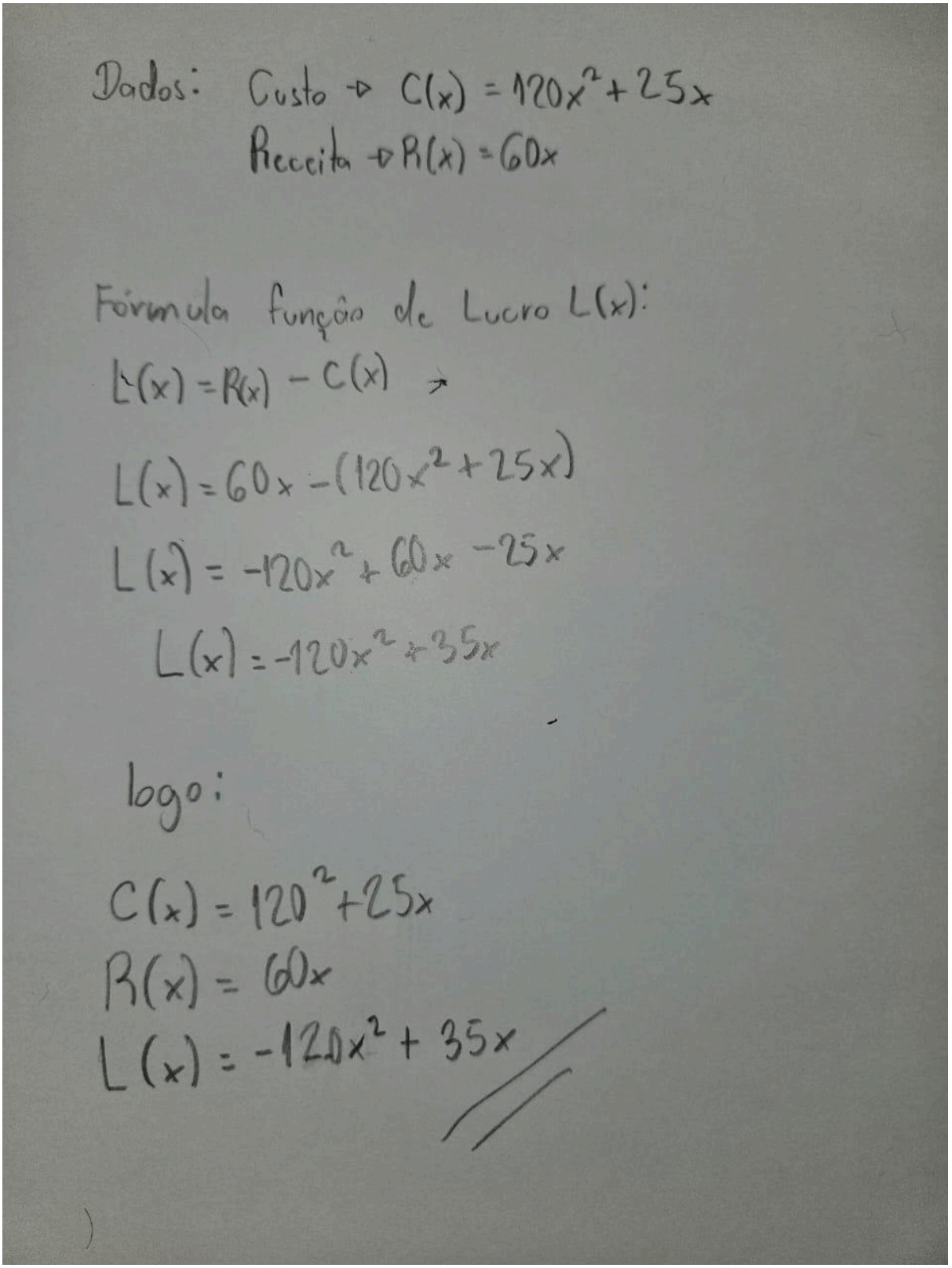
Checkpoint 3 - Differentiated Problem Solving
Jessica Rohden Schlickmann

# Modelagem e Otimização de um Sistema de Produção usando Python

Turma - 1ESPH
Felipe Men dos Santos - 557571
Otto Oliveira Candido - 557054
Lucas Rodrigues de Queiroz - 556323
João Pedro Silva Pinheiro - 557013

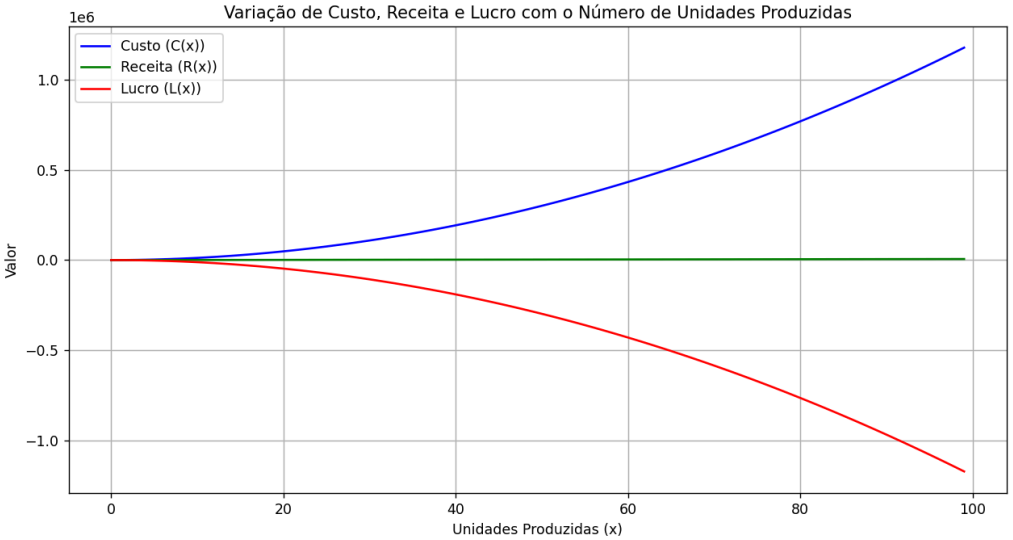
**OBS:** Alguns códigos não couberam inteiros na print, vou deixar um link com os código completos para testar e verificar no GITHUB no final do arquivo.

**1. Estipular a função Custo, Receita e Lucro de produção;**



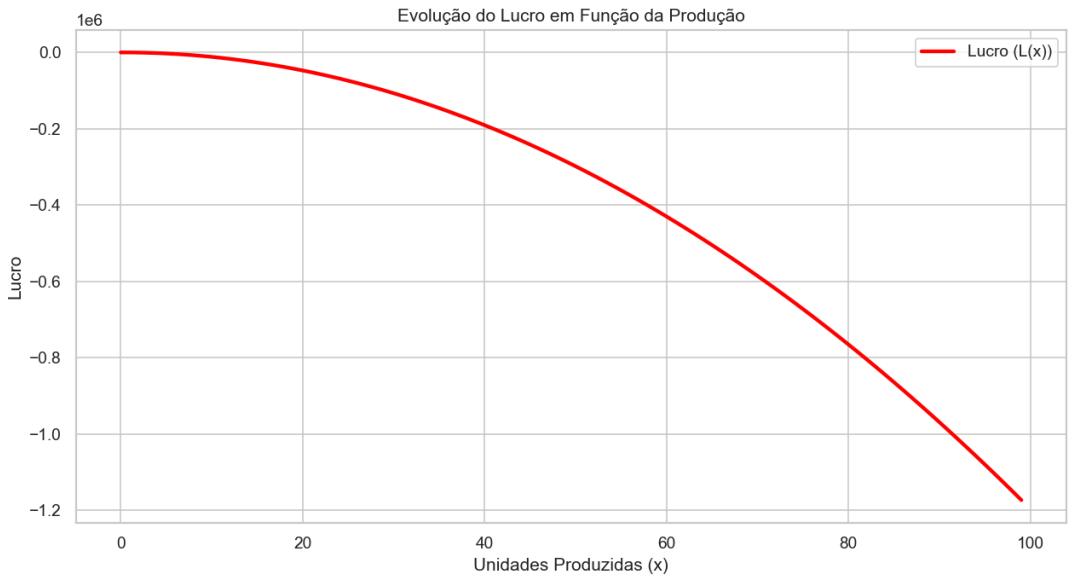
## 2. Implementar simulações no Python, variando o número de unidades produzidas e observando como o custo, receita e lucro variam;

```python
# Funções para Custo, Receita e Lucro
2 usages
def custo(x):
    return 120 * x**2 + 25 * x

2 usages
def receita(x):
    return 60 * x

1 usage
def lucro(x):
    return receita(x) - custo(x)

# Variando o número de unidades produzidas
x_values = np.arange(0, 100, 1)
custo_values = custo(x_values)
receita_values = receita(x_values)
lucro_values = lucro(x_values)

# Plotando os resultados
plt.figure(figsize=(12, 6))
plt.plot( *args: x_values, custo_values, label="Custo (C(x))", color="blue")
plt.plot( *args: x_values, receita_values, label="Receita (R(x))", color="green")
plt.plot( *args: x_values, lucro_values, label="Lucro (L(x))", color="red")
plt.xlabel("Unidades Produzidas (x)")
plt.ylabel("Valor")
plt.title("Variação de Custo, Receita e Lucro com o Número de Unidades Produzidas")
plt.legend()
plt.grid(True)
plt.show()
```



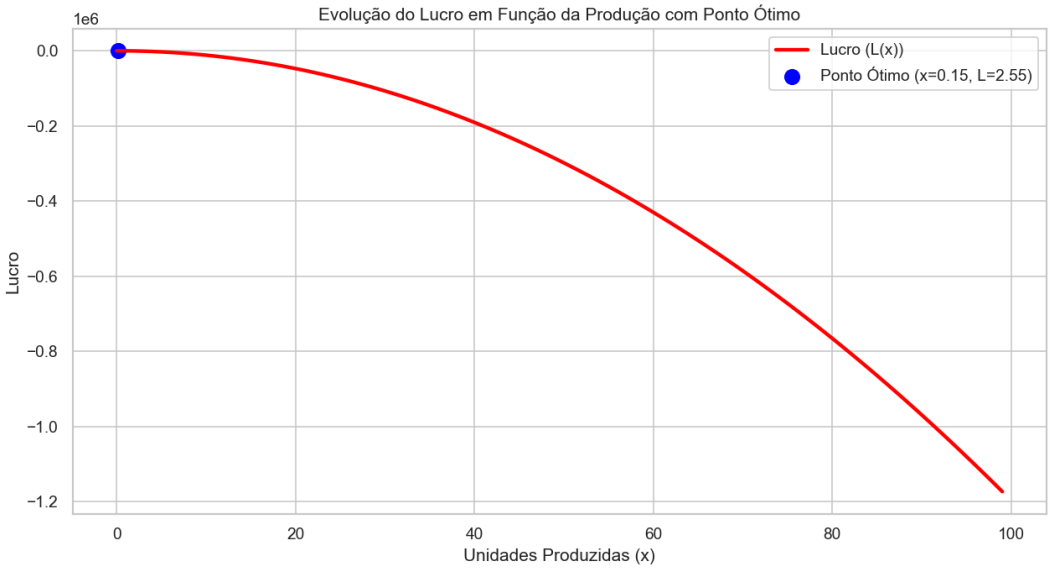
## 3. Usar bibliotecas como Matplotlib ou Seaborn para criar gráficos que mostrem a evolução do lucro em função da produção.

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Funções para Custo, Receita e Lucro
1 usage
def custo(x):
    return 120 * x**2 + 25 * x

1 usage
def receita(x):
    return 60 * x

1 usage
def lucro(x):
    return receita(x) - custo(x)

# Variando o número de unidades produzidas
x_values = np.arange(0, 100, 1)
lucro_values = lucro(x_values)

# Configuração do estilo do Seaborn
sns.set(style="whitegrid")

# Criando o gráfico para a evolução do lucro em função da produção
plt.figure(figsize=(12, 6))
sns.lineplot(x=x_values, y=lucro_values, color="red", linewidth=2.5, label="Lucro (L(x))")
plt.xlabel("Unidades Produzidas (x)")
plt.ylabel("Lucro")
plt.title("Evolução do Lucro em Função da Produção")
plt.legend()
plt.grid(True)
plt.show()
```



## 4. Exibir o ponto ótimo de produção no gráfico.

```python
def receita(x):
    return 60 * x

3 usages
def lucro(x):
    return receita(x) - custo(x)

# Função negativa do lucro para maximizar usando fmin
1 usage
def lucro_neg(x):
    return -lucro(x)

# Encontrando o ponto ótimo
x_otimo = fmin(lucro_neg, x0=1, disp=False)[0]
lucro_otimo = lucro(x_otimo)

# Variando o número de unidades produzidas
x_values = np.arange(0, 100, 1)
lucro_values = lucro(x_values)

# Plotando o gráfico com o ponto ótimo
sns.set(style="whitegrid")
plt.figure(figsize=(12, 6))
sns.lineplot(x=x_values, y=lucro_values, color="red", linewidth=2.5, label="Lucro (L(x))")
plt.scatter(x_otimo, lucro_otimo, color="blue", s=100, label=f'Ponto Ótimo (x={x_otimo:.2f}, L={lucro_otimo:.2f})')
plt.xlabel("Unidades Produzidas (x)")
plt.ylabel("Lucro")
plt.title("Evolução do Lucro em Função da Produção com Ponto Ótimo")
plt.legend()
plt.grid(True)
plt.show()
```



## 5. Considerar fatores externos que possam impactar a produção, como impostos, variação no custo das matérias-primas, etc

Para incorporar fatores externos que afetam a produção, como impostos e variação no custo das matérias-primas, podemos ajustar a função de custo ou receita para refletir essas variações. Aqui estão algumas sugestões de ajustes:

### 1. Impostos sobre Produção ou Lucro

- Um imposto fixo pode ser aplicado ao lucro final, ou podemos adicionar uma porcentagem fixa ao custo total.
- Suponha que o imposto seja de $t$% sobre o lucro. A nova função lucro será:

$$L_{ajustado}(x) = L(x) \cdot (1 - t / 100)$$

### 2. Variação no Custo das Matérias-Primas

- A variação no custo das matérias-primas pode ser representada com um fator multiplicativo no custo. Se o custo das matérias-primas aumenta em $m$%, o novo custo é:

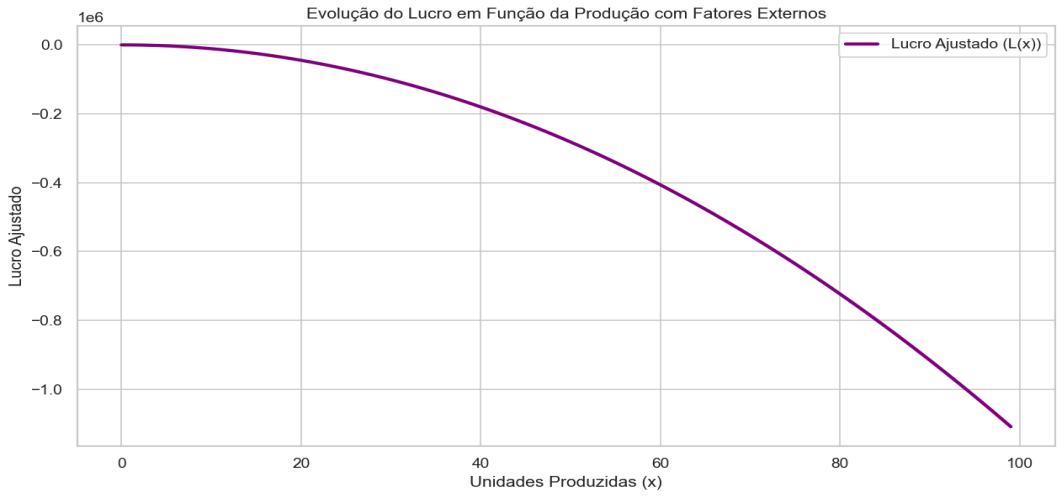
$$C_{ajustado}(x) = C(x) \cdot (1 + m / 100)$$

### 3. Custo de Mão de Obra e Logística Variáveis

- Podemos também considerar um fator variável para a logística, como aumento percentual baseado na quantidade produzida.

### Implementação com Ajustes Externos

Aqui está um exemplo que integra um imposto de 10% sobre o lucro e uma variação de 5% no custo das matérias-primas.



```python
taxa_imposto = 10  # em porcentagem sobre o lucro
aumento_materia_prima = 5  # em porcentagem de aumento no custo das matérias-primas

# Funções de Custo, Receita e Lucro ajustadas
1 usage
def custo_ajustado(x):
    return (120 * x**2 + 25 * x) * (1 + aumento_materia_prima / 100)

1 usage
def receita(x):
    return 60 * x

1 usage
def lucro_ajustado(x):
    lucro_bruto = receita(x) - custo_ajustado(x)
    return lucro_bruto * (1 - taxa_imposto / 100)

# Variando o número de unidades produzidas
x_values = np.arange(0, 100, 1)
lucro_values = lucro_ajustado(x_values)

# Plotando o gráfico com as novas variáveis
sns.set(style="whitegrid")
plt.figure(figsize=(12, 6))
sns.lineplot(x=x_values, y=lucro_values, color="purple", linewidth=2.5, label="Lucro Ajustado (L(x))")
plt.xlabel("Unidades Produzidas (x)")
plt.ylabel("Lucro Ajustado")
plt.title("Evolução do Lucro em Função da Produção com Fatores Externos")
plt.legend()
plt.grid(True)
plt.show()
```

**GITHUB:** https://github.com/FelipeMenDosSantos/CP3-Problem-Solving