Escuela Politécnica Nacional

Nombre: Dany Molina

Fecha: 18/05/2025

Tarea 5 - Método de Newton y la Secante

Ejercicio 6

La función descrita por $f(x) = ln(x^2 + 1) - e^{0.4x}cos(\pi x)$ tiene un número infinito de ceros.

a. Determine, dentro de 10^{-6}, el único cero negativo.

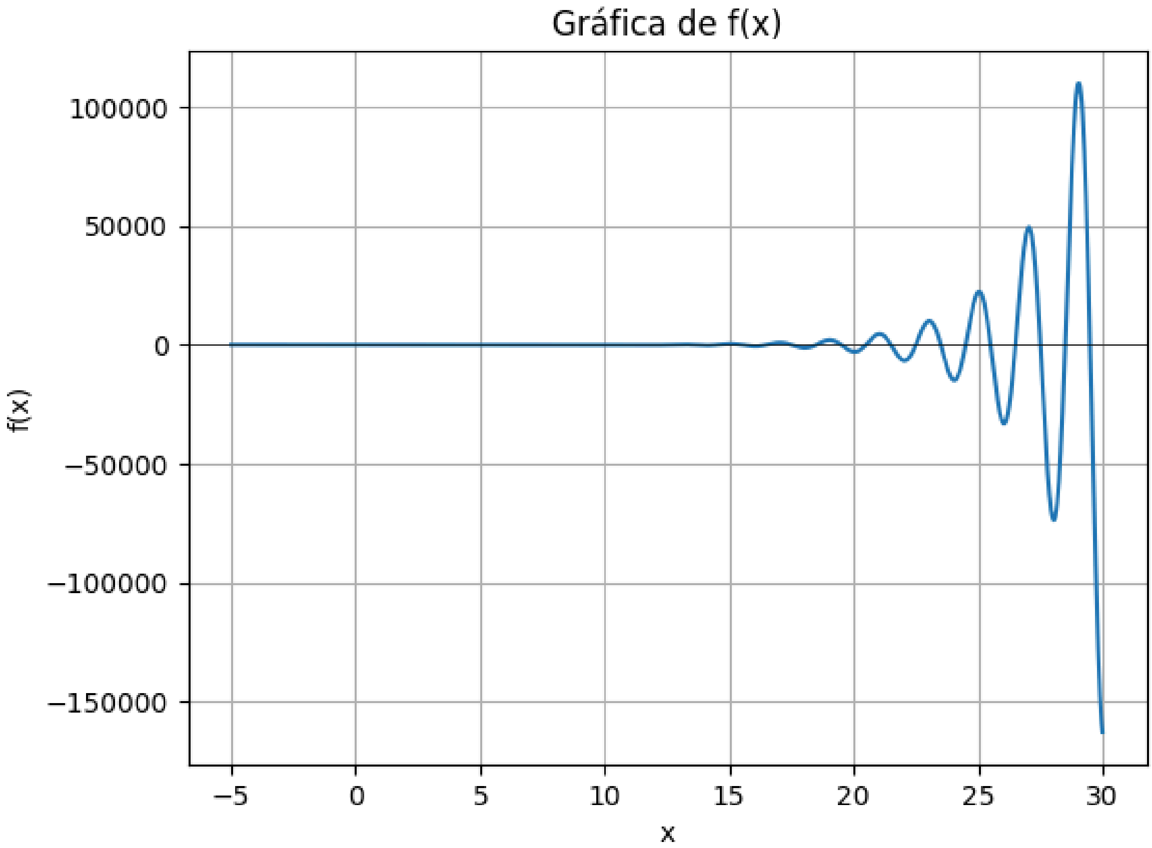
Gráfica de los ceros de la funcion

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return np.log(x**2 + 1) - np.exp(0.4*x) * np.cos(np.pi * x)

# Gráfica de f(x) en un rango
x_vals = np.linspace(-5, 30, 1000)
y_vals = f(x_vals)

plt.plot(x_vals, y_vals)
plt.axhline(0, color='black', linewidth=0.5)
plt.title('Gráfica de f(x)')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.grid(True)
plt.show()
```



In [5]:
```python
def secante(f, x0, x1, tol=1e-6, max_iter=100):
    for _ in range(max_iter):
        if abs(f(x1) - f(x0)) < 1e-12:
            return None  # Para evitar división por cero
        x2 = x1 - f(x1)*(x1 - x0)/(f(x1) - f(x0))
        if abs(x2 - x1) < tol:
            return x2
        x0, x1 = x1, x2
    return None

raiz_negativa = secante(f, -1.5, 0)
print("Cero negativo:", raiz_negativa)
```

```
Cero negativo: -0.43414304728574216
```

b. Determine, dentro de 10^{-6}, los cuatro ceros positivos más pequeños.

In [7]:
```python
ceros_positivos = []
intervalos = [(0.5,1.5), (1.5,2.5), (2.5,3.5), (3.5,4.5)]

for a, b in intervalos:
    cero = secante(f, a, b)
    ceros_positivos.append(cero)

for i, r in enumerate(ceros_positivos, 1):
    print(f"Cero positivo #{i}: {r}")
```

```
Cero positivo #1: 0.4506567478885528
Cero positivo #2: 0.45065674789001126
Cero positivo #3: 1.7447380534186228
Cero positivo #4: -0.4341430472606062
```

c. Determine una aproximación inicial razonable para encontrar el enésimo cero positivo más pequeño de f. [Sugerencia: Dibuje una gráfica aproximada de f.]

Analizando la gráfica, observamos que los ceros positivos están aproximadamente espaciados por cerca de 1 unidad, debido a la periodicidad de $cos(\pi x)$.

Entonces deducimos lo siguiente.

$$x_n \approx \quad npara \quad n \in N$$

Por lo que la aproxmiación inicial razonable para el enésimo cero ppositivo mas pequeño es:

$$x_n \approx n$$

d. Use la parte **c)** para determinar, dentro de 10^-6, el vigesimoquinto cero positivo más pequeño de f.

In [9]:
```
cero_25 = secante(f, 24.5, 25.5)
print("Cero positivo #25:", cero_25)
```

```
Cero positivo #25: -55.60501722333146
```