Форма № Н-9.02

Житомирський державний університет імені Івана Франка

Фізико-математичний факультет

Кафедра прикладної математики та інформатики

Пояснювальна записка

до дипломного проекту (роботи)

Магістр

на тему:

«Порівняльний аналіз методів чисельного обчислення інтегралів»

Виконав:
студент 6 курсу 63 групи
напряму підготовки
8.04030201 «Інформатика*»
Ольшевський І.В.
Керівник: доктор педагогічних наук
Спірін О.М.
Рецензент: кандидат технічних наук
Морозов А.В.

Житомир – 2015

ЗМІСТ

ЗМІСТ 2

ВСТУП 4

РОЗДІЛ І. ТЕОРЕТИЧНА ЧАСТИНА 7

1.1. Обчислювальна математика. Чисельні методи та чисельне інтегрування. 7

1.2. Чисельне інтегрування та його методи. Чисельні квадратури. 10

1.2.1. Метод прямокутників 12

1.2.2. Метод трапецій 14

1.2.3. Метод Сімпсона (парабол) 15

1.2.4. Метод Гауса 16

1.2.5. Квадратура Чебишева 17

1.3. Неквадратурні методи інтегрування 21

1.3.1. Метод Монте-Карло 21

1.3.2. Інтегрування з допомогою сплайнів 22

РОЗДІЛ ІІ. ПРАКТИЧНА ЧАСТИНА 26

2.1. Технічний огляд дослідження: опис використаних програмних засобів .. 26

2.1.1. Python 26

2.1.2. JetBrains PyCharm 31

2.1.3. Git 32

2.1.4. BitBucket 36

2.1.5. Python(x,y) 37

2.1.6. Kivy 38

2.2. Програмна реалізація методів чисельного інтегрування засобами мови Python та фреймворку Kivy 39

2.2.1. Модуль «main» 40

2.2.2. Модуль «NumComparator» 41

2.2.3. Модуль «Binders» 47

2.2.4. Модуль «MathParser» 49

2.2.5. Модуль «Methods» 53

2.2.6. Модуль «Utils» 59

2.2.6. Модуль «Utils» 59

2.2.7. Модуль «README» 63

ВИСНОВКИ 67

ЛІТЕРАТУРА 69

ВСТУП

На сучасному етапі набувають масового характеру інформаційні інтелектуальні процеси. Зріс об’єм інформаційних потоків і високотехнологічні виробництва висувають підвищені вимоги до фахівця XXI ст., які вимагають високої професійної компетентності спеціалістів у знанні сучасних інформаційних технологій та їх використанні. Тому сучасному фахівцеві необхідно безупинно підвищувати свою кваліфікацію.

На практиці в багатьох випадках знайти точне рішення виниклої математичної задачі не вдається. Це відбувається не тому, що ми не вміємо цього зробити, так як дані рішення зазвичай не виражаються в звичних для нас елементарних або інших відомих функціях. Тому важливе значення набули чисельні методи, особливо у зв'язку зі зростанням ролі математичних методів в різних галузях науки і техніки і з появою високопродуктивних ЕОМ.[45]

Актуальність обраної теми обумовлена необхідністю визначення оптимального методу чисельного інтегрування для знаходження максимально точного результату з мінімальними затратами ресурсів.

Мовою розробки було обрано Python - високорівневу мову програмування загального призначення, орієнтовану на підвищення продуктивності розробника і полегшене читання коду. Для розробки інтерфейсу користувача було обрано фреймворк Kivy, системою контролю версій – Git, хостингом репозитарію – BitBucket. Тестування методів та логіки користувача проходило в Spyder2 з допомогою numpy та matplotlib, які є частинами дистрибутиву Python(x,y), проте основним середовищем розробки було PyCharm від JetBrains.

Інформаційне суспільство сьогодення вимагає змін стратегії освіти, одним з головних чинників якої є широке використання інформаційних технологій. Внаслідок чого основним завданням вищої освіти в сучасних

умовах є формування в майбутніх фахівців наукового мислення, навичок самостійного засвоєння і критичного аналізу нових відомостей, уміння будувати наукові гіпотези і планувати експеримент щодо їх перевірки. Вирішення цієї задачі неможливе без широкого використання нових інформаційних технологій. Це обумовило в Україні якісну активізацію процесів інформатизації освіти.

У зв'язку з цим особливої актуальності набувають загальні наукові, методологічні та технологічні проблеми, пов'язані з організацією процесів створення, супроводу і ефективного використання програмних засобів навчального призначення протягом їх життєвого циклу. Вагомий внесок у розвиток теорії та практики використання інформаційних комп'ютерних технологій у галузі освіти внесли вітчизняні вчені: Г.О. Атанов, Г.О. Балл, В.Ю. Биков, М.М. Глибовець, В.М. Глушков, В.І. Гриценко, О.М. Довгялло, М.І. Жалдак, М.З. Згуровський, С.П. Кудрявцев, А.Ф. Манако, Г.Ю. Маклаков, Є.І. Машбіць, Н.В. Морзе, Н.Д. Панкратова, С.А. Раков, К.М. Синиця, О.В. Співаковський, В.А. Широков та ін.

Мета дослідження – теоретично обґрунтувати використання інформаційних технологій у вищих закладах освіти та розробити мультимедійний програмний комплекс з дисципліни "Чисельні методи: чисельне інтегрування".

Об'єкт дослідження – методи проектування та технології реалізації програмних засобів навчального призначення.

Предмет дослідження – мультимедійний програмний комплекс з дисципліни "Чисельні методи: чисельне інтегрування"

Гіпотеза дослідження. За умов використання досвіду з методів проектування та технологій реалізації програмних засобів прикладного призначення, проаналізованого набору технологій, можливо швидко і якісно

розробити мультимедійний програмний комплекс.

Завдання дослідження:

1. Вивчити сучасні підходи у науковій літературі про роль та місце чисельних методів в комп’ютерних технологіях.
2. Розглянути інструментальні засоби створення програмних комплексів.
3. Проаналізувати програми для створення програмних комплексів.

Спроектувати та розробити мультимедійний комплекс з дисципліни “Чисельні методи: чисельне інтегрування”, а саме програму для обчислення інтегралів чисельними методами.

Практичне значення дослідження полягає у тому, що з’являється можливість обрати найбільш оптимальний алгоритм чисельного інтегрування.

РОЗДІЛ І. ТЕОРЕТИЧНА ЧАСТИНА

1.1. Обчислювальна математика. Чисельні методи та чисельне інтегрування.

Обчислювальна математика - розділ математики, що включає коло питань, пов'язаних з виробництвом різноманітних обчислень. У вужчому розумінні обчислювальна математика - теорія чисельних методів вирішення типових математичних задач. Сучасна обчислювальна математика включає в коло своїх проблем вивчення особливостей обчислення із застосуванням комп'ютерів.[43]

Обчислювальна математика володіє широким колом прикладних застосувань для проведення наукових та інженерних розрахунків. На її основі в останнє десятиліття утворилися такі нові галузі природничих наук, як обчислювальна хімія, обчислювальна біологія і так далі.

Історія

Обчислювальна математика виникла досить давно. Ще в Стародавній Месопотамії були розроблені методи отримання квадратного кореня. В епоху наукової революції обчислювальна математика розвивалася швидкими темпами з практичних застосувань паралельно з математичним аналізом. Крім цього, подібні обчислення широко застосовувалися в небесній механіці для передбачення траєкторії руху небесних тіл. Це призвело до появи таких найважливіших складових фізики, як теорія про геліоцентричної системи устрою світу, закони Кеплера і закони Ньютона.[28] XVII і XVIII століття стали часом розробки значної кількості чисельних методів і алгоритмів. Застосування великої кількості інженерних обчислень в XIX і XX століттях зажадало створення відповідних приладів. Одним з таких приладів стала логарифмічна лінійка, також з'явилися таблиці значень функцій з точністю до 16 знаків після коми, допомагали проводити обчислення.[45] Також існували механічні

пристрої для виконання математичних операцій, що називалися арифмометрами. У першій половині XX століття для вирішення диференціальних рівнянь стали активно використовуватися аналогові ЕОМ. Винахід комп'ютера в середині XX століття означало створення універсального інструменту для математичних обчислень. Спільно з мейнфреймами в розпорядженні інженерів і вчених для виконання ручних операцій були тільки калькулятори, які активно використовувалися аж до початку масового виробництва персональних комп'ютерів.

Основні напрямки. У обчислювальної математики виділяють наступні напрямки: аналіз математичних моделей, розробка методів і алгоритмів вирішення стандартних математичних задач, автоматизація програмування. Аналіз обраних математичних моделей для поставленої задачі починається з аналізу і обробки вхідної інформації, що дуже важливо для більш точних вхідних даних. Для такої обробки найчастіше застосовуються методи математичної статистики. Наступним кроком є чисельне рішення математичних задач і аналіз результатів обчислень. Ступінь достовірності результатів аналізу повинна відповідати точності вхідних даних. Поява більш точних вхідних даних може зажадати удосконалення побудованої моделі або навіть її заміну.Методи та алгоритми вирішення типових математичних задач із застосуванням обчислювальної техніки звуться чисельних методів. До типових завдань відносять:

- Алгебра: рішення систем лінійних рівнянь, звернення матриць, пошук власних значень і векторів матриць (обмежена та повна проблема власних значень), пошук сингулярних значень і векторів матриць, рішення нелінійних алгебраїчних рівнянь, рішення систем нелінійних алгебраїчних рівнянь;
- Диференціальні рівняння: диференціювання та інтегрування функцій одного або декількох змінних, рішення звичайних диференціальних рівнянь, рішення рівнянь з приватними похідними, рішення систем

диференціальних рівнянь, рішення інтегральних рівнянь;

- Оптимізація: вивчення мінімальних і максимальних значень функціоналів на множинах;
- Дослідження операцій та теорія ігор: мінімаксні задачі (зокрема, для багатокрокових ігор);
- математичне програмування: задачі апроксимації, задачі інтерполяції, завдання екстраполяції.[26]

Проводиться вивчення та порівняльний аналіз методів вирішення типових завдань. Важливим елементом аналізу є пошук економічних моделей, що дозволяють отримати результат, використовуючи найменшу кількість операцій, оптимізація методів рішення. Для задач великих розмірів особливо важливим є дослідження стійкості методів і алгоритмів, у тому числі до помилок округлення. Прикладами нестійких завдань є зворотні задачі (зокрема, пошук зворотної матриці), а також автоматизація обробки результатів експериментів. Постійно збільшується коло типових завдань і зростання числа користувачів визначили підвищення вимог до автоматизації. В умовах, коли знання конкретних чисельних методів є несуттєвим для користувача, зростають вимоги до стандартних програм рішення. З їх використанням не потрібно програмування методів рішення, а досить задати вихідну інформацію.[27]

Програмне забезпечення. Алгоритми вирішення безлічі стандартних задач обчислювальної математики реалізовані на різних мовах програмування. Найчастіше для цих цілей використовуються мови Фортран і С, бібліотеки для яких можна знайти в репозиторії Netlib. Крім того, велику популярність мають комерційні бібліотеки IMSL і NAG, а також вільна GNU Scientific Library. Програмні пакети MATLAB, Mathematica, Maple, S-PLUS, LabVIEW і IDL, а також їхні вільні альтернативи FreeMat, Scilab, GNU Octave (схожа на Matlab), IT ++ (бібліотека C ++), R (схожий на S-PLUS) має різні чисельні методи, а також засоби для візуалізації та відображення результатів. Багато систем комп'ютерної алгебри, такі як Mathematica, мають можливість задавати

необхідну арифметичну точність, що дозволяє отримати результати більш високої точності. Також, більшість електронних таблиць можуть бути використані для вирішення простих задач обчислювальної математики.

Обчислювальні методи

Обчислювальні (чисельні) методи - це методи розв'язання математичних задач в чисельному вигляді Подання як вихідних даних в задачі, так і її вирішення - у вигляді числа або набору чисел. В системі підготовки інженерів технічних спеціальностей є важливою складовою. Основами для обчислювальних методів є:

- рішення систем лінійних рівнянь;
- інтерполювання і наближене обчислення функцій;
- чисельне інтегрування;
- чисельне рішення системи нелінійних рівнянь;
- чисельне рішення звичайних диференціальних рівнянь;
- чисельне рішення рівнянь в приватних похідних (рівнянь математичної фізики);
- рішення задач оптимізації.

1.2. Чисельне інтегрування та його методи. Чисельні квадратури.

Чисельне інтегрування (історична назва: (чисельна) квадратура) - обчислення значення певного інтеграла (як правило, наближене). Під чисельним інтегруванням розуміють набір чисельних методів для знаходження значення певного інтеграла.

Чисельне інтегрування застосовується, коли:

Сама підінтегральна функція не задана аналітично. Наприклад, вона

представлена у вигляді таблиці (масиву) значень у вузлах деякої розрахункової сітки.

Аналітичне подання підінтегральної функції відомо, але її первісна не виражається за через аналітичні функції. Наприклад,

$$f(x) = e(-x^2)$$

У цих двох випадках неможливо обчислення інтеграла за формулою Ньютона-Лейбніца. Також можлива ситуація, коли вид первісної настільки складний, що швидше обчислити значення інтеграла чисельним методом. Основна ідея більшості методів чисельного інтегрування полягає в заміні підінтегральної функції на більш просту, інтеграл від якої легко обчислюється аналітично. При цьому для оцінки значення інтеграла виходять формули виду

$$I \approx \sum_{i=1}^{n} \omega_i f(x_i)$$

де n - число точок, в яких обчислюється значення підінтегральної функції. Точки x_i називаються вузлами методу, числа ω_i вагами вузлів. При заміні підінтегральної функції на поліном нульового, першого та другого степенів виходять відповідно методи прямокутників, трапецій і парабол (Сімпсона). Часто формули для оцінки значення інтеграла називають квадратурними формулами.

Окремим випадком є метод побудови інтегральних квадратурних формул для рівномірних сіток, відомий як формули Котеса. Метод названий на честь Роджера Котса. Основною ідеєю методу є заміна підінтегральної функції яким-небудь інтерполяційним многочленом. Після взяття інтеграла можна написати

$$\int_a^b f(x)dx = \sum_{i=0}^{n} H_i f(x_i) + r_n(f)$$

де числа H_i називаються коефіцієнтами Котеса і обчислюються як інтеграли від відповідних многочленів, що стоять у вихідному інтерполяційних многочлене для підінтегральної функції при значенні функції у вузлі

$$x_i = a + ih$$

$$h = (b-a) / n$$

де h - крок сітки; n - число вузлів сітки, а індекс вузлів $i = 0 \ldots n$. Доданок $r_n (f)$ - похибка методу, яка може бути знайдена різними способами. Для непарних $n..1$ похибка може бути знайдена інтегруванням похибки інтерполяційного полінома підінтегральної функції. Окремими випадками формул Котеса є: формули прямокутників (n = 0), формули трапецій (n = 1), формула Сімпсона (n = 2), формула Ньютона (n = 3) і т.д.[41]

1.2.1. Метод прямокутників

Метод прямокутників - метод чисельного інтегрування функції однієї змінної, що полягає в заміні підінтегральної функції на многочлен нульової ступеня, тобто константу, на кожному елементарному відрізку. Якщо розглянути графік підінтегральної функції, то метод полягатиме в наближеному обчисленні площі під графіком підсумовуванням площ кінцевого числа прямокутників, ширина яких визначатиметься відстанню між відповідними сусідніми вузлами інтегрування, а висота - значенням підінтегральної функції в цих вузлах. Алгебраїчний порядок точності дорівнює 0. (Для формули середніх прямокутників дорівнює 1).

Якщо відрізок $[a,b]$ є елементарним і не піддається подальшому розбиттю, значення інтеграла можна знайти за

1. Формулою лівих прямокутників:

$$\int_a^b f(x)dx \approx f(a)(b-a).$$

2. Формулою правих прямокутників

$$\int_a^b f(x)dx \approx f(b)(b-a).$$

3. Формулою центральних прямокутників

$$\int_a^b f(x)dx \approx f\left(\frac{a+b}{2}\right)(b-a).$$

У разі розбиття відрізка інтегрування на n елементарних відрізків наведені вище формули застосовуються на кожному з цих елементарних відрізків між двома сусідніми вузлами. В результаті, виходять складені квадратурні формули[35]

1. Формула лівих прямокутників:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} f(x_i)(x_{i+1}-x_i).$$

2. Формула правих прямокутників

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} f(x_i)(x_i-x_{i-1}).$$

3. Формула центральних прямокутників

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} f\left(\frac{x_i+x_{i+1}}{2}\right)(x_{i+1}-x_i) = \sum_{i=1}^{n} f\left(\frac{x_{i-1}+x_i}{2}\right)(x_i-x_{i-1}).$$

Формулу з обчисленням значення в середній між двома вузлами точці можна застосовувати лише тоді, коли подинтегральная функція задана аналітично, або яким-небудь іншим способом, що допускає обчислення значення в довільній точці. В завданнях, де функція задана таблицею значень

залишається лише обчислювати середнє значення між інтегралами, підрахованими за формулами лівих і правих прямокутників відповідно, що призводить до складової квадратурной формулою трапецій.

Оскільки складові квадратурні формули є ні чим іншим, як сумами, що входять у визначення інтеграла Рімана, при $n \to \infty$ вони сходяться до точного значення інтеграла. Відповідно, із збільшенням n точність одержуваного по наближеним формулам результату зростає.

Похибка методу

1. Для формул правих і лівих прямокутників похибка становить

$$E(f) = \frac{f'(\xi)}{2}(b-a)^2.$$

2. Для формул центральних прямокутників похибка становить

$$E(f) = \frac{f''(\xi)}{24}(b-a)^3.$$

3. Для складеної формули прямокутників:

$$E(f) = \frac{f''(\xi)}{24}(b-a)h^2.$$

1.2.2. Метод трапецій

Метод трапецій - метод чисельного інтегрування функції однієї змінної, що полягає в заміні на кожному елементарному відрізку підінтегральної функції на многочлен першого ступеня, тобто лінійну функцію. Площа під графіком функції апроксимується прямокутними трапеціями. Алгебраїчний порядок точності дорівнює 1. Якщо відрізок $[a,b]$ є елементарним і не піддається подальшому розбиттю, значення інтеграла можна знайти за формулою[1]

$$\int_a^b f(x)dx = \frac{f(a)+f(b)}{2}(b-a) + E(f), \qquad E(f) = -\frac{f''(\xi)}{12}(b-a)^3.$$

Це просте застосування формули для площі трапеції - добуток напівсуми основ, якими в даному випадку є значення функції в крайніх точках відрізка, на висоту (довжину відрізка інтегрування). Похибка апроксимації можна оцінити через максимум другої похідної[31]

$$|E(f)| \le \frac{(b-a)^3}{12}\max_{x\in[a,b]}|f''(x)|.$$

Складена формула

Якщо відрізок розбивається вузлами інтегрування і на кожному з елементарних відрізків застосовується формула трапецій, то підсумовування дасть складену формулу трапецій[24]

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1}\frac{f(x_i)+f(x_{i+1})}{2}(x_{i+1}-x_i) = \frac{f(a)}{2}(x_1-a) + \sum_{i=1}^{n-1} f(x_i)\frac{x_{i+1}-x_{i-1}}{2} + \frac{f(b)}{2}(b-x_{n-1}).$$

$$x_j = a + jh, h = (b-a)/N$$

N - парне

1.2.3. Метод Сімпсона (парабол)

Формула Сімпсона (також Ньютона-Сімпсона) відноситься до прийомів чисельного інтегрування. Отримала назву на честь британського математика Томаса Сімпсона (1710-1761).

Суть методу полягає в наближенні підінтегральної функції на відрізку $[a,b]$ інтерполяційним многочленом другого ступеня $p_2(x)$, тобто наближення графіка функції на відрізку параболою. Метод Сімпсона має порядок похибки 4 і алгебраїчний порядок точності 3.[32]

Формулою Сімпсона називається інтеграл від інтерполяційного

многочлена другого ступеня на відрізку $[a,b]$

$$\int_a^b f(x)dx \approx \int_a^b p_2(x)dx = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right),$$ де

$f(a), f((a+b)/2), f(b)$ - значення функції у відповідних точках (на початку, середині та кінці відрізка, відповідно).[24]

Похибка методу

За умови, що у функції $f(x)$ на відрізку $[a,b]$ існує четверта похідна, похибка $E(f)$, згідно знайденою Джузеппе Пеано формулою, дорівнює:

$$E(f) = -\frac{(b-a)^5}{2880} f^{(4)}(\zeta), \quad \zeta \in [a,b].$$

У зв'язку з тим, що значення ζ часто невідомо, для оцінки похибки використовується наступна нерівність:

$$|E(f)|, \frac{(b-a)^5}{2880} \max_{x\in[a,b]} \left| f^{(4)}(x)\right|.$$

1.2.4. Метод Гауса

Метод Гауса - метод чисельного інтегрування, що дозволяє підвищити алгебраїчний порядок точності методів на основі інтерполяційних формул шляхом спеціального вибору вузлів інтегрування без збільшення числа використовуваних значень підінтегральної функції. Метод Гауса дозволяє досягти максимальної для даного числа вузлів інтегрування алгебраїчної точності.[34]

Наприклад, для двох вузлів можна отримати метод 3-го порядку точності

$$I \approx \frac{b-a}{2}\left(f\left(\frac{a+b}{2} - \frac{b-a}{2\sqrt{3}}\right) + f\left(\frac{a+b}{2} + \frac{b-a}{2\sqrt{3}}\right)\right)$$

Тоді як для рівновіддалених вузлів методу вище 2-го порядку отримати неможливо. У загальному випадку, використовуючи n точок, можна отримати метод з порядком точності 2n-1. Значення вузлів методу Гауса по n точкам є коренями полінома Лежандра степені n. Значення ваг обчислюються за формулою

$$a_i = \frac{2}{(1-x_i^2)[P_{n'}(x_i)]^2}$$, де

P_n' - перша похідна полінома Лежандра.

Для n = 3, вузли і ваги мають такі значення:

$x_{1,3} = \pm\sqrt{0.6}, x_2 = 0,$ ваги: $a_{1,3} = \frac{5}{9}, a_2 = \frac{8}{9}.$

(Поліном визначений на відрізку [-1,1]).

1.2.5. Квадратура Чебишева

Як відомо, обчислення певного інтеграла зводиться до обчислення площі криволінійної трапеції, обмеженої кривими

$$x = 0,\ y = a,\ y = b,\ y = f\ (x).$$

Нехай для функції y =f (x) відомі в n+1 точках X0, X1, X2..Xn проміжку [a, b] відповідні визначення f (xi) = yi (i = 0,1,2..n). За заданим значенням Yi будуємо поліном Лагранжа, замінюючи f (x) поліномом Ln (x), де Rn (f) - похибка квадратурної формули. Скориставшись виразом для Ln (x), отримаємо наближену квадратурну формулу.[33]

Зауваження

коефіцієнти Ai при даному розташуванні вузлів не залежить від вибору

функції f (x);

для полінома ступеня n остання формула точна.

Вважаючи, що y = xK (k = 0,1,2 .., n), отримаємо лінійну систему з n + 1 рівнянь, де (k = 0,1, .., n), з якої можна визначити коефіцієнти A0, A1, .., AN. Детермінант системи є детермінантом Вандермонда. Але також необхідно зауважити, що при застосуванні даного методу фактично побудова полінома Лагранжа Ln (x) є зайвим. Простий метод підрахунку похибки квадратурних формул розроблений С.М. Нікольським[22]. Застосовуючи метод трапецій і середніх прямокутників інтеграл буде чисельно дорівнювати сумі площ прямокутних трапецій, де підстава трапеції яка-небудь мала величина (точність), і сумі площ прямокутників, де підставу прямокутника якась мала величина (точність), а висота визначається по точці перетину верхнього підстави прямокутника, графік функції повинен перетинати в середині.[23] Застосувавши формулу Сімпсона до кожного подвоєного проміжку [x0, x2], [x2, x4] ... [x2m-2, x2m] довжини 2h і ввівши позначення s1 = y1 + y2 + ... + y2m-1 s2 = y2 + y4 + ... + y2m отримаємо узагальнену формулу Сімпсона і залишковий член формули Сімпсона в загальному вигляді, де xk I (x2к-2, x2к). Розглянемо квадратурну формулу Чебишева: нехай дана функція f (x) у вигляді многочлена f (x) = ao + a1x + ... + anxn. Проінтегрувавши, перетворивши і підставивши значення многочлена у вузлах

$$f(x_1)=a_0+a_1x_1+a_2x_{12}+a_3x_{13}+...+a_nx_{1n}$$

$$f(x_2)=a_0+a_1x_2+a_2x_{22}+a_3x_{23}+...+a_nx_{2n}$$

$$f(x_3)=a_0+a_1x_3+a_2x_{32}+a_3x_{33}+...+a_nx_{3n}$$

. . . . . .

$$f(x_n)=a_0+a_1x_n+a_2x_{n2}+a_3x_{n3}+...+a_nx_{nn}$$

отримаємо формулу Чебишева.

Значення x1, x2, .., xn для різних n наведені нижче в таблиці

n	I	t_i
2	1;2	± 0,577350
3	1;3	± 0,707107
	2	0
4	1;4	± 0,794654
	2;3	± 0,187592
5	1;5	± 0,832498
	2;4	± 0,374541
	3	0
6	1;6	± 0,866247
	2;5	± 0,422519
	3;4	± 0,266635
7	1;7	±

		0,883862
2;6	±	
		0,529657
3;5	±	
		0,321912
4	0	

1.3. Неквадратурні методи інтегрування

1.3.1. Метод Монте-Карло

Метод Монте-Карло— загальна назва групи числових методів, основаних на одержанні великої кількості реалізацій стохастичного (випадкового) процесу, який формується у той спосіб, щоб його ймовірнісні характеристики збігалися з аналогічними величинами задачі, яку потрібно розв'язати. Використовується для розв'язування задач у фізиці, математиці, економіці, оптимізації, теорії управління тощо.

Інтегрування методом Монте-Карло
Припустимо, потрібно обчислити визначений інтеграл

$$\int_a^b f(x)dx$$

Розглянемо випадкову величину u, рівномірно розподілену на відрізку інтегрування [a, b]. Тоді f (u) також буде випадковою величиною, причому її математичне сподівання виражається як

$$\mathsf{E}f(u) = \int_a^b f(x)\varphi(x)dx,$$

де $\varphi(x)$ - щільність розподілу випадкової величини u, рівна $\frac{1}{b-a}$ на відрізку [a, b].

Таким чином, шуканий інтеграл виражається як

$$\int_a^b f(x)dx = (b-a)\mathsf{E}f(u).$$

Але математичне сподівання випадкової величини f (u) можна легко оцінити, змоделювавши цю випадкову величину і порахувавши вибіркове

середнє.

Отже, кидаємо N точок, рівномірно розподілених на [a, b], для кожної точки u_i обчислюємо $f(u_i)$. Потім обчислюємо вибіркове середнє

$$\frac{1}{N}\sum_{i=1}^{N} f(u_i).$$

У результаті отримуємо оцінку інтеграла:

$$\int_a^b f(x)dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(u_i)$$

Точність якого залежить лише від кількості точок N

1.3.2. Інтегрування з допомогою сплайнів

Введення

Ставиться завдання обчислити інтеграл виду

$$J = \int_a^b f(t)dt,$$

де a і b - нижня і верхня межі інтегрування; f (t) - неперервна функція на відрізку [a,b].

Нехай на відрізку інтегрування дана сітка, тобто мається сукупність вузлів де a і b - нижній і верхній межі інтегрування; f (t) - безперервна функція на відрізку [a, b].

$$\{t_i\}_{i=0}^{N}, i \in [a,b].$$

Тоді інтервал [a, b] розіб'ється на ділянки

$$\tau_i = t_i - t_{i-1}, i = 1,\ldots,N.$$

Нехай також відомі значення функції у вузлах цієї сітки, тобто задана таблиця

$$f_i = \{ f(t_i) \}_{i=0}^{N}.$$

Представимо інтеграл у вигляді суми інтегралів по частковим відрізкам:

$$\int_a^b f(t)dt = \sum_{i=1}^{N} \int_{t_{i-1}}^{t_i} f(t)dt.$$

Сутність більшості методів обчислення певних інтегралів полягає в заміні підінтегральної функції f(t) на відрізку $[t_{i-1}, t_i]$ апроксимуючої функцією $\varphi(t)$, для якої можна легко записати первісну в елементарних функціях, тобто

$$\int_{t_{i-1}}^{t_i} f(t)dt = \int_{t_{i-1}}^{t_i} \varphi(t)dt + R_i = S_i + R_i,$$

$$J = \sum_{i=1}^{N} \left(\int_{t_{i-1}}^{t_i} \varphi(t)dt + R_i \right) = S + R,$$

де S_i - наближене значення інтеграла на і-му відрізку, $S = \sum_{i=1}^{N} S_i$, а R_i - похибка обчислення інтеграла, $R = \sum_{i=1}^{N} R_i$,. Найкраще вивчена заміна $f(t)$ алгебраїчним многочленом.

Загальний випадок

Візьмемо в якості апроксимуючої функції кубічний сплайн:

$$\int_{t_{i-1}}^{t_i} f(t)dt = \int_{t_{i-1}}^{t_i} \varphi_i(t)dt + R_i, \text{ де}$$

$$\varphi_i(t) = a_i + b_i(t - t_{i-1}) + c_i(t - t_{i-1})^2 + d_i(t - t_{i-1})^3, t \in [t_{i-1}, t_i].$$

Відомо, що коефіцієнти a_i, b_i, d_i обчислюються за такими формулами:

$$a_i = f_{\{i-1\}}$$

$$b_i = \frac{f_i - f_{i-1}}{\tau_i} - \frac{\tau_i(2c_i + c_{i+1})}{3}$$

$$d_i = \frac{c_{i+1} - c_i}{3\tau_i}$$

А коефіцієнти c_i є рішенням СЛАР (системи лінійних алгебраїчних рівнянь):

$$\tau_{i-1}c_{i-1} + 2(\tau_{i-1} + \tau_i)c_i + \tau_i c_{i+1} = 3\left(\frac{f_i - f_{i-1}}{\tau_i} - \frac{f_{i-1} - f_{i-2}}{\tau_{i-1}}\right),\ 2 \le i \le N$$

Легко бачити, що матриця для вирішення СЛАР є трьохдіагональною матрицею з діагональним переважанням. Тому коефіцієнти c_i можна обчислити за допомогою методу прогонки. Коефіцієнти виходять з умов вільних кінців сплайна. Зазвичай вимагають нульову кривизну на кінцях сплайна і беруть

$$c_1 = c_{\{N+1\}} = 0 = c_{\{N+1\}}$$

У результаті інтеграл запишеться як сума інтегралів від сплайнів:

$$J = \sum_{i=1}^{N} \int_{t_{i-1}}^{t_i} \varphi_i(t)dt + R = \sum_{i=1}^{N}\left(a_i\tau_i + \frac{b_i}{2}\tau_i^2 + \frac{c_i}{3}\tau_i^3 + \frac{d_i}{4}\tau_i^4\right) + R.$$

Остання формула спрощується при підстановці в неї виразів для коефіцієнтів a_i, b_i, d_i:

$$J = \sum_{i=1}^{N} \frac{f_i + f_{i-1}}{2}\tau_i - \sum_{i=1}^{N} \frac{\tau_i^3(c_{i+1} + c_i)}{12} + R.$$

Випадок з рівномірним розбиттям

Нехай на відрізку $[a,b]$ задана рівномірна сітка розбиття, тобто

$$\tau_1 = \tau_2 = \ldots = \tau_N = \tau.$$

$$J = \frac{\tau}{2}(f_0 + f_N) + \tau\sum_{i=1}^{N-1} f_i - \frac{\tau^3}{6}\sum_{i=2}^{N} c_i + R.$$

Підсумуємо рівняння СЛАР від i = 2 до *N*. Отримаємо:

$$6\sum_{i=2}^{N} c_i = (c_2 + c_N) + \frac{3}{\tau^2}(f_0 - f_1 + f_N - f_{N-1}).$$

$$J = \tau\sum_{i=2}^{N-2} f_i + \frac{\tau}{12}(5f_0 + 13f_1 + 13f_{N-1} + 5f_N) - \frac{\tau^3}{36}(c_2 + c_N) + R$$

Незважаючи на те, що c_2 і c_N все одно доведеться обчислювати методом прогонки, точність і швидкість обчислення наближеного значення інтеграла будуть збільшені за рахунок скорочення числа доданків[8].

РОЗДІЛ ІІ. ПРАКТИЧНА ЧАСТИНА

2.1. Технічний огляд дослідження: опис використаних програмних засобів

2.1.1. Python

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою і типізацією[25]. Розроблена в 1990 році Гвідо ван Россумом[19]. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду[44]. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах і знаходяться у вільному доступі, спільно з документацією та деякими додатковими інструментами і модулями.[21] В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.[47]

Серед основних її переваг можна назвати такі:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносимість програм з однієї платформи на іншу
- стандартний дистрибутив має велику кількість корисних модулів;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини)[4]

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C чи C++ (або на іншій мові, яку можна викликати із C). Python портований майже на всі відомі платформи — від КПК до мейнфреймів. Існують порти під Microsoft Windows, всі варіанти UNIX (включаючи FreeBSD та GNU/Linux), Plan 9, Mac OS та Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 та навіть OS/390, Symbian та Android.

При цьому, на відміну від багатьох портованих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій. Більше того, існує спеціальна версія Python для віртуальної машини Java — Jython, що дозволяє інтерпретатору виконуватися на будь-якій системі, яка підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть бути написаними на ньому[18]. Також кілька проектів забезпечують інтеграцію з платформою Microsoft.NET, основні з яких — IronPython та Python.Net.[42]

Python підтримує динамічну типізацію, тобто, тип змінної визначається лише під час виконання. З базових типів слід зазначити підтримку цілих чисел довільної довжини і комплексних чисел. Python має багату бібліотеку для роботи з текстом, зокрема, кодованим в юнікоді.[46]

З колекцій Python підтримує кортежі (tuples), списки (масиви), словники (асоціативні масиви).[48]

Система класів підтримує множинне успадкування і метапрограмування. Будь-який тип, включаючи базові, входить до системи класів, і за необхідності можливе успадкування навіть від базових типів[9].

Об'єктно-орієнтоване програмування

Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація ООП в Python є елегантною, потужною та добре продуманою, але разом з тим, достатньо специфічною в порівнянні з іншими об'єктно-орієнтованими мовами[29,30]. Опис специфічних особливостей Python

- Класи є одночасно об'єктами з усіма нижче наведеними можливостями.[14]
- Успадкування, в тому числі множинне.
- Поліморфізм (всі функції віртуальні).

- Інкапсуляція (два рівні — загальнодоступні та приховані методи і поля). Особливість — приховані члени доступні для використання та помічені як приховані лише особливими іменами.
- Спеціальні методи, що керують життєвим циклом об'єкта: конструктори, деструктори, розподільники пам'яті.
- Перевантаження операторів (усіх, крім is, '.', '=' і символьних логічних).
- Властивості (імітація поля за допомогою функцій).
- Управління доступу до полів (емуляція полів і методів, частковий доступ тощо).
- Метапрограмування (управління створенням класів, тригери на створення класів, та ін)
- Повна інтроспекція.
- Класові та статичні методи, класові поля.
- Класи, вкладені у функції та інші класи.[36-40]

Функціональне програмування

Python підтримує парадигму функціонального програмування, зокрема:

- функція є об'єктом;
- функції вищих порядків;
- рекурсія;
- розвинена обробка списків (спискові вирази, операції над послідовностями, ітератори, генератори);
- аналог замикань (closures);
- часткове застосування функції;
- анонімні функції
- можливість реалізації інших засобів на самій мові (наприклад, каррінг).

Програмне забезпечення на Python як правило, оформлюється у вигляді модулів, які у свою чергу можуть бути зібрані в пакети. Модулі можуть розташовуватися як у каталогах, так і в ZIP-архівах. Модулі можуть бути двох типів за своїм походженням: модулі, написані на «чистому» Python, і модулі розширення (extension modules), написані на інших мовах програмування. [2]

Інтроспекція

Python підтримує повну інтроспекцію часу виконання. Це означає, що для будь-якого об'єкта можна отримати всю інформацію про його внутрішню структуру.[20]

Застосування інтроспекції (метапрограмування) є важливою частиною того, що називають «pythonic style», і широко застосовується в бібліотеках і фреймворках Python, таких як PyRO, PLY, Cherry, Django та інших, заощаджуючи час програміста, що ними користується.

Бібліотеки

Стандартна бібліотека

Багата стандартна бібліотека є однією з привабливих сторін Python. Тут є засоби для роботи з багатьма мережевими протоколами та форматами Інтернету, наприклад, модулі для написання HTTP-серверів та клієнтів, для розбору та створення поштових повідомлень, для роботи з XML[3], тощо. Набір модулів для роботи з операційної системою дозволяє писати крос-платформове програмне забезпечення. Існують модулі для роботи з регулярними виразами, текстовими кодуваннями, мультимедійними форматами, криптографічними протоколами, архівами, серіалізацією даних, юніт-тестуванням та ін.[5]

Модулі розширення та програмні інтерфейси

Крім стандартної бібліотеки існує багато інших, що надають інтерфейс до всіх системних викликів на різних платформах; зокрема, на платформі Win32 підтримуються всі виклики Win32 API, а також COM в обсязі не меншому, ніж у Visual Basic або Delphi. Існує велика кількість прикладних бібліотек для Python у різноманітних галузях: веб-розробка, бази даних, обробка зображень, обробка тексту, програми операційної системи тощо.

Для Python прийнята специфікація програмного інтерфейсу до баз даних DB-API 2 та розроблено відповідні цій специфікації пакети для доступу до різних СУБД: PostgreSQL, Oracle, Sybase, Firebird (Interbase), Informix, Microsoft SQL Server, MySQL та sqlite.[13] На платформі Microsoft Windows доступ до БД можливий через ADO (ADOdb). Комерційний пакет mxODBC для доступу до СУБД через ODBC для платформ Windows і UNIX розроблений eGenix. Для Python написано багато ORM: (SQLObject, SQLAlchemy, Dejavu, Django), розроблено програмні каркаси для розробки веб-застосунків (Django, Pylons).

Бібліотека NumPy для роботи з багатовимірними масивами дозволяє досягти продуктивності наукових розрахунків, порівнянної зі спеціалізованими пакетами. SciPy використовує NumPy і надає доступ до великого спектру математичних алгоритмів (матрична алгебра — BLAS, level 1-3 і LAPACK; БПФ).

Графічні бібліотеки

З Python поставляється бібліотека tkinter на основі Tcl/Tk для створення крос-платформових програм з графічним інтерфейсом.[10] Для науково-технічної мети найбільшого поширення набуло використання matplotlib — бібліотеки з інтерфейсом, аналогічним MATLAB Plot Tool. Існують розширення, що дозволяють використовувати всі основні GUI бібліотеки — wxPython, засноване на бібліотеці wxWidgets, PyGTK для GTK+, PyQt та PySide для Qt, Kivy та інші. Деякі з них також надають широкі можливості для роботи з базами даних, графікою та мережами, використовуючи всі можливості

бібліотеки, на якій базуються. Для створення ігор та програм, що вимагають нестандартного інтерфейсу, можна використовувати бібліотеку Pygame. Вона також надає великі засоби роботи з мультимедіа: з її допомогою можна керувати звуком і зображеннями, відтворювати відео. Для роботи з растровою графікою використовується бібліотека Python Imaging Library (PIL).

2.1.2. JetBrains PyCharm

PyCharm — інтегроване середовище розробки для мови програмування Python. Надає засоби для аналізу коду, графічний дебаггер, інструмент для запуску юніт-тестів і підтримує веб-розробку на Django. PyCharm розроблена чеською компанією JetBrains на основі IntelliJ IDEA. PyCharm працює під операційними системами Windows, Mac OS X і Linux.

Можливості

1. Статичний аналіз коду , підсвічування синтаксису і помилок.
2. Навігація серед проектів і сирцевого коду: відображення файлової структури проекту, швидкий перехід між файлами, класами, методами і використаннями методів.
3. Рефакторинг : перейменування, винесення методу, введення змінної, введення константи, підняття і опускання методу тощо.
4. Інструменти для веб-розробки з використанням фреймворку Django
5. Вбудований зневаджувач для Python
6. Вбудовані інструменти для юніт-тестування
7. Розробка з використанням Google App Engine
8. Підтримка систем контролю версій: загальний користувацький інтерфейс для Mercurial, Git, Subversion, Perforce і CVS з підтримкою списків змін та злиття[15]

2.1.3. Git

Git - розподілена система керування версіями. Проект був створений Лінусом Торвальдсом для керування розробкою ядра Linux, перша версія випущена 7 квітня 2005. Прикладами проектів, що використовують Git, є ядро Linux, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt і деякі дистрибутиви Linux. Програма є вільною і випущена під ліцензією GNU GPL версії 2.

Система спроектована як набір програм, спеціально розроблений з врахуванням його використання у скриптах. Система має ряд користувацьких інтерфейсів: наприклад, gitk та git-gui розповсюджуються з самим Git. Віддалений доступ до репозиторіїв Git забезпечується git-демоном, SSH або HTTP сервером. TCP-сервіс git-daemon входить у дистрибутив Git і є разом з SSH найпоширенішим і надійним методом доступу. Метод доступу HTTP, незважаючи на ряд обмежень, дуже популярний в контрольованих мережах, тому що дозволяє використання існуючих конфігурацій мережевих фільтрів.

Ядро Git являє собою набір утиліт командного рядка з параметрами. Всі налаштування зберігаються в текстових файлах конфігурації. Така реалізація робить Git легко портується на будь-яку платформу і дає можливість легко інтегрувати Git в інші системи (зокрема, створювати графічні git-клієнти з будь-яким бажаним інтерфейсом).

Репозиторій Git являє собою каталог файлової системи, в якому знаходяться файли конфігурації репозиторію, файли журналів, що зберігають хід операцій, що виконувались над репозиторієм, індекс, що описує розташування файлів і сховище, що містить власне файли. Структура сховища файлів не відображає реальну структуру файлового дерева, вона орієнтована на підвищення швидкості виконання операцій з репозиторієм. Коли ядро обробляє команду зміни (неважливо, при локальних змінах або при отриманні

патча від іншого вузла), воно створює в сховище нові файли, що відповідають новим станам змінених файлів. Особливістю є, що ніякі операції не змінюють вмісту вже існуючих в сховищі файлів.

За замовчуванням репозиторій зберігається в підкаталозі з назвою «.git» в кореневому каталозі робочої копії дерева файлів, що зберігається в репозиторії. Будь-яке файлове дерево в системі можна перетворити в репозиторій git, віддавши команду створення сховища з кореневого каталогу цього дерева (або вказавши кореневий каталог в параметрах програми). Репозиторій може бути імпортований з іншого вузла, доступного по мережі. При імпорті нового репозиторію автоматично створюється робоча копія, відповідна останньому зафіксованому станом імпортованого репозиторію (тобто не копіюються зміни в робочій копії вихідного вузла, для яких на тому вузлі не було виконано команду commit).

У Windows версії (офіційна Windows-версія називається mSysGit) використовується пакет mSys, який є емулятором POSIX-сумісного командного рядка над Windows (похідний від MinGW, приблизний аналог Cygwin, але не має властивою останньому проблеми з вкрай примітивною і субоптимальною реалізацією fork () без використання відповідної можливості ядра Windows). Під mSys перенесені всі необхідні для git бібліотеки та інструменти, а також сам git. При роботі з віддаленими репозиторіями по протоколу SSL буде використовуватися сховище сертифікатів з mSys, а НЕ з Windows.

Існує чимало графічних оболонок для Git, наприклад, TortoiseGit, GitExtensions. Всі вони засновані на mSysGit, вимагають його встановлення на машину і працюють через виконання його командних рядків.

Мережеві можливості і серверні рішення

Git використовує мережу тільки для операцій обміну з віддаленими репозиторіями.

Можливе використання наступних протоколів:

1. git://, відкритий протокол, що вимагає наявність на сервері запущеного демона (поставляється разом з git). Протокол не має можливостей аутентифікації користувачів.
2. ssh:// Використовує аутентифікацію користувачів за допомогою пар ключів, а також вбудований в UNIX-систему «основний» SSH-сервер (sshd). З боку сервера потрібне створення акаунтів, шеллом у яких буде якась команда git.
3. http(s)://. Використовує всередині себе інструмент curl (для Windows - поставляється разом з git), і його можливості HTTP-аутентифікації, як і його підтримку SSL і сертифікатів.

В останньому випадку необхідна наявність на сервері якогось ПО у вигляді веб-додатку, яке буде виконувати роль прошарку між командами git на сервері і HTTP-сервером. Такі прошарки існують як під Linux, так і під Windows (наприклад, WebGitNet, розроблений на ASP.NET MVC 4).

Крім підтримки серверної сторони команд push і pull, такі веб-додатки можуть також давати доступ тільки на читання до сховища через веб-браузер[7].

Переваги

1. Висока продуктивність.
2. Розвинені засоби інтеграції з іншими VCS (Системами Контролю Версій), зокрема, з CVS, SVN і Mercurial. Крім різноспрямованих конвертерів репозиторіїв, наявні в комплекті програмні засоби дозволяють розробникам використовувати git при розміщенні центрального репозиторію в SVN або CVS, крім того, git може імітувати cvs-сервер, забезпечуючи роботу через клієнтські програми і підтримку в середовищах розробки, які спеціально не підтримують git.

3. Продумана система команд, що дозволяє зручно вбудовувати git в скрипти (сценарії).

4. Якісний веб-інтерфейс, доступний прямо з комплекту поставки

Репозиторії git можуть поширюватися і оновлюватися загальносистемними файловими утилітами архівації та оновлення, такими як rsync, завдяки тому, що фіксації змін і синхронізації не змінюють існуючі файли з даними, а тільки додають нові (за винятком деяких службових файлів, які можуть бути автоматично оновлені за допомогою наявних у складі системи утиліт). Для роздачі репозиторію по мережі достатньо будь-якого веб-сервера.

У числі недоліків git зазвичай називають:

1. Відсутність наскрізної нумерації коммітів монотонно безупинно зростаючими цілими числами. У багатьох проектах використовується автоматичні отримання номера цієї версії (наприклад, командою svnversion), побудова .H файлу на основі цього числа, і далі його використання при створенні штампа версії виконуваного файлу

2. Деяка незручність для користувачів, які переходять з інших VCS. Команди git, орієнтовані на набори змін, а не на файли, можуть викликати подив у користувачів, які звикли до файл-орієнтованим VCS, таким як SVN. Наприклад, команда «add», яка в більшості систем управління версіями виробляє додавання файлу до проекту, в git готує до фіксації зроблені у файлах зміни. При цьому зберігається не патч, що описує зміни, а нова версія цільового файлу.

3. Використання для ідентифікації ревізій хешів SHA1, що призводить до необхідності оперувати довгими рядками замість коротких номерів версій, як у багатьох інших системах (хоча в командах допускається використання неповних хеш-рядків).

4. Більші накладні витрати при роботі з проектами, в яких робляться численні незв'язані між собою зміни файлів. При роботі в такому режимі

розміри наборів змін стають досить великі і відбувається швидке зростання обсягу репозиторіїв.

5. Великі витрати часу, у порівнянні з файл-орієнтованими системами, на формування історії конкретного файлу, історії правок конкретного користувача, пошуку змін, що відносяться до заданого місця певного файлу.

6. Відсутність окремої команди перейменування / переміщення файлу, яка відображалася б в історії як відповідне єдине дію. Існуючий скрипт git mv фактично виконує перейменування, копіювання файлу і видалення його на старому місці, що вимагає спеціального аналізу для визначення, що насправді файл був просто перенесений (цей аналіз виконується автоматично командами перегляду історії). Однак, враховуючи той факт, що наявність спеціальної команди для перейменування / переміщення файлів технічно не змушує користувача використати саме її (і, як наслідок, в цьому випадку можливі розриви в історії), поведінка git може вважатися перевагою.

7. Система працює тільки з файлами і їх вмістом, і не відстежує порожні каталоги.[6]

2.1.4. BitBucket

Bitbucket («відро бітів») - веб-сервіс для хостингу проектів та їх спільної розробки, заснований на системі контролю версій Mercurial і Git. За призначенням і пропонованим функціям аналогічний GitHub (проте GitHub не надає безкоштовні «закриті» репозиторії, на відміну від Bitbucket), який підтримує Git і Subversion. Bitbucket підтримує OpenID. Слоган сервісу - «We're here to serve» (Ми тут, щоб служити).

В даний час всім користувачам безкоштовно надаються наступні можливості:

1. Дисковий простір в 2 ГБ на репозиторій.
2. Необмежена кількість публічних репозиторіїв.

3. Необмежена кількість приватних репозиторіїв для команд до п'яти чоловік.

4. Доступ до репозиторіїв за протоколами HTTP і SSH.

5. Можливість прив'язати обліковий запис на сервісі до власного домену.

6. Вікі (окремо для кожного репозиторію, можна відключити).

7. Система обліку помилок (окремо для кожного репозиторію, можна відключити).

8. Інтеграція з Google Analytics, Twitter, Basecamp та іншими службами.

9. RSS-стрічка історії змін.

10. Управління приватностью окремо для кожного репозиторію.

11. Для публічних репозиторіїв кількість користувачів не обмежена (BitBucket безкоштовний для проектів відкритого програмного забезпечення).

2.1.5. Python(x,y)

Python (x, y) - дистрибутив вільного наукового та інженерного програмного забезпечення для чисельних розрахунків, аналізу та візуалізації даних на основі мови програмування Python і великого числа модулів (бібліотек). Також включає прив'язки PyQt для побудови графічних інтерфейсів і науково-орієнтовану інтегровану середу розробки Spyder і велика кількість модулів, у тому числі для обробки масивів (numpy, scipy), машинного навчання (scikit-learn), візуалізації даних (matplotlib). Такий набір дозволяє домогтися функціональних можливостей, схожих з MATLAB, GNU Octave та іншими пакетами для математичного моделювання. З цього набору було використано Spyder, numpy, matplotlib з метою тестування і налагодження програмного коду методів інтегрування. Може бути використано лише на операційних систем сімейства Windows (Vista, 7, 8).[16]

2.1.6. Kivy

Для написання інтерфейсу користувача було обрано фреймворк Kivy. Kivy – набір бібліотек Python з відкритим вихідним кодом для розробки прикладного програмного забезпечення з підтримкою мультитач та природнім інтерфейсом користувача (NUI). Він може працювати на ОС: Android, IOS, Linux, Mac OS X і Windows. Поширюючись під умовами ліцензії MIT, Kivy є вільним і відкритим вихідним кодом. Kivy також підтримує Raspberry Pi, який фінансується за рахунок Bountysource. Фреймворк містить всі елементи для побудови додатків, таких як:

- Розширена підтримка введення для миші, клавіатури, TUIO і OS-специфічні мультитач-події,
- Графічна бібліотека з використанням тільки OpenGL ES 2 на основі буфера вершин об'єктів і шейдерів,
- Широкий вибір Віджетів, які підтримують мультитач,
- Проміжна мова (KV) використовується для легкого створення користувацьких віджетів.[12]

Kivy працює на Linux, Windows, OS X, Android і IOS. Ви можете запустити той самий код на всіх підтримуваних платформах. Він підтримує більшість пристроїв вводу, протоколів і пристроїв, включаючи WM_Touch, WM_Pen, Mac OS X Trackpad і Magic Mouse, Mtdev, Linux Kernel HID, TUIO. Multi-Touch симулятор миші входить у комплект поставки.

Бізнес-використання Kivy цілком є можливим. Фреймворк можна на 100% вільно використовувати, відповідно до ліцензії MIT (починаючи з 1.7.2) і LGPL 3 для попередніх версій. Інструментарій професійно розроблений, активно використовується і підтримується. Можна використовувати його в якості комерційного продукту.Kivy - стабільний фреймворк, який має добре документований API, а також «Керівництво з програмування», щоб допомогти вам розпочати роботу.

Графічний движок побудований на OpenGLES2, використовуючи сучасні швидкі графічний конвеєри. Інструментарій поставляється з бібліотекою більш ніж 20 віджетів, яку легко можна розширити. Більшість фреймворку написано на мові Python, і протестовано регресійним тестуванням.[11]

2.2. Програмна реалізація методів чисельного інтегрування засобами мови Python та фреймворку Kivy

Програмна реалізація складається з 7 модулів:

1. «main.py» - основна частина програми. Містить головний клас та точку входу фреймворку Kivy
2. «NumComparator.kv» - містить в собі опис UI (інтерфейсу)
3. «Binders.py» – модуль, який слугує для зв’язку UI (інтерфейсу) і методів обробки
4. «MathParser.py» - цей модуль містить код для парсингів вхідних даних та являється контейнером для змінних і функцій
5. «Methods.py» - цей модуль містить лише методи чисельного інтегрування та тести на їх коректність
6. «Utils.py» - містить в собі логіку розбору вхідних даних, форматування вхідних і вихідних даних
7. «README.rst» - файл допомоги, який відображається в програмі за допомогою вбудованого рендеру та в депозитарії

2.2.1. Модуль «main»

```
__author__ = 'Igor Olshevsky'
import kivy  # підключення фреймворку Kivy

import Binders  # підключення модулю Binders

kivy.require('1.8.0')  # встановлення мінімальної версії для запуску
from kivy.app import App  # підключення основного модуля фреймворку
from kivy.uix.boxlayout import BoxLayout  # підключення схеми розташування
віджетів


class RootWidget(BoxLayout):  # клас схеми розташування віджетів
    def __init__(self, **kwargs):  # конструктор
        super(RootWidget, self).__init__(**kwargs)  # ініціалізація схеми

    # підключення колбеків
    eval_callback = Binders.eval_callback
    get_var_callback = Binders.get_var_callback
    get_fx_callback = Binders.get_fx_callback
    clear = Binders.clear
    methods_callback = Binders.methods_callback


class NumComparator(App):  # основний клас програми
    def build(self):  # конструктор
        return RootWidget()


if __name__ == '__main__':  # запуск програми
    NumComparator().run()
```

2.2.2. Модуль «NumComparator»

```
<RootWidget>: #кореневий віджет


    #оголошення спливаючих вікон
    #popup crash fix begin
    methods_popup:methods_popup.__self__
    var_popup:var_popup.__self__
    fx_popup:fx_popup.__self__
    info_popup:info_popup.__self__
    #popup crash fix end
    BoxLayout: #головний віджет
          #wrapper
       id:wrap
       orientation:"vertical"
       #pop-ups for custom elements
       Popup: #спливаюче вікно для списку змінних
          id:var_popup
          size_hint:(0.5,0.5)
          title:"Available variables list"
          on_parent:
             if self.parent == wrap: self.parent.remove_widget(self)
          BoxLayout
             orientation:"vertical"
             ScrollView:
                id:var_scroll
                Label
                   id:v_list
                   text:''
                   size_hint:1,None
                   height: var_scroll.height*1.25
                   multiline:True
             Button:
                text:"Close"
                size_hint_y:0.1
                on_release:var_popup.dismiss()
       Popup: #спливаюче вікно для списку функцій
          id:fx_popup
          size_hint:(0.5,0.5)
```

```
    title:"Available functions list"
    on_parent:
        if self.parent == wrap: self.parent.remove_widget(self)
    BoxLayout
        orientation:"vertical"
        ScrollView:
            id:fx_scroll
            Label
                id:fx_list
                text:''
                size_hint:1,None
                height: fx_scroll.height*3
                multiline:True
        Button:
            text:"Close"
            size_hint_y:0.1
            on_release:fx_popup.dismiss()
#methods selectors popup
Popup: #спливаюче вікно для списку методів
    id: methods_popup
    title: "Methods"
    on_parent:
        if self.parent == wrap: self.parent.remove_widget(self)
    BoxLayout:
        orientation:"vertical"
        GridLayout:
            id:methods
            cols: 2
            Label:
                text: "Rectangles method"
            CheckBox:
                id: M1
                active:True
            Label:
                text: "Trapezium method"
            CheckBox:
                id: M2
                active:False
            Label:
                text: "Simpson method"
            CheckBox:
                id: M3
                active:False
```

```
            Label:
               text: "Monte-Carlo method"
            CheckBox:
               id: M4
               active:False
            Label:
               text: "Gauss 10-point method"
            CheckBox:
               id: M5
               active:False
            Label:
               text: "Chebishev method"
            CheckBox:
               id: M6
               active:False
            Label:
               text: "Spline method"
            CheckBox:
               id: M7
               active:False
         Button:
            text: 'Close'
            on_release: methods_popup.dismiss()
            on_press: root.methods_callback()
            size_hint_y:0.1
Popup: #спливаюче вікно з інформацією
   id: info_popup
   title: "Info"
   on_parent:
      if self.parent == wrap: self.parent.remove_widget(self)
   BoxLayout:
      orientation:'vertical'
      RstDocument:
         id:doc
         source:'README.rst'
      Button:
         text: 'Close'
         on_release: info_popup.dismiss()
         size_hint_y:0.1
BoxLayout: #віджет меню
   #menu
   id: topMenu
   height:0
```

```
        size_hint_y:0.05
        Label:
            text: 'Menu'
        Button: #кнопка виклику вікна методів
            id:popupOpener
            text: 'Methods'
            font_size:"16sp"
            on_release: methods_popup.open()
        Button: #кнопка виклику вікна інформації
            id:info
            italic: True
            text: 'Info'
            bold: True
            on_press:info_popup.open()
    BoxLayout: #віджет центральної частини
        padding:(0,10,0,10)
        id:center_wrap
        id: rl
        size_hint_y:0.5
        BoxLayout:
            id:centerLeft
            #main part
            BoxLayout:
                id: center-left-left
                spacing:self.height/10
                orientation:"vertical"
                #buttons
                Button: #кнопка виклику вікна змінних
                    text: 'Variables list'
                    id:cvar
                    on_release:root.get_var_callback()
                    on_press:var_popup.open()
                Button: #кнопка виклику вікна функцій
                    text: 'Functions list'
                    id: cfx
                    on_release:root.get_fx_callback()
                    on_press:fx_popup.open()
                Button: #кнопка очищення полів вводу та виводу
                    text: 'Clear'
                    on_release:root.clear()
        BoxLayout:
            id:center-left-right
            spacing:self.height/10
```

```
            orientation:'vertical'
            #text and eval
            TextInput: #поле вводу даних
                id:input
                multiline:False
                text:'#sin:x=[1,0]'
            Button: #кнопка обчислення
                text:'Evaluate'
                size_hint_y:None
                height:cfx.height
                on_press:root.eval_callback()
        Splitter: #віджет регулювання ширини області вводу і виводу
            size_hint_x:1.5
            padding:(1,0,0,0)
            sizable_from: 'left'
            max_size: self.parent.width-centerLeft.width
            min_size: self.parent.width/2.5
#           size:(self.parent.width+2,1)
            canvas.before:
                Color:
                    rgba: 0.5, 0.7, 1, 0.7
                Rectangle:
                    pos: self.pos
                    size: self.size
            BoxLayout:
                id:center-right
                canvas.after:
                    Color:
                        rgba: 0, 1, 0, 0.1
                    Rectangle:
                        pos: self.pos
                        size: self.size
            #output area
                BoxLayout:
                    orientation:'vertical'
                    ScrollView:
                        id:out_scroll
                        TextInput: #область виводу даних
                            id:comp_value
                            text:'Output\n'
                            readonly:True
                            size_hint:1,None
                            height: out_scroll.height*2
```

```
                multiline:True
BoxLayout: #віджет підвалу з часом обрахунків
   id:footer
   size_hint_y:0.02
   canvas.before:
      Color:
         rgba: 0, 1, 0, 0.1
      Rectangle:
         pos: self.pos
         size: self.size
   Label:
      text:'Calculations time elapsed:'
   Label:
      id:Time
      text:'Not runned yet'
```
[12]

2.2.3. Модуль «Binders»

```
__author__ = 'Igor Olshevsky'

import time #піключення модуля для виміру часу
import MathParser #піключення модуля математичного парсингу
import Methods #піключення модуля методів інтегрування
import Utils #піключення модуля парсингу вводу

m_parser = MathParser.instance #ініціалізація модуля математичного парсингу
meth_list = Methods.wrapper #ініціалізація модуля методів інтегрування
l = [True, False, False, False, False, False, False]
l_names = []
l_names.append("Method of Rectangles")
l_names.append("Trapezium Method")
l_names.append("Simpsons Method")
l_names.append("Monte-Carlo Method")
l_names.append("Gauss 10 points method")
l_names.append("Chebishev method")
l_names.append("Spline method")


def clear(self): #функція очищення областей вводу і виводу
   self.ids["input"].text = ""
   self.ids["comp_value"].text = 'Output\n'
   self.ids["comp_value"].height = self.ids["out_scroll"].height * 2


def get_var_callback(self): #функція виведення списку змінних
   var_names = m_parser.get_variable_names()
   for i in range(len(var_names)):
      self.ids["v_list"].text += var_names[i] + '=' +
str(m_parser.variables[var_names[i]]) +'\n'


def get_fx_callback(self): #функція виведення списку функцій
   fx_names = m_parser.get_functions_names()
   for i in range(len(fx_names)):
      self.ids["fx_list"].text += fx_names[i] + '\n'


def eval_callback(self): #функція обчислення
```

```
    start_time = time.time()
    target = self.ids["input"].text
    output = self.ids["comp_value"]
    output.text += '\n'
    if target != '':
        global l
        if target[0] == '#':
            smth = Utils.clarify(target)
            output.text += Utils.format_out(target)
            for i in range(0, len(l), 1):
                if l[i]:
                    output.text += l_names[i] + ' ' +
"{:.12f}".format((meth_list[i](smth[0], smth[1], smth[2]))) + '\n'
                    output.height = self.ids["out_scroll"].height * 2 + 1
        else:
            output.text += str(Utils.clarify(target))
    end_time = time.time()
    time_res = "{:.3f} sec".format(end_time - start_time)
    self.ids["Time"].text = time_res #виведення витраченого на обрахунки часу


def methods_callback(self): функція виведення вікна методів
    m = []
    for i in range(0, 14, 2):
        m.append(self.ids["methods"].children[i].active)
    global l
    l = m[::-1]
```

2.2.4. Модуль «MathParser»

```
__author__ = 'Igor Olshevsky'

from math import * #підключення стандартних математичних функцій



class PyMathParser(object): #головний клас модуля

   """
   Mathematical Expression Evaluator class.
   You can set the expression member, set the functions, variables and then call
   evaluate() function that will return you the result of the mathematical
expression
   given as a string.
   """

   """
   Mathematical expression to evaluate.
   """
   expression = '' #вираз для обчислень


   """
   Dictionary of functions that can be used in the expression.
   """
   functions = {'__builtins__': None}


   """
   Dictionary of variables that can be used in the expression.
   """
   variables = {'__builtins__': None}

   def __init__(self) #конструктор класу:
      """
      Constructor
      """

   def evaluate(self): #функція обчислень
```

```
        """
        Evaluate the mathematical expression given as a string in the expression
member variable.
        """
        return eval(self.expression, self.variables, self.functions)

    def add_default_functions(self): #додання стандартних функцій до списку

        """
        Add the following Python functions to be used in a mathemtical expression:
        """
        self.functions['acos'] = acos
        self.functions['asin'] = asin
        self.functions['atan'] = atan
        self.functions['atan2'] = atan2
        self.functions['ceil'] = ceil
        self.functions['cos'] = cos
        self.functions['cosh'] = cosh
        self.functions['degrees'] = degrees
        self.functions['exp'] = exp
        self.functions['fabs'] = fabs
        self.functions['floor'] = floor
        self.functions['fmod'] = fmod
        self.functions['frexp'] = frexp
        self.functions['hypot'] = hypot
        self.functions['ldexp'] = ldexp
        self.functions['log'] = log
        self.functions['log10'] = log10
        self.functions['modf'] = modf
        self.functions['pow'] = pow
        self.functions['radians'] = radians
        self.functions['sin'] = sin
        self.functions['sinh'] = sinh
        self.functions['sqrt'] = sqrt
        self.functions['tan'] = tan
        self.functions['tanh'] = tanh

    def add_default_variables(self): # додання стандартних змінних до списку
        """
        Add e and pi to the list of defined variables.
        """
        self.variables['e'] = e
        self.variables['pi'] = pi
```

```
    def add_var(self, text): #функція додання власної змінної до списку
        """
        add custom var to list
            syntax: <^variable name>=<variable value> i.e. "x=0"
        IMPORTANT!  If parent var was changed - child(ren) vars will not change
        """
        name = text.split("=")[0]
        means = eval(text.split("=")[1], self.variables, self.functions)
        self.variables[name] = means

    def add_function(self, text): #функція додання власної функції до списку
        """
        add custom function to list
            syntax: <function name>(<parameter(s)>)=<function body (variables with
operators)>
        """
        step_1 = text.split("=")
        step_2 = step_1[0].split("(")
        fx_name = step_2[0]
        exec("def " + step_1[0] + ": return " + step_1[1]) in locals()
        fx_body = locals()[fx_name]
        self.functions[fx_name] = fx_body
        globals().update(locals())

    def get_variable_names(self): #функція виведення списку змінних
        """
        Return a List of defined variables names in sorted order.
        """
        my_list = list(self.variables.keys())
        try:
            my_list.remove('__builtins__')
        except ValueError:
            pass
        my_list.sort()
        return my_list

    def get_functions_names(self): #функція виведення списку функцій
        """
        Return a List of defined function names in sorted order.
        """
        my_list = list(self.functions.keys())
        try:
```

```
            my_list.remove('__builtins__')
        except ValueError:
            pass
        my_list.sort()
        return my_list

    def init(self): #ініціалізація модуляі
        self.add_default_functions()
        self.add_default_variables()


'''init section'''
instance = PyMathParser()
instance.init()
```

2.2.5. Модуль «Methods»

```
__author__ = 'Igor Olshevsky'

import math
import random #підключення модуля генерації випадкових чисел

import MathParser
import Utils

m_parser = MathParser.instance

# метод прямокутників
def method_of_rectangles(func, min_lim=0, max_lim=1, delta=0.01):
   fx, min_lim, max_lim = Utils.format_input(func, min_lim, max_lim)

   def integrate(range_x):
      integral = 0.0
      step = (max_lim - min_lim) / range_x
      for x in Utils.frange(min_lim, max_lim - step, step):
         integral += step * fx(x + step / 2)
      return integral

   d, n = 1, 1
   while math.fabs(d) > delta:
      d = (integrate(n * 2) - integrate(n)) / 3
      n *= 2

    return integrate(n) + d

# метод трапецій
def trapezium_method(func, min_lim, max_lim, delta=0.01):
   fx, min_lim, max_lim = Utils.format_input(func, min_lim, max_lim)

   def integrate(range_x):
      integral = 0.0
      step = (max_lim - min_lim) / range_x
      for x in Utils.frange(min_lim, max_lim - step, step):
         integral += step * (fx(x) + fx(x + step)) / 2
      return integral

   d, n = 1, 1
```

```
    while math.fabs(d) > delta:
        d = (integrate(n * 2) - integrate(n)) / 3
        n *= 2

    return integrate(n) + d

# метод трапецій (Сімпсона)
def simpson_method(func, min_lim, max_lim, delta=0.01):
    fx, min_lim, max_lim = Utils.format_input(func, min_lim, max_lim)

    def integrate(range_x):
        integral = 0.0
        step = (max_lim - min_lim) / range_x
        for x in Utils.frange(min_lim + step / 2, max_lim - step / 2, step):
            integral += step / 6 * (fx(x - step / 2) + 4 * fx(x) + fx(x + step /
2))
        return integral

    d, n = 1, 1
    while math.fabs(d) > delta:
        d = (integrate(n * 2) - integrate(n)) / 15
        n *= 2

    return integrate(n) + d

#10-точковий метод Гауса
def gauss_10(func, min_lim, max_lim, delta=0.01):
    fx, min_lim, max_lim = Utils.format_input(func, min_lim, max_lim)

    def gauss_calc(a, b):
        g10c1 = 0.9739065285 / 6.2012983932
        g10c2 = 0.8650633667 / 6.2012983932
        g10c3 = 0.6794095683 / 6.2012983932
        g10c4 = 0.4333953941 / 6.2012983932
        g10c5 = 0.1488743390 / 6.2012983932
        g10x1 = 0.0666713443 / 6.2012983932
        g10x2 = 0.1494513492 / 6.2012983932
        g10x3 = 0.2190863625 / 6.2012983932
        g10x4 = 0.2692667193 / 6.2012983932
        g10x5 = 0.2955242247 / 6.2012983932
        m = (b + a) / 2.0
        n = (b - a) / 2.0
        s1 = g10c1 * (fx(m + n * g10x1) + fx(m - n * g10x1))
```

```
        s2 = g10c2 * (fx(m + n * g10x2) + fx(m - n * g10x2))
        s3 = g10c3 * (fx(m + n * g10x3) + fx(m - n * g10x3))
        s4 = g10c4 * (fx(m + n * g10x4) + fx(m - n * g10x4))
        s5 = g10c5 * (fx(m + n * g10x5) + fx(m - n * g10x5))
        s = s1 + s2 + s3 + s4 + s5
        return s * (b - a)

    def gauss(a, b, eps, gc):
        t = (a + b) / 2.0
        ga = gauss_calc(a, t)
        gb = gauss_calc(t, b)
        if math.fabs(ga + gb - gc) > eps:
            eps /= 2
            ga = gauss(a, t, eps, ga)
            gb = gauss(t, b, eps, gb)
        return ga + gb

    return gauss(min_lim, max_lim, delta, gauss_calc(min_lim, max_lim))

# метод Чебишева
def cheb_method(func, min_lim, max_lim):
    fx, min_lim, max_lim = Utils.format_input(func, min_lim, max_lim)
    n = 5
    k = -0.832498
    l = -0.374541
    z = 0.0
    h = (max_lim - min_lim) / n
    ich = 0.0
    x, y, t = [], [], []
    t = [k, l, z, l, k]
    for i in range(0, n - 2):
        x.append(((max_lim + min_lim) / 2 + (max_lim - min_lim) / 2 * t[i]))
    for i in range(n - 1, n + 1):
        x.append(1 - x[n - i])
    for i in range(0, n):
        y.append(fx(x[i]))
    for i in range(0, n):
        ich += y[i] * h
    return ich


# метод Монте Карло
```

```
def mc_method(func, min_lim=0, max_lim=1, n=100):
    fx, min_lim, max_lim = Utils.format_input(func, min_lim, max_lim)
    s = 0
    for i in range(n):
        x = random.uniform(min_lim, max_lim)
        s += fx(x)
    out = (float(max_lim - min_lim) / n) * s
    return out

# метод сплайнів
def spline_method(func, min_lim, max_lim):
    fx, min_lim, max_lim = Utils.format_input(func, min_lim, max_lim)
    total = 1000
    n = total - 1
    f = []
    c = []
    alpha = []
    beta = []
    tau = (max_lim - min_lim) / total
    tmp = 0

    for i in range(0, n + 1):
        c.append(0.0)
        alpha.append(0.0)
        beta.append(0.0)

    for i in range(0, total + 1):
        x = min_lim + tau * i
        f.append(fx(x))

    alpha[0] = (-1 / 4.0)
    beta[0] = (f[2] - 2.0 * f[1] + f[0])

    for i in range(1, n - 1):
        alpha[i] = -1.0 / (alpha[i - 1] + 4)
        beta[i] = (f[i + 2] - 2 * f[i + 1] + f[i] - beta[i - 1]) / (alpha[i - 1] +
4)

    c[n - 1] = (f[total] - 2 * f[total - 1] + f[total - 2] - beta[n - 1]) / (4 +
alpha[n - 1])
    for i in range(n - 2, 0, -1):
        c[i] = alpha[i + 1] * c[i + 1] + beta[i + 1]
    for i in range(0, n):
```

```
        c[i] = c[i] * 3 / (tau * tau)
    x = (5 * f[0] + 13 * f[1] + 13 * f[total - 1] + 5 * f[total]) / 12
    for i in range(2, total - 1):
        tmp += f[i]
    x = (x + tmp) * tau - (c[0] + c[n - 1]) * tau * tau * tau / 36

    return x

# тестова функція-інтегранд
def diff(data, min_lim, max_lim):
    return str(math.fabs(data - (math.e ** max_lim - math.e ** min_lim)))

# тестова функція-інтегранд
def testf(x):
    return math.e ** x

# тестування методів
def test(fx, min_lim, max_lim, delta=0.01):
    t1 = method_of_rectangles(fx, min_lim, max_lim, delta)
    t2 = trapezium_method(fx, min_lim, max_lim, delta)
    t3 = simpson_method(fx, min_lim, max_lim, delta)
    t4 = mc_method(fx, min_lim, max_lim)
    t5 = gauss_10(fx, min_lim, max_lim, delta)
    t6 = cheb_method(fx, min_lim, max_lim)
    t7 = spline_method(fx, min_lim, max_lim)
    out1 = "Method of Rectangles\n" + str(t1) + '\t' + 'Diff=' + diff(t1,
min_lim, max_lim)
    out2 = "Trapezium Method\n" + str(t2) + '\t' + 'Diff=' + diff(t2, min_lim,
max_lim)
    out3 = "Simpsons Method\n" + str(t2) + '\t' + 'Diff=' + diff(t3, min_lim,
max_lim)
    out4 = "Monte-Carlo Method\n" + str(t4) + '\t' + 'Diff=' + diff(t4, min_lim,
max_lim)
    out5 = "Gauss 10 points method\n" + str(t5) + '\t' + 'Diff=' + diff(t5,
min_lim, max_lim)
    out6 = "Chebishev quadrature method\n" + str(t6) + '\t' + 'Diff=' + diff(t6,
min_lim, max_lim)
    out7 = "Spline method\n" + str(t7) + '\t' + 'Diff=' + diff(t7, min_lim,
max_lim)
    print(out1)
    print(out2)
    print(out3)
    print(out4)
```

```
    print(out5)
    print(out6)
    print(out7)

# print('Test1\n')
# test(testf, 0.0, 1.0, 0.01)
# print('Test2\n')
# test(testf, -2.0, 2.05, 0.3)


#ініціалізація модуля
wrapper = [method_of_rectangles, trapezium_method, simpson_method, mc_method,
gauss_10, cheb_method, spline_method]
```

2.2.6. Модуль «Utils»

```
__author__ = 'Igor Olshevsky'
import MathParser

m_parser = MathParser.instance

#функція додання власної функції до списку

def setf(target):
   if target in m_parser.functions:
      return m_parser.functions[target]
   elif not hasattr(target, '__call__'):
      print('No ' + target + ' in functions list. Exp used')
      return m_parser.functions['exp']
   else:
      return target


#функція додання власної змінної до списку

def setv(target):
   try:
      return float(target)
   except ValueError:
      if target in m_parser.variables:
         return m_parser.variables[target]
      else:
         print('No target v in list. 0 used')
         return 0


#функція ітерації з дробовим кроком

def frange(start, end=None, inc=None):
   """A range function, that does accept float increments..."""

   if end is None:
      end = start + 0.0
      start = 0.0

   if inc is None:
      inc = 1.0
```

```
    result = []
    while 1:
        curr = start + len(result) * inc
        if inc > 0 and curr >= end:
            break
        elif inc < 0 and curr <= end:
            break
        result.append(curr)

    return result


#функція розбору введених даних

def eval_parser(data):
    """

    sin(x):x=[-1.0, 1.0]

    or

    sin(x):x=[a, b] if `a` and `b` variables are in list

    """

    p1 = data.split(':')[0]
    p5 = p1
    p1 = p1.split('(')[0][0:]
    p1 = setf(p1)
    p2 = data.split('[')[1].split(',')
    p3 = p2[0].split(']')[0]
    p3 = setv(p3)
    p4 = p2[1].split(',')[0]
    p4 = p4[:-1]
    p4 = setv(p4)
    return p1, p3, p4, p5


#тестові дані

text = []
text.append("^x=1")
```

```
text.append("^y=a")
text.append(":fx(x)=x+1")
text.append(":fy(x):z")
text.append("#fx:x=[0,1]")
text.append("#sin:x=[0,1]")
text.append("2*pi")
text.append("a=b")


#функція визначення типу введених даних

def w2use(data):
   p1 = data[0]
   if p1 == '^':
      return 1
   elif p1 == ':':
      return 2
   elif p1 == '#':
      return 3
   else:
      return 4


#функція обробки введених даних

def clarify(data):
   p1 = w2use(data)
   if p1 == 1:
      try:
         p2 = data[1:]
         m_parser.add_var(p2)
         return "Success in Var"
      except NameError:
         return "Fail in Var"
   elif p1 == 2:
      try:
         p2 = data[1:]
         m_parser.add_function(p2)
         return "Success in Func"
      except (NameError, SyntaxError):
         return "Fail in Func"
   elif p1 == 3:
      try:
```

```
            p2 = data[1:]
            return eval_parser(p2)
         except (NameError, SyntaxError, TypeError, IndexError):
            return "Fail in Integr"
      else:
         try:
            m_parser.expression = data
            return m_parser.evaluate()
         except (NameError, SyntaxError, TypeError, IndexError):
            return "Fail in Eval"


#функція форматування вводу

def format_input(func, min_lim, max_lim):
   fx = setf(func)
   min_lim = setv(min_lim)
   max_lim = setv(max_lim)
   if min_lim == max_lim:
      max_lim += 1
   return fx, min_lim, max_lim


#функція форматування виводу

def format_out(data):
   t_str1 = eval_parser(data)[-1][1:]
   if t_str1 not in m_parser.functions:
      t_str1 = 'e**x'
   t_str2 = str(eval_parser(data)[1])
   t_str3 = str(eval_parser(data)[2])
   t_out = '\n' + t_str1 + ' From ' + t_str2 + ' To ' + t_str3 + '\n'
   return t_out
```

2.2.7. Модуль «README»

```
.. _top:

Welcome to NumIntComparator - Numerical integration methods comparator
======================================================================
**Short overview**

Software written on Python programming language, and based on Kivy as GUI engine

.. note::
   There are embedded images. Be careful!

.. note::
   This document can be dispalyed incorrectly. To avoid this proceed to the
   http://bitbucket.org/oligval/pyintintegr

.. image:: http://i.imgur.com/ROoyt7T.png

Return to top_

List of available methods
=========================

It can integrate through this list of methods:

- Rectangles Method
- Trapezium Method
- Simpson's Method
- Monte Carlo Method
- Gauss 10-point method
- Chebishev Method
- Spline method

By default, selected method is *Rectangles Method*, but you can select **all**
them!

Return to top_

GUI overview
============
```

On the first screenshot You can some buttons:

Menu section

- **Methods** - methods select pop-up, just like as second screen
- **Info** - *this* window
- **Variables** list - pop-up with defined variables
- **Functions list** - pop-up with defined functions
- **Clear** - Clears *input* and *output* areas
- **Evaluate** - calculate everyting, that You print in input area

Return to top_

Input syntax

There are 4 types of input:

- Variable input
- Function input
- Integrand input
- Just evaluate input

Return to top_

Syntax of variable input

* ``^x=10`` - Creates a variable with name ``x`` and contains a value of *10*

* ``^y=x+2`` - If variable ``x`` exists - creates a new variable ``y`` with some changes, based on ``x`` (adds 2 to ``x`` in this case)

To check if variables exists and are correct - just press the 'Variable list' button

.. Note::
   Variables are using Python syntax: ' ** ' instead of ' ^ ' , ' // ' for using integer (floor) division*

.. image:: http://i.imgur.com/1M9M229.png

Return to top_

Syntax of functions input
----

* ``:fx(x)=x+10`` - Creates a function with name ``fx`` that returns ``value+10``

* ``:fy(x)=fx(x)+e**x`` - Functions can be more complex - based on other functions.

To check if functions exists - just press the 'Functions list' button

.. **Note::**
   Functions respectively to the varibles are using Python syntax: ' ** ' instead of ' ^ ' , ' // ' for using integer (floor) division*

.. **image::** http://i.imgur.com/WrPRA1Q.png

Return to **top_**

Syntax of integrand input
----

* ``#sin:x=[1,0]`` - Integrate ``sin`` function from ``1`` to ``0``

* ``#awesomeFunction:x=[a,b]`` - Integrate ``awesomeFunction`` function from ``a`` to ``b`` - if ``awesomeFunction``, ``a`` and ``b`` are defined in theirs lists.

.. **Note::**

   In other case, to maintain stability of software:

   If ``awesomeFunction`` is not defined - it will be replaced with ``e**x``;

   If ``a`` ****or**** ``b`` is not defined - then they will be replaced with ``0``;

   If ``a`` ****and**** ``b`` is not defined - then they will be replaced with ``0`` and ``1``, respectively;

Return to **top_**

Syntax of evaluation input

There are no special prefixes:

* ``1+2`` - without any prefixes will return ``3``

Also, You can use defined functions and variables in evaluation

* ``sin(pi+0.5)`` - will return ``-0.4794255386042029``

Return to top_[17]

ВИСНОВКИ

Здійснене дослідження дало підстави зробити наступні висновки:

1. За результатами аналізу наукової та навчально-методичної літератури встановлено, що впровадження комп'ютерних технологій у навчальний процес є одним з основних напрямів інформатизації освіти, які дозволяють виконувати обмін інформацією за короткий час, у достатньо великому обсязі, для широкого кола користувачів, створюють можливості наступного оброблення, зберігання та розповсюдження одержаної інформації.

2. Встановлено, що до основних вимог до ППЗ відносять: педагогічні вимоги (дидактичні; методичні; обґрунтування вибору тематики навчального курсу; перевірка на педагогічну доцільність використання та ефективність використання); технічні вимоги; ергономічні вимоги; естетичні вимоги; вимоги до оформлення документації.

3. Розглянуто інструментальні засоби створення ППЗ. До них відносять: різні мови програмування, такі як Python, Kv, RST.

4. Проаналізовано програми для створення комп'ютерних засобів чисельного інтегрування. До таких програм відносяться: об'єктно-орієнтована мова програмування Python, пакет програм для організації інтегрування Python(x,y), бібліотека SciPy. Результат аналізу показав, що є засоби, але немає зручного інтерфейсу користувача для них.

5. Спроектовано та розроблено мультимедійний комплекс – електронний ресурс для підтримки вивчення курсу "Чисельні методи", в якому реалізовано чисельні методи інтегрування і обрахунки взагалі.

При реалізації програмного комплексу було використано технології Python, Kivy, PyCharm Git, BitBucket, RST, Python(x,y).

Отже, маючи певний набір універсальних програмних модулів та компонентів, чи таких, що легко змінюються під потреби тієї чи іншої

предметної області, можна швидко та якісно розробляти програмні комплекси в певній галузі освіти.

Слід зазначити, що даний комплекс може бути корисний не тільки для фахівців з інформатики, а й для студентів інших спеціальностей, які вивчають схожі предмети.

Одержані результати дослідження дають підстави вважати, що завдання реалізовані, мета досягнута.

ЛІТЕРАТУРА

1. «Методи обчислень». Практикум на ЕОМ. К. «Вища школа» 1995.
2. Alan Gauld. Learn to Program Using Python: A Tutorial for Hobbyists, Self-Starters, and Those Who Want to Learn the Art of Programming. Addison-Wesley Professional, 2001
3. Christopher A. Jones, Fred L. Drake. Python & XML. O□Reilly & Associates, 2001
4. David Beazley, Guido Van Rossum. Python: Essential Reference. New Riders Publishing, 1999
5. Frederik Lundh. Python Standard Library. O□Reilly & Associates, 2001
6. Git - Book [Electronic Resource] – Mode of access: URL : https://git-scm.com/book/ru/v2
7. Git [Electronic Resource] – Mode of access: URL : https://git-scm.com
8. http://www.machinelearning.ru/wiki/index.php?title=Применение_сплайнов_для_численного_интегрирования
9. Ivan Van Laningham. Teach Yourself Python in 24 Hours. Sams, 2000
10. John E. Grayson. Python and Tkinter Programming. Manning Publications Company, 1999
11. Kivy: Cross-platform Python Framework for NUI Development [Electronic Resource] – Mode of access: URL : http://kivy.org
12. Kv language – Kyvy 1.9.1-dev documentation [Electronic Resource] – Mode of access: URL : http://kivy.org/docs/guide/lang.html
13. Mark Hammond, Andy Robinson. Python Programming on Win32. O□Reilly, 2000
14. Martin C. Brown. Python: The Complete Reference. McGraw-Hill Professional Publishing, 2001
15. Python IDE & Django IDE for Web developers : JetBrains PyCharm [Electronic Resource] – Mode of access: URL : https://www.jetbrains.com/pycharm/

16. pythonxy – Scientific-oriented Python Distribution based on Qt and Spyder [Electronic Resource] – Mode of access: URL : https://code.google.com/p/pythonxy/

17. Quick reStructuredText [Electronic Resource] – Mode of access: URL : http://docutils.sourceforge.net/docs/user/rst/quickref.html

18. Rashi Gupta. Making use of Python. Wiley, 2002. Sweigart Invent Your Own Computer Games with Python. 2008 2010. 436 с. ISBN 978-0-9821060-1-3

19. Welcome to Python.org [Electronic Resource] – Mode of access: URL : https://www.python.org/

20. Wesley J. Chun. Core Python Programming. Prentice Hall PTR, 2000

21. А. Н. Чаплыгин Учимся программировать вместе с Питоном. Учебник ревізія 226. 135 с.

22. Б. П. Демидович, И. А. Марон, Э. З. Шувалова. Численные методы анализа, 3-е изд. — М.: Наука, 1967.

23. Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы. — М.: Бином, 2001 — с. 363—375.

24. Березин И. С., Жидков Н. П. Методы вычислений, т. 2, М., 1959.

25. Бизли, Дэвид М. Язык программирования Python. Справочник. К.: ДиаСофт, 2000. ISBN 966-7393-54-2, ISBN 0-7357-0901-7

26. Вакалюк Т.А. Ляшенко Б. М, Кривонос. О. М. Методи обчислень: навчально-методичний посібник для студентів фізико-математичного факультету — Житомир, Вид-во ЖДУ ім. І. Франка, 2014 – 228с.

27. Вержбицкий В. М. Основы численных методов. — М.: Высшая школа, 2009. — С. 80—84. — 840 с. — ISBN 9785060061239.

28. Демидович и Марон “Основы вычислительной математики“

29. Дэвид М. Бизли. Python. Подробный справочник, 4-е видання. Переклад з англійської. Пітер, 2010. 864 с ISBN 978-5-93286-157-8

30. И. А. Хахаев Практикум по алгоритмизации и программированию на Python. Учебник. М.: Альт Линукс, 2010. 126 с. (Библиотека ALT Linux). ISBN 978-5-905167-02-7

31. Ильина В. А., Силаев П. К. Численные методы для физиков-теоретиков. т. 2. — Москва-Ижевск: Институт компьютерных исследований, 2004. — с. 16-30.

32. К.И. Бабенко. Основы численного анализа. Москва. Наука. 1986.

33. Каханер Д., Моулер К., Нэш С. Численные методы и программное обеспечение (пер. с англ.).. — Изд. второе, стереотип.. — М.: Мир, 2001. — 575 с. — ISBN 5-03-003392-0.

34. Копченова и Марон “Вычислительная математика в примерах и задачах”

35. Крылов В.И. “Приближенные вычисления интегралов“ - М. : Физмат.

36. М. Доусон Программируем на Python. СПб.: Питер, 2012. 432 с. ISBN 978-5-459-00314-7

37. Марк Лутц. Изучаем Python, 4-е видання. Переклад з англійської. СПб.: Символ-Плюс, 2010. ISBN 978-5-93286-159-2

38. Марк Лутц. Программирование на Python, 4-е видання, I том Переклад з англійської. СПб.: Символ-Плюс, 2011. 992 с. ISBN 978-5-93286-210-0

39. Марк Лутц. Программирование на Python, 4-е видання, II том Переклад з англійської. СПб.: Символ-Плюс, 2011. ISBN 978-5-93286-211-7

40. Марк Саммерфилд. Программирование на Python 3. Подробное руководство. Переклад з англійської. СПб.: Символ-Плюс, 2009. 608 с ISBN 978-5-93286-161-5

41. Муха В. С. Вычислительные методы и компьютерная алгебра: учеб.-метод. пособие. — 2-е изд., испр. и доп. — Минск: БГУИР, 2010.- 148 с.: ил, ISBN 978-985-488-522-3, УДК 519.6 (075.8), ББК 22.19я73, М92

42. Ноа Гифт, Джереми М. Джонс. Python в системном администрировании UNIX и Linux. Переклад з англійської. СПб.: Символ-Плюс, 2009. ISBN 978-5-93286-149-3

43. Ольшевський І.В. Порівняльний аналіз методів чисельного обчислення інтегралів. Матеріали Всеукраїнської науково-практичної конференції „Автоматизація та комп'ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку", 16-20 березня 2015, Черкаси.

44. С. Шапошникова Основы программирования на Python. Учебник. Вводный курс. версія 2. 2011. 44 с.

45. Самарский А. А., Гулин А. В. Численные методы: Учеб. пособие для вузов. — М.: Наука. Гл. ред. физ-мат. лит., 1989. — 432 с. — ISBN 5-02-013996-3.

46. Сузи Р. А. Язык программирования Python: Учебное пособие. М.: ИНТУИТ, БИНОМ. Лаборатория знаний, 2006. ISBN 5-9556-0058-2, ISBN 5-94774-442-2

47. Сузи, Р. А. Python. Наиболее полное руководство (+CD). СПб.: БХВ-Петербург, 2002. ISBN 5-94157-097-X