Difraccion de la luz en una rendija

UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

ESCUELA PROFESIONAL DE QUIMICA

Curso: Introduccion a la Electricidad Y Magnetismo

Max Serrano Arostegui 20230115I

5 de diciembre de 2024

Índice

1. Objetivos de aprendizaje

- Explicar el fenómeno de la difracción y las condiciones en las que se observa
- Describir la difracción a través de una rendija
- Brindar la explicacion respectiva del simulador

2. Marco Teorico

Tras pasar por una abertura estrecha, una onda que se propaga en una dirección determinada tiende a dispersarse. Por ejemplo, las ondas sonoras que entran en una habitación a través de una puerta abierta pueden oírse, aunque el oyente se encuentre en una parte de la habitación donde la geometría de la propagación de los rayos dicta que solo debe haber silencio. Del mismo modo, las olas del mar que pasan por una abertura en un rompeolas pueden propagarse por toda la bahía en su interior. (Figura 4.2). La propagación y la curvatura de las ondas sonoras y oceánicas son dos ejemplos de difracción, que es la curvatura de una onda alrededor de los bordes de una abertura o un obstáculo, un fenómeno que presentan todos los tipos de ondas.

Figura 1: Debido a la difracción de las ondas, las olas del océano que entran por una abertura en un rompeolas pueden propagarse por toda la bahía. (crédito: modificación de los datos del mapa de Google Earth)

La difracción de las ondas sonoras nos resulta evidente porque las longitudes de onda en la región audible tienen aproximadamente el mismo tamaño que los objetos con los que se encuentran, condición que debe cumplirse para que los efectos de la difracción puedan observarse fácilmente. Dado que las longitudes de onda de la luz visible van aproximadamente de 390 a 770

nm, la mayoría de los objetos no difractan la luz de forma significativa. Sin embargo, se dan situaciones en las que las aberturas son lo suficientemente pequeñas como para que la difracción de la luz sea observable. Por ejemplo, si se colocan los dedos medio e índice juntos y se mira a través de la abertura a una bombilla, se puede ver un patrón de difracción bastante claro, que consiste en líneas claras y oscuras que corren paralelas a los dedos.

2.1. Difracción a través de una rendija

La luz que pasa a través de una rendija forma un patrón de difracción algo diferente a los formados por rendijas dobles o rejillas de difracción. La Figura 2 muestra un patrón de difracción por una rendija. Observe que el máximo central es mayor que los máximos a ambos lados y que la intensidad disminuye rápidamente a ambos lados. En cambio, una rejilla de difracción (Rejillas de difracción) produce líneas espaciadas uniformemente que se atenúan lentamente a ambos lados del centro.

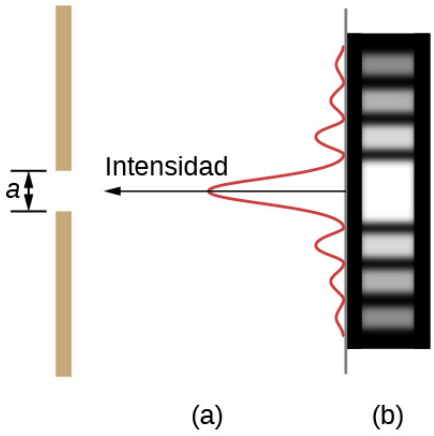

Figura 2: Patrón de difracción por una rendija. (a) La luz monocromática que pasa por una rendija tiene un máximo central y muchos máximos más pequeños y tenues a ambos lados. El máximo central es seis veces mayor que el mostrado. (b) El diagrama muestra el máximo central brillante, y los máximos más tenues y delgados a ambos lados.

En la Figura 3 se ilustra el análisis de la difracción de una rendija. En este caso, la luz llega a la rendija, la ilumina uniformemente y está en fase en todo su ancho. A continuación, consideramos la luz que se propaga hacia adelante desde distintas partes de la misma rendija. Según el principio de

Huygens, cada parte del frente de onda en la rendija emite ondas, tal y como comentamos en *La naturaleza de la luz*. Son como rayos que comienzan en fase y se dirigen en todas las direcciones. (Cada rayo es perpendicular al frente de onda de una ondícula.) Suponiendo que la pantalla esté muy lejos en comparación con el tamaño de la rendija, los rayos que se dirigen a un destino común son casi paralelos. Cuando se desplazan en línea recta, como en la parte (a) de la figura, permanecen en fase, y observamos un máximo central. Sin embargo, cuando los rayos viajan en ángulo θ respecto a la dirección original del haz, cada rayo recorre una distancia diferente hasta un lugar común, y pueden llegar en fase o fuera de fase. En la parte (b), el rayo de la parte inferior recorre una distancia de una longitud de onda λ más lejos que el rayo de la parte superior. Así, un rayo desde el centro recorre una distancia $\frac{\lambda}{2}$ menor que la del borde inferior de la rendija, llega desfasado e interfiere destructivamente. Un rayo procedente del centro y otro de la parte inferior también se anulan mutuamente. De hecho, cada rayo de la rendija interfiere destructivamente con otro rayo. En otras palabras, una cancelación por pares de todos los rayos da como resultado un mínimo oscuro de intensidad en este ángulo. Por simetría, se produce otro mínimo en el mismo ángulo a la derecha de la dirección de incidencia (hacia la parte inferior de la figura) de la luz.

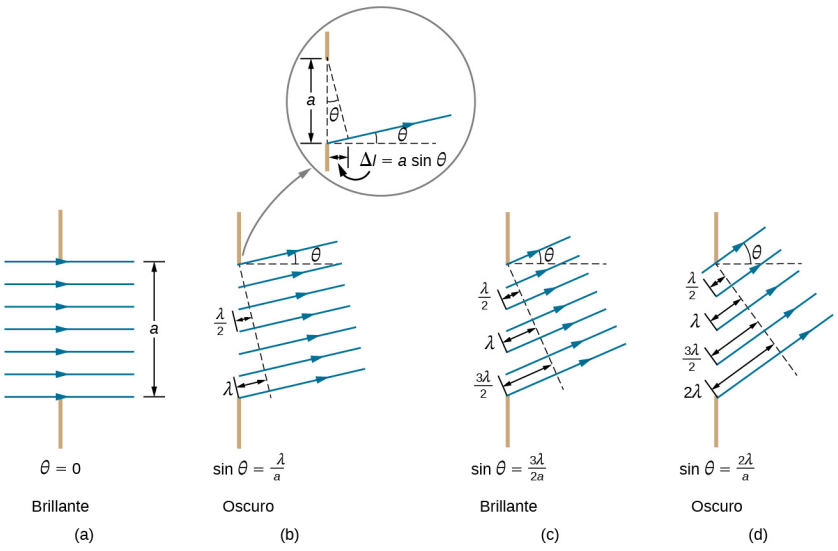

Figura 3: La luz que pasa por una rendija se difracta en todas las direcciones y puede interferir de forma constructiva o destructiva, según el ángulo. La diferencia en la longitud de la trayectoria de los rayos desde ambos lados de la rendija se ve como a $\sin\theta$.

En el ángulo mayor mostrado en la parte (c), las longitudes de las trayectorias difieren en $\frac{3\lambda}{2}$ para los rayos de la parte superior e inferior de la rendija. Un rayo recorre una distancia λ diferente del rayo del fondo y llega en fase, interfiriendo constructivamente. Dos rayos, cada uno de ellos ligeramente por encima de esos dos, también se suman constructivamente. La mayoría de los rayos procedentes de la rendija tienen otro rayo con el que interferir constructivamente, y en este ángulo se produce un máximo de intensidad.

Sin embargo, no todos los rayos interfieren constructivamente para esta situación, por lo que el máximo no es tan intenso como el máximo central. Finalmente, en la parte (d), el ángulo mostrado es lo suficientemente grande como para producir un segundo mínimo. Como se ve en la figura, la diferencia en la longitud de la trayectoria de los rayos desde ambos lados de la rendija es $a \sin\theta$, y vemos que se obtiene un mínimo destructivo cuando esta distancia es un múltiplo entero de la longitud de onda.

Así, para obtener una interferencia destructiva para una rendija,

$$a \sin\theta = m\lambda, \quad \text{donde } m = \pm 1, \pm 2, \pm 3, \ldots$$

donde a es el ancho de la rendija, λ es la longitud de onda de la luz, θ es el ángulo relativo a la dirección original de la luz, y m es el orden del mínimo. La Figura 4.5 muestra un gráfico de la intensidad para la interferencia de una rendija, y es evidente que los máximos a ambos lados del máximo central son mucho menos intensos y no tan amplios. Este efecto se explora en *Difracción de doble rendija.*

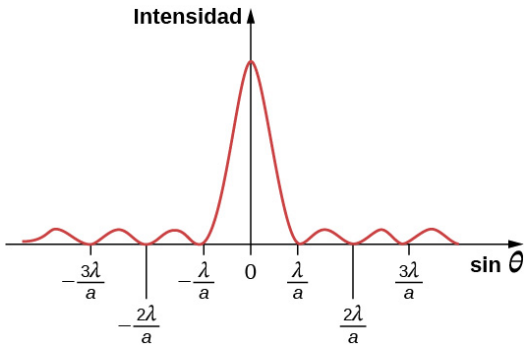

Figura 4: Un gráfico de la intensidad de difracción de una rendija que muestra que el máximo central es más amplio y mucho más intenso que los de los lados. De hecho, el máximo central es seis veces mayor que el mostrado aquí.

3. Simulador de la difraccion de la luz

3.1. Configuración del sistema óptico

El código simula un sistema óptico compuesto por:

- **Un láser** como fuente de luz monocromática.
- **Una lente** para enfocar el haz.
- **Una rendija** para generar difracción.
- **Una pantalla** donde se observa el patrón de difracción.

Este arreglo es fundamental para estudiar el fenómeno de **difracción de Fraunhofer**.

3.2. Propagación del haz láser

Se modela la trayectoria del haz láser a través del sistema óptico utilizando los principios de la óptica geométrica:

1. **Inicio del haz:** Desde el láser.
2. **Paso por la lente:** Cambia la dirección de propagación.
3. **Interacción con la rendija:** Se produce la difracción.
4. **Proyección en la pantalla:** Los rayos forman el patrón observable.

El método `dibujar_haz_laser()` calcula y visualiza esta trayectoria.

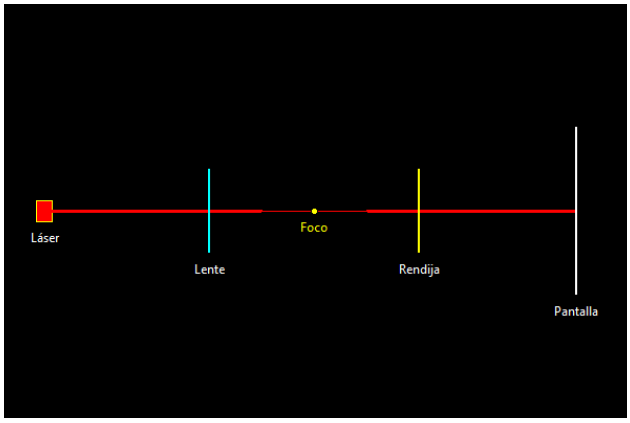

Figura 5: Montaje Experimental del Sistema Optico.

3.3. Cálculo del patrón de difracción

El método `calcular_patron_difraccion(y)` implementa la fórmula de **difracción de Fraunhofer** para una rendija única:

$$I(\theta) = I_0 \left(\frac{\sin(\beta)}{\beta} \right)^2$$

Donde:

- $I(\theta)$: Intensidad en función del ángulo de difracción (θ).
- I_0: Intensidad máxima.
- $\beta = \frac{\pi a \sin(\theta)}{\lambda}$.
- a: Ancho de la rendija.
- λ: Longitud de onda del láser.

Este cálculo es esencial para predecir cómo se distribuye la luz en la pantalla.

3.4. Visualización del patrón de difracción

El método `dibujar_patron_pantalla()` traduce los cálculos en una representación visual:

- Los colores en la pantalla simulan la intensidad.
- El rojo más brillante indica la máxima intensidad.

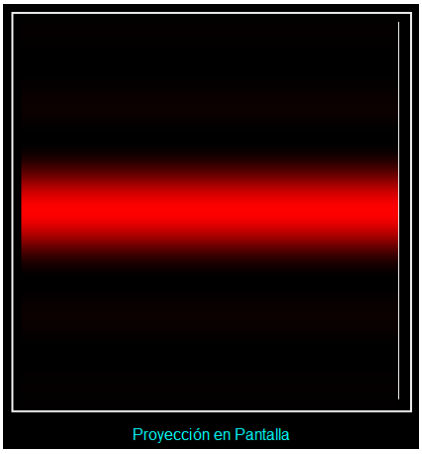

Figura 6: Visualización del laser en la pantalla.

3.5. Distribución de intensidad

El método `dibujar_distribucion_intensidad()` genera una gráfica que muestra cómo varía la intensidad en la pantalla:

- **Eje horizontal:** Posición en la pantalla.
- **Eje vertical:** Intensidad relativa.

Esto complementa la visualización del patrón con un análisis cuantitativo.

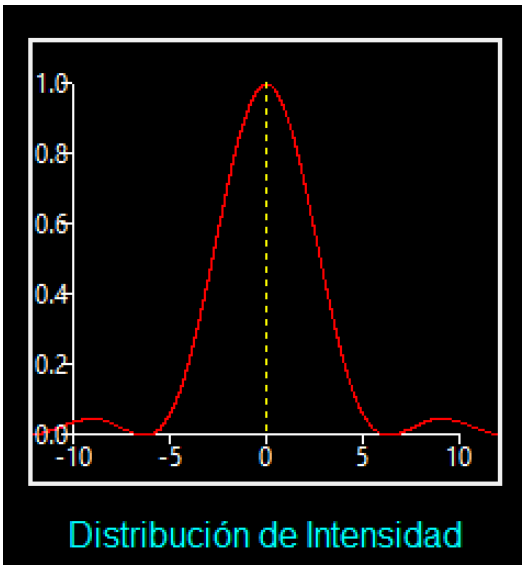

Figura 7: Distribución de la Intensidad

3.6. Interactividad y ajustes en tiempo real

El código permite ajustar parámetros clave para explorar el fenómeno de difracción:

- **Ancho de la rendija.**
- **Distancia de la rendija a la pantalla.**
- **Posición de observación en la pantalla.**

Los cálculos y visualizaciones se actualizan dinámicamente para reflejar los cambios.

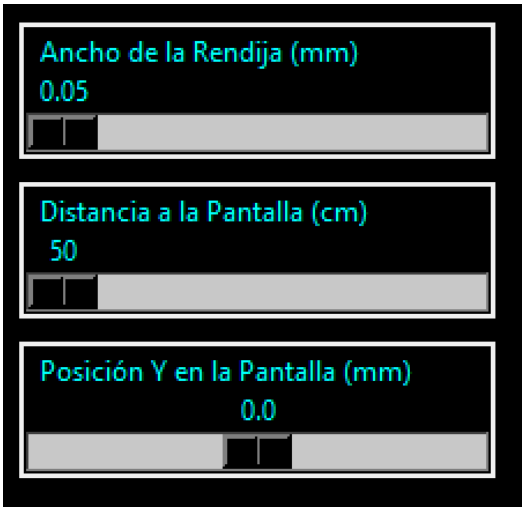

Figura 8: Interactividad de variables

3.7. Física subyacente

1. **Interferencia:** El patrón es resultado de la interferencia constructiva y destructiva entre ondas de luz.
2. **Principio de Huygens-Fresnel:** Cada punto de la rendija actúa como una fuente puntual de ondas secundarias.
3. **Difracción de Fraunhofer:** Se asume que la pantalla está lo suficientemente lejos para cumplir la condición de campo lejano.

3.8. Aproximaciones y limitaciones

- **Fuente monocromática:** Se utiliza un láser HeNe ($\lambda = 633\,\text{nm}$).
- **Rendija delgada:** No se consideran efectos tridimensionales.
- **Exclusión de efectos secundarios:** No se modelan polarización ni aberraciones ópticas.

3.9. Codigo Completo

```
1 import tkinter as tk
2 import math
3
```

```
4  class SimulacionDifraccionLaser:
5      def __init__(self):
6          #ACreacion del titulo de la ventana
7          self.raiz = tk.Tk()
8          self.raiz.title("Simulador de Láser con Lente y Rendija")
9          self.raiz.configure(bg='black')
10
11         #TConstantes físicas
12         self.longitud_onda = 633e-9  # Longitud de onda del láser HeNe
           ↪ (rojo)
13         self.k = 2 * math.pi / self.longitud_onda
14         self.ancho_haz = 2  # Ancho del haz láser en mm
15         self.distancia_focal = 50  # Distancia focal fija de la lente
           ↪ (cm)
16
17         #AParámetros ajustables
18         self.ancho_rendija = tk.DoubleVar(value=0.1)  # Ancho de la
           ↪ rendija (mm)
19         self.distancia_pantalla = tk.DoubleVar(value=100)  # Distancia a
           ↪ la pantalla (cm)
20         self.posicion_y = tk.DoubleVar(value=0)  # Posición Y en la
           ↪ pantalla (mm)
21
22         #ACrear frames para los paneles
23         self.marco_montaje = tk.Frame(self.raiz, bg='black')
24         self.marco_montaje.pack(side=tk.LEFT, padx=10, pady=10)
25
26         self.marco_pantalla = tk.Frame(self.raiz, bg='black')
27         self.marco_pantalla.pack(side=tk.RIGHT, padx=10, pady=10)
28
29         self.marco_distribucion = tk.Frame(self.raiz, bg='black')
30         self.marco_distribucion.pack(side=tk.BOTTOM, fill=tk.X,
           ↪ expand=True, padx=10, pady=10)
31
32         #ACanvas para el montaje óptico y la pantalla
33         self.lienzo_montaje = tk.Canvas(self.marco_montaje, width=600,
           ↪ height=400, bg='black')
34         self.lienzo_montaje.pack(pady=10)
35
36         self.lienzo_pantalla = tk.Canvas(self.marco_pantalla, width=400,
           ↪ height=400, bg='black')
```

```
37          self.lienzo_pantalla.pack(pady=10)
38          self.lienzo_pantalla.bind("<Motion>", self.mostrar_intensidad)
39
40          #ACanvas para la distribución de intensidad
41          self.lienzo_distribucion = tk.Canvas(self.marco_distribucion,
42                                               width=self.raiz.winfo_scree
                                                 ↪ nwidth(),
43                                               height=200,
44                                               bg='black')
45          self.lienzo_distribucion.pack(fill=tk.X, expand=True, pady=10)
46
47          #AEtiquetas
48          tk.Label(self.marco_montaje,
49                   text="Montaje Experimental",
50                   bg='black',
51                   fg='white',
52                   font=('Arial', 12)).pack(pady=5)
53          tk.Label(self.marco_pantalla,
54                   text="Proyección en Pantalla",
55                   bg='black', fg='cyan',
56                   font=('Arial', 12)).pack()
57
58          tk.Label(self.marco_distribucion,
59                   text="Distribución de Intensidad",
60                   bg='black', fg='cyan',
61                   font=('Arial', 12)).pack()
62
63          self.etiqueta_intensidad = tk.Label(self.marco_pantalla,
64                                              text="Intensidad: ",
65                                              bg='black', fg='cyan',
66                                              font=('Arial', 10))
67          self.etiqueta_intensidad.pack()
68
69          self.crear_controles()
70          self.actualizar_simulacion()
71      #APP
72      def crear_controles(self):
73          marco_control = tk.Frame(self.raiz, bg='black')
74          marco_control.pack(side=tk.BOTTOM, pady=10)
75
```

```
76          # Control para el ancho de la rendija
77          tk.Scale(marco_control,
78                   variable=self.ancho_rendija,
79                   from_=0.05, to=0.5,
80                   resolution=0.01,
81                   label='Ancho de la Rendija (mm)',
82                   orient='horizontal',
83                   command=lambda x: self.actualizar_simulacion(),
84                   length=200,
85                   bg='black',
86                   fg='cyan').pack(pady=5)
87
88          # Control para la distancia a la pantalla
89          tk.Scale(marco_control,
90                   variable=self.distancia_pantalla,
91                   from_=50, to=600,
92                   resolution=1,
93                   label='Distancia a la Pantalla (cm)',
94                   orient='horizontal',
95                   command=lambda x: self.actualizar_simulacion(),
96                   length=200,
97                   bg='black',
98                   fg='cyan').pack(pady=5)
99
100         # Control para la posición Y en la pantalla
101         tk.Scale(marco_control,
102                  variable=self.posicion_y,
103                  from_=-10, to=10,
104                  resolution=0.1,
105                  label='Posición Y en la Pantalla (mm)',
106                  orient='horizontal',
107                  command=lambda x: self.actualizar_simulacion(),
108                  length=200,
109                  bg='black',
110                  fg='cyan').pack(pady=5)
111
112     def dibujar_haz_laser(self):
113         """Dibuja el haz láser con rayos calculados usando fórmulas
            ↪  ópticas"""
114         ancho = 600
```

```
115         alto = 400
116         centro_y = alto / 2
117 
118         # Posiciones de los componentes
119         x_laser = 50
120         x_lente = 200
121         x_rendija = 400
122         x_pantalla = 550
123 
124         # Dibujar el láser
125         self.lienzo_montaje.create_rectangle(x_laser - 15, centro_y - 10,
    ↪  x_laser, centro_y + 10,
126                                              fill='red',
                                                 ↪  outline='yellow')
127         self.lienzo_montaje.create_text(x_laser - 7, centro_y + 25,
128                                         text="Láser", fill='white')
129 
130         # Parámetros del sistema óptico
131         f = self.distancia_focal * 1e-2  # Distancia focal en metros
132         d1 = (x_lente - x_laser) * 1e-2  # Distancia del láser a la lente
    ↪  en metros
133         d2 = (x_rendija - x_lente) * 1e-2  # Distancia de la lente a la
    ↪  rendija en metros
134 
135         # Calcular la posición de la imagen usando la fórmula de la lente
    ↪  delgada
136         d2_calc = 1 / (1 / f - 1 / d1)
137 
138         # Número de rayos a dibujar
139         n_rayos = 5
140         separacion_rayos = self.ancho_haz / 2  # Mitad del ancho del haz
    ↪  láser
141 
142         for i in range(n_rayos):
143             # Posición vertical inicial del rayo
144             desplazamiento_y = (i - (n_rayos - 1) / 2) * separacion_rayos
145 
146             # Rayo antes de la lente
147             self.lienzo_montaje.create_line(x_laser, centro_y +
                ↪  desplazamiento_y,
```

```
                                        x_lente, centro_y +
                                        ↪ desplazamiento_y,
                                        fill='red', width=1)

            # Calcular la altura del rayo en la rendija usando la fórmula
            ↪ de aumento
            m = -d2_calc / d1
            y_en_rendija = m * desplazamiento_y

            # Rayo desde la lente hasta la rendija
            self.lienzo_montaje.create_line(x_lente, centro_y +
            ↪ desplazamiento_y,
                                        x_rendija, centro_y +
                                        ↪ y_en_rendija,
                                        fill='red', width=1)

            # Rayo desde la rendija hasta la pantalla
            self.lienzo_montaje.create_line(x_rendija, centro_y +
            ↪ y_en_rendija,
                                        x_pantalla, centro_y +
                                        ↪ y_en_rendija,
                                        fill='red', width=1)

        # Dibujar la lente (simplificada)
        altura_lente = 80
        self.lienzo_montaje.create_line(x_lente, centro_y - altura_lente
        ↪ / 2,
                                        x_lente, centro_y + altura_lente
                                        ↪ / 2,
                                        fill='cyan', width=2)
        self.lienzo_montaje.create_text(x_lente, centro_y + altura_lente
        ↪ / 2 + 15,
                                        text=f"Lente (f={self.distancia_
                                        ↪ focal}cm)",
                                        ↪ fill='white')

        # Dibujar la rendija
        altura_rendija = 80
        self.lienzo_montaje.create_line(x_rendija, centro_y -
        ↪ altura_rendija / 2,
```

```
                                        x_rendija, centro_y +
                                        ↪  altura_rendija / 2,
                                        fill='yellow', width=2)
        self.lienzo_montaje.create_text(x_rendija, centro_y +
        ↪  altura_rendija / 2 + 15,
                                        text="Rendija", fill='white')

        # Dibujar la pantalla
        altura_pantalla = 160
        self.lienzo_montaje.create_line(x_pantalla, centro_y -
        ↪  altura_pantalla / 2,
                                        x_pantalla, centro_y +
                                        ↪  altura_pantalla / 2,
                                        fill='white', width=2)
        self.lienzo_montaje.create_text(x_pantalla, centro_y +
        ↪  altura_pantalla / 2 + 15,
                                        text="Pantalla", fill='white')

    def calcular_patron_difraccion(self, y):
        """Calcula el patrón de difracción de una rendija única usando la
        ↪  fórmula de Fraunhofer"""
        a = self.ancho_rendija.get() * 1e-3  # Ancho de la rendija en
        ↪  metros
        L = self.distancia_pantalla.get() * 1e-2  # Distancia a la
        ↪  pantalla en metros
        k = 2 * math.pi / self.longitud_onda  # Número de onda

        theta = math.atan(y / L)
        beta = k * a * math.sin(theta) / 2

        if beta == 0:
            return 1.0
        else:
            return (math.sin(beta) / beta) ** 2

    def dibujar_patron_pantalla(self):
        """Dibuja el patrón de difracción en la pantalla"""
        ancho = 400
        alto = 400

```

```
208         # Limpiar el canvas de la pantalla
209         self.lienzo_pantalla.delete('all')
210 
211         # Dibujar el marco de la pantalla
212         self.lienzo_pantalla.create_rectangle(10, 10, ancho - 10, alto -
            ↪ 10, outline='white')
213 
214         # Calcular y dibujar el patrón de difracción
215         escala_y = 1e-3  # Factor de escala para la coordenada y (en
            ↪ metros)
216 
217         # Calcular intensidades
218         valores_y = [(py - alto / 2) * escala_y for py in range(alto)]
219         intensidades = [self.calcular_patron_difraccion(y) for y in
            ↪ valores_y]
220 
221         # Normalizar intensidades
222         intensidad_maxima = max(intensidades)
223         if intensidad_maxima > 0:
224             intensidades = [i / intensidad_maxima for i in intensidades]
225 
226         # Dibujar el patrón
227         for py, intensidad in enumerate(intensidades):
228             color = self.intensidad_a_color(intensidad)
229             self.lienzo_pantalla.create_line(10, py, ancho - 10, py,
                ↪ fill=color)
230 
231         # Dibujar el punto ajustable.
232         y_pixel = int(alto / 2 - self.posicion_y.get() / escala_y)
233 
234 
235     def intensidad_a_color(self, intensidad):
236         """Convierte una intensidad (0-1) a un color RGB"""
237         rojo = int(255 * intensidad)
238         return f'#{rojo:02x}0000'
239 
240     def dibujar_distribucion_intensidad(self):
241         """Dibuja la distribución de intensidad usando tkinter"""
242         ancho = self.lienzo_distribucion.winfo_width()  # Obtener el
            ↪ ancho real del canvas
```

```
243         alto = 200
244         margen = 20
245         escala_x = (ancho - 2 * margen) / 20  # Rango de 20mm centrado en
            ↪ 0
246         escala_y = alto - 2 * margen
247 
248         # Limpiar el canvas
249         self.lienzo_distribucion.delete('all')
250 
251         # Dibujar ejes
252         self.lienzo_distribucion.create_line(margen, alto - margen, ancho
            ↪ - margen, alto - margen,
253                                              fill='white')  # Eje X
254         self.lienzo_distribucion.create_line(margen, alto - margen,
            ↪ margen, margen, fill='white')  # Eje Y
255 
256         # Etiquetas de los ejes
257         self.lienzo_distribucion.create_text(ancho / 2, alto - 5,
            ↪ text="", fill='white')
258         self.lienzo_distribucion.create_text(10, alto / 2, text="",
            ↪ angle=90, fill='white')
259 
260         # Calcular y dibujar la distribución de intensidad
261         puntos = []
262         for x in range(ancho):
263             y = (x - ancho / 2) / escala_x  # Convertir pixel a mm
264             intensidad = self.calcular_patron_difraccion(y * 1e-3)  #
                ↪ Convertir mm a m
265             y_pixel = alto - margen - int(intensidad * escala_y)
266             puntos.extend([x, y_pixel])
267 
268         # Asegurarse de que hay al menos dos puntos antes de dibujar la
            ↪ línea
269         if len(puntos) >= 4:
270             self.lienzo_distribucion.create_line(puntos, fill='red',
                ↪ smooth=True)
271 
272         # Dibujar marcas en el eje X
273         for i in range(-10, 11, 5):
274             x = margen + (i + 10) * escala_x
```

```
275             self.lienzo_distribucion.create_line(x, alto - margen, x,
                ↪  alto - margen + 5, fill='white')
276             self.lienzo_distribucion.create_text(x, alto - margen + 10,
                ↪  text=str(i), fill='white')
277 
278         # Dibujar marcas en el eje Y
279         for i in range(0, 11, 2):
280             y = alto - margen - i * (escala_y / 10)
281             self.lienzo_distribucion.create_line(margen - 5, y, margen,
                ↪  y, fill='white')
282             self.lienzo_distribucion.create_text(margen - 10, y,
                ↪  text=f"{i / 10:.1f}", fill='white')
283 
284         # Dibujar línea vertical en la posición Y actual
285         y_actual = self.posicion_y.get()
286         x_actual = margen + (y_actual + 10) * escala_x
287         self.lienzo_distribucion.create_line(x_actual, margen, x_actual,
                ↪  alto - margen, fill='yellow', dash=(4, 4))
288 
289     def mostrar_intensidad(self, evento):
290         """Muestra la intensidad en el punto donde está el cursor"""
291         ancho = 400
292         alto = 400
293         escala_y = 1e-3  # Factor de escala para la coordenada y (en
                ↪  metros)
294 
295         # Calcular la posición y en metros
296         y = (evento.y - alto / 2) * escala_y
297 
298         # Calcular la intensidad en ese punto
299         intensidad = self.calcular_patron_difraccion(y)
300 
301         # Actualizar la etiqueta con la intensidad
302         self.etiqueta_intensidad.config(text=f"Intensidad:
                ↪  {intensidad:.4f} W/m²")
303 
304     def actualizar_simulacion(self):
305         """Actualiza toda la simulación"""
306         self.lienzo_montaje.delete('all')
307         self.lienzo_pantalla.delete('all')
```

```
        self.lienzo_distribucion.delete('all')
        self.dibujar_haz_laser()
        self.dibujar_patron_pantalla()
        self.dibujar_distribucion_intensidad()

        # Actualizar la intensidad en el punto ajustable
        y = self.posicion_y.get() * 1e-3  # Convertir mm a metros
        intensidad = self.calcular_patron_difraccion(y)
        self.etiqueta_intensidad.config(text=f"Intensidad en
        ↪  Y={self.posicion_y.get():.1f} mm: {intensidad:.4f} W/m²")

    def ejecutar(self):
        self.raiz.mainloop()

if __name__ == "__main__":
    app = SimulacionDifraccionLaser()
    app.ejecutar()

print("El código de la simulación ha sido actualizado. Ejecute la
↪  simulación creando una instancia de SimulacionDifraccionLaser y
↪  llamando a su método ejecutar.")
```

4. Referencias

- OpenStax. (n.d.). Difracción de una rendija. En Física universitaria volumen 3. Recuperado el [28/11/24], de https://openstax.org/books/f
- Nave, C. R. (n.d.). Difracción por una rendija simple. En HyperPhysics. Recuperado el [28/11/24], de http://hyperphysics.phy-astr.gsu.edu/hbasees/phyopt/sinslitd.html
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. *Nature, 585*(7825), 357–362. `https://doi.org/10.1038/s41586-020-2649-2`
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90–95. `https://doi.org/10.1109/MCSE.2007.55`

- Van Rossum, G., & Drake, F. L. (2001). *The Python Language Reference Manual*. Python Software Foundation. Recuperado de `https://www.python.org`