Estimating and Modeling Volatility

Thus far, we have assumed that the volatility of the underlying asset is constant or varying in a non-random way during the lifetime of the derivative. In this chapter we will look at models that relax this assumption and allow the volatility to change randomly. This is very important, because there is plenty of evidence that volatilities do change over time in a random way.

In the first three sections, we will consider the problem of estimating the volatility. The discussion of estimation methods leads naturally into the discussion of modeling a changing volatility.

Statistics Review

We begin with a brief review of basic statistics. Given a random sample $\{x_1, \ldots, x_N\}$ of size N from a population with mean μ and variance σ^2, the best estimate of μ is of course the sample mean

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \ .$$

The variance is the expected value of $(x - \mu)^2$, so an obvious estimate of the variance is the sample average of $(x_i - \mu)^2$, replacing μ with its estimate $\bar{x}$. This would be

$$\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2$$

However, because $\bar{x}$ is computed from the x_i, the x_i will deviate less on average from $\bar{x}$ than they do from the true mean μ. Hence the estimate proposed above will on average be less than σ^2. To eliminate this bias, it suffices just to scale the estimate up by a factor of $N/(N-1)$. This leads to the estimate

$$s^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2 \ ,$$

and the best estimate of σ is the square root

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \bar{x})^2} \ .$$

To calculate s^2, notice that

$$\begin{aligned}\sum_{i=1}^{N}(x_i-\bar{x})^2 &= \sum_{i=1}^{N}(x_i^2-2x_i\bar{x}+\bar{x}^2)\\ &= \sum_{i=1}^{N}x_i^2-2\bar{x}\sum_{i=1}^{N}x_i+\sum_{i=1}^{N}\bar{x}^2\\ &= \sum_{i=1}^{N}x_i^2-2\bar{x}(N\bar{x})+N\bar{x}^2\\ &= \sum_{i=1}^{N}x_i^2-N\bar{x}^2\ .\end{aligned}$$

Therefore

$$s=\sqrt{\frac{1}{N-1}\left(\sum_{i=1}^{N}x_i^2-N\bar{x}^2\right)}\ .$$

It is important to know how much variation there would be in $\bar{x}$ if one had access to multiple random samples. More variation means that an $\bar{x}$ computed from a single sample will be a less reliable estimate of μ. The variance of $\bar{x}$ in repeated samples is σ^2/N,[1] and our best estimate of this variance is s^2/N. The standard deviation of $\bar{x}$ in repeated samples, which is called the standard error of $\bar{x}$, is $\sigma/\sqrt{N}$, and we estimate this by $s/\sqrt{N}$, which equals

$$\sqrt{\frac{1}{N(N-1)}\left(\sum_{i=1}^{N}x_i^2-N\bar{x}^2\right)}\ .$$

If the population from which x is sampled has a normal distribution, then a 95% confidence interval for μ will be $\bar{x}$ plus or minus 1.96 standard errors. Even if x does not have a normal distribution, by the Central Limit Theorem, $\bar{x}/\sqrt{N}$ will be approximately normally distributed if the sample size N is large enough, and plus or minus 1.96 standard errors will still be approximately a 95% confidence interval for μ.

Estimating a Constant Volatility and Mean

Consider an asset price that is a geometric Brownian motion under the actual probability measure:

$$\frac{\mathrm{d}S}{S}=\mu\,\mathrm{d}t+\sigma\,\mathrm{d}B\ ,$$

[1] The variance of $\bar{x}=(1/N)(x_1+\cdots+x_N)$ is, by independence of the x_i, equal to $(1/N)^2(\mathrm{var}x_1+\cdots+\mathrm{var}x_N)$, and, because the x_i all have the same variance σ^2, this is equal to $(1/N)^2\times N\sigma^2=\sigma^2/N$.

where μ and σ are unknown constants and B is a Brownian motion. We can as usual write this in log form as

$$\mathrm{d}\log S = \left(\mu - \frac{1}{2}\sigma^2\right)\,\mathrm{d}t + \sigma\,\mathrm{d}B\ .$$

Over a discrete time period of length Δt, this implies

$$\Delta \log S = \left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma \Delta B\ . \tag{1}$$

Suppose we have observed the asset price S at dates $0 = t_0 < t_1 < \cdots < t_N = T$, where $t_i - t_{i-1} = \Delta t$. If the asset pays dividends, we will take S to be the value of the portfolio in which the dividends are reinvested in new shares. Thus, in general, $S(t_i)/S(t_{i-1})$ denotes the gross return (one plus the rate of return) between dates t_{i-1} and t_i. This return is measured on a non-compounded and non-annualized basis. The annualized continuously-compounded rate of return is the rate r_i defined by

$$\frac{S(t_i)}{S(t_{i-1})} = \mathrm{e}^{r_i \Delta t}\ .$$

This implies that

$$r_i = \frac{\log S(t_i) - \log S(t_{i-1})}{\Delta t} = \mu - \frac{1}{2}\sigma^2 + \sigma\frac{B(t_i) - B(t_{i-1})}{\Delta t}\ . \tag{2}$$

Because $B(t_i) - B(t_{i-1})$ is normally distributed with mean zero and variance Δt, the sample $\{r_1, \ldots, r_N\}$ is a sample of independent random variables each of which is normally distributed with mean $\mu - \sigma^2/2$ and variance $\sigma^2/\Delta t$. We are focused on estimating σ^2, so it will simplify things to define

$$y_i = r_i\sqrt{\Delta t} = \frac{\log S(t_i) - \log S(t_{i-1})}{\sqrt{\Delta t}}\ . \tag{3}$$

The sample $\{y_1, \ldots, y_N\}$ is a sample of independent random variables each of which is normally distributed with mean $(\mu - \sigma^2/2)\sqrt{\Delta t}$ and variance σ^2. As was discussed in the previous section, the best estimate of the mean of y is the sample mean

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i\ ,$$

and the best estimate of σ^2 is

$$\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \bar{y})^2\ .$$

This means that we estimate μ as

$$\hat{\mu} = \frac{\bar{y}}{\sqrt{\Delta t}} + \frac{1}{2}\hat{\sigma}^2 = \bar{r} + \frac{1}{2}\hat{\sigma}^2\ .$$

Let us digress for a moment to discuss the reliability of $\hat{\mu}$ as an estimate of μ. Notice that

$$\begin{aligned}\bar{r} &= \frac{\sum_{i=1}^{N} \log S(t_i) - \log S(t_{i-1})}{N\Delta t} \\ &= \frac{\log S(T) - \log S(0)}{N\Delta t} \\ &= \frac{\log S(T) - \log S(0)}{T} \ . \end{aligned} \tag{4}$$

Therefore the first component $\bar{r}$ of the estimate of μ depends only on the total change in S over the time period. Hence, the reliability of this component cannot depend on how frequently we observe S within the time period $[0, T]$. The standard deviation of $\bar{r}$ in repeated samples is the standard deviation of $[\log S(T) - \log S(0)]/T$, which is $\sigma/\sqrt{T}$. This is likely to be quite large. For example, with $\sigma = 0.3$ and ten years of data ($T = 10$), the standard deviation of $\bar{r}$ is 9.5%, which means that a 95% confidence interval will be a band of roughly 38%. Given that μ itself should be of the order of magnitude of 10%, such a wide confidence interval is useless for all practical purposes.

Fortunately, it is easier to estimate σ. We observed in the previous section that the $\hat{\sigma}^2$ defined above can be calculated as

$$\frac{1}{N-1}\sum_{i=1}^{N} y_i^2 - \frac{N\bar{y}^2}{N-1} \ . \tag{5}$$

From Equation 3 of y_i and Equation 4, we have

$$\bar{y} = \frac{\sqrt{\Delta t}}{T}[\log S(T) - \log S(0)] \ .$$

Hence, the second term in Equation 5 is

$$\frac{N}{N-1}\left(\frac{\Delta t}{T^2}\right)[\log S(T) - \log S(0)]^2 \ .$$

If we observe the stock price sufficiently frequently, so that Δt is very small, this term will be negligible. In this circumstance, $\hat{\sigma}^2$ is approximately

$$\begin{aligned}\frac{1}{N-1}\sum_{i=1}^{N} y_i^2 &= \frac{1}{N-1}\sum_{i=1}^{N} \frac{[\log S(t_i) - \log S(t_{i-1})]^2}{\Delta t} \\ &= \frac{N}{N-1} \times \frac{1}{T} \times \sum_{i=1}^{N} [\log S(t_i) - \log S(t_{i-1})]^2 \ . \end{aligned} \tag{6}$$

If we observe S more and more frequently, letting $\Delta t \to 0$ and $N \to \infty$, the sum

$$\sum_{i=1}^{N} [\log S(t_i) - \log S(t_{i-1})]^2$$

will converge with probability one to $\sigma^2 T$, as explained in **?@sec-s_quadraticvariation**. This implies that $\hat{\sigma}^2$ will converge to σ^2. Thus, in theory, we can estimate σ^2 with any desired degree of precision by simply observing S sufficiently frequently. This is true no matter how short the overall time period $[0, T]$ may be.

In practice, this doesn't work out quite so well. If we observe minute-by-minute data, or we observe each transaction, much of the variation in the price S will be due to bouncing back and forth between the bid price and the ask price. This is not really what we want to estimate, and this source of variation will be much less important if we look at weekly or even daily data. So, there are practical limits to how frequently we should observe S. Nevertheless, it is still true that, if σ^2 were truly constant, we could estimate it with a very high degree of precision. In fact, we can estimate the volatility of a stock with enough precision to determine that it really isn't constant! The real problem that we face is to estimate and model a changing volatility.

Estimating a Changing Volatility

Without attempting yet to model how the volatility may change, we can say a few things about how we might estimate a changing volatility. In this and following sections, we will take the observation interval Δt to be fixed. We assume it is small (say, a day or a week) and focus on the estimate Equation 6. Recall from Section that the reason we are dividing by $N - 1$ rather than N is that the sample standard deviation usually underestimates the actual standard deviation, because it uses the sample mean, which will be closer to the points x_i than will be the true mean. However, Equation 6 does not employ the sample mean (it replaces it with zero), so there is no reason to make this correction. So, we take as our point of departure the estimate

$$\frac{1}{T} \sum_{i=1}^{N} [\log S(t_i) - \log S(t_{i-1})]^2 = \frac{1}{N} \sum_{i=1}^{N} y_i^2 \ .$$

An obvious response to the volatility changing over time is simply to avoid using data from the distant past. Such data is not likely to be informative about the current value of the volatility. What distant should mean in this context is not entirely clear, but, for example, we might want to use only the last 60 observations. If we are using daily data, this would mean that at the end of each day we would add that day's observation and drop the observation from 61 days past. This leads to a somewhat abruptly varying estimate. For example, a very large movement in the price on a particular day increases the volatility estimate for the next 60 days. On the 61st day, this observation would drop from the sample, leading to an abrupt drop in the estimate (presuming that there is not an equally large change in S on the 61st

day). This seems unreasonable. An estimate in which the impact of each observation decays smoothly over time is more attractive.

We can construct such an estimate as

$$\hat{\sigma}_{i+1}^2 = (1-\lambda)y_i^2 + \lambda\hat{\sigma}_i^2 \tag{7}$$

for any constant $0 < \lambda < 1$. Here, $\hat{\sigma}_{i+1}^2$ denotes the estimate of the volatility from date t_i to date t_{i+1}. The estimate Equation 7 is a weighted average of the estimate $\hat{\sigma}_i^2$ for the previous time period and the most recently observed squared change y_i^2. Following the same procedure, the next estimate will be

$$\begin{aligned}\hat{\sigma}_{i+2}^2 &= (1-\lambda)y_{i+1}^2 + \lambda\hat{\sigma}_{i+1}^2 \\ &= (1-\lambda)y_{i+1}^2 + \lambda(1-\lambda)y_i^2 + \lambda^2\hat{\sigma}_i^2 \ .\end{aligned}$$

Likewise, the estimate at the following date will be

$$\hat{\sigma}_{i+3}^2 = (1-\lambda)y_{i+2}^2 + \lambda(1-\lambda)y_{i+1}^2 + \lambda^2(1-\lambda)^2 y_i^2 + \lambda^3\hat{\sigma}_i^2 \ .$$

This demonstrates the declining importance of the squared deviation y_i^2 for future estimates. At each date, y_i^2 enters with a weight that is lower by a factor of λ, compared to the previous date. If λ is small, the decay in the importance of each squared deviation will be fast. In fact, Equation 7 shows that, if λ is close to zero, the estimate $\hat{\sigma}_{i+1}^2$ is approximately equal to the squared deviation y_i^2—previous squared deviations are relatively unimportant. On the other hand, if λ is close to one, the decay will be slow; i.e., the importance of y_i^2 for the estimate $\hat{\sigma}_{i+2}^2$ will be nearly the same as for $\hat{\sigma}_{i+1}^2$, and nearly the same for $\hat{\sigma}_{i+3}^2$ as for $\hat{\sigma}_{i+2}^2$, etc. This will lead to a smooth (slowly varying) volatility estimate. The slowly varying nature of the estimate in this case is also clear from Equation 7, because it shows that if λ is close to one, then $\hat{\sigma}_{i+1}^2$ will be approximately the same as $\hat{\sigma}_i^2$.

This method can also be used to estimate covariances, simply by replacing the squared deviations y_i^2 by the product of deviations for two different assets. And, of course, given covariance and variance estimates, we can construct estimates of correlations. To ensure that an estimated correlation is between -1 and $+1$, we will need to use the same λ to estimate each of the variances and the covariance. This is the method used by RiskMetrics.^[See Mina and Xiao [@MX], available online at *www.riskmetrics.com].*

GARCH Models

We are going to adopt a subtle but important change of perspective now. Instead of considering Equation 7 as simply an estimation procedure, we are going to assume that the actual volatility evolves according to Equation 7, or a generalization thereof. We are also going to reintroduce the expected change in $\log S$, which we dropped in going from Equation 5 to Equation 6.

Specifically, we return to Equation 1, but we operate under the risk-neutral measure, so $\mu = r - q$, and we have

$$\log S(t_{i+1}) - \log S(t_i) = \left(r - q - \frac{1}{2}\sigma_{i+1}^2\right)\Delta t + \sigma_{i+1}\Delta B \; . \tag{8}$$

We assume the volatility σ_{i+1} between dates t_i and t_{i+1} is given by

$$\sigma_{i+1}^2 = a + by_i^2 + c\sigma_i^2 \; , \tag{9}$$

for some constants $a > 0$, $b \geq 0$ and $c \geq 0$, with y_i now defined by

$$y_i = \frac{\log S(t_i) - \log S(t_{i-1}) - \left(r - q - \frac{1}{2}\sigma_i^2\right)\Delta t}{\sqrt{\Delta t}} \; .$$

From Equation 8, applied to the period from t_{i-1} to t_i, this implies that y_i is normally distributed with mean zero and variance σ_i^2, and of course y_{i+1} has variance σ_{i+1}^2, etc.
Under these assumptions, the random process $\log S$ is called a GARCH(1,1) process.[2] There are many varieties of GARCH processes that have been proposed in the literature, but we will only consider GARCH(1,1), which is the simplest.

We assume $b + c < 1$, in which case we can write the variance equation as a generalization of Equation 7. Namely, %

$$\begin{aligned} \sigma_{i+1}^2 &= (1-\phi)d + \phi\left[(1-\lambda)y_i^2 + \lambda\sigma_i^2\right] \; , \\ \sigma_{i+1}^2 &= \kappa\theta + (1-\kappa)\left[(1-\lambda)y_i^2 + \lambda\sigma_i^2\right] \; , \end{aligned} \tag{10}$$

where $\lambda = c/(b+c)$, $\%\phi = b + c$, and $d = a/(1-b-c)$.
$\kappa = 1 - b - c$, and $\theta = a/(1-b-c)$. Hence, σ_{i+1}^2 is a weighted average with weights κ and $1-\kappa$, of two parts, one being the constant θ and the other being itself a weighted average of y_i^2 and σ_i^2. Whatever the variance might be at time t_i, the variance of y_j at any date t_j far into the future, computed without knowing the intervening $y_{i+1}, y_{i+2}, ...$, will be approximately the constant θ. The constant θ is called the unconditional variance, whereas σ_i^2 is the conditional variance of y_i.

To understand the unconditional variance, it is useful to consider the variance forecasting equation. Specifically, we can calculate $E_{t_i}\left[\sigma_{i+n}^2\right]$, which is the estimate made at date t_i of the variance of y_{i+n}; i.e, we estimate the variance without having observed $y_{i+1}, \ldots, y_{i+n-1}$. Note that by definition $E_{t_i}[y_{i+1}^2] = \sigma_{i+1}^2$, so Equation 10 implies

$$\begin{aligned} E_{t_i}\left[\sigma_{i+2}^2\right] &= \kappa\theta + (1-\kappa)\left[(1-\lambda)E_{t_i}[y_{i+1}^2] + \lambda\sigma_{i+1}^2\right] \\ &= \kappa\theta + (1-\kappa)\sigma_{i+1}^2 \; . \end{aligned}$$

[2] GARCH is the acronym for Generalized Autoregressive Conditional Heteroskedastic. GARCH(1,1) means that there is only one past y (no y_{i-1}, y_{i-2}, etc.) and one past σ (no σ_{i-1}, σ_{i-2}, etc.) in Equation 9. See Bollerslev [@Bollerslev].

Likewise,

$$E_{t_{i+1}}\left[\sigma_{i+3}^2\right] = \kappa\theta + (1-\kappa)\sigma_{i+2}^2 \ ,$$

and taking the expectation at date t_i of both sides of this yields

$$\begin{aligned} E_{t_i}\left[\sigma_{i+3}^2\right] = E_{t_i}\left[E_{t_{i+1}}\left[\sigma_{i+3}^2\right]\right] &= \kappa\theta + (1-\kappa)E_{t_i}\left[\sigma_{i+2}^2\right] \\ &= \kappa\theta + (1-\kappa)\left[\kappa\theta + (1-\kappa)\sigma_{i+1}^2\right] \\ &= \kappa\theta[1 + (1-\kappa)] + (1-\kappa)^2\sigma_{i+1}^2 \ . \end{aligned}$$

This generalizes to

$$E_{t_i}\left[\sigma_{i+n}^2\right] = \kappa\theta\left[1 + (1-\kappa) + \cdots (1-\kappa)^{n-2}\right] + (1-\kappa)^{n-1}\sigma_{i+1}^2 \ .$$

Thus, there is decay at rate κ in the importance of the current volatility σ_{i+1}^2 for forecasting the future volatility. Furthermore, as $n \to \infty$, the geometric series

$$1 + (1-\kappa) + \cdots (1-\kappa)^{n-2}$$

converges to $1/\kappa$, so, as $n \to \infty$ we obtain

$$E_{t_i}\left[\sigma_{i+n}^2\right] \to \theta \ .$$

This means that our best estimate of the conditional variance, at some date far in the future, is approximately the unconditional variance θ.

The most interesting feature of the volatility equation is that large returns (in absolute value) lead to an increase in the variance and hence are likely to be followed by more large returns (whether positive or negative). This is the phenomenon of volatility clustering, which is quite observable in actual markets. This feature also implies that the distribution of returns will be fat tailed (more technically, leptokurtic). This means that the probability of extreme returns is higher than under a normal distribution with the same standard deviation.[3] It is well documented that daily and weekly returns in most markets have this fat-tailed property.

We can simulate a path of an asset price that follows a GARCH process and the path of its volatility as follows. The following python code produces three columns of data (with headings), the first column being time, the second the asset price, and the third the volatility.

```
import numpy as np
import pandas as pd

def simulating_garch(S, sigma, r, q, dt, N, theta, kappa, lambd):
    """
    Inputs:
```

[3] Conversely, the probability of returns very near the mean must also be higher than under a normal distribution with the same standard deviation—a fat-tailed distribution must also have a relatively narrow peak.

```
    S = initial stock price
    sigma = initial volatility
    r = risk-free rate
    q = dividend yield
    dt = length of each time period (Delta t)
    N = number of time periods
    theta = theta parameter for GARCH
    kappa = kappa parameter for GARCH
    lambd = lambda parameter for GARCH
    """
    LogS = np.log(S)
    Sqrdt = np.sqrt(dt)
    a = kappa * theta
    b = (1 - kappa) * (1 - lambd)
    c = (1 - kappa) * lambd

    time = np.zeros(N + 1)
    stock_price = np.zeros(N + 1)
    volatility = np.zeros(N + 1)

    stock_price[0] = S
    volatility[0] = sigma

    for i in range(1, N + 1):
        time[i] = i * dt
        y = sigma * np.random.randn()
        LogS = LogS + (r - q - 0.5 * sigma * sigma) * dt + Sqrdt * y
        S = np.exp(LogS)
        stock_price[i] = S
        sigma = np.sqrt(a + b * y ** 2 + c * sigma ** 2)
        volatility[i] = sigma

    df_garch = pd.DataFrame({'Time': time, 'Stock Price': stock_price, 'Volatility': volatili
    df_garch.to_csv('garch_simulation.csv', index=False)
    return df_garch

# Example usage:
S = 100       # Initial stock price
sigma = 0.2   # Initial volatility
r = 0.05      # Risk-free rate
q = 0.02      # Dividend yield
dt = 1/252    # Length of each time period (daily)
```

```
N = 252       # Number of time periods (one year)
theta = 0.1   # Theta parameter for GARCH
kappa = 0.1   # Kappa parameter for GARCH
lambd = 0.9   # Lambda parameter for GARCH

df_garch = simulating_garch(S, sigma, r, q, dt, N, theta, kappa, lambd)
print(df_garch)
```

```
         Time  Stock Price  Volatility
0    0.000000   100.000000    0.200000
1    0.003968   100.092299    0.205956
2    0.007937   100.325207    0.210896
3    0.011905   101.151540    0.218039
4    0.015873   101.298580    0.220351
..        ...          ...         ...
248  0.984127    96.534121    0.456626
249  0.988095    92.819286    0.461838
250  0.992063    90.704186    0.441025
251  0.996032    89.940986    0.411177
252  1.000000    90.348938    0.383997

[253 rows x 3 columns]
```

To price European options, we need to compute the usual probabilities $\text{prob}^S(S(T) > K)$ and $\text{prob}^R(S(T) > K)$. Heston and Nandi [@HN] provide a fast method for computing these probabilities in a GARCH (1,1) model.[4] Rather than developing this approach, we will show in **?@sec-c_introcomputation** how to apply Monte-Carlo methods.

Stochastic Volatility Models

The volatility is stochastic (random) in a GARCH model, but it is determined by the changes in the stock price. In this section, in contrast, we will consider models in which the volatility depends on a second Brownian motion. The most popular model of this type is the model of Heston [@Heston]. In this model, we have, as usual,

$$\mathrm{d}\log S = \left(r - q - \frac{1}{2}\sigma^2\right)\,\mathrm{d}t + \sigma\,\mathrm{d}B_s\,, \tag{11}$$

[4] Actually, a slightly more general model is considered in [@HN], in which large negative returns lead to a greater increase in volatility than do large positive returns. This accommodates the empirically observed negative correlation between stock returns and volatility.

where B_s is a Brownian motion under the risk-neutral measure but now σ is not a constant but instead evolves as $\sigma(t) = \sqrt{v(t)}$ where

$$dv(t) = \kappa[\theta - v(t)]\,\mathrm{d}t + \gamma\sqrt{v(t)}\,\mathrm{d}B_v\,, \tag{12}$$

where B_v is a Brownian motion under the risk-neutral measure having a constant correlation ρ with the Brownian motion B_s. In this equation, κ, θ and γ are positive constants. Given the empirical fact that negative return shocks have a bigger impact on future volatility than do positive shocks, one would expect the correlation ρ to be negative.

The term $\kappa(\theta - v)$ will be positive when $v < \theta$ and negative when $v > \theta$ and hence $\sigma^2 = v$ will tend to drift towards θ, which, as in the GARCH model, is the long-run or unconditional mean of σ^2. Thus, the volatility is said to mean revert. The rate at which it drifts towards θ is obviously determined by the magnitude of κ, also as in the GARCH model.

The specification Equation 12 implies that the volatility of v approaches zero whenever v approaches zero. In this circumstance, one might expect the drift towards θ to dominate the volatility and keep v nonnegative, and this is indeed the case; thus, the definition $\sigma(t) = \sqrt{v(t)}$ is possible. Moreover, the parameter γ plays a role here that is similar to the role of $1 - \lambda$ in the GARCH model—the variance of the variance in the GARCH model Equation 10 depends on the weight $1 - \lambda$ placed on the scaled return y_i, just as the variance of the variance in the stochastic volatility model Equation 12 depends on the weight γ placed on $\mathrm{d}B_v$.

We could discretize Equation 11 - Equation 12 as:

$$\log S(t_{i+1}) = \log S(t_i) + \left(r - q - \frac{1}{2}\sigma(t_i)^2\right)\,\Delta t + \sqrt{v(t_i)}\,\Delta B_s, \tag{13}$$

$$v(t_{i+1}) = v(t_i) + \kappa[\theta - v(t_i)]\,\Delta t + \gamma\sqrt{v(t_i)}\,\Delta B_v\,. \tag{14}$$

However, even though in the continuous-time model Equation 11 - Equation 12 we always have $v(t) \geq 0$ and hence can define $\sigma(t) = \sqrt{v(t)}$, there is no guarantee that $v(t_{i+1})$ defined by Equation 14 will be nonnegative. A simple remedy is to define $v(t_{i+1})$ as the larger of zero and the right-hand side of Equation 14; thus, we will simulate the Heston model as Equation 13 and[5]

$$v(t_{i+1}) = \max\left\{0, v(t_i) + \kappa[\theta - v(t_i)]\,\Delta t + \gamma\sqrt{v(t_i)}\,\Delta B_v\right\}\,. \tag{15}$$

[5]There are better (but more complicated) ways to simulate the Heston model. An excellent discussion of ways to simulate the volatility process can be found in Glasserman [@Glasserman]. Broadie and Kaya [@broadiekaya] present a method for simulating from the exact distribution of the asset price in the Heston model and related models.

A simple way to simulate the changes ΔB_s and ΔB_v in the two correlated Brownian motions is to generate two independent standard normals z_1 and z_2 and take

$$\Delta B_s = \sqrt{\Delta t}\, z \qquad \text{and} \qquad \Delta B_v = \sqrt{\Delta t}\, z^* \,,$$

where we define

$$z = z_1 \qquad \text{and} \qquad z^* = \rho z_1 + \sqrt{1-\rho^2}\, z_2 \,.$$

The random variable z^* is also a standard normal, and the correlation between z and z^* is ρ.

As in the GARCH model, we can simulate a path of an asset price that follows a GARCH process and the path of its volatility as follows. The following python code produces three columns of data (with headings), the first column being time, the second the asset price, and the third the volatility in the Heston model.

```
import numpy as np
import pandas as pd

def simulating_stochastic_volatility(S, V0, r, q, dt, N, theta, kappa, sigma, rho):
    """
    Inputs:
    S = initial stock price
    V0 = initial variance
    r = risk-free rate
    q = dividend yield
    dt = length of each time period (Delta t)
    N = number of time periods
    theta = long-term variance (mean of variance)
    kappa = rate of mean reversion of variance
    sigma = volatility of variance
    rho = correlation between the two Wiener processes
    """
    LogS = np.log(S)
    Sqrdt = np.sqrt(dt)

    time = np.zeros(N + 1)
    stock_price = np.zeros(N + 1)
    volatility = np.zeros(N + 1)

    stock_price[0] = S
    volatility[0] = V0

    for i in range(1, N + 1):
        time[i] = i * dt
```

```
        Z1 = np.random.randn()
        Z2 = np.random.randn()
        W1 = Z1
        W2 = rho * Z1 + np.sqrt(1 - rho**2) * Z2

        LogS = LogS + (r - q - 0.5 * volatility[i-1]**2) * dt + np.sqrt(volatility[i-1]**2 *
        S = np.exp(LogS)
        stock_price[i] = S

        volatility[i] = np.sqrt(np.maximum(volatility[i-1]**2 + kappa * (theta - volatility[

    df_stochastic_vol = pd.DataFrame({'Time': time, 'Stock Price': stock_price, 'Volatility'
    df_stochastic_vol.to_csv('stochastic_volatility_simulation.csv', index=False)
    return df_stochastic_vol

# Example usage:
S = 100       # Initial stock price
V0 = 0.04     # Initial variance
r = 0.05      # Risk-free rate
q = 0.02      # Dividend yield
dt = 1/252    # Length of each time period (daily)
N = 252       # Number of time periods (one year)
theta = 0.04  # Long-term variance
kappa = 2.0   # Rate of mean reversion of variance
sigma = 0.3   # Volatility of variance
rho = -0.7    # Correlation between the two Wiener processes

df_stochastic_vol = simulating_stochastic_volatility(S, V0, r, q, dt, N, theta, kappa, sigma
print(df_stochastic_vol)
```

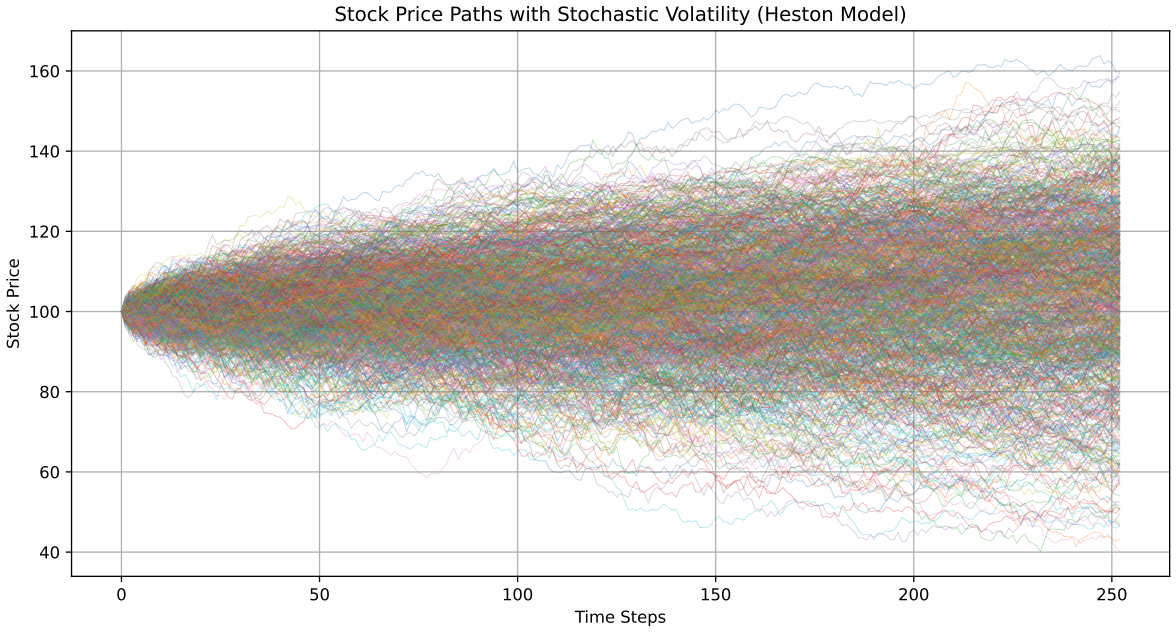
```
         Time  Stock Price  Volatility
0    0.000000   100.000000    0.040000
1    0.003968    99.768187    0.045402
2    0.007937   100.164853    0.042892
3    0.011905   100.280682    0.043155
4    0.015873   100.306242    0.047336
..        ...          ...         ...
248  0.984127   117.956373    0.152890
249  0.988095   117.050663    0.153168
250  0.992063   114.722191    0.165168
251  0.996032   115.218875    0.162966
252  1.000000   116.023920    0.142093
```

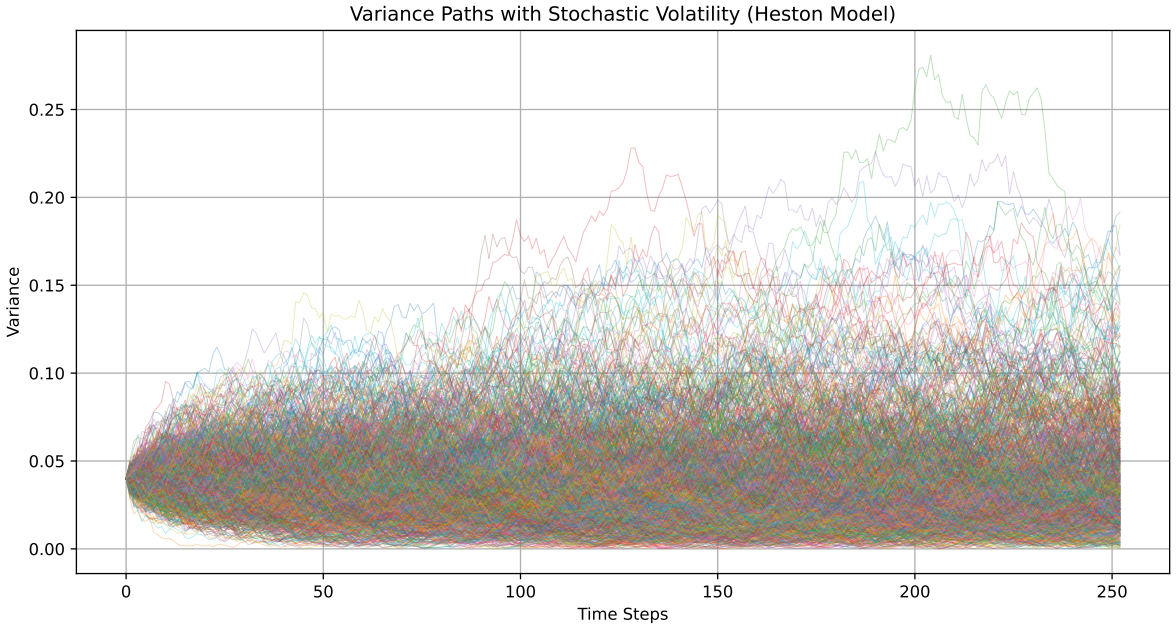
```
[253 rows x 3 columns]
```

The following code plots the simulated stock price and the variance paths.

```python
import numpy as np
import matplotlib.pyplot as plt

# Heston model parameters
S0 = 100     # Initial stock price
V0 = 0.04    # Initial variance
rho = -0.7   # Correlation between the two Wiener processes
kappa = 2.0  # Rate of mean reversion of variance
theta = 0.04 # Long-term variance
sigma = 0.3  # Volatility of variance
r = 0.05     # Risk-free rate
T = 1.0      # Time to maturity
N = 252      # Number of time steps
dt = T / N   # Time step size
n_simulations = 1000  # Number of simulations

# Preallocate arrays
S = np.zeros((N+1, n_simulations))
V = np.zeros((N+1, n_simulations))
S[0] = S0
V[0] = V0

# Simulate the paths
for t in range(1, N+1):
    Z1 = np.random.normal(size=n_simulations)
    Z2 = np.random.normal(size=n_simulations)
    W1 = Z1
    W2 = rho * Z1 + np.sqrt(1 - rho**2) * Z2

    V[t] = np.maximum(V[t-1] + kappa * (theta - V[t-1]) * dt + sigma * np.sqrt(V[t-1] * dt) *
    S[t] = S[t-1] * np.exp((r - 0.5 * V[t-1]) * dt + np.sqrt(V[t-1] * dt) * W1)

# Plot the results
plt.figure(figsize=(12, 6))
for i in range(n_simulations):
    plt.plot(S[:, i], lw=0.5, alpha=0.3)
plt.title('Stock Price Paths with Stochastic Volatility (Heston Model)')
```

```
plt.xlabel('Time Steps')
plt.ylabel('Stock Price')
plt.grid(True)
plt.show()

# Plot the volatility paths
plt.figure(figsize=(12, 6))
for i in range(n_simulations):
    plt.plot(V[:, i], lw=0.5, alpha=0.3)
plt.title('Variance Paths with Stochastic Volatility (Heston Model)')
plt.xlabel('Time Steps')
plt.ylabel('Variance')
plt.grid(True)
plt.show()
```



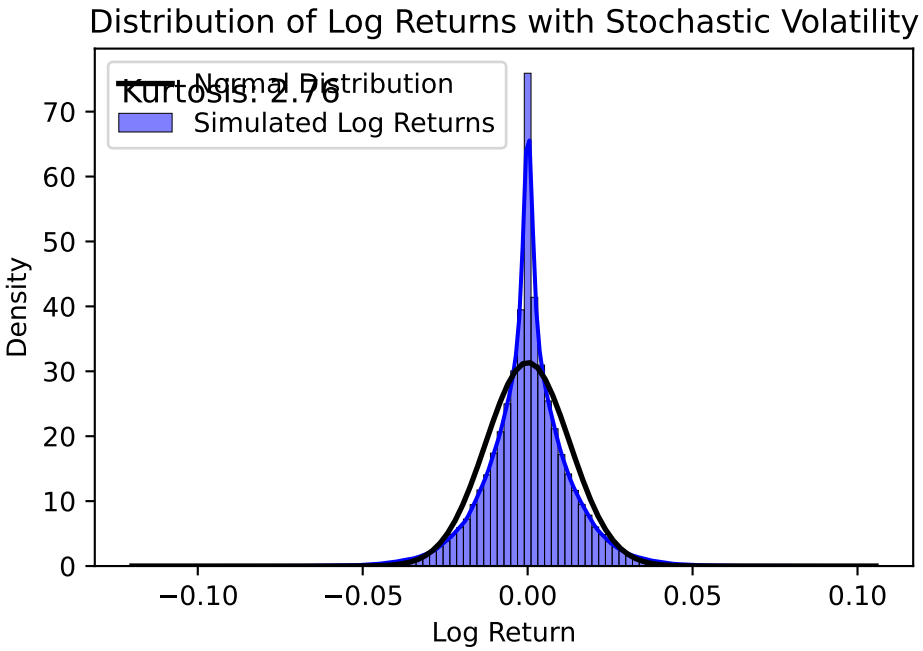


The following code shows that the stock returns under the stochastic volatility model display fat-tails with positive kurtosis.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm, kurtosis

def simulating_stochastic_volatility(S, V0, r, q, dt, N, theta, kappa, sigma, rho, n_simulati
    """
    Inputs:
    S = initial stock price
    V0 = initial variance
    r = risk-free rate
    q = dividend yield
    dt = length of each time period (Delta t)
    N = number of time periods
    theta = long-term variance (mean of variance)
    kappa = rate of mean reversion of variance
    sigma = volatility of variance
    rho = correlation between the two Wiener processes
    n_simulations = number of simulations
    """
```

```
    LogS = np.log(S)
    Sqrdt = np.sqrt(dt)

    log_returns = []

    for _ in range(n_simulations):
        stock_price = S
        variance = V0
        for _ in range(N):
            Z1 = np.random.randn()
            Z2 = np.random.randn()
            W1 = Z1
            W2 = rho * Z1 + np.sqrt(1 - rho**2) * Z2

            log_return = (r - q - 0.5 * variance) * dt + np.sqrt(variance * dt) * W1
            LogS = np.log(stock_price) + log_return
            stock_price = np.exp(LogS)

            variance = np.maximum(variance + kappa * (theta - variance) * dt + sigma * np.sq

            log_returns.append(log_return)

    return log_returns

# Example usage:
S = 100       # Initial stock price
V0 = 0.04     # Initial variance
r = 0.05      # Risk-free rate
q = 0.02      # Dividend yield
dt = 1/252    # Length of each time period (daily)
N = 252       # Number of time periods (one year)
theta = 0.04  # Long-term variance
kappa = 0.2   # Rate of mean reversion of variance
sigma = 0.3   # Volatility of variance
rho = -0.7    # Correlation between the two Wiener processes
n_simulations = 1000  # Number of simulations

log_returns = simulating_stochastic_volatility(S, V0, r, q, dt, N, theta, kappa, sigma, rho,

# Plotting the distribution of log returns
sns.histplot(log_returns, bins=100, kde=True, stat="density", color="blue", label="Simulated
xmin, xmax = plt.xlim()
```

```
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, np.mean(log_returns), np.std(log_returns))
plt.plot(x, p, 'k', linewidth=2, label="Normal Distribution")
plt.title('Distribution of Log Returns with Stochastic Volatility')
plt.xlabel('Log Return')
plt.ylabel('Density')
plt.legend()

# Display kurtosis
kurt = kurtosis(log_returns)
plt.figtext(0.15, 0.8, f'Kurtosis: {kurt:.2f}', fontsize=12)

plt.show()
```



To price European options, we again need to compute

$$\text{prob}^S(S(T) > K) \qquad \text{and} \qquad \text{prob}^R(S(T) > K)\ .$$

The virtue of modelling volatility as in Equation 12 is that these probabilities can be computed quite efficiently, as shown by Heston [@Heston].[6] There are many other ways in which one could model volatility, but the computations may be more difficult. For example, one could

[6] Further discussion can be found in Epps [@Epps].

replace Equation 12 by

$$\sigma(t) = \mathrm{e}^{v(t)} \quad \text{and} \quad dv(t) = \kappa(\theta - v(t))\,\mathrm{d}t + \lambda\,\mathrm{d}B^* \,. \tag{16}$$

This implies a lognormal volatility and is simpler to simulate than Equation 12—because e^v is well defined even when v is negative—but it is easier to calculate the probabilities $\mathrm{prob}^S(S(T) > K)$ and $\mathrm{prob}^R(S(T) > K)$ if we assume Equation 12.

One way to implement the GARCH or stochastic volatility model is to imply both the initial volatility $\sigma(0)$ and the constants κ, θ and λ or κ, θ, γ and ρ from observed option prices. These four (or five) constants can be computed by forcing the model prices of four (or five) options to equal the observed market prices. Or, a larger set of prices can be used and the constants can be chosen to minimize the average squared error or some other measure of goodness-of-fit between the model and market prices.

Jump Diffusion Models

In the classical Black-Scholes model, stock prices are assumed to follow a geometric Brownian motion, which is characterized by continuous paths and normally distributed returns. While this model has been widely used due to its simplicity and analytical tractability, it fails to capture certain empirical phenomena observed in financial markets, such as sudden and significant price changes (jumps) and the heavy tails of return distributions.

To address these shortcomings, the jump diffusion model was introduced by Robert C. Merton in 1976. This model extends the Black-Scholes framework by incorporating jumps into the stock price dynamics, thereby allowing for discontinuous price paths. The jump diffusion model is better suited to describe the behavior of financial assets that exhibit sudden price changes due to news, earnings announcements, or other market events.

In a jump diffusion model, the stock price S_t is governed by the following stochastic differential equation (SDE):

$$dS_t = \mu S_t\,dt + \sigma S_t\,dW_t + S_t\,dJ_t$$

where μ is the drift rate, σ is the volatility, B_t is a standard Brownian motion, J_t is a jump process.

The jump process J_t is typically modeled as a compound Poisson process:

$$J_t = \sum_{i=1}^{N_t} (Y_i - 1)$$

where N_t is a Poisson process with intensity λ, Y_i are i.i.d. random variables representing the relative jump sizes, with $Y_i - 1$ being the actual jump size.

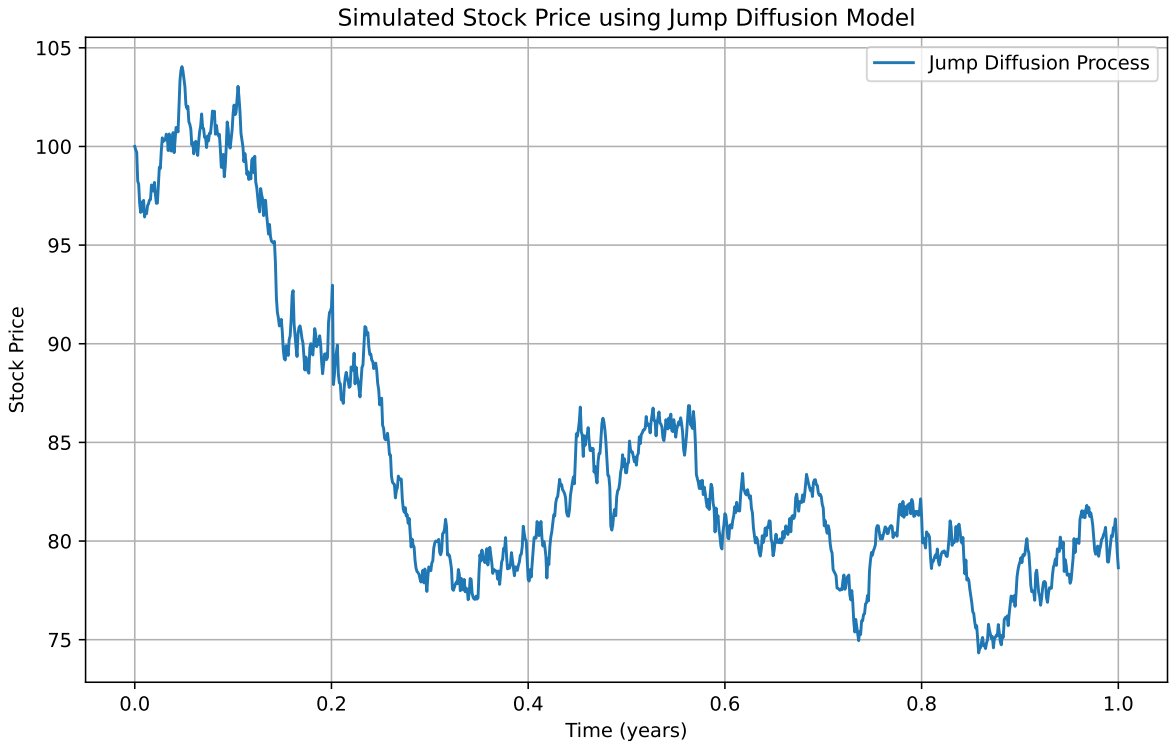
The jump diffusion model introduces several important implications for the behavior of stock prices and the pricing of derivative securities:

1. **Heavy Tails**: The inclusion of jumps leads to a return distribution with heavier tails compared to the normal distribution, aligning better with empirical observations.
2. **Volatility Smile**: The model can generate implied volatility smiles, where implied volatility varies with strike price and maturity, a feature commonly observed in market data.
3. **Risk Management**: Understanding the jump component is crucial for risk management, as it affects the likelihood of extreme price movements and the potential for large losses.

The following python code simulates the stock price that evolves as a jump diffusion.

```
import numpy as np
import matplotlib.pyplot as plt

def simulate_jump_diffusion(S0, mu, sigma, lamb, m, delta, T, N):
    """
    Simulate a jump diffusion process.

    Parameters:
    S0     : float - initial stock price
    mu     : float - drift rate
    sigma  : float - volatility
    lamb   : float - intensity of the Poisson process
    m      : float - mean of the jump size distribution
    delta  : float - standard deviation of the jump size distribution
    T      : float - total time
    N      : int   - number of time steps

    Returns:
    t      : numpy array - time points
    S      : numpy array - simulated stock prices
    """
    dt = T / N
    t = np.linspace(0, T, N + 1)
    S = np.zeros(N + 1)
    S[0] = S0

    for i in range(1, N + 1):
        Z = np.random.normal(0, 1)  # Normal random variable for the diffusion part
```

```
        J = np.random.poisson(lamb * dt)  # Poisson random variable for jumps

        # Sum of log-normal distributed jumps
        Y = np.sum(np.random.normal(m, delta, J))

        # Update stock price
        S[i] = S[i - 1] * np.exp((mu - 0.5 * sigma ** 2) * dt + sigma * np.sqrt(dt) * Z + Y)

    return t, S

# Parameters
S0 = 100          # Initial stock price
mu = 0.1          # Drift rate
sigma = 0.2       # Volatility
lamb = 0.75       # Intensity of the Poisson process (average number of jumps per unit time)
m = 0.02          # Mean of the jump size distribution (log-normal)
delta = 0.1       # Standard deviation of the jump size distribution (log-normal)
T = 1.0           # Total time (1 year)
N = 1000          # Number of time steps

# Simulate the jump diffusion process
t, S = simulate_jump_diffusion(S0, mu, sigma, lamb, m, delta, T, N)

# Plot the simulated stock prices
plt.figure(figsize=(10, 6))
plt.plot(t, S, label='Jump Diffusion Process')
plt.title('Simulated Stock Price using Jump Diffusion Model')
plt.xlabel('Time (years)')
plt.ylabel('Stock Price')
plt.legend()
plt.grid(True)
plt.show()
```



The jump diffusion model offers a more realistic framework for modeling stock prices by incorporating the possibility of sudden jumps. This enhancement over the classical Black-Scholes model allows for better capturing the empirical characteristics of financial markets, thereby improving the accuracy of option pricing and risk management practices.

Smiles and Smirks Again

As mentioned before, the GARCH, stochastic volatility and jump diffusion models can generate fat-tailed distributions for the asset price $S(T)$. Thus, they can be more nearly consistent with the option smiles discussed in **?@sec-s_smiles** than is the Black-Scholes model (though it appears that one must include jumps in asset prices as well as stochastic volatility in order to duplicate market prices with an option pricing formula). To understand the relation, let σ_{am} denote the implied volatility from an at-the-money call option, i.e., a call option with strike $K = S(0)$. The characteristic of a smile is that implied volatilities from options of the same maturity with strike prices significantly above and below $S(0)$ are higher than σ_{am}.

A strike price higher than $S(0)$ corresponds to an out-of-the money call option. The high implied volatility means that the market is pricing the right to buy at $K > S(0)$ above the Black-Scholes price computed from the volatility σ_{am}; thus, the market must attach a higher probability to stock prices $S(T) > S(0)$ than the volatility σ_{am} would suggest.

A strike price lower than $S(0)$ corresponds to an in-the-money call option. The put option with the same strike is out of the money. The high implied volatility means that the market is pricing call options above the Black-Scholes price computed from the volatility σ_{am}. By put-call parity, the market must also be pricing put options above the Black-Scholes price computed from the volatility σ_{am}. The high prices for the rights to buy and sell at $K < S(0)$ means that the market must attach a higher probability to stock prices $S(T) < S(0)$ than the volatility σ_{am} would suggest. In particular, the high price for the right to sell at $K < S(0)$ means a high insurance premium for owners of the asset who seek to insure their positions, which is consistent with a market view that there is a significant probability of a large loss. This can be interpreted as a crash premium. Indeed, the implied volatilities at strikes less than $S(0)$ are typically higher than the implied volatilities at strikes above $S(0)$ (giving the smile the appearance of a smirk, as discussed in **?@sec-s_smiles**), which is consistent with a larger probability of crashes than of booms (a fatter tail for low returns than for high).

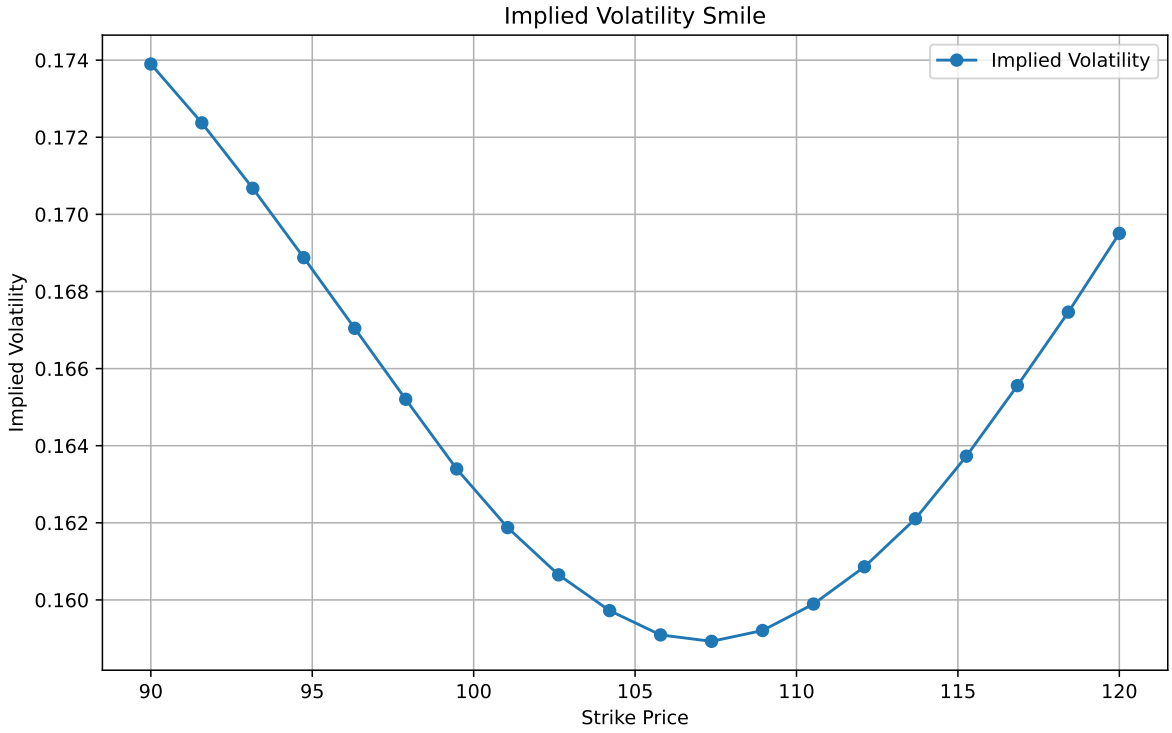
As an example, the following code shows that the stochstic volatility model can generate implied volatility smiles.

This program involes (1) Simulating Heston Model: simulate_heston_paths function generates stock price paths using the Heston model parameters. (2) Calculating call price by discounting the averaged call payoffs across the stock price sample paths (3) Calculating Black Scholes call price : black_scholes_call_price function calculates the call option price using the Black-Scholes formula. (4) Calculating implied volatility: implied_volatility function computes the implied volatility by solving for the volatility that matches the Black-Scholes call price to the simulated call price. (5) Repeating Steps (2)-(4) for different strik prices. (6) Plotting the implied volatility against strike prices fixing the initial stock price at $100.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
from scipy.optimize import brentq

# Heston model parameters
S0 = 100      # Initial stock price
V0 = 0.04     # Initial variance
r = 0.05      # Risk-free rate
q = 0.01      # Dividend yield
T = 1         # Time to maturity (in years)
kappa = 0.25   # Rate of mean reversion of variance
theta = 0.04  # Long-term variance
sigma = 0.5   # Volatility of variance
rho = -0.2    # Correlation between the two Wiener processes
dt = 1/252    # Length of each time period (daily)
N = 252       # Number of time periods (one year)
n_simulations = 10000  # Number of simulations
```

```
def simulate_heston_paths(S0, V0, r, q, T, kappa, theta, sigma, rho, dt, N, n_simulations):
    S = np.zeros((N + 1, n_simulations))
    V = np.zeros((N + 1, n_simulations))
    S[0] = S0
    V[0] = V0

    for t in range(1, N + 1):
        Z1 = np.random.normal(size=n_simulations)
        Z2 = np.random.normal(size=n_simulations)
        W1 = Z1
        W2 = rho * Z1 + np.sqrt(1 - rho**2) * Z2

        V[t] = np.maximum(V[t-1] + kappa * (theta - V[t-1]) * dt + sigma * np.sqrt(V[t-1] * d
        S[t] = S[t-1] * np.exp((r - q - 0.5 * V[t-1]) * dt + np.sqrt(V[t-1] * dt) * W1)

    return S, V

def black_scholes_call_price(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    call_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return call_price

def implied_volatility(C, S, K, T, r):
    def objective_function(sigma):
        return black_scholes_call_price(S, K, T, r, sigma) - C
    try:
        return brentq(objective_function, 0.001, 5.0)
    except ValueError:
        return np.nan

# Simulate paths using the Heston model
S_paths, _ = simulate_heston_paths(S0, V0, r, q, T, kappa, theta, sigma, rho, dt, N, n_simula

# Calculate option prices at different strike prices
strike_prices = np.linspace(90, 120, 20)
call_prices = np.zeros_like(strike_prices)

for i, K in enumerate(strike_prices):
    call_payoffs = np.maximum(S_paths[-1] - K, 0)
    call_prices[i] = np.mean(call_payoffs) * np.exp(-r * T)
```

```python
# Calculate implied volatilities
implied_vols = [implied_volatility(C, S0, K, T, r) for C, K in zip(call_prices, strike_prices

# Filter out NaN values that may occur
valid_indices = ~np.isnan(implied_vols)
strike_prices = strike_prices[valid_indices]
implied_vols = np.array(implied_vols)[valid_indices]

# Plot the implied volatility smile
plt.figure(figsize=(10, 6))
plt.plot(strike_prices, implied_vols, label='Implied Volatility', marker='o')
plt.title('Implied Volatility Smile')
plt.xlabel('Strike Price')
plt.ylabel('Implied Volatility')
plt.legend()
plt.grid(True)
plt.show()
```



Hedging and Market Completeness

The GARCH model is inherently a discrete-time model. If returns have a GARCH structure at one frequency (e.g., monthly), they will not have a GARCH structure at a different frequency (e.g., weekly). Hence, the return period (monthly, weekly, …) is part of the specification of the model. One interpretation of the model is that the dates t_i at which the variance changes are the only dates at which investors can trade. Under this interpretation, it is impossible to perfectly hedge an option: the gross return $S(t_i)/S(t_{i-1})$ over the interval (t_{i-1}, t_i) is lognormally distributed, so no portfolio of the stock and riskless asset formed at t_{i-1} and held over the interval (t_{i-1}, t_i) can perfectly replicate the return of an option over the interval. As discussed in **?@sec-s_incomplete**, we call a market in which some derivatives cannot be perfectly hedged an incomplete market. Thus, the GARCH model is an example of an incomplete market, if investors can only trade at the frequency at which returns have a GARCH structure. However, it is unreasonable to assume that investors can only trade weekly or monthly or even daily.

Another interpretation of the GARCH model is that investors can trade continuously and the asset has a constant volatility within each period (t_{i-1}, t_i). Under this interpretation, the market is complete and options can be delta-hedged. The completeness is a result of the fact that the change $\sigma_{i+1} - \sigma_i$ in the volatility at date t_i (recall that σ_i is the volatility over the period (t_{i-1}, t_i) and σ_{i+1} is the volatility over the period (t_i, t_{i+1})) depends only on $\log S(t_i)$. Thus, the only random factor in the model that needs to be hedged is, as usual, the underlying asset price. However, this interpretation of the model is also a bit strange. Suppose for example that monthly returns are assumed to have a GARCH structure. Then the model states that the volatility in February will be higher if there is an unusually large return (in absolute value) in January. Suppose there is an unusually large return in the first half of January. Then, intuitively, one would expect the change in the volatility to occur in the second half of January rather than being delayed until February. However, the model specifies that the volatility is constant during each month, hence constant during January in this example.

The stochastic volatility model is more straightforward. The market is definitely incomplete. The value of a call option at date $t < T$, where T is the maturity of the option, will depend on the underlying asset price $S(t)$ and the volatility $\sigma(t)$. Denoting the value by $C(t, S(t), \sigma(t))$, we have from Ito's formula that

$$\mathrm{d}C(t) = \text{something}\,\mathrm{d}t + \frac{\partial C}{\partial S}\,\mathrm{d}S(t) + \frac{\partial C}{\partial \sigma}\,\mathrm{d}\sigma(t)\ .$$

A replicating portfolio must have the same dollar change at each date t. If we hold $\partial C/\partial S$ shares of the underlying asset, then the change in the value of the shares will be $(\partial C/\partial S)\,\mathrm{d}S$. However, there is no way to match the
$(\partial C/\partial \sigma)\,\mathrm{d}\sigma$ term using the underlying asset and the riskless asset.

The significance of the market being incomplete is that the value of a derivative asset that cannot be replicated using traded assets (e.g., the underlying and riskless assets) is not uniquely

determined by arbitrage considerations. As discussed in **?@sec-s_incomplete**, one must use equilibrium pricing in this circumstance. That is what we have implicitly done in this chapter. By assuming particular dynamics for the volatility under the risk-neutral measure, we have implicitly selected a particular risk-neutral measure from the set of risk-neutral measures that are consistent with the absence of arbitrage.

Exercises

Exercise 0.1. The purpose of this exercise is to generate a fat-tailed distribution from a model that is simpler than the GARCH and stochastic volatility models but has somewhat the same flavor. The distribution will be a mixture of normals. Create a python program in which the user can input S, r, q, T, σ_1 and σ_2. Use these inputs to produce a column of 500 simulated $\log S(T)$. In each simulation, define $\log S(T)$ as

$$\log S(T) = \log S(0) + \left(r - q - \frac{1}{2}\sigma^2\right) T + \sigma\sqrt{T}z \ ,$$

where z is a standard normal, $\sigma = x\sigma_1 + (1-x)\sigma_2$, and x is a random variable that equals zero or one with equal probabilities.

Calculate the mean and standard deviation of the $\log S(T)$ and calculate the fraction that lie more than two standard deviations below the mean. If the $\log S(T)$ all came from a normal distribution with the same variance, then this fraction should equal $\mathrm{N}(-2) = 2.275\%$. If the fraction is higher, then the distribution is fat tailed. (Of course, the actual fraction would differ from 2.275% in any particular case due to the randomness of the simulation, even if all of the $\log S(T)$ came from a normal distribution with the same variance).

Exercise 0.2. Create a python program prompting the user to input the same inputs as in the `simulating_garch` function except for the initial volatility and θ. Simulate 500 paths of a GARCH process and output $\log S(T)$ for each simulation (you don't need to output the entire paths as in the `simulating_garch` function). Take the initial volatility to be 0.3 and $\theta = 0.09$. Determine whether the distribution is fat-tailed by computing the fraction of the $\log S(T)$ that lie two or more standard deviations below the mean, as in the previous exercise. For what values of κ and λ does the distribution appear to be especially fat-tailed?

Exercise 0.3. Repeat Exercise 0.2 for the Heston stochastic volatility model, describing the values of κ, γ and ρ that appear to generate especially fat-tailed distributions.