Laboratório 01 - Expansão em Séries de Taylor

Aluno: Daniel Medeiros Soares Carneiro (20220012347)

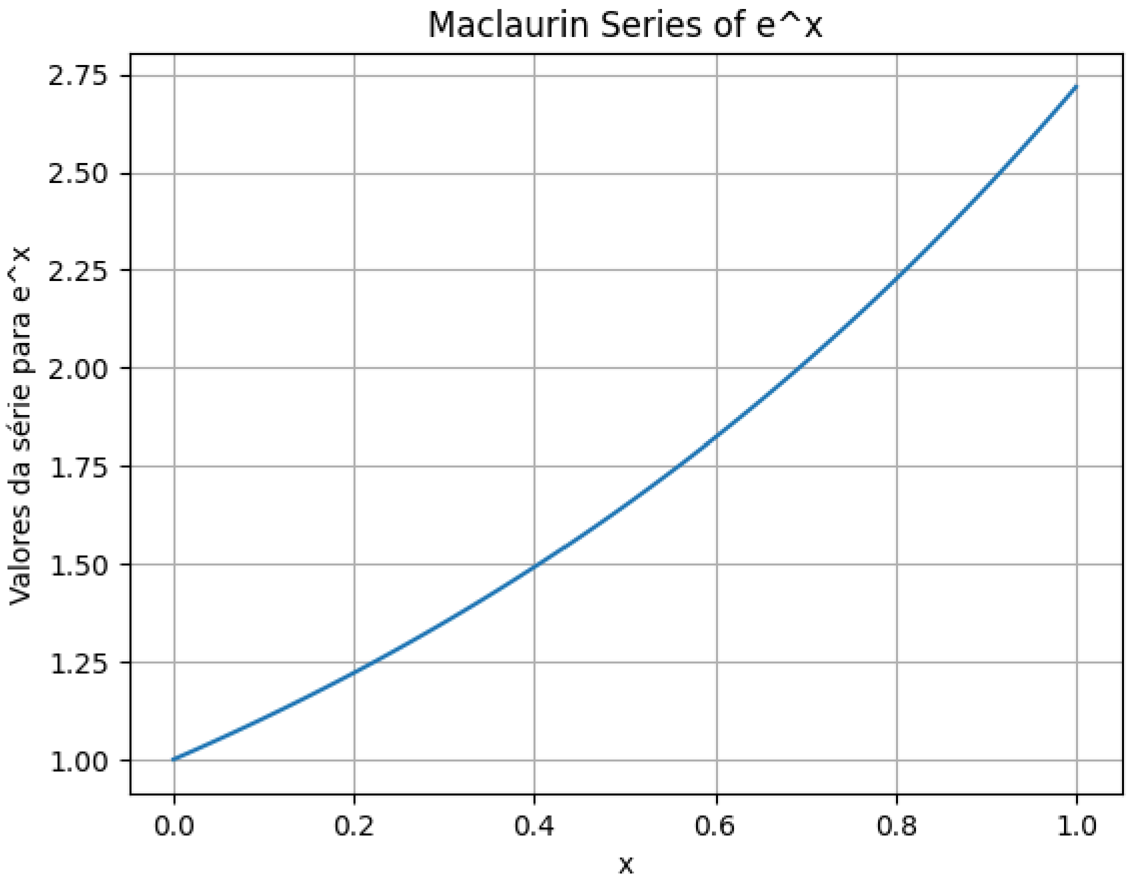
Exemplo 01 - Expansão da função $f(x) = e^x$, via Série de MacLaurin

$$f(x) = e^x \approx \sum_{k=0}^{N} \frac{x^k}{k!}$$

In [35]:

```
import numpy as np
import matplotlib.pyplot as plt
import math

def maclaurin(x, num=10):
    soma = 0
    for k in range(num+1):
        expressao = (x ** k) / math.factorial(k)
        soma += expressao
    return soma

x = np.linspace(0, 1, 100)
k = 10
resultado = maclaurin(x,k)
print(f"e^x ≈ {resultado[-1]}")

plt.plot(x, resultado)
plt.xlabel("x")
plt.ylabel("Valores da série para e^x")
plt.title("Maclaurin Series of e^x")
plt.grid(True)
plt.show()
```

```
e^x ≈ 2.7182818011463845
```



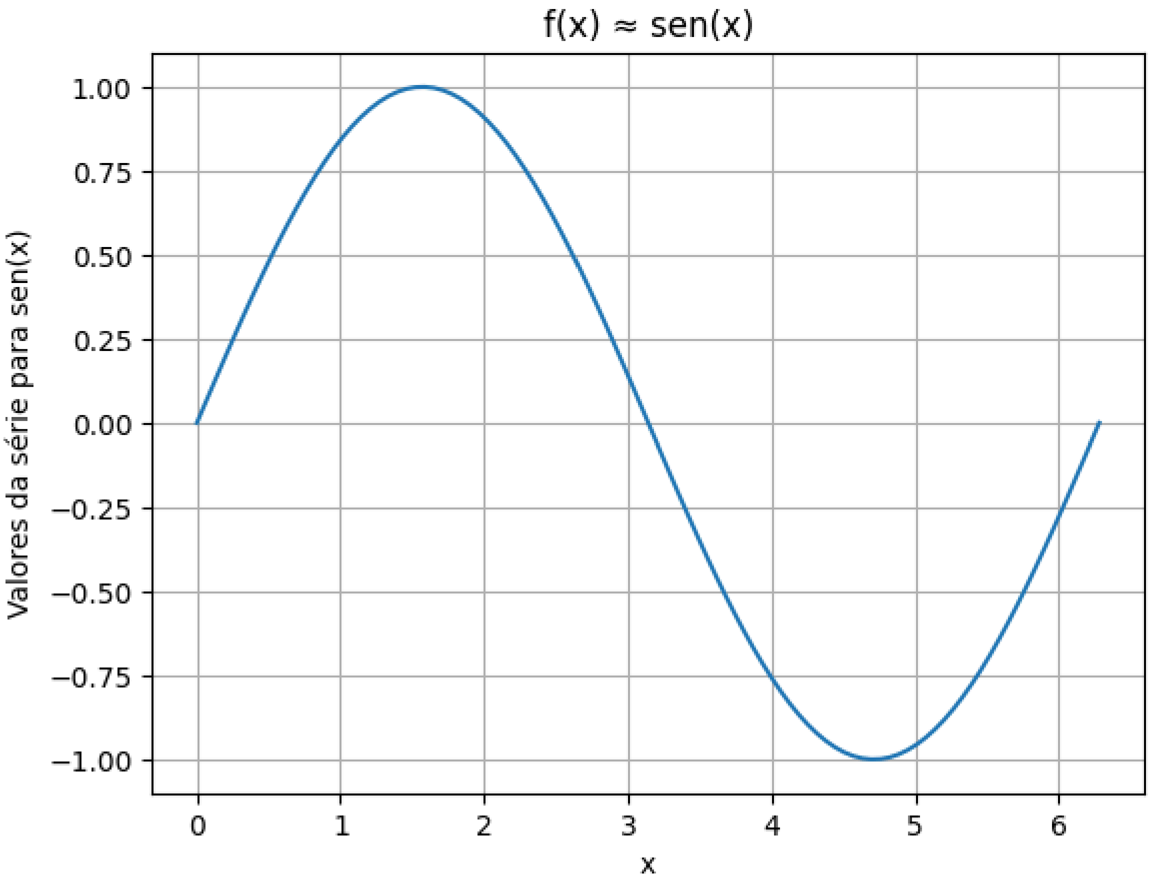
Exemplo 02 - Implementar a função

$$f(x) = sen(x) \approx \sum_{k=0}^{N} (-1)^k \frac{x^{2k+1}}{(2k+1)!} \tag{1}$$

In [36]:

```
import numpy as np
import matplotlib.pyplot as plt
import math

def senx(x, num=10):
    soma = 0
    for k in range(num+1):
        expressao = (-1)**k * (pow(x, 2*k+1)) / math.factorial(2*k +1)
        soma += expressao
    return soma


x = np.linspace(0, 2*(math.pi), 100)
k = 10
resultado = senx(x, k)
print(f"sen(x) ≈ {resultado[-1]}")


plt.plot(x, resultado)
plt.xlabel("x")
plt.ylabel("Valores da série para sen(x)")
plt.title(f"f(x) ≈ sen(x)")
plt.grid(True)
plt.show()
```

```
sen(x) ≈ 8.274095221328433e-05
```



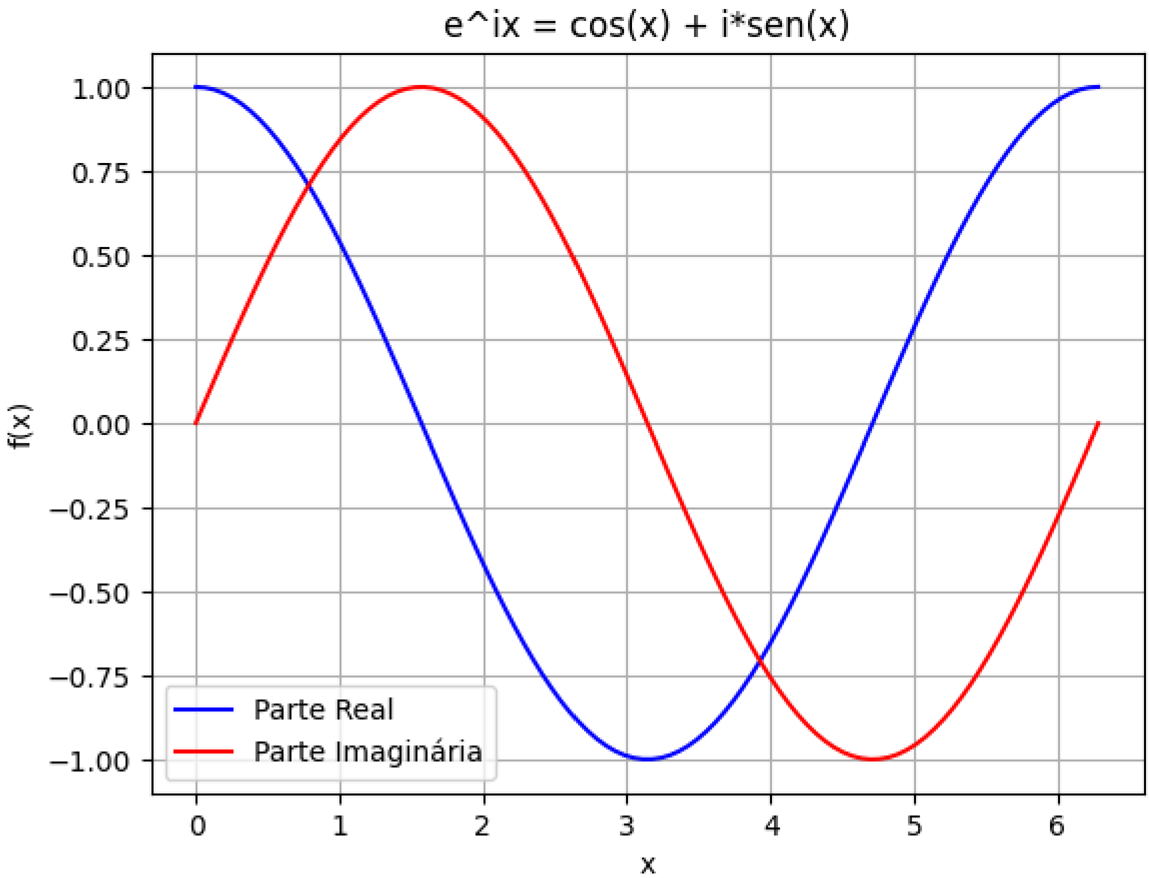
Exemplo 03 : Use a expansão em Série de MacLaurin para mostrar que

$$e^{ix} = cos\, x + i\, sen\, x, \quad i = \sqrt{-1}. \tag{2}$$

In [37]:

```
import numpy as np
import matplotlib.pyplot as plt
import math

def euler(x, termos = 20):
    soma = np.zeros_like(x, dtype=complex)
    for n in range(termos+1):
        expressao = (1j*x)**n / math.factorial(n)
        soma += expressao
    return soma

x = np.linspace(0, 2*(math.pi), 100)
n = 20
resultado = euler(x, n)
print(f"e^ix ≈ {resultado[-1]}")

plt.plot(x, np.real(resultado), label="Parte Real", color="blue")
plt.plot(x, np.imag(resultado), label="Parte Imaginária", color="red")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("e^ix = cos(x) + i*sen(x)")
plt.grid(True)
plt.legend()
plt.show()
```

```
e^ix ≈ (1.000301224041832-0.0010481827960275355j)
```



Exercícios

01 - O valor de π pode ser aproximado por meio da seguinte expansão em série:

$$\pi \approx 4 \sum_{i=1}^{n} (-1)^{i-1} \frac{1}{2i-1} \tag{3}$$

escreva uma função que retorne o valor aproximado de π que receba como parâmetro de entrada os n termos da série. Calcule o erro relativo para diferentes valores de n. Por exemplo: $n = 5,\ 10,\ 20,\ 30$.

In [38]:
```
import numpy as np
import matplotlib.pyplot as plt
import math

def erro_relativo(a, b):
    return np.abs(a - b) / np.abs(a)

def pi(n):
    soma = 0
    for i in range(1, n+1):
        expressao = (-1)**(i-1) * 1/(2*i - 1)
        soma += expressao
    return 4*soma


valores_n = [5, 10, 15, 20, 25, 30, 35, 40]
```

```
for n in valores_n:
    valor_pi_aprox = pi(n)
    valor_pi_real = math.pi
    Er = erro_relativo(valor_pi_real, valor_pi_aprox)
    print(f"n = {n}:")
    print(f"Valor aproximado de pi: {valor_pi_aprox}")
    print(f"Erro relativo: {Er}\n")
```

```
n = 5:
Valor aproximado de pi: 3.3396825396825403
Erro relativo: 0.06305396909634241

n = 10:
Valor aproximado de pi: 3.0418396189294032
Erro relativo: 0.031752377109236 43

n = 15:
Valor aproximado de pi: 3.208185652261944
Erro relativo: 0.021197209827969625

n = 20:
Valor aproximado de pi: 3.09162380666784
Erro relativo: 0.015905577976462196

n = 25:
Valor aproximado de pi: 3.1815766854350325
Erro relativo: 0.0127273126258272

n = 30:
Valor aproximado de pi: 3.108268566698947
Erro relativo: 0.010607386305403936

n = 35:
Valor aproximado de pi: 3.1701582571925884
Erro relativo: 0.009092714031577038

n = 40:
Valor aproximado de pi: 3.116596556793833
Erro relativo: 0.007956504726161022
```

02 - Use a expansão em série de Taylor para mostrar que

$$\frac{sen\,x}{x} \approx 1 \tag{4}$$

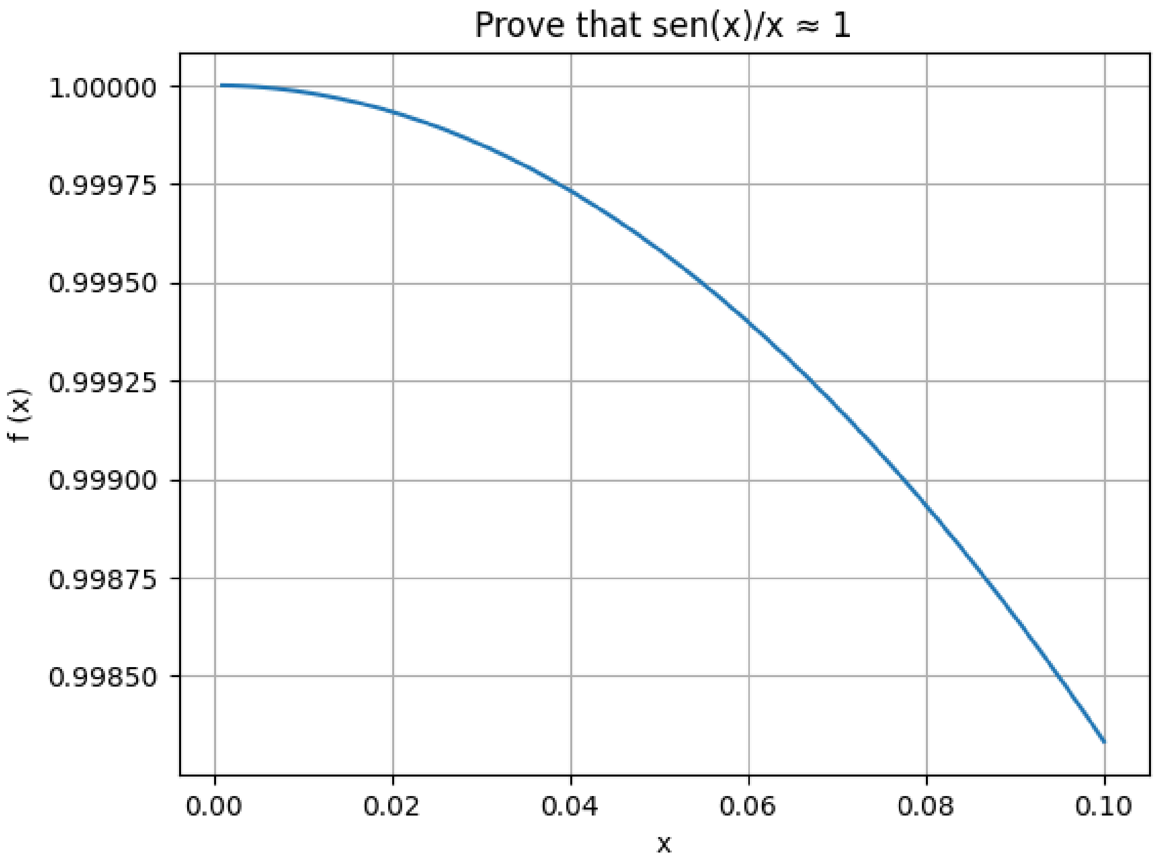
com um valor bem pequeno em x.

In [39]:

```
import numpy as np
import matplotlib.pyplot as plt
import math

def limite_fundamental(x, termos = 20):
    soma = 0
    for k in range(termos+1):
        expressao = (-1)**k * x**(2*k) / math.factorial(2*k + 1)
        soma += expressao
    return soma
```

```
x = np.linspace(0.1, 0.001, 100)
k = 20
resultado = limite_fundamental(x, k)
print(f"sen(x)/x ≈ {resultado[-1]}")

plt.plot(x, resultado)
plt.xlabel("x")
plt.ylabel("f (x)")
plt.title("Prove that sen(x)/x ≈ 1")
plt.grid(True)
plt.show()
```

```
sen(x)/x ≈ 0.9999998333333416
```

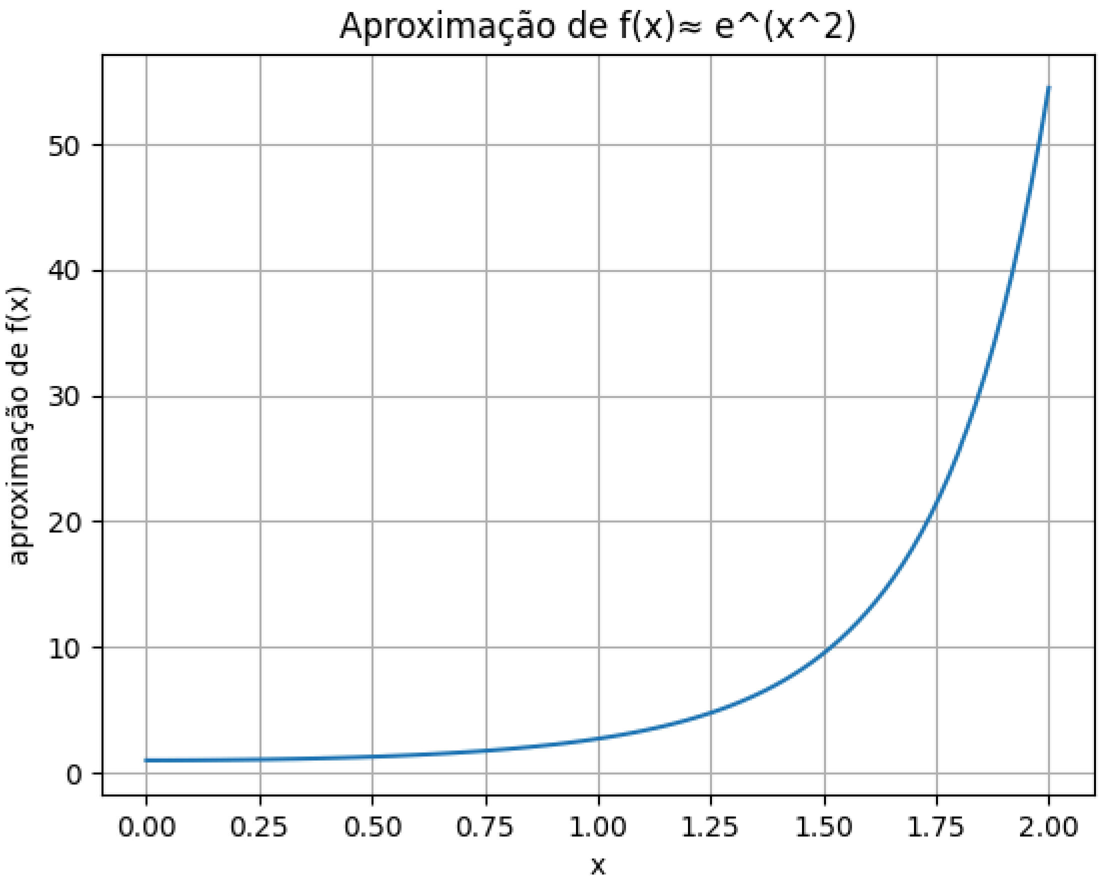


03 - Escreva a expansão em série de Taylor para $f(x) = e^{x^2}$ em torno da origem. Defina uma função $exp\_dupla(x, n)$, para calcular uma aproximação para $f(x)$ utilizando os n primeiros termos da expansão. Adapte a função para receber como parâmetro de entrada um vetor.

In [40]:
```
import numpy as np
import matplotlib.pyplot as plt
import math

def exp_dupla(x, n):
    x = np.array(x)
    soma = np.zeros_like(x, dtype= float)

    for k in range(n+1):
        expressao = pow(x**2,k) / math.factorial(k)
        soma += expressao
    return soma

x = np.linspace(0, 2, 100)
```

```
k = 10
resultado = exp_dupla(x, k)
print(f"e^(x^2)= {resultado[-1]}")

plt.plot(x, resultado)
plt.xlabel("x")
plt.ylabel("aproximação de f(x)")
plt.title(f"Aproximação de f(x)≈ e^(x^2)")
plt.grid(True)
plt.show()
```

```
e^(x^2)= 54.44310405643739
```



04 - Sabendo que o seno hiperbólico pode ser obtido por meio da seguinte expansão,

$$senh\, x = \sum_{k=0}^{\infty} \frac{x^{2k+1}}{(2k+1)!} \tag{5}$$

o cosseno hiperbólico por,

$$cosh\, x = \sum_{k=0}^{\infty} \frac{x^{2k}}{(2k)!} \tag{6}$$

e a tangente hiperbólica é dada por,

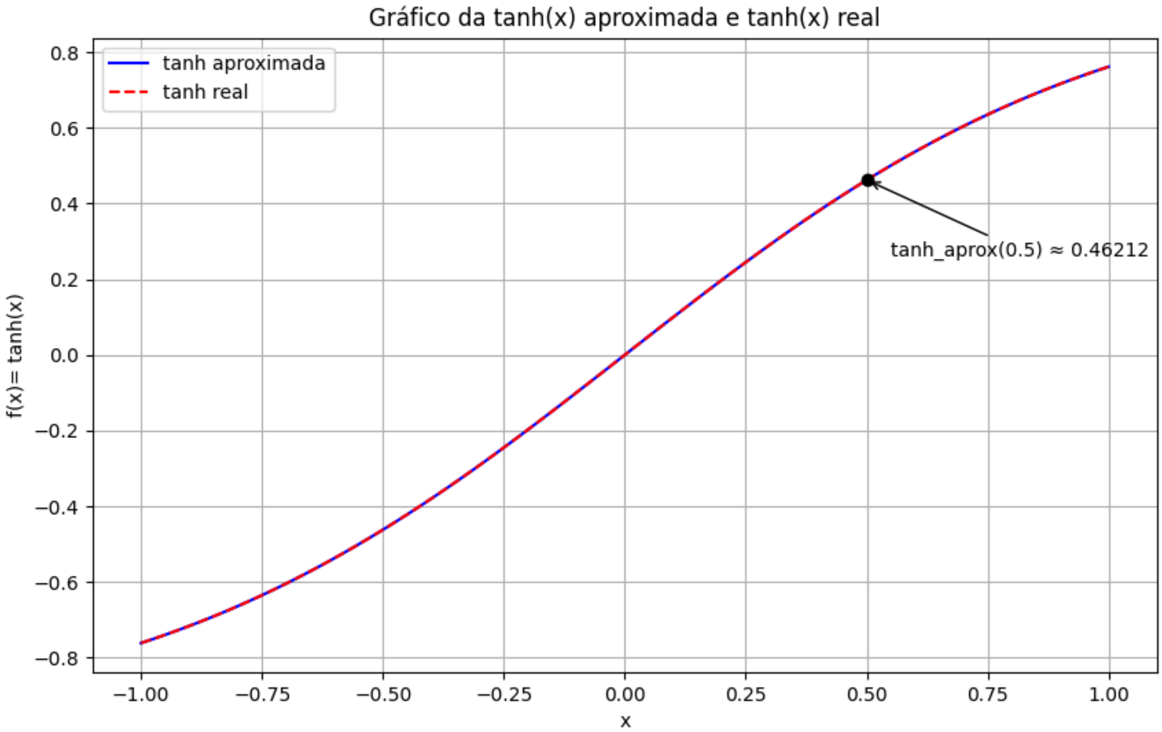
$$tanh\, x = \frac{senh\, x}{cosh\, x} \tag{7}$$

calcule a tangente hiperbólica de $x_0 = 0,5$ a partir das funções $senh\, x$ e $cosh\, x$, com um erro relatico de aproximadamente 10^{-5}. Gerar os gráficos e comparar com a função

$np.tanh(x)$.

In [41]:
```python
import numpy as np
import matplotlib.pyplot as plt
import math

def erro_relativo(a, b):
    return np.abs(a - b) / np.abs(a)


def tanh(x, erro_questao):
    n = 1
    while True:

        senh = sum(x**(2*k+1) / math.factorial(2*k+1) for k in range(n+1))

        cosh = sum(x**(2*k) / math.factorial(2*k) for k in range(n+1))

        tanh_aprox = senh / cosh
        tanh_real = np.tanh(x)
        Er = erro_relativo(tanh_real, tanh_aprox)

        if Er < erro_questao:
            break
        else:
            n += 1

    return tanh_aprox

vetor_x = np.linspace(-1, 1, 100)
resultado1 = [tanh(x, 1e-5) for x in vetor_x]
resultado2 = np.tanh(vetor_x)
print(f"tanh_aprox = {resultado1[-1]}")
print(f"tanh_real = {resultado2[-1]}")

x_da_questao = 0.5
tanh_aprox_0_5 = tanh(x_da_questao, 1e-5)
tanh_real_0_5 = np.tanh(x_da_questao)
print("\n")
print(f"tanh_aprox_0.5 = {tanh_aprox_0_5}")
print(f"tanh_real_0.5 = {tanh_real_0_5}")


plt.figure(figsize = (10, 6))
plt.plot(vetor_x, resultado1, label = "tanh aproximada", color="blue")
plt.plot(vetor_x, resultado2, linestyle="--", label = "tanh real", color ="red")
plt.scatter([x_da_questao], [tanh_aprox_0_5], color='black', zorder=5)
plt.annotate(f"tanh_aprox(0.5) ≈ {tanh_aprox_0_5:.5f}",
             xy=(x_da_questao, tanh_aprox_0_5),
             xytext=(x_da_questao + 0.05, tanh_aprox_0_5 - 0.2),
             arrowprops=dict(facecolor='black', arrowstyle="->"),
             fontsize=10, color='black')
plt.xlabel("x")
plt.ylabel("f(x)= tanh(x)")
plt.title("Gráfico da tanh(x) aproximada e tanh(x) real")
plt.grid(True)
plt.legend()
plt.show()
```

```
tanh_aprox = 0.7615942766625056
tanh_real = 0.7615941559557649


tanh_aprox_0.5 = 0.4621171922898218
tanh_real_0.5 = 0.46211715726000974
```




```
In [ ]:
```