เป็นตัวเลือกที่กำหนดลักษณะการแจกแจง (distribution) ของจำนวนสุ่มที่สร้าง กล่าวคือถ้า
  - `key = "uniform"` จำนวนสุ่มที่สร้างขึ้นมาจะมีลักษณะการแจกแจงเอกรูปมีค่าอยู่ระหว่าง `0` ถึง `1` (เป็นค่าโดยปริยาย)
  - `key = "normal"` จำนวนสุ่มที่สร้างขึ้นมาจะมีลักษณะการแจกแจงปรกติ (หรือแบบเกาส์เซียน) ที่มีค่าเฉลี่ยเท่ากับค่า `0` และมีค่าความแปรปรวน (varaince) เท่ากับค่า `1`

ตัวอย่างการใช้งานของคำสั่งนี้ เช่น

```
-->X = rand(2, 4, 'uniform')      //สร้างเมทริกซ์สุ่มขนาด 2×4
 X  =
    0.3095371    0.9706916    0.0204748    0.3490364
    0.6762972    0.5441797    0.8941365    0.1105365
-->W = rand(1, 100000, 'normal');
-->[mean(W), variance(W)]   //ค่าเฉลี่ยและค่าความแปรปรวนสอดคล้องกับลักษณะการแจกแจงปรกติ
 ans  =
    0.0048048    0.9988003
```

ข.7 การเขียนโปรแกรมด้วย SCILAB

ในส่วนนี้จะอธิบายการใช้งานคำสั่งวนซ้ำและคำสั่งทดสอบเงื่อนไข เพื่อให้ผู้อ่านสามารถเขียนโปรแกรมอย่างง่ายขึ้นมาใช้งานได้อย่างรวดเร็วและมีประสิทธิผล

ข.7.1 คำสั่งวนซ้ำ

บ่อยครั้งในการเขียนโปรแกรมมีความจำเป็นที่จะต้องคำนวณชุดคำสั่งบางอย่างซ้ำเป็นจำนวนหลายๆ รอบ ซึ่งในกรณีนี้การใช้คำสั่งวนซ้ำจึงมีความจำเป็นมาก โปรแกรม SCILAB ได้เตรียมคำสั่งสำหรับการวนซ้ำไว้อยู่สองรูปแบบคือ คำสั่ง `for` และคำสั่ง `while` ซึ่งมีหลักการใช้งานดังนี้

ข.7.1.1 คำสั่ง for

คำสั่ง for เหมาะสำหรับการใช้งานที่ต้องการให้โปรแกรมทำซ้ำชุดคำสั่งเดิมที่อยู่ภายในลูป (loop) เป็นจำนวนรอบตามที่กำหนดไว้ในนิพจน์ (expression) คำสั่ง for มีรูปแบบการใช้งาน ดังนี้

```
for variable = expression
    instruction_1;
        ⋮
    instruction_n;
end
```

กล่าวคือโปรแกรมจะทำซ้ำคำสั่ง (instruction) ทั้งหมดภายในลูปเป็นจำนวนรอบตามที่กำหนดโดยตัวแปรที่เป็นไปตามเงื่อนไขของนิพจน์ ตัวอย่างการใช้งานคำสั่ง for ตัวอย่างเช่น

```
-->L = 5;
-->x = [];
-->for i = 1:L
-->    x(i) = i;
-->end
```

ชุดคำสั่งนี้หมายความว่าเมื่อเริ่มต้นใช้งาน ตัวแปร L จะมีค่าเท่ากับ 5 และกำหนดให้ x เป็นเมทริกซ์ว่าง (empty matrix) จากนั้นก็ทำการวนซ้ำโดยใช้ตัวแปร i เป็นตัวนับจำนวนซ้ำ นั่นคือตัวแปร i จะเริ่มจากค่า 1 แล้วเพิ่มขึ้นทีละ +1 จนไปถึงค่า 5 โดยที่ค่าของตัวแปร i แต่ละค่าจะถูกบรรจุไว้ในสมาชิกลำดับที่ i ของเวกเตอร์ x ผลลัพธ์ของการประมวลผลชุดคำสั่งนี้คือ

```
-->x'
 ans  =
    1.    2.    3.    4.    5.
```

ข.7.1.2 คำสั่ง while

คำสั่ง while มีลักษณะการทำงานคล้ายกับคำสั่ง for เพียงแต่คำสั่ง while จะมีการทดสอบเงื่อนไขที่ผู้เขียนโปรแกรมกำหนดไว้ในนิพจน์ทุกๆ รอบของการวนซ้ำ กล่าวคือถ้าผลการทดสอบให้ค่าตรรกะเป็นค่า 1 (เป็นจริง) โปรแกรมก็จะทำซ้ำชุดคำสั่งภายในลูปนั้นต่ออีกหนึ่งรอบ แต่ถ้าผลการทดสอบให้ค่าตรรกะเป็นค่า 0 (เป็นเท็จ) โปรแกรมก็จะยกเลิกการทำงานชุดคำสั่งภายในลูปนั้นทันที คำสั่ง while มีรูปแบบการใช้งานดังนี้

```
while  expression
        instruction_1;
            ⋮
        instruction_n;
end
```

ตัวอย่างเช่น

```
-->L = 5;
-->x = [];
-->i = 1;
-->while i <= L
-->       x(i) = i;
-->       i = i + 1;
-->end
```

ชุดคำสั่งนี้ให้ผลลัพธ์เหมือนกับตัวอย่างของการใช้คำสั่ง `for` เพียงแต่การใช้คำสั่ง `while` จะต้องกำหนดค่าเริ่มต้นของตัวแปร `i` ก่อนที่จะนำค่า `i` ไปทำการเปรียบเทียบกับค่า `L` ตามเงื่อนไขที่กำหนด

ข.7.2 คำสั่งทดสอบเงื่อนไข

คำสั่งทดสอบเงื่อนไขมีความจำเป็นมากสำหรับการเขียนโปรแกรมคอมพิวเตอร์ที่ซับซ้อน คำสั่งทดสอบเงื่อนไขที่ใช้บ่อยคือ คำสั่ง `if` ซึ่งมีหลักการใช้งานดังนี้

```
if  expression  then
    instruction_1;
        ⋮
    instruction_n;
end
```

กล่าวคือถ้าผลการทดสอบเงื่อนไขในนิพจน์เป็นจริง โปรแกรม SCILAB ก็จะทำคำสั่งทั้งหมดที่อยู่ระหว่างคำว่า `then` และ `end` แต่ถ้าผลการทดสอบเป็นเท็จ โปรแกรม SCILAB จะไม่ทำคำสั่งทั้งหมดที่อยู่ระหว่างคำว่า `then` และ `end`

นอกจากนี้คำสั่ง `if` ยังสามารถนำไปใช้งานกับการตัดสินใจที่ซับซ้อนมากขึ้นได้โดยการใช้งานร่วมกับ `else` ซึ่งมีรูปแบบการใช้งานคือ

```
if expression then
    instructions_set1;
else
    instructions_set2;
end
```

นั่นคือถ้าผลการทดสอบเงื่อนไขในนิพจน์เป็นจริง โปรแกรม SCILAB จะทำคำสั่งทั้งหมดที่อยู่ระหว่างคำว่า `then` และ `else` แต่ถ้าผลการทดสอบเป็นเท็จ โปรแกรม SCILAB จะทำคำสั่งทั้งหมดที่อยู่ระหว่างคำว่า `else` และ `end`

ในการใช้งานที่มีการตัดสินใจที่ซับซ้อนมากยิ่งขึ้น ผู้ใช้ก็สามารถใช้งานคำสั่ง `if` ร่วมกับ `elseif` ได้ โดยมีรูปแบบการใช้งานดังนี้

```
If expression_1 then
    instructions_set1;
elseif expression_2 then
    instructions_set2;
else
    instructions_set3;
end
```

นั่นคือถ้าผลการทดสอบเงื่อนไขในนิพจน์ expression_1 เป็นจริง โปรแกรม SCILAB จะทำชุดคำสั่ง instructions_set1 แต่ถ้าผลการทดสอบเป็นเท็จ โปรแกรม SCILAB ก็จะทำการทดสอบเงื่อนไขในนิพจน์ expression_2 ต่อไปทันที โดยที่ถ้าผลการทดสอบในนิพจน์ expression_2 เป็นจริง โปรแกรม SCILAB ก็จะทำชุดคำสั่ง instructions_set2 แต่ถ้าผลการทดสอบเป็นเท็จ ก็จะทำชุดคำสั่ง instructions_set3

ข.7.3 การเขียนฟังก์ชันแบบอินไลน์

โปรแกรม SCILAB อนุญาตให้ผู้ใช้สามารถสร้างฟังก์ชันใหม่ๆ ขึ้นมาใช้งานร่วมกับโปรแกรม SCILAB ได้ ในที่นี้จะอธิบายเฉพาะการเขียนฟังก์ชันแบบอินไลน์ (in-line function) ซึ่งมีลักษณะการใช้งานดังนี้

```
function [เอาต์พุต] = function_name(อินพุต), ชุดคำสั่ง, endfunction
```

นั่นคือจะต้องเริ่มต้นด้วยคำว่า function และปิดท้ายด้วยคำว่า endfunction โดยที่ภายในฟังก์ชันจะมีคำสั่งแต่ละคำสั่งจะต้องคั้นด้วยเครื่องหมายคอมม่า ตัวอย่างเช่นถ้าต้องการสร้างฟังก์ชันแบบอินไลน์ที่ชื่อว่า MyMax เพื่อใช้ในการหาค่าสูงสุดของเลขจำนวนจริงสองจำนวน ก็สามารถทำได้ดังนี้

```
-->function [y] = MyMax(x1,x2), if x1>=x2 then y=x1, ...
-->else y=x2; end; endfunction;
-->y = MyMax(1,5)                                    //เรียกฟังก์ชัน MyMax ขึ้นมาใช้งาน
 y  =
    5.
```

นอกจากนี้ยังสามารถใช้คำสั่ง deff ในการสร้างฟังก์ชันแบบอินไลน์ได้เช่นกัน โดยมีรูปแบบการใช้งานคือ

```
deff('[เอาต์พุต]=function_name(อินพุต)', 'ชุดคำสั่ง')
```

ตัวอย่างเช่น

```
-->deff('y = MyMax(x1,x2)', 'if x1>x2 then y = x1; ...
-->else y = x2; end');
-->y = MyMax(1,5)
 y  =
    5.
```

ซึ่งให้ผลลัพธ์เท่ากัน

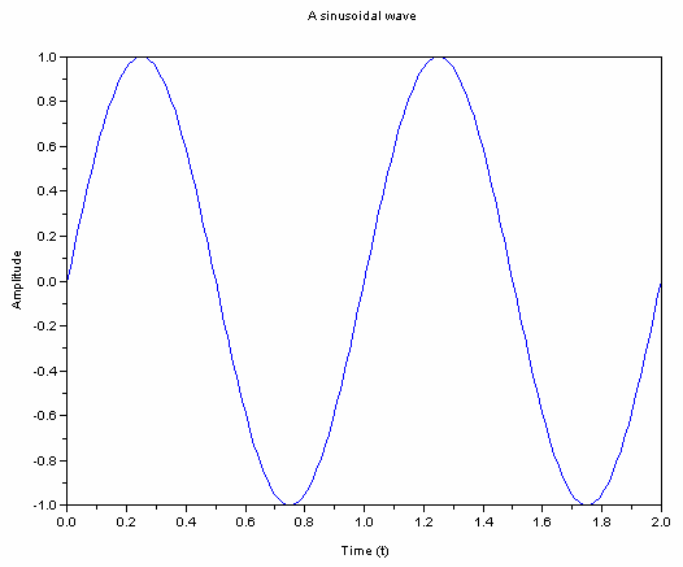
ข.8 การวาดกราฟสองมิติ

คำสั่งพื้นฐานสำหรับการวาดกราฟสองมิติบนระบบพิกัดฉาก x-y คือคำสั่ง plot ซึ่งมีรูปแบบการเรียกใช้งานดังนี้

```
plot(x,y)
```

เมื่อเวกเตอร์ x เป็นตัวแปรอิสระที่กำหนดค่าในเส้นแกน x และเวกเตอร์ y ซึ่งเป็นตัวแปรตามที่กำหนดค่าในเส้นแกน y (โดยที่เวกเตอร์ y จะต้องมีขนาดเท่ากันกับเวกเตอร์ x เสมอ) นอกจากนี้คำสั่ง plot ยังสามารถที่จะถูกเรียกใช้งานได้ในอีกรูปแบบหนึ่งคือ

```
plot(y)
```



รูปที่ ข.2 สัญญาณไซนูซอยด์ $y = \sin(2\pi ft)$

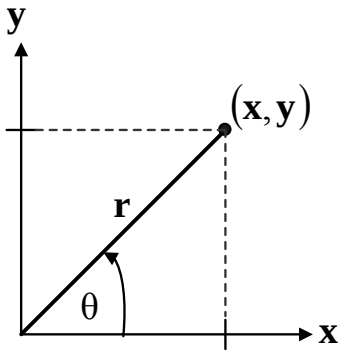
ซึ่งในกรณีนี้โปรแกรม SCILAB จะสมมติว่าพารามิเตอร์ `x` มีค่าเท่ากับค่า `1` ถึงจำนวนสมาชิกทั้งหมดของเวกเตอร์ `y` นั่นคือ `x = 1:length(y)` โดยอัตโนมัติ

ตัวอย่าง จงวาดกราฟของรูปสัญญาณไซนูซอยด์ (sinusoid waveform) ตามสมการ $y = \sin(2\pi ft)$ สำหรับเวลาที่ t = 0 ถึง 2 วินาที ถ้ากำหนดให้ความถี่ f = 1 เฮิตร์ซ (Hertz)

วิธีทำ จากโจทย์สามารถเขียนเป็นชุดคำสั่งของโปรแกรม SCILAB ได้ดังนี้

```
-->t = 0:0.01:2;
-->f = 1;
-->y = sin(2*%pi*f*t);
-->plot(t, y)
-->xtitle('A sinusoidal wave','Time (t)','Amplitude')
```

คำสั่งแรกเป็นการกำหนดให้ตัวแปร `t` ให้มีค่าอยู่ระหว่าง `0` ถึง `2` โดยที่สมาชิกแต่ละตัวที่อยู่ติดกันจะมีค่าห่างกันคงที่เท่ากับ `0.01` (ขนาดของตัวแปร `t` คือ `1×201`) จากนั้นก็กำหนดค่าความถี่ `f` ให้เท่ากับหนึ่ง แล้วก็หาค่าของสัญญาณ `y` โดยค่า `y` ที่หามาได้จะมีขนาดเท่ากับตัวแปร `t` จากนั้นก็สั่งให้วาดกราฟขึ้นมา ซึ่งผลลัพธ์ที่ได้จะเป็นกราฟตามรูปที่ ข.2 ส่วนคำสั่ง `xtitle` เป็นคำสั่งที่ใช้ในการกำหนดชื่อของกราฟ, ชื่อของเส้นแกน `x`, และชื่อของเส้นแกน `y`



รูปที่ ข.3 ความสัมพันธ์ระหว่างจุด (x, y) ในระบบพิกัดฉาก และจุด (r, θ) ในระบบพิกัดเชิงขั้ว

ข.8.1 กราฟเชิงขั้ว

โดยทั่วไปจุด (x, y) ที่แสดงถึงตำแหน่ง (location) บนรูปกราฟในระบบพิกัดฉากสามารถที่จะเปลี่ยนให้อยู่ในรูปของจุด (r, θ) ในระบบพิกัดเชิงขั้วได้ โดยที่ r คือขนาด และ θ คือมุมเรเดียน (เทียบกับแกน x ในทิศทวนเข็มนาฬิกา) โดยอาศัยกฎของตรีโกณมิติ ดังนั้นจากรูปที่ ข.3 จะได้ว่า

$$r = \sqrt{x^2 + y^2} \quad \text{และ} \quad \theta = \tan^{-1}\left(\frac{y}{x}\right)$$

ในทำนองเดียวกันจุดพิกัด (r, θ) ในระบบพิกัดเชิงขั้วก็สามารถที่จะแปลงกลับไปเป็นจุดพิกัด (x, y) ในระบบพิกัดฉากได้จากความสัมพันธ์ดังนี้

$$x = r\cos(\theta) \quad \text{และ} \quad y = r\sin(\theta)$$

การวาดกราฟเชิงขั้วในโปรแกรม SCILAB สามารถทำได้โดยการใช้คำสั่ง

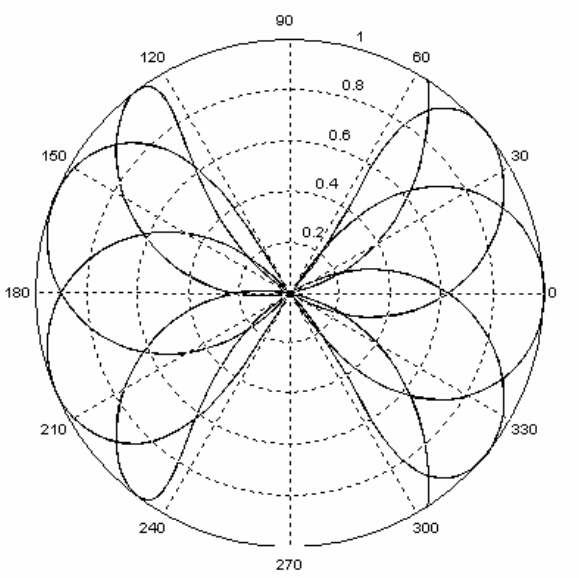
```
polarplot(theta, r)
```

เมื่อพารามิเตอร์ `theta` คือค่ามุม θ (มีหน่วยเป็นเรเดียน) และพารามิเตอร์ `r` คือค่าความยาวของรัศมี ตัวอย่างเช่น

```
-->t = 0:0.01:2*%pi;
-->polarplot(sin(7*t), cos(8*t))
```

ผลลัพธ์แสดงในรูปที่ ข.4

หมายเหตุ นอกจากนี้โปรแกรม SCILAB ยังได้เตรียมคำสั่งสำหรับการวาดกราฟสองมิติแบบอื่นๆ ไว้ใช้งานเฉพาะด้านมากมายดังแสดงในตารางที่ ข.8



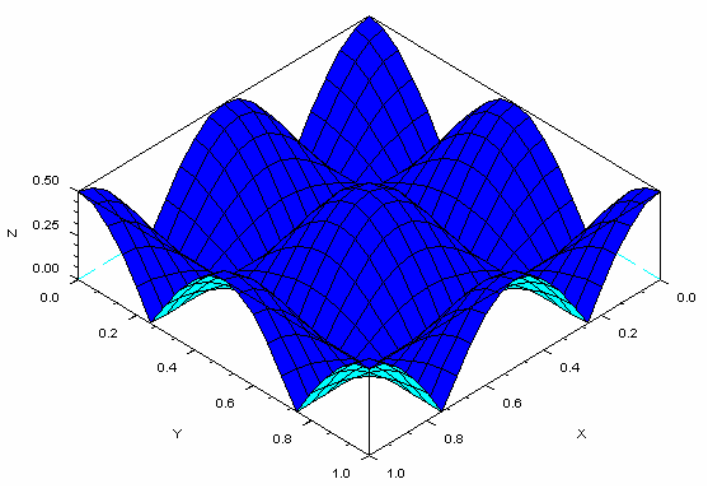
รูปที่ ข.4 ตัวอย่างรูปกราฟแสดงผลลัพธ์จากการใช้คำสั่ง `polarplot`

ตารางที่ ข.8 ตัวอย่างคำสั่งในการวาดกราฟสองมิติสำหรับการใช้งานเฉพาะด้าน

คำสั่ง	คำอธิบาย
`contour2d`	วาดกราฟคอนทัวร์ (contour surface) จากรูปกราฟสองมิติ
`champ`	วาดกราฟสนามเวกเตอร์แบบสองมิติ (2-D vector field)
`fchamp`	วาดกราฟสนามเวกเตอร์แบบสองมิติ ที่กำหนดโดยสมการอนุพันธ์อันดับหนึ่ง (first-order ordinary differential equation)
`bode`	วาดกราฟของโบดไดอะแกรม (Bode diagram) ทั้งกราฟแสดงขนาด (magnitude plot) และกราฟแสดงมุม (phase plot) ซึ่งมีประโยชน์มากทางด้านวิศวกรรม
`gainplot`	วาดกราฟแสดงขนาดของโบดไดอะแกรม
`nyquist`	วาดกราฟไนควิสซ์ (Nyquist plot)
`evans`	วาดกราฟอีแวนรูทโลคัส (Evans root locus)
`plzr`	วาดกราฟโพล-ซีโร่ (pole-zero plot)

ข.9 การวาดกราฟสามมิติ

สมการคณิตศาสตร์แบบสามตัวแปรใดๆ สามารถที่จะแสดงให้อยู่ในรูปของกราฟสามมิติได้ เพื่อใช้แสดงความสัมพันธ์ของตัวแปรทั้งสาม การใช้งานคำสั่งวาดกราฟสามมิตินั้นไม่ยากเพียงแต่ต้องเข้าใจถึงรูปแบบของข้อมูลที่จะป้อนให้กับคำสั่งเหล่านี้ การวาดกราฟสามมิติจะใช้ข้อมูลทั้งหมดสามชุดสำหรับเส้นแกน `x`, เส้นแกน `y`, และเส้นแกน `z` ที่อยู่ในพิกัดคาร์ทีเซียน (Cartesian coordinate) `x-y-z` โดยเวกเตอร์ `x` จะเป็นตัวกำหนดค่าในเส้นแกน `x`, เวกเตอร์ `y` จะเป็นตัวกำหนดค่าในเส้นแกน `y`, และตัวแปรตามที่มีค่าเปลี่ยนแปลง



รูปที่ ข.5 ตัวอย่างรูปกราฟแสดงผลลัพธ์จากการใช้คำสั่ง `plot3d`

ไปตามค่า `x` และ `y` ซึ่งก็คือขนาดของค่าบนเส้นแกน `z` นั่นเอง ดังนั้นตัวแปรตาม `z` นี้จะต้องมีจำนวนเท่ากับผลคูณของจำนวนข้อมูลในเวกเตอร์ `x` กับจำนวนข้อมูลในเวกเตอร์ `y`

คำสั่งพื้นฐานสำหรับการวาดกราฟแบบสามมิติในโปรแกรม SCILAB มีรูปแบบดังนี้

```
plot3d(x, y, z)
```

โดยที่พารามิเตอร์ `x` และ `y` คือเวกเตอร์ที่มีขนาดเท่ากัน และพารามิเตอร์ `z` คือตัวแปรตามที่ขึ้นกับค่าของ `x` และ `y` ซึ่งจะมีจำนวนเท่ากับผลคูณของจำนวนข้อมูลในเวกเตอร์ `x` กับจำนวนข้อมูลในเวกเตอร์ `y`

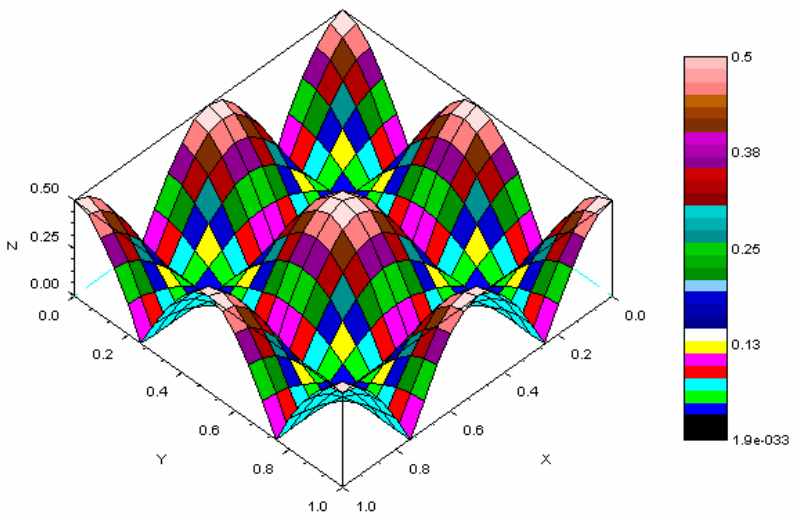
ตัวอย่าง กำหนดให้ตัวแปร `x` และ `y` มีค่าระหว่าง `0` ถึง `1` จงวาดกราฟสามมิติจากสมการ

$$z = \left|0.5\cos\left(2x\pi\right)\cos\left(2y\pi\right)\right|$$

วิธีทำ จากโจทย์ สามารถวาดกราฟสามมิติได้โดยใช้ชุดคำสั่งของโปรแกรม SCILAB ดังนี้

```
-->x = linspace(0, 1, 21);
-->y = linspace(0, 1, 21);
-->z = abs(0.5 * cos(2*%pi*x)' * cos(2*%pi*y));
-->plot3d(x, y, z);
```

ผลลัพธ์ที่ได้จากชุดคำสั่งเหล่านี้แสดงในรูปที่ ข.5



รูปที่ ข.6 ตัวอย่างรูปกราฟแสดงผลลัพธ์จากการใช้คำสั่ง `colorbar` ร่วมกับ `plot3d1`

ถ้าต้องการให้มีการไล่โทนสีตามขนาดของค่าในแกน z ก็สามารถทำได้โดยการใช้คำสั่ง `plot3d1` และถ้าต้องการทราบความสัมพันธ์ระหว่างสีกับขนาดของค่าในแกน z ก็สามารถทำได้โดยใช้คำสั่ง `colorbar` ซึ่งมีรูปแบบการใช้งานคือ

```
colorbar(umin, umax, [colminmax])
```

โดยที่พารามิเตอร์

- `umin` เป็นเลขจำนวนจริงของค่าต่ำสุดของขนาดของค่าในแกน z
- `umax` เป็นเลขจำนวนจริงของค่าสูงสุดของขนาดของค่าในแกน z
- `colminmax` เป็นตัวเลือกที่มีรูปแบบการใช้งานคือ `colminmax = [1 nb_colors]` เป็นเวกเตอร์ขนาด 1×2 โดยที่ `nb_colors` คือจำนวนสีที่จะใช้ในรูปกราฟ

ตัวอย่างการใช้งานคำสั่งนี้ เช่น (ต่อเนื่องจากรูปที่ ข.5)

```
-->clf;  zmin = min(z);
-->zmax = max(z);
-->colorbar(zmin, zmax, [1 30]);
-->plot3d1(x, y, z);
```

ผลลัพธ์ที่ได้แสดงในรูปที่ ข.6 ซึ่งจะมีแถบสีแสดงความสัมพันธ์ระหว่างสีต่างๆ กับขนาดของค่าในแกน z

หมายเหตุ จากที่กล่าวมาทั้งหมดนี้เป็นเพียงพื้นฐานการใช้งานโปรแกรม SCILAB สำหรับผู้ที่สนใจวิธีการใช้งานโปรแกรม SCILAB เพิ่มเติมสามารถศึกษาได้จาก [5] หรือ http://home.npru.ac.th/piya/webscilab