## Contents

0.1 Linear and quadratic interpolation of option prices ........................................ 2
  0.1.1 Black-Scholes functions ................................................................. 2
  0.1.2 SVI parameterisation ................................................................. 2
  0.1.3 Interpolation of option prices ....................................................... 2
  0.1.4 Numerical tests ................................................................. 4
0.1 Linear and quadratic interpolation of option prices

We illustrate here the influence of linear/quadratic interpolation of (European) option prices on the shape of the corresponding implied volatility smile.

0.1.1 Black-Scholes functions

In [1]: from math import log, sqrt, exp
   from scipy.stats import norm
   from scipy.optimize import bisect

0.1.2 SVI parameterisation

We wish to study here the influence, on the implied volatility smile, of linear and quadratic interpolation (in strike) of option prices. We generate a fixed number of option prices, for a given maturity, using the SVI parameterisation:

$$\text{SVI}(x) = a + b \left\{ \rho(x - m) + \sqrt{(x - m)^2 + \sigma^2} \right\}. $$

In [2]: def SVI(sviParams, x):
   
   # SVI parameterisation for the implied volatility sm
   # We return the square root!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
   a, b, rho, m, sigma = sviParams
   return sqrt(a + b * (rho * (x - m) + sqrt((x - m) * (x - m) + sigma * sigma)))

0.1.3 Interpolation of option prices

In [4]: def interpolationPricesVols(S, v, T, n, nb, minVal, maxVal, sviParams):
   
   prices, strikes, myStrikes, vols, volsSVI, pricesSVI, volsLin, volsQuad, pricesQuad, pricesLin = [], [], [], [], [], [], [], [], [], []
   counter = 1

   # Initialisation
   X = minVal
   strikes.append(X)
   vol = SVI(sviParams, log(X / S))
   P = BlackScholes(True, S, X, T, 0.0, 0.0, vol)
   prices.append(P)
   vols.append(vol)
# Loop 

```python
for i in range(1, n):
    X = minVal + 1.0 * i * (maxVal - minVal) / (1.0 * n)
    strikes.append(X)
    vol = SVI(sviParams, log(X / S))
    P = BlackScholes(True, S, X, T, 0.0, 0.0, vol)
    prices.append(P)
    vols.append(vol)
    if (i % 2 == 0):
        a = (prices[i] - prices[i - 2]) / (strikes[i] - strikes[i - 2])
        b = prices[i] - a * strikes[i]
    for j in range(1, nb):
        X = strikes[i - 2] + 1.0 * j * (strikes[i] - strikes[i - 2]) / (1.0 * nb)
        P = a * X + b
        myStrikes.append(X)
        myVol = impliedVolCore(
            True, 1.0, S, X, T, P, tolerance=1e-6, itermax=100)
        pricesLin.append(P)
        volsLin.append(myVol)

    # Price by quadratic interpolation
    P = prices[i - 2] * (X - strikes[i - 1]) * (X - strikes[i]) / ((strikes[i - 2] - strikes[i - 1]) * (strikes[i - 2] - strikes[i]))
    + prices[i - 1] * (X - strikes[i - 2]) * (X - strikes[i]) / ((strikes[i - 1] - strikes[i - 2]) * (strikes[i - 1] - strikes[i]))
    + prices[i] * (X - strikes[i - 2]) * (X - strikes[i - 1]) / ((strikes[i] - strikes[i - 2]) * (strikes[i] - strikes[i - 1]))
    myVol = impliedVolCore(
        True, 1.0, S, X, T, P, tolerance=1e-6, itermax=100)
    pricesQuad.append(P)
    volsQuad.append(myVol)
    volsSVI.append(SVI(sviParams, log(X / S)))
    pricesSVI.append(BlackScholes(True, S, X, T, 0.0, 0.0, SVI(sviParams, log(X / S))))
return myStrikes, pricesLin, pricesQuad, pricesSVI, volsLin, volsQuad, volsSVI
```
0.1.4 Numerical tests

In [7]: S, v, T = 100., 0.2, 1.
    n = 10  # has to be an even integer
    nb = 10  # number of interpolated points between two strikes
    
    minVal, maxVal = 40., 300.

    ########################### SVI parameters ###########################
    sviParams = 0.04, 0.4, -0.4, 0., 0.1

    strikes, pricesLin, pricesQuad, pricesSVI, volsLin, volsQuad, volsSVI = interpolationPricesVols(
        S, v, T, n, nb, minVal, maxVal, sviParams)

In [9]: plot(strikes, pricesLin, 'b', label="Linear interpolation")
    plot(strikes, pricesQuad, 'g', label="Quadratic interpolation")
    plot(strikes, pricesSVI, 'r+', label="True price")
    legend(loc=1)
    title("Linear and quadratic interpolations of option prices")
    show()

In [10]: plot(strikes, volsLin, 'b', label="Linear interpolation")
    plot(strikes, volsQuad, 'g', label="Quadratic interpolation")
plot(strikes, volsSVI, 'r+', label="True smile")
legend(loc=1)
title("Implied volatility outputs")
show()