

Properties of Determinants and Matlab

Math 45 — Linear Algebra

David Arnold

Fall 1997

Abstract. Matlab is used to investigate a number of important properties of the determinant. The effect of the elementary row operations on the determinant is examined and the determinants of triangular, identity, and permutation matrices are calculated. The LU decomposition is introduced as used to find the determinant of a matrix.

Prerequisites. Elementary row operations. Rudimentary understanding of the determinant.

Elementary Row Operations

The purpose of this section is to investigate how each of the elementary row operations affects the determinant. However, before we begin let's take a moment to see how Matlab generates matrices with random numbers.

It's easy to generate a matrix with random entries (*Note: In the exercises that follow, because the matrices are generated at random, your results may differ. In fact, it almost 100% certain that you will generate matrices different from those given in the examples.*)

```
>> A=rand(3,5)
```

A =

0.1564	0.7218	0.6632	0.4194	0.0812
0.1221	0.6516	0.8835	0.2130	0.8506
0.7627	0.7540	0.2722	0.0356	0.3402

Note that the command `rand(3,5)` generates a 3×5 matrix with random numbers. Note that each random number is between 0 and 1. The `rand` command selects numbers from a *uniform distribution*, so named because each random number between 0 and 1 has an equal probability of being selected. Technically, any number between 0 and 1, including 0, has an equal chance of being selected, but the number 1 has no chance of being selected. Therefore, each random number r selected by the `rand` command satisfies the inequality $0 \leq r < 1$.

Suppose that you multiply each element of the matrix A by 10 to form the matrix B .

```
>> B=10*A
```

B =

1.5638	7.2180	6.6316	4.1943	0.8116
1.2212	6.5164	8.8349	2.1299	8.5057
7.6267	7.5402	2.7216	0.3560	3.4020

Note that each entry of the matrix B could be floating point number between 0 and 10, possibly including 0 but not 10. That is, each entry b in the matrix B satisfies the inequality $0 \leq b < 10$.

Subtract 5 from each element of the matrix B to form the matrix C .

```
>> C=B-5
```

C =

-3.4362	2.2180	1.6316	-0.8057	-4.1884
-3.7788	1.5164	3.8349	-2.8701	3.5057
2.6267	2.5402	-2.2784	-4.6440	-1.5980

Note that each entry in the matrix C could be a floating point number between -5 and 5 , possibly including -5 but not 5 .

There is an easy way to track the size of the random entries in the matrices A , B , and C . Remember, each element r of matrix A satisfied the inequality $0 \leq r < 1$. Then we multiplied by 10 and subtracted 5 , as follows:

$$\begin{aligned} 0 &\leq r < 1 \\ 0 &\leq 10r < 10 \\ -5 &\leq 10r - 5 < 5 \end{aligned}$$

Consequently, the elements of matrix C can be random floating point numbers between -5 and 5 , possibly including -5 but not 5 .

Floor, Ceil, Fix, and Round

`floor(X)` rounds the elements of X to the nearest integers towards minus infinity.

```
>> C, floor(C)
```

```
C =
```

```
-3.4362    2.2180    1.6316   -0.8057   -4.1884
-3.7788    1.5164    3.8349   -2.8701    3.5057
 2.6267    2.5402   -2.2784   -4.6440   -1.5980
```

```
ans =
```

```
-4     2     1    -1    -5
-4     1     3    -3     3
 2     2    -3    -5    -2
```

Note that each entry in C is rounded “down” to the nearest integer.

`ceil(X)` rounds the elements of X to the nearest integers towards positive infinity.

```
>> C, ceil(C)
```

```
C =
```

```
-3.4362    2.2180    1.6316   -0.8057   -4.1884
-3.7788    1.5164    3.8349   -2.8701    3.5057
 2.6267    2.5402   -2.2784   -4.6440   -1.5980
```

```
ans =
```

```
-3     3     2     0    -4
-3     2     4    -2     4
 3     3    -2    -4    -1
```

Note that each entry in C is rounded “up” to the nearest integer.

`round(X)` rounds the elements of X to the nearest integers.

```
>> C, round(C)
```

```
C =
```

```
-3.4362    2.2180    1.6316   -0.8057   -4.1884
-3.7788    1.5164    3.8349   -2.8701    3.5057
 2.6267    2.5402   -2.2784   -4.6440   -1.5980
```

```
ans =
```

```
-3    2    2   -1   -4
-4    2    4   -3    4
 3    3   -2   -5   -2
```

Note that each entry in C is rounded to the “closest” integer.

Finally, `fix(X)` rounds the elements of X to the nearest integers towards zero.

```
>> C, fix(C)
```

```
C =
```

```
-3.4362    2.2180    1.6316   -0.8057   -4.1884
-3.7788    1.5164    3.8349   -2.8701    3.5057
 2.6267    2.5402   -2.2784   -4.6440   -1.5980
```

```
ans =
```

```
-3    2    1    0   -4
-3    1    3   -2    3
 2    2   -2   -4   -1
```

Note that negative numbers are rounded “up” and positive numbers are rounded “down.”

Multiplying a Row by a Scalar

Create a 3×3 matrix A whose entries are random integers between -5 and 5 . Then take the determinant of A .

```
>> A=round(10*rand(3)-5)
```

```
A =
```

```
 0    4    0
 4    2    5
-3    4   -1
```

```
>> det(A)
```

```
ans =
```

```
-44
```

Replace the first row of A with 5 times the first row of A . Then take the determinant of the result.

```
>> A(1,:)=5*A(1,:)
```

```
A =
```

```

0    20    0
4     2    5
-3    4   -1

```

```
>> det(A)
```

```
ans =
```

```
-220
```

Repeat this process several times by replaying the following instruction.

```
>> A=round(10*rand(3)-5),det(A),A(1,:)=5*A(1,:),det(A)
```

In the space that follows, explain what happens to the determinant of a matrix if you multiply its first row by 5.

Conjecture.

How would your conjecture change if you multiplied (Try

`A=round(10*rand(3)-5),det(A),A(:,3)=5*A(:,3),det(A)`) the third column by 5?

Exchanging Two Rows

Create a 4×4 matrix A whose entries are random integers between -10 and 10 . Then take the determinant of A .

```
>> A=round(20*rand(4)-10)
```

```
A =
```

```

-7    -9     5   -10
-10    7     1     8
-5    -6    -4     7
-8    -9    -3    -5

```

```
>> det(A)
```

```
ans =
```

```
-10638
```

Exchange rows 1 and 3 and take the determinant of the result.

```
>> A=A([3,2,1,4],:)
```

```
A =
```

```

-5    -6    -4     7
-10   7     1     8
-7    -9     5   -10

```

```

-8    -9    -3    -5
>> det(A)
ans =
    10638

```

Repeat this process several times by replaying the following instruction.

```
>> A=round(20*rand(4)-10),det(A),A=A([3 2 1 4],:),det(A)
```

In the space that follows, explain what happens to the determinant of a matrix if you exchange two rows.

Conjecture.

How would your conjecture change if you exchanged two *columns* (Try

```
A=round(20*rand(4)-10),det(A),A=A(:,[3 2 1 4]),det(A))?
```

Replace a Row with the Sum of Itself and a Scalar Multiple of Another Row

Enter the following matrix A . Then take the determinant of A .

```
>> A=[1 2 0;-2 1 3;4 1 1]
A =
     1     2     0
    -2     1     3
     4     1     1
>> det(A)
ans =
    26

```

Replace the second row of A with the sum of itself and 2 times the first row of A . Then take the determinant of the result.

```
>> A(2,:)=A(2,:)+2*A(1,:)
A =
     1     2     0
     0     5     3
     4     1     1
>> det(A)
```

```
ans =
```

```
26
```

The determinant is still 26. Let's replace the third row of A with the sum of itself and -4 times the first row of A .

```
>> A(3,:) = A(3,:) - 4*A(1,:)
```

```
A =
```

```
1     2     0
0     5     3
0    -7     1
```

```
>> det(A)
```

```
ans =
```

```
26
```

The determinant is still 26. Are we just lucky? Let's replace the third row of A with the sum of itself and $7/5$ times the second row of A .

```
>> A(3,:) = A(3,:) + 7/5*A(2,:)
```

```
A =
```

```
1.0000    2.0000         0
         0    5.0000    3.0000
         0         0    5.2000
```

```
>> det(A)
```

```
ans =
```

```
26.0000
```

This can't be luck! In the space provided below, explain what happens to the determinant of a matrix if you replace a row with the sum of itself and a scalar multiple of another row.

Conjecture.

What would happen to the determinant if you replaced a column with the sum of itself and a scalar multiple of another column (Try $A = \begin{bmatrix} 1 & -2 & 0 \\ 3 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix}$, $\det(A)$, $A(:,2) = A(:,2) + 2*A(:,1)$, $\det(A)$)?

Triangular Matrices

Upper Triangular Matrices. For an example of an upper triangular matrix, try

```
>> U = triu(round(2*rand(5)+1))
```

```
U =
     2     1     3     2     3
     0     2     1     1     2
     0     0     2     2     2
     0     0     0     2     1
     0     0     0     0     1
```

```
>> det(U)
```

```
ans =
```

```
    16
```

Each entry of an upper triangular matrix that falls below the main diagonal is zero. Repeat this experiment by replaying the following command several times.

```
>> U=triu(round(2*rand(5)+1)),det(U)
```

In the space that follows, explain how to find the determinant of an upper triangular matrix.

Conjecture.

Lower Triangular Matrices. For an example of a lower triangular matrix, try

```
>> L=tril(round(2*rand(5)+1)),det(L)
```

```
L =
```

```
     2     0     0     0     0
     1     1     0     0     0
     1     3     1     0     0
     1     2     2     3     0
     1     2     1     1     1
```

```
>> det(L)
```

```
ans =
```

```
     6
```

Each entry of a lower triangular matrix that lies above the main diagonal is zero. Repeat this experiment by replaying the following command several times.

```
>> L=tril(round(2*rand(5)+1)),det(L)
```

In the space that follows, explain how to find the determinant of a lower triangular matrix.

Conjecture.

Identity Matrices

Let's investigate the determinant of identity matrices.

```
>> I=eye(5)
```

```
I =
```

```

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1

```

```
>> det(I)
```

```
ans =
```

```

     1

```

Repeat this experiment by replaying the following command for several different values of n .

```
>> n=3, I=eye(n), det(I)
```

In the space that follows explain how to find the determinant of an $n \times n$ identity matrix. Also, explain how you can use your conjecture about upper and lower triangular matrices to calculate the determinant of an $n \times n$ identity matrix.

Conjecture.

Permutation Matrices

A permutation is a reordering of numbers. For example, repeat the following command several times to capture random permutations of the numbers 1 through 10.

```
>> p=randperm(10)
```

A square matrix with a single 1 in every row and every column is called a *permutation matrix*. Permutation matrices are built by permuting the rows (or columns) of the identity matrix.

```
>> I=eye(5)
```

```
I =
    1     0     0     0     0
    0     1     0     0     0
    0     0     1     0     0
    0     0     0     1     0
    0     0     0     0     1
```

```
>> P=I([1 4 3 5 2],:)
```

```
P =
    1     0     0     0     0
    0     0     0     1     0
    0     0     1     0     0
    0     0     0     0     1
    0     1     0     0     0
```

Note that the rows of P are a rearrangement of the rows of I . Each time you exchange a pair of rows you negate the determinant. Start with the matrix I and exchange rows 2 and 5. Then exchange rows 4 and 5 of the result and you get the matrix P . The determinant of the identity matrix I is 1 and two row exchanges negates the determinant twice. Consequently, $\det(P) = \det(I) = 1$.

```
>> det(P)
```

```
ans =
```

```
1
```

Repeat the command

```
>> I=eye(6),P=I(randperm(6),:),det(P)
```

several times then make a conjecture about the determinant of a permutation matrix.

Conjecture.

Determinants of Sums and Products

Generate two random 3×3 matrices with integer entries between -2 and 2 . Then find the sum of their determinants.

```
>> A=round(4*rand(3)-2),B=round(4*rand(3)-2),det(A)+det(B)
```

```
A =
```

```
    0    -1     1
   -2     1     2
```

```
0    -2    2
```

```
B =
```

```
1    -1    2
0     1    1
0    -1    0
```

```
ans =
```

```
1
```

Sum the matrices A and B and take the determinant of the result.

```
>> det(A+B)
```

```
ans =
```

```
23
```

Repeat this experiment several times by replaying the following instruction.

```
>> A=round(4*rand(3)-2);B=round(4*rand(3)-2);det(A)+det(B),det(A+B)
```

Switch to *multiplication*. Replay the following instruction several times.

```
>> A=round(4*rand(3)-2);B=round(4*rand(3)-2);det(A)*det(B),det(A*B)
```

Make a conjecture based on the last two experiments.

Conjecture.

LU Factorization

There are a number of techniques in linear algebra designed to aid in expressing a matrix as a product of two or more matrices. One of the more important factorizations is called *LU decomposition*.

Without Row Exchanges

Matlab is very good at performing LU decompositions.. If row exchanges are not needed, then the command `[L,U]=lu(A)` will find a lower triangular matrix L and an upper triangular matrix U so that $A = LU$.

```
>> format rat
>> A=[2 1 -1;0 1 -1;1 0 1]
```

A =

```

      2      1      -1
      0      1      -1
      1      0       1

```

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

L =

```

      1      0      0
      0      1      0
     1/2    -1/2     1

```

U =

```

      2      1      -1
      0      1      -1
      0      0       1

```

Note that L is lower triangular (with 1's on its main diagonal) and U is upper triangular. Check to see that $A = LU$.

>> A,L*U

A =

```

      2      1      -1
      0      1      -1
      1      0       1

```

ans =

```

      2      1      -1
      0      1      -1
      1      0       1

```

Note that A and LU are equal.

Finally, because L and U are triangular matrices, their determinants are computed by taking the product of their diagonal elements.

$$\det(L) = \det \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/2 & -1/2 & 1 \end{bmatrix} = (1)(1)(1) = 1$$

$$\det(U) = \det \begin{bmatrix} 2 & 1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} = (2)(1)(1) = 2$$

Use $A = LU$ to Calculate $\det(A)$. Hopefully, you conjectured earlier that $\det(AB) = \det(A)\det(B)$. That is, the determinant of a product of two matrices is the product of the determinants. Once you have the LU decomposition of matrix A you can use this multiplicative property of determinants to your advantage.

$$\begin{aligned}
 A &= LU \\
 \det(A) &= \det(LU) \\
 \det(A) &= \det(L)\det(U) \\
 \det(A) &= (1)(2) \\
 \det(A) &= 2
 \end{aligned}$$

With Row Exchanges

If Matlab has to make row interchanges to perform the LU decomposition then it returns matrices L , U , and P so that $AP = LU$. The matrix P is a permutation matrix that reveals the row interchanges.

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

```
A =
```

```

     1     1     0
     2    -2     1
     0     1     5

```

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

```
L =
```

```

     1     0     0
    1/2     1     0
     0    1/2     1

```

```
U =
```

```

     2     -2     1
     0     2    -1/2
     0     0    21/4

```

```
P =
```

```

     0     1     0
     1     0     0
     0     0     1

```

Note again that the matrix L has 1's on its main diagonal. Check to see that $PA = LU$.

```
>> P*A,L*U
```

```
ans =
```

```

     2     -2     1
     1     1     0
     0     1     5

```

```
ans =
```

```

     2     -2     1
     1     1     0

```

0 1 5

Also, note that exchanging the first and second rows of the identity matrix produces the matrix P . One exchange of rows negates the determinant. Therefore, $\det(P) = -\det(I) = -1$.

```
>> det(P)
ans =
    -1
```

Use $PA = LU$ to Calculate $\det(A)$.

$$\begin{aligned}
 PA &= LU \\
 \det(PA) &= \det(LU) \\
 \det(P)\det(A) &= \det(L)\det(U) \\
 (-1)\det(A) &= [(1)(1)(1)][(2)(2)(21/4)] \\
 \det(A) &= -21
 \end{aligned}$$

Of course, you should always check your work.

```
>> det(A)
ans =
   -21
```

What's the Big Deal About LU Decomposition?

You will see a number of other important matrix factorizations in your linear algebra course. All of these factorizations have important applications. Let's finish this activity by hinting at the power and speed of the LU factorization in Matlab.

You might think that it is more expensive, in terms of computer time, to find three matrices so that $PA = LU$ than it is to find the reduced row echelon form of a matrix. Let's do a little experiment.

```
>> A=round(10*rand(50)-5);
>> flops(0);
>> rref(A);
>> flops
ans =
  189033
```

The Matlab command `flops` counts the cumulative number of floating operations to date. By setting the `flops` to zero, executing your routine, then making a final call to `flops`, you can find the number of floating point operations (Each floating point operation requires a certain number of cycles of your computer's CPU. Type `help flops` to get a short description of how Matlab counts floating point operations) required to perform your routine.

For comparison, let's count the number of floating point operations required to perform an

LU decomposition of the matrix A .

```
>> A=round(10*rand(50)-5);  
>> flops(0);  
>> [L,U,P]=lu(A);  
>> flops
```

ans =

82075

You can also record the amount of time required for Matlab to execute an instruction (Type `help tic` to get a description of how these Matlab commands work). You might want to use your wristwatch to time the execution of the following commands (elapsed time will vary with different processor speeds).

```
>> A=10*rand(50)-5;  
>> tic,rref(A);toc
```

elapsed_time =

32.1900

```
>> A=10*rand(50)-5;  
>> tic,[L,U,P]=lu(A);toc
```

elapsed_time =

0.1100

As you can see, Matlab's `lu` command is quick and efficient. Consequently, Matlab uses this powerful command in a number of its routines.