

Министерство образования и науки Российской Федерации
Рязанский Государственный Радиотехнический Университет

М.Б. Никифоров, М.В. Акинин, А.В. Соколова

Теория планирования эксперимента

Методические указания
к лабораторным работам
по курсу «Теория планирования эксперимента»



Рязань 2014

Содержание

1	Методические указания	4
1.1	Лабораторная работа № 1. Эксперимент. Метод наименьших квадратов. Язык программирования R	4
1.1.1	Цель работы	4
1.1.2	Теоретическая часть	4
1.1.2.1	Исследуемый объект	4
1.1.2.2	Инструментарий проведения эксперимента	4
1.1.2.2.1	Руководство оператора	5
1.1.2.3	Обработка результатов эксперимента. Построение регрессии методом наименьших квадратов. Язык программирования R	13
1.1.2.3.1	Язык программирования R	13
1.1.2.3.2	Построение регрессии методом наименьших квадратов	13
1.1.2.3.3	Использование языка программирования R для обработки результатов эксперимента	14
1.1.2.3.4	Пример расчета оптимального набора значений параметров функционирования кластера	18
1.1.3	Задание	23
1.1.4	Требования к оформлению отчета по лабораторной работе	24
1.2	Лабораторная работа № 2. Полный и дробный факторные эксперименты	26
1.2.1	Цель работы	26
1.2.2	Теоретическая часть	26
1.2.2.1	Полный факторный эксперимент	26
1.2.2.2	Дробный факторный эксперимент	28
1.2.2.3	Решение системы линейных уравнений средствами языка программирования R	28
1.2.3	Задание	33
1.2.4	Требования к оформлению отчета по лабораторной работе	35
1.3	Лабораторная работа № 3. Линейная регрессионная модель	37
1.3.1	Цель работы	37
1.3.2	Теоретическая часть	37
1.3.2.1	Линейная регрессионная модель	37

1.3.2.1.1	Расчет параметров линейной регрессионной модели средствами языка программирования R	38
1.3.2.2	Демонстрация построения регрессии	38
1.3.3	Задание	41
1.3.4	Требования к оформлению отчета по лабораторной работе	42
1.4	Лабораторная работа № 4.	
	Симплекс - метод планирования эксперимента	44
1.4.1	Цель работы	44
1.4.2	Теоретическая часть	44
1.4.2.1	Симплекс - метод	44
1.4.2.2	Задача линейного программирования	44
1.4.2.3	Усиленная постановка задачи	46
1.4.2.4	Алгоритм симплекс - метода	46
1.4.2.5	Пример реализации симплекс-метода	48
1.4.2.6	Решение системы уравнений симплекс - метода средствами языка программирования R	54
1.4.3	Задание	57
1.4.4	Требования к оформлению отчета по лабораторной работе	59
	Библиография	60

1 Методические указания

1.1 Лабораторная работа № 1.

Эксперимент. Метод наименьших квадратов. Язык программирования R

1.1.1 Цель работы

- ознакомление с объектом и инструментарием исследования;
- получение базовых навыков использования языка программирования R для статистических расчетов;
- получение навыка построения регрессионной зависимости методом наименьших квадратов.

1.1.2 Теоретическая часть

1.1.2.1 Исследуемый объект

Исследуемым объектом является вычислительный кластер, составленный из нескольких вычислительных систем и решающий задачу генерации последовательности псевдослучайных чисел, имеющих гамма-распределение.

Вычислительные системы, входящие в кластер, работают под управлением ОС GNU/Linux. Для организации кластера использована спецификация MPI (Message Passing Interface), программный комплекс Open MPI, реализующий данную спецификацию, и программный комплекс OpenSSH, реализующий протокол SSH, используемый вычислительным кластером для организации обмена данными между вычислительными системами (вычислительными узлами), входящими в кластер.

1.1.2.2 Инструментарий проведения эксперимента

Для проведения единичного эксперимента с заданными параметрами используется программный комплекс, разработанный специально для данного курса лабораторных работ. Программный комплекс состоит из следующих компонент:

- сервер.

Сервер представляет из себя программу, написанную на языке программирования C. Сервер функционирует в вычислительных системах, работающих под управлением ОС GNU/Linux, и должен быть запущен на главном вычислительном узле кластера.

Сервер реализует следующий функционал:

- управление вычислительным кластером;
 - запуск расчета задачи на кластере;
 - установ параметров очередного прогона кластера;
 - замер времени выполнения очередного расчета задачи;
 - управление очередью заявок на выполнение задачи на кластере;
- клиент.

Клиент представляет из себя программу, написанную на языке программирования C++ с использованием библиотеки QT4. Клиент может быть запущен в вычислительных системах, работающих под управлением ОС GNU/Linux, ОС семейства Windows и прочих ОС, в которые портированы коллекция компиляторов GNU Compiler Collection (или существует коллекция компиляторов, совместимая с данной) и библиотека QT4 языка программирования C++.

Клиент реализует следующий функционал:

- постанов в очередь сервера заявки на выполнение задачи на кластере;
- указание параметров очередного прогона кластера;
- вывод на экран результатов очередного прогона кластера;
- сохранение набора результатов прогона кластера в файл с целью их дальнейшей обработки средствами языка программирования R.

1.1.2.2.1 Руководство оператора

Для запуска клиента необходимо:

- (ОС GNU/Linux, ОС семейства BSD) Запустить на выполнение файл `experiment-client` (например, двойным щелчком по файлу в окне файлового менеджера);
- (ОС Windows) Двойным щелчком запустить на выполнение файл `experiment-client.exe`.

Главное окно клиента приведено на рисунке 1.

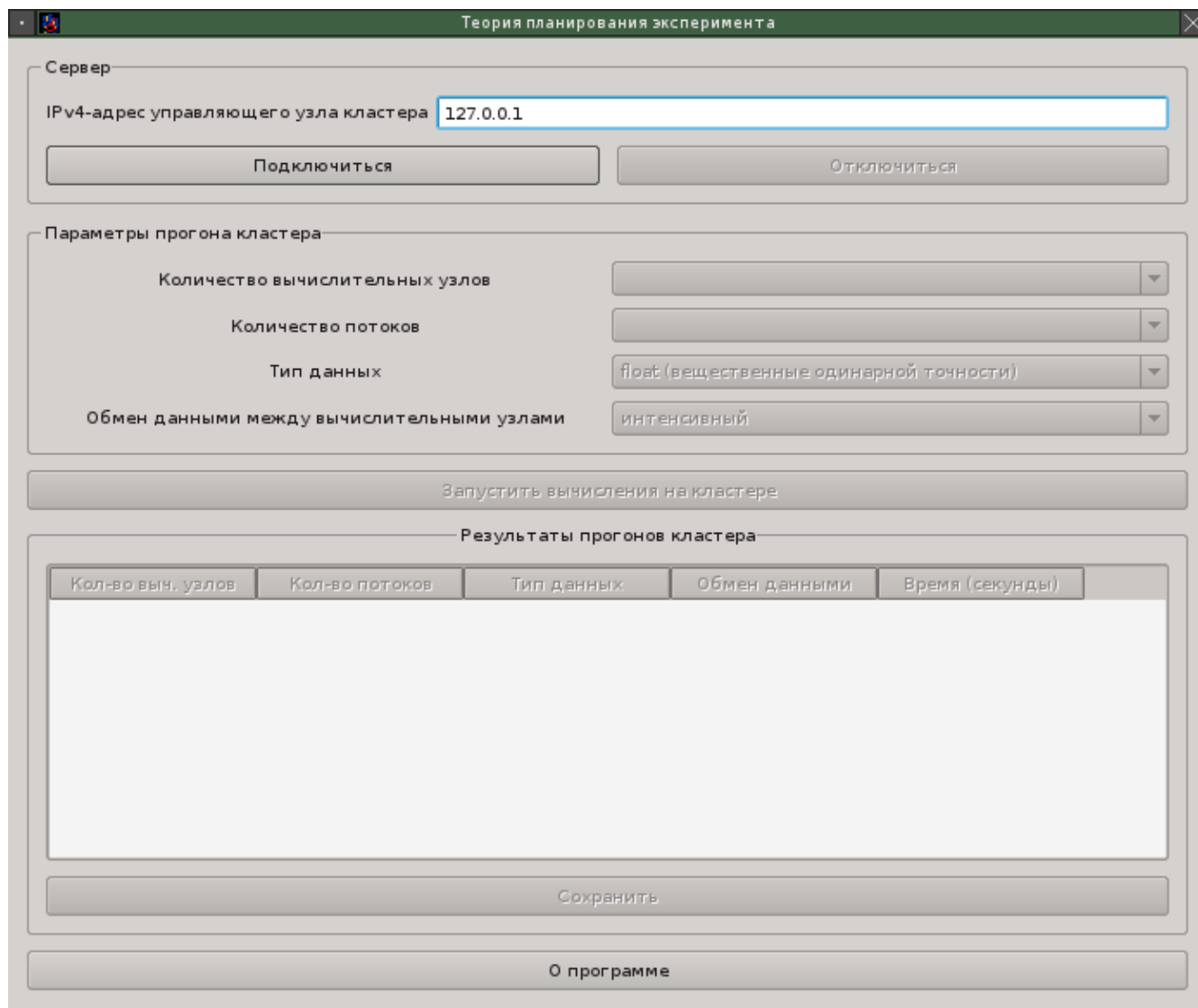


Рисунок 1 — Главное окно клиента

Для подключения к серверу необходимо:

- 1) в поле «IPv4-адрес управляющего узла кластера» ввести IPv4 - адрес сетевого узла, на котором запущен сервер;
- 2) нажать кнопку «Подключиться».

В случае успешного подключения главное окно клиента примет вид, приведенный на рисунке 2. В случае неудачного подключения клиентом будет выведено сообщение об ошибке, приведенное на рисунке 3.

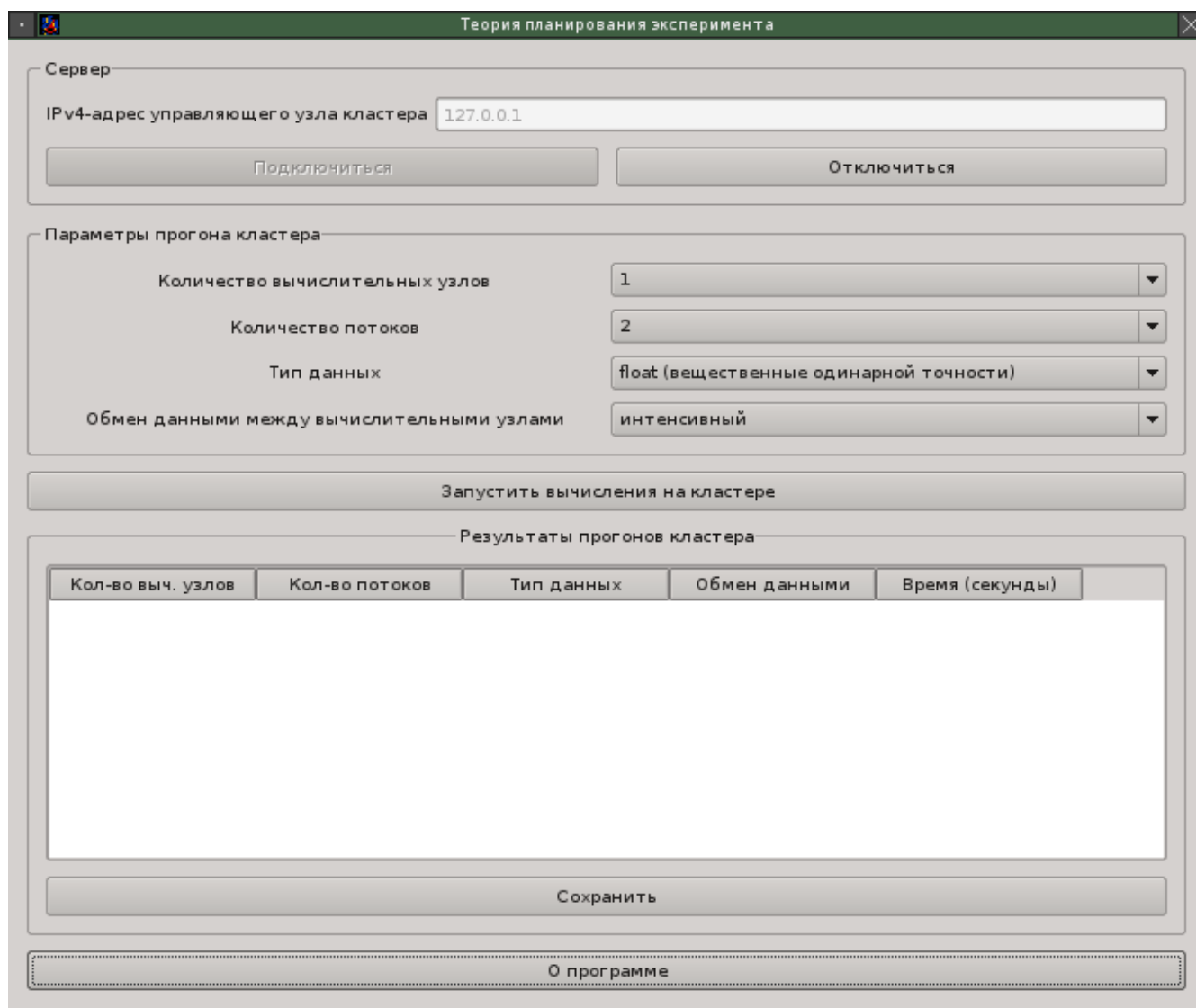


Рисунок 2 — Клиент успешно подключился к серверу

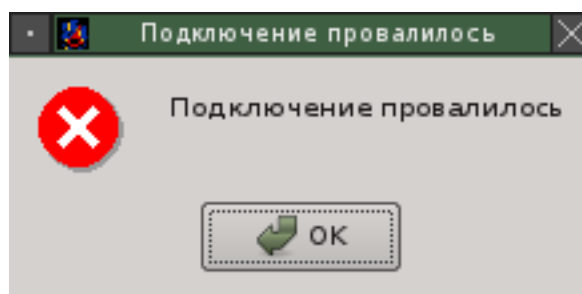


Рисунок 3 — Подключение клиента к серверу завершилось неудачей

Для отключение клиента от сервера оператор должен нажать кнопку «Отключиться».

Для запуска очередного прогона кластера оператор должен:

1) установить параметры прогона кластера.

Оператор может установить следующие параметры прогона кластера:

- количество вычислительных узлов кластера, участвующих в решении задачи - устанавливается с помощью списка «Количество вычислительных узлов».

Минимальное значение - 1.

Максимальное значение - количество вычислительных узлов в кластере;

- количество вычислительных потоков, на которые равномерно распределяется задача - устанавливается с помощью списка «Количество потоков».

Минимальное значение - 2.

Максимальное значение - зависит от настроек сервера;

- тип вещественных данных, используемых при решении задачи - устанавливается с помощью списка «Тип данных».

Возможные значения:

- float (вещественные одинарной точности; размер - 4 байта);
- double (вещественные двойной точности; размер - 8 байт);
- long double (вещественные максимальной точности в вычислительных системах, из которых составлен кластер; размер - 16 байт);
- интенсивность обмена данными между вычислительными узлами, составляющими кластер - устанавливается с помощью списка «Обмен данными между вычислительными узлами».

Возможные значения:

- интенсивный;
- средний;
- небольшой;

2) поставить заявку на прогон кластера в очередь заявок сервера.

Для постановки заявки на прогон кластера в очередь заявок сервера оператор должен нажать кнопку «Запустить вычисления на кластере».

После успешного постановки заявки на прогон кластера в очередь заявок сервера главное окно клиента примет вид, приведенный на рисунке 4. В случае, если постановка заявки в очередь сервера потерпел неудачу, клиент выведет сообщение об ошибке, приведенное на рисунке 5.

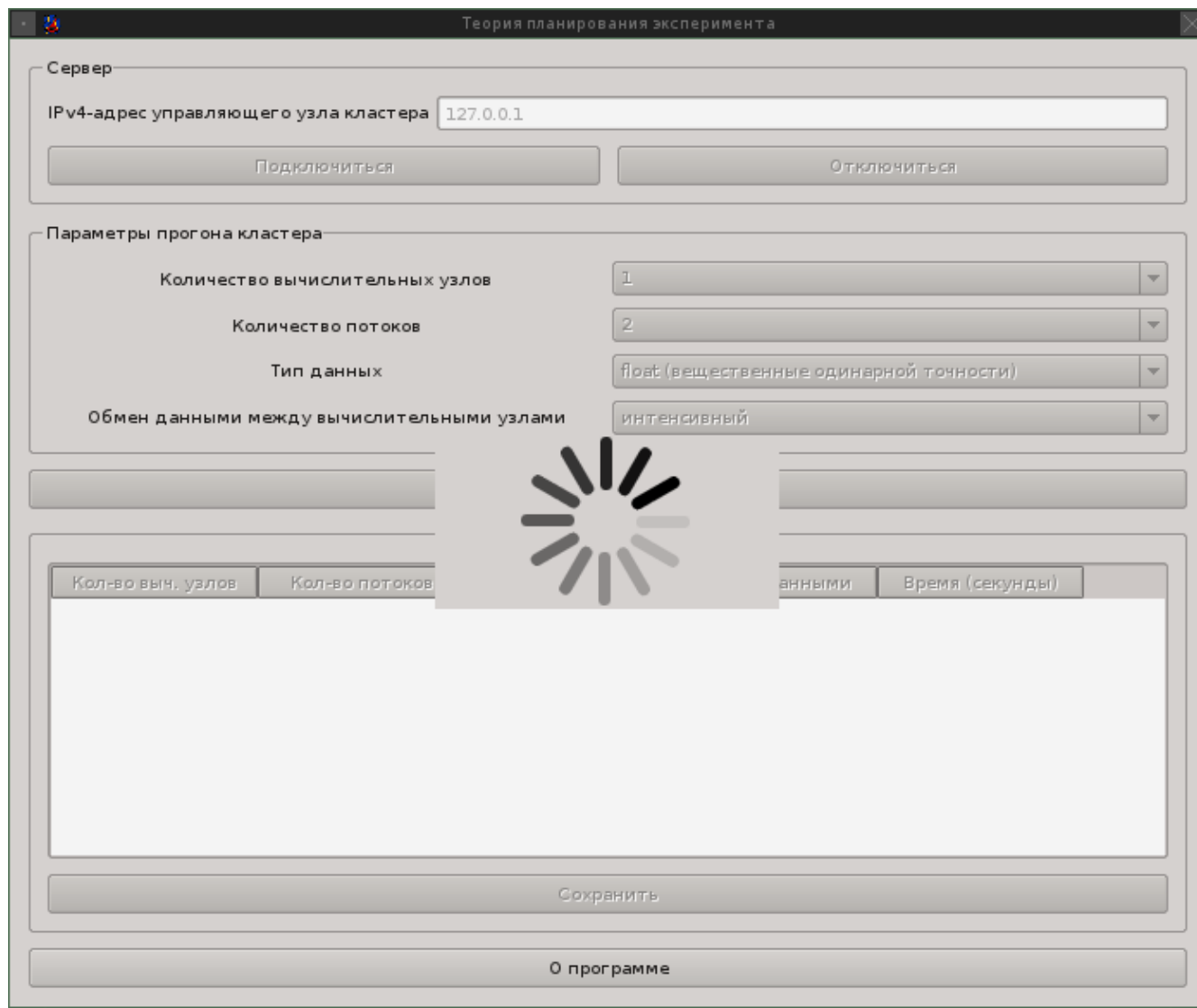


Рисунок 4 — Заявка на прогон кластера успешно поставлена в очередь заявок сервера

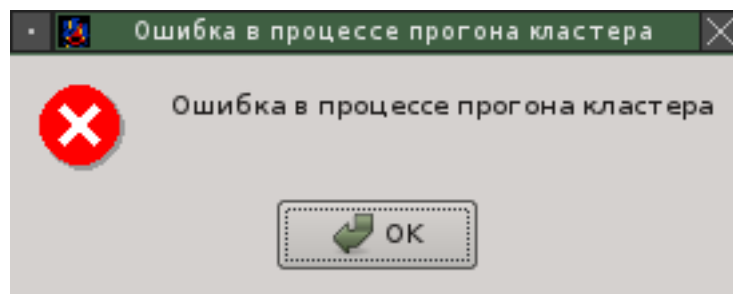


Рисунок 5 — Постанов заявки на прогон кластера в очередь заявок сервера потерпел неудачу

После того, как очередной прогон кластера будет выполнен, результаты прогона будут помещены в таблицу «Результаты прогонов кластера», что проиллюстрировано рисунком 6.

Теория планирования эксперимента

Сервер

IPv4-адрес управляющего узла кластера

Параметры прогона кластера

Количество вычислительных узлов

Количество потоков

Тип данных

Обмен данными между вычислительными узлами

Результаты прогонов кластера

	Кол-во выч. узлов	Кол-во потоков	Тип данных	Обмен данными	Время (секунды)
1	1	2	float	интенсивный	4.8284127860

Рисунок 6 — Результаты прогона кластера

После выполнения нескольких прогонов кластера результаты прогонов необходимо сохранить в файле для их дальнейшей обработки с помощью языка программирования R. Для сохранения результатов прогонов кластера оператор должен:

- 1) нажать кнопку «Сохранить»;
- 2) в окне «Сохранение», содержимое которого приведено на рисунке 7, выбрать каталог, в котором необходимо создать результирующий файл, и ввести имя результирующего файла;
- 3) нажать кнопку «Save» («Сохранить») окна «Сохранение».

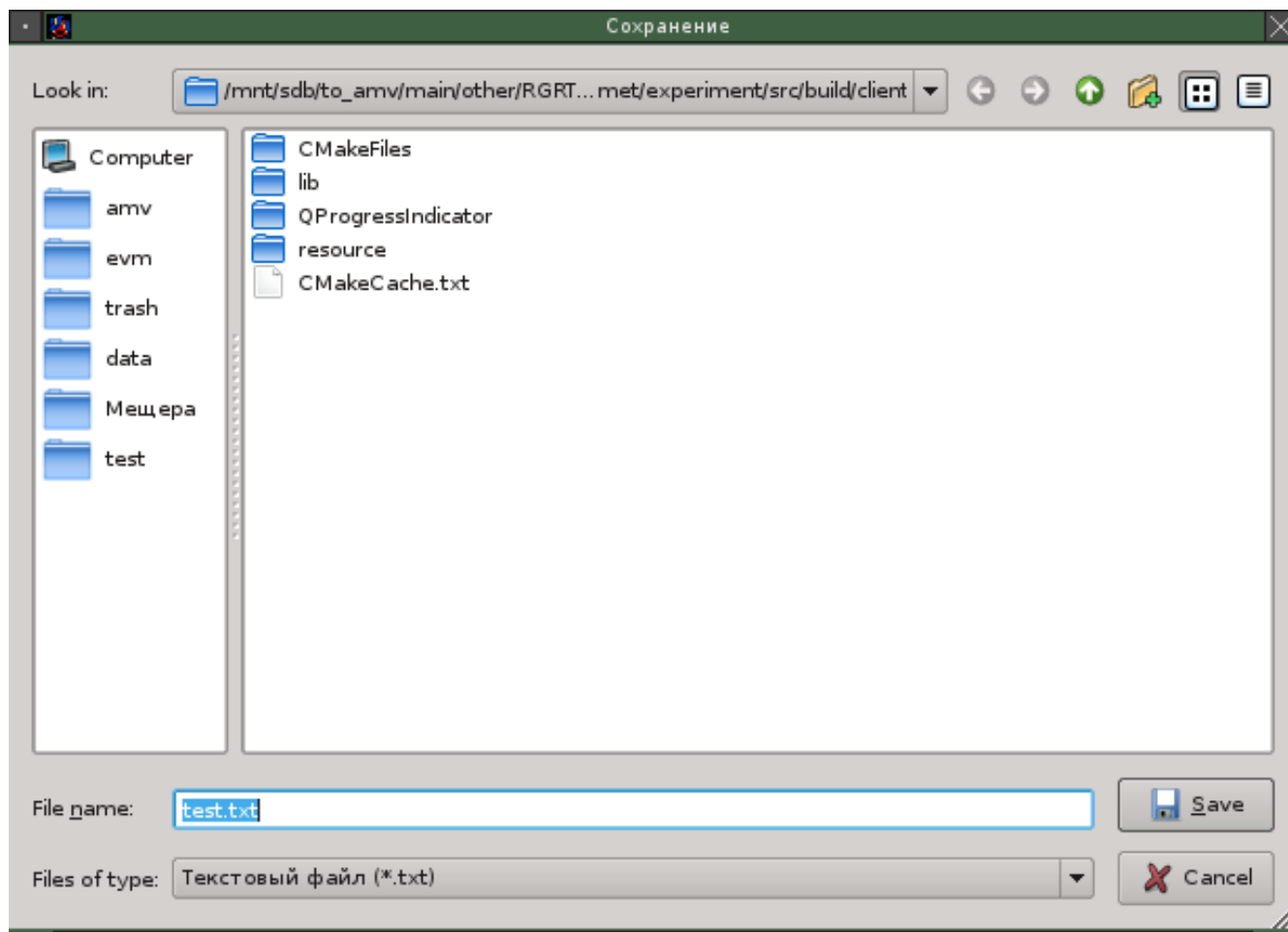


Рисунок 7 — Сохранение результатов прогонов кластера в файл

В случае, если сохранение результатов прогонов кластера в файл завершилось неудачей, клиентом будет выведено сообщение, приведенное на рисунке 8.

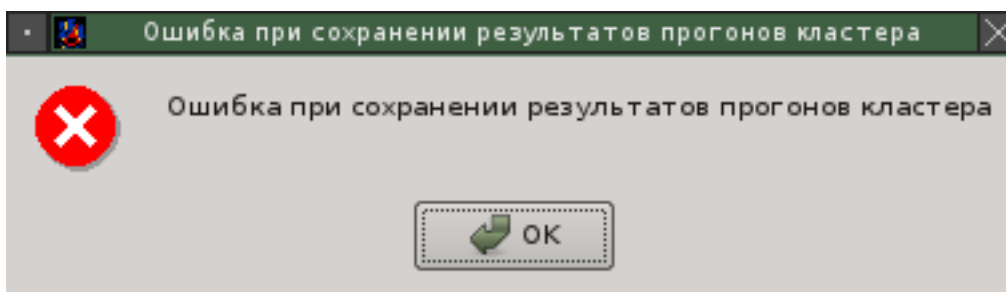


Рисунок 8 — Сохранение результатов прогонов кластера в файл завершилось неудачей

Содержимое результирующего файла структурировано в виде таблицы. Каждая строка таблицы, кроме первой, содержит результаты одного прогона класте-

ра. Первая строка содержит заголовок таблицы. Таблица состоит из следующих столбцов:

- «Количество_узлов» - количество вычислительных узлов кластера, задействованных в данном прогоне;
- «Количество_потоков» - количество вычислительных потоков, на которые была разделена задача в данном прогоне;
- «Тип_данных» - тип вещественных данных, использованный в данном прогоне (0 - float, 1 - double, 2 - long double);
- «Интенсивность» - интенсивность обмена данными между вычислительными узлами, составляющими кластер (0 - высокая, 1 - средняя, 2 - низкая);
- «Время» - время в секундах, потребовавшееся кластеру на решение задачи.

1.1.2.3 Обработка результатов эксперимента. Построение регрессии методом наименьших квадратов. Язык программирования R

1.1.2.3.1 Язык программирования R

Язык программирования R является популярной средой численных вычислений, предназначенной, прежде всего, для статистической обработки данных.

Язык программирования R был создан как свободная бесплатная альтернатива языку программирования S, разработанному компанией Bell Laboratories. Изначально R был разработан сотрудниками статистического факультета Оклендского университета Россом Айхэкой и Робертом Джентлменом, на текущий момент R поддерживается и развивается организацией «R Foundation».

Официальный сайт языка программирования R - [1]. Хорошее русскоязычное введение в R было издано циклом статей в журнале «Linux Format» и доступно в электронном виде на официальном сайте журнала - [2].

1.1.2.3.2 Построение регрессии методом наименьших квадратов

Метод наименьших квадратов - метод регрессионного анализа, позволяющий оценить неизвестные величины по зашумленным результатам их измерений, выполненных на реальном объекте / модели объекта.

Алгоритм аппроксимации неизвестной функциональной зависимости $f(x)$ по набору значений функции в нескольких точках:

- 1) выбрать аппроксимирующую функцию (например, многочлен K -го порядка);
- 2) выделить смежные не пересекающиеся участки аппроксимации;
- 3) рассчитать значения параметров аппроксимирующей функции на каждом из участков (рассчитывается система дифференциальных уравнений);
- 4) если качество аппроксимации не устраивает (значение среднеквадратичного отклонения больше, чем требуется), повторить алгоритм с аппроксимирующей функцией другого вида или с другим количеством участков аппроксимации.

На рисунках 9, 10, 11 и 12 приведен пример аппроксимации зашумленной параболы с помощью многочлена первого порядка.

Рисунок 9 — Количество аппроксимирующих
прямых = 1

Рисунок 10 — Количество
аппроксимирующих прямых = 3

Рисунок 11 — Количество
аппроксимирующих прямых = 5

Рисунок 12 — Аппроксимируемая парабол

1.1.2.3.3 Использование языка программирования R для обработки результатов эксперимента

Язык программирования R является интерпретируемым. Для запуска интерпретатора в интерактивном режиме оператор вычислительной системы должен:

- (ОС GNU/Linux, ОС семейства BSD) Запустить на выполнение исполняемый файл интерпретатора R (например, выполнив в консоли команду R);
- (ОС Windows) Щелкнуть левой кнопкой мыши по пункту «R»¹ меню «Пуск / Стандартные / R».

Каждая команда интерпретатора должна быть завершена нажатием клавиши «Enter».

¹В некоторых случаях - «R 2.14.0» или «R VERSION», где VERSION - версия языка программирования R.

Для **завершения работы интерпретатора** необходимо воспользоваться функцией `quit()`. На предложение интерпретатора сохранить рабочее пространство допускается произвольный ответ.

Язык программирования R оперирует данными различных типов. Тип переменной определяется при присваивании ей значения по типу присваиваемого значения. Тип переменной может быть переопределен при следующем присваивании.

Для **присваивания значения** переменной используется оператор `<-`.

Язык программирования R располагает большой стандартной библиотекой функций.

Для **чтения из файла набора данных**, структурированных в виде таблицы, оператор может воспользоваться функцией `read.table()`, один из форматов вызова которой приведен в листинге 1.

```
VAR <- read.table("FNAME", skip = 1)
```

Листинг 1 — Функция `read.table()`

Здесь переменной `VAR` будет присвоено значение типа «таблица», считанное из файла, путь к которому и имя которого указаны в первом параметре функции (`FNAME`). Параметр `skip` функции `read.table()` позволяет указать количество строк от начала файла, пропускаемых при чтении (в данном случае - одна строка - заголовок таблицы).

Для **вывода содержимого таблицы** необходимо ввести ее имя.

Доступ к столбцу таблицы или к значению отдельной ячейки таблицы может быть получен с помощью оператора `[]`. Нумерация строк и столбцов таблиц в R (ровно как и прочая индексация) ведется от единицы. В листинге 2 переменной `VAR2` присваивается значение ячейки таблицы `VAR1`, расположенной на строке `ROW` в колонке `COL`. В листинге 3 переменной `VAR3` присваивается массив, состоящий из элементов, составляющих столбец `COL` таблицы `VAR1`.

```
VAR2 <- VAR1[ROW, COL]
```

Листинг 2 — Получение значения ячейки таблицы

```
VAR3 <- VAR1[[COL]]
```

Листинг 3 — Доступ к столбцу таблицы

Другим типом данных, существующим в R, является формула. Примеры **присвоений значений типа «формула»** переменным приведены в листинге 4.

```
VAR <- Y ~ FORMULA
a <- y ~ x1 + a * x2
b <- z ~ x1 / x3 - x1 * x2
```

Листинг 4 — Присвоение переменным значение типа «формула»

Здесь:

- **VAR** - имя переменной, которой присваивается значение типа «формула»;
- **Y** - функция;
- **FORMULA** - непосредственно формула - корректная комбинация аргументов, констант и знаков операций.

Для построения **регрессионной зависимости оптимизируемой величины от набора аргументов по произвольной функции** с помощью метода наименьших квадратов необходимо использовать функцию `nls()`. Пример использования функции `nls()` приведен в листинге 5.

```
tbl <- read.table("test.txt", skip = 1)
x_1 <- tbl[[1]]
x_2 <- tbl[[2]]
y <- tbl[[5]]

f <- y ~ a * x_1 + b * x_2
m <- nls(f, start = c(a = 0, b = 0))

m

coefficients(m) ["a"]
coefficients(m) ["b"]
```

Листинг 5 — Построение регрессионной зависимости оптимизируемой величины от набора аргументов по произвольной функции

В листинге 5 в переменную `m` помещаются параметры регрессии пятого столбца таблицы `tbl` от ее первого и второго столбцов по многочлену $ax_1 + bx_2$. Ввод имени переменной `m` на экран выводится полная информация о параметрах

регрессии. К каждому из параметров может быть получен доступ с помощью функции `coefficients`.

Параметр `start` функции `nls()` содержит вектор (вектора описываются с помощью функции `c()`) начальных значений параметров рассчитываемой регрессионной зависимости.

После расчета регрессии необходимо минимизировать ее функцию и получить на выходе минимизации оптимальный набор параметров функционирования анализируемого объекта. **Минимизацию произвольной функции** можно осуществить с помощью функции `optim()`, пример использования которой приведен в листинге 6.

```
fn <- function(x) { eval( formula(m)[[3]], c(as.list(coefficients(m)), x_1 = x[1], x_2 = x[2])) }

opt <- optim(c(0, 0), fn, lower = c(1, 2), upper = c(4, 100),
  method = "L-BFGS-B")

opt$par
```

Листинг 6 — Минимизация функции

В листинге 6:

- описывается функция `fn()`, принимающая на вход вектор `x` и рассчитывающая значение функции регрессии `m`, подставляя вместо параметров функции параметры рассчитанной регрессии (`as.list(coefficients(m))`), а на место аргументов функции - элементы вектора `x` (`x_1 = x[1]`, `x_2 = x[2]`);
- минимизируется функция `fn()`.

Параметры минимизации:

- начальные значения компонент вектора входных параметров функции `fn()` - первый параметр функции `optim()` - вектор `c(0, 0)`;
- диапазон поиска оптимальных значений компонент вектора параметров функции `fn()` - параметры `lower` (нижние границы поиска) и `upper` (верхние границы поиска);
- метод оптимизации - параметр `method`;
- выводится вектор оптимальных значений компонент вектора параметров функции `fn()`.

1.1.2.3.4 Пример расчета оптимального набора значений параметров функционирования кластера

На рисунках 13, 14, 15 и 16 приведен пример расчета оптимального набора значений параметров функционирования кластера.

В данном примере минимизируется время прогона кластера по количеству вычислительных узлов, задействованных в прогоне, и по количеству вычислительных потоков, по которым распределяется задача, решаемая кластером. Тип вещественных данных - float, интенсивность обмена данными - низкая.

В приводимом примере регрессия рассчитывается по функции (1).

$$y = ax_1^2 + bx_2^2 + \exp(cx_1x_2) + d \quad (1)$$

Параметры регрессии:

- $a = -0.07875$;
- $b = 0.01657$;
- $c = -0.08837$;
- $d = 1.63984$.

Оптимальные параметры прогона кластера:

- количество вычислительных узлов кластера, участвующих в решении задачи - 4;
- количество вычислительных потоков, решающих задачу - 3.310398 - округление до 3 или до 4 в зависимости от способа округления.

```
amv@amv:~/main/other/RGRTA/evm/met/experiment/trash
akinin@experiment $
akinin@experiment $ R

R version 2.14.0 (2011-10-31)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-unknown-linux-gnu (64-bit)

R -- это свободное ПО, и оно поставляется безо всяких гарантий.
Вы вольны распространять его при соблюдении некоторых условий.
Введите 'license()' для получения более подробной информации.

R -- это проект, в котором сотрудничает множество разработчиков.
Введите 'contributors()' для получения дополнительной информации и
'citation()' для ознакомления с правилами упоминания R и его пакетов
в публикациях.

Введите 'demo()' для запуска демонстрационных программ, 'help()' -- для
получения справки, 'help.start()' -- для доступа к справке через браузер.
Введите 'q()', чтобы выйти из R.

>
> tbl <- read.table("test.txt", skip = 1)
> all_tbl <- read.table("all_test.txt", skip = 1)
>
> tbl
  V1 V2 V3 V4      V5
1  1  2  0  2 3.117686
2  1 10  0  2 3.759302
3  2 10  0  2 3.468460
4  2  2  0  2 1.325673
5  3  2  0  2 1.314025
6  3 10  0  2 2.031883
7  4 10  0  2 2.248048
8  4  2  0  2 1.326412
>
>
```

Рисунок 13 — Загрузка таблиц с результатами эксперимента

```
amv@amv:~/main/other/RGRTA/evm/met/experiment/trash
>
> x1 <- tbl[[1]]
> x2 <- tbl[[2]]
>
> y <- tbl[[5]]
>
> f <- y ~ a * x1 * x1 + b * x2 * x2 + exp(c * x1 * x2) + d
>
> m <- nls(f, start = c(a = 0, b = 0, c = 0, d = 0))
>
> m
Nonlinear regression model
  model: y ~ a * x1 * x1 + b * x2 * x2 + exp(c * x1 * x2) + d
 data: parent.frame()
      a      b      c      d
-0.07875 0.01657 -0.08837 1.63984
residual sum-of-squares: 1.781

Number of iterations to convergence: 16
Achieved convergence tolerance: 6.935e-06
>
> █
```

Рисунок 14 — Расчет регрессии

```
amv@amv:~/main/other/RGRTA/evm/met/experiment/trash
>
> fn <- function(x) { eval( formula(m)[[3]], c( as.list(coefficients(m)), x1 = x[1],
x2 = x[2] ) ) }
>
> opt <- optim( c(0, 0), fn, lower = c(1, 2), upper = c(4, 10), method = "L-BFGS-B")
>
> opt
$par
[1] 4.000000 3.310398

$value
[1] 0.8717152

$counts
function gradient
      11      11

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"

>
> opt$par
[1] 4.000000 3.310398
>
> █
```

Рисунок 15 — Минимизация функции регрессии - поиск оптимальных параметров прогона кластера

```
amv@amv: ~/main/other/RGRTA/evm/met/experiment/trash
>
> opt$par
[1] 4.000000 3.310398
>
> all_tbl
  V1 V2 V3 V4      V5
1   1  2  0  2 3.1176861
2   1  3  0  2 2.6784716
3   1  4  0  2 3.5848857
4   1  5  0  2 2.5679483
5   1  6  0  2 3.5811359
6   1  7  0  2 3.6206986
7   1  8  0  2 3.6560804
8   1  9  0  2 3.7025474
9   1 10  0  2 3.7593018
10  2 10  0  2 3.4684603
11  2  9  0  2 3.2300864
12  2  8  0  2 3.1216231
13  2  7  0  2 2.2031403
14  2  6  0  2 2.0791916
15  2  5  0  2 2.2124084
16  2  4  0  2 2.0752408
17  2  3  0  2 2.3782921
18  2  2  0  2 1.3256734
19  3  2  0  2 1.3140246
20  3  3  0  2 1.1894120
21  3  4  0  2 2.0937015
22  3  5  0  2 1.9514700
23  3  6  0  2 1.8635241
24  3  7  0  2 2.0180233
25  3  8  0  2 1.9537805
26  3  9  0  2 1.9021133
27  3 10  0  2 2.0318834
28  4 10  0  2 2.2480479
29  4  9  0  2 1.9387789
30  4  8  0  2 1.7824892
31  4  7  0  2 1.8324502
32  4  6  0  2 1.8909685
33  4  5  0  2 1.9793662
34  4  4  0  2 0.7586967
35  4  3  0  2 0.8764989
36  4  2  0  2 1.3264119
>
>
```

Рисунок 16 — Сравнение результатов минимизации функции регрессии с полными результатами эксперимента

1.1.3 Задание

Лабораторная работа выполняется побригадно - бригада может состоять из одного или нескольких человек (не более трех) - всего допускается существование восьми бригад.

Каждая из бригад получает свой уникальный набор значений параметров «Тип вещественных данных» и «Интенсивность обмена данными между вычислительными узлами, составляющими кластер»:

- 1) «Тип вещественных данных» - float (вещественные одинарной точности);
«Интенсивность обмена данными между вычислительными узлами» - средняя;
- 2) «Тип вещественных данных» - float (вещественные одинарной точности);
«Интенсивность обмена данными между вычислительными узлами» - высокая;
- 3) «Тип вещественных данных» - double (вещественные двойной точности);
«Интенсивность обмена данными между вычислительными узлами» - низкая;
- 4) «Тип вещественных данных» - double (вещественные двойной точности);
«Интенсивность обмена данными между вычислительными узлами» - средняя;
- 5) «Тип вещественных данных» - double (вещественные двойной точности);
«Интенсивность обмена данными между вычислительными узлами» - высокая;
- 6) «Тип вещественных данных» - long double (вещественные максимальной точности);
«Интенсивность обмена данными между вычислительными узлами» - низкая;
- 7) «Тип вещественных данных» - long double (вещественные максимальной точности);
«Интенсивность обмена данными между вычислительными узлами» - средняя;
- 8) «Тип вещественных данных» - long double (вещественные максимальной точности);
«Интенсивность обмена данными между вычислительными узлами» - высокая.

Для выполнения лабораторной работы необходимо:

- выбрать функцию аппроксимации зависимости времени решения задачи на кластере от количества вычислительных узлов, задействованных в прогоне кластера, и от количества вычислительных потоков, по которым распределяется задача в ходе своего решения;
- провести серию экспериментов (не более 12-ти прогонов кластера);
- рассчитать регрессию времени решения задачи от количества вычислительных узлов и от количества вычислительных потоков;
- найти оптимальные параметры функционирования кластера.

В случае, если найденные параметры не являются оптимальными по той или иной оценки - необходимо выполнить лабораторную работу заново с другой функцией аппроксимации.

Расчет регрессии и подбор оптимальных параметров функционирования кластера необходимо выполнить с помощью языка программирования R.

Отчет по лабораторной работе оформляется индивидуально каждым студентом.

1.1.4 Требования к оформлению отчета по лабораторной работе

Отчет по лабораторной работе должен содержать:

- титульный лист, оформленный в соответствии с требованиями преподавателя, кафедры и университета и действующих государственных стандартов по оформлению титульных листов отчетов по лабораторным работам;
- задание к лабораторной работе;
- скриншоты, иллюстрирующие процесс выполнения серии экспериментов (скриншот необходимо делать после каждого прогона кластера);
- скриншоты, иллюстрирующие процесс загрузки данных в интерпретатор языка программирования R, процесс расчета регрессии и процесс подбора оптимальных параметров функционирования кластера;
- скриншоты, содержащие вывод результатов эксперимента, вывод аппроксимирующей функции, вывод параметров регрессии, вывод оптимальных значений параметров функционирования кластера - все объекты должны быть выведены на экран средствами интерпретатора языка программирования R;

- вывод.

Вывод по выполнению лабораторной работы должен содержать:

- обоснование выбора вида регрессионной зависимости;
- выводы о найденных оптимальных значениях коэффициентов регрессионной зависимости;
- заключение об оптимальности (или отсутствию таковой) подобранных параметров функционирования кластера.

1.2 Лабораторная работа № 2.

Полный и дробный факторные эксперименты

1.2.1 Цель работы

- получить навыки оценки оптимальных параметров функционирования объекта исследований с помощью полного факторного эксперимента;
- получить навыки оценки оптимальных параметров функционирования объекта исследований с помощью дробного факторного эксперимента;
- получить навыки использования языка программирования R для обработки результатов полного и дробного факторных экспериментов.

1.2.2 Теоретическая часть

1.2.2.1 Полный факторный эксперимент

Полный факторный эксперимент – совокупность нескольких опытов, удовлетворяющих следующим условиям:

- количество измерений составляет $2n$, где n – количество факторов;
- каждый фактор принимает только два значения – верхнее и нижнее;
- в процессе измерения верхние и нижние значения факторов комбинируются во всех возможных сочетаниях.

Преимуществами полного факторного эксперимента являются:

- простота решения системы уравнений оценивания параметров;
- статистическая избыточность количества измерений, которая уменьшает влияние погрешностей отдельных измерений на оценку параметров.

Фактически, полный факторный эксперимент является способом оценки линейной регрессии зависимой величины y от набора параметров $P = \{p_i\}$; $i = \overline{1, N}$ исследуемого объекта f .

Полный факторный эксперимент предполагает выполнение следующих действий:

- 1) для каждого из параметров исследуемого объекта выбираются границы диапазонов, в которых параметры будут оцениваться:

$$p_i \in [l_i, u_i]; \quad i = \overline{1, N}$$

Каждый из оцениваемых параметров суть есть **фактор**, верхние и нижние границы диапазонов возможных значений параметров - **верхними и нижними значениями факторов**;

- 2) составляется матрица (2) всех возможных комбинаций границ диапазонов параметров.

$$X = \begin{pmatrix} 1 & l_1 & l_2 & \dots & l_i & \dots & l_N \\ 1 & u_1 & l_2 & \dots & l_i & \dots & l_N \\ \dots & & & & & & \\ 1 & u_1 & u_2 & \dots & u_i & \dots & u_N \end{pmatrix} \quad (2)$$

Всего комбинаций (число строк в X): $M = 2^N$;

- 3) для каждой комбинации параметров x_j ; $j = \overline{1, M}$ из X (x_j - строка матрицы X) рассчитывается значение зависимой величины - для каждой комбинации x_j проводится эксперимент (прогон объекта с заданными значениями параметров) - $y_j = f(x_j)$;
- 4) составляется система линейных уравнений (3).

$$\begin{cases} A * x_1^T = y_1 \\ A * x_2^T = y_2 \\ \dots \\ A * x_j^T = y_j \\ A * x_M^T = y_M \end{cases} \quad (3)$$

$$A = \{a_0, a_1, a_2, \dots, a_j, \dots, a_N\}$$

- 5) система линейных уравнений (очевидно, избыточная, так как $M > N + 1$) решается - например, методом Гаусса - в результате чего находится одно из допустимых решений: $A' = \{a'_0, a'_1, a'_2, \dots, a'_j, \dots, a'_N\}$;
- 6) компоненты решения A' системы уравнений (3) становятся параметрами линейной регрессии y от P .

Для большинства задач линейная регрессия дает погрешность, выходящую за допустимые рамки. Возможны следующие способы уменьшения погрешности:

- построение нелинейной регрессии - например, с помощью метода наименьших квадратов по произвольной функции;
- разбиение диапазона поиска одного из параметров на два или более поддиапазона - построение регрессии y от P с помощью полного факторного эксперимента в каждом из поддиапазонов.

1.2.2.2 Дробный факторный эксперимент

Дробный факторный эксперимент снижает или вовсе исключает избыточность системы уравнений, решаемой в рамках полного факторного эксперимента, путем исключения из системы одного или нескольких уравнений.

План дробного эксперимента создается из плана полного эксперимента с помощью **генератора плана дробного эксперимента** - с помощью правила исключения уравнений из системы уравнений полного факторного эксперимента.

Очевидно, результирующая система уравнений должна быть полной.

1.2.2.3 Решение системы линейных уравнений средствами языка программирования R

Ключевой задачей, решаемой в ходе факторного эксперимента, является решение системы линейных уравнений. Стандартная библиотека языка программирования R содержит функционал, предназначенный для решения **системы линейных уравнений** - функцию `solve()`, пример использования которой приведен в листинге 7.

```
RES <- solve(X, Y)
```

Листинг 7 — Решение системы линейных уравнений

Здесь:

- **X** - матрица коэффициентов уравнений;
- **Y** - вектор значений свободных членов;
- **RES** - вектор значений неизвестных.

Для **создания матрицы** может быть использована функция `matrix()`, что продемонстрировано в листинге 8.

```
MATR <- matrix(VEC, ncol = COL_NUM)
X <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), ncol = 3)
```

Листинг 8 — Создание матрицы

В первой строке листинга 8 создается матрица **MATR** - результат развертывания вектора **VEC** по трем столбцам матрицы (и наоборот - матрица развертывается в вектор по столбцам).

Во второй строке листинга 8 создается матрица X (4).

$$X = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix} \quad (4)$$

Проверить правильность решения системы линейных уравнений можно способом, приведенным в листинге 9.

```
RES <- solve(X, Y)
sum(X %*% RES - Y)
```

Листинг 9 — Проверка правильности решения системы линейных уравнений

Во второй строке листинга 9 рассчитывается произведение матрицы **X** на вектор значений неизвестных **RES**, из полученного произведения вычитается вектор значений свободных членов, после чего компоненты вектора результатов вычитаний суммируются друг с другом с помощью функции **sum()**. Результат суммирования для правильного решения системы линейных уравнений должен быть равен нулю с точностью до погрешности, обусловленной приближительностью операций с плавающей точкой в вычислительной системе.

На рисунках 17, 18, 19 и 20 приведен пример расчета регрессии времени выполнения прогона кластера (пятый столбец таблицы) от количества вычислительных узлов, задействованных в прогоне (первый столбец таблицы), и от количества вычислительных потоков, по которым распределяется решаемая задача (второй столбец таблицы).

Рисунками 17 - 20 иллюстрируется дробный факторный эксперимент - как видно из рисунка 19 системы линейных уравнений не являются избыточными (количество уравнений в каждой из систем - 3, тогда как в случае полного факторного эксперимента в каждой из систем было бы по $2^2 = 4$ уравнения).

```
amv@amv:~/main/other/RGRTA/evm/met/experiment/trash
>
> tbl <- read.table("all_test.txt", skip = 1)
>
> tbl
  V1 V2 V3 V4      V5
1  1  2  0  2 3.1176861
2  1  3  0  2 2.6784716
3  1  4  0  2 3.5848857
4  1  5  0  2 2.5679483
5  1  6  0  2 3.5811359
6  1  7  0  2 3.6206986
7  1  8  0  2 3.6560804
8  1  9  0  2 3.7025474
9  1 10  0  2 3.7593018
10 2 10  0  2 3.4684603
11 2  9  0  2 3.2300864
12 2  8  0  2 3.1216231
13 2  7  0  2 2.2031403
14 2  6  0  2 2.0791916
15 2  5  0  2 2.2124084
16 2  4  0  2 2.0752408
17 2  3  0  2 2.3782921
18 2  2  0  2 1.3256734
19 3  2  0  2 1.3140246
20 3  3  0  2 1.1894120
21 3  4  0  2 2.0937015
22 3  5  0  2 1.9514700
23 3  6  0  2 1.8635241
24 3  7  0  2 2.0180233
25 3  8  0  2 1.9537805
26 3  9  0  2 1.9021133
27 3 10  0  2 2.0318834
28 4 10  0  2 2.2480479
29 4  9  0  2 1.9387789
30 4  8  0  2 1.7824892
31 4  7  0  2 1.8324502
32 4  6  0  2 1.8909685
33 4  5  0  2 1.9793662
34 4  4  0  2 0.7586967
35 4  3  0  2 0.8764989
36 4  2  0  2 1.3264119
>
>
```

Рисунок 17 — Загрузка результатов экспериментов

```
amv@amv:~/main/other/RGRTA/evm/met/experiment/trash
>
> a1 <- matrix(c(1, 1, 2, 2, 5, 2, 1, 1, 1), ncol = 3)
> a2 <- matrix(c(1, 1, 2, 6, 10, 6, 1, 1, 1), ncol = 3)
> a3 <- matrix(c(3, 3, 4, 2, 5, 2, 1, 1, 1), ncol = 3)
> a4 <- matrix(c(3, 3, 4, 6, 10, 6, 1, 1, 1), ncol = 3)
>
> a1
      [,1] [,2] [,3]
[1,]    1    2    1
[2,]    1    5    1
[3,]    2    2    1
>
> a2
      [,1] [,2] [,3]
[1,]    1    6    1
[2,]    1   10    1
[3,]    2    6    1
>
> a3
      [,1] [,2] [,3]
[1,]    3    2    1
[2,]    3    5    1
[3,]    4    2    1
>
> a4
      [,1] [,2] [,3]
[1,]    3    6    1
[2,]    3   10    1
[3,]    4    6    1
>
> b1 <- c(3.1176861, 2.5679483, 1.3256734)
> b2 <- c(3.5811359, 3.7593018, 2.0791916)
> b3 <- c(1.3140246, 1.9514700, 1.3264119)
> b4 <- c(1.8635241, 2.0318834, 1.8909685)
>
> b1
[1] 3.117686 2.567948 1.325673
>
> b2
[1] 3.581136 3.759302 2.079192
>
> b3
[1] 1.314025 1.951470 1.326412
>
> b4
[1] 1.863524 2.031883 1.890969
>
>
```

Рисунок 18 — Создание четырех матриц коэффициентов уравнений и четырех векторов свободных членов - по одной матрице и одному вектору на каждую комбинацию поддиапазонов диапазонов значений параметров прогона кластера

```
amv@amv: ~/main/other/RGRTA/evm/met/experiment/trash
>
> m1 <- solve(a1, b1)
> m2 <- solve(a2, b2)
> m3 <- solve(a3, b3)
> m4 <- solve(a4, b4)
>
> m1
[1] -1.7920127 -0.1832459  5.2761907
>
> m2
[1] -1.50194430  0.04454147  4.81583135
>
> m3
[1] 0.0123873 0.2124818 0.8518991
>
> m4
[1] 0.02744440 0.04208982 1.52865195
>
> sum(a1 %*% m1 - b1)
[1] -4.440892e-16
>
> sum(a2 %*% m2 - b2)
[1] -4.440892e-16
>
> sum(a3 %*% m3 - b3)
[1] -2.220446e-16
>
> sum(a4 %*% m4 - b4)
[1] -2.220446e-16
>
>
```

Рисунок 19 — Решение четырех систем линейных уравнений


```
amv@amv:~/main/other/RGRTA/evm/met/experiment/trash
>
> x <- tbl[25, 1 : 2]
> x[[3]] <- 1
>
> x
  V1 V2 V3
25  3  8  1
>
> empiric <- sum(x * m4)
>
> empiric
[1] 1.947704
>
> empiric - tbl[25, 5]
[1] -0.006076798
>
>
```

Рисунок 20 — Расчет отклонения предсказанного времени выполнения прогона кластера от времени, имевшего место быть на практике, для одной из возможных комбинаций значений параметров прогона кластера

1.2.3 Задание

Лабораторная работа выполняется побригадно - бригада может состоять из одного или нескольких человек (не более трех) - всего допускается существование восьми бригад.

Каждая из бригад получает свой уникальный набор значений параметров «Тип вещественных данных» и «Интенсивность обмена данными между вычислительными узлами, составляющими кластер»:

- 1) «Тип вещественных данных» - float (вещественные одинарной точности);
«Интенсивность обмена данными между вычислительными узлами» - средняя;
- 2) «Тип вещественных данных» - float (вещественные одинарной точности);
«Интенсивность обмена данными между вычислительными узлами» - высокая;
- 3) «Тип вещественных данных» - double (вещественные двойной точности);
«Интенсивность обмена данными между вычислительными узлами» - низкая;

- 4) «Тип вещественных данных» - double (вещественные двойной точности);
«Интенсивность обмена данными между вычислительными узлами» - средняя;
- 5) «Тип вещественных данных» - double (вещественные двойной точности);
«Интенсивность обмена данными между вычислительными узлами» - высокая;
- 6) «Тип вещественных данных» - long double (вещественные максимальной точности);
«Интенсивность обмена данными между вычислительными узлами» - низкая;
- 7) «Тип вещественных данных» - long double (вещественные максимальной точности);
«Интенсивность обмена данными между вычислительными узлами» - средняя;
- 8) «Тип вещественных данных» - long double (вещественные максимальной точности);
«Интенсивность обмена данными между вычислительными узлами» - высокая.

Для выполнения лабораторной работы необходимо:

- 1) разбить диапазон возможных значений параметра «Количество вычислительных узлов, задействованных в прогоне кластера» на два поддиапазона; аналогичное разбиение, но на пять диапазонов, выполнить для параметра «Количество вычислительных потоков, между которыми распределяется решаемая задача»;
- 2) составить все возможные пары поддиапазонов диапазонов возможных значений параметров прогона кластера;
- 3) для каждой пары поддиапазонов построить линейную регрессию времени решения задачи на кластере от количества вычислительных узлов, задействованных в прогоне, и от количества вычислительных потоков, на которые распределяется задача.

Для построения линейной регрессии следует использовать дробный факторный эксперимент (три уравнения в системе линейных уравнений);

- 4) выбрать по три контрольных точки из каждой пары поддиапазонов;

- 5) для каждой контрольной точки предсказать время решения задачи с помощью построенной регрессии;
- 6) выполнить прогон кластера в каждой из контрольных точек;
- 7) для каждой пары поддиапазонов рассчитать среднее абсолютное отклонение предсказанного времени решения задачи от времени решения задачи, имевшего место быть на практике;
- 8) рассчитать среднее абсолютное отклонение на полных диапазонах возможных значений параметров регрессии.

Все вычисления должны быть выполнены с помощью языка программирования R.

Отчет по лабораторной работе оформляется индивидуально каждым студентом.

1.2.4 Требования к оформлению отчета по лабораторной работе

Отчет по лабораторной работе должен содержать:

- титульный лист, оформленный в соответствии с требованиями преподавателя, кафедры и университета и действующих государственных стандартов по оформлению титульных листов отчетов по лабораторным работам;
- задание к лабораторной работе;
- скриншоты, содержащие результаты всех поставленных экспериментов (в том числе - результаты прогонов кластера в контрольных точках);
- скриншоты с выводом процесса построения и значений коэффициентов линейной регрессии на каждой комбинации поддиапазонов диапазонов возможных значений параметров прогона кластера, предсказанного времени прогона кластера в двенадцати контрольных точках, расчета отклонений предсказанных значений от действительных значений времени прогона кластера;
- вывод.

Вывод по выполнению лабораторной работы должен содержать:

- выводы о найденных оптимальных значениях коэффициентов линейной регрессии на каждой комбинации поддиапазонов возможных значений параметров прогона кластера;

- заключение об эффективности использования многочленов первой степени для описания регрессии времени расчета задачи на кластере от количества вычислительных узлов, участвующих в прогоне кластера, и от количества вычислительных потоков, между которыми распределяется задача.

1.3 Лабораторная работа № 3.

Линейная регрессионная модель

1.3.1 Цель работы

- получение базовых навыков использования языка программирования R для статистических расчетов;
- получение навыков подбора вида и параметров линейной регрессионной модели.

1.3.2 Теоретическая часть

1.3.2.1 Линейная регрессионная модель

Линейная регрессионная модель - регрессионная модель $y = f(x)$, описывающая регрессию f зависимой переменной y от независимых переменных $x = \{x_1, x_2, \dots, x_N\}$ и удовлетворяющая следующим условиям:

- задача расчета параметров регрессии f сводится к задаче квадратичной минимизации - к задаче минимизации квадратичной функции $q_{opt}(\beta)$, где β - вектор параметров регрессии f ;
- частные производные минимизируемой функции $q'_{opt}(\beta)$ суть есть линейные зависимости от β ;
- β , в свою очередь, линейно зависит от y ;
- шум, имеющий место быть в серии экспериментов (X, Y) , может быть описан как линейная составляющая коэффициентов β .

Частным случаем линейной регрессионной модели является линейная регрессия (5).

$$y = \beta_0 + \sum_{i=1}^N \beta_i x_i \quad (5)$$

Линейная регрессия — регрессионная модель зависимости переменной y от вектора независимых переменных x с линейной функцией зависимости $y = f(x)$.

Модель линейной регрессии является часто используемой и наиболее изученной - хорошо изучены свойства оценок параметров, получаемых различными методами при тех или иных предположениях о вероятностных характеристиках

факторов и случайных ошибок модели. Предельные (асимптотические) свойства оценок нелинейных моделей также выводятся исходя из аппроксимации последних линейными моделями.

В случае линейной регрессии ее параметры могут быть эффективно найдены с помощью метода наименьших квадратов, что представляет собой задачу квадратичной оптимизации. Нетрудно также показать, что каждая частная производная минимизируемой функции является линейной зависимостью от β .

1.3.2.1.1 Расчет параметров линейной регрессионной модели средствами языка программирования R

Для расчета параметров линейной регрессионной модели в языке программирования R применяется функция `lm()`, пример использования которой приведен в листинге 10.

```
tbl <- read.table("test.txt", skip = 1)
x_1 <- tbl[[1]]
x_2 <- tbl[[2]]
y <- tbl[[5]]

f <- y ~ x_1 + x_2 + x_1 * x_2
m <- nls(f)
```

Листинг 10 — Расчет параметров линейной регрессионной модели

В листинге 10 рассматривается построение регрессии пятой колонки данных, считанных из файла `test.txt`, от первой и второй колонок того же файла. Вид регрессии описывается формулой `f`, которая, очевидно, удовлетворяет требованиям, предъявляемым к виду линейной регрессионной зависимости. Описатель построенной модели сохраняется функцией `lm()` в переменную `m`.

Для просмотра описания построенной регрессии можно воспользоваться функцией `summary()`, единственный параметр которой - переменная, описывающая регрессионную модель (в листинге 10 - переменная `m`).

1.3.2.2 Демонстрация построения регрессии

На рисунках 21, 22, 23 и 24 приведен пример расчета оптимального набора значений параметров функционирования кластера.

В данном примере минимизируется время прогона кластера по количеству вычислительных узлов, задействованных в прогоне, и по количеству вычислительных потоков, по которым распределяется задача, решаемая кластером. Тип вещественных данных - `float`, интенсивность обмена данными - низкая.

В приводимом примере регрессия рассчитывается по функции (6).

$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + (\beta_1 x_1)^2 + (\beta_2 x_2)^2 \quad (6)$$

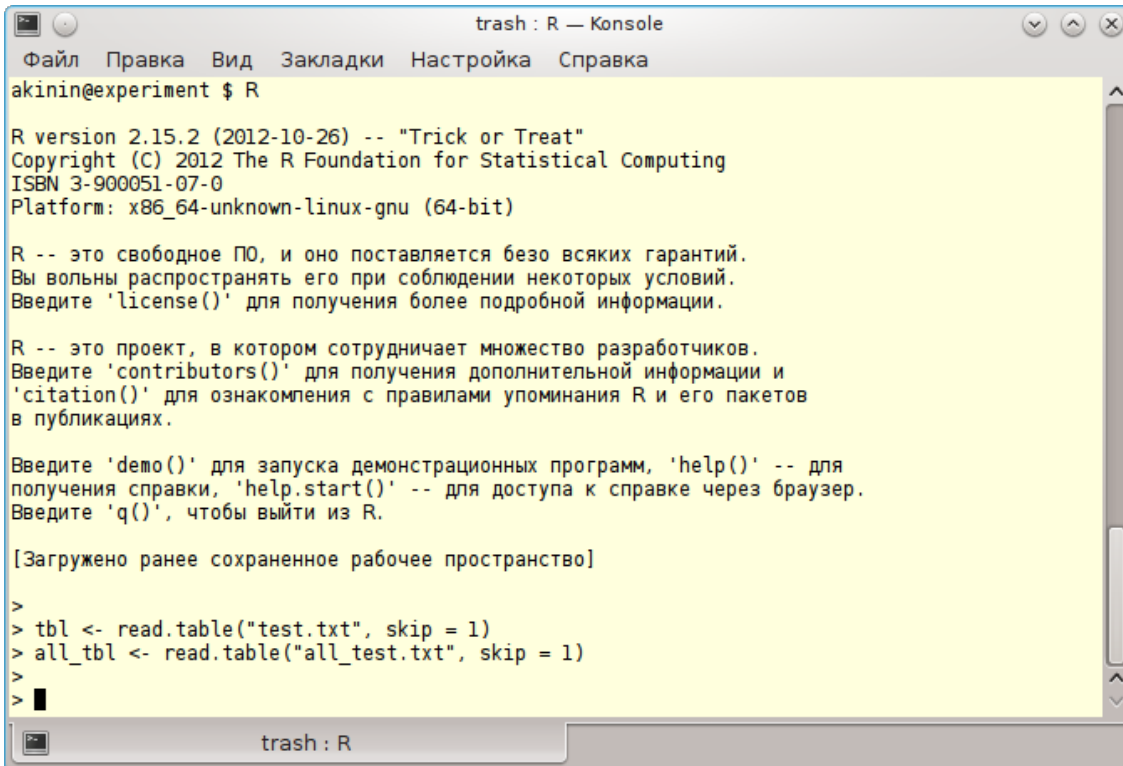
Параметры регрессии:

- $\beta_1 = -0.523925$;
- $\beta_2 = 0.156524$;
- $\beta_3 = -0.007311$.

Оптимальные параметры прогона кластера:

- количество вычислительных узлов кластера, участвующих в решении задачи - 4;
- количество вычислительных потоков, решающих задачу - 3.

Полученные оптимальные параметры прогона кластера совпадают с аналогичными, полученными в предыдущих лабораторных работах.



```
trash : R — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
akini@experiment $ R

R version 2.15.2 (2012-10-26) -- "Trick or Treat"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-unknown-linux-gnu (64-bit)

R -- это свободное ПО, и оно поставляется безо всяких гарантий.
Вы вольны распространять его при соблюдении некоторых условий.
Введите 'license()' для получения более подробной информации.

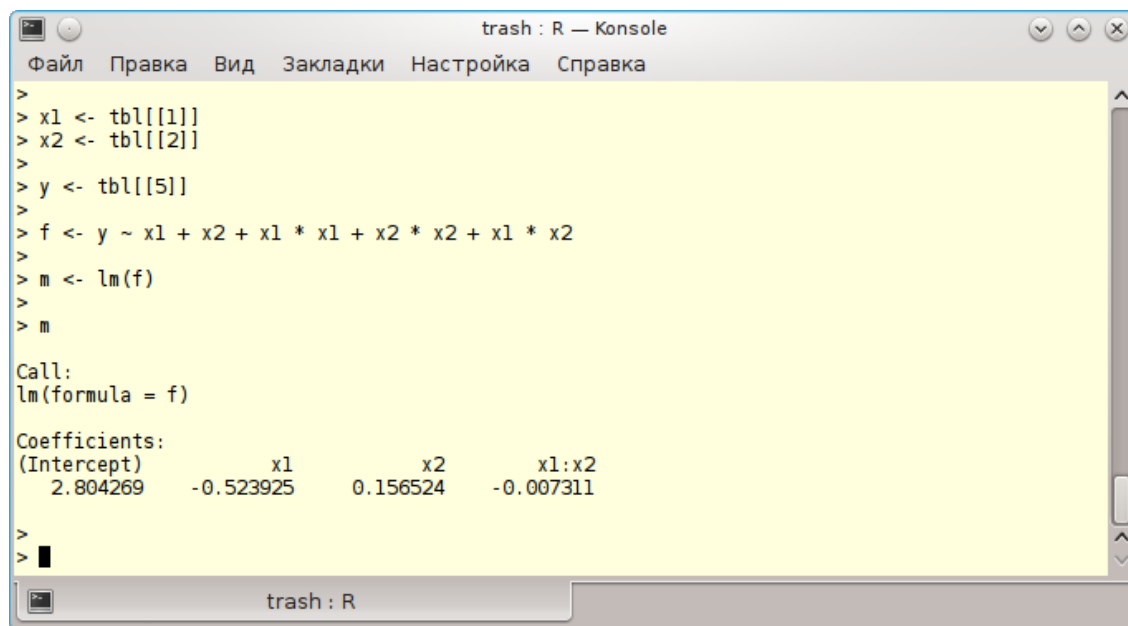
R -- это проект, в котором сотрудничает множество разработчиков.
Введите 'contributors()' для получения дополнительной информации и
'citation()' для ознакомления с правилами упоминания R и его пакетов
в публикациях.

Введите 'demo()' для запуска демонстрационных программ, 'help()' -- для
получения справки, 'help.start()' -- для доступа к справке через браузер.
Введите 'q()', чтобы выйти из R.

[Загружено ранее сохраненное рабочее пространство]

>
> tbl <- read.table("test.txt", skip = 1)
> all_tbl <- read.table("all_test.txt", skip = 1)
>
> █
```

Рисунок 21 — Загрузка таблиц с результатами эксперимента



```
trash : R — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка

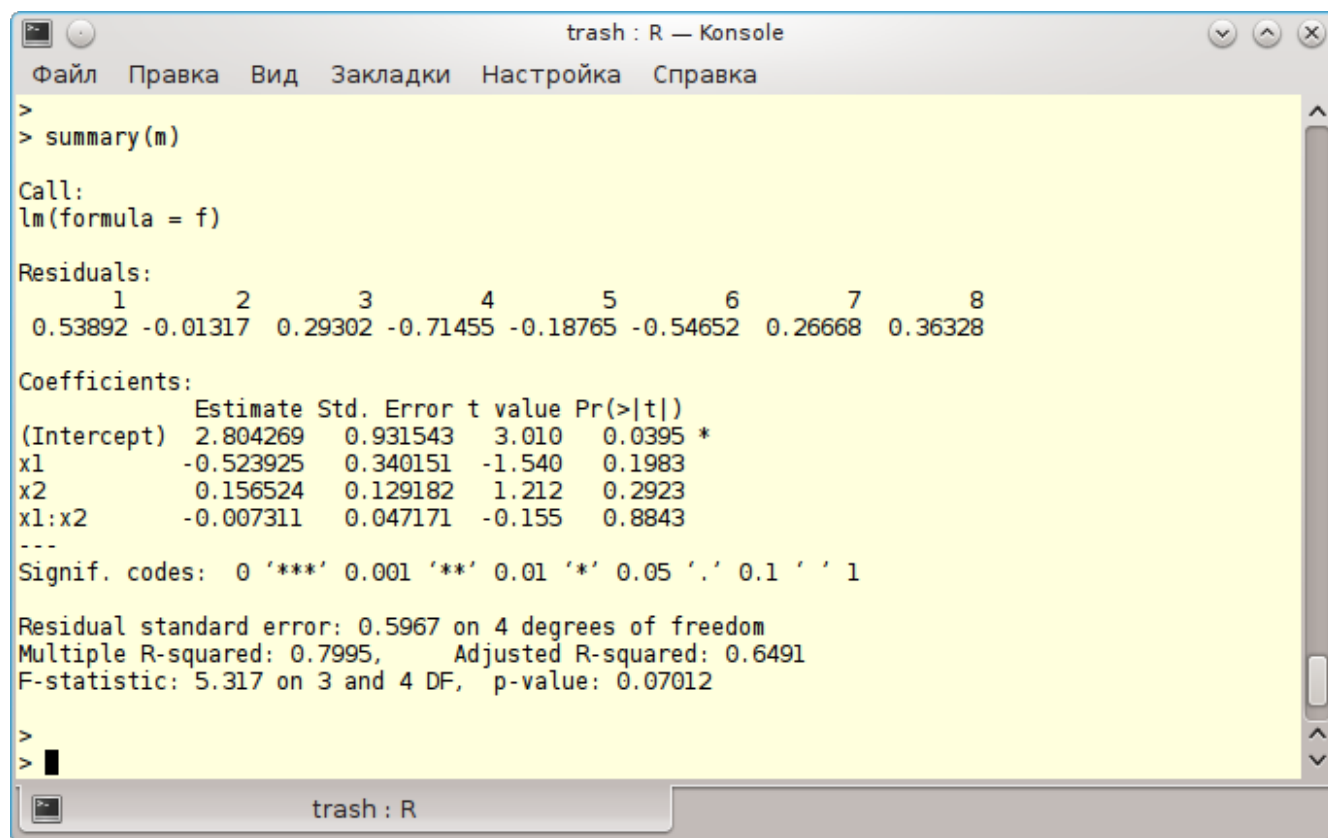
>
> x1 <- tbl[[1]]
> x2 <- tbl[[2]]
>
> y <- tbl[[5]]
>
> f <- y ~ x1 + x2 + x1 * x1 + x2 * x2 + x1 * x2
>
> m <- lm(f)
>
> m

Call:
lm(formula = f)

Coefficients:
(Intercept)      x1      x2    x1:x2
  2.804269   -0.523925   0.156524  -0.007311

>
> 
```

Рисунок 22 — Расчет линейной регрессионной модели



```
trash : R — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка

>
> summary(m)

Call:
lm(formula = f)

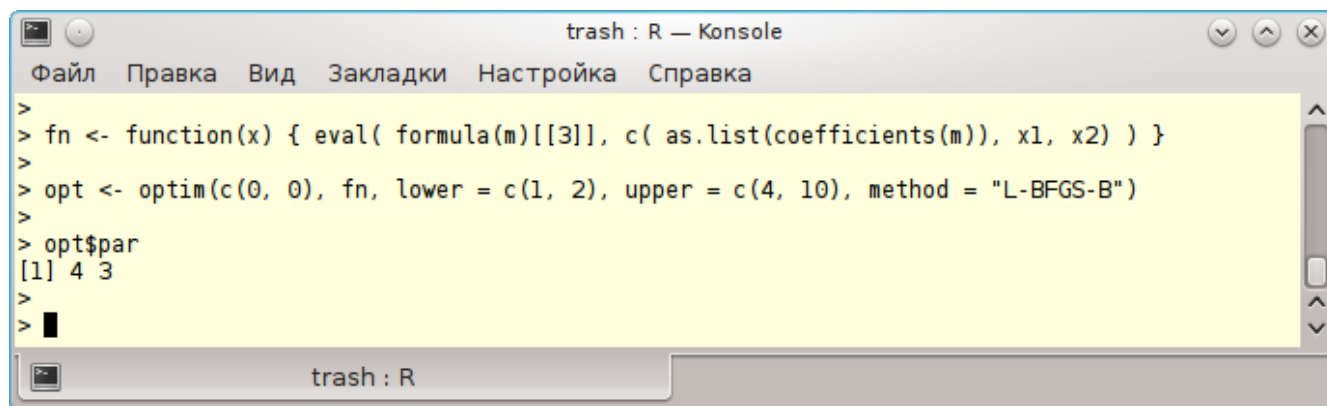
Residuals:
    1     2     3     4     5     6     7     8 
0.53892 -0.01317  0.29302 -0.71455 -0.18765 -0.54652  0.26668  0.36328 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.804269   0.931543   3.010   0.0395 *
x1          -0.523925   0.340151  -1.540   0.1983
x2           0.156524   0.129182   1.212   0.2923
x1:x2       -0.007311   0.047171  -0.155   0.8843
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5967 on 4 degrees of freedom
Multiple R-squared:  0.7995,    Adjusted R-squared:  0.6491 
F-statistic: 5.317 on 3 and 4 DF,  p-value: 0.07012

>
> 
```

Рисунок 23 — Получение общей информации о рассчитанной регрессионной модели



```
>
> fn <- function(x) { eval( formula(m)[[3]], c( as.list(coefficients(m)), x1, x2) ) }
>
> opt <- optim(c(0, 0), fn, lower = c(1, 2), upper = c(4, 10), method = "L-BFGS-B")
>
> opt$par
[1] 4 3
>
> █
```

Рисунок 24 — Расчет оптимальных параметров прогона вычислительного кластера

1.3.3 Задание

Лабораторная работа выполняется побригадно - бригада может состоять из одного или нескольких человек (не более трех) - всего допускается существование восьми бригад.

Каждая из бригад получает свой уникальный набор значений параметров «Тип вещественных данных» и «Интенсивность обмена данными между вычислительными узлами, составляющими кластер»:

- 1) «Тип вещественных данных» - float (вещественные одинарной точности);
«Интенсивность обмена данными между вычислительными узлами» - средняя;
- 2) «Тип вещественных данных» - float (вещественные одинарной точности);
«Интенсивность обмена данными между вычислительными узлами» - высокая;
- 3) «Тип вещественных данных» - double (вещественные двойной точности);
«Интенсивность обмена данными между вычислительными узлами» - низкая;
- 4) «Тип вещественных данных» - double (вещественные двойной точности);
«Интенсивность обмена данными между вычислительными узлами» - средняя;
- 5) «Тип вещественных данных» - double (вещественные двойной точности);
«Интенсивность обмена данными между вычислительными узлами» - высокая;

- 6) «Тип вещественных данных» - long double (вещественные максимальной точности);
«Интенсивность обмена данными между вычислительными узлами» - низкая;
- 7) «Тип вещественных данных» - long double (вещественные максимальной точности);
«Интенсивность обмена данными между вычислительными узлами» - средняя;
- 8) «Тип вещественных данных» - long double (вещественные максимальной точности);
«Интенсивность обмена данными между вычислительными узлами» - высокая.

Для выполнения лабораторной работы необходимо:

- выбрать вид регрессии времени решения задачи на кластере от количества вычислительных узлов, задействованных в прогоне кластера, и от количества вычислительных потоков, по которым распределяется задача в ходе своего решения;
- рассчитать регрессию времени решения задачи от количества вычислительных узлов и от количества вычислительных потоков по результатам экспериментов, выполненных в предыдущих лабораторных работах;
- найти оптимальные параметры функционирования кластера.

В случае, если найденные параметры не являются оптимальными по той или иной оценки - необходимо выполнить лабораторную работу заново с другим видом регрессии.

Расчет регрессии и подбор оптимальных параметров функционирования кластера необходимо выполнить с помощью языка программирования R.

Отчет по лабораторной работе оформляется индивидуально каждым студентом.

1.3.4 Требования к оформлению отчета по лабораторной работе

Отчет по лабораторной работе должен содержать:

- титульный лист, оформленный в соответствии с требованиями преподавателя, кафедры и университета и действующих государственных стандартов по оформлению титульных листов отчетов по лабораторным работам;

- задание к лабораторной работе;
- скриншоты, иллюстрирующие процесс загрузки данных в интерпретатор языка программирования R, процесс расчета регрессии и процесс подбора оптимальных параметров функционирования кластера;
- скриншоты, содержащие вывод результатов эксперимента, вывод вида регрессии, вывод параметров регрессии, вывод оптимальных значений параметров функционирования кластера - все объекты должны быть выведены на экран средствами интерпретатора языка программирования R;
- вывод.

Вывод по выполнению лабораторной работы должен содержать:

- выводы о найденных оптимальных значениях параметров регрессии;
- заключение об оптимальности (или отсутствию таковой) подобранных параметров функционирования кластера.

1.4 Лабораторная работа № 4.

Симплекс - метод планирования эксперимента

1.4.1 Цель работы

- получить навыки оценки оптимальных параметров функционирования объекта исследований с помощью симплекс-метода;
- получить навыки использования языка программирования R для машинной проверки результатов расчета симплекс-метода.

1.4.2 Теоретическая часть

1.4.2.1 Симплекс - метод

Симплекс - метод — алгоритм решения оптимизационной задачи линейного программирования путем перебора вершин выпуклого многогранника в многомерном пространстве. Метод был разработан американским математиком Джорджем Данцигом (George Dantzig) в 1947 году.

Задача линейного программирования состоит в том, что необходимо максимизировать или минимизировать некоторый линейный функционал на многомерном пространстве при заданных линейных ограничениях.

Последовательность вычислений симплекс-методом можно разделить на две основные фазы:

- нахождение исходной вершины множества допустимых решений;
- последовательный переход от одной вершины к другой, ведущий к оптимизации значения целевой функции.

1.4.2.2 Задача линейного программирования

Задача линейного программирования в общем виде формулируется следующим образом.

Необходимо найти экстремум целевой функции (7) при ограничениях (8).

$$y = \sum_{j=1}^n c_j x_j \quad (7)$$

$$a_j x_j (<, \leq, \geq, >, =) b_i, x_j \geq 0 \quad (8)$$
$$j = \overline{1, n}; i = \overline{1, m}$$

Экономическая интерпретация моделей линейного программирования состоит в следующем: моделируемая система характеризуется наличием нескольких видов «производственной деятельности» j , для осуществления которых требуются имеющиеся в ограниченном количестве ресурсы b_i . Расход j -ого ресурса на 1 продукт равен a_{ij} . При таком потреблении результат j -ого вида «производственной деятельности» для единицы соответствующего продукта (удельная стоимость или прибыль) характеризуется c_i . Цель построения модели состоит в определении уровней или объемов каждого вида производственной деятельности, при которой оптимизируется общий результат деятельности всей системы в целом без нарушения ограничений, накладываемых на используемые ресурсы. Функцию y называют целевой функцией, а лимиты потребления ресурсов называют ограничениями.

При получении линейных моделей принимаются следующие допущения: пропорциональность, аддитивность и неотрицательность.

Пропорциональность означает, что затраты ресурсов на некоторый вид «производственной деятельности», а так же вклад в целевую функцию прямо пропорциональны его уровню.

Аддитивность означает, что общая величина ресурсов потребляемых системе всеми видами «производственной деятельности» равна сумме затрат ресурсов на отдельные виды «производственной деятельности».

Неотрицательность означает, что ни одному виду «производственной деятельности» не может быть приписано отрицательное значение. На практике такое требование часто является логическим следствием реального функционирования системы. Однако целый ряд задач этому требованию не отвечают. Это модели целевого программирования, для которых уровень «производственной деятельности» может и не иметь ограничения в знаке.

Такие ограничения могут быть заданы в виде: $\sum_{j=1}^n a_{ij}x_j = b_j; i = \overline{1, m}$.

Модели целевого проектирования приводятся к стандартному виду с использованием функций штрафа. Для этого в левую часть ограничения добавляют некоторую неограниченную в знаке переменную, которая характеризует величину нарушения. Для того чтобы использовать аппарат линейного программирования, данную переменную заменяют подстановкой вида: $y_i = y_i^+ + y_i^-; y_i^+; y_i^- \geq 0$.

Модели линейного программирования чаще детерминированные, но имеются их интерпретации и для решения задач вероятностного характера. Одним из примеров такой задачи является транспортная задача, в которой спрос в местах доставки является величиной случайной. Элементы неопределенности могут включаться и в ограничения задачи. Общим методом решения задач такого вида является их сведение, часто даже приближенными методами, к эквивалентным детерминированным соотношениям. Методы линейного программи-

рования повсеместно используются и для решения нелинейных задач. В этом случае, нелинейные задачи обычно аппроксимируются линейными функциями.

1.4.2.3 Усиленная постановка задачи

Рассмотрим следующую задачу линейного программирования:

$$c^T x \rightarrow \max, Ax \leq b, x \geq 0$$

Теперь поставим эту задачу в эквивалентной усиленной форме. Необходимо максимизировать Z , где:

$$\begin{bmatrix} 1 & -c^T & 0 \\ 0 & A & E \end{bmatrix} \begin{bmatrix} Z \\ x \\ x_s \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}$$

$$x, x_s \geq 0$$

Здесь:

- x — переменные из исходного линейного функционала;
- x_s — новые переменные, дополняющие старые таким образом, что неравенство переходит в равенство;
- c — коэффициенты исходного линейного функционала;
- Z — переменная, которую необходимо максимизировать.

Полупространства в пересечении образуют многогранник, представляющий множество допустимых решений. Разница между числом переменных и уравнений дает нам число степеней свободы. Проще говоря, если мы рассматриваем вершину многогранника, то это число ребер, по которым мы можем продолжать движение. Тогда мы можем присвоить этому числу переменных значение 0 и назвать их «непростыми». Остальные переменные при этом будут вычисляться однозначно и называться «простыми». Полученная точка будет вершиной в пересечении соответствующих непростым переменным гиперплоскостей. Для того, чтобы найти т. н. начальное допустимое решение (вершину, из которой мы начнем движение), присвоим всем изначальным переменным x значение 0 и будем их считать непростыми, а все новые будем считать простыми. При этом начальное допустимое решение вычисляется однозначно: $x_{si} = b_i$.

1.4.2.4 Алгоритм симплекс - метода

Теперь приведем шаги алгоритма. На каждом шаге мы будем менять множества простых и непростых векторов (двигаться по ребрам), и матрица будет иметь вид (9).

$$\begin{bmatrix} 1 & c_B^T B^{-1} A - c^T & c_B^T B^{-1} \\ 0 & B^{-1} A & B^{-1} \end{bmatrix} \begin{bmatrix} Z \\ x \\ x_s \end{bmatrix} = \begin{bmatrix} c_B^T B^{-1} b \\ B^{-1} b \end{bmatrix} \quad (9)$$

Здесь C^b — коэффициенты вектора с соответствующие простым переменным (переменным x_s соответствуют нули), B — столбцы, соответствующие простым переменным. Матрицу, образованную оставшимися столбцами обозначим D . Почему матрица будет иметь такой вид поясним в описании шагов алгоритма.

Шаги алгоритма:

— первый шаг.

Выбираем начальное допустимое значение, как указано выше. На первом шаге B — единичная матрица, так как простыми переменными являются x_s . c_B — нулевой вектор по тем же причинам;

— второй шаг.

Покажем, что в выражении $(c_B^T B^{-1} A - c^T) x + (c_B^T B^{-1}) x_s$ только непростые переменные имеют ненулевой коэффициент. Заметим, что из выражения $Ax + x_s = b$ простые переменные однозначно выражаются через непростые, так как число простых переменных равно числу уравнений. Пусть x' — простые, а x'' — непростые переменные на данной итерации. Уравнение $Ax + x_s = b$ можно переписать, как $Bx' + Dx'' = b$. Умножим его на B^{-1} слева: $x' + B^{-1} Dx'' = B^{-1} b$.

Выберем ребро, по которому мы будем перемещаться. Поскольку мы хотим максимизировать Z , то необходимо выбрать переменную, которая будет более всех уменьшать выражение $(c_B^T B^{-1} A - c^T) x + (c_B^T B^{-1}) x_s$.

Для этого выберем переменную, которая имеет наибольший по модулю отрицательный коэффициент. Если таких переменных нет, то есть все коэффициенты этого выражения неотрицательны, то мы пришли в искомую вершину и нашли оптимальное решение. В противном случае начнем увеличивать эту непростую переменную, то есть перемещаться по соответствующему ей ребру. Эту переменную назовем входящей;

— третий шаг.

Теперь необходимо понять, какая простая переменная первой обратится в ноль по мере увеличения входящей переменной. Для этого достаточно

рассмотреть систему:

$$\begin{bmatrix} B^{-1} & A & B^{-1} \end{bmatrix} \begin{bmatrix} x \\ x_s \end{bmatrix} = B^{-1}b$$

При фиксированных значениях непростых переменных система однозначно разрешима относительно простых, поэтому мы можем определить, какая из простых переменных первой достигнет нуля при увеличении входящей. Эту переменную назовем выходящей. Это будет означать, что мы натолкнулись на новую вершину. Теперь входящую и выходящую переменную поменяем местами — входящая «войдет» в простую, а выходящая из них «выйдет» в непростые. Теперь перепишем матрицу B и вектор c_B в соответствии с новыми наборами простых и непростых переменных, после чего вернемся ко второму шагу.

Поскольку число вершин конечно, то алгоритм сходится. Найденная вершина будет являться оптимальным решением.

Для реализации двойственного метода необходимо перейти от задачи на минимум к задаче на максимум путем транспонирования матрицы коэффициентов.

1.4.2.5 Пример реализации симплекс-метода

Начальные условия:

$$\begin{aligned} y &= 12x_1 + x_2 - 2x_3 \rightarrow \max \\ -2x_1 - x_2 &\geq -10 \\ 2x_1 - x_2 - 2x_3 &\leq 8 \\ -x_1 + x_3 &\leq -2 \end{aligned}$$

Решение задачи:

$$\begin{aligned} y &= 0 - (-12x_1 - x_2 + 2x_3) \\ x_4 &= 10 - (2x_1 + x_2) \\ x_5 &= 8 - (2x_1 - x_2 - 2x_3) \\ x_6 &= -2 - (-x_1 + x_3) \end{aligned}$$

Нам необходимо найти начальное опорное (абсолютно произвольное) решение для функции L , которое бы удовлетворяло системе наложенных ограничений. Далее, применяя симплекс таблицы, мы будем получать решения, при которых значение функции будет, как минимум, не убывать. И так до тех пор,

пока не достигнем оптимально решения, при котором функция достигает своего максимума. Если, конечно, рассматриваемая нами линейная функция обладаем максимальным значением при заданной системе ограничений. Перед применением симплекс таблиц, необходимо преобразовать систему линейных ограничений и рассматриваемую нами функцию L к стандартному виду.

Свободные члены системы ограничений должны быть неотрицательными.

Умножим коэффициенты ограничения 1 на -1, свободный член ограничения станет неотрицательным.

Умножим коэффициенты ограничения 3 на -1, свободный член ограничения станет неотрицательным.

$$\begin{aligned} 2x_1 + x_2 &\leq 10 \\ 2x_1 - x_2 - 2x_3 &\leq 8 \\ x_1 - x_3 &\geq 2 \end{aligned}$$

Определимся с начальным опорным решением.

Наличие единичного базиса в системе ограничений позволяет легко найти начальное опорное решение. Рассмотрим подробнее:

Переменная x_4 входит в уравнение 1 с коэффициентом 1, а в остальные уравнения системы с коэффициентом ноль, т.е. x_4 - базисная переменная.

Переменная x_5 входит в уравнение 2 с коэффициентом 1, а в остальные уравнения системы с коэффициентом ноль, т.е. x_5 - базисная переменная.

В уравнении 3 нет переменной, которая входила бы в него с коэффициентом 1, а в остальные уравнения системы входила бы с коэффициентом ноль. Добавим к данному уравнению искусственную переменную r_1 . Очевидно, переменная r_1 будет являться базисной переменной, т.к. входит в уравнение 3 с коэффициентом 1 и не входит в оставшиеся уравнения системы ограничений.

$$\begin{aligned} 2x_1 + x_2 + x_4 &= 10 \\ 2x_1 - x_2 - 2x_3 + x_5 &= 8 \\ x_1 - x_3 + r_1 &= 2 \end{aligned}$$

Переменные, которые не являются базисными называются свободными переменными. Приравняв свободные переменные нулю в получившийся системе ограничений мы получим начальное решение.

$$X_{\text{нач}} = \{0, 0, 0, 10, 8, 0, 2\}$$

Для получения единичного базиса нам пришлось ввести искусственные переменные, поэтому наше начальное решение является псевдорешением.

Для нахождения начального опорного решения функции L , сначала придется решить вспомогательную задачу.

Введем в рассмотрение вспомогательную функцию $W = -r_1$. Найдем наибольшее значение функции W .

Схема решения вспомогательной задачи абсолютно аналогична схеме описанной выше. Есть только одно исключение: вспомогательная задача всегда имеет решение. В процессе решения данной задачи возможны два варианта.

Если максимальное значение вспомогательной функции W равно нулю, т.е. все искусственные переменные обращаются в нуль - это будет свидетельствовать о том, что мы нашли начальное опорное решение функции L .

В противном случае, не существует решений, удовлетворяющих системе ограничений нашей задачи.

Функция L и вспомогательная функция W не должны содержать базисных переменных.

Из уравнения 3 последней системы выразим r_1 и подставим в выражение функции W , получим: $W = -2 + x_1 - x_3 - x_6$.

Значение функции W для начального решения: $W(X_{\text{нач}}) = -2$.

Вернемся к рассмотрению функции $L = 12x_1 + x_2 - 2x_3$.

Функция L и вспомогательная функция W не содержат базисных переменных.

Для составления начальной симплекс - таблицы мы выполнили все условия.

— шаг 1.

За ведущий выберем столбец 1, так как -1 наименьший элемент в W строке. Элемент W строки, принадлежащий столбцу свободных членов не рассматриваем.

За ведущую выберем строку 3, так как отношение свободного члена к соответствующему элементу выбранного столбца для 3 строки является наименьшим. Обратите внимание, что отношение мы вычисляем только для положительных элементов столбца 1.

Базисные переменные	x_1	x_2	x_3	x_4	x_5	x_6	r_1	Свободные члены	Отношение
x_4								10	5
x_5								8	4
r_1								2	2
L	-12	-1	2	0	0	0	0	0	-
W	-1	0	1	0	0	1	0	-2	-

От элементов строки 1 отнимает соответствующие элементы строки 3,

умноженные на 2.

От элементов строки 2 отнимает соответствующие элементы строки 3, умноженные на 2.

От элементов строки L отнимает соответствующие элементы строки 3, умноженные на -12.

От элементов строки W отнимает соответствующие элементы строки 3, умноженные на -1.

Элементы столбца r_1 можно не пересчитывать, так как переменная r_1 больше не является базисной.

Базисные перемен- ные	x_1	x_2	x_3	x_4	x_5	x_6	Свободные члены
x_4							6
x_5							4
r_1							2
L	0	-1	-10	0	0	-12	24
W	0	0	0	0	0	0	0

$$x_1 = \{2, 0, 0, 6, 4, 0\}; W = 0$$

Значение функции W для данного решения: $W(x_1) = 0$.

Находим начальное опорное решение:

$$X_{\text{нач.}} = \{2, 0, 0, 6, 4, 0\}$$

$$L = 24 + x_2 + 10x_3 + 12x_6$$

Значение функции для данного решения: $L(X_{\text{нач.}}) = 24$;

— шаг 2.

За ведущий выберем столбец 6, так как -12 наименьший элемент в L строке. Элемент L строки, принадлежащий столбцу свободных членов не рассматриваем.

За ведущую выберем строку 2, так как отношение свободного члена к соответствующему элементу выбранного столбца для 2 строки является наименьшим. Обратите внимание, что отношение мы вычисляем только для положительных элементов столбца 6:

Базисные переменные	x_1	x_2	x_3	x_4	x_5	x_6	Свободные члены	Отношение
x_4							6	3
x_5							4	2
x_1							2	-
L	0	-1	-10	0	0	-12	24	-

Разделим элементы строки 2 на 2:

Базисные переменные	x_1	x_2	x_3	x_4	x_5	x_6	Свободные члены	Отношение
x_4							6	3
x_5							2	2
x_1							2	-
L	0	-1	-10	0	0	-12	24	-

От элементов строки 1 отнимает соответствующие элементы строки 2, умноженные на 2.

От элементов строки 3 отнимает соответствующие элементы строки 2, умноженные на -1.

От элементов строки L отнимает соответствующие элементы строки 2, умноженные на -12:

Базисные переменные	x_1	x_2	x_3	x_4	x_5	x_6	Свободные члены
x_4							2
x_6							2
x_1							4
L	0	-7	-10	0	6	0	48

$$x_1 = \{4, 0, 0, 2, 0, 2\}; L = 48 + 7x_2 + 10x_3 - 6x_5.$$

Значение функции L для данного решения: $L(x_1) = 48$.

— шаг 3.

За ведущий выберем столбец 3, так как -10 наименьший элемент в L стро-

ке. Элемент L строки, принадлежащий столбцу свободных членов не рассматриваем.

За ведущую выберем строку 1, так как отношение свободного члена к соответствующему элементу выбранного столбца для 1 строки является наименьшим. Обратим внимание, что отношение мы вычисляем только для положительных элементов столбца 3.

Базисные переменные	x_1	x_2	x_3	x_4	x_5	x_6	Свободные члены	Отношение
x_4							6	3
x_5							2	2
x_1							2	-
L	0	-1	-10	0	0	-12	24	-

Базисные переменные	x_1	x_2	x_3	x_4	x_5	x_6	Свободные члены	Отношение
x_4							2	1
x_6							2	-
x_1							4	-
L	0	-7	-10	0	6	0	48	-

Разделим элементы строки 1 на 2:

Базисные переменные	x_1	x_2	x_3	x_4	x_5	x_6	Свободные члены	Отношение
x_4							1	1
x_6							2	-
x_1							4	-
L	0	-7	-10	0	6	0	48	-

От элементов строки 3 отнимает соответствующие элементы строки 1, умноженные на -1.

От элементов строки L отнимает соответствующие элементы строки 1, умноженные на -10.

Базисные перемен- ные	x_1	x_2	x_3	x_4	x_5	x_6	Свободные члены
x_3							1
x_6							2
x_1							5
L	0	3	0	5	1	0	58

$$x_2 = \{5, 0, 1, 0, 0, 2\}, L = 58 - 3x_2 - 5x_4 - x_5.$$

Значение функции L для данного решения: $L(x_2) = 58$.

Учитывая, что все $x_i \geq 0$, по условию задачи, наибольшее значение функции L равно свободному члену 58, т.е. мы получили оптимальное решение.

Итоговое количество проделанных итераций: 3.

1.4.2.6 Решение системы уравнений симплекс - метода средствами языка программирования R

Любой может загрузить библиотеку R, используя функцию `library()`. Если вы хотите узнать список функций и набора данных включенных в библиотеку, достаточно использовать аргумент `h` или `help` для функции `library()`.

Стандартная библиотека языка программирования R содержит функционал, предназначенный для решения системы линейных уравнений - функцию `solve()`, для создания матрицы может быть использована функция `matrix()`, описанные подробно в лабораторной работе № 1 и 2.

Для генерации любой последовательности и вывода значения с условием используется команда `ifelse()`. Пример использования команды приведен в листинге 11.

```
> x <- 1 : 10
> ifelse(x < 5 | x > 8, x, 0)
[1] 1 2 3 4 0 0 0 0 9 10
```

Листинг 11 — Генерация последовательности и вывод значений с условием

В листинге 11 - генерируем последовательность от 1 до 10 и выводим значения меньше чем 5 и больше чем 8.

R позволяет реализовать циклы тремя способами:

— циклы `for`;

- циклы с предусловием (**while**);
- «Бесконечные» циклы (**repeat**).

Для использования оператора **for** требуется указать индекс и вектор, а также указать выполняемую конструкцию и указать повторяемые операторы в фигурных скобках. Механизм работы следующий:

- интерпретатор последовательно выбирает значения из вектора и присваивает их переменной;
- с каждым значением переменной выполняется список функций в фигурных скобках.

```
> for (i in -1 : 11)
+ print(i)
[1] -1
[1] 0
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
[1] 11
```

Листинг 12 — Механизм реализации цикла **for()**

Синтаксис цикла **while** вполне стандартен: **while(условие выполнимости)** {список функций для итерированного выполнения}.

```
> g <- 0
> while (gi < 1) {
+   g <- rnorm(1)
+   cat(g, "/n")
+ }
-0.08111594
0.1732847
0.2428368
0.3359238
-0.2080000
0.05458533
0.2627001
1.009195
```

Листинг 13 — Реализация цикла while()

Определение функции - это присвоение блока операторов переменной.

Синтаксис: переменная <- function (список аргументов) {тело функции}.

```
> fn = function (a) {
+   if (a > 3) {
+     print(10);
+   } else {
+     print(15);
+   }
+ }
> fn(1)
[1] 15
> fn(7)
[1] 10
```

Листинг 14 — Определение функции

В листингах 15 и 16 приведены примеры задания матриц и вычисления системы линейных уравнений в соответствии с заданными значениями.

```
> a1 <- matrix(c(
  (0,0,0,-12,-1,0,0,0,-1,0,0,0,0,2,1,0,0,0,0,0,0,0,0,0,0,0),
  1,0,0,0,0,10,8,2,0,-2,5,4,2,0,0),ncol=9)
> a1
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]
[1,]	0	0	0	0	0	0	1	10	5
[2,]	0	0	0	0	0	0	0	8	4
[3,]	0	0	0	0	0	0	0	2	2
[4,]	-12	-1	2	0	0	0	0	0	0
[5,]	-1	0	1	0	0	0	0	-2	0

Листинг 15 — Пример задания матрицы

В листинге 16 представлен один из вариантов нахождения начального решения из примера 1. Учитывая, что базисная переменная **a4** принята нами за 1 в рамках построенной модели, ответ согласуется с аналитическим решением. Аналогичным образом, задавая матрицу коэффициентов, строятся решения остальных базисных переменных и производится построение и решение симплекс - таблиц.

```
> a1 <- matrix(c(2,2,1),ncol=1)
> a2 <- matrix(c(1,-1,0),ncol=1)
> a3 <- matrix(c(0,-2,1),ncol=1)
> a4 <- matrix(c(1,0,0),ncol=1)
> a5 <- matrix(c(0,1,0),ncol=1)
> a6 <- matrix(c(0,0,-1),ncol=1)
> a7 <- matrix(c(0,0,1),ncol=1)
```



```

> a1
      [,1]
[1,]    2
[2,]    2
[3,]    1
> a2
      [,1]
[1,]    1
[2,]   -1
[3,]    0
> a3
      [,1]
[1,]    0
[2,]   -2
[3,]    1
> a4
      [,1]
[1,]    1
[2,]    0
[3,]    0
> a5
      [,1]
[1,]    0
[2,]    1
[3,]    0
> b1 <- c(10,0,0)
> b1
[1] 10 0 0
> sum(a1*0+a2*0+a4*1-b1)
[1] -9

```

Листинг 16 — Пример вычисления уравнений

1.4.3 Задание

В соответствии с номером бригады рассчитать заданные уравнения симплекс-методом, сравнить полученные значения с машинным расчетом и сделать вывод по полученным данным.

1)

$$y = x_1 + 5x_2 + x_3 \rightarrow \max$$

$$4x_1 - 2x_2 + 2x_3 = 3$$

$$2x_1 + x_2 + x_3 \leq 6$$

$$2x_1 + 2x_2 + x_3 \geq 1$$

2)

$$\begin{aligned}
y &= 5x_1 + 8x_2 + x_3 \rightarrow \min \\
x_1 - 2x_2 + 8x_3 &\geq 4 \\
7x_1 + 3x_2 + 11x_3 &\leq 11 \\
5x_1 - 1x_2 + x_3 &\leq 3
\end{aligned}$$

3)

$$\begin{aligned}
y &= 7x_1 + x_2 + 2x_3 \rightarrow \max \\
13x_1 - 2x_2 + 6x_3 &\leq 3 \\
6x_1 + x_2 + 8x_3 &\leq 2 \\
4x_1 - x_2 + x_3 &\leq 3
\end{aligned}$$

4)

$$\begin{aligned}
y &= 7x_1 + 2x_2 + 4x_3 \rightarrow \min \\
5x_1 + x_2 + 7x_3 &\leq 1 \\
19x_1 + 6x_2 + 3x_3 &\geq 5 \\
7x_1 + 3x_2 + 2x_3 &\geq 1
\end{aligned}$$

5)

$$\begin{aligned}
y &= 5x_1 + 14x_2 + 8x_3 \rightarrow \max \\
3x_1 + 15x_2 + 7x_3 &\leq 1 \\
5x_1 + 9x_2 + 2x_3 &\leq 2 \\
x_1 + x_2 + 14x_3 &\geq -1
\end{aligned}$$

6)

$$\begin{aligned}
y &= 4x_1 + 2x_2 + 9x_3 \rightarrow \max \\
5x_1 + 2x_2 + 7x_3 &\geq 1 \\
5x_1 + 6x_2 + 2x_3 &\leq 1 \\
x_1 + x_2 + 2x_3 &\geq -1
\end{aligned}$$

7)

$$\begin{aligned}
y &= 4x_1 + 2x_2 + 2x_3 \rightarrow \max \\
7x_1 + 3x_2 + 5x_3 &\geq 1 \\
3x_1 + 2x_2 + x_3 &\leq 1 \\
4x_1 + x_2 + 6x_3 &\geq 2
\end{aligned}$$

8)

$$\begin{aligned}y &= 7x_1 + 4x_2 + x_3 \rightarrow \max \\4x_1 + 7x_2 + 2x_3 &\leq 3 \\2x_1 + 2x_2 + x_3 &\leq 8 \\11x_1 + 1x_2 + 6x_3 &\leq 4\end{aligned}$$

- в соответствии с вариантом задания выполнить ручной расчет на основе симплекс-метода;
- основываясь на базовых командах языка программирования R, выполнить аналогичные вычисления в командной строке интерпретатора;
- сравнить полученные результаты машинной обработки с ручным расчетом;
- сделать вывод по полученным данным.

1.4.4 Требования к оформлению отчета по лабораторной работе

Отчет по лабораторной работе должен содержать:

- титульный лист, оформленный в соответствии с требованиями преподавателя;
- задание к лабораторной работе;
- ручной расчет в соответствии с вариантом задания;
- скриншоты, содержащие вывод результатов с использованием интерпретатора языка программирования R.

Список литературы

1. The R Project for Statistical Computing [Электронный ресурс]. — URL: <http://www.r-project.org>, Дата обращения: 01.12.2011.
2. LXF100-101:R - Linuxformat [Электронный ресурс]. — URL: <http://wiki.linuxformat.ru/index.php/LXF100-101:R>, Дата обращения: 01.12.2011.
3. R Programming - Wikibooks, open books for an open world [Электронный ресурс]. — URL: http://en.wikibooks.org/wiki/R_Programming, Дата обращения: 30.01.2012.
4. Симплекс метод - О методе [Электронный ресурс]. — URL: <http://simplex-metod.narod.ru>, Дата обращения: 30.01.2012.