Modelo de Oferta e Demanda

Ian Teixeira Barreiro

Outubro 2021

1 Introdução

No contexto do estudo da economia, a fim de compreender o fenômeno econômico, a ferramenta mais utilizada são os modelos econômicos. Modelos econômicos, geralmente representados por meio de um conjunto de equações, são uma simplificação da realidade que, apesar de não captarem toda a complexidade do objeto estudado, auxiliam na compreensão de suas características essenciais.

Nesse sentido, o modelo mais básico, tradicionalmente o primeiro a ser apresentado aos estudantes de economia, é o modelo de oferta e demanda. Nesse modelo, dado um determinado mercado (que nós tomaremos como perfeitamente competitivo) com consumidores (cujo comportamento é bem modelado por uma função demanda) e empresas (cujo comportamento é bem modelado por uma função oferta) buscamos o ponto de equilíbrio (onde as funções se cruzam) que contenha a informação do preço e da quantidade de equilíbrio nesse mercado.

No presente texto apresentaremos um resumo desse modelo, junto com uma implementação computacional em linguagem Python que permita o cálculo do equilíbrio do mercado através de técnicas de ponto fixo. Para a descrição do modelo, tomaremos como referência o livro-texto "Microeconomia: Uma abordagem moderna"de Hal Varian (2012) e a seguinte estrutura: i) uma descrição da demanda de mercado; ii) uma descrição da oferta de mercado; iii) uma descrição do equilíbrio de mercado, junto com uma explicação de seu cálculo por vias computacionais; iv) os efeitos de tarifas sobre o equilíbrio, junto com uma ilustração gráfica desses impactos.

2 Comportamento dos Consumidores: A Demanda

Suponha que em um dado mercado competitivo de um bem homogêneo, seja possível perguntar para cada consumidor o valor máximo que este pagaria por este produto. Vamos supor, a título ilustrativo, que estamos no mercado de maçãs. Então perguntaríamos para cada consumidor de maçãs qual seria o maior valor que estariam dispostos a pagar por uma maçã. Esse valor máximo, geralmente chamado de disposição a pagar ou preço de reserva, mede o valor que cada indivíduo dá para um certo produto.

É legítimo pensar que pessoas diferentes valorizam mais ou menos um mesmo produto. Desse modo, consumidores terão preços de reserva diferentes em um mesmo mercado, uns mais altos, outros mais baixos. Vamos supor que o preço das maçãs esteja muito alto. Apenas as pessoas que gostam muito de maçãs e, portanto, estão dispostas a pagar um valor mais alto, comprariam maçãs nesse caso. Conforme o preço abaixar, mais pessoas que valorizam menos as maçãs estariam dispostas a comprá-las. Assim, quanto menor o preço, maior é o total de pessoas dispostas a comprar maçãs. Definamos a quantidade demandada como sendo a quantidade total de um bem que os consumidores estão podem e estão dispostos a comprar. Então a um dado preço, a quantidade demandada será dada pela quantidade consumida por todas as pessoas com um preço de reserva ao menos igual ao preço de mercado. Suponhamos que a quantidade demandada seja bem modelada por uma função linear decrescente, por exemplo:

$$Q_d = 6 - 2P$$

cujo gráfico é mostrado na figura 1.

Então, como se vê pelo gráfico, supondo um preço de mercado de R$ 1,00, a quantidade demandada seria 4. Caso o preço aumente para R$ 2,00, a quantidade demandada cai para 2 unidades, uma vez que menos pessoas tem um preço de reserva suficiente para comprar o produto.

Implementamos uma função demanda linear computacionalmente da seguinte maneira:

```
1
2 def demanda(intrd, incld, x, inv = False):
3
4     if inv:
```

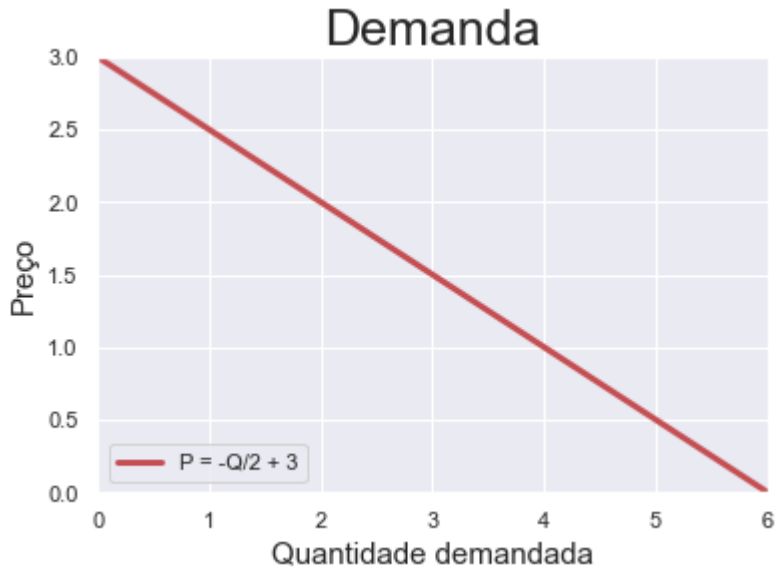

Figura 1: Uma demanda linear

```
        return (x / incld) - (intrd / incld)

    return intrd + incld * x
```

onde **intrd** é o intercepto, **incld** é a inclinação da reta, **inv** é um parâmetro que se verdadeiro retorna a demanda inversa (preço em termos da quantidade) e **x** são os dados (quantidade ou preço a depender se é a função demanda ou a função demanda inversa).

3 Comportamento dos Vendedores: A Oferta

De modo análogo ao que estabelecemos na demanda, para o comportamento dos vendedores há uma quantidade ofertada, que representa a quantidade de um determinado produto que os vendedores podem e querem vender a cada preço. Os vendedores desejam maximizar os seus lucros (ou minimizar seus prejuízos), sujeitos a um determinado custo de produção. A eficiência dos vendedores em produzir é variável de acordo com a tecnologia e a produtividade que eles tem ao seu dispor. Além disso, o custo de oportunidade de cada vendedor também é variável. Assim, os custos de cada vendedor variam. Um vendedor não estará disposto a vender caso o preço de mercado

do produto não seja suficiente para cobrir seus custos[1]. Quanto menor o preço menos vendedores terão o custo baixo o suficiente para tornar atrativo ofertar no mercado. A recíproca também é verdadeira, de modo que quanto maior o preço de mercado, mais vendedores estarão dispostos a ofertar.

Vamos supor que uma oferta seja bem modelada por uma função linear tal como:

$$Q_s = 3 + P$$

cujo gráfico é mostrado na figura 2.

Figura 2: Uma oferta linear

Desse modo, supondo que o preço de mercado seja R$ 2,00, os vendedores desse mercado cumulativamente estariam dispostos a venderem 5 unidades de produto. Caso o preço caísse para R$ 1,00, apenas 4 unidades seriam ofertadas.

A Implementação computacional da função oferta foi feita do seguinte modo:

```
1
2 def oferta(intro, inclo, x, inv = False):
```

[1]Sendo mais específico, o vendedor só produzirá caso o preço seja superior ao custo variável médio, mas isso é um aprofundamento desnecessário para ilustrar o modelo

```
3
4       if inv:
5           return (x / inclo) - (intro / inclo)
6
7       return intro + inclo * x
```

em que os parâmetros tem significado análogo aos parâmetros da função demanda descrita anteriormente.

4 O Equilíbrio de Mercado

O equilíbrio de mercado é atingido no preço em que a quantidade ofertada se iguala à quantidade demandada. Nessa situação, todos os vendedores conseguem vender seus produtos ao preço de mercado e todos os compradores com preço de reserva ao menos igual ao preço de mercado compram o produto. Um desvio desse estado leva a uma perda de bem-estar no sentido de Pareto. Caso o preço aumentasse, haveria um excesso de oferta (mais pessoas dispostas a vender que pessoas dispostas a comprar) e o preço tornaria a baixar até o equilíbrio para que os vendedores não mantivessem excesso de produto. Caso o preço estivesse abaixo do equilíbrio, haveria excesso de demanda (as pessoas querem comprar mais produtos do que estão sendo ofertados). Nesse estado, é lógico para os vendedores aumentarem seus preços para aumentarem seus lucros, levando o preço de volta para o equilíbrio.

Abaixo está uma ilustração gráfica do equilíbrio de mercado para a oferta e a demanda que estabelecemos anteriormente.

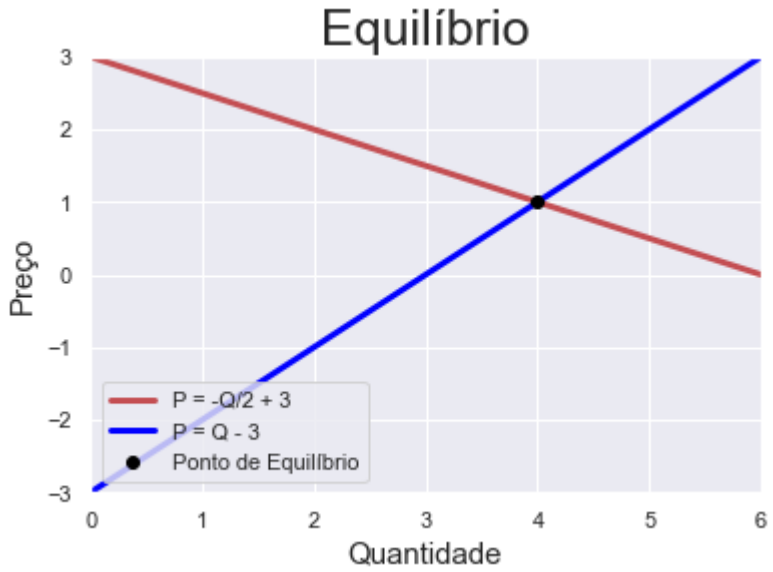

Figura 3: O equilíbrio de mercado

Para encontrar o ponto de equilíbrio implementamos o seguinte código:

```
1
2  import numpy as np
3
4  def excesso_de_demanda(intr_d, incl_d, intr_o, incl_o, p):
5
6      """
7      Retorna o excesso de demanda para um dado preco de oferta
8      dados os interceptos e inclinacoes das funcoes oferta e
       demanda
9      """
10
11     dem = demanda(intr_d, incl_d, p)
12     oft = oferta(intr_o, incl_o, p)
13
14     return dem - oft
15
16 def equilibrio(intr_d, incl_d, intr_o, incl_o, grid_s, grid_e
       , grid_tam = 1000):
17
18     """
19     intr_d   = intercepto da demanda
20     incl_d   = inclinacao da demanda
```

```
    intr_o   = intercepto da oferta
    incl_o   = inclinacao da oferta
    grid_s   = comeco do grid
    grid_e   = fim do grid
    grid_tam = tamanho do grid (1000 = default)
    """

    # Criando o grid e a lista dos valores de excesso de demanda
    grid = np.linspace(grid_s, grid_e, grid_tam)
    val_preco_excesso = []

    # Preenchendo os valores de excesso de demanda
    for ponto in grid:
        val_preco_excesso.append(abs(excesso_de_demanda(intr_d, incl_d,
                                                        intr_o, incl_o, ponto)))

    # Transformando a lista em np.array
    val_preco_excesso = np.array(val_preco_excesso)

    # Encontrando o equilibrio
    preco_eq = grid[np.argmin(val_preco_excesso)]
    qtd_eq = demanda(intr_d, incl_d, preco_eq)

    return qtd_eq, preco_eq
```

Primeiro definimos uma função excesso de demanda, que calcula a diferença entre os valores da demanda e da oferta, para funções com parâmetros estabelecidos pelo usuário. A função equilíbrio usa a função excesso de demanda para encontrar o ponto de equilíbrio, utilizando o conhecimento de que no equilíbrio o excesso de demanda é zero. Desse modo, é possível resolver numericamente o problema do equilíbrio. Criamos um intervalo determinado pelos parâmetros grid_s e grid_e, e subdividimos esse intervalo em grid_tam partes. Iteramos por cada um dos pontos desse intervalo, testando o excesso de demanda nesse ponto. O ponto em que o excesso é 0 será o preço de equilíbrio, e a quantidade de equilíbrio pode então ser encontrada facilmente inserindo o valor do preço de equilíbrio na função demanda ou na função oferta.

5 Extensão do Modelo

O governo pode agir de diversas maneiras na economia, seja estabelecendo preços máximos ou mínimos para determinados bens na economia, concedendo subsídios e estabelecendo empresas públicas, seja regulamentando a atuação de determinados setores por meio da lei. Aqui iremos nos ater a uma atuação em particular: a tributação.

5.1 Impostos

Impostos cobrados pelo Estado sobre um determinado mercado alteram a situação de equilíbrio desse mercado, ao colocar uma cunha entre o preço pago pelo consumidor e o valor recebido pelo ofertante. Impostos podem ser cobrados de diversas maneiras. Podem ser cobrados tanto sobre o consumidor quanto sobre o ofertante. No entanto, pode ser demonstrado que independente de quem recebe o ônus do imposto, o resultado em termos de mudança do ponto de equilíbrio será o mesmo tanto quando o imposto é cobrado sobre a oferta quanto quando o imposto é cobrado sobre a demanda. Nas seções seguintes examinaremos dois tipos de impostos em particular: o imposto sobre quantidade e o imposto sobre valor (ou ad valorem).

5.1.1 Impostos por Quantidade

O imposto sobre quantidade é aquele em que um valor adicional é cobrado para cada unidade vendida do bem. Ou seja, há um acréscimo no preço do bem no valor da taxa, que pode ser expresso como abaixo.

$$P_D = P_S + \tau \tag{1}$$
$$P_D - \tau = P_S \tag{2}$$

As equações acima mostram que o imposto τ faz com que o preço pago pela demanda P_D seja distinto do preço recebido pela oferta P_S. A condição de equilíbrio, como anteriormente é encontrada na situação em que a quantidade ofertada se iguala à quantidade demandada para um dado preço. Vamos assumir que a oferta que paga o imposto. Nesse caso o equilíbrio é atingido no ponto onde $Q_D(P_D) = Q_S(P_S) \implies Q_D(P_D) = Q_S(P_D - \tau)$. Pode ser mostrado que na situação em que o demandante paga o imposto a situação de equilíbrio é a mesma. Suponhamos que tanto a oferta quanto a demanda

sejam lineares e tenham a forma $Q_D(P_D) = a - bP_D$ e $Q_S(P_S) = c + dP_S$. Então o equilíbrio será nos preços expressos abaixo.

$$\begin{aligned} a - bP_D &= c + dP_S \\ a - b(P_s + \tau) &= c + dP_S \\ a - b\tau - c &= dP_S + bP_S \\ P_S &= \frac{a - b\tau - c}{d + b} \\ P_D &= P_S + \tau \end{aligned}$$

A função abaixo encontra o equilíbrio usando essa relação.

```
1
2  # Calcula preco de equilibrio analiticamente para impostos
       por
3  # quantidade
4  def eq_quant_analitico_lin(d_intr, o_intr, d_incl, o_incl,
       imp):
5
6      """
7      d_intr: intercepto da demanda
8      o_intr: intercepto da oferta
9      d_incl: inclinacao da demanda
10     o_incl: inclinacao da oferta
11     imp   : % de impostos
12     """
13
14     # Calcula o preco da oferta e da demanda de equilibrio
15     ps = (d_intr - d_incl * imp - o_intr) / (o_incl + d_incl)
16     pd = ps_eq + imp
17
18     # Calcula as quantidades de equilibrio
19     qd = d_intr - d_incl * pd
20     qs = o_intr + o_incl * ps
21
22     return ps, pd, qs, qd
```

5.1.2 Impostos Ad-Valorem

Os impostos sobre valor são aqueles em que é cobrada uma alíquota percentual sobre o valor de um bem.

$$P_D = (1 + \tau)P_S \tag{3}$$

Nesse caso, assumindo ofertas e demandas lineares temos os seguintes preços de equilíbrio.

$$\begin{aligned}
a - bP_D &= c + dP_S \\
a - b((1 + \tau)P_S) &= c + dP_S \\
a - bP_S - b\tau P_S &= c + dP_S \\
a - c &= P_S(d + b\tau + b) \\
P_S &= \frac{a - c}{d + b\tau + b} \\
P_D &= (1 + \tau)P_S
\end{aligned}$$

Essas equações são usadas no programa abaixo para encontrar a condição de equilíbrio.

```
# Encontra os precos e quantidades de equilibrio
# para um mercado com oferta e demanda lineares
# e imposto ad valorem
def eq_ad_val_analitico_lin(d_intr, o_intr, d_incl, o_incl,
    imp):

    """
    d_intr: intercepto da demanda
    o_intr: intercepto da oferta
    d_incl: inclinacao da demanda
    o_incl: inclinacao da oferta
    imp   : % de impostos
    """
    # Calcula o preco da oferta e da demanda de equilibrio
    ps = (d_intr - o_intr) / (o_incl + d_incl * (1 + imp))
    pd = ps * (1 + imp)

    # Calcula as quantidades de equilibrio
    qd = d_intr - d_incl * pd
    qs = o_intr + o_incl * ps

    return ps, pd, qs, qd
```

5.1.3 Equilíbrio de Mercado: Solução por Força Bruta

Abaixo implementamos uma solução numérica que encontra o equilíbrio percorrendo um grid unidimensional e calculando o valor absoluto do excesso de demanda para em cada ponto. O ponto de equilíbrio será aproximadamente aquele com o menor excesso de demanda absoluto, ou seja, o ponto em que a oferta e a demanda estão mais próximos. O programa funciona tanto para impostos ad valorem quanto para impostos por quantidade, o que pode ser regulado pelo parâmetro $ad_valorem$.

```
# Calcula os precos e quantidades de equilibrio
# dadas funcoes de oferta e demanda atraves de
# tecncias de ponto fixo
def eq_imp_grid(f_of, f_dem, imp, pe, ps = 0, pt = 1000,
    ad_valorem = True):

    """
    f_of      : funcao de oferta
    f_dem     : funcao de demanda
    imp       : imposto
    ps        : primeiro ponto do grid (default = 0)
    pe        : ultimo ponto do grid
    ad_valorem: se o imposto sera advalorem ou por quantidade
    (default = True)
    """

    if ad_valorem:
        assert imp >= 0 and imp <= 1

    # Cria vetor de precos e excesso de demanda
    p_vec = np.linspace(ps, pe, pt)
    val_excesso_dem = []

    # Para cada preco no vetor
    for ps in p_vec:

        # Preco pago pela demanda eh o preco
        # recebido pela oferta acrecido do imposto
        if ad_valorem:
            pd = (1 + imp) * ps

        else:
            pd = ps + imp

        Qd = f_dem(pd)
```

```
        Qs = f_of(ps)

        # Calcula excesso de demanda
        excesso = abs(Qd - Qs)
        val_excesso_dem.append(excesso)

    # Encontra equilibrio
    val_excesso_dem = np.array(val_excesso_dem)
    ps_eq = p_vec[np.argmin(val_excesso_dem)]

    if ad_valorem:
        pd_eq = (1 + imp) * ps_eq

    else:
        pd_eq = ps_eq + imp

    qs_eq = f_of(ps_eq)
    qd_eq = f_dem(pd_eq)

    #assert round(qs_eq, 2) == round(qd_eq, 2)

    return ps_eq, pd_eq, qs_eq, qd_eq
```

5.1.4 Equilíbrio de Mercado: Solução por Bisseção

A presente função encontra o equilíbrio numericamente usando um método mais eficiente que o anterior. Ela é também uma função mais flexível, aceitando ofertas e demandas não lineares. Isso é facilitado pelo uso de keyword arguments, como mostrado pelo parâmetro **params. Esse parâmetro permite a introdução de um número variável de entradas na função, facilitando o uso de funções não lineares que possuam vários parâmetros. O método da bisseção opera da seguinte maneira: 1. são escolhidos dois pontos no domínio da função que se queira encontrar a raiz, um em que a imagem é positiva e outro em que a imagem é negativa; 2. calcula-se a imagem da função no ponto na média entre os dois primeiros pontos escolhidos; 3. caso a imagem seja positiva, o primeiro valor que determinava uma imagem positiva é atualizado para o ponto da média; 4. caso a imagem seja negativa, o segundo valor, que determinava uma imagem negativa é atualizado para o ponto da média; 5. se a imagem for 0 dentro de uma certa tolerância, encontramos a raiz da função.

```

```

```
# Algoritimo que implementa o algoritimo da bissecao
def bissecao(f, x1, x2, tol = 4, **params):

    """
    f        : funcao que queremos a raiz
    x1       : valor que determina imagem positiva
    x2       : valor que determina imagem negativa
    tol      : tolerancia no arredondamento
    **params: kwargs para a funcao
    """

    # Busca um valor que leve a funcao
    # a uma imagem negativa
    i = 0
    while f(x2, **params) >= 0 and i < 100:

        x2 += 2
        i += 1

    # Caso na iteracao acima i == 100
    # assumimos que nao encontramos um
    # valor que leve a funcao para uma
    # imagem negativa
    assert i < 100

    # Media dos valores
    xm = (x1 + x2) / 2

    # Enquanto nao encontrarmos a raiz
    while round(abs(f(xm, **params)), tol) > 0:

        if f(xm, **params) > 0:

            x1 = xm

        elif f(xm, **params) < 0:

            x2 = xm

        xm = (x1 + x2) / 2

    return xm

# Funcao encontra o excesso de demanda para
# quaisquer funcoes de oferta e demanda
```

```
47 def exc_de_dem(ps, **params):
48
49     """
50     ps : preco da oferta
51     **params: parametros para as funcoes oferta ('f_of') e
52               demanda ('f_dem').
53               1. Paramentros da oferta devem comecar com a
54               letra 'o' e parametros da demanda devem
55               comecar com a letra 'd'.
56
57               2. 'ad_valorem' eh o parametro booleano que
58               indica se estamos lidando com um imposto
    ad_valorem
59               ou por quantidade.
60
61               3. 'imp' eh o parametro que da o imposto.
62     """
63
64     # Preco da demanda se o imposto for por quantidade
65     pd = ps + params['imp']
66
67     # Preco da demanda se o imposto for por quantidade
68     if params['ad_valorem']:
69
70         pd = ps * (1 + params['imp'])
71
72     # Dicionarios para os parametros da funcoes oferta e
    demanda
73     params_oferta = {}
74     params_demanda = {}
75
76     # Para cada parametro e seu respectivo valor
77     for key, value in params.items():
78
79         # Se comecar com d eh da demanda
80         if key[0] == 'd':
81
82             params_demanda[key] = value
83
84         # Se comecar com o eh da oferta
85         elif key[0] == 'o':
86
87             params_oferta[key] = value
88
89     # Calcula demanda e oferta para os precos
```

```
90      Qd = params['f_dem'](pd, **params_demanda)
91      Qs = params['f_of'](ps, **params_oferta)
92
93      return Qd - Qs
94
95  # Calcula o equilibrio com impostos para qualquer
96  # oferta e demanda por meio do metodo da bissecao
97  def eq_imp(ps1, ps2, **params):
98
99      # Se for ad valorem garantir que esta entre
100     # 0 e 1
101     if params['ad_valorem']:
102         assert params['imp'] >= 0 and params['imp'] <= 1
103
104     # Preco da oferta de equilibrio pela bissecao
105     ps_eq = bissecao(exc_de_dem, ps1, ps2, **params)
106
107     # Calcula preco da demanda de equilibrio
108     if params['ad_valorem']:
109         pd_eq = (1 + params['imp']) * ps_eq
110
111     else:
112         pd_eq = ps_eq + params['imp']
113
114     params_oferta = {}
115     params_demanda = {}
116
117     # Encontra parametros da oferta e da demanda
118     for key, value in params.items():
119
120         if key[0] == 'd':
121
122             params_demanda[key] = value
123
124         elif key[0] == 'o':
125
126             params_oferta[key] = value
127
128     # Calcula as quantidades de equilibrio
129     Qd_eq = params['f_dem'](pd_eq, **params_demanda)
130     Qs_eq = params['f_of'](ps_eq, **params_oferta)
131
132     #assert round(qs_eq, 2) == round(qd_eq, 2)
133
134     return ps_eq, pd_eq, Qs_eq, Qd_eq
```

5.2 O Impacto das Tarifas sobre o Equilíbrio

Examinaremos agora como ações simples do governo podem alterar o equilíbrio no nosso modelo. Examinaremos os impactos no equilíbrio de mercado de um imposto cobrado sobre i) consumidores e ii) empresas.

Para implementar este cenário de impostos utilizamo o seguinte código, que altera o intercepto vertical da oferta ou da demanda de acordo com o imposto, já que o imposto não altera a inclinação da reta, sendo aplicado igualmente sobre todos os consumidores.

```
def equilibrio_com_imposto(intr_d, incl_d, intr_o, incl_o, imp,
                           grid_e, grid_s = 0, grid_tam = 1000,
                           sobre_demanda = False):
    """
    intr_d        = intercepto da demanda
    incl_d        = inclinacao da demanda
    intr_o        = intercepto da oferta
    incl_o        = inclinacao da oferta
    grid_s        = comeco do grid (0 = default)
    grid_e        = fim do grid
    imp           = porcentagem de imposto
    grid_tam      = tamanho do grid (1000 = default)
    sobre_demanda = se o imposto eh sobre a oferta ou a demanda
    """

    # Criando o grid e a lista dos valores de excesso de demanda
    grid = np.linspace(grid_s, grid_e, grid_tam)
    val_preco_excesso = []

    # Garantindo que o imposto eh um valor entre 0 e 1
    assert imp >= 0 and imp <= 1

    if sobre_demanda:
        intr_d = (1 - imp) * intr_d
        for ponto in grid:
            val_preco_excesso.append(abs(excesso_de_demanda(intr_d, incl_d,
                                                            intr_o, incl_o, ponto)))
    else:
        intr_o = (1 - imp) * intr_o
```

```
30          for ponto in grid:
31              val_preco_excesso.append(abs(excesso_de_demanda(
    intr_d, incl_d,
32                                                      intr_o,
    incl_o, ponto)))
33
34      # Transformando a lista em np.array
35      val_preco_excesso = np.array(val_preco_excesso)
36
37      # Encontrando o equilibrio
38      preco_eq = grid[np.argmin(val_preco_excesso)]
39      qtd_eq = demanda(intr_d, incl_d, preco_eq)
40
41      return qtd_eq, preco_eq
```

5.2.1 Imposto sobre os Consumidores

Um imposto sobre os consumidores afetará unicamente a função demanda, não alterando a função oferta, já que a qualquer preço os vendedores ainda estão dispostos a vender as mesmas quantidades. Vamos supor que o imposto cobrado é uma porcentagem fixa e igual para todos os consumidores. Desse modo, cada consumidor reduzirá sua disposição a pagar no valor da porcentagem, deslocando a demanda para baixo. Por exemplo, para um imposto de 20%, caso uma pessoa tivesse uma disposição a pagar de R$ 6,00 anteriormente, agora ela poderá pagar apenas R$ 4,80 para o mesmo produto. Isso deslocará a curva de demanda para baixo, reduzindo também a quantidade e o preço de equilíbrio. Abaixo consta uma ilustração gráfica desse exemplo.

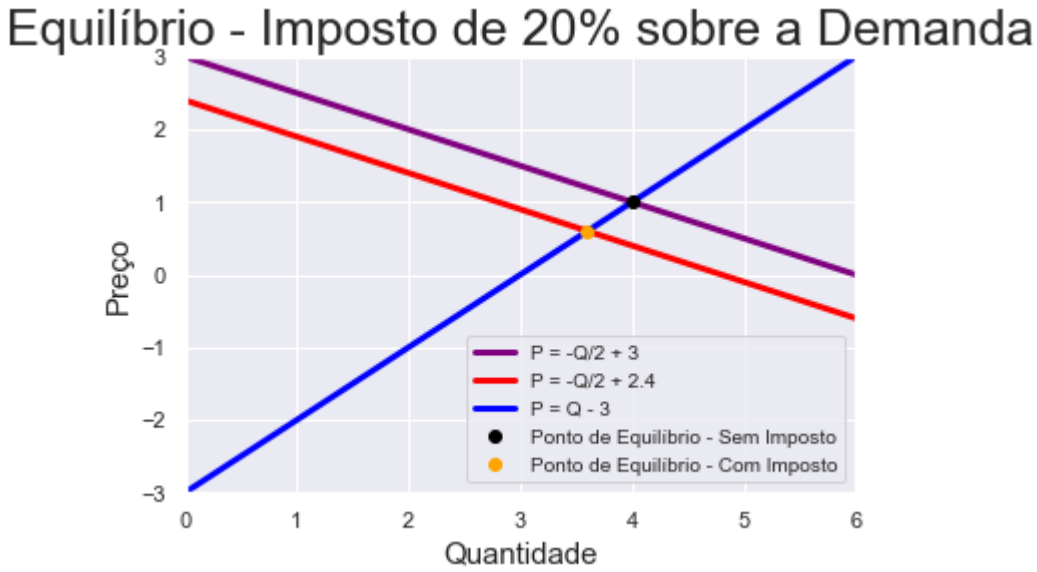

Figura 4: O equilíbrio de mercado com um imposto de 20% sobre a demanda

5.2.2 Imposto sobre os Vendedores

Analogamente ao cenário do imposto sobre os consumidores, um imposto cobrado unicamente sobre os vendedores afeta apenas a oferta. O aumento dos custos dos vendedores faz com que a cada preço eles sejam capazes de ofertar uma quantidade menor de bens, deslocando a curva da oferta para a esquerda e reduzindo a quantidade de equilíbrio e aumentando o preço de equilíbrio. Abaixo consta um exemplo gráfico do deslocamento do equilíbrio em uma situação em que é cobrado um imposto uniforme de 20% sobre os vendedores.

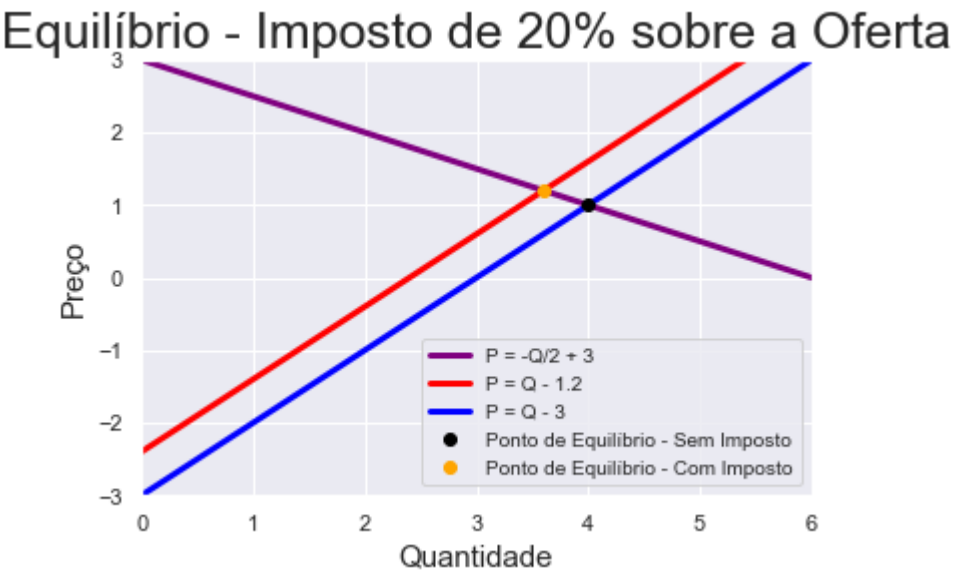

Figura 5: O equilíbrio de mercado com um imposto de 20% sobre a oferta