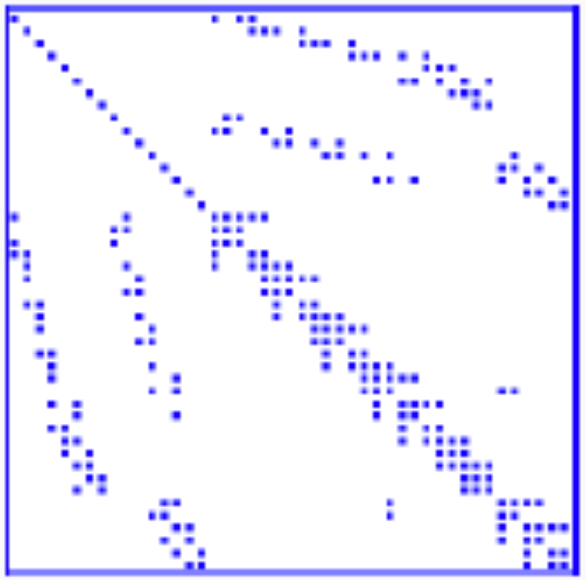
Sparse Systems

Sparse Matrices

Some type of matrices contain many zeros.
Storing all those zero entries is wasteful!

How can we efficiently store large
matrices without storing tons of zeros?

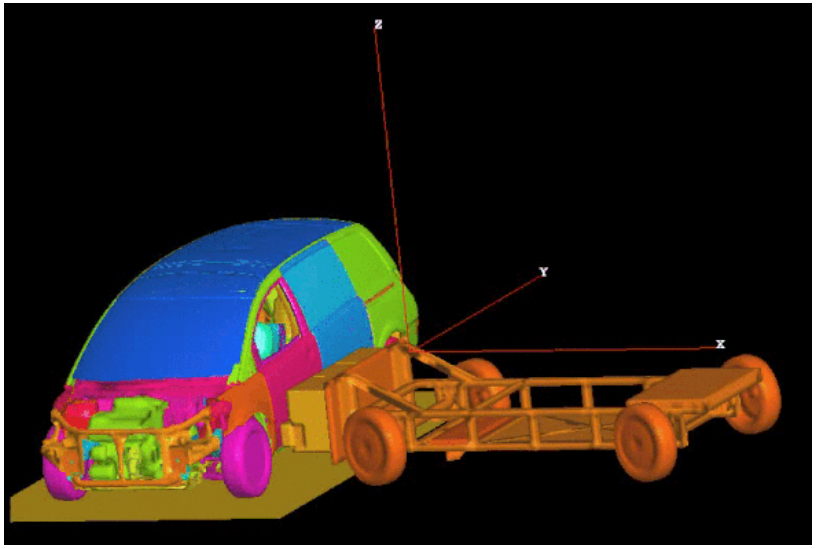


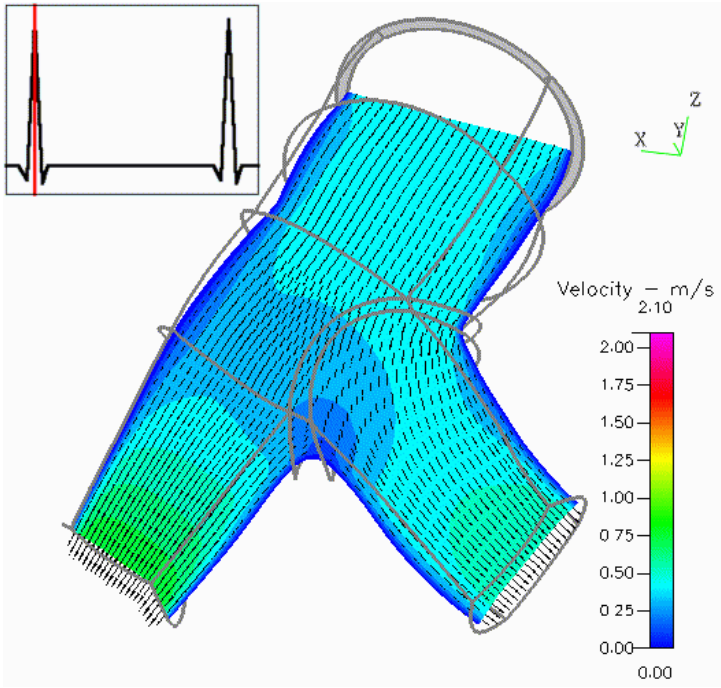
- **Sparse matrices** (vague definition): matrix with few non-zero entries.
- For practical purposes: an $m \times n$ matrix is sparse if it has $O(\min(m,n))$ non-zero entries.
- This means roughly a constant number of non-zero entries per row and column.
- Another definition: "matrices that allow special techniques to take advantage of the large number of zero elements" (J. Wilkinson)

Sparse Matrices: Goals

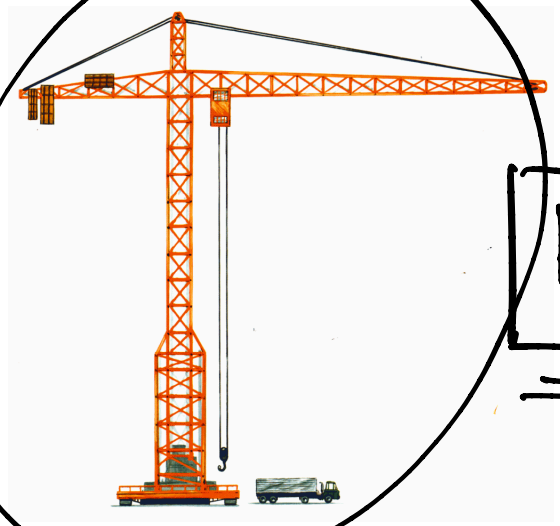
- **Perform standard matrix computations economically, i.e., without storing the zeros of the matrix.**

- For typical Finite Element and Finite Difference matrices, the number of non-zero entries is $O(n)$



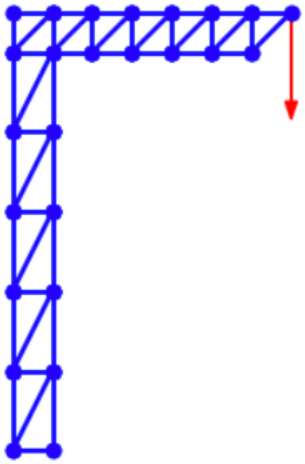


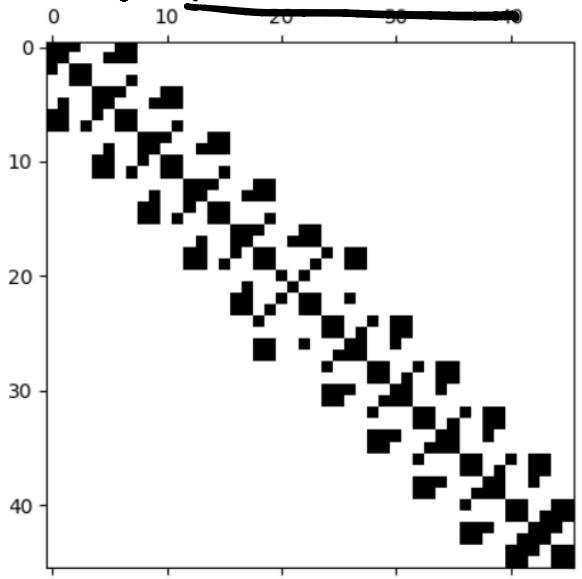
Sparse Matrices: MP example



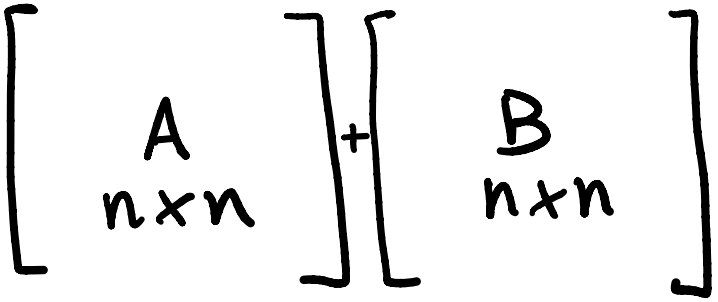
$$K\,U = F$$

spy(K)





Sparse Matrices



EXAMPLE:

Number of operations required to add two square dense matrices:

$$O(n^2)$$

Number of operations required to add two sparse matrices **A** and **B**:

$$O(\text{nnz}(\mathbf{A}) + \text{nnz}(\mathbf{B}))$$

where $\text{nnz}(\mathbf{X}) =$ number of non-zero elements of a matrix $\mathbf{X}$

Popular Storage Structures

DNS	Dense	**ELL**	Ellpack-Itpack
BND	Linpack Banded	**DIA**	Diagonal
COO	Coordinate	**BSR**	Block Sparse Row
CSR	Compressed Sparse Row	**SSK**	Symmetric Skyline
CSC	Compressed Sparse Column	**BSR**	Nonsymmetric Skyline
MSR	Modified CSR	**JAD**	Jagged Diagonal
LIL	Linked List		

note: CSR = CRS, CCS = CSC, SSK = SKS in some references

We will focus on COO and CSR!

Dense (DNS)

$$A = \begin{bmatrix} 0. & 1.9 & 0. & -5.2 \\ 0.3 & 0. & 9.1 & 0. \\ 4.4 & 5.8 & 3.6 & 0. \\ 0. & 0. & 7.2 & 2.7 \end{bmatrix}$$

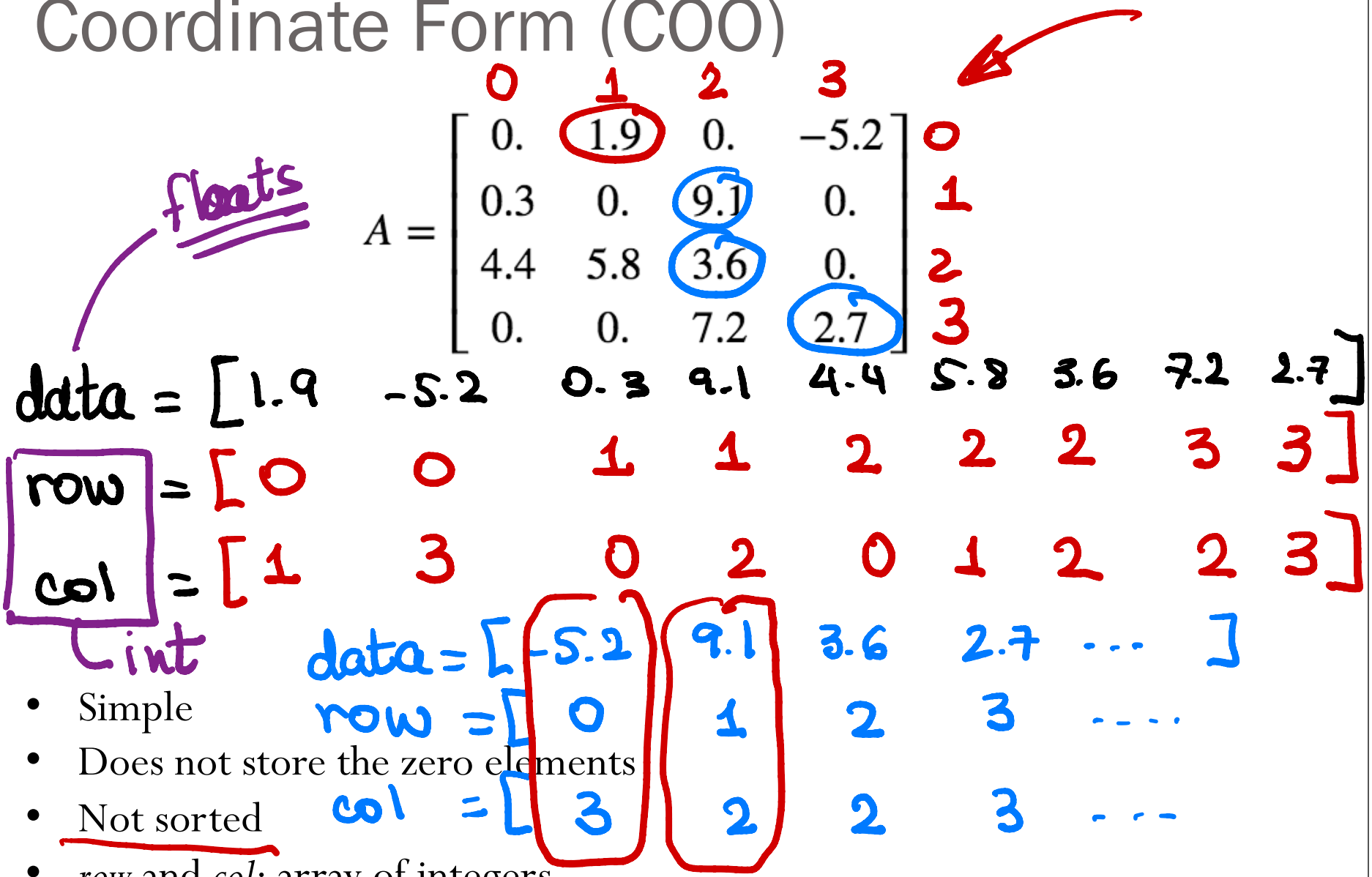
$Ashape = (nrow, ncol)$

$A_{dense} = [0. \quad 1.9 \quad 0. \quad -5.2 \quad 0.3 \quad 0. \quad 9.1 \quad 0. \quad 4.4 \quad 5.8 \quad 3.6 \quad 0. \quad 0. \quad 0. \quad 7.2 \quad 2.7]$

Row 0 Row 1 Row 2 Row 3

- Simple
- Row-wise
- Easy blocked formats
- Stores all the zeros

Coordinate Form (COO)



- Simple
- Does not store the zero elements
- Not sorted
- *row* and *col*: array of integers
- *data*: array of doubles

Representing a Sparse Matrix in Coordinate (COO) Form

1 point

Consider the following matrix:

$$A = \begin{bmatrix} 0 & 0 & 1.3 \\ -1.5 & 0.2 & 0 \\ 5 & 0 & 0 \\ 0 & 0.3 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

0 1 2

0 1 2 3 4

nnz(A) = 6

Suppose we store one row index (a 32-bit integer), one column index (a 32-bit integer), and one data value (a 64-bit float) for each non-zero entry in A. How many bytes in total are stored? Please note that 1 byte is equal to 8 bits.

data = [1.3 0.2 5 -1.5 3 0.3]

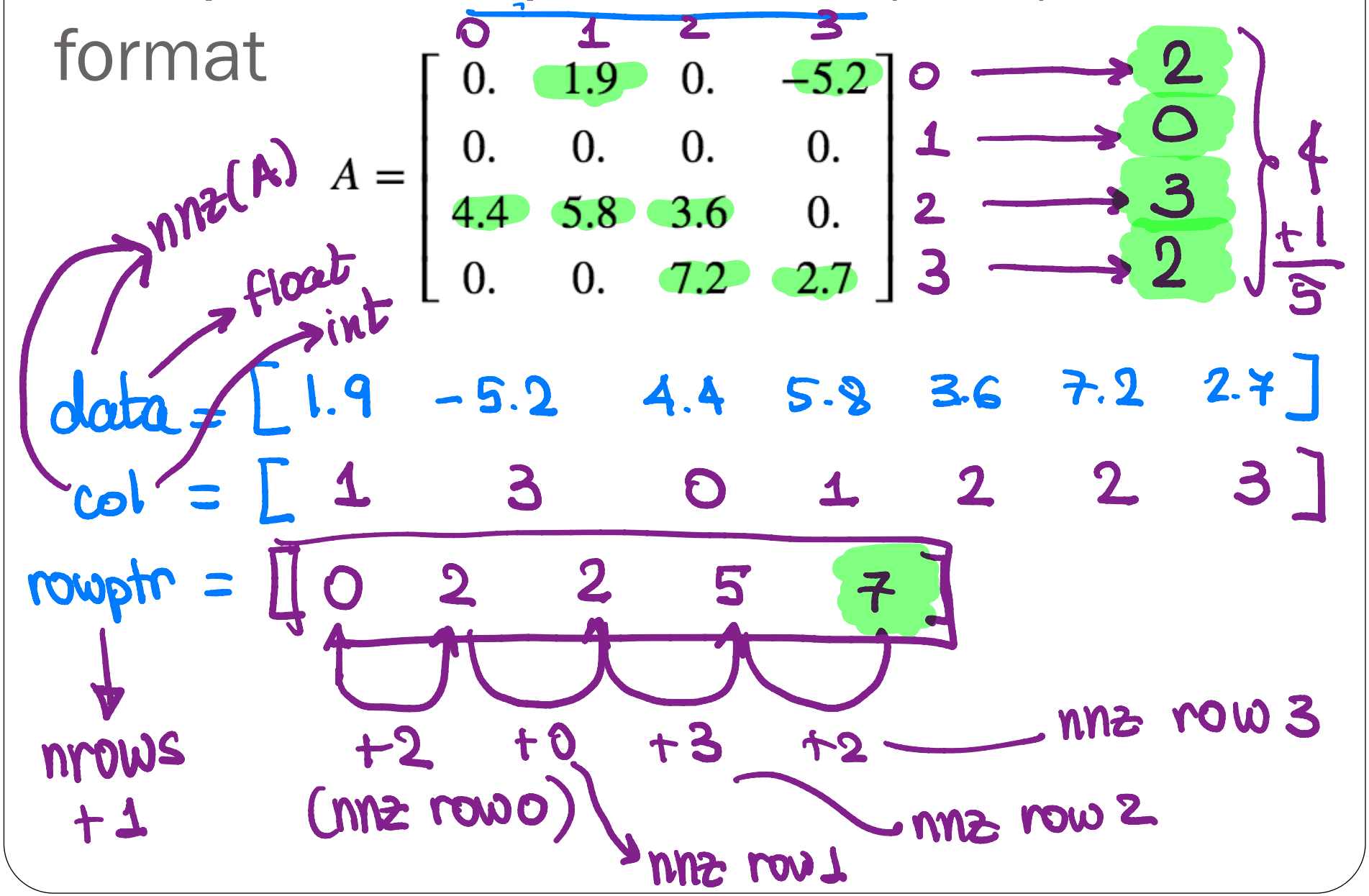
col = [2 1 0 0 2 1]

row = [0 1 2 1 3 3]

6 floats + 6 integers

(6 × 64 + 12 × 32) bits = 96 bytes

Compressed Sparse Row (CSR) format



Compressed Sparse Row (CSR)

$$A = \begin{bmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{bmatrix}$$

$$\begin{aligned} data &= [\ 1.0\ \ 2.0\ \ 3.0\ \ 4.0\ \ 5.0\ \ 6.0\ \ 7.0\ \ 8.0\ \ 9.0\ \ 10.0\ \ 11.0\ \ 12.0\ ] \\ col &= [\ 0\ \ 3\ \ 0\ \ 1\ \ 3\ \ 0\ \ 2\ \ 3\ \ 4\ \ 2\ \ 3\ \ 4\ ] \\ rowptr &= [\ 0\ \ 2\ \ 5\ \ 9\ \ 11\ \ 12\ ] \end{aligned}$$

- Does not store the zero elements
- Fast arithmetic operations between sparse matrices, and fast matrix-vector product
- *col*: contain the column indices (array of nnz integers)
- *data*: contain the non-zero elements (array of nnz doubles)
- *rowptr*: contain the row offset (array of $n + 1$ integers)