UNIVERSITY OF MILA

Faculty of Science and Technology

Department of Process Engineering

Level: 1st year ST - ENG & LMD

Course : Introduction to Programming

Lecture 09:

Advanced Matrix Operations: Adjoint and Inverse

---

by:

Dr. Farouk KECITA

Academic Year 2025/2026

Contents

1 Advanced Matrix Operations

1.1 Adjoint of a Matrix

The adjoint (adjugate) of a square matrix is defined as the transpose of the cofactor matrix of that particular matrix. For a matrix A, the adjoint is denoted as $adj(A)$.

Important Steps to Find the Adjoint:

1. Find the minor matrix M of all the elements of matrix A.
2. Find the cofactor matrix C of all the minor elements of matrix M.
3. Find the $adj(A)$ by taking the transpose of the cofactor matrix C.

1.1.1 Adjoint of a 2×2 Matrix

For a 2×2 matrix, there is a simple trick to find the adjoint without calculating the cofactor matrix:

- Interchange the elements of the principal diagonal.
- Change the signs of the elements of the other diagonal.

For a matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, the adjoint is:

$$adj(A) = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Example: For $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$, the adjoint is $adj(A) = \begin{pmatrix} 4 & -2 \\ -3 & 1 \end{pmatrix}$.

```c
#include <stdio.h>
int main() {
    int A[2][2], i, j;

    printf("Enter the 4 elements of the matrix A:\n");
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            printf("Enter element A[%d][%d] = ", i, j);
            scanf("%d", &A[i][j]);
        }
    }

    printf("The entered matrix is: \n");
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
```

```c
            printf("%d\t", A[i][j]);
        }
        printf("\n");
    }

    // finding the adjoint of a 2x2 matrix
    int adj_A[2][2] = { {A[1][1], -A[0][1]},
                        {-A[1][0], A[0][0]} };

    printf("The adjoint of the matrix A is:\n");
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            printf("%d\t", adj_A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Listing 1: Program to Find Adjoint of a 2×2 Matrix

```
Enter the 4 elements of the matrix A:
Enter element A[0][0] = 1
Enter element A[0][1] = 2
Enter element A[1][0] = 3
Enter element A[1][1] = 4

The entered matrix is:
1 2
3 4

The adjoint of the matrix A is:
4 -2
-3 1
```

1.1.2 Adjoint of a 3×3 Matrix

The adjoint of a 3×3 matrix A is obtained by finding the transpose of the cofactor matrix of A. The cofactor of an element A_{ij} is obtained by multiplying its corresponding minor by $(-1)^{i+j}$.

For a matrix:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

The cofactor matrix is:

$$C = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}$$

where:

$$A_{11} = +(a_{22}a_{33} - a_{23}a_{32}) \quad A_{12} = -(a_{21}a_{33} - a_{23}a_{31}) \quad A_{13} = +(a_{21}a_{32} - a_{22}a_{31})$$
$$A_{21} = -(a_{12}a_{33} - a_{13}a_{32}) \quad A_{22} = +(a_{11}a_{33} - a_{13}a_{31}) \quad A_{23} = -(a_{11}a_{32} - a_{12}a_{31})$$
$$A_{31} = +(a_{12}a_{23} - a_{13}a_{22}) \quad A_{32} = -(a_{11}a_{23} - a_{13}a_{21}) \quad A_{33} = +(a_{11}a_{22} - a_{12}a_{21})$$

Then the adjoint is the transpose of the cofactor matrix:

$$adj(A) = C^T = \begin{pmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \\ A_{13} & A_{23} & A_{33} \end{pmatrix}$$

Property: The product of a matrix A and its adjoint is equal to the identity matrix multiplied by the determinant of A:

$$A \cdot adj(A) = adj(A) \cdot A = \det(A) \cdot I$$

```c
#include <stdio.h>
int main() {
    int A[3][3], adj_A[3][3], c_A[3][3], i, j, n = 3;

    printf("Enter the 9 elements of matrix:\n");
    for(i = 0; i < n; i++) {
        for(j = 0; j < n; j++) {
            printf("A[%d][%d] = ", i, j);
            scanf("%d", &A[i][j]);
        }
    }

    printf("\nThe matrix is:\n");
    for(i = 0; i < n; i++) {
        for(j = 0; j < n; j++) {
            printf("%d\t", A[i][j]);
        }
        printf("\n");
    }

    // Calculate cofactor matrix
    for(i = 0; i < n; i++) {
        for(j = 0; j < n; j++) {
            c_A[i][j] = (A[(i+1)%3][(j+1)%3] *
               A[(i+2)%3][(j+2)%3]) -
                        (A[(i+1)%3][(j+2)%3] *
                           A[(i+2)%3][(j+1)%3]);
        }
    }

    // Transpose of cofactor matrix gives adjoint
    printf("\nThe adjoint of matrix A is:\n");
    for(i = 0; i < n; i++) {
```

```c
            for(j = 0; j < n; j++) {
                adj_A[i][j] = c_A[j][i];
                printf("%d\t", adj_A[i][j]);
            }
            printf("\n");
        }

        return 0;
}
```

Listing 2: Program to Find Adjoint of a 3×3 Matrix

```
Enter the 9 elements of matrix:
A[0][0] = 3
A[0][1] = -1
A[0][2] = 2
A[1][0] = 2
A[1][1] = -2
A[1][2] = 3
A[2][0] = 4
A[2][1] = 1
A[2][2] = -4

The matrix is:
3 -1 2
2 -2 3
4 1 -4

The adjoint of matrix A is:
5 -2 1
20 -20 -5
10 -7 -4
```

1.2 Inverse of a Matrix

The inverse of a matrix A is defined as a matrix of the same order denoted by A^{-1} such that $A \cdot A^{-1} = A^{-1} \cdot A = I$, where I is the identity matrix.

Conditions for Inverse:

- The matrix must be square (same number of rows and columns).
- The determinant must not be zero: $\det(A) \neq 0$.
- If $\det(A) \neq 0$, the matrix is called invertible or nonsingular.
- If $\det(A) = 0$, the matrix is called singular and has no inverse.

The inverse of a matrix can be found using the determinant method:

$$A^{-1} = \frac{1}{\det(A)} \cdot adj(A)$$

1.2.1 Inverse of a 2×2 Matrix

```c
#include <stdio.h>
int main() {
    int A[2][2], i, j, det_A;
    float inv_A[2][2];

    printf("Enter the 4 elements of the matrix A:\n");
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            printf("A[%d][%d] = ", i, j);
            scanf("%d", &A[i][j]);
        }
    }

    printf("The entered matrix is: \n");
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            printf("%d\t", A[i][j]);
        }
        printf("\n");
    }

    // finding the determinant of a 2x2 matrix
    det_A = A[0][0] * A[1][1] - A[1][0] * A[0][1];
    printf("Determinant of the matrix A is: %d\n", det_A);

    if(det_A == 0) {
        printf("Determinant is 0, inverse of A does not
           exist.\n");
        return 0;
    }

    // finding the adjoint of a 2x2 matrix
    int adj_A[2][2] = { {A[1][1], -A[0][1]},
                        {-A[1][0], A[0][0]} };

    printf("The adjoint of the matrix A is:\n");
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            printf("%d\t", adj_A[i][j]);
        }
        printf("\n");
    }

    // finding the Inverse of A
    printf("\nInverse of matrix A is: \n");
```

```c
    for(i = 0; i < 2; i++) {
        for(j = 0; j < 2; j++) {
            inv_A[i][j] = (float)adj_A[i][j] / det_A;
            printf("%.2f\t", inv_A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Listing 3: Program to Find Inverse of a 2×2 Matrix

```
Enter the 4 elements of the matrix A:
A[0][0] = 1
A[0][1] = 2
A[1][0] = 3
A[1][1] = 4

The entered matrix is:
1 2
3 4
Determinant of the matrix A is: -2
The adjoint of the matrix A is:
4 -2
-3 1

Inverse of matrix A is:
-2.00 1.00
1.50 -0.50
```

```
Enter the 4 elements of the matrix A:
A[0][0] = 1
A[0][1] = 2
A[1][0] = 1
A[1][1] = 2

The entered matrix is:
1 2
1 2
Determinant of the matrix A is: 0
Determinant is 0, inverse of A does not exist.
```

1.2.2 Inverse of a 3×3 Matrix

```c
#include <stdio.h>
int main() {
    int A[3][3], C_A[3][3], adj_A[3][3], det = 0, i, j;
    float inv_A[3][3];

```

```
    printf("Enter the 9 elements of matrix:\n");
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            printf("A[%d][%d] = ", i, j);
            scanf("%d", &A[i][j]);
        }
    }

    printf("\nThe matrix A is:\n");
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            printf("%d\t", A[i][j]);
        }
        printf("\n");
    }

    // finding the Determinant of A
    for(i = 0; i < 3; i++) {
        det = det + (A[0][i] * (A[1][(i+1)%3] * A[2][(i+2)%3] -
                    A[1][(i+2)%3] * A[2][(i+1)%3]));
    }

    printf("Determinant is: %d\n", det);

    if(det == 0) {
        printf("Determinant is 0, inverse of A does not
           exist.\n");
        return 0;
    }

    // finding the cofactor matrix
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            C_A[i][j] = (A[(i+1)%3][(j+1)%3] *
               A[(i+2)%3][(j+2)%3]) -
                        (A[(i+1)%3][(j+2)%3] *
                           A[(i+2)%3][(j+1)%3]);
        }
    }

    // finding the adjoint (transpose of cofactor matrix)
    printf("\nAdjoint of matrix A is: \n");
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            adj_A[i][j] = C_A[j][i];
            printf("%d\t", adj_A[i][j]);
        }
        printf("\n");
    }

    // finding the Inverse of A
```

```c
    printf("\nInverse of matrix A is: \n");
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++) {
            inv_A[i][j] = (float)adj_A[i][j] / det;
            printf("%.2f\t", inv_A[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Listing 4: Program to Find Inverse of a 3×3 Matrix

```
Enter the 9 elements of matrix:
A[0][0] = 3
A[0][1] = -1
A[0][2] = 2
A[1][0] = 2
A[1][1] = -2
A[1][2] = 3
A[2][0] = 4
A[2][1] = 1
A[2][2] = -4

The matrix A is:
3 -1 2
2 -2 3
4 1 -4
Determinant is: 15

Adjoint of matrix A is:
5 -2 1
20 -20 -5
10 -7 -4

Inverse of matrix A is:
0.33 -0.13 0.07
1.33 -1.33 -0.33
0.67 -0.47 -0.27
```

```
Enter the 9 elements of matrix:
A[0][0] = 1
A[0][1] = 2
A[0][2] = 3
A[1][0] = 4
A[1][1] = 5
A[1][2] = 6
A[2][0] = 7
A[2][1] = 8
A[2][2] = 9

The matrix A is:
1 2 3
4 5 6
7 8 9
Determinant is: 0
Determinant is 0, inverse of A does not exist.
```

Summary of Advanced Matrix Operations

Operation	Description	Formula
Adjoint (2×2)	Transpose of cofactor matrix	$\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$
Adjoint (3×3)	Transpose of cofactor matrix	$adj(A) = C^T$
Inverse	Reciprocal of matrix	$A^{-1} = \frac{1}{\det(A)} \cdot adj(A)$
Singular Matrix	No inverse exists	$\det(A) = 0$

Table 1: Summary of Advanced Matrix Operations

Key Takeaways:

- The adjoint is the transpose of the cofactor matrix
- A matrix must be square and have non-zero determinant to be invertible
- The inverse formula: $A^{-1} = \frac{adj(A)}{\det(A)}$
- Always check if $\det(A) \neq 0$ before attempting to find the inverse
- For 2×2 matrices, there's a simple formula for both adjoint and inverse