Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Санкт-Петербургский государственный архитектурно-строительный университет»**

Факультет строительный

Кафедра информационных технологий

**ОТЧЕТ ПО НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ**

**Линейно-упругая задача для стальной плиты**

Обучающийся Мельниченко Д. С.

Направление подготовки: 01.03.02 – Прикладная математика и информатика

Группа: ПМИб-3

Оценка

Дата

Преподаватель

Семенов А.А., заведующий каф. ИТ

*(подпись) (Ф.И.О., должность)*

Санкт-Петербург

2023 г.

# ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc134657260)

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc134657261)

[ОБЩИЕ ТЕОРЕТИЧЕСКИЕ МАТЕРИАЛЫ 4](#_Toc134657262)

[МАТЕМАТИЧЕСКАЯ МОДЕЛЬ 5](#_Toc134657263)

[АЛГОРИТМ РЕШЕНИЯ 7](#_Toc134657264)

[РАСЧЕТЫ 8](#_Toc134657265)

[1.Ортотропный материал – углепластик Т300/976 8](#_Toc134657266)

[2.Изотропный материал – Сталь С345 11](#_Toc134657267)

[3.Изотропный материал – Оргстекло 14](#_Toc134657268)

[Графики 17](#_Toc134657269)

[1.Графики для углепластика Т300/976 17](#_Toc134657270)

[2.Графики для углепластик Сталь 21](#_Toc134657271)

[3.Графики для углепластик Оргстекло 25](#_Toc134657272)

[ЗАКЛЮЧЕНИЕ 29](#_Toc134657273)

[СПИСОК ЛИТЕРАТУРЫ 30](#_Toc134657274)

[ПРИЛОЖЕНИЕ 31](#_Toc134657275)

# ПОСТАНОВКА ЗАДАЧИ

По полученному индивидуальному заданию, необходимо провести расчеты напряженно – деформированного состояния с применением метода Ритца к функционалу полной энергии деформации оболочки. В данном случае, был получен вариант пологой оболочки прямоугольного плана. Необходимо рассмотреть пологую оболочку двоякой кривизны с параметрами Ляме *A*= 1*, B*= 1 и главными кривизнами (, )

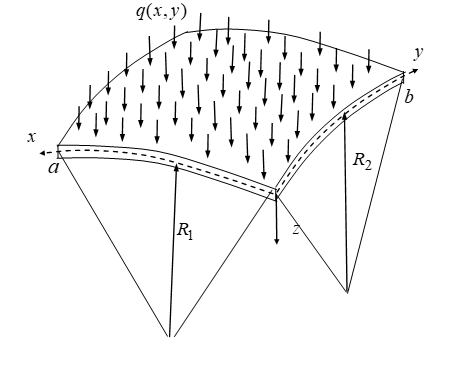


Рис.1 Визуальная модель рассматриваемой оболочки

# ОБЩИЕ ТЕОРЕТИЧЕСКИЕ МАТЕРИАЛЫ

Рассмотрим основные пункты решения задач теории оболочек и допущения:

1. Геометрически линейный вариант оболочки (модель Кирхгофа – Лява).
2. Геометрические соотношения в слое, отстающим на *z* от срединного.

Геометрические соотношения в срединной поверхности оболочки имеют вид:



Геометрические соотношения в срединной поверхности, отстающей на z от срединной поверхности оболочки, примут вид:



где:



В данной модели, основным допущением является то, что материалы, из которых состоит оболочка представляют собой однородную и сплошную среду. Таким образом, работая с ними можно использовать все возможности математического анализа, а также вариационного исчисления.

# МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

Для твёрдого деформируемого тела вводится понятие полной энергии , равной сумме потенциальной энергии деформации тела (потенциалу внутренних сил)  и энергии внешних сил(потенциалу внешних сил) *A*:



В данном случае, работа внешних сил берется с минусом ввиду взятой системы отсчета.





Где:

 – нормальные усилия в направлении осей *x,y* и сдвиговые условия в плоскости *xOy* соответственно;

 – изгибающие моменты в направлении осей *x,y* и крутящие моменты.

Для Изотропного случая:

Для Ортотропного случая:

# АЛГОРИТМ РЕШЕНИЯ

Для решения данной задачи, нам необходимо применить метод Ритца[3], который заключается в том, чтобы найти минимум функционала полной энергии оболочки. Иными словами, необходимо найти такие , что приведут к наиболее правильным результатам. При нахождении коэф. аппроксимирующих функций, можно будет построить наиболее точную модель, которая наглядно может показать прогибы оболочки в координате (*x, y*) при выбранной нагрузке.

В случае пологой оболочки, в качестве аппроксимирующих функций, выбираются следующие:



Где *a ,b –*линейныеразмеры оболочки, а в данной задаче *a*= *b*= 0.2 [м].

При выбранных аппроксимирующих функциях, соблюдаются все краевые условия под шарнирно – неподвижный тип заделки.

Для нахождения всех аппроксимирующих коэффициентов, необходимо продифференцировать функционал по каждому неизвестному, и составить из них систему. Данная систему представляет собой систему линейных уравнений, решить которую можно применяя правила решения СЛАУ из линейной алгебры.

# РАСЧЕТЫ

# 1.Ортотропный материал – углепластик Т300/976

Характеристики материала:



Также, его линейные размеры – 0,2 [м] вдоль *x* и 0,2 [м] вдоль *y*, а закрепление – шарнирно неподвижное.

Таблица 1. Зависимость  от различной толщины и разного кол-ва *N*

при нагрузке *q* = 0.0024 [МПа]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | *h* [м] | *N* | Max (*W*) [м] | Max () [МПа] | Max () [МПа] | Max () [МПа] |
| 1 | 0.00022 | 1 | 0.00206 | -101.354 | -8.575 | 0.615 |
| 4 | 0.00192 | -91.675 | -7.791 | 0.747 |
| 9 | 0.00194 | -94.064 | -7.998 | 0.784 |
| 16 | 0.00193 | -93.255 | -7.912 | 0.797 |
| 2 | 0.00044 | 1 | 0.000953 | -52.048 | -4.325 | 0.569 |
| 4 | 0.000901 | -47.027 | -3.916 | 0.669 |
| 9 | 0.000906 | -47.998 | -4.015 | 0.689 |
| 3 | 0.00088 | 1 | 0.000446 | -29.155 | -2.356 | 0.534 |
| 4 | 0.0004306 | -26.968 | -2.165 | 0.592 |
| 9 | 0.000432 | -27.286 | -2.204 | 0.600502 |

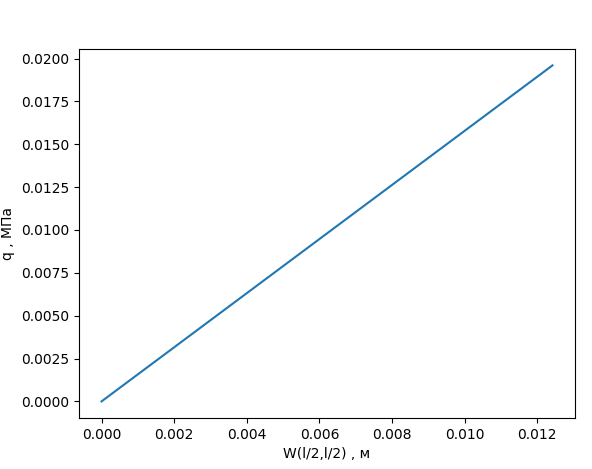


Рисунок 2 График *W (x,y)* при поиске *q* критического для материала Т300/976

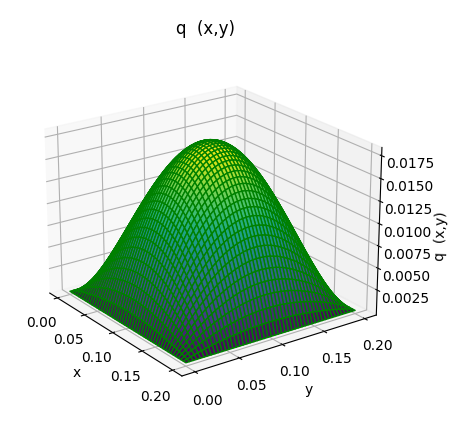


Рисунок 3 График *q (x,y)* вдоль *x* и *y* при *q* = 0.018036 приведен в [МПа]

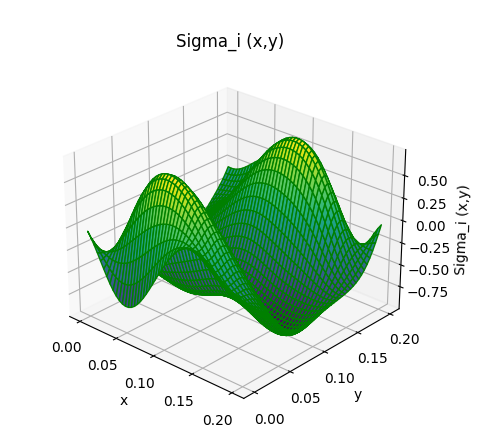


Рисунок 4 График  вдоль *x* и *y* при *q* = 0.018036 приведен в [МПа]

Найденное *q*= 0.018036 [МПа] максимальное, которое не нарушало бы критерий Хоффмана. Как можно заметить, критерий нарушается по бокам от центра, вдоль оси *x.*

# 2.Изотропный материал – Сталь С345

Характеристики материала:



Также, его линейные размеры – 0,2 [м] вдоль *x* и 0,2 [м] вдоль *y,*а закрепление – шарнирно неподвижное.

Таблица 2. Зависимость  от различной толщины и разного кол-ва *N*

при нагрузке *q* = 0.017345 [МПа]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | *h* [м] | *N* | Max (*W*) [м] | Max () [МПа] | Max () [МПа] | Max () [МПа] |
| 1 | 0.00022 | 1 | 0.00258 | -194.343 | -236.279 | 11.359 |
| 4 | 0.00249 | -184.423 | -224.987 | 12.627 |
| 9 | 0.0025 | -185.644 | -226.294 | 12.801 |
| 16 | 0.0025 | -185.382 | -226.022 | 12.836 |
| 2 | 0.00044 | 1 | 0.00154 | -128.15 | -153.097 | 13.515 |
| 4 | 0.00145 | -115.299 | -138.837 | 15.892 |
| 9 | 0.00146 | -117.913 | -141.603 | 16.335 |
| 3 | 0.00088 | 1 | 0.000719 | -71.651 | -83.314 | 12.639 |
| 4 | 0.000694 | -65.897 | -77.156 | 13.993 |
| 9 | 0.000696 | -66.707 | -77.992 | 14.147 |

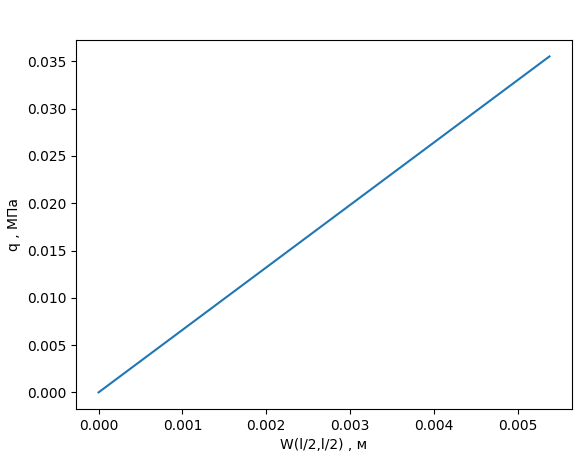


Рисунок 5 График *W (x,y)* при поиске *q* критического для материала Сталь С345

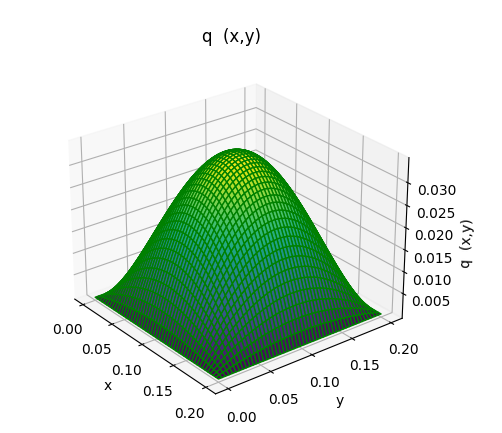


Рисунок 6 График *q (x,y)* вдоль всей оболочки при *q* = 0.03479 приведен в [МПа]

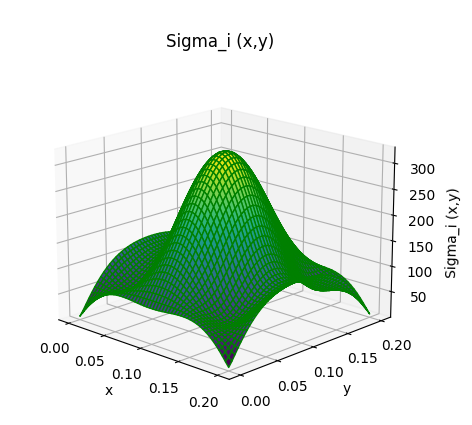


Рисунок 7 График  вдоль *x* и *y* при *q* = 0.03479 приведен в [МПа]

Найденное *q* = 0.03479 [МПа] максимальное, которое не нарушало бы критерий Мизиса. Так как нагрузка по оси *x* и *y* было задано параболическим законом, то мы можем сделать вывод, что критерий Мизиса был нарушен в середине оболочки, где нагрузка достигала своего максимального значения. Это можно пронаблюдать на Рис.7.

# 3.Изотропный материал – Оргстекло

Характеристики материала:



Также, его линейные размеры – 0,2 [м] вдоль *x* и 0,2 [м] вдоль *y*,а закрепление – шарнирно неподвижное.

Таблица 3. Зависимость  от различной толщины и разного кол-ва *N*

при нагрузке *q* = 0.002 [МПа]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | *h* [м] | *N* | Max (*W*) [м] | Max () [МПа] | Max () [МПа] | Max () [МПа] |
| 1 | 0.00022 | 1 | 0.018 | -21.423 | -25.51 | 1.107 |
| 4 | 0.0171 | -19.364 | -23.175 | 1.344 |
| 9 | 0.0173 | -19.895 | -23.751 | 1.412 |
| 16 | 0.0173 | -19.719 | -23.565 | 1.432 |
| 2 | 0.00044 | 1 | 0.00803 | -10.389 | -12.179 | 0.969 |
| 4 | 0.00758 | -9.358 | -11.048 | 1.139 |
| 9 | 0.00763 | -9.56 | -11.26 | 1.17 |
| 3 | 0.00088 | 1 | 0.00376 | -5.811 | -6.651 | 0.908 |
| 4 | 0.00363 | -5.354 | -6.164 | 1.0049 |
| 9 | 0.00364 | -5.415 | -6.227 | 1.015 |

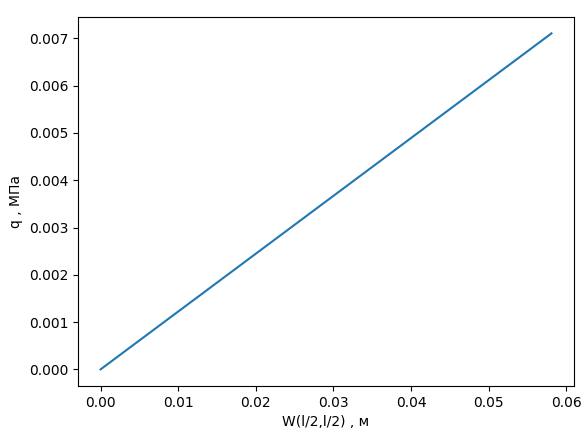


Рисунок 8 График *W (x,y)* при поиске *q* критического для материала Оргстекло

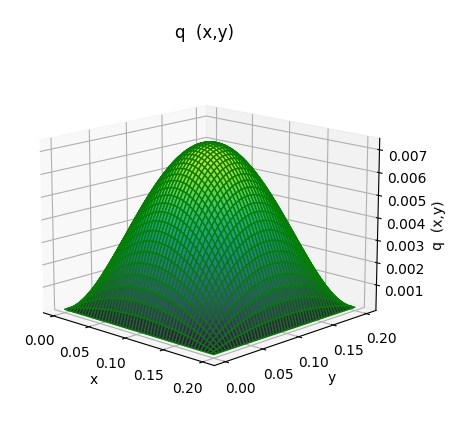


Рисунок 9 График *q (x,y)* вдоль всей оболочки при *q* = 0.00735 приведен в [МПа]

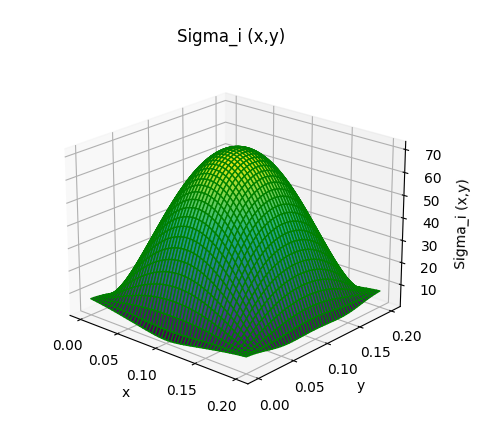


Рисунок 10 График  вдоль всей оболочки при *q* = 0.00735 приведен в [МПа]

Найденное *q*= 0.00735 [МПа] максимальное, которое не нарушало бы критерий Мизиса. Так как нагрузка по оси *x* и *y* было задано параболическим законом, то мы можем сделать вывод, что критерий Мизиса был нарушен в середине оболочки, где нагрузка достигала своего максимального значения. Это можно пронаблюдать на Рис.10.

# Графики

# 1.Графики для углепластика Т300/976

Для данных графиков использовались изначальные данные, а в качестве нагрузки было выбрано значение q = 0.002 [МПа]

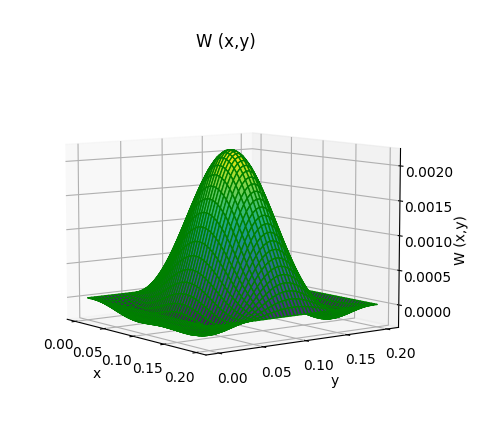


Рисунок 11. График *W (x,y)* при *N* = 4 приведен в [м]

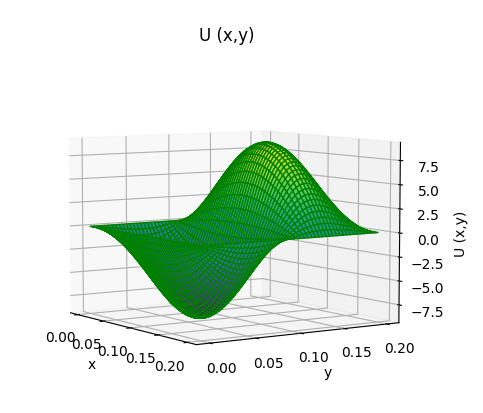


Рисунок 12. График *U (x,y)* при *N* = 4 приведенные значения в [м] увеличены в 106

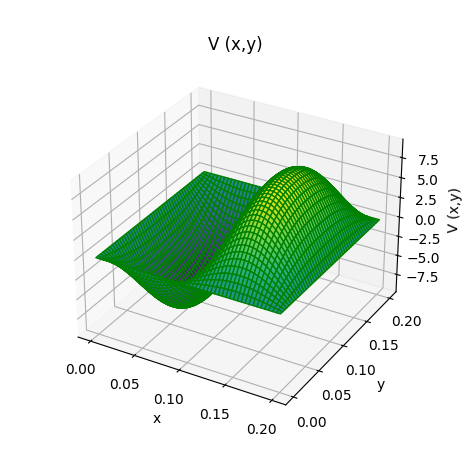


Рисунок 13. График *U (x,y)* при *N* = 4 приведенные значения в [м] увеличены в 106

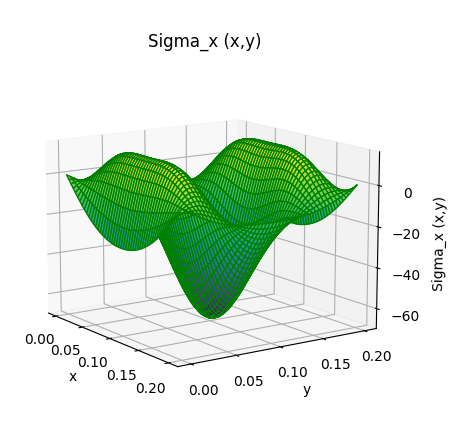


Рисунок 14. График  при *N* = 4 приведен в [МПа]

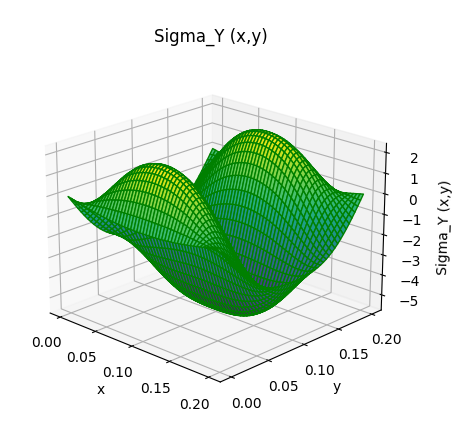


Рисунок 15. График  при *N* = 4 приведен в [МПа]

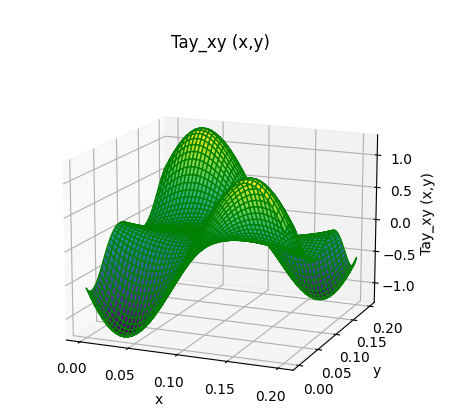


Рисунок 16. График  при *N* = 4 приведен в [МПа]

# 2.Графики для углепластик Сталь

Для данных графиков использовались изначальные данные, а в качестве нагрузки было выбрано значение q = 0.017345  [МПа]

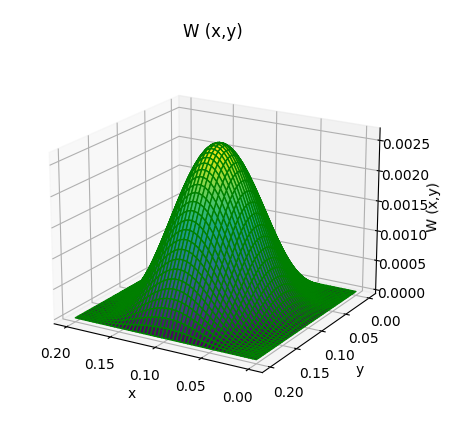


Рисунок 17. График *W (x,y)* при *N* = 4 приведен в [м]

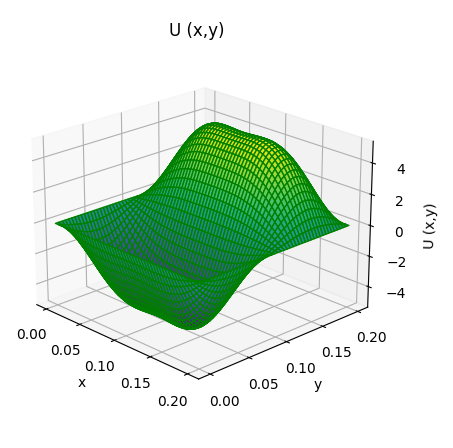


Рисунок 18. График *U (x,y)* при *N* = 4 приведенные значения в [м] увеличены в 106

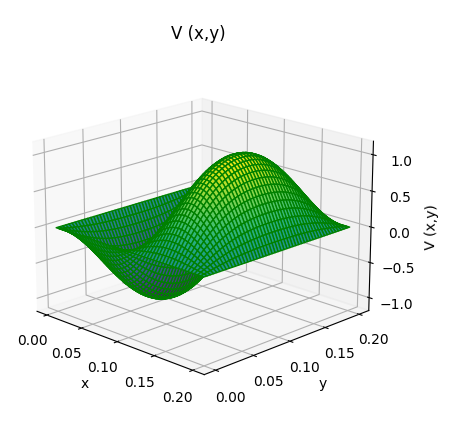


Рисунок 19. График *V (x,y)* при *N* = 4 приведенные значения в [м] увеличены в 106

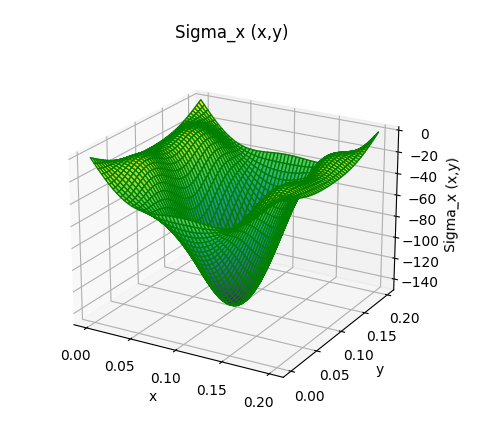


Рисунок 20. График  при *N* = 4 приведен в [МПа]

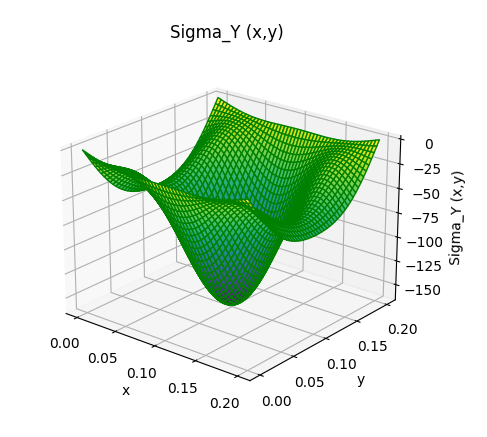


Рисунок 21. График  при *N* = 4 приведен в [МПа]

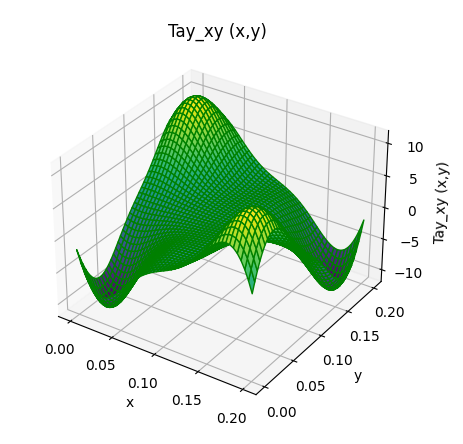


Рисунок 22. График  при *N* = 4 приведен в [МПа]

# 3.Графики для углепластик Оргстекло

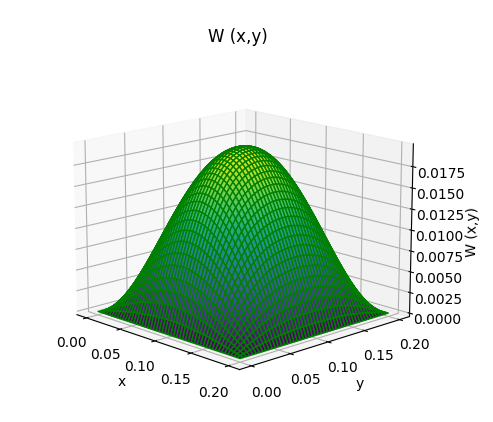


Рисунок 23. График *W (x,y)* при *N* = 4 приведен в [м]

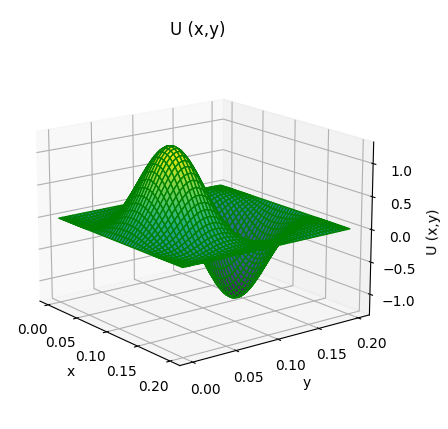
****

Рисунок 24. График *U (x,y)* при *N* = 4 приведенные значения в [м] увеличены в 106

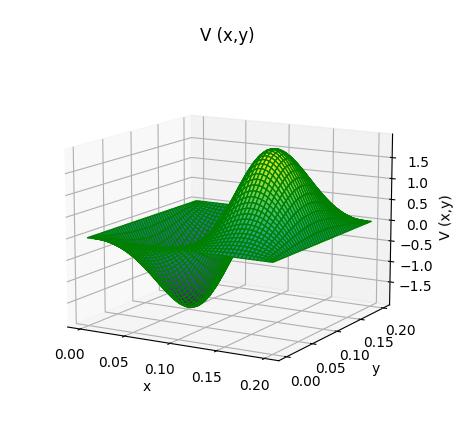
****

Рисунок 25. График *V (x,y)* при *N* = 4 приведенные значения в [м] увеличены в 106

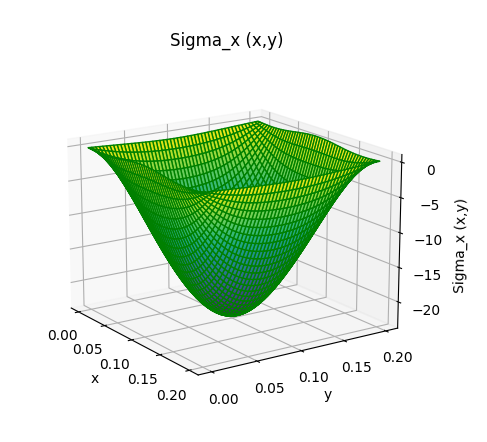
****

Рисунок 26. График  при *N* = 4 приведен в [МПа]

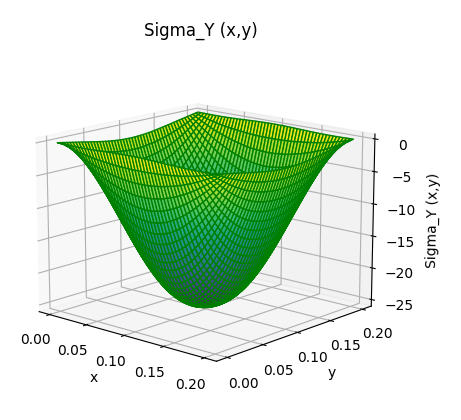
****

Рисунок 27. График  при *N* = 4 приведен в [МПа]

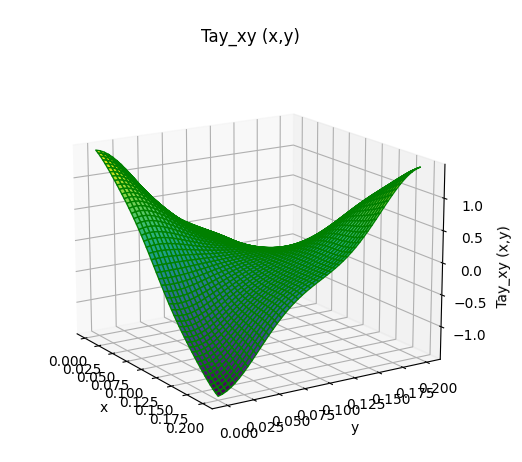
****

Рисунок 28. График  при *N* = 4 приведен в [МПа]

# ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены три материала: сталь С345, оргстекло, углепластик модели T300/976. Была про варьирована высота: базовая, умноженная на 2 и умноженная на 4. При изменении высоты, было выявлено, что максимальные прогибы уменьшаются прямо пропорционально увеличению толщины при одинаковой нагрузке.

Моделирование оболочек проводилось в среде PyCharm на языке программирования Python. В ходе решения были использованы основные математические пакеты данного языка: Sympy, Numpy, Math. Также, для построения графиков использовалась библиотека matplotlib интегрированная из среды MatLab.

Для каждой оболочки с заданными параметрами были найдены критические нагрузки, при которых оболочка терпела такие прогибы, после которых уже не могла прийти в изначальное состояние.

Также, были выведены графики прогибов для каждого материала при нагрузке, которая была ниже критической, и давала представление о прогибах в конструкции.

СПИСОК ЛИТЕРАТУРЫ

1. Карпов В. В. Математическое моделирование, алгоритмы исследования модели, вычислительный эксперимент в теории оболочек / В. В. Карпов; СПбГАСУ. – СПб., 2006. – 330 с.
2. Карпов В. В. Прочность и устойчивость подкрепленных оболочек вращения: в 2-х ч. Модели и алгоритмы исследования прочности и устойчивости подкрепленных оболочек вращения / В. В. Карпов. – М.: Физматлит, 2010. – Ч. 1. – 288 с
3. Вальтер Ритц (1909) "Über eine neue Methode zur Lösung gewisser Variationsprobleme der Mathematischen Physik" *Journal für die Reine und Angewandte Mathematik*, т. **135**, страницы 1–61. Доступно в Интернете по адресу: <http://gdz.sub.uni-goettingen.de/no_cache/dms/load/img/?IDDOC=261182> .

# ПРИЛОЖЕНИЕ

from sympy import \*  
import numpy as np  
import math as m  
import matplotlib.pyplot as plt  
  
  
#Change var : 1 for T300/976 (izo) ;2 for Org\_Glass ;3 for Still  
Change = 1  
Only\_Value\_in\_Break\_point = 1  
Q\_start = 0.003479 #1  
#Q\_start = 0.002 #2  
#Q\_start = 0.025 #3  
#Q\_start = 0.000  
#DATA  
  
  
V = 0.2\*0.2\*0.00022 #m^3  
P1=1500  
P2=7800  
P3=1190  
Square = 20\*20  
  
  
N\_x = 2  
N\_y = 2  
N = N\_x\*N\_y  
Q\_max = 245/(10\*\*6) #1 кг максимум  
mass=0  
if Change == 1:  
 mass = P1\*V  
if Change == 2:  
 mass = P2\*V  
if Change == 3:  
 mass = P3\*V  
Q\_my = (mass/Square)/(10\*\*6)  
  
#Symbols  
Xx = Symbol('x')  
Yy = Symbol('y')  
  
#Static\_DATA  
h=0.00022  
#h=0.00044  
#h=0.00088  
A\_lenght\_x = 0.2  
B\_lenght\_y = 0.2  
R\_1\_x = 5  
R\_2\_y = 3.33  
A = 1  
B = 1  
K\_x = 1/R\_1\_x  
K\_y = 1/R\_2\_y  
z = 0  
  
  
#Static\_DATA for variants:  
  
#Org:  
#T300/976  
E1\_T300 = 1.4 \* (10\*\*5)  
E2\_T300 = 0.97 \* (10\*\*4)  
nu\_T300 = 0.29  
G\_12\_T300 = 0.55 \* (10\*\*4)  
G\_13\_T300 = G\_12\_T300  
G\_23\_T300 = 0.33 \* (10\*\*4)  
F1\_min\_T300 = -1599  
F1\_max\_T300 = 1517  
F2\_min\_T300 = -253  
F2\_max\_T300 = 46  
F12\_max\_T300 = 41.4  
Density\_T300 = 1500  
Sigma\_t\_T300 = 100  
  
#Izo:  
#Org\_Glass\_1:  
E1\_OG = 0.03 \* (10\*\*5)  
E2\_OG = E1\_OG  
nu\_OG = 0.35  
G\_12\_OG = 0.012 \* (10\*\*5)  
G\_13\_OG = G\_12\_T300  
G\_23\_OG = G\_12\_T300  
Sigma\_t\_OG = 75  
Density\_OG = 1190  
  
#Still:  
E1\_Still = 2.1 \* (10\*\*5)  
E2\_Still = E1\_OG  
nu\_Still = 0.3  
G\_12\_Still = 0.807 \* (10\*\*5)  
G\_13\_Still = G\_12\_T300  
G\_23\_Still = G\_12\_T300  
Sigma\_t\_Still = 300  
Density\_Still = 7800  
  
  
  
#Settings to integral  
Start\_integral = 0  
  
#graph points  
Size = 50  
  
def Get\_W\_Plane(x\_val,y\_val,function,Values,type):  
 W\_result = 0  
  
 # Hard  
 if type == 1:  
 for j in range(1, N\_x + 1):  
 for i in range(1, N\_y + 1):  
 W\_result += Values[(j-1)\*N\_x + i-1] \* sin(2\*i \* x\_val \* m.pi / A\_lenght\_x) \* sin((2\*j - 1) \* y\_val \* m.pi / B\_lenght\_y)  
 if type == 2:  
 for j in range(1, N\_x + 1):  
 for i in range(1, N\_y + 1):  
 W\_result += Values[(j-1)\*N\_x + i-1] \* sin((2\*i - 1) \* x\_val \* m.pi / A\_lenght\_x) \* sin(2\*j \* y\_val \* m.pi / B\_lenght\_y)  
 if type == 3:  
 for j in range(1, N\_x + 1):  
 for i in range(1, N\_y + 1):  
 W\_result += Values[(j-1)\*N\_x + i-1] \* sin((2\*i - 1) \* x\_val \* m.pi / A\_lenght\_x) \* sin((2\*j - 1) \* y\_val \* m.pi / B\_lenght\_y)  
  
 #print(W\_result)  
 return W\_result  
def Draw\_3d\_W(Function,Values\_Result,type\_f):  
 x\_array = []  
 y\_array = []  
 z\_array = [0]\*Size  
 Max\_Value = 0  
 #print(Values\_Result)  
  
 #print()  
 #print(Function)  
  
 for i in range (0,Size):  
 z\_array[i] = [0]\*Size  
  
 step\_x = A\_lenght\_x / (Size - 1)  
 step\_y = B\_lenght\_y / (Size - 1)  
  
 for i in range(0,Size):  
 x\_array.append(i\*step\_x)  
 y\_array.append(i\*step\_y)  
  
 for j in range(0,Size):  
 for i in range(0,Size):  
 z\_array[i][j] = (Get\_W\_Plane(x\_array[i],y\_array[j],Function,Values\_Result,type\_f))  
 if abs(z\_array[i][j]) > abs(Max\_Value):  
 Max\_Value = z\_array[i][j]  
  
 z\_array = np.array(z\_array)  
 x\_array = np.array(x\_array)  
 y\_array = np.array(y\_array)  
 X, Y = np.meshgrid(x\_array, y\_array)  
  
 fig = plt.figure()  
 ax = plt.axes(projection='3d')  
 ax.plot\_surface(X, Y, z\_array, cmap='viridis', edgecolor='green')  
 ax.set\_xlabel('x')  
 ax.set\_ylabel('y')  
 ax.set\_zlabel(Function + ' (x,y)')  
 ax.set\_title(Function + ' (x,y)')  
 plt.show()  
 return Max\_Value  
def Draw\_3d\_Sigmas(Function, Values\_Result,Type\_Sigmas,U\_function,V\_function,W\_Function,z\_val,W\_val,U\_val,V\_val):  
 x\_array = []  
 y\_array = []  
 z\_array = [0] \* Size  
 Max\_Value = 0  
  
 for i in range(0, Size):  
 z\_array[i] = [0] \* Size  
  
 step\_x = A\_lenght\_x / (Size - 1)  
 step\_y = B\_lenght\_y / (Size - 1)  
  
 for i in range(0, Size):  
 x\_array.append(i \* step\_x)  
 y\_array.append(i \* step\_y)  
  
 Sigma\_x\_Function = 5 \* Xx  
 Sigma\_y\_Function = 5 \* Xx  
 Tay\_xy\_Function = 5 \* Xx  
  
 if Change == 1:  
 if Type\_Sigmas == 1:  
 Sigma\_x\_Function = (Get\_Sigma\_x\_Orto(U\_function,V\_function,W\_Function,E1\_T300,nu\_T300,nu\_T300,z\_val))  
 if Type\_Sigmas == 2:  
 Sigma\_y\_Function = (Get\_Sigma\_y\_Orto(U\_function,V\_function,W\_Function,E2\_T300,nu\_T300,nu\_T300,z\_val))  
 if Type\_Sigmas == 3:  
 Tay\_xy\_Function = (Get\_Sigma\_tay\_Orto(U\_function,V\_function,W\_Function,G\_12\_T300,z\_val))  
 if Change == 2:  
 if Type\_Sigmas == 1:  
 Sigma\_x\_Function = (Get\_Sigma\_x\_Izo(U\_function,V\_function,W\_Function,E1\_OG,nu\_OG,z\_val))  
 if Type\_Sigmas == 2:  
 Sigma\_y\_Function = (Get\_Sigma\_y\_Izo(U\_function,V\_function,W\_Function,E1\_OG,nu\_OG,z\_val))  
 if Type\_Sigmas == 3:  
 Tay\_xy\_Function = (Get\_Sigma\_tay\_Izo(U\_function,V\_function,W\_Function,E1\_OG,nu\_OG,z\_val))  
 if Change == 3:  
 if Type\_Sigmas == 1:  
 Sigma\_x\_Function = (Get\_Sigma\_x\_Izo(U\_function, V\_function, W\_Function, E1\_Still, nu\_Still, z\_val))  
 if Type\_Sigmas == 2:  
 Sigma\_y\_Function = (Get\_Sigma\_y\_Izo(U\_function, V\_function, W\_Function, E1\_Still, nu\_Still, z\_val))  
 if Type\_Sigmas == 3:  
 Tay\_xy\_Function = (Get\_Sigma\_tay\_Izo(U\_function, V\_function, W\_Function, E1\_Still, nu\_Still, z\_val))  
  
  
 for i in range(N+1):  
 Sigma\_x\_Function = Sigma\_x\_Function.subs([('w' + str(i),W\_val[i-1]),('u' + str(i),U\_val[i-1]),('v' + str(i),V\_val[i-1])])  
 Sigma\_y\_Function = Sigma\_y\_Function.subs([('w' + str(i),W\_val[i-1]),('u' + str(i),U\_val[i-1]),('v' + str(i),V\_val[i-1])])  
 Tay\_xy\_Function = Tay\_xy\_Function.subs([('w' + str(i),W\_val[i-1]),('u' + str(i),U\_val[i-1]),('v' + str(i),V\_val[i-1])])  
  
 #print(Sigma\_x\_Function)  
 #print(Sigma\_y\_Function)  
 #print(Tay\_xy\_Function)  
  
  
 for j in range(0, Size):  
 for i in range(0, Size):  
 if Type\_Sigmas == 1:  
 z\_array[i][j] = Sigma\_x\_Function.subs([(Xx, x\_array[i]), (Yy, y\_array[j])])  
 if Type\_Sigmas == 2:  
 z\_array[i][j] = Sigma\_y\_Function.subs([(Xx, x\_array[i]), (Yy, y\_array[j])])  
 if Type\_Sigmas == 3:  
 z\_array[i][j] = Tay\_xy\_Function.subs([(Xx, x\_array[i]), (Yy, y\_array[j])])  
 if abs(z\_array[i][j]) > abs(Max\_Value):  
 Max\_Value = z\_array[i][j]  
  
  
 z\_array = np.array(z\_array)  
 x\_array = np.array(x\_array)  
 y\_array = np.array(y\_array)  
 X, Y = np.meshgrid(x\_array, y\_array)  
  
 fig = plt.figure()  
 ax = plt.axes(projection='3d')  
 ax.plot\_surface(X, Y, z\_array, cmap='viridis', edgecolor='green')  
 ax.set\_xlabel('x')  
 ax.set\_ylabel('y')  
 ax.set\_zlabel(Function + ' (x,y)')  
 ax.set\_title(Function + ' (x,y)')  
 plt.show()  
 return Max\_Value  
def Get\_Sigmas\_max\_values(Function, Values\_Result,Type\_Sigmas,Sigma\_x\_Function,Sigma\_y\_Function,Tay\_xy\_Function,z\_val,W\_val,U\_val,V\_val):  
 x\_array = []  
 y\_array = []  
 z\_array = [0] \* Size  
  
 step\_x = A\_lenght\_x / (Size - 1)  
 step\_y = B\_lenght\_y / (Size - 1)  
  
 Max\_Value = 0  
  
 for i in range(0, Size):  
 z\_array[i] = [0] \* Size  
 x\_array.append(i \* step\_x)  
 y\_array.append(i \* step\_y)  
  
 for i in range(N+1):  
 if Type\_Sigmas == 1:  
 Sigma\_x\_Function = Sigma\_x\_Function.subs([('w' + str(i),W\_val[i-1]),('u' + str(i),U\_val[i-1]),('v' + str(i),V\_val[i-1])])  
 if Type\_Sigmas == 2:  
 Sigma\_y\_Function = Sigma\_y\_Function.subs([('w' + str(i),W\_val[i-1]),('u' + str(i),U\_val[i-1]),('v' + str(i),V\_val[i-1])])  
 if Type\_Sigmas == 3:  
 Tay\_xy\_Function = Tay\_xy\_Function.subs([('w' + str(i),W\_val[i-1]),('u' + str(i),U\_val[i-1]),('v' + str(i),V\_val[i-1])])  
  
 for j in range(0, Size):  
 for i in range(0, Size):  
 if Type\_Sigmas == 1:  
 z\_array[i][j] = Sigma\_x\_Function.subs([(Xx, x\_array[i]), (Yy, y\_array[j])])  
 if Type\_Sigmas == 2:  
 z\_array[i][j] = Sigma\_y\_Function.subs([(Xx, x\_array[i]), (Yy, y\_array[j])])  
 if Type\_Sigmas == 3:  
 z\_array[i][j] = Tay\_xy\_Function.subs([(Xx, x\_array[i]), (Yy, y\_array[j])])  
 if abs(z\_array[i][j]) > abs(Max\_Value):  
 Max\_Value = z\_array[i][j]  
  
 return Max\_Value  
  
  
def Get\_v\_coefs():  
 w\_coefs = []  
 for i in range(1, N + 1):  
 w\_coefs.append(Symbol('v' + str(i)))  
 return w\_coefs  
def Get\_u\_coefs():  
 w\_coefs = []  
 for i in range(1, N + 1):  
 w\_coefs.append(Symbol('u' + str(i)))  
 return w\_coefs  
def Get\_w\_coefs():  
 w\_coefs = []  
 for i in range(1, N + 1):  
 w\_coefs.append(Symbol('w' + str(i)))  
 return w\_coefs  
  
def Get\_w\_sin\_x(i):  
 return sin(i \* Xx \* m.pi / A\_lenght\_x)  
def Get\_w\_sin\_y(j):  
 return sin(j \* Yy \* m.pi / B\_lenght\_y)  
def Get\_w\_sin\_x\_2(i):  
 return sin((2\*i) \* Xx \* m.pi / A\_lenght\_x)  
def Get\_w\_sin\_y\_2(j):  
 return sin((2\*j) \* Yy \* m.pi / B\_lenght\_y)  
  
def Get\_U\_function\_vals(u\_vals):  
 Result = 0  
 for j in range(1, N\_x + 1):  
 for i in range(1, N\_y + 1):  
 Result += u\_vals[(j-1)\*N\_x + i-1] \* Get\_w\_sin\_x(i)\*Get\_w\_sin\_y(j)  
 return Result  
def Get\_V\_function\_vals(v\_vals):  
 Result = 0  
 for j in range(1, N\_x + 1):  
 for i in range(1, N\_y + 1):  
 Result += v\_vals[(j-1)\*N\_x + i-1] \* Get\_w\_sin\_x(i) \* Get\_w\_sin\_y(j)  
 return Result  
def Get\_W\_function\_vals(w\_vals):  
 Result = 0  
 for j in range(1, N\_x + 1):  
 for i in range(1, N\_y + 1):  
 Result += w\_vals[(j-1)\*N\_x + i-1] \* Get\_w\_sin\_x(i) \* Get\_w\_sin\_y(j)  
 return Result  
  
def ksi\_1(W\_function):  
 Buf = W\_function.diff(Xx)  
 Result = Buf.diff(Xx) \* (-1)  
 return Result  
def ksi\_2(W\_function):  
 Buf = W\_function.diff(Yy)  
 Result = Buf.diff(Yy) \* (-1)  
 return Result  
def ksi\_12(W\_function):  
 Result\_1 = -(W\_function.diff(Xx)).diff(Yy)  
 Result\_2 = -(W\_function.diff(Yy)).diff(Xx)  
 #Result = (Result\_1 + Result\_2)/2  
 Result = (Result\_1 + Result\_2)/2  
 return Result  
  
def e\_x(U\_function,V\_function,W\_Function):  
 Result = U\_function.diff(Xx) - K\_x \* W\_Function  
 return Result  
def e\_y(U\_function,V\_function,W\_Function):  
 Result = V\_function.diff(Yy) - K\_y \* W\_Function  
 return Result  
def y\_xy(U\_function,V\_function,W\_Function):  
 Result = V\_function.diff(Xx) + U\_function.diff(Yy)  
 return Result  
  
def e\_xz(z\_value,U\_function,V\_function,W\_Function):  
 Result = e\_x(U\_function,V\_function,W\_Function) + z\_value\*ksi\_1(W\_Function)  
 return Result  
def e\_yz(z\_value,U\_function,V\_function,W\_Function):  
 Result = e\_y(U\_function,V\_function,W\_Function) + z\_value\*ksi\_2(W\_Function)  
 return Result  
def y\_xz(z\_value,U\_function,V\_function,W\_Function):  
 Result = y\_xy(U\_function,V\_function,W\_Function) + 2\*z\_value\*ksi\_12(W\_Function)  
 return Result  
  
#Ortotrop functions  
def N\_x\_Orto(z,U\_function,V\_function,W\_Function,E\_1,nu\_12,nu\_21):  
 Result = (E\_1 / (1 -nu\_12\*nu\_21)) \* h \*(e\_xz(z,U\_function,V\_function,W\_Function) + nu\_21 \* e\_yz(z,U\_function,V\_function,W\_Function))  
 return Result  
def N\_y\_Orto(z, U\_function, V\_function, W\_Function,E\_2,nu\_12,nu\_21):  
 Result = (E\_2 / (1 - nu\_12 \* nu\_21)) \* h \* ( e\_yz(z, U\_function, V\_function, W\_Function) + nu\_12 \* e\_xz(z, U\_function, V\_function, W\_Function))  
 return Result  
def N\_xy\_Orto(z, U\_function, V\_function, W\_Function,G\_12):  
 Result = G\_12 \* h \* y\_xz(z,U\_function,V\_function,W\_Function)  
 return Result  
  
def M\_x\_Orto(W\_Function,E\_1,nu\_12,nu\_21):  
 Result = (E\_1 / (1 - nu\_12 \* nu\_21)) \* ((h\*\*3)/12) \* ( ksi\_2(W\_Function) + nu\_21 \* ksi\_1(W\_Function))  
 return Result  
def M\_y\_Orto(W\_Function,E\_2,nu\_12,nu\_21):  
 Result = (E\_2 / (1 - nu\_12 \* nu\_21)) \* ((h\*\*3)/12) \* ( ksi\_1(W\_Function) + nu\_12 \* ksi\_2(W\_Function))  
 return Result  
def M\_xy\_Orto(W\_Function,G\_12):  
 Result = 2 \* G\_12 \* ((h\*\*3)/12) \* ksi\_12(W\_Function)  
 return Result  
  
  
def Get\_Sigma\_x\_Orto(U\_function,V\_function,W\_Function,E\_1,nu\_12,nu\_21,z\_val):  
 Result = (E\_1 / (1 - nu\_12 \* nu\_21)) \* (e\_xz(z\_val, U\_function, V\_function, W\_Function) + nu\_21 \* e\_yz(z\_val, U\_function, V\_function, W\_Function))  
 return Result  
def Get\_Sigma\_y\_Orto(U\_function,V\_function,W\_Function,E\_2,nu\_12,nu\_21,z\_val):  
 Result = (E\_2 / (1 - nu\_12 \* nu\_21)) \* (e\_yz(z\_val, U\_function, V\_function, W\_Function) + nu\_12 \* e\_xz(z\_val, U\_function, V\_function, W\_Function))  
 return Result  
def Get\_Sigma\_tay\_Orto(U\_function,V\_function,W\_Function,G\_12,z\_val):  
 Result = G\_12 \* y\_xz(z\_val,U\_function,V\_function,W\_Function)  
 return Result  
  
#Izotrop functions  
def N\_x\_Izo(z,U\_function,V\_function,W\_Function,E,nu):  
 Result = (E / (1 -nu\*nu)) \* h \*(e\_xz(z,U\_function,V\_function,W\_Function) + nu \* e\_yz(z,U\_function,V\_function,W\_Function))  
 return Result  
def N\_y\_Izo(z, U\_function, V\_function, W\_Function,E,nu):  
 Result = (E / (1 - nu \* nu)) \* h \* ( e\_yz(z, U\_function, V\_function, W\_Function) + nu \* e\_xz(z, U\_function, V\_function, W\_Function))  
 return Result  
def N\_xy\_Izo(z, U\_function, V\_function, W\_Function,E,nu):  
 Result = (E\*h\*y\_xz(z, U\_function, V\_function, W\_Function)) / (2 + 2\*nu)  
 return Result  
  
def M\_x\_Izo(W\_Function,E,nu):  
 Result = (E / (1 - nu \* nu)) \* ((h\*\*3)/12) \* ( ksi\_2(W\_Function) + nu \* ksi\_1(W\_Function))  
 return Result  
def M\_y\_Izo(W\_Function,E,nu):  
 Result = (E / (1 - nu \* nu)) \* ((h\*\*3)/12) \* ( ksi\_1(W\_Function) + nu \* ksi\_2(W\_Function))  
 return Result  
def M\_xy\_Izo(W\_Function,E,nu):  
 Result = E \* ((h\*\*3)/12) \* ksi\_12(W\_Function)/(1 + nu)  
 return Result  
  
def Get\_Sigma\_x\_Izo(U\_function,V\_function,W\_Function,E,nu,z\_val):  
 Result = (E / (1 -nu\*nu))\*(e\_xz(z\_val,U\_function,V\_function,W\_Function) + nu \* e\_yz(z\_val,U\_function,V\_function,W\_Function))  
 return Result  
def Get\_Sigma\_y\_Izo(U\_function,V\_function,W\_Function,E,nu,z\_val):  
 Result = (E / (1 - nu \* nu)) \* (e\_yz(z\_val, U\_function, V\_function, W\_Function) + nu \* e\_xz(z\_val, U\_function, V\_function, W\_Function))  
 return Result  
def Get\_Sigma\_tay\_Izo(U\_function,V\_function,W\_Function,E,nu,z\_val):  
 Result = (E / (2 + 2\*nu)) \* y\_xz(z\_val,U\_function,V\_function,W\_Function)  
 return Result  
  
def q\_function(q\_0,q\_sv):  
 Result = 0  
 A1 = 0  
 A\_1 = [0] \* 3  
 A\_2 = [0] \* 3  
  
 A\_1[0] = 1 - ((A\_lenght\_x + A1) / (A\_lenght\_x - A1))\*\*2  
 A\_1[1] = 4\*(A\_lenght\_x + A1) / ((A\_lenght\_x - A1)\*\*2)  
 A\_1[2] = -4 / ((A\_lenght\_x - A1)\*\*2)  
  
 A\_2[0] = 1 - ((B\_lenght\_y + A1) / (B\_lenght\_y - A1)) \*\* 2  
 A\_2[1] = 4 \* (B\_lenght\_y + A1) / ((B\_lenght\_y - A1) \*\* 2)  
 A\_2[2] = -4 / ((B\_lenght\_y - A1) \*\* 2)  
  
  
 Result = q\_0\*(A\_1[0] + A\_1[1]\*Xx + A\_1[2]\*(Xx\*\*2)) \* (A\_2[0] + A\_2[1]\*Yy + A\_2[2]\*(Yy\*\*2)) + q\_sv  
 return Result  
def Get\_Answer(Es\_Get,w\_coef,u\_coef,v\_coef):  
 Es = Es\_Get.copy()  
  
 Es[0] = integrate(Es[0], (Xx, Start\_integral, A\_lenght\_x))  
 Es[1] = integrate(Es[1], (Xx, Start\_integral, A\_lenght\_x))  
 Es[2] = integrate(Es[2], (Xx, Start\_integral, A\_lenght\_x))  
 Es[3] = integrate(Es[3], (Xx, Start\_integral, A\_lenght\_x))  
 Es[4] = integrate(Es[4], (Xx, Start\_integral, A\_lenght\_x))  
 Es[5] = integrate(Es[5], (Xx, Start\_integral, A\_lenght\_x))  
  
 Es[0] = (1/2) \* integrate(Es[0], (Yy, Start\_integral, B\_lenght\_y))  
 Es[1] = (1/2) \* integrate(Es[1], (Yy, Start\_integral, B\_lenght\_y))  
 Es[2] = (1/2) \* integrate(Es[2], (Yy, Start\_integral, B\_lenght\_y))  
 Es[3] = (1/2) \* integrate(Es[3], (Yy, Start\_integral, B\_lenght\_y))  
 Es[4] = (1/2) \* integrate(Es[4], (Yy, Start\_integral, B\_lenght\_y))  
 Es[5] = (1/2) \* integrate(Es[5], (Yy, Start\_integral, B\_lenght\_y))  
  
 Result = []  
 Result\_buf = []  
 Buf = []  
 Zeroes = [0] \* N \* 3  
 Buf\_Symbols = []  
  
 #print("Start diff)")  
 for index in range(1, 4):  
 Result.append(Result\_buf)  
  
 for i in range(1, N + 1):  
 Buf.append((Es[0].diff(w\_coef[i - 1]) + Es[1].diff(w\_coef[i - 1]) + Es[2].diff(w\_coef[i - 1]) + Es[3].diff(  
 w\_coef[i - 1]) + Es[4].diff(w\_coef[i - 1]) + Es[5].diff(w\_coef[i - 1])))  
 Buf\_Symbols.append(w\_coef[i - 1])  
  
 for i in range(1, N + 1):  
 Buf.append((Es[0].diff(u\_coef[i - 1]) + Es[1].diff(u\_coef[i - 1]) + Es[2].diff(u\_coef[i - 1]) + Es[3].diff(  
 w\_coef[i - 1]) + Es[4].diff(u\_coef[i - 1]) + Es[5].diff(u\_coef[i - 1])))  
 Buf\_Symbols.append(u\_coef[i - 1])  
  
 for i in range(1, N + 1):  
 Buf.append((Es[0].diff(v\_coef[i - 1]) + Es[1].diff(v\_coef[i - 1]) + Es[2].diff(v\_coef[i - 1]) + Es[3].diff(  
 v\_coef[i - 1]) + Es[4].diff(v\_coef[i - 1]) + Es[5].diff(v\_coef[i - 1])))  
 Buf\_Symbols.append(v\_coef[i - 1])  
  
 for i in range(len(Buf)):  
 Buf[i] = nsimplify(Buf[i], tolerance=1e-20).evalf(15)  
  
 for solution in linsolve(Buf, Buf\_Symbols):  
 Result = solution  
 #print("End diff)")  
 return Result  
  
  
#Main code  
#Collect W-V func  
w\_vals = Get\_w\_coefs()  
u\_vals = Get\_u\_coefs()  
v\_vals = Get\_v\_coefs()  
W\_Function = Get\_W\_function\_vals(w\_vals)  
U\_function = Get\_U\_function\_vals(u\_vals)  
V\_function = Get\_V\_function\_vals(v\_vals)  
  
#Заготовочка array's  
Max\_W\_values = [0]\*3  
Max\_Sigmas\_values = [0]\*3  
Es\_main = [0]\*6  
W\_middle\_values = []  
Q\_values = []  
  
#Заготовочка  
Sigma\_x\_Function = 5 \* Xx  
Sigma\_y\_Function = 5 \* Xx  
Tay\_xy\_Function = 5 \* Xx  
Q\_function = 5 \* Xx  
  
  
#Sigma max  
QQ = 0  
Check = 1  
if Change == 1:  
 QQ = Sigma\_t\_T300  
if Change == 2:  
 QQ = Sigma\_t\_OG  
if Change == 3:  
 QQ = Sigma\_t\_Still  
  
#Sigmas  
U\_function\_buf = U\_function.copy()  
V\_function\_buf = V\_function.copy()  
W\_function\_buf = W\_Function.copy()  
z\_val = -h / 2  
if Change == 1:  
 Sigma\_x\_Function = Get\_Sigma\_x\_Orto(U\_function\_buf, V\_function\_buf, W\_function\_buf, E1\_T300, nu\_T300, nu\_T300, z\_val)  
 Sigma\_y\_Function = Get\_Sigma\_y\_Orto(U\_function\_buf, V\_function\_buf, W\_function\_buf, E2\_T300, nu\_T300, nu\_T300, z\_val)  
 Tay\_xy\_Function = Get\_Sigma\_tay\_Orto(U\_function\_buf, V\_function\_buf, W\_function\_buf, G\_12\_T300, z\_val)  
if Change == 2:  
 Sigma\_x\_Function = (Get\_Sigma\_x\_Izo(U\_function\_buf, V\_function\_buf, W\_function\_buf, E1\_OG, nu\_OG, z\_val))  
 Sigma\_y\_Function = (Get\_Sigma\_y\_Izo(U\_function\_buf, V\_function\_buf, W\_function\_buf, E1\_OG, nu\_OG, z\_val))  
 Tay\_xy\_Function = (Get\_Sigma\_tay\_Izo(U\_function\_buf, V\_function\_buf, W\_function\_buf, E1\_OG, nu\_OG, z\_val))  
if Change == 3:  
 Sigma\_x\_Function = (Get\_Sigma\_x\_Izo(U\_function\_buf, V\_function\_buf, W\_function\_buf, E1\_Still, nu\_Still, z\_val))  
 Sigma\_y\_Function = (Get\_Sigma\_y\_Izo(U\_function\_buf, V\_function\_buf, W\_function\_buf, E1\_Still, nu\_Still, z\_val))  
 Tay\_xy\_Function = (Get\_Sigma\_tay\_Izo(U\_function\_buf, V\_function\_buf, W\_function\_buf, E1\_Still, nu\_Still, z\_val))  
  
Count\_num = 0  
Q\_now =Q\_start  
  
while Check:  
 Count\_num +=1  
 W\_val = []  
 U\_val = []  
 V\_val = []  
 W\_values = []  
  
 # Es main  
 z\_num = 0  
 if Change == 1:  
 Es\_main[0] = N\_x\_Orto(z\_num, U\_function, V\_function, W\_Function, E1\_T300, nu\_T300, nu\_T300) \* e\_xz(z\_num, U\_function, V\_function,W\_Function)  
 Es\_main[1] = N\_y\_Orto(z\_num, U\_function, V\_function, W\_Function, E2\_T300, nu\_T300, nu\_T300) \* e\_yz(z\_num, U\_function,V\_function,W\_Function)  
 Es\_main[2] = N\_xy\_Orto(z\_num, U\_function, V\_function, W\_Function, G\_12\_T300) \* y\_xz(z\_num, U\_function, V\_function,W\_Function)  
 Es\_main[3] = M\_x\_Orto(W\_Function, E1\_T300, nu\_T300, nu\_T300) \* ksi\_1(W\_Function) + M\_y\_Orto(W\_Function, E2\_T300,nu\_T300,nu\_T300) \* ksi\_2(W\_Function)  
 Es\_main[4] = 2 \* M\_xy\_Orto(W\_Function, G\_12\_T300) \* ksi\_12(W\_Function)  
 if Change == 2:  
 Es\_main[0] = N\_x\_Izo(z\_num, U\_function, V\_function, W\_Function, E1\_OG, nu\_OG) \* e\_xz(z\_num, U\_function, V\_function,W\_Function)  
 Es\_main[1] = N\_y\_Izo(z\_num, U\_function, V\_function, W\_Function, E1\_OG, nu\_OG) \* e\_yz(z\_num, U\_function, V\_function,W\_Function)  
 Es\_main[2] = (N\_xy\_Izo(z\_num, U\_function, V\_function, W\_Function, E1\_Still, nu\_Still) + N\_xy\_Izo(z\_num, U\_function,V\_function,W\_Function, E1\_Still,nu\_Still)) \* y\_xz(z\_num, U\_function, V\_function, W\_Function)  
 Es\_main[3] = M\_x\_Izo(W\_Function, E1\_OG, nu\_OG) \* ksi\_1(W\_Function) + M\_y\_Izo(W\_Function, E1\_OG, nu\_OG) \* ksi\_2(W\_Function)  
 Es\_main[4] = (M\_xy\_Izo(W\_Function, E1\_OG, nu\_OG) + M\_xy\_Izo(W\_Function, E1\_OG, nu\_OG)) \* ksi\_12(W\_Function)  
 if Change == 3:  
 Es\_main[0] = N\_x\_Izo(z\_num, U\_function, V\_function, W\_Function, E1\_Still, nu\_Still) \* e\_xz(z\_num, U\_function,V\_function, W\_Function)  
 Es\_main[1] = N\_y\_Izo(z\_num, U\_function, V\_function, W\_Function, E1\_Still, nu\_Still) \* e\_yz(z\_num, U\_function,V\_function, W\_Function)  
 Es\_main[2] = (N\_xy\_Izo(z\_num, U\_function, V\_function, W\_Function, E1\_Still, nu\_Still) + N\_xy\_Izo(z\_num, U\_function,V\_function,W\_Function,E1\_Still,nu\_Still)) \* y\_xz(z\_num, U\_function, V\_function, W\_Function)  
 Es\_main[3] = M\_x\_Izo(W\_Function, E1\_Still, nu\_Still) \* ksi\_1(W\_Function) + M\_y\_Izo(W\_Function, E1\_Still,nu\_Still) \* ksi\_2(W\_Function)  
 Es\_main[4] = (M\_xy\_Izo(W\_Function, E1\_Still, nu\_Still) + M\_xy\_Izo(W\_Function, E1\_Still, nu\_Still)) \* ksi\_12(W\_Function)  
  
 Es\_main[5] = (-2)\*q\_function(Q\_now,Q\_my)\*W\_Function  
 Es\_main\_buf = Es\_main.copy()  
 Q\_function = q\_function(Q\_now, Q\_my)  
  
  
 W\_values = Get\_Answer(Es\_main\_buf, w\_vals, u\_vals, v\_vals)  
  
 print(W\_values)  
 for i in range(1, N + 1):  
 W\_val.append(W\_values[i - 1])  
 U\_val.append(W\_values[i - 1 + N])  
 V\_val.append(W\_values[i - 1 + 2 \* N])  
  
 for i in range(N + 1):  
 Q\_function = Q\_function.subs('w' + str(i), W\_val[i - 1])  
 Q\_function = Q\_function.subs(Xx,A\_lenght\_x/2)  
 Q\_function = Q\_function.subs(Yy,B\_lenght\_y/ 2)  
 Q\_values.append(Q\_function)  
  
  
 z = -h/2  
 Max\_Sigmas\_values[0] = Get\_Sigmas\_max\_values('Sigma\_x', W\_val, 1, Sigma\_x\_Function, Sigma\_y\_Function, Tay\_xy\_Function, z, W\_val, U\_val, V\_val)  
 Max\_Sigmas\_values[1] = Get\_Sigmas\_max\_values('Sigma\_Y', W\_val, 2, Sigma\_x\_Function, Sigma\_y\_Function, Tay\_xy\_Function, z, W\_val, U\_val, V\_val)  
 Max\_Sigmas\_values[2] = Get\_Sigmas\_max\_values('Tay\_xy', W\_val, 3, Sigma\_x\_Function, Sigma\_y\_Function, Tay\_xy\_Function, z, W\_val, U\_val, V\_val)  
 Sigma\_max\_global = ((Max\_Sigmas\_values[0]\*\*2) + (Max\_Sigmas\_values[1]\*\*2) - Max\_Sigmas\_values[0]\*Max\_Sigmas\_values[1] + 3\*(Max\_Sigmas\_values[2]\*\*2))\*\*(1/2)  
  
 W\_middle\_values.append(Get\_W\_Plane(A\_lenght\_x / 2, B\_lenght\_y / 2, W\_Function, W\_val,3))  
  
  
 Q\_now += Q\_max  
 if Sigma\_max\_global > QQ and Only\_Value\_in\_Break\_point != 1:  
 Check = 0  
 if Only\_Value\_in\_Break\_point and Count\_num == 1:  
 Check = 0  
 Q\_now -= Q\_max  
  
#Q\_graph  
#plt.plot(W\_middle\_values,Q\_values)  
#plt.ylabel("q , МПа")  
#plt.xlabel('W(l/2,l/2) , м')  
#plt.show()  
#print((Q\_now + Q\_my)\*(10\*\*6))  
#print(Count\_num)  
  
#Prints data :  
print("W values = ",W\_val)  
print("U values = ",U\_val)  
print("V values = ",V\_val)  
  
  
#Graph's :  
  
Max\_W\_values[0] = Draw\_3d\_W('W',W\_val,3)  
Max\_W\_values[1] = Draw\_3d\_W('U',U\_val,1)  
Max\_W\_values[2] = Draw\_3d\_W('V',V\_val,2)  
  
print("max W val = ",Max\_W\_values[0])  
print("max U val = ",Max\_W\_values[1])  
print("max V val = ",Max\_W\_values[2])  
  
  
z=-h/2  
  
Max\_Sigmas\_values[0] = Draw\_3d\_Sigmas('Sigma\_x',W\_val,1,U\_function,V\_function,W\_Function,z,W\_val,U\_val,V\_val)  
Max\_Sigmas\_values[1] = Draw\_3d\_Sigmas('Sigma\_Y',W\_val,2,U\_function,V\_function,W\_Function,z,W\_val,U\_val,V\_val)  
Max\_Sigmas\_values[2] = Draw\_3d\_Sigmas('Tay\_xy',W\_val,3,U\_function,V\_function,W\_Function,z,W\_val,U\_val,V\_val)  
  
print("max Sigma\_x = ",Max\_Sigmas\_values[0])  
print("max Sigma\_Y = ",Max\_Sigmas\_values[1])  
print("max Tay\_xy = ",Max\_Sigmas\_values[2])  
  
Sigma\_max\_global = ((Max\_Sigmas\_values[0]\*\*2) + (Max\_Sigmas\_values[1]\*\*2) - Max\_Sigmas\_values[0]\*Max\_Sigmas\_values[1] + 3\*(Max\_Sigmas\_values[2]\*\*2))\*\*(1/2)  
  
print("Sigma max global = ",Sigma\_max\_global)