Міністерство освіти і науки України

Національний технічний університет України „КПІ”

Факультет інформатики і обчислювальної техніки

Кафедра обчислювальної техніки

ПОЯСНЮВАЛЬНА ЗАПИСКА

з курсової роботи з навчальної дисципліни "Системне програмування"

Тема. Бібліотека функцій для роботи з числами підвищеної

точності у мові програмування Java.

**Захищено з оцінкою**

**Керівник роботи**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пустоваров В.І.

(підпис)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ року

**Допущено до захисту**

**Керівник роботи**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пустоваров В.І.

|  |
| --- |
| **Роботу виконав**  Студент групи ІО-21  Кузьменко В.З.  (підпис студента)  ”\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р. |

(підпис)

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ року

Київ – 2014 р.

ЗМІСТ

[ВСТУП 5](#_Toc405835765)

[1. ПРОЕКТУВАННЯ БІБЛІОТЕКИ 6](#_Toc405835766)

[1.1 ОПИС КЛАСІВ BigDecimal і BigInteger 6](#_Toc405835767)

[1.3. ОПИС АЛГОРИТМІВ ФУНКЦІЙ 11](#_Toc405835768)

[1.4. ТЕСТУВАННЯ БІБЛІОТЕКИ 12](#_Toc405835769)

[2. ВИКОРИСТАННЯ БІБЛІОТЕКИ 13](#_Toc405835770)

[2.1 ІНСТРУКЦІЯ КОРИСТУВАЧА 13](#_Toc405835771)

[2.2. ДОКУМЕНТАЦІЯ БІБЛІОТЕКИ 14](#_Toc405835772)

[2.3. ПРИКЛАДИ РЕЗУЛЬТАТУ РОБОТИ 42](#_Toc405835799)

[ВИСНОВКИ 44](#_Toc405835800)

[СПИСОК ІНФОРМАЦІЙНИХ ДЖЕРЕЛ 45](#_Toc405835801)

[ДОДАТОК А. ПРОГРАМНИЙ КОД РОБОТИ 46](#_Toc405835802)

# ВСТУП

У курсовій роботі розробляється програмна бібліотека, що реалізує базові тригонометричні функції, функції для роботи з матрицями та векторами, та деякі інші стандартні математичні функції, реалізація і результат яких потребують використання чисел підвищеної точності. Чиста підвищеної точності подаються об’єктами класів BigDecimal, BigInteger. Числа матимуть четверну точність стандарту IEEE 754.

Даний проект призначений для полегшення виконання процесу розробки програмних продуктів для обчислень підвищеної точності у мові програмування Java.

Розроблено програмне забезпечення на мові Java в середовищі Eclipse. Робота містить документацію та код класів та інструкцію користувача.

1. ПРОЕКТУВАННЯ БІБЛІОТЕКИ

1.1 ОПИС КЛАСІВ BigDecimal і BigInteger

Всі примітивні типи чисел у Java мають обмежений діапазон значень. В цілочисленій арифметиці арифметиці Java нема переповнення, цілі числа приводяться по модулю, рівному діапазон значень.

Для того, щоб можна було проводити цілочисленні обчислення будь якої розрядності, в склад Java API введений клас BigInteger, що зберігається в пакеті java.math. Цей клас розширяє клас Number, відповідно перевизначає в ньому методи doubleValue(), floatValue(), intValue(), longValue(). Методи byteValue() і shortValue() не перевизначені, а прямо наслідуються від класу Number.

Дії з об’єктами класу BigInteger не приводять ні до переповнення, ні до приведення по модулю. Якщо результат операції великий, то число розрядів просто збільшується. Числа зберігаються у двійковій формі в доповняльному коді. Перед виконанням операцій числа вирівнюються по довжині розповсюдженням знакового розряду.

Клас BigDecimal розташований в пакеті java.math. Кожен об’єкт даного класу зберігає два цілочисельних значення: мантису дробового числа у вигляді об’єкту класу BigInteger, і не від’ємний десятковий порядок числа типу int. Наприклад для числа 76.34862 буде зберігатись мантиса 7 634862 в об’єкті класу BigInteger, і порядок 5 як ціле число типу int. Таким чином мантиса може містити будь яку кількість цифр, а порядок обмежений значенням константи Integer.MAX\_VALUE.  
 Для налаштувань роботи деяких методів з класів BigDecimal і BigInteger використовуються класи MathContext і RoundingMode. Незмінний клас MathContext інкапсулює налаштування контексту, який описує певні правила для числових операторів, реалізованих у BigDecimal. У класі MathContext виділені наступні налаштування а також реалізовані методи, для їх використання, що дозволяє ефективніше реалізувати програму:

* precision: число цифр, які будуть використовуватись для роботи, результат заокруглюється до цієї точності.
* roundingMode: RoundingMode об’єкт, це об’єкт який визначає алгоритм, який буде використовуватись для того, щоб округлити число.

У класі RoundingMode реалізовано вісім різних алгоритмів округлення чисел, що використовуються у BigDecimal, їх опис наведено у таблиці 1.1.1

Таблиця 1.1.1 алгоритми округлення у класі RoundingMode

|  |  |
| --- | --- |
| 1.ROUND\_CEILING | Округлення в сторону більшого цілого |
| 2.ROUND\_DOWN | Округлення до нуля, до меншого по модулю цілому значенню |
| 3.ROUND\_FLOOR | Округлення до меншого цілого |
| 4.ROUND\_HALF\_DOWN | Округлення до найближчого цілого, середнє значення округляється до меншого цілого |
| 5.ROUND\_HALF\_EVEN | Округлення до найближчого цілого, середнє значення округляється до парного цілого |
| 6.ROOND\_HALF\_UP | Округлення до найближчого цілого, середнє значення округляється до більшого цілого |
| 7.ROUND\_UNNECESSARY | Передбачається, що результат буде цілим і округлення не знадобиться |
| 8.ROUND\_UP | Округлення від нуля до більшого по модулю цілого значення. |

У класі BigDecimal використовується чотири різних конструктори(таблиця 1.1.2)

Таблиця 1.1.2 Конструктори класу BigDecimal

|  |  |
| --- | --- |
| BigDecimal (Biglnteger bi) | Об’єкт буде зберігати велике ціле значення bi, порядок рівний нулю. |
| BigDecimal (Biglnteger mantissa, int scale) | Задається мантиса (mantissa) і не від’ємний порядок (scale). |
| BigDecimal (double d) | Об’єкт буде містити дробове число подвійної точності d; якщо значення d безкінечне або NaN то виникає виключна ситуація |
| BigDecimal (String val) | Число задається стокою символів val, яка повинна містити запис числа що відповідає правилам мови Java. |

Слід зазначити, що при використанні третього з вище перерахованих конструкторів виникає неприємна особливість. Оскільки дробове число при переводі у двійкову форму представляється, як правило безкінечним двійковим дробом, то при створенні об’єкта, наприклад BigDecimal (0.1) мантиса, що зберігається у об’єкті виявиться дуже великою. У такому випадку рекомендовано використовувати четвертий конструктор, BigDecimal(“0.1”), мантиса буде рівною просто одиниці.

У класі BigDecimal реалізовані методи що моделюють основні, найпростіші операції з дробовими числами, їх опис наведений у таблиці 1.1.3.

Тут x означає об’єкт класу BigDecimal, n – ціле значення типу int, r – спосіб округлення, this – об’єкт BigDecimal до якого застосовується даний метод.

Таблиця 1.1.3. Методи класу BigDecimal

|  |  |
| --- | --- |
| Abs() | Абсолютне значення об’єкта this |
| Add(x) | Операція this + x |
| Divide(x,r) | Операція this/x з округленням по способу r |
| Divide(x,n,r) | Операція this/x з зміною порядку та округленням по способу r |
| Max(x) | Найбільше із this і х |
| Min(x) | Найменше із this і x |
| movePointLeft(n) | Здвиг вліво на nрозрядів |
| movePointRight(n) | Здвиг вправо на n розрядів |
| Multiply(x) | Операція this\*x |
| Negate() | Повертає об’єкт з оберненим знаком |
| Scale() | Повертає порядок числа |
| Setscaie(n) | Встановлює новий порядок n |
| Setscaie(n,r) | Встановлює новий порядок n і округляє число при необхідності по способу r |
| Signumo | Знак числа, що зберігається у об’єкті |
| Subtract(x) | Операція this – x |
| toBigInteger() | Округлення числа що знаходиться в об’єкті |
| Unscaiedvalue() | Повертає мантису числа |

1.2. ОПИС АЛГОРИТМУ ТА БЛОК-СХЕМИ

Для використання бібліотеки необіхдно підключити її одним із способів. Далі необхідно створити об’єкт класу BigDecimal і використовувати функції бібліотеки для обчислень.



Рисунок 1.1.1 Блок-схема алгоритму використання бібліотеки

Загальний алгоритм обчислення тригонометричних та інших функцій рядом Тейлора.



Рисунок1.1.2. Блок-схема алгоритму обчислень функцій рядом Тейлора

У деяких функцій можуть виникати виключні ситуації. Перш за все це зумовлено неправильними вхідними данними, або їх неправильним форматом. У цьому випадку функція біліотеки повертає об’єкт виключної ситуації. Нижче наведений алгоритм її виникнення. Аналіз виключної ситуації та її обробка покладаються на користувача бібліотеки.



Рисунок 1.1.3 Алгоритм виникнення виключної ситуації

## 1.3. ОПИС АЛГОРИТМІВ ФУНКЦІЙ

Для реалізацій базових тригонометричних функцій(sin, cos, arcsin, arctn) використовувались ряди Тейлора. Нижче наведено формули по яких проводився обрахунок.

Для синуса:

Для косинуса

Для арксинуса

Для арктангенса

Решта тригонометричних, обернено тригонометричних та гіперболічних функцій обчислюються через еквіваленті математичні перетворення.  
 Для реалізації функції обчислення експоненціальної, логарифмічної функцій використовується також розклад у ряд:

Для експоненціальної функції:

Для логарифмічної функції:

Для реалізації функції помилок використано такий розклад у ряд:

Для реалізації даних математичних функцій необхідно реалізувати обчислення за даними формулами враховуючи точність обчислень. Для даних реалізацій достатньо стандартних арифметичних операцій, що виконуються з об’єктами BigDecimal.

Для ефективнішого обчислення факторіалу числа використовується хеш-таблиця що зберігає попередньо обчисленні значення факторіалу.

## 1.4. ТЕСТУВАННЯ БІБЛІОТЕКИ

Перевірка правильності роботи здійснюється за допомогою системи тестів, розроблених з використанням інструменту тестування JUnit та збережених в пакеті tests. Варто відмітити, що даний пакет не був включений до документації та додатків, оскільки є громіздким та не призначений для користувача.

Наступні приклад прогамного коду тестів демонструють спосіб використання тригонометричних функцій бібліотеки:

Використання бібліотеки для обчислення значення тригонометричної функції на прикладі функції синуса:

@Test

public void testSin() {

BigDecimal x1 = new BigDecimal(“3.5”);

BigDecimal result = SimpleMathUtils.sin(x1);

System.out.println(result.toString());

}

Приклад використання експоненти числа:

public void test() {

BigDecimal x = new BigDecimal(“4.435342134”);

BigDecimal result = SimpleMathUtils.exp(x1);

System.out.println(result.toString());

}

Приклад обчислення складної функції:

@Test

public void testFunk(){

BigDecimal x = new BigDecimal(2.1);

BigDecimal result = sin(sqrt(sin(x).pow(2).multiply(cos(x.multiply(ln(x))))));

System.out.println(result.toString());

}

# 2. ВИКОРИСТАННЯ БІБЛІОТЕКИ

Розроблений продукт зберігається у файлі simpleBigDecimalUtils.jar та може бути використаний у будь-якому Java проекті.

## 2.1 ІНСТРУКЦІЯ КОРИСТУВАЧА

Для використання бібліотеки, необхідно підключити її до проекту. Це можна зробити декількома способами. Розглянемо детально два способи підключення бібліотеки до проекту у якому вона буде використовуватись. Для початку необхідно скопіювати jar бібліотеку simpleBigDecimalUtils.jar у папку lib проекту де зберігаються усі бібліотеки проекту. Далі, використовуючи середовище розробки необхідно у налаштуваннях збірки проекту додати jar бібліотеку. Також існують інші способи підключення бібліотеки, зокрема підключення через змінну середовища classpath. Дані операції можливо виконувати з консолі, проте рекомендовано виконувати їх через середовище розробки програми для уникнення помилок.

У бібліотекці числа мають четверну точність стандарту IEEE 754.

Після успішного підключення бібліотеки стає доступною можливість використовувати реалізовані функції. За змістом функції бібліотеки розподілені між трьома класами.

Тригонометричні функції, гіперболічні функції, логарифмічна, експоненціальна функції знаходяться у класі SimpleMathUtils.

Функції для роботи з векторами та матрицями (сума, різниця, добуток, транспонування, множення, обчислення визначника) знаходяться у класі SimpleLinearUtils.

Решта функцій, а саме, функція помилок, функція отримання випадкового числа розподіленого за нормальним або рівномірним законом знаходяться у класі OtherMathUtils.

Також для користувача доступна функція обчислення факторіалу, вона міститься у класі Utils.

Вхідні дані користувач повинен вводити лише у трьох наступних форматах:

* об’єктом класу BigDecimal
* об’єктом класу String
* простим типом double або його класом-обгорткою Double.

Формат повернення результатів роботи функції даної бібліотеки визначається лише об’єктом BigDecimal, який у разі необхідності можливо перетворити на строку(об’єкт класу String) або у double число.

За детальною інформацією щодо роботи тої чи іншої функції, їх аргументів та результату необхідно звернутись до документації бібліотеки, що наведена нижче (пункт 2.2) або до класу-тесту, що включено до бібліотеки.

2.2. ДОКУМЕНТАЦІЯ БІБЛІОТЕКИ

# SimpleBigDecimalUtils

|  |  |
| --- | --- |
| **Packages** | |
| **Package** | **Description** |
| [**com.sp2.courseWork.BigDemUtils**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\package-summary.html) |  |
| [**com.sp2.courseWork.test**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\test\package-summary.html) |  |

# Package com.sp2.courseWork.BigDemUtils

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**OtherMathUtils**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\OtherMathUtils.html) | Class for store erf, factorial nad random function. |
| [**SimpleLinearUtils**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html) | class for linear utils. |
| [**SimpleMathUtils**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html) | The Class SimpleMathUtils. |
| [**Utils**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\Utils.html) | utils for library. |

com.sp2.courseWork.BigDemUtils

## Class SimpleMathUtils

* java.lang.Object
  + com.sp2.courseWork.BigDemUtils.SimpleMathUtils

public class **SimpleMathUtils**

extends java.lang.Object

The Class SimpleMathUtils.

**Author:**

Kuzmenko Volodimir

### Constructor Summary

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**SimpleMathUtils**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#SimpleMathUtils())() |

### Method Summary

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| static java.math.BigDecimal | [**acos**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#acos(java.math.BigDecimal))(java.math.BigDecimal x)  arc cosine. |
| static java.math.BigDecimal | [**acos**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#acos(double))(double x)  arc cosine. |
| static java.math.BigDecimal | [**acos**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#acos(java.lang.String))(java.lang.String x)  arc cosine. |
| static java.math.BigDecimal | [**actan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#actan(java.math.BigDecimal))(java.math.BigDecimal x)  arc cotangent. |
| static java.math.BigDecimal | [**actan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#actan(double))(double x)  arc cotangent. |
| static java.math.BigDecimal | [**actan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#actan(java.lang.String))(java.lang.String x)  arc cotangent. |
| static java.math.BigDecimal | [**asin**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#asin(java.math.BigDecimal))(java.math.BigDecimal x)  arc sine. |
| static java.math.BigDecimal | [**asin**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#asin(double))(double x)  arc sine. |
| static java.math.BigDecimal | [**asin**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#asin(java.lang.String))(java.lang.String x)  arc sine. |
| static java.math.BigDecimal | [**atan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#atan(java.math.BigDecimal))(java.math.BigDecimal x)  arc tangent. |
| static java.math.BigDecimal | [**atan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#atan(double))(double x)  arc tangent. |
| static java.math.BigDecimal | [**atan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#atan(java.lang.String))(java.lang.String x)  arc tangent. |
| static java.math.BigDecimal | [**cos**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#cos(java.math.BigDecimal))(java.math.BigDecimal x)  cosine. |
| static java.math.BigDecimal | [**cos**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#cos(double))(double x)  cosine. |
| static java.math.BigDecimal | [**cos**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#cos(java.lang.String))(java.lang.String x)  cosine. |
| static java.math.BigDecimal | [**cosh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#cosh(java.math.BigDecimal))(java.math.BigDecimal x)  hyperbolic cosine. |
| static java.math.BigDecimal | [**cosh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#cosh(double))(double x)  hyperbolic cosine. |
| static java.math.BigDecimal | [**cosh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#cosh(java.lang.String))(java.lang.String x)  hyperbolic cosine. |
| static java.math.BigDecimal | [**ctan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#ctan(java.math.BigDecimal))(java.math.BigDecimal x)  cotangent. |
| static java.math.BigDecimal | [**ctan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#ctan(double))(double x)  cotangent. |
| static java.math.BigDecimal | [**ctan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#ctan(java.lang.String))(java.lang.String x)  cotangent. |
| static java.math.BigDecimal | [**ctanh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#ctanh(java.math.BigDecimal))(java.math.BigDecimal x)  hyperbolic cotangent. |
| static java.math.BigDecimal | [**ctanh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#ctanh(double))(double x)  hyperbolic cotangent. |
| static java.math.BigDecimal | [**ctanh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#ctanh(java.lang.String))(java.lang.String x)  hyperbolic cotangent. |
| static java.math.BigDecimal | [**exp**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#exp(java.math.BigDecimal))(java.math.BigDecimal x)  exponent with BigDecimal agruments. |
| static java.math.BigDecimal | [**exp**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#exp(double))(double x)  exponent with double arguments. |
| static java.math.BigDecimal | [**exp**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#exp(java.lang.String))(java.lang.String x)  exponent with String arguments. |
| static java.math.BigDecimal | [**lg**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#lg(java.math.BigDecimal))(java.math.BigDecimal x)  common logarithm. |
| static java.math.BigDecimal | [**lg**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#lg(double))(double x)  common logarithm. |
| static java.math.BigDecimal | [**lg**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#lg(java.lang.String))(java.lang.String x)  common logarithm. |
| static java.math.BigDecimal | [**ln**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#ln(java.math.BigDecimal))(java.math.BigDecimal x)  natural logarithm. |
| static java.math.BigDecimal | [**ln**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#ln(double))(double x)  natural logarithm with double arguments. |
| static java.math.BigDecimal | [**ln**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#ln(java.lang.String))(java.lang.String x) natural logarithm. |
| static java.math.BigDecimal | [**log**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#log(java.math.BigDecimal, java.math.BigDecimal))(java.math.BigDecimal x, java.math.BigDecimal a)logarithm. |
| static java.math.BigDecimal | [**log**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#log(double, double))(double x, double a)  logarithm. |
| static java.math.BigDecimal | [**log**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#log(java.lang.String, java.lang.String))(java.lang.String x, java.lang.String a)  logarithm. |
| static java.math.BigDecimal | [**sin**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#sin(java.math.BigDecimal))(java.math.BigDecimal x) sine. |
| static java.math.BigDecimal | [**sin**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#sin(double))(double x) sine. |
| static java.math.BigDecimal | [**sin**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#sin(java.lang.String))(java.lang.String x) sine. |
| static java.math.BigDecimal | [**sinh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#sinh(java.math.BigDecimal))(java.math.BigDecimal x) hyperbolic sine. |
| static java.math.BigDecimal | [**sinh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#sinh(double))(double x) hyperbolic sine. |
| static java.math.BigDecimal | [**sinh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#sinh(java.lang.String))(java.lang.String x)  hyperbolic sine. |
| static java.math.BigDecimal | [**sqrt**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#sqrt(java.math.BigDecimal))(java.math.BigDecimal x) square root. |
| static java.math.BigDecimal | [**sqrt**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#sqrt(double))(double x) square root. |
| static java.math.BigDecimal | [**sqrt**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#sqrt(java.lang.String))(java.lang.String x) square root. |
| static java.math.BigDecimal | [**tan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#tan(java.math.BigDecimal))(java.math.BigDecimal x) tangent. |
| static java.math.BigDecimal | [**tan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#tan(double))(double x) tangents. |
| static java.math.BigDecimal | [**tan**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#tan(java.lang.String))(java.lang.String x)  tangent. |
| static java.math.BigDecimal | [**tanh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#tanh(java.math.BigDecimal))(java.math.BigDecimal x)  hyperbolic tangent. |
| static java.math.BigDecimal | [**tanh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#tanh(double))(double x)  hyperbolic tangent. |
| static java.math.BigDecimal | [**tanh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleMathUtils.html#tanh(java.lang.String))(java.lang.String x)  hyperbolic tangent. |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

#### SimpleMathUtils

public SimpleMathUtils()

### Method Detail

#### sin

public static java.math.BigDecimal sin(java.math.BigDecimal x)

sine.

**Parameters:**

x - BigDecimal argument

**Returns:**

sin(x)

#### sin

public static java.math.BigDecimal sin(double x)

sine.

**Parameters:**

x - double argument

**Returns:**

sin(x)

#### sin

public static java.math.BigDecimal sin(java.lang.String x)

sine.

**Parameters:**

x - String argument

**Returns:**

sin(x)

#### cos

public static java.math.BigDecimal cos(java.math.BigDecimal x)

cosine.

**Parameters:**

x - BigDecimal argument

**Returns:**

cos(x)

#### cos

public static java.math.BigDecimal cos(double x)

cosine.

**Parameters:**

x - BigDecimal argument

**Returns:**

cos(x)

#### cos

public static java.math.BigDecimal cos(java.lang.String x)

cosine.

**Parameters:**

x - String argument

**Returns:**

cos(x)

#### tan

public static java.math.BigDecimal tan(java.math.BigDecimal x)

tangent.

**Parameters:**

x - BigDecimal argument

**Returns:**

tan(x)

#### tan

public static java.math.BigDecimal tan(double x)

tangents.

**Parameters:**

x - BigDecimal argument

**Returns:**

sin(x)

#### tan

public static java.math.BigDecimal tan(java.lang.String x)

tangent.

**Parameters:**

x - String argument

**Returns:**

sin(x)

#### ctan

public static java.math.BigDecimal ctan(java.math.BigDecimal x)

cotangent.

**Parameters:**

x - BigDecimal argument

**Returns:**

ctan(x)

#### ctan

public static java.math.BigDecimal ctan(double x)

cotangent.

**Parameters:**

x - BigDecimal argument

**Returns:**

ctan(x)

#### ctan

public static java.math.BigDecimal ctan(java.lang.String x)

cotangent.

**Parameters:**

x - String argument

**Returns:**

ctan(x)

#### asin

public static java.math.BigDecimal asin(java.math.BigDecimal x)

arc sine.

**Parameters:**

x - BigDecimal argument

**Returns:**

asin(x)

#### asin

public static java.math.BigDecimal asin(double x)

arc sine.

**Parameters:**

x - BigDecimal argument

**Returns:**

asin(x)

#### asin

public static java.math.BigDecimal asin(java.lang.String x)

arc sine.

**Parameters:**

x - String argument

**Returns:**

asin(x)

#### acos

public static java.math.BigDecimal acos(java.math.BigDecimal x)

arc cosine.

**Parameters:**

x - BigDecimal argument

**Returns:**

acos(x)

#### acos

public static java.math.BigDecimal acos(double x)

arc cosine.

**Parameters:**

x - BigDecimal argument

**Returns:**

cosine(x)

#### acos

public static java.math.BigDecimal acos(java.lang.String x)

arc cosine.

**Parameters:**

x - String argument

**Returns:**

acos(x)

#### atan

public static java.math.BigDecimal atan(java.math.BigDecimal x)

arc tangent.

**Parameters:**

x - BigDecimal argument

**Returns:**

atan(x)

#### atan

public static java.math.BigDecimal atan(double x)

arc tangent.

**Parameters:**

x - BigDecimal argument

**Returns:**

atan(x)

#### atan

public static java.math.BigDecimal atan(java.lang.String x)

arc tangent.

**Parameters:**

x - String argument

**Returns:**

atan(x)

#### actan

public static java.math.BigDecimal actan(java.math.BigDecimal x)

arc cotangent.

**Parameters:**

x - BigDecimal argument

**Returns:**

actan(x)

#### actan

public static java.math.BigDecimal actan(double x)

arc cotangent.

**Parameters:**

x - BigDecimal argument

**Returns:**

atan(x)

#### actan

public static java.math.BigDecimal actan(java.lang.String x)

arc cotangent.

**Parameters:**

x - String argument

**Returns:**

atan(x)

#### sinh

public static java.math.BigDecimal sinh(java.math.BigDecimal x)

hyperbolic sine.

**Parameters:**

x - BigDecimal argument

**Returns:**

sinh(x)

#### sinh

public static java.math.BigDecimal sinh(double x)

hyperbolic sine.

**Parameters:**

x - BigDecimal argument

**Returns:**

sinh(x)

#### sinh

public static java.math.BigDecimal sinh(java.lang.String x)

hyperbolic sine.

**Parameters:**

x - String argument

**Returns:**

sin(x)

#### cosh

public static java.math.BigDecimal cosh(java.math.BigDecimal x)

hyperbolic cosine.

**Parameters:**

x - BigDecimal argument

**Returns:**

cosh(x)

#### cosh

public static java.math.BigDecimal cosh(double x)

hyperbolic cosine.

**Parameters:**

x - BigDecimal argument

**Returns:**

cosh(x)

#### cosh

public static java.math.BigDecimal cosh(java.lang.String x)

hyperbolic cosine.

**Parameters:**

x - String argument

**Returns:**

cosh(x)

#### tanh

public static java.math.BigDecimal tanh(java.math.BigDecimal x)

hyperbolic tangent.

**Parameters:**

x - BigDecimal argument

**Returns:**

tanh(x)

#### tanh

public static java.math.BigDecimal tanh(double x)

hyperbolic tangent.

**Parameters:**

x - BigDecimal argument

**Returns:**

tanh(x)

#### tanh

public static java.math.BigDecimal tanh(java.lang.String x)

hyperbolic tangent.

**Parameters:**

x - String argument

**Returns:**

tanh(x)

#### ctanh

public static java.math.BigDecimal ctanh(java.math.BigDecimal x)

hyperbolic cotangent.

**Parameters:**

x - BigDecimal argument

**Returns:**

s(x)

#### ctanh

public static java.math.BigDecimal ctanh(double x)

hyperbolic cotangent.

**Parameters:**

x - double argument

**Returns:**

ctanh(x)

#### ctanh

public static java.math.BigDecimal ctanh(java.lang.String x)

hyperbolic cotangent.

**Parameters:**

x - BigDecimal argument

**Returns:**

sin(x)

#### exp

public static java.math.BigDecimal exp(double x)

exponent with double arguments.

**Parameters:**

x - double argument

**Returns:**

exp(x)

#### exp

public static java.math.BigDecimal exp(java.math.BigDecimal x)

exponent with BigDecimal agruments.

**Parameters:**

x - BigDecimal argument

**Returns:**

exp(x)

#### exp

public static java.math.BigDecimal exp(java.lang.String x)

exponent with String arguments.

**Parameters:**

x - String argument

**Returns:**

exp(x)

#### ln

public static java.math.BigDecimal ln(java.math.BigDecimal x)

natural logarithm.

**Parameters:**

x - BigDecimal argument

**Returns:**

ln(x)

#### ln

public static java.math.BigDecimal ln(double x)

natural logarithm with double arguments.

**Parameters:**

x - double argument

**Returns:**

ln(x)

#### ln

public static java.math.BigDecimal ln(java.lang.String x)

natural logarithm.

**Parameters:**

x - String argument

**Returns:**

ln(x)

#### lg

public static java.math.BigDecimal lg(java.math.BigDecimal x)

common logarithm.

**Parameters:**

x - BigDecimal argument

**Returns:**

sin(x)

#### lg

public static java.math.BigDecimal lg(double x)

common logarithm.

**Parameters:**

x - BigDecimal argument

**Returns:**

lg(x)

#### lg

public static java.math.BigDecimal lg(java.lang.String x)

common logarithm.

**Parameters:**

x - String argument

**Returns:**

lg(x)

#### log

* + - public static java.math.BigDecimal log(java.math.BigDecimal x,

java.math.BigDecimal a)

logarithm.

**Parameters:**

x - BigDecimal argument

a - the a

**Returns:**

log(x)

#### log

* + - public static java.math.BigDecimal log(double x,

double a)

logarithm.

**Parameters:**

x - BigDecimal argument

a - the a

**Returns:**

log(x)

#### log

* + - public static java.math.BigDecimal log(java.lang.String x,

java.lang.String a)

logarithm.

**Parameters:**

x - String argument

a - the a

**Returns:**

log(x)

#### sqrt

public static java.math.BigDecimal sqrt(java.math.BigDecimal x)

square root.

**Parameters:**

x - BigDecimal argument

**Returns:**

sqrt(x)

#### sqrt

public static java.math.BigDecimal sqrt(double x)

square root.

**Parameters:**

x - double argument

**Returns:**

sqrt(x)

#### sqrt

public static java.math.BigDecimal sqrt(java.lang.String x)

square root.

**Parameters:**

x - String argument

**Returns:**

sqrt(x)

com.sp2.courseWork.BigDemUtils

## Class SimpleLinearUtils

* java.lang.Object
  + com.sp2.courseWork.BigDemUtils.SimpleLinearUtils

public class **SimpleLinearUtils**

extends java.lang.Object

class for linear utils.

**Author:**

Kuzmenko Volodimir

### Constructor Summary

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**SimpleLinearUtils**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#SimpleLinearUtils())() |

### Method Summary

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| static java.math.BigDecimal[][] | [**add**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#add(java.math.BigDecimal[][], java.math.BigDecimal))(java.math.BigDecimal[][] a, java.math.BigDecimal b)  sum matrix and number with BigDecimal arguments MA+b. |
| static java.math.BigDecimal[][] | [**add**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#add(java.math.BigDecimal[][], java.math.BigDecimal[][]))(java.math.BigDecimal[][] a, java.math.BigDecimal[][] b)  matrix sum with BigDecimal arguments MA+MB. |
| static java.math.BigDecimal[] | [**add**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#add(java.math.BigDecimal[], java.math.BigDecimal))(java.math.BigDecimal[] a, java.math.BigDecimal b)  vectors sum with double arguments A+b. |
| static java.math.BigDecimal[] | [**add**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#add(java.math.BigDecimal[], java.math.BigDecimal[]))(java.math.BigDecimal[] a, java.math.BigDecimal[] b)  matrix on number sum with BigDecimal arguments A + B. |
| static java.math.BigDecimal[][] | [**add**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#add(double[][], double))(double[][] a, double b)  sum matrices with double arguments MA + b. |
| static java.math.BigDecimal[][] | [**add**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#add(double[][], double[][]))(double[][] a, double[][] b)  matrix sum with double arguments MA+MB. |
| static java.math.BigDecimal[] | [**add**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#add(double[], double))(double[] a, double b)  vector and number sum with double arguments A+b. |
| static java.math.BigDecimal[] | [**add**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#add(double[], double[]))(double[] a, double[] b)  vectors sum with BigDecimal arguments A+B. |
| static int | [**binarySearch**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#binarySearch(java.math.BigDecimal[][], java.math.BigDecimal))(java.math.BigDecimal[][] ma, java.math.BigDecimal key)  binary search search in matrix. |
| static java.math.BigDecimal | [**det**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#det(java.math.BigDecimal[][]))(java.math.BigDecimal[][] ma)  determinant calculation. |
| static java.math.BigDecimal | [**max**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#max(java.math.BigDecimal[]))(java.math.BigDecimal[] ma)  search max element in matrix ma. |
| static java.math.BigDecimal | [**min**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#min(java.math.BigDecimal[]))(java.math.BigDecimal[] ma)  search min element in matrix. |
| static java.math.BigDecimal[][] | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(java.math.BigDecimal[][], java.math.BigDecimal))(java.math.BigDecimal[][] a, java.math.BigDecimal b)  matrix on number multiplication with BigDecimal arguments MA\*b. |
| static java.math.BigDecimal[] | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(java.math.BigDecimal[][], java.math.BigDecimal[]))(java.math.BigDecimal[][] ma, java.math.BigDecimal[] b)  matrix on vector multiplication with BigDecimal arguments MA\*B . |
| static java.math.BigDecimal[][] | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(java.math.BigDecimal[][], java.math.BigDecimal[][]))(java.math.BigDecimal[][] a, java.math.BigDecimal[][] b)  matrix multiplication with BigDecimal arguments MA\*MB. |
| static java.math.BigDecimal[] | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(java.math.BigDecimal[], java.math.BigDecimal))(java.math.BigDecimal[] a, java.math.BigDecimal b)  vector on number multiplication with double arguments A\*b. |
| static java.math.BigDecimal | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(java.math.BigDecimal[], java.math.BigDecimal[]))(java.math.BigDecimal[] a, java.math.BigDecimal[] b)  vector multiplication with BigDecimal arguments A\*B. |
| static java.math.BigDecimal[] | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(java.math.BigDecimal[], java.math.BigDecimal[][]))(java.math.BigDecimal[] a, java.math.BigDecimal[][] mb)  vector on matrix multiplication with BigDecimal arguments A\*MB. |
| static java.math.BigDecimal[][] | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(double[][], double))(double[][] a, double b)  matrix on number multiplication with double arguments MA\*b. |
| static java.math.BigDecimal[] | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(double[][], double[]))(double[][] ma, double[] b)  matrix on vector multiplication with double arguments MA\*B. |
| static java.math.BigDecimal[][] | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(double[][], double[][]))(double[][] a, double[][] b)  matrix multiplication with double arguments MA\*MB. |
| static java.math.BigDecimal[] | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(double[], double))(double[] a, double b)  vector on number multiplication with double arguments A\*b. |
| static java.math.BigDecimal | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(double[], double[]))(double[] a, double[] b)  vector multiplication with double arguments A\*B. |
| static java.math.BigDecimal[] | [**mult**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#mult(double[], double[][]))(double[] a, double[][] mb)  vector on matrix multiplication with double arguments A\*MB . |
| static java.math.BigDecimal[] | [**sort**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#sort(java.math.BigDecimal[]))(java.math.BigDecimal[] a)  vector sorting. |
| static java.math.BigDecimal[][] | [**sub**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#sub(java.math.BigDecimal[][], java.math.BigDecimal[][]))(java.math.BigDecimal[][] a, java.math.BigDecimal[][] b)  difference matrices with BigDecimal arguments MA-MB. |
| static java.math.BigDecimal[] | [**sub**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#sub(java.math.BigDecimal[], java.math.BigDecimal[]))(java.math.BigDecimal[] a, java.math.BigDecimal[] b)  difference vectors with BigDecimal arguments A-B. |
| static java.math.BigDecimal[][] | [**sub**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#sub(double[][], double[][]))(double[][] a, double[][] b)  difference matrices with BigDecimal arguments MA-MB. |
| static java.math.BigDecimal[] | [**sub**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#sub(double[], double[]))(double[] a, double[] b)  difference vectors with double arguments A-A. |
| java.math.BigDecimal[][] | [**transpone**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\SimpleLinearUtils.html#transpone(java.math.BigDecimal[][]))(java.math.BigDecimal[][] ma)  matrix transposition. |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

#### SimpleLinearUtils

public SimpleLinearUtils()

### Method Detail

#### mult

* + - public static java.math.BigDecimal[][] mult(java.math.BigDecimal[][] a,

java.math.BigDecimal[][] b)

matrix multiplication with BigDecimal arguments MA\*MB.

**Parameters:**

a - matrix MA

b - matrix MB

**Returns:**

matrix MA\*MB

#### mult

* + - public static java.math.BigDecimal[][] mult(double[][] a,

double[][] b)

matrix multiplication with double arguments MA\*MB.

**Parameters:**

a - matrix MA

b - matrix MB

**Returns:**

matrix MA\*MB

#### mult

* + - public static java.math.BigDecimal mult(double[] a,

double[] b)

vector multiplication with double arguments A\*B.

**Parameters:**

a - vector A

b - vector B

**Returns:**

value result A\*B

#### mult

* + - public static java.math.BigDecimal mult(java.math.BigDecimal[] a,

java.math.BigDecimal[] b)

vector multiplication with BigDecimal arguments A\*B.

**Parameters:**

a - vector A

b - vector B

**Returns:**

value result A\*B

#### mult

* + - public static java.math.BigDecimal[] mult(java.math.BigDecimal[][] ma,

java.math.BigDecimal[] b)

matrix on vector multiplication with BigDecimal arguments MA\*B .

**Parameters:**

ma - matrix MA

b - vector B

**Returns:**

vector result MA\*B

#### mult

* + - public static java.math.BigDecimal[] mult(double[][] ma,

double[] b)

matrix on vector multiplication with double arguments MA\*B.

**Parameters:**

ma - matrix MA

b - vector B

**Returns:**

vector value MA\*B

#### mult

* + - public static java.math.BigDecimal[] mult(java.math.BigDecimal[] a,

java.math.BigDecimal[][] mb)

vector on matrix multiplication with BigDecimal arguments A\*MB.

**Parameters:**

a - vector A

mb - matrix MB

**Returns:**

vector result A\*MB

#### mult

* + - public static java.math.BigDecimal[] mult(double[] a,

double[][] mb)

vector on matrix multiplication with double arguments A\*MB .

**Parameters:**

a - vector A

mb - matrix MD

**Returns:**

vector result A\*MB

#### add

* + - public static java.math.BigDecimal[][] add(java.math.BigDecimal[][] a,

java.math.BigDecimal[][] b)

matrix sum with BigDecimal arguments MA+MB.

**Parameters:**

a - matrix MA

b - matrix MB

**Returns:**

matrix MA+MB

#### add

* + - public static java.math.BigDecimal[][] add(double[][] a,

double[][] b)

matrix sum with double arguments MA+MB.

**Parameters:**

a - matrix MA

b - matrix MB

**Returns:**

matrix MA+MB

#### add

* + - public static java.math.BigDecimal[][] add(java.math.BigDecimal[][] a,

java.math.BigDecimal b)

sum matrix and number with BigDecimal arguments MA+b.

**Parameters:**

a - matrix MA

b - number b

**Returns:**

matrix MA+b

#### add

* + - public static java.math.BigDecimal[][] add(double[][] a,

double b)

sum matrices with double arguments MA + b.

**Parameters:**

a - matrix MA

b - number b

**Returns:**

matrix MA+b

#### sub

* + - public static java.math.BigDecimal[][] sub(java.math.BigDecimal[][] a,

java.math.BigDecimal[][] b)

difference matrices with BigDecimal arguments MA-MB.

**Parameters:**

a - matrix MA

b - matrix MB

**Returns:**

matrix MA-MB

#### sub

* + - public static java.math.BigDecimal[][] sub(double[][] a,

double[][] b)

difference matrices with BigDecimal arguments MA-MB.

**Parameters:**

a - matrix MA

b - matrix MB

**Returns:**

matrix MA-MB

#### mult

* + - public static java.math.BigDecimal[][] mult(java.math.BigDecimal[][] a,

java.math.BigDecimal b)

matrix on number multiplication with BigDecimal arguments MA\*b.

**Parameters:**

a - matrix MA

b - value b

**Returns:**

matrix MA\*b

#### mult

* + - public static java.math.BigDecimal[][] mult(double[][] a,

double b)

matrix on number multiplication with double arguments MA\*b.

**Parameters:**

a - matrix MA

b - value b

**Returns:**

matix MA\*b

#### add

* + - public static java.math.BigDecimal[] add(java.math.BigDecimal[] a,

java.math.BigDecimal[] b)

matrix on number sum with BigDecimal arguments A + B.

**Parameters:**

a - vector A

b - vector B

**Returns:**

vector A+B

#### add

* + - public static java.math.BigDecimal[] add(double[] a,

double[] b)

vectors sum with BigDecimal arguments A+B.

**Parameters:**

a - vector A

b - vector B

**Returns:**

vector A+B

#### add

* + - public static java.math.BigDecimal[] add(java.math.BigDecimal[] a,

java.math.BigDecimal b)

vectors sum with double arguments A+b.

**Parameters:**

a - vector A

b - number b

**Returns:**

vector A+b

#### add

* + - public static java.math.BigDecimal[] add(double[] a,

double b)

vector and number sum with double arguments A+b.

**Parameters:**

a - vector A

b - number b

**Returns:**

vector A+b

#### sub

* + - public static java.math.BigDecimal[] sub(java.math.BigDecimal[] a,

java.math.BigDecimal[] b)

difference vectors with BigDecimal arguments A-B.

**Parameters:**

a - vector A

b - vector B

**Returns:**

vevtor A-B

#### sub

* + - public static java.math.BigDecimal[] sub(double[] a,

double[] b)

difference vectors with double arguments A-A.

**Parameters:**

a - vector A

b - vector A

**Returns:**

vector A-A

#### mult

* + - public static java.math.BigDecimal[] mult(java.math.BigDecimal[] a,

java.math.BigDecimal b)

vector on number multiplication with double arguments A\*b.

**Parameters:**

a - vector A

b - number b

**Returns:**

vector A\*b

#### mult

* + - public static java.math.BigDecimal[] mult(double[] a,

double b)

vector on number multiplication with double arguments A\*b.

**Parameters:**

a - vector A

b - number b

**Returns:**

vector A\*b

#### sort

public static java.math.BigDecimal[] sort(java.math.BigDecimal[] a)

vector sorting.

**Parameters:**

a - vector A

**Returns:**

result

#### binarySearch

* + - public static int binarySearch(java.math.BigDecimal[][] ma,

java.math.BigDecimal key)

binary search search in matrix.

**Parameters:**

ma - the ma

key - the key

**Returns:**

the int

#### det

public static java.math.BigDecimal det(java.math.BigDecimal[][] ma)

determinant calculation.

**Parameters:**

ma - matrix

**Returns:**

determinant

#### max

public static java.math.BigDecimal max(java.math.BigDecimal[] ma)

search max element in matrix ma.

**Parameters:**

ma - matrix

**Returns:**

max element

#### min

public static java.math.BigDecimal min(java.math.BigDecimal[] ma)

search min element in matrix.

**Parameters:**

ma - matrix

**Returns:**

min element

#### transpone

public java.math.BigDecimal[][] transpone(java.math.BigDecimal[][] ma)

matrix transposition.

**Parameters:**

ma - matrix

**Returns:**

transposition matrix

com.sp2.courseWork.BigDemUtils

## Class OtherMathUtils

* java.lang.Object
  + com.sp2.courseWork.BigDemUtils.OtherMathUtils

public class **OtherMathUtils**

extends java.lang.Object

Class for store erf, factorial nad random function.

**Author:**

Kuzmenko Volodimir

### Constructor Summary

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**OtherMathUtils**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\OtherMathUtils.html#OtherMathUtils())() |

### Method Summary

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| static java.math.BigDecimal | [**erf**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\OtherMathUtils.html#erf(java.math.BigDecimal))(java.math.BigDecimal x)  error function. |
| static java.math.BigDecimal | [**nextGaussian**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\OtherMathUtils.html#nextGaussian())()  function for standard normal distribution . |
| static java.math.BigDecimal | [**nextRandom**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\OtherMathUtils.html#nextRandom())()  function for uniformly distribution between 0 and 1. |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

#### OtherMathUtils

public OtherMathUtils()

### Method Detail

#### erf

public static java.math.BigDecimal erf(java.math.BigDecimal x)

error function.

**Parameters:**

x - argument function

**Returns:**

result value

#### nextRandom

public static java.math.BigDecimal nextRandom()

function for uniformly distribution between 0 and 1.

**Returns:**

random number

#### nextGaussian

public static java.math.BigDecimal nextGaussian()

function for standard normal distribution .

**Returns:**

random number for this distribution

com.sp2.courseWork.BigDemUtils

## Class Utils

* java.lang.Object
  + com.sp2.courseWork.BigDemUtils.Utils

public class **Utils**

extends java.lang.Object

utils for library.

**Author:**

Kuzmenko Volodimir

### Constructor Summary

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Utils**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\Utils.html#Utils())() |

### Method Summary

|  |  |
| --- | --- |
| **Methods** | |
| **Modifier and Type** | **Method and Description** |
| static java.math.BigDecimal | [**atanh**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\Utils.html#atanh(java.math.BigDecimal))(java.math.BigDecimal x)  arc tangent hyperbolic . |
| static java.math.BigInteger | [**factorial**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\Utils.html#factorial(java.math.BigInteger))(java.math.BigInteger n)  factorial with BigInteger argument. |
| static java.math.BigInteger | [**factorial**](file:///C:\Users\Vova\workspace%203%20kurs\sp.course_work\doc\com\sp2\courseWork\BigDemUtils\Utils.html#factorial(int))(int n)  factorial for int arguments. |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### Constructor Detail

#### Utils

public Utils()

### Method Detail

#### factorial

public static java.math.BigInteger factorial(int n)

factorial for int arguments.

**Parameters:**

n - int argument

**Returns:**

n!

#### factorial

public static java.math.BigInteger factorial(java.math.BigInteger n)

factorial with BigInteger argument.

**Parameters:**

n - BigInteger argument

**Returns:**

n!

#### atanh

public static java.math.BigDecimal atanh(java.math.BigDecimal x)

arc tangent hyperbolic .

**Parameters:**

x - BigDecimal agrument

**Returns:**

atanh(x)

2.3. ПРИКЛАДИ РЕЗУЛЬТАТУ РОБОТИ.  
Приклад імпорту бібліотеки та використання функцій бібліотеки:

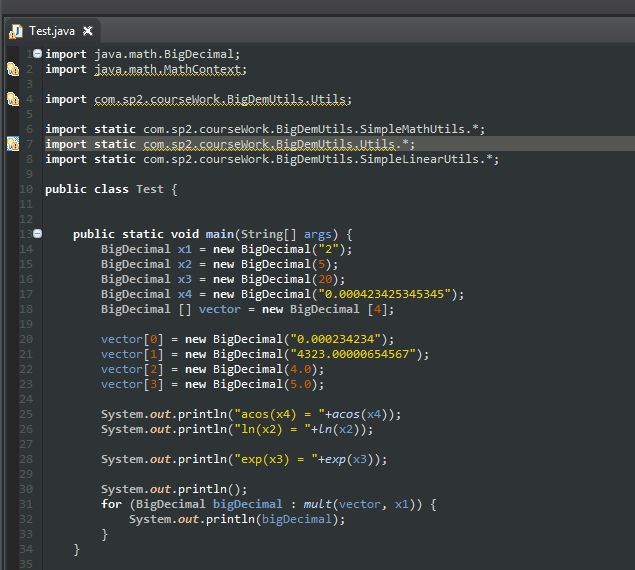


Рисунок 2.3.1 Результати. Використання бібліотеки

Результати роботи функцій відображені у текстовому вигляді, у консолі

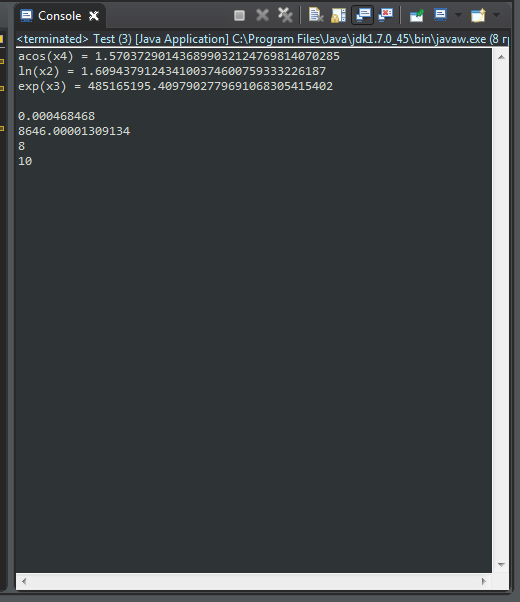


Рисунок 3.2.2. Результати роботи функцій. Результати у текстовому форматі.

Використання інших функцій бібліотеки є еквівалетним тому вважаємо зайвим їх усіх наводити. За деталями використання та результату слід звернутись до документації та тестових прикладів що розміщені у класі UtilsTest.

Висновки

Результатом виконання роботи є бібліотека функцій, що працює з числами підвищеної які у програмі представлені об’єктами класів BigDecimal і BigInteger.

Серед плюсів можна відмітити простоту у використанні бібліотеки та можливість задавити числа трьома різними способами: звичайним числом подвійної точності, об’єктом BigDecimal або BigInteger, або строкою(об’єктом класу String). Проте якщо задавати число третім способом, то необхідно його подати за правилами мови Java.

До недоліків реалізації можна віднести недостатнє, як для бібліотеки, використання абстракції – інтерфейсів та абстрактних класів. Це може негативно вплинути на подальше розширення продукту, проте забезпечує простоту та меншу кількість класів.

Список інформаційних джерел

1. Документація Java™ Platform, Standard Edition 7, http://docs.oracle.com/javase/7/docs/api/
2. Фихтенгольц Г. М. Курс дифференциального и интегрального исчисления, тома I, II. — М.: ФИЗМАТЛИТ, 2001.
3. **Gamma E., Helm R., Johnson R., Vlissides J.** Design Patterns. Elements of Reusable Object-Oriented Software [Addison-Wesley]. - СПб.: Питер, 2001. - 368 с. - ISBN 5-272-00355-1.
4. **Horstmann S. Cay, Cornell G.** Core Java 2. Volume II - Advanced Features [Sun Microsystems Press]. - М.: Издательский дом "Вильямс", 2007. - 1168 с. - ISBN 978-5-8459-1033-2.

Додаток А. Програмний код роботи

/\*

\*

\*/

package com.sp2.courseWork.BigDemUtils;

import java.math.BigDecimal;

import java.util.Arrays;

import static com.sp2.courseWork.BigDemUtils.Utils.\*;

// TODO: Auto-generated Javadoc

/\*\*

\* class for linear utils.

\*

\* @author Kuzmenko Volodimir

\*/

public class SimpleLinearUtils {

/\*\*

\* matrix multiplication with BigDecimal arguments MA\*MB.

\*

\* @param a matrix MA

\* @param b matrix MB

\* @return matrix MA\*MB

\*/

public static BigDecimal[][] mult(final BigDecimal[][] a,

final BigDecimal[][] b) {

int m = a.length;

int n = b[0].length;

if (n!=m)throw new IllegalArgumentException();

else{

int o = b.length;

BigDecimal[][] res = new BigDecimal[m][n];

BigDecimal sum;

for (int i = 0; i < m; i++) {

for (int j = 0; j < n; j++) {

sum = BigDecimal.ZERO;

for (int k = 0; k < o; k++) {

sum = sum.add(a[i][k].multiply(b[k][j]));

}

res[i][j] = sum;

}

}

return res;

}

}

/\*\*

\* matrix multiplication with double arguments MA\*MB.

\*

\* @param a matrix MA

\* @param b matrix MB

\* @return matrix MA\*MB

\*/

public static BigDecimal[][] mult(final double[][] a, final double[][] b) {

BigDecimal[][] ba = new BigDecimal[a.length][];

BigDecimal[][] bb = new BigDecimal[b.length][];

for (int i = 0; i < ba.length; i++) {

ba[i] = new BigDecimal[a[i].length];

bb[i] = new BigDecimal[b[i].length];

for (int j = 0; j < ba[i].length; j++) {

ba[i][j] = new BigDecimal(a[i][j]);

bb[i][j] = new BigDecimal(b[i][j]);

}

}

return mult(ba, bb);

}

/\*\*

\* vector multiplication with double arguments A\*B.

\*

\* @param a vector A

\* @param b vector B

\* @return value result A\*B

\*/

public static BigDecimal mult(final double[] a, final double[] b) {

BigDecimal[] ba = new BigDecimal[a.length];

BigDecimal[] bb = new BigDecimal[b.length];

for (int i = 0; i < bb.length; i++) {

ba[i] = new BigDecimal(a[i]);

bb[i] = new BigDecimal(b[i]);

}

return mult(ba, bb);

}

/\*\*

\* vector multiplication with BigDecimal arguments A\*B.

\*

\* @param a vector A

\* @param b vector B

\* @return value result A\*B

\*/

public static BigDecimal mult(final BigDecimal[] a, final BigDecimal[] b) {

if (a.length!=b.length){

throw new IllegalArgumentException();

}else{

BigDecimal res = BigDecimal.ZERO;

for (int i = 0; i < a.length; i++) {

for (int j = 0; j < a.length; j++) {

res = res.add(a[i].multiply(b[j]));

}

}

return res;

}

}

/\*\*

\* matrix on vector multiplication with BigDecimal arguments MA\*B .

\*

\* @param ma matrix MA

\* @param b vector B

\* @return vector result MA\*B

\*/

public static BigDecimal[] mult(final BigDecimal[][] ma,

final BigDecimal[] b) {

BigDecimal[] res = new BigDecimal[b.length];

for (int i = 0; i < b.length; i++) {

BigDecimal s = BigDecimal.ZERO;

for (int j = 0; j < b.length; j++) {

s = s.add(ma[i][j].multiply(b[j]));

}

res[i] = s;

}

return res;

}

/\*\*

\* matrix on vector multiplication with double arguments MA\*B.

\*

\* @param ma matrix MA

\* @param b vector B

\* @return vector value MA\*B

\*/

public static BigDecimal[] mult(final double[][] ma, final double[] b) {

BigDecimal[][] bma = new BigDecimal[ma.length][ma[0].length];

BigDecimal[] bb = new BigDecimal[b.length];

for (int i = 0; i < bma.length; i++) {

for (int j = 0; j < bma[i].length; j++) {

bma[i][j] = new BigDecimal(ma[i][j]);

}

bb[i] = new BigDecimal(b[i]);

}

return mult(bma, bb);

}

/\*\*

\* vector on matrix multiplication with BigDecimal arguments A\*MB.

\*

\* @param a vector A

\* @param mb matrix MB

\* @return vector result A\*MB

\*/

public static BigDecimal[] mult(final BigDecimal[] a,

final BigDecimal[][] mb) {

BigDecimal[] res = new BigDecimal[mb[0].length];

BigDecimal sum;

for (int j = 0; j < res.length; j++) {

sum = BigDecimal.ZERO;

for (int k = 0; k < res.length; k++) {

sum = sum.add(a[k].multiply(mb[k][j]));

}

res[j] = sum;

}

return res;

}

/\*\*

\* vector on matrix multiplication with double arguments A\*MB .

\*

\* @param a vector A

\* @param mb matrix MD

\* @return vector result A\*MB

\*/

public static BigDecimal[] mult(final double[] a, final double[][] mb) {

BigDecimal[] ba = new BigDecimal[a.length];

BigDecimal[][] bmb = new BigDecimal[mb.length][mb[0].length];

for (int i = 0; i < bmb.length; i++) {

for (int j = 0; j < bmb[0].length; j++) {

bmb[i][j] = new BigDecimal(mb[i][j]);

}

ba[i] = new BigDecimal(a[i]);

}

return mult(ba, bmb);

}

/\*\*

\* matrix sum with BigDecimal arguments MA+MB.

\*

\* @param a matrix MA

\* @param b matrix MB

\* @return matrix MA+MB

\*/

public static BigDecimal[][] add(final BigDecimal[][] a,

final BigDecimal[][] b) {

BigDecimal[][] res = new BigDecimal[a.length][];

for (int i = 0; i < a.length; i++) {

for (int j = 0; j < a[i].length; j++) {

res[i] = new BigDecimal[a[i].length];

res[i][j] = a[i][j].add(b[i][j]);

}

}

return res;

}

/\*\*

\* matrix sum with double arguments MA+MB.

\*

\* @param a matrix MA

\* @param b matrix MB

\* @return matrix MA+MB

\*/

public static BigDecimal[][] add(final double[][] a, final double[][] b) {

BigDecimal[][] ba = new BigDecimal[a.length][];

BigDecimal[][] bb = new BigDecimal[b.length][];

for (int i = 0; i < ba.length; i++) {

ba[i] = new BigDecimal[a[i].length];

bb[i] = new BigDecimal[b[i].length];

for (int j = 0; j < ba[i].length; j++) {

ba[i][j] = new BigDecimal(a[i][j]);

bb[i][j] = new BigDecimal(b[i][j]);

}

}

return add(ba, bb);

}

/\*\*

\* sum matrix and number with BigDecimal arguments MA+b.

\*

\* @param a matrix MA

\* @param b number b

\* @return matrix MA+b

\*/

public static BigDecimal[][] add(final BigDecimal[][] a, final BigDecimal b) {

BigDecimal[][] res = a;

for (int i = 0; i < res.length; i++) {

for (int j = 0; j < res[i].length; j++) {

res[i][j] = res[i][j].add(b);

}

}

return res;

}

/\*\*

\* sum matrices with double arguments MA + b.

\*

\* @param a matrix MA

\* @param b number b

\* @return matrix MA+b

\*/

public static BigDecimal[][] add(final double[][] a, final double b) {

BigDecimal[][] ba = new BigDecimal[a.length][];

for (int i = 0; i < ba.length; i++) {

ba[i] = new BigDecimal[a[i].length];

for (int j = 0; j < ba[i].length; j++) {

ba[i][j] = new BigDecimal(a[i][j]);

}

}

return add(ba, new BigDecimal(b));

}

/\*\*

\* difference matrices with BigDecimal arguments MA-MB.

\*

\* @param a matrix MA

\* @param b matrix MB

\* @return matrix MA-MB

\*/

public static BigDecimal[][] sub(final BigDecimal[][] a,

final BigDecimal[][] b) {

BigDecimal[][] res = new BigDecimal[a.length][];

for (int i = 0; i < a.length; i++) {

for (int j = 0; j < a[i].length; j++) {

res[i] = new BigDecimal[a[i].length];

res[i][j] = a[i][j].subtract(b[i][j]);

}

}

return res;

}

/\*\*

\* difference matrices with BigDecimal arguments MA-MB.

\*

\* @param a matrix MA

\* @param b matrix MB

\* @return matrix MA-MB

\*/

public static BigDecimal[][] sub(final double[][] a, final double[][] b) {

BigDecimal[][] ba = new BigDecimal[a.length][];

BigDecimal[][] bb = new BigDecimal[b.length][];

for (int i = 0; i < ba.length; i++) {

ba[i] = new BigDecimal[a[i].length];

bb[i] = new BigDecimal[b[i].length];

for (int j = 0; j < ba[i].length; j++) {

ba[i][j] = new BigDecimal(a[i][j]);

bb[i][j] = new BigDecimal(b[i][j]);

}

}

return sub(ba, bb);

}

/\*\*

\* matrix on number multiplication with BigDecimal arguments MA\*b.

\*

\* @param a matrix MA

\* @param b value b

\* @return matrix MA\*b

\*/

public static BigDecimal[][] mult(final BigDecimal[][] a, final BigDecimal b) {

BigDecimal[][] res = a;

for (int i = 0; i < res.length; i++) {

for (int j = 0; j < res[i].length; j++) {

res[i][j] = res[i][j].multiply(b);

}

}

return res;

}

/\*\*

\* matrix on number multiplication with double arguments MA\*b.

\*

\* @param a matrix MA

\* @param b value b

\* @return matix MA\*b

\*/

public static BigDecimal[][] mult(final double[][] a, final double b) {

BigDecimal[][] ba = new BigDecimal[a.length][];

for (int i = 0; i < ba.length; i++) {

ba[i] = new BigDecimal[a[i].length];

for (int j = 0; j < ba[i].length; j++) {

ba[i][j] = new BigDecimal(a[i][j]);

}

}

return mult(ba, new BigDecimal(b));

}

/\*\*

\* matrix on number sum with BigDecimal arguments A + B.

\*

\* @param a vector A

\* @param b vector B

\* @return vector A+B

\*/

public static BigDecimal[] add(final BigDecimal[] a, final BigDecimal[] b) {

BigDecimal[] res = new BigDecimal[a.length];

for (int i = 0; i < res.length; i++) {

res[i] = a[i].add(b[i]);

}

return res;

}

/\*\*

\* vectors sum with BigDecimal arguments A+B.

\*

\* @param a vector A

\* @param b vector B

\* @return vector A+B

\*/

public static BigDecimal[] add(final double[] a, final double[] b) {

BigDecimal[] ba = new BigDecimal[a.length];

BigDecimal[] bb = new BigDecimal[b.length];

for (int i = 0; i < ba.length; i++) {

ba[i] = new BigDecimal(a[i]);

bb[i] = new BigDecimal(b[i]);

}

return add(ba, bb);

}

/\*\*

\* vectors sum with double arguments A+b.

\*

\* @param a vector A

\* @param b number b

\* @return vector A+b

\*/

public static BigDecimal[] add(final BigDecimal[] a, final BigDecimal b) {

BigDecimal[] res = a;

for (int i = 0; i < res.length; i++) {

res[i] = res[i].add(b);

}

return res;

}

/\*\*

\* vector and number sum with double arguments A+b.

\*

\* @param a vector A

\* @param b number b

\* @return vector A+b

\*/

public static BigDecimal[] add(final double[] a, final double b) {

BigDecimal[] ba = new BigDecimal[a.length];

for (int i = 0; i < ba.length; i++) {

ba[i] = new BigDecimal(a[i]);

}

return add(ba, new BigDecimal(b));

}

/\*\*

\* difference vectors with BigDecimal arguments A-B.

\*

\* @param a vector A

\* @param b vector B

\* @return vevtor A-B

\*/

public static BigDecimal[] sub(final BigDecimal[] a, final BigDecimal[] b) {

BigDecimal[] res = new BigDecimal[a.length];

for (int i = 0; i < res.length; i++) {

res[i] = a[i].subtract(b[i]);

}

return res;

}

/\*\*

\* difference vectors with double arguments A-A.

\*

\* @param a vector A

\* @param b vector A

\* @return vector A-A

\*/

public static BigDecimal[] sub(final double[] a, final double[] b) {

BigDecimal[] ba = new BigDecimal[a.length];

BigDecimal[] bb = new BigDecimal[b.length];

for (int i = 0; i < ba.length; i++) {

ba[i] = new BigDecimal(a[i]);

bb[i] = new BigDecimal(b[i]);

}

return sub(ba, bb);

}

/\*\*

\* vector on number multiplication with double arguments A\*b.

\*

\* @param a vector A

\* @param b number b

\* @return vector A\*b

\*/

public static BigDecimal[] mult(final BigDecimal[] a, final BigDecimal b) {

BigDecimal[] res = a;

for (int i = 0; i < res.length; i++) {

res[i] = a[i].multiply(b);

}

return res;

}

/\*\*

\* vector on number multiplication with double arguments A\*b.

\*

\* @param a vector A

\* @param b number b

\* @return vector A\*b

\*/

public static BigDecimal[] mult(final double[] a, final double b) {

BigDecimal[] ba = new BigDecimal[a.length];

for (int i = 0; i < ba.length; i++) {

ba[i] = new BigDecimal(a[i]);

}

return mult(ba, new BigDecimal(b));

}

/\*\*

\* vector sorting.

\*

\* @param a vector A

\* @return result

\*/

public static BigDecimal[] sort(BigDecimal[] a) {

BigDecimal[] res = a;

Arrays.sort(res);

return res;

}

/\*\*

\* binary search search in matrix.

\*

\* @param ma the ma

\* @param key the key

\* @return the int

\*/

public static int binarySearch(BigDecimal[][] ma, BigDecimal key) {

return Arrays.binarySearch(ma, key);

}

/\*\*

\* determinant calculation.

\*

\* @param ma matrix

\* @return determinant

\*/

public static BigDecimal det(BigDecimal[][] ma) {

BigDecimal calcResult = BigDecimal.ZERO;

if (ma.length == 2) {

BigDecimal res1 = ma[0][0].multiply(ma[1][1]);

BigDecimal res2 = ma[1][0].multiply(ma[0][1]);

calcResult = res1.subtract(res2, mc);

} else {

int koeff = 1;

for (int i = 0; i < ma.length; i++) {

if (i % 2 == 1) {

koeff = -1;

} else {

koeff = 1;

}

;

BigDecimal b = ma[0][i].multiply(new BigDecimal(koeff));

b = b.multiply(det(getMinor(ma, 0, i)));

calcResult = calcResult.add(b);

}

}

return calcResult;

}

/\*\*

\* minor calculation.

\*

\* @param matrix matrix for calculation

\* @param row number row

\* @param column number column

\* @return the minor

\*/

private static BigDecimal[][] getMinor(BigDecimal[][] matrix, int row,

int column) {

int minorLength = matrix.length - 1;

BigDecimal[][] minor = new BigDecimal[minorLength][minorLength];

int dI = 0;

int dJ = 0;

for (int i = 0; i <= minorLength; i++) {

dJ = 0;

for (int j = 0; j <= minorLength; j++) {

if (i == row) {

dI = 1;

} else {

if (j == column) {

dJ = 1;

} else {

minor[i - dI][j - dJ] = matrix[i][j];

}

}

}

}

return minor;

}

/\*\*

\* search max element in matrix ma.

\*

\* @param ma matrix

\* @return max element

\*/

public static BigDecimal max(BigDecimal[] ma) {

BigDecimal max = ma[0];

for (int i = 1; i < ma.length; i++) {

if (max.compareTo(ma[i]) < 0)

max = ma[i];

}

return max;

}

/\*\*

\* search min element in matrix.

\*

\* @param ma matrix

\* @return min element

\*/

public static BigDecimal min(BigDecimal[] ma) {

BigDecimal min = ma[0];

for (int i = 1; i < ma.length; i++) {

if (min.compareTo(ma[i]) > 0)

min = ma[i];

}

return min;

}

/\*\*

\* matrix transposition.

\*

\* @param ma matrix

\* @return transposition matrix

\*/

public BigDecimal[][] transpone(BigDecimal[][] ma) {

BigDecimal[][] tmp = new BigDecimal[ma[0].length][ma.length];

for (int i = 0; i < ma[0].length; i++) {

for (int j = 0; j < ma.length; j++) {

tmp[i][j] = ma[j][i];

}

}

return tmp;

}

}

/\*

\*

\*/

package com.sp2.courseWork.BigDemUtils;

import java.math.BigDecimal;

import java.math.BigInteger;

import java.math.MathContext;

import java.util.HashMap;

// TODO: Auto-generated Javadoc

/\*\*

\* utils for library.

\*

\* @author Kuzmenko Volodimir

\*/

public class Utils {

/\*\* MathContext to config library. \*/

static MathContext mc = new MathContext((int) ((Math.pow(2, 7))));

/\*\* cache data for factorial. \*/

static HashMap<Integer,BigInteger> cache = new HashMap<Integer,BigInteger>();

/\*\*

\* factorial for int arguments.

\*

\* @param n int argument

\* @return n!

\*/

public static BigInteger factorial(int n)

{

BigInteger ret;

if (n == 0) return BigInteger.ONE;

if (null != (ret = cache.get(n))) return ret;

ret = BigInteger.valueOf(n).multiply(factorial(n-1));

cache.put(n, ret);

return ret;

}

/\*\*

\* factorial with BigInteger argument.

\*

\* @param n BigInteger argument

\* @return n!

\*/

public static BigInteger factorial(BigInteger n) {

return factorial(n.intValue());

}

/\*\*

\* arc tangent hyperbolic .

\*

\* @param x BigDecimal agrument

\* @return atanh(x)

\*/

public static BigDecimal atanh(BigDecimal x) {

BigDecimal res = BigDecimal.ZERO;

BigDecimal b;

for (int k = 0; k < 100; k++) {

b = x.pow(2 \* k + 1);

b = b.divide(new BigDecimal(2 \* k + 1), mc);

res = res.add(b);

}

return res;

}

/\*\* PI value. \*/

static final BigDecimal PI = new BigDecimal(

"3.14159265358979323846264338327950288419716939937510582097494459230781640628620"

+ "899862803482534211706798214808651328230664709384460955058223172535940812848111"

+ "745028410270193852110555964462294895493038196442881097566593344612847564823378"

+ "678316527120190914564856692346034861045432664821339360726024914127372458700660"

+ "631558817488152092096282925409171536436789259036001133053054882046652138414695"

+ "194151160943305727036575959195309218611738193261179310511854807446237996274956"

+ "735188575272489122793818301194912983367336244065664308602139494639522473719070"

+ "217986094370277053921717629317675238467481846766940513200056812714526356082778"

+ "577134275778960917363717872146844090122495343014654958537105079227968925892354"

+ "201995611212902196086403441815981362977477130996051870721134999999837297804995"

+ "105973173281609631859502445945534690830264252230825334468503526193118817101000"

+ "313783875288658753320838142061717766914730359825349042875546873115956286388235"

+ "378759375195778185778053217122680661300192787661119590921642019893809525720106"

+ "548586327886593615338182796823030195203530185296899577362259941389124972177528"

+ "347913151557485724245415069595082953311686172785588907509838175463746493931925"

+ "506040092770167113900984882401285836160356370766010471018194295559619894676783"

+ "744944825537977472684710404753464620804668425906949129331367702898915210475216"

+ "205696602405803815019351125338243003558764024749647326391419927260426992279678"

+ "235478163600934172164121992458631503028618297455570674983850549458858692699569"

+ "092721079750930295532116534498720275596023648066549911988183479775356636980742"

+ "654252786255181841757467289097777279380008164706001614524919217321721477235014",

mc);

/\*\* exponent value. \*/

static final BigDecimal E = new BigDecimal(

"2.71828182845904523536028747135266249775724709369995957496696762772407663035354"

+ "759457138217852516642742746639193200305992181741359662904357290033429526059563"

+ "073813232862794349076323382988075319525101901157383418793070215408914993488416"

+ "750924476146066808226480016847741185374234544243710753907774499206955170276183"

+ "860626133138458300075204493382656029760673711320070932870912744374704723069697"

+ "720931014169283681902551510865746377211125238978442505695369677078544996996794"

+ "686445490598793163688923009879312773617821542499922957635148220826989519366803"

+ "318252886939849646510582093923982948879332036250944311730123819706841614039701"

+ "983767932068328237646480429531180232878250981945581530175671736133206981125099"

+ "618188159304169035159888851934580727386673858942287922849989208680582574927961"

+ "048419844436346324496848756023362482704197862320900216099023530436994184914631"

+ "409343173814364054625315209618369088870701676839642437814059271456354906130310"

+ "720851038375051011574770417189861068739696552126715468895703503540212340784981"

+ "933432106817012100562788023519303322474501585390473041995777709350366041699732"

+ "972508868769664035557071622684471625607988265178713419512466520103059212366771"

+ "943252786753985589448969709640975459185695638023637016211204774272283648961342"

+ "251644507818244235294863637214174023889344124796357437026375529444833799801612"

+ "549227850925778256209262264832627793338656648162772516401910590049164499828931");

}

/\*

\*

\*/

package com.sp2.courseWork.BigDemUtils;

import java.math.BigDecimal;

import java.math.BigInteger;

import static com.sp2.courseWork.BigDemUtils.Utils.\*;

// TODO: Auto-generated Javadoc

/\*\*

\* The Class SimpleMathUtils.

\*

\* @author Kuzmenko Volodimir

\*/

public class SimpleMathUtils {

/\*\*

\* sine.

\*

\* @param x BigDecimal argument

\* @return sin(x)

\*/

public static BigDecimal sin(final BigDecimal x) {

BigDecimal res = new BigDecimal(BigInteger.ZERO,mc);

BigDecimal d;

BigInteger fact =BigInteger.ONE;

BigInteger minusOne = BigInteger.ONE;

minusOne = minusOne.multiply(new BigInteger("-1"));

for (int k = 0; k < 60; k++) {

d = x.pow(2\*k+1);

fact = factorial(2\*k+1).multiply(minusOne.pow(k));

d = d.divide(new BigDecimal(fact,mc), mc);

res = res.add(d,mc);

}

return res;

}

/\*\*

\* sine.

\*

\* @param x double argument

\* @return sin(x)

\*/

public static BigDecimal sin(final double x) {

return sin(new BigDecimal(x));

}

/\*\*

\* sine.

\*

\* @param x String argument

\* @return sin(x)

\*/

public static BigDecimal sin(final String x) {

return sin(new BigDecimal(x));

}

/\*\*

\* cosine.

\*

\* @param x BigDecimal argument

\* @return cos(x)

\*/

static public BigDecimal cos(final BigDecimal x) {

BigDecimal res = new BigDecimal(BigInteger.ZERO);

BigDecimal d;

BigInteger fact =BigInteger.ONE;

BigInteger minusOne = BigInteger.ONE;

minusOne = minusOne.multiply(new BigInteger("-1"));

for (int k = 0; k < 60; k++) {

d = x.pow(2\*k,mc);

fact = factorial(2\*k).multiply(minusOne.pow(k));

d = d.divide(new BigDecimal(fact), mc);

res = res.add(d,mc);

}

return res;

}

/\*\*

\* cosine.

\*

\* @param x BigDecimal argument

\* @return cos(x)

\*/

public static BigDecimal cos(final double x) {

return sin(new BigDecimal(x,mc));

}

/\*\*

\* cosine.

\*

\* @param x String argument

\* @return cos(x)

\*/

public static BigDecimal cos(final String x) {

return cos(new BigDecimal(x,mc));

}

/\*\*

\* tangent.

\*

\* @param x BigDecimal argument

\* @return tan(x)

\*/

public static BigDecimal tan(final BigDecimal x) {

BigDecimal b = sin(x);

b = b.divide(cos(x),mc);

return b;

}

/\*\*

\* tangents.

\*

\* @param x BigDecimal argument

\* @return sin(x)

\*/

public static BigDecimal tan(final double x) {

return tan(new BigDecimal(x,mc));

}

/\*\*

\* tangent.

\*

\* @param x String argument

\* @return sin(x)

\*/

public static BigDecimal tan(final String x) {

return tan( new BigDecimal(x,mc));

}

/\*\*

\* cotangent.

\*

\* @param x BigDecimal argument

\* @return ctan(x)

\*/

public static BigDecimal ctan(final BigDecimal x) {

BigDecimal b = cos (x);

b = b.divide(sin(x),mc);

return b;

}

/\*\*

\* cotangent.

\*

\* @param x BigDecimal argument

\* @return ctan(x)

\*/

public static BigDecimal ctan(final double x) {

return ctan(new BigDecimal(x,mc));

}

/\*\*

\* cotangent.

\*

\* @param x String argument

\* @return ctan(x)

\*/

public static BigDecimal ctan(final String x) {

return ctan(new BigDecimal(x,mc));

}

/\*\*

\* arc sine.

\*

\* @param x BigDecimal argument

\* @return asin(x)

\*/

public static BigDecimal asin(final BigDecimal x) {

if (x.abs().compareTo(BigDecimal.ONE)==1){

throw new IllegalArgumentException("|x|<1");

}else{

BigInteger fact1;

BigInteger fact2;

BigDecimal res=BigDecimal.ZERO;

BigDecimal b;

BigDecimal g;

BigInteger two = new BigInteger("2");

for (int k = 0; k <70 ; k++) {

fact1 = factorial(2\*k);

fact2 = factorial(k);

fact2 = fact2.pow(2);

fact2 = fact2.multiply(new BigInteger(Integer.toString(2\*k+1)));

fact2 = fact2.multiply(two.pow(2\*k));

b = new BigDecimal(fact1,mc);

b = b.divide(new BigDecimal(fact2,mc), mc);

g = x.pow(2\*k+1,mc);

b = b.multiply(g, mc);

res = res.add(b,mc);

}

return res;

}

}

/\*\*

\* arc sine.

\*

\* @param x BigDecimal argument

\* @return asin(x)

\*/

public static BigDecimal asin(final double x) {

return asin(new BigDecimal(x,mc));

}

/\*\*

\* arc sine.

\*

\* @param x String argument

\* @return asin(x)

\*/

public static BigDecimal asin(final String x) {

return asin(new BigDecimal(x,mc));

}

/\*\*

\* arc cosine.

\*

\* @param x BigDecimal argument

\* @return acos(x)

\*/

public static BigDecimal acos(final BigDecimal x) {

if (x.abs().compareTo(BigDecimal.ONE)==1){

throw new IllegalArgumentException("|x|<1");

}else{

BigDecimal pito2 = PI.divide(new BigDecimal(2));

return pito2.subtract(asin(x), mc);

}

}

/\*\*

\* arc cosine.

\*

\* @param x BigDecimal argument

\* @return cosine(x)

\*/

public static BigDecimal acos(final double x) {

return acos(new BigDecimal(x,mc));

}

/\*\*

\* arc cosine.

\*

\* @param x String argument

\* @return acos(x)

\*/

public static BigDecimal acos(final String x) {

return acos(new BigDecimal(x,mc));

}

/\*\*

\* arc tangent.

\*

\* @param x BigDecimal argument

\* @return atan(x)

\*/

public static BigDecimal atan(final BigDecimal x) {

BigDecimal b;

BigDecimal res = BigDecimal.ZERO;

BigDecimal g;

for (int k = 0; k < 60; k++) {

b = new BigDecimal((2\*k+1)\*Math.pow(-1, k));

g = x.pow(2\*k+1,mc);

g = g.divide(b,mc);

res = res.add(g,mc);

}

return res;

}

/\*\*

\* arc tangent.

\*

\* @param x BigDecimal argument

\* @return atan(x)

\*/

public static BigDecimal atan(final double x) {

return actan(new BigDecimal(x,mc));

}

/\*\*

\* arc tangent.

\*

\* @param x String argument

\* @return atan(x)

\*/

public static BigDecimal atan(final String x) {

return actan(new BigDecimal(x,mc));

}

/\*\*

\* arc cotangent.

\*

\* @param x BigDecimal argument

\* @return actan(x)

\*/

public static BigDecimal actan(final BigDecimal x) {

BigDecimal pito2 = PI.divide(new BigDecimal(2));

return pito2.subtract(atan(x), mc);

}

/\*\*

\* arc cotangent.

\*

\* @param x BigDecimal argument

\* @return atan(x)

\*/

public static BigDecimal actan(final double x) {

return actan(new BigDecimal(x,mc));

}

/\*\*

\* arc cotangent.

\*

\* @param x String argument

\* @return atan(x)

\*/

public static BigDecimal actan(final String x) {

return atan(new BigDecimal(x,mc));

}

/\*\*

\* hyperbolic sine.

\*

\* @param x BigDecimal argument

\* @return sinh(x)

\*/

public static BigDecimal sinh(final BigDecimal x) {

BigDecimal res = exp(x);

res = res.subtract(exp(x.multiply(new BigDecimal(-1))), mc);

res = res.divide(new BigDecimal(2), mc);

return res;

}

/\*\*

\* hyperbolic sine.

\*

\* @param x BigDecimal argument

\* @return sinh(x)

\*/

public static BigDecimal sinh(final double x) {

return sinh(new BigDecimal(x,mc));

}

/\*\*

\* hyperbolic sine.

\*

\* @param x String argument

\* @return sin(x)

\*/

public static BigDecimal sinh(final String x) {

return sinh(new BigDecimal(x,mc));

}

/\*\*

\* hyperbolic cosine.

\*

\* @param x BigDecimal argument

\* @return cosh(x)

\*/

public static BigDecimal cosh(final BigDecimal x) {

BigDecimal res = exp(x);

res = res.add(exp(x.multiply(new BigDecimal(-1))), mc);

res = res.divide(new BigDecimal(2), mc);

return res;

}

/\*\*

\* hyperbolic cosine.

\*

\* @param x BigDecimal argument

\* @return cosh(x)

\*/

public static BigDecimal cosh(final double x) {

return cos(new BigDecimal(x,mc));

}

/\*\*

\* hyperbolic cosine.

\*

\* @param x String argument

\* @return cosh(x)

\*/

public static BigDecimal cosh(final String x) {

return cos(new BigDecimal(x,mc));

}

/\*\*

\* hyperbolic tangent.

\*

\* @param x BigDecimal argument

\* @return tanh(x)

\*/

public static BigDecimal tanh(final BigDecimal x) {

BigDecimal b = sinh(x);

b = b.divide(cosh(x), mc);

return b;

}

/\*\*

\* hyperbolic tangent.

\*

\* @param x BigDecimal argument

\* @return tanh(x)

\*/

public static BigDecimal tanh(final double x) {

return tanh(new BigDecimal(x,mc));

}

/\*\*

\* hyperbolic tangent.

\*

\* @param x String argument

\* @return tanh(x)

\*/

public static BigDecimal tanh(final String x) {

return tanh(new BigDecimal(x,mc));

}

/\*\*

\* hyperbolic cotangent.

\*

\* @param x BigDecimal argument

\* @return s(x)

\*/

public static BigDecimal ctanh(final BigDecimal x) {

return BigDecimal.ONE.divide(tanh(x), mc);

}

/\*\*

\* hyperbolic cotangent.

\*

\* @param x double argument

\* @return ctanh(x)

\*/

public static BigDecimal ctanh(final double x) {

return ctanh(new BigDecimal(x,mc));

}

/\*\*

\* hyperbolic cotangent.

\*

\* @param x BigDecimal argument

\* @return sin(x)

\*/

public static BigDecimal ctanh(final String x) {

return exp(new BigDecimal(x,mc));

}

/\*\*

\* exponent with double arguments.

\*

\* @param x double argument

\* @return exp(x)

\*/

public static BigDecimal exp(double x) {

return exp(new BigDecimal(x,mc));

}

/\*\*

\* exponent with BigDecimal agruments.

\*

\* @param x BigDecimal argument

\* @return exp(x)

\*/

public static BigDecimal exp(BigDecimal x) {

BigDecimal d;

BigDecimal res = BigDecimal.ZERO;

for (int k = 0; k < 60; k++) {

d = x.pow(k,mc);

d = d.divide(new BigDecimal(factorial(k)), mc);

res = res.add(d, mc);

}

return res;

}

/\*\*

\* exponent with String arguments.

\*

\* @param x String argument

\* @return exp(x)

\*/

public static BigDecimal exp(String x) {

return exp(new BigDecimal(x,mc));

}

/\*\*

\* natural logarithm.

\*

\* @param x BigDecimal argument

\* @return ln(x)

\*/

public static BigDecimal ln(BigDecimal x) {

BigDecimal b = x.subtract(BigDecimal.ONE, mc);

b = b.divide(x.add(BigDecimal.ONE),mc);

BigDecimal res = BigDecimal.ZERO;

BigDecimal d;

for (int k = 0; k < 90; k++) {

d = BigDecimal.ONE.divide(new BigDecimal(2\*k+1),mc);

d = d.multiply(b.pow(2\*k+1),mc);

res = res.add(d,mc);

}

return res.multiply(new BigDecimal(2),mc);

}

/\*\*

\* natural logarithm with double arguments.

\*

\* @param x double argument

\* @return ln(x)

\*/

public static BigDecimal ln(double x) {

return ln(new BigDecimal(x));

}

/\*\*

\* natural logarithm.

\*

\* @param x String argument

\* @return ln(x)

\*/

public static BigDecimal ln(String x) {

return ln(new BigDecimal(x,mc));

}

/\*\*

\* common logarithm.

\*

\* @param x BigDecimal argument

\* @return sin(x)

\*/

public static BigDecimal lg(BigDecimal x) {

BigDecimal res = ln(x);

BigDecimal b = ln(10.0);

return res.divide(b, mc);

}

/\*\*

\* common logarithm.

\*

\* @param x BigDecimal argument

\* @return lg(x)

\*/

public static BigDecimal lg(double x) {

return lg(new BigDecimal(x,mc));

}

/\*\*

\* common logarithm.

\*

\* @param x String argument

\* @return lg(x)

\*/

public static BigDecimal lg(String x) {

return (new BigDecimal(x,mc));

}

/\*\*

\* logarithm.

\*

\* @param x BigDecimal argument

\* @param a the a

\* @return log(x)

\*/

public static BigDecimal log(BigDecimal x, BigDecimal a) {

BigDecimal res = ln(x);

BigDecimal b = ln(a);

return res.divide(b, mc);

}

/\*\*

\* logarithm.

\*

\* @param x BigDecimal argument

\* @param a the a

\* @return log(x)

\*/

public static BigDecimal log(double x, double a) {

return log(new BigDecimal(x), new BigDecimal(x));

}

/\*\*

\* logarithm.

\*

\* @param x String argument

\* @param a the a

\* @return log(x)

\*/

public static BigDecimal log(String x, String a) {

return log(new BigDecimal(x,mc), new BigDecimal(a,mc));

}

/\*\*

\* square root.

\*

\* @param x BigDecimal argument

\* @return sqrt(x)

\*/

public static BigDecimal sqrt(BigDecimal x)

{

if (x.signum() < 0) {

throw new IllegalArgumentException("x < 0");

}

BigInteger n = x.movePointRight(mc.getPrecision() << 1).toBigInteger();

int bits = (n.bitLength() + 1) >> 1;

BigInteger ix = n.shiftRight(bits);

BigInteger ixPrev;

do {

ixPrev = ix;

ix = ix.add(n.divide(ix)).shiftRight(1);

Thread.yield();

} while (ix.compareTo(ixPrev) != 0);

return new BigDecimal(ix,mc.getPrecision());

}

/\*\*

\* square root.

\*

\* @param x double argument

\* @return sqrt(x)

\*/

public static BigDecimal sqrt(double x) {

return sqrt(new BigDecimal(x,mc));

}

/\*\*

\* square root.

\*

\* @param x String argument

\* @return sqrt(x)

\*/

public static BigDecimal sqrt(String x) {

return sqrt(new BigDecimal(x,mc));

}

}

/\*

\*

\*/

package com.sp2.courseWork.BigDemUtils;

import java.math.BigDecimal;

import java.util.Random;

import static com.sp2.courseWork.BigDemUtils.Utils.\*;

import static com.sp2.courseWork.BigDemUtils.SimpleMathUtils.\*;

// TODO: Auto-generated Javadoc

/\*\*

\* Class for store erf, factorial nad random function.

\*

\* @author Kuzmenko Volodimir

\*/

public class OtherMathUtils {

/\*\*

\* error function.

\*

\* @param x argument function

\* @return result value

\*/

public static BigDecimal erf (final BigDecimal x){

BigDecimal b = new BigDecimal(2);

b = b.divide(sqrt(PI),mc);

BigDecimal d1, d2,d3,res;

res = BigDecimal.ZERO;

for (int n = 0; n < 100; n++) {

d1 = x.divide(new BigDecimal(2\*n+1),mc);

d2 = BigDecimal.ONE;

for (int k = 1; k <= n; k++) {

d3 = x.pow(2).multiply(new BigDecimal(-1));

d2 = d2.multiply(d3.divide(new BigDecimal(k),mc),mc);

}

d1 = d1.multiply(d2,mc);

res =res.add(d1,mc);

}

res = res.multiply(b,mc);

return res;

}

/\*\*

\* function for uniformly distribution between 0 and 1.

\*

\* @return random number

\*/

public static BigDecimal nextRandom(){

Random random = new Random();

return new BigDecimal(random.nextDouble(),mc);

}

/\*\*

\* function for standard normal distribution .

\*

\* @return random number for this distribution

\*/

public static BigDecimal nextGaussian(){

Random random = new Random();

return new BigDecimal(random.nextGaussian());

}

}