Fixed Income Derivatives Final Exam Fall 2023(19.01.2024) - SOLUTION

Problem 1

Your job is to compute the price of a complex derivative and to do that, you have decided to fit the Cox-Ingersoll-Ross model to a set of continuously compounded zero coupon spot rates that your colleague has extracted from market data and which are given in the table below. The spot rates are denoted by $R(0,T)$ and the corresponding maturities by T.

T	0.1	0.25	0.5	0.75	1	1.5	2	3	4	5	7	10
$R(0,T)$	0.0334	0.0352	0.0375	0.0392	0.0405	0.0422	0.0433	0.0445	0.0451	0.0455	0.0459	0.0462

In the CIR model, the dynamics of the short rate r_t under the risk neutral pricing measure $\mathbb{Q}$ are

$$\begin{aligned} dr_t &= a(b - r_t)dt + \sigma\sqrt{r_t}dW_t, \quad t > 0 \\ r(t=0) &= r_0 \end{aligned} \tag{1}$$

From years of experience, you know that $\sigma = 0.08$, so you will *not* need to estimate σ.

a) Using initial values $\tilde{r}_0 = 0.025$, $\tilde{a} = 1.5$, $\tilde{b} = 0.07$, fit a CIR model to the data from your colleague and estimate r_0, a and b.

   i) Report the estimates $\hat{r}_0$, $\hat{a}$ and $\hat{b}$.

   ii) Demonstrate that a CIR model with your estimated parameter values fits the data.

Next, you will simulate the short rate in the CIR model using the parameter estimates you have just found and a step in time of size δ. Denote by M, the number of steps in your simulation and index the time points in your simulation by m, $m \in \{0, 1, 2, ..., M-1, M\}$ so that the time points will be $\left[t_0, t_1, ..., t_{M-1}, t_M\right] = \left[0, \delta, 2\delta, ..., T-\delta, T = \delta M\right]$ and hence $\delta = \frac{T}{M}$. The scheme you will need to implement is a simple Euler first-order scheme of the form

$$r_m = r_{m-1} + a(b - r_{m-1})\delta + \sigma\sqrt{\delta}\sqrt{r_{m-1}}Z_m, \quad m \in \{1, 2, ..., M\} \tag{2}$$

where $Z_m \sim N(0,1)$, $m \in \{1, ..., M\}$ and all the standard normal random variables are independent. Set the parameters of the model you are simulating to the values you found when fitting the CIR model in a) above. That is, set $r_0 = \hat{r}_0$, $a = \hat{a}$, $b = \hat{b}$ and $\sigma = 0.08$. (If you were not able to fit the CIR model, you can use the initial values $\tilde{r}_0$, $\tilde{a}$ and $\tilde{b}$ from the fit).

b) Now set $t_0 = 0$, $T = 10$ and $M = 10000$ and construct a *single* trajectory for the short rate.

   i) Plot the trajectory of the short rate for the whole simulation from $t = 0$ to $t = 10$.

   ii) Compute a two-sided 99 percent confidence interval for the short rate in 1 year denoted r_1 and report the upper and lower confidence bounds.

   iii) Compute a two-sided 99 percent confidence interval for the short rate under the stationary distribution and report the upper and lower confidence bounds.

You now have to turn your attention to a complex derivative which pays the owner of the derivative the maximum of the short rate over the next $T = 2$ years. That is, the complex derivative will have a maturity of $T = 2$ and a pay-off function $\chi(T)$ given by

$$\chi(T) = \max_{0 \leq t \leq T} r_t \tag{3}$$

To compute the time $t = 0$ price $\Pi(0)$ of the derivative, you can use that

$$\Pi(0) = \mathrm{E}^{\mathbb{Q}}\big[e^{-\int_0^T r_t dt} \chi(T)\big] \tag{4}$$

To compute the fair price of this complex derivative, simulate the short rate over a two year period and repeat the simulation N times. For each simulation n, compute the approximate discounted pay-off χ_n

$$\chi_n = \exp\Big\{ -\frac{T}{M} \sum_{m=0}^{M-1} r_m \Big\} \cdot \Big(\max_{0 \leq m \leq M} r_m \Big) \tag{5}$$

and store the value in a vector. Once all N simulations have completed, compute the fair value by averaging over the expected discounted payoffs for $n = 1, ..., N$.

c) Set M to 1000 and simulate the short rate from $t_0 = 0$ to $T = 2$ at least $N = 1000$ times.

i) Report the fair value of the complex derivative.

ii) Briefly discuss if the method you have chosen will underestimate or overestimate the fair value of the complex derivative.

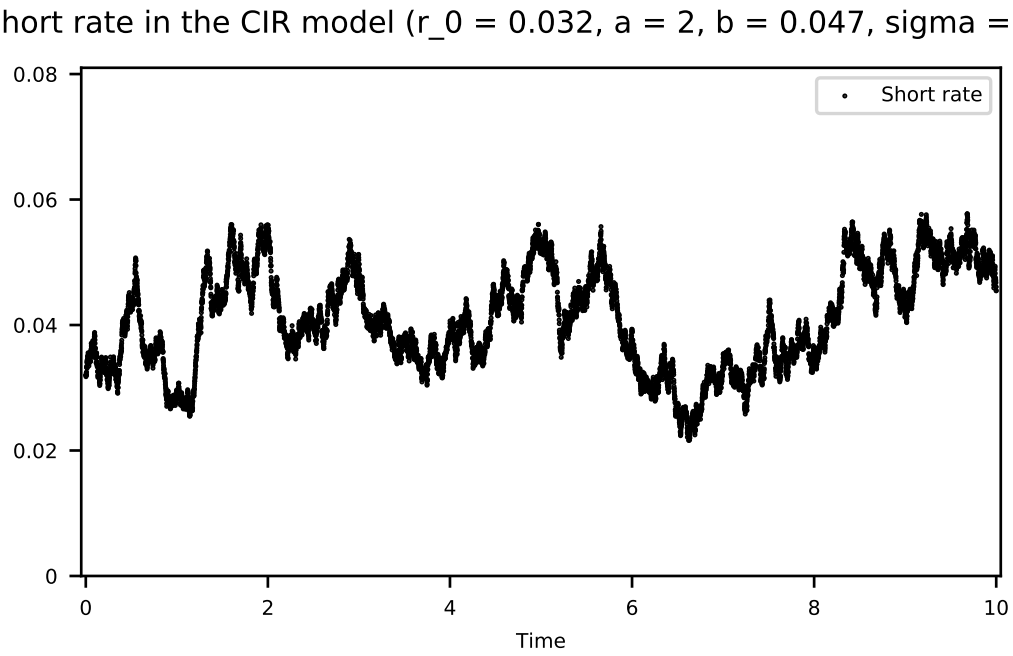
Problem 1 - Solution

a) The CIR model can be fitted to spot rates in many ways for example by minimizing the sum of squared deviations between fitted spot rates and observed spot rates using Nelder-mead.

i) The fitted values might deviate a little bit depending on the specific choice of minimization algorithm but the result should roughly be that $\hat{r}_0 \approx 0.032$, $\hat{a} \approx 2.00$ and $\hat{b} \approx 0.047$.

ii) To demonstrate that the fitted values fit the data, it suffices to report that the sum of squared deviations from the fit is very small. Alternatively, one could plot observed and fitted values of the spot rates.

b) i) The trajectories of simulated short rates will differ but should look something like in the below.



ii) In the CIR model, the short rate r_t follows a non-central chi-squared distribution and the 99 percent CI for r_1 is $[0.02652, 0.06905]$.

iii) Under the stationary distribution, r_t follows a Gamma distribution and the 99 percent CI for r_∞ is $[0.02766, 0.07232]$.

c)

i) The fair value of the derivative as computed by the simulation is of course a random variable but the result should be around 0.0543.

ii) The method proposed here will underestimate the value of the complex derivative. The reason for this is that the maximum of the short rate in the continuous time version of the CIR process can occur at any point in time but here, we are simulating on a grid and won't capture a maximum that happens in-between grid points. The higher the choice of M, the less severe this problem will be.

Problem 2

Your job is to assess the risk of your institution's interest rate swap exposure and therefore, you must first calibrate a zero coupon term structures of interest rates to market data. The 6M EURIBOR fixing has just been announced to be 0.02927 and in addition, you have the data shown in the table below for forward rate agreements and interest rate swaps. The interest rate swaps pay 6M EURIBOR semi-annually against the fixed par swap rates listed below also paid semi-annually.

EURIBOR	Fixing	FRA	Midquote	IRS	Midquote
6M	0.02927	1X7	0.03161	2Y	0.03824
		2X8	0.03295	3Y	0.04083
		3X9	0.03418	4Y	0.04242
		4X10	0.03531	5Y	0.04346
		5X11	0.03635	7Y	0.04468
		6X12	0.03731	10Y	0.04561
		7X13	0.03819	15Y	0.04633
		8X14	0.03900	20Y	0.04667
		9X15	0.03975	30Y	0.04700

a) Calibrate a zero coupon term structure of continuously compounded spot interest rates to the 6M EURIBOR fixing, the forward rate agreement rates and par swap rates given in the table above.

   i) Report zero coupon spot rates for the following six maturities: $T = [0.5, 1, 3, 5, 10, 20, 30]$ based on your calibration.

   ii) Plot the term structures of continuously compounded zero coupon spot rates for maturities ranging from 0 to 30 years.

You will now estimate the continuously compounded *instantaneous* forward rates so that you can assess the properties of your calibration. This should be done by first interpolating, using the same method of interpolation, the spot rates you found above so that you have say 96(or some other multiple of 12) time points per year, then converting spot rates to zcb prices and then converting zcb prices to estimates of instantaneous forward rates.

b) Estimate the term structure of continuously compounded *instantaneous* forward rates.

   i) Plot the instantaneous forward rates in the same type of plot as above for maturities ranging from 0 to 30 years.

   ii) Discuss properties the calibrated and interpolated term structures should posses and whether this can be achieved in practice depending on the choice of interpolation method.

   iii) Discuss whether your calibrated term structures of interest rates has these 'good' properties and relate your conclusion to the choice of interpolation method.

You will now investigate the interest rate risk associated with the $7Y$ Swap by computing the $DV01$ for various changes in the market.

c) Please 'bump' the yield curve in the appropriate ways given below, report the associated $DV01$'s for the $7Y$ swap and answer the question in iii).

   i) The $DV01$ for a 1 bp increase in *all* zero coupon spot rates.

   ii) The $DV01$ for a 1 bp increase in each of the zero coupon spot rates for the following five maturities $T = 1, 2, 3, 5, 7$.

   iii) Compare the values you computed in ii) and briefly discuss whether the 7Y swap is more sensitive to changes to the front end or the back end of the zero coupon yield curve.

Problem 2 - Solution

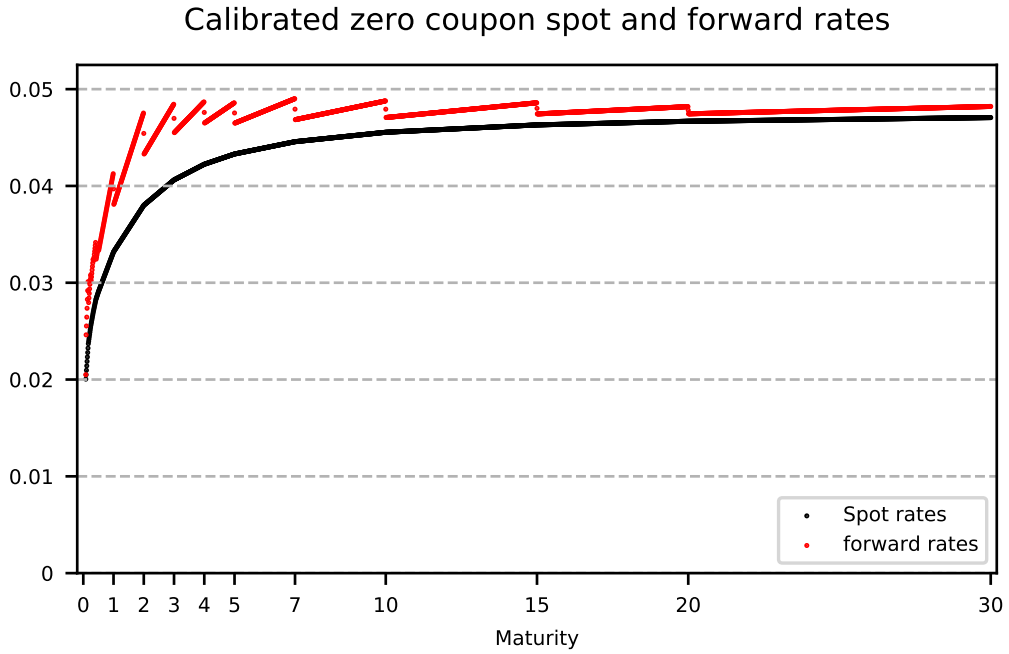
a) The Since the 6M EURIBOR fixing has just been announced, and the 6X12 FRA is known, we know the spot rates for $T = 0.5$ and $T = 1$. The knot-points to use should be the maturities of the swaps.

i) The exact spot rates resulting from the calibration depends on the method of interpolation but they should be roughly

T	0.5	1	3	5	10	20	30
$R(0,T)$	0.02906	0.03318	0.04062	0.04330	0.04555	0.04668	0.04707

ii) See the plot in the answer for b)

b) i) The plot of the spot and instantaneous forward rates will depend slightly on the choice of interpolation method but should look much like the below.



ii) The instantaneous forward rates should be continuous to prevent that interest rate derivatives with nearly the same maturity can have vastly different prices. Instantaneous forward rates will be continuous if and only if zero coupon bond prices are differentiable.

iii) Whether zero coupon bond prices are differentiable depends on the method of interpolation and if for example a piecewise linear interpolation, or for that matter any other 'piecewise' method, is used, instantaneous forward rates will be discontinuous as in the above.

c) The $DV01$'s for the 7Y swap will also depend slightly on the choice of interpolation method.

i) The $DV01$ for a bump of the entire spot rate curve by 1 bp is roughly 6.0767 bp.

ii) DV01 for a bump of the individual spot rates are

T	1	2	3	5	7
$DV01$	0.02161	0.04141	0.05933	0.08994	5.2365

iii) The 7Y swap is more exposed to bumps in the spot rate for higher maturities but is almost entirely exposed to the spot rate with same maturity as the swap itself. This reveals that though seemingly very different, an interest rate swap behaves much like a zero coupon bond in terms of delta risk exposure.

Problem 3

Imagine that your task is to help a corporate client decide on the best way to finance his operations. The client has a loan on which he pays a floating rate of 6M EURIBOR starting exactly 6 months from now and ending in exactly $T = 4$ years. The 6M EURIBOR fixing has just been announced and your quant colleague has computed continuously compounded zero coupon bond spot rates based on market prices. The spot rates $R(0, T_i)$ for various choices of maturity T_i are as shown in the table below. Also, you can observe caplet Black implied volatilities for caplets on 6M forward EURIBOR for a strike of $R = 0.05$ and various settlement dates. The caplet black implied volatilities are likewise shown in the table below. For example, the Black implied volatility for a caplet on the 6M EURIBOR fixing announced at time $T_{i-1} = 2.5$ and paid at time $T_i = 3$ is 0.312.

Table 1: Continuously compounded spot rates and caplet Black implied volatilities

T_i	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00
$R(0, T_i)$	0.0385	0.0431	0.0463	0.0486	0.0502	0.0513	0.0521	0.0527
caplet $\bar{\sigma}_i$	-	0.223	0.241	0.260	0.283	0.312	0.355	0.402

Finally, you can observe that the 2Y2Y payer swaption on 6M EURIBOR for a strike of $K = 0.05$ trades at a Black implied volatility of $\bar{\sigma} = 0.39$. Recall that a 2Y2Y payer swaption will give you the option at exercise in two years to enter into a two year payer swap in which you pay the fixed rate $K = 0.05$ and receive 6M EURIBOR.

a) Compute zero coupon bond prices and 6M forward EURIBOR rates $L(0; T_{i-1}, T_i)$ corresponding to the continuously compounded spot rates. Report, the 6M forward EURIBOR rates corresponding to the fixings taking place at T_{i-1} for $T_{i-1} = \{0.5, 1, 1.5, 2, 2.5, 3, 3.5\}$ in a table.

Your client is concerned that future 6M EURIBOR fixings will be very high so you offer three possibilities:

i) Convert *all* future floating rate payments into a known fixed coupon rate using a 4Y swap.

ii) Use an interest rate cap to insure that the clients floating rate payments will never exceed 0.05.

iii) Use a 2Y2Y payer swaption with a strike of 0.05 so that the client can be sure that his payments to be paid at times $T = 2.5, 3.0, 3.5, 4.0$ will be at most 0.05.

b) Your task is now to find the fair value for each of the choices i), ii), iii) and give the client a quote for each of them. For ii) and iii), you will quote the price in terms of a spread in basis points paid on top of the client's coupon payments.

   i) Report the fixed coupon rate the client would need to pay if he converts his floating rate payments into fixed payments using a 4Y interest rate swap.

   ii) Find the spread in bps, the client would pay on top of his regular interest payments to cap these payments at 0.05.

   iii) Find the spread in bps, the client would pay on top of his interest payments by buying the 2Y2Y swaption with a strike of $K = 0.05$.

c) The three different strategies have different risk and uncertainty profiles and also different costs. For each of the three possibilities i), ii) and iii), please *very* briefly discuss their upside, downside and cost.

Problem 3 - Solution

a) The 6M forward LIBOR rates can be computed directly to give us that

Table 2: 6M forward LIBOR rates

T_{i-1}	0.50	1.00	1.50	2.00	2.50	3.00	3.50
$L(0;T_{i-1}T_i)$	0.0483	0.0534	0.0563	0.0574	0.0576	0.0577	0.0577

b) To solve this problem, it is advantageous to compute the accrual factor, S, of the 4Y swap.

$$S_{swap} = \sum_{i=1}^{i=8} \alpha p(0, T_i) = 3.5808$$

i) The par swap rate for the 4Y swap becomes $R_{swap} = 0.053078$

ii) Caplet prices for caplets with a strike of $R = 0.05$ can be computed using Blacks formula and the price of the 0.05 interest rate cap can be found as the sum of the caplet prices and we get $\pi_{cap} = 0.03612$. To find the semi-annual premium, multiply the price of the cap by $\frac{\alpha}{S_{swap}}$ to get 50.44 basispoints.

iii) The price of the 2Y2Y swaption can be found from Black's formula and we get that $\Pi_{swaption} = 0.02684$. To find the semi-annual premium, divide the price of the swaption by $2S_{swap}$ to get 37.48 basispoints.

c) i) Simply swapping the floating rate payments into a fixed stream of coupon payments eliminates all uncertainty. However, this choice also has no upside in the sense that if interest rates fall, the client will not benefit from future lower interest rate payments.

ii) Entering into an interest rate cap starting right now will insure that the interest payments will never rise above 0.05 so this limits the downside to the client at all future points in time. Also, there is a potential for upside in that the investor will benefit from low future interest rates. The strategy however comes at the cost that the investor will have to pay the premium of roughly 50 bp twice a year regardless of whether the cap comes into effect or not.

iii) Entering into a swaption with an exercise time in two years will insure that interest payments cannot exceed 0.05 from two years into the future and beyond. However, the client is not insured against rises in interest rates prior to the exercise time. This option is thus more risky than ii) and it therefore makes sense that it is less costly. This strategy like ii) has an upside in that the client will benefit from low future interest rates.

Problem 4

You work for the options trading desk and your quant has computed zero coupon bond prices for a range of maturities. These prices can be seen in the table below.

Table 3: Zero coupon bond prices

T	0.50	1.00	1.50	2.00	2.50	3.00
$p(0,T)$	0.98322948	0.96455878	0.94449414	0.92344747	0.90175113	0.87967118
T	3.50	4.00	4.50	5.00	5.50	6.00
$p(0,T)$	0.85741902	0.83516131	0.81302835	0.79112104	0.76951663	0.7482734

In addition to zero coupon bond prices, you can also observe prices $\Pi_{swaption}$ of $2Y4Y$ payer swaptions for a notional of 1 and different strikes. The underlying swap pays a fixed rate semi-annually and against 6M EURIBOR received semi-annually and the swaption prices are given in the table below. Recall that a 2Y4Y payer swaption gives you the right but not obligation to enter into a 4 year payer swap at the time of exercise in 2 years. The strikes are to be interpreted such that a K_{offset} of 0 (ATMF) corresponds to a strike equal to the current 2Y4Y forward rate and a K_{offset} of 100 say corresponds to a strike 100 bps above the current 2Y4Y forward rate.

Table 4: $2Y4Y$ Swaption prices

$K_{offset}(bp)$	-300	-250	-200	-150	-100	-50	ATMF
$\Pi_{swaption}$	0.0995524	0.08350629	0.06774531	0.05248227	0.03808218	0.02519355	0.01482874
$K_{offset}(bp)$	50	100	150	200	250	300	-
$\Pi_{swaption}$	0.00785645	0.00404525	0.00219232	0.00128815	0.00081635	0.00054773	-

a) Compute the accrual factor S and the 2Y *forward* par swap rate for the 4Y swap that serves as the underlying asset for the swaptions and then, using the par swap rate you have found, find the strikes K of the swaptions in the table above.

   i) Report the $2Y4Y$ forward par swap rate.

   ii) Compute Black implied volatilities for all strikes and plot these as a function of K_{offset}.

   iii) Interpret the implied volatility plot and assess if the market is pricing swaptions according to Black's model. What can be said about the distribution of the $2Y4Y$ forward par swap rate implied by the pricing measure chosen by the market and how does that distribution compare to the log normal distribution?

You now decide to fit a SABR model to the implied volatilities you have computed from market swaption prices. The SABR model you should consider is of the usual form given below

$$\begin{aligned} dF_t &= \sigma_t F_t^{\beta} dW_t^{(1)}, \quad F(0) = F_0 \\ d\sigma_t &= \upsilon \sigma_t dW_t^{(2)}, \quad \sigma(0) = \sigma_0 \\ dW_t^{(1)} dW_t^{(2)} &= \rho dt \end{aligned} \tag{6}$$

where F_t here is the 2Y4Y forward par swap rate.

b) Using initial values $\tilde{\sigma}_0 = 0.04$, $\tilde{\beta} = 0.5$, $\tilde{\upsilon} = 0.4$ and $\tilde{\rho} = -0.3$, you now decide to fit a SABR model to the implied volatilities you have found above.

   i) Report the fitted parameter values $\hat{\sigma}_0$, $\hat{\beta}$, $\hat{\upsilon}$ and $\hat{\rho}$.

ii) Assess whether the parameter values you have found match the observed market implied volatilities.

c) Your colleague has just taken a large long position in the payer swaption with a strike 100 below ATMF. He is concerned about his exposure, so he asks you to compute how much he would gain/loose in a few different scenarios. Compute your colleagues gain/loss if:

i) υ falls by 0.02,

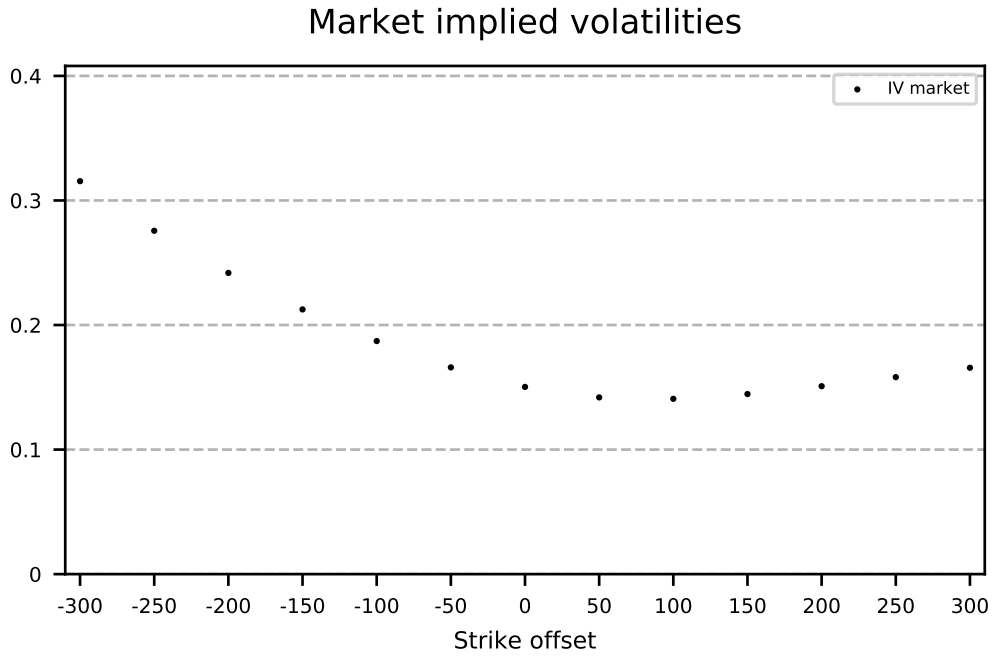
ii) ρ increases by 0.1,

iii) the entire term structure of zero coupon spot rates fall by 1 bp.

Problem 4 - Solution

a) i) The 2Y4Y forward par swap rate becomes roughly 0.05312.

ii) The plot of market implied volatilities looks like this:



iii) There is in a 'smirk' in implied volatilities clearly indicating that market prices are not equivalent to what would arise in a Black's model. The pricing measure chosen by the market is *not* compatible with the $2Y4Y$ forward par swap rate following a log-normal distribution. The distribution implied by the measure chosen by the market has more fat tails and displays more left skewness than that of a log-normal random variable. This is a finding that is very much consistent with typical market behavior.

b) i) The fitted values will differ a bit depending on the method used to fit the parameters but they should be roughly $\hat{\sigma} \approx 0.06$, $\hat{\beta} \approx 0.7$, $\hat{\upsilon} \approx 0.6$ and $\hat{\rho} \approx -0.35$.

ii) The fit should be near perfect which can be demonstrated by reporting the mean-squared error between observed implied volatilities and fitted implied volatilities or by plotting observed and fitted implied volatilities.

c) The DV's should all be negative but the values will differ a bit depending on the parameter values

i) $DV(\upsilon - 0.02) \approx -0.0001169$

ii) $DV(\rho + 0.1) \approx -0.0002004$

iii) $DV(R_i - 0.0001) \approx -0.0002840$. This value is as one might expect since the swaption is in the money and the accrual factor for the forward swap is around 3.

Python Code

```
import numpy as np
from scipy.optimize import minimize
from scipy.stats import gamma
import fixed_income_derivatives as fid
import matplotlib.pyplot as plt

# Problem 1
def fit_cir_obj(param,sigma,R_star,T,scaling = 1):
    r0, a, b = param
    M = len(T)
    R_fit = fid.spot_rate_cir(r0,a,b,sigma,T)
    y = 0
    for m in range(0,M):
        y += scaling*(R_fit[m] - R_star[m])**2
    return y

sigma = 0.08
T = np.array([0.1,0.25,0.5,0.75,1,1.5,2,3,4,5,7,10])
R_star = np.array([0.0334, 0.0352, 0.0375, 0.0392, 0.0405, 0.0422, 0.0433, 0.0445, 0.0451, 0.0455, 0.0459, 0.0462])
# a)
param_0 = 0.025, 1.5, 0.07
result = minimize(fit_cir_obj,param_0,method = 'nelder-mead',args = (sigma,R_star,T),options={'xatol': 1e-8,'disp': True})
print(f"Parameter estimates (r0, a, b): {result.x}")
print(f"Squared deviation from the fit: {result.fun} ")

# b)
r0, a, b = 0.032, 2, 0.047
T_simul, M = 10,10000
delta = T_simul/M
t_simul = np.array([delta*i for i in range(0,M+1)])
r_simul = fid.short_rate_simul(r0,(a,b,sigma),M,T_simul,method = "cir")
fig = plt.figure(constrained_layout=False, dpi = 300, figsize = (5,3))   #
fig.suptitle(f"Short rate in the CIR model (r_0 = 0.032, a = 2, b = 0.047, sigma = 0.08)", fontsize = 9)
gs = fig.add_gridspec(nrows=1,ncols=1,left=0.12,bottom=0.2,right=0.88,top=0.90,wspace=0,hspace=0)
ax = fig.add_subplot(gs[0,0])
xticks = [0,2,4,6,8,10]
ax.set_xticks(xticks)
ax.set_xticklabels(xticks,fontsize = 6)
ax.set_xlim([xticks[0]-0.05,xticks[-1]+0.05])
plt.xlabel(f"Time",fontsize = 6)
ax.set_yticks([0,0.02,0.04,0.06,0.08,0.1])
ax.set_yticklabels([0,0.02,0.04,0.06,0.08,0.1],fontsize = 6)
ax.set_ylim([0,0.081])
# ax.set_ylabel(f"",fontsize = 6)
p1 = ax.scatter(t_simul, r_simul, s = 1, color = 'black', marker = ".",label="Short rate")
plots = [p1]
labels = [item.get_label() for item in plots]
ax.legend(plots,labels,loc="upper right",fontsize = 6)
plt.show()
# fig.savefig("C:/Jacob/Uni_of_CPH/Interest rate derivatives/final_exam_2023/problem_1a.pdf")
lb, ub = fid.ci_cir(r0,a,b,sigma,1,0.99,method = "two_sided")
print(f"99 percent two-sided CI for r_1: {lb,ub}")
alpha, beta = (2*a*b)/(sigma**2), sigma**2/(2*a)
lb_sd, ub_sd = gamma.ppf(0.005, alpha, loc=0, scale=beta), gamma.ppf(0.995, alpha, loc=0, scale=beta)
print(f"99 percent two-sided CI for r_t under the stationary distribution: {lb_sd,ub_sd}")

# c)
M_deriv, T_deriv = 1000, 2
N_deriv = 10000
X = np.zeros([N_deriv])
for n in range(0,N_deriv):
    r_simul = fid.short_rate_simul(r0,(a,b,sigma),M_deriv,T_deriv,method = "cir")
    X[n] = np.exp(-(T_deriv/M_deriv)*sum(r_simul))*max(r_simul)
pi = sum(X)/N_deriv
print(f"Fair value of the derivative: {pi}")

# Investigating if the algorithm has converged - THIS WAS NOT REQUIRED AND STUDENTS ARE NOT PENALIZED IF THEY HAVE NOT DONE THIS!
N_conv = np.array([1000,2000,3000,4000,5000,6000,7000,8000,9000,10000])
for n in N_conv:
    pi_hat = sum(X[0:n])/n
    print(pi_hat)

# Problem 2
def idx_left_right_find(idx,indexes):
    idx_left, idx_right = None, None
    N = len(indexes)
    I_done = False
    n = 0
    while I_done is False and n < N - 3/2:
        if indexes[n] < idx and idx < indexes[n+1]:
            idx_left, idx_right = indexes[n],indexes[n+1]
            I_done = True
        n += 1
    return idx_left, idx_right

def spot_rate_inter_idx(idx_inter,idx_known,T,spot_rate,type = "linear"):
    if type == "linear":
        for idx in idx_inter:
            idx_left, idx_right = idx_left_right_find(idx,idx_known)
            if idx_left is not None and idx_right is not None:
                spot_rate[idx] = ((T[idx_right]-T[idx])*spot_rate[idx_left] + (T[idx]-T[idx_left])*spot_rate[idx_right])/(T[idx_right]-T[idx_left])
    return spot_rate

def spot_rate_inter_maturity(T,spot_rate_known,T_known,type_inter = "linear"):
    I_done = False
    if T < T_known[0]:
        if type_inter == "linear":
            spot_rate = ((T_known[1]-T)*spot_rate_known[0] + (T-T_known[0])*spot_rate_known[1])/(T_known[1]-T_known[0])
        I_done = True
    elif T_known[-1] < T:
        if type_inter == "linear":
```

```
            spot_rate = ((T_known[-1]-T)*spot_rate_known[-2] + (T-T_known[-2])*spot_rate_known[-1])/(T_known[-1]-T_known[-2])
        I_done = True
    else:
        i = 1
        while I_done is False and i < len(T_known) - 0.5:
            if T_known[i-1] <= T and T <= T_known[i]:
                spot_rate = ((T_known[i]-T)*spot_rate_known[i-1] + (T-T_known[i-1])*spot_rate_known[i])/(T_known[i]-T_known[i-1])
            i += 1
    return spot_rate

def rates_insert(rate_insert,indexes,rate):
    j = 0
    for idx in indexes:
        rate[idx] = rate_insert[j]
        j += 1
    return rate

def accrual_factor(idx_maturity,p,idx_swap_fixed):
    accrual_factor = 0
    j, I_done = 0,False
    while I_done == False and j < len(idx_swap_fixed):
        if idx_swap_fixed[j] < idx_maturity + 0.5:
            accrual_factor += p[idx_swap_fixed[j]]
            j += 1
        else:
            I_done = True
    accrual_factor *= delta_swap_fixed
    return accrual_factor

def zcb_curve_cali_obj(spot_rate_knot,idx_swap_knot,spot_rate,idx_inter,idx_knot,idx_swap_fixed,swap_rate_market,scaling = 10000,type_inter = "linear"):
    spot_rate = rates_insert(spot_rate_knot,idx_swap_knot,spot_rate)  # inserting candidate spotrates into vector of all spot rates
    spot_rate = spot_rate_inter_idx(idx_inter,idx_knot,T,spot_rate,type = type_inter)
    p = np.zeros([len(spot_rate)])
    for idx in idx_swap_fixed:
        p[idx] = np.exp(-spot_rate[idx]*T[idx])
    swap_rate = np.zeros([len(spot_rate_knot)])
    i = 0
    for idx in idx_swap_knot:
        S = accrual_factor(idx,p,idx_swap_fixed)
        swap_rate[i] = (1-p[idx])/S
        i += 1
    mse = 0
    for i in range(0,len(swap_rate_market)):
        mse += (swap_rate_market[i] - swap_rate[i])**2
    mse *= scaling/len(swap_rate_market)
    return mse

def zcb_yield_curve_cali(spot_rate,L,fra_rate_market,swap_rate_market,T,indices,scaling = 10000,type_inter = "linear"):
    idx_libor, idx_fra_knot, idx_fra_inter, idx_swap_fixed, idx_swap_knot, idx_swap_inter, idx_inter, idx_knot = indices
    spot_rate[idx_libor] = np.log(1+L*(T[idx_libor]-T[0]))/(T[idx_libor]-T[0])
    spot_rate[2*idx_libor] = (spot_rate[idx_libor]*(T[idx_libor]-T[0]) + fra_rate_market[idx_libor-1]*(T[2*idx_libor]-T[idx_libor]))/(T[2*idx_libor]-T[0])
    args = (idx_swap_knot,spot_rate,idx_swap_inter,idx_knot,idx_swap_fixed,swap_rate_market,scaling,type_inter)
    result = minimize(zcb_curve_cali_obj,swap_rate_market,args = args,options={'disp': False})
    print(f"Spot rates at knot points after minimization: {result.x}")
    # Inserting computed spot rates and interpolating to all time points where there is a cashflow for some swap
    for i in range(0,len(idx_swap_knot)):
        spot_rate[idx_swap_knot[i]] = result.x[i]
    spot_rate = spot_rate_inter_idx(idx_inter,idx_knot,T,spot_rate,type = type_inter)
    # Computing spot rates for maturities before the LIBOR fixing implied by the FRA's
    for i in range(1,idx_libor):
        spot_rate[i] = (spot_rate[i+idx_libor]*T[i+idx_libor] - np.log(1+fra_rate_market[i-1]*T[idx_libor]))/T[i]
    return spot_rate

type_inter = "linear"
scaling = 10000
L_6M = 0.029272109520565297
idx_libor = 6

M_fra = 16
fra_rate_market = [0.03161, 0.03295, 0.03418, 0.03531, 0.03635, 0.03731, 0.03819, 0.03900, 0.03975]
delta_fra = 1/12

M_swap = 58
delta_swap_fixed = 1/2
swap_rate_market = [0.03824, 0.04083, 0.04242, 0.04346, 0.04468, 0.04561, 0.04633, 0.04667, 0.04700]

T = np.array([i*delta_fra for i in range(0,M_fra)] + [3/2 + delta_swap_fixed*i for i in range(0,M_swap)])
spot_rate_known = np.zeros([M_fra+M_swap])
spot_rate_known[idx_libor] = np.log(1+L_6M*(T[idx_libor]-T[0]))/(T[idx_libor]-T[0])
spot_rate_known[2*idx_libor] = (spot_rate_known[idx_libor]*(T[idx_libor]-T[0]) + fra_rate_market[idx_libor-1]*(T[2*idx_libor]-T[idx_libor]))/(T[2*idx_libor]-T[0])

# Fitting the ZCB term structure to market swap rates
idx_all = set([i for i in range(0,M_fra+M_swap)])
idx_known = set([idx_libor,2*idx_libor])
idx_fra_knot = idx_known.union(set({M_fra+1}))
idx_fra_inter = set([i for i in range(idx_libor,M_fra)]).symmetric_difference(idx_fra_knot)
idx_swap_fixed = set({idx_libor,2*idx_libor}).union(set([i for i in range(M_fra,M_fra+M_swap)]))
idx_swap_knot = set({17,19,21,23,27,33,43,53,73})
idx_swap_inter = idx_swap_fixed.symmetric_difference(idx_swap_knot)
idx_knot = idx_fra_knot.union(idx_swap_knot)
idx_inter = idx_fra_inter.union(idx_swap_inter)

idx_fra_knot = sorted(idx_fra_knot,reverse = False)
idx_fra_inter = sorted(idx_fra_inter,reverse = False)
idx_swap_fixed = sorted(idx_swap_fixed,reverse = False)
idx_swap_knot = sorted(idx_swap_knot,reverse = False)
idx_swap_inter = sorted(idx_swap_inter,reverse = False)
idx_inter = sorted(idx_inter,reverse = False)
idx_knot = sorted(idx_knot,reverse = False)
indices = (idx_libor, idx_fra_knot,idx_fra_inter,idx_swap_fixed,idx_swap_knot,idx_swap_inter,idx_inter,idx_knot)

# Finding spot rates at knot points based on swap market data
args = (idx_swap_knot,spot_rate_known.copy(),idx_swap_inter,idx_knot,idx_swap_fixed,swap_rate_market,scaling,type_inter)
```

```
result = minimize(zcb_curve_cali_obj,swap_rate_market,args = args,options={'disp': False})

# Finding spot rates at knot points based on swap market data
args = (idx_swap_knot,spot_rate_known.copy(),idx_swap_inter,idx_knot,idx_swap_fixed,swap_rate_market,scaling,type_inter)
result = minimize(zcb_curve_cali_obj,swap_rate_market,args = args,options={'disp': False})
print(f"Spot rates at knot points after minimization: {result.x}")

# Inserting computed spot rates and interpolating to all time points where there is a cashflow for some swap
spot_rate = spot_rate_known.copy()
for i in range(0,len(idx_swap_knot)):
    spot_rate[idx_swap_knot[i]] = result.x[i]
spot_rate = spot_rate_inter_idx(idx_inter,idx_knot,T,spot_rate,type = type_inter)
# Computing spot rates for maturities before the LIBOR fixing implied by the FRA's
for i in range(1,idx_libor):
    spot_rate[i] = (spot_rate[i+idx_libor]*T[i+idx_libor] - np.log(1+fra_rate_market[i-1]*T[idx_libor]))/T[i]
p = fid.spot_rates_to_zcb(T,spot_rate)
print(f"spot_rates. 0.5Y: {spot_rate[6]}, 1Y: {spot_rate[12]}, 3Y: {spot_rate[19]}, 5Y: {spot_rate[23]}, 10Y: {spot_rate[33]}, 20Y: {spot_rate[53]}, 30Y: {spot_rate[73]}")

# b)
# Interpolating the zero coupon spot rate curve to a finer grid and computing instantaneous forward rates
N_inter = 96     # Should be a multiple of 12
delta = 1/N_inter
N_plot = int(round(30*N_inter - N_inter/12 + 2,0))
T_inter = np.zeros([N_plot])
for i in range(1,N_plot):
    T_inter[i] = 1/12 + (i-1)*delta
spot_rate_inter, forward_rate_inter = np.zeros(len(T_inter)), np.zeros(len(T_inter))
for i in range(0,len(T_inter)):
    spot_rate_inter[i] = spot_rate_inter_maturity(T_inter[i],spot_rate,T,type_inter = type_inter)
p_inter = fid.spot_rates_to_zcb(T_inter,spot_rate_inter)
forward_rate_inter = fid.zcb_to_forward_rates(T_inter,p_inter,horizon = 0)
# Plot
fig = plt.figure(constrained_layout=False, dpi = 300, figsize = (5,3))
fig.suptitle(f"Calibrated zero coupon spot and forward rates", fontsize = 9)
gs = fig.add_gridspec(nrows=1,ncols=1,left=0.12,bottom=0.2,right=0.88,top=0.90,wspace=0,hspace=0)
ax = fig.add_subplot(gs[0,0])
xticks = [0,1,2,3,4,5,7,10,15,20,30]
ax.set_xticks(xticks)
ax.set_xticklabels(xticks,fontsize = 6)
ax.set_xlim([xticks[0]+-0.2,xticks[-1]+0.2])
plt.xlabel(f"Maturity",fontsize = 6)
ax.set_yticks([0,0.01,0.02,0.03,0.04,0.05])
ax.set_yticklabels([0,0.01,0.02,0.03,0.04,0.05],fontsize = 6)
ax.set_ylim([0,0.0525])
plt.grid(axis = 'y', which='major', color=(0.7,0.7,0.7,0), linestyle='--')
p1 = ax.scatter(T_inter[1:], spot_rate_inter[1:], s = 1, color = 'black', marker = ".",label="Spot rates")
p2 = ax.scatter(T_inter[1:], forward_rate_inter[1:], s = 1, color = 'red', marker = ".",label="forward rates")
plots = [p1,p2]
labels = [item.get_label() for item in plots]
ax.legend(plots,labels,loc="lower right",fontsize = 6)
plt.show()
# fig.savefig("C:/Jacob/Uni_of_CPH/Interest rate derivatives/final_exam_2023/problem_2b.pdf")

# DV01 for a position in the 7Y swap
idx_position = 27
S_init = accrual_factor(idx_position,p,idx_swap_fixed)
par_swap_rate_position_init = (1-p[idx_position])/S_init
# DV01 for a bump of the entire zcb curve
spot_rate_bump = spot_rate.copy()
for i in range(0,len(spot_rate_bump)):
    spot_rate_bump[i] += 0.0001
p_bump = fid.spot_rates_to_zcb(T,spot_rate_bump)
S_bump = accrual_factor(idx_position,p_bump,idx_swap_fixed)
par_swap_rate_position = (1-p_bump[idx_position])/S_bump
print(f"DV01 for bumping entire zcb curve: {10000*S_bump*(par_swap_rate_position - par_swap_rate_position_init)}")
# DV01 for a bump of initial points on the zero coupon spot rate curve
idx_bump = [12,17,19,23,27]
DV01_bump = np.zeros([5])
j = 0
for idx in idx_bump:
    spot_rate_bump = spot_rate.copy()
    spot_rate_bump[idx] += 0.0001
    p_bump = fid.spot_rates_to_zcb(T,spot_rate_bump)
    S_bump = accrual_factor(idx_position,p_bump,idx_swap_fixed)
    par_swap_rate_position = (1-p_bump[idx_position])/S_bump
    DV01_bump[j] = 10000*S_bump*(par_swap_rate_position - par_swap_rate_position_init)
    print(f"DV01 for bumping a single point: {DV01_bump[j]}, {T[idx]}")
    j += 1

# Problem 3
M = 9
alpha = 0.5
R = 0.05
T = np.array([i*alpha for i in range(0,M)])
spot_rate = np.array([np.nan,0.0385,0.0431,0.0463,0.0486,0.0502,0.0513,0.0521,0.0527])
sigma_caplet = np.array([np.nan,np.nan,0.223,0.241,0.260,0.283,0.312,0.355,0.402])
sigma_swaption = 0.39
# a)
p = fid.spot_rates_to_zcb(T,spot_rate)
L = fid.zcb_to_forward_LIBOR_rates(T,p,horizon = 1)
print(f"6M forward LIBOR: {L}")
# b)
S_swap = 0
for i in range(1,M):
    S_swap += alpha*p[i]
R_swap = (1-p[M-1])/S_swap
print(f"S_swap: {S_swap}, R_swap: {R_swap}")
price_caplet = np.zeros([M])
for i in range(2,M):
    price_caplet[i] = fid.black_caplet_price(sigma_caplet[i],T[i],R,alpha,p[i],L[i],type = "call")
price_cap = sum(price_caplet)
R_cap = price_cap/S_swap
print(f"price_cap: {price_cap}, R_cap: {R_cap}, premium twice a year: {alpha*R_cap}")
```

```
S_swap_forward = 0
for i in range(5,M):
    S_swap_forward += alpha*p[i]
R_swap_forward = (p[4]-p[8])/S_swap_forward
price_swaption = fid.black_swaption_price(sigma_swaption,T[4],R,S_swap_forward,R_swap_forward,type = "call")
R_swaption = price_swaption/S_swap
print(f"price_swaption: {price_swaption}, R_swaption: {R_swaption}, premium twice a year: {R_swaption/2}")

# Problem 4
def fit_sabr_obj(param,sigma_market,K,T,R):
    sigma_0, beta, upsilon, rho = param
    N = len(sigma_market)
    y = 0
    for n in range(0,N):
        sigma_sabr = fid.sigma_sabr(K[n],T,R,sigma_0,beta,upsilon,rho,type = "call")
        y += (sigma_market[n]-sigma_sabr)**2
    return y

M = 13
alpha = 0.5
p = np.array([1,0.98322948, 0.96455878, 0.94449414, 0.92344747, 0.90175113, 0.87967118, 0.85741902, 0.83516131, 0.81302835, 0.79112104, 0.76951663, 0.7482734])
N = 13
K_swaption_offset = [-300,-250,-200,-150,-100,-50,0,50,100,150,200,250,300]
price_market = np.array([0.0995524, 0.08350629, 0.06774531, 0.05248227, 0.03808218, 0.02519355, 0.01482874, 0.00785645, 0.00404525, 0.00219232, 0.00128815, 0.00081635, 0.00054773])

T = np.array([i*alpha for i in range(0,M)])
idx_exer, idx_set = 4, 12
S = 0
for i in range(idx_exer+1,idx_set + 1):
    S += alpha*p[i]
R_swap = (p[idx_exer] - p[idx_set])/S
print(f"2Y4Y forward par swap rate: {R_swap}")
K, iv_market = np.zeros([N]), np.zeros([N])
for i in range(0,N):
    K[i] = R_swap + K_swaption_offset[i]/10000
    iv_market[i] = fid.black_swaption_iv(price_market[i],T[idx_exer],K[i],S,R_swap,type = "call", iv0 = 0.2, max_iter = 2000, prec = 1.0e-15)
print(f"iv_market")
print(iv_market)

# Plot of market implied volatilities
fig = plt.figure(constrained_layout=False, dpi = 300, figsize = (5,3))
fig.suptitle(f"Market implied volatilities", fontsize = 10)
gs = fig.add_gridspec(nrows=1,ncols=1,left=0.12,bottom=0.2,right=0.88,top=0.90,wspace=0,hspace=0)
ax = fig.add_subplot(gs[0,0])

xticks = K_swaption_offset
ax.set_xticks(xticks)
ax.set_xticklabels(xticks,fontsize = 6)
ax.set_xlim([xticks[0]-10,xticks[-1]+10])
plt.xlabel(f"Strike offset",fontsize = 7)
ax.set_yticks([0,0.1,0.2,0.3,0.4])
ax.set_yticklabels([0,0.1,0.2,0.3,0.4],fontsize = 6)
ax.set_ylim([0,0.408])
plt.grid(axis = 'y', which='major', color=(0.7,0.7,0.7,0), linestyle='--')

p1 = ax.scatter(K_swaption_offset, iv_market, s = 3, color = 'black', marker = ".",label="IV market")
plots = [p1]
labels = [item.get_label() for item in plots]
ax.legend(plots,labels,loc="upper right",fontsize = 5)
# fig.savefig("C:/Jacob/Uni_of_CPH/Interest rate derivatives/final_exam_2023/problem_4a.pdf")
# plt.show()

# b)
param_0 = 0.04, 0.5, 0.4,-0.3
result = minimize(fit_sabr_obj,param_0,method = 'nelder-mead',args = (iv_market,K,T[idx_exer],R_swap),options={'xatol': 1e-8,'disp': True})
print(f"Fitted Parameter values. sigma_0: {result.x[0]}, beta: {result.x[1]}, upsilon: {result.x[2]}, rho: {result.x[3]}")
sigma_0, beta, upsilon, rho = result.x

# c)
idx_position = 4
# Bumping upsilon
upsilon_bump = upsilon - 0.02
sigma_upsilon = fid.sigma_sabr(K[idx_position],T[idx_exer],R_swap,sigma_0,beta,upsilon_bump,rho,type = "call")
price_upsilon = fid.black_swaption_price(sigma_upsilon,T[idx_exer],K[idx_position],S,R_swap,type = "call")
print(f"price after bumping upsilon: {price_upsilon}, diff: {price_upsilon-price_market[idx_position]}")
# Bumping rho
rho_bump = rho + 0.1
sigma_rho = fid.sigma_sabr(K[idx_position],T[idx_exer],R_swap,sigma_0,beta,upsilon,rho_bump,type = "call")
price_rho = fid.black_swaption_price(sigma_rho,T[idx_exer],K[idx_position],S,R_swap,type = "call")
print(f"price after bumping rho: {price_rho}, diff: {price_rho- price_market[idx_position]}")
# Bumping the entire spot rate curve
R = fid.zcb_to_spot_rates(T,p)
R_bump = R - 0.0001*np.ones([M])
p_bump = fid.spot_rates_to_zcb(T,R_bump)
S_bump = 0
for i in range(idx_exer+1,idx_set + 1):
    S_bump += alpha*p_bump[i]
R_swap_bump = (p_bump[idx_exer] - p_bump[idx_set])/S_bump
sigma_delta = fid.sigma_sabr(K[idx_position],T[idx_exer],R_swap_bump,sigma_0,beta,upsilon,rho,type = "call")
print(sigma_delta)
price_delta = fid.black_swaption_price(sigma_delta,T[idx_exer],K[idx_position],S_bump,R_swap_bump,type = "call")
print(f"price after bumping spot rates: {price_delta}, diff: {price_delta-price_market[idx_position]}")
```