الفصل الثاني

حلول نظام المعادلات الخطية على MATLAB

(٢,١) نظام المعادلات الخطية Systems of Linear Equations

من أهم أسباب تطور علم الرياضيات هو البحث عن حلول لمسائل تطبيقية، وعندما نمثّل نظاماً تطبيقياً رياضياً فإننا أحيانا نلجأ لكتابته على شكل نظام معادلات خطية، وفي هذا الفصل سنعرض كيف يتم إيجاد حلول مباشرة لهذه الأنظمة على MATLAB، كما سنتطرق للطرق التكرارية في نهاية الفصل. يتم تمثيل المعادلات الخطية بمصفوفات ومتجهات وبذلك يكون MATLAB من أنسب البرامج لدراسة أنظمة المعادلات الخطية.

بصفة عامة، لإنشاء نظامٍ من n معادلات خطية في n متغيرات $x_1, x_2, \ldots x_n$ نكتب:

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

حيث يُمثل النظام كمصفوفة بـ Ax=b . وترمز A للمصفوفة المربعة بحجم nxn المكونة من المعاملات، و b متجه الطرف الأيمن بقياس n ونبحث عن الحل المتعلق بمتجه المجهول $x_1, x_2, \ldots x_n$.

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad \mathrm{x}=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

إذا فرضنا أن مصفوفة المعاملات A هي مصفوفة غير شاذة، أي يوجد لها معكوس، فبذلك يكون للنظام حل وحيد unique solution ، ويمثل بالمعادلة $x=A^{-1}b$. أما إذا كانت A غير مربعة و عدد المعادلات أكثر من عدد المتغيرات فإن النظام يُعدّ over-determined ولا يوجد حل له. وإذا كان عدد المعادلات أقل من عدد المتغيرات في النظام فإنه يُعدّ under-determined و يوجد عدد غير منتهٍ من الحلول.

(٢,٢) حل نظام المعادلات الخطية Ax=b باستخدام \ على MATLAB

يتم حل نظام معادلات خطية على **MATLAB** باستخدام أداة القسمة باليسار \ وهي تختلف عن القسمة باليمين / بالنسبة للمتجهات والمصفوفات. فعند ادخال A\B فهذا يكافئ inv(A)*B ، أما B/A فهو يكافئ B*inv(A) .

مثال رقم (٢,١)

نفرض أن لدينا النظام :

$$\begin{aligned} 3x_1+2x_2-x_3&=10 \\ -x_1+3x_2+2x_3&=5 \\ \mathrm{x}_1-x_2-x_3&=-1 \end{aligned}$$

ندخل في **MATLAB** مصفوفة المعاملات A ومتجه الطرف الأيمن b :

```
>> A
A =
    3    2   -1
   -1    3    2
    1   -1   -1
>> b=[10 5 -1]'
b =
   10
    5
   -1
>> A\b
ans =
  -2.0000
   5.0000
  -6.0000
```

بتطبيق عملية القسمة من اليسار نحصل على الحل $x_1= -2, x_2 = 5, x_3= -6$.

يمكن أيضاً استخدام المعكوس لإيجاد الحل بحساب $A^{-1}b$ ، ولكن الحل بالأداة \ أدق وأقل استهلاكاً للعمليات الحسابية .

```
>> inv(A)*b
ans =
  -2.0000
   5.0000
  -6.0000
```

عند استعمال عملية القسمة باليسار \ لحل نظام Ax=b فإن MATLAB يختار الطريقة الأنسب والأقل تكلفة حسب نوع المصفوفة A :

- إذا كانت A مصفوفة مثلثية (علوية أو سفلية) فإن MATLAB يستخدم التعويض التراجعي أو الأمامي فقط Backward or Forward substitution .
- إذا كانت A مصفوفة مربعة فإن MATLAB يستخدم الحذف الجاوسي Gaussian elimination .
- إذا كانت A مصفوفة غير مربعة فإن MATLAB يستخدم التحليل QR Factorization .

• إذا كانت A مصفوفة مربعة و sparse فإن MATLAB يستخدم التحليل Cholesky Factorization .

مثال رقم (٢،٢)

إذا كان عدد المعادلات أكثر من عدد المتغيرات في النظام over-determined system مثل :

$$
\begin{aligned}
x_1 + x_2 &= 2\\
2.05x_1 - x_2 &= 1\\
3.06x_1 + x_2 &= 3.5\\
-x_1 + 2x_2 &= 0.92\\
4x_1 + x_2 &= 3
\end{aligned}
$$

فإن MATLAB يقدّم الحل بطريقة التقريب بأصغر مربعات least squares approximation. ندخل مصفوفة المعاملات ومتجه اليمين ونستخدم القسمة باليسار :

```
>> c=[1 1 ;2.05 -1;3.06 1;-1 2;4 1]
c =
   1.0000    1.0000
   2.0500   -1.0000
   3.0600    1.0000
  -1.0000    2.0000
   4.0000    1.0000
>> d=[2;1;3.5;.92;3]
d =
   2.0000
   1.0000
   3.5000
   0.9200
   3.0000
>> c\d
ans =
   0.7159
   0.8087
```

ولأن النظام غير متّسق inconsistent system فإن الحل سيكون تقريبياً لبعض المعادلات وليس لكلها. في المقابل إذا كان عدداً المتغيرات أكثر من عدد المعادلات under-determined system فالناتج سيكون عدد غير منتهي من الحلول.

مثال رقم (٢،٣)

أوجد حل النظام التالي :

$$x_1 + 2x_2 + 3x_3 = 1$$
$$-4x_1 + 2x_2 + 5x_3 = 3$$

```
>> a=[1 2 3;-4 2 5]
a =
    1    2    3
   -4    2    5
>> b=[1;3];
>> a\b
ans =
  -0.2353
        0
   0.4118
```

يحسب MATLAB الحل بأداة \ ، و نلاحظ أن MATLAB عيّن القيمة صفر اختيارياً للمتغير x_2 ، و لم يتم تحذير المستخدم على أن هذا الحل هو واحد فقط من بين عدد غير منتهٍ من الحلول .

(٢،٢،١) الصيغة الدرجية الصفية المختزلة (RREF)

توجد لدى MATLAB دالة *rref* ، وهي تحول المصفوفة إلى الصيغة الدرجية الصفية المختزلة Reduced Row Echelon Form (RREF) . وهذه الصيغة تتحقق بالمواصفات التالية :

١- في كل صف غير صفري يجب أن يكون أول عنصر غير صفري فيه يســاوي ١.

٢- الصفوف الصفرية (إن وجدت) يجب أن تكون في أسفل المصفوفة.

٣- إذا وجد صفان غير صفريين فإن العنصر المتقدم ١ في الصف الأعلى يجب أن يكون على يسار العنصر المتقدم ١ في الصف الأسفل، ويكون باقي العمود (الذي يحتوي على ١) أصفاراً.

لنظام المعادلات Ax=b ننشئ المصفوفة الموسّعة [A b] augmented matrix بضم المصفوفة A مع المتجه b، وإذا تم تحويل هذه المصفوفة إلى شكلها RREF فيمكن استنتاج التالي :

- إذا صدرت [A b] من نظام غير متسق inconsistent system فإن في مصفوفة RREFسيكون هناك صفٌ على شكل [0...01] .
- إذا صدرت [A b] من نظام متسق consistent system بعدد غير منتهٍ من الحلول فإنه في مصفوفة المعاملات في RREF سيكون عدد الأعمدة أكثر من عدد الصفوف غير الصفرية، أو أن هناك حلاً وحيداً للنظام، وسيظهر في آخر عمود في RREF .
- إذا ظهر صف صفري في مصفوفة فهذا يدل على أن النظام الأصلي يحتوي على معادلة مكررة .

نستنتج من ذلك أن في نظام متسق Ax=b وفي حال كون A مصفوفة مربعة مع وجود حل وحيد، فإن الشكل RREF للمصفوفة A هو مصفوفة الوحدة. المثال (٢،٤) يوضح ذلك :

مثال رقم (٤,٢)

إذا كان لدينا نظام Ax=b حيث إن A و b :

```
A =
   8   1   6
   3   5   7
   4   9   2
>> b=ones(3,1)
b =
   1
   1
   1
```

استخدام *rref* على المصفوفة [A b] يعطي مصفوفة الوحدة، و عمود الحل هو العمود الذي يمكن فصله عن طريق *x(:,4)*:

```
>> [x, pivot]=rref([A b])
x =
   1.0000        0        0   0.0667
        0   1.0000        0   0.0667
        0        0   1.0000   0.0667
pivot =
   1   2   3
>> x=x(:,4)
x =
   0.0667
   0.0667
   0.0667
```

أما المتجه pivot الناتج من *rref* فيخّزن أماكن عمود المحورة pivot columns indices ويمكن استخدامه لحساب رتبة A، وللتأكد نحسب رتبة المصفوفةA بالأمر *rank*:

```
>> length(pivot)
ans =
   3
>> rank(A)
ans =
   3
```

استخدام آخر لدالة *rref* هو إيجاد معكوس المصفوفةA وذلك بتطبيق دالة *rref* على المصفوفة الموسعة لـ A ومصفوفة الوحدة. فيظهر معكوسA في الأعمدة الأخيرة:

```
H=rref([A eye(size(A))])
H =
   1    0    0   -1    3    7
   0    1    0    1   -2   -5
   0    0    1   -2    5   11

>>B=H(:,4:6)
B =
  -1    3    7
   1   -2   -5
  -2    5   11
```

للتأكد من المعكوس B نحسب A*B=B*A=I :

```
>> B*A
ans =
    1    0    0
    0    1    0
    0    0    1

>> A*B
ans =
    1    0    0
    0    1    0
    0    0    1
```

أو عن طريق دالة *inv(A)* التي تطابق المصفوفة *B*:

```
>> inv(A)

ans =
  -1.0000    3.0000    7.0000
   1.0000   -2.0000   -5.0000
  -2.0000    5.0000   11.0000
```

كما يوجد الأمر *rrefmovie* الذي يُمكن المستخدم من رؤية المصفوفات الناتجة في كل خطوة من عملية الاختزال، و ذلك بالضغط على أي حرف من لوحة المفاتيح حتى يصل للصيغة النهائية، مثل:

```
Original matrix

A =
     8      1      6
     3      5      7
     4      9      2
Press any key to continue. . .
  pivot = A(1,1)

A =
     1      1/8      3/4
     3      5        7
     4      9        2
Press any key to continue. . .
```

(٢,٢,٢) دالة Solve

يوجد في البيئة الرمزية symbolic إمكانية حل نظام من المعادلات من ذوات الحلول الفعلية exact solutions ، وذلك بالأمر solve، الذي يستخدم على المعادلة لينتج الحلول.

مثال رقم (٢,٥)

لحل النظام في البيئة الرمزية:

$$x+2y=8$$
$$3x+4y=18$$

```
>> syms x y
>> [x0,y0]=solve(x+2*y-8,3*x+4*y-18)
x0 =
2
y0 =
 3
```

(٢,٣) حل نظام المعادلات الخطية بالحذف الجاوسي Gaussian Elimination

الحذف الجاوسي طريقة عملية لحل نظام معادلات، خاصةً الأنظمة ذات المعاملات الصفرية القليلة. وتعتمد الطريقة على عمليات التبسيط الثلاث الأساسية على صفوف النظام (المعادلات):

١- التبديل بين الصفوف .

٢- ضرب الصف بعدد ثابت غير صفري.

٣- التعويض عن صف بحاصل جمع الصف ذاته وصف آخر مضروب بعدد ثابت.

باختصار، طريقة الحذف الجاوسي تُحول المصفوفة الموسعة للنظام إلى مصفوفة مثلثية علوية، ومن ثم نستخدم التعويض التراجعي لحساب قيم المتغيرات.

مثال رقم (٢,٦)

نفرض أن لدينا النظام :

$$
\begin{aligned}
3x_1 + 2x_2 - x_3 &= 10 \\
-x_1 + 3x_2 + 2x_3 &= 5 \\
x_1 - x_2 - x_3 &= -1
\end{aligned}
$$

ننشئ المصفوفة الموسعة :

$$
[A\ \ b] = \begin{bmatrix} 3 & 2 & -1 & 10 \\ -1 & 3 & 2 & 5 \\ 1 & -1 & -1 & -1 \end{bmatrix}
$$

وهنا نستطيع استخدام *rref* على هذه المصفوفة، وهذا موازٍ لعمل الحذف الجاوسي مع التعويض التراجعي وسيظهر متجه الحل في العمود الرابع :

```
>> A=[3 2 -1 10;-1 3 2 5;1 -1 -1 -1];
>> G=rref(A)
G =
   1   0   0  -2
   0   1   0   5
   0   0   1  -6

>> x=G(:,4)
x =
  -2
   5
  -6
```

بطريقة أخرى يمكن كتابة m-file لبرنامج Gaussian [7] ويتم حفظه تحت اسم Gaussian.m (خوارزمية ٢.١) مع مراعاة أن يبدأ الملف بكلمة function :

```
function x=Gaussian(B)
[n,t]=size(B);G=B;
for i=1:n-1
  for j=i:n-1
    m=G(j+1,i)/G(i,i);
   for k=1:t
      G(j+1,k)=G(j+1,k)-m*G(i,k);
   end
  end
end
j=n;x(j,1)=G(j,t)/G(j,j);
  for j=n-1:-1:1
   w=0;
   for k=n:-1:j+1
    w=w+G(j,k)*x(k,1);
   end
   x(j,1)=(G(j,t)-w)/G(j,j);
  end
disp(G)
```

خوارزمية (٢,١).

ويتم استخدام البرنامج Gaussian المعطى في الخوارزمية (٢.١) على المصفوفة الموسعة A للمثال السابق بكتابة الأمر *Gaussian(A)* لنحصل على المصفوفة الناتجة من إجراء خطوات الحذف الجاوسي ثم متجه الحل.

```
>> Gaussian(A)

   3.0000    2.0000   -1.0000   10.0000
        0    3.6667    1.6667    8.3333
        0         0    0.0909   -0.5455

  -2.0000
   5.0000
  -6.0000
```

عند وجود العدد صفر على قطر المصفوفة يمكن تبديل ترتيب المعادلات لتفادي ذلك قبل إجراء الحذف الجاوسي. وهناك طرق للمحورة pivoting strategies يتم إجراؤها مع الحذف الجاوسي ليس فقط لمنع ظهور الأصفار على القطر ولكن لوضع أكبر عدد في كل صف على القطر وذلك لتحاشي القسمة على عدد صغير، مما يؤدي إلى تراكم أخطاء التدوير، وتدعى هذه الطريقة بمحورة جزئية partial pivoting. وهناك أيضا محورة كاملة total pivoting التي يتم فيها المحورة على كل المصفوفة أي يمكن أن يتم التبادل بين الأعمدة أيضا حسب الحاجة. يمكن للقارىء كتابةm-files على MATLAB لبرامج الحذف الجاوسي مع محورة جزئية أو كاملة ومن ثم تطبيقها.

(٢,٤) حل نظام المعادلات الخطية بالتحليل Factorization

توجد طريقة أخرى مباشرة لحل النظام Ax=b وهي إعادة كتابة المصفوفةA على شكل جداء مصفوفتين، سنعرض في هذا الجزء بعض الجداءات المختلفة.

(٢,٤,١) التحليل A=LU

بحيث تكون L مصفوفة مثلثية سفلية وU مصفوفة مثلثية علوية، وكلاهما بنفس حجم A .

$$A = LU = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{12} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

تُعد طريقة التحليلLU طريقةً مباشرةً لحل نظام معادلات خطية، وتفيد في حال وجود أكثر من متجه b في الطرف الأيمن أو إذا كان b غير معلوم، لأن إيجاد L و U لا يعتمد على الطرف الأيمن b. الطريقة تعتمد على تحويل النظام Ax=b إلى LUx=b ، ثم نفرض أن y=Ux لتنتج Ly=b ، وهنا نستخدم التعويض الأمامي لأن L مصفوفة مثلثية سفلية. ولإيجاد متجه الحل x للنظام نستخدم التعويض التراجعي لحل Ux=y لأن U مصفوفة مثلثية علوية .

كما أن هناك عدة أشكال للتحليل على حسب اختيار قيم القطر للمصفوفتين L و U:

- إذا وضعنا قيم قطر L العدد 1 فإن التحليل يدعى طريقة دووليتل Doolittle's method .
- إذا وضعنا قيم قطر U العدد 1 فإن التحليل يدعى طريقة كراوت Crout's method .
- إذا كان النظام متناظراً symmetric و positive definite معرّفة إيجابيا أي $x^t A x > 0$ لكل متجه غير صفري x ، فإن التحليل $A=L L^t$ يدعى طريقة شلوسكي Cholesky method .

(٢،٤،٢) حل نظام معادلات خطية بتحليل A=LU

يقوم MATLAB بإيجاد التحليل A=LU بالأمر lu(A) ، فمثلاً تحليل المصفوفة التالية :

```
>>A =
    3    2   -1
   -1    3    2
    1   -1   -1

>> [L,U]=lu(A)

L =
    1.0000         0          0
   -0.3333    1.0000          0
    0.3333   -0.4545     1.0000

U =
    3.0000    2.0000    -1.0000
         0    3.6667     1.6667
         0         0     0.0909
```

ينتج lu المصفوفة السفلية

والمصفوفة العلوية

إذا حددنا المتجه اليمين b في النظام Ax=b :

```
>> b=[10 5 -1];
```

و نبدأ بالخطوة الأولى: حل النظام الأول Ly=b باستخدام الأداة \ لإيجاد المتجه y :

```
>> y=L\b'
y =
  10.0000
   8.3333
  -0.5455
```

الخطوة الثانية : إيجاد متجه الحل x بحل النظام Ux = y :

```
>> x=U\y
x =
-2.0000
5.0000
-6.0000
```

نلاحظ أن MATLAB يستخدم تحليل دووليتل Doolittle's ، وإذا أردنا تحليل كراوت Crout's فيجب كتابته في m-file .

(٢,٤,٣)حل نظام معادلات خطية تحليل شلوسكي $B = L^t L$

يوجـد في MATLAB تحليـل شلوسـكي cholesky بالدالـة الجـاهزة *chol(A)* . ويتوجب على المصفوفة أن تكون متناظرة symmetric ومعرّفة إيجابياً positive definite مثل :

```
>> B
B =
  2.0000          0 - 1.0000i     0
  0 + 1.0000i     2.0000          0
  0               0               3.0000
```

إذا طلبنا التحليل الشلوسكي للمصفوفة نحصل على المصفوفة L ، وللتأكد من التحليل نحسب $B = L^t L$:

```
>> L=chol(B)
L =
  1.4142          0 - 0.7071i     0
       0          1.2247          0
       0          0               1.7321

>> L'*L
ans =

  2.0000          0 - 1.0000i     0
  0 + 1.0000i     2.0000          0
  0               0               3.0000
```

مثال رقم (٢,٦)

إذاعرّفنا المتجه الأيمن $b = (1, 2, 3)$ نستطيع إيجاد حل النظام $B\,x = b$، بحل $L^t y = b$ ثم $L\,x = y$ باستخدام \:

```
>> y=L'\b'
y =
  0.7071
  1.6330 - 0.4082i
  1.7321

>> L\y
ans =
  0.6667 + 0.6667i
  1.3333 - 0.3333i
  1.0000
```

(٢,٤,٤) تحليل A = QR

يوفّر MATLAB دالة *qr* لتحليل $A = Q\,R$ بحيث تكون R مصفوفة مثلثية علوية و Q مصفوفة متعامدة، (تكون Q متعامدة إذا $Q^{-1} = Q^t$)، وليس من الضروري أن تكون المصفوفة A مربعة. فيصبح الحل للنظام $A\,x = b$ هو الحل للنظام $R\,x = Q^t b$.

```
>> A=[4 -2 7;6 2 -3; 3 4 5];
>> [Q,R]=qr(A)
Q =
  -0.5121   0.6852   0.5179
  -0.7682  -0.0958  -0.6330
  -0.3841  -0.7220   0.5754

R =
 -7.8102  -2.0486  -3.9691
       0  -4.4501   0.0295
       0        0   9.5522
```

(٢,٤,٥) تحليل القيمة الشاذة svd

يوفر MATLAB دالة svd التي تقدم تحليل القيمة الشاذة singular value decomposition، A=USV لمصفوفة A بحجم mxn و بحيث تكون مصفوفةU متعامدة بحجم mxm ، وV مصفوفة متعامدة بحجم nxn و مصفوفة S قطرية بحجم mxn ، كما أن القيم على قطر S تسمى القيم الشاذة وعددها يساوي رتبة المصفوفة . يُعد هذا النوع من التحليل الأكثر ضماناً ولكنه يحتاج إلى كمية حسابات أكثر من غيره. يتم استخدام svd غالباً في حل مسائل أصغر المربعات least squares problems والطرق المثلى .

لحساب تحليل svd للمصفوفة A :

```
>> A=[1 2 3;4 5 9;7 11 18;-2 3 1;7 1 9]
A =
    1    2    3
    4    5    9
    7   11   18
   -2    3    1
    7    1    9

>> [u,s,v]=svd(A)
u =
  -0.1364   0.0871   0.0284  -0.2001  -0.9659
  -0.4069   0.0334  -0.2544  -0.8465   0.2283
  -0.8161   0.2947  -0.1691   0.4656   0.0404
  -0.0511   0.5609   0.8018  -0.1632   0.1152
  -0.3836  -0.7680   0.5128   0.0000   0.0000

s =
  27.1420        0         0
        0   6.1825         0
        0        0    0.2968
        0        0         0
        0        0         0

v =
  -0.3706  -0.6816  -0.6309
  -0.4355   0.7275  -0.5301
  -0.8203  -0.0783   0.5665
```

وإذا أدخلنا *svd(A)* فنحصل على القيم القطرية فقط، أي القيم الشاذة مباشرة:

```
>> svd(A)
ans =
  27.1420
   6.1825
   0.2968
```

(٢,٥) طرق تكرارية Iterative Methods

في حالة وجود أنظمة خطية صغيرة فالطرق المباشرة السابقة تكون مناسبةً، ولكن للأنظمة الكبيرة و المحتوية على معاملات صفرية عديدة فالطرق التكرارية تكون أكثر عملية من ناحية استهلاك ذاكرة الحاسوب والدقة. الطرق التكرارية لحل النظام Ax=b تبدأ بقيمة تقريبية ابتدائية $x^{(0)}$ لحل النظام x ، وتوّلد متتالية $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ من المتجهات التي تتقارب من x . معظم الطرق التكرارية تحّول النظام إلى نظام مكافئ بالشكل x = Tx+c لمصفوفة مربعة T و متجه c . بعد اختيار المتجه الابتدائي $x^{(0)}$ نقوم بتوليد متتالية لمتجهات الحلول التقريبية من $x^{(k+1)} = Tx^{(k)} + c$ لكل k=0,1,2,... . ونتوقف عن التكرار إذا كان الخطأ صغيراً جداً بين المتجهات التقريبية المتتالية أي $\left\|x^{(k+1)} - x^{(k)}\right\| < \varepsilon$ لعدد صغير موجب ε . من بين الطرق التكرارية الأكثر شيوعاً طريقة جاكوبي التكرارية Jacobi iterative method و طريقة جاوس سيدال التكرارية Gauss-Seidel iterative method .

طريقة جاكوبي التكرارية تحل المعادلة رقم i في النظام للحصول على x_i :

$$x_i = \sum_{j=1, i\neq i}^{n} \left(-\frac{a_{ij}x_j}{a_{ii}}\right) + \frac{b_i}{a_{ii}} \qquad i = 1,2,3\ldots,n$$

وتولد $x_i^{(k)}$ باستخدام $x^{(k-1)}$ لكل $k \geq 1$ عن طريق:

$$x_i^{(k)} = \frac{\sum_{j=1, j \neq j}^{n} (-a_{ij} x_j^{(k-1)}) + b_i}{a_{ii}} \qquad i = 1,2,3 \ldots, n$$

أما طريقة جاوس سيدال فتستخدم :

$$x_i^{(k)} = \frac{-\sum_{j=1}^{i-1} (a_{ij} x_j^{(k)}) - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} + b_i}{a_{ii}} \qquad i = 1,2,3 \ldots, n$$

والفرق بين طريقة جاكوبي التكرارية وطريقة جاوس سيدال التكرارية هو أن الأخيرة تستخدم القيم الجديدة لـ x_i كلما حُسبت. يمكن برمجة الطرق التكرارية في m-file [7] واستخدامها كدالة للحصول على الحل التقريبي للنظام. الخوارزمية (٢،٢) تعطي برنامج GaussSeidel هي :

```
function x=GaussSeidel(B,x,tol)
[n,t]=size(B);
b=B(1:n,t); w=1;k=1;
d(1,1:n+1)=[0 x]; k=k+1;
while w>acc
for i=1:n
sum=0;
for j=1:n
if j<=i-1
    sum=sum+B(i,j)*d(k,j+1);
elseif j>=i+1
sum=sum+B(i,j)*d(k-1,j+1);
end ;end;
x(1,i)=(1/B(i,i))*(b(i,1)-sum);
d(k,1)=k-1;d(k,i+1)=x(1,i);
end
w=max(abs((d(k,2:n+1)-d(k-1,2:n+1))));
k=k+1;
if w>100 & k>10
   ('Gauss-Seidel method is Divergent')  end;end;x=d;
```

خوارزمية (٢،٢).

مثال رقم (٢,٧)

أوجد حل النظام بالطرق التكرارية :

$$5x_1 - 1x_2 + x_3 = 10$$
$$2x_1 + 8x_2 - x_3 = 11$$
$$-x_1 + x_2 + 4x_3 = 5$$

نُدخل المصفوفة الموسعة للنظام و المتجه الابتدائي و الدقة المطلوبة للحل :

```
>> B
B =
    5   -1    1   10
    2    8   -1   11
   -1    1    4    5
>> x
x =
    0    0    0
>> tol
tol =
  1.0000e-006
```

نستعمل برنامج جاكوبي لإيجاد حل النظام السابق ، بإدخال *Jaccobi(B,x,tol)* (الملحق) [7] بالمدخلات اللازمة ، و يعرضMATLAB النتائج في العمود الأول ، رقم خطوة التكرار ، والأعمدة الثلاثة التالية تعطي إحداثيات متجه الحل .

نلاحظ أن طريقة جاكوبي استغرقت ١٦ خطوة تكرارية لإيجاد الحل x= [1.923, 1.076, 1.461] بالدقة المطلوبة.

```
Jaccobi(B,x,tol)
ans =    k          x1         x2         x3
      0          0          0          0
   1.0000    2.0000    1.3750    1.2500
   2.0000    2.0250    1.0313    1.4063
   3.0000    1.9250    1.0445    1.4984
   4.0000    1.9092    1.0811    1.4701
```

5.0000	1.9222	1.0815	1.4570
6.0000	1.9249	1.0766	1.4602
7.0000	1.9233	1.0763	1.4621
8.0000	1.9228	1.0769	1.4617
9.0000	1.9230	1.0770	1.4615
10.0000	1.9231	1.0769	1.4615
11.0000	1.9231	1.0769	1.4615
12.0000	1.9231	1.0769	1.4615
13.0000	1.9231	1.0769	1.4615
14.0000	1.9231	1.0769	1.4615
15.0000	1.9231	1.0769	1.4615

للمقارنة ، نكتب الأمر *GaussSeidel* بالمدخلات اللازمة. ونلاحظ أن طريقة جاوس سيدال التكرارية احتاجت فقط ١١ خطوة تكرارية للوصول للحل بنفس الدقة :

>> GaussSeidel(B,x,tol)

ans =

k	x_1	x_2	x_3
0	0	0	0
1.0000	2.0000	0.8750	1.5313
2.0000	1.8688	1.0992	1.4424
3.0000	1.9314	1.0725	1.4647
4.0000	1.9215	1.0777	1.4610
5.0000	1.9233	1.0768	1.4616
6.0000	1.9230	1.0769	1.4615
7.0000	1.9231	1.0769	1.4615
8.0000	1.9231	1.0769	1.4615
9.0000	1.9231	1.0769	1.4615
10.0000	1.9231	1.0769	1.4615

وإذا كانت كل من الطريقتين جاكوبي و جاوس سيدال تتقارب من متجه الحل ، فإن جاوس سيدال هو الأسرع ، ولكن ليس دائماً لأن هناك أنظمة تكون فيها طريقة جاكوبي متقاربة ، ولكن جاوس سيدال متباعدة ، أو العكس. لنضمن التقارب لكل من الطريقتين ، يجب التحقق من شرط في النظام وهو أن تكون مصفوفة المعاملات للنظام $A=(a_{ij})$ مربعة بحجم nxn ومسيطرة قطرياً بدقة strictly diagonally dominant أي :

$$|a_{ii}| > \sum_{\substack{j=1 \\ i \neq j}}^{n} |a_{ij}| \quad \text{for } i = 1,2,\ldots,n$$

(٢،٦) مسائل القيم الذاتية Eigenvalue problem

تظهر مسائل القيم الذاتية في كثير من تطبيقات الجبر الخطي في فروع العلوم الطبيعية والهندسة، وصيغة هذا النوع من المسائل العامة تأخذ الشكل $Ax=\lambda x$ وهي معادلة جبرية للقيم الذاتية لـA ، حيث إن A مصفوفة مربعة بحجم nxn ونقول ان العدد λ قيمة ذاتية أومميزة Eigenvalue للمصفوفة A إذا وجد متجه x غير صفري يسمى متجهاً ذاتياً Eigenvector بحيث إن $Ax=\lambda x$ وهي . ويمكن إعادة كتابة مسألة القيمة الذاتية على شكل نظام المعادلات $(A-\lambda I)x=0$ وهي حيث إن I هي مصفوفة الوحدة بحجم nxn . ويكون هناك حلٌ للنظام إذا وإذا فقط كان $(A-\lambda I)$ نظاماً شاذاً أو $det(A-\lambda I)=0$. وهذه المعادلة هي كثيرة حدود بدرجة n في المتغير λ وتسمى المعادلة الذاتية characteristic equation لـ A. جذور المعادلة الذاتية هي القيم الذاتية $\lambda_1, \lambda_2, \ldots, \lambda_n$ ولكن المتجه الذاتي x المقابل لكل قيمة ذاتية λ_i ليس وحيداً.

في برنامج MATLAB نقوم بحساب القيم الذاتية بالأمر *eig* كما في المثال التالي.

مثال رقم (٢،٩)

إذا كان لدينا مصفوفة A :

```
>> A= [-6 0 0;11 -3 0;-3 6 7]
>> [X,D]=eig(A)
X =
      0        0           0.2348
      0        0.8575   -0.8608
   1.0000  -0.5145    0.4515
```

```
D =
    7    0    0
    0   -3    0
    0    0   -6

>> lambda=eig(A)
lambda =
    7
   -3
   -6
```

ينتج من الأمر *[X,D]=eig(A)* مصفوفتان X و D . الأعمدة في المصفوفة X تحتوي على المتجهات الذاتية المقابلة للقيم الذاتية التي تظهر على قطر المصفوفة D ، ويمكن إيجاد القيم الذاتية مباشرة بالدالة *eig(A)* فقط ، وتخزينها في متجه lambda . لإيجاد كثيرة الحدود المميزة نوجد أولاً المعاملات بدالة *poly* ثم بدالة *poly2sym* التي تحولها إلى معادلة في المتغير x .

```
>> coefChar=poly(A)
coefChar =
    1    2   -45  -126
>> CharEq=poly2sym(coefChar)
CharEq =
x^3+2*x^2-45*x-126
```

لقد عرضنا في هذا الباب أهم الطرق لحل أنظمة المعادلات الخطية ، وقدرة **MATLAB** على معالجتها بصورة عملية و دقيقة. يستطيع القارئ تطوير البرامج التي عرضت لاستخدامها في تطبيقات أخرى ، ومن ثم تسخيرها في إيجاد حلول لمسائل فيزيائية وهندسية مختلفة.

تمارين (٢,٧)

١- أوجد حل الأنظمة الخطية التالية باستعمال دالة " \ " وقارن باستخدام المعكوس :

أ)
$$\begin{aligned} 5x_1+3x_2+x_3&=3\\ 2x_1+3x_2+x_3&=-1\\ x_1+x_2+3x_3&=2 \end{aligned}$$

ب)
$$\begin{aligned} x_1+3x_2-x_3&=4\\ 2x_1+2x_2+x_3&=9\\ 5x_1-2x_2-x_3&=-2 \end{aligned}$$

جـ)
$$\begin{aligned} 2x_1-x_2+x_3&=-1\\ 3x_1+3x_2+9x_3&=0\\ 3x_1+3x_2+5x_3&=4 \end{aligned}$$

د)
$$\begin{aligned} 2x_1&=3\\ x_1+1.5x_2&=4.5\\ -3x_2+0.5x_3&=-6.6\\ 2x_1-2x_2+x_3+x_4&=0.8 \end{aligned}$$

٢- حل الأنظمة السابقة بالتحليل *A=LU* إن أمكن .

٣- حل الأنظمة السابقة بالحذف الجاوسي.

٤- أوجد حل النظام التالي بالحذف الجاوسي ، مع المحورة الجزئية ، وقارن الحل بالمحورة الكاملة :

$Ax=b$ إذا $a_{ij}=1/(i+j-1)$ لكل $i,j=1,2,....,n$.

٥- استخدم طريقة جاوس سيدال تكرارية لإيجاد حل النظام :

$$\begin{aligned} 2x_1+x_2+4x_3&=16\\ 4x_1+2x_2+x_3&=11\\ -x_1+2x_2&=3 \end{aligned}$$

مبتدئاً بالمتجه $x=[1,1,1]^t$.

٦- قارن حل النظام السابق بطريقة جاكوبي التكرارية.

٧- أوجد التكرارين الأولين من طريقة جاكوبي باستخدام $\mathbf{x}^{(0)}=\mathbf{0}$:

أ)
$$\begin{aligned} 10x_1-x_2\quad\;&=1\\ -x_1+10x_2-2x_3&=7\\ -2x_2+10x_3&=6 \end{aligned}$$

ب)
$$\begin{aligned} 3x_1-x_2+x_3&=1\\ 3x_1+6x_2+2x_3&=0\\ 3x_1+3x_2+7x_3&=4 \end{aligned}$$

جـ)

$$\begin{aligned} 4x_1 + x_2 - x_3 + x_4 &= -2 \\ x_1 + 4x_2 - x_3 - x_4 &= -1 \\ -x_1 - x_2 + 5x_3 + x_4 &= 0 \\ x_1 - x_2 + x_3 + 3x_4 &= 1 \end{aligned}$$

د)

$$\begin{aligned} 10x_1 + 5x_2 &= 1 \\ 5x_1 + 10x_2 - 4x_3 &= 25 \\ -4x_2 + 8x_3 - x_4 &= -11 \\ -x_3 + 5x_4 &= -11 \end{aligned}$$

٨- أوجد التكرارين الأولين في طريقة جاوس سيدال باستخدام $\mathbf{x}^{(0)}=\mathbf{0}$ للأنظمة في التمرين رقم ٧ .