Санкт-Петербургский политехнический университет Петра Великого

Физико-Механический Институт

Высшая школа механики и процессов управления

**“Создание программы для расчета жестко защемленных прямоугольных пластин под действием произвольно распределенной нагрузки.”**

Выполнил студент группы:

5031503/90401

Дроздов Святослав

Санкт-Петербург

05.11.22

**Оглавление**

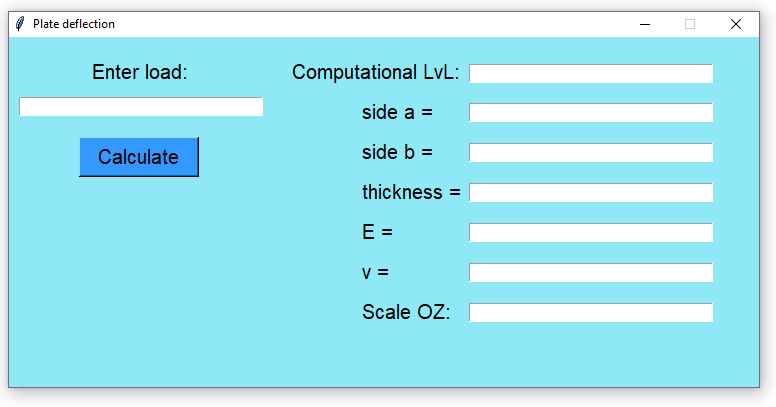
1. Введение ………………………………………………………………………………………………………………3
2. Общее описание программы Plate deflection………………………………………………………3
   1. Интерфейс программы Plate deflection……..…………………………………………………..3
   2. Результаты работы программы Plate deflection……………………………………………..5
3. Проверка вычислений на задачах с имеющимся аналитическим решением…….6
   1. Моделирование распределенной нагрузки……………………………………………………6
   2. Моделирование точечной нагрузки………………………………………………………………..8
4. Иллюстрация различных изгибов пластин…………………………………………………………….10
   1. Нагрузка с двумя пиками………………………………………………………………………………….10
   2. Возрастающая по ОХ нагрузка………………………………………………………………………….11
5. Математическая модель…………………………………………………..……………………………………12
6. Программный код………………………………………………………………..…………………………………14
   1. Window.py………………………………………………………………………..………………………………..14
   2. Plate\_deflection.py………………………………………………………………………………………………16
   3. Core.py…………………………………..………………………………………………..………………………….17
7. Заключение…………………………………………………..………………………………………………………….19
8. Источники информации………………………..…………..…………………………………………………….20

**0. Введение.**

Основной целью данной работы является создание программы, которая бы могла рассчитывать прогибы жестко защемленных пластин под действием произвольно-распределенных нагрузок. Данная цель выбрана не случайно: её достижение позволяет разобраться с применением метода Ритца приближенного построения решений в задаче об изгибе пластины, а также несколько углубиться в умение программировать.

1. **Общее описание программы Plate deflection:**
   1. **Интерфейс программы Plate deflection**

На Рис. 1 представлен интерфейс созданной программы. Далее, будем называть ее “Plate deflection”. Отметим, что при вводе физических величин следует использовать систему единиц СИ.

****

*Рис. 1 Интерфейс программы Plate deflection*

На Рис. 2 представлена изображение иконки *Plate deflection* на рабочем столе.

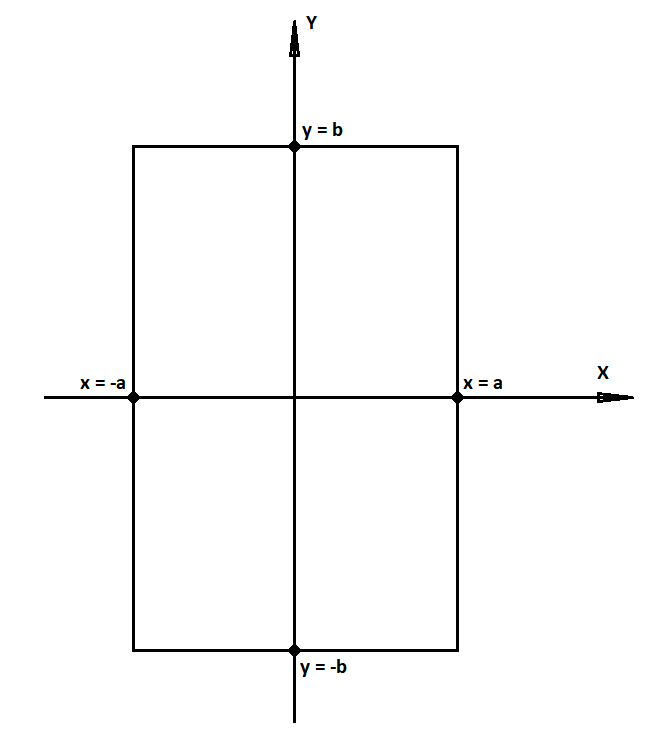


*Рис. 2 Изображение иконки программы Plate deflection*

*Руководство по использованию программы* “Plate deflection”:

1. Используемая система координат:

На Рис. 3 представлена система координат, которая используется при вводе значений в окна программы. Начало это системы координат располагается по центру пластины, а ее размеры составляют по оси OX и по оси OY.



*Рис. 3 Система координат.*

1. Окно “Enter load”

В этом окне вводится функция распределенной нагрузки. Для ввода используются обозначения:

1. Окно “Computational LvL”

В этом окне вводится натуральное число, называемое вычислительным уровнем. Вычислительный уровень отвечает за количество базисных функций, используемых в численном моделирование. Увеличение уровня на 1 отвечает увеличению числа базисных функций на 5.

Рекомендуется выбирать 1,2 или вычислительный уровень. (Увеличение вычислительного уровня позволяется добиться лучшей точности, но влечет значительнее увеличение вычислительного времени)

1. Окно “side a”

В окне “side a” задается параметр “a” равный половине измерения пластины по оси OX. (см. Рис.3)

1. Окно “side b”

В окне “side b” задается параметр “b” равный половине измерения пластины по оси OY. (см. Рис.3)

1. Окно “thickness”

В окне “thickness” задается толщина пластины.

1. Окно “E”

В окне “E” задается модуль Юнга материала пластины.

1. Окно “”

В окне “” задается коэффициент Пуассона материала пластины.

1. Окно “Scale OZ”

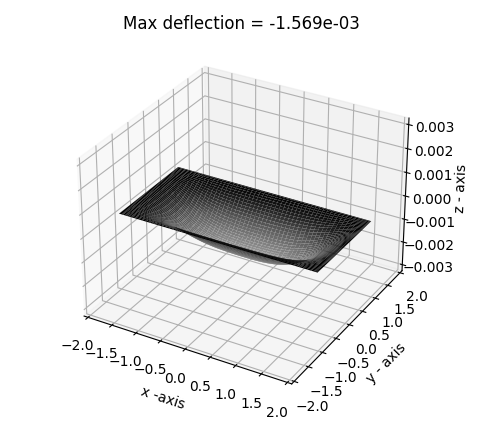
Заполнение окна “Scale OZ” не обязательно. По умолчанию программа будет масштабировать изображение изогнутой пластины так, что по оси OZ будет помещаться отрезок длиной в , где -максимальный расчетный прогиб. (масштабирование симметрично, относительно плоскости )

В окно “Scale OZ” можно ввести действительное положительное число. Если это число было введено, тогда масштабирование будет производиться так, чтобы масштаб по оси OZ вмещал 2 отрезка указанной длины, симметрично относительно уровня .

* 1. **Результаты работы программы Plate deflection**

После заполнения окон, для получения результатов, требуется нажать кнопку “Calculate” на темно голубом фоне. В качестве результатов программа выдает изображение искривленной пластины и значение максимального прогиба.

На Рис. 4 представлен пример результата расчетов, для пластины с параметрами:



*Рис. 4 Результат работы Plate deflection*

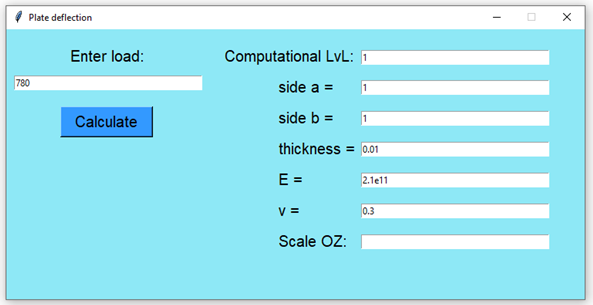
1. **Проверка вычислений на задачах с имеющимся аналитическим решением.**

Для того, чтобы показать работоспособность созданной программы, рассмотрим две задачи с известным аналитическим решением и сравним значения максимальных прогибов.

* 1. **Моделирование распределенной нагрузки.**

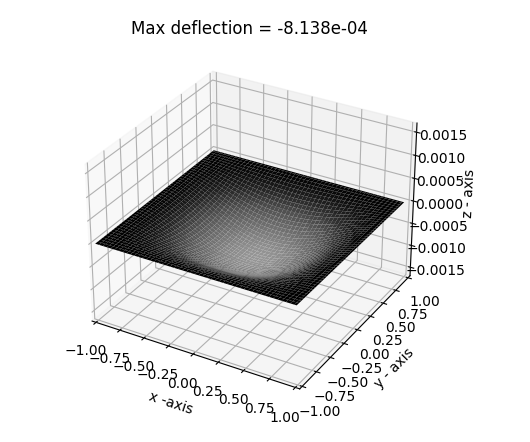
Зададим распределенную нагрузку в виде поверхностной плотности гравитационных сил, для стальной пластинки.

Используемые значения приведены на Рис.5



*Рис. 5 Постановка задачи прогиба пластины с распределёнными силами*

На Рис.6 представлены результаты вычислений.



*Рис. 6 Решение задачи прогиба пластины с распределёнными силами*

Приближенное значение прогиба составляет:

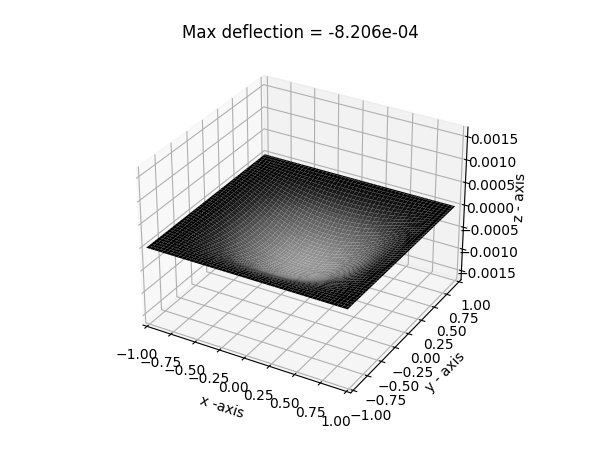
Аналитическое значение прогиба составляет: [1]

Погрешность:

Заметим, что, повысив вычислительный уровень до 2 (увеличив количество базисных функций), получим более точный результат. (См. Рис. 7)

Как видим, максимальный прогиб теперь равен:

И погрешность уменьшается до:

**

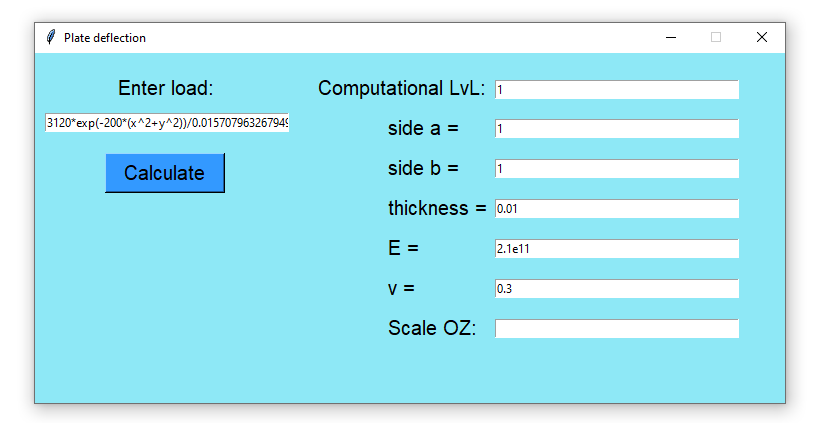
*Рис. 6 Решение задачи прогиба пластины с распределёнными силами Computational LvL =2*

* 1. **Моделирование точечной нагрузки.**

Зададим непрерывное поле функцией, имеющей большие значения в центре пластины, которые спадают, практически до нуля, в удаление от центра.

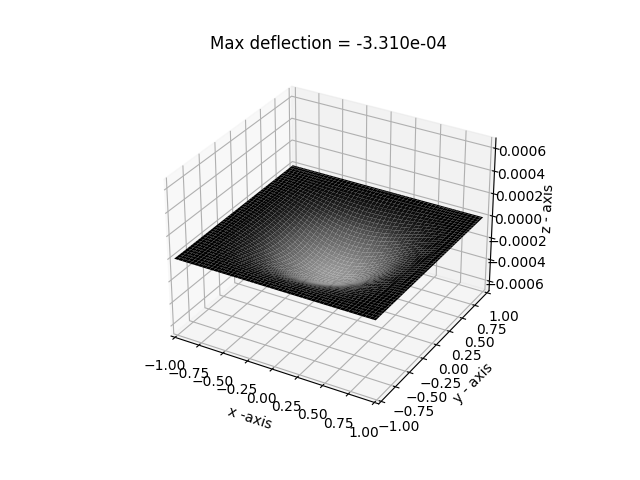
Так же, для удобства сравнения, выберем эту функцию так, чтобы полная сила , действующая на пластину, была такая же, как в предыдущем пункте. То есть, равнялась силе тяжести, действующую на стальную пластину.

Интеграл по площади пластины от равен 3120. Что соответствует силе тяжести, сосредоточенной в одной точке. Полная постановка задачи представлена на Рис. 7.



*Рис. 7 Постановка задачи прогиба пластины с сосредоточенной силой*

На Рис.8 представлены результаты вычислений.



*Рис. 8 Решение задачи прогиба пластины с сосредоточенными силами*

Приближенное значение прогиба составляет:

Аналитическое значение прогиба составляет: [1]

Погрешность:

Наличие погрешности объясняется следующими причинами: ошибки при численном дифференцировании, ошибки при численном интегрировании, отличие рассматриваемой функции от дельта функции Дирака. Также, к причинам появления высокой погрешности, можно добавить то, что, по всей видимости, используемые базисные функции недостаточно хороши, для задач с точечным нагружением.

**Выводы:** как видим, Plate deflection решает задачи с распределенной нагрузкой с высокой точностью. Чего нельзя сказать о задачах с точечной нагрузкой. Однако, для оценки решения с погрешностью около 10%, Plate deflection можно использовать, в том числе и для задач с точечными нагрузками.

1. **Иллюстрация различных изгибов пластин.**

Для того, чтобы показать, возможности Plate deflection представим решение пары задач со специфическим нагружением.

* 1. **Нагрузка с двумя пиками**

Зададим функцию распределенной нагрузки так, чтобы у нее было 2 пика:

Остальные параметры указаны на Рис. 9

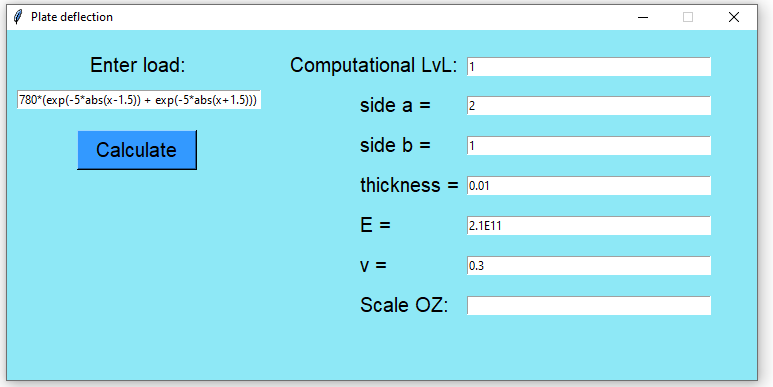
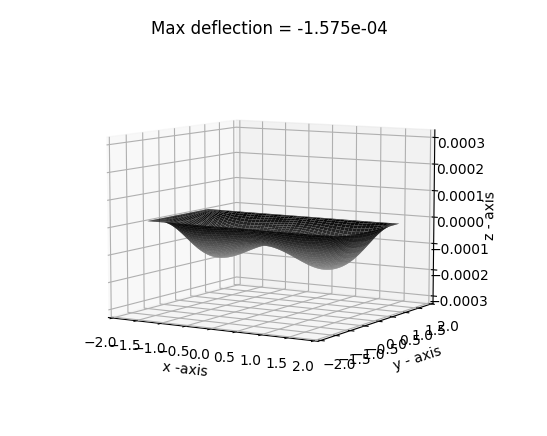


Рис. 9 *Постановка задачи прогиба пластины с распределенной силой, имеющей 2 пика*



*Рис. 10 Решение задачи прогиба пластины с распределенной силой, имеющей 2 пика*

* 1. **Возрастающая по OX нагрузка**

Зададим функцию распределенной нагрузки

Остальные параметры указаны на Рис. 11

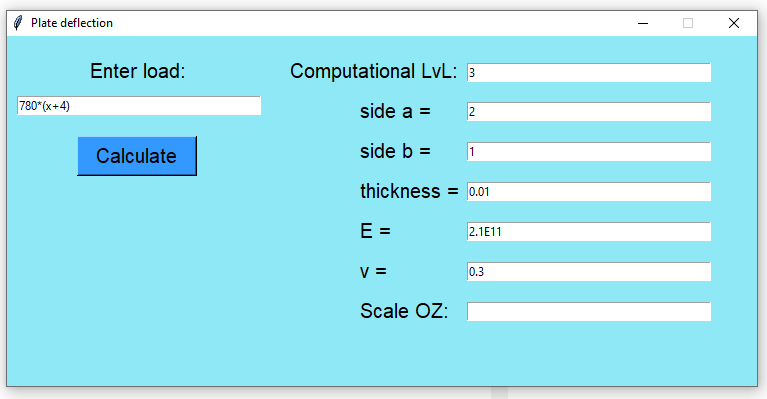
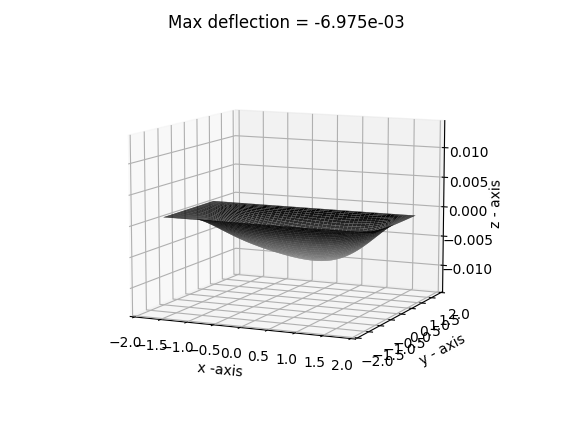
**

Рис. 11 *Постановка задачи прогиба пластины с возрастающей по ОХ нагрузкой*

**

*Рис. 12 Решение задачи прогиба пластины с возрастающей по ОХ нагрузкой*

**Выводы:** как видим, Plate deflection строит решения, физически адекватные нагрузке. Что позволяет использовать Plate deflection не только в случаях осесимметричных нагрузок, но и в случае произвольных нагрузок.

1. **Математическая модель**

Рассмотрим задачу прогиба прямоугольных пластин с граничными условиями жесткого закрепления. (Здесь и далее используется система координат, приведенная на Рис.3)

Рассмотрим базисные функции , удовлетворяющие этим условиям. (Конкретные функции будут выбраны позже)

Представим прогиб в виде линейной комбинации:

Запишем работу внешних сил на заданных перемещениях:

Вычислим потенциальную энергию деформации:

Полная энергия:

Условия минимума полной энергии:

Таким образом, получаем СЛАУ:

(В дальнейшем, говоря о СЛАУ, будем иметь в виду именно это выражение)

В качестве базисных функций будем брать:

Видно, что при .

Рассмотрим производную по x:

Так как:

Аналогично устанавливается:

Представленные базисные функции позволяют рассчитывать не симметричные нагрузки, за счет наличия экспонент.

1. **Программный код**

Программный код Plate deflection разбит на 3 основные задачи: создание интерфейса, визуализация трехмерной модели пластины и вычисление прогибов. В соответствие с этими задачами написаны 3 файла:

1. *Window.py – создание интерфейса*
2. *Plate\_deflection.py – визуализация трехмерной модели пластины*
3. *Core.py – вычисление прогибов*

Взаимодействие этих файлов следующее:

Файл *Window.py* создает интерфейс и обращается к *Plate\_deflection.py* после нажатия кнопки “*Calculate*”, передавая ему информацию, введённую пользователем. В свою очередь, *Plate\_deflection.py,* для визуализации прогибов, обращается за значениями прогибов к *Core.py,* которому передает значения, полученные до этого от *Window.py.*

**5.1 Window.py**

Числа N и M, заданные в начале, используются в *Core.py* для численного интегрирования и дифференцирования. Значение 100 выбрано исходя из того, что оно позволяет получать достаточную точность при не слишком большом времени вычислений. (Подобранно экспериментальным путем).

Функция ***Reading()*** *позволяется считывать информацию от пользователя.*

*Функция* ***transformation()*** *используется, что бы конвертировать введенные пользователем данные, в математические функции, используемые в python.*

import tkinter as tk  
import Plate\_deflection  
import math as m  
  
# These numbers are used for calculation of derivatives and integrals  
N = 100  
M = 100  
  
  
def transformation(Str):  
 St = Str.replace("^", "\*\*")  
 St = St.replace("sin", "m.sin")  
 St = St.replace("cos", "m.cos")  
 St = St.replace("exp", "m.exp")  
 return St  
  
  
def Reading():  
 Load = transformation(EnterLoad.get())  
 LvL = int(EnterLvL.get())  
 a = float(EnterOX.get())  
 b = float(EnterOY.get())  
 Thick = float(EnterThick.get())  
 Young = float(EnterYoung.get())  
 Nu = float(EnterNu.get())  
 try:  
 Scale = float(EnterScale.get())  
 except:  
 Scale = ""  
  
 exec('Q = lambda x,y:' + Load, globals())  
  
 Num = 5 \* LvL  
 dx = 2 \* a / N  
 dy = 2 \* b / M  
 D = Young \* Thick \*\* 3 / (12 \* (1 - Nu \*\* 2))  
 Plate\_deflection.DEFLECTION(a, b, Q, N, M, dx, dy, D, LvL, Num, Scale)  
  
  
window = tk.Tk()  
  
window.title("Plate deflection")  
w = 750  
h = 350  
  
window.geometry(f"{w}x{h}+600+300") # f-string  
window.resizable(False, False)  
window.config(bg='#8EE8F6') # RGB - color codes  
  
way = tk.Label(window, text="Enter load:", font=("Arial Bold", 15), fg="Black", bg='#8EE8F6')  
way.place(x=80, y=20)  
  
wid = 40  
EnterLoad = tk.Entry(fg="black", width=wid)  
EnterLoad.place(x=10, y=60)  
  
btn = tk.Button(window, text="Calculate", font=("Arial Bold", 15), command=Reading, width=int(wid / 4), height=" 1",  
 fg="Black",  
 bg='#3399FF')  
btn.place(x=70, y=100)  
  
# Title start  
lvl = tk.Label(window, text="Computational LvL:", font=("Arial Bold", 15), fg="Black", bg='#8EE8F6')  
lvl.place(x=280, y=20)  
EnterLvL = tk.Entry(fg="black", width=wid)  
EnterLvL.place(x=460, y=27)  
# Title end  
  
# OX and OY  
OXl = tk.Label(window, text="side a =", font=("Arial Bold", 15), fg="Black", bg='#8EE8F6')  
OXl.place(x=350, y=60)  
EnterOX = tk.Entry(fg="black", width=wid)  
EnterOX.place(x=460, y=66)  
OYl = tk.Label(window, text="side b =", font=("Arial Bold", 15), fg="Black", bg='#8EE8F6')  
OYl.place(x=350, y=60 + 40)  
EnterOY = tk.Entry(fg="black", width=wid)  
EnterOY.place(x=460, y=66 + 40)  
# OX and OY end  
  
# thickness start  
thick = tk.Label(window, text="thickness =", font=("Arial Bold", 15), fg="Black", bg='#8EE8F6')  
thick.place(x=350, y=60 + 80)  
EnterThick = tk.Entry(fg="black", width=wid)  
EnterThick.place(x=460, y=66 + 80)  
# end thickness  
  
# young start  
young = tk.Label(window, text="E =", font=("Arial Bold", 15), fg="Black", bg='#8EE8F6')  
young.place(x=350, y=60 + 120)  
EnterYoung = tk.Entry(fg="black", width=wid)  
EnterYoung.place(x=460, y=66 + 120)  
# end young  
  
# Nu start  
nu = tk.Label(window, text="ν =", font=("Arial Bold", 15), fg="Black", bg='#8EE8F6')  
nu.place(x=350, y=60 + 120 + 40)  
EnterNu = tk.Entry(fg="black", width=wid)  
EnterNu.place(x=460, y=66 + 120 + 40)  
# end Nu  
  
# ScaleZ start  
scale = tk.Label(window, text="Scale OZ:", font=("Arial Bold", 15), fg="Black", bg='#8EE8F6')  
scale.place(x=350, y=60 + 120 + 80)  
EnterScale = tk.Entry(fg="black", width=wid)  
EnterScale.place(x=460, y=66 + 120 + 40 + 40)  
# end Result  
  
window.mainloop()

**5.2 Plate\_deflection**

Функция **Fi(i\_n,iX,iY)** вместе с импортированным из списком **C** решения СЛАУ, задает значения координат Z точек поверхности деформированной пластины.

import Core  
import math as m  
import matplotlib.pyplot as plt  
from matplotlib import cm  
import numpy as np  
  
  
def DEFLECTION(a, b, Q, N, M, dx, dy, D, LvL, Num, Scale):  
 C = Core.core\_calculation(a, b, Q, N, M, dx, dy, D, LvL, Num)  
 # Make data  
 Num = 5 \* LvL  
 DX = 2 \* a / 100  
 DY = 2 \* b / 100  
 X = np.arange(-a, a, DX)  
 Y = np.arange(-b, b, DY)  
 X, Y = np.meshgrid(X, Y)  
  
 def Fi(i\_n, iX, iY):  
 if 1 <= i\_n <= LvL:  
 f = (1 + np.cos((2 \* i\_n - 1) \* m.pi \* iX / a)) \* (1 + np.cos((2 \* i\_n - 1) \* m.pi \* iY / b))  
 elif LvL + 1 <= i\_n <= 2 \* LvL:  
 f = (1 + np.cos(m.pi \* iX / a)) \* np.exp((i\_n - LvL) \* iX) \* (1 + np.cos(m.pi \* iY / b)) \* np.exp(  
 (i\_n - LvL) \* iY)  
 elif 2 \* LvL + 1 <= i\_n <= 3 \* LvL:  
 f = (1 + np.cos(m.pi \* iX / a)) \* np.exp(-(i\_n - 2 \* LvL) \* iX) \* (1 + np.cos(m.pi \* iY / b)) \* np.exp(  
 -(i\_n - 2 \* LvL) \* iY)  
 elif 3 \* LvL + 1 <= i\_n <= 4 \* LvL:  
 f = (1 + np.cos(m.pi \* iX / a)) \* np.exp(-(i\_n - 3 \* LvL) \* iX) \* (1 + np.cos(m.pi \* iY / b)) \* np.exp(  
 +(i\_n - 3 \* LvL) \* iY)  
 elif 4 \* LvL + 1 <= i\_n <= 5 \* LvL:  
 f = (1 + np.cos(m.pi \* iX / a)) \* np.exp(+(i\_n - 4 \* LvL) \* iX) \* (1 + np.cos(m.pi \* iY / b)) \* np.exp(  
 -(i\_n - 4 \* LvL) \* iY)  
 else:  
 f = 0  
 print("ERROR")  
 return f  
  
 Z = C[0] \* Fi(1, X, Y)  
 if Num > 1:  
 for i in range(2, Num + 1):  
 Z += C[i - 1] \* Fi(i, X, Y)  
  
 Max\_Deflection = min(np.amin(Z, axis=0))  
  
 # Plot the surface  
  
 fig, ax = plt.subplots(subplot\_kw={"projection": "3d"})  
 ax.plot\_surface(X, Y, Z, vmin=Z.min() \* 2, cmap=cm.Greys)  
  
 ax.set\_xlabel('x -axis')  
 ax.set\_ylabel('y - axis')  
 ax.set\_zlabel('z - axis')  
 dimensions = [a, b]  
 ax.set\_xlim(-max(dimensions), max(dimensions))  
 ax.set\_ylim(-max(dimensions), max(dimensions))  
 if Scale == "":  
 ax.set\_zlim(2 \* Max\_Deflection, -2 \* Max\_Deflection)  
 else:  
 ax.set\_zlim(-Scale, Scale)  
  
 plt.title("Max deflection = " + f"{format(Max\_Deflection, '.3e')}")  
 plt.show()

**5.3 Core.py**

Функция **Fi(i\_n,iX,iY)** задает базисные функции. Функции **dxFi, dyFi, d2xFi, d2yFi** считают соответствующие частные производные базисных функций. Функция **F(ik)** находит значения вектора из правой части СЛАУ. Функция **R(ik, i\_n)** находит значения коэффициентов СЛАУ.

import math as m  
import numpy  
  
  
def core\_calculation(a, b, Q, N, M, dx, dy, D, LvL, Num):  
 def Fi(i\_n, ix, iy):  
 if 1 <= i\_n <= LvL:  
 f = (1 + m.cos((2 \* i\_n - 1) \* m.pi \* ix / a)) \* (1 + m.cos((2 \* i\_n - 1) \* m.pi \* iy / b))  
 elif LvL + 1 <= i\_n <= 2 \* LvL:  
 f = (1 + m.cos(m.pi \* ix / a)) \* m.e \*\* ((i\_n - LvL) \* ix) \* (  
 1 + m.cos(m.pi \* iy / b)) \* m.e \*\* ((i\_n - LvL) \* iy)  
 elif 2 \* LvL + 1 <= i\_n <= 3 \* LvL:  
 f = (1 + m.cos(m.pi \* ix / a)) \* m.e \*\* (-(i\_n - 2 \* LvL) \* ix) \* (  
 1 + m.cos(m.pi \* iy / b)) \* m.e \*\* (-(i\_n - 2 \* LvL) \* iy)  
 elif 3 \* LvL + 1 <= i\_n <= 4 \* LvL:  
 f = (1 + m.cos(m.pi \* ix / a)) \* m.e \*\* (-(i\_n - 3 \* LvL) \* ix) \* (  
 1 + m.cos(m.pi \* iy / b)) \* m.e \*\* (+(i\_n - 3 \* LvL) \* iy)  
 elif 4 \* LvL + 1 <= i\_n <= 5 \* LvL:  
 f = (1 + m.cos(m.pi \* ix / a)) \* m.e \*\* (+(i\_n - 4 \* LvL) \* ix) \* (  
 1 + m.cos(m.pi \* iy / b)) \* m.e \*\* (-(i\_n - 4 \* LvL) \* iy)  
 else:  
 f = 0  
 print("ERROR")  
 return f  
  
 def dxFi(i\_n, ix, iy):  
 dxf = (Fi(i\_n, ix + dx, iy) - Fi(i\_n, ix - dx, iy)) / (2 \* dx)  
 return dxf  
  
 def dyFi(i\_n, ix, iy):  
 dyf = (Fi(i\_n, ix, iy + dy) - Fi(i\_n, ix, iy - dy)) / (2 \* dy)  
 return dyf  
  
 def d2xFi(i\_n, ix, iy):  
 d2xf = (dxFi(i\_n, ix + dx, iy) - dxFi(i\_n, ix - dx, iy)) / (2 \* dx)  
 return d2xf  
  
 def d2yFi(i\_n, ix, iy):  
 d2yf = (dyFi(i\_n, ix, iy + dy) - dyFi(i\_n, ix, iy - dy)) / (2 \* dy)  
 return d2yf  
  
 def F(ik):  
 p = 0  
 for jj in range(0, M):  
 y = -b + jj \* dy  
 for ii in range(0, N):  
 x = -a + ii \* dx  
 p += Q(x, y) \* Fi(ik, x, y) \* dx \* dy  
 p = -p  
 return p  
  
 def R(ik, i\_n):  
 R\_kn = 0  
 for jj in range(0, M):  
 y = -b + jj \* dy  
 for ii in range(0, N):  
 x = -a + ii \* dx  
 R\_kn += (d2xFi(i\_n, x, y) + d2yFi(i\_n, x, y)) \* (d2xFi(ik, x, y) + d2yFi(ik, x, y)) \* dx \* dy  
 R\_kn = D \* R\_kn  
 return R\_kn  
  
 Matrix = []  
 for j in range(1, Num + 1):  
 Matrix.append([])  
 for i in range(1, Num + 1):  
 Matrix[j - 1].append(R(j, i))  
  
 V = []  
 for j in range(1, Num + 1):  
 V.append(F(j))  
  
 Ms = numpy.array(Matrix)  
 Vs = numpy.array(V)  
 C = numpy.linalg.solve(Ms, Vs)  
 return C

**6. Заключение:**

На языке python, с помощью метода Ритца построения приближенных решений, была создана программа Plate deflection, которая решает задачу о прогибе жестко защемленных прямоугольных пластин, под действием произвольных нагрузок, после чего визуализирует прогиб пластины, а также выводит значение максимального прогиба.

Применение Plate deflection, в случаях с известным аналитическим решением, показало, что расчет для задач с распределенной нагрузкой может давать результаты с высокой точностью, в то время как, при моделировании задач с сосредоточенными силами, могут возникать погрешности, вплоть до 10%.

**7. Источники информации**

1. Тимошенко С. П., Войновский-Кригер С. Пластинки и оболочки, 1966
2. Интернет ресурс: https://prosopromat.ru/teoriya-uprugosti/raschet-pryamougolnoj-plastinki-variacionnym-metodom-ritca-timoshenko.html
3. Конспект лекций “Строительная механика машин”, лектор: Беляев А. К.