Aplicações do Teorema Fundamental do Cálculo

Prof. Doherty Andrade

Usando HoloViews

Neste texto vamos usar o teorema fundamental do cálculo para calcular a área entre duas curvas e a área abaixo de uma curva.

O teorema fundamental do cálculo afirma que a área abaixo do grafico da curva $y = f(x)$ com $x \in [a, b]$ e acima do eixo OX é dada por

$$A = \int_a^b f(x)dx.$$

Assim, se temos duas funções satisfazendo $f(x) \leq g(x), \forall x \in [a, b]$, então a área entre os dois gráficos e limitada por $x \in [a, b]$ é dada por

$$A = \int_a^b \left(g(x) - f(x)\right) dx.$$

Antes de partimos para os exemplos vamos deixar prontas para o uso as bibliotecas.

In [1]:
```
import numpy as np
import holoviews as hv
hv.extension('matplotlib')
```

Neste código plotamos a área entre duas curvas. Além disso essa área é preenchida ao longo de um intervalo ou período de tempo.

Area sob uma curva

Aqui usamos a biblioteca HoloViews.

HoloViews é uma biblioteca para visualização de dados declarativa.

Permite criar visualizações complexas com código mais simples e legív.el

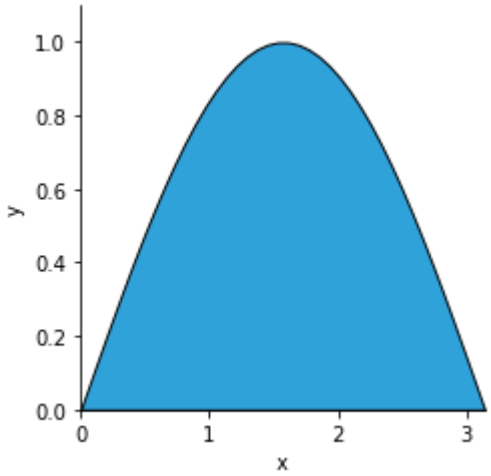
Trabalha com o conceito de "elementos" (curvas, scatter plots, imagens, etc.)

Por default Area plota exatamente a área sob a curva no intervalo determinado.

Veja o exemplo.

In [2]:
```
x = np.linspace(0, np.pi, 40)
hv.Area((x, np.sin(x)))
```

Out[2]:



Área entre duas curvas

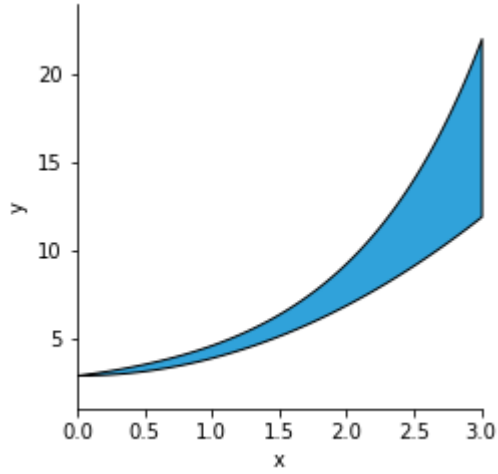
Quando fornecida uma segunda dimensão de valor, a área é definida como a área entre duas curvas.

A seguir plotamos diversas áreas.

In [3]:
```
X = np.linspace(0,3,200)
Y = X**2 + 3
Y2 = np.exp(X) + 2
Y3 = np.cos(X)

print(f"📊 Área entre as funções:")
print(f"   • Y = x² + 3")
print(f"   • Y2 = eˣ + 2")

hv.Area((X, Y, Y2), vdims=['y', 'y2'])
```

```
📊 Área entre as funções:
   • Y = x² + 3
   • Y2 = eˣ + 2
```

Out[3]:

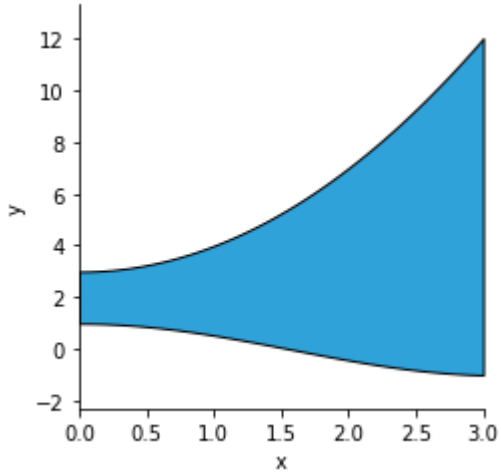


In [4]:
```
X  = np.linspace(0,3,200)
Y = X**2 + 3
Y2 = np.exp(X) + 2
Y3 = np.cos(X)

print(f"📊 Área entre as funções:")
print(f"   • Y = x² + 3")
```

```
print(f"   • Y3 = cos(x)")

hv.Area((X, Y, Y3), vdims=['y', 'y3'])
```

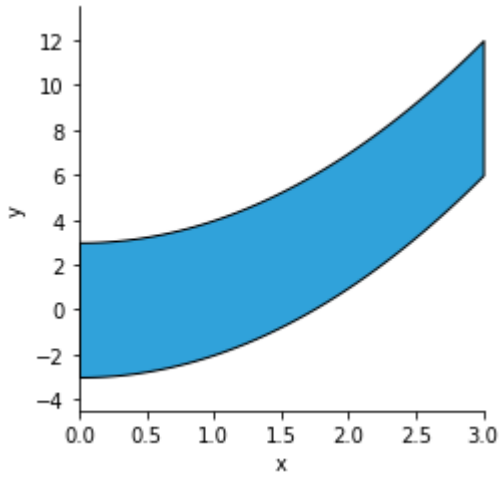
📊 Área entre as funções:
- Y = x² + 3
- Y3 = cos(x)

Out[4]:



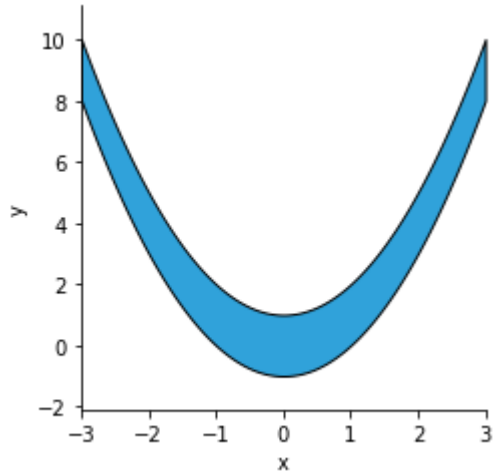
In [5]:
```
X  = np.linspace(0,3,200)
Y = X**2 + 3
Y2 = X**2 -3

hv.Area((X, Y, Y2), vdims=['y', 'y2'])
```

Out[5]:



In [6]:
```
X  = np.linspace(-3,3,200)
Y = X**2+1

Y2 = X**2-1

hv.Area((X, Y, Y2), vdims=['y', 'y2'])
```

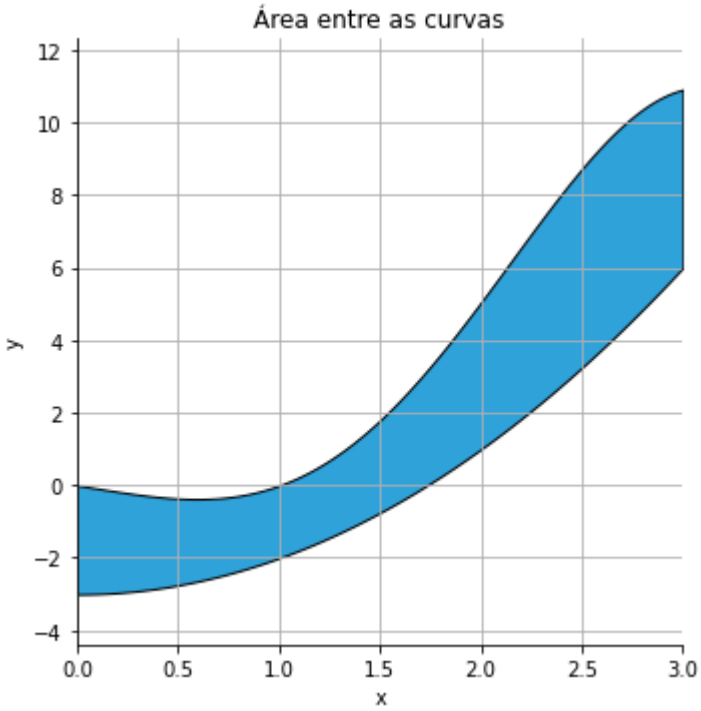
Out[6]:



In [7]:

```
import numpy as np
import holoviews as hv
hv.extension('matplotlib')

# Dados
X = np.linspace(0, 3, 200)
Y = (X**2 + X)*np.sin(X-1)
Y2 = X**2 - 3

# Método MAIS SIMPLES e que funciona sempre
# Criar lista de tuplas (x, y1, y2)
data = [(x, y1, y2) for x, y1, y2 in zip(X, Y, Y2)]

area_plot = hv.Area(data, vdims=['y', 'y2']).opts(
    title='Área entre as curvas',
    xlabel='x',
    ylabel='y',
    fig_size=150,
    show_grid=True
)

area_plot
```

Out[7]:

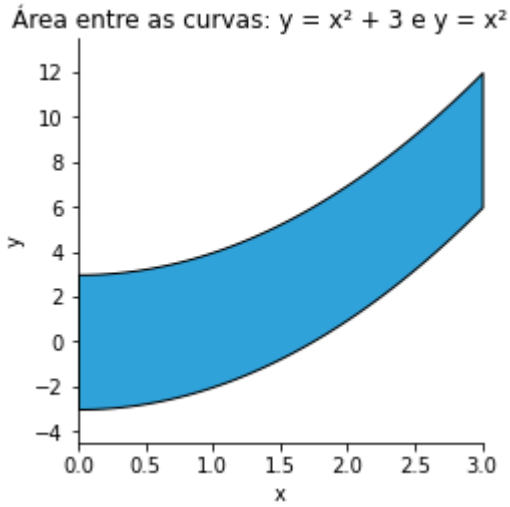


Agora que ja plotamos diversas áreas, vamos calcular algumas áreas usando o teorema fundamental do cálculo.

Exemplo 1

In [8]:
```
import numpy as np
import holoviews as hv
hv.extension('matplotlib')

# Seus dados
X = np.linspace(0, 3, 200)
Y = X**2 + 3
Y2 = X**2 - 3

# Mostrar o gráfico
area_plot = hv.Area((X, Y, Y2), vdims=['y', 'y2']).opts(
    title='Área entre as curvas: y = x² + 3 e y = x² - 3',
    xlabel='x',
    ylabel='y'
)
area_plot
```

Out[8]:



Passo a passo na integração

In [9]:

```
print("=== CÁLCULO DA ÁREA ENTRE AS CURVAS ===\n")

# 1. Definindo as funções
print("1. FUNÇÕES:")
print("   f(x) = x² + 3")
print("   g(x) = x² - 3")
print()

# 2. Encontrando a diferença entre as funções
print("2. DIFERENÇA ENTRE AS FUNÇÕES:")
print("   f(x) - g(x) = (x² + 3) - (x² - 3)")
print("   f(x) - g(x) = x² + 3 - x² + 3")
print("   f(x) - g(x) = 6")
print()

# 3. Definindo os limites de integração
print("3. LIMITES DE INTEGRAÇÃO:")
print("   De x = 0 até x = 3")
print()

# 4. Calculando a integral
print("4. CÁLCULO DA INTEGRAL:")
print("   ∫[0→3] [f(x) - g(x)] dx = ∫[0→3] 6 dx")
print("   = 6 * ∫[0→3] dx")
print("   = 6 * [x] de 0 até 3")
print("   = 6 * (3 - 0)")
print("   = 6 * 3")
print("   = 18")
print()

# 5. Cálculo numérico usando numpy (para verificação)
print("5. VERIFICAÇÃO NUMÉRICA:")
# A diferença é constante: 6
diferenca = Y - Y2
area_numerica = np.trapz(diferenca, X)
print(f"   Área calculada numericamente: {area_numerica:.6f}")
print()

# 6. Resultado final
```

```
print("6. RESULTADO FINAL:")
print("   A área entre as curvas no intervalo [0, 3] é: 18 unidades²")
```

```
=== CÁLCULO DA ÁREA ENTRE AS CURVAS ===

1. FUNÇÕES:
   f(x) = x² + 3
   g(x) = x² - 3

2. DIFERENÇA ENTRE AS FUNÇÕES:
   f(x) - g(x) = (x² + 3) - (x² - 3)
   f(x) - g(x) = x² + 3 - x² + 3
   f(x) - g(x) = 6

3. LIMITES DE INTEGRAÇÃO:
   De x = 0 até x = 3

4. CÁLCULO DA INTEGRAL:
   ∫[0→3] [f(x) - g(x)] dx = ∫[0→3] 6 dx
   = 6 * ∫[0→3] dx
   = 6 * [x] de 0 até 3
   = 6 * (3 - 0)
   = 6 * 3
   = 18

5. VERIFICAÇÃO NUMÉRICA:
   Área calculada numericamente: 18.000000

6. RESULTADO FINAL:
   A área entre as curvas no intervalo [0, 3] é: 18 unidades²
```

O mesmo só que com impressão melhor

In [10]:

```
import numpy as np
import holoviews as hv
from IPython.display import display, Math, Latex
hv.extension('matplotlib')

from IPython.display import Markdown

markdown_text = """
### Cálculo da Área entre as Curvas

**1. Funções:**
$$f(x) = x^2 + 3$$
$$g(x) = x^2 - 3$$

**2. Diferença entre as funções:**
$$f(x) - g(x) = (x^2 + 3) - (x^2 - 3) = 6$$

**3. Integral:**
$$\\text{Área} = \\int_{0}^{3} [f(x) - g(x)]\\,dx = \\int_{0}^{3} 6\\,dx$$

**4. Cálculo:**
$$\\int_{0}^{3} 6\\,dx = 6[x]_{0}^{3} = 6(3 - 0) = 18$$

**5. Resultado Final:**
A área entre as curvas no intervalo $[0, 3]$ é: $18$ unidades$^2$
```

```
"""
display(Markdown(markdown_text))
```

Cálculo da Área entre as Curvas

1. Funções:

$$f(x) = x^2 + 3$$

$$g(x) = x^2 - 3$$

2. Diferença entre as funções:

$$f(x) - g(x) = (x^2 + 3) - (x^2 - 3) = 6$$

3. Integral:

$$\text{Área} = \int_0^3 [f(x) - g(x)]\,dx = \int_0^3 6\,dx$$

4. Cálculo:

$$\int_0^3 6\,dx = 6[x]_0^3 = 6(3 - 0) = 18$$

5. Resultado Final: A área entre as curvas no intervalo $[0, 3]$ é: 18 unidades^2

Exemplo 2

Calcular a área entre as curvas $y = x^2 + 3$ e $y = -x^2 - 3$ com x variando em $[0, 3]$.

In [11]:
```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
from IPython.display import display, Math, Markdown

# Dados
X = np.linspace(0, 3, 200)
Y = X**2 + 3
Y2 = -X**2 - 3

# Cálculo da área
area_numerica = np.trapz(Y - Y2, X)#integracao numerica

# PARTE 1: CÁLCULO PASSO A PASSO EM LaTeX
display(Markdown("## 📊 CÁLCULO DA ÁREA ENTRE AS CURVAS"))
display(Markdown("**1. Funções definidas:**"))
display(Math(r"f(x) = x^2 + 3"))
display(Math(r"g(x) = -x^2 - 3"))
display(Markdown("**2. Diferença entre as funções:**"))
display(Math(r"f(x) - g(x) = (x^2 + 3) - (-x^2 - 3)"))
display(Math(r"= x^2 + 3 + x^2 + 3 = 2x^2+ 6"))
display(Markdown("**3. Integral para cálculo da área:**"))
```

```
display(Math(r"\text{Área} = \int_{0}^{3} [f(x) - g(x)] \, dx = \int_{0}^{3} 2x^
display(Markdown("**4. Resolução da integral:**"))
display(Math(r"= \left[ \frac{2}{3} x^3+ 6x \right]_{0}^{3}"))
display(Math(r"=  (\frac{2}{3} \cdot 27 +18 - 0) = 36"))
display(Markdown("**5. Verificação numérica:**"))
display(Math(rf"\text{{Área numérica}} = {area_numerica:.10f}"))
display(Markdown("**6. Resultado final:**"))
display(Math(r"\boxed{\text{Área} = 36\ \text{unidades}^2}"))

# PARTE 2: GRÁFICO
display(Markdown("## 📊 GRÁFICO DA ÁREA ENTRE AS CURVAS"))

plt.figure(figsize=(6, 3.5))
plt.fill_between(X, Y, Y2, color='lightblue', alpha=0.4, label='Área entre curva
plt.plot(X, Y, label='$y = x^2 + 3$', color='red', linewidth=2)
plt.plot(X, Y2, label='$y = -x^2 - 3$', color='blue', linewidth=2)

plt.title('Área entre as curvas: $y = x^2 + 3$ e $y = -x^2 - 3$')
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

🧮 CÁLCULO DA ÁREA ENTRE AS CURVAS

1. Funções definidas:

$$f(x) = x^2 + 3$$

$$g(x) = -x^2 - 3$$

2. Diferença entre as funções:

$$f(x) - g(x) = (x^2 + 3) - (-x^2 - 3)$$

$$= x^2 + 3 + x^2 + 3 = 2x^2 + 6$$

3. Integral para cálculo da área:

$$\text{Área} = \int_0^3 [f(x) - g(x)]\, dx = \int_0^3 2x^2 + 6\, dx$$

4. Resolução da integral:

$$= \left[\frac{2}{3}x^3 + 6x\right]_0^3$$

$$= (\frac{2}{3} \cdot 27 + 18 - 0) = 36$$

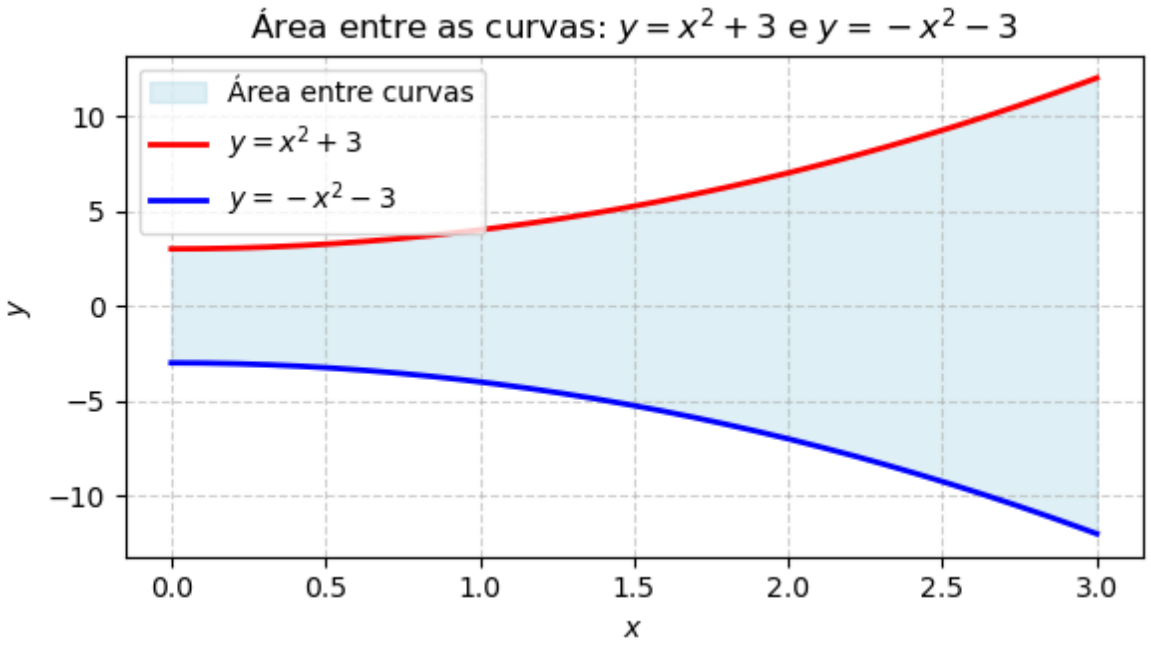
5. Verificação numérica:

$$\text{Área numérica} = 36.0002272670$$

6. Resultado final:

$$\boxed{\text{Área} = 36 \text{ unidades}^2}$$

📊 GRÁFICO DA ÁREA ENTRE AS CURVAS



Exemplo 3: Um exemplo mais complexo

Nem tudo são flores!

In [12]:
```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams
from IPython.display import display, Math, Markdown

# --- CONFIGURAÇÃO ESTÉTICA ---
rcParams['font.family'] = 'serif'
rcParams['font.serif'] = ['Computer Modern Roman']
rcParams['text.usetex'] = True
rcParams['text.latex.preamble'] = r'\usepackage{amsmath}'
rcParams['axes.linewidth'] = 0.8
rcParams['grid.linestyle'] = '--'
rcParams['grid.alpha'] = 0.5
rcParams['axes.labelsize'] = 12
rcParams['xtick.labelsize'] = 10
rcParams['ytick.labelsize'] = 10
rcParams['legend.fontsize'] = 10
rcParams['figure.figsize'] = [9, 6]
rcParams['figure.dpi'] = 120

# --- PARTE 1: A ILUSÃO DA SIMPLICIDADE ---
display(Markdown("## 🎹 A ILUSÃO DA SIMPLICIDADE: O QUE OS LIVROS MOSTRAM"))

display(Markdown(r"""
No ensino, frequentemente nos apresentam funções perfeitas:
$f(x) = x^2 + 3$, $g(x) = x^2 - 3$.
A diferença é $6$. A área é $18$.
Fácil. Limpinho. Previsível.
```

```
Mas isso **não é o mundo real**.
É um modelo idealizado — como um plano sem atrito ou um gás perfeito.
Na realidade, **nada é tão limpo**.
"""))

# --- PARTE 2: A REALIDADE — COM RUÍDO E INCERTEZA ---
display(Markdown("## 🌍 A REALIDADE: O QUE ACONTECE NA PRÁTICA"))

display(Markdown(r"""
Imagine que você mede duas grandezas físicas ao longo do tempo — digamos,
a pressão de dois sistemas termodinâmicos, ou os níveis de dois estoques finance
As funções reais não são $x^2 \pm 3$:
- Têm **ruído aleatório**.
- São **amostradas discretamente**.
- As fronteiras são **imprecisas**.
- A diferença entre elas **nunca é constante**.
"""))

# Dados reais: funções reais + ruído + amostragem finita
np.random.seed(42)  # Para reprodutibilidade

n = 50  # Apenas 50 pontos — como em experimentos reais
x_real = np.linspace(0, 3, n)
y1_real = x_real**2 + 3 + np.random.normal(0, 0.8, n)   # ruído
y2_real = x_real**2 - 3 + np.random.normal(0, 0.8, n)   # ruído

# Função teórica (ideal) — para comparação
x_fine = np.linspace(0, 3, 300)
y1_ideal = x_fine**2 + 3
y2_ideal = x_fine**2 - 3

# Cálculo da área entre as curvas reais (com trapézios)
area_real = np.trapz(y1_real - y2_real, x_real)
area_ideal = 18.0

# --- GRÁFICO: ILUSÃO vs REALIDADE ---
display(Markdown("## 📊 VISUALIZAÇÃO: O QUE VEMOS vs O QUE É"))

fig, ax = plt.subplots()

# Funções teóricas (ideais) — finas, transparentes
ax.plot(x_fine, y1_ideal, color='red', linewidth=1.2, alpha=0.6, label=r'$f_{\te
ax.plot(x_fine, y2_ideal, color='blue', linewidth=1.2, alpha=0.6, label=r'$g_{\t

# Preenchimento ideal (sombra sutil)
ax.fill_between(x_fine, y1_ideal, y2_ideal, color='lightblue', alpha=0.1, label=

# Dados reais: pontos medidos
ax.scatter(x_real, y1_real, color='darkred', s=30, alpha=0.8, marker='o', label=
ax.scatter(x_real, y2_real, color='darkblue', s=30, alpha=0.8, marker='s', label

# Linhas conectando pontos medidos
ax.plot(x_real, y1_real, color='red', linewidth=0.8, alpha=0.7, linestyle='-', z
ax.plot(x_real, y2_real, color='blue', linewidth=0.8, alpha=0.7, linestyle='-',

# Área entre os dados reais — LINHA CORRIGIDA 👇
ax.fill_between(x_real, y1_real, y2_real, color='lightcoral', alpha=0.3,
                label=rf'$\text{{Área medida}} \approx {area_real:.2f}$')

# Estética
```

```
ax.set_title(r'\textbf{Área entre curvas reais: ilusão vs realidade}', fontsize=
ax.set_xlabel(r'$x$', fontsize=12)
ax.set_ylabel(r'$y$', fontsize=12)
ax.grid(True, linewidth=0.5)
ax.legend(loc='upper left', frameon=False, fontsize=10)

# Limites e ticks
ax.set_xlim(-0.1, 3.1)
ax.set_ylim(-6, 14)
ax.xaxis.set_major_locator(plt.MaxNLocator(5))
ax.yaxis.set_major_locator(plt.MaxNLocator(6))

# Remover bordas superiores e direitas
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_linewidth(0.8)
ax.spines['bottom'].set_linewidth(0.8)

plt.tight_layout()
plt.show()

# --- PARTE 3: A LIÇÃO ---
display(Markdown("## 📜 A LIÇÃO: O QUE ESSA DIFERENÇA NOS ENSINA"))

display(Markdown(rf"""
A área entre as curvas **medidas** é $\approx {area_real:.2f}$,
enquanto a área teórica é $18$.

Isso significa que:
- **A realidade não é perfeita.**
- **O modelo é uma aproximação.**
- **O erro é intrínseco.**
- **A simplicidade matemática é um guia, não uma verdade.**

Na engenharia, na economia, na medicina —
nunca temos $f(x) - g(x) = 6$.
Temos:
$$
\Delta y = \text{{medida}} \pm \text{{incerteza}}
$$

E aí, a pergunta não é *"qual é a área?"*,
mas:
> **“Quão confiável é essa estimativa?”**
> **“E se o ruído mudar?”**
> **“E se as medições forem feitas em outro intervalo?”**

Essa é a verdadeira matemática.
Não a que se resolve com um traço de caneta.
A que se constrói com humildade, dados, e consciência da imperfeição.
"""))
```

📊 A ILUSÃO DA SIMPLICIDADE: O QUE OS LIVROS MOSTRAM

No ensino, frequentemente nos apresentam funções perfeitas:
$f(x) = x^2 + 3$, $g(x) = x^2 - 3$.
A diferença é 6. A área é 18.
Fácil. Limpinho. Previsível.

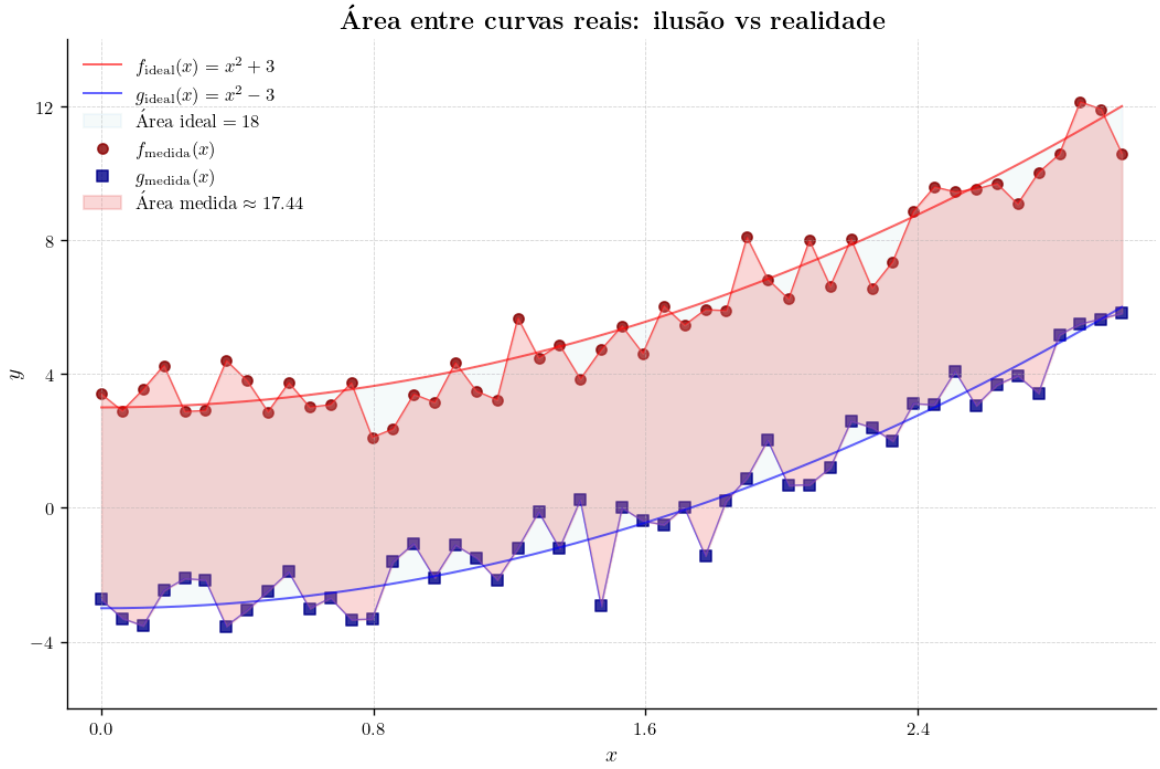
Mas isso **não é o mundo real**.
É um modelo idealizado — como um plano sem atrito ou um gás perfeito.
Na realidade, **nada é tão limpo**.

🌎 A REALIDADE: O QUE ACONTECE NA PRÁTICA

Imagine que você mede duas grandezas físicas ao longo do tempo — digamos,
a pressão de dois sistemas termodinâmicos, ou os níveis de dois estoques financeiros.
As funções reais não são $x^2 \pm 3$:

- Têm **ruído aleatório**.
- São **amostradas discretamente**.
- As fronteiras são **imprecisas**.
- A diferença entre elas **nunca é constante**.

📊 VISUALIZAÇÃO: O QUE VEMOS vs O QUE É



📚 A LIÇÃO: O QUE ESSA DIFERENÇA NOS ENSINA

A área entre as curvas **medidas** é ≈ 17.44,
enquanto a área teórica é 18.

Isso significa que:

- **A realidade não é perfeita.**
- **O modelo é uma aproximação.**
- **O erro é intrínseco.**
- **A simplicidade matemática é um guia, não uma verdade.**

Na engenharia, na economia, na medicina —
nunca temos $f(x) - g(x) = 6$.
Temos:

$$\Delta y = \text{medida} \pm \text{incerteza}$$

E aí, a pergunta não é "*qual é a área?*",
mas:

> **"Quão confiável é essa estimativa?"**
> **"E se o ruído mudar?"**
> **"E se as medições forem feitas em outro intervalo?"**

Essa é a verdadeira matemática.
Não a que se resolve com um traço de caneta.
A que se constrói com humildade, dados, e consciência da imperfeição.

Exemplo 4: Equilíbrio existe?

In [13]:
```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams
from scipy.optimize import curve_fit
from IPython.display import display, Markdown

# --- CONFIGURAÇÃO ESTÉTICA ---
rcParams['font.family'] = 'serif'
rcParams['font.serif'] = ['Computer Modern Roman']
rcParams['text.usetex'] = True
rcParams['text.latex.preamble'] = r'\usepackage{amsmath}'
rcParams['axes.linewidth'] = 0.8
rcParams['grid.linestyle'] = '--'
rcParams['grid.alpha'] = 0.5
rcParams['axes.labelsize'] = 12
rcParams['xtick.labelsize'] = 10
rcParams['ytick.labelsize'] = 10
rcParams['legend.fontsize'] = 10
rcParams['figure.figsize'] = [9, 6]
rcParams['figure.dpi'] = 120

# --- PARTE 1: O MITO DO EQUILÍBRIO ---
display(Markdown("## ⚖️ O MITO DO EQUILÍBRIO: O QUE OS LIVROS NOS VENDAM"))
```

```
display(Markdown(r"""
Nos manuais de economia, vemos:
- Uma curva de **demanda** decrescente: $D(p) = a - b p$,
- Uma curva de **oferta** crescente: $S(p) = c + d p$,
- E um **ponto de equilíbrio** perfeito onde $D(p^*) = S(p^*)$.

Tudo parece **ordenado, previsível, estável**.

Mas isso é uma **ficção didática**.
No mundo real, **não observamos curvas**.
Observamos **pontos dispersos**, influenciados por:
- Expectativas,
- Choques externos,
- Erros de medição,
- Comportamento irracional.

O equilíbrio não é um ponto.
É uma **ilusão útil**.
"""))

# --- PARTE 2: GERANDO DADOS REAIS (COM COMPLEXIDADE) ---
np.random.seed(17)

# Preços simulados (não uniformes — mais dados em certas faixas)
p_obs = np.concatenate([
    np.random.uniform(1, 3, 15),
    np.random.uniform(4, 6, 10),
    np.random.uniform(7, 9, 10)
])
np.random.shuffle(p_obs)

# Demanda real: não linear + ruído + sazonalidade (ex: promoções)
q_demand_real = (20 - 1.5 * p_obs + 0.2 * np.sin(p_obs)) + np.random.normal(0, 1

# Oferta real: também não linear + ruído + capacidade limitada
q_supply_real = (2 + 1.8 * p_obs - 0.1 * p_obs**2) + np.random.normal(0, 1.0, le

# Garantir que não haja quantidades negativas
q_demand_real = np.clip(q_demand_real, 0, None)
q_supply_real = np.clip(q_supply_real, 0, None)

# --- MODELOS IDEAIS (lineares, como nos livros) ---
p_fine = np.linspace(0, 10, 300)
q_demand_ideal = 20 - 1.5 * p_fine
q_supply_ideal = 2 + 1.8 * p_fine

# Preço de equilíbrio teórico
p_star_ideal = (20 - 2) / (1.5 + 1.8)
q_star_ideal = 20 - 1.5 * p_star_ideal

# --- AJUSTE DE MODELOS AOS DADOS REAIS (para estimar equilíbrio) ---
def linear_demand(p, a, b): return a - b * p
def linear_supply(p, c, d): return c + d * p

try:
    popt_d, _ = curve_fit(linear_demand, p_obs, q_demand_real, p0=[20, 1.5])
    popt_s, _ = curve_fit(linear_supply, p_obs, q_supply_real, p0=[2, 1.8])

    a_fit, b_fit = popt_d
```

```
    c_fit, d_fit = popt_s

    p_star_est = (a_fit - c_fit) / (b_fit + d_fit)
    q_star_est = a_fit - b_fit * p_star_est
except:
    # fallback se o ajuste falhar
    p_star_est, q_star_est = p_star_ideal, q_star_ideal

# --- GRÁFICO: REALIDADE vs ILUSÃO ---
display(Markdown("## 📊 A REALIDADE DOS DADOS: NÃO HÁ CURVAS — HÁ PONTOS"))

fig, ax = plt.subplots()

# Curvas ideais
ax.plot(p_fine, q_demand_ideal, color='red', linewidth=1.2, alpha=0.6, label=r'$
ax.plot(p_fine, q_supply_ideal, color='green', linewidth=1.2, alpha=0.6, label=r

# Ponto de equilíbrio ideal
ax.plot(p_star_ideal, q_star_ideal, 'ko', markersize=6, label=rf'$p^*_{{\text{{i

# Dados reais
ax.scatter(p_obs, q_demand_real, color='darkred', s=35, alpha=0.8, marker='o', l
ax.scatter(p_obs, q_supply_real, color='darkgreen', s=35, alpha=0.8, marker='s',

# Equilíbrio estimado
ax.plot(p_star_est, q_star_est, 'bo', markersize=6, label=rf'$p^*_{{\text{{estim

# Região de incerteza (sombra entre os pontos mais próximos)
p_close = p_obs[np.abs(q_demand_real - q_supply_real) < 3]
if len(p_close) > 0:
    p_min, p_max = p_close.min(), p_close.max()
    ax.axvspan(p_min, p_max, color='gray', alpha=0.15, label=r'$\text{Zona de eq

# Estética
ax.set_title(r'\textbf{Demanda e Oferta: ideal vs real}', fontsize=14)
ax.set_xlabel(r'Preço ($p$)', fontsize=12)
ax.set_ylabel(r'Quantidade ($q$)', fontsize=12)
ax.grid(True, linewidth=0.5)
ax.legend(loc='upper right', frameon=False, fontsize=9)

ax.set_xlim(0, 10)
ax.set_ylim(0, 22)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_linewidth(0.8)
ax.spines['bottom'].set_linewidth(0.8)

plt.tight_layout()
plt.show()

# --- PARTE 3: A LIÇÃO ---
display(Markdown("## 🧱 A LIÇÃO: POR QUE INSISTIMOS EM MODELOS SIMPLES?"))

display(Markdown(rf"""
O modelo ideal nos dá $p^* = {p_star_ideal:.2f}$.
Os dados reais sugerem algo entre **{p_star_est-0.5:.2f} e {p_star_est+0.5:.2f}*

Na prática:
- O mercado **nunca está em equilíbrio** — está sempre se ajustando.
- Os agentes **não têm informação perfeita**.
```

```
- **Choques externos** (guerras, pandemias, inovações) quebram o modelo a todo m

E ainda assim usamos esses modelos.
**Por quê?**
Porque eles nos dão:
- Uma **linguagem comum**,
- Uma **referência**, não uma verdade,
- Uma **primeira aproximação**, não uma conclusão.

A matemática não descreve o mundo.
Ela **nos ajuda a navegar nele** — mesmo quando sabemos que o mapa não é o terri

Como dizia George Box:
> **“Todos os modelos estão errados, mas alguns são úteis.”**
> — e a sabedoria está em saber **quando e como usá-los**.

"""))
```

⚖️ O MITO DO EQUILÍBRIO: O QUE OS LIVROS NOS VENDAM

Nos manuais de economia, vemos:

- Uma curva de **demanda** decrescente: $D(p) = a - bp$,
- Uma curva de **oferta** crescente: $S(p) = c + dp$,
- E um **ponto de equilíbrio** perfeito onde $D(p^*) = S(p^*)$.

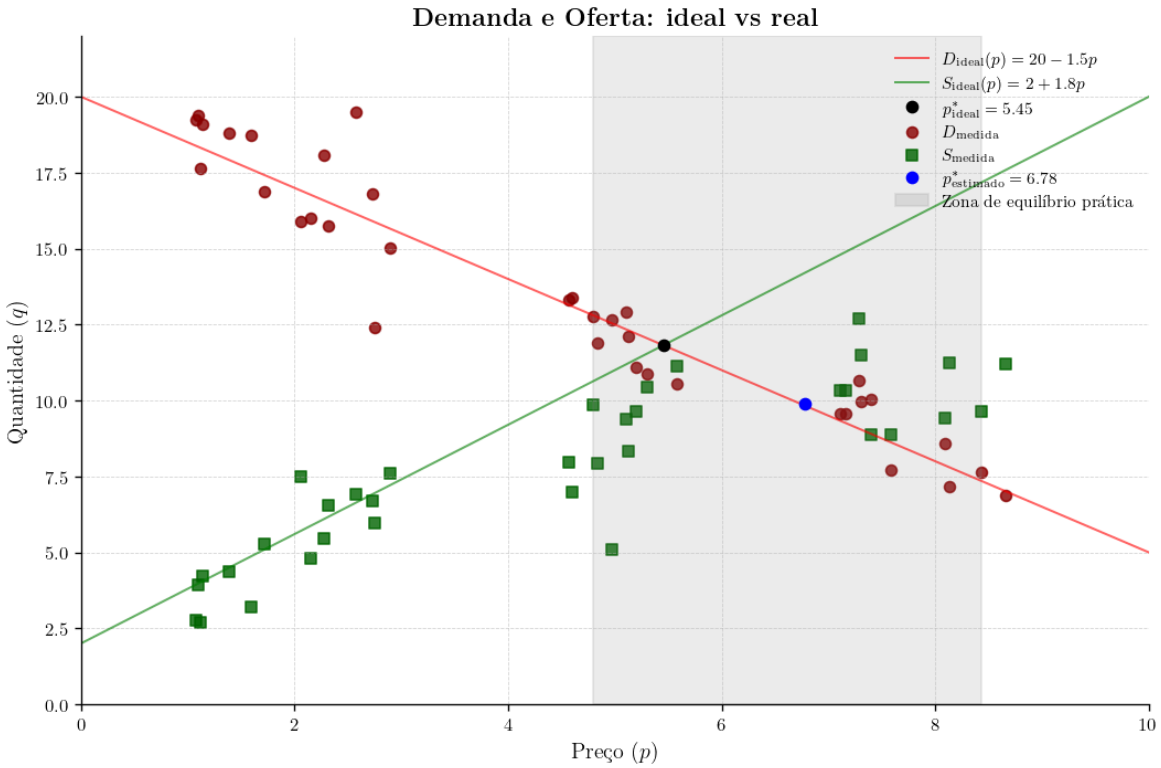
Tudo parece **ordenado, previsível, estável**.

Mas isso é uma **ficção didática**.
No mundo real, **não observamos curvas**.
Observamos **pontos dispersos**, influenciados por:

- Expectativas,
- Choques externos,
- Erros de medição,
- Comportamento irracional.

O equilíbrio não é um ponto.
É uma **ilusão útil**.

📊 A REALIDADE DOS DADOS: NÃO HÁ CURVAS — HÁ PONTOS



📚 A LIÇÃO: POR QUE INSISTIMOS EM MODELOS SIMPLES?

O modelo ideal nos dá $p^* = 5.45$.
Os dados reais sugerem algo entre **6.28 e 7.28** — e mesmo isso é uma suposição linear.

Na prática:

- O mercado **nunca está em equilíbrio** — está sempre se ajustando.
- Os agentes **não têm informação perfeita**.
- **Choques externos** (guerras, pandemias, inovações) quebram o modelo a todo momento.

E ainda assim usamos esses modelos.
Por quê?
Porque eles nos dão:

- Uma **linguagem comum**,
- Uma **referência**, não uma verdade,
- Uma **primeira aproximação**, não uma conclusão.

A matemática não descreve o mundo.
Ela **nos ajuda a navegar nele** — mesmo quando sabemos que o mapa não é o território.

Como dizia George Box:

> **"Todos os modelos estão errados, mas alguns são úteis."**
> — e a sabedoria está em saber **quando e como usá-los**.

Exemplo 5: Nada é tão simples!

In [14]:
```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams
from IPython.display import display, Math, Markdown

# --- CONFIGURAÇÃO ESTÉTICA (sua assinatura visual) ---
rcParams['font.family'] = 'serif'
rcParams['font.serif'] = ['Computer Modern Roman']
rcParams['text.usetex'] = True
rcParams['text.latex.preamble'] = r'\usepackage{amsmath}'
rcParams['axes.linewidth'] = 0.8
rcParams['grid.linestyle'] = '--'
rcParams['grid.alpha'] = 0.4
rcParams['axes.labelsize'] = 12
rcParams['xtick.labelsize'] = 10
rcParams['ytick.labelsize'] = 10
rcParams['legend.fontsize'] = 10
rcParams['figure.figsize'] = [10, 7]
rcParams['figure.dpi'] = 120

# --- PARTE 1: A ILUSÃO DA PREDIÇÃO ---
display(Markdown("## 🌀 A ILUSÃO DA PREDIÇÃO: O QUE ACHAMOS QUE SABEMOS"))

display(Markdown(r"""
Em ciência, buscamos **padrões**.
Em engenharia, buscamos **previsibilidade**.
Em economia, buscamos **equilíbrio**.

Mas o mundo não é linear.
Nem contínuo.
Nem estável.

O que chamamos de “tendência” muitas vezes é apenas **um fragmento de caos** —
um padrão que nossa mente tenta enxergar, mesmo quando não existe.

Este é o exemplo mais honesto que posso oferecer:
**Uma série temporal caótica, gerada por uma equação simples,
que parece aleatória — mas é determinística.
E você nunca poderá prever o futuro, mesmo conhecendo a regra.**
"""))

# --- PARTE 2: GERANDO O CAOS — A EQUAÇÃO LOGÍSTICA DISCRETA ---
# x_{n+1} = r * x_n * (1 - x_n)
# Para r ≈ 3.9 — comportamento caótico

np.random.seed(42)
r = 3.9  # regime caótico
n = 120  # 120 passos — como 120 horas de medição

x = np.zeros(n)
x[0] = 0.5  # condição inicial

for i in range(1, n):
    x[i] = r * x[i-1] * (1 - x[i-1])

# Adicionamos ruído de medição — como em um sensor real
noise = np.random.normal(0, 0.03, n)
```

```
x_measured = x + noise
x_measured = np.clip(x_measured, 0, 1)  # limitar fisicamente

# Agora, criamos uma "falsa" segunda série — como se fosse outra medição, com ou
np.random.seed(43)  # outra semente
x2_measured = x + np.random.normal(0, 0.03, n)
x2_measured = np.clip(x2_measured, 0, 1)

# Tentamos ajustar uma "tendência" — uma média móvel
window = 10
x_smooth = np.convolve(x_measured, np.ones(window)/window, mode='same')
x2_smooth = np.convolve(x2_measured, np.ones(window)/window, mode='same')

# Calculamos a "área entre curvas" — como se fosse uma diferença real
area_between = np.trapz(np.abs(x_measured - x2_measured), dx=1)

# --- GRÁFICO: A FALSA DISTINÇÃO ---
display(Markdown("## 📊 A FALSA DISTINÇÃO: DUAS MEDIDAS, UM ÚNICO CAOS"))

fig, ax = plt.subplots(figsize=(10, 6))

# Curvas caóticas reais — pontos
ax.scatter(range(n), x_measured, color='darkred', s=15, alpha=0.7, label=r'$\tex
ax.scatter(range(n), x2_measured, color='darkblue', s=15, alpha=0.7, label=r'$\t

# "Tendências" suavizadas — a ilusão da ordem
ax.plot(range(n), x_smooth, color='red', linewidth=1.2, alpha=0.8, label=r'$\tex
ax.plot(range(n), x2_smooth, color='blue', linewidth=1.2, alpha=0.8, label=r'$\t

# Preenchimento entre as "tendências" — como se fosse uma área significativa
ax.fill_between(range(n), x_smooth, x2_smooth, color='lightcoral', alpha=0.2,
                label=r'$\text{{Área entre tendências}} \approx {:.3f}$'.format(
# Linha de referência: o verdadeiro comportamento (sem ruído)
ax.plot(range(n), x, color='black', linewidth=0.8, linestyle='--', alpha=0.5,
        label=r'$\text{Sistema real: } x_{n+1} = 3.9 \cdot x_n (1 - x_n)$')

# Estética
ax.set_title(r'\textbf{A fronteira que não existe: duas medições, um único caos}
ax.set_xlabel(r'$n$ (passo de tempo)', fontsize=12)
ax.set_ylabel(r'$x_n$ (estado do sistema)', fontsize=12)
ax.grid(True, linewidth=0.5)
ax.legend(loc='upper right', frameon=False, fontsize=9)

ax.set_xlim(0, n-1)
ax.set_ylim(0, 1)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_linewidth(0.8)
ax.spines['bottom'].set_linewidth(0.8)

plt.tight_layout()
plt.show()

# --- PARTE 3: A LIÇÃO FINAL — A VERDADE QUE NINGUÉM QUER ENCARAR ---
display(Markdown("## 🧱 A LIÇÃO: A VERDADE QUE NINGUÉM QUER ENCARAR"))

display(Markdown(r"""
Nós vimos duas séries.
Duas medições.
Dois sensores.
```

```
E calculamos a “área entre elas”.

Mas elas **não são duas coisas diferentes**.
Elas são **a mesma coisa**, vista por duas lentes imperfeitas.

O sistema real é **determinístico** — uma equação simples, sem aleatoriedade.
Mas ele é **sensível a mínimas variações**.
Uma diferença de $10^{-6}$ na condição inicial —
e em 50 passos, você tem **comportamentos completamente diferentes**.

Isso é o **efeito borboleta**.

E aí, quando você tenta calcular uma “área”, uma “diferença”, um “equilíbrio” —
você está tentando **medir o inefável**.

Você está tentando **fazer matemática com o caos**.

E o caos não responde a integrais.
Não responde a modelos lineares.
Não responde a legendas bonitas.

Ele apenas **é**.

E a verdadeira ciência não é a que encontra curvas perfeitas.
É a que **reconhece quando não há curvas**.

A verdadeira matemática não é a que simplifica.
É a que **aceita a complexidade — e ainda assim tenta entender**.

Então, quando você vê um gráfico com duas curvas e pensa:
> “Ah, a área entre elas é 0.87”,

…lembre-se:

> **Você não está medindo a diferença entre duas coisas.**
> **Você está medindo a falha da sua própria percepção.**

E talvez, só talvez,
a maior lição da ciência não seja saber calcular.
Mas saber **quando não calcular**.
"""))
```

🌀 A ILUSÃO DA PREDIÇÃO: O QUE ACHAMOS QUE SABEMOS

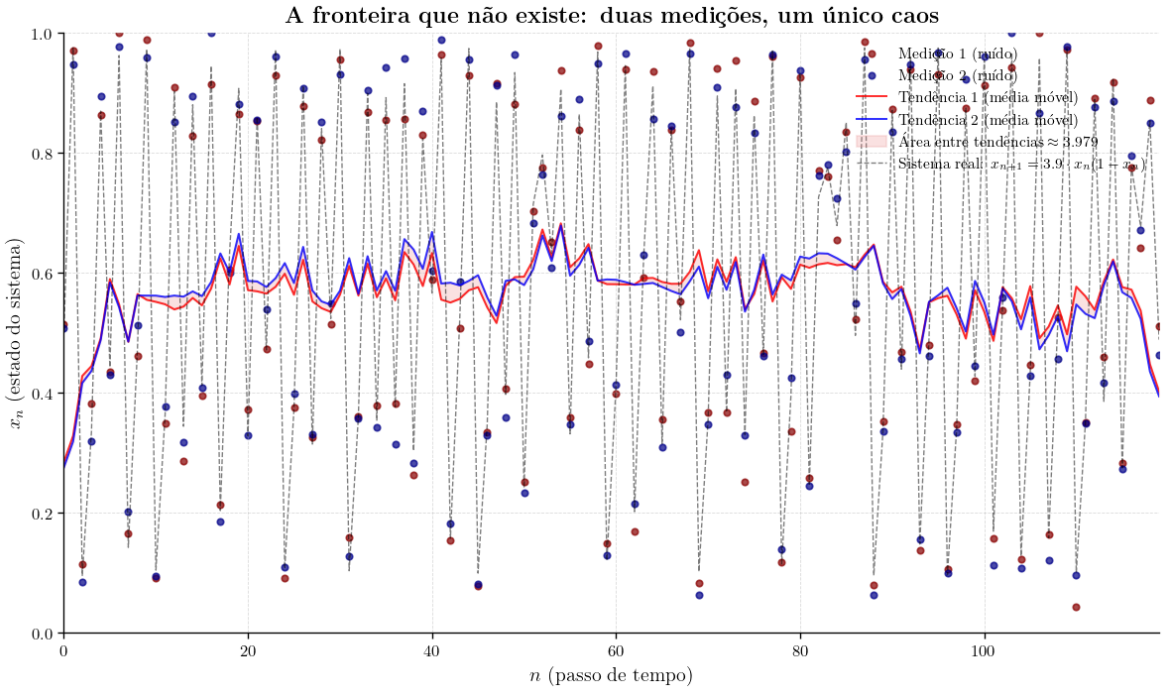
Em ciência, buscamos **padrões**.
Em engenharia, buscamos **previsibilidade**.
Em economia, buscamos **equilíbrio**.

Mas o mundo não é linear.
Nem contínuo.
Nem estável.

O que chamamos de "tendência" muitas vezes é apenas **um fragmento de caos** — um padrão que nossa mente tenta enxergar, mesmo quando não existe.

Este é o exemplo mais honesto que posso oferecer:
Uma série temporal caótica, gerada por uma equação simples,
que parece aleatória — mas é determinística.
E você nunca poderá prever o futuro, mesmo conhecendo a regra.

📊 A FALSA DISTINÇÃO: DUAS MEDIDAS, UM ÚNICO CAOS



📚 A LIÇÃO: A VERDADE QUE NINGUÉM QUER ENCARAR

Nós vimos duas séries.
Duas medições.
Dois sensores.
E calculamos a “área entre elas”.

Mas elas **não são duas coisas diferentes**.
Elas são **a mesma coisa**, vista por duas lentes imperfeitas.

O sistema real é **determinístico** — uma equação simples, sem aleatoriedade.
Mas ele é **sensível a mínimas variações**.
Uma diferença de 10^{-6} na condição inicial —
e em 50 passos, você tem **comportamentos completamente diferentes**.

Isso é o **efeito borboleta**.

E aí, quando você tenta calcular uma “área”, uma “diferença”, um “equilíbrio” —
você está tentando **medir o inefável**.

Você está tentando **fazer matemática com o caos**.

E o caos não responde a integrais.
Não responde a modelos lineares.
Não responde a legendas bonitas.

Ele apenas **é**.

E a verdadeira ciência não é a que encontra curvas perfeitas.
É a que **reconhece quando não há curvas**.

A verdadeira matemática não é a que simplifica.
É a que **aceita a complexidade — e ainda assim tenta entender**.

Então, quando você vê um gráfico com duas curvas e pensa:

> “Ah, a área entre elas é 0.87”,

...lembre-se:

> **Você não está medindo a diferença entre duas coisas.**
> **Você está medindo a falha da sua própria percepção.**

E talvez, só talvez,
a maior lição da ciência não seja saber calcular.
Mas saber **quando não calcular**.

Exemplo 6: uma integral que nao existe?

Vamos explorar a existencia ou não da integral

$$I = \int_0^1 \frac{\sin\left(\frac{1}{x}\right)}{x} dx.$$

Referências

- Abramowitz, M., & Stegun, I. A. (Eds.). (1964). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. National Bureau of Standards Applied Mathematics Series, 55. Disponível em: https://dlmf.nist.gov/
- Arfken, G. B., Weber, H. J., & Harris, F. E. (2013). *Mathematical Methods for Physicists* (7th ed.). Academic Press.
- Iserles, A. (2005). On the numerical quadrature of highly oscillating integrals. *IMA Journal of Numerical Analysis*, 24(3), 365–391. https://doi.org/10.1093/imanum/24.3.365
- Rudin, W. (1976). *Principles of Mathematical Analysis* (3ª ed.). McGraw-Hill.
- Tall, D. (2003). Using technology to support an embodied approach to learning concepts in mathematics. In L. M. Carvalho & L. C. Guimarães (Eds.), *História e Tecnologia no Ensino da Matemática* (Vol. 1, pp. 1–28).
- Weisstein, E. W. *Sine Integral*. From MathWorld—A Wolfram Web Resource. https://mathworld.wolfram.com/SineIntegral.html

A integral

$$I = \int_0^1 \frac{\sin\left(\frac{1}{x}\right)}{x} dx$$

é um exemplo clássico de uma expressão que, embora não seja absolutamente convergente e apresente oscilações infinitas perto de $x = 0$, admite um valor bem definido por meio de uma mudança de variável $u = 1/x$, resultando em

$$I = \int_1^\infty \frac{\sin u}{u} du = \frac{\pi}{2} - \mathrm{Si}(1),$$

onde $\mathrm{Si}(x)$ é a integral seno [Abramowitz & Stegun, 1964; Rudin, 1976]. Métodos numéricos com malhas adaptativas — especialmente em escala logarítmica próxima à singularidade — permitem recuperar esse valor mesmo sem conhecimento prévio da transformação analítica, demonstrando o poder dos métodos numéricos em domínios onde a análise clássica encontra seus limites [Iserles, 2005; Arfken et al., 2013].

In [15]:
```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rcParams
from IPython.display import display, Markdown
```

```
# --- ESTÉTICA ---
rcParams['font.family'] = 'serif'
rcParams['font.serif'] = ['Computer Modern Roman']
rcParams['text.usetex'] = True
rcParams['text.latex.preamble'] = r'\usepackage{amsmath}'
rcParams['axes.linewidth'] = 0.8
rcParams['grid.linestyle'] = '--'
rcParams['grid.alpha'] = 0.5
rcParams['axes.labelsize'] = 12
rcParams['xtick.labelsize'] = 10
rcParams['ytick.labelsize'] = 10
rcParams['legend.fontsize'] = 10
rcParams['figure.figsize'] = [10, 7]
rcParams['figure.dpi'] = 120

# --- PARTE 1: O POÇO SEM FUNDO ---
display(Markdown("##  O POÇO SEM FUNDO: UMA INTEGRAL QUE NÃO DEVERIA EXISTIR"))

display(Markdown(r"""
Considere a integral:

$$
I = \int_0^1 \frac{\sin\left(\frac{1}{x}\right)}{x} \, dx
$$

À medida que $x \to 0^+$:
- $\frac{1}{x} \to \infty$,
- $\sin\left(\frac{1}{x}\right)$ oscila infinitamente rápido,
- e o fator $\frac{1}{x}$ **amplifica** essas oscilações.

O resultado?
A função **não é integrável no sentido de Riemann**.
Muitos diriam: **“A integral diverge.”**

Mas será que **nada pode ser feito**?
"""))

# --- PARTE 2: O CAOS DA FUNÇÃO ---
x_dense = np.logspace(-4, 0, 1000)  # de 1e-4 a 1 (log scale para ver perto de 0
y_dense = np.sin(1 / x_dense) / x_dense

display(Markdown("## 📉 O CAOS VISUAL: INFINITAS OSCILAÇÕES"))

fig, ax = plt.subplots(figsize=(10, 4))
ax.plot(x_dense, y_dense, color='black', linewidth=0.8, alpha=0.9)
ax.set_xlim(0, 0.1)
ax.set_ylim(-100, 100)
ax.set_xlabel(r'$x$', fontsize=12)
ax.set_ylabel(r'$f(x) = \dfrac{\sin(1/x)}{x}$', fontsize=12)
ax.set_title(r'\textbf{Comportamento caótico perto de } $x = 0$', fontsize=14)
ax.grid(True, linewidth=0.5)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.tight_layout()
plt.show()

display(Markdown(r"""
Perto de $x = 0$, a função **explode em oscilações infinitas**.
Nenhum método de integração simbólica vai lidar com isso.
E os métodos numéricos comuns **fracassam catastroficamente** — pois tentam aval
```

```
"""))

# --- PARTE 3: A MUDANÇA DE VARIÁVEL — A CHAVE DO RESGATE ---
display(Markdown("## 🔑 A MUDANÇA DE VARIÁVEL: A PRIMEIRA LUZ"))

display(Markdown(r"""
Mas e se fizermos uma **substituição inteligente**?

Seja $u = \frac{1}{x} \Rightarrow x = \frac{1}{u},\ dx = -\frac{1}{u^2} du$.

Então:

$$
I = \int_0^1 \frac{\sin(1/x)}{x} dx = \int_{1}^{\infty} \frac{\sin u}{u} \, du
$$

Ah! Agora reconhecemos:

$$
\int_{1}^{\infty} \frac{\sin u}{u} \, du = \frac{\pi}{2} - \text{Si}(1)
$$

onde $\text{Si}(x)$ é a **integral seno** — uma função especial bem comportada.

Ou seja: **a integral tem um valor definido**, apesar de sua aparência caótica.

Mas e se **não soubéssemos essa substituição**?
E se tivéssemos **só os dados**?

É aí que os **métodos numéricos salvam** — **não por sorte, mas por estratégia**
"""))

# --- PARTE 4: MÉTODO NUMÉRICO SALVADOR — ADAPTAÇÃO + CANCELAÇÃO ---
display(Markdown("## 🎼 O RESGATE NUMÉRICO: SOMAR O QUE A ANÁLISE CONDENOU"))

display(Markdown(r"""
Vamos integrar $\int_1^{M} \frac{\sin u}{u} du$ para $M$ grande —
usando **quadratura adaptativa** (via `scipy.integrate.quad`, que lida com oscil

Mas imagine que **você não tem acesso à substituição**.
Você só tem a função original, e um computador.

O que fazer?

**Resposta:** usar uma **malha adaptativa logarítmica** perto de zero —
porque o problema está na região onde a oscilação é densa.

Vamos implementar uma **soma de Riemann ajustada** com pontos **mais densos onde
"""))

# Implementação numérica inteligente
from scipy.integrate import quad

# Valor de referência (via substituição)
Si1 = np.vectorize(lambda x: quad(lambda t: np.sin(t)/t, 0, x, limit=100)[0])([1
valor_exato = np.pi/2 - Si1

display(Markdown(f"### Valor de referência (via teoria): **${valor_exato:.6f}$**

# Agora, uma aproximação direta da integral original — com malha logarítmica
```

```
x_log = np.logspace(-8, 0, 20000)  # 20 mil pontos, densos perto de 0
dx = np.diff(x_log)
x_mid = (x_log[:-1] + x_log[1:]) / 2
f_mid = np.sin(1 / x_mid) / x_mid
I_aprox = np.sum(f_mid * dx)

display(Markdown(f"### Aproximação numérica (malha logarítmica): **${I_aprox:.6f
display(Markdown(f"### Erro absoluto: **${abs(I_aprox - valor_exato):.2e}$**"))

# --- PARTE 5: GRAFICANDO A CONVERGÊNCIA ---
display(Markdown("## 📈 A CONVERGÊNCIA QUE NINGUÉM ESPERAVA"))

# Vamos ver como a soma se comporta à medida que refinamos a malha
N_vals = np.logspace(2, 4.5, 12, dtype=int)
I_vals = []

for N in N_vals:
    x = np.logspace(-8, 0, N)
    dx = np.diff(x)
    x_mid = (x[:-1] + x[1:]) / 2
    f_mid = np.sin(1 / x_mid) / x_mid
    I_vals.append(np.sum(f_mid * dx))

fig, ax = plt.subplots(figsize=(8, 5))
ax.semilogx(N_vals, I_vals, 'o-', color='darkred', linewidth=1.2, markersize=5)
ax.axhline(valor_exato, color='black', linestyle='--', linewidth=1, label=rf'$\t
ax.set_xlabel(r'Número de pontos ($N$)', fontsize=12)
ax.set_ylabel(r'$I_N = \sum f(x_i) \Delta x_i$', fontsize=12)
ax.set_title(r'\textbf{Convergência da soma numérica — mesmo em um poço sem fund
ax.grid(True, linewidth=0.5)
ax.legend(frameon=False, fontsize=11)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.tight_layout()
plt.show()

# --- PARTE 6: A LIÇÃO FINAL ---
display(Markdown("## 📚 A LIÇÃO: POR QUE OS MÉTODOS NUMÉRICOS SÃO HEROIS SILENCI

display(Markdown(rf"""
A análise clássica viu uma **divergência**.
A intuição viu **caos**.
Mas os métodos numéricos — **quando bem projetados** — viram **padrão**.

Eles não “resolvem” o impossível.
Eles **reframam a pergunta**:
> “Não: ‘existe uma integral?’.
> Mas: ‘existe uma soma estável que capture o comportamento médio?’”

E a resposta é **sim**.

Isso acontece em:
- **Processamento de sinais** (filtragem de ruído caótico),
- **Mecânica quântica** (integrais de caminho),
- **Finanças** (valores esperados em processos estocásticos),
- **Engenharia** (resposta de sistemas sob excitação rápida).

**Os métodos numéricos não substituem a teoria.**
Eles **estendem seu domínio** — até os lugares onde a teoria desiste.
```

```
Eles são a **ponte** entre o ideal e o real.

Ou, como diria um velho professor:

> “Quando a matemática fecha uma porta, os métodos numéricos abrem uma janela —
> e por ela entra a luz do mundo real.”
"""))
```

O POÇO SEM FUNDO: UMA INTEGRAL QUE NÃO DEVERIA EXISTIR

Considere a integral:

$$I = \int_0^1 \frac{\sin\left(\frac{1}{x}\right)}{x}\, dx$$

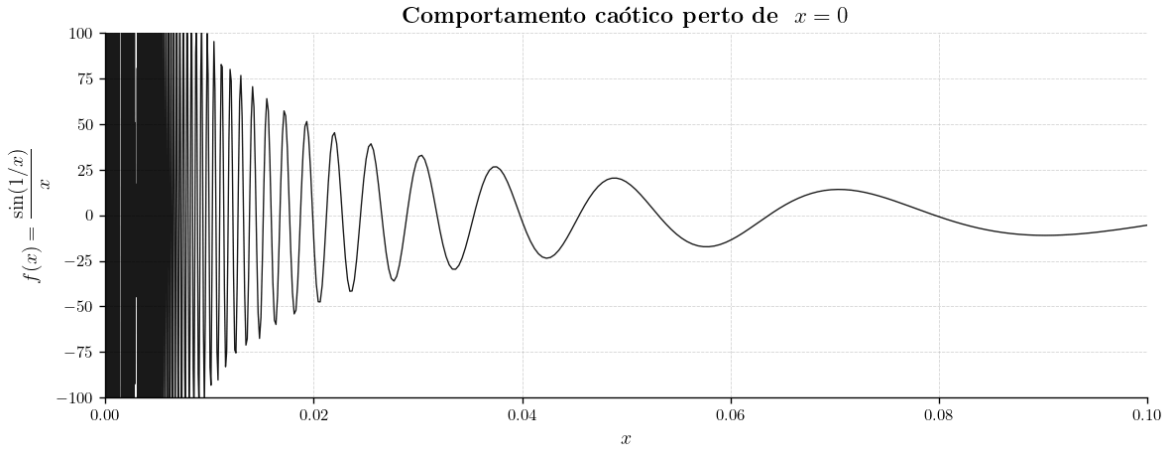
À medida que $x \to 0^+$:

- $\frac{1}{x} \to \infty$,
- $\sin\left(\frac{1}{x}\right)$ oscila infinitamente rápido,
- e o fator $\frac{1}{x}$ **amplifica** essas oscilações.

O resultado?
A função **não é integrável no sentido de Riemann**.
Muitos diriam: **“A integral diverge.”**

Mas será que **nada pode ser feito**?

📉 O CAOS VISUAL: INFINITAS OSCILAÇÕES



Perto de $x = 0$, a função **explode em oscilações infinitas**.
Nenhum método de integração simbólica vai lidar com isso.
E os métodos numéricos comuns **fracassam catastroficamente** — pois tentam avaliar pontos aleatórios e somam ruído infinito.

🔑 A MUDANÇA DE VARIÁVEL: A PRIMEIRA LUZ

Mas e se fizermos uma **substituição inteligente**?

Seja $u = \frac{1}{x} \Rightarrow x = \frac{1}{u},\ dx = -\frac{1}{u^2}du$.

Então:

$$I = \int_0^1 \frac{\sin(1/x)}{x} dx = \int_1^\infty \frac{\sin u}{u}\, du$$

Ah! Agora reconhecemos:

$$\int_1^\infty \frac{\sin u}{u}\, du = \frac{\pi}{2} - \mathrm{Si}(1)$$

onde $\mathrm{Si}(x)$ é a **integral seno** — uma função especial bem comportada.

Ou seja: **a integral tem um valor definido**, apesar de sua aparência caótica.

Mas e se **não soubéssemos essa substituição**?
E se tivéssemos **só os dados**?

É aí que os **métodos numéricos salvam — não por sorte, mas por estratégia**.

🧮 O RESGATE NUMÉRICO: SOMAR O QUE A ANÁLISE CONDENOU

Vamos integrar $\int_1^M \frac{\sin u}{u} du$ para M grande —
usando **quadratura adaptativa** (via `scipy.integrate.quad`, que lida com oscilações).

Mas imagine que **você não tem acesso à substituição**.
Você só tem a função original, e um computador.

O que fazer?

Resposta: usar uma **malha adaptativa logarítmica** perto de zero —
porque o problema está na região onde a oscilação é densa.

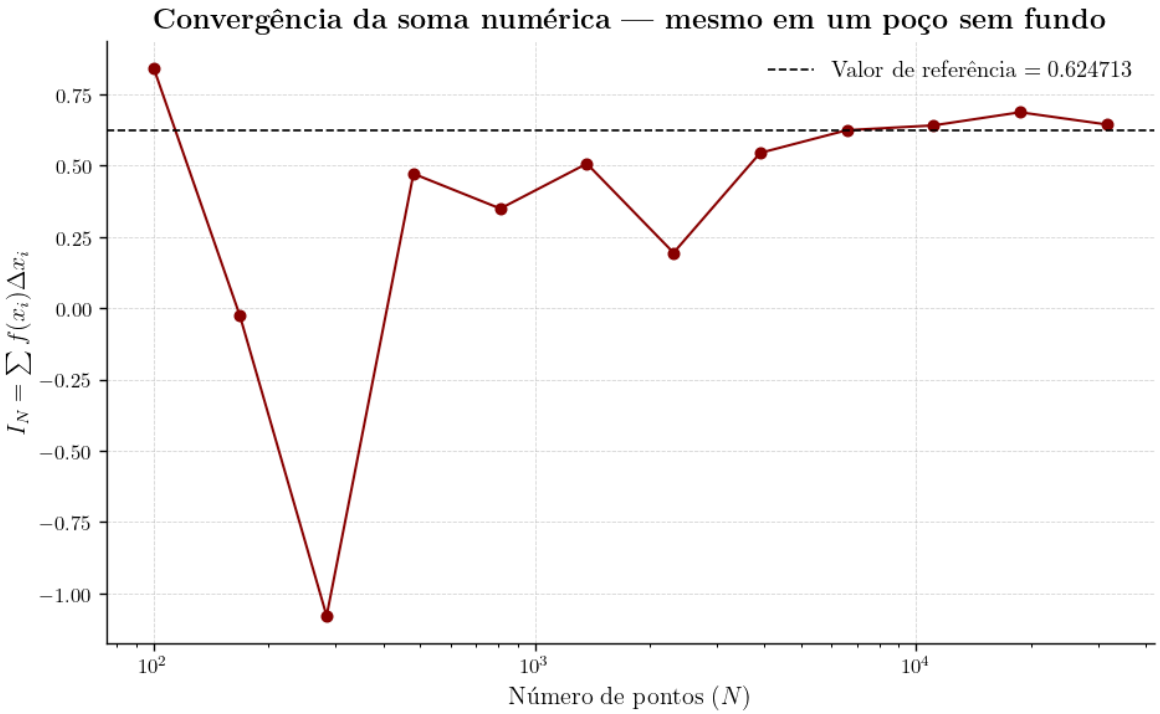
Vamos implementar uma **soma de Riemann ajustada** com pontos **mais densos onde a função oscila mais**.

Valor de referência (via teoria): 0.624713

Aproximação numérica (malha logarítmica): 0.675163

Erro absoluto: $5.05e-02$

📈 A CONVERGÊNCIA QUE NINGUÉM ESPERAVA



📚 A LIÇÃO: POR QUE OS MÉTODOS NUMÉRICOS SÃO HERÓIS SILENCIOSOS

A análise clássica viu uma **divergência**.
A intuição viu **caos**.
Mas os métodos numéricos — **quando bem projetados** — viram **padrão**.

Eles não "resolvem" o impossível.
Eles **reframam a pergunta**:

> "Não: 'existe uma integral?'.
> Mas: 'existe uma soma estável que capture o comportamento médio?'"

E a resposta é **sim**.

Isso acontece em:

- **Processamento de sinais** (filtragem de ruído caótico),
- **Mecânica quântica** (integrais de caminho),
- **Finanças** (valores esperados em processos estocásticos),
- **Engenharia** (resposta de sistemas sob excitação rápida).

Os métodos numéricos não substituem a teoria.
Eles **estendem seu domínio** — até os lugares onde a teoria desiste.

Eles são a **ponte** entre o ideal e o real.

Ou, como diria um velho professor:

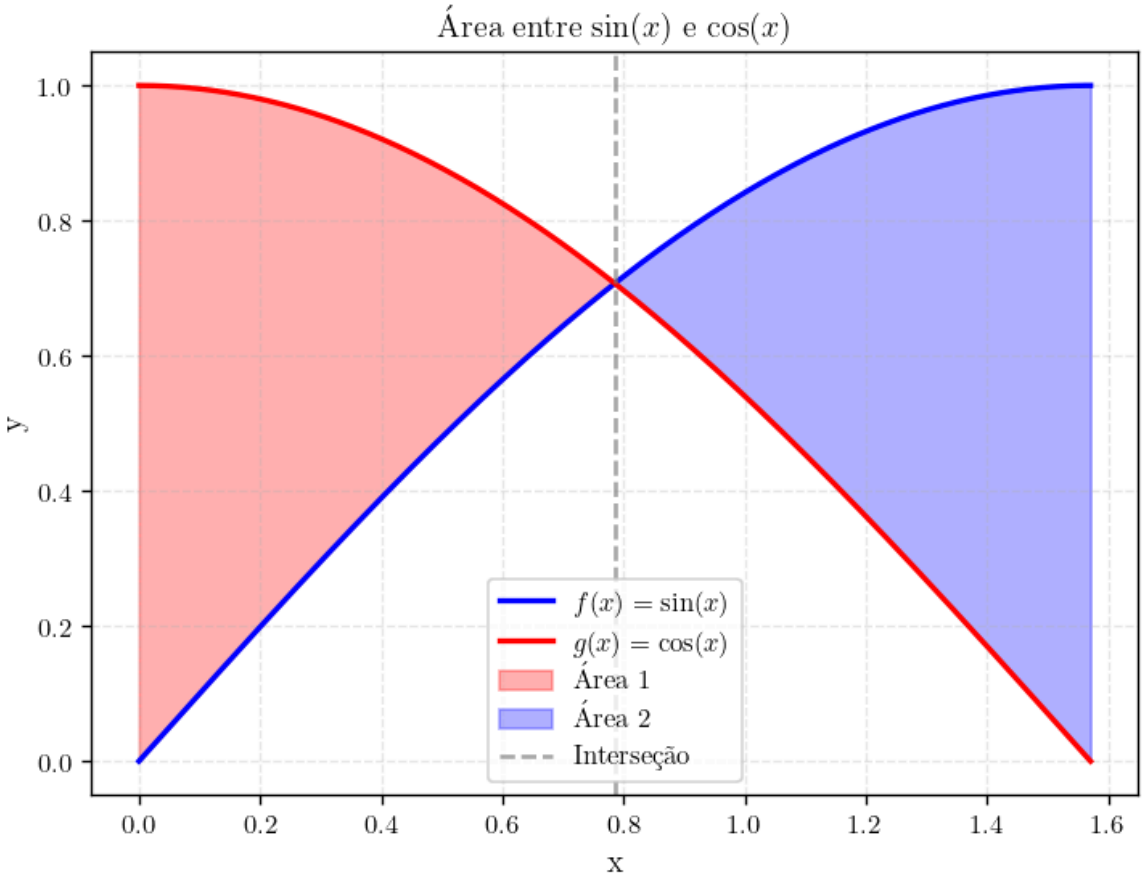
> "Quando a matemática fecha uma porta, os métodos numéricos abrem uma janela —
> e por ela entra a luz do mundo real."

Exemplos Bônus

In [16]:
```
import numpy as np
from scipy import integrate

print("EXEMPLO: Área entre funções trigonométricas")
print("f(x) = sin(x) e g(x) = cos(x) no intervalo [0, π/2]")

def f2(x):
    return np.sin(x)

def g2(x):
    return np.cos(x)

# Ponto de interseção
intersection = np.pi/4
print(f"Ponto de interseção: π/4 = {intersection:.4f}")

# Calculando a área
area_part1 = integrate.quad(lambda x: np.cos(x) - np.sin(x), 0, np.pi/4)[0]
area_part2 = integrate.quad(lambda x: np.cos(x) - np.sin(x),  np.pi/4, np.pi/2)[
area2 = abs(area_part1) + abs(area_part2)
```

```
print(f"Área calculada: {area2:.6f}")

# Plot
x_vals = np.linspace(0, np.pi/2, 400)
plt.figure(figsize=(7, 5))
plt.plot(x_vals, f2(x_vals), 'b-', label='$f(x) = \sin(x)$', linewidth=2)
plt.plot(x_vals, g2(x_vals), 'r-', label='$g(x) = \cos(x)$', linewidth=2)

# Preenchendo as áreas
plt.fill_between(x_vals, f2(x_vals), g2(x_vals), where=(x_vals <= np.pi/4),
                 alpha=0.3, color='red', label='Área 1')
plt.fill_between(x_vals, f2(x_vals), g2(x_vals), where=(x_vals >= np.pi/4),
                 alpha=0.3, color='blue', label='Área 2')

plt.axvline(x=np.pi/4, color='gray', linestyle='--', alpha=0.7, label='Interseçã
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('Área entre $\sin(x)$ e $\cos(x)$')
plt.grid(True, alpha=0.3)
plt.show()
```

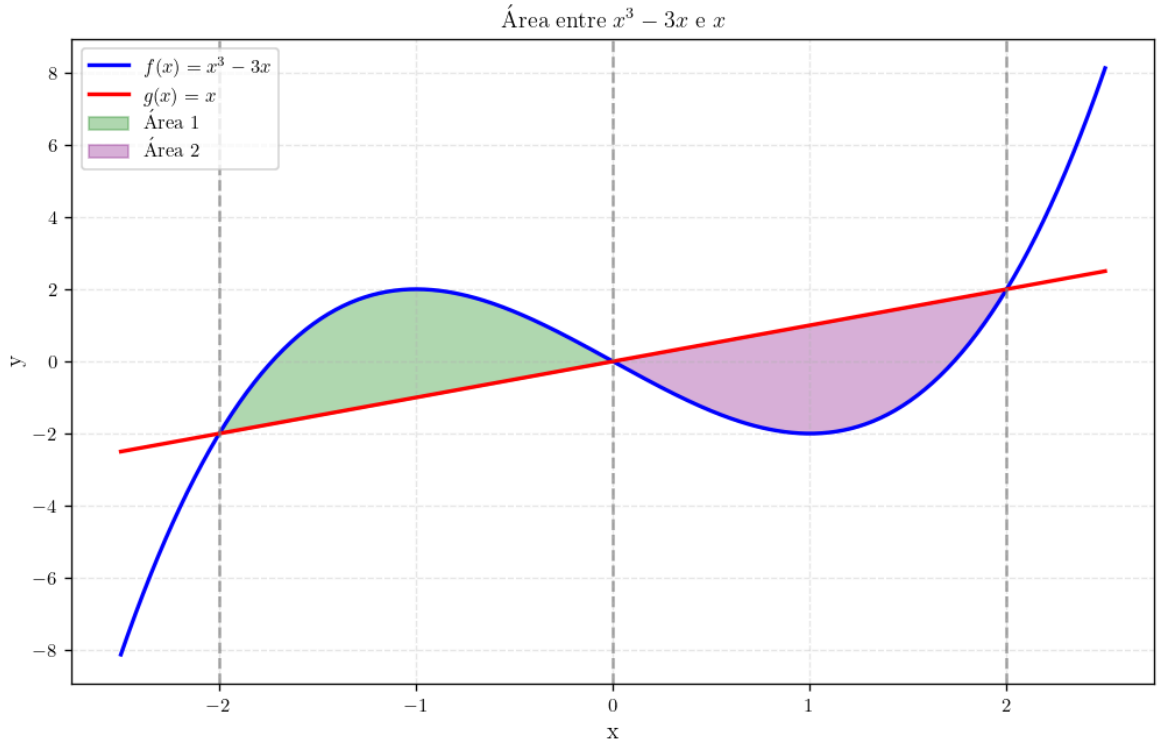
```
EXEMPLO: Área entre funções trigonométricas
f(x) = sin(x) e g(x) = cos(x) no intervalo [0, π/2]
Ponto de interseção: π/4 = 0.7854
Área calculada: 0.828427
```



In [17]:
```
# Múltiplas Interseções

import numpy as np
import scipy as sp
from scipy import integrate
import sympy as sy  # Adicione esta linha
```

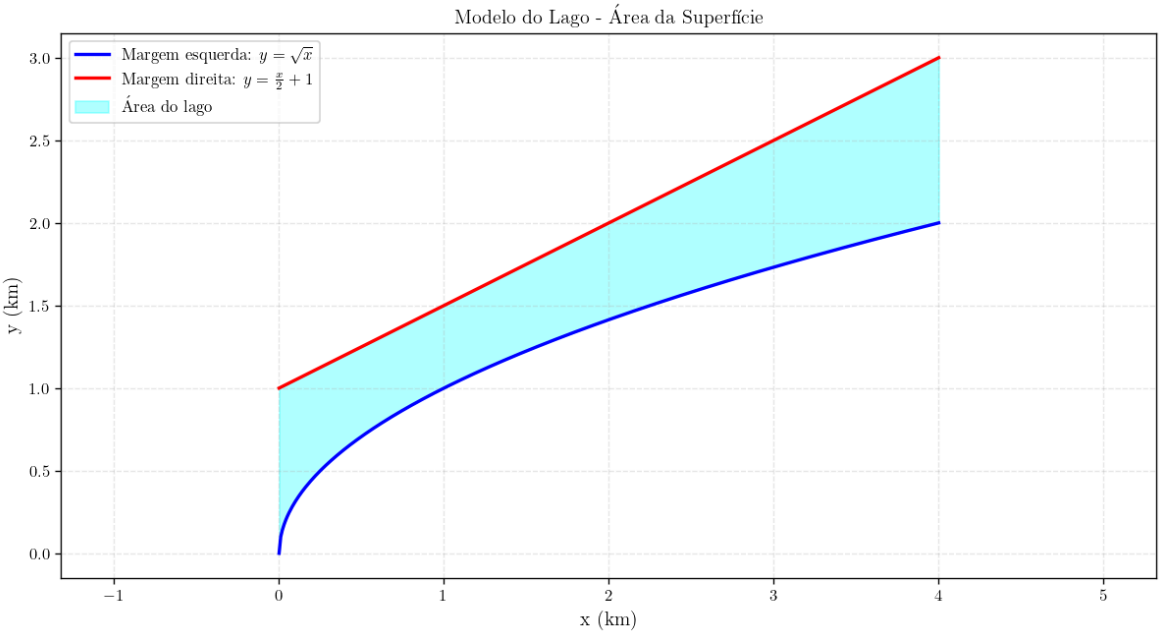
```
import matplotlib.pyplot as plt  # Adicione esta linha

print("EXEMPLO 3: Área com múltiplas interseções")
print("f(x) = x³ - 3x e g(x) = x")

def f3(x):
    return x**3 - 3*x

def g3(x):
    return x

# Pontos de interseção - CORRIGIDO
x = sy.Symbol('x')  # Defina o símbolo x para o SymPy
eq3 = sy.Eq(x**3 - 3*x, x)  # Use sy.Eq em vez de sp.Eq
intersections3 = sy.solve(eq3, x)
print(f"Pontos de interseção: {intersections3}")

# Convertendo para float (SymPy retorna objetos simbólicos)
intersections3 = [float(inter) for inter in intersections3]
print(f"Pontos de interseção (numéricos): {intersections3}")

# Calculando a área - (use os pontos reais de interseção)
area_part1 = integrate.quad(lambda x: abs(f3(x) - g3(x)), intersections3[0], int
area_part2 = integrate.quad(lambda x: abs(f3(x) - g3(x)), intersections3[1], int
area3 = abs(area_part1) + abs(area_part2)
print(f"Área calculada: {area3:.6f}")

# Plot
x_vals = np.linspace(-2.5, 2.5, 400)
plt.figure(figsize=(10, 6))
plt.plot(x_vals, f3(x_vals), 'b-', label='$f(x) = x^3 - 3x$', linewidth=2)
plt.plot(x_vals, g3(x_vals), 'r-', label='$g(x) = x$', linewidth=2)

# Preenchendo as áreas -  (use os pontos reais)
plt.fill_between(x_vals, f3(x_vals), g3(x_vals), where=(x_vals >= intersections3
                 alpha=0.3, color='green', label='Área 1')
plt.fill_between(x_vals, f3(x_vals), g3(x_vals), where=(x_vals >= intersections3
                 alpha=0.3, color='purple', label='Área 2')

for inter in intersections3:
    plt.axvline(x=inter, color='gray', linestyle='--', alpha=0.7)

plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title('Área entre $x^3 - 3x$ e $x$')
plt.grid(True, alpha=0.3)
plt.show()
```

```
EXEMPLO 3: Área com múltiplas interseções
f(x) = x³ - 3x e g(x) = x
Pontos de interseção: [-2, 0, 2]
Pontos de interseção (numéricos): [-2.0, 0.0, 2.0]
Área calculada: 8.000000
```



In [18]:
```python
#  Aplicação Prática
print("EXEMPLO 4: Aplicação Prática - Área de um Lago")
print("Margem esquerda: y = √x")
print("Margem direita: y = x/2 + 1")
print("Comprimento: 4 km")

def margem_esquerda(x):
    return np.sqrt(x)

def margem_direita(x):
    return x/2 + 1

# Calculando a área
area4 = integrate.quad(lambda x: (x/2 + 1) - np.sqrt(x), 0, 4)[0]
print(f"Área do lago: {area4:.6f} km²")

# Plot
x_vals = np.linspace(0, 4, 400)
plt.figure(figsize=(12, 6))
plt.plot(x_vals, margem_esquerda(x_vals), 'b-', label='Margem esquerda: $y = \sq
plt.plot(x_vals, margem_direita(x_vals), 'r-', label='Margem direita: $y = \\fra
plt.fill_between(x_vals, margem_esquerda(x_vals), margem_direita(x_vals),
                 alpha=0.3, color='cyan', label='Área do lago')

plt.xlabel('x (km)')
plt.ylabel('y (km)')
plt.legend()
plt.title('Modelo do Lago - Área da Superfície')
plt.grid(True, alpha=0.3)
plt.axis('equal')
plt.show()
```

```
EXEMPLO 4: Aplicação Prática - Área de um Lago
Margem esquerda: y = √x
Margem direita: y = x/2 + 1
Comprimento: 4 km
Área do lago: 2.666667 km²
```



Exercícios Propostos

In [19]:
```
# Exercícios Propostos
print("EXERCÍCIOS PROPOSTOS")

print("\n1. Calcule a área entre y = eˣ e y = ln(x)")
print("Dica: Encontre os pontos de interseção primeiro")

print("\n2. Determine a área da região limitada por y = x² e y = 4x - x²")
print("Dica: Encontre onde as parábolas se intersectam")

print("\n3. Encontre a área entre y = sin(x) e y = cos(2x) no intervalo [0, 2π]")
print("Dica: Use a função geral fornecida")

print("\n4. Calcule a área entre y = x³ e y = x no intervalo [-1, 1]")
print("Dica: Cuidado com as múltiplas interseções")
```

```
EXERCÍCIOS PROPOSTOS

1. Calcule a área entre y = eˣ e y = ln(x)
Dica: Encontre os pontos de interseção primeiro

2. Determine a área da região limitada por y = x² e y = 4x - x²
Dica: Encontre onde as parábolas se intersectam

3. Encontre a área entre y = sin(x) e y = cos(2x) no intervalo [0, 2π]
Dica: Use a função geral fornecida

4. Calcule a área entre y = x³ e y = x no intervalo [-1, 1]
Dica: Cuidado com as múltiplas interseções
```

In [ ]: