UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

HENRIQUE REIS CAMPOS

Análise de Taxa de Juros e Derivativos

Campinas
24/11/2023

HENRIQUE REIS CAMPOS

Análise de Taxa de Juros e Derivativos

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado, sob a orientação do(a) Prof. Laércio Vedite.

RESUMO

CONTEÚDOS

1. Introdução

As taxas de juros afetam as atividades econômicas em todos os níveis. Bancos centrais, incluindo o Bacen - Banco Central do Brasil - têm como alvo as taxas de juros como uma ferramenta de política para influenciar a atividade econômica. Derivativos de taxas de juros são populares entre investidores que necessitam de fluxos de caixa personalizados ou têm visões específicas sobre movimentos nas taxas de juros. Um dos principais desafios que os *traders* de derivativos de taxas de juros enfrentam é ter um procedimento de precificação sólido e robusto para esses produtos. Isso envolve entender o comportamento complicado de um movimento individual nas taxas de juros. Vários modelos de taxas de juros foram propostos para estudos financeiros. Alguns modelos comuns estudados em finanças são o modelo Vasicek, o modelo CIR e o modelo Hull-White. Esses modelos de taxas de juros envolvem a modelagem da taxa curta e dependem de fatores (ou fontes de incerteza), sendo que a maioria deles usa apenas um fator. Modelos de taxas de juros de dois fatores e multifatores também foram propostos. O trabalho a seguir tem como objetivo o entendimento de alguns desses métodos de precificação e utilização da linguagem de programação Python para implementação desses métodos.

2. Fundamentos

2.1. Instrumentos de Renda Fixa

Corporações e governos emitem instrumentos financeiros de renda fixa como meio de levantar dinheiro (HULL). O detentor de tais dívidas empresta dinheiro e espera receber o principal quando a dívida vencer. O emissor que deseja tomar empréstimo pode emitir uma quantia fixa de pagamento de juros durante a vida da dívida em momentos predefinidos. O detentor de títulos de dívida, como títulos, notas e bônus do Tesouro de países, enfrenta o risco de inadimplência pelo emissor. Acredita-se que títulos emitidos por governos enfrentam o menor risco de inadimplência, pois podem facilmente aumentar impostos e criar mais dinheiro para pagar as dívidas em aberto.

A maioria dos títulos paga uma quantia fixa de juros semestralmente, enquanto alguns pagam trimestral ou anualmente. Esses pagamentos de juros também são chamados de cupons. Eles são cotados como uma porcentagem do valor nominal ou valor de face do título em uma base anual. Por exemplo, um título do Tesouro de cinco anos de R$10.000 com uma taxa de cupom de 5% paga cupons de R$500 a cada ano, ou cupons de R$250 a cada seis meses, até a data de vencimento. Se as taxas de juros caírem e novos títulos do Tesouro pagarem uma taxa de cupom de 3%, o comprador do novo título receberá apenas cupons de R$300 anualmente, enquanto os detentores existentes do título de 5% continuarão a receber R$500 anualmente. À medida que as características dos títulos influenciam seus preços, eles estão intimamente relacionados aos níveis atuais das taxas de juros de maneira inversa, ou seja, o valor do título diminui à medida que as taxas de juros aumentam. À medida que as taxas de juros diminuem, os preços dos títulos aumentam.

2.2. Curva de Juros

Em um ambiente de curva de rendimento normal, as taxas de juros de longo prazo são mais altas do que as taxas de juros de curto prazo. Os investidores esperam ser compensados com retornos mais elevados quando emprestam dinheiro por um período mais longo, uma vez que estão expostos a um risco de inadimplência mais alto. A curva de rendimento normal ou positiva é considerada ascendente, como mostrado no gráfico a seguir:

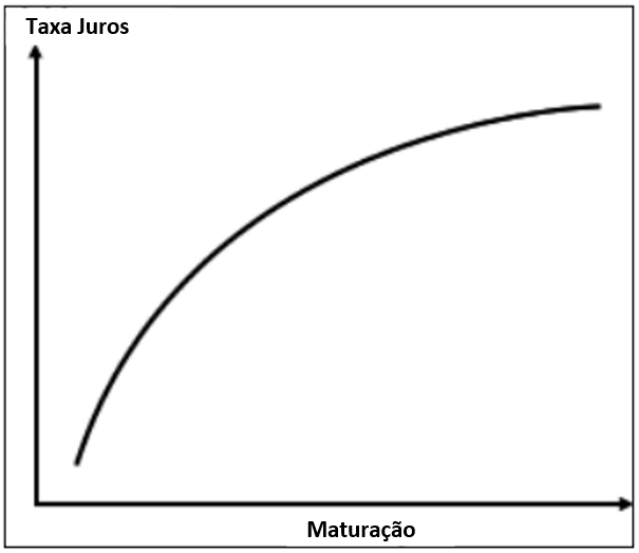

Figura 1: Curva juros em um ambiente econômico considerado normal.

Em determinadas condições econômicas, a curva de rendimento pode se inverter. As taxas de juros de longo prazo são mais baixas do que as taxas de juros de curto prazo. Essa condição ocorre quando há escassez de dinheiro, o que pode representar um cenário de aperto monetário por parte dos Bancos Centrais, por exemplo. Investidores estão dispostos a renunciar aos ganhos de longo prazo para preservar seu patrimônio a curto prazo. Durante períodos de alta inflação, onde a taxa de inflação supera a taxa de juros dos cupons, pode-se observar taxas de juros negativas. Investidores estão dispostos a pagar a curto prazo apenas para proteger seu patrimônio a longo prazo. A curva de rendimento invertida é considerada descendente, como mostrado no gráfico a seguir:

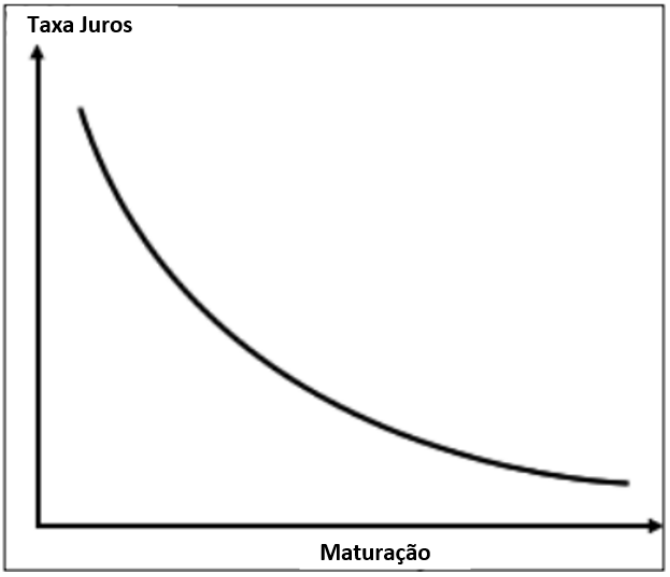

Figura 2: Curva juros em um ambiente econômico de condições monetárias adversas.

2.3. Precificando um instrumento de cupom zero

Um instrumento de cupom zero é um título que não paga nenhum juro periódico, exceto no vencimento, quando o principal ou valor nominal é reembolsado. Títulos de cupom zero também são chamados de títulos de desconto puro. (WEIMING)

Um título de cupom zero pode ser precificado da seguinte forma:

$$precificação\ de\ um\ título\ cupom\ zero\ = \frac{valor\ nominal}{(1+y)^t}$$

Aqui, y é o rendimento ou taxa anualmente compostos do título, e t é o tempo restante até o vencimento do título. Exemplificando, considere um título de cupom zero de 5 anos com um valor nominal de R$100. O rendimento é de 5%, com composição anual. O preço pode ser calculado da seguinte forma:

$$\frac{100}{(1+0{,}05)^5} = R\$78,35$$

Implementando esse método de precificação em Python, temos:

```
In [1]: def zero_cupon_bond(nom, y, t):
            """Precificando um título de cupom zero

            nom - Valor nominal do título
            y - Taxa anual de juros do título
            t - Tempo de manuração do título em anos
            """
            return nom/(1 + y) ** t

In [2]: zero_cupon_bond(100, 0.05, 5)

Out[2]: 78.35261664684589
```

Figura 3: Implementação em Python do método de precificação de um título de cupom zero.

2.4. Taxas *spot* e taxas de cupom zero

À medida que a frequência de composição aumenta (por exemplo, de anual para diária), o valor futuro do dinheiro atinge um limite exponencial. Ou seja, o valor presente de R$100 hoje alcançará um valor futuro de $R\$100e^{RT}$ quando for investido a uma taxa de composição contínua R por um período de tempo T. Descontando esses valores para um título que paga R$100 em um momento futuro T, com uma taxa de desconto continuamente composta R, seu valor no momento zero é $\frac{100}{e^{RT}}$. Essa taxa é conhecida como a taxa *spot*. (WEIMING)

As taxas spot representam as taxas de juros atuais para várias maturidades, caso se queira emprestar ou tomar empréstimo agora. As taxas zero representam a taxa interna de retorno de títulos de cupom zero. Pode-se usar as taxas *spot* e as taxas zero de títulos com diferentes vencimentos para construir a curva de rendimento atual.

2.5. Construindo uma curva de juros por *bootstrapping*

As taxas spot de curto prazo podem ser derivadas diretamente de vários títulos de curto prazo, como títulos de cupom zero, *T-bills*, notas e depósitos cambiais, entre outros. No entanto, as taxas *spot* de longo prazo são geralmente derivadas dos preços de títulos de longo prazo por meio de um processo de *bootstrapping*, levando em consideração as taxas *spot* das maturidades correspondentes à data de pagamento do cupom. Após obter as taxas spot de curto prazo e longo prazo, a curva de rendimento pode então ser construída. (WEIMING)

A seguir, será ilustrado o processo de *bootstrapping* da curva de juros com um exemplo. A tabela a seguir mostra uma lista de títulos com diferentes vencimentos e preços:

Valor nominal do título	Tempo para maturação em anos	Cupom anual em Reais (R$)	Valor do título em Reais (R$)
100	0,25	0	97,50
100	0,50	0	94,90
100	1,00	0	90,00
100	1,50	8	96,00
100	2,00	12	101,60

Um investidor que adquire hoje um título de cupom zero de 3 meses por R$97,50 receberia um juro de R$2,50. A taxa spot de 3 meses pode ser calculada da seguinte forma:

$$97,50 = \frac{100}{e^{0,25y}}$$
$$e^{0,25y} = 1,0256$$
$$y = 4\,ln(1,0256) = 0,10127$$

Assim, a taxa zero de 3 meses é de 10,127% com composição contínua. As taxas *spot* dos títulos de cupom zero são calculadas na tabela a seguir:

Tempo para maturação em anos	Taxa *spot* (%)
0,25	10,127
0,50	10,469
1,00	10,536

Usando essas taxas spot, agora podemos precificar o título de 1,5 anos da seguinte forma:

$$4e^{(-0,10469)(0,5)} + 4e^{(-0,10536)(1,0)} + 104e^{(-y)(1,5)} = 96$$

Para resolver para y, a taxa spot para o título de 1,5 anos e o título de 2 anos é de 10,681% e 10,808%, respectivamente.

O código a seguir é uma implementação do bootstrapping de uma curva de rendimento em Python.

In [22]:

```python
import math

class BootstrapYieldCurve():

    def __init__(self):
        self.zero_rates = dict() # Map each T to a zero rate
        self.instruments = dict() # Map each T to an instrument

    def add_instrument(self, par, T, coup, price, compounding_freq=2):
        self.instruments[T] = (par, coup, price, compounding_freq)

    def get_zero_rates(self):
        self.__bootstrap_zero_coupons__()
        self.__get_bond_spot_rates__()
        return [self.zero_rates[T] for T in self.get_maturities()]

    def get_maturities(self):
        return sorted(self.instruments.keys())

    def __bootstrap_zero_coupons__(self):
        for T in self.instruments.iterkeys():
            (par, coup, price, freq) = self.instruments[T]
            if coup == 0:
                self.zero_rates[T] = self.zero_coupon_spot_rate(par, price, T)

    def __get_bond_spot_rates__(self):
        for T in self.get_maturities():
            instrument = self.instruments[T]
            (par, coup, price, freq) = instrument
            if coup != 0:
                self.zero_rates[T] = self.__calculate_bond_spot_rate__(T, instrument)
```

```python
    def __calculate_bond_spot_rate__(self, T, instrument):
        try:
            (par, coup, price, freq) = instrument
            periods = T * freq # Number of coupon payments
            value = price
            per_coupon = coup / freq # Coupon per period
            for i in range(int(periods)-1):
                t = (i+1)/float(freq)
                spot_rate = self.zero_rates[t]
                discounted_coupon = per_coupon * math.exp(-spot_rate*t)
                value -= discounted_coupon
            last_period = int(periods)/float(freq)
            spot_rate = -math.log(value(par+per_coupon))/last_period
            return spot_rate
        except:
            print("Error: spot rate not found for T=%s" % t)

    def zero_coupon_spot_rate(self, par, price, T):
        spot_rate = math.log(par/price)/T
        return spot_rate
```

Figura 4: Implementação do método de bootstrapping em Python.

Podemos instanciar a classe "*BootstrapYieldCurve*" e adicionar as informações de cada título da tabela anterior:

```
In [23]: yield_curve = BootstrapYieldCurve()
         yield_curve.add_instrument(100, 0.25, 0., 97.5)
         yield_curve.add_instrument(100, 0.5, 0., 94.9)
         yield_curve.add_instrument(100, 1.0, 0., 90.)
         yield_curve.add_instrument(100, 1.5, 8, 96., 2)
         yield_curve.add_instrument(100, 2., 12, 101.6, 2)
         y = yield_curve.get_zero_rates()
         x = yield_curve.get_maturities()
```

Figura 5: Implementação de um exemplo do método de bootstrapping em Python.

Utilizando o método "*get_zero_rates*" na classe retorna uma lista de taxas *spot* na mesma ordem das maturidades, que são armazenadas nas variáveis y e x, respectivamente. Ao plotar os resultados obtidos em um gráfico, obtém-se o seguinte *output*:

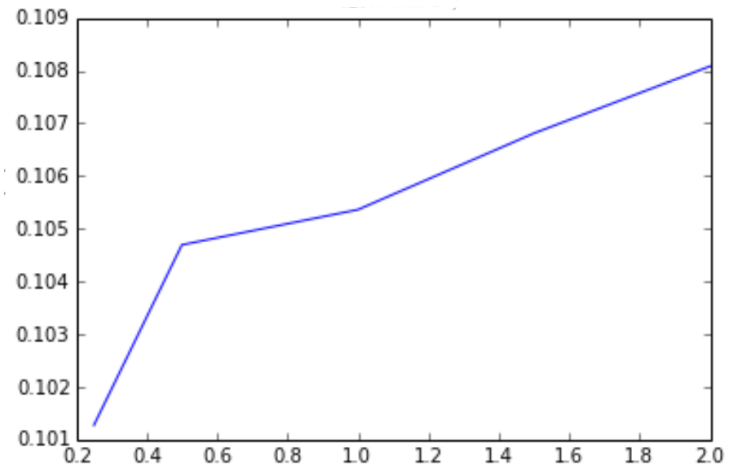

Figura 6: Output da implementação do método get_zero_rates.

Em um ambiente de curva de juros normal, onde as taxas de juros aumentam à medida que as maturidades aumentam, podemos obter uma curva de rendimento ascendente.

2.6. *Forward Rates*

Um investidor que planeja investir em um momento posterior pode necessitar saber como as taxas de juros futuras podem parecer, conforme implícito pela estrutura a termo atual das taxas de juros. Por exemplo, o investidor poderia se perguntar: Qual é a taxa *spot* de um ano daqui a um ano? Para responder a essa pergunta, pode-se calcular as taxas a termo para o período entre T_1 e T_2 usando a seguinte fórmula:

$$r_{forward} = \frac{r_2 T_2 - r_1 T_1}{T_2 - T_1}$$

Aqui, r_1 e r_2 são as taxas de juros anuais com composição contínua nos períodos de tempo T_1 e T_2, respectivamente. (WEIMING)

O seguinte código Python auxilia ajuda a gerar uma lista de taxas a termo a partir de uma lista de taxas *spot*:

```python
class ForwardRates(object):

    def __init__(self):
        self.forward_rates = []
        self.spot_rates = dict()

    def add_spot_rate(self, T, spot_rate):
        self.spot_rates[T] = spot_rate

    def __calculate_forward_rate___(self, T1, T2):
        R1 = self.spot_rates[T1]
        R2 = self.spot_rates[T2]
        forward_rate = (R2*T2 - R1*T1)/(T2 - T1)
        return forward_rate

    def get_forward_rates(self):
        periods = sorted(self.spot_rates.keys())
        for T2, T1 in zip(periods, periods[1:]):
            forward_rate = self.__calculate_forward_rate___(T1, T2)
            self.forward_rates.append(forward_rate)
        return self.forward_rates
```

Figura 7: Implementação em Python do método de cálculo das forward rates.

Usando as taxas spot derivadas da nossa curva de rendimento anterior, obtemos o seguinte resultado:

```
In [25]: fr = ForwardRates()
         fr.add_spot_rate(0.25, 10.127)
         fr.add_spot_rate(0.50, 10.469)
         fr.add_spot_rate(1.00, 10.536)
         fr.add_spot_rate(1.50, 10.681)
         fr.add_spot_rate(2.00, 10.808)
         print(fr.get_forward_rates())

         [10.810999999999998, 10.603, 10.971, 11.189]
```

Figura 8:Output da implementação do método presente na Figura 7.

Utilizando o método "*get_forward_rates*" da classe "*ForwardRates*", é retornada uma lista de taxas a termo, começando pelo próximo período de tempo.

2.7. Calculando o rendimento até o vencimento (*Yield to Maturity* - YTM)

O rendimento até o vencimento (*Yield to Maturity* - YTM) mede a taxa de juros, conforme implicitamente indicada pelo título, levando em consideração o valor presente de todos os pagamentos futuros de cupons e o principal. Assume-se que os detentores de títulos podem investir os cupons recebidos à taxa YTM até o vencimento do título; de acordo com as expectativas neutras ao risco, os pagamentos recebidos devem ser iguais ao preço pago pelo título. (WEIMING)

Exemplificando: um título de 5,75% que vencerá em 1,5 anos com um valor nominal de R$100. O preço do título é de R$95,0428 e os cupons são pagos semestralmente. A equação de precificação pode ser expressa da seguinte forma:

$$95,0428 = \frac{c}{(1+\frac{y}{n})^{nT_1}} + \frac{c}{(1+\frac{y}{n})^{nT_2}} + \frac{100+c}{(1+\frac{y}{n})^{nT_3}}$$

Aqui, c é o valor em dólares do cupom pago em cada período, T é o período de pagamento em anos, n é a frequência de pagamento de cupons, e y é o YTM que estamos interessados em resolver. Resolver o YTM é geralmente um processo complexo, e a maioria dos calculadores de YTM de títulos usa o método de Newton como um processo iterativo.

O método para calcular o YTM de títulos é ilustrado pelo seguinte código Python.

```
In [33]: from scipy import optimize

         def bond_ytm(price, par, T, coup, freq=2, guess=0.05):
             freq = float(freq)
             periods = T * freq
             coupon = coup / 100. * par / freq
             dt = [(i + 1) / freq for i in range(int(periods))]
             ytm_func = lambda y: sum([coupon / (1 + y / freq) ** (freq * t) for t in dt]) + par / (1 + y / freq) ** (freq * T) - price

             return optimize.newton(ytm_func, guess)

In [35]: ytm = bond_ytm(95.0428, 100, 1.5, 5.75, 2)
         print(ytm)

         0.09369155345239477
```

Figura 9: Implementação do método para cálculo do YTM, juntamente com output do método .

2.8. Calculando o preço de um título

Quando o YTM é conhecido, pode-se obter o preço do título da mesma maneira que usamos a equação de precificação investigada anteriormente.

```
In [36]: def bond_price(par, T, ytm, coup, freq=2):
             freq = float(freq)
             periods = T*freq
             coupon = coup/100.*par/freq
             dt = [(i+1)/freq for i in range(int(periods))]
             price = sum([coupon/(1+ytm/freq)**(freq*t) for t in dt]) + par/(1+ytm/freq)**(freq*T)
             return price
```

Figura 10: Implementando o método para calcular o preço de um título em Python.

Substituindo os mesmos valores do exemplo na Figura 9, obtém-se o seguinte resultado:

```
In [38]: bond_price(100, 1.5, ytm, 5.75, 2)
Out[38]: 95.04280000000004
```

Figura 11: Utilizando a função bond_price com os mesmos valores utilizados no exemplo da Figura 9.

Isso fornece o mesmo preço original do título discutido no exemplo anterior. Usando as funções "*bond_ytm*" e "*bond_price*", podemos utilizá-las para outros fins na precificação de títulos, como encontrar a duração modificada e a convexidade do título. Essas duas características dos títulos são importantes para os traders de títulos para ajudá-los a formular diversas estratégias de negociação e proteger o risco.

2.9. *Duration* de um título

A *duration* é uma medida de sensibilidade dos preços dos títulos a mudanças nas taxas de juros. Algumas medidas de *duration* incluem: *duration* efetiva, *duration* de Macaulay e *duration* modificada. O tipo de *duration* que será discutida é a *duration* modificada, que mede a variação percentual no preço do título em relação a uma variação percentual na taxa de juros (tipicamente 1% ou 100 *basis-points* (bps)). (WEIMING)

Quanto maior a *duration* de um título, mais sensível ele é a mudanças nas taxas de juros. Inversamente, quanto menor a *duration* de um título, menos sensível ele é a mudanças nas taxas de juros. A *duration* modificada de um título pode ser considerada como a primeira derivada da relação entre preço e taxa de juros:

$$duration\ modificada \simeq \frac{P^{-} - P^{+}}{2(P_0)(dY)}$$

Aqui, dY é a variação de rendimento fornecida, P^{-} é o preço do título a partir de uma diminuição de rendimento por dY, P^{+} é o preço do título a partir de um aumento de rendimento por dY, e P_0 é o preço inicial do título. Deve-se observar que a duração descreve a relação linear preço-rendimento para uma pequena mudança em Y. Como a curva de rendimento não é linear, o uso de um grande valor de dYnão aproxima bem a medida de duração.

A implementação do método para cálculo da *duration* modificada é fornecida no seguinte código Python. A função "*bond_mod_duration*" utiliza a função "*bond_ytm*" discutida anteriormente para determinar o rendimento do título com o valor inicial fornecido. Além disso, ele utiliza a função "*bond_price*" para determinar o preço do título com a variação dada no rendimento:

```
In [39]: def bond_mod_duration(price, par, T, coup, freq, dy=0.01):
             ytm = bond_ytm(price, par, T, coup, freq)

             ytm_minus = ytm - dy

             price_minus = bond_price(par, T, ytm_minus, coup, freq)

             ytm_plus = ytm + dy
             price_plus = bond_price(par, T, ytm_plus, coup, freq)

             mduration = (price_minus-price_plus)/(2*price*dy)
             return mduration
```

Figura 12: Implementando método para cálculo da duration modificada em Python .

Agora, é possível descobrir a *duration* modificada do título de 5,75% discutido anteriormente, que vencerá em 1,5 anos com um valor nominal de R$100 e um preço do título de R$95,0428, usando a função "*bond_mod_duration*":

```
In [40]: print(bond_mod_duration(95.04, 100, 1.5, 5.75, 2, 0.01))

         1.3921788121706968
```

Figura 13: Output do método implementado na Figura 12 .

Assim, a *duration* modificada do título é de 1,392 anos.

2.10. Convexidade do título

Convexidade é uma medida de sensibilidade da *duration* de um título a mudanças nas taxas de juros (WEIMING). Pode-se pensar na convexidade como a segunda derivada da relação entre preço e taxa de juros:

$$convexidade \simeq \frac{P^{-} + P^{+} - 2P_0}{(P_0)(dY)^2}$$

Os traders de títulos utilizam a convexidade como uma ferramenta de gerenciamento de risco para medir a quantidade de risco de mercado em seus portfólios de investimentos. Portfólios com maior convexidade são menos afetados pelas volatilidades das taxas de juros do que portfólios com menor convexidade, considerando a mesma duração e taxa de juros do título. Portanto, títulos com maior convexidade são mais caros do que aqueles com menor convexidade, mantendo tudo o mais constante.

A implementação do método para cálculo da convexidade de título em Python é fornecida da seguinte forma:

```
In [41]: def bond_convexity(price, par, T, coup, freq, dy=0.01):
             ytm = bond_ytm(price, par, T, coup, freq)
             ytm_minus = ytm - dy
             price_minus = bond_price(par, T, ytm_minus, coup, freq)

             ytm_plus = ytm + dy
             price_plus = bond_price(par, T, ytm_plus, coup, freq)

             convexity = (price_minus+price_plus-2*price)/(price*dy**2)
             return convexity
```

Figura 14: Implementação do método para cálculo da convexidade de título em Python.

Agora, é possível encontrar a convexidade do título de 5,75% discutido anteriormente, que vencerá em 1,5 anos com um valor nominal de R$100 e um preço do título de R$95,0428, usando a função "*bond_convexity*":

```
In [42]: print(bond_convexity(95.0428, 100, 1.5, 5.75, 2))

         2.6339593903438367
```

Figura 15: Output de exemplo utilizando o método implementado na Figura 14..

A convexidade do título é 2,63. Para dois títulos com o mesmo valor nominal, cupom e vencimento, a convexidade pode ser diferente, dependendo da sua posição na curva de juros. Títulos com maior convexidade exibirão maiores variações de preço para a mesma mudança na taxa de juros.

3. Estudo de caso

Em modelagem de taxas de juros curtas, a taxa curta $r(t)$ é a taxa *spot* em um momento específico. É descrita como uma taxa de juros anualizada com composição contínua para um período infinitesimalmente curto na curva de juros. A taxa curta assume a forma de uma variável estocástica em modelos de taxa de juros, onde as taxas de juros podem mudar em pequenas quantidades em cada ponto do tempo. Modelos de taxa curta tentam modelar a evolução das taxas de juros ao longo do tempo e, idealmente, descrever as condições econômicas em determinados períodos.

Os modelos de taxa curta são frequentemente usados na avaliação de derivativos de taxa de juros. Títulos, instrumentos de crédito, hipotecas e produtos de empréstimo são sensíveis a mudanças nas taxas de juros. Modelos de taxa curta são usados como componentes de taxa de juros em conjunto com implementações de precificação, como métodos numéricos, para auxiliar na precificação desses derivativos.

A modelagem de taxas de juros é considerada um tópico bastante complexo, uma vez que as taxas de juros são afetadas por uma multiplicidade de fatores, como estados econômicos, decisões

políticas, intervenção governamental e leis de oferta e demanda. Vários modelos de taxa de juros foram propostos para dar conta de várias características das taxas de juros.

Neste trabalho, analisaremos alguns dos modelos de taxa curta de um fator mais comumente usados em estudos financeiros, a saber, o modelo de Vasicek, modelo de Cox-Ingersoll-Ross, modelo de Rendleman e Bartter, e modelo de Brennan e Schwartz. Usando Python, será realizado uma simulação de um caminho para obter uma visão geral do processo do caminho da taxa de juros. Outros modelos comumente discutidos em finanças incluem o modelo de Ho-Lee, modelo de Hull-White e modelo de Black-Karasinski.

.

3.1. Modelo de Vasicek

No modelo de Vasicek de um fator, a taxa curta é modelada como um único fator estocástico (WEIMING):

$$dr(t) = K(\theta - r(t))dt + \sigma dW(t)$$

Aqui, K, θ, e σ são constantes, e σ é o desvio padrão instantâneo. $W(t)$ é o processo estocástico de Wiener. O modelo de Vasicek segue um processo de Ornstein-Uhlenbeck, onde o modelo reverte em torno da média θ com K, a velocidade de reversão à média. Como resultado, as taxas de juros podem se tornar negativas, o que é uma propriedade indesejável na maioria das condições econômicas normais.

Para ajudar no entendimento desse modelo, o seguinte código Python gera uma lista de taxas de juros.

```
In [4]: def vasicek(r0, K, theta, sigma, T=1., N=10, seed=777):
            np.random.seed(seed)
            dt = T/float(N)
            rates = [r0]
            for i in range(N):
                dr = K*(theta-rates[-1])*dt + sigma*np.random.normal()
                rates.append(rates[-1] + dr)
            return range(N+1), rates
```

Figura 16: Implementação do modelo de Vasicek em Python.

A função "vasicek" retorna uma lista de períodos de tempo e taxas de juros do modelo de Vasicek. Ela recebe como parâmetros de entrada: $r0$ é a taxa inicial de juros em $t = 0$; K, θ e σ são constantes; T é o período em termos de número de anos; N é o número de intervalos para o processo de modelagem; e "*seed*" é o valor de inicialização para o gerador de números aleatórios normais padrão do NumPy.

Assumindor que a taxa de juros atual é de 1,875%, K é 0,2, θ é 0,01 e σ é 0,012. Usando um valor de T de 10 e um valor de N de 200 para modelar as taxas de juros da seguinte forma:

```
In [5]: x, y = vasicek(0.01875, 0.20, 0.01, 0.012, 10., 200)
```

Figura 17: Implementação de exemplo utilizando modelo de Vasicek.

Plotando um gráfico os resultados obtidos ao implementar o modelo de Vasicek como exposta na Figura 17, temos:

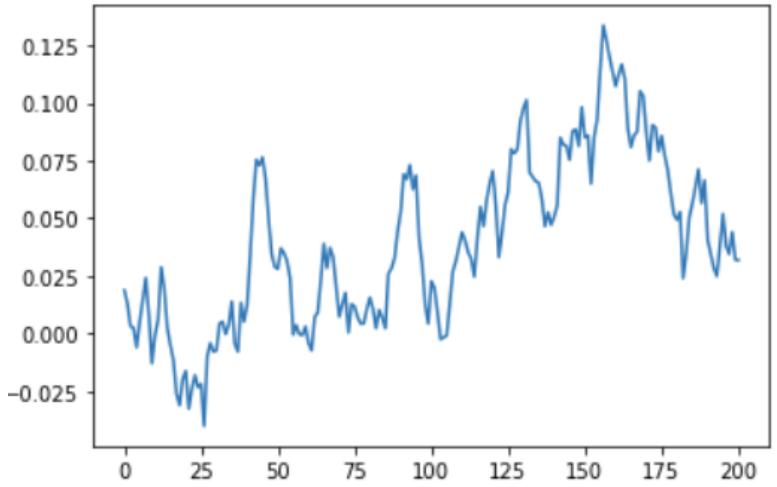

Figura 18:Visualização gráfica dos resultados do exemplo exposto na Figura 17.

Neste exemplo, será executado apenas uma simulação para observar como serão as taxas de juros do modelo de Vasicek. Como observado, as taxas de juros se tornaram negativas em algum ponto e cresceram, em média, a 0,01.

3.2. Modelo de Cox-Ingersoll-Ross

O modelo de Cox-Ingersoll-Ross (CIR) é um modelo de um fator proposto para lidar com as taxas de juros negativas encontradas no modelo de Vasicek (WEIMING). O processo é dado por:

$$dr(t) = K(\theta - r(t))dt + \sigma\sqrt{r(t)}W(t)$$

O termo $\sqrt{r(t)}$ aumenta o desvio padrão à medida que a taxa de juros curta aumenta. Agora, a função “vasicek” pode ser reescrita como o modelo CIR em Python:

```
In [9]: import math
        import numpy as np

        def cir(r0, K, theta, sigma, T=1.,N=10,seed=777):
            np.random.seed(seed)
            dt = T/float(N)
            rates = [r0]
            for i in range(N):
                dr = K*(theta-rates[-1])*dt + sigma*math.sqrt(rates[-1])*np.random.normal()
                rates.append(rates[-1] + dr)
            return range(N+1), rates
```

Figura 19: Implementação do modelo CIR em Python.

Usando o mesmo exemplo dado na seção do modelo de Vasicek, suponha que a taxa de juros atual seja de 1,875%, K é 0,2, θ é 0,01 e σ é 0,012. Usando um valor de T de 10 e um valor de N de 200 para modelar as taxas de juros da seguinte forma:

```
In [10]: x, y = cir(0.01875, 0.20, 0.01, 0.012, 10., 200)
```

Figura 20: Implementação de exemplo utilizando modelo CIR.

Plotando um gráfico os resultados obtidos ao implementar o modelo CIR como exposto na Figura 20, temos:

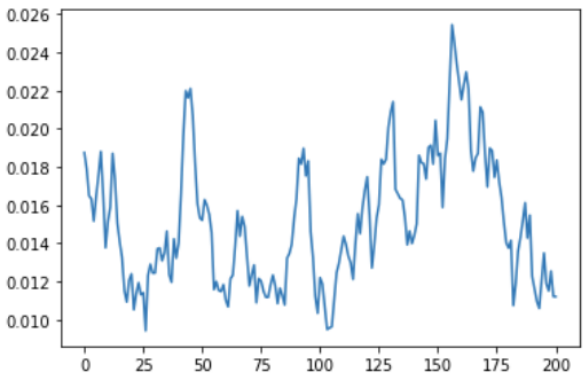

Figura 21: Visualização gráfica dos resultados do exemplo exposto na Figura 20.

Observe que o modelo de juros CIR não apresenta valores de taxas de juros negativas.

3.3. Modelo de Rendleman and Bartter

No modelo de Rendleman e Bartter, o processo de taxa de juros curta é dado por (WEIMING):

$$dr(t) = \theta r(t)dt + \sigma r(t)W(t)$$

Aqui, o desvio instantâneo é $\theta r(t)$, com um desvio padrão instantâneo $\sigma r(t)$. O modelo de Rendleman e Bartter pode ser considerado como um movimento Browniano geométrico, semelhante a um processo estocástico de preço de ações que é distribuído log-normalmente. Este modelo não possui a propriedade de reversão à média. A reversão à média é um fenômeno em que as taxas de juros parecem ser "puxadas" de volta para um nível médio de longo prazo.

O seguinte código Python modela o processo de taxa de juros de Rendleman e Bartter:

```
In [12]: import numpy as np

         def rendleman_bartter(r0, theta, sigma, T=1.,N=10,seed=777):
             np.random.seed(seed)
             dt = T/float(N)
             rates = [r0]
             for i in range(N):
                 dr = theta*rates[-1]*dt + sigma*rates[-1]*np.random.normal()
                 rates.append(rates[-1] + dr)
             return range(N+1), rates
```

Figura 22: Implementação do modelo de Rendleman e Bartter em Python.

Vamos continuar usando o exemplo das seções anteriores e comparar os modelos. Suponha que a taxa de juros atual seja de 1,875%, K é 0,2, θ é 0,01 e σ é 0,012. Usando um valor de T de 10 e um valor de N de 200 para modelar as taxas de juros da seguinte forma:

```
In [13]: x, y = rendleman_bartter(0.01875, 0.01, 0.012, 10., 200)
```

Figura 23: Implementação de exemplo utilizando modelo de Rendleman e Bartter.

Plotando um gráfico dos resultados obtidos ao implementar o modelo de Rendleman e Bartter como exposto na Figura 20, temos:

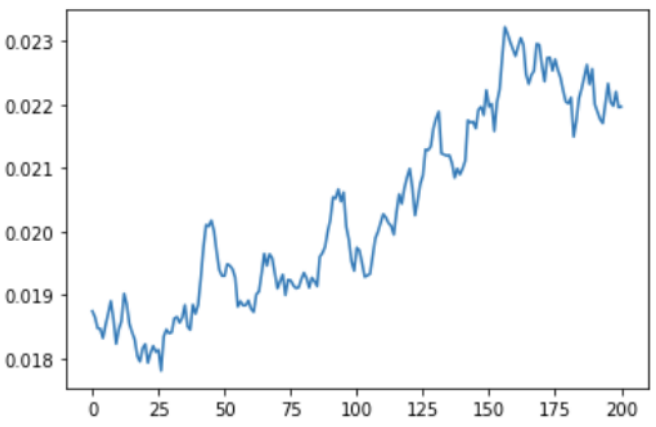

Figura 24: Visualização gráfica dos resultados do exemplo exposto na Figura 23.

Em geral, este modelo não possui a propriedade de reversão à média e cresce em direção a um nível médio de longo prazo.

3.4. Modelo de Brennan e Schwartz

O modelo de Brennan e Schwartz é um modelo de dois fatores em que a taxa de juros curta reverte para uma taxa longa como média, que também segue um processo estocástico. O processo de taxa curta é dado por (WEIMING):

$$dr(t) = K(\theta - t(t))dt + \sigma r(t)dW(t)$$

Pode-se ver que o modelo de Brennan e Schwartz é outra forma de um movimento Browniano geométrico. Dessa forma, o seguinte código em Python pode ser implementado:

```
In [15]: import numpy as np

def brennan_schwartz(r0, K, theta, sigma, T=1., N=10, seed=777):
    np.random.seed(seed)
    dt = T/float(N)
    rates = [r0]
    for i in range(N):
        dr = K*(theta-rates[-1])*dt + sigma*rates[-1]*np.random.normal()
        rates.append(rates[-1] + dr)
    return range(N+1), rates
```

Figura 25: Implementação do modelo de Brennan e Schwartz em Python.

Suponha que a taxa de juros atual seja de 1,875%, K é 0,2, θ é 0,01 e σ é 0,012. Usando um valor de T de 10 e um valor de N de 10.000 para modelar as taxas de juros da seguinte forma:

```
In [16]: x, y = brennan_schwartz(0.01875, 0.20, 0.01, 0.012, 10.,10000)
```

Figura 26: Implementação de exemplo utilizando modelo de Brennan e Schwartz..

Plotando um gráfico dos resultados obtidos ao implementar o modelo de Brennan e Schwartz como exposto na Figura 26, temos:

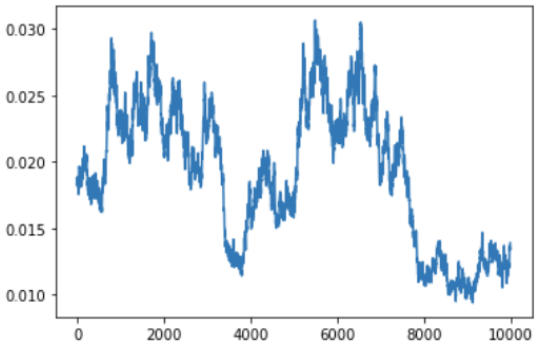

Figura 27: Visualização gráfica dos resultados do exemplo exposto na Figura 26.

4. Resultados e Conclusões

Os modelos de Vasicek, Cox-Ingersoll-Ross (CIR), Rendleman e Bartter, e Brennan e Schwartz são abordagens distintas para modelar o comportamento das taxas de juros. O modelo de Vasicek é baseado em um processo de Ornstein-Uhlenbeck, onde a taxa curta reverte para uma média de longo prazo com uma velocidade de reversão constante. No entanto, uma desvantagem do modelo é a possibilidade de taxas de juros negativas, o que pode ser indesejável em certas situações.

O modelo de Cox-Ingersoll-Ross (CIR) foi proposto como uma melhoria do modelo de Vasicek para evitar taxas de juros negativas. Ele incorpora uma dinâmica mais realista, onde a volatilidade é uma função do nível atual da taxa de juros. Isso proporciona uma melhor adaptação às características observadas no mercado real.

Por outro lado, o modelo de Rendleman e Bartter adota uma abordagem diferente, assemelhando-se a um movimento Browniano geométrico. Ao contrário dos modelos anteriores, ele não possui uma propriedade de reversão à média, o que significa que as taxas de juros não tendem a retornar a um nível médio ao longo do tempo.

O modelo de Brennan e Schwartz é um modelo de dois fatores, onde a taxa curta é influenciada por uma taxa longa estocástica. Isso introduz uma componente adicional de complexidade, permitindo que a taxa curta reverta para uma média de longo prazo que também segue um processo estocástico. Essa abordagem pode ser mais adequada para capturar nuances adicionais no comportamento das taxas de juros em comparação com modelos de fator único.

A implementação computacional dos códigos foi essencial para melhor compreensão dos modelos, além da visualização gráfica dos mesmos. O auxílio das bibliotecas computacionais *numpy* e *matplotlib* foi imprescindível para implementação computacional dos modelos, vistos os cálculos associados a cada um dos modelos.

5. Referências

HULL, John C. **Opções, futuros e outros derivativos**. Bookman Editora, 2016.

WEIMING, James Ma. **Mastering Python for Finance**. Packet Publishing 2015.