Московский Государственный Университет

имени М.В.Ломоносова

Механико-математический факультет

Кафедра вычислительной механики

Курсовая работа

студента 424 группы

Шошиашвили Александра

«Оценка геомеханических напряжений в окрестности горных выработок»

научный руководитель:

Вершинин Анатолий Викторович

Москва 2017

**Содержание**

Введение 3

Про Micromime 3

Модель штреков 4

Модель керна 7

Алгоритм работы скрипта 10

Заключение 13

Список литературы 13

Приложение 14

**Введение**

Горным инженерам важно уметь определять зоны высоких геомеханических напряжений, чтобы выявлять опасные места в туннелях и штреках, и моделировать установку крепей.

В качестве решения было предложено воспользоваться CAE Fidesys [1] для расчета напряжений в штреках. Чтобы это осуществить, необходимо было модель, предоставленную горными инженерами, в формате горно-геологического программного комплекса для 3D моделирования горных выработок Micromime преобразовать в модель, которую можно открыть в препроцессоре Fidesys, а затем произвести расчет.

**Про Micromime**

Micromine [5] является комплексным решением для 3D-моделирования месторождений, предлагающим средства проектирования, оптимизации и планирования горных работ.

Это один из самых популярных программных комплексов для моделирования горных выработок наряду с Surpac, Whittle, NPV Scheduler.

С помощью Micromime горные инженеры создали 3D модель штреков, используя программные модули программы. Модель представляет из себя набор элементов с заданными размерами и свойствами материала в каждом элементе. Чтобы можно было использовать полученную модель в программном комплексе CAE Fidesys, эту модель экспортируют в специальном dat-формате. В этом файле содержатся данные о координатах элементов и их свойствами материалов (модуль Юнга, коэффициент Пуассона, плотность).

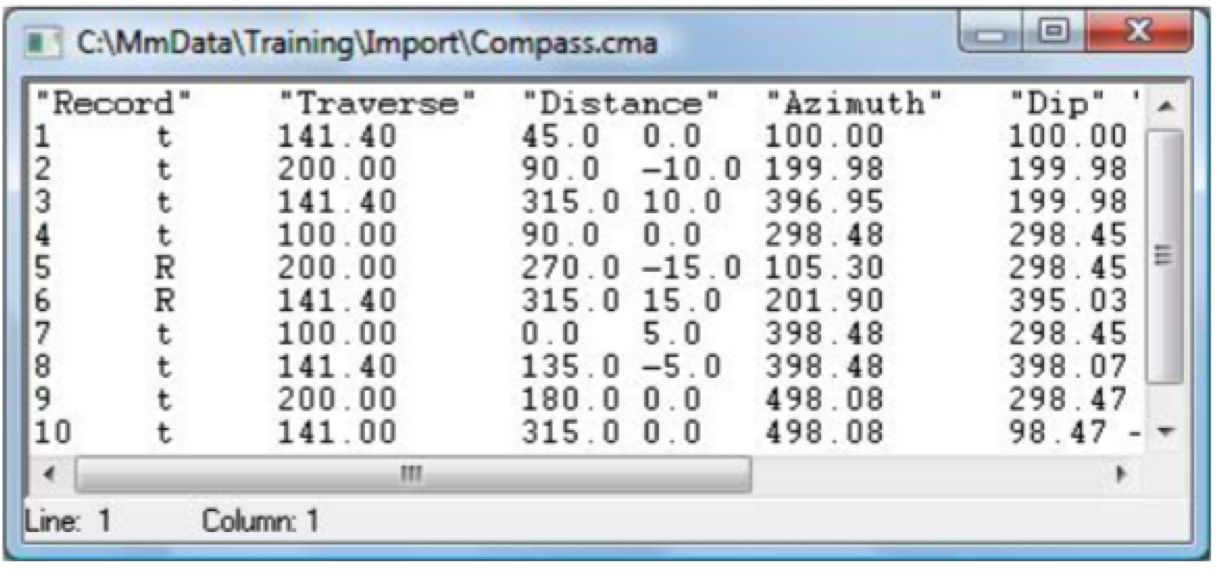


Рис. 1. Пример выходных данных при экспорте из Micromime

**Модель штреков**

Дано тело из упругого материала с полостями (Рис. 2). Примем следующие данные для решения задачи в условиях плоского деформированного состояния:

— модуль Юнга E = 7.7 ГПа,

— коэффициент Пуассона σ = 0.17,

— плотность ρ = 2200 кг/м³.

Размер элемента в программе составляет 6х6х1.5, чему соответсвует размер 6 м х 6 м х 1.5 м реальной модели. Для расчёта берется модель штреков с размерами 588 м х 594 м х 250.5 м.

Тело закреплено: на нижней грани все перемещения равны 0. На верхнюю грань к телу приложена сила тяжести по направлению нормали.

Решалась статическая задача нагружения под воздействием массовых сил. Таким образом, приходим к уравнению равновесия нашей задачи [4]:

∇jσij = 0

— уравнения равновесия;

σij = λ I1(ε) I + 2Gεij

— определяющие соотношения (закон Гука);

εij = ½(∇iuj + ∇jui)

— тензор деформаций (относительные смещения предполагаются малыми);

n σ = n P

— граничные условия на границе Г области D;

G = E/2(1 + μ)

λ = Eμ/((1 + μ)(1 – 2μ))

— связь коэффициентов Ламе с коэффициентом Пуассона и модулем Юнга;

σ — тензор напряжений, ε — тензор деформаций, λ , G — коэффициенты Ламе, μ — коэффициент Пуассона, E — модуль Юнга, I1(ε) — первый инвариант тензора деформаций, I - единичный тензор третьего ранга, P — давление на границе, n — нормаль к границе.

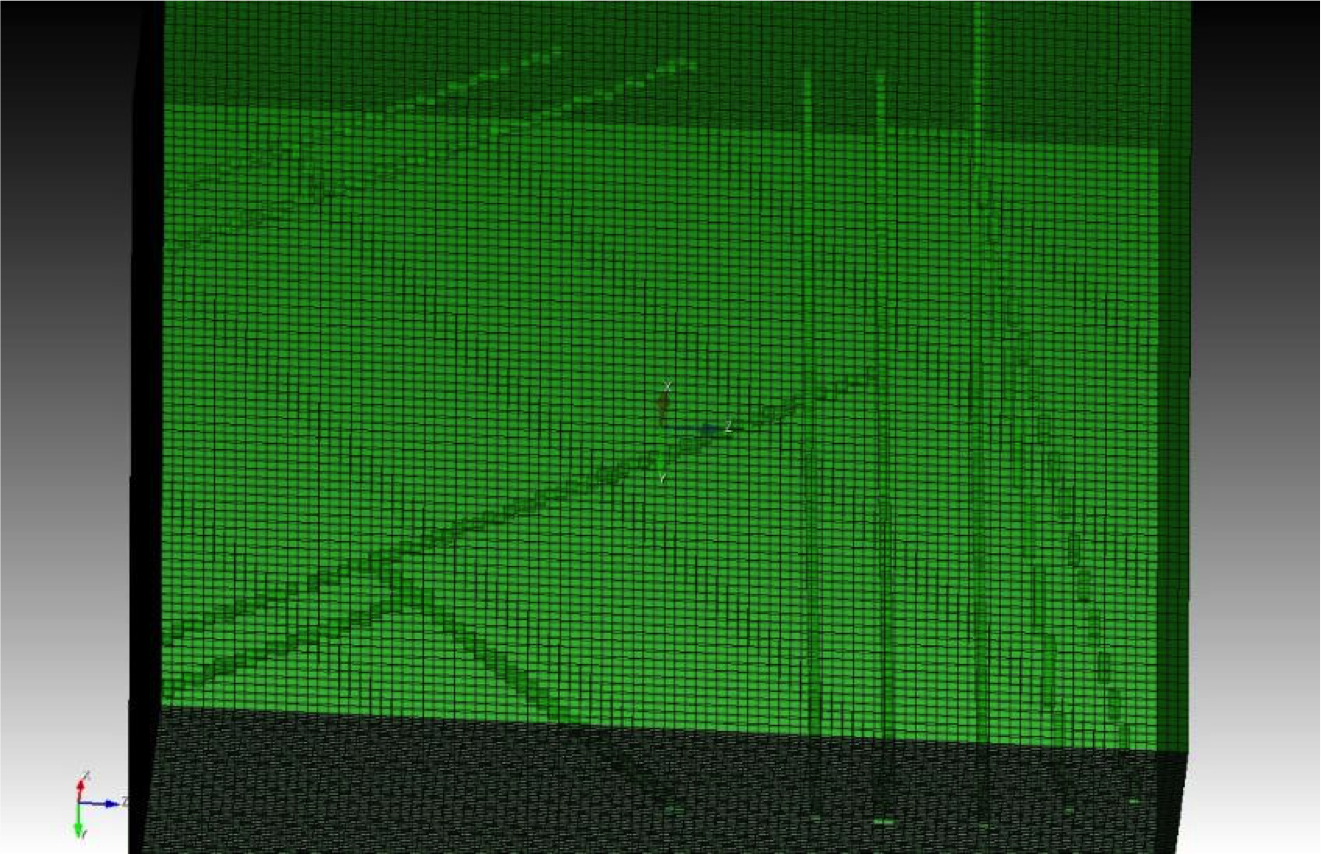


Рис. 2. Построенная модель с сеткой

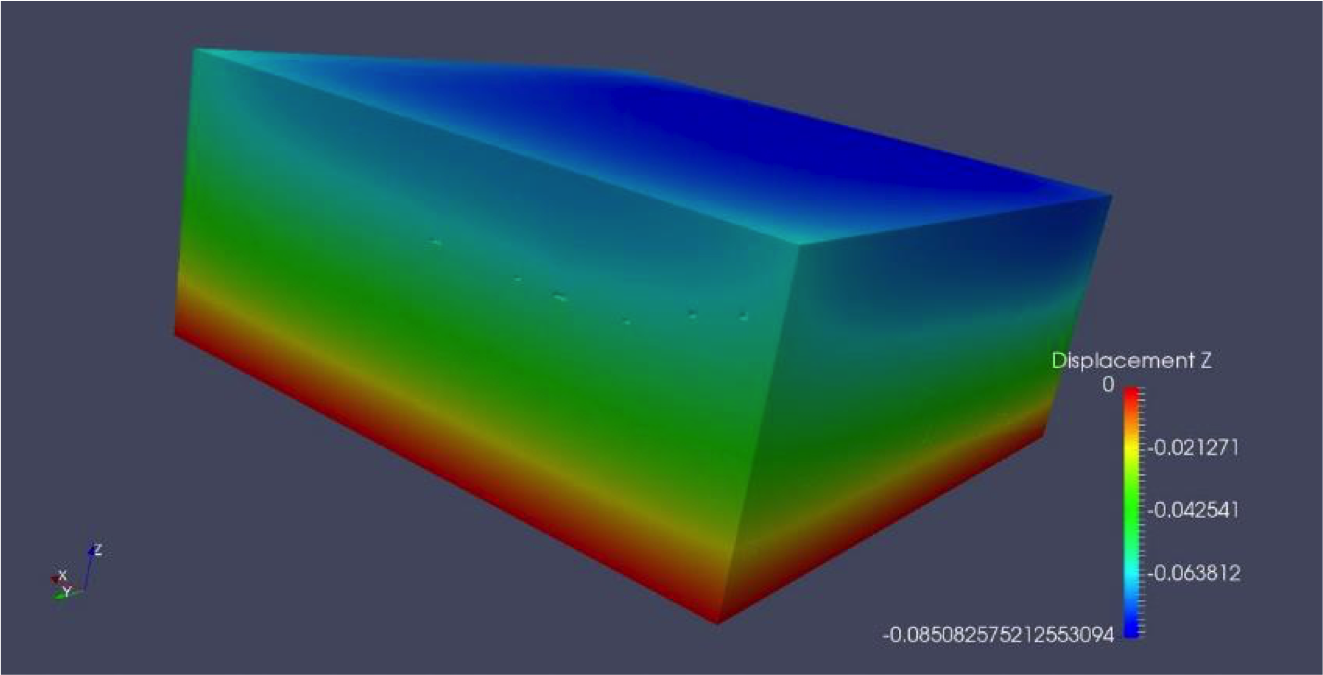


Рис. 3. Результат расчета модели с рисунка 2 на статическое нагружение

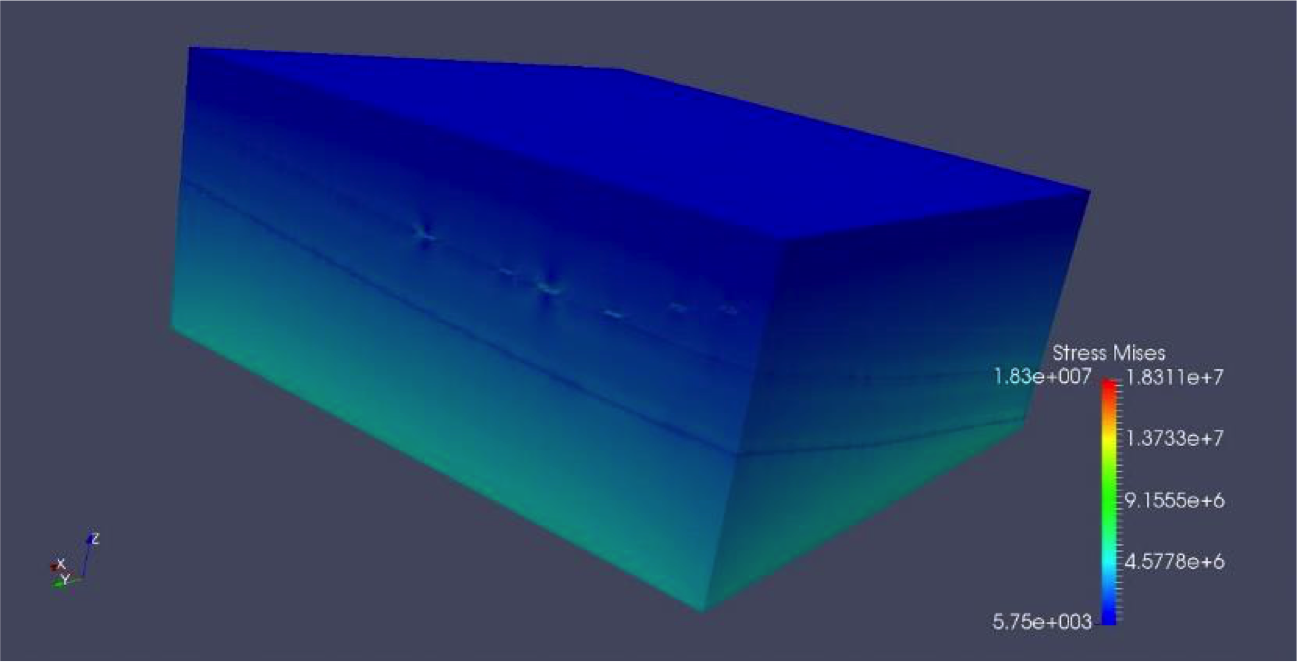


Рис. 4. Результат расчета модели с рисунка 2 на статическое нагружение

Табл. 1. Сравнение времен выполнения программы при использовании алгоритмов: на «чистом» Python и с использованием библиотек multiprocessing, threading, — для модели с сеткой, указанной на Рис.2. (для 200000 элементов)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Pure Python | multiprocessing + threading | Разница, с. | Разница, % |
| Общее | 550.5704 | 447.439 | 103.1314 | 18.7317 |
| Считывание файла и | 1.6572 | 0.3852 | 1.272 | 76.7559 |
| Приведение Point к началу координат | 0.3125 | 0.1591 | 0.1534 | 49.088 |
| Создание элементов из Point | 0.3337 | 0.2949 | 0.0388 | 11.6272 |
| Создание массива delete | 548.2670 | 446.5998 | 101.6672 | 18.5433 |

Диаграма 1. Сравнение времен выполнение программы из табл. 1

**Модель керна**

Дано тело из упругого материала с полостями (Рис. 5).

Примем следующие данные для решения задачи в условиях плоского деформированного состояния:

— модуль Юнга E = 70 ГПа,

— коэффициент Пуассона σ = 0.15,

Размер вокселя 1х1х1 — 10 мкм х 10 мкм х 10 мкм. Для расчёта берется воксельная модель с размерами 50х50х50.

Тело закреплено: на нижней грани все перемещения равны 0. На верхнюю грань к телу приложено давление P по направлению нормали, равное

P = 100000 Па

Боковые грани и поры свободны от нагрузок. Массовые силы отсутствуют.

Необходимо было увеличить скорость создания модели в препроцессоре CAE Fidesys. Для этого было решено распараллелить программу, используя библиотеку Threading для Python.

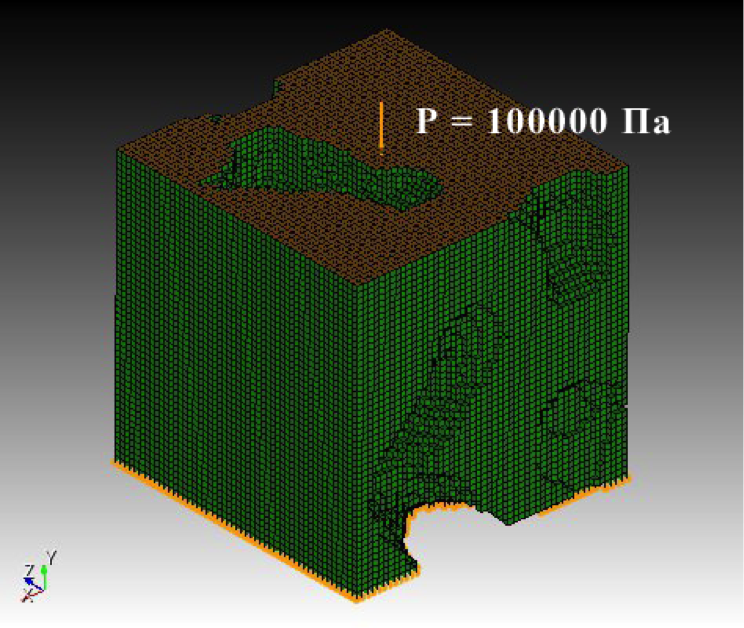


Рис. 5. Построенная модель керна с сеткой и приложенными силами

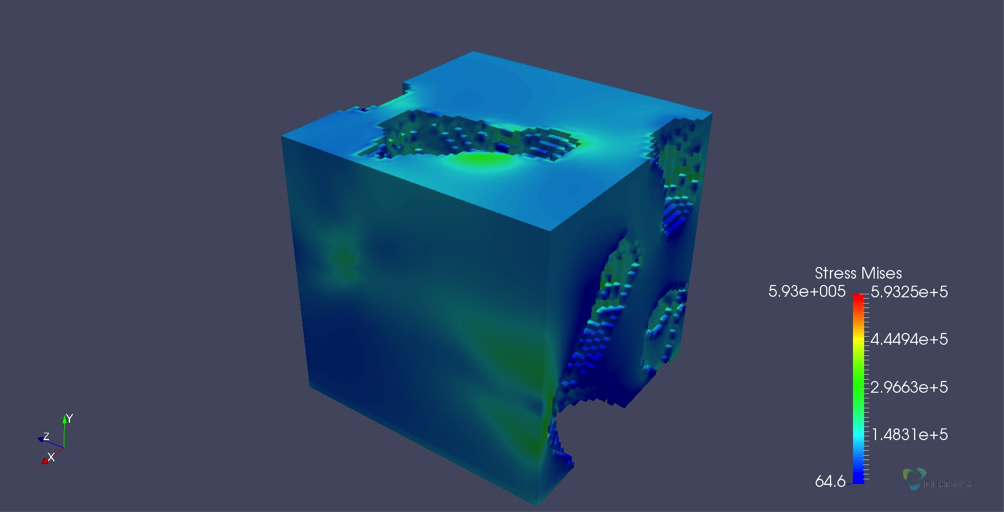


Рис. 6. Результат расчета модели с рисунка 5 на статическое нагружение

Табл. 2. Сравнение времен выполнения программы при использовании алгоритмов: на «чистом» Python и с использованием библиотек multiprocessing, threading, — для модели с сеткой, указанной на Рис.5. (для разного числа элементов)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Число элементов | Pure Python, с. | multiprocessing + threading, с. | Разница, с. | Разница, % |
| 1003 | 1 | 0.7 | 0.3 | 30 |
| 1503 | 4 | 3 | 1 | 25 |
| 2003 | 10 | 6 | 4 | 33.3 |
| 2503 | 17 | 12 | 5 | 29.4 |
| 3003 | 31 | 21 | 10 | 32.2 |
| 3303 | 44 | 30 | 14 | 31.8 |
| 4003 | 78 | 54 | 24 | 30.7 |
| 4503 | 112 | 76 | 36 | 32.1 |
| 5003 | 148 | 108 | 30 | 27.02 |
| 7503 | 514 | 371 | 143 | 27.8 |

Диаграма 2. Сравнение времен выполнение программы из табл. 2

**Алгоритм работы скрипта**

Сначала, строится выбранный тип модели по заданным параметрам, которые включают в себя масштаб, граничные условия и тип материала. Далее строится сетка, задаются свойства. После чего, производится рассчет [2], [3]. Примеры результатов рассчета показаны на рисунках 3, 4, 6.

Программная часть реализована на языке Python с помощью библиотек:

— Cubit для работы с препроцессором и постпроцессором Fidesys (создания модели, задания свойств, проведения расчетов),

— Threading для работы с многопоточностью

— Time для замера времени работы программы.

Обе программы из Приложения имеют расширение JOU, которое поддерживается программой CAE Fidesys.

Ниже приведены блок-схемы алгоритмов по созданию модели штреков и модели керна.

1. Модель штреков

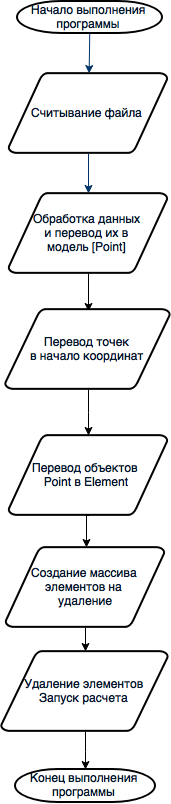


Рис. 7. Блок-схема работы программы по созданию модели штреков

1. Модель керна

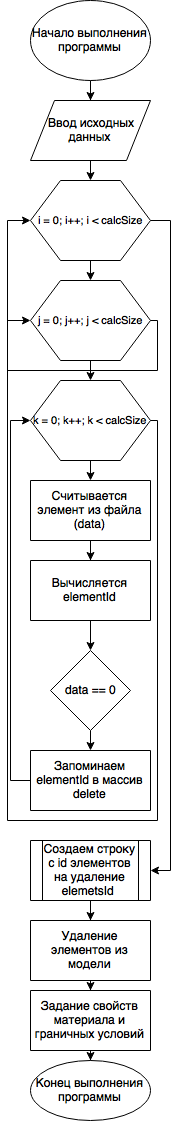


Рис. 8. Блок-схема работы программы по созданию модели керна

**Заключение**

Был разработан алгоритм на языке программирования Python, который реализует поставленные задачи, а именно: создание модели штреков из специальных данных из программного комплекса Micromime в предпроцессоре CAE Fidedys, а также произведение расчетов на полученной модели.

Тестирование алгоритма производилось на двух моделях (штреков и керна). На них также была посчитана статическая задача нагружения под воздействием массовых сил.

Так как создание больших моделей — ресурсоемкий процесс, то была произведена оптимизация алгоритма и его реализации по скорости для больших моделей. Сравнение скорости работы алгоритмов приведены в таблицах 1 и 2.

Таким образом, для определенных входных данных, процесс построения расчетной модели в CAE Fidesys был автоматизирован и оптимизирован.

**Список литературы**

1. cae-fidesys.ru
2. Прочностной анализ: Фидесис в руках инженера. Е. М. Морозов, В. А. Левин, А. В. Вершинин. (2015)
3. Модели и методы. Образование и развитие дефектов. В. А. Левин. (2015)
4. Седов Л.И. Механика сплошной среды. Т. 1, 2.  — М. Наука, 1994
5. micromime.com

**Приложение**

1. **Модель штреков**

# подключение внешних библиотек

from operator import attrgetter

import threading

import time

from multiprocessing import Pool, Process, Manager

exitFlag = 0

xSideSize = 6.0

ySideSize = 6.0

zSideSize = 1.5

points = []

file = "ju\_psn\_E\_ro.dat"

elementsId = []

delete = []

# класс точки

class Point:

x = 0.0

y = 0.0

z = 0.0

poisson = 0.0

e = 0.0

g = 0.0

def \_\_init\_\_(self, x, y, z, poisson, e, g):

self.x = x

self.y = y

self.z = z

self.poisson = poisson

self.e = e

self.g = g

# вспомогательная функция для распечатывания точки

def printPoint(self):

print(str(self.x) + ", " + str(self.y) + ", " + str(self.z) + ", " + str(self.poisson) + ", " + str(

self.e) + ", " + str(self.g))

class myThreadFirst (threading.Thread):

def \_\_init\_\_(self, threadID, name, i, reverseElements):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.name = name

self.i = i

self.reverseElements = reverseElements

def run(self):

loop\_step\_first(self.name, self.i, self.reverseElements)

class myThreadLine(threading.Thread):

def \_\_init\_\_(self, threadID, name, text\_file):

threading.Thread.\_\_init\_\_(self)

self.name = name

self.threadID = threadID

self.text\_file= text\_file

def run(self):

loop\_step(self.name, self.text\_file)

def loop\_step\_first(threadName, i, reverseElements):

reverse\_elements\_id\_helper(i, reverseElements)

if exitFlag:

threadName.exit()

def loop\_step(threadName, text\_file):

lines = []

lines.append(text\_file.readlines(100000))

if exitFlag:

threadName.exit()

def process(line):

pointsString = line.split()

point = Point(

x=float(pointsString[0]),

y=float(pointsString[1]),

z=float(pointsString[2]),

poisson=float(pointsString[3]),

e=float(pointsString[4]),

g=float(pointsString[5])

)

points.append(point)

def minElement(sequence):

low = sequence[0]

for i in sequence:

if i < low:

low = i

return low

def leadPointsToOrigin():

minX = min(points, key=attrgetter('x')).x

minY = min(points, key=attrgetter('y')).y

minZ = min(points, key=attrgetter('z')).z

for p in points:

p.x -= minX

p.y -= minY

p.z -= minZ

def reverseElementsId(elementsId, xSize, ySize, zSize):

print("Starting reverseElementsId " + " | " + time.ctime(time.time()))

start = time.time()

reverseElements = []

for i in [x \* xSideSize for x in range(0, int(xSize / xSideSize))]:

for j in [x \* ySideSize for x in range(0, int(ySize / ySideSize))]:

for k in [x \* zSideSize for x in range(0, int(zSize / zSideSize))]:

xCmpnt = i / xSideSize

yCmpnt = j \* xSize / xSideSize / ySideSize

zCmpnt = k \* xSize \* ySize / xSideSize / ySideSize / zSideSize

reverseElement = int(xCmpnt + yCmpnt + zCmpnt + 1)

reverseElements.append(reverseElement)

reverseElements = [x for x in reverseElements if x not in elementsId]

reverseElements.sort()

end = time.time()

print('total time (s)= ' + str(end - start))

print("Exiting reverseElementsId " + " | " + time.ctime(time.time()))

return reverseElements

print("Starting " + " | " + time.ctime(time.time()))

start = time.time()

with open(file) as f:

for line in f:

process(line)

end = time.time()

print('total time (s)= ' + str(end-start))

print("Exiting " + " | " + time.ctime(time.time()))

print("\nStarting lead\_points\_to\_origin " + " | " + time.ctime(time.time()))

start = time.time()

leadPointsToOrigin()

end = time.time()

print('total time (s)= ' + str(end - start))

print("Exiting lead\_points\_to\_origin " + " | " + time.ctime(time.time()))

maxX = max(points, key=attrgetter('x')).x + xSideSize

maxY = max(points, key=attrgetter('y')).y + ySideSize

maxZ = max(points, key=attrgetter('z')).z + zSideSize

elementsId = []

print("\nStarting elementsId " + " | " + time.ctime(time.time()))

start = time.time()

for point in points:

xComponent = point.x / xSideSize

yComponent = point.y \* maxX / xSideSize / ySideSize

zComponent = point.z \* maxX / xSideSize \* maxY / ySideSize / zSideSize

elementId = int(xComponent + yComponent + zComponent + 1)

elementsId.append(elementId)

end = time.time()

print('total time (s)= ' + str(end-start))

print("Exiting elementsId " + " | " + time.ctime(time.time()))

reverseElements = reverseElementsId(elementsId, maxX, maxY, maxZ)

def get\_list\_for\_delete(reverseElements):

print("\nStarting get\_list\_for\_delete " + " | " + time.ctime(time.time()))

start = time.time()

delete = ""

for el in reverseElements:

delete += str(el) + ','

delete = delete[:-1]

end = time.time()

print('total time (s)= ' + str(end - start))

print("Exiting get\_list\_for\_delete " + " | " + time.ctime(time.time()))

return delete

# расчет обратного массива к массиву элементов

def reverse\_elements\_id\_helper(i, reverseElements):

xSize = maxX

ySize = maxY

zSize = maxZ

for j in [x \* ySideSize for x in range(0, int(ySize / ySideSize))]:

for k in [x \* zSideSize for x in range(0, int(zSize / zSideSize))]:

xCmpnt = i / xSideSize

yCmpnt = j \* xSize / xSideSize / ySideSize

zCmpnt = k \* xSize \* ySize / xSideSize / ySideSize / zSideSize

reverseElement = int(xCmpnt + yCmpnt + zCmpnt + 1)

reverseElements.append(reverseElement)

# расчет обратного массива к массиву элементов

def reverse\_elements\_id():

processes = list()

print("\nStarting reverse\_elements\_id " + " | " + time.ctime(time.time()))

start = time.time()

reverseElements = []

xSize = maxX

m = Manager()

my\_shared\_list = m.list(reverseElements)

for i in [x \* xSideSize for x in range(0, int(xSize / xSideSize))]:

process = Process(target=reverse\_elements\_id\_helper, args=(i, my\_shared\_list,))

process.start()

processes.append(process)

for process in processes:

process.join()

my\_shared\_list = [x for x in my\_shared\_list if x not in elementsId]

my\_shared\_list.sort()

end = time.time()

print('total time (s)= ' + str(end - start))

print("Exiting reverse\_elements\_id " + " | " + time.ctime(time.time()))

return my\_shared\_list

def reverse\_elements\_id\_threading():

threads = []

print("\nStarting reverse\_elements\_id " + " | " + time.ctime(time.time()))

start = time.time()

reverseElements = []

xSize = maxX

for i in [x \* xSideSize for x in range(0, int(xSize / xSideSize))]:

thread = myThreadFirst(i + 1, "Thread-" + str(i + 1), i, reverseElements,)

thread.start()

threads.append(thread)

for thread in threads:

thread.join()

reverseElements = [x for x in reverseElements if x not in elementsId]

reverseElements.sort()

end = time.time()

print('total time (s)= ' + str(end - start))

print("Exiting reverse\_elements\_id " + " | " + time.ctime(time.time()))

return reverseElements

# преобразование массива точек в массив элементов

def get\_elements\_ids\_from\_point(point):

xComponent = point.x / xSideSize

yComponent = point.y \* maxX / xSideSize / ySideSize

zComponent = point.z \* maxX / xSideSize \* maxY / ySideSize / zSideSize

elementId = int(xComponent + yComponent + zComponent + 1)

elementsId.append(elementId)

def get\_elements\_ids\_from\_points():

print("\nStarting get\_elements\_ids\_from\_points " + " | " + time.ctime(time.time()))

start = time.time()

for point in points:

get\_elements\_ids\_from\_point(point)

end = time.time()

print('total time (s)= ' + str(end - start))

print("Exiting get\_elements\_ids\_from\_points " + " | " + time.ctime(time.time()))

def lead\_points\_to\_origin\_for\_point(point):

point.x -= minX

point.y -= minY

point.z -= minZ

# приведение к началу координат

def lead\_points\_to\_origin():

print("\nStarting lead\_points\_to\_origin " + " | " + time.ctime(time.time()))

start = time.time()

for p in points:

lead\_points\_to\_origin\_for\_point(p)

end = time.time()

print('total time (s)= ' + str(end - start))

print("Exiting lead\_points\_to\_origin " + " | " + time.ctime(time.time()))

# считывание точки из строчки

def get\_points\_from\_lines(lines):

pool = Pool()

pool.map(process, lines)

pool.close()

pool.join()

end = time.time()

print('total time (s)= ' + str(end - start))

print("Exiting get\_points\_from\_lines " + " | " + time.ctime(time.time()))

# функция, считывающая файл по строчкам

def get\_lines\_from\_file(file\_path):

text\_file = open(file\_path, "r")

lines = text\_file.readlines(3000000)

return lines

if \_\_name\_\_ == '\_\_main\_\_':

print("\nStarting get\_points\_from\_lines " + " | " + time.ctime(time.time()))

start = time.time()

lines = get\_lines\_from\_file(file)

get\_points\_from\_lines(lines)

# приведение к началу координат

minX = min(points, key=attrgetter('x')).x

minY = min(points, key=attrgetter('y')).y

minZ = min(points, key=attrgetter('z')).z

lead\_points\_to\_origin()

# вычисление размеров модели

maxX = max(points, key=attrgetter('x')).x + xSideSize

maxY = max(points, key=attrgetter('y')).y + ySideSize

maxZ = max(points, key=attrgetter('z')).z + zSideSize

print("maxX = " + str(maxX) + ", " + "maxY = " + str(maxY) + ", " + "maxZ = " + str(maxZ))

# преобразование точек в элементы

get\_elements\_ids\_from\_points()

reverseElements = reverse\_elements\_id\_threading()

deleteString = get\_list\_for\_delete(reverseElements)

# создание блока и задание свойств материала

newObjectId = 1

cubit.silent\_cmd('create brick x ' + str(maxX) + ' y ' + str(maxY) + ' z ' + str(maxZ))

cubit.silent\_cmd('curve 2 4 6 8 size 1.5' + + str(zSideSize))

cubit.silent\_cmd('curve 2 4 6 8 scheme equal')

cubit.silent\_cmd('curve 11 12 10 9 1 3 5 7 size ' + str(xSideSize))

cubit.silent\_cmd('curve 11 12 10 9 1 3 5 7 scheme equal')

cubit.silent\_cmd('volume 1 scheme Map')

cubit.silent\_cmd('mesh volume ' + str(newObjectId))

cubit.cmd('create material "material" property\_group "HOOK"')

cubit.cmd('modify material "material" scalar\_properties "MODULUS" 7e+10 "POISSON" 0.15')

cubit.silent\_cmd('set duplicate block elements off')

cubit.silent\_cmd('block 1 add volume ' + str(newObjectId))

cubit.silent\_cmd('block 1 element type hex8')

cubit.silent\_cmd('block 1 material \'material\'')

# удаление элементов

cubit.silent\_cmd('delete hex ' + str(delete))

cubit.cmd('draw block 1')

1. **Модель керна**

# подключение внешних библиотек

import threading

import time

exitFlag = 0

class myThread (threading.Thread):

def \_\_init\_\_(self, threadID, name, counter):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.name = name

self.counter = counter

def run(self):

print("Starting " + self.name + " | " + time.ctime(time.time()))

# print\_time(self.name, self.counter, 5)

loop\_test(self.name)

print("Exiting " + self.name + " | " + time.ctime(time.time()))

class myThreadFirst (threading.Thread):

def \_\_init\_\_(self, threadID, name, k, delete):

threading.Thread.\_\_init\_\_(self)

self.threadID = threadID

self.name = name

self.k = k

self.delete = delete

def run(self):

# print("Starting " + self.name + " | " + time.ctime(time.time()))

loop\_step(self.name, self.k, self.delete)

# print("Exiting " + self.name + " | " + time.ctime(time.time()))

def print\_time(threadName, delay, counter):

while counter:

if exitFlag:

threadName.exit()

time.sleep(delay)

counter -= 1

# функция, преобразующая значения из файла в id элемента

def loop\_step(threadName, k, delete):

for j in range(start\_y, calc\_y - 1):

for i in range(start\_x, calc\_x - 1):

data = i % 2

xCoord = i / meshSize

yCoord = j / meshSize

zCoord = k / meshSize

elementId = xCoord + yCoord \* (calc\_x - 1) + zCoord \* (calc\_x - 1) \* (calc\_z - 1)

if data == traceHold1:

delete.append(elementId)

if exitFlag:

threadName.exit()

def loop\_test(threadName):

for y in range(5):

for x in range(5):

if exitFlag:

threadName.exit()

print(x + y \* 10)

# задание констант для рассчета

orig\_x = 1301 # размер по x полной модели

orig\_y = 1301 # размер по y полной модели

orig\_z = 1301 # размер по z полной модели

calc\_x = 31 # вычисляемый размер по x

calc\_y = 31 # вычисляемый размер по y

calc\_z = 31 # вычисляемый размер по z

start\_x = 0

start\_y = 0

start\_z = 0

dist\_x = (orig\_x - calc\_x) / 2

dist\_y = (orig\_y - calc\_y) / 2

dist\_z = (orig\_z - calc\_z) / 2

elementId = 0

meshSize = 1

objectId = 2

newObjectId = 1

traceHold1 = 1

traceHold2 = 100

traceHold3 = 200

delete = []

threads = []

print("Starting " + " | " + time.ctime(time.time()))

start = time.time()

# считывание файла

for k in range(start\_z, calc\_z - 1):

thread = myThreadFirst(k + 1, "Thread-" + str(k + 1), k, delete)

thread.start()

threads.append(thread)

end = time.time()

print('total time (s)= ' + str(end-start))

print("Exiting " + " | " + time.ctime(time.time()))

for t in threads:

t.join()

# создание строки для удаления элементов

elementsId = ''

for element in delete:

elementsId += str(element) + ','

elementsId = elementsId[0:-1]

# создание блока и задание свойств материала

newObjectId = 1

cubit.silent\_cmd('create brick x ' + str(maxX) + ' y ' + str(maxY) + ' z ' + str(maxZ))

cubit.silent\_cmd('curve 2 4 6 8 size 1.5' + + str(zSideSize))

cubit.silent\_cmd('curve 2 4 6 8 scheme equal')

cubit.silent\_cmd('curve 11 12 10 9 1 3 5 7 size ' + str(xSideSize))

cubit.silent\_cmd('curve 11 12 10 9 1 3 5 7 scheme equal')

cubit.silent\_cmd('volume 1 scheme Map')

cubit.silent\_cmd('mesh volume ' + str(newObjectId))

cubit.cmd('create material "material" property\_group "HOOK"')

cubit.cmd('modify material "material" scalar\_properties "MODULUS" 7e+10 "POISSON" 0.15')

cubit.silent\_cmd('set duplicate block elements off')

cubit.silent\_cmd('block 1 add volume ' + str(newObjectId))

cubit.silent\_cmd('block 1 element type hex8')

cubit.silent\_cmd('block 1 material \'material\'')

# удаление элементов

cubit.silent\_cmd('delete hex ' + str(delete))

cubit.cmd('draw block 1')