Lectures 08:

Matrix Operations in C

Dr. Farouk KECITA

Department of Process Engineering
Faculty of Science and Technology
University of Mila
Level: 1st year ST - ENG & LMD

Academic Year 2025/2026

Outline

The Determinant of a Matrix

Definition

The determinant of a matrix is a special number that combines important information about the matrix into a single real number. It is defined only for square matrices (matrices with the same number of rows and columns).

Applications

- To solve systems of linear equations
- To record how a linear transformation changes area or volume
- To find the inverse of a matrix

Determinant of a 2×2 Matrix

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

Formula

$$\det(A) = |A| = ad - bc$$

Example

$$\begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} = 1 \times 4 - 3 \times 2 = 4 - 6 = -2$$

2×2 Determinant - C Program

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

    printf("Enter the 4 elements of the matrix A:\n");
    for(i = 0; i < 2; i++)
        for(j = 0; j < 2; 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");
    }

    det_A = A[0][0] * A[1][1] - A[1][0] * A[0][1];
    printf("Determinant = %ld\n", det_A);
    return 0;
}
```

2×2 Determinant - Output

Input/Output

```
Enter the 4 elements of the matrix A:
1 2 3 4
The entered matrix is:
1 2
3 4
Determinant = -2
```

Determinant of a 3×3 Matrix

For a matrix $A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$:

Formula

$$\det(A) = a(ei - fh) - b(di - fg) + c(dh - eg)$$

Visual Representation

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

3×3 Determinant - Method 1

```c
#include <stdio.h>
int main() {
    int a, b, c, d, e, f, g, h, i, det;

    printf("Enter the 9 elements:\n");
    scanf("%d%d%d%d%d%d%d%d%d",
          &a, &b, &c, &d, &e, &f, &g, &h, &i);

    det = a*(e*i - f*h) - b*(d*i - f*g) + c*(d*h - e*g);
    printf("Determinant = %d\n", det);
    return 0;
}
```

Output

```
Enter the 9 elements: 1 2 3 4 5 6 7 8 9
Determinant = 0
```

3×3 Determinant - Method 2 (Using Array)

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

    printf("Input elements in the matrix A:\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("Matrix A:\n");
    for(i = 0; i < 3; i++) {
        for(j = 0; j < 3; j++)
            printf("%d ", A[i][j]);
        printf("\n");
    }

    for(j = 0; j < 3; j++)
        det += A[0][j] * (A[1][(j+1)%3] * A[2][(j+2)%3] -
                          A[1][(j+2)%3] * A[2][(j+1)%3]);

    printf("Determinant = %d\n", det);
    return 0;
}
```

3×3 Determinant - Output

Input/Output

```
Input elements in the matrix A:
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
Matrix A:
1 2 3
4 5 6
7 8 9
Determinant = 0
```

Trace of a Matrix

Definition

The trace of a matrix is defined as the sum of the principal diagonal elements of a square matrix. It is represented as $\mathrm{tr}(\mathbf{A})$, where A is any square matrix of order $n \times n$.

For a 3×3 matrix $A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$:

Formula

$$\mathrm{tr}(A) = a + e + i$$

Example

$$\mathrm{tr} \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = 1 + 5 + 9 = 15$$

Algorithm to Find Trace

Steps

1. Initialize `trace = 0`
2. For `i = 0` to `n-1`
3. For `j = 0` to `n-1`
4. If `i == j`, then `trace += A[i][j]`
5. Print the result

Note

Only diagonal elements (where row index = column index) are added.

Trace of a Matrix - C Program

```c
#include <stdio.h>
int main() {
    int A[10][10], n, i, j, trace;

    printf("Enter order of the square matrix: ");
    scanf("%d", &n);

    printf("\nEnter the elements of the matrix A:\n");
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
            scanf("%d", &A[i][j]);
        printf("Matrix A:\n");
    for(i = 0; i < n; i++) {
        for(j = 0; j < n; j++)
            printf("%d ", A[i][j]);
        printf("\n");
    }
        trace = 0;
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
            if(i == j)
                trace += A[i][j];
        printf("Trace = %d\n", trace);
    return 0;
}
```

Trace of a Matrix - Output

Input/Output

```
Enter order of the square matrix: 3

Enter the elements of the matrix A:
1 2 3 4 5 6 7 8 9
Matrix A:
1 2 3
4 5 6
7 8 9
Trace = 15
```

Transpose of a Matrix

Definition

The transpose of a matrix is a new matrix obtained by exchanging its rows and columns. If A is of size $m \times n$, then its transpose A^T is of size $n \times m$.

Example

$$A_{3\times 2} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \quad \Rightarrow \quad A^T_{2\times 3} = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

In C Programming

The transpose is obtained by changing `A[i][j]` to `A[j][i]`.

Transpose of a Matrix - Algorithm

Steps

1. Input matrix A of size $m \times n$
2. Create matrix T of size $n \times m$ for the transpose
3. For `i = 0 to m-1`
4. `    For j = 0 to n-1`
5. `        T[j][i] = A[i][j]`
6. Print matrix T

Visual Representation

$$A[i][j] \rightarrow T[j][i]$$

Transpose of a Matrix - C Program

```c
#include <stdio.h>
int main() {
    int A[10][10], T[10][10], m, n, i, j;

    printf("Enter number of rows and columns: ");
    scanf("%d %d", &m, &n);
    printf("\nEnter matrix elements:\n");
    for(i = 0; i < m; i++){
        for(j = 0; j < n; j++) {
            printf("A[%d][%d] = ", i, j);
            scanf("%d", &A[i][j]);
        }}
    printf("\nOriginal matrix:\n");
    for(i = 0; i < m; i++) {
        for(j = 0; j < n; j++){
            printf("%d ", A[i][j]);}
        printf("\n");
    }
    // Compute transpose
    for(i = 0; i < m; i++){
        for(j = 0; j < n; j++){
            T[j][i] = A[i][j];}}
    printf("\nTranspose matrix:\n");
    for(i = 0; i < n; i++) {
        for(j = 0; j < m; j++){
            printf("%d ", T[i][j]);}
        printf("\n");
    }
    return 0;}
```

Transpose of a Matrix - Output

Input/Output

```
Enter number of rows and columns: 3 2
Enter matrix elements:
A[0][0] = 1
A[0][1] = 2
A[1][0] = 3
A[1][1] = 4
A[2][0] = 5
A[2][1] = 6
Original matrix:
1 2
3 4
5 6
Transpose matrix:
1 3 5
2 4 6
```

Summary of Matrix Operations

Operation	**Description**	**Formula**
Determinant (2X2)	Special number	$\det = ad - bc$
Determinant (3X3)	Special number	$\det = a(ei - fh)$ - b(di-fg) + c(dh-eg)
Trace	Sum of diagonal	$\text{tr}(A) = \sum_{i=1}^{n} A[i][i]$
Transpose	Exchange rows/columns	$A^T[i][j] = A[j][i]$

Key Takeaways

Important Points

- Determinant is defined only for square matrices
- Trace is the sum of diagonal elements
- Transpose swaps rows and columns
- 2D arrays in C are stored in row-major order
- Always check matrix dimensions before operations

Matrix Operations in C

- Use nested loops to access elements
- Array indices start at 0
- Be careful with pointer arithmetic in transpose