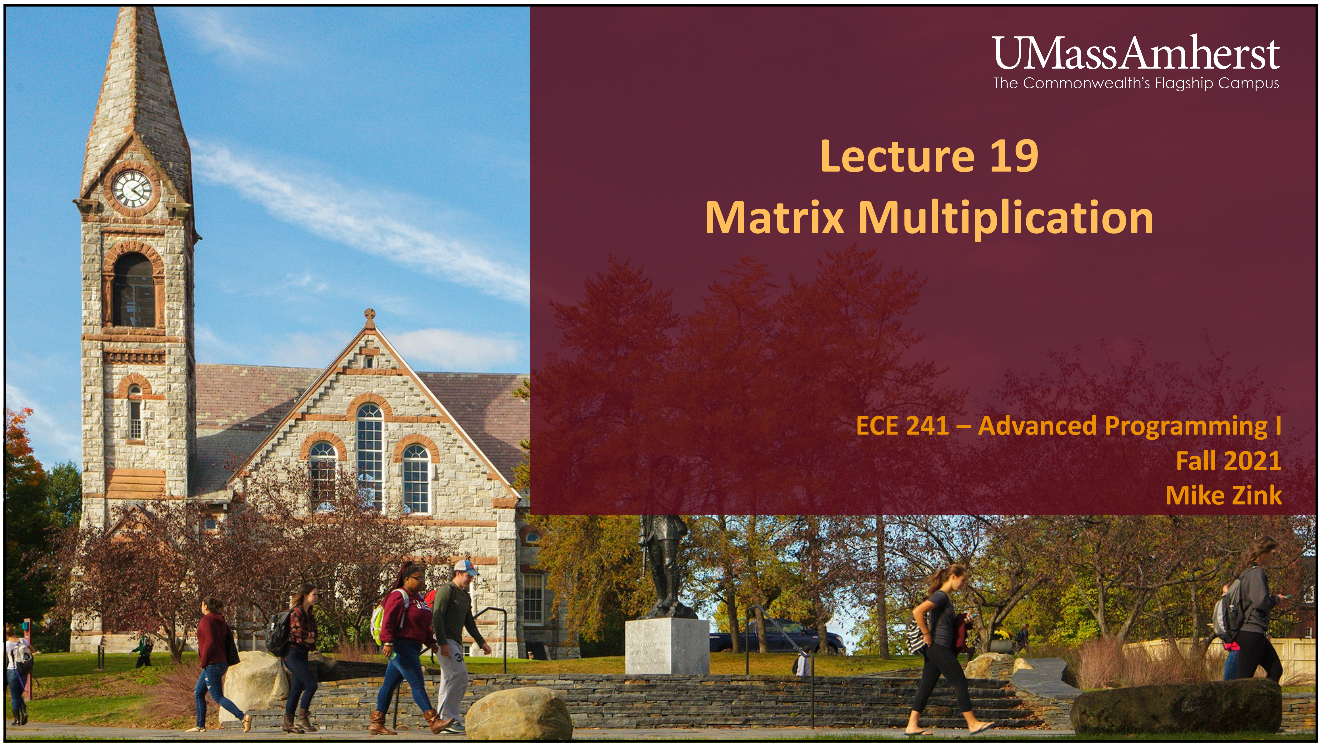

UMassAmherst
The Commonwealth's Flagship Campus

Lecture 19
Matrix Multiplication

ECE 241 – Advanced Programming I
Fall 2021
Mike Zink

0

UMassAmherst

Introduction

- Matrix operations are essential in science and engineering
- Traditional way of matrix multiplication is complex
- Divide and conquer approach to make matrix multiplication more efficient

1

1

Traditional Matrix Multiplication

- Multiplication of two matrices of size N, N x N
  E.g.: A x B = C
- $\begin{vmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{vmatrix} = \begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} \begin{vmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{vmatrix}$
- $C_{11} = A_{11}B_{11} + A_{12}B_{12}$
- $C_{12} = A_{11}B_{12} + A_{12}B_{22}$
- $C_{21} = A_{21}B_{11} + A_{22}B_{21}$
- $C_{22} = A_{21}B_{12} + A_{22}B_{22}$
- Can be accomplished in 8 multiplications $\left(2^{log_2 8} = 2^3\right)$

UMassAmherst

Traditional Matrix Multiplication

- Can be accomplished in 8 multiplications $\left(2^{log_2 8} = 2^3\right)$
- Time analysis:

$$C_{i,j} = \sum_{k=1}^{N} a_{i,k} b_{k,j}$$

Thus $T(N) = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} c = cN^3 = O(N^3)$

Traditional Matrix Multiplication

```
matrix1 = [[12, 7, 3],
           [4, 5, 6],
           [7, 8, 9]]
matrix2 = [[5, 8, 1],
           [6, 7, 3],
           [4, 5, 9]]

res = [[0 for x in range(3)] for y in range(3)]

# explicit for loops
for i in range(len(matrix1)):
    for j in range(len(matrix2[0])):
        for k in range(len(matrix2)):
            # resulted matrix
            res[i][j] += matrix1[i][k] * matrix2[k][j]

print(res)
```

4

4

Divide and Conquer

- Divide matrices into sub-matrices
- Use blocks to apply multiplication
- Recursively multiply sub-matrices

$$A \times B = R$$

$$\begin{bmatrix} A_0 & A_1 \\ A_2 & A_3 \end{bmatrix} \times \begin{bmatrix} B_0 & B_1 \\ B_2 & B_3 \end{bmatrix} = \begin{bmatrix} A_0 \times B_0 + A_1 \times B_2 & A_0 \times B_1 + A_1 \times B_3 \\ A_2 \times B_0 + A_3 \times B_2 & A_2 \times B_1 + A_3 \times B_3 \end{bmatrix}$$

5

5

UMassAmherst

Divide and Conquer Termination

- Terminate recursion with simple base case

$$A \times B = R$$

$$\boxed{a_0} \times \boxed{b_0} = \boxed{a_0 \times b_0}$$

6

UMassAmherst

Strassen's Matrix Multiplication

- $\begin{vmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{vmatrix} = \begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} \begin{vmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{vmatrix}$

$$P_1 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$P_2 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$P_3 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$P_4 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$P_5 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$P_6 = (A_{11} + A_{22})(B_{11} + B_{22})$$
$$P_7 = (A_{11} + A_{22})(B_{11} + B_{22})$$

$$C_{11} = P_1 + P_4 - P_5 + P_7$$
$$C_{12} = P_3 + P_5$$
$$C_{21} = P_2 + P_4$$
$$C_{22} = P_1 + P_3 - P_2 + P_6$$

7

7

UMassAmherst

Strassen Implementation -- Split

```python
def split(matrix):
    """
    Splits a given matrix into quarters.
    Input: nxn matrix
    Output: tuple containing 4 n/2 x n/2 matrices corresponding to a, b, c, d
    """
    row, col = matrix.shape
    row2, col2 = row // 2, col // 2
    return matrix[:row2, :col2], matrix[:row2, col2:], matrix[row2:, :col2], matrix[row2:, col2:]
```

8

8

UMassAmherst

Strassen

```python
def strassen(x, y):
    """
    Computes matrix product by divide and conquer approach, recursively.
    Input: nxn matrices x and y
    Output: nxn matrix, product of x and y
    """

    # Base case when size of matrices is 1x1
    if len(x) == 1:
        return x * y

    # Splitting the matrices into quadrants. This will be done recursively
    # until the base case is reached.
    a, b, c, d = split(x)
    e, f, g, h = split(y)

    # Computing the 7 products, recursively (p1, p2...p7)
    p1 = strassen(a, f - h)
    p2 = strassen(a + b, h)
    p3 = strassen(c + d, e)
    p4 = strassen(d, g - e)
    p5 = strassen(a + d, e + h)
    p6 = strassen(b - d, g + h)
    p7 = strassen(a - c, e + f)

    # Computing the values of the 4 quadrants of the final matrix c
    c11 = p5 + p4 - p2 + p6
    c12 = p1 + p2
    c21 = p3 + p4
    c22 = p1 + p5 - p3 - p7

    # Combining the 4 quadrants into a single matrix by stacking horizontally and vertically.
    c = np.vstack((np.hstack((c11, c12)), np.hstack((c21, c22))))

    return c
```

9

9

UMassAmherst

Time Complexity

- $T(1) = 1$ (assuming $N = 2^k$)
- $T(N) = 7T\left(\frac{N}{2}\right)$
- $T(N) = 7^k T\left(\frac{N}{2^k}\right) = 7^k$
- $T(N) = 7^{logN} = N^{log7} = N^{2.81}$

10

10

UMassAmherst

Next Steps

- Next lecture on Tuesday 11/23
- Discussion on 11/18 and 11/23

11

11

UMassAmherst
The Commonwealth's Flagship Campus

12