In [1]:

```
from black_scholes_monte_carlo import Simulator
import yfinance as yf
import numpy as np
import pandas as pd
```

Markovian Switching Monte Carlo for Optimal Liquidity Pool Range Analsyis

In [2]:

```
# close_data = eth_data['Close']['ETH-USD']

# initial_price = close_data['2025-04-24']
# price_data = close_data.values
# price_data
sim = Simulator(0,0,0,1)
# data = sim.get_security_data('ETH-USD', start='2020-04-24', end='2025-04-24')
data = pd.read_csv("data/Ethereum Historical Results Price Data.csv")
to_float  = lambda x: float(x.replace(',', ''))
data['Open'] = data['Open'].apply(to_float)
data['Returns'] = np.log(data['Open'] / data['Open'].shift(1))
data = data.dropna()

sim.data = data
initial_price = data['Open'].iloc[0]
```

In [3]:

```
sim.set_initial_price(initial_price)
```

In [4]:

```
# data[data['Returns'].isna()]
data.head()
```

Out[4]:

	Date	Price	Open	High	Low	Vol.	Change %	Returns
1	18/05/2025	2,498.97	2475.04	2,585.12	2,340.94	905.26K	0.97%	-0.009550
2	17/05/2025	2,475.04	2537.14	2,537.14	2,449.57	566.93K	-2.44%	0.024781
3	16/05/2025	2,537.04	2545.46	2,646.74	2,531.45	680.35K	-0.44%	0.003274
4	15/05/2025	2,548.16	2610.78	2,644.57	2,480.73	862.28K	-2.36%	0.025338
5	14/05/2025	2,609.68	2680.04	2,719.25	2,549.01	928.09K	-2.63%	0.026183

In [5]:

```
model = sim.markov_switching_model(data, num_regimes=2)
```

```
c:\Users\ab\Desktop\Durham\AFT\UNISWAP analysis\Optimal-Liquidity-Pool-Range\uniswap
_analysis\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: An
unsupported index was provided. As a result, forecasts cannot be generated. To use t
he model for forecasting, use one of the supported classes of index.
  self._init_dates(dates, freq)
```

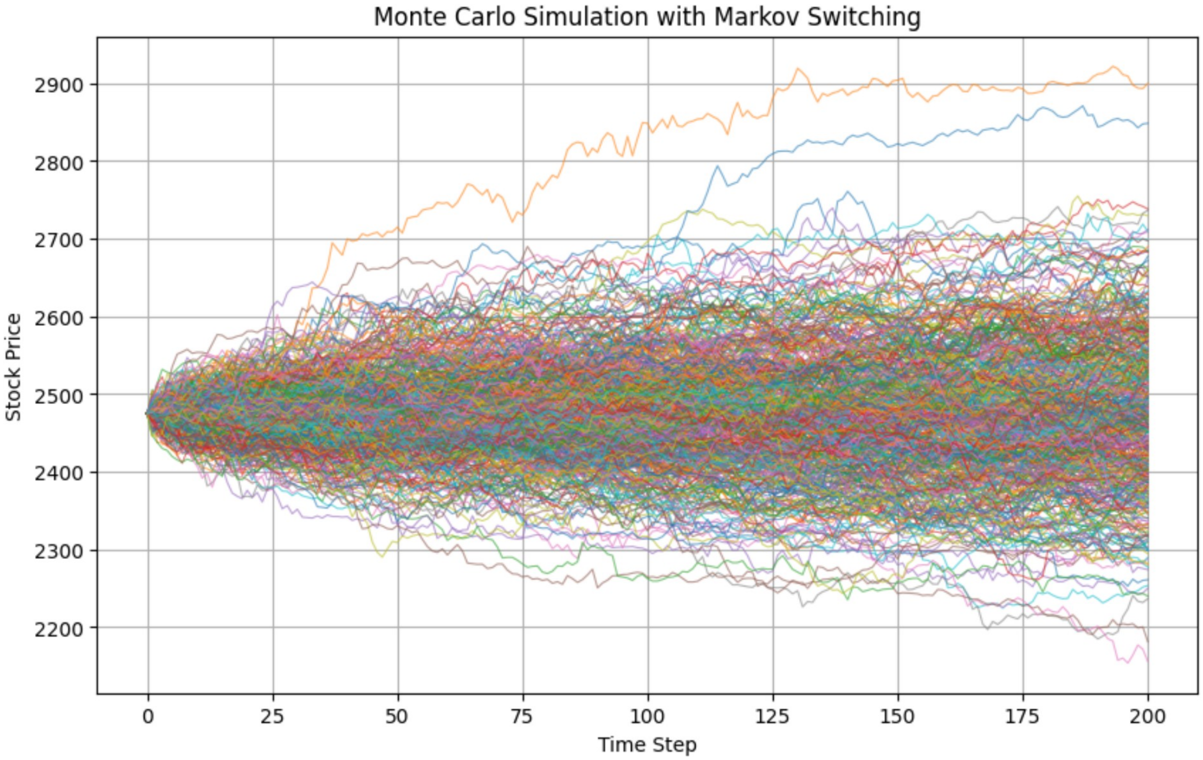
In [6]:

```
year_sim = sim.markov_switching_monte_carlo(200,500,model)
```

```
c:\Users\ab\Desktop\Durham\AFT\UNISWAP analysis\Optimal-Liquidity-Pool-Range\uniswap
_analysis\black_scholes_monte_carlo.py:108: FutureWarning: Series.__getitem__ treati
ng keys as positions is deprecated. In a future version, integer keys will always be
treated as labels (consistent with DataFrame behavior). To access a value by positio
n, use `ser.iloc[pos]`
  if np.isnan(S[i, t-1]) or np.isnan(mu[reg]) or np.isnan(sigma[reg]):
c:\Users\ab\Desktop\Durham\AFT\UNISWAP analysis\Optimal-Liquidity-Pool-Range\uniswap
_analysis\black_scholes_monte_carlo.py:112: FutureWarning: Series.__getitem__ treati
ng keys as positions is deprecated. In a future version, integer keys will always be
treated as labels (consistent with DataFrame behavior). To access a value by positio
n, use `ser.iloc[pos]`
  S[i, t] = S[i, t-1] * np.exp((mu[reg] - 0.5 * sigma[reg]**2) * dt + sigma[reg] * n
p.sqrt(dt) * z)
```

In [7]:

```
sim.plot_paths(year_sim)
```



In [8]:

```
optimal_range = sim.find_optimal_pool_range(year_sim)[:2]
apy = sim.estimate_apy_for_range(year_sim,optimal_range[0], optimal_range[1])
print(f"Optimal range 1 year: {optimal_range}")
print(f"optimal range apy: {apy}")
```

```
Optimal range 1 year: (np.float64(1982.4701133106091), np.float64(2973.705169965913
2))
optimal range apy: 6.027938015745954
```