I. Задача

На Рис. 1 изображена конструкция изгибного устройства. Необходимо найти точку на поверхности(Рис. 1), которая будет покоиться в пространстве в процессе изгиба.

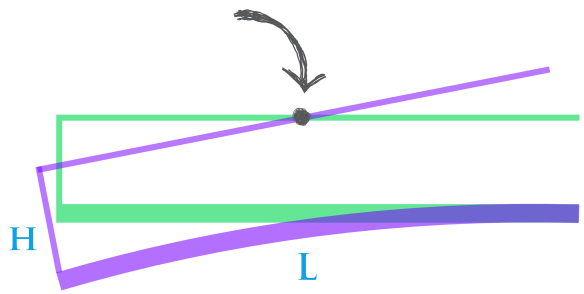

Рис. 1: Конструкция устройства

II. Формула

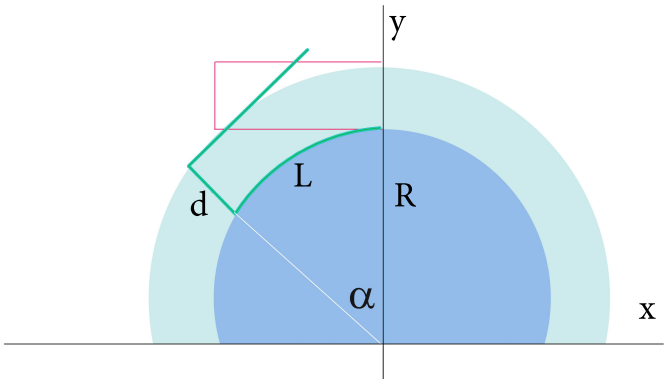

Рис. 2: к формуле

Для того, чтобы найти такую особую точку необходимо решить систему уравнений(1):

$$\begin{cases} y = d + R \\ y = x + (R + d) \cdot cos(a) \end{cases} \tag{1}$$

Приравняв правые части получим (2)

$$x = \frac{R}{L} \cdot (d + R)(1 - cos(a)) \tag{2}$$

Угол a (в радианах) и радиус R свяжем через дугу окружности:

$$R = \frac{L}{a}$$

Так же $cos(a)$ - можно разложить в ряд и ограничься первыми двумя членами, тогда(3):

$$x = \frac{L}{2} \tag{3}$$

III. Моделирование

Вкратце, для моделирования (точнее для визуализации) использовалась библиотека Matplotlib языка программирования Python. Использовался дискретный массив координат, от кадра к кадру все координаты точек переписывались снова, но с учетом их местонахождения в предыдущем кадре.

Система разделена на три составляющих, нижняя балка, перемычка и верхняя балка. Нижняя балка подвергается деформации изгиба, в этот момент ее можно представить в качестве дуги окружности. Окружность характеризуется радиусом или углом сектора, на длине дуги которого умещается длинна изгибного устройства L (Рис. 2).

Перемычка и верхняя балка соединены между собой жестко, всегда следуют вместе с крайней точкой изгибного кристалла. Так как все координаты заданы в явном виде, то угол поворота системы: (перемычка-верхняя балка), легко можно вычислить, учитывается что - это прямой угол к касательной в точке соединения или продолжение радиуса окружности в точку соединения(Рис. 2).

С результатами моделирования можно ознакомиться: https://youtu.be/Lh0ubmMHLqk (Рис. 3).

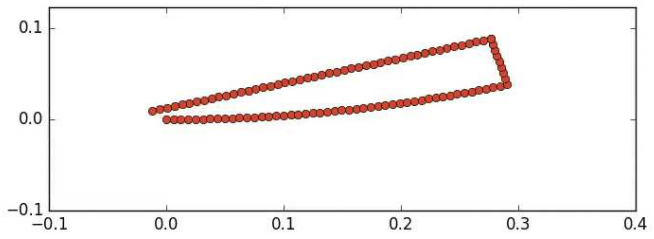

Рис. 3: Динамика изгибного устройства. L = 0.3 м, d = 0.05 м

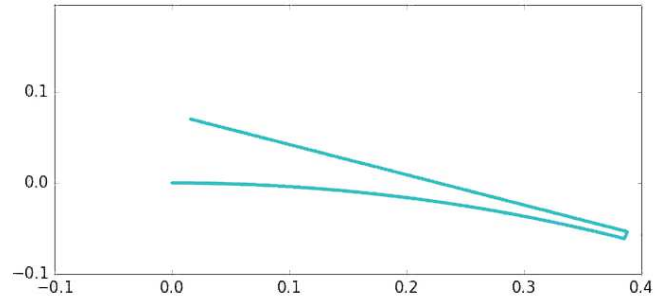

Рис. 4: Динамика изгибного устройства L = 0.4 м, d = 0.001 м

Для анализа модели, отбросим все ненужные части с поля зрения (изгибный и перемычку), оставим только необходимую для анализа верхнюю часть устройства.

Еще при каждом построение не будем убирать предыдущий результат, таким образом получится как бы трек этого движения (Рис. 5).

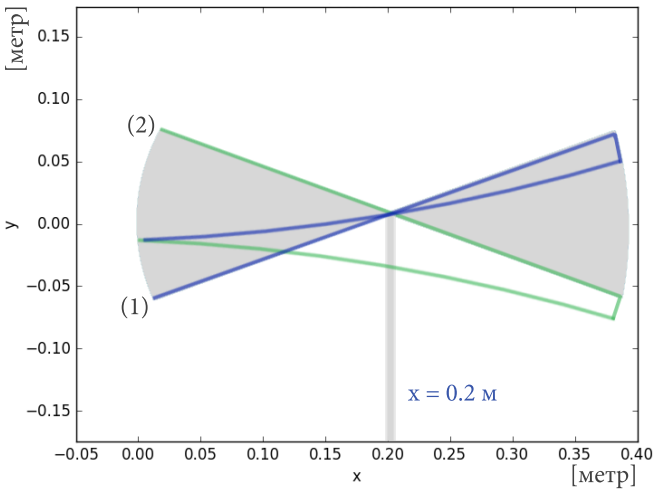

Рис. 5: Динамика изгибного устройства L = 0.4 м, d = 0.001 м, максимальный угол - 20^o

Если увеличивать угол изгиба кристалла вплоть до 50^o(Рис. 6), наблюдается “размывание” точки и смещение самого “узкого” участка вправо.

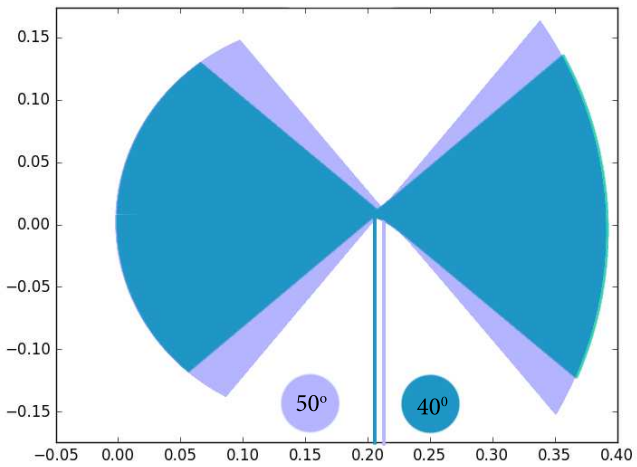

Рис. 6: Уширение “узкой” части и смещение вправо

Также можно привести картинку, расчитанную на основе формул во втором параграфе, без учета учета приближений(Рис. 6).

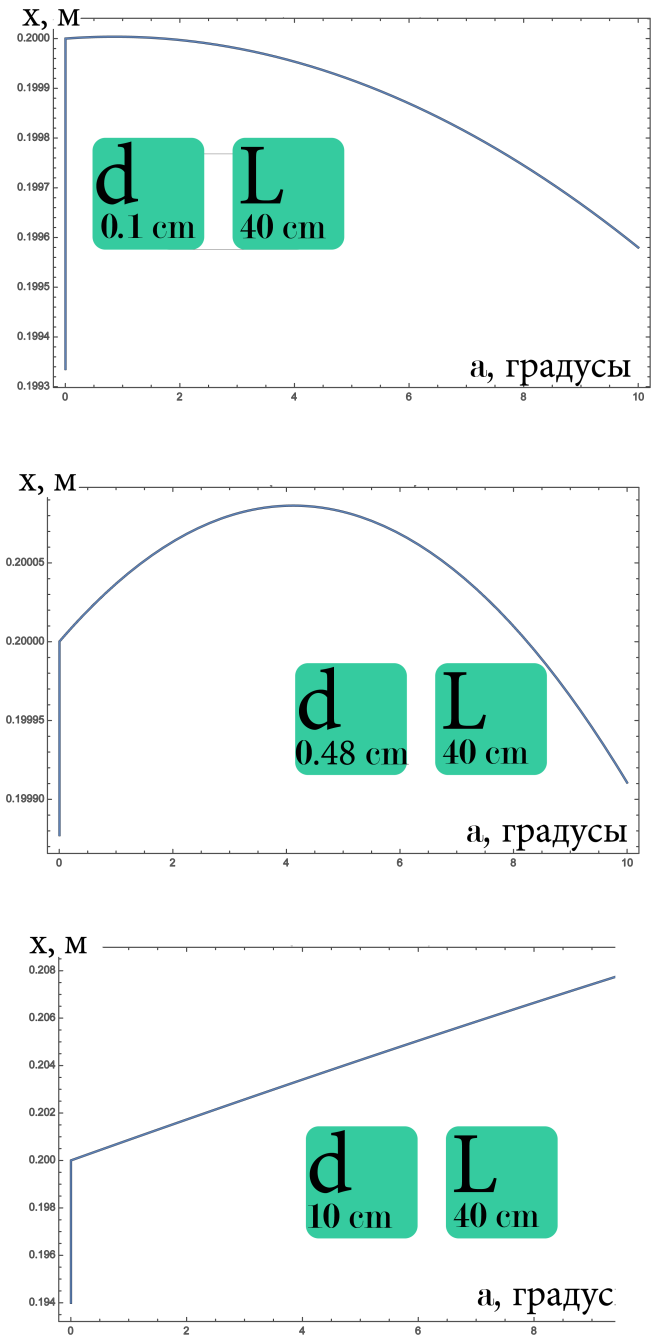

Рис. 7: Положение особой точки в зависимости от угла изгиба кристалла для разных размеров перемычки d

IV. ——————Listing (черновой вариант)——————

```

import numpy as np
from matplotlib import pyplot as plt
from matplotlib import animation
import math

hi_x = []
hi_y = []

# First set up the figure, the axis, and the plot element we want to animate
fig = plt.figure()

ax = plt.axes(xlim=(-0.1, 0.4), ylim=(0-0.1, 0.5-0.1))
line, = ax.plot([], [], lw=2)
x = []
y = []
ugol_m = []
N = 100
x0 = 0
y0 = 0
L = 40/100
d = 0.1/100
r = []
total = 2*L+d
shag = total/N
l = 0
while L>=l:
  l = l + shag
  x.append(x0)
  y.append(y0)
  x0 = x0+shag
  y0 = y0 +0

length1 = len(x)+1

d0 = 0
while d>=d0:
  d0 = d0 + shag
  x.append(x0)
  y.append(y0)
  x0 = x0+0
  y0 = y0 + shag

length2 = len(x)+1

l = 0
while L>=l:
  l = l + shag
  x.append(x0)
  y.append(y0)
  x0 = x0-shag
  y0 = y0 + 0

x.append(x0)
y.append(y0)

length3 = len(x)


for j in range(len(x)):

  if x[j] == 0 and y[j] < 0:
    ugol = 3*math.pi/2
  elif x[j] == 0 and y[j] > 0:
    ugol = math.pi/2
  elif y[j] == 0 and x[j] > 0:
    ugol = 2*math.pi
  elif y[j] == 0 and x[j] < 0:
    ugol = math.pi/2
  elif y[j] == 0 and x[j] == 0:
    ugol = math.pi/2
  else: ugol = math.atan(y[j]/x[j])
  r.append(math.sqrt(y[j]*y[j]+x[j]*x[j]))
  ugol_m.append(ugol)

# initialization function: plot the background of each frame
def init():
  line.set_data([], [])
  return line,

# animation function.  This is called sequentially
f_rame = 500
R = 10
alfa_0 = math.radians(-50)

def animate(i):
  alfa = alfa_0 - 2*alfa_0/f_rame*i
  if alfa == 0:
    pass
  else:

    j = 0
    while j < length1:
      c = L/alfa
      x[j] = (c*math.sin(j*L/c/length1))
      y[j] = (c*math.cos(j*L/c/length1)) - c
      j +=1

    alfa_new = math.pi/2 - math.atan((y[j-2]-y[j-1])/(x[j-2]-x[j-1]))
    x0 = x[j-1]
    y0 = y[j-1]
    r = 0
    while j < length2:
      r  = r + shag
      x[j] = x0 +((r)*math.sin(L/c))
      y[j] = y0 + ((r)*math.cos(L/c))
      j+=1

    x0 = x[j-1]
    y0 = y[j-1]
    r = 0
    while j < length3:
      x[j] = x0 + ((r)*math.sin(L/c-math.pi/2))
      y[j] = y0 + ((r)*math.cos(L/c-math.pi/2))
      hi_x.append(x[j])
      hi_y.append(y[j])
      r = r+shag
      j+=1


  line.set_data(x, y)
  return line,



anim = animation.FuncAnimation(fig, animate, init_func=None,repeat_delay=None, repeat=True,
                               frames=f_rame, interval=2000, blit=True)

anim.save('/Users/dns/Desktop/dynamic_images.mp4', fps=100)
plt.close()



plt.plot(hi_x,hi_y, 'b')
plt.title('Izgib point')
plt.axis('equal')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('/Users/dns/Desktop/dynamic_images.png', bbox_inches='tight')
plt.close()
```