Lecture 8

Finish Linear Algebra Review, Cost Analysis, Maybe Gaussian Elimination

Owen L. Lewis

Department of Mathematics and Statistics
University of New Mexico

Sept. 12, 2024

Notes

Singularity of A

If an $n \times n$ matrix, A, is **singular** then

- the columns of A are linearly dependent
- the rows of A are linearly dependent
- $\mathrm{rank}(A) < n$
- $\det(A) = 0$
- A^{-1} does not exist
- a solution to $Ax = b$ may not exist
- If a solution to $Ax = b$ exists, it is not unique

Know these conditions

Notes

Linear Independence

Two vectors lying along the same line are not independent

$$u = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \text{and} \qquad v = -2u = \begin{bmatrix} -2 \\ -2 \\ -2 \end{bmatrix}$$

Any two independent vectors, for example,

$$v = \begin{bmatrix} -2 \\ -2 \\ -2 \end{bmatrix} \qquad \text{and} \qquad w = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

define a plane. Any other vector in this plane of v and w can be represented by

$$x = \alpha v + \beta w$$

x is **linearly dependent** on v and w because it can be formed by a linear combination of v and w.

Linear Independence

A set of vectors is linearly independent if it is impossible to use a linear combination of vectors in the set to create another vector in the set. Linear independence is easy to see for vectors that are orthogonal, for example,

$$\begin{bmatrix} 4 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \begin{bmatrix} 0 \\ -3 \\ 0 \\ 0 \end{bmatrix}, \qquad \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

are linearly independent.

Linear Independence

Consider two linearly independent vectors, u and v.
If a third vector, w, *cannot* be expressed as a linear combination of u and v, then the set $\{u, v, w\}$ is linearly independent.

In other words, if $\{u, v, w\}$ is linearly independent then

$$\alpha u + \beta v = \delta w$$

can be true *only if* $\alpha = \beta = \delta = 0$.

More generally, if the only solution to

$$\alpha_1 v_{(1)} + \alpha_2 v_{(2)} + \cdots + \alpha_n v_{(n)} = 0 \tag{1}$$

is $\alpha_1 = \alpha_2 = \ldots = \alpha_n = 0$, then the set $\{v_{(1)}, v_{(2)}, \ldots, v_{(n)}\}$ is **linearly independent**. Conversely, if equation (1) is satisfied by at least one nonzero α_i, then the set of vectors is **linearly dependent**.

Linear Independence

Let the set of vectors $\{v_{(1)}, v_{(2)}, \ldots, v_{(n)}\}$ be organized as the columns of a matrix. Then the condition of linear independence is

$$\left[\begin{array}{c|c|c|c} & & & \\ v_{(1)} & v_{(2)} & \cdots & v_{(n)} \\ & & & \end{array}\right] \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2}$$

The columns of the $m \times n$ matrix, A, are linearly independent if and only if $x = (0, 0, \ldots, 0)^T$ is the only n element column vector that satisfies $Ax = 0$.

Summary of Requirements for Solution of $Ax = b$

Given the $n \times n$ matrix A and the $n \times 1$ vector, b

- the solution to $Ax = b$ exists and is unique for any b if and only if $\mathrm{rank}(A) = n$.

Recall: rank $=$ # of linearly independent rows or columns

Recall: $Range(A) =$ set of vectors y such that $Ax = y$ for some x

Solving a system

$$Ax = b$$

Three situations:

1. A is nonsingular: There exists a unique solution $x = A^{-1}b$
2. A is singular and $b \in Range(A)$: There are infinite solutions.
3. A is singular and $b \notin Range(A)$: There no solutions.

1. $A = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$ $b = \begin{bmatrix} 1 \\ 8 \end{bmatrix}$, then $x = \begin{bmatrix} 1/2 \\ 2 \end{bmatrix}$.
2. $A = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$ $b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, then infinitely many solutions. $x = \begin{bmatrix} 1/2 \\ \alpha \end{bmatrix}$.
3. $A = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$ $b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, then no solutions.

Big-O

How to measure the impact of n on algorithmic cost?

$\mathcal{O}(\cdot)$

Let $g(n)$ be a function of n. Then define

$$\mathcal{O}(g(n)) = \{f(n) \,|\, \exists c, n_0 > 0 \,:\, 0 \leqslant f(n) \leqslant cg(n),\ \forall n \geqslant n_0\}$$

That is, $f(n) \in \mathcal{O}(g(n))$ if there is a constant c such that $0 \leqslant f(n) \leqslant cg(n)$ is satisfied.

- assume non-negative functions (otherwise add $|\cdot|$) to the definitions
- $f(n) \in \mathcal{O}(g(n))$ represents an asymptotic upper bound on $f(n)$ up to a constant
- example: $f(n) = 3\sqrt{n} + 10^3 \log n + 8n + 0.004n^2 \in \mathcal{O}(n^2)$

BLAS

Basic Linear Algebra Subprograms (BLAS) interface introduced APIs for common linear algebra tasks

- Level 1: vector operations (dot products, vector norms, etc) e.g.

$$y \leftarrow \alpha x + y$$

- Level 2: matrix-vector operations, e.g.

$$y \leftarrow \alpha Ax + By$$

- Level 3: matrix-matrix operations, e.g.

$$C \leftarrow \alpha AB + \beta C$$

- optimized versions of the reference BLAS are used everyday: ATLAS, etc.

vec-vec, mat-vec, mat-mat

- inner product of u and v both $[n \times 1]$

$$\sigma = u^T v = u_1 v_1 + \cdots + u_n v_n$$

- $\rightarrow$ n multiplies, $n-1$ additions
- $\rightarrow$ $\mathcal{O}(n)$ flops

vec-vec, mat-vec, mat-mat

- mat-vec of A ($[n \times n]$) and u ($[n \times 1]$)

```
for i = 1, ..., n              %Loop over rows
  for j = 1, ..., n            %Loop over each column
    v(i) = a(i,j)u(j) + v(i)   %Multiply and add to this row
  end
end
```

- $\rightarrow n^2$ multiplies, n^2 additions
- $\rightarrow \mathcal{O}(n^2)$ flops

vec-vec, mat-vec, mat-mat

- mat-mat of A ($[n \times n]$) and B ($[n \times n]$)

```
for j = 1,...,n             %Loop over columns of output
  for i = 1,...,n           %Perform matrix-vector mult.
    for k = 1,...,n
      C(k,j) = A(k,i)B(i,j) + C(k,j)
    end
  end
end
```

- $\rightarrow n^3$ multiplies, n^3 additions
- $\rightarrow \mathcal{O}(n^3)$ flops

vec-vec, mat-vec, mat-mat

Operation	FLOPS
$u^T v$	$\mathcal{O}(n)$
Au	$\mathcal{O}(n^2)$
AB	$\mathcal{O}(n^3)$

FLOPS = “floating point operations”
(addition/subtraction/multiplication/division)

Notes

Remember how to spot $\mathcal{O}(n^{\alpha})$!!

Timing in Matlab: vec-vec

Listing 1: Matlab

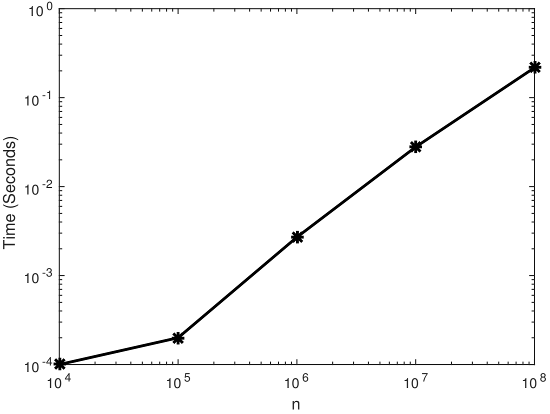
```
%Create two random vectors of size n
n = 50000;
u = rand(n,1);
v = rand(n,1);

%Measure time using the cputime command
t = cputime;

%Do the epxeriment 100 times
for j = 1 : 100
    %Inner Product
    ip = u'*v;
end

%average the times
timing = (cputime-t)/100;
```

Timing in Matlab: vec-vec



Let's dig into the timings a bit more with

Example

test_flops.m

vec-vec, mat-vec, mat-mat

- mat-vec of A ($[n \times n]$) and u ($[n \times 1]$)

```
for i = 1, ..., n
   for j = 1, ..., n
      v(i) = a(i,j)u(j) + v(i)
   end
end
```

- $\rightarrow n^2$ multiplies, n^2 additions
- $\rightarrow \mathcal{O}(n^2)$ flops

vec-vec, mat-vec, mat-mat

- mat-mat of A ($[n \times n]$) and B ($[n \times n]$)

```
for j = 1, ..., n
  for i = 1, ..., n
    for k = 1, ..., n
      C(k,j) = A(k,i)B(i,j) + C(k,j)
    end
  end
end
```

- $\rightarrow n^3$ multiplies, n^3 additions
- $\rightarrow \mathcal{O}(n^3)$ flops

vec-vec, mat-vec, mat-mat

Operation	FLOPS
$u^T v$	$\mathcal{O}(n)$
Au	$\mathcal{O}(n^2)$
AB	$\mathcal{O}(n^3)$

How you access memory can greatly affect the constant in front of the $\mathcal{O}(n^k)$. For instance, both row and column access patterns for a mat-vec behave like $\mathcal{O}(n^2)$, but with very different constants, depending on how A is stored.

Example

test_memory_patterns_matvec.m

Let's draw out the dot-product and column-wise view of a mat-vec.

Turning the Problem Around

The central problem is

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

So far: if we know A and $\vec{x}$, can we calculate $\vec{b}$.

Now: if we know A and $\vec{b}$, can we solve for $\vec{x}$?

Common Problems

Motivating example:

$$u''(x) = f(x), \quad u(0) = u(1) = 0.$$

Given an $f(x)$, can we find $u(x)$?

Common Problems

Motivating example:

$$u''(x) = f(x), \quad u(0) = u(1) = 0.$$

Given an $f(x)$, can we find $u(x)$?

Yes! (But we have to solve a matrix equation)

What's the big deal? ....cost

Look at 1D:

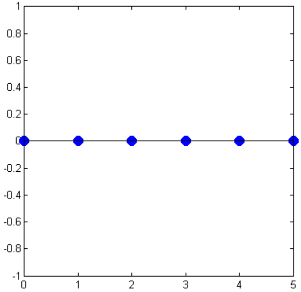
- 3 equations
- 3 unknowns
- each unknown coupled to its neighbor

$$\begin{array}{cccl} 2x_1 & -x_2 & & = 5.8 \\ -x_1 & 2x_2 & -x_3 & = 13.9 \\ & -x_2 & 2x_3 & = 0.03 \end{array}$$

or

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \begin{bmatrix} 5.8 \\ 13.9 \\ 0.03 \end{bmatrix}$$

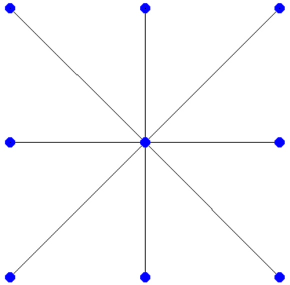
Easy in 1d

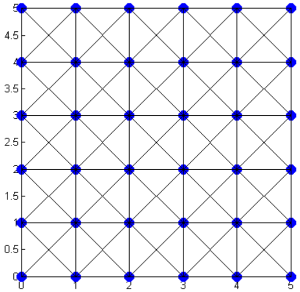


$$\begin{bmatrix} 2 & -1 & & & \\ & \ddots & & & \\ & -1 & 2 & -1 & \\ & & & \ddots & \\ & & & -1 & 2 \end{bmatrix}$$

- n points in the grid
- $3 * n - 2$ or about $3n$ nonzeros in the matrix
- tridiagonal (easy)

2D: harder

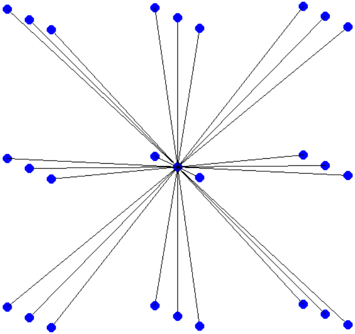


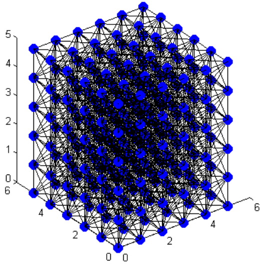


$$\begin{bmatrix} 8 & -1 & & -1 & & \\ & \ddots & & & & \\ -1 & -1 & 8 & -1 & -1 & \\ & & & \ddots & & \\ & -1 & & & -1 & 8 \end{bmatrix}$$

- n points in one direction
- n^2 points in grid
- about 9 nonzeros in each row
- about $9n^2$ nonzeros in the matrix
- n-banded (harder...we will see)

3D: hardest

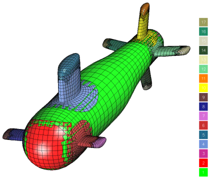




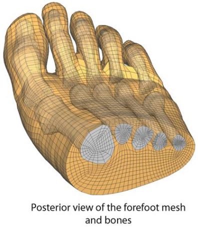
- n points in one direction
- n^3 points in grid

- about 27 nonzeros in each row
- about $27n^3$ nonzeros in the matrix
- n^2-banded (yikes!)

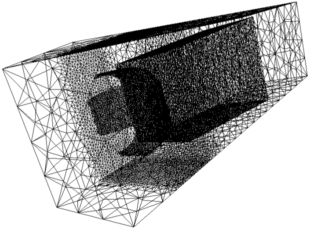
Applications get harder and harder...



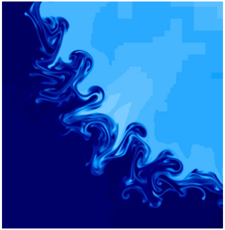
courtesy of LLNL



courtesy of TrueGrid



courtesy of Rice



courtesy of Warwick U.

Solving is a problem...

dim	unknowns	storage	example (n)		
1D	n	$3n$	10	100	1000
2D	n^2	$9n^2$	10^2	10^4	10^6
3D	n^3	$27n^3$	10^3	10^6	10^9

Gaussian Elimination (eventually)

- Solving Diagonal Systems
- Solving Triangular Systems
- Gaussian Elimination Without Pivoting
  - Hand Calculations
  - Cartoon Version
  - The Algorithm
- Gaussian Elimination with Pivoting
  - Row or Column Interchanges, or Both
  - Implementation
- Solving Systems with the Backslash Operator

Solving Diagonal Systems

The system defined by

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ 6 \\ -15 \end{bmatrix}$$

Solving Diagonal Systems

The system defined by

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ 6 \\ -15 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rrrcr} x_1 & & & = & -1 \\ & 3x_2 & & = & 6 \\ & & 5x_3 & = & -15 \end{array}$$

Solving Diagonal Systems

The system defined by

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 5 \end{bmatrix} \qquad b = \begin{bmatrix} -1 \\ 6 \\ -15 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rrrcr} x_1 & & & = & -1 \\ & 3x_2 & & = & 6 \\ & & 5x_3 & = & -15 \end{array}$$

The solution is

$$x_1 = -1 \qquad x_2 = \frac{6}{3} = 2 \qquad x_3 = \frac{-15}{5} = -3$$

Solving Diagonal Systems

Listing 2: Diagonal System Solution

```
given A, b
for i = 1 ... n
   x_i = b_i / a_{i,i}
end
```

In Matlab:

```
>> A = ...          % A is a diagonal matrix
>> b = ...
>> x = b./diag(A)
```

This is the *only* place where element-by-element division (`./`) has anything to do with solving linear systems of equations.

Example

Try this in Matlab using `A = diag(rand(5,1));`

Operations?

Try...

Sketch out an operation count to solve a diagonal system of equations...

Operations?

Try...

Sketch out an operation count to solve a diagonal system of equations...

cheap!

one division n times $\longrightarrow \mathcal{O}(n)$ FLOPS

This is the best we can *ever* do. Why?

Triangular Systems

The generic lower and upper triangular matrices are

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & & 0 \\ \vdots & & \ddots & \vdots \\ l_{n1} & & \cdots & l_{nn} \end{bmatrix}$$

and

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & & u_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & u_{nn} \end{bmatrix}$$

The triangular systems

$$Ly = b \qquad Ux = c$$

are easily solved by **forward substitution** and **backward substitution**, respectively

Solving Triangular Systems

$$A = \begin{bmatrix} 4 & 0 & 0 \\ -2 & 3 & 0 \\ 2 & 1 & -2 \end{bmatrix} \qquad b = \begin{bmatrix} 8 \\ -1 \\ 9 \end{bmatrix}$$

Solving Triangular Systems

$$A = \begin{bmatrix} 4 & 0 & 0 \\ -2 & 3 & 0 \\ 2 & 1 & -2 \end{bmatrix} \qquad b = \begin{bmatrix} 8 \\ -1 \\ 9 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} 4x_1 & & & & & = & 8 \\ -2x_1 & + & 3x_2 & & & = & -1 \\ 2x_1 & + & x_2 & + & -2x_3 & = & 9 \end{array}$$

Solving Triangular Systems

$$A = \begin{bmatrix} 4 & 0 & 0 \\ -2 & 3 & 0 \\ 2 & 1 & -2 \end{bmatrix} \qquad b = \begin{bmatrix} 8 \\ -1 \\ 9 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} 4x_1 & & & & & = & 8 \\ -2x_1 & + & 3x_2 & & & = & -1 \\ 2x_1 & + & x_2 & + & -2x_3 & = & 9 \end{array}$$

Solve in forward order (first equation is solved first, etc)

$$x_1 = \frac{8}{4} = 2 \qquad x_2 = \frac{1}{3}\left(-1 + 2x_1\right) = \frac{3}{3} = 1$$

$$x_3 = \frac{1}{-2}\left(9 - x_2 - 2x_1\right) = \frac{4}{-2} = -2$$

Notes

Solving Triangular Systems

Solving for $x_1, x_2, \ldots, x_n$ for a lower triangular system is called **forward substitution**.

```
given L (lower △), b
x_1 = b_1/ℓ_11
for i = 2...n
   s = b_i
   for j = 1...i-1
      s = s - ℓ_{i,j} x_j
   end
   x_i = s/ℓ_{i,i}
end
```

What about Upper Triangular?

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

What about Upper Triangular?

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} -2x_1 & + & x_2 & + & 2x_3 & = & 9 \\ & & 3x_2 & + & -2x_3 & = & -1 \\ & & & & 4x_3 & = & 8 \end{array}$$

What about Upper Triangular?

$$A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \qquad b = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

is equivalent to

$$\begin{array}{rcrcrcr} -2x_1 & + & x_2 & + & 2x_3 & = & 9 \\ & & 3x_2 & + & -2x_3 & = & -1 \\ & & & & 4x_3 & = & 8 \end{array}$$

Solve in backwards order (last equation is solved first, etc)

$$x_3 = \frac{8}{4} = 2 \qquad \begin{aligned} x_2 &= \frac{1}{3}\left(-1 + 2x_3\right) = \frac{3}{3} = 1 \\ x_1 &= \frac{1}{-2}\left(9 - x_2 - 2x_3\right) = \frac{4}{-2} = -2 \end{aligned}$$