PM002 - Atividade Computacional

Diogo Takamori Barbosa

July 2024

Uma metalúrgica está sendo contratada por uma fábrica de papel para projetar e construir um tanque retangular de aço, com base quadrada, sem tampa e com 500 m^3 de volume. O tanque será construído soldando-se chapas de aço umas às outras ao longo das bordas. O desafio é determinar as dimensões para a base e para a altura que farão o tanque pesar o mínimo possível.

(a) Descrição de como levar o peso em consideração para resolver o problema (custo-benefício):
Para minimizar o peso do tanque, devemos minimizar a área da superfície do tanque, uma vez que o peso do tanque é diretamente proporcional à quantidade de aço utilizado, que, por sua vez, é diretamente proporcional à área das chapas de aço usadas. Portanto, o objetivo é encontrar as dimensões da base e a altura que minimizam a área da superfície do tanque, mantendo o volume constante em 500 m^3.

(b) Fórmula $S(x)$ para a área do tanque em função da medida x do lado da base:
- x como o lado da base quadrada do tanque.
- h como a altura do tanque.

Sabemos que o volume V do tanque é dado por:

$$V = x^2 \cdot h$$

$$500 = x^2 \cdot h$$

$$h = \frac{500}{x^2}$$

A área da superfície S do tanque, que precisa ser minimizada, é composta pela área da base e pelas áreas das quatro paredes laterais.

Assim:

$$S(x) = x^2 + 4 \cdot (x \cdot h)$$

Substituindo h:

$$S(x) = x^2 + 4 \cdot x \cdot \frac{500}{x^2}$$

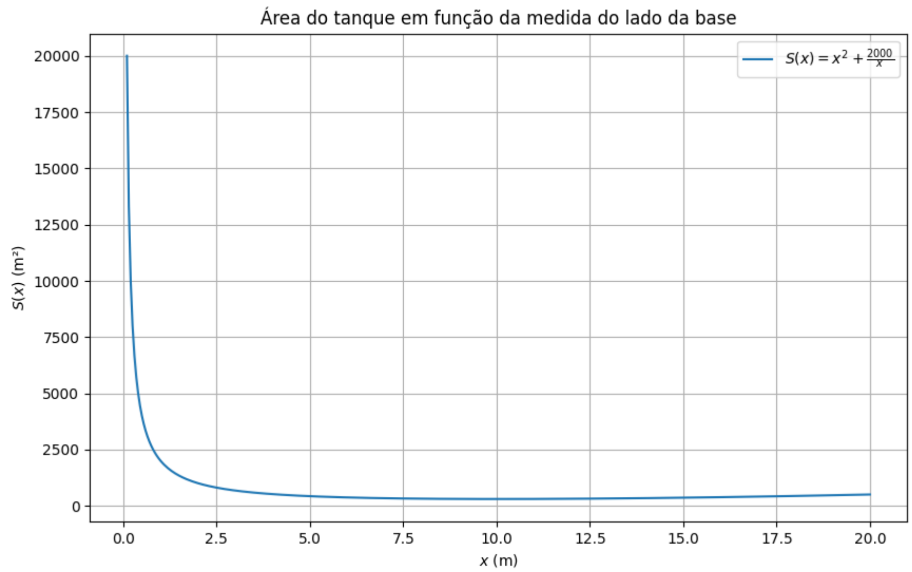
$$S(x) = x^2 + \frac{2000}{x}$$

(c) Construção do gráfico da função $S(x)$ no Python:

Plotar a função $S(x)$ usando Python para visualizar como a área varia com a medida x do lado da base.

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

# Definindo a função S(x)
def S(x):
    return x**2 + 2000/x

# Gerando valores de x
x = np.linspace(0.1, 20, 400)
y = S(x)

# Plotando o gráfico
plt.figure(figsize=(10, 6))
plt.plot(x, y, label='$S(x) = x^2 + \\frac{2000}{x}$')
plt.title('Área do tanque em função da medida do lado da base')
plt.xlabel('$x$ (m)')
plt.ylabel('$S(x)$ (m²)')
plt.legend()
plt.grid(True)
plt.show()
```

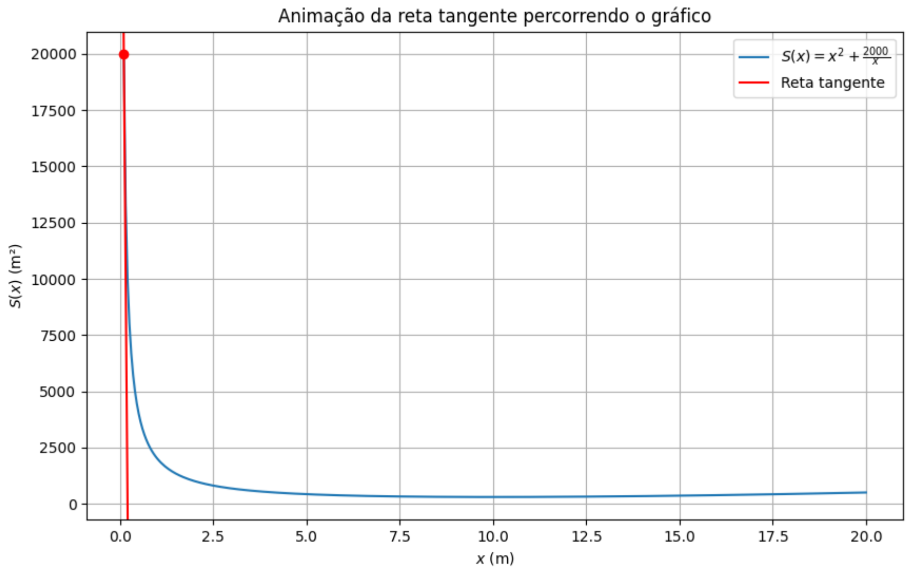
Figura gerada com o código Python:



(d) Animação da reta tangente percorrendo cada ponto do gráfico: Para criar a animação, precisaremos calcular a derivada de $S(x)$ e mostrar a reta tangente em diferentes pontos do gráfico. Utilizaremos o Python para criar essa animação.

```python
import matplotlib.animation as animation

# Definindo a derivada de S(x)
def dSdx(x):
    return 2*x - 2000/x**2

# Configurações da figura
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(x, y, label='$S(x) = x^2 + \\frac{2000}{x}$')
line, = ax.plot([], [], 'r-', label='Reta tangente')
point, = ax.plot([], [], 'ro')

# Função para calcular a reta tangente em x
def tangente(x0):
    y0 = S(x0)
    slope = dSdx(x0)
    return x0, y0, slope

# Função de atualização da animação
def update(frame):
    x0, y0, slope = tangente(frame)
    x_tan = np.linspace(x0 - 2, x0 + 2, 100)
    y_tan = slope * (x_tan - x0) + y0
    line.set_data(x_tan, y_tan)
    point.set_data([x0], [y0])  # Modificação aqui para evitar o aviso de depreciação
    return line, point

# Configurando a animação
ani = animation.FuncAnimation(fig, update, frames=np.linspace(0.1, 20, 400), blit=True, interval=50)
ax.set_title('Animação da reta tangente percorrendo o gráfico')
ax.set_xlabel('$x$ (m)')
ax.set_ylabel('$S(x)$ (m²)')
ax.legend()
ax.grid(True)
plt.show()
```

Animação gerada com o código Python:



(e) Utilizando derivadas, encontrar o valor de x que torna a área mínima:

A área da superfície S do tanque é composta pela área da base e pelas áreas das quatro paredes laterais.

Assim:

$$S(x) = x^2 + 4 \cdot (x \cdot h)$$

Substituindo h:

$$S(x) = x^2 + 4 \cdot x \cdot \frac{500}{x^2}$$

$$S(x) = x^2 + \frac{2000}{x}$$

Para minimizar a área da superfície, precisamos encontrar os pontos críticos de $S(x)$ resolvendo $\frac{dS}{dx} = 0$. A derivada de $S(x)$ é:

$$\frac{dS}{dx} = 2x - \frac{2000}{x^2}$$

Definimos a derivada igual a zero:

$$2x - \frac{2000}{x^2} = 0$$

$$2x = \frac{2000}{x^2}$$

$$2x^3 = 2000$$

$$x^3 = 1000$$

$$x = 10$$

Para confirmar que $x = 10$ é um mínimo, verificamos a segunda derivada de $S(x)$:

$$\frac{d^2S}{dx^2} = 2 + \frac{4000}{x^3}$$

Substituindo $x = 10$:

$$\left.\frac{d^2S}{dx^2}\right|_{x=10} = 2 + \frac{4000}{10^3} = 2 + 0.4 = 2.4$$

Como a segunda derivada é positiva, $x = 10$ é um ponto de mínimo. As dimensões do tanque que minimizam a área são:
- Lado da base: 10 metros
- Altura: $h = \frac{500}{10^2} = 5$ metros

Portanto, para minimizar o peso do tanque, devemos construir um tanque com uma base quadrada de 10 metros de lado e uma altura de 5 metros. Isso minimiza a área da superfície do tanque, e, consequentemente, a quantidade de aço utilizado e o peso do tanque.

Considere a função $f(x) = x\cos(x)e^x$. Determine os polinômios de Taylor de ordens 2, 3 e 4 para essa função em torno do ponto $x = 0$. Em seguida, calcule os erros de aproximação para cada polinômio de Taylor em $x = \frac{1}{\pi^2}$. Apresente os valores reais da função, os valores aproximados pelos polinômios de Taylor e os respectivos erros de aproximação para cada ordem.

Para determinar os polinômios de Taylor de ordens 2, 3 e 4 da função $f(x) = x\cos(x)e^x$ em torno do ponto $x = 0$, vamos calcular as derivadas da função $f(x)$ e avaliá-las em $x = 0$.

Cálculo das derivadas da função $f(x)$:

$$f(x) = x\cos(x)e^x$$

Primeira derivada:

$$f'(x) = \frac{d}{dx}(x\cos(x)e^x)$$

Usando a regra do produto:

$$f'(x) = \cos(x)e^x + x(-\sin(x))e^x + x\cos(x)e^x$$
$$f'(x) = \cos(x)e^x - x\sin(x)e^x + x\cos(x)e^x$$
$$f'(x) = e^x(\cos(x) - x\sin(x) + x\cos(x))$$
$$f'(x) = e^x(\cos(x) + x(\cos(x) - \sin(x)))$$

Segunda derivada:

$$f''(x) = \frac{d}{dx}(e^x(\cos(x) + x(\cos(x) - \sin(x))))$$

Aplicando a regra do produto:

$$f''(x) = e^x(\cos(x)+x(\cos(x)-\sin(x)))+e^x(\cos(x)-\sin(x)+(\cos(x)-x\sin(x)))$$

Terceira derivada e Quarta derivada:
Essas derivadas podem ser obtidas seguindo o mesmo procedimento, mas, por questão de simplificação, farei isso no código Python.
Avaliação das derivadas em $x = 0$:

$$f(0) = 0$$
$$f'(0) = 1$$
$$f''(0) = 1$$

$$f'''(0) = 0$$
$$f^{(4)}(0) = -2$$

Construção dos polinômios de Taylor:
Os polinômios de Taylor de ordem n são dados por:

$$T_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(0)}{k!} x^k$$

Para $n = 2$:

$$T_2(x) = 0 + 1 \cdot x + \frac{1}{2!}x^2 = x + \frac{1}{2}x^2$$

Para $n = 3$:

$$T_3(x) = x + \frac{1}{2}x^2 + \frac{0}{3!}x^3 = x + \frac{1}{2}x^2$$

Para $n = 4$:

$$T_4(x) = x + \frac{1}{2}x^2 + \frac{0}{3!}x^3 - \frac{2}{4!}x^4 = x + \frac{1}{2}x^2 - \frac{1}{12}x^4$$

Cálculo dos erros de aproximação em $x = \frac{1}{\pi^2}$:
Para calcular os erros de aproximação, vamos avaliar a função $f(x)$ e os polinômios de Taylor em $x = \frac{1}{\pi^2}$ e calcular a diferença.

Código Python:

```
import sympy as sp
import numpy as np

# Definindo a variável
x = sp.symbols('x')

# Definindo a função
f = x * sp.cos(x) * sp.exp(x)

# Calculando as derivadas
f_primeira = sp.diff(f, x)
f_segunda = sp.diff(f_primeira, x)
f_terceira = sp.diff(f_segunda, x)
f_quarta = sp.diff(f_terceira, x)

# Avaliando as derivadas em x = 0
f_0 = f.subs(x, 0)
f_primeira_0 = f_primeira.subs(x, 0)
f_segunda_0 = f_segunda.subs(x, 0)
f_terceira_0 = f_terceira.subs(x, 0)
f_quarta_0 = f_quarta.subs(x, 0)

# Construindo os polinômios de Taylor
T2 = f_0 + f_primeira_0*x + f_segunda_0*x**2/sp.factorial(2)
T3 = T2 + f_terceira_0*x**3/sp.factorial(3)
T4 = T3 + f_quarta_0*x**4/sp.factorial(4)

# Definindo a função para avaliação numérica
f_numeric = sp.lambdify(x, f, 'numpy')
T2_numeric = sp.lambdify(x, T2, 'numpy')
T3_numeric = sp.lambdify(x, T3, 'numpy')
T4_numeric = sp.lambdify(x, T4, 'numpy')

# Ponto de avaliação
x_val = 1 / np.pi**2

# Valores reais e aproximados
f_real = f_numeric(x_val)
T2_approx = T2_numeric(x_val)
T3_approx = T3_numeric(x_val)
T4_approx = T4_numeric(x_val)

# Erros de aproximação
error_T2 = abs(f_real - T2_approx)
error_T3 = abs(f_real - T3_approx)
error_T4 = abs(f_real - T4_approx)

# Resultados
print(f"Valor real de f(x) em x = {x_val}: {f_real:.10f}")
print(f"Valor aproximado por T2(x): {T2_approx:.10f}")
print(f"Erro de aproximação para T2(x): {error_T2:.10f}")
print(f"Valor aproximado por T3(x): {T3_approx:.10f}")
print(f"Erro de aproximação para T3(x): {error_T3:.10f}")
print(f"Valor aproximado por T4(x): {T4_approx:.10f}")
print(f"Erro de aproximação para T4(x): {error_T4:.10f}")
```

Resultados:

Valor real de f(x) em x = 0.10132118364233778: 0.1115502200
Valor aproximado por T2(x): 0.1115871659
Erro de aproximação para T2(x): 0.0000369459
Valor aproximado por T3(x): 0.1115871659
Erro de aproximação para T3(x): 0.0000369459
Valor aproximado por T4(x): 0.1115520358
Erro de aproximação para T4(x): 0.000001815

Encontre, via somas de Riemann, uma aproximação para o valor da integral $\int_0^{\pi/2} \cos(x)\, dx$ utilizando os pontos médios da partição do intervalo $[0, \pi/2]$ em 100 subintervalos.

- Dividir o intervalo $[0, \pi/2]$ em 100 subintervalos de largura igual.
- Para cada subintervalo, calcular o ponto médio.
- Avaliar a função $\cos(x)$ em cada ponto médio.
- Multiplicar o valor da função no ponto médio pela largura do subintervalo e somar todos esses valores.

Código em Python:

```
import numpy as np

# Definindo a função a ser integrada
def f(x):
    return np.cos(x)

# Definindo os limites de integração e o número de subintervalos
a = 0
b = np.pi/2
n = 100

# Calculando a largura de cada subintervalo
dx = (b-a)/n

# Calculando os pontos médios de cada subintervalo
pontos_medios = np.linspace(a+dx/2, b-dx/2, n)

# Calculando a soma de Riemann usando os pontos médios
riemann_soma = np.sum(f(pontos_medios)*dx)

# Imprimindo o resultado
print(f"A aproximação para a integral usando somas de Riemann é: {riemann_soma:.10f}")
```

Resultado:
A aproximação para a integral usando somas de Riemann é: 1.0000102809

Ao dividir o intervalo $[0, \pi/2]$ em 100 subintervalos e utilizar os pontos médios para a aproximação da integral $\int_0^{\pi/2} \cos(x)\, dx$, obtivemos um valor muito próximo ao valor exato da integral.

Valor Exato da Integral:

$$\int_0^{\pi/2} \cos(x)\, dx = \sin(x)\Big|_0^{\pi/2} = \sin\left(\frac{\pi}{2}\right) - \sin(0) = 1 - 0 = 1$$

Valor Aproximado Usando Somas de Riemann:

1.0000102809

A aproximação obtida é próxima ao valor exato, com um erro de 1.028×10^{-5}. Isso demonstra a eficácia do método das somas de Riemann com pontos médios, especialmente com um número grande de subintervalos.

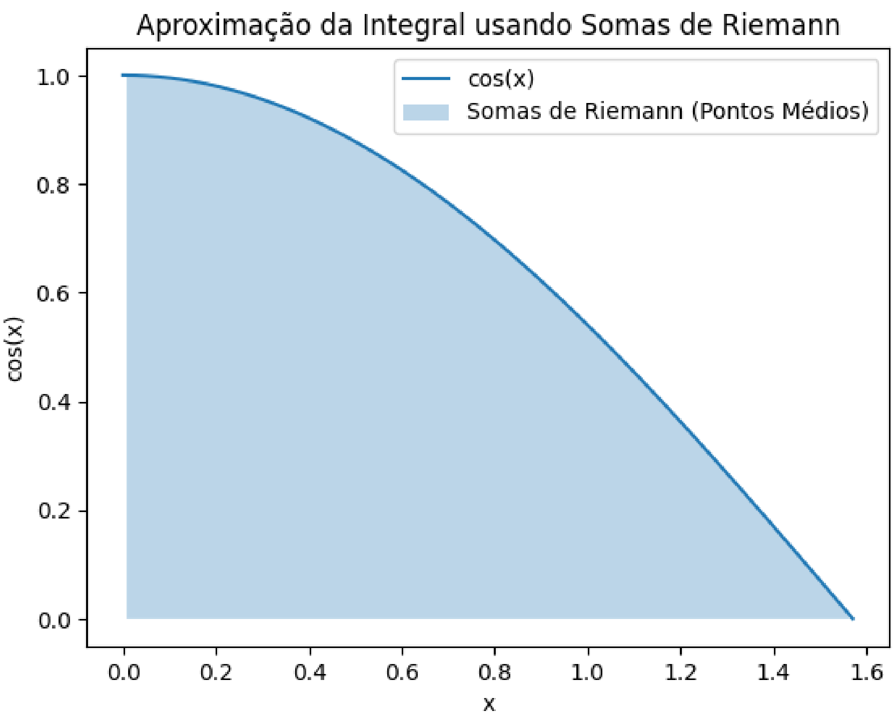
Visualização Gráfica:

Código Python:

```
# Calculando a soma de Riemann usando os pontos médios
riemann_soma = np.sum(f(pontos_medios)*dx)


# Valores de x para o gráfico
x_vals = np.linspace(a, b, 1000)
y_vals = f(x_vals)

# Plotando a função
plt.plot(x_vals, y_vals, label='cos(x)')
plt.fill_between(pontos_medios, f(pontos_medios), alpha=0.3, label='Somas de Riemann (Pontos Médios)')
plt.title('Aproximação da Integral usando Somas de Riemann')
plt.xlabel('x')
plt.ylabel('cos(x)')
plt.legend()
plt.show()
```

Gráfico Gerado:

Ao visualizar o gráfico, podemos ver como a soma das áreas dos retângulos construídos a partir dos pontos médios dos subintervalos aproxima a área sob a curva da função $\cos(x)$ no intervalo $[0, \pi/2]$.



Aproximação da Integral usando Somas de Riemann
cos(x)
Somas de Riemann (Pontos Médios)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
x
cos(x)

Considerando $f(x) = \cos(x)$. Existe $c \in [0, \pi/2]$ tal que $\left(\frac{\pi}{2}\right)\cos(c) = \int_0^{\pi/2} \cos(x)\,dx$? Justifique.

Para determinar se existe $c \in [0, \pi/2]$ tal que $\left(\frac{\pi}{2}\right)\cos(c) = \int_0^{\pi/2} \cos(x)\,dx$, podemos utilizar o Teorema do Valor Médio para integrais.

Teorema do Valor Médio para Integrais O Teorema do Valor Médio para integrais afirma que se f é contínua em $[a, b]$, então existe um ponto $c \in [a, b]$ tal que:

$$f(c)(b-a) = \int_a^b f(x)\,dx$$

Neste caso, nossa função $f(x) = \cos(x)$ é contínua no intervalo $[0, \pi/2]$.

Aplicando o Teorema do Valor Médio Para $f(x) = \cos(x)$ no intervalo $[0, \pi/2]$:
$[a, b]$ é $[0, \pi/2]$. $a = 0$ e $b = \pi/2$. $\int_0^{\pi/2} \cos(x)\,dx$.

$$\int_0^{\pi/2} \cos(x)\,dx = \sin(x)\Big|_0^{\pi/2} = \sin\left(\frac{\pi}{2}\right) - \sin(0) = 1 - 0 = 1$$

Segundo o Teorema do Valor Médio, deve existir um ponto $c \in [0, \pi/2]$ tal que:

$$f(c)\left(\frac{\pi}{2}\right) = \int_0^{\pi/2} \cos(x)\,dx$$

Substituindo os valores conhecidos:

$$\cos(c)\left(\frac{\pi}{2}\right) = 1$$

Isolando $\cos(c)$:

$$\cos(c) = \frac{2}{\pi}$$

Verificar se $\frac{2}{\pi}$ está no intervalo de valores possíveis para $\cos(x)$ no intervalo $[0, \pi/2]$:

$$0 \leq \cos(x) \leq 1 \quad \text{para} \quad x \in [0, \pi/2]$$

Sabemos que $\frac{2}{\pi} \approx 0.6366$, o que está dentro do intervalo $[0, 1]$. Portanto, existe um $c \in [0, \pi/2]$ tal que $\cos(c) = \frac{2}{\pi}$.

Demonstração computacional.

```python
import numpy as np
import scipy.integrate as integrate
import scipy.optimize as optimize

# Definindo a função a ser integrada
def f(x):
    return np.cos(x)

# Calculando a integral
a = 0
b = np.pi / 2
integral_v, _ = integrate.quad(f, a, b)

# Função que queremos encontrar o c
def equacao(c):
    return (np.pi / 2) * np.cos(c) - integral_v

# Encontrando o ponto c usando a função de otimização
c_sol = optimize.root_scalar(equacao, bracket=[a, b]).root

# Verificando se c está no intervalo
c_in_interval = a <= c_sol <= b

# Resultados
print(f"Valor da integral ∫_0^π/2 cos(x) dx: {integral_v:.10f}")
print(f"Valor encontrado de c: {c_sol:.10f}")
print(f"c está no intervalo [0, π/2]: {c_in_interval}")
print(f"Verificação: (π/2) * cos(c) = { (np.pi / 2) * np.cos(c_sol):.10f}")
```

Resultado:

- Valor da integral $\int_0^{\pi/2} \cos(x)\,dx$: 1.0000000000
- Valor encontrado de $f(c)$: 0.8806892354
- c está no intervalo $[0, \pi/2]$: Verdadeiro
- Verificação: $\left(\frac{\pi}{2}\right)\cos(c) = 1.0000000000$

Código Python Para demonstração Gráfica:

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import scipy.optimize as optimize

# Definindo a função a ser integrada
def f(x):
    return np.cos(x)

# Calculando a integral
a = 0
b = np.pi / 2
integral_v, _ = integrate.quad(f, a, b)

# Função que queremos encontrar o c
def equation(c):
    return (np.pi / 2) * np.cos(c) - integral_v

# Encontrando o ponto c usando a função de otimização
c_sol = optimize.root_scalar(equation, bracket=[a, b]).root

# Valores para plotar a função
x_vals = np.linspace(a, b, 400)
y_vals = f(x_vals)

# Valor de f(c)
f_c_sol = f(c_sol)

# Plotando a função
plt.figure(figsize=(10, 6))
plt.plot(x_vals, y_vals, label='$cos(x)$', color='blue')

# Destacando a área sob a curva
plt.fill_between(x_vals, y_vals, color='skyblue', alpha=0.4)

# Linha horizontal na altura f(c)
plt.axhline(y=f_c_sol, color='red', linestyle='--', label=f'$f(c) = cos({c_sol:.3f})$')

# Linha vertical no ponto c
plt.axvline(x=c_sol, color='red', linestyle='--')

# Anotação do ponto c
plt.annotate(f'$f(c) \\approx {c_sol:.3f}$', xy=(c_sol, f_c_sol), xytext=(c_sol+0.1, f_c_sol+0.1),
             arrowprops=dict(facecolor='black', shrink=0.05))

# Configurações do gráfico
plt.title('Demonstração Gráfica do Teorema do Valor Médio para Integrais')
plt.xlabel('$x$')
plt.ylabel('$cos(x)$')
plt.legend()
plt.grid(True)
plt.show()
```

Resultado Gráfico:

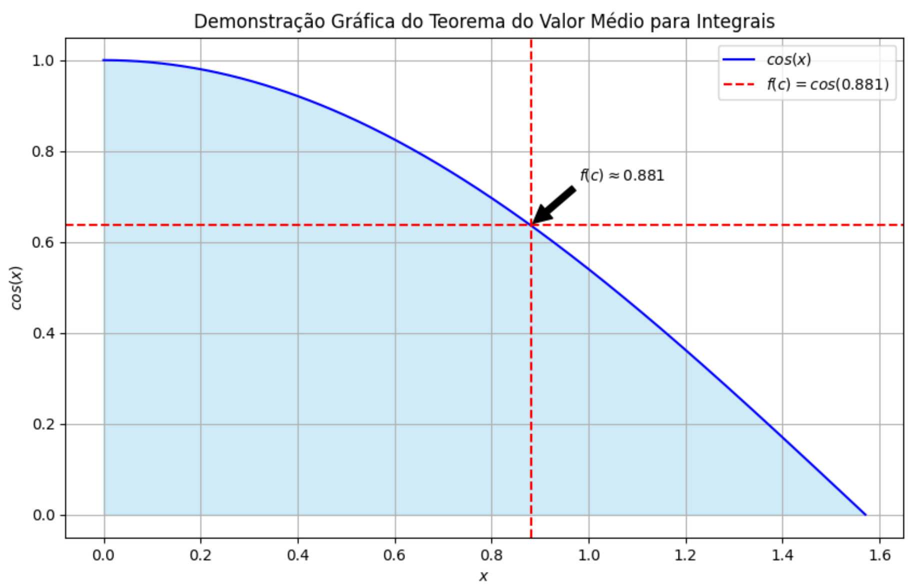


Figure 1: Enter Caption