EE8725

# STORAGE SCHEME FOR SPARSE MATRICES

Pei Xu
University of Minnesota
xuxx0884 at umn.edu

# Outline

- Motivation
- Co-ordinate Storage (COO)
- Compressed Row Storage (CSR)
- Compressed Column Storage (CSC)
- Linked List
- Other Storage Schemes
- Sparse Matrices in Matlab
- References

# Motivation

- Why we need special storage scheme for sparse matrices?

- Suppose a system with
  1000 buses with an average connectivity of 4 per bus

- Then, only (4+1) x 1000 = **5000 non-zero** entries in the Y matrix who has 1000 x 1000 = **$10^6$ entries** in total.
- => If store all entries in the matrix:
  - ($10^6$-5000)/$10^6$ = 99.5% units are used to store zero entries
  - Most of resources are wasted on computing 0 x 0 = 0 and 0 + n = n

# Motivation

- Why we need special storage scheme for sparse matrices?
- Minimize storage – Store only non-zero elements
- Simplify computations – Work upon only non-zero elements

# Co-ordinate Storage (COO)

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & \cdots \end{array} \\ \begin{bmatrix} a & & & & \cdots \\ & b & & c & \cdots \\ & & d & & \cdots \\ & e & & f & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \end{array}$$

| value | a | b | c | d | e | f | … |
|---|---|---|---|---|---|---|---|
| row | 1 | 2 | 2 | 3 | 4 | 4 | … |
| column | 1 | 2 | 4 | 3 | 2 | 4 | … |

$$A \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \end{bmatrix} \quad or \quad \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \end{bmatrix}^T \times A = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ \vdots \end{bmatrix} \quad or \quad \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ \vdots \end{bmatrix}^T$$

In order to calculate $N_1$,

search all elements to find those whose row is 1 or whose column is 1.

**Inefficient**

# Compressed Row Storage (CSR)

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} a & & & & \\ & b & & c & \\ & & d & & \\ & e & & f & \\ & & & & g \end{bmatrix} \end{array}$$

| order | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **value** | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| column | 1 | 2 | 4 | 3 | 2 | 4 | 5 |
| ***row_begin*** | 1 | 2 | | 4 | 5 | | 7 |

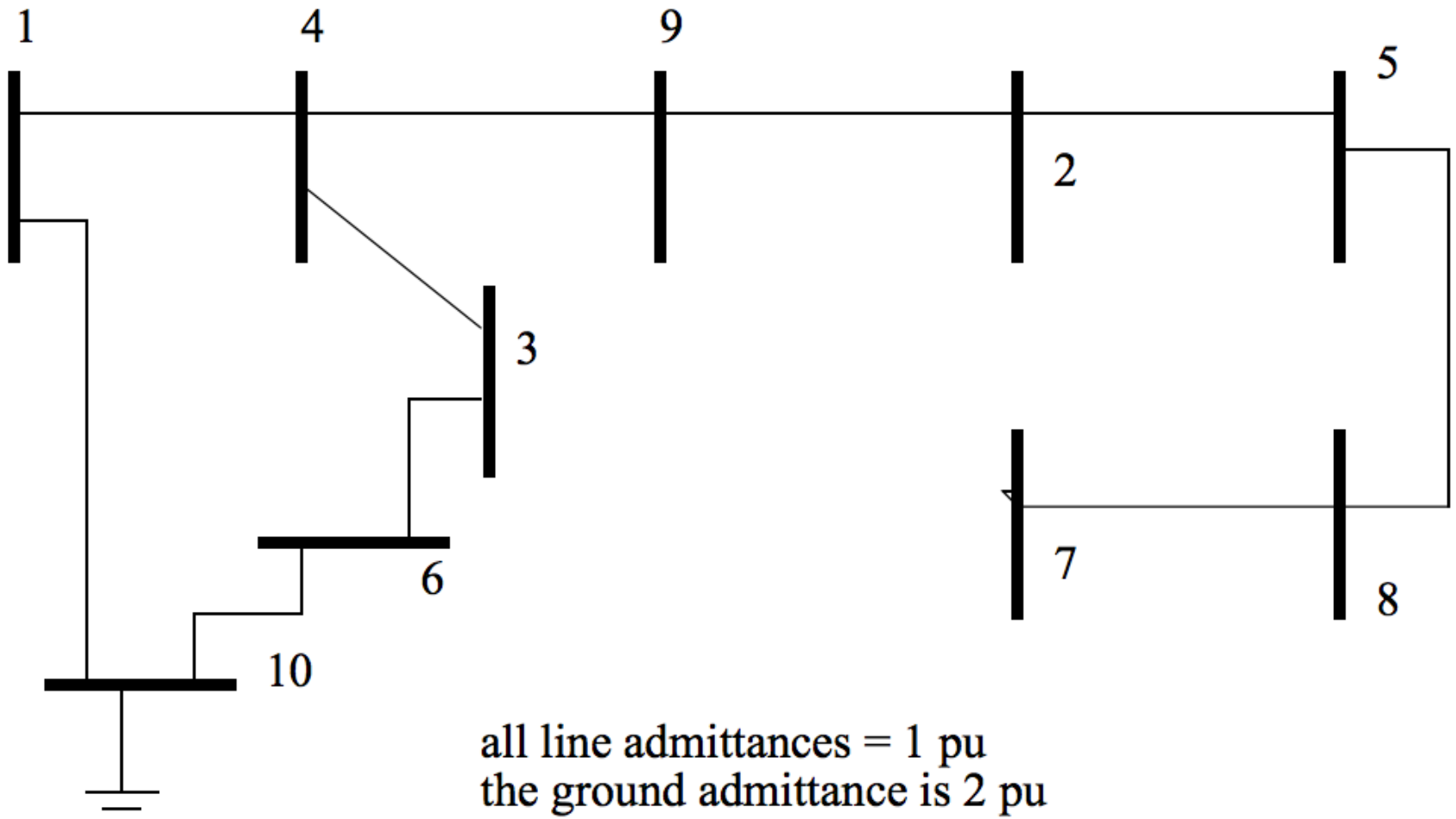
Put the elements in a same row together.

Use *row_begin* to identify the order of the number with which a row begins.

$$A \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \end{bmatrix}$$

When using CSR,
it is easy to find all elements in a row.

# Compressed Row Storage (CSR)



# Compressed Row Storage (CSR)

$$Y = \begin{bmatrix} 2 & & & -1 & & & & & -1 \\ & 2 & & & -1 & & & -1 & \\ & & 2 & -1 & & -1 & & & \\ -1 & & -1 & 3 & & & & -1 & \\ & -1 & & & 2 & & -1 & & \\ & & -1 & & & 2 & & & -1 \\ & & & & & & 1 & -1 & \\ & & & & -1 & & -1 & 2 & \\ & -1 & & -1 & & & & 2 & \\ -1 & & & & & -1 & & & 4 \end{bmatrix} \qquad DIAG = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 3 \\ 2 \\ 2 \\ 1 \\ 2 \\ 2 \\ 4 \end{bmatrix}$$

Value [ -1 -1 … -1]

Column [ 4 10 5 9 4 6 1 3 9 2 8 3 10 8 5 7 2 4 1 6]

First [ 1 3 5 7 10 12 14 16 18 20]

# Compressed Row Storage (CSR)

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} a & & & & \\ & b & & c & \\ & & d & & \\ & e & & f & \\ & & & & g \end{bmatrix} \end{array}$$

| order | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **value** | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| column | 1 | 2 | 4 | 3 | 2 | 4 | 5 |
| ***row_begin*** | 1 | 2 | | 4 | 5 | | 7 |

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} \times A = \begin{bmatrix} N_{11} & N_{12} & N_{13} & N_{14} & N_{15} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & & \end{bmatrix}$$

Search all elements to find those whose column Is 4,

in order to calculate $N_{14}$.

**Inefficient**

# Compressed Row Storage (CSR)

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} a & & \mathbf{x} & & \\ & b & & c & \\ & & d & & \\ & e & & f & \\ & & & & g \end{bmatrix} \end{array}$$

| order | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **value** | **a** | **b** | **c** | **d** | **e** | **f** | **g** |
| column | 1 | 2 | 4 | 3 | 2 | 4 | 5 |
| ***row_begin*** | 1 | 2 | | 4 | 5 | | 7 |

| order | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| **value** | **a** | **x** | **b** | **c** | **d** | **e** | **f** | **g** |
| column | 1 | 2 | 2 | 4 | 3 | 2 | 4 | 5 |
| ***row_begin*** | 1 | | 3 | | 5 | 6 | | 8 |

**Inefficient**

# Compressed Column Storage (CSC)

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} a & & & & \\ & b & & c & \\ & & d & & \\ & e & & f & \\ & & & & g \end{bmatrix} \end{array}$$

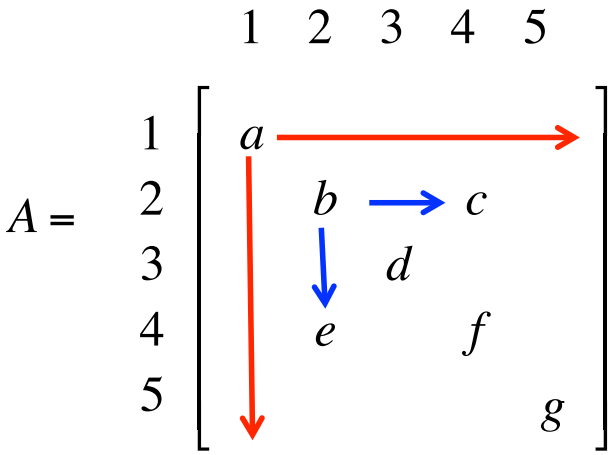
| order | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **value** | **a** | **b** | **e** | **d** | **c** | **f** | **g** |
| row | 1 | 2 | 4 | 3 | 2 | 4 | 5 |
| ***col_begin*** | 1 | 2 | | 4 | 5 | | 7 |

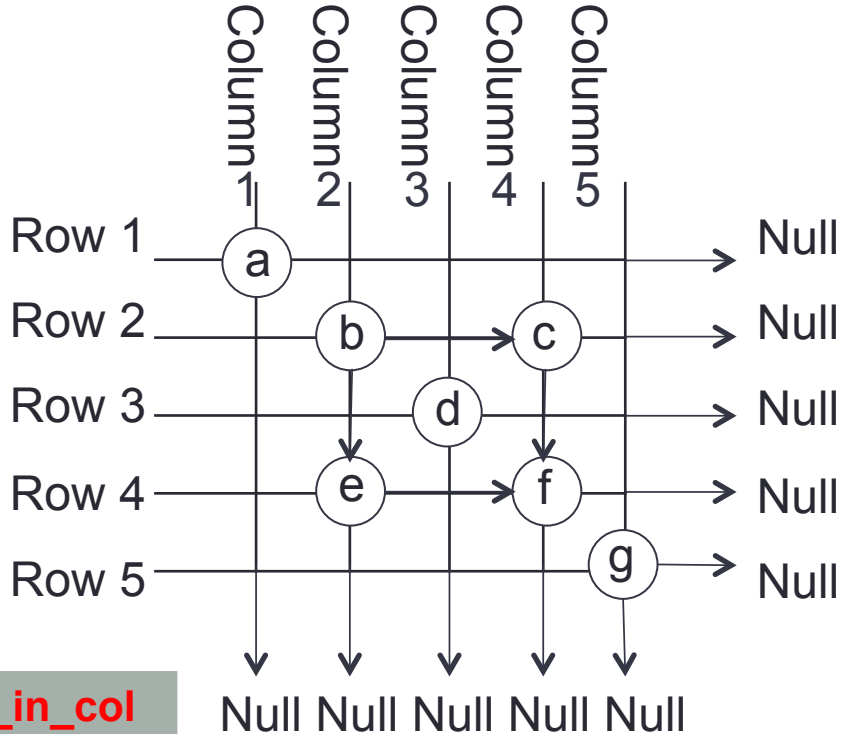
$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}^T \times A = \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \end{bmatrix}^T$$

Put elements in a same column together.
Use *col_begin* identifies the order of the number that begins in a column.

It is easy to find all elements in a column.

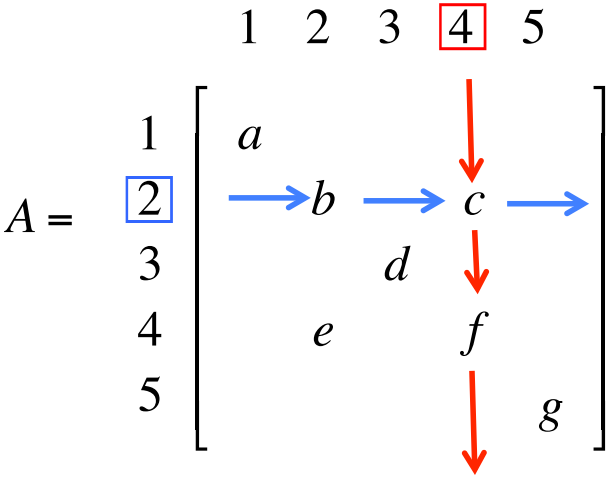
# Linked List




| Order | Row | Col | Val | Next_in_row | Next_in_col |
|---|---|---|---|---|---|
| 1 | 1 | 1 | a | Null | Null |
| 2 | 2 | 2 | b | 3 | 5 |
| 3 | 2 | 4 | c | Null | 6 |
| 4 | 3 | 3 | d | Null | Null |
| 5 | 4 | 2 | e | 6 | Null |
| 6 | 4 | 4 | f | Null | Null |
| 7 | 5 | 5 | g | Null | Null |

# Linked List

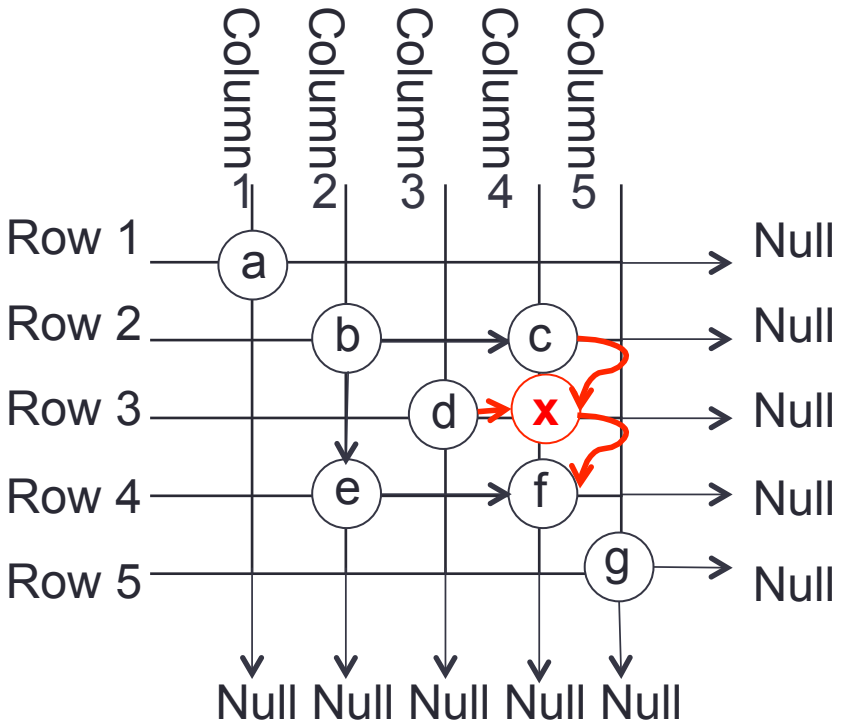



| Order | Row | Col | Val | Next_in_row | Next_in_col |
|---|---|---|---|---|---|
| 1 | 1 | 1 | a | Null | Null |
| 2 | 2 | 2 | b | 3 | 5 |
| 3 | 2 | 4 | c | Null | 6 |
| 4 | 3 | 3 | d | Null | Null |
| 5 | 4 | 2 | e | 6 | Null |
| 6 | 4 | 4 | f | Null | Null |
| 7 | 5 | 5 | g | Null | Null |

| Order | Row | Col | Val | Next_in_row | Next_in_col |
|---|---|---|---|---|---|
| 1 | 1 | 1 | a | Null | Null |
| 2 | 2 | 2 | b | 3 | 5 |
| 3 | 2 | 4 | c | Null | 6 |
| 4 | 3 | 3 | d | Null | Null |
| 5 | 4 | 2 | e | 6 | Null |
| 6 | 4 | 4 | f | Null | Null |
| 7 | 5 | 5 | g | Null | Null |

# Linked List

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} a & & & & \\ & b & & c & \\ & & d & \mathbf{x} & \\ & e & & f & \\ & & & & g \end{bmatrix} \end{array}$$




| Order | Row | Col | Val | N_in_r | N_in_c |
|---|---|---|---|---|---|
| 1 | 1 | 1 | a | Null | Null |
| 2 | 2 | 2 | b | 3 | 5 |
| 3 | 2 | 4 | c | Null | 6 |
| 4 | 3 | 3 | d | Null | Null |
| 5 | 4 | 2 | e | 6 | Null |
| 6 | 4 | 4 | f | Null | Null |
| 7 | 5 | 5 | g | Null | Null |

=>

| Order | Row | Col | Val | N_in_r | N_in_c |
|---|---|---|---|---|---|
| 1 | 1 | 1 | a | Null | Null |
| 2 | 2 | 2 | b | 3 | 5 |
| 3 | 2 | 4 | c | Null | **8** |
| 4 | 3 | 3 | d | **8** | Null |
| | | | … | | |
| **8** | **1** | **4** | **x** | **Null** | **6** |

# Comparison

| | COO | CSR | CSC | Linked List |
|---|---|---|---|---|
| Space Usage Rate | 3 | 2+N/NNZ < 3 | 2+N/NNZ < 3 | 5 |
| Find elements in a row | × | Excellent | × | Good |
| Find elements in a column | × | x | Excellent | Good |
| Add Elements | Excellent | x | x | Good |

Space Usage Rate: the average units the scheme needs to store 1 non-zero elements

N: no. of elements in a row or column

NNR: no. of non-zero elements

x : Inefficient

# Comparison

| | COO | CSR | CSC | Linked List |
|---|---|---|---|---|
| Space Usage Rate | 3 | **2+N/NNZ < 3** | 2+N/NNZ < 3 | *5* |
| Find elements in a row | × | **Excellent** | × | Good |
| Find elements in a column | × | **x** | Excellent | Good |
| Add Elements | Excellent | **x** | x | Good |

Space Usage Rate: the average units the scheme needs to store 1 non-zero elements

N: no. of elements in a row or column

NNR: no. of non-zero elements

x : Inefficient

# Other Storage Schemes

- Compressed Diagonal Storage
- Jagged Diagonal Storage
- List of Lists Storage
- Yale Storage
- Block Compressed Row Storage
- Skyline Storage

# Sparse Matrices in Matlab

CSR is used in Matlab

```
>> tic; Y*b'; disp(num2str(toc))
0.0010631
>>
>> tic; b*Y; disp(num2str(toc))
0.0058964
```

```
>> help sparse
   sparse Create sparse matrix.
   S = sparse(X) converts a
   sparse or full matrix to sparse
   form by squeezing out any
   zero elements.
   ……
```

# References


- [1] Soman, Shreevardhan Arunchandra, Khaparde, S.A., Pandit and Shubha, *Computational Methods for Large Sparse Power Systems Analysis  An Object Oriented Approach*. 2002
- [2] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide.* 2000

# THANK YOU

Pei Xu
University of Minnesota
xuxx0884 at umn.edu