المحاضرة الثالثة: لغة البرمجة MATLAB

(The MATLAB programming language)

~~~~~~

1.البحث عن المصفوفة الجزئية.
2.توابع التعامل مع المصفوفة.
3.حجم المصفوفة.
4.المصفوفات متعددة الابعاد.
5.مصفوفة الخلايا.

أعداد: - م.م. محمد وليد عبد الرضا

جامعة البصرة

كلية التربية للعلوم الصرفة

قسم الرياضيات
~~~~~~

البحث عن مصفوفة جزئية

من المفيد في بعض الأحيان إن تعرف موقع أو دليل العناصر التي تحقق شرطا معينا، والموجودة ضمن مصفوفة معينة. يقوم برنامج MATLAB بتحقيق هذه الغاية عبر الايعاز find، والذي يعيد لك دليل أو موقع العنصر الذي تكون نتيجة تحقيقه لشرط ما true، واليك المثال التالي:

```
>> x = -3: 3
x =
    -3   -2   -1   0   1   2   3
>> k = find (abs (x) > 1)
 k =              (الموقع)
    1   2   6   7
>> y = x (k)
y =
    -3   -2   2   3
>> y = x (abs (x) > 1)
y =
    -3   -2   2   3
```

ويستطيع الايعاز find أن يعمل في المصفوفات الثنائية البعد أيضا (عمود بعد عمود)، فمثلا:

```
>> A = [1   2   3;   4   5   6;   7   8   9]
A =
   1   2   3
   4   5   6
   7   8   9
>> [i, j] = find (A > 6)
i =
   3
```

```
   3
   3
j =
   1
   2
   3
```

ملاحظة: الايعاز diag يوجد عناصر القطر الرئيسي للمصفوفة.

$$A = \begin{bmatrix} \mathbf{7} & 8 & 9 & 9 \\ 7 & \mathbf{8} & 9 & 9 \\ 4 & 5 & \mathbf{8} & 6 \\ 7 & 8 & 9 & \mathbf{9} \end{bmatrix}$$

```
>> diag (A)
ans =
     7
     8
     8
     9
```

ملاحظة:

يوفر برنامج MATLAB الدالتين max ، min الذين يوجدان اكبر واصغر عنصر في المصفوفة ومواقعهما.

في حالة المصفوفة الأحادية:

```
>> v = rand (1, 6)
v =
   0.3046   0.1897   0.1934   0.6822   0.3028   0.5417
>> max (v)
ans =
     0.6822
```

```
>> [mx, i] = max (v)
mx =
      0.6822
i =
    4
>> min (v)
ans =
       0.1897
>> [mn, j] = min (v)
mn =
      0.1897
j =
    2
```

في حالة كون المصفوفة ثنائية البعد:

```
>> A = rand (4, 6)
A =
     0.1509    0.8537    0.8216    0.3420    0.7271    0.3704
     0.6979    0.5936    0.6449    0.2897    0.3093    0.7027
     0.3784    0.4966    0.8180    0.3412    0.8385    0.5466
     0.8600    0.8998    0.6602    0.5341    0.5681    0.4449
>> [mx, r] = max (A)
mx =
      0.8600    0.8998    0.8216    0.5341    0.8385    0.7027
r =
    4    4    1    4    3    2
```

<u>ملاحظة:</u>

```
>> max (A');                (اكبر عنصر لكل سطر)
```

```
>> [mn, r] = min (A)
mn =
    0.1509    0.4966    0.6449    0.2897    0.3093    0.3704
r =
    1    3    2    2    2    1
```

ملاحظة:

```
>> min (A');          (اصغر عنصر لكل سطر)
```

<<ملاحظة: اكبر عنصر في مصفوفة ثنائية البعد.

```
mmx = max (mx)
mmx =
    0.8998
>> [mmx, i] = max (A (:))
mmx =
    0.8998
i =
    8
```

ملاحظة: توجد طريقة أخرى:

```
>> z = max (max (A));
>> z = min (min (A));
```

ملاحظة: نفس الشيء لحساب المجموع sum.

```
>> z = sum (sum (A));
```

توابع التعامل مع المصفوفة

يزودك برنامج MATLAB، بالإضافة إلى عنونة المصفوفات والمقدرة على التعامل مع المصفوفات التي شرحناها سابقاً، بعمليات التعامل مع المصفوفات، وهي سهلة التطبيق مثل:

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

```
    4   5   6
    7   8   9
>> flipud (A)                قلب المصفوفة باتجاه up-down
ans =
      7   8   9
      4   5   6
      1   2   3
>> fliplr (A)                قلب المصفوفة باتجاه left-right
ans =
      3   2   1
      6   5   4
      9   8   7
>> triu (A)                  استخلاص الجزء المثلية العليا (upper)
ans =
      1   2   3
      0   5   6
      0   0   9
>> tril (A)                  استخلاص الجزء المثلية السفلى (lower)
ans =
      1   0   0
      4   5   0
      7   8   9
>> g = det (A);              حساب محدد المصفوفة (قيمة)
>> h = inv (A);              حساب معكوس المصفوفة (مصفوفة)
>> i = eig (A);              حساب القيم الذاتية للمصفوفة
>> j = eye (3)               حساب مصفوفة الوحدة
j =
```

```
    1    0    0
    0    1    0
    0    0    1
>> trace (A);          حساب مجموع عناصر القطر الرئيسي
```

حجم المصفوفة

إذا أردت أن تعرف حجم أو بعد مصفوفة أحادية أو ثنائية أو ثلاثية البعد غير معروفين وكنت بحاجة لحجمها لإجراء بعض العمليات الرياضية، فان برنامج MATLAB يمكنك من خلال الايعاز length و size و numel واليك الأمثلة التالية:

```
>> A = [1    2    3    4; 5    6    7    8]
A =
    1    2    3    4
    5    6    7    8
>> S = size (A)
S =
    2    4
```

يعبر العنصر الأول عن عدد الأسطر (2) بينما يعطي العنصر الثاني عدد الأعمدة (4).

```
>> [r, c] = size (A)
r =
    2
c =
    4
>> r = size (A, 1)
r =
    2
>> c = size (A, 2)
c =
    4
```

يعيد الايعاز numel العدد الكلي لعناصر مصفوفة فمثلاً:

```
>> numel (A)
ans =
      8
```

بينما يعيد الايعاز length عدد العناصر الموجودة ضمن البعد الأطول للمصفوفة، كما يلي:

```
>> length (A)
ans =
      4
>> B = -3: 3
B =
    -3    -2    -1    0    1    2    3
>> length (B)
ans =
      7
>> min (size (A))                مصفوفة ثنائية A
ans =
      2
```

ملاحظة: طريقة توليد مصفوفة بالدمج.

```
>> x = [1    2; 3    4];
>> y = [x    x .^ 2; x .^ 3    x .^ 4];
```

المصفوفات متعددة الإبعاد

لقد شرحنا في الفصل السابق المصفوفات أحادية وثنائية الأبعاد والعمليات التي تجري عليها. يدعم برنامج MATLAB المصفوفات متعددة الأبعاد (أي n-D arrays) وذلك نفس الايعازات وتقنيات العنونة المطبقة على المصفوفات أحادية وثنائية البعد. وبشكل عام، يرقم البعد الثالث عبر صفحات (pages)، ولذلك تمتلك المصفوفات ثلاثية البعد اسطرا وأعمدة وصفحات، حيث تتألف كل صفحة من مصفوفة ثنائية البعد ذات اسطر وأعمدة، ويجب أن تمتلك كل صفحة عددا متساويا من الأسطر والأعمدة والعكس بالعكس في كل صفحة.

ليس هناك حد لعدد الأبعاد في المصفوفات، ولكننا سنستخدم مصفوفات ثلاثية الأبعاد في هذا الفصل بسبب سهولة تخيلها وإظهارها.

تركيب المصفوفة

يمكن إنشاء المصفوفة المتعددة الأبعاد بطرق مختلفة، واليك بعضها:

```
>> A = zeros (4, 3, 2)
A (:, :, 1) =
     0     0     0
     0     0     0
     0     0     0
     0     0     0
A (:, :, 2) =
     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

تتألف هذه المصفوفة الصفرية من أربعة اسطر وثلاثة أعمدة وصفحتين، ولقد ظهرت الصفحة الأولى ثم الصفحة الثانية.

مثال:

```
>> B (:, :, 1) = zeros (2, 3);
```

```
>> B (:, :, 2) = ones (2, 3);
>> B (:, :, 3) = 4;
>> B
B (:, :, 1) =
        0    0    0
        0    0    0
B (:, :, 2) =
        1    1    1
        1    1    1
B (:, :, 3) =
        4    4    4
        4    4    4
```

يمكن استخدام الايعاز reshape لتحويل المصفوفة من مصفوفة ثنائية الأبعاد إلى مصفوفة ثلاثية الأبعاد وكالاتي:

```
>> C = [B (:, :, 1), B (:, :, 2), B (:, :, 3)]
C =
    0    0    0    1    1    1    4    4    4
    0    0    0    1    1    1    4    4    4
>> reshape (C, 2, 3, 3)
ans (:, :, 1) =
        0    0    0
        0    0    0
ans (:, :, 2) =
        1    1    1
        1    1    1
ans (:, :, 3) =
        4    4    4
```

```
     4   4   4
```

حجم المصفوفة

الايعاز size يعيد بعد المصفوفة وفق كل أبعادها كما شرحنا سابقاً.

```
>> [r, c, p] = size (C)
r =
    2
c =
    3
p =
    3
```

وإذا لم نعرف عدد إبعاد المصفوفة أو كانت ابعادها متغيرة، عندما نستطيع استخدام الايعاز ndims كما يلي:

```
>> ndims (C)
ans =
      3
>> numel (C)                إيجاد عدد عناصر المصفوفة
ans =
      18
>> length (size (C))        إيجاد طول اكبر بعد بالمصفوفة
ans =
      3
```

مصفوفة الخلايا Cell Arrays

تعتبر مصفوفات الخلايا مصفوفات في لغة MATLAB تكون عناصرها عبارة عن خلايا، وتتضمن كل خلية نوعاً من البيانات قد تكون مصفوفات عددية أو رمزية أو كائنات بسيطة أو مصفوفات خلايا أخرى. فمثلاً قد تحوي خلية من مصفوفة الخلية مصفوفة عددية وتحوي الخلية الأخرى مصفوفة رمزية،

بينما تحوي الثالثة على أعداد عقدية (يسمح باستخدام مصفوفات بأنواع بيانية مختلفة (غير متجانسة)) كما ويمكن إنشاء مصفوفات الخلايا بأي بعد كان كما هي الحال مع المصفوفات العددية، ولكن معظم مصفوفات الخلايا تكون عبارة عن مصفوفات أحادية البعد.

تنشأ مصفوفات الخلايا عبر استخدام تعابير الإسناد أو عبر إعادة تقسيم المصفوفة بالإيعاز cell، ثم نقوم بإسناد البيانات إلى الخلايا.

هناك طريقتان مختلفتان للوصول إلى الخلايا. وإذا أردت استخدام رموز مصفوفة قياسية للدلالة على المصفوفة، يجب عليك أن تحيط الخلية بأقواس مجموعة { }، إذ إن برنامج MATLAB يستخدم هذه الأقواس لتعريف مصفوفات الخلايا، واليك الأمثلة التالية:

```
>> A (1, 1) = {[1   2   3; 4   5   6; 7   8   9]};
>> A (1, 2) = {2 + 3i};
>> A (2, 1) = {'Ali Ahmed'};
>> A (2, 2) = {12: -2: 0};
>> A
A =
    [3×3 double]       [2.0000 + 3.0000i]
    'Ali Ahmed'        [1×7 double]
```

لاحظ إن برنامج MATLAB يظهر المصفوفة كمصفوفة خلية بعداها 2×2 ولكن ذلك لا يظهر محتويات الخلية، وإنما يظهر البرنامج محتويات الخلية بشكل أساسي إذا لم تأخذ هذه المحتويات حجماً كبيراً، كما ويوصف محتويات الخلية إذا أخذت حجماً معقولاً. إن وجود أقواس مجموعات على الجانب الأيمن من المساواة يدل على إن المشار إليه هو خلية وليس قيما عددية وهذا ما يسمى *بفهرسة الخلية (cell indexing)*، وسينشئ التعابير التالية مصفوفة الخلية نفسها.

ملاحظة:

يخبر كلا التعبيرين A (i, j) = {x} و A {i, j} = x برنامج MATLAB بأن يضع المتغير x في العنصر (i, j) من مصفوفة الخلية A.

يدعى التعبير A (i, j) *بفهرسة الخلية (cell indexing)* ، بينما يدعى التعبير A {i, j} *بعنونة المحتوى (content addressing)* أي تدل أقواس المجموعة { } على محتوى الخلية، بينما تعرف الأقواس العادية ( ) الخلايا دون النظر إلى محتواها.

مثال:

```
>> y = {1, 'hello', 1 > 5}
y =
    [1]    'hello'    [0]
>> y {1}
ans =
       1
>> y {2}
ans =
      hello
>> y {3}
ans =
       0
```

مثال:

```
>> ce = {[1    2    3; 5    6    7], 'yes', 3 > 2};
>> ce {1}(2, 2)
ans =
       6
```

مثال:

```
>> x = rand (3, 3);
>> y = rand (3, 3);
>> z = rand (3, 3);
>> w {1} = x;
>> w {2} = y;
```

```
>> w {3} = z;
>> w
ans =
    [3×3 double]    [3×3 double]    [3×3 double]
```

مثال:

```
>> x {1} = rand (3, 3);
>> x {2} = rand (3, 3);
>> x {3} = rand (3, 3);
      .
      .
      .
      .
      .
>> x {9} = rand (3, 3);
>> x {1}
ans =
    0.8462    0.6721    0.6813
    0.5252    0.8381    0.3795
    0.2026    0.0196    0.8318
```

>> x {1} (2, 2) العنصر الموجود في السطر الثاني والعمود الثاني في مصفوفة (الخلية) الأولى

```
ans =
    0.8381
```

مثال: برنامج لجمع المصفوفات التسعة في المثال السابق في مصفوفة واحدة.

```
>> L = length (x);
>> sum1 = 0;
>> for i = 1: L
```

```
    b = x {i};
    sum1 = sum1 + b;
  end;
```

يجبر الإيعاز celldisp برنامج MATLAB على إظهار محتوى الخلايا بالنموذج العادي، واليك المثال التالي الذي يوضح ذلك:

```
>> celldisp (A)
A (1, 1) =
     1   2   3
     4   5   6
     7   8   9
  A (2, 1) =
     Ali Ahmed
  A (1, 2) =
     2.0000 + 3.0000i
  A (2, 2) =
     12   10   8   6   4   2   0
```

كما يُظهر البرنامج محتوى الخلية المفردة عبر طلب محتوى الخلية باستخدام عنونة المحتوى، وهذا يتم بشكل مختلف عن فهرسة الخلية التي تعّرف الخلية فقط دون الدخول إلى محتوى الخلية، فمثلا:

```
>> A {2, 2}
ans =
     12   10   8   6   4   2   0
>> A (2, 2)
ans =
    [1×7 double]
>> A (1, :)
ans =
    [3×3 double]    [2.0000 + 3.0000i]
```

لاحظ بأن البرنامج استخدم لجميع الخلايا السابقة الاسم ans، وذلك لان خلايا البيانات المخزونة ليس لها اسم محدد.

لقد استخدمنا سابقاً الأقواس المربعة لإنشاء المصفوفات العددية، وتعمل أقواس المجموعة نفس العمل بالنسبة للخلايا، وتفصل الأعمدة بفواصل بينما تفصل الأسطر بفواصل منقوطة. واليك المثال التالي:

```
>> B = {[1     2], 'John Smith';  2 + 3i, 5}
B =
    [1×2 double]          'John Smith'
    [2.0000 + 3.0000i]    [5]
```

من المألوف عند استخدام المصفوفات العددية أن تُملأ المصفوفة بعناصر صفرية ثم تُملأ من جديد بالبيانات اللازمة، ويمكن استخدام نفس المنهج في مصفوفات الخلايا، حيث ينشأ الايعاز cell مصفوفة خلية ويملؤها بمصفوفات عددية فارغة [ ] ولنأخذ المثال التالي:

```
>> C = cell (2, 3)
C =
    [ ]   [ ]   [ ]
    [ ]   [ ]   [ ]
```

ما إن يتم تعريف مصفوفة الخلية فأنة يمكن تعميم الخلايا عن طريق عنونة المحتوى وفهرسة الخلايا، كما يبينه المثال التالي:

```
>> C (1, 1) = 'The does n't work'
    Error
```

لقد استخدمنا هنا في الجانب الأيسر دليل الخلية وبالتالي، يجب أن يكون الطرف الأيمن خلية وهذا ما سبب ظهور الخطأ، وليس كوننا لم نُخط محتوياتها بأقواس مجموعة.

```
>> C (1, 1) = {'The does n't work'}
C =
    'The does n't work'   [ ]   [ ]
    [ ]                   [ ]   [ ]
>> C (2, 3) = {'This works too'}
C =
```

'This does work' [ ] [ ]

[ ] [ ] 'This works too'

وبسبب وجود أقواس المجموعة في الجانب الأيسر من العبارة الأخيرة، فان برنامج MATLAB سيضع الخيط الرمزي في الخلية المعنونة. ويوجد هنا مرة أخرى عنونة محتوى، بينما تعتبر العبارة الأصلية مثالاً عن فهرسة المصفوفة.

التعامل مع مصفوفة الخلية

يمكن أن نستخدم الأقواس المربعة أيضاً لضم مصفوفات الخلايا ضمن مصفوفات اكبر، كما هو الحال للمصفوفة العددية، واليك المثال التالي:

```
>> A
A=
    [3×3 double]          [2.0000 + 3.0000i]
    'Ali Ahmed'           [1×7 double]

>> B
B =
     [1×2 double]         'John Smith'
     [2.0000 + 3.0000i]    [5]
>> C = [A; B]            (متساوية الأبعاد)
C =
    [3×3 double]          [2.0000 + 3.0000i]
    'Ali Ahmed'           [1×7 double]
    [1×2 double]          'John Smith'
    [2.0000 + 3.0000i]    [5]
```

يمكن تحديد خلايا جزئية لإنشاء خلايا جديدة عبر استخدام تقنيات مناسبة لعنونة مصفوفة الخلية كما في المثال التالي:

```
>> D = C ([1 3], :)
D =
```

```
    [3×3 double]        [2.0000 + 3.0000i]
    [1×2 double]        'John Smith'
```

ويمكن حذف سطر مصفوفة الخلية عبر استخدام الخلية الفارغة.

```
>> C (3, :) = [ ]
C =
    [3×3 double]        [2.0000 + 3.0000i]
    'Ali Ahmed'         [1×7 double]
    [2.0000 + 3.0000i]  [5]
```

ويستخدم الايعاز reshape لتغير مواضع الخلايا، ولكنه لا يستطيع إضافة أو حذف الخلايا وليبان ذلك، نأخذ المثال التالي:

```
>> x = cells (3, 4);
>> size (x)
ans =
     3    4
>> y = reshape (x, 6, 2);
>> size (y)
ans =
     6    2
```

أي إن الايعاز reshape يعيد تشكيل أية مصفوفة بدون تغير نوعها، وكذلك يعيد الايعاز size حجم أي نوع من المصفوفات.

كذلك يعيد الايعاز repmat بالتعامل مع مصفوفات الخلايا حيث يعمل على تكرارها وفقاً للتكرار المطلوب.

مثال:

```
>> y
y =
    [ ]    [ ]
    [ ]    [ ]
```

```
    [ ]  [ ]
    [ ]  [ ]
    [ ]  [ ]
    [ ]  [ ]
>> z = repmat (y, 1, 3)
z =
    [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
    [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
    [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
    [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
    [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
    [ ]  [ ]  [ ]  [ ]  [ ]  [ ]
```