In [2]:
```
# codes provided in "Quantlib Python Cookbook" by Balaraman and Ballal
#
import QuantLib as ql
todaysDate = ql.Date(31, 10, 2018)
ql.Settings.instance().evaluationDate = todaysDate
spotDates = [ql.Date(1, 11, 2018), ql.Date(30, 4, 2019), ql.Date(31, 1
spotRates = [0.02173, 0.0280013, 0.0308375]
dayCount = ql.Thirty360()
calendar = ql.UnitedStates()
interpolation = ql.Linear()
compounding = ql.Compounded
compoundingFrequency = ql.Annual
spotCurve = ql.ZeroCurve(spotDates, spotRates, dayCount, calendar, int
spotCurveHandle = ql.YieldTermStructureHandle(spotCurve)
#
#construct the fixed rate bond
issueDate = ql.Date(31, 10, 2018)
maturityDate = ql.Date(31, 10, 2019)
tenor = ql.Period(ql.Semiannual)
calendar = ql.UnitedStates()
bussinessConvention = ql.Unadjusted
dateGeneration = ql.DateGeneration.Backward
monthEnd = False
schedule = ql.Schedule (issueDate, maturityDate, tenor, calendar, buss
list(schedule)

# Now lets build the coupon
dayCount = ql.Thirty360()
couponRate = .03
coupons = [couponRate]

# Now lets construct the FixedRateBond
settlementDays = 0
faceValue = 100
fixedRateBond = ql.FixedRateBond(settlementDays, faceValue, schedule,

# create a bond engine with the term structure as input;
# set the bond to use this bond engine
bondEngine = ql.DiscountingBondEngine(spotCurveHandle)
fixedRateBond.setPricingEngine(bondEngine)

# Finally the price
fixedRateBond.NPV()
```

Out[2]:
```
99.94317139071408
```

In [1]:
```
# codes adapted from "Quantlib Python Cookbook" by Balaraman and Balla
#
from QuantLib import *

calc_date = Date(30,6,2020)
Settings.instance().evaluationDate = calc_date

day_count = ActualActual(ActualActual.Bond)
rate = 0.03
ts = FlatForward(calc_date, rate,
```

```
day_count, Compounded,
Semiannual)
ts_handle = YieldTermStructureHandle(ts)

callability_schedule = CallabilitySchedule()
call_price = 100.0
call_date = Date(30,September,2020);
null_calendar = NullCalendar();
for i in range(0,16):
    callability_price = CallabilityPrice(
        call_price, CallabilityPrice.Clean)
    callability_schedule.append(
        Callability(callability_price,
            Callability.Call,
            call_date))
call_date = null_calendar.advance(call_date, 3,
                            Months)

issue_date = Date(30,September,2019)
maturity_date = Date(30,September,2024)
calendar = UnitedStates(UnitedStates.GovernmentBond)
tenor = Period(Quarterly)
accrual_convention = Unadjusted
schedule = Schedule(issue_date, maturity_date, tenor,
                    calendar, accrual_convention,
                    accrual_convention,
                    DateGeneration.Backward, False)

settlement_days = 3
face_amount = 100
accrual_daycount = ActualActual(ActualActual.Bond)
coupon = 0.025
bond = CallableFixedRateBond(
            settlement_days, face_amount,
            schedule, [coupon], accrual_daycount,
            Following, face_amount, issue_date,
            callability_schedule)

def value_bond(a, s, grid_points, bond):
    model = HullWhite(ts_handle, a, s)
    engine = TreeCallableFixedRateBondEngine(model, grid_points)
    bond.setPricingEngine(engine)
    return bond

value_bond(0.00127, 0.00676, 40, bond)
print ("Callable bond price: ",bond.cleanPrice())
```

```
Callable bond price:  98.02719311711265
```

In [11]:

```
from QuantLib import *
import datetime
import numpy as np
import matplotlib.pyplot as plt


# global data
# Here enter the Yield Curve reference Data
calendar = TARGET()
todaysDate = Date(28,February,2014);
```

```
Settings.instance().evaluationDate = todaysDate
settlementDate = Date(4,March,2014);

# market quotes
# Update deposit Rates ( usual source will be LIBOR Fixings on the Cu
deposits = { (1,Weeks): 0.0023,
             (1,Months): 0.0023,
             (3,Months): 0.0023,
             (6,Months): 0.0023}
# Obtain Futures prices from CME traded Euro Dollar Futures
futures = { Date(19,3,2014): 99.765,
            Date(18,6,2014): 99.75,
            Date(17,9,2014): 99.73,
            Date(17,12,2014): 99.69,
            Date(18,3,2015): 99.605,
            Date(17,6,2015): 99.47,
            Date(16,9,2015): 99.3,
            Date(16,12,2015): 99.085 }
# Obtain Swap rates from Traded Swaps on the Curve data
swaps = { (3,Years): 0.0079,
          (4,Years): 0.012,
          (5,Years): 0.0157,
          (6,Years): 0.01865,
          (7,Years): 0.0216,
          (8,Years): 0.0235,
          (9,Years): 0.0254,
          (10,Years): 0.0273,
          (15,Years): 0.0297,
          (20,Years): 0.0316,
          (25,Years): 0.0335,
          (30,Years): 0.0354}

# convert them to Quote objects
for n,unit in deposits.keys():
    deposits[(n,unit)] = SimpleQuote(deposits[(n,unit)])
for d in futures.keys():
    futures[d] = SimpleQuote(futures[d])
for n,unit in swaps.keys():
    swaps[(n,unit)] = SimpleQuote(swaps[(n,unit)])

# build rate helpers

dayCounter = Actual360()
settlementDays = 2
depositHelpers = [ DepositRateHelper(QuoteHandle(deposits[(n,unit)]),
                                     Period(n,unit), settlementDays,
                                     calendar, ModifiedFollowing,
                                     False, dayCounter)
                   for n, unit in [(1,Weeks),(1,Months),(3,Months),
                                   (6,Months)] ]

dayCounter = Actual360()
months = 3
futuresHelpers = [ FuturesRateHelper(QuoteHandle(futures[d]),
                                     d, months,
                                     calendar, ModifiedFollowing,
                                     True, dayCounter,
                                     QuoteHandle(SimpleQuote(0.0)))
                   for d in futures.keys() ]
```

```
settlementDays = 2
fixedLegFrequency = Semiannual
fixedLegTenor = Period(6,Months)
fixedLegAdjustment = Unadjusted
fixedLegDayCounter = Thirty360()
floatingLegFrequency = Quarterly
floatingLegTenor = Period(3,Months)
floatingLegAdjustment = ModifiedFollowing
swapHelpers = [ SwapRateHelper(QuoteHandle(swaps[(n,unit)]),
                               Period(n,unit), calendar,
                               fixedLegFrequency, fixedLegAdjustment,
                               fixedLegDayCounter, Euribor3M())
                for n, unit in swaps.keys() ]

# term structure handles

discountTermStructure = RelinkableYieldTermStructureHandle()
forecastTermStructure = RelinkableYieldTermStructureHandle()

# term-structure construction

helpers = depositHelpers[:2] + futuresHelpers + swapHelpers[1:]
depoFuturesSwapCurve = PiecewiseFlatForward(settlementDate, helpers,
                                            Actual360())
print(depoFuturesSwapCurve.dates())

df=[]
dates1=[]
for c in depoFuturesSwapCurve.dates():
    df.append(depoFuturesSwapCurve.discount(c))
    dates1.append(c)
    print(depoFuturesSwapCurve.discount(c))

termStructure = YieldTermStructureHandle(depoFuturesSwapCurve)



#End of Yield Curve Construction

#Begin building forward Curve

# Forward swap underlying the Swaption to be priced
# In this case I am pricing a 5y into 5Y swap
swapEngine = DiscountingSwapEngine(discountTermStructure)

nominal = 1000000
length = 5
maturity = calendar.advance(settlementDate,length,Years)
payFixed = True

fixedLegFrequency = Semiannual
fixedLegAdjustment = Unadjusted
fixedLegDayCounter = Thirty360()

floatingLegFrequency = Quarterly
spread = 0.0
fixingDays = 2
index = Euribor3M(forecastTermStructure)
floatingLegAdjustment = ModifiedFollowing
floatingLegDayCounter = index.dayCounter()
```

```
#ATM forward Rate
fixedRate = 0.040852

forwardStart = calendar.advance(settlementDate,5,Years)
forwardEnd = calendar.advance(forwardStart,length,Years)
fixedSchedule = Schedule(forwardStart, forwardEnd,
                         fixedLegTenor, calendar,
                         fixedLegAdjustment, fixedLegAdjustment,
                         DateGeneration.Forward, False)
floatingSchedule = Schedule(forwardStart, forwardEnd,
                            floatingLegTenor, calendar,
                            floatingLegAdjustment, floatingLegAdjustme
                            DateGeneration.Forward, False)

forward = VanillaSwap(VanillaSwap.Payer, nominal,
                      fixedSchedule, fixedRate, fixedLegDayCounter,
                      floatingSchedule, index, spread,
                      floatingLegDayCounter)
forward.setPricingEngine(swapEngine)



def formatPrice(p,digits=2):
    format = '%%.%df' % digits
    return format % p

def formatRate(r,digits=2):
    format = '%%.%df %%%%' % digits
    return format % (r*100)

headers = ("term structure", "net present value",
           "fair spread", "fair fixed rate" )
separator = " | "

format = ''
width = 0
for h in headers[:-1]:
    format += '%%%ds' % len(h)
    format += separator
    width += len(h) + len(separator)
format += '%%%ds' % len(headers[-1])
width += len(headers[-1])

rule = "-" * width
dblrule = "=" * width
tab = " " * 8

def report(swap, name):
    print(format % (name, formatPrice(swap.NPV(),2),formatRate(swap.fa

print(dblrule)
print("5-year market Spot swap-rate = %s" % formatRate(swaps[(5,Years)
print(dblrule)

discountTermStructure.linkTo(depoFuturesSwapCurve)
forecastTermStructure.linkTo(depoFuturesSwapCurve)
report(forward,'depo-fut-swap')
```

```
##############################################
# Bulding the European Swaption pricer part

exercise = maturity
exercised = EuropeanExercise(exercise)
settlementtype="physical"
atmswaption = Swaption(forward,exercised)
#Applying a 15.3% implied volatility to 5y into 5y ATM swaption
vol1  = QuoteHandle(SimpleQuote(0.1533))
atmswaption.setPricingEngine(BlackSwaptionEngine(termStructure,vol1))
print(atmswaption.NPV())

#*****************************
#Now given Market Premium implying the underlying volatility

index = Euribor3M(termStructure)
#Place holder for the iterator to hold the implied volatility in the
#swaption Helper
swaptionVols = [ # maturity,          length,             volatility
                 (Period(5, Years), Period(5, Years), 0.1533)]

helpers = [ SwaptionHelper(maturity, length,
                           QuoteHandle(SimpleQuote(vol)),
                           index, index.tenor(), index.dayCounter(),
                           index.dayCounter(), termStructure)
                  for maturity, length, vol in swaptionVols ]

for swaption, helper in zip(swaptionVols, helpers):
        maturity, length, vol = swaption
        print(swaption)
        helper.setPricingEngine(BlackSwaptionEngine(termStructure,vol1
        NPV = helper.modelValue()
        print(NPV)
        NPV=0.06   # here we are adding a premium of 60,000 per 1 MM 1
        implied = helper.impliedVolatility(NPV, 1.0e-4, 1000, 0.05, 0.
        print(implied)
```

```
(Date(4,3,2014), Date(11,3,2014), Date(4,4,2014), Date(19,6,2014), Dat
e(18,9,2014), Date(17,12,2014), Date(17,3,2015), Date(18,6,2015), Date
(17,9,2015), Date(16,12,2015), Date(16,3,2016), Date(5,3,2018), Date
(4,3,2019), Date(4,3,2020), Date(4,3,2021), Date(4,3,2022), Date(6,3,2
023), Date(4,3,2024), Date(5,3,2029), Date(6,3,2034), Date(4,3,2039),
Date(4,3,2044))
1.0
0.9999552797777655
0.9998019836626801
0.9993040408678879
0.9986725503403595
0.9979983537512772
0.9972255039856889
0.9962089501943928
0.9948723760044732
0.9931297659369858
0.9908380401199864
0.9528936817980842
0.9237534456011471
0.8926361507010357
0.8569543184460349
0.8248052246196683
0.7903509635631968
```

```
0.7538424321181283
0.6305965353829129
0.5174507103535108
0.4118294386779761
0.31535261989746427
=====================================================================
5-year market Spot swap-rate = 1.57 %
=====================================================================
 depo-fut-swap |            -45.25 |    0.0011 % |        4.0841 %
23134.40637208518
(Period("5Y"), Period("5Y"), 0.1533)
0.023150444670187967
0.40926904416494736
```

In [ ]: