يمكن استخدام معاملات المقارنة للمقارنة بين مصفوفتين لها نفس الحجم، أو للمقارنة بين مصفوفة وعدد مفرد وتتم هذه الحالة مقارنة كل عنصر من المصفوفة مع العدد المفرد، وتكون المصفوفة الناتجة بنفس حجم المصفوفة التي تمت مقارنتها كما يبينه المثال التالي:

مثال (١):

```
>> a = 1; b = 5;
>> x = a > b
x =
    0
>> A = 1: 9, B = 9 - A
A =
    1   2   3   4   5   6   7   8   9
B =
    8   7   6   5   4   3   2   1   0
>> tf = A > 4
tf =
    0   0   0   0   1   1   1   1   1
```

لقد أوجدنا العناصر من A التي هي أكبر من 4، وتظهر الاصفار في المصفوفة الناتجة في مواقع العناصر عندما $A \leq 4$، بينما يظهر الرقم 1 عندما $A > 4$.

```
>> tf  = (A == B)
tf =
    0   0   0   0   0   0   0   0   0
```

لقد تم هنا إيجاد عناصر A التي تساوي العناصر في المصفوفة B.

ملاحظة:

لاحظ بان الإشارتين (=) و (==) تعنيان شيئاً مختلفاً، حيث يقوم (==) بمقارنة متغيرين وتعيد العدد واحد إذا كانا متساويين وصفراً إذا لم يكونا متساويين، بينما تستخدم (=) لإسناد إخراج العملية إلى متغير.

مثال (١): لتوليد مصفوفة أحادية منطقية عناصرها واحدات (في حالة اكبر من thr) واصفاراً (في حالة اصغر من أو تساوي thr).

```
>> inddent = [10   17   22   0   7   3   2];
>> thr = 7;
>> y = (indent > thr)
y =
    1   1   1  0   0    0   0
```

مثال (٢): لتوليد مصفوفة أحادية عناصرها نفس العناصر (في حالة اكبر من thr) واصفاراً (في حالة اصغر من أو تساوي thr).

```
>> z = inddent .* (inddent > thr)
z =
   10   17    22   0   0   0   0
```

المعاملات المنطقية (العوامل المنطقية) :Logical Operators

توفر المعاملات المنطقية طريقة لدمج أو نفي تعابير المقارنة، ويظهر الجدول التالي المعاملات المنطقية الموجودة في لغة MATLAB:

المعامل المنطقي	الوصف
&	(و) and
|	(أو) or
~	(نفي) not

وسنقدم لك فيما يلي بعض الأمثلة على استخدام المعاملات المنطقية:

```
>> a = 1;
>> b = 5;
>> x = a ~= b
x =
    1
>> b = (1 == 1) & (2 ~= 3)
```

```
b =
    1
>> b = (1==1) | (2 ~= 3)
b =
    1
>> b = (1==1) &  not ((2 ~= 3))
b =
    0
```

```
>> A = 1: 9; B = 9 - A;
>> tf = A > 4
tf =
    0    0    0    0    1    1    1    1    1
```

حيث قام بإيجاد عناصر A التي قيمها اكبر من 4

```
>> tf = ~ (A > 4)
tf =
    1    1    1    1    0    0    0    0    0
```

لقد قام البرنامج بقلب النتيجة السابقة، وتعني استبدال مواقع الاصفار والواحدات.

```
>> tf = (A > 2) & (A < 6)
tf =
    0    0    1    1    1    0    0    0    0
```

حيث تعيد هذه العبارة العدد واحد عندما يكون العنصر من A اكبر من 2 واقل من 6.

أسبقية المعامل

يقوم برنامج MATLAB بإيجاد قيمة تعبير مستنداً إلى مجموعة من القواعد الناظمة لأسبقية المعامل، وتحسب المعاملات ذات الأسبقية العليا قبل المعاملات ذات الأسبقية الدنيا، وتقيم المعاملات ذات

الأسبقية المتساوية من اليسار إلى اليمين. ويشرح الجدول التالي قواعد أسبقية المعامل التي يعتدها برامج MATLAB.

المعامل	مستوى الأسبقية
الأقواس ( )	الأعلى
المدور(')، القوة (^، .^)	
إشارة النفي (~)	
الضرب (*، .*)، القسمة (/، ./)	
الجمع (+)، والطرح (-)	
معامل النقطتين المتعامدتين (:)	
أصغر من (<)، واصغر أو يساوي(=<)، اكبر من (<)، اكبر من أو يساوي(=<)، المساواة (==)،عدم المساواة (=~)	
الجمع المنطقي(&) and	
المعامل المنطقي (|) or	الأدنى

الصيغة if-else-end

قد نحتاج إلى حساب مجموعة من أوامر استناداً إلى إخراج ناتج عن اختبار شرطي. وتنفذ هذه التعليمة في لغة MATLAB عبر استخدام الصيغة if-else-end وكما يلي:

```
if expression
   (commands)
end
```

وستنفذ الأوامر (commands) الواقعة بين العبارتين if وend إذا كانت قيمة التعبير (expression) تكون true. واليك المثال التالي:

```
>> x = 10;
>> if  x == 10
      disp ('ok')
   end;
```

وإذا كان لدنيا خياران، فتصبح الصيغة if-else-end كما يلي:

```
if expression
  (commands evaluated if True)
else
  (commands evaluated if False)
end
```

حيث ستنفذ المجموعة الأولى من الأوامر في حال امتلك التعبير expression القيمة true، بينما تنفذ المجموعة الثانية إذا امتلك التعبير expression القيمة false.

وإذا كانت هناك عدة حالات، فستأخذ التعبير if-else-end الشكل التالي:

```
if expression1
  (commands evaluated if expression1 is true)
elseif expression2
  (commands evaluated if expression2 is true)
elseif expression3
  (commands evaluated if expression3 is true)
elseif expression4
  (commands evaluated if expression4 is true)
.
.
.
```

```
else
  (commands evaluated if no other expression is true)
end
```

واليك الأمثلة التالية:

مثال (١):

```
>> x = 10;
>> if  x == 10
  msgbox ('ok', 'result');
```

مثال (٢):

```
>> if  x ==  10
      msgbox ('ok', 'result');
    else
      msgbox ('no', 'result');
    end;
```

مثال (٣):

```
>> x = 11;
>> if  x == 1
      disp ('1');
    elseif  x == 2
      disp ('2');
    else
      disp ('3');
   end;
```

الإخراج

3

الصيغة switch-case

عندما يتوجب علينا تنفيذ أوامر اعتماداً على استخدام متكرر لاختيار كمي لوسط ما، عندها من السهل استخدام الصيغة switch-case التي لها الصيغة العامة التالية:

```
switch expression
  case test-expression1
     (commands1)
  case test-expression2
     (commands2)
   otherwise
       (commands3)
end
```

يجب أن يكون expression هنا أما عدداً مفرداً أو سلسلة رمزية. يقارن التعبير expression الموجود في الصيغة السابقة بالتعبير test-expression1 الموجود في عبارة case الأولى. وإذا تساوى التعبيران، سيتم تنفيذ الأوامر (commands1) وتخطي التعليمات الواقعة بعدها حتى العبارة end. أما إذا لم يتحقق الشرط الأول، فسيختبر الشرط الثاني، حيث سيقارن expression في المثال السابق مع العبارات test-exoression2 الموجودة في عبارة case الثانية. وإذا تساوى التعبيران، سيتم تنفيذ (commands2) وتهمل بقية العبارات حتى عبارة end. إذا لم تحقق أي عبارة case المساواة مع التعبير expression، عندها ستنفذ الأوامر (commands3) التي تلي العبارة otherwise.

لاحظ من الشرح الذي أوردناه عن صيغة switch-case بأنه سيتم تنفيذ أحدى مجموعات الأوامر المكونة للصيغة switch-case واليك الأمثلة التالية:

مثال (١):

```
 x = 1;
switch x
   case {1, 2, 3, 4, 5}
      disp ('1..5');
   case {9, 10}
      disp ('9..10');
   otherwise
```

```
    disp ('this is impossible');
end;
```

الإخراج 1..5

مثال (٢):

```
clc;
clear;
n = 3;
switch n
  case {0}
    m = n + 3;
  case {2}
    m = 'ali';
 case {3}
    m = magic (n);
  otherwise
    disp ('error');
end;
disp (m);
```

الإخراج

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

مثال (٣):

```
x = 2.7;
units = 'm';
switch units
  case {'inch', 'in'}
```

```
    y = x * 2.54;
  case {'meter', 'm'}
    y = x / 100;
  case {'feed', 'ft'}
    y = x * 2.54 / 12;
  case {'millimeter', 'mm'}
    y = x * 10;
  case {'centimeter', 'cm'}
    y = x;
 otherwise
   disp (['unknown units:'    units]);
end;
```

الإخراج

```
y = 0.027
```

جمل الدوران والتكرار

توفر لغة MATLAB مجموعة من جمل الدوران والتكرار وهي:

جملة for

تقوم حلقات for بإعادة تنفيذ مجموعة من الأوامر لعدد معين من المرات وبخطوة معينة، وتعطى الصيغة العامة لحلقة for كما يلي:

```
for  i = x1: x3: x2
  (commands)
end;
```

حيث يعاد تنفيذ الأوامر (commands) الواقعة بين عبارتي for و end من القيمة الابتدائية x1 إلى القيمة النهائية x2 وبزيادة مقدارها x3. كما في المثال التالي:

مثال:

```
>> for n = 1: 10
       x (n) = sin (n * pi / 10);
   end;
>> x
x =
  Columns 1 through 7
```