XXXIII Congresso de Iniciação Científica da Unicamp

Calibração dos parâmetros do Modelo de Heston por Monte Carlo

Palavras-Chave: Modelo de Heston, Simulação de Monte Carlo, Cálculo Estocástico

Autores(as):

Clayton Alves Luiz, IMECC – UNICAMP

Prof(ª). Dr(ª). Pedro José Catuogno, IMECC – UNICAMP

Introdução

O Modelo de Heston foi desenvolvido por Steven L. Heston e apresentado em seu artigo "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options", publicado em 1993. Diferentemente do modelo de Black-Scholes, que assume uma volatilidade constante, o Modelo de Heston incorpora uma volatilidade estocástica, oferecendo uma representação mais realista do comportamento dos mercados financeiros.

Matematicamente, o modelo é descrito por um sistema de duas equações diferenciais estocásticas:

$$\begin{cases} dS_t = \mu S_t dt + \sqrt{V_t} S_t dW_t^s \\ dv_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t} dW_t^v \end{cases} \tag{1}$$

Onde dW_t^s e dW_t^s tem correlação ρ.

Este trabalho teve como objetivo utilizar dados reais de uma empresa — neste caso, a Apple — para simular numericamente os valores da ação e da volatilidade até um tempo T e, por fim, calibrar os parâmetros do modelo por meio da minimização do erro.

Metodologia

Parâmetros

Os parâmetros da equação (1) possuem interpretações relevantes. O parâmetro μ representa a taxa de juros neutra ao risco, geralmente associada à taxa básica de juros da economia. Neste projeto, foi utilizada a taxa de juros dos Estados Unidos, assim, $\mu = 0{,}0425$. O parâmetro κ é conhecido como a taxa de reversão à média, enquanto θ representa a média de longo prazo da volatilidade. Já ξ está associado à intensidade da aleatoriedade na dinâmica da volatilidade, sendo responsável pela parte difusiva de v_t.

Outro ponto importante é a condição de Feller, a qual será utilizada posteriormente para sortear os parâmetros. Essa condição garante que v_t permaneça sempre positiva, sendo satisfeita quando $2\kappa\theta > \xi^2$.

Além disso, destaca-se que a volatilidade inicial v_0 é também um parâmetro a ser estimado, ao contrário de S_0, cujo valor inicial pode ser facilmente obtido a partir do mercado financeiro.

Euler-Maruyama

Este foi o método numérico utilizado para simular o modelo de Heston, dado que se trata de uma equação diferencial estocástica da forma:

$$dX_t = f(X_t)dt + g(X_t)dW_t,$$

onde o tempo total T pode ser discretizado em n intervalos de tamanho δt, de modo que:

$$X_{i+1} = X_i + f(X_i)\delta t + g(X_i)\delta W_i,$$

em que δW_i é o incremento do movimento browniano com distribuição normal de média zero e variância δt, isto é, $\delta W_i \sim \mathcal{N}(0, \delta t)$.

Com este método, é possível simular uma aproximação da solução da equação diferencial estocástica.

Outra forma para o dS

Utilizando o Lema de Itô na equação do ativo, com a variável $u = \ln S_t$, temos:

$$d(\ln S_t) = \left(\mu - \frac{1}{2}v_t\right)dt + \sqrt{v_t}\,dW_t \tag{2}$$

Esta fórmula depende de $\ln S_t$. Por meio do método de Euler-Maruyama, podemos reescrevê-la da seguinte forma:

$$\ln S_{i+1} = \ln S_i + \delta t\left(\mu - \frac{1}{2}v_i\right) + \sqrt{v_i}\,\delta W_i^s$$

Logo,

$$S_{i+1} = S_i \exp\left(\delta t\left(\mu - \frac{1}{2}v_i\right) + \sqrt{v_i}\,\delta W_i^s\right) \tag{3}$$

Com a equação (3), encontramos uma maneira de simular facilmente essa equação diferencial, pois basta discretizar o período de tempo T em n partes de tamanho δt, gerar números aleatórios normalmente distribuídos com variância δt e iterar passo a passo.

Simulando alguns caminhos da ação e da volatilidade

Como já temos a equação (3), é preciso escrever a volatilidade utilizando o método de Euler-Maruyama, logo:

$$v_{i+1} = v_i + \kappa(\theta - v_i)\delta t + \xi\sqrt{v_i}\,\delta W_i^v \tag{4}$$

Por fim, é necessário criar duas variáveis normais com correlação ρ. Para isso, geramos uma variável aleatória normal N_1 e uma variável aleatória N_2, definida pela seguinte relação:

$$N_2 = \rho N_1 + \sqrt{1-\rho^2}\,Z, \tag{5}$$

onde Z é uma variável aleatória normal independente de N_1, com variância δt.

A partir das equações (3), (4) e (5), é possível simular o caminho de uma ação com sua volatilidade estocástica. A seguir, apresento duas simulações: uma representa a trajetória do preço da ação e a outra, a curva da volatilidade correspondente. Para esta simulação, o intervalo T foi dividido em 5000 partes.

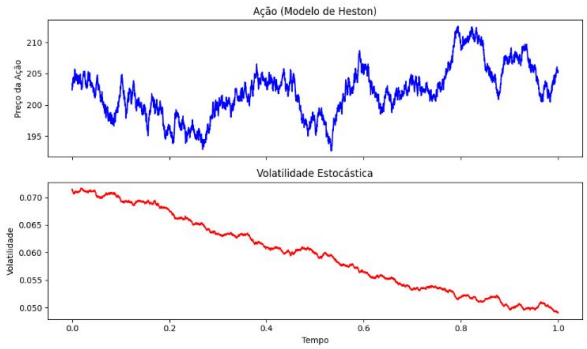

Figura 1: Simulações de curvas por Heston

Para esta simulação foram sorteados parâmetros aleatórios. Abaixo deixo o código escrito em python para simular estas duas curvas.

```python
def modelo_heston(kappa,theta,xi,mi,v_0,s_0,rho,k=5000,n=5000):
  # n é número de passos para cada simulação
  #k é o número de caminhos simulados ao mesmo tempo
  T = 55/252  #Período de tempo em anos,pois 252 é o número de dias úteis no ano
  delta_t = T/n
  S = np.zeros((k,n+1)) #Vou guardar os valores da ação em uma matriz (k,n+1)
  V = np.zeros((k,n+1)) #vou guardar os valores da volatilidade
  S[:,0] = s_0 #inicia a primeira coluna com o valor inicial da ação
  V[:,0] = v_0 #inicia a primeira coluna com o valor inicial da volatilidade
  #Com essa função gero uma matriz com valores de variáveis aleatórias normais com variancia delta_t
  N_1 = np.sqrt(delta_t)*np.random.normal(size=(k,n))
  z = np.sqrt(delta_t)*np.random.normal(size=(k,n))
  N_2 = rho*N_1 + np.sqrt(1-rho**2)*z #Com esssa parte gero o segundo movimento browniano correlacionado

  for j in range(n):
    #Agora vou gerar as iterações para resolver a EDE
    sqrt_V = np.sqrt(np.maximum(V[:, j], 1e-8)) #medida de segurança para garantir estabilidade
    S[:,j+1] = S[:,j]*np.exp(delta_t*(mi - 0.5*V[:,j]) + sqrt_V*N_1[:,j])
    #Garante que V nunca seja negativa e nem tão proxima de zero
    V[:,j+1] = np.maximum(V[:,j] + delta_t*kappa*(theta - V[:,j]) + xi*sqrt_V*N_2[:,j],1e-8)

  return S,V
```

Figura 2: Código para simular Heston por Euler-Maruyama

Monte Carlo

Com este código é possível precificar opções. Para isso, simularemos o caminho de uma ação 500 vezes utilizando o mesmo conjunto de parâmetros. Em seguida, calcularemos o payoff, definido por:

$$\max(S_T - K, 0),$$

onde S_T é o preço da ação no tempo T e K é o strike correspondente. Entretanto, uma única simulação não é suficiente, pois representaria apenas uma possibilidade dentre inúmeras. Assim, realizaremos 500 simulações e calcularemos o payoff de cada uma. Após isso, traremos a média dos payoffs a valor presente, descontando a taxa de juros μ, e tentaremos minimizar a diferença em relação ao valor atual de mercado, expressa por:

$$\min |C - C_0|, \tag{6}$$

onde

$$C_0 = e^{-\mu T}\mathbb{E}[\max(S_T - K, 0)]$$

é o valor teórico da opção e C é o valor de mercado atual. Podemos interpretar essa diferença como o erro associado ao modelo, ou seja,

$$\text{erro} = |C - C_0|.$$

Além disso, é importante destacar que C, K, μ e T são dados conhecidos. No nosso problema, $T \approx 0{,}218$ anos, considerando que o título escolhido possui vencimento em 17/10/2025 e são contabilizados apenas os dias úteis. Para esta data, utilizaremos $K = 100{,}0$ e $C = 110{,}34$, valor de fechamento em 01/08/2025.

A seguir, realizaremos o sorteio de novos parâmetros para tentar encontrar o conjunto que minimize essa diferença. E seguir fazendo esses sorteios até que alcancemos a tolerância desejada ou o máximo de iterações.

Abaixo estão as funções para o sorteio de parâmetros, cálculo do erro e a função de Monte Carlo, todas escritas em Python.

```python
def erro_heston(S_T,strike,lastprice,mi):
  T = 55/252 #tempo até o vencimento
  payoff = np.maximum(S_T - strike,0)
  payoff_medio = np.mean(payoff)
  C_0 = np.exp(-mi*T)*payoff_medio #Aqui trago os valores a valor presente
  erro =abs(C_0 - lastprice)  #calculo o erro
  return erro
```

```python
#Vou criar um breve código para sortear os parametros
def parametros_sorteio():
  while True:
    kappa = np.random.uniform(0,10)
    theta = np.random.uniform(0.01,0.2)
    xi = np.random.uniform(0.01,0.5)
    if 2*kappa*theta > xi**2:
      v_0 = np.random.uniform(0.01,0.15)
      rho = np.random.uniform(-1,0)
      break
  return kappa,theta,xi,v_0,rho
```

Figura 3: Códigos para calcular o erro e sortear os parâmetros

```python
#Monte Carlo
N = 1000 #número de iterações,isto é o número de vezes que sorteio os parâmetros
#Lembrando que o caminho é discretizado em 5000 passos
mi = 0.0425
erro = 1e-1 #critério de parada (usei uma constante)
melhor_erro = np.inf
for i in range(N):
  #Sortear os parametros utilizados
  kappa_i,theta_i,xi_i,v_i,rho_i = parametros_sorteio()
  #Calcular os camihos
  S,_ = modelo_heston(kappa_i,theta_i,xi_i,mi,v_i,preco_acao,rho_i,k=500)
  S_T = S[:,-1]#Separo a ultima coluna
  erro_i = erro_heston(S_T,strike,lastprice,mi) #Erro associado aos parametros sorteados
  #Comparo os erros
  if erro_i < melhor_erro:
    melhor_erro = erro_i
    kappa = kappa_i
    theta = theta_i
    xi = xi_i
    v_0 = v_i
    rho = rho_i
  #Caso encontre um erro que atenda meu critério eu paro
  if melhor_erro< erro:
    break
```

Figura 4: Algoritmo de monte Carlo

Resultados e Discussão

Para calcular o valor da opção, utilizei a equação (6) e, para cada conjunto de parâmetros sorteados, realizei 500 simulações do caminho da ação, partindo de $S_0 \approx 202{,}38$. Os parâmetros foram sorteados 1000 vezes, e o erro mínimo encontrado foi aproximadamente 2,9892 , com os seguintes valores associados: $\kappa = 1{,}4923$, $\theta = 0{,}1296$, $\xi = 0{,}2140$, $\rho = 0{,}9680$ e $v_0 = 0{,}1054$.

Embora o erro tenha ficado um pouco maior do que o esperado, ainda assim pode-se considerar que os parâmetros foram calibrados adequadamente, pois a diferença representa poucos reais. Como o método de Monte Carlo depende do número de iterações, espera-se que o aumento destas leve à redução do erro, melhorando a calibração do modelo. Entretanto, isso também aumentaria o tempo de simulação.

Por esse motivo, acredito que a combinação do método de Monte Carlo com outras técnicas numéricas pode ser uma abordagem interessante — por exemplo, utilizar Monte Carlo para determinar a população inicial em outros métodos estocásticos.

Conclusão

Embora o método de Monte Carlo seja eficaz para reduzir o erro, ele apresenta desempenho insatisfatório, visto que foram necessárias cerca de seis horas para executar as 1000 iterações da simulação, o que configura uma limitação prática.

Por outro lado, foi possível calibrar o modelo utilizando um método relativamente simples, o que é bastante interessante para fins didáticos, especialmente porque envolve a aplicação de diversos conceitos importantes, como movimento browniano, lema de Itô, método de Euler-Maruyama, entre outros.

Referências

[1] Heston, S. L. (1993). *A closed-form solution for options with stochastic volatility with applications to bond and currency options*. The Review of Financial Studies, 6(2), 327–343.

[2] Peter E. Kloeden and Eckhard Platen. *An Introduction to the Numerical Simulation of Stochastic Differential Equations*. Springer, Lecture Notes in Mathematics, vol. 1719, 2002.