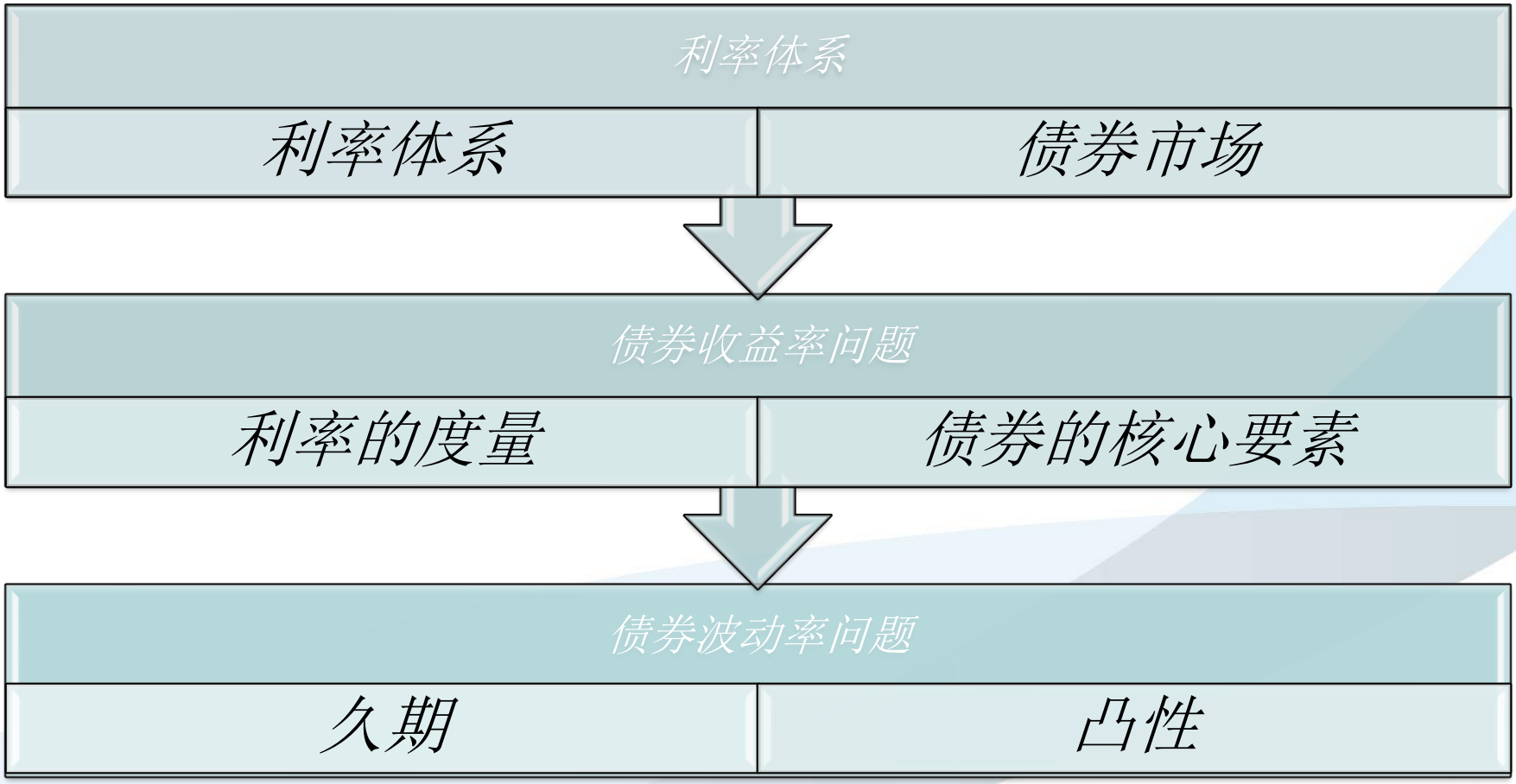
Chapter 7 – Python利率与债券分析

Frank Ziwei Zhang
School of Finance

上海对外经贸大学
SHANGHAI UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS

Contents



7.1 利率体系

7.1.1 中央银行利率

现行的利率是以中央银行利率为基础、金融机构利率为主体以及金融市场利率并存的利率体系。

1、再贷款利率

再贷款利率是指中央银行向金融机构发放再贷款所采用的利率，其中，中央银行对金融机构的贷款称为再贷款。自1984年中国人民银行专门行使中央银行职能以来，再贷款一直是我国中央银行的重要货币政策工具。根据中国人民银行官方网站公布的信息，2010年12月26日至今再贷款利率保持不变。

再贷款期限	再贷款利率水平
20天	3.25%
3个月	3.55%
6个月	3.75%
1年	3.85%

2、再贴现利率

再贴现利率是指金融机构将所持有的已贴现票据向中央银行办理再贴现所采用的利率。其中，贴现是指票据的持票人在票据到期日前，为了取得资金而贴付一定利息将票据权利转让给商业银行的行为；再贴现则是中央银行通过购入银行持有的已贴现但尚未到期的票据向商业银行提供融资支持的行为。根据中国人民银行公布的信息，2010年12月26日至今，对金融机构的再贴现利率一直保持在2.25%。

3、存款准备金利率与超额存款准备金利率

存款准备金利率是指中央银行对金融机构交存的法定存款准备金支付的利率。超额存款准备金利率则是指中央银行对金融机构交存的准备金中超过法定存款准备金水平的部分支付的利率。根据中国人民银行公布的信息，从2008年12月27日至今，对金融机构的存款准备金利率和超额存款准备金利率分别为1.62%和0.72%。

4、央行回购利率

央行回购利率是指中央银行在开展回购业务过程中支付或者收取的利率。所谓回购（repurchase agreement, repo）是指拥有证券的金融机构同意将证券出售给交易对方，并在未来约定时间以更高的价格将证券买回，出售证券的金融机构得到资金，所支付的利息等于证券卖出与买入之间的差价，相应的利率称为回购利率（repo rate）

回购是中国人民银行的一项公开市场业务，按照方向不同分为正回购和逆回购两种。正回购是中国人民银行向一级交易商卖出有价证券，并约定在未来特定日期买回有价证券的交易行为。逆回购是中国人民银行向一级交易商购买有价证券，并约定在未来特定日期将有价证券卖给一级交易商的交易行为。

2013年1月，中国人民银行在公开市场推出了短期流动性调节工具（Short-term Liquidity Operations,SLO）。短期流动性调节工具是以7天期以内短期回购为主，采用市场化利率招标方式开展操作。

5、央票利率

央票利率是中央银行在公开市场操作时发行票据的票面利率。中央银行票据（简称“央票”）是中央银行为调节商业银行超额准备金而向商业银行发行的短期债务凭证，其实质是中央银行债券，央票是重要的公开市场操作手段，同时还被视为提供市场基准利率的重要手段。我国首次发行央票是在2002年6月发行了期限6个月、金额50亿元、发行利率1.9624%的央票。

6、新型货币政策工具的利率

从2013年至今，除了上面介绍的短期流动性调节工具以外，中国人民银行先后创设了常备借贷便利、抵押补充贷款、中期借贷便利等新型货币政策工具，而这些工具本身就有支付利率和引导市场利率的要求。

常备借贷便利（Standing Lending Facility,SLF）是中国人民银行正常的流动性供给渠道

抵押补充贷款（Pledged Supplemental Lending,PSL），主要功能是支持国民经济重点领域、薄弱环节和社会事业发展而对金融机构提供的期限较长、金额较大的融资，采取质押方式发放。

中期借贷便利（Medium-term Lending Facility,MLF），对象是符合宏观审慎管理要求的商业银行、政策性银行，采取质押方式发放

定向中期借贷便利（Targeted Medium-term Lending Facility,TMLF），为机构提供长期稳定资金来源，定向支持、扩大对小微企业、民营企业信贷投放。

7.1.2 金融机构利率

1、金融机构存贷款利率

2013年7月20日中国人民银行全面放开金融机构贷款利率管制，取消金融机构贷款利率0.7倍的下限，人民币贷款利率完全市场化。2015年10月23日中国人民银行进一步宣布不再设置商业银行和农村合作金融机构存款利率浮动上限，人民币存款利率也完全市场化。

目前，商业银行吸收人民币存款、发放人民币贷款的利率均是在参考中国人民银行设定的基准利率基础上自行设置。

项目	期限	基准利率
金融机构人民币存款	活期存款	0.35%
	3个月	1.10%
	半年	1.30%
	1年	1.50%
	2年	2.10%
	3年	2.75%
金融机构人民币贷款	1年以内（含1年）	4.35%
	1至5年（含5年）	4.75%
	5年以上	4.90%

2、贷款基础利率

贷款基础利率（Loan Prime Rate，LPR）是商业银行对其最优质客户执行的贷款利率，其他贷款利率可在此基础上加减点生成。2013年10月25日，国内的贷款基础利率集中报价和发布机制正式运行，集中报价和发布机制是在报价行自主报出本行贷款基础利率的基础上，指定发布人对报价进行加权平均计算，形成报价行的贷款基础利率报价平均利率并对外予以公布。目前向社会公布的是1年期贷款基础利率，报价银行共10家。中国外汇交易中心暨全国银行间同业拆借中心是贷款基础利率的指定发布人。

交易中心官方网站提供相关的数据下载，通过下载贷款基础利率的全部数据（2013年10月至2019年2月），并且运用Python进行可视化（见图7-1），具体的代码如下：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
LPR = pd.read_excel('D:\Zhangzw\Python\Python金融数据分析\RawData\第7章\贷款基础利率（LPR）数据.xls',sheet_name="Sheet1",header=0,index_col=0) #导入外部数据
LPR.plot(figsize=(8,6),grid = True,fontsize = 13) #数据可视化
```

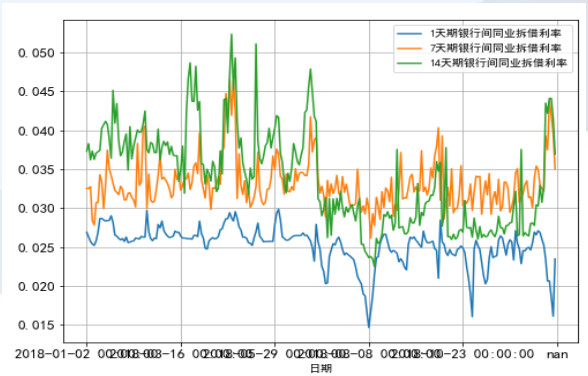
7.1.3 金融市场利率

1、银行间同业拆借利率

同业拆借（Interbank lending）是金融机构同业之间的短期资金融通行为，目的在于调剂头寸和临时性资金余缺。同业拆借利率就是基于同业拆借行为而发生的。目前，同业拆借是指与全国银行间同业拆借中心联网的金融机构之间通过同业拆借中心的交易系统进行的无担保资金融通行为，交易期限包括1天、7天、14天、21天、1个月、2个月、3个月、4个月、6个月、9个月和1年共11个品种。

全国银行间同业拆借中心提供了同业拆借利率的相关数据下载，下载2018年同业拆借利率的全部数据并且通过Python绘制1天、7天和14天这3个最常用的同业拆借利率的走势图（见图7-2），具体的代码如下：

```
IBL = pd.read_excel('D:\Zhangzw\Python\Python金融数据分析\RawData\第7章\银行间同业拆借利率（2018年）
.xls',sheet_name="Sheet1",header=0,index_col=0) #导入外部数据
(IBL.iloc[:,0:3]).plot(figsize=(8,6),grid = True,fontsize = 13) #数据可视化
```



2、回购利率

目前，货币市场的回购交易分为质押式回购与买断式回购两类，这两类回购均有各自不同的利率。

质押式回购是交易双方进行的以债券为权利质押的一种短期资金融通业务，指资金融入方（正回购方）在将债券出质给资金融出方（逆回购方）融入资金的同时，双方约定在将来某一日期由正回购方按约定回购利率计算的资金额向逆回购方返还资金，逆回购解除出质债券上质权的融资行为。同时，根据交易场所的不同，质押式回购可以分为银行间质押式回购、上海证券交易所质押式回购、深圳证券交易所质押式回购，交易期限与银行间同业拆借的期限保持相同，也依然是11个品种，其中，最具有参考价值的是1天期、7天期和14天期这3个品种。

通常而言，运用1天期、7天期、14天期的定盘回购利率作为回购市场的基准性利率指标。全国银行间同业拆借中心提供了银行间回购利率的相关数据下载。下载2018年的1天期、7天期和14天期的定盘回购利率数据并且通过Python绘制相关的走势图（见图7-3），具体的代码如下：

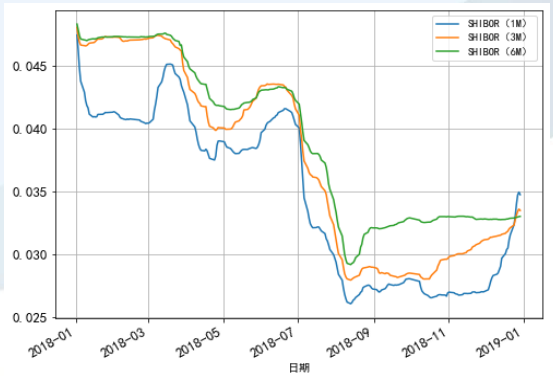
```
FR = pd.read_excel('D:\Zhangzw\Python\Python金融数据分析\RawData\第7章\银行间回购定盘利率（2018年）
.xls',sheet_name="Sheet1",header=0,index_col=0) #导入外部数据
FR.plot(figsize=(8,6),grid = True,fontsize = 13) #数据可视化
```

3、上海银行间同业拆放利率

上海银行间同业拆放利率（Shanghai Interbank Offered Rate, Shibor）。以位于上海的全国银行间同业拆借中心为技术平台计算、发布并命名，从2007年1月4日开始正式运行，是由信用等级较高的银行组成报价团自主报出的人民币同业拆出利率计算确定的算术平均利率，是单利、无担保、批发性利率。目前， Shibor品种包括隔夜（O/N）、1周（1W）、2周（2W）、1个月（1M）、3个月（3M）、6个月（6M）、9个月（9M）及1年（1Y），其中，最具有参考价值的是3个月期的报价。

上海银行间同业拆放利率网站提供Shibor利率的相关数据下载。通过下载2018年的Shibor利率数据并且通过Python绘制常用的1个月、3个月和6个月期Shibor利率的走势图（图7-4），具体的代码如下：

```
Shibor = pd.read_excel('D:\Zhangzw\Python\Python金融数据分析\RawData\第7章\Shibor利率（2018年）.xls',sheet_name="Sheet1",header=0,index_col=0) #导入外部数据
(Shibor.iloc[:,3:6]).plot(figsize=(8,6),grid = True,fontsize = 13) #数据可视化
```



7.2 债券市场

7.2.1 债券交易场所

1、银行间市场

银行间市场是债券市场的主体，市场参与者是各类机构投资者，实行双边谈判成交，属于场外批发市场。中央国债登记结算有限公司（简称“中央结算公司”）作为债券中央托管机构，为债券实行集中统一托管，为银行间市场投资者开立证券账户，并且为市场的交易结算服务。交易主体除了境内的金融机构以外，也向境外央行或货币当局、国际金融组织和主权财富基金开放。银行间债券市场的交易品种包括现券交易、质押式回购、买断式回购、远期交易以及债券借贷等。

2、交易所市场

交易所市场是债券市场的重要组成部分，目前开展债券交易的证券交易所有上海证券交易所和深圳证券交易所，它由包括个人在内的各类社会投资者参与，属于集中撮合交易的零售市场。交易所市场实行两级托管体制，中央结算公司是债券一级托管人，负贵为易所开立代理总账户；中国证券登记结算公司（简称“中证登”）是债券二级托管人，负责对交易所投资者账户的交易进行记录。目前，交易所债券市场的交易品种包括现券交易、质押式回购等。

3、商业银行柜台市场

商业银行柜台市场是银行间市场的延伸，参与者限定为个人投资者，属于场外零售市场。柜台市场实行两级托管体制，中央结算公司是债券一级托管人，负责为承办银行开立债券自营账户和代理总账户；承办银行是债券二级托管人。目前，商业银行柜台市场的交易品种仅为现券交易。

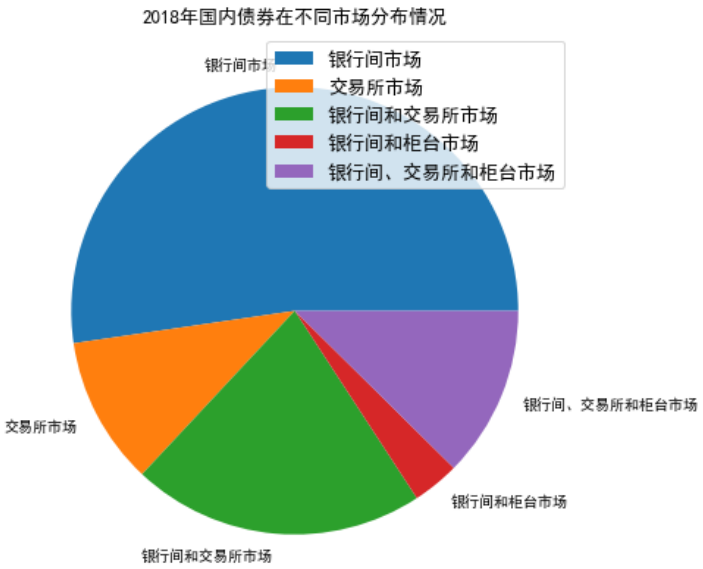
此外，一些债券也会选择在上述3个市场中的2个甚至是3个市场同时交易：

名称	债券余额（亿元）	余额比重
银行间市场	439 925.50	52.53%
交易所市场	90094.55	10.71%
银行间和交易所市场	178 020.30	21.17%
银行间和柜台市场	28 581.48	3.40%
银行间、交易所和柜台市场	104 232.50	12.40%
合计	840 854.33	100%

7.2.1 债券交易场所

为了更形象地展示债券的分布情况，通过外部导入数据，用Python绘制出2018年末存量债券在银行间市场、交易所市场以及柜台市场分布情况的饼图（见图7-6），具体的代码如下：

```
bond = pd.read_excel('D:\Zhangzw\Python\Python金融数据分析\RawData\第7章\国内债券市场按照交易场所分类（2018年末）.xlsx',sheet_name="Sheet1",header=0,index_col=0) #导入外部数据
plt.figure(figsize=(9,7))
plt.pie(x=bond['债券余额(亿元)'],labels=bond.index)
plt.legend(loc=1,fontsize=13)
plt.title(u'2018年国内债券在不同市场分布情况',fontsize=13)
```



7.2.2 债券品种

债券品种	债券余额（单位：亿元）	余额占比
政府债		
国债	148 803.67	17.36%
地方政府债	180 699.54	21.08%
政府支持机构债	16 095.00	1.88%
金融债		
政策银行债	143 826.63	16.77%
商业银行债	13 851.20	1.62%
商业银行次级债	24 234.82	2.83%
保险公司债	2 651.53	0.31%
证券公司债	13 312.28	1.55%
证券公司短期融资债	460.00	0.05%
其他金融机构债	4 781.50	0.56%
同业存单	98 826.70	11.53%
企业债		
一般企业债	25 605.00	2.99%
集合企业债	85.50	0.01%
公司债		
一般公司债	33 272.73	3.88%
私募债	24 953.981	2.91%
中期票据	56 402.23	6.58%
短期融资券		
一般短期融资券	4 718.30	0.55%
超短期融资券	14 567.80	1.70%
非公开定向债务融资工具	19 428.80	2.27%
资产支持证券		
银保监会主管的资产支持证券	10 934.79	1.28%
证监会主管的资产支持证券	14 056.19	1.64%
交易商协会的资产支持票据	1 692.49	0.20%
可转换公司债券	1 904.26	0.22%
可交换公司债券	1 975.99	0.23%
国际机构债	254.60	0.03%
合计	957 395.31	100%

7.3 利率的度量

7.3.1 利率的复利频次

【例7-1】国内有A银行、B银行、C银行、D银行、E银行、F银行共6家商业银行在2018年年底均对外发布"1年期存款利率为2%”的信息，这句话表面上感觉非常直接，含义清楚并且一致，但是事实上这句话的含义会伴随着利率的计算方式（即利率的复利频次）的变化而变化，具体如下：

A银行：利率计算方式是1年复利1次，在A银行存入100元，在一年以后得到的本息和就是

100×（1+2%）=102（元）

B银行：利率的计算方式变为每半年复利1次，这表示每6个月能获取2%/2=1%的利息，而且利息被用于再投资，在B银行存入100元在1年后得到的木息和就是

$$100 * （1 + \frac{2\%}{2}）^{2} = 102.01（元）$$

C银行：利率的计算方式变为每季度复利1次，这表示每个季度能尖取2%/4=0.5%的利息，而且利息被用于再投资，在C银行存入100在年后得到的本息和就是

$$100 * （1 + \frac{2\%}{4}）^{4} = 102.0151（元）$$

D银行：利率的计算方式变为每月复利1次，这表示每月能获取2%/12的利息，而且利息被用于再投资，在D银行存入100元在1年后得到的本息和就是

$$100 * （1 + \frac{2\%}{12}）^{12} = 102.0184（元）$$

7.3.1 利率的复利频次

E银行：E银行：利率的计算方式变为每周复利1次，并且假定一年有52周，这表示每周能获取2%/52的利息，而且利息被用于再投资，在E银行存入100元在1年后得到的本息和就是

$$100 * （1 + \frac{2\%}{52}）^{52} = 102.0197（元）$$

F银行：利率的计算方式变为每天复利1次，并且假定一年有365天，这表示每天能获取2%/365的利息，而且利息被用于再投资，在F银行存入100元在1年后得到的本息就是

$$100 * （1 + \frac{2\%}{365}）^{365} = 102.0201（元）$$

```
r=0.02                              #一年期利率2%
M=[1,2,4,12,52,365]                      #不同复利的频次
name=['一年复利1次','每半年复利1次','每季度复利1次','每月复利1次','每周复利1次','每天复利1次'] #建立一个包含不同复利频次的字符串列表
value=[]                    #建立一个初始的存放一年后的本息合计数的数列
i=0                                 #设置一个标量
for m in M:
   value.append(100*(1+r/m)**m)
   print(name[i],round(value[i],4))
    i=i+1
```

7.3.1 利率的复利频次

现在将例7-1的结论进行推广，并且用数学符号给出统一的数学表达式。假设初始的投资本金用A表示，投资期限n年，利率R是按年复利的年利率，每年复利频次用m表示，投资的终值FV的公式如下：

$$FV = A（1 + \frac{R}{\mathrm{m}}）^{\mathrm{mn}} \qquad （7\text{-}1）$$

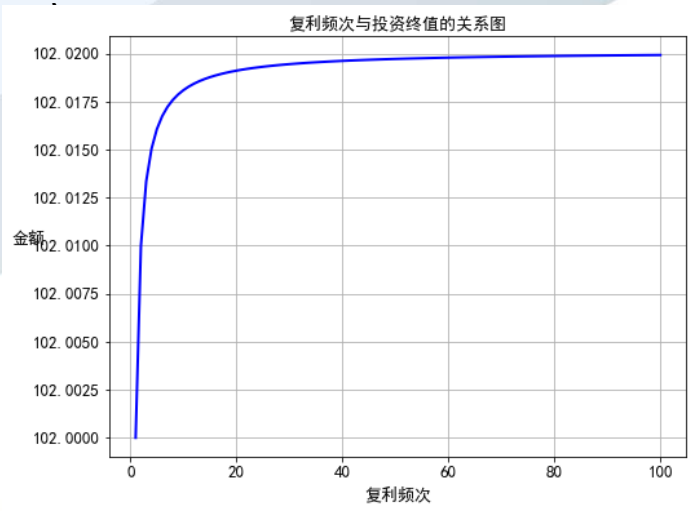
```
def FV(A,n,R,m):
    '''构建一个用于计算不同复利频次的投资终值函数
    A：表示初始的投资本金；
    n：表示投资期限（年）；
    R：表示年利率R是按年复利的利率；
    m：表示每年复利频次，输入Y代表1年复利1次，S代表每半年复利一次，
    Q代表每季度复利一次，M代表每月复利一次，W代表每周复利一次，
    D代表每天复利一次'''
    import numpy as np
    if m=='Y':
        return A*(1+R)**(n)
    elif m=='S':
        return A*(1+R/2)**(n*2)
    elif m=='Q':
        return  A*(1+R/4)**(n*4)
    elif m=='M':
        return  A*(1+R/12)**(n*12)
    elif m=='W':
        return A*(1+R/52)**(n*52)
    else:
        return A*(1+R/365)**(n*365)
FV_M=FV(A=100,n=1,R=0.02,m='M') #用于验证每月复利的结果
print('每月复利1次得到的本息和',round(FV_M,4))
```

7.3.1 利率的复利频次

假定初始投资本金是100元，每年复利一次的年化利率是2%，投资期限是1年，考察复利频次从1到100对应的一年后投资终值。

```
r=0.02                          #复利一次的年利率2%
M=np.arange(1,101)              #生成从1到100的自然数数列
PV=100                          #初始投资100元
FV=PV*(1+r/M)**M                #计算投资终值

plt.figure(figsize=(8,6))
plt.plot(M,FV,'b-',lw=2.0)
plt.xlabel(u'复利频次',fontsize =13)
plt.ylabel(u'金额',fontsize =13,rotation=0)
plt.xticks(fontsize=13)
plt.yticks(fontsize=13)
plt.title(u'复利频次与投资终值的关系图',fontsize=
plt.grid('True')
plt.show()
```



7.3.2 连续复利

在连续复利条件下，运用极限定理，式子（7-1）可以写成：

$$FV = \lim_{m\to+\infty} A(1+\frac{R}{M})^{mn} = Ae^{Rn} \qquad \text{（7-2）}$$

其中，e是自然对数的底数，是一个无限不循环小数，近似等于2.71828。Python的math模块、NumPy模块都有计算幂函数 e^x 的功能

假设Rc代表连续复利利率，Rm是与连续复利利率等价的每年m次复利利率，根据式子（7-1）和式子（7-2）。就可以得到

$$R_c = m \times \ln(1+\frac{R_m}{m}) \qquad \text{（7-3）}$$

$$R_m = m（e^{R_c/m} - 1） \qquad \text{（7-4）}$$

```
def Rc(Rm,m):
    '''构建已知复利频次和对应的复利利率，计算等价连续复利利率的函数
    Rm：代表了复利频次m的复利利率
    m：代表了复利频次。'''
    import numpy as np          #导入Numpy模块
    return m*np.log(1+Rm/m)     #输出等价的连续复利利率结果
```

```
def Rm(Rc,m):
    '''构建已知复利频次和连续复利利率，计算等价的对应复利m次复利利率函数
    Rc：代表了连续复利利率
    m：代表了复利频次。'''
    import numpy as np          #导入Numpy模块
    return m*(np.exp(Rc/m)-1)   #输出等价的对应复利频次的复利利率结果
```

7.3.2 连续复利

【例7-2】假定G商业银行对外的利率报价是5%，按季度复利，计算相对应的连续复利函数。由于m=4、Rn=5%，根据等式（7-3），等价的连续复利利率等于：

$$4\ln(1+\frac{5\%}{4}) = 4.969\%$$

下面，直接运用刚才通过 Python定义的计算等价连续复利利率的函数Rc进行求解，具体的代码如下：

```
R_c=Rc(Rm=0.05,m=4.)            #计算连续复利利率
print('等价的连续复利利率',round(R_c,6))
```

等价的连续复利利率 0.04969

【例7-3】假设H商业银行对外的利率报价是6%，该利率是连续复利利率，计算等价的每月复利的复利利率。由于m=12，Rn=6%，由等式（7-4）得出，与之等价的按月复利的利率等于

$$12 \times \left(e^{0.06/12} - 1\right) = 6.015\%$$

直接运用刚才通过Python定义的计算复利频次m的复利利率的函数Rm进行求解，具体的代码如下：

```
R_m=Rm(Rc=0.06,m=12)            #计算按月复利的复利利率
print('等价的按月复利的复利利率',round(R_m,6))
```

等价的按月复利的复利利率 0.06015

7.3.3 零息利率

【例7-4】假如3年期连续复利的零息利率是4%，这意味着今天的100元按照该零息利率投资并且只能在3年后获得本息合计100×e^4%x3=112.7497（元）。运用 Python进行计算的代码如下：

```
A=100
r=0.04
T=3
FV=A*np.exp(r*T)
print('三年后到期的本息合计数',round(FV,4))
```

三年后到期的本息合计数 112.7497

在金融市场上直接观察到的利率（如债券的票面利率）并不是一个零息利率，因此通常需要运用带票息的债券市场价格推算出相应的零息利率以及零息利率曲线。

7.4 债券的核心要素

7.4.1 债券的核心要素

债券的核心要素债券有几个核心要素，包括本金、债券期限、票面利率以及债券价格等。

本金（principle）也称为面值（par value），是债券发行人承诺偿还给债券持有人的货币总额。

债券期限是指债券上载明的偿还债券本金的期限，即债券发行日至到期日之间的时间间隔。

票面利率（coupon rate）是指债券利息与债券面值的比率，是发行人承诺在债券存续期支付给债券持有人利息的计算标准。票面利率乘以债券本金就得到了票面利息（简称“票息”），票面利息的支付方式可以每年支付一次，也可以每半年甚至每季度支付一次。票面利为零的债券被称为是零息债券（zero- coupon bond）。

债券价格是债券进行市场交易的价格，价格是按照本金的百分数报出，债券价格的面值基数是100元。债券价格分为净价和全价，其中，净价（clean price）是债券买卖的价格，也是债券市场的报价；净价加上应计利息就等于债券的全价（dirty price，也称为“发票价”）。

7.4.2 基于单一贴现率的债券定价

如果债券的票息是每年支付m次（m≥1），同时假定B代表债券价格，C代表票面利率， M代表债券本金，y代表贴现利率（连续复利），T代表债券期限（用年表示）。则债券定价公式如下：

$$B = \frac{C}{m} \times M \times \sum_{t=1}^{mT} e^{-yt/m} + Me^{-yT} = (\frac{C}{\mathrm{m}} \sum_{t=1}^{mT} e^{-yt/m} + e^{-yT})M \qquad \textbf{（7-5）}$$

```
def Bond_price(C,M,T,m,y):
  '''构建计算债券价格的函数
  C：表示债券的票面利率；
  M：表示债券的本金；
  T：表示债券的期限，用年表示；
  m：表示债券票面利率每年的支付频次；
  y：表示贴现利率，也就是债券到期收益率。'''
  import numpy as np
  coupon=[]                    #建立一个初始的存放每期票息现值的列表
  for i in np.arange(1,T*m+1):
    coupon.append(np.exp(-y*i/m)*M*C/m) #计算每期债券票息的现值并放入列表
  return np.sum(coupon)+np.exp(-y*T)*M  #输出最终的债券价格
```

7.4.2 基于单一贴现率的债券定价

【例7-5】2008年6月中国国家开发银行发行了“08国开1”债券，该债券面值100元，期限为20年，票面利率5.25%，每年付息2次，起息日为2008年6月24日。假定今天是2018年6月25日，此时该债券的剩余期限为10年，贴现率假定是4.2%（连续复利）计算今天该债券的价格。

$$B = 100 \times \left(\frac{5.25\%}{2} \sum_{t=1}^{20} e^{-4.2\% t/2} + e^{-4.2\% \times 10} \right)$$ =108.1253(元)

通过计算可以得到在2018年6月25日，“08国开11”债券的价格是108.1253元。同时，可以运用前面在 Python中自定义的计算债券价格函数 Bond_price直接求出结果，具体的代码如下：

```
Bond = Bond_price(C=0.0525,M=100,T=10,m=2,y=0.042)  #输入债券的相关要素
print('计算得到的债券价格',round(Bond,4))
```

计算得到的债券价格 108.1253

7.4.3 债券到期收益率

【例7-6】假定有一份期限是5年、面值为100元的债券，票面利率5%，每半年付息一次，假设该债券当前的市场价格是98元。根据式子（7-5），可以得到如下的等式：

$$100 \times (\frac{5\%}{2}\sum_{t=1}^{10} e^{-0.5yt} + e^{-5y}) = 98$$

但是通过这个等式求解y并非一件容易的事情，通常需要运用迭代的方式计算。运用Python可以很方便地得到结果，就是运用在6节介绍的SciPy子模块optimize中的函数fslove，具体的计算过程分为两个步骤。

```
def YTM(C,M,T,m,P):
  '''构建计算债券到期收益率（连续复利）的函数
  C：债券的票面利率；
  M：债券的本金；
  T：债券的期限，用年表示；
  m：债券票面利率每年的支付频次；
  P：债券的市场价格。'''
  import scipy.optimize as so          #导入SciPy的子模块optimize
  import numpy as np
  def f(y):
    coupon=[]          #建立一个初始的存放每一期票息现值的列表
    for i in np.arange(1,T*m+1):
      coupon.append(np.exp(-y*i/m)*M*C/m)
          #计算每一期债券票息的现值并放入列表
    return np.sum(coupon)+np.exp(-y*T)*M-P  #相当于输出一个等于零的式子
  return so.fsolve(f,0.1)
```

7.4.3 债券到期收益率

第2步：运用第一步中自定义的计算债券到期收益率（连续复利）的函数YTM，求解出例7-6的到期收益率，具体的代码如下：

```
Bond_yield=YTM(C=0.05,M=100,T=5,m=2,P=98) #得到的结果是一个列表
print('计算得到债券的到期收益率',np.round(Bond_yield,6))
```

计算得到债券的到期收益率 [0.053892]

7.4.4 基于不同期限贴现率的债券定价

针对债券存续期内不同时刻的现金流，采用对应不同期限的连续复利零息利率yt作为贴现率，则前面讨论的债券定价公式（7-5）就变为

$$B = (\frac{C}{m}\sum_{t=1}^{mT} e^{-y_t * t/m} + e^{-y_T T})M \qquad \text{（7-6）}$$

```
def Bond_value(c,t,y):
    '''构建基于不同期限零息利率作为贴现率计算债券价格的函数
    c：表示债券存续期内现金流，用数组的数据结构输入；
    t：表示对应于产生现金流的时刻或期限，用数组的数据结构输入；
    y：表示不同期限的零息利率，用数组的数据结构输入。'''
    import numpy as np
    cashflow=[]                    #生成存放每期现金流现值的初始数列
    for i in np.arange(len(c)):
        cashflow.append(c[i]*np.exp(-y[i]*t[i])) #计算每期现金流现值并放入列表
    return np.sum(cashflow)
```

对于债券定价公式，最核心的变量就是不同期限的零息利率yt，不同期限的零息利率在金融市场中并非可以直接观察到，通常需要运用带票息的债券市场价格推算出不同期限的零息利率。

7.4.5 通过票息剥离法计算零息利率

【例7-7】假定在2019年1月2日，债券市场上分别有剩余期限是0.25年（3个月期）、0.5年（半年期）、1年期、1.5年期和2年期的5只国债，具体债券的票面利率和债券价格信息见表7-5，基于这5只债券价格的信息运用票息剥离法计算对应期限的零息利率。

序号	剩余期限	票面利率	债券价格	本金
债券1	0.25	0	99.42元	100元
债券2	0.5	0	98.83元	100元
债券3	1.0	2.77% （每年付息1次）	100.09元	100元
债券4	1.5	3.46% （每年付息2次）	101.32元	100元
债券5	2.0	2.53% （每年付息2次）	99.39元	100元

7.4.5 通过票息剥离法计算零息利率

由于期限是0.25年、0.5年的国债无票息，因此很容易计算对应于这两只债券期限的零息利率。

债券1的实质是99.42元的投资在3个月后变成100元，3个月的连续复利利率R1满足等式100=99.42e^R1*0.25，得到R1=2.3268%。

债券2的实质是98.83元的投资在6个月后变成100元，6个月的连续复利利率R2满足100=98.83e^R2*0.5,得到R2=2.3538%

……

针对剩余期限为1.5年的国债，计算会稍复杂一些，因为债券有3.46%的票面利率，并且票息是每年付息两次，这就意味着在0.5年和1年分别收到0.5×3.46%×100的票息，在1.5年末债券到期时会收到100*（1+0.5×3.46%）的本金和利息。因此，期限为1.5年的连续复利利率R4满足等式

$$100 \times [0.5 \times 3.46\% e^{-R_2 \times 0.5} + 0.5 \times 3.46\% e^{-R_3} + (1 + 0.5 \times 3.46\%) e^{-R_4 \times 1.5}] = 101.32$$

其中，前面已经计算得到了R2、R3的具体金额，所以可以得到R4=25411%

7.4.5 通过票息剥离法计算零息利率

第1步：通过 Python自定义一个包含联立方程组的函数，具体的代码如下：

```
def f(R):
    R1,R2,R3,R4,R5 = R        #设置不同期限利率
    P1 =99.42                 #0.25年期国债价格
    P2 =98.83                 #0.5年期国债价格
    P3 =100.09                #1年期国债价格
    P4 =101.32                #1.5年期国债价格
    P5 =99.39                 #2年期国债价格
    par =100.0                #债券面值
    C3 =0.0277                #1年期国债的票面利率
    C4 =0.0346                #1.5年期国债的票面利率
    C5 =0.0253                #2年期国债的票面利率
    bond1=P1*np.exp(R1*0.25)-par    #第1只债券计算零息收益率的公式
    bond2=P2*np.exp(R2*0.5)-par     #第2只债券计算零息收益率的公式
    bond3=P3*np.exp(R3*1.0)-par*(1+C3)   #第3只债券计算零息收益率的公式
    bond4=par*(0.5*C4*np.exp(-R2*0.5)+0.5*C4*np.exp(-R3)+(1+0.5*C4)*
            np.exp(-R4*1.5))-P4   #第4只债券计算零息收益率的公式
    bond5=par*(0.5*C5*np.exp(-R2*0.5)+0.5*C5*np.exp(-R3)+0.5*C5
            *np.exp(-R4*1.5)+(1+0.5*C5)*np.exp(-R5*2))-P5   #第4只债券计算零息收益率的公
式
    return np.array([bond1,bond2,bond3,bond4,bond5])
```

7.4.5 通过票息剥离法计算零息利率

第2步：运用到 SciPy的子模块 optimize中的函数 fsolve，用于求解第1步中的联立方程式，具体的代码如下：

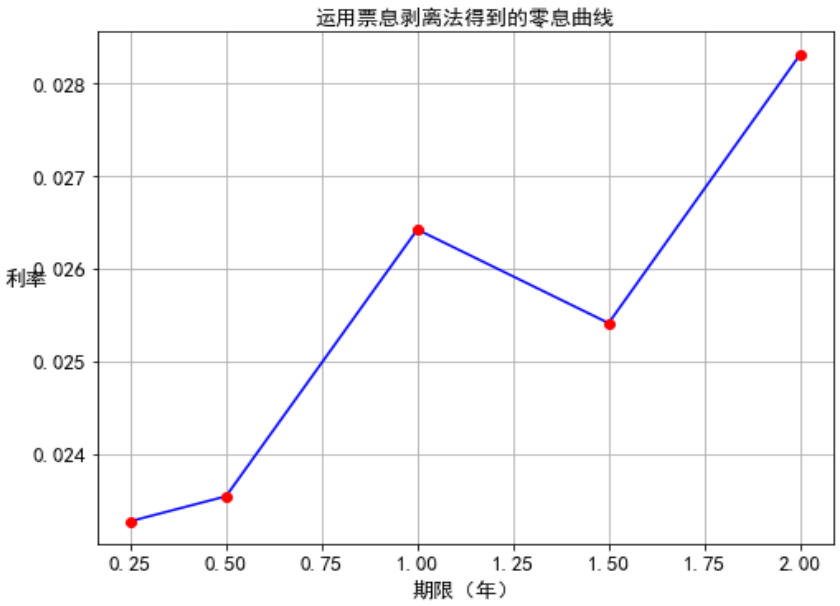
```
import scipy.optimize as so            #导入SciPy的子模块optimize
Zero_rates=so.fsolve(f,[0.1,0.1,0.1,0.1,0.1])
print('0.25年期零息利率（连续复利）',round(Zero_rates[0],6))
print('0.5年期零息利率（连续复利）',round(Zero_rates[1],6))
print('1年期零息利率（连续复利）',round(Zero_rates[2],6))
print('1.5年期零息利率（连续复利）',round(Zero_rates[3],6))
print('2年期零息利率（连续复利）',round(Zero_rates[4],6))
```

0.25年期零息利率（连续复利） 0.023268
0.5年期零息利率（连续复利） 0.023538
1年期零息利率（连续复利） 0.026424
1.5年期零息利率（连续复利） 0.025411
2年期零息利率（连续复利） 0.028313

7.4.5 通过票息剥离法计算零息利率

第3步：对计算得到的零息利率进行可视化（见下图），具体的代码如下：

```
T =np.array([0.25,0.5,1.0,1.5,2.0])    #生成包含五只国债期限的数组
plt.figure(figsize=(8,6))
plt.plot(T,Zero_rates,'b-')
plt.plot(T,Zero_rates,'ro')
plt.xlabel(u'期限（年）',fontsize =13)
plt.xticks(fontsize=13)
plt.ylabel(u'利率',fontsize =13,rotation=0)
plt.yticks(fontsize=13)
plt.title(u'运用票息剥离法得到的零息曲线',fontsize=13)
plt.grid('True')
plt.show()
```



7.4.5 通过票息剥离法计算零息利率

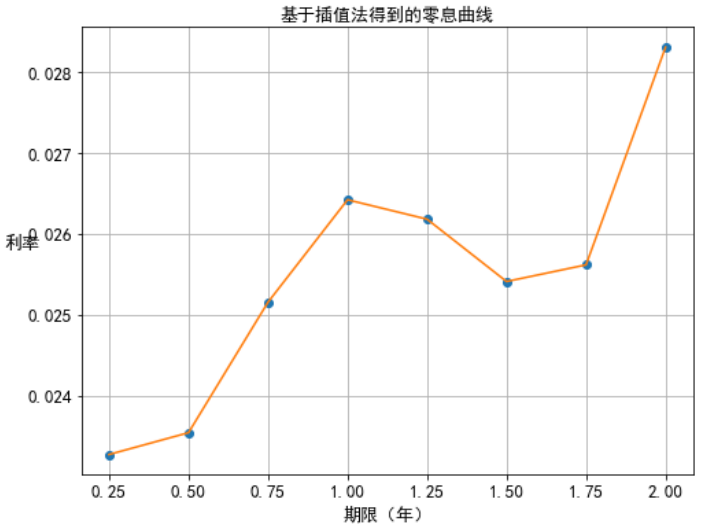
下面运用例7-7计算得到的零息利率并且运用插值法计算期限0.75年、1.25年和1.75年零息利率，具体分3个步骤展开。

第1步：选择插值的具体方法并且进行相应的计算，相关的代码如下：

```
import scipy.interpolate as si
func =si.interp1d(T,Zero_rates,kind='cubic')  #运用原有的数据构建一个插值函数，并运用3阶样条曲线插值法
T_new=np.array([0.25,0.5,0.75,1.0,1.25,1.5,1.75,2.0])  #生成包含0.75年、1.25年和1.75年的新数据
Zero_rates_new =func(T_new)          #计算得到基于插值法的零息利率
```

第2步：对计算得到的结果进行可视化（见下图），相关的代码如下：

```
plt.figure(figsize=(8,6))
plt.plot(T_new,Zero_rates_new,'o')
plt.plot(T_new,Zero_rates_new,'-')
plt.xlabel(u'期限（年）',fontsize =13)
plt.xticks(fontsize=13)
plt.ylabel(u'利率',fontsize =13,rotation=0)
plt.yticks(fontsize=13)
plt.title(u'基于插值法得到的零息曲线',fontsize=13)
plt.grid('True')
plt.show()
```



7.4.5 通过票息剥离法计算零息利率

第3步：运用for语句对全部结果进行输出，具体的代码如下

```
for i in range(len(T_new)):
   print('期限（年）',T_new[i],'零息利率',round(Zero_rates_new[i],6))
```

```
期限（年） 0.25 零息利率 0.023268
期限（年） 0.5 零息利率 0.023538
期限（年） 0.75 零息利率 0.025155
期限（年） 1.0 零息利率 0.026424
期限（年） 1.25 零息利率 0.026184
期限（年） 1.5 零息利率 0.025411
期限（年） 1.75 零息利率 0.025617
期限（年） 2.0 零息利率 0.028313
```

通过插值法，就得到了0.75年期限的零息利率是2.5155%，1.25年期限的零息利率是2.6184%，1.75年期限的零息利率是2.5617%。

7.4.6 运用零息利率对债券定价

【例7-8】假定在2019年1月2日，债券市场有一个债券，本金为100元，票面利率为6%，每年支付票息4次（每季度支付1.5%），剩余期限为2年，运用例7-7计算得到的零息利率曲线（见下表）对该债券进行定价。

期限（年）	零息利率
0.25	2.3268%
0.5	2.3538%
0.75	2.5155%
1.0	2.6424%
1.25	2.6184%
1.5	2.5411%
1.75	2.5617%
2.0	2.8313%

根据以上的信息，计算第0.25年支付的1.5元票息的现值，就用对应于0.25年期的零息利率2.3268%进行贴现；计算第0.5年支付的1.5元票息的现值，就用对应于0.5年期的零息利率2.3538%进行贴现；依此类推。因此，债券的价格就是如下的表达式:

$$1.5e^{-2.3268\%\times0.25}+1.5e^{-2.3538\%\times0.5}+1.5e^{-2.5155\%\times0.75}+1.5e^{-2.6424\%}+1.5e^{-2.6184\%\times1.25}+1.5e^{-2.5411\%\times1.5}+1.5e^{-2.5617\%\times1.75}+(100+1.5)e^{-2.8313\%\times2}=106.1486\text{（元）}$$

7.4.6 运用零息利率对债券定价

第1步：输入债券的相关参数信息，具体的代码如下：

```
coupon =0.06        #债券票面利率
m =4                #债券票面利率的复利频次
par=100             #债券面值
bond_cashflow=np.ones_like(T_new)*par*coupon/m     #生成存放债券票息的数组
bond_cashflow[-1]=par*(1+coupon/m)      #将债券本金和最后一期票息加入数组
```

第2步：运用函数 Bond_value计算债券价格并输出结果，具体的代码如下

```
bond_price=Bond_value(c=bond_cashflow,t=T_new,y=Zero_rates_new)  #输入债券的相关信息
print('计算的债券价格',round(bond_price,4))
```

计算的债券价格 106.1486

7.6 久期

7.6.1 麦考利久期

首先来看数学表达式，很定债券在ti时刻提供给债券持有人的现金流用ci表示,其中1≤i≤n。债券价格B与连续复利的到期收益率y之间的关系式用以下式子示：

$$B = \sum_{i=1}^{n} c_i e^{-yt_i} \qquad \textbf{（7-15）}$$

式子（7-15）就表示债券价格等于所有将来支付的现金流现值之和。

麦考利久期D的表达式如下：

$$D = \frac{\sum_{i=1}^{n} t_i c_i e^{-yt_i}}{B} = \sum_{i=1}^{n} t_i \left(\frac{c_i e^{-yt_i}}{B}\right) \qquad \textbf{（7-16）}$$

可以得到麦考利久期的一些重要结论：在其他条件相同的情况下（1）当债券的期限拉长，久期会增加；（2）当票面利率下降，久期也会增加。关于第二点可以举两个典型债券作为例子，一个债券是T年期零息债券，该债券的久期就等于T，另债券是T年期带票息的债券，该债券的久期是小于T年。

7.6.1 麦考利久期

接着，用 Python自定义计算债券麦考利久期的函数，具体的代码如下：

```
def M_Duration(c,y,t):
    '''构建一个计算麦考利久期的函数
    c：表示债券存续期内的现金流，用数组（ndarray)的数据结构输入；
    y：表示债券的到期收益率（连续复利）；
    t：表示对应于产生现金流的时刻，用数组（ndarray）的数据结构输入。'''
    cashflow=[]     #建立存放债券每一期现金流现值的列表
    weight=[]       #建立存放在债券每一期现金流现值与债券价格比率的列表
    n = len(t)
    for i in np.arange(n):
        cashflow.append(c[i]*np.exp(-y*t[i]))  #计算得到债券每一期现金流现值的列表
    for i in np.arange(n):
        weight.append(cashflow[i]/sum(cashflow))  #计算得到每一期现金流现值与债券价格比率的列表
    duration=np.sum(t*weight)  #计算得到债券麦考利久期
    return duration          #输出债券的麦考利久期
```

7.6.1 麦考利久期

【例7-12】在2019年1月17日，针对发债主体是中国国家开发银行的债券“13国开09”，该债券剩余期限为4年，面值为100元，票面利率为2.95%，票息支付是每年2次（半年1次），到期收益率为3.8%（连续复利）利用式子（7-16），计算该债券的麦考利久期。

期限	现金流（元）	现金流现值（元）	权重	期限*权重
0.5	1.475	1475e^(-0.5*3.8%) = 1.4472	1.4472/96.7421= 1.496%	0.0075
1.0	1.475	1.475e^(-1.0*3.8%)=1.42	1.42/96.7421=1.42	0.0147
1.5	1.475	1.475e^(-1.5x3.8%)=1.3933	1.3933/96.7421=1.4402%	0.0216
2.0	1.475	1.475e^(-2.0x3.8%)=1.3671	1.3671/96.7421=1.4131%	0.0283
2.5	1.475	1.475e^(-2.5x3.8%)=1.3413	1.3413/96.7421 =1.3865%	0.0347
3.0	1.475	1.475e^(-3.0x3.8%)=1.3161	1.3161/96.7421 =1.3604%	0.0408
3.5	1.475	1.475e^(-3.5x3.8%)=1.2913	1.2913/96.7421=1.3348%	0.0467
4.0	101.475	101,475e^(-4.0x3.8%)=87.1658	87.1658/96.7421=90.1012%	3.604
合计		96.7421（债券价格）	100%	3.7983

7.6.1 麦考利久期

下面，运用前面通过Python自定义计算债券麦考利久期的函数 M_Duration来计算例7-10的麦考利久期，具体分为两个步骤。

第1步：输入债券的相关参数，具体的代码如下：

```
coupon =0.0295                  #债券票面利率
par=100                         #债券面值
bond_yield=0.038                #债券到期收益率（连续复利）
t_list=np.arange(1,9)/2         #快速生成现金流期限的数组
cashflow=np.ones_like(t_list)*coupon*0.5*par  #生成一个不包括到期本金的现金流数组
cashflow[-1]=par*(1+coupon*0.5)   #将本金纳入现金流的数组
```

第2步：运用函数M_Duration进行计算，具体的代码如下:

```
Duration=M_Duration(c=cashflow,y=bond_yield,t=t_list)
print('13国开09债券的麦考利久期',round(Duration,4))
```

13国开09债券的麦考利久期 3.7983

7.6.2 修正久期

在前面讨论麦考利久期的时候，一个很重要的前提条件是债券的到期收益率y是连续复利。

现在假定当债券到期收益率y是每年复利1次，那么久期关系式就变为：

$$\Delta B = -\frac{BD\Delta \mathrm{y}}{1+\mathrm{y}}$$

$$D^* = \frac{D}{1+\mathrm{y/m}} \qquad \textbf{（7-21）}$$

就称为债券的修正久期（ modified duration）：

$$\Delta B = -BD^*\Delta y \qquad \textbf{（7-22）}$$

值得注意的是，在连续复利中，麦考利久期和修正久期是一样的。但现实生活中，连续复利的假设并不常见。

7.6.2 修正久期

```
def Modi_Duration(c,y,m,t):
    '''构建一个计算修正久期的函数
    c：表示债券存续期内现金流，用数组（ndarray）的数据结构输入；
    y：表示债券的到期收益率，复利频次是m次；
    m：表示复利频次；
    t：表示对应于产生现金流的时刻，用数组的数据结构输入。'''
    cashflow=[]     #建立存放债券每一期现金流现值的列表
    weight=[]       #建立存放在债券每一期现金流现值与债券价格比率的列表
    n = len(t)
    Rc=m*np.log(1+y/m)     #计算对应的连续复利的债券到期收益率
    for i in np.arange(n):
        cashflow.append(c[i]*np.exp(-Rc*t[i])) #计算得到债券每一期现金流现值的列表
    for i in np.arange(n):
        weight.append(cashflow[i]/sum(cashflow))  #计算得到每一期现金流现值与债券价格
比率的列表
    duration=np.sum(t*weight)  #计算得到债券麦考利久期
     return duration/(1+y/m)          #输出债券的麦考利久期
```

7.6.2 修正久期

【例7-14】沿用前面例7-12的“13国开09”债券，计算该债券的修正久期。同时假定当每年复利2次的债券到期收益率增加10个基点，分别运用债券的修正久期和债券定公式计算债券的最新价格，具体分为3个步骤。

第1步：计算债券的修正久期。由于例7-12已知的债券到期收益率3.8%是连续复利因此需要通过7.3节中讨论的式子（7-4）完成从连续复利到每年复利2次的转换。可以得到每年复利2次的收益率是3.8363%。同时已经得到麦考利久期是3.7983，根据式子（7-21）就可以得到修正久期

$$D^* = \frac{3.7983}{1 + 3.8363\%/2} = 3.7268$$

可以利用前面通过Python定义计算债券修正久期的函数Modi_Duration进行求解，但同时需要先运用7.3节中通过Python定义将连续复利利率转为复利频次m次利率的函数Rm，将债券到期收益率38%（连续复利）转化为复利频次2次的收益率。具体的Python代码如下：

```
R2=Rm(Rc=bond_yield,m=2)  #将连续复利的债券到期收益率转化为每年复利2次的收益率
print('每年复利2次的债券到期收益率',round(R2,6))
```

每年复利2次的债券到期收益率 0.038363

```
Modified_Duration=Modi_Duration(c=cashflow,y=R2,m=2,t=t_list)  #计算债券修正久期
print('13国开09债券的修正久期',round(Modified_Duration,4))
```

13国开09债券的修正久期 3.7268

第2步：运用式子（7-22）计算当每年复利2次的收益率增加10个基点，也就是从3.8363%增加至3.8373%时，债券价格的变化金额

△B=-96.7421×3.7268×0.001=-0.3605（元)

因此，债券价格下降至96.7421-0.3605=96.3816元。

第3步：运用债券定价公式计算精确的结果。依然是直接运用在7.4节通过Python定义的计算债券价格函数 Bond_price接得出结果。注意该函数是假定债券到期收益率是连续复利的情况才能适用，所以，需要先运用7.3节中通过Python定义将复利频次m次利率转为连续复利利率的函数Rc，具体的 Python代码如下：

```
y_continous=Rc(Rm=(R2+0.001),m=2)  #将最新的每年复利2次的债券收益率转化为连续复利
print('连续复利的债券到期收益率',round(y_continous,6))
```

连续复利的债券到期收益率 0.038981

```
Bond=Bond_price(C=coupon,M=par,T=4,m=2,y=y_continous)  #输入13国开09债券的要素
print('运用债券定价公式计算得到13国开09债券的最新债券价格',round(Bond,4))
```

运用债券定价公式计算得到13国开09债券的最新债券价格 96.3824

7.7 凸性

7.7.1 凸性的表达式

凸性（convexity）也称“凸度”或“曲率”，是衡量债券价格对债券到期收益率变化的非线性关系，是债券价格对收益率的二阶导数，由斯坦利·蒂勒（Stanley Diller）于1984年最引入到债券分析中。它的数学计算公式是

$$C=\frac{1}{B}\frac{\mathrm{d}^2B}{\mathrm{d}y^2}=\frac{\sum_{i=1}^{n}c_i t_i^2 e^{-yt_i}}{B}=\sum_{i=1}^{n}t_i^2\left(\frac{c_i e^{-yt_i}}{B}\right) \qquad \textbf{（7-25）}$$

从式子（7-25）不难发现，凸性的实质就是债券支付现金流时间ti平方的加权平均数，而权重与计算久期的权重一致，即在ti时刻债券支付的现金流现值与债券价格比率。

```
def Convexity(c,y,t):
    '''构建一个计算债券凸性的函数
    c：表示债券存续期内现金流，用数组（ndarray）的数据结构输入；
    y：表示债券的到期收益率（连续复利）；
    t：表示对应于产生现金流的时刻，用数组的数据结构输入。'''
    cashflow=[]    #建立存放债券每一期现金流现值的列表
    weight=[]      #建立存放在债券每一期现金流现值与债券价格比率的列表
    n = len(t)
    for i in np.arange(n):
        cashflow.append(c[i]*np.exp(-y*t[i]))  #计算债券每一期现金流现值的列表
    bond_price=sum(cashflow)
    for i in np.arange(n):
        weight.append(cashflow[i]/bond_price) #计算每一期现金流现值与债券价格比率的列表
    convexity=np.sum(weight*t**2)      #计算得到债券凸性
return convexity       #输出债券的凸性
```

7.7.2 案例

【例7-17】依然沿用例7-12关于“13国开09”债券的信息，计算该债券的凸性。下表7-11显示计算“13国开09”债券凸性的完整计算过程。

期限	现金流（元）	现金流现值（元）	权重	期限的平方×权重
0.5	1.475	1.4472	1.496%	0.0037
1.0	1.475	1.42	1.4678%	0.0147
1.5	1.475	1.3933	1.4402%	0.0324
2.0	1.475	1.3671	1.4131%	0.0565
2.5	1.475	1.3413	1.3865%	0.0867
3.0	1.475	1.3161	1.3604%	0.1224
3.5	1.475	1.2913	1.3348%	0.1635
4.0	101.475	87.1658	90.1012%	14.4162
合计		96.7421（债券价格）	100%	14.8961

从表7-11的计算中，可以得到该债券的凸性是14.8961。下面，运用通过Python自定义计算债券凸性的函数Convexity计算本例题的债券凸性，具体的代码如下：

```
Bond_conv=Convexity(c=cashflow,y=bond_yield,t=t_list)  #输入债券要素
print('13国开09债券的凸性',round(Bond_conv,4))
```

13国开09债券的凸性 14.8961

7·7·3 重要关系式

利用泰勒展开式，针对债券价格变动与债券到期收益率变动之间有如下近似关系式:

$$\Delta B = \frac{\mathrm{d}B}{dy}\Delta y + \frac{1}{2}\frac{d^2B}{dy^2}(\Delta y)^2$$

$$\frac{\Delta B}{B} = -D^*\Delta y + \frac{1}{2}C(\Delta \mathrm{y})^2$$

$$\Delta B = -\mathrm{D}^* B\Delta \mathrm{y} + \frac{1}{2}CB(\Delta \mathrm{y})^2 \qquad \textbf{（7-28）}$$

显然，式子（7-28）比式子（7-22）更精确地衡量了债券到期收益率变动对债券价格的影响。

7.7.3 重要关系式

【例7-18】沿用前面例7-14的信息，也就是当“13国开09”债券连续复利的债券到期收益率增加10个基点，通过式子（7-28）计算债券的最新价格。其中，修正久期是3.7268，债券凸性是14.8961，债券原先的价格是96.7421元。

△B=-3.7268×96.7421×0.001+0.5×14.8961×96.7421×0.001^2=-3.7983（元）

因此，可以得到当债券到期收益率提高100个基点时，债券的最新价格96.7421-3.7983=96.3823（元）。

这一价格与利用债券定价公式（例7-14 ）计算得到的最新价格93.3824元之间仅仅相差0.0001元，说明引入了债券凸性以后，债券定价明显改善了。

Your appreciation makes me a miracle. Thank you!

Frank Ziwei Zhang
18117228563
frank8027@163.com

上海对外经贸大学
SHANGHAI UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS