Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2022

Ville-Markus Aho

# Lähi-infrapunaspektrin visualisointi Pythonilla

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Opinnäytetyö (AMK) | Tiivistelmä

Turun ammattikorkeakoulu

Tieto- ja viestintätekniikka

2022 | 36 sivua, 5 liitesivua

Ville-Markus Aho

# Lähi-infrapunaspektrin visualisointi Pythonilla

Lähi-infrapunalla tarkoitetaan sähkömagneettisen säteilyn aluetta näkyvän valon spektrin lopusta keski-infrapunaspektrin alkuun. Infrapunan avulla pystytään tunnistamaan aineesta eri molekyylejä ja niiden lukumääriä. Lähi-infrapunaa käytetään esimerkiksi öljyteollisuuden laadunhallinnassa ja lääkinnällisissä tutkimuksissa.

Työssä tutkittiin lähi-infrapunan spektroskooppisia menetelmiä sekä Pythonin visualisointikirjastoja. Tavoitteena oli suunnitella ja toteuttaa lähi-infrapunaspektriä visualisoiva sovellus Python-ohjelmointikielellä. Suunnittelun lähtökohtana oli automatisoida koodissa tapahtuva tiedonkäsittely sekä prosessin visualisointi, jotta käyttäjäsyöte pysyisi sovelluksessa minimaalisena.

Työn tuotoksena valmistui sovellus, joka lukee käyttäjän valitseman tiedoston ja visualisoi sen datan. Loppukäyttäjiä huomioiden sovellus suunniteltiin käytettäväksi graafisen käyttöliittymän avulla. Lisäksi kehityksessä kiinnitettiin huomiota sovelluksen käyttäjäinteraktiivisiin toimintoihin, joiden avulla datan tarkastelu helpottuu. Kaikki sovelluksen toiminnot on sijoiteltu yhdelle näkymälle. Tämän tarkoituksena on saavuttaa yksinkertainen ja helppokäyttöinen käyttökokemus. Lähi-infrapunan ja toteutetun spektriä visualisoivan sovellustyökalun avulla suun terveyden huollon ammattilaiset pystyvät tehostamaan suusyövän diagnosointia sen varhaisessa vaiheessa.

Asiasanat:

Python, Spektroskopia, Matplotlib, infrapuna, NIR, Lähi-infrapuna

Bachelor’s Thesis | Abstract

Turku University of Applied Sciences

Information and Communications Technology

2022 | 36 pages, 5 pages in appendices

Ville-Markus Aho

# Visualizing near-infrared spectrum with Python

Near-infrared spectrum is electromagnetic radiation that begins at the end of the visible light spectrum and ends at the beginning of the mid-infrared spectrum. Infrared light can be used to identify the molecules of substances and their quantities. Near-infrared is used, for example in medical research and quality management in the oil industry.

The objective of this thesis was to design and implement an application for visualizing the near-infrared spectrum in the Python programming language. The purpose of the design was to automate the data processing in the code and the visualization process so that user input in the application would be kept to a minimum. To achieve the objective of this thesis near-infrared spectroscopic methods and Python visualization libraries were studied and utilized.

The output of this thesis was an application that reads a user-selected file and visualizes its data. With end users in mind, the app was designed to be used with a graphical user interface. The development also focused on the user-interactive features of the application, which facilitates the user to explore the data. The application GUI contains a single view. This is to achieve a simple and easy to use experience. With the use of near-infrared light and visualization application, oral health care professionals are able to make the diagnosis of oral cancer more effective at its early stage.

Keywords:

Python, Matplotlib, spectroscopy, infrared, NIR, near-infrared

# Sisältö



# Liitteet

# Kuvat



# Kuviot



# Taulukot

# Käytetyt lyhenteet

| | |
|---|---|
| nm | nanometri |
| IR | Infrapuna (engl. infrared) |
| NIR | lähi-infrapuna (engl. near-infrared) |
| NIRS | lähi-infrapunaspektroskopia (engl. Near infrared spectroscopy ) |
| GUI | graafinen käyttöliittymä (engl. graphical user interface) |
| FT | Fourier-muunnos (engl. Fourier-transform) |
| DG | Dispersive Grating on menetelmä, jolla valoa hajotetaan eri aallonpituuksille |
| PET | Positroniemissiotomografia |
| MRI | Magneettikuvaus |
| SPECT | Yksifotoniemissiotomografia |
| API | Ohjelmointirajapinta (engl. Application programming interface) |
| PNG | Portable Network Graphics, korkealaatuinen kuvien tallennusformaatti |
| JPEG | Joint Photographic Experts Group, kuvien tallennusformaatti |
| PDF | Portable Document Format, siirrettävä tiedostomuoto |

# 1 Johdanto

Visualisoinnin tarkoituksena on muuttaa informatiivista dataa ihmiselle helposti ymmärrettävään muotoon. Tämän avulla havaitsija pystyy ymmärtämään suuriakin määriä dataa nopeasti, tunnistamaan tiedon muuttujien väliset riippuvuudet sekä löytämään virheitä ja poikkeavaisuuksia tutkitusta datasta [1]. Visuaalisia esitystapoja eli graafeja käytetään analysoinnin apuna esimerkiksi erilaisissa tieteellisissä sovellutuksissa sekä yritys- ja elinkeinoelämän tilastollisissa tutkimuksissa.

Python on suosittu ohjelmointikieli, jota käytetään mm. tieteellisessä laskennassa [2]. Python-kielelle on saatavilla lukuisia tiedonkäsittelyyn ja -visualisointiin erikoistuneita ohjelmointikirjastoja, jotka ovat ilmaisia visualisointiratkaisuja kaupallisten ohjelmistojen rinnalle.

Tämä opinnäytetyö on tehty toimeksiantona Itä-Suomen yliopiston Sovelletun fysiikan laitokselle. Tavoitteena oli toteuttaa Python-kielellä sovellus lähi-infrapunaspektrin visualisointia varten, mitä hammaslääkäri tai -hoitaja pystyy käyttämään diagnostisena työkaluna suusyövän varhaisessa seulonnassa. Syövän varhainen diagnosointi on tärkeää, sillä se parantaa syöpäpotilaan ennustetta.

Tässä työssä tutkittiin lähi-infrapunan spektroskooppisiamenetelmiä ja Pythonille saatavia olevia visualisointikirjastoja.Tulosten pohjalta on suunniteltu ja toteutettu visualisointisovellus, jonka tarkoituksena on automatisoida datan visualisointiin vaadittavia prosesseja sekä sujuvoittaa käyttökokemusta datan tarkastelua helpottavien toimintojen avulla. Sovellus on suunniteltu käytettäväksi graafisen käyttöliittymän kautta, koska sen on oltavat terveydenhuollon toimihenkilöille eli loppukäyttäjille helppokäyttöinen. Vaikka toteutettu sovellustyökalu on yleiskäyttöinen, tullaan sitä käyttämään ensisijaisesti spektroskooppisten funktioiden visualisoimisessa. Tämän vuoksi sovellus tulee hyödyntämään perinteisiä viivadiagrammeja datan mallintamisessa, ja niihin tässä opinnäytetyössä keskitytään.

# 2 Lähi-infrapuna

Infrapuna on ei näkyvää sähkömagneettista säteilyä, jonka aallonpituus kattaa sähkömagneettisen spektrin 700–$10^6$ nm:n [3]. Sitä hyödynnetään laajasti eri teollissuuden aloilla ja myös arkielämän teknologioissa, kuten tietokoneiden hiirissä, televisioiden kaukosäätimissä, erilaisissa lämmitysmenetelmissä, kuten infrapunasaunoissa sekä huoneistojen lämmityksessä käytettävissä kattolämmityselementeissä. [4]

Lääketieteellisessä käytössä infrapunan avulla pystytään paikantaan esimerkiksi tulehduksia, ”sillä tulehdus tuottaa lämpöä”. Infrapunalla pystytään myös edistään haavojen parantumista ja lievittämään kipua. Hoitomenetelmät perustuvat infrapunasäteilyn kykyyn läpäistä kudosta ja sen tuottamaan lämpöön. [4]

Lähi-infrapunalla tarkoitetaan infrapunan 780–2500 nm:n aluetta, joka alkaa näkyvän valon spektrin lopusta eli valon punaisesta väristä, ja loppuu keski-infrapuna alueen alkuun [5]. Termillä lähi-infrapuna tarkoitetaankin infrapuna-aluetta, joka on lähellä näkyvää valoa. Spektroskooppisissa menetelmissä tyypillisesti käytetään 650–1100 nm säteilyä, koska se on osoittautunut optimaaliseksi alueeksi biologisten näytteiden analyysissä. [6]

## 2.1 NIR-spektroskopia

Spektroskopialla tarkoitetaan sähkömagneettisen säteilyn jakamista eri aallonpituuksille tai taajuuksille eli spektriksi. Menetelmällä pystytään tutkimaan miten molekyylit, atomit tai niiden sidokset imevät itseensä eli absorboivat, tai heijastavat säteilyä. [7]

Lähi-infrapunaspektroskopia eli NIRS (Near infrared spectroscopy) on infrapunaa hyödyntävä mittausmenetelmä, jota käytetään ja tutkitaan monilla eri tieteen aloilla, erityisesti fysiikassa ja kemiassa. Melkein kaikki molekyylit absorboivat infrapunavaloa jollakin aallonpituudella. Tämän ominaisuuden

avulla pystytään esimerkiksi tunnistamaan aineesta eri molekyylejä sekä niiden lukumääriä. [4]

Infrapunan NIR-spektrialueen ominaisuudet ovat osoittautuneet optimaalisiksi eri aineen olomuotojen analysoinnissa. NIRS menetelmä tuo lukuisia käytännön etuja näytteen analysoinnissa sen helpon, tarkan ja resurssittoman käytön ansiosta. Analysointioprosessiin kuuluu pelkästään näytteen asettaminen ja mittalaitteen käynnistäminen. Prosessia helpottaa myös se, että suurinta osaa näytteistä ei tarvitse esikäsitellä, vaan niitä pystytään analysoimaan täysin sellaisinaan. Itse analyysiprosessi on nopea, se kestää arviolta kymmenestä sekunnista kahteen minuuttiin. Menetelmällä pystytään näytteestä tunnistaan useita eri aineita yhdellä näyteanalyysillä. NIRS on myös ympäristöystävällinen ja taloudellinen, koska sen käyttö ei tuota haitallisia jätteitä ja eikä sitä käyttäessä kulu sähköä lukuunottamatta muita resursseja. [8]

NIR-valon hyödyntäminen perustuu sen vaikutukseen molekyylien happivety-, typpivety- sekä hiilivetysidoksissa. Sidokset reagoivat lähi-infrapunaan eri aallonpituuksilla, ja kun analysoitava näyte, jossa on paljon kyseisiä sidoksia altistetaan NIR-valolle, absorboi näyte itseensä energian niiltä valon aallonpituuksilta joihin nämä sidokset reagoivat, tämän seurauksena näytteestä heijastuneen valon voimakkuus näillä alueilla on pienempi. Heijastuneen valon signaalit, eli spektrin erot korreloivat kemiallisten pitoisuuksien eroihin näin muodostaen pohjan NIR-kalibroinnille. Sen avulla pystytään arvioimaan kemiallisia pitoisuuksia tuntemattomista näytteistä. [8]

## 2.2 Lähi-infrapunan sovellutukset

NIR-spektriä mittaavia laitteita eli spektrometrejä, käytetään teollisuudessa esim. kemiallisessa analyysissä. NIR-spektrometrejä on kahta tyyppiä: dispersiivinen spektrometri ja Fourier-muunnokseen perustuva spektrometri. [9]

Vaikka molemmat spektrometrityypit soveltuvat samoihin tarkoituksiin, pidetään dispersiivistä lähi-infrapunaspektrometriä (DG-NIR) erinomaisena menetelmänä analysoida kaasuja ja kirkkaita nesteitä, varsinkin silloin kun mittauksessa valoa

siirretään optisilla kuiduilla. Fourier-muunnokseen perustuva lähi-infrapunaspektrometri (FT-NIR-spektrometri) puolestaan soveltuu hyvin kiinteiden ja jauhemaisten aineiden mittaamisessa ilman optisia kuituja. Huomioitavaa kuitenkin on, että yleiskäyttöiset Fourier-muunnoksen-infrapunaspektrometrit (FT-IR) eivät ole optimaalisia NIR-alueen mittaamiseen ja ne menettävät monia etuja lähi-infrapunan alueella, koska FT-menetelmät soveltuvat hyvin energiarajoitetulle infrapuna-alueelle, jota NIR-säteily ei puolestaan ole. [10]

NIR-spektroskopiaa hyödynnetään erityisesti teollisissa laadunhallinta ja varmennus tehtävissä sen aineita mittaavien ominaisuuksien ansiosta. Öljyteollisuus käyttää NIRS-menetelmiä petrokemiallisten aineiden, kuten dieselien ja raakaöljyjen jalostuksen seurannassa. NIRS-analyysi on ei-destruktiivinen, eli se ei vahingoita, eikä muutenkaan vaikuta itse analysoitavaan näytteeseen. Siksi se soveltuu erinomaisesti arvokkaiden ja vaarallisten näytteiden analysointiin. [9]

Lääkinnällisessä käytössä NIRS:n suurimmat edut tulevat sen ei-invasiivisesta toiminnasta: sillä pystytään suorittamaan kehon sisäisiä mittauksia sen ulkopuolelta ilman kehonsisälle tunkeutuvia instrumentteja. NIR sitoutuu huonosti vesi- ja hemoglobiinimolekyyleihin, joten se pystyy tunkeutumaan kudoksen läpi vähäisellä energiahäviöllä. [6]

Yksi tyypillisimpiä ei-invasiivisia toimenpiteitä, mihin lähi-infrapunasäteilyä käytetään, on aivotoimintojen kuvantamisessa. Sen avulla pystytään esimerkiksi kuvantamaan aivojen verenkiertoa. Vaikka kyseiseen toimenpiteeseen pystytään käyttämään muita kuvantamistekniikoita, kuten PET, MRI ja SPECT, ovat kyseiset menetelmät hyvin herkkiä kuvattavan potilaan liikkeelle, jolloin kuvantaminen ja diagnostiikka vaikeentuu potilaan tahattomastakin liikehdinnästä. NIR-spektroskopia käyttää veren happiarvoja kuvatakseen hermoston toimintaa. Kun aivojen hermostollinen toiminta lisääntyy, vilkastuu myös aivojen verenkierto. Tämän seurauksena hapen määrä veressä lisääntyy, mikä puolestaan tarkoittaa veren happipitoisten hemoglobiinien määrän kasvua ja hapettomien laskua. NIRS käyttää näiden lukumäärien suhteellisia arvoja

mitatakseen hermoston aktiivisuutta, tämän vuoksi NIRS:n avulla pystytään kuvantamaan vaikka kävelevää potilasta. [6]

# 3 Visualisointimenetelmät

Datan visualisoinnin tarkoituksena on muuttaa yleensä numeerista tietoa ihmiselle helposti ymmärrettävään muotoon. Pythonilla on olemassa lukuisia datan visualisointiin erikoistuneita kirjastoja, jotka tarjoavat lukuisia diagrammeja ja kaavioita datan esittämiselle. Python ohjelmointikieltä käytetäänkin paljon tieteellisissä sovellutuksissa, kuten laskennassa ja datan käsittelyssä. Monet saatavilla olevista Python-kirjastoista on kirjoitettu Pythonia nopeammalla kielellä, kuten esimerkiksi C-kielellä. [11]

Spektroskooppisissa analyyseissa, kuten infrapunaspektroskopiassa, mitataan valon aallonpituutta tai taajuutta jonkin fysikaalisen suureen esim. absorption funktiona. [7]

Pythonin visualisointikirjastoja vertailtaessa tulisi kiinnittää huomiota seuraaviin ominaisuuksiin: minkälaiseen käyttötarkoitukseen kyseinen kirjasto soveltuu, mitä visualisointityylejä on saatavilla, esteettisyyteen, interaktiivisuuteen, käytettävyyteen kooditasolla, minkälaisia tietorakenteita kirjaston funktiot käyttävät, kuinka kustomoitavia graafit ja niiden ominaisuudet ovat, sekä minkälaisia käyttöliittymäympäristöjä kirjasto hyödyntää. [12]

## 3.1 Matplotlib-visualisointikirjasto

Matplotlib on yksi käytetyin Pythonin datan mallinnuskirjasto. Se on syntaksiltaan ja käytettävyydeltään samankaltainen MATLAB-ohjelmiston ja sen ohjelmointikielen kanssa. Sillä pystyy tuottamaan niin staattisia, kuin dynaamisia kaksiuloitteisia graafeja. Pystyäkseen tuottamaan 3D-diagrammeja tarvitaan erillinen mplot3d-työkalu. [12]

Kirjasto tarjoaa matalan tason ohjelmointirajapinnan, mikä mahdollistaa käyttäjän muokata ja laajentaa ohjelmaa omiin tarkoituksiinsa. Vastaavasti se tarkoittaa myös sitä, että kirjaston käyttäminen vaatii ylimääräisen koodin kirjoittamista pienienkin toimintojen lisäämiseen. [13]

Matplotlib mahdollistaa diagrammien upottamisen moniin Pythonin GUI-kirjastoihin, mm. Tkinter, PyQT ja wxPython ovat Matplotlibin tukemia. Kirjastolla pystyy myös lisäämään käyttäjän syötteeseen, kuten hiiren ja näppäimistön toimintoihin perustuvia tapahtumia (engl. event handling) diagrammeihin. [14]

Graafien visualisoinnista vastaavat funktiot ottavat datan vastaan taulukkoina (engl. array). Tiedonkäsittelyssä hyödynnetään usein Pythonin muita datan käsittelyyn erikoistuneita kirjastoja, kuten NumPy- ja pandas-kirjastoa. Niiden avulla pystyy helpottamaan monimutkaisten tietorakenteiden visualisointia, tällöin myös vaadittava koodin määrä ohjelmassa pienenee. [12]

Pyplot on Matplotlibin mukana tuleva moduuli, joka mukailee MATLAB-kielen syntaksia. Sen avulla kuvaajia pystyy visualisoimaan syöttämällä kuvaajien x- ja y-koordinaatit pyplotin funktioihin. Se sisältää myös GUI-managerin, joka huolehtii sovellusikkunan avaamisesta ja käyttöliittymän tilojen hallinnasta. Pyplotin avulla kirjastoa pystyy käyttämään komentokielenä Python Shellillä. [15]

Matplotlib on enimmäkseen suunnattu perusgraafien, kuten viiva-, pylväs- ja hajontakuvaajien visualisointiin [16]. Kirjastoa on helppo käyttää yksittäisten numeerisia arvoja sisältävien taulukkojen visualisoinnissa, mutta mikäli aikomuksena on kehittää ohjelma, joka sisältää kirjaston mukana tulevia toimintoja hienompia ratkaisuja ja lisäyksiä, tai silloin kun ohjelmassa käsiteltävä data on formatoitu hienojakoisempaan muotoon tai se koostuu useista tietojoukoista, vaati kirjaston käyttö huolellisempaa perehtymistä sen dokumentteihin. Tällaisissa käyttötapauksissa visualisointiprosessi tulee helposti riippuvaiseksi muista Python kirjastoista. [12]

### 3.2 Plotly Python -interaktiivinen grafiikkakirjasto

Plotlyn visualisointikirjasto on selainkäyttöinen viitekehys, joka muuntaa Python koodin syötteen HTML-muotoon. Se on rakennettu saman nimisen JavaScript-

kirjaston plotly.js:n päälle. Kirjasto on ilmainen ja avoimeen lähdekoodiin perustuva. [12]

Toisin kuin matalan tason Matplotlib, Plotlylla pystyy tuottaamaan interaktiivisia diagrammeja ja kaavioita ilman että näitä ominaisuuksia tarvitsitsi erikseen lisätä koodin tasolla. Kaavioita pystyy zoomaamaan ja panoroimaan, kursorin tooltipin avulla pystyy näkemään kuvaajan arvoja yms. Lisäksi käytettävyyttä ja interaktiivisuutta pystyy laajentamaan lisäämällä käyttöliittymään esim. painikkeita, slidereita tai pudotusvalikoita visualisoinnin helpottamiseksi ja käytettävyyden parantamiseksi. [17]

Kirjasto sisältää monia graafeja, joita Matplotlib kirjastossa ei ole, kuten esim. tilastollisia-, finanssialan-, tieteellisiädiagrammeja sekä erilaisia maantieteellisiä 3D-karttoja. [17]

Matplotlibin tavoin Plotlylla pystyy muodostaan graafeja syöttämällä taulukkoja plottausfunktioiden x ja y parametreihin. Plotlyn funktiot pystyvät myös renderöimään suoraan Pandaksen dataframe olioista. Datan käsittelyssä voi hyödyntää myös Pythonin dictionary-tietorakennetta, ja yleisesti ottaen Plotlyssa tiedonkäsittely on huomattavasti monimuotoisempaa kuin Matplotlibissä. [18] Koska kirjasto toimii web-käyttöliittymällä, pystyy sitä käyttämään myös nettisivujen frontend-kehityksessä Plotlyn Dash-kirjaston avulla. Dashilla pystyy tekemään kustomoitavia dashboard-näkymiä Plotlyn graafeista pelkällä Python koodilla. [19] Kirjastolla pystyy täysin hallitsemaan ja muokkaamaan graafien esteettisyyttä: graafien teemoja pystyy vaihtamaan kirjaston saatavilla oleviin teemoihin tai niitä pystyy myös luomaan itse. [20]

Vaikka Plotly on laaja ja monipuolinen kirjasto, kritisoidaan sitä siitä, että sen dokumentaatiot eivät ole ajantasalla itse kirjaston kehityksen kanssa. Myös oheisteknologioiden runsas määrä vaikeuttaa kirjaston kehityksen seuraamista. [12]

Toinen puute Plotlyssa on, että sillä on haastavaa visualisoida erillisiä tietojoukkoja ja yhdistää niiden graafeja samaan lähteeseen. Lisäksi silloin kun Plotlya käytetään dashboard-sovelluksena, sen viitekehys  Dash ei tallenna

käyttäjän tilojen tietueita (engl. records), eli on tilaton (engl. stateless). Tämä vaikeuttaa käyttöliittymän ja taustasovelluksen välistä viestintää siinä määrin, että laskennallisesti vaativien tehtävien toteuttaminen ja suorittaminen voi vaatia ylimääräisten tekniikoiden käyttöä esim. väylämuistin käyttöä, siksi kirjasto voi olla epäkäytännöllinen tilanteissa, joissa graafin kuvaajia päivitetään ohjelman ajon aikana. [21]

### 3.3 Bokeh-datan mallinnuskirjasto

Plotlyn tavoin Bokeh on selainkäyttöinen ja keskittyy visualisoimaan interaktiivisia diagrammeja ja kaavioita [22]. Bokeh on arkkitehtuuriltaan kolmikerroksinen, joista ensimmäinen keskittyy kaavioiden nopeaan piirtämiseen, toinen kerros puolestaan hallinnoi kaavioiden piirtämistä oletus komponenttien kanssa ja kolmannessa kerroksessa käyttäjällä on täysi hallinta siitä, kuinka kaavioita piirretään ja esitetään tarvittavilla ominaisuuksilla. Tämä arkkitehtuuri mahdollistaan sen, että kirjastoa pystyy käyttämään vähäisellä koodilla tuottamaan interaktiivisia ja yksinkertaisia graafeja ilman ylimääräisiä ominaisuuksia. Tai sillä pystyy tuottaan täysin kustomoituja graafeja, johon on lisätty haluttuja toiminnallisuuksia ja ominaisuuksia, mutta vastaavasti tarvittavan koodin määrä lisääntyy. [12]

Bokeh soveltuu parhaiten sellaisiin käyttötarkoituksiin, joissa visualisoidaan suuria määriä tietojoukkoja tai silloin kun dataa vastaanotetaan lähteestä jatkuvalla syötöllä, eli striimataan (engl. streaming data). [23]

Bokeh toimii palvelin-asiakas-tyyppisellä (engl. server-client) arkkitehtuurilla, josta Python mallintaa datan graafien objekteiksi, näitä ovat esim. diagrammin kuvaajat, koordinaatiston etäisyydet ja akselit. Nämä objektit viedään JSON-muotoon, josta JavaScriptin BokehJS-kirjasto tuottaa selaimeen näytettävän sisällön. [24]

Kirjastossa on valinnainen Bokeh Server -komponentti, jonka avulla pystyy tehostamaan selaimen ja Python-ympäristön välistä vuorovaikutusta. Sen tarkoitus on synkronoida niiden välillä kulkeva data. [25] Esimerkiksi, jos

aikomuksena on toteuttaa käyttöliittymä, jossa käyttäjä pystyy halutessaan muuttamaan kuvaajan arvoja käyttöliittymän kautta. Arvojen muuttuessa päivittyvät uudet arvot myös palvelinpuolella, jolloin palvelin lähettää takaisinkutsuna (engl. callback) uuden dokumentin selaimelle ja näin saadaan kuvaaja piirrettyä uusilla arvoilla. [24]

Kirjastolla pystyy käyttämään JavaScriptin WebGL-grafiikkarajapintaa diagrammien renderöinnissä. Sen avulla saadaan paremmin hyödynnettyä näytönohjainten laskennallisia tehoja datan visualisoinnissa. Näin saadaan minimoitua renderöintiin käytetty aika, vaikka dataa olisi paljon. [26]

Bokeh on enemmänkin suunnattu perinteisten 2D-diagrammien visualisointiin, jossa kuvaajia hahmotetaan xy-koordinaatistossa. Kirjastossa ei ole toistaiseksi natiivia tukea 3D-grafiikoille, vaan tarvittavat ominaisuudet on tuotava toisista paketeista esim. JavaScriptin VisJS-kirjastosta. Plotlyyn nähden saatavilla olevista visualisointi menetelmistä ei löydy sellaisia erikoisuuksia, mitä ei muista kirjastoista löytyisi. [27]

Plotlyn tavoin Bokeh-kirjastoa voi käyttää myös web-kehityksessä. Kirjasto on ensisijaisesti suunniteltu käytettäväksi Pythonilla, mutta on mahdollista myös käyttää pelkkää JavaScriptiä ja BokehJS-kirjastoa tuottamaan diagrammeja ja niiden kuvaajia kirjaston omalla BokehJS:n API:lla. [28]

Kuten muissa kirjastoissa, Bokeh pystyy hyödyntämään taulukoita, listoja ja Dataframe-tietorakenteita kuvaajien tuottamiseksi. Näiden lisäksi kirjastossa on omanlainen tietorakenne datan käsittelyä varten, ColumnDataSource. Sitä voisi kuvailla yhdistelmäksi sanakirjaa (engl. dictionary) ja kaksiulotteista taulukkoa. Sen avulla pystyy tuottaan useita erillisiä graafeja saman datalähteen arvoista, tällöin niitä on helppo linkittää toisiinsa, koska niiden yhteinen tietojoukko tunnetaan. ColumnDataSourcen dataa pystyy helposti lisäämään ja muokkaamaan, näin saadaan myös päivitettyä kaikki ne kuvaajat, jotka piirtyvät yhteisillä arvoilla. [21]

Pythonin visualisointiin tarkoitetuista kirjastoista Bokeh on parhaiten verratavissa Plotlyyn näiden samankaltaisen toimintamallin ja

selainkäyttöliittymän vuoksi, ja koska molempia voidaan käyttään web-kehityksessä, on Bokehissa sellaisia seikkoja Plotlyyn verratessa, joita olisi hyvä huomioida, kun arvioidaan kirjaston soveltuvuutta aiottuun käyttötarkoitukseen. Esimerkiksi, vaikka Bokehilla on mahdollista tuottaa diagrammeja pelkällä JavaScriptillä, on kirjasto siitä huolimatta vahvasti integroitu Python-kieleen, ja tästä syystä kirjaston ensisijainen web-kehityksessä käytettävä kieli on Python. Bokehin diagrammit eivät ole interaktiivisilta ominaisuuksiltaan niin monipuoliset ja laajennettavat kuin Plotlyssa. Myös diagrammien personointi ja esteettinen muokattavuus toteutuu kätevämmin Plotlylla. [29]

### 3.4 Seaborn-tilastokaaviokirjasto

Seaborn on Matplotlibin päälle rakennettu korkean tason kirjasto, jolla pystyy helposti tuottamaan tilastollisia diagrammeja esteettisesti miellyttävällä ulkoasulla [30]. Se on suunniteltu mukailemaan, mutta selkeyttämään Matplotlibin syntaksia. Kirjaston funktioihin on sisällytetty tiedonkäsittelyä helpottavia toimintoja, joiden avulla pystyy valita, muokata ja suodattaa dataa visualisointia varten. Lisäksi se sisältää diagrammeja, joita ei Matplotlibistä löydy. Seaborn kirjasto on Matplotlibin laajennos ja se vaatii toimiakseen Matplotlib-kirjastoa. [31]

Siinä missä Matplotlib mahdollistaa käyttöliittymän kustomoinnin erilaisiin käyttötapauksiin, keskittyy Seaborn-kirjasto yksinomaan tilastollisten graafien piirtämiseen sekä suoraviivaistamaan datan käsittelyä koodin tasolla. [16]

Koska Matplotlib toimii Seabornin moottorina, ovat kirjastojen laskennallinen tehokkuus ja oletuskäyttöliittymät lähes identtiset. Seaborn hyödyntää Matplotlibin käyttämiä graafisia käyttöliittymäkirjastoja luodakseen muista sovelluksista esim. selaimesta riippumattoman graafisen käyttöympäristön. [16]

Seaborn on päätoimintamalliltaan funktionaalinen. Se käyttää datan käsittelyssä funktioita jäsennelläkseen numeerisia tietojoukkoja yksittäisiksi kokonaisuuksiksi. Tämän vuoksi sillä on helppo yhdistellä eri tyylisiä kuvaajia

samalle diagrammille. Toisaalta funktionaalisuus on tilaton, ja aina kun halutaan päivittää kuvaajia, täytyy koko diagrammi uudelleen renderöidä. On myös mahdollista sitoa dataa olioihin Matplotlibin tavoin, jolloin pystytään päivittämään yksittäisiä kuvaajia ilman, että kaikkia kuvaajia tarvitsisi prosessoida uudelleen. [31]

Kirjasto on hyvin integroitu käsittelemään Pandas-datan käsittelykirjaston dataframe-tietorakenteita, joten dataa pystyy helposti käsittelemään Pandaksen funktioilla ja välittämään siitä suoraan Seabornin plottausfunktioille. Kyseiset funktiot ovat niitä kirjaston toimintoja, jotka vastaavat visualisoitavan datan käsittelystä ja sen renderöintiprosessista. [31] Niiden avulla on mahdollista valita, mitä dataframen sarakkeita halutaan visualisoida, eli on mahdollista suodattaa ei vältämättömiä tietoja pois esimerkiksi Excel-taulukosta. [32]

Kirjasto suosiikin selkeitä ja pitkämuotoisia dataframe-taulukoita syötteenä, jolloin visualisoitava data rakentuu myös luontevasti koodiin. Ongelmaksi saattaa koitua kuitenkin se, että käytettävää dataa voi joutua formatoimaan Seabornin hyväksymään muotoon, koska tällaista formaattia harvoin käytetään esimerkiksi Excel-tiedostoissa, joista dataa yleensä luetaan. On myös mahdollista syöttää kuvaajan x- ja y-koordinaattiarvoja taulukkoina kirjaston funktioihin, kuten Matplotlibissä. [32]

Diagrammeja ja kaavioita on mahdollista esteettisesti ehostaa ja personoida kirjastosta löytyvillä teemoilla. Tosin, mahdollinen ulkoasun kustomointi rajoittuukin siihen, sillä Seaborn keskittyykin korkean tasoisena kirjastona vähentämään visualisoinnissa tarvittavan koodin määrä tarjoamalla valmiita ratkaisuja. [33]

Seaborn ei itsessään tue interaktiivisuutta ja siksi sitä ei juurikaan käytetä interaktiiviseen visualisointiin. [12] Siinä on valmiina mukana Matplotlibin oletustoimintoja, kuten Zoomaus ja graafin tallentaminen kuva- tai pdf-tiedostoksi. Mikäli uusia interaktiivisiatoimintoja halutaan tuoda graafeihin, tulee muutokset tehdä Matplotlibin puolelle ja tällöin visualisointi on riippuvainen kahdesta kirjastosta.

Seabornin käyttöä kannattaa harkita esimerkiksi silloin, kun halutaan nopeasti visualisoida tietty aineisto raportissa käytettävää diagrammia varten. Mikäli diagrammin kuvaajia pitäisi pystyä analysoimaan oletuskäyttöliittymää hienommilla käyttökokemuksilla, on Seabornin käyttö hankalaa tällaiseen käyttötarkoitukseen.

### 3.5 Vertailu

Kooditason käytössä ja silloin kun data on jäsennelty yksinkertaiseen muotoon esim. ainoastaan numeroarvoja sisältäväksi taulukoksi, ovat kaikki edellämainitut kirjastot helppokäyttöisiä ja tuottavat pääpiirteittäin samanlaisen visuaalisen lopputuloksen. Kaikki kirjastot sisältävät yleisimmät data-analyysissä hyödynnettävät visualisointimenetelmät, kuten viiva-, pylväs- ja hajontadiagrammit, ja kaikissa pystytään apuna käyttämään Pythonin NumPy- ja Pandas-kirjastoja. Voidaankin todeta, että tämän tyyppisessä käytössä kirjaston valinta määräytyy pitkälti käyttäjän henkilökohtaisesta mieltymyksestä sekä aikaisemmasta käyttökokemuksesta. Jos käyttäjäkokemus on suuressa merkityksessä, halutaan suoraviivaistaa taulukkoja vaativampia tietorakenteiden käsittelyä koodissa tai silloin kun visualisoidaan perinteisiä diagrammeja erityisemmillä tavoilla, korostuvat tämän tyyppisissä käytöissä kirjastojen väliset erot.

Kaikki tässä opinnäytetyössä tutkitut visualisointikirjastot tukevat interaktiivista käyttöä. Seabornia on rajatapaus, sillä se ei itsessään ole interaktiivinen vaan sen käyttäjän vuorovaikutusta hyödyntävät ominaisuudet ovat peräisin Matplotlibistä. Kaikki vertailtavat kirjastot sisältävät oletuksena zoomaus-, panorointi- ja näkymän resetointi -ominaisuudet. Plotly eroaa toisista kirjastoista sillä, että siinä on mukana oletuksena hover-toiminto, jonka avulla pystyy tarkastelmaan kuvaajan arvoja viemällä kursorin kuvaajan pisteiden päälle.

Kirjastoista Plotly ja Bokeh tarvitsevat selaimen visualisointia varten, Matplotlib ja Seaborn puolestaan käyttävät käyttöjärjestelmän ikkunoita eivätkä tarvitse

toisia sovelluksia visualisoinnissa. Selain tuki tosin mahdollistaa Plotlyn ja Bokehin käyttöä web-kehityksessä.

Taulukko 1 vertailutaulukossa viitattavalla kustomoitavuutella tarkoitetaan sitä pystyykö kirjaston ominaisuuksia mukauttamaan erilaisiin käyttötapauksiin. Tällä tarkoitetaan esimerkiksi ohjelman graafisen käyttöliittymään tehtävillä muutoksilla ja lisäyksillä, kuten graafisen painikkeen tai liukusäätimen (engl. slider) lisäämisellä, tai uusien interaktiivisten toimintoja tuomisella sekä oletustoimintojen muokkaamisella. Seaborn on kirjastoista ainoa, joka ei mahdollista suoraan kustomoida käyttöliittymää tai sen interaktiivisia ominaisuuksia. Mikäli käyttökokemusta halutaan muokata tai ominaisuuksia lisätä, pitää muutokset tehdä Seabornin käyttämään Matplotlibiin.

Taulukko 1. Kirjastojen vertailutaulukko.

| Kirjaston ominaisuudet | Matplotlib | Plotly | Bokeh | Seaborn |
|---|---|---|---|---|
| Interaktiivinen käyttö | Kyllä | Kyllä | Kyllä | Osittain |
| Web-käyttöliittymä | Ei | Kyllä | Kyllä | Ei |
| Kustomoitavuus | Kyllä | Kyllä | Kyllä | Osittain |
| Pandas-dataframe | Ei | Kyllä | Kyllä | Kyllä |
| Pythonin Dictionary | Ei | Kyllä | Kyllä | Kyllä |
| NumPy-taulukko | Kyllä | Kyllä | Kyllä | Kyllä |
| Tukee 3D-grafiikkaa | Kyllä | Kyllä | Osittain | Ei |
| Esteettinen personoitavuus | Kyllä | Kyllä | Kyllä | Osittain |

Mikäli visualisoitava data on formatoitu helposti käsiteltävää numerosarjamaista taulukkoa hienojakoisempaan muotoon, kuten dataframe-rakenteeksi, jossa tieto on pyritty hahmottamaan ihmiselle helposti ymmärrettävään muotoon ja tällöin se myös yleensä sisältää tekstiotsikoita, on Bokeh, Plotly ja Seaborn kirjastojen datan käsittelyllä etunsa suhteessa Matplotlibiin. Edellä mainitut kirjastot, jotka tukevat dataframe-formaattia, niiden visualisoinnista vastaavat funktiot pystyvät käyttämään tietolähteenään suoraan Pandaksen dataframe-olioita. Matplotlibin funktioiden parametrit ottavat argumentteina vastaan

ainoastaan pelkkiä numeroarvoja sisältäviä listoina tai NumPy-taulukkoina. [34] Myös Pythonin natiivi dictionary-formaatti on hyvin yhteensopiva kirjastojen kanssa Matplotlibiä lukuunottamatta. Matplotlibin lisäksi kaikki vertailussa huomioidut kirjastot tukevat NumPyn taulukoita.

Taulukko 1 verratuista kirjastoista ainoastaan Matplotlib ja Plotly tukevat 3D-visualisointia konkreettisesti hyödyllisellä tavalla. Plotly on 3D-visualisoinnissa kirjastoista saumattomin, koska kolmiulotteisessa renderöinnissä tarvitut diagrammit ja toiminnot ovat sisällytetty itse kirjastoon. Matplotlib tarvitsee erillisen mplot3d-laajennoksen kyseiseen tarkoitukseen. Bokeh puolestaan ei sisällä natiivia 3D-toiminnallisuutta vaan tarvittavat toiminnot pitää tuoda toisista kirjastoista esim. VisJS-kirjastosta, ja tästä huolimatta Bokehin hyöty 3D-visualisoinnissa on hyvin vähäinen.

Kaikki keskenään verratut kirjastot tukevat jonkin asteista diagrammien esteettistä ehostamista ja personointia. Matplotlib, Plotly sekä Bokeh mahdollistavat muokata vapaasti diagrammien visuaalista ilmettä, kun taas Seaborn sisältää valmiita ratkaisuja, teemoja, joista diagrammien ilmettä pystyy vaihtamaan. Tässä opinnäytetyössä ei keskitytty siihen, miten varsinainen personointi tehdään eri kirjastoilla, joten suoraa vertailua ei pystytä kirjastojen eroavaisuuksista tekemään liittyen esteettisen personnoin helppouteen ja monipuolisuuteen.

Taulukko 1 pohjalta voidaan todeta, että Plotly ja Bokeh tarjoavat kirjastoista kattavimman kokonaisuuden sillä erotuksella, että Bokehista puuttuu ominaisuudet 3D-visualisointia varten. Matplotlib on myös laaja kirjasto, jonka datan käsittelyä hankaloittaa tuettujen dataformaattien rajallinen määrä. Seaborn puolestaan soveltuu helppoon ja monipuoliseen visualisointiin valmiiksi esteettisesti miellyttävillä asetuksilla, mutta on ominaisuuksiltaan muuten suppea.

# 4 Sovellus

Tutkituista kirjastoista Matplotlib soveltui parhaiten tässä opinnäytetyössä toteutetun visualisointityökalun suunnitteluun, koska kirjaston API osoittautui riittävän joustavaksi, jotta kirjaston käyttöä pystyttiin räätälöimään toimeksiannon tarpeiden mukaiseksi. Toinen ratkaiseva tekijä oli, että käytetty kirjasto käyttää verkkoselaimesta riippumatonta ikkunaympäristöä käyttöliittymänä, eikä Bokehin ja Plotlyn kaltaisten web-käyttöliittymäkirjastojen käyttämälle backend-frontend-arkkitehtuurille ollut toteutuksen kannalta erityistä tarvetta. Vaikka sovelluksen olisi pystynyt toteuttamaan edellä mainituilla kirjastoilla, vaikutti Matplotlibin valintaan tässä asiassa henkilökohtainen mieltymys. Myöskään muiden kirjastojen datan käsittelyä helpottaville funktioille tai muille erityisominaisuuksille ei ollut tarvetta, sillä sovellusta tullaan käyttämään helposti käsiteltävän raakadatan piirtämisessä viivadiagrammille eli suhteellisen vaatimattomaan käyttöön, jolloin suurintaosaa kirjastojen erityisominaisuuksista ei tultaisi toteutuksessa hyödyntämään, koska tarvittavat perustoiminnot löytyvät kaikista Taulukko 1 kirjastoista.

Sovelluksen suunnittelun lähtökohtana oli, että käyttäjä pystyisi visualisoimaan spektridataa suoraan tiedostosta. Visualisointikirjastoja tyypillisesti käytetään niin, että käyttäjä kirjoittaa kuvaajien arvot suoraan ohjelman koodiin, jonka jälkeen ohjelma piirtää arvoja vastaavan diagrammin visuaalista havainnointia varten. Loppukäyttäjiä huomioiden eli suun terveyden huollon toimijoita, joilla ei oletetusti ole tarvittavaa osaamista visualisointikirjastojen käytöstä tai ohjelmoinnista yleisesti, piti sovellus suunnitella käytettäväksi graafisen käyttöliittymän kautta. Tämän vuoksi kirjaston toimintarakennetta piti muuttaa sekä sovelluksen käyttöliittyymään lisätä tarvittavat graafiset elementit ja apuohjelmat, jotta sovellusta pystyttäisiin käyttämään ilman, että käyttäjän tarvitsisi kirjoittaa tai ymmärtää ohjelmiston koodia.

## 4.1 Python ja käytetyt kirjastot

Sovellus toteutettiin käyttämällä Pythonin 3.9-versiota. Se toimii luotetusti kyseisellä versiolla, mutta se toimii testatusti myös muilla Python 3:n versioilla. Toteutuksessa on käytetty Matplotlibin lisäksi muita Pythonin kirjastoja. Kaikki sovelluksen käyttämät kirjastot löytyvät lueteltuna alla:

- Matplotlib v.3.5.1
- Pandas
- Openpyxl
- Tkinter
- os
- mplcursors
- PyQt5.

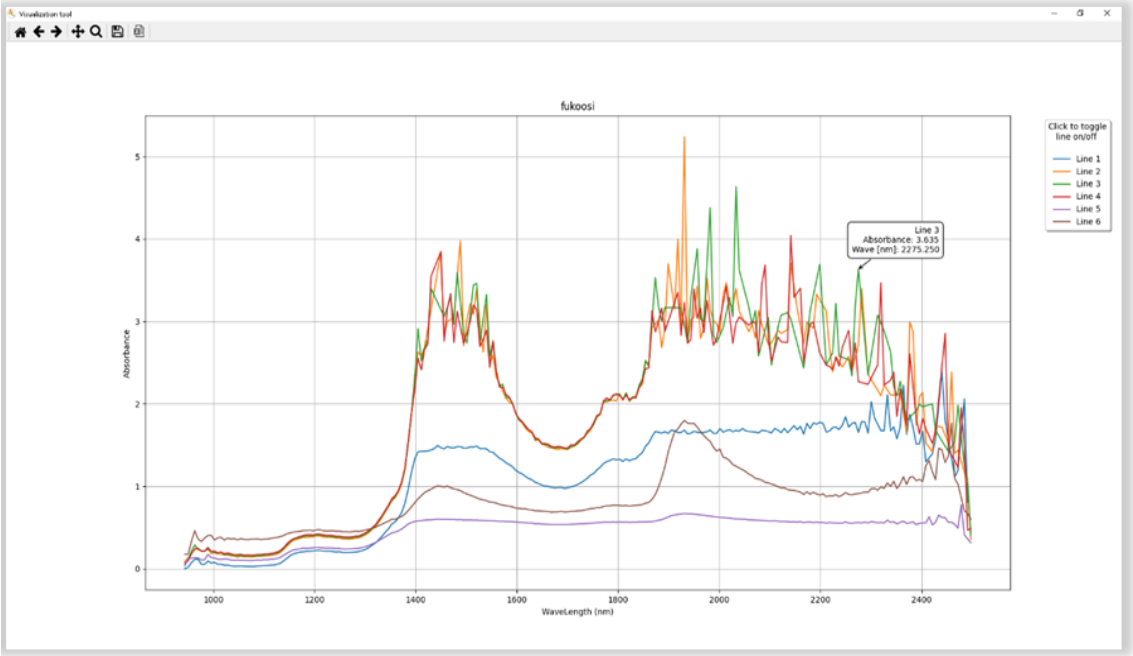
Sovellus kehitettiin käyttämällä Matplolibin 3.5.1-versiota. Kyseinen versio käyttää 1.5-versiossa esiteltyä rcParams-muuttujaa, jonka oletustietue ”toolbar” muutetaan sovelluksen Liite 1 pääohjelmassa arvoksi ”tool manager”. Tällä saadaan käyttöön työkalurivin hallinnoimista varten toolmanager-muuttuja (Liite 1 koodissa tm-muuttuja), jonka avulla Kuva 1 käyttöliittymän työkaluriviin pystyy lisäämään tai poistamaan painikkeita. Sovelluksen käynnistyessä Pythonin konsoli ilmoittaa, että rcParams-muuttujaa käsitellään kokeellisena ominaisuutena ja siihen liittyvät käytännöt saattavat muuttua myöhemmissä Matplotlibin versioissa. Tämän vuoksi sovelluksen koodi toimii luotettavasti Matplotlibin versioiden 1.5–3.5.1 kanssa.

Pandas-kirjastoa käytetään sovelluksessa Excel-tiedostojen lukemista sekä kyseisten funktioiden palauttaman dataframe-tietorakenteen käsittelyä varten. Jotta Pandas-funktiot pystyvät lukemaan Excelin .xlsx-tiedostoja, pitää Python-ympäristöön asentaa myös Openpyxl-kirjasto. Pythonin mukana tuleva Tkinter on graafisten käyttöliittymien luomiseen käytettävä kirjasto. Sen avulla toteutettiin sovelluksen Liite 3 filepath-apuohjelma, jolla käyttäjä pystyy valitsemaan visualisoitavan Excel-tiedoston graafisen tiedostonäkymän kautta.

Pythonin mukana tulevaa os-kirjastoa käytetään Kuva 2 työkaluriviin lisätyn painikkeen kuvan tiedostopolun rakentamista varten.

Kuvaajien tarkastelua varten sovellus käyttää mplcursors-kirjastoa. Sen avulla käyttäjä pystyy helposti näkemään kuvaajan x- ja y-koordinaatit kursorin kärjen osoittamasta kohdasta. Kuva 1 käyttöliittymässä havainnolistetaan kyseistä mplcursors-toimintoa. PyQt5 on Tkinterin tavoin graafisten käyttöliittymien luomiseen suunnattu kirjasto. Sitä käytetään sovelluksessa Matplotlibin grafiikkaliittymänä, koska mplcursors on toimii responsiivisesti kyseisen käyttöliittymäkirjaston kanssa.

## 4.2 Käyttöliittymä




Kuva 1. Sovelluksen käyttöliittymä ja sen visualisoitu graafi.

Sovelluksen käyttö tapahtuu Kuva 1 mukaisen näkymän kautta. Käyttöliittymä sisältää vain yhden näkymän, ja se sisältää kaikki sovelluksen käytön kannalta tarvittavat toiminnot. Kanvaasiksi sanotaan näkymän valkoista-aluetta, johon sovellus renderöi käyttäjäsyötteen. Diagrammin kuvaajat piirretään kanvaasissa sijaitsevien x- ja y-akseleiden rajaaman suorakulmaisen alueen sisäpuolelle.

Akselien koordinaattien asteikkovälit asettuvat automaattisesti diagrammia luodessa, ja ne skaalautuvat diagramminäkymää zoomatessa. Diagrammin päällä sijaitseva otsikkoteksti on visualisoitavan tiedoston nimi.

Kuva 1 oikeassa laidassa sijaitseva nelikulmion muotoinen legend-kuvio toimii kuvaajien selostustauluna. Se sisältää visualisoitujen kuvaajien järjestyksen sekä sitä vastaavan väritunnisteen. Legendiin on sisällytetty toiminto, jonka avulla käyttäjä pystyy piilottamaan tai tuomaan takaisin näkyviin diagrammin kuvaajia painamalla kuvaajaselosteen väriviivasta.

Kuva 2. Työkalurivin painikkeet.

Näkymän yläosan harmaa suorakulmainen alue on työkalurivi, joka sisältää Kuva 2 mukaiset painikkeet. Vasemman laidan Home-painike palauttaa näkymän kuvaajien asennon ja akseleiden vaihteluvälin, mikäli akselinäkymää on muutettu tai suurennettu. Home-painikkeen vieressä näkymien navigointipainikkeet, joilla pystyy siirtymään zoomattujen akselinäkymien väliltä. Keskimmäisen eli ristinuolipainikkeen avulla pystyy hiiren vasemmalla painikkeella liikuttelemaan diagrammin akseleita. Painamalla hiiren oikeanpuoleista painikketa pystyy näkymää zoomamaan hiiren liikkeen mukaiseen suuntaan. Hiiren rullalla puolestaan pystyy zoomamaan kursorin kohdistamaan kohtaan. Työkalurivin suurennuslasipainikkeella pystyy valitsemaan alueen, johon käyttäjä haluaa zoomata. Sen oikealla puolella olevalla tallennuspainikkeella pystyy viemään kanvaasinäkymän esim. PNG-, JPEG-, tai PDF-muotoon.

Jotta kirjaston hyödyntämä sovellus toimisi pelkän graafisen käyttöliittymän kautta, piti työkaluriviin lisätä ylimääräinen readfile-painike (Kuva 2 oikeanpuoleisin painike) tiedoston valintaa ja lukua varten. Sitä painamalla käynnistyy Liite 3 filepath-funktion suorittama tiedostonäkymäohjelma Excel-tiedoston valintaa varten. Kyseinen ohjelma tarkistaa myös, onko valittu tiedosto muotoa .xlsx tai .xls. Jos valinta on virheellinen Liite 3 error_message-ohjelma

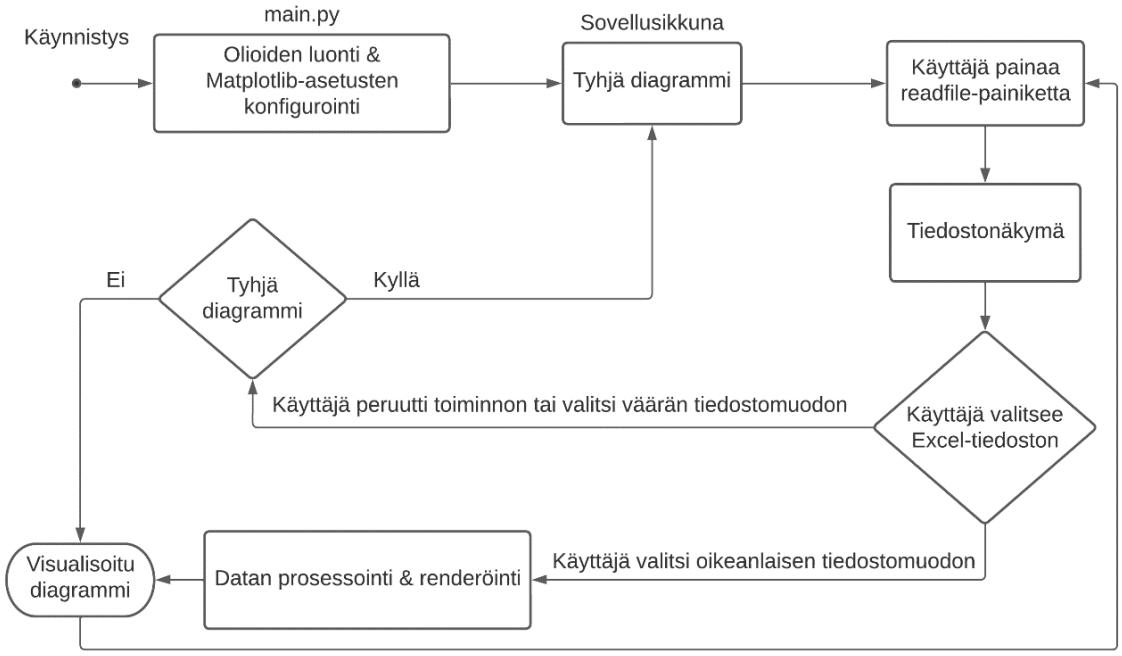
ilmoittaa siitä virheviestillä ja palauttaa tyhjän tiedostopolun. Mikäli ohjelma hyväksyy tiedoston valinnan, se palauttaa tiedoston polun, jonka jälkeen Liite 2 draw_plot-funktio piirtää käyttäjä syötteen diagrammiin.

### 4.3 Sovelluksen toiminta

Sovelluksen toimintaa havainnollistetaan pääpiirteittäin Kuvio 1 sekvenssikaaviossa. Sovelluksen käynnistyessä Liite 1 pääohjelma (main.py) alustaa sovellusikkunan fig-olion, diagrammin piirtämiseen tarvittavan ax-olion sekä diagrammin tilojen (engl. states) muistin hallintaan tarkoitetun toolbar-olion. Samassa ohjelmassa työkaluriviin lisätään tiedoston lukuun (Kuva 2 oikeanpuoleisin painike) tarkoitettu readfile-painike, ja sille välitetään luonnin yhteydessä kaikki edellä mainitut oliot. Kyseisen painikkeen alustusparametrien tarkoituksena on välittää tarvittavat oliot Liite 2 draw_plot-funktiolle diagrammin päivittämistä varten. Jotta painikkeita pystyy lisäämään työkaluriviin, pitää niille kirjoittaa luokka. Liite 2 sisältää openfile-painiketta varten OpenFileButton-luokan. Pääohjelmassa myös muutetaan Matplotlib-kirjaston käyttämiä oletusasetuksia esim. poistetaan työkalurivistä tarpeettomia painikkeita sekä asetetaan sovellus käynnistymään fullscreen-tilassa. Lopuksi pääohjelma avaa sovellusikkunan, jossa on näkyvissä tyhjä diagrammi.

Käyttäjän painattaessa readfile-painiketta käynnistyy draw_plot-ohjelma, ja sen myötä käynnistyy myös Liite 3 filepath-ohjelma, joka avaa käyttöjärjestelmän tiedostonäkymä Excel-tiedoston valintaa varten. Mikäli käyttäjä valitsee Excel-tiedoston, jonka data on formatoitu sovelluksen hyväksymällä (Kuvio 1) tavalla, filepath palauttaa draw_plotille valitun tiedoston polun, minkä jälkeen prosessi etenee datan käsittelyvaiheeseen, jossa tiedoston data välitetään Matplotlibin plottausfunktiolle. Samassa vaiheessa alustetaan myös kuvaajien tarkasteluun tarkoitetut lisätoiminnot. Prosessin lopuksi tiedoston sisältämät arvot renderöityvät kanvaasin diagrammille viivakuvaajina. Onnistuneen visualisointiprosessin jälkeen käyttäjä pystyy vaihtamaan tiedostoa painamalla readfile-painiketta uudelleen. Jos käyttäjä valitsee väärän tiedostotyypin tai Excel-tiedoston data on väärällä tavalla formatoitu, Liite 3 error_message-

funktio avaa ponnahdusikkunan ja ilmoittaa mahdollisesta virheestä. Käyttäjän suljettua ponnahdusikkunan visualisointiprosessi keskeytyy, ja diagrammi pysyy muuttumattomana eli on tyhjä tai se sisältää aikaisemman piirrosprosessin kuvaajat (tai yhden kuvaajan).




Kuvio 1. Sekvenssikaavio sovelluksen toiminnasta.

## 4.4 Excel-tiedostojen rakenne

Sovellus käyttää

Taulukko 2 mukaista tietorakennetta diagrammin visualisoinnissa. Tiedoston ensimmäinen eli ylin rivi voi sisältää kirjaimia ja muita merkkejä numeroiden lisäksi, ja sovellus käyttää kyseisen rivin tietoja x- ja y-akselien tunnisteina. Ylimmän rivin kahden ensimmäisen tietoa sisältävien solujen tunnisteet asetetaan diagrammin akseleiden otsikoiksi, joista ensimmäinen (

Taulukko 2 ”Wavelength [nm]”) toimii x-akselin tunnisteena ja toinen (

Taulukko 2 ”Absorbance”) y-akselin tunnisteena. Kaikkien muiden rivien tulee sisältää numeerisia arvoja, mutta ne voivat sisältää miinusmerkillä (-) ilmaistuja negatiivisia lukuja. Tyhjät solut ovat arvoltaan nolla. Arvojen desimaalierottimena voi käyttää pilkkua tai pistettä. Ohjelma muuttaa kunkin sarakkeen taulukoksi (engl. array), ja se muodostaa kahdesta vierekkäisestä sarakkeesta taulukkoparin yksittäistä kuvaajaa varten, joista ensimmäinen (vasemmanpuoleinen) toimii x-koordinaattien pistearvoina ja toinen (oikea) y-koordinaattien pistearvoina. Tiedostossa voi olla tyhjiä sarakkeita ja niiden sijainnilla tai lukumäärällä ei ole väliä, koska kaikki tyhjät sarakkeet poistetaan ohjelman ajon aikana. Dataa sisältäviä sarakkeita tulee olla parillinen määrä, muuten ohjelma ei hyväksi tiedostoa. Pariton määrä sarakkeita aiheuttaisi piirto-ohjelman for-silmukassa Index out of range -virheen ja siitä ohjelma ilmoittaa käyttäjälle asianmukaisella virheilmoituksella. Taulukkoparien määrä vastaa kuvaajien lukumäärää diagrammissa, eikä niiden lukumäärää ole sovelluksessa rajoitettu eli taulukkopareja voi olla tiedostossa rajattomasti. Sarakkeissa voi olla myös tyhjiä rivejä ennen lukujono alkua, mutta tyhjät rivit lukujonon sisällä näkyy kuvaajassa katkonaisina viivoina.

Taulukko 2. Esimerkki Excel-tiedoston rakenteesta.

| Wavelength [nm] | Absorbance | | Wavelength [nm] | Absorbance | | Wavelength [nm] | Absorbance |
|---|---|---|---|---|---|---|---|
| 943,84 | -0,049262095 | | 943,84 | -0,041628592 | | 943,84 | -0,029141692 |
| 950,37 | -0,031296423 | | 950,37 | -0,035638985 | | 950,37 | -0,039215723 |
| 956,9 | 0,009565929 | | 956,9 | 0,003752383 | | 956,9 | 0,006120725 |
| 963,43 | 0,036597889 | | 963,43 | 0,029744895 | | 963,43 | 0,036392123 |
| 969,96 | 0,025948594 | | 969,96 | 0,013616525 | | 969,96 | 0,013446323 |
| 976,48 | -0,014002085 | | 976,48 | -0,008176333 | | 976,48 | 0,004761203 |
| 983,01 | 0,038142757 | | 983,01 | 0,029512553 | | 983,01 | 0,014346682 |
| 989,53 | 0,004756234 | | 989,53 | 0,001876354 | | 989,53 | 0,01484768 |
| 996,06 | 0,01518493 | | 996,06 | 0,005900594 | | 996,06 | -0,003214383 |
| 1002,58 | 0,018561774 | | 1002,58 | 0,011555791 | | 1002,58 | 0,013785017 |
| 1009,1 | 0,004287946 | | 1009,1 | 1,22702E-05 | | 1009,1 | -0,003275427 |
| 1015,62 | 0,01047765 | | 1015,62 | 0,012336891 | | 1015,62 | 0,008121487 |
| 1022,14 | 0,003578392 | | 1022,14 | 0,001804896 | | 1022,14 | 0,005356122 |

## 4.5 Interaktiiviset lisätoiminnot

Tiedoston lukemisen ja kanvaasin renderöintiä varten lisätyn readfile-painikkeen lisäksi sovellukseen on lisätty kaksi interaktiivista ominaisuutta

käyttökokemuksen ja datan tutkimisen parantamiseksi. Opinnäytetyön 4.1 Python ja käytetyt kirjastot -osiossa mainittu ja lyhyesti selostettu mplcursors-kirjasto lisää tooltip-toiminnon diagrammin kuvaajille. Sen avulla käyttäjä saa kuvaajan pisteen arvot näkyviin kursorin osoittamasta kohdasta. Toiminnon seurauksena diagrammiin ilmestyy kursorin lähettyville suorakulmion muotoinen nuolenosoittama selostetaulu (tooltip), jossa näkyy kuvaajan tiedot, xy-akseleiden otsikot sekä kuvaajapisteen x- ja y-koordinaattien arvot. Koodissa mplcursors otetaan käyttöön Liite 2 draw_plot-funktiossa välittämällä kuvaajat (tai kuvaajan) sisältävän diagram-olion kirjaston mplcursors.cursor-funktiolle. Diagrammin kursoriin on sidottu kaksi mplcursors-toimintoa, joista Liite 2 tooltip-oliota käytetään hooverointitapahtuman (engl. hover event) avulla. Tapahtuma saa tooltipin ilmestymään heti, kun kursori on viety kuvaajan päälle. Jos kyseinen tooltip ei ilmesty, vaikka kursorin kärki on kuvaajan päällä, se pitää kohdistaa uudelleen painamalla hiiren painikkeesta kursorin osoittaessa diagrammin tyhjää tilaa vasten. Toinen toiminto, joka on sidottu Liite 2 marker-olioon, mahdollistaa useiden tooltip-taulujen kiinnittämisen diagrammille esim. vertailua tai tärkeiden kohtien merkitsemistä varten. Kyseinen tooltip ilmestyy osoittamalla kuvaajaan ja painamalla hiiren vasenta painiketta. Funktion ”draw_plot” koodi sisältää myös pohjan molempien tooltip-olioiden ominaisuuksien konfigurointia ja personointia varten.

Toinen sovellukseen lisätty interaktiivinen toiminto on Kuva 1 diagrammin viereisen Legend-taulun suodatustoiminto, jonka avulla pystyy piilottamaan tai tuomaan takaisin näkyviin yhden tai useamman diagrammin kuvaajan. Toiminto tapahtuu painamalla taulun sisältämän kuvaajan väriviivatunnisteesta. Kuvaajan piiloutumisen myötä kuvaajaa vastaava väriviiva harmaantuu, vastaavasti sen väri palautuu, kun kuvaaja tuodaan takaisin näkyviin. Siinä draw_plot-funktion vaiheessa, missä kuvaajat luodaan for-silmukassa välittämällä Matplotlibin plottausfunktiolle, luodaan kuvaajan olio ”line”-nimiseen muuttujaan. Kaikki line-muuttujat yhdistetään niitä vastaavan Legend-viivan (Liite 2 legline-muuttuja) arvopariksi dictionary-rakenteeseen. Jotta Legend-tauluun pystyi lisäämään hiiren painallus -toiminnon, sille piti tehdä Liite 2 onpick-funktio, ja se piti yhdistää Matplotlib API:n pick_event-tapahtumaan. Käyttäjän painettaessa

Legend-viivaa ohjelma suorittaa onpick-funktion, jossa se etsii painettua viivaa vastaavan kuvaajan ja muuttaa sen näkyvyyden renderöimällä diagrammin ilman kyseistä kuvaajaa.

# 5 Yhteenveto

Tämän työn tavoitteena oli suunnitella ja toteuttaa Python-ohjelmointikielellä sovellus lähi-infrapunaspektrin visualisointia varten. Työ aloitettiin tutustumalla lähi-infrapunan spektroskooppisiin menetelmiin sekä Pythonin visualisointikirjastoihin. Esiselvitystyön ensimmäinen tavoite oli hankkia tarvittavat tietotaidot lähi-infrapunan spektrianalyysistä ja sen käyttämistä visualisointimenetemistä. Toinen tavoite oli löytää sopiva visualisointikirjasto sovelluksen toteutusta varten. Selvitystyön tulokset osoittivat, että spektroskooppisissa analyyseissä mitataan valon aallonpituutta absorption funktiona ja sen visualisointiin soveltuu parhaiten viivakaaviomenetelmä. Tutkituista kirjastoista Matplotlib osoittautui riittävän joustavaksi, jotta sen käyttöä pystyi räätälöimään aiottuun käyttötapaukseen. Kirjasto ei myöskään tarvitse selainta toimiakseen.

Työssä toteutettiin yleiskäyttöinen visualisointisovellus, jota loppukäyttäjä pystyy käyttämään ilman Python-ohjelmointikielen tuntemusta graafisen käyttöliittymän kautta. Sovelluksen avulla käyttäjä pystyy visualisoimaan yksinkertaiseen muotoon koostettua dataa valitsemastaan Excel-tiedostosta. Kehitetyllä ratkaisulla suun terveyden huollon ammattilaiset pystyvät mallintamaan lähi-infrapunaspektriä visuaalisesti havainnoitavaan muotoon. Visualisoidun spektridatan avulla tehostetaan suusyövän varhaista diagnosointia, jolloin myös potilaan ennuste paranee.

Sovelluksen kehitystä pystyisi jatkamaan tuomalla siihen spektrianalyysiä helpottavia  toimintoja. Yksi tällainen toiminto voisi olla suodatustoiminto spektrin aallonpituuksille, jonka avulla käyttäjä pystyisi suodattamaan diagrammin käyristä tarpeettomat aallonpituudet pois. Datan käsittelyyn voisi myös lisätä uusia funktioita visualisointia varten. Käyttäjä pystyisi esimerkiksi tiedoston valinnan yhteydessä valitsemaan, minkälaiselle diagrammille tai kaaviolle ohjelma visualisoi datan. Jatkokehityksessä tulisi huomioida Matplotlibin API:n mahdollisia rajoitteita, sillä se ei välttämättä aina tue aiottua

toteutustapaa ja mahdollinen vaihtoehtoinen ratkaisu ei välttämättä ole käytöltään intuitiivinen.

# Lähteet

[1] Saranya K, “Data Visualization: Importance and Benefits | Bold BI,” Jul. 22, 2019. https://www.boldbi.com/blog/data-visualization-importance-and-benefits (accessed Jun. 03, 2022).

[2] Coursera, “What Is Python Used For? A Beginner’s Guide | Coursera,” May 26, 2022. https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python (accessed Jun. 01, 2022).

[3] M. Kallioinen, “Lähi-infrapunaspektroskopia (NIRS) aivokudoksen happeutumisen seurannassa,” *Finnanest*, vol. 45, no. 4, pp. 330–336, 2012, Accessed: Jan. 30, 2022. [Online]. Available: http://www.finnanest.fi/files/kallioinen_spektro.pdf

[4] T. Kuusela, “Mihin käytetään infrapunavaloa?,” *https://www.aka.fi/tietysti/kysy-tieteesta/mihin-kaytetaan-infrapunavaloa/#a73bbe1b*, Mar. 04, 2015.

[5] L. J., M. D. L., and V. H. Y., “Near-infrared spectroscopy applications in pharmaceutical analysis,” *Talanta*, vol. 72, no. 3. Elsevier, pp. 865–883, May 15, 2007. doi: 10.1016/j.talanta.2006.12.023.

[6] A. Sakudo, “Near-infrared spectroscopy for medical applications: Current status and future perspectives,” *Clinica Chimica Acta*, vol. 455. Elsevier B.V., pp. 181–188, Apr. 01, 2016. doi: 10.1016/j.cca.2016.02.009.

[7] S. Salminen, V. Sinikallio, and V. Kämppi, “Spektroskopia: johdanto,” Helsinki, Mar. 2021. Accessed: Mar. 07, 2022. [Online]. Available: https://www2.helsinki.fi/fi/tiedekasvatus/spektroskopia-tutuksi

[8] KPM Analytics, “What is NIR Spectroscopy and How Does It Work,” *https://www.kpmanalytics.com/news-events-stories/what-is-nir-spectroscopy-and-how-does-it-work*, 2022.

[9] Metrohm Middle East FZC, “Near-Infrared-Spectroscopy-(NIRS)-An-Overview-of-Benefits-and-Applications,” May 2021, Accessed: Feb. 28, 2022. [Online]. Available: https://www.azom.com/article.aspx?ArticleID=20445

[10] Guided Wave, “Advantages of Dispersion Spectrometers over FT-NIR Analyzers,” 2022. https://guided-wave.com/nir-process-spectrometers-ft-nir-dispersive-dual-beam-dg-nir-analyzers/ (accessed Feb. 28, 2022).

[11] Reitz Kenneth and Real Python, “Scientific Applications — The Hitchhiker’s Guide to Python.” https://docs.python-guide.org/scenarios/scientific/ (accessed Mar. 07, 2022).

[12] “10 Python Data Visualization Libraries to Win Over Your Insights,” *projectpro*, Jan. 31, 2022. https://www.projectpro.io/article/python-data-visualization-libraries/543#mcetoc_1fonchs0pf (accessed Mar. 07, 2022).

[13] Taylor-Morgan, “Precision data plotting in Python with Matplotlib | Opensource.com,” May 26, 2020. https://opensource.com/article/20/5/matplotlib-python (accessed Mar. 07, 2022).

[14] “Examples — Matplotlib 3.5.1 documentation.” https://matplotlib.org/stable/gallery/index.html (accessed Mar. 09, 2022).

[15] “matplotlib.pyplot — Matplotlib 3.5.1 documentation.” https://matplotlib.org/3.5.1/api/_as_gen/matplotlib.pyplot.html (accessed Apr. 03, 2022).

[16] “Comparing Python Data Visualization Tools: Matplotlib vs Seaborn.” https://analyticsindiamag.com/comparing-python-data-visualization-tools-matplotlib-vs-seaborn/ (accessed Mar. 29, 2022).

[17] N. Leong, “4 Reasons Why Plotly Is The Best Visualization Library | by Nicholas Leong | Towards Data Science,” Jul. 27, 2021.

https://towardsdatascience.com/4-reasons-why-plotly-is-the-best-visualization-library-18c27de05b95 (accessed Mar. 12, 2022).

[18] “Line charts with Python.” https://plotly.com/python/line-charts/ (accessed Apr. 11, 2022).

[19] A. Tomar, “Dash for Beginners : Python Dashboards | by Anmol Tomar | Towards Data Science,” Mar. 17, 2021. https://towardsdatascience.com/dash-for-beginners-create-interactive-python-dashboards-338bfcb6ffa4 (accessed Mar. 12, 2022).

[20] “Templates.” https://plotly.com/python/templates/ (accessed Mar. 12, 2022).

[21] P. Iacomi, “Plotly vs. Bokeh: Interactive Python Visualisation Pros and Cons | Paul Iacomi,” Jun. 07, 2020. https://pauliacomi.com/2020/06/07/plotly-v-bokeh.html (accessed Mar. 14, 2022).

[22] “Bokeh documentation — Bokeh 2.4.2 Documentation.” https://docs.bokeh.org/en/2.4.2/# (accessed Mar. 14, 2022).

[23] freeCodeCamp.org, “Charting the waters: between Bokeh and D3,” Jan. 04, 2018. https://www.freecodecamp.org/news/charting-the-waters-between-bokeh-and-d3-73b3ee517478/ (accessed Mar. 15, 2022).

[24] “Bokeh - Server.” https://www.tutorialspoint.com/bokeh/bokeh_server.htm (accessed Mar. 15, 2022).

[25] “Running a Bokeh server — Bokeh 2.4.2 Documentation.” https://docs.bokeh.org/en/latest/docs/user_guide/server.html#userguide-server (accessed Mar. 15, 2022).

[26] “Bokeh - WebGL.” https://www.tutorialspoint.com/bokeh/bokeh_extending_webgl.htm (accessed Mar. 15, 2022).

[27] “Surface3d.” https://demo.bokeh.org/surface3d (accessed Mar. 16, 2022).

[28] “Developing with JavaScript — Bokeh 2.4.2 Documentation.” https://docs.bokeh.org/en/latest/docs/user_guide/bokehjs.html (accessed Mar. 16, 2022).

[29] “Bokeh Vs Plotly: Which One Is Better In 2022? - Buggy Programmer,” Feb. 22, 2022. https://buggyprogrammer.com/bokeh-vs-plotly-which-one-is-better-in-2022/ (accessed Mar. 21, 2022).

[30] M. Waskom, “seaborn: statistical data visualization,” *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, Apr. 2021, doi: 10.21105/JOSS.03021.

[31] Srivastava Ishan, “Difference Between Matplotlib VS Seaborn - GeeksforGeeks,” Nov. 02, 2021. https://www.geeksforgeeks.org/difference-between-matplotlib-vs-seaborn/ (accessed Mar. 29, 2022).

[32] Waskom Michael, “Three common seaborn difficulties | by Michael Waskom | Medium,” Feb. 22, 2021. https://michaelwaskom.medium.com/three-common-seaborn-difficulties-10fdd0cc2a8b (accessed Mar. 31, 2022).

[33] D. Misal, “Comparing Python Data Visualization Tools: Matplotlib vs Seaborn,” Jul. 02, 2019. https://analyticsindiamag.com/comparing-python-data-visualization-tools-matplotlib-vs-seaborn/ (accessed Apr. 02, 2022).

[34] V. Kaushik, “Plot line graph from NumPy array - GeeksforGeeks,” Dec. 17, 2021. https://www.geeksforgeeks.org/plot-line-graph-from-numpy-array/ (accessed Apr. 15, 2022).

Liite 1. Pääohjelma – main.py.

```
import matplotlib as mpl  # pip install matplotlib==3.5.1
import matplotlib.pyplot as plt
from lib.GUI.toolbar import OpenFileButton
import os
# Also, install openpyxl package with 'pip install openpyxl'.

mpl.rcParams["toolbar"] = 'toolmanager'
mpl.use('Qt5Agg')  # pip install PyQt5

print(mpl.__version__)

fig, ax = plt.subplots()

# chart configurations.
ax.grid()

# window configurations.
manager = plt.get_current_fig_manager()
manager.window.showMaximized()
manager.set_window_title('Visualization tool')

# adds custom button to the toolbar.
tm = fig.canvas.manager.toolmanager
toolbar = manager.canvas.manager.toolbar
tm.add_tool('openfile', OpenFileButton, diagram=ax,
            figure=fig, toolbar=toolbar)

button = tm.get_tool('openfile')
# Sets icon to the button.
button.image = "{0}/assets/excel-icon.svg".format(os.getcwd())
# adds button to the toolbar.
manager.canvas.manager.toolbar.add_tool(button, 'toolgroup')

# All toolbar default buttons are listed below, uncomment
# or add "#" without quote marks if you want to remove or
# add specific button.
tm.remove_tool('subplots')
tm.remove_tool('help')
# tm.remove_tool('back')
# tm.remove_tool('forward')
# tm.remove_tool('home')

# shortcut configurations.
tm.update_keymap('zoom', 'x')

plt.show()  # Display all open figures.
```

Liite 2. Työkalurivin lisätoiminnot.

```
import mplcursors  # 'pip install
# git+https://github.com/anntzer/mplcursors'.
import pandas as pd  # 'pip install pandas'
from matplotlib.backend_tools import ToolToggleBase
from lib.GUI.readfile import *


# Custom toolbar button for reading excel files.
class OpenFileButton(ToolToggleBase):
    description = 'Choose excel file to read'

    def __init__(self, *args, figure, diagram, toolbar, **kwargs):
        self.toolbar = toolbar
        self.fig = figure
        self.diagram = diagram
        super().__init__(*args, **kwargs)

    def trigger(self, *args, **kwargs):
        draw_plot(self.fig, self.diagram, self.toolbar)


# Executed when OpenFileButton is pressed.
# Used to draw lines and configure charts.
def draw_plot(fig, diagram, toolbar):

    url = filepath()

    if url != '':
        # clear axis.
        diagram.clear()

        # Converts excel file into dataframe.
        # openpyxl engine is needed for reading.
        # .xlsx excel files
        df = pd.read_excel(url, engine='openpyxl')

        # Finds the columns where each value is null
        # (=empty columns).
        empty_cols = [col for col in df.columns if
                      df[col].isnull().all()]
        # Drops the columns where value is null.
        df.drop(empty_cols, axis=1, inplace=True)

        # Sets file name as a chart title.
        title = os.path.basename(url)
        diagram.set_title(os.path.splitext(title)[0])

        # Reads first and second cells on the first line
        # of the dataframe.

        # x-axis title.
        diagram.set_xlabel(df.iloc[:, 0].name)

        # y-axis title.
        diagram.set_ylabel(df.iloc[:, 1].name)
```

```
count = 1  # Tracks the plot line count
lines = []

# Converts every row in each column into array
try:
    for i in range(0, len(df.columns), 2):
        x = df.iloc[:, i]
        y = df.iloc[:, i + 1]

        line, = diagram.plot(x, y, label=f'Line {count}')
        lines.append(line)
        count += 1

except IndexError:
    error_message('Index out of range ERROR',
                  'Excel taulukosta puuttuu sarake')
    return
except ValueError:
    error_message('Invalid value ERROR',
                  'Sarakkeiden rivit sisältävät '
                  'huonoja arvoja')
    return

# Legend info configurations
leg = diagram.legend(bbox_to_anchor=(1.12, 1),
                     loc='upper right',
                     fancybox=True,
                     shadow=True,
                     title='Click to toggle\n    line on/off\n')


lined = dict()
for legline, origline in zip(leg.get_lines(), lines):
    legline.set_picker(5)  # 5 pts tolerance
    lined[legline] = origline

# Event triggered when clicking legend lines.
def onpick(event):
    # on the pick event, find the original line
    # corresponding to the legend proxy line,
    # and toggle the visibility.
    try:
        legline = event.artist
        origline = lined[legline]
        vis = not origline.get_visible()
        origline.set_visible(vis)
        # Change the alpha on the line in the
        # legend so we can see what lines have
        # been toggled.
        if vis:
            legline.set_alpha(1.0)
        else:
            legline.set_alpha(0.2)
        fig.canvas.draw()
    except KeyError:
        pass

fig.canvas.mpl_connect('pick_event', onpick)
```

```
# adds grid to the canvas.
diagram.grid()

# adds tooltip to the cursor.
# tooltip shows information about artist when
# hovering the cursor over it.
tooltip = mplcursors.cursor(diagram, hover=2)
# this tooltip is used for setting markers to the
# plot lines.
marker = mplcursors.cursor(diagram, multiple=True)

# tooltip configurations
@tooltip.connect("add")
def _(sel):
    # Sets background color of tooltips.
    sel.annotation.get_bbox_patch().set(fc="white", alpha=1)
    # Information showed in the tooltip.
    sel.annotation.set_text(
        # shows values in 10 and 3 decimal
        # accuracy.
        f'{sel.artist.get_label()}\n'
        f'{df.iloc[:, 1].name.strip()}:'
        f' {sel.annotation.xy[1]:.10f}'
        f'\n{df.iloc[:, 0].name.strip()}:'
        f' {sel.annotation.xy[0]:.3f}')

# second tooltip for tagging points of the plot.
@marker.connect("add")
def _(sel):
    sel.annotation.get_bbox_patch().set(fc="white", alpha=1)
    sel.annotation.set_text(
        f'{sel.artist.get_label()}\n'
        f'{df.iloc[:, 1].name.strip()}:'
        f' {sel.annotation.xy[1]:.10f}\n'
        f'{df.iloc[:, 0].name.strip()}:'
        f' {sel.annotation.xy[0]:.3f}')

fig.canvas.draw()

# functions below is for resetting view
# positions memory stack
toolbar.update()
tm = fig.canvas.manager.toolmanager
reset = tm.get_tool('viewpos')
reset.clear(fig)
reset.add_figure(fig)
reset.push_current()
reset.update_view()
```

Liite 3. Funktiot tiedoston valitsemista  ja virheden näyttämistä varten.

```
import os
from tkinter import *
from tkinter import filedialog, messagebox


# auxiliary program to help to choose readable file.
def filepath():
    root = Tk()
    root.withdraw()  # used to hide tkinter window.

    current_dir = os.getcwd()
    file_path = filedialog.askopenfilename(parent=root,
                                           initialdir=current_dir,
                                           title='Valitse tiedosto')

    # checks if chosen file is a excel file. If support
    # for .csv filetype is needed, add: file_path[-4:] != '.csv'.
    if file_path[-5:] != '.xlsx' and file_path[-4:] != '.xls' \
            and file_path != '':

        error_message('Filepath Error',
                      'Valittu tiedosto ei ole '
                      'excel-tiedostomuotoa .xlsx tai .xls')
        return ''
    else:
        return file_path  # returns chosen file's path url.


# Error messagebox.
def error_message(title, message):
    messagebox.showerror(title, message)
```