CEV1

Antoine Jacquier

TITLE:	CEV1
AUTHOR:	Antoine Jacquier
NUMBER OF PAGES:	8
FIRST VERSION:	February 13, 2017
CURRENT VERSION:	February 13, 2017
REVISION:	0.0.0

Contents

1 Pricing European Call options under the CEV model with $\beta = 1$

```
In [1]: from scipy.stats import norm
        import numpy as np
        import matplotlib.pyplot as plt
        from scipy.integrate import quad
        import pandas as pd
```

We consider here the Constant Elasticity of Volatility model $\mathrm{d}S_t = S_t^{1+\beta}\mathrm{d}W_t$, with $S_0 = S_0 > 0$, and where W is a one-dimensional standard Brownian motion. We are interested here in the particular case where $\beta = 1$. In this case, however, the stock price process is a strictly positive strict local martingale. We wish to study how this martingale defect impacts the price of European Calls and Puts.

2 European option pricing

2.1 Pricing with closed-form formulae

In the case $\beta = 1$ considered here, European Calls and Puts, defined as expected values of the final payoff (following Cox-Hobson's Theorem), admit closed-form representations:

$$\mathbb{E}(S_T - K)_+ = S_0\Big(\mathcal{N}(\kappa-\delta) - \mathcal{N}(-\delta) + \mathcal{N}(\delta) - \mathcal{N}(\kappa+\delta)\Big) - K\left(\mathcal{N}(\kappa+\delta) - \mathcal{N}(\delta-\kappa) + \frac{n(\kappa+\delta) - n(\kappa-\delta)}{\delta}\right),$$
$$\mathbb{E}(K - S_T)_+ = S_0 K\sqrt{T}\left(\zeta_+\mathcal{N}(\zeta_+) + n(\zeta_+) - \zeta_-\mathcal{N}(\zeta_-) - n(\zeta_-)\right),$$

where

$$\delta := \frac{1}{S_0\sqrt{T}}, \qquad \kappa := \frac{1}{K\sqrt{T}}, \qquad \zeta_\pm := \frac{1}{\sqrt{T}}\left(\pm\frac{1}{S_0} - \frac{1}{K}\right).$$

In [3]:
```
def n(x):
    return np.exp(-x * x / 2.) / np.sqrt(2. * pi)


def cev2Call_closed(S, K, T):
    delta = 1. / (S * np.sqrt(T))
    kappa = 1. / (K * np.sqrt(T))
    tPlus = n(kappa + delta)
    NMinus = norm.cdf(kappa - delta)
    NPlus = norm.cdf(kappa + delta)
    temp1 = NMinus - norm.cdf(-delta) + norm.cdf(delta) - NPlus
    temp2 = NPlus - (1. - NMinus) + \
        (n(kappa + delta) - n(kappa - delta)) / delta
    return S * temp1 - K * temp2


def cev2Put_closed(S, K, T):
    dp = (1. / S - 1. / K) / np.sqrt(T)
    dm = -(1. / S + 1. / K) / np.sqrt(T)
    return S * K * np.sqrt(T) * (dp * norm.cdf(dp) + n(dp) - dm * norm.cdf(dm) - n(dm))
```

2.2 Pricing by Put-Call parity

In [5]:
```
def cev2Call_pc(S, K, T):  # By PC Parity
    return S - K + cev2Put_closed(S, K, T)


# Computes the expectation of the stock price at time zero
```

```
def expectS(S, T):
    return S * (1. - 2. * norm.cdf(-1 / (S * np.sqrt(T))))
```

2.3 Pricing by integration

In the case $\beta = 1$ (as well as in the other cases for that matter), the density of the CEV is known in closed form:

$$\mathbb{P}(S_T \in \mathrm{d}z) = \frac{S_0}{z^3}\left\{\exp\left(-\frac{\left(z^{-1} - S_0^{-1}\right)^2}{2T}\right) - \exp\left(-\frac{\left(z^{-1} + S_0^{-1}\right)^2}{2T}\right)\right\}\frac{\mathrm{d}z}{\sqrt{2\pi T}}, \qquad \text{for all } z > 0.$$

```
In [6]: def cev2_density(z, s, T):
            temp1 = -(1. / z - 1. / s) * (1. / z - 1. / s) / (2. * T)
            temp2 = -(1. / z + 1. / s) * (1. / z + 1. / s) / (2. * T)
            return S0 * (np.exp(temp1) - np.exp(temp2)) / (z * z * z * np.sqrt(2. * pi * T))


        def cev2Put_int(s, K, T):
            def toIntegrate(z, s, K, T):
                return (K - z) * cev2_density(z, s, T)
            return quad(toIntegrate, 1.E-10, K, args=(s, K, T), epsabs=1.49e-15)[0]


        def cev2Call_int(s, K, T):
            def toIntegrate(z, s, K, T):
                return (z - K) * cev2_density(z, s, T)
            return quad(toIntegrate, K, 40. * s, args=(s, K, T), epsabs=1.49e-15)[0]
```

2.4 Numerical tests

```
In [7]: strikes = arange(0., 2., 0.1)
        S0, T = 1., 1.
        lowerBound = array([max(S0 - K, 0.) for K in strikes])
        upperBound = array([S0 for K in strikes])
```

```
In [8]: print "Expectation S_T= ", expectS(S0, T)
```

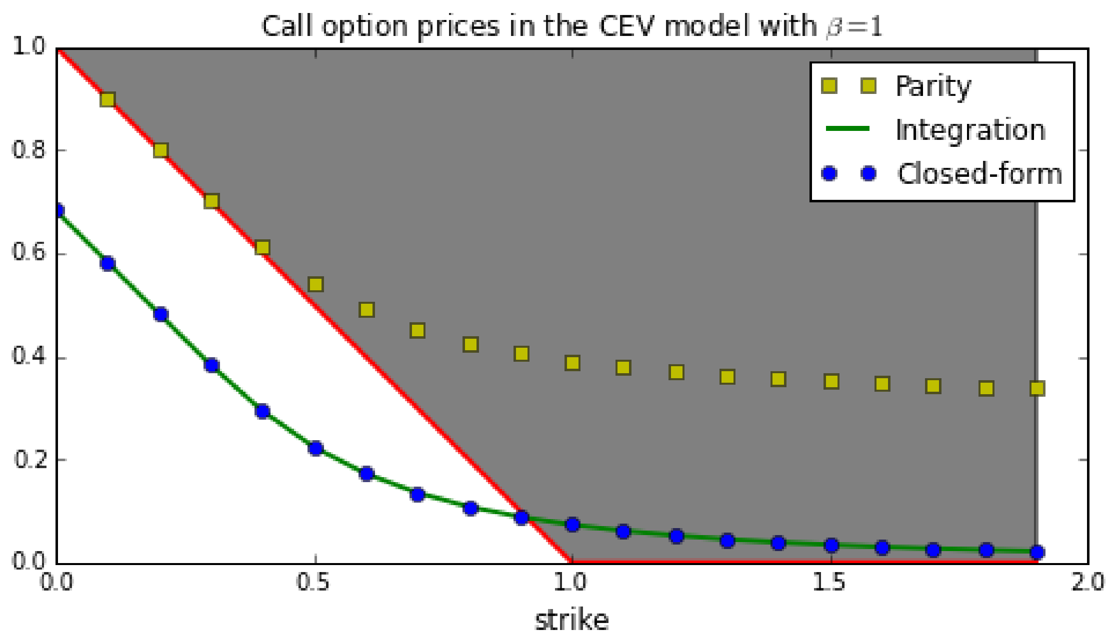
```
Expectation S_T=  0.682689492137
```

2.4.1 Call option

```
In [10]: Vcev2Call_pc = [cev2Call_pc(S0, K, T) for K in strikes]
         Vcev2Call_int = [cev2Call_int(S0, K, T) for K in strikes]
         Vcev2Call_closed = [cev2Call_closed(S0, K, T) for K in strikes]
```

```
plt.figure(figsize=(8, 4))
plt.plot(strikes, lowerBound, 'r', linewidth=2)
plt.plot(strikes, upperBound, 'r', linewidth=2)
plt.plot(strikes, Vcev2Call_pc, 'ys', label='Parity')
plt.plot(strikes, Vcev2Call_int, 'g-',
         label='Integration', linewidth=2)
plt.plot(strikes, Vcev2Call_closed, 'bo', label='Closed-form')
plt.fill_between(strikes, lowerBound, upperBound,
                 where=upperBound >= lowerBound, facecolor='grey', interpolate=True)
# lt.figtext(.95, .9, "The shaded area corresponds to the no-arbitrage region, \
# nwhile the red plot is its boundary £(S_0 - K)_+£", size=15)
plt.title(
    "Call option prices in the CEV model with " r"$\beta=1$")
plt.legend()
plt.xlabel(u'strike', fontsize=12)
print 'The shaded area corresponds to the no-arbitrage region, while the red plot is its bound
plt.show()
```

```
The shaded area corresponds to the no-arbitrage region, while the red plot is its boundary $(S_0 - K)_+$
```



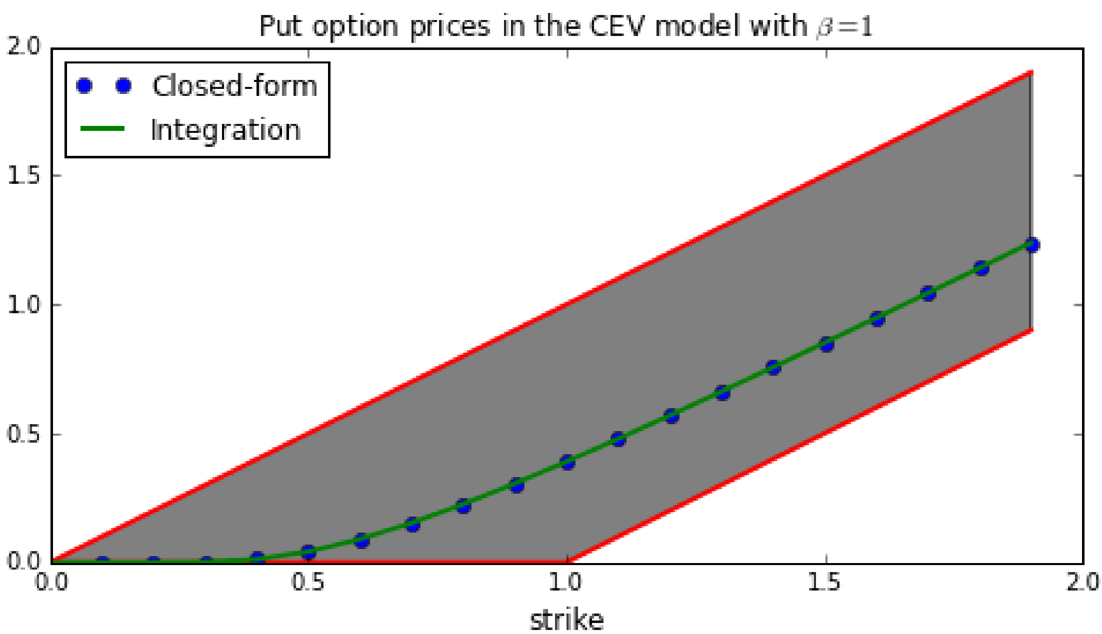
2.4.2 Put options

```
In [11]: lowerBound = array([max(K - S0, 0.0) for K in strikes])
         upperBound = array([K for K in strikes])
```

```
Vcev2Put_closed = [cev2Put_closed(S0, K, T) for K in strikes]
Vcev2Put_int = [cev2Put_int(S0, K, T) for K in strikes]

plt.figure(figsize=(8, 4))
plt.plot(strikes, lowerBound, 'r', linewidth=2)
plt.plot(strikes, upperBound, 'r', linewidth=2)
plt.plot(strikes, Vcev2Put_closed, 'bo', label='Closed-form')
plt.plot(strikes, Vcev2Put_int, 'g-',
         label='Integration', linewidth=2)
plt.fill_between(strikes, lowerBound, upperBound,
                 where=upperBound >= lowerBound, facecolor='grey', interpolate=True)
plt.title(
    "Put option prices in the CEV model with " r"$\beta=1$")
plt.legend(loc='upper left')
plt.xlabel(u'strike', fontsize=12)
print 'The shaded area corresponds to the no-arbitrage region, while the red plot is its bound
plt.show()
```

```
The shaded area corresponds to the no-arbitrage region, while the red plot is its boundary $(K-S_0)_+$
```



2.4.3 At-the money comparison: $K = S_0$

```
In [12]: myK = 1.
         myCol = ['Value']
```



```
myTable = pd.DataFrame.from_items([
    ('Put by closed form', [cev2Put_closed(S0, myK, T)]),
    ('Put by integration', [cev2Put_int(S0, myK, T)]),
    ('Call by integration', [cev2Call_int(S0, myK, T)]),
    ('Call by parity', [cev2Call_pc(S0, myK, T)]),
    ('Call by closed form', [cev2Call_closed(S0, myK, T)]),
    ('Expectation', [expectS(S0, T)])
], orient='index', columns=myCol)
print myTable
print "\nNote that by Put-Call Parity, \
the Call (evaluated by PC parity) and the Put have the same price"
```

```
Value
Put by closed form   0.390452
Put by integration   0.390452
Call by integration  0.072992
Call by parity       0.390452
Call by closed form  0.073141
Expectation          0.682689

Note that by Put-Call Parity, the Call (evaluated by PC parity) and the Put have the same price
```

2.4.4 Low-strike comparison

In [8]:
```
#################################################################
#################################################################
myK = 0.001
print "************************************"
print "************************************"
print "Strike = ", myK
myCol = ['Value']
myTable = pd.DataFrame.from_items([
    ('Put by closed form', [cev2Put_closed(S0, myK, T)]),
    ('Put by integration', [cev2Put_int(S0, myK, T)]),
    ('Call by integration', [cev2Call_int(S0, myK, T)]),
    ('Call by parity', [cev2Call_pc(S0, myK, T)]),
    ('Call by closed form', [cev2Call_closed(S0, myK, T)]),
    ('E(S_T)', [expectS(S0, T)])
], orient='index', columns=myCol)
print myTable
```

```
************************************
************************************
Strike =  0.001
```

	Value
Put by closed form	0.000000
Put by integration	0.000000
Call by integration	0.681538
Call by parity	0.999000
Call by closed form	0.681689
E(S_T)	0.682689

articleAndersen, author = Leif Andersen, title = Option pricing with quadratic volatility: a revisit, journal = Finance and Stochastics, year = 2011, volume = 15, issue = 2, pages = 191-219 @articleCoxHobson, author = Alex Cox and David Hobson, title = Local martingales, bubbles and option prices, journal = Finance and Stochastics, year = 2005, volume = 9, pages = 477-49