Testing IV Slopes - Apple

Antoine Jacquier

Title:	Testing IV Slopes - Apple
Author:	Antoine Jacquier
Number of pages:	8
First version:	February 19, 2017
Current version:	February 19, 2017
Revision:	0.0.0

Contents

0.1 Apple implied volatility smiles: testing for Roger Lee's bounds

---

```
In [6]: import pandas as pd
        from zanadu.Groups.ImperialCollege.Root.Tools.BlackScholes import BlackScholes, ImpliedVol
        from zanadu.Groups.ImperialCollege.Root.Tools.ImpliedVolLeeLi import impliedVolCore
        import matplotlib.pyplot as plt
        from scipy.optimize import minimize
```

Options chain data from Google: https://www.google.co.uk/finance/option_chain?q=NASDAQ:AAPL

To calculate durations: http://www.timeanddate.com/date/durationresult.html?d1=27&m1=5&y1=2015&d2=19&m2=6&y

1 Choose the data file

```
In [8]: %reload_ext datamagic

        myfile = %getfile Imperial College / me / AppleOptionsChain3.csv
        spot, r, T, dateToday, dateMaturity = 129.62, 0.0, 23.0 / \
            365, "27 May 2015", "19 June 2015"
```

1.1 Sorting the data

```
In [9]: df = pd.read_csv(myfile)
        df = df.sort(["Strike"])  # sorting the strikes

        # print 'Columns of the file: %s'%list(df.columns)
        options, smiles, logMoneynesses = [], [], []


        # Sort all the strikes
        allStrikes = [df['Strike'][i] for i in range(len(df))]
        strikes = np.unique(allStrikes).tolist()
```

1.2 Sort the options chain

We only keep Calls or Puts with the largest liquidity, when the volume traded is not null

```
In [10]: volumeIntoAccount = True
```

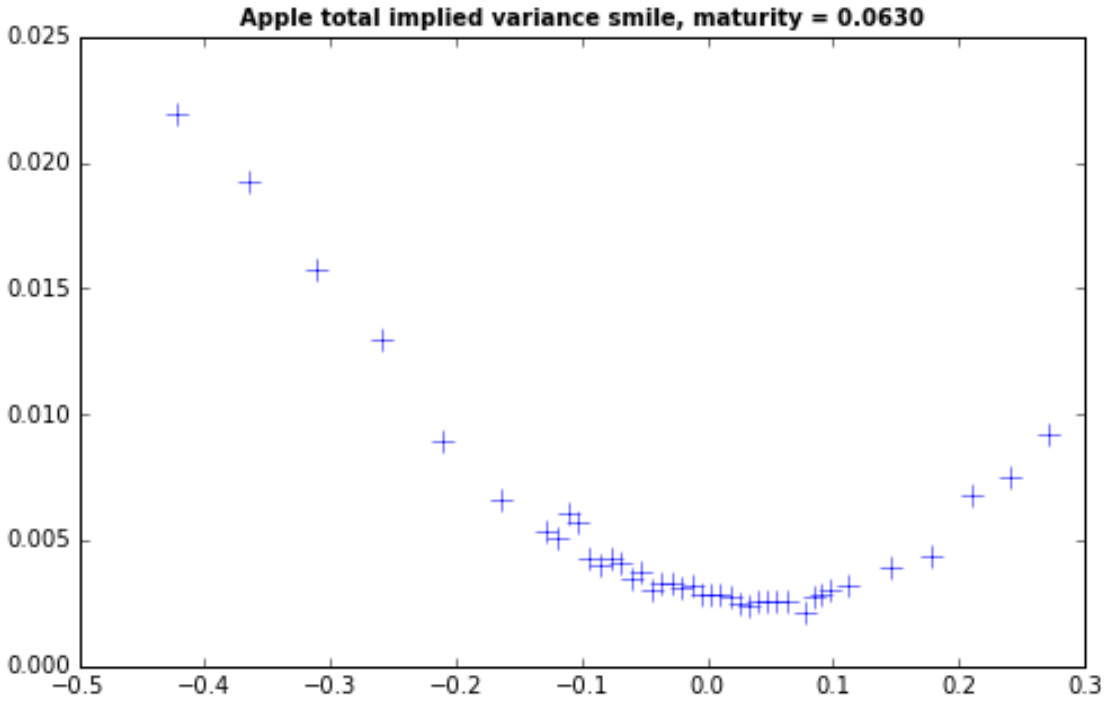
```
In [11]: if volumeIntoAccount:
             for strike in strikes:
                myList = []
                for i in range(len(df)):
                   if df['Volume'][i] <> "-" and df['Open Int'][i] <> "-" and df['Strike'][i] == strike:
                      myList.append(i)
                # only one price available at this strike
                if len(myList) == 1:
                   ix = myList[0]
                   # Price of the option
                   options.append(float(df['Price'][ix]))
                   # determines whether it is a Call or a Put
                   myBool = (df['Type'][ix] == "Call")
                   iv = impliedVolCore(myBool, 1.0, spot, df['Strike'][ix], T, float(
                       df['Price'][ix]), tolerance=1e-4, itermax=100)
                   logMoneynesses.append(log(df['Strike'][ix] / spot))
                   smiles.append(iv)
```

```
        # two prices available at this strike
        elif len(myList) == 2:
            # choose the strike with the highest traded volume
            if df['Volume'][myList[0]] > df['Volume'][myList[1]]:
                ix = myList[0]
            else:
                if df['Price'][myList[1]] == "-":
                    ix = myList[0]
                else:
                    ix = myList[1]

            # Price of the option
            options.append(float(df['Price'][ix]))
            # determines whether it is a Call or a Put
            myBool = (df['Type'][ix] == "Call")
            iv = impliedVolCore(myBool, 1.0, spot, df['Strike'][ix], T, float(
                df['Price'][ix]), tolerance=1e-4, itermax=100)
            logMoneynesses.append(log(df['Strike'][ix] / spot))
            smiles.append(iv)
    else:
        for strike in strikes:
            myList = []
            for i in range(len(df)):
                if df['Open Int'][i] <> "-" and df['Strike'][i] == strike:
                    myList.append(i)
            # only one price available at this strike
            if len(myList) == 1:
                ix = myList[0]
                # Price of the option
                options.append(float(df['Price'][ix]))
                # determines whether it is a Call or a Put
                myBool = (df['Type'][ix] == "Call")
                iv = impliedVolCore(myBool, 1.0, spot, df['Strike'][ix], T, float(
                    df['Price'][ix]), tolerance=1e-4, itermax=100)
                logMoneynesses.append(log(df['Strike'][ix] / spot))
                smiles.append(iv)
            # two prices available at this strike
            elif len(myList) == 2:
                # choose the strike with the highest traded volume
                if df['Volume'][myList[0]] > df['Volume'][myList[1]]:
                    ix = myList[0]
                else:
                    if df['Price'][myList[1]] == "-":
                        ix = myList[0]
```

```
            else:
                ix = myList[1]

            # Price of the option
            options.append(float(df['Price'][ix]))
            # determines whether it is a Call or a Put
            myBool = (df['Type'][ix] == "Call")
            iv = impliedVolCore(myBool, 1.0, spot, df['Strike'][ix], T, float(
                df['Price'][ix]), tolerance=1e-4, itermax=100)
            logMoneynesses.append(log(df['Strike'][ix] / spot))
            smiles.append(iv)

    # Total implied variance
    totalImpliedVar = [
        smiles[i] * smiles[i] * T for i in range(len(smiles))]
```

1.3 Implied volatility smiles

In [12]:
```
plt.figure(figsize=(8, 5))
plt.plot(logMoneynesses, totalImpliedVar, 'b+', markersize=10)
plt.title("Apple total implied variance smile, maturity = %1.4f" %
          T, fontsize=10, fontweight='bold')
plt.savefig('AppleSmile.eps', format='eps', dpi=1000)
plt.show()
```

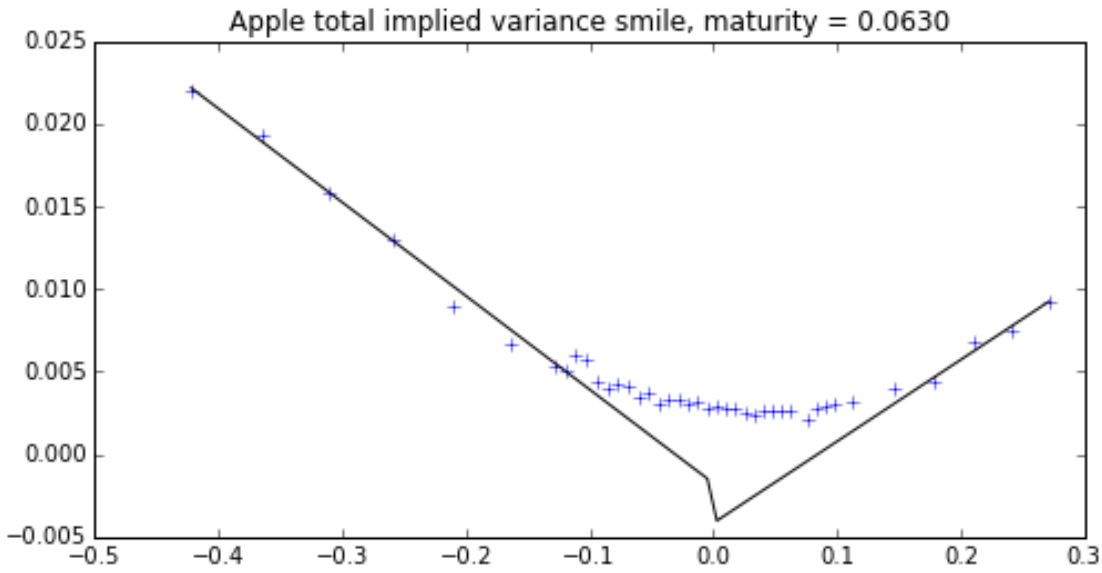


2 Linear fit of the wings - Testing Roger Lee's formula

In [13]:
```
# number of points to take for the linear fit, minimum value: 2
n = 4
```

In [15]:
```
def myFunction(params, *args):
    xx, totVarSqrt = args[0], args[1]
    return sum([pow((totVarSqrt[i] - (params[0] * xx[i] + params[1])), 2.0) for i in range(len(

initialParam = (2.0, 0.0)
outputRight = minimize(myFunction, initialParam, method='TNC', args=(
    logMoneynesses[-n:], totalImpliedVar[-n:]))  # , bounds=mybounds)
outputLeft = minimize(myFunction, initialParam, method='TNC', args=(
    logMoneynesses[:n], totalImpliedVar[:n]))  # , bounds=mybounds)

print "Calibrated parameters Right: ", outputRight.x
print "Calibrated parameters Left: ", outputLeft.x

med = int(floor(size(logMoneynesses) / 2)) + 1


lines = []
for k in range(0, med):
    lines.append(
        outputLeft.x[0] * logMoneynesses[k] + outputLeft.x[1])
for k in range(med, size(logMoneynesses)):
    lines.append(
        outputRight.x[0] * logMoneynesses[k] + outputRight.x[1])

fig = plt.figure(figsize=(8, 4))
plt.plot(logMoneynesses, totalImpliedVar, 'b+')
plt.plot(logMoneynesses, lines, 'k')
plt.title(
    "Apple total implied variance smile, maturity = %1.4f" % T)
plt.savefig('AppleFit.eps', format='eps', dpi=1000)
plt.show()
```

```
Calibrated parameters Right:  [ 0.04940753 -0.00416687]
Calibrated parameters Left:  [-0.0566354  -0.00174406]
```


Apple total implied variance smile, maturity = 0.0630
0.025
0.020
0.015
0.010
0.005
0.000
−0.005
−0.5
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3