

CSE 548: Analysis of Algorithms

Lecture 3

(Divide-and-Conquer Algorithms: Matrix Multiplication)

Rezaul A. Chowdhury

Department of Computer Science

SUNY Stony Brook

Fall 2016

Iterative Matrix Multiplication

$$z_{ij} = \sum_{k=1}^n x_{ik} y_{kj}$$

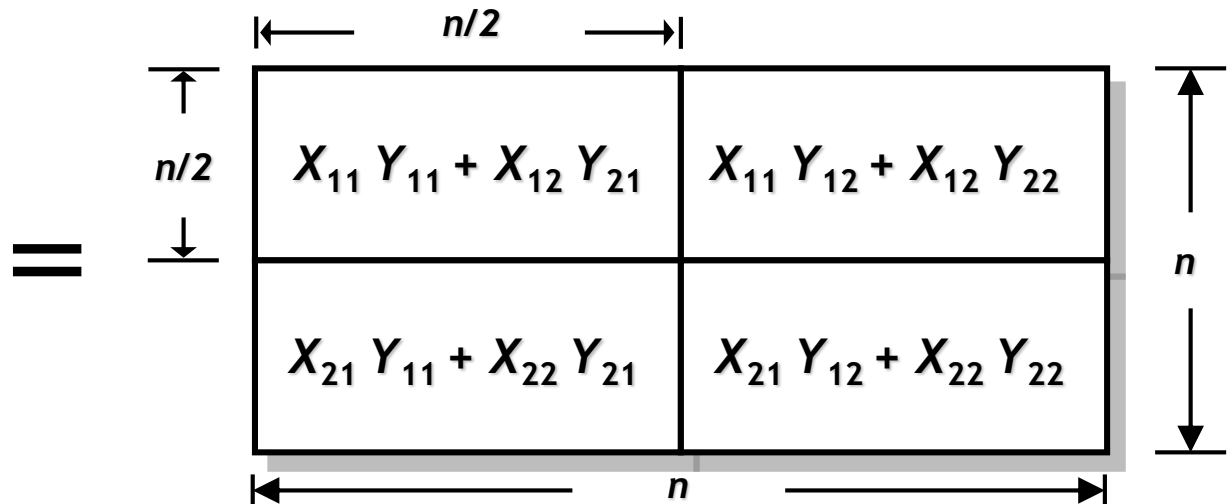
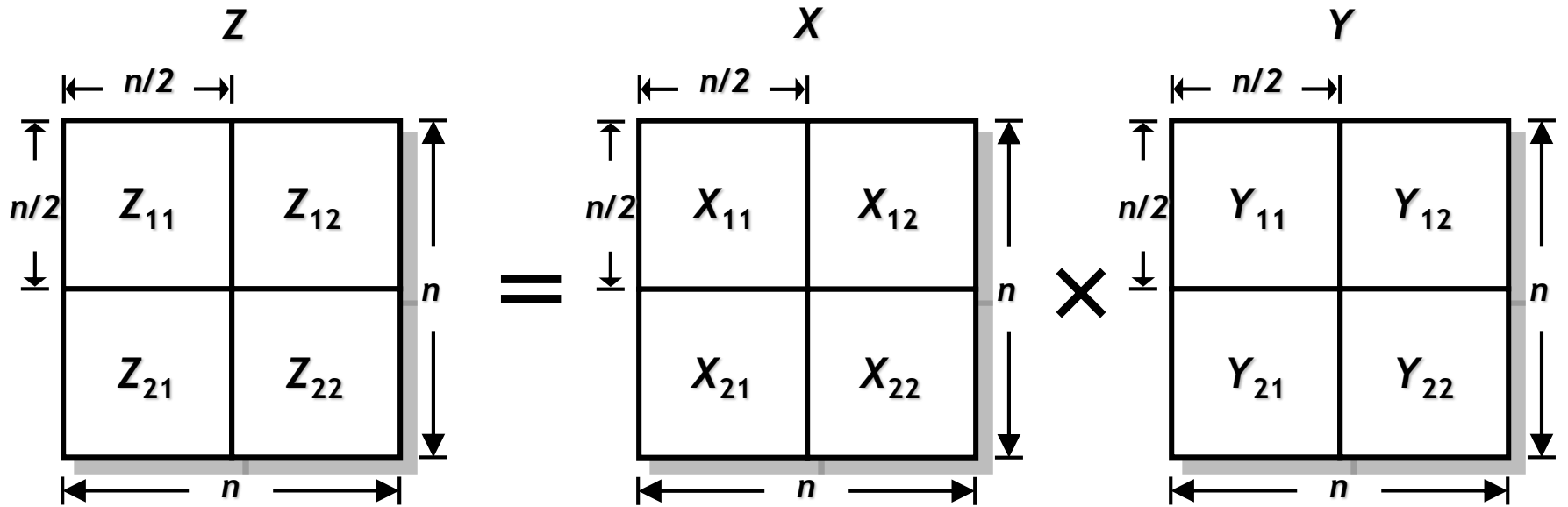
$$\begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & z_{22} & \cdots & z_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \cdots & z_{nn} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix} \times \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{nn} \end{bmatrix}$$

Iter-MM (Z, X, Y)

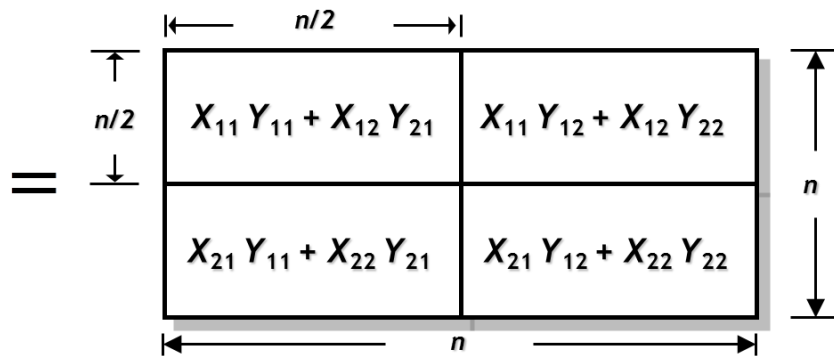
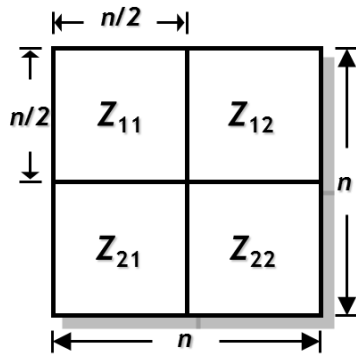
{ X, Y, Z are $n \times n$ matrices,
where n is a positive integer }

1. *for* $i \leftarrow 1$ *to* n *do*
2. *for* $j \leftarrow 1$ *to* n *do*
3. $Z[i][j] \leftarrow 0$
4. *for* $k \leftarrow 1$ *to* n *do*
5. $Z[i][j] \leftarrow Z[i][j] + X[i][k] \cdot Y[k][j]$

Recursive (Divide & Conquer) Matrix Multiplication



Recursive (Divide & Conquer) Matrix Multiplication



Rec-MM (X, Y) { X and Y are $n \times n$ matrices,
where $n = 2^k$ for integer $k \geq 0$ }

1. Let Z be a new $n \times n$ matrix
2. if $n = 1$ then
3. $Z \leftarrow X \cdot Y$
4. else
5. $Z_{11} \leftarrow \text{Rec-MM} (X_{11}, Y_{11}) + \text{Rec-MM} (X_{12}, Y_{21})$
6. $Z_{12} \leftarrow \text{Rec-MM} (X_{11}, Y_{12}) + \text{Rec-MM} (X_{12}, Y_{22})$
7. $Z_{21} \leftarrow \text{Rec-MM} (X_{21}, Y_{11}) + \text{Rec-MM} (X_{22}, Y_{21})$
8. $Z_{22} \leftarrow \text{Rec-MM} (X_{21}, Y_{12}) + \text{Rec-MM} (X_{22}, Y_{22})$
9. endif
10. return Z

recursive matrix products: 8

matrix sums: 4

$$T(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ 8T\left(\frac{n}{2}\right) + \Theta(n^2), & \text{otherwise.} \end{cases}$$

$$= \Theta(n^3)$$

Strassen's Algorithms for Matrix Multiplication (MM)

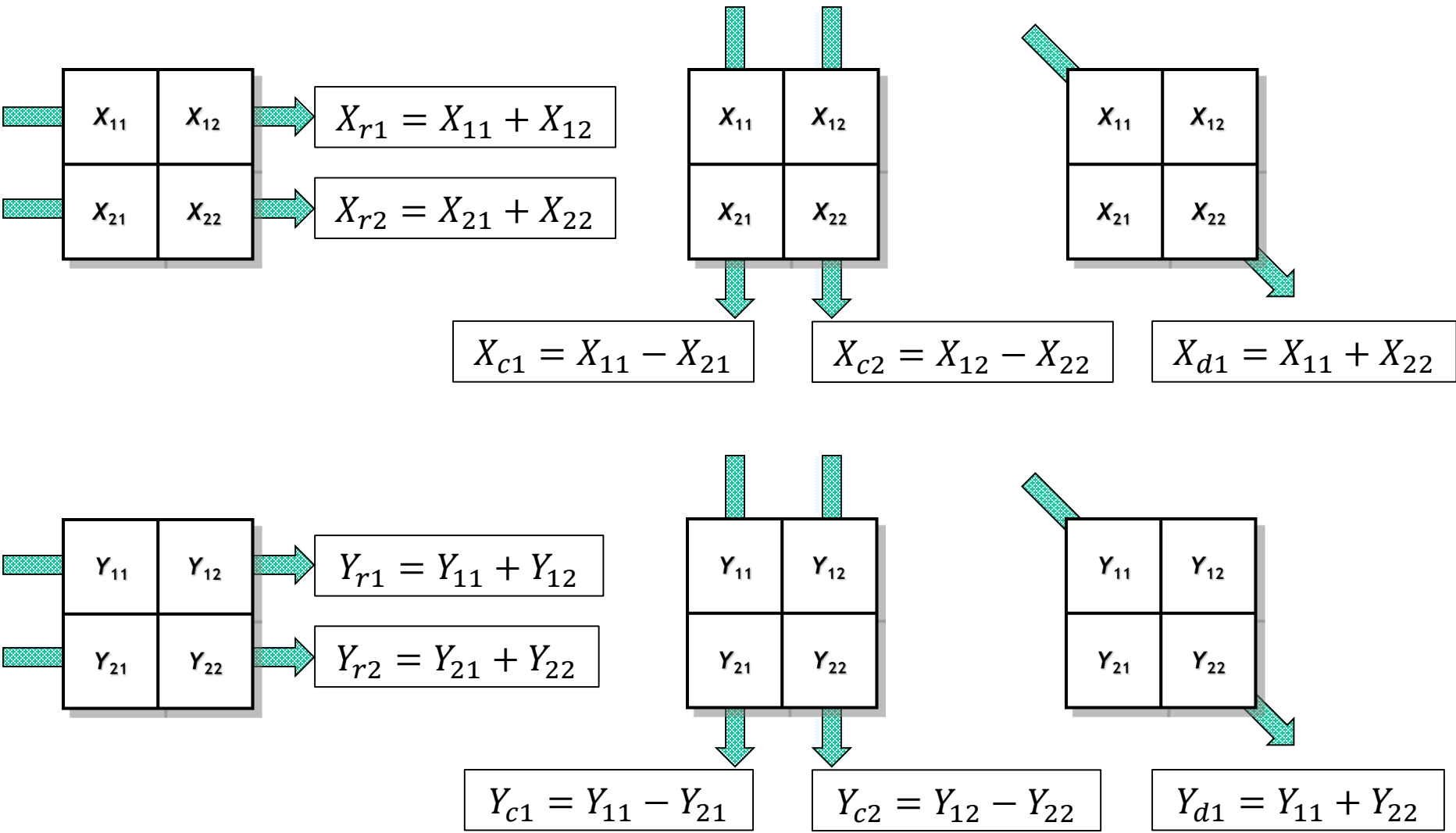


In 1968 Volker Strassen came up with a recursive MM algorithm that runs asymptotically faster than the classical $\Theta(n^3)$ algorithm.

In each level of recursion the algorithm uses:

7 recursive matrix multiplications (instead to 8), and
18 matrix additions (instead of 4).

Strassen's MM: 10 Matrix Additions/Subtractions



Strassen's MM: 7 Matrix Products

X-cell • Y-col

	X ₁₁	X ₁₂	X ₂₁	X ₂₂
Y ₁₁				
Y ₂₁				
Y ₁₂	+			
Y ₂₂	-			

$$P_{11} = X_{11} \cdot Y_{c2}$$

	X ₁₁	X ₁₂	X ₂₁	X ₂₂
Y ₁₁				+
Y ₂₁				-
Y ₁₂				
Y ₂₂				

$$P_{22} = X_{22} \cdot Y_{c1}$$

X-row • Y-cell

	X ₁₁	X ₁₂	X ₂₁	X ₂₂
Y ₁₁				
Y ₂₁				
Y ₁₂				
Y ₂₂	+	+		

$$P_{r1} = X_{r1} \cdot Y_{22}$$

	X ₁₁	X ₁₂	X ₂₁	X ₂₂
Y ₁₁			+	+
Y ₂₁				
Y ₁₂				
Y ₂₂				

$$P_{r2} = X_{r2} \cdot Y_{11}$$

X-col • Y-row

	X ₁₁	X ₁₂	X ₂₁	X ₂₂
Y ₁₁	+		-	
Y ₂₁				
Y ₁₂	+		-	
Y ₂₂				

$$P_{c1} = X_{c1} \cdot Y_{r1}$$

	X ₁₁	X ₁₂	X ₂₁	X ₂₂
Y ₁₁				
Y ₂₁		+		-
Y ₁₂				
Y ₂₂		+		-

$$P_{c2} = X_{c2} \cdot Y_{r2}$$

X-diag • Y-diag

	X ₁₁	X ₁₂	X ₂₁	X ₂₂
Y ₁₁	+			+
Y ₂₁				
Y ₁₂				
Y ₂₂	+			+

$$P_{d1} = X_{d1} \cdot Y_{d1}$$

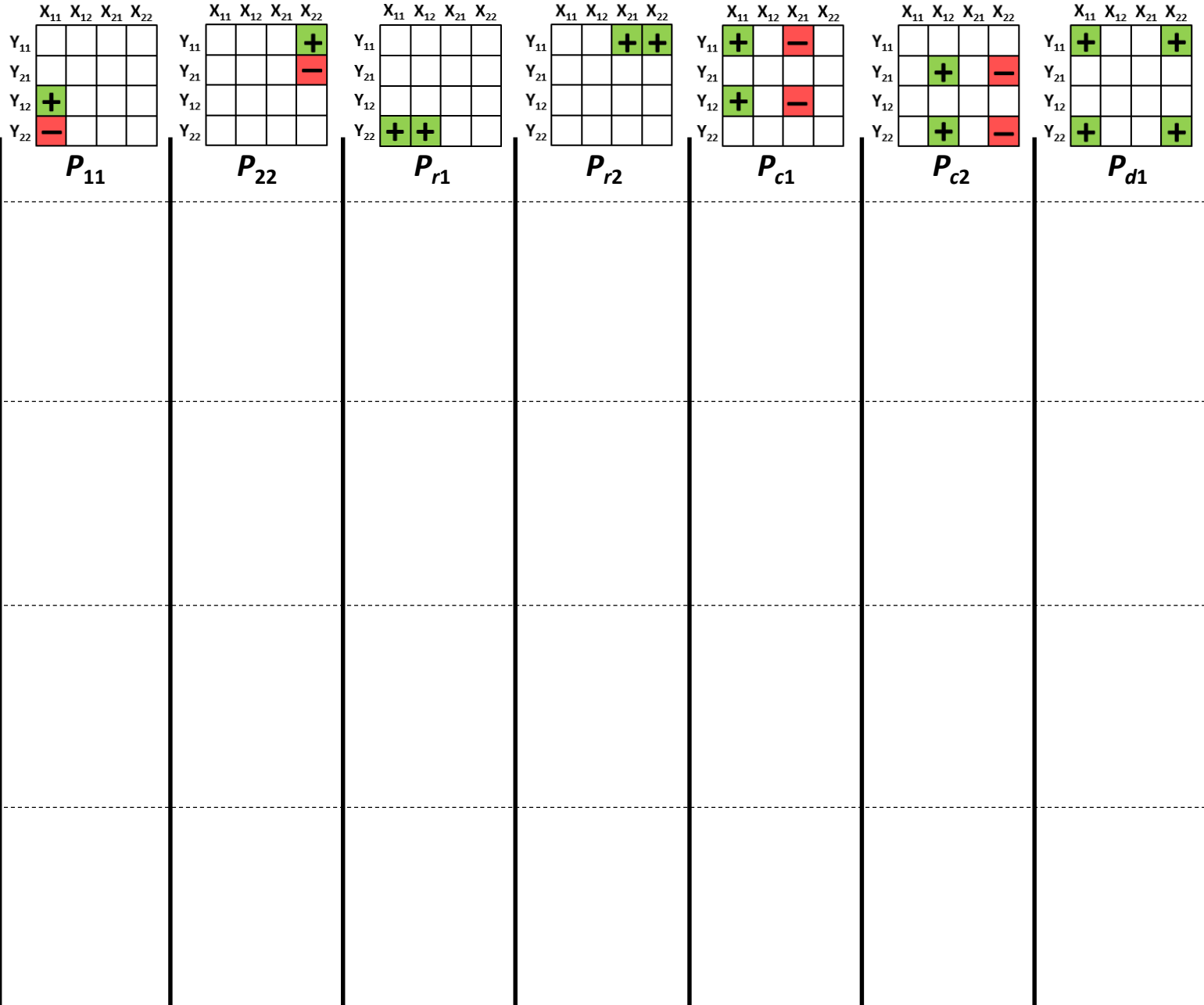
Strassen's MM: 8 More Matrix Additions/Subtractions

$$\begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \times \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}$$

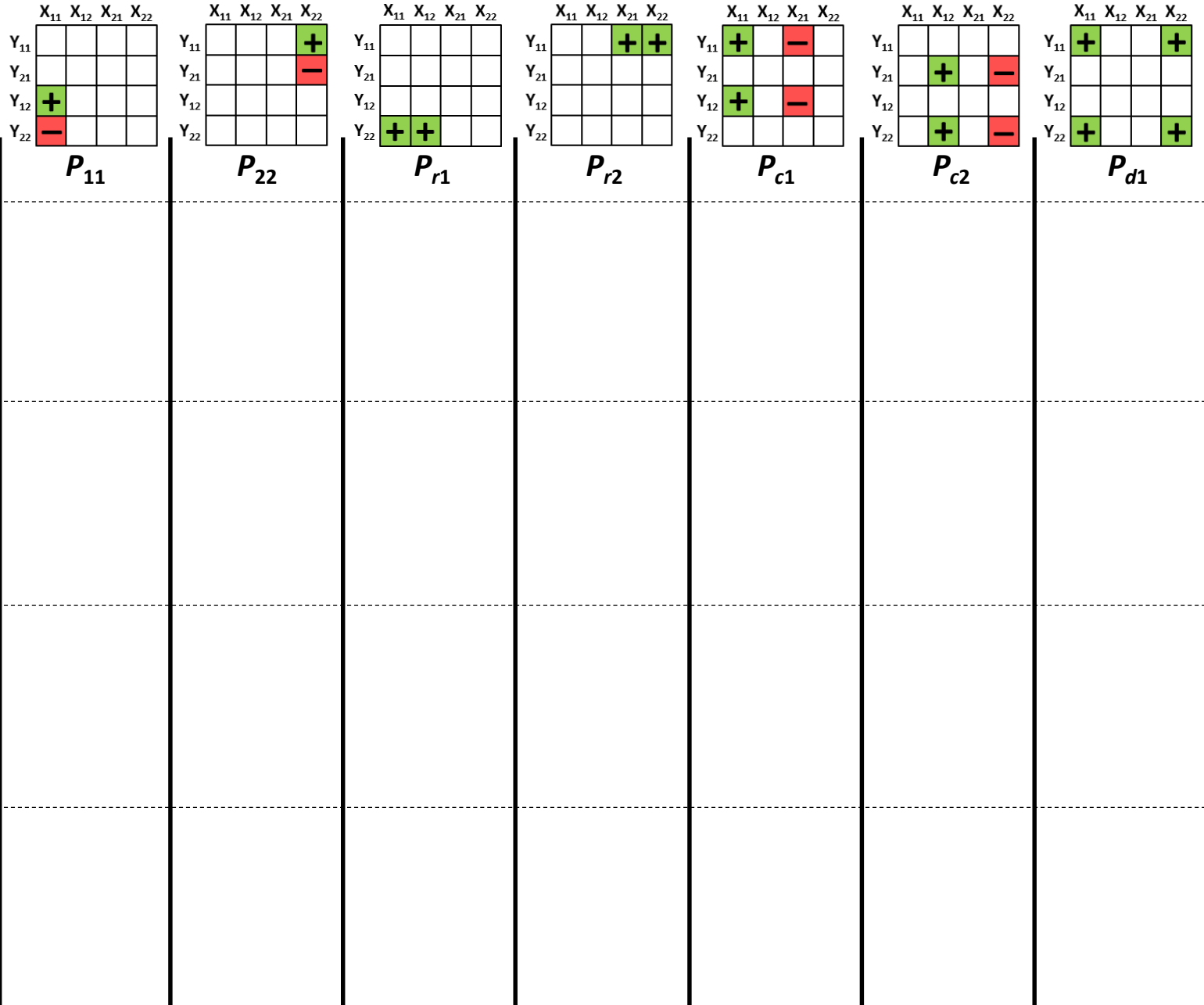
$$= \begin{bmatrix} X_{11} Y_{11} + X_{12} Y_{21} & X_{11} Y_{12} + X_{12} Y_{22} \\ X_{21} Y_{11} + X_{22} Y_{21} & X_{21} Y_{12} + X_{22} Y_{22} \end{bmatrix}$$

$$= \begin{bmatrix} -P_{r1} & & -P_{22} & +P_{r1} & & +P_{11} \\ +P_{d1} & & +P_{c2} & & & \\ +P_{r2} & & -P_{22} & -P_{r2} & & +P_{11} \\ & & & +P_{d1} & & -P_{c1} \end{bmatrix}$$

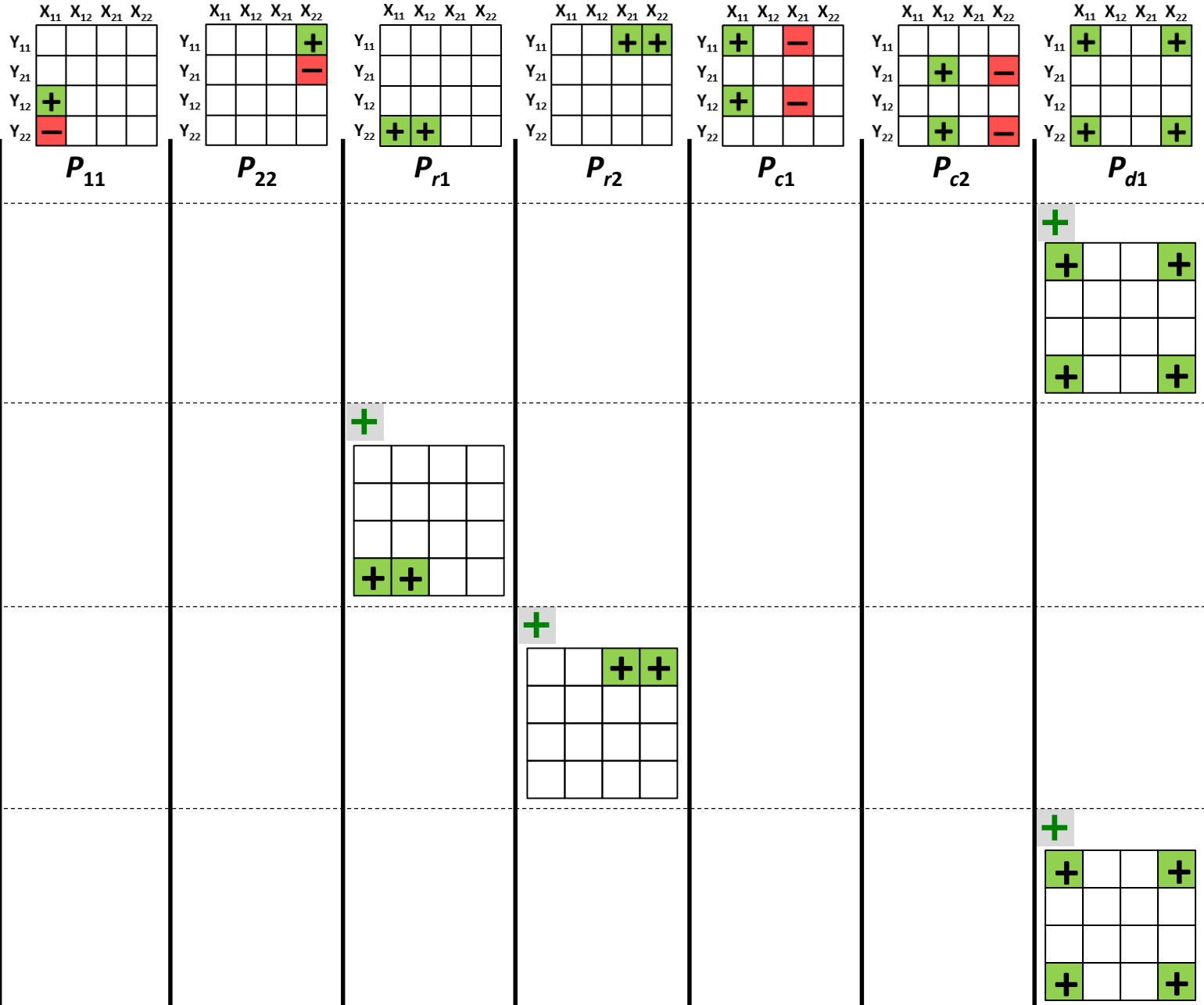
Strassen's Matrix Multiplication



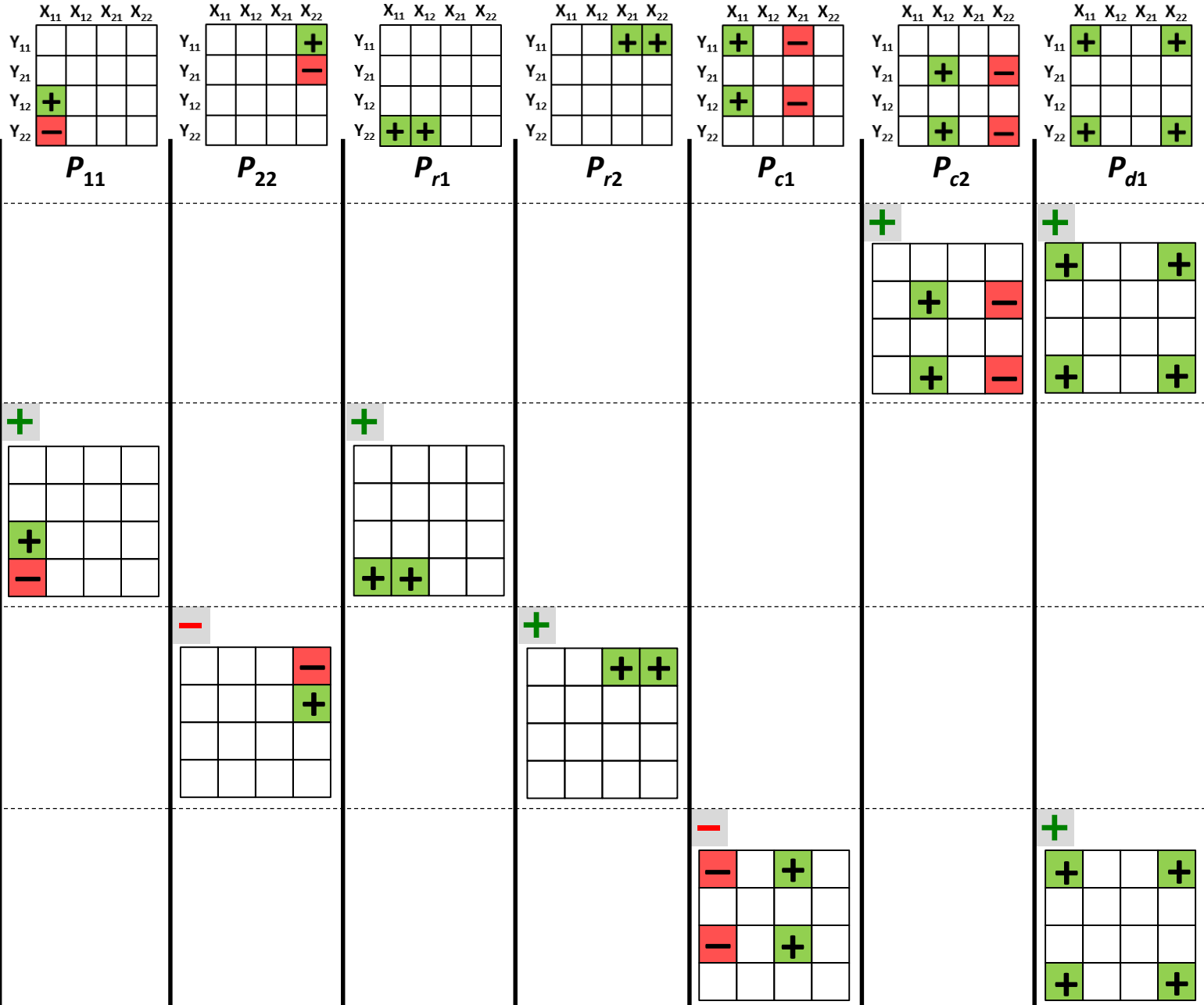
Strassen's Matrix Multiplication



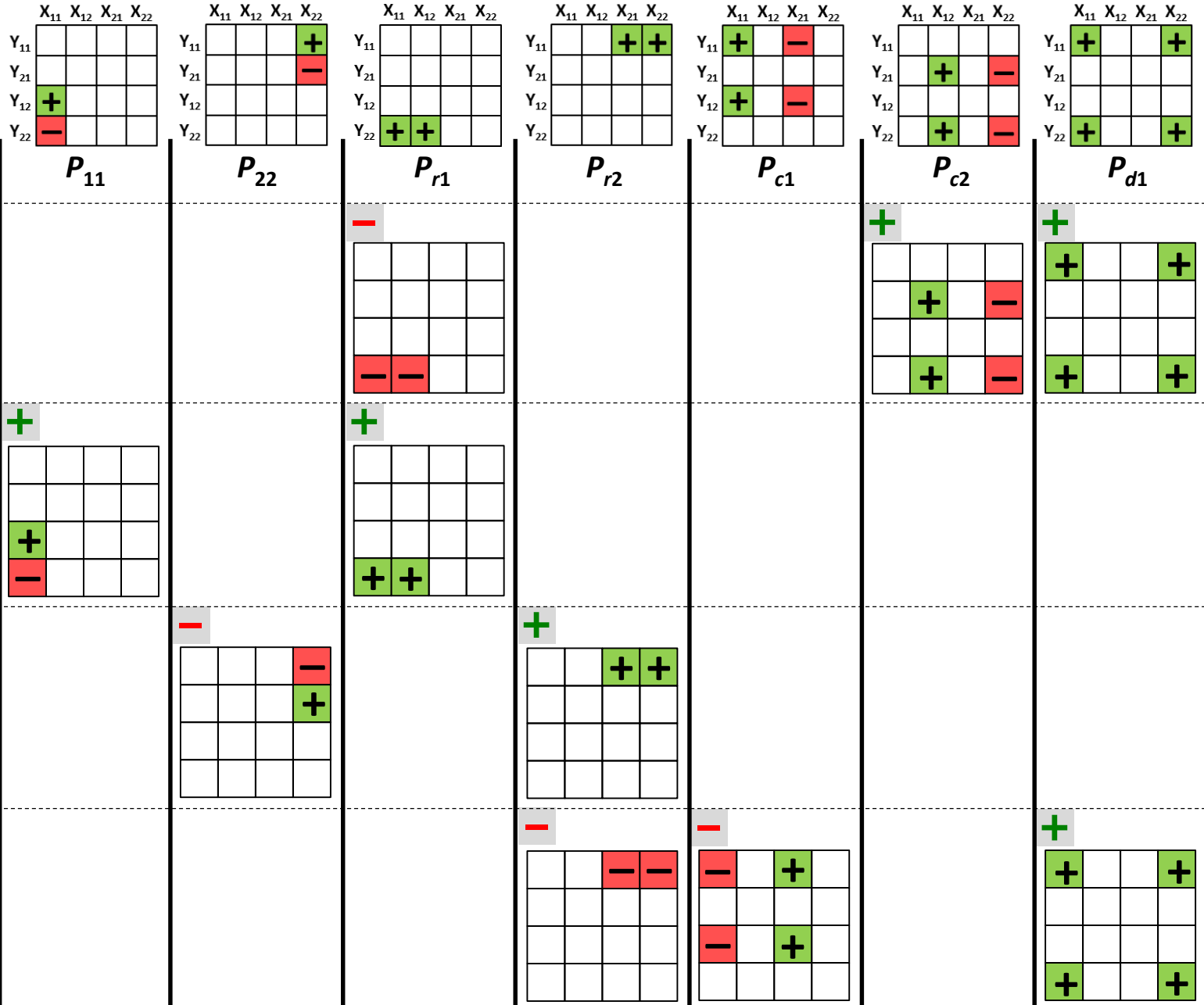
Strassen's Matrix Multiplication



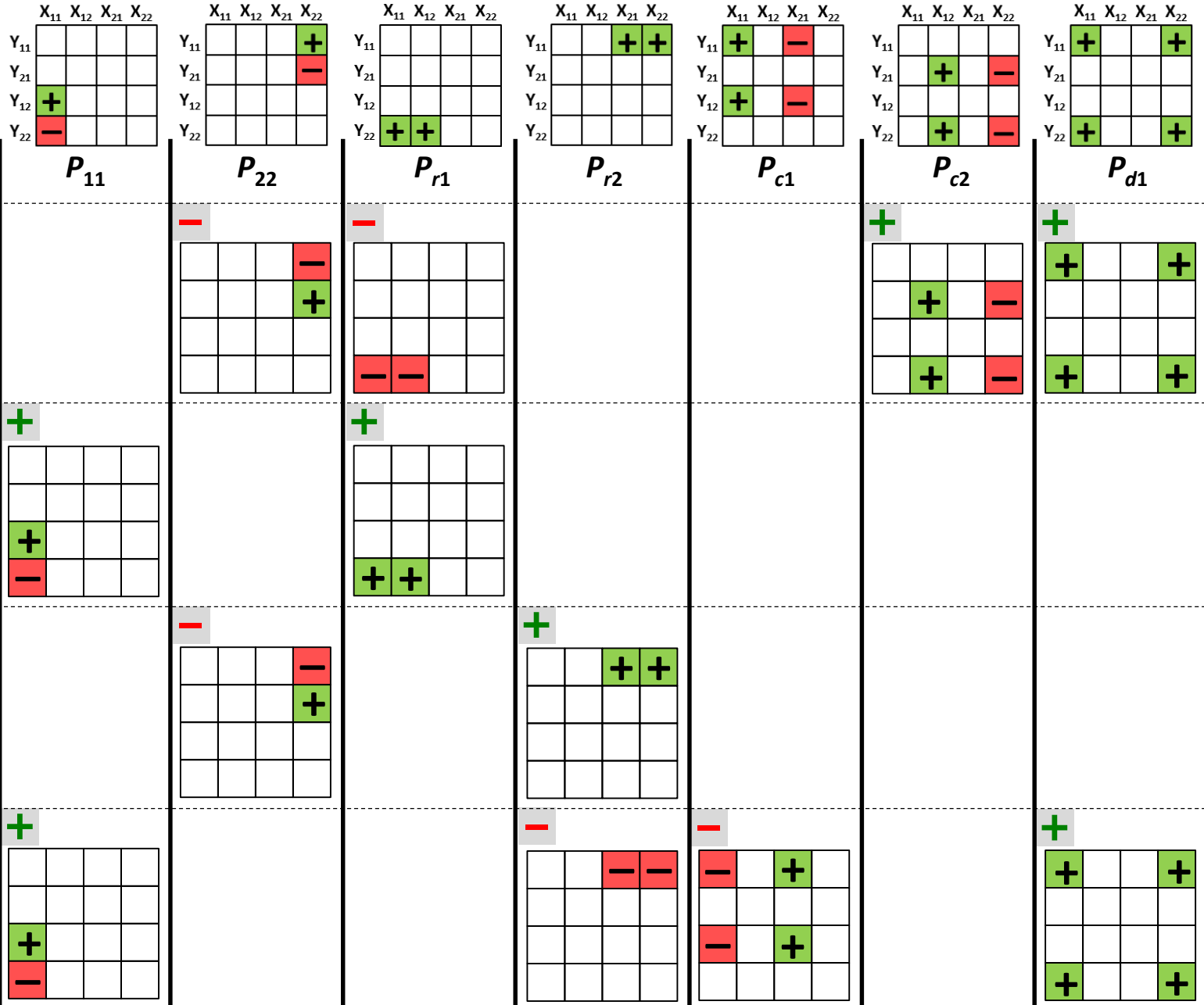
Strassen's Matrix Multiplication



Strassen's Matrix Multiplication



Strassen's Matrix Multiplication



Strassen's Matrix Multiplication

$$\begin{array}{|c|c|} \hline Z_{11} & Z_{12} \\ \hline Z_{21} & Z_{22} \\ \hline \end{array} = \begin{array}{|c|c|} \hline X_{11} & X_{12} \\ \hline X_{21} & X_{22} \\ \hline \end{array} \times \begin{array}{|c|c|} \hline Y_{11} & Y_{12} \\ \hline Y_{21} & Y_{22} \\ \hline \end{array} = \begin{array}{|c|c|} \hline X_{11} Y_{11} + X_{12} Y_{21} & X_{11} Y_{12} + X_{12} Y_{22} \\ \hline X_{21} Y_{11} + X_{22} Y_{21} & X_{21} Y_{12} + X_{22} Y_{22} \\ \hline \end{array}$$

Sums:

$$\begin{array}{ll}
 X_{r1} = X_{11} + X_{12} & Y_{r1} = Y_{11} + Y_{12} \\
 X_{r2} = X_{21} + X_{22} & Y_{r2} = Y_{21} + Y_{22} \\
 X_{c1} = X_{11} - X_{21} & Y_{c1} = Y_{11} - Y_{21} \\
 X_{c2} = X_{12} - X_{22} & Y_{c2} = Y_{12} - Y_{22} \\
 X_{d1} = X_{11} + X_{22} & Y_{d1} = Y_{11} + Y_{22}
 \end{array}$$

$$\begin{array}{|c|c|c|c|} \hline -P_{r1} & & -P_{22} & +P_{r1} \\ \hline +P_{d1} & & +P_{c2} & \\ \hline +P_{r2} & & -P_{22} & -P_{r2} \\ \hline & & +P_{d1} & -P_{c1} \\ \hline \end{array}$$

Products:

$$\begin{array}{ll}
 P_{11} = X_{11} \cdot Y_{c2} & P_{c1} = X_{c1} \cdot Y_{r1} \\
 P_{22} = X_{22} \cdot Y_{c1} & P_{c2} = X_{c2} \cdot Y_{r2} \\
 P_{r1} = X_{r1} \cdot Y_{22} & P_{d1} = X_{d1} \cdot Y_{d1} \\
 P_{r2} = X_{r2} \cdot Y_{11} &
 \end{array}$$

Running Time:

$$T(n) = \begin{cases} \Theta(1), & \text{if } n = 1, \\ 7T\left(\frac{n}{2}\right) + \Theta(n^2), & \text{otherwise.} \end{cases}$$

$$= \Theta(n^{\log_2 7}) = O(n^{2.81})$$

Deriving Strassen's Algorithm

Use the *Feynman Algorithm*:

Step 1: write down the problem

Step 2: think real hard

Step 3: write down the solution

Deriving Strassen's Algorithm

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} p & q \\ r & s \end{bmatrix} \Rightarrow \underbrace{\begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}}_X \underbrace{\begin{bmatrix} e \\ g \\ f \\ h \end{bmatrix}}_Y = \underbrace{\begin{bmatrix} p \\ r \\ q \\ s \end{bmatrix}}_Z$$

We will try to minimize the number of multiplications needed to evaluate Z using special matrix products that are easy to compute.

<u>Type</u>	<u>Product</u>	<u>#Mults</u>
(\cdot)	$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e \\ g \end{bmatrix} = \begin{bmatrix} ae + bg \\ ce + dg \end{bmatrix}$	4
(A)	$\begin{bmatrix} a & a \\ a & a \end{bmatrix} \begin{bmatrix} e \\ g \end{bmatrix} = \begin{bmatrix} a(e + g) \\ a(e + g) \end{bmatrix}$	1
(B)	$\begin{bmatrix} a & a \\ -a & -a \end{bmatrix} \begin{bmatrix} e \\ g \end{bmatrix} = \begin{bmatrix} a(e + g) \\ -a(e + g) \end{bmatrix}$	1
(C)	$\begin{bmatrix} a & 0 \\ a - b & b \end{bmatrix} \begin{bmatrix} e \\ g \end{bmatrix} = \begin{bmatrix} ae \\ ae + b(g - e) \end{bmatrix}$	2
(D)	$\begin{bmatrix} a & b - a \\ 0 & b \end{bmatrix} \begin{bmatrix} e \\ g \end{bmatrix} = \begin{bmatrix} a(e - g) + bf \\ bf \end{bmatrix}$	2

Deriving Strassen's Algorithm

$$\begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} = \underbrace{\begin{bmatrix} b & b & 0 & 0 \\ b & b & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{Type A (1 Mult)}} + \underbrace{\begin{bmatrix} a-b & 0 & 0 & 0 \\ c-b & d-b & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}}_{\Delta_1}$$

$$\Delta_1 = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & c & c \\ 0 & 0 & c & c \end{bmatrix}}_{\text{Type A (1 Mult)}} + \underbrace{\begin{bmatrix} a-b & 0 & 0 & 0 \\ c-b & d-b & 0 & 0 \\ 0 & 0 & a-c & b-c \\ 0 & 0 & 0 & d-c \end{bmatrix}}_{\Delta_2}$$

$$\Delta_2 = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ c-b & 0 & 0 & c-b \\ -(c-b) & 0 & 0 & -(c-b) \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{Type B (1 Mult)}} + \underbrace{\begin{bmatrix} a-b & 0 & 0 & 0 \\ 0 & d-b & 0 & b-c \\ c-b & 0 & a-c & 0 \\ 0 & 0 & 0 & d-c \end{bmatrix}}_{\Delta_3}$$

$$\Delta_3 = \underbrace{\begin{bmatrix} a-b & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ (a-b) - (a-c) & 0 & a-c & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{Type C (2 Mult)}} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & d-b & 0 & (d-c) - (d-b) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & d-c \end{bmatrix}}_{\text{Type D (2 Mult)}}$$

Algorithms for Multiplying Two $n \times n$ Matrices

A recursive algorithm based on multiplying two $m \times m$ matrices using k multiplications will yield an $O(n^{\log_m k})$ algorithm.

To beat Strassen's algorithm: $\log_m k < \log_2 7 \Rightarrow k < m^{\log_2 7}$.

So, for a 3×3 matrix, we must have: $k < 3^{\log_2 7} < 22$.

But the best known algorithm uses 23 multiplications!

Inventor	Year	Complexity
Classical	—	$\Theta(n^3)$
Volker Strassen	1968	$\Theta(n^{2.807})$
Victor Pan (multiply two 70×70 matrices using 143,640 multiplications)	1978	$\Theta(n^{2.795})$
Don Coppersmith & Shmuel Winograd (arithmetic progressions)	1990	$\Theta(n^{2.3737})$
Andrew Stothers	2010	$\Theta(n^{2.3736})$
Virginia Williams	2011	$\Theta(n^{2.3727})$

Lower bound: $\Omega(n^2)$ (why?)