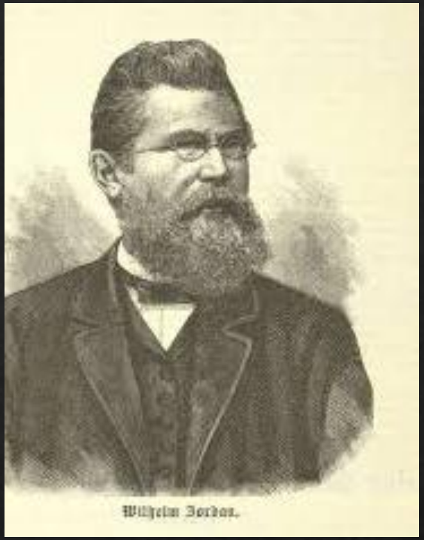
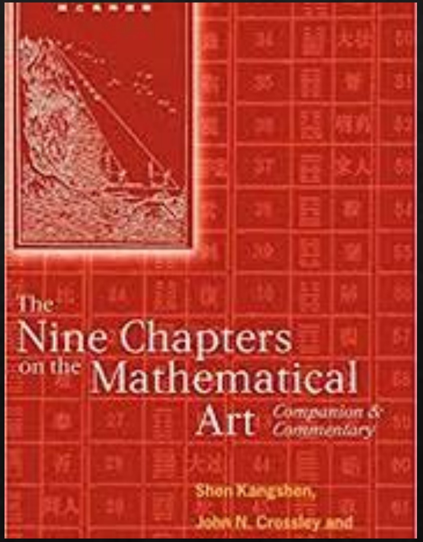
Row Reduction Algorithm

Christina Baris, Nicole Evans, Jacob Fyda, Shayne Miller





HISTORY

- First recorded use in *Chapter Eight: Rectangular Arrays* of T*he Nine Chapters on Mathematical Art* in 179 CE in China.
- Isaac Newton – 1707
- Carl Friedrich Gauss – 1810
- Wilheilm Jordan – 1888

Row Reduction Proof

Let A be an m x n augmented matrix representing a system of linear equations.

Let rows $R_1, \dots, R_m$ be rows of matrix A.

A has elements a_{11} to a_{mn}.

Proof: Three elementary row operations do not change the solution set of the system represented by the matrix, and, consequently, can be used to solve systems of linear equations.

Row Reduction Proof

- Swapping two rows

Swapping two rows does not change the solution set of the system because none of the equations, variables, or coefficients are altered in any way.

$$\begin{aligned} a_1x_1 + \dots + a_nx_n &= c \\ b_1x_1 + \dots + b_nx_n &= d \end{aligned} \qquad \begin{aligned} b_1x_1 + \dots + b_nx_n &= d \\ a_1x_1 + \dots + a_nx_n &= c \end{aligned}$$

Both of these have the same solution set even with the order in which they are written swapped. This is generalizable to any number of equations and swaps.

Row Reduction Proof

- Scaling a row

Scaling a row does not change the solution set of the system because multiplying two sides of an equation by the same value does not change the solution of that equation. Each row of A represents an equation, so scaling rows does not alter the solution of the augmented matrix.

Row Reduction Proof

- Adding a multiple of one row to another row

This operation does not change the solution set because it is reversible, same as the other two elementary row operations. Any addition of rows results in a row that is a consequence of a different row. In order to return to the original row, the opposite operation can be applied to the consequence and the row will return to its original form. This does not change the solution because the operations can occur in both directions.

Let *I* and *II* be two rows of matrix A.

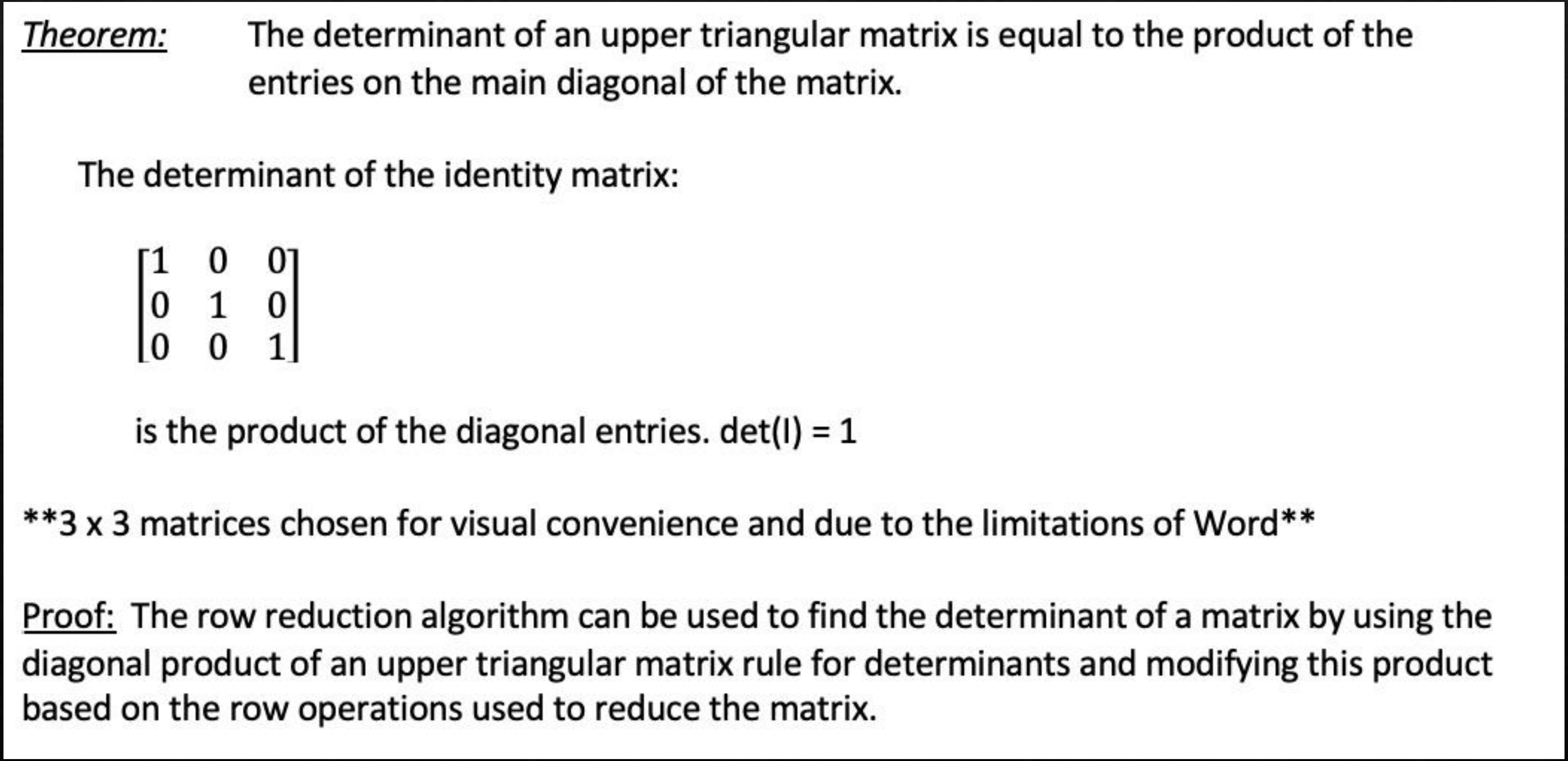
$a \bullet I + b \bullet II = II'$, where II' is a consequence of *II* and is equivalent to *II,*

and *a* and *b* are real numbers

In order to return II' to *II*, simply subtract the row that was added to *II.*

$II' - a \bullet I = II$

Determinant Proof



Theorem: The determinant of an upper triangular matrix is equal to the product of the entries on the main diagonal of the matrix.

The determinant of the identity matrix:

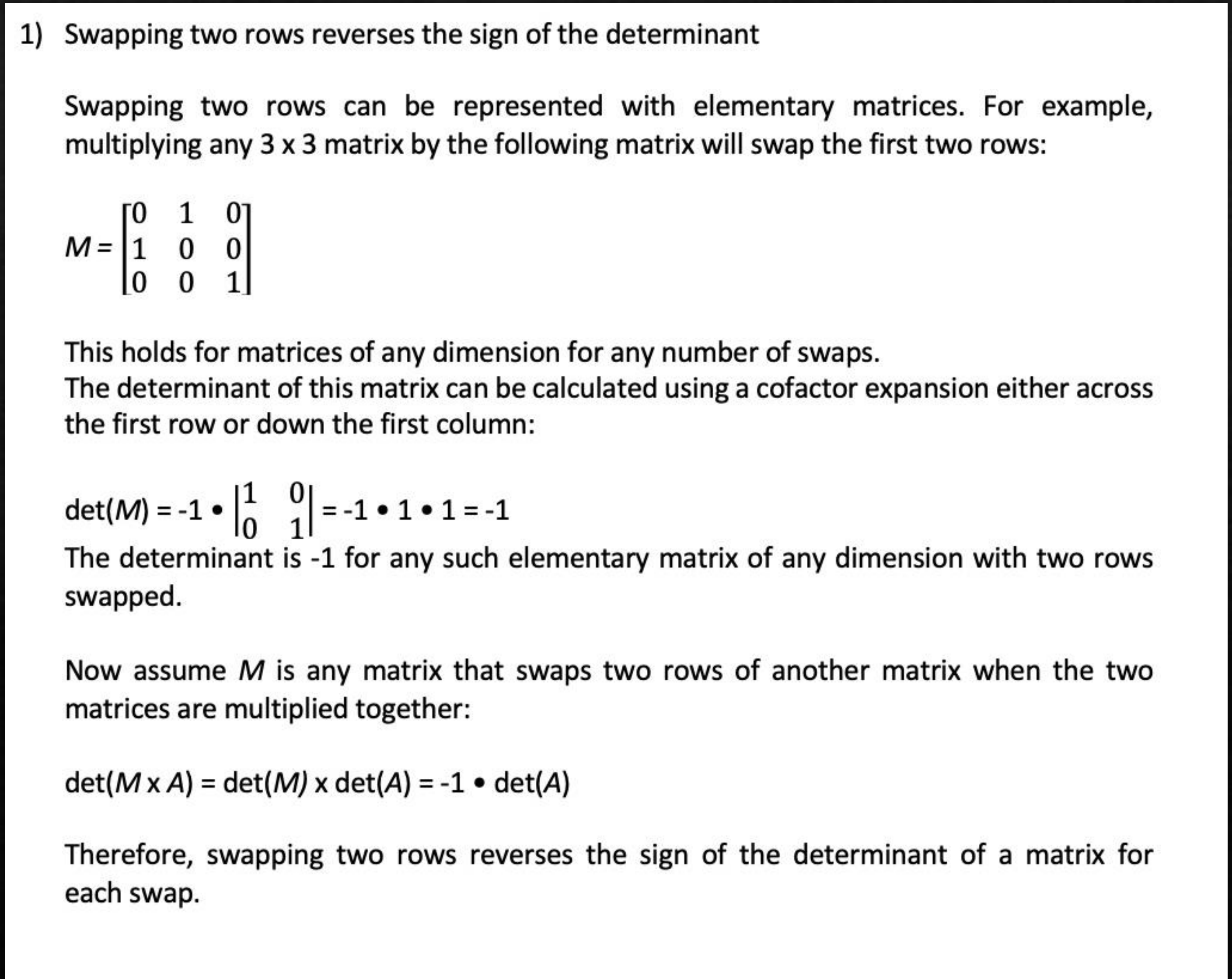
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

is the product of the diagonal entries. det(I) = 1

3 x 3 matrices chosen for visual convenience and due to the limitations of Word

Proof: The row reduction algorithm can be used to find the determinant of a matrix by using the diagonal product of an upper triangular matrix rule for determinants and modifying this product based on the row operations used to reduce the matrix.

Determinant Proof



1) Swapping two rows reverses the sign of the determinant

Swapping two rows can be represented with elementary matrices. For example, multiplying any 3 x 3 matrix by the following matrix will swap the first two rows:

$$M = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This holds for matrices of any dimension for any number of swaps.
The determinant of this matrix can be calculated using a cofactor expansion either across the first row or down the first column:

$$\det(M) = -1 \bullet \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = -1 \bullet 1 \bullet 1 = -1$$

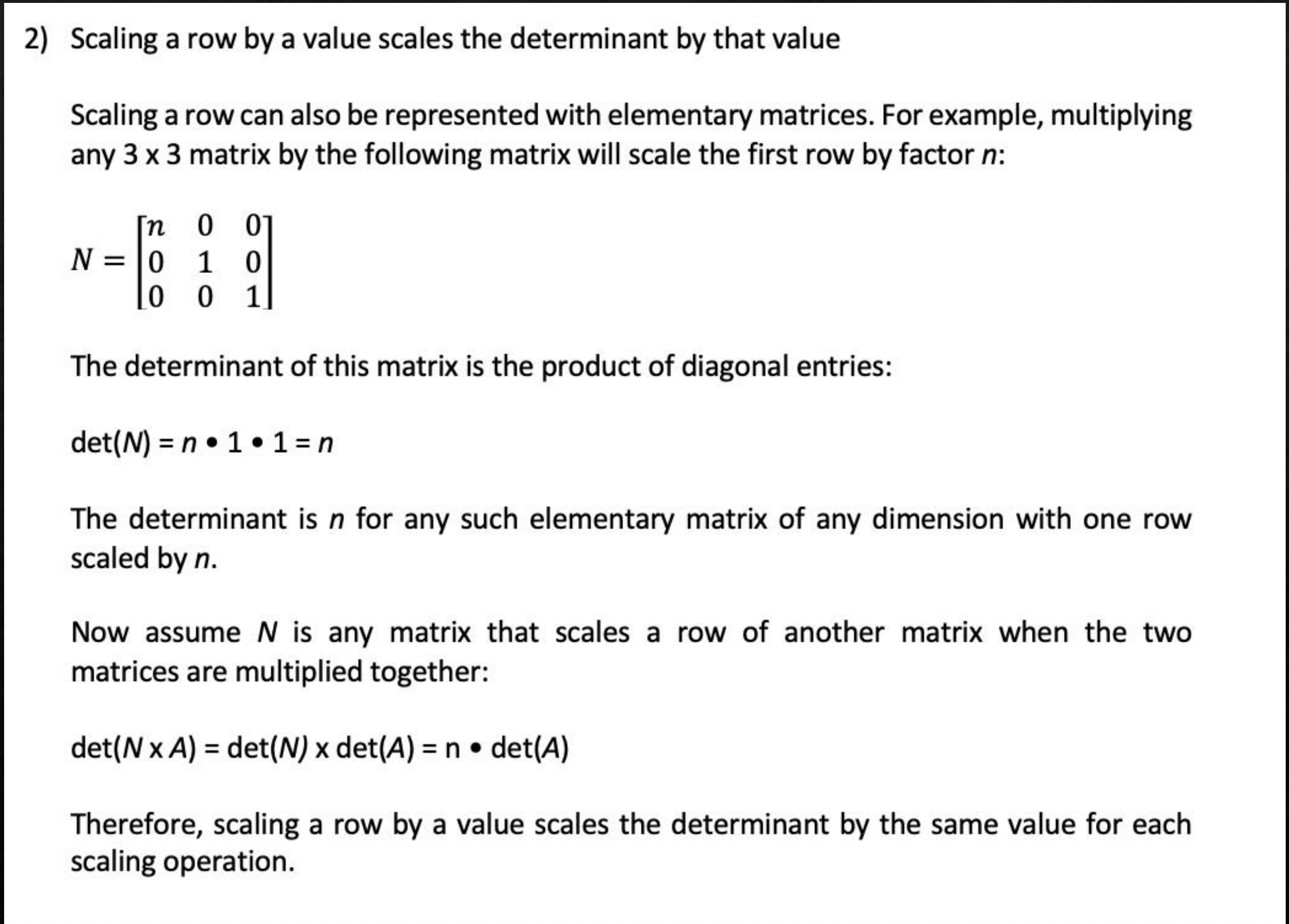
The determinant is -1 for any such elementary matrix of any dimension with two rows swapped.

Now assume M is any matrix that swaps two rows of another matrix when the two matrices are multiplied together:

$$\det(M \times A) = \det(M) \times \det(A) = -1 \bullet \det(A)$$

Therefore, swapping two rows reverses the sign of the determinant of a matrix for each swap.

Determinant Proof



2) Scaling a row by a value scales the determinant by that value

Scaling a row can also be represented with elementary matrices. For example, multiplying any 3 x 3 matrix by the following matrix will scale the first row by factor *n*:

$$N = \begin{bmatrix} n & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The determinant of this matrix is the product of diagonal entries:

$\det(N) = n \bullet 1 \bullet 1 = n$

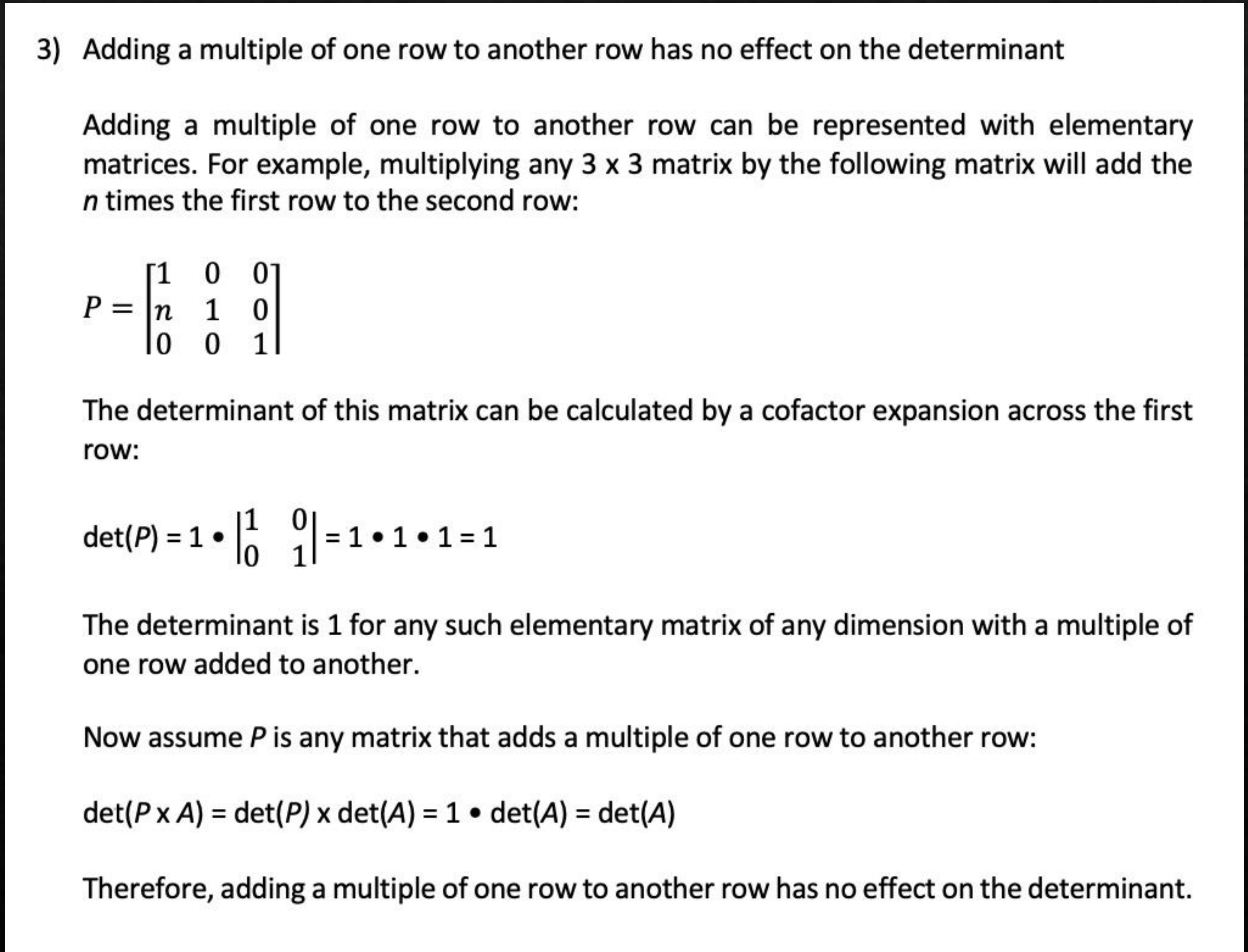
The determinant is *n* for any such elementary matrix of any dimension with one row scaled by *n*.

Now assume *N* is any matrix that scales a row of another matrix when the two matrices are multiplied together:

$\det(N \times A) = \det(N) \times \det(A) = \mathrm{n} \bullet \det(A)$

Therefore, scaling a row by a value scales the determinant by the same value for each scaling operation.

Determinant Proof



3) Adding a multiple of one row to another row has no effect on the determinant

Adding a multiple of one row to another row can be represented with elementary matrices. For example, multiplying any 3 x 3 matrix by the following matrix will add the *n* times the first row to the second row:

$$P = \begin{bmatrix} 1 & 0 & 0 \\ n & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The determinant of this matrix can be calculated by a cofactor expansion across the first row:

$$\det(P) = 1 \bullet \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 1 \bullet 1 \bullet 1 = 1$$

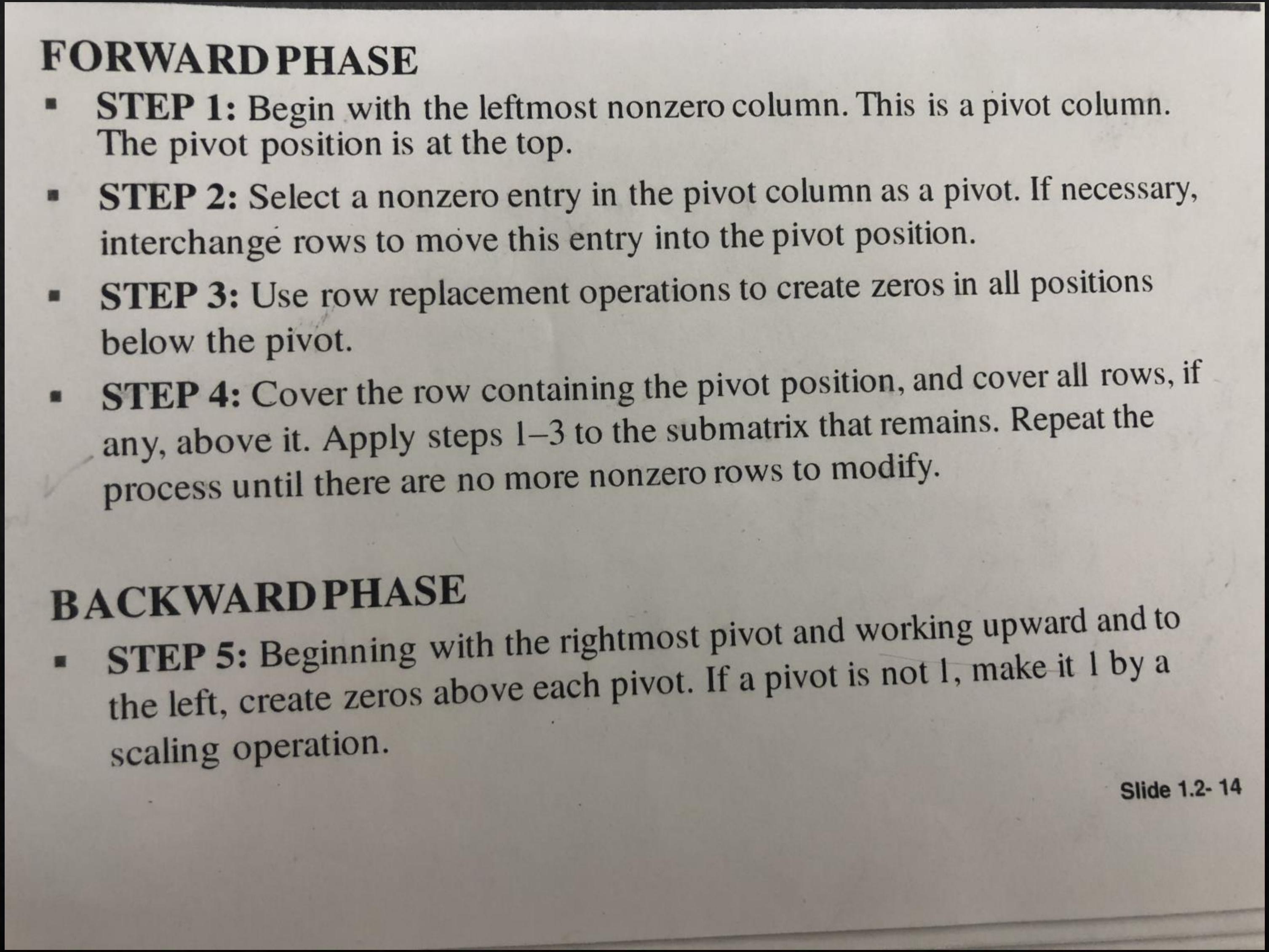
The determinant is 1 for any such elementary matrix of any dimension with a multiple of one row added to another.

Now assume *P* is any matrix that adds a multiple of one row to another row:

$$\det(P \times A) = \det(P) \times \det(A) = 1 \bullet \det(A) = \det(A)$$

Therefore, adding a multiple of one row to another row has no effect on the determinant.

The Row Reduction Algorithm



FORWARD PHASE

- **STEP 1:** Begin with the leftmost nonzero column. This is a pivot column. The pivot position is at the top.
- **STEP 2:** Select a nonzero entry in the pivot column as a pivot. If necessary, interchange rows to move this entry into the pivot position.
- **STEP 3:** Use row replacement operations to create zeros in all positions below the pivot.
- **STEP 4:** Cover the row containing the pivot position, and cover all rows, if any, above it. Apply steps 1–3 to the submatrix that remains. Repeat the process until there are no more nonzero rows to modify.

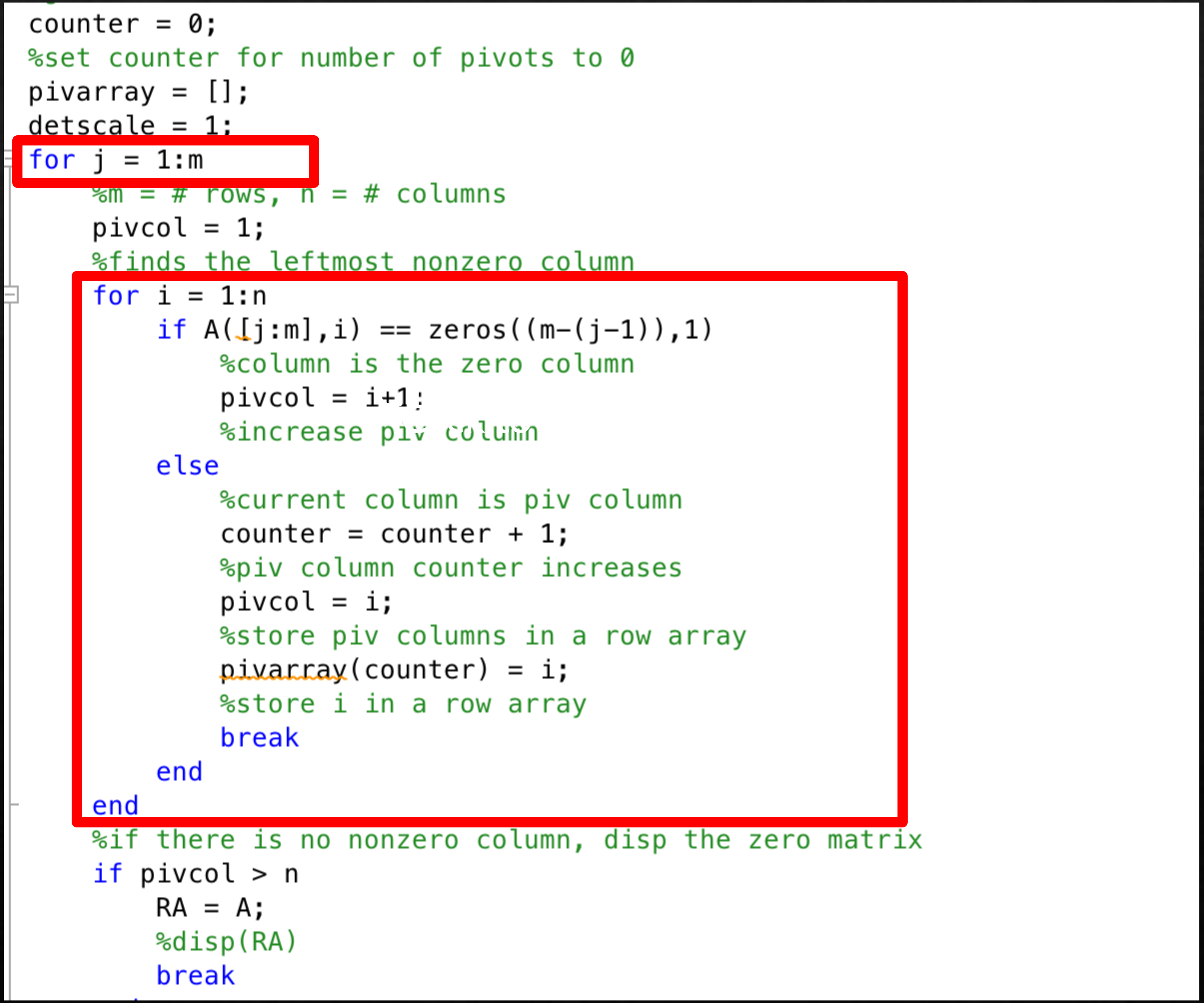
BACKWARD PHASE

- **STEP 5:** Beginning with the rightmost pivot and working upward and to the left, create zeros above each pivot. If a pivot is not 1, make it 1 by a scaling operation.

Row Reduction Algorithm Program

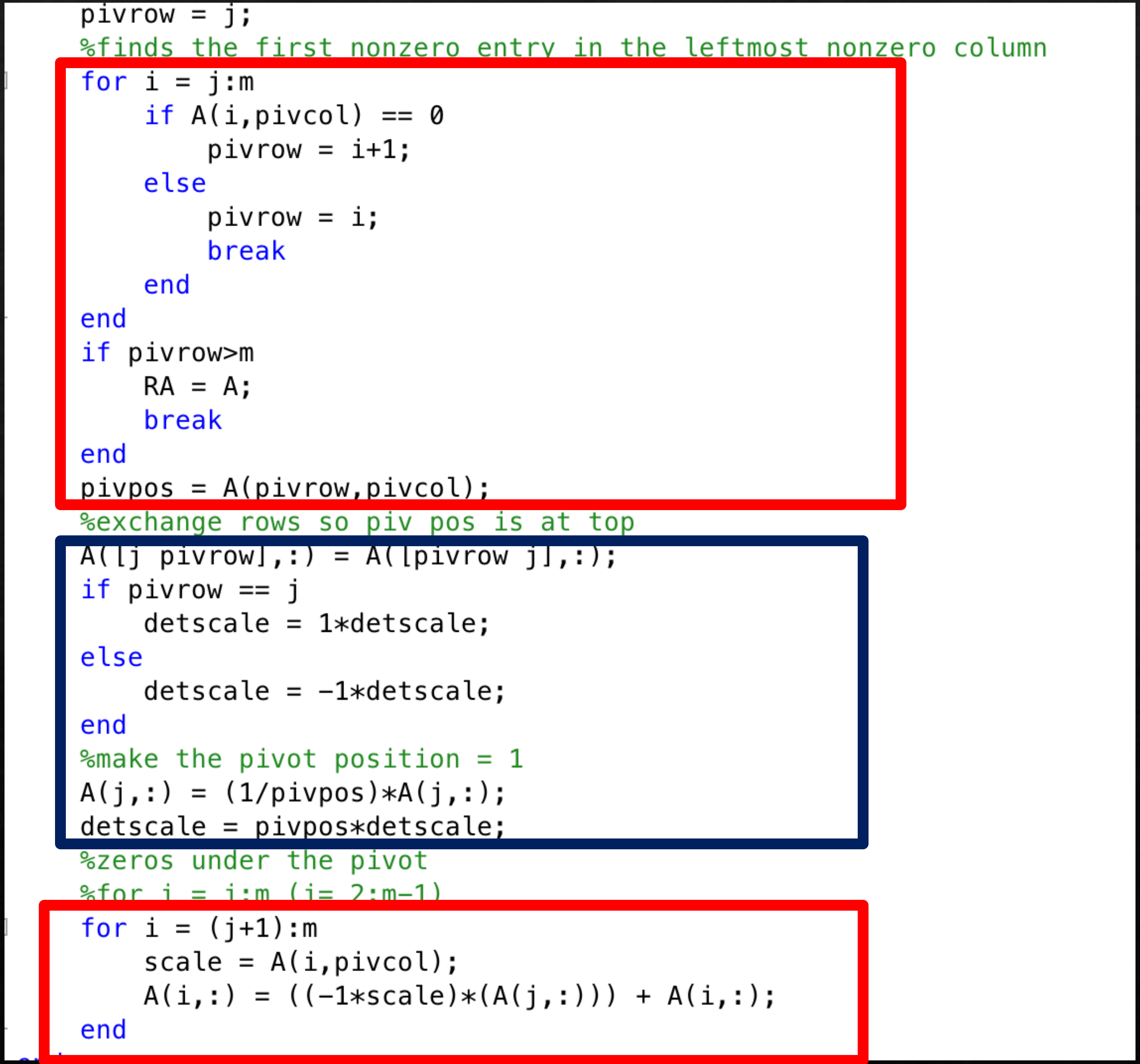
4

1



```
counter = 0;
%set counter for number of pivots to 0
pivarray = [];
detscale = 1;
for j = 1:m
    %m = # rows, n = # columns
    pivcol = 1;
    %finds the leftmost nonzero column
    for i = 1:n
        if A([j:m],i) == zeros((m-(j-1)),1)
            %column is the zero column
            pivcol = i+1;
            %increase piv column
        else
            %current column is piv column
            counter = counter + 1;
            %piv column counter increases
            pivcol = i;
            %store piv columns in a row array
            pivarray(counter) = i;
            %store i in a row array
            break
        end
    end
    %if there is no nonzero column, disp the zero matrix
    if pivcol > n
        RA = A;
        %disp(RA)
        break
```

Row Reduction Algorithm Program



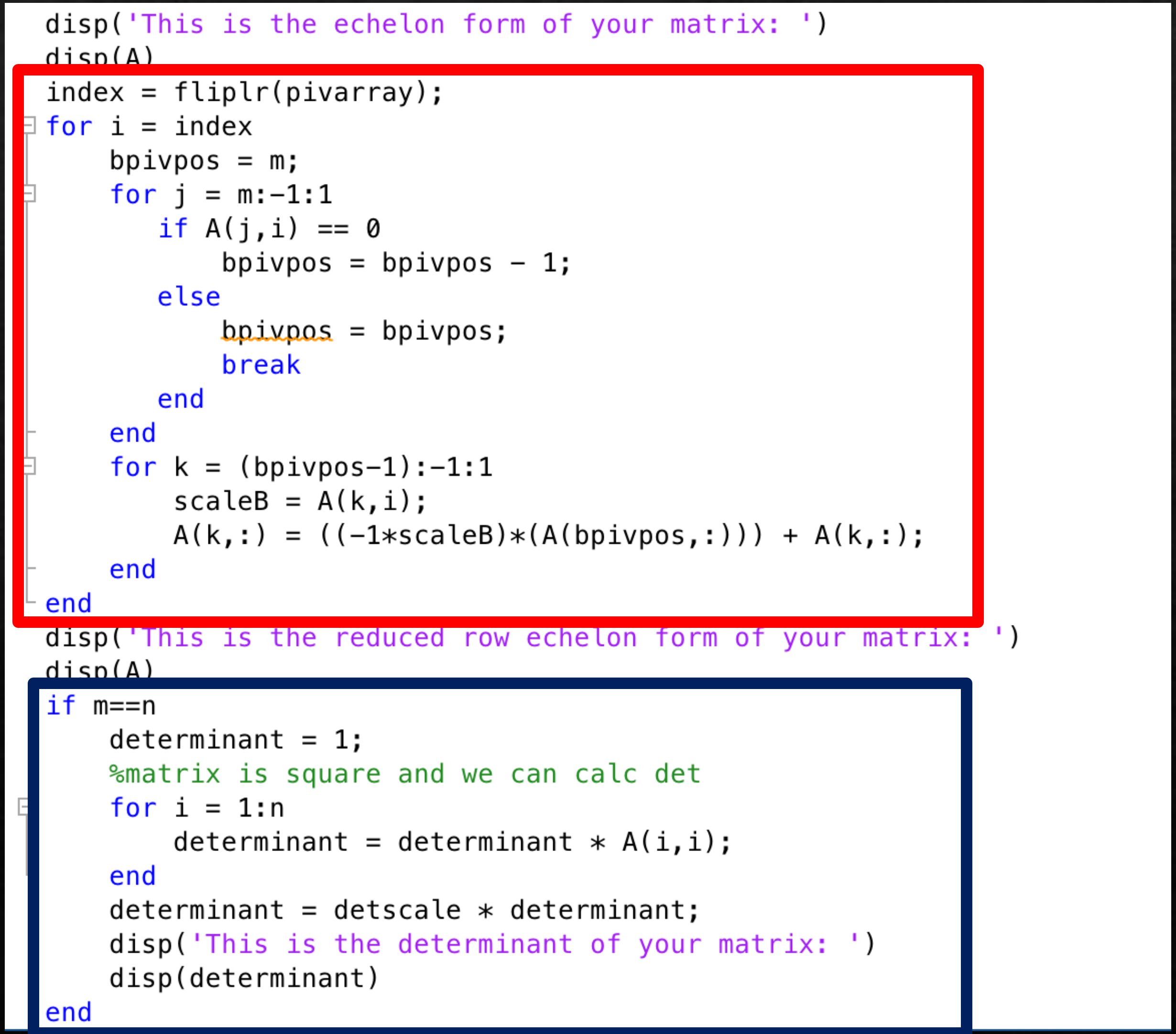
```
pivrow = j;
%finds the first nonzero entry in the leftmost nonzero column
for i = j:m
    if A(i,pivcol) == 0
        pivrow = i+1;
    else
        pivrow = i;
        break
    end
end
if pivrow>m
    RA = A;
    break
end
pivpos = A(pivrow,pivcol);
%exchange rows so piv pos is at top
A([j pivrow],:) = A([pivrow j],:);
if pivrow == j
    detscale = 1*detscale;
else
    detscale = -1*detscale;
end
%make the pivot position = 1
A(j,:) = (1/pivpos)*A(j,:);
detscale = pivpos*detscale;
%zeros under the pivot
%for i = j:m (j= 2:m-1)
for i = (j+1):m
    scale = A(i,pivcol);
    A(i,:) = ((-1*scale)*(A(j,:))) + A(i,:);
end
```

2

3

Row Reduction Algorithm Program

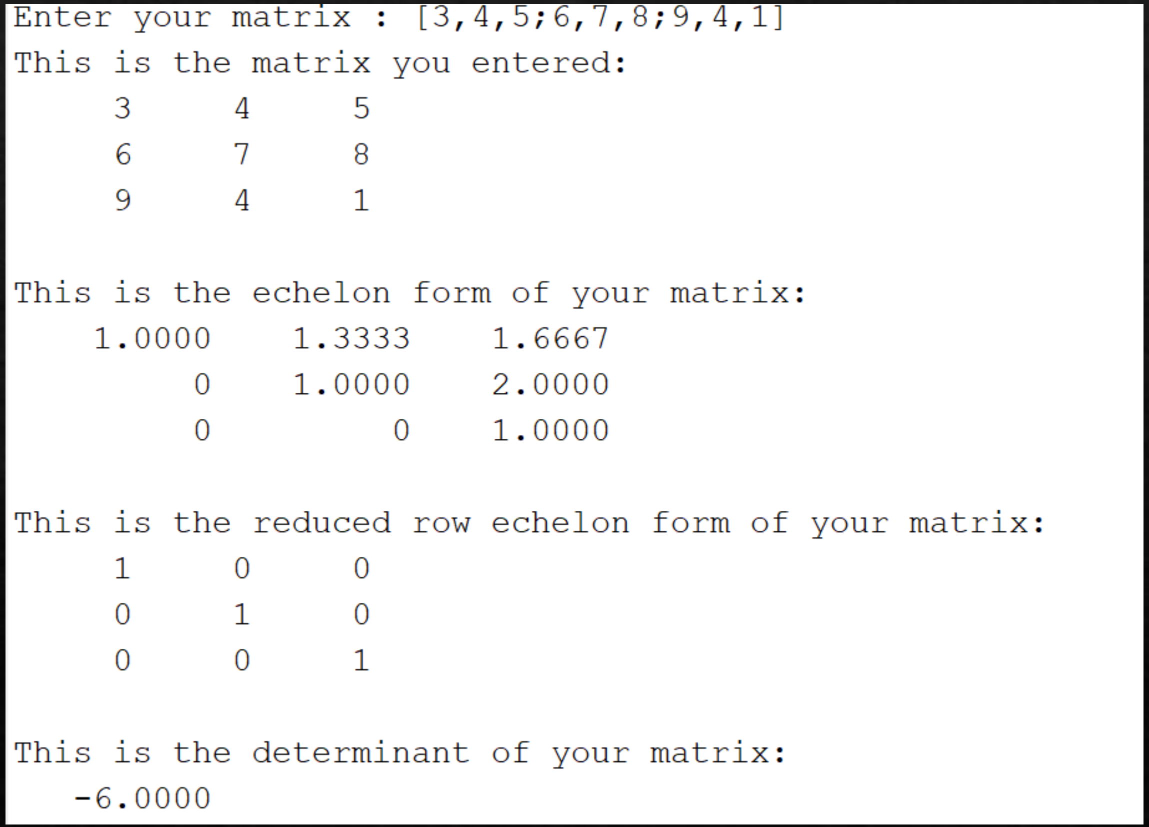
5



```
disp('This is the echelon form of your matrix: ')
disp(A)
index = fliplr(pivarray);
for i = index
    bpivpos = m;
    for j = m:-1:1
       if A(j,i) == 0
           bpivpos = bpivpos - 1;
       else
           bpivpos = bpivpos;
           break
       end
    end
    for k = (bpivpos-1):-1:1
        scaleB = A(k,i);
        A(k,:) = ((-1*scaleB)*(A(bpivpos,:))) + A(k,:);
    end
end
disp('This is the reduced row echelon form of your matrix: ')
disp(A)
if m==n
    determinant = 1;
    %matrix is square and we can calc det
    for i = 1:n
        determinant = determinant * A(i,i);
    end
    determinant = detscale * determinant;
    disp('This is the determinant of your matrix: ')
    disp(determinant)
end
```

Row Reduction Algorithm Program

Execution of code



```
Enter your matrix : [3,4,5;6,7,8;9,4,1]
This is the matrix you entered:
     3     4     5
     6     7     8
     9     4     1

This is the echelon form of your matrix:
    1.0000    1.3333    1.6667
         0    1.0000    2.0000
         0         0    1.0000

This is the reduced row echelon form of your matrix:
     1     0     0
     0     1     0
     0     0     1

This is the determinant of your matrix:
   -6.0000
```

Thanks for Watching!