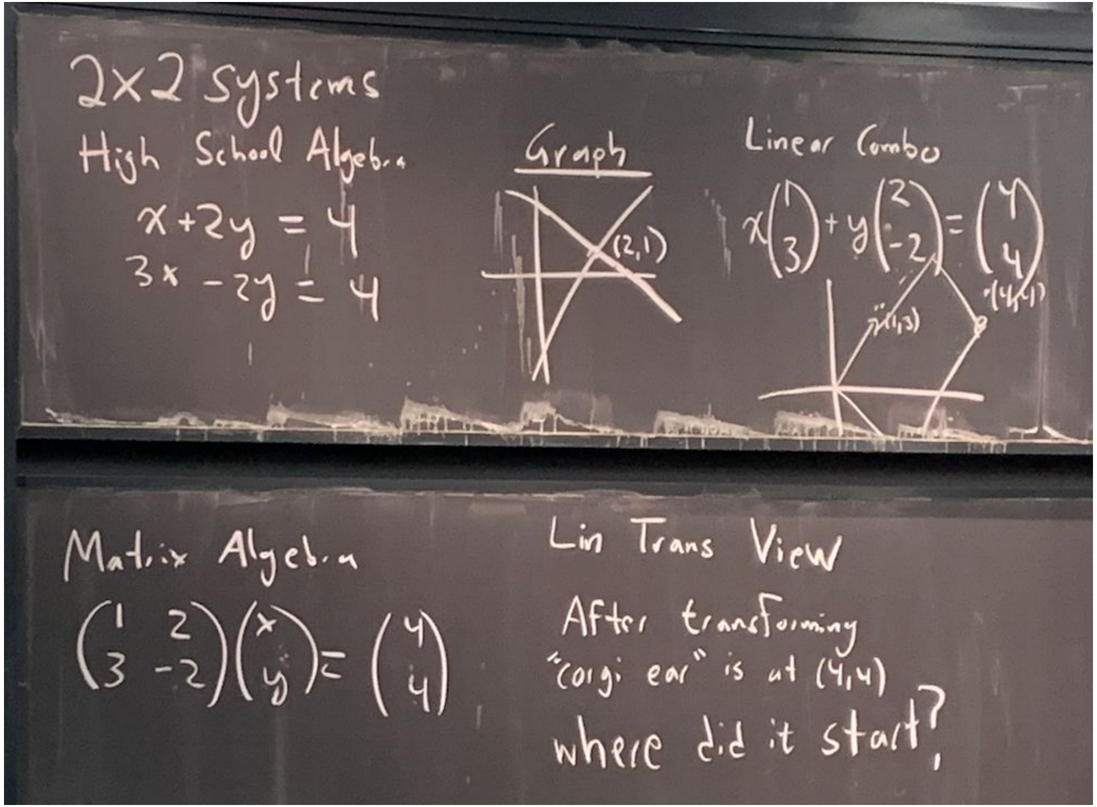
Lecture 3 18.06 (2020) Syllabus link
Solving Ax=b

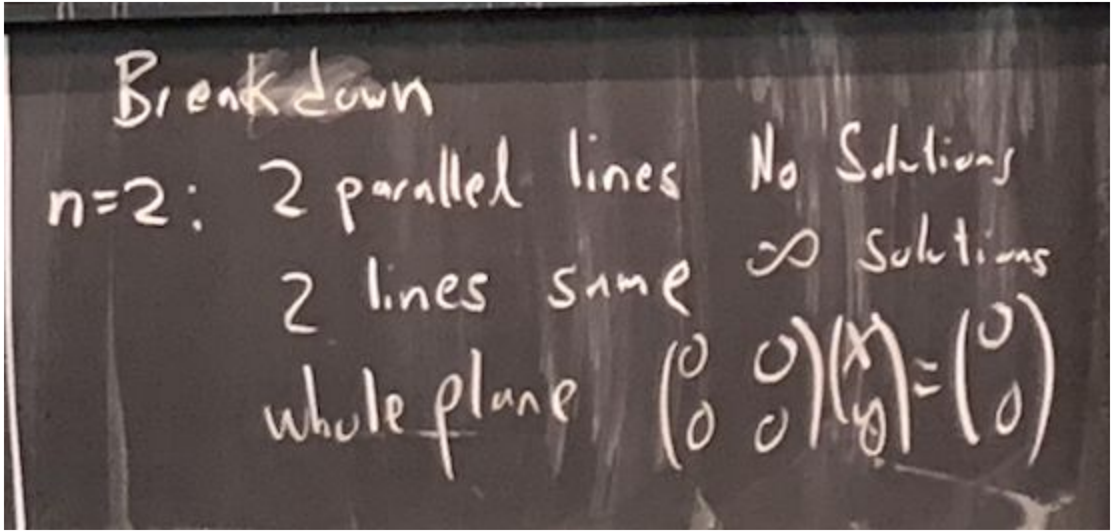
Outline

1. Many views on the same problem
2. Breakdown
3. Special Matrices, Complexity
   a. Diagonal
   b. Identity
   c. Upper Triangular
   d. General Matrices = Gaussian Elimination

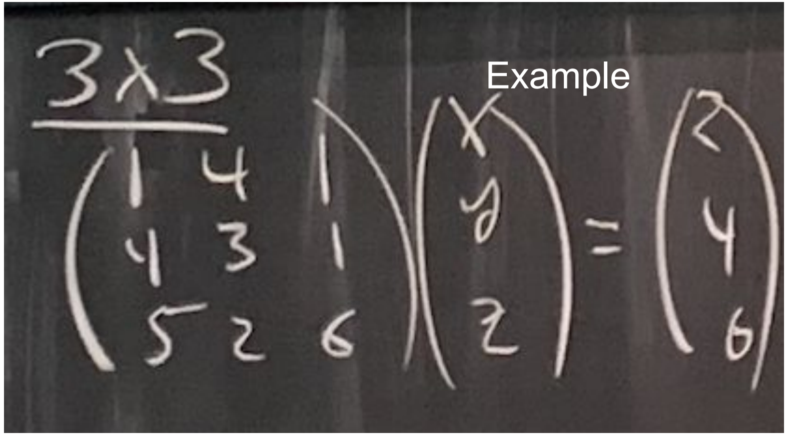
2x2 systems: High School view / Linear Transformation View



2x2 examples of breakdown

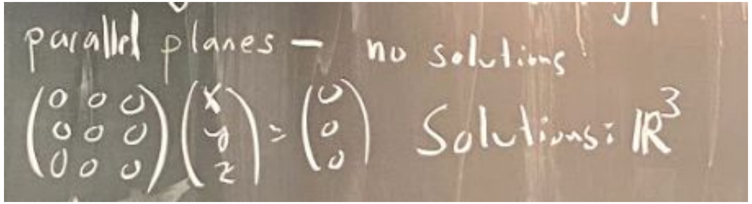


3x3 systems

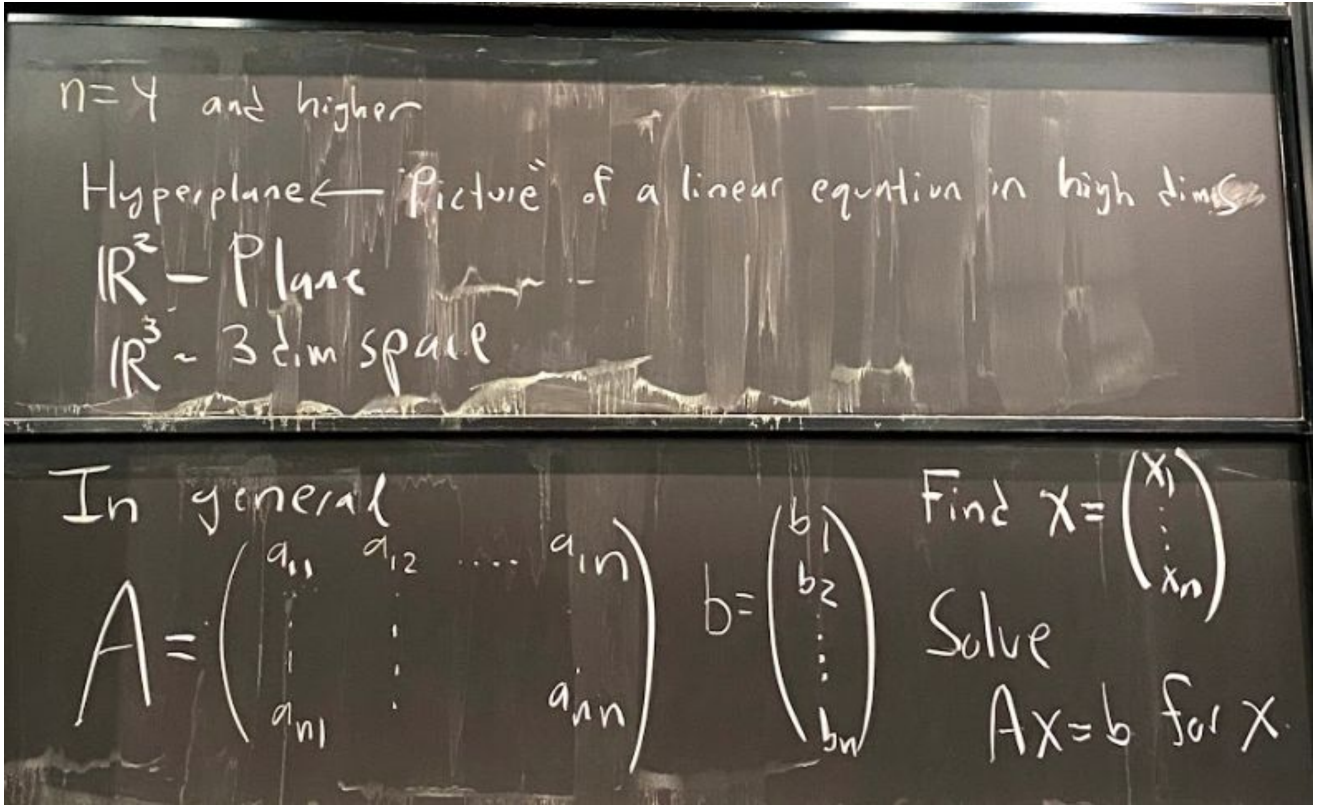


Generically the picture would be three planes intersecting at a point.

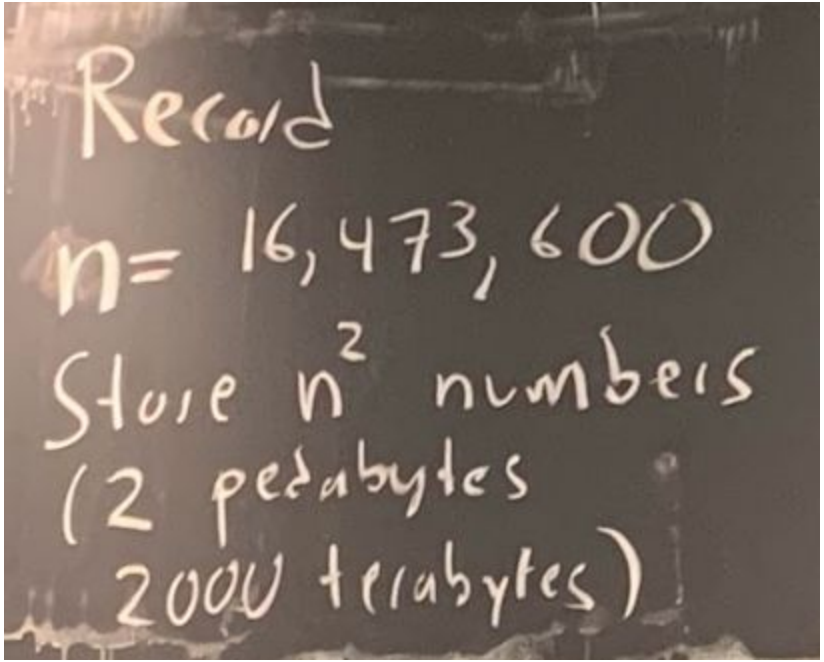
Some breakdown possibilities: no solutions or infinitely many (think about what else there could be)



n=4 and higher



The record in 2020 for (general) systems solving



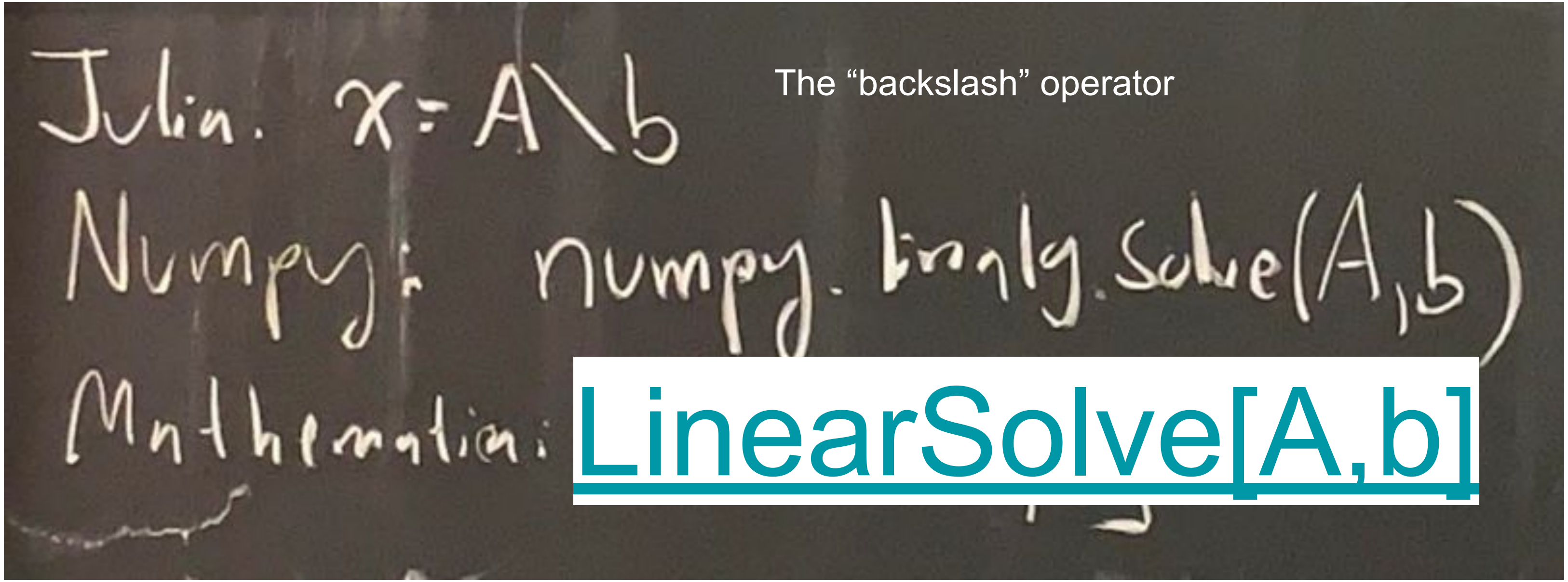
(Storing the matrix might require the memory
In thousands of your laptops)

Another view : find the linear combination

Find $x_1, \dots, x_n$

$$x_1 \begin{pmatrix} a_{11} \\ \vdots \\ a_{n1} \end{pmatrix} + x_2 \begin{pmatrix} a_{12} \\ \vdots \\ a_{n2} \end{pmatrix} + \dots + x_n \begin{pmatrix} a_{1n} \\ \vdots \\ a_{nn} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

Solving in a few packages (requires an A and a b)



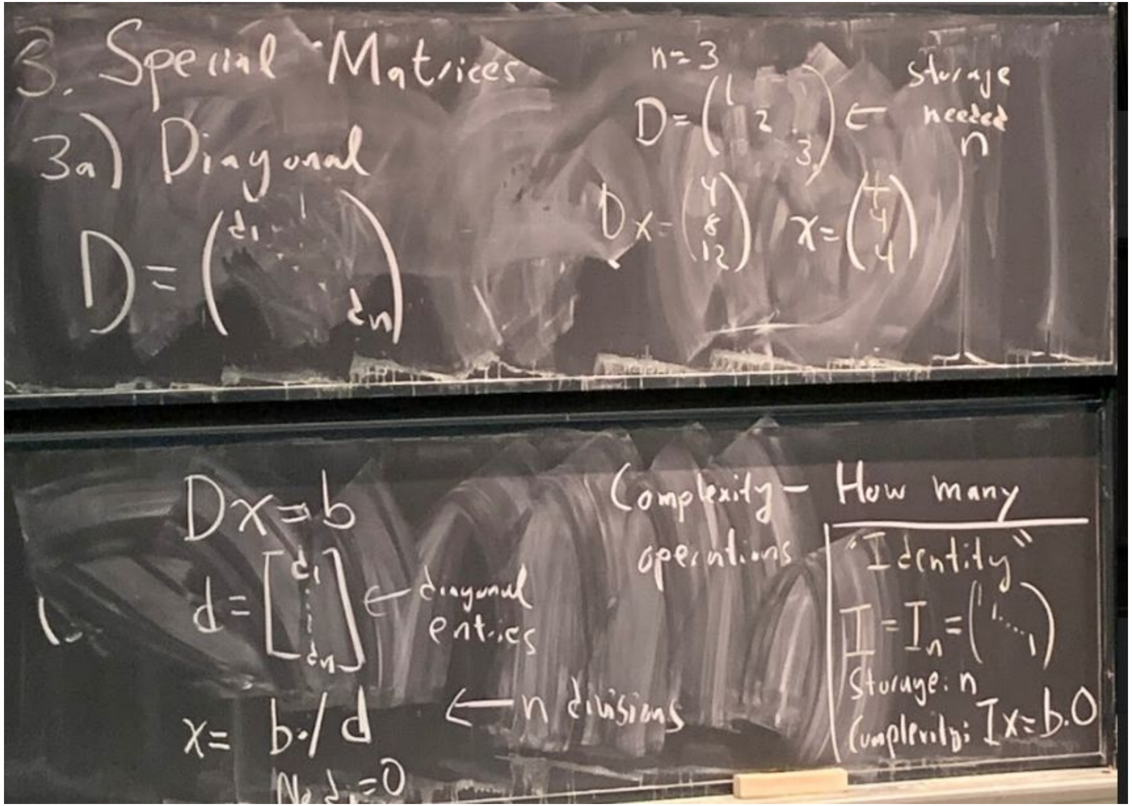
The “backslash” operator

[LinearSolve[A,b]]

Numpy: https://numpy.org/doc/1.18/reference/generated/numpy.linalg.solve.html

All packages use LAPACK under the hood for general systems. So solving is not truly different in any of these packages.

Some Special Matrices, Storage and Complexity



Diagonals need to store n numbers
And solving Ax=b requires only n
Divisions. We say Diagonals have n
parameters.

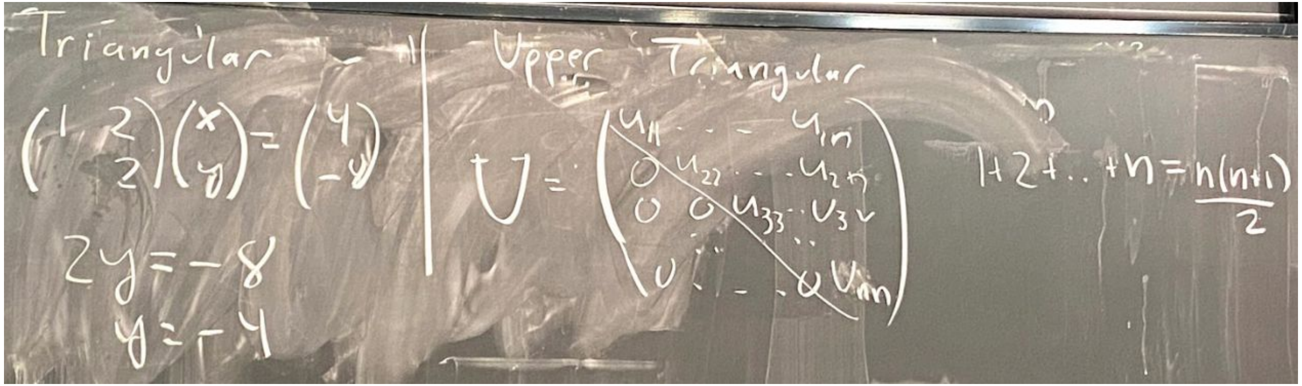
The identity has only one parameter
(the number “n”)
And requires no operations to solve
Ix=b.

Upper Triangular

Often denoted U for “upper”

Requires n(n+1)/2 storage locations (or parameters) which we approximate for n largish by $n^2/2$

Solution method called “back substitution” as we can work backwards with only one unknown at a time that we can solve for as long as the diagonal is not zero.



Back Substitution

Step 0: $u_{nn}x_n=b_n$ Solution: $x_n = b_n / u_{nn}$

Step 1: $u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n=b_{n-1}$ Solution: $x_{n-1} = (b_{n-1} - u_{n-1,n}x_n) / u_{n-1,n-1}$

...

Step k: Solve $\Sigma_{i=k,...,n}\, u_{ki}\, x_i = b_k$ for x_k given x_{k+1} through x_n are now known

The solution: $x_k = (b_k - \Sigma_{i=k+1,...,n}\, u_{ki}\, x_i) / u_{kk}$

...

Complexity of upper triangular solve

Step k has k “±”s, k multplies, and one divided

Approximate count is n^2 operations (work it out!)

Using $(1+2+...+n-1) = n(n-1)/2 \approx n^2/2$ for large n

What is a parameter?

A parameter is defined as a number that is needed to specify an object.

So we have seen diagonals have n parameters, upper triangular n(n+1)/2, and unit lower triangular n(n-1)/2.

A vector that sums to 0 only needs n-1 parameters because if you know n-1 you can deduce the last one.

The identity needs only the number n. So does the nxn zero matrix.

When do we need to include the “n” as a parameter?

When talking about parameters, normally we begin with some set in context.

- Example 1. The set of all vectors in R^n. Parameters, $v_1,...,v_n$. The n is “built” into the set.
- Example 2. The set of all vectors. Here n can be inferred from the parameters. On a computer if you type [1,5,6,9], the computer can infer you have a vector in R^4.
- Example 3: The set of all nxn upper triangular matrices. Again, “built-in”
- Example 4: the set of all upper triangular matrices. The n can be inferred.
- Example 5: The set of vectors orthogonal to [1,1,...,1]. If you specify n-1 numbers, then n can be inferred as one more than the list size.
- Example 6: The set of all identity matrices. n is required as a parameter.