УДК 622.333+543.429.23 https://doi.org/10.37101/ftpgv25.01.002

МОДЕРНІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОДЕЛІ ПОРТАТИВНОГО СПЕКТРОМЕТРУ ЯМР

О.М. Молчанов[1]*, Т.В. Пічка[1]
[1]Відділення фізики гірничих процесів Інституту геотехнічної механіки ім. М.С. Полякова Національної академії наук України, м. Дніпро, Україна
*Відповідальний автор: e-mail: molchanov@nas.gov.ua

SOFTWARE MODERNIZATION FOR THE PORTABLE NMR SPECTROMETER MODEL

O.M. Molchanov[1*], T.V. Pichka[1]
[1]Branch for Physics of Mining Processes of the M.S. Poliakov Institute of Geotechnical Mechanics of the National Academy of Sciences of Ukraine, Dnipro, Ukraine
*Corresponding author: e-mail: molchanov@nas.gov.ua

ABSTRACT

Purpose. Modernization of the software, which provides the working process of a portable specialized nuclear magnetic resonance spectrometer on ^{1}H hydrogen nuclei, that are aimed for fossil coal samples study, processing and interpretation of research results.
Methods. Was used universal general-purpose programming language Python. It made possible to work on the development of all software modules for a specialized NMR spectrometer.
Results. To improve portable NMR spectrometer performance, was created new software. It includes three parts: user interface, instrument control module, and calculation module. Each module has its own tasks and responsible for performing program functions. To store the experiments and calculations data will be used structured data system, which forms Database. To build the software modules, we took into account technical characteristics and features of the specialized NMR spectrometers used for the fossil coal study, which are designed for recording NMR on protons (^{1}H). In the new software, calculation module was created on Python. It could be used as a component of the general software, as well as an independent program. Was given algorithm for processing complex experimental spectra.
Originality. To equip a portable NMR spectrometer, proposed to use a single-building software. Program module is developed using Python. It will take into account all needs for registration and processing of complex spectra of coal samples. Developed program will provide control for of discrete and analog output

channels operations and necessary parameters settings, processing and interpretation of spectra.
Practical implication. Software modernization for specialized NMR spectrometers for coal samples study, including, mobile portable NMR spectrometer, which is used in mine laboratory, will allow to improve the performance of express analysis of coal samples. It will improve the quality of the analysis and increase the reliability of the recommendations provided on the basis of the analysis for coal mining operations.
***Keywords*:** spectrometer, nuclear magnetic resonance, coal, methane, automation of scientific experiment, laboratory device

1. ВСТУП

Встановлення закономірностей виділення метану з викопного вугілля та порід, що його вміщають, при розробці, зберіганні та транспортуванні вугілля наразі лишається актуальною науковою і прикладною проблемою. Для вирішення цієї проблеми необхідно досліджувати фільтраційні, сорбційні властивості вугілля та гірничих порід, кінетику газовиділення з них. Дослідження вмісту метану у вугіллі та кінетики його виділення дає фізичний базис для обґрунтування рентабельності видобутку газу з вугільних пластів, дозволяє прогнозувати вибухи та викиди метану, розробляти способи боротьби з ними.

Лабораторні та шахтні експерименти, розробка теоретичних моделей масопереносу метану потребують отримання достовірних експериментальних даних. Для отримання надійних результатів дослідниками залучаються різноманітні експериментальні методи досліджень та сучасне наукове обладнання. Одним з таких сучасних методів досліджень є метод ядерного магнітного резонансу (ЯМР), на онові якого розроблено спеціалізовані спектрометри для дослідження викопного вугілля і стану флюїдів (метану і води) у ньому [1]. Всі ці спектрометри – стаціонарні (лабораторні) і мобільні (портативні), що використовуються безпосередньо на вугільних шахтах, – надійно працюють протягом багатьох років. Але всі вони час від часу потребують осучаснення елементної бази та програмного забезпечення, за допомогою якого вони функціонують і на основі якого проводиться обробка отриманих результатів.

Метою цієї роботи є модернізація програмного забезпечення для моделі портативного ЯМР-спектрометра, розробленого в ІФГП НАН України [2].

2. МЕТОДИКА ДОСЛІДЖЕНЬ

При розробці для управління ЯМР-спектрометром [2] була застосована система автоматизації наукового експерименту LabView. Програмна частина, що була написана мовою G (G – мова програмування з числовим керуванням, є частиною автоматизованого проектування, а також є вбудованим апаратом програмування для автоматизованих систем LabView), і призначена для управління установкою та отримання експериментальних даних. Для розробки програми, що виконує обробку спектрів, було використано

MathCAD. Код програми відкритий – це призводить до того, що оператор може легко ввести будь-які зміни до коду, що може призвести до того, що програма не зможе працювати коректно. LabView і MathCAD є ліцензованими програмами – для їх роботи необхідно закуповувати окрему ліцензію на кожний пристрій, а також підтримувати ліцензійну угоду. В іншому випадку, програми не будуть мати сучасного оновлення, що призведе до втрати продуктивності, а також некоректної роботи програм в цілому, а також програмного забезпечення (ПЗ), яке було створено на їх базі.

Для уніфікації процесу забезпечення портативного спектрометру програмним забезпеченням необхідно зробити так, щоб всі програмні модулі були написані єдиною мовою. Це полегшить обслуговування ПЗ приладу та його подальшу модернізацію.

Для вирішення поставленої задачі було вивчено придатність ряду найбільш поширених мов програмування, які б змогли забезпечити нам можливість створення єдиного програмного комплексу. У якості можливих мов програмування для вдосконалення роботи нашого портативного спектрометру було обрано декілька мов, а саме C, Python і R. Це мови програмування, за допомогою яких ми можемо створювати програмний продукт без необхідності долучення інших спеціалістів. Якщо використовувати C для побудови нашої програми, то це займе досить багато часу, через те, що ця мова не має вбудованих бібліотек розрахунку. Усі параметри, та формули буде необхідно описати власноруч. Після цього буде необхідно настроїти програму. Для цього її роботу треба буде покрити тестами, які також потрібно написати. Тести дозволять нам знайти місця, де програма працює некоректно та виправити це.

Розробники використовують Python, тому що він ефективний, простий у вивченні та працює на різних платформах. Програми мовою Python можна завантажити безкоштовно, вони сумісні з усіма типами систем та підвищують швидкість розробки.

Мова Python має такі переваги:

- Розробники можуть легко читати та розуміти програми на Python, оскільки мова має базовий синтаксис.
- Python допомагає розробникам бути більш продуктивними, оскільки вони можуть писати програми на Python, використовуючи менше рядків коду, ніж іншими мовами.
- Python має велику стандартну бібліотеку, що містить багаторазові коди практично для будь-якого завдання. В результаті розробникам не потрібно писати код із нуля.
- Розробники можуть легко поєднувати Python з іншими популярними мовами програмування: Java, C та C++.
- Python можна переносити на різні операційні системи: Windows, MacOS, Linux і Unix.

R – це мова програмування з відкритим вихідним кодом, що використовується для обробки та аналізу даних. R включає не лише мову з унікальним синтаксисом і можливостями, але й відповідний фреймворк, а також середовище запуску програм. R та його компоненти часто використовуються в науці, наприклад, для створення програм на базі машинного навчання.

Python часто застосовується в обробці статистичних даних, зборі та аналізі даних. Він також популярний серед математиків, біологів та підприємців. При цьому Python є мовою загального призначення. Він підтримує більше форматів даних, дозволяє писати звичні функції тощо. R спрямований на вирішення конкретних завдань. Ця мова менш функціональна і не дозволяє створювати повноцінні програми, як Python. R підійде для конкретних завдань в галузі обробки великих масивів даних. Python більше підійде для багатофункціональної розробки.

Тобто для наших задач більше підійде Python, оскільки на відміну від мови програмування С, він має багато бібліотек, що значно спростить розробку програми і прискорить час створення коду. Мова R також має свої переваги, у вигляді більш детальних бібліотек, для обробки даних. Але вона нам не підходить, через те, що її ми можемо застосувати лише для модуля обробки спектрів. У той самий час Python є більш універсальною мовою, на якій ми зможемо працювати над розробкою обох модулів.

Таким чином, серед декількох мов було обрано мову програмування Python [3]. Python завдяки своїм перевагам дасть можливість працювати над розробкою всіх модулів ПЗ. Створення комплексного програмного забезпечення для нашого приладу за допомогою однієї мови програмування дасть можливість створити керуючу програму, яка дозволить проводити експеримент в повністю автоматичному режимі. При розробці програмного забезпечення в Python були написані функції та класи даних для роботи програмного модулю. Назви класів та функцій створювались так, щоб при перегляді програми було зрозуміло за що відповідає створена функція або клас.

3. СТРУКТУРА ОНОВЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розглянемо побудову і функціонування оновленого ПЗ.

Програмний модуль передбачає наступні сценарії роботи:

- Після введення параметрів та даних експерименту – відбувається запис спектрів, їх розрахунок та відображення результатів у повністю автоматичному режимі.
- Після введення параметрів та даних експерименту – відбувається тільки запис спектрів зі збереженням їх до БД.

Для поліпшення роботи портативного спектрометру ЯМР створено програмне забезпечення, яке складається із трьох частин: інтерфейс користувача (**UI**), модуль керування приладом, розрахунковий модуль. Кожен модуль має свої задачі та відповідає за виконання найбільш важливих функцій.

Так, **UI** організує взаємодію з користувачем, тобто:

1. Реалізує створення запису для проведення експерименту, встановлення всіх необхідних параметрів для кожного виміру та експерименту в цілому.
2. Перегляд записів, які стосуються раніше проведених експериментів.
3. Перегляд усіх розрахунків спектрів, що було виконано у розрахунковому модулі.
4. Можливо передбачити ще декілька функцій за запитом користувача.

Модуль керування приладом:

1. Отримує попередній запис, щодо формування експерименту (параметри, кількість вимірів).
2. Формує керуючий сигнал для приладу.
3. Отримує данні з приладу.
4. Заносить відзняті спектри та попередній опис експерименту до Бази Даних (БД).

Розрахунковий модуль виконує наступні функції:

1. Отримує запит на обробку спектрів від користувача.
2. Проводить розрахунок спектрів, за розробленою методикою.
3. Зберігає отримані результати у (БД).

Для зберігання даних експериментів та розрахунків буде використано структуровану систему зберігання даних, тобто Базу Даних (БД).

Структура розробленої БД має виглядпоказаний на Рис. 1.

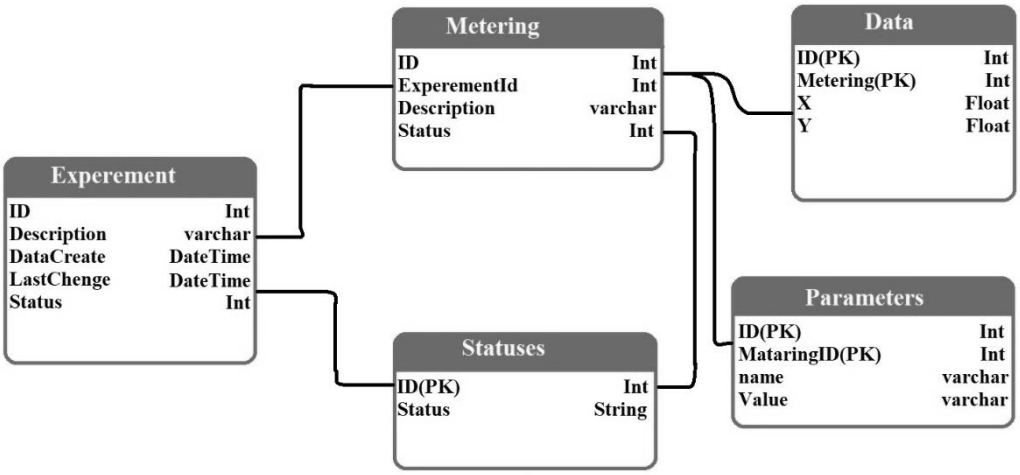

Рисунок 1. Структура розробленої Бази Даних

Experiment – опис проведення експерименту: індивідуальний номер запису; опис, будь-які додаткові данні стосовно зразка; дата, коли було створено запис; дата останньої зміни запису; статус – статус, що має запис (записано, прораховано, змінено, створено та інші).

Metering – структура, яка зберігає параметри запису того чи іншого спектру.

Data – структура, яка за запитом відкриє необхідний запис експерименту, для подальшої роботи з ним.

Parameter – параметри, які можуть бути обрані при записі спектру, а також використанні для пошуку необхідного запису у БД. Структура параметрів є динамічною тому дозволяє включати додаткові параметри до записів. Має перевірку на доречність вводу параметру, тобто, при запиті параметру дати не можна ввести «вівторок», у такому разі програма виведе помилку та попросить перевірити доречність вводу параметрів.

Status – таблиця статусів, які можуть бути присвоєні запису: записано, прораховано, змінено, створено, параметри введено із помилкою, перевірте правильність вводу даних та інші.

Модуль програми, що створює запис у БД виглядає наступним чином:

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy import Column, ForeignKey
from sqlalchemy import Integer, String, Float, DateTime
from sqlalchemy.ext.declarative import declarative_base
import numpy as np

Base = declarative_base()

class Status(Base):
    __tablename__ = "status"
    id = Column('id', Integer, primary_key=True)
    status = Column('value', String)

class Experiment(Base):
    __tablename__ = "experiment"
    id = Column('id', Integer, primary_key=True)
    description = Column('description', String, unique=True)
    dateCreate = Column('date_create', DateTime)
    lastChange = Column('lastChange', DateTime)
    status  = Column('status_id', Integer, ForeignKey('status.id'))

class Metering(Base):
    __tablename__ = "metering"
    id = Column('id', Integer, primary_key=True)
        experiment_id = Column('experiment_id', Integer, For-
eignKey('experiment.id'))
    description = Column('description', String, unique=True)
    status  = Column('status_id', Integer, ForeignKey('status.id'))

class Parameter(Base):
    __tablename__ = "parameter"
    id = Column('id', Integer, primary_key=True)
        experiment_id = Column('experiment_id', Integer, For-
eignKey('experiment.id'))
    name = Column('name', String, unique=True)
    value = Column('value', String, unique=True)

class Data(Base):
    __tablename__ = "data"
    id = Column('id', Integer, primary_key=True)
    metering_id = Column('metering_id', Integer, ForeignKey('metering.id'))
```

```
    x = Column('x', Float)
    y = Column('y', Float)

def saveDatas(engine, experimentId:int, meteringId:int, data_x:np.array, data_y:np.array):
    session_maker = sessionmaker(bind=engine)
    session = session_maker()
    experiment = session.query(Experiment).where(Experiment.id==experimentId)
    metering = session.query(Metering).where(Metering.id==meteringId)
    i=0
    datas = []
    while (i<x.lenth()):
      d = Data(metering_id=meteringId, x=data_x[i], y=data_y[i])
      i=i+1
      datas.append(d)
    session.add_all(datas)
    experiment.status=2
    session.commit()
```

Перший блок модулю:

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy import Column, ForeignKey
from sqlalchemy import Integer, String, Float, DateTime
from sqlalchemy.ext.declarative import declarative_base
import numpy as np
```

Це бібліотеки, які допомагають формувати та змінювати нашу базу відповідно до наших потреб.

Другий блок модулю, який включає опис класів:

```
class Status(Base):
class Experiment(Base):
class Metering(Base):
class Parameter(Base):
class Data(Base):
```

Задає структуру бази даних, та описує параметри.

Функція def saveDatas(engine, experimentId:int, meteringId:int, data_x:np.array, data_y:np.array) приймає 4 параметри: номер експерименту, номер запису, масив значень X, масив значень У – та записує у сформований запис бази даних.

Ймовірний сценарій роботи ПЗ:

1. Користувач вибирає проведення експерименту, та встановлює кількість вимірів, типи вимірів та їх параметри.
2. Зберігає запис в БД, як чернетку.
3. Коли всі параметри експерименту створені і він готовий до зміни статусу на відправлений в роботу, id експерименту (відповідний запис експерименту) поміщаємо в ‘чергу’.
4. Сервіс читає ‘чергу’ і отримує id експерименту. З БД отримує всю інформацію про виміри та їх параметри. Можливо перевіряє правильність вказаних параметрів.
5. Відправляє повідомлення, що він готовий до роботи.
6. Користувач перевіряє, що в прилад помістили потрібний екземпляр і все готове. Після чого надсилає повідомлення, що все ГОТОВО. У користувача з’являється попереджувальна табличка, що прилад у роботі і його чіпати не можна.
7. Починається знімання показань.
8. Результат запису спектрів зберігається в БД із прив’язкою до id експерименту. І id спектру кладеться у ‘чергу’.
9. Так буде тримати поки всі спектри (задана користувачем кількість, або виявлені за параметрами) не будуть отримані.
10. Розрахунковий модуль отримує id спектру з ‘черги’ і починає розраховувати результат, який зберігає в БД і змінює статус розрахунку («не виконано», «частково виконано», «виконано»).

Для побудови програмних модулів було враховано технічні характеристики та особливості роботи спеціалізованих спектрометрів ЯМР, що використовуються нами для дослідження викопного вугілля, які призначені для реєстрації ЯМР на протонах (^{1}H) [2, 4].

Одне з головних завдань, яке вирішувалося при розробці портативного спектрометра, – досягти порівняно легкої його мобільності, без втрати функціональних можливостей повноцінного спектрометра. Для реєстрації спектрів ЯМР формується сигнал керування, який передається через блок комутації на прилад.

Для побудови модулю розрахунку необхідно врахувати методи розрахунку спектрів ЯМР.

Дослідження зразків викопного вугілля, насичених вологою і метаном, методом спектроскопії ядерного магнітного резонансу на ядрах водню пов’язане з труднощами, викликаними складною структурою спектрів, що отримуються. Вважається, що метан у вугіллі залежно від локалізації знаходиться в декількох фазових станах [1]. Питання про розподіл всього метану за формами існування у вугільній речовині, про перерозподіл кількості метану між фазовими станами при порушенні рівноважних умов та про параметри газовиділення з вугілля при його видобутку мають велике наукове та практичне значення. Від правильного вирішення цих питань практично залежить безаварійна організація гірничих робіт.

Правильна оцінка вмісту метану у різних фазових станах необхідна задля забезпечення достовірності теоретичних досліджень кінетики виділення метану з вугілля. На практиці розподіл метану за фазовими станами у вугіллі з достатньою точністю може бути визначений фізичними методами дослі-

дження, наприклад, методом ядерного магнітного резонансу. Форма і структура спектрів ЯМР ^{1}H вугілля, що насичене метаном, а також методи їх математичної обробки наведені, наприклад, в [1, 5]. Оскільки основну інформацію про стан метану, що знаходиться у вугіллі, несе вузька компонента (вузька лінія) повного спектру ЯМР ^{1}H, то при математичній обробці отриманих під час експерименту спектрів широку лінію повного спектру (від органіки вугілля) необхідно математично «видалити», усунувши цим її вплив на точність розрахунку вузької лінії.

Відомо, що вузька лінія також є складною. Її формують ядра водню від метану CH_4 в різних фазових станах і, відповідно, з різною рухливістю. Інтегральні інтенсивності виділених компонентів вузької лінії визначаються кількістю резонуючих ядер водню з приблизно однаковою рухливістю, що відповідає різним фазовим станам метану. Це дає можливість практично оцінити співвідношення кількостей метану в різних фазах, орієнтуючись на ширини отриманих компонент вузької лінії, і, провівши додатковий калібрувальний експеримент, перейти до вагових або об'ємних співвідношень.

В новому ПЗ було створено модуль розрахунку на мові програмування Python. За основу розрахункової моделі, була використана методика, розроблена в [5]. Попередня версія модулю має суцільну структуру, але в майбутньому при необхідності можливо буде поділити її на структурні підмодулі.

Щоб описати розрахункову модель у коді Python було створено class SpectrCalculation (відповідає за модель розрахунку спектрів). Його використовують для того, щоб модель було «видно» у інших функціях.

```
class SpectrCalculation:
    def __init__(self) -> None:
        self.c = 0
        self.iter =0
        pass

    def func1(self, x, a, b, c1, d, q, k):
          return a*(exp(-2*(x-b+0.32)**2/c1**2)-exp(-2*(x-b-0.32)**2/c1**2)) +
d*q**2*(1/(q**2 + (x-b+0.32)**2) - 1/(q**2+(x-b-0.32)**2)) - k

    def func2(self, x, a, b, d, q, w, v, k):
        print(f"c = {self.c}")
        print(f"interation =  {self.iter}")
        self.iter+=1
     return a*(exp(-2*(x-b+0.08)**2/self.c**2)-exp(-2*(x-b-0.08)**2/self.c**2))
+ d*(q**2)*(1/(q**2 + (x-b+0.08)**2) - 1/(q**2+(x-b-0.08)**2))
          +w*v*(1/(v**2 + (x-b+0.08)**2) - 1/(v**2+(x-b-0.08)**2))- k

    def setC(self, c):
        self.c = c
```

Для виводу графіків, використовуються вбудовані функції graph(x, y, label), з настроюваними параметрами – такими, як колір лінії, ширина лінії,

її вид, легенда. Ці функції є частинної бібліотек, що доступні в Python. Вбудовані функції, розроблені як частина мови програмування і розподілені на бібліотеки для спрощення роботи з написання власного коду. Таким чином, можливо використати функцію з бібліотеки замість написання власних функцій, це допомагає скоротити час на розробку програмних модулів та зменшити кількість строчок коду.

Щоб отримати данні з початкових спектрів, нам необхідно загрузити данні з файлів у програму. Для відкриття файлу використовуємо вбудовану функцію namfl1, яка за допомогою комунікації з оператором отримує ім'я файлу. Обов'язково вказувати з розширенням, адже програма може працювати з будь-яким форматом файлів. Після цього за допомогою функції dat1(filename) ми відкриваємо файл, та передаємо данні у масив, з яким потім працює програма.

Розрахунок спектрів відбувається за допомогою наступного коду:

```
# розрахунок повного спектру
calculater = SpectrCalculation()
x = ob_array[:,0]
y = ob_array[:,1]
t1 = np.array([2,0,6,0.7,0.3,0])
popt, pcov = curve_fit(calculater.func1, x, y, t1)
c = popt[2]
print(f"set c = {c}")
calculater.setC(c)
a = popt[0]
a_min = 0.9*a
min = 0.5*popt[3]
print("Povnii: ")
print(*popt)
# розрахунок спектру вузької лінії
x1 = uz_array[:,0]
y1 = uz_array[:,1]
t2 = np.array([a, 0.3, 0.5, 0.2,0.5,0.5,0])
 popt1, pcov1 = curve_fit(calculater.func2, x1, y1, t2, bounds=((a_min, -np.inf, min, -np.inf, min, -np.inf, -np.inf),(np.inf, np.inf, np.inf, 2, np.inf, 2, np.inf)))

print("Vuzka: ")
print(*popt1)

graph1(x1,y1, 'повний спектр')
graphfunc2(x1,calculater.func2(x1,*popt1), 'вузька лінія')
plt.legend(fontsize=14)
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

Модуль розрахунку нашого приладу може бути використаний як складова частина загального ПО, а також як самостійна програма. Для коректної роботи модулю досліджені можливі «збої» розрахунку та отримання результату без втручання оператора.

Наразі розрахунок проходить за наступним алгоритмом:

1. Розрахунок спектру широкої лінії, та встановлення коефіцієнтів.
2. Розрахунок спектру вузької лінії, з урахуванням коефіцієнтів широкої лінії, які ми отримали на першому кроці.

При розрахунку спектру широкої лінії не виникає проблем, тому код розрахунку не потребує введення додаткових компонентів для обмеження ступенів свободи розрахунку. У той же час вузька лінія має більш складний характер розрахунку. Для її розрахунку ми використовуємо функцію, що є суперпозицією декількох функцій, оскільки ми намагаємось отримати данні, які залежать від внеску різних компонент флюїду. Для вузької лінії дуже важливим параметром є якість записаного спектру. Тому через введення додаткових обмежень у параметри розрахунку можна знизити вплив «зашумленості» спектру. У приведеній частині коду спектр вузької лінії розподіляється на 3 компоненти.

```
# розрахунок спектру вузької лінії
t2 = np.array([a, 0.3, 0.5, 0.2,0.5,0.5,0])
 popt1, pcov1 = curve_fit(calculater.func2, x1, y1, t2, bounds=((a_min, -np.inf, min, -np.inf, min, -np.inf, -np.inf),(np.inf, np.inf, np.inf, 2, np.inf, 2, np.inf)))
```

Іноді при розрахунку спектрів підсушеного вугілля, розрахунок спектру вузької лінії дає збій, через малі значення тих чи інших параметрів різних компонентів флюїду. Алгоритм роботи модулю має наступний вигляд:

1. Розрахунок спектру широкої лінії та встановлення коефіцієнтів.
2. На базі коефіцієнтів, отриманих для широкої лінії, видаляється вплив компонентів широкої лінії зі спектру вузької лінії.
3. Розрахунок спектру вузької лінії з розділенням її на 3 компоненти.
4. Якщо при розрахунку вузької лінії з виділенням трьох компонент проходить збій розрахунку, або в результаті отримано помилкові данні – проводиться розрахунок спектру вузької лінії з урахуванням двох компонент (лоренцевої + гаусової форми лінії) та ще один розрахунок з урахуванням двох компонент (лоренцевої + лоренцевої форми лінії).
5. В результаті розрахунку отримуємо параметри амплітуд та ширин компонентів лінії. При розрахунку на дві компоненти спочатку відображається напис: «Нажаль, розрахунок за трьохкомпонентною моделлю не можливо здійснити. Було проведено розрахунок за двохкомпонентною моделлю».

Розробка інтерфейсу користувача розподілялася на декілька етапів: перший етап – розробка зовнішнього вигляду інтерфейсу користувача; другий етап – поєднання зовнішнього вигляду та модулів програми, формування правил роботи модулів при визові їх через структурні кнопки, та командні строки; третій етап – формування БД для збереження даних.

View – зовнішній вигляд. До нього входить набір класів, які формують графічний інтерфейс користувача.

```
class ExperimentLayout(QHBoxLayout):
    def __init__(self) -> None:
        super().__init__()
        self.__description = LineEdit("Description")
        self.__dateCreate = LineEdit("Date Create")
        self.__lastChange = LineEdit("Last update")
        self.__status = LineEdit("Status")
        self.__saveButton = QPushButton(text="Save")
        self.addLayout(self.__description)
        self.addLayout(self.__dateCreate)
        self.addLayout(self.__lastChange)
        self.addLayout(self.__status)
        self.addWidget(self.__saveButton)


class MeasurementParametrsLayout(QHBoxLayout):
    def __init__(self):
        super().__init__()
        self.__lineWidth = LineEdit("line width")
        self.__modulation = LineEdit("modulation")
        self.__type = LineEdit("Type")
        self.addLayout(self.__lineWidth)
        self.addLayout(self.__modulation)
        self.addLayout(self.__type)

class MeasurementTitleLayout(QVBoxLayout):
    def __init__(self):
        super().__init__()
        title = QHBoxLayout()
        self.__name = LineEdit("Description")
        self.__status = LineEdit("Status")
        self.__action = QPushButton(text="Action")
        title.addLayout(self.__name)
        title.addLayout(self.__status)
        title.addWidget(self.__action)
        self.addLayout(title)

class MeasurementResultLayout(QHBoxLayout):
    def __init__(self):
        super().__init__()
        self.__table = QTableView()
        self.addWidget(self.__table)
```

Presenter – головна структура, яка надає данні для відображення у першому етапі. Також на цьому етапі створюється логіка верифікації усіх да-

них, прописується зв’язок верифікації даних з іншими програмними модулями, наприклад, модулем розрахунку.

```
class Presenter():
    def __init__(self, model:Model) -> None:
        self.__model = model
        self.__listExperimentView = MainMenu()
        self.__experimentView = ExperimentLayout()
        self.__measurementView = MeasurementLayout()
        listExperiment = self.__model.getExperimentList()
        self.__listExperimentView.setData(listExperiment)
    def getListExperemetView(self):
        return self.__listExperimentView
    def getExperemetView(self):
        return self.__experimentView
    def getMeasurementView(self):
        return self.__measurementView
```

Збереження БД. Для збереження нових експериментів, а також результатів розрахунку. Створено базу даних. Фрагмент коду опису БД:

```
class ExperimentStatus(Enum):
    EDIT = "Edit"
    PROCESING = "Procesing"
    CALCULATE = "Calculate"
    PARTLY_COMLETE = "partly_complete"
    COMLETE = "complete"

class MeteringStatus(Enum):
    EDIT = "Edit"
    PROCESING = "Procesing"
    CALCULATE = "Calculate"
    PARTLY_COMLETE = "partly_complete"
    COMLETE = "complete"
```

Кожен клас відповідає за той чи інший запис, який має свій унікальний статус. Статуси запису можуть мати наступні параметри: редагувати, обробка, розрахунок, частково завершений, завершений.

Зараз інтерфейс користувача має вигляд, показаний на рисунку 2.

Зовнішній вигляд інтерфейсу користувача створювався за допомогою модулю PyQt. PyQt – оболонка на мові програмування Python для бібліотеки Qt. Бібліотека реалізована в Python-модулях, та охоплює близько 1000 класів. Підтримуються операційні системи Microsoft Windows, Linux, OS X, iOS та Android [6]. PyQt також включає Qt Designer (Qt Creator) – дизайнер графічного інтерфейсу користувача. Програма PyQt генерує Python-код із файлів, створених у Qt Designer.

У відкритому вікні інтерфейсу будуть відображатись лист експериментів; лист розрахованих моделей; короткий опис експерименту; дата створення

експерименту, а також дата останньої модернізації (внесення змін до опису чи проведення додаткового розрахунку); також для запису ліній необхідно вказати параметри спектрів (вносяться попарно), та кількість записів (із розрахунку парності спектрів); також тут будуть виведені результати розрахунку у вигляді списку параметрів, та графік залежності параметрів.

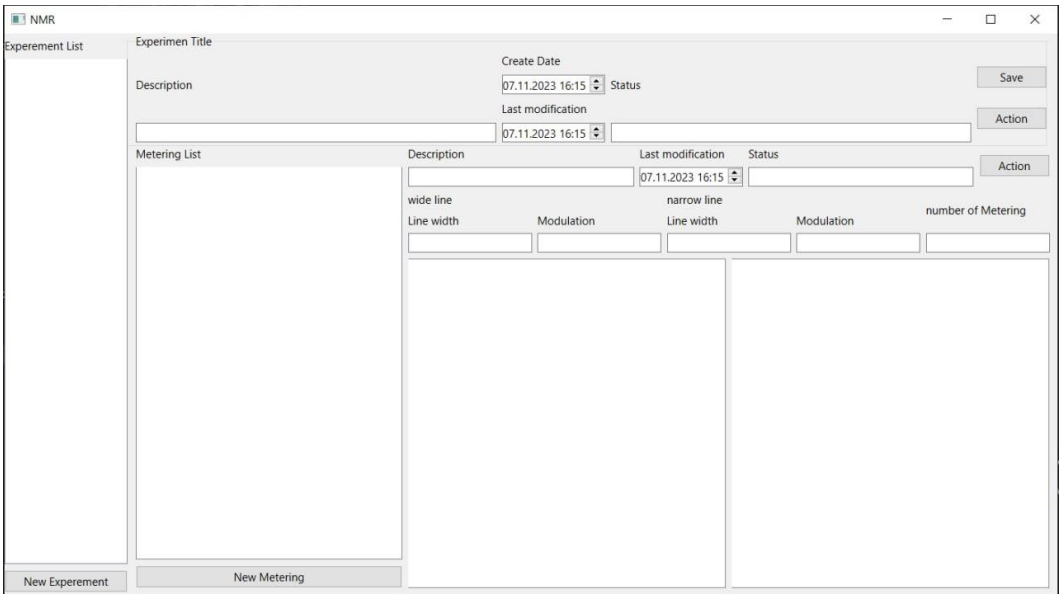

Рисунок 2. Інтерфейс користувача у новому вигляді

4. ВИСНОВКИ

Для оснащення портативного ЯМР-спектрометра запропоновано використати єдину систему для побудови програмного забезпечення. Створений програмний продукт розроблено за допомогою мови програмування Python. Він буде враховувати всі потреби при реєстрації та обробці складних спектрів вугільних зразків. Розроблено програму, яка буде забезпечувати керування роботою дискретних та аналогових каналів виведення, та встановлювати параметри, що необхідні для запису, обробки та інтерпретації спектрів.

У випадку дослідження вугілля, що містить невелику кількість флюїду, запропоновано змінити алгоритм дії розрахункової програми з урахуванням можливих збоїв. Також в розрахунок спектру вузької лінії було включено деякі обмеження, наприклад амплітуда лінії не може бути меншою за нуль, оскільки для математичного розрахунку це можливо, але з фізичного сенсу цих змінних така поведінка не допускається. Введенні обмеження, які допомагають знизити вплив «зашумленості» спектру.

Для підсушеного вугілля при отриманні помилки розрахунку за трьохкомпонентною моделлю був введений автоматичний розрахунок спектру з розподілом вузької лінії на дві компоненти. У даний час програма розраховує загальний спектр, та спектр вузької лінії – з поділом її на три компоненти, з можливістю автоматичного розрахунку за двохкомпонентною моделлю.

СПИСОК ЛІТЕРАТУРИ

1. Алексеев А.Д. (2010*). Физика угля и горных процессов*. Київ. 424 с.

2. Костенко И.Г., Молчанов А.Н., Сапунов Е.П., Службин Ю.А., Пичка Т.В. (2012). Встроенная система обработки данных для портативного спектрометра ЯМР. *Физико-технические проблемы горного производства*, 15, 20-25.

3. Python Software Foundation: https://www.python.org/.

4. Молчанов А.Н. (2011) Усовершенствованный комплекс оборудования для исследования сорбционных свойств ископаемых углей. *Физико-технические проблемы горного производства*, 4, 42-53.

5. Молчанов А.Н., Троицкий Г.А., Пичка Т.В. (2015). Интерполяция сложных спектров ЯМР ^{1}Н ископаемых углей. *Геотехническая механика*, 124, 85-96.

6. Riverbank Computing. What is PyQt?: https://riverbankcomputing.com/software/pyqt/intro.

REFERENCES

1. Alexeev, A.D. (2010), *Fizika uglja i gornih protsesov* [Coal and mining processes physics], Naukova Dumka, Kiev, Ukraine.

2. Kostenko, I.G., Molchanov, A.N., Sapunov, E.P., Sluzbin, U.A., Pichka, T.V. (2012). Build-in data processing system for portable NMR spectrometr. *Fiziko-tehnicheskie problemi gornogo proizvodstva*, 15, 20-25.

3. Python Software Foundation: https://www.python.org/.

4. Molchanov, A.N. (2011), “Improved equipment complex for studying of coal sorption properties”. *Fiziko-tehnicheskie problemi gornogo proizvodstva*, Vol. 14, pp. 42-53.

5. Molchanov, A.N., Troitskii, G.A., Pichka, T.V (2015). Complex ^{1}H NMR fossil coals spectra interpolation. *Geotehnicheskay mehanika*, 124, 85-96.

6. Riverbank Computing. What is PyQt?: https://riverbankcomputing.com/software/pyqt/intro.

ABSTRACT (IN UKRAINIAN)

Мета. Модернізація програмного забезпечення, що забезпечує роботу портативного спеціалізованого спектрометру ядерного магнітного резонансу на ядрах водню ^{1}Н, призначеного для дослідження зразків викопного вугілля, обробку та інтерпретацію результатів досліджень.

Методика. Для досягнення мети використано універсальну мову програмування загального призначення Python, яка дала можливість працювати над розробкою всіх модулів програмного забезпечення для спеціалізованого спектрометру ЯМР.

Результати. Для поліпшення роботи портативного спектрометру ЯМР створено нове програмне забезпечення, яке складається із трьох частин: інтерфейсу користувача, модуля керування приладом та розрахункового модуля. Кожен модуль має свої задачі та відповідає за виконання найбільш важливих функцій. Для зберігання даних експериментів та розрахунків буде використано структуровану систему зберігання даних, що утворює Базу Даних. Для побудови програмних модулів було враховано технічні характеристики та особливості роботи спеціалізованих спектрометрів ЯМР, що використову-

ються для дослідження викопного вугілля, які призначені для реєстрації ЯМР на протонах (^{1}H). В новому ПЗ було створено модуль розрахунку на мові програмування Python. Модуль розрахунку приладу може бути використаний як складова частина загального ПО, а також як самостійна програма. Надано опис алгоритму, за яким виконується обробка складних експериментальних спектрів. Оновлений інтерфейс користувача поліпшує та спрощує роботу оператора зі спектрометром.

Наукова новизна. Для оснащення портативного ЯМР-спектрометра запропоновано використати єдину систему для побудови програмного забезпечення. Створений програмний продукт розроблено за допомогою мови програмування Python. Він буде враховувати всі потреби при реєстрації та обробці складних спектрів вугільних зразків. Розроблено програму, яка буде забезпечувати керування роботою дискретних та аналогових каналів виведення, та встановлювати параметри, що необхідні для запису, обробки та інтерпретації спектрів.

Практична значимість. Модернізація програмного забезпечення, що підтримує роботу спеціалізованих спектрометрів ЯМР для дослідження вугільних зразків і, в тому числі, мобільного портативного ЯМР-спектрометра, який призначений для використання в умовах шахтної лабораторії, дозволить поліпшити виконання експрес-аналізу проб вугілля, покращить якість проведення аналізу та підвищить надійність рекомендацій, що надаються на основі проведеного аналізу, для виконання робіт з видобутку вугілля.

Ключові слова: спектрометр, ядерний магнітний резонанс, вугілля, метан, автоматизація наукового експерименту, лабораторний прилад

ABOUT AUTHORS

Molchanov Oleksandr, Doctor of Engineering Sciences, (D.Sc.), Professor, Director of the Branch for Physics of Mining Processes of the M.S. Poliakov Institute of Geotechnical Mechanics of the National Academy of Sciences of Ukraine, 15 Simferopolskaya Street, Dnipro, Ukraine, 49005. E-mail: molchanov@nas.gov.ua

Pichka Tetiana, Candidate of Technical Sciences (Ph.D), Senior Researcher in Department of Physics of Sorption Processes, Branch for Physics of Mining Processes of the M.S. Poliakov Institute of Geotechnical Mechanics of the National Academy of Sciences of Ukraine, 15 Simferopolskaya Street, Dnipro, Ukraine, 49005. E-mail: teaodinn@gmail.com