KL

July 10, 2021

[1]:
```
import numpy as np
import pandas as pd
from scipy.optimize import minimize_scalar
import seaborn as sns
import matplotlib.pyplot as plt

prob_mtx = pd.read_csv("prob_mtx_weib_single_threshld.csv").iloc[:, 1:]
zc_bond = pd.read_csv("cir_prices.csv").iloc[:, 1:]
K = 100.  # face value
A = 0.5  # amount lost if bond is triggered
payoff = K * prob_mtx + (K * A * (1 - prob_mtx));
payoff = payoff.to_numpy()

scaler = 1000  # a scaler to V_o to avoid overflow in np.exp; after scaling,␣
↪the unit of price is one thousand dollar.
cir_price = pd.read_csv("cir_prices.csv").iloc[:, 1:].to_numpy() / scaler


def ann_to_inst(r):
    """
    Converts annualized to short rate
    """
    return np.log1p(r)


def mkt_price(payoff, cir_price, risk_p, T, t):
    pi_ = [1 / payoff.shape[0]] * payoff.shape[0]
    market_price = np.zeros(payoff.shape)

    for i in range(1, payoff.shape[1] + 1):
        market_price[i - 1, :] = (
                cir_price[i - 1, :] * payoff[i - 1, :] * np.
↪exp(ann_to_inst(risk_p) * (T - t[i - 1])) * pi_[i - 1])

    return sum(market_price.sum(0))
```

```
t_time = np.linspace(0., 2., 100)
market_price = mkt_price(payoff, cir_price, risk_p=4.35 / 100, T=2., t=t_time)
print("market price V_0 = ", market_price)

# Finding the risk neutral probabilities
T, N = cir_price.shape
v_0 = market_price

def lagr_func_scale(lam):
    Rsum = [0] * N
    alpha = [0] * N
    for i in range(1, N + 1):
        for t in range(1, T + 1):
            alpha[i - 1] += cir_price[t - 1, i - 1] * payoff[t - 1, i - 1]
        Rsum[i - 1] = np.exp(lam * (alpha[i - 1] - v_0))
    return np.sum(Rsum)

print("optimizing with T = " + str(T) + " N = " + str(N))
res = minimize_scalar(fun=lagr_func_scale, method='brent')
opt_lam = res['x']
print("opt_lam = ", opt_lam)

alpha = [0] * N
for i in range(1, N + 1):
    for t in range(1, T + 1):
        alpha[i - 1] += cir_price[t - 1, i - 1] * payoff[t - 1, i - 1]

for i in range(1, N + 1):
    print("alpha[", i, "]=", alpha[i - 1], ", alpha-v_0=", alpha[i - 1] - v_0)

lam_alpha = [0] * N
for i in range(1, N + 1):
    lam_alpha[i - 1] = opt_lam * alpha[i - 1]

max_lam_alpha = np.max(lam_alpha)

lam_alpha_normalized = [0] * N
for i in range(1, N + 1):
    lam_alpha_normalized[i - 1] = lam_alpha[i - 1] - max_lam_alpha

pi_ = [1 / N] * N  # equal probability of all N scenarios generated/simulated
Denominator = 0
for i in range(1, N + 1):
    Denominator += pi_[i - 1] * np.exp(lam_alpha_normalized[i - 1])

pi_star = [0] * N
for i in range(1, N + 1):
```

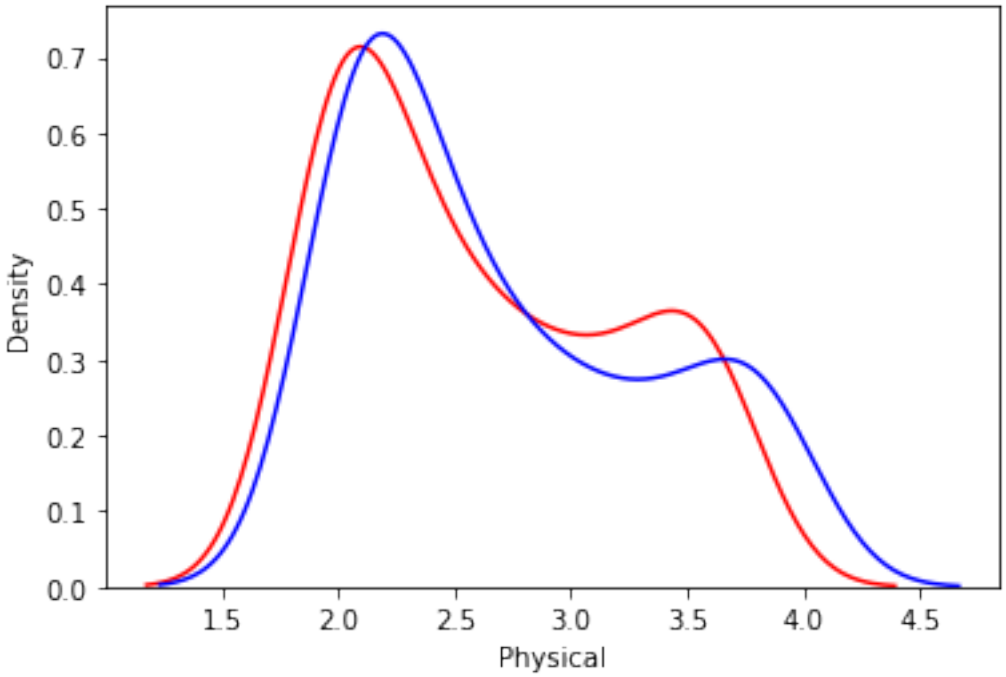
```
    pi_star[i - 1] = pi_[i - 1] * np.exp(lam_alpha_normalized[i - 1]) /␣
 ↪Denominator

print("sum of pi_start = ", np.sum(pi_star))

v_start = 0
for i in range(1, N + 1):
    v_start += pi_star[i - 1] * alpha[i - 1]

print('V_0=', v_0, ", V_start=", v_start)

print("pi_star = ", pi_star)

rskn_pv = np.zeros((T, N))
for i in range(1, T + 1):
    rskn_pv[i - 1, :] = cir_price[i - 1, :] * payoff[i - 1, :] * pi_star[i - 1]

rskn_pv = rskn_pv.sum(0)

# physical present value
pi_ = [1 / N] * N
phy_pv = np.zeros((T, N))
for i in range(1, T + 1):
    phy_pv[i - 1, :] = cir_price[i - 1, :] * payoff[i - 1, :] * pi_[i - 1]

phy_pv = phy_pv.sum(0)

pv = pd.DataFrame({'Physical': phy_pv, 'Risk Neutral': rskn_pv})
pv.sum(0)

# expected risk premium per annum ( risk neutral - physical )
print("pv sum diff = ", pv.sum(0)[1] - pv.sum(0)[0])

sns.kdeplot(pv.iloc[:, 0], color='r')
sns.kdeplot(pv.iloc[:, 1], color='b')
plt.show()
```

```
market price V_0 =  273.50321914932334
optimizing with T = 100 N = 100
opt_lam =  0.003374269266631613
alpha[ 1 ]= 364.9628437631844 , alpha-v_0= 91.45962461386108
alpha[ 2 ]= 365.8401561139969 , alpha-v_0= 92.33693696467355
alpha[ 3 ]= 365.1507121245603 , alpha-v_0= 91.64749297523696
alpha[ 4 ]= 367.0700246549372 , alpha-v_0= 93.56680550561384
alpha[ 5 ]= 366.21146110748475 , alpha-v_0= 92.70824195816141
alpha[ 6 ]= 365.27694451818394 , alpha-v_0= 91.7737253688606
alpha[ 7 ]= 363.86362369857113 , alpha-v_0= 90.36040454924779
alpha[ 8 ]= 362.32125235348656 , alpha-v_0= 88.81803320416321
```

```
alpha[ 9 ]= 359.8526560566455 , alpha-v_0= 86.34943690732217
alpha[ 10 ]= 358.48559630097355 , alpha-v_0= 84.98237715165021
alpha[ 11 ]= 357.316622695866 , alpha-v_0= 83.81340354654265
alpha[ 12 ]= 355.74690202223513 , alpha-v_0= 82.24368287291179
alpha[ 13 ]= 352.1898132657787 , alpha-v_0= 78.68659411645535
alpha[ 14 ]= 349.95172087089094 , alpha-v_0= 76.4485017215676
alpha[ 15 ]= 347.80156343112907 , alpha-v_0= 74.29834428180573
alpha[ 16 ]= 343.9153076018108 , alpha-v_0= 70.41208845248747
alpha[ 17 ]= 340.59912408223005 , alpha-v_0= 67.0959049329067
alpha[ 18 ]= 339.253984422536 , alpha-v_0= 65.75076527321266
alpha[ 19 ]= 335.5589113045685 , alpha-v_0= 62.05569215524514
alpha[ 20 ]= 331.5755099557072 , alpha-v_0= 58.072290806383876
alpha[ 21 ]= 329.3634294882433 , alpha-v_0= 55.860210338919956
alpha[ 22 ]= 326.4942794300896 , alpha-v_0= 52.99106028076625
alpha[ 23 ]= 324.0613770395006 , alpha-v_0= 50.558157890177256
alpha[ 24 ]= 319.5101150301981 , alpha-v_0= 46.006895880874765
alpha[ 25 ]= 315.60109546050336 , alpha-v_0= 42.097876311180016
alpha[ 26 ]= 312.5235966325085 , alpha-v_0= 39.02037748318514
alpha[ 27 ]= 308.46283409487813 , alpha-v_0= 34.95961494555479
alpha[ 28 ]= 306.1636492782752 , alpha-v_0= 32.66043012895187
alpha[ 29 ]= 304.0088281698043 , alpha-v_0= 30.50560902048096
alpha[ 30 ]= 300.8412276536241 , alpha-v_0= 27.338008504300774
alpha[ 31 ]= 295.85622119447805 , alpha-v_0= 22.35300204515471
alpha[ 32 ]= 294.8411729185514 , alpha-v_0= 21.33795376922808
alpha[ 33 ]= 291.4899624406479 , alpha-v_0= 17.986743291324558
alpha[ 34 ]= 287.32718478580597 , alpha-v_0= 13.823965636482626
alpha[ 35 ]= 285.3151607232513 , alpha-v_0= 11.811941573927982
alpha[ 36 ]= 280.7229676999833 , alpha-v_0= 7.219748550659972
alpha[ 37 ]= 278.26916253510063 , alpha-v_0= 4.765943385777291
alpha[ 38 ]= 275.8140272254715 , alpha-v_0= 2.3108080761481347
alpha[ 39 ]= 273.427943792442 , alpha-v_0= -0.07527535688132048
alpha[ 40 ]= 271.0121247649781 , alpha-v_0= -2.4910943843452173
alpha[ 41 ]= 266.9629925213151 , alpha-v_0= -6.54022662800827
alpha[ 42 ]= 265.14367274370017 , alpha-v_0= -8.359546405623178
alpha[ 43 ]= 262.1640021387391 , alpha-v_0= -11.339217010584264
alpha[ 44 ]= 260.44940269515155 , alpha-v_0= -13.053816454171795
alpha[ 45 ]= 256.476600587096 , alpha-v_0= -17.026618562227327
alpha[ 46 ]= 255.77559185736126 , alpha-v_0= -17.72762729196208
alpha[ 47 ]= 251.37688760884248 , alpha-v_0= -22.126331540480862
alpha[ 48 ]= 251.01821922310032 , alpha-v_0= -22.484999926223026
alpha[ 49 ]= 247.16252510466327 , alpha-v_0= -26.34069404466007
alpha[ 50 ]= 245.4825048673469 , alpha-v_0= -28.02071428197644
alpha[ 51 ]= 243.7971312931791 , alpha-v_0= -29.70608785614425
alpha[ 52 ]= 241.6172750960865 , alpha-v_0= -31.88594405323684
alpha[ 53 ]= 239.38903529188482 , alpha-v_0= -34.11418385743852
alpha[ 54 ]= 237.3971112848971 , alpha-v_0= -36.106107864426235
alpha[ 55 ]= 235.20865606669264 , alpha-v_0= -38.2945630826307
alpha[ 56 ]= 233.00066149241437 , alpha-v_0= -40.50255765690898
```

```
alpha[ 57 ]= 230.65823148512678 , alpha-v_0= -42.844987664196566
alpha[ 58 ]= 229.1447478099095 , alpha-v_0= -44.35847133941385
alpha[ 59 ]= 226.86591205955256 , alpha-v_0= -46.637307089770786
alpha[ 60 ]= 225.18811663181162 , alpha-v_0= -48.31510251751172
alpha[ 61 ]= 223.95344082877315 , alpha-v_0= -49.54977832055019
alpha[ 62 ]= 222.09018420762666 , alpha-v_0= -51.41303494169668
alpha[ 63 ]= 220.4148585831675 , alpha-v_0= -53.08836056615584
alpha[ 64 ]= 219.47878057827225 , alpha-v_0= -54.02443857105109
alpha[ 65 ]= 217.20496696878874 , alpha-v_0= -56.298252180534604
alpha[ 66 ]= 216.13631894330135 , alpha-v_0= -57.366900206021995
alpha[ 67 ]= 215.58464134738117 , alpha-v_0= -57.91857780194218
alpha[ 68 ]= 213.62736695428075 , alpha-v_0= -59.8758521950426
alpha[ 69 ]= 212.81326812400877 , alpha-v_0= -60.68995102531457
alpha[ 70 ]= 210.82298711703345 , alpha-v_0= -62.680232032289894
alpha[ 71 ]= 210.140054765962 , alpha-v_0= -63.36316438336135
alpha[ 72 ]= 208.3700094505844 , alpha-v_0= -65.13320969873894
alpha[ 73 ]= 207.92554508647333 , alpha-v_0= -65.57767406285001
alpha[ 74 ]= 206.0466683470329 , alpha-v_0= -67.45655080229045
alpha[ 75 ]= 204.85866495557138 , alpha-v_0= -68.64455419375196
alpha[ 76 ]= 204.40158797523304 , alpha-v_0= -69.1016311740903
alpha[ 77 ]= 202.87557476833814 , alpha-v_0= -70.6276443809852
alpha[ 78 ]= 202.0356531358771 , alpha-v_0= -71.46756601344623
alpha[ 79 ]= 201.10020226168172 , alpha-v_0= -72.40301688764163
alpha[ 80 ]= 200.44428680364248 , alpha-v_0= -73.05893234568086
alpha[ 81 ]= 200.1548235437582 , alpha-v_0= -73.34839560556514
alpha[ 82 ]= 199.19949746535696 , alpha-v_0= -74.30372168396639
alpha[ 83 ]= 198.32013855133576 , alpha-v_0= -75.18308059798758
alpha[ 84 ]= 197.5203430356537 , alpha-v_0= -75.98287611366965
alpha[ 85 ]= 198.16912812105303 , alpha-v_0= -75.33409102827031
alpha[ 86 ]= 196.18625319581813 , alpha-v_0= -77.31696595350522
alpha[ 87 ]= 196.07170900068584 , alpha-v_0= -77.4315101486375
alpha[ 88 ]= 194.98408039799742 , alpha-v_0= -78.51913875132593
alpha[ 89 ]= 193.7046473321193 , alpha-v_0= -79.79857181720405
alpha[ 90 ]= 194.5013943206634 , alpha-v_0= -79.00182482865995
alpha[ 91 ]= 193.90218132348102 , alpha-v_0= -79.60103782584233
alpha[ 92 ]= 193.11088467764532 , alpha-v_0= -80.39233447167803
alpha[ 93 ]= 192.3059629427822 , alpha-v_0= -81.19725620654114
alpha[ 94 ]= 191.45708234859305 , alpha-v_0= -82.0461368007303
alpha[ 95 ]= 191.06849066190347 , alpha-v_0= -82.43472848741987
alpha[ 96 ]= 191.37483033574745 , alpha-v_0= -82.1283888135759
alpha[ 97 ]= 190.2420274410595 , alpha-v_0= -83.26119170826385
alpha[ 98 ]= 190.68592473599918 , alpha-v_0= -82.81729441332416
alpha[ 99 ]= 189.78538412824193 , alpha-v_0= -83.71783502108141
alpha[ 100 ]= 189.82494941189154 , alpha-v_0= -83.6782697374318
sum of pi_start =  0.9999999999999997
V_0= 273.50321914932334 , V_start= 273.50321881811715
pi_star =  [0.013913166083396529, 0.013954414086286168, 0.013921988692018925,
0.014012443941619393, 0.013971908291669674, 0.013927919916016546,
```

```
0.013861656837862581, 0.013789702960475997, 0.013675315820072566,
0.013612379213648074, 0.013558791868729506, 0.013487165413076369,
0.01332625231261401, 0.013225992490072393, 0.013130382389963497,
0.012959324111351492, 0.012815121508736016, 0.012757087227057028,
0.012599017143826358, 0.012430805890195008, 0.012338365849140592,
0.012219490973864988, 0.012119588524535763, 0.01193488765196819,
0.011778499130445841, 0.011656820415668307, 0.01149818671386629,
0.011409328123023973, 0.011326672363978229, 0.011206253755589007,
0.011019332601523693, 0.01098165543809772, 0.010858175588408731,
0.010706724199923363, 0.010634281251195204, 0.010470770016850078,
0.010384432065724444, 0.010298759800526603, 0.01021617442481664,
0.01013323451995539, 0.009995727042128236, 0.009934552488743786,
0.009835168851583573, 0.009778431569060208, 0.009648223410552485,
0.00962542854805719, 0.00948361896798302, 0.00947214842024943,
0.009349712528575692, 0.00929686044464733, 0.00924414013918454,
0.009176395041857433, 0.009107659378552957, 0.009046649439608524,
0.008980091075575057, 0.008913434705876706, 0.008843260703438006,
0.008798214164541698, 0.008730820569120643, 0.008681532132519104,
0.008645438989609573, 0.008591254505561986, 0.008542825160358045,
0.008515884539570048, 0.008450796772267484, 0.00842037886689811,
0.00840471883806262, 0.008349393852581537, 0.008326489635540781,
0.008270758410232374, 0.008251721237075451, 0.008202583813745935,
0.008190291270778817, 0.008138530418051739, 0.008105971254991346,
0.008093479044830475, 0.008051911380262752, 0.008029123590023574,
0.0080038199134734, 0.007986125164678193, 0.007978328708875234,
0.007952651754885916, 0.007929089670162445, 0.007907720082541972,
0.007925050432863684, 0.007872202862807227, 0.00786916082103718,
0.007840334308682036, 0.007806559354562566, 0.007827575049272244,
0.007811764417502214, 0.00779093445278858, 0.007769802808182012,
0.0077475792036038925, 0.007737427134361687, 0.0077454292354354,
0.007715879756875025, 0.00772744548491388, 0.007703999994771367,
0.007705028577633708]
pv sum diff =  8.992751149466017
```


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Density
1.5
2.0
2.5
3.0
3.5
4.0
4.5
Physical

[ ]: