1. ОПЕРАЦИИ СО СЛАБОПОПОЛНЕТИ МАТРИЦИ

Познато е дека решението на еден систем линеарни равенки напишан во матричен облик на следниот начин

$$\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}\,, \tag{1}$$

наједноставно може да се изрази со помош на инверзната матрица на системот како

$$\boldsymbol{x} = \boldsymbol{A}^{-1} \cdot \boldsymbol{b}\,, \tag{2}$$

но таквиот начин на одредување на векторот со непознатите променливи $\boldsymbol{x}$ не е секогаш најефикасен.

Постојат случаи во коишто матрицата $\boldsymbol{A}$ содржи многу елементи со вредност нула при што таа се нарекува слабопополнета матрица (sparse matrix на англиски). Во таквите случаи далеку подобро е да се користат посебни методи за решавање, но и за меморирање на елементите на матрицата $\boldsymbol{A}$.

Нека е даден следниот систем равенки во којшто матрицата $\boldsymbol{A}$ е слабополнета

$$\begin{bmatrix} 3 & 0 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 7 & 5 & 9 & 0 \\ 0 & 0 & 0 & 8 & 2 \\ 0 & 0 & 0 & 6 & 5 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \\ 22 \\ 27 \end{bmatrix}, \tag{3}$$

којшто, според досегашното знаење за Matlab, можеме да го решиме на следниот начин

```
A = [3 0 1 0 0; 0 4 0 0 0; 0 7 5 9 0; 0 0 0 8 2; 0 0 0 6 5];
b = [3; 4; 25; 22; 27];
x = A^-1 * b
x =
    1.0000
    1.0000
   -0.0000
    2.0000
    3.0000
```

Во претходно опишаната постапка дефиниравме матрица $\boldsymbol{A}$ со 5 редици и 5 колони која што содржи 25 елементи. Но долколку во командниот простор ја напишеме наредбата

```
nenulti = nnz(A)
nenulti =
    10
```

гледаме дека таа има само 10 ненултри елементи, што значи дека неа можеме да ја мемориграме користејќи далеку помалку меморија.

Матрицата $\boldsymbol{A}$ можеме да ја претвориме во слабопополнета матрица во Matlab со помош на командата

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

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

од каде што гледаме во новата матрица ***A*** се запомнети само ненултите елементи и локациите каде што тие се наоѓаат. Но секако дека ваквата постапка нема да ја користиме за да ја формираме матрицата ***A*** затоа што претходно треба да ја формираме како полна матрица, со сите 25 елементи, а токму тоа сакаме да го избегнеме.

Една од постапките за пополнување на ретка матрица е следната: формираме помошна матрицата ***B***

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

која што има 10 редици (колку што има ненулти елементи во A) и три колони. Во една редица од матрицата ***B*** се дадени податоци за еден ненулти елемент на матрицата ***A***. При тоа во таа редица во првите две колони се дадени индексите на редицата и колоната од матрицата од матрицата ***A*** каде што се наоѓа ненултиот елемент, како и неговата вредност која што е дадена во третата колона. Така на пример од матрицата ***B*** гледаме дека првиот третиот ненулти елемент во матрицата ***A*** се наоѓа на позицијата (3,2) во матрицата ***A***, а неговата вредност изнесува 7.

Користејќи ја матрицата ***B*** можеме да ја добиеме матрицата ***A*** во форма на слабопополнета матрица со помош на наредбата

```
A = spconvert(B)
```

Но можеме да постапиме и на друг начин. Прво формираме слабопоплнета матрицата со димензии 5×5 во која што сите елементи се нули

```
A = sparse(5,5)
```

```
A =
   All zero sparse: 5-by-5
```

со што сме создале променлива во Matlab која ќе биде запомнета како слабопоплнета матрица и која што на почетокот нема ниту еден елемент. Потоа можеме да ги додаваме елементите во таа празна матрица пишувајќи ги следните наредби

```
A(1,1) = 3; A(2,2) = 4; A(3,2) = 7; A(1,3) = 1; A(3,3) = 5;
A(3,4) = 9; A(4,4) = 8; A(5,4) = 6; A(4,5) = 2; A(5,5) = 5;
```

со што сакаме да покажеме дека до елементите на една слабопополнета матрицат во Matlab можеме да пристапиме и да ги менуваме како и елементите на секоја друга обична (полна) матрица. Оваа постапка тука изгледа покомплицирана за пишување од првата со помошната матрица ***B***, но таа е посебно погодна за случаите во коишто матрицата ***A*** ќе ја формираме програмски како што тоа ќе биде покажано во понатамошните примери.

Откако сме ја формирале слабопополнетата матрица ***A*** системот равенки можеме да го решиме на следниот начин

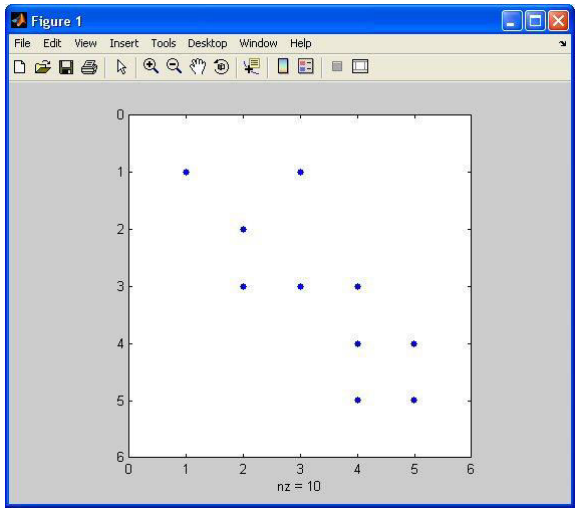
```
x = A\b
x =
     1
     1
     0
     2
     3
```

каде што го користиме операторот „\“ којшто во Matlab означува решавање на систем линеарни равенки каде што променливата пред него е матрицата ***A*** на системот равенки, а променливата по него е векторот од десната страна на системот ***b***. Овој оператор води сметка за слабата пополнетост на матрицата и пресметките ги прави далеку побрзо отколку во случајот кога решението го добивавме преку инверзија на матрицата ***A***.

Доколку сакаме да видиме како изгледа структурата на една матрицата (без разлика дали е слабопополнета или не) во командниот простор пишуваме

```
spy(A)
```

со што ја добиваме сликата 1 каде што со точки се означени позициите на коишто се наоѓаат ненултите елементи во матрицата.



Слика 1 Позиција на ненултите елементи во матрицата

Пример 1. Да се реши систем равенки со 1000 непознати, зададен на следниот начин

$$\begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 3 & -1 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 3 & -1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 3 & -1 \\ 0 & 0 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{998} \\ x_{999} \\ x_{1000} \end{bmatrix} = \begin{bmatrix} 3000 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

каде што матрицата на системот $\boldsymbol{A}$ е слабопополнета. При тоа е познато дека матрицата $\boldsymbol{A}$ има ненулти елементи по главната дијагонала коишто имаат вредности еднаква на 3 освен првиот и последниот коишто имаат вредност еднаква на 2. Покрај тоа, ненулти елемети постојат и на двете соседни дијагонали на главната дијагонала и при тоа сите елементи изнесуваат –1. Во векторот $\boldsymbol{b}$ од десната страна на системот равенки само првиот елемент не е нула и има вредност еднаква на 3000. Проблемот да се реши на два начина:

а) Со матрична инверзија и користење на полни матрици,

б) Со примена на операторот „\“ и користење на слабопополнети матрици.

Решение

Решението за првиот случај е дадено во датотеката primer_1a.m чија што содржина е следната

```
1.  n = 1000; A = zeros(n,n); b = zeros(n,1);
2.
3.  A(1,1) = 2;
4.  for i = 2:n-1
5.      A(i,i) = 3;
6.  end
7.  A(n,n) = 2;
8.
9.  for k = 2:n
10.     A(k,k-1) = -1;
11.     A(k-1,k) = -1;
12. end
13.
14. b(1) = 3000;
15.
16. tic
17. x = A^-1 * b;
18. toc
```

Во првиот ред од програмата ги дефинираме матрицата $\boldsymbol{A}$ и векторот $\boldsymbol{b}$ како матрици со сите елементи еднакви на нули со помош на наредбата zeros. Потоа ја пополнуваме матрицата $\boldsymbol{A}$. Согласно со зададените вредности и обликот на матрицата, во редиците 3-7 ги пополнуваме дијагоналните елементи на матрицата, а во редиците 9-12 вондијагоналните, т.е. елементите на соседните дијагонали на главната дијагонала. Ненултиот елемент во векторот $\boldsymbol{b}$ го задавамево редицата 14. Решавањето на системот равенки се прави во редицата 17, а пред неа и по неа се напишани командите tic и toc со

кои што во Matlab е овозможено мерење на времето за извршувањето на операциите кои се помеѓу нив.

Со активирање на програмата добиваме решение, кое не го прикажуваме заради големината на векторот $\boldsymbol{x}$, а во командниот простор ја добиваме следната порака

```
Elapsed time is 1.116915 seconds.
```

што е всушност времето потребно за решавање на системот равенки користејќи матрична инверзија.

Во случајот под б) проблемот го решаваме со програмата primer_1b.m која што многу малку се разликува од програмата primer_1a.m. Разликите се во редиците 1 и 17 коишто сега гласат

```
 1.  n = 1000; A = sparse(n,n); b = zeros(n,1);
17.  x = A \ b;
```

Резултатот којшто го добиваме во овој случај е идентичен како претходно, а во командниот простор ја добиваме следната порака

```
Elapsed time is 0.000091 seconds.
```

од каде што дека времето потребно за решавање на системот равенки користејќи ја операцијата „\“ е околу 12000 пати помало од времето потребно за решавање на истиот систем равенки со примена на матрична инверзија. Освен тоа, во овој случај меморимаме само 2997 елементи од матрицата $\boldsymbol{A}$, наместо 1.000.000 колку што мемориравме во првиот случај.

□ □ □