Apple_Heston_Calls

January 7, 2019

```
In [1]: import warnings
        warnings.filterwarnings("ignore")
        import modulesForCalibration as mfc
        import matplotlib.pyplot as plt

        #import readPlotOptionSurface_granular_k2_5 as marketSurface
        import readPlotOptionSurfaceedited as marketSurface
        import numpy as np
        import pandas as pd

        from mpl_toolkits.mplot3d import Axes3D
        import matplotlib.pyplot as plt

        from matplotlib import cm
        import cmath
        import math
        from scipy.optimize import fmin


        import plotly.plotly as py
        import plotly.graph_objs as go
```

```
<Figure size 800x600 with 1 Axes>
```

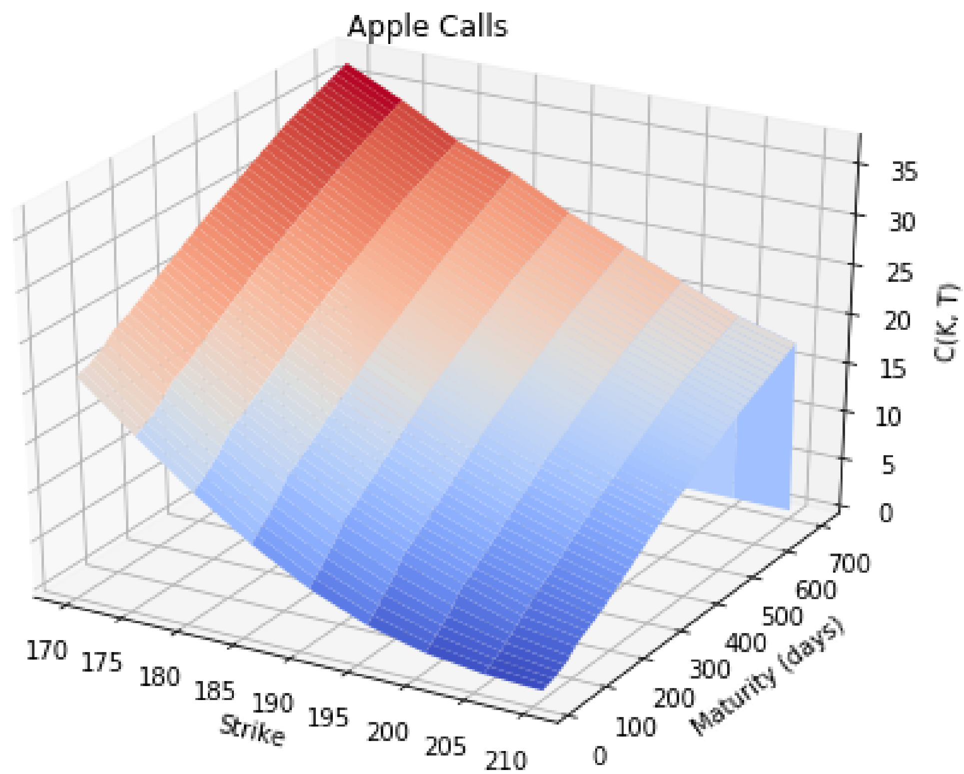
1 This report reflects the work of Lisa He, Alban Zapke, and Naijia Yao, for the project of volatility surface in Computational Methods in Finance with Prof. Hirsa.

1.1 APPL

1.1.1 We set up Grid for Model Prices as provided in readPlotOptionSurface.py provided by Prof. Hirsa

1.1.2 deltaK = 5 & deltaTau = 1/52

```
In [2]: maturities, strikes, marketPrices = marketSurface.readNPlot()
```




```
In [3]: maturities_years = maturities/365
```

2 I. Model Prices

2.0.1 Global Parameters

```
In [4]: # Contract Parameters
        S0 = 190
        K = 190
        k = np.log(K)
        # risk free rate
        r = 0.0245
        # dividend rate
        q = 0.005

        # Parameters
        alpha = 1.5                          # Why?
        eta = 0.2

        n = 12
        N = 2**n
```

```
        # step-size in log strike space  --> FFT constraint
        #lda = (2*np.pi/N)/eta

        # Choice of beta
        #beta = np.log(S0)-N*lda/2
        # beta = np.log(K)
```

- Grid for Model Prices was set up in readPlotOptionSurface

2.1 1. Finding a starting point; code provided as in exampleCalibration_FindingStartingPoint.py by Prof. Hirsa

```
In [5]: iArray = []                             # The alphas (0,1) which are plotted agains RMSE
        rmseArray = []
        rmseMin = 1e10                          # Random; in order to have an error to start wit
```

2.1.1 Model specific parameters:

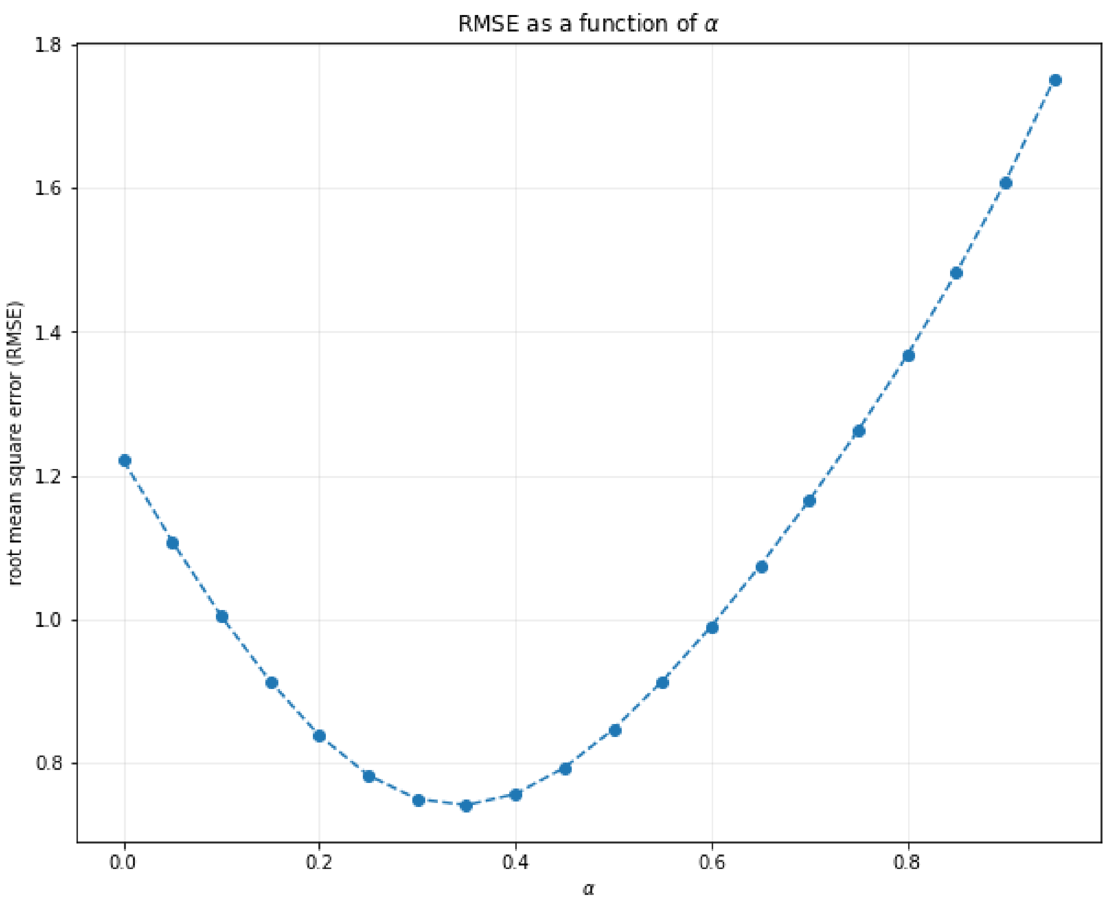
Heston

```
In [6]: model = 'Heston'

        #set 1: promising starting point
        params1 = (1.0, 0.02, 0.05, -0.4, 0.08)
        params2 = (3.0, 0.06, 0.10, -0.6, 0.04)

In [7]: lenT = len(maturities_years)
        lenK = len(strikes)
        modelPrices = np.zeros((lenT, lenK))

In [8]: modelPrices.shape == marketPrices.shape

Out[8]: True

In [9]: marketPrices.shape

Out[9]: (98, 9)

In [10]: iArray = []
         rmseArray=[]
         for i in mfc.myRange(0.0, 1.0, 0.05):

             params = i*np.array(params1) + (1.0-i)*np.array(params2)
             iArray.append(i)

             rmse = mfc.eValue(params, marketPrices, maturities_years, strikes, r, q, S0, alpha,
             rmseArray.append(rmse)
             if (rmse < rmseMin):
                 rmseMin = rmse
                 optimParams = params
```

```
In [27]: len(rmseArray) == len(iArray)
         print(len(rmseArray))
         print(len(iArray))
         #print(rmseArray)
         #print(iArray)
```

```
20
20
```

```
In [28]: fig = plt.figure(figsize=(10,8))
         plt.plot(iArray, rmseArray, 'o--')
         plt.grid(alpha=0.25)
         plt.xlabel('$\\alpha$')
         plt.ylabel('root mean square error (RMSE)')
         plt.title('RMSE as a function of $\\alpha$')
         plt.savefig('startingPoint4.png')
         plt.show()

         print(rmseMin)
         print(optimParams)
```



```
0.7410834358357488
[ 2.3     0.046   0.0825 -0.53    0.054 ]
```

```
In [29]: # Starting point Parameters
         kappa = 2.3
         theta = 0.046
         sig = 0.0825
         rho = -0.53
         v0 = 0.054
```

2.2 2. Optimization of Parameter Set

- Objective Function -

```
In [30]: def objFunc(v, x0, x1, x2):
             # Paraboloid centered on (x, y), with scale factors (10, 20) and minimum 30
             return 10.0*(v[0]-x0)**2 + 20.0*(v[1]-x1)**2 + 30.0*(v[2]-x2)**2 + 40.0
```

```
In [31]: lenT = len(maturities)
         lenK = len(strikes)
```

A) Brute Force Algorithm

```
In [33]: # maturities, strikes, marketPrices = marketSurface.readNPlot()

         #===========================================
         # Grid Search around the starting point
         #-------------------------------------------
         # [ 2.3     0.046   0.0825 -0.53    0.054 ]
         ind_iter = 1
         rmseMin = 1.0e6

         for kappa in mfc.myRange(1.8,2.8,0.5):
             for theta in mfc.myRange(0.036,0.056,0.01):
                 for sig in mfc.myRange(0.0725,0.0925,0.01):
                     for rho in mfc.myRange(-0.63,-0.43,0.1):
                         for v0 in mfc.myRange(0.044,0.064,0.01):
                             params = []
                             params.append(kappa)
                             params.append(theta)
                             params.append(sig)
                             params.append(rho)
                             params.append(v0)

                             print('i = ' + str(ind_iter))
```

```
                                ind_iter += 1
                                print(params)
                                rmse = mfc.eValue(params, marketPrices, maturities_years, strikes,

                                if (rmse < rmseMin):
                                    rmseMin = rmse
                                    params2 = params
                                    print('\nnew min found')
                                    print(rmseMin)
                                    print(params2)
                                    print('')

        print('\nSolution of grid search:')
        print(params2)
        print('Optimal rmse = ' + str(rmseMin))
```

```
i = 1
[1.7, 0.034, 0.07, -0.62, 0.046]

new min found
0.9061558281366756
[1.7, 0.034, 0.07, -0.62, 0.046]

i = 2
[1.7, 0.034, 0.07, -0.62, 0.056]

new min found
0.7495517706837034
[1.7, 0.034, 0.07, -0.62, 0.056]

i = 3
[1.7, 0.034, 0.07, -0.62, 0.066]
i = 4
[1.7, 0.034, 0.07, -0.52, 0.046]
i = 5
[1.7, 0.034, 0.07, -0.52, 0.056]
i = 6
[1.7, 0.034, 0.07, -0.52, 0.066]
i = 7
[1.7, 0.034, 0.07, -0.42000000000000004, 0.046]
i = 8
[1.7, 0.034, 0.07, -0.42000000000000004, 0.056]
i = 9
[1.7, 0.034, 0.07, -0.42000000000000004, 0.066]
i = 10
[1.7, 0.034, 0.08, -0.62, 0.046]
i = 11
[1.7, 0.034, 0.08, -0.62, 0.056]
```

```
new min found
0.7480136822229901
[1.7, 0.034, 0.08, -0.62, 0.056]

i = 12
[1.7, 0.034, 0.08, -0.62, 0.066]
i = 13
[1.7, 0.034, 0.08, -0.52, 0.046]
i = 14
[1.7, 0.034, 0.08, -0.52, 0.056]
i = 15
[1.7, 0.034, 0.08, -0.52, 0.066]
i = 16
[1.7, 0.034, 0.08, -0.42000000000000004, 0.046]
i = 17
[1.7, 0.034, 0.08, -0.42000000000000004, 0.056]
i = 18
[1.7, 0.034, 0.08, -0.42000000000000004, 0.066]
i = 19
[1.7, 0.034, 0.09, -0.62, 0.046]
i = 20
[1.7, 0.034, 0.09, -0.62, 0.056]

new min found
0.7471040695013035
[1.7, 0.034, 0.09, -0.62, 0.056]

i = 21
[1.7, 0.034, 0.09, -0.62, 0.066]
i = 22
[1.7, 0.034, 0.09, -0.52, 0.046]
i = 23
[1.7, 0.034, 0.09, -0.52, 0.056]
i = 24
[1.7, 0.034, 0.09, -0.52, 0.066]
i = 25
[1.7, 0.034, 0.09, -0.42000000000000004, 0.046]
i = 26
[1.7, 0.034, 0.09, -0.42000000000000004, 0.056]
i = 27
[1.7, 0.034, 0.09, -0.42000000000000004, 0.066]
i = 28
[1.7, 0.044000000000000004, 0.07, -0.62, 0.046]

new min found
0.3925847869697346
[1.7, 0.044000000000000004, 0.07, -0.62, 0.046]
```

```
i = 29
[1.7, 0.044000000000000004, 0.07, -0.62, 0.056]
i = 30
[1.7, 0.044000000000000004, 0.07, -0.62, 0.066]
i = 31
[1.7, 0.044000000000000004, 0.07, -0.52, 0.046]
i = 32
[1.7, 0.044000000000000004, 0.07, -0.52, 0.056]
i = 33
[1.7, 0.044000000000000004, 0.07, -0.52, 0.066]
i = 34
[1.7, 0.044000000000000004, 0.07, -0.42000000000000004, 0.046]
i = 35
[1.7, 0.044000000000000004, 0.07, -0.42000000000000004, 0.056]
i = 36
[1.7, 0.044000000000000004, 0.07, -0.42000000000000004, 0.066]
i = 37
[1.7, 0.044000000000000004, 0.08, -0.62, 0.046]

new min found
0.387074050793953
[1.7, 0.044000000000000004, 0.08, -0.62, 0.046]

i = 38
[1.7, 0.044000000000000004, 0.08, -0.62, 0.056]
i = 39
[1.7, 0.044000000000000004, 0.08, -0.62, 0.066]
i = 40
[1.7, 0.044000000000000004, 0.08, -0.52, 0.046]
i = 41
[1.7, 0.044000000000000004, 0.08, -0.52, 0.056]
i = 42
[1.7, 0.044000000000000004, 0.08, -0.52, 0.066]
i = 43
[1.7, 0.044000000000000004, 0.08, -0.42000000000000004, 0.046]
i = 44
[1.7, 0.044000000000000004, 0.08, -0.42000000000000004, 0.056]
i = 45
[1.7, 0.044000000000000004, 0.08, -0.42000000000000004, 0.066]
i = 46
[1.7, 0.044000000000000004, 0.09, -0.62, 0.046]

new min found
0.3822081498347296
[1.7, 0.044000000000000004, 0.09, -0.62, 0.046]

i = 47
```

```
[1.7, 0.044000000000000004, 0.09, -0.62, 0.056]
i = 48
[1.7, 0.044000000000000004, 0.09, -0.62, 0.066]
i = 49
[1.7, 0.044000000000000004, 0.09, -0.52, 0.046]
i = 50
[1.7, 0.044000000000000004, 0.09, -0.52, 0.056]
i = 51
[1.7, 0.044000000000000004, 0.09, -0.52, 0.066]
i = 52
[1.7, 0.044000000000000004, 0.09, -0.42000000000000004, 0.046]
i = 53
[1.7, 0.044000000000000004, 0.09, -0.42000000000000004, 0.056]
i = 54
[1.7, 0.044000000000000004, 0.09, -0.42000000000000004, 0.066]
i = 55
[1.7, 0.054000000000000006, 0.07, -0.62, 0.046]
i = 56
[1.7, 0.054000000000000006, 0.07, -0.62, 0.056]
i = 57
[1.7, 0.054000000000000006, 0.07, -0.62, 0.066]
i = 58
[1.7, 0.054000000000000006, 0.07, -0.52, 0.046]
i = 59
[1.7, 0.054000000000000006, 0.07, -0.52, 0.056]
i = 60
[1.7, 0.054000000000000006, 0.07, -0.52, 0.066]
i = 61
[1.7, 0.054000000000000006, 0.07, -0.42000000000000004, 0.046]
i = 62
[1.7, 0.054000000000000006, 0.07, -0.42000000000000004, 0.056]
i = 63
[1.7, 0.054000000000000006, 0.07, -0.42000000000000004, 0.066]
i = 64
[1.7, 0.054000000000000006, 0.08, -0.62, 0.046]
i = 65
[1.7, 0.054000000000000006, 0.08, -0.62, 0.056]
i = 66
[1.7, 0.054000000000000006, 0.08, -0.62, 0.066]
i = 67
[1.7, 0.054000000000000006, 0.08, -0.52, 0.046]
i = 68
[1.7, 0.054000000000000006, 0.08, -0.52, 0.056]
i = 69
[1.7, 0.054000000000000006, 0.08, -0.52, 0.066]
i = 70
[1.7, 0.054000000000000006, 0.08, -0.42000000000000004, 0.046]
i = 71
```

```
[1.7, 0.054000000000000006, 0.08, -0.42000000000000004, 0.056]
i = 72
[1.7, 0.054000000000000006, 0.08, -0.42000000000000004, 0.066]
i = 73
[1.7, 0.054000000000000006, 0.09, -0.62, 0.046]
i = 74
[1.7, 0.054000000000000006, 0.09, -0.62, 0.056]
i = 75
[1.7, 0.054000000000000006, 0.09, -0.62, 0.066]
i = 76
[1.7, 0.054000000000000006, 0.09, -0.52, 0.046]
i = 77
[1.7, 0.054000000000000006, 0.09, -0.52, 0.056]
i = 78
[1.7, 0.054000000000000006, 0.09, -0.52, 0.066]
i = 79
[1.7, 0.054000000000000006, 0.09, -0.42000000000000004, 0.046]
i = 80
[1.7, 0.054000000000000006, 0.09, -0.42000000000000004, 0.056]
i = 81
[1.7, 0.054000000000000006, 0.09, -0.42000000000000004, 0.066]
i = 82
[2.2, 0.034, 0.07, -0.62, 0.046]
i = 83
[2.2, 0.034, 0.07, -0.62, 0.056]
i = 84
[2.2, 0.034, 0.07, -0.62, 0.066]
i = 85
[2.2, 0.034, 0.07, -0.52, 0.046]
i = 86
[2.2, 0.034, 0.07, -0.52, 0.056]
i = 87
[2.2, 0.034, 0.07, -0.52, 0.066]
i = 88
[2.2, 0.034, 0.07, -0.42000000000000004, 0.046]
i = 89
[2.2, 0.034, 0.07, -0.42000000000000004, 0.056]
i = 90
[2.2, 0.034, 0.07, -0.42000000000000004, 0.066]
i = 91
[2.2, 0.034, 0.08, -0.62, 0.046]
i = 92
[2.2, 0.034, 0.08, -0.62, 0.056]
i = 93
[2.2, 0.034, 0.08, -0.62, 0.066]
i = 94
[2.2, 0.034, 0.08, -0.52, 0.046]
i = 95
```

```
[2.2, 0.034, 0.08, -0.52, 0.056]
i = 96
[2.2, 0.034, 0.08, -0.52, 0.066]
i = 97
[2.2, 0.034, 0.08, -0.42000000000000004, 0.046]
i = 98
[2.2, 0.034, 0.08, -0.42000000000000004, 0.056]
i = 99
[2.2, 0.034, 0.08, -0.42000000000000004, 0.066]
i = 100
[2.2, 0.034, 0.09, -0.62, 0.046]
i = 101
[2.2, 0.034, 0.09, -0.62, 0.056]
i = 102
[2.2, 0.034, 0.09, -0.62, 0.066]
i = 103
[2.2, 0.034, 0.09, -0.52, 0.046]
i = 104
[2.2, 0.034, 0.09, -0.52, 0.056]
i = 105
[2.2, 0.034, 0.09, -0.52, 0.066]
i = 106
[2.2, 0.034, 0.09, -0.42000000000000004, 0.046]
i = 107
[2.2, 0.034, 0.09, -0.42000000000000004, 0.056]
i = 108
[2.2, 0.034, 0.09, -0.42000000000000004, 0.066]
i = 109
[2.2, 0.044000000000000004, 0.07, -0.62, 0.046]
i = 110
[2.2, 0.044000000000000004, 0.07, -0.62, 0.056]
i = 111
[2.2, 0.044000000000000004, 0.07, -0.62, 0.066]
i = 112
[2.2, 0.044000000000000004, 0.07, -0.52, 0.046]
i = 113
[2.2, 0.044000000000000004, 0.07, -0.52, 0.056]
i = 114
[2.2, 0.044000000000000004, 0.07, -0.52, 0.066]
i = 115
[2.2, 0.044000000000000004, 0.07, -0.42000000000000004, 0.046]
i = 116
[2.2, 0.044000000000000004, 0.07, -0.42000000000000004, 0.056]
i = 117
[2.2, 0.044000000000000004, 0.07, -0.42000000000000004, 0.066]
i = 118
[2.2, 0.044000000000000004, 0.08, -0.62, 0.046]
i = 119
```

```
[2.2, 0.044000000000000004, 0.08, -0.62, 0.056]
i = 120
[2.2, 0.044000000000000004, 0.08, -0.62, 0.066]
i = 121
[2.2, 0.044000000000000004, 0.08, -0.52, 0.046]
i = 122
[2.2, 0.044000000000000004, 0.08, -0.52, 0.056]
i = 123
[2.2, 0.044000000000000004, 0.08, -0.52, 0.066]
i = 124
[2.2, 0.044000000000000004, 0.08, -0.42000000000000004, 0.046]
i = 125
[2.2, 0.044000000000000004, 0.08, -0.42000000000000004, 0.056]
i = 126
[2.2, 0.044000000000000004, 0.08, -0.42000000000000004, 0.066]
i = 127
[2.2, 0.044000000000000004, 0.09, -0.62, 0.046]
i = 128
[2.2, 0.044000000000000004, 0.09, -0.62, 0.056]
i = 129
[2.2, 0.044000000000000004, 0.09, -0.62, 0.066]
i = 130
[2.2, 0.044000000000000004, 0.09, -0.52, 0.046]
i = 131
[2.2, 0.044000000000000004, 0.09, -0.52, 0.056]
i = 132
[2.2, 0.044000000000000004, 0.09, -0.52, 0.066]
i = 133
[2.2, 0.044000000000000004, 0.09, -0.42000000000000004, 0.046]
i = 134
[2.2, 0.044000000000000004, 0.09, -0.42000000000000004, 0.056]
i = 135
[2.2, 0.044000000000000004, 0.09, -0.42000000000000004, 0.066]
i = 136
[2.2, 0.054000000000000006, 0.07, -0.62, 0.046]
i = 137
[2.2, 0.054000000000000006, 0.07, -0.62, 0.056]
i = 138
[2.2, 0.054000000000000006, 0.07, -0.62, 0.066]
i = 139
[2.2, 0.054000000000000006, 0.07, -0.52, 0.046]
i = 140
[2.2, 0.054000000000000006, 0.07, -0.52, 0.056]
i = 141
[2.2, 0.054000000000000006, 0.07, -0.52, 0.066]
i = 142
[2.2, 0.054000000000000006, 0.07, -0.42000000000000004, 0.046]
i = 143
```

```
[2.2, 0.054000000000000006, 0.07, -0.42000000000000004, 0.056]
i = 144
[2.2, 0.054000000000000006, 0.07, -0.42000000000000004, 0.066]
i = 145
[2.2, 0.054000000000000006, 0.08, -0.62, 0.046]
i = 146
[2.2, 0.054000000000000006, 0.08, -0.62, 0.056]
i = 147
[2.2, 0.054000000000000006, 0.08, -0.62, 0.066]
i = 148
[2.2, 0.054000000000000006, 0.08, -0.52, 0.046]
i = 149
[2.2, 0.054000000000000006, 0.08, -0.52, 0.056]
i = 150
[2.2, 0.054000000000000006, 0.08, -0.52, 0.066]
i = 151
[2.2, 0.054000000000000006, 0.08, -0.42000000000000004, 0.046]
i = 152
[2.2, 0.054000000000000006, 0.08, -0.42000000000000004, 0.056]
i = 153
[2.2, 0.054000000000000006, 0.08, -0.42000000000000004, 0.066]
i = 154
[2.2, 0.054000000000000006, 0.09, -0.62, 0.046]
i = 155
[2.2, 0.054000000000000006, 0.09, -0.62, 0.056]
i = 156
[2.2, 0.054000000000000006, 0.09, -0.62, 0.066]
i = 157
[2.2, 0.054000000000000006, 0.09, -0.52, 0.046]
i = 158
[2.2, 0.054000000000000006, 0.09, -0.52, 0.056]
i = 159
[2.2, 0.054000000000000006, 0.09, -0.52, 0.066]
i = 160
[2.2, 0.054000000000000006, 0.09, -0.42000000000000004, 0.046]
i = 161
[2.2, 0.054000000000000006, 0.09, -0.42000000000000004, 0.056]
i = 162
[2.2, 0.054000000000000006, 0.09, -0.42000000000000004, 0.066]
i = 163
[2.7, 0.034, 0.07, -0.62, 0.046]
i = 164
[2.7, 0.034, 0.07, -0.62, 0.056]
i = 165
[2.7, 0.034, 0.07, -0.62, 0.066]
i = 166
[2.7, 0.034, 0.07, -0.52, 0.046]
i = 167
```

```
[2.7, 0.034, 0.07, -0.52, 0.056]
i = 168
[2.7, 0.034, 0.07, -0.52, 0.066]
i = 169
[2.7, 0.034, 0.07, -0.42000000000000004, 0.046]
i = 170
[2.7, 0.034, 0.07, -0.42000000000000004, 0.056]
i = 171
[2.7, 0.034, 0.07, -0.42000000000000004, 0.066]
i = 172
[2.7, 0.034, 0.08, -0.62, 0.046]
i = 173
[2.7, 0.034, 0.08, -0.62, 0.056]
i = 174
[2.7, 0.034, 0.08, -0.62, 0.066]
i = 175
[2.7, 0.034, 0.08, -0.52, 0.046]
i = 176
[2.7, 0.034, 0.08, -0.52, 0.056]
i = 177
[2.7, 0.034, 0.08, -0.52, 0.066]
i = 178
[2.7, 0.034, 0.08, -0.42000000000000004, 0.046]
i = 179
[2.7, 0.034, 0.08, -0.42000000000000004, 0.056]
i = 180
[2.7, 0.034, 0.08, -0.42000000000000004, 0.066]
i = 181
[2.7, 0.034, 0.09, -0.62, 0.046]
i = 182
[2.7, 0.034, 0.09, -0.62, 0.056]
i = 183
[2.7, 0.034, 0.09, -0.62, 0.066]
i = 184
[2.7, 0.034, 0.09, -0.52, 0.046]
i = 185
[2.7, 0.034, 0.09, -0.52, 0.056]
i = 186
[2.7, 0.034, 0.09, -0.52, 0.066]
i = 187
[2.7, 0.034, 0.09, -0.42000000000000004, 0.046]
i = 188
[2.7, 0.034, 0.09, -0.42000000000000004, 0.056]
i = 189
[2.7, 0.034, 0.09, -0.42000000000000004, 0.066]
i = 190
[2.7, 0.044000000000000004, 0.07, -0.62, 0.046]
i = 191
```

```
[2.7, 0.044000000000000004, 0.07, -0.62, 0.056]
i = 192
[2.7, 0.044000000000000004, 0.07, -0.62, 0.066]
i = 193
[2.7, 0.044000000000000004, 0.07, -0.52, 0.046]
i = 194
[2.7, 0.044000000000000004, 0.07, -0.52, 0.056]
i = 195
[2.7, 0.044000000000000004, 0.07, -0.52, 0.066]
i = 196
[2.7, 0.044000000000000004, 0.07, -0.42000000000000004, 0.046]
i = 197
[2.7, 0.044000000000000004, 0.07, -0.42000000000000004, 0.056]
i = 198
[2.7, 0.044000000000000004, 0.07, -0.42000000000000004, 0.066]
i = 199
[2.7, 0.044000000000000004, 0.08, -0.62, 0.046]
i = 200
[2.7, 0.044000000000000004, 0.08, -0.62, 0.056]
i = 201
[2.7, 0.044000000000000004, 0.08, -0.62, 0.066]
i = 202
[2.7, 0.044000000000000004, 0.08, -0.52, 0.046]
i = 203
[2.7, 0.044000000000000004, 0.08, -0.52, 0.056]
i = 204
[2.7, 0.044000000000000004, 0.08, -0.52, 0.066]
i = 205
[2.7, 0.044000000000000004, 0.08, -0.42000000000000004, 0.046]
i = 206
[2.7, 0.044000000000000004, 0.08, -0.42000000000000004, 0.056]
i = 207
[2.7, 0.044000000000000004, 0.08, -0.42000000000000004, 0.066]
i = 208
[2.7, 0.044000000000000004, 0.09, -0.62, 0.046]
i = 209
[2.7, 0.044000000000000004, 0.09, -0.62, 0.056]
i = 210
[2.7, 0.044000000000000004, 0.09, -0.62, 0.066]
i = 211
[2.7, 0.044000000000000004, 0.09, -0.52, 0.046]
i = 212
[2.7, 0.044000000000000004, 0.09, -0.52, 0.056]
i = 213
[2.7, 0.044000000000000004, 0.09, -0.52, 0.066]
i = 214
[2.7, 0.044000000000000004, 0.09, -0.42000000000000004, 0.046]
i = 215
```

```
[2.7, 0.044000000000000004, 0.09, -0.42000000000000004, 0.056]
i = 216
[2.7, 0.044000000000000004, 0.09, -0.42000000000000004, 0.066]
i = 217
[2.7, 0.054000000000000006, 0.07, -0.62, 0.046]
i = 218
[2.7, 0.054000000000000006, 0.07, -0.62, 0.056]
i = 219
[2.7, 0.054000000000000006, 0.07, -0.62, 0.066]
i = 220
[2.7, 0.054000000000000006, 0.07, -0.52, 0.046]
i = 221
[2.7, 0.054000000000000006, 0.07, -0.52, 0.056]
i = 222
[2.7, 0.054000000000000006, 0.07, -0.52, 0.066]
i = 223
[2.7, 0.054000000000000006, 0.07, -0.42000000000000004, 0.046]
i = 224
[2.7, 0.054000000000000006, 0.07, -0.42000000000000004, 0.056]
i = 225
[2.7, 0.054000000000000006, 0.07, -0.42000000000000004, 0.066]
i = 226
[2.7, 0.054000000000000006, 0.08, -0.62, 0.046]
i = 227
[2.7, 0.054000000000000006, 0.08, -0.62, 0.056]
i = 228
[2.7, 0.054000000000000006, 0.08, -0.62, 0.066]
i = 229
[2.7, 0.054000000000000006, 0.08, -0.52, 0.046]
i = 230
[2.7, 0.054000000000000006, 0.08, -0.52, 0.056]
i = 231
[2.7, 0.054000000000000006, 0.08, -0.52, 0.066]
i = 232
[2.7, 0.054000000000000006, 0.08, -0.42000000000000004, 0.046]
i = 233
[2.7, 0.054000000000000006, 0.08, -0.42000000000000004, 0.056]
i = 234
[2.7, 0.054000000000000006, 0.08, -0.42000000000000004, 0.066]
i = 235
[2.7, 0.054000000000000006, 0.09, -0.62, 0.046]
i = 236
[2.7, 0.054000000000000006, 0.09, -0.62, 0.056]
i = 237
[2.7, 0.054000000000000006, 0.09, -0.62, 0.066]
i = 238
[2.7, 0.054000000000000006, 0.09, -0.52, 0.046]
i = 239
```

```
[2.7, 0.054000000000000006, 0.09, -0.52, 0.056]
i = 240
[2.7, 0.054000000000000006, 0.09, -0.52, 0.066]
i = 241
[2.7, 0.054000000000000006, 0.09, -0.42000000000000004, 0.046]
i = 242
[2.7, 0.054000000000000006, 0.09, -0.42000000000000004, 0.056]
i = 243
[2.7, 0.054000000000000006, 0.09, -0.42000000000000004, 0.066]

Solution of grid search:
[1.7, 0.044000000000000004, 0.09, -0.62, 0.046]
Optimal rmse = 0.3822081498347296
```

In [ ]:
```
# Solution of grid search:
# [1.7, 0.044000000000000004, 0.09, -0.62, 0.046]
# Optimal rmse = 0.3822081498347296
```

- Market vs. Model Surface -

In [34]:
```
params2
```

Out[34]:
```
[1.7, 0.044000000000000004, 0.09, -0.62, 0.046]
```

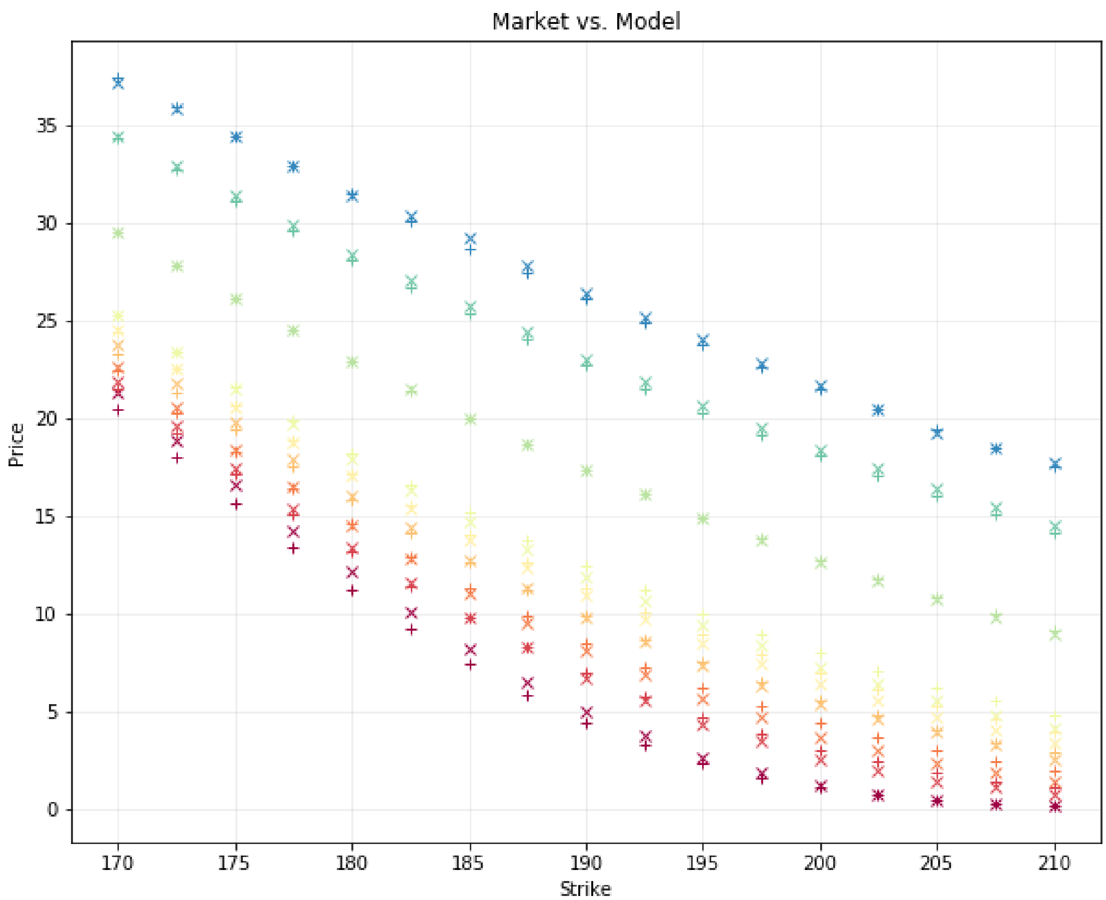
In [35]:
```
lenT = len(maturities)
lenK = len(strikes)
modelPrices = np.zeros((lenT, lenK))

for i in range(lenT):
    for j in range(lenK):
        T = maturities_years[i]
        K = strikes[j]
        [km, cT_km] = mfc.genericFFT(params2, S0, K, r, q, T, alpha, eta, n, model)
        modelPrices[i,j] = cT_km[0]
```

In [36]:
```
modelPrices.shape == marketPrices.shape
```

Out[36]:
```
True
```

In [37]:
```
# plot
fig = plt.figure(figsize=(10,8))
labels = []
colormap = cm.Spectral
plt.gca().set_color_cycle([colormap(i) for i in np.linspace(0, 0.9, len(maturities))])
for i in range(len(maturities)):
    plt.plot(strikes, marketPrices[i,:], 'x')
    labels.append('T = ' + str(maturities[i]))
```

```python
for i in range(len(maturities)):
    plt.plot(strikes, modelPrices[i,:], '+')
    labels.append('T = ' + str(maturities[i]))

#plt.legend(labels, loc='upper right', ncol=2)
plt.grid(alpha=0.25)
plt.xlabel('Strike')
plt.ylabel('Price')
plt.title('Market vs. Model')
plt.savefig('MarketvsModel_GridSearch.png')
plt.show()
```



B) Nelder Mead Algorithm (Gradient-free) from exampleCalibration_NelderMead.py

In [316]:
```python
#=========================================
# Nelder-Mead Simplex Algorithm -- fmin
#=========================================
# [ 2.3      0.046    0.0825 -0.53     0.054 ]
```

```python
params = [ 2.3   ,  0.046 ,  0.0825 ,-0.53 ,   0.054 ]

def callbackF(xi):
    global num_iter
    global arg
    print('i = ' + str(num_iter))
    print('x_i = ' + str(xi))
    print('f_i = ' + str(mfc.eValue(xi, *arg)))
    num_iter += 1

arg = (marketPrices, maturities_years, strikes, r, q, S0, alpha, eta, n, model)

num_iter = 1
#xopt, fopt, iters, funcalls, warnflag, allvecs = fmin(
t = fmin(
        mfc.eValue,
        params,
        args=arg,
        xtol=1e-4,
        ftol=1e-4,
        maxiter=200,
        maxfun=400,
        callback=callbackF,
        disp=True,
        retall=False,
        full_output=True)

print('optimal params = ')
print(t[0])
print('f = ' + str(t[1]))
```

```
i = 1
x_i = [ 2.266    0.0396   0.0824  -0.5356   0.05768]
f_i = 0.5999402451624422
i = 2
x_i = [ 2.3056    0.04136   0.08384  -0.54496   0.051408]
f_i = 0.42666037838617843
i = 3
x_i = [ 2.3056    0.04136   0.08384  -0.54496   0.051408]
f_i = 0.42666037838617843
i = 4
x_i = [ 2.3056    0.04136   0.08384  -0.54496   0.051408]
f_i = 0.42666037838617843
i = 5
x_i = [ 2.3056    0.04136   0.08384  -0.54496   0.051408]
f_i = 0.42666037838617843
i = 6
x_i = [ 2.3056    0.04136   0.08384  -0.54496   0.051408]
```

```
f_i = 0.42666037838617843
i = 7
x_i = [ 2.49878886  0.04269028  0.08596905 -0.54232522  0.04635825]
f_i = 0.3534186703723513
i = 8
x_i = [ 2.49878886  0.04269028  0.08596905 -0.54232522  0.04635825]
f_i = 0.3534186703723513
i = 9
x_i = [ 2.49878886  0.04269028  0.08596905 -0.54232522  0.04635825]
f_i = 0.3534186703723513
i = 10
x_i = [ 2.49878886  0.04269028  0.08596905 -0.54232522  0.04635825]
f_i = 0.3534186703723513
i = 11
x_i = [ 2.49878886  0.04269028  0.08596905 -0.54232522  0.04635825]
f_i = 0.3534186703723513
i = 12
x_i = [ 2.44169614  0.04398799  0.08715379 -0.55740401  0.04247574]
f_i = 0.33298501802396074
i = 13
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 14
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 15
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 16
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 17
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 18
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 19
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 20
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 21
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 22
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
```

```
f_i = 0.23513172993769496
i = 23
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 24
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 25
x_i = [ 2.42325954  0.04692504  0.08731114 -0.53580678  0.04048194]
f_i = 0.23513172993769496
i = 26
x_i = [ 2.46419308  0.04703995  0.08916744 -0.54093642  0.03954043]
f_i = 0.23495361056125946
i = 27
x_i = [ 2.46419308  0.04703995  0.08916744 -0.54093642  0.03954043]
f_i = 0.23495361056125946
i = 28
x_i = [ 2.44407125  0.04746913  0.08894755 -0.54238945  0.03926347]
f_i = 0.2334855910436223
i = 29
x_i = [ 2.44407125  0.04746913  0.08894755 -0.54238945  0.03926347]
f_i = 0.2334855910436223
i = 30
x_i = [ 2.44407125  0.04746913  0.08894755 -0.54238945  0.03926347]
f_i = 0.2334855910436223
i = 31
x_i = [ 2.44407125  0.04746913  0.08894755 -0.54238945  0.03926347]
f_i = 0.2334855910436223
i = 32
x_i = [ 2.45888952  0.04754798  0.09050239 -0.54436559  0.03897749]
f_i = 0.23331041745177195
i = 33
x_i = [ 2.45888952  0.04754798  0.09050239 -0.54436559  0.03897749]
f_i = 0.23331041745177195
i = 34
x_i = [ 2.4338437   0.04718557  0.08906251 -0.54097841  0.04001404]
f_i = 0.23303728327928516
i = 35
x_i = [ 2.4338437   0.04718557  0.08906251 -0.54097841  0.04001404]
f_i = 0.23303728327928516
i = 36
x_i = [ 2.471399    0.04758943  0.09078664 -0.5535581   0.03911045]
f_i = 0.2322910629820369
i = 37
x_i = [ 2.46976699  0.04710791  0.09156002 -0.5480975   0.03974459]
f_i = 0.23141349148948395
i = 38
x_i = [ 2.46976699  0.04710791  0.09156002 -0.5480975   0.03974459]
```

```
f_i = 0.23141349148948395
i = 39
x_i = [ 2.46976699  0.04710791  0.09156002 -0.5480975   0.03974459]
f_i = 0.23141349148948395
i = 40
x_i = [ 2.46349108  0.04720803  0.09133552 -0.55427754  0.04007794]
f_i = 0.23128352877397965
i = 41
x_i = [ 2.53367192  0.04777634  0.09599563 -0.57398734  0.0385934 ]
f_i = 0.2300813828859211
i = 42
x_i = [ 2.49592508  0.04708077  0.09567277 -0.5610855   0.04009036]
f_i = 0.2280967783363342
i = 43
x_i = [ 2.49592508  0.04708077  0.09567277 -0.5610855   0.04009036]
f_i = 0.2280967783363342
i = 44
x_i = [ 2.55636063  0.04682898  0.09867294 -0.5783442   0.04025472]
f_i = 0.22620387456850474
i = 45
x_i = [ 2.58925274  0.04739773  0.10266448 -0.60155193  0.03974208]
f_i = 0.22347969311384042
i = 46
x_i = [ 2.67726156  0.04731132  0.10991087 -0.6175565   0.03887386]
f_i = 0.22005882084564993
i = 47
x_i = [ 2.67726156  0.04731132  0.10991087 -0.6175565   0.03887386]
f_i = 0.22005882084564993
i = 48
x_i = [ 2.75017756  0.04682787  0.11766632 -0.64786441  0.04049507]
f_i = 0.21437450622562637
i = 49
x_i = [ 2.90793717  0.04677154  0.12877825 -0.70152804  0.03996189]
f_i = 0.20924858754412817
i = 50
x_i = [ 2.99801201  0.04724067  0.14084111 -0.74092096  0.03945747]
f_i = 0.2062199430236543
i = 51
x_i = [ 2.99801201  0.04724067  0.14084111 -0.74092096  0.03945747]
f_i = 0.2062199430236543
i = 52
x_i = [ 2.99801201  0.04724067  0.14084111 -0.74092096  0.03945747]
f_i = 0.2062199430236543
i = 53
x_i = [ 2.99801201  0.04724067  0.14084111 -0.74092096  0.03945747]
f_i = 0.2062199430236543
i = 54
x_i = [ 2.99801201  0.04724067  0.14084111 -0.74092096  0.03945747]
```

```
f_i = 0.2062199430236543
i = 55
x_i = [ 2.99801201  0.04724067  0.14084111 -0.74092096  0.03945747]
f_i = 0.2062199430236543
i = 56
x_i = [ 2.99801201  0.04724067  0.14084111 -0.74092096  0.03945747]
f_i = 0.2062199430236543
i = 57
x_i = [ 3.07844395  0.04700464  0.14570419 -0.75947765  0.03944928]
f_i = 0.20611712621077014
i = 58
x_i = [ 3.11078825  0.04698636  0.14881083 -0.77662041  0.03987304]
f_i = 0.20580841413396422
i = 59
x_i = [ 3.11078825  0.04698636  0.14881083 -0.77662041  0.03987304]
f_i = 0.20580841413396422
i = 60
x_i = [ 3.11078825  0.04698636  0.14881083 -0.77662041  0.03987304]
f_i = 0.20580841413396422
i = 61
x_i = [ 3.02952496  0.04697977  0.14141585 -0.74629067  0.03969883]
f_i = 0.2057839906463977
i = 62
x_i = [ 3.02952496  0.04697977  0.14141585 -0.74629067  0.03969883]
f_i = 0.2057839906463977
i = 63
x_i = [ 3.03942212  0.04712709  0.14351705 -0.75258016  0.0395411 ]
f_i = 0.20569217128348014
i = 64
x_i = [ 3.03942212  0.04712709  0.14351705 -0.75258016  0.0395411 ]
f_i = 0.20569217128348014
i = 65
x_i = [ 3.04824114  0.04694959  0.14402064 -0.75590816  0.03995908]
f_i = 0.20565162344905752
i = 66
x_i = [ 3.06494591  0.04703355  0.14538897 -0.76286     0.03980384]
f_i = 0.20562436599538997
i = 67
x_i = [ 3.08194991  0.04699576  0.14654873 -0.76667313  0.03981498]
f_i = 0.20561692398466672
i = 68
x_i = [ 3.08194991  0.04699576  0.14654873 -0.76667313  0.03981498]
f_i = 0.20561692398466672
i = 69
x_i = [ 3.04503042  0.04699813  0.14321891 -0.75309011  0.03973858]
f_i = 0.205607302334112
i = 70
x_i = [ 3.05053983  0.0470589   0.14423969 -0.75628585  0.03967946]
```

```
f_i = 0.2055854674098366
i = 71
x_i = [ 3.05053983  0.0470589   0.14423969 -0.75628585  0.03967946]
f_i = 0.2055854674098366
i = 72
x_i = [ 3.06394254  0.04702411  0.14520855 -0.76139219  0.03976881]
f_i = 0.20558077441140762
i = 73
x_i = [ 3.06394254  0.04702411  0.14520855 -0.76139219  0.03976881]
f_i = 0.20558077441140762
i = 74
x_i = [ 3.06944046  0.04701759  0.14551612 -0.76246892  0.03976185]
f_i = 0.20557736410766028
i = 75
x_i = [ 3.06944046  0.04701759  0.14551612 -0.76246892  0.03976185]
f_i = 0.20557736410766028
i = 76
x_i = [ 3.05359746  0.04701855  0.14410517 -0.75664868  0.03972742]
f_i = 0.20557215865223796
i = 77
x_i = [ 3.05359746  0.04701855  0.14410517 -0.75664868  0.03972742]
f_i = 0.20557215865223796
i = 78
x_i = [ 3.05359746  0.04701855  0.14410517 -0.75664868  0.03972742]
f_i = 0.20557215865223796
i = 79
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
f_i = 0.2055644665110863
i = 80
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
f_i = 0.2055644665110863
i = 81
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
f_i = 0.2055644665110863
i = 82
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
f_i = 0.2055644665110863
i = 83
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
f_i = 0.2055644665110863
i = 84
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
f_i = 0.2055644665110863
i = 85
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
f_i = 0.2055644665110863
i = 86
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
```

```
f_i = 0.2055644665110863
i = 87
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
f_i = 0.2055644665110863
i = 88
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
f_i = 0.2055644665110863
i = 89
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
f_i = 0.2055644665110863
i = 90
x_i = [ 3.05765215  0.04702068  0.14425245 -0.75776744  0.03973954]
f_i = 0.2055644665110863
i = 91
x_i = [ 3.06799906  0.04701001  0.14475348 -0.76101962  0.03976142]
f_i = 0.2055605910861548
i = 92
x_i = [ 3.06799906  0.04701001  0.14475348 -0.76101962  0.03976142]
f_i = 0.2055605910861548
i = 93
x_i = [ 3.06799906  0.04701001  0.14475348 -0.76101962  0.03976142]
f_i = 0.2055605910861548
i = 94
x_i = [ 3.06799906  0.04701001  0.14475348 -0.76101962  0.03976142]
f_i = 0.2055605910861548
i = 95
x_i = [ 3.0654622   0.04701048  0.14422387 -0.75921912  0.03974836]
f_i = 0.20555518923365526
i = 96
x_i = [ 3.07661197  0.0470128   0.14478376 -0.76276344  0.03973727]
f_i = 0.20555177636427058
i = 97
x_i = [ 3.07661197  0.0470128   0.14478376 -0.76276344  0.03973727]
f_i = 0.20555177636427058
i = 98
x_i = [ 3.07661197  0.0470128   0.14478376 -0.76276344  0.03973727]
f_i = 0.20555177636427058
i = 99
x_i = [ 3.07661197  0.0470128   0.14478376 -0.76276344  0.03973727]
f_i = 0.20555177636427058
i = 100
x_i = [ 3.08742797  0.04699453  0.14451399 -0.76398809  0.03974151]
f_i = 0.2055342943403706
i = 101
x_i = [ 3.08742797  0.04699453  0.14451399 -0.76398809  0.03974151]
f_i = 0.2055342943403706
i = 102
x_i = [ 3.08742797  0.04699453  0.14451399 -0.76398809  0.03974151]
```

```
f_i = 0.2055342943403706
i = 103
x_i = [ 3.08742797  0.04699453  0.14451399 -0.76398809  0.03974151]
f_i = 0.2055342943403706
i = 104
x_i = [ 3.11201839  0.04699966  0.14525198 -0.7705261   0.03971244]
f_i = 0.20552973199545577
i = 105
x_i = [ 3.12749392  0.0469586   0.1450021  -0.77280958  0.03974454]
f_i = 0.20551026804146919
i = 106
x_i = [ 3.12749392  0.0469586   0.1450021  -0.77280958  0.03974454]
f_i = 0.20551026804146919
i = 107
x_i = [ 3.14307272  0.04695976  0.14522142 -0.77638145  0.03975298]
f_i = 0.20550263757717865
i = 108
x_i = [ 3.16752297  0.04695916  0.14516866 -0.7803125   0.03967948]
f_i = 0.20547135416522996
i = 109
x_i = [ 3.16752297  0.04695916  0.14516866 -0.7803125   0.03967948]
f_i = 0.20547135416522996
i = 110
x_i = [ 3.20991087  0.04688664  0.14465561 -0.78734952  0.03973077]
f_i = 0.20544203569898412
i = 111
x_i = [ 3.20991087  0.04688664  0.14465561 -0.78734952  0.03973077]
f_i = 0.20544203569898412
i = 112
x_i = [ 3.29546808  0.04687738  0.14582341 -0.80597051  0.03966586]
f_i = 0.2054142564795019
i = 113
x_i = [ 3.35574771  0.04682562  0.14574598 -0.8165802   0.0395967 ]
f_i = 0.20539766196492626
i = 114
x_i = [ 3.3972258   0.04677179  0.14534905 -0.82295233  0.03963962]
f_i = 0.20537872409414049
i = 115
x_i = [ 3.3972258   0.04677179  0.14534905 -0.82295233  0.03963962]
f_i = 0.20537872409414049
i = 116
x_i = [ 3.4522621   0.04674059  0.14510963 -0.83293753  0.03959121]
f_i = 0.20536725513309545
i = 117
x_i = [ 3.4522621   0.04674059  0.14510963 -0.83293753  0.03959121]
f_i = 0.20536725513309545
i = 118
x_i = [ 3.36516991  0.04683781  0.14542267 -0.81722308  0.03956532]
```

```
f_i = 0.20535420249192016
i = 119
x_i = [ 3.36516991  0.04683781  0.14542267 -0.81722308  0.03956532]
f_i = 0.20535420249192016
i = 120
x_i = [ 3.51200588  0.0467045   0.14505441 -0.84345762  0.03961179]
f_i = 0.20532781451293086
i = 121
x_i = [ 3.51200588  0.0467045   0.14505441 -0.84345762  0.03961179]
f_i = 0.20532781451293086
i = 122
x_i = [ 3.42981745  0.04681001  0.14483497 -0.82828809  0.03953995]
f_i = 0.20529224475037422
i = 123
x_i = [ 3.56689879  0.04667954  0.14367543 -0.85000773  0.03950407]
f_i = 0.20528541836421607
i = 124
x_i = [ 3.44088958  0.04679574  0.14433549 -0.82833564  0.03955055]
f_i = 0.20526398452316652
i = 125
x_i = [ 3.55826963  0.04668169  0.14358758 -0.84849511  0.03957052]
f_i = 0.20525226617579528
i = 126
x_i = [ 3.64416537  0.04665962  0.14396981 -0.86522728  0.0394775 ]
f_i = 0.20523357257216507
i = 127
x_i = [ 3.54401044  0.04674615  0.1431069  -0.84468391  0.03944524]
f_i = 0.2052090042895352
i = 128
x_i = [ 3.54401044  0.04674615  0.1431069  -0.84468391  0.03944524]
f_i = 0.2052090042895352
i = 129
x_i = [ 3.57678565  0.04671978  0.14337853 -0.85125376  0.03950514]
f_i = 0.20520198126241795
i = 130
x_i = [ 3.57678565  0.04671978  0.14337853 -0.85125376  0.03950514]
f_i = 0.20520198126241795
i = 131
x_i = [ 3.57678565  0.04671978  0.14337853 -0.85125376  0.03950514]
f_i = 0.20520198126241795
i = 132
x_i = [ 3.60379984  0.04672948  0.14337089 -0.85602063  0.03940654]
f_i = 0.20519351195967195
i = 133
x_i = [ 3.60379984  0.04672948  0.14337089 -0.85602063  0.03940654]
f_i = 0.20519351195967195
i = 134
x_i = [ 3.6186887   0.04671477  0.14244879 -0.8565904   0.0394049 ]
```

```
f_i = 0.20518632434010048
i = 135
x_i = [ 3.6186887   0.04671477  0.14244879 -0.8565904   0.0394049 ]
f_i = 0.20518632434010048
i = 136
x_i = [ 3.6186887   0.04671477  0.14244879 -0.8565904   0.0394049 ]
f_i = 0.20518632434010048
i = 137
x_i = [ 3.76051159  0.04663786  0.14325221 -0.88448468  0.03940466]
f_i = 0.20517195847900568
i = 138
x_i = [ 3.76051159  0.04663786  0.14325221 -0.88448468  0.03940466]
f_i = 0.20517195847900568
i = 139
x_i = [ 3.76051159  0.04663786  0.14325221 -0.88448468  0.03940466]
f_i = 0.20517195847900568
i = 140
x_i = [ 3.7457418   0.04663922  0.14290222 -0.88078964  0.03937301]
f_i = 0.20517134484132976
i = 141
x_i = [ 3.79667339  0.04661039  0.14250723 -0.88941256  0.03936481]
f_i = 0.205169026703801
i = 142
x_i = [ 3.79667339  0.04661039  0.14250723 -0.88941256  0.03936481]
f_i = 0.205169026703801
i = 143
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 144
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 145
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 146
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 147
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 148
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 149
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 150
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
```

```
f_i = 0.20516221110910463
i = 151
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 152
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 153
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 154
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 155
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 156
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 157
x_i = [ 3.82192684  0.04662083  0.1424049  -0.89349698  0.03931203]
f_i = 0.20516221110910463
i = 158
x_i = [ 3.80737104  0.04662733  0.14250091 -0.89119261  0.03933151]
f_i = 0.2051621858643872
i = 159
x_i = [ 3.80737104  0.04662733  0.14250091 -0.89119261  0.03933151]
f_i = 0.2051621858643872
i = 160
x_i = [ 3.81561488  0.04662026  0.14246581 -0.89263123  0.03933588]
f_i = 0.20516206326376413
i = 161
x_i = [ 3.82370747  0.04661984  0.14247685 -0.89408041  0.03932488]
f_i = 0.2051620438566246
i = 162
x_i = [ 3.79864941  0.04663028  0.14258011 -0.88967592  0.03933943]
f_i = 0.20516204286511666
i = 163
x_i = [ 3.79864941  0.04663028  0.14258011 -0.88967592  0.03933943]
f_i = 0.20516204286511666
i = 164
x_i = [ 3.81796742  0.04662234  0.14244282 -0.89290288  0.0393205 ]
f_i = 0.205161934377883
i = 165
x_i = [ 3.83363975  0.04661389  0.14240554 -0.8955673   0.03931729]
f_i = 0.20516174332579296
i = 166
x_i = [ 3.83363975  0.04661389  0.14240554 -0.8955673   0.03931729]
```

```
f_i = 0.20516174332579296
i = 167
x_i = [ 3.81982929  0.04662396  0.14255262 -0.89326105  0.03931471]
f_i = 0.20516160224353272
i = 168
x_i = [ 3.80878062  0.0466248   0.14254295 -0.89127419  0.0393287 ]
f_i = 0.20516147634167492
i = 169
x_i = [ 3.80878062  0.0466248   0.14254295 -0.89127419  0.0393287 ]
f_i = 0.20516147634167492
i = 170
x_i = [ 3.80878062  0.0466248   0.14254295 -0.89127419  0.0393287 ]
f_i = 0.20516147634167492
i = 171
x_i = [ 3.80878062  0.0466248   0.14254295 -0.89127419  0.0393287 ]
f_i = 0.20516147634167492
i = 172
x_i = [ 3.81252988  0.04662304  0.14266439 -0.89204075  0.03933429]
f_i = 0.20516105062573195
i = 173
x_i = [ 3.81252988  0.04662304  0.14266439 -0.89204075  0.03933429]
f_i = 0.20516105062573195
i = 174
x_i = [ 3.81252988  0.04662304  0.14266439 -0.89204075  0.03933429]
f_i = 0.20516105062573195
i = 175
x_i = [ 3.81378985  0.04662675  0.14274924 -0.89205814  0.03931481]
f_i = 0.20516082433349742
i = 176
x_i = [ 3.81378985  0.04662675  0.14274924 -0.89205814  0.03931481]
f_i = 0.20516082433349742
i = 177
x_i = [ 3.8217028   0.04662007  0.14280091 -0.89361345  0.03932402]
f_i = 0.20516066914854622
i = 178
x_i = [ 3.8217028   0.04662007  0.14280091 -0.89361345  0.03932402]
f_i = 0.20516066914854622
i = 179
x_i = [ 3.82027433  0.04662059  0.14290361 -0.89308302  0.0393256 ]
f_i = 0.20515963716490657
i = 180
x_i = [ 3.82027433  0.04662059  0.14290361 -0.89308302  0.0393256 ]
f_i = 0.20515963716490657
i = 181
x_i = [ 3.82027433  0.04662059  0.14290361 -0.89308302  0.0393256 ]
f_i = 0.20515963716490657
i = 182
x_i = [ 3.82027433  0.04662059  0.14290361 -0.89308302  0.0393256 ]
```

```
f_i = 0.20515963716490657
i = 183
x_i = [ 3.82027433  0.04662059  0.14290361 -0.89308302  0.0393256 ]
f_i = 0.20515963716490657
i = 184
x_i = [ 3.82027433  0.04662059  0.14290361 -0.89308302  0.0393256 ]
f_i = 0.20515963716490657
i = 185
x_i = [ 3.84386961  0.04661213  0.1430048  -0.8968736   0.03930303]
f_i = 0.205159499306971
i = 186
x_i = [ 3.8448443   0.04661259  0.14312423 -0.89723276  0.03931445]
f_i = 0.205159207585719
i = 187
x_i = [ 3.8448443   0.04661259  0.14312423 -0.89723276  0.03931445]
f_i = 0.205159207585719
i = 188
x_i = [ 3.85253844  0.0466042   0.14291273 -0.8984554   0.03931795]
f_i = 0.20515905398701123
i = 189
x_i = [ 3.81803333  0.04662593  0.14338234 -0.8922191   0.03932578]
f_i = 0.20515847505315768
i = 190
x_i = [ 3.81803333  0.04662593  0.14338234 -0.8922191   0.03932578]
f_i = 0.20515847505315768
i = 191
x_i = [ 3.81803333  0.04662593  0.14338234 -0.8922191   0.03932578]
f_i = 0.20515847505315768
i = 192
x_i = [ 3.85238791  0.04661123  0.14328722 -0.89814229  0.03931229]
f_i = 0.20515811792567681
i = 193
x_i = [ 3.83974316  0.04661179  0.14351064 -0.89550329  0.03933317]
f_i = 0.20515685928014438
i = 194
x_i = [ 3.83974316  0.04661179  0.14351064 -0.89550329  0.03933317]
f_i = 0.20515685928014438
i = 195
x_i = [ 3.83974316  0.04661179  0.14351064 -0.89550329  0.03933317]
f_i = 0.20515685928014438
i = 196
x_i = [ 3.8051818   0.04663538  0.14364609 -0.88916331  0.03933284]
f_i = 0.20515655774883437
i = 197
x_i = [ 3.87299009  0.04660308  0.14369465 -0.90039543  0.03930726]
f_i = 0.20515491820753376
i = 198
x_i = [ 3.87299009  0.04660308  0.14369465 -0.90039543  0.03930726]
```

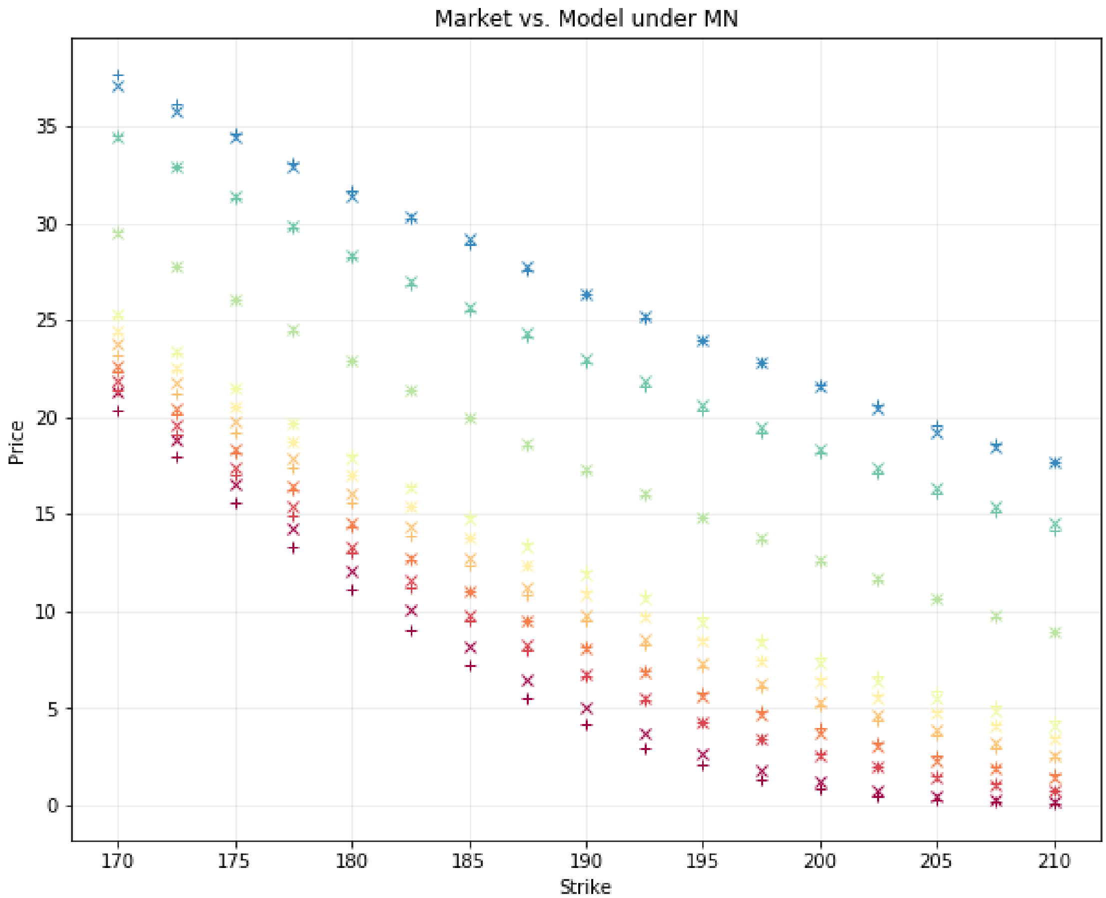
```
f_i = 0.20515491820753376
i = 199
x_i = [ 3.82879801  0.04661798  0.14376313 -0.89199663  0.03933305]
f_i = 0.20515098097908327
Warning: Maximum number of iterations has been exceeded.
optimal params =
[ 3.82879801  0.04661798  0.14376313 -0.89199663  0.03933305]
f = 0.20515098097908327
```

```
In [ ]: # AT 199 ITERATIONS IT BLEW UP - MAX # OF ITERATIONS
        #[ 3.82879801  0.04661798  0.14376313 -0.89199663  0.03933305]
        #f = 0.20515098097908327
```

- Market vs. Model Surface -

```
In [38]: params_NM = [ 3.82879801,  0.04661798,  0.14376313, -0.89199663,  0.03933305]
         lenT = len(maturities)
         lenK = len(strikes)
         modelPrices_MN = np.zeros((lenT, lenK))

         for i in range(lenT):
             for j in range(lenK):
                 T = maturities_years[i]
                 K = strikes[j]
                 [km, cT_km] = mfc.genericFFT(params_NM, S0, K, r, q, T, alpha, eta, n, model)
                 modelPrices_MN[i,j] = cT_km[0]
```

```
In [39]: # plot
         fig = plt.figure(figsize=(10,8))
         labels = []
         colormap = cm.Spectral
         plt.gca().set_color_cycle([colormap(i) for i in np.linspace(0, 0.9, len(maturities))])
         for i in range(len(maturities)):
             plt.plot(strikes, marketPrices[i,:], 'x')
             labels.append('T = ' + str(maturities[i]))

         for i in range(len(maturities)):
             plt.plot(strikes, modelPrices_MN[i,:], '+')
             labels.append('T = ' + str(maturities[i]))
         #plt.legend(labels, loc='upper right', ncol=2)
         plt.grid(alpha=0.25)
         plt.xlabel('Strike')
         plt.ylabel('Price')
         plt.title('Market vs. Model under MN')
         #plt.savefig('MarketvsModel_NelderMead.png')
         plt.show()
```



C) BFGS Algorithm (Gradient-based) from exampleCalibration_BFGS.py

```
In [44]: import warnings
         warnings.filterwarnings("ignore")

         import readPlotOptionSurface
         import modulesForCalibration as mfc

         from scipy.optimize import fmin_bfgs

         import numpy as np
         import matplotlib.pyplot as plt
         from matplotlib import cm

         params = [ 2.3,      0.046 ,  0.0825 ,-0.53   ,  0.054 ]

         def callbackF(xi):
             global num_iter
             global arg
```

```python
        print(' ')
        print('i = ' + str(num_iter))
        print('x_i = ' + str(xi))
        print('f_i = ' + str(mfc.eValue(xi, *arg)))
        num_iter += 1

    arg = (marketPrices, maturities_years, strikes, r, q, S0, alpha, eta, n, model)

    num_iter = 1
    [xopt, fopt, gopt, Bopt, func_calls, grad_calls, warnflg] = fmin_bfgs(
            mfc.eValue,
            params,
            args=arg,
            fprime=None,
            callback=callbackF,
            maxiter=20,
            full_output=True,
            retall=False)

    print('optimal params = ')
    print(xopt)
    print('f = ' + str(fopt))
```

```
i = 1
x_i = [ 2.30000933  0.03976261  0.08258009 -0.53000747  0.04813749]
f_i = 0.5369125315296658

i = 2
x_i = [ 2.30001144  0.04317609  0.08272487 -0.53002916  0.04179787]
f_i = 0.46072398862013314

i = 3
x_i = [ 2.33026752  0.04588309  0.21801585 -0.55012924  0.04134229]
f_i = 0.3662316103511714

i = 4
x_i = [ 2.37026569  0.04765539  0.1975895  -0.54910741  0.04115752]
f_i = 0.3211739202480106

i = 5
x_i = [ 2.37348347  0.04704073  0.22614396 -0.57784545  0.04361032]
f_i = 0.31514088046246974

i = 6
x_i = [ 2.38705214  0.04739526  0.22906021 -0.57220899  0.04277752]
f_i = 0.313251725805429
```

```
i = 7
x_i = [ 2.43903372  0.04746071  0.2448277  -0.5199881   0.04265592]
f_i = 0.31240909739188943

i = 8
x_i = [ 2.54290583  0.04763839  0.28286388 -0.431771    0.04267665]
f_i = 0.3112141393930959

i = 9
x_i = [ 2.65751598  0.04785946  0.32979541 -0.36291643  0.04289731]
f_i = 0.30979083155613935

i = 10
x_i = [ 2.75773659  0.04807037  0.37338272 -0.33428837  0.04324972]
f_i = 0.307244154183504

i = 11
x_i = [ 2.91561544  0.04840977  0.44088241 -0.31654452  0.04388996]
f_i = 0.3030361174051982

i = 12
x_i = [ 3.0896001   0.04875473  0.51167015 -0.27548456  0.04447014]
f_i = 0.3005202365445257

i = 13
x_i = [ 3.20937499  0.04895842  0.55967972 -0.26367453  0.04494586]
f_i = 0.29862290220158677

i = 14
x_i = [ 3.27200366  0.04905342  0.58370243 -0.25888835  0.0451924 ]
f_i = 0.2977594248539742

i = 15
x_i = [ 3.31728461  0.04911341  0.60042281 -0.25555136  0.04536675]
f_i = 0.29721582022037235

i = 16
x_i = [ 3.32571199  0.04902366  0.59653591 -0.2789372   0.04551571]
f_i = 0.296995197222578

i = 17
x_i = [ 3.35738369  0.04900884  0.60202594 -0.2745725   0.04560096]
f_i = 0.29661607994768713

i = 18
x_i = [ 3.39639817  0.04885382  0.60225583 -0.26671659  0.04564811]
f_i = 0.29623979850099913
```

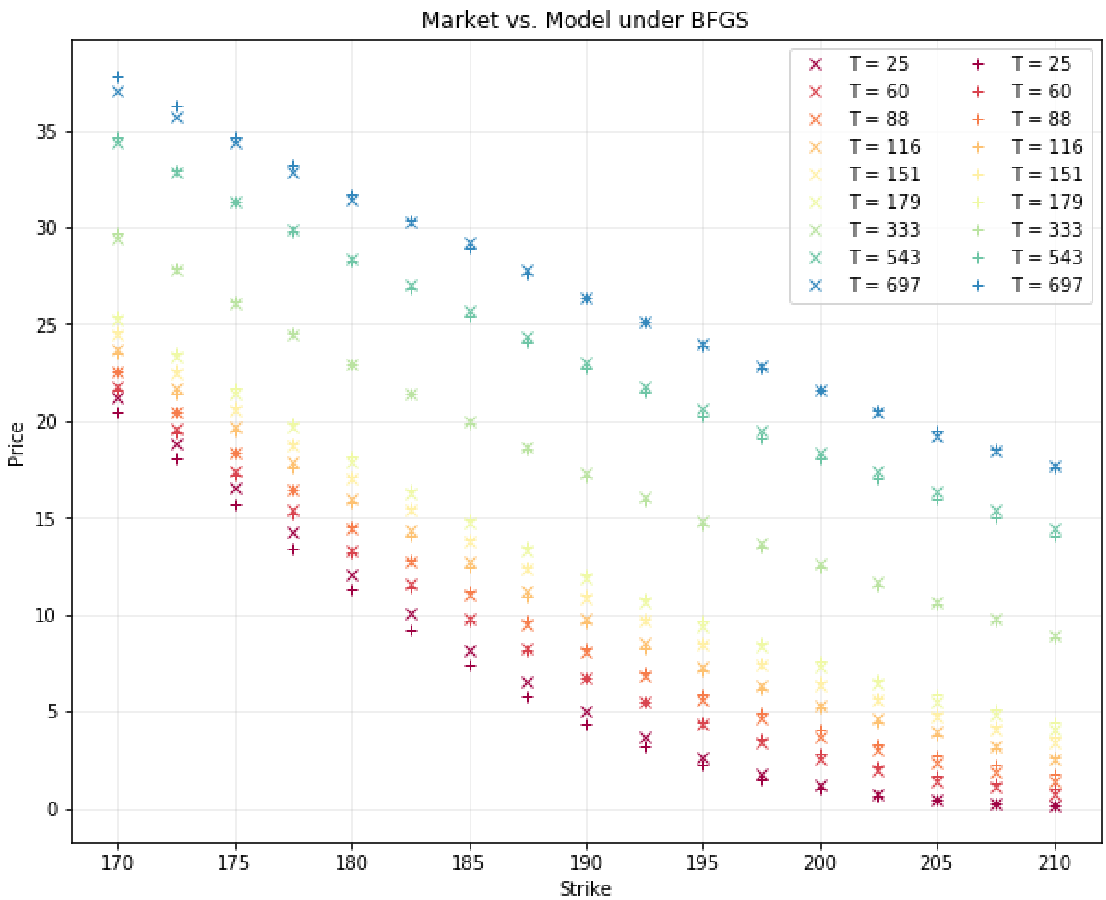
```
i = 19
x_i = [ 3.48513134  0.04867759  0.60355456 -0.26671592  0.04567234]
f_i = 0.29554387639279156

i = 20
x_i = [ 3.63793835  0.04837702  0.60186371 -0.2696817   0.04564052]
f_i = 0.29430818842741424
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.294308
         Iterations: 20
         Function evaluations: 287
         Gradient evaluations: 41
optimal params =
[ 3.63793835  0.04837702  0.60186371 -0.2696817   0.04564052]
f = 0.29430818842741424
```

In [ ]:
```
#optimal params =
#[ 3.63793835  0.04837702  0.60186371 -0.2696817   0.04564052]
#f = 0.29430818842741424
```

In [46]:
```
params_BFGS = xopt
lenT = len(maturities)
lenK = len(strikes)
modelPrices_BFGS = np.zeros((lenT, lenK))

for i in range(lenT):
    for j in range(lenK):
        T = maturities_years[i]
        K = strikes[j]
        [km, cT_km] = mfc.genericFFT(params_BFGS, S0, K, r, q, T, alpha, eta, n, model)
        modelPrices_BFGS[i,j] = cT_km[0]
```

In [47]:
```
# plot
fig = plt.figure(figsize=(10,8))
labels = []
colormap = cm.Spectral
plt.gca().set_color_cycle([colormap(i) for i in np.linspace(0, 0.9, len(maturities))])
for i in range(len(maturities)):
    plt.plot(strikes, marketPrices[i,:], 'x')
    labels.append('T = ' + str(maturities[i]))

for i in range(len(maturities)):
    plt.plot(strikes, modelPrices_BFGS[i,:], '+')
    labels.append('T = ' + str(maturities[i]))
plt.legend(labels, loc='upper right', ncol=2)
plt.grid(alpha=0.25)
```

```
plt.xlabel('Strike')
plt.ylabel('Price')
plt.title('Market vs. Model under BFGS')
#plt.savefig('MarketvsModel_BFGS.png')
plt.show()
```



2.2.1 Evaluate which Optimization yields best results: Loss Function

```
In [49]: # set 1: Brute-Force vs. Nelder-Mead
         params_BF = [1.7, 0.044000000000000004, 0.09, -0.62, 0.046]
         params_NM = [ 3.82879801,  0.04661798,  0.14376313, -0.89199663,  0.03933305]

In [50]: iArray = []
         rmseArray = []
         rmseMin = 1e10

         for i in mfc.myRange(-0.5, 1.5, 0.05):

             params = i*np.array(params_BF) + (1.0-i)*np.array(params_NM)
```

```
            print('')
            print(i)
            print(params)
            iArray.append(i)

            rmse = mfc.eValue(params, marketPrices, maturities_years, strikes, r, q, S0, alpha,
            rmseArray.append(rmse)
            if (rmse < rmseMin):
                rmseMin = rmse
                optimParams = params

        print(rmseMin)
        print(optimParams)
```

```
-0.5
[ 4.89319702  0.04792697  0.1706447  -1.02799494  0.03599957]

-0.45
[ 4.78675711  0.04779607  0.16795654 -1.01439511  0.03633292]

-0.4
[ 4.68031721  0.04766517  0.16526838 -1.00079528  0.03666627]

-0.35000000000000003
[ 4.57387731  0.04753427  0.16258023 -0.98719545  0.03699962]

-0.30000000000000004
[ 4.46743741  0.04740337  0.15989207 -0.97359562  0.03733297]

-0.25000000000000006
[ 4.36099751  0.04727248  0.15720391 -0.95999579  0.03766631]

-0.20000000000000007
[ 4.25455761  0.04714158  0.15451576 -0.94639596  0.03799966]

-0.15000000000000008
[ 4.14811771  0.04701068  0.1518276  -0.93279612  0.03833301]

-0.10000000000000007
[ 4.04167781  0.04687978  0.14913944 -0.91919629  0.03866635]

-0.05000000000000007
[ 3.93523791  0.04674888  0.14645129 -0.90559646  0.0389997 ]

-6.938893903907228e-17
[ 3.82879801  0.04661798  0.14376313 -0.89199663  0.03933305]
```

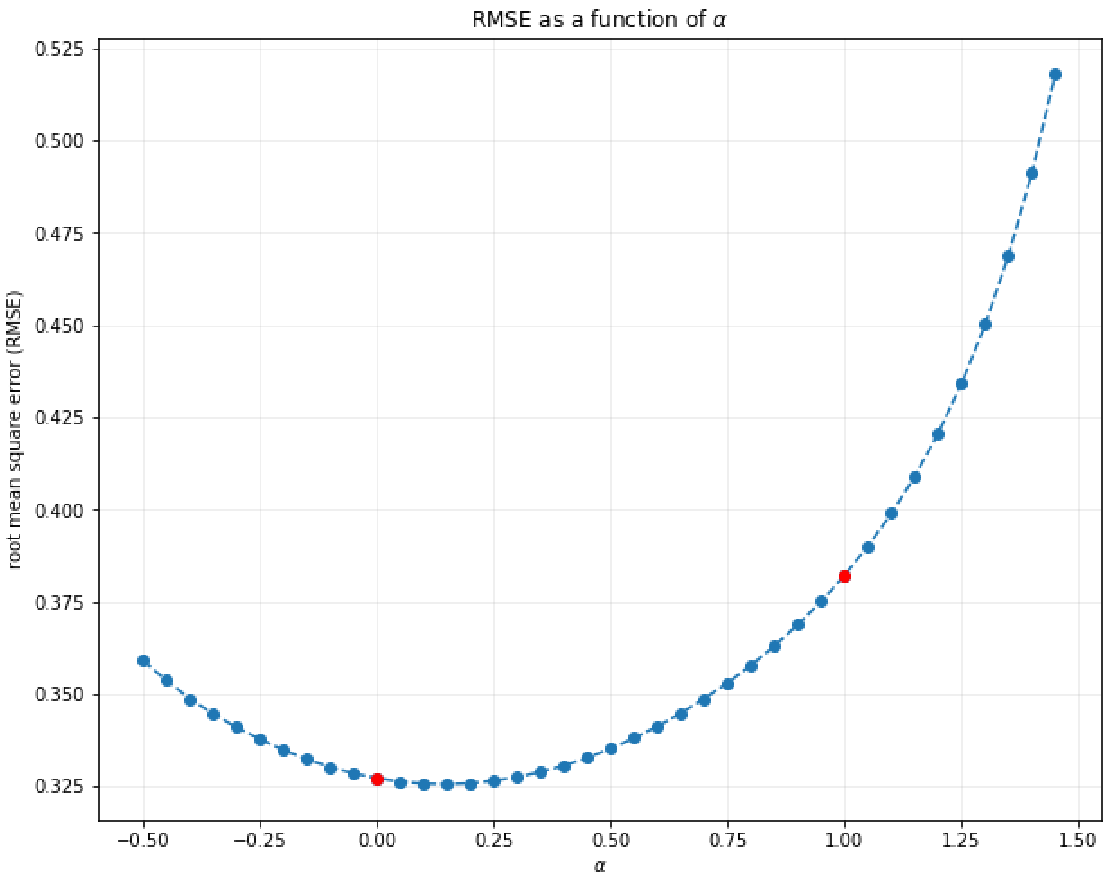
```
0.04999999999999993
[ 3.72235811  0.04648708  0.14107497 -0.8783968   0.0396664 ]

0.09999999999999994
[ 3.61591821  0.04635618  0.13838682 -0.86479697  0.03999975]

0.14999999999999994
[ 3.50947831  0.04622528  0.13569866 -0.85119714  0.04033309]

0.19999999999999996
[ 3.40303841  0.04609438  0.1330105  -0.8375973   0.04066644]

0.24999999999999994
[ 3.29659851  0.04596348  0.13032235 -0.82399747  0.04099979]

0.29999999999999993
[ 3.19015861  0.04583259  0.12763419 -0.81039764  0.04133314]

0.3499999999999999
[ 3.08371871  0.04570169  0.12494603 -0.79679781  0.04166648]

0.3999999999999999
[ 2.97727881  0.04557079  0.12225788 -0.78319798  0.04199983]

0.4499999999999999
[ 2.87083891  0.04543989  0.11956972 -0.76959815  0.04233318]

0.4999999999999999
[ 2.76439901  0.04530899  0.11688156 -0.75599832  0.04266652]

0.5499999999999999
[ 2.6579591   0.04517809  0.11419341 -0.74239848  0.04299987]

0.6
[ 2.5515192   0.04504719  0.11150525 -0.72879865  0.04333322]

0.65
[ 2.4450793   0.04491629  0.1088171  -0.71519882  0.04366657]

0.7000000000000001
[ 2.3386394   0.04478539  0.10612894 -0.70159899  0.04399992]

0.7500000000000001
[ 2.2321995   0.0446545   0.10344078 -0.68799916  0.04433326]

0.8000000000000002
[ 2.1257596   0.0445236   0.10075263 -0.67439933  0.04466661]
```

```
0.8500000000000002
[ 2.0193197   0.0443927   0.09806447 -0.66079949  0.04499996]

0.9000000000000002
[ 1.9128798   0.0442618   0.09537631 -0.64719966  0.04533331]

0.9500000000000003
[ 1.8064399   0.0441309   0.09268816 -0.63359983  0.04566665]

1.0000000000000002
[ 1.7    0.044  0.09   -0.62   0.046]

1.0500000000000003
[ 1.5935601   0.0438691   0.08731184 -0.60640017  0.04633335]

1.1000000000000003
[ 1.4871202   0.0437382   0.08462369 -0.59280034  0.0466667 ]

1.1500000000000004
[ 1.3806803   0.0436073   0.08193553 -0.57920051  0.04700004]

1.2000000000000004
[ 1.2742404   0.0434764   0.07924737 -0.56560067  0.04733339]

1.2500000000000004
[ 1.1678005   0.04334551  0.07655922 -0.55200084  0.04766674]

1.3000000000000005
[ 1.0613606   0.04321461  0.07387106 -0.53840101  0.04800008]

1.3500000000000005
[ 0.9549207   0.04308371  0.0711829  -0.52480118  0.04833343]

1.4000000000000006
[ 0.8484808   0.04295281  0.06849475 -0.51120135  0.04866678]

1.4500000000000006
[ 0.7420409   0.04282191  0.06580659 -0.49760152  0.04900013]
0.32557393153218905
[ 3.50947831  0.04622528  0.13569866 -0.85119714  0.04033309]
```

```
In [ ]: #optimal results are:
        #0.32557393153218905
        #[ 3.50947831  0.04622528  0.13569866 -0.85119714  0.04033309]
        #NM is better
```

```
In [51]: fig = plt.figure(figsize=(10,8))
         plt.plot(iArray, rmseArray, 'o--')
```

```
plt.plot(iArray[10], rmseArray[10], 'ro')
plt.plot(iArray[30], rmseArray[30], 'ro')
plt.grid(alpha=0.25)
plt.xlabel('$\\alpha$')
plt.ylabel('root mean square error (RMSE)')
plt.title('RMSE as a function of $\\alpha$')
#plt.savefig('NelderMeadvsBruteForce.png')
plt.show()
```



Concludes that NM Parameters yield better result

In [52]:
```
# set 2: Brute-Force vs. BFGS
params_BF = [1.7, 0.044000000000000004, 0.09, -0.62, 0.046]
params_BFGS = [ 3.63793835 , 0.04837702 , 0.60186371 ,-0.2696817 ,  0.04564052]
```

In [53]:
```
iArray = []
rmseArray = []
rmseMin = 1e10

for i in mfc.myRange(-0.5, 1.5, 0.05):
```

```
        params = i*np.array(params_BF) + (1.0-i)*np.array(params_BFGS)
        print('')
        print(i)
        print(params)
        iArray.append(i)

        rmse = mfc.eValue(params, marketPrices, maturities_years, strikes, r, q, S0, alpha,
        rmseArray.append(rmse)
        if (rmse < rmseMin):
            rmseMin = rmse
            optimParams = params

    print(rmseMin)
    print(optimParams)
```

```
-0.5
[ 4.60690753  0.05056553  0.85779557 -0.09452255  0.04546078]

-0.45
[ 4.51001061  0.05034668  0.83220238 -0.11203846  0.04547875]

-0.4
[ 4.41311369  0.05012783  0.80660919 -0.12955438  0.04549673]

-0.35000000000000003
[ 4.31621677  0.04990898  0.78101601 -0.1470703   0.0455147 ]

-0.30000000000000004
[ 4.21931986  0.04969013  0.75542282 -0.16458621  0.04553268]

-0.25000000000000006
[ 4.12242294  0.04947127  0.72982964 -0.18210213  0.04555065]

-0.20000000000000007
[ 4.02552602  0.04925242  0.70423645 -0.19961804  0.04556862]

-0.15000000000000008
[ 3.9286291   0.04903357  0.67864327 -0.21713396  0.0455866 ]

-0.10000000000000007
[ 3.83173219  0.04881472  0.65305008 -0.23464987  0.04560457]

-0.05000000000000007
[ 3.73483527  0.04859587  0.6274569  -0.25216579  0.04562255]

-6.938893903907228e-17
[ 3.63793835  0.04837702  0.60186371 -0.2696817   0.04564052]
```

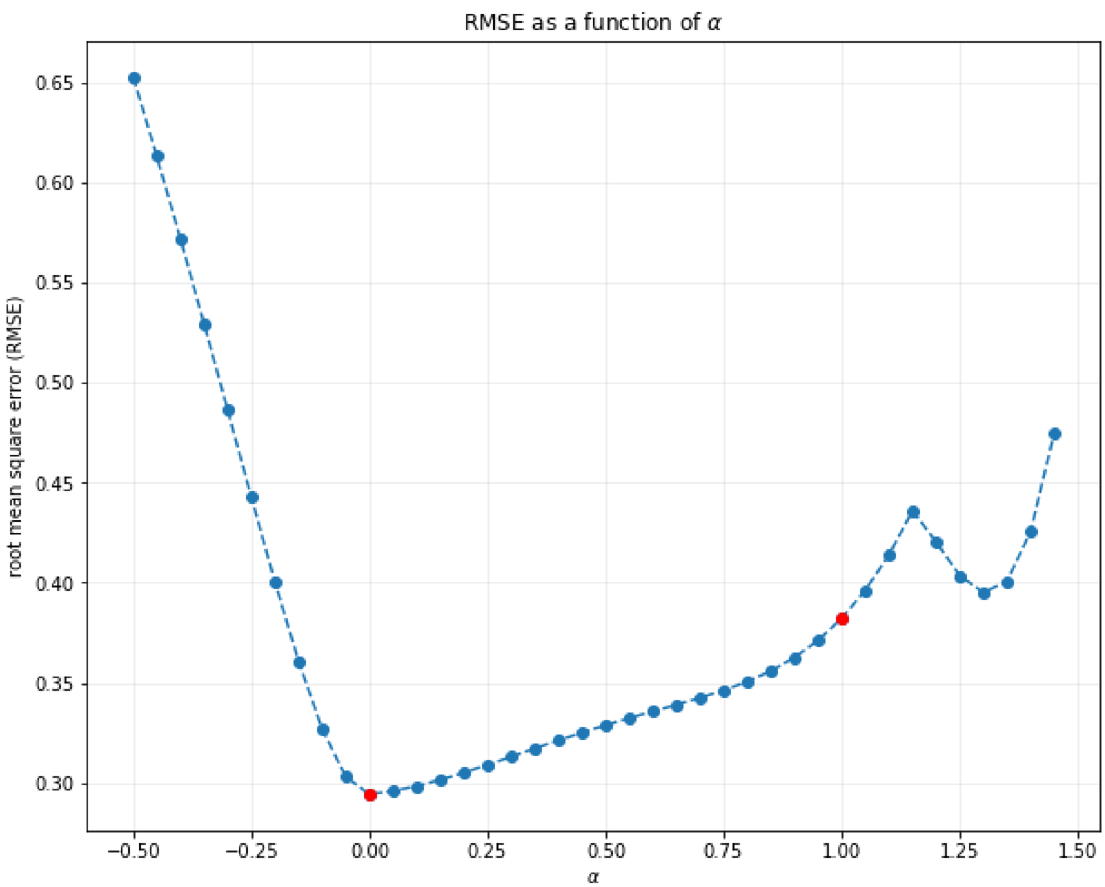
```
0.04999999999999993
[ 3.54104143  0.04815817  0.57627052 -0.28719762  0.04565849]

0.09999999999999994
[ 3.44414452  0.04793932  0.55067734 -0.30471353  0.04567647]

0.14999999999999994
[ 3.3472476   0.04772047  0.52508415 -0.32222945  0.04569444]

0.19999999999999996
[ 3.25035068  0.04750162  0.49949097 -0.33974536  0.04571242]

0.24999999999999994
[ 3.15345376  0.04728277  0.47389778 -0.35726127  0.04573039]

0.29999999999999993
[ 3.05655685  0.04706391  0.4483046  -0.37477719  0.04574836]

0.3499999999999999
[ 2.95965993  0.04684506  0.42271141 -0.39229311  0.04576634]

0.3999999999999999
[ 2.86276301  0.04662621  0.39711823 -0.40980902  0.04578431]

0.4499999999999999
[ 2.76586609  0.04640736  0.37152504 -0.42732493  0.04580229]

0.4999999999999999
[ 2.66896918  0.04618851  0.34593186 -0.44484085  0.04582026]

0.5499999999999999
[ 2.57207226  0.04596966  0.32033867 -0.46235677  0.04583823]

0.6
[ 2.47517534  0.04575081  0.29474548 -0.47987268  0.04585621]

0.65
[ 2.37827842  0.04553196  0.2691523  -0.4973886   0.04587418]

0.7000000000000001
[ 2.2813815   0.04531311  0.24355911 -0.51490451  0.04589216]

0.7500000000000001
[ 2.18448459  0.04509425  0.21796593 -0.53242043  0.04591013]

0.8000000000000002
[ 2.08758767  0.0448754   0.19237274 -0.54993634  0.0459281 ]
```

```
0.8500000000000002
[ 1.99069075  0.04465655  0.16677956 -0.56745226  0.04594608]

0.9000000000000002
[ 1.89379383  0.0444377   0.14118637 -0.58496817  0.04596405]

0.9500000000000003
[ 1.79689692  0.04421885  0.11559319 -0.60248409  0.04598203]

1.0000000000000002
[ 1.7    0.044  0.09  -0.62   0.046]

1.0500000000000003
[ 1.60310308  0.04378115  0.06440681 -0.63751592  0.04601797]

1.1000000000000003
[ 1.50620616  0.0435623   0.03881363 -0.65503183  0.04603595]

1.1500000000000004
[ 1.40930925  0.04334345  0.01322044 -0.67254775  0.04605392]

1.2000000000000004
[ 1.31241233  0.0431246  -0.01237274 -0.69006366  0.0460719 ]

1.2500000000000004
[ 1.21551541  0.04290575 -0.03796593 -0.70757958  0.04608987]

1.3000000000000005
[ 1.11861849  0.04268689 -0.06355911 -0.72509549  0.04610784]

1.3500000000000005
[ 1.02172158  0.04246804 -0.0891523  -0.74261141  0.04612582]

1.4000000000000006
[ 0.92482466  0.04224919 -0.11474548 -0.76012732  0.04614379]

1.4500000000000006
[ 0.82792774  0.04203034 -0.14033867 -0.77764324  0.04616177]
0.29430818581882634
[ 3.63793835  0.04837702  0.60186371 -0.2696817   0.04564052]
```

In [ ]:
```
#optimal results are
#0.29430818581882634
#[ 3.63793835  0.04837702  0.60186371 -0.2696817   0.04564052]
```

In [54]:
```
fig = plt.figure(figsize=(10,8))
plt.plot(iArray, rmseArray, 'o--')
```

```
plt.plot(iArray[10], rmseArray[10], 'ro')
plt.plot(iArray[30], rmseArray[30], 'ro')
plt.grid(alpha=0.25)
plt.xlabel('$\\alpha$')
plt.ylabel('root mean square error (RMSE)')
plt.title('RMSE as a function of $\\alpha$')
#plt.savefig('NelderMeadvsBruteForce.png')
plt.show()
```



In [ ]:
```
#BFGD yields better result
```

In [55]:
```
# set 3: NelderMead vs. BFGS
params_NM = [ 3.82879801,  0.04661798,  0.14376313, -0.89199663,  0.03933305]
params_BFGS = [ 3.63793835 , 0.04837702 , 0.60186371 ,-0.2696817 ,  0.04564052]
```

In [56]:
```
iArray = []
rmseArray = []
rmseMin = 1e10
```

```
for i in mfc.myRange(-0.5, 1.5, 0.05):

    params = i*np.array(params_NM) + (1.0-i)*np.array(params_BFGS)
    print('')
    print(i)
    print(params)
    iArray.append(i)

    rmse = mfc.eValue(params, marketPrices, maturities_years, strikes, r, q, S0, alpha,
    rmseArray.append(rmse)
    if (rmse < rmseMin):
        rmseMin = rmse
        optimParams = params

print(rmseMin)
print(optimParams)
```

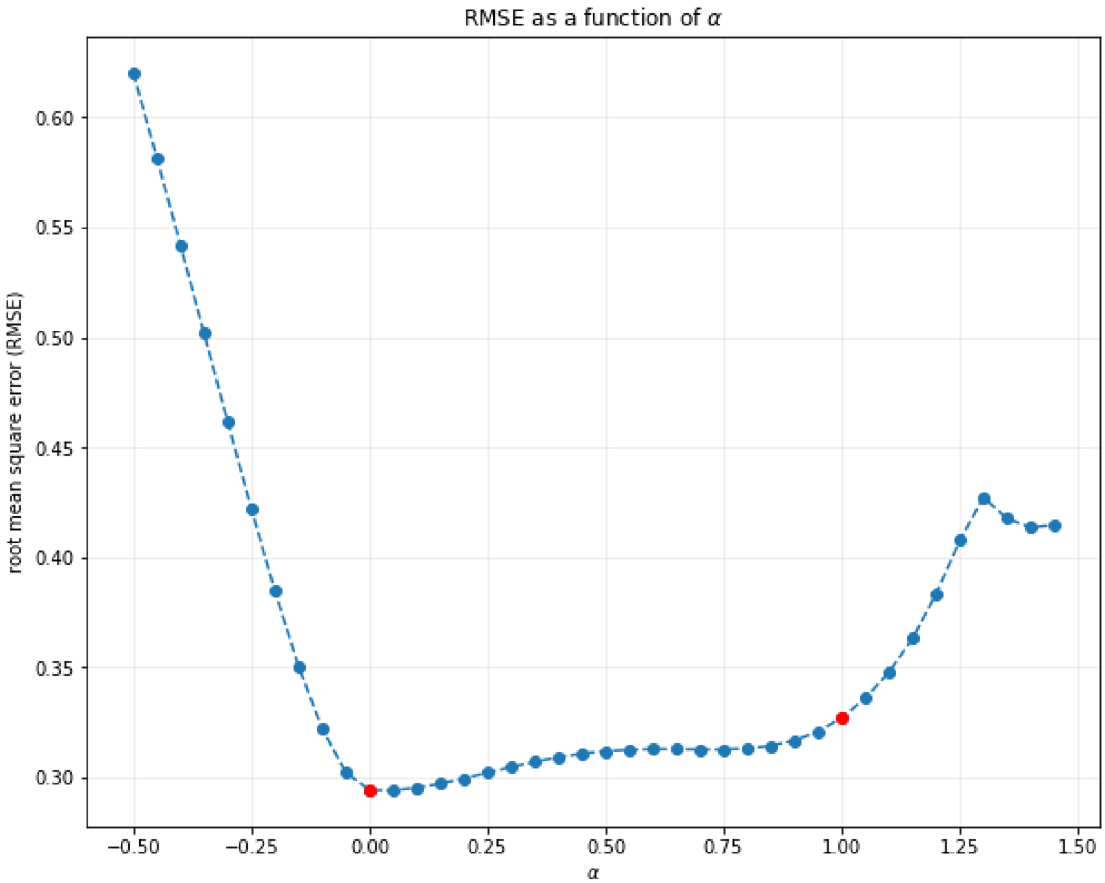
```
-0.5
[3.54250852 0.04925654 0.830914   0.04147576 0.04879425]

-0.45
[3.5520515  0.04916859 0.80800897 0.01036002 0.04847888]

-0.4
[ 3.56159449  0.04908064  0.78510394 -0.02075573  0.04816351]

-0.35000000000000003
[ 3.57113747  0.04899268  0.76219891 -0.05187147  0.04784813]

-0.30000000000000004
[ 3.58068045  0.04890473  0.73929388 -0.08298722  0.04753276]

-0.25000000000000006
[ 3.59022344  0.04881678  0.71638886 -0.11410297  0.04721739]

-0.20000000000000007
[ 3.59976642  0.04872883  0.69348383 -0.14521871  0.04690201]

-0.15000000000000008
[ 3.6093094   0.04864088  0.6705788  -0.17633446  0.04658664]

-0.10000000000000007
[ 3.61885238  0.04855292  0.64767377 -0.20745021  0.04627127]

-0.05000000000000007
[ 3.62839537  0.04846497  0.62476874 -0.23856595  0.04595589]
```

```
-6.938893903907228e-17
[ 3.63793835  0.04837702  0.60186371 -0.2696817   0.04564052]

0.04999999999999993
[ 3.64748133  0.04828907  0.57895868 -0.30079745  0.04532515]

0.09999999999999994
[ 3.65702432  0.04820112  0.55605365 -0.33191319  0.04500977]

0.14999999999999994
[ 3.6665673   0.04811316  0.53314862 -0.36302894  0.0446944 ]

0.19999999999999996
[ 3.67611028  0.04802521  0.51024359 -0.39414469  0.04437903]

0.24999999999999994
[ 3.68565327  0.04793726  0.48733857 -0.42526043  0.04406365]

0.29999999999999993
[ 3.69519625  0.04784931  0.46443354 -0.45637618  0.04374828]

0.3499999999999999
[ 3.70473923  0.04776136  0.44152851 -0.48749193  0.04343291]

0.3999999999999999
[ 3.71428221  0.0476734   0.41862348 -0.51860767  0.04311753]

0.4499999999999999
[ 3.7238252   0.04758545  0.39571845 -0.54972342  0.04280216]

0.4999999999999999
[ 3.73336818  0.0474975   0.37281342 -0.58083916  0.04248678]

0.5499999999999999
[ 3.74291116  0.04740955  0.34990839 -0.61195491  0.04217141]

0.6
[ 3.75245415  0.0473216   0.32700336 -0.64307066  0.04185604]

0.65
[ 3.76199713  0.04723364  0.30409833 -0.6741864   0.04154066]

0.7000000000000001
[ 3.77154011  0.04714569  0.2811933  -0.70530215  0.04122529]

0.7500000000000001
[ 3.7810831   0.04705774  0.25828827 -0.7364179   0.04090992]
```

```
0.8000000000000002
[ 3.79062608  0.04696979  0.23538325 -0.76753364  0.04059454]

0.8500000000000002
[ 3.80016906  0.04688184  0.21247822 -0.79864939  0.04027917]

0.9000000000000002
[ 3.80971204  0.04679388  0.18957319 -0.82976514  0.0399638 ]

0.9500000000000003
[ 3.81925503  0.04670593  0.16666816 -0.86088088  0.03964842]

1.0000000000000002
[ 3.82879801  0.04661798  0.14376313 -0.89199663  0.03933305]

1.0500000000000003
[ 3.83834099  0.04653003  0.1208581  -0.92311238  0.03901768]

1.1000000000000003
[ 3.84788398  0.04644208  0.09795307 -0.95422812  0.0387023 ]

1.1500000000000004
[ 3.85742696  0.04635412  0.07504804 -0.98534387  0.03838693]

1.2000000000000004
[ 3.86696994  0.04626617  0.05214301 -1.01645962  0.03807156]

1.2500000000000004
[ 3.87651293  0.04617822  0.02923798 -1.04757536  0.03775618]

1.3000000000000005
[ 3.88605591  0.04609027  0.00633296 -1.07869111  0.03744081]

1.3500000000000005
[ 3.89559889  0.04600232 -0.01657207 -1.10980686  0.03712544]

1.4000000000000006
[ 3.90514187  0.04591436 -0.0394771  -1.1409226   0.03681006]

1.4500000000000006
[ 3.91468486  0.04582641 -0.06238213 -1.17203835  0.03649469]
0.29424796472109044
[ 3.64748133  0.04828907  0.57895868 -0.30079745  0.04532515]
```

In [ ]:
```
#Optimal results are:
#0.29424796472109044
#[ 3.64748133  0.04828907  0.57895868 -0.30079745  0.04532515]
```

```
In [57]: fig = plt.figure(figsize=(10,8))
         plt.plot(iArray, rmseArray, 'o--')
         plt.plot(iArray[10], rmseArray[10], 'ro')
         plt.plot(iArray[30], rmseArray[30], 'ro')
         plt.grid(alpha=0.25)
         plt.xlabel('$\\alpha$')
         plt.ylabel('root mean square error (RMSE)')
         plt.title('RMSE as a function of $\\alpha$')
         #plt.savefig('NelderMeadvsBruteForce.png')
         plt.show()
```




```
In [ ]: #BFGS is the best
```

2.3 3. Plot Model Price Surface

```
In [64]: data = [go.Surface(z=modelPrices.tolist(), colorscale='Viridis'),
                 go.Surface(z=marketPrices.tolist(), colorscale='Portland', showscale=False, opa

         layout = go.Layout(
```

```
        width=800,
        height=700,
        autosize=False,
        title='Calibrated Call Premium Surface Heston',
        scene=dict(
            xaxis=dict(
                gridcolor='rgb(255, 255, 255)',
                zerolinecolor='rgb(255, 255, 255)',
                showbackground=True,
                backgroundcolor='rgb(230, 230,230)'
            ),
            yaxis=dict(
                gridcolor='rgb(255, 255, 255)',
                zerolinecolor='rgb(255, 255, 255)',
                showbackground=True,
                backgroundcolor='rgb(230, 230,230)'
            ),
            zaxis=dict(
                gridcolor='rgb(255, 255, 255)',
                zerolinecolor='rgb(255, 255, 255)',
                showbackground=True,
                backgroundcolor='rgb(230, 230,230)'
            ),
            aspectratio = dict( x=1, y=1, z=0.7 ),
            aspectmode = 'manual'
        )
    )

    fig = dict(data=data, layout=layout)
```

```
In [65]: py.iplot(fig, filename='Calibrated-Call-Premium-Surface-Heston')
```

```
Aw, snap! We didn't get a username with your request.

Don't have an account? https://plot.ly/api_signup

Questions? accounts@plot.ly
```

```
        ---------------------------------------------------------------------------

        PlotlyError                               Traceback (most recent call last)

        <ipython-input-65-7d4ed6937f16> in <module>()
    ----> 1 py.iplot(fig, filename='Calibrated-Call-Premium-Surface-Heston')
```

```
~/anaconda3/lib/python3.6/site-packages/plotly/plotly/plotly.py in iplot(figure_or_data,
    168         embed_options['height'] = str(embed_options['height']) + 'px'
    169
--> 170     return tools.embed(url, **embed_options)
    171
    172


~/anaconda3/lib/python3.6/site-packages/plotly/tools.py in embed(file_owner_or_url, file
    399         else:
    400             url = file_owner_or_url
--> 401         return PlotlyDisplay(url, width, height)
    402     else:
    403         if (get_config_defaults()['plotly_domain']


~/anaconda3/lib/python3.6/site-packages/plotly/tools.py in __init__(self, url, width, he
   1479         def __init__(self, url, width, height):
   1480             self.resource = url
-> 1481             self.embed_code = get_embed(url, width=width, height=height)
   1482             super(PlotlyDisplay, self).__init__(data=self.embed_code)
   1483


~/anaconda3/lib/python3.6/site-packages/plotly/tools.py in get_embed(file_owner_or_url,
    304                 "'{1}'."
    305                 "\nRun help on this function for more information."
--> 306                 "".format(url, plotly_rest_url))
    307         urlsplit = six.moves.urllib.parse.urlparse(url)
    308         file_owner = urlsplit.path.split('/')[1].split('~')[1]


PlotlyError: Because you didn't supply a 'file_id' in the call, we're assuming you're tr
Run help on this function for more information.
```

3 II. Local Volatility Surface

** Explicitly compute local volatility for each point in our grid **

3.1 1. Calculate Finite differences

In [11]:
```python
deltaK = 5
deltaT = 1/52
```

In [15]:
```python
params_BFGS = [ 3.63793835 , 0.04837702 , 0.60186371 ,-0.2696817 ,  0.04564052]
lenT = len(maturities)
lenK = len(strikes)
```

```
modelPrices = np.zeros((lenT, lenK))

for i in range(lenT):
    for j in range(lenK):
        T = maturities_years[i]
        K = strikes[j]
        [km, cT_km] = mfc.genericFFT(params_BFGS, S0, K, r, q, T, alpha, eta, n, model)
        modelPrices[i,j] = cT_km[0]
```

In [16]:
```
modelPrices.shape
```

Out[16]:
```
(98, 9)
```

In [17]:
```
# v_j,k Option prices for all points on the grid
modelPrices_df = pd.DataFrame(modelPrices, columns = strikes)
modelPrices_df.head()
```

Out[17]:

	170.0	175.0	180.0	185.0	190.0	195.0	200.0	\
0	20.440630	15.703254	11.277736	7.401358	4.333137	2.223275	0.997278	
1	20.647614	16.010738	11.708753	7.940705	4.911118	2.739629	1.379419	
2	20.873399	16.324738	12.121172	8.433477	5.430860	3.213591	1.751035	
3	21.109894	16.638245	12.515266	8.890375	5.908046	3.654954	2.110730	
4	21.351979	16.947915	12.892556	9.318729	6.352544	4.070433	2.458881	

	205.0	210.0
0	0.396668	0.143132
1	0.634839	0.271969
2	0.888469	0.426244
3	1.149274	0.597788
4	1.412864	0.781102

dC/dT for every point on the grid

In [18]:
```
dcdT = (modelPrices_df.diff()/deltaT).shift(-1)
dcdT.head()
```

Out[18]:

	170.0	175.0	180.0	185.0	190.0	195.0	\
0	10.763172	15.989144	22.412853	28.046028	30.055008	26.850400	
1	11.740827	16.328011	21.445817	25.624179	27.026631	24.646026	
2	12.297724	16.302395	20.492861	23.758661	24.813646	22.950895	
3	12.588442	16.102817	19.619106	22.274414	23.113899	21.604901	
4	12.712139	15.823386	18.836836	21.063130	21.760838	20.508010	

	200.0	205.0	210.0
0	19.871359	12.384895	6.699522
1	19.324051	13.188751	8.022271
2	18.704099	13.561850	8.920292
3	18.103895	13.706706	9.532348
4	17.551166	13.727735	9.953967

In [19]: (16.010738 - 15.703254)/ deltaT

Out[19]: 15.989168000000028

dC/dK for every point on the grid

```
In [20]: dcdK = (modelPrices_df.diff(axis=1, periods = 2)/(2*deltaK)).shift(-1,axis=1)
         dcdK.head()
```

```
Out[20]:    170.0     175.0     180.0     185.0     190.0     195.0     200.0  \
         0    NaN -0.916289 -0.830190 -0.694460 -0.517808 -0.333586 -0.182661
         1    NaN -0.893886 -0.807003 -0.679763 -0.520108 -0.353170 -0.210479
         2    NaN -0.875223 -0.789126 -0.669031 -0.521989 -0.367983 -0.232512
         3    NaN -0.859463 -0.774787 -0.660722 -0.523542 -0.379732 -0.250568
         4    NaN -0.845942 -0.762919 -0.654001 -0.524830 -0.389366 -0.265757

               205.0  210.0
         0 -0.085415    NaN
         1 -0.110745    NaN
         2 -0.132479    NaN
         3 -0.151294    NaN
         4 -0.167778    NaN
```

In [22]: (11.277736 - 20.440630)/(2*deltaK)

Out[22]: -0.9162893999999998

```
In [23]: # For vol surface calculation
         dcdK_v = dcdK
         for i in modelPrices_df.columns:
             dcdK_v[i] = i * dcdK[i]*(r-q)
```

In [24]: dcdK_v.head()

```
Out[24]:    170.0     175.0     180.0     185.0     190.0     195.0     200.0  \
         0    NaN -3.126837 -2.913965 -2.505264 -1.918480 -1.268460 -0.712377
         1    NaN -3.050386 -2.832581 -2.452247 -1.926999 -1.342928 -0.820868
         2    NaN -2.986697 -2.769832 -2.413530 -1.933968 -1.399253 -0.906798
         3    NaN -2.932917 -2.719503 -2.383554 -1.939723 -1.443930 -0.977215
         4    NaN -2.886778 -2.677844 -2.359309 -1.944494 -1.480565 -1.036452

               205.0  210.0
         0 -0.341445    NaN
         1 -0.442703    NaN
         2 -0.529586    NaN
         3 -0.604799    NaN
         4 -0.670692    NaN
```

In [25]: (r-q)*175*(-0.916289)

Out[25]: -3.1268362125

d2C/dK2 for every point on the grid

```
In [26]: # For d2C/dK2: - 2*v_j,k
         modelPrices_df_neg2 = modelPrices_df*(-2)

In [27]: modelPrices_df_neg2.head()

Out[27]:         170.0      175.0      180.0      185.0      190.0     195.0     200.0  \
         0 -40.881260 -31.406508 -22.555472 -14.802716  -8.666273 -4.446550 -1.994555
         1 -41.295228 -32.021475 -23.417505 -15.881410  -9.822235 -5.479257 -2.758838
         2 -41.746798 -32.649475 -24.242344 -16.866955 -10.861721 -6.427181 -3.502071
         3 -42.219787 -33.276491 -25.030531 -17.780750 -11.816092 -7.309908 -4.221459
         4 -42.703958 -33.895830 -25.785112 -18.637458 -12.705088 -8.140866 -4.917763

              205.0     210.0
         0 -0.793336 -0.286264
         1 -1.269678 -0.543938
         2 -1.776938 -0.852487
         3 -2.298547 -1.195575
         4 -2.825728 -1.562204

In [28]: d2cdK2 = pd.DataFrame()
         for i in range(len(modelPrices_df.columns)):
             try:
                 d2cdK2[modelPrices_df.columns[i]] = (modelPrices_df[modelPrices_df.columns[i-1]
             except:
                 d2cdK2[modelPrices_df.columns[i]] = np.repeat(np.nan, len(modelPrices_df))

In [29]: modelPrices_df.shape == modelPrices_df_neg2.shape == d2cdK2.shape

Out[29]: True

In [30]: d2cdK2[modelPrices_df.columns[0]] = np.repeat(np.nan, len(modelPrices_df))
         d2cdK2.head()

Out[30]:    170.0     175.0     180.0     185.0     190.0     195.0     200.0  \
         0    NaN  0.012474  0.021966  0.032326  0.038334  0.035355  0.025016
         1    NaN  0.013396  0.021357  0.029538  0.034324  0.032451  0.024625
         2    NaN  0.013804  0.020635  0.027403  0.031414  0.030189  0.024000
         3    NaN  0.013947  0.019924  0.025702  0.029169  0.028355  0.023311
         4    NaN  0.013948  0.019261  0.024306  0.027363  0.026822  0.022621

              205.0  210.0
         0  0.013883    NaN
         1  0.015268    NaN
         2  0.016014    NaN
         3  0.016399    NaN
         4  0.016570    NaN

In [349]: (15.627181 - 2* 17.977095 + 20.385484)/(deltaK**2)
```

Out[349]: 0.00935600000000079

In [31]:
```
# For vol surface calculation
d2cdK2_v = d2cdK2
for i in modelPrices_df.columns:
    d2cdK2_v[i] = i**2 * d2cdK2[i] * 1/2
```

In [32]:
```
d2cdK2_v.head()
```

Out[32]:
```
   170.0       175.0       180.0       185.0       190.0       195.0  \
0    NaN  191.012999  355.842620  553.183163  691.935619  672.179047
1    NaN  205.120967  345.991343  505.476229  619.546941  616.977958
2    NaN  211.371079  334.284423  468.935647  567.020655  573.960416
3    NaN  213.559529  322.761671  439.833656  526.509011  539.093772
4    NaN  213.582058  312.032542  415.931126  493.901238  509.960650

        200.0       205.0  210.0
0  500.310079  291.715554    NaN
1  492.503585  320.827411    NaN
2  479.990887  336.486917    NaN
3  466.214671  344.579821    NaN
4  452.427236  348.181557    NaN
```

In [366]:
```
0.009356*172.5**2*0.5
```

Out[366]: 139.1997375

In [33]:
```
dcdT.shape == modelPrices_df.shape == dcdK_v.shape == d2cdK2_v.shape
```

Out[33]: True

3.2 2. Calculate Local Volatility Surface

In [34]:
```
# Vol Surface
vol_surface = ((dcdT + dcdK_v + q*modelPrices_df)/d2cdK2_v)**0.5
```

In [36]:
```
vol_surface
```

Out[36]:
```
   170.0     175.0     180.0     185.0     190.0     195.0     200.0  \
0    NaN  0.260285  0.234424  0.215029  0.201730  0.195128  0.195715
1    NaN  0.255188  0.232306  0.214290  0.201377  0.194401  0.193865
2    NaN  0.251760  0.230649  0.213561  0.200994  0.193848  0.192605
3    NaN  0.249115  0.229245  0.212896  0.200680  0.193473  0.191724
4    NaN  0.246914  0.228019  0.212322  0.200465  0.193265  0.191127
5    NaN  0.245008  0.226942  0.211848  0.200357  0.193206  0.190756
6    NaN  0.243322  0.225995  0.211472  0.200347  0.193273  0.190567
7    NaN  0.241815  0.225169  0.211190  0.200424  0.193445  0.190530
8    NaN  0.240463  0.224452  0.210993  0.200577  0.193705  0.190616
9    NaN  0.239247  0.223836  0.210872  0.200794  0.194034  0.190802
```

10	NaN	0.238155	0.223310	0.210817	0.201062	0.194420	0.191071
11	NaN	0.237175	0.222866	0.210820	0.201374	0.194851	0.191406
12	NaN	0.236295	0.222495	0.210872	0.201719	0.195316	0.191793
13	NaN	0.235508	0.222189	0.210966	0.202092	0.195807	0.192223
14	NaN	0.234804	0.221941	0.211096	0.202486	0.196317	0.192686
15	NaN	0.234176	0.221745	0.211255	0.202896	0.196841	0.193174
16	NaN	0.233616	0.221593	0.211439	0.203317	0.197374	0.193681
17	NaN	0.233118	0.221481	0.211644	0.203746	0.197913	0.194201
18	NaN	0.232676	0.221403	0.211866	0.204180	0.198453	0.194731
19	NaN	0.232284	0.221356	0.212102	0.204617	0.198992	0.195266
20	NaN	0.231937	0.221336	0.212348	0.205054	0.199529	0.195804
21	NaN	0.231630	0.221338	0.212603	0.205490	0.200062	0.196343
22	NaN	0.231361	0.221361	0.212865	0.205924	0.200590	0.196880
23	NaN	0.231124	0.221402	0.213131	0.206353	0.201110	0.197413
24	NaN	0.230917	0.221457	0.213401	0.206779	0.201623	0.197942
25	NaN	0.230736	0.221527	0.213673	0.207199	0.202129	0.198465
26	NaN	0.230580	0.221607	0.213946	0.207613	0.202625	0.198981
27	NaN	0.230446	0.221697	0.214220	0.208021	0.203113	0.199490
28	NaN	0.230331	0.221796	0.214493	0.208423	0.203591	0.199991
29	NaN	0.230234	0.221903	0.214765	0.208817	0.204061	0.200484
..	...	...	...	...	...	...	...
68	NaN	0.231510	0.227018	0.223045	0.219547	0.216487	0.213834
69	NaN	0.231583	0.227136	0.223203	0.219736	0.216701	0.214065
70	NaN	0.231656	0.227254	0.223358	0.219923	0.216912	0.214293
71	NaN	0.231729	0.227372	0.223512	0.220107	0.217119	0.214518
72	NaN	0.231803	0.227488	0.223664	0.220288	0.217323	0.214739
73	NaN	0.231877	0.227603	0.223815	0.220467	0.217525	0.214956
74	NaN	0.231951	0.227717	0.223963	0.220644	0.217723	0.215171
75	NaN	0.232025	0.227831	0.224110	0.220818	0.217919	0.215382
76	NaN	0.232100	0.227944	0.224255	0.220989	0.218111	0.215590
77	NaN	0.232174	0.228056	0.224398	0.221159	0.218301	0.215795
78	NaN	0.232249	0.228167	0.224540	0.221326	0.218488	0.215997
79	NaN	0.232324	0.228277	0.224681	0.221491	0.218673	0.216197
80	NaN	0.232399	0.228387	0.224819	0.221654	0.218855	0.216393
81	NaN	0.232474	0.228495	0.224957	0.221815	0.219035	0.216587
82	NaN	0.232549	0.228603	0.225093	0.221974	0.219212	0.216778
83	NaN	0.232624	0.228710	0.225227	0.222131	0.219387	0.216967
84	NaN	0.232699	0.228817	0.225360	0.222286	0.219560	0.217153
85	NaN	0.232774	0.228923	0.225492	0.222439	0.219731	0.217337
86	NaN	0.232849	0.229028	0.225622	0.222591	0.219899	0.217518
87	NaN	0.232925	0.229132	0.225751	0.222741	0.220066	0.217698
88	NaN	0.233000	0.229236	0.225879	0.222889	0.220230	0.217874
89	NaN	0.233075	0.229339	0.226006	0.223035	0.220392	0.218049
90	NaN	0.233150	0.229441	0.226131	0.223180	0.220553	0.218222
91	NaN	0.233225	0.229543	0.226256	0.223323	0.220711	0.218392
92	NaN	0.233300	0.229644	0.226379	0.223465	0.220868	0.218560
93	NaN	0.233375	0.229744	0.226501	0.223605	0.221023	0.218727
94	NaN	0.233450	0.229844	0.226622	0.223744	0.221176	0.218891

95	NaN	0.233525	0.229943	0.226742	0.223881	0.221328	0.219054
96	NaN	0.233600	0.230042	0.226861	0.224017	0.221478	0.219215
97	NaN	NaN	NaN	NaN	NaN	NaN	NaN

	205.0	210.0
0	0.203204	NaN
1	0.199345	NaN
2	0.196834	NaN
3	0.195037	NaN
4	0.193703	NaN
5	0.192708	NaN
6	0.191978	NaN
7	0.191465	NaN
8	0.191130	NaN
9	0.190944	NaN
10	0.190883	NaN
11	0.190925	NaN
12	0.191054	NaN
13	0.191255	NaN
14	0.191516	NaN
15	0.191825	NaN
16	0.192174	NaN
17	0.192555	NaN
18	0.192962	NaN
19	0.193390	NaN
20	0.193833	NaN
21	0.194288	NaN
22	0.194752	NaN
23	0.195221	NaN
24	0.195694	NaN
25	0.196169	NaN
26	0.196644	NaN
27	0.197117	NaN
28	0.197587	NaN
29	0.198054	NaN
..	...	...
68	0.211561	NaN
69	0.211803	NaN
70	0.212042	NaN
71	0.212277	NaN
72	0.212508	NaN
73	0.212736	NaN
74	0.212961	NaN
75	0.213182	NaN
76	0.213400	NaN
77	0.213615	NaN
78	0.213827	NaN
79	0.214037	NaN

```
80  0.214243    NaN
81  0.214446    NaN
82  0.214647    NaN
83  0.214845    NaN
84  0.215040    NaN
85  0.215233    NaN
86  0.215424    NaN
87  0.215612    NaN
88  0.215798    NaN
89  0.215981    NaN
90  0.216162    NaN
91  0.216341    NaN
92  0.216518    NaN
93  0.216693    NaN
94  0.216865    NaN
95  0.217036    NaN
96  0.217205    NaN
97       NaN    NaN

[98 rows x 9 columns]
```

```
In [101]: vol_surface_new = vol_surface[[175.0, 180.0,185.0,190.0,195.0,200.0,205.0]]
          #pd.DataFrame.to_csv(vol_surface)
```

```
In [102]: vol_surface_new = vol_surface_new.loc[np.arange(0,97)]
```

```
In [107]: vol_surface_new.to_csv('heston_call.csv')
```

```
In [111]: import plotly
          plotly.tools.set_credentials_file(username='lisayhe', api_key='FioDIUbTjZMAu76NCdei')
```

```
In [115]: data = [
              go.Surface(
                  z=vol_surface_new.as_matrix()
              )
          ]
          layout = go.Layout(
              title='Apple Heston Call Vol Surface',
              autosize=True,

          )
          fig = go.Figure(data=data, layout=layout)
          py.iplot(fig, filename='elevations-3d-surface', auto_open=True)
```

```
High five! You successfully sent some data to your account on plotly. View your plot in your bro
```

```
Out[115]: <plotly.tools.PlotlyDisplay object>
```