Cost of LU factorization

```
## Algorithm 1
## Factorization using the block-format,
## creating new matrices L and U
## and not modifying A
print("LU factorization using Algorithm 1")
L = np.zeros((n,n))
U = np.zeros((n,n))
M = A.copy()
for i in range(n):
    U[i,i:] = M[i,i:]
    L[i:,i] = M[i:,i]/U[i,i]
    M[i+1:,i+1:] -= np.outer(L[i+1:,i],U[i,i+1:])
```

Side note:

$$\sum_{i=1}^{m} i = \frac{1}{2} m(m+1)$$

$$\sum_{i=1}^{m} i^2 = \frac{1}{6} m(m+1)(2m+1)$$

Example

Which of the following statements are true about the LU factorization of an $n \times n$ matrix A, assuming LU factorization of A exists and not considering any row/column interchanges?

Select all that apply:

1) $A = LU$.
2) LU factorization is exactly performing Gaussian elimination.
3) We can solve for $LUx = b$ instead of solving $Ax = b$ to obtain x.
4) L is a lower triangular matrix, and is exactly the lower part of A but with unit diagonal.
5) U is an upper triangular matrix, and is exactly the upper part of A (including diagonal).

A) 1,2,3
B) 1,2,3,5
C) 1,3
D) 1,2,3,4,5
E) 4,5

Solving linear systems

In general, we can solve a linear system of equations following the steps:

1) Factorize the matrix $\boldsymbol{A}$: $\boldsymbol{A = LU}$ (complexity $O(n^3)$)

2) Solve $\boldsymbol{L\,y = b}$ (complexity $O(n^2)$)

3) Solve $\boldsymbol{U\,x = y}$ (complexity $O(n^2)$)

But why should we decouple the factorization from the actual solve? (Remember from Linear Algebra, Gaussian Elimination does not decouple these two steps…)

What can go wrong with the previous algorithm?

Demo "Little c"

$$\boldsymbol{M} = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 1 & \mathbf{4} & 3 & 3 \\ 1 & 2 & 6 & 2 \\ 1 & 3 & 4 & 2 \end{pmatrix} \quad \boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{U} = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{l}_{21}\boldsymbol{u}_{12} = \begin{pmatrix} 4 & 2 & 0.5 \\ 4 & 2 & 0.5 \\ 4 & 2 & 0.5 \end{pmatrix} \quad \boldsymbol{M} - \boldsymbol{l}_{21}\boldsymbol{u}_{12} = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 1 & \mathbf{0} & 1 & 2.5 \\ 1 & -2 & 4 & 1.5 \\ 1 & -1 & 2 & 1.5 \end{pmatrix}$$

The next update for the lower triangular matrix will result in a division by zero! LU factorization fails.

What can we do to get something like an LU factorization?

Pivoting

Approach:

1. Swap rows if there is a zero entry in the diagonal
2. Even better idea: Find the largest entry (by absolute value) and swap it to the top row.

The entry we divide by is called the pivot.

Swapping rows to get a bigger pivot is called (partial) pivoting.

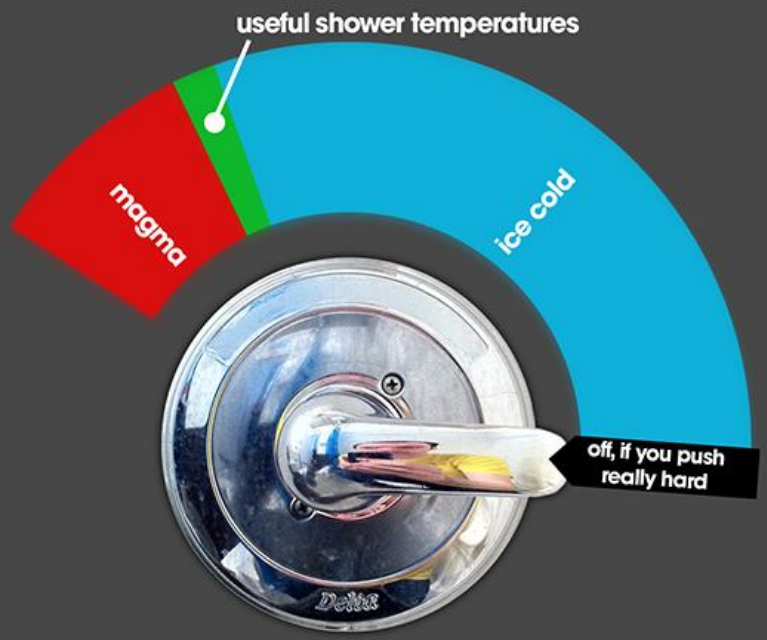
$$\begin{pmatrix} a_{11} & \boldsymbol{a_{12}} \\ \boldsymbol{a_{21}} & \boldsymbol{A_{22}} \end{pmatrix} = \begin{pmatrix} u_{11} & \boldsymbol{u_{12}} \\ u_{11}\,\boldsymbol{l_{21}} & \boldsymbol{l_{21}u_{12}} + \boldsymbol{L_{22}U_{22}} \end{pmatrix}$$

Find the largest entry (in magnitude)

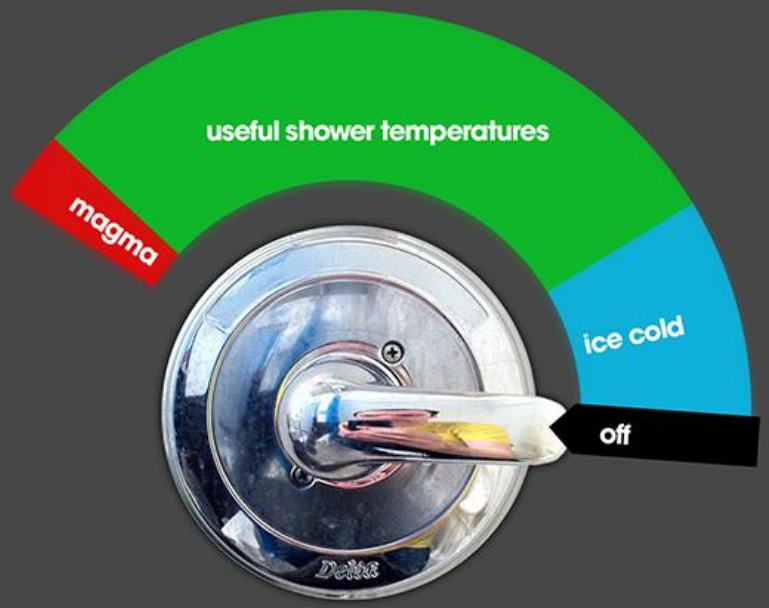
Linear System of Equations - Conditioning

the shower faucet

how they are:



how they should be:



Numerical experiments

Input has uncertainties:

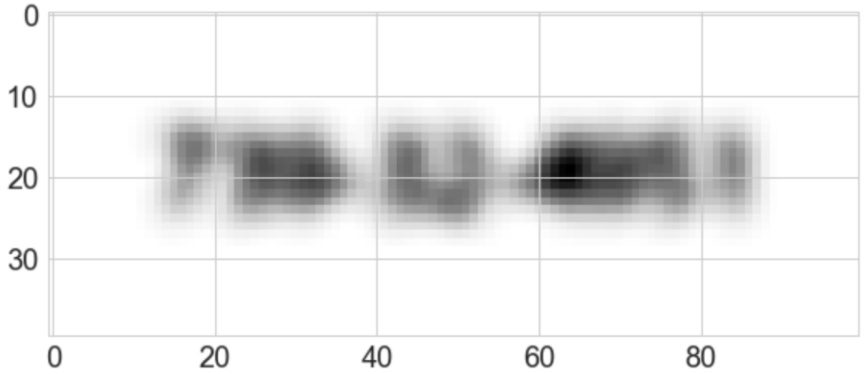
- Errors due to representation with finite precision
- Error in the sampling

Once you select your numerical method , how much error should you expect to see in your **output?**

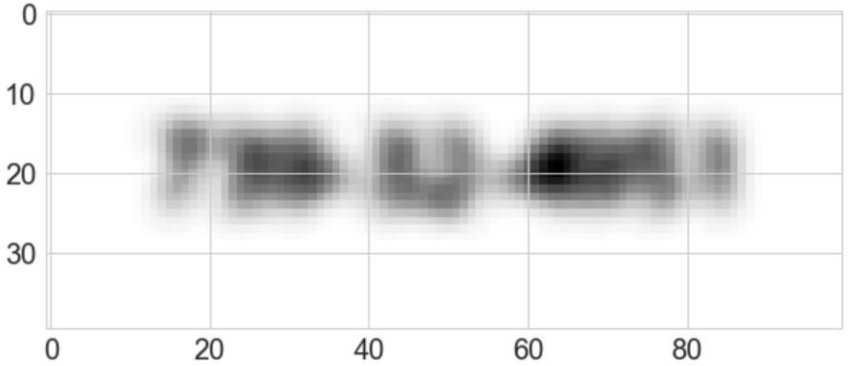
Is your method sensitive to errors (perturbation) in the input?

Demo "HilbertMatrix-ConditionNumber"

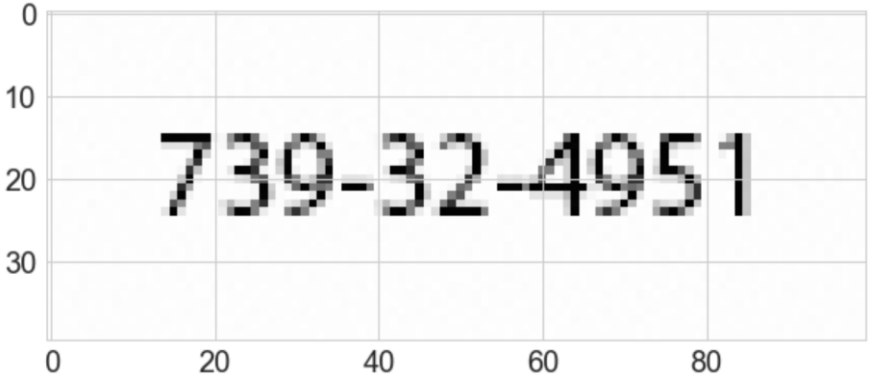
$\boldsymbol{b} + a * 10^{-6}$ $(a \in (0,1))$

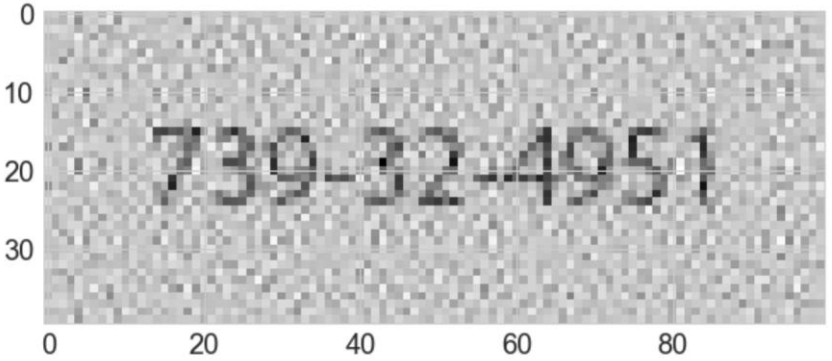


$\boldsymbol{b} + a * 10^{-4}$ $(a \in (0,1))$



Solve $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$ for $\boldsymbol{x}$





Is your method sensitive to errors (perturbation) in the input?
How much noise can we add to the input data?
How can we define "little" amount of noise? Should be relative with the magnitude of the data.

Sensitivity of Solutions of Linear Systems

Suppose we start with a non-singular system of linear equations $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$.

We change the right-hand side vector $\boldsymbol{b}$ (input) by a small amount $\Delta\boldsymbol{b}$.

How much the solution $\boldsymbol{x}$ (output) changes, i.e., how large is $\Delta\boldsymbol{x}$?

Sensitivity of Solutions of Linear Systems

Sensitivity of Solutions of Linear Systems

We can also add a perturbation to the matrix $\boldsymbol{A}$ (input) by a small amount $\boldsymbol{E}$, such that

$$(\boldsymbol{A} + \boldsymbol{E})\,\widehat{\boldsymbol{x}} = \boldsymbol{b}$$

and in a similar way obtain:

$$\frac{\|\Delta \boldsymbol{x}\,\|}{\|\boldsymbol{x}\|} \leq \left\|\boldsymbol{A}^{-1}\right\| \|\boldsymbol{A}\| \, \frac{\|\boldsymbol{E}\|}{\|\boldsymbol{A}\|}$$

Condition number

The condition number is a measure of sensitivity of solving a linear system of equations to variations in the input.

The condition number of a matrix $\boldsymbol{A}$:

$$cond(\boldsymbol{A}) = \left\|\boldsymbol{A}^{-1}\right\| \|\boldsymbol{A}\|$$

Recall that the induced matrix norm is given by

$$\|\boldsymbol{A}\| = \max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\|$$

And since the condition number is relative to a given norm, we should be precise and for example write:

$$cond_2(\boldsymbol{A}) \text{ or } cond_\infty(\boldsymbol{A})$$

Demo "HilbertMatrix-ConditionNumber"

Iclicker question

Give an example of a matrix that is very well-conditioned (i.e., has a condition number that is good for computation). Select the best possible condition number(s) of a matrix?

A) $cond(\boldsymbol{A}) < 0$
B) $cond(\boldsymbol{A}) = 0$
C) $0 < cond(\boldsymbol{A}) < 1$
D) $cond(\boldsymbol{A}) = 1$
E) $cond(\boldsymbol{A}) =$ large numbers

Condition number

Demo "HilbertMatrix-ConditionNumber"

$$\frac{\|\Delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq cond(\boldsymbol{A}) \frac{\|\Delta \boldsymbol{b}\|}{\|\boldsymbol{b}\|}$$

Small condition numbers mean not a lot of error amplification. Small condition numbers are good!

The identity matrix should be well-conditioned:

$$\|\boldsymbol{I}\| = \max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{I}\,\boldsymbol{x}\| = 1$$

It turns out that this is the smallest possible condition number:

$$cond(\boldsymbol{A}) = \left\|\boldsymbol{A}^{-1}\right\| \|\boldsymbol{A}\| \geq \left\|\boldsymbol{A}^{-1}\boldsymbol{A}\right\| = \|\boldsymbol{I}\| = 1$$

If $\boldsymbol{A}^{-1}$ does not exist, then $cond(\boldsymbol{A}) = \infty$ (by convention)

Recall Induced Matrix Norms

$$\|\boldsymbol{A}\|_1 = \max_j \sum_{i=1}^{n} |A_{ij}|$$

Maximum absolute column sum of the matrix $\boldsymbol{A}$

$$\|\boldsymbol{A}\|_\infty = \max_i \sum_{j=1}^{n} |A_{ij}|$$

Maximum absolute row sum of the matrix $\boldsymbol{A}$

$$\|\boldsymbol{A}\|_2 = \max_k \sigma_k$$

σ_k are the singular value of the matrix $\boldsymbol{A}$

Iclicker question

Condition Number of a Diagonal Matrix

What is the 2-norm-based condition number of the diagonal matrix

$$A = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 13 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} ?$$

A) 1
B) *50*
C) 100
D) 200

Demo "HilbertMatrix-ConditionNumber"

About condition numbers

1. For any matrix $\boldsymbol{A}$, $cond(\boldsymbol{A}) \geq 1$

2. For the identity matrix $\boldsymbol{I}$, $cond(\boldsymbol{I}) = 1$

3. For any matrix $\boldsymbol{A}$ and a nonzero scalar γ, $cond(\gamma \boldsymbol{A}) = cond(\boldsymbol{A})$

4. For any diagonal matrix $\boldsymbol{D}$, $cond(\boldsymbol{D}) = \frac{\boldsymbol{max}|d_i|}{\boldsymbol{min}|d_i|}$

5. The condition number is a measure of how close a matrix is to being singular: a matrix with large condition number is nearly singular, whereas a matrix with a condition number close to 1 is far from being singular

6. The determinant of a matrix is NOT a good indicator is a matrix is near singularity

Iclicker question

The need for pivoting depends on whether the matrix is singular.

A) True
B) False

Which of the following statements is correct?

Choice*

A) A singular matrix does not have a solution
B) A matrix is well conditioned if its condition number is less or equal to 1
C) A nonsingular matrix always has a solution
D) 1-norm of a matrix is the absolute column sum

Condition Number of Orthogonal Matrices

What is the 2-norm condition number of an orthogonal matrix A?

$$cond(\boldsymbol{A}) = \left\|\boldsymbol{A}^{-1}\right\|_{\boldsymbol{2}} \|\boldsymbol{A}\|_{\boldsymbol{2}} = \left\|\boldsymbol{A}^{\boldsymbol{T}}\right\|_{\boldsymbol{2}} \|\boldsymbol{A}\|_{\boldsymbol{2}} = 1$$

That means orthogonal matrices have optimal conditioning.

They are very well-behaved in computation.

Residual versus error

Our goal is to find the solution $\boldsymbol{x}$ to the linear system of equations $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$

Let us recall the solution of the perturbed problem

Demo

Residual versus error

Residual versus error

$$\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{A}\|\|\widehat{\boldsymbol{x}}\|} \leq c\ \epsilon_m$$

Where c is large without pivoting and small with partial pivoting. Therefore, Gaussian elimination with partial pivoting yields **small relative residual regardless of conditioning of the system**.

Iclicker question

When solving a system of linear equations via LU with partial pivoting, which of the following is guaranteed to be small?

A) Relative residual: $\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{A}\|\|\boldsymbol{x}\|}$

B) Relative error: $\frac{\|\Delta \boldsymbol{x}\ \|}{\|\boldsymbol{x}\|}$

C) Neither one of them

D) Both of them

Residual versus error

Let us first obtain the norm of the error:

Residual versus error

Let us first obtain the norm of the error:

Rule of thumb for conditioning

Suppose we want to find the solution $\boldsymbol{x}$ to the linear system of equations $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$ using LU factorization with partial pivoting and backward/forward substitutions.

Suppose we compute the solution $\hat{\boldsymbol{x}}$.

If the entries in $\boldsymbol{A}$ and $\boldsymbol{b}$ are accurate to S decimal digits,

and $cond(\boldsymbol{A}) = \boldsymbol{10^W}$,

then the elements of the solution vector $\hat{\boldsymbol{x}}$ will be accurate to about

$$S - W$$

decimal digits

Iclicker question

Matrix Conditioning: Accurate digits

1 point

Let's say we want to solve the following linear system:

$$Ax = b$$

Assuming you are working with IEEE double precision floating point numbers, how many digits of accuracy will your answer have if $\kappa(A) = 1000$?

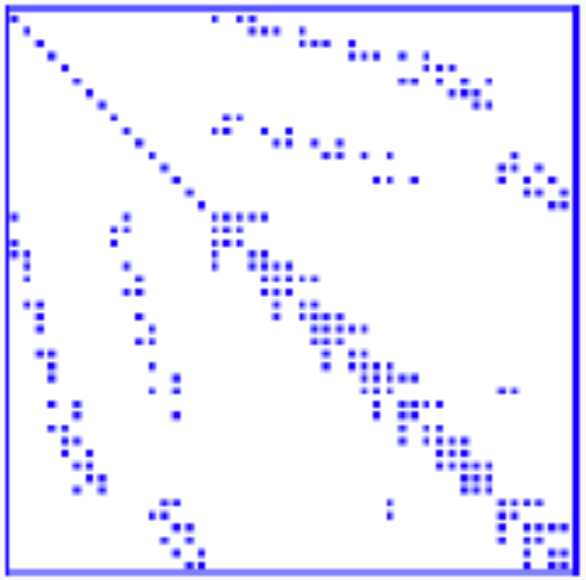
A) 3
B) 10
C) 13
D) 16
E) 32

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?

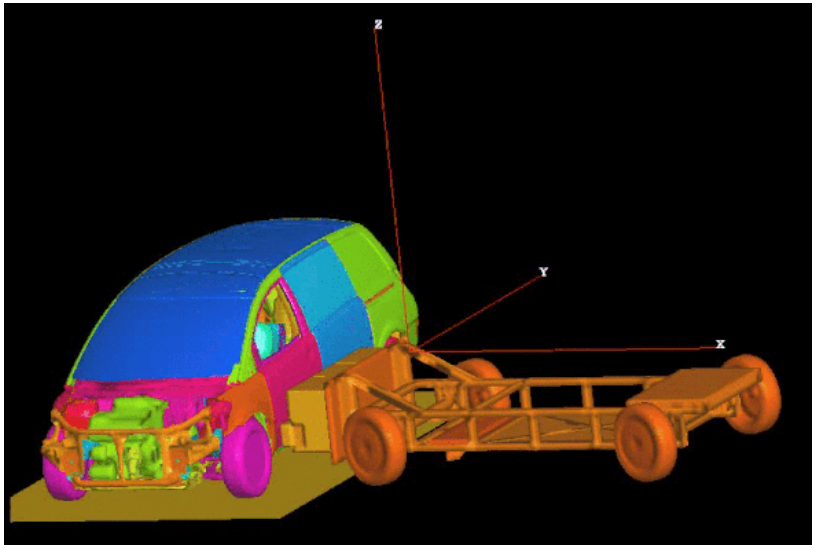


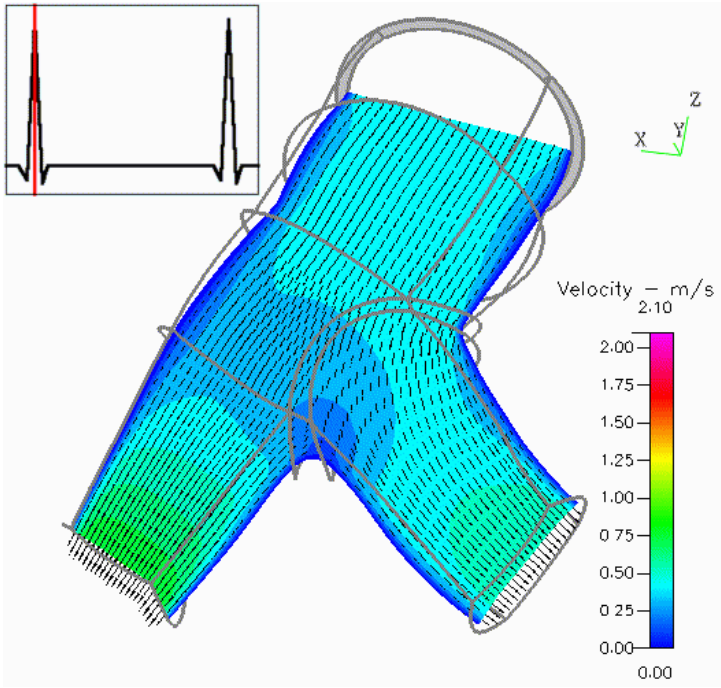
- **Sparse matrices** (vague definition): matrix with few non-zero entries.
- For practical purposes: an $m \times n$ matrix is sparse if it has $O(\min(m,n))$ non-zero entries.
- This means roughly a constant number of non-zero entries per row and column.
- Another definition: "matrices that allow special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

Sparse Matrices: Goals

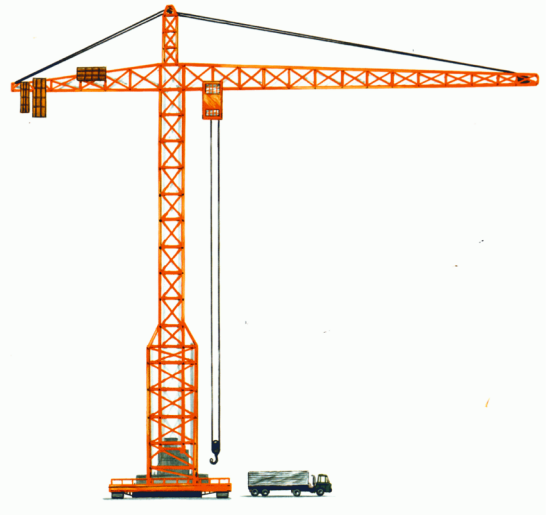
- **Perform standard matrix computations economically, i.e., without storing the zeros of the matrix.**

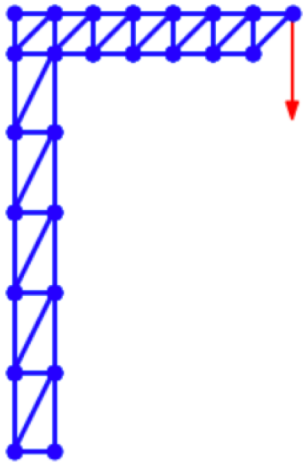
- For typical Finite Element and Finite Difference matrices, the number of non-zero entries is $O(n)$

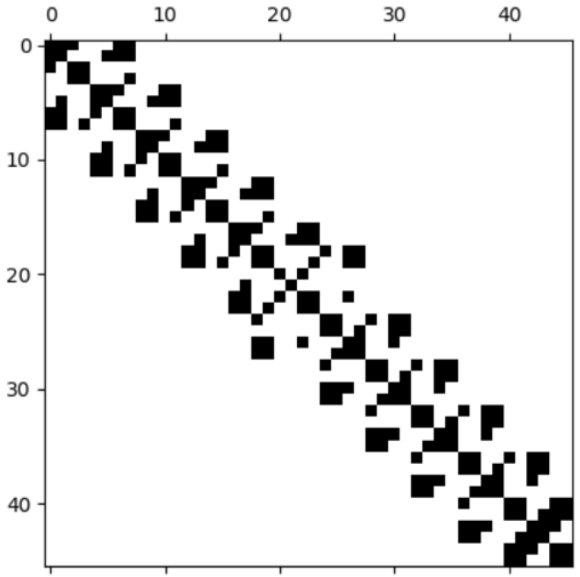




Sparse Matrices: MP example







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

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 (COO)

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

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

Example

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

data	=	[	12.0	9.0	7.0	5.0	1.0	2.0	11.0	3.0	6.0	4.0	8.0	10.0	]
row	=	[	4	2	2	1	0	0	3	1	2	1	2	3	]
col	=	[	4	4	2	3	0	3	3	0	0	1	3	2	]

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

A) nnz
B) n
C) $2\ nnz$
D) n^2
E) $2\ n$

Example

Representing a Sparse Matrix in Coordinate (COO) Form

1 point

Consider the following matrix:

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

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)

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

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)

Example - CSR format

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