General Mathematics Review, Linear Algebra Overview, and Using Python for Linear Algebra

Julie Butler

Disclaimer: You will not have to do *much* math by hand in this course, but it is good to know how to do some basic operations and what the terminology means.

Scalars

Scalars

- A scalar is just a number, what we are use to dealing with in math.
- Scalars can have a decimal place or be a whole number (called an integer).
- Scalars can be greater than zero (positive) or less than zero (negative)
- Examples:
  - 5
  - 6.7292
  - -8.1

Arithmetic with Scalars

- Scalars can have four basic arithmetic operators performed on them: addition, subtraction, multiplication, and division.
  - Order does not matter for addition and multiplication, but does matter for subtraction and division.
- Example: Give a = 5 and b = 2, find:
  - a+b
  - a-b
  - b-a
  - ab ($a * b$)
  - a/b
  - b/a

Factoring and Distribution

- Consider the scalars a, b, c:
  - If they are arranged as ab+ac, then since both terms of the sum have the same scalar (a) you can *factor* it out: a(b+c)
    - Implicit multiplication
  - If they are arranged as (a+b)(a-c) then we can *distribute* the first part of the term (a+b) to both parts of the second term.
    - (a+b)a - (a+b)c
    - We can then *distribute* the a or the c to both scalars in parentheses
    - aa + ab - ac + bc = a^2 + ab - ac + bc (exponents)
    - Note that we could also have used FOIL here (First, Outer, Inner, Last)

Scalars in Python

```
# Defining scalars (as floats, not ints)
a = 5.0
b = 4.0

# Addition, Subtraction, Multiplication, Division
add = a+b
sub = a-b
mul = a*b
div = a/b
print("Scalar Math:", add, sub, mul, div)

# Multiplication with Parentheses
# Multiplication is not implicit.
mul_paren = (a+b)*(a-b)
print("Multiplication with Partentheses:", mul_paren)
```

Complex Numbers

Complex Numbers

- A complex number has both a real part (a) and an imaginary part (b): z = a+ib
  - i: the imaginary number $i = \sqrt{-1}$
  - Note that the imaginary part is just b, not ib.
- Example: Find the real and imaginary parts of the following numbers.
  - z = 2+3i
  - z = -5i
  - z = 7

Arithmetic with Complex Numbers

- Consider the complex numbers x = a+ib and y = c+id
  - Addition and Subtraction: $x \pm y = (a \pm c) + i(b \pm d)$
    - Add/subtract the real parts and then add/subtract the imaginary parts
  - Multiplication: xy = (a+ib)(c+id) = ac + iad + ibc + i(i)bd = (ac-bd) + i(ad+bc)
    - FOIL two imaginary numbers to multiply
    - Note that $i * i = \sqrt{-1}\sqrt{-1} = (-1)^2 = -1$
  - Example: If x = 2-3i and y = 4+i, find x+y, x-y, and xy.

Complex Conjugate

- The complex conjugate of a complex number z is denoted as z^* and is found by negating the imaginary part of z
- If z = a+ib then z^* = a-ib
- Example: If z = 5+2i, find z^*.

Norm or Modulus

- The modulus squared of a complex number (denoted as $|z|^2$) is just the number times its complex conjugate.
- If z = a+ib then $zz^* = |z|^2 = (a+ib)(a-ib) = a^2 - iab + iab - i(i)b^2 = a^2 + b^2$
- The modulus (or norm) is just the square root of the modulus squared: $|z| = \sqrt{a^2+b^2}$
- The modulus and modulus squared of a complex number is always a real number.
- Example: If z = 4+3i, find the modulus and the modulus squared.

Complex Numbers in Python

- Complex numbers are built into Python, but you denote the imaginary number, i, with a j instead
- Note that because we defined the numbers to be complex, it does not remove the imaginary term, even when it's zero

```
import numpy as np

# Define a complex number
z = 4+3j
print("Complex Number:", z)

# Find the complex conjugate
z_conj = z.conjugate()
print("Complex Conjugate:", z_conj)

# Find the modulus and modulus squared
# Note that the modulus and modulus squared are still
# complex numbers
z_mod_squared = z*z_conj
z_mod = np.sqrt(z_mod_squared)
print("Modulus Squared and Modulus:", z_mod_squared, z_mod

# Basic Arithmetic
x = 1+2j
y = 3-4j
print("Complex Number Math:", x+y, x-y, x*y)
```

Vectors, Bras, Kets (Dirac Notation)

Vectors

- A vector is a one-dimensional structure of numbers arranged in either a row or a column.

$$\vec{a} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

$$\vec{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

- Each number in a vector is called an *element*, the number of elements is the size/length/rank of the vector
  - Elements can be real or imaginary

Magnitude and Unit Vectors

- A vector has a size and a direction.
- The size of a vector is called its magnitude, calculated by summing the square of all the elements then taking the square root.
  - Example: $\vec{a} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$. Find a (also denoted as $|\vec{a}|$), the magnitude of $\vec{a}$
- The direction of a vector is denoted by its unit vector, found by dividing the vector by its magnitude.
  - Example: $\vec{a} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$. Find $\hat{a}$, the unit vector of $\vec{a}$.
  - Note that the magnitude of a unit vector is 1.
- Note that $|\vec{a}|\hat{a} = \vec{a}$

Arithmetic with Vectors

- If you want to add or subtract two vectors, just add or subtract their corresponding components:
- Example: $\vec{a} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ and $\vec{b} = \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$. Find a+b and a-b.
- The vectors must be the same length to be added or subtracted.
- You can multiply or divide a vector by a scalar if you perform that operation on every element of the vector.
- Example: $\vec{a} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ and c = 10. Find ca and a/c.
- Note that you **cannot** divide by a vector.

Dot Products

- There are two ways to "multiply" vectors; the first is called the dot product (also called the inner product and the scalar product)
- Multiple elements in like locations and then add them all together
  - Example: $\vec{a} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ and $\vec{b} = \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$. Find $\vec{a} \cdot \vec{b}$.
  - Note that the result of calculating a dot product is a **scalar**
- Note that $\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}|cos\theta$, where θ is the angle between the two vectors

Cross Products

- The other way to “multiply” vectors is using a cross product, these are a bit complicated to calculate by hand so we won’t do that in this course.
  - However, note that $|\vec{a} \times \vec{b}| = |\vec{a}||\vec{b}|sin\theta$.
- A cross product will produce a **vector** which is perpendicular to both vectors.

Bras and Kets (Dirac Notation)

- In mathematics, vectors will be represented as a, **a**, or $\vec{a}$, almost always with a lowercase letter.
- In quantum mechanics, we use a slightly different notation
  - Greek letters: $\alpha, \beta, \phi, \psi$
  - Ket: $|\psi\rangle \longrightarrow$ always a column vector
  - Bra: $\langle\psi| \longrightarrow$ always a row vector
- A ket can be transitioned into a bra by turning it into a row vector and taking the complex conjugate of each element:
  - Example:

$$|\psi\rangle = \begin{bmatrix} 2-5i \\ 6i \\ 3 \end{bmatrix}$$

$$\langle\psi| = \begin{bmatrix} 2+5i & -6i & 3 \end{bmatrix}$$

Dirac Notation Continued

- A bra and a ket written together form a bracket: $\langle \phi | \psi \rangle$
  - A bracket is equivalent to performing an inner product (dot product)
- Example:

$$\langle \phi | = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

$$| \psi \rangle = \begin{bmatrix} 2 - 5i \\ 6i \\ 3 \end{bmatrix}$$

$$\langle \phi | \psi \rangle = 1(2 - 5i) + 2(6i) + 3(3) = 2 - 5i + 12i + 9 = 7i + 11$$

- Note that we typically what bras and kets to have a magnitude of 1, so $\langle \psi | \psi \rangle = 1$

Vectors in Python

```
import numpy as np

# Define vectors as Numpy arrays
a = np.array([1,2,3])
b = np.array([4,5,6])
c = 10

# Arithmetic with Vectors
add = a+b
sub = a-b
mul = c*a
print("Vector Math:", add, sub, mul)

# Dot product two ways
dot1 = np.dot(a,b) #Order is important
dot2 = a.dot(b) # Order is important
print("Dot Product:", dot1, dot2)

# Cross product
cross = np.cross(a,b)
print("Cross Product:", cross)

# Magnitude and unit vector
mag = np.linalg.norm(a)
unit_vector = a/mag
unit_vector_mag = np.linalg.norm(unit_vector)
```

Matrices

Matrices

- Matrices are two-dimensional structures where each *element* can be a real or imaginary number
  - The width and height can be the same (square matrix) or different. We will mostly deal with square matrices
- Example:

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

Arithmetic with Matrices

- Matrices add and subtract just like vectors (do the operations to like elements), and can be multiplied or divided by a scalar like vectors.
- Example: Given the following matrices and c = 10, find A+B, A-B, cA and A/c.

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

$$B = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

Matrix Multiplication and Commutator

- Two matrices can be multiplied together but this process is tedious to do by hand so we will use Python or other solvers
- In general, $AB \neq BA$, or phrased another way, $AB - BA \neq 0$
- We can define the commutator of two matrices: $[A,B] = AB - BA$
  - if $[A,B] = 0$ (meaning $AB = BA$) we say that A and B commute

Arithmetic with Matrices and Vectors

- You cannot add or subtract vectors or matrices, but you can multiply a vector and a matrix. The result will be a vector
- This again is a bit tedious to do by hand so we will use Python or other solvers

Transpose, Adjoint, Conjugate, and More

- Consider the matrix

$$A = \begin{bmatrix} 1 & 2i & 3 \\ 4+i & 5 & 6 \\ 7 & 8 & 9-3i \end{bmatrix}$$

- The transpose of A, A^T, is found by switching the rows and columns.
- The complex conjugate of A, A^* is found by taking the complex conjugate of every element.
- The adjoint of a matrix, $(A^T)^* = A^\dagger$ is found by taking the transpose **and** the complex conjugate of a matrix If $A = A^\dagger$ then the matrix is said to be *Hermitian*

Dirac Notation

- Matrices are not represented differently in Dirac notation.
- $A|\psi\rangle$ is a matrix multiplied by a vector (gives a vector)
- $\langle\phi|A|\psi\rangle$ is a vector multiplied by a matrix multiplied by a vector

Eigenvalues and Eigenvectors

- Eigenvalues (scalars) and eigenvectors are special values and vectors that each matrix has
  - The eigenvalue, λ and the eigenvector, x, solve the equation $Ax = \lambda x$
- You can think of eigenvalues as the "roots" of the matrix
- Finding these by hand can be complicated, we will use Python or other solvers

Matrices in Python

```
import numpy as np
# Matrices are defined as two-dimensional Numpy arrays
A = np.array([[1,2,3], [4,5,6], [7,8,9]])
B = np.array([[9,8,7], [6,5,4], [3,2,1]])
# Define a vector
x = np.array([2,4,6])
# Define a scalar
c = 10

# Matrix addition and subtraction
add = A+B
sub = A-B
print("Matrix Addition:", add)
print("Matrix Subtraction:", sub)
print()

# Matrix-matrix multiplication
# Do not use the * symbol!!
mul1 = A@B
mul2 = B@A
commutator = mul1 - mul2
print("Matrix Multiplication 1:",mul1)
print("Matrix Multiplication 2:", mul2)
print("Commutator:", commutator)
print()
```

```
# Matrix-Vector Multiplication
mul3 = A@x
print("Matrix-Vector Multiplication:", mul3)
print()

# Matrix-Scalar Multiplication
# Use the * symbol!!
mul4 = c*B
print("Matrix-Scalar Multiplixation:", mul4)
print()

# Transpose and Adjoint
transpose = A.T
adjoint = np.conj(B).T
print("Transpose:", transpose)
print("Adjoint:", adjoint)
print()

# Eigenvalues and Eigenvectors
# Note that the eigenvectors of A are the columns of the r
# eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(A)
print("Eigenvalues:",eigenvalues)
print("Eigenvectors:",eigenvectors)
```