# Общие сведения

## Обозначение и наименование программы

Название программы – «polish»

## Программное обеспечение, необходимое для функционирования программы

* Интерпретатор языка программирования Python не ниже версии Python-3.10.0
* Библиотеки необходимые для работы ПО
  + os
  + pathlib
  + shutil
  + pandas
  + functools
  + operator
  + numpyencoder
  + sys
  + argparse
  + yaml
  + json
  + toml
  + subprocess

## Языки программирования, на которых написана программа

Высокоуровневый язык программирования общего назначения Python

# Функциональное назначение

## Назначение программа

Проведение в автоматическом режиме расчетов с помощью программных модулей компьютерного моделирования, автоматическая обработка результатов, накопление базы постановок задач для тестирования (валидации и верификации), хранение информации о изменениях в результатах моделирования в зависимости от версии ПМ.

Выдача подробной и обобщенной информации о результатах тестирования.

Индикация о возможных проблемах в ПМ КМ по результатам моделирования.

Система предназначена для внутреннего использования. Основные пользователи — разработчики кода и тестировщики.

## Цели создания

* Тестирование новых версий ПО
* Актуализация данных о работе тестируемых программ на различных типах задач
* Создание базы результатов для анализа дальнейшего улучшения и оптимизации тестируемого ПО

## Задачи:

* Разработать систему инициализации кейсов для каждого ПО, которое тестируется с помощью «polish»
* Разработать систему запуска задач на кластере, а также на системе Windows
* Разработать систему обработки результатов
* Разработать систему верификации данных
* Разработать систему хранения истории результатов по версиям ПО, тестируемого с помощью polish

## Требование к программе

* Программа должна позволять пользователю запускать как единичные тесты, так и совокупность из нескольких тестовых задач
* Программа должна автоматически запускать задачи, проводить пост обработку, и выдавать вердикт об успешном/провальном результате теста на конкретной задаче по заранее утверждённым критериям
* В результате тестирования нужно видеть, как изменился результат относительно прошлой версии программы и как он отличается от эксперимента или иной выбранной референтной величины.
* Система должна выдавать короткий простой отчет о результате тестирования
* Для каждой задачи должен быть сформирован подробный набор данных/отчет для анализа
* Результаты тестирования должны сохранятся и помечаться версией программы, пользователем, который ее запустил, временем запуска
* Для каждой задачи должна формироваться статистика запусков, включающая в себя версию программы, время выполнения, успешность выполнения отклонения от референсных величин.
* Выполнение тестовых расчетов должно по возможности минимальным образом мешать выполнению основных расчетов на кластере. Лучше если тестирование будет занимать кластер при его простое.
* Для каждой задачи должна быть оценка временных затрат на ее расчет
* Система должна обеспечивать наиболее простой и быстрый способ проведения тестирования, добавления тестов, просмотра результатов тестирования
* Хранение постановок задач и результатов должно быть простым в использовании
* Хранение постановок задач должно быть по возможности компактным для наборов постановок с одними данными, но разными режимами

# Описание логической структуры

## Алгоритм программы

## Используемые методы

## Структура программы с описанием функций составных частей и связи между ними

## Связи программы с другими программами

# Используемые технические средства

# Создание кейса для программы

## Описание списка файлов и данных, требуемых для создания кейса

Для создания кейса необходимы следующие данные:

* Сетка (формат .PFG/.IBC и .GGD)
* Lazurit\_data для запуска без импорта сетки
* Скрипты и макросы обработки
* Заполненные конфиги, который послужит шаблоном для всех задач
* Таблица параметров задач (tasks.xlsx)

## Описание процесса создания кейса

Для создания кейса нужно разместить описанные выше файлы по соответствующим директориям в папке с названием нового кейса. Схематичное изображение директории изображено на Рисунке 1.

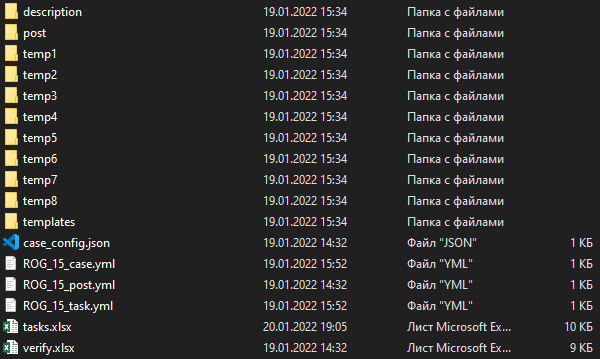


Рисунок 1 — схема директории кейса

1. Таблица с параметрами задач выглядит как показано на Рисунке 2, и заполняется следующим образом: Первый столбец таблицы «tasks» должен содержать названия параметров задачи и их значений разделённых нижним подчёркиванием, это используется для именований директорий для расчёта. Начиная со второго столбца их имена формируются из разделов конфига, которые требуют замены, разделённых точкой. *Значения столбцов могут иметь следующие типы значений: число, строка, список.*



Рисунок 2 — Таблица с параметрами задач кейса

1. Скрипты и макросы для обработки задач должны соответствовать нескольким требованиям:
   1. Если файл обработки — это скрипт (.py), то он должен содержать итоговую функцию, которая на вход принимает путь до последней итерации расчёта и путь до директории, куда будут сохранены результаты обработки
   2. Если файл обработки — это макрос (.mcr), то путь до последней итерации расчёта должен заменяться по ключу «path\_to\_last\_iteration», а путь до директории сохранения результатов — по фразе «path\_to\_report».

*Пример:*

*«D:\task\tecplot\T-10000\TEC\_FLOW\_T-100000\_B-164.plt» в тексте заменяется на*

*«input\_files»,*

*а путь для сохранение поля скоростей*

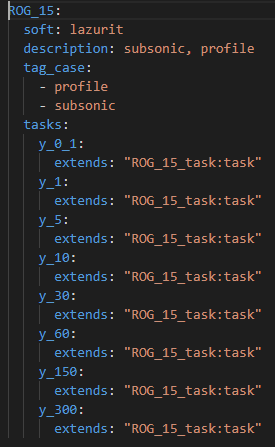
*«D:\post\_task \field\_U.png»*

*заменится на*

*«path\_to\_report /field\_U.png»*

## Конфиги кейсов и описание

## Конфиг кейса



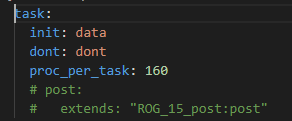
Soft – ПО под которое создаётся кейс

Description — категория задач к которой причисляется кейс

Tag\_case – теги кейса

Tasks – задачи кейса (там идут задачи кейса с импортом стандартных настроек для задачи для него, их можно поменять в этом же файле, переназначением поля из конфига стандартного для задачи)

## Конфиг задачи

\

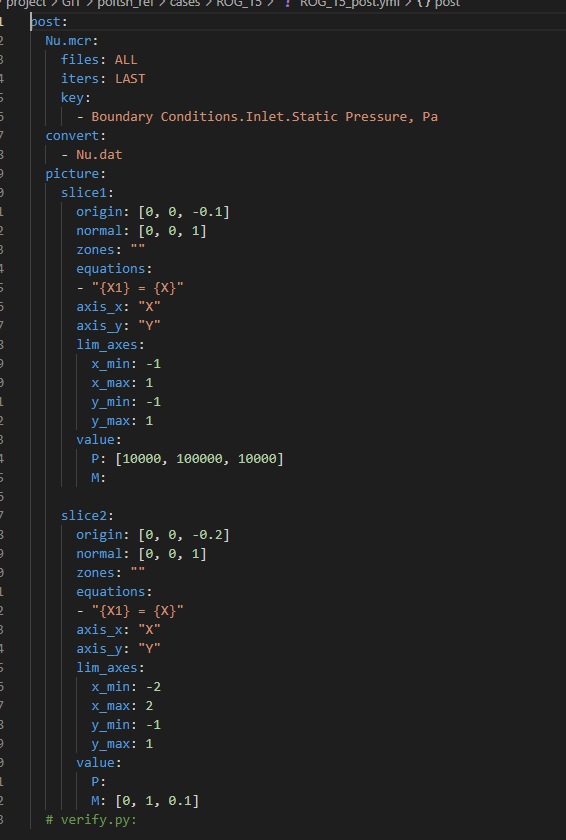
Init – способ инициализации (data – с заранее заготовленной даты (Lazurit\_data))

Dont — способ запуска задачи, в продолжение (cont), или с начала (dont)

Proc\_per\_task — количество процессов на которое можно запустить задачу (параметр для кластера)

Post — импорт настроек для постобработки задачи, так же можно некоторые поля заменить индивидуальными в шаблоне кейса

## Конфиг пособработки



Пока в разработке…

# Входные данные

Для того что бы запустить программу нужно составить конфиг и заполнить файл запуска программы. Файл конфига выглядит следующим образом:

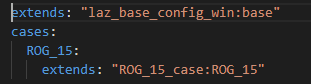


Рисунок — config.yml

Поле extends сообщает утилите по формированию конфига, что мы импортируем поле base из файла laz\_base\_config\_win. Можно создавать свои пресеты для автотеста, например, для запуска на различных устройствах.

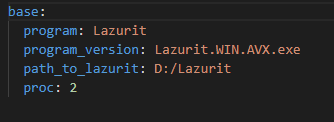


Рисунок 2 — laz\_base\_config\_win

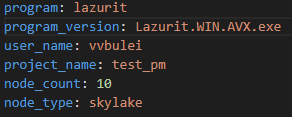


Рисунок 3 — laz\_base\_config\_cluster

* Program — название тестируемой программы
* Program\_version — версия тестируемой программы
* Path\_to\_Lazurit — путь до папки с необходимыми библиотеками для запуска Lazurit
* Proc — количество потоков для распараллеливания
* User\_name — имя пользователя для заполнения passport на кластере
* Project\_name — Название проекта в рамках которого запускается автотест, заполняет passport
* Node\_count — количество узлов выделенных под тестирование
* Node\_type — тип узлов для запуска тестирования

## Характер, организация и предварительная подготовка входных данных

Для запуска программы «polish» необходимы 2 файла:

* Для Windows
  + Config
  + Start\_autotest.bat — необходимо указать только путь до программы «polish»

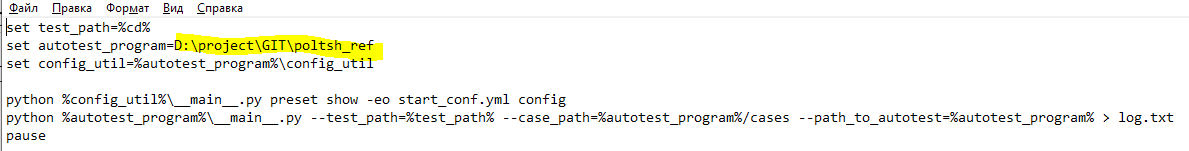


Рисунок — start\_autotest.bat

* Для Linux
  + Config
  + Start\_autotest.sh

## Формат, описание и способ кодирования входных данных

# Выходные данные

## Характер и организация выходных данных

## Формат, описание и способ кодирования выходных данных