Parallel Numerics, WT 2012/2013

4 Sparse Matrices

Contents

4.1. General Properties of Sparse Matrices

- Full $n \times n$-matrix: storage $\mathcal{O}(n^2)$, solution $\mathcal{O}(n^3) \rightarrow$ too costly for most applications, esp. for fine discretization (large n)
- Formulate given problem in clever way that leads to a linear system that is sparse: $\mathcal{O}(n)$, solution $\mathcal{O}(n)$?
  - (that is structured: storage $\mathcal{O}(n)$, solution $O(n\log(n))$, e.g., FFT)
  - (that is dense, but reduced from, e.g., 3D to 2D)
  - (based on sparse grids)
  - (based on tensor approximations)
- Examples:
  - tridiagonal matrix
  - banded matrix
  - block band matrix

Sparse Matrix Example

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

Additionally we need to store:

- the size of the matrix $n = 5$
- the number of nonzero entries $\text{nnz} = 12$

4.1.1. Storage in Coordinate Form

values	AA	12	9	7	5	1	2	11	3	6	4	8	10
row	JR	5	3	3	2	1	1	4	2	3	2	3	4
column	JC	5	5	3	4	1	4	4	1	1	2	4	3

To store:

- n
- nnz
- $2 \cdot$ nnz integer for row and column indices in JR and JC
- nnz float in AA

No sorting included. Redundant information.

Storage in Coordinate Form (cont.)

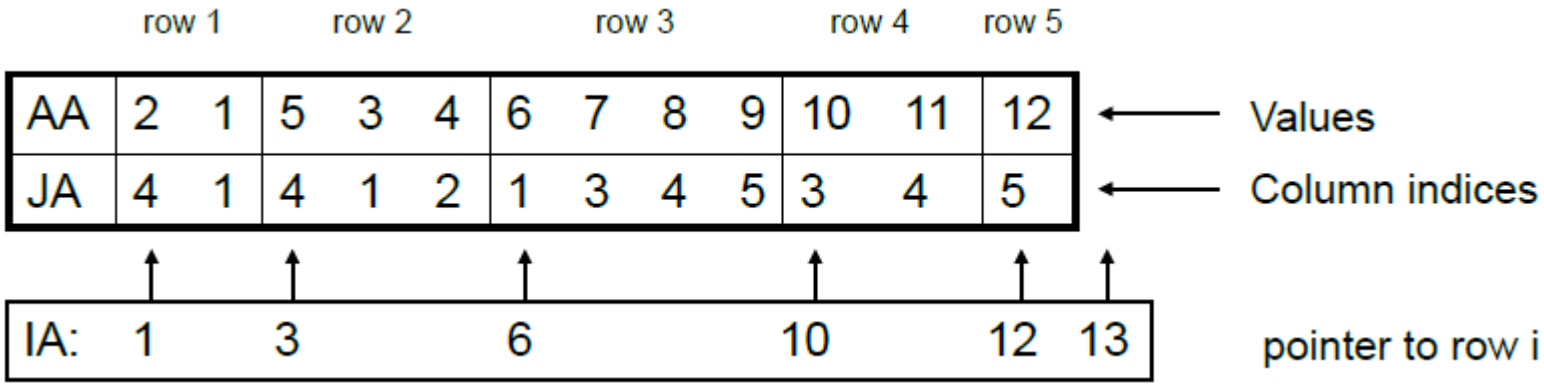
values	AA	12	9	7	5	1	2	11	3	6	4	8	10
row	JR	5	3	3	2	1	1	4	2	3	2	3	4
column	JC	5	5	3	4	1	4	4	1	1	2	4	3

Pseudocode for computing $c = A \cdot b$:

$c = 0;$
`for` $j = 1 \ : \ \text{nnz}(A)$
$\quad c_{JR(j)} = c_{JR(j)} + \underbrace{AA(j)}_{A_{JR(j),JC(j)}} * b_{JC(j)};$
`end`

- Disadvantage: Indirect addressing (indexing) in vector c and $b \rightarrow$ jumps in memory
- Advantage: No difference between columns and rows (A and A^T), simple.

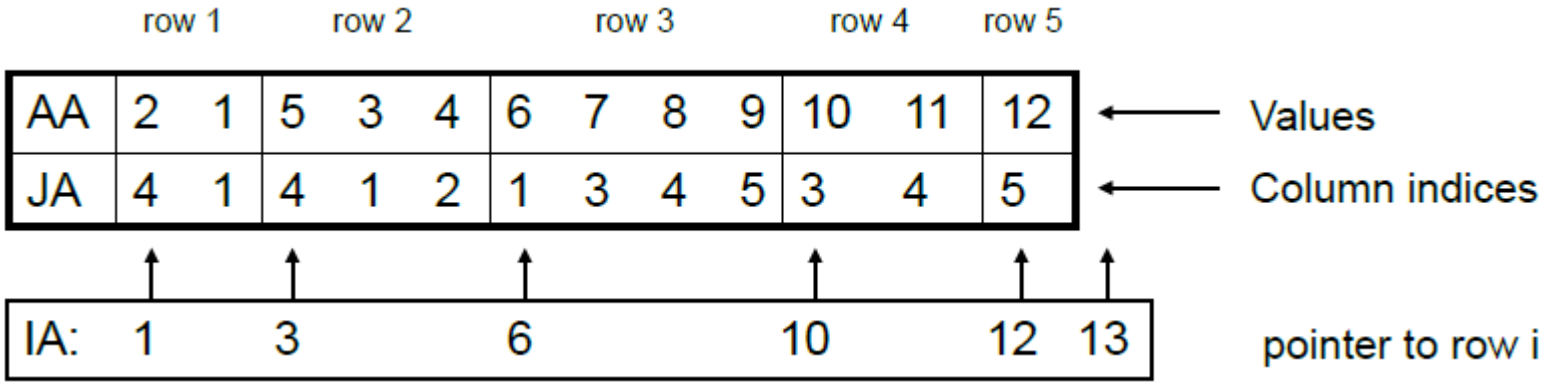
4.1.2. Compressed Sparse Row Format: CSR



Storage:

- n and nnz
- $n + \text{nnz} + 1$ integer
- nnz float

Compressed Sparse Row: CSR (cont.)



Pseudocode for computing $c = A \cdot b$:

```
c = 0;
for i = 1 : n
   for j = IA(i) : IA(i+1) - 1
      c_i = c_i + AA(j) * b_JA(j);
   end
end
```

- Indirect addressing only in b.
- Columnwise $\rightarrow$ compressed sparse column format.

4.1.3. CSR with Extracted Main Diagonal

main diagonal entries nondiagonal entries in CSR

Pointer to begin of *i*th row

Storage:

- *n* and nnz
- nnz +1 integer
- nnz +1 float

CSR with Extracted Main Diagonal (cont.)

main diagonal entries | nondiagonal entries in CSR

Pseudocode for computing $c = A \cdot b$:

```
c = 0;
for i = 1 : n
   c_i = AA_i * b_i;
   for j = JA(i) : JA(i+1) - 1
      c_i = c_i + AA_j * b_JA(j);
   end
end
```

4.1.4. Diagonalwise Storage

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

New matrix A!
Different matrix to slides before!

Diagonal numbers: $-1 \;\; 0 \;\; 2$

Values in:

$$\text{DIAG} = \begin{pmatrix} * & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & * \\ 11 & 12 & * \end{pmatrix}, \quad \text{IOFF} = (-1 \;\; 0 \;\; 2)$$

Storage: n, $nd :=$ #diagonals, nd integers for IOFF and $n \cdot nd$ float

4.1.5. Rectangular Storage Scheme by Pressing from the Right

$$\left(\begin{array}{ccccc} 1 & 0 & 2 & 0 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 0 & 6 & 7 & 0 & 8 \\ 0 & 0 & 9 & 10 & 0 \\ 0 & 0 & 0 & 11 & 12 \end{array}\right| \leftarrow \text{pressing from right}$$

gives

$$\text{COEF} = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \\ 9 & 10 & 0 \\ 11 & 12 & 0 \end{pmatrix} \qquad \text{JCOEF} = \begin{pmatrix} 1 & 3 & * \\ 1 & 2 & 4 \\ 2 & 3 & 5 \\ 3 & 4 & * \\ 4 & 5 & * \end{pmatrix}$$

Storage: n, $n \cdot nl$ integer and float ($nl :=$ nnz of longest row)

Rectangular Storage Scheme by Pressing from the Right (cont.)

Pseudocode for computing $c = A \cdot b$:

```
c = 0;
for i = 1 : n
   for j = 1 : nl
      c_i = c_i + COEF(i,j) * b(JCOEF(i,j));
   end
end
```

This format was used in ELLPACK (package of subroutines for elliptic PDEs).

4.1.6. Jagged Diagonal Form

Prestep: Sort rows after their length. Long rows first.

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

Length 3

Length 2

- Values of PA: DJ = (3 6 1 9 11 | 4 7 2 10 12 | 5 8|)
- Column indices: JDIAG = (1 2 1 3 4 | 2 3 3 4 5 | 4 5|)
- Pointer to beginning of jth diagonal: IDIAG = (1 6 11 13)

Jagged Diagonal Form (cont.)

NDIAG := number of jagged diagonals

Storage: n, NDIAG, nnz float, nnz + NDIAG integer

Pseudocode for computing $c = A \cdot b$:

```
c = 0;
for j = 1 : NDIAG
  for i = 1 : IDIAG(j+1) - IDIAG(j)
    k = IDIAG(j) + i - 1;
    c_i = c_i + DJ(k) * b(JDIAG(k));
  end
end
```

- Always start with row 1.
- More operations on neighboring data.
- Less indirect addressing.
- Pre-permutation changes only rows. Can be done implicitly.

Survey on Sparse Storage Formats

Coordinate form and CSR traditional way to specify sparse matrix in MATLAB.

	global int	idx int	value floats
Coordiante	2	$2 \cdot \text{nnz}$	nnz
CSR	2	$n + \text{nnz} + 1$	nnz
CSR with Extract. Diag.	2	$\text{nnz} + 1$	$\text{nnz} + 1$
Diagonalwise	2	nd	$n \cdot nd$
Rectang. with Pressing	1	$n \cdot nl$	$n \cdot nl$
Jagged Diagonal	2	$\text{nnz} + nd$	nnz