



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Σχολή Τεχνολογίας

Τμήμα Ψηφιακών Συστημάτων

Παραγωγή Μουσικής Με Κβαντι- κούς Αλγόριθμους

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Διονυσία Γεωργακοπούλου (3119052)

Επιβλέπων: Ηλίας Κ. Σάββας, Καθηγητής

ΛΑΡΙΣΑ, Ιανουάριος, 2024



UNIVERSITY OF THESSALY

School of Technology

Digital Systems Department

Quantum Algorhythmic Music Composition

BACHELOR THESIS

Dionysia Georgakopoulou(3119052)

Supervisor: *Ilias K. Savvas, professor*

LARISSA, January, 2024

Υπεύθυνη Δήλωση περί Ακαδημαϊκής Δεοντολογίας
και Πνευματικών Δικαιωμάτων

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα πτυχιακή εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον, και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Η Δηλούσα

Γεωργακοπούλου Διονυσία

22/01/2024

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων

Ηλίας Κ. Σάββας

Καθηγητής

Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας

Μέλος

Κωνσταντίνος Χαϊκάλης

Αναπληρωτής Καθηγητής

Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας

Μέλος

Απόστολος Ξενάκης

Επίκουρος Καθηγητής

Τμήμα Ψηφιακών Συστημάτων, Πανεπιστήμιο Θεσσαλίας

Ημερομηνία έγκρισης: 29/01/2024

Περίληψη

Η παρούσα πτυχιακή πραγματεύεται την παραγωγή μουσικής με χρήση κβαντικών αλγορίθμων. Ανέκαθεν, η επιστήμη έπαιζε σημαντικό ρόλο στην σύνθεση της μουσικής, για παράδειγμα στα έργα του συνθέτη Mozart γίνεται αντιληπτή η επιρροή που είχε από τους νόμους του φυσικού Isaac Newton. Σήμερα, η Κβαντική Υπολογιστική αναπτύσσεται όλο και περισσότερο όπου μέσα από μια πληθώρα κβαντικών αλγορίθμων αντικατοπτρίζεται η μουσική. Κβαντικοί αλγόριθμοι όπως ο αλγόριθμος του Groover και ο Random (Quantum) Walk έχουν υλοποιηθεί σε IBM υπολογιστή οδηγώντας στην παραγωγή μιας μελωδίας.

Σκοπός της πτυχιακής είναι αρχικά η εμβάθυνση στην κβαντική υπολογιστική και στους κβαντικούς αλγόριθμους και στην συνέχεια η μελέτη και υλοποίηση στο μέτρο του δυνατού των αλγορίθμων αλλά και της μουσικής που μπορεί να προκύψει μέσα από αυτούς.

Πιο αναλυτικά, η παρούσα πτυχιακή αποτελείται από 5 κεφάλαια.

Το πρώτο κεφάλαιο είναι μια εισαγωγή στην κβαντική μηχανική, στην κβαντική υπολογιστική και στις εφαρμογές τις κβαντικής υπολογιστικής σε τομείς όπως η μουσική, η κρυπτογραφία και τα παιχνίδια. Υστέρα στο δεύτερο κεφάλαιο πραγματοποιείται περιγραφή του μαθηματικού υπόβαθρου που είναι απαραίτητο για την κβαντική υπολογιστική, δηλαδή πράξεις ανάμεσα σε πίνακες ταυστικό γινόμενο και μιγαδικοί αριθμοί. Στο ίδιο κεφάλαιο περιγράφεται και η γλώσσα προγραμματισμού και το λογισμικό που χρησιμοποιήθηκαν στην παρούσα πτυχιακή. Οπου η γλώσσα προγραμματισμού δεν θα μπορούσε να είναι άλλη από την γλώσσα Python. Ενώ το λογισμικό που χρησιμοποιήθηκε είναι το Jupyter Notebook για την πλήρη κατανόηση της και του μαθηματικού υπόβαθρου αλλά και της γλώσσας Python πραγματοποιήθηκαν στο Jupyter Notebook προγράμματα που εκτελούν τους απαραίτητους μαθηματικούς υπολογισμούς για την κβαντική. Η περιγραφή των εννοιών της κβαντικής υπολογιστικής ξεκινάει με το τρίτο κεφάλαιο το οποίο περιέχει τα qubits τα οποία αποθηκεύουν τις κβαντικές πληροφορίες. Το ίδιο κεφάλαιο συνεχίζεται με την τους κβαντικούς καταχωρητές οι οποίοι περιέχουν ένα ή περισσότερα qubits και στην συνέχεια περιγράφονται οι κβαντικές πύλες οι οποίες καθορίζουν την συμπεριφορά που θα έχουν τα qubits. Το τρίτο κεφάλαιο ολοκληρώνεται με τα κβαντικά κυκλώματα τα οποία αναπαριστούν των αριθμό των qubits που υπάρχουν και τις πύλες που εφαρμόζονται σε αυτά. Πριν την εισαγωγή στην παραγωγή μουσικής με κβαντικούς αλγόριθμους απαραίτητη είναι η αλγοριθμική μουσική. Η αλγοριθμική μουσική αποτελεί το τέταρτο κεφάλαιο της πτυχιακής το οποίο πέρα από την αναφορά στο τι είναι η αλγοριθμική μουσική και της εφαρμογές της περιέχει και μια εισαγωγή στην μουσική και πώς η μουσική σε συνδυασμό με τον κλάδο της θετικής επιστήμης οδήγησε σε ένα πρωτοπόρο είδος μουσικής. Στο ίδιο κεφάλαιο περιέχεται ένα λογισμικό παραγωγής μουσικής το οποίο διαφέρει από όλα τα υπόλοιπα καθώς στο παρών λογισμικό η μουσική προκύπτει μέσα από κώδικα γραμμένο στην γλώσσα Ruby το λογισμικό αυτό είναι η Sonic Pi και μαζί με αυτό ακολουθούν και παραδείγματα γραμμένα σε Sonic Pi.

Η κβαντική ακουστική όπως και η κβαντική μουσική είναι κλάδοι μέσα από τους οποίους παράγεται η μουσική από κβαντικούς αλγόριθμους. Στο πέμπτο κεφάλαιο ακολουθεί η εισαγωγή σε ορολογίες όπως συμπαντική μουσική, κβαντική ακουστική και κβαντική μουσική και αναλύεται η διαδικασία με την οποία παράγεται μουσική μέσα από κβαντικούς αλγόριθμους.

Πριν την παραγωγή μουσικής από κβαντικούς αλγόριθμους υπάρχει παραγωγή μουσικής μέσα από τα Bell States. Υστερα μέσα από τις Μαρκοβιανές αλυσίδες την χρήση αλγορίθμων όπως ο Quantum Walk, ο Grover, και ο Bernstein-Vazirani παράγεται μουσική. Στο έκτο και τελευταίο κεφάλαιο ακολουθούν συμπεράσματα που προκύπτουν από την παραγωγή μουσικής μέσα από κβαντικούς αλγόριθμους.

Abstract

The according thesis is about music composition with the use of quantum algorithms. Science has always played an important role in music production. Music composers such as Mozart were affected from physicist such as Issac Newton. These days quantum computing is developing more and more where music is created through quantum algorithms. Quantum algorithms such as Grover's Algorithm and Random(Quantum)Walk have been implemented on an IBM computer leading to the production of a melody. The thesis consists of five chapters. The first chapter is an introduction to quantum mechanics, quantum computing and the applications of quantum computing in music, cryptography and video games. In the second chapter of this thesis, follows a description of the mathematical background necessary for quantum computing is carried out, i.e. operations between matrices, tensor product and complex numbers. In the same chapter is described, the programming language Python and the software Jupyter Notebook. Then for a complete understanding both mathematical background and the Python language, are executed programs about the necessary mathematical calculations for quantum computing. The description of the notions of quantum computing begins with the third chapter which contains the qubits which store the quantum information. The same chapter continues with the quantum registers which contain one or more qubits and then describes the quantum gates which determine the behavior of the qubits. The third chapter concludes with the quantum circuits which represent the number of qubits that exist, and the gates applied to them. Algorithmic music is necessary before the introduction to quantum algorithmic music composition. Algorithmic music is the fourth chapter of the thesis which expects the reference to algorithmic music and the applications of algorithmic music, contains an introduction to music and how music in combination with science creates a new music gender. In the same chapter there is a different from all the other DAW and the name of this software in Sonic Pi. The main difference between Sonic Pi and all the others DAW is that Sonic Pi use the programming language Ruby to create music. In chapter four there are some examples in Sonic Pi. Quantum algorithmic music composition is offshoot of quantum acoustics and quantum music. In the fifth chapter follows terminologies such as universal music, quantum acoustics and quantum music and analyzes the process of creating music throw quantum algorithms.

But before the production of music through quantum algorithms there is the production of music through Bell States. Then through Markovian chains and algorithms such as Quantum Walk, Grover, and Bernstein-Vazirani music is produced. In the sixth and last chapter, conclusions are drawn from the production of music through quantum algorithms

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής θα ήθελα να ευχαριστήσω πρώτα από όλους τον επιβλέπων καθηγητή μου κύριο Σάββα Ηλία για την στήριξη και την βοήθεια που μου πρόσφερε κατά την διάρκεια της εργασίας. Υστέρα θα ήθελα να ευχαριστήσω τους γονείς μου τον αδερφό μου, την θεία μου Χριστίνα και τους φίλους που και αυτοί με στηρίζαν μέχρι την ολοκλήρωση της πτυχιακής μου εργασίας.

Γεωργακοπούλου Διονυσία

22/01/2024

Περιεχόμενα

ΠΕΡΙΛΗΨΗ	IV
ABSTRACT.....	VII
ΕΥΧΑΡΙΣΤΙΕΣ.....	IX
ΠΕΡΙΕΧΟΜΕΝΑ.....	XI
1 Εισαγωγή στην κβαντομηχανική	1
2 Μαθηματικό υπόβαθρο και Python	17
2.1 Εισαγωγή στους πίνακες.....	17
2.2 Μιγαδικοί Αριθμοί.....	20
2.2.1 Πράξεις με μιγαδικούς αριθμούς.....	21
2.2.2 Μιγαδικό επίπεδο.....	23
2.2.2.1 Καρτεσιανές Συντεταγμένες	23
2.2.2.2 Πολικές Συντεταγμένες.....	24
2.2.2.3 Σφαιρικές Συντεταγμένες.....	25
2.3.Εισαγωγή στην Python.....	26
2.3.1.Εφαρμογές και προγράμματα στην Python	26
2.3.1.1 Πολλαπλασιασμός πινάκων.....	28
2.3.1.2 Τανυστικό γινόμενο.....	29
2.3.1.3 Μένου για ανάμεσα σε δύο μιγαδικούς αριθμούς	31
2.3.1.4 Μενού για μετατροπή συντεταγμένων	36
3.0 Κβαντική υπολογιστική	39
3.1 Qubits.....	39
3.1 Bracket.....	43
3.2 Κβαντικοί καταχωρητές.....	45
3.3 Κβαντικές Πύλες.....	49
3.3.1 Πύλη Αδράνειας.....	49
3.3.2 Πύλη Handamard.....	50
3.3.3 Πύλες Pauli	50

3.3.3.1 Pauli -X(Πύλη X)	50
3.3.3.2 Pauli -Z(Πύλη Z)	51
3.3.3.3 Pauli -Y(Πύλη Y)	51
3.3.4 cX(controlled -X)	51
3.3.5 cZ(controlled -Z)	52
3.3.6 cH(controlled -Handamard)	53
3.3.7 Πύλη Swap	54
3.3.8 Toffoli Πύλη(CCNOT πύλη)	54
3.3.9 Fredkin Πύλη(CSWAP Πύλη)	55
3.3.10 Μενού πυλών	56
3.3.11 Bell States	75
3.3.12 Παράλληλες πύλες	82
4.0 Αλγοριθμική μουσική	83
4.1 Εισαγωγή στην μουσική	83
4.1.1 Μουσική κρυπτογραφία	84
4.1.2 Η άνθηση της αλγοριθμικής μουσικής	87
4.1.3 Η εξέλιξη της αλγοριθμικής μουσικής	88
4.1.3.1 Σκοπός της αλγοριθμικής μουσικής	88
4.1.3.2 Οφέλη της αλγοριθμικής μουσικής	89
4.1.3.3 Εφαρμογές της αλγοριθμικής μουσικής	89
4.2 Sonic Pi	90
4.2.1 Εισαγωγή στο Live Coding και στο Super Collider	91
4.2.2 Παραδείγματα χρήσης της Sonic Pi	92
4.2.2.1 Το ζάρι του Mozart	92
4.2.2.2 Μουσικό Bit Sequence	95
4.2.2.3 Mp3 Player	97
5.0 Παραγωγή μουσικής με κβαντικούς αλγόριθμους	98
5.1 Συμπαντική μουσική	102
5.2 Κβαντική ακουστική και κβαντική μουσική	103
5.2.1 Τι είναι κβαντική ακουστική	103
5.2.1.1 Τι είναι φωνόνια	103
5.2.1.2 Διαφορές φωτονίων με φωνονίων	104
5.2.2 Κβαντική μουσική	104

5.3 Κβαντικοί αλγόριθμοι(Quantum Algo-rhythms).....	105
5.3.1 Παραγωγή μουσικής από Bell State (Bell State of the art).....	105
5.3.2 Μαρκοβιανές Αλυσίδες.....	110
5.3.3 Κλασικός και κβαντικός περίπατος	118
5.3.3.1.Κβαντικός περίπατος	119
5.4 Αλγόριθμος του Grover	135
5.5 Αλγόριθμος Bernstein–Vazirani	148
6.0 Συμπεράσματα	152
7.0 Βιβλιογραφία	154

1 Εισαγωγή στην Κβαντομηχανική

Ο σκοπός δεν είναι τόσο να δούμε αυτό που κανείς δεν έχει δει ακόμα, αλλά να σκεφτούμε αυτό που κανείς δεν έχει σκεφτεί ακόμα για αυτό που όλοι βλέπουν.

Η κλασική μηχανική εξετάζει και αναλύει φαινόμενα τα οποία συμβαίνουν στον μακρόκοσμο και τα οποία είναι σε θέση να αντιληφθεί ο καθημερινός άνθρωπος.

Όμως εκτός από τον μακρόκοσμο υφίσταται και ο μικρόκοσμος. Δηλαδή ο κόσμος των πολύ μικρών σωματιδίων(ατόμων και υποατόμων όπως πρωτόνια νετρόνια ηλεκτρόνια) ,ο οποίος περιέχει φαινόμενα τα οποία η κλασική φυσική δεν μπορεί να αντιληφθεί. Για την μελέτη των φαινομένων του μακρόκοσμου χρησιμοποιείται μια θεμελιώδης φυσική η οποία είναι γνωστή ως κβαντική μηχανική.

Η κβαντική μηχανική έγινε αποδεκτή μέσα από ένα σύνολο πειραμάτων που εκτέλεσαν μια ομάδα φυσικών οδηγώντας όπως είναι αναμενόμενο σε αποτελέσματα τα οποία διέφεραν από αυτά της κλασικής φυσικής. Για αυτό τον λόγο μέχρι και επιστήμονες που συνέβαλαν σε πειράματα για να αποδειχθούν τα κβαντικά φαινόμενα εν τέλει αμφέβαλαν για την ύπαρξη της κβαντικής θεωρίας. Ανάμεσα σε αυτούς τους επιστήμονες ήταν και ο γνωστός μαθηματικός Αλμπερτ Αϊνστάιν ο οποίος χρησιμοποίησε την φράση «Ο θεός δεν παίζει ζάρια με το σύμπαν» ώστε να δείξει πως η κβαντική θεωρία δεν υφίσταται στον μακρόκοσμο καθώς βασίζεται σε πιθανότητες .

Ωστόσο η ιστορία της κβαντικής θεωρίας ξεκινάει το 1900 με την θεωρία του Γερμανού φυσικού Max Planck πως η ενέργεια που παράγεται από ένα μελανό σώμα είναι κβαντισμένη.

Από τους νόμους της κλασικής φυσικής είναι γνωστό πως ως ακτινοβολία που έχει ένα μαύρο σωματίδιο ορίζεται το φάσμα του φωτός που παράγεται από ένα θερμό αντικείμενο.

Αρχικά ο φυσικός παρατήρησε πώς όταν ένα μαύρο σώμα θερμαίνεται παράγει ηλεκτρομαγνητική ακτινοβολία η οποία είναι ένα φάσμα φωτός, δηλαδή ορατή στο ανθρώπινο μάτι.

Η άνοδος της θερμοκρασίας του σώματος θα έχει σαν αποτέλεσμα την άνοδο της ακτινοβολίας. Οπότε σε περίπτωση που η θερμοκρασία του σώματος είναι αρκετά υψηλή τότε η ακτινοβολία είναι άπειρη. Για την επίλυση αυτού του προβλήματος ο Planck χρησιμοποίησε τις εξισώσεις του φυσικού Boltzmann οι οποίες αναπαριστούν την συμπεριφορά των αερίων σωμάτων όπου κάθε ένα διασπάται σε μικρότερα σωματίδια.

Μέσα από αυτές οδηγήθηκε στην δημιουργία της παρακάτω εξίσωσής:

$$E=h\cdot\nu$$

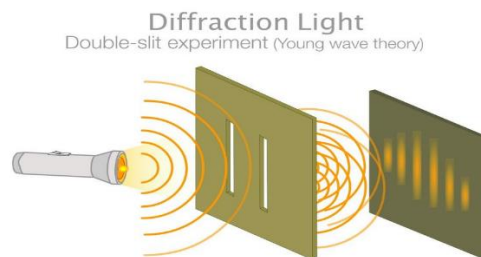
- (E) Αναπαριστά την ενέργεια του κβαντισμένου σωματιδίου
- (h) Αναπαριστά την σταθερά Planck δηλαδή μια φυσική σταθερά της κβαντομηχανικής με τιμή 6.626×10^{-34} joule
- (ν) Αναπαριστά την συχνότητα της ακτινοβολίας

όπου η μικρότερη ποσότητα ενέργειας της ηλεκτρομαγνητικής ακτινοβολίας που θα εκπέμψει ή θα απορροφήσει ένα σώμα αποθηκεύεται σε μικρά πακέτα τα κβάντα. Οπότε η ενέργεια είναι κβαντισμένη δηλαδή οι τιμές της ενέργειας είναι διακριτές (και όχι συνεχείς) ανάλογα με την ακτινοβολία που εκπέμπεται από κάθε σωματίδιο. Το συγκεκριμένο συμπέρασμα δημιούργησε αντιδράσεις καθώς μέχρι τότε το φως είχε την μορφή ενός συνεχούς κύματος. Ως κβάντο μπορεί να θεωρηθεί και ένα φωτόνιο ενώ η περιγραφή του μεγέθους των κβάντων γίνεται με την σταθερά (h) γνωστή και ως σταθερά του Planck.

Το 1923 ο Γάλλος φυσικός κατά την διάρκεια της διατριβής του ισχυρίστηκε ότι όπως το φως μπορεί να πάρει μορφή σωματιδίου (φωτόνιο) και μορφή κύματος έτσι και τα ηλεκτρόνια μπορούν να παρουσιάσουν και αυτά μια κυματοειδή συμπεριφορά. Σκοπός της διατριβής ήταν να επιβεβαιωθεί η διπλή φύση ενός ηλεκτρονίου.

Για την απόδειξή του ισχυρισμού του χρησιμοποίησε το πείραμα της διπλής σχισμής, σύμφωνα με το οποίο τα ηλεκτρόνια τοποθετούνται σε μια σανίδα η οποία έχει δύο σχισμές και πίσω από αυτή την σανίδα βρίσκεται μια οθόνη η οποία ανάβει κάθε φορά που ένα ηλεκτρόνιο χτυπάει πάνω σε αυτή.

Οπότε κάθε φορά που εκτελείται το εξής πείραμα δημιουργούνται παρεμβολές στην οθόνη οι οποίες οδηγούν στο συμπέρασμα πως τα ηλεκτρόνια είναι κύματα τα οποία παρεμβάλλονται μεταξύ τους. Το πείραμα της διπλής σχισμής αναπαρίσταται στην εικόνα 0



Εικόνα 0: Το πείραμα της διπλής σχισμής όπου τα ηλεκτρόνια παράγουν κυματοειδής μορφή(What Is Quantum Mechanics? Quantum Physics Defined, Explained | Live Science)

Ανέπτυξε την εξής εξίσωση η οποία είναι γνωστή και ως εξίσωση de Broglie: $\lambda = \frac{p}{h}$

Όπου:

- (λ) το μήκος της κυματομορφή του σωματιδίου,
- (p) δηλώνει την ορμή δηλαδή την ποσότητα κίνησης των σωματιδίων (ηλεκτρονίων). Και από τον παραπάνω τύπο συμπεραίνεται πως η ορμή είναι αντίστροφος ανάλογη από το μήκος κύματος , οπότε τα σωματίδια με μεγάλη ορμή έχουν μικρό μήκος κύματος
- (h) η σταθερά του Planck η οποία όπως αναφέρθηκε και προηγουμένως

Περιέχει τα κβάντα. Και δείχνει πως ένα ηλεκτρόνιο μπορεί να έχει και κυματοειδής μορφή πέρα από μορφή σωματιδίου. Οπότε η κυματομορφή ενός ηλεκτρονίου αναπαριστά μια κβαντική κατάσταση [3]

Το 1926 ο Erwin Schrödinger βασισμένος στην εξίσωση de Broglie δημιούργησε μια μαθηματική εξίσωση μέσω της οποίας είναι ορατή η συμπεριφορά του ηλεκτρονίου άρα και η θέση που βρίσκεται μέσα στο άτομο.

Δηλαδή η εξίσωση του Schrödinger δείχνει την πιθανότητα εμφάνισης του ηλεκτρονίου (το οποίο αρχικά μπορεί να βρίσκεται σε οποιαδήποτε θέση) σε μια συγκεκριμένη θέση του χώρου μια συγκεκριμένη χρονική στιγμή.

Στην προσέγγισή της θέσης του ηλεκτρονίου συνέβαλε και ο φυσικός Werner Heisenberg μέσω της θεωρίας του η οποία έμεινε γνωστή ως αρχή απροσδιοριστίας του Heisenberg. Η αρχή απροσδιοριστίας του Heisenberg λέει πως η θέση και η ενέργεια ενός ηλεκτρονίου είναι αδύνατο να είναι γνωστές ταυτόχρονα για το ηλεκτρόνιο. Οπότε βασιζόμενοι στην αρχή απροσδιοριστίας του Heisenberg και στο μοντέλο του Schrödinger οι χημικοί θεώρησαν την ενέργεια δεδομένη καθώς είναι απαραίτητη για την πρόβλεψη της χημικής αντιδραστικότητας ενός ατόμου οπότε η εύρεση της θέσης του ηλεκτρονίου είναι εφικτή από την στιγμή που η ενέργεια του είναι γνωστή.

Για να αποδειχθεί η λογική της εξίσωσης του Schrödinger χρησιμοποιήθηκε το εξής παράδειγμα έστω πως υπάρχει ένα κουτί με δηλητήριο μέσα στο οποίο βρίσκεται μια γάτα. Το κουτί είναι κλειστό, ένας εξωτερικός παρατηρητής θα μάθει την κατάσταση της γάτας (δηλαδή αν είναι ζωντανή ή είναι νεκρή) μόνο όταν ανοίξει το κουτί. Οπότε η γάτα όταν βρίσκεται στο κουτί είναι και ζωντανή και νεκρή ταυτόχρονα. Το φαινόμενο αυτό ονομάστηκε υπέρθεση και είναι μια ιδιότητα που έχουν τα κβαντικά σωματίδια η οποία τους επιτρέπει να βρίσκονται ταυτόχρονα σε 2 καταστάσεις. Η γάτα του Schrödinger αναπαρίσταται στην εικόνα 1



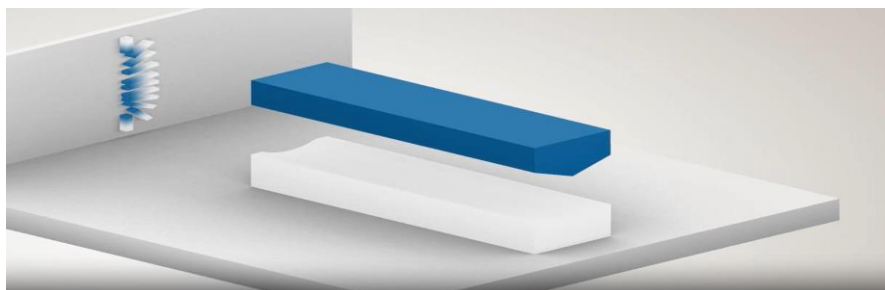
Εικόνα 1: Γάτα του Schrödinger(<https://www.motherjones.com/kevin-drum/2018/09/schrodingers-cat-is-alive-one-twelfth-of-the-time/>)

Αξίζει να σημειωθεί πως ο Γερμανός φυσικός Heisenberg πέρα από την αρχή της αβεβαιότητας δημιούργησε ένα πίνακα γνωστό και ως Μηχανική Μήτρα του Heisenberg. Η Μηχανική μήτρα του Heisenberg είναι ένας πίνακας που περιέχει όλες τις κβαντικές καταστάσεις όταν υποστούν κάποια αλληλεπίδραση άλλα περιέχει και τις καταστάσεις πριν υποστούν την αλληλεπίδραση και αναπαριστάτε στην εικόνα 2. Στην οποία εικόνα οι στήλες m αναπαριστούν τις κβαντικές καταστάσεις όταν αυτές δεν έχουν υποστεί κάποια μεταβολή, ενώ οι σειρές n αναπαριστούν τις καταστάσεις όταν αυτές μεταβληθούν.

$$f_{m,n} \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} & f_{15} & \text{etc.} \\ f_{21} & f_{22} & f_{23} & f_{24} & f_{25} & \text{etc.} \\ f_{31} & f_{32} & f_{33} & f_{34} & f_{35} & \text{etc.} \\ f_{41} & f_{42} & f_{43} & f_{44} & f_{45} & \text{etc.} \\ f_{51} & f_{52} & f_{53} & f_{54} & f_{55} & \text{etc.} \\ \text{etc.} & \text{etc.} & \text{etc.} & \text{etc.} & \text{etc.} & \text{etc.} \end{bmatrix}$$

Εικόνα2:Αναπαράσταση Μηχανικής Μήτρας του Heisenberg (Heisenberg's Matrix Mechanics (mpoweruk.com))

Ένα από τα σημαντικότερα κβαντικά φαινόμενα το οποίο ερεύνησαν το 1922 οι δύο Γερμανοί φυσικοί Otto Stern και Walther Gerlach είναι το σπιν των ηλεκτρονίων. Για τον σκοπό της έρευνας τους δημιούργησαν ένα μαγνητικό πεδίο το οποίο αποτελείται από δύο μαγνήτες ο ένας παράλληλα από τον άλλον ,κάθε μαγνήτης αναπαριστά τον βόρειο και τον νότιο πόλο. Κάθε εξωτερικός μαγνήτης δέχεται μια δύναμη από το πεδίο και φτάνει στην απέναντι επιφάνεια κάθετα σε οποιοδήποτε σημείο της επιφάνειας.

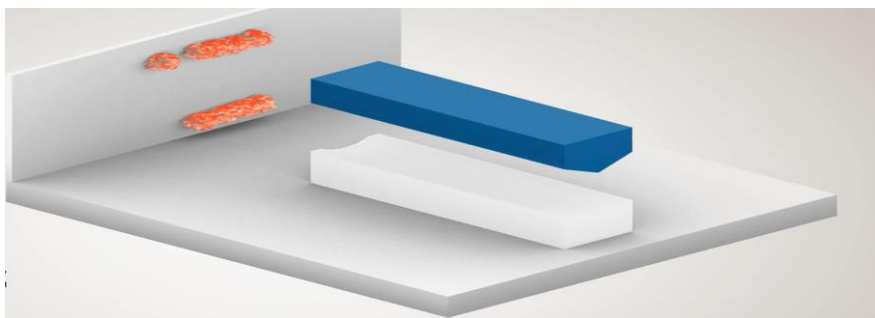


Εικόνα3: Αναπαράσταση του κλασικού μαγνητικού πεδίου, ο μπλε μαγνήτης αναπαριστά τον Βόρειο πόλο ενώ ο άσπρος των νότιο πόλο (upload.wikimedia.org/wikipedia/commons/9/9e/Quantum_spin_and_the_Stern-Gerlach_experiment.ogv)

Στην συνέχεια οι 2 φυσικοί παρατήρησαν πως αν στο μαγνητικό πεδίο εισέλθουν ηλεκτρόνια κάθε ηλεκτρόνιο θα φτάνει στην απέναντι επιφάνεια πάντα απέναντι από τους μαγνήτες (δηλαδή προς τα πάνω ή προς τα κάτω ποτέ ενδιάμεσα). Όπως φαίνεται στην εικόνα 3

Δηλαδή σε αντίθεση με τους ραβδωτούς μαγνήτες τα ηλεκτρόνια είχαν μόνο 2 πιθανούς προσανατολισμούς η προς τα πάνω ή προς τα κάτω. Το φαινόμενο αυτό ονομάστηκε σπιν(spin) ηλεκτρονίων και αναπαριστάτε στην εικόνα 4 και όταν το ηλεκτρόνιο βρίσκεται προς τα κάτω αναπαριστάτε με το εξής βέλος (↓) και αναφερόμαστε στον όρο spin down.

Ενώ όταν είναι προς τα πάνω αναφερόμαστε στον όρο spin up και αναπαρίσταται με το εξής βέλος(↑)



Εικόνα4:Σπιν ηλεκτρονίων

(upload.wikimedia.org/wikipedia/commons/9/9e/Quantum_spin_and_the_Stern-Gerlach_experiment.ogg)

Εκτός από την υπέρθεση μια άλλη ιδιότητα που έχουν τα κβαντικά σωματίδια είναι η διεμπλοκή η οποία εμφανίζεται στο Einstein–Podolsky–Rosen παράδοξο (ERP). Ως κβαντική διεμπλοκή ορίζεται το φαινόμενο το οποίο εφαρμόζεται σε 2 σωματίδια τα οποία συνδέονται με τέτοιο τρόπο ώστε η κατάσταση του ενός σωματιδίου να μεταβάλλει τις καταστάσεις και των άλλων σωματιδίων ακαριαία και ανεξάρτητα από την απόσταση.

Το παράδοξο ERP αναφέρεται σε 2 σωματίδια τα οποία βρίσκονται σε διεμπλοκή οπότε η μέτρηση της κατάστασης ενός σωματιδίου καθορίζει ακαριαία και την κατάσταση του άλλου.

Οι 3 φυσικοί υποστήριξαν πως κάτι τέτοιο δεν θα ήταν εφικτό και πως υπάρχουν μεταβλητές οι οποίες καθορίζουν τα αποτελέσματα των μετρήσεων.

Στα τέλη του 20^{ου} αιώνα και συγκεκριμένα στις αρχές του 1980 ερευνητές όπως ο Richard Feynman, ο Paul Benioff και ο David Deutsch ενσωμάτωσαν τις θεωρίες της κβαντικής μηχανικής στην επιστήμη των υπολογιστών με σκοπό δημιουργία μοντέλων με πολύ υψηλές υπολογιστικές δυνατότητες.

Ο κλάδος που δημιουργήθηκε ονομάστηκε κβαντική υπολογιστική και μελετάτε και εξελίσσεται μέχρι και σήμερα

Ο κόσμος είναι ένα δυναμικό χάος από πράγματα που δύσκολο να πιστέψεις
Richard Feynman

Συγκεκριμένα ο φυσικός Richard Feynman ήταν ο πρώτος που ισχυρίστηκε πως η εκτέλεση υπολογισμών με χρήση των κβαντικών φαινομένων θα μπορεί να ξεπεράσει τις δυνατότητες ενός κλασικού υπολογιστή.

Σκοπός του ήταν η δημιουργία ενός υπολογιστή που θα περιέχει τις αρχές της κβαντικής φυσικής και θα αναπαριστούσε δηλαδή ένα κβαντικό σύστημα το οποίο επιλύει πολύπλοκους συνδυασμούς.

Ένα πρόβλημα το οποίο θα μπορούσε να επιλύσει ένα κβαντικό σύστημα με βάση τον Feynman είναι το μόριο φαρμάκου το οποίο απαιτεί χιλιάδες συνδυασμούς που παίζουν σημαντικό ρόλο στην σύνθεση του, οι οποίοι με την βοήθεια μια κβαντικής προσομοίωσης θα βρεθούν πιο γρήγορα. Μέσα από αυτόν τον ισχυρισμό προέκυψε η πρώτη ιδιότητα ενός κβαντικού συστήματος και ονομάστηκε κβαντική υπεροχή (Quantum Supremacy), μέσω αυτής της ιδιότητας ένας κβαντικός αλγόριθμος που υλοποιεί πολύπλοκους συνδυασμούς μπορεί να εκτελεστεί από το κβαντικό σύστημα σε ένα πολύ μικρό χρονικό διάστημα. Σε αυτό το σημείο αξίζει να αναφερθεί πως μεγάλες εταιρίες όπως η Google η Dell και η Microsoft δαπανούν μεγάλο μέρος των χρημάτων τους ώστε να δημιουργήσουν ένα κβαντικό σύστημα το οποίο θα φτάνει την κβαντική υπεροχή.



Εικόνα 5: Κβαντική υπεροχή(<https://www.pbs.org/wgbh/nova/article/google-quantum-supremacy/>)

Το πρώτο κβαντικό σύστημα δημιουργήθηκε από τον Paul Benioff και μέχρι και σήμερα είναι γνωστό ως κβαντική μηχανή του Turing (Quantum Turing Machine(QTM)). Αποτελεί επέκταση της μηχανής Turing δηλαδή ενός μοντέλου που χειρίζεται τα σύμβολα μίας Μαρκοβιανής Αλυσίδας¹.

Όπως προαναφέρθηκε το συγκεκριμένο μοντέλο αποτέλεσε την βάση ενός κβαντικού υπολογιστική καθώς ήταν το πρώτο υπολογιστικό σύστημα που λειτούργησε με αρχές της κβαντικής μηχανικής.

Στην θεωρία του φυσικού Paul Benioff στηρίχτηκε ο επίσης φυσικός David Deutsch ο οποίος εισήγαγε την έννοια του καθολικού κβαντικού υπολογιστή.

Αποδεικνύοντας αργότερα πως τελικά ένας καθολικός κβαντικός υπολογιστής είναι ένα υπολογιστικό, σύστημα το οποίο λειτουργεί παράλληλα και με τις αρχές του κλασικού υπολογιστή αλλά και με τις αρχές ενός κβαντικού υπολογιστή.

Συγκεκριμένα ο καθολικός κβαντικό υπολογιστής έχει την δυνατότητα να λειτουργήσει ως κλασικός υπολογιστής αλλά με τις αρχές και ιδιότητες της κβαντικής μηχανικής οι οποίες καταλήγουν να είναι και ιδιότητες ενός κβαντικού υπολογιστή.

Οι ιδιότητές όμως ενός κβαντικού υπολογιστή όπως είναι φυσικό διαφέρουν από αυτές ενός κλασικού υπολογιστή αρχικά οι κβαντικές πληροφορίες αποθηκεύονται σε qubits(ανάλυση των οποίων θα γίνει σε επόμενο κεφάλαιο).

Εκτός από την κβαντική υπεροχή που αναφέρθηκε νωρίτερα, μια ακόμα ιδιότητα που έχουν τα κβαντικά συστήματα είναι του κβαντικού παραλληλισμού ή αλλιώς κβαντικής υπέρθεσης. Η παραλληλία σε ένα κβαντικό υπολογιστή χωρίζει το κάθε πρόβλημα σε μικρότερα και επιτρέπει την ταυτόχρονη εκτέλεση των προβλημάτων αυξάνοντας και με αυτόν τον τρόπο την επιτάχυνση του υπολογιστή. Οπότε με την έννοια της κβαντικής υπέρθεσης η κβαντική πληροφορία μπορεί να έχει ταυτόχρονα δύο τιμές 0 και 1.

¹ Ως Μαρκοβιανή Αλυσίδα(Markov Chain) ορίζεται ένα μοντέλο όπου η πιθανότητα εμφάνισης του κάθε συμβόλου της αλυσίδας καθορίζεται από το σύμβολο που έχει προηγηθεί. Σε μια Μαρκοβιανή Αλυσίδα η μετάβαση από την μια κατάσταση στην άλλη άρα και η πιθανότητα εμφάνισης της κάθε κατάστασης καθορίζεται από τον πίνακα μετάβασης δηλαδή από ένα στοχαστικό πίνακα.

Όπως και στην κβαντική φυσική έτσι και στην κβαντική υπολογιστική σε ένα κβαντικό σύστημα υπάρχει η ιδιότητα της διεμπλοκής.

Στην κβαντική υπολογιστική όταν 2 qubits βρεθούν σε διεμπλοκή και το ένα qubit μετρηθεί και καταρρεύσει σε μια κατάσταση τότε το αποτέλεσμα της κατάστασης του επηρεάζει ακαριαία και ανεξαρτήτου απόστασης την κατάσταση του άλλου qubit. Στην εικόνα 5 αναπαριστώνται 2 σωματίδια τα οποία βρίσκονται σε κατάσταση διεμπλοκής



Εικόνα5: Διεμπλοκή(Nobel Prize in Physics Awarded for Work on Quantum Entanglement (popularmechanics.com))

Ένα επίσης χαρακτηριστικό ενός κβαντικού υπολογιστή είναι η κβαντική παρεμβολή.

Ως κβαντική παρεμβολή ορίζεται το φαινόμενο σύμφωνα με το οποίο κατά την μεταφορά της πληροφορίας η πληροφορία μπορεί να αλλοιωθεί ή να χαθεί, καθώς οι πληροφορίες αποθηκεύονται σε κύματα και όπως είναι γνωστό τα κύματα αλληλοεπιδρούν ή καταστρέφονται.

Πέρα από την κβαντική παρεμβολή οι αλγόριθμοι για να επιλύσουν σωστά το πρόβλημα τους βασίζονται στα θεωρήματα.

Ένα από αυτά είναι το θεώρημα της μη κλωνοποίησης το οποίο αναφέρει πως η αντιγραφή μια κβαντικής κατάστασης δεν είναι εφικτή. Καθώς το 1982 αποδείχθηκε από τον Wootters, Zurek, Dikes πως αν γίνει προσπάθειά αντιγραφής μια κβαντικής κατάστασης το αποτέλεσμα αντιγραφής η αλλιώς κλωνοποίησης της κατάστασης θα διαφέρει από την αρχική κατάσταση.

Η ιδιότητα της κβαντικής διεμπλοκής που αναφέρθηκε νωρίτερα μπορεί να χρησιμοποιηθεί και σε διάφορες κβαντικές τεχνικές, όπως αυτή της τηλεμεταφοράς.

Η τηλεμεταφορά είναι μια τεχνική για την μεταφορά κβαντικού μηνύματος ανάμεσα σε 2 άτομα (qubits) με την βοήθεια ενός τρίτου ατόμου(qubit).

Σημαντικό ρόλο στην τηλεμεταφορά έχει η διεμπλοκή καθώς το βοηθητικό άτομο (το τρίτο άτομο) βρίσκεται σε διεμπλοκή με τον αποστολέα αρχικά και ύστερα σε διεμπλοκή με τον παραλήπτη και μεταφέρονται οι πληροφορίες του κβαντικού μηνύματος.

Για καλύτερα κατανόηση θα εξηγηθεί το φαινόμενο της τηλεμεταφοράς με το εξής παράδειγμα.

Έστω πως υπάρχουν δύο άτομα ο Bob και ο Alice, ο Bob περιέχει το μήνυμα με την κβαντική πληροφορία και θέλει να το στείλει στον Alice. Όμως δεν μπορεί να αντιγράψει το μήνυμα καθώς σύμφωνα με το θεώρημα μη κλωνοποίησης η αντιγραφή μιας κβαντικής κατάστασης δεν είναι εφικτή σε αυτή την περίπτωση εμφανίζεται ένα τρίτο άτομο ο Eric. Ο Eric βρίσκεται σε κατάσταση διεμπλοκής με τον Bob και αποθηκεύει την κβαντική πληροφορία του μηνύματος, στην συνέχεια Ο Eric βρίσκεται σε κατάσταση διεμπλοκής με τον Alice και του μεταφέρει την πληροφορία που πήρε από τον Bob. Με αυτόν τον τρόπο το μήνυμα φτάνει επιτυχώς από τον αποστολέα στον παραλήπτη. Στις εικόνες6,7,8 αναπαριστώνται ο Bob,ο Alice και ο Eric.



Εικόνα6:Bob Dylan(Bob Dylan birthday: Celebrate with photos of his legendary careerCNN), Εικόνα7:Alice Cooper (Alice Cooper - Singer, Musician, Actor, Restaurateur, Songwriter (tvinsider.com)), Εικόνα8:Eric Clapton(<https://guitar.com/guides/essential-guide/eric-claptons-20-greatest-guitar-moments-ranked/>)

Οι κβαντικοί υπολογιστές είναι ακόμα σε πειραματικό στάδιο και συνεχώς αναπτύσσονται, μπορούν να λύσουν οποιοδήποτε πρόβλημα όμως ο χρόνος εκτέλεσης του προβλήματος σε ένα κβαντικό υπολογιστή δεν είναι πάντα μικρότερος από τον χρόνο εκτέλεσης του προβλήματος σε ένα κλασικό υπολογιστή.

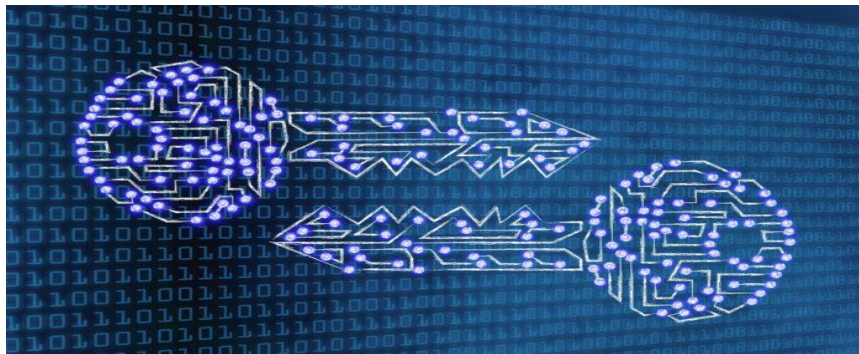
Παρολαυτά η κβαντική υπολογιστική επηρεάζει πολλούς τομείς μερικοί από αυτούς είναι οι εξής:

Κβαντική Κρυπτογραφία: Όπως είναι γνωστό η κρυπτογραφία είναι μια τεχνική διατήρησης των δεδομένων ασφαλή μέσω κρυτοσυστημάτων τα οποία περιέχουν αλγόριθμους κρυπτογράφησης.

Δηλαδή οι αλγόριθμοι κρυπτογράφησης είναι μαθηματικές συναρτήσεις οι οποίες χρησιμοποιούνται για να διατηρήσουν ασφαλές το (plain text) δηλαδή το απλό κείμενο το οποίο περιέχει δεδομένα.

Η κρυπτογράφηση του απλού κειμένου γίνεται και καθορίζεται με την βοήθεια μιας μεταβλητής που ονομάζεται κλειδί.

Ένα από τα πιο γνωστά κρυπτοσυστήματα είναι το RSA το οποίο είναι γνωστό ως κρυπτοσύστημα δημόσιου κλειδιού. Σύμφωνα με αυτό το κρυπτοσύστημα τα κλειδιά κρυπτογράφησης είναι δύο το ένα είναι ιδιωτικό και θεωρείτε ως ένα σύνολο από πρώτους αριθμούς για το οποίο έχει γνώση μόνο ο χρήστης του απλού κειμένου, με σκοπό να αποκρυπτογραφήσει το μήνυμα μόνο αυτός. Ενώ το άλλο κλειδί είναι δημόσιο στο οποίο η πρόσβασή είναι εφικτή από όλους, όμως η κωδικοποίηση RSA με την χρήση ενός κβαντικού υπολογιστή μπορεί να σπάσει σε ένα πάρα πολύ μικρό χρονικό διάστημα και τα δεδομένα που προστάτευε το συγκεκριμένο κρυπτοσύστημα όχι μόνο ενός ή 2 ατόμων αλλά πολλών τραπεζών να διαρρεύσουν σε ένα πάρα πολύ μικρό χρονικό διάστημα, δημιουργώντας με αυτόν τον τρόπο καταστροφικά αποτελέσματα. Ένας κβαντικός υπολογιστής θα είναι σε θέση σε μερικά χρόνια να σπάσει ένα κρυπτοσύστημα όπως το RSA. Καθώς λειτουργεί με ιδιότητες της κβαντικής υπολογιστικής τις οποίες χρησιμοποιούν αλγόριθμοι όπως ο αλγόριθμος Peter Shor που είναι αλγόριθμος εύρεσης πρώτων παραγόντων ενός ακεραίου αριθμού σε πολύ μικρό χρονικό διάστημα



Εικόνα 9:Κβαντική Κρυπτογραφία(Post Quantum Cryptography: Data Security in a Post-Quantum World - Hashed Out by The SSL Store™)

Κβαντική Μουσική: Παρόλο που ένας μουσικός δεν συμβαδίζει με ένα φυσικό καθώς ο ρόλος ενός μουσικού είναι να δημιουργήσει μέσα από την δουλειά του ένα αρμονικά και ακουστικά όμορφο αποτέλεσμα και ο ρόλος ενός φυσικού είναι να δώσει λογική εξήγηση σε κάποιο πρόβλημα.

Η κβαντική μηχανική κατάφερε να συνδεθεί με την μουσική θεωρία και να δημιουργήσει έναν νέο κλάδο την κβαντική μουσική. Το 2015 ξεκίνησε ένα έργο με το όνομα Quantum Music το οποίο βασίστηκε στην κβαντική μουσική.

Το μεγαλύτερο κομμάτι της κβαντικής μουσικής πραγματοποιήθηκε σε εργαστήριο και συγκεκριμένα όλες οι μελωδίες προκύπτουν από την μελέτη των κβαντικών φαινομένων.

Συγκεκριμένα για την παραγωγή μίας μελωδίας έγινε χρήση της κυματοσυνάρτησης της συχνότητας από ένα συμπύκνωμα Bose-Einstein.

Δηλαδή οι επιστήμονες παίρνουν τα δεδομένα που προκύπτουν από την συνάρτηση τα μετατρέπουν σε ήχο, με την βοήθεια αλγορίθμων και διάφορων μουσικών λογισμικών. Έστερά το αποτελέσματα ή οι μουσικές πληροφορίες στέλνονται μέσα από MIDI² αισθητήρες σε πιανίστες.

Τα αποτελέσματα αναπαράχθηκαν μέσα από δύο υβριδικά πιάνο, από τα οποία το ένα ήταν κλασικό πιάνο ενώ το άλλο ήταν ηλεκτρονικό πιάνο, τα δύο αυτά μουσικά όργανα θεωρούνται υβριδικά καθώς συνδέονται μεταξύ τους αλλά και με τον υπολογιστή μέσα από MIDI αισθητήρες και από λογισμικά που είναι εγκατεστημένα στον υπολογιστή. Τα εργαλεία αυτά αλλά και αλγόριθμοι που τρέχουν στον κβαντικό υπολογιστή συμβάλουν στην παραγωγή δεδομένων και στην παραγωγή της μελωδίας στο ακροατήριο. Τα κβαντικά αποτελέσματα που προκύπτουν κάθε φορά από τα φαινόμενα είναι απρόβλεπτα και βασίζονται σε πιθανότητες με αποτέλεσμα κάθε φορά να δημιουργούνται διαφορετικές και απρόβλεπτες μελωδίες. Πέρα από την αναπαραγωγή των κβαντικών δεδομένων στο ακροατήριο το project Quantum Music το οποίο συνεχίστηκε αργότερα ως project Beyond Quantum Music παρήγαγε και βιβλιοθήκη στην γλώσσα Python που περιέχει κβαντικές μουσικές συνθέσεις δημιουργώντας με αυτό τον τρόπο ένα καινούριο είδους μουσικής.

Εκτός από την δημιουργία του κβαντικού πιάνου η κβαντική μουσική αλλά και η παραγωγή μουσικής από κβαντικούς αλγόριθμους το 2022 εμφανίζεται και στις εικονικές τέχνες (visual arts).

²Το MIDI είναι ένα πρωτόκολλο μέσα από το οποία επικοινωνεί ένας υπολογιστής με μουσικά όργανα και εικονικά μουσικά όργανα. Δεν παράγει μουσική μεταφέρει μόνο τις πληροφορίες της μουσικής θεωρίας. Συνεπώς ένας MIDI αισθητήρας περιέχει αποκλειστικά μουσικές πληροφορίες

Συγκεκριμένα, μια ομάδα ερευνητών από την Αυστραλία μέσα από την συμμετοχή τους στον διαγωνισμό Qiskit Hackathon Melbourne³ δημιούργησε μια βιβλιοθήκη στην Python η οποία μετατρέπει ένα κβαντικό κύκλωμα σε μουσικό βίντεο.

Η βιβλιοθήκη όπως και ολόκληρο το έργο που παρουσιάστηκε στον διαγωνισμό ονομάστηκε qMuVi(quantum music video) μέσα από αυτή την βιβλιοθήκη γίνεται εφικτή η ακουστική αναπαράσταση μίας κβαντικής κατάστασης, όπου η κάθε κβαντική κατάσταση αναπαρίσταται με τον πίνακα πυκνότητας⁴ της. Καθώς σκοπός των ερευνητών ήταν να έχουν σωστά δεδομένα οι κβαντικές καταστάσεις χωρίστηκαν σε δύο κατηγορίες στις καθαρές κβαντικές καταστάσεις και στις μικτές κβαντικές καταστάσεις. Ως καθαρές κβαντικές καταστάσεις, ορίζονται οι καταστάσεις υπέρθεσής ενός ή περισσότερων qubits, ενώ ως μικτές κβαντικές καταστάσεις ορίζονται οι καταστάσεις όπου το αποτέλεσμα δεν είναι σαφές, οπότε μια μικτή κατάσταση θα αναλυθεί ως ένα σύνολο απλών κβαντικών καταστάσεων. Οι καθαρές καταστάσεις αναπαριστώνται κάθε μια ως μια ξεχωριστή νότα, δηλαδή στην υπέρθεση κάθε κατάσταση είναι μια διαφορετική νότα και ο συνδυασμός και των δύο καταστάσεων δημιουργεί μια χορδή. Σχετικά με τις μικτές καταστάσεις καθώς είναι ένα σύνολο από καθαρές καταστάσεις η αναπαράσταση τους μέσω της βιβλιοθήκης έγινε με την μορφή συγχορδιών. Με αυτό τον τρόπο προέκυψε η παραγωγή της μουσικής μέσα από το κβαντικό κύκλωμα.

Ένας ακόμα τρόπος παραγωγής μουσικής είναι από κβαντικούς αλγόριθμους, ο οποίος χρησιμοποιήθηκε και στο project που αναφέρθηκε νωρίτερα.

Ο τρόπος αυτός χρησιμοποιήθηκε και στο έργο qMuVi και μπορεί να χρησιμοποιηθεί για την παραγωγή μουσικής από το υβριδικό πιάνο

Ένας αλγόριθμος που θα βοηθάει στην παραγωγή μουσικής είναι ο αλγόριθμος του Grover. Με μία σύντομη περιγραφή καθώς στην συνέχεια θα αναλυθεί πλήρως ο παρακάτω αλγόριθμος, ο αλγόριθμος του Grover πήρε αρχικά το όνομα του από τον Lov Grover το 1996 και είναι αλγόριθμος αναζήτησης ενός στοιχείου μέσα σε ένα μη δομημένο σύνολο.

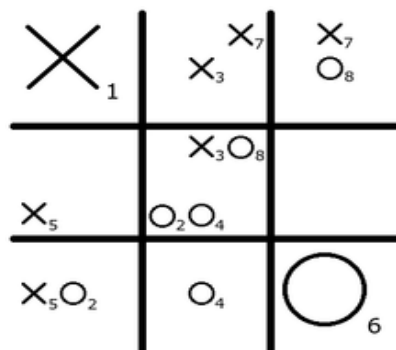
³ Μεγαλύτερος διαγωνισμός πάνω στον κάδο της κβαντικής υπολογιστικής, διοργανώνεται από την κβαντική ομάδα της IBM

⁴ Πίνακας πυκνότητάς είναι ένας πίνακας που περιγράφει την κβαντική κατάσταση ενός συστήματος

Ο αλγόριθμος του Grover μπορεί να χρησιμοποιηθεί στην κβαντική μουσική αναζήτη-
τώντας κάθε φορά νότες που πρέπει να αναπαραχθούν κάθε φορά με βάση κανόνες
της θεωρίας της μουσικής.

Ο αλγόριθμος του Grover σε αντίθεση με τους άλλους κβαντικούς αλγορίθμους πα-
ρέχει τετραγωνική επιτάχυνση γεγονός που τον καθιστά πιο γρήγορο από τους κλα-
σικούς αλγόριθμους.

Κβαντικά Παιχνίδια: Η κβαντική θεωρία παιγνίων αποτελεί επέκταση της κλασικής
θεωρίας παιγνίων, ως παίγνια ορίζονται οι μαθηματικοί μέθοδοι που χρησιμοποιού-
νται για την σωστή αντιμετώπιση των καταστάσεων σε περιπτώσεις σύγκρουσης ή
συνεργασίας. Στην θεωρία των παιγνίων ο παίκτης πρέπει να είναι ευφυής έτσι ώστε
μπορεί να παίξει σωστά το παιχνίδι και πρέπει να έχει μια λογική στον τρόπο που
παίζει έτσι ώστε να κερδίσει το παιχνίδι. Το παίγνιο αποτελείται από τον παίκτη που
μπορεί να είναι ένα άτομο, την στρατηγική που είναι ο τρόπος με τον οποίο θα επι-
λέξει ο παίκτης να κινηθεί ανάλογα με τους κανόνες τους παιχνιδιού και την λύση
του παιγνίου που είναι ο βέλτιστος τρόπος με τον οποίο κινούνται όλοι οι παίκτες.
Στην κβαντική θεωρία των παιγνίων εμφανίζονται και οι ιδιότητες της κβαντικής υ-
πολογιστικής οι οποίες είναι η υπέρθεση και η κβαντική διεμπλοκή. Ένα κλασικό
παράδειγμα κβαντικού παιχνιδιού είναι η κβαντική τρίλιζα σύμφωνα με την οποία
κάθε κίνηση του παίκτη μπορεί να βρίσκετε σε 2 σημεία του ταμπλό της τρίλιζας
λόγω της διεμπλοκής των κβαντικών σωματιδίων. Τα σωματίδια αυτά ονομάζονται
μπλεγμένα. Η τρίλιζα καταρρέει σε κλασικές κατάστασης δημιουργώντας μια κυ-
κλική εμπλοκή στην εικόνα10:



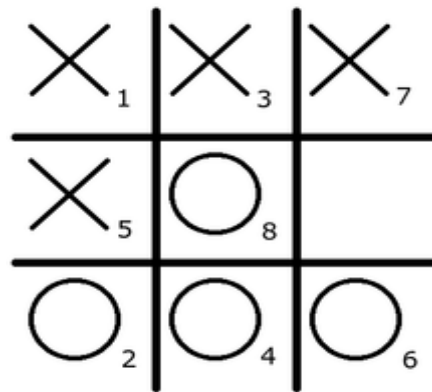
Εικόνα 10: Κβαντική Τρίλιζα(Quantum tic-tac-toe - Wikipedia)

- Το τετράγωνο 2 εμπλέκεται με το τετράγωνο 3 και με το τετράγωνο 5
- Το τετράγωνο 3 εμπλέκεται με το τετράγωνο 2 και το τετράγωνο 5

- Το τετράγωνο 7 εμπλέκεται με το τετράγωνο 4
- Το τετράγωνο 8 εμπλέκεται με το τετράγωνο 5

Αυτά τα τετράγωνα δημιουργούν την κυκλική εμπλοκή, στο τέλος ο παίκτης που δεν δημιούργησε την κβαντική εμπλοκή επιλέγει ένα τρόπο να μετρήσει τον κύκλο. Με σκοπό να κερδίσει ο ίδιος, ένας τρόπος να μετρηθεί η κβαντική εμπλοκή είναι ο εξής και αναπαριστάτε στην εικόνα 11:

- Το x7 συμπιέζεται στο τετράγωνο 3 οδηγώντας το ο8 να συμπιεστεί στο τετράγωνο 5
- Το x3 συμπιέζεται στο τετράγωνο 2 οδηγώντας το ο4 να συμπιεστεί στο τετράγωνο 8 και οδηγώντας και το ο2 να συμπιεστεί στο τετράγωνο 7 οπότε το αποτέλεσμα που θα προκύψει φαίνεται στην παρακάτω εικόνα



Εικόνα11: Κβαντική Τρίλιζα(Quantum tic-tac-toe - Wikipedia)

Ένα άλλο παράδειγμα κβαντικού παιχνιδιού διαφορετικό από την κβαντική τρίλιζα είναι το κβαντικό πόκερ το οποίο δημιουργήθηκε για τις ανάγκες του διαγωνισμού Quantum Game Jam 2023.

Σε αντίθεση με το κλασικό πόκερ στο κβαντικό πόκερ ο κάθε παίκτης αντί για μία κάρτα έχει στην αρχή του παιχνιδιού δύο κάρτες, κάθε κάρτα περιέχει δύο καταστάσεις (δηλαδή δύο κάρτες) οπότε η τελική κατάσταση της κάθε κάρτας δεν είναι γνωστή και πέντε κάρτες είναι στο τραπέζι του πόκερ. Πέρα από τις 2 κάρτες ο κάθε παίκτης έχει και έξι ακόμα κάρτες τις οποίες μπορεί να παίζει κάθε φορά ώστε να επηρεάζει τις κάρτες που βρίσκονται στο τραπέζι με σκοπό να κερδίσει ο ίδιος, μόλις τελειώσουν οι έξι κβαντικές κάρτες το παιχνίδι τελειώνει, ο παίκτης με την υψηλότερη βαθμολογία είναι ο νικητής. Οι έξι κάρτες που έχει ο κάθε παίκτης είναι οι εξής

- Μέτρηση(Measure): Ο παίκτης επιλέγει σε ποια από τις δύο καταστάσεις θα καταρρεύσει η κάρτα που βρίσκεται στο ταμπλό

- Παρατήρηση (Observe): Η κάρτα καταρρέει σε μια κατάσταση και αν έχει προηγηθεί η κάρτα measure τότε η κάρτα θα καταρρεύσει στην κατάσταση που επιλέχτηκε με την χρήση της measure
 - Εμπλοκή(Entangle):Εμπλοκή δύο φύλλων, τα οποία δεν έχουν μετρηθεί, το ένα φύλλο επηρεάζει το άλλο από την στιγμή που και τα δύο είναι μπλεγμένα, για παράδειγμα αν γίνει μέτρησή του ενός φύλλου και ύστερα παρατήρηση του φύλλου τότε το ίδιο συμβαίνει και στο άλλο φύλλο
 - Αρμονική Ταλάντωση(Harmonic Oscillation): Μέσω αυτής της κάρτας ο παίκτης έχει την δυνατότητα να εμπλέκει κάθε κάρτα με τις γειτονικές της (δεξιά και αριστερά της). Αν οι κάρτες είναι ήδη μπλεγμένες τότε η κάρτα Harmonic Oscillation δεν λειτουργεί
 - Περπάτημα(Walk): Ο παίκτης έχει την δυνατότητα να αλλάξει την θέση ανάμεσα σε δύο κβαντικές κάρτες
 - Σήραγγά(Tunnel): Ο παίκτης ξαναπαίζει μια κάρτα που είχε παίξει και νωρίτερα
- Το κβαντικό πόκερ μαζί με τις κβαντικές βοηθητικές κάρτες αναπαρίσταται στην εικόνα12:



Εικόνα12:Κβαντικό Πόκερ

(https://frib.itch.io/quantumpoker?fbclid=IwAR04aHriXehyo-bTyi7P_eN7YKwRH1rzsgp7gZwn7bsrlZIGxPQQTfFX78Q)

2.0 Μαθηματικό Υπόβαθρό και Python

Τα μαθηματικά και συγκεκριμένα η γραμμική άλγεβρα αποτελούν σημαντικό μέρος για την κβαντική υπολογιστική. Καθώς η αναπαράσταση των qubits γίνεται με την μορφή διανυσμάτων ενώ οι κβαντικές πύλες που χρησιμοποιούνται στα κυκλώματα είναι πίνακες.

2.1 Εισαγωγή στους πίνακες

Ως πίνακας ορίζεται μιας ορθογώνια διάταξη αριθμών ή άλλων συμβόλων διατεταγμένων σε γραμμές και στήλες. Στους πίνακες με ίδιες διαστάσεις μπορεί να γίνει πρόσθεση πολλαπλασιασμός των στοιχείων όπως γίνεται κανονικά πράξεις ανάμεσα σε αριθμούς. Η μόνη ιδιαιτερότητα στις πράξεις των πινάκων είναι στον πολλαπλασιασμό 2 πινάκων καθώς για να γίνει η συγκεκριμένη πράξη χρειάζεται ο αριθμός των γραμμών του πρώτου πίνακα να ισούται με το αριθμό των στηλών του δεύτερου πίνακα.

Η γενική μορφή ενός πίνακα όπου m είναι οι γραμμές του πίνακα και n είναι οι στήλες του πίνακα είναι η εξής :

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Ακόμα υπάρχουν πίνακες που καθορίζονται μόνο από μια γραμμή ή μόνο από μία στήλη. Αυτοί οι πίνακες ορίζονται ως διανύσματα γραμμής και διανύσματα στήλης αντίστοιχα.

Τα διανύσματα γραμμής έχουν διαστάσεις $1 \times n$ όπου n είναι ο αριθμός των στηλών ενός πίνακα. Παράδειγμα διανύσματος γραμμής είναι το εξής:

- $A = [1 \quad 4 \quad 6]$

Ενώ Τα διανύσματα στήλης έχουν διαστάσεις $n \times 1$ όπου n είναι ο αριθμός των στηλών ενός πίνακα. Παράδειγμα διανύσματος γραμμής είναι το εξής:

$$\text{o } A = \begin{bmatrix} 1 \\ 5 \\ 9 \end{bmatrix}$$

Εκτός από τον κλασικό πίνακα υπάρχει και ο ανάστροφος του που δίνεται από την αντανάκλαση ως προς την κύρια διαγώνιο του.

Συμβολίζεται ως A^T με $(A^T)_{ij} = A_{ji}$ και έχει διαστάσεις

Όπου ισχύει πως : $1 \leq i \leq n$ και $1 \leq j \leq m$

Οπότε αν :

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \text{ τότε } A^T = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}$$

Η αναπαράσταση ενός πίνακα πέρα από αγκύλες μπορεί να πραγματοποιηθεί και μέσα από παρενθέσεις

Κάποια παραδείγματα κανονικών και ανάστροφων πινάκων είναι τα εξής:

$$A_{2 \times 2} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

$$B = [3 \quad 4] \quad B^T = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

Πρόσθεση και Αφαίρεση Πινάκων

Η πρόσθεση και αφαίρεση των στοιχείων δύο πινάκων ίδιων διαστάσεων γίνεται με την βοήθεια του παρακάτω τύπου:

$$\text{o } c_{i,j} = a_{i,j} + b_{i,j}$$

ο Όπου a είναι κάθε στοιχείο ενός πίνακα $A_{m \times n}$ ενώ b είναι κάθε στοιχείο ενός πίνακα $B_{m \times n}$, ακόμα το i δηλώνει την γραμμή του πίνακα ενώ το j δηλώνει την στήλη του πίνακα

Ενώ παραδείγματα με πράξεις πινάκων είναι τα εξής:

$$A_{2 \times 3} + B_{2 \times 3} = \begin{bmatrix} 1 & 6 \\ 3 & 4 \\ 5 & 2 \end{bmatrix} + \begin{bmatrix} 7 & 1 \\ 9 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1+7 & 6+1 \\ 3+9 & 4+2 \\ 5+0 & 2+1 \end{bmatrix} = \begin{bmatrix} 8 & 7 \\ 11 & 6 \\ 5 & 3 \end{bmatrix} = C_{2 \times 3}$$

$$A_{2 \times 3} - B_{2 \times 3} = \begin{bmatrix} 7 & 6 \\ 8 & 4 \\ 5 & 2 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 3 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 7-1 & 6-1 \\ 8-3 & 4-2 \\ 5-0 & 2-1 \end{bmatrix} = \begin{bmatrix} 6 & 5 \\ 5 & 2 \\ 5 & 1 \end{bmatrix} = C_{2 \times 3}$$

Πολλαπλασιασμός Πινάκων

Όπως αναφέρθηκε και νωρίτερα κατά τον πολλαπλασιασμό δύο πινάκων ο αριθμός των γραμμών του πρώτου πίνακα πρέπει να είναι ίδιος με τον αριθμό των στηλών του δεύτερου πίνακα δηλαδή πρέπει να ισχύει ο κανόνας:

- $C_{n,m} = A_{n,l} \times C_{l,m}$
- $c_{i,j} = \sum_{k=0}^{l-1} (a_{i,k} \times b_{k,j})$

ο συγκεκριμένος τύπος αυτό που δείχνει είναι πως το κάθε στοιχείο του τελικού πίνακα είναι το άθροισμα των γινομένων των στοιχείων της κάθε γραμμής με τα στοιχεία της αντίστοιχης στήλης

Ένα παράδειγμα για την καλύτερη κατανόηση του παραπάνω ισχυρισμού είναι το εξής:

$$A_{2 \times 3} \times B_{3 \times 2} = \begin{bmatrix} 7 & 1 & 6 \\ 8 & 2 & 4 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 4 & 1 \\ 3 & 2 \end{bmatrix} =$$

$$\begin{bmatrix} 7 * 1 + 1 * 4 + 6 * 3 & 7 * 2 + 1 * 1 + 6 * 2 \\ 8 * 1 + 2 * 4 + 4 * 3 & 8 * 2 + 2 * 1 + 4 * 2 \end{bmatrix} = C_{2 \times 2} \begin{bmatrix} 29 & 27 \\ 28 & 26 \end{bmatrix}$$

Εκτός από τον κλασικό πολλαπλασιασμό πινάκων μια πράξη απαραίτητη ανάμεσα σε δύο πίνακες ή σε δύο διανύσματα είναι το τανυστικό γινόμενο (Kronecker product). Σύμφωνα με το τανυστικό γινόμενο όταν υπάρχουν δύο πίνακες γίνεται πολλαπλασιασμός του κάθε στοιχείου του πρώτου πίνακα με όλα τα υπόλοιπα στοιχεία του δεύτερου πίνακα.

Τα στοιχεία τοποθετούνται στο καινούριο πίνακα με βάση το μήκος του δεύτερου πίνακα. Το σύμβολο του τανυστικού γινομένου είναι $\otimes$

$$\text{Αν } A_{2 \times 2} \text{ και } B_{2 \times 3} \text{ τότε } A_{2 \times 2} \otimes B_{2 \times 3} = C_{4 \times 6}$$

$$\text{Έστω } A_{2 \times 2} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \text{ και } B_{2 \times 3} = \begin{pmatrix} 3 & 1 & 5 \\ 2 & 2 & 4 \end{pmatrix}$$

$$\text{Τότε } A_{2 \times 2} \otimes B_{2 \times 3} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix} \otimes \begin{pmatrix} 3 & 1 & 5 \\ 2 & 2 & 4 \end{pmatrix} =$$

$$\begin{pmatrix} 1 * \begin{pmatrix} 3 & 1 & 5 \\ 2 & 2 & 4 \end{pmatrix} & 3 * \begin{pmatrix} 3 & 1 & 5 \\ 2 & 2 & 4 \end{pmatrix} \\ 2 * \begin{pmatrix} 3 & 1 & 5 \\ 2 & 2 & 4 \end{pmatrix} & 4 * \begin{pmatrix} 3 & 1 & 5 \\ 2 & 2 & 4 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 3 & 1 & 5 & 9 & 3 & 15 \\ 2 & 2 & 4 & 6 & 6 & 12 \\ 6 & 2 & 10 & 12 & 4 & 20 \\ 4 & 4 & 8 & 8 & 8 & 16 \end{pmatrix}$$

Αξίζει να σημειωθεί πως στο τανυστικό γινόμενο δεν υπάρχει η ιδιότητα της αντιμεταθετής πράξης δηλαδή:

$$\circ \text{ Τότε } A \otimes B \neq B \otimes A$$

Ενώ οι υπόλοιπες ιδιότητες του τανυστικού γινομένου είναι οι εξής:

$$\circ A \otimes (B + C) = (A \otimes B) + (A \otimes C)$$

$$\circ (A + B) \otimes C = (A \otimes C) + (B \otimes C)$$

$$\circ (A \otimes B) \otimes C = A \otimes (B \otimes C)$$

$$\circ (A \otimes B)(C \otimes D) = (A * C) \otimes (B * D)$$

2.2 Μιγαδικοί Αριθμοί

Οι μιγαδικοί αριθμοί δημιουργήθηκαν από τον μαθηματικό Τζερόλαμο Καρτάνο στην προσπάθεια του να λύσει κυβικές εξισώσεις της μορφής:

$$\bullet \quad x^2 = -1 \Rightarrow x = \sqrt{-1} = i \text{ όπου } i \text{ είναι ένας φανταστικός αριθμός}$$

Οπότε οι μιγαδικοί αριθμοί αποτελούν επέκταση των πραγματικών αριθμών $: R \rightarrow C$ με μία επιπλέον φανταστική μονάδα το i

- C είναι το σύνολο των μιγαδικών αριθμών
- Όπου η φανταστική μονάδα i έχει την εξής ιδιότητα: $i^2 = -1$

Ένας μιγαδικός αριθμός που ανήκει στο σύνολο C έχει την εξής μορφή: $z = a + b*i$

- $\text{Re}(z) = a$ και είναι το πραγματικό μέρος του αριθμού
- $\text{Im}(z) = b$ είναι το φανταστικό μέρος του αριθμού

Ένας πραγματικός αριθμός είναι μιγαδικός αριθμός μόνο αν είναι της μορφής:

$$\bullet \quad z = a + b * 0$$

Ενώ ένας μιγαδικός αριθμός είναι φανταστικός αριθμός αν το πραγματικό μέρος είναι 0 δηλαδή αν ο αριθμός είναι της μορφής:

$$\bullet \quad z = 0 + b * i$$

Είναι εξίσου σημαντικό να αναφερθεί πως δυο μιγαδικοί αριθμοί είναι ίσοι μόνο όταν το πραγματικό τους μέρος είναι ίσο δηλαδή όταν οι μιγαδικοί αριθμοί είναι οι $z_1 = x_1 + y_1 * i$ και $z_2 = x_2 + y_2 * i$ τότε οι δύο αριθμοί είναι ίσοι εάν $x_1 = x_2$ και $y_1 = y_2$

2.2.1 Πράξεις με μιγαδικούς αριθμούς

Όπως και στους πραγματικούς έτσι και στους μιγαδικούς αριθμούς υφίστανται οι πράξεις της πρόσθεσης της αφαίρεσης και του πολλαπλασιασμού.

Στην πρόσθεση και στην αφαίρεση γίνεται πρόσθεση/αφαίρεση του πραγματικού μέρους με το πραγματικό και του φανταστικού με το φανταστικό. Δηλαδή:

- Εάν $x_1 = a + b * i$ και $x_2 = c + d * i$ τότε $x = (a + c) + (b + d) * i$
- Εάν $x_1 = a + b * i$ και $x_2 = c + d * i$ τότε $x = (a - c) + (b - d) * i$

Η πρόσθεση δύο μιγαδικών αριθμών

Οπότε με βάση τους παραπάνω τύπους με $x_1 = 5 + 2 * i$ και $x_2 = 2 + 3 * i$ ισχύει:

- Πρόσθεση: $x = x_1 + x_2 = (5 + 2 * i) + (2 + 3 * i) = (5 + 2) + (2 + 3) * i = 7 + 5 * i$
- Αφαίρεση: $x = x_1 - x_2 = (5 + 2 * i) - (2 + 3 * i) = (5 - 2) + (2 - 3) * i = 2 - i$

Ένα άλλο παράδειγμα με πρόσθεση και αφαίρεση μιγαδικών αριθμών μπορεί να είναι το εξής με $x_1 = 4 + 2 * i$ και $x_2 = 3 + 2 * i$:

- Πρόσθεση: $x = x_1 + x_2 = (4 + 2 * i) + (3 + 2 * i) = (4 + 3) + (2 + 2) * i = 7 + 4 * i$
- Αφαίρεση: $x = x_1 - x_2 = (4 + i * 2) - (3 + i * 2) = (4 - 3) + i * (2 - 2) = 1 + i * 0$

Σε αντίθεση με την πρόσθεση και την αφαίρεση μιγαδικών αριθμών ο πολλαπλασιασμός των μιγαδικών δεν απαιτεί διαχωρισμό φανταστικού με πραγματικού μέρους πραγματοποιείται μέσω της επιμεριστικής ιδιότητας δηλαδή:

- Εάν $x_1 = a + b * i$ και $x_2 = c + d * i$ τότε $x = x_1 * x_2 = (a + b * i) * (c + d * i) = ac + ad * i + bc * i + bd * i^2 = (ac - bd) + (ad + bc) * i \Rightarrow x = (ac - bd) + (ad + bc) * i$
- Παράδειγμα με πολλαπλασιασμό μιγαδικών αριθμών είναι το εξής:

$$\begin{aligned} \text{Εάν } x_1 &= 2 + 3 * i \text{ και } x_2 = 4 + 2 * i \text{ τότε } x = x_1 * x_2 \Rightarrow x = \\ (2 + 3 * i) * (4 + 2 * i) &= (6 + 4 * i + 12 * i + 6 * i^2) = (6 + 16 * i - \\ 6) &= 0 + 16 * i \end{aligned}$$

Συγκεκριμένα στους μιγαδικούς αριθμούς ο πολλαπλασιασμός κιόλας συμβολίζεται με το σύμβολο $\otimes$ όπου: $(a, b) \otimes (c, d) = (a * c - b * d, a * d + c * b)$

- Ένα παράδειγμα είναι το εξής: Έστω ένα στοιχείο i οποίο είναι $(0,1) \in \mathbb{C}$
 $i^2 = (0,1) \otimes (0,1) = (0 * 0 - 1 * 1, 1 * 0 + 0 * 1) = (-1, 0) = -1$
 Όπου προκύπτει πως το -1 δεν είναι τελικά αρνητικός αριθμός αλλά ένας διαφορετικός τρόπος αναπαράστασης του μιγαδικού $(-1,0)$

Σε αυτό το σημείο αξίζει να αναφερθεί ότι στις πράξεις των μιγαδικών αριθμών και κυρίως στην διαίρεση εμφανίζονται οι συζυγή μιγαδικοί αριθμοί. Όπου δύο μιγαδικοί αριθμοί είναι συζυγείς μόνο όταν το πραγματικό τους μέρος είναι ίδιο και το φανταστικό μέρος είναι αντίθετο για παράδειγμα

- Εάν υπάρχει ο μιγαδικός αριθμός $z = a + b * i$ τότε ο συζυγής μιγαδικός του z συμβολίζεται με $\bar{z} = a - b * i$
- Έστω ο μιγαδικός $z = 4 + 7 * i$ τότε ο συζυγής μιγαδικός του z είναι ο $\bar{z} = 4 - 7 * i$

Ενώ στην διαίρεση δύο μιγαδικών αριθμών αρχικά γίνεται πολλαπλασιασμός αριθμητή και παρανομαστή με τον συζυγή του παρανομαστή έτσι ώστε να είναι ο παρανομαστής μεγαλύτερος του 0 για να έχει ισχύ το κλάσμα.

Οπότε η διαίρεση δύο μιγαδικών αριθμών έχει την εξής μορφή:

- $z = \frac{x}{y} = \frac{a + b * i}{c + d * i} = \frac{(a + b * i) * (c - d * i)}{(c + d * i) * (c - d * i)} = \frac{(ac + bd) + (bc - ad) * i}{(c^2 + d^2)} = \frac{ac + bd}{(c^2 + d^2)} + \frac{bc - ad}{(c^2 + d^2)} * i$
- Ενώ ένα παράδειγμα με διαίρεση δύο μιγαδικών αριθμών μπορεί να είναι το
 $\text{εξής } z = \frac{x}{y} = \frac{2 + 3 * i}{4 + 2 * i} = \frac{(2 + 3 * i) * (4 - 2 * i)}{(4 + 2 * i) * (4 - 2 * i)} = \frac{(2 * 4 + 3 * 2) + (3 * 4 - 2 * 2) * i}{(4^2 - 2^2)} = \frac{14}{20} + \frac{16}{20} * i$

2.2.2 Μιγαδικό Επίπεδο

2.2.2.1 Καρτεσιανές Συντεταγμένες

Κάθε μιγαδικός αριθμός $z = a + b * i$ αναπαριστάτε σε ένα καρτεσιανό σύστημα συντεταγμένων.

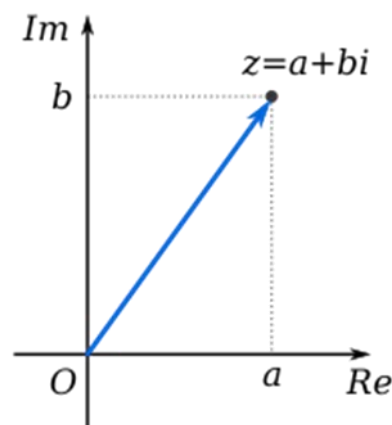
Στην εικόνα 13 το σημείο που βρίσκεται ο μιγαδικός αριθμός z συμβολίζεται κυρίως $M(z)$.

Όπου ως M ορίζεται η εικόνα ενός μιγαδικού αριθμού, ο άξονας $x x'$ αναπαριστά των άξονα των πραγματικών αριθμών ενώ ο $y y'$ αναπαριστά τον άξονα των φανταστικών αριθμών. Η απόσταση από την αρχή των αξόνων μέχρι το σημείο M ονομάζεται μέτρο του μιγαδικού αριθμού και είναι ένα διάνυσμα το οποίο αναπαριστάτε με την εξής μορφή:

- $|z| = r = \sqrt{a^2 + b^2}$
- Εάν $z = 2 + 3 * i$ τότε $|z| = \sqrt{2^2 + 3^2} = \sqrt{13}$

Ακόμα η γωνία που δημιουργείτε με τον άξονα των πραγματικών αριθμών ονομάζεται φάση (γωνία φ) και μαζί με το μέτρο καθορίζουν τον μιγαδικό αριθμό.

- Η γωνία φ μετρείται σε ακτίνια



Εικόνα 13: Αναπαράσταση Μιγαδικού στο
καρτεσιανό επίπεδο

(Μιγαδικός αριθμός - Βικιπαίδεια (wikipedia.org))

2.2.2.2 Πολικές Συντεταγμένες

Η αναπαράσταση ενός μιγαδικού αριθμού είναι εφικτή και με πολική ή τριγωνομετρική μορφή.

Η πολική μορφή των μιγαδικών τους δίνει την δυνατότητα να αναπαριστώνται στον χώρο ως περιστρεφόμενά διανύσματα.

Ένας μιγαδικός αριθμός έχει σαν πολικές συντεταγμένες το ζευγάρι (r, φ) .

Οπού r το μέτρο του μιγαδικού αριθμού όπως αναφέρθηκε και πριν ενώ φ είναι η φάση.

Η τριγωνομετρική αναπαράσταση ενός μιγαδικού αριθμού γίνεται ως εξής:

- Έστω $z = a + b * i$ από την εικόνα 13 είναι αντιληπτό ότι το πραγματικό μέρος του μιγαδικού αριθμού δηλαδή το a μπορεί να αναπαρασταθεί και ως το συνημίτονο της γωνίας φ με την υποτείνουσα δηλαδή με το r . Δηλαδή $a = r * \cos \varphi$.
- Ενώ το b αναπαριστάτε ως το ημίτονο της γωνίας φ με την υποτείνουσα δηλαδή με το r . Δηλαδή $b = r * \sin \varphi$.
- Προκύπτει λοιπόν το εξής : $z = a + b * i = r(\cos \varphi + i \sin \varphi)$

Μια άλλη αναπαράσταση ενός μιγαδικού είναι μέσω του τύπου Όιλερ δηλαδή:

- $z = r * e^{i\varphi}$ καθώς $e^{i\varphi} = \cos \varphi + i \sin \varphi$

Για καλύτερη κατανόηση των παραπάνω θα χρησιμοποιηθεί ένα παράδειγμα:

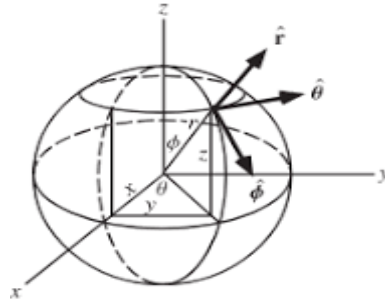
- Έστω $z = r \cos \varphi + i \sin \varphi = r e^{i\varphi}$ από την εύρεση των συντεταγμένων είναι γνωστό το πραγματικό και το φανταστικό μέρος του μιγαδικού αριθμού δηλαδή του a και του b
1. Είναι γνωστό πως $z = a + b * i$
 2. $a = r * \cos \varphi$, $b = r * \sin \varphi$

Οπότε εάν $z = 10e^{i3.14159}$ τότε $r = 10$, $\varphi = 3.14159$

Από το βήμα 2 παραπάνω προκύπτει πως $a = 10 \cos 3.14159$, $b = 10 \sin 3.14159$ οπότε: $a = -10$, $b = 0$, $z = -10 + 0 * i = -10$

2.2.2.3 Σφαιρικές Συντεταγμένες

Ένας μιγαδικός μπορεί να αναπαρασταθεί επίσης ως ένα σημείο σε ένα τρισδιάστατο χώρο όπως το σφαιρικό σύστημα συντεταγμένων το οποίο είναι επέκταση του πολικού συστήματος συνταγμένων και αναπαρίσταται στην εικόνα 14



Εικόνα 14.0: Αναπαράσταση

Μιγαδικού σε σφαιρικές συντεταγμένες

Το κάθε σημείο σε έναν τρισδιάστατο χώρο καθορίζεται από το κέντρο των αξόνων O , από την πολική γωνία που αναπαρίσταται με το γράμμα θ και μετράτε από την κατεύθυνση του άξονα z και από το αζιμούθιο το οποίο αναπαριστάτε με το γράμμα φ και είναι η γωνία του άξονα δηλαδή του άξονα προβολής.

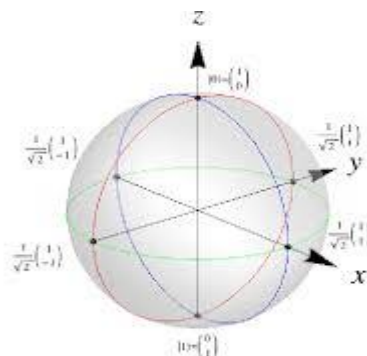
Ένα σφαιρικό σύστημα αναπαρίσταται στην εικόνα 14 συνήθως εμφανίζεται ως (r, φ, θ) και οι καρτεσιανές συντεταγμένες προκύπτουν από τις σφαιρικές συντεταγμένες με βάση τους τύπος :

$$x = r \cdot \sin\theta \cdot \cos\varphi$$

$$y = r \cdot \sin\theta \cdot \sin\varphi$$

$$z = r \cdot \cos\theta$$

Αξίζει να αναφερθεί πως το διάνυσμα κατάστασης ενός qubit απεικονίζεται σε μια διάφανη σφαίρα γνωστή και ως σφαίρα Bloch. Το διάνυσμα κατάστασης του qubit, η αλλιώς η κβαντική κατάσταση αναπαρίσταται με την βοήθεια των πολικών συντεταγμένων. Συγκεκριμένα με την βοήθεια των γωνιών θ , φ αναπαρίσταται στην σφαίρα. Παράδειγμα αναπαράστασης της κβαντικής κατάστασης στην σφαίρα Bloch βρίσκετε στην εικόνα 14.1



Εικόνα 14.1:Αναπαράσταση σφαίρας Bloch

(https://science.fandom.com/el/wiki/%CE%A3%CF%86%CE%B1%CE%AF%CF%81%CE%B1_Bloch)

2.3 Εισαγωγή στην Python

Η Python είναι μια από τις πιο γνωστές αντικειμενοστραφές γλώσσες προγραμματισμού υψηλού επιπέδου (high level programming language), δηλαδή γλώσσα προγραμματισμού που έχει σχεδιαστεί για να είναι κατανοητή και από τον άνθρωπο και από τον υπολογιστή. Περιέχει διερμηνέα (interpreter) και όχι μεταγλωττιστή. Ο διερμηνέας σε αντίθεση με τον μεταγλωττιστή εκτελεί τον κώδικα γραμμή προς γραμμή.

Χρησιμοποιείται στα σχολεία ως μια εισαγωγή στον προγραμματισμό καθώς είναι από τις πιο εύκολες γλώσσες προγραμματισμού και είναι φιλική και για κάποιον που γνωρίζει προγραμματισμό και για κάποιον που δεν γνωρίζει προγραμματισμό.

Είναι μια γλώσσα γενικής χρήσης (general-purpose language), δηλαδή μπορεί να χρησιμοποιηθεί για την επίλυση διαφόρων προβλημάτων

όπως να συμβάλει για την δημιουργία μιας ιστοσελίδας, για την ανάπτυξη λογισμικού κλπ., δεν ειδικεύεται αποκλειστικά σε συγκεκριμένα προβλήματα.

Επίσης η Python είναι μια γλώσσα ανοιχτού κώδικα (open source), διανέμεται δωρεάν και δεν έχει κάποιο κόστος, ακόμα περιέχει πληθώρα βιβλιοθηκών δίνοντας με αυτόν τον τρόπο την δυνατότητα στους χρήστες να επεκτείνουν και να βελτιώσουν την δουλειά τους.

2.3.1 Εφαρμογές και προγράμματα της Python

Η Python όπως προαναφέρθηκε εφαρμόζεται σε διάφορους τομείς όπως:

- **Ανάλυση Δεμένων και μηχανική μάθηση:** Αποτελεί βασικό στοιχείο στην επιστήμη των δεδομένων καθώς μέσα από τις βιβλιοθήκες και τα εργαλεία που περιέχει δίνει την δυνατότητα στους αναλυτές δεδομένων να διεξάγουν και να απεικονίζουν τα στατιστικά δεδομένα με διάφορων ειδών γραφήματα όπως γράφημα πίτας ή ραβδόγραμμα. Επίσης μέσω της Python οι αναλυτές έχουν την ικανότητα να χρησιμοποιούν αλγόριθμους μηχανικής μάθησης για καλύτερα αποτελέσματα και πιο γρήγορα.

- **Ανάπτυξη ιστοσελίδων:** Στην ανάπτυξη ιστοσελίδων η Python χρησιμοποιείται κυρίως στο back-end της ιστοσελίδας . Δηλαδή στο μέρος στο οποίο ο χρήστης δεν μπορεί να δει και δεν ξέρει τι γίνεται. Σε αυτό το κομμάτι ο σκοπός των προγραμματιστών (back-end developers)που δουλεύουν την ‘πίσω μεριά’ της σελίδας με python είναι η αποστολή δεδομένων από και προς τον διακομιστή(server). Ακόμα ο ρόλος τους είναι δρομολόγηση των URL(Uniform Resource Locator) δηλαδή των ηλεκτρονικών διευθύνσεων των ιστοσελίδων για την διατήρηση της ασφάλειας των χρηστών. Web frameworks τα οποία μπορούν να χρησιμοποιηθούν στην ανάπτυξη ιστοσελίδας και είναι γραμμένα σε python είναι τα Django, Flask
- **Δοκιμή Λογικού**(Software Testing):Σε αυτό το κομμάτι η χρήση της Python μέσα από διάφορα εργαλεία βοηθάει στην δοκιμή, αντιμετώπιση σφαλμάτων σε διάφορες εφαρμογές οι οποίες θα βγουν στο εμπόριο.
- **Scripting:** Python χρησιμοποιείται συχνά ως γλώσσα σεναρίων, λόγω της ευέλικτης σύνταξης της επιτρέπει στους προγραμματιστές να γραφούν μικρότερα προγράμματα.

Τα προγράμματα αυτά δεν αποτελούν κάποια κλάση η συνάρτηση όμως μπορούν να χρησιμοποιηθούν για να βελτιώσουν το αρχικό πρόγραμμα.

Ένας από το πολλούς τρόπους εκτέλεσης των προγραμμάτων στην Python γίνεται μέσα από το Jupyter Notebook.

Οι λόγοι που προτιμάτε το jupyter notebook είναι πως αρχικά περιέχει ένα φιλικό προς τον χρήστη περιβάλλον, το οποίο δίνει την δυνατότητα εκτέλεσης τμημάτων κώδικα και όχι ολόκληρου του κώδικα απαραίτητα. Το οποίο βοηθάει τον προγραμματιστή να εντοπίσει πιο εύκολα το σφάλμα στον κώδικα του όταν αυτός είναι αντιμέτωπος με έναν μεγάλο κώδικα.

Ένα άλλο χαρακτηριστικό που έχει το jupyter notebook είναι προσφέρει προγραμματισμό ανοιχτού κώδικα με σκοπό έτσι ώστε οι χρήστες να μπορούν συνεργάζονται μεταξύ τους και να συμμετέχουν και σε projects μεγάλων εταιριών. Κάποια παραδείγματα στην γλώσσα Python με την βοήθεια του jupyter notebook φαίνονται στις παρακάτω εικόνες, αρχικά εμφανίζεται ο πολλαπλασιασμός δύο πινάκων ο οποίος αναφέρθηκε και νωρίτερα, και στην συνέχεια ακολουθεί το τανυστικό γινόμενο, και μενού σχετικά με πράξεις μιγαδικών αριθμών καθώς και μενού με μετατροπή συντεταγμένων

2.3.1.1 Πολλαπλασιασμός πινάκων :

```
1 # Example 2
2 import random
3 N = int(input("Εισαγωγή Διαστάσεων :"))
4 M = int(input("Εισαγωγή Διαστάσεων :"))
5 K = int(input("Εισαγωγή Διαστάσεων :"))
6
7 a = [0] * N
8 b = []
9 c = []
10 print("Πίνακας A")
11
12 for i in range(N):
13     a[i] = ([i] * K)
14     for j in range(K):
15         a[i][j] = random.randint(0,5)
16
17 #
18     a[i][j] = random.randint(0,5)
19     print(a[i][j] , end = " ")
20 print()
21 print("Πίνακας B")
22
23 for i in range(K):
24     b.append([i]*M)
25     for j in range(M):
26         b[i][j] = random.randint(0,5)
27         print(b[i][j] , end = " ")
28     print()
29 print()
30 print("Πίνακας Γ")
31 for i in range(N):
32     for j in range(M):
33         total = 0
34         for k in range(K):
35             total += a[i][k] * b[k][j]
36         c.append(total)
37         print(total , end = " ")
38     print()
39 print()
```

Εικόνα 15: Πολλαπλασιασμός Πινάκων

Ο παραπάνω κώδικας λειτουργεί ως εξής:

Γραμμή 3-5: Εισαγωγή διαστάσεων των πινάκων

Γραμμή 7-9: Δήλωση πινάκων

Γραμμή 11-17: Δημιουργία Πρώτου πίνακα

Γραμμή 19-24: Δημιουργία δεύτερου πίνακα

Γραμμή 26-33: Πολλαπλασιασμός Πινάκων

Το αποτέλεσμα από τον κώδικα στην εικόνα 15 είναι:

```
Εισαγωγή Διαστάσεων :2
Εισαγωγή Διαστάσεων :3
Εισαγωγή Διαστάσεων :2
Πίνακας A
3 2
3 2

Πίνακας B
0 2 0
5 1 5

Πίνακας Γ
10 8 10
10 8 10
```

Εικόνα 16: Αποτελέσματα από πολλαπλασιασμό πινάκων(κεντρο με βαση word)

2.3.2.2 Τανυστικό Γινόμενο

Το τανυστικό γινόμενο είναι βασική πράξη στην κβαντική υπολογιστική, πραγματοποιείται ανάμεσα σε πύλες που είναι παράλληλες η μία με την άλλη και χρησιμοποιείται και για την δημιουργία καταχωρητών που περιλαμβάνουν πάνω από ένα qubit.

Ο κώδικάς του τανυστικού γινομένου γραμμένο σε python αναπαριστάτε στις εικόνες 17 και εικόνα 18:

```
1 import random
2 if __name__ == "__main__":
3     ra = 2
4     ca = 2
5
6     rb = 2
7     cb = 2
8     A = []
9     B = []
10    C = []
11
12    print("Πινάκας A:")
13    for i in range(ra):
14        A.append([])
15        for j in range(ca):
16            A[i].append(j)
17            A[i][j] = int(input(f"A[{i+1},{j+1}]:"))
18
19    print(A)
20
21    print("Πινάκας B:")
22    for k in range(rb):
23        B.append([])
24
25        for l in range(cb):
26            B[k].append(l)
27            B[k][l] = int(input(f"B[{k+1},{l+1}]:"))
28        # print("\n ")
29    print(B)
30    print("\n ")
31
```

Εικόνα 17: Τανυστικό γινόμενο

Γραμμή 1-2: Κύρια συνάρτηση

Γραμμή 3- 10: Εισαγωγή διαστάσεων των πινάκων A,B και αρχικοποίηση των πινάκων A,B,C

Γραμμή 12-20: Εισαγωγή στοιχείων στον πίνακα A και εκτύπωση του πίνακα

Γραμμή 22-30: Εισαγωγή στοιχείων στον πίνακα B και εκτύπωση του πίνακα

```
33 C = []
34 print("Πινάκας C")
35 for i in range(ra):
36     for k in range(ca):
37         start_row = i * rb + k
38         C.append([start_row])
39
40     for j in range(rb):
41         for l in range(cb):
42             start_column = j * rb + l
43             C[start_row].append(start_column)
44             C[start_row][start_column] = A[i][j] * B[k][l]
45             print(C[start_row][start_column], end=" ")
46     print()
47
```

Εικόνα 18: Συνέχεια τανυστικού γινομένου

Γραμμή 35-38: Με βάση το μήκος του πρώτου πίνακα το κάθε στοιχείο των γραμμών του τρίτου πίνακα θα προκύπτει από τον πολλαπλασιασμό του κάθε στοιχείου των γραμμών του πρώτου πίνακα με όλα τα στοιχεία της γραμμής του δεύτερου πίνακα μετατοπισμένα κατά 1 κάθε φορά.

Γραμμή 40-43: Με βάση το μήκος του δεύτερου πίνακα το κάθε στοιχείο των στηλών του τρίτου πίνακα θα προκύπτει από τον πολλαπλασιασμό του κάθε στοιχείου των στηλών του πρώτου πίνακα με όλα τα στοιχεία της στήλης του δεύτερου πίνακα μετατοπισμένα κατά 1 κάθε φορά.

Γραμμή 44: Δημιουργία πίνακα C

Γραμμή 45; Εμφάνιση πίνακα C

Το αποτέλεσμα από την εκτέλεση του κώδικα αναπαρίσται στην εικόνα 19 είναι:

```
Πίνακας A:  
A[1,1]:1  
A[1,2]:2  
A[2,1]:3  
A[2,2]:4  
[[1, 2], [3, 4]]
```

```
Πίνακας B:  
B[1,1]:5  
B[1,2]:6  
B[2,1]:7  
B[2,2]:8  
[[5, 6], [7, 8]]
```

```
Πίνακας C  
5 6 10 12  
7 8 14 16  
15 18 20 24  
21 24 28 32
```

Εικόνα 19: Αποτέλεσμα τανυστικού γινομένου

2.3.2.3:Μενού για πράξεις ανάμεσα σε δυο μιγαδικούς αριθμούς

Υστέρα ακολουθεί ένα μενού που εκτελεί πράξεις ανάμεσα σε μιγαδικούς αριθμούς ο κώδικας του αναπαριστάτε στις εικόνες

```

1 def menu(n1,n2):
2     print("Ενέλεξε: 1-->7")
3     print("1.Προσθεση \n2.Αφαίρεση \n3.Πολλαπλασιασμός \n4.Διαίρεση \n5.Δύναμη \n6.Μέτρο \n7.συζήτης ")
4     option = int(input("In Επιλογή:"))
5     if option == 1:
6         add = n1 + n2
7         return add
8     elif option == 2:
9         sub = n1 - n2
10        return sub
11    elif option == 3:
12        mul = n1 * n2
13        return mul
14    elif option == 4:
15        div = n1 / n2
16        return div
17    elif option == 5:
18        n3 = pow(n1,b)
19        n4 = pow(n2,d)
20        return n3,n4
21    #option 6 is for the meter
22    elif option == 6:
23        #Print the Meter of the First
24        n3 = abs(n1)
25        #Print the Meter of the Second
26        n4 = abs(n2)
27        return n3,n4
28    #option 7 is for Conjugate
29    elif option == 7:
30        #Print the Conjugate of the n1
31        n3 = n1.conjugate()
32        #Print the Conjugate of the n2
33        n4 = n2.conjugate()
34        return n3,n4
35    else:
36        print("Ξεκίνα Πάλι")
37
38 print ("Θες να ξεκινήσεις πάλι : NAI/OXI" )

```

Εικόνα 20: Μενού πράξεις
ανάμεσα σε μιγαδικούς αριθμούς

```

37
38 print ("Θες να ξεκινήσεις πάλι : NAI/OXI" )
39 answer = str(input("Απάντηση:"))
40 while answer == "NAI":
41
42     #Creation of the First Complex Number
43     #Real Part
44     a = int(input("Πραγματικό Μέρος:"))
45     #Imaginary Part
46     b = int(input("Φανταστικό Μέρος:"))
47     x= complex(a,b)
48
49
50     #Creation of The Second Complex Number
51     #Real Part
52     c = int(input("Πραγματικό Μέρος:"))
53     #Imaginary Part
54     d = int(input("Φανταστικό Μέρος:"))
55     y = complex(c,d)
56
57     print(f"\n Ο πρώτος μιγαδικός είναι:{x} \n ο δεύτερος μιγαδικός είναι:{y}\n")
58     numbers = menu(x,y)
59     print("\nΑποτέλεσμα: \n" + str(numbers))
60
61     print ("In Ξεκίνα Πάλι : NAI/OXI\n" )
62     answer = str(input("Απάντηση:"))
63
64

```

Εικόνα 21: Συνέχεια μενού

Γραμμή 1: Δημιουργία συνάρτησης που αναπαριστά το μενού, δέχεται ως είσοδο δύο μιγαδικούς αριθμούς

Γραμμή 2-3: Εμφάνιση των επιλογών ανάμεσα στις πράξεις που έχει ο χρήστης

Γραμμή 4: Ο χρήστης επιλέγει ποια πράξη θα πραγματοποιήσει η συνάρτηση μέσα από τον αριθμό που δίνει

Γραμμή 5-34: Γίνεται εκτέλεση των πράξεων αν ο χρήστης δώσει επιλογή από 1 μέχρι 7

Γραμμή 35: Αν ο χρήστης δώσει τιμή μεγαλύτερη του 7 ξεκινάει ξανά το μενού

Γραμμή 38-39: Γίνεται ερώτηση στον χρήστη αν θέλει να αρχίσει το μενού και ο χρήστης απαντάει ναι ή όχι

Γραμμή 40: Όσο ο χρήστης θέλει να αρχίζει το μενού τότε γίνεται εισαγωγή των μιγαδικών που θα χρησιμοποιηθούν στην συνάρτηση του μενού και ανάλογα με την επιλογή του χρήστη η συνάρτηση επιστρέφει την κατάλληλη πράξη

Γραμμή 44: Εισαγωγή πραγματικού μέρους του πρώτου μιγαδικού

Γραμμή 47: Ο πρώτος μιγαδικός

Γραμμή 52: Εισαγωγή πραγματικού μέρους του δεύτερου μιγαδικού

Γραμμή 54: Εισαγωγή φανταστικού μέρους του δεύτερου μιγαδικού

Γραμμή 55: Ο δεύτερος μιγαδικός

Γραμμή 57: Εμφάνιση των δύο μιγαδικών

Γραμμή 58: Εμφάνιση του μενού στον χρήστη

Γραμμή 59: Εμφάνιση του αποτελέσματος της πράξης

Γραμμή 60-61: Ερώτηση αν ο χρήστης θέλει να ανοίξει ξανά το μενού και απάντηση του χρήστη

Το αποτέλεσμα από την εκτέλεση του παρακάτω προγράμματος στις εικόνες 22 και 23 είναι :

```

Θες να ξεκινήσεις πάλι : NAI/OXI
Απάντηση:NAI
Πραγματικό Μέρος:2
Φανταστικό Μέρος:1
Πραγματικό Μέρος:3
Φανταστικό Μέρος:1

Ο πρώτος μιγαδικός είναι:(2+1j)
ο δεύτερος μιγαδικός είναι:(3+1j)

Επέλεξε: 1-->7
1.Πρόσθεση
2.Αφαίρεση
3.Πολλαπλασιασμός
4.Διαίρεση
5.Δύναμη
6.Μέτρο
7.συζήγής

Επιλογή:1

Αποτέλεσμα:
(5+2j)

Ξεκίνα Πάλι : NAI/OXI
Απάντηση:NAI

```

Εικόνα23: Μενού με επιλογή της πράξης της πρόσθεσης

Η εικόνα 23 απεικονίζει μια εκτέλεση του μενού στο οποίο έγινε επιλογή της πρόσθεσης και ο χρήστης θέλησε να εκτελέσει μετά ξανά το πρόγραμμα.

Για τις υπόλοιπες επιλογές το πρόγραμμά θα ήταν ως εξής:

Στην εικόνα 24 αναπαριστάτε το αποτέλεσμα εάν η επιλογή είναι αφαίρεση

```

Θες να ξεκινήσεις πάλι : NAI/OXI
Απάντηση:NAI
Πραγματικό Μέρος:2
Φανταστικό Μέρος:3
Πραγματικό Μέρος:4
Φανταστικό Μέρος:5

Ο πρώτος μιγαδικός είναι:(2+3j)
ο δεύτερος μιγαδικός είναι:(4+5j)

Επέλεξε: 1-->7
1.Πρόσθεση
2.Αφαίρεση
3.Πολλαπλασιασμός
4.Διαίρεση
5.Δύναμη
6.Μέτρο
7.συζήγής

Επιλογή:2

Αποτέλεσμα:
(-2-2j)

Ξεκίνα Πάλι : NAI/OXI

```

Εικόνα24: Μενού με επιλογή της πράξης της αφαίρεση

Στην εικόνα 25 αναπαριστάτε το αποτέλεσμα εάν η επιλογή είναι Πολλαπλασιασμός

Ξεκίνα Πάλι : ΝΑΙ/ΟΧΙ

Απάντηση:ΝΑΙ
 Πραγματικό Μέρος:2
 Φανταστικό Μέρος:5
 Πραγματικό Μέρος:6
 Φανταστικό Μέρος:7

ο πρώτος μιγαδικός είναι:(2+5j)
 ο δεύτερος μιγαδικός είναι:(6+7j)

Επέλεξε: 1-->7
 1.Πρόσθεση
 2.Αφαίρεση
 3.Πολλαπλασιασμός
 4.Διαίρεση
 5.Δύναμη
 6.Μέτρο
 7.συζηγής

Επιλογή:3

Αποτέλεσμα:
 (-23+44j)

Ξεκίνα Πάλι : ΝΑΙ/ΟΧΙ

Εικόνα 25: Μενού με επιλογή την πράξη πολλαπλασιασμού

Στην εικόνα 26 αναπαριστάτε το αποτέλεσμα εάν η επιλογή είναι Διαίρεση

Ξεκίνα Πάλι : ΝΑΙ/ΟΧΙ

Απάντηση:ΝΑΙ
 Πραγματικό Μέρος:8
 Φανταστικό Μέρος:9
 Πραγματικό Μέρος:1
 Φανταστικό Μέρος:2

ο πρώτος μιγαδικός είναι:(8+9j)
 ο δεύτερος μιγαδικός είναι:(1+2j)

Επέλεξε: 1-->7
 1.Πρόσθεση
 2.Αφαίρεση
 3.Πολλαπλασιασμός
 4.Διαίρεση
 5.Δύναμη
 6.Μέτρο
 7.συζηγής

Επιλογή:4

Αποτέλεσμα:
 (5.2-1.4j)

Ξεκίνα Πάλι : ΝΑΙ/ΟΧΙ

Εικόνα 26: Μενού με επιλογή την πράξη της διαίρεση

Στην εικόνα 27 αναπαριστάτε το αποτέλεσμα εάν η επιλογή είναι Δύναμη

Ξεκίνα Πάλι : NAI/OXI

Απάντηση: NAI
 Πραγματικό Μέρος: 1
 Φανταστικό Μέρος: 3
 Πραγματικό Μέρος: 2
 Φανταστικό Μέρος: 5

Ο πρώτος μιγαδικός είναι: $(1+3j)$
 ο δεύτερος μιγαδικός είναι: $(2+5j)$

Επέλεξε: 1-->7
 1. Πρόσθεση
 2. Αφαίρεση
 3. Πολλαπλασιασμός
 4. Διαίρεση
 5. Δύναμη
 6. Μέτρο
 7. συζυγής

Επιλογή: 5

Αποτέλεσμα:
 $((-26-18j), (4282-1475j))$

Ξεκίνα Πάλι : NAI/OXI

Εικόνα 27: Μενού με επιλογή την πράξη της δύναμης

Στην εικόνα αναπαριστάτε το αποτέλεσμα εάν η επιλογή είναι μέτρο

Θες να ξεκινήσεις πάλι : NAI/OXI

Απάντηση: NAI
 Πραγματικό Μέρος: 1
 Φανταστικό Μέρος: 2
 Πραγματικό Μέρος: 3
 Φανταστικό Μέρος: 4

Ο πρώτος μιγαδικός είναι: $(1+2j)$
 ο δεύτερος μιγαδικός είναι: $(3+4j)$

Επέλεξε: 1-->7
 1. Πρόσθεση
 2. Αφαίρεση
 3. Πολλαπλασιασμός
 4. Διαίρεση
 5. Δύναμη
 6. Μέτρο
 7. συζυγής

Επιλογή: 6

Αποτέλεσμα:
 $(2.23606797749979, 5.0)$

Ξεκίνα Πάλι : NAI/OXI

Εικόνα 28: Μενού με επιλογή της πράξη του μέτρου

Στην εικόνα 29 αναπαριστάτε το αποτέλεσμα εάν η επιλογή είναι συζυγής

```

Ξεκίνα Πάλι : NAI/OXI

Απάντηση: NAI
Πραγματικό Μέρος: 2
Φανταστικό Μέρος: 3
Πραγματικό Μέρος: 1
Φανταστικό Μέρος: 4

Ο πρώτος μιγαδικός είναι: (2+3j)
ο δεύτερος μιγαδικός είναι: (1+4j)

Επέλεξε: 1-->7
1. Πρόσθεση
2. Αφαίρεση
3. Πολλαπλασιασμός
4. Διαίρεση
5. Δύναμη
6. Μέτρο
7. συζήγης

Επιλογή: 7

Αποτέλεσμα:
((2-3j), (1-4j))

Ξεκίνα Πάλι : NAI/OXI

```

Εικόνα29: Μενού με επιλογή συζυγή

2.3.2.4 Μενού για μετατροπή συντεταγμένων

Υστέρα στις εικόνες 30,31,32 απεικονίζεται μενού που μετατρέπει συντεταγμένες ,από καρτεσιανές σε πολικές, από πολικές σε καρτεσιανές, από σφαιρικές σε πολικές και από πολικές σε σφαιρικές

```

1: 1
2: from cmath import phase, atan, acos, sqrt
3: # from numpy import rad2deg
4: from math import radians, degrees, cos, sin, e, sqrt
5: # import numpy as np
6: import math
7: from numpy import rad2deg
8: def menu():
9:     print("\n1.Turn Polar Coordinates To Cartesian")
10:    print("\n2.Turn Cartesian Coordinates To Polar")
11:    print("\n3 Turn Spherical Coordinates To Polar")
12:    print("\n4 Turn Cartesian Coordinates To Polar")
13:    choice = int(input("\n Input Option:"))
14:    return choice
15:
16:
17: # Polar_Form = r*math.exp(y*j)
18: # Polar_Form = r*math.exp(b*j)
19:
20: #Create The Complex Number
21: x = int(input("Give Numbers For The Real Part Of The Complex:"))
22: y = int(input("Give Numbers For The Imaginary Part Of The Complex:"))
23: z = complex(x,y)
24: #Calculate The Meter Of The Complex
25: r = abs(z)
26: # r(meter/radian) is the distance of the reference point from the beginning of the axes
27: #With the help of r (meter/radian) and the angle φ(phase) we can determine complex numbers
28:
29: #Calculate The angle
30: print("Input Your Angle")
31: print("\n Input Degrees or Radians")
32: print("\nAngle Choices: \n1.Radians \n2.Degrees")
33: choice = int(input("\nInput Your Choice:"))
34: if choice == 1:
35:     #The angle is measured in radians
36:     #We let the user enter float numbers and convert them with radians()

```

Εικόνα 30: Μενού μετατροπής συντεταγμένων

Γραμμή 2-6:Εισαγωγή Βιβλιοθηκών

Γραμμή 8-14:Μένου που δίνει της επιλογές του χρήστη και τον ρωτάει ποια θα επιλέξει

Γραμμή 21-22: Δημιουργία πραγματικού και φανταστικού μέρους τους μιγαδικού αριθμού

Γραμμή 23: Δημιουργία μιγαδικού αριθμού

Γραμμή 25: Δημιουργία μέτρου του μιγαδικού

Γραμμή 30-33: Εισαγωγή γωνίας σε ακτίνια ή μοίρες

```
33 choice = int(input("Give your choice: "))
34 if choice == 1:
35     #The angle is measured in radians
36     #We let the user enter float numbers and convert them with radians()
37     ϕ = radians(float(input("Input Numbers:")))
38     θ = radians(float(input("Input Numbers:")))
39     #θ is the angle between r and z axis
40     ϕ = phase(z) # ϕ ∈ [0, 2π]
41     ϕ = rad2deg(ϕ)
42
43     # ϕ is the angle between r and xx'
44
45 elif choice == 2:
46     #Convert radians to degrees with rad2deg
47     ϕ = degrees(float(input("Input Numbers:")))
48     θ = degrees(float(input("Input Numbers:")))
49     ϕ = phase(z) # ϕ ∈ [0, 2π]
50
51
52 #Start The Menu
53 print("Start The Menu")
54 #Give Your Choice
55 option = menu()
56 answer = "Y"
57 while (option <= 4 and answer == "Y"):
58     if option == 1:
59         #For the conversion from Polar To Cartesian Form the z is: z = r*(cosα + isinβ)
60         #The new cartesian form of the 'Polar' changes is Polar_Form = r*math.exp(b*i)
61
62         #For the conversion from Polar To Cartesian Form the new a, b are:
63         x = r * cos(ϕ) # a = r*cos(ϕ)
64         y = r * sin(ϕ) # b = r*sin(ϕ)
65
66         print("Convert From Polar Coordinates: %.3f , %.3f \n To Cartesian Coordinates: %.3f , %.3f " % (r, ϕ, x, y))
67     if option == 2:
68         #Cartesian coordinates X = a , Y = bi
69         #Cartesian Form Z = a + b*i
70         #Python Cartesian Form Z = complex(a,b)
71
72         # Polar Form = r*math.exp(b*i)
```

Εικόνα 31: Συνέχεια μενού μετατροπής συντεταγμένων

Γραμμή 34: Επιλογή γωνίας σε ακτίνια

Γραμμή 37-38: Υπολογισμός αζιμούθιου και πολικής γωνίας

Γραμμή 40: Υπολογισμός φάσης

Γραμμή 41: Μετατροπή ακτινίων σε μοίρες

Γραμμή 45-49: Υπολογισμός φάσης όταν οι γωνίες είναι σε μοίρες

Γραμμή 53-55: Έναρξη μενού

Γραμμή 56-57: Εκτέλεση μενού όσο ο χρήστης απαντάει ναι

Γραμμή 58-64: Μετατροπή από πολικές σε καρτεσιανές συντεταγμένες

```

70 #Python Cartesian Form Z = complex(a,b)
71
72 # Polar Form =r*math.exp(b*j)
73 print("Convert From Cartesian Coordinates: %.f , %.f \n To Polar Coordinates: %.3f , %.3f \n" %(x,y,r,phi))
74
75 #Polar Coordinates r = sqrt(pow(x,2) + pow(y,2))
76 #Python Polar Coordinates r = abs(complex)
77 #Python Polar Coordinates phi = phase(complex)
78 elif option == 3:
79     #For The Spherical Coordinates To Cartesian Coordinates we calculate x,y,z
80     x = r * sin(phi) * cos(theta) #u = r*cos(phi)
81     y = r * sin(phi) * sin(theta)
82     z = r * cos(phi)
83     print("Convert From Spherical Coordinates :%.3f , %.3f , %.3f \n To Cartesian Coordinates: %.3f , %.3f , %.3f \n" %(phi,theta,r,x,y,z))
84
85 elif option == 4:
86     r = sqrt(pow(x,2) + (pow(y,2) + ( pow(z,2))
87     phi = atan(y/x)
88     theta = acos(z/r)
89     print("Convert From Cartesian Coordinates :%.3f , %.3f , %.3f Convert From Spherical Coordinates: %.3f , %.3f , %.3f \n" %(phi,theta,r,x,y,z))
90
91 print("Do You Wanna Start The Menu Again :Y/N \n")
92 answer = str(input(" Your Answer:\n"))
93
94 print("Exit")
95

```

Εικόνα 32:Συνέχεια μενού μετατροπής συντεταγμένων

Γραμμή 64-77: Μετατροπή από Καρτεσιανές σε πολικές συντεταγμένες

Γραμμή 78-83: Μετατροπή από σφαιρικές σε καρτεσιανές

Γραμμή 85-89: Μετατροπή από καρτεσιανές σε σφαιρικές

Γραμμή 91-92:Ερώτηση στον χρήστη αν θέλει να ξεκινήσει από την αρχή

Με την εκτέλεση του προγράμματος τα αποτελέσματα θα είναι τα εξής και απεικονίζονται στην εικόνα :

```

Give Numbers For The Real Part Of The Complex:2
Give Numbers For The Imaginary Part Of The Complex:3
Input Your Angle

Input Degrees or Radians

Angle Choices:
1.Radians
2.Degrees

Input Your Choice:1
Input Numbers:23
Input Numbers:34
Start The Menu

1.Turn Polar Coordinates To Cartesian
2.Turn Cartesian Coordinates To Polar
3.Turn Spherical Coordinates To Polar
4.Turn Cartesian Coordinates To Polar

Input Option:1
Convert From Polar Coordinates: 3.606 , 56.310
To Cartesian Coordinates:3.503 , -0.853
Do You Wanna Start The Menu Again :Y/N

Your Answer:
n
Exit

```

3.0 Κβαντική υπολογιστική

Όπως αναφέρθηκε και στα προηγούμενα κεφάλαια, ένα κβαντικός υπολογιστής διαφέρει αρκετά από ένα κλασικό υπολογιστή λόγω των ιδιοτήτων του, μέσα από την κβαντική παραλληλία έχει την δυνατότητα να είναι ταχύτερος από έναν κλασικό υπολογιστή. Σε έναν κβαντικό υπολογιστή και συνεπώς στην κβαντική υπολογιστική η κβαντική πληροφορία παίρνει ταυτόχρονα δύο τιμές και 0 και 1 και

ονομάζεται qubit, ενώ η συμπεριφορά ενός qubit καθορίζεται από ένα σύνολο κβαντικών πυλών οι οποίες αναπαριστώνται σε κυκλώματα όπως οι κλασικές πύλες.

3.1 Qubits

Πως ο στροβιλισμός ενός κέρματός στον αέρα μπορεί να θεωρηθεί Qubit?

Αρχικά ένα κλασικό bit το οποίο έχει ως τιμή 0 ή 1 μπορεί να αναπαρασταθεί ως ένα κλασικό κέρμα. Στο κλασικό κέρμα η κατάσταση 0 του νομίσματος αναπαριστά την πλευρά που θεωρείτε η ‘κορώνα’ του κέρματος και η κατάσταση 1 αναπαριστά την πλευρά που θεωρείτε τα ‘γράμματα του κέρματος’. Όταν γίνει στροβιλισμός του κέρματος τότε οι πιθανότητες το κέρμα να προσγειωθεί σε κάθε πλευρά είναι ίσες δηλαδή 50-50. Σε ένα τίμιο κέρμα πάντα οι πιθανότητες εμφάνισης της κάθε κατάστασης είναι ίδιες αν γίνει ρίξει δύο νομισμάτων τότε δημιουργούνται τα εξής τέσσερα σύνολα:

- (0,0), (0,1), (1,0), (1,1)
- Καθώς πριν η πιθανότητα εμφάνισης κάθε κατάστασης ήταν $\frac{1}{2}$ δηλαδή 50 % για κάθε κατάσταση, τώρα με τα δύο νομίσματα θα ισχύει το εξής:
$$\frac{1}{2} * \frac{1}{2} = \frac{1}{4}$$
 οπότε η πιθανότητα εμφάνισης της κάθε κατάστασης είναι 25%

Προηγουμένως έγινε αναφορά στην κατάσταση στην οποία θα βρεθεί το κέρμα όταν σταματήσει τον στροβιλισμό,

κατά την διάρκεια του στροβιλισμού η κατάσταση του κέρματος δεν είναι γνωστή οπότε είναι την ίδια χρονική περίοδο και κορώνα και γράμματα.

Δηλαδή βρίσκεται το κέρμα κατά την διάρκεια του στροβιλισμού σε κατάσταση υπέρθεσης όπως ένα qubit

Τι είναι όμως ένα qubit?

Ως qubit(quantum bit) ορίζεται η μονάδα αποθήκευσης των κβαντικών πληροφοριών σε ένα κβαντικό υπολογιστή. Ένα qubit μπορεί να παίρνει δύο διαφορετικές τιμές μπορεί να παίρνει την τιμή 0 και την τιμή 1 ή οποιαδήποτε τιμή από το 0 μέχρι το 1. Το κάθε qubit μπορεί να αναπαρασταθεί ως ένα διάνυσμα κατάστασης και οι τιμές που περιέχει το συγκεκριμένο διάνυσμα αναπαριστούν τις πιθανότητες εμφάνισης κάθε κβαντικής κατάστασης. Οπότε κάθε qubit μπορεί να έχει την εξής μορφή:

- $|\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$,όπου a, b μιγαδικοί αριθμοί και πιθανότητες εμφάνισης κάθε κατάστασης

Ποιες είναι οι βασικές κβαντικές καταστάσεις ?

- $|0\rangle, |1\rangle$ όπου $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ και $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
- Τα διανύσματα κάθε μίας κατάστασης είναι γνωστά ως διανύσματα κατάστασης

Οπότε κάθε qubit με βάση τις παραπάνω καταστάσεις μπορεί να γραφεί ως γραμμικός συνδυασμός τους συνδυασμός με μορφή την εξής:

$$|\psi\rangle = a \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \end{pmatrix} = a * |0\rangle + b * |1\rangle$$

Σύμφωνα με των παραπάνω τύπο η πιθανότητα το qubit να βρίσκεται στην κατάσταση $|0\rangle$ είναι $|a|^2$, ενώ η πιθανότητα το qubit να βρίσκεται στην κατάσταση $|1\rangle$ είναι $|b|^2$. Καθώς τα απόλυτα τετράγωνά αντιστοιχούν σε πλάτη πιθανοτήτων τότε θα πρέπει το άθροισμα τους να είναι ίσο με το 1 έτσι ώστε η κβαντική κατάσταση να μπορεί να θεωρηθεί ως αποδεκτή. Δηλαδή πρέπει να ισχύει

- $|a|^2 + |b|^2 = 1$ ώστε $|\psi\rangle = a * |0\rangle + b * |1\rangle$ να είναι αποδεκτή κβαντική κατάσταση

- Σε περίπτωση που το $a = 0$ τότε το qubit θα βρίσκεται μόνο στη κατάσταση $|1\rangle$
- Και αντίστοιχα το qubit θα βρίσκεται στην κατάσταση $|0\rangle$ εάν $b = 0$.
- Σε περίπτωση που ούτε το a ούτε το b είναι μηδέν τότε το qubit βρίσκεται σε υπέρθεση.
- Όταν γίνει μέτρηση ενός qubit εάν αυτό βρίσκεται σε υπέρθεση τότε θα διαταράξει και θα σπάσει την υπέρθεση καθώς το qubit θα καταλήξει σε μία κατάσταση 0 ή 1.

Έστω ότι $|\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} = a\begin{pmatrix} 1 \\ 0 \end{pmatrix} + b\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ όπου $b = 0.3$, $a = 0.4$ δηλαδή

$|\psi\rangle = 0.3\begin{pmatrix} 1 \\ 0 \end{pmatrix} + 0.4\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ η παραπάνω κατάσταση δεν είναι κβαντικά αποδεκτή καθώς το άθροισμα των τετραγώνων από τα πλάτη πιθανοτήτων δεν είναι ίσα με 1. Δηλαδή $(0.3)^2 + (0.4)^2 = 0.09 + 0.16 = 0.25 < 1$ οπότε η παραπάνω κατάσταση δεν μπορεί να αναπαρασταθεί ως γραμμικός συνδυασμός των δύο διανυσμάτων

Στην συνέχεια θα γίνει αναπαράσταση της ρίψης ενός τίμιου νομίσματος αλλά με κβαντική μορφή, όπου η πιθανότητα εμφάνισης

της κατάστασης 'κορώνας' και 'γράμματα' είναι ίδια, και αντιστοιχεί σε 50%. Δηλαδή ισχύει $\text{Πιθανότητα_Κορώνας} = \left(\frac{1}{\sqrt{2}}\right)^2 = 0.5$, οπότε $\text{Κορώνα} = \frac{1}{\sqrt{2}}$

και $\text{Πιθανότητα_Γραμμάτων} = \left(\frac{1}{\sqrt{2}}\right)^2 = 0.5$, οπότε $\text{Γράμματα} = \left(\frac{1}{\sqrt{2}}\right)^2 = 0.5$ καθώς $\text{Πιθανότητα_Γραμμάτων} + \text{Πιθανότητα_Κορώνας} = 0.5 + 0.5 = 1$

Οπότε μια κβαντική κατάσταση της μορφής $|\text{Νόμισμα}\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} \text{Κορώνα} \\ 0 \end{pmatrix} + \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ \text{Γράμματα} \end{pmatrix}$
 $= \frac{1}{\sqrt{2}}\begin{pmatrix} \text{Κορώνα} \\ \text{Γράμματα} \end{pmatrix} = \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}|\text{Γράμματα}\rangle$ είναι κβαντικά αποδεκτή.

Υστέρα με την ρίψη δύο ζαριών οι συνδυασμοί που θα προκύψουν είναι οι εξής: Κορώνα-Γράμματα, Κορώνα-Κορώνα, Γράμματα-Κορώνα, Γράμματα-Γράμματα. Οπού κάθε συνδυασμός θα έχει ίδια πιθανότητα εμφάνισης δηλαδή ισχύει:

$$\begin{aligned} \text{Πιθανότητα_Κορώνας_Γράμματα} &= \left(\frac{1}{2}\right)^2 = \text{Πιθανότητα_Κορώνας_Κορρώνα} = \\ \left(\frac{1}{2}\right)^2 &= \text{Πιθανότητα_Γράμματα_Κορώνας} = \left(\frac{1}{2}\right)^2 = \text{Πιθανότητα_Γράμματα_Γράμ-} \\ \text{ματα} &= \left(\frac{1}{2}\right)^2 = \frac{1}{4} \end{aligned}$$

Ακόμα $\text{Πιθανότητα_Κορώνας_Γράμματα} + \text{Πιθανότητα_Κορώνας_Κορρώνα} +$
 $\text{Πιθανότητα_Γράμματα_Κορώνας} + \text{Πιθανότητα_Γράμματα_Γράμματα} = 1$

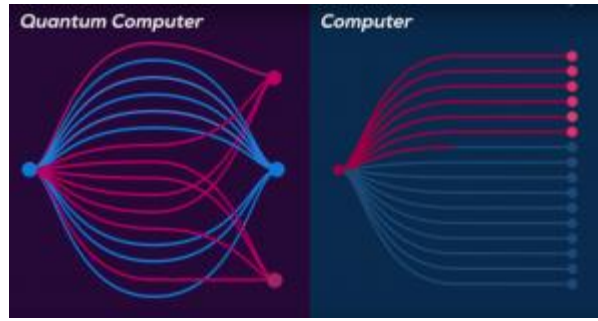
Οπότε η κβαντική κατάσταση που θα προκύψει είναι αποδεκτή και είναι η εξής:

$$\begin{aligned} |N1\rangle|N2\rangle &= \frac{1}{2} \begin{pmatrix} \text{Κορώνα} - \text{Κορώνα} \\ 0 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 0 \\ \text{Κορώνα} - \text{Γράμματα} \\ 0 \\ 0 \end{pmatrix} + \\ &\frac{1}{2} \begin{pmatrix} 0 \\ 0 \\ \text{Γράμματα} - \text{Κορώνα} \\ 0 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 0 \\ 0 \\ 0 \\ \text{Γράμματα} - \text{Γράμματα} \end{pmatrix} = \\ &\frac{1}{2} \begin{pmatrix} \text{Κορώνα} - \text{Κορώνα} \\ \text{Κορώνα} - \text{Γράμματα} \\ \text{Γράμματα} - \text{Κορώνα} \\ \text{Γράμματα} - \text{Γράμματα} \end{pmatrix} = \frac{1}{2} (|\text{Κορώνα} - \text{Κορώνα}\rangle + \\ &|\text{Κορώνα}\text{Γράμματα}\rangle + |\text{Γράμματα} - \text{Κορώνα}\rangle + |\text{Γράμματα} - \\ &\text{Γράμματα}\rangle) \end{aligned}$$

Ένας άλλος τρόπος αναπαράστασης της κβαντικής κατάστασης των 2 νομισμάτων είναι ο εξής:

$$\begin{aligned} |N1\rangle|N2\rangle &= \frac{1}{2} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{2} (|00\rangle + |01\rangle + |10\rangle + \\ &|11\rangle) \end{aligned}$$

Για την πλήρη κατανόηση του τρόπου λειτουργίας ενός qubit ακολουθεί η εικόνα 34 που συγκρίνει τον τρόπο που δουλεύει ένα qubit σε ένα κβαντικό υπολογιστή σε σχέση με ένα bit σε ένα κλασικό υπολογιστή. Στον κβαντικό υπολογιστή οι ροζ γραμμές αναπαριστούν την μια κατάσταση και οι μπλε την άλλη



Εικόνα34: Σύγκριση τρόπου λειτουργίας qubit με bit (What is a Qubit? gammasolutions.com)

3.1.1 Bracket

Κάθε κβαντική κατάσταση συμβολίζεται ως $|\psi\rangle$, τα εξής σύμβολα $| \text{και} \rangle$ αναπαριστούν το λεγόμενο ‘κετ’, και δηλώνει την διανυσματική μορφή των qubits . Το κέτ ή αλλιώς η σημειογραφία κέτ (Ket Notation) αποτελεί μέρος της ‘σημειογραφίας Dirac’ ή αλλιώς ‘σημειογραφία bra-ket’. Η εφεύρεση της οποίας προήρθε από τον φυσικό Paul Dirac. Το ket είναι ένα διάνυσμα $N \times 1$ (όπου N είναι ο αριθμός των γραμμών) σε ένα ορθοκανονικό σύστημα αξόνων και αποτελείται από μιγαδικούς αριθμούς. Η γενική μορφή του ket είναι η εξής:

$$\bullet \quad |a\rangle = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ \vdots \\ a_{n-1} \end{pmatrix} \quad \text{όπου } a = x + y * i$$

Πέρα από το ket υπάρχει και το ‘bra’ το οποίο αποτελεί και αυτό μέρος της ‘σημειογραφίας bra-ket’, αναπαρίσταται με τα σύμβολα $\langle \dots |$ και $|$ είναι ένας $1 \times N$

πίνακας (όπου είναι ο αριθμός των στηλών) και αποτελεί τον ανάστροφο πίνακα του ket καθώς περιέχει όλους τους συζυγείς μιγαδικούς αριθμούς του ket η μορφή του bra είναι η εξής:

$$\bullet \quad \langle a| = (a_0^*, a_1^* \dots a_{n-1}^*) \quad a^* = x - y * i$$

Τα διανύσματα των bra και ket μέσω του εσωτερικού γινομένου δημιουργούν ένα διαφορετικό διανυσματικό χώρο με περισσότερες από 2 διαστάσεις. Δηλαδή έναν διανυσματικό χώρο από που αποτελείται από τρεις διαστάσεις και πάνω και στον οποίο μπορεί να αναπαρασταθούν τα διανύσματα κατάστασης. Ο χώρος αυτός ονομάζεται χώρος Hilbert και πήρε την ονομασία του από τον David Hilbert

Τι είναι όμως το εσωτερικό γινόμενο;

Είναι η εξής πράξη: $\langle a|b \rangle$ όπου $\langle a| = (a_0^*, a_1^* \dots a_{n-1}^*)^*$

$$\begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} = a_0^* *$$

$$b_0 + a_1^* * b_1 + \dots + a_{n-1}^* * b_{n-1}$$

Επίσης αν υπάρχουν οι εξής κβαντικές καταστάσεις $|0\rangle$ και $|1\rangle$ τότε το εσωτερικό γινόμενο θα είναι το εξής :

$$\langle 0|0 \rangle = (1 \ 0)^* \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1*1 + 0*0 = 1$$

$$\langle 0|1 \rangle = (1 \ 0)^* \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 1*0 + 0*1 = 0$$

$$\langle 1|0 \rangle = (0 \ 1)^* \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0*1 + 1*0 = 0$$

$$\langle 1|1 \rangle = (0 \ 1)^* \begin{pmatrix} 0 \\ 1 \end{pmatrix} = 0*0 + 1*1 = 1$$

Όπου $|0\rangle$ και $|1\rangle$ είναι κάθετα μεταξύ τους και γραμμικός ανεξάρτητα. Πέρα του εσωτερικού γινομένου στο χώρο Hilbert υπάρχουν

και άλλα δύο γινόμενα το εξωτερικό γινόμενο που είναι ένα κλασικό γινόμενο δύο πινάκων και το τανυστικό γινόμενο το οποίο αναφέρθηκε και στο προηγούμενο κεφάλαιο.

Η μορφή του εσωτερικού γινομένου είναι η εξής:

$$\bullet \quad |a\rangle\langle b| = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} * (b_0^*, b_1^* \dots b_{n-1}^*)$$

Ενώ το τανυστικό γινόμενο το οποίο είναι αρκετά χρήσιμο στην κβαντική υπολογιστική καθώς καθορίζει έναν κβαντικό καταχωρητή έχει την εξής μορφή :

$$\bullet \quad |a\rangle \otimes |b\rangle = |ab\rangle = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \otimes \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} a_0 * \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} \\ a_1 * \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} \\ \vdots \\ a_{n-1} * \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} \end{pmatrix}$$

3.2 Κβαντικοί καταχωρητές

Ένας κλασικός υπολογιστής λειτουργεί πιο γρήγορα μεταφέροντας τα δεδομένα από την κεντρική μνήμη σε έναν καταχωρητή, ύστερα τα δεδομένα μετά την επεξεργασία που θα υποστούν επιστρέφονται ξανά στην κύρια μνήμη.

Οπότε ως καταχωρητής ορίζεται η μνήμη του υπολογιστή στην οποία πραγματοποιείται η προσωρινή αποθήκευση των δεδομένων με αποτέλεσμα την ταχύτερη εκτέλεση των προγραμμάτων.

Τα δεδομένα αποθηκεύονται στον καταχωρητή ενός κλασικού υπολογιστή ως Bit (0 ή 1).

Αντίθετα με ένα κλασικό υπολογιστή σε έναν κβαντικό υπολογιστή ένας καταχωρητής περιέχει την κατάσταση ενός *qubit* ή την κατάσταση που προκύπτει από τον συνδυασμό των *qubits*. Οπότε ένας κβαντικός καταχωρητής αντί για 2 bit που περιέχει ένας κλασικός καταχωρητής, θα περιέχει 2^n bits.

Για παράδειγμα ένας καταχωρητής ενός qubit που μπορεί να είναι ο εξής:

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

Ενώ στην περίπτωση που ο καταχωρητής περιέχει πάνω από 1 qubit τότε το αποτέλεσμα του καταχωρητή που σαν είσοδο έχει τις καταστάσεις 2 ή περισσότερων qubit καθορίζεται από τον συνδυασμό των qubit.

Δηλαδή από την πράξη του τανυστικού γινομένου ($\otimes$) ανάμεσα στα διανύσματα τους.

Παραδείγματα καταχωρητών που περιέχουν πάνω από 1 qubit είναι τα εξής :

- Καταχωρητές που περιέχουν 2 qubits :

$$|1\rangle \otimes |0\rangle = |10\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 * \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 1 * \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}. \text{ Για να είναι αποδεκτή}$$

η κατάσταση πρέπει $(0^2 + 0^2 + 1^2 + 0^2) = 1$, που ισχύει οπότε η κατάσταση είναι αποδεκτή

$$|0\rangle \otimes |0\rangle = |00\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 * \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0 * \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \text{ Για να είναι αποδεκτή η κα-}$$

τάσταση πρέπει $(1^2 + 0^2 + 0^2 + 0^2) = 1$, που ισχύει οπότε η κατάσταση είναι αποδεκτή

$$|0\rangle \otimes |1\rangle = |01\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}. \text{ Για να είναι αποδεκτή η κα-}$$

τάσταση πρέπει $(0^2 + 1^2 + 0^2 + 0^2) = 1$, που ισχύει οπότε η κατάσταση είναι αποδεκτή

$$|1\rangle \otimes |1\rangle = |11\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 1 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \text{ Για να είναι αποδεκτή η κα-}$$

τάσταση πρέπει $(0^2 + 0^2 + 0^2 + 1^2) = 1$, που ισχύει οπότε η κατάσταση είναι αποδεκτή

Καταχωρητές που περιέχουν 3 qubits:

$$|1\rangle \otimes |0\rangle \otimes |1\rangle = |101\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 1 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}. \text{ Για να είναι αποδεκτή}$$

η κατάσταση πρέπει $(0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 1^2 + 0^2 + 0^2) = 1$, που ισχύει.

$$|1\rangle \otimes |0\rangle \otimes |1\rangle \otimes |1\rangle = |1011\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 1 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Για να είναι αποδεκτή η κατάσταση πρέπει $(0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 1^2 + 0^2 + 0^2) = 1$, που ισχύει οπότε η κατάσταση είναι αποδεκτή

Με qubits q0: $|a\rangle = \frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle$ και q1: $|b\rangle = \frac{8}{17}|0\rangle + \frac{15}{17}|1\rangle$.

Ένας καταχωρητής που θα συνδυάζει τα 2 qubits όπου:

- $|a\rangle = \frac{3}{5} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{4}{5} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{3}{5} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{4}{5} \end{pmatrix} = \begin{pmatrix} \frac{3}{5} \\ \frac{4}{5} \end{pmatrix}$
- $|b\rangle = \frac{8}{17} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{15}{17} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{8}{17} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{15}{17} \end{pmatrix} = \begin{pmatrix} \frac{8}{17} \\ \frac{15}{17} \end{pmatrix}$ θα είναι ο εξής:

$$\bullet \quad |ab\rangle = |a\rangle \otimes |b\rangle = \begin{pmatrix} \frac{3}{5} \\ \frac{4}{5} \\ \frac{4}{5} \\ \frac{3}{5} \end{pmatrix} \otimes \begin{pmatrix} \frac{8}{17} \\ \frac{15}{17} \\ \frac{15}{17} \\ \frac{8}{17} \end{pmatrix} = \begin{pmatrix} \frac{3}{5} * \begin{pmatrix} \frac{8}{17} \\ \frac{15}{17} \end{pmatrix} \\ \frac{4}{5} * \begin{pmatrix} \frac{8}{17} \\ \frac{15}{17} \end{pmatrix} \\ \frac{4}{5} * \begin{pmatrix} \frac{8}{17} \\ \frac{15}{17} \end{pmatrix} \\ \frac{3}{5} * \begin{pmatrix} \frac{8}{17} \\ \frac{15}{17} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \frac{24}{85} \\ \frac{75}{85} \\ \frac{32}{85} \\ \frac{60}{85} \end{pmatrix} = \frac{24}{85} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \frac{75}{85} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \frac{32}{85} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \frac{60}{85} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \frac{24}{85} |00\rangle + \frac{75}{85} |01\rangle + \frac{32}{85} |10\rangle + \frac{60}{85} |11\rangle$$

Και η κατάσταση είναι αποδεκτή καθώς $\left(\frac{24}{85}\right)^2 + \left(\frac{75}{85}\right)^2 + \left(\frac{32}{85}\right)^2 + \left(\frac{60}{85}\right)^2 = 1$

Το παρακάτω παράδειγμα υλοποιείται στην γλώσσα προγραμματισμού python και στο περιβάλλον jupyter notebook, η αναπαράσταση σε κβαντικό προσομοιωτή αλλά και σε πραγματικό κβαντικό υπολογιστή γίνεται με την βοήθεια της Qiskit. Η Qiskit είναι ανοιχτού κώδικα SDK. Η Python περιέχει βιβλιοθήκες της Qiskit με σκοπό να πραγματοποιούνται και να εκτελούνται τα πειράματα και να δημιουργούνται τα κυκλώματα που όπως της εικόνας 35

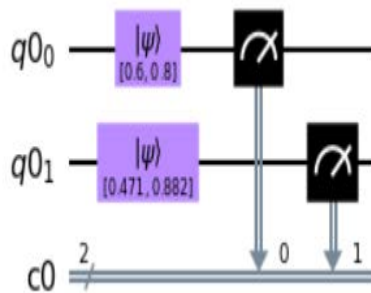
```
1 from qiskit import *
2 from qiskit.tools.visualization import plot_histogram
3 from qiskit.tools.monitor import job_monitor
4 import matplotlib.pyplot as plt
5 from math import sqrt
6 first_qubit = [3/5, 4/5]
7 second_qubit = [8/17, 15/17]
8
9
10 q = QuantumRegister(2)
11 c = ClassicalRegister(2)
12 qc = QuantumCircuit(q,c)
13 qc.initialize(first_qubit, 0)
14 qc.initialize(second_qubit, 1)
15
16 qc.measure(q,c)
17 qc.draw()
```

Εικόνα35:Υλοποίηση κυκλώματος

Ο κώδικας λειτουργεί ως εξής:

- Γραμμή 1 – 5 :Εισαγωγή Βιβλιοθηκών
- Γραμμή 6 – 7:Δημιουργία των καταστάσεων των qubit
- Γραμμή 10 :Δημιουργία Κβαντικού Καταχωρητή που περιέχει τις καταστάσεις των qubits
- Γραμμή 11 :Δημιουργία κλασικού καταχωρητή στον οποίο θα τοποθετηθούν και θα αποθηκευτούν οι καταστάσεις των qubit ώστε μετά την μέτρηση να μην χαθούν
- Γραμμή 12:Δημιουργία Κυκλώματος
- Γραμμή 13-14:Εισαγωγή των qubits στο κύκλωμα
- Γραμμή 16:Μέτρηση της κατάστασης του κβαντικού καταχωρητή και έλεγχος εγκυρότητας της. Μετά από κάθε μέτρηση τα qubits καταστρέφονται και δεν μπορούν να ξαναχρησιμοποιηθούν
- Γραμμή 17:Αναπαράσταση του κυκλώματος

Με το πέρας της γραμμής 17 το κύκλωμα αναπαρίσταται στην εικόνα 36:



Εικόνα36:Αναπαράσταση Κυκλώματος

3.3 Κβαντικές Πύλες

Όπως έχει αναφερθεί και προηγουμένως τα qubits είναι αυτά τα οποία βοηθάνε τον κβαντικό υπολογιστή να εκτελέσει τους υπολογισμούς που χρειάζεται δίνοντας τους ταυτόχρονα μεγάλη δύναμή. Αλλά πέρα από τα qubits εξίσου σημαντικές για την εκτέλεση των υπολογισμών είναι και οι κβαντικές πύλες. Κάθε κβαντική πύλη που θα εφαρμοστεί σε ένα n αριθμό qubits είναι ένας $2^n \times 2^n$ μοναδιαίος πίνακας. Οι πύλες καθορίζουν τον τρόπο με τον οποίο θα λειτουργήσει ένα qubit, χωρίς αυτές τα qubits θεωρούνται άχρηστα. Οι πύλες μπορεί να εφαρμόζονται σε ένα ή σε περισσότερα qubit, κάθε μία πύλη αναπαριστάτε μέσα από ένα κβαντικό κύκλωμα και εφαρμόζεται σε ένα μικρό αριθμό qubits. Ένα κβαντικό κύκλωμα δε διαφέρει από ένα κλασικό κύκλωμα, χρησιμοποιείται για να γίνει αναπαράσταση των αρχικών καταστάσεων των qubit, να εφαρμοστούν οι πύλες και τέλος για να γίνει η μέτρηση των qubit.

3.3.1 Πύλη Αδράνειας

Ποια είναι η λειτουργία της πύλης αδράνειας?

Η πύλη αδράνειας (Identity Gate) επιδρά σε ένα qubit αποκλειστικά και αφήνει ανεπηρέαστη την κατάσταση του qubit

Αναπαριστάτε με τον εξής πίνακα:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Η εφαρμογή της παραπάνω πύλης στις βασικές καταστάσεις:

- $I|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$
- $I|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$

3.3.2 Πύλη Handamard

Ποια είναι η λειτουργία της πύλης Handamard?

Η πύλη Handamard η αλλιώς πύλη υπέρθεσης είναι η σημαντικότερη κβαντική πύλη καθώς τοποθετεί ένα qubit σε υπέρθεση. Οπότε μέχρι να γίνει μέτρηση του qubit η ακριβής του κατάσταση δεν είναι γνωστή

Η μαθηματική αναπαράσταση είναι εξής:

- $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

Υστέρα γίνεται εφαρμογή της παραπάνω πύλης στις βασικές καταστάσεις:

- $H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} |0\rangle + |1\rangle = |+\rangle$
- $H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}} |0\rangle - |1\rangle = |-\rangle$

Ακόμα $H(H|0\rangle) = |0\rangle$ και $H(H|1\rangle) = |1\rangle$, δηλαδή η δεύτερη εφαρμογή της πύλης Handamard και οποιασδήποτε πύλης εξουδετερώνει την πρώτη πύλη

3.3.3 Πύλες Pauli

Πόσες και ποιες είναι οι πύλες Pauli?

Οι πύλες Pauli είναι 4 η X,Y,Z κάθε μια από αυτές εφαρμόζεται σέ ένα qubit κάθε μια από αυτές περιστρέφει το qubit γύρω από τους άξονες X, Y, Z

3.3.3.1 Pauli -X(Πύλη X)

Η πύλη X περιστρέφει το qubit 180 μοίρες γύρω από τον X άξονα μαθηματική αναπαράσταση και εφαρμογή της πύλης X στις βασικές κβαντικές καταστάσεις είναι ως εξής:

- $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- $X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$
- ο $X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$

Οπότε η πύλη X αντιστρέφει την κατάσταση του κάθε qubit, δηλαδή λειτουργεί σαν την κλασική πύλη NOT

3.3.3.2 Pauli -Z(Πύλη Z)

Η πύλη Z περιστρέφει το qubit 180 μοίρες γύρω από τον Z άξονα, θεωρείτε πύλη αλλαγής φάσης και αλλάζει μόνο την φάση του qubit όταν αυτό

βρίσκεται στην $|1\rangle$. Η μαθηματική αναπαράσταση και εφαρμογή της πύλης Z στις βασικές κβαντικές καταστάσεις είναι ως εξής

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

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

3.3.3.3 Pauli -Y(Πύλη Y)

Η πύλη Y περιστρέφει το qubit 180 μοίρες γύρω από τον Y άξονα, συνεπώς αφήνει ανεπηρέαστες τις καταστάσεις $|0\rangle$ και $|1\rangle$ και αντιστρέφει τις καταστάσεις $|+\rangle$ και $|-\rangle$. Θεωρείτε ως πύλη αλλαγής φάσης και η μαθηματική της αναπαράσταση και εφαρμογή της πύλης Y στις βασικές κβαντικές καταστάσεις είναι ως εξής:

- $Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
- $Y|0\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -i \\ 1 \end{pmatrix} = i \begin{pmatrix} 1 \\ 0 \end{pmatrix} = i|0\rangle$
- $Y|1\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -i \\ 0 \end{pmatrix} = -i \begin{pmatrix} 0 \\ 1 \end{pmatrix} = -i|1\rangle$

3.3.4 Πύλη cX (controlled -X):

Πέρα από τις πύλες που δρουν σε ένα qubit υπάρχουν και αυτές που δρουν σε πάνω από ένα qubit. Μια από αυτές είναι η cX πύλη η αλλιώς c-not, επιδρά σε δύο qubits όπου το πρώτο ονομάζεται qubit ελέγχου(control qubit) ενώ το δεύτερο ονομάζεται qubit στόχου(target qubit). Το control qubit καθορίζει την κατάσταση του δεύτερου qubit καθώς αν η κατάσταση του control qubit είναι $|1\rangle$ τότε η κατάσταση του target qubit αλλάζει, αλλιώς αν η κατάσταση του control qubit είναι $|0\rangle$ τότε η κατάσταση του target qubit παραμένει ίδια. Η πύλη αναπαρίστατε με τον εξής πίνακα :

$$\bullet \quad cX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Η εφαρμογή της cX σε ένα καταχωρητή αποτελούμενο από 2 qubits και συγκεκριμένα στο $|11\rangle$ θα έχει το εξής αποτέλεσμα:

$$\bullet \quad cX|11\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = |10\rangle \text{ οπότε είναι αντιληπτό πως όταν το}$$

control qubit είναι 1 τότε το target qubit αλλάζει

3.3.5 Πύλη cZ(controlled -Z)

Εκτός από την πύλη cX μια άλλη πύλη η οποία επιδρά σε 2 qubits είναι η cZ πύλη με το πρώτο control qubit και το δεύτερο target qubit.

Η πύλη δρά ως εξής εάν το control qubit είναι σε κατάσταση $|0\rangle$ τότε το control qubit δεν επηρεάζει καθόλου το target qubit, ενώ εάν το control qubit είναι σε κατάσταση $|1\rangle$ τότε δρα η cZ πύλη και αλλάζει την φάση του qubit όταν το target qubit βρεθεί στην. Αφήνει ανεπηρέαστη την φάση του target qubit όταν αυτό βρίσκεται στην $|0\rangle$ το target qubit αλλάζει φάση

Δηλαδή η εξής πύλης λειτουργεί ως εξής:

- $CZ|00\rangle = |00\rangle$
- $CZ|01\rangle = |01\rangle$
- $CZ|10\rangle = |10\rangle$
- $CZ|11\rangle = -|11\rangle$

Ενώ η μαθηματική της αναπαράσταση είναι η εξής:

$$\bullet \quad cZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

Η εφαρμογή της cZ σε ένα καταχωρητή αποτελούμενο από 2 qubits και συγκεκριμένα στο $|11\rangle$ θα έχει το εξής αποτέλεσμα:

$$\bullet \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\bullet \quad CZ|11\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \end{pmatrix} = -|11\rangle$$

3.3.6 Πύλη cH (controlled -Handamard)

Όπως και οι άλλες controlled πύλες έτσι και η controlled Handamard δρά σε 2 qubits και δρά όταν το control qubit βρίσκεται στην κατάσταση $|1\rangle$. Δηλαδή το control qubit όταν είναι στην $|1\rangle$ τοποθετεί σε υπέρθεση το target qubit

Ο πίνακας για την πύλη είναι ο εξής:

$$\bullet \quad cH = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Η εφαρμογή της cH σε ένα καταχωρητή αποτελούμενο από 2 qubits και συγκεκριμένα στο $|11\rangle$ θα έχει το εξής αποτέλεσμα:

$$\begin{aligned} \circ \quad CH|11\rangle &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} - \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \\ &= \frac{1}{\sqrt{2}} (|01\rangle - |11\rangle) \text{ το target qubit μπήκε σε υπέρθεση} \end{aligned}$$

Ακόμα η εφαρμογή της CH στον $|00\rangle$ θα έχει το εξής αποτέλεσμα

$$\begin{aligned} \circ \quad CH|00\rangle &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00\rangle \text{ όπου η κατά-} \\ &\text{σταση του target qubit δεν άλλαξε το target qubit} \end{aligned}$$

3.3.7 Πύλη Swap

Η πύλη Swap εφαρμόζεται σε δύο qubits, ο ρόλος αυτής της πύλης είναι να αντιστρέψει τις καταστάσεις των δύο qubit.

Η παραπάνω πύλη αναπαρίσταται με την βοήθεια τους εξής πίνακα:

$$\bullet \quad \text{Swap} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Η εφαρμογή της πύλης Swap στον $|10\rangle$ θα έχει το εξής αποτέλεσμα:

$$\bullet \quad \text{SWAP}|10\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle$$

3.3.8 Toffoli Πύλη(CCNOT πύλη)

Δημιουργήθηκε από τον Tommaso Toffoli είναι γνωστή και ως control-control-not πύλη, εφαρμόζεται σε 3 και όχι σε 2 qubits.

Τα δύο πρώτα qubits είναι τα control qubits ενώ το τελευταίο αποτελεί το target qubit. Η πύλη εφαρμόζεται μόνο αν τα δύο πρώτα qubit είναι στην κατάσταση $|1\rangle$ και αλλάζουν την κατάσταση του τρίτου qubit.

Ο πίνακας αναπαράστασης της πύλης CCX είναι ο εξής:

$$\bullet \text{ CCX} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Ακόμα η εφαρμογή της CH στον $|110\rangle$ θα έχει το εξής αποτέλεσμα

$$\bullet \text{ CCX}|110\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |111\rangle$$

3.3.9 Fredkin Πύλη(CSWAP Πύλη)

Η CSWAP πύλη δημιουργήθηκε από τον Edward Fredkin, όπως και η CCX πύλη εφαρμόζεται σε 3 qubits και όταν το πρώτο qubit βρίσκεται στην κατάσταση $|1\rangle$ τότε γίνεται αλλαγή των δύο άλλων qubit.

Ο πίνακας που αναπαριστά την πύλη είναι ο εξής:

$$\bullet \text{ CSWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Η εφαρμογή της CSWAP στον $|101\rangle$ είναι η εξής:

$$\bullet \text{ CSWAP}|101\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |110\rangle$$

3.3.10 Μενού πυλών

Για τις παραπάνω πύλες ακολουθεί ένα μενού το οποίο ανάλογα με την επιλογή του χρήστη, θα δημιουργείτε το αντίστοιχο κύκλωμα, και το ιστόγραμμα το οποίο θα παρουσιάζει τις πιθανότητες εμφάνισης της κάθε κατάστασης και τέλος ακολουθεί η αναπαράσταση του qubit με μορφή της σφαίρας Bloch. Στις εικόνες 35,36,37,38 ακολουθεί ο κώδικας του μενού

```

1 from qiskit import *
2 from qiskit.visualization import circuit_drawer
3 from qiskit.tools.visualization import plot_histogram
4 from qiskit.tools.monitor import job_monitor
5 from qiskit.visualization import plot_bloch_multivector, plot_histogram
6 from math import *
7 from qiskit.providers.ibmq import least_busy
8 from qiskit import IBMQ, QuantumCircuit, transpile, assemble
9
10
11 n = int(input('numbers of qubits:'))
12 qr = QuantumRegister(n,'qubit')
13 cr = ClassicalRegister(n,'bit')
14 qc = QuantumCircuit(qr,cr)
15
16 def menu():
17     print("0 Handamard Gate",
18           "\n1 Identity Gate",
19           "\n2 X Gate",
20           "\n3 Z Gate",
21           "\n4 Y Gate",
22           "\n5 cX Gate",
23           "\n6 cZ Gate",
24           "\n7 cH Gate",
25           "\n8 Swap",
26           "\n9 Cswap",
27           "\n10 ccX Gate"
28         )
29

```

Εικόνα35: Μενού Πυλών

Γραμμή 1-8:Εισαγωγή των απαραίτητων βιβλιοθηκών

Γραμμή 11:Εισαγωγή του αριθμού των qubits που θα χρησιμοποιηθούν στο κύκλωμα

Γραμμή 12: Χρήση κβαντικού καταχωρητή, ο οποίος περιέχει τον αριθμό των qubit

Γραμμή 13: Χρήση κλασικού καταχωρητή για την αποθήκευση των καταστάσεων των qubit καθώς μετά την μέτρηση τα qubits καταστρέφονται

Γραμμή 14: Δημιουργία του κυκλώματος

Γραμμή 15: Δημιουργία συνάρτησης που αναπαριστά το μενού

Γραμμή 16-27: Εμφάνιση των δυνατών επιλογών με την κλήση του μενού

```

32 def Handamard_Gate():
33     qc.h(0)
34     qc.measure(qr, cr)
35
36 def Identity_Gate():
37     qc.i(0)
38     qc.measure(qr, cr)
39
40 def X_Gate():
41     qc.x(0)
42     qc.measure(qr, cr)
43
44 def Z_Gate():
45     qc.z(0)
46     qc.measure(qr, cr)
47
48 def Y_Gate():
49     qc.y(0)
50     qc.measure(qr, cr)
51
52 def cX_Gate():
53     qc.x(0)
54     qc.i(1)
55     qc.cx(0,1)
56     qc.measure(qr, cr)
57
58 def cZ_Gate():
59     qc.x(0)
60     qc.i(1)
61     qc.cz(0,1)
62     qc.measure(qr, cr)
63
64 def cH_Gate():
65     qc.x(0)
66     qc.i(1)
67     qc.ch(0,1)
68     qc.measure(qr, cr)
69

```

Εικόνα36: Συνέχεια Μενού Πυλών

Γραμμή 32-68: Συναρτήσεις που περιέχουν τις πύλες που θα εκτελεστούν ανάλογα με την επιλογή του χρήστη, κάθε φορά εκτελείτε μόνο μία πύλη

Γραμμή 32-34: Συνάρτηση για πύλη Handamard

Γραμμή 36-38: Συνάρτηση για πύλη Identity

Γραμμή 40-42: Συνάρτηση για πύλη Pauli-X

Γραμμή 44-46: Συνάρτηση για πύλη Pauli-Z

Γραμμή 48-50: Συνάρτηση για πύλη Pauli-Y

Γραμμή 52: Συνάρτηση για πύλη CX Γραμμή 53: Εφαρμογή X πύλης στο πρώτο qubit καθώς η κατάσταση του πρέπει να είναι $|1\rangle$

Γραμμή 54: Εφαρμογή Identity πύλης στο δεύτερο qubit ώστε το κύκλωμα να έχει μια ομοιομορφία.

Γραμμή 55-56: Εφαρμογή πύλης CX και μέτρηση κυκλώματος

Γραμμή 58: Συνάρτηση για πύλη CZ

Γραμμή 59: Εφαρμογή X πύλης στο πρώτο qubit καθώς η κατάσταση του πρέπει να είναι $|1\rangle$

Γραμμή 60: Εφαρμογή Identity πύλης το δεύτερο qubit ώστε το κύκλωμα να έχει μια ομοιομορφία.

Γραμμή 61-62: Εφαρμογή πύλης CZ και μέτρηση κυκλώματος

Γραμμή 64: Συνάρτηση για πύλη CH

Γραμμή 65: Εφαρμογή X πύλης στο πρώτο qubit καθώς η κατάσταση του πρέπει να είναι $|1\rangle$

Γραμμή 66: Εφαρμογή Identity πύλης το δεύτερο qubit ώστε το κύκλωμα να έχει μια ομοιομορφία.

Γραμμή 67-68: Εφαρμογή πύλης CH και μέτρηση κυκλώματος

```
70 def Swap_Gate():
71     qc.x(0)
72     qc.i(1)
73     qc.swap(0,1)
74     qc.measure(qr,cr)
75
76 def Cswap_Gate():
77     qc.x(0)
78     qc.i(1)
79     qc.x(2)
80
81     qc.cswap(0,1,2)
82
83     qc.measure(qr,cr)
84
85 def ccX_Gate():
86     qc.x(0)
87     qc.x(1)
88     qc.i(2)
89     qc.ccx(0,1,2)
90     qc.measure(qr,cr)
91
```

Εικόνα37: Συνέχεια Μενού Πυλών

Γραμμή 70: Συνάρτηση για πύλη SWAP

Γραμμή 71: Εφαρμογή X πύλης στο πρώτο qubit καθώς η κατάσταση του πρέπει να είναι $|1\rangle$ για την δημιουργία της $|10\rangle$

Γραμμή 72: Εφαρμογή Identity πύλης το δεύτερο qubit ώστε το κύκλωμα να έχει μια ομοιομορφία.

Γραμμή 73-74: Εφαρμογή πύλης swap και μέτρηση κυκλώματος

Γραμμή 76: Συνάρτηση για πύλη CSWAP

Γραμμή 77 -79: Εφαρμογή X πύλης στο πρώτο και στο τρίτο qubit καθώς η κατάσταση τους πρέπει να είναι $|1\rangle$ για την δημιουργία της $|101\rangle$.

Γραμμή 81-83: Εφαρμογή πύλης CSWAP και μέτρηση κυκλώματος

Γραμμή 85: Συνάρτηση για πύλη CCX

Γραμμή 86-87: Εφαρμογή X πύλης στο πρώτο και στο δεύτερο qubit καθώς η κατάσταση τους πρέπει να είναι $|1\rangle$ για την δημιουργία της $|110\rangle$.

Γραμμή 89-90: Εφαρμογή πύλης ccx και μέτρηση κυκλώματος

```
145 menu()
146
147 choice = int(input("Input Your Choice:"))
148
149 if choice == 0:
150     Hadamard_Gate()
151 elif choice == 1:
152     Identity_Gate()
153 elif choice == 2:
154     X_Gate()
155 elif choice == 3:
156     Z_Gate()
157 elif choice == 4:
158     Y_Gate()
159 elif choice == 5:
160     cX_Gate()
161 elif choice == 6:
162     cZ_Gate()
163 elif choice == 7:
164     cH_Gate()
165 elif choice == 8:
166     Swap_Gate()
167 elif choice == 9:
168     CSwap_Gate()
169 elif choice == 10:
170     ccX_Gate()
```

Εικόνα 38: Εκτέλεση συνάρτησης ανάλογα με την επιλογή του χρήστη

Γραμμή 145: Εμφάνιση μενού στον χρήστη

Γραμμή 147: Απάντηση χρήστη

Γραμμή 149-170: Εκτέλεση συνάρτησης ανάλογα με την επιλογή του χρήστη

Με την εκτέλεση του κώδικα από τις εικόνες 35, 36, 37, 38 το μενού που εμφανίζεται και τα κυκλώματα που δημιουργούνται είναι τα εξής :

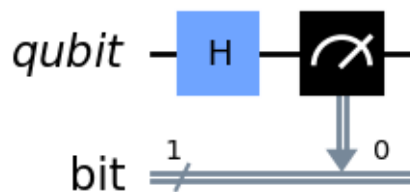
```
numbers of qubits:1
```

```
-----  
0 Handamard Gate  
1 Identity Gate  
2 X Gate  
3 Z Gate  
4 Y Gate  
5 cX Gate  
6 cZ Gate  
7 cH Gate  
8 Swap  
9 Cswap  
10 ccX Gate  
Input Your Choice:0
```

Εικόνα 39:Εκτέλεση του μενού και εμφάνιση των δυνατών επιλογών

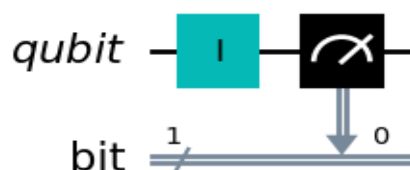
Το κύκλωμα για την πύλη Handamard με αριθμό qubit 1 αναπαρίσταται στην εικόνα

40



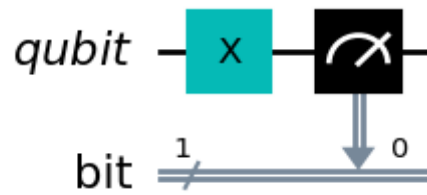
Εικόνα 40:Κύκλωμα για πύλη Handamard

Το κύκλωμα για την πύλη Identity με αριθμό qubit 1 αναπαρίσταται στην εικόνα 41



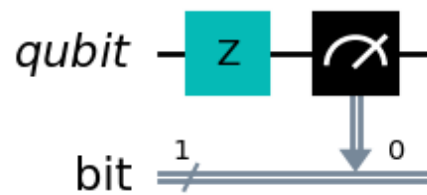
Εικόνα 41: Κύκλωμα για πύλη Identity

Το κύκλωμα για την πύλη Pauli-X με αριθμό qubit 1 αναπαρίσταται στην εικόνα 42



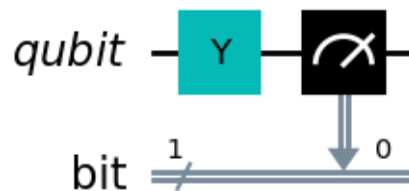
Εικόνα 42: Κύκλωμα για πύλη Pauli-X

Το κύκλωμα για την πύλη Pauli-Z με αριθμό qubit 1 αναπαρίσταται στην εικόνα 43:



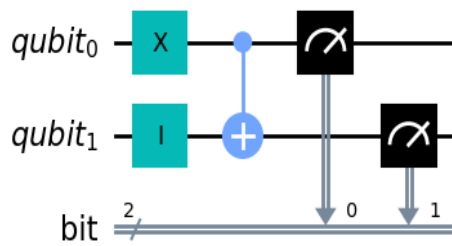
Εικόνα 43: Κύκλωμα για πύλη Pauli-Z

Το κύκλωμα για την πύλη Pauli-Y με αριθμό qubit 1 αναπαρίσταται στην εικόνα 44



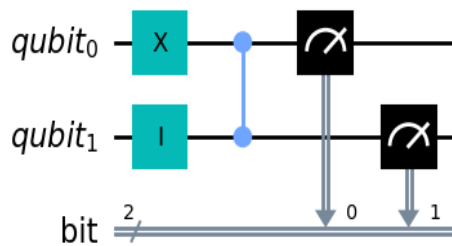
Εικόνα 45: Κύκλωμα για πύλη Pauli-Y

Το κύκλωμα για την πύλη CX με αριθμό qubit 2 αναπαρίσταται στην εικόνα 46



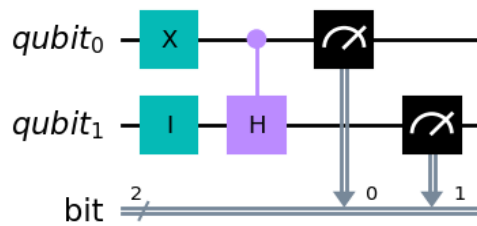
Εικόνα 46: Κύκλωμα για πύλη CX

Το κύκλωμα για την πύλη CZ με αριθμό qubit 2 αναπαρίστανται στην εικόνα 47



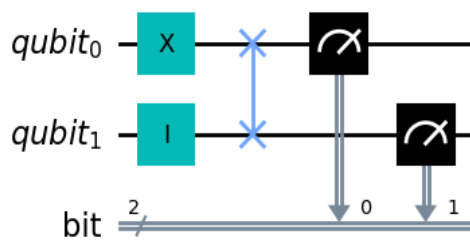
Εικόνα 47: Κύκλωμα για πύλη CZ

Το κύκλωμα για την πύλη CH με αριθμό qubit 2 αναπαρίστανται στην εικόνα 48



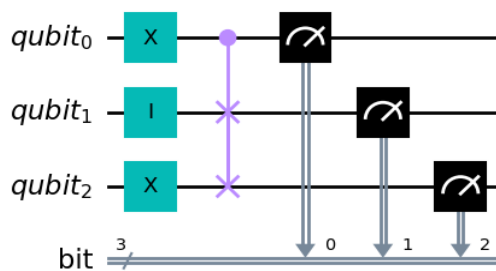
Εικόνα 48: Κύκλωμα για πύλη CH

Το κύκλωμα για την πύλη SWAP με αριθμό qubit 2 αναπαρίστανται στην εικόνα 49



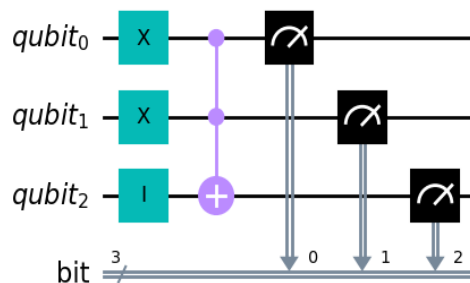
Εικόνα 49: Κύκλωμα για πύλη Swap

Το κύκλωμα για την πύλη CSWAP με αριθμό qubit 3 αναπαρίσταται στην εικόνα



Εικόνα 50: Κύκλωμα για πύλη Cswap

Το κύκλωμα για την πύλη CCX με αριθμό qubit 3 αναπαρίσταται στην εικόνα 51



Εικόνα 51: Κύκλωμα για πύλη CCX

Στην συνέχεια το κύκλωμα της εικόνας θα εκτελεστεί σε πραγματικό κβαντικό υπολογιστή και η εικόνα 52 δείχνει τον τρόπο προσβάσης σε ένα κβαντικό υπολογιστή της IBM:

```

2 IBMQ.load_account()
3 # from qiskit import IBMQ
4 provider = IBMQ.load_account()
5
6 # Get all available backends
7 backends = provider.backends()
8
9 real_backends=[backend for backend in backends if not backend.configuration().simulator and backend.status().operational]
10
11 least_busy_backend = least_busy(real_backends)
12 print(least_busy_backend)
13
14

```

Εικόνα 52: Πρόσβαση σε πραγματικό κβαντικό υπολογιστή

Γραμμή 2-4: Πρόσβαση στον προσωπικό λογαριασμό που έχει δημιουργηθεί στην πλατφόρμα της IBM

Γραμμή 7-9: Εμφάνιση όλων των διαθέσιμων κβαντικών υπολογιστών

Γραμμή 11: Επιλογή του κβαντικού υπολογιστή που είναι λιγότερο απασχολημένος ώστε να μην αργήσει να εκτελεστεί το πρόγραμμα

Υστερα στην εικόνα 53 γίνεται εκτέλεση του κυκλώματος σε πραγματικό κβαντικό υπολογιστή

```

1 device = provider.get_backend(least_busy_backend)
2 job = execute(qc, backend = device, shots=1024)
3 device_result = job.result()
4
5 quantum_results = device_result
6 classical_results=answer

```

Εικόνα 53: Εκτέλεση κυκλωμάτων σε πραγματικό κβαντικό υπολογιστή

Γραμμή 1: Επιλογή του κβαντικού υπολογιστή που θα εκτελεστεί το κύκλωμα

Γραμμή 2: Εκτέλεση του κυκλώματος σε κβαντικό υπολογιστή

Γραμμή 3: Αποθήκευση των αποτελεσμάτων

Υστερά ακολουθεί το ιστόγραμμα για κάθε πύλη, στην εικόνα 54 ακολουθεί ο τίτλος που θα έχει κάθε ιστόγραμμα ανάλογα με την πύλη που εφαρμόζεται .

```

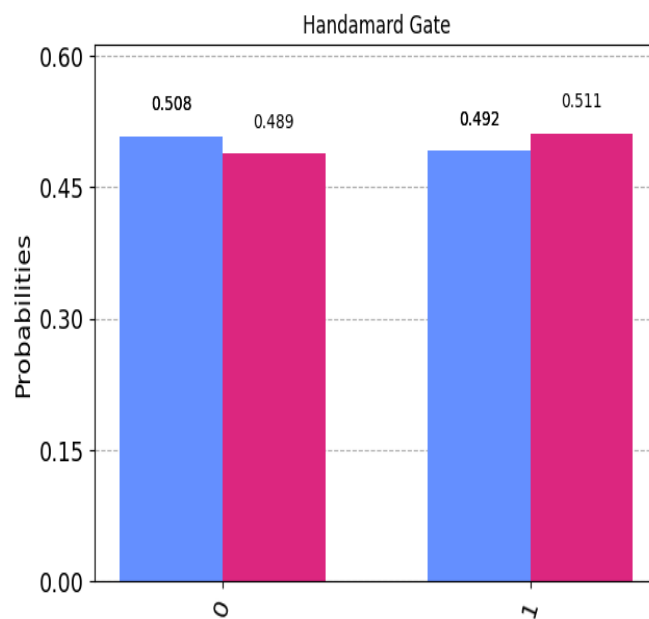
1 if choice == 0:
2     name='Handamard Gate'
3 elif choice == 1:
4     name='Identity Gate'
5 elif choice == 2:
6     name='Pauli-X Gate'
7 elif choice == 3:
8     name='Pauli-Z Gate'
9 elif choice == 4:
10    name='Pauli-Y Gate'
11 elif choice == 5:
12    name='controlled-Not Gate'
13 if choice == 6:
14    name = 'controlled-Z Gate'
15 elif choice == 7:
16    name = 'controlled-Handamard Gate'
17 elif choice == 8:
18    name='Swap Gate'
19 elif choice == 9:
20    name='controlled-Swap Gate'
21 elif choice == 10:
22    name='controlled-controlled-X Gate'
23
24 plot_histogram([quantum_results,classical_results], bar_labels=True,title = name)

```

Εικόνα 54 :Τίτλος ιστογράμματος ανάλογα με την πύλη που εφαρμόστηκε

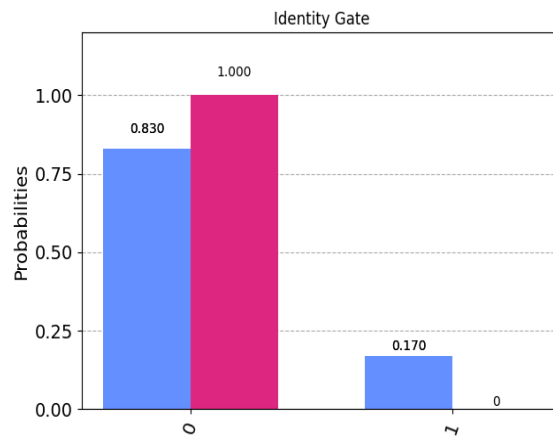
Γραμμή 1-24: Εκτέλεση και αναπαράσταση ιστογράμματος για κάθε πύλη

Το ιστόγραμμα για την πύλη Handamard με 1 qubit αναπαρίσταται στην εικόνα 55



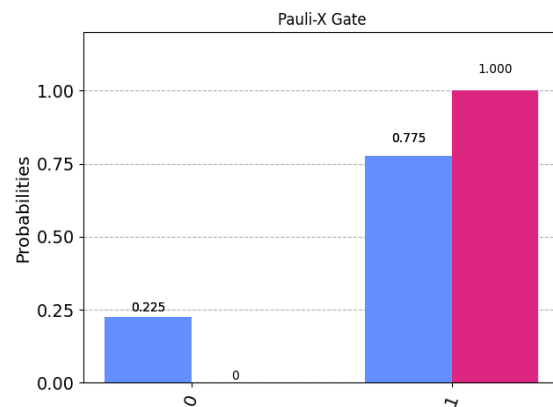
Εικόνα 55: Ιστόγραμμα για πύλη Handamard

Το ιστόγραμμα για την πύλη Identity με 1 qubit αναπαρίσταται στην εικόνα 56



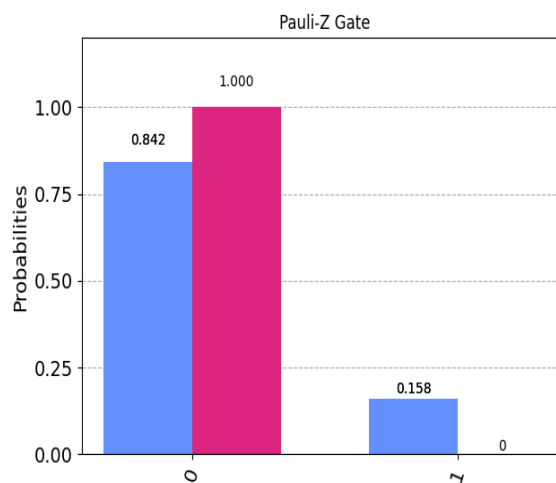
Εικόνα 56: Ιστόγραμμα για πύλη Identity

Το ιστόγραμμα για την πύλη Pauli-X με 1 qubit αναπαρίσταται στην εικόνα 57



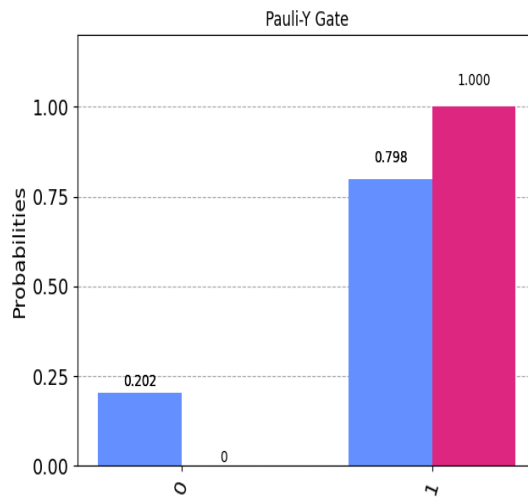
Εικόνα 57: Ιστόγραμμα για πύλη Pauli-X

Το ιστόγραμμα για την πύλη Pauli-Z με 1 qubit αναπαρίσταται στην εικόνα 58



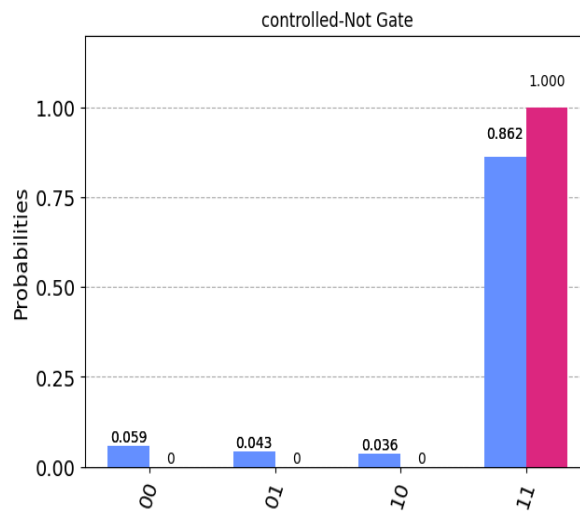
Εικόνα 58: Ιστόγραμμα για πύλη Pauli-Z

Το ιστόγραμμα για την πύλη Pauli-Y με 1 qubit αναπαρίσται στην εικόνα 59



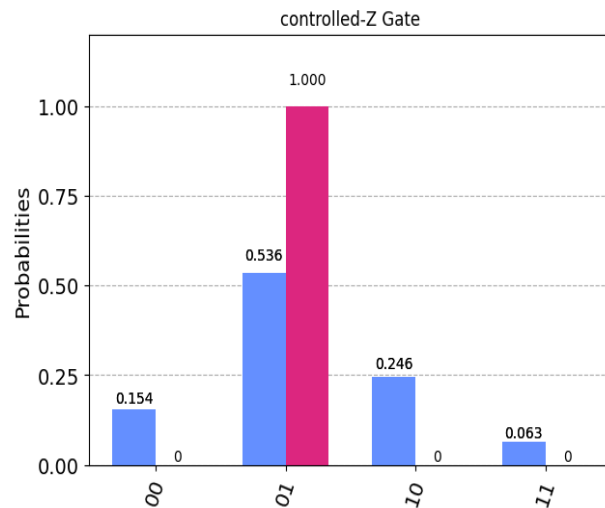
Εικόνα 59: Ιστόγραμμα για πύλη Pauli-Y

Το ιστόγραμμα για την πύλη CX με 2 qubits αναπαρίσται στην εικόνα 60



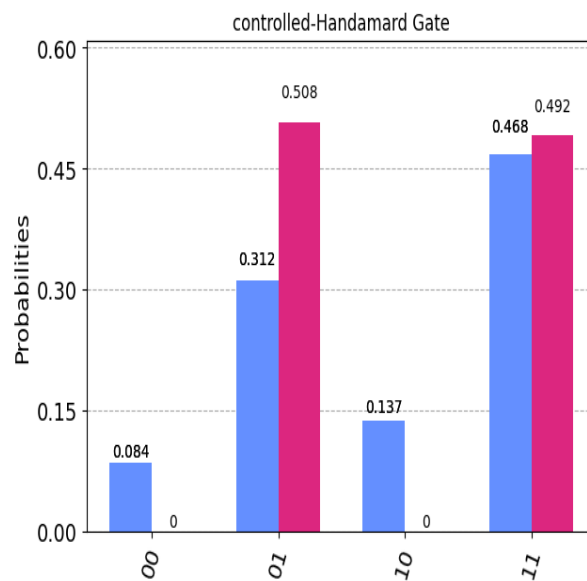
Εικόνα 60: Ιστόγραμμα για CX πύλη

Το ιστόγραμμα για την πύλη CZ με 2 qubits αναπαρίσται στην εικόνα 61



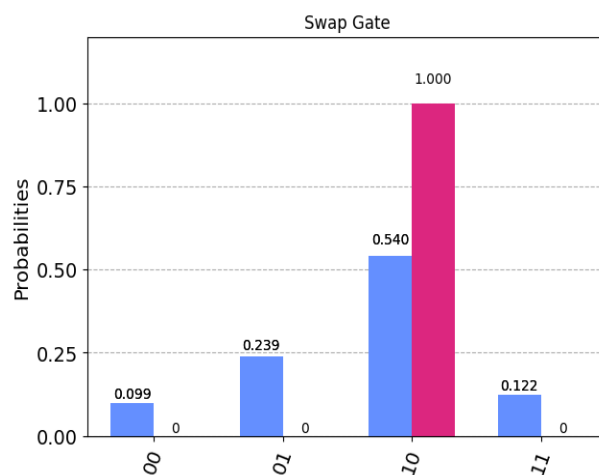
Εικόνα 61: Ιστόγραμμα για CZ πύλη

Το ιστόγραμμα για την πύλη CH με 2 qubits αναπαρίσταται στην εικόνα 62



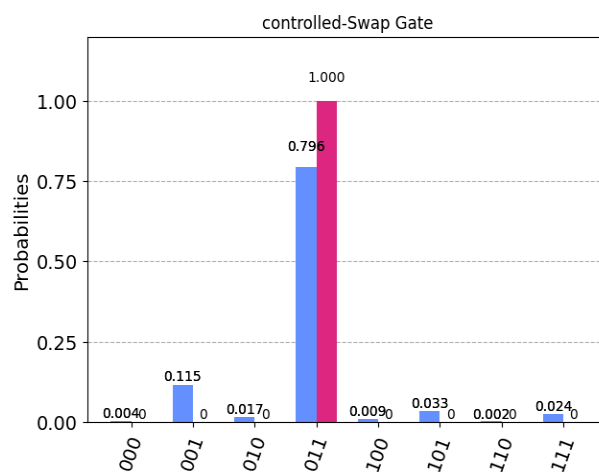
Εικόνα 62: Ιστόγραμμα για CH πύλη

Το ιστόγραμμα για την πύλη SWAP με 2 qubits αναπαρίσταται στην εικόνα 63



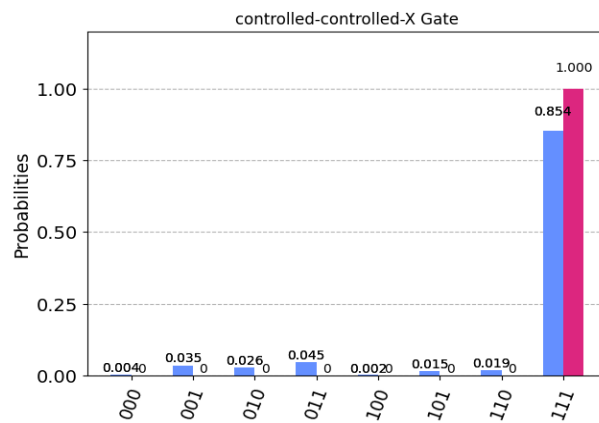
Εικόνα 63: Ιστόγραμμα για SWAP πύλη

Το ιστόγραμμα για την πύλη CSWAP με 3 qubits αναπαρίσται στην εικόνα 64



Εικόνα 64: Ιστόγραμμα για CSWAP πύλη

Το ιστόγραμμα για την πύλη CCX με 3 qubits αναπαρίσται στην εικόνα 65



Εικόνα 65: Ιστόγραμμα για CCX πύλη

Υστέρα στις παρακάτω εικόνες απεικονίζεται η σφαίρα Bloch και αναπαριστά την επιρροή που ασκεί η κάθε πύλη στα qubits

Στην εικόνα 66.0 αναπαρίσταται ο κώδικας που δημιουργεί την σφαίρα Bloch μετά την εφαρμογή της κάθε πύλης στα qubits, ο κώδικας αναπαρίσταται στην εικόνα 66.0

```
1 backend = BasicAer.get_backend('statevector_simulator')
2 result = execute(qc, backend).result()
3 x = result.get_statevector(qc)
4 plot_bloch_multivector(x, title=name)
```

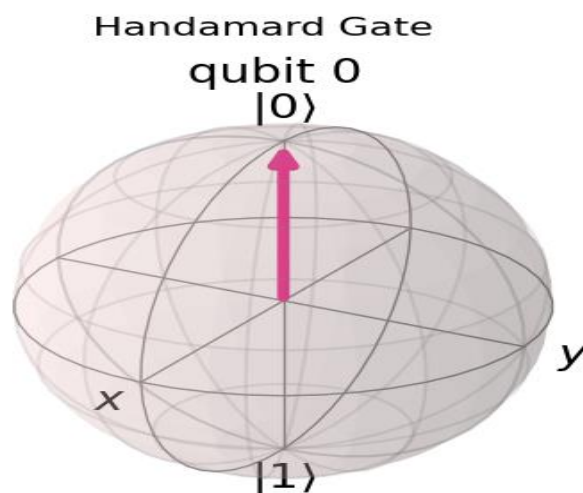
Εικόνα 66.0: Κώδικας για σφαίρα Bloch

Γραμμή 1-2: Εκτέλεση κυκλώματος σε κβαντικό προσομοιωτή

Γραμμή 3: Αποθήκευση δεδομένων με μορφή διανύσματος κατάστασης

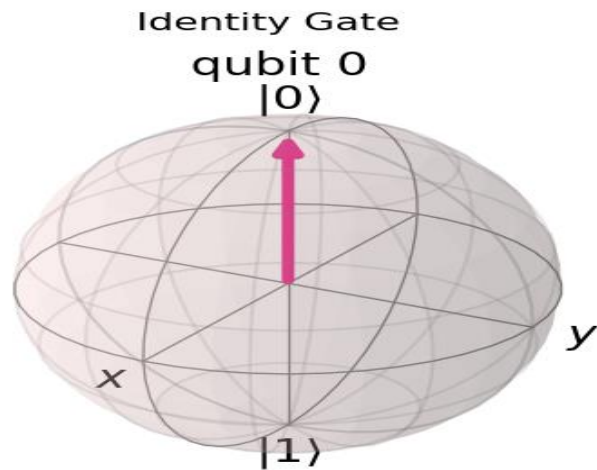
Γραμμή 4: Αναπαράσταση σφαίρας Bloch

Η σφαίρα Bloch μετά την εφαρμογή της πύλης Handamard αναπαρίσταται στην εικόνα 66.1



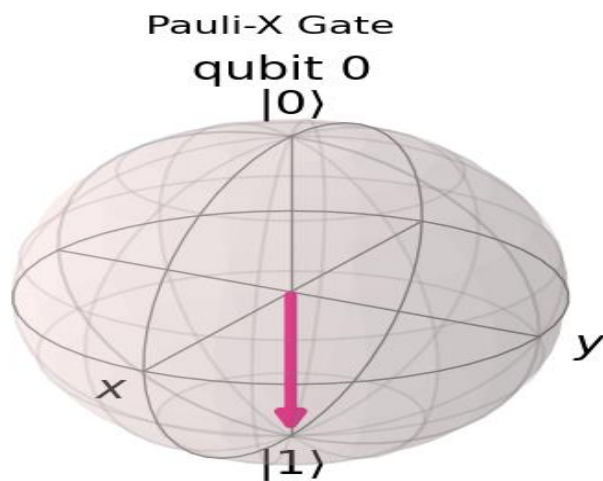
Εικόνα 66.1: Αναπαράσταση σφαίρας Bloch μετά την εφαρμογή Handamard πύλης

Η σφαίρα Bloch μετά την εφαρμογή της πύλης Identity αναπαρίσταται στην εικόνα 66.2



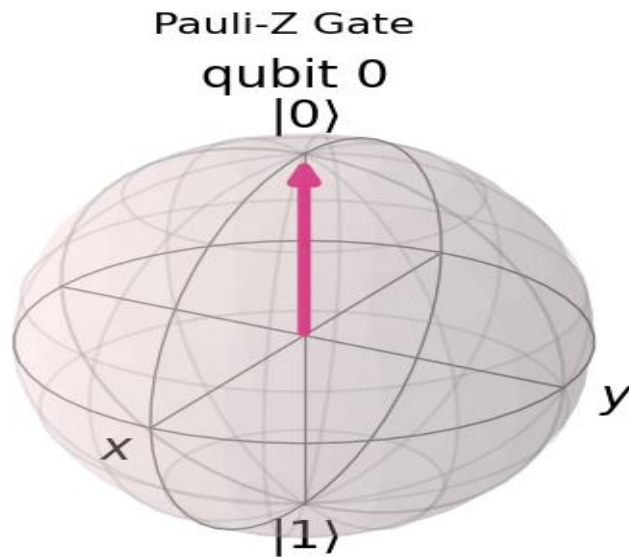
Εικόνα 66.2: Αναπαράσταση σφαίρας Bloch μετά την εφαρμογή Identity πύλης

Η σφαίρα Bloch μετά την εφαρμογή της πύλης Pauli -X αναπαρίσταται στην εικόνα 66.3



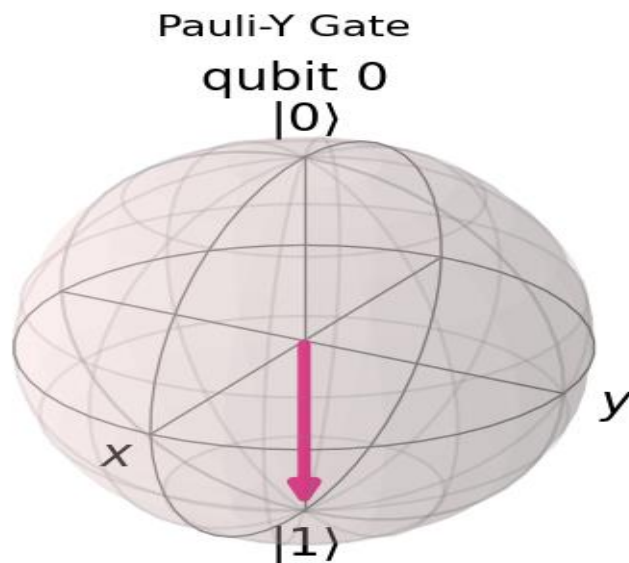
Εικόνα 66.3: Αναπαράσταση σφαίρας Bloch μετά την εφαρμογή Pauli -X πύλης

Η σφαίρα Bloch μετά την εφαρμογή της πύλης Pauli -Z αναπαρίσταται στην εικόνα 66.4



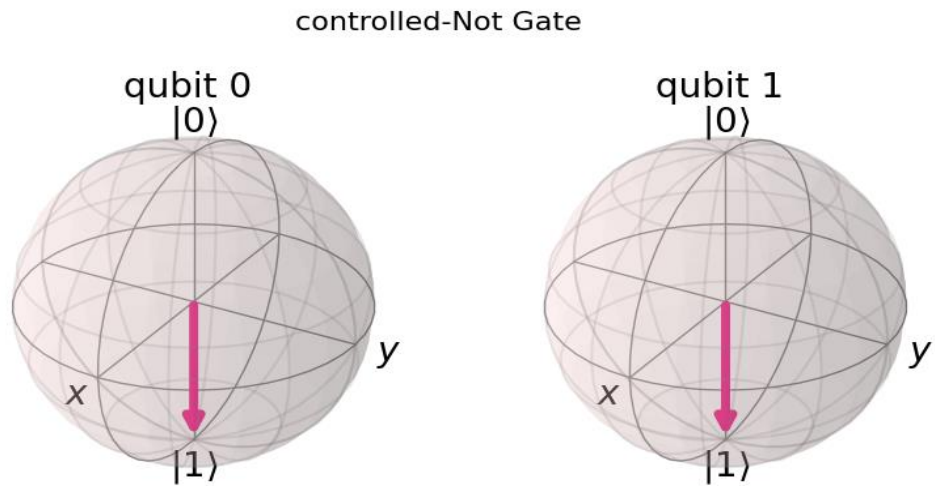
Εικόνα 66.4: Αναπαράσταση σφαίρας Bloch μετά την εφαρμογή Pauli -Z πύλης

Η σφαίρα Bloch μετά την εφαρμογή της πύλης Pauli -Y αναπαρίσταται στην εικόνα 66.5



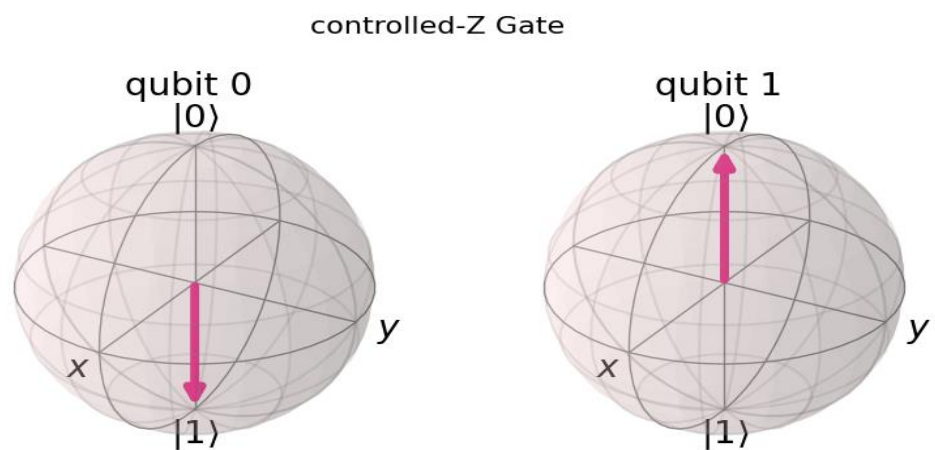
Εικόνα 66.5: Αναπαράσταση σφαίρας Bloch μετά την εφαρμογή Pauli -Y πύλης

Η σφαίρα Bloch μετά την εφαρμογή της πύλης CX αναπαρίσταται στην εικόνα 66.6



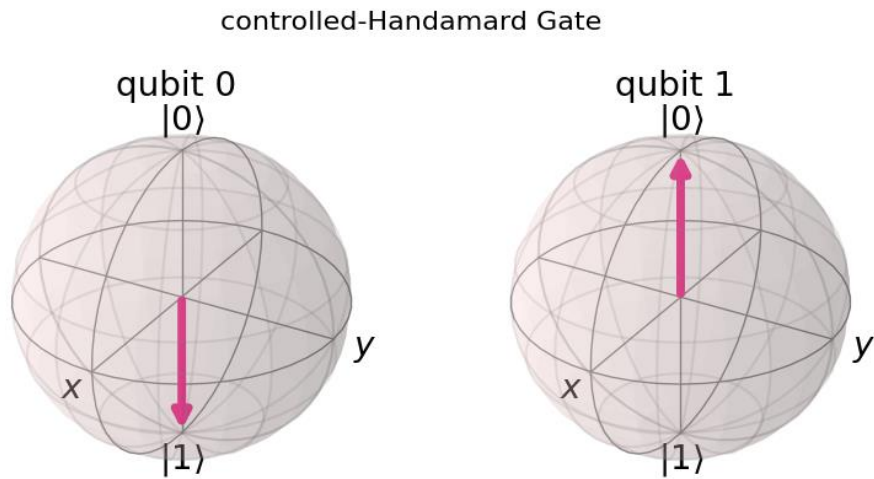
Εικόνα 66.6: Αναπαράσταση σφαίρας Bloch μετά την εφαρμογή CX πύλης

Η σφαίρα Bloch μετά την εφαρμογή της πύλης CZ αναπαρίσταται στην εικόνα 66.7



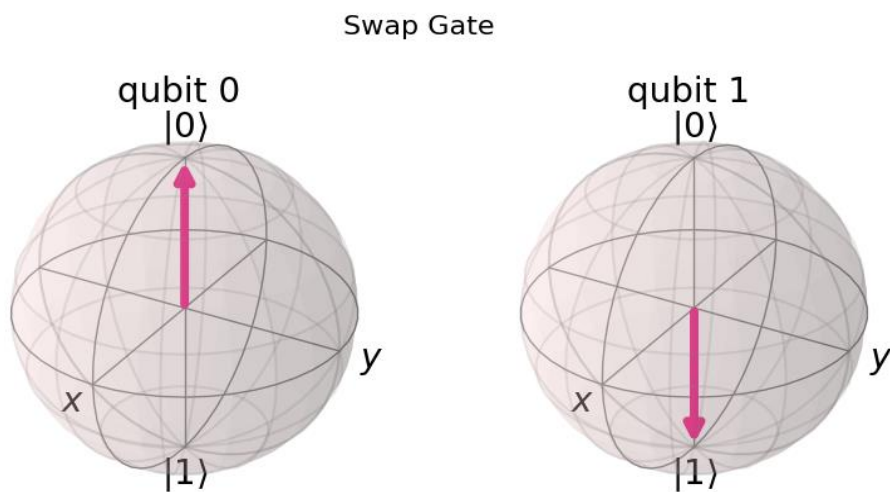
Εικόνα 66.7: Αναπαράσταση σφαίρας Bloch μετά την εφαρμογή CZ πύλης

Η σφαίρα Bloch μετά την εφαρμογή της πύλης CH αναπαρίσταται στην εικόνα 66.8



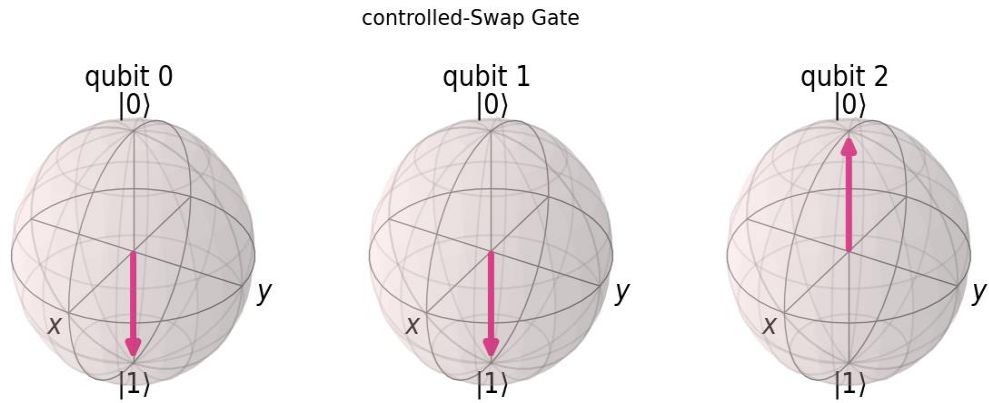
Εικόνα 66.8: Αναπαράσταση σφαίρας Bloch μετά την εφαρμογή CH πύλης

Η σφαίρα Bloch μετά την εφαρμογή της πύλης Swap αναπαρίσταται στην εικόνα 66.9



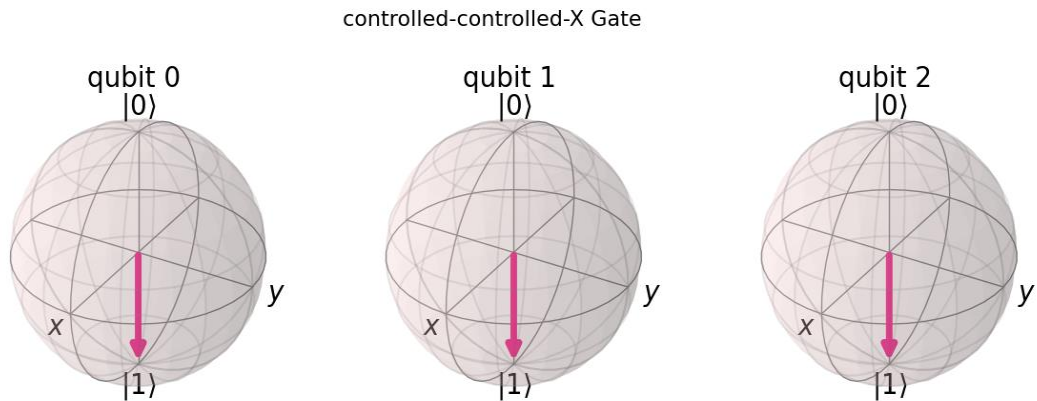
Εικόνα 66.9: Αναπαράσταση σφαίρας Bloch μετά την εφαρμογή Swap πύλης

Η σφαίρα Bloch μετά την εφαρμογή της πύλης CSWAP αναπαρίσταται στην εικόνα 66.10



Εικόνα 66.10: Αναπαράσταση σφαίρας Bloch μετά την εφαρμογή CSWAP πύλης

Η σφαίρα Bloch μετά την εφαρμογή της πύλης CCX αναπαρίσταται στην εικόνα 66.11



Εικόνα 66.11: Αναπαράσταση σφαίρας Bloch μετά την εφαρμογή CCX πύλης

3.3.11 Bell States

Η πύλη cX έχει σημαντικό ρόλο σε μια κλάση καταστάσεων γνωστές και ως Bell States ή EPR States, είναι κβαντικές καταστάσεις που εφαρμόζονται σε δύο qubits. Τα Bell States είναι γνωστά και ως καταστάσεις διεμπλοκής. Όταν γίνεται μέτρηση της κάθε Bell State τότε η αλλαγή στην κατάσταση του ενός qubit θα επιφέρει αυτόματα αλλαγή και στην κατάσταση του άλλου qubit. Οι τέσσερις Bell States είναι οι εξής:

- $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$

- $|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$
- $|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$
- $|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

Στις εικόνες 66 και 67 αναπαρίστατε ένα πρόγραμμα, το οποίο είναι ένα μένου όπου ανάλογα με τον αριθμό που θα δώσει ο χρήστης θα εμφανιστεί και το ανάλογο Bell State

```

1 from qiskit import *
2 from qiskit.tools.monitor import job_monitor
3 from qiskit.tools.visualization import plot_histogram
4 from qiskit.providers.ibmq import least_busy
5 from qiskit import IBMQ, QuantumCircuit, transpile, assemble
6 from pythonosc import udp_client, dispatcher
7 import numpy as np
8 import socket
9 import random
10 import time
11
12
13
14 def menu():
15     print("1. |Φ+>\n",
16           "2. |Φ->\n",
17           "3. |Ψ+>\n",
18           "4. |Ψ->\n")
19
20
21 menu()
22 q = QuantumRegister(2, 'qubit')
23 c = ClassicalRegister(2, 'bit')
24 qc = QuantumCircuit(q, c)
25 choice = int(input("Δώσε έναν αριθμό απο το 1 μέχρι το 4: "))
26
27
28 if choice == 1:
29     qc.h(q[0])
30     qc.i(q[1])
31     qc.cx(q[0], q[1])
32     qc.measure(q, c)
33 elif choice == 2:
34     qc.x(q[0])
35     qc.i(q[1])
36     qc.h(q[0])
37     qc.i(q[1])
38     qc.cx(q[0], q[1])

```

Εικόνα 66: Bell States μενού

```

41 elif choice == 3:
42     qc.h(q[0])
43     qc.x(q[1])
44     qc.cx(q[0],q[1])
45     qc.measure(q,c)
46 elif choice == 4:
47     qc.h(q[0])
48     qc.x(q[1])
49     qc.z(q[0])
50     qc.z(q[1])
51     qc.cx(q[0],q[1])
52
53
54     qc.measure(q,c)
55
56 qc.draw(output = "mpl")
57
58

```

Εικόνα 67: Bell States μενού

Γραμμή 1 - 3: Εισαγωγή όλων των εργαλείων που περιέχει η Qiskit και χρειάζονται για την σωστή εκτέλεση του προγράμματος

Γραμμή 14-18: Δημιουργία συνάρτησης που αναπαριστά το μενού

Γραμμή 21: Εμφάνιση του μενού που είναι οι τέσσερις καταστάσεις διεμπλοκής που έχει την δυνατότητα ο χρήστης να επιλέξει.

Γραμμή 22: Δημιουργία κβαντικού καταχωρητή

Γραμμή 23: Δημιουργία κλασσικού καταχωρητή

Γραμμή 24: Δημιουργία του κυκλώματος

Γραμμή 25: Εισαγωγή αριθμού από τον χρήστη έτσι ώστε να εκτελεστεί το αντίστοιχο bell state

Γραμμή 28 -51: Δημιουργία του κάθε Bell State που ανάλογα με την επιλογή του χρήστη

Γραμμή 56: Αναπαράσταση του κυκλώματος που αντιστοιχεί στο Bell State που επέλεξε ο χρήστης

Υστερα στην εικόνα 68 αναπαρίστανται το αποτέλεσμα που θα δώσει το μενού των

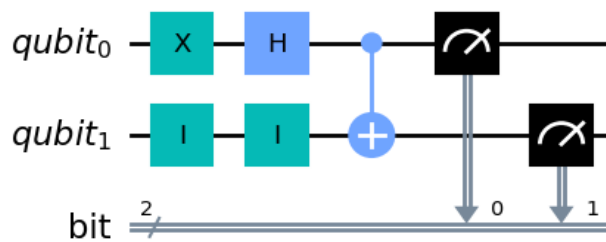
Bell States:

1. $|\phi+\rangle$
2. $|\phi-\rangle$
3. $|\psi+\rangle$
4. $|\psi-\rangle$

Δώσε έναν αριθμό απο το 1 μέχρι το 4: 2

Εικόνα 68: Αποτέλεσμα μενού

Το κύκλωμα για την επιλογή που έδωσε ο χρήστης δηλαδή για την $|\Phi^-\rangle$ φαίνεται στην εικόνα 69:



Εικόνα 69: Κύκλωμα για $|\Phi^-\rangle$

Υστέρα στην εικόνα 70 αναγράφεται ο κώδικας για την εκτέλεση του κυκλώματος σε κβαντικό προσομοιωτή

```
1
2 job = execute(qc, Aer.get_backend('qasm_simulator'), shots = 1024).result()
3 answer = job.get_counts()
```

Εικόνα 70: Εκτέλεση κυκλώματος σε κβαντικό προσομοιωτή

Γραμμή 2: Εκτέλεση του κυκλώματος σε κβαντικό προσομοιωτή 1024 φορές

Γραμμή 3: Αποθήκευση των δεδομένων από την εκτέλεση του κυκλώματος σε μια μεταβλητή

Στην συνέχεια το πρόγραμμα των εικόνων 66,67 θα εκτελεστεί σε πραγματικό κβαντικό υπολογιστή και η εικόνα 71 δείχνει τον τρόπο πρόσβασης σε ένα κβαντικό υπολογιστή της IBM:

```

2 IBMQ.load_account()
3 # from qiskit import IBMQ
4 provider = IBMQ.load_account()
5
6 # Get all available backends
7 backends = provider.backends()
8
9 real_backends=[backend for backend in backends if not backend.configuration().simulator and backend.status().operational]
10
11 least_busy_backend = least_busy(real_backends)
12 print(least_busy_backend)
13
14

```

Εικόνα 71: Πρόσβαση σε πραγματικό κβαντικό υπολογιστή

Γραμμή 2-4: Πρόσβαση στον προσωπικό λογαριασμό που έχει δημιουργηθεί στην πλατφόρμα της IBM

Γραμμή 7-9: Εμφάνιση όλων των διαθέσιμων κβαντικών υπολογιστών

Γραμμή 11: Επιλογή του κβαντικού υπολογιστή που είναι λιγότερο απασχολημένος ώστε να μην αργήσει να εκτελεστεί το πρόγραμμα

Υστερα γίνεται εκτέλεση του προγράμματος σε πραγματικό κβαντικό υπολογιστή

```

1 device = provider.get_backend('ibm_kyoto')
2 job = execute(qc, backend = device, shots = 1024)
3 device_result = job.result()
4
5 quantum_results = device_result.get_counts()
6 classical_results=answer
7
8
9 plot_histogram([quantum_results,classical_results], bar_labels=True)

```

Εικόνα 71: Εκτέλεση προγράμματος σε πραγματικό κβαντικό υπολογιστή

Γραμμή 1: Επιλογή του κβαντικού υπολογιστή που θα εκτελεστεί το πρόγραμμα

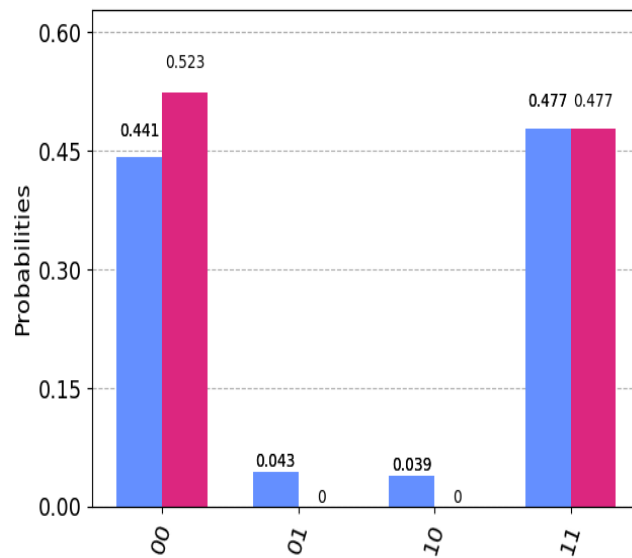
Γραμμή 2: Εκτέλεση δεδομένων σε κβαντικό υπολογιστή

Γραμμή 3: Αποθήκευση των αποτελεσμάτων

Γραμμή 5: Αποθήκευση των αποτελεσμάτων για όλες τις φορές που θα εκτελεστεί το πρόγραμμα

Γραμμή 9: Αναπαράσταση ιστογράμματος που περιέχει τα δεδομένα του κυκλώματος όταν το κύκλωμα έτρεξε σε κβαντικό υπολογιστή και όταν έτρεξε σε προσομοίωση κβαντικού υπολογιστή

Το ιστόγραμμα αναπαρίσταται στην εικόνα 72

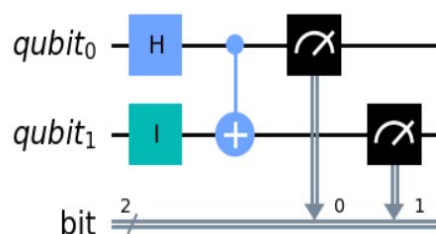


Εικόνα 72: Ιστόγραμμα για $|\Phi^-\rangle$

Οι ρόζ οι μπάρες είναι τα αποτελέσματα που έδωσε ο κβαντικός προσομοιωτής, ενώ οι μπλέ μπάρες είναι τα αποτελέσματα που έδωσε ο κβαντικός υπολογιστής. Από το παραπάνω κύκλωμα της εικόνας είναι αντιληπτό πως ένα κβαντικός υπολογιστής θα δώσει μεγαλύτερη ακρίβεια στα αποτελέσματα που θα προκύψουν από το κύκλωμα, από ότι ο κβαντικός προσομοιωτής στον οποίο θα τρέξει το ίδιο κύκλωμα.

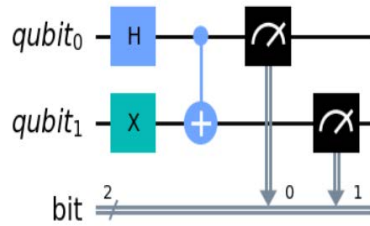
Τα κυκλώματα των υπόλοιπων πυλών απεικονίζονται στις παρακάτω εικόνες:

Το κύκλωμα για την $|\Phi^+\rangle$ είναι το εξής στην εικόνα 73:



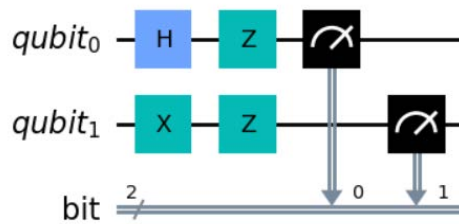
Εικόνα 73: Κύκλωμα για $|\Phi^+\rangle$

Το κύκλωμα για την $|\Psi^+\rangle$ είναι το εξής στην εικόνα 74:



Εικόνα 74: Κύκλωμα για $|\Psi^+\rangle$

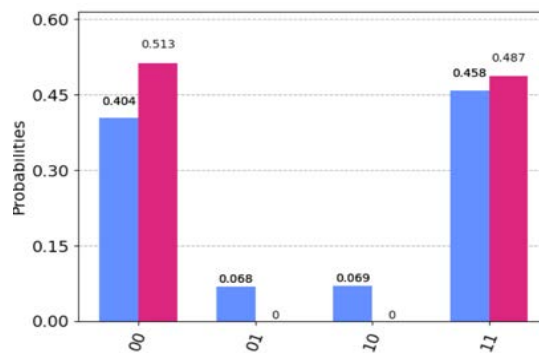
Το κύκλωμα για την $|\Psi^-\rangle$ είναι το εξής στην εικόνα 75:



Εικόνα 75: Κύκλωμα για $|\Psi^-\rangle$

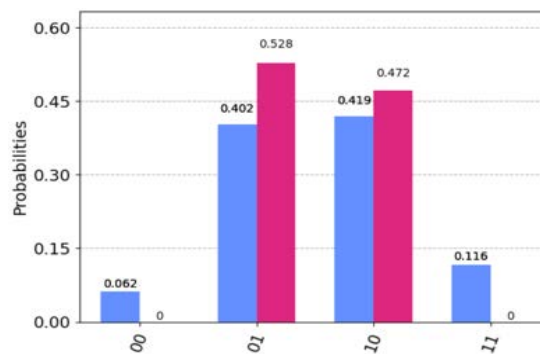
Αντίστοιχα τα ιστογράμματα για κάθε Bell State εμφανίζονται στις εικόνες 76,77,78:

- Το ιστόγραμμα για την $|\Phi^+\rangle$ είναι το εξής στην εικόνα 76:



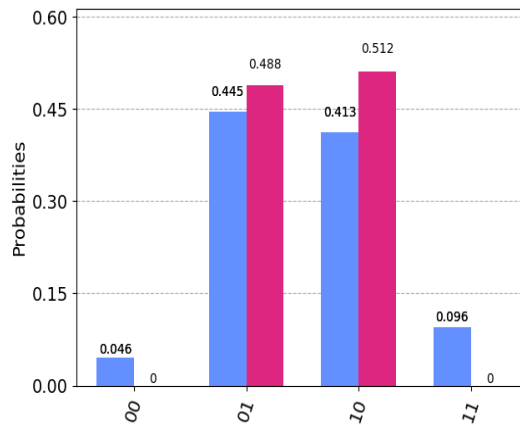
Εικόνα 76: Ιστόγραμμα για $|\Phi^+\rangle$

- Το ιστόγραμμα για την $|\Psi^+\rangle$ είναι το εξής στην εικόνα 77:



Εικόνα 77: Ιστόγραμμα για $|\Psi^+\rangle$

- Το ιστόγραμμα για την $|\Psi^-\rangle$ είναι το εξής στην εικόνα 78:



Εικόνα 78: Ιστόγραμμα για $|\Psi^-\rangle$

3.3.12 Παράλληλες Πύλες

Ένα τελευταίο είδος πυλών είναι οι παράλληλες πύλες, το τανυστικό γινόμενο δύο ή περισσότερων πυλών δημιουργεί μια έννοια πύλη που είναι αποτέλεσμα των δύο παράλληλων πυλών. Για παράδειγμα ένα κύκλωμα όπως αυτό της εικόνας μαθηματικά ερμηνεύεται με τα εξής βήματα .

Βήμα 1: Ευρεση κβαντικού καταχωρητή

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Βήμα 2: Δημιουργία πρώτης παράλληλης πύλης ώστε να μπορεί να εφαρμοστεί στον καταχωρητή των 2 qubit

$$H \otimes I |00\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Βήμα 3: Δημιουργία δεύτερης παράλληλης πύλης ώστε να μπορεί να εφαρμοστεί στον καταχωρητή των 2 qubit

$$H \otimes I |10\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} * \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \end{pmatrix}$$

Βήμα 4: Εφαρμογή CX στον καταχωρητή του προηγούμενου βήματος

$$cX = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} |00\rangle - |11\rangle$$

4 Αλγοριθμική μουσική

Ως αλγόριθμος χαρακτηρίζεται ένα σύνολο οδηγιών που χρησιμοποιούνται για την επίλυση ενός προβλήματος. Ως μουσική ορίζεται μια τέχνη που συντονίζει τον ήχο με αποτέλεσμα την δημιουργία ενός ακουστικά ορθού αποτελέσματος. Οπότε αλγοριθμική μουσική είναι το είδος της μουσικής που χρησιμοποιεί ελάχιστα ή και καθόλου τα κλασσικά μουσικά όργανα για την δημιουργία ενός αποτελέσματος

4.1 Εισαγωγή στην Μουσική

Η τέχνη είναι πίστη και επιβάλλει το καθήκον να αγνοήσεις την κοινή γνώμη.

Μια από τις αρχαιότερες τέχνες είναι η μουσική η οποία απαντάται σε κάθε πολιτισμό και κάθε κουλτούρα. Συνεχώς προσαρμόζεται και μεταβάλλεται από την εκάστοτε κοινωνία και την πάροδο των χρόνων

Τι είναι μουσική?

Ως μουσική ορίζεται ένας ήχος που παράγει ένα σύνολο αντικείμενων.

Δηλαδή ως ήχος ορίζεται ένα αόρατο κύμα το οποίο προκύπτει μέσα από δονήσεις ενός αντικειμένου όταν αυτό κινείται. Στον αέρα υπάρχουν μόρια τα οποία έχουν ίδια κατεύθυνση με το κύμα κατά την εκτέλεση του, διότι το κύμα είναι διάμηκες.

Οπότε οι δονήσεις που το δημιουργούν σπρώχνουν τα μόρια του αέρα και έτσι μεταδίδονται σε όλο τον χώρο. Όσο μεγαλύτερη είναι η δόνηση τόσο μεγαλύτερη είναι η πίεση που ασκείται στα σωματίδια άρα και ο ήχος που παράγεται θα είναι πιο έντονος.

Η μέτρηση της έντασης του ήχου ενός κύματος γίνεται με το μέτρο πλάτος του ήχου(amplitude) και μονάδα μέτρησης του είναι το dB.

Ενώ το μέτρο που καθορίζει πόσες φορές θα επαναληφθεί το κύμα είναι η συχνότητα(frequency) και μονάδα μέτρησης της είναι το Hz.

Σχετικά με την σύνθεση ενός μουσικού κομματιού απαραίτητη είναι μια μελωδία η οποία περιέχει ένα σύνολο από νότες. Το σύνολο έχει την δυνατότητα να εκτελεστεί αποκλειστικά αυτό και κανένα άλλο κατά την διάρκεια της μελωδίας, δείχνοντας με αυτόν τον τρόπο την μουσική εξέλιξη του κομματιού.

Ένα χαρακτηριστικό παράδειγμα μελωδίας προέρχεται μέσα από το τραγούδι των γενέθλιών με την επανάληψη του ίδιου μουσικού συνόλου.

Σε μια μελωδία παρατηρείται και ο ρυθμός δηλαδή ο τρόπος με τον οποίο οι νότες μεταβάλλονται μέσα στον χρόνο και τοποθετούνται στον χώρο. Παράδειγμα ρυθμού θα θεωρηθεί η αλλαγή στην χροιά της φωνής κατά την διάρκεια μιας ομιλίας. Άρα η φωνή όπως και μια μουσική νότα περιέχει ύψος το οποίο καθορίζεται από την συχνότητα του ήχου. Όσο μεγαλύτερη είναι η συχνότητα του ήχου τόσο αυξάνεται και το ύψος της κάθε νότας, το ύψος δεν μπορεί να ορίσει την ακουστική ποιότητα.

Η ποιότητα του ήχου διαμορφώνεται από τον μουσικό τόνο. Ένας τόνος μπορεί να είναι απλός και να περιέχει μόνο το ύψος μιας μουσικής νότας ή σύνθετος και να περιέχει πάνω από μια μουσικές νότες. Στον σύνθετο τόνο γίνεται διαχωρισμός των νοτών, δημιουργείται ένα σύνολο από “καλές νότες”, όπου μόνο αυτές θα εκτελούνται ώστε να επιτευχθεί καλύτερη ποιότητα ήχου.

Εκτός από την μελωδία σημαντικό ρόλο σε ένα μουσικό κομμάτι έχει και η αρμονία. Ο ρόλος της είναι να ταιριάζει σύνολα από νότες που δεν εφαρμόζονται στην μελωδία και να τα εκτελέσει παράλληλα. Αρμονία παρατηρείτε σε ένα τραγούδι όταν ο ρυθμός παραμένει σταθερός αλλά η συχνότητα του ήχου μεταβάλλεται.

4.1.1 Μουσική κρυπτογραφία

Στον κλάδο της επιστήμης και ειδικότερα στον κλάδο της επιστήμης των υπολογιστών εντάσσεται η κρυπτογραφία, η οποία διατηρεί προσωπικές πληροφορίες ασφαλής και επιτρέπει μόνο στο αποστολέα και στον δεκτή να δει το περιεχόμενο

Η κρυπτογραφία δημιουργήθηκε στην αρχαία Ρώμη με σκοπό άνθρωποι αριστοκρατικής καταγωγής να έχουν ένα δικό τους κώδικα επικοινωνίας ο οποίος δεν θα γίνεται αντιληπτός από τον λαό.

Ο απλούστερος τρόπος κρυπτογράφησης είναι με αντικατάσταση.

Όπου η μετατόπιση του κάθε γράμματος από την αρχική θέση του κειμένου στην θέση του κρυπτοκειμένου γίνεται με την βοήθεια ενός κλειδιού, όταν δεν υπάρχει πρόσβαση στο κλειδί τότε δεν μπορεί να διαβαστεί το κρυπτοκείμενο. Η μέθοδος αυτή ονομάστηκε κρυπτογράφηση του Καίσαρα.

Πέρα από τους μουσικούς συνθέτες η μουσική κρυπτογραφία χρησιμοποιήθηκε τον 15ο αιώνα και από την αστυνομία της Νέας Υόρκης. Δημιούργησε ένα κωδικοποιημένο μήνυμα με κλειδί μια μελωδία. Σκοπός δημιουργίας του μουσικού μηνύματος ήταν η παρακολούθηση από τους πράκτορες των παράνομων στοιχημάτων.

Υπάρχουν πολλές περιπτώσεις χρήσης μουσικής κρυπτογραφίας, ακόμα και από μουσικούς συνθέτες, για απόκτηση προσωπικών πληροφοριών. Καμία όμως φήμη δεν είναι επιβεβαιωμένη

Σε σχέση με τους υπόλοιπους συνθέτες που χρησιμοποίησαν την μουσική κρυπτογραφία ως μέσο ψυχαγωγίας ο Γερμανός συνθέτης Johann Sebastian Bach δημιούργησε ένα μουσικό κρυπτογράμμα το οποίο έμοιαζε με κανονικό. Χρησιμοποιώντας την γερμανική αλφάβητο και γνωρίζοντας πως μια μουσική κλίμακα περιέχει επτά νότες δημιούργησε ένα πίνακα. Όπου στα πρώτα επτά κελιά τοποθέτησε τα γράμματα Α μέχρι Η και κάθε γράμμα το αντιστοίχησε με μια νότα. Ενώ τα γράμματα στα των κάτω γραμμών αντιστοιχούν στην ίδια νότα με το κελί που βρίσκεται από πάνω τους. Και ήταν ο πρώτος συνθέτης που με την χρήση κρυπτογράμματος σύνθεσε την μουσική του. Το κρυπτογράμμα του Bach αναπαρίσταται στην εικόνα 79.0



Εικόνα 79.0: Κρυπτόγραμμα του Bach (What is a Musical Cryptogram? Using a Cryptogram Maker | Blog (audiocipher.com))

Τον 17ο αιώνα ο μαθηματικός και John Wilinkins και ο Philip Thicknesse χρησιμοποιούν ως κλειδί κρυπτογράφησης των μηνυμάτων την μουσική. Και υποστηρίζουν πως μια μελωδία δύσκολα θα γίνει αντιληπτή από τον μέσο ακροατή ως τρόπο κωδικοποίησης μηνυμάτων. Η δημιουργία όμως ένα μουσικού κρυπτογράφματος μπορεί να γίνει αντιληπτή από κάποιον που είναι εξοικειωμένο με την μουσική καθώς ανάμεσα στις νότες υπάρχουν μεγάλα διαστήματα τα οποία κανονικά δεν υφίστανται.

Ο Thicknesse δημιουργεί την ίδια χρονική περίοδο ένα μουσικό αλφάβητο ή αλλιώς ένα μουσικό κρυπτογράμμα όπου κάθε γράμμα της αλφαβήτου αντιστοιχεί σε μια μουσική νότα. Πεισμένος ότι στις μουσικές νότες πέρα από αρμονία είναι δυνατή και η αποθήκευση της μουσικής αλφαβήτου. Το μουσικό κρυπτογράμμα αναπαρίσταται στην εικόνα 79.1



Εικόνα 79.1: Μουσικό κρυπτογράμμα Wilinkins και Thicknesse (What is a Musical Cryptogram? Using a Cryptogram Maker | Blog (audiocipher.com))

Στα μέσα του 17^{ου} αιώνα ένας ακόμα μουσικός συνθέτης ο Βόλφανγκ Αμαντέους Μότσαρτ δημιούργησε μουσική με έναν ξεχωριστό τρόπο αυτός ήταν μέσα από ένα παιχνίδι. Σύμφωνα με το παιχνίδι ο μουσικός έριχνε δύο ζάρια δεκαέξι φορές το καθένα έχοντας δημιουργήσει τον πίνακα της εικόνας 79.2, επέλεγε την νότα που αντιστοιχούσε στην κάθε στήλη. Η επιλογή της στήλης προκύπτει από το άθροισμα που θα φέρουν τα δύο ζάρια για αυτό και η πρώτη στήλη περιέχει αριθμούς από το 2 μέχρι το 12 ενώ η πρώτη γραμμή περιέχει τις φορές που θα γίνει ρίψη των δύο ζαριών.

Όσες περισσότερες φορές αναπαραχθεί η κάθε στήλη τόσο μεγαλύτερη θα είναι κι η συχνότητα με την οποία θα ακουστεί η νότα.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	96	22	141	41	105	122	11	30	70	121	26	9	112	49	109	14
3	32	6	128	63	146	46	134	81	117	39	126	56	174	18	116	83
4	69	95	158	13	153	55	110	24	66	139	15	132	73	58	145	79
5	40	17	113	85	161	2	159	100	90	176	7	34	67	160	52	170
6	148	74	163	45	80	97	36	107	25	143	64	125	76	136	1	93
7	104	157	27	167	154	68	118	91	138	71	150	29	101	162	23	151
8	152	60	171	53	99	133	21	127	16	155	57	175	43	168	89	172
9	119	84	114	50	140	86	169	94	120	88	48	166	51	115	72	111
10	98	142	42	156	75	129	62	123	65	77	19	82	137	38	149	8
11	3	87	165	61	135	47	147	33	102	4	31	164	144	59	173	78
12	54	130	10	103	28	37	106	5	35	20	108	92	12	124	44	131

Εικόνα 79.2: Πίνακας για το ζάρι του Mozart

(https://mathhmagic.blogspot.com/2012/08/blog-post_25.html)

4.1.2 Η άνθηση της αλγοριθμικής μουσικής

Καθοριστικό ρόλο στην εξέλιξη της μουσικής είχαν οι κλάδοι της φυσικής αλλά και των μαθηματικών.

Στο έργο Symphony No.29 του συνθέτη Wolfgang Amadeus Mozart είναι εμφανής η επιρροή του από τον τρίτο νόμο του φυσικού Isaac Newton.

Καθώς η πρώτη μουσική περίοδος της συμφωνίας ερμηνεύεται ως την δύναμη που ασκείτε σε ένα αντικείμενο, ενώ η δεύτερη περίοδος της συμφωνίας ερμηνεύεται ως την πίεση που θα ασκούσε το αντικείμενο πίσω σε αυτόν που θα το έσπρωχνε.

Μέσα από την χρήση παλινδρομικών αριθμών⁵ στις αρχές του 19^{ου} αιώνα ο συνθέτης Anton Webern δημιούργησε το έργο του Symphony Op21.

Μια συμφωνία που ακουγόταν διαφορετική από κάθε άλλη συμφωνία που είχε δημιουργηθεί και που οδήγησε στην δημιουργία ενός νέου μουσικού είδους της μουσικής προς ανακάλυψη (music of exploration).

Μιας μουσικής δηλαδή που εμφανίζεται μέχρι και σήμερα μέσα από τα μουσικά είδη (progressive rock και storm rock) δημιουργώντας στον ακροατή την επιθυμία να ζητήσει την πηγή προέλευσης της μελωδίας αλλά και τους λόγους δημιουργίας της. Δημιουργός της αλγοριθμικής μουσικής θεωρήθηκε ο Ιάnnης Ξενάκης. Επηρεασμένος από την θεωρία των «**μεγάλων αριθμών**»⁶, ο Ξενάκης οδηγήθηκε στην στη δημιουργία μια νέας μουσικής της στοχαστικής μουσικής. Όπου στοχαστικά μοντέλα, ⁷η θεωρία των πιθανοτήτων και οι νόμοι της στατιστικής, δημιουργούν ηχητικά συμβάντα τα οποία είναι μεμονωμένοι ήχοι.

Εκτός από τα στοχαστικά μοντέλα, δημιούργησε μουσικό έργο μέσα από την θεωρία παίγνιων παράγοντας ένα έργο όπου υπάρχουν δύο μουσικοί και ο ένας δρα επηρεαζόμενος από τις επιλογές του άλλου μουσικού. Οι μαθηματικοί υπολογισμοί που πραγματοποίησε ο Ξενάκης έγιναν μέσα από τον υπολογιστή ILLIAC⁸, τον οποίο χρησιμοποίησαν οι Lejaren Hiller και Leonard Isaacson ώστε να πράξουν αλγοριθμική μουσική. Αποτέλεσμα των τριών μουσικών ήταν η δημιουργώντας με αυτόν τον τρόπο ένα νέο είδος μουσικής το οποίο παράγεται μέσα από τον υπολογιστή οδήγησε στην εξέλιξη της μουσικής.

4.1.3 Η εξέλιξη της αλγοριθμικής μουσικής

Όπως αναφέρθηκε και νωρίτερα η αλγοριθμική μουσική ξεκίνησε από πολύ παλιά με κύριο σκοπό να παράξει μελωδίες οι οποίες

⁵ Παλινδρομικοί αριθμοί: Ένα σύνολο αριθμών που διαβάζονται το ίδιο αν αντιστρέψουμε την φορά τους πχ 11, 22, 33

⁶ Θεωρία των μεγάλων αριθμών:

⁷ Στοχαστικά Μοντέλα: Μέσα από τα στοχαστικά μοντέλα και ιδιαίτερα στις μαρκοβιανές αλυσίδες δημιουργεί την Μαρκοβιανή στοχαστική μουσική

⁸ ILLIAC: Ήταν ο πρώτος υπολογιστής που κατασκευάστηκε

δεν θα βασίζονται στην αντίληψη του ανθρώπου αποκλειστικά αλλά σε κάποιο μοντέλο(όπως τα στοχαστικά μοντέλα που χρησιμοποίησε ο Ξενάκης). Με την βοήθεια αυτών των μοντέλων είναι εφικτή η δημιουργία καινούριων και διαφορετικών μουσικών έργων.

4.1.3.1 Σκοπός της αλγοριθμικής μουσικής

Η αλγοριθμική μουσική βοηθάει στην ανάπτυξη νέων συνθέσεων καθώς παράγει μουσική μέσα από κανόνες που έχουν προκύψει με την βοήθεια του υπολογιστή. Οι αλγόριθμοι βοηθούν στην μουσική ανάλυση καθώς διαχωρίζουν τα μουσικά χαρακτηριστικά ενός κομματιού τα επεξεργάζονται και βελτιώνουν με αυτό τον τρόπο το αποτέλεσμα. Η επεξεργασία των κομματιών γίνεται με κάποιο DAW. Ακόμα τα αλγοριθμικά συστήματα προσφέρουν στους καλλιτέχνες την δυνατότητα να συνδέουν τα οπτικά εφέ με τον ήχο παράγοντας ένα ιδανικό αποτέλεσμα στις ζωντανές τους παραστάσεις ένα τέτοιο παράδειγμα απεικονίζεται στην εικόνα.

Ο συνδυασμός του φωτισμού με τα οπτικά εφέ και την αλγοριθμική μουσική πραγματοποιούνται σε παραστάσεις που ονομάζονται Algarve και ένα παράδειγμα algarave απεικονίζεται στην εικόνα 80.0



Εικόνα 80.0: Αναπαράσταση Algorave

(Algoraves Put Live Performance Into Programming - SXSW Magazine)

4.1.3.2 Οφέλη της αλγοριθμικής μουσικής

Βασικό όφελος της αλγοριθμικής μουσικής είναι πώς είναι αποτελεσματική καθώς έχει την δυνατότητα να παράγει ένα μεγάλο αριθμό τραγουδιών.

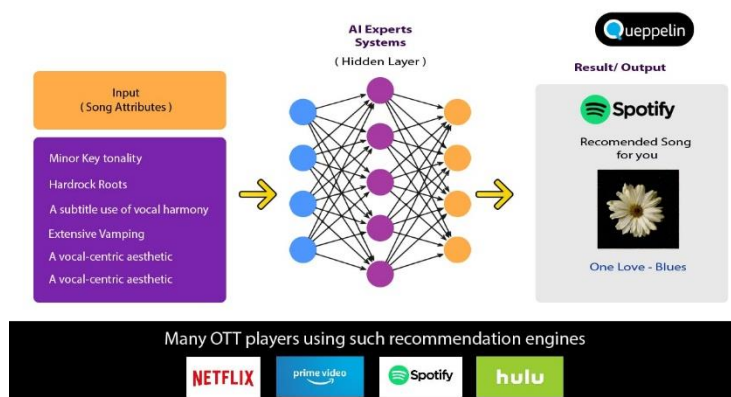
Είναι απεριόριστη καθώς μπορεί να επεκταθεί σε όλα τα είδη μουσικής και να παράξει μουσική μέσα από την σύνδεση τους.

Παράλληλα η αλγοριθμική μουσική μπορεί να ενσωματωθεί σε διάφορες εφαρμογές και παιχνίδια. Οπού ανάλογα με κάθε αλλαγή πίστας υπάρχει και η αντίστοιχη αλλαγή στην μουσική της προσφέροντας με αυτόν τον τρόπο ένα ξεχωριστό ήχο για τον παίκτη.

4.1.3.3 Εφαρμογές της αλγοριθμικής μουσικής

Τα νευρωνικά δίκτυα είναι δίκτυα αποτελούμενα από κόμβους ή και νευρώνες οι οποίοι συνδέονται μεταξύ τους.

Τα νευρωνικά δίκτυα στην αλγοριθμική μουσική χρησιμοποιούνται κυρίως σε πλατφόρμες όπως το Spotify με τον εξής τρόπο. Κάθε τραγούδι μετατρέπεται σε μια κυματομορφή όπου επεξεργάζεται από τα νευρωνικά δίκτυα, σε αυτές τις κυματομορφές αποδίδονται παράμετροι με βάση τις οποίες το Spotify ταιριάζει τα τραγούδια και προτείνει κάθε φορά τα κατάλληλα στους ακροατές. Αναπαράσταση ενός νευρωνικού δικτύου είναι στην εικόνα 80.1



Εικόνα 80.1: Τα νευρωνικά δίκτυα στην αλγοριθμική μουσική

(<https://www.queppelin.com/recommendation-engine-using-convolutional-neural-networks-collaborative-filtering-nlp/>)

Η αλγοριθμική μουσική χρησιμοποιείται και ως μέσο για την εκμάθηση μουσικής στα σχολεία.

Μέσω της οποίας τα παιδιά αναπτύσσουν τις καλλιτεχνικές τους δεξιότητες, ακόμα αντιλαμβάνονται καλύτερα μαθήματα όπως τα μαθηματικά καθώς υπάρχουν οι έννοιες των συχνοτήτων και των αλγορίθμων που είναι συνδεδεμένες με τις μαθηματικές ορολογίες.

Η αλγοριθμική μουσική προσφέρει εμπειρία και στον προγραμματισμό καθώς και οι δύο κλάδοι περιέχουν κοινούς όρους όπως τις επαναλήψεις, ακόμα για την μάθησης μουσικής και προγραμματισμού προσφέρονται σεμινάρια στα σχολεία με την χρήση διάφορων λογισμικών αλλά όμως το πιο διαδεδομένο είναι η Sonic Pi

4.2 Sonic Pi

Η Sonic Pi σχεδιάστηκε για να τρέχει σε Raspberry Pi. Είναι ένα περιβάλλον ζωντανής κωδικοποίησης(live coding) και λειτουργεί με την γλώσσα προγραμματισμού Ruby.

Ο λόγος που χρησιμοποιείται η Ruby είναι διότι είναι μια φιλική προς τον χρήστη γλώσσα προγραμματισμού και εύκολή στην κατανόηση.

Η Sonic Pi δημιουργήθηκε το 2012 από τον Sam Aaron και ο βασικός λόγος δημιουργίας της ήταν για να μπορέσουν οι μαθητές μέσα από τον προγραμματισμό να μάθουν μουσική και το αντίστροφο στην συνέχεια εξελίχθηκε σε ένα εργαλείο το οποίο μπορεί να χρησιμοποιήσει οποιοσδήποτε θέλει να δημιουργήσει μουσική.

4.2.1 Εισαγωγή στο Live Coding και στο Super Collider

Τι είναι το Live Coding

Το Live Coding είναι ένα είδος προγραμματισμού στο οποίο ο καλλιτέχνης/προγραμματιστής συνθέτει το πρόγραμμα του χωρίς οποιαδήποτε διακοπή, ενώ αυτό εκτελείται.

Λειτουργεί με just in time μεταγλωττιστές, όπου αυτό σημαίνει όπως αναφέρθηκε και πριν ότι το πρόγραμμα τροποποιείτε και μεταβάλλεται την ώρα που εκτελείται χωρίς να σταματήσει.

Συνήθως το Live coding είναι ένα οπτικό έργο στο οποίο αναπαριστάτε και ο κώδικας που εκτελείτε εκείνη την στιγμή.

Μια αναπαράσταση του live coding βρίσκετε στην εικόνα 81



Εικόνα 81: Live Coding και Algarve παράσταση (<http://www.sonicwriting.org/blog/live-coding>)

Πως προκύπτει μουσική μέσα από την Sonic Pi?

Η μουσική στην Sonic Pi η παράγεται αποκλειστικά μέσα από το περιβάλλον Super Collider.

Το super collider δημιουργήθηκε το 1996 από τον James McCartney θεωρείτε ένα προγραμματιστικό περιβάλλον το οποίο χρησιμοποιείτε αποκλειστικά για την παραγωγή μουσικής μέσα από αλγόριθμους.

Παρέχει ένα framework το οποίο αποκλειστικά χρησιμοποιείται για αλγοριθμική μουσική.

Το συγκεκριμένο περιβάλλον δεν χρησιμοποιείτε για να αναπαράγει ήχο ακριβώς ίδιο με αυτόν που θα δημιουργούσε ένας μουσικός με μουσικά όργανα, χωρίς όμως αυτό να σημαίνει πως ο ήχος που προκύπτει από το συγκεκριμένο περιβάλλον δεν είναι επηρεασμένος από την μουσική που δεν παράγεται από κάποιο περιβάλλον παραγωγής ηλεκτρονικής μουσικής. Ο ήχος που δημιουργείτε μέσα από το περιβάλλον του Super Collider και μέσα από περιβάλλοντα παρόμοια με αυτό ονομάζεται ηλεκτρονική μουσική και το είδος αυτής της μουσικής μοιάζει περισσότερο στο είδος της rap και hip hop μουσικής και ονομάζεται synthwave.

Το Supercollider διαθέτει server και client όπου δέχεται και στέλνει από και σε άλλα λογισμικά ή μουσικά περιβάλλοντα τις μουσικές πληροφορίες που αναπαρίστανται με MIDI μορφή. Αξίζει να αναφερθεί πως οι MIDI πληροφορίες μπορούν να μεταδοθούν μέχρι και σε 16 διαφορετικά μουσικά όργανα. Στην διαδικασία μεταφοράς των πληροφοριών βοηθάει το πρωτόκολλο Open Sound Control (OSC).

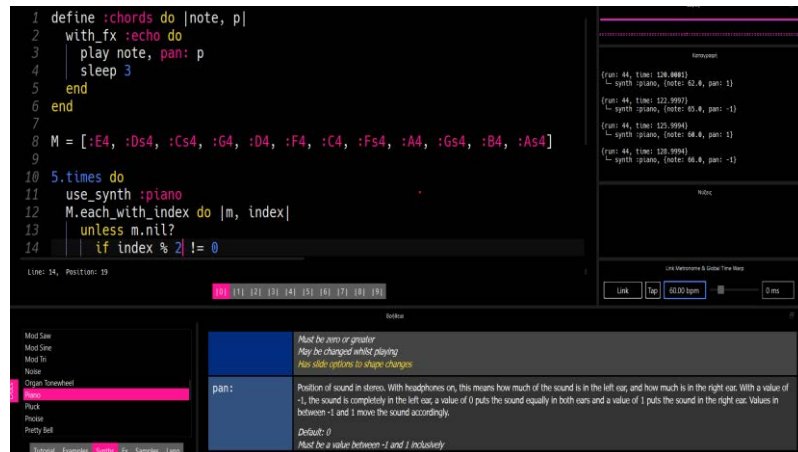
Τι είναι το OSC(Open Sound Control)

Το OSC είναι ένα πρωτόκολλο το οποίο δημιουργήθηκε για να επιτρέπει την επικοινωνία ανάμεσα σε μουσικά όργανα όπως το συνθεσάιζερ με τον υπολογιστή. Μερικές φορές χρησιμοποιείτε αντί για το πρότυπο MIDI δηλαδή μεταφέρει το ίδιο τις μουσικές πληροφορίες που χρειάζονται για την εκτέλεση του μουσικού κομματιού. Ένα μουσικό μήνυμα OSC μεταφέρεται με την βοήθεια UDP/IP δηλαδή μέσω πρωτοκόλλων που μεταφέρουν τα μηνύματα χρησιμοποιώντας το Internet.

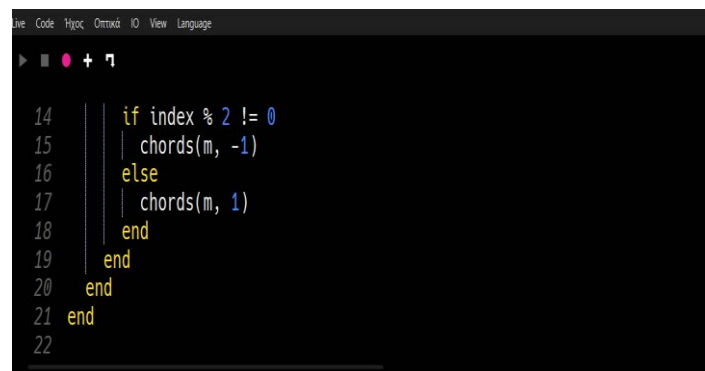
Η Sonic Pi περιέχει OSC πρωτόκολλο μέσω του οποίου επικοινωνεί με το super collider και παράγει την μουσική.

4.2.2 Παραδείγματα χρήσης της Sonic Pi

Στις εικόνες 82 και 83 ακολουθούν παραδείγματα στην Sonic Pi



Εικόνα 82: Εκτέλεση προγράμματος σε Sonic Pi



Εικόνα 83: Συνέχεια εκτέλεσης προγράμματος σε Sonic Pi

Γραμμή 1: Εισαγωγή μιας συνάρτησης που περιέχει δύο μεταβλητές

Γραμμή 2: Πρόσθεση ηχητικού εφέ δηλαδή τόνου στην μελωδία που θα παραχθεί

Γραμμή 3; Εκτέλεση νότες και ανάλογα με την τιμή που θα προκύψει 1 ή -1 η νότα θα αναπαραχθεί από αριστερά ή από δεξιά η λειτουργία αναπαραγωγής της νότας από αριστερά η δεξιά γίνεται με την εντολή pan και είναι περισσότερο αντιληπτή με την χρήση ακουστικών

Γραμμή 4: Εκτέλεση της νότας κάθε τρία δευτερόλεπτα

Γραμμή 8: Λίστα από νότες η οποία θα αποτελέσει μεταβλητή εισόδου στην συνάρτηση

Γραμμή 10: Εκτέλεση 5 φορές την μελωδία

Γραμμή 11: Χρήση μουσικού ήχου με βάση του οποίου θα αναπαραχθούν οι νότες

Γραμμή 12: Για κάθε στοιχείο της λίστας επιλέγεται το στοιχείο και η θέση που βρίσκεται στην λίστα

Γραμμή 13: Όσο το στοιχείο δεν είναι 0

Γραμμή 14: Αν το στοιχείο δεν βρίσκεται σε ζυγό αριθμό θέσης

Γραμμή 15: Εκτέλεση της συνάρτησης όπου θα αναπαριστώνται από την αριστερή μεριά τα στοιχεία που δεν βρίσκονται σε ζυγό αριθμό θέσης

Γραμμή 16 – 17 : Εκτέλεση της συνάρτησης όπου θα αναπαριστώνται από την δεξιά μεριά τα στοιχεία που βρίσκονται σε ζυγό αριθμό θέσης

Ένα άλλο παράδειγμα εκτέλεσης σε sonic pi είναι το εξής στην εικόνα 84:

```
2 with_fx :echo do
3   live_loop :hihat do
4     sample :drum_cymbal_closed, amp: 0.10, rate: 1.5
5     sleep 0.5
6   end
7   sleep 0.6
8   live_loop :bass do
9     use_synth :fm
10    play :D2, release: 0.2
11    sleep 0.5
12  end
13 end
```

Εικόνα 84: Παράδειγμα 2 σε Sonic Pi

Γραμμή 2: Εισαγωγή εφέ που θα χρησιμοποιηθεί για την αναπαραγωγή του αποτελέσματος

Γραμμή 3: Ορισμός μια συνεχόμενης επανάληψης η οποία θα έχει ως ήχο ένα κρουστό όργανο

Γραμμή 4: Αναπαραγωγή ενός έτοιμου μουσικού συνόλου που προσφέρει η sonic pi όπου όσο μεγαλύτερος είναι το πλάτος το οποίο συμβολίζεται με την εντολή amp τόσο πιο έντονα θα ακούγεται ο ήχος καθώς θα είναι μεγαλύτερο το (amplitude) δηλαδή το πλάτος του ήχου και όσο πιο μεγάλη είναι η συχνότητα του ήχου που

αναπαριστάτε με την εντολή rate τότε τόσο μεγαλύτερη θα είναι η οκτάβα στην οποία αναπαριστάτε ο ήχος

Γραμμή 5 : Παύση του ήχου

Γραμμή 6: Σταματάει η αναπαραγωγή της επανάληψης συνεχίζεται όμως να εκτελείται το μουσικό εφέ

Γραμμή 8: Έναρξη επόμενης επανάληψης η οποία θα έχει πιο έντονο ήχο από την προηγούμενη καθώς το μπάσο που χρησιμοποιείτε ως μέσο αναπαραγωγής της επανάληψης έχει βαθύ ήχο

Γραμμή 9: Επιλογή ήχου με βάση τον οποίο θα αναπαραχθούν οι νότες

Γραμμή10: Εκτέλεση (κάθε νότας)

4.2.2.1 Το ζάρι του Mozart

Η sonic pi μέσα από την βοήθεια της βιβλιοθήκης rsonic μπορεί να εκτελεστεί στην python. Το επόμενο παράδειγμα αποτελεί μία προσέγγιση στην αναπαράστασή του ζαριού του Mozart, ο οποίος ήταν ο πρώτος τρόπος παραγωγής μουσικής μέσα από ένα αλγόριθμο.

Καθώς ο μουσικός για να δημιουργήσει την μουσική του χρησιμοποιούσε έναν διαφορετικό τρόπο δηλαδή αντιστοίχησε ένα μουσικό ήχο σε κάθε αριθμό του ζαριού, όσο περισσότερες φορές τύχαινε μία πλευρά του ζαριού τόσο περισσότερο γινόταν αναπαραγωγή του ήχου που αντιστοιχούσε στην πλευρά που κατέληγε το ζάρι.

Το παραπάνω παράδειγμα αναπαρίσταται στις εικόνες 85 , 86 και είναι ένα μη τίμιο ζάρι όπου οι πιθανότητες εμφάνισης της κάθε πλευράς δεν είναι ίδιες το ζάρι είναι ως εξής:

```

2 import random
3
4 from psonic import*
5 c1 = 0
6 c2 = 0
7 c3 = 0
8 c4 = 0
9 c5 = 0
10 c6 = 0
11 p1 = 0
12 p2 = 0
13 p3 = 0
14 p4 = 0
15 p5 = 0
16 p6 = 0
17 N = 0
18
19 def Propability(P):
20     for i in range(6):
21         if P[i] == p1:
22             with Fx(ECHO):
23                 use_synth(SUBPULSE)
24                 play (60 ,amp = P[i] , pan = -1 , release = 2)
25                 sleep(0.5)
26         elif P[i] == p2:
27             with Fx(REVERB):
28                 play (40 ,amp = P[i] , pan = 1 , release = 2)
29                 sleep(0.5)
30         elif P[i] == p3:
31             with Fx(ECHO):
32                 use_synth(SUBPULSE)
33                 play (70 ,amp = P[i] , pan = -1 , release = 2)
34                 sleep(0.5)
35         elif P[i] == p4:
36             with Fx(REVERB):
37                 play (80 ,amp = P[i] , pan = 1 , release = 2)
38                 sleep(0.5)
39         elif P[i] == p5:
40             with Fx(ECHO):

```

Εικόνα 85: Ζάρι του Mozart

```

36         with Fx(REVERB):
37             play (80 ,amp = P[i] , pan = 1 , release = 2)
38             sleep(0.5)
39     elif P[i] == p5:
40         with Fx(ECHO):
41             use_synth(SUBPULSE)
42             play (50 ,amp = P[i] , pan = -1 , release = 2)
43             sleep(0.5)
44     elif P[i] == p6:
45         with Fx(REVERB):
46             play (65 ,amp = P[i] , pan = 1 , release = 2)
47             sleep(0.5)
48
49
50 N = int(input("Input Numbers:"))
51 dice = random.randint(1 , 100)
52 print("Start Rolling")
53
54 for dice in range (N):
55     if dice >=0 and dice <= 10 :
56         c1 += 1
57     if dice > 10 and dice <= 30:
58         c2 += 1
59     if dice > 30 and dice <= 50:
60         c3 += 1
61     if(dice > 50 and dice <= 60):
62         c4 += 1
63     if(dice > 60 and dice <= 65):
64         c5 += 1
65     if (dice < 65 and dice > 90):
66         c6 += 1
67
68
69 #p=[p1,p2,p3,p4,p5,p6]
70 p1 = c1 / N
71 p2 = c2 / N
72 p3 = c3 / N
73 p4 = c4 / N

```

Εικόνα 86: Συνέχεια από το ζάρι του Mozart

Γραμμή2-4: Εισαγωγή Βιβλιοθηκών

Γραμμή 5-17: Αρχικοποίηση των καταχωρητών που μετράνε τις πιθανότητες εμφάνισης κάθε ζαριάς καθώς και το της κάθε πλευράς του ζαριού

Γραμμή 19 – 47: Δημιουργία συνάρτησης η οποία αναπαριστά την κάθε πλευρά του ζαριού ως μια νότα με βάση την πιθανότητα εμφάνισης της. Οπότε οι πλευρές με μεγαλύτερη πιθανότητα εμφάνισης θα ακούγονται πιο έντονα και με πιο υψηλές νότες από ότι οι πλευρές με χαμηλότερη πιθανότητα εμφάνισης και η αναπαράσταση της κάθε νότας πιο έντονα από τις άλλες πραγματοποιείται με την εντολή amp. Ακόμα κάθε νότα θα αναπαριστάτε και με ένα διαφορετικό μουσικό εφέ ενώ οι μονές νότες θα αναπαριστώνται από αριστερά ενώ οι ζυγές από δεξιά.

Γραμμή 50: Ρίψεις του ζαριού

Γραμμή 54 - 65: Δημιουργία του μη τίμιου ζαριού καθώς κάθε πλευρά όπως φαίνεται και στον κώδικα δεν θα έχει ίδια πιθανότητα εμφάνισης με τις άλλες. Οπότε αυξάνετε κατά μία μονάδα κάθε φορά ο καταχωρητής της πλευράς, όταν οι ρίψεις βρεθούν στο πεδίο ορισμού της κάθε πλευράς

Γραμμή 70 – 77: Υπολογισμός πιθανοτήτων εμφάνισης της κάθε πλευράς και δημιουργία λίστας που περιέχει όλες τις πιθανότητες εμφάνισης

Γραμμή 80 – 87: Εμφάνισης των πιθανοτήτων της κάθε πλευράς

Γραμμή 88: Εκτέλεση της μουσικής συνάρτησης

Το αποτελέσματα του παραπάνω προγράμματος φαίνονται στην εικόνα 87:

```
Input Numbers:4
Start Rolling
1 is : 0.250000
2 is : 0.000000
3 is : 0.000000
4 is : 0.000000
5 is : 0.000000
6 is : 0.000000
Stop Rolling
1 is : 0.500000
2 is : 0.000000
3 is : 0.000000
4 is : 0.000000
5 is : 0.000000
6 is : 0.000000
Stop Rolling
1 is : 0.750000
2 is : 0.000000
3 is : 0.000000
4 is : 0.000000
5 is : 0.000000
6 is : 0.000000
Stop Rolling
1 is : 1.000000
2 is : 0.000000
3 is : 0.000000
4 is : 0.000000
5 is : 0.000000
6 is : 0.000000
Stop Rolling
```

Εικόνα 87: Αποτελέσματα του ζαριού του Mozart

Η παραπάνω εικόνα απεικονίζει την εκτέλεση ενός μη τίμιου ζαριού. Το οποίο εκτελέστηκε τέσσερις φορές, ο αριθμός των ρίψεων βρίσκεται στο σύνολο τιμών της πρώτης πλευράς του ζαριού για αυτό αυξάνεται μόνο αυτή η πιθανότητα εμφάνισης κάθε φορά.

4.2.2.2 Μουσικό Bit Sequence

Ένα ακόμα παράδειγμα το οποίο μπορεί να αναπαρασταθεί μέσα από την αλγοριθμική μουσική είναι η εύρεση ενός κρυφού bit sequence το οποίο αναπαρίσταται στην εικόνα 88:

```
1 from psonic import *
2
3 def loop():
4     use_synth(HOLLOW)
5     with Fx(SLICER):
6         sample(AMBI_PIANO , amp = 1 , release = 2)
7
8
9 a = '111001011011010' #secret
10 b = '' #guess
11 c=list(a)
12 d=list(b)
13
14 for i in range(len(a)):
15     d.append(0)
16     if (c[i] != d[i]):
17         d[i] = int(c[i]) + int(d[i])
18     elif (c[i] == d[i]):
19         d[i] = int(c[i])
20
21     print(d[i])
22 lenght = len(d)
23
24 for i in range(len(d)):
25     loop()
26     if (d[i] == 1):
27         with Fx(ECHO):
28             play(50,pan = 1,attack = 0.3,release = 3 , amp = 1)
29             sleep(1)
30     elif d[i] == 0:
31         play(80,pan = -1,attack = 0.3 , amp = 1)
32         sleep(1)
33     print(d[i])
```

Εικόνα 88: Μουσικό bit sequence

Γραμμή 1: Εισαγωγή βιβλιοθηκών

Γραμμή 3-7: Δημιουργία συνάρτησης η οποία πριν την εκτέλεση του μουσικού bit sequence θα αναπαράγει μια μελωδία

Γραμμή 9: Δημιουργία ενός κρυφού bit sequence

Γραμμή 10: Αρχικοποίηση του υποτιθέμενου bit sequence

Γραμμή 11-12: Δημιουργία λίστας για καθένα από τα μηνύματα

Γραμμή 14-15: Τοποθέτηση μηδενικών στην υποτιθέμενη λίστα τόσα όσο το μήκος της κρυφής λίστας

Γραμμή 16-19: Σύγκριση των στοιχείων της κάθε λίστας αν τα στοιχεία του είναι ίδια δηλαδή μηδέν τότε το στοιχείο της λίστας που αποτελεί το τελικό κρυφό μήνυμα δεν αλλάζει, αλλιώς αλλάζει και γίνεται ένα

Γραμμή 24-32: Εκτέλεση της μουσικής συνάρτησης, αν το στοιχείο είναι ένα εκτελείτε μια μελωδία από τα δεξιά αλλιώς αν είναι μηδέν τότε εκτελείται μια μελωδία από τα αριστερά.

Σκοπός της διαφορετικής μελωδίας που εκτελείτε είναι να γίνεται αντιληπτό πότε το bit είναι 0 και πότε 1.

4.2.2.3 MP3 Player

Στις εικόνες 89 , 90 , 91 αναπαριστάτε ο κώδικας για το MP3 Player

```
3 from gtts import gTTS
4 from pygame import mixer
5 import time
6 import pyttsx3
7 from tkinter import *
8 from psonic import *
9 import socket
10 import random
11 import time
12 from pythonosc import udp_client
13
14 address = socket.gethostname(socket.gethostname())
15 port = 4560
16 client = udp_client.SimpleUDPClient(address, port)
17
18 def start_song():
19     mixer.init()
20     mixer.music.load("Music.wav")
21     mixer.music.play()
22
23 def pause_song():
24     mixer.music.pause()
25     mixer.music.get_busy()
26
27 def rewind_song():
28     restart = sample(VINYL_REWIND)
29     client.send_message("/Sample",restart)
30
31     mixer.init()
32     mixer.music.load("Music.wav")
33     mixer.music.play()
34     mixer.music.rewind()
35     mixer.music.get_busy()
36
37 def stop_song():
38     mixer.music.stop()
39
```

Εικόνα 89: Δημιουργία MP3 Player

Γραμμή 1-11: Εισαγωγή απαραίτητων βιβλιοθηκών

Γραμμή 14: Εύρεση IP διεύθυνση ώστε να επικοινωνήσουν με UDP το περιβάλλον της Sonic Pi με το jupyter notebook και στην συνέχεια με OSC μηνύματα να προκύψει η μουσική

Γραμμή 15: Διεύθυνση της Sonic Pi

Γραμμή 16: Δημιουργία ‘πελάτη’ ο οποίος θα στείλει το μήνυμα στο server της Sonic Pi όταν του δοθεί η εντολή.

Γραμμή 18: Συνάρτηση η οποία ανοίγει το music player

Γραμμή 19: Δημιουργία music player με την χρήση του mixer από την βιβλιοθήκη py game

Γραμμή 20: Εισαγωγή τραγουδιού

Γραμμή 21: Αναπαραγωγή τραγουδιού

Γραμμή 23: Συνάρτηση παύσης τραγουδιού

Γραμμή 24 - 25: Παύση τραγουδιού

Γραμμή 27: Συνάρτηση επανεκκίνησης τραγουδιού

Γραμμή 28: Αναπαραγωγή ενός μουσικού κομματιού πριν την επανεκκίνηση

Γραμμή 29: Αποστολή μηνύματος στην Sonic Pi ώστε να εκτελέσει τον ήχο της προηγούμενης γραμμής

Γραμμή 31-35: Άνοιγμά ξανά music player και αναπαραγωγή του τραγουδιού

Γραμμή 37: Συνάρτηση παύσης του τραγουδιού

Γραμμή 38: Παύση του τραγουδιού

```
40
41 if __name__ == "__main__":
42     window = Tk()
43     window.title("Menu")
44     # window.geometry('100x150')
45     Play = Button(window, text = "Play", command = start_song, fg = "white", bg = "#CC0066", width = 80)
46     Play.pack()
47
48     Pause = Button(window, text="Pause", command = pause_song, fg = "white", bg = "#3300FF", width = 80)
49     Pause.pack()
50
51     Rewind = Button(window, text="Rewind", command = rewind_song, fg = "white", bg = "#FF0033", width = 80)
52     Rewind.pack()
53
54     Stop = Button(window, text="Stop", command = stop_song, fg = "white", bg = "#6600FF", width = 80)
55     Stop.pack()
56     window.mainloop()
```

Εικόνα 90: Συνέχεια Mp3 player

Με την εκτέλεση της γραμμής 29 ο αντίστοιχος κώδικας στην Sonic Pi είναι ο εξής της εικόνα

```

3 use_real_time
4 live_loop :Mp3__ do
5
6   use_bpm 60 # Beats per minute
7   restart = sync "/osc*/Sample"
8   restart
9 end

```

Εικόνα 91: Σύνδεση με Sonic Pi για αναπαραγωγή μελωδίας

Γραμμή 4: Επανάληψη της μελωδίας για όσο χρειαστεί

Γραμμή 7: μήνυμα που στάλθηκε από το Jupyter notebook στην Sonic Pi

Γραμμή 8: Αναπαραγωγή μηνύματος, το μήνυμα είναι ένα sample όμως δεν θα γίνει χρήση της εντολής `sample :vinyl_rewind`, καθώς η εντολή αυτή έχει αποθηκευτεί στην μεταβλητή `restart`

Στην παρακάτω εικόνα αναπαριστάτε και ο κώδικας για την βασική συνάρτηση του μενού ο οποίος είναι:

```

40
41 if __name__ == "__main__":
42     window = Tk()
43     window.title("Menu")
44     # window.geometry('100x150')
45     Play = Button(window, text = "Play", command = start_song, fg = "white", bg = "#CC0066", width = 80)
46     Play.pack()
47
48     Pause = Button(window, text="Pause", command = pause_song, fg = "white", bg = "#3300FF", width = 80)
49     Pause.pack()
50
51     Rewind = Button(window, text="Rewind", command = rewind_song, fg = "white", bg = "#FF0033", width = 80)
52     Rewind.pack()
53
54     Stop = Button(window, text="Stop", command = stop_song, fg = "white", bg = "#6600FF", width = 80)
55     Stop.pack()
56     window.mainloop()

```

Εικόνα 92: Συνέχεια του MP3 Player

Γραμμή 41: Άνοιγμα αρχικής συνάρτησης

Γραμμή 42-43: Δημιουργία του πλαισίου που αναπαριστά το μενού

Γραμμή 45-46: Δημιουργία του κουμπιού που θα ξεκινάει το τραγούδι και είσοδος του στο music player

Γραμμή 48-49: Δημιουργία του κουμπιού που θα κάνει παύση στο τραγούδι και είσοδος του στο music player

Γραμμή 51-52: Δημιουργία του κουμπιού που θα κάνει επανεκκίνηση το τραγούδι και είσοδος του στο music player

Γραμμή 54-55: Δημιουργία του κουμπιού που θα σταματάει το τραγούδι και είσοδος του στο music player

Γραμμή 55-56: Κλείσιμο παραθύρου

Το αποτέλεσμα από την εκτέλεση του παραπάνω κώδικά φαίνεται στην εικόνα 93



Εικόνα 93: Εκτέλεση του MP3 Player

5. Παραγωγή μουσικής με κβαντικούς αλγόριθμους

Ορολογίες όπως κβαντική μηχανική, κβαντική υπολογιστική είναι γνωστές. Αλλά όμως ορολογίες όπως κβαντική ακουστική, κβαντική μουσική και εν τέλει παραγωγή μουσικής από κβαντικούς αλγόριθμους είναι γνωστές? Η κβαντική μουσική υφίσταται όντως?

5.1 Συμπαντική Μουσική

Η έννοια της συμπαντικής μουσικής η αλλιώς μουσικής των σφαιρών είναι μια ιδέα που ξεκίνησε από τον μαθηματικό Πυθαγόρα,

ο οποίος στηρίχτηκε στο γεγονός πως η κίνηση των πλανητών παράγει αρμονικούς ήχους οι οποίοι ερμηνεύονται ως μουσική.

Η μουσική προκύπτει από μαθηματικές πράξεις και συγκεκριμένα ο

Πυθαγόρας υποστήριξε πως το άθροισμα των αριθμών 1,2,3,4 είναι 10 και αυτός ο αριθμός αποτελεί το κλειδί για δημιουργία αρμονικών ήχων.

Γιατί συγκεκριμένα αυτοί οι τέσσερις αριθμοί και όχι κάποιοι άλλοι?

Διότι μουσικές αναλογίες μήκους $2:3^9$, $4:3$, $3:2$ παράγουν από μόνες του ένα αρμονικό αποτέλεσμα οπότε και το άθροισμα τους είναι πάντα ο αριθμός δέκα και είναι το αποτέλεσμα ενός μουσικά αρμονικού συνόλου. Ο Πυθαγόρας και οι ακόλουθοί του γνωστοί και ως Πυθαγόρειοι βασίστηκαν στο γεγονός πως οποιοδήποτε σώμα το οποίο βρίσκεται στην γη κατά την διάρκεια τα κίνησης του μπορεί να παράγει έναν ήχο, (για παράδειγμα οι χορδές μια κιθάρας κατά την ταλάντωση τους παράγουν ήχο ή ένα βέλος όταν περιστρέφεται παράγει έναν ήχο). Οπότε θεώρησαν πως τα ουράνια σώματα δηλαδή οι πλανήτες και οι αστέρες αν κινούνται με ένα συγκεκριμένο ρυθμό μπορούν να πράξουν μια μελωδία.

Και καθώς ο κάθε πλανήτης έχει μορφή σφαίρας η μουσική που παράγεται από αυτούς ονομάστηκε σφαιρική μουσική (music of spheres). Οι Πυθαγόρειοι χρειάστηκαν δέκα ουράνια σώματα ώστε να πράξουν την αρμονική τους μελωδία, δημιούργησαν ένα δέκατο ουράνιο σώμα την αντι-γή, με την δημιουργία του δέκατου πλανήτη δημιουργήθηκαν αρκετές αντιδράσεις. Μια από αυτές τις αντιδράσεις ήταν του Αριστοτέλη ο οποίος υποστήριξε πως δεν είναι δυνατή η παραγωγή μουσικής μέσα από ουράνια σώματα καθώς δεν είναι αντιληπτή από το ανθρώπινο αυτί.

Ως απάντηση στον Αριστοτέλη οι Πυθαγόρειοι ισχυρίστηκαν πως ο ήχος των ουράνιων σωμάτων πότε δεν θα μπορέσει να γίνει αντιληπτός από τον άνθρωπο καθώς από την στιγμή της γέννησης του ακούει εκατομμύρια ήχους χωρίς να έχει την δυνατότητα να τους ερμηνεύσει όλους. Η απάντηση των Πυθαγόρειων στον Αριστοτέλη δημιούργησε ερωτήματα σχετικά με το εάν τελικά η ουράνια μουσική υφίσταται ή όχι.

5.2 Κβαντική ακουστική και κβαντική μουσική

⁹ $2:3$: Μουσική αναλογία που διαβάζεται δύο προς τρία, το ίδιο ισχύει και για αναλογίες $3:4$ κλπ.

Πέρα από την μουσική των σφαιρών, πιο σύγχρονοι και εξελιγμένοι κλάδοι που συνδέουν το πεδίο του ήχου με την επιστήμη είναι η κβαντική ακουστική και η κβαντική μουσική.

Όπως είναι προφανές η κβαντική μουσική προήλθε από την κβαντική ακουστική.

5.2.1.Τι είναι κβαντική ακουστική?

Η κβαντική ακουστική αναφέρεται στην μελέτη του ήχου μέσα από τις κβαντικές έννοιες. Οι οποίες χρησιμοποιούνται ώστε να υπάρξει καλύτερη κατανόηση των ηχητικών κυμάτων όπως αυτών που δημιουργούνται σε πολύ υψηλές ή χαμηλές συχνότητες και η κλασική φυσική αδυνατεί να εντοπίσει. Μέσω της κβαντικής ακουστικής γίνονται προσπάθειες να συνδεθούν φωνόνια με ηχητικά κύματα και συγκεκριμένα με επιφανειακά ακουστικά κύματα (SAW) ώστε να μπορεί να γίνει κατανόηση των σωματιδίων του αέρα ¹⁰ από τα φωνόνια και να μπορούν να επηρεάσουν στην συνέχεια τα ηχητικά κύματα.

5.2.1.1 Τι είναι φωνόνια?

Φωνόνια είναι συλλογικές οντότητες που αναπαριστούν τους τρόπους δόνησης των ατόμων μέσα σε σωματίδια όπως ο κρύσταλλος.

Τα άτομα μέσα στα στερεά σωματίδια αναπαριστούν ένα πλέγμα όπου η σύνδεση ανάμεσα στα σωματίδια γίνεται με πολύ μικρά ελατήρια τα οποία προκαλούν διαφορές δονήσεις με αποτέλεσμα το πλέγμα που αναπαριστά την συνολική μάζα των ατόμων να δονείται.

Με αυτό τον τρόπο παράγονται ηχητικά κύματα και κύματα που μεταφέρουν την θερμότητα, όπου κάθε πακέτο κύματος μπορεί να ταξιδέψει μέσα στο στερεό σωματίδιο. Τα πακέτα των ηχητικών κυμάτων στην κβαντική φυσική ονομάζονται φωνόνια.

Οπότε τα φωνόνια θεωρούνται κβάντα δονητικής ενέργειας. Σε ένα στερεό σώμα τα φωνόνια και τα ηλεκτρόνια είναι υπεύθυνα για αυτά, καθώς τα φωνόνια καθορίζουν την ταχύτητα του ήχου και την θερμοκρασία ενώ τα ηλεκτρόνια καθορίζουν τις ηλεκτρικές ιδιότητες των σωμάτων.

¹⁰ Κατανόηση των σωματιδίων του αέρα είναι απαραίτητη καθώς όλα τα ηχητικά κύματα ταξιδεύουν μέσω του αέρα

5.2.1.2 Διαφορές φωνονίων με φωτονίων

- Τα φωτόνια είναι σωματίδια φωτός και ηλεκτρομαγνητικής ακτινοβολίας ενώ τα φωνόνια είναι σωματίδια που καθορίζουν τις δονήσεις σε ένα στερεό πλέγμα.
- Τα φωτόνια αλληλοεπιδρούν ηλεκτρομαγνητικά ενώ τα φωνόνια αλληλοεπιδρούν μέσω των δονήσεων
- Τα φωτόνια μεταφέρουν ηλεκτρομαγνητικές δυνάμεις είναι υπεύθυνα για την διάδοση του φωτός καθώς και για τις αλληλεπιδράσεις ανάμεσα σε φορτισμένα σωματίδια. Ενώ τα φωτόνια μεταφέρουν κυρίως ηχητική και θερμική ενέργεια.

5.2.2 Κβαντική Μουσική

Τι είναι η Κβαντική Μουσική?

Η κβαντική μουσική όπως και όλοι οι κλάδοι της κβαντικής υπολογιστικής είναι φανταστικοί και ακόμα ερευνώνται. Η κβαντική μουσική αποτελεί συνέχεια της κβαντικής ακουστικής καθώς η μουσική είναι διάφορες δονήσεις που καθορίζονται από τις συχνότητες και μεταβάλλονται στον αέρα. Η εξέλιξη της κβαντικής μουσικής δεν απαιτεί μόνο το ταλέντο και τις γνώσεις ενός φυσικού ή ερευνητή που έχει γνώσει πάνω στην κβαντική μηχανική αλλά χρειάζεται και το ταλέντο ενός μουσικού.

Όπου η σύνδεση των κλάδων της κβαντικής φυσικής με την μουσική και η παραγωγή κβαντικής μουσικής δεν αποτελεί ένα εύκολο ζήτημα, καθώς η μελέτη των κβαντικών φαινομένων αποτελεί κυρίως θεωρητικό και όχι πρακτικό κομμάτι. Αν συνεχιστεί εξέλιξη της κβαντικής μουσικής θα περάσει το πειραματικό στάδιο και θα καταφέρει να δημιουργήσει ένα καινούριο και νέο είδος μουσικής το οποίο θα βασίζεται σε αρχές της κβαντικής υπολογιστικής όπως υπέρθεση και διεμπλοκή.

Κβαντική μουσική vs Παραγωγή μουσικής με κβαντικούς αλγόριθμους

Η κβαντική μουσική και η παραγωγή μουσικής από κβαντικούς αλγόριθμους δεν θεωρούνται το ίδιο. Καθώς η κβαντική μουσική δημιουργείτε μέσα από την χρήση των αρχών της κβαντικής υπολογιστικής και συγκεκριμένα για την παραγωγή της χρησιμοποιούνται και μελετώνται κυματοσυναρτήσεις αυτών των αρχών όπως αναφέρθηκε και νωρίτερα.

Ενώ η παραγωγή μουσικής από κβαντικούς αλγόριθμους βασίζεται και αυτήν στις αρχές τις κβαντικής υπολογιστικής άρα και τις κβαντικής μηχανικής όμως παράγεται αποκλειστικά μέσα από ένα ή περισσότερους αλγόριθμους. Όπου η μουσική που θα παιχτεί εξαρτάται από τα δεδομένα που θα επιστρέψει ο αλγόριθμος.

5.3 Κβαντικοί αλγόριθμοι(Quantum Algo-rhythms)

Πριν ξεκινήσει να γίνει αναφορά στον τρόπο που μπορεί να παραχθεί μουσική από κβαντικούς αλγόριθμους θα ακολουθήσει μια μουσική αναπαράσταση ορισμένων κβαντικών πυλών.

5.3.1 Παραγωγή μουσικής από Bell State (Bell State of the art)

Μέσα από τα Bell States ή αλλιώς από τις καταστάσεις διεμπλοκής που υλοποιήθηκαν στον προηγούμενο κεφάλαιο αποστέλλονται τα δεδομένα από το jupyter notebook στην Sonic Pi.

Η αποστολή των δεδομένων πραγματοποιείται με την βοήθεια osc πρωτόκολλου(για την αποστολή των μουσικών δεδομένων) αλλά και με την βοήθεια udp πρωτόκολλου(για την επικοινωνία ανάμεσα στα δίκτυα). Οπότε εάν γίνεται εκτέλεση τους δεύτερου Bell State τότε τα δεδομένα από αυτή την κβαντική κατάσταση θα αποσταλούν στην Sonic Pi στην εικόνα 94 ακολουθεί η αποστολή δεδομένων από το Jupyter Notebook στην Sonic Pi.

```
1 job = execute(qc, Aer.get_backend('qasm_simulator'),shots = 1024).result()
2 answer = job.get_counts()
3
4 address = socket.gethostbyname(socket.gethostname())
5 port = 4560
6 first_states = []
7 final_states = []
8 bell_states_music = list(answer.keys())
9 bell_states_music_phase = list(answer.values())
10
11
12 for i in bell_states_music[0]:
13     first_states.append(i)
14
15 for j in bell_states_music[1]:
16     final_states.append(j)
17
18 Phase_0 = (bell_states_music_phase[0]/1000)
19 Phase_1 = (bell_states_music_phase[1]/1000)
20
21 phase_0 = float(round(Phase_0,1))
22 phase_1 = float(round(Phase_1,1))
23
24 client = udp_client.SimpleUDPClient(address, 4560)
25
26 client.send_message("/first_states",first_states)
27 client.send_message("/final_states",final_states)
28 client.send_message("/phase_0",phase_0)
29 client.send_message("/phase_1",phase_1)
```

Εικόνα 94: Μουσικά Bell States και αποστολή δεδομένων από το Jupyter Notebook στην Sonic Pi.

Γραμμή 1: Εκτέλεση σε κβαντικό προσωμοιωτή

Γραμμή 2: Αποθήκευση δεδομένων

Γραμμή 8: Δημιουργία λίστας η οποία περιέχει τις καταστάσεις που καταλήγουν τα qubit

Γραμμή 9: Δημιουργία λίστας η οποία δείχνει τις πιθανότητες εμφάνισης της κάθε κατάστασης

Γραμμή 12-16: Εισαγωγή των δεδομένων της κάθε κατάστασης σε δύο λίστες όπου μία θα είναι για κάθε κατάσταση

Γραμμή 18-22: Υπολογισμός φάσης, και αποθήκευσης της φάσης κάθε κατάστασης σε μια μεταβλητή. Η φάση κάθε κατάστασης θα έχει ένα δεκαδικό ψηφίο μόνο και αυτό υλοποιείται με την βοήθεια της round.

Γραμμή 21-29: Αποστολή δεδομένων στην Sonic Pi

Υστέρα στις εικόνες 95 , 96 αναπαριστάτε ο κώδικας ο οποίος δείχνει την μουσική που προκύπτει

```
1 use_osc "192.168.1.3", 4560
2 # Example UDP Receiver in Sonic Pi
3 live_loop :udp_bell_10101 do
4   use_real_time
5   first_states = sync "/osc//first_states"
6   final_states = sync "/osc//final_states"
7   phase_0 = sync "/osc//phase_0"
8   phase_1 = sync "/osc//phase_1"
9
10  define :melody_0 do |m1, phase_0|
11    if m1 == "0"
12      with_fx :echo do
13        with_synth :hollow do
14          play 60, phase: phase_0, pan: -1, attack: 0.6, attack_level: 0.8, sustain: 1.5, decay: 1, sustain_level: 0.6, release: 6
15          sleep 0.4
16        end
17      end
18    end
19    if m1 == "1"
20      with_fx :ping_pong do
21        with_synth :piano do
22          play 60, phase: phase_0, pan: 1, attack: 0.7, attack_level: 1, sustain: 0.7, decay: 0.7, sustain_level: 1, release: 9
23          sleep 0.4
24        end
25      end
26    end
27  end
```

Εικόνα 95: Δημιουργία μουσικής για τα Bell State, με βάση τα δεδομένα εισόδου

Γραμμή 5-8: Εισαγωγή δεδομένων από Jupyter Notebook

Γραμμή 10-26: Δημιουργία πρώτης συνάρτησης η οποία ανάλογα με την αρχική τιμή των qubits παράγει και την αντίστοιχη μουσική

Γραμμή 11:Περίπτωση όπου κάποιο από τα 2 qubits έχει ως αρχική κατάσταση την κατάσταση 0

Γραμμή 12: Μουσικό εφέ

Γραμμή 13: Ήχος με τον οποίο θα γίνει αναπαραγωγή της μελωδίας/νότας

Γραμμή 14: Εκτέλεση μια συγκεκριμένης νότας και χρήση παραμέτρων οι οποίες καθορίζουν την μουσική αναπαράσταση της νότας η παράμετρος attack καθορίζει τον χρόνο που θα χρειαστεί ώστε να ακουστεί η αρχή της μελωδίας. Όσο μεγαλύτερη τιμή έχει κάθε παράμετρος τόσο μεγαλύτερη είναι και η διάρκεια της υστερα η παράμετρος attack_level και κάθε level παράμετρος καθορίζει την ένταση

πχ η παράμετρος attack_level καθορίζει την ένταση του attack δηλαδή την ένταση της αρχής.

Στην συνέχεια οι παράμετροι sustain και decay καθορίζουν το πλάτος και την διάρκεια που θα έχει η μελωδία ανάμεσα στην αρχή που είναι το attack και στο τέλος της μελωδίας που είναι το release. Ενώ η παράμετρος release όπως αναφέρθηκε και πριν καθορίζει το τέλος της μελωδίας και όσο μεγαλύτερη είναι η τιμή της παραμέτρου τόσο περισσότερο χρόνο θα κάνει η μελωδία να τελειώσει.

Η παράμετρος pan καθορίζει την κατεύθυνση από την οποία θα ακουστεί η μουσική νότα, εάν η τιμή της παραμέτρου pan είναι 1 τότε η μουσική νότα θα ακουστεί δεξιά, εάν είναι -1 η μουσική νότα θα ακουστεί από αριστερά.

Γραμμή 15: Χρόνος που θα χρειαστεί ώστε να πραγματοποιηθεί η επόμενη επανάληψη της μελωδίας.

Γραμμή 19:Περίπτωση όπου κάποιο από τα 2 qubits έχει ως αρχική κατάσταση την κατάσταση 1

Γραμμή 20: Μουσικό εφέ

Γραμμή 21: Ήχος με τον οποίο θα γίνει αναπαραγωγή της μελωδίας/νότας

Γραμμή 22: Εκτέλεση μια συγκεκριμένης νότας και χρήση παραμέτρων οι οποίες καθορίζουν την μουσική αναπαράσταση της νότας

Γραμμή 23: Χρόνος που θα χρειαστεί ώστε να πραγματοποιηθεί η επόμενη επανάληψη της μελωδίας.

Υστερά στην εικόνα αναπαρίστατε η συνάρτηση που παράγει την μελωδία ανάλογα με την τελική κατάσταση στην οποία καταστασή στην οποία θα καταλήξει το κάθε qubit

```
29
30 define :melody_1 do [m2 , phase_1]
31   if m2 == "0"
32     with_fx :echo do
33       with_synth :hollow do
34         with_synth :dark_ambience do
35           play 60 ,phase: phase_1, pan: -1,attack: 0.6 , attack_level: 0.8, sustain: 0.7, decay: 1, sustain_level: 0.6, release: 3
36           sleep 0.4
37         end
38       end
39     end
40   end
41
42   if m2 == "1"
43     with_fx :ping_pong do
44       with_synth :hollow do
45         play 60 ,phase: phase_1, pan: 1,attack: 1 , attack_level: 0.5, sustain: 1.7, decay: 0.7, sustain_level: 0.8, release: 5, r
46         sleep 0.4
47       end
48     end
49   end
50 end
```

Εικόνα 96: Δημιουργία μουσικής για τα Bell State, με βάση τα δεδομένα εισόδου

Γραμμή 36-46: Δημιουργία πρώτης συνάρτησης η οποία ανάλογα με την αρχική τιμή των qubits παράγει και την αντίστοιχη μουσική

Γραμμή 37:Περίπτωση όπου κάποιο από τα 2 qubits έχει ως τελική κατάσταση την κατάσταση 0

Γραμμή 38: Μουσικό εφέ

Γραμμή 39: Ήχος με τον οποίο θα γίνει αναπαραγωγή της μελωδίας/νότας

Γραμμή 42:Περίπτωση όπου κάποιο από τα 2 qubits έχει ως αρχική κατάσταση την κατάσταση 1

Γραμμή 43: Μουσικό εφέ

Γραμμή 44: Ήχος με τον οποίο θα γίνει αναπαραγωγή της μελωδίας/νότας

Γραμμή 45: Εκτέλεση μια συγκεκριμένης νότας και χρήση παραμέτρων οι οποίες καθορίζουν την μουσική αναπαράσταση της νότας

Γραμμή 46: Χρόνος που θα χρειαστεί ώστε να πραγματοποιηθεί η επόμενη επανάληψη της μελωδίας.

```

51
52   first_states.each do |first,phase_0|
53     |
54     melody_0(first,phase_0)
55     puts "#{first}"
56   |
57   end
58
59   final_states.each do |last,phase_1|
60     |
61     melody_1(last,phase_1)
62     puts "#{last}"
63   |
64   end
65 end

```

Εικόνα 97: Δημιουργία μουσικής για τα Bell State, με βάση τα δεδομένα εισόδου

Γραμμή 52 - 57: Για κάθε στοιχείο από την λίστα που αποτελεί την αρχική κατάσταση εκτέλεση της αντίστοιχης συνάρτησης και εμφάνισης των δεδομένων

Γραμμή 59 - 65: Για κάθε στοιχείο από την λίστα που αποτελεί την τελική κατάσταση εκτέλεση της αντίστοιχης συνάρτησης και εμφάνισης των δεδομένων

Με την εκτέλεση των γραμμών 55 και 62 τα αποτελέσματα που προκύπτουν δείχνουν τα δεδομένα που εισέρχονται από το jupyter notebook

Στις εικόνες 98 , 99 με τα μπλε γράμματα αναγράφονται τα δεδομένα από το jupyter notebook και ακολουθεί η μελωδία που εκτελείται

```

└─ "1"
└─ synth :piano, {note: 60.0, pan: 1, attack: 0.7, attack_level: 1, sustain: 0.7, decay: 0.7, sustain_level: 1, release: 9.0, room: 0.5}
(run: 17, time: 6312.6711, thread: :live_loop_udp_bell_10101)
└─ "1"
└─ synth :dark_ambience, {note: 60.0, pan: -1, attack: 0.6, attack_level: 0.8, sustain: 0.7, decay: 1.0, sustain_level: 0.6, release: 3.0, room: 0.4}
(run: 17, time: 6313.2685, thread: :live_loop_udp_bell_10101)
└─ "0"
└─ synth :dark_ambience, {note: 60.0, pan: -1, attack: 0.6, attack_level: 0.8, sustain: 0.7, decay: 1.0, sustain_level: 0.6, release: 3.0, room: 0.4}

```

Εικόνα 98: Δεδομένα εισόδου

```

└─ synth :dark_ambience, {note: 60.0, pan: -1, attack: 0.6, attack_level: 0.8, sustain: 0.7, decay: 1.0, sustain_level: 0.6, release: 3.0, room: 0.4}
(run: 17, time: 6313.6679, thread: :live_loop_udp_bell_10101)
└─ "0"

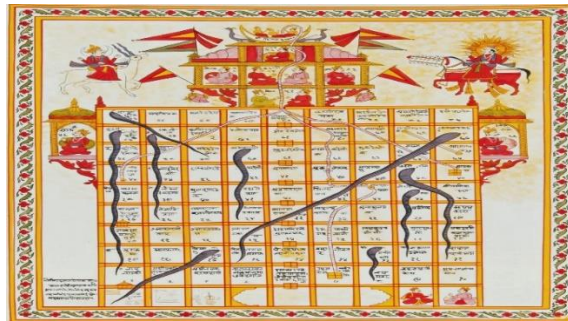
```

Εικόνα 99: Δεδομένα εισόδου

5.3.2 Μαρκοβιανές Αλυσίδες

Μια μαρκοβιανή αλυσίδα είναι ένα στοχαστικό μοντέλο που περιγράφει μια πιθανή σειρά εμφάνισης ενός συνόλου γεγονότων, όπου η πιθανότητα εμφάνισης κάθε γεγονότος εξαρτάται από το γεγονός που εκτελέστηκε πριν. Οι μαρκοβιανές αλυσίδες δημιουργήθηκαν από τον μαθηματικό Andrey Markov έχουν εφαρμογές σε διάφορους τομείς όπως στα τυχερά παιχνίδια, καθώς αυτά βασίζονται σε πιθανότητες.

Ένα παιχνίδι τα οποίο χρησιμοποιεί τις μαρκοβιανές αλυσίδες είναι το φίδια και σκάλες όπου κάθε φορά ο παίκτης μετακινείται στην επόμενη κατάσταση με βάση τις πιθανότητες που του δίνονται από την προηγούμενη κατάσταση. Το παιχνίδι φίδια και σκάλες αναπαρίσταται στην εικόνα 100



Εικόνα 100:Αναπαράσταση παιχνιδιού φίδια και σκάλες
(https://en.wikipedia.org/wiki/Snakes_and_ladders)

Πέρα από τα παιχνίδια οι μαρκοβιανές αλυσίδες χρησιμοποιούνται στην μουσική και κυρίως στην αλγοριθμική μουσική, όπως έχει αναφερθεί και στο κεφάλαιο 4 ο Ιάννης Ξενάκης ήταν αυτός που χρησιμοποίησε τις μαρκοβιανές αλυσίδες και δημιούργησε μέσα από αυτές την στοχαστική μουσική.

Στην παραγωγή μουσικής με την βοήθεια των μαρκοβιανών αλυσίδων υπάρχουν δύο τάξεις αλυσίδων οι αλυσίδες πρώτης τάξης και οι αλυσίδες δεύτερης τάξης.

Στις αλυσίδες πρώτης τάξης τα γεγονότα μετατρέπονται σε μουσικές νότες και υπάρχει ένας πίνακας πιθανοτήτων ο οποίος είναι γνωστός ως πίνακας μετάβασης καθώς δείχνει τις πιθανότητες που έχουν οι νότες να αναπαραχθούν μετά από μια συγκεκριμένη νότα. Οι αλυσίδες πρώτης τάξης αναπαριστώνται στην εικόνα 101.

1st-order matrix			
Note	A	C#	E $\flat$
A	0.1	0.6	0.3
C#	0.25	0.05	0.7
E $\flat$	0.7	0.3	0

Εικόνα 101: Μαρκοβιανές αλυσίδες πρώτης τάξης

Οι αλυσίδες δεύτερης τάξης σε σχέση με τις αλυσίδες της πρώτης τάξης παράγουν ένα καλύτερο μουσικό αποτέλεσμα. Καθώς στις αλυσίδες δεύτερης τάξης λαμβάνεται υπόψη ως είσοδος και η τρέχουσα νότα αλλά και η νότα που έχει παραχθεί πριν την τρέχουσα νότα. Οι αλυσίδες δεύτερη τάξης αναπαριστώνται στην εικόνα 102

2nd-order matrix			
Notes	A	D	G
AA	0.18	0.6	0.22
AD	0.5	0.5	0
AG	0.15	0.75	0.1
DD	0	0	1
DA	0.25	0	0.75
DG	0.9	0.1	0
GG	0.4	0.4	0.2
GA	0.5	0.25	0.25
GD	1	0	0

Εικόνα 102: Μαρκοβιανές αλυσίδες δεύτερης τάξης

Οι μαρκοβιανές αλυσίδες μπορούν να αναπαραστήσουν την αρχή σε ένα project για την παραγωγή μουσικής μέσα από κβαντικούς αλγόριθμους καθώς δείχνουν την σειρά με την οποία θα εκτελεστούν οι μουσικές νότες. Μια υλοποίηση για την δημιουργία μιας μαρκοβιανής αλυσίδας ακολουθεί στον κώδικα της εικόνας 105

Η μαρκοβιανή αλυσίδα όπως και ο πίνακας μετάβασης από την μία νότα στην άλλη δημιουργείτε με βάση τους παρακάτω μουσικούς κανόνες. Οι κανόνες είναι οι εξής και το 'V' συμβολίζει το 'ή', πχ μετά την νότα E θα αναπαραχθεί η νότα ή η νότα:

- Κανόνας 1: εάν $E \Rightarrow F$ εάν Ds
- Κανόνας 2: εάν $Ds \Rightarrow E \vee Cs \vee Fs \vee Gs$
- Κανόνας 3: εάν $Cs \Rightarrow G \vee Fs \vee Ds$
- Κανόνας 4: εάν $G \Rightarrow F \vee Cs \vee D$

- Κανόνας 5: εάν $D \Rightarrow F \vee G \vee G_s \vee B$
- Κανόνας 6: εάν $F \Rightarrow E \vee G \vee D \vee C$
- Κανόνας 7: εάν $C \Rightarrow F \vee F_s, B \vee A$
- Κανόνας 8: εάν $F_s \Rightarrow C_s \vee D_s \vee C \vee A$
- Κανόνας 9: εάν $A \Rightarrow F_s \vee G_s \vee C \vee A_s$
- Κανόνας 10: εάν $G_s \Rightarrow D_s \vee D \vee B \vee A$
- Κανόνας 11: εάν $B \Rightarrow G_s \vee D \vee C \vee A_s$
- Κανόνας 12: εάν $A_s \Rightarrow B \vee A$

Με βάση τους παραπάνω κανόνες δημιουργείτε ένα λεξικό που έχει την λειτουργία πίνακα μετάβασης ο κώδικας υλοποιείται ως εξής και φαίνεται στην εικόνα 103 :

```

1 import numpy as np
2 import random
3 #Markov Chains
4 V = []
5 transition_matrix = {
6     'E': {'E':0, 'F':1/2, 'G':0, 'Cs':0, 'Fs':0, 'Ds':1/2, 'Gs':0, 'D':0, 'B':0, 'C':0, 'A':0, 'As':0},
7     'Ds': {'E':1/4, 'F':0, 'G':0, 'Cs':1/4, 'Fs':1/4, 'Ds':0, 'Gs':1/4, 'D':0, 'B':0, 'C':0, 'A':0, 'As':0},
8     'Cs': {'E':0, 'F':0, 'G':1/3, 'Cs':0, 'Fs':1/3, 'Ds':1/3, 'Gs':0, 'D':0, 'B':0, 'C':0, 'A':0, 'As':0},
9     'G': {'E':0, 'F':1/3, 'G':0, 'Cs':1/3, 'Fs':0, 'Ds':0, 'Gs':0, 'D':1/3, 'B':0, 'C':0, 'A':0, 'As':0},
10    'D': {'E':0, 'F':1/4, 'G':1/4, 'Cs':0, 'Fs':0, 'Ds':0, 'Gs':1/4, 'D':0, 'B':1/4, 'C':0, 'A':0, 'As':0},
11    'F': {'E':1/4, 'F':0, 'G':1/4, 'Cs':0, 'Fs':0, 'Ds':0, 'Gs':0, 'D':1/4, 'B':0, 'C':1/4, 'A':0, 'As':0},
12    'C': {'E':0, 'F':1/4, 'G':0, 'Cs':0, 'Fs':1/4, 'Ds':0, 'Gs':0, 'D':0, 'B':1/4, 'C':0, 'A':1/4, 'As':0},
13    'Fs': {'E':0, 'F':0, 'G':0, 'Cs':1/4, 'Fs':0, 'Ds':1/4, 'Gs':0, 'D':0, 'B':0, 'C':1/4, 'A':1/4, 'As':0},
14    'A': {'E':0, 'F':0, 'G':0, 'Cs':0, 'Fs':1/4, 'Ds':0, 'Gs':1/4, 'D':0, 'B':0, 'C':1/4, 'A':0, 'As':1/4},
15    'Gs': {'E':0, 'F':0, 'G':0, 'Cs':0, 'Fs':0, 'Ds':1/4, 'Gs':0, 'D':1/4, 'B':1/4, 'C':0, 'A':1/4, 'As':0},
16    'B': {'E':0, 'F':0, 'G':0, 'Cs':0, 'Fs':0, 'Ds':0, 'Gs':1/4, 'D':1/4, 'B':0, 'C':1/4, 'A':0, 'As':1/4},
17    'As': {'E':0, 'F':0, 'G':0, 'Cs':0, 'Fs':0, 'Ds':0, 'Gs':0, 'D':0, 'B':1/2, 'C':0, 'A':1/2, 'As':0},
18 }
19 values_1 = list(transition_matrix.keys())
20
21 for i in values_1:
22     values_2 = list(transition_matrix[i].keys())
23
24 print("The Transition Matrix Is:\n ")
25
26 print(f"{values_1}")
27
28 a = []
29 for i, h in enumerate(transition_matrix):
30     a.append(list(transition_matrix[h].values()))
31
32
33 print(f'{values_2[i]}', a[i])

```

Εικόνα 103: Δημιουργία λεξικού που έχει ρόλο πίνακα μετάβασης

Γραμμή 1-17: Δημιουργία ενός διπλού λεξικού το οποίο αναπαριστά τους κανόνες που αναφέρθηκαν νωρίτερα. Όπου κάθε κλειδί του εξωτερικού λεξικού αναπαριστά το πρώτο στοιχείο κάθε μουσικού κανόνα και περιέχει ένα εσωτερικό λεξικό.

Τα εσωτερικά λεξικά δείχνουν ποιες είναι οι πιθανότητες εμφάνισης των άλλων μουσικών νοτών για την κάθε νότα/κλειδί του εξωτερικού λεξικού. είναι ένα στοιχείο ένας από τους μουσικούς κανόνες που αναφέρθηκαν νωρίτερα.

Γραμμή 19: Αποθήκευση σε μία λίστα όλα τα κλειδιά του εξωτερικού λεξικού (δηλαδή ('E','Ds','Cs','G','D','F','C','Fs','A','Gs','B','As'))

Γραμμή 21-22: Δημιουργία μιας λίστας που περιέχει τα κλειδιά των εσωτερικών λεξικών δηλαδή ('E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As')

Γραμμή 26: Εμφάνιση των κλειδιών του εξωτερικού λεξικού

Γραμμή 25-29: Αναπαράσταση του λεξικού με μορφή πίνακα για την καλύτερη κατανόηση

Το αποτέλεσμα που προκύπτει από την εκτέλεση της γραμμή 29 αναπαρίσταται στην εικόνα 104 και είναι ο πίνακας μετάβασης που δείχνει τις πιθανότητες εκτέλεσης της κάθε νότας για κάθε ένα από τα στοιχεία.

Είναι αντιληπτό πως η σειρά εμφάνισης των στοιχείων του διπλού λεξικού δημιουργεί μια μαρκοβιανή αλυσίδα.

The Transition Matrix Is:

```
['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
E [0, 0.5, 0, 0, 0, 0.5, 0, 0, 0, 0, 0, 0]
Ds [0.25, 0, 0, 0.25, 0.25, 0, 0.25, 0, 0, 0, 0, 0]
Cs [0, 0, 0.3333333333333333, 0, 0.3333333333333333, 0.3333333333333333, 0, 0, 0, 0, 0, 0]
G [0, 0.3333333333333333, 0, 0.3333333333333333, 0, 0, 0.3333333333333333, 0, 0, 0, 0, 0]
D [0, 0.25, 0.25, 0, 0, 0.25, 0, 0.25, 0, 0, 0, 0]
F [0.25, 0, 0.25, 0, 0, 0, 0.25, 0, 0.25, 0, 0, 0]
C [0, 0.25, 0, 0, 0.25, 0, 0, 0.25, 0, 0.25, 0, 0]
Fs [0, 0, 0, 0.25, 0, 0.25, 0, 0, 0.25, 0.25, 0, 0]
A [0, 0, 0, 0, 0.25, 0, 0.25, 0, 0, 0.25, 0, 0.25]
Gs [0, 0, 0, 0, 0.25, 0, 0.25, 0.25, 0, 0.25, 0, 0]
B [0, 0, 0, 0, 0, 0.25, 0.25, 0, 0.25, 0, 0.25, 0]
As [0, 0, 0, 0, 0, 0, 0, 0.5, 0, 0.5, 0, 0]
```

Εικόνα 104: Αρχικός πίνακας μετάβασης

Στην συνέχεια ακολουθεί μια συνάρτηση που παράγει κάθε φορά μια διαφορετική μαρκοβιανή αλυσίδα με σκοπό η μελωδία που θα προκύπτει να μην είναι πάντα ίδια και αναπαρίσταται στην εικόνα 105:

```

1 def Markov_Chain(trasition_matrix):
2     sequence = []
3     firsts_notes = list(trasition_matrix.keys())
4     initial_state = 'E'
5     current_note = initial_state
6     sequence.append(current_note)
7
8     while len(sequence) - 1 != len(firsts_notes) - 1:
9
10
11         target_notes = list(trasition_matrix[current_note].keys())
12         propab = list(trasition_matrix[current_note].values())
13         print(target_notes)
14         print(propab)
15
16         next_note = np.random.choice(target_notes, p=propab)
17
18         print("Current_Note --> Next_Note")
19
20
21         print(f'{current_note}-->{next_note}\n')
22         current_note = next_note
23
24         sequence.append(current_note)
25
26
27     return (sequence)
28

```

Εικόνα 105: Δημιουργία Μαρκοβιανής αλυσίδας

Γραμμή 1: Δημιουργία συνάρτησης που αναπαριστά την μαρκοβιανή αλυσίδα

Γραμμή 2-3: Δημιουργία πινάκων που θα χρησιμοποιηθούν για αποθήκευση των στοιχείων της αλυσίδας

Γραμμή 4: Δημιουργία λίστας που περιέχει τίς νότες που αναπαριστούν τους κανόνες δηλαδή τα κλειδιά του εξωτερικού λεξικού

Γραμμή 5: Αρχικοποίηση της αλυσίδας με μια νότα

Γραμμή 6: Η αρχική νότα γίνεται τωρινή νότα και ο αλγόριθμός την πρώτη φορά εκτελείτε με βάση αυτή.

Γραμμή 8: Όσο το μήκος της λίστας ή αλλιώς της αλυσίδας περιέχει δώδεκα νότες όσες δηλαδή και οι αρχικοί κανόνες τότε επανάλαβε την διαδικασία

Γραμμή 11: Δημιουργία μιας λίστας η οποία ανάλογα την τιμή της τωρινής νότας δηλαδή ανάλογα με το κλειδί του εξωτερικού λεξικού αποθηκεύει στην λίστα τα κλειδιά (δηλαδή τις αντίστοιχες νότες) του συγκεκριμένου εσωτερικού λεξικού.. Το αποτέλεσμα αυτής της εντολής με τωρινή νότα την 'E' αναπαρίσταται στην εικόνα 106:

`['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']`

Εικόνα 106: Δημιουργία λίστας που περιέχει τα κλειδιά των εσωτερικών λεξικών

Γραμμή 12: Δημιουργία μια λίστας η οποία με βάση το (τωρινή νότα) κλειδί του εξωτερικού λεξικού περιέχει τις τιμές του κλειδιού δηλαδή τις αντίστοιχες τιμές του εσωτερικού λεξικού. Το αποτέλεσμα αυτής της εντολής με τωρινή νότα την 'Ε' αναπαρίσταται στην εικόνα 107 και δείχνει τις πιθανότητες εμφάνισης της κάθε νότας της γραμμής 11

`[0, 0.5, 0, 0, 0, 0.5, 0, 0, 0, 0, 0]`

Εικόνα 107: Δημιουργία λίστας που περιέχει τις τιμές για το λεξικό 'Ε'

Γραμμή 16: Η αναπαραγωγή της επόμενης νότας στηρίζεται στην προηγούμενη νότα δηλαδή στην τωρινή νότα και είναι μια τυχαία επιλογή από τις νότες που θα προκύψουν από την γραμμή 12 με τις πιθανότητες τους οι οποίες βρίσκονται στην γραμμή 13.

Γραμμή 21: Εμφάνιση Τωρινής και επόμενης νότας

Γραμμή 22: Η επόμενη νότα γίνεται τωρινή νότα η αλλιώς η προηγούμενη νότα και επαναλαμβάνεται η διαδικασία για την εύρεση της επόμενης νότας

Γραμμή 24: Εισαγωγή στην αλυσίδα όλα τα στοιχεία

Το συνολικό αποτέλεσμα του παραπάνω κώδικα πραγματοποιείται με την εκτέλεση των εντολών στην εικόνα 108 και αναπαρίσταται στις εικόνες 109, 110, 111 αναπαριστώνται κάθε φορά η λίστα με τα κλειδιά και οι πιθανότητες εμφάνισης του κάθε στοιχείου για κάθε επόμενη νότα:

```
print("For The Following Markov Chain Of The Transition Matrix : \n")
Markov_Chain_1 , B= Markov_Chain(transition_matrix)
print(Markov_Chain_1)
```

Εικόνα 108: Εκτέλεση συνάρτησης και εμφάνιση μαρκοβιανής αλυσίδας

```

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0, 0.5, 0, 0, 0, 0.5, 0, 0, 0, 0, 0, 0]
Current_Note --> Next_Note
E-->F

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0.25, 0, 0.25, 0, 0, 0, 0, 0.25, 0, 0.25, 0, 0]
Current_Note --> Next_Note
F-->G

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0, 0.3333333333333333, 0, 0.3333333333333333, 0, 0, 0, 0.3333333333333333, 0, 0, 0, 0]
Current_Note --> Next_Note
G-->F

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0.25, 0, 0.25, 0, 0, 0, 0, 0.25, 0, 0.25, 0, 0]
Current_Note --> Next_Note
F-->D

```

Εικόνα 109: Αναπαράσταση λίστας που περιέχει τα κλειδιά των εσωτερικών λεξικών και τις πιθανότητες εμφάνισης της κάθε νότας με κύριά νότα την επόμενη νότα (next_note)

```

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0, 0.3333333333333333, 0, 0.3333333333333333, 0, 0, 0, 0.3333333333333333, 0, 0, 0, 0]
Current_Note --> Next_Note
G-->F

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0.25, 0, 0.25, 0, 0, 0, 0, 0.25, 0, 0.25, 0, 0]
Current_Note --> Next_Note
F-->D

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0, 0.25, 0.25, 0, 0, 0, 0.25, 0, 0.25, 0, 0, 0]
Current_Note --> Next_Note
D-->Gs

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0, 0, 0, 0, 0, 0.25, 0, 0.25, 0.25, 0, 0.25, 0]
Current_Note --> Next_Note
Gs-->A

```

Εικόνα 110: Αναπαράσταση λίστας που περιέχει τα κλειδιά των εσωτερικών λεξικών και τις πιθανότητες εμφάνισης της κάθε νότας με κύριά νότα την επόμενη νότα (next_note)

```

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0, 0, 0, 0, 0.25, 0, 0.25, 0, 0, 0.25, 0, 0.25]
Current_Note --> Next_Note
A-->C

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0, 0.25, 0, 0, 0.25, 0, 0, 0, 0.25, 0, 0.25, 0]
Current_Note --> Next_Note
C-->F

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0.25, 0, 0.25, 0, 0, 0, 0, 0.25, 0, 0.25, 0, 0]
Current_Note --> Next_Note
F-->D

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
[0, 0.25, 0.25, 0, 0, 0, 0.25, 0, 0.25, 0, 0, 0]
Current_Note --> Next_Note
D-->G

```

Εικόνα 111: Αναπαράσταση λίστας που περιέχει τα κλειδιά των εσωτερικών λεξικών και τις πιθανότητες εμφάνισης της κάθε νότας με κύρια νότα την επόμενη νότα (next_note)

Ενώ η Μαρκοβιανή αλυσίδα είναι στην εικόνα 112:

```
['E', 'F', 'C', 'F', 'D', 'Gs', 'Ds', 'Gs', 'Ds', 'Cs', 'G', 'F']
```

Εικόνα 112: Αναπαράσταση μαρκοβιανής αλυσίδας

Υστέρα ο πίνακας μετάβαση για την παραπάνω μαρκοβιανής αλυσίδα δημιουργείτε στην εικόνα 113:

```

1 values_1 = list(trasition_matrix.keys())
2
3
4 for i in values_1:
5     values_2 = list( trasition_matrix[i].keys())
6
7 print("The Trasition Matrix Is:\n ")
8
9 print(f"{values_2}")
10
11 a = []
12 for i , values_2 in enumerate (Markov_Chain_1):
13     a.append(list(trasition_matrix[values_2].values()))
14
15
16 print(Markov_Chain_1[i],a[i])

```

Εικόνα 113: Δημιουργία πίνακα μετάβασης

Γραμμή 1: Αποθήκευση σε μία λίστα όλα τα κλειδιά του εξωτερικού λεξικού (δηλαδή ('E','F','C','F','D','Gs','Ds','Gs','Ds','Cs','G','F'))

Γραμμή 3-4: Δημιουργία μιας λίστας που περιέχει τα κλειδιά των εσωτερικών λεξικών δηλαδή ('E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As')

Γραμμή 8: Εμφάνιση της λίστας

Γραμμή 11-12: Για τα κλειδιά του εσωτερικού λεξικού, αποθήκευση κάθε φορά σε μια λίστα τις τιμές της

Γραμμή15: Εμφάνιση του κάθε στοιχείου της αλυσίδας και εμφάνισης της κάθε λίστας που περιέχει τις τιμές

Το αποτέλεσμα είναι ο πίνακας μετάβασης που δείχνει τις πιθανότητες εμφάνισης κάθε στοιχείου της αλυσίδας με βάση τους κανόνες και αναπαρίσταται στην εικόνα 114

```
The Transition Matrix Is:

['E', 'F', 'G', 'Cs', 'Fs', 'Ds', 'Gs', 'D', 'B', 'C', 'A', 'As']
E [0, 0.5, 0, 0, 0, 0.5, 0, 0, 0, 0, 0, 0]
F [0.25, 0, 0.25, 0, 0, 0, 0.25, 0, 0.25, 0, 0, 0]
C [0, 0.25, 0, 0, 0.25, 0, 0, 0, 0.25, 0, 0.25, 0]
F [0.25, 0, 0.25, 0, 0, 0, 0.25, 0, 0.25, 0, 0, 0]
D [0, 0.25, 0.25, 0, 0, 0, 0.25, 0, 0.25, 0, 0, 0]
Gs [0, 0, 0, 0, 0, 0.25, 0, 0.25, 0.25, 0, 0.25, 0]
Ds [0.25, 0, 0, 0.25, 0.25, 0, 0.25, 0, 0, 0, 0, 0]
Gs [0, 0, 0, 0, 0, 0.25, 0, 0.25, 0.25, 0, 0.25, 0]
Ds [0.25, 0, 0, 0.25, 0.25, 0, 0.25, 0, 0, 0, 0, 0]
Cs [0, 0, 0.3333333333333333, 0, 0.3333333333333333, 0.3333333333333333, 0, 0, 0, 0, 0, 0]
G [0, 0.3333333333333333, 0, 0.3333333333333333, 0, 0, 0.3333333333333333, 0, 0, 0, 0, 0]
F [0.25, 0, 0.25, 0, 0, 0, 0.25, 0, 0.25, 0, 0, 0]
```

Εικόνα 114: Αναπαράσταση πίνακα μετάβασης

5.3.3 Κλασικός και κβαντικός περίπατος

Οποιαδήποτε μαρκοβιανή αλυσίδα διάστασης όπως αυτή που αναπαρίσταται στην εικόνα 112 μπορεί να θεωρηθεί ως παράδειγμα τυχαίου περιπάτου μιας διάστασης.

Τι είναι ο τυχαίος περίπατος (Random Walk)

Ο τυχαίος περίπατος αλλιώς ο περίπατος του μεθυσμένου είναι μια διαδικασία που περιγράφει μια διαδρομή στην οποία η κατεύθυνση και τα βήματα γίνονται με βάση τις πιθανότητες. Έστω πως υπάρχει ένας μεθυσμένος σε έναν έρημο δρόμο από την στιγμή που είναι μεθυσμένος θα κινηθεί τυχαία δεξιά ή αριστερά. Βρίσκετε στη αρχή του δρόμου και η κίνηση του συμβολίζεται με μηδέν καθώς δεν έχει αρχίσει να μετακινείται. Η κίνηση δεξιά συμβολίζεται με +1 ενώ η κίνηση αριστερά συμβολίζεται με -1. Οι πιθανότητες να βρεθεί σε οποία από τις δύο κατευθύνσεις είναι ίδια.

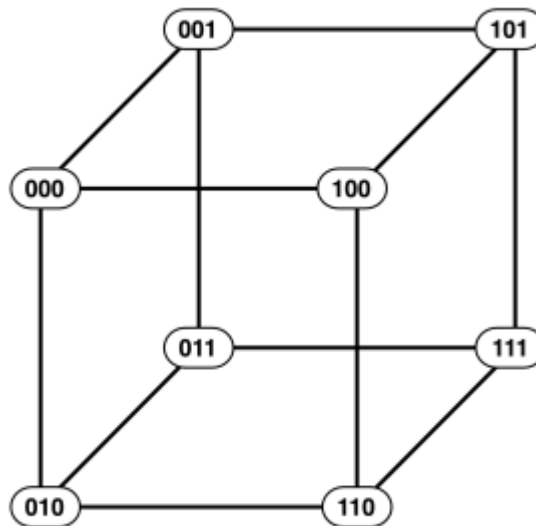
Η κατάσταση που θα καταλήξει είναι τυχαία και δεν εξαρτάται από την αρχική κατάσταση. Μια μαρκοβιανή αλυσίδα θεωρείτε ως τυχαίος περίπατος καθώς το κάθε στοιχείο προκύπτει από τις πιθανότητες που στηρίζονται σε αυτό και δεν εξαρτάται από τα άλλα στοιχεία. Στην Μαρκοβιανή αλυσίδα στην εικόνα , δηλαδή στην $M = ['E', 'F', 'C', 'F', 'D', 'Gs', 'Ds', 'Gs', 'Ds', 'Cs', 'G', 'F']$ είναι αντιληπτό πως για οποιοδήποτε στοιχείο π για το στοιχείο 'F' αν η πιθανότητα να βρεθεί στην θέση 'C' είναι $q = 0.5$ τότε η πιθανότητα να βρεθεί στην θέση 'E' είναι $p = 1 - q = 0.5$ οπότε η πιθανότητα του F να βρεθεί σε οποιαδήποτε από τις 2 θέσεις είναι ίδια και όπως είναι αντιληπτό η θέση στην οποία θα βρεθεί δεν επηρεάζεται από τις κινήσεις των άλλων στοιχείων.

5.3.3.1 Κβαντικός περίπατος(Quantum Walk)

Εκτός από τον κλασικό περίπατο υπάρχει και ο κβαντικός περίπατος(Quantum Walk) όπου η διαφορά ανάμεσα σε αυτούς τους δύο είναι πως ο κβαντικός περπατητής μπορεί ταυτόχρονα να βρίσκεται σε όλες τις πιθανές θέσεις και αυτό οφείλετε στην ιδιότητα της υπέρθεσης.

Ένας περπατητής μπορεί να βρίσκετε και σε διαστάσεις μεγαλύτερες από μία. Αυτό έχει σαν αποτέλεσμα οι πιθανότητες να καταλήξει στην κατάσταση που ξεκίνησε να μειώνονται όσο αυξάνονται οι διαστάσεις. Ύστερα θα αναλυθεί η περίπτωση που ο περπατητής μπορεί να βρίσκεται σε μιγαδικό χώρο 3 διαστάσεων άρα σε πραγματικό χώρο 8 διαστάσεων. Όπου η κάθε κατάσταση που θα βρίσκετε ο κβαντικός περπατητής θα αναπαριστάτε με την εξής μορφή $|xyz\rangle = (|x\rangle \otimes |y\rangle \otimes |z\rangle)$.

Έστω λοιπόν ο παρακάτω κύβος όπου οι κορυφές του είναι οι θέσεις στις οποίες βρίσκετε ο περπατητής και οι ακμές δείχνουν τις κατευθύνσεις τις οποίες μπορεί να ακολουθήσει για να βρεθεί από την μια κορυφή στην άλλη. Ο κύβος αναπαρίσταται στην εικόνα 115



Εικόνα 115: Αναπαράσταση κύβου, ενός κβαντικού περπατητή

Στην συνέχεια ο παραπάνω κύβος υλοποιείται ως ένα κβαντικό κύκλωμα το οποίο απο-τελείται από 5 qubits τα 3 πρώτα qubits δημιουργούν 8 κβαντικούς

καταχωρητές όπου αναπαριστούν όλες τις θέσεις στις οποίες μπορεί να βρεθεί ο περπατητής και τα 2 τελευταία είναι βοηθητικά qubits και δεν αναπαριστώνται στον κύβο. Τα βοηθητικά qubits αρχικά με την βοήθεια της πύλης Hadamard τοποθετούνται σε υπέρ-θεση και έχουν τον ρόλο των ζαριών τα οποία δείχνουν όλες τις κατευθύνσεις στις οποίες μπορεί να κινηθεί ο περπατητής. Δηλαδή αν το πρώτο ζάρι οδηγήσει σε κατάσταση 0 το αντίστοιχο qubit, τότε υπάρχει μετακίνηση προς τα δεξιά αλλιώς μετακίνηση προς τα αριστερά, το πρώτο ζάρι επιδρά στα qubits μέσω της πύλης cx . Ύστερα για το δεύτερο ζάρι αν οδηγήσει σε κατάσταση 0 υπάρχει κίνηση προς τα πάνω αλλιώς προς τα κάτω, το δεύτερο ζάρι επιδρά στα qubits με μια πύλη ccx . Το δεύτερο ζάρι εξαρτάται από το πρώτο καθώς μετακινεί το περπατητή αφού έχει μετακινηθεί δεξιά ή αριστερά

Ο κώδικας για την παραγωγή του κυκλώματος του κβαντικού περπατητή καθώς και η πρώτη φορά που θα περπατήσει στον κύβο εμφανίζεται στην εικόνα 116:

```

1 from qiskit import *
2 from qiskit.visualization import circuit_drawer
3 from qiskit.tools.visualization import plot_histogram
4 from qiskit.tools.monitor import job_monitor
5 from qiskit.visualization import plot_bloch_multivector, plot_histogram
6 import random
7 n = 3
8 n1 = 2
9
10 qr = QuantumRegister(n,"input_qubit")
11 qr1 =QuantumRegister(n1,"dice_operators")
12
13 cr = ClassicalRegister(n,"classic_operator")
14
15 circuit = QuantumCircuit(qr,qr1,cr)
16 #First State Of The Walker
17 def walker(qc):
18     for i in range(3):
19         qc.i(i)
20         qc.h(qr1[0:2])
21         qc.barrier()
22
23         qc.cx(qr1[1],qr[0])
24         for i in range(3):
25
26             if i % 2 == 0:
27                 qc.x(qr1[0])
28             elif i % 2 == 1:
29                 qc.x(qr1[1])
30
31             qc.cx(qr1[1],qr[i%3])
32             qc.ccx(qr1[0],qr1[1],qr[i%3])
33
34
35         qc.barrier()
36         qc.h(qr1[0:2])
37         qc.measure(qr,cr)
38     return qc
39

```

Εικόνα 116: Υλοποίηση κβαντικού περιπάτου

Γραμμή 1-6: Εισαγωγή των απαραίτητων βιβλιοθηκών

Γραμμή 7-8: Εισαγωγή τιμών των qubits και των dice qubits

Γραμμή 10 -13: Δημιουργία κβαντικού , κλασικού καταχωρητή και κυκλώματος

Γραμμή 17: Εκτέλεση πρώτη φορά τον αλγόριθμο του περίπατου

Γραμμή 20: Εφαρμογή Handamard πύλης στα 3 πρώτα qubits καθώς ο περπατητής όπως αναφέρθηκε και πριν βρίσκετε σε όλα τα σημεία του χώρου

Γραμμή 23:Εφαρμογή cx πύλης με σκοπό να γίνει η αλλαγή της κατεύθυνσης του περπατητή προς τα δεξιά η προς τα αριστερά με την βοήθεια του πρώτου ζαριού, το οποίο είναι το πρώτο βοηθητικό qubit.

Γραμμή 24:Επανάληψη τρεις φορές της ακόλουθης διαδικασίας οπου αυτές οι τρεις φορές καθορίζουν την πορεία της διαδρομής του περπατητή, η μια έγινε με την cx στην γραμμή 23

Γραμμή 26 – 29: Ανά δύο βήματα του περπατητή εφαρμογή μίας πύλης x με σκοπό οι τιμές των βοηθητικών ζαριών να αλλάζουν κάθε φορά. Συνολικά οι φορές είναι τρεις καθώς τρεις είναι και οι πιθανές διαδρομές που μπορεί να ακολουθήσει, όπως φαίνεται και στον κύβο κάθε κατάσταση συνδέεται με τρεις άλλες καταστάσεις και οι διαδρομές που μπορεί να ακολουθήσει ο περπατητής είναι τρεις

Γραμμή 31: Εφαρμογή cx πύλης η οποία θα καθορίσει την κίνηση του περπατητή αν είναι δεξιά ή αριστερά

Γραμμή 32: Εφαρμογή πύλης cxx η οποία ξεκινάει από το πρώτο βοηθητικό qubit και όχι από το δεύτερο καθώς με βάση την κατεύθυνση που έχει δώσει το πρώτο ζάρι (δεξιά/αριστερά) μετακινείται αντίστοιχα ο περπατητής πάνω ή κάτω

Γραμμή 36: Αναίρεση πύλης Hadamard

Γραμμή 37: Μέτρηση του κυκλώματος

Γραμμή 38: Επιστροφή του κυκλώματος

Στο κύκλωμα υπάρχει και η πύλη X (η οποία σε ένα κβαντικό κύκλωμα αντιστρέφει την κατάσταση στην οποία βρίσκεται το qubit) και στο παράδειγμα εφαρμόζεται ώστε να αντιστρέψει την κατάσταση του ζαριού. Όπως αναφέρθηκε προηγουμένως το κάθε ζάρι έχει σαν ρόλο να καθοδηγεί τον περπατητή,

το πρώτο ζάρι μέσω της πύλης $c\text{-not}$ δίνει την αρχική κατεύθυνση στον περπατητή η οποία είναι δεξιά και αριστερά. Ενώ το δεύτερο ζάρι επεκτείνει μέσω της πύλης $c\text{cnot}$ την κατεύθυνση του περπατητή σε πάνω και κάτω.

Για την αναπαράσταση του κυκλώματος γίνεται η εκτέλεση των έξης εντολών που βρίσκονται στην εικόνα 116.1:

```

1 gate_colours = {
2     "displaycolor": {
3         "x": [ # gate name
4             "#3E065F", # box color (red)
5             "#FFFFFF" # box text color (white)
6         ],
7         "h": [ # gate name
8             "#8B0000", # box color (maroon)
9             "#FFFFFF" # box text color (white)
10        ],
11        "cx": [ # gate name
12            "#700B97", # box color (red)
13            "#FFFFFF" # box text color (white)
14        ],
15        ],
16        "ccx": [ # gate name
17            "#700B97", # box color (red)
18            "#FFFFFF" # box text color (white)
19        ],
20        ],
21        "i": [ # gate name
22            "#700B97", # box color (red)
23            "#FFFFFF" # box text color (white)
24        ],
25        ],
26        "Oracle": [ # gate name
27            "#951556", # box color (red)
28            "#FFFFFF" # box text color (white)
29        ],
30        ],
31        "Diffuser": [ # gate name
32            "#411E8F", # box color (red)
33            "#FFFFFF" # box text color (white)
34        ],
35    },
36 }
37 Walker = walker(circuit)
38 Walker.draw("mpl", style=gate_colours)

```

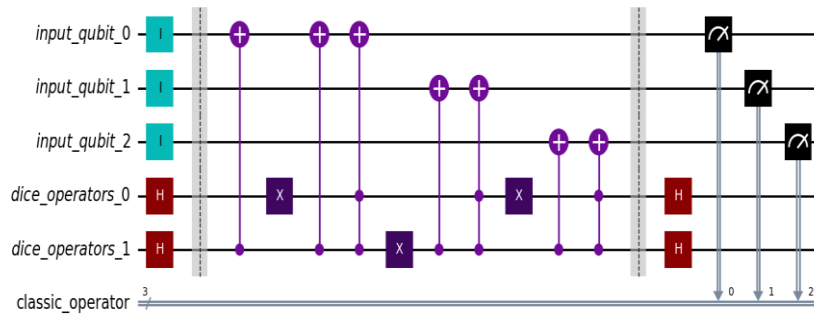
Εικόνα 116.1: Αναπαράσταση κυκλώματος, ενός κβαντικού περιπατητή και ορισμός χρωμάτων στις πύλες

Γραμμή 1-33: Λεξικό το οποίο καθορίζει το χρώμα της κάθε πύλης

Γραμμή 34: Κλήση της συνάρτησης για εκτέλεση του αλγόριθμου του κβαντικού περιπατητή

Γραμμή 35: Αναπαράσταση κυκλώματος

Το κύκλωμα για την πρώτη εκτέλεση του περιπατητή αναπαρίσταται στην εικόνα 117



Εικόνα 117: Κύκλωμα κβαντικού περιπάτου

Δεύτερη εκτέλεση του αλγορίθμου του κβαντικού περιπάτου

Αφού γίνει η εκτέλεση του αλγορίθμου στην συνέχεια θα πραγματοποιηθεί μέτρηση του κυκλώματος ώστε να εμφανιστούν οι τέσσερις πιθανές καταστάσεις στις οποίες βρίσκεται ο περιπατητής και η κατάσταση με την μεγαλύτερη πιθανότητα εμφάνισης είναι αυτή στην οποία κατέληξε τελικά οπότε είναι η επόμενη κατάσταση στην οποία θα βρεθεί αν θεωρηθεί πως η αρχική του κατάσταση είναι η 000.

Η επιστροφή των καταστάσεων θα γίνει με την χρήση των εντολών στις εικόνες 118, 119:

```

1 def Answer(qc):
2     # walker = walker()
3     job = execute(qc, Aer.get_backend('qasm_simulator'), shots = 1024).result()
4     answer = job.get_counts()
5     return(answer)
6

```

Εικόνα 118: Επιστροφή 4 πιθανών καταστάσεων

Γραμμή1: Δημιουργία συνάρτησης οποία παίρνει το κύκλωμα του κβαντικού περιπατητή

Γραμμή 3:Εκτέλεση του κυκλώματος σε κβαντικό προσωμοιωτή

Γραμμή4: Αποθήκευση δεδομένων του κυκλώματος

Γραμμή 5:Επιστροφή δεδομένων του κυκλώματος

```

1 answer = Answer(walker)
2 print(answer)
3 plot_histogram(answer, color='maroon', title='Quantum Walk Algorithm')
4

```

Εικόνα 119: Εμφάνιση 4 πιθανών καταστάσεων και εμφάνιση ιστογράμματος

Γραμμή 1: Εφαρμογή συνάρτησης για ανάκτηση των δεδομένων του κυκλώματος του κβαντικού περιπατητή

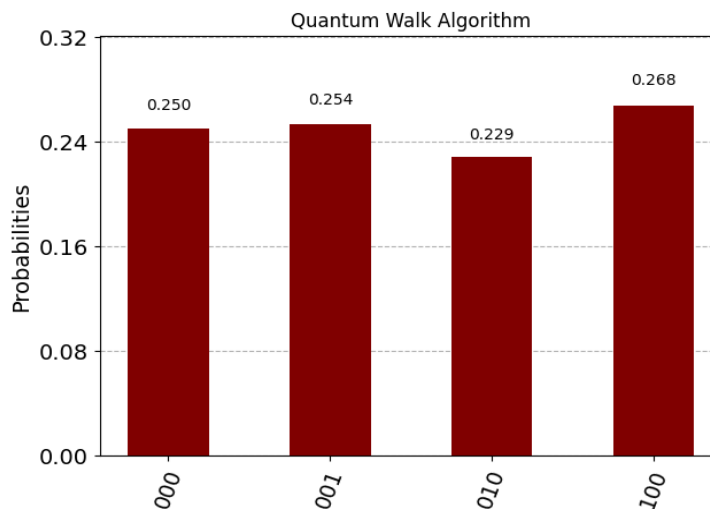
Γραμμή2: Εμφάνιση δεδομένων

Γραμμή 3:Αναπαράσταση ιστογράμματος

Τα δεδομένα και το ιστόγραμμα από την εκτέλεση των γραμμών 2, φαίνονται στις εικόνες 120, 121:

{'010': 234, '000': 256, '001': 260, '100': 274}

Εικόνα 120: Δεδομένα κυκλώματος



Εικόνα 121: Αναπαράσταση ιστογράμματος

Στην εικόνα 121 τα δεδομένα αναπαριστώνται με φορά αντίθετη από ότι είναι στην πραγματικότητα και από ότι είναι στο λεξικό, αυτό συμβαίνει καθώς στο ιστόγραμμα τα qubits εμφανίζονται από πάνω προς τα κάτω δηλαδή ανάποδα.

Για την εύρεση της καλύτερης κατάστασης και αυτής στην οποία έχει περισσότερες πιθανότητες να καταλήξει ο περπατητής χρησιμοποιείται ο κώδικας της εικόνας 121.1:

```

1 def Next_state(answer):
2
3     dict_keys = list(answer.keys())
4     dict_vals = list(answer.values())
5
6     max_state = dict_keys[dict_vals.index(max(dict_vals))]
7
8
9     return (max_state)

```

Εικόνα 121.1: Εύρεση καλύτερης κατάστασης η οποία έχει μεγαλύτερη τιμή από τις άλλες καταστάσεις

Γραμμή 1: Δημιουργία συνάρτησης η οποία δέχεται ως είσοδο τα δεδομένα της εικόνας που είναι όλες οι καταστάσεις με τις πιθανότητες εμφάνισης, λειτουργία αυτής της συνάρτησης είναι η επιστροφή της κατάστασης με την μεγαλύτερη πιθανότητα εμφάνισης

Γραμμή 3: Δημιουργία λίστας που περιέχει όλες τις καταστάσεις

Γραμμή 4: Δημιουργία λίστας που περιέχει τις πιθανότητες εμφάνισης για τις καταστάσεις

Γραμμή 5: Εύρεση κατάστασης η οποία είναι στην θέση όπου βρίσκεται η μέγιστή τιμή πιθανότητας και θεωρείτε η επόμενη κατάσταση από την αρχική

Γραμμή 9: Επιστροφή μέγιστης κατάστασης

Στην εικόνα 122 γίνεται η κλήση της συνάρτησης για την επόμενη κατάσταση η οποία είναι η θέση από την οποία ξεκινάει ο κβαντικός περίπατος:

```

1 next_state = Next_state(answer)
2 print(f"The real next state is: {next_state}")
3
4 Next_State = next_state[::-1]
5 print(f"The next state for the circuit is: {Next_State}")
6

```

Εικόνα 122: Κλήση της συνάρτησης για την καλύτερη κατάσταση, που είναι η αρχική κατάστας του επόμενου περιπάτου

Γραμμή 1: Κλήση της συνάρτησης

Γραμμή 2: Εμφάνιση της κατάστασης όπως είναι στον πραγματικό κόσμο

Γραμμή 4: Αντιστροφή της κατάστασης για να χρησιμοποιηθεί στον κβαντικό αλγόριθμο

Γραμμή 5: Εμφάνιση της κατάστασης όπως χρειάζεται για τον κβαντικό αλγόριθμο

Το αποτέλεσμα του παραπάνω κώδικά βρίσκετε στην εικόνα 123:

```
The real next state is: 100
The next state for the circuit is: 001
```

Εικόνα 123: Εμφάνιση επόμενης κατάστασης

Πλέον η επόμενη κατάσταση η αλλιώς η κατάσταση με την μεγαλύτερη πιθανότητα εμφάνισης αποτελεί την κατάσταση από την οποία θα ξεκινάει κάθε φορά ο επόμενος κβαντικός περίπατος. Οπότε στο κύκλωμα του κβαντικού περιπάτου πρέπει να συμπεριληφθεί και το γεγονός ότι η κατάσταση από την οποία ξεκινάει ο περίπατος περιέχει qubits τα οποία ως αρχική κατάσταση πρέπει να έχουν την κατάσταση 1. Σε αυτήν την περίπτωση εκτελείται το κύκλωμα της εικόνας 124

```
1 n = 3
2 n1 = 2
3 qr = QuantumRegister(n,"input_qubit")
4 qr1 = QuantumRegister(n1,"dice_operators")
5 cr = ClassicalRegister(n,"classic_operator")
6
7 #Creating New Circuit For The Next State Of The Walker
8 second_circuit = QuantumCircuit(qr,qr1,cr)
9
10 def move(qc1 ,state):
11     for i , j in enumerate (state) :
12         if j == '1':
13             qc1.x(qr[i])
14     qc1.barrier()
15     return qc1
```

Εικόνα 124: Επόμενος κβαντικός περίπατος

Γραμμή1-8: Εισαγωγή τιμών των qubits, dice qubits και δημιουργία κβαντικού , κλασικού καταχωρητή και κυκλώματος

Γραμμή 10: Δημιουργία συνάρτησης η οποία καθορίζει την αρχική κατάσταση των qubits ανάλογα με την συμβολοσειρά της 'επόμενης κατάστασης'

Γραμμή 11-13: Αν η συμβολοσειρά της 'επόμενη κατάσταση' περιέχει 1 τότε εφαρμογή x πύλης ώστε η αρχική κατάσταση του qubit να είναι 1 με την επόμενη κλήση του κυκλώματος του κβαντικού περιπάτου

Υστερά ο κώδικας για τον επόμενο κβαντικό περίπατο είναι στην εικόνα 125:

```
2 caution = move(second_circuit,Next_State)
3 Next_Walk = walker(n)
4
5 Next_Walk.draw("mpl")
```

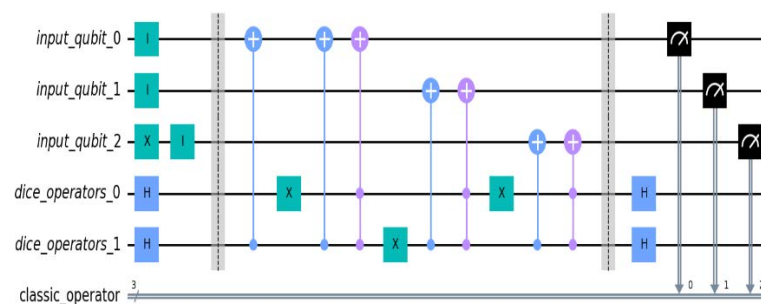
Εικόνα 125: Κώδικας για εμφάνιση του επόμενου κβαντικού περιπάτου

Γραμμή 2: Δημιουργία ενός απλου κυκλώματος όπου οι αρχικές καταστάσεις των qubits θα καθορίζονται από ένα bit string

Γραμμή 3: Συνέχεια εκτέλεσης του κυκλώματος της γραμμής 2 ως κύκλο του κβαντικού περιπάτου

Γραμμή 5:Εμφάνιση του κυκλώματος

Το κύκλωμα του κβαντικού περιπάτου με επόμενη κατάσταση την 001 όταν εκτελεστεί δεύτερη φορά είναι στην εικόνα 126:



Εικόνα 126: Κύκλωμα επόμενου κβαντικού περιπάτου

Κάθε κατάσταση με την μεγαλύτερη πιθανότητα εμφάνισης που προκύπτει από τον αλγόριθμο του κβαντικού περιπάτου καθορίζει την κίνηση του πάνω στην μαρκοβιανή αλυσίδα, το αποτέλεσμα που προκύπτει χρησιμοποιείται για να παράξει μελωδίαΣ. Ο κωδικός αναπαρίσταται στην εικόνα 127

```

: 1 def sound_of_the_next_state(Next_state ,Markov_Chain_1):
2     note = []
3     current_state= 0
4
5     for i in range ((len(Next_state))):
6         if Next_state[i] == '1':
7             current_state -= 1
8         elif Next_state[i] == '0':
9             current_state += 1
10
11         next_note = Markov_Chain_1[current_state]
12
13         note.append(next_note)
14
15
16     return note
17
: 1 sound = sound_of_the_next_state(Next_State ,Markov_Chain_1)
2 print(sound)
3 print(Next_State)
.

```

Εικόνα 127: Παραγωγή μουσικής με βάση την μεγαλύτερη κατάσταση

Γραμμή 1: Δημιουργία συνάρτησης που θα παράξει ήχο μέσα από την μαρκοβιανή αλυσίδα και την συμβολοσειρά της επόμενης κατάστασης

Γραμμή 3: Ορισμός αρχικής νότας/κατάστασης

Γραμμή 5-7: Για κάθε στοιχείο του bit string εάν το στοιχείο είναι 1 τότε η νότα που αναπαραχθεί θα είναι αυτή που βρίσκετε μια θέση πίσω από την τωρινή νότα

Γραμμή 8-9: Εάν το στοιχείο της κατάστασης είναι 0 τότε η νότα που αναπαραχθεί θα είναι αυτή που βρίσκετε μια θέση μπροστά από την τωρινή νότα

Γραμμή 11: Εμφάνιση νότα με βάση την θέση της τωρινής νότας στην μαρκοβιανή αλυσίδα

Γραμμή 12: Αποθήκευση κάθε νότας στην λίστα

Γραμμή 1: Κλείση συνάρτησης

Γραμμή 2: Εμφάνιση λιστας με νότες που θα αναπαραχθούν

Γραμμή 3: Εμφάνιση bit sting

Το αποτέλεσμα από την εκτέλεση του παραπάνω κώδικα βρίσκετε στην εικόνα 128

['F', 'C', 'F']
001

Εικόνα 128: Αποτέλεσμα κώδικα της εικόνας 127

Υστέρα όλη η διαδικασία του κβαντικού περιπάτου πραγματοποιείται δύο φορές η μια φορά καθορίζει τη χροιά της μελωδίας και η άλλη φορά καθορίζει τον ρυθμό της μελωδίας η διαδικασία αυτή εκτελείτε παρακάτω στην εικόνα 129:

Η ίδια διαδικασία θα επαναληφθεί ξανά αλλά όχι για την μαρκοβιανή αλυσίδα για μια άλλη λίστα που αναπαριστά τον ρυθμό.

```
1 pitch = Next_State
2
3 Rhythm = Next_State
4
5
6 next_pitch = []
7 next_rhythm = []
8
9 Sound_P = []
10 Rhythms = []
11 for i in range (30):
12
13     n = 3
14     n1 = 2
15     qr = QuantumRegister(n,"input_qubit")
16     qr1 = QuantumRegister(n1,"dice_operators")
17     cr = ClassicalRegister(n,"classic_operator")
18
19
20     third_circuit = QuantumCircuit(qr,qr1,cr)
21
22     n= move(third_circuit,pitch)
23
24     third_circuit = walker(third_circuit)
25
26     answers = Answer(third_circuit)
27
28     pitch = Next_state(answers)
29     sound_of_pitch = sound_of_the_next_state(pitch , Markov_chain_1)
30     Sound_P.append(sound_of_pitch)
31
32     next_pitch.append(pitch)
33
```

Εικόνα 129: Εκτέλεση κβαντικού περιπάτου για την εύρεση της μουσικής χροιάς

Γραμμή 2-4: Ως χροιά και ρυθμός αρχικά ορίζεται η κατάσταση που επιτρέφεται κάθε φορά από την εκτέλεση του κβαντικού περιπάτου

Γραμμή 6-7: Δημιουργία 2 λιστών, όπου στην καθεμία μετά το τέλος της διαδικασίας του κβαντικού περιπατητή αποθηκεύεται η επόμενη συμβολοσειρά

Γραμμή 9-10: Δημιουργία 2 λιστών μια αποθηκεύει τα μουσικά δεδομένα για τον ρυθμό η άλλη αποθηκεύει τα δεδομένα για την χροιά.

Γραμμή 11-17: Εισαγωγή αριθμού qubit και των βοηθητικών qubits, δημιουργία κβαντικών και κλασικού καταχωρητή

Γραμμή 20: Δημιουργία κυκλώματος

Γραμμή 22: Δημιουργία κυκλώματος το οποίο καθορίζει την κατάσταση από την οποία θα ξεκινήσει το qubit με βάση την κατάσταση που πιθανώς βρίσκετε ο περιπατητής ή αλλιώς με βάση το bit sequence της χροιάς

Γραμμή 24: Εκτέλεση του αλγόριθμου του κβαντικού περιπάτου

Γραμμή 26-28: Ευρεση της μεγιστης κατάστασης η οποία είναι η θέση που μπορεί να βρεθεί ο περιπατητής

Γραμμή 29: Χρήση του bit sequence που προέκυψε από την εκτέλεση της προηγούμενης συνάρτησης, για την βρεθούν ποιες νότες από την μαρκοβιανή αλυσίδα θα αναπαραχθούν.

Γραμμή 30: Αποθήκευση του αποτελέσματος της γραμμής 29 στην λίστα Sound_P

Γραμμή 31: Αποθήκευση του επόμενου bit sequence που καθορίζει την χροιά στην λίστα next_pitch

Η ίδια διαδικασία θα πραγματοποιηθεί για την εύρεση του ρύθμου και ο κώδικάς είναι αυτός της εικόνας 130:

```
36                                     #RHYTHM
37  rhythm = ['1n', '2dn', '2n', '4dn', '4n', '8dn', '8n', '16n']
38
39  fouth_circuit=QuantumCircuit(qr,qr1,cr)
40
41  n = move(fouth_circuit,Rhythm)
42
43  fouth_circuit = walker(fouth_circuit)
44
45  answers = Answer(fouth_circuit)
46
47  Rhythm = Next_state(answers)
48
49  Rhythm_Of_Melodies =sound_of_the_next_state (Rhythm,rhythm)
50
51  Rhythms.append(Rhythm_Of_Melodies)
52  next_rhythm.append(Rhythm)
53
54
55
56
57  for i in range(30):
58      print(next_pitch[i],'\t',next_rhythm[i],'\t',Sound_P[i],'\t',Rhythms[i])
59
```

Εικόνα 130: Εκτέλεση κβαντικού περιπάτου για την εύρεση μουσικού ρυθμού

Γραμμή 40: Δημιουργία κυκλώματος το οποίο καθορίζει την κατάσταση από την οποία θα ξεκινήσει το qubit με βάση την κατάσταση που πιθανώς βρίσκετε ο περιπατητής ή αλλιώς με βάση το bit sequence του ρυθμού

Γραμμή 42: Εκτέλεση του αλγόριθμου του κβαντικού περιπάτου

Γραμμή 44-46: Ευρεση της μεγιστης κατάστασης η οποία είναι η θέση που μπορεί να βρεθεί ο περιπατητής

Γραμμή 48: Χρήση του bit sequence που προέκυψε από την εκτέλεση της προηγούμενης συνάρτησης, για να βρεθούν ποιες νότες από την λίστα που περιέχει ένα σύνολο από ρυθμούς

Γραμμή 50: Αποθήκευση του αποτεέσματος της γραμμής 29 στην λίστα Sound_P

Γραμμή 51: Αποθήκευση του επόμενου bit sequence που καθορίζει την χροιά στην λίστα next_pitch

Γραμμή 56: Εμφάνιση του bit-sequence της χροιάς και του ρυθμού τα οποία καθορίζουν την εμφάνιση των μουσικών νοτών και του ρυθμου της κάθε νότας

Το αποτέλεσμα που προκύπτει μετα την εκτέλεση της γραμμής 56 αναπαρίστατε σ την εικονα 131:

000	100	['F', 'C', 'F']	['16n', '1n', '2dn']
100	011	['F', 'E', 'F']	['2dn', '1n', '16n']
001	100	['F', 'C', 'F']	['16n', '1n', '2dn']
000	000	['F', 'C', 'F']	['2dn', '2n', '4dn']
001	000	['F', 'C', 'F']	['2dn', '2n', '4dn']
100	100	['F', 'E', 'F']	['16n', '1n', '2dn']
101	011	['F', 'E', 'F']	['2dn', '1n', '16n']
100	010	['F', 'E', 'F']	['2dn', '1n', '2dn']
000	011	['F', 'C', 'F']	['2dn', '1n', '16n']
001	110	['F', 'C', 'F']	['16n', '8n', '16n']
101	111	['F', 'E', 'F']	['16n', '8n', '8dn']
101	111	['F', 'E', 'F']	['16n', '8n', '8dn']
101	111	['F', 'E', 'F']	['16n', '8n', '8dn']
001	110	['F', 'C', 'F']	['16n', '8n', '16n']
100	010	['F', 'E', 'F']	['2dn', '1n', '2dn']
001	011	['F', 'C', 'F']	['2dn', '1n', '16n']
101	100	['F', 'E', 'F']	['16n', '1n', '2dn']
100	001	['F', 'E', 'F']	['2dn', '2n', '2dn']
001	110	['F', 'C', 'F']	['16n', '8n', '16n']
101	011	['F', 'E', 'F']	['2dn', '1n', '16n']
001	010	['F', 'C', 'F']	['2dn', '1n', '2dn']
100	010	['F', 'E', 'F']	['2dn', '1n', '2dn']
000	110	['F', 'C', 'F']	['16n', '8n', '16n']
000	010	['F', 'C', 'F']	['2dn', '1n', '2dn']
100	010	['F', 'E', 'F']	['2dn', '1n', '2dn']
001	010	['F', 'C', 'F']	['2dn', '1n', '2dn']
110	110	['F', 'G', 'F']	['16n', '8n', '16n']
111	011	['F', 'G', 'Cs']	['2dn', '1n', '16n']
110	010	['F', 'G', 'F']	['2dn', '1n', '2dn']
111	110	['F', 'G', 'Cs']	['16n', '8n', '16n']

Εικόνα 131: Αποτελέσματα χροιάς και ρυθμού

Υστέρα γίνεται επιλογή της τελευταίας νότας και του τελευταίου μουσικού ρυθμου

Και στην συνέχεια τα δεδομένα από το jupyter notebook αποστέλονται στην Sonic Pi και δημιουργείτε μια μελωδία και ο κώδικας είναι στις εικόνες 132, 133:

```
1 def last_played_note (Sound_P):
2     for i in range(29):
3         return(Sound_P[29])
4
5
6 last_note= last_played_note(Sound_P)
7 print(last_note)
8
9 def last_played_rhythm(Rhythms):
10     for i in range(29):
11         return(Rhythms[29])
12
13 last_rhythm = last_played_rhythm(Rhythms)
14
15 client = udp_client.SimpleUDPClient("192.168.1.3", 4560)
16 client.send_message("/play",last_note)
17 client.send_message("/rhythm",last_rhythm)
```

Εικόνα 132: Αποστολή δεδομένων από Jupyter Notebook σε Sonic Pi

Γραμμή1-3 :Δημιουργία συνάρτησης που παράγει την τελευταία νότα

Γραμμή 9-10 :Δημιουργία συνάρτησης που παράγει τον τελευταίο ρυθμό

Γραμμή 15: Δημιουργία πελάτη που θα στείλει την τελευταία νότα και τον τελευταίο ρυθμό

Γραμμή 16-17: Αποστολή νότας και ρυθμού στην Sonic Pi

Ο κώδικας στην Sonic Pi είναι στις εικόνες 133, 134:

```
4 live_loop :Quantum_Walk_1 do
5   use_bpm 60 # Beats per minute
6   use_synth :prophet # Synthesizer type
7   music = sync "/osc*/play"
8   rhythm = sync "/osc*/rhythm"
9
10  define :sound do|root,type_1|
11    with_fx :echo do
12      with_synth :tb303 do
13        play_chord chord(root,type_1), release: 9 ,attach: 1.5
14      end
15    end
16  end
17
18  music each do |m|
```

Εικόνα 133: Αναπαραγωγή μουσικής σε Sonic Pi

Γραμμή 4: Δημιουργία επαναληπτικού βρόγχου

Γραμμή 6:Χρήση πρώτου ηλεκτρονικού οργάνου

Γραμμή 7-8: Ανάκτηση δεδομένων από το jupyter notebook

Γραμμή 10: Ορισμός συνάρτησης που καθορίζει τον ήχο που θα αναπαραχθεί από καθένα από τα στοιχεία των στηλών

Γραμμή 11: Χρήση πρώτου μουσικού εφέ

Γραμμή 12: Χρήση δευτερού ηλεκτρονικού οργάνου

Γραμμή 13: Εκτέλεση των μουσικών στοιχείων με βάση παραμέτρους της sonic pi, όπως το release το οποίο μεγαλώνει την διάρκεια της νότας, το attack το οποίο κάθε 2 δευτερόλεπτα δίνει μια έμφαση στο ήχο

```
18 music.each do |m|
19   | rhythm.each do |r|
20     |   sound(m, :minor)
21     |   sleep r
22     |   end
23   end
24 end
```

Εικόνα 134: Αναπαραγωγή μουσικής σε Sonic Pi

Γραμμή 18: Για κάθε μουσική νότα

Γραμμή 19:Για κάθε μουσικό ρυθμό

Γραμμή 20:Εκτέλεσε την συναρτηση ήχου

Γραμμή 21: Παύση της μουσικής με βάση τα στοιχεία του ρυθμού

Τα στοιχεία του πίνακα που καθορίζουν τον μουσικό ρυθμό ερμηνεύονται ως εξής:

- 1n' (one note) - Μία νότα. Αναφέρεται συνήθως σε μία πλήρη νότα.
- 2dn (two dots and a note) - Μισή διπλή νότα, η οποία έχει τη μισή διάρκεια μιας πλήρους νότας.
- 2n (two notes) - Μισή νότα, η οποία έχει τη μισή διάρκεια μιας πλήρους νότας.
- 4dn (four dots and a note) - Τέταρτη διπλή νότα, η οποία έχει το ένα τέταρτο της διάρκειας μιας πλήρους νότας.
- 4n (four notes) - Τέταρτη νότα, η οποία έχει το ένα τέταρτο της διάρκειας μιας πλήρους νότας.
- 8dn (eight dots and a note) - Όγδοη διπλή νότα, η οποία έχει το ένα όγδοο της διάρκειας μιας πλήρους νότας.
- 8n (eight notes) - Όγδοη νότα, η οποία έχει το ένα όγδοο της διάρκειας μιας πλήρους νότας.
- 16n (sixteen notes) - Δέκα έκτο της διάρκειας μιας πλήρους νότας.

5.4 Αλγόριθμος του Grover

Αξίζει να αναφερθεί πως ο αλγόριθμος του κβαντικού περιπάτου μπορεί να οδηγήσει στην παραγωγή μουσικής, όμως η μουσική που παράγει δεν συμβαδίζει πάντα με τους μουσικούς κανόνες. Καθώς σε κάθε μουσικό στοιχείο δίνει την δυνατότητα πρόσβασης μόνο στο επόμενο και στο προηγούμενο μουσικό στοιχείο από το τωρινό. Αυτό έχει σαν αποτέλεσμα ένας κανόνας όπως ο “Κανόνας 2” να μην μπορεί να εκτελέσει όλες τις νότες

Πως μπορεί να λυθεί αυτό το πρόβλημα?

Αυτό το πρόβλημα μπορεί να λυθεί αν για την παραγωγή μουσικής από κβαντικούς αλγόριθμους χρησιμοποιηθεί ο αλγόριθμος του Grover. Ο αλγόριθμος του Grover έχει την δυνατότητα να βρεί οποιαδήποτε νότα χρειαστεί και γενικά οποιοδήποτε στοιχείο μέσα σε ένα σύνολο. Ο αλγόριθμος του Grover αρχικά πήρε το όνομα του από τον Lon Grover και είναι ο πιο γρήγορος αλγόριθμος αναζήτησης καθώς σε μια μη ταξινομημένη λιστα μπορεί να βρεί το υποαναζήτητο στοιχείο σε $O(\sqrt{N})$ χρόνο. Δηλαδή η επιτάχυνση του είναι τετραγωνική σε σχέση με την επιτάχυνση ενός κλασικού αλγόριθμου.

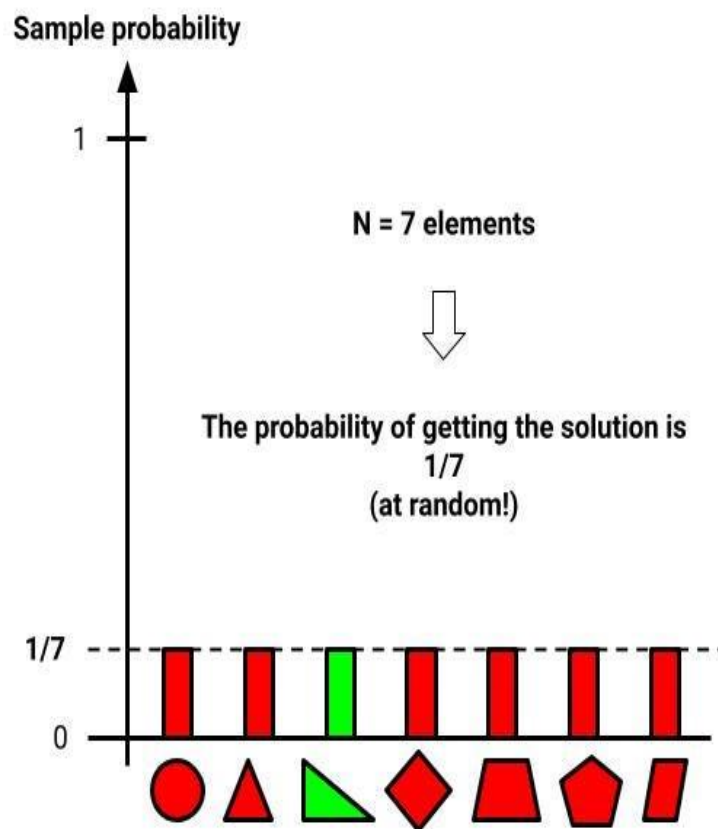
Πως μπορεί ένας αλγοριθμός να είναι τόσο γρήγορος σε ένα τόσο μικρό χρονικό διάστημα?

Μέσω μια τεχνικής η οποία είναι γνωστή ως ενίσχυση πλάτους. Η ενίσχυση πλάτους βασίζεται σε δύο τελεστές η αλλιώς πύλες το ένα είναι το μαντείο(Oracle) και το άλλο είναι η αντανάκλαση ως προς τον μέσω όρο(Diffuser).

Αρχική με την βοήθεια του μαντείου είναι ευκολή η εύρεση του υποαναζήτησης στοιχείου καθώς μαρκάρει το στοιχείο η τα στοιχεία που είναι προς αναζήτηση. Ο τρόπος με τον οποίο μαρκάρει τα υποαναζήτησης στοιχεία είναι με την αναστροφή της φάσης των στοιχείων. Δεν μεταβάλλει ούτε αυξάνει την πιθανότητα εμφάνισης των στοιχείων, απλώς τα εντοπίζει ποια είναι έτσι ώστε ο Diffuser στην συνέχεια να μπορεί να εκτελεστεί. Το Oracle (μαντείο) χωρίς τον Diffuser δεν θα είχε κάποια χρησιμότητα καθώς όπως αναφέρθηκε και πριν δεν μεταβάλλει την πιθανότητα εμφάνισης κάποιου στοιχείου απλά αλλάζει την φάση του. Για καλύτερη κατανόηση του μαντείου χρησιμοποιείται η εικόνα 135 η οποία δείχνει την επιροή του μαντείου σε όλα τα στοιχεία και πως καμία πιθανότητα δεν αλλάζει.

Σε αντίθεση με τον τελεστή Oracle Diffuser, εκτελείται πάνω από μια φορές ο ρόλος του είναι να ενισχύει το πλάτος της μαρκαρισμένης από το Oracle κατάστασης.

Οσές φορές αναπαραχθεί αντιστροφή των καταστάσεων γύρω από την μέση κατάσταση δηλαδή γύρω από την υποαναζήτηση κατάσταση, τόσο μεγαλύτερη θα είναι και η πιθανότητα της να βρεθεί.



Εικόνα 135: Εφαρμογή του Oracle

(<https://towardsdatascience.com/behind-oracles-grovers-algorithm-amplitude-amplification-46b928b46f1e>)

Υστερά για καλύτερη κατανόηση θα γίνει αναπαράσταση του αλγόριθμου Grover με το Oracle και το Diffuser με σκοπό την εύρεση κάθε φοράς της νότας που είναι απαραίτητη.

Αρχικά η πρώτη νότα που θα χρειαστεί να αναζητηθεί είναι η τελευταία νότα που αναπαράχθηκε από τον αλγόριθμο του κβαντικού περπατητή. Στην εικόνα 136 αναπαρίστατε η νότα με την οποία θα γίνει η πρώτη εκτέλεση του αλγόριθμου

```
1 Final_Note = (last_note[2])  
2 print(Final_Note)
```

Cs

Εικόνα 136: Αρχική νότα του Grover

Υστερά για την νότα γίνεται εύρεση της θέσης που έχει στον αρχικό πίνακα μετάβασης και δημιουργείτε μια λίστα με βάση τις πιθανότητες που έχουν οι άλλες νοτές να εμφανιστούν μετά την τωρινή νότα..

Δηλαδή με την τιμή 1 θα συμβολίζεται η θέση αυτών που εμφανίζονται ενώ με την τιμή 0 συμβολίζεται η θέση αυτών που δεν εμφανίζονται μετά την τωρινή νότα.

Στην εικόνα 137 ακολουθεί η δημιουργία της λίστας για την νότα:

```

1 # print(Last_Note)
2 def Finaly_Note(Note):
3     Last_Note = list(trasition_matrix[Note].values())
4
5     notes = []
6     m = 0
7     for i in range(len(Last_Note)):
8         if Last_Note[i] != 0:
9             Last_Note[i] = 1
10            m+=1
11        elif Last_Note[i] == 0:
12            Last_Note[i] ==0
13        notes.append(Last_Note[i])
14    return(notes)
15
16 Notes =Finaly_Note(Final_Note)
17 print(Notes)

```

Εικόνα 137: Δημιουργία της λίστας για την νότα

Γραμμή 1: Συναρτηση που δημιουργεί την νότα σε λίστα

Γραμμή 2: Ευρεσή λίστας που αντιστοιχεί στην λίστα της τελευταίας νότας που αναπαράχτηκε και περιέχει τις πιθανότητες εμφάνισης κάθε άλλης νότας μετά την εκτέλεση της συγκεκριμένης νότας

Γραμμή 7-13: Εάν στην λίστα με τις πιθανότητες η πιθανότητα εμφάνισης κάποιας νότας είναι 0, τότε στην τελική λίστα στην αντιστοιχη θέση θα υπάρχει η τιμή 0, ενώ όταν η πιθανότητα εμφάνισης κάποια νότας δεν είναι μηδέν τότε στην τελική λίστα η τιμή θα είναι 1

Για την Cs νότα η λίστα αναπαρίστατε στην εικόνα 138:

```
[0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0]
```

Εικόνα 138: Λίστα που αντιστοιχεί στην Cs

Στην συνέχεια σύμφωνα με τις θέσεις των στοιχείων της στήλης ακολουθεί αντιστοίχηση τους σε. Ο κώδικας αναπαρίστατε στην εικόνα 139

```

1  d = {
2      0: '0000',
3      1: '0001',
4      2: '0010',
5      3: '0011',
6      4: '0100',
7      5: '0101',
8      6: '0110',
9      7: '0111',
10     8: '1000',
11     9: '1001',
12     10: '1010',
13     11: '1011',
14     12: '1100',
15     13: '1101',
16     14: '1110',
17     15: '1111',
18 }
19
20 def Upsidown_Notes(List):
21     import random
22     states = []
23     reversed_states = []
24     for i, j in enumerate (List):
25         if j == 1:
26             All_States = d[i]
27             states.append(All_States)
28
29     for i in states:
30         reversed_states.append(i[::-1])
31     r_states = random.choice(reversed_states)
32     # print(reversed_states)
33     return r_states
34
35 State =Upsidown_Notes(Notes)
36 # State = random.choice(r_states)
37 # print(r_states)
38 print(State)

```

Εικόνα 139: Δημιουργία λίστας που αντιστοιχεί στην Cs

Γραμμή 20: Δημιουργία συνάρτησης

Γραμμή 24-27: Ανάλογα με την θέση του κάθε στοιχείο της λίστας, αποθήκευση στο σύνολο των καταστάσεων την κατάσταση.

Αν το στοιχείο είναι 1 τότε εμφάνιση και αποθήκευση στο σύνολο των καταστάσεων αυτή την κατάσταση που αντιστοιχεί στην θέση που βρίσκεται το 1.

Γραμμή 29-30: Για κάθε κατάσταση του συνόλου γίνεται ανστροφή καθώς στην συνέχεια θα χρησιμοποιηθούν στο κύκλωμα.

Γραμμή 31: Τυχαία επιλογή της κατάσταση που θα αποτελέσει την υποαναζήτηση κατάσταση στον αλγόριθμο του Grover

Γραμμή 35: Κλήση συνάρτησης

Γραμμή 38: Εμφάνιση κατάστασης

Η προς αναζήτηση κατάσταση αναπαρίστατε στην εικόνα 140:

1010

Εικόνα 140: Προς αναζήτηση κατάσταση

Υστερα ακολουθεί ο κώδικας για το Oracle στην εικόνα 141

```

1 from qiskit.quantum_info import Operator
2 from qiskit.visualization import array_to_latex
3 import qiskit.circuit.library
4 η
5 n = 6
6 qr = QuantumRegister(n,"input_qubit")
7 cr = ClassicalRegister(4,"classic_operator")
8
9 ORACLE = QuantumCircuit(qr , cr)
10
11
12 def Oracle(q,State):
13
14     for i , j in enumerate(State):
15         if j == '1':
16             q.i(qr[i])
17         elif j == '0':
18             q.x(qr[i])
19
20     q.barrier()
21     q.barrier()
22     q.mct([qr[0],qr[1] ,qr[2],qr[3]],qr[4])
23
24     for i , j in enumerate(State):
25         if j == '1':
26             q.i(qr[i])
27         elif j == '0':
28             q.x(qr[i])
29     # qc.barrier()
30     # 0 =qc.to_gate()
31     return q

```

Εικόνα 140: Δημιουργία Oracle

Γραμμή 1-3: Εισαγωγή των απαραίτητων βιβλιοθηκών

Γραμμή 5:Εισαγωγή αριθμού qubit

Γραμμή 6-9: Δημιουργία κβαντικού καταχωρητη,κλασικου καταχωρητη και το κύκλωμα

Γραμμή 12: Συνάρτηση μαντείο, εισαγωγή κυκλώματος και κβαντικής κατάστασης

Γραμμή 14-18: Εάν το bit της κατάστασης είναι 1 τότε δεν εφαρμόζεται καμιά πύλη εάν είναι 0 εφαρμόζεται η πύλη x ώστε να γίνει αναστροφή της φάσης, με σκοπό να «στοχοποιηθεί» το υπο αναζήτηση στοιχείο

Γραμμή 22: Εφαρμογή multi controled gate η οποία θα αντιστρέψει την φάση του qubit

Γραμμή 24-28: Εφαρμογή ξανά των ίδιων πυλών με τις γραμμές 14-18 ώστε να ανερεθούν

Γραμμή 31: Επιστροφή κυκλώματος

Στην συνέχεια η εμφάνιση του κυκλώματος πραγματοποιείται με την χρήση της εντολής στην εικόνα 141:

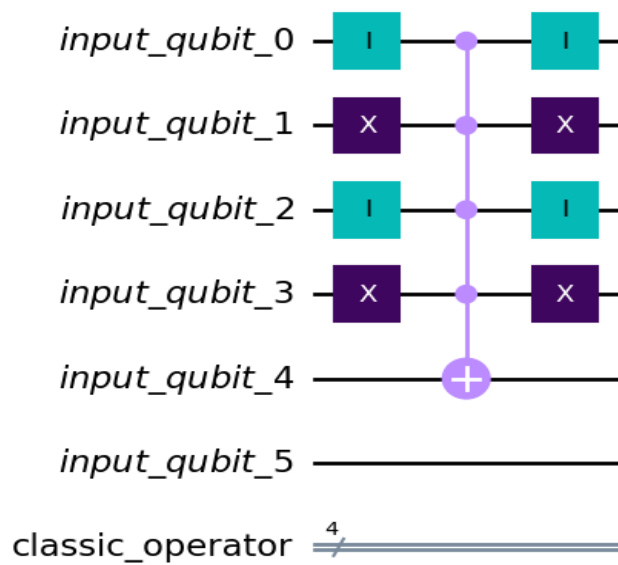
```

38 ORACLE.draw('mpl',style = gate_colours)

```

Εικόνα 141: Κωδικάς για αναπαράσταση κυκλώματος του Oracle

Το κύκλωμα αναπαρίστατε στην εξής εικόνα 142:



Εικόνα 142: Κυκλώμα Oracle

Υστερά ακολουθεί η εικόνα 143 περιέχει τον κώδικα για τον Diffuser

```

3  n = 5
4  qr = QuantumRegister(n,"input_qubit")
5
6  # cr = ClassicalRegister(3,"classic_operator")
7
8  circuit = QuantumCircuit(qr)
9
10 def diffusser(qc):
11     for i in range (n - 2):
12         qc.h(i)
13         qc.x(i)
14
15     qc.mct([qr[0],qr[1] ,qr[2],qr[3]],qr[4])
16
17     for i in range (n - 2):
18         qc.h(i)
19     for i in range (n - 1):
20         qc.x(i)
21
22     D =qc.to_gate()
23     D.name = 'Diffuser'
24     return D
25
26 Diffusser = diffusser(circuit)
27

```

Εικόνα 143: Κώδικας για Diffuser

Γραμμή 3: Εισαγωγή αριθμών qubit

Γραμμή 4: Δημιουργία κυκλώματος για να εφαρμοστεί

Γραμμή 10: Δημιουργία συνάρτησης που αποτελεί τον diffusser

Γραμμή 11 – 13: Εφαρμογή πυλών ώστε πριν την αντανάκλαση προς την μαρκαρισμένη κατάσταση ώστε όλες οι καταστάσεις να έχουν ίδιες πιθανότητες εμφάνισης και όσο γίνεται η αντανάκλαση προς την μαρκαρισμένη περιοχή τόσο να μειώνονται οι πιθανότητες των άλλων καταστάσεων

Γραμμή 15: Εφαρμογή πύλης που θα κάνει την αντανάκλαση ως προς την υπο αναζήτηση κατάσταση

Γραμμή 17-19: Αναίρεση πυλών

Γραμμή 24: Επιστροφή της συνάρτησης ως πύλη

Στην συνέχεια ακολουθεί ο αλγόριθμος του Grover ο κώδικας αναπαρίστατε στην εικόνα 144

```
1 import numpy as np
2
3 # state = State
4 # State_1 = state [::-1]
5 n = 6
6 n1 = 4
7 qr = QuantumRegister(4,"input_qubit")
8 aux1 = QuantumRegister(1,"auxiliary1")
9 aux2 = QuantumRegister(1,"auxiliary2")
10 cr = ClassicalRegister(4,"classic_operator")
11
12 qc = QuantumCircuit(qr,cr,aux1,aux2)
13 def Grover_Algorithm(qc,Unitary_matrix,Diffusser):
14     Oracle = np.array(Unitary_matrix, dtype = complex)
15
16     per = int ((np.pi /4)*np.sqrt(2**(n - 2)) )
17     print(per)
18     for i in range(n - 2):
19         qc.h(i)
20
21     qc.x(aux1[0])#Grovers Qubit
22     qc.i(aux2[0])#ORACLES/DIFFUSERS QUBIT
23
24
25     for i in range(4):
26         qc.i(i)
27         qc.h(aux1[0])#Grovers Qubit
28         qc.barrier()
29
30     for k in range((per)):
31         qc.unitary(Oracle, [0,1,2,3,4,5],label = 'Oracle')
32         qc.barrier()
33
34
35     qc.append(Diffusser, [0,1,2,3,4])
36
37     qc.barrier()
```

Εικόνα 144: Αλγόριθμος του Grover

Γραμμή 5-6: Εισαγωγή βοηθητικών και μη qubit

Γραμμή 7- 10: Δημιουργία κβαντικών και κλασικού καταχωτή

Γραμμή 13: Δημιουργία συνάρτησης του αλγορίθμου

Γραμμή 16: Ο αριθμός των φορών που θα εκτελεστεί ο αλγόριθμος

Γραμμή 18-19: Handmard σε όλα τα qubits ώστε να έχουν όλες οι καταστάσεις ίδιες πιθανότητες εμφάνισης

Γραμμή 21: Εφαρμογή x πύλης στο πρώτο βοηθητικό qubit

Γραμμή 22: Εφαρμογή I πύλης στο δεύτερο qubit, υπάρχουν δύο βοηθητικά qubit καθώς με αυτόν τον τρόπο βρίσκεται πιο ευκόλα η αναζητούμενη κατάσταση

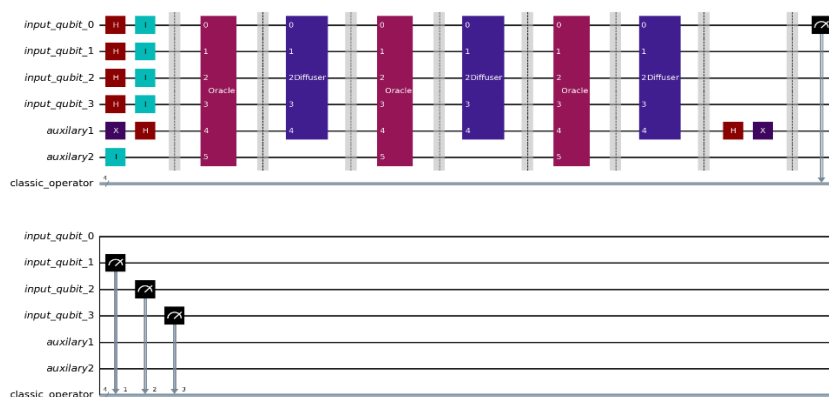
Γραμμή 30 : Εφαρμογή και Oracle και Diffuser όσες φορές χρειαστεί

Γραμμή 31-35: Εφαρμογή των πυλών

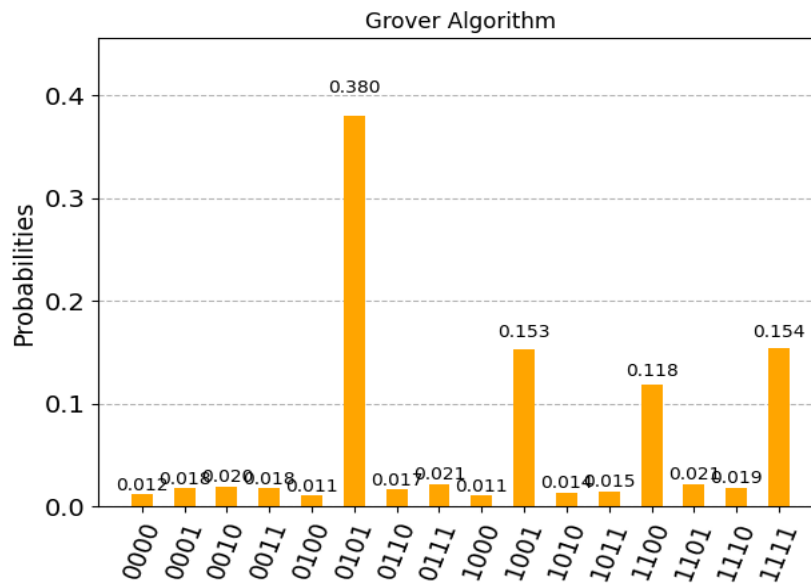
```
50  
51 Grover = Grover_Algorithm(qc,Unitary_matrix,Diffusser)  
52 Grover.draw("mpl",style = gate_colours)  
53
```

Εικόνα 145: Κώδικας για αναπαράσταση του κυκλώματος του Grover

Με την εκτέση των εντολών στην εικόνα 145 πραγματοποιείται η αναπαράσταση του κυκλωματος και του ιστογράμματος του αλγορίθμου, τα οποίο βρίσκονται στις εικόνες 146, 147



Εικόνα 146: Κύκλωμα του Grover



Εικόνα 147: Ιστόγραμμα του Grover

Υστερα αφού εκτελεστεί ο αλγόριθμος την πρώτη φορά τα δεδομένα θα αποθηκευτούν σε μια μεταβλητή, στην εικόνα 148 γίνεται επιστροφή αυτής της μεταβλητής ενώ στην εικόνα γίνεται ευρεση της κατάστασης με την μέγιστη πιθανότητα εμφάνισης

```

1 def Grovers_States(circuit):
2     job = execute(circuit, Aer.get_backend('qasm_simulator'), shots = 1024).result()
3     answer = job.get_counts()
4     plot_histogram(job.get_counts(circuit), color='maroon', title='Grover Algorithm')
5     return answer
6

```

Εικόνα 148: Αποθήκευση δεδομένων σε μια μεταβλητή και επιστροφή της

Γραμμή 1: Δημιουργία συνάρτησης που αποθηκεύει τα δεδομένα του κυκλώματος μετα την πρώτη αναπαράσταση του ιστογράμματος

Γραμμή 2: Εκτέλεση προγράμματος σε κβαντικό προσομοιωτή

Γραμμή 3: Αποθήκευση δεδομένων σε μια μεταβλητή

Γραμμή 5: Επιστροφή μεταβλητής

```

1 All_Grovers_States = Grovers_States(qc)
2
3 def Next_Grovers_State(all_states):
4     # print(All_Grovers_States)
5
6     Keys = list(All_Grovers_States.keys())
7     Values = list(list(All_Grovers_States.values()))
8
9
10    next_grover_state = Keys[Values.index(max(Values))]
11    print(next_grover_state)
12    return next_grover_state
13

```

Εικόνα 149: Ευρεση μέγιστης κατάστασης και επιστροφής της

Γραμμή 3: Δημιουργία συνάρτησης η οποία δέχεται ως είσοδο τα δεδομένα του αλγόριθμου που είναι όλες οι καταστάσεις με τις πιθανότητες εμφάνισης, λειτουργία αυτής της συνάρτησης είναι η επιστροφή της κατάστασης με την μεγαλύτερη πιθανότητα εμφάνισης

Γραμμή 6: Δημιουργία λίστας που περιέχει όλες τις καταστάσεις

Γραμμή 7: Δημιουργία λίστας που περιέχει τις πιθανότητες εμφάνισης για τις καταστάσεις

Γραμμή 10: Εύρεση κατάστασης η οποία είναι στην θέση όπου βρίσκεται η μέγιστη τιμή πιθανότητας και θεωρείτε η επόμενη κατάσταση

Γραμμή 12: Επιστροφή μέγιστης κατάστασης

Στην συνέχεια για την εύρεση της επόμενης κατάστασης η διαδικασία βρίσκεται στην εικόνα 150

```

1 NEXT = Next_Grovers_State(All_Grovers_States)
2
3 index_of_grover_state = d1[NEXT]
4 print(index_of_grover_state)
5
6 Keys_Of_Last_Note = list(trasition_matrix.keys())
7 print(Keys_Of_Last_Note)
8
9 k = Keys_Of_Last_Note[index_of_grover_state]
10 print(k)
11 # print(A)

```

0111
7
['E', 'Ds', 'Cs', 'G', 'D', 'F', 'C', 'Fs', 'A', 'Gs', 'B', 'As']
Fs

Εικόνα 150: Διαδικασία για την εύρεση της επόμενης κατάστασης που θα εισαχθεί στο Grover

Γραμμή 1: Εύρεση επόμενη κατάσταση

Γραμμή 2: Αντικατάσταση της κατάστασης που στο αρχικό λεξιλόγιο

Γραμμή 6: Εμφάνιση όλων των υποψήφιων κλειδιών

Γραμμή 7: Εμφάνιση της νότας που τελικά προκύπτει με βάση τα παραπάνω

Τέλος για να παραχθεί μια μελωδία από το αλγόριθμο Grover ακολουθεί η εξής διαδικασία η οποία φαίνεται στις εικόνες 151,152,153,154:

```

1 List = []
2 N2 = 0
3 while N2 < 12:
4     # List.append(k)
5
6     Note = Finally_Note(k)
7     # print(Note)
8     #Тых;ia Epilog;h
9
10    Upside = Upsidown_Notes(Note)
11    print(Upside)
12
13    # Oracle
14    n = 6
15    qr = QuantumRegister(n,"input_qubit")
16    cr = ClassicalRegister(4,"classic_operator")
17
18    ORACLE_1 = QuantumCircuit(qr , cr)
19
20    oracle_1 = Oracle(ORACLE_1,Upside)
21    oracle_operator = Operator(oracle_1)
22
23    # # Get the unitary matrix of the oracle
24    Unitary_matrix = oracle_operator.data
25    #Diffuser
26    n = 5
27    qr = QuantumRegister(n,"input_qubit")
28
29    # cr = ClassicalRegister(3,"classic_operator")
30
31    circuit_1 = QuantumCircuit(qr)
32    Diffuser = diffusser(circuit_1)

```

Εικόνα 151: Δημιουργία Oracle και Difusser

Γραμμή 6: Είσοδος νότας

Γραμμή 10: Αναστροφή κατάστασης ώστε να χρησιμοποιηθεί στο κύκλωμα

Γραμμή 14-20: Υπολογισμός μαντείου

Γραμμή 20-28: Υπολογισμός Diffuser

```
30 circuit_1 = QuantumCircuit(qr)
31 Diffuser = diffusser(circuit_1)
32
33 #Grover
34 n = 6
35 n1 = 3
36 qr = QuantumRegister(4,"input qubit")
37 aux1 = QuantumRegister(1,"auxiliary1")
38 aux2 = QuantumRegister(1,"auxiliary2")
39 cr = ClassicalRegister(4,"classic_operator")
40
41 Grover_Circuit=QuantumCircuit(qr,cr,aux1,aux2)
42
43
44
45 Grover_1 = Grover_Algorithm(Grover_Circuit,Unitary_matrix,Diffuser)
46 # Grover_1.draw("mpl",style = gate_colours)
47
48 All_Grover_States = Grover_States(Grover_1)
49
50 NEX1 = Next_Grover_State(All_Grover_States)#Εμφανίζει την καλύτερη λύση
51 # AI = RUN_Again(NEX1,outside,d1)
52 index_of_grover_state = d1[NEX1]
53 print(index_of_grover_state)
54
55 Keys_of_Last_Note = list(trasition_matrix.keys())#Την νότα που αντιστοιχεί στην θέση 1
56
57 k = Keys_of_Last_Note[index_of_grover_state]
58 Note = Finaly_Note(k)
59 if k not in List:
60     List.append(k)
61 N2+=1
62
63 client = udp_client.SimpleUDPClient("192.168.1.3", 4560)
64 client.send_message("/Play",List)
65 print(List)
```

Εικόνα 152: Εκτέλεση του Grover της διαδικασίας της εικόνας 150 και αποστολή δεδομένων στην Sonic Pi

Γραμμή 30-45 : Εκτέλεση Grover

Γραμμή 50-58: Διαδικασία εύρεσης επόμενης κατάστασης

Γραμμή 59-60 : Εισαγωγή νότα στην τελική λίστα μόνο αυτή αν δεν είναι μέσα

Γραμμή 63-65: Αποστολή λίστας στην Sonic Pi

Το αποτέλεσμα στην Sonic Pi αναπαριστάτε στις εικόνες 153, 154 ως

```
3 use_bpm 60
4 use_real_time
5 live_loop :Groover_24 do
6   use_bpm 60
7   music = sync "/osc*/Play_1"
8
9   define :Grover do|root , type|
10     with_fx :reverb, room: 0.9 do
11       with_synth :prophet do
12         play_chord chord(root, type),attack: 3,release: 8, cutoff: 70
13         sleep 8
14       end
```

Εικόνα 153: Αποστολή δεδομένων σε Sonic Pi και δημιουργία μελωδίας

Γραμμή 5: Εισαγωγή δεδομένων από Jupyter Notebook

Γραμμή 9: Δημιουργία συνάρτησης που καθορίζει την μουσική που προκύπτει από τα δεδομένα

Γραμμή 10-11: Δημιουργία μουσικής κατεύθυνσης της μελωδίας

Γραμμή 12: Αναπαραγωγή κάθε στοιχείου της μελωδίας

```
18 music.each do |m|
19   1.times do
20     Grover(m, :major)
21   end
22 end
23 end
24 end
```

Εικόνα 154: Αποστολή δεδομένων σε
Sonic Pi και δημιουργία μελωδίας

Γραμμή 18-24: Για κάθε στοιχείο που εισάγεται από το jupyter notebook εκτέλεση της συνάρτησης για αναπαραγωγής μελωδίας

5.5 Αλγόριθμος Bernstein–Vazirani

Στο προηγούμενο κεφάλαιο πραγματοποιήθηκε η εύρεση κρυφής συμβολοσειράς μέσω κλασικής και όχι κβαντικής προσέγγισης. Αν η συμβολοσειρά αποτελείται από n bits τότε θα χρειαστεί να γίνουν n αναζητήσεις. Ενώ μέσω κβαντικής προσέγγισης θα χρειαστεί να γίνει μία αναζήτηση και όχι n αναζητήσεις. Η εύρεση μια κβαντικής συμβολοσειράς γίνεται με την χρήση του αλγορίθμου Bernstein–Vazirani. Ο συγκεκριμένος αλγόριθμος δημιουργήθηκε το 1997 από τον Ethan Bernstein και Umesh Vazirani και σκοπός του αλγορίθμου είναι να βρίσκει την κρυφή συμβολοσειρά με μια μόνο αναζήτηση.

Στην εικόνα 155 αναπαριστάτε ο εξής αλγόριθμος όπου η κρυφή συμβολοσειρά καθορίζεται από τον χρήστη

```

1  from qiskit import *
2
3  bit_string = (input('Input bit string:'))
4
5  secret_message = bit_string[::-1]
6  def bernstein_vazirani(message):
7      n1 = len(message)
8      qr_1 = QuantumRegister(n1 + 1, 'qubit')
9      cr = ClassicalRegister(n1, 'bit')
10
11     circuit = QuantumCircuit(qr_1, cr)
12
13     for i in range((n1)):
14         circuit.i(i)
15         circuit.x(n1)
16
17     for i in range((n1)):
18         circuit.h(i)
19         circuit.h(n1)
20
21     circuit.barrier()
22
23     for k, j in enumerate(message):
24         if j == '1':
25             circuit.cx(k, n1)
26     circuit.barrier()
27
28     for i in range(n1):
29         circuit.h(i)
30     circuit.barrier()
31     circuit.measure(range(n1), range(n1))
32
33     return circuit
34

```

Εικόνα 155: Αλγόριθμος του Bernstein- Vazirani

Γραμμή 1: Εισαγωγή βιβλιοθηκών

Γραμμή 3: Εισαγωγή του κρυφού bit string

Γραμμή 5: Αναστροφή του bit sequence καθώς τα δεδομένα μετά την εκτέλεση ενός κβαντικού προγράμματος εμφανίζονται ανάποδα από ότι είναι κανονικά

Γραμμή 6: Συνάρτηση αλγορίθμου Bernstein Vazirani

Γραμμή 7: Μήκος συμβολοσειράς

Γραμμή 9:Εισαγωγή κβαντικού καταχωρητή ο οποίος περιέχει ένα βοηθητικό qubit .Ωστε ο αριθμός των qubits που θα χρησιμοποιηθούν να συμβαδίζει με το μήκος της συμβολοσειράς

Γραμμή 10: Δημιουργία κλασικού καταχωρητή Γραμμή 12:Δημιουργία κβαντικού κυκλώματος

Γραμμή 18-20: Εφαρμογή Handmard πύλης σε όλα τα qubits έτσι ώστε να έχουν ταυτοχρονά και την τιμή 0 και την τιμή 1

Γραμμή 25-27: Για κάθε στοιχείο του bit sequence, εάν το στοιχείο έχει την τιμή 1 τότε εφαρμόζεται cx με control qubit να είναι στην θέση που είναι το στοιχείο που έχει την τιμή 1 και target qubit να είναι το βοηθητικό στοιχείο, το οποίο θα μεταβάλεται κάθε φορά.

Γραμμή 30 – 33: Αναίρεση της πύλης Handamard και μέτρηση του κυκλώματος

Εστώ πως η κρυφή συμβολοσειρά αναπαρίστατε στην εικόνα 156

Input bit string:1010

Εικόνα 156: Κρυφή συμβολοσειρά

Για την ευρέση της μέσ την βοήθεια του κβαντικού αλγόριθμου θα χρησιμοποιηθεί ο κώδικας της εικόνας 157

```
1 Bernstein_Vazirani=bernstein_vazirani(secret_message)
2 job = execute(Bernstein_Vazirani, Aer.get_backend('qasm_simulator'),shots = 1).result()
3 result = job.get_counts()
4 data = result.keys()
5 Data = list(data)
6 print(Data)
```

Εικόνα 157: Ευρεση κρυφής συμβολοσειρά

Γραμμή 1: Εκτέλεση της συνάρτησης τους αλγορίθμου με κρυφήσυμβολοσειρά την 1010

Γραμμή 2: Εκτέλεση αλγορίθμου σε κβαντικό προσομοιωτή

Γραμμή 3 : Αποθηκευση δεδομένων σε μια μεταβλητή

Γραμμή 4-6: Λίστα που περιέχει τα δεδομένα και άρα την κρυφή συμβολοσειρά

Το αποτέλεσμα από την εκτέλεση του αλγορίθμου βρίσκεται στην εικόνα 158 και είναι η κρυφή συμβολοσειρά που δόθηκε στην αρχή. Οπότε ο αλγόριθμος με μια μόνο αναζήτηση βρήκε την κρυφή συμβολοσειρά.

['1010']

Εικόνα 158: Κρυφή συμβολοσειρά

Στην συνέχεια η κάθε συμβολοσειρά θα μετατέπεται σε μια μελωδία όπου ο κώδικας που ακολουθεί στέλνει τα δεδομένα της συμβολοσειράς στην sonic pi ώστε να παραχθει η μουσική με βάση τα δεδομένα. Η διαδικασία αυτή αναπαρίσταται στην εικόνα 159

```

1 List = []
2 client = udp_client.SimpleUDPClient(address, 4560)
3 for i , j in enumerate(Data):
4     for k , l in enumerate(j) :
5         List.append(l)
6 print(List)
7 client.send_message("/play_again",List)
8
9

```

Εικόνα 158: Αποστολή δεδομένων σε Sonic Pi

Γραμμή 1-2: Δημιουργία λίστας που θα περιέχει τα δεδομένα, δημιουργία “πελάτη” που θα στείλει τα δεδομένα στην Sonic Pi

Γραμμή 3-5:Εισαγωγή των στοιχείων της συμβολοσειράς σε μια λίστας

Γραμμή 6-7: Εκτύπωση στοιχείων και αποστολή της λίστας στην Sonic Pi

Η λίστα που θα σταλεί στην Sonic Pi αναπαρίστατε στην εικόνα και είναι

Ενώ ο κώδικας που δημιουργείτε μόλις σταλούν τα δεδομένα στην Sonic Pi αναπαρίστατε στις εικόνες 160, 161

```

4 use_bpm 60
5 use_real_time
6 live_loop :Bernstein_Vazirani do
7   use_bpm 60 # Beats per minute
8   data = sync "/osc*/play_again"
9
10  define :music do |element|
11    with_fx :reverb, room: 0.9 do
12      if element == '1'
13        with_synth :hoover do
14          play 60, attack: 1.5, attack_level: 0.7, decay: 1, sustain_level: 0.8, sustain: 1, release: 5
15          sleep 0.5
16        end
17      end
18      if element == '0'
19        with_synth :dark_ambience do
20          play 60, attack: 2, attack_level: 0.7, decay: 0.8, sustain: 2, sustain_level: 1, release: 9
21          sleep 0.5
22        end
23      end
24    end
25  end

```

Εικόνα 160: Παραγωγή μουσικής μέσα από τα δεδομένα του Bernstein- Vazirani

Γραμμή 6: Δημιουργία βασικού βρόγχου επανάληψης

Γραμμή 8: Εισαγωγή δεδομένων από jupyter notebook με βοήθεια udp πρωτοκόλλου

Γραμμή 10: Συνάρτηση που δεχεται τα δεδομένα του Bit Sequence και παράγει την κατάλληλη μουσική

Γραμμή 11: Εισαγωγή fx που θα χρησιμοποιηθεί και θα καθορίσει το ακουστικό αποτέλεσμα

Γραμμή 12-13: Ορισμός ήχου με βάση τον οποίο θα ακούγεται η μουσική, κάθε ήχος διαφέρει ανάλογα με τα δεδομένα

Γραμμή 14: Εκτέλεση μια συγκεκριμένης νότας και χρήση παραμέτρων οι οποίες καθορίζουν την μουσική αναπαράσταση της νότας

Γραμμή 15: Χρόνος που θα χρειαστεί ώστε να πραγματοποιηθεί η επόμενη επανάληψη της μελωδίας.

Γραμμή 18-21: Δημιουργία ήχου όταν τα δεδομένα έχουν τιμή 0, είναι αντιληπτό πως τα δεδομένα που έχουν τιμή 1 θα ακούγονται λιγότερο έντονα και θα έχουν μικρότερη διάρκεια από τα δεδομένα που έχουν τιμή 0 καθώς τα δεδομένα με την τιμή 1 έχουν μικρότερες τιμές στις παραμέτρους τους

```
28 data.each do |d|
29   1.times do
30     music(d)
31     puts "#{d}"
32   end
33 end
34 end
```

Εικόνα 161: Παραγωγή μουσικής μέσα από τα δεδομένα του Bernstein- Vazirani

Γραμμή 28-30: Για κάθε στοιχείο από τα δεδομένα που έχουν σταλεί στην Sonic Pi, γίνεται κλήση της συνάρτησης για αναπαραγωγή μελωδίας

Γραμμή 31: Εμφάνιση του στοιχείο για το οποίο θα εκτελεστεί η μουσική

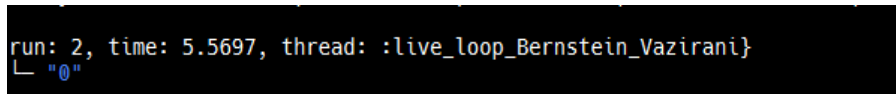
Στις εικόνες 162,163 αναπαρίστανται το αποτέλεσμα της εντολής 31 καθώς και η μουσική που θα ακούγεται κάθε φορά

```
{run: 2, time: 4.0697, thread: :live_loop_Bernstein_Vazirani}
└─ "1"
  synth :dark_ambience, {note: 60.0, attack: 2.0, attack_level: 0.7, decay: 0.8, sustain: 2.0, sustain_level: 1, release: 9.0}

{run: 2, time: 4.5697, thread: :live_loop_Bernstein_Vazirani}
└─ "0"
  synth :hollow, {note: 60.0, attack: 1.5, attack_level: 0.7, decay: 1.0, sustain_level: 0.8, sustain: 1.0, release: 5.0}

{run: 2, time: 5.0697, thread: :live_loop_Bernstein_Vazirani}
└─ "1"
  synth :dark_ambience, {note: 60.0, attack: 2.0, attack_level: 0.7, decay: 0.8, sustain: 2.0, sustain_level: 1, release: 9.0}
```

Εικόνα 162: Εμφάνιση δεδομένων από το Jupyter notebook και μελωδίας που αναπαράγεται



Εικόνα 163: Εμφάνιση δεδομένων από το Jupyter notebook και μελωδίας που αναπαράγεται

6.0 Συμπεράσματα

Κατά την διάρκεια της πτυχιακής υπήρξε μια πλήρη ερέυνα γύρω από τον τομέα της κβαντικής υπολογιστικής και των τρόπων με τους οποίους δύο κλάδοι με όχι και τόσα πολλά κοινά στοιχεία όπως ο κλάδος της μουσικής με τον κλάδο της κβαντικής υπολογιστικής μπορούν εν τέλη να συνδεθούν. Βασικός σκοπός κατά την διάρκεια της έρευνας ήταν τόσο η κατανόηση των εννοιών της κβαντικής υπολογιστικής και των κβαντικών αλγορίθμων αλλά ταυτόχρονα και των τρόπων με τους οποίους η μουσική θα μπορεί να εισαχθεί μέσα σε αυτές τις έννοιες. Στόχος της παρούσας πτυχιακής δεν ήταν η δημιουργία ενός δύσκολου μουσικού συνόλου αλλά η απόδειξη πως μέσα από τους κβαντικούς αλγόριθμους και πως από την σωστή τους διαχείριση μπορεί να παραχθεί ένα αρμονικά ορθό αποτέλεσμα.

Όπως έχει αναφερθεί και πολλές φορές προηγούμενος όλοι οι κλάδοι της κβαντικής υπολογιστικής κβαντική κρυπτογραφία, κβαντικά παιχνίδια κλπ. εξελίσσονται συνεχώς. Στο μέλλον οι κλάδοι όπως της κβαντικής ακουστικής και της κβαντικής μουσικής θα μπορούσαν να συνδεθούν με τους υπόλοιπους κλάδους δημιουργώντας ιδιαίτερα αποτελέσματα. Μερικά παραδείγματα είναι τα εξής η δημιουργία κβαντικού παιχνιδιού του οποίου ο ήχος θα προκύπτει από την παραγωγή μουσικής μέσα από κβαντικούς αλγόριθμους. Ενώ ακόμα οι ερευνητές υποστηρίζουν πως στο μέλλον η κβαντική μουσική θα μπορέσει να χρησιμοποιηθεί στην κβαντική κρυπτογραφία για την κωδικοποίηση μουσικών πληροφοριών ώστε η αποκωδικοποίησή τους να είναι αδύνατη. Επίσης σύμφωνα με τους ερευνητές η κβαντική μουσική θα μπορέσει να χρησιμοποιηθεί για την αποκρυπτογράφηση μουσικών συνθέσεων με σκοπό την προστασία πνευματικών δικαιωμάτων των καλλιτεχνών. Όπου η αποκρυπτογράφηση των μουσικών κομματιών θα πραγματοποιείται με την κβαντική κατανομή κλειδιών (Quantum Key Distribution). Η κβαντική κατανομή κλειδιών αποτελεί έναν ασφαλή τρόπο ανταλλαγής μηνυμάτων.

Αρχικά ο αποστολέας και ο παραλήπτης έχουν ορίσει ως κλειδί ένα bit sequence, ύστερα μέσα από μια σειρά κβαντικών καταστάσεων δημιουργείτε το κρυφό κλειδί και ακολουθεί, μέτρηση της κάθε κβαντικής κατάστασης η οποία καθιστά την υποκλοπή του κβαντικού κλειδιού αδύνατη διότι η μεταβολή σε κάποια κβαντική κατάσταση θα γίνει αντιληπτή. Οπότε όπως έγινε αντιληπτό και παραπάνω η κβαντική μουσική μπορεί να εξελιχθεί ακόμα και να εξελιχθεί και μέσα από άλλους κλάδους.

7.0 Βιβλιογραφία

[1]. Εκδόσεις ΤΣΙΟΛΑ,” ΚΒΑΝΤΙΚΗ ΥΠΟΛΟΓΙΣΤΙΚΗ: Από τη Θεωρία στην Πράξη”

URL:<https://www.tziola.gr/book/kvantiki-ypologis-tiki/#:~:text=%CE%9F%20%CE%97%CE%BB%CE%AF%CE%B1%CF%82%20%CE%A3%CE%AC%CE%B2%CE%B2%CE%B1%CF%82%20%CF%83%CF%80%CE%BF%CF%8D%CE%B4%CE%B1%CF%83%CE%B5%20%CE%9C%CE%B1%CE%B8%CE%B7%CE%BC%CE%B1%CF%84%CE%B9%CE%BA%CE%AC,%CF%84%CE%B1%20%CE%BC%CE%B5%CE%B3%CE%AC%CE%BB%CE%B1%20%CE%B5%CF%84%CE%B5%CF%81%CE%BF%CE%B3%CE%B5%CE%BD%CE%AE%20%CE%9A%CE%B1%CF%84%CE%B1%CE%BD%CE%B5%CE%BC%CE%B7%CE%BC%CE%AD%CE%BD%CE%B1%20%CE%A3%CF%85%CF%83%CF%84%CE%AE%CE%BC%CE%B1%CF%84%CE%B1>

[2].Wikipedia,” Quantum Mechanics”, URL: Quantum Mechanics

[3]. Live Science, “What is quantum mechanics? “, URL : What Is Quantum Mechanics? Quantum Physics Defined, Explained | Live Science

[4].Mac Tutor History of Mathematics , Quotations by Erwin Schrödinger-Mac Tutor History of Mathematics , URL: Quotations by Erwin Schrödinger - MacTutor History of Mathematics (st-andrews.ac.uk)

[5].Electropaedia, “Heisenberg's Matrix Mechanics”, URL: <https://www.mpoweruk.com/figs/Matrix-Mechanics.htm>

[6].Medium, ” A Brief History of Quantum Computing”, URL: Nobel Prize in Physics Awarded for Work on Quantum Entanglement (popularmechanics.com)

[7].Medium, “Quantum cryptography”, URL: <https://medium.com/many-body-physics/quantum-cryptography-d63594917c4a>

[8].Quantum Music, “Quantum music”, URL: <https://quantummusic.org/>

[9]. Medium,” Turning Quantum States into Music at Australia’s First Ever Qiskit Hackathon”, URL: <https://medium.com/qiskit/turning-quantum-states-into-music-at-australias-first-ever-qiskit-hackathon-25da7f09d226>

[10].Wikipedia, ”Quantum game theory”, URL: https://en.wikipedia.org/wiki/Quantum_game_theory

[11].arXiv, ”Quantum Poker”,

URL:https://frib.itch.io/quantum-poker?fbclid=IwAR04aHriXehyo-bTyi7P_eN7YKwRH1rzsgp7gZwn7bsrIZIGxPQQTfFX78Q

[12].Medium,” Top 10 Python Applications in the Real World You Need to Know”

URL: <https://medium.com/edureka/python-applications-18b780d64f3b>

[13] Gamma Solutions, ”What is a Qubit?”, URL: What is a Qubit? gamma-solutions.com

[14]. YouTube, “What Is Sound? The Fundamental Science Behind Sound”, URL: (8)

What is Sound? The Fundamental Science Behind Sound - YouTube

[15]. University of Toronto, “What Is Sound?”, URL: Microsoft PowerPoint - PP02 Spoken Language 1 F04 (toronto.edu)

[16]. Wikipedia,” Musical Tone”, URL: Musical tone - Wikipedia

[17]. YouTube,” Quantum: Music at the Frontier of Science”, URL: (8) Quantum: Music at the Frontier of Science – YouTube

[18].Wikipedia,”Ιάωνης Ξενάκης”, URL: Ιάωνης Ξενάκης - Βικιπαίδεια (wikipedia.org)

[19].Medium,” Python in Music Production: Algorithmic Composition”, URL: <https://medium.com/@alexandragrosu03/python-in-music-production-algorithmic-composition-dfa35470d088>

[20]Audio Cipher, “What is a Musical Cryptogram?”, URL: <https://www.audiocipher.com/post/musical-cryptogram>

[21].Stamford CCRMA, “A Brief History of Algorithmic Composition”, URL: <https://ccrma.stanford.edu/~blackrse/algorithm.html>

[22].Wikipedia, ”Sonic Pi”, URL: https://en.wikipedia.org/wiki/Sonic_Pi

[23].Wikipedia, ”Live Coding”, URL: https://en.wikipedia.org/wiki/Live_coding

[24].Wikipedia,” Open Sound Control”,

URL: https://en.wikipedia.org/wiki/Open_Sound_Control

[25]. Science News, “How Pythagoras turned math into a tool for understanding reality”, URL: <https://www.sciencenews.org/article/pythagoras-math-reality-music-spheres>

[26].Wikipedia, ”Quantum acoustics”, URL: https://en.wikipedia.org/wiki/Quantum_acoustics

[27].Britannica,” Phonon | Quantum Mechanics, Wave-Particle”, URL: <https://www.britannica.com/science/phonon>

[28] arXiv, “Quantum Computer Music: Foundations and Initial Experiments”, URL: <https://arxiv.org/pdf/2110.12408.pdf>

[29] ctm-festival.de, “Quantum Computer Music. What the heck?”, URL: <https://www.ctm-festival.de/magazine/quantum-computer-music-what-the-heck>

[30].Wikipedia, ”Markov chain”, URL: https://en.wikipedia.org/wiki/Markov_chain

[31].Medium,” Introduction to Quantum Walks”, URL: <https://medium.com/mit-6-s089-intro-to-quantum-computing/introduction-to-quantum-walks-b2f741dc5c87>

[32].Medium, ”Behind Oracles: Grover’s Algorithm”, URL: <https://towardsdatascience.com/behind-oracles-grovers-algorithm-amplitude-amplification-46b928b46f1e>

[33].Medium,” Quantum Computing - Grover's Algorithm and Amplitude Amplification”, URL: <https://www.linkedin.com/pulse/quantum-computing-grovers-algorithm-amplitude-arun-serr%C3%A3o/>