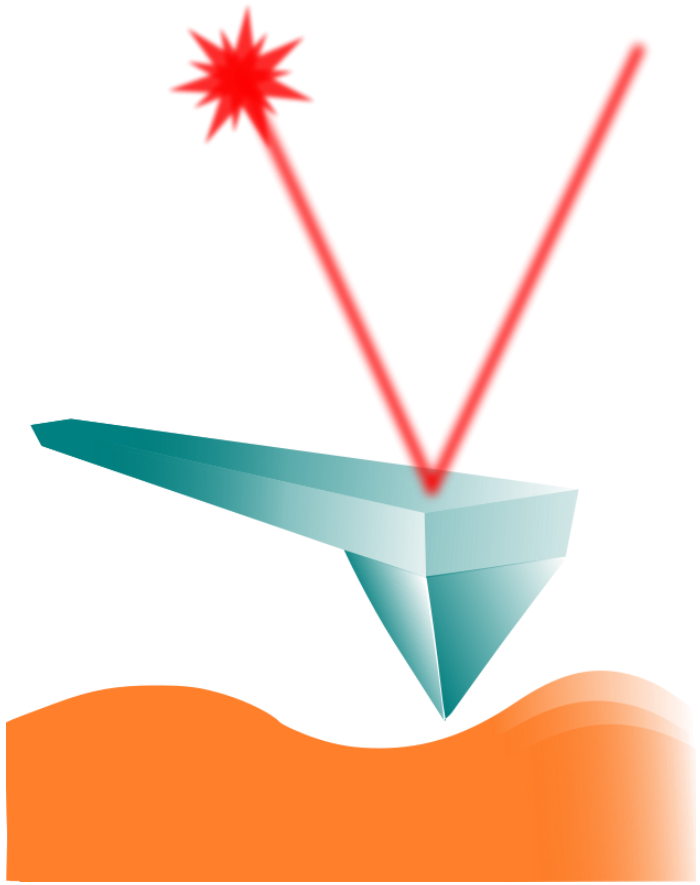
Simulación del Efecto de Convolución de Punta en Microscopia de Fuerzas Atómicas (AFM)

Rafael Jiménez Parra NIU : 1638106

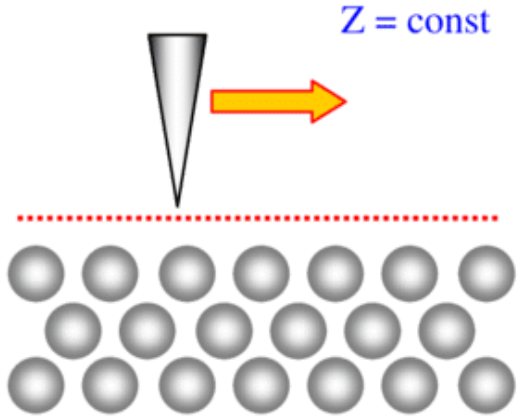
UAB Universitat Autònoma de Barcelona

INTRODUCCIÓN

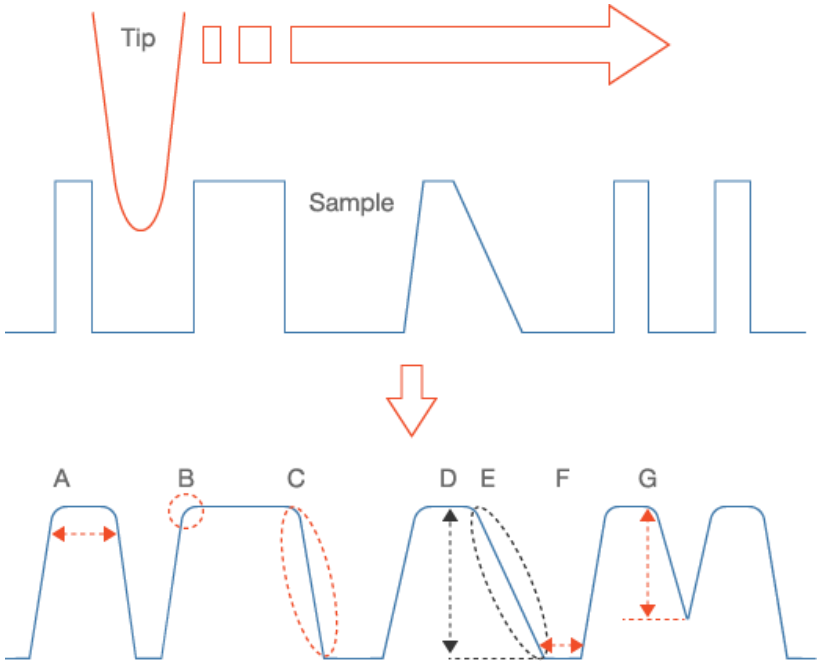
Atomic Force Microscope

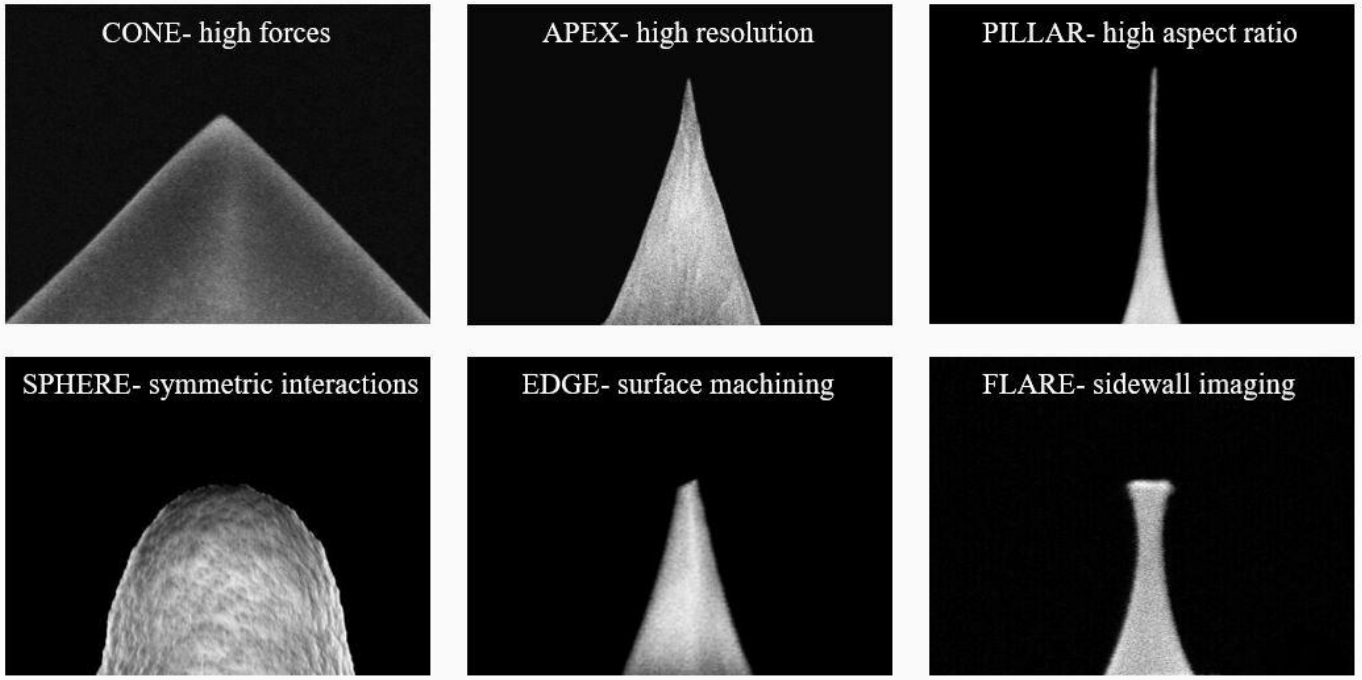


Distancia Punta Muestra	Fuerza Dominante	Rango de Fuerza
+10 / 100 micrómetros	Nada	-
pocos micrómetros	Fuerzas Eléctricas	±1 nN ~ ±1μN
menos de pocos nanómetros	Van der Waals atractivas Capilaridad	±100 nN ~ ±10μN
Contacto	Fuerzas Pauli Repulsivas Fuerzas Mecánicas	±1 nN ~ ±10 μN



Efecto Convolución





OBJETIVOS

Desarrollo modelo computacional de AFM

Modelar el efecto de convolución

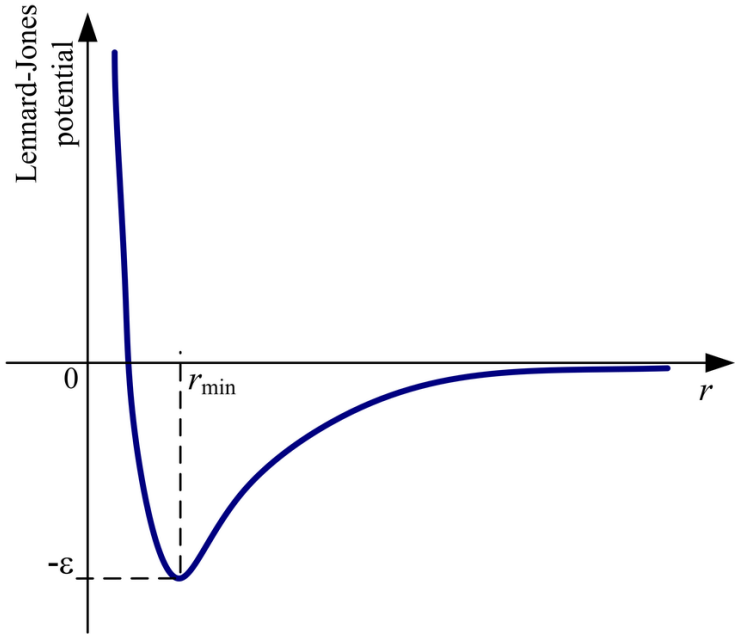
Repetir metodología para otras muestras

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

```python
# 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 importáncia
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

```python
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 - Superfície

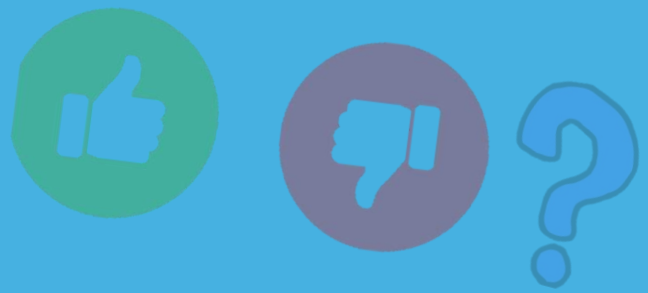
```python
def crear_superficie_muestra(tamano, resolucion):
    """Crea una superficie de prueba con algunas características."""
    x = np.linspace(-tamano/2, tamano/2, resolucion)
    y = np.linspace(-tamano/2, tamano/2, resolucion)
    X, Y = np.meshgrid(x, y)
    Z = np.zeros_like(X)

    # 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

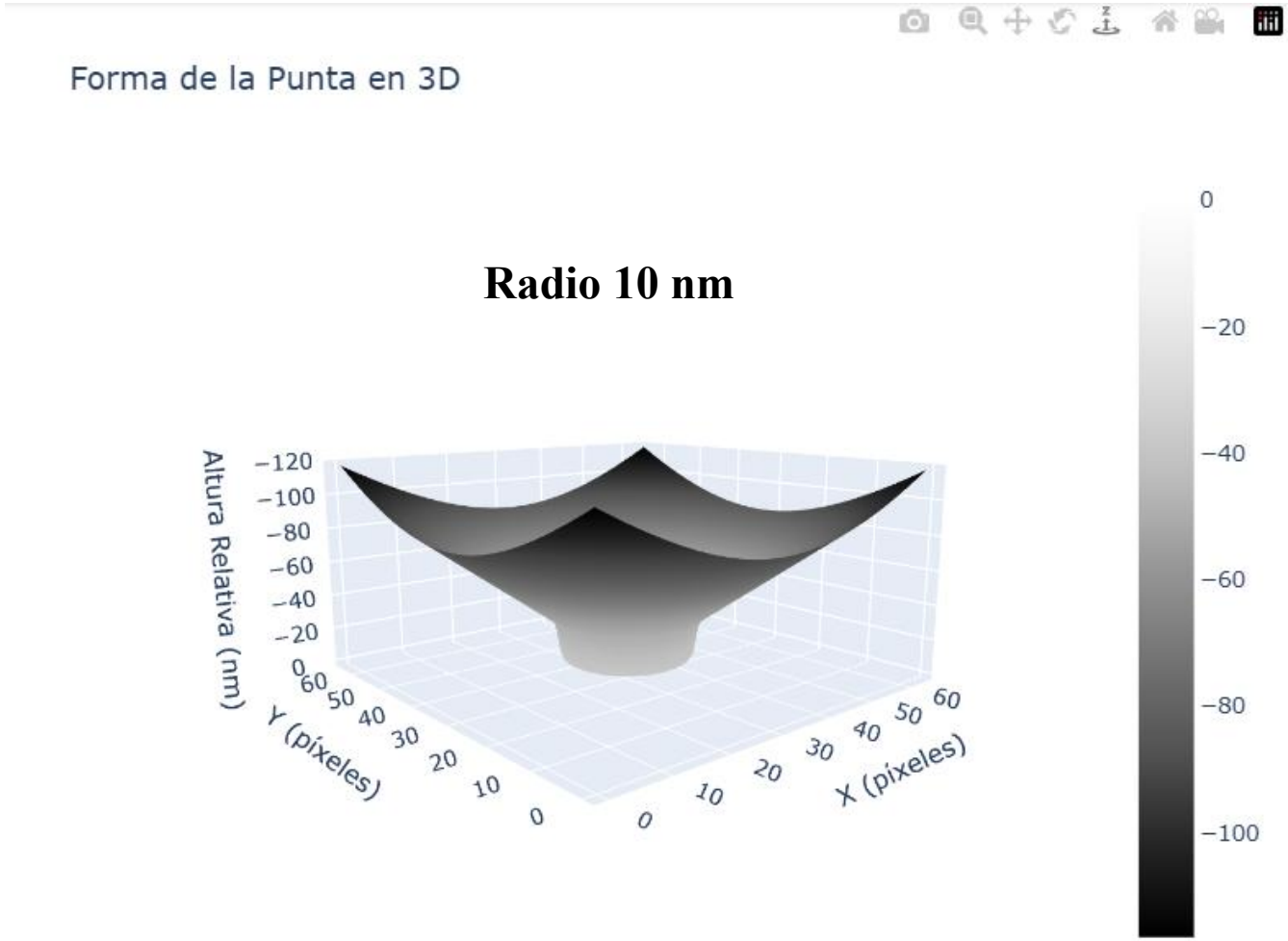
```python
def crear_forma_punta(radio, angulo, tamano, resolucion, pixels=65):
    """Crea la geometría 3D de la punta."""
    nm_por_pixel = tamano / resolucion
    centro = pixels // 2

    # 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
```

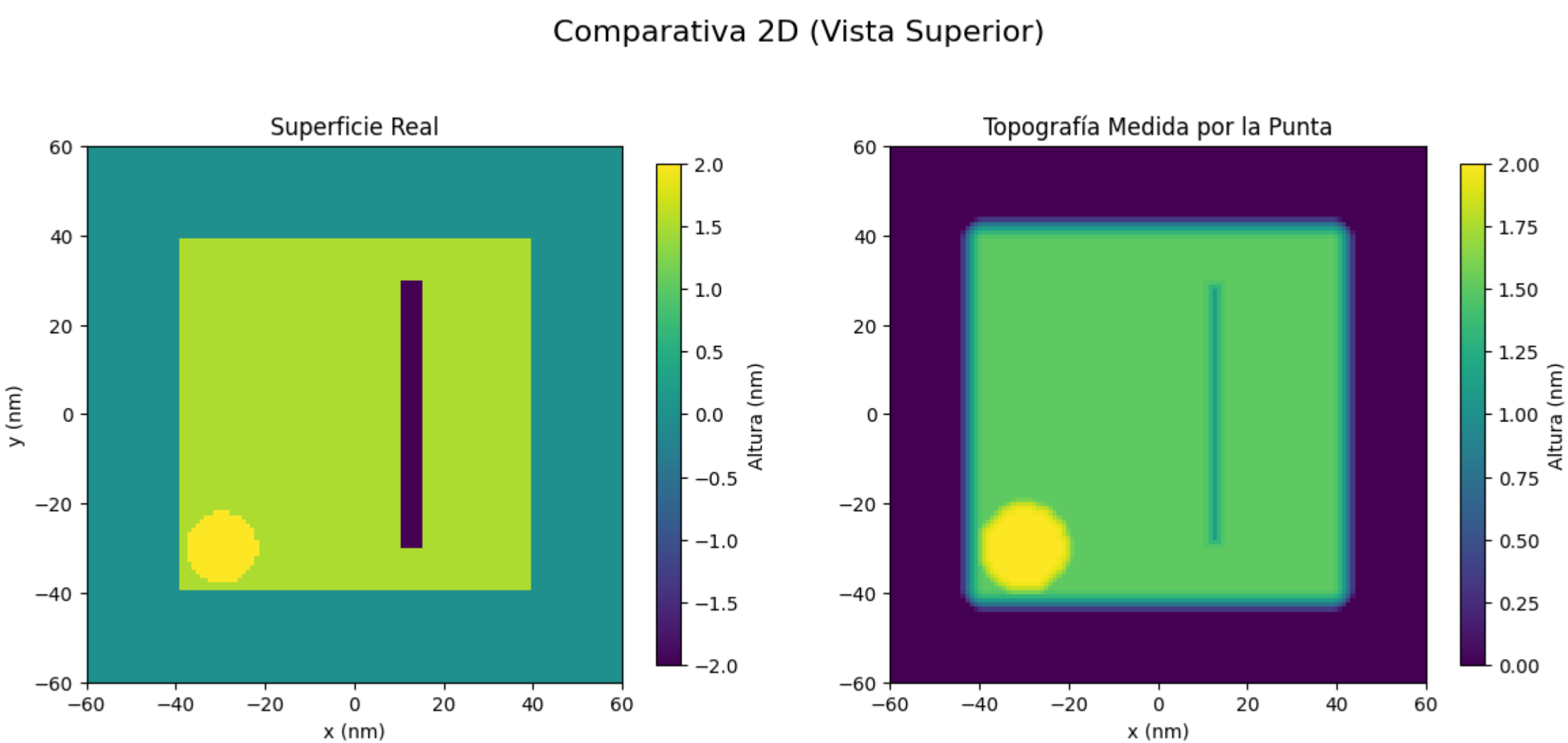
Código – Convolución + Campo Fuerzas

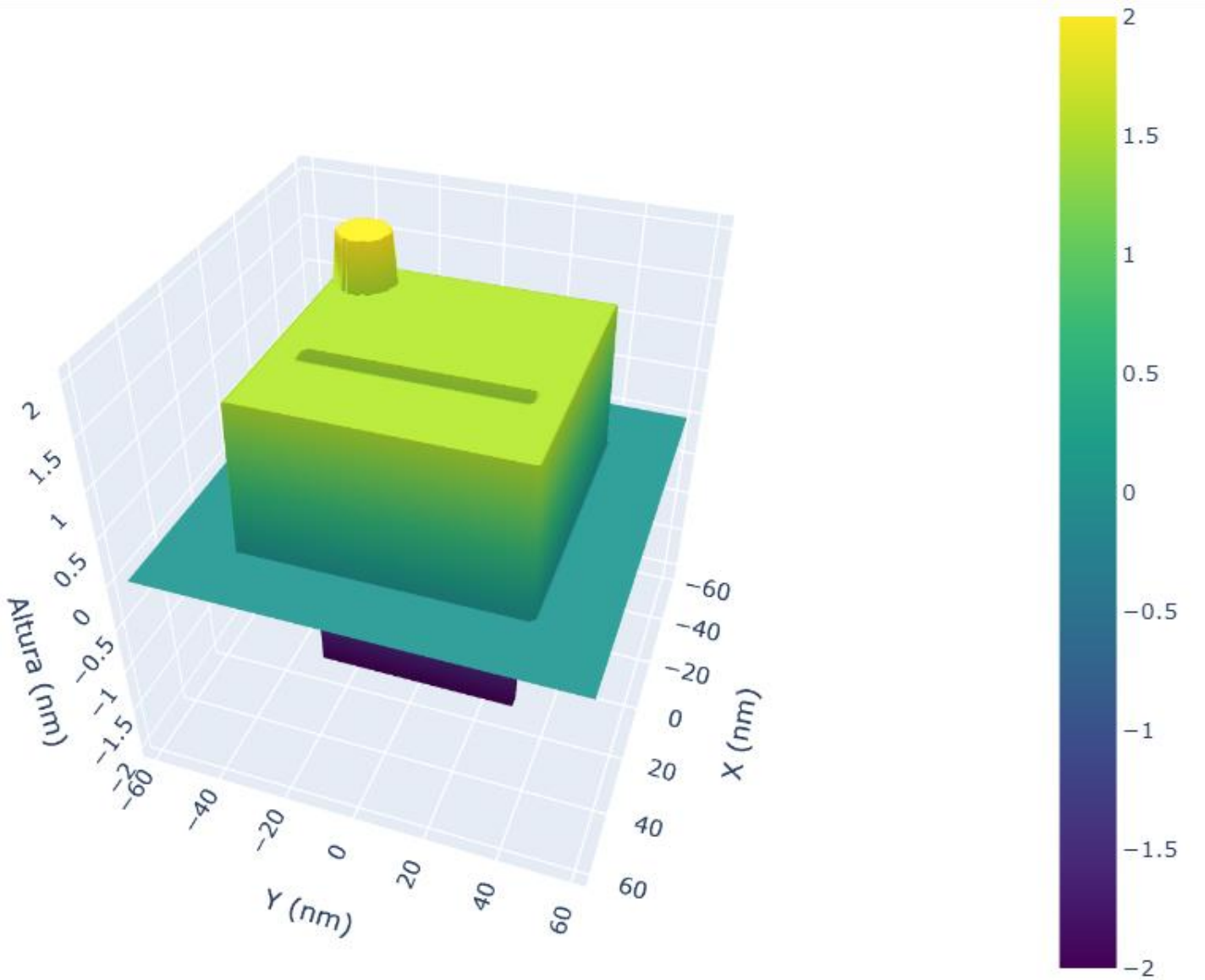
```
print("Primero se genera la superfície y punta")
X, Y, Z_real = crear_superficie_muestra(TAMANO, RESOLUCION)
punta = crear_forma_punta(RADIO_PUNTA, ANGULO_PUNTA, TAMANO, RESOLUCION)

# 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)
```

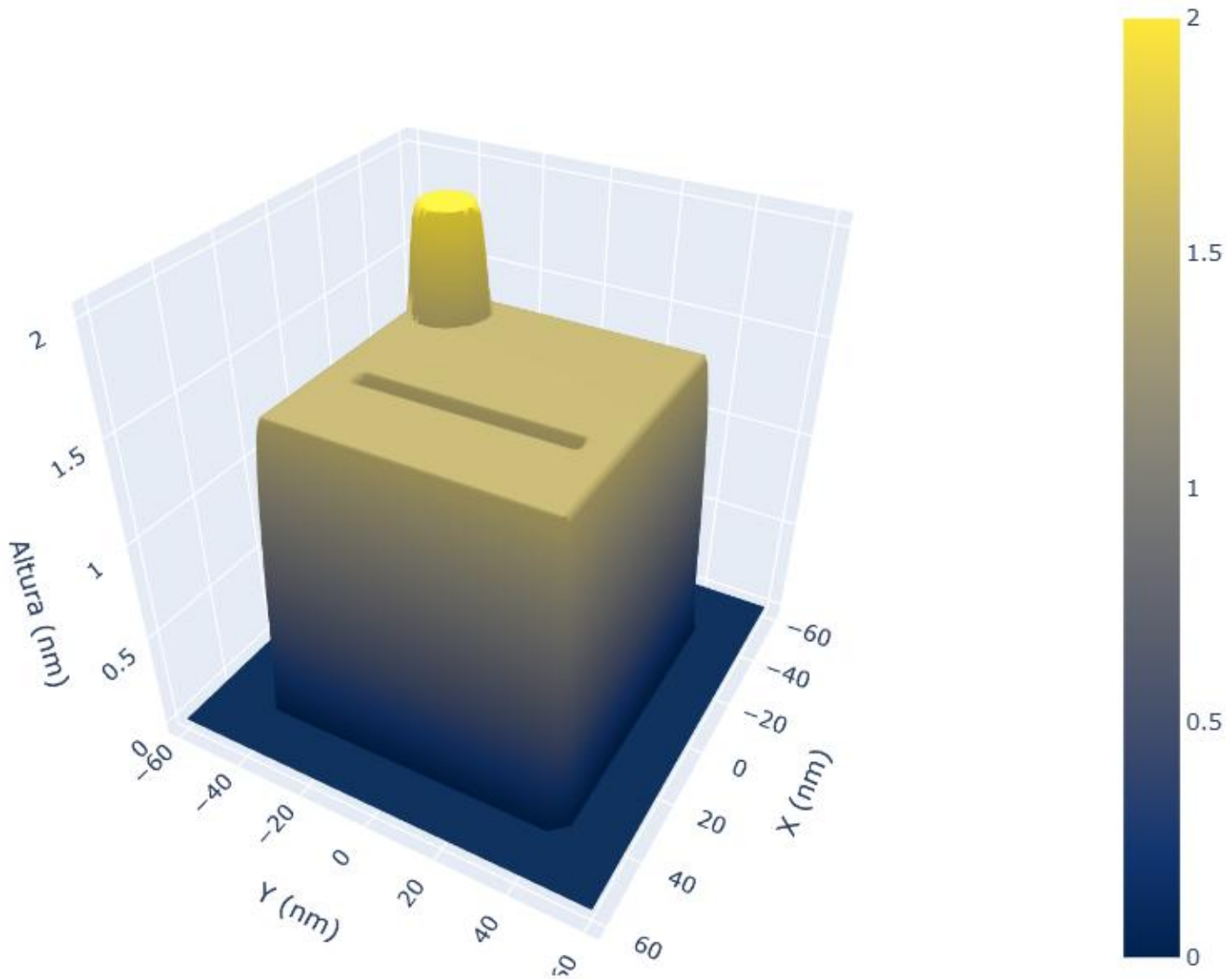


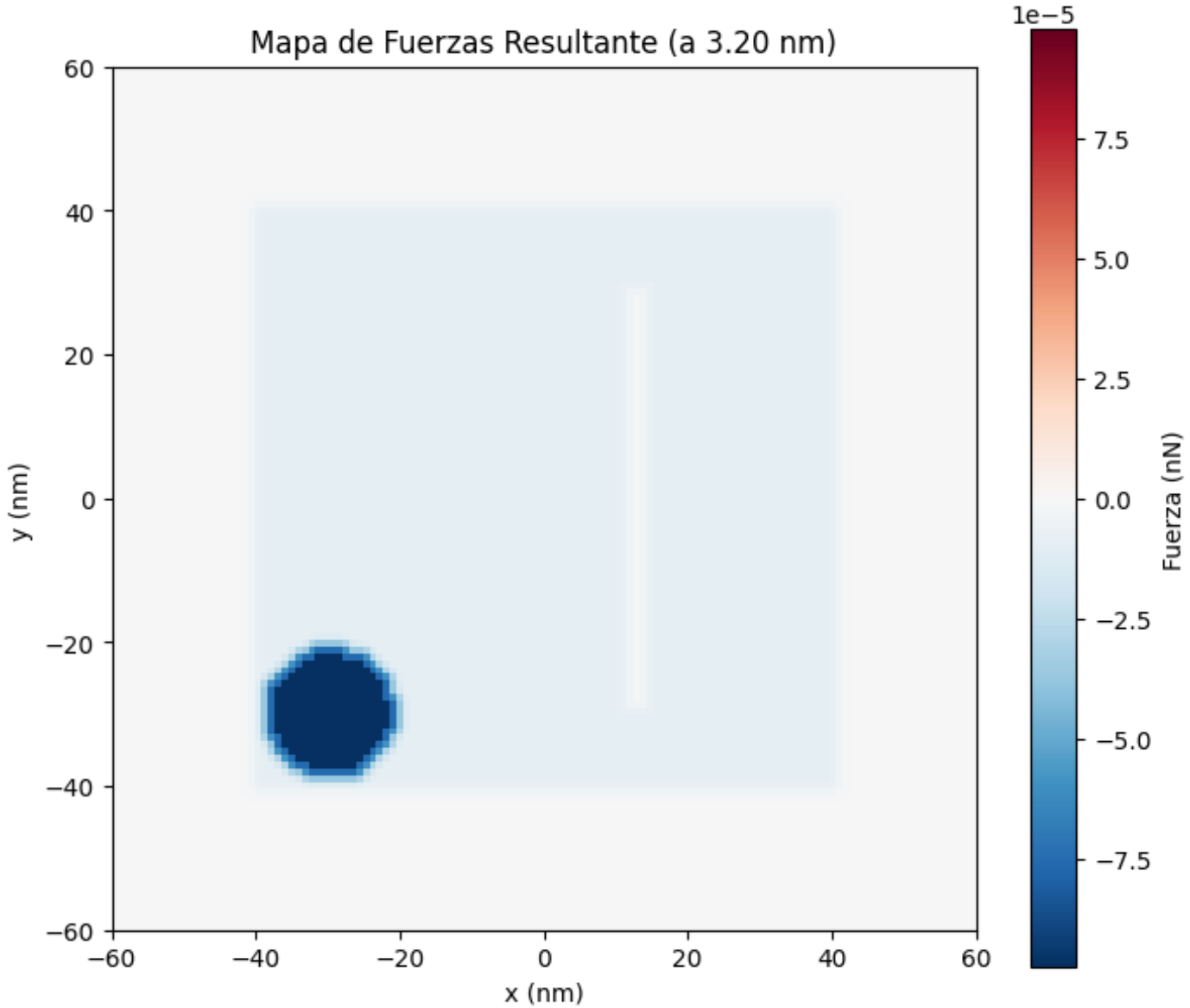
RESULTADOS


Forma de la Punta en 3D
Radio 10 nm
Altura Relativa (nm)
−120
−100
−80
−60
−40
−20
0
Y (píxeles)
0
10
20
30
40
50
60
X (píxeles)
0
10
20
30
40
50
60
0
−20
−40
−60
−80
−100

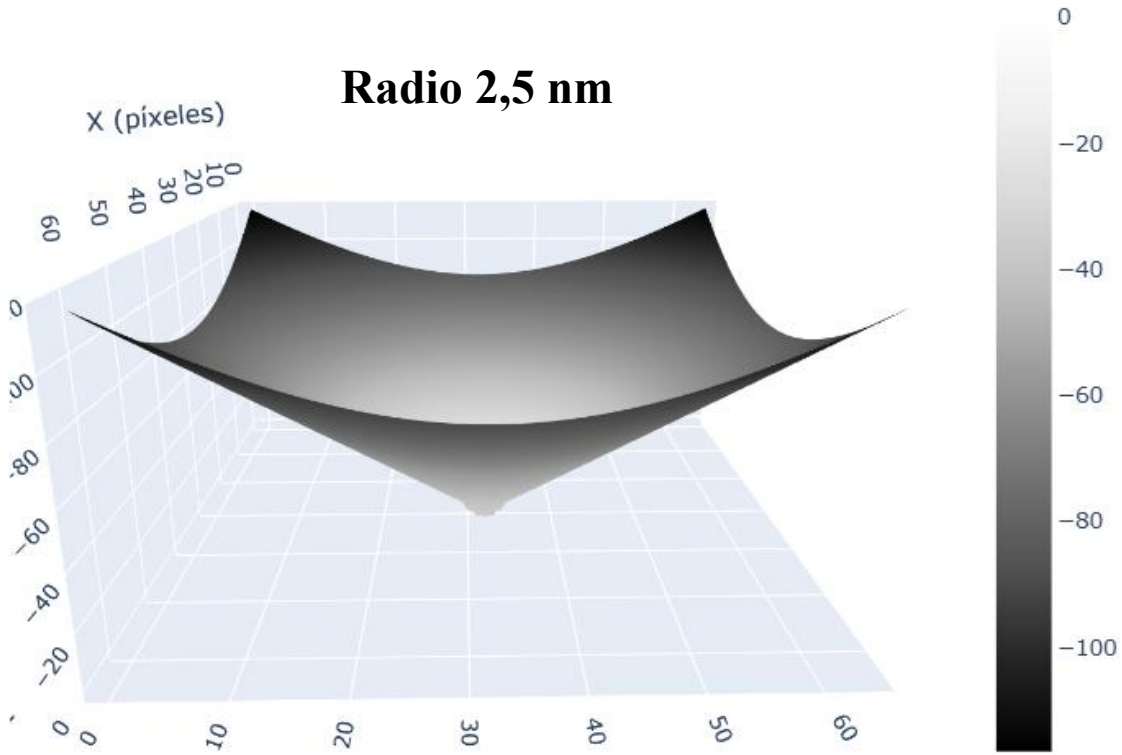

Comparativa 2D (Vista Superior)
Superficie Real
Topografía Medida por la Punta
y (nm)
x (nm)
Altura (nm)
60
40
20
0
−20
−40
−60
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00


Altura (nm)
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
Y (nm)
−60
−40
−20
0
20
40
60
X (nm)
−60
−40
−20
0
20
40
60
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2

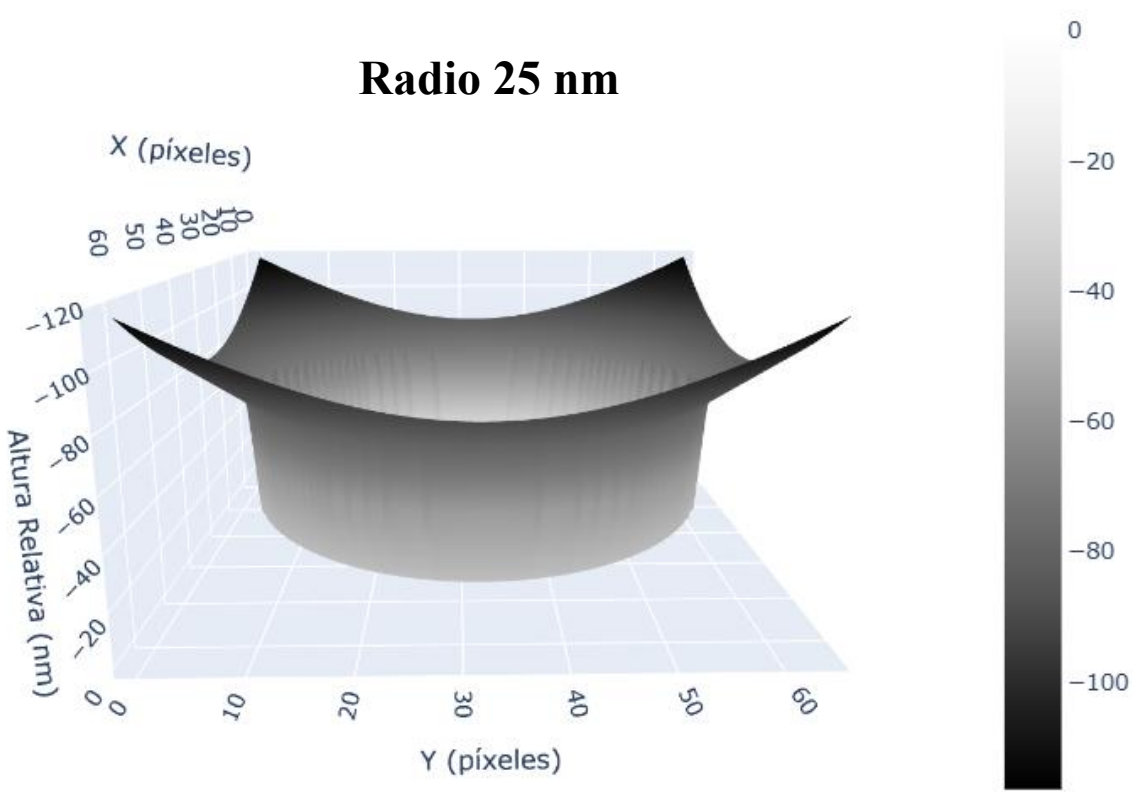

Altura (nm)
2
1.5
1
0.5
0
−60
−40
−20
0
20
40
60
Y (nm)
X (nm)
2
1.5
1
0.5
0

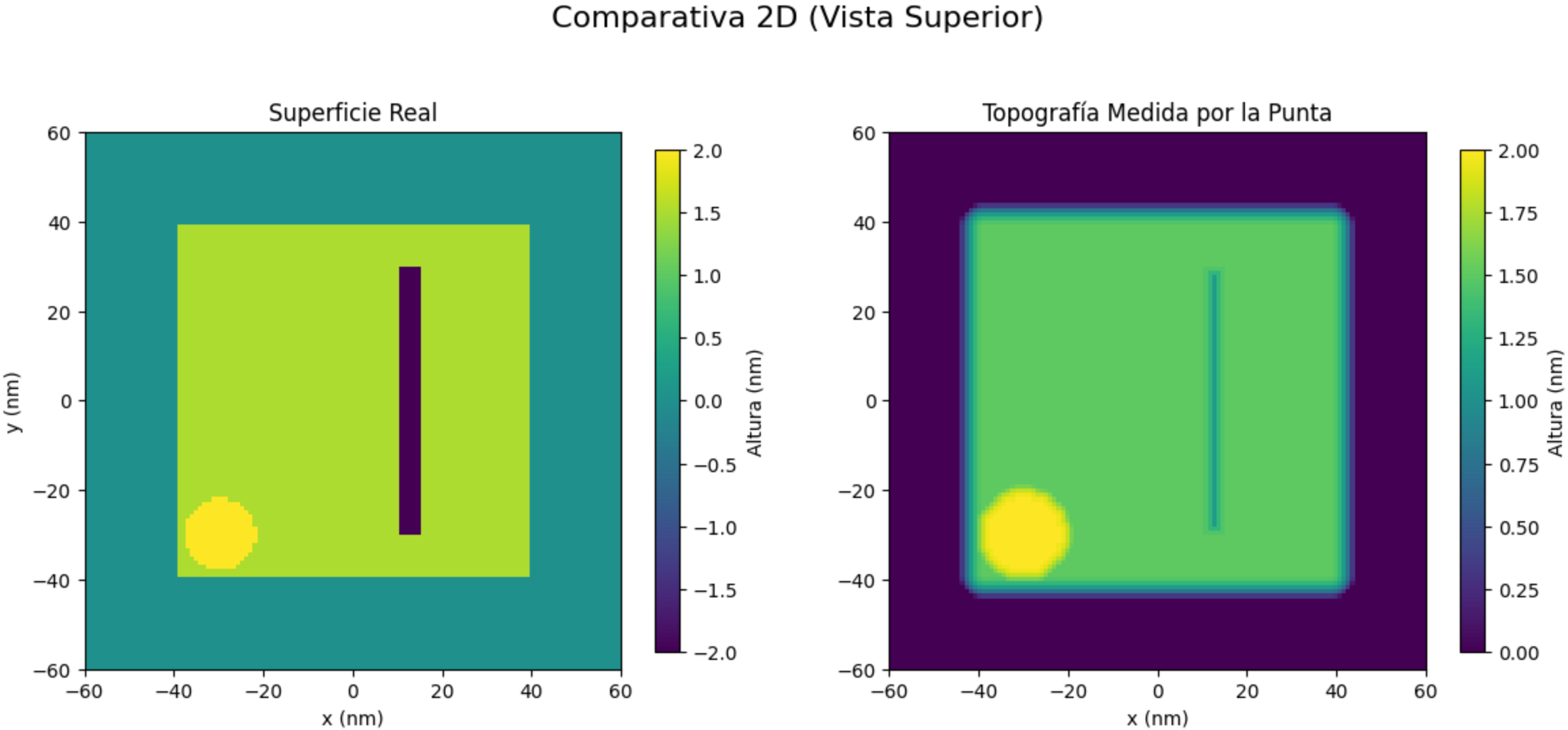

Mapa de Fuerzas Resultante (a 3.20 nm)
1e−5
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
Fuerza (nN)
60
40
20
0
−20
−40
−60
y (nm)
−60
−40
−20
0
20
40
60
x (nm)

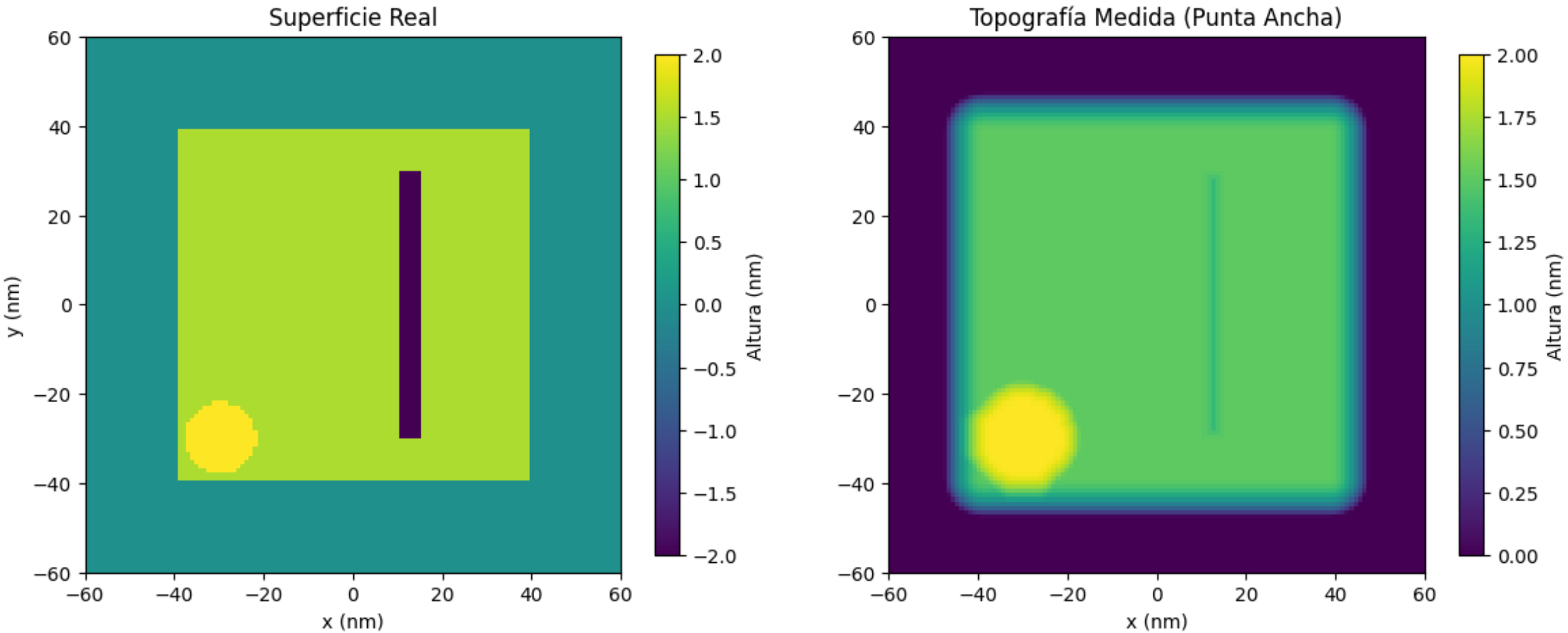
Radio 2,5 nm



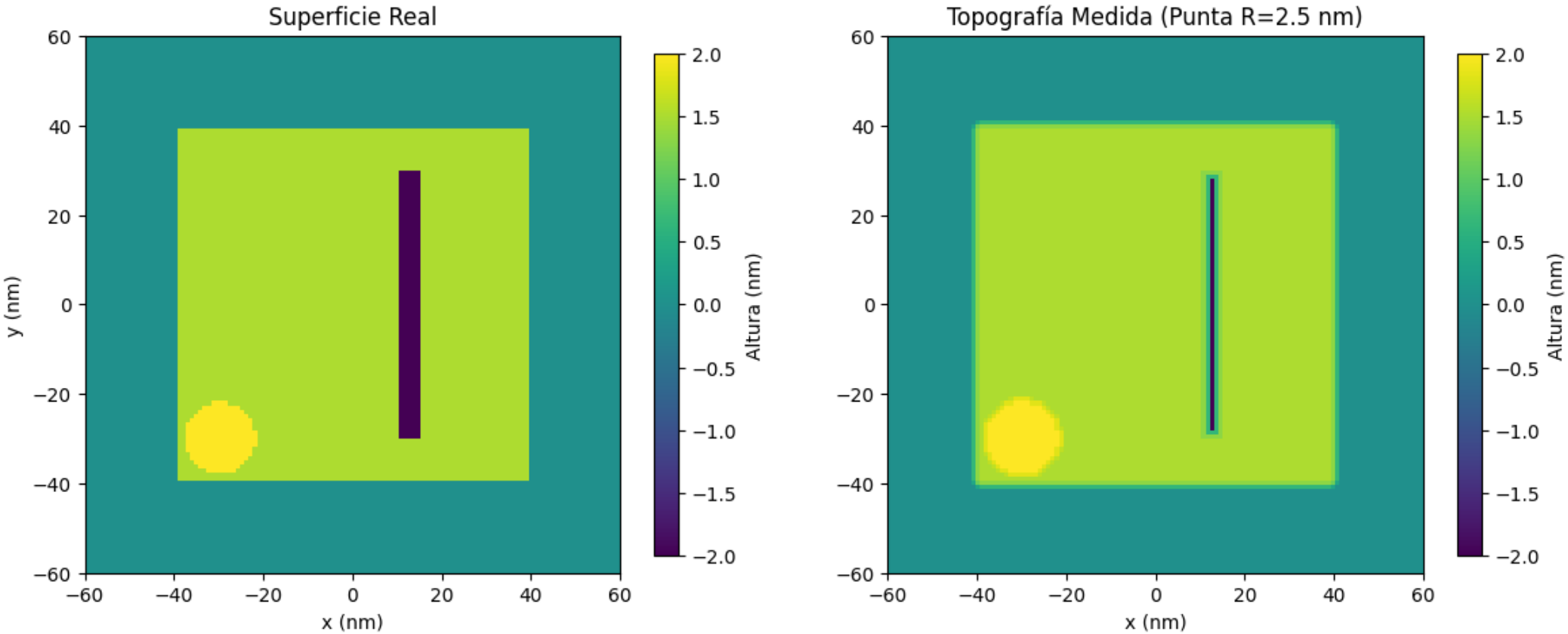
Radio 25 nm



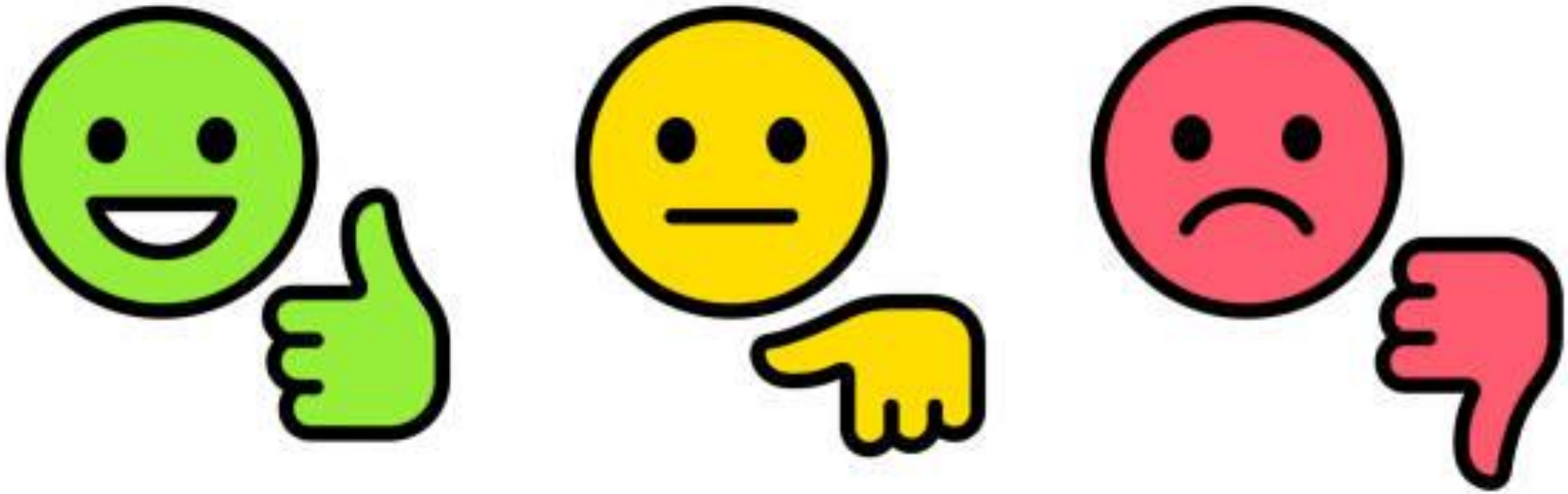

Comparativa 2D (Vista Superior)
Superficie Real
Topografía Medida por la Punta
y (nm)
x (nm)
Altura (nm)
60
40
20
0
−20
−40
−60
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00


Efecto de una Punta Ancha (Radio = 25 nm)
Superficie Real
Topografía Medida (Punta Ancha)
y (nm)
x (nm)
Altura (nm)
60
40
20
0
−20
−40
−60
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

Efecto de una Punta de Radio = 2.5 nm



CONCLUSIÓN



FIN