Lista 6 - Igor Patrício Michels

November 15, 2021

1 Item a

Implementado no arquivo `btcs.py`.

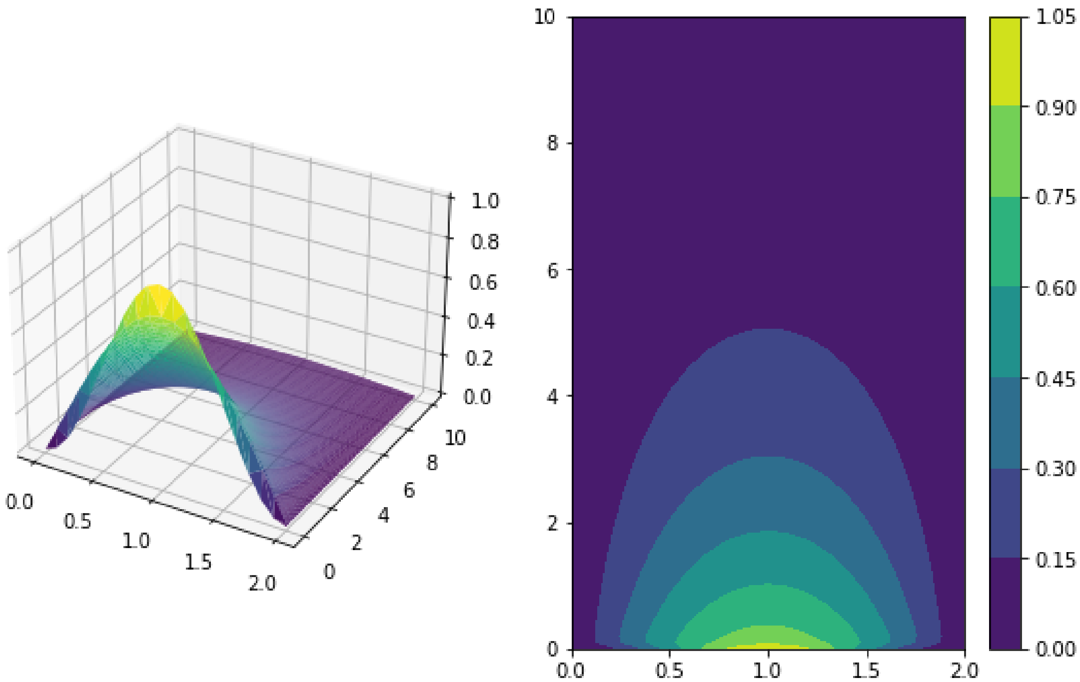
[1]:
```
from btcs import *
```

2 Item b

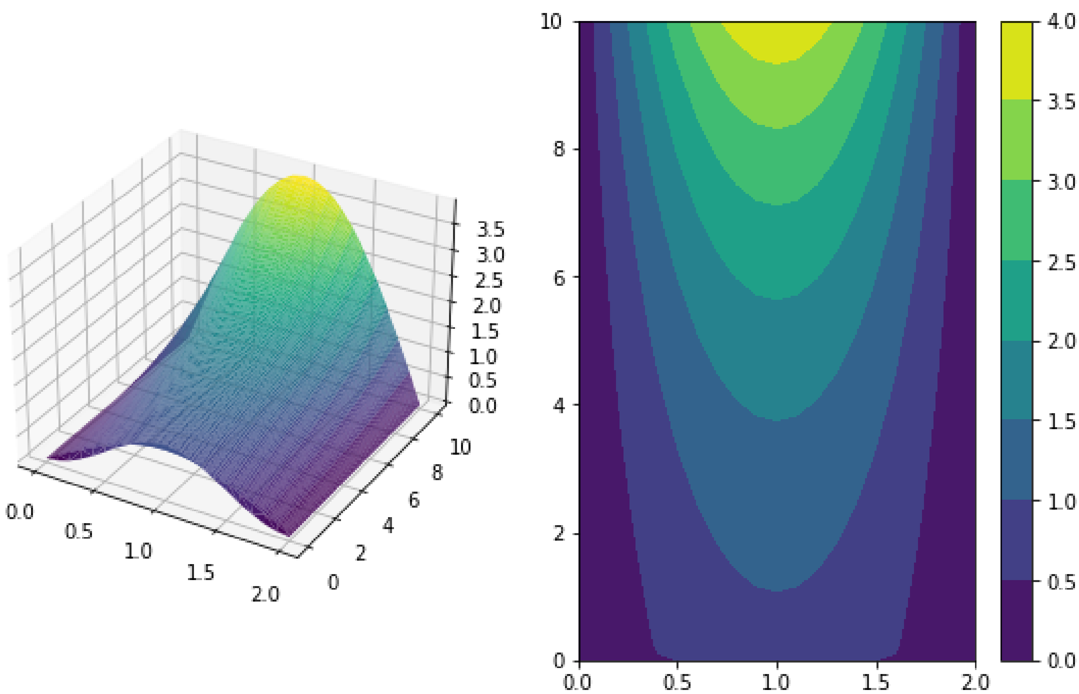
Usando como parâmetros $L = 2$ e $c = 4$, podemos manter a comparação com os valores de d como 9.5 e 10, pois a razão $\frac{c}{L^2}$ se mantém igual a 1, além de garantirmos continuidade na solução:

[2]:
```
c = 4
L = 2
T = 10
dt = 0.1
dx = 0.1

print('A população tende a se extinguir para d = ', d := 9.5, ':', sep = '')
btcs(c, d, L, T, dt, dx, f, g, h)

print('A população tende a aumentar para d = ', d := 10, ':', sep = '')
btcs(c, d, L, T, dt, dx, f, g, h)
```

```
A população tende a se extinguir para d = 9.5:
```



A população tende a aumentar para d = 10:



3 Item c

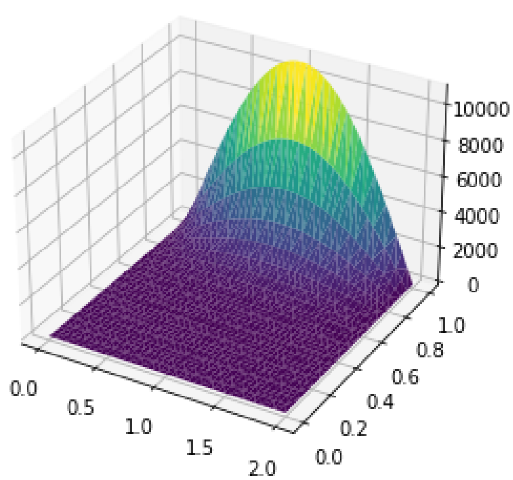
Vamos dar uma chutada no balde para a relação entre d e a razão $\frac{c}{L^2}$:

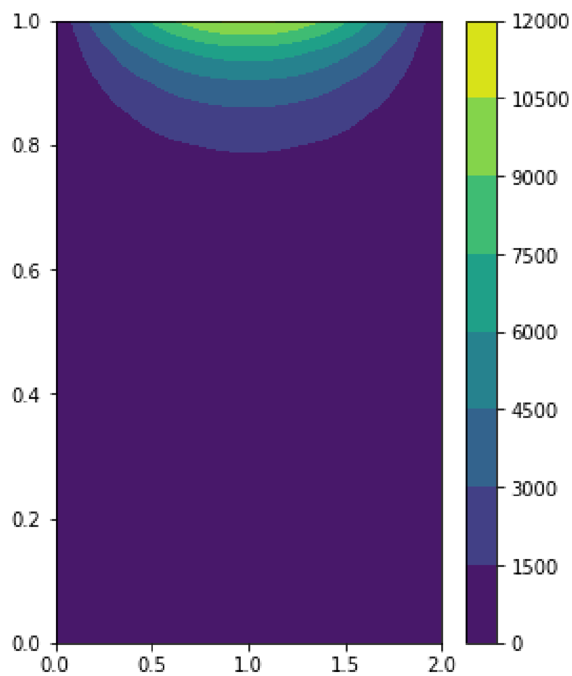
[3]:
```
c = 1
d = 10
L = 2
T = 1
dx = 0.05
```

Agora vamos plotar os gráficos para diferentes valores de Δt e Δx:

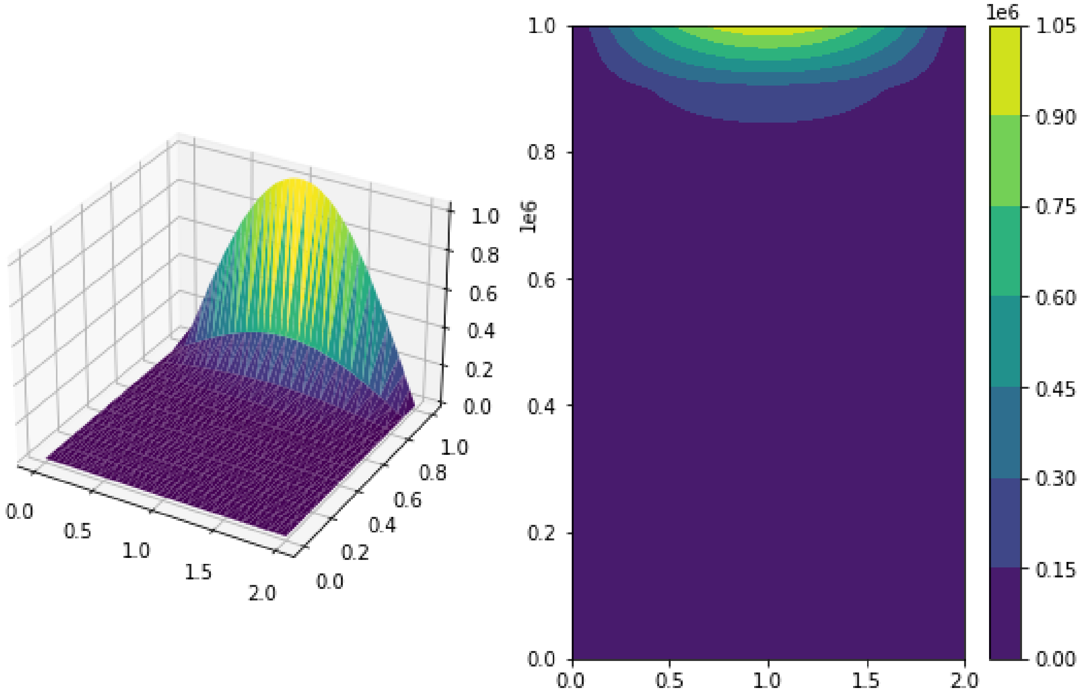
[4]:
```
for dt in [0.05, 0.1, 0.2]:
    print(f'Simulação com dt = {dt} e dx = {dx}:')
    btcs(c, d, L, T, dt, dx, f, g, h)
```

```
Simulação com dt = 0.05 e dx = 0.05:
```

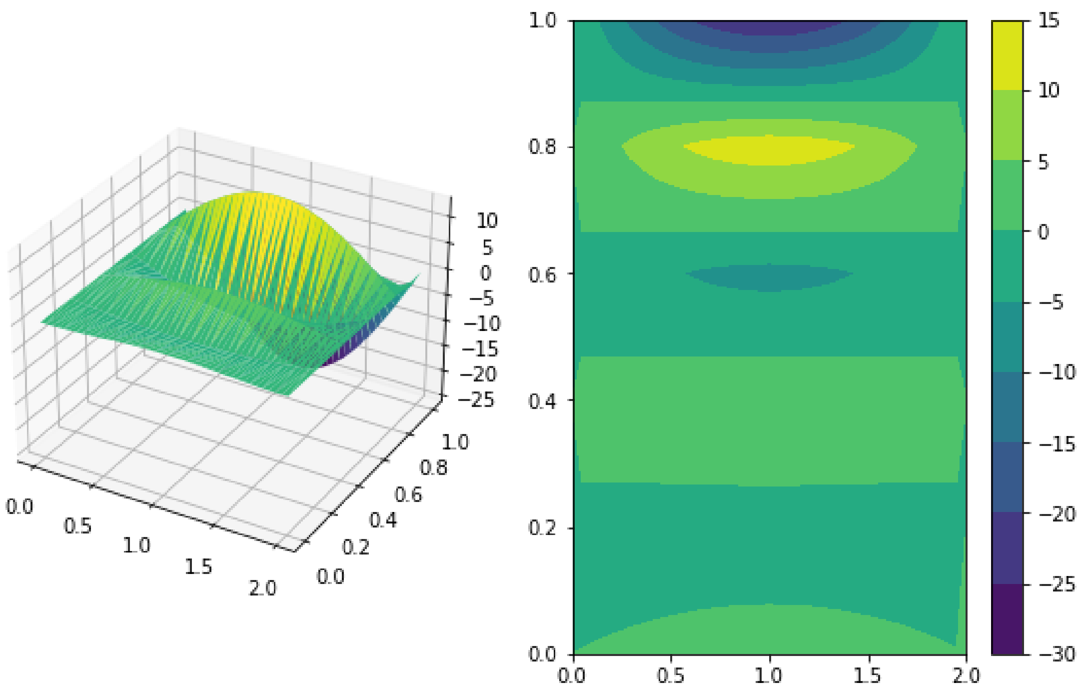





```
Simulação com dt = 0.1 e dx = 0.05:
```



`Simulação com dt = 0.2 e dx = 0.05:`



Podemos ver que, variando Δt, o eixo z tem sua escala alterada, ou seja, temos que os valores do passo acabam influenciando nos resultados computacionais, o que, para os dois primeiros valores, pode ser melhor percebido no mapa de calor, à direita, onde o maior passo faz com que as curvas fiquem mais estranhas, bem como as faixas fiquem mais estreitas e surgindo mais cedo, o que mostra que os valores crescem mais rapidamente. Além disso, um grande passo faz com que valores negativos apareçam, o que não faz sentido para o modelo.

Uma ideia intuitiva para essa mudança nos resultados em função da alteração do Δt é que agora nossa matriz continua sendo tridiagonal, mas tem diagonais dadas por $1 + 2 \cdot v - d \cdot \Delta t$, com $v = \dfrac{c \cdot \Delta t}{(\Delta x)^2}$, enquanto as demais entradas entradas da matriz são 0 ou $-v$, assim, a matriz tem diagonal estritamente dominante se, e somente se, $\dfrac{1}{d} > \Delta t$. Dessa forma, como aqui temos $d = 10$, o método funciona bem para $\Delta t = 0.05$ e, para $\Delta t = 0.1$, a matriz tem a primeira e a última linha com diagonal estritamente dominante e ocorre a igualdade nas demais linhas, o que também garante a convergência, por isso os resultados, apesar de um pouco distintos, ainda são razoáveis. Já para $\Delta t = 0.2$ nossa condição não é satisfeita, o que explica a não convergência e a aparição de valores negativos.

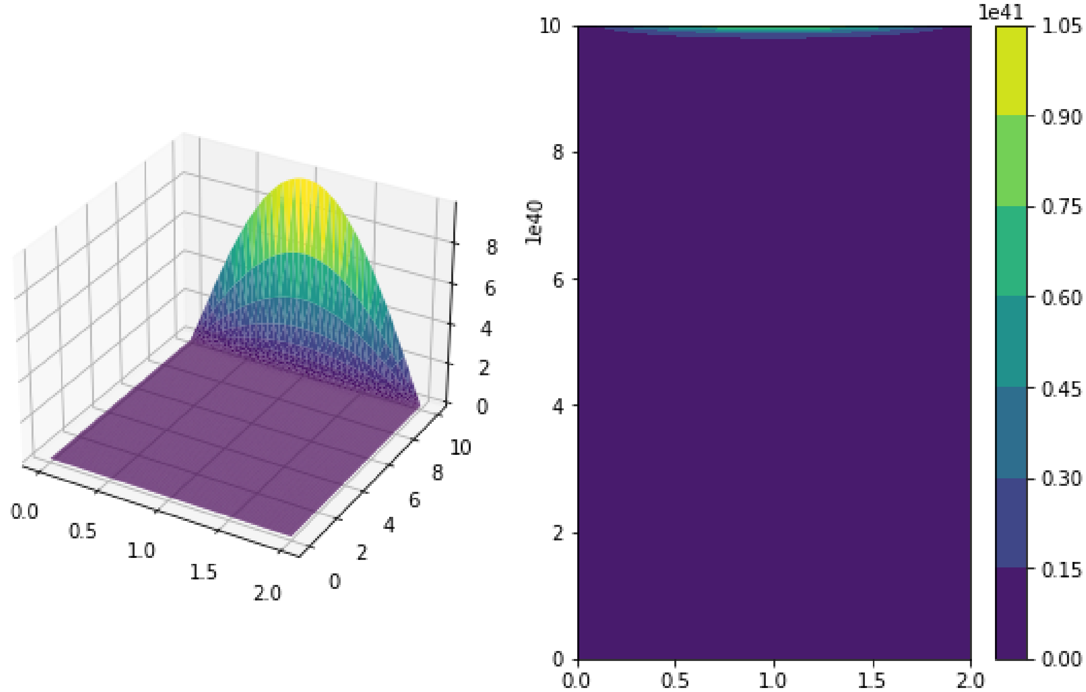
Agora vamos ver o que ocorre com um maior passo no eixo x. Para isso, vamos redefinir os parâmetros:

[5]:
```
c = 1
d = 10
L = 2
T = 10
dt = 0.05
```

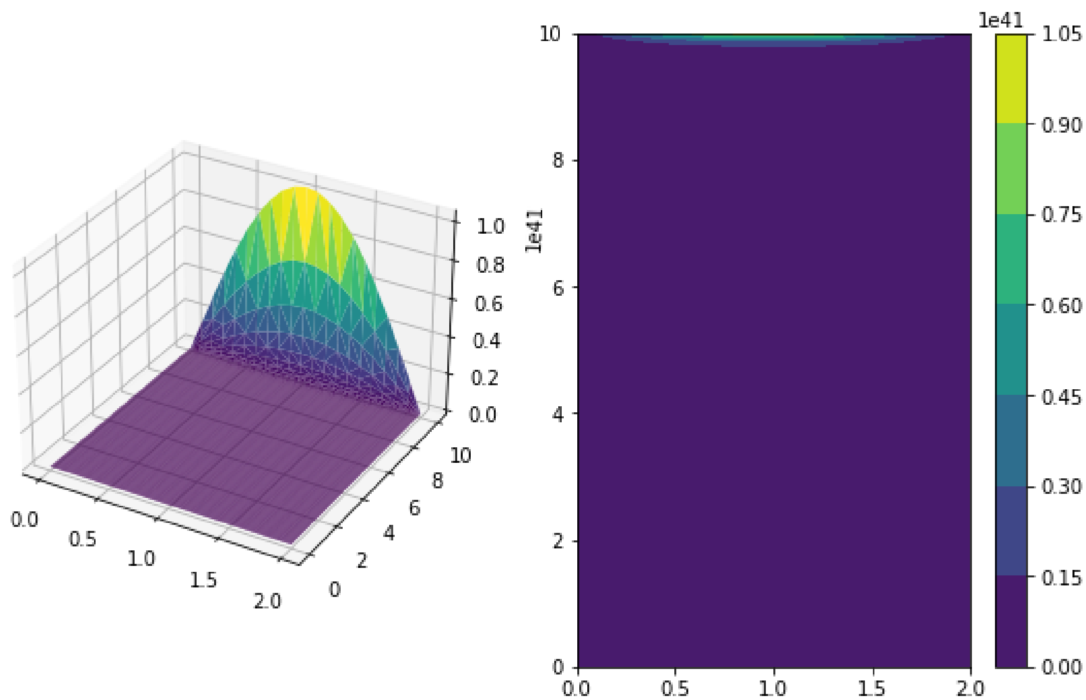
E agora vamos testar para diferentes valores para Δx:

[6]:
```
for dx in [0.05, 0.1, 0.2]:
    print(f'Simulação com dt = {dt} e dx = {dx}:')
    btcs(c, d, L, T, dt, dx, f, g, h)
```

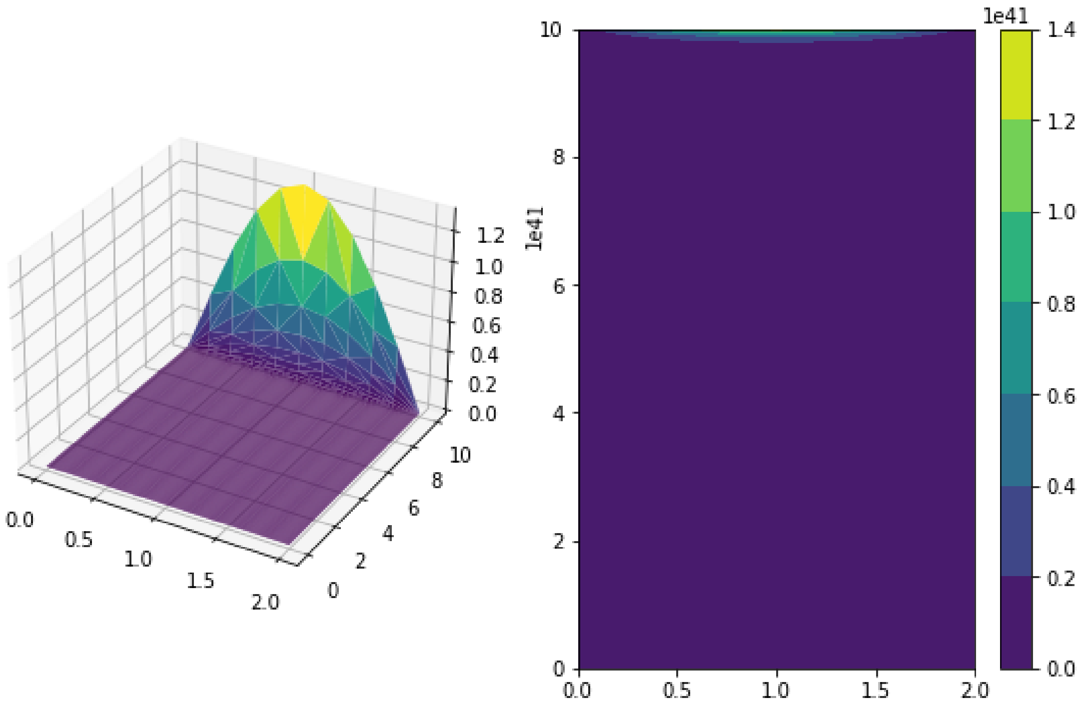
```
Simulação com dt = 0.05 e dx = 0.05:
```



`Simulação com dt = 0.05 e dx = 0.1:`



```
Simulação com dt = 0.05 e dx = 0.2:
```



Como aumentamos o tempo, agora é mais fácil olharmos para o plot 3D, em especial para a escala do eixo z, a qual muda do primeiro para o segundo plot. Notamos que os valores aumentam, bem como o mapa de calor do segundo para o terceiro plot, ou seja, os valores de Δx também fazem com que os resultados computacionais variem.

4 Item d

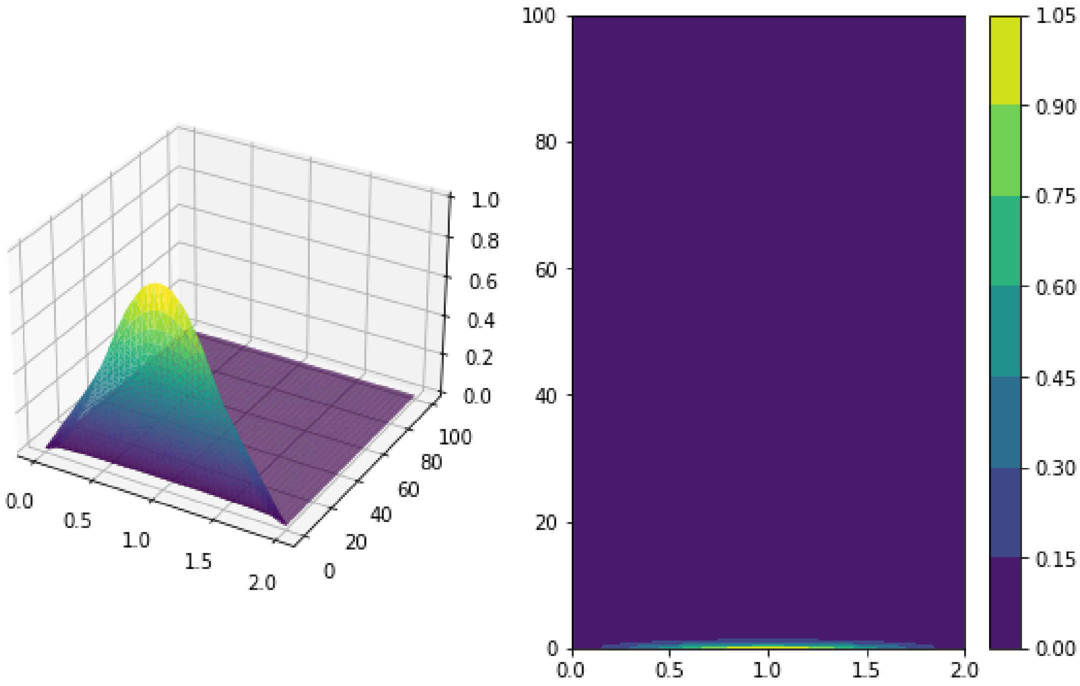
Primeiramente, vamos definir alguns parâmetros para as simulações:

[7]:
```
c = 1
d = 1
T = 100
dt = 0.05
dx = 0.05
```

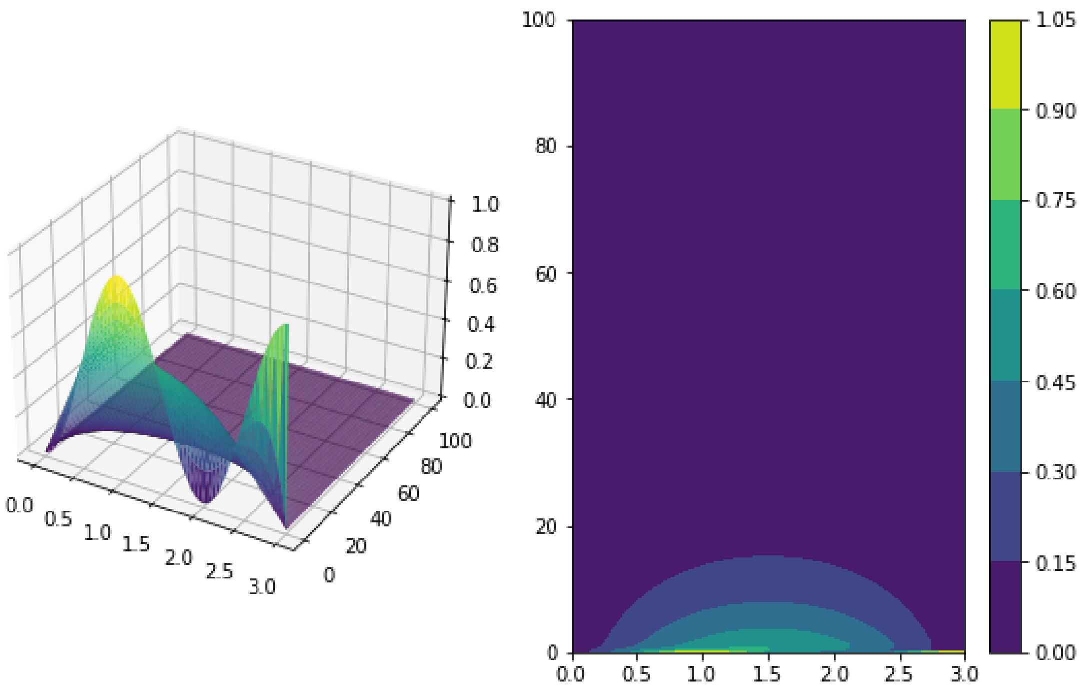
Agora vamos testar alguns valores de L, usando que, pela teoria, a população deve sobreviver quando $L > \pi$ e ir para extinção quando $L < \pi$. Assim, vamos testar alguns valores próximos, sem nos importar muito com a continuidade da solução (problema causado pela inconsistência da condição inicial com as condições de fronteira para o ponto $(L, 0)$):

[8]:
```
for L in [2, 3, 3.1, 3.14, np.pi, 3.15, 3.2, 4]:
    print(f'Simulação com L = {L}:')
    btcs(c, d, L, T, dt, dx, f, g, h)
```

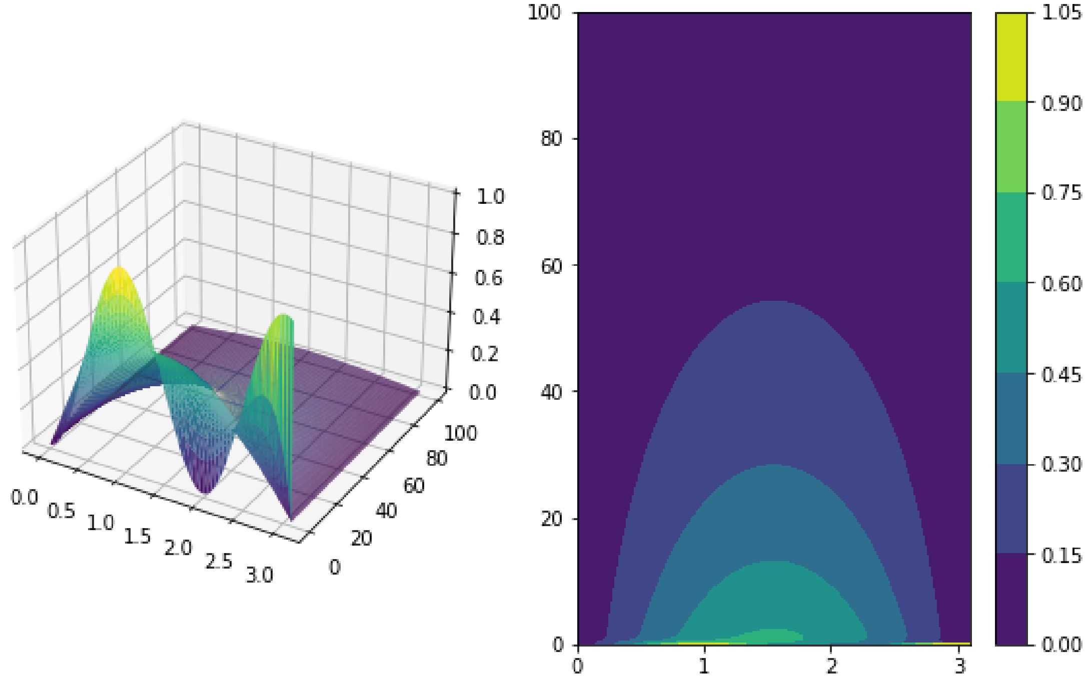
Simulação com L = 2:



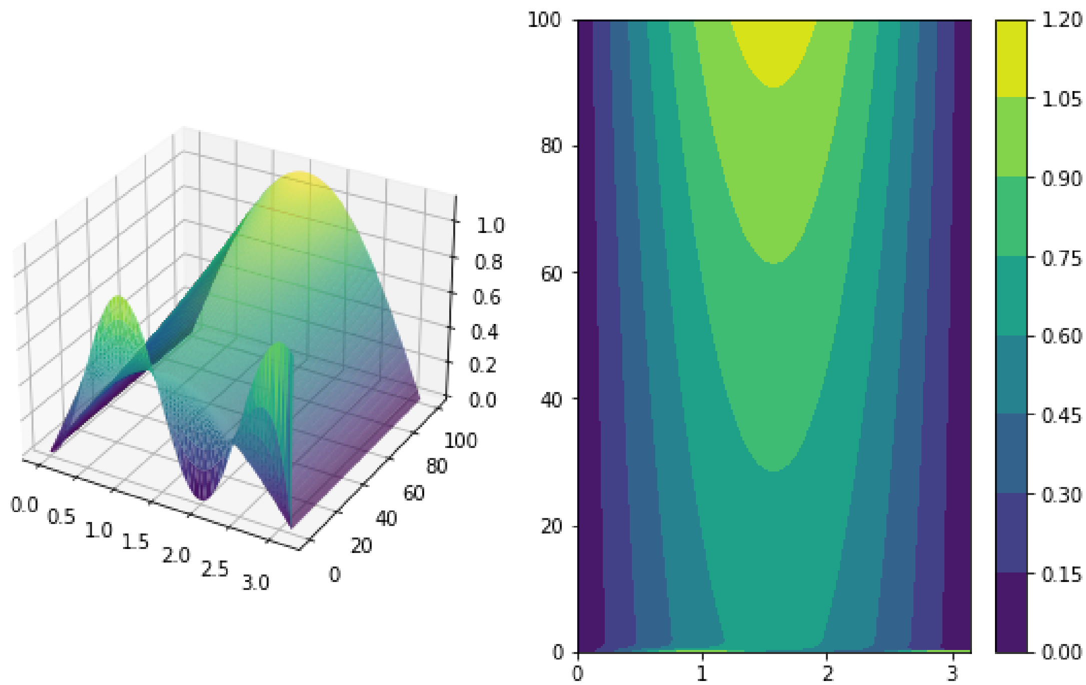
Simulação com L = 3:



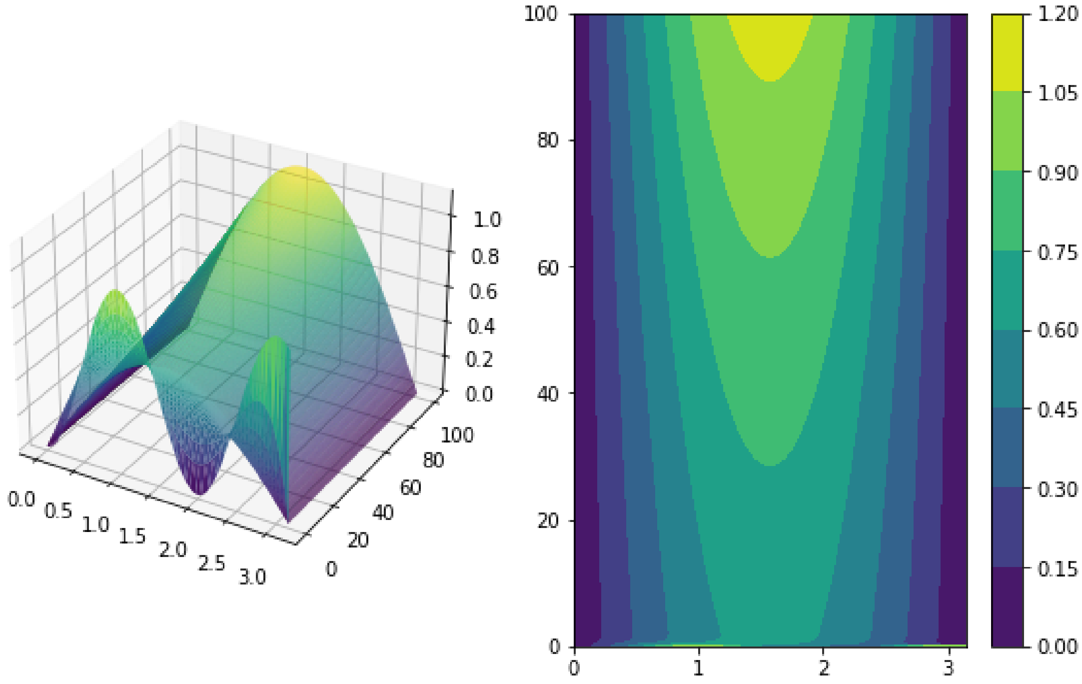
Simulação com L = 3.1:



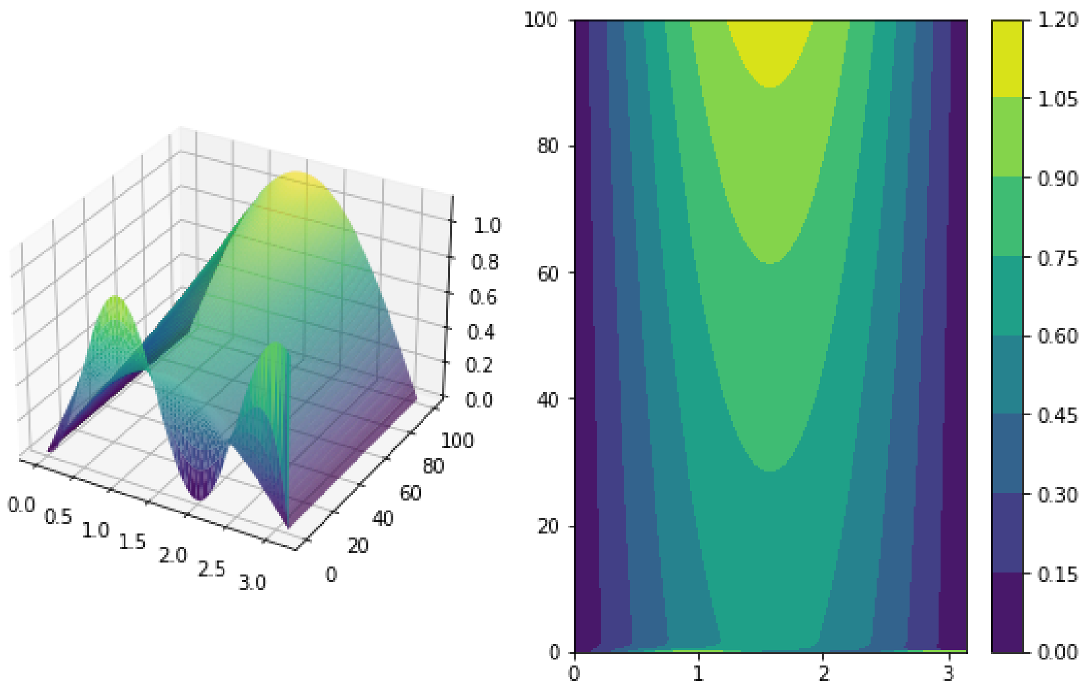
Simulação com L = 3.14:



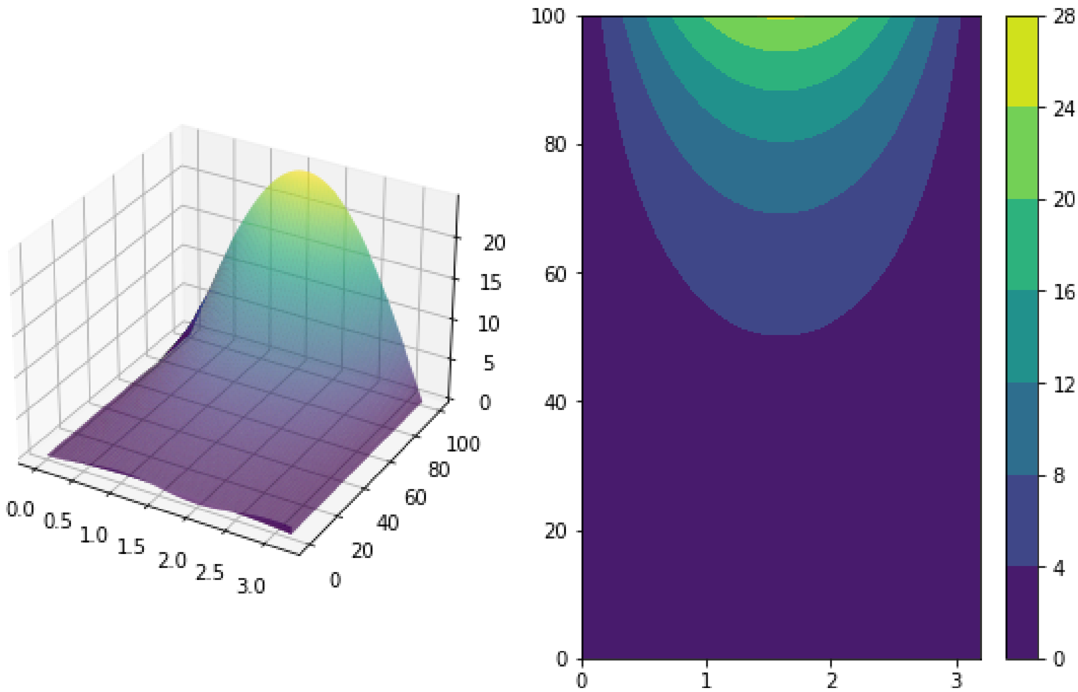
Simulação com L = 3.141592653589793:



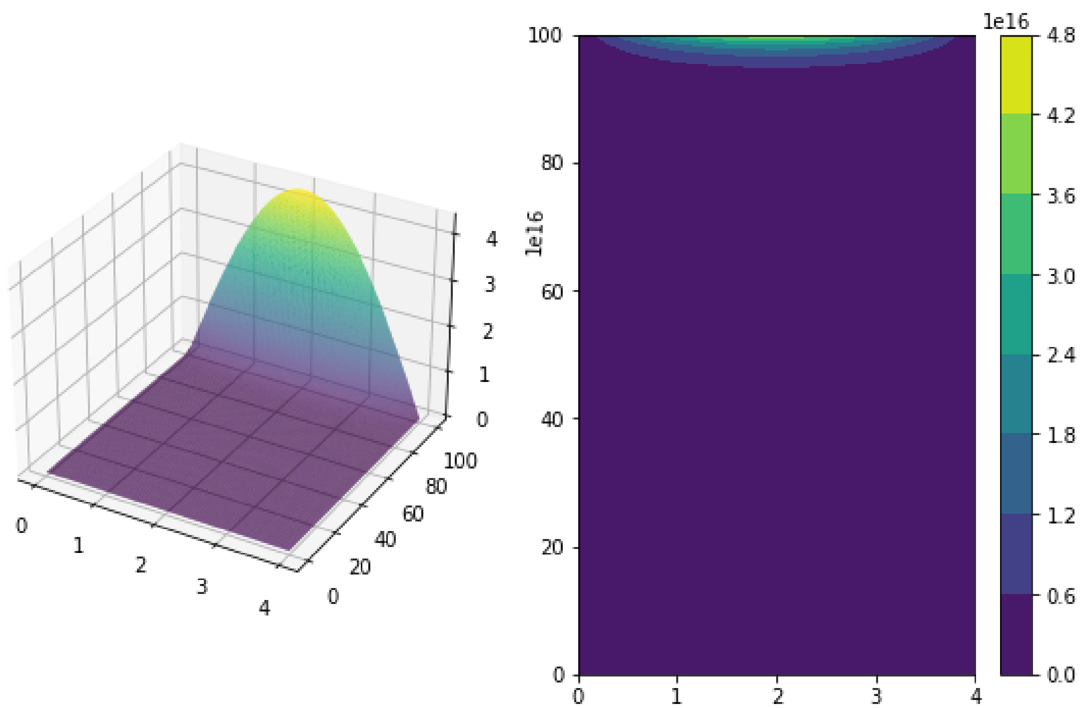
Simulação com L = 3.15:



Simulação com L = 3.2:



Simulação com L = 4:



Notamos que o valor mínimo para L de modo que a população não fique extinta está bem próximo de π, com a população indo a extinção até quando $L = 3.1$ e aumentando nos demais casos, conforme sugere o resultado teórico do item b.

OBS: o crescimento da população para quando $L = 3.14 < \pi$ deve estar relacionado a erros numéricos da aritmética do computador.