Pontos Interiores vs Simplex

Pedro Loes e Felipe Sadock

20/06/2021

O método de otimização Simplex possui a característica de encontrar a solução exata em problemas de programação linear, porém, em problemas de grande dimensão, o algoritmo apresenta alto custo computacional para atingir o máximo global. O método de optimização Pontos Interiores apresenta baixo custo computacional em problemas de grande dimensão, mas apresenta resultados que apenas aproximam a solução ótima. O método Simplex Revisado apresenta baixo custo computacional em relação ao Simplex tradicional mas pode sofrer problemas de degeneração. O objetivo deste projeto foi comparar o funcionamento destes 3 algoritmos por meio de simulações em diversos cenários para identificar suas vantagens e desvantagens indicando as situações em que devem ser preferidos.

Introdução

- O projeto foi dividido em 5 etapas que objetivam explorar a performance, número de iterações e custo computacional dos algoritmos de otimização Simplex, Simplex Revisado e Pontos Interiores, da biblioteca scipy.linprog.
- 0. Simulação
  - O desenho da simulação e suas particularides foram descritos e o seu funcionamento foi ilustrado.
- 1. Número de Iterações
  - A primeira comparação consistiu em ilustrar a peformance por meio de um mapa de calor para avaliar o número de iterações utilizadas por cada algoritmo.
- 2. Diferença de Máximos
  - A segunda comparação consistiu em ilustrar a eficácia por meio de uma mapa de calor para avaliar a diferença de máximos do algoritmo Pontos Interiores em comparação com os algoritmos Simplex e Simplex revisado.
- 3. Custo Computacional
  - A terceira comparação consistiu em ilustrar a eficiencia por meio de uma mapa de calor para avaliar o custo computacional de cada algoritmo.
- 4. Conclusões
  - Finalmente foram apresentadas sugestões para uso dos algoritmos em função dos cenários.

0. Simulações

- A função optimizar foi definida para amostrar o problema com **V** variáveis, $V \in [2, 1000]$ e **R** restrições, $R \in [1, 1000]$, definir limites das variáveis, executar os 3 métodos de optimização e calcular o custo computacional. Nove matrizes de dimensão $M_{1000,1000}$ foram geradas para armazenar os resultados das 500000 simulações. Dois laços foram utilizados para combinar variáveis e restrições. Para padronizar o experimento foi adotado o critério de número de restrições menor ou igual ao número de variáveis. Por exemplo, simulações com 2 variáveis e 1000 restrições não foram consideradas devido à redundância de cortes no politopo.

- Passos da Simulação:

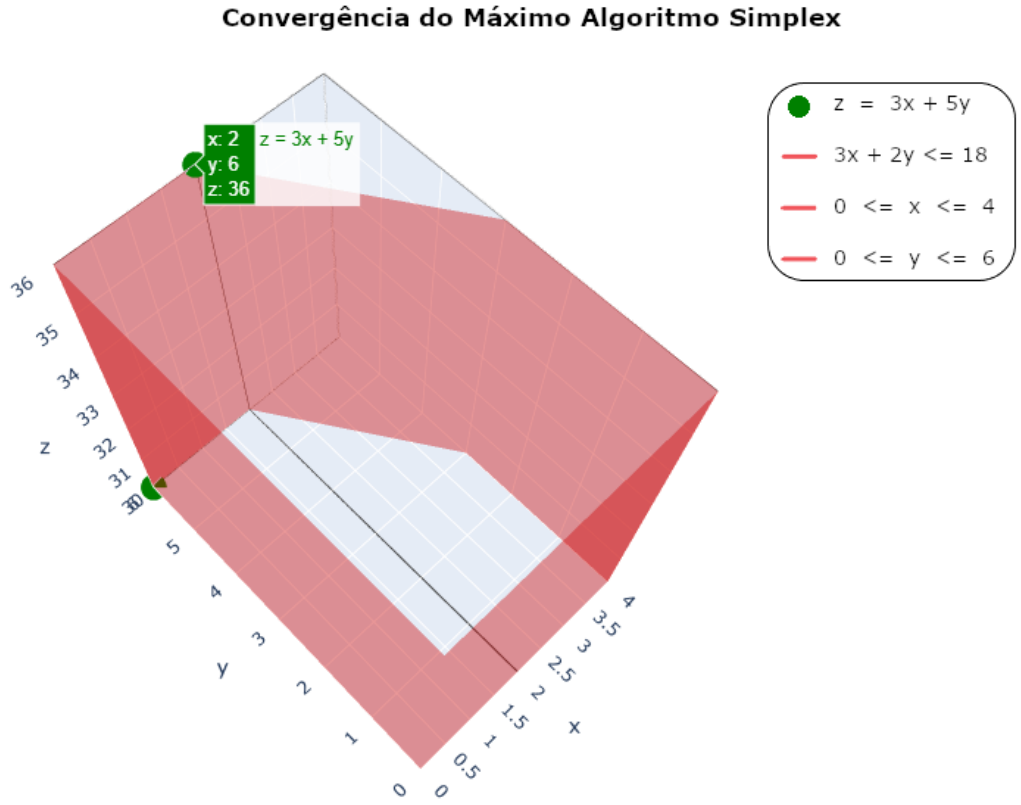
  1. Sorteio das amostras em intervalos de inteiros aleatórios
  2. Definição dos limites das variáveis
  3. Executa as 3 otimizações:
  4. Os resultados das simulações são recuperados em matrizes $M_{(V,R)}$ com:
  5. Laço para combinar dimensões dos problemas:

```
# Define função optimizar
def optimizar(V, R):
  A = np.random.randint(1000, size=(R * V)).reshape((R, V))
  b = np.random.randint(1000, 10000, size=(R))
  c = np.random.randint(1, 1000, size=(V)) * -1
  limites_x = [(0, superior) for superior in np.random.randint(1, 100, size=(V))]

  inicio_s = time.time()
  s = linprog(c, A_ub=A, b_ub=b, bounds=(limites_x),  method='simplex')
  fim_s = time.time()
  s_t = fim_s - inicio_s
  inicio_rs = time.time()
  rs = linprog(c, A_ub=A, b_ub=b, bounds=(limites_x),  method='revised simplex')
  fim_rs = time.time()
  rs_t = fim_rs - inicio_rs
  inicio_pi = time.time()
  pi = linprog(c, A_ub=A, b_ub=b, bounds=(limites_x),  method='interior-point')
  fim_pi = time.time()
  pi_t = fim_pi - inicio_pi

  return(s.nit, -1 * s.fun, s_t, pi.nit, -1 * pi.fun, pi_t, rs.nit, rs.fun, rs_t)

# Declara matrizes de resultados
pi_i = np.zeros(shape=(V, V)); pi_f = np.zeros(shape=(V, V)); pi_c = np.zeros(shape=(V, V))
s_i = np.zeros(shape = (V, V)); s_f = np.zeros(shape = (V, V)); s_c = np.zeros(shape = (V, V))
rs_i = np.zeros(shape = (V, V)); rs_f = np.zeros(shape = (V, V)); rs_c = np.zeros(shape = (V, V))

# Declara número de variáveis para simulação
V = 1000

# Laço para simulação de variáveis e restrições
for v in np.arange(2, V):
  r = 1
  while(r <= v):
    s_i[v, r], s_f[v, r], s_c[v, r], pi_i[v, r], pi_f[v, r], \
    pi_c[v, r], rs_i[v, r], rs_f[v, r], rs_c[v, r] = optimizar(v, r)
    r += 1
```

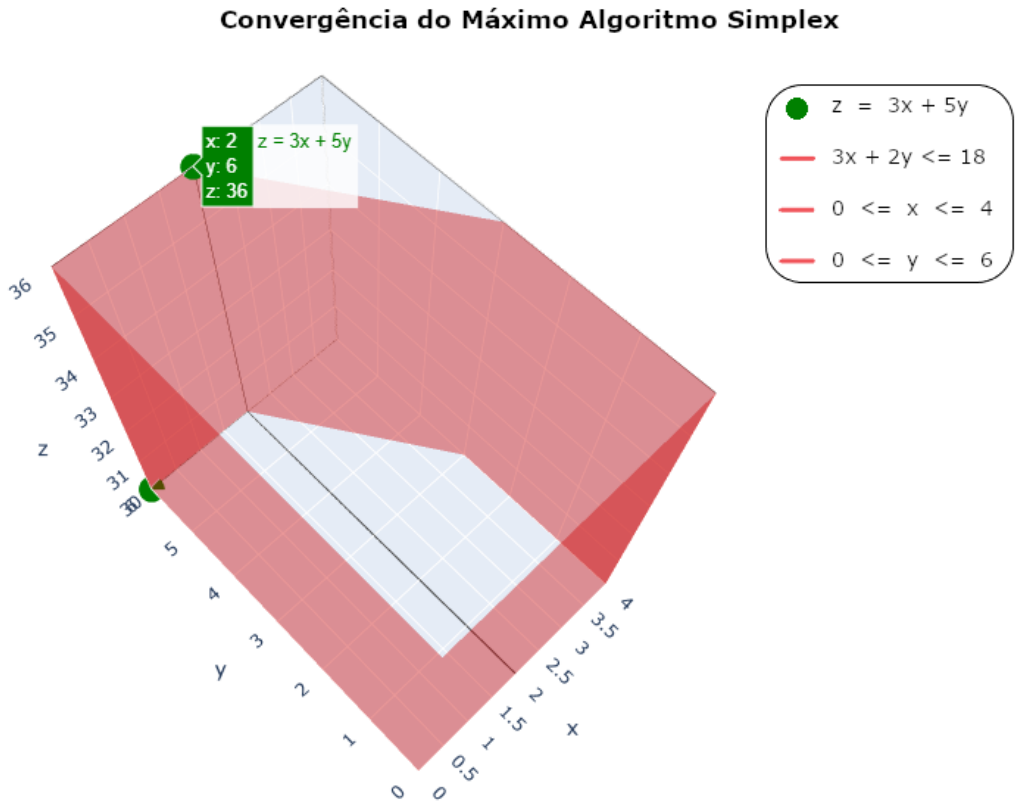
1.Número de Iterações

- Os números de iterações gastas por cada algoritmo para cada cenário foram calculados e os valores foram armazenados em matrizes $M_{V,R}$ para produção de mapas de calor que facilitassem a análise de padrões.
- Os mapas de calor foram desenhados com o número de variáveis no eixo horizontal e o número de restrições no eixo vertical.
- Uma legenda com a escala gradiente nas cores azul, cinza e vermelho foi colocada abaixo do eixo vertical representado o eixo da terceira dimensão do gráfico que exibe a medida do número de iterações.
- O fundo do gráfico foi colorido com a cor preta para representar os cenários não simulados devido a padronização escolhida.

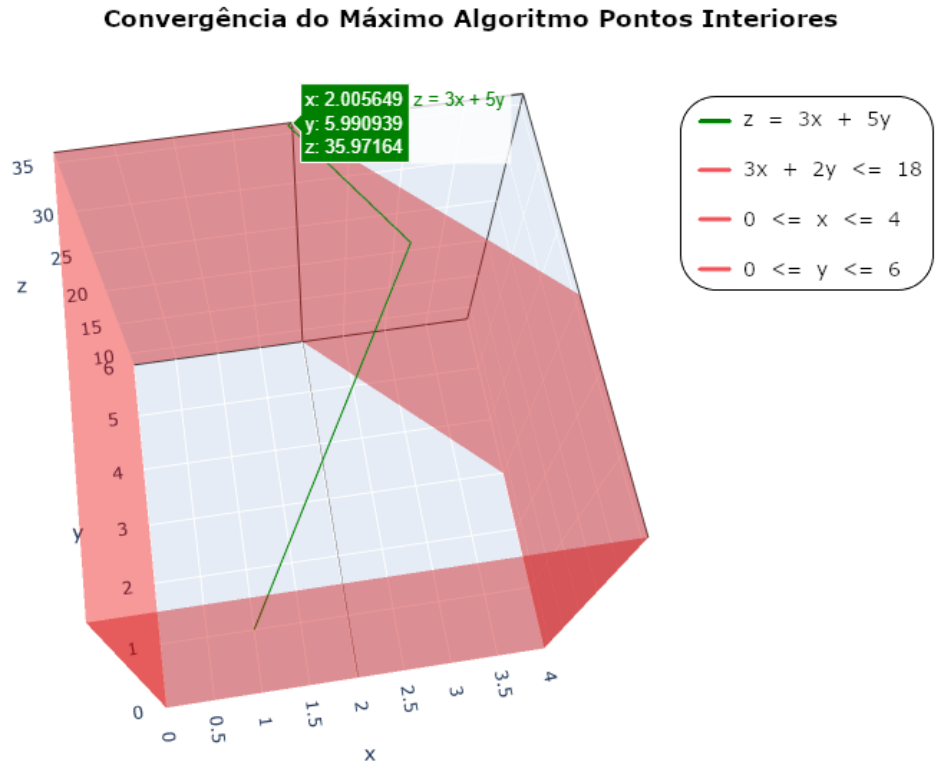
Método Simplex



Método Simplex Revisado



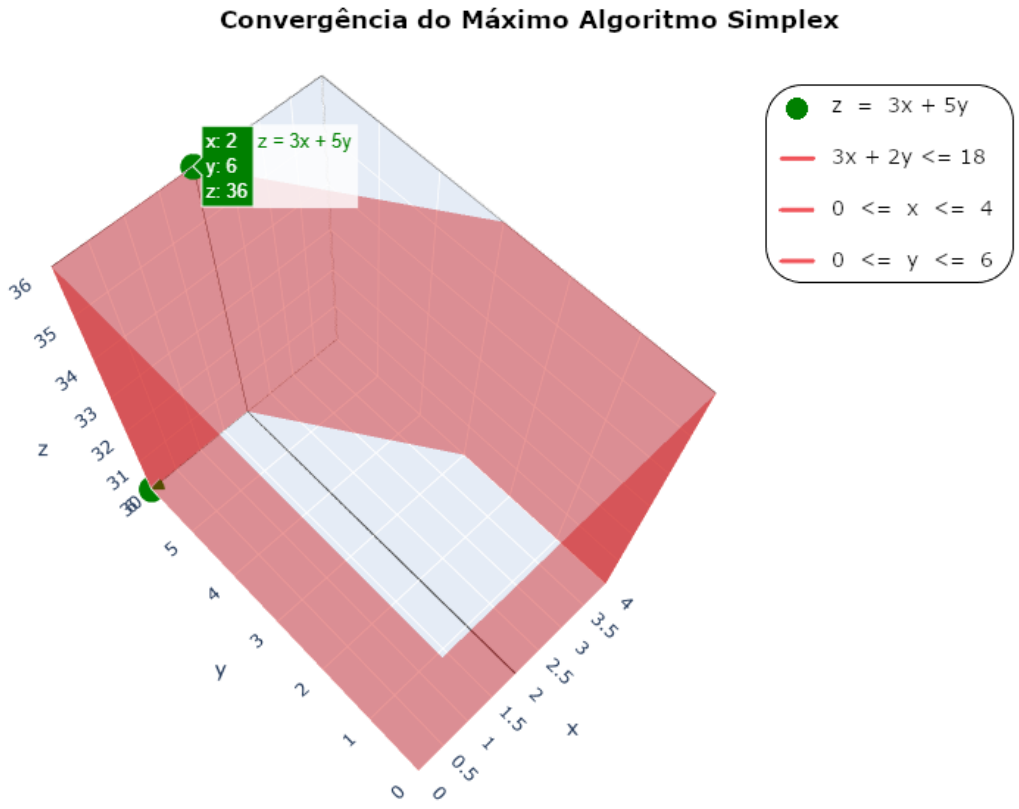
Método Pontos Interiores



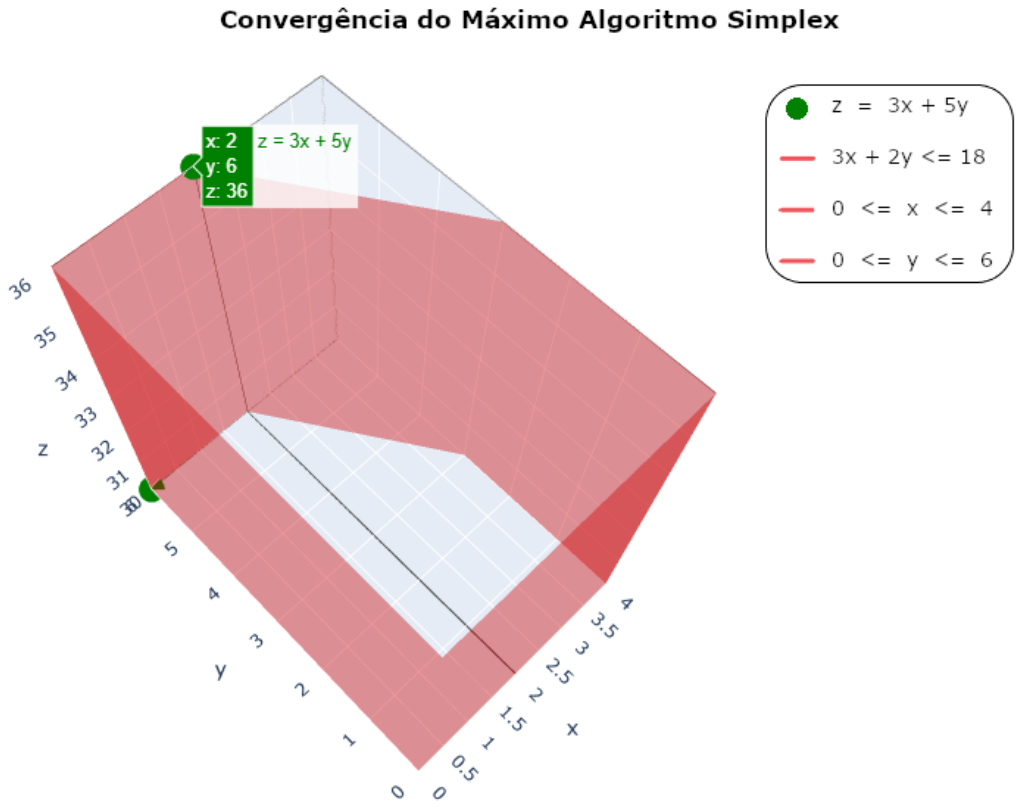
2. Diferença de Máximos

- A diferença entre os máximos de cada algoritmo para cada cenário foram calculados e os valores foram armazenados em matrizes $M_{(V,R)}$ para produção de mapas de calor que facilitassem a análise de padrões.
- Os mapas de calor foram desenhados com o número de variáveis no eixo horizontal e o número de restrições no eixo vertical.
- Uma legenda com a escala gradiente nas cores azul, cinza e vermelho foi colocada abaixo do eixo vertical representado o eixo da terceira dimensão do gráfico que exibe a diferença de máximos na escala logarítmica.
- O fundo do gráfico foi colorido com a cor preta para representar os cenários não simulados devido a padronização escolhida.

Método Simplex vs Pontos Interiores



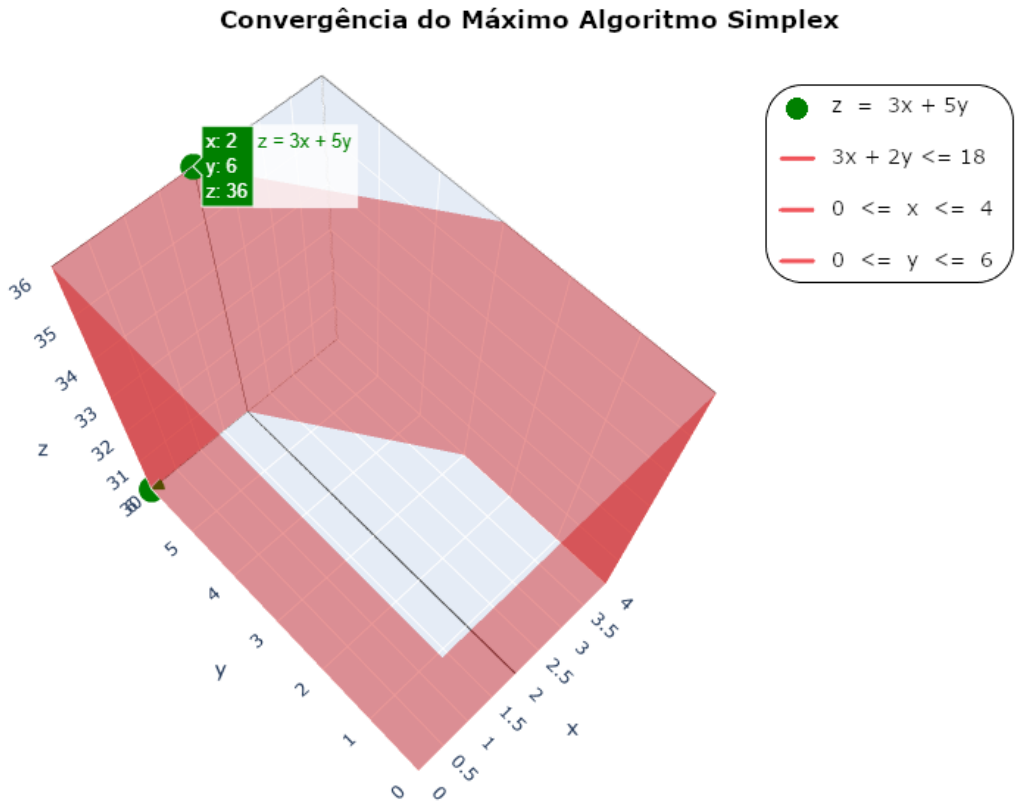
Método Simplex Revisado vs Pontos Interiores



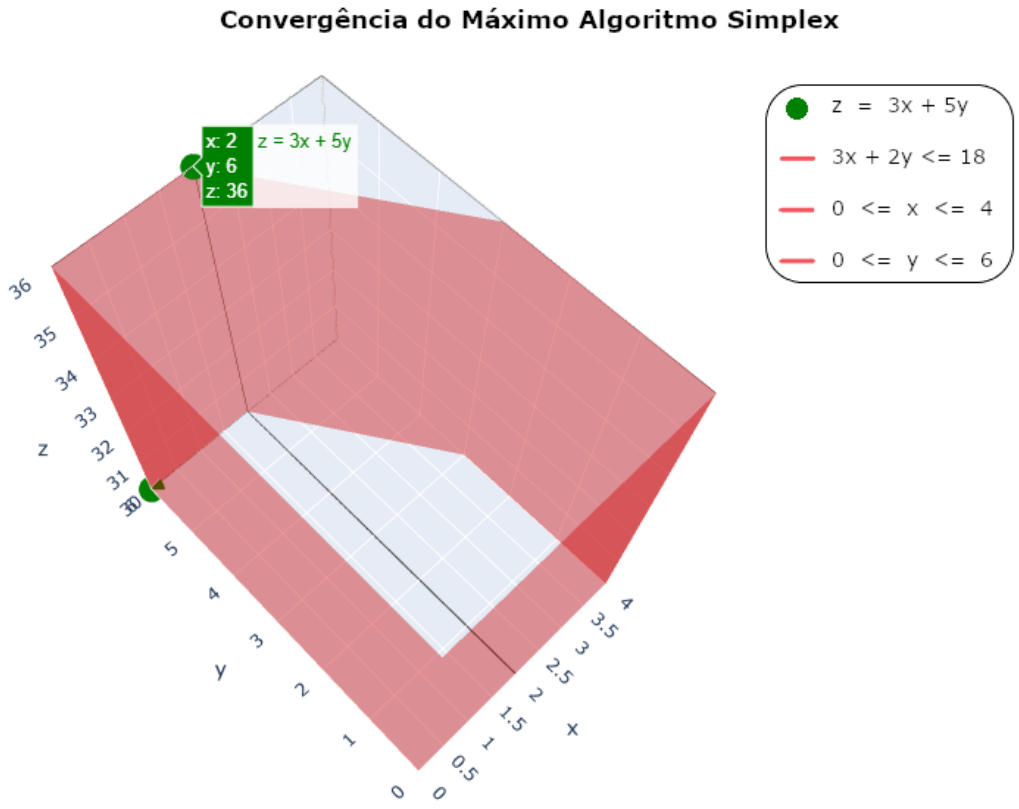
3. Custo Computacional

- A diferença entre os máximos de gastas por cada algoritmo para cada cenário foram calculados e os valores foram armazenados em matrizes $M_{(V,R)}$ para produção de mapas de calor que facilitassem a análise de padrões.
- Os mapas de calor foram desenhados com o número de variáveis no eixo horizontal e o número de restrições no eixo vertical.
- Uma legenda com a escala gradiente nas cores azul, cinza e vermelho foi colocada abaixo do eixo vertical representado o eixo da terceira dimensão do gráfico que exibe a escala o custo computacional em segundos.
- O fundo do gráfico foi colorido com a cor preta para representar os cenários não simulados devido a padronização escolhida.

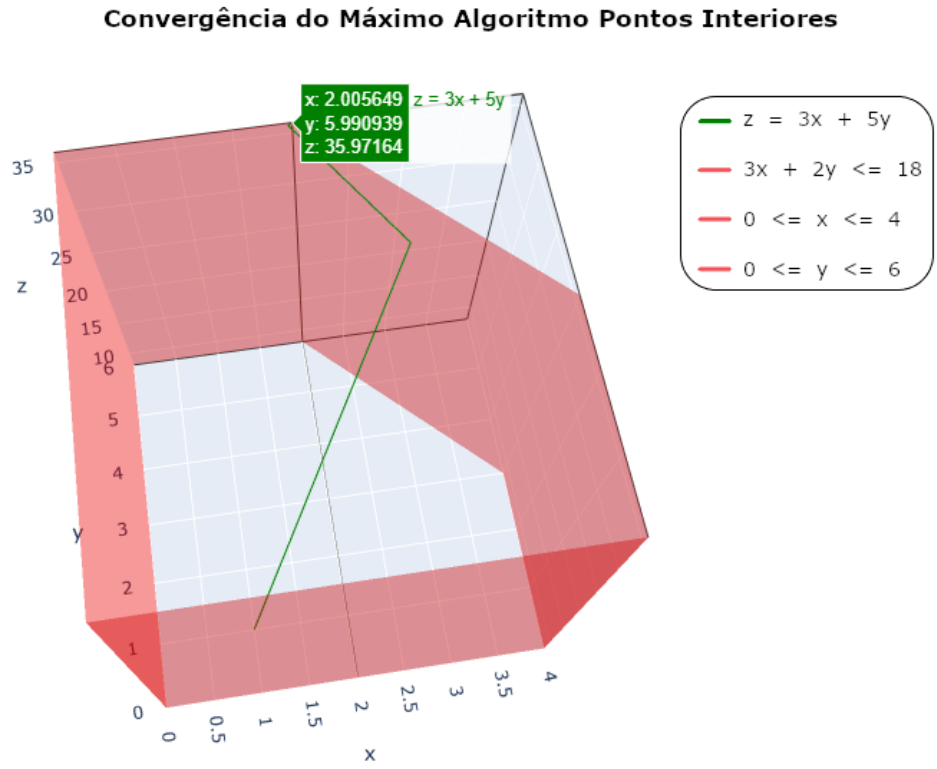
Método Simplex



Método Simplex Revisado



Método Pontos Interiores



4. Conclusões

5. Referências

- Dantzig, George Bernard, "The Simplex Method" Santa Monica, CA: RAND Corporation, 1956.
- A. Vannelli, "Teaching large-scale optimization by an interior point approach" in IEEE Transactions on Education, vol. 36, no. 1, pp. 204-209, Feb. 1993, doi: 10.1109/13.204847.