МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра прикладної математики
(повна назва кафедри)

Магістерська робота

Sinc-апроксимації та функції від матриць

Виконав: студент групи ПМпМ-22с
спеціальності
113 – прикладна математика
(шифр і назва спеціальності)

Козловський Ю.А.
(підпис) (прізвище та ініціали)

Керівник Вавричук В.Г.
(підпис) (прізвище та ініціали)

Рецензент Стягар А.О.
(підпис) (прізвище та ініціали)

Львів – 2022

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
Кафедра прикладної математики
Спеціальність 113 Прикладна математика
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри_____

"____"_____2022 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Козловському Юрію Андрійовичу
(прізвище, ім'я, по батькові)

1.**Тема роботи** Sinc-апроксимації та функції від матриць

керівник роботи Вавричук Василь Григорович, доцент кафедри обчислювальної математики, кандидат фізико-математичних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від "22" вересня 2021 року № 6

2. Строк подання студентом роботи 16 травня 2022 р.

3. Вихідні дані до роботи
1. Stenger F. Handbook of sinc numerical methods. Англ. CRCPress, 2011, с.92-100.
2. Davies I., Highman N. J. A Schur–Parlett algorithm for computing matrix functions, с.464-485.
3. Francis J. The QR transformation – Parts 1 and 2, Comput. J., 4 (1961-1962), с. 265–271, с.332–345.

4. Зміст магістерської роботи (перелік питань, які потрібно розробити)
1. Розв’язати наближено інтеграли зі змінною межею за допомогою програмної реалізації, використовуючи sinc-апроксимації, перевірити ефективність даного методу;
2. Розв’язати наближено згортки за допомогою програмної реалізації, використовуючи sinc-апроксимації, перевірити ефективність даного методу та оцінити характер похибки;
3. Реалізувати QR-алгоритм та QR-алгоритм зі зсувами для виконання розкладу Шура, оцінити виконанння і результати алгоритмів. Порівняти базовий QR-алгоритм і QR-алгоритм з зсувами з власним розкладом матриці, оцінити використання кращого підходу для декомпозиції матриці у sinc-методах;
4. Реалізувати алгоритм Шура-Парлетта і на основі розкладу Шура перевірити виконання даного алгоритму для знаходження функції від матриці, застосувати для sinc-згорток і перевірити коректність отриманого результату;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Таблиці чисельних експериментів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 02.09.2021р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Проведення теоретичних досліджень	Вересень-жовтень 2021р.	
2.	Практична реалізація sinc-методу для інтегралів зі змінною межею та sinc-методу для згорток, використовуючи для цього власний розклад матриці	Жовтень-листопад 2021р.	
3.	Практична реалізація базового QR-алгоритму	Листопад – грудень 2021р.	
4.	Аппробація програм та аналіз результату	Грудень 2021р.	
5.	Проведення теоретичних досліджень	Січень 2022р.	
6.	Практична реалізація QR-алгоритму зі зсувами	Січень 2022р.	
7.	Практична реалізація алгоритму Шура-Парлетта	Лютий – Березень 2022р	
8.	Практична реалізація sinc-методу для згорток, використовуючи алгоритм Шура-Парлетта	Березень-Квітень 2022р.	
9.	Аппробація програм та аналіз результату	Квітень 2022р.	
10.	Оформлення магістерської роботи	Квітень-Травень 2022р.	

Студент _____ **Козловський Ю.А.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Вавричук В.Г.**
(підпис) (прізвище та ініціали)

Зміст

Вступ

Актуальність проблеми. У багатьох фізичних або математичних задачах можна зіткнутися з необхідністю розв'язати інтегральне рівняння для знаходження розв'язків. Проте у деяких випадках це неможливо зробити аналітично. Наприклад, підінтегральна функція відома лише у деяких точках або достатньо складна. Тоді виникає необхідність для наближеного обчислення інтегрального рівняння, використовуючи певні чисельні методи.

Найчастіше для знаходження наближеного розв'язку застосовують квадратурні формули. Їх вибір залежить від типу особливостей або інших властивостей підінтегральних функцій, вимог до точності розв'язку, тощо. Існує багато прикладів квадратурних формул, таких як метод трапецій, метод Сімпсона, метод Нистрьома.

Проте варто зауважити, що досить проблематично підібрати правильну квадратуру у випадку обчислення розв'язку інтегральних рівнянь зі змінною межею, яка підпадала б під необхідні вимоги. Особливо вартий уваги випадок, коли підінтегральний вираз містить ядро, тобто функцію з двома аргументами. Згортка є яскравим прикладом інтеграла з ядром, який активно застосовується у операціях, пов'язаних з обробкою зображень та статистиці.

Одним з запропованих методів є використання sinc-апроксимацій для інтегралів з фіксованим та змінним інтервалом інтегрування. До їх переваг можна зарахувати: наближене розв'зування інтегралів з необмеженим інтервалом інтегрування, допустимість сингулярностей на інтервалі та експоненційний порядок спадання похибки. Sinc-методи можна також використовувати для знаходження наближеного розв'язку згорток з фіксованим та змінним інтервалом.

Варто зауважити, що перед використанням sinc-методів до інтегралів потрібно врахувати декілька нюансів. По-перше, у процесі застосування sinc-апроксимацій необхідно використовувати декомпозицію матриці. Від вибору коректного алгоритму залежить відповідно точність розкладу, час виконання та подальший результат. По-друге, після виконання декомпозиції матриці необхідно порахувати функцію від матриці.

Об'єкт дослідження – інтеграли зі змінною межею та згортки.

Предметом дослідження визначаємо використання QR-алгоритму для виконання декомпозиції матриці і алгоритму Парлетта для обчислення функції від матриці.

Мета дослідження полягає у наступному: порівняти результати застосування sinc-апроксимацій до інтегральних перетворень, залежно від використання різних підходів для декомпозиції матриць та обчислення функцій від матриць.

Завдання цієї роботи:

1) Розв'язати наближено інтеграли зі змінною межею за допомогою програмної реалізації, використовуючи sinc-апроксимації, перевірити ефективність даного методу;

2) Розв'язати наближено згортки за допомогою програмної реалізації, використовуючи sinc-апроксимації, перевірити ефективність даного методу та оцінити характер похибки;

3) Реалізувати QR-алгоритм та QR-алгоритм зі зсувами для виконання розкладу Шура, оцінити виконанння і результати алгоритмів. Порівняти базовий QR-алгоритм і QR-алгоритм з зсувами з власним розкладом матриці, оцінити використання кращого підходу для декомпозиції матриці у sinc-методах;

4) Реалізувати алгоритм Шура-Парлетта і на основі розкладу Шура перевірити виконання даного алгоритму для знаходження функції від матриці, застосувати для sinc-згорток і перевірити коректність отриманого результату;

Основною новизною даної курсової роботи є використання QR-алгоритму або його модифікованих версій для здійснення декомпозиції матриці. Ідея вибору даного алгоритму полягає у тому, що він вважається одним з найшвидших та найефективніших алгоритмів для пошуку власних значень і власних векторів. Наприклад, роботи Бауманна і Стенгера по дослідженню sinc-методів полягали у використанні власного розкладу матриці для декомпозиції матриці. Крім того, обчислення функції від матриці не здійснювалися за допомогою алгоритму Парлетта чи інших подібних алгоритмів. Доцільно перевірити, чи можливо застосувати даний підхід у sinc-методах.

Також до новизни можна віднести порівняння результатів, залежно від того, який підхід вибрано, і демонстрацію порядку спадання похибки при різних значеннях параметра дискретизації.

1. Sinc-методи

1.1. Sinc-функція

Означення 1.1 (Sinc-функція). *Sinc-функція позначаютъ як sinc(x) і її можна подати наступним чином*

$$\operatorname{sinc}(x) = \begin{cases} \dfrac{\sin(\pi x)}{\pi x}, & x \neq 0, \\ 1, & x = 0. \end{cases}$$

Нехай маємо деякий параметр дискретизації $h > 0$. Тоді можна визначити множину функцій

$$S(k,h)(x) = \operatorname{sinc}\left(\frac{x}{h} - k\right) = \frac{\sin[(\pi/h)(x-kh)]}{(\pi/h)(x-kh)}, \quad k \in \mathbb{Z}. \tag{1}$$

Сформулюємо наступне означення.

Означення 1.2. *Функцію f можна подати як*

$$f(x) = \sum_{k\in\mathbb{Z}} f(kh)S(k,h)(x), \quad x \in \mathbb{R}, \tag{2}$$

якщо $f \in L^2(\mathbb{R})$ з перетворенням Фур'є, яке є фінітно визначеним на інтервалі $(-\pi/h; \pi/h)$.

Функцію (2) можна використовувати для апроксимації розв'язку функції f, інтегралів від неї або її похідних [12,13]. Це можливе завдяки тому, що хоч рівність (2) і виконується наближено, проте з хорошою точністю і спаданням похибки у випадку зростання кількості вузлів. Для перевірки даного твердження введемо нове позначення і сформулюємо теорему.

Введемо область $D_d = \{z \in \mathbb{C} : |\Im(z)| < d\}$, де $d > 0$ і $p \geq 1$. Крім того, введемо множину функцій f, які аналітичні у D_d. Тепер позначимо через $B_p(D_d)$ множину f таким чином, що $\int\limits_{-d}^{d} |f(x+iy)|dy < \infty$, $x \to \pm\infty$ і $\lim\limits_{y\to d^-} \left\{ \left(\int\limits_R |f(x+iy)|^p dx \right)^{1/p} + \left(\int\limits_R |f(x-iy)|^p dx \right)^{1/p} \right\} < \infty$ [1]. На основі даних нововведень сформулюємо наступну теорему.

Теорема 1.1. *Нехай $f \in B_p(D_d)$ і задовольняє умову*

$$|f(x)| \leq Ce^{-\alpha|x|},\ x \in \mathbb{R},\ C > 0,\ \alpha > 0. \tag{3}$$

Тоді справджуються оцінки

$$\Big\| f - \sum_{k=-N}^{N} f(kh)S(k,h)) \Big\|_{\infty} \leq C_1 \varepsilon N, \tag{4}$$

$$\Big| \int_{-\infty}^{\infty} f(x)dx - h \sum_{k=-N}^{N} f(kh) \Big| \leq C_2 N^{-1/2} \varepsilon_N, \tag{5}$$

де $h = C_0/N^{1/2}$, $\varepsilon_N = N^{1/2}e^{-C_4 N^{1/2}}$, C_k, $\tilde{C}_k$ - додатні константи.

У [12] наведено доведення даної теореми. Отже, можна зробити висновок з теореми, що наближені розв'язки збігаються до f і інтегралу $\int_{-\infty}^{\infty} f(x)dx$ відповідно, і оцінка збіжності є хорошою. Крім того, при збільшенні N похибка буде експоненційно спадати.

Проте даний підхід є обмеженим і вимагає модифікації. Зокрема, проблема полягає в тому, що розв'язати інтеграл можна лише на інтервалі $(-\infty; \infty)$, і до того ж можуть бути обмеження для спадання похибки. Одним з способів сформулювати наближений метод для розв'язування таких інтегралів є введення конформних відображень для області інтегрування.

1.2. Конформні відображення

Конформні відображення застосовуються для побудови апроксимації на інтервалах, відмінних від $(-\infty; \infty)$, послаблення обмежень на швидкість спадання функції та наближень, для яких можна застосовувати sinc-квадратури.

Сформулюємо наступні позначення. Нехай D – однозв'язна область у комплексній площині, а ∂D - її границя. Позначимо $\varphi : D \to D_d$ – конформне відображення. У випадку (a, b) - область інтегрування, то для $a, b \in \partial D$ $\varphi(a) = -\infty$ і $\varphi(b) = +\infty$. Введемо $\psi = \varphi^{-1}$ – обернене відображення таке, що $(a, b) = \psi(\mathbb{R})$. Крім того, позначимо $z = \psi(x)$ для $x \in \partial D$.

Відповідно до [13], функція f, що є аналітична в D_d і для якої виконується умова $|f(z)| \leq \dfrac{c|e^{\alpha z}|}{(1 + |e^z|)^{2\alpha}}$ $(c > 0, \alpha > 0)$ при $z \in D_d$, належить до класу $B_p(D_d)$. Отже, у випадку, якщо F - підінтегральна функція, від якої потрібно обчислити наближено інтеграл на інтервалі (a, b), то F - аналітична в D, якщо $f = (F \circ \psi)$ є аналітичною в D_d. Це дозволяє розглядати f у подальших формулюваннях для sinc-методів.[1]

Перед тим, як перейти до застосування конформних відображень для sinc-апроксимацій, розглянемо деякі їх приклади. Варто зауважити, що нижче перечислені випадки будуть використовуватися у подальших чисельних експериментах.

1) $(a,b) = \mathbb{R}$ і $D_d = \{z \in \mathbb{C} : |\arg[z + (1+z^2)^{1/2}]| < d\}$.
Тоді $\varphi(z) = \ln\left(z + \sqrt{1+z^2}\right)$ і $\psi(w) = \sinh w$ і відповідно до (3) виконується умова
$$|F(x)| \leq \frac{c}{1+|x|^\beta},\ x \in \mathbb{R}.$$

2) $(a,b) = [0,\infty)$ і $D_d = \{z \in \mathbb{C} : |\arg z| < d\}$.
Тоді $\varphi(z) = \ln(z)$, $\psi(w) = e^w$ і відповідно до (3) виконується умова
$$|F(x)| = \begin{cases} cx^\beta, & 0 \leq x \leq 1, \\ cx^{-\beta}, & 1 \leq x < \infty. \end{cases}$$

3) (a,b) - скінченний інтервал і $D_d = \{z \in \mathbb{C} : |\arg\left(\frac{z-a}{b-z}\right)| < d\}$.
Тоді $\varphi(z) = \ln\left(\frac{z-a}{b-z}\right)$ і $\psi(w) = \ln\left(\frac{a+be^w}{1+e^w}\right)$ і відповідно до (3) виконується умова
$$|F(x)| \leq (x-a)^\beta (b-x)^\beta,\ x \in (a,b).$$

Отже, розглянувши приклади конформних відображень на різних інтервалах, можна перейти до їх застосування для інтегралів, які розв'язуватимемо наближено за допомогою sinc-методів.

1.3. Sinc-апроксимація інтегралів зі змінною межею

Перед тим, як розглянути sinc-апроксимацію інтегралів зі змінною межею, повернемося до конформних відображень. Для кількості вузлів $N > 0$ та параметра дискретизації $h > 0$ задамо вузли $z_j = \psi(jh)$ та базис-функції $w_j = S(j,h) \circ \varphi(z_j)$ для $j = -N, \ldots, N$. Далі введемо оператор табуляції $V(f) := (f(z_{-N}), ..., f(z_N))^T$ та вектор-рядок $\mathbf{w} = (w_{-N}, ..., w_N)$. За допомогою w_j можна інтерполювати функції на заданому інтервалі в вузлах сітки z_j.

Для $f \in B_p(D_d)$, $d > 0$ і z_j визначимо ваги $W_j = h/\varphi'(z_j)$. Позначимо множину вагів $\mathbf{W} = (W_{-N}, ..., W_N)$. У подальшому використовуватимемо вузли, ваги, базис-функції для формування наближеного sinc-методу для розв'язання інтегралів зі змінною межею та згорток [15]. Отже, на основі теореми про sinc-апроксимацію з [4], сформулюємо модифіковану теорему для інтегралів зі змінною межею для оцінки апроксимації.

Означення 1.3. *Інтеграли зі змінною межею, залежні від параметра x, з підінтегральною функцією $f \in \mathbb{R}$ позначають так*

$$(\mathcal{J}^+ f)(x) = \int_a^x f(t)dt, \quad (\mathcal{J}^- f)(x) = \int_x^b f(t)dt, \quad x \in (a,b). \tag{6}$$

Теорема 1.2. *Якщо $f \in B_p(D_d)$, тоді для параметрів $N > 0$, $h > 0$,*

$$I^{(-1)} = \Big[\int_0^{i-j} \operatorname{sinc}(x)dx + \frac{1}{2} \Big]_{i,j=-N}^{N} \tag{7}$$

інтеграл зі змінною межею можна подати у вигляді

$$\mathcal{J}_m^{\pm} f = wA^{\pm}V(f). \tag{8}$$

Для рівняння справджується оцінка

$$\|\mathcal{J}^{\pm} f - \mathcal{J}_m^{+} f\|_\infty = O(\xi_N), \tag{9}$$

де матриці ваг задаються таким чином

$$A^+ = I^{(-1)} \operatorname{diag}(\mathbf{W}), \tag{10}$$

$$A^- = (I^{(-1)})^T \operatorname{diag}(\mathbf{W}). \tag{11}$$

Формулювання даного sinc-методу доведено в [4,15], оцінку даній оцінці збіжності надано в [15].

Отже, можна зробити висновок, що до інтегралів зі змінною межею теж можна застосувати sinc-методи, внаслідок чого похибка у разі наближеного обчислення інтегралу буде невеликою і буде зменшуватися у результаті зростання N. Тому тепер можемо перейти до огляду складніших інтегралів зі змінною межею.

1.4. Sinc-згортки

1.4.1. Формулювання методу

Одною з переваг sinc-методів є можливість застосувати їх для інтегралів, у яких підінтегральна функція містить добуток функцій або ядро з певними особливостями. Одним з таких типів інтегралів є згортки. Зокрема їх використовують у перетвореннях Коші та Гільберта, перетворенні Лапласа, диференціальних рівняннях, тощо. Зважаючи на особливості даного типу інтегралу та проблеми у обчисленні аналітично, існує потреба у наближеному обчисленні. Отож, існує можливість розв'язати згортку, використовуючи sinc-апроксимацію.

Перш за все, сформулюємо означення згортки.

Означення 1.4 (Згортка) *Нехай існують відображення функцій $f, g : \mathbb{R} \to \mathbb{R}$ в одновимірному просторі. Згорткою називають інтегральне рівняння з підінтегральною функцією $f * g : \mathbb{R} \to \mathbb{R}$ вигляду*

$$(f * g)(x) = \int_{-\infty}^{\infty} f(x-t)g(t)dt = \int_{-\infty}^{\infty} f(t)g(x-t)dt.$$

Згортки можуть бути визначені та інтегровані на фінітно визначеному інтервалі зі змінною межею, тобто набувати вигляд

$$p(x) = \int_{a}^{x} f(x-t)g(t)dt, \quad q(x) = \int_{x}^{b} f(t-x)g(t)dt, \quad x \in (a,b). \tag{12}$$

Конкретно у цьому розділі буде описано застосування апроксимацій для рівнянь (12). Перш за все, для формулювання sinc-апроксимацій необхідно скористатися визначеннями та теорією з розділу 1.3., зокрема інтегральними операторами $\mathcal{J}^{\pm}$,вагами W та оператором табуляції V. Це можна зробити, оскільки згортки є розширеним випадком інтегралів зі змінною межею, тому (7) і (8) справджуватимуться для sinc-згорток.

Перед тим, як перейти до формування апроксимаційного методу для згорток, застосуємо перетворення Лапласа $\hat{f}$ для функції f в (12) [15]. Дане перетворення можна подати таким чином

$$\hat{f}(s) = \int_{0}^{\infty} e^{-st} f(t)dt, \quad \Re(s) > 0. \tag{13}$$

У нашому випадку набагато зручніше використати $F(s)$, визначене як $F(s) = \hat{f}(1/s)$. Тоді модифіковане перетворення Лапласа виглядає так

$$F(s) = \int_{0}^{\infty} e^{-t/s} f(t)dt, \quad \Re(s) > 0. \tag{14}$$

Перевага використання модифікованого перетворення Лапласа полягає ще й у тому, що функція від матриці $F(A)$ добре зумовлена, що дозволяє отримати хороший наближений розв'язок [4]. Тепер, як у випадку для інтегралів з змінною межею, можна подати теорему для апроксимації (12), використовуючи (14) і дискретний оператор $\mathcal{J}_m^{\pm}$.

Теорема 1.3.[14] / *Якщо інтеграли* $p(x) = \int_a^x f(x-t)g(t)dt, \quad q(x) = \int_x^b f(t-x)g(t)dt$ *існують і рівномірно обмежені на проміжку* (a,b)*, і модифіковане перетворення Лапласа функції* f *існує в півплощині* $\Re(s) > 0$*, тоді*

$$p(x) = F(\mathcal{J}^+)g \approx F(\mathcal{J}_m^+)g, \tag{15}$$

$$q(x) = F(\mathcal{J}^-)g \approx F(\mathcal{J}_m^-)g. \tag{16}$$

Якщо $|F^{'}(s)| < C$ *для* $\Re(s) \geq 0,\ g \in B_p(D_d)$*, тоді*

$$\|p(x) - F(\mathcal{J}^+)g\|_\infty = O(\xi_N), \tag{17}$$

$$\|q(x) - F(\mathcal{J}^-)g\|_\infty = O(\xi_N). \tag{18}$$

Отже, ми отримали наближене подання sinc-згортки і оцінку збіжності. Тепер, використовуючи подання $\mathcal{J}_m^\pm = wA^\pm V$ з теореми 1.2., сформулюємо дискретне подання для (15), (16)

$$(F(\mathcal{J}_m^\pm)g)(x) = w(x)F(A^\pm)V(g). \tag{19}$$

1.4.2. Проблема обчислення функції від матриці у sinc-згортках

Для обчислення (19) перед нами постає проблема у обчисленні функції від матриці. Існує ряд методів для її обчислення [9], деякі з них будуть розглянуті у розділі 3. Загалом при виборі коректного алгоритму необхідно враховувати властивості матриці, вимоги до точності, тощо.

Оскільки наша матриця $A^\pm$ є невиродженою та квадратною, то можна застосувати розклад матриці $A^\pm$, щоб полегшити вибір правильного підходу для обчислення функції У [15] наведено власний розклад матриці $A^\pm = XS^\pm X^{-1}$ як спосіб отримання функції від матриці у (19), де $X^{\pm|}$ - матриця з власними векторами A, S - діагональна матриця, де на головній діагоналі - власні значення A. Тоді відповідно до властивостей матриці [6]

$$F(A^\pm) = XF(S^\pm)X^{-1}. \tag{20}$$

Перевагою такої декомпозиції є поелементне обчислення функції від діагональної матриці S, що легко обчислити. Проте проблема полягає в тому, що на практиці реалізувати даний алгоритм складно. Також у випадку зростання абсолютної похибки при збільшенні кількості вузлів у sinc-методі число обумовленості матриці X

$$\kappa(X) = \|X^\pm\|\|(X^\pm)^{-1}\|$$

зростатиме відповідно [9]. Це робить даний алгоритм нестійким, і різниця між точним і наближеним значенням (15), (16) зростатиме. У такому випадку sinc-метод для згортки виконуватиметься нестабільно і є потреба у алгоритмі, який дозволить мінімізувати похибку під час обчислення функції від матриці.

Для знаходження $F(A^{\pm})$ можна застосувати наступний метод, який називається алгоритмом Шура-Парлетта [10]. Складається він з таких кроків:

1) Здійснити для матриці $A^{\pm} \in \mathbb{R}^{n \times n}$ розклад Шура $A^{\pm} = UT^{\pm}U^T$;

2) Виконати обчислення функції від матриці $F(T^{\pm})$, використовуючи алгоритм Парлетта;

3) Обчислити $F(A) = UF(T^{\pm})Q^T$;

Перевагою даного підходу є те, що отримана матриця U є ортогональна. Це дозволить покращити точність отриманої матриці $F(A)$ і відповідно sinc-згортки[9]. Також це може вплинути на швидкодію виконання даного методу. У наступних розділах буде обгрунтовано, як можна виконати кроки 1,2,3. Детальніше про розклад Шура буде наведено у розділі 2, про реалізацію алгоритма Парлетта і відповідно алгоритма Шура-Парлетта - у розділі 3.

2. Розклад Шура

2.1. Означення розкладу Шура

Теорема 2.1. (Розклад Шура)[7] *Матриця $A \in \mathbb{C}^{n\times n}$ може бути подана у вигляді*

$$A = UTU^*, \tag{21}$$

де U - унітарна матриця розміром $n \times n$ з комплексними числами, T - верхня трикутна матриця розміром $n \times n$ з комплексними числами. Матрицю T називають формою Шура матриці A, а матрицю U - векторами Шура. Оскільки матриця T подібна до матриці A, то вона має такі самі власні значення. До того ж, власні значення матриці A розташовані на головній діагоналі утвореної матриці T.

Варто уточнити, що з рівняння (19) матриця $A^{\pm} \in \mathbb{R}^{n\times n}$. Оскільки матриця з дійсними елементами може містити комплексні власні числа, які складаються з пари спряжених чисел, то може виникнути необхідність у здійснені комплексних обчислень.

Легко показати, що $\alpha + i\beta$ і $\alpha - i\beta$ при $\alpha, \beta \in \mathbb{R}$ є власними значеннями матриці

$$R = \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix}. \tag{22}$$

Знайдемо власні числа для матриці R. Тоді

$$det(|R - \lambda I|) = det \begin{pmatrix} \alpha - I & \beta \\ -\beta & \alpha - I \end{pmatrix} = \lambda^2 - 2\lambda\alpha + \lambda^2 + \beta = 0.$$

Отже, знаходимо розв'язки $\alpha + i\beta$ і $\alpha - i\beta$. Це означає, що для матриці з дійсними елементами виконується (21), проте на головній діагоналі матриці T будуть міститися блоки розмірністю 1×1 у випадку дійсного власного значення або 2×2 у випадку комплексних спряжених власних значень. Таку верхню трикутну матрицю називають верхньою квазітрикутною.

На основі теореми 2.1. і вище перечислених висновків сформулюємо теорему про розклад Шура для матриці $A^{\pm}$ з дійсними елементами.

Теорема 2.2. (Розклад Шура для матриці з дійсними елементами)[7] *Матриця $A \in \mathbb{R}^{n\times n}$ може бути подана у вигляді*

$$A = UTU^T, \tag{23}$$

де U - ортогональна матриця розміром $n \times n$ з дійсними числами, T - верхня квазітрикутна матриця розміром $n \times n$ з дійсними числами, тобто набуває вигляду

$$T = \begin{pmatrix} T_{11} & T_{12} & \cdots & T_{1m} \\ 0 & T_{22} & \cdots & T_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & T_{mm} \end{pmatrix}, \tag{24}$$

де T_{ij} для $i, j = 1, \ldots, m$ є або матрицями розмірністю 1×1, або матрицями розмірністю 2×2, де m - кількість блокових підматриць в T. На головній діагоналі T блокові матриці T_{ii} є або матрицями розмірністю 1×1 з дійсним власним значенням, або матрицями розмірністю 2×2 з комплексними спряженими власними значеннями.

Матрицю T називають формою Шура матриці A, а матрицю U - векторами Шура. Оскільки матриця R подібна до матриці A, то вона має такі самі власні значення. До того ж, власні значення матриці A розташовані на головній діагоналі утвореної матриці T.

Отже, перевагою даного розкладу є те, що вектори Шура є ортогональними і матриця Шура містить власні значення на головні діагоналі. Згодом ці дані нам знадобляться для обчислення функції від матриці. Розклад Шура можливо обчислити за допомогою QR-алгоритму. Тоді перейдемо до опису QR-алгоритму для здійснення розкладу Шура для матриці з дійсними елементами.

2.2. Базовий QR-алгоритм

QR-алгоритм - це чисельний метод в лінійній алгебрі, який використовують для обчислення власних значень і власних векторів матриці. Цей алгоритм є одним з найважливіших алгоритмів лінійної алгебри й використовується в аналітичній геометрії, при розв'язуванні систем інтегральних рівнянь та у математичній фізиці [8]. Ідея використання QR-алгоритму полягатиме у здійснені розкладу Шура для матриці і оцінки роботи алгоритму.

Означення 2.1. (QR-розклад матриці) *Матриця $A \in \mathbb{R}^{n \times n}$ може бути представлена у вигляді*

$$A = QR, \tag{25}$$

де Q - ортогональна матриця розміром $n \times n$ з дійсними числами, R - верхня квазітрикутна матриця розміром $n \times n$ з дійсними числами.

Існує декілька способів отримати QR-розклад матриці, проте детальніше розглянемо перетворення Хаусхолдера.

Означення 2.2. (Перетворення Хаусхолдера)[11] *Якщо гіперплощину можна описати одиничним вектором u, що є ортогональним до неї, та $\langle ., . \rangle$ - скалярний добуток в V, тоді оператор Хаусхолдера має такий вигляд*

$$P_u(x) = x - 2\langle x, u \rangle u.$$

З даної рівності отримуємо матрицю Хаусхолдера

$$P = I - 2uu^T. \tag{26}$$

Іншими словами, перетворення Хаусхолдера — це трансформація, яка приймає вектор і відбиває його від певної площини або гіперплощини. Ми можемо використати цю операцію для обчислення QR-розкладу n×n матриці A.

Нехай x буде довільним дійсним m-вимірним вектором стовпчиком матриці A таким, що $\|x\| = |\alpha|$ для скаляра α. Тоді візьмемо вектор $e_1 = (1, 0, ..., 0)^T$, $\|.\|$ - евклідова норма і I є одиничною матрицею розміром $n \times n$. Встановимо, що

$$u = x - \alpha e_1, \tag{27}$$

$$v = \frac{u}{\|u\|}, \tag{28}$$

$$Q = I - 2vv^T. \tag{29}$$

Отже, згідно з означенням 2.2, матриця Q є $n \times n$ матрицею Хаусхолдера і $Qx = (\alpha, 0, ..., 0)^T$. Вектор Qx дозволяє трансформувати $n \times n$ матрицю A у верхню квазітрикутну форму. Для цього множимо матрицю A на матрицю Хаусхолдера Q_1, яку ми отримали для першого стовпчика. Тоді отримуємо матрицю

$$Q_1 A = \begin{pmatrix} \alpha & \star & \cdots & \star \\ 0 & & & \\ \vdots & & A' & \\ 0 & & & \end{pmatrix}.$$

Далі потрібно повторити цю саму дію з матриці A', у результаті чого отримуємо матрицю Хаусхолдера Q_2'. Розширемо її за допомогою одиничної матриці I. Отже, матриця Q_k має такий вигляд

$$Q_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & Q_k' \end{pmatrix}. \tag{30}$$

Після t ітерацій отримуємо ортогональну матрицю Q та верхню квазітрикутну матрицю R.

$$Q = Q_1^T Q_2^T \cdots Q_t^T, \tag{31}$$

$$R = Q_t \cdots Q_2 Q_1 A. \tag{32}$$

Отже, перемноживши матриці Q і R, отримуємо A.

Тепер можна перейти до формулювання базового QR-алгоритму. Здійснемо одну ітерацію QR-розкладу довільної матриці $A_k \in \mathbb{R}^{n \times n}$, $n \in \mathbb{N}$. Перемножимо отримані

матриці R_{k+1} і Q_{k+1} і отримаємо наступне

$$A_{k+1} = R_{k+1}Q_{k+1}. \tag{33}$$

Відповідно до означення 2.2, матрицю R_{k+1} можна подати як $R_{k+1} = Q_{k+1}^T A_{k+1}$. Отже, з цього можна зробити висновок, що матриця A_{k+1} після однієї ітерації виглядає так

$$A_{k+1} = Q_{k+1}^T A_k Q_{k+1} \tag{34}$$

При збільшенні кількості ітерацій матриця A_k буде збігатися до верхньої квазітрикутної матриці [16], Q_{k+1} - ортогональна матриця, тобто $Q_{k+1}Q_{k+1}^T = I$. Таким чином ми можемо отримати розклад Шура матриці A_{k+1}, що і потрібно нам для sinc-згорток [2]. Отже, QR-розклад дозволяє здійснити розклад Шура.

Перейдемо до формування QR-алгоритму, використовуючи попередні доведення.

Algorithm 1 Базовий QR-алгоритм

1: Дано $A \in \mathbb{R}^{n\times n}$, $A = UTU^T$ - розклад Шура матриці A
2: Вибираємо $A_0 = A$, $U_0 = I$
3: **for** k=1,2,... until termination **do**
4: $\quad Q_k R_k = A_{k-1}$ (Перетворення Хаусхолдера)
5: $\quad A_k = R_k Q_k$
6: $\quad U_k = U_{k-1}Q_k$
7: Повертаємо $T = A_\infty$, $U = U_\infty$

Варто ще додати, у випадку, якщо для власних значень матриці виконується умова $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$, тоді елементи матриці A_k, розташовані нижче головної діагоналі, збігаються до нуля зі складністю $O(|\lambda_i/\lambda_j|)$, $i > j$ для $i, j = 1, \ldots, n$ [3].

2.3. QR-алгоритм з використанням матриці Хессенберга

Однак, яким би простим не був базовий QR-алгоритм у реалізації, у нього є свої недоліки. По-перше, збіжність алгоритму є низькою, особливо у випадку, якщо деякі з власних значень є близькими за значенням. По-друге, алгоритм має складність $O(n^3)$, а деколи може досягати і $O(n^4)$. [3]

Як варіант, покращити час виконання алгоритму можна, застосувавши QR-алгоритм до верхньої матриці Хессенберга. Але для початку сформулюємо означення верхньої матриці Хессенберга.

Означення 2.3.(Верхня матриця Хессенберга). *Квадратна матриця H є верхньою матрицею Хессенберга, якщо містить нульові елементи нижче першої*

діагоналі, тобто

$$h_{ij} = 0, \quad i > j + 1, \quad i, j = 1, n, \tag{35}$$

де n - розмірність матриці H.

Приклад верхньої матриці Хессенберга

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} & \cdots & h_{1n} \\ h_{21} & h_{22} & h_{23} & \cdots & h_{2n} \\ 0 & h_{32} & h_{33} & \cdots & h_{3n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & h_{nn-1} & h_{nn} \end{pmatrix}. \tag{36}$$

Розглянема можливість перетворення матриці A у матрицю Хессенберга. Один з способів є використання перетворення Хаусхолдера P, уже описаної вище у (28). Також використаємо вектори (27) і (28) для матриці P. Можна показати, як це можна зробити на матриці розміром 4×4.

Нехай $A_0 = A$. Помножимо матрицю P_1 на A_0, а тоді добуток цих матриць на P_1. Тоді отримуємо наступне

$$A_1 = P_1 A_0 P_1 = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{pmatrix}. \tag{37}$$

Ми отримали перший стовпець, як у верхньої матриці Хессенберга. Якщо знову повторити таку саму дію з матрицею P_2 і A_1, то отримуємо готову верхню матрицю Хессенберга

$$A_2 = P_2 A_1 P_2 = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{pmatrix} = P_2 P_1 A_0 P_1 P_2 \tag{38}$$

Якщо задати, що $U = P_1 P_2$, і врахувати, що матриця A_0 є квазітрикутною, то ми отримуємо розклад Шура матриці A_2. Тоді матриця A_0 є формою Шура матриці A_2, а добуток матриць $P_2 P_1$ - векторами Шура. З цього слідує

Складність множення P_k зліва складає $\frac{4}{3}n^3$, справа - $2n^3$, формування матриці $U = P_1 \dots P_{n-2} = \frac{4}{3}n^3$ [3]. На основі викладаних ідей сформуємо алгоритм перетворення матриці до матриці Хессенберга, який повертатиме дві матриці: верхню матрицю Хессенберга й вектори Шура.

Тепер можемо перейти до застосування матриці Хессенберга замість початкової матриці у алгоритмі базового QR-алгоритму. Ідея полягає у тому, що як початкове

Algorithm 2 Перетворення до матриці Хессенберга

1: Перетворюємо матрицю $A \in \mathbb{R}^{n\times n}$ у верхню матрицю Хессенберга.
2: **for** k=1,2,..., $n-2$ **do**
3: Ініцілізуємо перетворення Хаусхолдера P_k
4: $S_k = P_k A$
5: $A = S_k A$
6: $U = I_n$
7: **for** k = $n-2$, ..., 1 **do**
8: $U = P_k U$
9: Повертаємо A, U

значення матриці A_0 задаємо матрицю Хессенберга. Також для початкового значення векторів Шура замість одиничної матриці задаємо матрицю U, яку отримаємо з Алгоритму 2. Тоді модифікований алгоритм виглядатиме таким чином.

Algorithm 3 QR-алгоритм з використанням матриці Хессенберга

1: Дано $A \in \mathbb{R}^{n\times n}$, $A = UTU^T$ - розклад Шура матриці A, (H, Z) - результат перетворення матриці A до матриці Хессенберга
2: Вибираємо $A_0 = H$
3: Вибираємо $U_0 = Z$
4: **for** k=1,2,... until termination **do**
5: $Q_k R_k = A_{k-1}$ (Перетворення Хаусхолдера)
6: $A_k = R_k Q_k$
7: $U_k = U_{k-1} Q_k$
8: Повертаємо $T = A_\infty$, $U = U_\infty$

Швидкодія Алгоритму 3 є кращою порівняно з Алгоритмом 1. Складність алгоритму буде не перевищувати $O(n^2)$, що набагато краще за швидкодію Алгоритму 1.

2.4. QR-алгоритм з зсувами

QR-алгоритм з використанням матриці Хессенберга можна вважати дієвим, проте проблема полягає у точності і швидкодії. Алгоритм вимагає від нас задавати кількість кроків і не надає можливості встановити необхідну нам точність. Збіжність відбувається надто повільно, що робить алгоритм не надто практичним у користуванні [8]. Насправді є можливість суттєво покращити швидкість виконання алгоритму за допомогою так званих зсувів.

Як відомо, QR-алгоритм формує верхню квазітрикутну матрицю T з нижньої діагоналі до верхньої. Таким чином, для матриці A розмірністю $n\times n$ алгоритм обрахує

власне значення λ_n. Тоді матриця A_k після k-ітерацій у випадку знаходженння λ_n виглядатиме наступним чином

$$A_k = \begin{bmatrix} A_k^{'} & a_1 \\ 0 & \lambda_n \end{bmatrix}, \tag{39}$$

де $A_k^{'}$ має розмірність $n-1 \times n-1$. У цьому випадку доцільніше виконувати QR-розклад для матриці $A_k - \lambda_n I$. Тоді

$$A_k - \lambda_n I = Q_k R_k. \tag{40}$$

У випадку виконання (33) добуток матриць набуває вигляду

$$R_k Q_k = \begin{pmatrix} \ddots & \vdots & \vdots \\ \dots & \ddots & \vdots \\ \dots & 0 & 0 \end{pmatrix}. \tag{41}$$

Тоді можна зробити висновок, що $\tilde{A_k} = R_k Q_k + \lambda_n I \sim A_k$. У цьому випадку λ_n називають зсувом для цього кроку QR-алгоритму [3]. Зазвичай зсув позначають так σ_k. Ще можна окремо сказати, що даний підхід дозволяє нам задати необхідну точність для пошуку власного числа. Таким чином після обчислення власного числа з необхідною точністю можливо застосувати даний алгоритм для матриці $A_k^{'}$, тобто розмірність матриць буде зменшуватися і виконання QR-кроку буде швидшим.

Algorithm 4 QR-алгоритм з зсувами

1: Дано $A \in \mathbb{C}^{n \times n}$, $A = UTU^*$ - розклад Шура матриці A, (H, Z) - результат перетворення матриці A до матриці Хессенберга
2: Вибираємо $A_0 = H$
3: Вибираємо $U_0 = Z$
4: **for** $m = n, \dots, 2$ **do**
5: **while** $|a_{m,m-1}^{(k)}| < \epsilon$ **do** (Задаємо точність за допомогою ϵ)
6: Вибираємо зсув σ_k
7: $A_{k-1} - \sigma_k I = Q_k R_k$ (Перетворення Хаусхолдера)
8: $A_k = R_k Q_k + \sigma_k I$
9: $U_k = U_{k-1} Q_k$
10: Повертаємо $T = A$, U

Хоч Алгоритм 4 працює швидко, проте і є недоліки його використання. По-перше, він строго залежить від вибору зсуву і для деяких симетричних матриць не підійде $\sigma_k = a_{k,k}$. Тому важливим є проблема підбору коректного зсуву. По-друге, порівняно з Алгоритмом 3, не виконується умова $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$.

На жаль, для матриць (10), (11) не підходить цей підхід. Даний алгоритм працює лише, якщо $A \in \mathbb{C}^n \times n$, тобто розрахований для верхніх трикутних матриць.

Оскільки відповідно до теореми 2.2. верхня квазітрикутна матриця може містити на головній діагоналі комплексні спряжені власні значення у вигляді підматриці розмірністю 2×2, то в Алгоритмі 4 цикл з 5 по 9 рядок буде виконуватися дуже багато часу у випажку задання хорошої точності.

[3] пропонує наступний підхід для вирішення цієї проблеми для дійсних матриць. Для цього перевіримо наступне.

Нехай $A_{mm} \in R^{2\times 2}$ - матриця на головній діагоналі матриці $A \in R^{n\times n}$, де m - кількість блочних підматриць на головній діагоналі. Відповідно виглядає вона так

$$A_{mm} = \begin{pmatrix} a_{n-1n-1} & a_{n-1n} \\ a_{nn-1} & a_{nn} \end{pmatrix}. \tag{42}$$

Відповідно до (22), під час проведення QR-ітерацій матриця A_{mm} набуватиме такого вигляду. Тому доцільно вибрати як зсув власні значення (42), як у прикладі (40). Позначимо їх σ_1, σ_2. Оскільки вони спряжені, то $\sigma_1 = \tilde{\sigma_2}$.

Виконаємо два кроки Алгоритму 4 для $A \in \mathbb{R}^{n\times n}$.

$$A_0 - \sigma_1 I = Q_0 R_0,$$

$$A_1 = R_0 Q_0 + \sigma_1 I,$$
$$A_1 - \sigma_2 I = Q_1 R_1,$$
$$A_2 = R_1 Q_1 + \sigma_2 I.$$

На основі цих рівнянь отримуємо наступне

$$R_1 Q_1 + (\sigma_1 + \sigma_2) I = Q_2 R_2. \tag{43}$$

Домноживши Q_1 зліва, а R_1 справа до кожної матриці, і після ряду перетворень отримуємо фінальний вираз

$$(A_0 - \sigma_1 I)(A_0 - \sigma_2 I) = A_0^2 - 2\Re(\sigma) A_0 + |\sigma|^2 I = Q_1 Q_2 R_1 R_2. \tag{44}$$

З (44) слідує, що даний подвійний QR-розклад дозволяє нам уникнути комплексних чисел, що і необхідно для формування QR-алгоритму для дійсних матриць. У кінцевому варіанті формула

$$A_2 = (Q_1 Q_2)^* A_0 (Q_1 Q_2) \tag{45}$$

аналогічна (34). [8] пропонує наступний дієвий алгоритм на основі даного підходу з подвійними зсувами (43), (44). Він базується на Алгоритмі 4 з застосуванням

спеціальних зсувів, необхідністю порахувати матрицю Хаусхолдера (26) під час QR-ітерації. Також вказано спеціальні умови, які дозволяють встановити точність та для деяких елементів матриці відбувається заокруглення, щоб прискорити процес.

Можна зробити висновок, що QR-алгоритм справді дієвий для здійснення розкладу Шура як для комплексних, так і для дійсних матриць. У даному розділі продемонстровано приклади і алгоритми як базового QR-алгоритму, так і дійсно дієвого QR-алгоритму з зсувами. У прикладах далі буде наведено, як дані QR-алгоритми використовуються на практиці і в подальшому для обчислення функції для матриці і відповідно sinc-згорток.

Algorithm 5 QR-алгоритм з подвійними зсувами (Алгоритм Френсіса)

Дано $A \in \mathbb{R}^{n\times n}$, $A = UTU^T$ - розклад Шура матриці A, (H, Q) - результат перетворення матриці A до матриці Хессенберга
$p = n$
while $p > 2$ **do**
    $q = p - 1$
    % Визначаємо кроки σ_1 і σ_2 %
    $\sigma_1 = H_{q,q} + H_{p,p}$
    $\sigma_2 = H_{q,q}H_{p,p} + H_{q,p}H_{p,q}$
    $x = H_{1,1}^2 + H_{1,2}H2, 1 - \sigma_1 H_{1,1} + \sigma_2$
    $y = H_{2,1}(H_{1,1} + H_{2,2} - s)\sigma_1$
    $z = H_{2,1}H_{3,2}$
    for $k = 0, \ldots, p - 3$ **do**
        % Застосувати перетворення Хаусхолдера до $[x, y, z]^T$%
        % Повернути матрицю Хаусхолдера P %
        $r = max\{1, k\}$
        $H_{k+1:k+3,r:n} = P^T H_{k+1:k+3,r:n}$
        $Q_{1:n,k+1:k+3} = Q_{1:n,k+1:k+3}P$
        $x = H_{k+2,k+1}$
        $y = H_{k+3,k+1}$
        if $k < p - 3$ **then**
            $z = H_{k+4,k+1}$
    % Застосувати перетворення Хаусхолдера до $[x, y]^T$%
    % Повернути матрицю Хаусхолдера P %
    $H_{q:p,p-2:n} = P^T H_{q:p,p-2:n}$
    $H_{1:p,p-1:p} = H_{1:p,p-1:p}P$
    $Q_{1:n,p-1:p} = Q_{1:n,p-1:p}P$
    if $|H_{p,q}| < \epsilon(|H_{q,q}| + |H_{p,p}|)|$ **then**
        $H_{p,q} = 0$
        $p = p - 1$
    if $|H_{p,q}| < \epsilon(|H_{q,q}| + |H_{p,p}|)|$ **then**
        $H_{p-1,q-1} = 0$
        $p = p - 2$
Повертаємо $T = H$, $U = Q$

3. Алгоритм Шура-Парлетта

3.1. Означення функції від матриці

Перш за все, необхідно сформулювати означення функції від матриці. Цей термін має багато значень, тому потрібно чітко окреслити, що це означає в даному контексті.

Означення 3.1. (Функція від матриці). *Нехай матриця $A \in \mathbb{C}^{n\times n}$, яка має скінченну кількість власних значень λ_i. Тоді за допомогою скалярної функції $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ у випадку, якщо $f(t)$ визначена на $\lambda(A)$, можемо подати матрицю $f(A) \in \mathbb{C}^{n\times n}$, замінивши z на A.*

Наприклад, розглянемо функцію $f(t) = \dfrac{(t^2+1)}{(1-t)}$. У випадку, якщо ми хочемо обчислити $f(A)$ для довільної матриці $A \in \mathbb{R}^{n\times n}$, то вона набуває такого вигляду

$$f(A) = (I - A)^{-1}(I + A^2).$$

Проте це не означає, що можна в функції f замінити t на будь-яку матрицю. Наступне означення охарактеризує, що $f(t)$ точно є визначена на спектрі A.

Означення 3.2.[6] *Скалярна функція f визначена на спектрі матриці $A \in \mathbb{C}^{n\times n}$, тобто $\lambda_1, \ldots, \lambda_s$, тоді і тільки тоді, коли визначені на спектрі наступні функції*

$$f^{(j)}, \quad 0 \le j \le n_i, \quad 1 \le i \le s. \tag{46}$$

Це можна продемонструвати за допомогою наступної таблиці. У ній показано, як ми визначаємо для кожної функції і її похідних спектр власних значень.

λ	$f^{(0)}$	$f^{(1)}$	$f^{(2)}$	$\cdots$	
λ_1	$f(\lambda_1)$	$f'(\lambda_1)$	$f''(\lambda_1)$	$\cdots$	$f^{(n_1)}(\lambda_1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
λ_s	$f(\lambda_s)$	$f'(\lambda_s)$	$f''(\lambda_s)$	$\cdots$	$f^{(n_s)}(\lambda_s)$

Проте це поки не дає чіткої картини, як саме виглядає функція від матриці. На основі даної таблиці і означення 3.2. можна сформувати означення подання функції

від матриці через канонічну форму Жордана [9]. Так будь-яку матрицю $A \in C^{n\times n}$ можна виразити через канонічну форму Жордана

$$J = Z^{-1}AZ = diag(J_1, \ldots, J_p), \tag{47}$$

$$J_i = J_i(\lambda_i) \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix} \in \mathbb{C}^{n_i \times n_i}. \tag{48}$$

У випадку, якщо f визначена на спектрі матриці A, тоді її можна подати наступним чином

$$f(A) = Zf(J)Z^{-1} = Zdiag(f(J_i))Z^{-1}, \tag{49}$$

де

$$f(J_i) = \begin{pmatrix} f(\lambda_i) & f^{'}(\lambda_i) & \ldots & \frac{f^{(n_k-1)}(\lambda_i))}{(n_k-1)} \\ & f(\lambda_i) & \ddots & \vdots \\ & & \ddots & f^{'}(\lambda_i) \\ & & & f(\lambda_i) \end{pmatrix} \in \mathbb{C}^{n_i \times n_i} \tag{50}$$

Отже, таким чином можна подати функцію від матриці через (49) і (50). Проблема полягає у тому, що не зовсім практично, оскільки, окрім декомпозиції матриці (47), вона вимагає обчислення $n_k - 1$ похідних функції f для кожного власного значення матриці. Для випадку (19) можна застосувати підхід, запропонований в розділі 1.4.2.

3.2. Метод Шура

Завдяки одному з QR-алгоритмів можна здійснити розклад Шура, як це було продемонстровано в розділі 2. Отже, можна записати розклад матриці $A^{\pm} \in \mathbb{R}^{n\times n}$ з рівняння (19) так

$$A^{\pm} = UT^{\pm}U^T, \tag{51}$$

де $U \in \mathbb{R}^{n\times n}$ - ортогональна матриця, $T^{\pm} \in \mathbb{R}^{n\times n}$ - верхня квазітрикутна матриця.

Сформуємо метод Шура, який полягатиме у тому, щоб за допомогою (51) подати функцію від матриці $f(A^{\pm})$ у такій формі, щоб можливо було застосувати до неї алгоритм Парлетта. Перед тим, як перейти до формулювання підходу для обчислення, потрібно розглянути деякі властивості для функцій від матриць з наступної теореми.

Теорема 3.1.

Нехай існує матриця $A \in \mathbb{R}^{n\times n}$ і скалярна функція f визначена для A. Тоді справджуються наступні властивості

1) $f(XAX^{-1}) = Xf(A)X^{-1}$, $det(X) \neq 0$;

2) *Якщо λ_i є власними значеннями A, тоді власні значення $f(A)$ можна подати через $f(\lambda_i)$;*

3) *Якщо X і A - переставні матриці, то X і $f(A)$ - також переставні;*

4) *Якщо A є трикутною матрицею з блоками, тоді $F = f(A)$ є трикутною матрицею з блоками, до того ж такою самою структурою, як і A;*

5) *Якщо A є трикутною матрицею з блоками, тоді справджується $F_{ii} = f(A_{ii})$, де $F_{11}, \ldots, F_{mm}$ - діагональні блоки матриці F;*

6) *Якщо $A = diag(A_{11}, \ldots, A_{mm})$ є діагональною матрицею з блоками, тоді $f(A) = diag(f(A_{11}), \ldots, f(A_{mm}))$;*

Доведення даної теореми детальніше наведено в [6]. Проте для деяких цих властивостей можна швидко перевірити, чи справджуються вони для (51). Оскільки U - ортогональна матриця, то виконується рівність $U^T = U^{-1}$, значить можна використати властивість 1. Крім того, оскільки власні значення $T^{\pm}$ рівні $A^{\pm}$ відповідно до теореми 3.2., то властивість 2 також справджується.

Перейдемо до формулювання. Відповідно до означень 3.1. і 3.2., f повинна бути визначена на спектрі $A^{\pm}$. Як відомо з теореми 3.2., матриця $T^{\pm}$ має такі самі власні значення, що й матриця $A^{\pm}$, тобто вони подібні. Отже, ми можемо перевести проблему з обчислення $f(A^{\pm})$ до проблеми обчислення $f(T^{\pm})$. Тобто

$$f(A^{\pm}) = f(UT^{\pm}U^T) = Uf(T^{\pm})U^T. \tag{52}$$

Праву частину (52) називають методом Шура для $f(A^{\pm})$. Надалі використовуватиметься запис $F = f(T^{\pm})$. Оскільки $T^{\pm}$ - верхня квазітрикутна матриця, то ми можемо записати її у вигляді верхньої трикутної матриці з блоками. Тоді F теж буде трикутною матрицею з блоками, згідно з властивістю 3 теореми 3.1. Нехай матриця $T^{\pm}$ складається з m блокових матриць. Тоді

$$F = f(T^{\pm}) = \begin{pmatrix} F_{11} & F_{12} & \cdots & F_{1n} \\ 0 & F_{22} & \cdots & F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & F_{nn} \end{pmatrix}, \tag{53}$$

де $F \in \mathbb{R}^{n\times n}$, F_{ij} - блокові матриці розмірністю 1×1 або розмірністю 2×2, $F_{ii} = f(T_{ii})$.

Тепер завдання знайти F полягає у 2 кроках:

1) Обчислити $F_{ii} = f(T_{ii})$;

2) Знайти $F_{ij} = f(T_{ij})$ для $i \neq j$;

3.3. Обчислення діагональних блокових матриць за допомогою ряду Тейлора

Для вирішення проблеми 1 можна застосувати алгоритму на основі ряду Тейлора для обчислення блокових матриць на головній діагоналі з (53) [6]. За допомогою ряду Тейлора f можна подати у такому вигляді

$$f(z) = \sum_{k=0}^{\infty} a_k (z - \alpha)^k. \tag{54}$$

Відповідно таким самим чином можемо розкласти $f(T)$ для $T \in \mathbb{C}^{n \times n}$ у ряд Тейлора

$$f(T) = \sum_{k=0}^{\infty} a_k (T - \alpha I)^k, \tag{55}$$

якщо виконується $|\lambda_i - \alpha| < r$ для деякого радіусу збіжносі r для кожного λ_i у T.

Загалом даний алгоритм можна застосовувати навіть для знаходження (52). Проте він занадто громіздкий у реалізації й необхідно порахувати кожне власне значення. У випадку ж обчислення F_{ii} нам відомо про власне значення даної матриці, навіть у випадку спряжених комплексних власних значень. Тому доречно застосувати ряд Тейлора для діагональних блоків (53).

Зазвичай його застосовують не до (55), а до модифікованої версії ряду Тейлора для матриць [9]. Якщо (54) можливо розкласти в ряд Тейлора з необмеженим радіусом збіжності і можливо знайти похідні від даної функції. Тоді якщо подати $T = \sigma I + M$, де $\sigma = trace(T)/n$, n - розмірність квадратної матриці T, то (55) набуває вигляду

$$f(T) = \sum_{k=0}^{\infty} \frac{f^{(k)}(\sigma)}{k!} M^k. \tag{56}$$

У [5] наведено один з методів для обчислення (56) і оцінки збіжності та швидкодію даного алгоритму.

3.4. Алгоритм Парлетта

Тепер проблема полягає у знаходженні блокових матриць F_{ij} з (53). Для вирішення цієї проблеми можна використати можна використати властивіть 3 з теореми 3.1, тобто $FT^{\pm} = T^{\pm}F$. Парлетт запропонував у [17] наступну рекурсію у випадку $T = (T_{ij})$ і $f(T) = (F_{ij}) = (F(T_{ij})_{ij})$ - верхні квазітрикутні матриці. Тоді

$$\sum_{k=i}^{j} T_{ik}F_{kj} = \sum_{k=i}^{j} F_{ik}T_{kj}, \quad i < j \tag{57}$$

або

$$T_{ii}F_{ij} - F_{ij}T_{jj} = F_{ii}T_{ij} - T_{ij}F_{jj} + \sum_{k=i+1}^{j-1} (F_{ik}T_{kj} - T_{ik}F_{kj}), \quad i < j. \tag{58}$$

Отже, рівняння (58) дозволяє обчислити недіагональні блоки F_{ij}. Для обчислення F_{ii} розмірністю 1×1 або 2×2 можна застосувати ряд Тейлора (57) з розділу 3.3. У випадку діагональні блоки мають розмірність 2×2, то їх можна обчислити за допомогою ряду Тейлора для апроксимації функції від матриці або іншого похідного алгоритму.

Algorithm 6 Алгоритм Парлетта

1: $T \in \mathbb{R}^{n \times n}$ - верхня квазітрикутна матриця з блоками розмірністю 1×1 або 2×2.
2: $f(T) = (F_{ij})$ - верхня квазітрикутна матриця з блоками розмірністю 1×1 або 2×2.
3: **for** i = 1,. . . , n **do**
4: % Порахувати F_{ii}, розклавши через ряд Тейлора%
5: $F_{ii} = f(T_{ii})$
6: **for** j=2,3,. . . ,n **do**
7: **for** i=j-1, . . . , 1 **do**
8: $T_{ii}F_{ij} - F_{ij}T_{jj} = F_{ii}T_{ij} - T_{ij}F_{jj} + \sum_{k=i+1}^{j-1}(F_{ik}T_{kj} - T_{ik}F_{kj})$
9: Повертаємо F

Варто також зауважити, що алгоритм не є ідеальний. У випадку, якщо власні значення всередині діагональних блоків розміром 2×2 матриці T_{ii} близькі за значенням, тобто існує $\beta > 0$ для $\lambda_1 \in \lambda(T_{ii})$, $\lambda_2 \in \lambda(T_{ii})$, $\lambda_1 \neq \lambda_2$ таке, що

$$|\lambda_1 - \lambda_2| \leq \beta, \tag{59}$$

то у Алгоритмі 6 отримуємо ділення на нуль у рядку 8. Тоді потрібно обчислювати функцію від матриці $\tilde{T} = (\tilde{T}_{ij}) = U^T T U$, де U - вектори Шура і перевіряти наступні умови

$$min\{|\lambda - \mu| : \lambda \in \lambda(\tilde{T_{ii}}), \lambda \in \tilde{T_{jj}}, i \neq j\} > \beta, \tag{60}$$

$$max\{|\lambda - \mu| : \lambda, \mu \in (\tilde{T})_{ii}, \lambda \neq \mu\} \leq (m-1)\beta, \tag{61}$$

і відповідно знаходити функцію від матриці $f(\tilde{T})$ [5].

3.5. Оцінка алгоритма Шура-Парлетта

Тепер можна сформувати остаточний вигляд алгоритма Шура-Парлетта:

1) Виконати розклад Шура $A = UTU^T$;

2) Задіяти метод Шура для $f(A) = Uf(T)U^T$;

3) За допомогою алгоритма Парлетта знайти $f(T)$;

4) Отримати $f(A)$;

Швидкодія алгоритму дорівнює $\mathcal{O}(n^3)$. Проблема у обчисленні може виникнути у випадку великої кількості комплексних спряжених чисел, що вимагає обрахунку діагональних блокових матриць, використовуючи ряд Тейлора. Особливо це може проявитися у випадку матриць з великою розмірністю. Також проблема полягає у підборі β у (59). У [5] пропонується вибирати $\beta = 0.1$ для отримання найкращої оцінки.

Варто сказати, що даний підхід вважається одним з найкращих для обчислення функцій від матриць. Використовується у мові програмування MATLAB (функція funm) і бібліотеці SciPy для мови програмування Python (функція funm).

Використавши алгоритм Шура-Парлетта для (19), отримуємо наступні перетворення

$$(F(\mathcal{J}_m^+)g)(x) = w(x)UF(T)U^TV(g), \tag{62}$$

$$(F(\mathcal{J}_m^-)g)(x) = w(x)UF(T)U^TV(g). \tag{63}$$

4. Приклади

4.1. Інтеграли зі змінною межею

Приклад 1. Розглянемо інтеграл з змінною межею на інтервалі $x \in (-\infty, \infty)$

$$I_1(x) = \int_{-\infty}^{x} \left\{(1+t^2)(3+t^2+t^4)\right\}^{-1/2} dt. \tag{64}$$

Даний інтеграл взято з [15]. Це один з прикладів інтегралу зі змінною межею, які пропонуються для розв'язку. Для розв'язку даного інтегралу використовується квадратурна формула (8). Оскільки інтегруємо (64) на $(-\infty, x)$, $x \in (-\infty, \infty)$, то застосуємо конформне відображення

$$\psi(x) = \sinh(x)$$

для підрахунку (7), (10).

Розв'яжемо (64) на інтервалі $(-\infty, 0)$, використовуючи sinc-апроксимацію. Наближений розв'язок $I_1(0)$ з точністю $= 10e-11$ становить 0.76908594260. У таблиці 1 наведено кількість кроків N та абсолютні похибки $\Delta I_1(0)$ для інтегралу $I_1(0)$ при різних значеннях параметра дискретизації h.

Табл. 1.

N	$h = 1/\sqrt{N}$	$h = \pi/\sqrt{N}$	$h = \sqrt{\pi/N}$
20	2,0695E-004	8,3889E-005	6,1558E-007
40	5,4590E-006	8,0149E-006	2,5297E-009
80	3,0386E-008	3,2325E-007	1,6335E-012
160	1,8942E-011	1,5140E-010	8,4377E-014
320	8,3156E-014	1,1546E-014	8,3933E-014

Можна побачити, що найкраща оцінка при $h = \sqrt{\pi/N}$. Проте навіть для інших параметрів дискретизації видно, що для даного підходу похибка падає експоненційно, тобто виконується умова (9).

Приклад 2. Розглянемо інтеграл з змінною межею на інтервалі $x \in (0, 1)$

$$I_2(x) = \int_{0}^{x} t^{-1/2}(1-t)^{-2/3}\, dt. \tag{65}$$

Даний інтеграл взято з [15]. Це один з прикладів інтегралу зі змінною межею, які пропонуються для розв'язку. Для розв'язку даного інтегралу використовується квадратурна формула (8). Оскільки інтегруємо (65) на $(0, x)$, $x \in (0, 1)$, то застосуємо конформне відображення

$$\psi(x) = \ln\left(\frac{a + be^x}{1 + e^x}\right)$$

для підрахунку (7), (10).

Для цього прикладу розв'яжемо (65) на різних інтервалах в діапазоні (0, 1), використовуючи sinc-апроксимацію. Як параметр дискретизації вкажемо $h = \sqrt{\pi/N}$, оскільки вони проявила себе найкраще для Прикладу 1. У таблиці 2 наведено кількість кроків N та абсолютні похибки $\Delta I_2(x)$ для інтегралу $I_2(x)$ при різних значеннях.

Табл. 2.

N	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$
20	0,09173	0,17839	0,34452	0,63110
40	0,05001	0,15731	0,31019	0,62350
80	0,04495	0,14313	0,31262	0,61590
160	0,04496	0,14272	0,31204	0,61449
320	0,04462	0,14330	0,31163	0,61484

Як показують досліди, застосування sinc-методів для (65) не виправдило себе. Похибка після 40 ітерацій перестала спадати експоненційно. Крім того, при збільшенні інтервалу похибка навіть для $N = 20$ приблизно вдвічі. Це означає, що оцінка (9) не виконується, отже, за допомогою методу (8) розв'язати даний інтеграл не можна. Помилка може полягати у тому, що або потрібно підібрати краще конформне відображення, або (65) не підпадає під необхідні умови для виконання sinc-методу.

4.2. QR-алгоритм

Приклад 3. Розглянемо матрицю

$$A = \begin{pmatrix} 7 & 3 & 4 & -11 & -9 & -2 \\ -6 & 4 & -5 & 7 & 1 & 12 \\ -1 & -9 & 2 & 2 & 9 & 1 \\ -8 & 0 & -1 & 5 & 0 & 8 \\ -4 & 3 & -5 & 7 & 2 & 10 \\ 6 & 1 & 4 & -11 & -7 & -1 \end{pmatrix}. \tag{66}$$

Перевіримо виконання базового QR-алгоритму, QR-алгоритму з матрицею Хессенберга і QR-алгоритму з зсувами.

Власні значення матриці B знайдемо за допомогою розкладу $B = XSX^{-1}$, використавши метод з бібліотеки **MathNet.Numerics** для мови **C#**. Тоді

$$S = (1 \pm 2i, 3, 4, 5 \pm 6)^T. \tag{67}$$

Бачимо, що власні значення матриці B є комплексними, це означає, що у випадку виконання QR-алгоритму отримана матриця буде верхньою квазітрикутною матрицею, Тому умовно матрицю A можна подати у вигляді блочної матриці розмірністю 4×4, де на діагональних блоках будуть власні числа. Нехай матриці A_1 і A_2 - отримані матриці внаслідок виконання базового і модифікованого QR-алгоритму відповідно, а A_3 - отримана матриця внаслідок виконаного QR-алгоритму з зсувами. Задамо для A_3 точність $eps = 10e - 20$. Перейдемо до обчислень.

При $k = 12$:

$$A_1^{(12)} = \begin{pmatrix} 4,9989 & -6,0071 & -1,9373 & -11,7947 & -8,1304 & 13,6761 \\ 5,9961 & 4,9740 & -7,1663 & -15,4837 & 5,8666 & 11,0810 \\ -0,0223 & -0,0003 & 4,0531 & 3,3625 & -4,8962 & -11,7078 \\ 0,0000 & 0,0000 & -0,0073 & 2,8269 & 10,9873 & -3,0004 \\ 0,0000 & 0,0000 & 0,0003 & 0,0711 & 1,2058 & -2,0559 \\ 0,0000 & 0,0000 & 0,0005 & -0,0028 & 2,2024 & 0,9413 \end{pmatrix}, \tag{68}$$

$$A_2^{(12)} = \begin{pmatrix} 4,9989 & -6,0071 & -1,9373 & -11,7947 & -8,1304 & 13,6761 \\ 5,9961 & 4,9740 & -7,1663 & -15,4837 & 5,8666 & 11,0810 \\ -0,0223 & -0,0003 & 4,0531 & 3,3625 & -4,8962 & -11,7078 \\ 0,0000 & 0,0000 & -0,0073 & 2,8269 & 10,9873 & -3,0004 \\ 0,0000 & 0,0000 & 0,0003 & 0,0711 & 1,2058 & -2,0559 \\ 0,0000 & 0,0000 & 0,0005 & -0,0028 & 2,2024 & 0,9413 \end{pmatrix}, \tag{69}$$

$$A_3^{(12)} = \begin{pmatrix} 5,0000 & 6,0000 & -3,5225 & 2,3966 & -2,1084 & 14,2821 \\ -6,0000 & 5,0000 & 6,6550 & -7,9076 & -16,4503 & -15,5561 \\ 0,0000 & 0,0000 & 1,4115 & 0,7216 & -2,3095 & -10,6102 \\ 0,0000 & 0,0000 & -5,7778 & 0,5885 & -11,4741 & 0,9803 \\ 0,0000 & 0,0000 & 0,0000 & 0,0000 & 4,0000 & 4,9224 \\ 0,0000 & 0,0000 & 0,0000 & 0,0000 & 0,0000 & 3,0000 \end{pmatrix}. \tag{70}$$

Яскраво продемонстровано, що QR-алгоритму з зсувами (а саме Алгоритм 4) вже став подібний на верхню квазітрикутну матрицю, і власні значення відповідають (67), до того ж з дуже хорошою точністю. При тому A_1 і A_2 ще не закінчили QR-ітерації, щоб перетворити (66) в квазітрикутну. Крім того, ми ніяк не можемо задати точність, лише кількість кроків для ітерації, що робить дані два алгоритми не практичними.

Перевіримо виконання алгоритмів при $k = 40$.

$$A_1^{(40)} = \begin{pmatrix} 5,0000 & -6,0000 & -6,1588 & -11,0353 & -8,9058 & 5,8406 \\ 6,0000 & 5,0000 & 3,8819 & 15,7233 & 14,1821 & 10,4615 \\ 0,0000 & 0,0000 & 4,0000 & 3,0000 & 12,6647 & -1,2671 \\ 0,0000 & 0,0000 & 0,0000 & 3,0000 & -0,2809 & 11,3983 \\ 0,0000 & 0,0000 & 0,0000 & 0,0000 & 1,0000 & -2,0000 \\ 0,0000 & 0,0000 & 0,0000 & 0,0000 & 2,0000 & 1,0000 \end{pmatrix}, \tag{71}$$

$$A_2^{(40)} = \begin{pmatrix} 5,0000 & -6,0000 & -6,1588 & -11,0353 & -8,9058 & 5,8406 \\ 6,0000 & 5,0000 & 3,8819 & 15,7233 & 14,1821 & 10,4615 \\ 0,0000 & 0,0000 & 4,0000 & 3,0000 & 12,6647 & -1,2671 \\ 0,0000 & 0,0000 & 0,0000 & 3,0000 & -0,2809 & 11,3983 \\ 0,0000 & 0,0000 & 0,0000 & 0,0000 & 1,0000 & -2,0000 \\ 0,0000 & 0,0000 & 0,0000 & 0,0000 & 2,0000 & 1,0000 \end{pmatrix}, \tag{72}$$

$$A_3^{(40)} = \begin{pmatrix} 5,0000 & 6,0000 & 7,4970 & -3,6563 & 5,4420 & 19,7172 \\ -6,0000 & 5,0000 & -7,1912 & 1,9192 & -15,6666 & -7,5632 \\ 0,0000 & 0,0000 & 2,3999 & -5,3947 & -9,8639 & 4,7487 \\ 0,0000 & 0,0000 & 1,1047 & -0,3999 & -6,3003 & -9,5387 \\ 0,0000 & 0,0000 & 0,0000 & 0,0000 & 4,0000 & 4,9224 \\ 0,0000 & 0,0000 & 0,0000 & 0,0000 & 0,0000 & 3,0000 \end{pmatrix}. \tag{73}$$

Можна побачити, що матриці A_1 і A_2 збігаються майже однаково до верхньої квазітрикутної матриці і на головній діагоналі власні значення спадають з більшого значення до меншого. A_2 швидше збіглася до верхної квазітрикутної матриці з власними значеннями на діагоналях, проте A_1 має кращу точність. Насправді, завдяки матриці Хессенберга необхідно менше кроків, оскільки вона вже містить нулі у нижній трикутній матриці. Це буде особливо відчуватися при збільшенні розмірності матриці. Також власні значення розташовані у спорядку спадання і можна побачити, що $5 \pm 6i$ і $1 \pm 2i$ подані у матричному вигляді і відповідають діагональним блокам

$$T_{11} = \begin{pmatrix} 5 & 6 \\ -6 & 5 \end{pmatrix}, \tag{74}$$

$$T_{33} = \begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix} \tag{75}$$

Щодо матриці A_3, то варто зауважити, що власні значення матриці A_3 співпадають як з (67), так і з (73), (74). Проте проблема в тому, що вони розташовані хаотично. Крім того, елементи блоку T_{22} ще не пропорційні власним значенням $1 + 2i$, $1 - 2i$. Отже, можна зробити висновок, що QR-алгоритм з зсувами гарантує швидке виконання і перетворення матриці Шура в верхню квазітикутну. Проте власні значення розташовані нелінійно.

4.3. Sinc-згортки

Приклад 4. Розглянемо інтеграл

$$I_3(x) = \int_0^x (x-t)^{-1/2} t^{-1/2}\, dt, \qquad x \in (0,1). \tag{76}$$

Даний інтеграл взято з [15]. Це є одним з прикладів згорток. Для розв'язку даного інтегралу в цьому прикладі будуть розглянути підхід (19). Застосуємо два підходи для розв'язку даного інтегралу функції від матриці в (19) і порівняємо оцінку отриманих результатів.

Виберемо параметри дискретизації $h_1 = 1/\sqrt{N}$ і $h_2 = \sqrt{\pi/N}$. Точний розв'язок інтегралу (76) може бути поданий як

$$I_1(x) = \begin{cases} 0; & x = 0, \\ \pi; & x \neq 0. \end{cases}$$

Оскільки інтегруємо (76) на $(0, x)$, $x \in (0, 1)$, то застосуємо до нього конформне відображення

$$\psi(w) = \ln\Big(\frac{a + be^w}{1 + e^w}\Big). \tag{77}$$

для підрахунку (7), (10).

Модифіковане перетворення Лапласа (14) для інтегрального ядра (76) виглядатиме так

$$F(s) = \sqrt{\pi s}. \tag{78}$$

Завдання полягає у пошуку наближеного розв'язку (76) за допомогою підходу (19). Спочатку розглянемо підхід з декомпозицією матриці

$$A = XSX^{-1} \tag{79}$$

для обчислення функції від матриці (19). Скористаємся для цього методами з бібліотеки **MathNet.Numerics** для мови **C#** для обчислення власних значень S та власних векторів X. Застосувавши до отриманих матриць (20), отримаємо функцію від матриці

$$F_1(A) = XF(S)X^{-1}. \tag{80}$$

Підставимо отримане значення в апроксимаційну формулу (19) і отримаємо наступні значення.

Табл. 3.

N	$\Delta I_3(0.2)$	$\Delta I_3(0.4)$	$\Delta I_3(0.6)$	$\Delta I_3(0.8)$
8	1,03504	0,71084	0,53578	0,48041
16	0,59250	0,42452	0,30140	0,24550
40	0,17527	0,11797	0,11672	0,10002
80	0,04468	0,04039	0,02301	0,03136

У таблиці 3 наведено при h_1 кількість кроків N та абсолютні похибку $\Delta I_3(x)$ для інтегралу $I_3(x)$ на 4 інтервалах зі змінною межею.

У таблиці 4 наведено при h_2 кількість кроків N та абсолютні похибку $\Delta I_3(x)$ для інтегралу $I_3(x)$ на 4 інтервалах зі змінною межею.

Табл. 4.

N	$\Delta I_3(0.2)$	$\Delta I_3(0.4)$	$\Delta I_3(0.6)$	$\Delta I_3(0.8)$
8	0,29152	0,17928	0,22162	0,19741
16	0,12336	0,07462	0,07307	0,04475
40	0,01364	0,02138	0,00241	0,00971
80	0,00106	0,00037	0,00344	0,00470

Згідно з Табл. 3 і 4, похибка спадає експоненційно при збільшенні кроку N і незалежно від вибору інтервалу для (76). При h_2 набагато краще похибка спадає, ніж при h_1, що відповідає умові (17).

Проте доцільно перевірити точність обрахунку (79) і (80). Перевіримо спочатку виконання декомпозиції матриці (79). Порівняємо це значення з точним значенням (79). Як параметр дискретизації виберемо h_2, оскільки він показав найкращі результати у точності. Оскільки мова тут йде лише про формування матриці A, то нема значення, на якому інтервалі воно відбувається. Перевіримо для I_3 в $x = 0.2$.

Табл. 5.

N	$\|A - XSX^{-1}\|_\infty$
8	7,3825e-013
16	4,8461e-011
40	3,4375e-007
80	6,8849e-004

Відповідно до Табл.5, для випадку застосування власного методу розкладу похибка критично спадає при збільшенні кількості вузлів A. Тоді як у випадку розкладу Шура у випадку збільшення вузлів похибка майже не змінюється.

Тепер перевіримо обчислення функції від матриці (80). Порівняємо це значення з точним значенням (80). Як параметр дискретизації виберемо h_2, оскільки він показав найкращі результати у точності. Точний розв'язок F отримаємо, застосувавши до даної функції від матриці метод **funm** в мові програмування **Matlab**. Перевіримо для I_3 в $x = 0.2$.

Табл. 6.

N	$\|F - F_1\|$
8	1,7701e+000
16	1,9094e+000
40	1,9642e+000
80	1,7701e+000

Табл. 7.

N	$\|A - UTU^T\|_\infty$
8	1,7659e-014
16	1,9965e-014
40	7,4135e-014
80	1,6848e-013

Як можна побачити, існує проблема у обчисленні функції від матриці (80). Похибка достатньо велика, тому існує потреба застосувати інакший підхід для (19). Застосуємо алгоритм Шура-Парлетта для інтегралу, тобто виконаємо (62). Спочатку перевіримо виконання розкладу Шура

$$A = UTU^T \tag{81}$$

за допомогою Алгоритму 5, а згодом застосуємо Алгоритм 6 для (81). Тоді отримаємо таку функцію від матриці

$$F_2(A) = UF(T)U^T. \tag{82}$$

Використаємо ті ж самі вхідні дані, що й для попереднього підходу, конформне відображення (77), перетворення Лапласса (78). Порівняємо виконання (81) і (82) з точними результатами. Точний розв'язок F отримаємо, застосувавши до даної

Табл. 8.

N	$\|F - F_2\|_\infty$
8	1,1550e-014
16	1,1817e-014
40	3,0101e-014
80	5,4858e-014

Табл. 9.

N	$\Delta I_3(0.2)$	$\Delta I_3(0.4)$	$\Delta I_3(0.6)$	$\Delta I_3(0.8)$
8	2,30482	2,36232	2,21603	2,16842
16	2,43214	2,52602	2,45646	2,44212
40	2,73069	2,56433	2,85190	2,64404
80	2,83593	2,92762	2,74941	2,79600

функції від матриці метод **funm** в мові програмування **Matlab**. Перевіримо для I_3 в $x = 0.2$.

На основі табл.7 і табл.8 можна переконатися, що точність набагато покращилася, і підхід Шура-Парлетта краще рахує функцію від матриці, ніж для попереднього випадку. Тепер можна спробувати розв'язати інтеграл (76), використовуючи формулу (62). Тоді отримуємо такі результати

У табл 9. наведено абсолютну похибку для розв'язку інтегралу (76). На жаль, отримати хороші результати не вдалося. Хоча функцію від матриці підхід Шура-Парлетта порахував точніше, проте самий sinc-метод видає результат з великою похибкою. Ймовірно необхідно дослідити детальніше sinc-метод для апроксимації sinc-згорток і адаптувати його під підхід для обчислення функції від матриці, наведений у цій роботі.

Висновок

У цій роботі розглянуто основні принципи sinc-методів для інтегралів зі змінною межею та згорток. Описано такі ключові поняття як sinc-функція, конформні відображення, згортки, розклад Шура, функція від матриці, ряд Тейлора, алгоритм Шура-Парлетта тощо. У Розділі 1 сформульвано sinc-методи для розв'язування інтегралів зі змінною межею та згорток, показано оцінки збіжностей для цих методів. Описано, чому для sinc-згорток погано обчислювати функції від матриці, застосовуючи до матриці власний розклад матриці.

Для sinc-згорток розглянуто 2 способи здійснення декомпозиції матриці і обчислення функції від матриці: власний розклад матриці і обчислення функції діагональних елементів, розклад Шура і алгоритм Парлетта. У Розділі 2 описано формулювання розкладу Шура для матриці з дійсними або комплексними елементами, подано реалізації QR-алгоритмів для здійснення розкладу Шура, а саме базовий QR-алгоритм, QR-алгоритм з застосуванням перетворення Хаусхолдера та QR-алгоритм з зсувами. Для QR-алгоритму з зсувами розглянуто дві окремі реалізації для матриці з дійсними та комплексними елементами. Порівняно переваги і недоліки використання відповідних QR-алгоритмів для розкладу Шура.

У Розілі 3 описано поняття функції від матриці, прості приклади та спосіб її подання через канонічну форму Жордана. Розглянуто властивості функцій від матриць, сформульовано метод Шура-Парлетта для обчислення функції від матриці. Наведено спосіб, як застосувати метод Шура-Парлетта для sinc-згорток.

Наведено приклади обчислення інтегральних рівнянь з застосуванням sinc-методів для інтегралів зі змінною межею та згорток. Порівняно різницю між точним і наближеним розв'язком інтегралу на різних проміжках і для різних параметрів дискретизації. Крім того, наведено приклади QR-алгоритмів для матриць з дійсними та комплексними власними числами, показано кількість ітерацій і надано оцінку алгоритмів. Показано, чому QR-алгоритм з зсувами використовується на практиці, його переваги і недоліки.

Важливою частиною є застосування різних методів для декомпозиції матриці та обчислення функції від матриці. Продемонстровано використання розкладу Шура для декомпозиції матриці та алгоритму Парлетта для обчислення функції для матриці в sinc-згортці. Перевірено на практиці, що метод Шура-Парлетта дієвий і швидкий для обчислення функції від матриці. Проте застосування його на практиці до sinc-апроксимацій згорток кращого результату не дало.

Список використаної літератури

[1] Хапко Р., Вінтоняк Н. Про використання sinc-квадратур для наближеного обчислення інтегралів з різними типами особливостей. В:Вісник Львівського університету. Серія прикладна математика та інформатика11(2006),с.35—42.

[2] Шахно С., Дудикевич А. Практична реалізація чисельних методів лінійної алгебри. Навчальний посібник, 2009, с.104-110.

[3] Arbenz P. Numerical Methods for Solving Large Scale Eigenvalue Problems (Spring semester 2018), Chapter 4, c.64-84.
url: http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter4.pdf.

[4] Baumann G., Stenger F. Sinc-approximations of Fractional Operators: A Computing Approach, c.446-450, c.270-280.

[5] Davies I., Highman N. J. A Schur–Parlett algorithm for computing matrix functions, c.464-485.

[6] Gimeno J. On computation of matrix logarithm times a vector, c.7-22.

[7] Golub G. and Van Loan C. Matrix Computations, third ed., 1996.

[8] Francis J. The QR transformation – Parts 1 and 2, Comput. J., 4 (1961-1962), c. 265–271, c.332–345.

[9] Higham N., Nicholas J. Matrix Functions: A Short Course, c.5-19.

[10] Higham N., Nicholas J. A Schur-Parlett Algorithm for Computing Matrix Functions, c.12-19.

[11] Kerl J. The Householder transformation in numerical linear algebra, 2008.

[12] Stenger F. Numerical methods based on Whittaker cardinal, or sinc functions. B: SIAM review 23.2(1981), c.165—224.

[13] Stenger F. Numerical methods based on sinc and analytic functions. Springer, 1993.

[14] Stenger F.Summary of Sinc numerical methods».Англ.B:Journal of Computational and Applied Mathematics 121.1(2000), c.379—420.

[15] Stenger F.Handbook of sinc numerical methods.Англ.CRCPress,2011, c.92-100.

[16] Wilkinson J. H. The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

[17] Beresford N. Parlett. A recurrence among the elements of functions of triangular matrices. Linear Algebra Appl., 14:117–121, 1976.