Оглавление

[Введение 2](#_Toc497289037)

[Описание модуля run\_scenario\_array\_gmp 3](#_Toc497289038)

[Алгоритм работы модуля 3](#_Toc497289039)

[Технические подробности проведения расчетов (run\_scenario\_array\_gmp) 4](#_Toc497289040)

[Результаты проведения расчетов (run\_scenario\_array\_gmp) 5](#_Toc497289041)

[Описание модуля createVariationalSeriesFI 6](#_Toc497289042)

[Алгоритм работы модуля 6](#_Toc497289043)

[Технические подробности проведения расчётов (createVariationalSeriesFI) 7](#_Toc497289044)

[Результаты проведения расчетов(createVariationalSeriesFI) 8](#_Toc497289045)

[Описание модуля createVariationalSeriesGRD 9](#_Toc497289046)

[Алгоритм работы модуля 9](#_Toc497289047)

[Технические подробности проведения расчетов (createVariationalSeriesGRD) 10](#_Toc497289048)

[Результаты проведения расчетов (createVariationalSeriesGRD) 12](#_Toc497289049)

[Описание модуля createTrackAxis 13](#_Toc497289050)

[Алгоритм работы модуля 13](#_Toc497289051)

# Введение

В ходе работ по подготовке вероятностного анализа безопасности второго уровня проектируемых в Российской Федерации и зарубежом АЭС рассматриваются тяжелые запроектные аварии с плавлением топлива, сопровождающиеся выбросом значительного количества радиоактивных веществ в атмосферу. Одним из критериев приемлемости является не превышение суммарной вероятности большого аварийного выброса (выброс радиоактивных веществ в окружающую среду при аварии на АС, при котором необходимо выполнение мер защиты населения на границе зоны планирования защитных мероприятий на начальном периоде аварии, установленной в соответствии с требованиями норм и правил по размещению АС, и за ее пределами) для каждого блока АС на интервале в один год, равной 10-7. Для проведения таких оценок необходимо учитывать географическое положение объекта, неоднородную пространственную и временную структуру атмосферных течений, которая неразрывно связана с климатическими особенностями географического положения площадки размещения АЭС. Современным и научно обоснованным подходом для решения такого рода задач является использование данных метеорологического реанализа совместно с программными средствами, моделирующими перенос радиоактивных веществ в окружающей среде, осаждение их на подстилающую поверхность и позволяющих оценивать возможные дозы на население.

Для решения данной задачи использовались:

* Расчетный код СОКРАТ для моделирования поведения ПД вплоть до выхода РВ в окружающую среду
* модель НОСТРАДАМУС для моделирования распространения радиоактивности в окружающей среде и оценки доз облучения населения
* модель WRF-ARW для подготовки метеорологических полей за длительный промежуток времени.

Для проведения многовариантного анализа возможных последствий использовался набор вспомогательных инструментов, позволяющих проводить пакетные расчеты и их статистический анализ с использованием современных высокопроизводительных вычислительных систем.

# Описание модуля run\_scenario\_array\_gmp

## Алгоритм работы модуля

Модуль run\_scenario\_array\_gmp предназначен непосредственно для проведения многовариантных расчетов параметров радиационной обстановки с варьируемым временем начала выброса. Он ориентирован на запуск и управление расчетом на кластерной вычислительной установке ИБРАЭ РАН, которая включает в себя набор вычислительных узлов, каждый из которых включает 32 процессора. Поскольку на данный момент расчетный модуль НОСТРАДАМУС не распараллелен, на каждом процессе запускается копия расчетного модуля с измененным временем началом выброса. Идентификатором является условный номер процесса (от 1 до количества процессоров, задействованных в проведении расчетов). В такой постановке задачи синхронизация между процессами не требуется, изменяется только время начала выброса, которое для каждого расчета независимо. Т.о. имеется набор независимых расчетов, которые проводятся параллельно, используя выделенные вычислительные ресурсы.

Модуль run\_scenario\_array\_gmp разработан на языке Python версии 2.7 это интерпретируемый язык программирования, поэтому настройка для конкретного расчета проводится непосредственно внутри скрипта путем изменения ряда параметров (ниже будет представлено подробное описание основных настроечных параметров).

Задается начальное время расчета аварии, зависящее от идентификатора процесса.

Для расчета аварии в течение 366 дней с интервалом в 2 часа, запускается цикл по времени.

В цикле меняются входные данные (файл AAK\_VVER\_TOI\_S3.xml). Изменения происходят по следующей схеме: ко всем фазам выброса источника прибавляется время смещения аварии по времени(далее время смещения), затем для каждой целевой функции ко времени окончания расчета прибавляется время смещения и, для того что бы оставаться консервативными, время выброса источника.

Произведенные изменения фиксируются в файле in.xml. Затем файл in.xml предается на вход программе, которая на его основе проводит моделирование, результаты которого записываются в файл out.xml. Полученные результаты помещаются в папку в папку /results/time s/, где time-текущее время расчёта.

## Технические подробности проведения расчетов (run\_scenario\_array\_gmp)

Запуск модуля происходит в скрипте run\_py\_gmp

Передача идентификатора (idt) в параметрах командной строки при вызове модуля на исполнение.

Общий алгоритм проведения расчетов:

1. Задается начальное время смещения расчета (time):

 (1)

где t – время начала аварии (первой фазы выброса), idt – идентификатор процесса в случае использования модуля на кластерной вычислительной установке.

1. Запускается цикл расчёта по времени, время изменяется от начального времени до 366 дней**(**при необходимости возможно изменить время окончания цикла по времени, например, если доступны метеорологические данные за другой промежуток времени ) )
2. Открытие файла исходных данных \*.xml (в строке требуется указать корректное название и путь к файлу со сценарием аварии)
3. Из него формируется выходной файл in.xml. Для этого к относительному времени начала всех источников и всех фаз источника прибавляется время смещения аварии по временной шкале

где val – текущее время начала фаз источника (берется из файла исходных данных), time – текущее время начала аварии

1. К заданным в файле со сценарием временам окончания расчетов для всех целевых функционалов (timeEnd) прибавляется время смещения и суммарное время действия источника выброса. **При установке сценария необходимо установить корректные времена в блоке**:

где val – текущее значение времени окончания расчета функционала, time – текущее время начала аварии, time\_src – суммарное время действия источника выброса.

1. Отформатированный файл, записывается в in.xml в папку calculation\_x, где x-id процесса (при отсутствии создается, при наличии затирается старый).
2. Файл in.xml предается на вход программе, которая на его основе проводит моделирование, результаты которого записываются в файл out.xml.
3. Полученные результаты помещаются в папку в папку /results/time s/, где time-текущее время расчета.
4. После этого происходит изменение времени смещения. Интервал изменения времени равен 7200 секунд (2 часа) умноженный на общее количество процессов (64).

где – time текущее время начала аварии

## Результаты проведения расчетов (run\_scenario\_array\_gmp)

После работы модуля мы получаем набор директорий с результатами расчета (файлы out.xml), которые подготовлены для обработки сторонними ресурсами, например модулем createVariationSeriesFI.py

# Описание модуля createVariationalSeriesFI

## Алгоритм работы модуля

Модуль createVariationalSeriesFI предназначен для первичной обработки данных, предоставленных модулем run\_scenario\_array\_gmp с целью ускорения последующего анализа.

Модуль createVariationalSeriesFI разработан на языке Python версии 2.7 это интерпретируемый язык программирования, поэтому настройка для конкретного расчета производится непосредственно внутри скрипта путем изменения ряда параметров (ниже будет представлено подробное описание основных настроечных параметров).

Данными для обработки являются результаты работы модуля run\_scenario\_array\_gmp.

В процессе работы модуля формируются бинарные файлы (f\*.bin) для каждой целевой функции. Бинарные файлы нужны нам для того, чтобы обработка данных происходила быстрее. Те же данные, но в формате txt занимают более 1Gb (бинарные файлы занимают 180Mb)

Бинарные файлы имеют структуру:

Первые 8 чисел:

* сountx – количество точек сетки по широте
* county - количество точек сетки по долготе
* lonmin – долгота юго-западного угла сетки
* latmin – широта юго-западного угла сетки
* dlon – интервал изменения долготы
* dlat – интервал изменения широты
* lonmax – долгота северо-восточного угла сетки
* latmax – широта северо-восточного угла сетки

Далее последовательно записываются сетки со значениями функционалов для текущего времени начала аварии.

Всего 101\*101=10201 точек каждой сетки.

Всего 4394 сетки

При этом во всех точках сетки ищется максимальное значение функционалов и записывается в бинарный файл, как последняя сетка.

Так же создается файл headerFile.dat, который содержит названия всех целевых функций.

На выходе из модуля получаем папку /result/ с набором бинарных файлов, соответствующих каждой целевой функции.

## Технические подробности проведения расчётов (createVariationalSeriesFI)

Общий алгоритм проведения расчетов:

1. Путь к папке с результатами указывается в переменой patToFolderWithResults, к папке с входными данными - в переменной patToFIdir.
2. Поиск времени начала аварии, равного нулю. Это делается для того, чтобы получить не измененные времена окончания расчета аварии.
3. Считывание параметров текущей целевой функции (calcFunction, tipObl, isotop, phchForm, organ, age, timeEnd), помещение этих данных в массив headers.
4. Для текущего времени начала аварии, для данной целевой функции копируются данные сетки в массив self.data.
5. Создание файла f\*.bin. Запись в него параметров целевой функции.

где “<” - символ, указывающий на формат записи чисел(little-endian, от младшего к старшему), “%d” отвечает за количество ячеек для записи, “d” – формат данных(double 8 байт), grid – массив параметров:

Так заполняются первые 64 байта файла (8 значений по 8 байт)

1. Далее записываем сетку в файл:

где “<” символ, указывающий на формат записи чисел(little-endian, от младшего к старшему), “%d” отвечает за количество ячеек для записи, которое равно длине массива data (101x101), “f” – формат данных (float 4 байта)

1. Создается файл headerFile.dat в который записываются данные функционалов (массив headers).
2. После этого, в бинарные файлы дописываются только данные сеток (шаг 5 и шаг 7пропускается).
3. Поиск максимальных значений функционалов:

где positionInXml – номер функционала, gridsWithMaxValues – массив максимальных значений

1. Массив gridsWithMaxValues записывается последним в бинарный файл (аналогично пункту 6).

## Результаты проведения расчетов(createVariationalSeriesFI)

После работы модуля мы получаем набор бинарных файлов, соответствующих своим целевым функциям. Эти файлы подготовлены для обработки сторонними ресурсами, например модулем createVariationalSeriesNP или createVariationalSeriesGRD.

# Описание модуля createVariationalSeriesGRD

## Алгоритм работы модуля

Модуль createVariationalSeriesGRD предназначен для визуализации значений целевых функционалов с заданным пользователем доверительным интервалом.

Модуль createVariationalSeriesGRD разработан на языке Python версии 2.7 это интерпретируемый язык программирования, поэтому настройка для конкретного расчета проводится непосредственно внутри скрипта путем изменения ряда параметров (ниже будет представлено подробное описание основных настроечных параметров).

На вход модулю подается директория, содержащая бинарные файлы, созданные модулем createVariationalSeriesFI.

Выбирается первый файл. Считываются начальные параметры:

* сountx – количество точек сетки по широте
* county - количество точек сетки по долготе
* lonmin – долгота юго-западного угла сетки
* latmin – широта юго-западного угла сетки
* dlon – интервал изменения долготы
* dlat – интервал изменения широты
* lonmax – долгота северо-восточного угла сетки
* latmax – широта северо-восточного угла сетки

После этого, выбирается первая точка, затем данные всех сеток целевых функций для этой точки копируются в массив. Далее, из этого массива строится вариационный ряд и выбирается для него доверительный интервал. Полученное значение записывается в другой массив (data). Затем происходит переход к другой точке сетки. Когда массив data заполнен, он записывается в файл f\*.grd. Так же, создается файл f\*\_max.grd, в котором лежат максимальные значения целевых функций на сетке.

После завершения работы модуля получаем файлы формата .grd, готовый для открытия в программе qgis (или любой другой геоинформационной программе).

## Технические подробности проведения расчетов (createVariationalSeriesGRD)

Общий алгоритм проведения расчетов:

1. Ввод директорий входных(patToFIdir) и выходных (pathToSIdir) данных:
2. Список всех доступных в папке patToFIdir бинарных файлов, помещается в список “listOfFiles”.
3. Запускается цикл по всем файлам в списке (. В цикле:
   * 1. Считываем первые 64 байта(8 значений формата double), в них лежат начальные параметры

**б)** Распаковываем считанные данные в список datastr:

где “<” - символ, указывающий на формат записи чисел(little-endian, от младшего к старшему), “%d” отвечает за количество ячеек для записи,

“d” – формат данных(double 8 байт)

**в)** Эти данные передаются в конструктор класса GeoGrid

г) Формируем сдвиги для чтения бинарного файла:

Количество байтов, которое занимают начальные параметры:

Количество байтов до следующей сетки:

где countx - количество точек сетки по широте, county - количество точек сетки по долготе, “4” – количество байт на 1 число типа float

д) Считаем количество точек сетки:

где os.stat(fileName) - размер файла, offsetStart - сдвиг на начальные данные (минус нужен, чтобы сделать из номера точки, позицию)

е) Формируем доверительный интервал:

где ceil(float(gridsCount)\*0,95) – округление сверху количества точек, помноженных на доверительный интервал(0.95)

**Значение доверительного интервала можно менять в зависимости от требований задачи.**

ж) Далее запускаем цикл по всем точкам (, где count – количество точек сетки) и в нем обнуляем массив arr[]

з) Внутри цикла по точкам запускаем цикл по всем сеткам

Формируем шаг сдвига между сетками и сдвигаем на это значение:

где grid – номер сетки, pp – номер узла сетки

Считываем значение целевой функции в данной точке:

Заполняем массив данных одной точки

Создаем вариационный ряд:

Берем доверительный интервал для точки и записываем в конечный массив data

и) Заполняем массив максимальных значений:

к) Записываем полученные массивы и в файл:

## Результаты проведения расчетов (createVariationalSeriesGRD)

После работы модуля мы получаем набор .grd файлов, которые подготовлены для обработки сторонними ресурсами, например программой qgis.

# Описание модуля createTrackAxis

## Алгоритм работы модуля

Модуль createTrackAxis предназначен для обработки данных, предоставленных модулем createVariationalSeriesGRD.

Модуль createVariationalSeriesGRD разработан на языке Python версии 2.7 это интерпретируемый язык программирования, поэтому настройка для конкретного расчета проводится непосредственно внутри скрипта путем изменения ряда параметров (ниже будет представлено подробное описание основных настроечных параметров).

На основе сеток, построенных для целевых функций, модуль строит ось следа функционала. От координат источника строятся окружности с заданными радиусами. Затем, окружности разбиваются на отрезки по углам от 0 до 360 с нужным шагом в градусах. Шаг задается пользователем. В точках разбиения находятся координаты максимального значения функционала на этой окружности. Найденные координаты записываются в файл maxPoint\_f\*.dat , где \* - номер целевого функционала.

Для оптимизации скорости процесса расчета модуль находит файл с заранее насчитанными координатами разбиения окружностей для заданного шага разбиения, если такового файла нет, то он его создает и впоследствии обращается к нему.

## Технические подробности проведения расчетов (createTrackAxis)

Общий алгоритм проведения расчетов

1. Указание пути к входным данным (pathToSI), результатам расчета (pathToAxis) и файла с координатами разбиений окружностей (pathToDphi).

pathToSI="/home/egor/Programs/stat/SI/"

pathToAxis="/home/egor/Programs/stat/Axis/"

pathToDphi="/home/egor/Programs/stat/dphi\_%s.dat",

где вместо %s подставляется текущий шаг разбиения dphi

1. **Задается точность разбиения**, определяемая глобальной переменной dphi(например, если точность 0.5 градуса, то окружность разбивается на 720 частей по градусам с шагом 0.5)
2. В функции findOrCreate происходит задание количества разбиений окружности:

где phiMax – значение максимального угла (360) phiMin – минимального угла(0)

Далее происходит проверка пути pathToDphi на существование такого файла. Если такой файл существует (в указанном пути или в директории с модулем), то происходит считывание координат, если нет, то файл создается в директории рядом с модулем createTrackAxis.

1. Создание файла с координатами происходит по следующему алгоритму:
   1. Запускается цикл по всем радиусам окружности(for i, r in enumerate(axeRadius))
   2. В цикле по радиусам запускается цикл по всем углам разбиения
   3. Функция angleOf(dx, dy) находит направление на текущую точку в градусах.
   4. Находим широту и долготу точки:

где метод geod.Direct() вычисляет широту и долготу точки расположенной на расстоянии r1 от точки с координатами , , по азимуту ,

– искомая долгота – искомая широта.

Сохраняем эти значения соответственно в

и

где i – счетчик по радиусам, j – счетчик по разбиениям

* 1. Записываем значения axePoints[i][j][0] и axePoints[i][j][1] через пробел, по одной точке на строке, в файл dphi\_\*.dat, где \* - точнось разбиения по углам

1. Если файл найден, то оттуда считываются значения долготы и широты, и заполняется массив axePoints:

Долгота:

Широта:

1. Для каждого целевого функционала выполняются следующие действия:
   1. Создается объект класса GeoGrid
   2. Считываются начальные параметры функционала:

• сountx – количество точек сетки по широте

• county - количество точек сетки по долготе

• lonmin – долгота юго-западного угла сетки

• latmin – широта юго-западного угла сетки

• dlon – интервал изменения долготы

• dlat – интервал изменения широты

• lonmax – долгота северо-восточного угла сетки

• latmax – широта северо-восточного угла сетки

• fmin – минимальное значение данного функционала

• fmax – максимальное значение данного функционала

* 1. Заполняется массив data значениями функционалов на сетке.
  2. Вызывается метод getPlumeAxisPoints(). В нем для полученных точек axePoints производится интерполяция значений функционалов и нахождение максимального значения для этого радиуса.

getValue() – функция, обеспечивающая интерполяцию.

Максимальные значения записываются в массив axeMaxPoint.

* 1. Массив axeMaxPoint записывается в файл “maxPoint\_f\*.dat”, где \* - номер функционала.

## Результаты проведения расчетов (createTrackAxis)

По итогу, после выполнения модуля, на выходе получается набор файлов, предназначенных для открытия в геоинформационных программах(например qgis).