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

Дослідження числового розв'язку задачі про нелінійні коливання математичного маятника

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

(підпис) Ханик В.А. (прізвище та ініціали)

Керівник (підпис) доц. Муха І.С. (прізвище та ініціали)

Рецензент (підпис) доц. Вовк В. Д. (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 _____

" __" _____ 2021 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

_____Ханику Віктору Андрійовичу_____
(прізвище, ім'я, по батькові)

1.Тема роботи Дослідження Числового розв'язку задачі про нелінійні коливання математичного маятника

керівник роботи____канд. фіз-мат наук, доцент Муха І. С.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від затверджені Вченою радою факультету від "22" вересня 2021 року № 6

2. Строк подання студентом роботи 16 травня 2022 року

3. Вихідні дані до роботи _____

1. Полянин А.Д., Зайцев В.Ф. Справочник по нелинейным уравнениям математической физики: Точные решения.- М.:ФИЗМАТЛИТ,2002 – 432с.
2. Тихонов А.Н., Самарский А.А. Уравнения математической физики. - М. Наука 1977 – 728с
3. Крылов, Н.М. Введение в нелинейную механику : (Приближенные и асимптотические методы нелинейной механики) / Акад. Н.М. Крылов и д-р Н.Н. Боголюбов. - Киев : Изд-во АН УССР, 1937.

4. Зміст магістерської роботи (перелік питань, які потрібно розробити) _____
Ознайомитись з літературними джерелами, сформулювати математичну модель, вибрати методи для реалізації поставлених задач. Розробити програмне забезпечення та провести числові експерименти.
_____

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

Скріншоти екрана з графіками.
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
всі розділи	Муха І.С.		

7. Дата видачі завдання 07.09.2021 р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Дослідження проблематики та формулювання математичної моделі	30.10.2021	виконано
2.	Вибір методів та створення чисельної схеми для розв'язування поставленої задачі	15.12.2021	виконано
3.	Розробка програмного забезпечення	31.01.2022	виконано
4.	Дослідження та аналіз результатів	31.03.2022	виконано
5.	Формулювання дипломної роботи	07.05.2022	виконано

Студент Ханик В. А.
(підпис) (прізвище та ініціали)

Керівник роботи **Муха** І.С.
(підпис) (прізвище та ініціали)

Зміст

Вступ

Незважаючи на різноманіття коливальних процесів, що зустрічаються в природі, існують спільні закономірності таких явищ і математичних методів їх дослідження. Однією із найбільш зручних для розгляду моделей коливальних систем є теоретична модель математичного маятника.

Згідно з даною моделлю розглядається точка з масою m, підвішена на нерозтяжній продовгуватій нитці довжиною l, що здійснює коливання у вертикальній площині під впливом сил тяжіння з прискоренням вільного падіння g. Розглядаються коливання в одній площині. Модель нехтує розмірами тіла, деформацією підвісу та тертям в точці підвісу

В ході виконання роботи досліджуються розв’язки рівняння нелінійної механіки для коливання маятника, що розгойдується з кутовою швидкістю протягом деякого часу.

Для візуалізації розв’язку найдоречнішим є застосування графічної бібліотеки OpenGl. OpenGL — (англ. Open Graphics Library — відкрита графічна бібліотека) — специфікація, що визначає незалежний від мови програмування кросс-платформовий програмний інтерфейс (API) для написання застосунків, що використовують 2D та 3D комп'ютерну графіку. Цей інтерфейс містить понад 250 функцій, які можуть використовуватися для візуалізації складних тривимірних сцен з простих примітивів. Широко застосовується при візуалізації наукових даних.

Розділ 1

Рівняння коливань маятника

Розглянемо рівняння коливань маятника, що розгойдується з кутовою швидкістю Ω протягом 5 секунд, після чого здійснює самостійні коливання [1].

$$\ddot{\theta} + \left(\omega^2 - \frac{\mathrm{A}\Omega^2 \sin(\Omega t)}{L}\right) sin\theta = 0 \tag{1.1}$$

Використовуючи вказані умови розглянемо наступну крайову задачу:

$$\begin{cases} \ddot{\theta} + \left(\omega^2 - \dfrac{\mathrm{A}\Omega^2 \sin(\Omega t)}{L}\right) sin\theta = 0 \\ \theta(0) = 0 \\ \dot{\theta}(5) = 0 \end{cases} \tag{1.2}$$

Дану задачу будемо розв’язувати з використанням методу Ньютона-Канторовича та різницевого методу сіток.

Розділ 2

2.1 Метод Ньютона-Канторовича

Розглянемо цей метод на прикладі задачі:

$$\frac{d}{dx}\left[\left(C_0 + C_1 u(\vec{\mathbf{x}})\right)\frac{du(\vec{\mathbf{x}})}{dx}\right] = f(\vec{\mathbf{x}}) \tag{2.1}$$

Нехай $u^{(i)}(\vec{\mathbf{x}})$ – деяке наближення до розв'язку задачі $u(\vec{\mathbf{x}})$. Подамо розв'язок у вигляді $u(\vec{\mathbf{x}}) = u^{(i)}(\vec{\mathbf{x}}) + \Delta u^{(i)}(\vec{\mathbf{x}})$. Отримаємо

$$\begin{aligned}
&\frac{d}{dx}\left[\left(C_0 + C_1 u^{(i)}(\vec{\mathbf{x}})\right)\left(\frac{du^{(i)}(\vec{\mathbf{x}})}{dx}\right)\right] + \\
&+\frac{d}{dx}\left[C_1 \Delta u^{(i)}(\vec{\mathbf{x}})\left(\frac{du^{(i)}(\vec{\mathbf{x}})}{dx}\right)\right] + \\
&+\frac{d}{dx}\left[\left(C_0 + C_1 u^{(i)}(\vec{\mathbf{x}})\right)\left(\frac{d\Delta u^{(i)}(\vec{\mathbf{x}})}{dx}\right)\right] + \\
&+\frac{d}{dx}\left[C_1 \Delta u^{(i)}(\vec{\mathbf{x}})\left(\frac{d\Delta u^{(i)}(\vec{\mathbf{x}})}{dx}\right)\right] = f(\vec{\mathbf{x}})
\end{aligned} \tag{2.2}$$

За умови, що $\left\|\Delta u^{(i)}(\vec{\mathbf{x}})\right\|_2 << 1$, знехтуємо четвертим доданком в останньому співвідношенні. Отримаємо лінійну крайову задачу

$$\left(C_0 + C_1 u^{(i)}(\vec{\mathbf{x}})\right)\left(\frac{d^2 \Delta u^{(i)}(\vec{\mathbf{x}})}{dx^2}\right) + 2C_1 \frac{du^{(i)}(\vec{\mathbf{x}})}{dx}\frac{d\Delta u^{(i)}(\vec{\mathbf{x}})}{dx} +$$

$$+C_1 \frac{d^2 u^{(i)}(\vec{\mathbf{x}})}{dx^2}\Delta u^{(i)}(\vec{\mathbf{x}}) = f(\vec{\mathbf{x}}) - C_1\left(\frac{du^{(i)}(\vec{\mathbf{x}})}{dx}\right)^2 - \tag{2.3}$$

$$-\left(C_0 + C_1 u^{(i)}(\vec{\mathbf{x}})\right)\frac{d^2 u^{(i)}(\vec{\mathbf{x}})}{dx^2}$$

$$\Delta u^{(i)}(\vec{\mathbf{x}})\Big|_{\vec{\mathbf{x}}\in S} = 0$$

Після того, як задача буде розв'язана, приймемо $u^{(i+1)}(\vec{\mathbf{x}}) = u^{(i)}(\vec{\mathbf{x}}) + \Delta u^{(i)}(\vec{\mathbf{x}})$. Умова закінчення ітераційного процесу:

$$\left\|\Delta u^{(i)}(\vec{\mathbf{x}})\right\|_2 < \varepsilon \left\|u^{(i)}(\vec{\mathbf{x}})\right\|_2 \tag{2.4}$$

де ε – точність обчислень. Якщо $i > i_{\max}$, то процес зупиняють (страховка від розбіжності або зациклювання).

2.2 Різницевий метод. Метод сіток

Для апроксимації однієї і тої ж самої задачі можна використовувати різні сіткові схеми. З поміж таких схем необхідно обрати таку, що на відносно грубих сітках дозволяла б отримати бажану точність для наближення розв'язку. Отже, сіткова схема повинна відповідати основним властивостям операторного рівняння, таких як: самоспряженість, додатня визначеність тощо. Враховуючи той факт, що властивості диференціальної задачі в математичній фізиці базуються на фізичних законах, то можна стверджувати, що вимогою до сіткової схеми є виконання основних фізичних законів для розв'язування на сітці. Розглянемо метод побудови сіткових схем для апроксимації крайових задач для звичайних диференціальних

рівнянь, що широко застосовується на практиці [2]. Також, варто зауважити, що зважаючи на той факт, що ці схеми виражаються через різниці шуканих функцій — їх називають різницевими схемами.

Для того, щоб подати різницеву схему для наближеного опису диференціального рівняння, виконуються кроки в наступній послідовності:

1) Неперервна зміна аргументу замінюється на дискретну область його зміни. Іншими словами, виконується дискретизація області неперервної зміни.
2) Диференціальний оператор замінюється деяким різницевим оператором.

В результаті виконання цих кроків, отримується система алгебраїчних рівнянь. Таким чином, задача чисельного розв’язування диференціального рівняння зводиться до пошуку розв’язків системи лінійних алгебраїчних рівнянь.

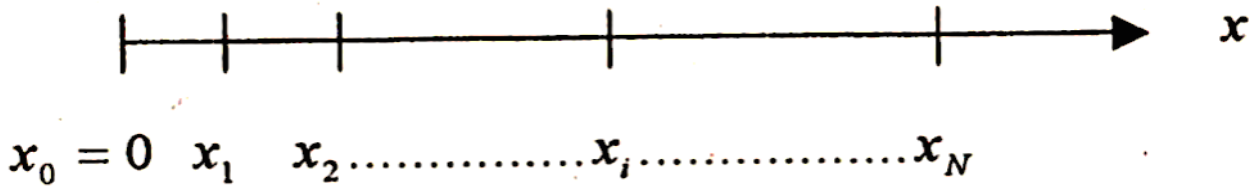
У разі чисельного розв’язування тієї, чи іншої математичної задачі ми не можемо відтворити різницевий розв’язок для всіх значень аргументу, що змінюється в деякій області евклідового простору. Отже, в такій області обирають скінчену множину точок, в якій і будуть шукати наближений розв’язок.

Отриману множину точок називають сіткою, окремі точки якої використовуються в якості вузлів сітки. Функція, що визначена у вузлах сітки називається сітковою функцією. Отже, ми замінили область неперервної заміни аргумента сіткою, тобто областю дискретної зміни аргумента. Іншими словами, було виконано апроксимацію простору роз в’язків диференціального рівняння простором сіткових функцій. Близькість до точного розв’язку та властивості різницевого розв’язку напряму залежать від вибору сітки.

Розглянемо деяку рівномірну сітку на відрізку:

Поділимо таку одиничну область [0,1] на N рівних частин. Відстань між сусідніми вузлами $x_i - x_{i-1} = h = 1/N$ будемо в подальшому називати кроком сітки. Точки

розбиття xi =ih – називаються вузлами сітки. Множина всіх вузлів yh =xi =ih, i=1,2,..., N−1 і складає сітку [9].



Нехай задано лінійний диференціальний оператор L, що діє на функцію v = v(x). Замінимо похідні, що входять у Lv різницевими співвідношеннями. Унаслідок цього замість Lv одержимо такий різницевий вираз, що буде лінійною комбінацією значень сіткової функції v_h на деякій множині вузлів сітки, яку називають шаблоном. Різницеві апроксимації оператора L спочатку вивчають локально, тобто в доцільній фіксованій точці x простору. Якщо v(x) - неперервна функція, то $v_h(x) = v(h)$. Перш ніж перейти до різницевої апроксимації оператора L, потрібно вибрати шаблон, тобто задати множину сусідніх до вузла x вузлів, у яких значення сіткової функції v(x) можна використовувати для апроксимації оператора L. Для того, щоб написати різницеву апроксимацію першої похідної, потрібно використати 2 точки, тобто брати двоточковий шаблон. Зафіксуємо деяку точку x на осі Ox і візьмемо точки x - h, x + h, де h > 0. Для апроксимації Lv можна використати одне з таких співвідношень:

$$L_h^+ v = \frac{v(x+h) - v(x)}{h}$$
$$L_h^- v = \frac{-v(x-h) + v(x)}{h} \qquad (2.5)$$

Для того, щоб написати різницеву апроксимацію другої похідної, потрібно використати 3 точки (x - h, x, x+h), тобто брати триточковий шаблон.

$$L_h v = \frac{v(x+h) - 2v(x) + v(x-h)}{h^2} \tag{2.6}$$

2.3 Лінеаризація та розв'язування

Розглянемо рівняння:

$$\ddot{\theta} + \left(\omega^2 - \frac{\mathrm{A}\Omega^2 \sin(\Omega t)}{L}\right) sin\theta = 0 \tag{2.7}$$

Лінеаризацію задачі здійснимо методом Ньютона-Канторовича. Для цього введемо $\theta^{(i)}$ – деяке наближення. Тоді,

$$\theta = \theta^{(i)} + \Delta\theta^{(i)} \tag{2.8}$$

Розкладемо нелінійну частину:

$$\begin{aligned}\sin\left(\theta^{(i)} + \Delta\theta^{(i)}\right) = \sin\left(\theta^{(i)}\right) + \Delta\theta^{(i)}\cos\left(\theta^{(i)}\right) - \frac{1}{2}\Delta\theta^{(i)^2}\sin\left(\theta^{(i)}\right) - \\ \frac{1}{6}\Delta\theta^{(i)^3}\cos\left(\theta^{(i)}\right) + \frac{1}{24}\Delta\theta^{(i)^4}\sin\left(\theta^{(i)}\right) + \frac{1}{120}\Delta\theta^{(i)^5}\cos\left(\theta^{(i)}\right) + O\left(x\theta^{(i)^6}\right)\end{aligned} \tag{2.9}$$

Нехай $\Delta\theta^{(i)}$ – мала функція, вищими степенями якої ми можемо знехтувати при подальших обрахунках. Тоді подаємо нелінійну частину у такому вигляді:

$$\begin{gathered}\sin\left(\theta^{(i)}+\Delta\theta^{(i)}\right)=\sin\left(\theta^{(i)}\right)+\cos\left(\theta^{(i)}\right)\Delta\theta^{(i)}\\ \ddot{\theta}^{(i)}+\omega^2\sin\left(\theta^{(i)}\right)+\Delta\ddot{\theta}^{(i)}+\omega^2\cos\left(\theta^{(i)}\right)\Delta\theta^{(i)}=0\end{gathered} \tag{2.10}$$

$$\begin{gathered}\Delta\ddot{\theta}^{(i)}+\cos\left(\theta^{(i)}\right)\Delta\theta^{(i)}\left(\omega^2-\frac{\mathrm{A}\Omega^2\sin(\Omega t)}{L}\right)\\ =\left(-\omega^2+\frac{\mathrm{A}\Omega^2\sin(\Omega t)}{L}\right)\sin\left(\theta^{(i)}\right)-\ddot{\theta}^{(i)}\end{gathered} \tag{2.11}$$

Запишемо різницеві формули:

$$\begin{gathered}\Delta\ddot{\theta}^{(i)}=\frac{\Delta\theta_{j+1}^{(i)}-2\Delta\theta_j^{(i)}+\Delta\theta_{j-1}^{(i)}}{h^2}\\ \Delta\dot{\theta}^{(i)}=\frac{\Delta\theta_{j+1}^{(i)}-\Delta\theta_{j-1}^{(i)}}{2h}\\ \Delta\theta^{(i)}=\Delta\theta_j^{(i)}\end{gathered} \tag{2.12}$$

де h – крок розбиття та (i) – номер ітерації.

Для обрахування h використовується формула:

$$h=\frac{T}{n} \tag{2.13}$$

де n – кількість розбиттів , T – період коливань.

Таким чином:

$$t_j = j\frac{T}{n} \tag{2.14}$$

t	t_0	t_1			t_{n-1}	$t_n = T$
$x^{(i)}$	$x_0^{(i)}$	$x_1^{(i)}$			$x_{n-1}^{(i)}$	$x_n^{(i)}$
$\Delta x^{(i)}$	$\Delta x_0^{(i)}$	$\Delta x_1^{(i)}$			$\Delta x_{n-1}^{(i)}$	$\Delta x_n^{(i)}$

Таблиця 1

Перепишемо рівняння згідно з різницевими формулами:

$$\frac{\Delta\theta_{j+1}^{(i)} - 2\Delta\theta_j^{(i)} + \Delta\theta_{j-1}^{(i)}}{h^2} + \cos\left(\theta^{(i)}\right)\Delta\theta_j^{(i)}\left(\omega^2 - \frac{\mathrm{A}\Omega^2\sin(\Omega t)}{L}\right) = \left(-\omega^2 + \frac{\mathrm{A}\Omega^2\sin(\Omega t)}{L}\right)\sin\left(\theta^{(i)}\right) - \frac{\theta_{j+1}^{(i)} - 2\theta_j^{(i)} + \theta_{j-1}^{(i)}}{h^2} \tag{2.15}$$

Наступним етапом потрібно сформувати систему алгебричних лінійних рівнянь. Запишемо рівняння в наступному вигляді:

$$P_{1j}\Delta x_{(j-1)} + P_{2j}\Delta x_{(j)} + P_{3j}\Delta x_{(j+1)} = P_{4j} \tag{2.16}$$

де $P_{1j},\ P_{2j},\ P_{3j}$ – коефіцієнти, отримані під час зведення отриманого рівняння при $\Delta x_{(j-1)}$, $\Delta x_{(j)}$, та $\Delta x_{(j+1)}$ відповідно. З отриманих коефіцієнтів формуємо матрицю А. Значення P_{4j} попередньо відомі та записані в таблицю.

Схематичний вигляд СЛАР:

P_{21}	P_{31}						$\Delta\theta_{(1)}$		P_{41}
P_{12}	P_{22}	P_{32}					$\Delta\theta_{(2)}$		P_{42}
	P_{13}	P_{23}	P_{33}				$\Delta\theta_{(3)}$		P_{43}
		P_{14}	P_{24}	P_{34}			$\Delta\theta_{(4)}$	=	P_{44}
			P_{15}	P_{25}	P_{35}		$\Delta\theta_{(5)}$		P_{45}
				P_{16}	P_{26}	P_{36}	$\Delta\theta_{(6)}$		P_{46}
					P_{17}	P_{27}	$\Delta\theta_{(7)}$		P_{47}

Таблиця 2

В прикладі зображена СЛАР для n=8.

В результаті розв’язання такої системи лінійних алгебраїчних рівнянь отримуємо значення вектору $\Delta x^{(i)}$ на даній ітерації та записуємо відомі значення у відповідний рядок Таблиці 1. Далі обчислюємо $x^{(i+1)}$ за формулою

$$\theta^{(i+1)}=\theta^{(i)}+\Delta\theta^{(i)} \tag{2.17}$$

Ітераційний процес зупиняється, якщо

$$\left\|\Delta\theta^{(i)}\right\| \leq \varepsilon\left\|\theta^{(i)}\right\| \tag{2.18}$$

Розділ 3

3.1 Програмна реалізація

Програмна реалізація алгоритму описаного в попередньому розділі здійснена за допомогою пакету Matlab. Для візуалізації отриманих результатів та створення анімації використано пакет OpenGL [3] і мову програмування C++ в середовищі Visual Studio 2019.

В результаті виконання програми, отримуємо файл з протабульованими значеннями наступного вигляду з відхиленням маятника від початкового положення відносно часу:

t	x
0	5,54433313
0,25	-1,9414099
0,5	3,00091949
0,75	-6,1740376
1	2,6669286
1,25	5,7556657
1,5	3,14470596
1,75	0,75320107
2	0.00751545
2,25	0,61705467
2,5	2,15175158
2,75	3,76061653
3	3,98384593
3,25	1,70027929
3,5	-1,966187
3,75	-3,5038699
4	-0,8623465
4,25	2,71461867
4,5	2,13819596
4,75	-1,6483568

3.2 Графічна реалізація

3.2.1 Графічний Інтерфейс OpenGL

Комп’ютерна графіка (computer graphics) – це область інформатики (науки про комп’ютери), до вивчення якої можна віднести усі частини формування зображень з використанням комп’ютерної техніки. Дана галузь розпочала свій розвиток майже п’ятдесят років тому. В ті роки вдалось добитись відображення лише декількох десятків відрізків на екрані електро-променевої трубки, а сучасні системи машинної графіки дозволяють створювати зображення, які практично не відрізняються за якістю від фотознімків [3].

Комп’ютерна графіка на сьогоднішній день практично повністю реалізувала себе як наукову галузь. Враховуючи використання у великій кількості наукових та інженерних дисциплін для наочності сприйняття і передачі інформації, можна це сміливо стверджувати. Комп’ютерна графіка здобула широке застосування у повсякденному житті - у бізнесі, медицині, індустрії розваг та в значній кількості інших областей. Учені знаходять застосування комп’ютерної графіки у аналізі результатів моделювання. Для прикладу, архітектори використовують тривимірну графіку для створення тривимірних моделей своїх проектів.

Розвиток комп’ютерної графіки визначається двома факторами: справжніми вимогами користувачів та можливості апаратного та програмного забезпечення. Більша частка сьогоднішніх досліджень в галузі графіки пов’язана з покращенням ефективності та швидкості генерації зображення. З метою отримання реалістичного зображення природної сцени, графічна програма має реалізувати імітацію ефектів реального заломлення та відображення світла від фізичних об’єктів. Так, в комп’ютерній графіці спостерігається рух до впровадження в

графічні алгоритми оптимізованих апроксимацій фізичних принципів, з метою кращої імітації складних взаємодій між об’єктами та середовищем.

3.2.2 Історія і ідеї OpenGL

Одним із найбільш застосовних інтерфейсів для розробки застосунків у царині двовимірної і тривимірної графіки є OpenGL.

OpenGL - Open Graphics Language, відкрита графічна мова. Термінологія "відкритості" – має на увазі незалежний від виробників програмний код.

OpenGL був розроблений у 1992 році компанією Silicon Graphics Incorporated (SGI), що займалась створенням високотехнологічного графічного обладнання та програмних засобів. Недовго після створення OpenGL перетворився в широко використовуваний інтерфейс для прикладних графічних програм (graphics application programming interface - API). У ньому представленні інструменти для малювання, які діють з допомогою широкого набору можливих функцій. Даний інтерфейс є домтупний для усіх класів комп’ютерних систем. OpenGL пропонує широкий та відносно зручний вибір API- функцій для 2D - графіки та обробки зображення, проте його дійсні переваги можуть себе проявити саме у 3D- графіці.

Головна ідея OpenGL полягає у тому, що графічна бібліотека має бути апаратно незалежною та легко розширюваною. Велика частина сучасних графічних процесорів (зазвичай це відеокарти) напряму сприймають OpenGL як мову вхідного рівня без будь-якої потреби у трансляції.

OpenGL оперує графічними примітивами «початкового рівня», такими як точки тривимірного простору, відрізки прямих, випуклі полігони та растрові зображення. До важливих функцій можна віднести натягування бітових карт на

тривимірні поверхні, згладження переходів кольорів –як локальне, в рамках одного об’єкта, так і глобальне, по всій сцені.

З точки зору програміста, OpenGL- це система викликів процедур з передачею їм параметрів, тобто ця мова представляє собою Call Level API. OpenGL складений з набору бібліотек. Ключовими моментами з точки зору виробництва, особливо в мережному оточенні, є наявність двох режимів : покрокового і пакетного. Пакет групує команди опису об’єктів і режими в пакет, що називається списком відображення (display list). Недолік списку відображення проявляється при частому внесенні в нього змін. Для роботи з такими «змінними» об’єктами в OpenGL передбачений режим прямого відображення, коли кожен додаток інтерпретується в порядку подачі, не очікуючи закриваючого списку тегу чи якоїсь подібної команди на промальовування. В практиці обидва методи знайшли широке застосування [4].

3.2.3 Специфікація OpenGL

На базовому рівні, OpenGL — це по своїй суті специфікація, іншими словами, — деякий документ, що описує набір функцій та їх поведінку. Базуючись на цих специфікаціях виробники апаратного забезпечення створюють власні реалізації — бібліотеки функцій, що відповідають заявленій в OpenGL специфікації. Такі реалізації спроектовані з метою можливості використовувати засоби апаратного забезпечення. У ситуаціях, де апаратне прискорення не допускається, виконання функцій здійснюється за допомогою програмного забезпечення. Виробники зобов’язані пройти кваліфікаційні тести на відповідність, перед тим, як їхню реалізацію можна буде класифікувати, як реалізацію OpenGL. Таким чином, розробникам програмного забезпечення необхідно всього лиш навчитися

використовувати описані у специфікації функції, і залишити їхню реалізацію розробникам апаратного забезпечення.

Ефективні реалізації OpenGL існують для операційних систем Linux, MacOS X, Microsoft Windows та багатьох UNIX-подібних ОС, а також для таких ігрових консолей, як Sony PlayStation. Різні програмні реалізації OpenGL існують для платформ, виробники яких не підтримують дану специфікацію. Відкрита (open source) бібліотека Mesa — повністю OpenGL сумісний програмний API. Однак, для того, щоб заради уникнення витрат на ліцензування, пов'язаних з формалізацією, що є вимогою для офіційного признання реалізації, Mesa не є офіційною реалізацією специфікації, незважаючи на повну сумісність з нею.

3.2.4 Архітектура OpenGL

OpenGL вирішує такі завдання: приховати складності адаптації різних 3D-прискорювачів, використовуючи надання розробнику єдиного API. Приховати відмінності у можливостях різних апаратних платформ з вимогою реалізації відсутньої функціональності з використанням програмної емуляції. Ключовим принципом роботи OpenGL є отримання наборів векторних примітивів у вигляді набору точок, ліній, а також багатокутників із послідовною обробкою отриманих даних для побудови растрової картинки на екрані або в пам'яті комп’ютера. Векторні трансформації та растеризація виконуються графічним конвеєром (graphics pipeline), який власне і є по своїй суті дискретним автоматом. Переважна більшість команд OpenGL потрапляють в одну із двох груп: вони або додають примітиви на вхід у конвеєр, або конфігурують сам конвеєр на різні виконання вказаних трансформацій. OpenGL є низькорівневим процедурним API, що змушує користувача вказувати точну послідовність кроків, для побудуви результуючої

растрової графіки (імперативний підхід). Це і є важливою відмінністю від дескрипторних підходів, у яких сцена повністю передається у вигляді деякої структури даних, що обробляється та вибудовується безпосередньо на екрані. З одного боку, імперативний підхід вимагає від користувача ґрунтовного знання законів тривимірної графіки та математичних моделей, з іншого боку — надає свободу впровадження різних інновацій [5].

3.2.5 Пріоритети OpenGL

Характерними особливостями OpenGL є:

1. **Продуктивність.** З самого початку в OpenGL була закладена можливість прорисовування динамічних сцен. Для отримання потрібних результатів в систему введено множину параметрів, чи інакше кажучи – режимів малювання. Якщо деякий режим чи комбінація режимів на даному устаткуванні не в змозі забезпечити інтерактивної взаємодії чи необхідності частоти оновлення сцени, то користувач чи сама програма повинні бути в змозі підключати стільки додаткових функцій.

2. **Ортогональність.** По можливості всі функції OpenGL є ортогональними, тобто незалежними. Їх можна використовувати в довільній комбінації.

3. **Повнота.** Наскільки це можливо, OpenGL відповідає набору функцій, представленому сучасними апаратними засобами графічної акселерації. OpenGL старається уникати всього, що повинно бути реалізовано програмно. З іншої сторони, гарантується отримання робочої картинки, навіть якщо продуктивність не дозволяє отримати її з усіма деталями.

4.**Інтерперабельність.** В мережі важко передавати дані між різними платформами. Тому OpenGL орієнтований на роботу в режимі – клієнт-сервер, навіть якщо і клієнт і сервер розміщені на одному комп'ютері.

5. **Розширюваність.** В OpenGL вкладені механізми підключення нових функцій. З іншої сторони, нестабільний інтерфейс ускладнює життя розробників, тому нові можливості назбируються тривалий час і застосовуються згідно з виходом нової версії.

6. **Стабільність.** Доповнення і зміни в стандарті реалізуються таким чином, щоб зберегти відповідність з розробленим раніше програмним забезпеченням.

7. **Надійність і переносність.** OpenGL гарантує однаковий візуальний результат незалежно від типу використовуваної операційної системи і організації відображення інформації.

8.**Легкість застосування.** Стандарт OpenGL має продуману структуру і інтуїтивно зрозумілий інтерфейс. Необхідні функції для забезпечення сумісності з різним устаткуванням реалізовані на рівні бібліотеки.

3.2.6 Синтаксис команд OpenGL

Команди бібліотеки OpenGL використовують префікс gl. Кожне слово, яке становить найменування команди, розпочинається із великої літери (наприклад, функція glClearColor ()). Точно так само імена констант, визначених у бібліотеці OpenGL, починаються з префікса GL_, записуються цілком великими літерами і використовують символи підкреслення, щоб розділити окремі слова (наприклад, GL__COLOR_BUFFER_BIT).

Деякі частини команди додаються в кінець найменування деяких команд (наприклад, 3f у функціях glColor3f () і glVertex3f ()). Дійсно, частина Color в найменуванні функції glColor3f () достатня для того, щоб визначити цю команду як команду, яка встановлює поточний колір. Однак було визначено кілька таких команд, щоб можна було використовувати їх з різними типами параметрів. Зокрема, частина 3 суфікса вказує, що для цієї команди задаються три параметри, інша версія команди Color використовує чотири параметри. Частина f суфікса вказує на те, що параметри даної команди представляють собою числа з плаваючою крапкою. Наявність різних форматів дозволяє бібліотеці OpenGL приймати дані користувача в його власному форматі даних.

Деякі команди бібліотеки OpenGL допускають використання 8 різних типів даних як своїх параметрів. Літери, які використовуються в якості суфіксів для того, щоб визначити ці типи даних для реалізації ISO З бібліотеки OpenGL, представлені в Таблиці; там же приведені відповідні визначення типів [4] у бібліотеці OpenGL.

Тип даних	Типовий відповідний тип даних мови з програмування	Визначення типів даних бібліотеки OpenGL
8-розрядне ціле	signed char	GLbyte
16-розрядне ціле	short	GLshort
32-розрядне ціле	Int або long	GLint, GLsizei
32-розрядне число з плаваючою точкою	float	GLfloat, GLclampf
64-розрядне число	double	GLdouble, GLclampd

з плаваючою точкою		
8-розрядне беззнаковое ціле	unsigned char	GLubyte, GLboolean
16-розрядне беззнаковое ціле	unsigned short	GLushort
32-розрядне беззнаковое ціле	unsignedint або unsigned long	GLuint, GLenum, GLbitfield

Таблиця 2.1 Суфікси найменувань команд і типи даних параметрів

3.3 Побудова візуалізації

Для візуалізації використаємо файл, створення якого описано у попередньому підрозділі. Потрібно зчитати дані з цього фалу в масив для подальшої побудови маятника:

```
 int step = 1;
ifstream in("text11.txt");
in >> Nt >> Nx;
Uxt = new double*[Nt+1];
 for (int i = 0; i < Nt; i ++)
 {
   Uxt[i] = new double[Nx+1];
      for (int j = 0; j < Nx; j ++)
   { in >> Uxt[i][j];
    }
 }
```

Далі поточково зобразимо маятник відповідно до зчитаного масиву:

```
for (int i = 0; i < Nx; i ++)
 {
```

```
    glBegin(GL_LINES);
glColor3f(0.0, 0.0, 0.0);
glVertex3f(0.95*Uxt[step][i], -6.0*i / Nx, 0.0);
glVertex3f(0.95*Uxt[step][i+1], -6.0*(i + 1) / Nx, 0.0);
    glEnd();
    glColor3f(0.5, 0.0, 1.0);
    glPointSize(10);
    glEnable(GL_POINT_SMOOTH);
    glBegin(GL_POINTS);
    glVertex3f(0.0, 0.0, 0);
    glVertex3f(0.95*Uxt[step][i + 1], -6.0*(i + 1) / Nx, 0.0);
    glEnd()
    }
```

Чисельні результати

На наступному зображенні подано вигляд матриці сформованої в ході виконання програми при n=16.

```
[-271.67  128.  0.  0.  0.  0.  0.  0.  0. 0.  0.  0.  0.  0.  0.  ]
[ 128.   -271.67  128.  0.  0.  0.  0.  0.  0.     0.  0.  0.  0.  0.  0.  ]
[   0.    128.   -271.67  128.  0.  0.  0.  0.  0.     0.  0.  0.  0.  0.  0.  ]
[   0.  0.    128.   -271.67  128.  0.  0.  0.  0.     0.  0.  0.  0.  0.  0.  ]
[   0.  0.  0.    128.   -271.67  128.  0.  0.  0.     0.  0.  0.  0.  0.  0.  ]
[   0.  0.  0.  0.    128.   -271.67  128.  0.  0.     0.  0.  0.  0.  0.  0.  ]
[   0.  0.  0.  0.  0.    128.   -271.67  128.  0.     0.  0.  0.  0.  0.  0.  ]
[   0.  0.  0.  0.  0.  0.    128.   -271.67  128.     0.  0.  0.  0.  0.  0.  ]
[   0.  0.  0.  0.  0.  0.  0.    128.   -271.67   128.  0.  0.  0.  0.  0.  ]
[   0.  0.  0.  0.  0.  0.  0.  0.    128.  -271.67  128.  0.  0.  0.  0.  ]
[   0.  0.  0.  0.  0.  0.  0.  0.  0.   128.   -271.67  128.  0.  0.  0.  ]
[   0.  0.  0.  0.  0.  0.  0.  0.  0.     0.    128.   -271.67  128.  0.  0.  ]
[   0.  0.  0.  0.  0.  0.  0.  0.  0.     0.  0.    128.   -271.67  128.  0.  ]
[   0.  0.  0.  0.  0.  0.  0.  0.  0.     0.  0.  0.    128.   -271.67  128.  ]
[   0.  0.  0.  0.  0.  0.  0.  0.  0.     0.  0.  0.  0.    128.   -271.67]
```

В результаті розв’язання СЛАР отримаємо результати при n=16:

[0.00000e-0, -5.04846e-4, -4.90261e-4, 9.99859e-4, 5.19289e-4, -4.75537e-4, 9.99435e-4, -5.33584e-4, 4.60679e-4, 9.98728e-4, -5.47729e-4, -4.4569e-4, 9.97739e-4, -5.61719e-4, -9.49868e-5, 7.08043e-4, -9.03692e-4]

n=64:

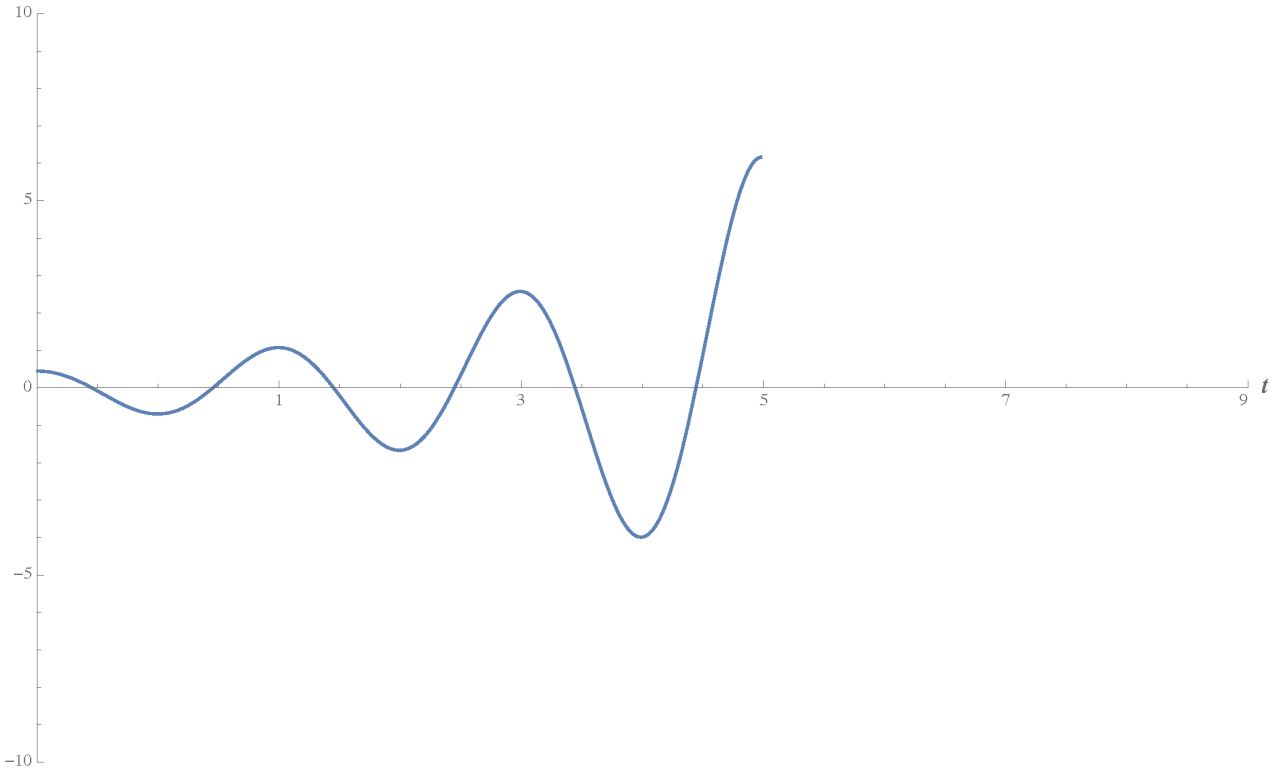
```
[[-4159.67  2048.       0.   ...     0.       0.       0.  ]
 [ 2048.   -4159.67  2048.   ...     0.       0.       0.  ]
 [    0.    2048.   -4159.67 ...     0.       0.       0.  ]
 ...
 [    0.       0.       0.   ... -4159.67  2048.       0.  ]
 [    0.       0.       0.   ...  2048.   -4159.67  2048.  ]
 [    0.       0.       0.   ...     0.    2048.   -4159.67]]
```

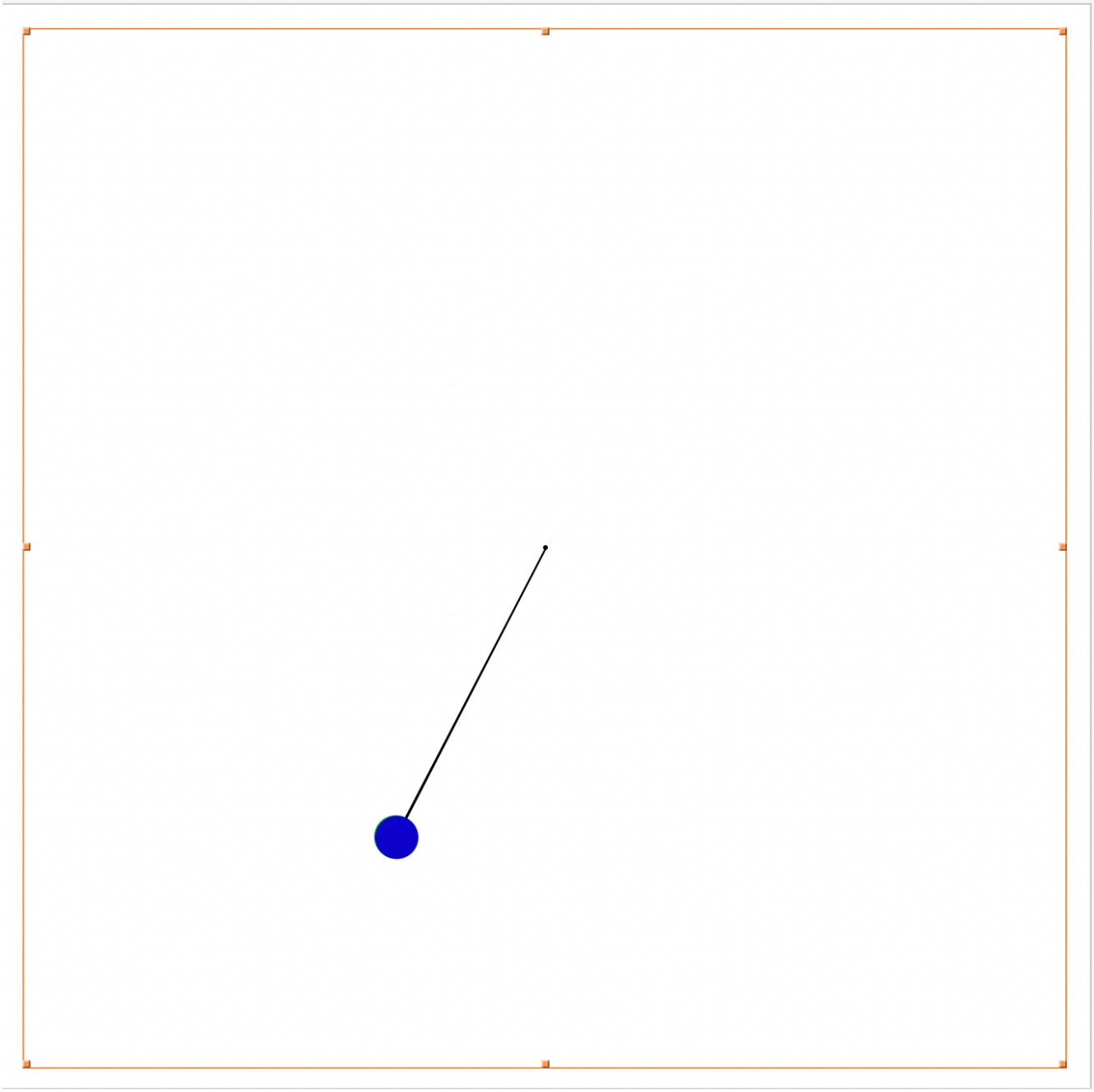
[0.00000e-0, 8.82333e-9, 5.57023e-9, 1.00626e-9, -3.79452e-9, -7.70231e-9, -9.91354e-9, -9.36457e-9, -6.61179e-9, -2.30303e-9, 2.54771e-9, 6.26934e-9, 9.19816e-9, 9.96234e-9, 8.38204e-9, 7.53902e-9, 3.55998e-9, -1.25685e-9, -5.7779e-9, -8.93921e-9, -9.99682e-9, -8.33589e-9, -4.75537e-9, -5.57426e-6, 4.657e-9, 8.27379e-9, 9.84487e-9, 9.5122e-9, 6.94099e-9, 2.73632e-9, 1.36737e-9, -3.45558e-9, -7.46531e-9, -9.7182e-9 , -9.68407e-9, -7.37094e-9, -2.65536e-9, 2.1944e-9, 6.52774e-9, 9.32488e-9, 9.92755e-9, 8.5748e-9, 5.14438e-9, 5.0331e-5, -4.25621e-9, -5.47729e-9, -7.10105e-9, -8.01409e-9, -8.77031e-9, -9.75646e-9, -9.98269e-9, -8.87525e-9, -1.11711e-9, 8.06409e-9, 9.14203e-9, -5.02478e-9, -9.62606e-9, 8.16239e-9, 9.96468e-9, 7.08043e-9, 8.66142e-5, -5.7555e-9, -9.67025e-9, -9.03692e-9]

Розгляд значення похибки з різним кроком розбиття:

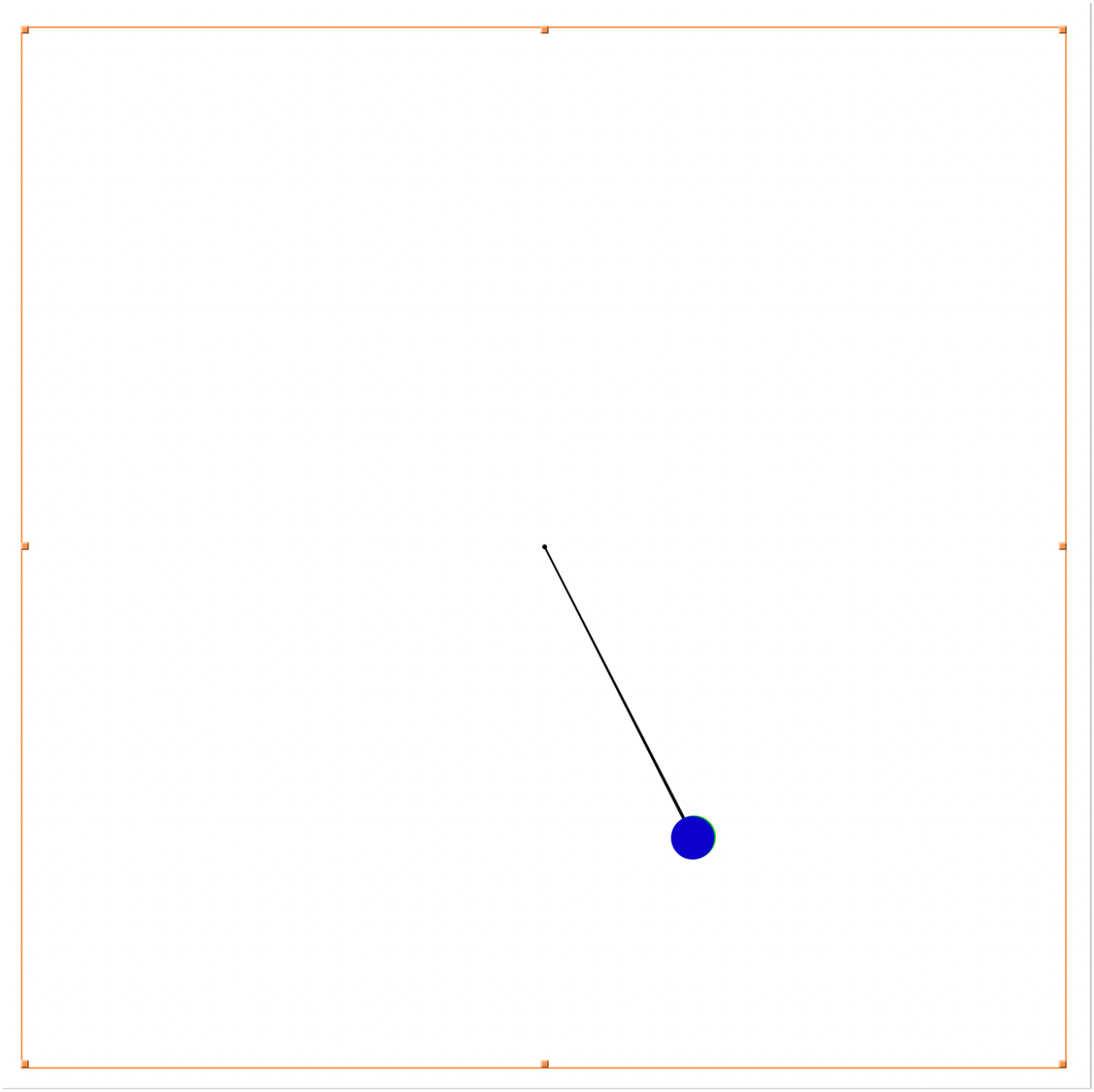
N	
4	0.458698
8	0.37867802
16	0.2567844
32	0.0788945
64	0.0156787
128	0.0025411

Чисельні експерименти показують, що зі збільшенням розбиття сітки, метод Ньютона-Канторовича очікувано дає кращі результати. Варто зауважити, що даний метод має хорошу збіжність навіть за використання невеликої сітки з кількістю кроків розбиття n=32, отримані результати є задовільними та похибка є невеликою. Легко бачити, що з подальшим збільшенням кількості розбиттів похибка буде зменшуватись.

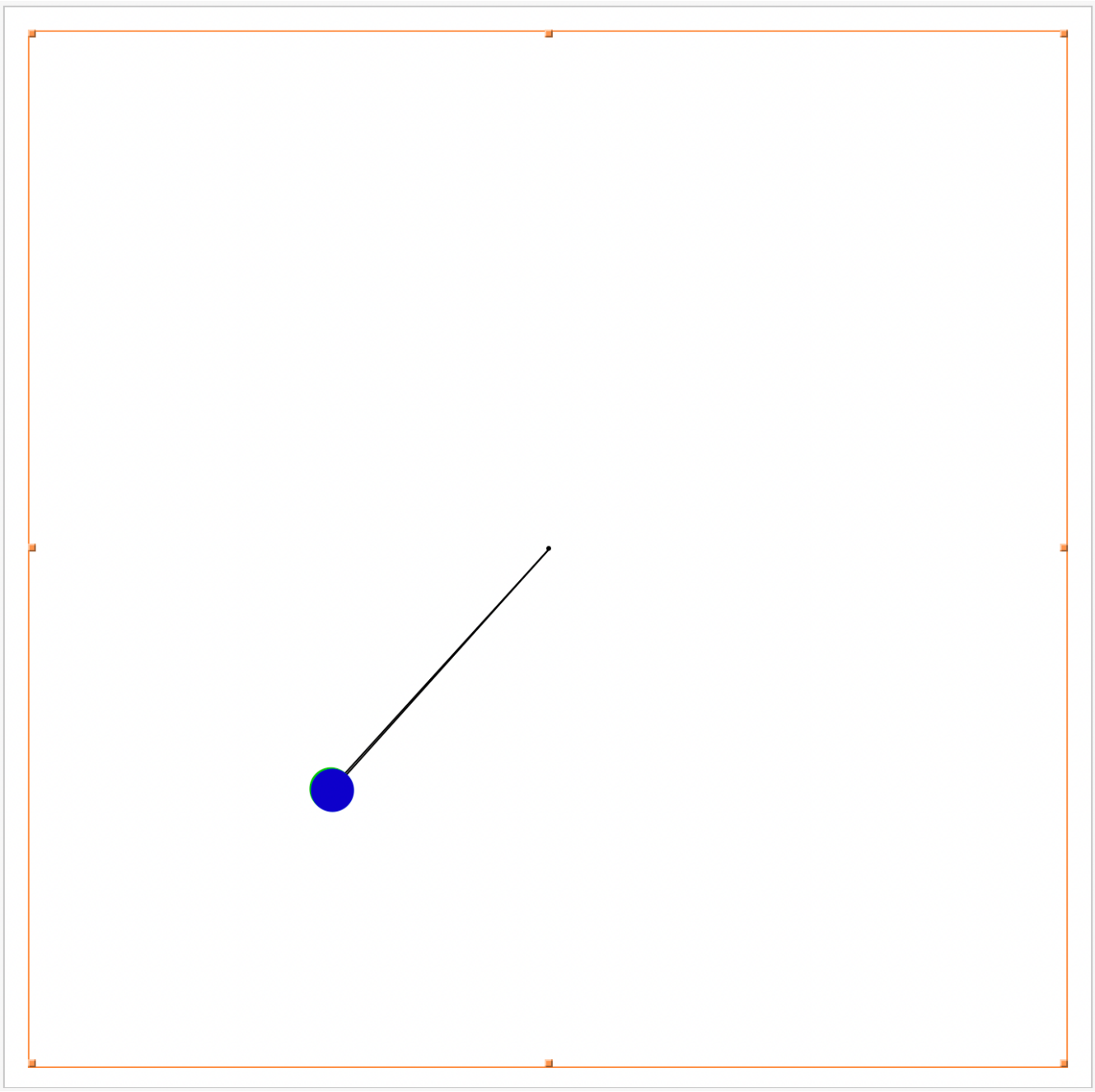

10
5
0
−5
−10
1
3
5
7
9
t



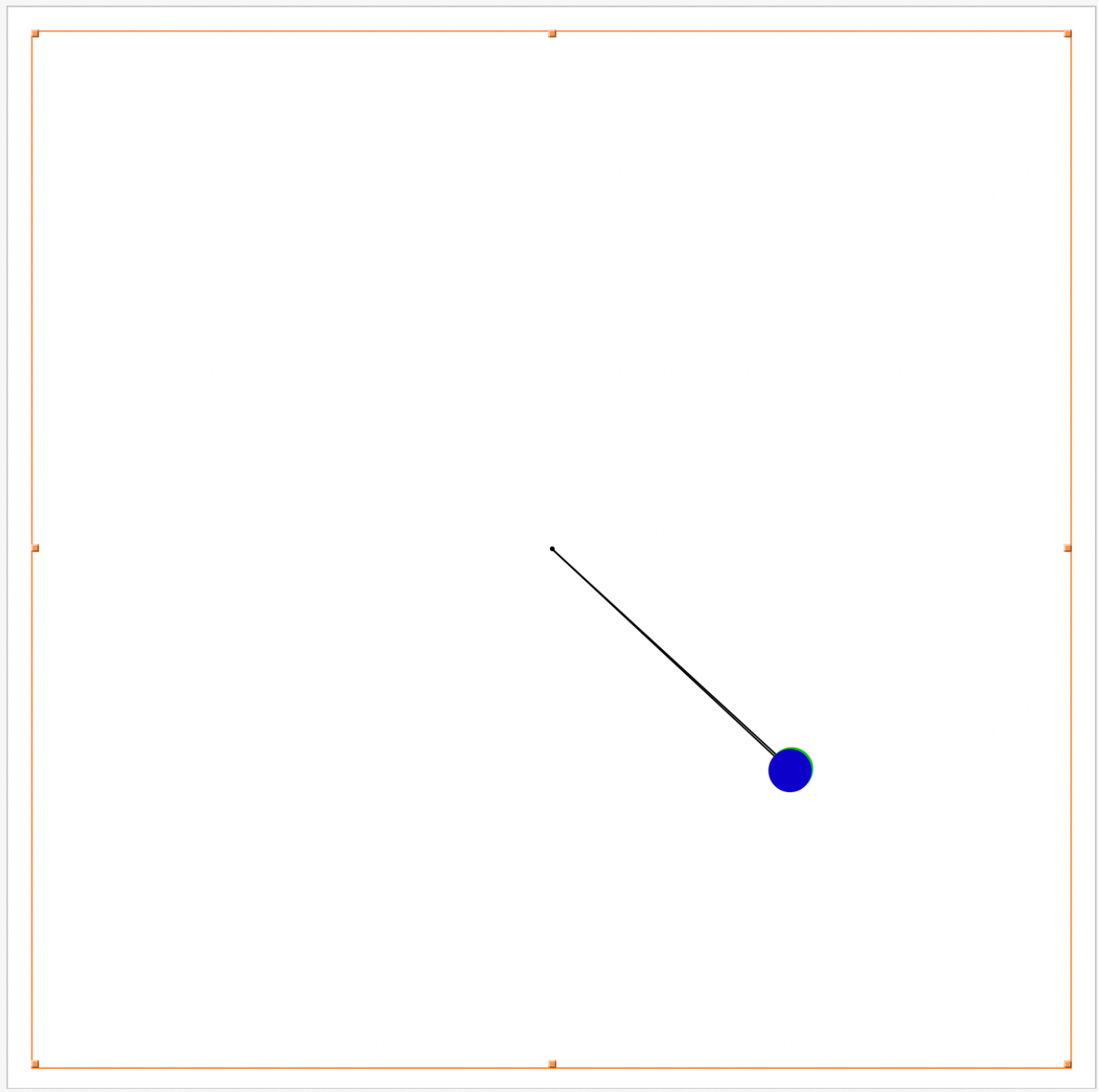
t=1.3



t=2.1



t=3.8



t=5

Висновок

В даній роботі розглянуто математичний маятник з розгойдувачем, що розгойдується з кутовою швидкістю Ω протягом часу t, після чого здійснює самостійні коливання. Розглянуто нелінійну математичну модель, яка описує коливання маятника та крайову задачу про його нелінійні коливання. Для розв'язання даної задачі, використано метод Ньютона-Канторовича та методом сіток, який використовується для задач такого типу. Скориставшись вищезгаданими методами було здійснено програмну реалізацію чисельного розв'язування задачі про нелінійні коливання математичного маятника.

Експерименти показали, що метод сіток дає наближені розв'язки з мінімальною похибкою, що забезпечує ефективність цього методу.

В кінці була досліджена збіжність методу Ньютона-Канторовича при різному розбитті сітки. Дослідження показало, що з густішим розбиттям сітки, метод має кращі результати і похибка зменшується.

З використанням OpenGL здійснено візуалізацію отриманих розв'язків задачі.

Список використаних джерел

1. Крылов, Н.М. Введение в нелинейную механику : (Приближенные и асимптотические методы нелинейной механики) / Акад. Н.М. Крылов и д-р Н.Н. Боголюбов. - Киев : Изд-во АН УССР, 1937. – 365 с.
2. А.А.Самарский ВВЕДЕНИЕ В ТЕОРИЮ РАЗНОСТНЫХ СХЕМ Главная редакция физико-математической литературы изд-ва «Наука», М., 1971.
3. Ву М., Девис Т., Недер Дж., Шранер Д. OpenGL. Руководство по программированию. Библиотека программиста. 4-е издание.-СПб.: Питер, 2006.-624 с.
4. Херн Дональд, Бекер М. Паулин Компьютерная графика и стандарт OpenGL. М.: Вильямс, 2005. — 1168 с., с ил. — ISBN 5-8459-0772-1
5. "Інтерактивна комп'ютерна графіка. Вступни курс на базі OpenGL", Едвард Енджел М.: Видавничи дім «Вільямс», 2001. - 592 с
6. Сборник задач по уравнениям математической физики. / Под ред. В. С. Владисирова. – 3-е изд., исправл. – М.: ФИЗМАТЛИТ, 2001. – 288с.
7. Смирнов М. М., Главная редакция физико-математической литературы изд-ва «Наука», 1975.
8. Емельянов В. М., Рыбакина Е. А. Уравнения математической физики. Практикум по решению задач: Учебное пособие. — 22е изд., стер. — СПб.: Издательство «Лань», 2016. — 216 с.
9. Гаврилюк І.П., Макаров В.Л. Методи обчислень : Підручник : У 2.ч.—К.: Вища шк., 1995.—Ч. 2.—431 с.