МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**«Челябинский государственный университет»**

**(ФГБОУ ВО «ЧелГУ»)**

Математический факультет

Кафедра вычислительной математики

**КУРСОВАЯ РАБОТА**

О свойствах числа золотого сечения

Выполнил студент Русин Владислав Сергеевич

Группы МП-203

очной формы обучения

направления подготовки (специальности)

01.03.02 Прикладная математика и информатика \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022г.

|  |  |
| --- | --- |
|  | Научный руководитель  Овчинников Михаил Алексеевич  Должность доцент  Ученая степень канд. физ.-мат. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2022г. |

Челябинск

2022

Содержание

[1. Число золотого сечения 4](#_Toc74082996)

[1.1. Некоторые свойства числа золотого сечения 5](#_Toc74082997)

[1.2. Арифметика чисел, порождаемых числом золотого сечения 6](#_Toc74082998)

[2. Задача о матрицах 10](#_Toc74083002)

[2.1. Условия задачи 11](#_Toc74083003)

[2.2. Программа, реализующая решение поставленной задачи 14](#_Toc74083004)

[2.2.1. Внешний вид вывода программы 19](#_Toc74083009)

[2.2.2. Частичный разбор кода программы 19](#_Toc74083009)

[2.3. Анализ полученных результатов 15](#_Toc74083005)

[2.4. Интерпретация полученных результатов 17](#_Toc74083006)

[3. Игра с природой 18](#_Toc74083007)

[3.1. Классические критерии решения задачи 19](#_Toc74083008)

[3.1.1. Критерий Байеса-Лапласа 19](#_Toc74083009)

[3.1.2. Критерий Лапласа 20](#_Toc74083010)

[3.1.3. Минимаксный критерий (ММ-критерий) 21](#_Toc74083011)

[3.1.4. Критерий Вальда 22](#_Toc74083012)

[3.1.5. Критерий Сэвиджа 23](#_Toc74083013)

[3.2. Производный критерий BL(MM) 24](#_Toc74083014)

[3.3. Программа, реализующая производный критерий BL(MM) 26](#_Toc74083015)

[3.4. Сравнение решений для различных критериев, интерпретация полученных результатов 26](#_Toc74083016)

[Список литературы 28](#_Toc74083017)

[Приложение 29](#_Toc74083018)

1. **Число золотого сечения**

Золотое сечение – это такое соотношение двух величин, при котором отношение большей величины к меньшей равно отношению суммы этих величин к большей, т. е. . Принимая за неизвестное , получим уравнение или

(1)

Числом золотого сечения называется иррациональное число , являющееся положительным корнем уравнения (1), приблизительно равное 1,618.

* 1. **Некоторые свойства числа золотого сечения**
* Из квадратного уравнения (1) следует, что или

(2)

* Число представимо в виде цепной дроби:

(3)

Это единственная цепная дробь, содержащая только единицы. Она получается записью квадратного уравнения (1) в виде  с последующей заменой в правой части уравнения на его значение . Таким образом, получим . Продолжая применять к правой части уравнения замену , получим бесконечную цепную дробь (3).

* Число представимо в виде бесконечной цепочки квадратных уравнений:

(4)

Эта запись получается при поиске квадратного корня уравнения (1) с последующей заменой в правой части уравнения на его значение (т. е. аналогично способу, используемому для представления в виде бесконечной цепной дроби (3)):

* 1. **Арифметика чисел, порождаемых числом золотого сечения**

Самой известной последовательностью чисел, порождаемых числом , являются числа Фибоначчи (обозначим эту последовательность как ), названные так в честь математика XII–XIII веков Леонардо Пизанского (известного как Фибоначчи). Числа в последовательности образуются по следующему правилу: пусть , и каждое последующее число равно сумме двух предшествующих ему чисел, т. е. , где   
.

Первые члены последовательности Фибоначчи: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

Интересно вот что: отношение любого члена последовательности к члену является приближённым значением числа , причём тем приближение будет точнее, чем больше будет n. Так, , ,   
, , … Таким образом, .

1. **Задача о матрицах**

Рассматриваемая в данной работе задача состоит в том, чтобы выяснить, по какому принципу устроено множество матриц, получаемых в результате перемножений исходных матриц, которые были представлены в качестве примера ТКТП (топологическая квантовая теория поля) британским математиком Майклом Атья.

* 1. **Условия задачи**

Дана группа из 20 пар матриц порядка 3 (будем называть эти пары «базовыми»), разбитых на две подгруппы: пары, пронумерованные как , – , , и пары, пронумерованные , – , . Также дана формула, по которой эти пары перемножаются:

(5)

здесь и представляют пары с индексом , и представляют пары с индексом , где . Из свойства (2) числа золотого сечения следует:

(6)

В результате перемножения пар по этой формуле получается новая пара матриц, ( задаёт первую матрицу, – вторую.

Примечательно то, что при перемножении по формуле двух пар одной подгруппы с индексами результатом будет являться пара матриц той же подгруппы с индексом ( – взятие остатка от деления). Например, при перемножении по формуле (5) пар , и , получим пару , , а при перемножении пар , и , получим , .

Отметим, что матрицы в парах с индексом 0 являются по сути единичными матрицами, и результатом умножения на них пары с индексом будет та же пара с индексом , поэтому для дальнейших вычислений пары с индексом 0 применяться не будут.

Обозначим результат перемножения по формуле (5) пар , и , как  
. Назовём это «цепочками» длины 2. Тогда результатом перемножения и , будет . Это цепочки длины 3. Таким образом, будем составлять цепочки длины n, причём элементами цепочки на нечётных позициях будут базовые пары , , а на чётных - базовые пары , .

Вопрос задачи: как устроено множество пар матриц, получаемых при составлении цепочек длины n?

* 1. **Программа, реализующая решение поставленной задачи**

Нужно сказать, что в данной работе используются не совсем те базовые пары матриц, которые были представлены Майклом Атья. Дело в том, что "настоящие" матрицы содержат в себе элементы - полуцелые числа, из-за чего результаты вычислений получаются более громоздкими. Чтобы решить эту проблему, в работе используются удвоенные "настоящие" матрицы, содержащие в себе только целые числа. Однако в результате перемножений по формуле (5) такие удвоенные "настоящие" удваиваются снова, становясь уже учетверёнными, поэтому все элементы матриц в парах, получаемых по формуле, должны быть поделены на 2, чтобы они снова стали удвоенными.

**2.2.1. Внешний вид вывода программы**

Результатом работы программы являются два текстовых файла. В одном содержатся пары матриц, полученные по формуле, в виде:

*Название пары :*

*{*

*[Строка 1 матрицы 1] [Строка 1 матрицы 2]*

*[Строка 2 матрицы 1] [Строка 2 матрицы 2]*

*[Строка 3 матрицы 1] [Строка 3 матрицы 2]*

*};*

Элементы в строках разделяются запятыми (название файла: matrices.txt). В другом файле – записи об одинаковых парах в виде:

*Название пары 1 - копия Название пары 2*

(название файла: knownDuplicates.txt).

Программа вычисляет пары матриц и ищет среди них совпадающие во всех цепочках длины 2–5, т. е. всего в файл matrices.txt будет занесено   
 записей о парах матриц.

**2.2.2. Частичный разбор кода программы**

В программе постоянно повторяются следующие три шага:

1. вычисление очередной пары матриц по формуле (5);
2. запись полученной пары в файл matrices.txt;
3. поиск среди предыдущих вычисленных пар такой, которая будет одинаковой с последней. При нахождении одной такой пары поиск прекращается, а в файл knownDuplicates.txt заносится запись о найденном совпадении.

Для реализации этих шагов в программе есть три соответствующие функции:

1. calculate\_by\_formula;
2. print\_to\_file;
3. search\_for\_duplicate.

Рассмотрим эти функции подробнее, начиная с calculate\_by\_formula:

* Возвращаемый тип: ***void***
* Список параметров: \*a, \*b, \*c, \*d (указатели типа ***const*** vector<vector<**int**>>\*), \*result1, \*result2 (указатели типа vector<vector<**int**>>\*)

Тип vector<vector<**int**>> представляет матрицу. \*a, \*b, \*c, \*d – указатели на матрицы, \*a, \*b – первая пара, \*c, \*d – вторая пара.

Функция представляет собой реализацию формулы (6). Сначала сохраняются результаты обычного матричного перемножения нужных матриц:

vector<vector<int>> ac = multiply(a, c);

vector<vector<int>> ad = multiply(a, d);

vector<vector<int>> bc = multiply(b, c);

vector<vector<int>> bd = multiply(b, d);

Затем составляются необходимые суммы матриц, которые записываются в \*result1 и \*result2:

\*result1 = add(&ac, &ad, &bc);

\*result2 = add(&ac, &bd);

Как было замечено ранее, по формуле получаются удвоенные матрицы, поэтому не забываем делить результаты на 2:

divide\_by\_2(result1);

divide\_by\_2(result2);

Полученная в результате вычислений пара матриц записана в   
\*result1, \*result2, причём т. к. это указатели, то результат вычислений доступен по адресам, на которые эти указатели указывают, после завершения работы функции, что позволяет передать получившиеся матрицы назад, в функцию, из которой вызывалась calculate\_by\_formula, и оттуда вызвать следующую функцию - print\_to\_file:

* Возвращаемый тип: ***void***
* Список параметров: fstream \*matrices, string pairHeader, \*res1, \*res2 (указатели типа **const** vector<vector<**int**>>\*)

\*matrices - ссылка на поток для чтения/редактирования файла matrices.txt; pairHeader - название, которым будет обозначена пара, записываемая в файл; \*res1, \*res2 - указатели на пару матриц, которую необходимо записать в файл.

Прежде всего, конец файла устанавливается как позиция, с которой начинается запись в файл:

(\*matrices).clear();

(\*matrices).seekp(0, ios::end);

Затем в файл заносится запись, оформленная так, как было описано в   
пункте **2.2.1.**:

*(\*matrices) << multiplicationName << "****\n****{" << endl;*

***for*** *(****int*** *i = 0; i < N; i++)*

*{*

*(\*matrices) << "****\t****[ ";*

***for*** *(****int*** *j = 0; j < N; j++)*

*(\*matrices) << setw(numWidth) << (\*res1)[i][j] << (j < 2 ? ", " : " ]****\t****");*

*(\*matrices) << "****\t****[ ";*

***for*** *(****int*** *j = 0; j < N; j++)*

*(\*matrices) << setw(numWidth) << (\*res2)[i][j] << (j < 2 ? ", " : " ]****\n****");*

*}*

*(\*matrices) << "};" << endl << endl;*

Это конец функции print\_to\_file. Управление далее передаётся в функцию search\_for\_duplicate:

* Возвращаемый тип: **void**
* Список параметров: *string pairHeader, fstream \*matrices, ofstream \*knownDuplicates,* \*res1, \*res2 (указатели типа **const** vector<vector<**int**>>\*)

*\*knownDuplicates* – ссылка на поток для записи в файл knownDuplicates.txt; \*res1, \*res2, pairHeader и \*matrices те же, что и для функции print\_to\_file.

Сначала создаём переменные, в которые будем впоследствии записывать название и матрицы каждой пары, с которой будет сравниваться пара \*res1, \*res2:

*string comparedPairHeader;*

*vector<vector<****int****>> comparedM1(N, vector<****int****>(N));*

*vector<vector<****int****>> comparedM2(N, vector<****int****>(N));*

Создаём переменную типа ***bool***, в которую будет записываться результат сравнения с каждой отдельной парой:

***bool*** *isDuplicate;*

Устанавливаем начало файла matrices.txt в качестве точки, с которой начнётся считывание пар:

*(\*matrices).clear();*

*(\*matrices).seekg(0, ios::beg);*

В цикле ***while*** считываются пары матриц и проводится сравнение. Записываем название каждой пары в переменную comparedPairHeader. Поиск совпадений прекратится, если в comparedPairHeader будет записано название пары, для которой совпадения ищутся:

***while*** *(getline((\*matrices), comparedPairHeader) && comparedPairHeader != pairHeader)*

Очередной шаг цикла начинаем с установки значения isDuplicate (изначально считаем, что рассматриваемая на данном шаге цикла пара является идентичной паре, для которой таковые ищутся) и считывания пары из файла matrices.txt:

isDuplicate = true;

read\_pair(matrices, &comparedM1, &comparedM2);

Сравниваем пары и, если хотя бы один элемент с индексом i, j хотя бы одной матрицы из пары отличается от элемента с теми же индексами соответствующей матрицы другой пары, прекращаем сопоставление этих пар и переходим к следующему шагу цикла **while**:

***for*** *(****int*** *i = 0; i < N; i++)*

*{*

***for*** *(****int*** *j = 0; j < N; j++)*

*{*

***if*** *((\*res1)[i][j] != comparedM1[i][j] ||*

*(\*res2)[i][j] != comparedM2[i][j])*

*{*

*isDuplicate = false;*

***break****;*

*}*

*}*

***if*** *(!isDuplicate)*

***break****;*

*}*

Если же пара, рассматриваемая на текущем шаге цикла **while**, оказывается такой же, как пара с названием pairHeader, то в файле knownDuplicates.txt делается пометка об обнаруженном совпадении, и поиск прекращается:

***if*** *(isDuplicate)*

*{*

*(\*knownDuplicates) << pairHeader.substr(0, pairHeader.length() - 2) << " - копия "*

*<< comparedPairHeader.substr(0, comparedPairHeader.length() - 2) << endl*

*<< endl;*

***break****;*

*}*

На этом заканчивается работа функции search\_for\_duplicate. Дальше программа либо начнёт выполнять все три шага заново, либо, если последняя обработанная пара матриц называлась , завершится.

* 1. **Анализ полученных результатов**

После завершения работы программы файл knownDuplicates.txt содержит 43500 записей о совпадениях, т. е. из 66420 пар матриц 43500 являются копиями оставшихся 22920.