

## Modelagem e Otimização de um Sistema de Produção usando Python

**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;**

Dados: Custo  $\rightarrow C(x) = 120x^2 + 25x$   
Receita  $\rightarrow R(x) = 60x$

Fórmula função de Lucro  $L(x)$ :

$$L(x) = R(x) - C(x) \rightarrow$$

$$L(x) = 60x - (120x^2 + 25x)$$

$$L(x) = -120x^2 + 60x - 25x$$

$$L(x) = -120x^2 + 35x$$

logo:

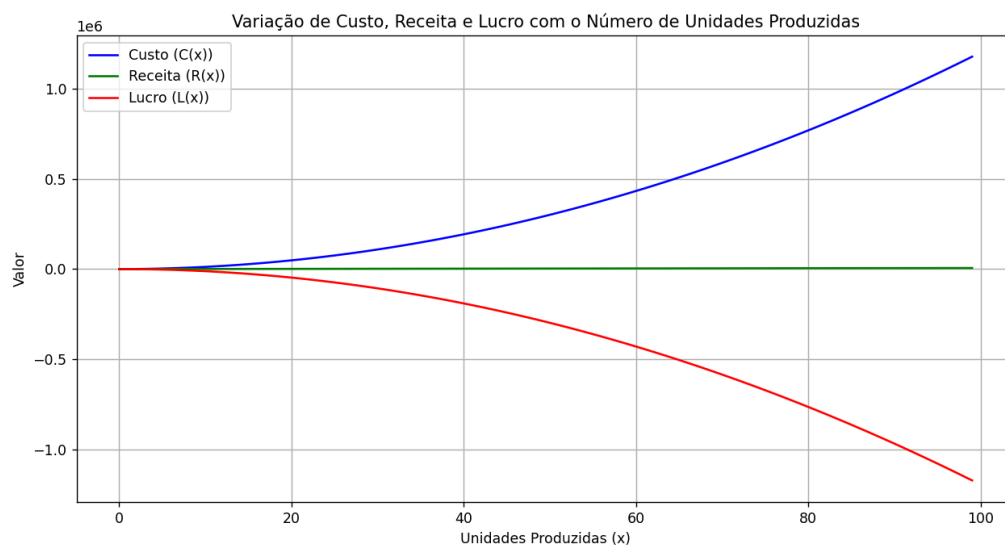
$$C(x) = 120x^2 + 25x$$

$$R(x) = 60x$$

$$L(x) = -120x^2 + 35x$$

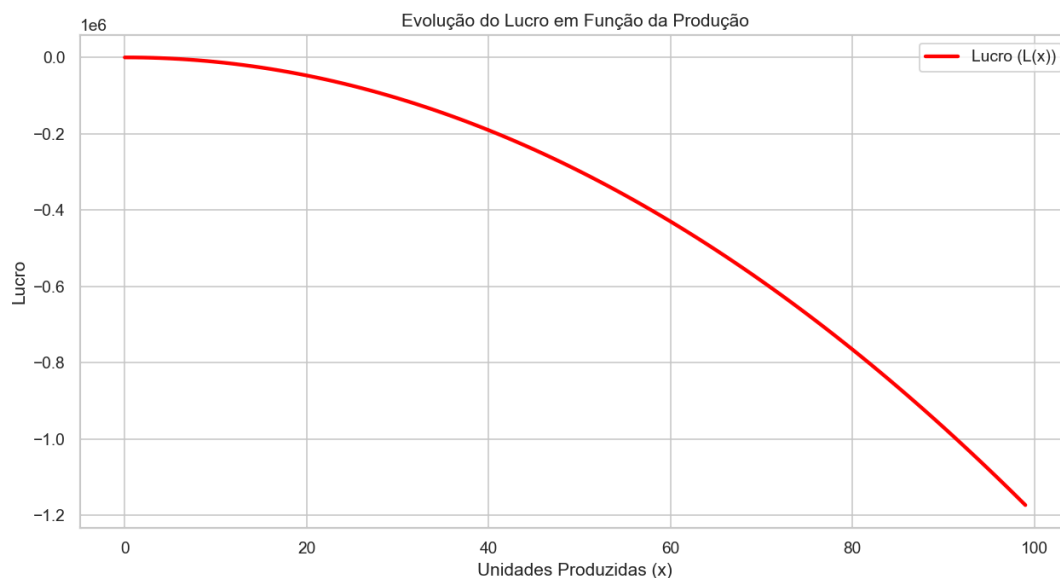
## 2. Implementar simulações no Python, variando o número de unidades produzidas e observando como o custo, receita e lucro variam;

```
4 # Funções para Custo, Receita e Lucro
  2 usages
5 def custo(x):
6     return 120 * x**2 + 25 * x
7
  2 usages
8 def receita(x):
9     return 60 * x
10
  1 usage
11 def lucro(x):
12     return receita(x) - custo(x)
13
14 # Variando o número de unidades produzidas
15 x_values = np.arange(0, 100, 1)
16 custo_values = custo(x_values)
17 receita_values = receita(x_values)
18 lucro_values = lucro(x_values)
19
20 # Plotando os resultados
21 plt.figure(figsize=(12, 6))
22 plt.plot(*args: x_values, custo_values, label="Custo (C(x))", color="blue")
23 plt.plot(*args: x_values, receita_values, label="Receita (R(x))", color="green")
24 plt.plot(*args: x_values, lucro_values, label="Lucro (L(x))", color="red")
25 plt.xlabel("Unidades Produzidas (x)")
26 plt.ylabel("Valor")
27 plt.title("Variação de Custo, Receita e Lucro com o Número de Unidades Produzidas")
28 plt.legend()
29 plt.grid(True)
30 plt.show()
```



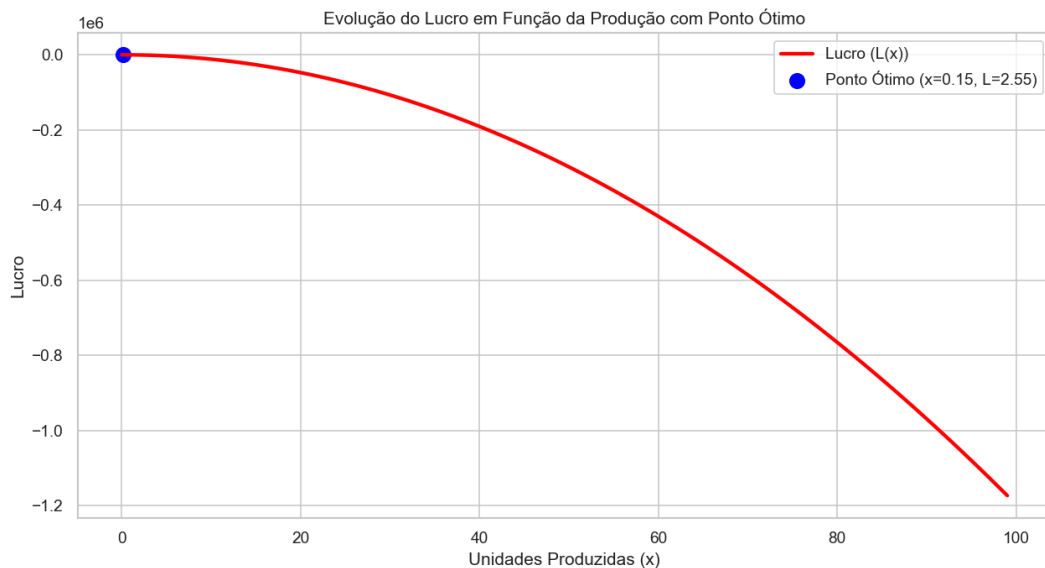
### 3. Usar bibliotecas como Matplotlib ou Seaborn para criar gráficos que mostrem a evolução do lucro em função da produção.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 # Funções para Custo, Receita e Lucro
6
7 1 usage
8 def custo(x):
9     return 120 * x**2 + 25 * x
10
11 1 usage
12 def receita(x):
13     return 60 * x
14
15 1 usage
16 def lucro(x):
17     return receita(x) - custo(x)
18
19 # Variando o número de unidades produzidas
20 x_values = np.arange(0, 100, 1)
21 lucro_values = lucro(x_values)
22
23 # Configuração do estilo do Seaborn
24 sns.set(style="whitegrid")
25
26 # Criando o gráfico para a evolução do lucro em função da produção
27 plt.figure(figsize=(12, 6))
28 sns.lineplot(x=x_values, y=lucro_values, color="red", linewidth=2.5, label="Lucro (L(x))")
29 plt.xlabel("Unidades Produzidas (x)")
30 plt.ylabel("Lucro")
31 plt.title("Evolução do Lucro em Função da Produção")
32 plt.legend()
33 plt.grid(True)
34 plt.show()
```



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

```
10 def receita(x):
11     return 60 * x
12
13 3 usages
14 def lucro(x):
15     return receita(x) - custo(x)
16
17 # Função negativa do lucro para maximizar usando fmin
18 1 usage
19 def lucro_neg(x):
20     return -lucro(x)
21
22 # Encontrando o ponto ótimo
23 x_otimo = fmin(lucro_neg, x0=1, disp=False)[0]
24 lucro_otimo = lucro(x_otimo)
25
26 # Variando o número de unidades produzidas
27 x_values = np.arange(0, 100, 1)
28 lucro_values = lucro(x_values)
29
30 # Plotando o gráfico com o ponto ótimo
31 sns.set(style="whitegrid")
32 plt.figure(figsize=(12, 6))
33 sns.lineplot(x=x_values, y=lucro_values, color="red", linewidth=2.5, label="Lucro (L(x))")
34 plt.scatter(x_otimo, lucro_otimo, color="blue", s=100, label=f'Ponto Ótimo (x={x_otimo:.2f}, L={lucro_otimo:.2f})')
35 plt.xlabel("Unidades Produzidas (x)")
36 plt.ylabel("Lucro")
37 plt.title("Evolução do Lucro em Função da Produção com Ponto Ótimo")
38 plt.legend()
39 plt.grid(True)
40 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:

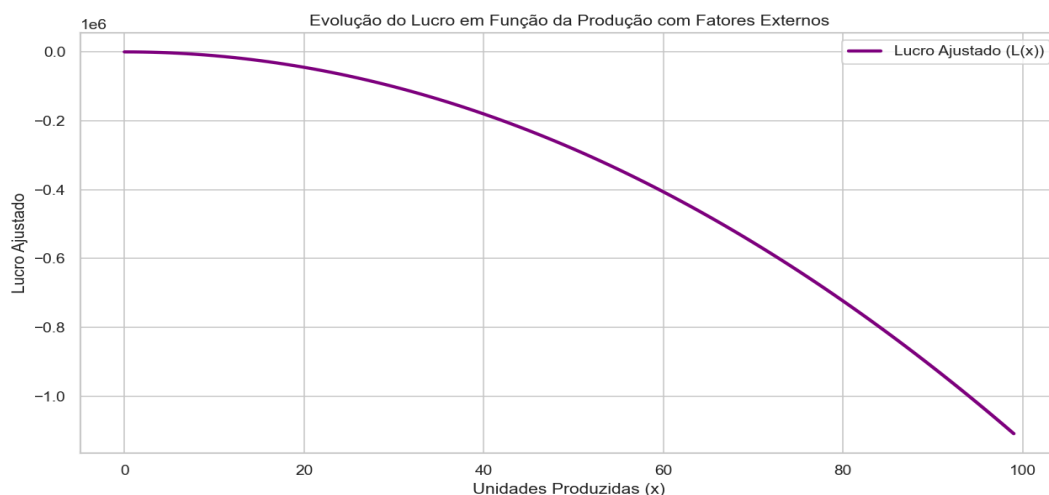
$$C_{ajustado}(x) = C(x) \cdot (1 + m / 100)$$

### 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.



```

6 taxa_imposto = 10 # em porcentagem sobre o lucro
7 aumento_materia_prima = 5 # em porcentagem de aumento no custo das matérias-primas
8
9 # Funções de Custo, Receita e Lucro ajustadas
10 1 usage
11 def custo_ajustado(x):
12     return (120 * x**2 + 25 * x) * (1 + aumento_materia_prima / 100)
13
14 1 usage
15 def receita(x):
16     return 60 * x
17
18 1 usage
19 def lucro_ajustado(x):
20     lucro_bruto = receita(x) - custo_ajustado(x)
21     return lucro_bruto * (1 - taxa_imposto / 100)
22
23 # Variando o número de unidades produzidas
24 x_values = np.arange(0, 100, 1)
25 lucro_values = lucro_ajustado(x_values)
26
27 # Plotando o gráfico com as novas variáveis
28 sns.set(style="whitegrid")
29 plt.figure(figsize=(12, 6))
30 sns.lineplot(x=x_values, y=lucro_values, color="purple", linewidth=2.5, label="Lucro Ajustado (L(x))")
31 plt.xlabel("Unidades Produzidas (x)")
32 plt.ylabel("Lucro Ajustado")
33 plt.title("Evolução do Lucro em Função da Produção com Fatores Externos")
34 plt.legend()
35 plt.grid(True)
36 plt.show()

```

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