Practice 4
Inverses and LU decomposition

Contents

1 Computation of the inverse matrix

1.1 The inv function

To compute the inverse matrix of a square matrix A we use the **inv** function applied over A. If A is invertible then the command

```
-->inv(A)
```

provides the inverse matrix A^{-1}. In other case, either it appears an error message saying that the matrix is singular (that is, non-invertible) or it appears a warning message saying that the matrix is close to be singular or badly scaled. In the last case it is not clear if the matrix is singular or not; we can use, in this situation, the **rank** function to compute the rank of A and check if A has maximum rank.

Example 1. Consider the matrix

$$A = \begin{bmatrix} 2 & 5 & -1 \\ 0 & 0 & 9 \\ 0 & 5 & 4 \end{bmatrix}.$$

```
-->A=[2 5 -1;0 0 9;0 -5 4]
 A  =

    2.    5.  - 1.
    0.    0.    9.
    0.  - 5.    4.

-->inv(A)
 ans  =

    0.5  - 0.1666667    0.5
    0.     0.0888889  - 0.2
    0.     0.1111111    0.
```

We can check that the provided inverse matrix is correct:

```
-->A*inv(A)
ans  =

   1.    5.551D-17    0.
   0.    1.           0.
   0.  - 5.551D-17    1.
```

The entries $(1,2)$ and $(3,2)$ (that is, $5.551 \cdot 10^{-17}$ and $-5.551 \cdot 10^{-17}$) should, probably, be zero (they are not zero, probably, due to rounding errors in the computations). Using the **clean** function we can replace these numbers by 0:

```
-->clean(ans)
 ans  =

    1.    0.    0.
    0.    1.    0.
    0.    0.    1.
```

This is the identity matrix and, therefore, the result is correct.

Example 2. Consider the matrix

$$\mathsf{B} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}.$$

```
-->B=[1 1 1; 2 2 2; 3 3 3]
 B  =

    1.    1.    1.
```

```
    2.    2.    2.
    3.    3.    3.

-->inv(B)
       !--error 19
Problem is singular.
```

Computing the rank of B we can see that it has not maximum rank:

```
->rank(B)
 ans  =

    1.
```

So, the matrix is not invertible in this case.

Example 3. Consider now the matrix

$$C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

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

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

-->inv(C)
warning :
matrix is close to singular or badly scaled. rcond =    0.0000D+00

 ans  =

   1.0D+15 *

  - 4.5035996    9.0071993  - 4.5035996
    9.0071993  - 18.014399    9.0071993
  - 4.5035996    9.0071993  - 4.5035996

-->rank(C)
 ans  =

    2.
```

We conclude that C is not invertible.

1.2 Computing inverse matrices with Gauss-Jordan algorithm

Given a square matrix A, we can also obtain the inverse of A by applying the Gauss-Jordan algorithm to the matrix A1 = [A I], where I is the identity matrix:

Example 4. Let A be the matrix of Example 1.

```
-->A1=[A,eye(3,3)]
 A1  =

    2.    5.  - 1.    1.    0.    0.
    0.    0.    9.    0.    1.    0.
    0.  - 5.    4.    0.    0.    1.

-->rref(A1)
 ans  =

    1.    0.    0.    0.5  - 0.1666667    0.5
    0.    1.    0.    0.     0.0888889  - 0.2
    0.    0.    1.    0.     0.1111111    0.
```

Therefore A is invertible and its inverse is

```
-->ans(:,4:6)
 ans  =

    0.5  - 0.1666667    0.5
    0.     0.0888889  - 0.2
    0.     0.1111111    0.
```

Example 5. We do the same with the matrix C of example 3:

```
-->C1=[C, eye(3,3)]
 C1  =

    1.    2.    3.    1.    0.    0.
    4.    5.    6.    0.    1.    0.
    7.    8.    9.    0.    0.    1.

-->rref(C1)
 ans  =

    1.    0.  - 1.    0.  - 2.6666667    1.6666667
    0.    1.    2.    0.    2.3333333  - 1.3333333
    0.    0.    0.    1.  - 2.           1.
```

Since the matrix on the left is not the identity matrix, C is not invertible.

2 LU decomposition of a square matrix

2.1 Computation

An LU decomposition (also called LU factorization) is a matrix decomposition which writes a matrix as the product of a lower triangular matrix and an upper triangular matrix. This decomposition has several applications. We will present here three of them: resolution of systems of linear equations, computation of the determinant of a square matrix and computation of inverse matrices.

First of all we will explain the process of obtention (by hand) of an LU decomposition by means of a pair of examples. Consider the matrix

$$A = \begin{bmatrix} 2 & 5 & -3 \\ 4 & 7 & -4 \\ -6 & -3 & 1 \end{bmatrix}$$

Performing elementary row operations (that is, multiplying A by suitable elementary matrices) we can obtain, in this case, an upper triangular matrix which is equivalent (by rows) to A:

$$E_{32}(4)E_{31}(3)E_{21}(-2)A = \begin{bmatrix} 2 & 5 & -3 \\ 0 & -3 & 2 \\ 0 & 0 & 0 \end{bmatrix}.$$

Taking $B = E_{32}(4)E_{31}(3)E_{21}(-2)$ and

$$U = \begin{bmatrix} 2 & 5 & -3 \\ 0 & -3 & 2 \\ 0 & 0 & 0 \end{bmatrix}$$

one has that

$$BA = U,$$

where B is a lower triangular matrix and U is an upper triangular matrix. Since B is product of elementary matrices, it is invertible. Multiplying the above equality by B^{-1} (by the left) we get:

$$A = B^{-1}U.$$

It is not difficult to prove that the inverse of a lower triangular matrix is also a lower triangular matrix. Therefore, taking $L = B$ we get an LU decomposition of A:

$$A = LU.$$

It is very easy to compute L by hand because we know the inverses of the elementary matrices:

$$L = B^{-1} = (E_{32}(4)E_{31}(3)E_{21}(-2))^{-1} = E_{21}(-2)^{-1}E_{31}(3)^{-1}E_{32}(4)^{-1} =$$

$$E_{21}(2)E_{31}(-3)E_{32}(-4) = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -4 & 1 \end{bmatrix}.$$

Summarizing, we have obtained the following LU decomposition of A:

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -3 & -4 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & 5 & -3 \\ 0 & -3 & 2 \\ 0 & 0 & 0 \end{bmatrix}}_{U}.$$

Let us see other example. Consider the matrix

$$C = \begin{bmatrix} 0 & -2 & 1 \\ 3 & 0 & 2 \\ 0 & 2 & 4 \end{bmatrix}.$$

If we apply the same process as above to the matrix C we obtain:

$$E_{32}(1)E_{12}C = \underbrace{\begin{bmatrix} 3 & 0 & 2 \\ 0 & -2 & 1 \\ 0 & 0 & 5 \end{bmatrix}}_{U}.$$

Then

$$L = (E_{32}(1)E_{12})^{-1} = E_{12}^{-1}E_{32}(1)^{-1} = E_{12}E_{32}(-1) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 1 \end{bmatrix}.$$

But... this matrix L is not lower triangular! The reason is that, in the process of obtention, we have switched rows (that is, we have performed elementary operations of type 1). This is the difference with the previous example. However, in this case, we will keep the name of "LU decomposition" to refer to the obtained factorization. That is: **an LU decomposition of a matrix A will be a factorization $A = LU$ where U is upper triangular and L is lower triangular (up to reordering of rows).**

LU decompositions can be obtained easily with Scilab using the **lu** function. But... warning! This function applies Gaussian elimination algorithm with partial pivotation to obtain the decomposition. This means that, very probably, Scilab will use row switching in the process and the obtained matrix L will not be lower triangular (although, of course, it will be up to reordering or rows). For example, if we compute, using Scilab, a LU decomposition of the above matrix A:

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

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

```
U  =

 - 6.  - 3.    1.
   0.    5.  - 3.3333333
   0.    0.    4.441D-16
L  =

 - 0.3333333    0.8    1.
 - 0.6666667    1.     0.
   1.           0.     0.
```

"Cleaning" elements "near zero" in the factor U:

```
->clean(U)
 ans  =

 - 6.  - 3.    1.
   0.    5.  - 3.3333333
   0.    0.    0.
```

Observe that the factor L which Scilab has computed is not lower triangular.

One can also use the Scilab command $[\mathsf{L},\mathsf{U},\mathsf{P}] = \mathbf{lu}(\mathsf{A})$ (adding another parameter to the output):

```
-->[L1,U1,P]=lu(A)
 P  =
   0.    0.    1.
   0.    1.    0.
   1.    0.    0.
 U1  =
 - 6.  - 3.    1.
   0.    5.  - 3.3333333
   0.    0.    4.441D-16
 L1  =
   1.           0.     0.
 - 0.6666667    1.     0.
 - 0.3333333    0.8    1.

 -->P*L1*U1
 ans  =
   2.    5.  - 3.
   4.    7.  - 4.
 - 6.  - 3.    1.
```

```
-->U-U1
 ans  =
    0.    0.    0.
    0.    0.    0.
    0.    0.    0.

-->L-P*L1
 ans  =
    0.    0.    0.
    0.    0.    0.
    0.    0.    0.
```

That is, when we write 3 output parameters, $[\mathsf{L},\mathsf{U},\mathsf{P}] = \mathrm{lu}(\mathsf{A})$, it is returned the permutation matrix P such that $\mathsf{PA} = \mathsf{LU}$, where U is upper triangular and L is lower triangular with all its diagonal elements equal to 1; if we write only 2 output parameters, $[\mathsf{L},\mathsf{U}] = \mathrm{lu}(\mathsf{A})$, the matrix U is the same but the matrix L is already given permuted; therefore this L satisfies: $\mathsf{A} = \mathsf{LU}$.

2.2 Applications

2.2.1 Resolution of systems of linear equations

The LU decomposition can be applied to the resolution of systems of linear equations. Let us see an example.

Consider the system

$$\left.\begin{array}{rcl} -2y+z & = & 1 \\ 3x+2y & = & 2 \\ 2y+4z & = & 3 \end{array}\right\},$$

whose coefficient matrix is the matrix C of the above example and whose matrix expression is:

$$\begin{bmatrix} 0 & -2 & 1 \\ 3 & 0 & 2 \\ 0 & 2 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

Replacing the coefficient matrix by its LU decomposition one has

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 1 \end{bmatrix}}_{\mathsf{L}} \underbrace{\begin{bmatrix} 3 & 0 & 2 \\ 0 & -2 & 1 \\ 0 & 0 & 5 \end{bmatrix}}_{\mathsf{U}} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}}_{\vec{b}}. \tag{1}$$

The product $\mathsf{U}\vec{x}$ is a column vector of three components which we will denote by $\vec{y}$, that is:

$$\mathsf{U}\vec{x} = \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}.$$

Therefore, the equality (1) can be written:

$$\underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 1 \end{bmatrix}}_{\mathsf{L}} \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}}_{\vec{y}} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}}_{\vec{b}}.$$

This is the matrix expression of the following system:

$$\left.\begin{array}{rcl} y_2 & = & 1 \\ y_1 & = & 2 \\ -y_2 + y_3 & = & 3 \end{array}\right\} \tag{2}$$

This system has an "almost triangular" coefficient matrix and it can be solved directly by forward substitution, giving rise to this solution:

$$\vec{y} = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix}.$$

Now, since $\mathsf{U}\vec{x} = \vec{y}$ and we know $\vec{y}$, we can compute $\vec{x}$ by solving the system of linear equations whose matrix expression is $\mathsf{U}\vec{x} = \vec{y}$:

$$\underbrace{\begin{bmatrix} 3 & 0 & 2 \\ 0 & -2 & 1 \\ 0 & 0 & 5 \end{bmatrix}}_{\mathsf{U}} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\vec{x}} = \underbrace{\begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix}}_{\vec{y}}.$$

This is the "triangular system"

$$\left.\begin{array}{rcl} 3x + 2z & = & 2 \\ -2y + z & = & 1 \\ -3z & = & 4 \end{array}\right\}. \tag{3}$$

Solving it by back substitution one obtains the solution:

$$\vec{x} = \begin{bmatrix} 2/15 \\ -1/10 \\ 4/5 \end{bmatrix}.$$

Summarizing: The LU decomposition of the coefficient matrix has allowed us to transform the initial system of equations into two systems that can be solved directly by forward and back substitution respectively: (2) and (3).

Notice that this method to solve linear systems is specially useful when one has to solve many square linear systems with the same coefficient matrix. The reason is that the LU decomposition is applied only over the coefficient matrix; this means that the same LU decomposition is valid for all the linear systems.

Example 6. If we want to solve the above system of linear equations using the LU decomposition and with Scilab, we can write the following:

```
-->C=[0 -2 1;3 0 2;0 2 4];[L,U]=lu(C)
 U  =
    3.    0.    2.
    0.  - 2.    1.
    0.    0.    5.
 L  =
    0.    1.    0.
    1.    0.    0.
    0.  - 1.    1.

-->y=L\[1;2;3]
 y  =
    2.
    1.
    4.

-->x=U\y
 x  =
    0.1333333
  - 0.1
    0.8
```

2.2.2 Application to the computation of determinants

The determinant of a square matrix can be computed by Scilab using the **det** function. For example:

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

    1.    2.
    3.    4.

-->det(D)
 ans  =

  - 2.
```

The procedure used internally by Scilab when this function is applied to a matriz A is the following one:

1. It is obtained the LU decomposition of A,

2. Taking into account that A = LU, the determinant of A is computed as $\det(\mathrm{L})\det(\mathrm{U})$. Notice that both involved determinants are very easy to compute:

   - Since U is upper triangular, its determinant is the product of the diagonal entries.
   - The determinant of L is ± 1, where the sign is $+$ when the number of swapping elementary operations is even, and $-$ when this number is odd[1].

```
-->A=[0 2 3; -4 6 0; 2 -5 5]
 A  =

    0.    2.    3.
  - 4.    6.    0.
    2.  - 5.    5.

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

  - 4.    6.    0.
    0.    2.    3.
    0.    0.    8.
 L  =

    0.     1.    0.
    1.     0.    0.
  - 0.5  - 1.    1.

-->det(L)
 ans  =

  - 1.

-->det(U)
 ans  =

  - 64.

-->det(L)*det(U)
 ans  =

    64.
```

[1]Scilab computes L is such a way that, after conveniently permuting the rows to obtain a lower triangular matrix, all the diagonal entries are equal to 1.

```
-->det(A)
 ans  =

    64.
```

2.2.3 Application to the computation of inverse matrices

When we use the **inv** function to compute the inverse of a square matrix A of order n, Scilab performs, actually, the following operations:

1. It obtains an LU decomposition of A.
2. It computes $\det(\mathsf{U})$.
3. If $\det(\mathsf{U}) = 0$ then Scilab indicates that the matrix is singular (non-invertible).
4. If $\det(\mathsf{U}) \neq 0$ then it computes U^{-1} in the following manner: If $\vec{u}_j$ denotes the jth column of U^{-1} and I is the identity matrix, it is clear that the equality $\mathsf{U}\mathsf{U}^{-1} = \mathsf{I}$ is equivalent to the set of equalities
$$\mathsf{U}\vec{u}_j = \vec{e}_j \quad j = 1, 2, \ldots n,$$
where $\vec{e}_j$ is the jth column vector of the identity matrix; therefore, solving all the linear systems $\mathsf{U}\vec{x} = \vec{e}_j$ one obtains all the columns of U^{-1}, that is, one obtains the whole inverse matrix of U. The inverse of the matrix L, L^{-1}, is computed following the same algorithm. Then A^{-1} is obtained as the product $\mathsf{U}^{-1}\mathsf{L}^{-1}$.

Notice that it is possible to obtain a singular matrix such that all the diagonal entries of U are non-zero (due to rounding errors in Gaussian elimination). In this case, the inverse matrix is computed by Scilab and, in some cases, it appears a warning message. By this reason we recommend to check if either the matrix obtained after entering inv(A) is actually the inverse matrix of A or A has maximum rank (using the **rank** function).

Example 7. Consider the matrix:

$$A = \begin{bmatrix} 1 & 2 & -5 \\ -4 & 1 & 0 \\ 1 & 2 & 7 \end{bmatrix}.$$

Let us compute its inverse from its LU decomposition and using the **inv** function:

```
-->det(L)*det(U)
 ans  =

    64.
```

```
-->A=[1 2 -5; -4 1 0; 1 2 7];

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

  - 4.    1.      0.
    0.    2.25  - 5.
    0.    0.      12.
 L  =

  - 0.25    1.    0.
    1.      0.    0.
  - 0.25    1.    1.

-->inv(U)*inv(L)
 ans  =

    0.0648148  - 0.2222222    0.0462963
    0.2592593    0.1111111    0.1851852
  - 0.0833333    0.           0.0833333

-->inv(A)
 ans  =

    0.0648148  - 0.2222222    0.0462963
    0.2592593    0.1111111    0.1851852
  - 0.0833333    0.           0.0833333
```

We check now that the obtained matrix is the inverse of A:

```
-->clean(A*ans)
 ans  =

    1.    0.    0.
    0.    1.    0.
    0.    0.    1.
```