Implementing Bjerksund and Stensland Approximation for American Option in Python

AARON DE LA ROSA

The Bjerksund-Stensland approximation is a closed-form model used to price American options. Developed by Petter Bjerksund and Gunnar Stensland in 1993, this model is particularly useful for American options, which can be exercised at any time before expiration, unlike European options that can only be exercised at maturity.

Here’s a brief overview of how the Bjerksund-Stensland model works:

Flat Exercise Boundary: The model assumes a flat early exercise boundary, which simplifies the complex problem of determining the optimal exercise strategy for American options.

Two Periods: It divides the time to maturity into two periods, each with its own flat exercise boundary.

Closed-Form Solution: By imposing a feasible but non-optimal exercise strategy, the model provides a closed-form solution that serves as a lower bound to the true option value.

Efficiency: This approach is computationally efficient and provides accurate approximations for the value of American call and put options.

The Bjerksund-Stensland model is particularly advantageous for its speed and efficiency in calculating option prices, making it a popular choice among practitioners in quantitative finance.

The Bjerksund-Stensland model stands out among American option pricing models due to its unique approach and specific advantages. Here’s a comparison with some other popular models:

1. Binomial Tree Model

Approach: The binomial tree model uses a discrete-time framework to model the underlying asset’s price movements. It constructs a tree of possible price paths and evaluates the option’s value at each node.

Complexity: It can be computationally intensive, especially for options with long maturities or high volatility.

Flexibility: Highly flexible and can handle various types of options and payoffs.

Accuracy: Provides accurate results but requires a large number of time steps for convergence.

2. Black-Scholes Model with Adjustments

Approach: The Black-Scholes model is originally designed for European options. For American options, adjustments like the Barone-Adesi and Whaley approximation are used.

Complexity: Less complex than the binomial tree but still requires numerical methods for American options.

Flexibility: Limited to simpler options and payoffs.

Accuracy: Provides good approximations but may not be as accurate for options with complex features.

3. Longstaff-Schwartz Model (Least-Squares Monte Carlo)

Approach: Uses Monte Carlo simulation to model the underlying asset's price paths and employs regression techniques to estimate the optimal exercise strategy.

Complexity: Computationally intensive due to the need for numerous simulations and regressions.

Flexibility: Highly flexible and can handle complex options and payoffs.

Accuracy: Very accurate, especially for options with complex features.

4. Bjerksund-Stensland Model

Approach: Uses a closed-form approximation with a flat early exercise boundary, dividing the time to maturity into two periods.

Complexity: Less complex and computationally efficient due to its closed-form nature.

Flexibility: Limited to simpler options and payoffs.

Accuracy: Provides a lower bound approximation, which is generally accurate for standard American options but may not be as precise for options with complex features.

Key Differences

Computational Efficiency: The Bjerksund-Stensland model is more efficient than the binomial tree and Longstaff-Schwartz models due to its closed-form solution.

Accuracy: While the Bjerksund-Stensland model provides a good approximation, it may not be as accurate as the Longstaff-Schwartz model for complex options.

Flexibility: The binomial tree and Longstaff-Schwartz models offer greater flexibility in handling various types of options and payoffs compared to the Bjerksund-Stensland model.

Bjerksund-Stensland Closed American Option Pricing Model

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se editó por última vez: 28 de septiembre

Bjerksund-Stensland Closed American Option Pricing Model

```python
import numpy as np
import scipy.stats as si
from scipy.stats import multivariate_normal as mvn
import pandas as pd
```

```python
##############
#Phi functions#
##############

def expec(S,T,sigma,b,Y,H,X, rf):
    """
    Phi function for flat-boundary implementation
    """
    lambdas= -rf + Y*b+0.5*Y*(Y-1)*(sigma)**2
    k = ((2*b)/(sigma**2)) + (2*Y-1)

    d1= - (np.log(S / H) + ( b + (Y- 0.5 )* sigma**2) * T) / (sigma * np.sqrt(T))
    Nd1 =si.norm.cdf(d1, 0.0, 1.0)

    d2= - (np.log(X**2 /(S* H)) + ( b + (Y- 0.5 )* sigma**2) * T) / (sigma * np.sqrt(T))
```

Bjerksund-Stensland Closed American Option Pricing Model

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se editó por última vez: 28 de septiembre

```python

    expec=(np.exp(lambdas*T)*(S**Y))*(Nd1 - ((X/S)**k)*Nd2)
    return expec

def expec2(S,T,sigma,b,Y,H,X,x,t,rf):
    """
    Phi function for 2-step implementation
    """
    d1= - (np.log(S / x) + ( b + (Y- 0.5 )* sigma**2) * t) / (sigma * np.sqrt(t))
    d2= - (np.log(X**2 / (S*x)) + ( b + (Y- 0.5 )* sigma**2) * t) / (sigma * np.sqrt(t))
    d3= - (np.log(S / x) - ( b + (Y- 0.5 )* sigma**2) * t) / (sigma * np.sqrt(t))
    d4= - (np.log(X**2 / (S*x)) - ( b + (Y- 0.5 )* sigma**2) * t) / (sigma * np.sqrt(t))
    D1 =  - (np.log(S / H) + ( b + (Y- 0.5 )* sigma**2) * T) / (sigma * np.sqrt(T))
    D2 = - (np.log(X**2 /(S* H)) + ( b + (Y- 0.5 )* sigma**2) * T) / (sigma * np.sqrt(T))
    D3= - (np.log(x**2 /(S* H)) + ( b + (Y- 0.5 )* sigma**2) * T) / (sigma * np.sqrt(T))
    D4= - (np.log(S*x**2 /(H*(X**2))) + ( b + (Y- 0.5 )* sigma**2) * T) / (sigma * np.sqrt(T))

    Corr= np.sqrt(t/T)
    dist = mvn(mean=np.array([0,0]), cov=np.array([[1, Corr],[Corr, 1]]))
    dist2 = mvn(mean=np.array([0,0]), cov=np.array([[1, -Corr],[-Corr, 1]]))

    lambdas= -rf + Y*b+0.5*Y*(Y-1)*sigma**2
    k= 2*b/sigma**2 + (2*Y-1)

    expec2 = np.exp(lambdas*T)*(S**Y)* \
```

Bjerksund-Stensland Closed American Option Pricing Model

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se editó por última vez: 28 de septiembre

Comentar Compartir

+ Código + Texto Conectar Gemini

```python
                ((x/S)**k)*dist2.cdf(np.array([d3,D3])) +  ((x/X)**k)*dist2.cdf(np.array([d4,D4])))#dist.cdf(np.array([D1,E1]))
    return expec2

##########################################################################################



#################################
# Bjerksund Stensland Model 2002#
#################################


def Bjerksund_Stensland(S,K,T,r,b,sigma,ndigits=4):
    """
    Put-Call Transformation: P(S,K,T,r,b,sigma) <=> C(K,S,T,r-b,-b,sigma)
    """
    beta = (0.5-(b/sigma**2)) + np.sqrt( ( (b/sigma**2)-0.5)**2+ 2*r/sigma**2)

    B0=max(K,(r/(r-b))*K)
    B8=(beta/(beta-1))*K
    h_T= -(b*T+2*sigma*np.sqrt(T))*((K**2)/((B8-B0)*B0))

    X=B0+(B8-B0)*(1-np.exp(h_T))

    if S>X: # Condición de ejercicio automático
        premium = [round(float(S-K),2), round(float(S-K),2), round(float(S-K),2)]
```

Bjerksund-Stensland Closed American Option Pricing Model

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se editó por última vez: 28 de septiembre

Comentar Compartir

+ Código + Texto Conectar Gemini

```python

        Call =alfa_X*(S**beta) -alfa_X*expec(S,T,sigma,b,beta,X,X,r)+expec(S,T,sigma,b,1,X,X,r) \
                -expec(S,T,sigma,b,1,K,X,r)-K*expec(S,T,sigma,b,0,X,X,r) +K*expec(S,T,sigma,b,0,K,X,r)

        t =0.5*(np.sqrt(5)-1)*T



        h_t= -(b*(T-t)+2*sigma*np.sqrt(T-t))*((K**2)/((B8-B0)*B0))

        x=B0+(B8-B0)*(1-np.exp(h_t))
        alfa_x=(x-K)*(x**(-beta))


        Call_c= alfa_X*(S**beta)-alfa_X*expec(S,t,sigma,b,beta,X,X,r) +expec(S,t,sigma,b,1,X,X,r)-expec(S,t,sigma,b,1,x,X,r) \
               -K*expec(S,t,sigma,b,0,X,X,r) +K*expec(S,t,sigma,b,0,x,X,r) \
                + alfa_x*expec(S,t,sigma,b,beta,x,X,r) -alfa_x*expec2(S,T,sigma,b,beta,x,X,x,t,r)\
                +expec2(S,T,sigma,b,1,x,X,x,t,r)-expec2(S,T,sigma,b,1,K,X,x,t,r)    \
                -K*expec2(S,T,sigma,b,0,x,X,x,t,r)  +K*expec2(S,T,sigma,b,0,K,X,x,t,r)

        premium = [round(Call, ndigits), round(Call_c, ndigits), round(2*Call_c-Call, ndigits)]
        return premium


###########################################################################################
```

Bjerksund-Stensland Closed American Option Pricing Model

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se editó por última vez: 28 de septiembre

Comentar Compartir

+ Código + Texto Conectar Gemini

```python


def binomialCall_Am(S,K,T,r,q,sigma,n):
    dt=T/n # time partitions
    u=np.exp(sigma*np.sqrt(dt))
    d=1/u
    p = (np.exp((r-q)*dt) - d) / (u-d)

    stockvalue = np.zeros((n+1,n+1))
    stockvalue[0,0] = S

    for i in range(1,n+1):
        stockvalue[i,0] = stockvalue[i-1,0]*u
        for j in range(1,n+1):
            stockvalue[i,j] = stockvalue[i-1,j-1]*d

    # binomial tree for option's value
    optionvalue=np.zeros((n+1,n+1))
    # 1. terminal nodes (at maturity)
    for i in range(n+1):
        optionvalue[n,i] = max(stockvalue[n,i]-K,0)
    # 2. intermediate nodes (at every dt interval)
    for i in range(n-1,-1,-1):
        for j in range(i+1):
            F1=np.exp(-r*dt)*(p*optionvalue[i+1,j]+(1-p)*optionvalue[i+1,j+1])
```

Bjerksund-Stensland Closed American Option Pricing Model

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se editó por última vez: 28 de septiembre

Comentar Compartir

+ Código + Texto Conectar Gemini

Call Aproximations

```python
# Parameter preparation for iterative valutaions
Spots = np.arange(80,130,10) #spot price
K = 100  #strike price
b = -.04  # drift rate: b = r-q
Ti = iter([0.25, 0.25, 0.25,0.50]) #time to maturity
ri = iter([0.08, 0.12, 0.08, 0.08]) #interest rate
qi = iter([0.12, 0.16, 0.12, 0.12]) # div. yield: q = r-b
sigmai = iter([0.20, 0.20, 0.40, 0.20]) #volatility of underlying asset
n = 250 # time partitions (binomial model)


call_premiums_bs = []
call_premiums_bin = []

for _ in range(4):
    r = next(ri)
    sigma = next(sigmai)
    T = next(Ti)
    q = next(qi)
    [call_premiums_bs.append(Bjerksund_Stensland(S,K,T,r,b,sigma,2)) for S in Spots]  # calls
    [call_premiums_bin.append(round(binomialCall_Am(S,K,T,r,q,sigma,n),2)) for S in Spots]
```

CO Bjerksund-Stensland Closed American Option Pricing Model ☆

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda Se editó por última vez: 28 de septiembre

Comentar Compartir

+ Código + Texto Conectar Gemini

Put Approximations (within Call procedure context)

```
[ ]
Spots = np.arange(80,130,10) #spot price
K = 100  #strike price
b = -.04  # drift rate
Ti = iter([0.25, 0.25, 0.25,0.50]) #time to maturity
ri = iter([0.08, 0.12, 0.08, 0.08]) #interest rate
sigmai = iter([0.20, 0.20, 0.40, 0.20]) #volatility of underlying asset
qi = iter([0.12, 0.16, 0.12, 0.12]) # div. yield: q = r-b
n = 250 # time partitions (binomial model)

put_premiums_bs = []
put_premiums_bin = []

for _ in range(4):
    r = next(ri)
    sigma = next(sigmai)
    T = next(Ti)
    q = next(qi)
    [put_premiums_bs.append(Bjerksund_Stensland(K,S,T,r-b,-b,sigma,2)) for S in Spots]  # call context valuations
    [put_premiums_bin.append(round(binomialPut_Am(S,K,T,r,q,sigma,n),2)) for S in Spots]


PutsBS = pd.DataFrame(put_premiums_bs, columns=['c_bar','c_2bar','2*c_2bar-c_bar'])
```

	C	c_bar	c_2bar	2*c_2bar-c_bar	P	c_bar	c_2bar	2*c_2bar-c_bar
0	0.03	0.03	0.03	0.03	20.41	20.41	20.41	20.41
1	0.58	0.57	0.58	0.58	11.25	11.25	11.25	11.25
2	3.52	3.49	3.51	3.54	4.39	4.40	4.40	4.40
3	10.36	10.32	10.34	10.37	1.12	1.12	1.12	1.12
4	20.00	20.00	20.00	20.00	0.18	0.18	0.18	0.18
5	0.03	0.03	0.03	0.03	20.23	20.22	20.23	20.23
6	0.58	0.57	0.57	0.58	11.14	11.14	11.14	11.14
7	3.50	3.46	3.49	3.51	4.35	4.35	4.35	4.35
8	10.33	10.29	10.31	10.34	1.11	1.11	1.11	1.11
9	20.00	20.00	20.00	20.00	0.18	0.18	0.18	0.18
10	1.06	1.05	1.05	1.06	21.44	21.44	21.44	21.45
11	3.27	3.25	3.26	3.27	13.92	13.91	13.91	13.92
12	7.40	7.37	7.39	7.42	8.26	8.27	8.27	8.27
13	13.53	13.47	13.51	13.54	4.52	4.52	4.52	4.52
14	21.29	21.23	21.26	21.28	2.29	2.29	2.29	2.29
15	0.21	0.21	0.21	0.21	20.96	20.95	20.96	20.96
16	1.36	1.34	1.35	1.36	12.63	12.63	12.63	12.63
17	4.71	4.65	4.69	4.73	6.36	6.37	6.37	6.37
18	11.00	10.94	10.98	11.01	2.65	2.65	2.65	2.65
19	20.00	20.00	20.00	20.00	0.92	0.92	0.92	0.92

Table 1: Approx. option values. Strike $K = 100$, cost of carry $b = -0.04$

Parameters:		American call				American put			
	$S =$	C	$\overline{c}$	$\overline{\overline{c}}$	$2\overline{\overline{c}} - \overline{c}$	P	$\overline{p}$	$\overline{\overline{p}}$	$2\overline{\overline{p}} - \overline{p}$
$r = 0.08$,	80	0.03	0.03	0.03	0.03	20.41	20.41	20.41	20.41
$\sigma = 0.20$,	90	0.58	0.57	0.58	0.58	11.25	11.25	11.25	11.25
$T = 0.25$	100	3.52	3.49	3.51	3.54	4.39	4.40	4.40	4.40
	110	10.36	10.32	10.34	10.37	1.12	1.12	1.12	1.12
	120	20.00	20.00	20.00	20.00	0.18	0.18	0.18	0.18
$r = 0.12$,	80	0.03	0.03	0.03	0.03	20.23	20.22	20.23	20.23
$\sigma = 0.20$,	90	0.57	0.57	0.57	0.58	11.14	11.14	11.14	11.14
$T = 0.25$	100	3.50	3.46	3.49	3.51	4.35	4.35	4.35	4.35
	110	10.32	10.29	10.31	10.34	1.11	1.11	1.11	1.11
	120	20.00	20.00	20.00	20.00	0.18	0.18	0.18	0.18
$r = 0.08$,	80	1.05	1.05	1.05	1.06	21.44	21.44	21.44	21.45
$\sigma = 0.40$,	90	3.26	3.25	3.26	3.27	13.92	13.91	13.91	13.92
$T = 0.25$	100	7.41	7.37	7.39	7.42	8.26	8.27	8.27	8.27
	110	13.52	13.47	13.51	13.54	4.52	4.52	4.52	4.52
	120	21.29	21.23*	21.26*	21.28	2.29	2.29	2.29	2.29
$r = 0.08$,	80	0.22	0.21	0.21	0.21	20.96	20.95	20.96	20.96
$\sigma = 0.20$,	90	1.36	1.34	1.35	1.36	12.63	12.63	12.63	12.63
$T = 0.5$	100	4.71	4.65*	4.69	4.73	6.37	6.37	6.37	6.37
	110	11.00	10.94*	10.98	11.01	2.65	2.65	2.65	2.65
	120	20.00	20.00	20.00	20.00	0.92	0.92	0.92	0.92

Notation: S: asset value; r: interest rate; σ: volatility; T: time to exercise.

C and P: Binomial call and put approximation with 3201 points on the lattice.

$\overline{c}$ and $\overline{p}$: Flat boundary call and put approximation.

$\overline{\overline{c}}$ and $\overline{\overline{p}}$: Two-step boundary call and put approximation.

$2\overline{\overline{c}} - \overline{c}$ and $2\overline{\overline{p}} - \overline{p}$: Call and put approximation using flat and two-step boundary results.

* Max. error from flat bdy. approx. 0.06; max. error from two-step bdy. approx. 0.03.