ГОУ ВПО

Уфимский государственный авиационный технический университет

Кафедра вычислительной математики и кибернетики

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| **100** |  |  |  |  |  |  |  |  |  |  |  |  |
| **90** |  |  |  |  |  |  |  |  |  |  |  |  |
| **80** |  |  |  |  |  |  |  |  |  |  |  |  |
| **70** |  |  |  |  |  |  |  |  |  |  |  |  |
| **60** |  |  |  |  |  |  |  |  |  |  |  |  |
| **50** |  |  |  |  |  |  |  |  |  |  |  |  |
| **40** |  |  |  |  |  |  |  |  |  |  |  |  |
| **30** |  |  |  |  |  |  |  |  |  |  |  |  |
| **20** |  |  |  |  |  |  |  |  |  |  |  |  |
| **10** |  |  |  |  |  |  |  |  |  |  |  |  |

***Разработка алгоритма***

**«Метод экспоненциального сглаживания»**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе**

**по дисциплине *«******Компьютерная обработка  
экспериментальных данных»***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа | Ф.И.О. | Подпись | Дата | Оценка |
| Студент | Доронин С.Г. |  |  |  |
| Консультант |  |  |  |  |
| Принял |  |  |  |  |

Уфа - 2013 г.

Федеральное агентство по образованию

Государственное образовательное учреждение высшего профессионального образования

«УФИМСКИЙ ГОСУДАРСТВЕННЫЙ АВИАЦИОННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной математики и кибернетики

ЗАДАНИЕ

На курсовую работу по дисциплине «Компьютерная обработка экспериментальных данных»

Студент \_\_Доронин С.Г. Группа \_ПРО-301в Консультант\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О номер акад. гр. Ф.И.О. .

1. Тема курсовой работы

Метод экспоненциального сглаживания

|  |
| --- |
| наименование темы |
|  |

|  |  |
| --- | --- |
| 1. Основное содержание: | Разбор работы алгоритма «Метода экспоненциального |
| сглаживания» на примере и проверка на программе, написанной для данной | |
| курсовой работы. | |

1. Требования к структуре и оформлению курсовой работы

3.1. Курсовая работа должна содержать следующие разделы:

Введение

Глава 1. Постановка задачи

Глава 2. Основные понятия теории метода экспоненциального сглаживания

Глава 3. Область применения алгоритма

Глава 4. Теория по алгоритму

Глава 5. Задача с решением

Глава 6. Написание алгоритма на Groovy

Глава 7. Обзор работы программы

Глава 8. Текст программы

Заключение

Список использованной литературы

1. Источники информации

Книги, учебники, учебные пособия, материалы сети Интернет.

На все источники, приведенные в списке литературы, должны быть ссылки по тексту курсовой работы.

Дата выдачи \_\_\_\_\_\_\_\_\_\_\_\_ Дата окончания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Абдрахманова Р.П.

Подпись

# Содержание

# Введение

Экстраполяция - это метод научного исследования, который основан на распространении прошлых и настоящих тенденций, закономерностей, связей на будущее развитие объекта прогнозирования. К методам экстраполяции относятся метод скользящей средней, метод экспоненциального сглаживания, метод наименьших квадратов.

Метод экспоненциального сглаживания наиболее эффективен при разработке среднесрочных прогнозов. Он приемлем при прогнозировании только на один период вперед. Его основные достоинства простота процедуры вычислений и возможность учета весов исходной информации.

# Постановка задачи

* Изучить теорию метода экспоненциального сглаживания;
* разобрать алгоритм для прогнозирования на основе метода экспоненциального сглаживания;
* написать программу на языке Groovy, которая бы при заданных пользователем параметрах, находила прогнозируемое значение.

# Основные понятия теории метода экспоненциального сглаживания

Метод экспоненциального сглаживания в отличие от метода скользящих средних еще и может быть использован для краткосрочных прогнозов будущей тенденции на один период вперед и автоматически корректирует любой прогноз в свете различий между фактическим и спрогнозированным результатом. Именно поэтому метод обладает явным преимуществом над ранее рассмотренным.

Название метода происходит из того факта, что при его применении получаются экспоненциально взвешенные скользящие средние по всему временному ряду. При экспоненциальном сглаживании учитываются все предшествующие наблюдения - предыдущее учитывается с максимальным весом, предшествующее ему - с несколько меньшим, самое ранее наблюдение влияет на результат с минимальным статистическим весом.

# Область применения алгоритма

Теория метода экспоненциального сглаживания находит свое применение во многих отраслях, где нужно сделать прогноз таких как:

* в экономике (прогнозирование бюджета);
* в бизнесе (прогноз продаж);
* в интернете (прогноз трафика);
* в домашнем хозяйстве (прогноз расходов)
* в банках (краткосрочное прогнозирование финансовых параметров деятельности);
* и т.д.

# Теория алгоритма

Рабочая формула метода экспоненциального сглаживания:

формула

где t – период, предшествующий прогнозному; t+1 – прогнозный период; Ut+1 - прогнозируемый показатель; α - параметр сглаживания; Уt - фактическое значение исследуемого показателя за период, предшествующий прогнозному; Ut - экспоненциально взвешенная средняя для периода, предшествующего прогнозному.

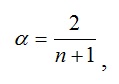
При прогнозировании данным методом возникает два затруднения:

* выбор значения параметра сглаживания α;
* определение начального значения Uo.

От величины α зависит, как быстро снижается вес влияния предшествующих наблюдений. Чем больше α, тем меньше сказывается влияние предшествующих лет. Если значение α близко к единице, то это приводит к учету при прогнозе в основном влияния лишь последних наблюдений. Если значение α близко к нулю, то веса, по которым взвешиваются уровни временного ряда, убывают медленно, т.е. при прогнозе учитываются все (или почти все) прошлые наблюдения.

Таким образом, если есть уверенность, что начальные условия, на основании которых разрабатывается прогноз, достоверны, следует использовать небольшую величину параметра сглаживания (α→0). Когда параметр сглаживания мал, то исследуемая функция ведет себя как средняя из большого числа прошлых уровней. Если нет достаточной уверенности в начальных условиях прогнозирования, то следует использовать большую величину α, что приведет к учету при прогнозе в основном влияния последних наблюдений.

Точного метода для выбора оптимальной величины параметра сглаживания α нет. В отдельных случаях автор данного метода профессор Браун предлагал определять величину α, исходя из длины интервала сглаживания. При этом α вычисляется по формуле:



Задача выбора Uo (экспоненциально взвешенного среднего начального) решается следующими способами:

* если есть данные о развитии явления в прошлом, то можно воспользоваться средней арифметической и приравнять к ней Uo;
* если таких сведений нет, то в качестве Uo используют исходное первое значение базы прогноза У1.

Также можно воспользоваться экспертными оценками.

Отметим, что при изучении экономических временных рядов и прогнозировании экономических процессов метод экспоненциального сглаживания не всегда «срабатывает». Это обусловлено тем, что экономические временные ряды бывают слишком короткими (15-20 наблюдений), и в случае, когда темпы роста и прироста велики, данный метод не «успевает» отразить все изменения.

# Задача с решением

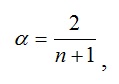
Имеются данные, характеризующие уровень безработицы в регионе, %:

Таблица 1. Уровень безработицы в регионе, %

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Январь | Февраль | Март | Апрель | Май | Июнь | Июль | Август | Сентябрь | Октябрь |
| 2,99 | 2,66 | 2,63 | 2,56 | 2,40 | 2,22 | 1,97 | 1,72 | 1,56 | 1,42 |

* Постройте прогноз уровня безработицы в регионе на ноябрь, декабрь, январь месяцы, используя метод экспоненциального сглаживания.
* Рассчитайте ошибки полученных прогнозов при использовании каждого метода.

1. Определяем значение параметра сглаживания по формуле:



где n – число наблюдений, входящих в интервал сглаживания. α = 2/ (10+1) = 0,2

1. Определяем начальное значение Uo двумя способами:

І способ (средняя арифметическая) Uo = (2,99 + 2,66 + 2,63 + 2,56 + 2,40 + 2,22 + 1,97 + 1,72 + 1,56 + 1,42)/10 = 22,13/10 = 2,21

II способ (принимаем первое значение базы прогноза) Uo = 2,99

1. Рассчитываем экспоненциально взвешенную среднюю для каждого периода, используя формулу:

формула

где t – период, предшествующий прогнозному; t+1 – прогнозный период; Ut+1 - прогнозируемый показатель; α - параметр сглаживания; Уt - фактическое значение исследуемого показателя за период, предшествующий прогнозному; Ut - экспоненциально взвешенная средняя для периода, предшествующего прогнозному.

Например:

Uфев = 2,99\*0,2 +(1-0,2) \* 2,21 = 2,37 (І способ)

Uмарт = 2,66\*0,2+(1-0,2) \* 2,37 = 2,43 (І способ) и т.д.

Uфев = 2,99\*0,2 +(1-0,2) \* 2,99 = 2,99 (II способ)

Uмарт = 2,66\*0,2+(1-0,2) \* 2,99 = 2,92 (II способ)

Uапр = 2,63\*0,2+(1-0,2) \* 2,92 = 2,86 (II способ) и т.д.

1. По этой же формуле вычисляем прогнозное значение

Uноябрь= 1,42\*0,2+(1-0,2) \* 2,08 = 1,95 (І способ)

Uноябрь= 1,42\*0,2+(1-0,2) \* 2,18 = 2,03 (ІІ способ)

Результаты заносим в таблицу.

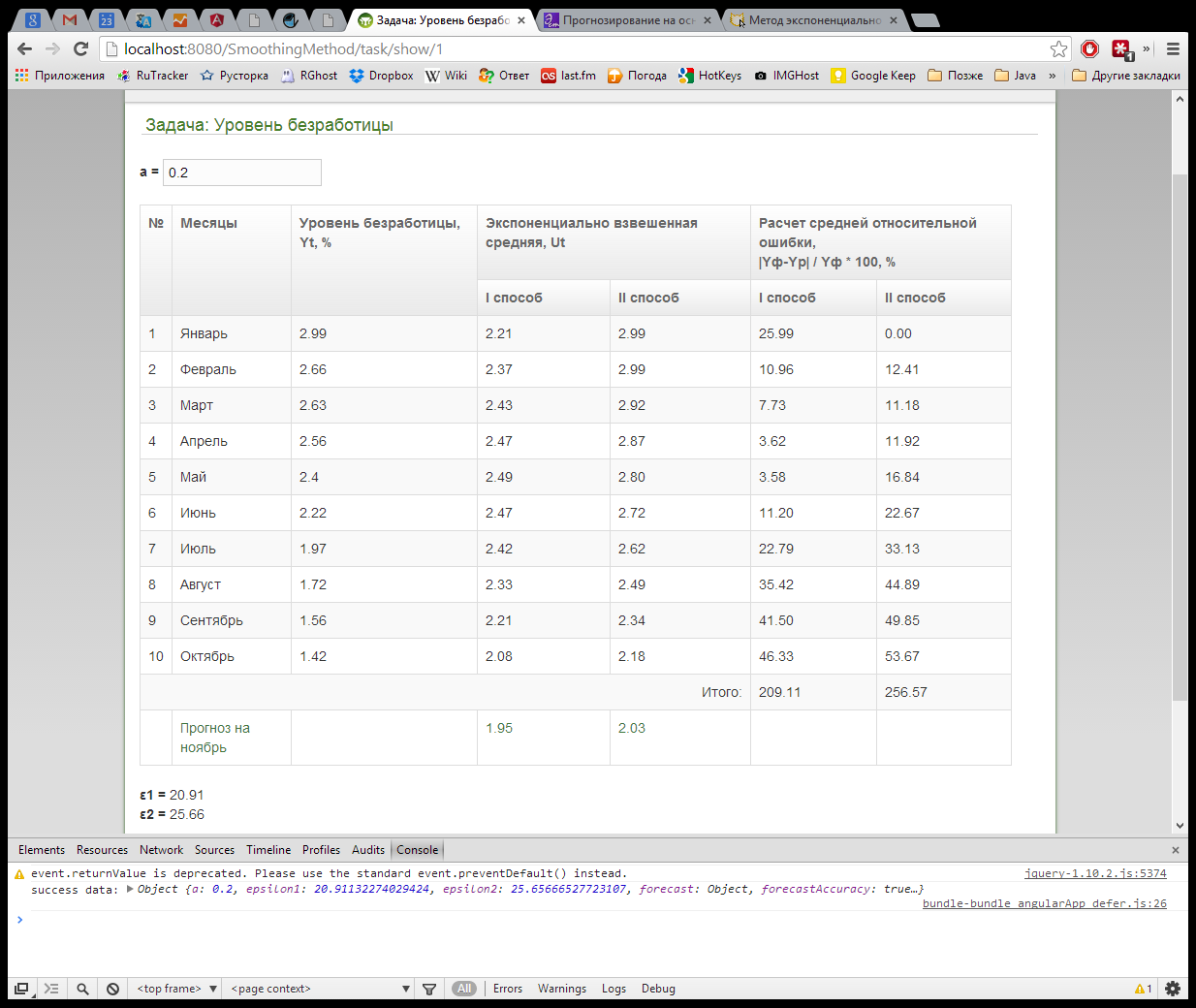
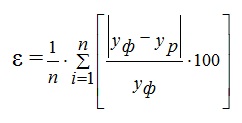


Рис. 1 - Результирующая таблица

1. Рассчитываем среднюю относительную ошибку по формуле:



ε = 209,58/10 = 20,91% (І способ)

ε = 255,63/10 = 25,66% (ІІ способ)

В каждом случае точность прогноза является удовлетворительной поскольку средняя относительная ошибка попадает в пределы 20-50%.

# Написание алгоритма на Groovy

Функция «calc()» находит начальное значение для U0 двумя способами с помощью функций «getU0Var1()» и «getU0Var2()».

Далее для каждого значения данных задачи вычисляет двумя способами экспоненциально взвешенную среднюю (Ut), с помощью функции «calcExpAvg(Double y, Double U)», так же вычисляет среднюю относительную ошибку, c помощью функции «calcAvgError(Double yFact, Double yCalc)».

Затем, собирает все данные в класс «SmoothingMethodDataWrapper» и передает в вызывающую функцию на вывод.

*/\*\**

*\* Метод экспоненциального сглаживания*

*\*/*

class SmoothingMethod {

*/\*\* Задача \*/*

private Task task

*/\*\* Данные задачи \*/*

private List<TaskData> taskDataList

*/\*\* Кол-во данных задачи \*/*

private int taskDataSize

*/\*\* Параметр сглаживания \*/*

private Double a

*/\*\* Конструктор метода \*/*

SmoothingMethod(Task task, Double a) {

this.task = task

taskDataList = task.data.sort{it.id}

taskDataSize = taskDataList.size()

this.a = (a != null) ? a : task.smoothingParameter

}

*/\*\* Расчет метода \*/*

SmoothingMethodDataWrapper calc() {

List<SmoothingMethodData> smDataList = new LinkedList<SmoothingMethodData>()

Double UVar1 = getU0Var1()

Double UVar2 = getU0Var2()

Double sumRelativeError1 = 0, sumRelativeError2 = 0

taskDataList << new TaskData(title: task.titleForecast, value: taskDataList.last().value)

taskDataList.eachWithIndex { taskData, i ->

SmoothingMethodData smData = new SmoothingMethodData(title: taskData.title, value: taskData.value)

smData.expAvg1 = UVar1

smData.expAvg2 = UVar2

smData.relativeError1 = calcAvgError(taskData.value, UVar1)

smData.relativeError2 = calcAvgError(taskData.value, UVar2)

if (i < taskDataSize) {

sumRelativeError1 += smData.relativeError1

sumRelativeError2 += smData.relativeError2

}

UVar1 = calcExpAvg(taskData.value, UVar1)

UVar2 = calcExpAvg(taskData.value, UVar2)

smDataList << smData

}

taskDataList.remove(taskDataSize)

SmoothingMethodDataWrapper dataWrapper = new SmoothingMethodDataWrapper(task: task, taskData: smDataList, forecast: smDataList.removeLast(), a: a,

sumRelativeError1: sumRelativeError1, sumRelativeError2: sumRelativeError2,

epsilon1: sumRelativeError1/taskDataSize, epsilon2: sumRelativeError2/taskDataSize)

dataWrapper.forecastAccuracy = checkForecastAccuracy(dataWrapper.epsilon1) && checkForecastAccuracy(dataWrapper.epsilon2)

return dataWrapper

}

*/\*\* Проверка точности прогноза \*/*

private static boolean checkForecastAccuracy(Double value) {

return value >= AppConst.FORECAST\_ACCURACY\_VALUE1 && value <= AppConst.FORECAST\_ACCURACY\_VALUE2

}

*/\*\* Начальное значение v1 (U0) \*/*

private Double getU0Var1() {

return taskDataList.sum{it.value} / taskDataSize

}

*/\*\* Начальное значение v2 (U0) \*/*

private Double getU0Var2() {

return taskDataList.getAt(0).value

}

*/\*\* Расчет экспоненциально взвешенной средней (Ut) \*/*

private Double calcExpAvg(Double y, Double U) {

return a \* y + (1 - a) \* U

}

*/\*\* Расчет средней относительной ошибки \*/*

private static Double calcAvgError(Double yFact, Double yCalc) {

Double val = Math.abs(yFact - yCalc) / Math.abs(yFact) \* 100

if (Double.isInfinite(val) || Double.isNaN(val)) {

throw new ArithmeticException(**"illegal double value:** $val**"**);

} else {

return val

}

}

}

# Обзор работы программы

1. Переходим на главную страницу программы и нажимаем кнопку «Создать новую задачу» (Рис. 2).

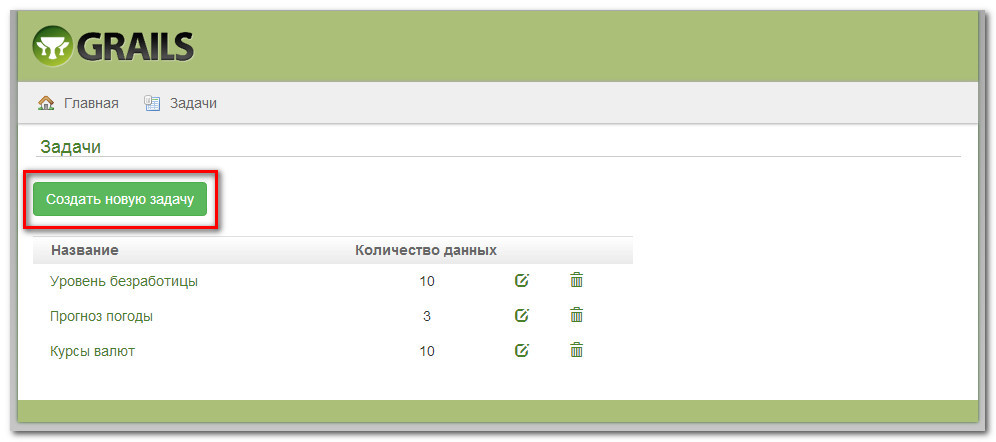


Рис. 2 – Создание задачи

1. Заполняем поля отмеченные красным, выбираем количество данных для задачи и заполняем их ниже, нажимаем кнопку «Создать» для сохранения задачи (Рис. 3).

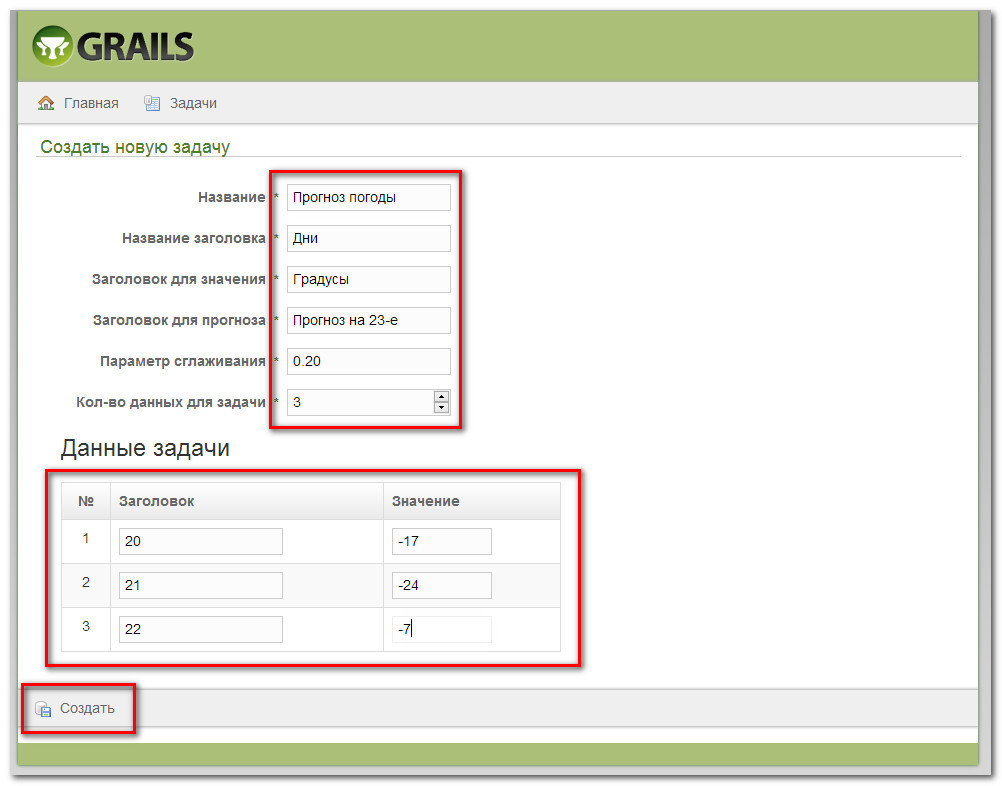


Рис. 3 – Заполнение задачи данными

1. После сохранения откроется окно с подсчетами с помощью метода экспоненциального сглаживания (Рис. 4). Прогнозируемое значение будет находиться в последней строке таблицы. По результатам средней относительной ошибки будет выдано сообщение, является ли точность прогноза удовлетворительной или нет.

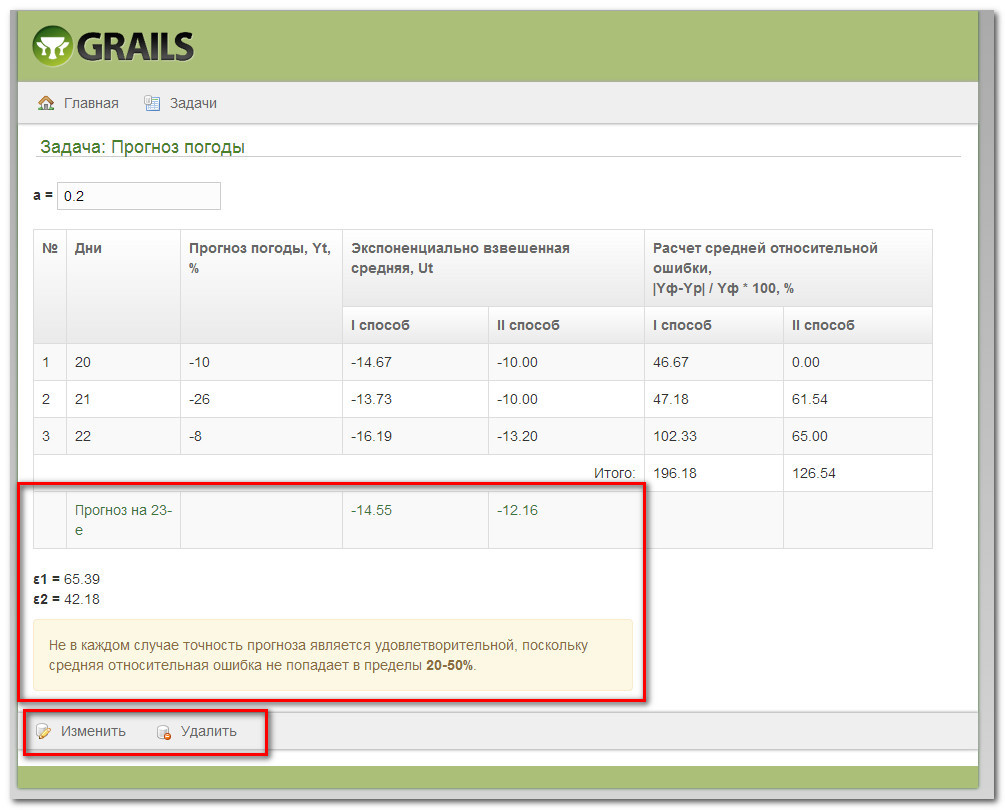


Рис. 4 – Прогнозирование с помощью метода экспоненциального сглаживания

1. Для редактирования задачи, нужно нажать на кнопку «Изменить» (откроется окно редактирования задачи), для удаления «Удалить» (откроется диалоговое окно с подтверждением, при положительном ответе задача будет удалена).

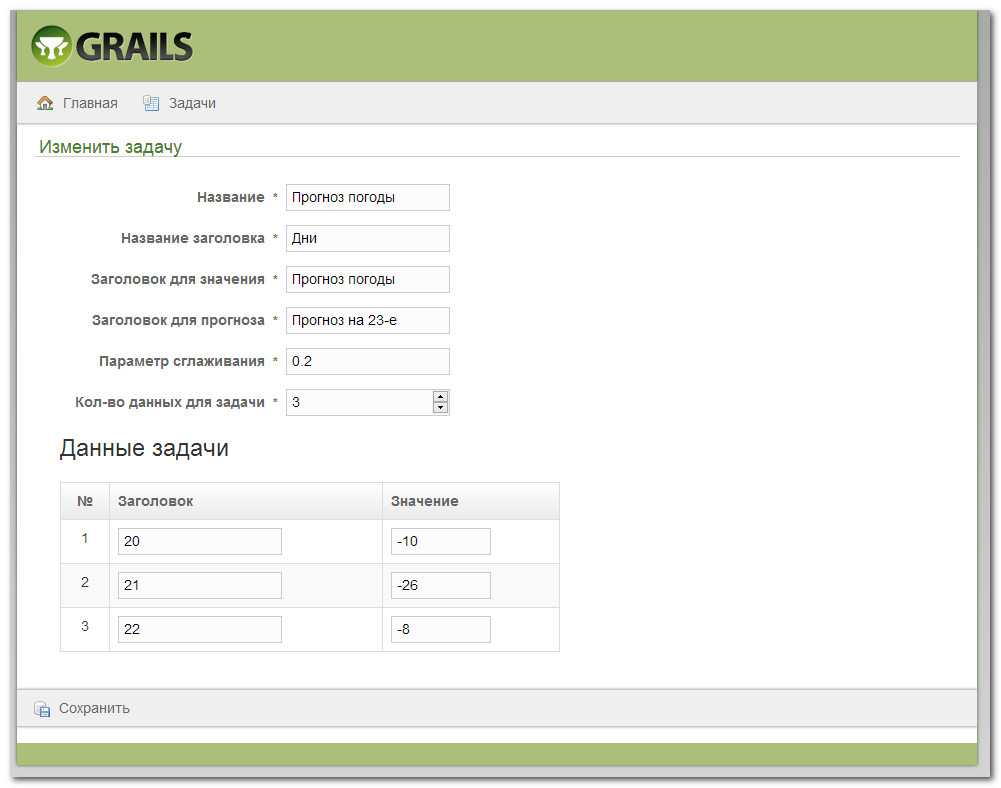


Рис. 5 – Редактирование задачи

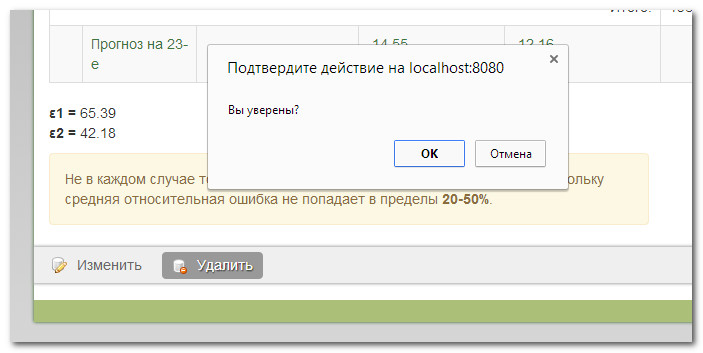


Рис. 6 – Удаление задачи

# Текст программы

## Класс управляющий поведением программы (TaskController)

package smoothingmethod.method

import grails.converters.JSON

import grails.transaction.Transactional

import static org.springframework.http.HttpStatus.\*

class TaskController {

static allowedMethods = [list: **'GET'**, save: **'POST'**, update: **'PUT'**]

def index() {

redirect action:**'list'**

}

def list() {

def taskList = Task.list(params)

respond taskList

}

def show(Task taskInstance, Double a) { *//TODO a парсится нормально с точкой, а объект из params, кажется нормально создается с запятой: new News(params)*

if (!taskInstance) {

flash.message = message(code: **'default.not.found.message'**, args: [message(code: **'task.label'**), params.id])

redirect action:**"list"**, method:**"GET"**

}

withFormat {

html taskInstance: taskInstance

json {

if (taskInstance.data.size() == 0) {

def TDE = [status: [name: **'TaskDataSizeException'**], message: message(code: **'taskData.size.error.message'**)]

render TDE as JSON

return

}

def dataWrapper

try {

dataWrapper = new SmoothingMethod(taskInstance, a).calc()

} catch(ArithmeticException e) {

def AE = [status: [name: **'ArithmeticException'**], message: message(code: **'exception.arithmetic.error.message'**)]

render AE as JSON

return

}

render dataWrapper as JSON

}

}

}

def create() {

respond new Task(params)

}

@Transactional

def save() {

def jsonObject = request.JSON;

Task taskInstance = new Task(jsonObject.task as Map)

if (!taskInstance.save(flush:true)) {

hasErrors()

return

}

jsonObject.taskData.each { Map td ->

TaskData taskDataInstance = new TaskData(title: td.title, value: td.value, task: taskInstance).save(flush:true)

if (!taskDataInstance) {

hasErrors()

return

}

}

render ([status: OK, id: taskInstance.id] as JSON)

}

def edit(Task taskInstance) {

withFormat {

form taskInstance: taskInstance

json {

def taskData = taskInstance?.data?.sort(){ it.id }

respond ([task: taskInstance, taskData: taskData])

}

}

}

@Transactional

def update() {

def jsonObject = request.JSON;

Task taskInstance = Task.get(jsonObject.task.id as Long)

*//Удаление старых и добавление новых TaskData*

taskInstance.data.removeAll(taskInstance.data)

List<TaskData> taskDataList = []

jsonObject.taskData.each { Map td ->

TaskData taskDataInstance = new TaskData(title: td.title, value: td.value, task: taskInstance).save(flush:true)

if (!taskDataInstance) {

hasErrors()

return

}

taskDataList << taskDataInstance

}

*//Обновление Task*

jsonObject.task.data = taskDataList

taskInstance.properties = jsonObject.task

if (!taskInstance.save(flush:true)) {

hasErrors()

return

}

render ([status: OK, id: taskInstance.id] as JSON)

}

@Transactional

def delete(Task taskInstance) {

if (taskInstance == null) {

notFound()

return

}

taskInstance.delete flush:true

request.withFormat {

form {

flash.message = message(code: **'default.deleted.message'**, args: [message(code: **'task.label'**), taskInstance.name])

redirect action:**"list"**, method:**"GET"**

}

**'\*'**{ render status: NO\_CONTENT }

}

}

protected void notFound() {

render ([status: NOT\_FOUND] as JSON)

}

protected void hasErrors() {

def obj = [status: INTERNAL\_SERVER\_ERROR, message: g.message(code: **'default.error.message'**)]

render obj as JSON

}

}

## Класс для работы с методом экспоненциального сглаживания (SmoothingMethod)

package smoothingmethod.method

import smoothingmethod.constants.AppConst

*/\*\**

*\** ***@author*** *Doronin Sergey*

*\* Date: 20.12.13*

*\* Time: 16:43*

*\*/*

*/\*\**

*\* Метод экспоненциального сглаживания*

*\*/*

class SmoothingMethod {

*/\*\* Задача \*/*

private Task task

*/\*\* Данные задачи \*/*

private List<TaskData> taskDataList

*/\*\* Кол-во данных задачи \*/*

private int taskDataSize

*/\*\* Параметр сглаживания \*/*

private Double a

*/\*\* Конструктор метода \*/*

SmoothingMethod(Task task, Double a) {

this.task = task

taskDataList = task.data.sort{it.id}

taskDataSize = taskDataList.size()

this.a = (a != null) ? a : task.smoothingParameter

}

*/\*\* Расчет метода \*/*

SmoothingMethodDataWrapper calc() {

println(**">>calc():** ${task.name}**"**)

List<SmoothingMethodData> smDataList = new LinkedList<SmoothingMethodData>()

Double UVar1 = getU0Var1()

Double UVar2 = getU0Var2()

Double sumRelativeError1 = 0, sumRelativeError2 = 0

taskDataList << new TaskData(title: task.titleForecast, value: taskDataList.last().value)

taskDataList.eachWithIndex { taskData, i ->

SmoothingMethodData smData = new SmoothingMethodData(title: taskData.title, value: taskData.value)

smData.expAvg1 = UVar1

smData.expAvg2 = UVar2

smData.relativeError1 = calcAvgError(taskData.value, UVar1)

smData.relativeError2 = calcAvgError(taskData.value, UVar2)

if (i < taskDataSize) {

sumRelativeError1 += smData.relativeError1

sumRelativeError2 += smData.relativeError2

}

UVar1 = calcExpAvg(taskData.value, UVar1)

UVar2 = calcExpAvg(taskData.value, UVar2)

smDataList << smData

}

taskDataList.remove(taskDataSize)

SmoothingMethodDataWrapper dataWrapper = new SmoothingMethodDataWrapper(task: task, taskData: smDataList, forecast: smDataList.removeLast(), a: a,

sumRelativeError1: sumRelativeError1, sumRelativeError2: sumRelativeError2,

epsilon1: sumRelativeError1/taskDataSize, epsilon2: sumRelativeError2/taskDataSize)

dataWrapper.forecastAccuracy = checkForecastAccuracy(dataWrapper.epsilon1) && checkForecastAccuracy(dataWrapper.epsilon2)

return dataWrapper

}

*/\*\* Проверка точности прогноза \*/*

private static boolean checkForecastAccuracy(Double value) {

return value >= AppConst.FORECAST\_ACCURACY\_VALUE1 && value <= AppConst.FORECAST\_ACCURACY\_VALUE2

}

*/\*\* Начальное значение v1 (U0) \*/*

private Double getU0Var1() {

return taskDataList.sum{it.value} / taskDataSize

}

*/\*\* Начальное значение v2 (U0) \*/*

private Double getU0Var2() {

return taskDataList.getAt(0).value

}

*/\*\* Расчет экспоненциально взвешенной средней (Ut) \*/*

private Double calcExpAvg(Double y, Double U) {

return a \* y + (1 - a) \* U

}

*/\*\* Расчет средней относительной ошибки \*/*

private static Double calcAvgError(Double yFact, Double yCalc) {

Double val = Math.abs(yFact - yCalc) / Math.abs(yFact) \* 100

if (Double.isInfinite(val) || Double.isNaN(val)) {

throw new ArithmeticException(**"illegal double value:** $val**"**);

} else {

return val

}

}

}

## Скрипты на стороне клиента (controllers.js)

**'use strict'**;

*/\* Controllers \*/*

**var** controllers = angular.module(**'controllers'**, []);

*/\*\* TaskShowController \*/*

controllers.controller(**'TaskShowController'**, [**'$scope'**, **'$http'**, **'$location'**, **function**($scope, $http, $location) {

*/\*\* Обновление содержимого страницы \*/*

$scope.updateView = **function**(a) {

$http.get($location.absUrl() + **'.json?a='** + a).success(**function**(data) {

console.log(**'success data:'**, data);

**if** (data.status) {

$scope.hasErrors = **true**;

$scope.errorMessage = data.message;

} **else** {

$scope.data = data;

}

});

};

$scope.updateView();

}]);

*/\*\* Изменение кол-ва данных задачи \*/*

**function** changeCountData(taskData, countData) {

countData = parseInt(countData);

**var** times = Math.abs(countData - taskData.length);

**var** push = countData > taskData.length;

**for** (**var** i=0; i < times; i++) {

**if** (push) {

taskData.push(**new** Data(**''**, **''**))

} **else** {

taskData.pop();

}

}

}

*/\*\* Конвертирует value в Float \*/*

**function** convertValue(task, taskData) {

task.smoothingParameter = parseFloat(task.smoothingParameter)

**for** (**var** i=0; i<taskData.length; i++) {

taskData[i].value = parseFloat(taskData[i].value);

}

**return** taskData

}

*/\*\* TaskCreateController \*/*

controllers.controller(**'TaskCreateController'**, [**'$scope'**, **'$http'**, **function**($scope, $http) {

$scope.task = {};

$scope.taskData = [];

*/\*\* Изменение кол-ва данных задачи \*/*

$scope.changeCountData = **function**() {

changeCountData($scope.taskData, $scope.countData)

};

*/\*\* Сохранение Задачи и ее данных \*/*

$scope.save = **function**(actionSave, actionShow) {

convertValue($scope.task, $scope.taskData);

**var** dataIn = {task: $scope.task, taskData: $scope.taskData};

$http.post(actionSave, dataIn).success(**function**(data) {

console.log(**'OK'**, data);

**if** (data.status.name == **'OK'**) {

document.location = actionShow + **'/'** + data.id;

} **else** {

$scope.hasErrors = **true**;

$scope.errorMessage = data.message

}

});

};

*//Начальная инициализация*

$scope.countData = 5;

$scope.changeCountData();

}]);

*/\*\* TaskEditController \*/*

controllers.controller(**'TaskEditController'**, [**'$scope'**, **'$http'**, **'$location'**, **function**($scope, $http, $location) {

*/\*\* Обновление содержимого страницы \*/*

$scope.updateView = **function**(a) {

$http.get($location.absUrl() + **'.json'**).success(**function**(data) {

console.log(**'success data:'**, data);

$scope.task = data.task;

$scope.countData = data.taskData.length;

$scope.taskData = data.taskData;

});

};

*/\*\* Изменение кол-ва данных задачи \*/*

$scope.changeCountData = **function**() {

changeCountData($scope.taskData, $scope.countData)

};

*/\*\* Изменение Задачи и ее данных \*/*

$scope.update = **function**(actionUpdate, actionShow) {

convertValue($scope.task, $scope.taskData);

**var** dataIn = {task: $scope.task, taskData: $scope.taskData};

$http.put(actionUpdate, dataIn).success(**function**(data) {

console.log(**'success data:'**, data);

**if** (data.status.name == **'OK'**) {

document.location = actionShow + **'/'** + data.id;

} **else** {

$scope.hasErrors = **true**;

$scope.errorMessage = data.message

}

});

};

$scope.updateView();

}]);

# TODO

Поискать в тексте Граф

Поискать название старой курсовой

Имя препода

Нумерацмя разделов

Содержание