Linear Algebra for Quantum Computing

Balbinder Singh
Research Scholar, Department of Mathematics, SunRise University, Alwar, Rajasthan (India)
Corresponding Author Email: singhbalbinder1986@gmail.com

Abstract: Quantum linear algebra lies at the heart of many quantum algorithms that promise computational advantages over classical methods. As linear algebra underpins a wide range of scientific computing tasks – from solving systems of equations to eigenvalue problems and signal processing – quantum linear algebra is expected to play a foundational role in quantum-enhanced numerical simulation, machine learning, and optimization. While the theoretical development of quantum algorithms has seen rapid progress, their practical realization is limited by the current state of quantum hardware. Today's devices are constrained by short coherence times, gate infidelities, and limited qubit connectivity – collectively referred to as noise – which significantly restrict the depth and complexity of executable circuits. As a result, many of the most promising quantum algorithms, while asymptotically efficient, are not yet viable for deployment on existing quantum processors. Bridging this gap between theoretical algorithm design and practical implementation is essential to realizing quantum utility in the near term.

[Singh, B. **Linear Algebra for Quantum Computing**. *The International Journal of Interpretation, Observation and Analysis*, 2026; Volume 2, Issue 1 (April-June): Page Number:58-65. ISSN 2349-0713, Peer-reviewed (online/offline), Refereed, Indexed and International Journal (Since 2013), Global Impact Factor: 6.489

Keywords: Linear Algebra, Quantum Computing, Algorithms

Introduction

Linear algebra, the language of matrices and vectors, is the fundamental language of quantum computing and quantum information. Approaching quantum computing through linear algebra is the approach taken in the most cited textbook on the subject: "Quantum Computation and Quantum Information" by Nielsen & Chuang. This approach is also mathematically the most approachable for the complete beginner. If you have very little background in mathematics, physics, and computer programming, this approach is for you!

What Quantum Computing is, and is not

Quantum computing is a way of processing information in a fundamentally different way from the way your laptop or smartphone processes data. The "*classical*" way of processing information is in terms of bits, binary values of zeros and ones. Quantum computing takes advantage of three different properties of quantum physics to process information.

Superposition

Superposition is the fancy word used to describe how **qubits**, the quantum version of bits, can exist in a combination of both zero and one. Qubits can also be measured in different *bases*, allowing for things like *adaptive measurement* of states, and *measurement based quantum computation*. Since qubits can be in a state that is a mixture of both zero and one, this gives them computational properties that classical bits don't have.

Entanglement

Entanglement is the word used to describe how qubits can have states that are correlated with other qubit states. Qubits interact with each other and become *entangled*, meaning their states are now statistically correlated and information about one can sometimes reveal information about the other. Multiple qubits can be entangled at once and complicated behavior can emerge. One area of research that scientists have shown much interest in is **multipartite entanglement**. This is a complex form of entanglement that exhibits highly non-classical behavior. This can be used in quantum teleportation and quantum cryptography protocols. It is also the source of a famous experiment named after Einstein, Podolsky, and Rosen, as well as John Bell's famous experiments showing that local hidden variables theories are highly unlikely to be true.

Interference

Interference, like the interference we see in waves of light, sound, and water, is exhibited by quantum computers. Quantum computers can have *constructive* (additive) interference, or they can have *destructive* interference, where waves are out of sync and cancel. This interesteing and strange behavior is often the subject of debate when discussing *wave-particle duality* and the famous *measurement problem of quantum physics*. Interference can be used to enhance "good information" and to cancel "bad information" in computations. This allows us to set up computations in clever ways and use interference as an information processing technique to solve problems.

Quantum Computers as ASICs

ASICs or **Application Specific Integrated Circuits** are specialized computer chips that are built

Paper received: 17.05.2026 Reviewed: 22.05.2026 Accepted: 25.05.2026 Published: 01.06.2026

for very specific kinds of computational tasks. A good example of this the GPU or *Graphics Processing Unit*, or Google's TPU (Tensor Processing Unit) which have enhanced machine learning drastically in recent years. Other examples might be the *neuromorphic computing chips* that Intel has designed. These were all designed to solve specific kinds of problems, such as enhancing machine learning tasks. Quantum computers can be thought of in a very similar way. There is a classical computer interface that delegates certain computational tasks to the quantum hardware. The quantum hardware then processes that computational task and then the classical computer receives the output data. There are many kinds of problems that quantum computers can solve. In fact, any computational task a classical computer can solve, a quantum computer can also solve. However, classical computers are still more efficient at certain things, so they will not be going anywhere. There are many algorithms and problems that can be solved exponentially faster on a quantum computer compared to all known classical methods.

Finding out where this computational advantage comes from is one of the central tasks of quantum computing research. There is a Quantum Algorithm Zoo, which lists many of the known algorithms that give speedup over classical methods. Much of the focus of The Singularity is to research these various applications and algorithms and implement them in various quantum computing languages. Some applications include using Shor's algorithm to break RSA public key cryptography and other algorithms that break elliptic curve cryptography. If you are interested in discovering the inner workings of quantum algorithms and trying to understand where this quantum supremacy or advantage comes from, check out these lectures.

Quantikz

If you are a Donald Knuth fan and love LaTeX too, check out the TikZ library quantikz. It is a LaTeX library for constructing quantum circuit diagrams. It was used for rendering many of the quantum circuit diagrams in the following lectures.

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

Basis vectors for **V**

Also, both of the above vectors belong to the space **V.** So, the first condition is fulfilled. If you remember how dot product works, we have ,

as our basis vectors for now.

$$\begin{bmatrix}1\\0\end{bmatrix} \cdot \begin{bmatrix}0\\1\end{bmatrix} = 0$$

Dot product of bases

which proves that it is a valid *basis set* for the vector space **V.** You may also easily prove that

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

Basis for 3-D vector space

available in the denominations of 50 and 100 units. So although money can only be dealt in forms of

forms a basis set for a 3-D vector space.

Following is a useful analogy to think of the basis vectors. Imagine that a country only has the currency —

$$Amount = a\begin{bmatrix}50\\0\end{bmatrix} + b\begin{bmatrix}0\\100\end{bmatrix}$$

Money being dealt in currency basis

of currency. This is like saying that 50 and 100 form a basis for the 2-dimensional currency space!

it reveals that whatever transaction happens *can be expressed as a linear combination of 50 and 100 units*

Now, why do we care about basis vectors?

This is because every quantum state, by definition, belongs to a vector space. Since each quantum state may have any arbitrary value, it can be difficult to accurately record their unique state. A **basis** provides a way to write down quantum states in a defined

https://ijioa.com/ cosmospub2023@gmail.com

Paper received: 17.05.2026 Reviewed: 22.05.2026 Accepted: 25.05.2026 Published: 01.06.2026

manner and thus only look at how *close* that state is to a certain basis vector.

With the above knowledge we can now look how we represent quantum states in a basis and then ultimately go on to combine them. Since quantum states belong to a 2-D complex vector space, they can be represented as a *linear combination* of its basis vectors (as we proved above). Reason for that is a convention that is followed in quantum mechanics and is actually based on empirical evidence.

$$|\Psi\rangle = a\begin{bmatrix}1\\0\end{bmatrix} + b\begin{bmatrix}0\\1\end{bmatrix}$$

Representing a quantum state in the 2-D basis

Now that you are familiar with how quantum states are represented in a basis let us move on to how we may combine *two or more states*. Let us start with an analogy.

Suppose you have a quantum die and a quantum coin. We know that a normal six sided die can have only one of the six possible outcomes when rolled and same is the case for the coin i.e. it can only have two outcomes — heads or tails. Their quantum versions are quite similar.

The states of a quantum coin can be represented in the basis —

$$\{|H\rangle, |T\rangle\}$$

Basis states for quantum coin

where the basis states are represented in Dirac notation — which is just a convenient way to write quantum states.

The states of the quantum die on the other hand can be represented as a linear combination of the following six basis states —

$$\{|1\rangle, |2\rangle, |3\rangle, |4\rangle, |5\rangle, |6\rangle\}$$

Basis states for quantum die

So we say that any state of the quantum coin can be represented as a linear combination of |H> and |T> viz — α|H> + β|T> where α,β **∈ C.**

The states of the die on the other hand can be represented as a linear combination of *its* six basis states viz —α|1> +β|3> +γ|4> where all coefficients are complex and some may even be zero.

This analogy can help you understand bases and how they help in the representations of quantum systems.

If you have been paying close attention, you'll notice that we have only taken a *single system* into account during our discussions. You may think that "wait, if a die has 6 possible states, shouldn't that be six systems?" The point is that a die required *6 basis vectors* to define **its one state.** The die is what constitutes a system, not the basis vectors.

This is all fine and good but where are tensor products and how do they fit in everything we discussed ? Let us find out.

The intuition behind Tensor products

What if we want to combine systems?

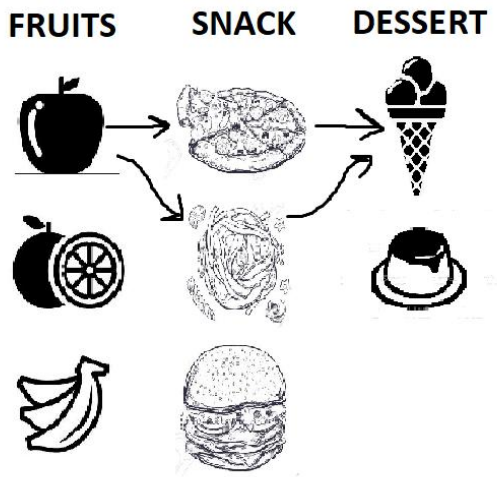
Think of tensor products as a way of capturing all possible combinations of basis vectors. Bases of what? Different systems. As we saw above, basis vectors are representatives of how we define a system. So, it makes sense to define *combination of systems* as a combination of the *constituents of the system* since any other component (vector) can be described in terms of those fundamental components. The following analogy would make it clearer what we mean by combinations.

Suppose you wanted to eat a meal consisting of 3 things — a fruit, snack and a dessert. The starter has a choice of 3 fruits: apples, oranges or bananas. The snack has options of a burger, pizza or pasta, and finally dessert has 2 options, ice cream or custard (quite fulfilling I suppose!)

In how many possible ways can you eat your meal?

Let's define an order for our three courses as follows —

Paper received: 17.05.2026 Reviewed: 22.05.2026 Accepted: 25.05.2026 Published: 01.06.2026



An analogy for how tensor products work where items are the different bases.

How can you go on selecting a meal then ? You first select a fruit, then a snack and lastly a dessert. Referencing the image, one option may be that you eat an apple, a pizza and then ice-cream or it may be that you like pasta more and you go on eating that instead of the pizza.

The Mathematics

The term tensor product contains a tensor. What is that? A Tensor is a container which can house data in *N* dimensions. Tensors are in fact generalizations of matrices to ***N*-dimensional** spaces, which is why they're often used interchangeably with the matrix, (which is specifically a 2-dimensional or Rank2 tensor) .

Let us formally define the tensor product now and dissect it bit by bit, following with some examples.

The **tensor product** V ⊗ W of two vector spaces V and W is a vector space, containing a bilinear map { (v,w) -> v⊗ w} from the Cartesian product V × W to V⊗ W.

The above statement means that any two vector spaces **V** and **W** can be combined with the tensor product to generate a new vector space **V** ⊗ **W.** Now, this vector space (which we'll call **Z** for now) contains vectors which are formed by taking all the possible combinations of vectors from **V** and **W,** and generating a new vector through the operation '⊗' between them.

Without going into the mathematical notations a lot, just remember that another way to think of tensor products is through the combination of basis vectors. The new tensor product space generated is essentially a *cross product* of the basis sets of the vector spaces **V** and **W.** Let us see a concrete example to clear things up a bit.

We bring back the quantum die and the quantum coin to illustrate the combination of multiple systems. The first system is our die and the second, the coin. Let the vector space **C** represent the vector space of the coin with the basis set **{|H>,|T>}.** Also, let the die be represented by the vector space **D** which has its basis set — **{|1>,|2>,|3>,|4>,|5>,|6>}.**

Now, given that we have the individual systems, let us look at what the combined system would look like. Following up from our above discussion, this system is defined as the vector space **C** ⊗ **D.** Some of the basis states for the space would look like |1>⊗ |T> or |2> ⊗ |H>. The following table captures the possibilities quite nicely —

Press enter or click to view image in full size

BASIS	|1>	|2>	|3>	|4>	|5>	|6>
|T>	|T>|1>	|T>|2>	|T>|3>	|T>|4>	|T>|5>	|T>|6>
|H>	.|H>|1>	.|H>|2>	.|H>|3>	.|H>|4>	.|H>|5>	.|H>|6>

https://ijioa.com/ cosmospub2023@gmail.com

Paper received: 17.05.2026 Reviewed: 22.05.2026 Accepted: 25.05.2026 Published: 01.06.2026

The combined basis vectors for the coin-die system
Note that the dimensionality of this product space is **m x n** where m is the size of the set of basis vectors of first vector space & n, the second.
Tensor product is also defined on elements of a vector space & note that unlike matrix multiplication, the elements' dimensions *do not have to commute.* This means that a 4 x 1 vector would be able to 'tensored' with a 3 x 1, 5 x 1, 6 x 1 … vectors too. In fact, any *tensor* of shape (n, m) can be 'tensored' with any other *tensor* of shape (p, q) to produce a new tensor of shape (np x mq). Also, like matrix multiplication, a thing to note is that a ⊗ b ≠ b ⊗ a . The following example would make it clear how the workings of calculating a tensor product go about.
Suppose you have two vectors —

$$a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Vectors a and b
and you wish to calculate the tensor product of these two vectors i.e. a ⊗ b The way to capture all the combinations is, as we defined above, making pairwise multiplications of the elements in the vectors. Another useful way to represent that is highlighted in the following procedure.

The *b vector* is actually treated as an 'element' and then multiplied by each of the element of the first vector. Now, this element is expanded and the *components of the first vector* are multiplied by the *second vector* element wise. This image would clear the above steps further —

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \otimes \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_1(b) \\ a_2(b) \end{bmatrix}$$

Step 1 — Treat the second vector as an element

$$\begin{bmatrix} a_1(b) \\ a_2(b) \end{bmatrix} = \begin{bmatrix} a_1(\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}) \\ a_2(\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}) \end{bmatrix}$$

Step 2 — Expand the vector

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \otimes \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_1 b_1 \\ a_1 b_2 \\ a_1 b_3 \\ a_2 b_1 \\ a_2 b_2 \\ a_2 b_3 \end{bmatrix}$$

Step 3 — Multiply and get the result
The final results for tensor products can get a bit scary, especially when matrices get involved, but always remember to *treat the second vector as an element* and the rules are pretty straightforward, which would lead you to the correct result.
Now that we have the knowledge of tensor products under our belt, let us go look into how they are employed in quantum computing.

This is actually one of the fundamental postulates of quantum systems where if you have n systems with individual quantum states:

Application in Quantum Computing
Superposition, entanglement, and quantum interference are the phenomena which actually make the quantum model a more powerful model of computation than classical computing. How do we leverage these tools? Unless we expand our systems to *multiple qubits*, we can't really employ them to our advantage. How do we manipulate or study *multi-qubit* systems?
The answer to that is *tensor products.*

https://ijioa.com/ cosmospub2023@gmail.com

Paper received: 17.05.2026 Reviewed: 22.05.2026 Accepted: 25.05.2026 Published: 01.06.2026

$$|\Psi_0\rangle, |\Psi_1\rangle, |\Psi_2\rangle, \ldots, |\Psi_n\rangle$$

Multiple quantum systems

Press enter or click to view image in full size

The combined system describing these n systems as one unified quantum state is defined by the tensor product of all the states —

$$|\Phi\rangle = |\Psi_0\rangle \otimes |\Psi_1\rangle \otimes |\Psi_2\rangle \otimes \ldots \otimes |\Psi_n\rangle$$

The combined quantum system

Think of it like having n particles with you in a closed box. How do you capture *every possible combination* of that group of particles? This echoes the very definition of tensor products and their capacity for describing multiple combinations of systems, and is thus the reason as to why physicists used tensor products to characterize a multi-qubit system.

For an example, consider two qubits in the |0> and the |1> states —

$$|\psi_1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |\psi_2\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

How do we combine the state of these two qubits? We take the *tensor product* of the two and if, say, the first qubit is in |0> and the second qubit is in |1> state, we get —

$$|\phi\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

The final state vector

This state vector actually signifies that this system is described by a 4-dimensional basis. Why? Well there can be 4 possible combinations for a 2-qubit system — both |0>|0> , both |1>|1> or 2 different states where either first is |1> and second |0> or vice versa. We can also have more than 2 qubit systems where the system is just represented as the 2^n sized vector but for simplicity we look at 2 qubits.

Finally, let's look at how the operations done on qubits are transformed when systems go beyond single element. If you are familiar with the basic gates of quantum computing, you would be familiar with the bit flip gate or the **X** gate. For those of you who aren't, just think of it as —

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

What if we wanted to apply gates to our multi-qubit system? Let's take the above system only — first qubit is in |0> state and the second in the |1> state. If you want to apply an **X** gate on the first qubit only, how would you go about it? Again, using *tensor products.*

Any combination of gates being applied individually on a multi-qubit system is again defined as a tensor product of the gates applied to the *whole system.* Why this is true can be seen by this identity which is —

$$(A|\Psi\rangle) \otimes (B|\Phi\rangle) = (A \otimes B)(|\Psi\rangle \otimes |\Phi\rangle)$$

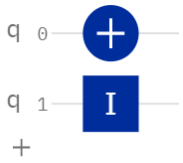
Identity for tensor product of products

The proof for that identity can be done by simply evaluating both of the sides and comparison of corresponding elements, for general vectors and matrices. While not going into that, what we'll see is an example to clear how **X** gate can be applied to a single qubit in a multi-qubit system.

Assume that the **X** gate was to be applied over the first qubit and the second qubit were to be left as it is. We could model it as-

$$(X|\psi_1\rangle) \otimes (I|\psi_2\rangle) = (X \otimes I)(|\psi_1\rangle \otimes |\psi_2\rangle)$$

Applying X gate on the first qubit



https://ijioa.com/ cosmospub2023@gmail.com

Paper received: 17.05.2026 Reviewed: 22.05.2026 Accepted: 25.05.2026 Published: 01.06.2026

Composed circuit in IBMQ experience

$$X \otimes I = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0(I) & 1(I) \\ 1(I) & 0(I) \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix}$$

The tensor product for the matrices as the identity (**I**) simply models the 'do nothing' result we want on the second qubit. With a little bit of calculation you may confirm that it results in the following state-

$$|\Phi\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Resultant State

Although we admit that the actual intuition behind 'combinations' gets muddled up quickly as and when we go into more number of elements, the basic rules to evaluate, work out, & play around with tensor products is what is really important in the initial stages.

Conclusion: the tensor product is a *tool* to breakthrough into the domain of multiple elements & thus expand our abilities to model and study larger systems. In this post, we looked at what tensor products mean intuitively, dug into the math and finally explored how multiple qubits are represented mathematically. I hope the essence of what tensor products mean was not lost in the mathematics and you had fun reading it.

References:

1. Aaronson, S., Ambainis, A.: Quantum search of spatial regions. Theory of Computing, 1, 47– 79 (2003)
2. Abal, G., Donangelo, R., Marquezino, F.L., Portugal, R.: Spatial search on a honeycomb network. Math. Struct. Comput. Sci. 20(Special Issue 06), 999–1009 (2010)
3. Aharonov, D., Ambainis, A., Kempe, J., Vazirani, U.: Quantum walks on graphs. In: Proceedings of 33th STOC, pp. 50–59. ACM, New York (2001)
4. Aharonov, D.: Quantum computation – a review. In: Stauffer, D. (ed.) Annual Review of Computational Physics, vol. VI, pp. 1–77. World Scientific, Singapore (1998)
5. Aharonov, Y., Davidovich, L., Zagury, N.: Quantum random walks. Phys. Rev. A 48(2), 1687–1690 (1993)
6. Aldous, D.J., Fill, J.A.: Reversible Markov Chains and Random Walks on Graphs. Book in preparation, http://www.stat.berkeley.edu/aldous/RWG/book.html (2002)
7. Ambainis, A., Bach, E., Nayak, A., Vishwanath, A., Watrous, J.: One-dimensional quantum walks. In: Proceedings of 33th STOC, pp. 60–69. ACM, New York (2001)
8. Ambainis, A., Backurs, A., Nahimovs, N., Ozols, R., Rivosh, A.: Search by quantum walks on two-dimensional grid without amplitude amplification. arxiv:1112.3337 (2011)
9. Ambainis, A.: Quantum walk algorithm for element distinctness. In: FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 22–31. IEEE Computer Society, Washington, DC (2004)
10. Ambainis, A., Kempe, J., Rivosh, A.: Coins make quantum walks faster. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1099–1108. SIAM, Philadelphia (2005)
11. Apostol, T.M.: Calculus, vol. 1: One-Variable Calculus with an Introduction to Linear Algebra. Wiley, New York (1967)
12. Axler, S.: Linear Algebra Done Right. Springer, New York (1997)
13. Bednarska, M., Grudka, A., Kurzynski, P., Luczak, T., Wojcik, A.: Quantum walks on cycles. ´ Phys. Lett. A 317(1–2), 21–25 (2003)
14. Bednarska, M., Grudka, A., Kurzynski, P., Luczak, T., Wojcik, A.: Examples of non-uniform ´ limiting distributions for the quantum walk on even cycles. Int. J. Quant. Inform. 2(4), 453–459 (2004)
15. Benioff, P.: Space searches with a quantum robot. (ed.) Samuel J. Lomonaco, Jr. and Howard D. Brandt Contemporary Mathematics, AMS, as a special session about Quantum Computation and

Paper received: 17.05.2026 Reviewed: 22.05.2026 Accepted: 25.05.2026 Published: 01.06.2026

Information, vol. 305, pp. 1–12. Washington, D.C (2002)
16. Bennett, C.H., Bernstein, E., Brassard, G., Vazirani, U.V.: Strengths and weaknesses of quantum computing. SIAM J. Comput. 26(5), 1510–1523 (1997)

https://ijioa.com/ cosmospub2023@gmail.com

Paper received: 17.05.2026 Reviewed: 22.05.2026 Accepted: 25.05.2026 Published: 01.06.2026