Project 9: Pricing Mortgage Backed Securities

By - Ameya Shete

In [364]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize
import warnings
warnings.filterwarnings("ignore")
```

In [269]:
```
# Inputs for CIR model
r_mean, r0, sigma, kappa, T = 0.08, 0.078, 0.12, 0.6, 30
notional, wac = 1e5, 0.08
np.random.seed(100)
```

In [270]:
```
# In the CIR function, we want to predict the daily interest rates. Wh
# interest rates and will sum up the interest rates for each month. We
# for which we will sum up the interest rates from t-1 to t-1+10
def CIR(r_mean, r0, sigma, kappa, T):
    delta_t = 1/252
    N, n = 1000, int(T/delta_t)
    z = np.random.normal(0, 1, size=(N, n))
    r = np.zeros((N, n+1))
    r[:, 0] = r0
    for i in range(n):
        r[:, i+1] = r[:, i] + kappa*(r_mean - r[:, i])*delta_t + sigma
    r = r[:, 1:]
    return r
```

In [271]:
```
def CPR(pv0, pv_t_minus_1, R, t, r_10):
    sy_array = [0.94,0.76,0.74,0.95,0.98,0.92,0.98,1.10,1.18,1.22,1.23
    sy_t = sy_array[int((t%12)-1)]

    ri_t = 0.28 + 0.14*np.arctan(-8.57 + 430*(R - r_10))

    bu_t = 0.3 + 0.7*(pv_t_minus_1/pv0)

    sg_t = min(1, t/30)

    cpr_t = sy_t * ri_t * bu_t * sg_t
    return cpr_t
```

In [289]:
```
def calculate_cash_flow(pv_t_minus_1, r, N, pv0, R, t, r_mean, sigma,
    mp_t = (pv_t_minus_1*r)/(1 - (1 + r)**(-N + (t - 1)))
    spp_t = pv_t_minus_1*r*(1/(1 - (1 + r)**(-N + (t - 1))) -  1)
    pp_t = (pv_t_minus_1 - spp_t)*(1 - (1 - CPR(pv0, pv_t_minus_1, R,
    return mp_t + pp_t
```

In [290]:
```
def ZCB_price(rt, r_mean, sigma, kappa, T, t):

    h1 = np.sqrt(kappa**2 + 2*sigma**2)
    h2 = (kappa + h1)/2
    h3 = (2*kappa*r_mean)/sigma**2

    A = ((h1 * np.exp(h2*(T - t)))/(h2 * (np.exp(h1*(T - t)) - 1) + h1
    B = (np.exp(h1*(T - t)) - 1)/(h2 * (np.exp(h1*(T - t)) - 1) + h1)

    return A*np.exp(-B*rt)
```

In [291]:
```
def scheduled_payment(pv_t_minus_1, r, N, t):
    return (pv_t_minus_1*r)/(1 - (1 + r)**(-N+t-1))

def prepayment(pv_t_minus_1, sp_t, pv0, R, t, r_10):
    return (pv_t_minus_1 - sp_t)*(1 - (1 - CPR(pv0, pv_t_minus_1, R, t
```

In [382]:
```
def MBS_pricing(r_mean, r0, sigma, kappa, T, notional, wac, simulated_
    N, num_of_months = 1000, T*12
    delta_t = 1/252
    one_month_trading_days = 21
    total_days = T*12*one_month_trading_days
    total_months = int(total_days/one_month_trading_days)
    pv0 = notional
    pv_t_minus_1 = [pv0]*N
    r = wac/12
    summation_array, io_sum, tpp_sum = [0]*N, [0]*N, [0]*N
    for i in range(1, total_months+1):
        # Interest rates for all simulations at t-1
        new_r0 = simulated_r[:, i*one_month_trading_days-1]
        # 10 year interest rate at t-1
        r_10 = [ZCB_price(r0, r_mean, sigma, kappa, T, i) for j in new

        simulated_r_until_t = simulated_r[:, :i*one_month_trading_days
        r_t = np.sum(simulated_r_until_t + OAS, axis=1)*delta_t

        cash_flows = [calculate_cash_flow(pv_t_minus_1[j], r, num_of_m
                                          T, r_10[j]) for j in range

        summation_array = np.sum([summation_array, np.exp(-r_t) * cash

        ip_t = np.multiply(pv_t_minus_1, r)
        io_sum = np.sum([io_sum, np.exp(-r_t)*ip_t], axis=0)

        tpp_t = np.subtract(cash_flows, np.multiply(r, pv_t_minus_1))
        tpp_sum = np.sum([tpp_sum, np.exp(-r_t)*tpp_t], axis=0)

        pv_t_minus_1 = np.subtract(pv_t_minus_1, tpp_t)

    return np.mean(summation_array), np.mean(io_sum), np.mean(tpp_sum)
```

In [375]:
```
r = CIR(r_mean, r0, sigma, kappa, T)
```

Question 1

a)

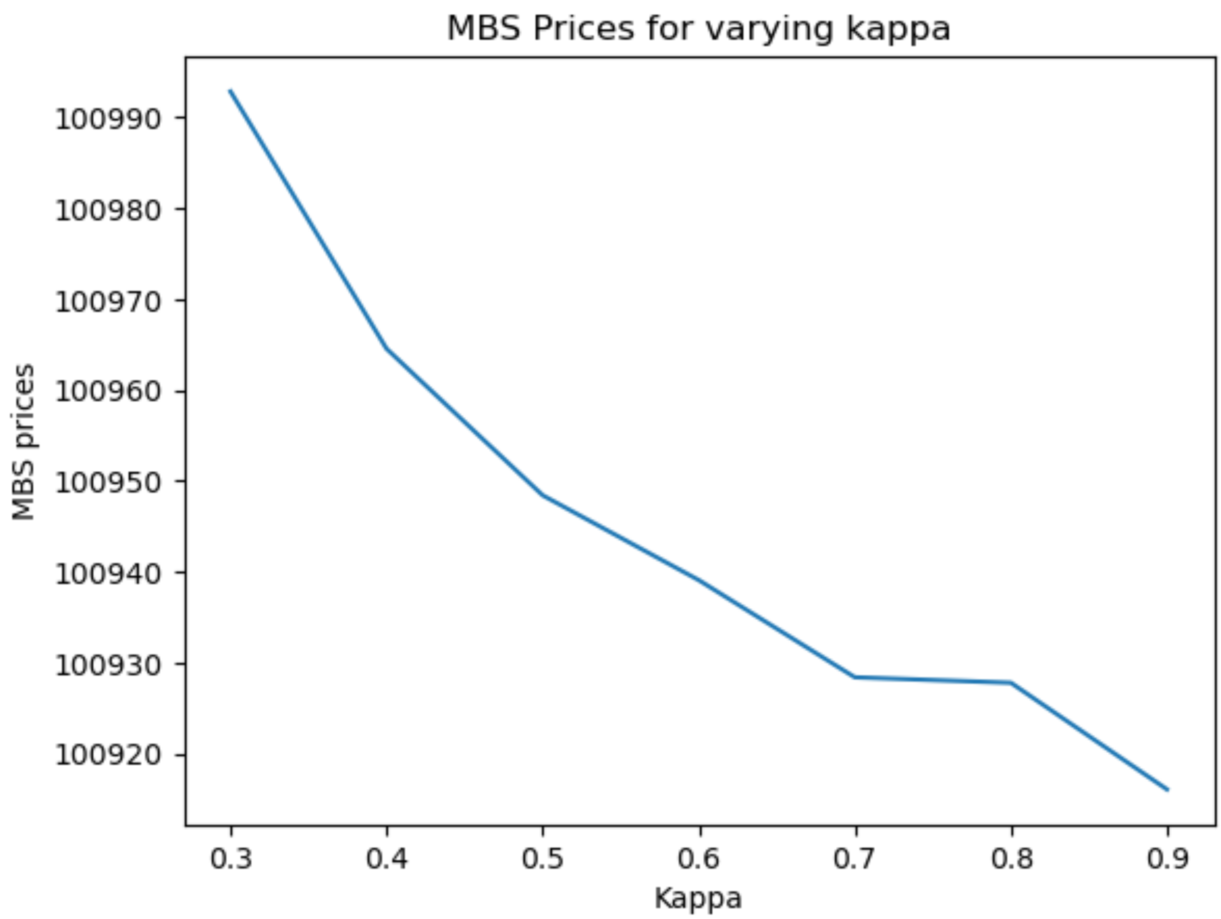
In [354]:
```
x = 0
price = MBS_pricing(r_mean, r0, sigma, kappa, T, notional, wac, r, 0)
print("The price of the MBS is: $", price)
```

```
The price of the MBS is: $ 100585.29356904737
```

b)

In [341]:
```
kappa_array = np.arange(0.3, 1, 0.1)
mbs_prices = [MBS_pricing(r_mean, r0, sigma, i, T, notional, wac, r, 0
```

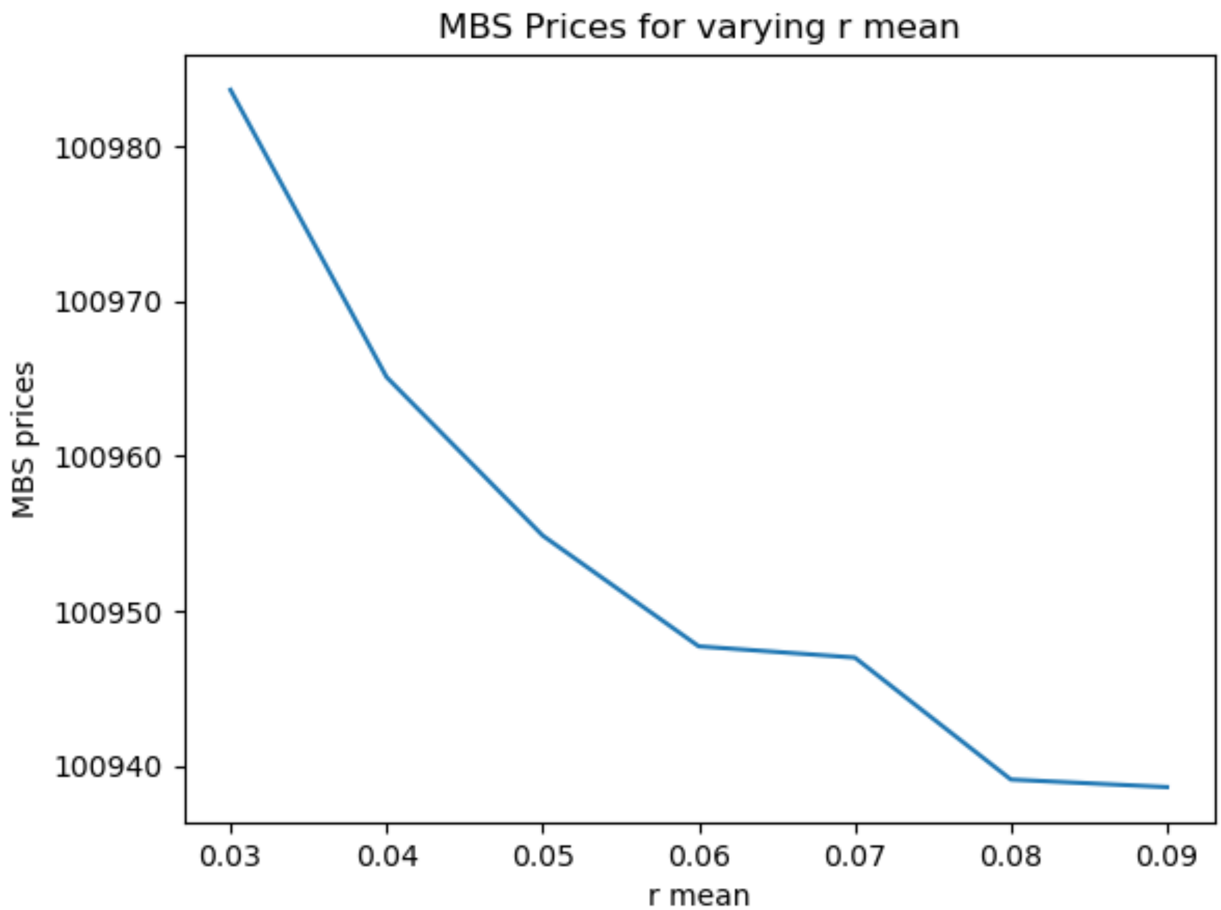
In [342]:
```
fig, ax = plt.subplots()
ax.plot(kappa_array, mbs_prices)
ax.set_xlabel("Kappa")
ax.set_ylabel("MBS prices")
ax.set_title("MBS Prices for varying kappa")
ax.ticklabel_format(useOffset=False)
```



c)

In [343]:
```
r_mean_array = np.arange(0.03, 0.09+0.01, 0.01)
mbs_prices = [MBS_pricing(i, r0, sigma, kappa, T, notional, wac, r, 0)
```

In [344]:
```
fig, ax = plt.subplots()
ax.plot(r_mean_array, mbs_prices)
ax.set_xlabel("r mean")
ax.set_ylabel("MBS prices")
ax.set_title("MBS Prices for varying r mean")
ax.ticklabel_format(useOffset=False)
```



Question 2

In [362]:
```
mbs_market_price = 102000
def optimize(OAS, mbs_market_price):
    my_price = MBS_pricing(r_mean, r0, sigma, kappa, T, notional, wac,
    diff = my_price - mbs_market_price
    return diff

def optimal_OAS(mbs_market_price):
    OAS = scipy.optimize.newton(optimize, 0, args = (mbs_market_price,
    return OAS
```

In [365]:
```
option_adjusted_spread = optimal_OAS(mbs_market_price)
print("The OAS is: ", option_adjusted_spread)
```

```
The OAS is:  -0.003324417967554995
```

Question 3

In [366]:
```
def duration(option_adjusted_spread, mbs_market_price, y):
    P_minus = MBS_pricing(r_mean, r0, sigma, kappa, T, notional, wac,
    P_plus = MBS_pricing(r_mean, r0, sigma, kappa, T, notional, wac, r
    duration = (P_minus - P_plus)/(2 * y * mbs_market_price)
    return duration
```

In [368]:
```
mbs_duration = duration(option_adjusted_spread, mbs_market_price, 0.00
print("The MBS duration is: ", mbs_duration)
```

```
The MBS duration is:  4.215849936401498
```

In [369]:
```
def convexity(option_adjusted_spread, mbs_market_price, y):
    P_minus = MBS_pricing(r_mean, r0, sigma, kappa, T, notional, wac,
    P_plus = MBS_pricing(r_mean, r0, sigma, kappa, T, notional, wac, r
    convexity = (P_plus + P_minus - (2 * mbs_market_price))/(2 * mbs_m
    return convexity
```

In [370]:
```
mbs_convexity = convexity(option_adjusted_spread, mbs_market_price, 0.
print("The MBS convexity is: ", mbs_convexity)
```

```
The MBS convexity is:  13.301101267821723
```

Question 4

In [386]:
```
io_price = MBS_pricing(r_mean, r0, sigma, kappa, T, notional, wac, r,
print("The IO price is:", io_price)
```

```
The IO price is: 33385.524902913516
```

In [387]:
```
po_price = MBS_pricing(r_mean, r0, sigma, kappa, T, notional, wac, r,
print("The PO price is:", po_price)
```

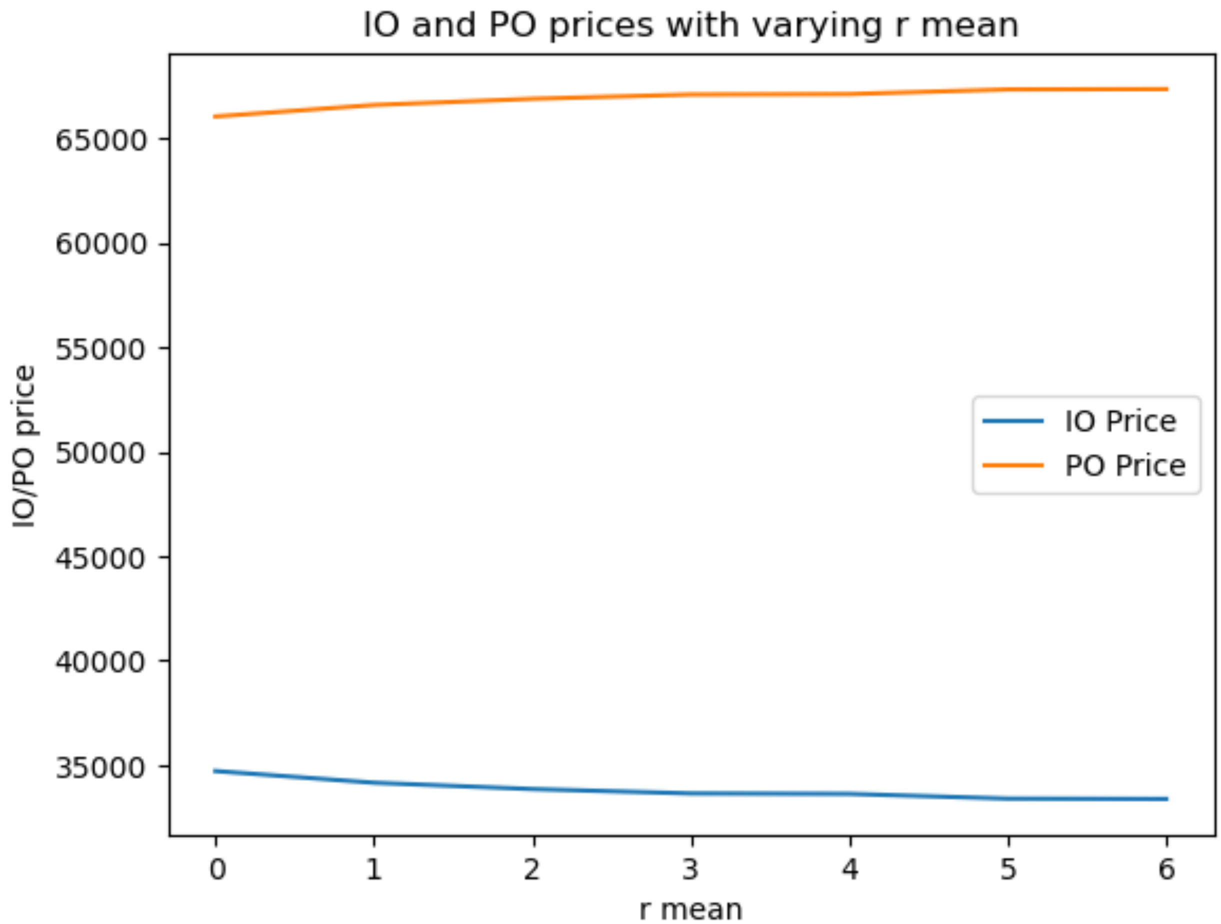
```
The PO price is: 67356.8904296695
```

In [389]:

```
r_mean_array = np.arange(0.03, 0.09+0.01, 0.01)
io_price_array = [MBS_pricing(i, r0, sigma, kappa, T, notional, wac, r
po_price_array = [MBS_pricing(i, r0, sigma, kappa, T, notional, wac, r
```

In [393]:

```
plt.plot(io_price_array, label='IO Price')
plt.plot(po_price_array, label='PO Price')
plt.title("IO and PO prices with varying r mean")
plt.xlabel("r mean")
plt.ylabel("IO/PO price")
plt.legend()
plt.show()
```



In [ ]: