In [1]:
```
import numpy as np
import pandas as pd
import arch
import math
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.sandbox.regression.gmm import GMM
from sklearn.metrics import mean_squared_error
import scipy
from scipy import optimize
```

```
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/scipy/__init__.py:137: UserWarning: NumPy 1.16.5 or a
bove is required for this version of SciPy (detected version 1.16.2)
  UserWarning)
```

Question 3

In [2]:
```
monthly_SP500 = pd.read_csv('mon_ret.csv')
daily_SP500 = pd.read_csv('day_ret.csv')
```

In [3]:
```
monthly_SP500['Date'] = pd.to_datetime(monthly_SP500['caldt'])
monthly_SP500['Value'] = monthly_SP500['vwretd']

daily_SP500['Date'] = pd.to_datetime(daily_SP500['caldt'])
daily_SP500['Value'] = daily_SP500['vwretd']

monthly_SP500.drop(columns=['caldt', 'vwretd'], inplace=True)
daily_SP500.drop(columns=['caldt', 'vwretd'], inplace=True)
```

In [4]:
```
mu = 0.01
omega = 0.0073
alpha = 0.93
beta = 0.06
```

3.1

In [5]:
```
np.random.seed(100)

n=13200
error_list = np.random.normal(loc=0, scale=1, size=n)
epsilon_list = np.zeros_like(error_list)
sigma_sq_list = np.zeros_like(error_list)
daily_return_list = np.zeros_like(error_list)

sigma_sq_list[0] = omega
epsilon_list[0] = np.sqrt(sigma_sq_list[0])*error_list[0]
daily_return_list[0] = mu + epsilon_list[0]

for i in range(1, n):
    sigma_sq_list[i] = omega + alpha*(epsilon_list[i-1]**2) + beta*sig
ma_sq_list[i-1]
    epsilon_list[i] = np.sqrt(sigma_sq_list[i])*error_list[i]
    daily_return_list[i] = mu + epsilon_list[i]

monthly_return_list = np.sum(daily_return_list.reshape((600,22)), axis
=1)
```

3.2

In [6]:
```
# Fit GARCH(1,1) using simulated log returns
sim_am = arch.arch_model(monthly_return_list, vol='Garch', p=1, q=1)
sim_am_fit = sim_am.fit()
print(sim_am_fit.summary())
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 1444.759233035
1179
Iteration:      2,   Func. Count:     14,   Neg. LLF: 1042.430779492
922
Iteration:      3,   Func. Count:     20,   Neg. LLF: 1096.369375423
5067
Iteration:      4,   Func. Count:     26,   Neg. LLF: 1001.101190606
9872
Iteration:      5,   Func. Count:     31,   Neg. LLF: 1000.143752801
8495
Iteration:      6,   Func. Count:     36,   Neg. LLF: 999.7967681225
705
Iteration:      7,   Func. Count:     41,   Neg. LLF: 999.5136850873
464
Iteration:      8,   Func. Count:     46,   Neg. LLF: 999.1376842040
677
Iteration:      9,   Func. Count:     51,   Neg. LLF: 998.8130590612
179
Iteration:     10,   Func. Count:     56,   Neg. LLF: 998.7356414632
656
Iteration:     11,   Func. Count:     61,   Neg. LLF: 998.7164316737
626
```

```
Iteration:     12,   Func. Count:     66,   Neg. LLF: 998.7160086302
614
Iteration:     13,   Func. Count:     71,   Neg. LLF: 998.7159928521
223
Iteration:     14,   Func. Count:     75,   Neg. LLF: 998.7159935571
938
Optimization terminated successfully    (Exit mode 0)
            Current function value: 998.7159928521223
            Iterations: 14
            Function evaluations: 75
            Gradient evaluations: 14
                     Constant Mean - GARCH Model Results
====================================================================
==========
Dep. Variable:                      y   R-squared:
0.000
Mean Model:             Constant Mean   Adj. R-squared:
0.000
Vol Model:                      GARCH   Log-Likelihood:
-998.716
Distribution:                  Normal   AIC:
2005.43
Method:            Maximum Likelihood   BIC:
2023.02
                                        No. Observations:
600
Date:                Wed, Feb 17 2021   Df Residuals:
599
Time:                        21:02:06   Df Model:
1
                               Mean Model
====================================================================
====
                 coef    std err          t      P>|t|  95.0% Conf.
Int.
--------------------------------------------------------------------
----
mu             0.2336  4.905e-02      4.763  1.904e-06 [  0.138,  0.
330]
                             Volatility Model
====================================================================
======
                 coef    std err          t      P>|t|    95.0% Conf
. Int.
--------------------------------------------------------------------
------
omega          1.3555      0.306      4.430  9.418e-06   [  0.756,
1.955]
alpha[1]       0.2893      0.121      2.388  1.695e-02 [5.184e-02,
0.527]
beta[1]        0.0000  5.531e-02      0.000      1.000   [ -0.108,
0.108]
====================================================================
```

```
======

Covariance estimator: robust
```

$r_t = 0.2338 + \epsilon_t$

$\epsilon_t = \sigma_t \cdot e_t$

${\sigma_t}^2 = 1.3557 + 0.2892{\epsilon_{t-1}}^2$

$e_t \sim N(0, 1)$

Comment: estimated beta is 0, so the model is actually an ARCH(1). Parameters are different from the simulation.

In [7]:

```python
# Fit GARCH(1,1) using monthly SP500 returns
sp500_am = arch.arch_model(monthly_SP500['Value'], vol='Garch', p=1, q
=1)
sp500_am_fit = sp500_am.fit()
print(sp500_am_fit.summary())
```

```
Iteration:      1,   Func. Count:      6,   Neg. LLF: 104588362.0230
1483
Iteration:      2,   Func. Count:     17,   Neg. LLF: 23856.64163941
0704
Iteration:      3,   Func. Count:     29,   Neg. LLF: 2695.003805583
4057
Iteration:      4,   Func. Count:     38,   Neg. LLF: 2828.031179390
6376
Iteration:      5,   Func. Count:     48,   Neg. LLF: 1321.132909836
3567
Iteration:      6,   Func. Count:     57,   Neg. LLF: -1083.02798598
75764
Iteration:      7,   Func. Count:     65,   Neg. LLF: -1126.42271083
37475
Iteration:      8,   Func. Count:     72,   Neg. LLF: -1151.19671849
37255
Iteration:      9,   Func. Count:     78,   Neg. LLF: -1073.68777288
62505
Iteration:     10,   Func. Count:     85,   Neg. LLF: -1152.51618796
52273
Iteration:     11,   Func. Count:     90,   Neg. LLF: -1152.51625937
08292
Iteration:     12,   Func. Count:     95,   Neg. LLF: -1152.51626270
57448
Iteration:     13,   Func. Count:     99,   Neg. LLF: -1152.51626270
5782
Optimization terminated successfully    (Exit mode 0)
            Current function value: -1152.5162627057448
            Iterations: 13
```

```
            Function evaluations: 99
            Gradient evaluations: 13
                     Constant Mean - GARCH Model Results
====================================================================
==========
Dep. Variable:                    Value   R-squared:
0.000
Mean Model:               Constant Mean   Adj. R-squared:
0.000
Vol Model:                        GARCH   Log-Likelihood:
1152.52
Distribution:                    Normal   AIC:
-2297.03
Method:              Maximum Likelihood   BIC:
-2279.16
                                          No. Observations:
645
Date:                  Wed, Feb 17 2021   Df Residuals:
644
Time:                          21:02:06   Df Model:
1
                                Mean Model
====================================================================
========
                 coef    std err          t      P>|t|      95.0% Co
nf. Int.
--------------------------------------------------------------------
--------
mu         9.7667e-03  1.519e-03      6.431  1.266e-10 [6.790e-03,1.
274e-02]
                             Volatility Model
====================================================================
========
                 coef    std err          t      P>|t|      95.0% Co
nf. Int.
--------------------------------------------------------------------
--------
omega      5.9943e-05  2.570e-05      2.333  1.967e-02 [9.577e-06,1.
103e-04]
alpha[1]       0.1189  2.963e-02      4.013  6.008e-05   [6.082e-02,
0.177]
beta[1]        0.8559  2.699e-02     31.709 1.176e-220     [  0.803,
0.909]
====================================================================
========

Covariance estimator: robust
```

```
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/arch/univariate/base.py:293: DataScaleWarning: y is p
oorly scaled, which may affect convergence of the optimizer when
estimating the model parameters. The scale of y is 0.001802. Paramet
er
estimation work better when this value is between 1 and 1000. The re
commended
rescaling is 10 * y.

This warning can be disabled by either rescaling y before initializi
ng the
model or by setting rescale=False.

  data_scale_warning.format(orig_scale, rescale), DataScaleWarning
```

$r_t = 0.0098 + \epsilon_t$

$\epsilon_t = \sigma_t \cdot e_t$

${\sigma_t}^2 = 0.1189{\epsilon_{t-1}}^2 + 0.8559{\sigma_{t-1}}^2$

$e_t \sim N(0, 1)$

Comment: estimated omega is 0. Parameters are different from the simulation.

3.3

In [8]:
```
sim_data = daily_return_list.reshape((600,22))
sim_realized_vol = np.sum(sim_data**2, axis=1)
```

In [9]:
```
sim_realized_vol
```

Out[9]:
```
array([ 0.33696419,  0.64452492,  0.82536148,  0.71033854,  2.426050
2 ,
        1.8166057 ,  0.66263787,  0.88905317,  0.38591452,  0.414680
33,
        0.8088404 ,  0.75799977,  1.12432919, 14.00072638,  1.049721
89,
        3.83286656,  0.2987802 ,  0.30686165,  0.28109249,  0.871764
39,
        0.37570341,  0.41380882,  1.12748954, 49.21029024,  4.478481
42,
        0.72855334,  0.48641925,  0.62873936,  1.4209782 ,  0.347258
61,
        0.24751665,  0.5541451 ,  1.25858826,  0.54994952,  0.482921
66,
        0.13780333,  0.69854846,  0.66308217,  0.20625372,  0.367081
```

```
31,
         4.04873239,  2.02098843,  1.58698514,  0.48661718,  0.421390
3 ,
         0.49321065,  1.90936743,  0.26448925,  0.32776589,  0.165426
95,
         1.31317061,  0.59282267,  0.76015407,  0.12788704,  0.360733
15,
         4.21643603,  0.65649163,  0.23154853,  2.50721286,  5.103794
64,
         0.91492072,  0.88298146,  0.19077711,  1.25950752,  0.910768
04,
         0.31381253,  1.15084063,  1.10284443,  0.51082166,  0.298918
43,
         0.32921948,  0.67102037,  1.40454348,  0.3886656 ,  0.846618
87,
         0.32896543,  0.80176144,  2.66210305,  0.48733211,  0.561081
41,
         1.30421572,  1.19916831,  0.40452907,  3.21146608,  4.836800
55,
         0.49672245,  0.22824824,  0.4490011 ,  0.56651198,  0.480491
83,
         1.15817322,  0.74963638,  0.58763785,  0.85837335,  0.205977
58,
         0.3737807 ,  0.84347374,  6.93257528,  2.53768985,  0.649698
11,
         0.28937821,  0.51097466,  2.73450925,  0.47087165,  0.441518
2 ,
         1.64789388,  0.84025232,  0.41933006,  0.33655243,  0.619123
17,
         0.54612261,  0.34461587,  0.55483571,  0.30143262,  0.203953
85,
         1.55606058,  1.15349136,  0.39015067,  0.58718226,  4.549430
79,
         0.12312689,  7.28334721,  0.32092525,  0.21711317,  0.628844
56,
         1.24466352,  3.9227124 ,  1.40195569,  1.21344413,  0.492005
89,
         0.60110126,  1.17993582,  0.76818929,  0.4373716 ,  0.439950
74,
         0.84713499,  0.38818344,  0.80955805,  0.47390696,  0.206855
07,
         7.91555453,  1.66294235,  1.58692745,  0.83740199,  0.483664
73,
         0.91233699,  0.22677204,  0.78551504,  1.19429878,  1.966700
86,
        31.27796587, 32.33464093,  0.45076858,  0.42285661,  1.103865
24,
         2.96268532,  0.20610199,  0.86666353,  0.56598725,  3.995945
87,
         2.99510309,  2.7345149 ,  0.14244553,  0.11642949,  0.125360
28,
         4.69986849,  1.60704683, 37.89131077,  2.38561739,  1.970724
16,
```

```
         0.68674026,  0.55157113,  2.29025635,  0.60456771,  0.940754
06,
         0.40571413,  0.33692311,  0.49342834,  0.42691148,  0.809762
9 ,
         0.31170666,  0.73209397, 12.20088295,  0.53855277,  0.609387
24,
         6.39413285, 83.48932576,  0.45420219,  0.7966636 ,  0.401209
08,
         0.92111654,  0.20806234,  0.38002585,  2.2944348 ,  8.517090
01,
         0.8022801 ,  0.27586368,  0.24852296,  0.41111499,  0.193018
71,
         0.63930813,  0.26138176,  0.55113119,  0.50112207,  0.312020
21,
         0.55052501,  2.45462565,  0.67892615,  0.16449444,  0.153355
51,
         0.86861228,  1.84840847,  0.48150264,  0.50861049,  0.568594
25,
         0.30209535,  0.27707597,  0.75139345,  0.27811771,  0.244302
16,
         0.58444763,  0.52684587,  1.84008098,  3.70184941,  0.278526
72,
         1.71584252,  0.32960085,  1.00135494,  2.23042916,  0.529786
34,
         0.51701041,  0.98524917,  0.68461156,  1.1383359 ,  0.460909
74,
         0.17904861,  0.74361602,  1.82528642,  0.45448936,  0.377352
16,
         3.61815194,  1.70309993,  0.20247817,  0.21894955,  2.661712
28,
         2.10186356,  1.76065161,  0.53613647,  1.48022608,  3.596294
2 ,
         0.32452043,  0.73715647,  0.64395111,  1.31872677,  0.170581
1 ,
         0.46725243,  0.5545259 ,  0.51842541,  0.34129811,  0.342465
6 ,
         0.25797463,  1.39516401,  8.13978131,  0.47255096, 10.906975
82,
         0.33928081,  0.42295655,  0.61754856,  1.58300678,  0.194214
1 ,
         0.39277039,  0.34786215,  0.59723184,  1.60707958,  0.258600
59,
         0.25843402,  2.1889941 ,  0.36993821,  1.93620369,  0.382218
11,
         0.75082229,  0.21898043,  0.38044143,  0.62181805,  0.392126
01,
         0.33503921,  0.30918332,  0.36917796, 25.50942092,  0.448046
8 ,
         0.58117752,  0.37969836,  0.25961285,  0.42429512,  0.437742
23,
         0.30776115,  0.62467082,  1.07297991,  0.16317736,  2.429898
78,
         0.42106278,  8.98542149,  0.39118766,  0.37100186,  0.786875
```

```
51,
        0.88864009,  0.64218125,  1.65614208,  0.14943557,  0.393661
75,
        1.57078308,  1.23336201,  1.10245927,  1.26186107,  0.249389
43,
        1.34938273,  0.16058966,  0.12972248,  0.92311727,  0.380726
99,
        0.19496585,  0.28991333,  0.73058342,  0.24685159, 25.184797
17,
       38.9671448 ,  1.15130338,  0.67534513,  0.44821557,  0.144289
18,
        0.7580035 ,  4.81079908,  1.74950362,  0.72780524,  0.502668
29,
        0.66014263,  1.9535939 ,  0.34382941,  0.42954599,  0.387475
16,
        0.52537459,  0.5852028 ,  0.35962534,  0.48908979,  0.295073
82,
        1.95108789,  0.22639076,  0.16902375,  0.2812895 ,  3.176887
07,
        0.26727972,  0.77993498,  0.09795767,  4.30139776,  1.416512
07,
        2.79097665,  1.51650815,  0.99989854,  0.39783475,  0.769733
91,
        0.34785564,  1.2244073 ,  0.18898201,  0.36144473,  0.633180
64,
        1.27705111,  0.23095138,  0.30866193,  0.33850924,  0.368216
87,
        0.29538794,  1.9537727 ,  1.98363556,  0.28003554,  0.211504
59,
        1.01333441,  0.83480351,  2.71237767,  1.1104815 ,  0.337243
68,
        1.69972619,  0.36364665,  0.31032694,  0.30427356,  0.524995
66,
        1.31843585,  0.41298977,  0.15820304,  0.19001538,  0.370310
5 ,
        0.97335351,  0.89391159, 21.91735745,  0.38820609,  0.266426
81,
        1.13043023,  9.84619174,  1.75835966,  0.36713578,  0.324984
39,
        0.24099758,  0.23076508,  0.13204037,  1.10677178,  1.061036
93,
        0.17351825,  0.84235272,  1.80840834,  0.30138043,  0.246625
44,
        1.38589331,  4.65888671,  0.20043638,  1.93901627,  0.671532
62,
        0.39658246,  0.14685199,  0.2762488 ,  1.52854208,  0.269320
53,
        1.73872907,  0.19395754,  0.28943823,  0.68736405,  2.961942
59,
        1.33976286,  0.94706816,  0.15807716,  0.48445393,  2.265393
76,
        0.26454561,  1.15962685,  0.75795853,  3.19221266,  1.109209
49,
```

```
        0.88384647,  0.71357296,  0.88774289,  0.48819766,  0.084339
83,
        0.31143324,  1.78744862,  0.34648049,  6.76633285,  1.338870
62,
        0.57529745,  0.7078694 ,  0.77205743,  0.16697904,  0.424023
03,
        0.85055894,  0.28310807,  0.36249058,  0.21043769,  0.317004
32,
        0.20124993,  0.64158202,  0.26036878,  0.39546377,  0.977340
86,
        0.23275834,  1.50826592,  0.5213839 ,  0.75013097,  1.859699
76,
        1.16203241,  2.15433957,  0.95189311,  0.28618246,  0.286881
47,
        0.84112139,  0.18278458,  0.85950372,  0.74719088,  0.425066
3 ,
        0.79014379,  0.47372375,  1.76917202,  0.35353999,  0.234730
65,
        0.8027955 ,  0.77087565,  0.49946351,  0.28202528,  1.696547
96,
        2.31426516,  0.45801977,  0.25953781,  0.62622312,  1.194046
04,
        0.84101158,  0.40065699, 33.63844074,  0.30274793,  0.342215
54,
        1.192478  ,  0.38126526,  0.56167941,  0.42760459,  0.350369
06,
        0.53295789,  1.09176302,  2.48761263,  0.16466812,  0.783667
01,
        0.39949589,  0.50960918,  0.19711324,  0.61612078,  0.938065
12,
        0.55116566,  0.16797561,  1.02423562,  0.26524472,  0.245011
1 ,
        1.24032239,  0.55009748,  0.36521081,  1.48935886,  4.160036
44,
        2.63087629,  4.8055313 ,  2.01367005,  1.67851932,  0.273769
27,
        5.04351317, 11.16857691,  4.55311152,  1.05675737,  0.204001
49,
        0.50692416,  0.24257198,  1.43829461,  1.46428708,  0.373002
75,
        0.91084194,  1.87726965,  1.01599801, 13.5757576 ,  0.736022
34,
        0.34206693,  0.27925639,  0.38567803,  2.42857406,  1.853957
77,
        0.24333946,  0.43347398,  0.26320492,  0.37386627,  0.221361
58,
        0.92875604,  0.30854854,  0.8580061 ,  0.50274152,  2.641896
4 ,
        0.81855108,  3.09194707,  0.47655377,  1.47259735,  0.642002
79,
        0.51761058,  0.32291548,  0.27699043,  0.16256915,  0.290254
09,
        0.22143603,  3.5549595 ,  0.24161369,  1.00016494,  0.390348
```

```
67,
          0.28975554,  0.27822805,  0.98487849,  0.58000176,  0.689699
67,
          0.72653994,  0.16414759,  0.26695834,  0.20663004,  0.875184
23,
          0.17594178,  0.13167604,  0.37478277,  3.23429263,  0.683231
04,
          3.27890486,  0.76142445,  1.20701142,  2.85184284,  0.332303
37,
          0.19074225,  0.23849809,  0.387323  ,  0.78559119,  1.321076
16,
          0.24564721,  0.81308745,  0.60053518,  0.31150959,  0.454993
79])
```

In [10]:
```python
sp500_data = np.array(daily_SP500['Value'][1:]).reshape((615,22))
sp500_realized_vol = np.sum(sp500_data**2, axis=1)
```

In [11]:
```python
sp500_realized_vol
```

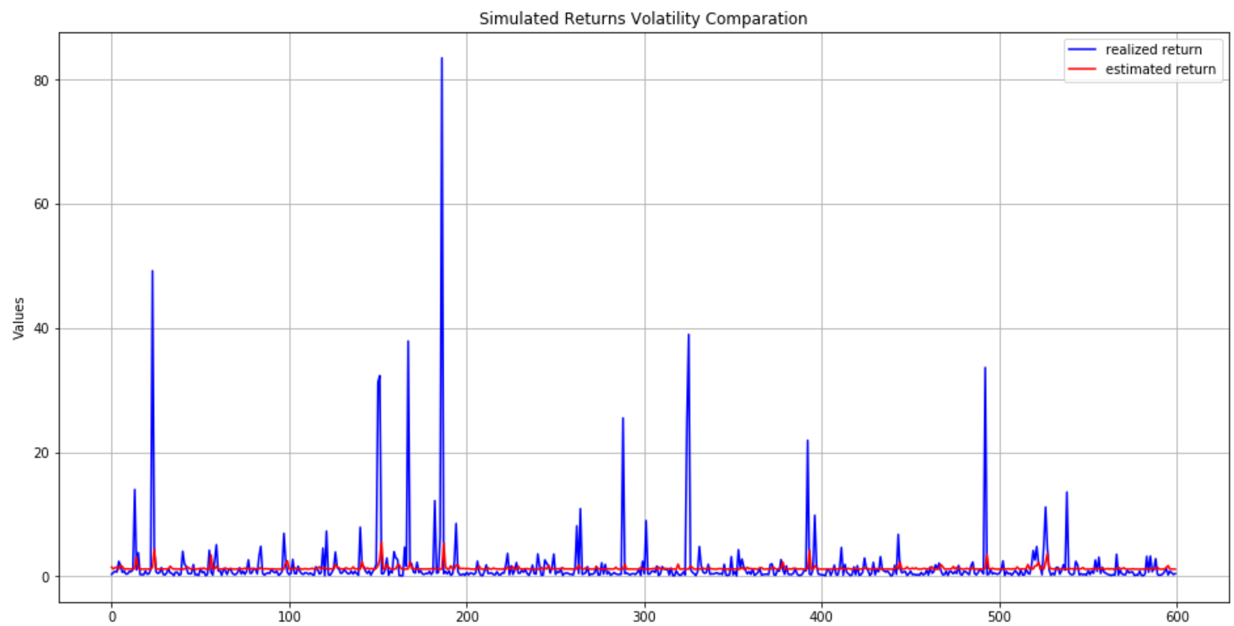
Out[11]:
```
array([0.00047069, 0.00033978, 0.0004403 , 0.00044539, 0.00348084,
       0.00023786, 0.00015733, 0.00010861, 0.00013726, 0.00033965,
       0.00043314, 0.00022755, 0.00035599, 0.00027348, 0.00016534,
       0.00014795, 0.00037695, 0.00021964, 0.00038581, 0.00013608,
       0.00016963, 0.00044791, 0.00181669, 0.00045828, 0.0001653 ,
       0.00027399, 0.0001775 , 0.00029173, 0.00020654, 0.00025188,
       0.00087954, 0.00049762, 0.00163429, 0.0007086 , 0.0007832 ,
       0.00158807, 0.00321782, 0.00272368, 0.00095788, 0.00063015,
       0.00075128, 0.00056099, 0.00048604, 0.00076862, 0.00163825,
       0.00030428, 0.00026229, 0.00025167, 0.00045376, 0.0005842 ,
       0.0007613 , 0.00034125, 0.00083309, 0.00141393, 0.00183557,
       0.00029327, 0.00064197, 0.00084226, 0.00043982, 0.00028966,
       0.00034163, 0.00071548, 0.00025293, 0.00085606, 0.00054862,
       0.00046159, 0.00074363, 0.00154407, 0.00168345, 0.00097486,
       0.00079757, 0.00055606, 0.0011314 , 0.00093524, 0.0013145 ,
       0.00100532, 0.00216166, 0.00782013, 0.00256336, 0.0020452 ,
       0.00208443, 0.00108206, 0.00078388, 0.00109833, 0.00056459,
       0.0003984 , 0.00074537, 0.00024801, 0.0005358 , 0.0006474 ,
       0.00065139, 0.00245268, 0.00081751, 0.00056918, 0.00136724,
       0.00169913, 0.00047922, 0.00041275, 0.00052292, 0.00050121,
       0.00071318, 0.00036915, 0.00081688, 0.00041027, 0.00058715,
       0.00096109, 0.00038633, 0.00066955, 0.00106211, 0.0011155 ,
       0.00172324, 0.0025175 , 0.00312932, 0.00207   , 0.00115353,
       0.00107382, 0.00109946, 0.00406109, 0.00602654, 0.00339303,
       0.00195194, 0.00185119, 0.00210253, 0.00254938, 0.00280648,
       0.00478838, 0.00630998, 0.01003087, 0.00615355, 0.00489158,
       0.00317761, 0.00349399, 0.00253067, 0.00279709, 0.00194677,
       0.00182009, 0.00111546, 0.00181587, 0.0026875 , 0.00236181,
       0.00107113, 0.00133831, 0.00203191, 0.00126882, 0.00122636,
       0.00111625, 0.00094269, 0.00094794, 0.00048924, 0.00082937,
       0.00100988, 0.00156906, 0.0006707 , 0.00084219, 0.00043706,
       0.00044845, 0.00089126, 0.00081176, 0.00077401, 0.00043409,
       0.00072368, 0.00051414, 0.00080361, 0.00148132, 0.00065279,
       0.00098589, 0.00059417, 0.00075187, 0.00193814, 0.00123694,
```

```
0.00077716, 0.00146358, 0.00068295, 0.00098886, 0.0043691 ,
0.00135755, 0.00179713, 0.00079748, 0.00095551, 0.00088045,
0.0008732 , 0.00060323, 0.0007612 , 0.00063386, 0.00121126,
0.00223432, 0.00208551, 0.00094818, 0.0016661 , 0.00217031,
0.0048112 , 0.00262924, 0.00185403, 0.00144994, 0.00140998,
0.00219126, 0.00284124, 0.00329583, 0.00272459, 0.00170743,
0.00188539, 0.00173788, 0.00129125, 0.00107158, 0.00106884,
0.00103104, 0.0017856 , 0.00287051, 0.00157484, 0.00214313,
0.00085327, 0.00310061, 0.00188754, 0.00150656, 0.00111133,
0.00092173, 0.00152869, 0.00152715, 0.00671831, 0.00431006,
0.00616382, 0.00366987, 0.0027507 , 0.00271568, 0.00200539,
0.00120915, 0.00148151, 0.00109902, 0.00227022, 0.00160074,
0.00110041, 0.00117121, 0.00084733, 0.00050936, 0.00078489,
0.00218057, 0.001374  , 0.00106486, 0.00105814, 0.00154447,
0.00306207, 0.00086012, 0.00095285, 0.00139077, 0.00077099,
0.00182117, 0.00140585, 0.00083524, 0.00039528, 0.00095273,
0.00074933, 0.00051079, 0.00082179, 0.00084543, 0.00091816,
0.00104131, 0.00127407, 0.00193664, 0.00102211, 0.00241326,
0.00156064, 0.0018997 , 0.00220104, 0.00155929, 0.00422112,
0.00108454, 0.00090504, 0.00221217, 0.00197044, 0.00229931,
0.00118729, 0.00486793, 0.00265159, 0.00094835, 0.00088588,
0.00179083, 0.00255731, 0.06542112, 0.00684711, 0.00815975,
0.00787948, 0.00184978, 0.00263391, 0.00254021, 0.00325051,
0.00191291, 0.00177295, 0.00164165, 0.00106226, 0.0015533 ,
0.00104363, 0.00072716, 0.00100945, 0.00172904, 0.00092084,
0.00098429, 0.00108902, 0.00133035, 0.00145673, 0.00056052,
0.00553646, 0.00101606, 0.00109954, 0.0025167 , 0.00132755,
0.00094748, 0.0009263 , 0.00174065, 0.0013249 , 0.00252287,
0.00451544, 0.00438727, 0.00348396, 0.00127088, 0.00265316,
0.00275042, 0.00139141, 0.0021367 , 0.00164756, 0.00123896,
0.00113047, 0.00200408, 0.00051001, 0.00109807, 0.00229914,
0.00181713, 0.00096489, 0.00093225, 0.0009277 , 0.00120181,
0.00074018, 0.00091872, 0.00052677, 0.00085405, 0.0010087 ,
0.00053824, 0.00042048, 0.00045387, 0.00106146, 0.00107779,
0.00100856, 0.00090621, 0.00058838, 0.00057224, 0.00030243,
0.00046521, 0.00030403, 0.0005734 , 0.00027364, 0.00043793,
0.00105668, 0.0016065 , 0.00094806, 0.0004108 , 0.00088733,
0.00033166, 0.00074467, 0.00100557, 0.00088483, 0.00107448,
0.00032314, 0.00038999, 0.00060157, 0.00029269, 0.00067776,
0.00102196, 0.00078229, 0.00020752, 0.00040849, 0.00048302,
0.00065143, 0.00077965, 0.00143535, 0.00223421, 0.00121396,
0.0008495 , 0.0007968 , 0.00170276, 0.00163251, 0.00102066,
0.00043129, 0.00068531, 0.00121617, 0.00150526, 0.00171313,
0.00145221, 0.00303356, 0.0028278 , 0.001983  , 0.00203249,
0.00240358, 0.00261547, 0.00886935, 0.00306707, 0.00189534,
0.00290488, 0.00117914, 0.00076996, 0.0017275 , 0.00167955,
0.00148062, 0.00413438, 0.01400549, 0.0072771 , 0.00137163,
0.0038493 , 0.00331487, 0.0041683 , 0.00302953, 0.00285383,
0.00314177, 0.00241292, 0.00160397, 0.00288656, 0.0028357 ,
0.00466504, 0.00173749, 0.00323941, 0.00370755, 0.0043551 ,
0.00533965, 0.0097481 , 0.0042404 , 0.00196974, 0.00194126,
0.00095454, 0.00513779, 0.00361001, 0.00564664, 0.00579812,
0.00247813, 0.00726132, 0.00785043, 0.00238302, 0.00161387,
```

```
0.00284062, 0.00215501, 0.00849518, 0.00280771, 0.00254203,
0.00117176, 0.0032649 , 0.00255478, 0.0021658 , 0.00432977,
0.00370694, 0.01255585, 0.01222048, 0.00754735, 0.01171173,
0.00460669, 0.00372494, 0.0042513 , 0.00282697, 0.006474  ,
0.00286429, 0.00228661, 0.00229094, 0.00172174, 0.00094815,
0.00197206, 0.00102107, 0.00108283, 0.00082849, 0.00105173,
0.00153581, 0.00138974, 0.00122265, 0.00077812, 0.00077926,
0.00143591, 0.0009343 , 0.00100563, 0.00108408, 0.00070445,
0.00080247, 0.00090957, 0.00079767, 0.00204387, 0.00060232,
0.00070922, 0.00075913, 0.0006349 , 0.00088143, 0.00154526,
0.00047742, 0.00099654, 0.00066886, 0.00059747, 0.0006731 ,
0.00134624, 0.00207594, 0.00151589, 0.00044564, 0.00064465,
0.00035042, 0.00065755, 0.00030448, 0.00049497, 0.0023604 ,
0.00091918, 0.00072166, 0.00121988, 0.00142934, 0.00506232,
0.00330722, 0.0013668 , 0.00434382, 0.00375652, 0.00466353,
0.00350783, 0.00757251, 0.00188879, 0.00145984, 0.00365895,
0.00498399, 0.00382492, 0.0287566 , 0.05831261, 0.04236686,
0.00831339, 0.01266011, 0.01756344, 0.01410414, 0.00589161,
0.00445426, 0.00418116, 0.00222808, 0.00158366, 0.00252557,
0.00337186, 0.00113791, 0.00225297, 0.00227652, 0.00064801,
0.00347641, 0.00869296, 0.00429861, 0.00277901, 0.00340626,
0.00154141, 0.00107197, 0.0018572 , 0.00044224, 0.0009931 ,
0.00271242, 0.00069261, 0.00092937, 0.00188669, 0.00194229,
0.0185383 , 0.00688217, 0.00789766, 0.00642992, 0.0046424 ,
0.00073352, 0.0008532 , 0.00103278, 0.00183469, 0.00234183,
0.00241031, 0.00180365, 0.00075054, 0.00078125, 0.00181726,
0.00112502, 0.00142455, 0.00077801, 0.00101689, 0.0016216 ,
0.0008668 , 0.00244957, 0.00050823, 0.00094032, 0.00069504,
0.00138284, 0.00066191, 0.00084389, 0.0021223 , 0.00066108,
0.00155613, 0.00062264, 0.00046582, 0.00049972, 0.00104367,
0.00029817, 0.00248615, 0.00104932, 0.00191019, 0.00174657,
0.00141772, 0.00166137, 0.00064856, 0.00088734, 0.00109391,
0.0011124 , 0.00701596, 0.00341131, 0.00121328, 0.0018924 ,
0.00294156, 0.00425564, 0.0025749 , 0.00086854, 0.00087787,
0.00053732, 0.00304719, 0.00025645, 0.00137695, 0.00067949,
0.00095391, 0.00048906, 0.00039418, 0.00045683, 0.0003794 ])
```

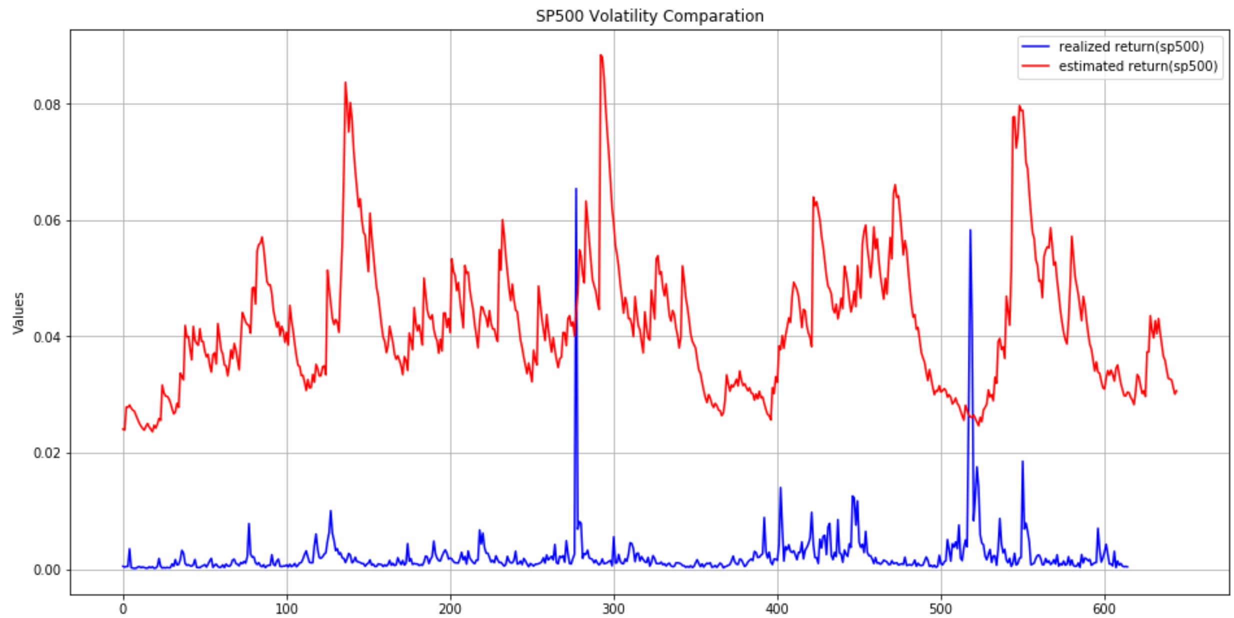
3.4

In [12]:
```
plt.figure(figsize=(16,8))
plt.grid(True)
plt.ylabel('Values')
plt.title('Simulated Returns Volatility Comparation')
plt.plot(sim_realized_vol, color='blue', label='realized return')
plt.plot(sim_am_fit.conditional_volatility, color='red', label='estima
ted return')
plt.legend()
plt.plot()
plt.show()
```



Comment: for simulated returns, realized volatility has a larger variance than estimated volatility.

In [13]:
```
plt.figure(figsize=(16,8))
plt.grid(True)
plt.ylabel('Values')
plt.title('SP500 Volatility Comparation')
plt.plot(sp500_realized_vol, color='blue', label='realized return(sp50
0)')
plt.plot(sp500_am_fit.conditional_volatility, color='red', label='esti
mated return(sp500)')
plt.legend()
plt.plot()
plt.show()
```



Comment: for SP500 returns, estimated volatility has a larger variance than realized volatility.

3.5

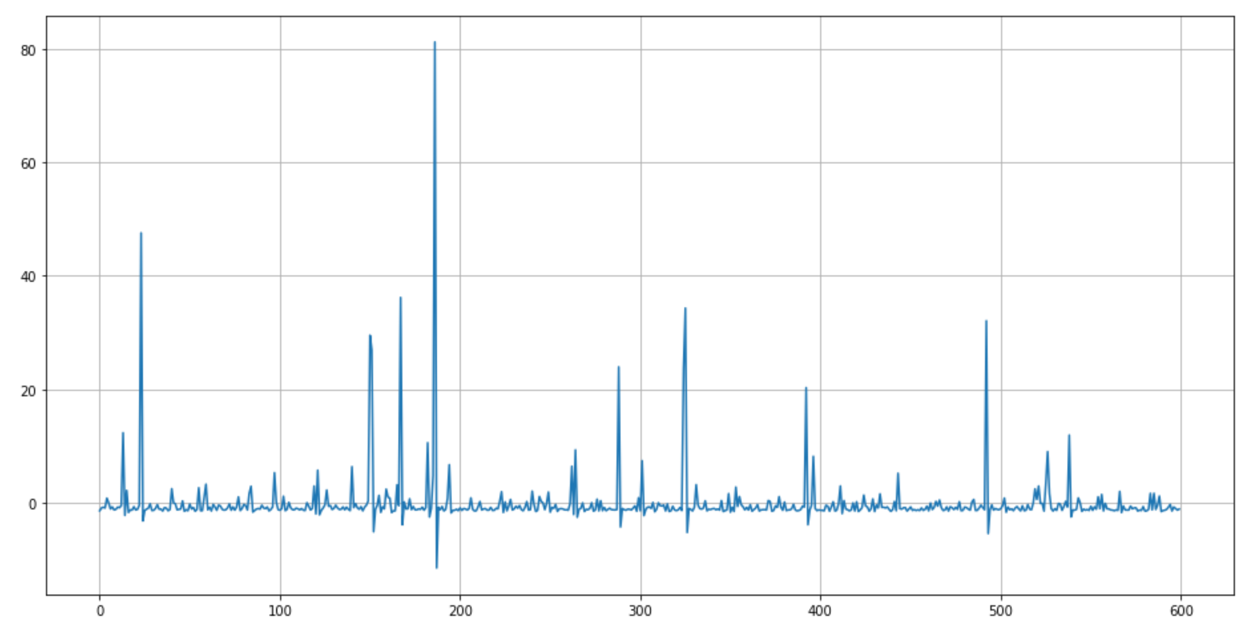
In [14]:
```
sim_ar = sm.tsa.ARIMA(sim_realized_vol, order=(1,0,0))
sim_ar_fit = sim_ar.fit(disp=0)
print(sim_ar_fit.summary())
```

```
                              ARMA Model Results
=====================================================================
==========
Dep. Variable:                      y   No. Observations:
600
Model:                     ARMA(1, 0)   Log Likelihood
-1863.573
Method:                       css-mle   S.D. of innovations
5.403
Date:                Wed, 17 Feb 2021   AIC
3733.146
Time:                        21:02:07   BIC
3746.337
Sample:                             0   HQIC
3738.281

=====================================================================
==========
                 coef    std err          z      P>|z|      [0.025
0.975]
---------------------------------------------------------------------
----------
const          1.7243      0.252      6.843      0.000       1.230
2.218
ar.L1.y        0.1248      0.040      3.083      0.002       0.045
0.204
                                    Roots
=====================================================================
=========
                  Real          Imaginary           Modulus
Frequency
---------------------------------------------------------------------
---------
AR.1           8.0143           +0.0000j            8.0143
0.0000
---------------------------------------------------------------------
---------
```

```
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARI
MA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note
the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 releas
e.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framew
ork and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they
are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA'
,
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA
',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
```

In [15]:
```
plt.figure(figsize=(16,8))
plt.grid(True)
plt.plot(sim_ar_fit.resid)
```

Out[15]: [<matplotlib.lines.Line2D at 0x11e60ae10>]



$$\sigma_t^2 = 1.7243 + 0.1248\sigma_{t-1}^2$$

In [16]:

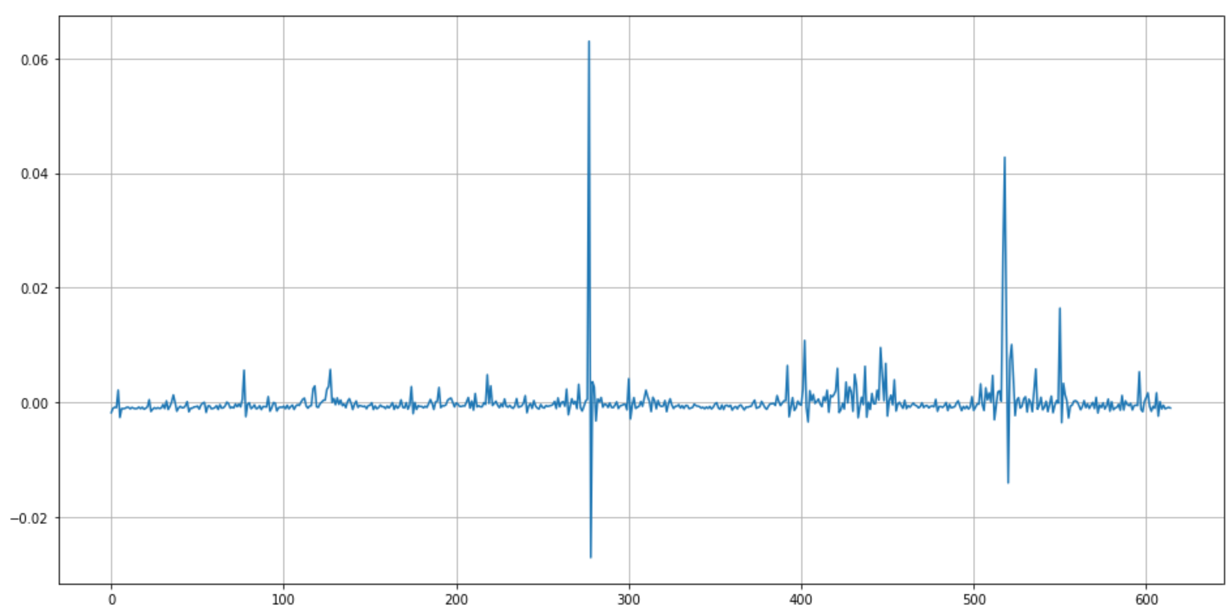
```
sp500_ar = sm.tsa.ARIMA(sp500_realized_vol, order=(1,0,0))
sp500_ar_fit = sp500_ar.fit(disp=0)
print(sp500_ar_fit.summary())
```

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:
615
Model:                     ARMA(1, 0)   Log Likelihood
2540.938
Method:                       css-mle   S.D. of innovations
0.004
Date:                Wed, 17 Feb 2021   AIC
-5075.875
Time:                        21:02:07   BIC
-5062.610
Sample:                             0   HQIC
-5070.717

==============================================================================
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
const          0.0022      0.000      7.171      0.000       0.002
0.003
ar.L1.y        0.5014      0.035     14.393      0.000       0.433
0.570
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus
Frequency
-----------------------------------------------------------------------------
AR.1            1.9944           +0.0000j            1.9944
0.0000
-----------------------------------------------------------------------------
```

```
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARI
MA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note
the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 releas
e.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framew
ork and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they
are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA'
,
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA
',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
```

In [17]:
```
plt.figure(figsize=(16,8))
plt.grid(True)
plt.plot(sp500_ar_fit.resid)
```

Out[17]: [<matplotlib.lines.Line2D at 0x130badeb8>]

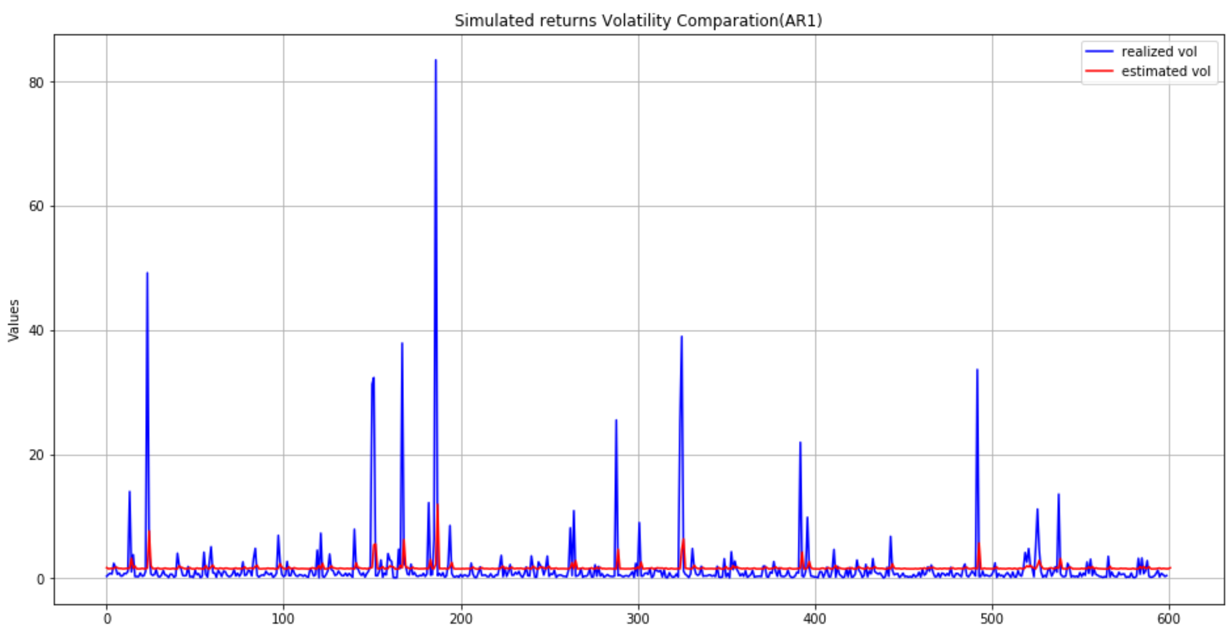


$$\sigma_t^2 = 0.0022 + 0.5014\sigma_{t-1}^2$$

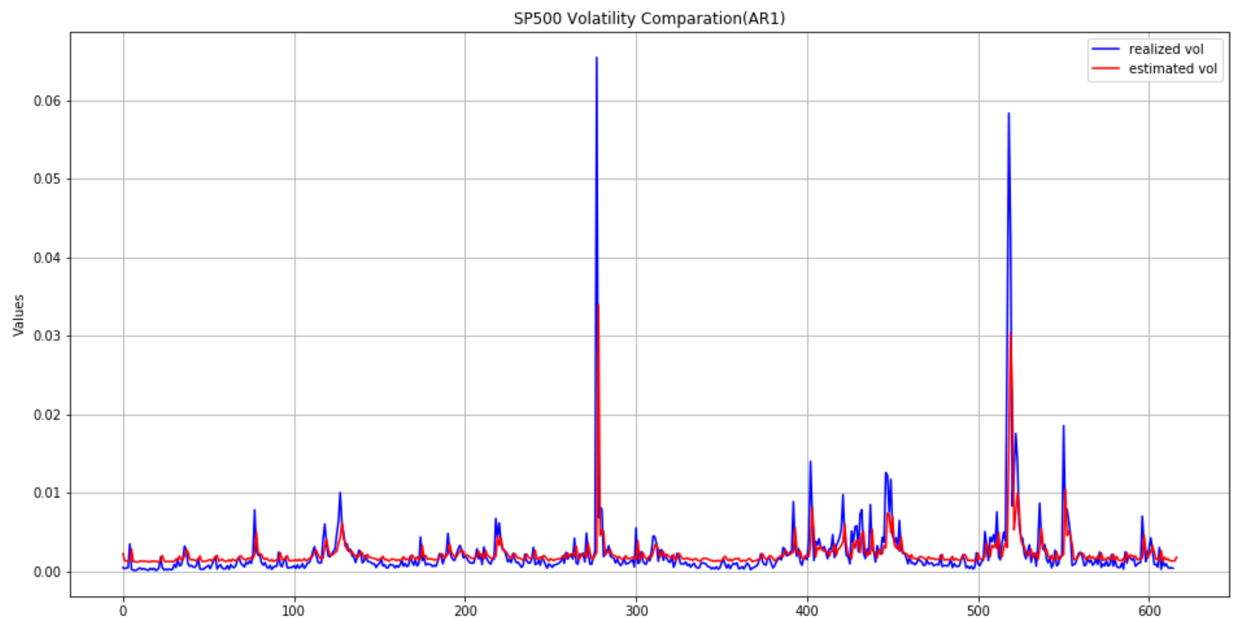
Comment: for both dataset, the residuals from an AR(1) model showed GARCH behavior(volatility clustering).

3.6

In [18]:
```
plt.figure(figsize=(16,8))
plt.grid(True)
plt.ylabel('Values')
plt.title('Simulated returns Volatility Comparation(AR1)')
plt.plot(sim_realized_vol, color='blue', label='realized vol')
plt.plot(sim_ar_fit.predict(start=0, end=601), color='red', label='est
imated vol')
plt.legend()
plt.plot()
plt.show()
```



```
In [19]: plt.figure(figsize=(16,8))
plt.grid(True)
plt.ylabel('Values')
plt.title('SP500 Volatility Comparation(AR1)')
plt.plot(sp500_realized_vol, color='blue', label='realized vol')
plt.plot(sp500_ar_fit.predict(start=0, end=616), color='red', label='e
stimated vol')
plt.legend()
plt.plot()
plt.show()
```



3.7

```
In [20]: # MSE of the AR(1) forecasts (simulated returns)
mse = mean_squared_error(sim_ar_fit.predict(), sim_realized_vol)
print('MSE: '+str(mse))
```

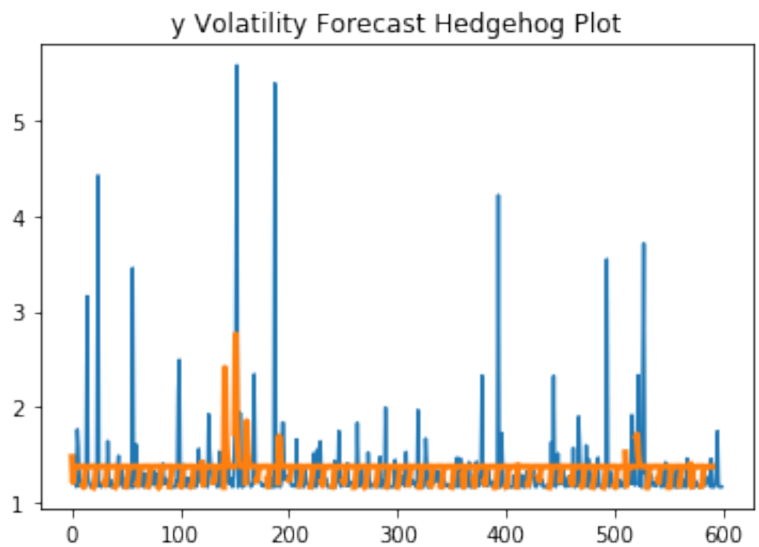
```
MSE: 29.19533587236354
```

```
In [21]: # MSE of the AR(1) forecasts (SP500 returns)
mse = mean_squared_error(sp500_ar_fit.predict(), sp500_realized_vol)
print('MSE: '+str(mse))
```

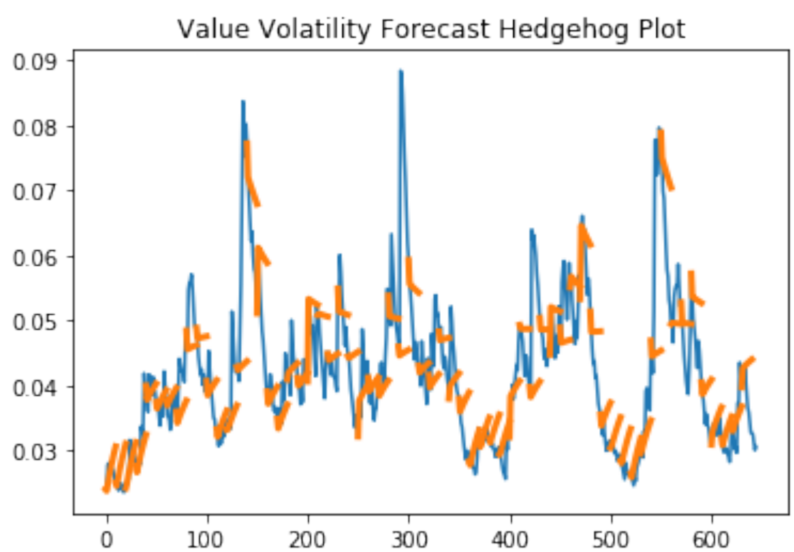
```
MSE: 1.5090067997866108e-05
```

3.8

In [22]:
```
plot = sim_am_fit.hedgehog_plot()
```



In [23]:
```
plot = sp500_am_fit.hedgehog_plot()
```



3.9

In [24]:
```
# MSE of the AR(1) forecasts (simulated returns)
mse = mean_squared_error(sim_am_fit.conditional_volatility, sim_realiz
ed_vol)
print('MSE: '+str(mse))
```

```
MSE: 29.741459534195045
```

In [25]:
```
# MSE of the AR(1) forecasts (SP500 returns)
mse = mean_squared_error(sp500_am_fit.conditional_volatility[:len(sp50
0_realized_vol)], sp500_realized_vol)
print('MSE: '+str(mse))
```

```
MSE: 0.0017705720218543573
```

Question 4

4.1 GMM

In [26]:
```
kappa = 0.234
theta = 0.081
sigma_sqr = 0.0073
```

In [27]:
```
np.sqrt(0.0073)
```

Out[27]:
```
0.08544003745317531
```

In [28]:
```
np.random.seed(100)
```

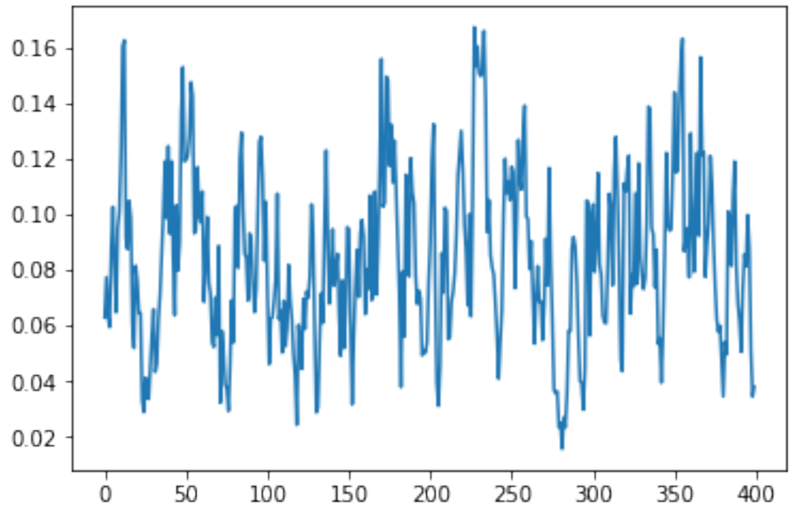
In [29]:
```
interest_rate = pd.read_csv('interest_rates.csv')
```

In [30]:
```
rates = interest_rate['NFCP_M1']
```

In [31]:
```
n = 100000
steps = 250
delta = 1/steps

sim_rates = np.zeros(int(n/steps))
ir = theta

cur = 0

for i in range(n):
    epsilon = np.random.normal(loc=0, scale=np.sqrt(sigma_sqr*ir*delta
))
    ir = ir + kappa*(theta-ir)*delta + epsilon
    if (i+1) % 250 == 0:
        sim_rates[cur] = ir
        cur+=1
```

In [32]:
```
plt.plot(sim_rates)
```

Out[32]: [<matplotlib.lines.Line2D at 0x132409d30>]



Simulated Data

In [33]:
```
rt_new = sim_rates[1:]
rt_old = sim_rates[:-1]

def moments(endog, exog, params):
    kappa, theta, sig = params
    delta = 1
    h1 = (endog - exog - kappa * (theta - exog) * delta)
    h2 = h1 * exog
    h3 = ((endog - exog - kappa * (theta - exog) * delta)**2) - (sig**
2)*exog*delta
    h4 = h3 * exog

    g = np.column_stack((h1, h2, h3, h4))

    return g
```

In [34]:
```
class gmm(GMM):

    def momcond(self, params):

        endog = self.endog
        exog = self.exog.squeeze()

        return moments(endog, exog, params)
```

In [35]:
```
beta0 = np.array([kappa, theta, np.sqrt(sigma_sqr)])
inst = np.column_stack((np.ones(len(rt_new)), rt_new))

mm = gmm(endog=rt_new, exog=rt_old, maxiter=1000, instrument=inst, k_m
oms=4, k_params=3, missing='none')
```

In [36]:
```
res = mm.fit(beta0)
print(res.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.000000
         Iterations: 4
         Function evaluations: 5
         Gradient evaluations: 5
Optimization terminated successfully.
         Current function value: 0.002534
         Iterations: 9
         Function evaluations: 16
         Gradient evaluations: 16
Optimization terminated successfully.
         Current function value: 0.004808
         Iterations: 5
         Function evaluations: 10
         Gradient evaluations: 10
Optimization terminated successfully.
         Current function value: 0.003391
         Iterations: 6
         Function evaluations: 10
         Gradient evaluations: 10
Optimization terminated successfully.
         Current function value: 0.005233
         Iterations: 4
         Function evaluations: 8
         Gradient evaluations: 8
Optimization terminated successfully.
         Current function value: 0.002842
         Iterations: 5
         Function evaluations: 9
         Gradient evaluations: 9
Optimization terminated successfully.
         Current function value: 0.006440
         Iterations: 4
         Function evaluations: 8
         Gradient evaluations: 8
Optimization terminated successfully.
         Current function value: 0.002042
         Iterations: 5
         Function evaluations: 9
         Gradient evaluations: 9
Optimization terminated successfully.
         Current function value: 0.008882
         Iterations: 5
```

```
         Function evaluations: 9
         Gradient evaluations: 9
Optimization terminated successfully.
         Current function value: 0.001182
         Iterations: 5
         Function evaluations: 9
         Gradient evaluations: 9
                                 gmm Results
====================================================================
==========
Dep. Variable:                          y   Hansen J:
0.4715
Model:                                gmm   Prob (Hansen J):
0.492
Method:                               GMM
Date:                    Wed, 17 Feb 2021
Time:                            21:02:09
No. Observations:                     399
====================================================================
==========
                 coef    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------
----------
p 0            0.2844      0.036      7.857      0.000       0.213
0.355
p 1            0.0841      0.004     22.459      0.000       0.077
0.091
p 2            0.0745      0.002     30.304      0.000       0.070
0.079
====================================================================
==========
```

In [37]:
```
delta = 1
k,t,s = res.params
print(k,t,s)
```

```
0.28437355842459117 0.0841028972461179 0.0744774728745909
```

The estimated three parameters κ, θ and σ by GMM is represented above respectively.

Compared with result from real data:

```
In [38]: delta = 1
         rt_real_new = rates[1:]
         rt_real_old = rates[:-1]

         def gmm_initial(gamma):

             kappa, theta, sigma_sqr = gamma

             h1 = rt_real_new - rt_real_old - kappa*(theta - rt_real_old)*delta
             h2 = h1*rt_real_old
             h3 = h1**2 - sigma_sqr*rt_real_old*delta
             h4 = h3*rt_real_old

             g = np.column_stack((h1, h2, h3, h4)).mean(axis=0)

             obj = np.sum(g**2)

             return obj

         gmm_optimal_real = optimize.fmin(gmm_initial, [0.2, 0.1, 0.01], maxite
         r=1000)
         print(gmm_optimal_real)
```

```
Warning: Maximum number of iterations has been exceeded.
[0.2  0.1  0.01]
```

Now, back to simulated data, calculate covariance matrix W and take W_prime = inverse(W) as the new weight matrix:

```
In [39]: endog = rt_new
         exog = rt_old

         error1 = (endog - exog - k * (t - exog) * delta)
         error2 = error1 * exog
         error3 = error1**2 - s**2*exog*delta
         error4 = error3 * exog

         g = np.column_stack((error1, error2, error3, error4))/(len(endog))
```

```
In [40]: W = mm.calc_weightmatrix(moms = g, weights_method='cov', wargs=(), par
         ams=None)
         W_prime = np.linalg.inv(W)
```

```
In [41]: print(W)
```

```
[[2.93353873e-09 2.64853313e-10 1.77491761e-11 1.12594577e-12]
 [2.64853313e-10 2.67267470e-11 1.12870251e-12 5.55688697e-14]
 [1.77491761e-11 1.12870251e-12 2.42776394e-12 2.24676598e-13]
 [1.12594577e-12 5.55688697e-14 2.24676598e-13 2.39031606e-14]]
```

In [42]:
```
print(W_prime) # inverse
```

```
[[ 3.44411477e+09 -3.36482744e+10 -1.35359195e+10  4.32208319e+10]
 [-3.36482744e+10  3.68054488e+11  5.67768223e+10  1.95678594e+11]
 [-1.35359195e+10  5.67768223e+10  3.36333367e+12 -3.11078812e+13]
 [ 4.32208319e+10  1.95678594e+11 -3.11078812e+13  3.31741694e+14]]
```

In [43]:
```
res2 = mm.fit(beta0, inv_weights = W_prime)
```

```
Optimization terminated successfully.
         Current function value: 0.000000
         Iterations: 0
         Function evaluations: 1
         Gradient evaluations: 1
Optimization terminated successfully.
         Current function value: 0.000314
         Iterations: 9
         Function evaluations: 17
         Gradient evaluations: 17
Optimization terminated successfully.
         Current function value: 0.015799
         Iterations: 7
         Function evaluations: 12
         Gradient evaluations: 12
Optimization terminated successfully.
         Current function value: 0.000406
         Iterations: 5
         Function evaluations: 9
         Gradient evaluations: 9
Optimization terminated successfully.
         Current function value: 0.015974
         Iterations: 5
         Function evaluations: 10
         Gradient evaluations: 10
Optimization terminated successfully.
         Current function value: 0.000404
         Iterations: 7
         Function evaluations: 10
         Gradient evaluations: 10
Optimization terminated successfully.
         Current function value: 0.015981
         Iterations: 5
         Function evaluations: 10
         Gradient evaluations: 10
Optimization terminated successfully.
         Current function value: 0.000403
         Iterations: 7
         Function evaluations: 10
         Gradient evaluations: 10
Optimization terminated successfully.
         Current function value: 0.015983
         Iterations: 5
         Function evaluations: 10
         Gradient evaluations: 10
Optimization terminated successfully.
         Current function value: 0.000403
         Iterations: 7
         Function evaluations: 10
         Gradient evaluations: 10
```

Report standard errors for each estimate by the covariance matrix W:

In [44]:
```
diag = np.sqrt(np.diag(W))
diag*(1/math.sqrt(len(sim_rates)))
```

Out[44]:
```
array([2.70810761e-06, 2.58489589e-07, 7.79064173e-08, 7.73032351e-0
9])
```

Now, estimate the new parameters with the new weight matrix W_prime

In [45]:
```
delta = 1
rt_new = sim_rates[1:]
rt_old = sim_rates[:-1]

def gmm_sim(gamma):

    kappa, theta, sigma_sqr = gamma

    h1 = rt_new - rt_old - kappa*(theta - rt_old)*delta
    h2 = h1*rt_old
    h3 = h1**2 - sigma_sqr*rt_old*delta
    h4 = h3*rt_old

    g = np.column_stack((h1, h2, h3, h4)).mean(axis=0)

    obj = np.sum(np.transpose(g)*W_prime*g)

    return obj
```

In [46]:
```
gmm_optimal_sim_new_weight = optimize.fmin(gmm_sim, [0.2, 0.1, 0.01],
maxiter=1000)
print(gmm_optimal_sim_new_weight)
```

```
Warning: Maximum number of iterations has been exceeded.
[-9.15951495e+22 -5.82096721e+50  1.14729229e+50]

/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/ipykernel_launcher.py:16: RuntimeWarning: overflow en
countered in multiply
  app.launch_new_instance()
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/numpy/core/fromnumeric.py:86: RuntimeWarning: invalid
value encountered in reduce
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/scipy/optimize/optimize.py:734: RuntimeWarning: inval
id value encountered in subtract
  np.max(np.abs(fsim[0] - fsim[1:])) <= fatol):
```

4.2 QML

In [47]:

```
delta = 1
```

In [48]:

```
def qml_sim(gamma):

    kappa, theta, sigma_sqr = gamma

    mu_rt = sim_rates * np.exp(-kappa*delta) + theta * (1-np.exp(-kappa*delta))
    sigma_sqr_rt = sim_rates * sigma_sqr / kappa *(np.exp(-kappa*delta) - np.exp(-2*kappa)) + theta * sigma_sqr / (2*kappa*delta) * (1 - np.exp(- kappa*delta))**2

    mu_rt = mu_rt[:-1]
    sigma_sqr_rt = sigma_sqr_rt[:-1]
    rt = sim_rates[1:]

    return np.sum(np.log(sigma_sqr_rt) + (rt-mu_rt)**2/sigma_sqr_rt)/len(sim_rates)

def qml_rates(gamma):

    kappa, theta, sigma_sqr = gamma

    mu_rt = rates * np.exp(-kappa*delta) + theta * (1-np.exp(-kappa*delta))
    sigma_sqr_rt = rates * sigma_sqr / kappa *(np.exp(-kappa*delta) - np.exp(-2*kappa)) + theta * sigma_sqr / (2*kappa*delta) * (1 - np.exp(- kappa*delta))**2

    mu_rt = mu_rt[:-1]
    sigma_sqr_rt = sigma_sqr_rt[:-1]
    rt = rates[1:]

    return np.sum(np.log(sigma_sqr_rt) + (rt-mu_rt)**2/sigma_sqr_rt)/len(rates)
```

In [49]:
```
qml_optimal_sim = optimize.fmin(qml_sim, [0.2, 0.1, 0.01], maxiter=100
0)
print(qml_optimal_sim)

qml_optimal_rates = optimize.fmin(qml_rates, [0.2, 0.1, 0.01], maxiter
=1000)
print(qml_optimal_rates)
```

```
Optimization terminated successfully.
         Current function value: -6.665843
         Iterations: 64
         Function evaluations: 116
[0.31486353 0.08421142 0.00773132]

/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/ipykernel_launcher.py:25: RuntimeWarning: invalid val
ue encountered in log
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/ipykernel_launcher.py:25: RuntimeWarning: divide by z
ero encountered in log

Optimization terminated successfully.
         Current function value: -65.567240
         Iterations: 410
         Function evaluations: 773
[1.35971836e-16 2.34850676e-01 1.24555219e-31]
```

To numerically calculate the derivatives for the covariance matrix from 4.2 to 4.4, it could be done with the aim of functions written below,

In [50]:
```
def first_derivative(gamma, f):

    tol = 1e-8

    g_1 = gamma.copy()
    g_1[0] += tol
    g_2 = gamma.copy()
    g_2[0] -= tol
    g_first = (f(g_1) - f(g_2))/(2*tol)

    g_1 = gamma.copy()
    g_1[1] += tol
    g_2 = gamma.copy()
    g_2[1] -= tol
    g_second = (f(g_1) - f(g_2))/(2*tol)

    g_1 = gamma.copy()
    g_1[2] += tol
    g_2 = gamma.copy()
    g_2[2] -= tol
    g_third = (f(g_1) - f(g_2))/(2*tol)

    derivative = np.array([g_first, g_second, g_third])

    return derivative
```

```
In [51]: def second_gradient(gamma, f):

    tol = 1e-8

    g_1 = gamma.copy()
    g_1[0] += tol
    g_2 = gamma.copy()
    g_2[0] -= tol
    g_first = (first_derivative(g_1, f) - first_derivative(g_2, f))/(2
*tol)

    g_1 = gamma.copy()
    g_1[1] += tol
    g_2 = gamma.copy()
    g_2[1] -= tol
    g_second = (first_derivative(g_1, f) - first_derivative(g_2, f))/(
2*tol)

    g_1 = gamma.copy()
    g_1[2] += tol
    g_2 = gamma.copy()
    g_2[2] -= tol
    g_third = (first_derivative(g_1, f) - first_derivative(g_2, f))/(2
*tol)

    derivative = np.array([g_first, g_second, g_third])

    return derivative
```

Then, by passing in our optimal estimates and likelihood function into the functions above, we could then get the covariance matrix with `{linalg.inv() @ I@ linalg()}` . This also applies to the section 4.3 and 4.4 in calculating covariance matrix.

4.3 Transformed Approximated Likelihood

```
In [52]: y_sim = 2 * np.sqrt(sim_rates)
y_rates = 2 * np.sqrt(rates)
```

In [53]:
```
def trans_sim(gamma):

    kappa, theta, sigma_sqr = gamma

    yt = y_sim[:-1]

    a = 2/yt *(kappa * theta - sigma_sqr/4) - kappa *yt/2
    b = -2/(yt**2) *(kappa * theta - sigma_sqr/4) - kappa/2
    c = 4/(yt**3) *(kappa * theta - sigma_sqr/4)
    K = np.exp(b*delta) - 1
    v = sigma_sqr/(2*b) * (np.exp(2*b*delta)-1)
    m = yt + a/b*K + sigma_sqr * c / (2*b**2) * (K - b)
    l = np.sum((y_sim[1:] - m)/(2*v) + 1/2 * np.log(2 * np.pi * v))

    return l/len(y_sim)

def trans_rates(gamma):

    kappa, theta, sigma_sqr = gamma

    yt = y_rates[:-1]

    a = 2/yt *(kappa * theta - sigma_sqr/4) - kappa *yt/2
    b = -2/(yt**2) *(kappa * theta - sigma_sqr/4) - kappa/2
    c = 4/(yt**3) *(kappa * theta - sigma_sqr/4)
    K = np.exp(b) - 1
    v = sigma_sqr/ 2/ b * (np.exp(2*b)-1)
    m = yt + a/b*K + sigma_sqr * c / (2*b**2) * (K - b)
    l = np.sum((y_rates[1:] - m)/(2*v) + 1/2 * np.log(2 * np.pi * v))

    return l/len(y_rates)
```

In [54]:
```
trans_optimal_sim = optimize.fmin(trans_sim, [0.2, 0.1, 0.01], maxiter
=1000)
print(trans_optimal_sim)

trans_optimal_rates = optimize.fmin(trans_rates, [0.2, 0.1, 0.01], max
iter=1000)
print(trans_optimal_rates)
```

```
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/ipykernel_launcher.py:13: RuntimeWarning: invalid val
ue encountered in log
  del sys.path[0]

Warning: Maximum number of iterations has been exceeded.
[2.65459149e-01 1.81994851e-01 2.81600286e-92]

/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/ipykernel_launcher.py:29: RuntimeWarning: invalid val
ue encountered in log

Warning: Maximum number of iterations has been exceeded.
[-1.13161774e+00  4.71622701e-01  1.69450684e-90]
```

4.4 Exact Likelihood for the CIR model

In [55]:
```
def exact_CIR_sim(gamma):

    kappa, theta, sigma_sqr = gamma

    c = 2 * kappa / ((sigma_sqr)*(1- np.exp(-kappa*delta)))
    u = c * sim_rates[:-1] * np.exp(-kappa*delta)
    v = c * sim_rates[1:]
    q = 2 * kappa * theta / sigma_sqr - 1
    I = scipy.special.iv(1, 2* np.sqrt(u*v))
    f = -np.log(c * np.exp(-u-v) * (v/u) ** (q/2) * I)

    return np.sum(f)

def exact_CIR_rates(gamma):

    kappa, theta, sigma_sqr = gamma

    c = 2 * kappa / ((sigma_sqr)*(1- np.exp(-kappa*delta)))
    u = c * rates[:-1] * np.exp(-kappa*delta)
    v = c * rates[1:]
    q = 2 * kappa * theta / sigma_sqr - 1
    I = scipy.special.iv(1, 2* np.sqrt(u*v))
    f = -np.log(c * np.exp(-u-v) * (v/u) ** (q/2) * I)

    return np.sum(f)
```

In [56]:
```python
exact_optimal_sim = optimize.fmin(exact_CIR_sim, [0.2, 0.1, 0.01], max
iter=1000)
print(exact_optimal_sim)

exact_optimal_rates = optimize.fmin(exact_CIR_rates, [0.2, 0.1, 0.01],
maxiter=1000)
print(exact_optimal_rates)
```

```
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/ipykernel_launcher.py:10: RuntimeWarning: overflow en
countered in power
  # Remove the CWD from sys.path while we load stuff.
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/ipykernel_launcher.py:10: RuntimeWarning: invalid val
ue encountered in multiply
  # Remove the CWD from sys.path while we load stuff.
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/ipykernel_launcher.py:10: RuntimeWarning: invalid val
ue encountered in log
  # Remove the CWD from sys.path while we load stuff.
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/ipykernel_launcher.py:10: RuntimeWarning: overflow en
countered in multiply
  # Remove the CWD from sys.path while we load stuff.
/Users/chih-hsuankao/.pyenv/versions/anaconda3-2019.03/lib/python3.7
/site-packages/ipykernel_launcher.py:10: RuntimeWarning: divide by z
ero encountered in log
  # Remove the CWD from sys.path while we load stuff.

Warning: Maximum number of iterations has been exceeded.
[0.34729966 0.13012886 0.00088032]
Warning: Maximum number of iterations has been exceeded.
[0.2  0.1  0.01]
```

In [ ]:

4.6 Comment

In general, for simulated data, the estimated parameters are all more accurate compared to the real monthly interest rates. Among them QML can give us the most accurate estimations for all 3 parameters. Surprisingly, the parameters estimated for the real monthly interest rates can be unreasonable for some cases and it's very hard to get the correct estimation.