Санкт-Петербургский политехнический университет

Высшая школа теоретической механики, ФизМех

Направление подготовки

«01.03.03 Механика и математическое моделирование»

Индивидуальное задание № 4

тема "Метод конечных элементов. Решение плоской задачи теории упругости"

дисциплина "Вычислительная механика"

Вариант 2

Выполнила студентка гр. 5030103/90301 М.А.Бенюх

Преподаватель: Е.Ю. Витохин

Санкт-Петербург

2022

Содержание:

1. Формулировка задачи.

Рассматривается плотина, состоящая из двух инженерно-геологических элементов, внешнего и внутреннего, вместе с основанием. Задана граница контакта с водой. Требуется вычислить узловые перемещения, деформации и напряжения, возникающие в плотине, используя метод конечных элементов. На плотину действует гравитационная сила, а на ее элементы, соприкасающиеся с водой, действует давление столба жидкости. Будем полагать, что боковые стороны основания плотины закреплены по оси OX, а его низ закреплен по оси OY.

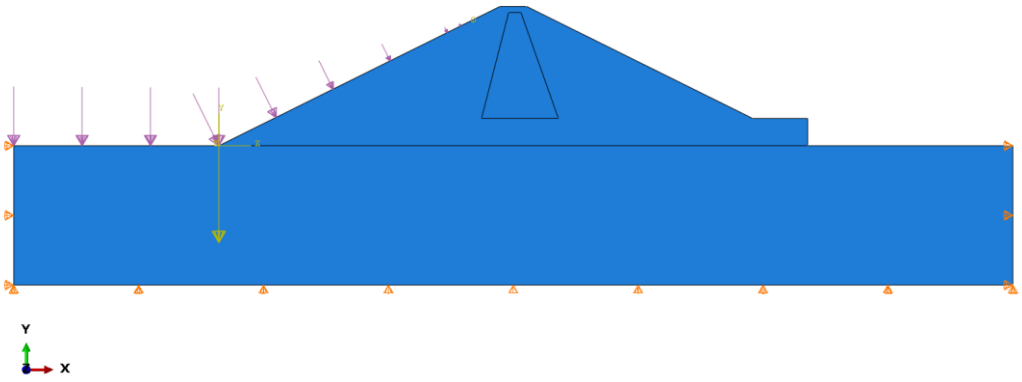

Рис.1. Постановка задачи

Параметр	Значение
Ускорение свободного падения, $\boldsymbol{g}$	$9.8\ \frac{м}{с^2}$
Плотность воды, $\boldsymbol{\rho}$	$1000\ \frac{кг}{м^3}$
Плотность внешней части плотины, $\boldsymbol{\rho_1}$	$2500\ \frac{кг}{м^3}$
Плотность внутренней части плотины, $\boldsymbol{\rho_2}$	$2200\ \frac{кг}{м^3}$
Модуль Юнга для внешней части плотины, $\boldsymbol{E_1}$	$25 * 10^9$ Па
Модуль Юнга для внутренней части плотины, $\boldsymbol{E_2}$	$22 * 10^9$ Па
Модуль Юнга для основания плотины, $\boldsymbol{E_f}$	$17 * 10^9$ Па
Коэффициент Пуассона (одинаковый для всех трех элементов) $\boldsymbol{\nu_1 = \nu_2 = \nu_f}$	0.2

Таблица 1. Параметры задачи

2. Алгоритм метода.

Рассмотрим треугольный конечный элемент 1-го порядка. Перемещение в каждом элементе в этом случае описывается линейным многочленом:

$$T = A + Bx + Cy$$

Запишем вектор-столбец узловых перемещений в конечном элементе.

$$\{u\}^T = \{u_i^x \;\; u_i^y \;\; u_j^x \;\; u_j^y \;\; u_k^x \;\; u_k^y\}$$

Перемещения в точках конечного элемента зададим с помощью функций форм:

$$\begin{aligned} u_x &= u_i^x N_i^x + u_j^x N_j^x + u_k^x N_k^x \\ u_y &= u_i^y N_i^y + u_j^y N_j^y + u_k^y N_k^y \end{aligned} \tag{1}$$

$$u = \begin{Bmatrix} u_x \\ u_y \end{Bmatrix} = [N]\{u\}, \quad \text{где}$$

$$[N] = \begin{bmatrix} N_i^x & 0 & N_j^x & 0 & N_k^x & 0 \\ 0 & N_i^y & 0 & N_j^y & 0 & N_k^y \end{bmatrix} - \text{матрица функций форм.}$$

Будем использовать принцип минимизации функционала потенциальной энергии. Потенциальную энергию можно найти как разность энергии внутренних сил Λ и работы внешних сил W:

$$\Pi = \Lambda - W$$

$$d\Pi = d\Lambda - dW, \qquad \mathrm{d}\Lambda = \frac{1}{2}\{\varepsilon\}^T\{\sigma\}\, dV$$

$$\Lambda = \frac{1}{2}\int_V \{\varepsilon\}^T\{\sigma\}\, dV \tag{2}$$

$$\{\varepsilon\}^T = \{\varepsilon_x \;\; \varepsilon_y \;\; 2\varepsilon_{xy}\} - \text{вектор} - \text{столбец деформаций.} \tag{3}$$

Компоненты вектора деформаций запишутся как:

$$\varepsilon_x = \frac{\partial u_x}{\partial x}, \qquad \varepsilon_y = \frac{\partial u_y}{\partial y}, \quad 2\varepsilon_{xy} = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \tag{4}$$

Подставим (1) в (2), а затем в (3):

$$\{\varepsilon\} = \begin{Bmatrix} \dfrac{\partial N_i}{\partial x}u_i^x + \dfrac{\partial N_j}{\partial x}u_j^x + \dfrac{\partial N_k}{\partial x}u_k^x \\ \dfrac{\partial N_i}{\partial y}u_i^y + \dfrac{\partial N_j}{\partial y}u_j^y + \dfrac{\partial N_k}{\partial y}u_k^y \\ \dfrac{\partial N_i}{\partial y}u_i^x + \dfrac{\partial N_i}{\partial x}u_i^y + \dfrac{\partial N_j}{\partial y}u_j^x + \dfrac{\partial N_j}{\partial x}u_j^y + \dfrac{\partial N_k}{\partial y}u_k^x + \dfrac{\partial N_k}{\partial x}u_k^y \end{Bmatrix}$$

Можем вынести компоненты вектора перемещений.

$$\{\varepsilon\} = [B]\{u\} \tag{5}$$

Где $[B]$ – матрица градиентов,

$$[B] = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 & \frac{\partial N_j}{\partial x} & 0 & \frac{\partial N_k}{\partial x} & 0 \\ 0 & \frac{\partial N_i}{\partial y} & 0 & \frac{\partial N_j}{\partial y} & 0 & \frac{\partial N_k}{\partial y} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} & \frac{\partial N_j}{\partial y} & \frac{\partial N_j}{\partial x} & \frac{\partial N_k}{\partial y} & \frac{\partial N_k}{\partial x} \end{bmatrix}$$

Зададим вектор-столбец напряжений: $\{\sigma\}^T = \{\sigma_x \;\; \sigma_y \;\; \sigma_{xy}\}$

Физические соотношения для плосконапряженного состояния:

$$\sigma_x = \frac{E}{1-\nu^2}(\varepsilon_x + \nu\varepsilon_y)$$
$$\sigma_y = \frac{E}{1-\nu^2}(\nu\varepsilon_x + \varepsilon_y)$$
$$\sigma_{xy} = \frac{E}{(1-\nu^2)^2}\varepsilon_{xy}$$

С учетом этих соотношений вектор-столбец напряжений распишется в виде:

$$\{\sigma\} = [D]\{\varepsilon\} = [D][B]\{u\}, \tag{6}$$

$$\text{где } [D] = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2(1-\nu)} \end{bmatrix}$$

- матрица упругих характеристик для плоского деформированного состояния.
Подставим (5) и (6) в (2):

$$\Lambda = \frac{1}{2}\int_V \{u\}^T[B]^T[D][B]\{u\}\,dV$$

$$W = W^c + W^i + W^V$$

Где $W^c -$ работа сосредоточенных сил, $W^i -$ работа поверхностных сил, $W^V -$ работа объемных сил.

Минимизируем функционал потенциальной энергии:

$$\frac{\delta\Pi}{\delta\{u\}} = 0$$

$$\int_V [B]^T[D][B]\{u\}\,dV = \{f^e\}$$

$$\{k^e\} = \int_V [B]^T[D][B]\,dV - \text{матрица жесткости конечного элемента}$$

$$\{k^e\} = \int_V [B]^T[D][B]\,dV = [B]^T[D][B]\int_V dV$$

Следовательно, уравнение метода конечных элементов будет выглядеть:

$$\{k^e\}\{u\} = \{f^e\}$$

Вычислим глобальную матрицу жесткости и вектор-столбец нагрузок:

$$[K] = \sum_e [k^e]$$

$$[F] = \sum_e [k^e]$$

Итоговое уравнение МКЭ:

$$[K]\{U\} = [F]$$

Для вычисления $[B]$ введем матрицу $[J]$:

$$\begin{Bmatrix} \frac{\partial N_m}{\partial x} \\ \frac{\partial N_m}{\partial y} \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \frac{\partial N_m}{\partial \xi} \\ \frac{\partial N_m}{\partial \eta} \end{Bmatrix}, \qquad m = i, j, k$$

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_i}{\partial \xi} & \frac{\partial N_j}{\partial \xi} & \frac{\partial N_k}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} & \frac{\partial N_j}{\partial \eta} & \frac{\partial N_k}{\partial \eta} \end{bmatrix} \begin{bmatrix} x_i & y_i \\ x_j & y_j \\ x_k & y_k \end{bmatrix}$$

Функции форм для треугольного элемента:

$$N_i = 1 - \xi - \eta, \qquad N_j = \xi, \qquad N_k = \eta$$

Вычисление интеграла по объему распишется как:

$$\int_V dV = t\int_{V^*} |J|\,d\xi d\eta = t\frac{|J|}{2}, \qquad t - \text{толщина}$$

3.1. Результаты

номер узла	U1 Abaqus	U1 Python	U2 Abaqus	U2 Python
1	1,692176E-30	0,000000E+00	3,725814E-04	3,581674E-04
2	1,534408E-03	1,434220E-03	-6,108267E-04	-6,170981E-04
3	1,047517E-03	9,479322E-04	-2,353027E-03	-2,343518E-03
4	-3,181032E-31	0,000000E+00	-2,204853E-03	-2,194059E-03
5	-4,790206E-31	0,000000E+00	-3,451336E-30	0,000000E+00
6	2,116019E-30	0,000000E+00	1,212714E-31	0,000000E+00
7	1,228434E-03	1,121351E-03	-6,634132E-03	-6,545845E-03
8	1,419055E-03	1,291594E-03	-6,632608E-03	-6,506091E-03
9	1,626057E-03	1,468080E-03	-6,493271E-03	-6,323039E-03
10	1,232332E-03	1,121081E-03	-7,049136E-04	-7,122144E-04
11	1,344611E-03	1,225033E-03	-1,625914E-03	-1,637445E-03
12	1,106176E-03	9,984208E-04	-6,496141E-03	-6,434948E-03
13	1,668911E-03	1,509751E-03	-5,307710E-03	-5,271008E-03
14	1,568139E-03	1,394845E-03	-5,680173E-03	-5,586259E-03
15	1,218439E-03	1,104896E-03	-6,600267E-03	-6,521401E-03
16	1,157494E-03	1,044072E-03	-6,537027E-03	-6,470535E-03
17	2,109047E-04	1,999612E-04	3,722779E-04	3,579221E-04
18	4,162470E-04	3,943577E-04	3,618603E-04	3,478024E-04
19	6,140665E-04	5,811137E-04	3,409785E-04	3,274461E-04
20	7,994879E-04	7,552763E-04	2,985574E-04	2,858033E-04
21	9,697017E-04	9,138482E-04	2,208118E-04	2,091617E-04
22	1,128900E-03	1,060639E-03	8,441410E-05	7,423192E-05
23	1,293024E-03	1,210686E-03	-1,520114E-04	-1,604644E-04
24	1,624669E-03	1,517569E-03	-1,004429E-03	-1,014600E-03
25	1,734544E-03	1,620470E-03	-1,428680E-03	-1,439062E-03
26	1,846534E-03	1,725783E-03	-1,911735E-03	-1,921340E-03
27	1,926581E-03	1,799353E-03	-2,408337E-03	-2,416427E-03
28	1,977735E-03	1,844206E-03	-2,901971E-03	-2,907354E-03
29	1,999986E-03	1,860258E-03	-3,369923E-03	-3,371206E-03
30	1,993738E-03	1,847938E-03	-3,794357E-03	-3,789892E-03
31	1,961593E-03	1,809838E-03	-4,165527E-03	-4,153591E-03
32	1,905708E-03	1,748252E-03	-4,469669E-03	-4,448331E-03
33	1,832374E-03	1,669565E-03	-4,702331E-03	-4,669558E-03
34	1,760020E-03	1,592322E-03	-4,876026E-03	-4,829307E-03
35	1,686306E-03	1,514732E-03	-4,997692E-03	-4,935163E-03
36	1,607930E-03	1,434168E-03	-5,070107E-03	-4,990977E-03
37	1,526732E-03	1,353051E-03	-5,082422E-03	-4,987656E-03
38	1,441637E-03	1,270666E-03	-5,025352E-03	-4,917733E-03
39	1,367469E-03	1,201208E-03	-4,905925E-03	-4,789819E-03
40	1,307519E-03	1,147288E-03	-4,722816E-03	-4,603977E-03
41	1,268149E-03	1,114386E-03	-4,476493E-03	-4,361019E-03
42	1,233653E-03	1,086860E-03	-4,181359E-03	-4,075018E-03
43	1,223299E-03	1,083102E-03	-3,843132E-03	-3,751005E-03
44	1,213323E-03	1,080480E-03	-3,479550E-03	-3,405406E-03
45	1,219470E-03	1,093550E-03	-3,064704E-03	-3,012219E-03

46	1,184613E-03	1,068612E-03	-2,671157E-03	-2,641011E-03
47	7,632061E-04	6,842344E-04	-2,238465E-03	-2,229826E-03
48	5,668793E-04	5,034999E-04	-2,190292E-03	-2,182320E-03
49	4,219081E-04	3,715206E-04	-2,177318E-03	-2,169064E-03
50	3,087242E-04	2,697728E-04	-2,180035E-03	-2,171097E-03
51	2,167385E-04	1,882148E-04	-2,188177E-03	-2,178510E-03
52	1,381060E-04	1,193842E-04	-2,196667E-03	-2,186391E-03
53	6,767424E-05	5,837007E-05	-2,202786E-03	-2,192107E-03
54	-6,407217E-31	0,000000E+00	-1,827796E-03	-1,819223E-03
55	-6,881282E-31	0,000000E+00	-1,453099E-03	-1,446664E-03
56	-7,694835E-31	0,000000E+00	-1,083385E-03	-1,078863E-03
57	-8,558164E-31	0,000000E+00	-7,187590E-04	-7,159132E-04
58	-9,159327E-31	0,000000E+00	-3,583053E-04	-3,569354E-04
59	4,073410E-05	3,280834E-05	-6,880994E-30	0,000000E+00
60	7,954989E-05	6,380116E-05	-6,856452E-30	0,000000E+00
61	1,140643E-04	9,072054E-05	-6,816795E-30	0,000000E+00
62	1,421653E-04	1,115838E-04	-6,804799E-30	0,000000E+00
63	1,623043E-04	1,249538E-04	-6,839006E-30	0,000000E+00
64	1,746290E-04	1,310135E-04	-6,976262E-30	0,000000E+00
65	1,818641E-04	1,323854E-04	-7,289442E-30	0,000000E+00
66	1,902109E-04	1,349657E-04	-7,827019E-30	0,000000E+00
67	2,080800E-04	1,467128E-04	-8,600309E-30	0,000000E+00
68	2,437712E-04	1,754642E-04	-9,570506E-30	0,000000E+00
69	3,036470E-04	2,272550E-04	-1,065506E-29	0,000000E+00
70	3,896960E-04	3,040700E-04	-1,175070E-29	0,000000E+00
71	5,012550E-04	4,054787E-04	-1,278693E-29	0,000000E+00
72	6,358036E-04	5,293657E-04	-1,370984E-29	0,000000E+00
73	7,897092E-04	6,725913E-04	-1,448583E-29	0,000000E+00
74	9,587662E-04	8,314689E-04	-1,510032E-29	0,000000E+00
75	1,138745E-03	1,002301E-03	-1,554078E-29	0,000000E+00
76	1,325368E-03	1,181235E-03	-1,579701E-29	0,000000E+00
77	1,513995E-03	1,363961E-03	-1,586717E-29	0,000000E+00
78	1,700029E-03	1,546101E-03	-1,575650E-29	0,000000E+00
79	1,879025E-03	1,723245E-03	-1,547027E-29	0,000000E+00
80	2,046569E-03	1,890927E-03	-1,501121E-29	0,000000E+00
81	2,198229E-03	2,044571E-03	-1,437245E-29	0,000000E+00
82	2,328736E-03	2,178699E-03	-1,356228E-29	0,000000E+00
83	2,432890E-03	2,287860E-03	-1,258593E-29	0,000000E+00
84	2,505970E-03	2,367104E-03	-1,146695E-29	0,000000E+00
85	2,544706E-03	2,412899E-03	-1,024368E-29	0,000000E+00
86	2,545827E-03	2,421788E-03	-8,939679E-30	0,000000E+00
87	2,506996E-03	2,391271E-03	-7,588904E-30	0,000000E+00
88	2,427958E-03	2,320929E-03	-6,240211E-30	0,000000E+00
89	2,309922E-03	2,211828E-03	-4,930656E-30	0,000000E+00
90	2,155122E-03	2,066086E-03	-3,711872E-30	0,000000E+00
91	1,967919E-03	1,888025E-03	-2,636132E-30	0,000000E+00
92	1,753977E-03	1,683341E-03	-1,734820E-30	0,000000E+00
93	1,520037E-03	1,458858E-03	-1,025268E-30	0,000000E+00

94	1,273255E-03	1,221789E-03	-5,074873E-31	0,000000E+00
95	1,019610E-03	9,781201E-04	-1,572212E-31	0,000000E+00
96	7,636870E-04	7,323929E-04	5,755993E-32	0,000000E+00
97	5,081731E-04	4,872332E-04	1,770797E-31	0,000000E+00
98	2,539580E-04	2,434655E-04	2,350830E-31	0,000000E+00
99	4,179649E-30	0,000000E+00	7,281161E-05	7,050827E-05
100	4,080495E-30	0,000000E+00	1,437636E-04	1,391270E-04
101	3,923390E-30	0,000000E+00	2,103073E-04	2,032835E-04
102	3,730771E-30	0,000000E+00	2,709672E-04	2,614993E-04
103	3,553131E-30	0,000000E+00	3,248717E-04	3,129203E-04
104	1,788390E-03	1,609424E-03	-6,281095E-03	-6,096612E-03
105	1,917192E-03	1,725310E-03	-5,990917E-03	-5,809009E-03
106	2,003856E-03	1,805490E-03	-5,628614E-03	-5,458437E-03
107	2,044567E-03	1,845455E-03	-5,215987E-03	-5,063253E-03
108	2,035974E-03	1,841432E-03	-4,761582E-03	-4,630526E-03
109	1,970588E-03	1,785410E-03	-4,274821E-03	-4,167775E-03
110	1,852153E-03	1,681077E-03	-3,763764E-03	-3,683000E-03
111	1,683476E-03	1,530239E-03	-3,268916E-03	-3,213265E-03
112	1,445127E-03	1,314291E-03	-2,776091E-03	-2,745197E-03
113	1,263208E-03	1,149367E-03	-1,115851E-03	-1,127289E-03
114	1,238129E-03	1,111816E-03	-2,156732E-03	-2,167560E-03
115	1,133353E-03	1,002542E-03	-2,754682E-03	-2,764447E-03
116	1,030507E-03	8,970940E-04	-3,350088E-03	-3,357507E-03
117	9,512723E-04	8,166256E-04	-3,948516E-03	-3,952134E-03
118	9,041855E-04	7,697171E-04	-4,526478E-03	-4,524837E-03
119	8,927133E-04	7,597769E-04	-5,062583E-03	-5,053871E-03
120	9,171084E-04	7,872971E-04	-5,559785E-03	-5,542262E-03
121	9,632841E-04	8,379221E-04	-5,978697E-03	-5,950127E-03
122	1,037648E-03	9,193849E-04	-6,308489E-03	-6,266401E-03
123	1,634025E-03	1,469497E-03	-5,478092E-03	-5,424668E-03
124	1,599365E-03	1,429777E-03	-5,608510E-03	-5,535538E-03
125	1,540652E-03	1,369310E-03	-6,137058E-03	-6,040580E-03
126	1,475398E-03	1,312881E-03	-6,436926E-03	-6,342918E-03
127	1,365453E-03	1,222729E-03	-6,588500E-03	-6,502084E-03
128	1,217819E-03	1,085997E-03	-6,472062E-03	-6,413152E-03
129	1,302287E-03	1,158279E-03	-6,314605E-03	-6,261323E-03
130	1,411844E-03	1,260015E-03	-6,069726E-03	-6,021956E-03
131	1,538652E-03	1,382187E-03	-5,734508E-03	-5,692240E-03
132	2,245950E-04	2,147394E-04	1,742792E-04	1,685726E-04
133	5,137537E-05	4,354076E-05	-1,263453E-03	-1,258116E-03
134	5,804207E-05	4,985970E-05	-1,679375E-03	-1,671772E-03
135	4,405428E-05	3,665791E-05	-8,707126E-04	-8,672531E-04
136	3,816629E-05	3,110821E-05	-4,796145E-04	-4,777935E-04
137	2,082846E-04	1,986247E-04	2,428442E-04	2,345307E-04
138	1,931564E-04	1,836148E-04	3,042655E-04	2,932760E-04
139	2,848463E-04	2,118773E-04	-4,737903E-04	-4,659421E-04
140	1,388949E-03	1,332467E-03	2,041563E-05	1,906115E-05
141	2,053857E-03	1,969262E-03	-1,026694E-04	-1,038652E-04

142	2,460983E-03	2,349396E-03	-2,994392E-04	-3,007622E-04
143	1,794148E-04	1,383636E-04	-3,197869E-04	-3,192816E-04
144	1,907098E-04	1,434986E-04	-3,280160E-04	-3,272887E-04
145	1,987241E-04	1,456799E-04	-3,467334E-04	-3,452269E-04
146	2,111225E-04	1,521954E-04	-3,769865E-04	-3,740045E-04
147	2,368736E-04	1,714991E-04	-4,185498E-04	-4,134225E-04
148	3,577801E-04	2,760575E-04	-5,319945E-04	-5,213441E-04
149	7,217439E-04	6,090383E-04	-6,816594E-04	-6,652518E-04
150	1,236347E-03	1,095096E-03	-7,610185E-04	-7,452819E-04
151	1,789006E-03	1,633393E-03	-7,567333E-04	-7,464547E-04
152	2,259684E-03	2,107388E-03	-6,708886E-04	-6,666503E-04
153	2,519684E-03	2,384091E-03	-5,075436E-04	-5,072986E-04
154	9,906932E-05	7,975125E-05	-3,307747E-04	-3,297398E-04
155	4,562731E-04	3,647702E-04	-5,877623E-04	-5,745999E-04
156	5,785836E-04	4,766008E-04	-6,382887E-04	-6,231587E-04
157	8,819766E-04	7,588093E-04	-7,168967E-04	-6,999624E-04
158	1,054840E-03	9,219892E-04	-7,435235E-04	-7,268573E-04
159	1,422288E-03	1,274278E-03	-7,689471E-04	-7,546892E-04
160	1,608011E-03	1,455183E-03	-7,674221E-04	-7,550692E-04
161	2,119156E-03	1,963962E-03	-7,087780E-04	-7,026733E-04
162	2,376323E-03	2,228435E-03	-6,240772E-04	-6,214456E-04
163	2,464043E-03	2,321813E-03	-5,687918E-04	-5,675219E-04
164	2,539693E-03	2,411528E-03	-4,411214E-04	-4,416374E-04
165	2,520512E-03	2,400402E-03	-3,708475E-04	-3,718857E-04
166	2,361208E-03	2,258511E-03	-2,284328E-04	-2,298309E-04
167	2,223752E-03	2,130131E-03	-1,612824E-04	-1,625962E-04
168	1,853616E-03	1,778172E-03	-5,175139E-05	-5,288345E-05
169	1,629312E-03	1,563221E-03	-1,028687E-05	-1,147437E-05
170	1,139141E-03	1,092545E-03	4,147024E-05	3,989959E-05
171	8,853629E-04	8,488928E-04	5,433222E-05	5,255435E-05
172	6,348932E-04	6,085769E-04	6,114323E-05	5,921111E-05
173	4,220877E-04	4,045525E-04	5,853829E-05	5,669318E-05
174	1,337738E-04	1,068260E-04	-3,387779E-04	-3,379272E-04
175	1,609865E-04	1,266626E-04	-3,240090E-04	-3,234024E-04
176	1,960834E-03	1,804469E-03	-7,372019E-04	-7,290479E-04
177	1,763567E-03	1,667749E-03	-6,523416E-04	-6,594745E-04
178	2,144121E-03	2,015292E-03	-2,274708E-03	-2,280399E-03
179	1,895788E-03	1,790411E-03	-1,019703E-03	-1,028362E-03
180	7,471960E-04	6,586877E-04	-2,087327E-03	-2,071351E-03
181	8,234328E-04	7,239311E-04	-2,415063E-03	-2,380897E-03
182	1,001286E-03	8,725477E-04	-3,441796E-03	-3,357509E-03
183	1,224702E-03	1,071378E-03	-4,074437E-03	-3,976381E-03
184	1,553403E-03	1,385484E-03	-4,265574E-03	-4,191785E-03
185	1,876184E-03	1,712157E-03	-4,073036E-03	-4,037658E-03
186	2,122169E-03	1,973614E-03	-3,373778E-03	-3,365882E-03
187	8,945714E-04	7,841321E-04	-2,799257E-03	-2,745465E-03
188	9,504398E-04	8,304526E-04	-3,143221E-03	-3,072452E-03
189	1,064920E-03	9,274025E-04	-3,710094E-03	-3,616141E-03

190	1,140401E-03	9,944468E-04	-3,934747E-03	-3,835786E-03
191	1,329109E-03	1,169005E-03	-4,194968E-03	-4,101535E-03
192	1,436488E-03	1,271570E-03	-4,250715E-03	-4,165644E-03
193	1,667617E-03	1,498877E-03	-4,262267E-03	-4,201103E-03
194	1,775014E-03	1,607745E-03	-4,193554E-03	-4,145647E-03
195	1,976438E-03	1,816976E-03	-3,863151E-03	-3,839002E-03
196	2,063564E-03	1,909324E-03	-3,625395E-03	-3,610369E-03
197	2,161317E-03	2,018930E-03	-3,045812E-03	-3,043804E-03
198	2,166984E-03	2,031215E-03	-2,682929E-03	-2,685460E-03
199	2,087213E-03	1,965546E-03	-1,858495E-03	-1,866162E-03
200	2,001743E-03	1,887705E-03	-1,436527E-03	-1,445214E-03
201	3,372303E-04	2,935826E-04	-1,793903E-03	-1,787854E-03
202	4,354566E-04	3,807662E-04	-1,783748E-03	-1,778498E-03
203	5,463963E-04	4,798990E-04	-1,809629E-03	-1,804424E-03
204	6,602579E-04	5,818049E-04	-1,877927E-03	-1,871480E-03
205	2,484282E-04	2,153284E-04	-1,802854E-03	-1,795916E-03
206	1,707700E-04	1,475319E-04	-1,824602E-03	-1,816831E-03
207	7,470485E-04	7,088528E-04	2,789977E-04	2,683707E-04
208	1,385827E-03	1,311799E-03	-4,804116E-05	-5,476553E-05
209	9,613985E-04	9,115126E-04	2,254791E-04	2,157946E-04
210	1,175923E-03	1,114089E-03	1,251157E-04	1,168131E-04
211	5,265512E-04	4,998050E-04	3,082492E-04	2,968449E-04
212	1,597252E-03	1,511583E-03	-3,280794E-04	-3,331397E-04
213	2,301117E-04	2,204290E-04	9,803157E-05	9,490985E-05
214	9,354852E-05	7,832822E-05	-1,014561E-03	-1,010696E-03
215	4,318084E-04	4,122942E-04	2,095172E-04	2,025025E-04
216	2,504170E-03	2,379361E-03	-8,026810E-04	-8,043112E-04
217	9,925024E-04	9,506437E-04	9,925322E-05	9,579537E-05
218	1,716269E-03	1,644773E-03	-5,933807E-05	-6,163530E-05
219	2,667416E-04	2,021736E-04	-7,975718E-04	-7,896137E-04
220	3,032344E-04	2,315997E-04	-8,850940E-04	-8,721868E-04
221	5,557845E-04	4,562850E-04	-1,211343E-03	-1,183071E-03
222	9,919123E-04	8,612600E-04	-1,461003E-03	-1,427071E-03
223	1,522132E-03	1,369554E-03	-1,526612E-03	-1,500239E-03
224	2,028474E-03	1,871458E-03	-1,422504E-03	-1,408663E-03
225	2,399205E-03	2,253765E-03	-1,211445E-03	-1,207809E-03
226	1,276182E-04	1,048965E-04	-7,346511E-04	-7,323451E-04
227	6,850443E-04	5,748832E-04	-1,301819E-03	-1,270403E-03
228	1,162912E-03	1,023174E-03	-1,557075E-03	-1,523314E-03
229	1,698243E-03	1,542234E-03	-1,484626E-03	-1,462667E-03
230	2,469045E-03	2,329680E-03	-1,056072E-03	-1,054762E-03
231	2,464250E-03	2,347708E-03	-6,609691E-04	-6,633897E-04
232	1,488704E-03	1,426643E-03	1,315527E-05	1,060366E-05
233	4,759884E-04	4,556652E-04	1,301067E-04	1,259255E-04
234	1,926610E-03	1,845862E-03	-1,558264E-04	-1,581680E-04
235	2,171427E-03	2,016652E-03	-1,385091E-03	-1,374908E-03
236	2,113051E-03	2,023122E-03	-2,678289E-04	-2,703824E-04
237	7,461037E-04	7,144625E-04	1,170970E-04	1,132876E-04

238	1,993188E-04	1,600566E-04	-6,540889E-04	-6,529137E-04
239	2,266400E-03	2,167439E-03	-3,885847E-04	-3,913523E-04
240	3,623026E-04	2,824963E-04	-9,895022E-04	-9,711527E-04
241	4,474253E-04	3,581715E-04	-1,102833E-03	-1,079110E-03
242	8,323313E-04	7,115946E-04	-1,379716E-03	-1,346532E-03
243	1,343206E-03	1,195917E-03	-1,586088E-03	-1,554877E-03
244	2,298781E-03	2,147912E-03	-1,355772E-03	-1,348901E-03
245	2,505014E-03	2,372489E-03	-9,348250E-04	-9,351870E-04
246	2,384599E-03	2,276758E-03	-5,178427E-04	-5,205964E-04
247	2,208209E-04	1,746888E-04	-6,539173E-04	-6,527417E-04
248	2,362601E-04	1,838219E-04	-6,833136E-04	-6,812799E-04
249	2,493510E-04	1,908705E-04	-7,392347E-04	-7,349115E-04
250	1,244611E-03	1,192432E-03	6,594946E-05	6,293009E-05
251	1,699041E-04	1,381819E-04	-6,833833E-04	-6,817491E-04
252	1,869075E-03	1,711603E-03	-1,443661E-03	-1,425982E-03
253	7,187836E-04	6,155372E-04	-2,491973E-03	-2,440046E-03
254	9,776220E-04	8,449602E-04	-3,205774E-03	-3,125729E-03
255	1,336470E-03	1,180383E-03	-3,512183E-03	-3,438345E-03
256	1,748231E-03	1,583428E-03	-3,496124E-03	-3,450271E-03
257	1,993906E-03	1,833417E-03	-3,281228E-03	-3,255444E-03
258	2,129332E-03	1,973824E-03	-2,932982E-03	-2,917007E-03
259	2,292767E-03	2,154244E-03	-2,385718E-03	-2,385496E-03
260	2,195816E-03	2,079868E-03	-1,391139E-03	-1,397948E-03
261	6,382737E-04	5,455038E-04	-2,169444E-03	-2,132808E-03
262	7,954084E-04	6,822362E-04	-2,766646E-03	-2,701997E-03
263	8,807706E-04	7,577424E-04	-3,006333E-03	-2,932105E-03
264	1,079772E-03	9,386819E-04	-3,322002E-03	-3,241086E-03
265	1,189730E-03	1,041950E-03	-3,251866E-03	-3,177639E-03
266	1,462020E-03	1,302503E-03	-3,327488E-03	-3,265062E-03
267	1,616659E-03	1,452692E-03	-3,507588E-03	-3,451939E-03
268	2,186435E-03	2,035858E-03	-2,939866E-03	-2,929644E-03
269	2,259235E-03	2,113982E-03	-2,657551E-03	-2,652612E-03
270	2,292261E-03	2,160873E-03	-2,081865E-03	-2,085080E-03
271	2,260275E-03	2,136402E-03	-1,742688E-03	-1,748196E-03
272	2,105867E-03	1,998553E-03	-1,032308E-03	-1,039432E-03
273	1,874546E-03	1,710993E-03	-3,435623E-03	-3,399981E-03
274	2,007645E-03	1,910987E-03	-6,493697E-04	-6,552861E-04
275	5,143477E-04	4,346942E-04	-1,742466E-03	-1,722951E-03
276	4,830332E-04	4,111657E-04	-1,552423E-03	-1,543728E-03
277	1,808422E-03	1,722299E-03	-3,660455E-04	-3,707884E-04
278	1,116310E-04	9,630988E-05	-1,873592E-03	-1,865093E-03
279	2,495758E-04	2,134262E-04	-1,411012E-03	-1,406628E-03
280	3,147347E-04	2,690828E-04	-1,386483E-03	-1,382983E-03
281	3,771337E-04	3,216489E-04	-1,367449E-03	-1,364402E-03
282	1,857925E-04	1,588785E-04	-1,445584E-03	-1,440245E-03
283	3,471821E-04	3,295473E-04	3,267413E-04	3,146078E-04
284	6,429950E-04	6,127168E-04	2,389923E-04	2,306897E-04
285	9,073850E-04	8,643778E-04	2,062907E-04	1,985961E-04

286	1,152970E-03	1,098303E-03	1,408560E-04	1,341725E-04
287	1,417774E-03	1,351043E-03	1,598302E-05	1,068708E-05
288	4,548758E-04	3,898054E-04	-1,474331E-03	-1,470063E-03
289	1,610463E-03	1,533382E-03	-1,557573E-04	-1,603186E-04
290	3,925770E-04	3,736640E-04	2,758056E-04	2,660737E-04
291	1,167945E-04	9,975976E-05	-1,459619E-03	-1,453577E-03
292	7,841927E-05	6,430049E-05	-6,204571E-04	-6,182153E-04
293	1,418564E-04	1,195222E-04	-1,141195E-03	-1,137098E-03
294	2,171810E-03	2,068983E-03	-7,665781E-04	-7,720132E-04
295	4,772149E-04	3,950869E-04	-1,688412E-03	-1,662278E-03
296	4,727361E-04	3,840524E-04	-1,537067E-03	-1,505736E-03
297	8,109499E-04	6,911032E-04	-1,974304E-03	-1,926169E-03
298	1,286637E-03	1,137584E-03	-2,562366E-03	-2,508901E-03
299	2,070938E-03	1,913439E-03	-2,071837E-03	-2,054341E-03
300	2,447877E-03	2,311093E-03	-1,453587E-03	-1,453083E-03
301	2,378454E-03	2,265227E-03	-8,644225E-04	-8,683819E-04
302	9,492876E-04	8,197248E-04	-2,088183E-03	-2,038248E-03
303	2,304459E-03	2,155786E-03	-2,112801E-03	-2,105524E-03
304	1,933046E-03	1,773880E-03	-2,102836E-03	-2,080021E-03
305	5,667797E-04	4,680644E-04	-1,700526E-03	-1,661745E-03
306	1,462852E-03	1,308778E-03	-2,344016E-03	-2,301246E-03
307	2,461077E-03	2,331527E-03	-1,277043E-03	-1,278685E-03
308	1,083953E-03	1,036071E-03	1,308383E-04	1,257260E-04
309	2,187442E-03	2,033004E-03	-2,093465E-03	-2,080555E-03
310	6,794746E-04	5,702832E-04	-1,844797E-03	-1,800367E-03
311	1,099084E-03	9,602687E-04	-2,280734E-03	-2,228572E-03
312	1,620934E-03	1,463692E-03	-2,191538E-03	-2,156761E-03
313	2,408350E-03	2,265843E-03	-1,568382E-03	-1,565528E-03
314	2,438940E-03	2,317323E-03	-1,072719E-03	-1,075852E-03
315	1,779412E-03	1,620316E-03	-2,137985E-03	-2,109342E-03
316	1,336098E-03	1,277465E-03	6,368585E-05	5,939443E-05
317	1,563505E-03	1,495369E-03	-3,035487E-05	-3,391417E-05
318	2,283292E-03	2,178508E-03	-6,700237E-04	-6,742037E-04
319	4,036357E-04	3,236753E-04	-1,376530E-03	-1,353188E-03
320	2,673054E-04	2,212570E-04	-1,001876E-03	-9,999652E-04
321	2,945643E-04	2,416106E-04	-9,995874E-04	-9,973588E-04
322	3,465971E-04	2,797862E-04	-1,187027E-03	-1,177657E-03
323	1,759176E-03	1,681818E-03	-1,588368E-04	-1,623355E-04
324	3,313996E-04	2,712519E-04	-1,111903E-03	-1,107669E-03
325	1,979829E-03	1,892275E-03	-3,398514E-04	-3,436043E-04
326	2,287788E-04	1,907557E-04	-1,027860E-03	-1,025433E-03
327	8,379461E-04	8,008363E-04	1,662372E-04	1,605571E-04
328	1,835445E-04	1,538929E-04	-1,070863E-03	-1,067629E-03
329	6,297434E-04	6,019593E-04	1,750143E-04	1,692056E-04
330	1,571100E-03	1,411638E-03	-2,841012E-03	-2,793650E-03
331	1,698344E-03	1,536968E-03	-2,829317E-03	-2,788290E-03
332	1,840271E-03	1,678468E-03	-2,799569E-03	-2,766181E-03
333	1,972864E-03	1,812444E-03	-2,727876E-03	-2,702069E-03

334	6,661687E-05	5,380523E-05	-2,860120E-04	-2,849967E-04
335	2,078679E-03	1,920755E-03	-2,627396E-03	-2,607766E-03
336	3,667724E-04	2,936176E-04	-1,276267E-03	-1,259953E-03
337	5,359913E-04	4,459462E-04	-1,899974E-03	-1,863833E-03
338	3,639813E-04	3,055544E-04	-1,233649E-03	-1,230494E-03
339	6,165277E-04	5,168032E-04	-2,127691E-03	-2,080875E-03
340	7,116497E-04	6,017045E-04	-2,332607E-03	-2,276964E-03
341	8,215100E-04	7,011961E-04	-2,513098E-03	-2,451233E-03
342	9,384377E-04	8,085504E-04	-2,646508E-03	-2,581752E-03
343	1,045540E-03	9,081596E-04	-2,722677E-03	-2,657912E-03
344	1,591906E-03	1,519540E-03	-8,371891E-05	-8,781676E-05
345	2,152844E-03	2,056244E-03	-5,118513E-04	-5,160073E-04
346	2,263919E-03	2,153513E-03	-9,984510E-04	-1,003998E-03
347	2,338642E-03	2,220028E-03	-1,270974E-03	-1,275882E-03
348	2,381838E-03	2,255253E-03	-1,545894E-03	-1,549317E-03
349	2,390865E-03	2,256761E-03	-1,798438E-03	-1,799488E-03
350	2,366672E-03	2,225544E-03	-2,007103E-03	-2,004827E-03
351	1,399935E-03	1,251549E-03	-5,816444E-03	-5,782745E-03
352	1,161628E-03	1,027626E-03	-6,221073E-03	-6,180732E-03
353	1,289465E-03	1,145844E-03	-6,086634E-03	-6,046787E-03
354	1,562357E-03	1,409411E-03	-5,451509E-03	-5,422170E-03
355	1,578699E-03	1,413701E-03	-4,467689E-03	-4,352671E-03
356	1,481270E-03	1,310299E-03	-5,300604E-03	-5,167456E-03
357	1,454103E-03	1,291992E-03	-4,679186E-03	-4,556259E-03
358	1,504476E-03	1,335564E-03	-5,088212E-03	-4,953569E-03
359	1,490690E-03	1,317446E-03	-5,461369E-03	-5,335642E-03
360	1,763246E-03	1,611899E-03	-4,705763E-03	-4,690090E-03
361	1,757590E-03	1,616452E-03	-3,888980E-03	-3,888578E-03
362	1,714008E-03	1,577900E-03	-3,402830E-03	-3,407423E-03
363	1,639029E-03	1,507966E-03	-2,889917E-03	-2,897877E-03
364	1,538551E-03	1,412493E-03	-2,320317E-03	-2,330355E-03
365	1,768696E-03	1,622395E-03	-4,329580E-03	-4,322483E-03
366	1,277947E-03	1,139239E-03	-4,475358E-03	-4,473304E-03
367	1,409462E-03	1,272316E-03	-3,870434E-03	-3,873626E-03
368	1,120291E-03	9,830309E-04	-5,424323E-03	-5,408359E-03
369	1,398084E-03	1,264355E-03	-3,284384E-03	-3,291395E-03
370	1,197221E-03	1,058191E-03	-5,017305E-03	-5,008308E-03
371	1,715766E-03	1,537668E-03	-5,003087E-03	-4,865981E-03
372	1,556504E-03	1,398472E-03	-4,007678E-03	-3,912853E-03
373	1,763017E-03	1,577189E-03	-5,811681E-03	-5,648605E-03
374	1,718355E-03	1,535245E-03	-6,138231E-03	-5,973155E-03
375	1,766489E-03	1,582109E-03	-5,434618E-03	-5,281815E-03
376	1,635010E-03	1,459729E-03	-6,392153E-03	-6,237587E-03
377	1,554998E-03	1,379128E-03	-6,010949E-03	-5,894035E-03
378	1,515993E-03	1,354957E-03	-6,555228E-03	-6,430299E-03
379	1,553882E-03	1,380577E-03	-6,343561E-03	-6,221414E-03
380	1,732118E-03	1,576158E-03	-5,009133E-03	-4,983472E-03
381	1,377608E-03	1,232644E-03	-5,428186E-03	-5,408365E-03

382	1,524395E-03	1,349882E-03	-5,570662E-03	-5,457898E-03
383	1,602839E-03	1,423912E-03	-5,814932E-03	-5,668832E-03
384	1,522210E-03	1,377268E-03	-4,837248E-03	-4,827032E-03
385	1,556161E-03	1,414409E-03	-4,313777E-03	-4,310885E-03
386	1,178093E-03	1,039715E-03	-5,855433E-03	-5,828337E-03
387	1,601848E-03	1,423071E-03	-6,145379E-03	-6,003953E-03
388	1,554968E-03	1,378248E-03	-5,836759E-03	-5,707094E-03
389	1,620341E-03	1,442368E-03	-5,528883E-03	-5,381371E-03
390	1,597251E-03	1,446985E-03	-5,122637E-03	-5,102815E-03
391	1,533336E-03	1,371430E-03	-5,926486E-03	-5,866363E-03
392	1,362544E-03	1,214507E-03	-6,446361E-03	-6,375244E-03
393	1,437416E-03	1,280842E-03	-6,259387E-03	-6,192820E-03
394	1,517583E-03	1,352142E-03	-6,114857E-03	-6,036908E-03

Рис.1. Абсолютная разность решения в Abaqus и Python

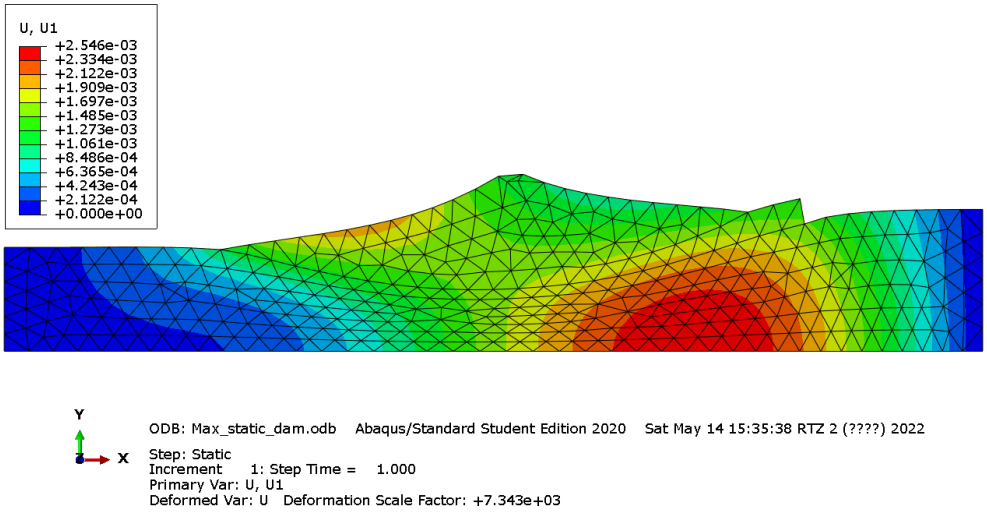

Рис.1. U1 Abaqus

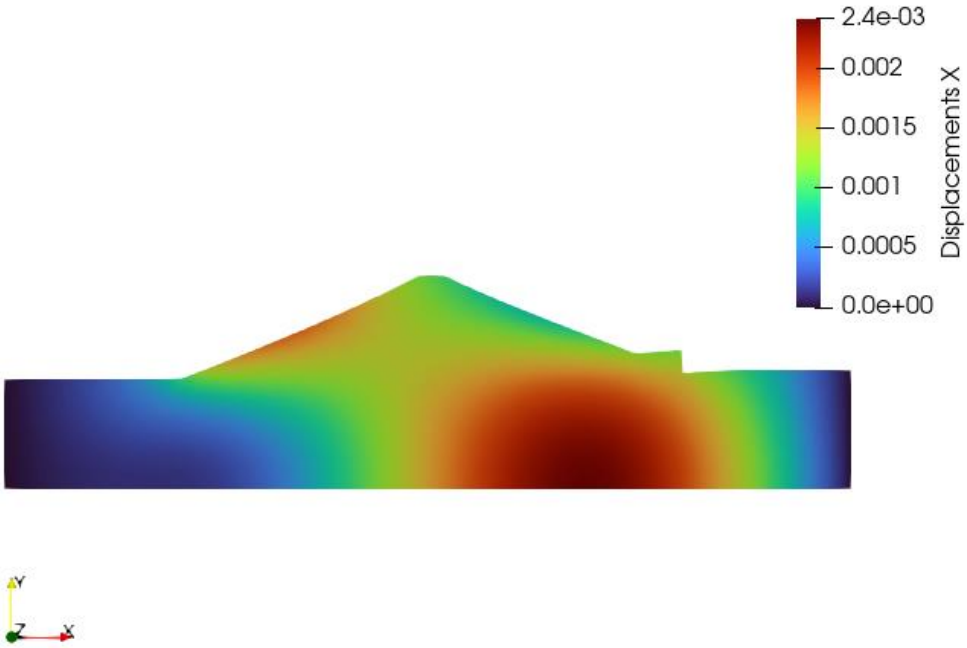

Рис.1. U1 Python

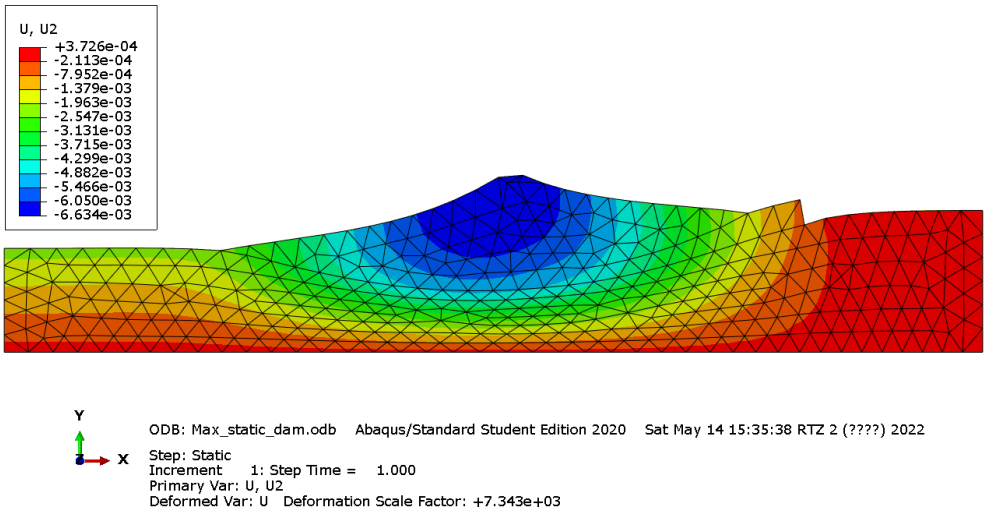

Рис.1. U2 Abaqus

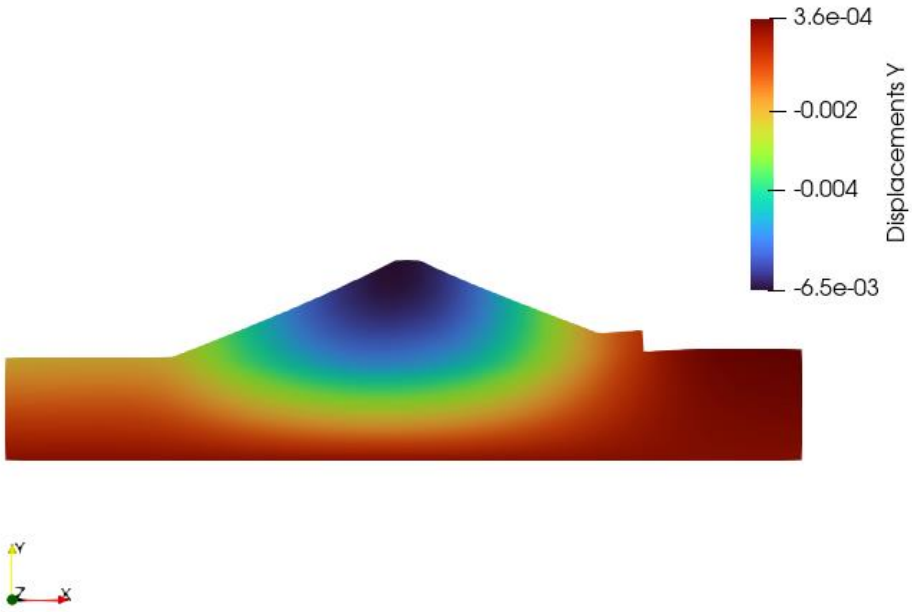

Рис.1. U2 Python

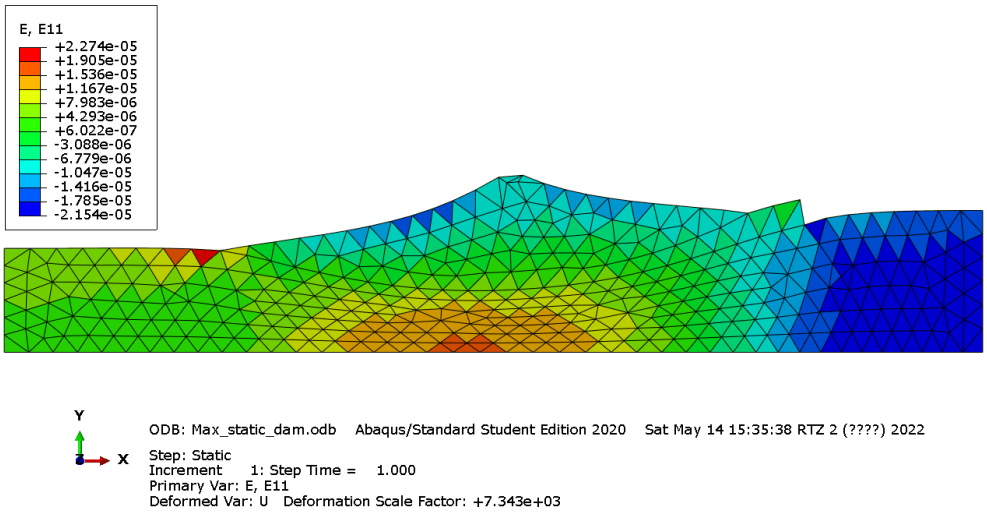

Рис.1. E11 Abaqus

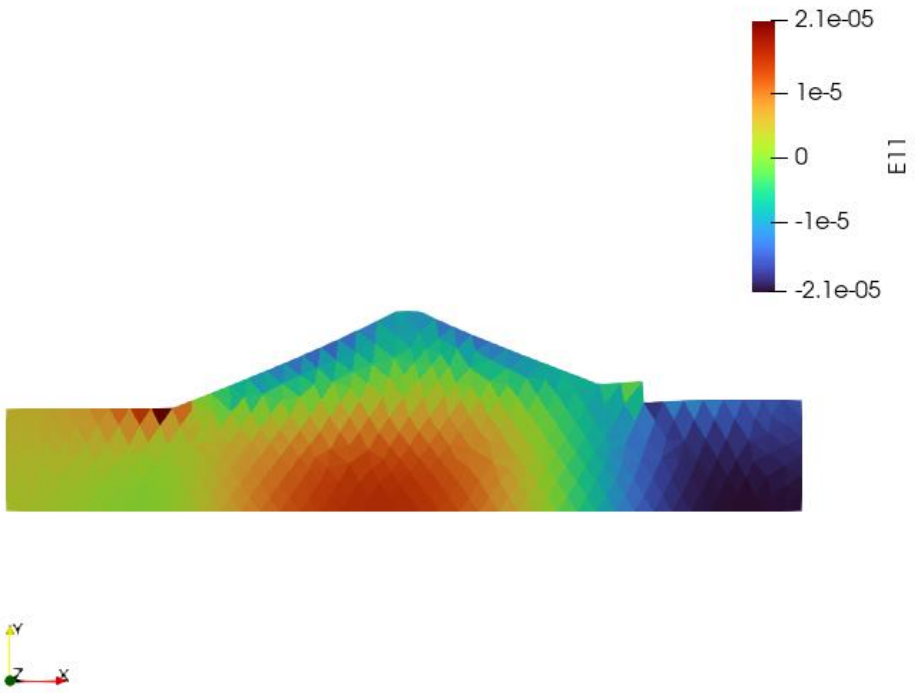

Рис.1. E11 Python

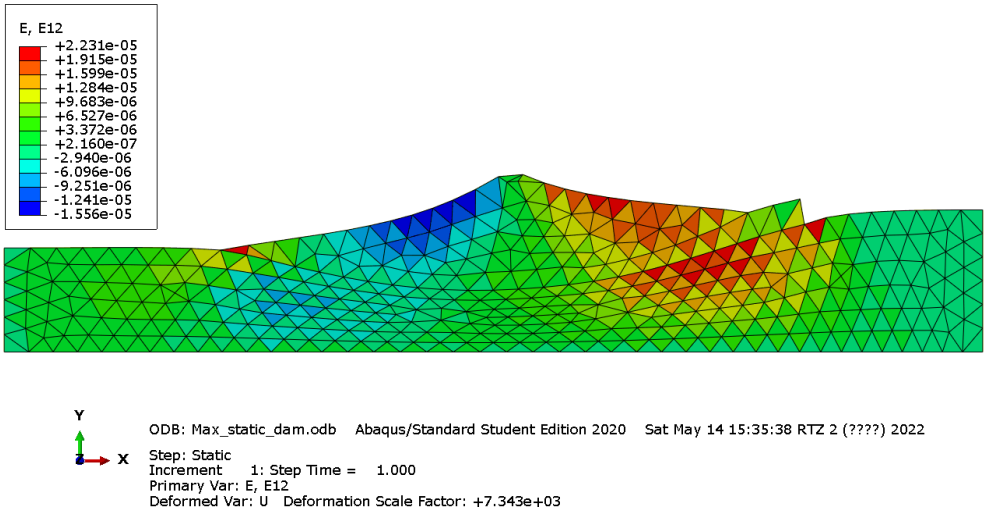

Рис.1. E12 Abaqus

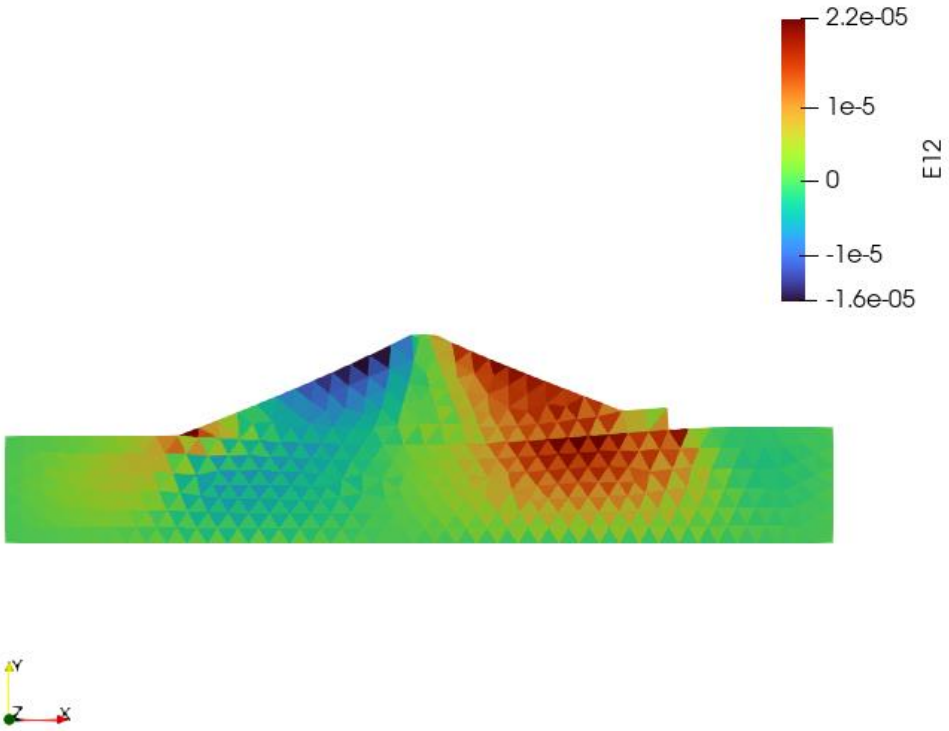

Рис.1. E12 Python

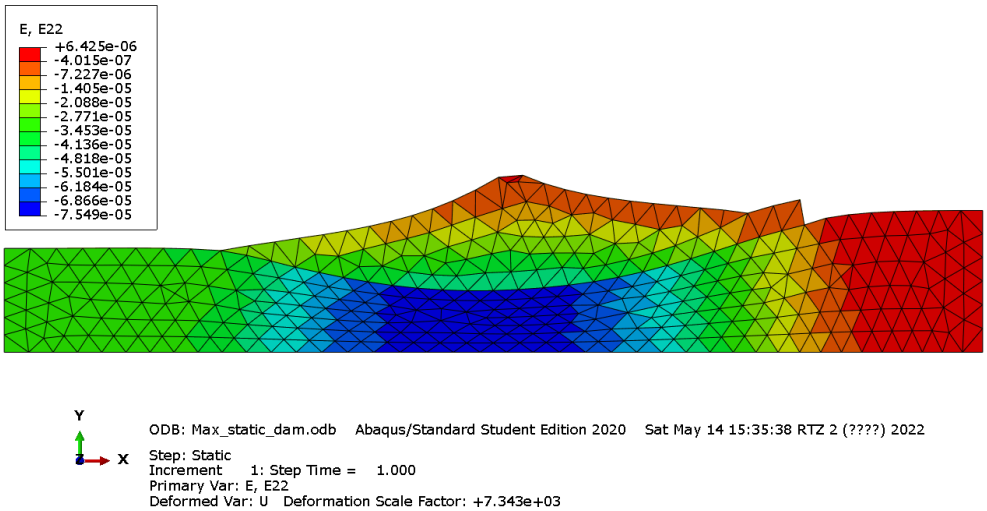

Рис.1. E22 Abaqus

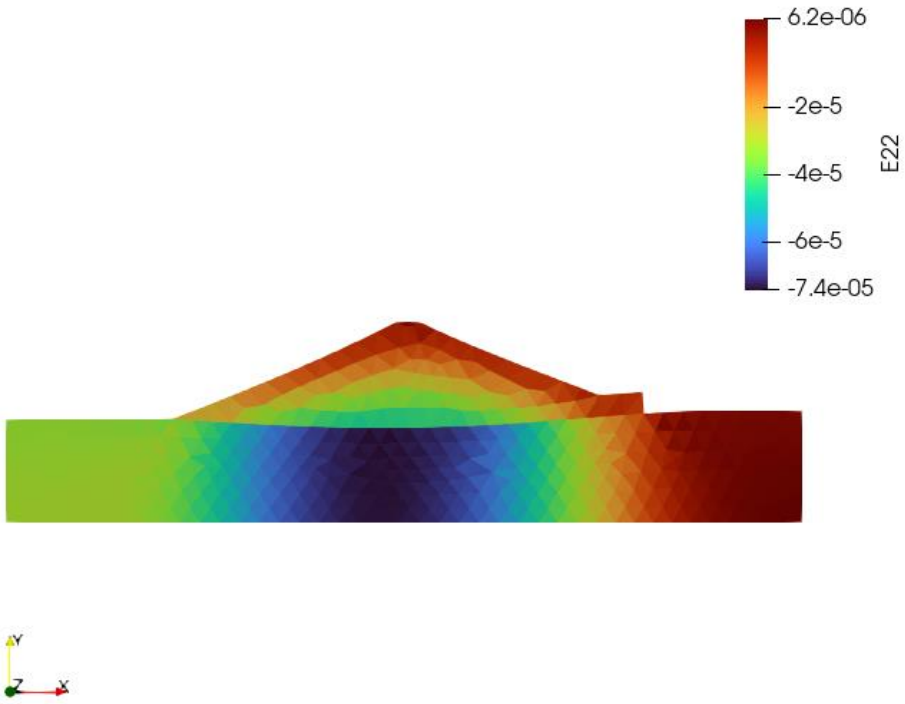

Рис.1. E22 Python

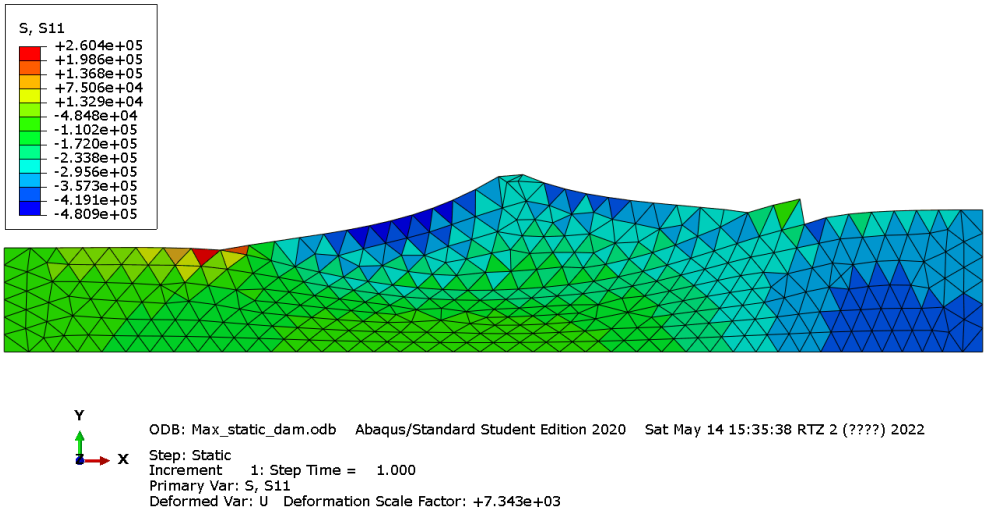

Рис.1. S11 Abaqus

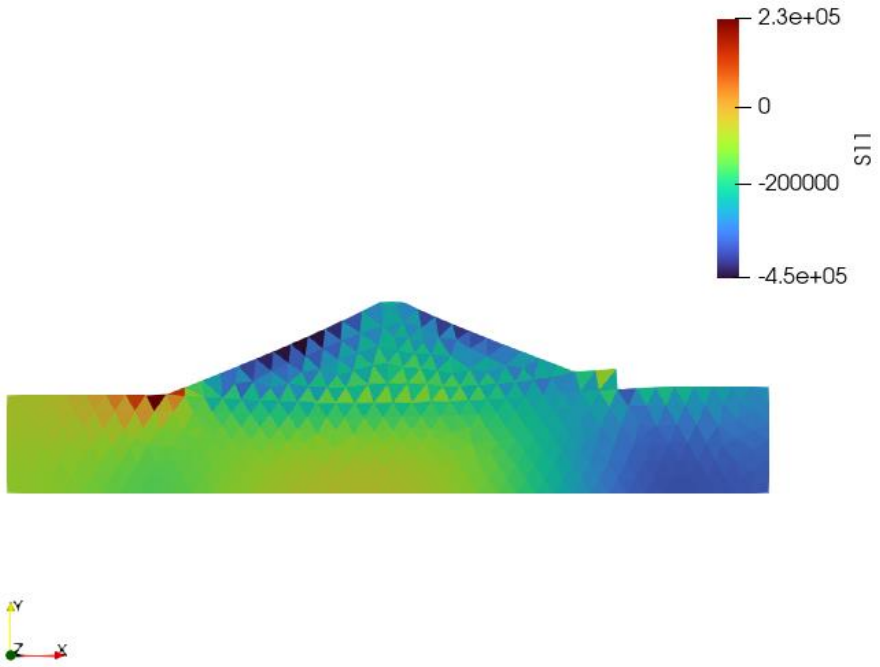

Рис.1. S11 Python

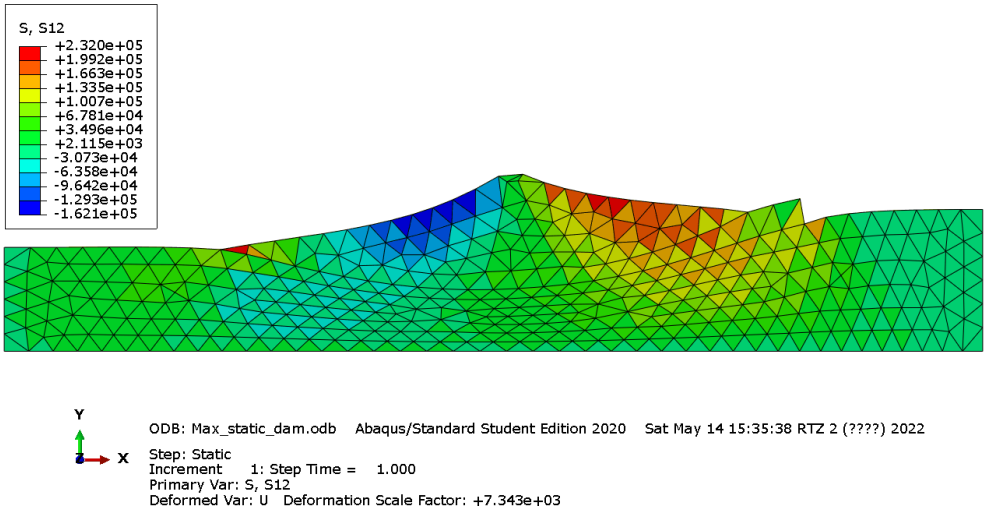

Рис.1. S12 Abaqus

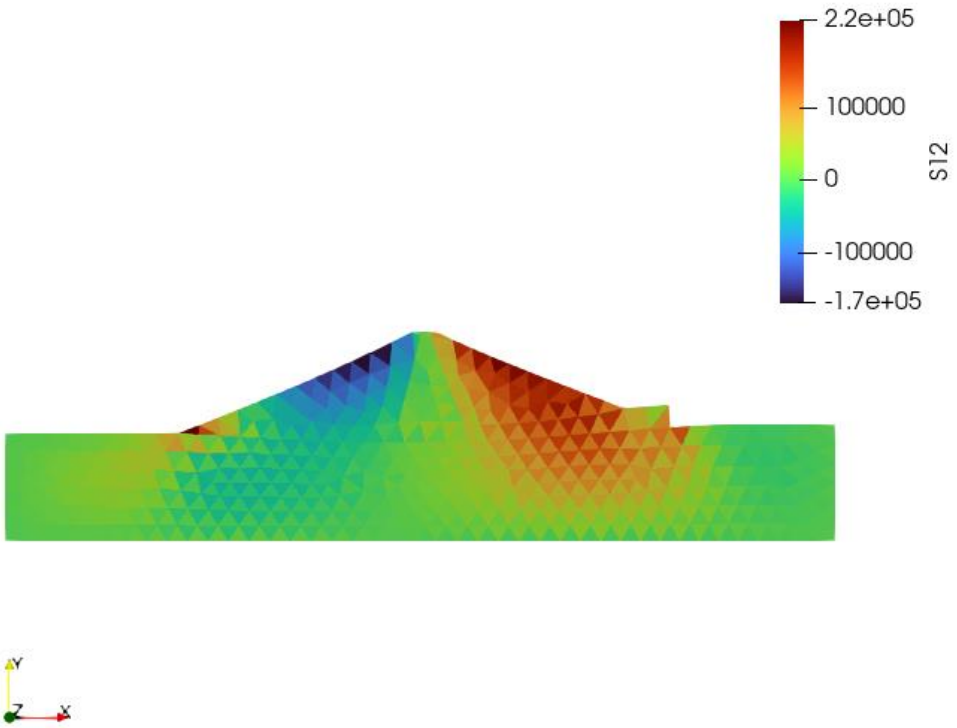

Рис.1. S12 Python

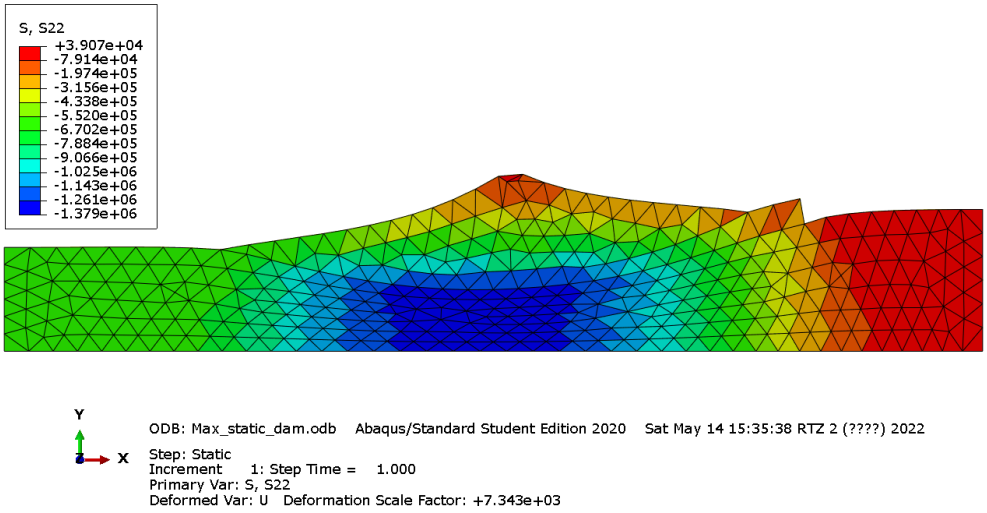

Рис.1. S22 Abaqus

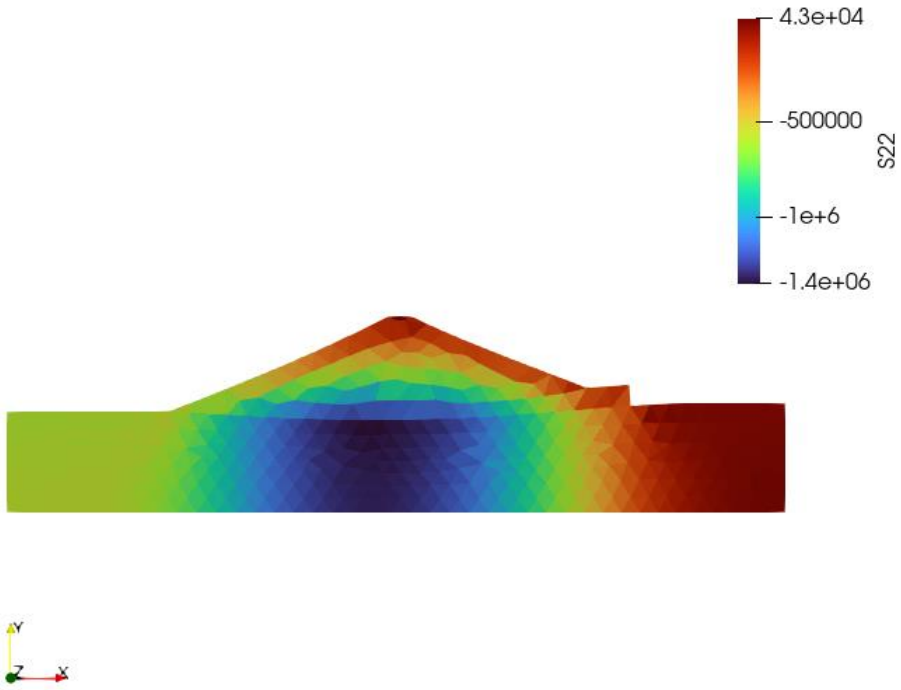

Рис.1. S22 Python

4. Заключение

В работе произведен расчет плоско-деформированного состояния бетонной плотины на каменном основании под воздействием силы тяжести и давления воды на стену плотины. Решение производилось 2 методами – в КЭ пакете SIMULIA Abaqus и в самописной программе на ЯП Python. Решения крайне близки друг другу и их разность мала относительно порядка вычисляемых величин.

5. Приложение 1. Код программы

```
class dam_static:
    def __init__(self, x, l, Mass_node, Mass_Element, Mass_Element_Priming,
Mass_Element_C, Mass_Element_C2, E_Priming,
                 nu_Priming, E_C,
                 nu_C, E_C2, nu_C2, Mass_node_B_X, Mass_node_B_Y):
        print("__init__ termal")
        self.x = x
        self.l = l
        self.E_Priming = E_Priming
        self.E_C = E_C
        self.E_C2 = E_C2
        self.nu_Priming = nu_Priming
        self.nu_C = nu_C
        self.nu_C2 = nu_C2
        # self.rho = rho
        self.g = 9.8
        self.B_X = -1 * self.g
        # self.P = P
        # self.T_air = T_air
        # self.T_water = T_water
        # self.Lambda_Concrete = Lambda_Concrete
        # self.Lambda_Priming = Lambda_Priming

        self.Mass_node_B_X = Mass_node_B_X
        self.Mass_node_B_Y = Mass_node_B_Y

        self.Mass_node = Mass_node
        self.Mass_Element = Mass_Element
        self.Mass_Element_Priming = Mass_Element_Priming
        self.Mass_Element_C = Mass_Element_C
        self.Mass_Element_C2 = Mass_Element_C2
        # self.Mass_node_T_air = Mass_node_T_air
        # self.Mass_node_T_water = Mass_node_T_water
        self.sym_eta = sym.Symbol('x')
        self.sym_ksi = sym.Symbol('y')
        self.sym_N_i = 1 - self.sym_ksi - self.sym_eta
        self.sym_N_j = self.sym_eta
        self.sym_N_k = self.sym_ksi
        self.B = np.matrix([[-1, 0, 0, 0, 1, 0],
                            [-1, 0, 1, 0, 0, 0],
                            [-1, -1, 1, 0, 0, 1]])

        self.B_t = np.matrix([[-1, 0, 1],
```

```
                    [-1, 1, 0]])

        # self.D = (self.E * (1 - self.nu) / ((1 - 2 * self.nu) * (1 + self.nu))) *
np.matrix(
        #     [[1, self.nu / (1 - self.nu), 0], [self.nu / (1 - self.nu), 1, 0],
        #      [0, 0, (1 - 2 * self.nu) / (2 * (1 - self.nu))]])
        # self.k_e = self.CreateMatrix_k_e()
        # self.M_e = self.CreateMatrix_m_e()

    def get_global(self, ind, Matrix_e):
        print('Matrix_e', Matrix_e)  # проверить ее размер
        print(len(Matrix_e))
        K = np.zeros((len(self.Mass_node) * 2, len(self.Mass_node) * 2))
        print(len(K))

        # for i in range(len(Matrix_e)):
        #     for j in range(len(Matrix_e) - 1):
        #         if (i % 2 == 0):
        #             firtst = ind[math.floor(i / 2)] * 2
        #         else:
        #             firtst = (ind[math.floor(i / 2)] * 2) + 1
        #         if (j % 2 == 0):
        #             twice = ind[math.floor(j / 2)] * 2
        #         else:
        #             twice = (ind[math.floor(j / 2)] * 2) + 1
        #         K[firtst, twice] = Matrix_e[i, j]

        K[ind[0] * 2, ind[0] * 2] = Matrix_e[0, 0]
        K[(ind[0] * 2) + 1, (ind[0] * 2) + 1] = Matrix_e[0 + 1, 0 + 1]
        K[(ind[0] * 2), (ind[0] * 2) + 1] = Matrix_e[0, 0 + 1]
        K[(ind[0] * 2) + 1, (ind[0] * 2)] = Matrix_e[0 + 1, 0]

        K[ind[0] * 2, ind[1] * 2] = Matrix_e[0, 1 * 2]
        K[(ind[0] * 2) + 1, (ind[1] * 2) + 1] = Matrix_e[0 + 1, 1 * 2 + 1]
        K[(ind[0] * 2), (ind[1] * 2) + 1] = Matrix_e[0, 1 * 2 + 1]
        K[(ind[0] * 2) + 1, (ind[1] * 2)] = Matrix_e[0 + 1, 1 * 2]

        K[ind[1] * 2, ind[0] * 2] = Matrix_e[1 * 2, 0]
        K[(ind[1] * 2) + 1, (ind[0] * 2) + 1] = Matrix_e[1 * 2 + 1, 0 + 1]
        K[(ind[1] * 2), (ind[0] * 2) + 1] = Matrix_e[1 * 2, 0 + 1]
        K[(ind[1] * 2) + 1, (ind[0] * 2)] = Matrix_e[1 * 2 + 1, 0]

        K[ind[1] * 2, ind[2] * 2] = Matrix_e[1 * 2, 2 * 2]
        K[(ind[1] * 2) + 1, (ind[2] * 2) + 1] = Matrix_e[1 * 2 + 1, 2 * 2 + 1]
        K[(ind[1] * 2), (ind[2] * 2) + 1] = Matrix_e[1 * 2, 2 * 2 + 1]
```

```
        K[(ind[1] * 2) + 1, (ind[2] * 2)] = Matrix_e[1 * 2 + 1, 2 * 2]

        K[ind[2] * 2, ind[1] * 2] = Matrix_e[2 * 2, 1 * 2]
        K[(ind[2] * 2) + 1, (ind[1] * 2) + 1] = Matrix_e[2 * 2 + 1, 1 * 2 + 1]
        K[(ind[2] * 2), (ind[1] * 2) + 1] = Matrix_e[2 * 2, 1 * 2 + 1]
        K[(ind[2] * 2) + 1, (ind[1] * 2)] = Matrix_e[2 * 2 + 1, 1 * 2]

        K[ind[0] * 2, ind[2] * 2] = Matrix_e[0, 2 * 2]
        K[(ind[0] * 2) + 1, (ind[2] * 2) + 1] = Matrix_e[0 + 1, 2 * 2 + 1]
        K[(ind[0] * 2), (ind[2] * 2) + 1] = Matrix_e[0, 2 * 2 + 1]
        K[(ind[0] * 2) + 1, (ind[2] * 2)] = Matrix_e[0 + 1, 2 * 2]

        K[ind[2] * 2, ind[0] * 2] = Matrix_e[2 * 2, 0]
        K[(ind[2] * 2) + 1, (ind[0] * 2) + 1] = Matrix_e[2 * 2 + 1, 0 + 1]
        K[(ind[2] * 2), (ind[0] * 2) + 1] = Matrix_e[2 * 2, 0 + 1]
        K[(ind[2] * 2) + 1, (ind[0] * 2)] = Matrix_e[2 * 2 + 1, 0]

        K[ind[1] * 2, ind[1] * 2] = Matrix_e[1 * 2, 1 * 2]
        K[(ind[1] * 2) + 1, (ind[1] * 2) + 1] = Matrix_e[1 * 2 + 1, 1 * 2 + 1]
        K[(ind[1] * 2), (ind[1] * 2) + 1] = Matrix_e[1 * 2, 1 * 2 + 1]
        K[(ind[1] * 2) + 1, (ind[1] * 2)] = Matrix_e[1 * 2 + 1, 1 * 2]

        K[ind[2] * 2, ind[2] * 2] = Matrix_e[2 * 2, 2 * 2]
        K[(ind[2] * 2) + 1, (ind[2] * 2) + 1] = Matrix_e[2 * 2 + 1, 2 * 2 + 1]
        K[(ind[2] * 2), (ind[2] * 2) + 1] = Matrix_e[2 * 2, 2 * 2 + 1]
        K[(ind[2] * 2) + 1, (ind[2] * 2)] = Matrix_e[2 * 2 + 1, 2 * 2]

        # K[ind[0], ind[1]] = Matrix_e[0, 1]
        # K[ind[1], ind[0]] = Matrix_e[1, 0]
        # K[ind[1], ind[2]] = Matrix_e[1, 2]
        # K[ind[2], ind[1]] = Matrix_e[2, 1]
        # K[ind[0], ind[2]] = Matrix_e[0, 2]
        # K[ind[2], ind[0]] = Matrix_e[2, 0]
        # K[ind[1], ind[1]] = Matrix_e[1, 1]
        # K[ind[2], ind[2]] = Matrix_e[2, 2]

        return K
    def get_J(self, coords):
        Coord = np.matrix([[coords[0, 0], coords[0, 1]], [coords[1, 0], coords[1, 1]],
[coords[2, 0], coords[2, 1]]])
        J = self.B_t * Coord
        return J
    def get_stiffness_matrix(self, coords, D):
        Coord = np.matrix([[coords[0, 0], coords[0, 1]], [coords[1, 0], coords[1, 1]],
[coords[2, 0], coords[2, 1]]])
```

```
        print('Coord', Coord)
        print('self.B', self.B)
        J = self.B_t * Coord  # разобраться какой B использовать
        print('J', J)

        # print(self.B)
        Res_B = np.zeros((3, 6))
        # new_B = np.matrix([[-1, -1], [0, 1], [1, 0]])

        for i in range(3):
            print('i', i)
            vec_B = np.array([self.B_t[0, i], self.B_t[1, i]])
            # print('vec_B', vec_B)
            B = solve(J, vec_B)

            Res_B[0, i * 2] = B[0]
            Res_B[1, (i * 2) + 1] = B[1]
            Res_B[2, i * 2] = B[1]
            Res_B[2, (i * 2) + 1] = B[0]
            # print('B', B)
        print(''Res_B'', Res_B)
        print(np.dot(Res_B, Res_B.T))
        Ki = np.dot(np.dot(Res_B.T, D), Res_B) * np.linalg.det(J) / 2

        # print('Ki', Res_Ki)
        print('Ki', Ki)
        return Ki

    def Solve(self):  # , GU, time, dt
        print('solve')
        N = len(self.Mass_node)
        # print(N)
        K = np.zeros((N * 2, N * 2))
        D_Prinmig = (self.E_Priming * (1 - self.nu_Priming) / (
                (1 - 2 * self.nu_Priming) * (1 + self.nu_Priming))) * np.matrix(
            [[1, self.nu_Priming / (1 - self.nu_Priming), 0],
             [self.nu_Priming / (1 - self.nu_Priming), 1, 0],
             [0, 0, (1 - 2 * self.nu_Priming) / (2 * (1 - self.nu_Priming))]
             ])
        D_C = (self.E_C * (1 - self.nu_C) / (
                (1 - 2 * self.nu_C) * (1 + self.nu_C))) * np.matrix(
            [[1, self.nu_C / (1 - self.nu_C), 0], [self.nu_C / (1 - self.nu_C), 1, 0],
             [0, 0, (1 - 2 * self.nu_C) / (2 * (1 - self.nu_C))]])
        D_C2 = (self.E_C2 * (1 - self.nu_C2) / (
                (1 - 2 * self.nu_C2) * (1 + self.nu_C2))) * np.matrix(
```

```
            [[1, self.nu_C2 / (1 - self.nu_C2), 0], [self.nu_C2 / (1 - self.nu_C2), 1, 0],
             [0, 0, (1 - 2 * self.nu_C2) / (2 * (1 - self.nu_C2))]])

        for i in range(len(self.Mass_Element)):
            el = self.Mass_Element[i]
            # print('rrrrrr', i, el)
            Enodes = np.matrix([self.Mass_node[el[0]], self.Mass_node[el[1]],
self.Mass_node[el[2]]])
            print(Enodes)

            if i in self.Mass_Element_Priming:
                # print("Lambda_Priming")
                D = D_Prinmig
            elif i in self.Mass_Element_C:
                D = D_C
            else:
                D = D_C2
                # print('Lambda_Concrete')

            Ki = self.get_stiffness_matrix(Enodes, D)
            Ki = self.get_global(self.Mass_Element[i], Ki)
            K = K + Ki
        F = np.zeros((N * 2, 1))

        def p_hydro(y):
            return 9800 * (115.171875 - y)

        def S_trapeze(a, b, h):
            return 1/2*(a + b)*h


        def distance(coord_1, coord_2):
            return float(math.sqrt((coord_1[0] - coord_2[0]) ** 2 + (coord_1[1] -
coord_2[1]) ** 2))



        # Гу x
        for i in range(len(self.Mass_node_B_X)):
            print('ii', i)
            K[self.Mass_node_B_X[i] * 2, :] = 0
            # K[(self.Mass_node_B_X[i] * 2) + 1, :] = 0
            # K[:,self.Mass_node_T_air[i]] = 0
            K[self.Mass_node_B_X[i] * 2, self.Mass_node_B_X[i] * 2] = 1
```

```
            # K[(self.Mass_node_B_X[i] * 2) + 1, (self.Mass_node_B_X[i] * 2) + 1] =
1
            F[self.Mass_node_B_X[i] * 2] = 0
        # ГУ y
        for i, index in enumerate(range(len(self.Mass_node_B_Y))):
            print('ii', i)
            print('index', index)
            # K[self.Mass_node_B_Y[i] * 2, :] = 0
            K[(self.Mass_node_B_Y[i] * 2) + 1, :] = 0
            # K[:,self.Mass_node_T_air[i]] = 0
            # K[self.Mass_node_B_Y[i] * 2, self.Mass_node_B_Y[i] * 2] = 1
            K[(self.Mass_node_B_Y[i] * 2) + 1, (self.Mass_node_B_Y[i] * 2) + 1] = 1
            F[(self.Mass_node_B_Y[i] * 2) + 1] = 0
        nodes = self.Mass_node
        # # давление на горизонтальное верхнее
        # F[2 * 12] += S_trapeze(p_hydro(nodes[12][1]), p_hydro(1 / 2 *
(nodes[12][1] + nodes[9][1])),
        #                        1 / 2 * (nodes[12][1] - nodes[9][1]))
        # F[2 * 9] += S_trapeze(p_hydro(nodes[9][1]), p_hydro(1 / 2 * (nodes[9][1] +
nodes[12][1])),
        #                       1 / 2 * (nodes[12][1] - nodes[9][1]))
        # давление на наклонное чудо
        F[2 * 13] += S_trapeze(p_hydro(nodes[13][1]), p_hydro(1 / 2 * (nodes[13][1]
+ nodes[135][1])),
                               1 / 2 * distance(nodes[13], nodes[135])) * 0.995
        F[2 * 13+1] += -S_trapeze(p_hydro(nodes[13][1]), p_hydro(1 / 2 *
(nodes[13][1] + nodes[135][1])),
                                1 / 2 * distance(nodes[13], nodes[135])) * 1.395
        F[2*135] += S_trapeze(p_hydro(nodes[135][1]), p_hydro(1/2*(nodes[135][1]
+ nodes[13][1])), 1/2*distance(nodes[13], nodes[135]))*0.995
        F[2 * 135+1] += -S_trapeze(p_hydro(nodes[135][1]), p_hydro(1 / 2 *
(nodes[135][1] + nodes[13][1])),
                               1 / 2 * distance(nodes[135], nodes[13])) * 1.395
        F[2 * 135] += S_trapeze(p_hydro(nodes[135][1]), p_hydro(1 / 2 *
(nodes[135][1] + nodes[136][1])),
                               1 / 2 * distance(nodes[136], nodes[135])) * 0.995
        F[2 * 135 + 1] += -S_trapeze(p_hydro(nodes[135][1]), p_hydro(1 / 2 *
(nodes[135][1] + nodes[136][1])),
                                 1 / 2 * distance(nodes[135], nodes[136])) * 1.395

        F[2 * 136] += S_trapeze(p_hydro(nodes[136][1]), p_hydro(1 / 2 *
(nodes[136][1] + nodes[135][1])),
                               1 / 2 * distance(nodes[136], nodes[135])) * 0.995
        F[2 * 136+1] += -S_trapeze(p_hydro(nodes[136][1]), p_hydro(1 / 2 *
(nodes[136][1] + nodes[135][1])),
```

```
                    1 / 2 * distance(nodes[136], nodes[135])) * 1.395
    F[2 * 136] += S_trapeze(p_hydro(nodes[136][1]), p_hydro(1 / 2 *
(nodes[136][1] + nodes[137][1])),
                    1 / 2 * distance(nodes[136], nodes[137])) * 0.995
    F[2 * 136+1] += -S_trapeze(p_hydro(nodes[136][1]), p_hydro(1 / 2 *
(nodes[136][1] + nodes[137][1])),
                    1 / 2 * distance(nodes[136], nodes[137])) * 1.395

    F[2 * 137] += S_trapeze(p_hydro(nodes[137][1]), p_hydro(1 / 2 *
(nodes[136][1] + nodes[137][1])),
                    1 / 2 * distance(nodes[136], nodes[137])) * 0.995
    F[2 * 137+1] += -S_trapeze(p_hydro(nodes[137][1]), p_hydro(1 / 2 *
(nodes[136][1] + nodes[137][1])),
                    1 / 2 * distance(nodes[136], nodes[137])) * 1.395
    F[2 * 137] += S_trapeze(p_hydro(nodes[137][1]), p_hydro(1 / 2 *
(nodes[138][1] + nodes[137][1])),
                    1 / 2 * distance(nodes[138], nodes[137])) * 0.995
    F[2 * 137+1] += -S_trapeze(p_hydro(nodes[137][1]), p_hydro(1 / 2 *
(nodes[138][1] + nodes[137][1])),
                    1 / 2 * distance(nodes[138], nodes[137])) * 1.395

    F[2 * 138] += S_trapeze(p_hydro(nodes[138][1]), p_hydro(1 / 2 *
(nodes[138][1] + nodes[137][1])),
                    1 / 2 * distance(nodes[138], nodes[137])) * 0.995
    F[2 * 138 + 1] += -S_trapeze(p_hydro(nodes[138][1]), p_hydro(1 / 2 *
(nodes[138][1] + nodes[137][1])),
                      1 / 2 * distance(nodes[138], nodes[137])) * 1.395
    F[2 * 138] += S_trapeze(p_hydro(nodes[138][1]), p_hydro(1 / 2 *
(nodes[138][1] + nodes[139][1])),
                    1 / 2 * distance(nodes[138], nodes[139])) * 0.995
    F[2 * 138 + 1] += -S_trapeze(p_hydro(nodes[138][1]), p_hydro(1 / 2 *
(nodes[138][1] + nodes[139][1])),
                      1 / 2 * distance(nodes[138], nodes[139])) * 1.395

    F[2 * 139] += S_trapeze(p_hydro(nodes[139][1]), p_hydro(1 / 2 *
(nodes[138][1] + nodes[139][1])),
                    1 / 2 * distance(nodes[138], nodes[139])) * 0.995
    F[2 * 139 + 1] += -S_trapeze(p_hydro(nodes[139][1]), p_hydro(1 / 2 *
(nodes[138][1] + nodes[139][1])),
                      1 / 2 * distance(nodes[138], nodes[139])) * 1.395
    F[2 * 139] += S_trapeze(p_hydro(nodes[139][1]), p_hydro(1 / 2 *
(nodes[140][1] + nodes[139][1])),
                    1 / 2 * distance(nodes[140], nodes[139])) * 0.995
    F[2 * 139 + 1] += -S_trapeze(p_hydro(nodes[139][1]), p_hydro(1 / 2 *
(nodes[140][1] + nodes[139][1])),
```

```
                1 / 2 * distance(nodes[140], nodes[139])) * 1.395

    F[2 * 140] += S_trapeze(p_hydro(nodes[140][1]), p_hydro(1 / 2 *
(nodes[140][1] + nodes[139][1])),
                1 / 2 * distance(nodes[140], nodes[139])) * 0.995
    F[2 * 140 + 1] += -S_trapeze(p_hydro(nodes[140][1]), p_hydro(1 / 2 *
(nodes[140][1] + nodes[139][1])),
                1 / 2 * distance(nodes[140], nodes[139])) * 1.395
    F[2 * 140] += S_trapeze(p_hydro(nodes[140][1]), p_hydro(1 / 2 *
(nodes[140][1] + nodes[141][1])),
                1 / 2 * distance(nodes[140], nodes[141])) * 0.995
    F[2 * 140 + 1] += -S_trapeze(p_hydro(nodes[140][1]), p_hydro(1 / 2 *
(nodes[140][1] + nodes[141][1])),
                1 / 2 * distance(nodes[140], nodes[141])) * 1.395

    F[2 * 141] += S_trapeze(p_hydro(nodes[141][1]), p_hydro(1 / 2 *
(nodes[140][1] + nodes[141][1])),
                1 / 2 * distance(nodes[140], nodes[141])) * 0.995
    F[2 * 141 + 1] += -S_trapeze(p_hydro(nodes[141][1]), p_hydro(1 / 2 *
(nodes[140][1] + nodes[141][1])),
                1 / 2 * distance(nodes[140], nodes[141])) * 1.395
    F[2 * 141] += S_trapeze(p_hydro(nodes[141][1]), p_hydro(1 / 2 *
(nodes[142][1] + nodes[141][1])),
                1 / 2 * distance(nodes[142], nodes[141])) * 0.995
    F[2 * 141 + 1] += -S_trapeze(p_hydro(nodes[141][1]), p_hydro(1 / 2 *
(nodes[142][1] + nodes[141][1])),
                1 / 2 * distance(nodes[142], nodes[141])) * 1.395

    F[2 * 142] += S_trapeze(p_hydro(nodes[142][1]), p_hydro(1 / 2 *
(nodes[142][1] + nodes[141][1])),
                1 / 2 * distance(nodes[142], nodes[141])) * 0.995
    F[2 * 142 + 1] += -S_trapeze(p_hydro(nodes[142][1]), p_hydro(1 / 2 *
(nodes[142][1] + nodes[141][1])),
                1 / 2 * distance(nodes[142], nodes[141])) * 1.395
    F[2 * 142] += S_trapeze(p_hydro(nodes[142][1]), p_hydro(1 / 2 *
(nodes[142][1] + nodes[143][1])),
                1 / 2 * distance(nodes[142], nodes[143])) * 0.995
    F[2 * 142 + 1] += -S_trapeze(p_hydro(nodes[142][1]), p_hydro(1 / 2 *
(nodes[142][1] + nodes[143][1])),
                1 / 2 * distance(nodes[142], nodes[143])) * 1.395

    F[2 * 143] += S_trapeze(p_hydro(nodes[143][1]), p_hydro(1 / 2 *
(nodes[142][1] + nodes[143][1])),
                1 / 2 * distance(nodes[142], nodes[143])) * 0.995
```

```
        F[2 * 143 + 1] += -S_trapeze(p_hydro(nodes[143][1]), p_hydro(1 / 2 *
(nodes[142][1] + nodes[143][1])),
                             1 / 2 * distance(nodes[142], nodes[143])) * 1.395
        F[2 * 143] += S_trapeze(p_hydro(nodes[143][1]), p_hydro(1 / 2 *
(nodes[144][1] + nodes[143][1])),
                           1 / 2 * distance(nodes[144], nodes[143])) * 0.995
        F[2 * 143 + 1] += -S_trapeze(p_hydro(nodes[143][1]), p_hydro(1 / 2 *
(nodes[144][1] + nodes[143][1])),
                             1 / 2 * distance(nodes[144], nodes[143])) * 1.395

        F[2 * 144] += S_trapeze(p_hydro(nodes[144][1]), p_hydro(1 / 2 *
(nodes[144][1] + nodes[143][1])),
                           1 / 2 * distance(nodes[144], nodes[143])) * 0.995
        F[2 * 144 + 1] += -S_trapeze(p_hydro(nodes[144][1]), p_hydro(1 / 2 *
(nodes[144][1] + nodes[143][1])),
                             1 / 2 * distance(nodes[144], nodes[143])) * 1.395
        F[2 * 144] += S_trapeze(p_hydro(nodes[144][1]), p_hydro(1 / 2 *
(nodes[144][1] + nodes[5][1])),
                           1 / 2 * distance(nodes[144], nodes[5])) * 0.995
        F[2 * 144 + 1] += -S_trapeze(p_hydro(nodes[144][1]), p_hydro(1 / 2 *
(nodes[144][1] + nodes[5][1])),
                             1 / 2 * distance(nodes[144], nodes[5])) * 1.395

        F[2 * 5] += S_trapeze(p_hydro(nodes[5][1]), p_hydro(1 / 2 * (nodes[144][1]
+ nodes[5][1])),
                           1 / 2 * distance(nodes[144], nodes[5])) * 0.995
        F[2 * 5 + 1] += -S_trapeze(p_hydro(nodes[5][1]), p_hydro(1 / 2 *
(nodes[144][1] + nodes[5][1])),
                             1 / 2 * distance(nodes[144], nodes[5])) * 1.395
        # давление на горизонтальное ниженне
        p_vert = p_hydro(0)
        our_vert_value = p_vert * (
            abs(nodes[42][0] - nodes[5][0]))
        F[5 * 2 + 1] += -our_vert_value / 2
        F[42 * 2 + 1] += -our_vert_value
        F[43 * 2 + 1] += -our_vert_value
        F[44 * 2 + 1] += -our_vert_value
        F[45 * 2 + 1] += -our_vert_value
        F[46 * 2 + 1] += -our_vert_value
        F[47 * 2 + 1] += -our_vert_value
        F[48 * 2 + 1] += -our_vert_value
        F[49 * 2 + 1] += -our_vert_value
        F[50 * 2 + 1] += -our_vert_value
        F[51 * 2 + 1] += -our_vert_value
        F[6 * 2 + 1] += -our_vert_value/2
```

```
        # for i in range(len(self.Mass_node_B_X)):
        #     print('ii', i)
        #     K[self.Mass_node_B_X[i] * 2, :] = 0
        #     K[(self.Mass_node_B_X[i] * 2) + 1, :] = 0
        #     # K[:,self.Mass_node_T_air[i]] = 0
        #     K[self.Mass_node_B_X[i] * 2, self.Mass_node_B_X[i] * 2] = 1
        #     K[(self.Mass_node_B_X[i] * 2) + 1, (self.Mass_node_B_X[i] * 2) + 1] =
1
        #     F[self.Mass_node_B_X[i] * 2] = 0
        #     if i == 0 or i == len(self.Mass_node_B_X) - 1:
        #         F[(self.Mass_node_B_X[i] * 2) + 1] = -our_vert_value / 2
        #     else:
        #         F[(self.Mass_node_B_X[i] * 2) + 1] = -our_vert_value

        rho_1 = 2500
        rho_2 = 2200
        g = 9.8
        #массовые силы
        for number_el in self.Mass_Element_C:
            el = self.Mass_Element[number_el]
            # print('rrrrrr', i, el)
            Enodes = np.matrix([self.Mass_node[el[0]], self.Mass_node[el[1]],
self.Mass_node[el[2]]])
            nodes_current = self.Mass_Element[i]
            F_current = abs(np.linalg.det(self.get_J(Enodes))) / 2 * rho_1 * g / 3
            for j in range(3):
                F[2 * (nodes_current[j] - 1) + 1] += -F_current

        for number_el in self.Mass_Element_C2:
            el = self.Mass_Element[number_el]
            # print('rrrrrr', i, el)
            Enodes = np.matrix([self.Mass_node[el[0]], self.Mass_node[el[1]],
self.Mass_node[el[2]]])
            nodes_current = self.Mass_Element[i]
            F_current = abs(np.linalg.det(self.get_J(Enodes))) / 2 * rho_2 * g / 3
            for j in range(3):
                F[2 * (nodes_current[j] - 1) + 1] += -F_current



        print(K)
        # print('\n'.join('\t'.join(map(str, row)) for row in matrix))
        # for i in range(len(K)):
        #     for j in range(len(K[i])):
        #         print(K[i][j], end=' ')
```

```
        # print(K)
        T = solve(K, F)
        # np.savetxt('test1.txt', T, fmt='%.7f')
        # np.savetxt('test1.txt', [T,T], fmt='%.8e')
        my_file = open("uuuu12.txt", 'w')
        X = []
        Y = []
        for i, index in enumerate(T):
            if i % 2 == 0:
                X.append((index[0]))
            else:
                Y.append(((index[0])))
        XY = np.zeros((len(X), 3))
        for i in range(len(X)):
            XY[i, 0] = X[i]
            XY[i, 1] = Y[i]
            XY[i, 2] = 0
        print('qwer', X, Y, XY)
        my_file.write('\n'.join([str(i[0]) + ' ' + str(i[1]) + ' 0' for i in XY]))
        # '\n'.join([i[1:-1] for i in ','.join(map(str,T[0])).split(",")])
        my_file.close()
        print(T)
        print(len(T))


if __name__ == '__main__':
    node = open('nodes.txt', 'r')
    Mass_node = [[float(i) for i in (line.replace("  ", ").split(",")[1:])] for line in
node.read().splitlines()]
    elem_all = open('elem_nodes.txt', 'r')
    Mass_Element = [[int(i) - 1 for i in line.replace("  ", ").split(",")[1:]] for line in
elem_all.read().splitlines()]
    np.savetxt('MMMMM.txt', Mass_node, fmt='%d')

    el_Priming = [int(i) - 1 for i in
                  open('Priming_el_range.txt', 'r').read().replace("\n", ',').replace("  ",
").split(",")]
    Rage_el_Priming = [i for i in range(el_Priming[0], el_Priming[1] + 1)]
    str_P = ", ".join(map(str, Rage_el_Priming))
    # print(str_P)
    np.savetxt('elem_Priming.txt', Rage_el_Priming, fmt='%i')
    print(Rage_el_Priming)
    elem_Priming = open('elem_Priming.txt', 'r')
    # int(i) - 1
```

```
    Mass_Element_Priming = [int(i) - 1 for i in elem_Priming.read()[:-
1].replace("\n", ',').replace(" ", '').split(",")]
    print('aaa', Mass_Element_Priming)

    elem_Concrete = open('elem_Concrete.txt', 'r')
    Mass_Element_C = [int(i) - 1 for i in elem_Concrete.read().replace("\n",
',').replace(" ", '').split(",")]

    el_C2 = [int(i) - 1 for i in open('C2_el_range.txt', 'r').read().replace("\n",
',').replace(" ", '').split(",")]
    Rage_el_C2 = [i for i in range(el_C2[0], el_C2[1] + 1)]
    np.savetxt('elem_Concrete2.txt', Rage_el_C2, fmt='%i')
    print(Rage_el_C2)
    elem_Concrete2 = open('elem_Concrete2.txt', 'r')
    Mass_Element_C2 = [int(i) - 1 for i in elem_Concrete2.read()[:-1].replace("\n",
',').replace(" ", '').split(",")]

    Mass_node_B_X = [int(i) - 1 for i in
                open('nodes_BC_X.txt', 'r').read().replace("\n", ',').replace(" ",
'').split(",")]

    Mass_node_B_Y = [int(i) - 1 for i in
                open('nodes_BC_Y.txt', 'r').read().replace("\n", ',').replace(" ",
'').split(",")]
    E_Priming = 1.7e+10
    nu_Priming = 0.2
    E_C = 2.5e+10
    nu_C = 0.2
    E_C2 = 2.2e+10
    nu_C2 = 0.2
    # rho = 7800
    # M = 10000
    l = 0.1
    x = 1
    test = dam_static (x, l, Mass_node, Mass_Element, Mass_Element_Priming,
Mass_Element_C, Mass_Element_C,
                       E_Priming,
                       nu_Priming, E_C, nu_C, E_C2, nu_C2, Mass_node_B_X,
Mass_node_B_Y)
    test.Solve()
```