

# Matrices

Dov Kruger

Department of Electrical and Computer Engineering  
Stevens Institute of Technology

November 20, 2022



## Matrices

- Major topic in mathematics
- Very important in numerical computation
- Illustrate some of the limitations of current technology
- Show where we should not use simplify assumptions such as all memory is random access

A matrix is a 2D table of numbers (rows, columns)

Matrices can be thought of in a number of ways

- A set of rows each representing the coefficients of a linear equation
- A set of columns each representing an n-dimensional vector

$$\begin{pmatrix} 1 & 4 & -2 \\ 2 & 9 & 2 \\ 3 & -2 & 1 \end{pmatrix}$$

# Two ways of Looking at Matrices

1	4	-2	.....→	$1x + 4y - 2z$
-1	3	3	.....→	$-1x + 3y + 3z$
4	1	5		

1	4	-2
-1	3	3
4	1	5

# Matrix Type

A square matrix is a matrix where the number of rows and columns are equal

The main diagonal is the locations in the matrix where row = column

The identity matrix has all zeros except for ones down the main diagonal

Example:

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

# Identity Matrix

In an Algebra, there must exist an identity element such that

$$Ix = xI = x$$

For scalars this is the number 1. For matrices it is the identity matrix:

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The same definition will apply, but in order to understand it we must define matrix multiplication

# Matrix Multiplication

Matrix multiplication is defined on two matrices of size

- $m$  rows  $\times$   $n$  columns
- $n$  rows  $\times$   $p$  columns

Notice that the  $\text{columns}(A) = \text{rows}(B)$

Each element of the answer  $C_{jk}$  is the dot product of a  $\text{row}(A)$   $\text{column}(B)$

1	3	0
2	-1	3

\*

-1	2
-2	4
0	3

=

-7	

# Matrix Multiplication, contd

$$\begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 3 \end{bmatrix} * \begin{bmatrix} -1 & 2 \\ -2 & 4 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} -7 & 14 \\ & \end{bmatrix}$$

Calculation for the element in the first row, second column:  $1(2) + 3(4) + 0(3) = 14$

$$\begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 3 \end{bmatrix} * \begin{bmatrix} -1 & 2 \\ -2 & 4 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} -7 & 14 \\ 0 & \end{bmatrix}$$

Calculation for the element in the second row, first column:  $2(-1) + -1(-2) + 3(0) = 0$



# Matrix Multiplication, done!

$$\begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 3 \end{bmatrix} * \begin{bmatrix} -1 & 2 \\ -2 & 4 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} -7 & 14 \\ 0 & 9 \end{bmatrix}$$

$$2(2) + -1(4) + 3(3) = 9$$

# Inverse

The inverse of a matrix  $A^{-1}$  is a matrix such that  
 $AA^{-1} = A^{-1}A = I$

With scalars, only the number 0 has no inverse.

Many matrices are singular matrix and have no inverse

Example:

$$\begin{pmatrix} 1 & -2 \\ -3 & 6 \end{pmatrix}$$

# Matrix Multiplication Pseudocode

```
for i ← 0 to m-1
  for j ← 0 to p-1
    dot ← 0                                // dot product of  $A_i * B_j$ 
    for k ← 0 to n-1
      dot ← dot +  $A_{ik}B_{kj}$ 
    end
     $C_{ij} \leftarrow$  dot
  end
end
```

# Matrix Multiplication is not Commutative

With scalars  $AB = BA$

Example:  $2 * 3 = 3 * 2$

With matrices, not only is  $AB \neq BA$ , they are not even the same dimension

Example (let's use octave to see this)

$$A = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

$$A = 1 \quad 2 \quad 3$$

$$B = \begin{bmatrix} 1; 2; 3 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$$A * B = 14$$

$$B * A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}$$

# Complexity of Matrix Multiplication

$$(m \times n) * (n \times p) \rightarrow (m \times p)$$

Each element of the answer is the dot product of  $n$  elements  $\rightarrow O(mnp)$

If  $m = n = p$  then  $O(n^3)$

Everything interesting in matrices is  $O(n^3)$

In fact, just to read every element of a matrix of size  $n \times n$  is  $O(n^2)$

# Faster Matrix Multiplication

It turns out that while the brute force matrix multiplication is  $O(n^3)$  there is a cleverer way to beat that

It is not that useful in practice, but it points to a general pattern

Brute force algorithm  $O(n^2)$  (sorting)  $\rightarrow O(n \log n)$  (quicksort, heapsort, mergesort)

Many things that seem to be  $O(n^2)$  can be done faster

- Multiplication of two  $n$ -digit numbers
- Fourier Transform

# Strassen: Faster Multiplication

In 1969 Volker Strassen derived an algorithm equivalent to matrix multiplication, but faster

Consider a matrix in which both dimensions are a power of 2

- Each matrix can be broken up into quarters
- The relationship between the quarters is as shown

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$

$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$

Notice, there are 8 multiplications

Each is  $(\frac{1}{2})(\frac{1}{2})(\frac{1}{2}) = \frac{1}{8}$  the size

Complexity is the same as the original problem

# Strassen, continued

The genius of Strassen's solution was in figuring out how to combine terms and do less computation

$$\begin{aligned}M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) & 7 \text{ multiplications} &= \left(\frac{7}{8}n\right)^3 \\M_2 &= (A_{21} + A_{22})B_{11} & 18 \text{ additions} &= 18n^2 \\M_3 &= A_{11}(B_{12} - B_{22}) & O(2^{\log 7}) &= O(n^{2.8074})\end{aligned}$$

$$\begin{aligned}M_4 &= A_{22}(B_{21} - B_{11}) \\M_5 &= (A_{11} + A_{12})B_{22} \\M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\M_7 &= (A_{12} - A_{22})(B_{21} + B_{22})\end{aligned}$$

$$\begin{aligned}C_{11} &= M_1 + M_4 - M_5 + M_7 & C_{12} &= M_3 + M_5 \\C_{21} &= M_2 + M_4 & C_{22} &= M_1 - M_2 + M_3 + M_6\end{aligned}$$



Strassen is important theoretically

- It means it is possible to multiply matrices in less than  $O(n^3)$
- Strassen is  $O(n^{2.807})$
- Others have followed: Coppersmith-Winograd  $O(n^{2.36})$
- None are very practical for most cases
  - Require copying and reordering that are slow (faster with very large matrices)
  - Numerical results become less stable (see later chapter on numerical methods)

In practice faster methods are not used, but the idea that  $O(n^3)$  algorithms can be faster is important

# Matrix Addition

Matrix addition requires two matrices with the same number of rows and columns

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & -1 & 1 \end{pmatrix} + \begin{pmatrix} 3 & 0 & -1 \\ 1 & -2 & -3 \end{pmatrix} = \begin{pmatrix} 4 & 2 & 2 \\ 3 & -3 & -2 \end{pmatrix}$$

It results in a matrix where each element is the sum of the corresponding element in the other two

# Solving Systems of Linear Equations

In high school, you learned to solve systems of linear equations

$$\begin{aligned}x + 2y + z &= 4 \\ 3x + 2y - z &= 15 \\ -x + y + 2z &= 5\end{aligned}$$

1	2	1	4
3	2	-1	15
-1	1	2	5

You might have been taught to do this either by

- substitution (figure out what  $z$  is and replace it)
- add/subtract equations from each other (row reduction)

Row reduction works better as the problems scale up. Neither solution is numerically stable without help

# How did they do it in high school? They cheated

The methods taught in high school for solving systems of linear equations do not work in general

- So how did they make it work? They cheated
- The analogy is to shoot a hole in a wall, then paint the bullseye around it
- Perfect shot! You are an expert marksman

1. Pick a set of answers at random:  $x=3$ ,  $y = 4$ ,  $z = 2$
2. Pick a random set of coefficients (must be linearly independent, more later)

$$x + 2y + z =$$

$$3x + 2y - z =$$

$$-x + y + 2z =$$

# How did they do it in high school (contd)

Compute the answers. You now know that there is a perfect solution to this problem

$$x + 2y + z = 3 + 2(4) + 2 = 13$$

$$3x + 2y - z = 3(3) + 2(4) - 2 = 15$$

$$-x + y + 2z = -3 + 4 + 2(2) = 5$$

# Error Conditions with Systems of Linear Equations

For Solving a set of linear equations

- No row may be zero (undetermined)
- **There must be as many rows as columns (n equations, n unknowns)**
- Each row must be linearly independent from the others
- Each column must contain some information

Not enough equations

$$3x + 2y + z = 5$$

$$4x - y + 3z = 7$$

# Error Conditions with Systems of Linear Equations

For Solving a set of linear equations

- No row may be zero (undetermined)
- **There must be as many rows as columns (n equations, n unknowns)**
- Each row must be linearly independent from the others
- Each column must contain some information

Too many equations (overdetermined)

$$3x + 2y + z = 5$$

$$4x - y + 3z = 7$$

$$2x + 3y - 4z = 11$$

$$x - 4y + 2z = 5$$

# Error Conditions with Systems of Linear Equations, case 2

For Solving a set of linear equations

- No row may be zero (undetermined)
- There must be as many rows as columns (n equations, n unknowns)
- **Each row must be linearly independent from the others**
- Each column must contain some information

Equations are not linearly independent

$$4x + 2y + z = 5$$

$$2x + y + 1/2z = 6$$

$$x - 3y + 4z = 19$$



# Error Conditions with Systems of Linear Equations, case 2

For Solving a set of linear equations

- No row may be zero (undetermined)
- There must be as many rows as columns (n equations, n unknowns)
- Each row must be linearly independent from the others
- **Each column must contain some information**

One variable is not used at all

$$4x + 0y + z = 5$$

$$2x + 0y + 1/2z = 6$$

$$x - 0y + 4z = 19$$

# Solving a System of Linear Equations with Row Reduction

Row Reduction, called Gauss-Jordan (yes, that Gauss!)

Augment the matrix with one extra column (the answers)

Reduce to an upper-triangular form by multiplying each row by a constant

Add to the other row to zero out specific values

Back-propagate to solve the variables

$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 2 & -1 \\ -1 & 1 & 2 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 13 \\ 15 \\ 5 \end{bmatrix}$$



## Row Reduction, part 2

- Augment the matrix with one extra column (the answers)
- For the first column, use the top element to zero out the others

1	2	1	13
3	2	-1	15
-1	1	2	5

$$s = 1$$

$$s * r + 3 = 0 \quad \leftarrow -3$$

$$s * r + (-1) = 0 \quad r \leftarrow 1$$

zero these

## Row Reduction, part 3

Multiply the first row by -3 and add it to the second row:  $n+1 = O(n)$

1	2	1	13
3	2	-1	15
-1	1	2	5

1	2	1	13
0	-4	-4	-24
-1	1	2	5

# Row Reduction, part 3

Multiply the first row by 1 and add it to the third row

1	2	1	13
3	2	-1	15
-1	1	2	5

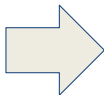
1	2	1	4
0	-4	-4	-24
0	3	3	9

Complexity:  $n * (n - 1) = \theta(n^2)$

# Row Reduction, part 4

Repeat for the second column

1	2	1	13
0	-4	-4	-24
0	3	3	9



1	2	1	13
0	-4	-4	-24
0	0	0	11.25

Complexity:  $n * (n-1) = \Theta(n^2)$

$r = -3 / -4 = .75$   
 $-4 * .75 + 3 \rightarrow 0$   
 $3 * .75 + 9 \rightarrow 11 \frac{1}{4}$

# Oops!

For this small example, that's all the row reduction, but in general

$$n(n-1) + (n-1)(n-2) + \dots + (2)(1) = O(n^3)$$

The system is not yet solved. It must be back-propagated



# III-Conditioned Systems

A system that has all zeros on one row, or two rows that are linearly dependent cannot be solved

# The Unpleasant Reality: Matrices are Hard

It is extremely easy to get an ill conditioned matrix just by row reduction

This happens when a pivot value is too small Example:

$$\begin{pmatrix} .001 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 5 & -1 \end{pmatrix} \rightarrow \begin{pmatrix} .001 & 2 & 3 \\ 0 & -1997 & -2998 \\ 2 & 5 & -1 \end{pmatrix}$$

$row2 = row2 - 1000 * row1$  The resulting matrix is ill conditioned

# Making Matrices More Stable: Partial Pivoting

In order to maximize the stability, use the biggest number

Swap rows until the biggest number is on top

Example: 
$$\begin{pmatrix} .001 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 5 & -1 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & 5 & -1 \\ 1 & 3 & 2 \\ .001 & 2 & 3 \end{pmatrix}$$

Now, instead of multiplying by -1000, the coefficients are -0.5 and -0.0005

$$\begin{pmatrix} 2 & 5 & -1 \\ 0 & 0.5 & 2.5 \\ 0 & 1.9995 & 3.0005 \end{pmatrix}$$

# Making Matrices More Stable: Full Pivoting

For even greater stability, at a cost in bookkeeping

Find the largest value in the matrix

Swap rows *and* columns to get the value with the largest absolute value to top-left

Example: 
$$\begin{pmatrix} .001 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 5 & -1 \end{pmatrix} \rightarrow \begin{pmatrix} 5 & 2 & -1 \\ 3 & 1 & 2 \\ 2 & .001 & 3 \end{pmatrix}$$

However, now you must track which columns have been transposed

This can be done by augmenting the matrix with a new first row of length n Each element stores the position of the variable in that column

# Complexity of Partial and Full Pivoting

1. Reducing a row  $O(n)$
2. Reducing  $n$  rows  $O(n^2)$ . This zeros one column
3. Reducing  $n$  columns  $O(n^3)$
4. Finding largest value in a column  $O(n)$
5. Doing that for  $n$  columns (Partial Pivoting)  $O(n^2)$

# Gram-Schmidt Process

Gram Schmidt orthogonalization is an algorithm to make a matrix orthonormal

1. Each column vector becomes normal (unit length)
2. Each column vector is perpendicular to the others (orthogonal)

$$\begin{pmatrix} 5 & 2 & -1 \\ 3 & 1 & 2 \\ 2 & .001 & 3 \end{pmatrix}$$

# Gram-Schmidt Complexity

1. To calculate the length of a vector is  $O(n)$
2. To divide every element by that length is  $O(n)$
3. To subtract the component of one vector in the direction of another is  $O(n)$
4.  $n + 2n + 3n + \dots n^2 = O(n^3)$

# Representation of Matrices

There are two ways

- Single block of contiguous memory
- Array of pointers



# Row Major vs. Column Major Order

# Indexing Equation for Row and Column Major Matrices

# Impact of Sequential vs. Non-Sequential Memory Access

Normally in algorithms we don't worry about machine specifics

But in the case of matrices, memory order has a significant impact

# Sparse Matrices

A sparse matrix is one where most of the elements are zero

An encoding scheme is needed to encode the non-zero elements

A general sparse matrix is not particularly common

Instead, we will deal with specific shapes

- Upper and Lower triangular matrices
- Diagonal matrices
- Tridiagonal matrices

# Upper Triangular Matrices

In an upper-triangular matrix, all elements below the main diagonal are zero

This means:

- The first row has  $n$  elements
- The second row has  $n-1$  elements
- The last row has 1 element

*	*	*	*	*
0	*	*	*	*
0	0	*	*	*
0	0	0	*	*
0	0	0	0	*

# Lower Triangular Matrices

In a lower-triangular matrix, all elements above the main diagonal are zero

This means:

- The first row has 1 element
- The second row has 2 elements
- The last row has n elements

*	0	0	0	0
*	*	0	0	0
*	*	*	0	0
*	*	*	*	0
*	*	*	*	*

# Diagonal Matrices

A Diagonal Matrix has non-zero elements only on the main diagonal

Because of this storage is only  $O(n)$

Complexity of matrix multiplication is also  $O(n)$

*	0	0	0	0
0	*	0	0	0
0	0	*	0	0
0	0	0	*	0
0	0	0	0	*

# Applications of Diagonal Matrices

Calculating  $A^n$  for a matrix is quite expensive

$A * A$  is  $O(n^3)$

Brute force would be  $O(n^4)$

- We can do a lot better with number-theoretic algorithms, covered later
- The main problem is that multiplication is so expensive

However, if you could just decompose the matrix into three pieces:

$A = PDP^{-1}$  (this is called an EigenValue decomposition,  $O(n^3)$ )

where  $D$  is a diagonal matrix, then  $A^n$  can be computed as

$A^n = PD^nP^{-1}$   $D^n$  is  $O(n)$



# Tridiagonal Matrices

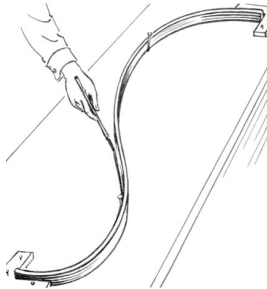
A Tridiagonal Matrix has non-zero elements on the main diagonal and one on each side

*	*	0	0	0
*	*	*	0	0
0	*	*	*	0
0	0	*	*	*
0	0	0	*	*

# Applications of Tridiagonal Matrices: Numerical Splines

A spline was originally a thin strip of wood used to create smooth curves

- Used to create the shape for optimal boat hulls that have less drag (laminar flow)
- The thin wood means that the curvature is constant (2nd derivative is constant)



# Applications of Tridiagonal Matrices: Numerical Splines, contd.

Once computers began to be used to design objects, question is how to simulate a spline?

