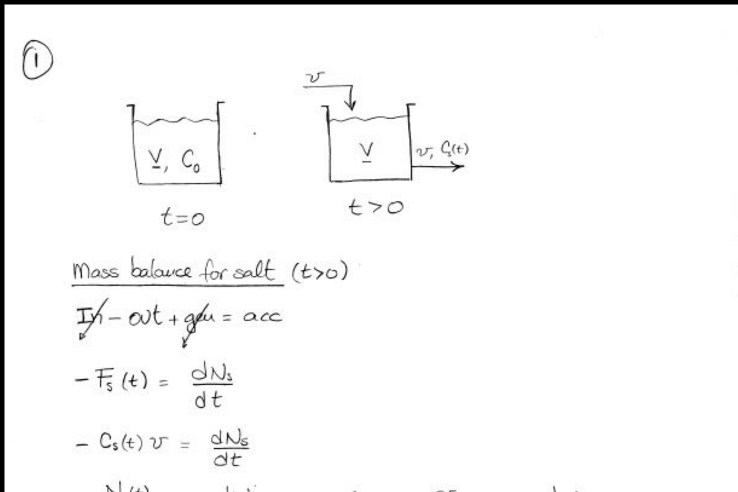
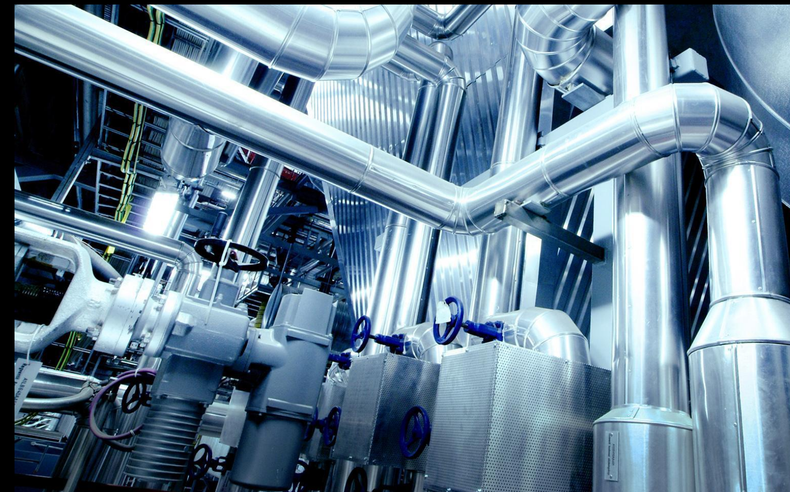




ChE352

Numerical Techniques for Chemical Engineers

Professor Stevenson

Lecture 14

Linear algebra is easy in numpy

```
b = np.array([1, 1, 1])  # Define vector b
A = np.array([[4, -1, 1], [-1, 4.25, 2.75],
[1, 2.75, 3.5]])  # Define matrix A
x = np.linalg.solve(A, b)  # Find Ax = b
C = A + B  # Find A plus B
D = A @ B  # Find A times B (matrix multiply)
E = A * B  # Find A times B COMPONENT-WISE
Bsq = B**2  # Square each element of B (≠B@B)
L = np.linalg.cholesky(A)  # Cholesky factor
Ainv = np.linalg.inv(A)  # Inverse (unwise)
Apnv = np.linalg.pinv(A)  # pseudo-inverse
```

Linear algebra is easy(?) in numpy

```
# by default, a 1-D np.array is "flat"
a = np.array([1, 2, 3])  # not row OR col
a2 = np.array([[1, 2, 3]])  # row vector
b = a.reshape(-1, 1)  # column vector
c = a.reshape(1, -1)  # row vector
d = b.flatten()  # flat version again
# shape affects matrix operations:
one_number = c @ b  # [[14]]
nine_numbers = b @ c  # 3x3 matrix
```

Writing out a large matrix in numpy

```
A = np.array([
    [9,   1.5, 0,   2.5, 0.5],
    [1.5, 10,  1.5, 0.5, 2  ],
    [0,   1.5, 11,  0,   2  ],
    [2.5, 0.5, 0,   8,   1  ],
    [0.5, 2,   2,   1,   5  ],
 ])
# Aligning the columns makes it
# easier to spot errors
```

How do you confirm your data?

```
import hashlib
# Get the unique hash of the data
sha = hashlib.sha256()
sha.update(A.dumps())  # dump string
print(sha.hexdigest()[:8])
# then you can compare your hash to
# someone else's to see immediately
# if you have the same data
```

What does the [:8] do? Why does it help?

How do you confirm an answer?

```
# example system of linear equations
b = np.array([1.4, 1.7, 2.0])
A = np.array([[ 4, -1,    1   ],
              [-1,  4.25, 2.75],
              [ 1,  2.75, 3.5]])
# first we'll solve, then test the solution
x = np.linalg.solve(A, b)  # Ax = b: find x
# check whether x is close (default=1e-8)
Ax = A @ x  # matrix multiply A times x
print(np.allclose(Ax, b))  # test Ax = b
```

Matrices are also linear operators

- Any linear operation over vectors of length N can be represented as an NxN matrix
  - *Linear operator* means: maps vector x to vector y such that all entries of y are weighted sums of entries of x
  - Example: *rotate* a vector
- This means we can treat an **operator** / transformation (a function from one vector to another) as **data** (NxN grid of numbers)

Rotation matrices

- Any rotation or scaling of a vector can be represented with a special matrix
- A rotation matrix is one that changes the *direction* of a vector but not its *magnitude*

Simple 2D rotation matrix examples

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

for 90°, 180°, and 270° counter-clockwise rotations.

- A 3D rotation can be represented with Euler angles ($\mathbf{R}^3$) or quaternions ($\mathbf{R}^4$), but a rotation matrix is easiest: just multiply

Matrix vocabulary

You should know what all of these things are:

- Scalar (dot) product, Matrix product
- Square, Diagonal, Identity matrices
- Upper and Lower Triangular matrices
- Transpose, Symmetric

A^{-1}, A^T, $|A|$, M_{ij}

You should know what all of these things are:

- Matrix inverse, existence of inverse (When?)
- Singular/noninvertible v. nonsingular/invertible
- Determinant
- Eigenvector / eigenvalue

Nonsingular Matrices are Nice

The following statements are equivalent for A in R^{nxn}:

iff, $\leftrightarrow$

What if it isn't square?

1. rank(A) = n
2. A is nonsingular
3. A^{-1} exists
4. The rows and columns of A are lin. indep.
5. det(A) ≠ 0
6. range(A) = R^n
7. nullspace(A) = {0}
8. Ax = b has a unique solution x^* for each b in R^n
9. The only solution to Ax = 0 is $x^* = 0$
10. Zero is not an eigenvalue of A

How could a matrix be ***close*** to singular?

Factoring Matrices

- Solving a linear system Ax = b requires $O(n^3)$ operations with Gaussian Elim. for A in R^{nxn}
- If n is large (>1000) **or** if we need $x = A^{-1}b$ for many different choices of b, this is expensive
- <u>If we can find triangular matrices L & U such that A = LU</u>, then we can find x differently:

$$Ax = b \quad \rightarrow \quad LUx = b, \quad let\ y = Ux \quad \rightarrow \quad Ly = b$$

- If U and L are <u>triangular</u>, solving Ly = b for y and Ux = y for x takes only $O(n^2)$ operations
- If n=1000, how much faster is this than $O(n^3)$?

LU solution for $Ax = b$

1. Let $A = LU \rightarrow LUx = b$
2. Let y be a new n-vector: $y = Ux \rightarrow Ly = b$
3. Since L is lower triangular, the first equation in $Ly = b$ says that $y_1 = b_1 / L_{11} \rightarrow$ 1 flop for y_1
4. y_1 is now known and the second equation involves only y_1 and $y_2 \rightarrow$ Calculate y_2
5. Repeat until y_n $\boxed{\text{This takes } 2n^2 \text{ flops total}}$
6. y is now known; repeat the same process for $Ux = y$, starting now with x_n and going up to x_1 since U is upper triangular.

Activity: Ly = b
5 min to do, 5 min discuss

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 4 & 0 \\ 5 & 2 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 4 \\ 6 \end{bmatrix}$$

Solve for y if Ly = b. It should only take $O(n^2) \approx 9$ flops.

Is this faster than Gauss. Elimination?

Answer: Ly = b

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 4 & 0 \\ 5 & 2 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 4 \\ 6 \end{bmatrix}$$

$y_1 = 0$
$y_2 = 1$
$y_3 = 4/3$

NOTE: getting LU form takes $O(n^3)$ flops

Special Matrices

- Sparse matrices – Have many more zeros than non-zeros, can save computation (What would this mean in a physical system?)
- Symmetric matrix: $A = A^T$ (What shape is A?)
- Positive definite matrices: $x^T P x > 0$ for any non-zero vector x (square, symmetric)
- Negative definite matrices: $x^T N x < 0$ for any non-zero vector x (square, symmetric)
- Positive semidefinite: $x^T S x \geq 0$ for any x
- Negative semidefinite: $x^T D x \leq 0$ for any x

Properties of PD Matrices

If a matrix P in R^{nxn} is PD, it implies the following:

1. P is non-singular
2. All diagonal entries of P are positive
3. -P is negative definite
4. Solving Px = b has stable growth of error
5. All leading principle minors (1x1, 2x2, . . . nxn) must be positive (Sylvester's criterion)
6. $P = U^TU = LL^T$ for some upper triangular U with positive diagonal entries. We call this the Cholesky Decomposition: $L = U^T$, $U = L^T$

Vector Norms = *Kinds of Length*

$$\forall x \in \mathbb{R}^n, \quad \exists \|\cdot\|_p : \mathbb{R}^n \rightarrow \mathbb{R} \quad \textit{such that:}$$

$$\|x\|_p = \begin{cases} \sqrt[p]{|x_1|^p + |x_2|^p + \ldots + |x_{n-1}|^p + |x_n|^p} & 1 \le p < \infty \\ \max\left(|x_1|, |x_2|, \ldots, |x_{n-1}|, |x_n|\right) & p = \infty \end{cases}$$

$$e.g. \quad \|x\|_2 = \sqrt{x^T x} \quad \left(always > 0 \quad if \quad x \neq 0\right)$$

Every **p** corresponds to a kind of vector length

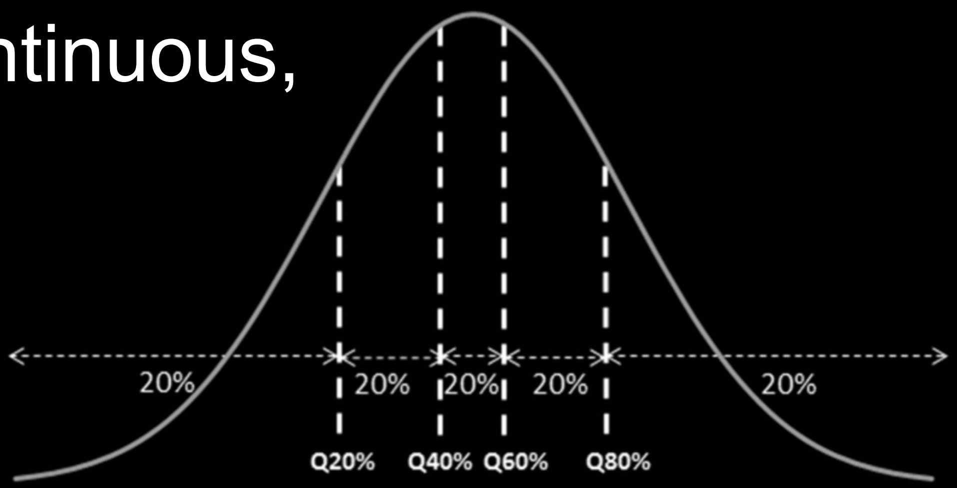
Inner product: (how is it like the 2-norm?)

$$\langle x, y \rangle \equiv x^T y = y^T x \quad \forall x, y \in \mathbb{R}^n$$

Example: Probability Vectors

- For any event, the set of probabilities of all outcomes form a "probability vector" with a 1-norm (Manhattan norm) of exactly 1
- Some matrices can act on probability vectors and give new probability vectors
- For non-finite outcomes, the 1-norm sum ∑ is continuous, aka an integral ∫

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} p \\ 1-p \end{pmatrix} = \begin{pmatrix} 1-p \\ p \end{pmatrix}$$



Vector Spaces

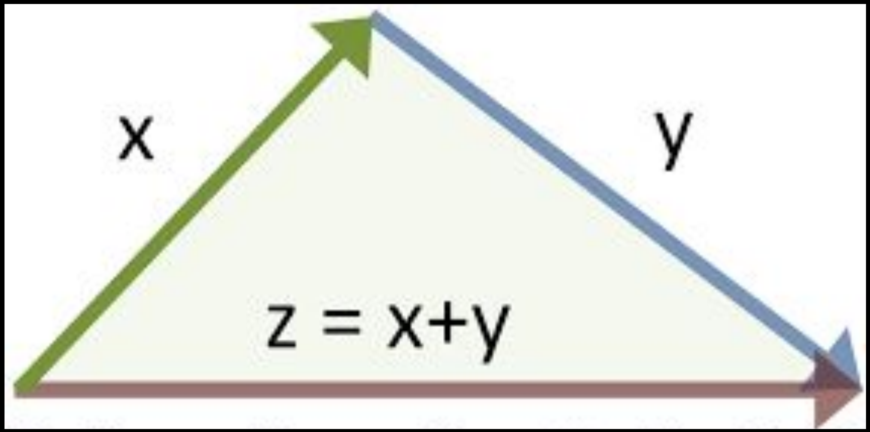
The following are true for any norm and any elements x and y of a <u>normed vector space</u>:

Norms are a measure of distance between two points/vectors

1. $\|x\| \geq 0$
2. $\|x\| = 0 \quad \leftrightarrow \quad x = 0$
3. $\|\alpha x\| = |\alpha| \|x\| \quad \forall \alpha \in \mathbb{R}$
4. $\|x + y\| \leq \|x\| + \|y\| \quad (\text{triangle inequality})$
5. $|\langle x, y \rangle| \leq \|x\| \|y\| \quad (\text{Cauchy-Schwarz inequality})$

What vector spaces do we use in this class?

Vector norm inequalities

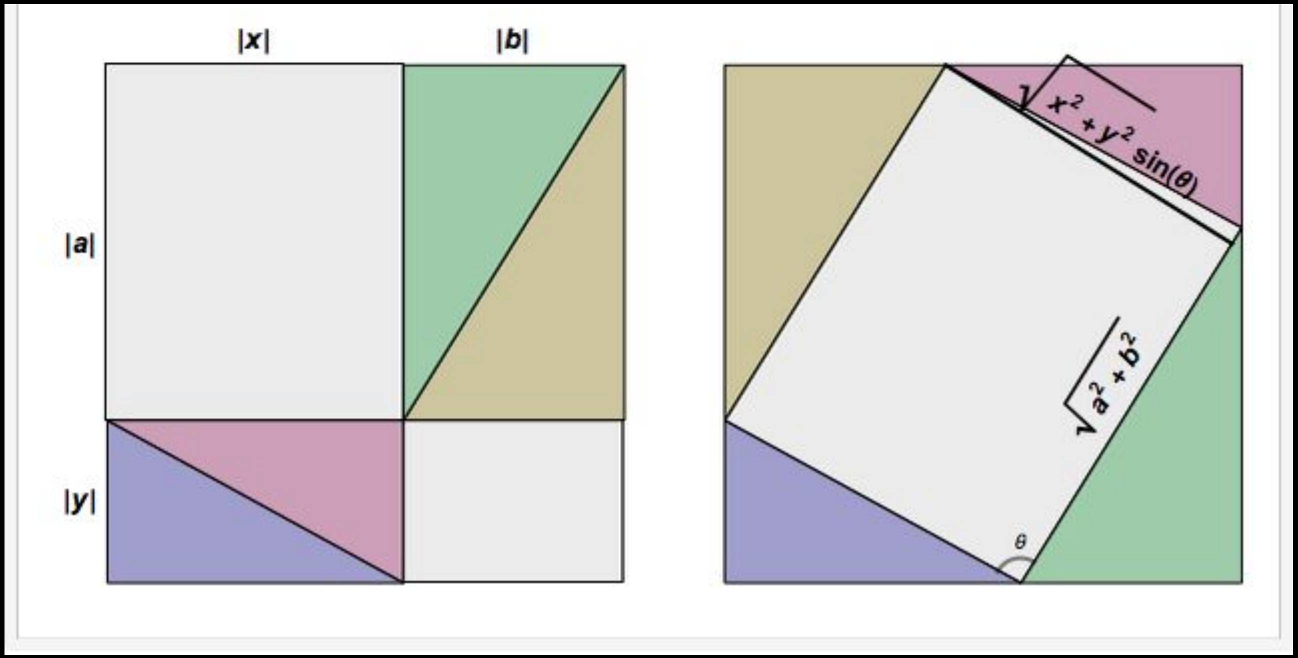


Triangle inequality

$|x + y| \leq |x| + |y|$

Cauchy-Schwarz inequality

$|x \bullet y| \leq |x| \bullet |y|$



Activity: Vector Norms
5 min to do, 5 min discuss

Find the 1-norm, 2-norm (Euclidean norm), and the infinity-norm of the following vectors:

$$x = \begin{bmatrix} 1 \\ 4 \\ 9 \\ 1 \\ 15 \end{bmatrix}, \quad y = \begin{bmatrix} 2 \\ 0 \\ -7 \\ 0 \\ 0 \end{bmatrix}$$

Then show that the triangle & Cauchy-Schwarz inequalities hold for the 2-norm.

Cauchy-Schwarz	Triangle
$\lvert x \cdot y \rvert \le \lvert x \rvert \cdot \lvert y \rvert$	$\lvert x + y \rvert \le \lvert x \rvert + \lvert y \rvert$

Answer: Vector Norms

$$\|x\|_1 = |1|+|4|+|9|+|1|+|15| = 30,$$

$$\|y\|_1 = |2|+|0|+|-7|+|0|+|0| = 9$$

$$\|x\|_2 = \sqrt{|1|^2+|4|^2+|9|^2+|1|^2+|15|^2} = 18,$$

$$\|y\|_2 = \sqrt{|2|^2+|0|^2+|-7|^2+|0|^2+|0|^2} = \sqrt{53}$$

$$\|x\|_\infty = \max_{i=1\ldots n} = |x_i| = 15, \quad \|y\|_\infty = \max_{i=1\ldots n} = |y_i| = 7$$

$$\|x+y\|_2 = \left\|\begin{bmatrix}3 & 4 & 2 & 1 & 15\end{bmatrix}^T\right\|_2$$

$$= \sqrt{255} \approx 16 \le 18 + \sqrt{53} = \|x\|_2 + \|y\|_2 \approx 25$$

$$|\langle x, y\rangle| = |2-63| = 61 \le 18\sqrt{53} = \|x\|_2\,\|y\|_2 \approx 131$$

Which norm is "the" norm?

How is the ∞-norm related to infinity?

Eigenvalues and Eigenvectors

Heard of these before?

What is the characteristic polynomial?

What can we do with eigen stuff?

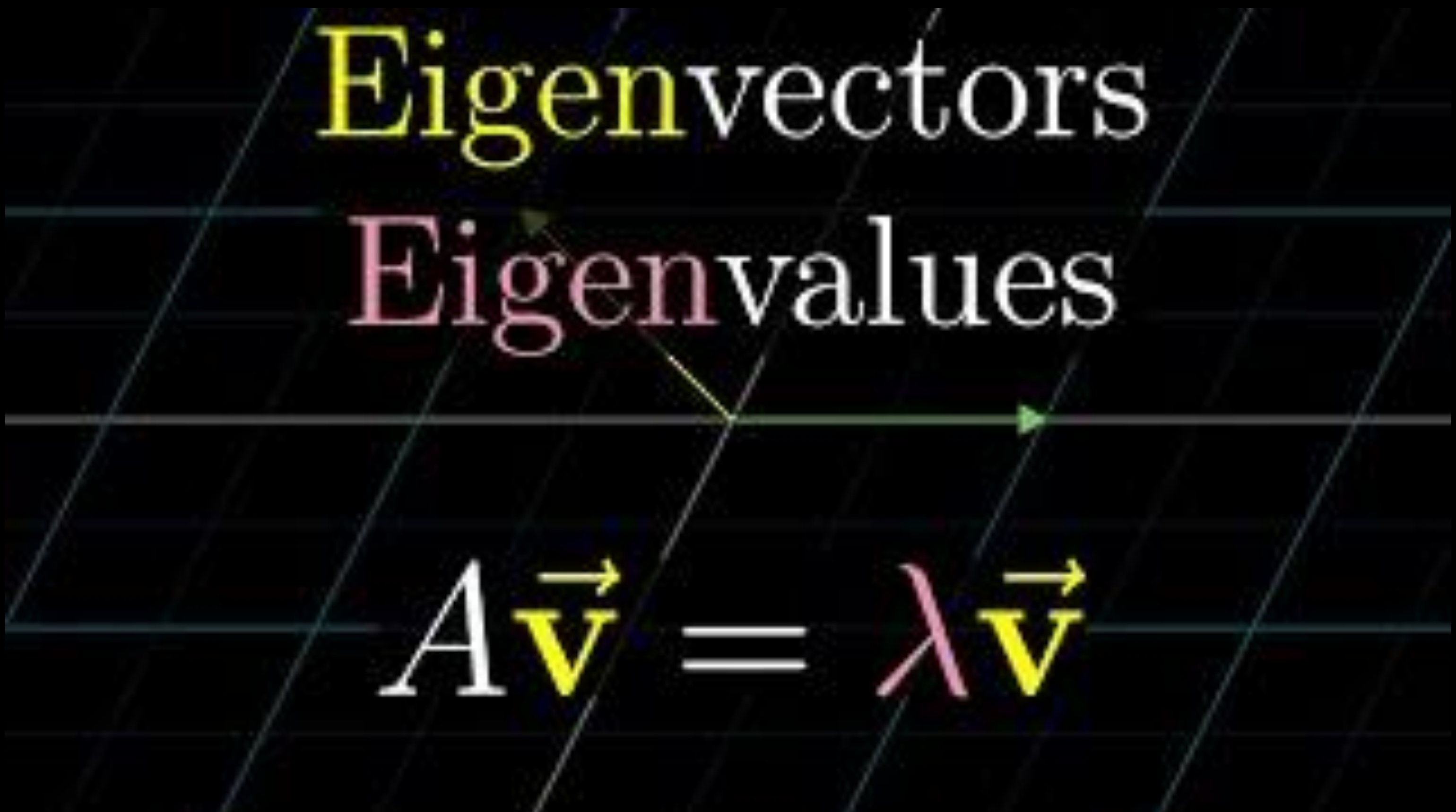
$$Ax = \lambda x$$

$$p(\lambda) = \det(A - \lambda I)$$

"Eigen" means "characteristic", or "own" as used in the phrase "my own room"

Eigenvalues and Eigenvectors

Eigenvalues & eigenvectors make the most sense when you think of matrices as *operators*, aka linear transformations, not just grids of numbers (though both ideas are true)



How do we find λ_i numerically?

$$A = \begin{bmatrix} 5 & -1 & 3 \\ 2 & 8 & 0 \\ 3 & -1 & 11 \end{bmatrix}$$

We can find the eigenvalues by solving for λ in $\det(A - \lambda I) = 0$ (the "characteristic polynomial"). This is roughly $O(N^3)$, like matrix multiplication.

We can find the eigenvectors by substituting each eigenvalue λ_i into the definition $Av_i = \lambda_i v_i$

Finding eigenvalues

"Characteristic polynomial"

$$p(\lambda) = \det(A - \lambda I) = \det \begin{bmatrix} 5-\lambda & -1 & 3 \\ 2 & 8-\lambda & 0 \\ 3 & -1 & 11-\lambda \end{bmatrix}$$

$-\lambda I$ means subtract λ from the diagonal

(Why?)

$$= (5-\lambda)(8-\lambda)(11-\lambda) - 0 - 6 - 0 + 2(11-\lambda) - 3^2(8-\lambda)$$

$$= (40 - 13\lambda + \lambda^2)(11-\lambda) + 7\lambda - 56$$

$$= -\lambda^3 + 24\lambda^2 - 176\lambda + 384 = 0 = (\lambda-4)(\lambda-8)(\lambda-12) \rightarrow \boxed{\begin{matrix} \lambda_1 = 4 \\ \lambda_2 = 8 \\ \lambda_3 = 12 \end{matrix}}$$

Polynomial is cubic in λ, because A is a rank-3 matrix

Cubic polynomial, so 3 roots

Finding eigenvectors

Eigenvector v^1

$$\lambda_1 = 4 \quad \rightarrow \quad Av^1 = \lambda_1 v^1 = 4v^1 \quad \rightarrow \quad Av^1 - 4v^1 = 0 \quad \rightarrow$$

$$(A - 4I)v^1 = 0 \quad \left(Bx = 0 \; means \; rank\left(B\right) < n\right) \quad \rightarrow$$

$$\begin{bmatrix} 1 & -1 & 3 \\ 2 & 4 & 0 \\ 3 & -1 & 7 \end{bmatrix} \begin{bmatrix} v_1^1 \\ v_2^1 \\ v_3^1 \end{bmatrix} \rightarrow \begin{matrix} v_1^1 - v_2^1 + 3v_3^1 = 0 \\ 2v_1^1 + 4v_2^1 = 0 \\ 3v_1^1 - v_2^1 + 7v_3^1 = 0 \end{matrix} \rightarrow \begin{matrix} 3v_3^1 = v_2^1 - v_1^1 \\ v_1^1 = -2v_2^1 \\ 3v_1^1 + 7v_3^1 = v_2^1 \end{matrix}$$

A system of linear equations Ax = b, where b is all zeros

$$\rightarrow \boxed{v^1 = \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix} \quad OR \quad v^1 = \begin{bmatrix} \sqrt{6}/3 \\ -\sqrt{6}/6 \\ -\sqrt{6}/6 \end{bmatrix}}$$

Vector quantum mechanics

- Probability vectors have 1-norm = 1.0
- What if we used **2-norm** = 1.0 instead?
- Generalized probability vectors = quantum state vectors
- Matrices which keep the 2-norm are called *Hermitian* matrices (in QM, *operators*)

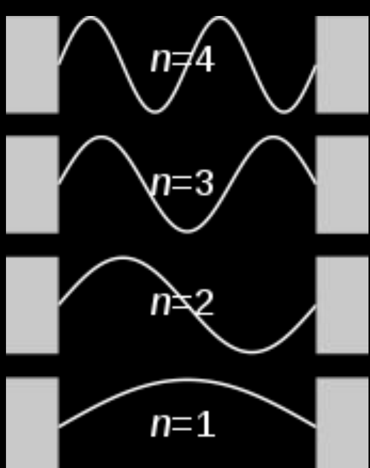
$$\boxed{A \text{ Hermitian} \quad \Longleftrightarrow \quad A = \overline{A^{\mathsf{T}}}}$$

A matrix which maintains the 2-norm is its own conjugate transpose

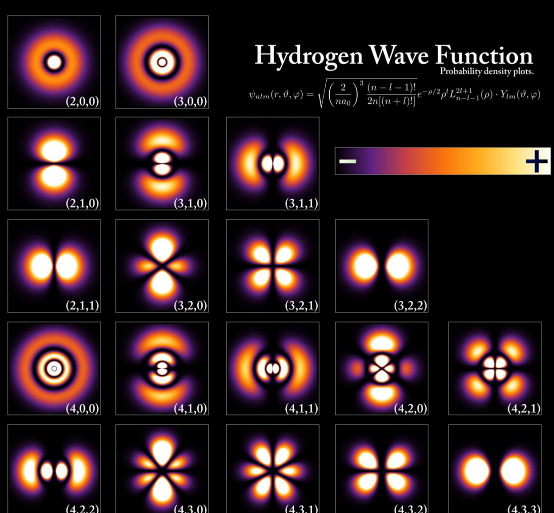
- Energy is just an eigenvalue: HΨ = EΨ
- ~100% of quantum chemistry is with vectors

How do we build a quantum state?

- For discrete properties, like spin ↑↓, it's easy: vector contains all discrete possibilities
  - For one particle, spin state vector = [a, b] where a & b are amplitudes of |↑〉& |↓〉
- For continuous properties, like position, we can pick a set of functions that approximate it



The state vector consists of an amplitude for each basis function



Eigenvalues in numpy

```
from numpy.linalg import (det, diag, eig,
                          matrix_rank, norm)
d = det(A)  # Determinant of A
r = matrix_rank(A)  # Rank of A
# if A is Hermitian, can use faster eigh(A)
eigenvalues, eigenvectors = eig(A)
# norm works for p = 1, 2, np.inf
x = norm(A, p)  # Matrix norm
```

Next time: Optimization

"World domination is such an ugly phrase. I prefer to call it *world optimization*."

– Eliezer Yudkowsky, author

(PNM is weak here, so all F&B this time)

Iterative search, steepest descent, nonlinear solvers, Newton, quasi-Newton: F&B 7.6 and 10.1-4