**Міністерство освіти і науки України**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ІВАНА ФРАНКА**

Кафедра теорії

оптимальних процесів

**Курсова робота**

**“**.NET–бібліотека математичних застосунків AppliedMathLibrary**”**

**Виконав:**

студент 5 курсу групи ПМА–11м,

спеціальності 124 – “системний аналіз”

Байцар Р. М.

**Керівник**:

к. т. н., доцент кафедри теорії оптимальних процесів, Мельничин А. В.

Національна шкала\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів:\_\_\_ Оцінка ECTS\_\_\_

Львів – 2022

**Зміст**

[**Вступ** 3](#_Toc104582242)

[**Розділ I. аналіз предметної області та постановка задачі** 4](#_Toc104582243)

[***1.1.*** ***Постановка задачі*** 4](#_Toc104582244)

[***1.2.*** ***Вибір платформи .NET*** 5](#_Toc104582245)

[***1.3.*** ***Структура бібліотеки*** 5](#_Toc104582246)

[***1.4.*** ***Основні визначення та використані технології*** 6](#_Toc104582247)

[**Розділ II. Технічна документація** 9](#_Toc104582248)

[***2.1.*** ***Програмний об’єкт Матриця*** 9](#_Toc104582249)

[***2.2.*** ***Програмний об’єкт Вектор*** 11](#_Toc104582250)

[***2.3.*** ***Програмний об’єкт Точка*** 13](#_Toc104582251)

[***2.4.*** ***Метод Гауса*** 14](#_Toc104582252)

[***2.5.*** ***Методи Парето та Слейтера*** 15](#_Toc104582253)

[***2.6.*** ***Статистичні методи*** 16](#_Toc104582254)

[**Розділ III. Приклад Застосування** 19](#_Toc104582255)

[***3.1.*** ***Інструкція з встановлення бібліотеки*** 19](#_Toc104582256)

[***3.2.*** ***Тестова програма*** 20](#_Toc104582257)

[**Розділ IV. тестування** 21](#_Toc104582258)

[***4.1.*** ***Модульні тести*** 21](#_Toc104582259)

[***4.2.*** ***Тест продуктивності*** 21](#_Toc104582260)

[**Висновки** 23](#_Toc104582261)

[**Список використаних джерел** 24](#_Toc104582262)

# **Вступ**

У зв’язку з великою популярністю .NET платформи, сфера її застосування зростає, зростає також й потреба в готових програмних рішеннях, в тому числі – бібліотек.

Дана бібліотека є ідейним аналогом популярної Python бібліотеки “NumPy”. Потреба в появі бібліотеки виникла з відсутності схожого рішення для мови C#.

У період здобування ступеня бакалавра, мені довелося виконувати безліч лабораторних робіт з яких я запозичав цілі блоки коду. Вирішенням проблеми копіювання і є дана бібліотека.

# **Розділ I. аналіз предметної області та постановка задачі**

## ***Постановка задачі***

Усі реалізовані методи та об’єкти у даній роботі базуються на моєму досвіді набутому у процесі здобуття ступеня бакалавра «системного аналізу», факультету прикладної математики (applied math).

Від назви факультету і походить назва бібліотеки. Оскільки термінологія, найменування та реалізація певних методів породжена термінологією, використовуваною викладачами мого університету, підкреслити це у назві я вважаю доречним.

Необхідно розробити бібліотеку, що реалізовує основні методи та об’єкти, що найбільш поширені при розв’язуванні більш складних та нетривіальних задач.

Вирішити проблему копіювання коду та надати безліч готовий рішень «з коробки».

Забезпечити високу надійність та продуктивність коду, в той же час простоту та зрозумілість.

Найбільш поширеними та часто використовуваними об’єктами є:

* Вектор
* Точка
* Матриця

Ці прости математичні абстракції формалізують абсолютну більшість проблем, що вирішує прикладна математика.

Оскільки дані об’єкти вміють взаємодіяти з іншими об’єктами та володіють певними властивостями, ці властивості потрібно формалізувати в коді. Додавання матриць породжує нову матрицю, множення на вектор також породжує матрицю. Всю базову алгебраїчну взаємодію потрібно передбачити.

Також для розв’язування окремих, простих, але поширених задач необхідно реалізувати наступні методи:

* Метод Гауса
* Методи Парето та Слейтера
* Деякі статистичні методи (наприклад середнє значення, медіана, мода, мат. очікування, дисперсія, тощо)

Передбачити масштабованість бібліотеки та здатність до впровадження нових методів. Реалізувати систему версій, релізів, забезпечити ведення журналу змін.

## ***Вибір платформи .NET***

Оскільки .NET для мене це основна платформа написання програмного забезпечення, а мої знання в інших середовищах розробки обмежені, питання вибору навіть не стояло.

Окрім того, на просторах NuGet важко знайти схожу бібліотеку, що розв’язує поставлений список проблем.

## ***Структура бібліотеки***

Усі методи та об’єкти розділено по окремих директоріях, а сама структура має ієрархічну, деревовидну структуру. Наступний малюнок відображає будову бібліотеки

A screenshot of a computer screen

Description automatically generated with medium confidence

Рис. 1.1. Структура бібліотеки AppliedMathLibrary

Варто зазначити, що дана структура відповідає поточній проблематиці бібліотеки, тому по мірі розширення функціоналу дана структура може змінюватися, деякі методи можуть замінувати свої детективи, об’єднуватися з іншими, або розділятися на окремі під модулі. Це створює деякі труднощі в зворотній сумісності, але є необхідною жертвою на благо покращення кінцевого результату.

## ***Основні визначення та використані технології***

При написанні програмної бібліотеки доведеться познайомитися з новими термінами, які я не використовував у своїх попередніх курсових роботах.

*Бібліотека* *(library)* – збірка об'єктів чи підпрограм для вирішення близьких за тематикою задач. У комп'ютерних науках бібліотеки розділяють на статичні та динамічні.

У випадку моєї роботи мова йде про динамічну бібліотеку.

*Динамічні бібліотеки* (або розподілена бібліотека (shared library), або бібліотеками, що динамічно підключаються (Dynamic Link Library, DLL). Це окремі файли, що надають програмі набір використовуваних функцій для завантажування на етапі виконання при зверненні програми до ОС із заявкою на виконання функції з бібліотеки. Якщо необхідна бібліотека вже завантажена в оперативну пам'ять, програма використовуватиме завантажену копію бібліотеки. Такий підхід дозволяє зекономити час і пам'ять, оскільки декілька програм використовують одну копію бібліотеки, вже завантажену в пам'ять.

*NuGet* – система керування пакунками, розроблена компанією Microsoft. Створений у 2010–му, NuGet фактично є єдиним пакетним менеджером .NET світу, популярність якого не уступає пакетним менеджерам JavaScript та Python. NuGet повністю інтегрований у Visual Studio, де має простий на зручний графічний інтерфейс.

*.NET* – програмна технологія, запропонована фірмою Microsoft як платформа для створення програмних рішень будь якої складності.

*Статичні методи (або властивості)* – це методи, виклик яких здійснюється від імені класу в якому вони знаходяться. Таким чином метод абстрагується від екземпляру класу та дозволяє працювати з будь якими вхідними параметрами. Вважається потокобезпечним (thread–safe).

*Інкапсуляція –* В парадигмі ООП програмування, означає приховування від кінцевого користувача деталей реалізації і неможливість впливати на внутрішні процеси інкапсульованого об’єкту.

*Модульне тестування (Unit testing)* – це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестованою. В об'єктно–орієнтованому програмуванні ця найменша частина — метод. Unit–тести розробляються разом з частиною коду яку вони зобов’язані тестувати. Зазвичай unit–тести необхідні для того, щоб упевнитися, що код відповідає вимогам архітектури та має очікувану поведінку.

*Тест продуктивності (benchmark)* – як і unit–тест має на меті перевірити програму на швидкодію на об’єм використаної пам’яті. Це дозволяє виокремити кращі та гірші методи для вирішення певної задачі, або знайти області аргументів коли той чи інший метод є доцільнішим для використання. Кращими є ті методи що споживають менше пам’яті та працюють швидше.

*Git* – система контролю версій що дозволяє фіксувати кожен крок написання програми та зберігати його на віддаленому репозиторію.

*Microsoft Visual Studio 2022* – потужне IDE для розробки програмних продуктів на базі .NET. Версія 2022 є необхідною для використання .NET 6.

*.NET* — це модульна платформа для розробки програмного забезпечення різного типу з відкритим вихідним кодом. Варто зауважити, що у версії 6 порушена зворотна сумісність з попередніми версіями.

*BenchmarkDotNet* – бібліотека з відкритим програмним кодом для написання тестів продуктивності.

*xUnit* – популярний в середовищі .NET фреймворк для написання unit–тестів. Надає широкий спектр можливостей у тестуванні.

*FluentAssertions* – бібліотека – розширення xUnit для написання тестів сценарії яких більше походять на людську мову (в даному випадку англійську). Приклад тесту з перевіркою результату виконання методами FluentAssertions: “actualResult.Should().BeEquivalentTo(expectedResult);”.

# **Розділ II. Технічна документація**

## ***2.1. Програмний об’єкт Матриця***

Матриця – це сукупність математичних величин, певним способом розміщених у прямокутній таблиці.

Бібліотека реалізовує довільну матрицю розміром n x m, елементами якої є дійсні числа (тип double). Очевидно, що n та m повинні бути більшими за 0.

Для створення об’єкту Matrix реалізовано наступні конструктори:

Matrix(int n, int m) – створює матрицю n x m де всі елементи 0;

Matrix(int n, int m, params double[] values) – створює матрицю n x m де всі елементи передані через параметр values. Очікується що кількість елементів values буде рівна n\*m;

Matrix(int n, int m, IEnumerable<double> values) – аналогічно з попереднім конструктором;

Matrix(int n) – створює матрицю n x n де всі елементи 0;

Matrix(int n, params double[] values) – створює матрицю n x n де всі елементи передані через параметр values. Очікується що кількість елементів values буде рівна n\*n;

Matrix(int n, IEnumerable<double> values) – аналогічно з попереднім конструктором;

Matrix(Matrix matrix) – створює нову матрицю на основі переданої (конструктор копіювання);

Matrix(params Vector[] vectors) – створює матрицю з векторів де кожен вектор стає рядком матриці. Очікується, що всі вектори однакової розмірності та передано принаймні один вектор;

Властивості об’єкту Matrix:

bool IsSquare – повертає true якщо матриця квадратна;

int Rows – повертає кількість рядків матриці (число n);

int Columns – повертає кількість стовпців матриці (число m);

double this[int i, int j] – індексатор. Дозволяє звертатися до кожного елементу матриці як до двовимірного масиву;

Методи об’єкту Matrix:

static Matrix Transpose(Matrix matrix) – створює нову транспоновану матрицю на основі переданої. Статичний метод;

Matrix Transpose() – транспонує матрицю;

List<Vector> ToVectors() – розкладає матрицю на масив векторів, де кожен вектор це рядок матриці;

Matrix Subtract(Matrix matrix) – віднімає від матриці передану. Результатом операції є нова матриці. Очікується, що передана матриця схожої розмірності;

static Matrix Subtract(Matrix A, Matrix B) – аналогічно до попереднього методу повертає різницю двох матриць. Статичний метод;

static Matrix operator –(Matrix A, Matrix B) – аналогічно до попередніх двох методів повертає різницю двох матриць. Статичний метод. Дозволяє використовувати оператор “–” (наприклад A – B);

Vector MultiplyBy(Vector vector) – множення матриці на вектор справа. Результатом є новий вектор. Очікується, що розмірність вектора рівна кількості стовпців матриці;

static Vector Multiply(Matrix matrix, Vector vector) – аналогічно до попереднього, множення матриці на вектор справа. Статичний метод;

static Vector operator \*(Matrix matrix, Vector vector) – аналогічно до попередніх двох методів, множення матриці на вектор справа. Статичний метод. Дозволяє використовувати оператор “\*” (наприклад M \* v);

IEnumerator<double> GetEnumerator() – дозволяє використовувати матрицю в методі foreach;

## ***Програмний об’єкт Вектор***

Вектор — математичний об'єкт, який характеризується величиною і напрямком. Найбільш поширена математична абстракція.

Бібліотека реалізовує довільний вектор розміром n , елементи якого є дійсні числа (тип double). Очевидно, що n повинно бути більшим за 0.

Для створення об’єкту Vector реалізовано наступні конструктори:

Vector(int n) – створює вектор розміром n, де всі елементи 0;

Vector(params double[] values) – створює вектор на основі переданих значень параметру values. Розмірність n обчислюється з кількості переданих значень;

Vector(Vector vector) – створює копію переданого вектора

(конструктор копіювання);

Vector(Point point) – створює вектор на основі точки. Точка від веткора відрізняється лише методами (поведінкою), тому це фактично конструктор копіювання;

Властивості об’єкту Vector:

int Dimension – повертає розмірність вектора;

double this[int i] – індексатор. Дозволяє звертатися до кожного елементу вектора як до масиву;

Методи об’єкту Vector:

bool CompareByValueWith(Vector vector) – порівнює два вектори за значеннями. Якщо значення відповідних координат векторів рівні то вектори рівні. Очікує вектори однакових розмірностей.

static bool CompareByValue(Vector vector1, Vector vector2) – аналогічно до попереднього методу. Статичний метод;

static bool Comparable(Vector vector1, Vector vector2) – визначає чи два вектори можна порівнювати. Статичний метод;

bool ComparableWith(Vector vector) – аналогічно до попереднього методу, визначає чи можна порівнювати вектор з переданим;

Vector Subtract(Vector vector) – віднімає від вектора переданий вектор. Очікує вектор тієї самої розмірності. Повертає новий вектор;

static Vector Subtract(Vector vector1, Vector vector2) – аналогічно до попереднього методу, повертає різницю двох векторів. Статичний метод;

static Vector operator –(Vector vector1, Vector vector2) – аналогічно до попереднього методу, повертає різницю двох векторів. Статичний метод. Дозволяє використовувати оператор “–” (наприклад v1 – v2);

static Vector Sum(Vector vector1, Vector vector2) – повертає новий вектор, що є результатом додавання двох переданих векторів. Очікує, що вектори однакової розмірності. Статичний метод;

Vector Add(Vector vector) – аналогічно до попереднього методу, повертає суму двох векторів.

static Vector operator +(Vector vector1, Vector vector2) – аналогічно до попередніх двох методів, повертає суму двох векторів. Статичний метод. Дозволяє використовувати оператор “+” (наприклад v1 + v2);

Vector DivideBy(double scalar) – дозволяє розділити всі елементи вектора на скаляр. Повертає новий вектор. Скаляр не може бути рівним 0;

static Vector Divide(Vector vector, double scalar) – аналогічно до попереднього методу. Статичний метод;

static Vector operator /(Vector vector, double scalar) – аналогічно до попередніх двох методів. Статичний метод. Дозволяє використовувати оператор “/” (наприклад v / a);

Vector MultiplyBy(double scalar) – домножає скаляр до вектора. Повертає новий вектор.

static Vector Multiply(Vector vector, double scalar) – аналогічно до попереднього методу. Статичний метод;

static Vector operator \*(Vector vector, double scalar) – аналогічно до попередніх двох методів. Статичний метод. Дозволяє використовувати оператор “\*” (наприклад v \* a);

double Norm() – обчислює евклідову норму даного вектора.

static double Norm(Vector vector) – обчислює евклідову норму переданого вектора. Статичний метод;

double DistanceTo(Vector vector) – обчислює евклідову відстань до переданого вектора. Очікує, що вектори однакової розмірності;

static double DistanceBetween(Vector vector1, Vector vector2) – аналогічно до попереднього методу обчислює відстань між векторами. Статичний метод;

IEnumerator<double> GetEnumerator() – дозволяє використовувати вектор в методі foreach;

## ***Програмний об’єкт Точка***

Точка позначає точне положення в просторі. Це матеріальний об'єкт, що позначає місце або стан.

В даній бібліотеці точка схожа на вектор, але має значно менший функціонал.

Конструктори об’єкту Point:

Point(int n) – створює точку розміром n, де всі елементи 0;

Point(params double[] values) – створює точку на основі переданих значень параметру values. Розмірність n обчислюється з кількості переданих значень;

Point(Point point) – створює копію переданої точки (конструктор копіювання);

Point(Vector vector) – створює точку на основі переданого вектора. Точка від веткора відрізняється лише методами (поведінкою), тому це фактично конструктор копіювання;

Властивості об’єкту Point:

int Dimension – повертає розмірність точки;

double this[int i] – індексатор. Дозволяє звертатися до кожного елементу точки як до масиву;

Методи об’єкту Point:

IEnumerator<double> GetEnumerator() – дозволяє використовувати точку в методі foreach;

## ***Метод Гауса***

Метод Гауса — класичний метод для розв’язування систем лінійних алгебричних рівнянь.

В даній бібліотеці метод представлений статичним класом GaussMethod, що повністю інкапсулює логіку розв’язування СЛАР.

Розв’язування відбувається зведенням вхідної матриці до верхньотрикутної, що дозволяє також знайти визначник вхідної матриці як побічний ефект.

Розв’язок відбувається викликом методу SolveMatrixSystem.

Сигнатура методу виглядає наступним чином:

static Vector SolveMatrixSystem(Matrix A, Vector b, out double detA) – де A це квадратна матриця розмірності n x n, b – вектор стовпець розмірності n. Вихідний параметр detA визначник матриці A, обчислений в ході розв’язання системи.

Результатом роботи алгоритму є розв’язок СЛАР, якщо такий існує, представлений новим вектором.

Оскільки метод перед початком роботи копіює вхідні параметри, його можна вважати thread–safe.

## ***Методи Парето та Слейтера***

Метод Парето є одним з найпростіших та найпопулярніших у теорії прийняття рішень. Кожна раціональна людина по повсякденному житті користується цим методом підсвідомо. Обрати найкращу альтернативу з множини доступних і є оптимальним рішенням.

У бібліотеці метод Парето представлений статичним класом ParetoMethods та має наступні статичні методи:

static bool BetterByParetoThan(this Vector current, Vector vector) – метод розширення об’єкта Vector. Оцінює чи поточний вектор альтернатив є кращим за переданий вектор за принципом Парето. Очікує що два вектори однакової розмірності;

static bool CompareByPareto(Vector vector1, Vector vector2) – аналогічно до попереднього, оцінює чи vector1 є кращим за vector2 за принципом Парето;

static List<Vector> BestByPareto(IEnumerable<Vector> vectors) – метод, що знаходить найкращі альтернативи з множини переданих. Ітераційний метод, що послідовно відкидає доміновані за Парето альтернативи. Очікує що всі вектори однакової розмірності;

При згадці метода Парето, зазвичай зустрічається метод Слейтера, що є тим самим методом відбору альтернатив зі строгішою умовою: краща альтернатива повинна бути кращою за всіма параметрами.

Статичний клас SlaterMethods містить наступні статичні методи:

static bool BetterBySlaterThan(this Vector current, Vector vector) – метод розширення об’єкта Vector. Оцінює чи поточний вектор альтернатив є кращим за переданий вектор за принципом Слейтера. Очікує що два вектори однакової розмірності;

static bool CompareBySlater(Vector vector1, Vector vector2) – аналогічно до попереднього, оцінює чи vector1 є кращим за vector2 за принципом Слейтера;

static List<Vector> BestBySlater(IEnumerable<Vector> vectors) – метод, що знаходить найкращі альтернативи з множини переданих. Ітераційний метод, що послідовно відкидає доміновані за Слейтером альтернативи. Очікує що всі вектори однакової розмірності;

## ***Статистичні методи***

Досить часто при роботі з масивами даних доводиться проводити статистичну оцінку цих даних.

Для вирішення цієї проблеми у бібліотеці присутній статичний клас Statistics, що агрегує в собі наступні статистичні методи:

static double Mean(IEnumerable<double> items) – метод, що знаходить середнє значення переданого масиву чисел;

static double Median(IEnumerable<double> items) – метод, що знаходить медіану переданого масиву чисел, значення, що розташоване посередині масиву;

static double[] Mode(IEnumerable<double> items) – метод, що знаходить моду переданого масиву чисел, значення, що зустрічаються найчастіше. Мод може бути декілька;

static double MathExpectation(IEnumerable<double> items, IEnumerable<double> vectorProbability) – метод, що обчислює математичне сподівання переданого масиву чисел. Параметр vectorProbability позначає розподіл ймовірностей відповідних елементів масиву items. Очікується, що кількість елементів у двох вхідних масивах рівна, а сума ймовірностей рівна 1;

static double Var(IEnumerable<double> items) – метод, що обчислює дисперсію переданого масиву чисел. Центральним моментом є середнє значення;

static double Var(IEnumerable<double> items, IEnumerable<double> vectorProbability) – аналогічно до попереднього методу, обчислює дисперсію масиву чисел, при заданому розподілі ймовірностей елементів масиву items. Очікується, що кількість елементів у двох вхідних масивах рівна, а сума ймовірностей рівна 1;

static double Std(IEnumerable<double> items) – метод, що обчислює стандартне відхилення (середнє квадратичне відхилення) переданого масиву чисел;

static double Std(IEnumerable<double> items, IEnumerable<double> vectorProbability) – аналогічно до попереднього методу, обчислює стандартне відхилення масиву чисел, при заданому розподілі ймовірностей елементів масиву items. Очікується, що кількість елементів у двох вхідних масивах рівна, а сума ймовірностей рівна 1;

static List<double> GenerateRandomArray(int length, double min = 0, double max = 1) – метод, що створює масив заданої довжини, елементами якого є випадкові числа на відрізку [min; max];

static List<T> RandomSampleByRule1<T>(IEnumerable<T> generalSample, int n) – метод, що випадковим відбором без повернення бере з переданого масиву даних n елементів. Результатом є новий масив даних.

static List<T> RandomSampleByRule2<T>(IEnumerable<T> generalSample, int n) – аналогічно до попереднього методу, бере з переданого масиву даних n елементів. Метод є повільнішим, проте потребує менше пам’яті для роботи.

# **Розділ III. Приклад Застосування**

## ***Інструкція з встановлення бібліотеки***

Оскільки бібліотека вільно поширюється пакетним менеджером NuGet, знайти її можна за ім’ям «AppliedMathLibrary».

Покрокова інструкція із встановлення у Visual Studio:

1. створити новий проект у Visual Studio на основі .NET 6 або відкрити існуючий. Важливо саме .NET 6 або вище;
2. клікнувши правою кнопкою мишки по відкритому проекті у solution explorer, вибрати Manage NuGet Packages..;
3. у відкритому вікні, на вкладці Browse ввести у вікно пошуку «AppliedMathLibrary»;
4. встановити знайдену бібліотеку натиском кнопки Install.

A screenshot of a computer

Description automatically generated with medium confidence

Рис. 3. 1. Встановлення бібліотеки AppliedMathLibrary.

Станом на травень 2022 року актуальною версією бібліотеки є 1.0.0.

## ***Тестова програма***

Для того щоб зрозуміти як саме працює бібліотека та яку проблему вона вирішує, потрібно написати програму з наступними вимогами:

* Користувач вводить матрицю та вектор вільних членів СЛАР.
* Розв’язати СЛАР та вивести результат на екран.

Для вирішення проблеми написано просту консольну аплікацію, результат роботи якої представлений на наступному рисунку

Graphical user interface, text

Description automatically generated  
Рис. 3. 2. Результат роботи тестової програми.

Вихідний код програми зображений на Рис. 3. 3.

Text

Description automatically generated  
Рис. 3. 3. Вихідний код тестової програми.

Як видно з малюнку 4, весь код програми, що розв’язує складну математичну задачу помістився у декілька лінійок. Додатково метод SolveMatrixSystem встановив визначник введеної матриці.

# **Розділ IV. тестування**

## ***Модульні тести***

У момент написання будь якої бібліотеки немає можливості практично застосувати написаний метод. Єдиним рішенням проблеми є модульні тести.

Для забезпечення високої надійності рішень, що бібліотека пропонує, написано понад сотню тестів з різноманітними сценаріями використання.

Graphical user interface, text

Description automatically generated  
Рис. 4. 1. Модульні тести бібліотеки AppliedMathLibrary.

Також тести відіграють важливу роль для збереження надійності коду, роблять код толерантним до змін. Оскільки при написанні коду передбачається внесення зміну до нього, тести показують порушення в першопочатковій логіці програми. Це може захистити від ненавмисного пошкодження початкового коду.

## ***Тест продуктивності***

Оскільки у вирішенні деяких проблем може стояти обмеження в часі, або оперативній пам’яті комп’ютера, варто оцінити ефективність методів.

Для вирішення цієї проблеми слугують так звані бенчмарки (або тести продуктивності).

Для прикладу оцінимо два схожі між собою методи: RandomSampleByRule1 та RandomSampleByRule2.

Завдання полягає в наступному: для спільної вибірки з 1000 елементів здійснити відбір 100, 500 та 900 елементів.

Результат оцінки зображену на малюнку:

Calendar

Description automatically generated  
Рис. 4. 2. Результат тестування на продуктивність двох схожих методів.   
1 us = 0.000001 sec.

Як видно із результатів метод RandomSampleByRule1 працює швидше, у всіх трьох випадках, проте виділяє більше оперативної пам’яті.

# **Висновки**

Бібліотека розроблена та знаходиться у вільному доступі для всіх бажаючих.

Особливо корисною буде для студентів – бакалаврів прикладної математики. Безліч рішень реалізовано на основі літератури університету, це може допомогти у розумінні тих чи інших методів.

Варто зауважити що це лише перша верша версія даного продукту.

Кожен бажаючий може вплинути на розвиток даного продукту, запропонувавши свої ідеї для втілення, або допомогу у виправлені помилок.

Вихідний код бібліотеки знаходиться у вільному доступі за посиланнями:

<https://github.com/Rostik18/AppliedMathLibrary>  
 Сторінка бібліотеки на сайті пакетного менеджера NuGet:  
 <https://www.nuget.org/packages/AppliedMathLibrary/>

# **Список використаних джерел**

1. Зеліско В.Р. Основи лінійної алгебри і аналітичної геометрії. / В.Р. Зеліско, Г.В. Зеліско / – Львів: Львівський національний університет імені Івана Франка, 2011. – 326 с.

2. Шахно С.М. Практикум з чисельних методів / С.М. Шахно, А.Т. Дудикевич, С.М. Левицька / − Львів, 2013. − 133 с.

3. Сеньо П.С. Теорія ймовірностей та математична статистика / П.С. Сеньо / – К.: Центр навчальної літератури, 2004. – 448 с.

4. Василик О.І. З теорії і методів вибіркових обстежень. Навчальний посібник / О.І. Василик, Т.О. Яковенко / – Київський національний університет імені Тараса Шевченка, ВПЦ "Київський університет", 2010. – 206 с.

5. Install and manage packages in Visual Studio using the NuGet Package Manager. [Електронний ресурс]. – Режим доступу: [https://docs.microsoft.com/en–us/nuget/consume–packages/install–use–packages–visual–studio](https://docs.microsoft.com/en-us/nuget/consume-packages/install-use-packages-visual-studio)