КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ВИСОКИХ ТЕХНОЛОГІЙ

Завідувач кафедри нанофізики конденсованих середовищ

проф. Валерій Антонович Скришевський

Протокол №_____ засідання кафедри

Від «_____»_____2024 р.

Розробка бази даних для N-розмірних патернів аналітів систем "штучний язик"

Випускна кваліфікаційна робота магістра

студента спеціальності 105

Прикладна фізика та наноматеріали

ОП «Високі технології (Прикладна фізика та наноматеріали)»

Поліщука Євгенія Ігоровича

Науковий керівник

доцент кафедри

нанофізики конденсованих середовищ

к.ф.-м.н. **Іванов Іван Іванович**

Оцінка захисту роботи

_____

Київ – 2024 р.

АНОТАЦІЯ

Поліщук Є.І. *Розробка бази даних для N-розмірних патернів аналітів систем "штучний язик"*. – Випускна кваліфікаційна робота магістра за спеціальністю 105 Прикладна фізика та наноматеріали ОП «Високі технології: (Прикладна фізика та наноматеріали)». 62 ст., 25 рис., 17 джерел.

У ході дипломної роботи проведено стислий теоретичний огляд явища люмінесценції, а саме: типи, механізми виникнення, поведінку наночастинок на поверхі речовини, явище SPCE. Розглянуто різні люмінесцентні матеріали. Згадано люмінесцентну спектроскопію, а саме: закон Стокса для люмінесценції, у якому йдеться про «стоксів зсув»; інтеграцію спектрів із зондами; люмінесцентні картки, які по суті є комбінаціями спектрів люмінесценції. Також розглянуто дослідження двох видів ситем «штучний язик», а саме, рідинний (набір полімерних барвників для тестування рідин) та електронний (має 4 датчики, що відповідають різним смакам). Детально розглянуто різні види баз даних, особливо реляційні БД; мову програмування python та її бібліотеки, необхідні для виконання поставленого завдання. Розроблено БД SQLite та програму для отримання відгуків на SQL-запити -- побудову графіків.

Ключові слова: **база даних, MySQL, SQLite, python, pandas, numpy, tkinter, sqlite3, matplotlib, спектр, люмінесценція, наночастинка, люмінесцентна картка, система “штучний язик”.**

ABSTRACT

Polishchuk Y.I. Development of a database for N-dimensional patterns of analytes of "artificial tongue" systems. - Master's thesis on the speciality 105 Applied Physics and Nanomaterials EP "High technologies (Applied physics and nanomaterials)". 62 pp., 25 figs., 17 sources.

In the course of the thesis, a brief theoretical review of the phenomenon of luminescence was carried out, namely: types, mechanisms of occurrence, behavior of nanoparticles on the surface of matter, the phenomenon of SPCE. Various luminescent materials are considered. Luminescence spectroscopy is mentioned, namely: Stokes' law for luminescence, which refers to a "Stokes shift"; integration of spectra with probes; luminescent cards, which are essentially combinations of luminescence spectra. The study of two types of "artificial tongue" systems is also considered, namely, liquid (a set of polymer dyes for testing liquids) and electronic (has 4 sensors corresponding to different tastes). Various types of databases, especially relational databases, are considered in detail; the python programming language and its libraries, which are necessary to perform the given task. A SQLite database and a program for receiving responses to SQL queries - graphing have been developed.

Key words: **database, MySQL, SQLite, python, pandas, numpy, tkinter, sqlite3, matplotlib, spectrum, luminescence, nanoparticle, luminescent card, “artificial tongue” system.**

ЗМІСТ

ВСТУП

Швидкий розвиток наукових досліджень у сучасному світі постійно посилює потребу в ефективних інструментах аналізу та обробки великих обсягів даних. Одним із найбільш перспективних напрямків є аналіз систем "штучного язика", який вимагає глибокого розуміння та обробки великої кількості інформації з різних джерел.

У даній дипломній роботі ми пропонуємо розробку бази даних для зберігання та обробки N-розмірних патернів аналітики систем "штучного язика". Наша основна мета полягає в створенні структурованої бази даних, заснованої на технології MySQL/**SQLite**, що дозволить зберігати дані вимірів багатьох фізичних величин для подальшої аналізу та обробки.

Ця база даних буде включати в себе інформацію, необхідну для побудови різноманітних спектрів, що є ключовим етапом у формуванні люмінесцентних карт. Особлива увага буде приділена оптимізації структури бази даних для ефективного використання ресурсів та максимізації швидкодії системи.

Розробка такої бази даних відіграє важливу роль у подальшому розвитку систем "штучного язика", забезпечуючи необхідну інформаційну підґрунтя для аналізу та вдосконалення їх функціональності. Результати даної роботи можуть мати практичне застосування у різних сферах, від наукових досліджень до технологічних розробок.

1.ТЕОРЕТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ

1.1.Явище люмінесценції

Люмінесценція - це фізичне явище, що полягає в емісії світла тілами після їхнього збудження зовнішнім джерелом енергії (рис.1). Коли атоми чи молекули поглинають енергію, наприклад, внаслідок зіткнень з електронами або фотонами, їхні електрони переходять на вищі енергетичні рівні. Під час повернення на нижчі рівні енергії ці електрони випромінюють надлишкову енергію у вигляді фотонів світла. Цей процес є характерною реакцією для багатьох речовин, від природних мінералів до штучних фарб і покриттів, і знаходить широке застосування у фізиці, хімії та технології.

Рис.1.Приклад явища люмінесценції.

1.1.1.Механізм виникнення явища

Механізми виникнення явища люмінесценції різних речовин та тіл можуть бути різноманітними, і вони залежать від структури та властивостей матеріалу (рис.2). Ось кілька основних механізмів:

Рис.2. Механізми виникнення люмінесценції.

1. **Флуоресценція**: Це один з найпоширеніших механізмів люмінесценції. У цьому випадку електрони в піднятому стані, зазвичай в результаті поглинання світла або електронного збудження, повертаються до основного стану шляхом випромінювання світла (рис.3). Прикладами речовин, які демонструють флуоресценцію, є фарби, оксиди металів, різноманітні хімічні сполуки.
2. **Фосфоресценція**: Цей механізм схожий на флуоресценцію, але з ефектом затримки. Електрони після збудження затримуються на певному часі у високому енергетичному стані, перед тим як повернутися до основного стану та випромінити світло (рис.3). Це призводить до того, що світло випромінюється після того, як збудження припинилося. Фосфоресценція може спостерігатися у деяких мінералах, фарбах, пластику та інших матеріалах.

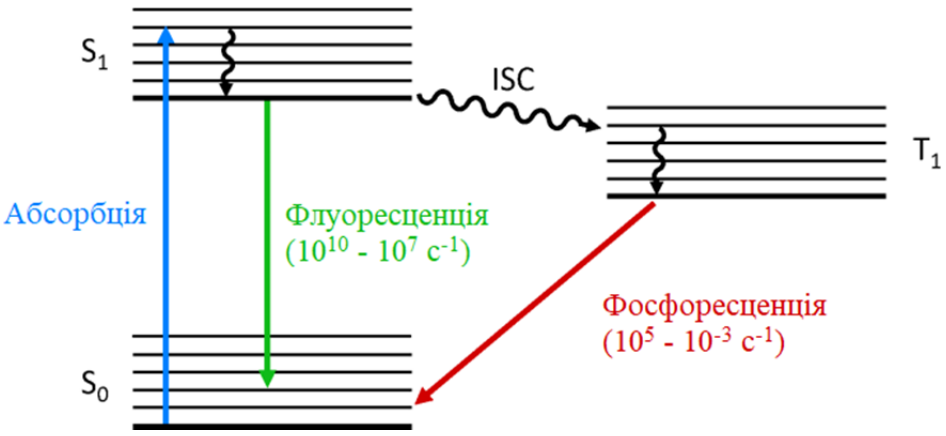


Рис.3. Механізм виникнення флуоресценції та фосфоресценції.

3. **Катодолюмінесценція**: Це явище спостерігається у кристалах деяких мінералів, наприклад, в сульфідах, коли вони піддаються удару електронами. Електрони збуджуються, переходять до вищих енергетичних рівнів і, коли повертаються до основного стану, випромінюють світло.
4. **Термолюмінесценція**: Цей процес виникає внаслідок нагрівання матеріалу. При підвищенні температури електрони можуть переходити на вищі енергетичні рівні, а потім випромінювати світло при поверненні до основного стану.
5. **Хемілюмінесценція**: Цей механізм включає світіння, яке виникає внаслідок хімічних реакцій. Наприклад, хемілюмінесценція може відбуватися при окисленні люмінофорів у хімічних реакціях.
6. **Радіаційна люмінесценція**: Деякі матеріали, такі як фосфори, можуть стати збудженими під впливом радіації, наприклад, ультрафіолетового світла або рентгенівських променів. Після збудження вони випромінюють світло.
7. **Електролюмінесценція**: Це явище виникає у напівпровідниках або електролюмінесцентних матеріалах, коли електрони переходять через зону забороненої провідності і видають світло під впливом електричного поля.

1.1.2.Люмінесценція на поверхні металів. Явище SPCE.

Поверхневе плазмонно-зв’язане випромінювання (SPCE) – це явище, яке виникає, коли флуорофори розміщуються в безпосередній близькості до металевих поверхонь, які підтримують поверхневі плазмони. Ця взаємодія призводить до підвищення інтенсивності випромінювання та чутливості, що робить його цінним інструментом для різноманітних аналітичних застосувань [7].

Принципи SPCE, включаючи збудження поверхневих плазмонів на металевих поверхнях та їх з’єднання з ближніми флуорофорами. Він також

може досліджувати фактори, що впливають на SPCE, такі як відстань між флуорофорами та металевою поверхнею, вибір металу та властивості самих флуорофорів [7].

З точки зору розробки аналізів, SPCE пропонує кілька переваг. Він може підвищити чутливість і межі виявлення аналізів, дозволяючи виявляти низькі концентрації аналітів. Крім того, аналізи на основі SPCE можуть запропонувати підвищену вибірковість і специфічність, що призводить до більш точних вимірювань [7].

1.1.3.Поведінка наночастинок на поверхні тіла

Наночастинки виявляють унікальну поведінку під час взаємодії з поверхнями, суттєво впливаючи на їхні оптичні властивості, включаючи люмінесценцію. Коли наночастинки осаджуються на поверхню, їхні електронні стани можуть бути змінені через взаємодію з підкладкою, що призводить до зміни їхніх люмінесцентних властивостей. Це може включати зміни довжини хвилі випромінювання, зміни інтенсивності та варіації тривалості люмінесценції. Ці ефекти особливо виражені в квантових точках, де стани поверхні можуть відігравати домінуючу роль у визначенні оптичних характеристик [12].

Хімічний склад поверхні наночастинок має вирішальне значення для визначення їхніх люмінесцентних властивостей. Пасивація поверхні, наприклад, може зменшити шляхи безвипромінювальної рекомбінації шляхом покриття поверхневих дефектів органічними або неорганічними лігандами. Цей процес збільшує квантовий вихід люмінесценції. Крім того, фізичне середовище навколо наночастинок, таке як наявність інших наночастинок або діелектричні властивості навколишнього середовища, також може впливати на люмінесцентну поведінку. Це пов’язано з такими ефектами, як резонансна

передача енергії Ферстера (FRET) або плазмонний зв’язок, які можуть гасити або посилювати люмінесценцію [12].

На ефективність підвищення люмінесценції (UCL) істотно впливають характеристики поверхні наночастинок. Для легованих лантаноїдом гексагональних частинок NaYF□ ефективність UCL вища порівняно з їхніми кубічними фазовими аналогами. Однак із збільшенням співвідношення поверхні до об’єму ефективність UCL зменшується через нерадіаційну дезактивацію іонів лантаноїдів у збудженому стані поверхневими дефектами та поверхнево-зв’язаними лігандами. На цю дезактивацію також можуть впливати молекули розчинника з модами високої вібраційної енергії, що оточують UCNP [2].

Синтез монодисперсних β-NaYF□:Yb^{3+} , Er^{3+} UCNP демонструє вузький розподіл за розміром і однорідну форму з гексагональною кристалічною структурою. TEM-зображення показують, що ці UCNP мають середній діаметр ядра 22,7 ± 0,7 нм і мають однакову сферичну форму. Модифікації поверхні, такі як олеатне покриття, покращують люмінесцентні властивості та колоїдну стабільність. Вимірювання динамічного розсіювання світла (DLS) показують сольводинамічний діаметр 29 ± 3 нм з індексом полідисперсності (PdI) 0,19 у циклогексані, що підтверджує вузький розподіл розмірів і однорідність наночастинок [2].

XRD-вимірювання підтверджують гексагональну кристалічну структуру, а якість наночастинок підкреслюється незмінним елементним складом, який відповідає теоретичним значенням, розрахованим на основі синтезу. Ці характеристики є важливими для розуміння поверхневих взаємодій і властивостей люмінесценції UCNP, що робить їх придатними для різноманітних люмінесцентних застосувань. Здатність контролювати властивості поверхні та підтримувати високу ефективність UCL має вирішальне значення для практичного використання цих наночастинок у технологічному застосуванні [2].

Останні досягнення в області люмінесцентних наночастинок значно розширили їх застосування в різних областях, включаючи біозображення, зондування та покриття поверхонь. Взаємодія цих наночастинок з різними поверхнями має вирішальне значення для оптимізації їхніх люмінесцентних властивостей. Коли люмінесцентні наночастинки осідають на поверхні, місцеве мікрооточення може впливати на їхні характеристики випромінювання. Такі фактори, як шорсткість поверхні, хімічний склад і наявність функціональних груп, можуть змінювати електронні стани наночастинок, таким чином впливаючи на ефективність і стабільність їх люмінесценції [8].

Наприклад, інтеграція люмінесцентних наночастинок з полімерними матрицями або скляними підкладками може підвищити їх стабільність і яскравість. Поверхнева функціональність наночастинок за допомогою специфічних лігандів або полімерів також може призвести до кращої дисперсії та зниження агрегації, що важливо для підтримки високого люмінесцентного виходу. Крім того, люмінесцентні наночастинки, пов’язані з поверхнею, досліджуються на предмет їх потенціалу для створення високочутливих платформ виявлення. Змінюючи властивості поверхні та керуючи просторовим розташуванням наночастинок, можна розробляти системи з адаптованими оптичними відгуками для конкретних застосувань, таких як біосенсори та пристрої моніторингу навколишнього середовища [8].

Теоретично можна моделювати поведінку люмінесцентних наночастинок на поверхнях, щоб передбачити їх взаємодію з різними субстратами. Обчислювальні методи, включаючи моделювання функціоналу густини (DFT) і молекулярної динаміки (MD), дають змогу зрозуміти механізми передачі енергії та вплив характеристик поверхні на люмінесцентні властивості. Ці моделі допомагають розробляти наночастинки з оптимізованими люмінесцентними характеристиками для конкретних застосувань, гарантуючи, що властивості матеріалів точно налаштовані відповідно до бажаних функціональних вимог [8].

1.2.Люмінесцентні матеріали

1.2.1.Звичайні люмінесцентні матеріали

Звичайні люмінесцентні матеріали широко використовуються в різних сферах, включаючи наукові дослідження, медицину, промисловість та побут. Ось кілька прикладів з описами:

1. **Барієві сульфіди** (BaS): Цей матеріал має яскравий люмінесцентний відгук на ультрафіолетове (УФ) світло. Барієві сульфіди часто використовуються у виробництві світловідбиваючих покриттів та як джерело світла у люмінофорних лампах.
2. **Цинковий сульфід** (ZnS): Цей білий порошок також є ефективним люмінофором. Його використовують для виготовлення фосфоресцентних пігментів, які додаються до пластмас для створення світловідбиваючих матеріалів, які світяться у темряві.
3. **Стронцієвий алюмінат** ($SrAl_2O_4$): Цей матеріал відомий своїм яскравим світловим відгуком, який може тривати дуже довго після взаємодії з УФ-світлом. Його часто застосовують у виробництві люмінофорних барвників для різноманітних застосувань, від нічних світловідбиваючих покриттів до яскравих фарб для фарбування.
4. **Ксенонові газорозрядні лампи**: Ці лампи містять газове заповнення, що включає ксенон. Під час роботи ксенон емітує УФ-світло, яке потім збуджує люмінофори, наприклад, барієві сульфіди або цинковий сульфід, роблячи світло видимим для людського ока.
5. **Фосфоресцентні барвники**: Ці барвники здатні поглинати світло і потім випромінювати його протягом тривалого часу. Вони використовуються у фарбах, світловідбиваючих матеріалах, лампах, електронних дисплеях і деяких типах сонячних панелей.

6. **Фторесцентні барвники**: Ці матеріали поглинають УФ-світло і випромінюють світло видимого спектру. Вони широко використовуються у яскравих фарбах, підсвічувальних системах, сигнальних пристроях та світлодіодах.
7. **Селенід кадмію** (CdSe): Цей напівпровідниковий матеріал відомий своїми квантовими точковими структурами, які виявляють сильну флуоресценцію. Використовуються для виробництва світлодіодів, сонячних елементів, біомедичних сенсорів та інших електронних пристроїв.

1.2.2.Сучасні люмінесцентні матеріали

Люмінесцентні матеріали в наноскопії STED:

Наноскопія зі стимульованим випромінюванням (STED) — це передова техніка флуоресцентної мікроскопії, яка дозволяє отримати зображення з надвисокою роздільною здатністю за межею дифракції світла. Використання люмінесцентних матеріалів у наноскопії STED має вирішальне значення для отримання зображень високої роздільної здатності біологічних і матеріальних структур. Останні досягнення в розробці флуорофорів STED включають різні матеріали, такі як неорганічні флуорофори, флуоресцентні білки, органічні люмінесцентні матеріали, люміногени, викликані агрегацією (AIE), і флуоресцентні наночастинки. Ці матеріали оптимізовано для покращення роздільної здатності та візуалізації зображень, а також для забезпечення тривалого відстеження на рівні нанорозміру. Вибір відповідних зондів і розробка нових люмінесцентних матеріалів мають важливе значення для вдосконалення зображень із високою роздільною здатністю як у матеріалознавстві, так і в біологічних дослідженнях [10].

Нетрадиційні люмінесцентні полімери:

Традиційні люмінесцентні полімери часто страждають від гасіння, викликаного агрегацією (ACQ), і високої цитотоксичності, що обмежує їх практичне застосування. Для вирішення цих проблем були розроблені нетрадиційні люмінесцентні полімери (НЛП). НЛП не покладаються на класичні хромофори, а замість цього використовують ізольовані бензольні кільця або багаті електронами фрагменти, такі як аміни, карбонільні групи, гідроксильні групи, прості ефіри, іміди та різні гетероатоми (наприклад, N, O, P, S). Ці полімери демонструють унікальні фотофізичні властивості, включаючи флуоресценцію та фосфоресценцію, що робить їх придатними для широкого спектру застосувань у матеріалах і біологічних науках. Останні дослідження висвітлюють синтез, механізми випромінювання та різноманітні застосування НЛП, пропонуючи нові можливості для їх використання в інноваційних технологіях зондування та зображення [10].

Люміногени, викликані агрегацією (AIE):

Люміногени, викликані агрегацією (AIE) — це клас матеріалів, які виявляють посилену люмінесценцію під час агрегації, що контрастує з типовим гасінням, що спостерігається у звичайних люмінофорах. Ця унікальна властивість робить люміногени AIE дуже цінними для біологічного та навколишнього зондування та зображень. AIE-активні матеріали успішно використовуються для виявлення різних аналітів, включаючи іони металів і біомолекули, а також для візуалізації живих клітин і тканин. Їх висока стабільність флуоресценції та чутливість роблять їх ідеальними кандидатами для розробки міцних і надійних сенсорних платформ. Дослідження, що тривають, спрямовані на покращення здатності до диспергування у воді та стабільності флуоресценції матеріалів AIE для розширення їх практичного застосування [10].

1.3.Люмінесцентна спектроскопія

Флуоресцентна спектроскопія є важливим методом для вивчення властивостей як органічних, так і неорганічних речовин. Коли певні молекули поглинають світло, вони збуджуються і згодом випромінюють світло з більшою довжиною хвилі, процес, відомий як флуоресценція. Це випромінювання зазвичай відбувається протягом 10 наносекунд після поглинання світла, що дозволяє вченим спостерігати швидкі молекулярні події [1].

Чутливість і часова роздільна здатність флуоресцентної спектроскопії є особливо вигідними для дослідження молекулярних взаємодій у біологічних системах, таких як білки та мембрани. Ці функції дозволяють дослідникам виявляти зміни в середовищі навколо флуоресцентних молекул, що робить його потужним інструментом для біохімічних і медичних досліджень. Наприклад, його можна використовувати для вивчення динаміки згортання білків, взаємодії між нуклеїновими кислотами та білками та властивостей клітинних мембран [1].

1.3.1.Закон Стокса для фотолюмінесценції

Закон Стокса є фундаментальним принципом, що описує явище фотолюмінесценції, яке відбувається в речовинах після їхнього освітлення ультрафіолетовим (UV) або видимим світлом. Цей закон був відкритий і описаний у 1852 році ірландським фізиком та математиком Джорджем Гебріелем Стоксом.

Згідно із законом Стокса, емісія фотолюмінесценції відбувається при більш довгих довжинах хвиль, ніж довжина хвилі поглинутого світла. Іншими

словами, фотолюмінесценція зазвичай випромінюється на довжинах хвиль, які більші за довжину хвилі світла, що викликало фотоемісію (рис.4).

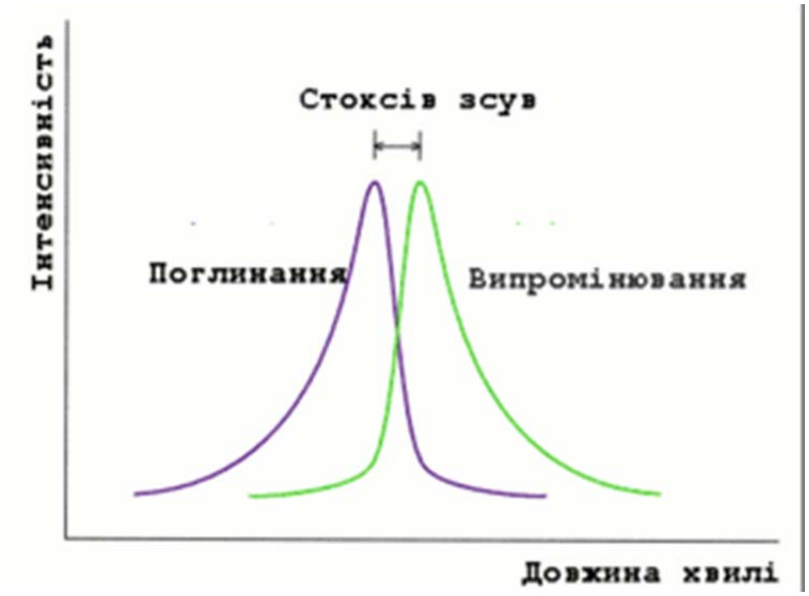


Рис.4. Закон Стокса для фотолюмінесценції.

Це пояснюється тим, що під час фотоемісії частина енергії, поглинутої атомом або молекулою, використовується для переходу електронів на вищі енергетичні рівні. Після припинення впливу зовнішнього світла ці електрони можуть повернутися на свої початкові енергетичні рівні, випромінюючи енергію у вигляді світла. Оскільки енергія фотонів, яка випромінюється під час фотолюмінесценції, зазвичай менша за енергію фотонів, що викликали фотоемісію, то довжина хвилі світла фотолюмінесценції зазвичай довша.

1.3.2.Інтеграція із зондами

Особливо варто відзначити інтеграцію люмінесцентних зондів у спектроскопію раманівського розсіювання з розширеним покриттям (SERS). Ці передові методи використовують унікальні оптичні властивості наноструктурованих матеріалів для значного посилення Раманівських сигналів, дозволяючи виявляти молекули в надзвичайно низьких концентраціях [9].

SERS поєднується з люмінесцентною спектроскопією за допомогою флуоресцентних або хемілюмінесцентних зондів, які підвищують чутливість і специфічність процесу виявлення. Наприклад, наночастинки золота (Au NP), функціоналізовані спеціальними зондами, можуть посилювати сигнал Рамана завдяки поверхневому плазмонному резонансу, який є колективним коливанням електронів у відповідь на світло. Це вдосконалення робить SERS потужним інструментом для біоаналізу та діагностики, оскільки він може забезпечити швидке, неінвазивне та високочутливе виявлення аж до рівня однієї молекули [9].

Останні досягнення показали розробку гібридних нанокомпозитів, таких як оксид графену в поєднанні з металевими наночастинками. Ці композити покращують стабільність і продуктивність субстратів SERS, роблячи їх придатними для застосування в режимі реального часу в клінічній діагностиці. Наприклад, високоефективні SERS-активні підкладки були створені шляхом вирощування графену безпосередньо на наночастинках срібла (Ag NPs), які не тільки захищають від окислення, але й посилюють Раманівський сигнал, не впливаючи на люмінесцентні властивості зондів [9].

1.3.3.Люмінесцентні картки

Люмінесцентні картки, які використовуються у наукових фізичних дослідженнях, зокрема в системі "штучний язик", — це спеціальні матеріали, які видають світло при збудженні енергією (рис.5). Зазвичай вони містять фосфоресцентні або люмінесцентні речовини, які можуть поглинати енергію від джерела світла і випромінювати її у вигляді світла певного спектрального складу.

У системі "штучний язик", яка є системою аналізу хімічних сполук, такі картки використовуються для визначення спектральних характеристик різних

речовин. Коли зразок розміщується на люмінесцентній картці та піддається збудженню, він може випромінювати світло з унікальними спектральними властивостями, які можуть бути зареєстровані та проаналізовані спеціальними приладами, наприклад, спектрофотометром. Це дозволяє встановити ідентифікацію речовини або визначити її концентрацію у зразку.

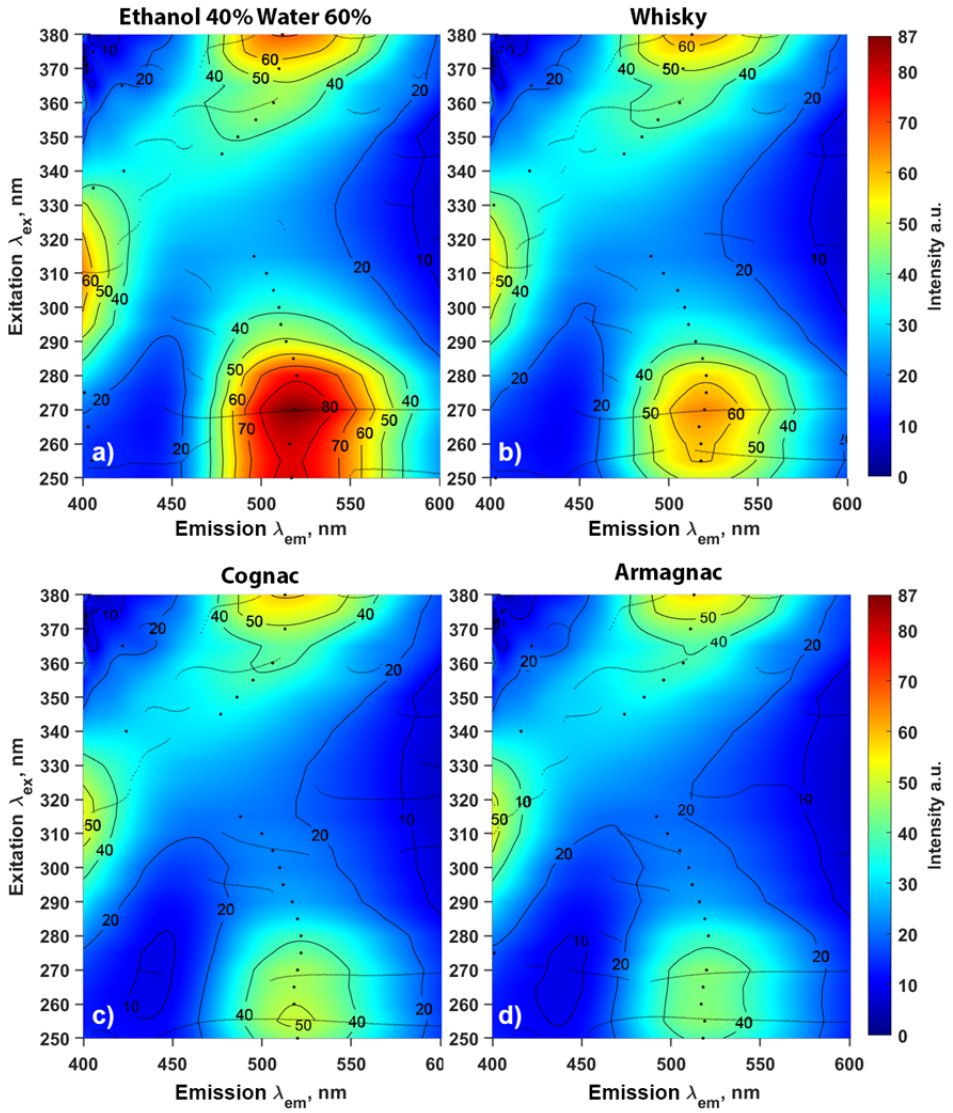


Рис.5. Приклад спектральної люмінесцентної картки. Залежність інтенсивності від довжини хвилі емісія-вихід.

Фізичне значення таких карток полягає в їхній здатності служити чутливими детекторами світла з високою роздільною здатністю. Вони дозволяють отримувати інформацію про спектральні характеристики зразка, що допомагає у точному визначенні його хімічного складу та властивостей.

1.4. Система «штучний язик»

1.4.1. Штучний язик вчиться розпізнавати сорти віскі, як дегустатор

Проблема ідентифікації високоякісних алкогольних напоїв у тому, що при належному дотриманні технології виробництва вони мають досить схожий хімічний склад. Різницю, і то вельми суб'єктивну, можуть розпізнати язик та ніс професійного дегустатора, але таких на планеті мало, а послуги недешеві. Не дивно, що у Гейдельберзькому університеті вирішили створити їм заміну.

Перший, «штучний язик» – зовсім не язик і навіть не сенсор, який потрібно занурювати в келих. Це набір ємностей з особливими полімерними барвниками, кожен з яких має індивідуальну реакцію. Виражається вона у флуоресценції барвника при контакті з алкоголем і залежить не від наявності у ньому конкретних речовин, а від унікальної їх комбінації. Невелике відхилення дає кардинально інший результат.

Рис.6. Набір ємностей з полімерними барвниками

Процес «дегустації» зводиться до того, що оператор заливає в ємності по краплі досліджуваного віскі, потім аналізує зміну світіння флуоресцентного барвника. Поєднання відтінків світла у всіх контрольних ємностях інтерпретується як індивідуальний «смак» або мітка напою. У хімічно ідентичної рідини іншого походження вона буде відрізнятися.

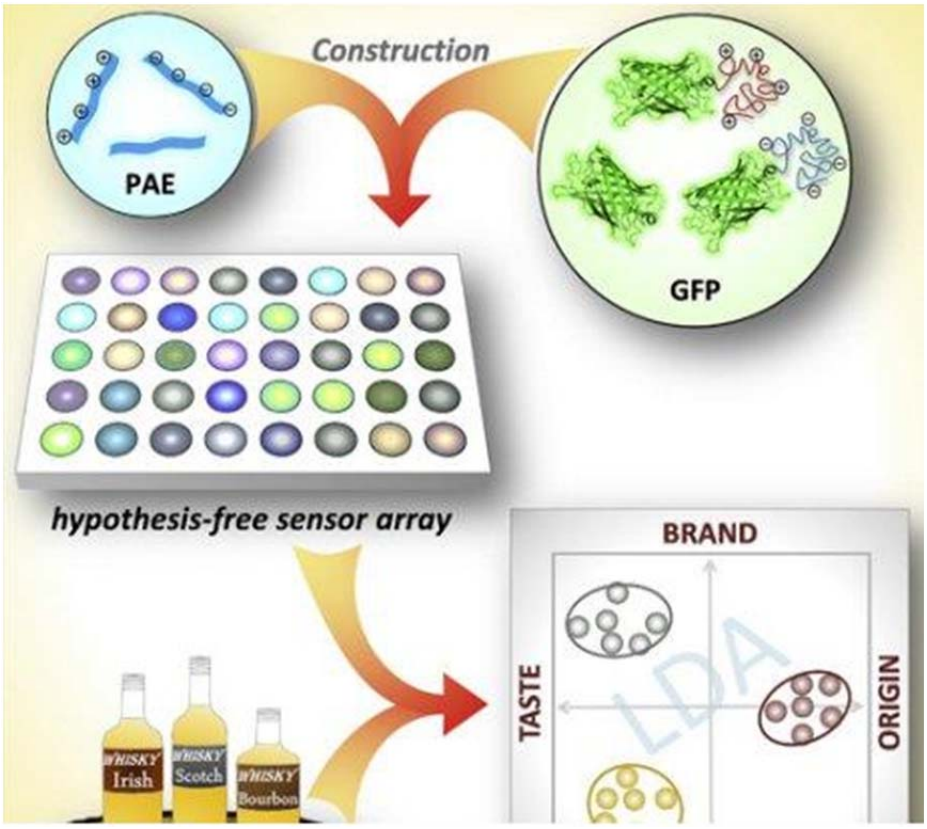


Рис.7.Схема процесу «дегустації».

Плюс розробки у тому, що вона досить точна, а великий мінус – така система тестування украй «дурна». За аналогією із детектором брехні, вона лише видає стовпчик чисел, інтерпретувати які доведеться оператору. Технологія дозволяє відрізнити напої за віком, країною походження, солодовий віскі від купажного, але тільки за умови попереднього внесення цих даних у БД. Самогонка, залита в аналізатор уперше, залишається «темною конячкою».

Віскі взято лише задля прикладу, щоб проілюструвати можливості штучного дегустатора. За словами провідного розробника, Уве Банца: «Немає принципових обмежень, щоб адаптувати його для аналізу будь-яких інших рідин зі складним хімічним складом, які часто підробляють або плутають» [15].

1.4.2. Штучний язик, який розпізнає різні смаки

Електронний язик працює за принципом людського, і за допомогою штучного інтелекту (ШІ) може аналізувати складні поєднання смаків.

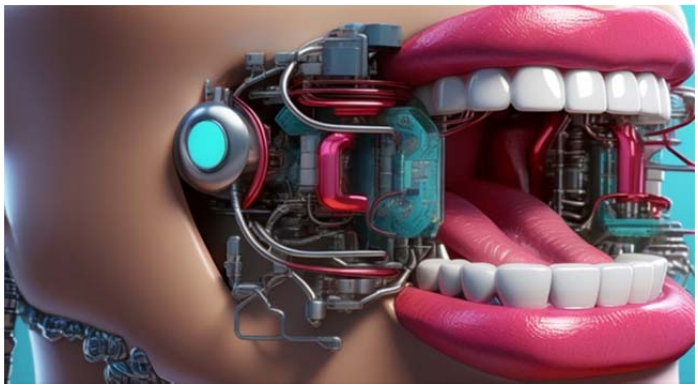


Рис.8. «Штучний язик», згенерований нейромережею.

Пристрій, створений науковцями з південнокорейського Інституту науки та технологій DGIST, може розпізнавати у зразку їжі чотири основні смаки. Зокрема, наступні: солоний, кислий, солодкий та терпкий. Він робить це наступним чином: імітує роботу смакових сосочків, які є на людському язику. П’ятий смак — умамі — штучний язик наразі не фіксує, зазначають у DGIST.

Умамі — японська інтерпретація “приємного смаку”, яка починається із посилення смакових якостей їжі. Це елемент, який гармонізує чотири основні смаки та доповнює їхню насиченість. Багато хто описує цей смак як “бульйонний, м’ясний”.

Як працює штучний язик

Інженери використали чотири датчики, кожен із яких призначений для визначення певного смаку. Пристрої розмірому кілька мм працюють як смакові рецептори: спочатку вони виявляють різні речовини, потім перетворюють хімічну інформацію на електричні сигнали.

Автори пояснили, що подібні системи були створені раніше, але їм не вистачало «мозку», щоб обробляти поєднання смаків у реальних стравах та напоях. Для розв’язання цієї проблеми було створено систему, яка поєднує датчики та технології глибокого навчання.

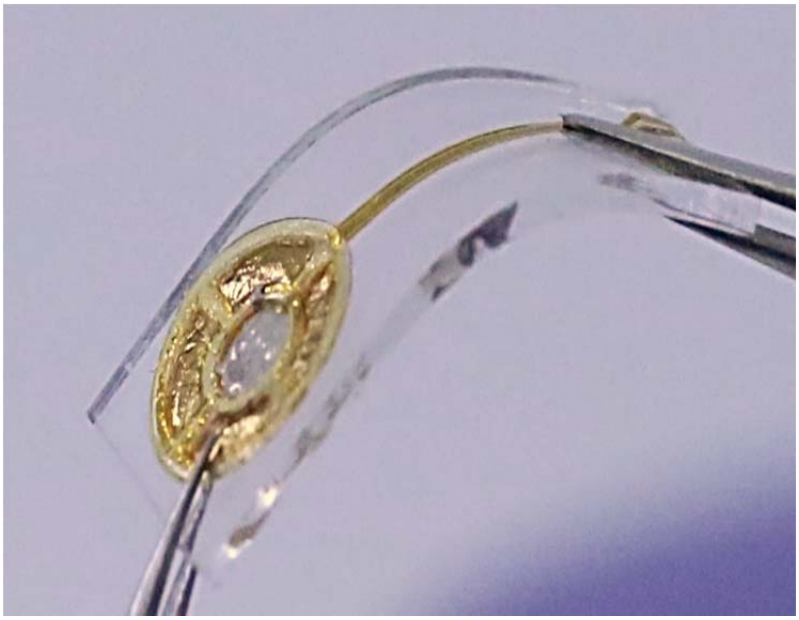


Рис.9. Електронний «штучний язик».

Чотири типи датчика для кожного смаку розміщуються у спеціальних лунках міліметрового масштабу, щоб забезпечити стабільні вимірювання. Водночас алгоритм глибокого навчання на базі ШІ використовується для аналізу смаку.

У рамках експериментів інженери протестували штучний язик на шести різних винах та змусили алгоритм скласти смакові характеристики. Результати показали, що електронний язик зміг класифікувати вина із точністю понад 95% та запровадити систему рекомендацій, яка пропонувала вина, схожі на наявні та відповідні різним стравам.

На думку вчених, електронний язик можна буде застосовувати у різних галузях промисловості, включаючи харчову, лікеро-горілчану, косметичну та фармацевтичну, оскільки вона ефективно поєднує датчики та технології

глибокого навчання для одночасного і точного вимірювання солоності, кислоти, гіркоти та солодкості [16].

2. ПОСТАНОВКА ЗАДАЧІ

Огляд проблеми:

У зв'язку з розвитком систем "штучний язик", які базуються на аналізі різних фізичних параметрів, виникає потреба в системі зберігання та обробки даних для побудови аналітичних моделей. Ці моделі включають в себе структуровані дані вимірювань багатьох фізичних величин, необхідних для побудови різних спектрів та подальшої генерації люмінесцентних карт.

Мета роботи:

Розробити базу даних MySQL/**SQLite** для зберігання та обробки вимірювань багатьох фізичних величин, які будуть використовуватися для аналізу та побудови спектрів у системах "штучний язик". Визначити оптимальну структуру бази даних для забезпечення ефективного доступу до інформації та швидкого виконання аналітичних операцій.

Задачі дослідження:

1. Розробити структуру бази даних, що включатиме таблиці для зберігання даних вимірювань фізичних величин, метаданих та інших відомостей.
2. Реалізувати механізми внесення, оновлення та видалення даних у базу даних через SQL-запити або програмний інтерфейс.
3. Розробити модулі для обробки та аналізу даних, необхідних для побудови різних спектрів, з використанням можливостей мови програмування або інших інструментів.

4. Реалізувати функціонал для побудови люмінесцентних карт на основі оброблених даних та зберігання їх у базі даних.
5. Провести тестування розробленої бази даних та програмного забезпечення для переконанняся в їхній працездатності та ефективності.

Очікувані результати:

1. Розроблена база даних MySQL/**SQLite**, яка здатна ефективно зберігати та організовувати дані вимірювань.
2. Реалізовані модулі для обробки та аналізу даних, які дозволяють побудувати різноманітні спектри.
3. Функціонал для побудови та зберігання люмінесцентних карт на основі оброблених даних.
4. Документація з описом розроблених компонентів та інструкціями щодо їх використання.

Обмеження:

1. Реалізація бази даних та програмного забезпечення повинна відповідати вимогам до продуктивності та ефективності.
2. Врахування можливих обмежень обсягу та швидкодії при роботі зі значними обсягами даних.

Перспективи використання:

Розроблена база даних може бути використана в різних областях, де необхідно аналізувати великі обсяги даних для побудови аналітичних моделей та генерації відповідних результатів.

3.ПРАКТИЧНА ЧАСТИНА

3.1.Бази даних

Бази даних є фундаментальним елементом сучасних інформаційних систем, дозволяючи ефективно зберігати, організовувати та отримувати великі обсяги даних (рис.10). Вони відіграють ключову роль у різних галузях, включаючи фінанси, охорону здоров'я, електронну комерцію, соціальні медіа та багато інших. Основна мета баз даних - забезпечити надійне зберігання даних з можливістю їх швидкого доступу та маніпулювання. Сучасні бази даних підтримують різні типи даних, такі як текстові, числові, мультимедійні, а також складні структури даних.

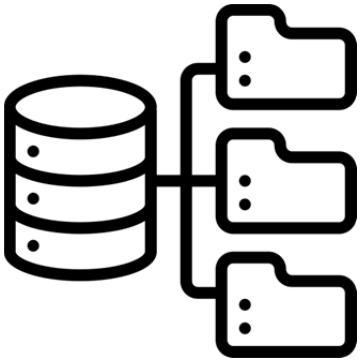


Рис.10. Схематичне зображення бази даних.

Різноманітність типів баз даних дозволяє вибрати оптимальне рішення для конкретних задач. Реляційні бази даних забезпечують цілісність і структурованість, що робить їх ідеальними для транзакційних систем. NoSQL бази даних пропонують гнучкість і масштабованість для обробки великих обсягів даних та роботи з неструктурованими даними. Використання правильного типу бази даних може значно підвищити продуктивність і ефективність роботи системи, забезпечуючи при цьому високу доступність та надійність даних [5,11].

3.1.1.Типи баз даних

Існує декілька основних типів баз даних, кожен з яких має свої переваги, недоліки та області застосування. Ось деякі з них:

Реляційні бази даних (RDBMS)

Опис: Реляційні бази даних зберігають дані у вигляді таблиць, які складаються з рядків і стовпців. Вони використовують мову SQL (Structured Query Language) для управління та запитів даних.

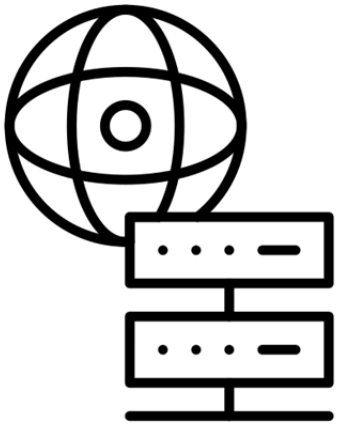


Рис.11. Схематичне зображення реляційної БД.

Переваги:

- Сильна підтримка цілісності даних через ключі та обмеження.
- Можливість складних запитів і транзакцій.
- Добре підходять для структурованих даних.

Недоліки:

- Може бути важко масштабувати горизонтально (тобто додавати більше серверів).
- Структурована природа може обмежувати гнучкість.

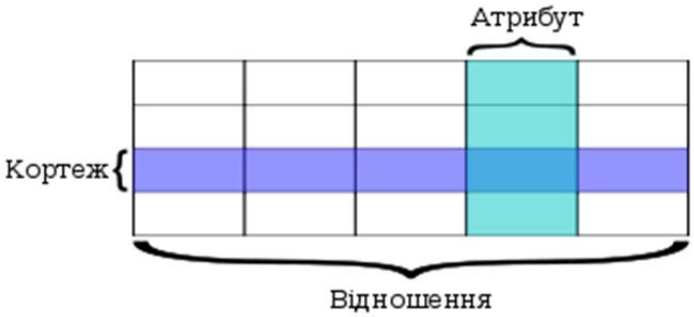


Рис.12. Складові таблиці у реляційній БД

Приклади застосування: Використовуються в банківській системі, для управління фінансами, у системах управління замовленнями та клієнтами (CRM), в ERP-системах (системи планування ресурсів підприємства) [5,14,17].

NoSQL (нереляційні) бази даних

Опис: NoSQL бази даних включають різні типи баз даних, що не використовують традиційну реляційну модель. Серед них виділяють документо-орієнтовані, графові, колоно-орієнтовані та бази даних з ключем-значенням.

Переваги:

- Висока продуктивність і масштабованість.
- Гнучкість у роботі з неструктурованими та напівструктурованими даними.
- Підтримка швидких операцій читання та запису.

Недоліки:

- Менше гарантій цілісності даних.
- Може бути складніше реалізувати складні запити.
- Відсутність стандарту для мови запитів.

Приклади застосування: Використовуються в соціальних мережах (Facebook, Twitter), для обробки великих обсягів даних (Big Data), в реальних

час системах аналізу даних, для зберігання користувацьких сесій та кешування [5,17].

Документо-орієнтовані бази даних

Опис: Зберігають дані у вигляді документів, які зазвичай використовують формат JSON, BSON або XML. Популярні представники - MongoDB та CouchDB.

Переваги:

- Гнучкість у моделюванні даних.
- Легкість у зберіганні та запитах напівструктурованих даних.
- Добре підходять для швидкої розробки ітеративних додатків.

Недоліки:

- Відсутність складних транзакцій.
- Можливі проблеми з консистентністю даних.

Приклади застосування: Використовуються для зберігання контенту веб-додатків, в системах управління контентом (CMS), в додатках, що потребують швидкої розробки та змін, таких як блоги, новинні сайти [17].

Графові бази даних

Опис: Спеціалізовані для зберігання і обробки даних, представлених у вигляді графів, де вузли представляють сутності, а ребра - відносини між ними. Приклад - Neo4j.

Переваги:

- Ефективні для роботи зі складними взаємозв’язками.
- Підтримка швидких запитів до графових структур.
- Добре підходять для аналізу соціальних мереж і рекомендаційних систем.

Недоліки:

- Може бути складно інтегрувати з іншими типами баз даних.
- Менш ефективні для звичайних транзакційних операцій.

Приклади застосування: Використовуються в соціальних мережах для моделювання взаємозв'язків між користувачами, в рекомендаційних системах, для управління мережевими структурами, в біоінформатиці [17].

Колоно-орієнтовані бази даних

Опис: Зберігають дані у вигляді колон, а не рядків, що дозволяє ефективно обробляти великі обсяги даних. Приклади - Apache Cassandra, HBase.

Переваги:

- Висока продуктивність при читанні і записі великих обсягів даних.
- Добре підходять для аналітичних запитів і обробки даних у режимі реального часу.

Недоліки:

- Менша гнучкість у запитах, ніж у реляційних баз даних.
- Складність управління та налаштування.

Приклади застосування: Використовуються для аналізу великих обсягів даних (Big Data), в телекомунікаційних системах, для зберігання історичних даних та журналів подій [17].

Бази даних із ключем-значенням

Опис: Найпростіший тип NoSQL баз даних, де кожен елемент даних зберігається як пара ключ-значення. Приклади - Redis, Amazon DynamoDB.

Переваги:

- Дуже висока продуктивність і швидкість доступу до даних.
- Простота реалізації і використання.

- Ідеальні для кешування та обробки сесій.

Недоліки:

- Обмежена можливість складних запитів.
- Не підходять для складних структурованих даних.

Приклади застосування: Використовуються для кешування веб-додатків, зберігання сесій користувачів, управління налаштуваннями та конфігураціями, в реальних час системах обробки даних [17].

3.1.2. Реляційні БД MySQL та SQLite

MySQL

Опис: MySQL – це популярна система управління реляційними базами даних (RDBMS), що використовується для зберігання, організації та управління даними. Вона базується на SQL (Structured Query Language) і широко використовується у веб-додатках та корпоративних системах. MySQL підтримує складні запити, транзакції і має потужні механізми забезпечення цілісності даних.

Переваги:

- Висока продуктивність і надійність.
- Підтримка масштабування та кластеризації.
- Поширене використання і велика спільнота підтримки.
- Підтримка ACID-транзакцій (у версії InnoDB).

Недоліки:

- Може бути складною у налаштуванні та управлінні для новачків.

- Не підходить для зберігання великих обсягів неструктурованих даних.
- Ліцензійні обмеження для деяких комерційних використань.

Випадки використання: MySQL часто використовується для веб-додатків (наприклад, WordPress, Joomla), електронної комерції (Magento), а також у корпоративних додатках, де необхідна висока надійність та підтримка транзакцій [4,14].

SQLite

Опис: SQLite – це легка, вбудована система управління реляційними базами даних. Вона відрізняється тим, що є бібліотекою, яку можна вбудувати безпосередньо в додаток, а не окремим сервером. SQLite зберігає всю базу даних в одному файлі, що робить її простою у використанні та розгортанні.

Переваги:

- Легка вага та простота вбудовування в додатки.
- Відсутність необхідності у налаштуванні та адмініструванні.
- Висока продуктивність для невеликих до середніх навантажень.
- Повністю безкоштовна і відкрита (Public Domain).

Недоліки:

- Обмежена продуктивність і масштабованість для великих навантажень.
- Відсутність підтримки складних транзакцій і паралельних запитів на високому рівні.
- Обмежена підтримка багатокористувацьких середовищ.

Випадки використання: SQLite ідеально підходить для мобільних додатків (наприклад, Android, iOS), настільних додатків, а також для веб-браузерів і додатків, де потрібна легка, вбудована база даних без необхідності складної конфігурації [4,14].

Спільне та відмінне між MySQL та SQLite

Спільне: Обидві системи є реляційними базами даних і підтримують SQL для запитів і управління даними. Вони обидві забезпечують високий рівень надійності та продуктивності у своїх відповідних областях застосування.

Відмінне: Основна відмінність полягає у способі розгортання та масштабованості. MySQL є серверною базою даних, яка підходить для великих систем і підтримує багатокористувацькі середовища, тоді як SQLite є вбудованою базою даних, оптимізованою для невеликих і середніх додатків без потреби в адмініструванні. MySQL підтримує складні транзакції та паралельні запити на високому рівні, тоді як SQLite обмежена в цьому плані [4,14].

3.2.Мова програмування Python

Python — це високорівнева мова програмування, яка була створена у 1991 році Гвідо ван Россумом. Її ключовими особливостями є простота у вивченні та читанні коду, а також висока продуктивність при розробці різних додатків. Завдяки своїй зрозумілості та лаконічному синтаксису, Python ідеально підходить для новачків у програмуванні, але також має потужні можливості, які цінують досвідчені розробники.

Python відомий своєю універсальністю і широко використовується у різних галузях. Він є відмінним вибором для задач машинного навчання та аналізу даних завдяки таким бібліотекам, як TensorFlow та Pandas. Веб-розробка на Python також популярна завдяки фреймворкам Django та Flask. Крім того, Python часто використовують для автоматизації рутинних завдань, обробки тексту, створення скриптів та написання тестів. Сильна підтримка спільноти та велика кількість готових бібліотек роблять його ще більш привабливим для широкого спектру застосувань.

Розглянемо кілька популярних бібліотек на Python, які розширюють його можливості. TensorFlow — це бібліотека для машинного навчання, яка дозволяє створювати складні нейронні мережі для розпізнавання образів, обробки природної мови та інших задач AI. Pandas — це бібліотека для аналізу даних, яка пропонує потужні інструменти для роботи з табличними даними, зокрема, для їх фільтрації, агрегації та візуалізації. Flask — легкий веб-фреймворк, який дозволяє швидко створювати веб-додатки та API. Ці бібліотеки демонструють, як Python може бути використаний для різних видів завдань, від аналізу даних до розробки веб-додатків [3,6,13].

3.2.1. Бібліотеки Python, потрібні для виконання практичного завдання

Pandas

Pandas — це бібліотека для аналізу та маніпулювання даними, яка надає високо продуктивні, зручні у використанні структури даних, такі як DataFrame та Series. Вона широко використовується для роботи з табличними даними, включаючи обробку пропущених значень, злиття та групування даних, а також виконання складних обчислень. Pandas дозволяє легко імпортувати дані з різних форматів, таких як CSV, Excel, SQL-бази даних та інші [3,13].

Приклад коду:

```
import pandas as pd

# Читання даних з CSV-файлу

df = pd.read_csv('data.csv')

# Показати перші 5 рядків

print(df.head())
```

```
# Групування даних за стовпцем і підрахунок середнього значення
grouped = df.groupby('category').mean()
print(grouped)
```

NumPy

NumPy (Numerical Python) — це фундаментальна бібліотека для наукових обчислень у Python, яка забезпечує підтримку великих багатовимірних масивів та матриць, а також набір високорівневих математичних функцій для роботи з ними. NumPy є основою для багатьох інших бібліотек, таких як Pandas, SciPy та Matplotlib. Основною перевагою NumPy є його швидкість та ефективність обробки числових даних [3,13].

Приклад коду:

```
import numpy as np
# Створення масиву NumPy
array = np.array([1, 2, 3, 4, 5])
# Виконання математичних операцій
squared = np.square(array)
print(squared)
# Створення 2D матриці та обчислення транспонованої
matrix = np.array([[1, 2, 3], [4, 5, 6]])
transposed = np.transpose(matrix)
print(transposed)
```

sqlite3

Бібліотека sqlite3 в Python забезпечує інтерфейс для роботи з базами даних SQLite, які є легковаговими, вбудованими базами даних, що зберігаються у файлі на диску. SQLite не вимагає встановлення окремого серверу бази даних і є ідеальним вибором для невеликих та середніх додатків, де потрібно зберігати дані локально. Використовуючи sqlite3, розробники можуть виконувати стандартні операції з базою даних, такі як створення таблиць, вставка, оновлення, видалення та вибірка даних за допомогою SQL-запитів [3,13].

Приклад коду:

```
import sqlite3

# Підключення до бази даних (або створення нової, якщо не існує)

conn = sqlite3.connect('example.db')

cursor = conn.cursor()

# Створення таблиці

cursor.execute('''CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name TEXT, age INTEGER)''')

# Вставка даних

cursor.execute('''INSERT INTO users (name, age) VALUES (?, ?)''', ('Alice', 30))

cursor.execute('''INSERT INTO users (name, age) VALUES (?, ?)''', ('Bob', 25))

# Вибірка даних

cursor.execute('''SELECT * FROM users''')

rows = cursor.fetchall()

for row in rows:
```

```
    print(row)

# Збереження змін та закриття підключення

conn.commit()

conn.close()
```

Matplotlib

Matplotlib — це бібліотека для візуалізації даних, яка дозволяє створювати високоякісні графіки, такі як лінійні графіки, гістограми, діаграми розсіювання, і багато інших. Вона дуже гнучка і забезпечує широкі можливості налаштування графіків, що робить її популярною серед науковців і інженерів для представлення даних у зрозумілому вигляді [3,13].

Приклад коду:

```
import matplotlib.pyplot as plt

# Дані для побудови графіка

x = np.linspace(0, 10, 100)

y = np.sin(x)

# Створення лінійного графіка

plt.plot(x, y)

plt.title('Sine Wave')

plt.xlabel('x')

plt.ylabel('sin(x)')

plt.show()
```

Tkinter

Tkinter — це стандартна бібліотека Python для створення графічних інтерфейсів користувача (GUI). Вона забезпечує прості у використанні інструменти для створення вікон, кнопок, міток, полів введення та інших GUI-елементів. Tkinter часто використовується для розробки простих настільних додатків завдяки своїй інтеграції з Python та простоті використання [3,13].

Приклад коду:

```
import tkinter as tk

# Створення головного вікна

root = tk.Tk()

root.title("Hello Tkinter")

# Додавання мітки

label = tk.Label(root, text="Hello, World!")

label.pack()

# Створення кнопки, що закриває вікно

button = tk.Button(root, text="Close", command=root.quit)

button.pack()

# Запуск GUI

root.mainloop()
```

3.3. Опис задачі

Є наступне завдання:

1. Створити базу даних на MySQL чи **SQLite** для наукових даних.
2. Створити для неї запити на вибірку даних.

Наприклад, є спектр відбиття, промодельований для 10 різних вхідних параметрів (P1, P2.....P10) і є 5 показників визначених з цього спектра (R1, R2....R5). Тобто, це вимір і його можна подати як рядок у таблиці базі даних, де назви параметрів і показників це назви стовпчиків. Потім потрібно створити інтерфейс, де користувач вибирає вхідні параметри і по ним робиться запит. Наприклад, вибрати усі рядки з таблиці, де P2, P3, ітд. – фіксовані, та побудувати залежності R1(P1), R2(P1)....R5(P1). Потім міняємо інший параметр і виводимо знову на графік.

Тобто, робота складається із 3 частин:

1. Створення Бази даних.
2. Завантаження в неї існуючих даних (автоматично) - тобто скрипт зчитує excel файл і заносить дані з нього у БД.
3. Отримання відгуків на запит, побудова графіків.

3.4. Виконання задачі

3.4.1. Етапи 1-2. Створення БД та завантаження даних у БД

Опис етапу 1: створення БД.

Програма запитує у користувача розмірність даних і мітки - тобто назви.

Наприклад:

Введіть скільки параметрів характеризує вимір (ікси) і які їх назви, наприклад M.

Скільки параметрів визначаєтья під час виміру (ігреки) і які їх назви, наприклад N.

Далі цикл від 1 до M, де питають назву M параметра і в яких межах яким кроком від буде мінятися (X1…Xm).

Далі цикл від 1 до N, де питають назву N параметра і в яких межах яким кроком від буде мінятися (Y1…Yn).

Створюється база даних, де стовпчики - це назва параметра.

Створюється таблиця в базі даних MySQL/**SQLite**, де стовпчики - це назва параметрів, що характеризують вимір (ікси) + парметри, що визначаються (ігреки).

Опис таблиці Excel, яка заливатиметься у нашу БД (рис.13):

Кожна вкладка - це виміряний параметр. Заголовки стовпчиків і першого лівого стовпчика це параметри що мінялися, а на перетині це визначена величина. Назва файлу містить фіксосвані параметри, наприклад:

results3DTopApod_WL550_nL1.6_nH1.95_Lmetal20_40_Nbi08_nAnalyte1_v01

WL0= 550 нм - довжина хвилі на яку проектувалася структура

Lmetal20_40 - товщина металу мінялася від 20 до 40 нм (це назви стовпчиків)

Nbi08 - кількість бішарів Nbi=08

nAnalyte1 - показник заломлення аналіту в порах = 1

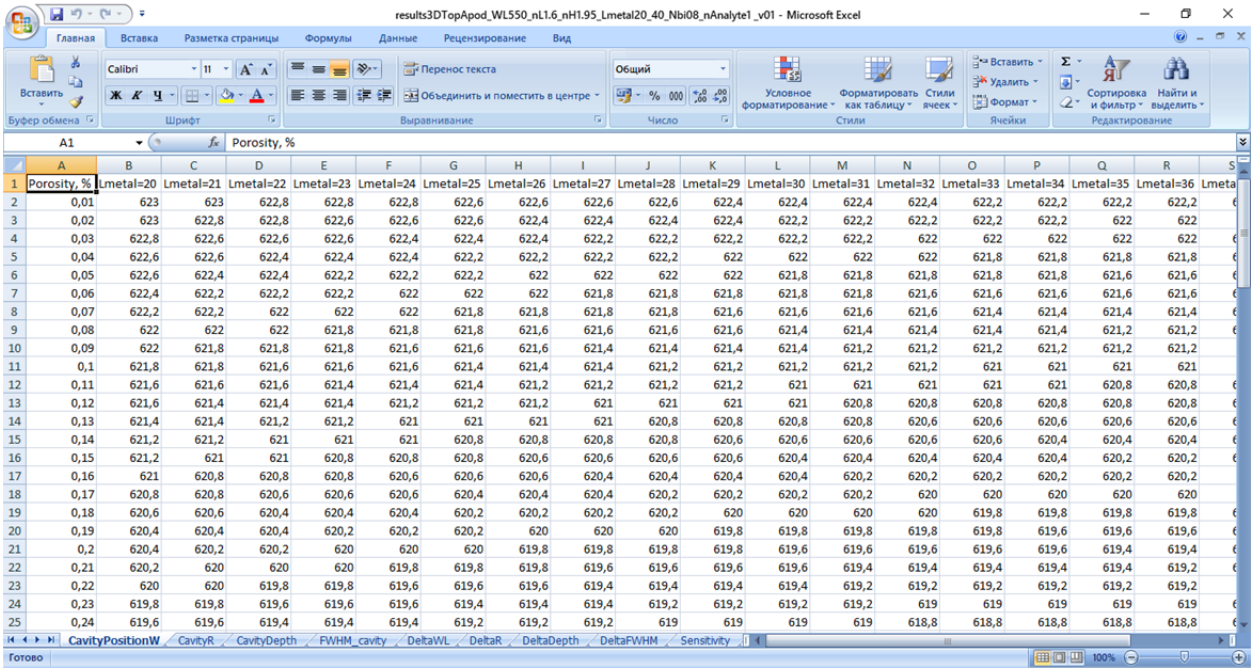


Рис.13. Приклад таблиці Excel, дані з якої заливатимуться у нашу БД.

Код програми, яка створює БД:

```
import tkinter as tk

from tkinter import messagebox

import sqlite3

import tkinter.filedialog as filedialog

import pandas as pd

import numpy as np

class TableApp:

    def __init__(self, master):

        self.master = master

        self.master.title("Дві таблиці")

        self.create_parameters_window()

    def import_excel_to_db(self, file_path):
```

```
try:
    df = pd.read_excel(file_path)
    print(df)
    conn = sqlite3.connect('data.db')
    c = conn.cursor()
    c.execute("PRAGMA table_info(Table_M)")
    columns_M = [col[1] for col in c.fetchall()]
    c.execute("PRAGMA table_info(Table_N)")
    columns_N = [col[1] for col in c.fetchall()]
    for col in df.columns:
        if col in columns_M:
            for i, val in enumerate(df[col]):
                try:
                    float_val = float(val)
                    df.at[i, col] = float_val
                except ValueError:
                    df.at[i, col] = np.nan
    for col in df.columns:
        if col in columns_M:
            print('col in columns_M ', col)
            df[col].to_sql('Table_M', conn, if_exists='append', index=False)
    for col in df.columns:
```

```
                if col in columns_N:
                    for i, val in enumerate(df[col]):
                        try:
                            float_val = float(val)
                            df.at[i, col] = float_val
                        except ValueError:
                            df.at[i, col] = np.nan
            for col in df.columns:
                if col in columns_N:
                    print('col in columns_N ', col)
                    df[col].to_sql('Table_N', conn, if_exists='append', index=False)
            conn.commit()
            conn.close()
            messagebox.showinfo("Успіх", "Дані успішно імпортовано із файлу Excel в базу даних!")
        except Exception as e:
            print(e)
            messagebox.showerror("Помилка", f"Помилка при імпорті даних із файлу Excel: {e}")
    def create_sqlite_tables(self, column_names_M, column_names_N):
        column_names_M = list(filter(None, column_names_M))
        column_names_N = list(filter(None, column_names_N))
```

```
            print(column_names_M, column_names_N)
            try:
                conn = sqlite3.connect('data.db')
                c = conn.cursor()
                c.execute('"DROP TABLE IF EXISTS Table_M"')
                c.execute('"DROP TABLE IF EXISTS Table_N"')
                c.execute('"CREATE TABLE Table_M ({});"'.format(', '.join(['"{}"'.format(col) for col in column_names_M])))
                c.execute('"CREATE TABLE Table_N ({});"'.format(', '.join(['"{}"'.format(col) for col in column_names_N])))
                conn.commit()
                conn.close()
                messagebox.showinfo("Успіх", "Таблиці SQLite успішно створені!")
            except Exception as e:
                print('aa', e)
                messagebox.showerror("Помилка", f"Помилка при створенні таблиць SQLite: {e}")
        def create_parameters_window(self):
            self.dialog = tk.Toplevel(self.master)
            self.dialog.title("Параметри таблиці")
            tk.Label(self.dialog, text="Параметри таблиці:").grid(row=0, column=0, columnspan=2, padx=5, pady=5, sticky="w")
```

```
tk.Label(self.dialog, text="Кількість стовпців M:").grid(row=1, column=0, padx=5, pady=5, sticky="w")

self.num_columns_M = tk.Entry(self.dialog)

self.num_columns_M.insert(tk.END, "3")

self.num_columns_M.grid(row=1, column=1, padx=5, pady=5)

tk.Label(self.dialog, text="Мінімальне значення:").grid(row=3, column=0, padx=5, pady=5, sticky="w")

self.lower_limit_M = tk.Entry(self.dialog)

self.lower_limit_M.insert(tk.END, "0")

self.lower_limit_M.grid(row=3, column=1, padx=5, pady=5)

tk.Label(self.dialog, text="Максимальне значення:").grid(row=4, column=0, padx=5, pady=5, sticky="w")

self.upper_limit_M = tk.Entry(self.dialog)

self.upper_limit_M.insert(tk.END, "10")

self.upper_limit_M.grid(row=4, column=1, padx=5, pady=5)

tk.Label(self.dialog, text="Кількість стовпців N:").grid(row=5, column=0, padx=5, pady=5, sticky="w")

self.num_columns_N = tk.Entry(self.dialog)

self.num_columns_N.insert(tk.END, "3")

self.num_columns_N.grid(row=5, column=1, padx=5, pady=5)

tk.Label(self.dialog, text="Мінімальне значення:").grid(row=7, column=0, padx=5, pady=5, sticky="w")

self.lower_limit_N = tk.Entry(self.dialog)
```

```
        self.lower_limit_N.insert(tk.END, "0")
        self.lower_limit_N.grid(row=7, column=1, padx=5, pady=5)
        tk.Label(self.dialog, text="Максимальне значення:").grid(row=8, column=0, padx=5, pady=5, sticky="w")
        self.upper_limit_N = tk.Entry(self.dialog)
        self.upper_limit_N.insert(tk.END, "10")
        self.upper_limit_N.grid(row=8, column=1, padx=5, pady=5)
        tk.Button(self.dialog, text="Створити таблицю", command=self.create_table).grid(row=9, columnspan=2, padx=5, pady=10)
    def create_table(self):
        try:
            num_columns_M = int(self.num_columns_M.get())
            num_columns_N = int(self.num_columns_N.get())
            table_window = tk.Toplevel(self.master)
            table_window.title("Таблиця")
            tk.Label(table_window, text="Стовпці таблиці M:").grid(row=0, column=0, padx=5, pady=5, sticky="w")
            self.column_M_entries = []
            for i in range(num_columns_M):
                entry = tk.Entry(table_window, width=20)
                entry.grid(row=i+1, column=0, padx=5, pady=2)
                self.column_M_entries.append(entry)
```

```
            tk.Label(table_window, text="Стовпці таблиці N:").grid(row=0, column=1, padx=5, pady=5, sticky="w")
            self.column_N_entries = []
            for i in range(num_columns_N):
                entry = tk.Entry(table_window, width=20)
                entry.grid(row=i+1, column=1, padx=5, pady=2)
                self.column_N_entries.append(entry)
            tk.Button(table_window, text="Створити таблицю", command=self.read_table_data).grid(row=10, columnspan=2, padx=5, pady=10)
            tk.Button(table_window, text="Імпортувати із Excel", command=self.import_excel).grid(row=11, columnspan=2, padx=5, pady=10)
        except Exception as e:
            print(e)
            messagebox.showerror("Помилка", f"Відбулася помилка: {e}")
    def read_table_data(self):
        try:
            column_names_M = [entry.get() for entry in self.column_M_entries]
            column_names_N = [entry.get() for entry in self.column_N_entries]
            self.create_sqlite_tables(column_names_M, column_names_N)
        except Exception as e:
            print(e)
            messagebox.showerror("Помилка", f"Помилка зчитування даних: {e}")
```

```
        def import_excel(self):
            try:
                file_path = filedialog.askopenfilename(filetypes=[("Excel files", "*.xlsx;*.xls")])
                if file_path:
                    self.import_excel_to_db(file_path)
            except Exception as e:
                print(e)
                messagebox.showerror("Помилка", f"Помилка імпорту файла Excel: {e}")
    if __name__ == "__main__":
        root = tk.Tk()
        app = TableApp(root)
        root.mainloop()
```

Код програми, яка відображає дані SQL-запитом:

```
import sqlite3
import pandas as pd
conn = sqlite3.connect('data.db')
df_M = pd.read_sql_query("SELECT * FROM Table_M", conn)
df_N = pd.read_sql_query("SELECT * FROM Table_N", conn)
conn.close()
print("Таблиця M:")
```

```
display(df_M)

print("\nТаблиця N:")

display(df_N)
```

Результати виконання програми (скріншоти):

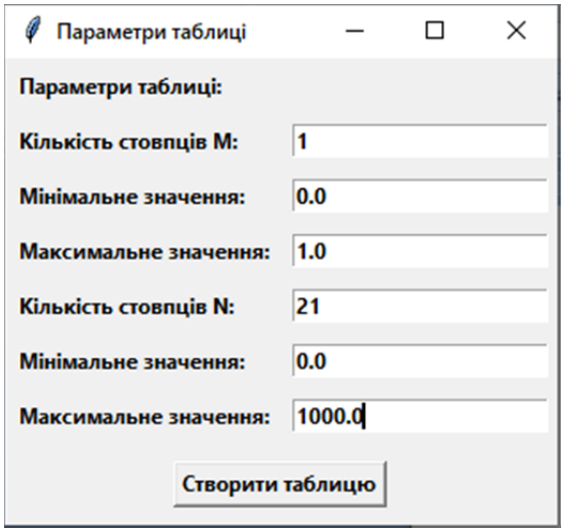


Рис.14. Параметри таблиці.

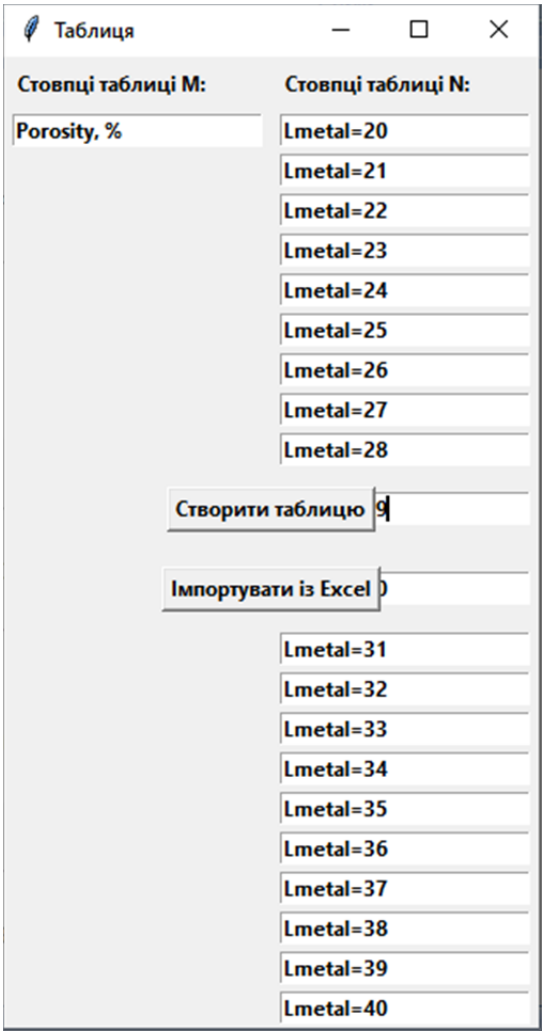


Рис.15.Назви стовпчиків таблиці.

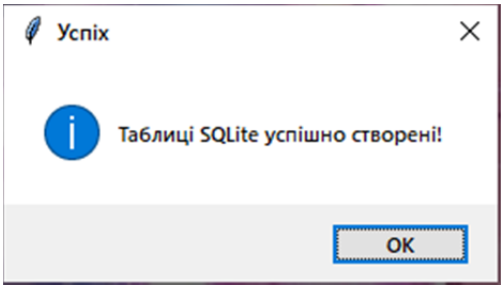


Рис.16.Успіх створення таблиць SQLite.

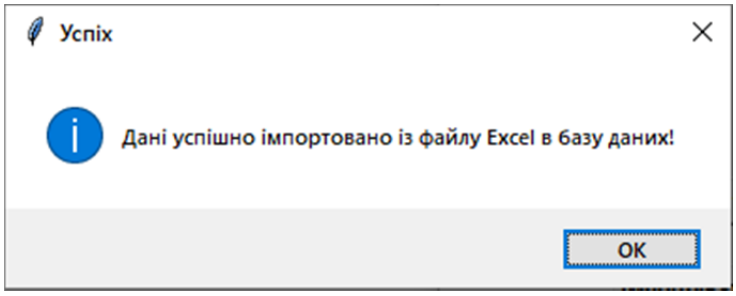


Рис.17.Успіх імпорту даних із таблиці Excel.

3.4.2. Етап 3. Створення графіків

Код програми:

```
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
import pandas as pd
import sqlite3
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import re
class DatabasePlotter(tk.Frame):
    def __init__(self, master=None, db_file="data.db", *args, **kwargs):
        super().__init__(master, *args, **kwargs)
        self.master = master
        self.db_file = db_file
        self.columns = self.get_lmetal_columns()
        self.canvas = None
        self.create_widgets()
        self.create_plot_canvas()
    def get_lmetal_columns(self):
        conn = sqlite3.connect(self.db_file)
        cursor = conn.cursor()
```

```
        cursor.execute("PRAGMA table_info(table_name);")
        columns = cursor.fetchall()
        columns = [column[1] for column in columns if column[1].startswith('Lmetal')]
        conn.close()
        return columns
    def create_widgets(self):
        self.column_name_var = tk.StringVar()
        column_name_label = tk.Label(self, text="Виберіть стовпець Lmetal:")
        column_name_label.pack(padx=5, pady=5)
        self.column_name_dropdown = ttk.Combobox(self, textvariable=self.column_name_var, values=self.columns)
        self.column_name_dropdown.pack(padx=5, pady=5)
        plot_button = tk.Button(self, text="Побудувати графік", command=self.plot_graph)
        plot_button.pack(padx=5, pady=5)
    def create_plot_canvas(self):
        fig, ax = plt.subplots()
        ax.set_xlabel('Porosity, %')
        ax.set_ylabel('Lmetal')
        ax.set_title('Графік Lmetal від Porosity, %')
        ax.grid(True)
```

```
        self.canvas = FigureCanvasTkAgg(fig, master=self)
        self.canvas.draw()
        self.canvas.get_tk_widget().pack(padx=10, pady=10)
    def update_columns(self):
        self.columns = self.get_lmetal_columns()
        self.column_name_dropdown['values'] = self.columns
    def get_constants(self):
        conn = sqlite3.connect(self.db_file)
        query = "SELECT * FROM constants"
        constants = pd.read_sql_query(query, conn)
        conn.close()
        return constants.iloc[0].to_dict() if not constants.empty else {}
    def plot_graph(self):
        column_name = self.column_name_var.get()
        if not column_name:
            messagebox.showwarning("Попередження", "Будь ласка, оберіть стовпець Lmetal")
            return
        conn = sqlite3.connect(self.db_file)
        query = f"SELECT `Porosity, %`, `{column_name}` FROM table_name"
        data = pd.read_sql_query(query, conn)
        conn.close()
```

```
            if data.empty:
                messagebox.showinfo("Немає даних", f"Немає даних для стовпця{column_name}")
                return
            constants = self.get_constants()
            legend_text = ', '.join([f'{k}={v}' for k, v in constants.items()])
            fig, ax = plt.subplots()
            ax.plot(data['Porosity, %'], data[column_name], marker=None, linestyle='-')
            ax.set_xlabel('Porosity, %')
            ax.set_ylabel(column_name)
            ax.set_title(f"Графік {column_name} от Porosity, %")
            ax.legend([legend_text])
            ax.grid(True)

            self.canvas.get_tk_widget().destroy()
            self.canvas = FigureCanvasTkAgg(fig, master=self)
            self.canvas.draw()
            self.canvas.get_tk_widget().pack(padx=10, pady=10)
        def extract_constants(filename):
            constants = {}
            pattern = r'(WL\d+|nL\d+\.\d+|nH\d+\.\d+|Nbi\d+\.\d+)'
```

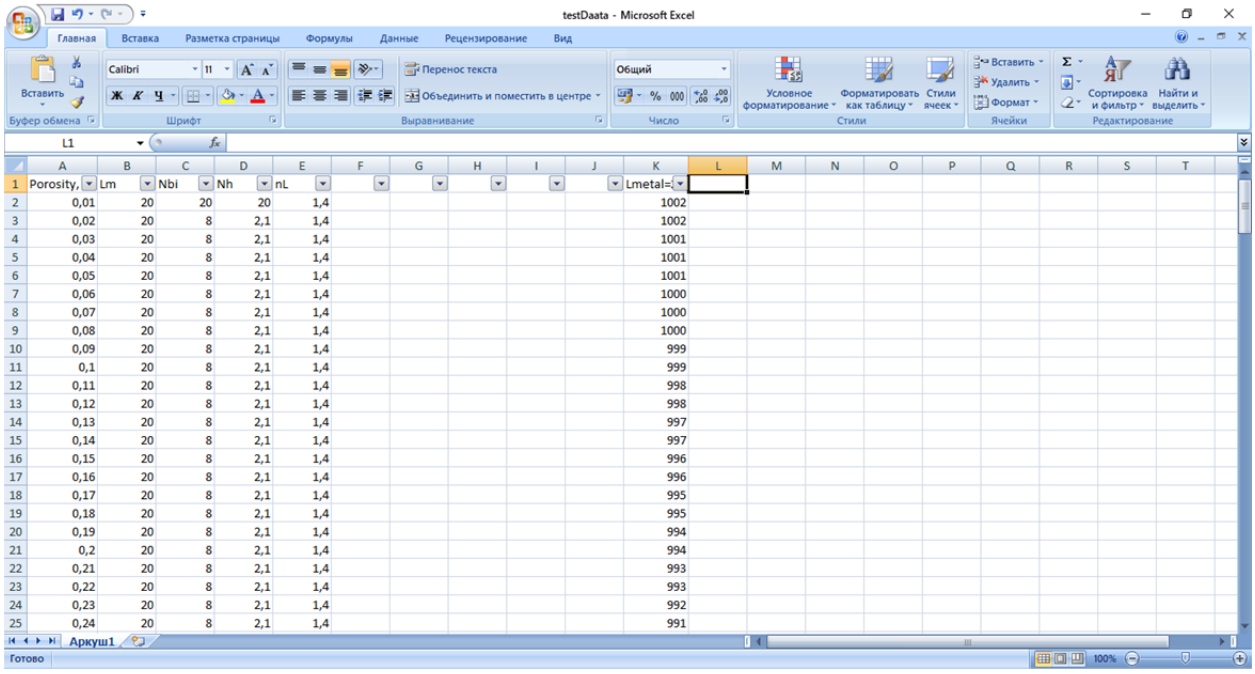
```
        matches = re.findall(pattern, filename)
        for match in matches:
            if match.startswith('WL'):
                constants['wl'] = match[2:]
            elif match.startswith('nL'):
                constants['nl'] = match[2:]
            elif match.startswith('nH'):
                constants['nh'] = match[2:]
            elif match.startswith('Nbi'):
                constants['nbi'] = match[3:]
        return constants
    def load_file():
        filepath = filedialog.askopenfilename(filetypes=[("Excel files", "*.xlsx;*.xls")])
        if not filepath:
            return
        try:
            data = pd.read_excel(filepath)
            constants = extract_constants(filepath)
            #print(constants)
            save_to_sqlite(data, constants)
            plotter.update_columns()
```

```
        except Exception as e:
            messagebox.showerror("Помилка", f"Не удалось прочитать файл: {e}")
    def save_to_sqlite(data, constants):
        try:
            conn = sqlite3.connect('data.db')
            data.to_sql('table_name', conn, if_exists='replace', index=False)
            const_df = pd.DataFrame([constants])
            const_df.to_sql('constants', conn, if_exists='replace', index=False)
            conn.close()
            messagebox.showinfo("Успіх", "Дані успішно збережені в базі даних SQLite!")
        except Exception as e:
            messagebox.showerror("Помилка", f"Не удалось сохранить данные в базу: {e}")
    root = tk.Tk()
    root.title("Excel to SQLite Converter and Plotter")
    frame = tk.Frame(root, padx=10, pady=10)
    frame.pack(padx=10, pady=10)
    load_button = tk.Button(frame, text="Завантажити файл Excel", command=load_file)
    load_button.pack(pady=10)
    plotter = DatabasePlotter(root)
```

plotter.pack(padx=10, pady=10)

root.mainloop()

Приклад відібраних тестових даних для побудови графіка (Excel):



	A	B	C	D	E	K
1	Porosity,	Lm	Nbi	Nh	nL	Lmetal=
2	0,01	20	20	20	1,4	1002
3	0,02	20	8	2,1	1,4	1002
4	0,03	20	8	2,1	1,4	1001
5	0,04	20	8	2,1	1,4	1001
6	0,05	20	8	2,1	1,4	1001
7	0,06	20	8	2,1	1,4	1000
8	0,07	20	8	2,1	1,4	1000
9	0,08	20	8	2,1	1,4	1000
10	0,09	20	8	2,1	1,4	999
11	0,1	20	8	2,1	1,4	999
12	0,11	20	8	2,1	1,4	998
13	0,12	20	8	2,1	1,4	998
14	0,13	20	8	2,1	1,4	997
15	0,14	20	8	2,1	1,4	997
16	0,15	20	8	2,1	1,4	996
17	0,16	20	8	2,1	1,4	996
18	0,17	20	8	2,1	1,4	995
19	0,18	20	8	2,1	1,4	995
20	0,19	20	8	2,1	1,4	994
21	0,2	20	8	2,1	1,4	994
22	0,21	20	8	2,1	1,4	993
23	0,22	20	8	2,1	1,4	993
24	0,23	20	8	2,1	1,4	992
25	0,24	20	8	2,1	1,4	991

Рис.18. Тестові дані для пояснення логіки побудови графіка.

Нехай стоїть задача: ми хочемо побудувати залежність *висоти піка* при Lmetal=20 або Lmetal=21 від *пористості* структури (porosity), при наступних фіксованих параметрах: Nbi = 8, nH=2.1, nL=1.4. Хочемо відфільтрувати дані за нашими константами: Lm, Nbi, nH, nL. У Lm обираємо одне із значень, наприклад 20. Будуємо графік залежності *висота_піка* = f(*пористість*), тобто, у нашому випадку, “Lmetal=20” – Y, “Porosity, %” -- X.(рис.18)

Мовою SQL наш запит на вибірку даних виглядатиме наступним чином:

SELECT “Porosity, %”, “Lmetal=20” FROM Аркуш1 (testDaata)

WHERE Lm=20 AND Nbi=8 AND Nh=2.1 AND nL=1.4;

Результати роботи коду:

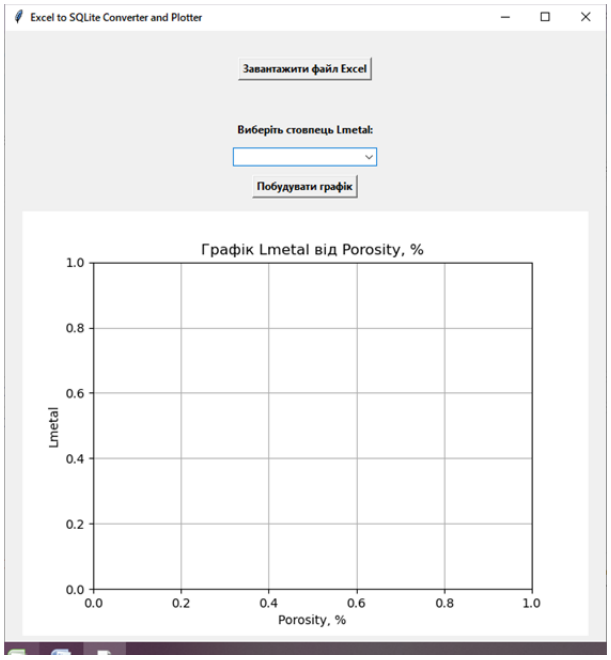


Рис.19. Стартове вікно програми.

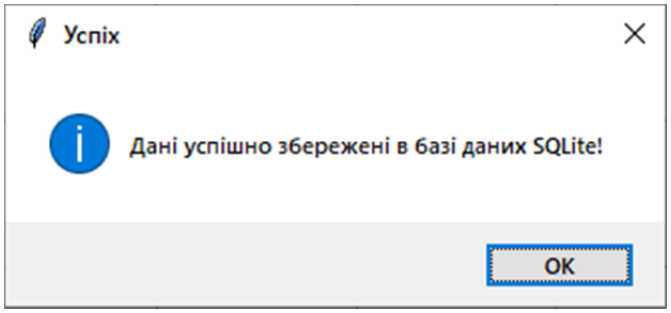


Рис.20. Успішне завантаження даних Excel у БД.

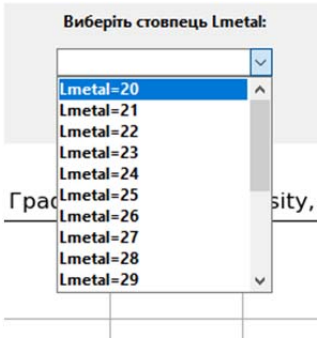


Рис.21. Випадаючий список вибору Y-параметра

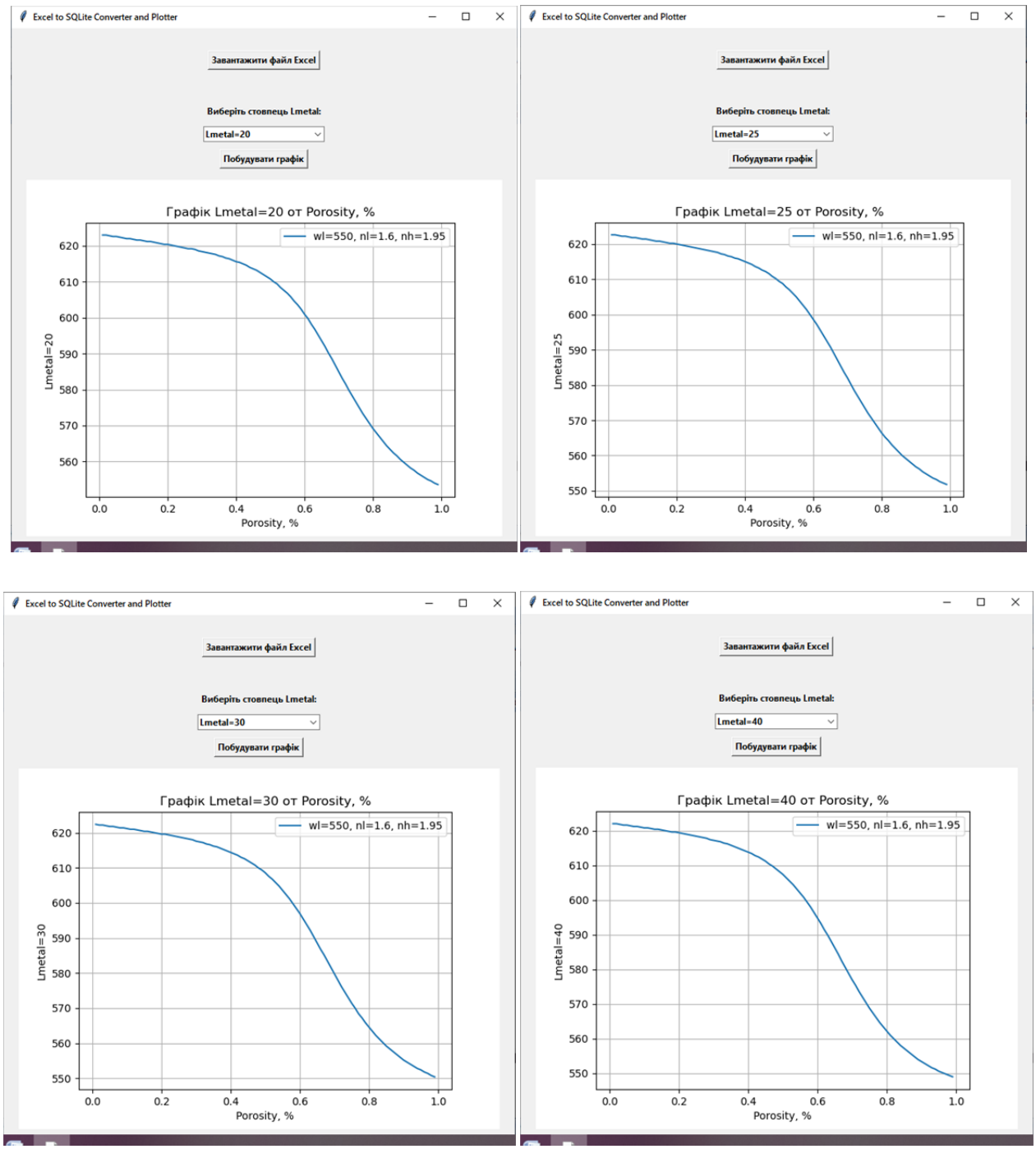


Рис.22. Приклади успішно побудованих графіків.

ВИСНОВКИ

Створено БД SQLite за допомогою Python. Також створено програму для відгуків на запити щодо побудови графіків (для спектрів).

Маємо наступні переваги наших БД та програми:

1. *Економія ресурсів*. Програма дозволяє нам заощадити час і ресурси, які були б витрачені на ручний пошук та аналіз даних. Тепер можна зосередитися на більш важливих елементах дослідження, наприклад, на інтерпретації отриманих результатів.
2. *Простота використання*. Для програми розроблено інтуїтивно зрозумілий графічний інтерфейс користувача, який робить її доступною як для досвідчених науковців, так і для студентів та молодих дослідників.
3. *Візуалізація даних*. Надано можливість візуалізувати результати за допомогою графіків, що полегшує аналіз та інтерпретацію даних. Це допомагає краще зрозуміти результати.

Проте, також маємо і певні недоліки:

1. Робота лише з однією таблицею у нашій базі даних, що є аналогом одного листа у файлі Excel (.xlsx), замість усього файла, себто групою листів, що є аналогом власне БД, у разі якщо проводити аналогію із файлом .csv.
2. Тестові дані сформовано без урахування константних величин у власне даних таблиці. Тобто, відсутні відповідні стовпці даних, що ускладнює їх заливання у БД та використання у побудові графіків.
3. Поки що неможливо будувати тривимірні графіки, що знову ж таки є наслідком відсутності важливого стовпця даних (фільтру) у таблиці(рис.18).

Із вищезгаданих недоліків випливають шляхи покращення ПЗ:

1. Пофіксити код python в цілях можливості заливання усіх таблиць-листів файлу Excel.
2. Додати нові стовпчики у тестові дані (таблиця-лист .xlsx файлу).
3. Доповнити код можливістю побудови 3D-графіків (площини), задля розширення можливостей застосування у науково-дослідницькій сфері.

СПИСОК ЛІТЕРАТУРИ

1. "Principles of Fluorescence Spectroscopy" by Joseph R. Lakowicz
2. "Nanoparticle Behavior and Applications" edited by Andrew P. Monkman, et al.
3. "Python for Data Analysis" by Wes McKinney
4. "Learning MySQL" by Seyed M.M. Tahaghoghi, Hugh E. Williams
5. "Introduction to Database Systems" by C.J. Date, A.Kannan, S. Swamynathan
6. "Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking" by Foster Provost and Tom Fawcett
7. "Surface Plasmon-Coupled Emission: What Is It and What Does It Mean for Assay Development?" by Brian Cullum et al.
8. "Luminescent Nanoparticles: A Review of Recent Advances in Materials and Applications" by Linlin Li et al.
9. "Advanced Techniques for Surface-Enhanced Raman Spectroscopy-Based Sensing" by Juan J. Diaz-Mochon et al.
10. "Recent Advances in Luminescent Materials for Biological and Environmental Sensing and Imaging" by Fanglong Yuan et al.
11. "Database Systems: The Complete Book" by Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom
12. "Handbook of Nanophysics: Nanoparticles and Quantum Dots" edited by Klaus D. Sattler
13. "Python Data Science Handbook" by Jake VanderPlas
14. "SQL for Data Scientists" by Hernan Vazquez
15. https://phoneinfo8.info/shtychnii-movy-vchitsia-rozpiznavati-sorti-viski-na-smak-iak-degystator/
16. https://psm7.com/uk/technology/uchenye-sozdali-iskusstvennyj-yazyk-kotoryj-raspoznaet-raznye-vkusy.html
17. https://dou.ua/lenta/articles/types-of-databases/