Linear Algebra Project: Applications of Linear Algebra in Computer Science Engineering

ISHAM SINHA-PES1UG21CS598
SKANDA SHREESHA PRASAD-PES1UG21CS603
SHREYAS M-PES1UG21CS581
SIDDARTH RAO-PES2UG21CS521

Importance of Linear Algebra in Computer Science:

Linear Algebra is a fundamental part of mathematics, science and computer science engineering. In the domain of computer science engineering, a good understanding of linear algebra is vital, especially while working with machine learning and deep learning models. Almost all the features that are available in computer graphics and vision, for instance, transformation of an image, image processing, or introducing visual effects to a photo or video, or when you search up information on the internet, or making a phone call, one thing they have in common is that the technology used is built on Linear Algebra. The concepts of Linear Algebra are in use in many applications within the many areas of Computer Science, such as graphics, image processing, image transformation, cryptography, machine learning, computer vision, quantum computing, computational biology, information retrieval and web searches.

Introduction to Quantum Computing:

Quantum computing is a multidisciplinary field that uses concepts of computer science, mathematics and physics to solve problems that can be solved faster than can be solved with classical computers or supercomputers. On a microscopic scale, physical matter tends to show both wave and particle properties. Classical physics cannot explain the operation of these devices and a scalable quantum computer can perform some calculations exponentially faster than any modern “classical” computer. Quantum computing leverages this concept using some specialised hardware.

The basic functional unit of a quantum computer is called a qubit. A qubit must satisfy the condition $|a|^2+|b|^2=1$. Unlike a classical bit, a qubit can exist in a superposition in its two basic states. If a quantum computer manipulates a qubit in any way, wave interference effects may amplify the desired measurement results. In theory, any problem solved by a classical computer can be solved with a quantum computer. Conversely, any problem solved by a quantum computer can also be solved by a classical computer, at least in principle, if given enough time.

Quantum computers obey the Church-Turing thesis, meaning that while quantum computers provide no additional advantage computationally compared to classical computers, quantum computers, for certain problems, have significantly lower time complexities when compared to classical computers for those same problems.

Applications of Linear Algebra in Quantum Computing:

Linear Algebra is the language of quantum computing. Although it is not important to know it to implement or write quantum programs, it's widely used to describe qubit states, quantum operations, and to predict what a quantum computer does in response to a sequence of instructions. Just like knowing the basics of quantum physics can help you understand quantum computing, knowing the basics of linear algebra can help you understand how quantum algorithms work.

It is therefore vital to develop a good understanding of the basic mathematical concepts that linear algebra is built upon, in order to arrive at the computations as seen in quantum computers. The many concepts of linear algebra used in quantum computing are vectors and vector spaces, matrices and matrix operations, spanning sets, linear dependence and bases of a vector space, orthonormality, eigenvalues and eigenvectors, matrix exponentials, etc. to name a few. We will look at the applications of some of the above concepts down below.

Applications of Quantum Computing in real time:

Quantum computing has opened up opportunities in several disciplines and industries, from pharmaceuticals, to chemical engineering, to information and communications technology to finance, automotives and aerospace. Here are a few applications that have sprung up as results of quantum computing

- Machine Learning: Quantum computing can fine-tune algorithms such as neural networks that are used extensively in machine learning. Moreover, popular deep learning and machine learning problems used broadly for optimisation purposes can be optimised further using quantum computers as they combine classical and quantum simulations to solve problems. As these computers can process large quantities of data, it can aid in better decisions and predictions, such as in applications such as facial recognition, object recognition and fraud detection.

- Drug development: Quantum computing plays a crucial role in drug development, wherein drugs can be tested for stable molecular configuration using molecular modelling processes. Also quantum computing can run advanced simulations on various participating organic molecules that help decide the sustainability of the organic molecules for the drug. It's typically suitable for addressing combinatorial optimisation problems where specific molecules necessary for drug development aren't known or available.

- Model chemical processes: Quantum computing plays a pivotal role in designing molecular structures, such as the nitrogenous enzyme observed in ammonia based fertilisers. Quantum based simulations can be used to model chemical processes and complex atomic interactions, which are processed faster than traditional methods of lab experimentation which apply trial and error techniques

- Finance: Quantum computing can overcome the drawbacks of traditional algorithms that perform poorly in time-sensitive financial transactions. It can help in stock-portfolio management, investments and financial trading. Quantum computing can be used for portfolio optimisations by banks for processing, scheduling and prioritising several financial transactions. One can harness quantum computing through quantum processors, a combination of multiple qubits that use quantum properties to determine the best possible answer.

- Aircraft development: Companies can use quantum computing in the process of aircraft design. The entire aircraft can be modelled and digitalised which facilitates faster simulation. It can perform mathematical calculations faster, thereby boosting aircraft design efficiency. They also use quantum principles for other purposes, such as optimising, fuel consumption and managing the aircraft's speed, which will fulfil the sustainability targets of the industry

- Automotive industry: Automotive industry is already benefiting from the quantum computing paradigm. For example, Volkswagen has partnered with quantum computing provider 'D-Wave' to leap into the quantum enabled world. In 2019, a few VW buses were fitted with a navigation app designed by D-Wave that offers real-time quantum services, such as information on congestion data or the best possible routes to a destination scaled down to the millisecond time frame. Similarly, Mercedes-Benz Group partnered with IBM in January 2020 to develop next-generation lithium batteries with the help of quantum computers.

- Development of climate models: Today's climate models used to make weather forecasts aren't as accurate as they need to be. This is primarily because several inputs representing real-world conditions are required to be simulated. Classical computing systems cannot handle large quantities of data inputs. Quantum computing can accommodate as many inputs that can help develop accurate climate models. They can simulate the minutest of environmental variables such as wind, temperature, humidity, pressure, etc., and understand how they respond to different weather conditions in the atmosphere. Such granular details can allow researchers to calculate and track several environmental parameters in real-time, which will help in accurate weather prediction

- Solid material development: Researchers can use quantum computing to design quantum-mechanical simulations that determine the characteristics and properties of advanced materials. The quantum Monte Carlo model can be used where the internal structure of solid materials can be modelled, enabling the discovery of entirely new materials that are far more difficult for conventional computing models. For example, in 2020, the researchers at the University of Chicago used IBM's 53-qubit Hummingbird quantum processor to develop a jew quantum material, 'exciton condensate', through quantum simulations.

Vectors and matrices:

A vector is defined as the elements belonging to a set known as a vector space. In quantum computing, we often deal with state vectors, which can be visualised using a Bloch sphere, the state space of all the possible points to which these vectors can point to.

As specified above, a qubit is the basic functional unit of a quantum computer that must satisfy the given condition, $|a|^2+|b|^2=1$. A qubit can be in a state of 0 or 1 or in a superposition of both. Using linear algebra, a qubit can be described as a vector and can be defined with a single column matrix . It is also known as a quantum state vector. The elements of the matrix present the possibility of the qubit collapsing to the 0 state or the 1 state, with $|a|^2$ being the probability of collapsing to the 0 state and $|b|^2$ being the probability of collapsing to the 1 state. The following single column matrices all represent valid quantum state vectors:

$$\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}, \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{-1}{\sqrt{2}}\end{bmatrix}, \text{and } \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{i}{\sqrt{2}}\end{bmatrix}.$$

Quantum operations can also be represented in terms of matrix operations. When a quantum operation is applied to a qubit, the two matrices that represent them are multiplied and the resulting answer represents the new state of the qubit after the operation. A matrix that represents a quantum operation must be a unitary matrix. A matrix is unitary if the inverse of the matrix is equal to the conjugate transpose of the matrix.

Vector spaces used in qubit states:

We have been introduced to what a qubit is in quantum computing, and applying an operation to it was described by multiplying the two matrices. However quantum computers use more than one qubit.

We must know that each qubit is a vector space, and they cannot just be multiplied, instead tensor product is used. It's an operation that creates new vector spaces from individual vector spaces and is represented by $\otimes$. For example, the tensor product of two qubit states are calculated by:

$$\begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} a \begin{bmatrix} c \\ d \end{bmatrix} \\ b \begin{bmatrix} c \\ d \end{bmatrix} \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix}.$$

The result is a four dimensional matrix, with each element representing a probability. For example |ac| is the probability of two qubits collapsing to 0 and 0, |ad| being the probability of collapsing to 0 and 1, and so on. Just as a single qubit state must meet the requirement $|a|^2+|b|^2=1$ in order to represent a quantum state, a two qubit state [ac, ad, bc, bd] must satisfy the condition $|ac|^2+|ad|^2+|bc|^2+|bd|^2=1$.

Eigenvalues and eigenvectors in quantum computing:

A vector v is an eigenvector of M if Mv=Ev, for some number E. Here, E is any integer known as eigenvalue, corresponding to the eigenvector v. An eigenvector is special as it's left unchanged except for being multiplied by a number.

In quantum computing, there are essentially two matrices we concern ourselves with, namely Hermitian matrices and Unitary matrices. A Hermitian matrix (also called self adjoint matrix) is a complex square matrix which is equal to its own complex conjugate transpose, while a unitary matrix is a complex square matrix whose inverse equals its own complex conjugate transpose. $A=S*D*S^T$, and similarly for any power p, $A^p=S*D^p*S^T$.

Matrix exponentials in quantum computing:

A matrix exponential can also be defined in exact analogy to the exponential function. The matrix exponential of matrix A can be expressed as:

$$e^A = \mathbf{1} + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots$$

This is important as quantum mechanical time evolution is described by a unitary matrix in the form of a^{iB} for Hermitian matrix B. For this reason, performing matrix exponentials is a crucial part of quantum computing and as such Q# offers intrinsic routines for describing these operations. There are many ways of computing matrix exponentials using classical computers, not without great difficulty. The easiest way of computing matrix exponentials is through eigenvalues and eigenvectors of a matrix**.** From a number of methods of diagonalising a matrix A, there exists a matrix S containing the eigenvectors, each as a column vectors and a diagonal matrix D such that $A=S*D*S^T$, and similarly for any power p, $A^p=S*D^p*S^T$.

As many operations in quantum computing involve performing matrix exponentials, the trick is transforming into eigenbasis of a matrix to simplify performing the operator exponential appears frequently. It's the basis of many quantum algorithms and other quantum simulation methods.

Qubit:

It's the quantum equivalent of a bit of classical computer. While a bit can have either 0 or 1 as the values, the quantum bit can have a value that's either 0, or 1 or a quantum superposition of 0 and 1.

The state of a qubit can be described by a 2-dimensional column vector of unit norm. This vector, called the quantum state vector, holds all the information needed to describe the one-qubit quantum system. The quantum state vectors [1, 0] and [0, 1] play a special role.These two vectors form the basis for the vector space that describes the qubit's state. This means that any quantum state vector can be written as a sum of these two basis vectors. While any rotation of these vectors would serve as a perfectly valid basis for the qubit, this particular set of bases is selected, making them the computational bases.

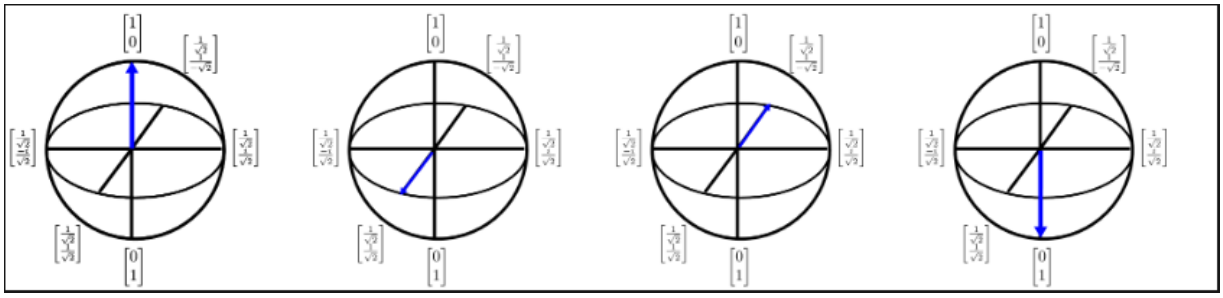
Measuring a qubit:

A measurement corresponds to the informal idea of "looking" at a qubit , which immediately collapses to one of the quantum states to one of the two classical states [1,0] and [0,1]. The properties of measurement also mean that the overall sign of the quantum state vectors is irrelevant, because the probability of measuring 0 and 1 depends on the magnitude of squared terms

Another important property of measurement is that it doesn't necessarily damage all quantum state vectors. If one starts in the qubit in the state [1, 0] which corresponds to the classical state 0, measuring this state will always yield the outcome 0 and leave the quantum state unchanged. If there are only classical bits, then measurement doesn't damage the system. This means that one can replicate classical data and manipulate it on a quantum computer and just as one could do on a classical computer.

Visualisation of qubits and transformations using Bloch sphere:

Qubits are pictured in 3-d using the Bloch sphere representation. The Bloch sphere gives way to describing a single-qubit quantum state as a three dimensional real valued vector. This is important because it helps us visualise single-qubit states and thereby develop reasoning that can be invaluable in multi-qubit states. The Bloch sphere can be visualised as follows:



The arrows in this diagram show the directions in which the quantum state vector is pointing and each transformation of the arrow can be thought of as a rotation about one of the cardinal axes.

While thinking about a quantum computation as a sequence of rotations is a powerful institution, it is challenging to use this institution to design and describe algorithms, although Q# alleviates this issue by providing a language for describing such rotations.

Single qubit operations:

Quantum computers process data by applying a universal set of quantum gates that can emulate any rotation of the quantum state vector. This universality is similar to the universality for traditional computing, where the gate set is considered to be universal if every transformation of the input bits can be performed using a finite length circuit.

In quantum computing, the valid transformations that are allowed to be performed on a qubit are unitary transformations and measurements. The adjoint operation or the complex conjugate transpose is important to quantum computing because it is needed to invert quantum transformations. Single qubit quantum gates can be classified into two categories: Clifford gates and non Clifford gates. Non Clifford gates consist of T gates, (also known as $\pi/8$ gates).

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$$

The standard set of single-qubit Clifford gates include

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} = T^2, \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = HT^4H,$$
$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = T^2HT^4HT^6, \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = T^4.$$

Here, the X, Y, Z operators are called Pauli operators, after Wolfgang Pauli. Together with non Clifford gates, these operations can be performed to approximate any unitary transformations on a single qubit.

The simplest primitive operation is the single qubit rotation. Three single qubit rotations are generally considered: R_x, R_y and R_z. Its corresponding unitary matrices are:

$$R_z(\theta) = e^{-i\theta Z/2} = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix},$$
$$R_x(\theta) = e^{-i\theta X/2} = HR_z(\theta)H = \begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix},$$
$$R_y(\theta) = e^{-i\theta Y/2} = SHR_z(\theta)HS^\dagger = \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}.$$

Just as three rotations can be combined together to perform an arbitrary rotation in three dimensions, it can be seen from the Bloch sphere representation that any unitary matrix can be written as a sequence of three rotations as well. For this reason and due to applications in quantum simulation, such continuous gates are crucial for quantum computation, especially at quantum algorithm design level.

Multiple qubits:

While single-qubit state has the ability to stay in more than one state at any given time, if single-qubits were all the quantum computers could offer, then the calculator and a classical supercomputer would dwarf its computational power.

Quantum computing power arises, in part, because the dimension of the vector space of the quantum state vector grows exponentially with the number of qubits. Increasing the size of the computation by only one extra qubit doubles the memory required to store the state and roughly doubles the computational time. This rapid doubling of computational power is why a quantum computer with a relatively small number of qubits can surpass the most powerful supercomputers for some computational task.

Two qubit states:

If there are two separate qubits, the corresponding two-qubit state is given by the tensor product of the vectors. Therefore, given two single-qubit states a and b, each of dimension 2. The corresponding two-qubit state after the tensor product is four dimensional.

$$\begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix}$$

The vector represents a quantum state on two qubits if $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$.

The computational basis of two qubit-states is formed by the tensor products of one qubit basis states.

Measuring two qubit states:

Measuring two qubit states is very similar to measuring single qubit states. For a two-bit quantum state [ac, ad, bc, bd], measuring the state yields 00 with a probability $|ac|^2$, 01 with a probability $|ad|^2$ and so on and so on. After the measurement, if the outcome is 00, then the quantum state of the two-qubit system has collapsed and has become [1,0,0,0]

It's also possible to measure just one qubit of the two-qubit quantum state. While measuring one qubit out of two-qubit quantum state, the impact of measurement is subtly different than measuring two qubits. This is because the entire state isn't collapsed to a computational basis state, rather it has collapsed to only one subsystem.

Two-qubit system operations:

As in the single qubit case, any unitary transformation is a valid operation on qubits. In general, a unitary transformation on n qubits is a matrix of size $2^n x 2^n$, such that $U^t=U^{-1}$. For example, a CNOT (Controlled NOT) gate is a commonly used two qubit gate and is represented as such:

$$\mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

We can also form two-qubit gates by applying single qubit gates on both qubits. For example:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \otimes \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae & af & be & bf \\ ag & ah & bg & bh \\ ce & cf & de & df \\ cg & ch & dg & dh \end{bmatrix}$$

This is equivalent to applying the two-qubit unitary given by their tensor product.

Gates can also be controlled using classical information. A classically controlled NOT gate is just an ordinary NOT gate but it's only applied if a classical bit is 1 as opposed to a quantum bit. In this case, a classically controlled gate can be thought of as a statement in a quantum code wherein the gate is applied to only one branch of the code.

Quantum entanglement:

When we take an example of two qubits A and B in superpositions such that the state of the global system is

$$|\psi\rangle_{AB} = \frac{1}{\sqrt{2}}|00\rangle \frac{1}{\sqrt{2}}|11\rangle$$

In such a state, only two outcomes are possible when you measure the state of two qubits in the standard basis: 00 and 11. Each outcome has the same probability of 0.5. There's 0 probability of obtaining 01 or 10. I.e, If you measure the first qubit and it comes out to be in the 0 state, then we can be positive that the second state will also be a 0 state. These measurement outcomes are correlated and the qubits are entangled.

Entangled qubits are correlated such that they cannot be described independently from each other. Meaning that if an operation happens to a qubit in an entangled pair, it also affects the state of the other qubit.

Many-qubit system:

We follow the exact same patterns explored in the two-qubit case to build many-qubit quantum states from smaller states. Such states are built by forming tensor products of smaller states.

In many qubit systems, there is often a need to allocate and deallocate qubits that serve as a temporary memory for the quantum computer. Such a qubit is called auxiliary. By default, the qubit states can be initialised upon allocation. You can further assume that the qubit returns to the initialised state before deallocation. This assumption is important because if an auxiliary qubit becomes entangled with another qubit register when it becomes deallocated then the process of deallocation will damage the auxiliary qubit.

Although new gates are needed to be added to the gate to achieve universal quantum computing for two qubit quantum computers, there's no need to introduce new gates in the multi-qubit case.

Spanning sets, linear dependence and bases:

A linear combination of some collection vectors in some vector space over a field f is defined as an arbitrary sum of these vectors. If we have a set of vectors that spans a space, then any other vector in the vector space can be written as a linear combination of these vectors.

A set of vectors is linearly independent if there is no vector in the set that can be written as a linear combination of all the other vectors. A set of vectors are said to be linearly dependent if there exists corresponding coefficients for each vector such that:

$$b_1|v_1\rangle + b_2|v_2\rangle + \ldots + b_n|v_n\rangle = \sum_i b_i|v_i\rangle = 0,$$

Where at least one of the coefficients is non-zero

Basis is simply a linearly independent spanning set. In this context, the basis of a vector space is the minimum possible set that spans the entire space. The size of the basis sets the dimensions of vector space. They are more important because they allow us to reduce the vector spaces and express them in terms of only a few vectors. We can come to certain conclusions about our basis set that we can generalise to the entire vector space, because we know that every vector in the space can be written in terms of the basis vectors.

In quantum computing, one of the bases we often encounter is $|0\rangle$, $|1\rangle$. We can often write any other qubit state as a linear combination of these basis vectors. For instance, the linear combination

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

represents a superposition between $|0\rangle$ and $|1\rangle$ basis states, with equal probability of measuring the state to be in either one of the basis vectors.

Hilbert Spaces, Orthonormality and Inner Product:

Hilbert spaces are one of the most important constructs of quantum mechanics and quantum computing. A Hilbert Space can be visualised as the state space where all the quantum vectors “live”. The difference between a Hilbert space and a random vector space is that Hilbert spaces are equipped with an inner product, which is an operation that can be performed between two vectors, returning a scalar.

In quantum mechanics and computing the inner product returns the amount by which the first vector lies along the second vector. From this, the probabilities of measurement in different quantum states can be calculated. The inner product between two vectors in a Hilbert space looks something like:

$$\langle a|b\rangle = \begin{pmatrix} a_1^* & a_2^* & \dots & a_n^* \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ . \\ . \\ b_n \end{pmatrix} = a_1^* b_1 + a_2^* b_2 + \dots + a_n^* b_n$$

One of the most important conditions for a Hilbert space representing a quantum system is that the inner product of a vector with itself equals 1. This is the so-called normalisation condition, which states that the length of the vector squared must equal 1. The consequence of this is that the length of a vector in a particular direction is representative of the “probability amplitude” of the quantum systems with regards to measurement in that particular state.

Unitary vectors are important in quantum computation because they preserve the inner product. No matter how many times a vector is transformed under a sequence of unitary matrices, the normalisation condition still holds true.

Advantages of quantum computing:

- Quantum computing offers an innovative technique to solve some of the most complicated optimization issues while providing a significant performance advantage over traditional methods. This is visible in QC approaches such as Grover's search algorithm for unstructured databases. Quantum computers compute by inducing quantum speedups whose scaling greatly outstrips that of the most powerful classical computers. The main applications of QC may be found in optimization, machine learning, encryption, and quantum chemistry.

- Supercomputers are used by scientists and engineers to solve challenging issues. These are massive traditional computers, generally with thousands of traditional CPU and GPU cores. However, even supercomputers have difficulty solving some types of issues.
- Complex issues have many variables that interact in intricate ways. Because of all the various electrons interacting with one another, modelling the behaviour of individual atoms in a molecule is a difficult challenge. Choosing the best routes for a few hundred tankers in a worldwide transportation network is also difficult.

- As it turns out, optimising complex systems is one of the best use cases for quantum computers, and one of the areas where quantum computers could provide an advantage first.

The following here are a few examples of algorithms that make the best use of optimisation techniques:

Deutsch-Jozsa algorithm:

The Deutsch-Jozsa method is an algorithm which checks if a given function returns the same value all the time or if it will return different values (balanced function) at least 50 % of the time. This algorithm operates on the superposition of all the input values simultaneously. It can be performed by both Classical computers as well as quantum computers. The plus point of using a quantum computer for this algorithm is its time complexity, i.e, the time complexity for this algorithm is 2^{n-2}, where n is the number of inputs, while the time complexity of this algorithm is 1. The code for this algorithm is presented below:

```
from qiskit import QuantumCircuit, Aer, execute
import random

# Define the Deutsch-Jozsa circuit
def deutsch_jozsa(n, oracle):
    # Create a quantum circuit with n qubits and n+1 classical bits
    circuit = QuantumCircuit(n+1, n)

    # Apply a Hadamard gate to all qubits, which flips them to the superposition state
    circuit.h(range(n+1))

    # Apply the oracle function
    for i in range(n):
        if oracle[i] == '1':
            circuit.cx(i, n)

    # Apply a Hadamard gate to the first n qubits
    circuit.h(range(n))

    # Measure the first n qubits
    circuit.measure(range(n), range(n))

    return circuit

# Define the oracle function
def oracle(n, type):
    if type == 1:  # constant function returning 0
        return '00000000'
    elif type == 2:  # constant function returning 1
        return '11111111'
    elif type == 3:  # balanced function
        s = bin(random.getrandbits(n))[2:].zfill(n)
        return s + s

# Choose the number of qubits and the oracle type
n = 4
type = 3

# Generate the oracle function
oracle_string = oracle(n, type)
```

```
# Construct the Deutsch-Jozsa circuit
circuit = deutsch_jozsa(n, oracle_string)

# Run the circuit on a simulator
simulator = Aer.get_backend('qasm_simulator')
result = execute(circuit, simulator, shots=5).result()    # running 5 tests

# Print the result
print(result.get_counts(circuit))
```

Grover's algorithm:

In classical computing, if you want to search a database with N items, the best you can do is perform N/2 queries on average, which is known as linear search algorithm.

By contrast, Grover's algorithm can find the solution in only O(sqrt(n)) queries. The algorithm works by preparing a quantum state that represents all possible solutions to the problem and then applying a series of quantum gates that amplify the amplitude of the desired solution.

Here's the code implementation of the same algorithm:

```
[10:51, 26/04/2023] Darth Sid: from qiskit import QuantumCircuit, Aer, execute
import random

# Define the Deutsch-Jozsa circuit
def deutsch_jozsa(n, oracle):
    # Create a quantum circuit with n qubits and n+1 classical bits
    circuit = QuantumCircuit(n+1, n)

    # Apply a Hadamard gate to all qubits, which flips them to the superposition state
    circuit.h(range(n+1))

    # Apply the oracle function
    for i in range(n):
        if oracle[i] == '1':
            circuit.cx(i, n)

    # Apply a Hadamard gate to the first n qubits
```

```
    circuit.h(range(n))

    # Measure the first n qubits
    circuit.measure(range(n), range(n))

    return circuit

# Define the oracle function
def oracle(n, type):
    if type == 1:  # constant function returning 0
        …
[10:51, 26/04/2023] Darth Sid: DJ algo
[10:51, 26/04/2023] Darth Sid: from qiskit import QuantumCircuit, Aer, execute

# Define the number of items in the database
N = 8

# Create a quantum circuit with N qubits
circ = QuantumCircuit(N, N)

# Apply a Hadamard gate to all qubits to create a superposition of all possible states
circ.h(range(N))

# Define the target item to search for (in this case, item 5)
target = 5

# Apply the Grover diffusion operator multiple times
num_iterations = int(round(N**(1/2)))
for i in range(num_iterations):
    # Apply a phase inversion gate to the target item
    circ.x(target)
    circ.h(target)
    # circ.mct(list(range(N-1)), target)
    circ.mct(list(range(N))[:target] + list(range(N))[target+1:], target)
    circ.h(target)
    circ.x(target)

    # Apply a phase inversion gate to all items except the target item
    circ.h(range(N))
    circ.x(range(N))
```

```
    circ.mct(list(range(N-1)), N-1)
    circ.x(range(N))
    circ.h(range(N))

# Measure the qubits to extract the result
circ.measure(range(N), range(N))

# Run the circuit on a simulator
backend = Aer.get_backend('qasm_simulator')
job = execute(circ, backend)
result = job.result()

# Print the measurement results
counts = result.get_counts()
print(counts)
```

Conclusion:

Quantum computing is still in its infancy stage. The quantum computers that are currently available have specific requirements regarding hardware and cooling temperature conditions. The applications of this domain looks highly promising and several large companies, governments and academic institutions continue to invest millions into the research and development of quantum computing, it will not be long before quantum computing becomes a ubiquitous computational technology.

In conclusion, quantum computing systems require complex mathematical algorithms to simulate and manipulate the behaviour of qubits and linear algebra provides the mathematical framework to facilitate these computations. Specifically, linear algebra concepts such as linear transformations, eigenvectors and matrices are fundamental in the design and analysis of quantum algorithms. Furthermore the use of linear algebraic techniques such as quantum state tomography and quantum error correction, enables researchers to measure and control the state of qubits in a quantum system. As quantum computing technology continues to evolve, linear algebra will remain an essential tool for the advancement and implementation of quantum computing applications.

References:

1. https://www.ibm.com/topics/quantum-computing
2. https://qiskit.org/textbook/ch-appendix/linear_algebra.html#Spanning-Sets,-Linear-Dependence,-and-Bases
3. https://learn.microsoft.com/en-us/azure/quantum/overview-algebra-for-quantum-computing
4. https://learn.microsoft.com/en-us/azure/quantum/concepts-vectors-and-matrices
5. https://learn.microsoft.com/en-us/azure/quantum/concepts-advanced-matrices
6. https://learn.microsoft.com/en-us/azure/quantum/concepts-the-qubit
7. https://learn.microsoft.com/en-us/azure/quantum/concepts-multiple-qubits
8. https://en.wikipedia.org/wiki/Quantum_computing
9. https://www.classcentral.com/course/matrix-505#:~:text=Linear%20algebra%20provides%20concepts%20that,information%20retrieval%20and%20web%20search.
10. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-quantum-computing/#:~:text=Quantum%20computing%20can%20be%20used,determine%20the%20best%20possible%20answer.