LCO

Laboratory Calculations Optimizator

Alpha 1.3

1 Введение

Laboratory Calculations Optimizator - программа-калькулятор, которая позволяет считать значение любых формул и их погрешность, строить графики, а также выполнять ряд операций, которые необходимы для оформления отчета по лабораторной работе (например, метод наименьших квадратов).

2 Запуск программы

Для запуска программы необходима установленная Java, ее можно скачать с официального сайта: *https://java.com/ru/download/*.

2.1 Windows

Запуск программы LCO на Windows очень прост, вам просто нужно запустить файл с программой LCO.jar. Далее откроется меню, в котором можно либо создать новый файл, либо открыть уже существующий.

Рис. 1: Интерфейс программы, главное меню

2.2 Linux

Для запуска программы нужно в терминале перейти в директорию, где находится файл с программой и написать команду: *java -jar LCO.jar*

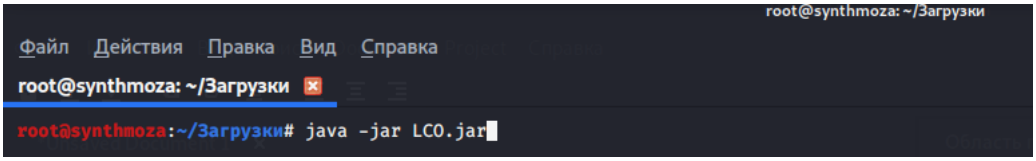

Рис. 2: Запуск из терминала Linux

3 Использование программы

3.1 Переменные и массивы

Как и любой другой язык программирования, язык LCO позволяет создавать переменные, присваивать им значения, выполнять арифметические операции $(+, -, *, /, sin, cos, ln)$ с ними и указывать их погрешности. Тип переменных только один - действительные числа. Имена переменных могут быть любые(даже на русском языке), но не могут начинаться с цифр.

Source code:

```
a = 4.5;
//объявление переменной a со значением 4.5
b = 5.32 # 0.03;
// переменная b с погрешностью 0.03
колличествоИзмереній = a + b * ln(2) + 2^3 * b;
//выражение
```

Рис. 3: Работа с переменными

Так же присутствуют массивы, которые объявляются следующим образом:

Source code:

```
F = [2.3, 4.5, 4.3, 8.2];
//объявление массива F длиной 4
```

Рис. 4: Объявление массива

Чтобы обозначить погрешность элементов массива, нужно либо указать их в объявлении массива, либо в случае, когда погрешности элементов массива одинаковые, можно указать погрешность проще:

Чтобы вывести значение переменной, массива или функции, перед необходимой строкой нужно поставить знак доллара "$". Чтобы сделать перенос на новую строку и вывести строку, нужно поставить два доллара "$$".

Source code:

```
result = [1.3, 2.4, 5.6, 8.7];
$ result = result # 0.25;
// вывод значений массива result

length = [5.6, 5.56, 5.46, 5.64];
$$ length = length;
//чтобы вывести переменную в любой момент, достаточно
//написать ее присваивание самой себе с долларом

mass = 3.41 # 0.23;
velocity = 20.34 # 0.67;
$ k_energy = (mass * velocity^2) / 2;
//вывод конечной формулы с полученной погрешностью
// по методу наименьших квадратов
```

Рис. 5: Вывод переменных на экран, код программы

Compiled code:

```
result = [1.3 # 0.25, 2.4 # 0.25, 5.6 # 0.25, 8.7 # 0.25];
length = [5.6, 5.56, 5.46, 5.64];

k_energy = (mass * velocity ^ 2) / 2 = 710 # 60;
```

Рис. 6: Вывод переменных на экран, результат

3.2 Особенности работы с массивами

Для массивов так же доступны основные арифметические операции, но они работают немного по-другому. Результатом операции является массив длины наибольшего из исходных массивов, операции выполняются по-элементно. Если массивы разных длин, то операция все равно может быть выполнена с особым правилом циклического сдвига:

Source code:

```
x = [1, 2, 5, 6, 7];
y = [4, 2];
// массивы разных длин
$ z = x + y;
// результирующий массив имеет максимальную из
// длин данных массивов.
// Для всех элементов, у которых есть соответствующая
// пара в другом массиве, выполянется заданная операция,
// а для остальных операция будет выполняться циклично,
// начиная с первого элемента наименьшего массива
```

Рис. 7: Операции с массивами, код

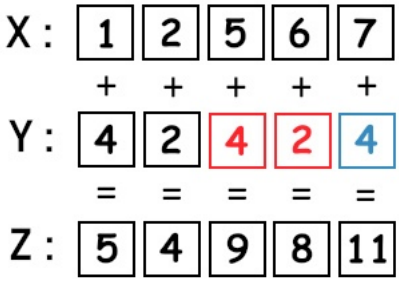

Рис. 8: Операции с массивами разных длин, визуализация

Compiled code:

```
z = [5, 4, 9, 8, 11];
```

Рис. 9: Операции с массивами, результат

Замечание: если появляется операция деления или возведения в степень, то все то, что стоит после нее, идет в знаменатель или в показатель степени. Пример:

Source code:

```
a = 16;
b = 4;
c = 2;
$ d = 1 / a * b * c;
// все, что идет после знака деления идет в знаменатель
$d = (1 / a) * b * c;

$ d = a ^ b / c;
// все, что идет после знака степени, идет в показатель
// степени
$ d = (a ^ b) / c;
```

Рис. 10: Замечание, код программы

Compiled code:

```
d = 1 / (a * (b * c)) = 0.0078125;
d = (1 / a) * (b * c) = 0.5000;
d = a ^ (b / c) = 256;
d = a ^ b / c = 32768;
```

Рис. 11: Замечание, результат

4 Функции

В языке LCO предусмотрен набор функция для удобной обработки результатов лабораторной работы. Сейчас мы их рассмотрим.

4.1 leastSquares - метод наименьших квадратов

Данная функция работает по методу наименьших квадратов - получает на вход набор значений(два массива), и по методу наименьших квадратов находит наилучшие коэффициенты прямой $y = a + b * x$, создавая соответствующие переменные a и b (если таковые уже существуют, то функциях их перепишет).

leastSquares(x, y):

- x - массив величин, соответствущий оси абсциисс.
- y - массив величин, соответствующий оси ординат.

Source code:

```
mass = [123.34, 145.65, 178.45, 195.43];
volume = [34.54, 45.23, 59.89, 78.38];
$ leastSquares(mass, volume);
// получение прямой зависимости массы от обьема с
// помощью метода наименьших квадратов
```

Рис. 12: Пример использования функции leastSquares, код программы

Compiled code:

```
a = -38 # 1;
b = 0.58 # 0.05;
```

Рис. 13: Пример использования функции leastSquares, результат

4.2 makeGraph - построение графиков

Данная функция позволяет построить линии с различными компонентами. Вывод изображения происходит в файл в папку с файлом, который вы редактируете в данный момент (по умолчанию формат *.png*). Программа расчитана только на положительные значения.

makeGraph([arrX, arrY, type]..., fileName, axisXname, axisYname)

- arrX - массив величин, соответствущий оси абсциисс.
- arrY - массив величин, соответствующий оси ординат.
- type - тип линии, указывается в виде строки, состоящей из ВСЕХ необходимых ключевых слов типов:

points - ставит точки на плоскости;

line - соединяет точки;

infl - рисует погрешность,

т.е. примером параметра *type* могут быть: *"points"* - построить только точки, *"pointsline"* - построить точки и соединить их линиями, *"pointslineinfl"* - построить точки, соединить их линиями и изобразить погрешность.

•[arrX, arrY, type]... - для построения нескольких линий данные параметры можно указывать несколько раз.

•fileName - имя выходного файла.

•axisXname - название оси абсцисс.

•axisYname - название оси ординат.

Source code:

```
mass = [123.34, 145.65, 178.45, 225.43];
volume = [34.54, 45.23, 59.89, 78.38];
makeGraph(mass, volume, "pointsline", "output.png", "m", "V"
);
//построение графика с соединенными точками и
// осями "m" и "V", который сохранится как
// изображение с именем "output.png"
```

Рис. 14: Пример использования makeGraph, код программы

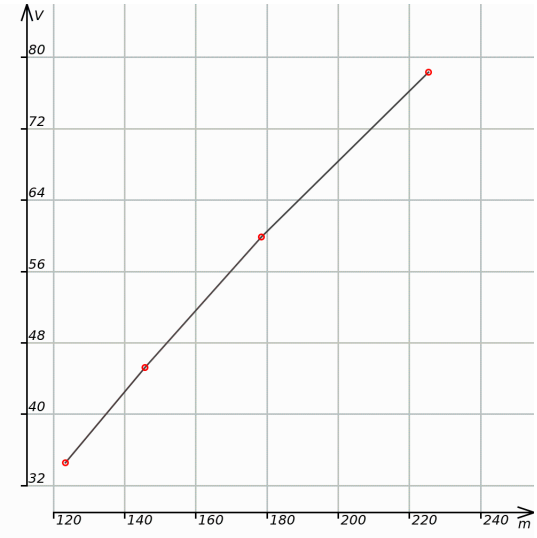

Рис. 15: Пример использования makeGraph, график

4.3 useExp() / disableExp() - эспоненциальная запись

Язык LCO поддерживает по умолчанию запись чисел как в обычном виде, так и в экспоненциальной записи. Но для того, чтобы числа выводились в научном формате, предусмотрены

две функции: *useExp()* и *disableExp()*, которые, соответственно, включают вывод в научном формате и выключают.

Source code:

```
$ a = 5.31e10 # 1e5;
useExp();
$ b = 6.02e24 #1.05e10;
disableExp();
$ c = 34.5e19;
```

Рис. 16: Научный формат, код программы

Compiled code:

```
a = 53100000000 # 100000;
b = 6.0200000000000E+24 # 10.5E+9;
c = 345000000000000000000;
```

Рис. 17: Научный формат, результат

5 Примеры использования

5.1 Пример 1. Математический маятник.

Пусть нам нужно посчитать период колебаний математического маятника по известной формуле:

$$T = 2\pi\sqrt{\frac{L}{g}}, \tag{1}$$

Мы измерили длину нити с некоторой погрешностью, т.е. $L = 10 \pm 1$ м, также известно ускорение свободного падения: $g = 9.8 \pm 0.1$ м/c^2, теперь необходимо посчитать погрешность значения периода T и после округлить период до первой значащей цифры погрешности.

Source code:

```
L = 10 # 1;
g = 9.8 # 0.1;
PI = 3.1415 # 0.0001;

$ T = 2 * PI * (L / g) ^ 0.5;
```

Рис. 18: Пример 1, код программы

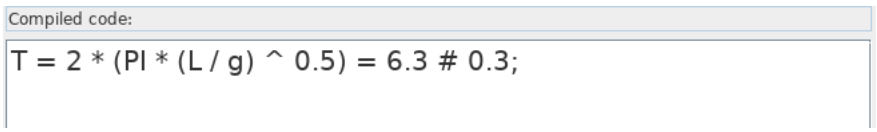

Рис. 19: Пример 1, результат

5.2 Пример 2. Модуль Юнга

Рассмотрим более сложный пример. Необходимо посчитать значение модуля Юнга (Лабораторная работа 1.3.1 первого семестра) по формуле

$$E = \frac{4kL}{\pi d^2}, \tag{2}$$

где l, d - характеристики объекта, k - коэффициент Гука, т.е. $k = F/dl$, более того, имеется ряд значений $F = (f_1, f_2, ...)$ и $dl = (dl_1, dl_2, ...)$ таких, что $k = f_i/dl_i$, где f_i - i-ая сила, а dl_i - растяжение, которое соответствует этой силе.

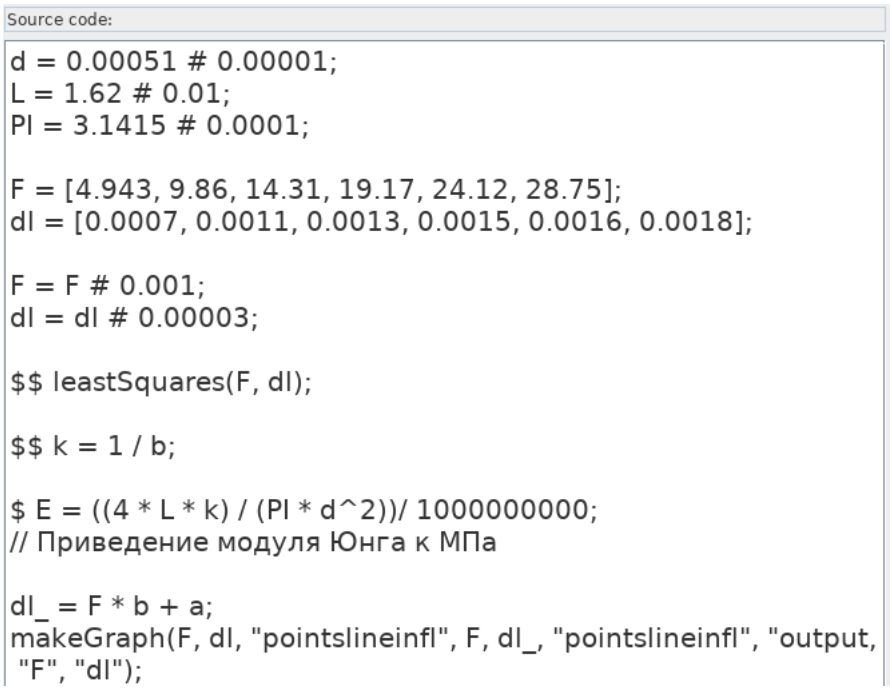

Рис. 20: Пример 2, код

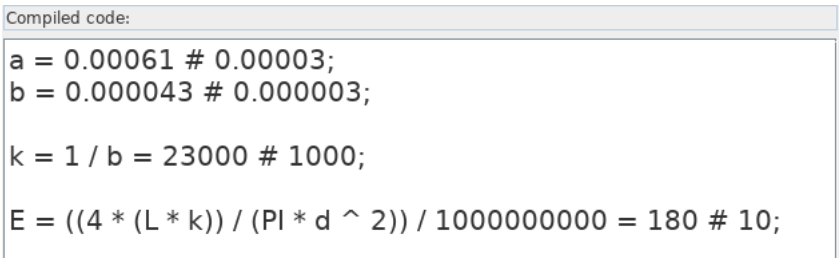

Рис. 21: Пример 2, результат

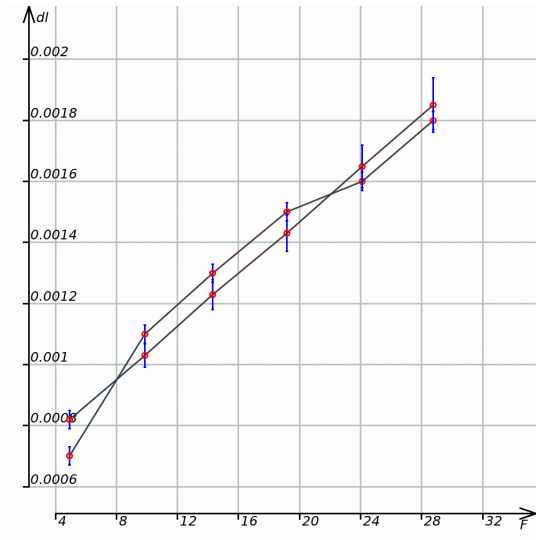

Рис. 22: Пример 2, график

6 В заключении

Если вы обнаружили ошибки в выполнении программы или в документации к ней, любую неточность в вычислениях или есть идеи новых полезных функций в программу, то пишите сюда: *trofimenko.tia@phystech.edu* или сюда *https://vk.com/timatttttt*.