Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»

кафедра физики

Индивидуальное задание №2 для группы 2370 по разделу

"Оптика"

Название: Искривление луча в оптическом канале

Фамилия И.О.: Аннеков К.А.
Вариант: 5
Преподаватель: Альтмарк А.М.
Крайний срок сдачи: 21.05

Санкт-Петербург

2023

Условия ИДЗ 2

Найти длину траектории светового луча S в прямолинейном оптоволоконном канале, Рис.1. Функцию распределения показателя преломления n1(y) по поперечной координате Y, начальный угол ввода луча α в волновод, длину канала L, радиус канала D можно взять в таблице 1. Ввод луча осуществляется из центральной части канала с координатой y=0. Параметры *L* и D даны в безразмерных координатах.

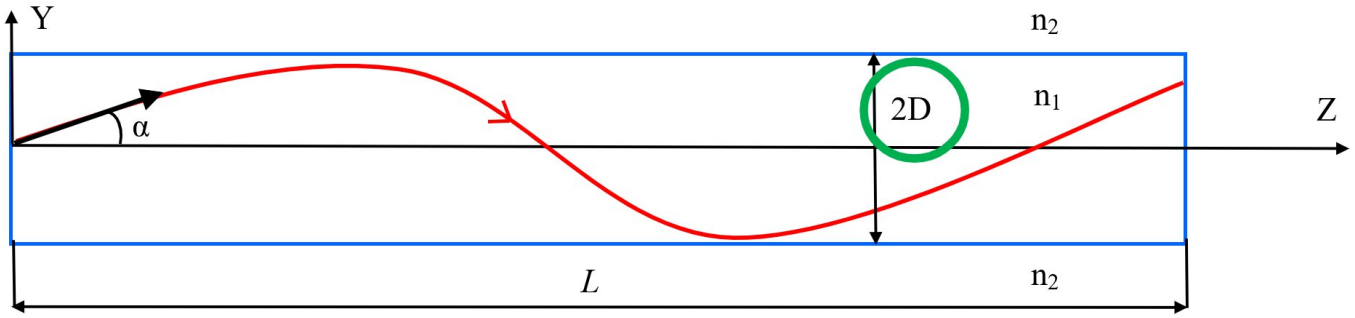


Рисунок.1

Таблица 1 Исходные данные

Вариант	*L*	D	n_2	$n_1(y)$	α, градусы
5	50	1,4	1	1.3 + 0.2*Cos[4*y]	10

Решение:

Для решения этой задачи мы разделим наше исследование на множество маленьких частей.

Это позволит нам более детально изучить каждую маленькую часть и понять, как она влияет на всю задачу. Такой подход помогает нам разобраться с каждым кусочком по отдельности и сделать выводы о всем исследовании в целом.

$$dl = \frac{L}{10000} = \frac{50}{10000}$$

У каждого отрезка, который мы разделили, есть свой треугольник. В этом треугольнике у нас есть известный угол α_1 (это угол в первом треугольнике) и одна сторона, которую мы называем катетом (это сам отрезок).

Используя эти данные, мы можем найти высоту треугольника, которую обозначим как y_1. Для этого мы умножаем тангенс угла α_1 на длину отрезка dl.

Таким образом, мы можем вычислить значение y_1, которое представляет собой высоту первого треугольника. Это поможет нам лучше понять свойства каждого отрезка и его влияние на всю задачу

$$y_1 = \tan\alpha_1 * dl$$

И сразу же начнем суммировать высоты всех треугольников:

$$Y_n = \sum_{i=1}^{n} y_i$$

Далее мы можем вычислить ds_1 (длину пути светового луча в первом треугольнике):

$$ds_1 = \sqrt{dl^2 + y_1^2}$$

Теперь нам нужно повторить эти действия для каждого треугольника, но для этого нам потребуется уравнение, которое поможет нам найти угол следующего треугольника, зная предыдущий угол:

$$\alpha_i = 90^o - \sin^{-1} * \left(\frac{n_{i-1} * (\sin 90^o - \alpha_{i-1})}{n_i} \right)$$

Для использования этого уравнения, надо найти n_i:

$$n_i = 1.3 - 0.2 * \cos(4 * y)$$

Теперь, когда у нас есть длины пути светового луча для всех треугольников, мы можем сложить их все вместе и получить их сумму:

$$S = \sum_{i=1}^{10000} ds_i$$

В данном алгоритме есть несколько случаев, которые требуют особого внимания:

1. В случае экстремумов функции, мы сталкиваемся с ситуацией, когда пытаемся вычислить арксинус числа, которое больше единицы. Такое вычисление невозможно. Поэтому в экстремумах нам нужно вручную определить угол следующего треугольника:

- Если предыдущий угол обозначен как a_{i-1}, то угол следующего треугольника будет равен отрицательному значению предыдущего угла:

$$a_i = -a_{i-1}.$$

2. Если предыдущий угол отрицателен, то алгоритм по умолчанию будет считать положительный угол для следующего треугольника. Однако, в данном случае нам нужно изменить это поведение. Если предыдущий угол отрицателен ($a_{i-1} < 0$), то угол следующего треугольника будет равен отрицательному значению предыдущего угла:

$$a_i = -a_{i-1}.$$

3. Когда световой луч достигает стенки оптоволокна, возникает экстремум. Мы должны обработать эту ситуацию аналогично первому случаю и вручную определить угол следующего треугольника.

Эти исключения позволяют нам правильно обработать специальные случаи и получить корректные результаты при использовании алгоритма.

Ответ: S = 51.5472500424130

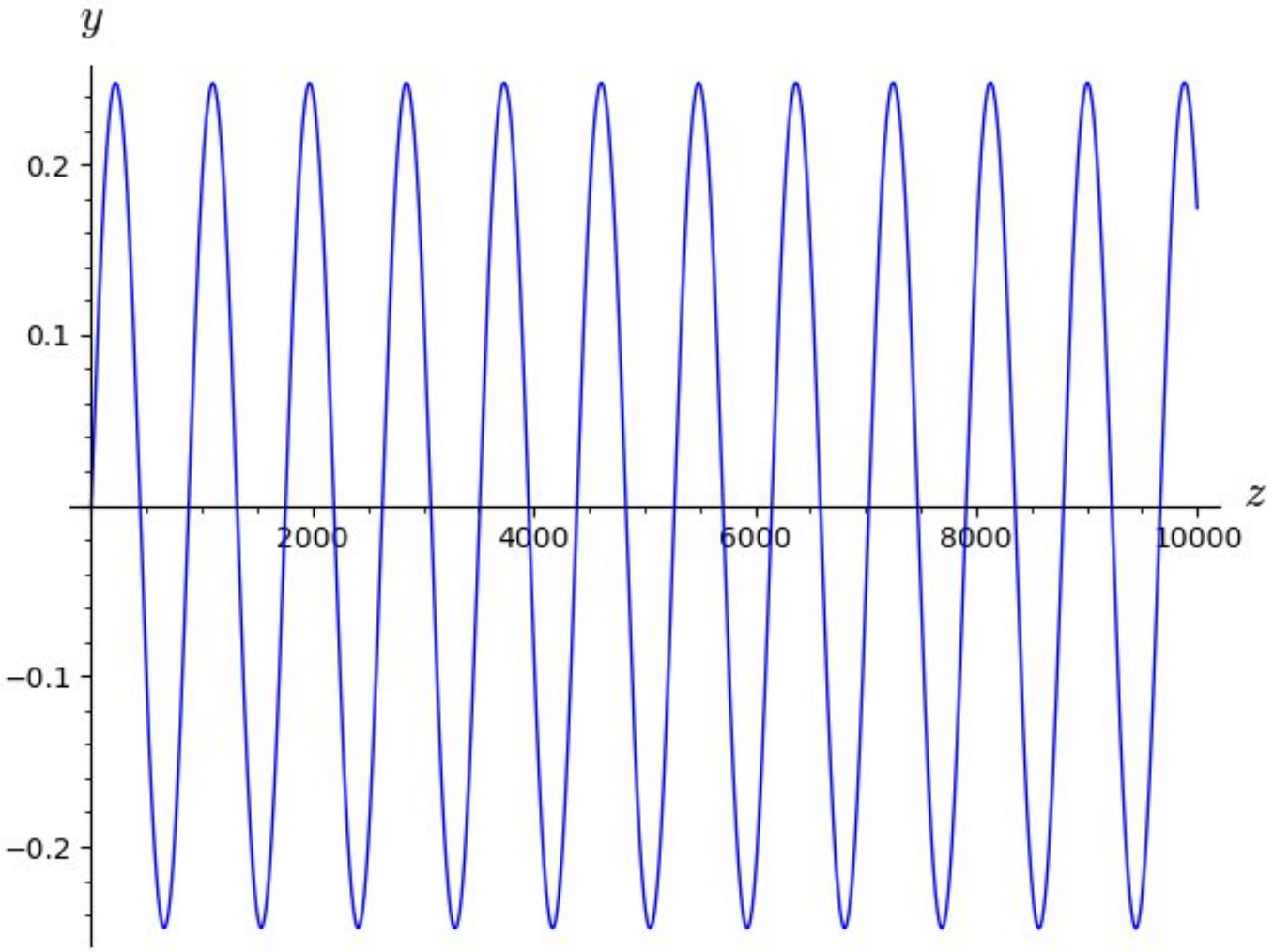
График смоделированный в sagemath:

```
def grad_to_rad(x):
    return (x*n(pi))/180

def find_N(x):
    return 1.3 + 0.2*cos(4*x)

def find_y(x):
    return tan(x)*dl
def find_ds(x):
    return sqrt(dl**2 + y[x]**2)
y = []
Y = []
alpha = []
N = []
alpha.append(grad_to_rad(20))
L = 50
step = 10000
dl = L/step
D = 1.4
y.append(0)
Y.append(y[0])
N.append( find_N(Y[0]))
S = find_ds(0)
max = 0
for i in range(1, step+1):
    y.append(find_y(alpha[i-1]))
    Y.append(Y[i-1]+y[i])
    N.append( find_N( Y[i] ) )
    alpha.append( grad_to_rad(90) - arcsin((N[i-1] * sin(grad_to_rad(90) -
alpha[i-1])/(N[i]) )))
    if alpha[i-1] < 0:
        alpha[i] = - alpha[i]
    if alpha[i] != alpha[i] or abs(Y[i]) >= D:
        alpha[i] = -alpha[i-1]
    S += find_ds(i)
    if max < Y[i]:
        max = Y[i]
print(S)
print(max)
list_plot([Y[i] for i in range(step+1)], plotjoined=True, axes_labels=['$z$',
'$y$'])
```

S = 51.5472500424130


y
0.2
0.1
−0.1
−0.2
2000
4000
6000
8000
10000
z