

# **Simulación del Efecto de Convolución de Punta en Microscopia de Fuerzas Atómicas (AFM)**

---

**Rafael Jiménez Parra NIU : 1638106**

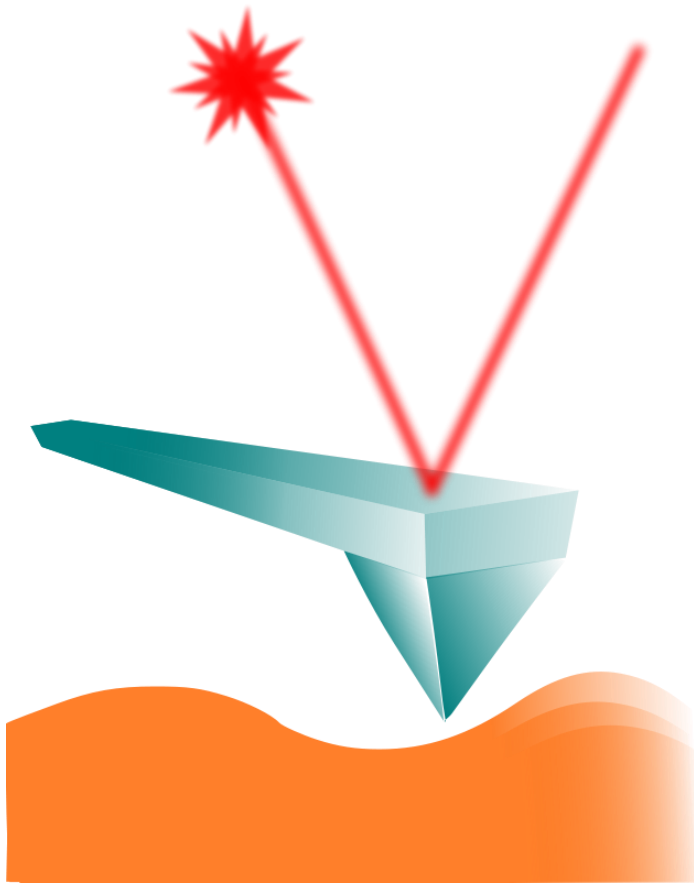


# INTRODUCCIÓN

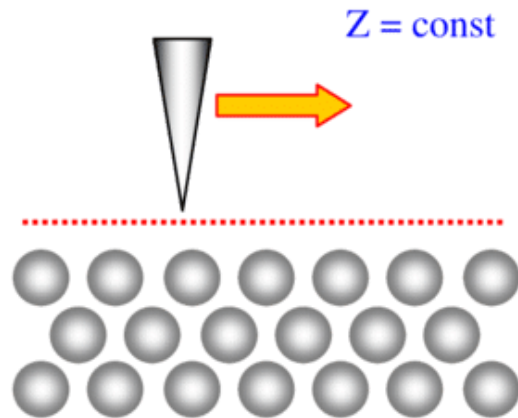
---

---

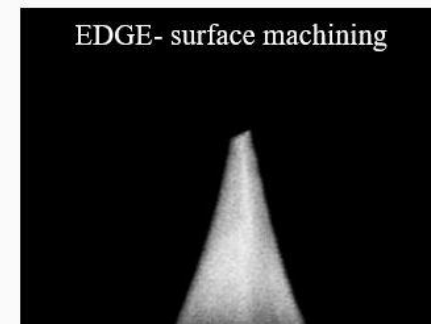
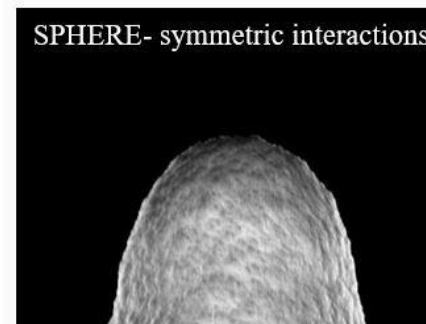
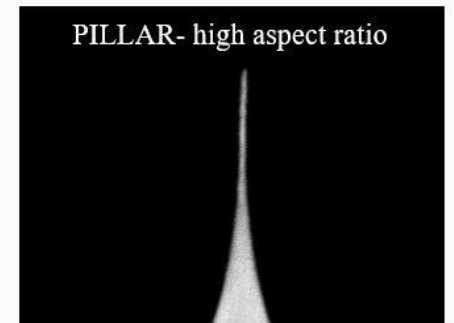
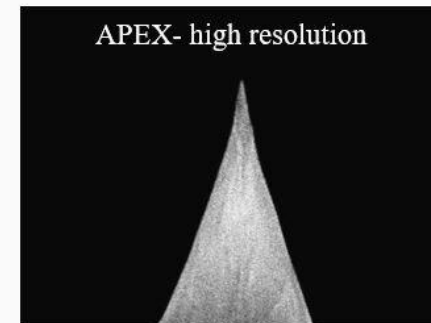
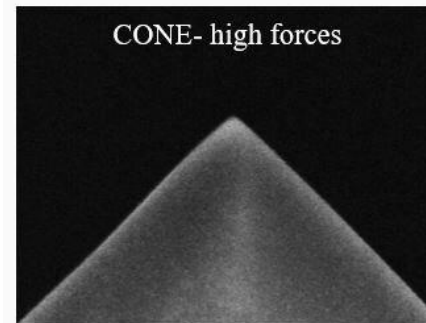
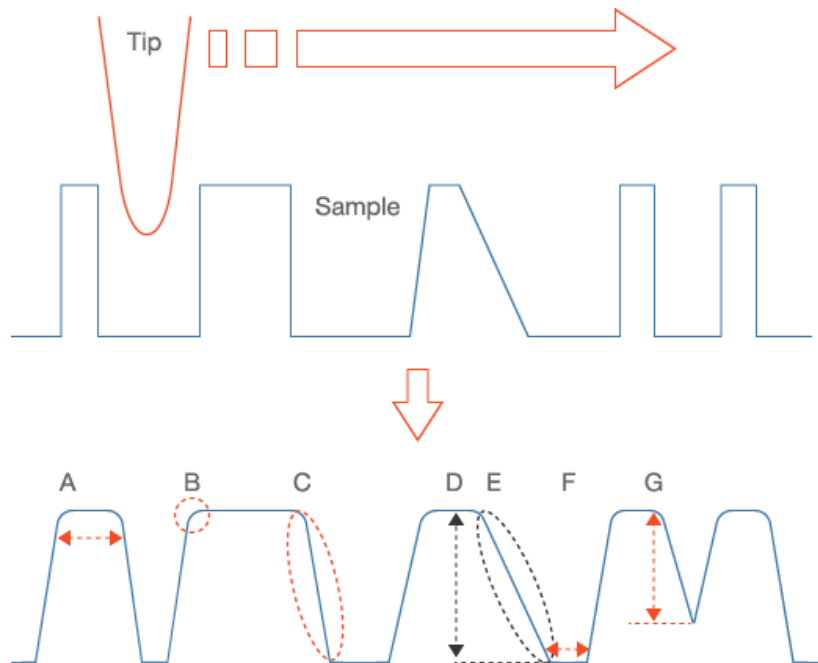
# Atomic Force Microscope



Distancia Punta Muestra	Fuerza Dominante	Rango de Fuerza
+10 / 100 micrómetros	Nada	-
pocos micrómetros	Fuerzas Eléctricas	$\pm 1 \text{ nN} \sim \pm 1 \mu\text{N}$
menos de pocos nanómetros	Van der Waals atractivas Capilaridad	$\pm 100 \text{ nN} \sim \pm 10 \mu\text{N}$
Contacto	Fuerzas Pauli Repulsivas Fuerzas Mecánicas	$\pm 1 \text{ nN} \sim \pm 10 \mu\text{N}$



# Efecto Convolución





# OBJETIVOS

---

---



Desarrollo modelo computacional de AFM



Modelar el efecto de convolución



Repetir metodología para otras muestras



# METODOLOGÍA

---

---

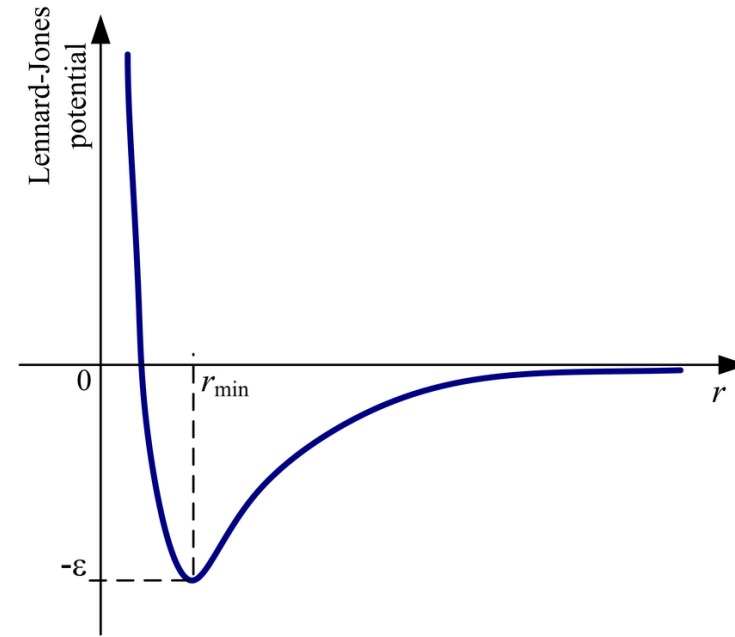


# Software



- **NumPy – Computación Numérica**
- **SciPy - Convolución**
- **Matplotlib – Graficar 2D**
- **Plotly – Graficar 3D**
- **tqdm - Utilidad**

# Potencial Lennard - Jones



$$V(r) = 4\varepsilon \left[ \left( \frac{\sigma}{r} \right)^{12} - \left( \frac{\sigma}{r} \right)^6 \right]$$



# Código - Constantes

---

```
# Parte 1: Cuerpo de la simulación, definición de parámetros más importantes y funciones
import numpy as np
from scipy.ndimage import grey_dilation
from tqdm.auto import tqdm
import plotly.graph_objects as go
import matplotlib.pyplot as plt

# Parámetros de gran importancia
TAMANO = 120.0      # Tamaño de la imagen en nm
RESOLUCION = 128    # Píxeles de la imagen
RADIO_PUNTA = 10.0   # Radio del extremo de la punta en nm
ANGULO_PUNTA = 40.0 # Ángulo del cono de la punta en gradosº

# Parámetros físicos para la fuerza de Lennard-Jones
EPSILON = 2e-20     # En Joules
SIGMA = 0.3         # nm
```

# Código – Lennard-Jones

```
def lennard_jones_force(d_nm):  
    """  
    Calcula la fuerza de Lennard-Jones con la fórmula matemáticamente exacta,  
    derivada del potencial.  
    """  
  
    # Parámetros físicos  
    EPSILON = 2e-20 # Joules  
    SIGMA = 0.3      # nm  
    # Conversión a metros para consistencia  
    sigma_m = SIGMA * 1e-9  
    d_m = d_nm * 1e-9  
    # Evitar valores muy pequeños para la estabilidad numérica  
    d_safe_m = np.maximum(d_m, sigma_m * 0.85)  
    # Cálculo de los términos con las potencias correctas (13 y 7)  
    r7 = (sigma_m / d_safe_m)**7  
    r13 = (sigma_m / d_safe_m)**13  
    # Fuerza en Newtons, usando la fórmula correcta  
    force_N = (24 * EPSILON / sigma_m) * (2 * r13 - r7)  
    # Conversión final a nanoNewtons  
    return force_N * 1e9
```

# Código - Superficie

```
def crear_superficie_muestra(tamano, resolucion):  
    """Crea una superficie de prueba con algunas características."""  
    x = np.linspace(-tamano/2, tamano/2, resolucion)  
    y = np.linspace(-tamano/2, tamano/2, resolucion)  
    X, Y = np.meshgrid(x, y)  
    Z = np.zeros_like(X)  
  
    # Estructuras creadas sobre un plano Z=0  
    Z[(X > -40) & (X < 40) & (Y > -40) & (Y < 40)] = 1.5  
    Z[(X > 10) & (X < 15) & (Y > -30) & (Y < 30)] = -2.0  
    Z[(X + 30)**2 + (Y + 30)**2 < 8**2] = 2.0  
    return X, Y, Z
```

# Código - Punta

---

```
def crear_forma_punta(radio, angulo, tamano, resolucion, pixels=65):
    """Crea la geometría 3D de la punta."""
    nm_por_pixel = tamano / resolucion
    centro = pixels // 2

    # Creación de la malla de coordenadas para la punta
    x_px = np.arange(-centro, centro + 1)
    y_px = np.arange(-centro, centro + 1)
    X_px, Y_px = np.meshgrid(x_px, y_px)
    R_nm = np.sqrt(X_px**2 + Y_px**2) * nm_por_pixel

    # Parte esférica del extremo
    parte_esferica = np.full(R_nm.shape, -np.inf)
    mascara = R_nm < radio
    parte_esferica[mascara] = np.sqrt(radio**2 - R_nm[mascara]**2) - radio

    # Parte cónica
    parte_conica = R_nm * -np.tan(np.radians(90 - angulo / 2))

    # Combinación de las dos formas
    forma_punta = np.maximum(parte_esferica, parte_conica)
    # Normalización (el punto más alto de la punta es Z=0)
    forma_punta -= np.max(forma_punta)
    return forma_punta
```

# Código – Convolución + Campo Fuerzas

```
print("Primero se genera la superficie y punta")
X, Y, Z_real = crear_superficie_muestra(TAMANO, RESOLUCION)
punta = crear_forma_punta(RADIO_PUNTA, ANGULO_PUNTA, TAMANO, RESOLUCION)

# Cálculo de la convolución
print("Segundo se calcula el efecto de la punta")
Z_convolucionada = grey_dilation(Z_real, structure=np.flip(punta))

# Cálculo del mapa de fuerzas
print("Tercero hacemos el AFM en modo altura constante")
altura_escaneo = np.max(Z_convolucionada) + 1.2
mapa_fuerzas = np.zeros_like(Z_convolucionada)

for y in tqdm(range(RESOLUCION), desc="Escaneando"):
    for x in range(RESOLUCION):
        distancia = altura_escaneo - Z_convolucionada[y, x]
        mapa_fuerzas[y, x] = lennard_jones_force(distancia)
```



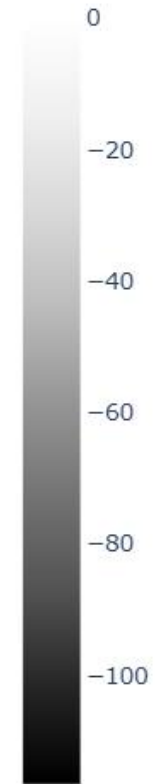
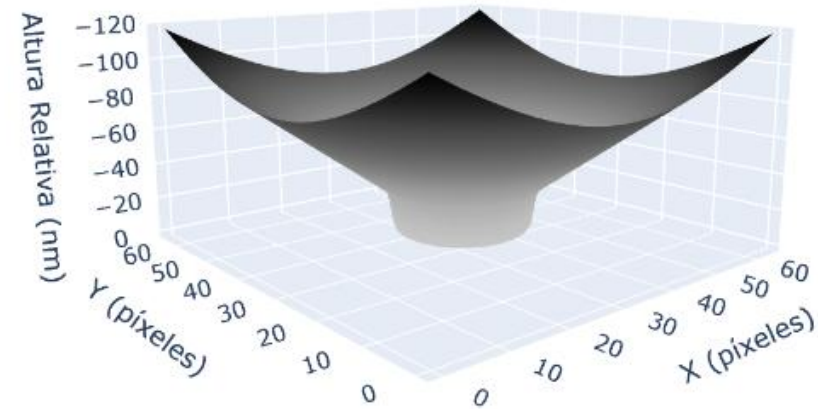
# RESULTADOS



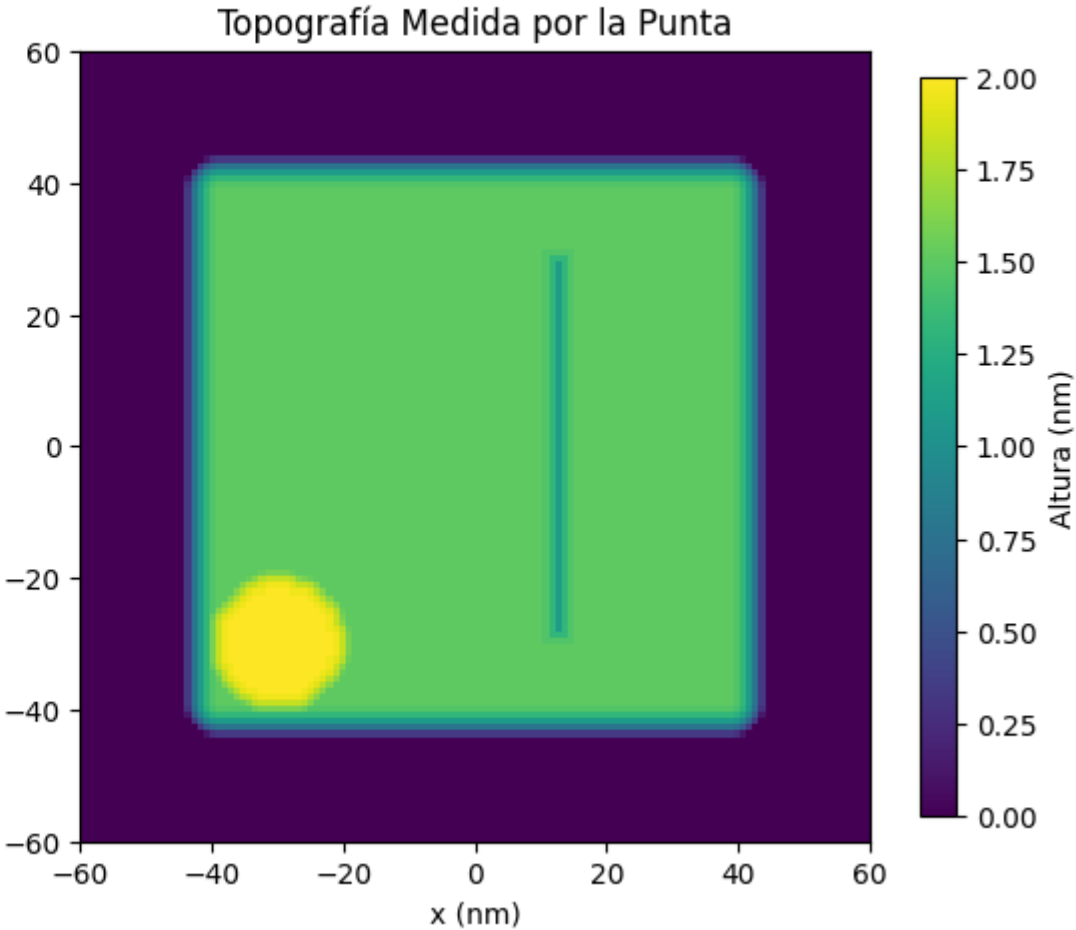
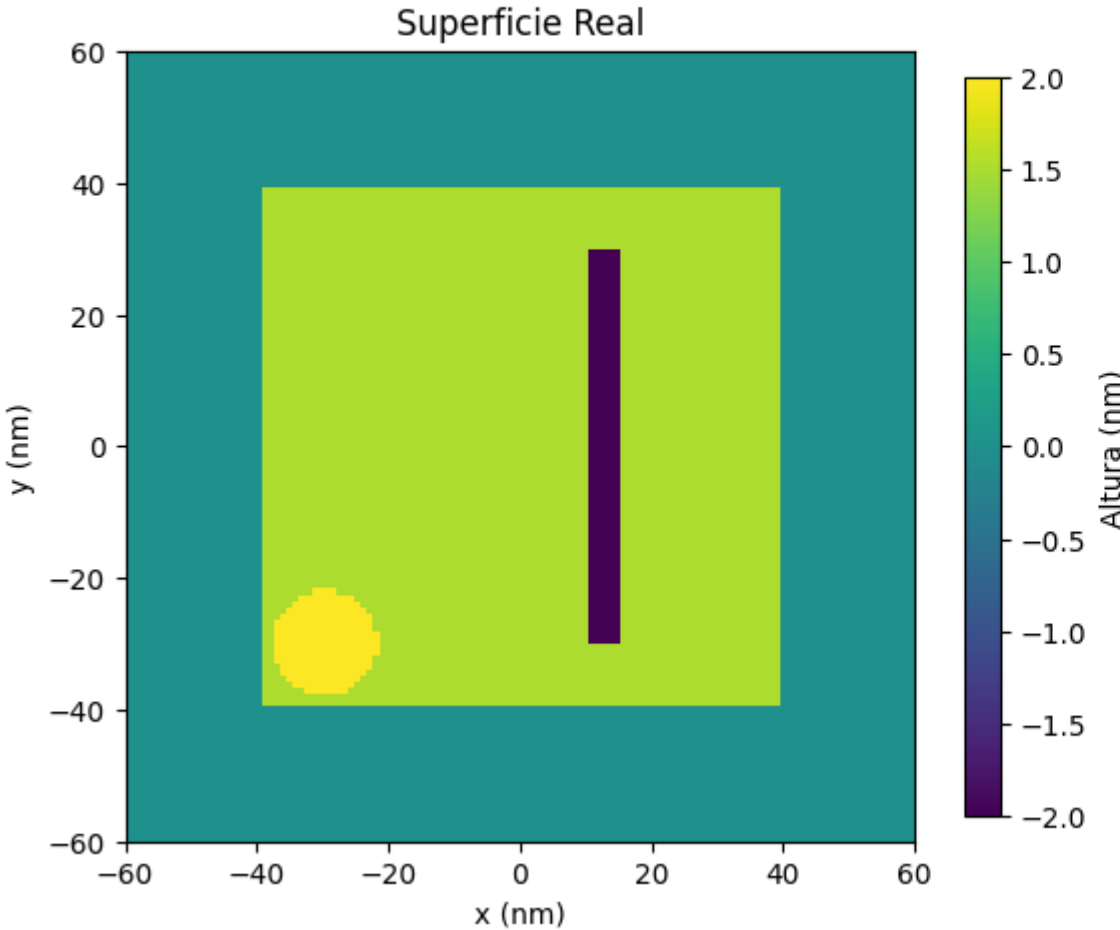


## Forma de la Punta en 3D

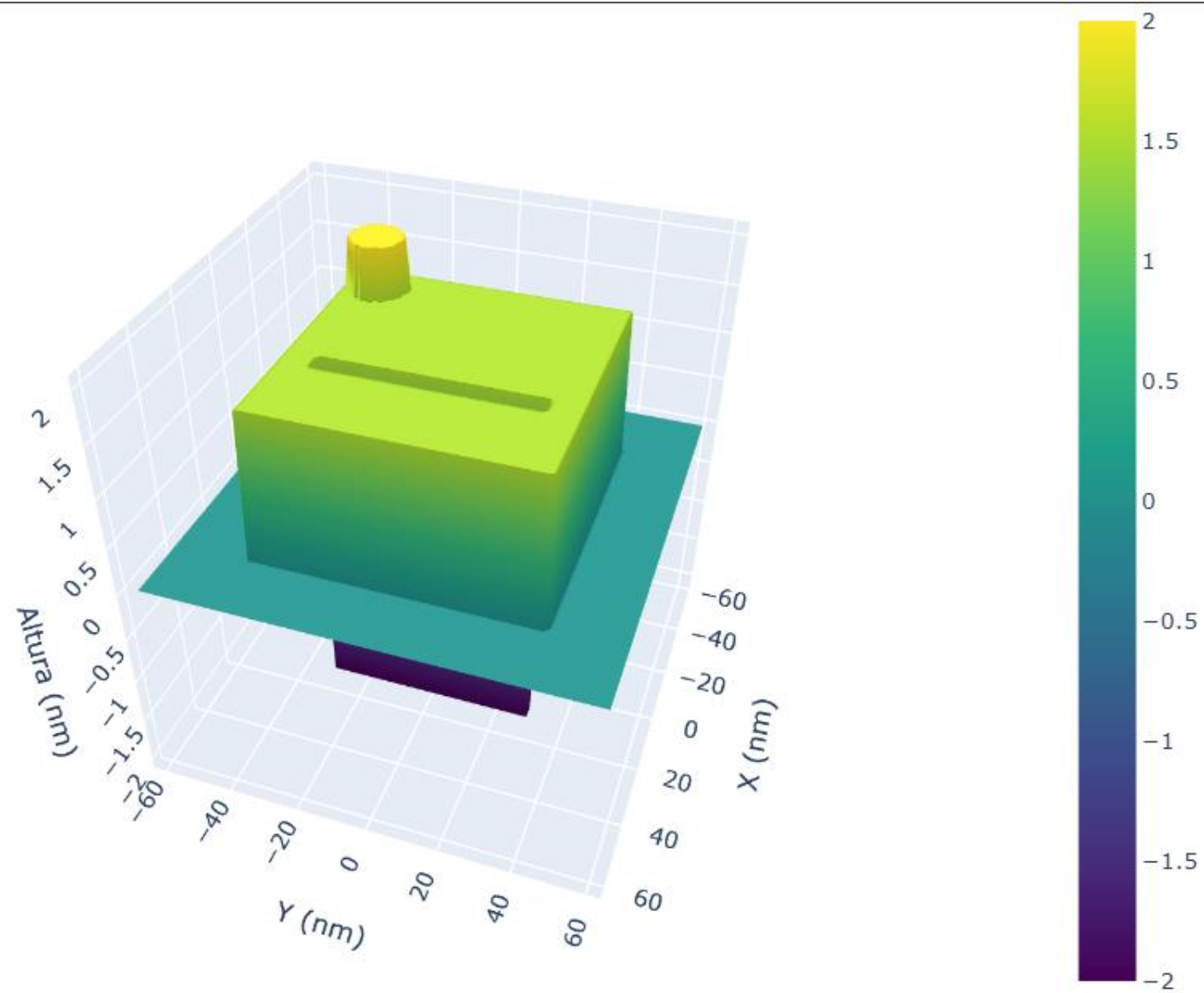
**Radio 10 nm**



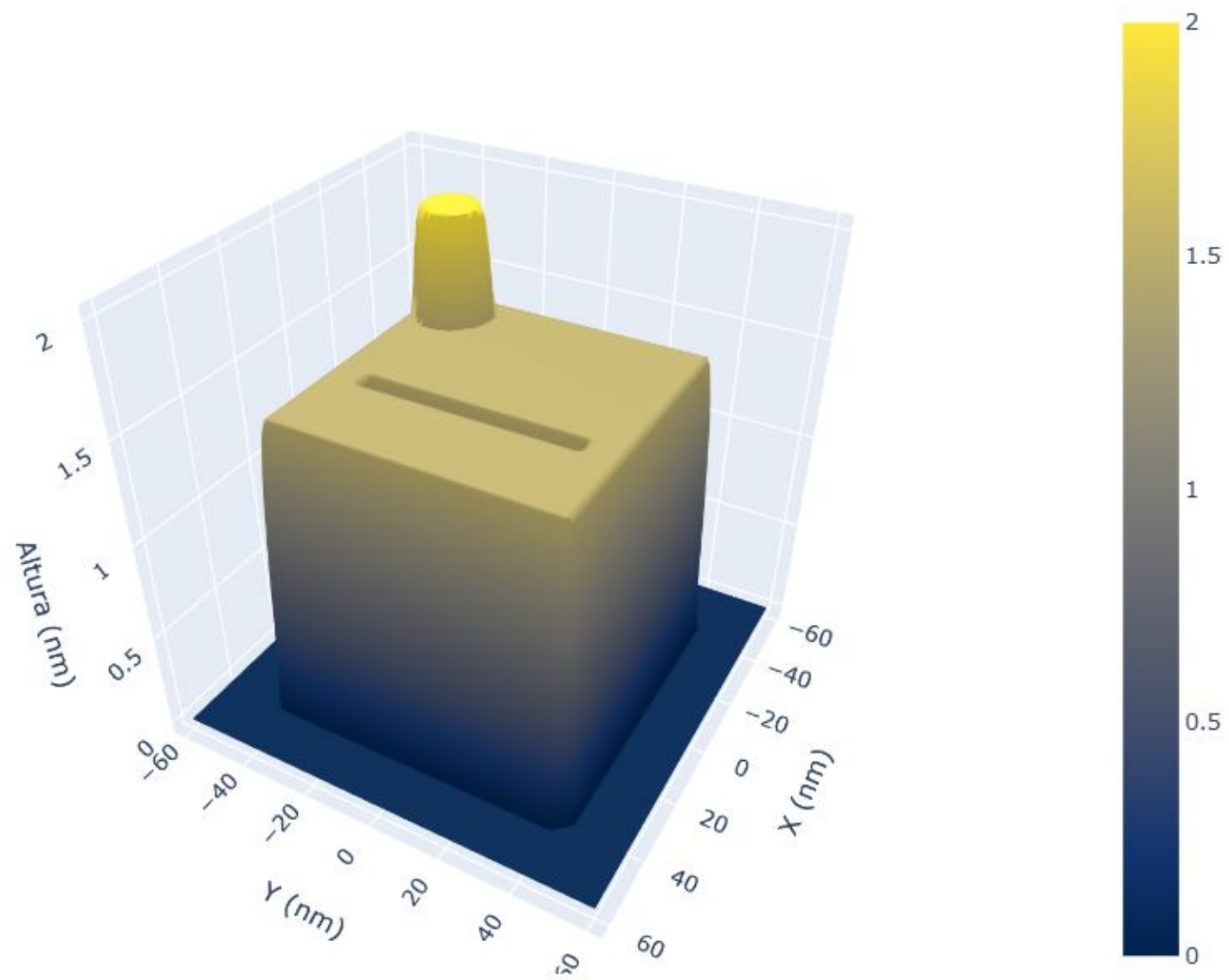
# Comparativa 2D (Vista Superior)

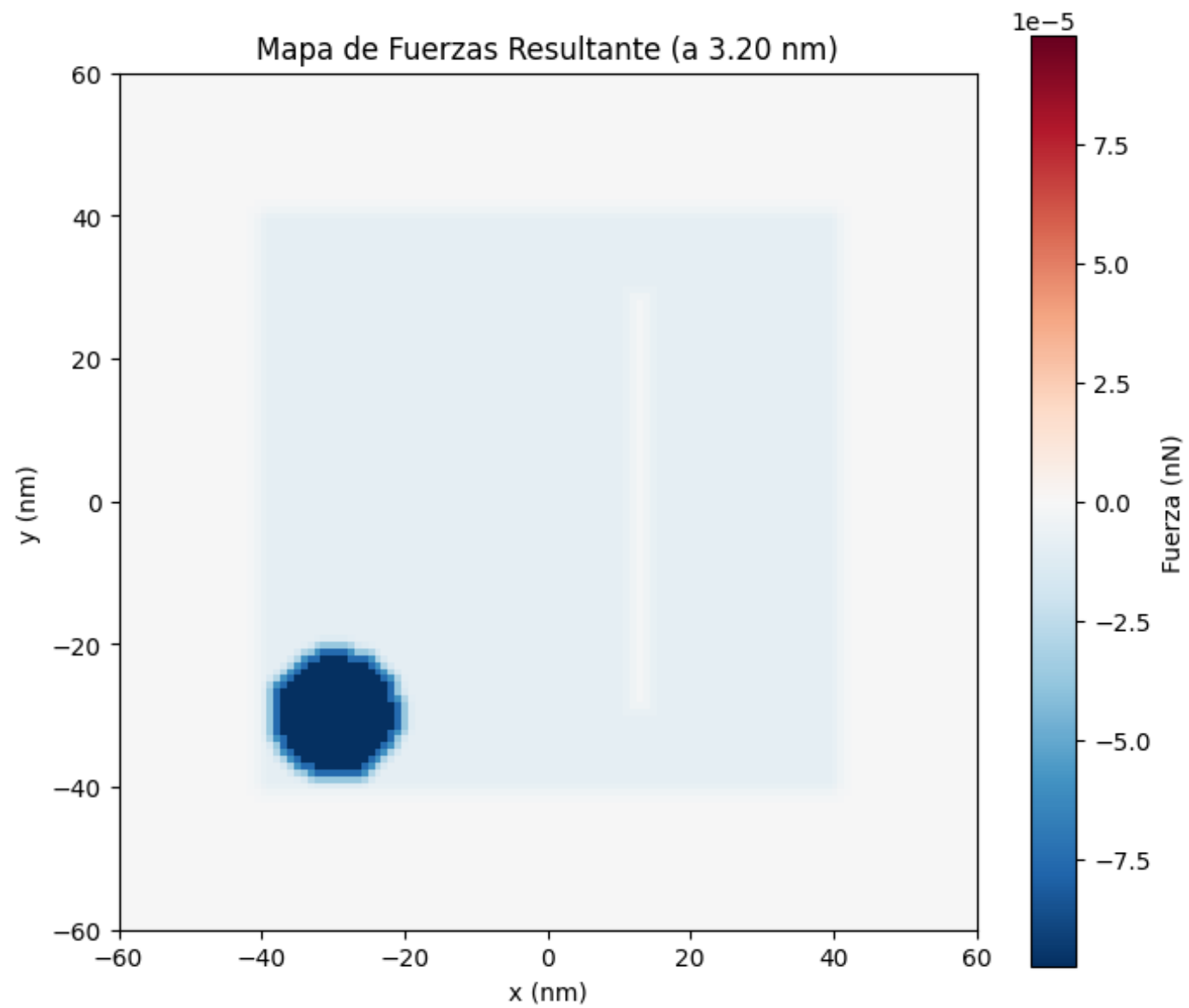


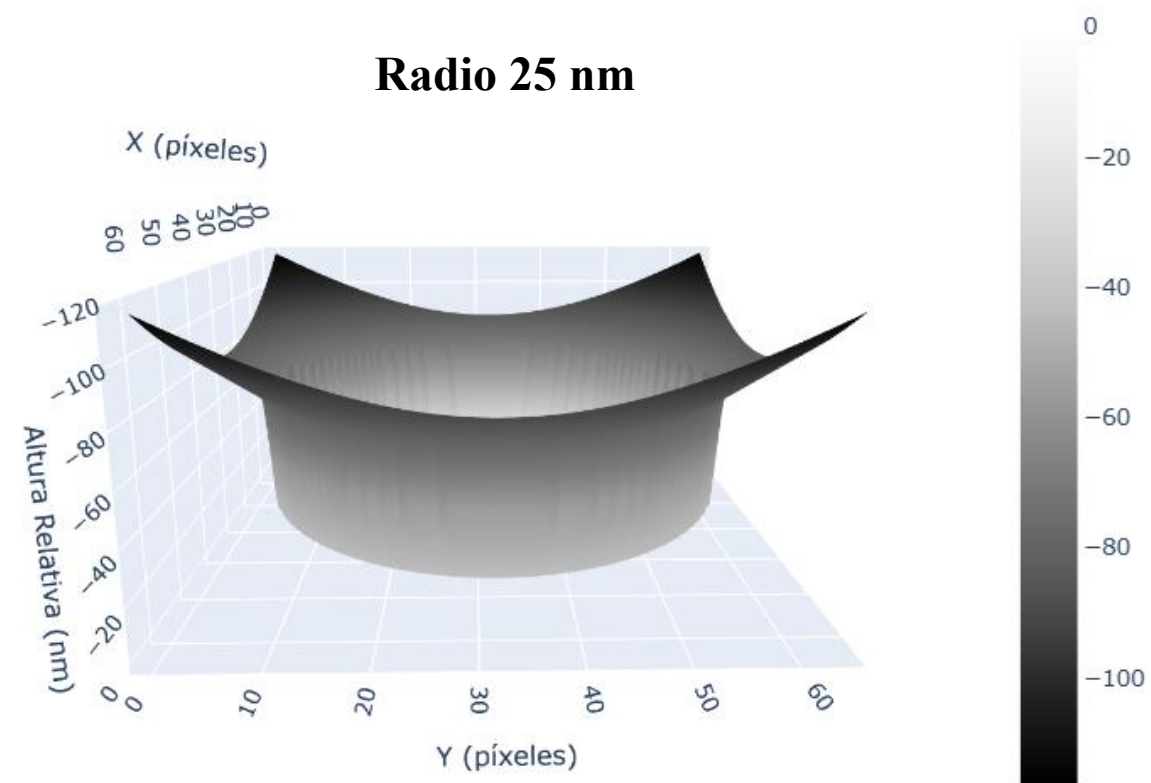
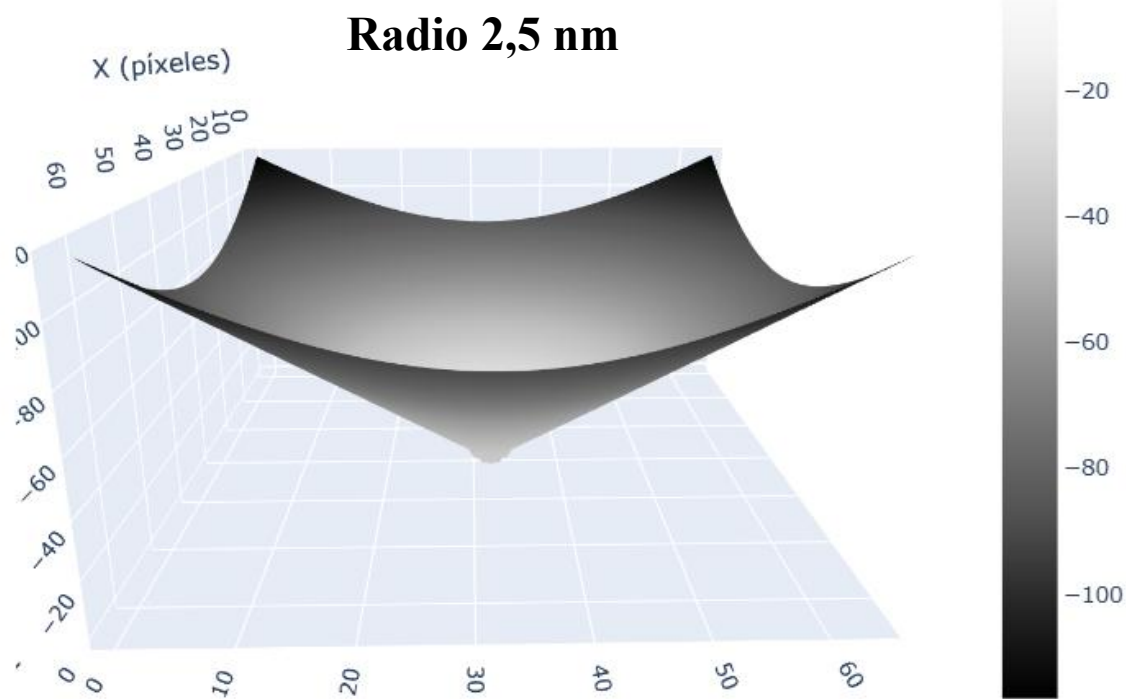




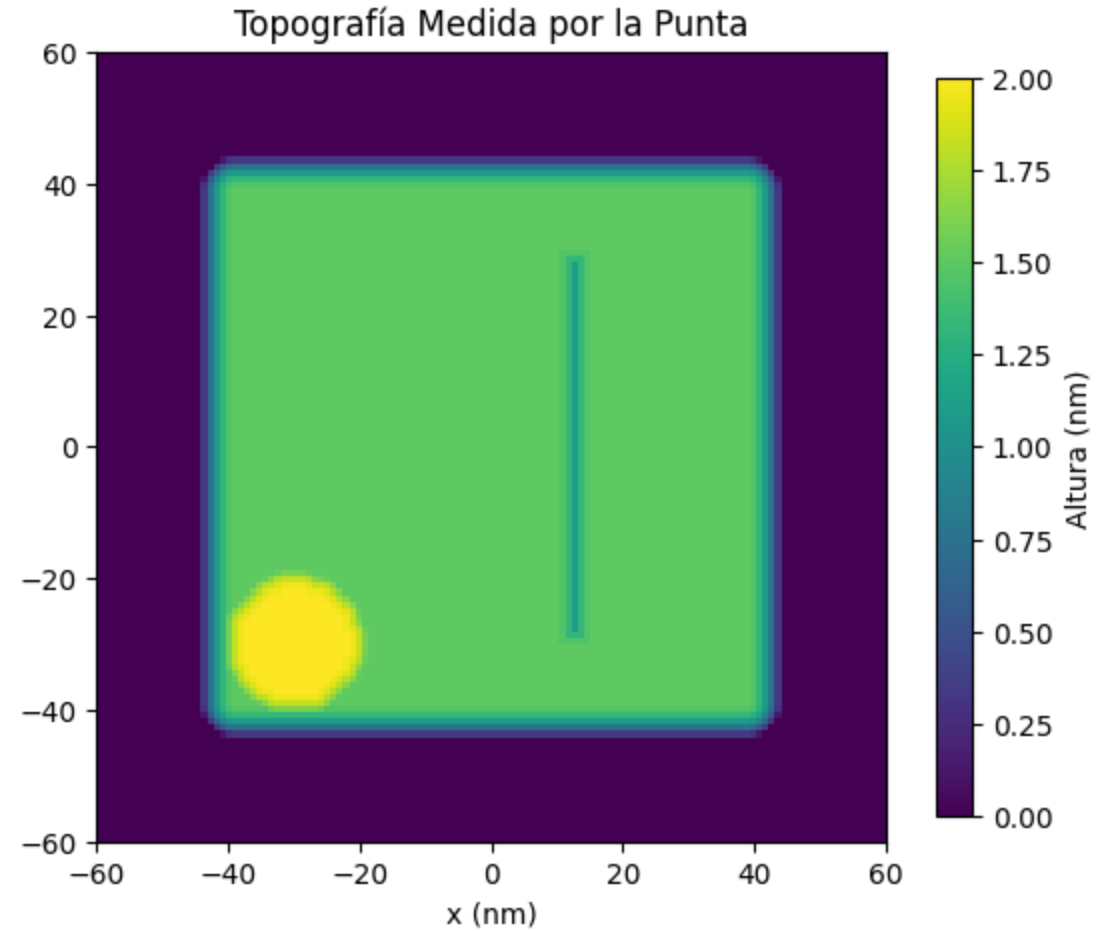
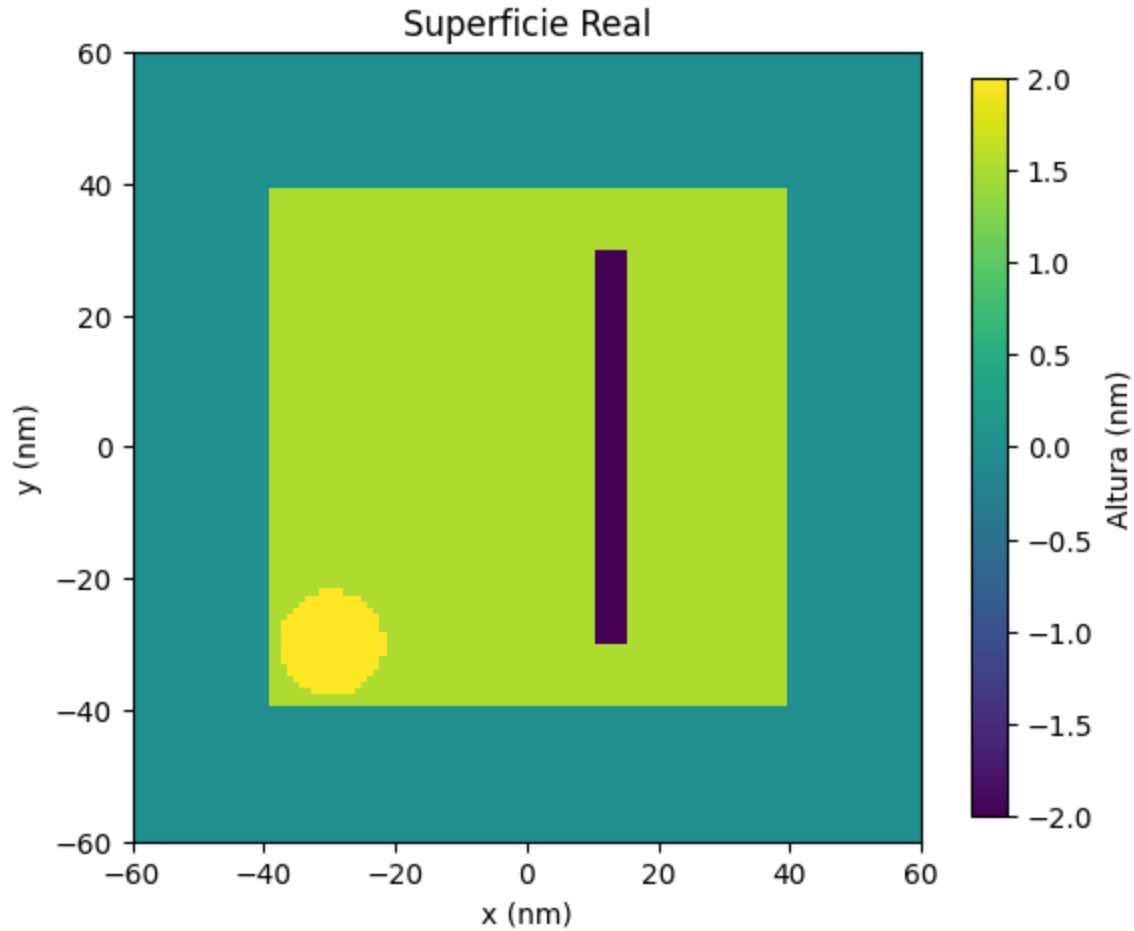
*Resultados – Diapositiva 12*



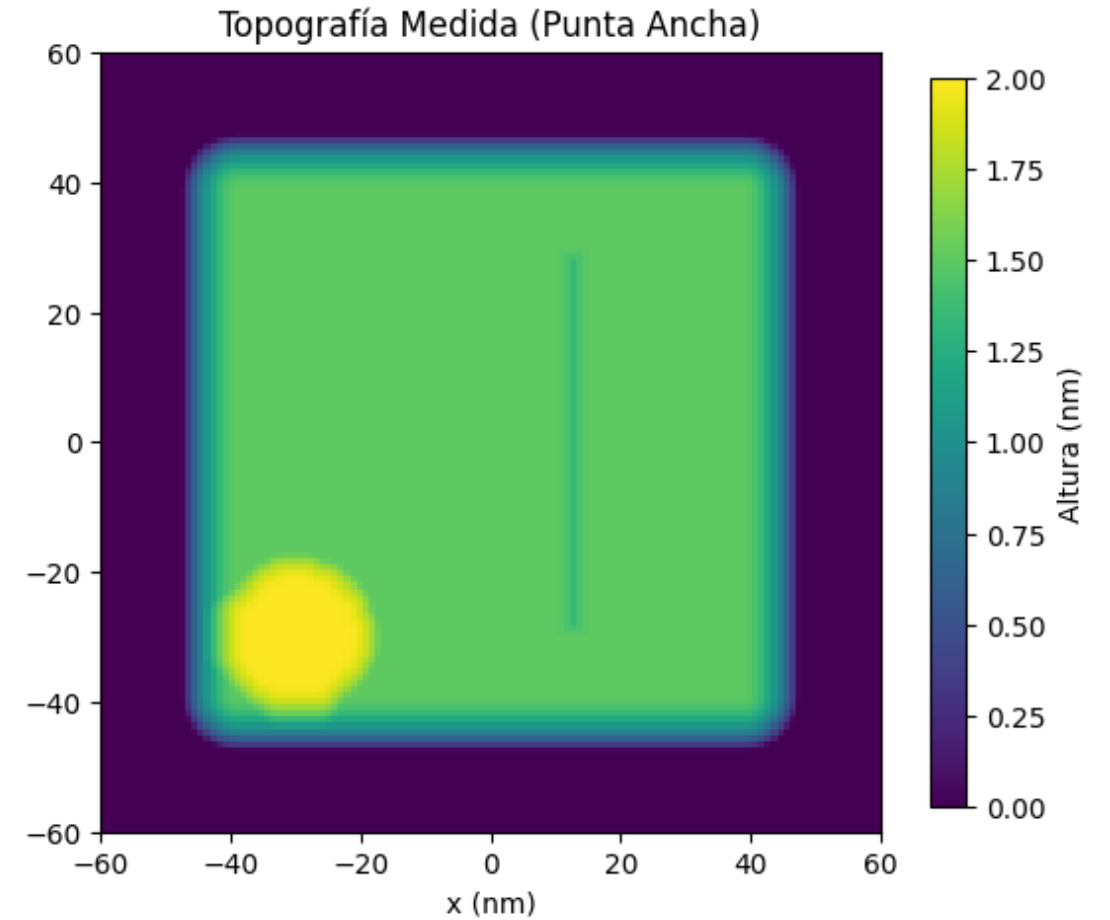
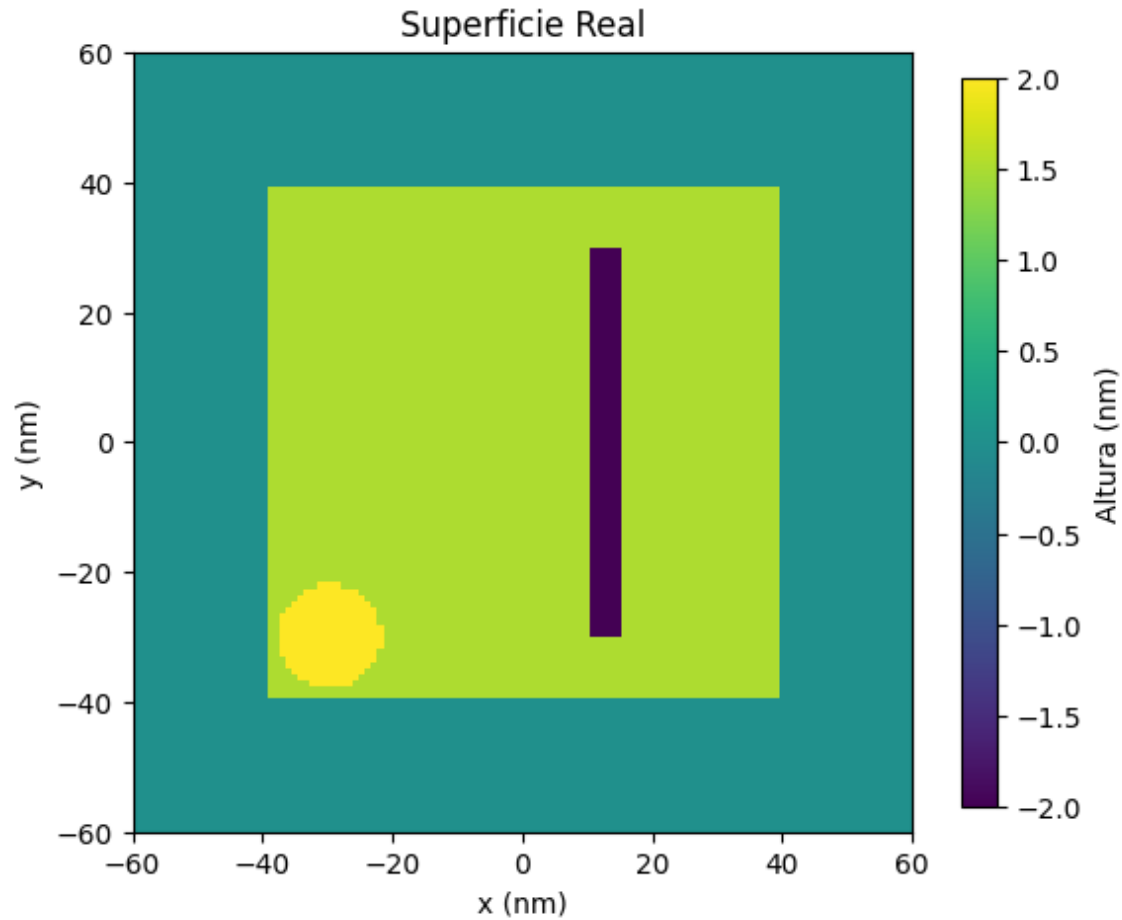




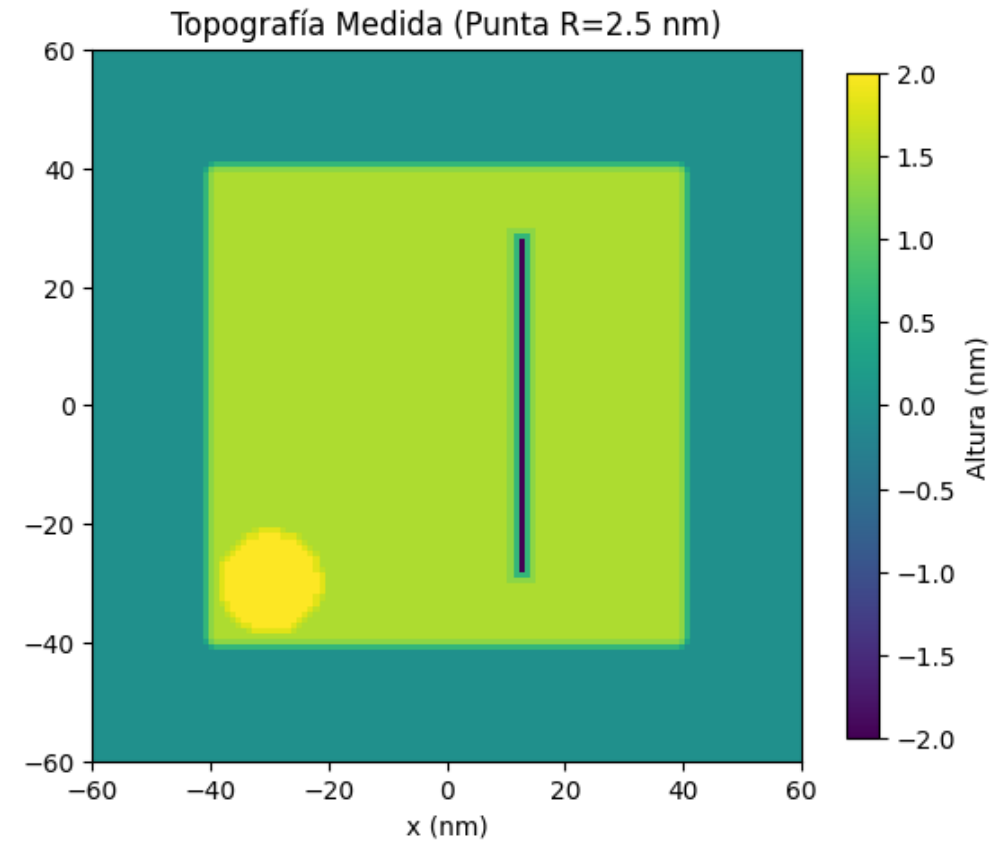
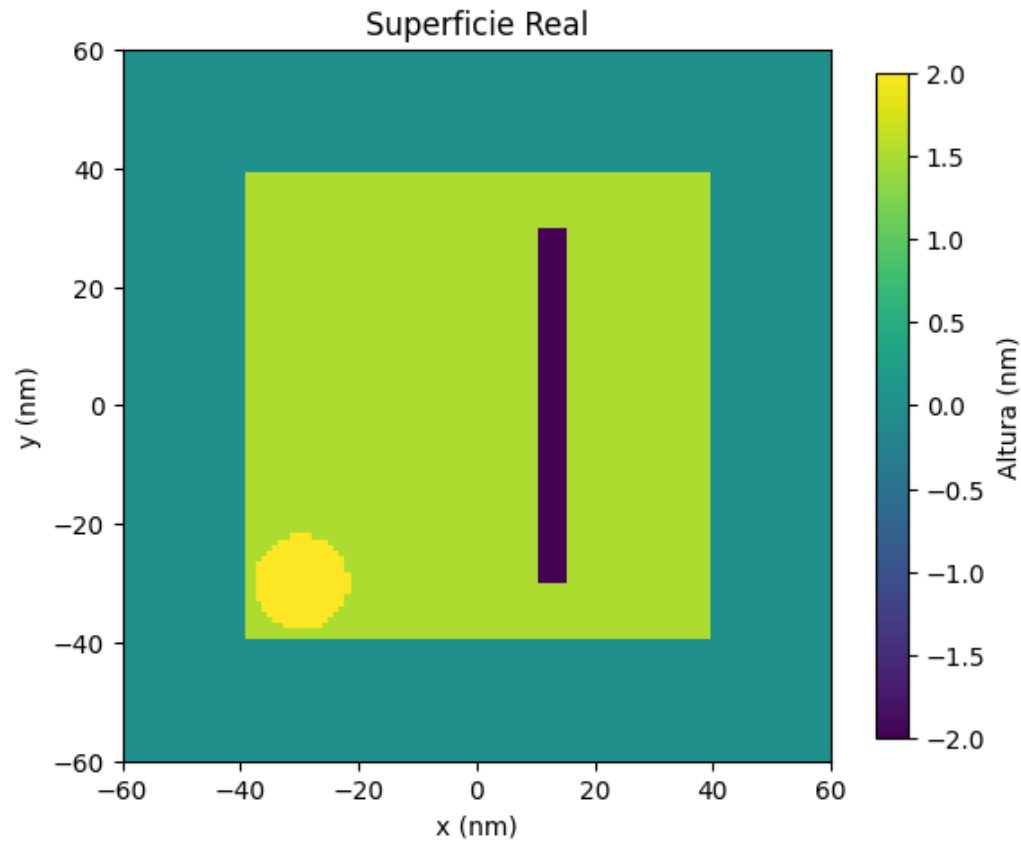
## Comparativa 2D (Vista Superior)



## Efecto de una Punta Ancha (Radio = 25 nm)



## Efecto de una Punta de Radio = 2.5 nm



# CONCLUSIÓN





