Step 1 首先嘗試理解 Hull-White Model

The Hull-White Model

- The Hull-White model is the following special case,

Short rate

$$dr = (\theta(t) - ar)\,dt + \sigma\,dW.$$

- When the current term structure is matched,

$$\theta(t) = \frac{\partial f(0,t)}{\partial t} + af(0,t) + \frac{\sigma^2}{2a}\left(1 - e^{-2at}\right).$$

– Recall that $f(0,t)$ defines the forward rate curve.

```
In [1]: import QuantLib as ql
        import matplotlib.pyplot as plt
        import numpy as np
        % matplotlib inline
```

插入所需套件

```
UsageError: Line magic function `%` not found.
```

```
In [3]: sigma = 0.1
        a = 0.1
        timestep = 360
        length = 30 # in years
        forward_rate = 0.05
        day_count = ql.Thirty360()
        todays_date = ql.Date(15, 1, 2015)
```

設定參數(注意這裡日期要改➔可能採用手動輸入或 import time)

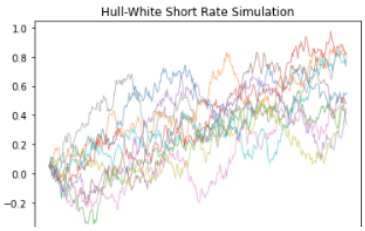
```
In [4]: ql.Settings.instance().evaluationDate = todays_date

        spot_curve = ql.FlatForward(todays_date, ql.QuoteHandle(ql.SimpleQuote(forward_rate)), day_count)
        spot_curve_handle = ql.YieldTermStructureHandle(spot_curve)
```

```
In [5]: hw_process = ql.HullWhiteProcess(spot_curve_handle, a, sigma)
        rng = ql.GaussianRandomSequenceGenerator(ql.UniformRandomSequenceGenerator(timestep, ql.UniformRandomGenerator()))
        seq = ql.GaussianPathGenerator(hw_process, length, timestep, rng, False)
```

```
In [6]: def generate_paths(num_paths, timestep):
            arr = np.zeros((num_paths, timestep+1))
            for i in range(num_paths):
                sample_path = seq.next()
                path = sample_path.value()
                time = [path.time(j) for j in range(len(path))]
                value = [path[j] for j in range(len(path))]
                arr[i, :] = np.array(value)
            return np.array(time), arr
```

特別注意到這邊 num_paths 要設定

```
In [7]: num_paths = 10
        time, paths = generate_paths(num_paths, timestep)
        for i in range(num_paths):
            plt.plot(time, paths[i, :], lw=0.8, alpha=0.6)
        plt.title("Hull-White Short Rate Simulation")
        plt.show()
```

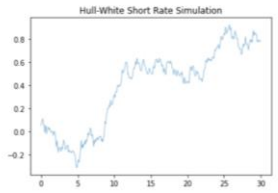


Step 1-2 透過修改程式參數逐步理解個參數的意義

num_paths

藉由更改參數得出 num_paths 為生成的路徑數量

In [16]:
```
num_paths = 1
time, paths = generate_paths(num_paths, timestep)
for i in range(num_paths):
    plt.plot(time, paths[i, :], lw=0.8, alpha=0.6)
plt.title("Hull-White Short Rate Simulation")
plt.show()
```

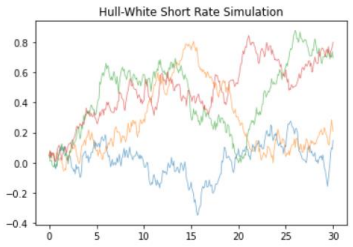


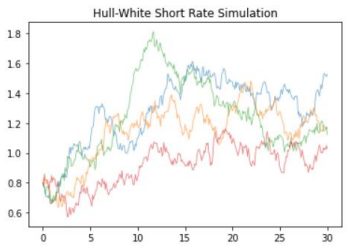
In [17]:
```
num_paths = 2
time, paths = generate_paths(num_paths, timestep)
for i in range(num_paths):
    plt.plot(time, paths[i, :], lw=0.8, alpha=0.6)
plt.title("Hull-White Short Rate Simulation")
plt.show()
```

In [18]:
```
num_paths = 4
time, paths = generate_paths(num_paths, timestep)
for i in range(num_paths):
    plt.plot(time, paths[i, :], lw=0.8, alpha=0.6)
plt.title("Hull-White Short Rate Simulation")
plt.show()
```

forward_rate

基本上就是 forward rate,值得注意的是每個 path 的起始點都設定的 forward rate 出發(觀察 Y 軸)





length

決定 X 軸長度

timestep

決定經過幾次波動

sigma

決定各線之間差異程度

a

決定波動率

Step 2 測試完之後寫入參數設定功能

此處一併導入時間套件，避免每次都需自己輸入時間

In [2]:
```
#導入套件
import QuantLib as ql
import matplotlib.pyplot as plt
import numpy as np
import time
import datetime

#輸入參數
sigma = float(input("請輸入參數sigma= "))
a = float(input("請輸入參數a= "))
timestep = int(input("請輸入timestep= "))
length = float(input("請輸入length(年數)= "))
forward_rate = float(input("請輸入forward rate= "))
day_count = ql.Thirty360()
time = time.localtime( time.time() )
todays_date = ql.Date(time.tm_mday, time.tm_mon, time.tm_year)
num_paths = int(input("請輸入num_paths= "))

#導入計算模型
ql.Settings.instance().evaluationDate = todays_date
spot_curve = ql.FlatForward(todays_date, ql.QuoteHandle(ql.SimpleQuote(forward_rate)), day_count)
spot_curve_handle = ql.YieldTermStructureHandle(spot_curve)
hw_process = ql.HullWhiteProcess(spot_curve_handle, a, sigma)
rng = ql.GaussianRandomSequenceGenerator(ql.UniformRandomSequenceGenerator(timestep, ql.UniformRandomGenerator()))
seq = ql.GaussianPathGenerator(hw_process, length, timestep, rng, False)

def generate_paths(num_paths, timestep):
    arr = np.zeros((num_paths, timestep+1))
    for i in range(num_paths):
        sample_path = seq.next()
        path = sample_path.value()
        time = [path.time(j) for j in range(len(path))]
        value = [path[j] for j in range(len(path))]
        arr[i, :] = np.array(value)
    return np.array(time), arr

#num_paths = 4
time, paths = generate_paths(num_paths, timestep)
#print(paths) ==> short rate
for i in range(num_paths):
    plt.plot(time, paths[i, :], lw=0.8, alpha=0.6)
plt.title("Hull-White Short Rate Simulation")
plt.show()
```

```
請輸入參數sigma= 0.1
請輸入參數a= 0.1
請輸入timestep= 360
請輸入length(年數)= 30
請輸入forward rate= 0.05
請輸入num_paths= 10
```

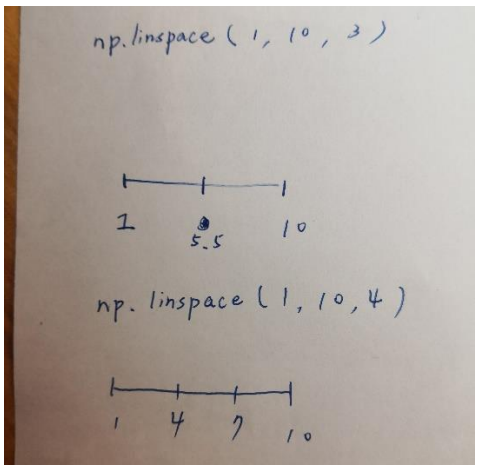
Step 3 再次觀察和幾何布朗運動

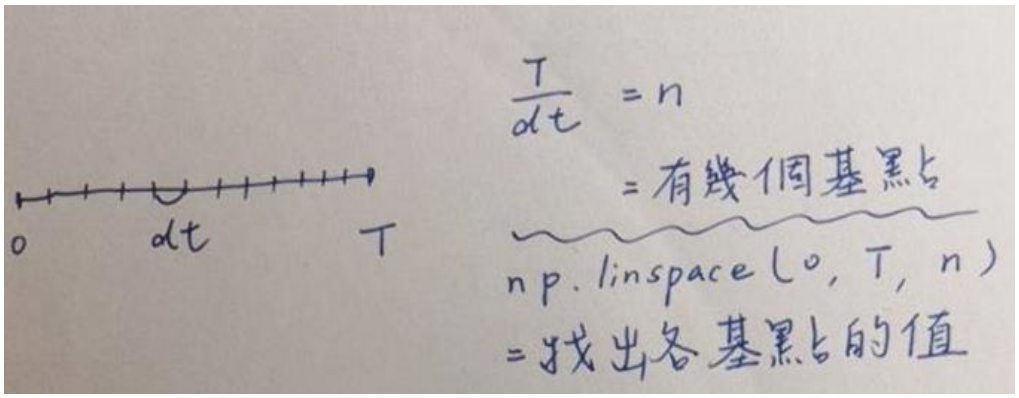
$$S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right)$$

```
def genBrownPath (T, mu, sigma, S0, dt):

    n = round(T/dt)
    t = np.linspace(0, T, n)
    W = [0] + np.random.standard_normal(size = n)
    W = np.cumsum(W)*np.sqrt(dt) # == standard brownian motion
    X = (mu-0.5*sigma**2)*t + sigma*W
    S = S0*np.exp(X) # == geometric brownian motion
    plt.plot(t, S)
    return S
```

X 與 S 都可以較輕鬆的從公式上理解，這邊比較麻煩的是 W 與 t 是怎麼來的，查了維基百科(與註解)後推測 W 應是利用傅立葉級數推導出布朗運動，其中 np.cumsum()是用來累加陣列中的數值，是一個非常有趣的功能。

另外 t 的部分就比較容易理解了 np.linspace(a, b, c) 是用來創造 a 到 b 之間(包含 a 與 b)總共有 c 個基點，每個基點的值是多少





幾何布朗運動的缺陷

在真實股票價格中波動隨時間變化 (possibly stochastically)，但是在幾何布朗運動中，波動是不隨時間變化的。

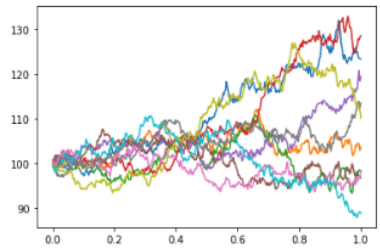
在真實股票價格中，收益通常不服從常態分布 (真實股票收益有更高的 峰度 ('fatter tails')，代表了有可能形成更大的價格波動)。

故導入 Hull-White Model 建立短期利率變動➔這邊就是把 mu 代換成模擬出來的短期利率(paths)

Step 4 寫入參數設定功能

In [6]:
```
#模擬股價
import matplotlib.pyplot as plt
import numpy as np

# State Variables
#T = 10
#N = 10 # Path
#dt = 0.01
#mu = 0.01*dt # Rendite pro Zeiteinheit
#sigma = 0.1
#S0 = 100
#np.random.seed(1)

T = length
N = num_paths
dt = 1/timestep
mu = paths
sigma = float(input("請輸入股價每年波動度(volatility)= "))
S0 = float(input("請輸入初始股價= "))
np.random.seed(1)

def genBrownPath (T, mu, sigma, S0, dt):

    n = round(T/dt)
    t = np.linspace(0, T, n + 1 )
    W = [0] + np.random.standard_normal(size = n+1)
    W = np.cumsum(W)*np.sqrt(dt) # == standard brownian motion
    X = (mu-0.5*sigma**2)*t + sigma*W
    S = S0*np.exp(X) # == geometric brownian motion
    plt.plot(t, S)
    return S

Paths = []
for i in range(0,num_paths):
    Paths.append(genBrownPath(T, mu[i], sigma, S0, dt))
plt.show()
```

```
請輸入股價每年波動度(volatility)= 0.1
請輸入初始股價= 100
```



Step 5 計算選擇權價格

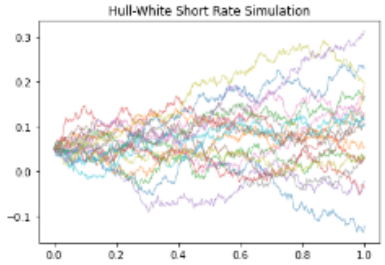
In [7]:
```
#輸入基本資訊
K = float(input('請輸入strike price = '))
R = float(input('請輸入risk free rate = '))

#計算各個路徑的 Option Price 取平均再折現
Call = []
Put = []
for i in range(0, num_paths):
    Call.append(max(0, Paths[i][-1] -K) )
    Put.append(max(0, K - Paths[i][-1]) )

#計算折現值
Call_price = round(np.mean(Call)*np.exp(-R*T), 4)
Put_price = round(np.mean(Put)*np.exp(-R*T), 4)

print("Call Price = " + str(Call_price))
print("Put Price = " + str(Put_price))
```

```
請輸入strike price = 120
請輸入risk free rate = 0.1
Call Price = 1.0799
Put Price = 11.8595
```

實作結果:

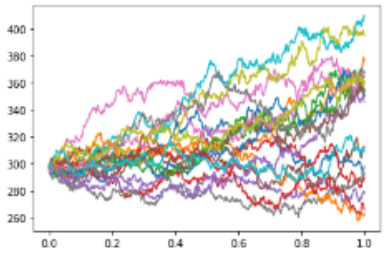
In [8]:
```
#導入套件
import QuantLib as ql
import matplotlib.pyplot as plt
import numpy as np
import time
import datetime

#輸入參數
sigma = float(input("請輸入參數sigma= "))
a = float(input("請輸入參數a= "))
timestep = int(input("請輸入timestep= "))
length = float(input("請輸入length(年數)= "))
forward_rate = float(input("請輸入forward rate= "))
day_count = ql.Thirty360()
time = time.localtime( time.time() )
todays_date = ql.Date(time.tm_mday, time.tm_mon, time.tm_year)
num_paths = int(input("請輸入num_paths= "))

#導入計算模型
ql.Settings.instance().evaluationDate = todays_date
spot_curve = ql.FlatForward(todays_date, ql.QuoteHandle(ql.SimpleQuote(forward_rate)), day
spot_curve_handle = ql.YieldTermStructureHandle(spot_curve)
hw_process = ql.HullWhiteProcess(spot_curve_handle, a, sigma)
rng = ql.GaussianRandomSequenceGenerator(ql.UniformRandomSequenceGenerator(timestep, ql.Ur
seq = ql.GaussianPathGenerator(hw_process, length, timestep, rng, False)

def generate_paths(num_paths, timestep):
    arr = np.zeros((num_paths, timestep+1))
    for i in range(num_paths):
        sample_path = seq.next()
        path = sample_path.value()
        time = [path.time(j) for j in range(len(path))]
        value = [path[j] for j in range(len(path))]
        arr[i, :] = np.array(value)
    return np.array(time), arr

#num_paths = 4
time, paths = generate_paths(num_paths, timestep)
#print(paths) ==> short rate
for i in range(num_paths):
    plt.plot(time, paths[i, :], lw=0.8, alpha=0.6)
plt.title("Hull-White Short Rate Simulation")
plt.show()
```

```
請輸入參數sigma= 0.1
請輸入參數a= 0.1
請輸入timestep= 500
請輸入length(年數)= 1
請輸入forward rate= 0.05
請輸入num_paths= 20
```



In [9]:
```
#模擬股價
import matplotlib.pyplot as plt
import numpy as np

# State Variables
#T = 10
#N = 10 # Path
#dt = 0.01
#mu = 0.01*dt # Rendite pro Zeiteinheit
#sigma = 0.1
#S0 = 100
#np.random.seed(1)

T = length
N = num_paths
dt = 1/timestep
mu = paths
sigma = float(input("請輸入股價每年波動度(volatility)= "))
S0 = float(input("請輸入初始股價= "))
np.random.seed(1)

def genBrownPath (T, mu, sigma, S0, dt):

    n = round(T/dt)
    t = np.linspace(0, T, n + 1 )
    W = [0] + np.random.standard_normal(size = n+1)
    W = np.cumsum(W)*np.sqrt(dt) # == standard brownian motion
    X = (mu-0.5*sigma**2)*t + sigma*W
    S = S0*np.exp(X) # == geometric brownian motion
    plt.plot(t, S)
    return S

Paths = []
for i in range(0,num_paths):
    Paths.append(genBrownPath(T, mu[i], sigma, S0, dt))
plt.show()
```

```
請輸入股價每年波動度(volatility)= 0.1
請輸入初始股價= 297
```



In [10]:
```
#輸入基本資訊
K = float(input('請輸入strike price = '))
R = float(input('請輸入risk free rate = '))

#計算各個路徑的 Option Price 取平均再折現
Call = []
Put = []
for i in range(0, num_paths):
    Call.append(max(0, Paths[i][-1] -K) )
    Put.append(max(0, K - Paths[i][-1]) )

#計算折現值
Call_price = round(np.mean(Call)*np.exp(-R*T), 4)
Put_price = round(np.mean(Put)*np.exp(-R*T), 4)

print("Call Price = " + str(Call_price))
print("Put Price = " + str(Put_price))
```

```
請輸入strike price = 320
請輸入risk free rate = 0.1
Call Price = 25.3772
Put Price = 11.2541
```