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
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 \( \mathbf{A} \) and \( \mathbf{B} \):

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

where \( \text{nnz}(\mathbf{X}) = \) number of non-zero elements of a matrix \( \mathbf{X} \)
# Popular Storage Structures

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS</td>
<td>Dense</td>
<td>ELL</td>
<td>Ellpack-Itpack</td>
</tr>
<tr>
<td>BND</td>
<td>Linpack Banded</td>
<td>DIA</td>
<td>Diagonal</td>
</tr>
<tr>
<td>COO</td>
<td>Coordinate</td>
<td>BSR</td>
<td>Block Sparse Row</td>
</tr>
<tr>
<td>CSR</td>
<td>Compressed Sparse Row</td>
<td>SSK</td>
<td>Symmetric Skyline</td>
</tr>
<tr>
<td>CSC</td>
<td>Compressed Sparse Column</td>
<td>BSR</td>
<td>Nonsymmetric Skyline</td>
</tr>
<tr>
<td>MSR</td>
<td>Modified CSR</td>
<td>JAD</td>
<td>Jagged Diagonal</td>
</tr>
<tr>
<td>LIL</td>
<td>Linked List</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

We will focus on COO and CSR!
Dense (DNS)

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

\[A_{\text{shape}} = (nrow, ncol)\]

\[A_{\text{dense}} = \begin{bmatrix}
0. & 1.9 & 0. & -5.2 & 0.3 & 0. & 9.1 & 0. & 4.4 & 5.8 & 3.6 & 0. & 0. & 0. & 7.2 & 2.7 \\
\end{bmatrix}\]

- Simple
- Row-wise
- Easy blocked formats
- Stores all the zeros
Coordinate (COO)

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

\[ \text{data} = [1.9, -5.2, 0.3, 9.1, 4.4, 5.8, 3.6, 7.2, 2.7] \]

\[ \text{row} = [0, 0, 1, 1, 2, 2, 2, 3, 3] \]

\[ \text{col} = [1, 3, 0, 2, 0, 1, 2, 2, 3] \]

• Simple
• Does not store the zero elements
• Not sorted
• row and col: array of integers
• data: array of doubles
Iclicker question

How many integers are stored in COO format
($A$ has dimensions $n \times n$)?

A) $nnz$
B) $n$
C) $2 \ nnz$
D) $n^2$
E) $2 \ n$
Representing a Sparse Matrix in Coordinate (COO) Form

Consider the following matrix:

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

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.

A) 56 bytes
B) 72 bytes
C) 96 bytes
D) 120 bytes
E) 144 bytes
Compressed Sparse Row (CSR)

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

\[
data = [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0]
\]

\[
col = [0, 3, 0, 1, 3, 0, 2, 3, 4, 2, 3, 4]
\]

\[
rowptr = [0, 2, 5, 9, 11, 12]
\]

Row offsets

\( nnz(A) \)
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}

- 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)
Example - CSR format

\[
A = \begin{bmatrix}
0. & 1.9 & 0. & -5.2 \\
0. & 0. & 0. & 0. \\
4.4 & 5.8 & 3.6 & 0. \\
0. & 0. & 7.2 & 2.7
\end{bmatrix}
\]

data = [1.9, -5.2, 4.4, 5.8, 3.6, 7.2, 2.7]

rowptr = [0, 2, 2, 5, 7]

col = [1, 3, 0, 1, 2, 2, 3]