Quick Math!

$$(4+3i)\cdot\begin{pmatrix}3\\1\\4\end{pmatrix}=\begin{pmatrix}12+9i\\4+3i\\16+12i\end{pmatrix}$$

Matrices

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,0} & a_{2,1} & a_{2,2} \end{pmatrix}$$

$$A[1][0] = a_{1,0}$$

Speaker notes

As vectors are 1-D arrays, matrices are 2-D arrays. It's an array of arrays! If you wanted, you could scale up to 3-D arrays and higher dimensions, but we're not going to worry about that We first specify the row, then column This is a square matrix, as it has the same number of rows as columns. It also just looks like a square You might also realize by this point that vectors are just a special case of matrices, where the number of columns is 1

Representation in Code

```
my_vector = [3, 1, 4, 1, 5, 9, 2]
my_matrix = [
  [6, 5, 3, 5, 8, 9],
  [7, 9, 3, 2, 3, 8],
  [4, 6, 2, 6, 4, 3]
]

print(my_vector[0]) # 3
print(my_matrix[0][1]) # 5
```

Basic Matrix Operations

$$\begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} \\ c_{1,0} & c_{1,1} & c_{1,2} \\ c_{2,0} & c_{2,1} & c_{2,2} \end{pmatrix} + \begin{pmatrix} d_{0,0} & d_{0,1} & d_{0,2} \\ d_{1,0} & d_{1,1} & d_{1,2} \\ d_{2,0} & d_{2,1} & d_{2,2} \end{pmatrix}$$

$$= \begin{pmatrix} c_{0,0} + d_{0,0} & c_{0,1} + d_{0,1} & c_{0,1} + d_{0,2} \\ c_{1,0} + d_{1,0} & c_{1,1} + d_{1,1} & c_{1,1} + d_{1,2} \\ c_{2,0} + d_{1,0} & c_{2,1} + d_{2,1} & c_{2,1} + d_{2,2} \end{pmatrix}$$

Speaker notes

Just add each element, pretty straightforward It's just a lot of work to do by hand

$$c \cdot \begin{pmatrix} d_{0,0} & d_{0,1} \\ d_{1,0} & d_{1,1} \\ d_{2,0} & d_{2,1} \end{pmatrix}$$

$$= \begin{pmatrix} c \times d_{0,0} & c \times d_{0,1} \\ c \times d_{1,0} & c \times d_{1,1} \\ c \times d_{1,0} & c \times d_{2,1} \end{pmatrix}$$

Speaker notes

We're multiplying by a scalar here. We cannot multiply two matrices together like this. But like addition, this is fairly straightforward: We just multiply component-wise

Funny Matrix Business

Transpose

"Reflection across the diagonal"

$$A^T[i, j] = A[j, i]$$

Speaker notes

The superscript T denotes the transpose operation

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \end{pmatrix}$$

What is A^T?

Speaker notes

Show math on Jamboard

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \end{pmatrix}$$

What is A^T?

Speaker notes

Show math on Jamboard Note that transpose changes the shape of non-square matrices There are some other operaions such as conjugate, where each element of the resultant matrix is the conjugate of the original. Conjugate as in the pair $3 + 4i$ and $3 - 4i$

Matrix Multiplication

Special condition: $m = n$

Speaker notes

This is probably the most important thing you need to understand from today. Just at least get the general idea Only valid when the number columns in the 1st matrix equals the number of rows in the 2nd. m is the number of columns in the 1st matrix, n is the number of rows in the 2nd I always forget which way it is, don't worry too much. Just get an idea of how it works

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} \\ a_{1,0} & a_{1,1} & a_{1,2} \end{pmatrix}$$

$$B = \begin{pmatrix} b_{0,0} & b_{0,1} \\ b_{1,0} & b_{1,1} \\ b_{2,0} & b_{2,1} \end{pmatrix}$$

What is $\mathbf{AB}$?

Speaker notes

You kind of take teach row of the first matrix, flip it and multiply each element of the second, then add the result and put it in the resulting matrix The resultant matrix should have the same number of rows as the first matrix and the same number of columns as the second matrix Matrix multiplication is **not commutative**

(Not So) Quick Math!

$$\begin{pmatrix} 2 & 7 & 1 \\ 8 & 2 & 8 \end{pmatrix} \begin{pmatrix} 6 & 2 \\ 8 & 3 \\ 1 & 8 \end{pmatrix} =$$

Speaker notes

Show math on Jamboard Let's do this with some actual numbers in the matrix We have a 2 x 3 and a 3 x 2 matrix, so we can multiply. The resulting matrix will have 2 rows, 2 columns

Identity Matrix

For every matrix exists an identity $\mathbf{I}$ such that

$$\mathbf{AI} = A$$

$$\begin{pmatrix} 2 & 7 & 1 \\ 8 & 2 & 8 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 7 & 1 \\ 8 & 2 & 8 \end{pmatrix}$$

Speaker notes

The identity is a square matrix with 1s on the diagonal and 0s everywhere else. You can convince yourself by doing this multiplication

Tensor Products + Factoring

$$\begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \otimes \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} a_0 \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \\ a_1 \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{pmatrix}$$

Speaker notes

You're kind of just multiplying the second vector by each element in the first vector Tensor products are going to be very important for keeping track of the state of our quantum computer This is a tensor product with vectors, but we can do the same thing with matrices. I'm not going to cover it here, but we can do a quick review if we run into them in the future. The idea, though, is basically the same: Take the second matrix and multiply it by each element of the first matrix

Quick Math!

$$\begin{pmatrix} 3 \\ 42 \end{pmatrix} \otimes \begin{pmatrix} 2 \\ 6 \\ 12 \end{pmatrix}$$

Speaker notes

Show math on Jamboard Unsimplified answers are fine

Factoring Product States

$$\begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \otimes \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{pmatrix}$$

Speaker notes

Factoring product states will also be important for quantum computing We know this information holds true, so we can use the product state and our knowledge about tensor products to return to the intial two vectors

More Quick Math!

$$\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Factor this into two vectors

Speaker notes

Show math on Jamboard

Even More Quick Math!

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Factor this into two vectors

Speaker notes

Show math on Jamboard Some product states cannot be factored fully. This is an important fact to remember.

A Few Special Vectors

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Speaker notes

We're going to be very familiar with these vectors and with this notation. The notation you see on the left is called bra-ket notation, and we use it to represent vectors.

Bra-Ket Notation

$$\langle q_0| = (0, 1)$$

$$|\psi\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{pmatrix}$$

Speaker notes

Don't be afraid of this notation. The values are just examples Column vectors, such as the two here, are called **kets** We'll call row vectors **bras**, which we'll run into later This notation is called Dirac bra-ket notation That Greek letter is psi, we're just going to treat it as some arbitrary vector. You'll see this letter a bit later

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$$

$$= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{i}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{i}{\sqrt{2}} \end{pmatrix}$$

Speaker notes

And just for fun, let's see how we can write our vector psi in terms of the 0 and 1 vectors. By the last step, I hope you can see we're back at our original vector for psi

Intro to Qiskit

Speaker notes

Setup: Google Colab and installation

QuantumCircuit, Aer (discuss backends and simulators), assemble

Creating a basic quantum circuit with qubits, classical bits, and measurement gates Qiskit Textbook. The Atoms of Computation

Applying the X gate and the CNOT gate.

Talk about probabilities briefly

Send experiments to a real quantum computer

References

- https://en.wikipedia.org/wiki/Transpose
- Yanofsky, Mannucci. Quantum Computing for Computer Scientists
- https://www.youtube.com/watch?v=2spTnAiQg4M
- https://en.wikipedia.org/wiki/Identity_matrix
- https://qiskit.org/textbook/ch-appendix/linear_algebra.html#Outer-Products-and-Tensor-Products
- https://qiskit.org/textbook/ch-states/representing-qubit-states.html
- Qiskit Textbook. The Atoms of Computation