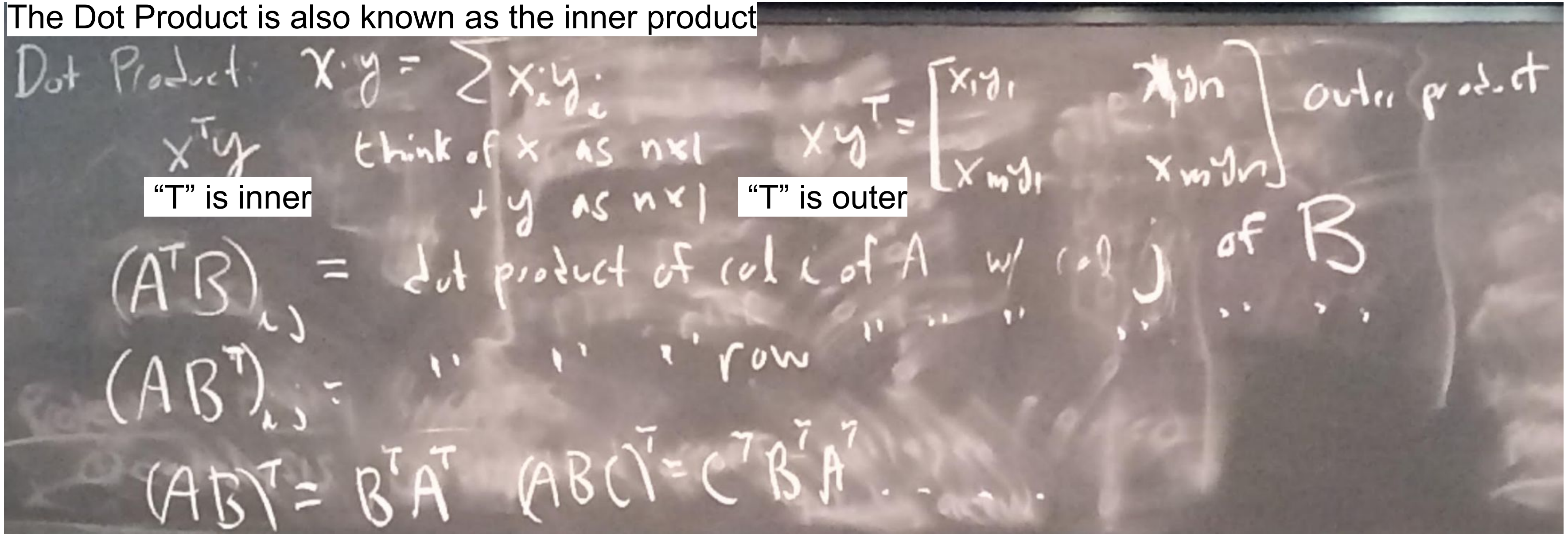
Lecture 5

Orthogonal Matrices
2x2 Matrix Factorizations
Block Matrices

Inner and Outer Products, and matmul tricks

n vectors are sometimes thought of as n x 1 matrices. This causes little trouble in a linear algebra context and tons of trouble in software with vectors and matrices.

The Dot Product is also known as the inner product



Dot Product: $x \cdot y = \sum x_i y_i$

$x^T y$ think of x as $n \times 1$ & y as $n \times 1$

$xy^T = \begin{bmatrix} x_1 y_1 & x_1 y_n \\ x_m y_1 & x_m y_n \end{bmatrix}$ outer product

“T” is inner

“T” is outer

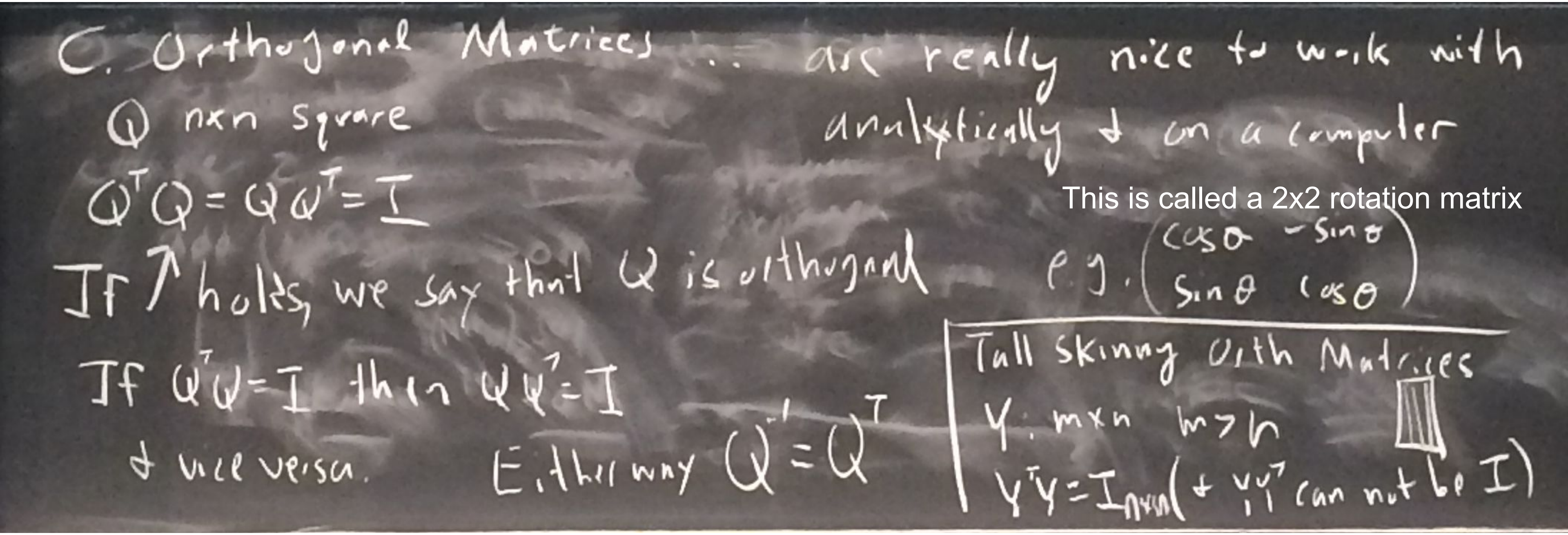
$(A^T B)_{ij}$ = dot product of col i of A w/ col j of B

$(AB^T)_{ij}$ = " " " row " " " " "

$(AB)^T = B^T A^T$ $(ABC)^T = C^T B^T A^T$ . . .

(Matmul is a commonly used abbreviation for Matrix Multiply)

Orthogonal Matrices



C. Orthogonal Matrices ... are really nice to work with analytically & on a computer

Q $n \times n$ square

$Q^TQ = QQ^T = I$

If ↑ holds, we say that Q is orthogonal

e.g. $\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$

This is called a 2x2 rotation matrix

If $Q^TQ = I$ then $QQ^T = I$ & vice versa. Either way $Q^{-1} = Q^T$

Tall Skinny Orth Matrices

Y: $m \times n$ $m > n$

$Y^TY = I_{n \times n}$ (& YY^T can not be I)

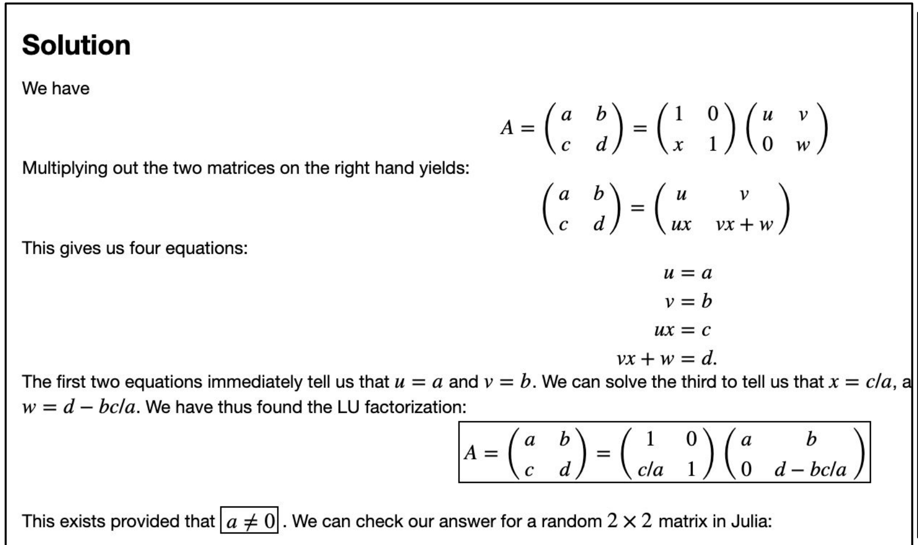
Terminology note: we never say that a matrix is orthonormal, but we do say that the columns of an orthogonal (or tall-skinny) orthogonal matrix are orthonormal.

Solving for 2x2 Factorizations

Solving for the 2x2 LU factorization

This is called the 2 x 2 LU factorization: Given $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, find a factorization of A in the form $\begin{pmatrix} 1 & 0 \\ x & 1 \end{pmatrix} \begin{pmatrix} u & v \\ 0 & w \end{pmatrix}$.

(Write down x, u, v, w in terms of a, b, c, d) On what conditions for a, b, c, d does the LU factorization exist?



Solution

We have

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ x & 1 \end{pmatrix} \begin{pmatrix} u & v \\ 0 & w \end{pmatrix}$$

Multiplying out the two matrices on the right hand yields:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} u & v \\ ux & vx + w \end{pmatrix}$$

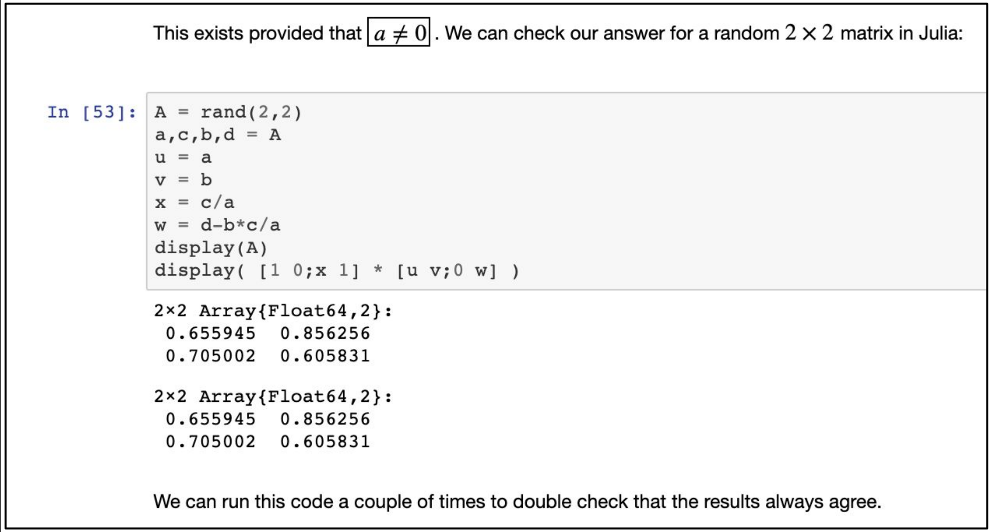
This gives us four equations:

$$\begin{aligned} u &= a \\ v &= b \\ ux &= c \\ vx + w &= d. \end{aligned}$$

The first two equations immediately tell us that $u = a$ and $v = b$. We can solve the third to tell us that $x = c/a$, a
$w = d - bc/a$. We have thus found the LU factorization:

$$\boxed{A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ c/a & 1 \end{pmatrix} \begin{pmatrix} a & b \\ 0 & d - bc/a \end{pmatrix}}$$

This exists provided that $\boxed{a \neq 0}$. We can check our answer for a random 2×2 matrix in Julia:



This exists provided that $\boxed{a \neq 0}$. We can check our answer for a random 2×2 matrix in Julia:

```
In [53]: A = rand(2,2)
         a,c,b,d = A
         u = a
         v = b
         x = c/a
         w = d-b*c/a
         display(A)
         display( [1 0;x 1] * [u v;0 w] )

2×2 Array{Float64,2}:
 0.655945  0.856256
 0.705002  0.605831

2×2 Array{Float64,2}:
 0.655945  0.856256
 0.705002  0.605831
```

We can run this code a couple of times to double check that the results always agree.

Solving for the 2x2 “QR” Factorization

We will later see the nxn. The 2x2 QR Factorization and its solution is here:

This is called the 2x2 QR factorization: Given $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, find a factorization of A in the form $\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} u & v \\ 0 & w \end{pmatrix}$. The matrix $Q = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ is known as a 2x2 rotation matrix. (Hint1: what are the polar coordinates for the point(a,c)?? You should be able to write a formula for $u, \cos\theta, \sin\theta$ from a and c using only division and square roots. (at least if a and c are not both 0). Hint 2, use what we just mentioned about rotation matrices.)

Solution

Firstly we multiply out

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} \cos^2\theta + \sin^2\theta & -\sin\theta\cos\theta + \sin\theta\cos\theta \\ -\sin\theta\cos\theta + \sin\theta\cos\theta & \cos^2\theta + \sin^2\theta \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

We have just shown that if $Q = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$, then $Q^{-1} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$.

To perform the QR factorization, we want to find u, v, w, θ such that

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} u & v \\ 0 & w \end{pmatrix}.$$

Comparing the first column of both sides of the equation, we obtain $a = u\cos\theta$ and $c = u\sin\theta$. But these are just the polar equations for the point (a, c), and so:

$$u = \sqrt{a^2 + c^2}$$
$$\cos\theta = \frac{a}{\sqrt{a^2 + c^2}}$$
$$\sin\theta = \frac{c}{\sqrt{a^2 + c^2}}.$$

Next we notice that

$$\begin{pmatrix} b \\ d \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} v \\ w \end{pmatrix}$$
$$= Q\begin{pmatrix} v \\ w \end{pmatrix}$$

and so

$$\begin{pmatrix} v \\ w \end{pmatrix} = Q^{-1}\begin{pmatrix} b \\ d \end{pmatrix}$$
$$= \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}\begin{pmatrix} b \\ d \end{pmatrix}$$
$$= \begin{pmatrix} a/u & c/u \\ -c/u & a/u \end{pmatrix}\begin{pmatrix} b \\ d \end{pmatrix}$$
$$= \begin{pmatrix} \frac{ab+cd}{\sqrt{a^2+c^2}} \\ \frac{ad-bc}{\sqrt{a^2+c^2}} \end{pmatrix}$$

These expressions all hold provided that both a and c are not identically 0. We can double check the calculation using Julia:

These expressions all hold provided that both a and c are not identically 0. We can double check the calculation using Julia:

```
A = rand(2,2)
a,c,b,d = A
u = sqrt(a^2+c^2)
v = (a*b+c*d)/u
w = (a*d-b*c)/u
display(A)
display( [a/u -c/u;c/u a/u]*[u v; 0 w])
```

```
2×2 Array{Float64,2}:
 0.805679  0.139147
 0.49966   0.66488

2×2 Array{Float64,2}:
 0.805679  0.139147
 0.49966   0.66488
```

The 2x2 svd

You will not be asked to memorize the svd formula or compute by hand. Later we will see larger svd's. Please do not wikipedia svd, there is too much there for reading at this point and it will be overwhelming.

In 2d, the SVD says every linear transformation can be factored into

A = (rotation) x (scaling) x (rotation)

$$A = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \cdot \begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix}$$

The letter σ ("sigma") is used for the diagonal scaling (σ_1 and σ_2)

Formula for the svd

See Lulu's blog for a way to calculate the 2x2 svd

It uses the atan2 arctan function to pick out the right quadrant.

In practice, svd's are widely available in packages such as Julia, NumPy, mathematica, etc, and so nobody memorizes the formulas.

While mathematically correct, we are not in favor of the approach used in some classes that start with eigenvalues, and move to svd's as an afterthought.

Matrix Multiply as an algorithm

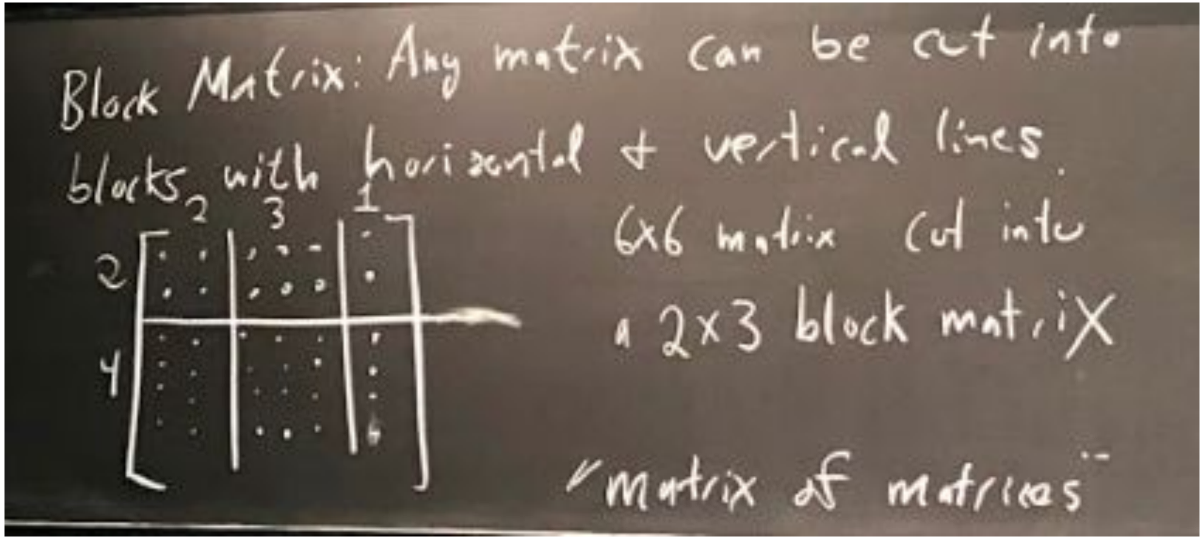
Assume A,B nxn

```
C = fill(0,n,n)  # make an nxn matrix of zeros

for i=1:n, j=1:n, k=1:n

   C[i,j] += A[i,k] * B[k,j]  # add $A_{ik}B_{kj}$ to the current value of $C_{ij}$

end
```

Think about why the i,j,k loops can be permuted. How are the dot products being filled in different orders, or perhaps being partially computed and then returned to.

4. Block Matrices (covered in recitation, not lecture)

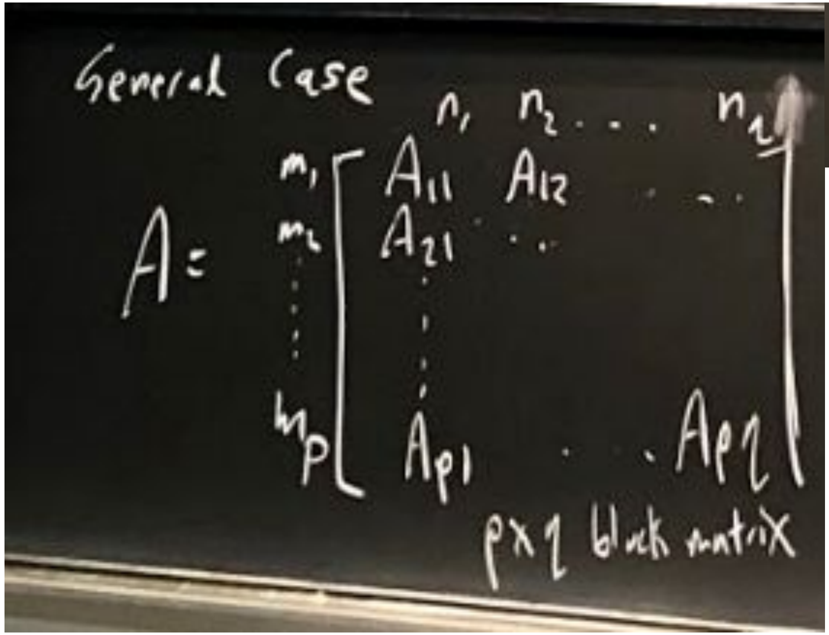
Leads to many new views of matrix multiply. Fuels the matrix multiplies used by scientific applications and machine learning on GPUs (graphics processing units)


Block Matrix: Any matrix can be cut into blocks, with horizontal & vertical lines.

6x6 matrix cut into a 2x3 block matrix

"matrix of matrices"

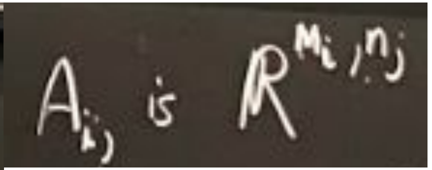
General Case of a p x q block matrix



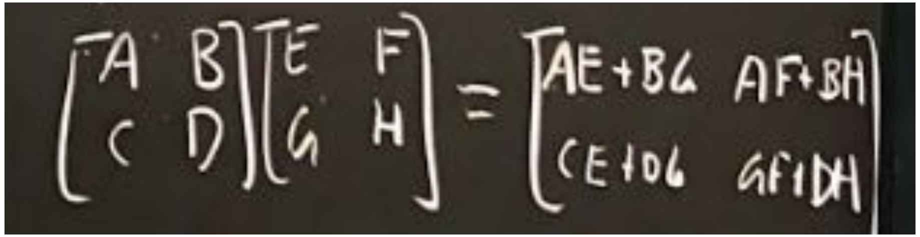
General Case

$$A = \begin{array}{c|cccc} & n_1 & n_2 & \cdots & n_q \\ \hline m_1 & A_{11} & A_{12} & \cdots & \cdots \\ m_2 & A_{21} & \cdots & & \\ \vdots & \vdots & & & \\ m_p & A_{p1} & \cdots & \cdots & A_{pq} \end{array}$$

p x q block matrix



A_{ij} is $\mathbb{R}^{m_i, n_j}$

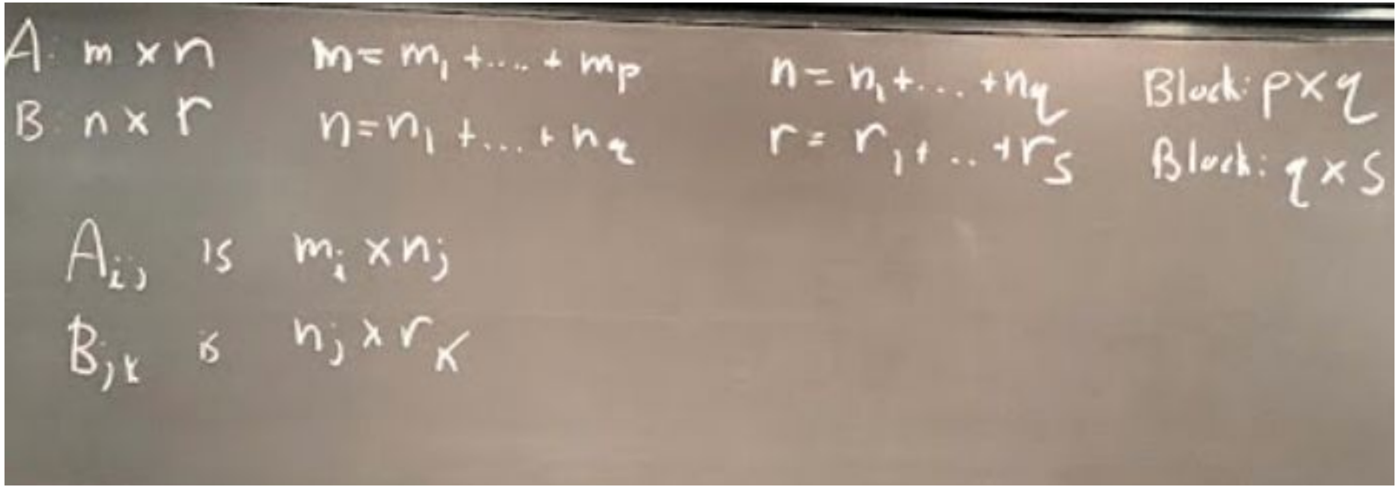


$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE+BG & AF+BH \\ CE+DG & GF+DH \end{bmatrix}$$

Example Block Matrix Multiply

When compatible , the matrix multiply always works! (Next slide spells out what compatible means)

Compatibility Conditions



$A: m \times n$ $\quad m = m_1 + \dots + m_p \quad n = n_1 + \dots + n_q \quad$ Block: $p \times q$

$B: n \times r$ $\quad n = n_1 + \dots + n_q \quad r = r_1 + \dots + r_s \quad$ Block: $q \times s$

A_{ij} is $m_i \times n_j$

B_{jk} is $n_j \times r_k$

Concrete Example

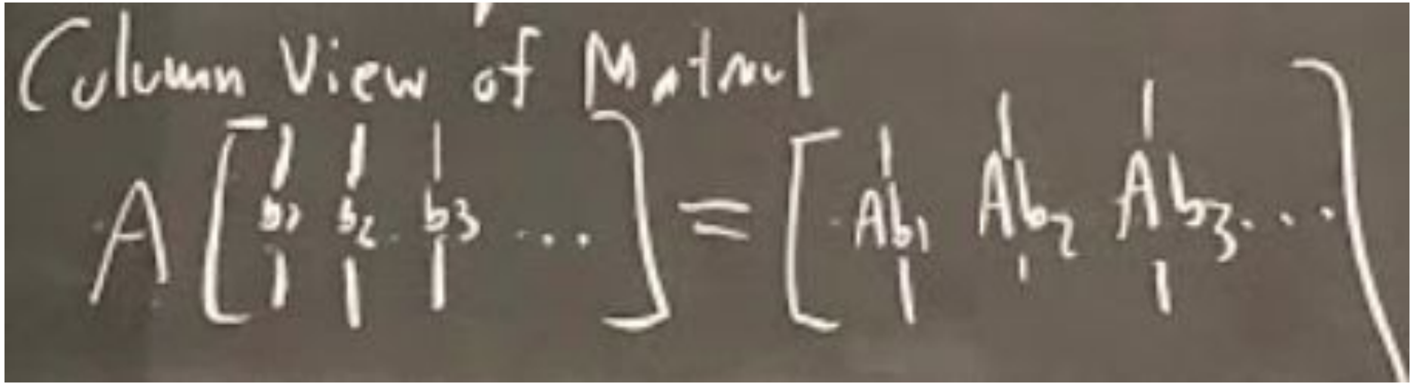
$$A = \left[\begin{array}{cc|cc} 1 & 2 & 0 & 1 \\ 3 & 4 & 0 & 0 \\ \hline 2 & 0 & 1 & 0 \\ 0 & 3 & 0 & 1 \end{array}\right] \qquad B = \left[\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{array}\right]$$

$$AB = \begin{array}{cc|cc} 0 & 1 & 2 & 3 \\ 0 & 0 & 3 & 4 \\ \hline 1 & 0 & 3 & 1 \\ 0 & 1 & 1 & 4 \end{array}$$

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} \times \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} + \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix} \times \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} = \begin{matrix} 0 & 1 \\ 0 & 0 \end{matrix}$$

Etc. Do check the other 2x2 ! (I blew it in class)

Column view of matmul is a special case



Column View of Matmul

$$A\begin{bmatrix} | & | & | & \\ b_1 & b_2 & b_3 & \cdots \\ | & | & | & \end{bmatrix} = \begin{bmatrix} | & | & | & \\ Ab_1 & Ab_2 & Ab_3 & \cdots \\ | & | & | & \end{bmatrix}$$

A is block 1 x 1 (A is made up of one m x n matrix)
B is block 1 x r   (B is made up of r columns of size n)

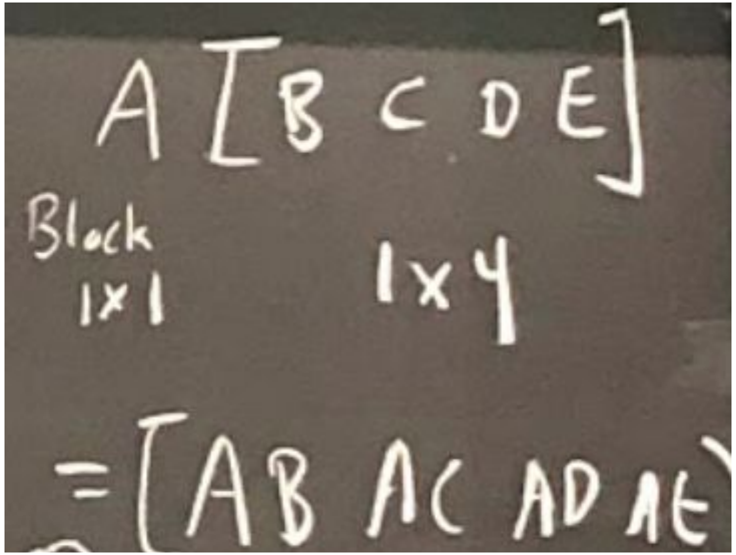
=

A*B is block 1 x r

So is the row view

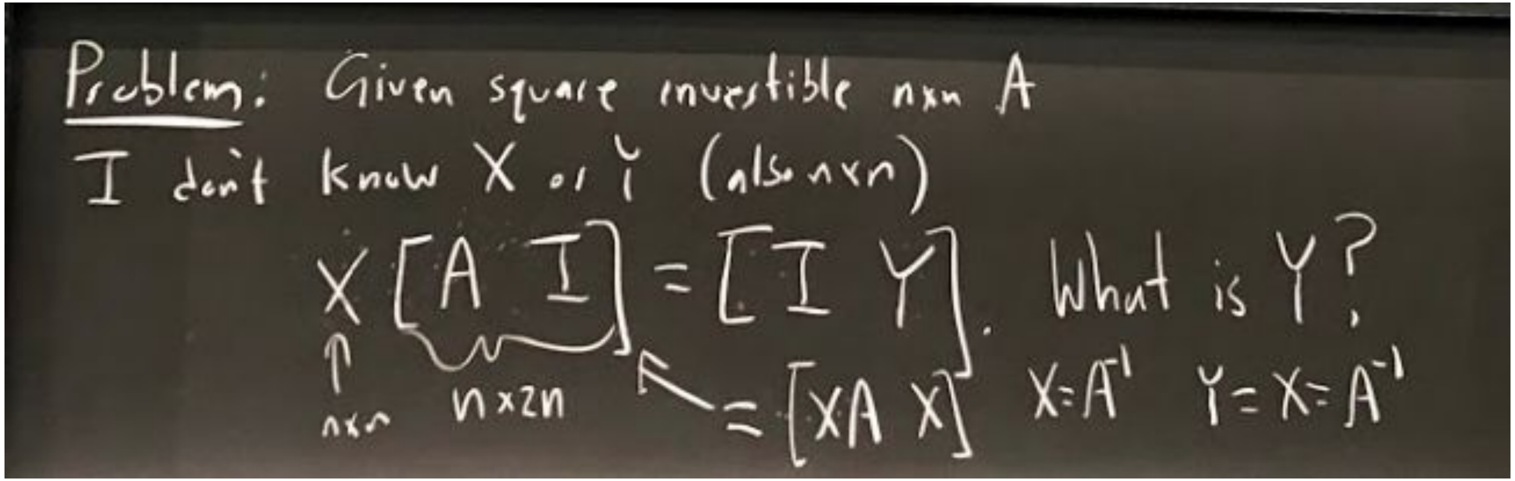
$$\begin{bmatrix} — a_1 — \\ — a_2 — \\ — a_3 — \end{bmatrix} B = \begin{bmatrix} a_1 B \\ a_2 B \\ \vdots \end{bmatrix}$$

Another Example



In general, A [B C D E F … G] = [AB AC AD AE AF … AG]

Application



<u>Problem:</u> Given square invertible $n \times n$ A

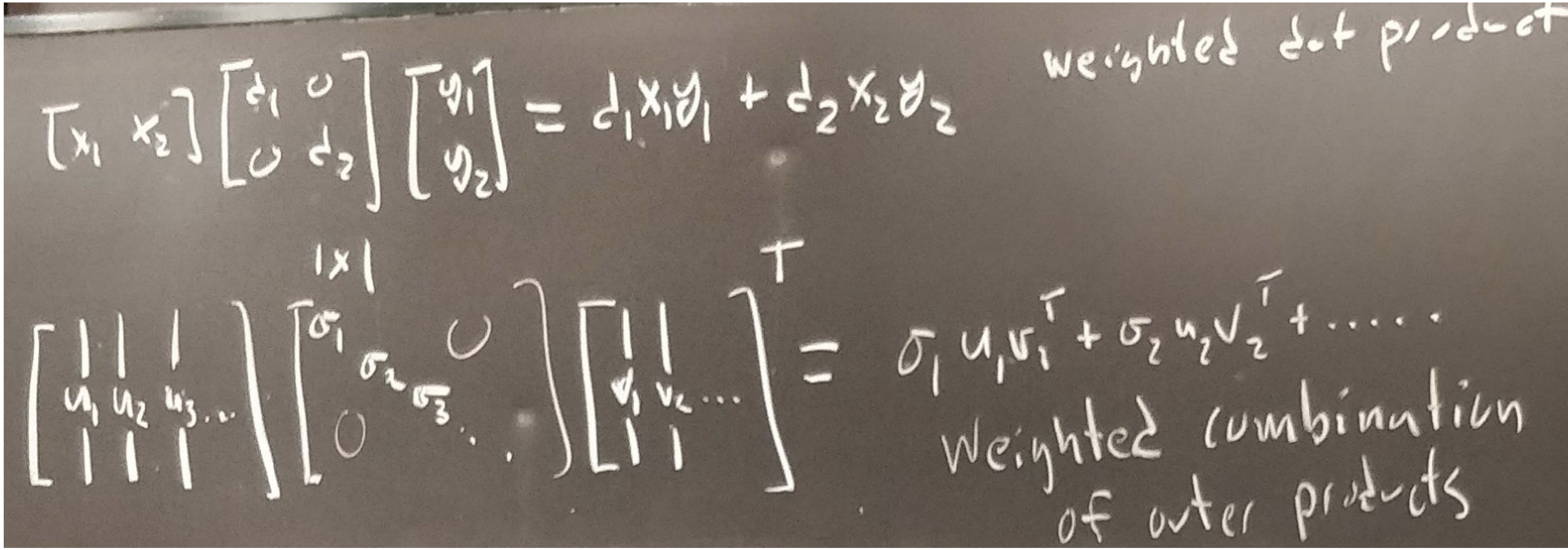
I don't know X or Y (also $n \times n$)

$$\underset{n\times n}{\underset{\uparrow}{X}}\underbrace{[A \;\; I]}_{n\times 2n} = [I \;\; Y]. \quad \text{What is } Y?$$

$$= [XA \;\; X] \qquad X = A^{-1} \quad Y = X = A^{-1}$$

This idea underlies a famous pencil and paper method (not used on computers) called Gauss-Jordan which we are not covering at this time but some of you may have seen.

Weighted dot products and weighted combinations of outer products



$$\begin{bmatrix} x_1 & x_2 \end{bmatrix}\begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = d_1 x_1 y_1 + d_2 x_2 y_2$$

weighted dot product

1×1

$$\begin{bmatrix} | & | & | & \\ u_1 & u_2 & u_3 & \dots \\ | & | & | & \end{bmatrix}\begin{bmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \sigma_3 & \\ 0 & & & \ddots \end{bmatrix}\begin{bmatrix} | & | & \\ v_1 & v_2 & \dots \\ | & | & \end{bmatrix}^T = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots\dots$$

weighted combination of outer products

A diagonal matrix might be written as Diagonal([d_1,d_2]) or Diagonal([σ_1,σ_2,σ_3,...])

When might you use a weighted dot product?

Suppose you were buying 100 European Widgets at €7 each and 200 British Gadgets at £56 each, and 34 Japanese Mechanisms at ¥2300 each how much is the total cost in dollars?

Answer: x = [100 200 34] y=[€7 £56 ¥2300]’ and D=the diagonal matrix of today’s exchange rates: D=diagonal([1.13 $/€ , 1.29 $/£, .0090 $/¥])

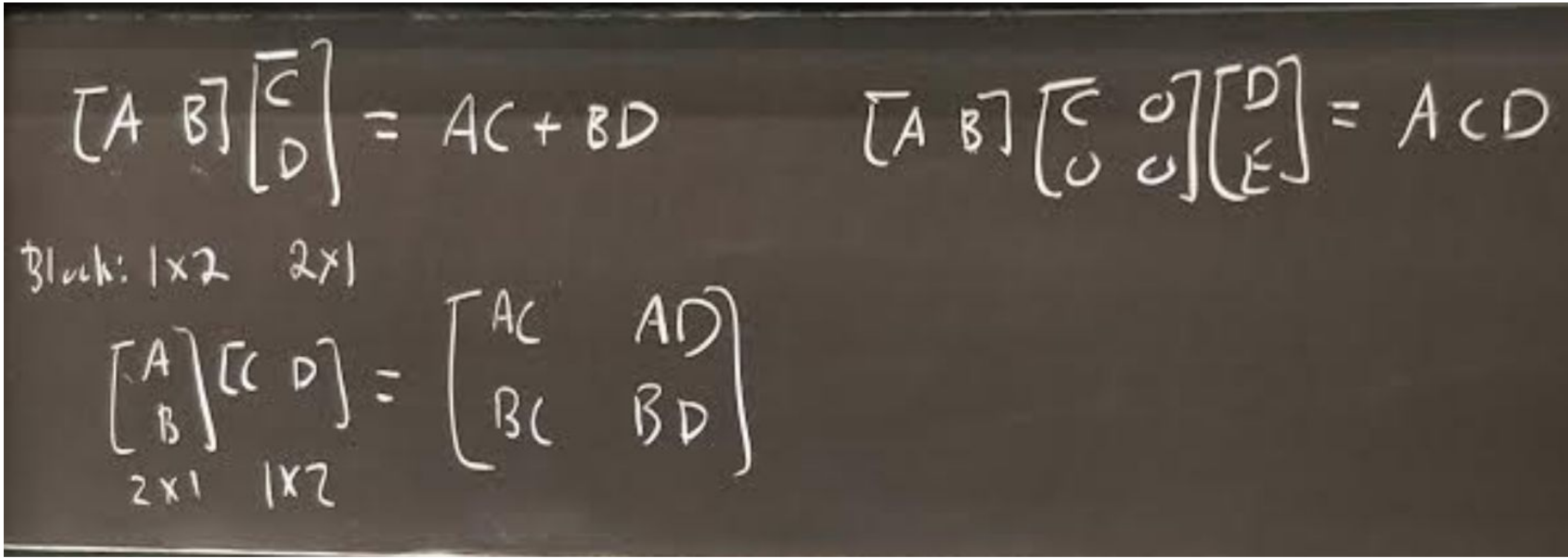
x * D * y is the dollar cost for the whole shopping cart

When might you use a weighted sum of outer products?

Compression -- an approximation that can be very powerful is to take the outer products corresponding to the largest weights.   It can be more art than science to decide which ones, but nonetheless it can be very effective.

In a few lectures, when we do the svd, you will see an example of image compression using this very approach.

Further Applications



$$\begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = AC + BD$$

Block: 1×2 $\quad 2 \times 1$

$$\begin{bmatrix} A \\ B \end{bmatrix} \begin{bmatrix} C & D \end{bmatrix} = \begin{bmatrix} AC & AD \\ BC & BD \end{bmatrix}$$

2×1 $\quad 1 \times 2$

$$\begin{bmatrix} A & B \end{bmatrix} \begin{bmatrix} C & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} D \\ E \end{bmatrix} = ACD$$

Factorizations We will See

	Solve Linear Systems	Least Squares	Data Compression and Other Applications and Theory
LU (lower times upper)	✓	❌	❌
QR (orthogonal times upper)	✓	✓	❌
SVD (U Diag(s) V’) Singular value decomposition	✓	✓	✓