Lecture 17: Stability of linear equations (class)

Admin:

Recall: Singular-value decomposition

$$A = \sum_i \sigma_i \vec{v}_i \vec{u}_i^T$$

condition number $\kappa(A) = \frac{\text{largest } \sigma}{\text{smallest } \sigma}$

SVD applications ...

* Solving linear equations $A\vec{x} = \vec{b}$
  - What is the sensitivity, e.g., to numerical errors?
  - Find the shortest solution
  - When there is no solution, find x to minimize $\|A\vec{x} - \vec{b}\|$
    - Least-squares regression analysis
* Rank minimization
  - Principal Component Analysis (PCA)

Data mining, clustering, recommendation systems, ...

NUMERICAL STABILITY OF $A\vec{x} = \vec{b}$

systems of linear equations

Reading:
Meyer 1.6
5.12.1-2
Strang 7.2

Example:

Let $A = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix}$, $\vec{b} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$

How sensitive is the solution $\vec{x} = A^{-1}\vec{b}$ to small errors (eg., numerical errors) in $\vec{b}$?

$\vec{b}' = \vec{b} + \vec{\delta}$

$A^{-1}\vec{b}$ $A^{-1}\vec{b}'$ ↑error

$\|\vec{x} - \vec{x}'\|$

$= \|A^{-1}b - A^{-1}b'\|$

$= \|A^{-1}(b - b')\|$

$= \|A^{-1}\vec{\delta}\|$

$\leq \|A^{-1}\| \cdot \|\vec{\delta}\|$ ie $\delta \| v_2$

with equality if $\delta \cdot v_2 = \|\delta\|$

$\|A\|$

$A = \sigma_1 \vec{v}_1 \vec{u}_1^T + \sigma_2 \vec{v}_2 \vec{u}_2^T$ $\sigma_1 \geq \sigma_2$

$A^{-1} = \frac{1}{\sigma_1} u_1 v_1^T + \frac{1}{\sigma_2} u_2 v_2^T$ $\frac{1}{\|A^{-1}\|}$

Let $B = \begin{pmatrix} 1+10^{-4} & 1 \\ 1 & 1 \end{pmatrix}$

$.0001x + y = b_1$

$x + y = b_2$

How sensitive is $B^{-1}\vec{b}$ to small errors in $\vec{b}$?

$B\vec{x} = \vec{b}$

Why not experiment?

```
>> A = [10^-4 1; 1 1];
B = [1+10^-4 1; 1 1];
b = [5;5];
berror = 10^(-5) * randn(2,1);
>> (A\b) - (A\(b+berror))

ans =

   1.0304e-05
  -7.2699e-06
```

← pretty small
solution not too sensitive — good!

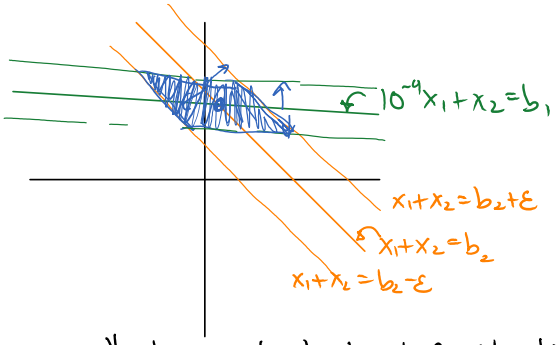
```
>> (B\b) - (B\(b+berror))

ans =

  -1.0303e-01
   1.0304e-01
```

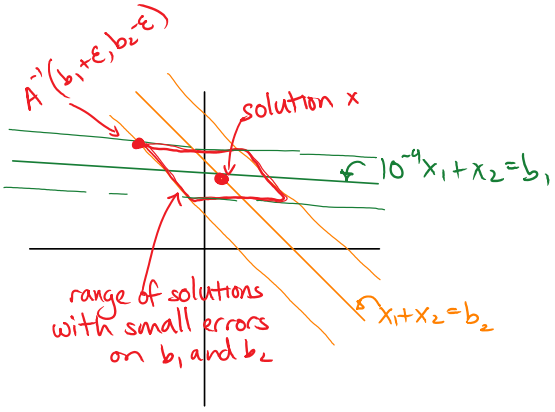
← huge error!

Why?

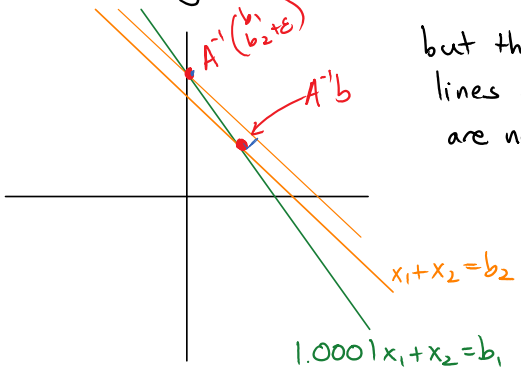
① Geometrically

$Ax=b \iff 10^{-4}x_1+x_2=b_1$, $x_1+x_2=b_2$





small changes to b_1, b_2 shift the lines by small amounts



but the intersection of the lines can shift a lot, if they are nearly parallel

② Algebraic explanation for stability/instability

$$(*)\quad \|A^{-1}b - A^{-1}(b+\delta)\| = \|A^{-1}\delta\| \le \|A^{-1}\| \cdot \|\delta\|$$

$\Rightarrow$ If $\|A^{-1}\|$ is large, then a small error in b can be amplified to a large error in $x=A^{-1}b$

Continuing the previous example:

```
octave:7> norm(A^-1)
ans =  1.6182
octave:8> norm(B^-1)
ans =  2.0001e+04
```

In the worst case, the upper bound in (*) is reached!

$$A = \lambda_1 u_1 v_1^T + \lambda_2 u_2 v_2^T + \cdots + \lambda_n u_n v_n^T$$

SVD, with $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n$

$$A^{-1} = \frac{1}{\lambda_1} v_1 u_1^T + \frac{1}{\lambda_2} v_2 u_2^T + \cdots + \frac{1}{\lambda_n} v_n u_n^T$$

$\Rightarrow \|A^{-1}\| = \lambda_n^{-1}$

and if error vector $\vec{\delta} = \varepsilon \vec{u}_n$,

then $A^{-1}\vec{\delta} = \frac{\varepsilon}{\lambda_n} v_n$

$\Rightarrow \|A^{-1}\delta\| = \|A^{-1}\| \cdot \|\delta\|$

Easy solution?

Instead of solving $Ax = b$,

$A\vec{x} = \vec{b}$

$B = 1000A$

$B\vec{x} = 1000\vec{b}$

solve $(1000A)y = b \Rightarrow y = \frac{1}{1000}A^{-1}b$

$\|B^{-1}\| = \frac{1}{1000}\|A^{-1}\|$

and then set $x = 1000y$

$\|(1000A)^{-1}\| = \frac{1}{1000}\|A^{-1}\| \Rightarrow$ errors in b now lead to much smaller errors in y

$\rightsquigarrow$ But this is cheating; the error in x will be the same

$\Rightarrow$ **Definition**: The condition number of a matrix A is

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|$$

equivalently, $= \frac{\text{largest singular value of } A}{\text{smallest singular value}} \geq 1$

Observe: This is invariant under scaling:

$$\|(1000A)\| \cdot \|(1000A)^{-1}\| = \|A\| \cdot \|A^{-1}\|.$$

moral:

HIGH CONDITION NUMBER $\longleftrightarrow$ $Ax = b$ UNSTABLE

$\frac{\max}{\min}$ sing. value

Example:

$\kappa\left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right) = 1$

$\kappa\left(\underset{\text{unitary}}{U}\right) = 1$ unitary $\longleftrightarrow$ singular values all $= 1$

$\kappa\left(\begin{pmatrix} 10^3 & 0 \\ 0 & 1 \end{pmatrix}\right) = 10^3 = \kappa\left(\begin{pmatrix} 1 & 0 \\ 0 & 10^{-3} \end{pmatrix}\right)$

① consider sensitivity to errors in $\vec{b}$

Formally: We have

$x = A^{-1}\vec{b}$ $\qquad x' = A^{-1}(b+\delta)$

$\|x - x'\| = \|A^{-1}\vec{\delta}\| \leq \|A^{-1}\| \cdot \|\delta\|$

and $\|b\| = \|A \cdot A^{-1}b\| \leq \|A\| \cdot \|A^{-1}b\|$

$$\frac{\text{error} \rightarrow \|A^{-1}\delta\|}{\text{correct answer} \rightarrow \|A^{-1}b\|} \leq \underbrace{\|A\| \cdot \|A^{-1}\|}_{\kappa(A)\ \text{condition \#}} \cdot \frac{\|\delta\|}{\|b\|}$$

relative error in $\vec{x}$ $\qquad$ relative error in $\vec{b}$

$$\Rightarrow \frac{\|A^{-1}\delta\|}{\|A^{-1}b\|} \le \|A\|\cdot\|A^{-1}\| \cdot \frac{\|\delta\|}{\|b\|}$$

- $\frac{\|A^{-1}\delta\|}{\|A^{-1}b\|}$: relative error in the solution ($\|A^{-1}\delta\|/\|x\|$)
- $\|A\|\cdot\|A^{-1}\|$: condition # of A
- $\frac{\|\delta\|}{\|b\|}$: relative error in b

$\Rightarrow$ condition number bounds the relative errors

Three sources of error in solving $Ax=b$

$$\frac{\|A^{-1}\delta\|}{\|A^{-1}x\|} \le \kappa \cdot \frac{\|\delta\|}{\|b\|}$$

1. Errors in the data $\vec{b}$
2. Errors in the linear solve routine

Even if b is known exactly, errors such as floating point errors might arise and accumulate while solving the equations.

$\kappa(B) \approx 2\cdot 10^4$

Example:

$$B = \begin{pmatrix} 1+10^{-4} & 1 \\ 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 5 \\ 5 \end{pmatrix} \text{ as above}$$

Say you solve these and got $x = \begin{pmatrix} .5 \\ 4.5 \end{pmatrix}$

```
>> B * [.5;4.5]

ans =

   5.0000e+00
   5.0000e+00
```

← Looks good!

$Bx = \begin{pmatrix} 5.00005 \\ 5 \end{pmatrix}$

But is it ?

The right answer is $\begin{pmatrix} 0 \\ 5 \end{pmatrix}$!

```
>> format long e
B * [.5;4.5]

ans =

   5.000050000000000e+00
   5.000000000000000e+00
```

when you check the answer, it looks great — but it is very far from the correct answer!

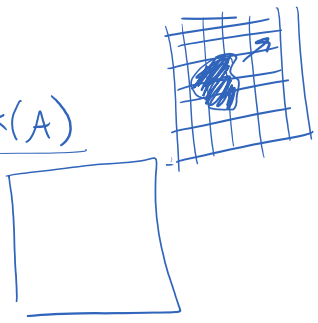
This is actually **the same** problem we already considered. Even though the error in b is small, the corresponding error in x is huge. This is only a good check for well-conditioned systems.

Moral: With an ill-conditioned system of linear equations
- use high-precision arithmetic
- find another way to check the answer?
- be careful! possibly try to reformulate the problem
  → use a **preconditioner**

Solve $P^{-1}Ax = P^{-1}b$, where P is such that $P^{-1}A$ is better conditioned, but P^{-1} can be applied efficiently. This is an important, but more advanced, topic, that we might get to later.

$A\vec{x} = \vec{b}$ $\quad PAx = Pb$

hopefully $\kappa(PA) \ll \kappa(A)$



$$\begin{pmatrix} \frac{1}{1000} & 0 \\ 0 & \frac{1}{1} \end{pmatrix} \begin{pmatrix} 1000 & 10 \\ 5 & 1 \end{pmatrix}$$

(3) Errors in the matrix A

Comparing

$$Ax = b \quad \text{and} \quad (A + \delta A)(x + \delta x) = b$$

(δA: some error in the matrix)

$$A \cdot \delta x = -\delta A (x + \delta x)$$

$$\delta x = -A^{-1} \cdot \delta A \cdot (x + \delta x)$$

$$\frac{\|\delta x\|}{\|x + \delta x\|} \le \underbrace{\|A^{-1}\| \|A\|}_{\kappa} \cdot \frac{\|\delta A\|}{\|A\|}$$

$$\frac{\|\delta x\|}{\|x + \delta x\|} \le \underbrace{\|A^{-1}\| \cdot \|A\|}_{\text{condition \# of } A} \cdot \frac{\|\delta A\|}{\|A\|}$$

relative error in the solution | condition # of A | relative error in A

Example:

```
octave:11> Aerror = 10^(-5) * randn(2,2);
octave:12> (A\b) - ((A+Aerror)\b)
ans =

  -1.1004e-04
   7.6036e-05

octave:13> (B\b) - ((B+Aerror)\b)
ans =

  -210.22
   210.26
```

Moral: In all three cases, the (relative) effect of the error is bounded by the condition number

$= \|A\| \cdot \|A^{-1}\| =$ largest singular value / smallest singular value

$\|\cdot\|_2$ $\|\cdot\|_2$ smallest singular value

Note: In practice, computing the condition number can be a lot of work.

It is enough usually to approximate it, or give upper bounds.

Certain classes of matrices reliably have large condition numbers; with experience you learn when to be careful.

Example: What is the condition number of

$$A = \begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & 1 & \ddots & \ddots & \\ & & & \ddots & -2 & 1 \\ 1 & & & & 1 & -2 \end{pmatrix}$$

? $A\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \vec{0}$

$\sigma_1 = \|A\| = 4$

$\sigma_n = 0$

A^{-1} doesn't exist

$\kappa(A) = \frac{\sigma_1}{\sigma_n} = \infty$

$$B = \begin{pmatrix} 4 & 0 \\ 0 & 0 \end{pmatrix} \quad \kappa = \infty$$

$\sigma_1 = 4$

$\sigma_2 = 0$

Example: What is the condition number of

$$C = \begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & 1 & \ddots & \ddots & \\ & & & \ddots & -2 & 1 \\ & & & & 1 & -2 \end{pmatrix}_{n \times n}$$

$N(C) = \{\vec{0}\}$

$C = -2I + U + U^T \Rightarrow \|C\| \le 2 + 2\|U\| = 4$

$U = \begin{pmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ 0 & & & & 0 \end{pmatrix}$

$\|C^{-1}\| = \frac{1}{\min_{x:\|x\|=1} \|Cx\|}$

$C = \sigma_1 \vec{v}_1 \vec{u}_1^T + \cdots + \sigma_n v_n u_n^T$

$\sigma_n = \min \|Cx\| = \sigma_n$

$x : \|x\| = 1$ for $x = v_n$

$$\left\| C \frac{\vec{x}}{\|\vec{x}\|} \right\| \left\| \frac{1}{\sqrt{n}} \begin{pmatrix} -1 \\ 0 \\ 0 \\ \vdots \\ -1 \end{pmatrix} \right\| = \sqrt{\frac{2}{n}} \Rightarrow \sigma_n \le \sqrt{\frac{2}{n}}$$

$$\Rightarrow \frac{1}{\sigma_n} \ge \sqrt{\frac{n}{2}} \Rightarrow \frac{\sigma_1}{\sigma_n} \ge \frac{1}{2}\sqrt{n}$$

Answer: Largest s.v. = norm ≈ 4 still.

Smallest singular value?

Observe:

$x = \frac{1}{\sqrt{n}}(1, 1, \dots, 1)$ has norm 1

$Ax = (-1/\sqrt{n}, 0, 0, 0, \dots, 0, -1/\sqrt{n})$

has norm $\sqrt{2/n}$

$\Rightarrow$ smallest singular value $\le \sqrt{\frac{2}{n}}$

$\Rightarrow$ condition # $\gtrsim \sqrt{n}$

So be careful!

In fact, the condition # is $\sim \frac{4}{\pi^2}(n+1)^2 = \Theta(n^2)$

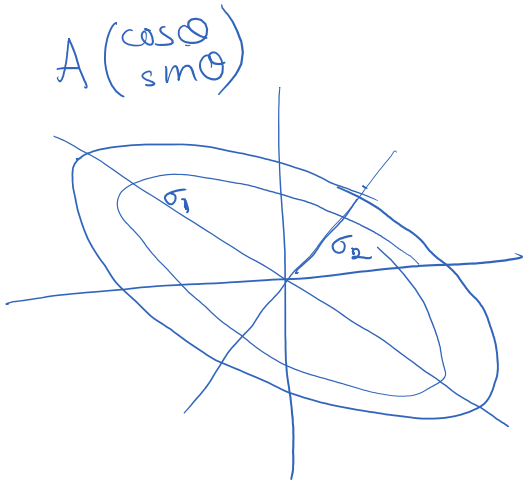
A better test vector is $x = \sqrt{\frac{2}{n+1}}\left(\sin\frac{\pi}{n+1}, \sin\frac{2\pi}{n+1}, \dots, \sin\frac{n\pi}{n+1}\right)$

ANOTHER REASON THE CONDITION NUMBER IS IMPORTANT:

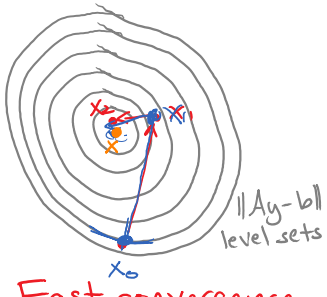
POORLY CONDITIONED SYSTEMS OF EQUATIONS TAKE LONGER TO SOLVE

Rough intuition: Iterative solvers for $Ax = b$ start with a guess, and try to improve it.

$x_0 \longrightarrow x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow \cdots$



Well-conditioned A



Fast convergence

Poorly conditioned A

Slow convergence

$\|Cx - b\|^2$

Theorem:

Conjugate-gradient method on a positive semi-definite matrix A, with condition number κ, takes

$$O\left(\sqrt{\kappa} \cdot \log\frac{1}{\varepsilon}\right)$$

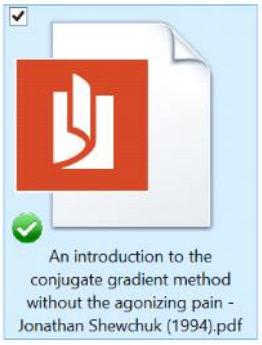
steps to get within ε of the true solution x.

(Or $O(\kappa \log\frac{1}{\varepsilon})$ steps for a general matrix A.)

or $O(\kappa \log\frac{1}{\varepsilon})$ for a general A

Each step uses a matrix-vector multiplication
$= O(\#\text{ of nonzero entries in } A)$ steps.

Time permitting, we'll discuss this later. Or, see →



Example: GRADIENT DESCENT

Goal: Solve $A\vec{x} = \vec{b}$.

Define the cost function

$$C(\vec{x}) = \|Ax - b\|^2$$
$$= \|A(x - x^*)\|^2 \quad \text{where } Ax^* = b$$
$$= (x - x^*)^T A^T A (x - x^*)$$

(A^TA: positive semi-definite)

If A has SVD

$$A = \sum_i \lambda_i u_i v_i^T$$
$$A^TA = \sum_i \lambda_i^2 v_i v_i^T$$

$$\Rightarrow C(\vec{x}) = \sum_i \lambda_i^2 |v_i \cdot (x - x^*)|^2$$

$\Rightarrow$ level set $\{\vec{x} \mid C(\vec{x}) = b\}$ is an ellipse!
(like $\lambda_1^2 x_1^2 + \lambda_2^2 x_2^2 + \cdots + \lambda_n^2 x_n^2 = b$)

Algorithm: Repeat until convergence:

$$x^t = x^{t-1} - \alpha\left(\frac{\partial C}{\partial x_1}(x^{t-1}), \ldots, \frac{\partial C}{\partial x_n}(x^{t-1})\right)$$

i.e., step in the direction opposite the largest increase in C.

$$\frac{\partial C}{\partial x_j} = \frac{\partial}{\partial x_j}\left[(Ax-b)^T(Ax-b)\right]$$
$$= \frac{\partial}{\partial x_j} \sum_i (Ax-b)_i^2$$
$$= 2\sum_i (Ax-b)_i \cdot \frac{\partial}{\partial x_j}\left[(Ax-b)_i\right]$$
$$= 2\sum_i (Ax-b)_i \cdot a_{ij}$$

$$\Rightarrow \left(\frac{\partial C}{\partial x_1}, \ldots, \frac{\partial C}{\partial x_n}\right) = 2\sum_i \underbrace{(Ax-b)_i}_{(Ax-b)^T e_i} \cdot \underbrace{(a_{i1}, \ldots, a_{in})}_{e_i^T A}$$
$$= 2(Ax-b)^T \left(\sum_i e_i e_i^T\right) A$$
$$= 2(Ax-b)^T A$$

(As you might guess!)

Example:

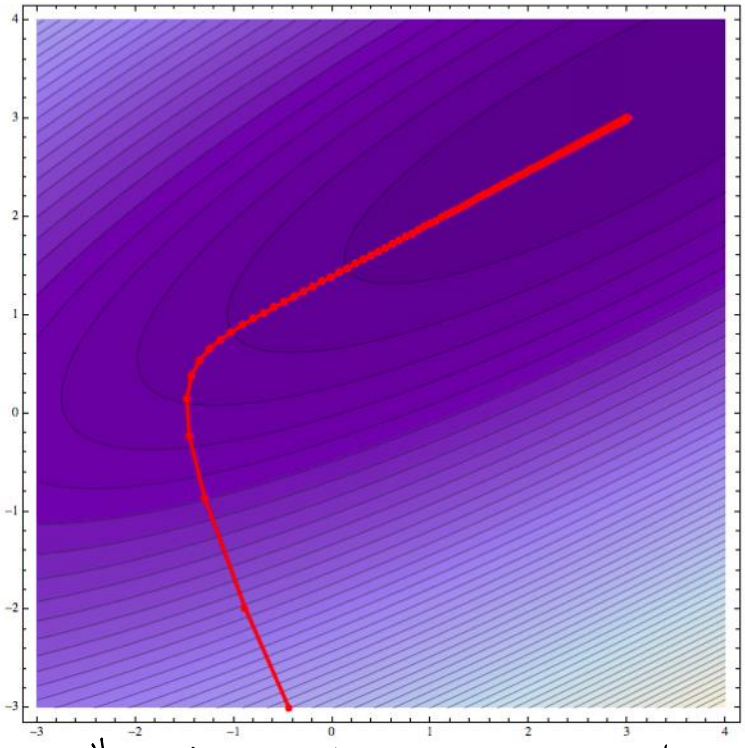
$$A = \begin{pmatrix} 2 & -1.5 \\ -1.5 & 4 \end{pmatrix};$$

Example:

```
A = {{2, -1.5}, {-1.5, 4}};
b = A.{3, 3};
Eigenvalues[A]

α = .01;
x = {0, -4};
list = {x};
For[k = 1, k ≤ 1000, k++,
  x = x - 2 α (A.x - b).A;
  AppendTo[list, x];
 ];
"The solution is:"
x
```

```
{4.80278, 1.19722}

The solution is:

{3., 3.}
```

```
"Display the results:";
iter = ListPlot[list, PlotRange → 3 {{-1, 1}, {-1, 1}}, Joined → True,
   PlotStyle → {Red, Thickness[.005]}, PlotMarkers → Automatic];
contour = ContourPlot[Norm[A.{x, y} - b] ^ 2, {x, -3, 4}, {y, -3, 4}, Contours → 50];
Show[contour, iter]
```



For small enough α, this has to converge, since there is a unique local and global minimum.

Observe: Smaller condition number
$\Rightarrow$ Rounder ellipse $\Rightarrow$ Faster convergence

Remark: Although not generally the fastest way of solving a set of linear equations, the gradient descent approach is very robust, and **many** variants are used in **many** applications.