Solving Linear System of Equations

The “Undo” button for Linear Operations

Matrix-vector multiplication: given the data $\boldsymbol{x}$ and the operator $\boldsymbol{A}$, we can find $\boldsymbol{y}$ such that

$\boldsymbol{y} = \boldsymbol{A}\,\boldsymbol{x}$

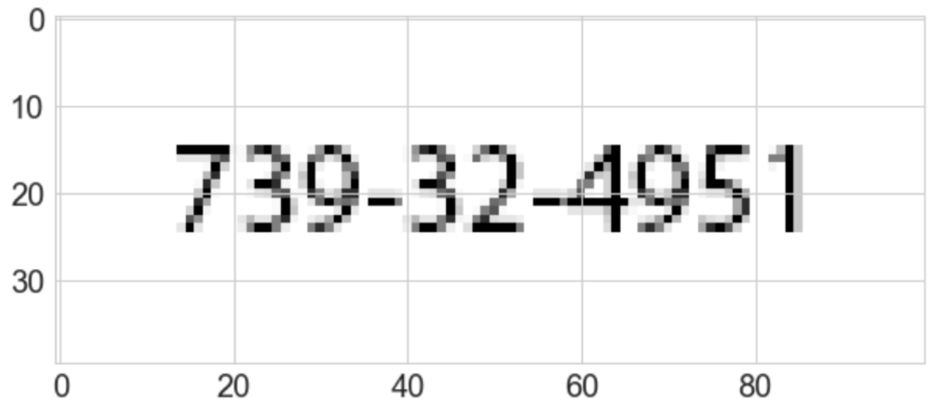
$$\begin{tikzcd}\boldsymbol{x} \arrow[r, "\boldsymbol{A}", "\text{transformation}"'] & \boldsymbol{y}\end{tikzcd}$$

What if we know $\boldsymbol{y}$ but not $\boldsymbol{x}$? How can we “undo” the transformation?

$$\begin{tikzcd}\boldsymbol{y} \arrow[r, "\boldsymbol{A}^{-1}", "?"'] & \boldsymbol{x}\end{tikzcd}$$

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

Image Blurring Example

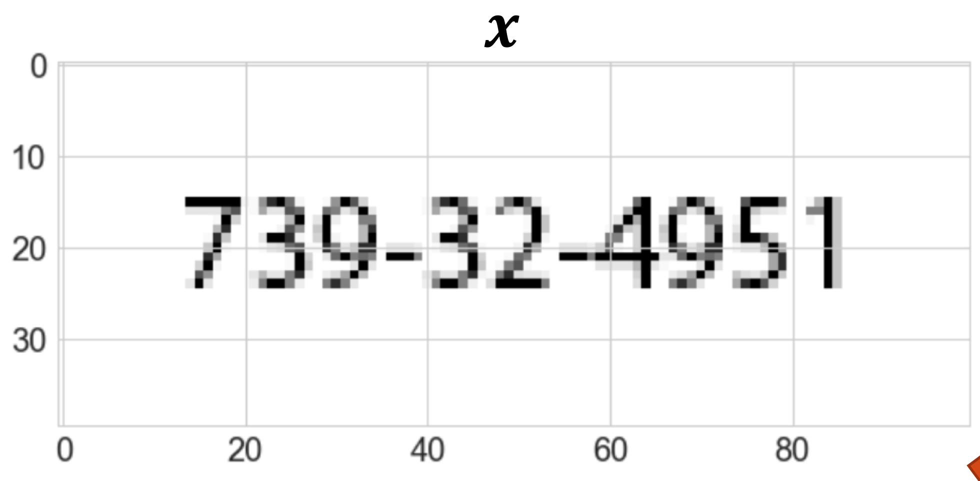


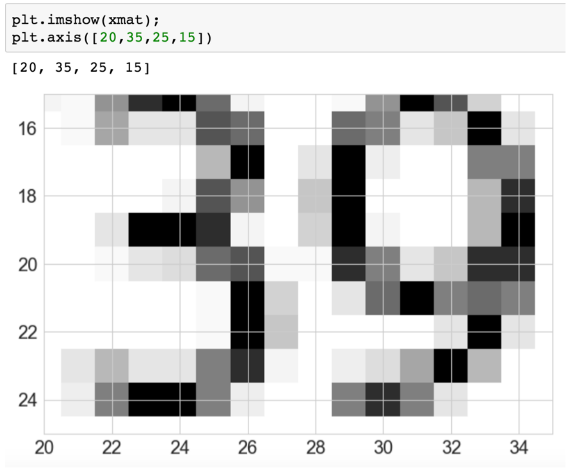
- Image is stored as a 2D array of real numbers between 0 and 1 (0 represents a white pixel, 1 represents a black pixel)
- $\boldsymbol{xmat}$ contains the 2D data (the image) with dimensions 100x40
- Flatten the 2D array as a 1D array
- $\boldsymbol{x}$ contains the 1D data with dimension 4000,
- Apply blurring operation to data $\boldsymbol{x}$, i.e.

$$\boldsymbol{y} = \boldsymbol{A}\,\boldsymbol{x}$$

where $\boldsymbol{A}$ is the blur operator and $\boldsymbol{y}$ is the blurred image

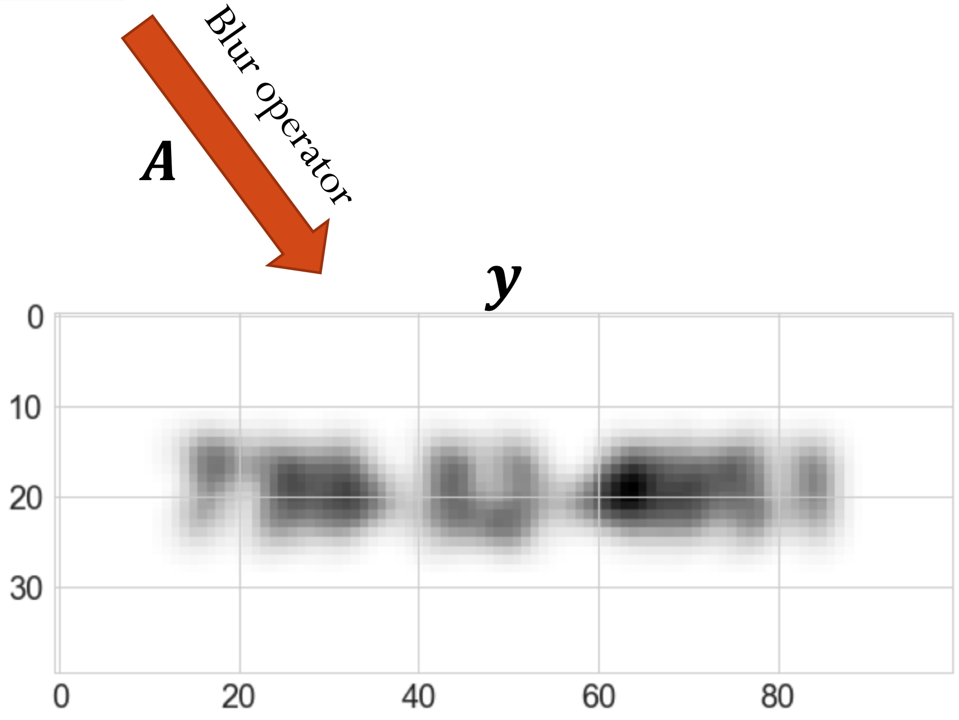
Blur operator





```
plt.imshow(xmat);
plt.axis([20,35,25,15])
```

```
[20, 35, 25, 15]
```



blurred image (4000,)

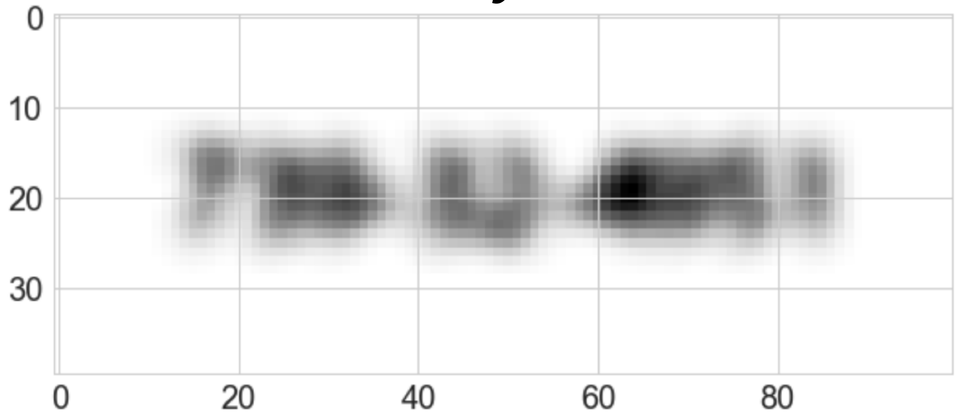
Blur operator (4000,4000)

"original" image (4000,)

$$y = A\,x$$

"Undo" Blur to recover original image

$\boldsymbol{y}$

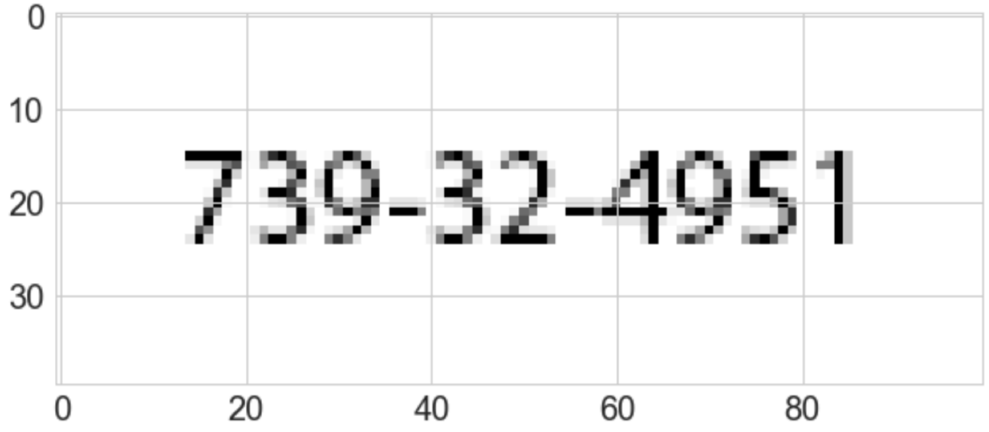


Solve

$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{y}$

for $\boldsymbol{x}$

$\boldsymbol{x}$

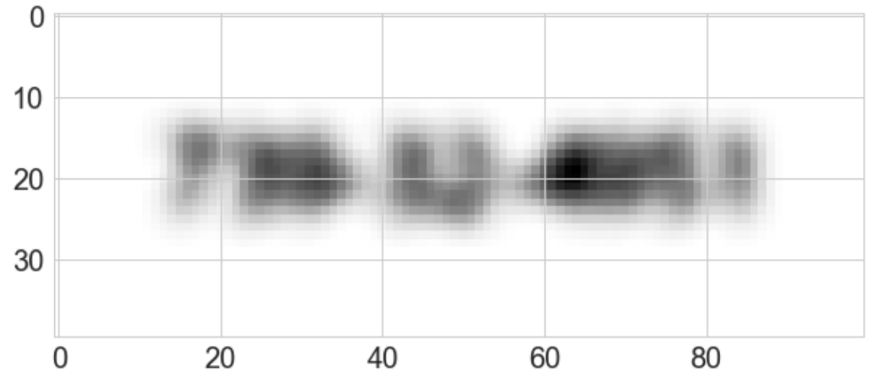


Assumptions:

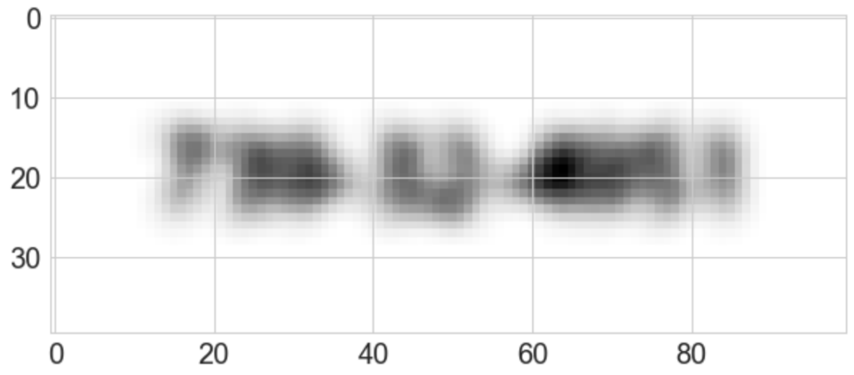
1. we know the blur operator $\boldsymbol{A}$
2. the data set $\boldsymbol{y}$ does not have any noise ("clean data"

”Undo” Blur to recover original image

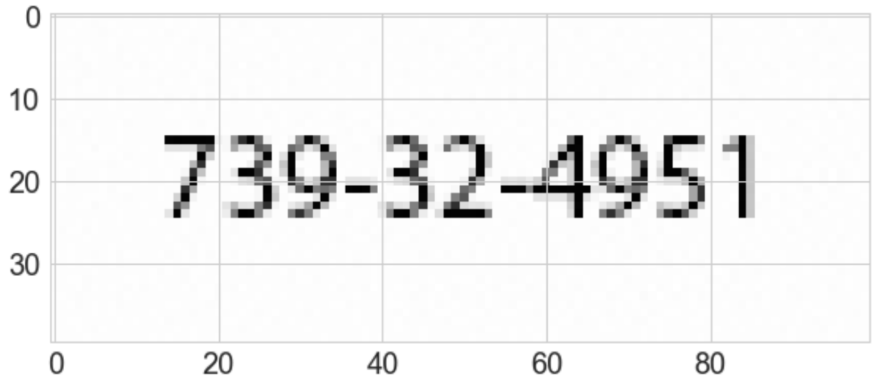
$\boldsymbol{y} + a * 10^{-6}\ (a \in (0,1))$

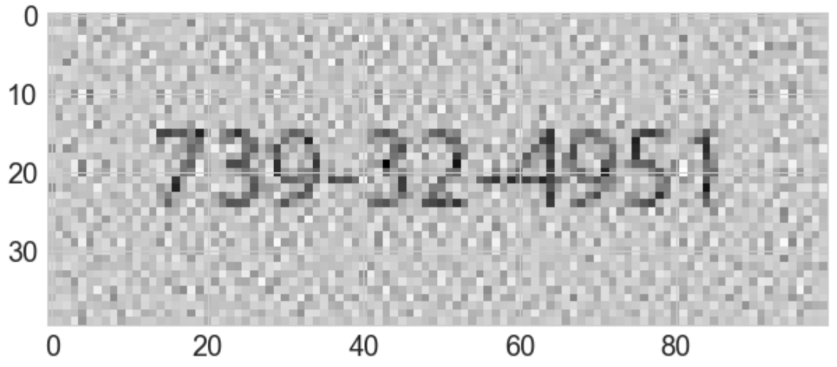


$\boldsymbol{y} + a * 10^{-4}\ (a \in (0,1))$



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





How much noise can we add and still be able to recover meaningful information from the original image? At which point this inverse transformation fails?
We will talk about sensitivity of the “undo” operation later.

Linear System of Equations

How do we actually solve $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$?

We can start with an "easier" system of equations…

Let's consider triangular matrices (lower and upper):

$$\begin{pmatrix} L_{11} & 0 & \dots & 0 \\ L_{21} & L_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & \dots & L_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$$\begin{pmatrix} U_{11} & U_{12} & \dots & U_{1n} \\ 0 & U_{22} & \dots & U_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & U_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Example: Forward-substitution for lower triangular systems

$$\begin{pmatrix} 2 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 \\ 1 & 2 & 6 & 0 \\ 1 & 3 & 4 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 6 \\ 4 \end{pmatrix}$$

$$2\,x_1 = 2 \rightarrow x_1 = 1$$

$$3\,x_1 + 2\,x_2 = 2 \rightarrow x_2 = \frac{2-3}{2} = -0.5$$

$$1\,x_1 + 2\,x_2 + 6\,x_3 = 6 \rightarrow x_3 = \frac{6-1+1}{6} = 1.0$$

$$1\,x_1 + 3\,x_2 + 4\,x_3 + 2\,x_4 = 4 \rightarrow x_3 = \frac{4-1+1.5-4}{2} = 0.25$$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ -0.5 \\ 1.0 \\ 0.25 \end{pmatrix}$$

Triangular Matrices

$$\begin{pmatrix} U_{11} & U_{12} & \dots & U_{1n} \\ 0 & U_{22} & \dots & U_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & U_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Recall that we can also write $\boldsymbol{U}\,\boldsymbol{x} = \boldsymbol{b}$ as a linear combination of the columns of $\boldsymbol{U}$

$$x_1\,\mathbf{U}[:,1] + \ x_2\,\mathbf{U}[:,2] + \cdots + \ x_n\,\mathbf{U}[:,n] = \boldsymbol{b}$$

Hence we can write the solution as

$U_{nn}\ x_n = b_n$

$x_1\,\mathbf{U}[:,1] + \cdots + \ x_{n-1}\,\mathbf{U}[:,n-1] = \boldsymbol{b} - x_n\,\mathbf{U}[:,n] \rightarrow U_{n-1,n-1}\ x_{n-1} = b_{n-1} - U_{n-1,n}\,x_n$

$x_1\,\mathbf{U}[:,1] + \cdots + \ x_{n-2}\,\mathbf{U}[:,n-2] = \boldsymbol{b} - x_n\,\mathbf{U}[:,n] - \ x_{n-1}\,\mathbf{U}[:,n-1]$

Or in general (backward-substitution for upper triangular systems):

$$x_n = b_n/U_{nn} \qquad x_i = \frac{b_i - \sum_{j=i+1}^{n} U_{ij}x_j}{U_{ii}}, \qquad i = n-1, n-2, \dots, 1$$

Triangular Matrices

Forward-substitution for lower-triangular systems:

$$\begin{pmatrix} L_{11} & 0 & \dots & 0 \\ L_{21} & L_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & \dots & L_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$$x_1 = b_1/L_{11} \qquad x_i = \frac{b_i - \sum_{j=1}^{i-1} L_{ij}x_j}{L_{ii}}, \qquad i = 2,3,\dots,n$$

Cost of solving triangular systems

$x_n = b_n/U_{nn}$ $\qquad x_i = \dfrac{b_i - \sum_{j=i+1}^{n} U_{ij}x_j}{U_{ii}}, \qquad i = n-1, n-2, \dots, 1$

n divisions
$n(n-1)/2$ subtractions/additions
$n(n-1)/2$ multiplications

$\Rightarrow$ Computational complexity is $O(n^2)$

$x_1 = b_1/L_{11}$ $\qquad x_i = \dfrac{b_i - \sum_{j=1}^{i-1} L_{ij}x_j}{L_{ii}}, \qquad i = 2,3,\dots,n$

n divisions
$n(n-1)/2$ subtractions/additions
$n(n-1)/2$ multiplications

$\Rightarrow$ Computational complexity is $O(n^2)$

Linear System of Equations

How do we solve $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$ when $\boldsymbol{A}$ is a non-triangular matrix?

We can perform LU factorization: given a $n \times n$ matrix $\boldsymbol{A}$, obtain lower triangular matrix $\boldsymbol{L}$ and upper triangular matrix $\boldsymbol{U}$ such that

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$$

where we set the diagonal entries of $\boldsymbol{L}$ to be equal to 1.

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ L_{21} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & \dots & 1 \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & \dots & U_{1n} \\ 0 & U_{22} & \dots & U_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & U_{nn} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}$$

LU Factorization

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ L_{21} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & \dots & 1 \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & \dots & U_{1n} \\ 0 & U_{22} & \dots & U_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & U_{nn} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{pmatrix}$$

Assuming the LU factorization is know, we can solve the general system

$$\boldsymbol{LU\,x = b}$$

By solving two triangular systems:

$$\boldsymbol{L\,y = b} \quad \xrightarrow{\text{Solve for } \boldsymbol{y}} \quad \text{Forward-substitution with complexity } O(n^2)$$

$$\boldsymbol{U\,x = y} \quad \xrightarrow{\text{Solve for } \boldsymbol{x}} \quad \text{Backward-substitution with complexity } O(n^2)$$

But what is the cost of the LU factorization? Is it beneficial?

2x2 LU Factorization (simple example)

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ L_{21} & 1 \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}$$

$$\begin{pmatrix} \boxed{A_{11} \quad A_{12}} \\ \boxed{A_{21}} \quad A_{22} \end{pmatrix} = \begin{pmatrix} \boxed{U_{11} \qquad U_{12}} \\ \boxed{L_{21}U_{11}} \quad \boxed{L_{21}U_{12} + U_{22}} \end{pmatrix} \longrightarrow U_{11} = A_{21}/U_{11}$$

2) $L_{21} = A_{21}/U_{11}$

3) $U_{22} = A_{22} - L_{21}U_{12}$

Seems quite simple! Can we generalize this for a $n \times n$ matrix $\boldsymbol{A}$?

Computing the Lower-Triangular Factor in LU

1 point

Consider the matrix

$$A = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}$$

and its corresponding LU factorization ($A = LU$), where the lower and upper triangular matrices given respectively by

$$L = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}.$$

l_{21} =

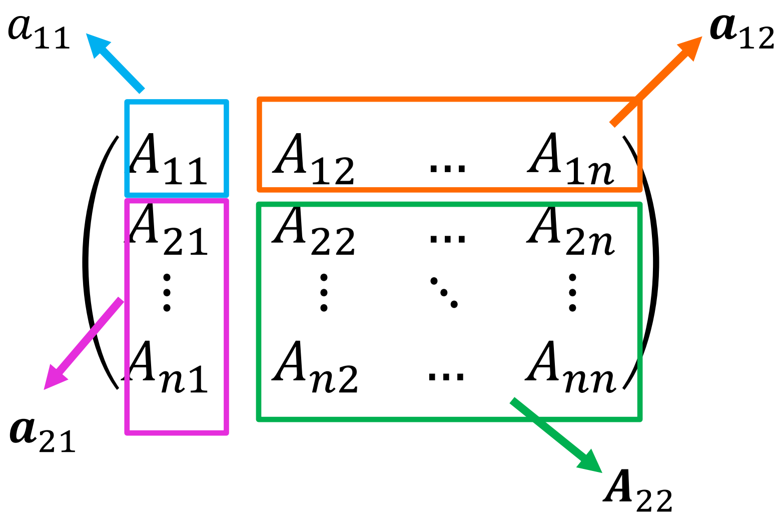
u_{11} =

u_{12} =

u_{22} =

LU Factorization

a_{11}: scalar
$\boldsymbol{a}_{12}$: row vector $(1\times(n-1))$
$\boldsymbol{a}_{21}$: column vector $((n-1)\times 1)$
$\boldsymbol{A}_{22}$: matrix $((n-1)\times(n-1))$


a_{11}
$\boldsymbol{a}_{12}$
$\boldsymbol{a}_{21}$
$\boldsymbol{A}_{22}$
A_{11} A_{12} … A_{1n}
A_{21} A_{22} … A_{2n}
A_{n1} A_{n2} … A_{nn}

$$\begin{pmatrix} \boxed{a_{11} \quad \boldsymbol{a}_{12}} & \\ \boxed{\boldsymbol{a}_{21}} & \boxed{\boldsymbol{A}_{22}} \end{pmatrix} = \begin{pmatrix} \boxed{u_{11} \quad\quad \boldsymbol{u}_{12}} & \\ \boxed{u_{11}\,\boldsymbol{l}_{21}} & \boxed{\boldsymbol{l}_{21}\boldsymbol{u}_{12} + \boldsymbol{L}_{22}\boldsymbol{U}_{22}} \end{pmatrix}$$

1) First row of $\boldsymbol{U}$ is the first row of $\boldsymbol{A}$

$$\boldsymbol{l}_{21} = \frac{1}{u_{11}}\boldsymbol{a}_{21}$$

$$\boldsymbol{A}_{22} = \boldsymbol{l}_{21}\boldsymbol{u}_{12} + \boldsymbol{L}_{22}\boldsymbol{U}_{22}$$

2) First column of $\boldsymbol{L}$ is the first column of $\boldsymbol{A}/\ u_{11}$

3) $\boldsymbol{L}_{22}\boldsymbol{U}_{22} = \overbrace{\boldsymbol{A}_{22} - \boldsymbol{l}_{21}\boldsymbol{u}_{12}}^{\text{Known!}}$

Need another factorization!

LU Factorization

a_{11}: scalar
$\boldsymbol{a}_{12}$: row vector $(1\times(n-1))$
$\boldsymbol{a}_{21}$: column vector $((n-1)\times 1)$
$\boldsymbol{A}_{22}$: matrix $((n-1)\times(n-1))$

a_{11} $\boldsymbol{a}_{12}$

$$\begin{pmatrix} \boxed{A_{11}} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & \boldsymbol{a}_{12} \\ \boldsymbol{a}_{21} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} 1 & \boldsymbol{0} \\ \boldsymbol{l}_{21} & \boldsymbol{L}_{22} \end{pmatrix} \begin{pmatrix} u_{11} & \boldsymbol{u}_{12} \\ \boldsymbol{0} & \boldsymbol{U}_{22} \end{pmatrix}$$

$\boldsymbol{a}_{21}$ $\boldsymbol{A}_{22}$

1) First row of $\boldsymbol{U}$ is the first row of $\boldsymbol{A}$ ✓

$$\begin{pmatrix} a_{11} & \boldsymbol{a}_{12} \\ \boldsymbol{a}_{21} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} u_{11} & \boldsymbol{u}_{12} \\ u_{11}\,\boldsymbol{l}_{21} & \boldsymbol{l}_{21}\boldsymbol{u}_{12} + \boldsymbol{L}_{22}\boldsymbol{U}_{22} \end{pmatrix}$$

2) $\boldsymbol{l}_{21} = \dfrac{1}{u_{11}}\boldsymbol{a}_{21}$

First column of $\boldsymbol{L}$ is the first column of $\boldsymbol{A}/\ u_{11}$ ✓

3) $\boldsymbol{M} = \boldsymbol{L}_{22}\boldsymbol{U}_{22} = \overbrace{\boldsymbol{A}_{22} - \boldsymbol{l}_{21}\boldsymbol{u}_{12}}^{\text{Known!}}$

Need another factorization!

Example

$$\boldsymbol{M} = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 1 & 2 & 3 & 3 \\ 1 & 2 & 6 & 2 \\ 1 & 3 & 4 & 2 \end{pmatrix}$$

1) First row of $\boldsymbol{U}$ is the first row of $\boldsymbol{A}$

2) First column of $\boldsymbol{L}$ is the first column of $\boldsymbol{A}/\ u_{11}$

3) $\boldsymbol{L}_{22}\boldsymbol{U}_{22} = \boldsymbol{A}_{22} - \boldsymbol{l}_{21}\boldsymbol{u}_{12}$

Example

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

$$M = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 1 & -2 & 1 & 2.5 \\ 1 & -2 & 4 & 1.5 \\ 1 & -1 & 2 & 1.5 \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \end{pmatrix} \quad U = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 0 & -2 & 1 & 2.5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$M = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 1 & -2 & 1 & 2.5 \\ 1 & -2 & 3 & -1 \\ 1 & -1 & 1.5 & 0.25 \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0.5 & 1 & 1 & 0 \\ 0.5 & 0.5 & \boxed{0.5} & 0 \end{pmatrix} \quad U = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 0 & -2 & 1 & 2.5 \\ 0 & 0 & 3 & \boxed{-1} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$M = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 1 & -2 & 1 & 2.5 \\ 1 & -2 & 3 & -1 \\ 1 & -1 & 1.5 & \boxed{0.75} \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0.5 & 1 & 1 & 0 \\ 0.5 & 0.5 & 0.5 & 1 \end{pmatrix} \quad U = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 0 & -2 & 1 & 2.5 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 0 & 0.75 \end{pmatrix}$$

Algorithm: LU Factorization of matrix A

```
## 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:])
```

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:

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

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

Cost of LU factorization

Side note:

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

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

```python
## 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:])
```

Number of divisions: $(n-1)+(n-2)+\cdots+1 = n(n-1)/2$

Number of multiplications $(n-1)^2+(n-2)^2+\ldots+(1)^2 = \frac{n^3}{3}-\frac{n^2}{2}+\frac{n}{6}$

Number of subtractions: $(n-1)^2+(n-2)^2+\ldots+(1)^2 = \frac{n^3}{3}-\frac{n^2}{2}+\frac{n}{6}$

$\Downarrow$

Computational complexity is $O(n^3)$

Demo "Complexity of Mat-Mat multiplication and LU"

Solving linear systems

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

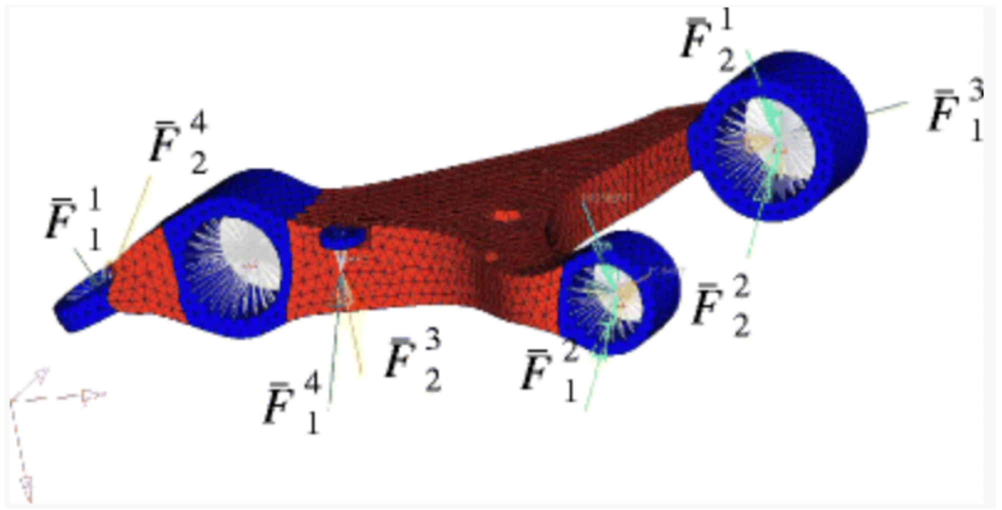
1) Factorize the matrix $\boldsymbol{A}$: $\boldsymbol{A} = \boldsymbol{LU}$ (complexity $O(n^3)$)

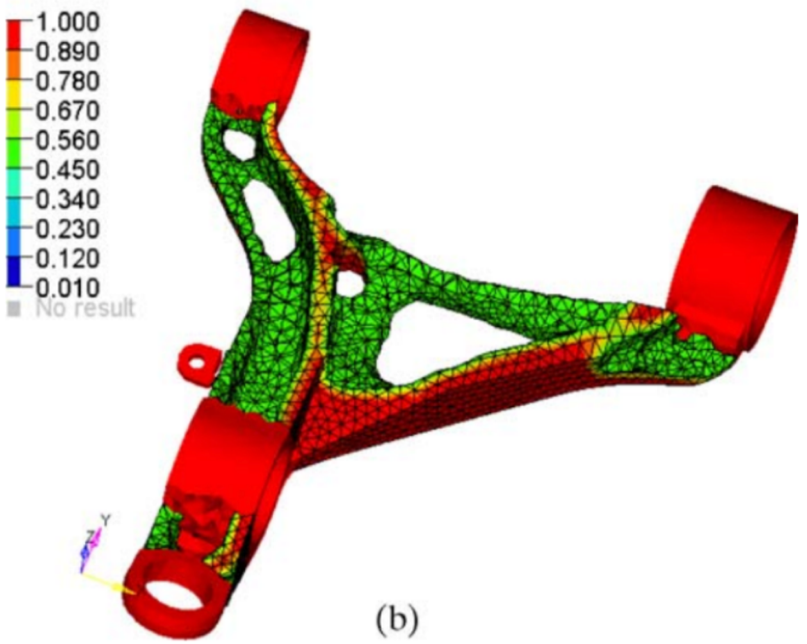
2) Solve $\boldsymbol{L}\,\boldsymbol{y} = \boldsymbol{b}$ (complexity $O(n^2)$)

3) Solve $\boldsymbol{U}\,\boldsymbol{x} = \boldsymbol{y}$ (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…)

Example: Optimization of automotive control arm





Find the distribution of material inside the design space $(\boldsymbol{d})$ that maximizes the stiffness, i.e.,

$\min \boldsymbol{U}^T\boldsymbol{F}$ where $\boldsymbol{K}(\boldsymbol{d})\,\boldsymbol{U} = \boldsymbol{F}$ ($\boldsymbol{U}$: displacement vector, $\boldsymbol{F}$: load vector, $\boldsymbol{K}$: stiffness matrix)

Solve the linear system of equations

$$\boldsymbol{K}\,\boldsymbol{U} = \boldsymbol{F}$$

for the load vector $\boldsymbol{F}$. What if we have many different loading conditions (pothole, hitting a curb, breaking, etc)?

Iclicker question

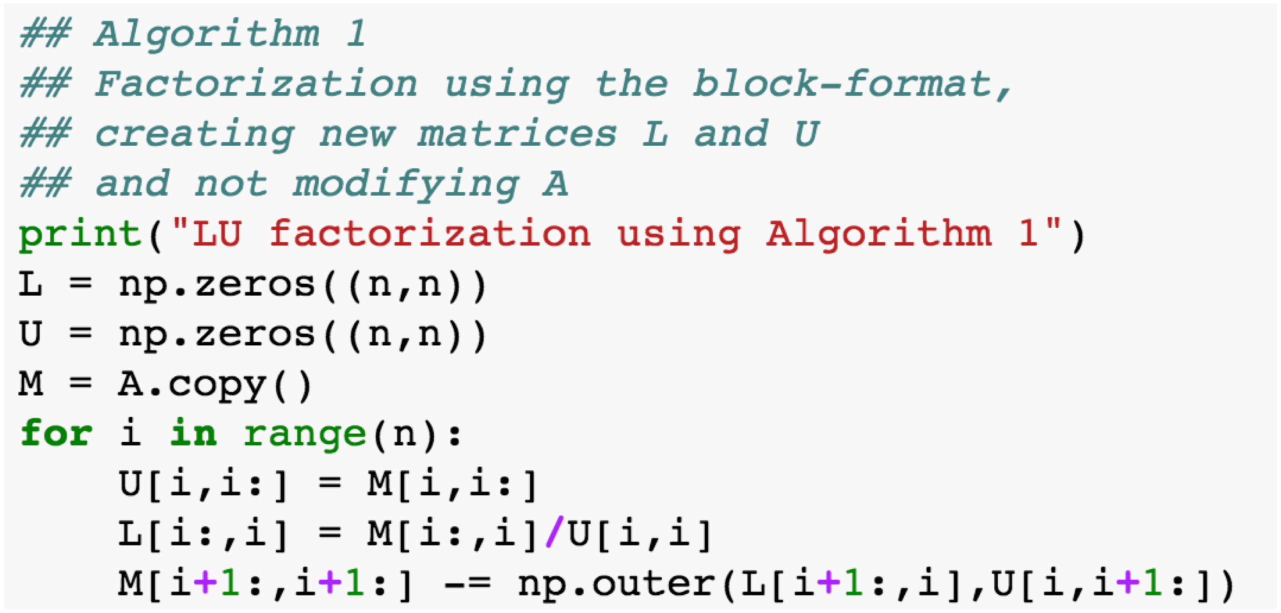
Let’s assume that when solving the system of equations $\boldsymbol{K}\,\boldsymbol{U} = \boldsymbol{F}$, we observe the following:

- When the stiffness matrix has dimensions (100,100), computing the LU factorization takes about 1 second and each solve (forward + backward substitution) takes about 0.01 seconds.

Estimate the total time it will take to find the displacement response corresponding to 10 different load vectors $\boldsymbol{F}$ when the stiffness matrix has dimensions (1000,1000)?

$A)\ {\sim}10\ seconds$
$B)\ {\sim}10^2\ seconds$
$C)\ {\sim}10^3\ seconds$
$D)\ {\sim}10^4\ seconds$
$E)\ {\sim}10^5\ seconds$

What can go wrong with the previous algorithm?



```
## 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:])
```

If division by zero occurs, LU factorization fails.

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

What can go wrong with the previous algorithm?

Demo "Little c"

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

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

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.

$$\begin{pmatrix} a_{11} & \boldsymbol{a}_{12} \\ \boxed{\boldsymbol{a}_{21}} & \boldsymbol{A}_{22} \end{pmatrix} = \begin{pmatrix} u_{11} & \boldsymbol{u}_{12} \\ u_{11}\,\boldsymbol{l}_{21} & \boldsymbol{l}_{21}\boldsymbol{u}_{12} + \boldsymbol{L}_{22}\boldsymbol{U}_{22} \end{pmatrix}$$

$\searrow$ Find the largest entry (in magnitude)

LU Factorization with Partial Pivoting

$$\boldsymbol{A} = \boldsymbol{PLU}$$

where $\boldsymbol{P}$ is a permutation matrix

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b} \;\rightarrow \boldsymbol{PLU}\,\boldsymbol{x} = \boldsymbol{b} \rightarrow \boldsymbol{LU}\,\boldsymbol{x} = \boldsymbol{P}^T\boldsymbol{b}$$

Then solve two triangular systems:

$\boldsymbol{L}\,\boldsymbol{y} = \boldsymbol{P}^T\boldsymbol{b}$ (Solve for $\boldsymbol{y}$)

$\boldsymbol{U}\,\boldsymbol{x} = \boldsymbol{y}$ (Solve for $\boldsymbol{x}$)

Example

$$\boldsymbol{A} = \boldsymbol{M} = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 1 & 2 & 3 & 3 \\ 1 & 2 & {\color{red}3} & 2 \\ 1 & 3 & 4 & 2 \end{pmatrix} \quad \boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \boxed{0.5} & 0 & 0 & 0 \\ \boxed{0.5} & 0 & 0 & 0 \\ \boxed{0.5} & 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{U} = \begin{pmatrix} 2 & \boxed{8} & \boxed{4} & \boxed{1} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{M} = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 1 & \boxed{-2} & \boxed{1} & \boxed{2.5} \\ 1 & \boxed{-2} & \boxed{1} & \boxed{1.5} \\ 1 & \boxed{-1} & \boxed{2} & \boxed{1.5} \end{pmatrix} \quad \boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0.5 & \boxed{1} & 0 & 0 \\ 0.5 & \boxed{0.5} & 0 & 0 \end{pmatrix} \quad \boldsymbol{U} = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 0 & -2 & \boxed{1} & \boxed{2.5} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{M} = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 1 & -2 & 1 & 2.5 \\ 1 & -2 & \boxed{0} & \boxed{-1} \\ 1 & -1 & \boxed{1.5} & \boxed{0.25} \end{pmatrix} \quad \boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0.5 & \boxed{0.5} & 0 & 0 \\ 0.5 & \boxed{1.0} & 0 & 0 \end{pmatrix} \quad \boldsymbol{U} = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 0 & -2 & \boxed{1} & \boxed{2.5} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{M} = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 1 & -2 & 1 & 2.5 \\ 1 & -1 & \boxed{1.5} & \boxed{0.25} \\ 1 & -2 & \boxed{0} & \boxed{-1} \end{pmatrix} \quad \boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0.5 & 0.5 & 1.0 & 0 \\ 0.5 & 1.0 & \boxed{0} & 1.0 \end{pmatrix} \quad \boldsymbol{U} = \begin{pmatrix} 2 & 8 & 4 & 1 \\ 0 & -2 & 1 & 2.5 \\ 0 & 0 & 1.5 & \boxed{0.25} \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

Demo “Pivoting example”

$$\boldsymbol{A} = \begin{pmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{pmatrix}$$

$$\bar{\boldsymbol{A}} = \boldsymbol{PA} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 1 & 0 \\ 4 & 3 & 3 & 1 \\ 8 & 7 & 9 & 5 \\ 6 & 7 & 9 & 8 \end{pmatrix} = \begin{pmatrix} 8 & 7 & 9 & 5 \\ 4 & 3 & 3 & 1 \\ 2 & 1 & 1 & 0 \\ 6 & 7 & 9 & 8 \end{pmatrix}$$

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \\ 0.25 & 0 & 0 & 0 \\ 0.75 & 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{U} = \begin{pmatrix} 8 & 7 & 9 & 5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{l}_{21}\boldsymbol{u}_{12} = \begin{pmatrix} 3.5 & 4.5 & 2.5 \\ 1.75 & 2.25 & 1.25 \\ 5.25 & 6.75 & 3.75 \end{pmatrix}$$

$$\bar{\boldsymbol{A}} - \boldsymbol{l}_{21}\boldsymbol{u}_{12} = \begin{pmatrix} 8 & 7 & 9 & 5 \\ 4 & -0.5 & -1.5 & -1.5 \\ 2 & -0.75 & -1.25 & -1.25 \\ 6 & 1.75 & 2.25 & 4.25 \end{pmatrix}$$

Demo "Pivoting example"

$$\overline{\boldsymbol{A}} = \overline{\boldsymbol{A}} - \boldsymbol{l}_{21}\boldsymbol{u}_{12} = \begin{pmatrix} 8 & 7 & 9 & 5 \\ 4 & \boxed{\begin{matrix}-0.5 \\ -0.75 \\ 1.75\end{matrix}} & \begin{matrix}-1.5 \\ -1.25 \\ 2.25\end{matrix} & \begin{matrix}-1.5 \\ -1.25 \\ 4.25\end{matrix} \end{pmatrix}$$

$$\overline{\boldsymbol{A}} = \boldsymbol{P}\overline{\boldsymbol{A}} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 8 & 7 & 9 & 5 \\ 6 & 1.75 & 2.25 & 4.25 \\ 2 & -0.75 & -1.25 & -1.25 \\ 4 & -0.5 & -1.5 & -1.5 \end{pmatrix} = \begin{pmatrix} 8 & 7 & 9 & 5 \\ 6 & \boxed{\begin{matrix}1.75 \\ -0.75 \\ -0.5\end{matrix}} & \begin{matrix}2.25 \\ -1.25 \\ -1.5\end{matrix} & \begin{matrix}4.25 \\ -1.25 \\ -1.5\end{matrix} \end{pmatrix}$$

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.75 & 1 & 0 & 0 \\ 0.25 & -0.428 & 0 & 0 \\ 0.5 & -0.285 & 0 & 0 \end{pmatrix} \quad \boldsymbol{U} = \begin{pmatrix} 8 & 7 & 9 & 5 \\ 0 & 1.75 & 2.25 & 4.25 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \boldsymbol{l}_{21}\boldsymbol{u}_{12} = \begin{pmatrix} -0.963 & -1.819 \\ -0.6412 & -1.2112 \end{pmatrix}$$

$$\overline{\boldsymbol{A}} = \overline{\boldsymbol{A}} - \boldsymbol{l}_{21}\boldsymbol{u}_{12} = \begin{pmatrix} 8 & 7 & 9 & 5 \\ 6 & 1.75 & 2.25 & 4.25 \\ 2 & -0.75 & \boxed{\begin{matrix}-0.287 \\ -0.8587\end{matrix}} & \begin{matrix}0.569 \\ -0.2887\end{matrix} \\ 4 & -0.5 & & \end{pmatrix}$$

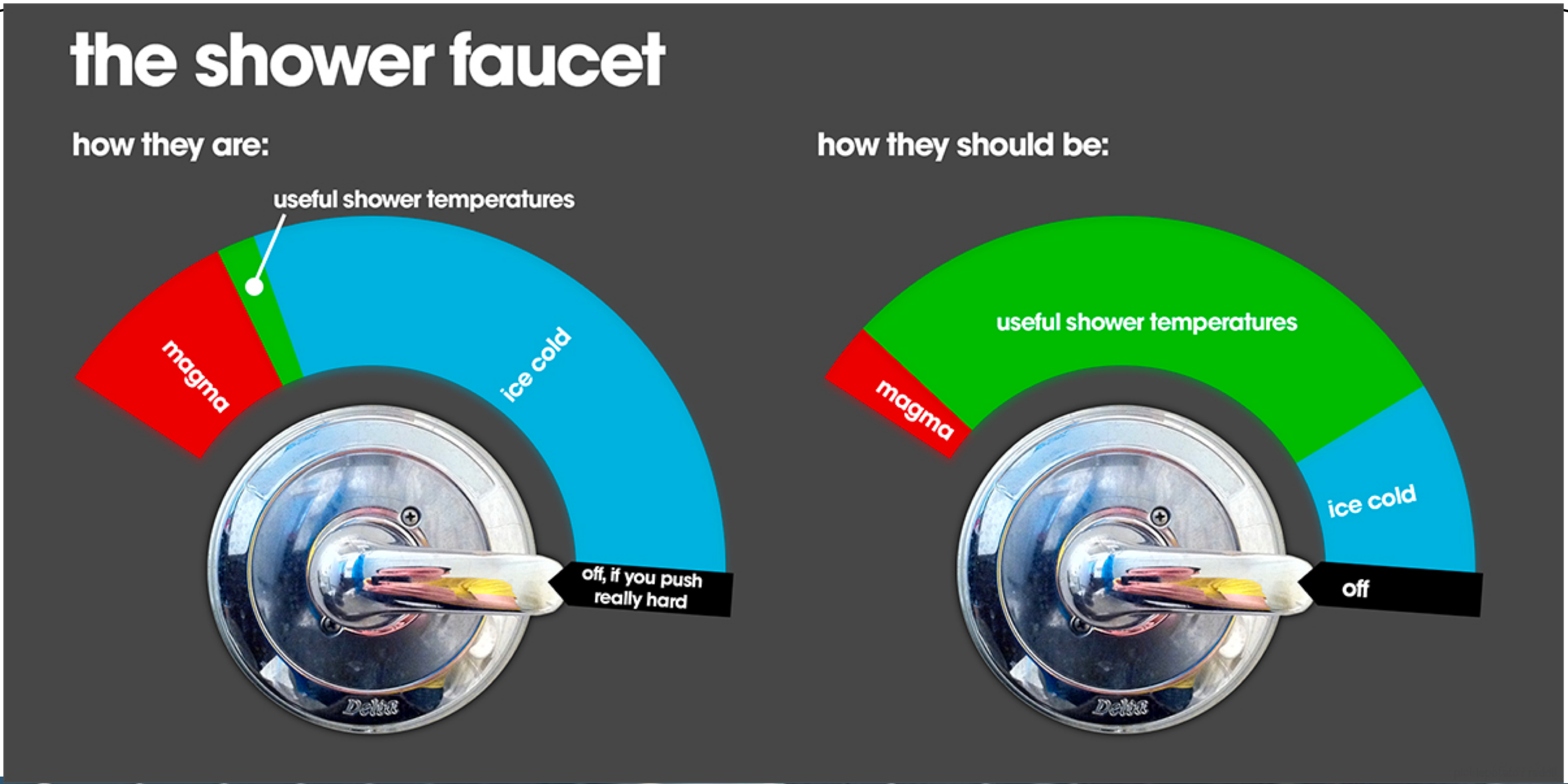
Demo "Pivoting example"

$$\bar{\boldsymbol{A}} = \bar{\boldsymbol{A}} - \boldsymbol{l}_{21}\boldsymbol{u}_{12} = \begin{pmatrix} 8 & 7 & 9 & 5 \\ 6 & 1.75 & 2.25 & 4.25 \\ 2 & -0.75 & \boxed{-0.287} & 0.569 \\ 4 & -0.5 & \boxed{-0.8587} & -0.2887 \end{pmatrix}$$

$$\bar{\boldsymbol{A}} = \boldsymbol{P}\bar{\boldsymbol{A}} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 8 & 7 & 9 & 5 \\ 6 & 1.75 & 2.25 & 4.25 \\ 2 & -0.75 & -0.287 & 0.569 \\ 4 & -0.5 & -0.8587 & -0.2887 \end{pmatrix} = \begin{pmatrix} 8 & 7 & 9 & 5 \\ 6 & 1.75 & 2.25 & 4.25 \\ 4 & -0.5 & \boxed{-0.8587} & -0.2887 \\ 2 & -0.75 & \boxed{-0.287} & 0.569 \end{pmatrix}$$

$$\boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.75 & 1 & 0 & 0 \\ \color{red}{0.5} & \color{red}{-0.285} & 1 & 0 \\ \color{red}{0.25} & \color{red}{-0.428} & \boxed{0.334} & 0 \end{pmatrix} \qquad \boldsymbol{U} = \begin{pmatrix} 8 & 7 & 9 & 5 \\ 0 & 1.75 & 2.25 & 4.25 \\ 0 & 0 & -0.86 & \boxed{-0.29} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\boxed{\begin{aligned} &\boldsymbol{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.75 & 1 & 0 & 0 \\ 0.5 & -0.285 & 1 & 0 \\ 0.25 & -0.428 & 0.334 & 1 \end{pmatrix} \qquad \boldsymbol{U} = \begin{pmatrix} 8 & 7 & 9 & 5 \\ 0 & 1.75 & 2.25 & 4.25 \\ 0 & 0 & -0.86 & -0.29 \\ 0 & 0 & 0 & 0.67 \end{pmatrix} \\ &\boldsymbol{P} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{aligned}}$$


the shower faucet
how they are:
useful shower temperatures
magma
ice cold
off, if you push really hard
how they should be:
useful shower temperatures
magma
ice cold
off

WHAT IT LOOKS LIKE

WHAT IT FEELS LIKE

Numerical experiments

Input has uncertainties:

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

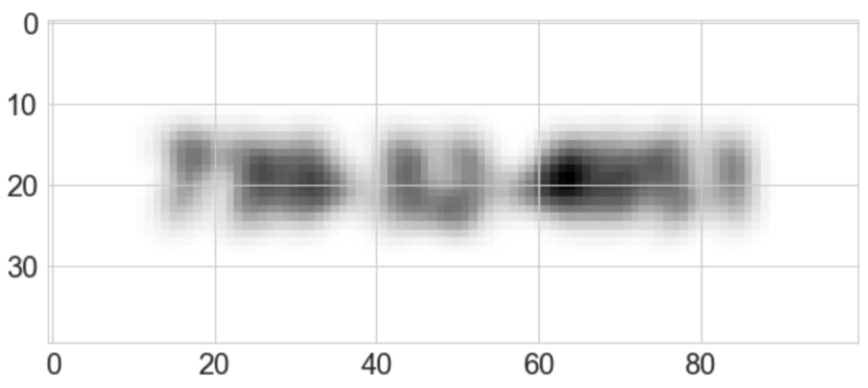
Once you select your numerical method , how much error should you expect to see in your **output?**

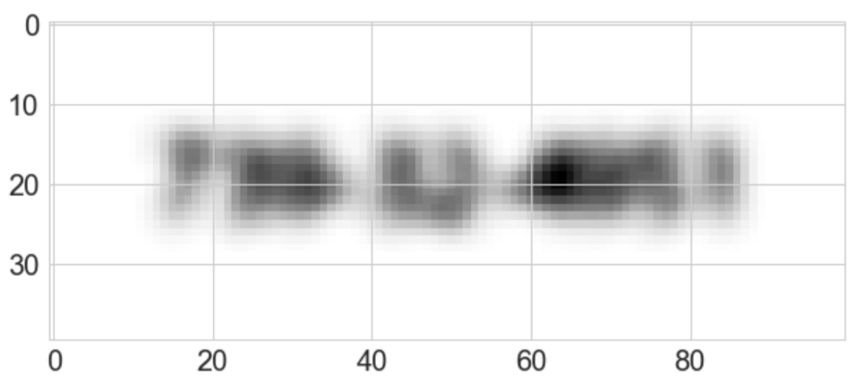
Is your method sensitive to errors (perturbation) in the input?

Demo "HilbertMatrix-ConditionNumber"

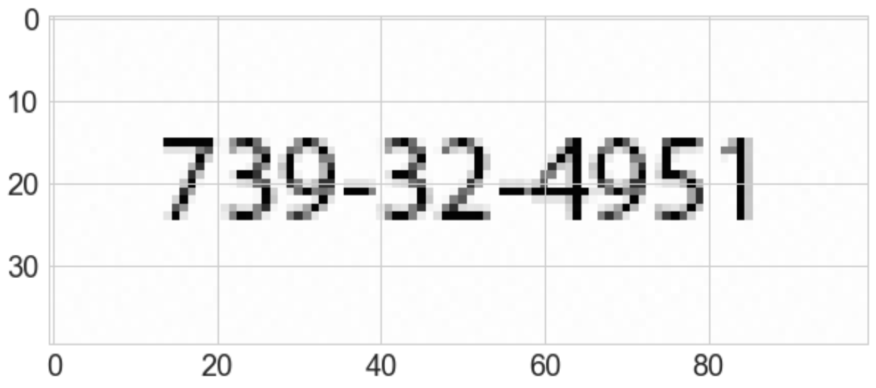
$\boldsymbol{b} + a * 10^{-6}\ (a \in (0,1))$

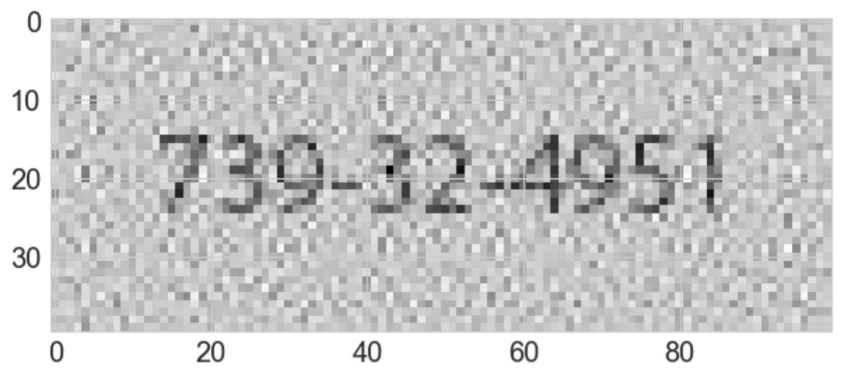
$\boldsymbol{b} + a * 10^{-4}\ (a \in (0,1))$





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 $\Delta\boldsymbol{x}$?

$$\frac{\text{Output Relative error}}{\text{Input Relative error}} = \frac{\|\Delta\boldsymbol{x}\|/\|\boldsymbol{x}\|}{\|\Delta\boldsymbol{b}\|/\|\boldsymbol{b}\|} = \frac{\|\Delta\boldsymbol{x}\|\,\|\boldsymbol{b}\|}{\|\Delta\boldsymbol{b}\|\,\|\boldsymbol{x}\|}$$

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

$$\frac{\text{Output Relative error}}{\text{Input Relative error}} = \frac{\|\boldsymbol{A}^{-1}\,\Delta\boldsymbol{b}\|\,\|\boldsymbol{A}\,\boldsymbol{x}\|}{\|\Delta\boldsymbol{b}\|\,\|\boldsymbol{x}\|} \leq \frac{\|\boldsymbol{A}^{-1}\|\|\Delta\boldsymbol{b}\|\,\|\boldsymbol{A}\|\|\boldsymbol{x}\|}{\|\Delta\boldsymbol{b}\|\,\|\boldsymbol{x}\|}$$

$$\frac{\|\Delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \|\boldsymbol{A}^{-1}\|\,\|\boldsymbol{A}\|\,\frac{\|\Delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}$$

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:

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

Demo “HilbertMatrix-ConditionNumber”

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 $\boldsymbol{A}$:

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

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:

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

Iclicker question

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

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) $cond(\boldsymbol{A}) = \text{large numbers}$

Condition number

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

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:

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

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

Recall Induced Matrix Norms

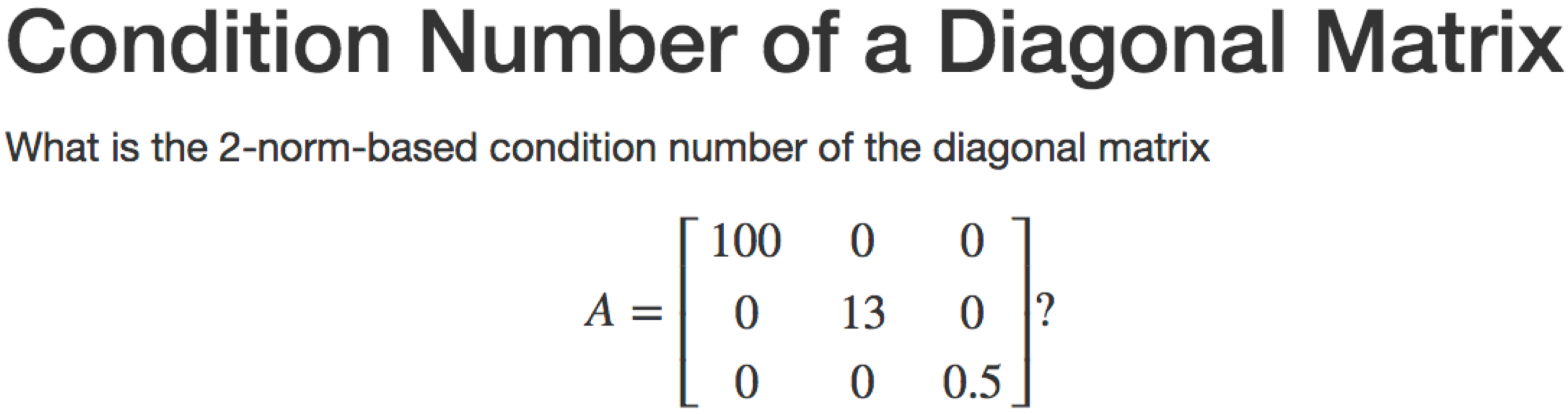
$$\|\boldsymbol{A}\|_1 = \max_j \sum_{i=1}^{n} |A_{ij}|$$ Maximum absolute column sum of the matrix $\boldsymbol{A}$

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

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 $\boldsymbol{D}$, $cond(\boldsymbol{D}) = \frac{\boldsymbol{max}|d_{\boldsymbol{i}}|}{\boldsymbol{min}|d_{\boldsymbol{i}}|}$

“Little c” demo

Discuss what happens when c is ”close” to zero
What are the eigenvalues of triangular matrices?
We need to pivot!

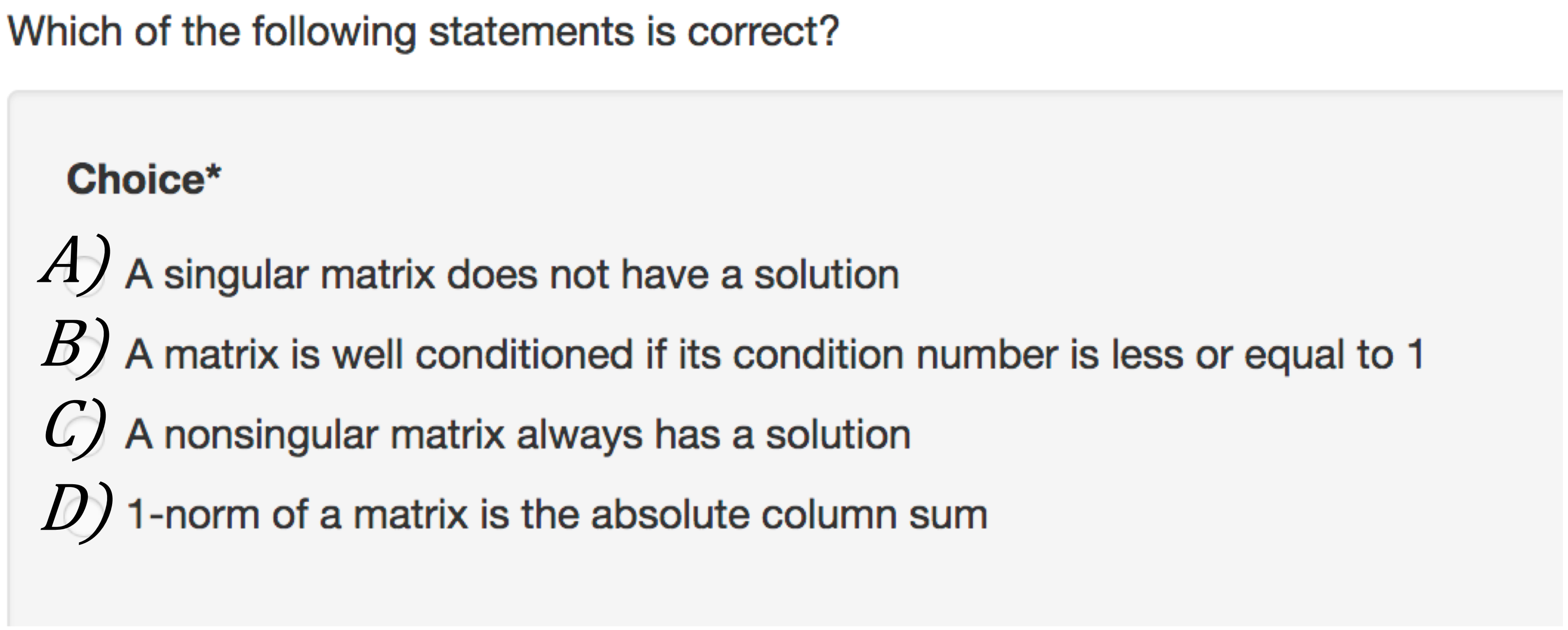
Remarks:
The need for pivoting does not depend on whether the matrix is singular.
A non-singular matrix always has a solution.
A singular matrix may not have a solution, or may have infinitely many solutions.

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

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 $\boldsymbol{D}$, $cond(\boldsymbol{D}) = \frac{\boldsymbol{max}|d_{\boldsymbol{i}}|}{\boldsymbol{min}|d_{\boldsymbol{i}}|}$

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

Condition Number of Orthogonal Matrices

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

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

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

$$\widehat{\boldsymbol{x}} = (\boldsymbol{x} + \Delta\boldsymbol{x})$$

which could be the solution of

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

And the **error vector** as

$$\boldsymbol{e} = \Delta\boldsymbol{x} = \widehat{\boldsymbol{x}} - \boldsymbol{x}$$

We can write the **residual vector** as

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\,\widehat{\boldsymbol{x}}$$

Demo “Rule of Thumb on Conditioning”

Relative residual: $\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{A}\|\|\boldsymbol{x}\|}$ **(How well the solution satisfies the problem)**

Relative error: $\frac{\|\Delta\boldsymbol{x}\ \|}{\|\boldsymbol{x}\|}$ **(How close the approximated solution is from the exact one)**

When solving a system of linear equations via LU with partial pivoting, the relative residual is guaranteed to be small!

Residual versus error

Let us first obtain the norm of the error:

$$\|\Delta \boldsymbol{x}\| = \|\,\widehat{\boldsymbol{x}} - \boldsymbol{x}\| = \left\|\boldsymbol{A}^{-1}\boldsymbol{A}\,\widehat{\boldsymbol{x}} - \boldsymbol{A}^{-1}\boldsymbol{b}\right\| = \left\|\boldsymbol{A}^{-1}(\boldsymbol{A}\,\widehat{\boldsymbol{x}} - \boldsymbol{b})\right\| = \left\|-\boldsymbol{A}^{-1}\,\boldsymbol{r}\right\|$$

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

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

For well-conditioned matrices, small relative residual implies small relative error.

Residual versus error

Without loss of generality, let us assume that the perturbed solution $\hat{\boldsymbol{x}}$ satisfies

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

Then the **residual vector** becomes

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\,\hat{\boldsymbol{x}} = \boldsymbol{b} - (\boldsymbol{b} - \boldsymbol{E}\,\hat{\boldsymbol{x}}) = \boldsymbol{E}\,\hat{\boldsymbol{x}}$$

And the norm of the residual is $\|\boldsymbol{r}\| = \|\boldsymbol{E}\,\hat{\boldsymbol{x}}\| \leq \|\boldsymbol{E}\|\,\|\hat{\boldsymbol{x}}\|$. After normalizing the residual norm, we obtain

$$\frac{\|\boldsymbol{r}\|}{\|\boldsymbol{A}\|\|\hat{\boldsymbol{x}}\|} \leq \frac{\|\boldsymbol{E}\|}{\|\boldsymbol{A}\|} \leq c\;\epsilon_m$$

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**.

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 $\widehat{\boldsymbol{x}}$.

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

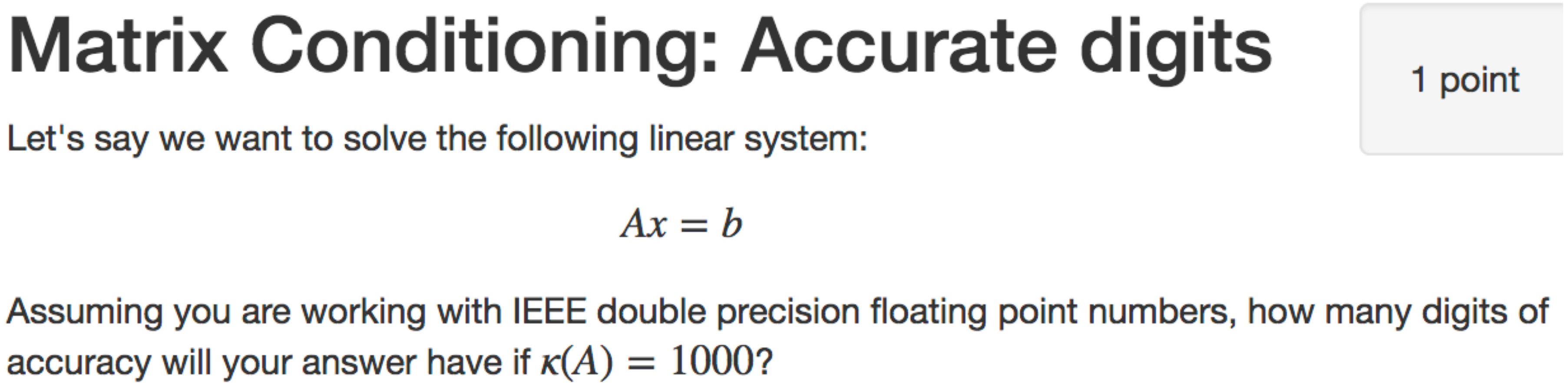
and $cond(\boldsymbol{A}) = \boldsymbol{10^W}$,

then the elements of the solution vector $\widehat{\boldsymbol{x}}$ 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