Tampereen yliopisto

Heikki Metsäpuro

KVANTTIOHJELMOINTI JA OPETUS

Kvanttiohjelmointi: opetus, opetustarjonta ja ohjelmoija

Kandidaatin tutkielma
Informaatioteknologian ja viestinnän tiedekunta
Joulukuu 2022

TIIVISTELMÄ

Heikki Metsäpuro: Kvanttiohjelmointi ja opetus
Kandidaatin tutkielma
Tampereen yliopisto
Luonnontieteiden kandidaatti
Joulukuu 2022

Tämän tutkielman tavoitteena on selvittää kvanttitietokoneiden ohjelmoinnin opetustarjontaa Suomessa. Tulos on, että opetustarjonta on vielä melko vähäistä. Tutkielmassa esitetään syitä siihen sekä haetaan keinoja lisätä kvanttiohjelmoinnin opetusta ja sen integrointia nykyiseen ohjelmoinnin opetukseen.

Tutkielmassa pohditaan mitä didaktisia ja pedagogisia erityisvaatimuksia kvanttiohjelmoinnin opetus asettaa perinteisen ohjelmoinnin opetukseen verrattuna sekä avataan opetukseen liittyviä haasteita: miten opettajien pedagogiset taidot ja oppijoiden vaihtelevat taustatiedot vaikuttavat opetuksen suunnitteluun ja vaikuttavuuden seurantaan. Vaikka kvanttiohjelmointi jakaakin klassisen ohjelmoinnin opetuksen kanssa samat didaktiiviset ja pedagogiset periaatteet, eroavaisuuksiakin kuitenkin on. Esimerkiksi arkikokemuksen vastaiset kvantti-ilmiöt tarvitsevat lisähuomiota ja tutkielmassa käsitellään näihin eroihin liittyviä opetuskokemuksia.

Suhtautumisessa ja ryhtymisessä kvanttiohjelmointiin on ollut kynnyksenä ajatus, että se on kaikilta osin erittäin vaikeata, eikä ohjelmointia ole osattu erottaa kvanttimekaniikan teoreettisesta ja kvanttitietokoneiden toteutusten vaativuudesta. Kvanttiohjelmoinnin opetukseen ja oppimiseen on kuitenkin tarjolla menetelmiä ja välineitä, jotka mahdollistavat kvanttiohjelmoinnin aloittamisen ja integroimisen perinteiseen opetukseen jopa jo alemmilla oppiasteilla ja erityisesti korkeammilla asteilla. Esimerkkinä näistä tarkastellaan tarkemmin Jupyter-ohjelmointiympäristöä.

Tutkielma sijoittaa kvanttiohjelmoinnin tietotekniikan historialliseen kehykseen sekä pohtii kvanttitietokoneiden nykyisiä ja tulevia käyttötapoja. Kvanttitietokoneiden kehitys on alun hitaan, enimmäkseen teoreettisen pohdinnan, jälkeen nopeutunut tekniikan kehityksen ja uusien merkittävien sovellusalueiden vaatimusten myötä. Kehitystahti tullee lähivuosina edelleen kiihtymään ja kvanttitietokoneet saavuttavat merkittävän aseman tieteellisessä tutkimuksessa ja kaupallisessa käytössä. Tämä kehitys tarvitsee tekijöitä, tutkijoita ja osaajia. Fysikaalisten ilmiöiden tutkimuksen lisäksi tarvitaan myös algoritmiikan osaamista ja tutkimusta ja tähän tarpeeseen vastaamiseen hyvä lähtökohta on kvanttiohjelmoinnin opetus.

Kvanttiohjelmointia käsitellään myös ohjelmoijan näkökulmasta: keskeisiä eroja klassiseen ohjelmointiin nähden, kuten ohjelmointiparadigmat ja ongelmanratkaisutavat. Lisäksi käsitellään ohjelmointiin liittyviä peruselementtejä: kubitit ja niistä muodostettavat piirit sekä esimerkkiohjelma satunnaislukujen generointiin. Tutkielmassa on katsaus kahden toimittajan ohjelmointiympäristöihin, jotka tarjoavat koko kehityspolun ohjelman suunnittelusta, simuloituun ajoon ja kvanttitietokoneella suoritukseen asti, mahdollistaen kokonaisvaltaisen kvanttiohjelmoinnin esittelyn ja oppimiskokemuksen.

Avainsanat: didaktiikka, pedagogiikka, ohjelmointi, kvanttimekaniikka, kvanttiohjelmointi, Qiskit, Microsoft Azure, Jupyter

ALKUSANAT

Tämä kandidaatin tutkielma on laadittu syksyllä 2022 osana Tampereen yliopisto Luonnontieteiden kandidaatin (LuK) tutkintoa.

Tahdon kiittää ohjaajiani Tomi Janhusta ja Anssi Yli-Jyrää mahdollisuudesta mielenkiintoiseen aihevalintaan ja se tarkentamiseen sekä kaikesta saamastani avusta työn aikana.

Haluan myös kiittää perhettäni saamastani kannustavista kommenteista työn aikana.

Tampereella, 30. joulukuuta 2022

Heikki Metsäpuro

SISÄLLYSLUETTELO

1. Johdanto 1
2. Kvanttimekaniikan ja -tietokoneiden tausta 2
2.1 Klassinen laskenta ja tietotekniikka 2
2.2 Kvanttimekaniikka 2
2.3 Kvanttitietokone 3
2.4 Uhka vai mahdollisuus? 3
3. Kvanttiohjelmoinnin opetus 5
3.1 Didaktiikasta 5
3.2 Pedagogiikasta 6
3.3 Korkeakoulutasoinen opetus Suomessa 6
3.4 Opetustarjontaa muualla 7
3.5 Verkkopohjaiset kurssit 7
3.6 Ohjelmointiympäristöt 8
3.7 Entäpä jatkossa? 10
4. Kvanttitietokone ohjelmoijan näkökulmasta 11
4.1 Ohjelmointiparadigmat 11
4.2 Kvanttitietokoneen ohjelmointi ja komponentit 11
4.3 Algoritmiesimerkki: Satunnaislukugeneraattori 14
5. Yhteenveto 18
Lähteet 19

1. JOHDANTO

Mielenkiinto kvanttimekaniikkaan perustuvien kvanttitietokoneiden ohjelmointiin ja niiden mahdollistavien algoritmisten ongelmien ratkaisemisen on ollut jatkuvassa kasvussa. Kvanttitietokoneiden syvällinen ymmärrys vaatii kuitenkin laajaa fysiikan ja matematiikan tietämystä, mikä on vaikeuttanut niiden ohjelmoinnin oppimista. Uudet simulointi- ja ohjelmointiympäristöt tarjoavat erinomaisen pääsyn kvanttitietokoneiden maailmaan.

Tässä tutkielmassa tutkitaan kvanttitietokoneiden ohjelmointiin soveltuvien ohjelmointikielien ja -ympäristöjen soveltuvuutta ohjelmoinnin opiskeluun ja opetukseen sekä opetustarjontaa Suomessa ja muualla maailmassa.

Tutkimusmenetelminä on käytetty hakuja kirjallisuuteen ja artikkelitietokantoihin sekä kaupallisten toimittajien tarjoamien verkkopohjaisien palvelujen verkkosivuille. Kvanttiohjelmoinnin opetuksesta ja etenkin opetuskokemuksista löytyy kuitenkin vielä kovin niukasti tutkimustuloksia.

Luvussa 2 käsitellään kvanttimekaniikan ja kvanttitietokoneiden historian sijoittumista yleisen laskennan ja tietotekniikan historiaan. Luvussa 3 tutkitaan kvanttiohjelmoinnin opetuksen eroja perinteiseen ohjelmoinnin opetukseen myös didaktisesta ja pedagogisesta näkökulmista. Luvussa 4 tarkastellaan kvanttiohjelmointia ohjelmoijan näkökulmasta: kvanttiohjelmoinnin fysikaalista taustaa ja ohjelmoinnin abstrahointia ohjelmointiympäristöissä ja -kielissä. Luvussa 5 on tutkielman analyysi ja yhteenveto.

Terminologiasta

Kvanttiohjelmoinnista (*Quantum computing, QC*) käytetään usein suomenkielistä termiä kvanttilaskenta; tässä tutkielmassa on käytetty ensin mainittua, koska se korostaa ohjelmoijan roolia ja jälkimmäinen keskittyy tekniseen prosessiin. Perinteisestä ohjelmoinnista, siis kvanttiohjelmointia edeltävästä ohjelmoinnista on käytetty myös termiä ”klassinen ohjelmointi” ja vastaavasti perinteisistä tietokoneista termiä ”klassinen tietokone”.

2. KVANTTIMEKANIIKAN JA -TIETOKONEIDEN TAUSTA

Kvanttitietokonetekniikan ja klassisen tietotekniikan kehityspoluissa on yhteneväisyyksiä: edeltävästä teoreettisesta ideasta ja laskennallisen tarpeen synteesistä syntyy toteutus, joka hitaan alkuvaiheen jälkeen lähtee kehittymään kiihtyvällä vauhdilla. Kvanttitietokoneiden käytännön sovellukset ovat vasta ottamassa ensi askeliaan ja – kuten usein tekniikan kehityshistoriassa – ensimmäisistä ajatusrakennelmista ei voi tietää, mihin niihin perustuva kehitys johtaa.

2.1 Klassinen laskenta ja tietotekniikka

Ohjelmoidun laskennan historian katsotaan alkaneen Charles Babbagen suunnitelmista rakentaa analyyttinen kone, joka kykenisi suorittamaan laskutoimituksia ennalta määrättyjen vaiheiden mukaisesti ja tallettamaan laskennan välituloksia. Augusta Ada Byron (*Lady Lovelace*) luonnosteli ajatuksia ohjelmista Babbagen kanssa käymässään kirjeenvaihdossa. Babbagen analyyttinen kone ei koskaan edennyt keskeneräistä protyyppiastetta pidemmälle ja muutamia aikalaiskokeiluja lukuun ottamatta hänen ideansa unohtui. Laskennan kehitys jatkui mekaanisten, aritmeettisia ja differentiaalilaskutoimituksia tekevien koneiden kehittämisellä. Ennen toista maailmansotaa ja sen aikana tarve suorittaa suuria määriä toistuvia laskutoimituksia kasvoi niin suureksi, ettei mekaanisilla laskukoneilla niitä voinut enää tehdä. Ratkaisuksi kehitettiin Saksassa Zusen sähkömekaaniset koneet (Bauer & Wössner, 1972), Englannissa Turingin Bombe (Plimmer, 1998) ja Yhdysvalloissa ENIAC (Haigh ym., 2014). ENIACin arkkitehtuuri on pysynyt tietokoneiden suunnittelun pohjana nykyaikaan asti. ENIACin tekninen toteutus perustui tyhjiöputkiin ja sen jälkeen, kun ne korvattiin transistoreilla ja myöhemmin mikropiireillä, tietokoneiden suorituskyvyn ja samalla koko tietojenkäsittelyalan kehitys on ollut nopeata.

2.2 Kvanttimekaniikka

1900-luvun alussa kävi ilmeiseksi, ettei Newtonilainen klassinen fysiikka kyennyt selittämään sen paremmin kosmisen mittaluokan kuin atomitasonkaan ilmiöitä. Albert Einstein loi yleisen suhteellisuusteorian, joka toimi hyvin kosmologisen tason ilmiöiden selittäjänä,

mutta ei kuitenkaan selittänyt ilmiöitä atomitasolla saati atomia pienempien hiukkasten tasolla. Tätä teoreettista aukkoa alettiin täyttää kehittämällä uusi teoria, **kvanttimekaniikka**. Sen kehittämiseen osallistui joukko teoreettisen fysiikan tutkijoita, muun muassa Niels Bohr, Max Born, Louis de Broglie, Paul Dirac, Werner Heisenberg, Wolfgang Pauli, Max Planck, Erwin Schrödinger ja myös Albert Einstein (Jones, 2008). Kvanttimekaanisten ilmiöiden olemassaolosta käytiin pitkällinen ja kiivaskin keskustelu mm. Einsteinin ja Bohrin välillä. Kvanttimekaanisten ilmiöiden olemassaolo kuitenkin on todistettu myöhemmin useissa fysikaalisissa kokeissa ja havaintoja on tarkennettu aivan viime aikoihin asti. Kvanttimekaniikkaan perustavia ilmiöitä on hyödynnetty jo pitkään mm. puolijohteissa ja lasersäteisiin pohjautuvissa laitteissa ja nykyaikainen elektroniikkaan pohjautuva elämäntapamme tuskin olisi olemassa ilman tähän teoriaan perustuvia teknologioita.

2.3 Kvanttitietokone

Kvanttitietokoneiden toiminta perustuu kvanttimekaanisiin ilmiöihin. Niiden mahdollisuutta spekuloitiin 1950-luvun lopulta alkaen, ja eritysesti Richard Feynmann kehitti niiden teoreettisesta perustaa. Feynmann päätteli, että mikroelektroniikan jatkuvasti pienentyessä eteen tulevat kvanttimekaaniset ilmiöt ja pohti näiden ilmiöiden hyödyntämistä uudenlaisen tietokonesukupolven kehittämiseksi. Kvanttitietokoneen kehityksen merkkipaaluna pidetään David Deutschin vuonna 1985 julkaisemaa artikkelia (Phillips, 2013). Kvanttitietokoneiden tekninen kehitys on kiihtynyt ja algoritminen tutkimustyö on avannut näkymiä uusille sovellusalueille. Algoritmien kehityksen merkittäviä askeleita olivat Peter Shoren kokonaislukujen tekijöintijakoalgoritmi (Mousavi ym., 2021) ja Lov Groverin hakualgoritmit (Patel, 2021). Algoritmeja siis kehitettiin jo ennen kuin koneita oli vielä olemassa, mielenkiintoinen yksityiskohta on se, että tietojenkäsittelyn historiasta löytyy samankaltainen asetelma Babbagen analyyttisen koneen ja Ada Lovelacen ohjelmien suhteen.

2.4 Uhka vai mahdollisuus?

"Tuli on hyvä renki, mutta huono isäntä."

Kaikkiin teknologisiin edistysaskeliin on aina liittynyt eettisiä kysymyksiä niiden käyttämisestä hyvään tai pahaan, haitaksi tai hyödyksi eikä hyvä-paha-asetelman puolistakaan useimmiten ole ollut yksimielisyyttä. Laskenta- ja tietotekniikka ei tee tässä suhteessa poikkeusta. Laskentatehon kasvaminen ei juurikaan herättänyt huolia – pois lukien sotilaallisten sovellusten mukanaan tuomat uhat – ennen kuin vasta internetin kehityksen myötä, ja nimenomaan käyttäjien yksityisyyden suojaan liittyen. Kvanttitietokoneen alkuajan filosofisten kiistojen ja epäilyjen jälkeen viime aikoina on herännyt toiveita uusista tieteellisistä ja kaupallisista sovellusaloista, mutta uhkakuviakin on noussut esiin: esimerkkeinä suunnattomasti lisääntynyt laskentateho mahdollistaa nykyisten salaustekniikoiden

murtamisen, epätasa-arvoisuuden lisääntyminen teknologisien kyvykkyyksien keskittyessä sekä poliittisesti että maantieteellisesti. Tekoälyn (*Artificial Intelligence, AI*) ja koneoppimisen (*Machine Learning, ML*) kehitys on syventänyt huolia yksityisyyden suojasta ja herättänyt filosofisia pohdintoja teknologioiden eettisistä rajoista.

3. KVANTTIOHJELMOINNIN OPETUS

Tässä luvussa tarkastellaan kvanttiohjelmoinnin opetusta didaktisesta ja pedagogisesta näkökulmista. Luvussa tehdään myös katsaus nykyiseen opetustarjontaan, toisaalta Suomessa ja ulkomailla sekä esittellään verkkopohjaista opetustarjontaa. Luvun loppupuolella esittellään kaksi ohjelmointiympäristöä, *Azure Quantum* sekä *IQM Qiskit*, sekä tarkastellaan tarkemmin *Jupyter* -ohjelmointialustaa.

3.1 Didaktiikasta

Suomessa ohjelmoinnin opetus on osa alempien kouluasteiden opetussuunnitelmia ja opettajankoulutustakin, ja niiden soveltamisesta on tutkimuksia, esimerkiksi Janne Fagerlundin väitöskirja (Fagerlund ym., 2021), jossa tutkitaan ohjelmoinnillisen ajattelun opettamista. Korkeakouluissa opetussuunnitelmatyö on alempia kouluasteita vapaampaa, se määrittelee opintojen rakenteen ja laajuuden, eikä se puutu varsinaisiin oppimistavoitteisiin. Didaktisesti keski- ja korkea-asteiden tietotekniikan opetus ei toisaalta eroa paljon alempien asteiden opetuksesta. Selkeä ero korkeakoulutasoisessa opetuksessa on ensiksi se, että opettajat ovat usein alojensa asiatuntijoita, joilta puuttuu pedagogista koulutusta. Toiseksi oppijoiden lähtötasot ja ennakkotiedot (etenkin peruskursseilla) vaihtelevat suuresti ja ennakkotiedoissa saattaa olla ratkaisevia puutteita ja jopa väärinkäsityksiä, jotka pitää ottaa huomioon opetusta suunniteltaessa. Ongelmaa käsitellään laajasti Sue Sentancen kokoamassa teoksessa (Sentance ym., 2018) ja siinä tarjotaan ratkaisuksi esimerkiksi oppijoiden ja opetuksen jatkuvaa evaluointia.

Erityisesti kvanttiohjelmoinnin didaktiikkaa käsittelevistä tutkimuksista löysin tietoa niukasti. Seuraavissa artikkeleissa on aihetta avattu ja esitetty.

Tutkimusartikkelissa (Seegerer ym., 2021) todetaan, että tärkeä lähestymistapa kvanttiohjelmoinnin opetuksessa on keskeisten käsitteiden esittely. Kirjallisuuskatsausosassa todetaan, että kirjallisuudessa tutkituissa teoksissa usein toistuvat samat keskeiset käsitteet: superpositio, lomittuminen, kvanttitietokone, kvanttialgoritmit ja kvanttikryptografia. Artikkelin haastatteluosassa haastatellut asiantuntijat asettivat tärkeimmäksi käsitteeksi kubitin, muilta osin tulos vastasi kirjallisuuskatsauksen tulosta.

3.2 Pedagogiikasta

Ohjelmointia on opetettu pitkään monilla erilaisilla tavoilla: kirjallisilla oppimateriaaleilla, luokkaopetuksella, tietokonepohjaisilla harjoituksilla, verkkopohjaisilla järjestelmillä ja näiden yhdistelmillä. Opetukseksi luen myös työssä tapahtuvan oppimisen, jota alalla käytetään etenkin silloin, kun alan nopean kehityksen takia institutionaalista opetusta ei ole ollut saatavilla.

Kvanttiohjelmoinnin opetus ei mielestäni eroa perinteisen ohjelmoinnin opetuksesta, pelkistetysti: opetetaan teoreettisen tiedon ja peruskäsitteiden soveltamista käytäntöön. Laiteläheisen kvanttiohjelmoinnin edellytykset ovat olleet olemassa jo pitkään, mutta kuten edellä totesin, se on teknisesti vaativaa. Ohjelmoinnin oppimiseen ja opetukseen on tullut käyttöön ohjelmistokieliä ja -ympäristöjä, jotka alentavat merkittävästi kynnystä ohjelmoinnin aloittamiseen.

3.3 Korkeakoulutasoinen opetus Suomessa

Kurssien rakenne ja laajuus on kaikissa luetelluissa saman kaltainen: peruskäsitteiden oppiminen, kubitin ominaisuudet, piirien rakentaminen, simulointi ja algoritmien toteutus. Osa kursseista on kvanttimekaniikan yleistä, fysiikan perusteiden opetusta, mutta itsenäisiä, nimenomaan ohjelmointiin keskittyviäkin kursseja on tarjolla.

Suomessa kvanttitietokoneiden ohjelmoinnin instituutionaalista opetusta tarjotaan toistaiseksi vähän, säännöllistä opetusta on kuitenkin jo käynnissä. Seuraavana on kaksi poimintoa Suomalaisissa korkeakouluisissa tarjottavista kvanttiohjelmoinnin kursseista.

Aalto yliopisto, Practical Quantum Computing (CS-C3260) (”Aalto yliopisto”, 2022)

Aalto-yliopiston kurssi ”Practical Quantum Computing” on 5 opintopisteen laajuinen, aineopintoihin kuuluva kurssi, joka koostuu luennoista ja harjoituksista (6 * 2h). Kurssin esivaatimuuksiin kuuluu suoritettu lineaarialgebran kurssi ja kurssilla käsiteltäviä aiheita ovat muun muassa: kvanttitietokoneiden liittyvät peruskäsitteet, kvanttiohjelmointi ja kvanttiherruus (*quantum supremacy*), nykyisten kvanttitietokoneiden (*Noisy Intermediate-Scale Quantum, NISQ*) laskennalliset rajat, kvanttitietokoneiden käyttökohteet, aritmetiikan ohjelmointi ja virheenkorjausmenetelmät (*Surface code, braiding, Lattice surgery*). Kurssi järjestetään seuraavan kerran keväällä 2023.

Oulun yliopisto, Quantum information (763635S) (”Oulun yliopisto”, 2022)

Sisällöltään Aalto-yliopistokurssia vastaava kurssi järjestetään Oulun yliopistossa, ”Quantum information”. Kurssi on 5 opintopisteen laajuinen, syventäviin opintoihin kuuluva kurssi. Kurssin esitietovaatimuksiin kuuluu kvanttimekaniikan kurssit (Quantum Mechanics I

ja II). Kurssiin kuuluu luentoja 26h, harjoituksia 14h, ryhmätöitä 10h ja itseopiskelua 85h. Kurssilla käsitellään kvanttitietokoneiden peruskäsitteitä, kvanttiohjelmointia, -kryptografiaa, ja -kommunikaatiota (*quantum communication*). Kurssilla käsitellään myös virheenkorjausmenetelmiä. Kurssi järjestetään seuraavan kerran keväällä 2023.

3.4 Opetustarjontaa muualla

Kvanttitietokoneiden ja niiden ohjelmoinnin opetuksella alkaa olla jo vakiintunut asema etenkin suurten oppilaitosten opetusohjelmissa, seuraavassa niistä kaksi esimerkkiä.

Oxford Universityn kurssi **(”Quantum information”, 2022)** on algoritmipainotteinen kurssi. Kurssilla käsiteltäviä aiheita: kvanttitietokoneiden peruskäsitteet, kubittien mittaaminen, laskennallisten mallien toteuttaminen, Fouriermuunnokset, kvanttikommunikointi ja keskeiset algoritmit, kuten *Deutsch-Jozsa, Grover ja Shor*. Kurssi koostuu 16 luentokerrasta ja järjestetään lukukaudella 2022–2023; kurssi on osasuorituksena lukuisissa tietojenkäsittelytieteiden tutkinnoissa.

Göteborgin yliopiston kurssi **(”Quantum Computing”, 2022)** on osa fysiikan maisterin tutkintoa ja esivaatimuksena on soveltuva kandidaatin tutkinto. Kurssin laajuus on 7,5 opintopistettä ja koostuu luennoista, harjoituksista ja labororatoriatöistä sekä lopputentistä. Kurssi on teoreettispainoitteinen, mutta kurssilla käsitellään aluksi kvanttitietokoneiden ja -laskennan perusteet. Teoreettinen osuus kattaa keskeiset teoriat kvanttietokoneiden toimintaperiaateista ja käytännön läheisessä osuudessa syvennytään algoritmiikkaan, esimerkiksi Fourier-muunnokset ja koneoppiminen.

3.5 Verkkopohjaiset kurssit

Verkkopohjaisissa opetusympäristöissä on tarjolla runsaasti monen tasoisia kursseja ja kurssien valikoima laajenee nopeasti. Verkkopohjaisia kursseja on tarjolla sekä ilmaisia että maksullisia, erillisiä osia ja kokonaisiin tutkintoihin tähtääviä kokonaisuuksia. Useimmiten kurssit koostuvat video-oppitunneista ja ohjelmointialustoilla suoritettavista tehtävistä. Taulukossa 3.1 luetellaan esimerkkejä verkkokursseista ja niiden käyttämistä opetusmenetelmistä.

Tarjoaja	Kurssi	Laajuus	Sisältö
Coursera	Introduction to Quantum Information	11+ h	Kvanttiohjelmointi portit ja piirit
edX	Introduction to Quantum Science & Technology	6 viikkoa	Kvanttiohjelmointi portit ja piirit virheen käsittely simulaatiot koneoppiminen
MIT XPRO	Quantum Computing Fundamentals	8 viikkoa	Kvanttiohjelmointi portit ja piirit virheen käsittely simulaatiot koneoppiminen

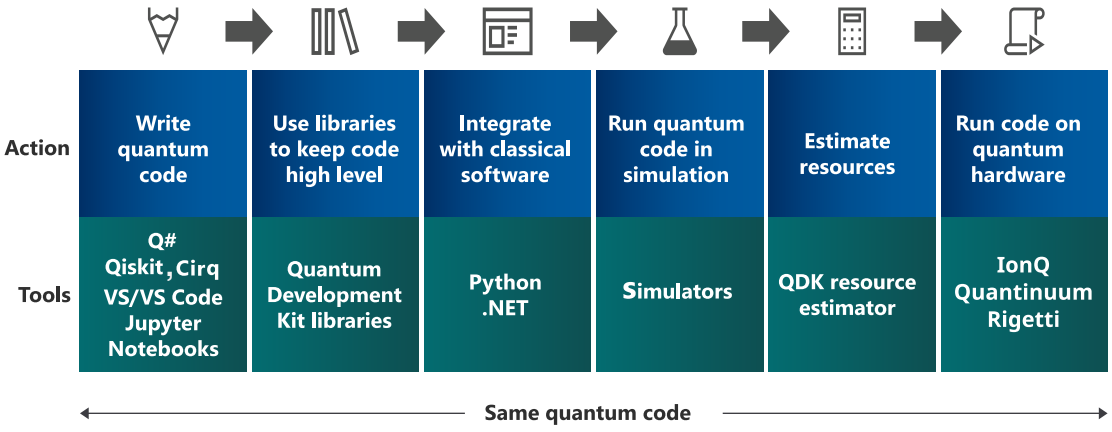
Taulukko 3.1. *Verkkopohjaisia kursseja*

Kurssien maksullisista versioista on useimmiten tarjolla ilmainen versio, josta ei saa todistusta. Kurssien laajuus, sisältö ja laatu vaihtelevat paljon, ja suosittelen aina ensin tutustumaan kurssin ilmaiseen versioon.

3.6 Ohjelmointiympäristöt

Tyypillinen työnkulku

Kuvassa 3.1 on kuvattu kvanttiohjelmoinnin työnkulku yleisellä tasolla, tässä Azure Quantum ympäristössä, muissa ympäristöissä työskentely noudattaa pitkälti samaa mallia. Aluksi algoritmi kirjoitetaan valitussa ohjelmointiympäristössä ja -kielellä, seuraavaksi suoritetaan simulaationa ja lopuksi saatetaan konkreettisen kvanttitietokoneen suoritusjonoon. Ajon tulos saadaan simulaatiosta välittömästi, mutta etäajon tulosta voi joutua odottamaan jopa useita tunteja.



Kuva 3.1. *Ohjelmoinnin työkulku ("Azure Quantum Workflow", 2022)*

Taulukko 3.2. *Ohjelmointiympäristöt*

Tarjoaja	IDE	Kielet	Hinta
Azure Quantum	Jupyter Visual Studio Code Visual Studio	Python, Q#, OpenQASM	Alkaen 0€
IBM Qiskit	Jupyter	Python, OpenQASM	0€
Amazon Braket	Amazon SageMaker	Python	Alkaen 0€

Tarjolla on lukuisia vaihtoehtoja, taulukossa 3.2 luetellaan yleisesti tunnettuja vaihtoehtoja: Azure Quantum, IBM Qiskit ja Jupyter.

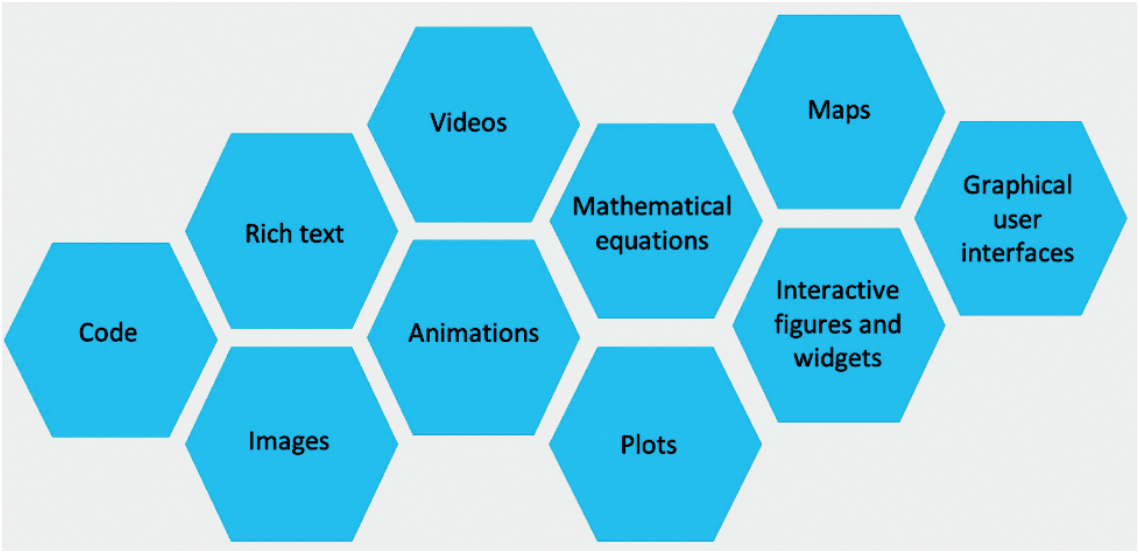
Microsoftin **Azure Quantum** (”Azure Quantum”, 2022) on järjestelmä, jossa ohjelmia voi kirjoittaa useilla ohjelmointikielillä, käyttäen useita graafisia ohjelmointityökaluja. Ohjelmia voi suorittaa simuloituna joko paikallisesti tai eräajona kvanttitietokoneella. Erityisesti *Azure Portal*issa ohjelmia voi kirjoittaa Jupyter -alustalla, joka tarjoaa monipuoliset käyttömahdollisuudet. Azure Quantum vaatii Microsoft Azure tunnuksen ja sen käyttö on maksullista, joskin Microsoft tarjoaa opiskelijoille ilmaisen, rajoitetun vaihtoehdon.

IBM:n **Qiskit** (”IBM Qiskit Toolkit”, 2022) on avoimen koodin järjestelmä, joka tarjoaa kattavan työkalupaketin kvanttiohjelmoin oppimiseen ja opettamiseen. Qiskitin käyttö on ilmaista, mutta vaatii käyttäjätunnuksen luomisen (*IBMId*) palveluun. Qiskit hyödyntää laajasti ohjelmoinnissa Jupyter-alustalla Python ja OpenQASM -ohjelmointikieliä. Qiskitissä on dokumentaatio, joka käsittelee laajasti kvanttiohjelmointia yleisellä tasolla sekä opastaa perusteellisesti ympäristön käyttöön. Kyseisen dokumentaation lukemista voi suositella jokaiselle aiheesta kiinnostuneelle, sekä pedagogisena inspiraationa että tiedonlähteenä. Wootton (Wootton ym., 2021) kuvaa artikkelissaan Qiskitin käyttöä kvanttiohjelmoinnin opetuksessa ja toteaa, että Qiskit tukee erinomaisesti erilaisia oppimispolkuja. Temporão (Temporão ym., 2022) puolestaan kuvaa hyviä tuloksia Qiskitin käytöstä osana monimuoto-opetusta.

Kumpikin edellä kuvattu ympäristo tukee ohjelmointia **Jupyter** -alustalla, joka on avoimen koodin ilmainen tuote. Jupyterissä työskentely tapahtuu *työkirjoissa (workbook)* ja sen *soluissa (cell)*, joihin kirjoitetaan tekstiä, esimerkiksi matemaattista tekstiä (mm. LaTex-muoto), ohjelmakoodia tai vaikkapa upotetaan kuvia. Jupyterin *ajoympäristöt (kernel)* tukevat lukuisa ohjelmointikieliä laitteistoläheisistä kielistä Javaan, ja kvanttiohjelmointiinkin on olemassa lukuisia vaihtoehtoja.

Jupyterin vahvuus on sen tarjoama esitysmuotojen ja -tapojen monipuolisuus, mikä tekee siitä erinomaisen työskentely-ympäristön. Kuvassa 3.2 on kuvattu Jupyterin käyttötapoja: esimerkiksi ohjelmointi, matematiikka, opetusmateriaalin luonti, dokumentointi ja havaintomateriaalin luonti.

Toinen Jupyterin vahvuus on sen mahdollistamat monet pedagogiset ratkaisut: etäopis-



Kuva 3.2. *Jupyterin käyttötapoja (Rossant, 2018)*

kelu, omatoiminen opiskelu, ohjattu opettaminen ja luokkamuotoinen opetus. Lupaava ja mielenkiintoinen Jupyterin käyttökohde on *kokemuksellinen oppiminen (experiential learning)*; artikkelissa *Notes on Using Google Colaboratory in AI Education* (Nelson & Hoover, 2020) kuvataan Jupyterin käyttöä tekoälykursseissa.

Jupyter on alustana monissa verkkopohjaisissa opetusalustoissa ja sen käytöstä luokkaopetuksessa ja monimuoto-opetuksessa on kertynyt jo kokemusperäistä tietoa. Artikkelissa Al-Gahmi (Al-Gahmi ym., 2022) raportoidaan hyvistä kokemuksista Jupyterin käytöstä heidän järjestämillään tietotekniikan kursseilla.

3.7 Entäpä jatkossa?

Kvanttiohjelmoinnin algoritminen lähestymistapa eroaa perinteisestä ohjelmoinnista. Kvanttitietokoneiden ja algoritmiikan edelleen kehittyessä syntyy uusia ohjelmointiympäristöjä ja -menetelmiä, ehkä jopa uusia ohjelmointikieliäkin. Kuitenkin tämä kehitys avaa uusia mahdollisuuksia myös perinteiseen ohjelmoinnin opetukseen: eri lähestymistapoja voidaan integroida ja näin hyödyntää ja kehittää kumpaakin.

4. KVANTTITIETOKONE OHJELMOIJAN NÄKÖKULMASTA

Tässä luvussa esittellään kvanttiohjelmoinnin yhtymäkohtia ja eroja perinteiseen ohjelmointiin sekä kuvataan satunnaislukuja generoiva esimerkkiohjelma.

4.1 Ohjelmointiparadigmat

Perinteisessä ohjelmoinnissa abstrahoidaan - useimmiten von Neumannin arkkitehtuurilla toteutettujen, puolijohdetekniikkaan perustuvien tietokoneiden fyysinen toteutus. Abstraktion ansiosta ohjelmoijan ei tarvitse hallita puolijohdetekniikkaa tai olla edes siitä tietoinen. Abstraktio toteutetaan eritasoisilla ohjelmointikielillä, alkaen laitteistoläheisistä, päätyen loogisia rakenteita kuvaaviin kieliin. Nämä ohjelmointikielet noudattavat useita ohjelmointiparadigmoja (muun muassa imperatiivinen, sekventiaalinen, funktionaalinen) ja niihin liittyy keskeisesti monipuoliset kontrollirakenteet (muun muassa toisto, sekventiaalisuus, ehdollisuus, enkapsulointi ja rekursiivisuus) ja muistivaraiset tietorakenteet, joiden avulla saavutetaan haluttu tulos tallennettavaksi tai tulostettavaksi käyttäjälle.

Kvanttiohjelmoinnilla luodaan vastaavasti abstraktiotasoja fyysisen laitetoteutuksen ja ohjelmoijan väliin. Kvanttiohjelmointiin vakiintuneita paradigmoja ei ole vielä perinteisen ohjelmoinnin tapaan muodostunut, johtuen alan nuoruudesta. Luokittelutyötä on kuitenkin jo tehty (Sánchez & Alonso, 2021). Kvanttiohjelmoinnista puuttuu monia perinteisen ohjelmoinnin piirteitä, esimerkiksi toistot ja rekursio. Kvanttiohjelmat ovat luonteeltaan sekventiaalisia ja muistuttavat usein optimointiohjelmointia, mutta joitakin funktionaalisiakin kieliäkin on kehitetty. Kvanttiohjelmointia voidaankin pitää myös omana ohjelmointiparadigmanaan.

4.2 Kvanttitietokoneen ohjelmointi ja komponentit

Kuten edellä on todettu, kvanttitietokoneen laitteistoläheinen ohjelmointi on vaikeata. Seuraavassa kuvaankin niiden ohjelmoinnissa käytettäviä kohteita, kieliä ja ohjelmoinnin välineitä. Kvanttitietokoneen ohjelmoinnissa manipuloidaan kubitin tiloja eli *amplitudia ja vaihetta* sekä lomittuneiden kubittien vuorovaikusta käyttäen valitun ohjelmointikielen komentoja.

Kubitti

Klassisen ohjelmoinnin peruskomponentti on *bitti*, joka saa diskreetin arvon 0 tai 1 ja näitä arvoja edustaa esimerkiksi jännitetaso. Kvanttiohjelmoinnin peruskomponentti on puolestaan *kubitti*. Kubitin fyysinen ilmentymä on tyypillisesti lähelle absoluuttista nollapistettä jäähdytetty heliumatomi, mutta lukuisia muitakin tapoja käytetään ja uusiakin kehitetään jatkuvasti (Dahm ym., 2003). Ennen havainnointia kubitti on *superpositiossa* ja sen tilaa kuvaavat fysikaaliset suureet *amplitudi* ja *vaihe*. Kubitin tilaa manipuloidaan käyttäen monenlaisia fysikaalisia menetelmiä, esimerkiksi magneettikenttiä ja lasersäteitä. Manipulaatioiden jälkeen kubitille voidaan *mitata* klassinen arvo, joka riippuu manipulaatioiden vaikutuksesta syntyneestä tilasta ja arvo sijoitetaan klassiseen rekisteriin. Mittauksen jälkeen kubitti ei ole enää superpositiotilassa. Johtuen kvanttimekaniikan ilmiöiden stokastisesta luonteesta mitattu arvo vaihtelee mittaustilanteesta toiseen, vaikka kubitin manipulaatio toistetaan täsmälleen samalla tavalla. Tästä vaihtelusta johtuva mittauksen epävarmuutta kutsutaan *virheeksi*.

Kubitin superpositiota voi kuvata kolikon heitolla: ilmassa pyöriessään sillä ei ole klassista arvoa - kruuna tai klaava - ja se saa arvon vasta pinnalle asettuessaan.

Useamman kubitin järjestelmässä kubitit ovat vuorovaikutuksessa keskenään, lomittuneena. Samoin kuin yksittäisiin kubitteihin, voidaan kubittien lomittumiseen vaikuttaa erilaisilla fysikaalisilla voimilla. Lomittuminen puolestaan vaikuttaa yksittäisten kubittien käyttäytymiseen ja sitä kautta niiden mittaamiseen. Operaatioiden ja kubittien määrän kasvaessa virheen määrä kuitenkin kasvaa nopeasti, jopa eksponentiaalisesti.

Käyttäen kolikkovertausta, useamman kubitin interaktiota voidaan kuvata useamman, toisiinsa vaikuttavan (esimerkiksi magneeteilla) kolikon heittämisellä, mittaustulos saadaan, kun kolikot asettuvat pinnalle.

Siinä missä perinteisen bitin kuvaamiseen riittää binäärinen arvo 0 ja 1, mutta kubitin amplitudin ja vaiheen kuvaamiseen tarvitaan erilaisia matemaattisia kuvaustapoja (Kuva 4.1), joiden avulla laskennallisesti mallinnetaan kubittien käyttäytymistä:

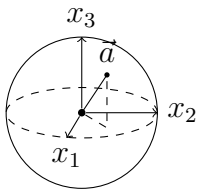
Pystyvektori: $Q = \begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$

Kompleksiluku: $Q = 0 + \frac{1}{\sqrt{2}}i$

BraKet-muoto: $Q = |0\rangle + \frac{1}{\sqrt{2}}|1\rangle$

Kuva 4.1. *Kubitin kuvaustapoja*

Lisäksi kubitin tilaa kuvaamaan käytetään graafista kuvaajaa (Kuva 4.2), **Blochin pallokaaviota**, jossa kubitin vaihe ja amplitudi kuvataan kolmiuloitteisessa avaruudessa:



Kuva 4.2. *Blochin ympyräkaavio*

Yhden kubitin portit

Kuten edellä on todettu, kubittien tilaa voidaan muokata useilla tavoilla. Tavallisimpia kubitin tilaa muokkaavia toimenpiteitä ovat *Hadamard, X, Y ja Z*, joista seuraavassa esitetään vektorimuotoisena sekä vastaava toimenpide Python -kielisenä ohjelmakoodina.

Hadamard, H

Hadamard-portti asettaa kubitin superpositioon, jonka jälkeen kubitin mittaus antaa tulokseksi yhtä suurella todennäköisyydellä 0 tai 1.

Vektorimuoto: $H(Q) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} Q$

Python:

```
qc_hadamard = QuantumCircuit(1,1)
qc_hadamard.h(0)
```

X-portti (X-Gate, X)

X-portti toimii samaan tapaan kuin perinteinen NOT-portti: jos kubitin tilan mittaus (siis jos näin voitaisiin ennustaa) antaisi tulokseksi 0, niin X-portin vaikutuksen jälkeen mittaustuloksesi tulee todennäköisesti 1.

Vektorimuoto: $X(Q) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} Q$

Python:

```
qc_x = QuantumCircuit(1,1)
qc_x.h(0)
```

Y ja Z portit

Y ja Z -portit muuttavat kubitin vaihetta, eivät amplitudia, toisin sanoen ne eivät vaikuta mittaustulokseen mittaushetkellä.

Vektorimuodot ovat: $Y(Q) = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} Q \quad ja \; Z(Q) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} Q$

Python:

```
qc_yz = QuantumCircuit(2,2)
qc_yz.y(0)
qc_yz.z(1)
```

Kahden tai useamman portin piirit, lomittuminen

Kubitit voivat olla myös yhdessä, lomittuneena, jolloin ne ovat vuorovaikutuksessa keskenään. Kuten edellä yhteen kubittiin kohdistuvaa operaatiota voidaan kuvata algebrallisesti, lomittuminenkin voidaan esittää Python -ohjelmalla. Seuraavassa esimerkki yleisesti käytetystä lomittuneesta piiristä, *CNOT*.

CNOT

CNOT-piiri on kahden kubitin muodostama piiri, jossa kubitit ovat lomittuneena; jos ensimmäisen kubitin tila on 1, toisen kubitin tilaan tehdään X-operaatio.

Vektorimuoto: $CNOT(Q) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} Q$

Python:

```
qc_cnot = QuantumCircuit(2,2)
qc_cnot.cx(0,1)
```

4.3 Algoritmiesimerkki: Satunnaislukugeneraattori

Luotettavasti satunnaisten lukusarjojen generointi on ollut yksi kaikkein vaikeimmista ohjelmointitehtävistä. Luonnossa on tiedetty esiintyvän satunnaisia ilmiötä, mutta niiden hyödyntäminen satunnaislukujen generoinnissa on ollut vaikeata. Satunnaisluvut ja -sarjat ovat kuitenkin olennainen osa monissa algoritmeissa ja niiden vahvistamisessa, toisaalta puutteellinen satunnaislukujen generointi on johtanut salausalgoritmien ja salattujen tietojen murtamiseen. Kvanttitietokone antaa luonnonlakeihin perustuvan tavan generoida

täysin satunnaisia lukuja ja lukusarjoja, käyttäen superpositiossa olevien kubittien mittaamista.

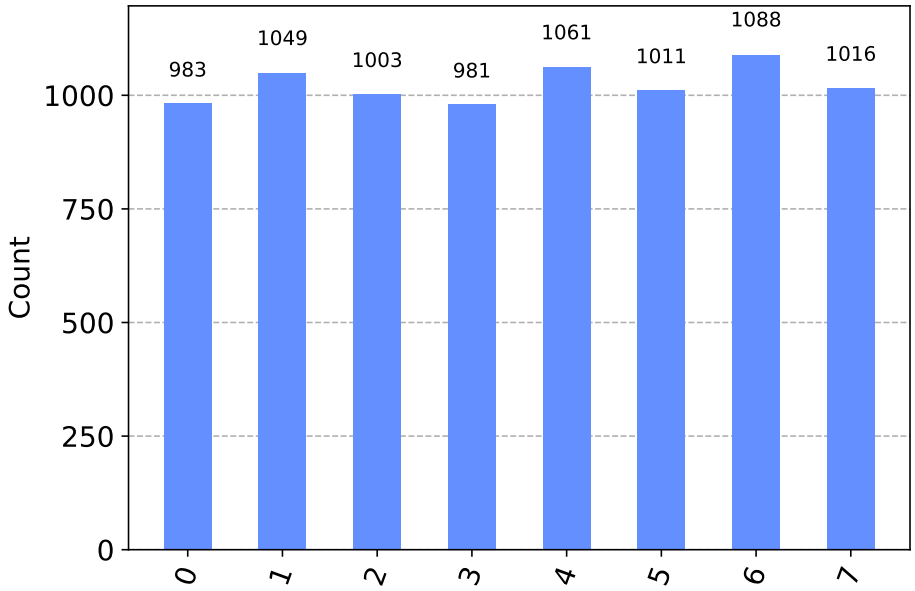
Ohjelmalistauksessa *Ohjelma 4.1* on toteutettu Python-ohjelma, joka generoi 3 bitin satunnaisen luvun. Ohjelma käyttää kolmea kubittia, asettaa ne superpositioon ja lopuksi lukee niiden arvot klassisiin rekistereihin.

Ohjelma 4.1. *Satunnaisluvun generointi*

```
# (1) Ladataan tarvittavat kirjastot
from qiskit import BasicAer, execute
from qiskit import QuantumRegister, \\
ClassicalRegister, QuantumCircuit
from qiskit.tools.visualization import plot_histogram

# (2) Varataan resurssit 3 kubitille (q)
# ja klassiselle rekisterille (c)
n = 3
q = QuantumRegister(n)
c = ClassicalRegister(n)

# (3) Alustetaan piiri
circuit = QuantumCircuit(q, c)

# (4) Asetetaan kubitit superpositioon
for j in range(n):
  circuit.h(q[j])

# (5) Mitataan kubitit klassisiin rekistereihin
circuit.measure(q,c)

# (6) Suoritetaan piiri simulaattorissa
# 8192 ajossa
job = execute(circuit, BasicAer.get_backend(\\
  simulator), shots=8192)

# (7) Haetaan simulaation tulokset
bit_counts = job.result().get_counts()
int_counts = {}
for bitstring in bit_counts:
  int_counts[ int(bitstring,2) ] = bit_counts[bitstring]
```

```
# (7) Tulosten visualisointi
plot_histogram(int_counts)
```

Edellä kuvattu esimerkkiohjelma havainnollistaa Python -ohjelmointikielen käyttöä kvanttiohjelmoinnissa: varsinainen kvanttiohjelmointi tapahtuu vaiheissa (3–5), joissa määritellään suoritettava piiri. Vaiheessa 6 ohjelma (*job*) suoritetaan simulaationa, tässä tapauksessa 8192 kertaa, ja mittaustulokset luetaan taulukkoon, josta tulokset puolestaan syötetään pylväskuvaajan piirtoon (vaihe 4). Huomion arvoista tässä esimerkissä on forsilmukan käyttö kohdassa 4: kvanttitietokonehan ei tue toistoa, vaan ohjelmatulkki purkaa silmukan osiin, se on ohjelmointikielen tasolla tapahtuva kuvaus, joka helpottaa ohjelmoijan työtä.



Kuva 4.3. *Satunnaislukujen generoinnin tulos*

Ajon jälkeen (vaihe 7) muodostettu histogrammi (*Kuva 4.3*) visualisoi tuloksien (luku 0-7) todennäköisyydet, josta nähdään, että luvut ovat generoituneet melko tasaisesti. Se, että jakauma ei ole täysin tasainen, johtuu kvanttitietokoneen epädeterminisestä luonteesta eli aikaisemmin kuvatusta virheestä, ja ajojen tulokset vaativatkin aina tulkintaa. Tässä simulaatiossa virhekin on simuloitu eikä sinällään vastaa todellisuutta, kuitenkin ajetaessa ohjelmaa oikeassa kvanttitietokoneessa, tulos on toki samankaltainen.

Simulaatiosta

Tämä esimerkkiohjelma toteuttaa varsin yksinkertaisen algoritmin, monimutkaisempien algoritmien suunnittelu on luonnollisesti vaikeaa ja vaatiikin lukuisia iteraatiokierroksia ja tulosten tulkintametodien kehittämistä. Simulaatioympäristöt mahdollistavatkin algoritmien kehitystyön tekemisen, jopa laitteilla joita ei vielä ole olemassakaan: esimerkiksi simulaatiossa kubittien määrää voi kasvattaa lähes rajattomasti. Simulaatiossa virhekin on simuloitua, seikka joka voi tuottaa liian optimistisia algoritmeja, joiden käyttökelpoisuus selviää vasta, kun ohjelma suoritetaan konkreettisella laitteistolla. Toisaalta simulaatioympäristöissäkin virheiden simulointi paranee jatkuvasti.

Kvanttiohjelmien suoritus simulaatioympäristöissä on joka tapauksessa tehokas – ja usein ainoa – menetelmä opetukseen, sekä pedagogisesti että kustannusten suhteen.

5. YHTEENVETO

Kvanttitietokoneet kehittyvät nopeudella, jota voi verrata mikropiiriteknologian edistymiseen 1960-alusta alkaen. Mooren laki onnistui mallintamaan mikropiiritekniikan kehitystä, mutta puolijohdetekniikan pakkaus- ja suorituskyky on alkanut saavuttaa rajansa. Kvanttitietotekniikalla ja perinteisellä puolijohdetekniikalla on paljon yhteistä teoreettista taustaa, mutta kuitenkin ratkaisevia teknisiä eroja. Kvanttimekaniikkaan perustuvia tietokoneita voikin pitää teknologisena jatkumona puolijohteisiin perustuvien tekniikoille.

Kvanttimekaniikkaan perustuvien teknologioiden kehitykseen ja tutkimukseen panostetaan runsaasti resursseja. Suuret teknologia- ja ohjelmistoyritykset - esimerkiksi Microsoft, IBM, Google ja Amazon - allokoivat siihen runsaasti henkilöstöä, teknologian tutkimusta ja taloudellisia resursseja. Samaan aikaan on alalle päässyt vakiintumaan uusia yrityksiä, esimerkiksi D-Wave ja myös startup-yritysten, esimerkkinä suomessakin toimiva IQM, määrä kasvaa vuosi vuodelta. Korkeakouluissa tehdään myös merkittävää, innovatiivista kehitystyötä ja niidenkin rahoitusta on lisätty huomattavasti. Valtiot (ja niiden yhteenliittymät) on kolmas merkittävä toimija.

Uusien tuotteistuksien myötä kvanttitietokoneiden markkina on ottanut merkittävän askeleen eteenpäin luomalla kvanttitietokoneista ja samalla niiden ohjelmoinnista yrityksille uudenlaisen kilpailuedun, samaan tapaan kuin 1960-luvulta alkaen mikropiiritekniikoihin perustuva tietokonetekniikka.

Yllä kuvattu teknologiakentän laajeneminen tarvitsee osaajia pystyäkseen ylläpitämään positiivista kehitystä: uusien ja kehittyvien teknologioiden ja tekniikoiden osaajia tarvitaan lisää; toisaalta osaajien koulutus edistää alan kehitystä ja innovaatioita.

Opetuksen on syytä pysyä mukana tämänkin alan teoreettisessa ja konkreettisessa kehityksessä. Suomessa kvanttiohjelmoinnin opetuksen aloittaminen on ollut hidasta ja tarve kvanttiohjelmoinnin opetuksen edistämiseen onkin suuri. Suomalaisella koulutusjärjestelmällä on hyvät edellytykset tähän hyödyntäen olemassa olevaa perinteisen ohjelmoinnin kokemusta ja tietämystä. Opetukseen on tarjolla monia ymäristöjä ja välineitä ja tutkielman kirjoittaja koki näistä lupaavimpana IBM:n Qiskit -ympäristön ja Jupyter -alustan. Ensin mainitulla kvanttiohjelmoinnin opetuksen voi integroida osaksi nykyistä ohjelmoinnin opetusta ja jälkimmäistä voi käyttää muussakin luokka- ja monimuotoisessa tietotekniikan opetuksessa.

LÄHTEET

Aalto yliopisto: Practical Quantum Computing (CS-C3260). (2022, marraskuuta 3). Haettu marraskuuta 3, 2022, osoitteesta https://sisu.aalto.fi/student/courseunit/aalto-CU-1150933383-20220801/brochure

Al-Gahmi, A., Zhang, Y., & Valle, H. (2022). Jupyter in the Classroom: An Experience Report [ISBN:9781450390705]. *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, 425–431. https://doi.org/10.1145/3478431.3499379

Azure Quantum. (2022). Haettu marraskuuta 22, 2022, osoitteesta https://learn.microsoft.com/en-us/azure/quantum/overview-azure-quantum

Azure Quantum Workflow. (2022). Haettu marraskuuta 22, 2022, osoitteesta https://learn.microsoft.com/en-us/azure/quantum/media/quantum-development-kit-flow-diagram.svg

Bauer, F. L., & Wössner, H. (1972). The “Plankalkül” of Konrad Zuse: A Forerunner of Today’s Programming Languages. *Commun. ACM*, *15*(7), 678–685. https://doi.org/10.1145/361454.361515

Dahm, A. J., Heilman, J. A., Karakurt, I., & Peshek, T. J. (2003). Quantum computing with electrons on helium. *Physica E: Low-dimensional Systems and Nanostructures*, *18*(1), 169–172. https://doi.org/https://doi.org/10.1016/S1386-9477(02)01075-5 23rd International Conference on Low Temperature Physics (LT23).

Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2021). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, *29*(1), 12–28. https://doi.org/10.1002/cae.22255

Haigh, T., Priestley, M., & Rope, C. (2014). Engineering "The Miracle of the ENIAC": Implementing the Modern Code Paradigm. *IEEE Annals of the History of Computing*, *36*(2), 41–59. https://doi.org/10.1109/MAHC.2014.15

IBM Qiskit Toolkit. (2022, marraskuuta 16). Haettu marraskuuta 22, 2022, osoitteesta https://qiskit.org/overview/

Jones, S. (2008). *The quantum ten a story of passion, tragedy, ambition and science* [ISBN: 978-0195369090]. Oxford University Press.

Mousavi, M., Houshmand, M., & Bolokian, M. (2021). The Cost Reduction of Distributed Quantum Factorization Circuits. *International journal of theoretical physics*, *60*(4), 1292–1298. https://doi.org/https://doi.org/10.1007/s10773-021-04756-6

Nelson, M. J., & Hoover, A. K. (2020). Notes on Using Google Colaboratory in AI Education. *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, 533–534. https://doi.org/10.1145/3341525.3393997

Oulun yliopisto: Oulun yliopisto, Quantum information (763635S). (2022). Haettu marraskuuta 3, 2022, osoitteesta https://opas.peppi.oulu.fi/en/course/763635S/7967

Patel, A. D. (2021). Grover's algorithm in natural settings. *Quantum information & computation*, *21*(9-10), 945–954. https://doi.org/https://doi.org/10.48550/arXiv.2001.00214

Phillips, R. (2013). In retrospect: The Feynman Lectures on Physics. *Nature*, *504*(7478), 30–31. https://doi.org/10.1038/504030a

Plimmer, B. (1998). Machines Invented for WW II Code Breaking. *SIGCSE Bull.*, *30*(4), 37–40. https://doi.org/10.1145/306286.306309

Quantum Computing: FCC155. (2022). Haettu marraskuuta 3, 2022, osoitteesta https://www.gu.se/en/study-gothenburg/quantum-computing-fcc155

Quantum information. (2022). Haettu marraskuuta 3, 2022, osoitteesta https://www.cs.ox.ac.uk/teaching/courses/2022-2023/qi/

Rossant, C. (2018). *Interactive Computing with Jupyter Notebook (Video)* [ISBN: 978-1-789-53150-3]. Packt Publishing.

Sánchez, P., & Alonso, D. (2021). On the Definition of Quantum Programming Modules. *Applied Sciences*, *11*(13). https://doi.org/10.3390/app11135843

Seegerer, S., Michaeli, T., & Romeike, R. (2021). Quantum Computing As a Topic in Computer Science Education. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3481312.3481348

Sentance, S., Barendsen, E., & Schulte, C. (Toim.). (2018). *Computer science education: perspectives on teaching and learning in school* [ISBN: 978-1-350-05711-1]. Bloomsbury Academic.

Temporão, G. P., Guerreiro, T. B. S., Ripper, P. S. C., & Pavani, A. M. B. (2022). Teaching Quantum Computing without prerequisites: a case study. *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 673–676. https://doi.org/10.1109/QCE53715.2022.00090

Wootton, J. R., Harkins, F., Bronn, N. T., Vazquez, A. C., Phan, A., & Asfaw, A. T. (2021). Teaching quantum computing with an interactive textbook. *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 385–391. https://doi.org/10.1109/QCE52317.2021.00058