第6回：物価連動国債のキャッシュフローと名目利回り (その1)

(1月26日 2-3節にゼロインフレ率の説明を追加)

第5回記事ではYTMベースの**Break-even Inflation** (以下 `BEI`)を算出した。今回はBEIでインフレが発生した場合 (つまり 前回算出した1.777%で、インフレが発生した場合) 、物価連動国債 (以下 `物国`) の名目利回りはベンチマーク国債の利回り 2.052%になるかを確認する。

この確認作業は、インフレ後の物国のキャッシュフローの作成と利回り算出となるが、この2つの作業は物国を理解する王道である。

かなり長い記事となり、我慢が必要となることを最初に断っておく。

1. Reference CPI (適用指数)、 Index Ratio (連動係数)

初めに物国のキャッシュフロー作成の下準備をする。

- 物国キャッシュフローの基礎となる**生鮮食品を除く全国消費者物価指数** (以下 `CPI`) は総務省から、前月分のデータが**毎月20日前後**に発表され、CPIの数値は 政府統計ポータルサイト e-Stat で確認できる
- 財務省は発表されたCPIを基に、 物価連動国債（１０年）の適用指数及び連動係数 という次のExcelを公表

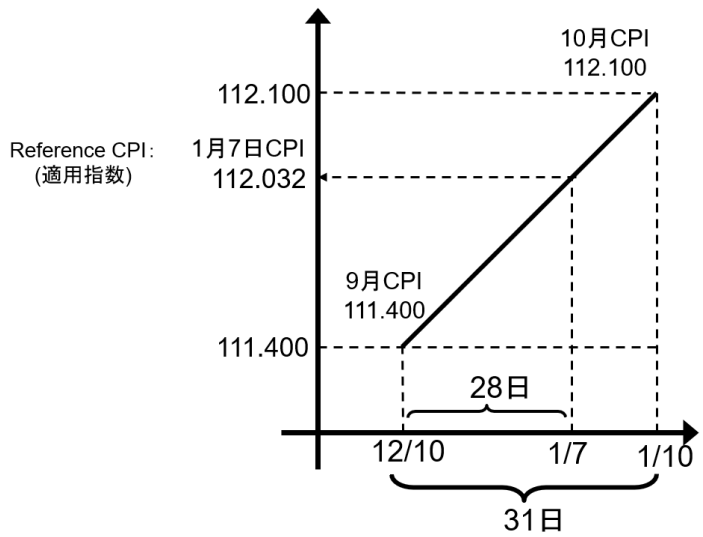
	A	B	G	H	I	J	K	L
1	令和7年12月10日～令和8年1月10日の適用指数・連動係数							
2			連動係数					
3		基準CPI／適用指数	第２５回	第２６回	第２７回	第２８回	第２９回	第３０回
4			102.2	101.1	100	104.1	106.4	109.6
5	2025/12/10	111.400	1.11412	1.12624	1.11400	1.07012	1.04699	1.01642
6	2025/12/11	111.423	1.11435	1.12648	1.11423	1.07035	1.04721	1.01663
7	2025/12/12	111.445	1.11457	1.12670	1.11445	1.07056	1.04742	1.01683
	(途中省略)							
29	2026/1/3	111.942	1.11954	1.13172	1.11942	1.07533	1.05209	1.02137
30	2026/1/4	111.965	1.11977	1.13195	1.11965	1.07555	1.05230	1.02158
31	2026/1/5	111.987	1.11999	1.13218	1.11987	1.07576	1.05251	1.02178
32	2026/1/6	112.010	1.12022	1.13241	1.12010	1.07598	1.05273	1.02199
33	2026/1/7	112.032	1.12044	1.13263	1.12032	1.07620	1.05293	1.02219
34	2026/1/8	112.055	1.12067	1.13286	1.12055	1.07642	1.05315	1.02240
35	2026/1/9	112.077	1.12089	1.13309	1.12077	1.07663	1.05336	1.02260
36	2026/1/10	112.100	1.12112	1.13332	1.12100	1.07685	1.05357	1.02281
37							(注:表示をコンパクトに編集した)	

まず、このExcelについて 説明するが、物国の取引日 / 受渡日は第5回記事同様 2026年1月6日 / 7日とする。重要な点は、**受渡日のCPI**がB33セルの112.032と算出されている点である。

1-1. Reference CPI (適用指数) `B33セル：112.032`

- 物国が参照するCPIは3ヶ月前のデータ。このことを「**3ヶ月ラグ**のCPI」などと言う

- `B5セル 111.40`：2025/12/10 `(A5セル)` の3ヶ月前の2025年9月10日のCPIが111.40を意味
- `B36セル 112.10`：同様に2026/1/10 `(A36セル)` の3ヶ月前の2025年10月10日のCPIが112.10
    - 本来のCPIの解釈にしたがえば、9月CPIが111.40、10月CPIが112.10であり、9月10日や10月10日という**各月10日**のCPIではない
    - たぶん、この `10日` は物国の利払日が10日なため、適当に財務省が決めたものか？
    - 欧米の標準仕様は月初の `1日` で、この違いがあとあと コーディングに影響
- `B6〜B35セル`：2025/12/11〜2026/1/9の各日付けのCPIは111.40と112.10を単純に線形補間した値
    - これら補間したCPIを **Reference CPI (適用指数)**と呼ぶ
    - 下は2026年1月7日のReference CPI計算のイメージ図



(図6-0：Reference CPI `適用指数`)

- この線形補間は次式のように 初めにウエイト w を算出し、

$$w = \frac{1/7 - 12/10}{1/10 - 12/10} = \frac{28日}{31日}$$

    - この w を利用して、

$$(1-w) \times 111.400 + w \times 112.100 = 112.032$$

        - `テキスト p.11 脚注4 式(1.3)` に線形補間の原型の式を記したが、その右辺が上のウエイトに対応
        - このウエイト算出で注意すべき点は2025/12/10〜2026/1/10を使うこと。3ヶ月ラグの期間 2025/9/10〜10/10ではない `(ここはよく間違える！)`

- 線形補間の式は使わないとすぐに忘れるので、メモしておく

$$y(x) = (1-w)\,y_0 + w\,y_1, \quad \text{ウエイト } w := \frac{x - x_0}{x_1 - x_0} \qquad (6\text{-}1)$$

- 同値な形： $y(x) = y_0 + \frac{x-x_0}{x_1-x_0}(y_1 - y_0)$ （テキスト p.11 脚注4 と同じ式）
- ウエイトは12/10からの経過日数に対応し、12/10では $w = 0$ 、1/10では $w = 1$

1-2. Base CPI (基準CPI) `L4セル：109.6`

- `G4～L4セル：102.2, ... , 109.6` 物国25回債～30回債が発行された時のCPI
  - CPIの改定があれば、このBase CPIの値は改定されるが、それ以外は不変
  - 図6-1の2行目では、baseCPIとせず、`発行時CPI` としての**issCPI**という変数名を使用 `(baseという用語が他の状況で使われるため)`

1-3. Index Ratio (連動係数) `L33セル：1.02219`

- Index Ratioは次式のように、Reference CPI ÷ Base CPIで算出
  - 発生したインフレの結果として、物国額面の増減を表す数値

$$\text{Index Ratio (連動係数)} = \frac{\overset{\text{(適用指数)}}{Ref.CPI}}{\underset{\text{(基本}CPI\text{)}}{BaseCPI}} = \frac{112.032}{109.6} = 1.02219 \qquad (6\text{-}2)$$

- 上式では インフレーションにより、額面が1.0から1.02219に増加

1-4. 2つのクリーン価格： `Real clean price` と `Clean price`

- 第5回 2節では市場で取引されている価格 `97.55` を**実質価格 (Real Clean Price)**と呼んだ
- 物国の**Clean Price** `（クリーン価格）` はIndex Ratioを使い、次式で算出する

$$\text{クリーン価格} = \text{Real Clean Price} \times \text{Index Ratio} \qquad (6\text{-}3)$$

- つまり、物国は2つの利回り(名目と実質)に対応し、2つのクリーン価格を持つ
- `実質価格` に対応し、クリーン価格を**名目価格**とも言う
- このクリーン価格に経過利息を加算したものがダーティー価格であり、受渡代金となる `(ダーティー価格は1つのみ)`

2. CPI指数オブジェクト、 インフレカーブ オブジェクト

Reference CPI や Base CPI は大した計算を行っていないが、ここまでの部分をQuantLibで計算するには、**インフレカーブを持ったCPI指数**のオブジェクト`(図6-1 10行目のcpiIX)`を作成し、そのメソッドで計算させる。

- インフレカーブとCPI指数の関係は、スワップカーブとIborの関係に対応
  - Iborのフォワードレートを計算する際、スワップカーブが必要と同様に、
  - CPI指数のフォワードを計算させるのに、インフレカーブが必要となる
    - この回の目標であるインフレ後の物国キャッシュフローを作成するには、CPI指数のフォワードの計算が必須

- 図6-1がコードと出力例 `(見慣れないキャメルケースは"myABBRモジュールで定義"と考えよう)`

```
# CPIデータ、日付の調整
issCPI,       stCPI,          enCPI,          =\
109.60,  (2025,9,111.40), (2025,10,112.10)

aTrdDT,       aStlDT                          =\
trdDT-ds9,  stlDT-ds9;     setEvDT(aTrdDT)

# CPI指数オブジェクト設定と関連数値の計算
infCvHDL = ql.RelinkableZeroInflationTermStructureHandle()
cpiIX    = ql.ZeroInflationIndex("Japan CPI",
                                 jpRegion, reviseF, freqM, lag0M, jpyFX, infCvHDL)
for yy, mm, xx in [stCPI, enCPI]: cpiIX.addFixing( jDT(yy,mm,1), xx )
enCpiDT = cpiIX.lastFixingDate()
refCPI  = round( ql.CPI.laggedFixing(cpiIX, aStlDT, lag3M, cpiLNR ),3)
ixRTO   = round( refCPI / issCPI , 5)
print(f'lastFixing DT:{enCpiDT.ISO()}, '
                                   f'refCPI:{refCPI:.8f}, ixRatio:{ixRTO:.8f}')
```

✓ 0.0s Python

```
lastFixing DT:2025-10-01, refCPI:112.03200000, ixRatio:1.02219000
```

(図6-1：CPIオブジェクトコード)

  - インフレ関連で新たに追加した`myABBRモジュール`の短縮形

```
# bond, CPI, クリーン価格等
parPR    = 100.0
parAMT   = 100.0
cpiLNR   = ql.CPI.Linear
cpiFLT   = ql.CPI.Flat
ds9      = 9
lag3M    = ql.Period('3M')
lag0M    = ql.Period('0M')
gwOLY    = False
reviseF  = False
jpRegion = ql.CustomRegion("Japan", "JP")
usRegion = ql.CustomRegion("USA",   "US")

def cP(prc): return ql.BondPrice(prc, ql.BondPrice.Clean)
def dP(prc): return ql.BondPrice(prc, ql.BondPrice.Dirty)
```

(図6-2：`myABBR`でのインフレ関連の短縮形)

2-1. Reference CPI（適用指数）、Index Ratio（連動係数）の算出

まず、図6-1の出力結果が第1節のReference CPI、Index Ratioと一致している点を確認しよう。それぞれの数値は14行、15行目で計算されている。

- `14行目 refCPI`：CPIクラス（C++の構造体）の静的メソッド`laggedFixing`で3ヶ月ラグでの2026年1月7日のReference CPIを算出（静的メソッドはテキスト p.7を参照）
  - 14行目 右辺先頭の`round`はCPI指数を少数3桁へラウンド（財務省ルール）
- `15行目 ixRTO`：Index Ratioを式(6-2)より、`refCPI`÷`issCPI`で算出
  - 同様に先頭の`round`でindex Ratioを少数5桁へラウンド（財務省ルール）

14行目の**laggedFixing**メソッドは最近追加されたようで、ドキュメントが間に合っていないが、**QuantLib.pyファイル**では次のようなコードである（QuantLib.pyファイルはテキスト p.7を参照）

```
23730    @staticmethod
23731    def laggedFixing(index, date, observationLag, interpolationType):
23732        r"""laggedFixing(ext::shared_ptr< ZeroInflationIndex > const & index,
         Date date, Period observationLag, CPI::InterpolationType
         interpolationType) -> Real"""
23733        return _QuantLib.CPI_laggedFixing(index, date, observationLag,
         interpolationType)
```

- 引数は4つで、（CPIインデックス、ターゲットの日付、3ヶ月ラグ、補間方法）
- 図6-1の14行目で与えた4つの変数から、これらの引数の内容を推測しよう。以下が簡単な説明（ds9, lag3M, cpiLNRはmyABBRの短縮形、図6-2参照）
  - `cpiIX`：10行目で作成されたCPI指数のオブジェクト
  - `aStlDT`：adjusted settle dateの略で5行目で作成。
    - 本来の受渡日（`stlDT`）2026年1月7日から9日（`ds9`）を引いた日付（dsはdaysの略）
  - `lag3M`：ラグ3ヶ月を意味する短縮形
    - ラグ0ヶ月の`lag0M`もmyABBRでは定義
  - `cpiLNR`：CPIの補間を線形に指定する短縮形（LNRはlinearの略）
    - １ヶ月単位でCPIが更新されるため、月内を線形補間させる
    - 補間計算を行わない`cpiFLT`もmyABBRで定義

2-2. ZeroInflationIndexクラスのCPI指数オブジェクト

では図6-1のコードを1行目から見て行く

- `2,3行目 初期値設定`：3ヶ月ラグで参照する9月と10月のCPIを`stCPI, enCPI`に設定`(st, en はstard, endの略)`
    - Quantlibでは、CPIデータは月初～月末とするため、`年、月`という日付で設定
- `5,6行目 ds9による調整`：毎月10日という日本のCPIの日付を欧米の標準仕様に修正するため、取引日と受渡日から9日`(ds9)`を引いて、`aTrdDT, aStlDT`に設定`(aはadjustedの略)`
    - `setEvDT(aTrdDT)`によって、QuantLibのシステム日付も9日前に修正
    - この修正法は、図6-0で示した「Reference CPI計算のイメージ図」の日付を9日ずらしても同じ計算となり、結果は正しい
        - 国債の場合、利払い日・償還日が曜日に関係なく、固定されているため、このようなシフトが可能
- `9,10,11行目 CPI指数オブジェクトの設定`：**ZeroInflationIndexコンストラクタ**によって、日本のCPI指数オブジェクト`cpiIX`を作成
    - コンストラクタの引数に関しては、QLPドキュメントの説明`(下図)`と10,11行目を見較べて、推測しよう`(QLPドキュメントはテキスト p.6を参照)`

    class **ZeroInflationIndex**(*familyName: str, region: ql.Region, revised: bool, frequency: ql.Frequency, availabilityLag: ql.Period, currency: ql.Currency, h: ql.ZeroInflationTermStructureHandle | None*)

    Base class for zero inflation indices.

    Parameters:
    - **familyName** (*str*) – The name of the zero inflation index family (e.g., “CPI”, “HICP”).
    - **region** (*ql.Region*) – The geographical region for which the index is published.
    - **revised** (*bool*) – Whether the index can be revised after publication.
    - **frequency** (*ql.Frequency*) – The frequency with which the index is published (e.g., Monthly, Quarterly).
    - **availabilityLag** (*ql.Period*) – The lag between the reference period and the publication date.
    - **currency** (*ql.Currency*) – The currency in which the index is quoted.
    - **h** (*Optional[ql.ZeroInflationTermStructureHandle]*) – (Optional) The zero inflation term structure handle used for forecasting.

    (若干の補足説明)

    - `第2引数 jpRegion`：myABBRモジュールで、CustomRegionコンストラクタを使い、`ql.CustomRegion("Japan", "JP")`と定義
    - `第3引数 revisedF`：myABBRモジュールで`False`を指定し、CPI指数が改定がされていないと設定

        - `第5引数 lag0M`：インデックス自体にラグは無いので、myABBRモジュールで、`ql.Period('0M')`を設定
            - QLPドキュメントの説明とは異なる。ここを`lag3M`にするとエラーで身動きできなくなる
        - `第7引数 infCvHDL`：9行目で設定した空のインフレカーブ用ハンドル
            - インフレカーブの設定は2-3節で行う
            - 空のハンドルの設定はTiborと同じで、`テキスト p.43の図2.3 7,8行目`を参照

- `12行目 cpiIX.addFixing`：9月と10月に確定したCPI値を`cpiIX`オブジェクトに登録
    - addFixingメソッドはTiborと同じで、`テキスト p.59 2.4.4節`を参照
    - `jDT(yy,mm,1)`と毎月1日を指定　`(ここを10日にしても、1日でフィクシングされる)`
- `13行目 lastFixingDate`：最後にフィクシングした日付を戻すメソッド
    - 12行目で9月と10月のCPI値をフィクシングさせたので、最後にフィクシングした日付は2025年10月1日

2-3. ゼロインフレ率 と インフレカーブ 設定

直前に説明した13行目`lastFixingDateメソッド`が意味することは「`cpiIX`オブジェクトは10月1日以降のインフレデータを持っていない」ことである。インフレカーブの設定は、その日付以降の**将来のインフレ率**を`cpiIX`に教える。

図6-3の2行目 **ZeroInflationCurve** はインフレカーブを作成する最も簡便なコンストラクタである。

- CPI指数を作成した ZeroInflationIndex やこの ZeroInflationCurve の名前の由来である**Zero coupon inflation rate**`(以下 ゼロインフレ率)`の定義は 式(6-4)の z_t となる。`(金利の場合も、ゼロクーポンレートをゼロレートと言う点は同じ)`
    - 基準日 b と時点 tのCPIをそれぞれ I_b、I_t 、[b, t] 間の年数を τ として、**ゼロインフレ率** z_t は次式より算出

$$I_t = I_b(1 + z_t)^\tau \tag{6-4}$$

    - I_bを左辺に移せば、式(6-2)のIndex Ratioが左辺に現れる。右辺はゼロインフレ率 z_tを使って、増減した物国額面が計算される

$$\frac{I_t}{I_b} = (1 + z_t)^\tau$$

- 図6-3では BEI 1.7770% を ゼロインフレ率に設定し、インフレカーブとするコード

- つまり、10月1日以降 物国額面は年率1.7770% で増加させる設定となる

```
# インフレカーブ                                    (BEIでフラットカーブを設定)
infCvOBJ = ql.ZeroInflationCurve( aStlDT,
                [enCpiDT,enCpiDT+pD('10y')], [beiRT,beiRT], freqM, dcA365)  #dcA365n
infCvHDL.linkTo(infCvOBJ)
# カーブ表示
dfInfCv  = pd.DataFrame( infCvOBJ.nodes(),columns=["node", "rate"] )
dfInfCv.node = dfInfCv.node.map(lambda x: x.ISO())
dfInfCv[:5].style.format({"rate": "{:.4%}"})
```

✓ 0.0s Python

	node	rate
0	2025-10-01	1.7770%
1	2035-10-01	1.7770%

(図6-3：インフレカーブ オブジェクト)

コードを簡単に説明しておく。

- `3行目` ：日付リスト `[enCpiDT,enCpiDT+pD('10y')]` とインフレ率リスト `[beiRT,beiRT]` を与えて、インフレカーブのオブジェクト `infCvOBJ` を作成
  - 日付リストでは、2025年10月1日から10年間を指定し、インフレ率は第5回記事で計算したbeiRT=1.7770%をフラットで設定
  - インフレカーブの日数計算法は `dcA365` とした　　`(dcA365nの可能性もある)`
- `5行目 infCvHDL.linkTo(infCvOBJ)` ：図6-1の9行目で作成した空のハンドル `infCvHDL` へinfCvOBJオブジェクトをリンク
  - `infCvHDL` ハンドルは `cpiIX` オブジェクト作成時の引数であった `(同図 11行目)`
  - このリンクにより、 `cpiIX` オブジェクトはフォワードCPIの算出が可能になる
- `6～8行 infCvOBJ.nodes` ： 作成された `infCvOBJ` オブジェクトは金利カーブと同様なnodesメソッドを持ち、nodes情報を出力 `(nodesの例は、テキスト 図2.3など 参照)`

3. 物国の名目利回りとオブジェクト

長い準備が終わり、図6-4の2～4行目で物国30回債用の**インフレで元本が増大する物国オブジェクト**の作成にたどり着けた。

```python
# inf bond object    >> スケジュールをds9で調整
iScdOBJ = ql.Schedule(effDT-ds9,matDT-ds9,pdFreqSA,calNL,unADJ,unADJ,dtGENb,EoMf)
iBndOBJ = ql.CPIBond (Tp1, parAMT, gwOLY, issCPI, lag3M, cpiIX,
                                                    cpiLNR, iScdOBJ, [cpnRT], dcA365n, unADJ)
# discount curve
dsCvOBJ, dsCvHDL = mu.ffTSH(aStlDT, nmlYLD, dcA365n, cmpdCMP, freqSA)
iBndOBJ.setPricingEngine(ql.DiscountingBondEngine(dsCvHDL))
# pricing
clnPRC  = iBndOBJ.cleanPrice()  ;        rClnPRC = clnPRC / ixRTO
accAMT  = iBndOBJ.accruedAmount()
infYLD  = iBndOBJ.bondYield(cP(clnPRC),  dcA365n, cmpdCMP, freqSA)
print(f'clnPRC:{clnPRC:.5f},  '      f'accAMT:{accAMT:.7f},  '
  f'nominalYLD:{infYLD:.7%},  '  f'realClnPRC:{rClnPRC:.7f} ')
```

✓ 0.0s Python

```
clnPRC:99.73804,  accAMT:0.0016663,  nominalYLD:2.0520002%,  realClnPRC:97.5728931
```

(図6-4：物国オブジェクトとプライシング)

3-1. 物国30回債のプライシング

図6-4では ベンチマーク国債の利回り`nmlYLD=2.052%`で元本が増大した物国30回債をプライシングした。ここで出力結果を確認する。

- 物国の名目金利 `nominalYLD` は 2.05200%とベンチマーク国債の利回りとほぼ一致
    - どのような計算から、名目利回り 2.052%が計算されたかは、3-2節で概説し、詳細は第7回記事で説明
- 実質価格(Real Clean Price)を表すrealClnPRC=97.5728931は第5回記事 2節の97.55と0.02円ほどのズレ
    - この実質価格は97.55に一致すべきであり、図6-3で設定したBEIのインフレカーブを修正する。この修正法は第7回記事で説明
    - BEIの1.777%はYTMベースであり、インフレカーブはゼロクーポンベースで設定しているため、この差異が発生
    - 物国30回債のクーポンが0.005%とゼロに近いため、物国の名目利回り2.052%とベンチマーク国債の利回りが近くなったが、本来は数ベーシスから数十ベーシスの差が生ずる

次にどのメソッドがこの計算結果を算出したかを確認する。`(使用されている債券のメソッドの詳細はテキスト4.2節を参照)`

- 9行目の `cleanPrice` メソッドは、名目価格(Clean Price)を戻し、99.73804円
- 実質価格は式(6-3)より、9行目 2つ目のコードで `rClnPRC=clnPRC/ixRTO` で手計算

- 9行目で計算された名目価格 `clnPRC` を11行目の `bondYield` メソッドに与え、名目利回り 2.052% を算出

3-2. CPIBond クラスとメソッド

では 図6-4のコードで残った部分を確認する。

- `2行 iScdOBJ` ： 第5回記事で使用した `mu.makeJGB関数` は使わず、 `Scheduleコンストラクタ` を手打ちし、物国30回債用のiScdOBJオブジェクトを作成 `(iはinflationの略)`
  - 注意すべきは、 `effDT` と `matDT` を `ds9` で調整させている
  - その他は `mu.makeJGB関数` のScheduleコンストラクタと同じ
- `3,4行 iBndOBJ` ：**CPIBondコンストラクタ**によって、物国30回債のiBndOBJを作成

  このコンストラクタの引数に関しては、QLPドキュメントの説明 (下図)と3, 4行目を見較べて、推測しよう

  > **ql.CPIBond**(*settlementDays, notional, growthOnly, baseCPI, contractObservationLag, inflationIndex, observationInterpolation, fixedSchedule, fixedRates, fixedDayCounter, fixedPaymentConvention*)

  - 3番目の引数 **growthOnly** は常にFalseとし、図6-2で `gwONY=False` を設定
  - その他の引数は自明であろう
- `6,7行` ：6行目でディスカウントカーブを作成し、7行目でiBndOBJオブジェクトにエンジンをセット
  - この手順は `テキスト 図4.11 (p.128)` と同じ
  - 注意すべきはディスカウントカーブの受渡日が `aStlDT` としている。つまりディスカウントカーブも9日間シフトさせている
- `テキスト 図4.11` ではNPVメソッドで利含み価格を計算させているが、エンジンを搭載した債券オブジェクトの場合、この図6-4の9,10行目のように `cleanPrice, accruedAmount, dirtyPrice` 等は引数無しで計算できる
  - `テキスト 図4.3 (p.115)` では、エンジンを持たない債券オブジェクトを使い、これらのメソッドを計算させた点と比較

3-3. 物国30回債のキャッシュフロー表 概観

図6-5は 物国30回債のiBndOBJオブジェクトを用い、2, 3行目で `mu.cpiBondCashFlow関数` から、キャッシュフロー表を出力させた。4行目ではキャッシュフロー表のamount列とDF列から、クリーン価格を手計算させ、図6-4と同じ価格 99.73804を得ている。

```python
# cash flows
dfBND = mu.cpiBondCashFlow(iBndOBJ, yts=dsCvOBJ, past=1)
display( dfBND.style.format(fmtFUT) )
HClnPRC = (dfBND.amount *dfBND.DF).sum() - accAMT
print(f'(hc) clnPRC:{HClnPRC:.5f}')
```

✓ 0.0s Python

	payDate	accruStart	accruEnd	cpi	coupon	amount	DF
0	2025-09-01	nan	nan	nan	nan%	nan	1.000000
1	2026-03-01	2025-09-01	2026-03-01	112.430475	0.0051%	0.0025	0.996538
2	2026-09-01	2026-03-01	2026-09-01	113.422284	0.0052%	0.0026	0.986335
3	2027-03-01	2026-09-01	2027-03-01	114.428365	0.0052%	0.0026	0.976400
			(途中省略)				
18	2034-09-01	2034-03-01	2034-09-01	130.598284	0.0060%	0.0030	0.837710
19	2035-03-01	2034-09-01	2035-03-01	131.756719	0.0060%	0.0030	0.829272
20	2035-03-01	nan	nan	nan	nan%	120.2160	0.829272

```
(hc) clnPRC:99.73804
```

(図6-5：物国30回債キャッシュフロー表)

このキャッシュフロー表から、まず 理解すべき点を列挙する。

- 最も重要な列が、キャッシュフロー表 中央にある**cpi列**。この列はBEI 1.7770%でインフレが発生した場合の将来のCPIを式(6-4)で計算した値で、1-1節の `Reference CPI(適用指数)` となる
- 左3列の日付は9日間シフトさせたもので、本来はすべてが `10日`
  - index20の物国30回債の債券償還日はpayDate列の日付を９日間シフトさせ、2035年3月10日
  - `1日` という表示が嫌いな場合、下のコードのように データフレーム上で9日を足して表示させれば済む

```python
# 日付を10日に戻す (dfBNDのコピーでの処理)
dfBND2 = dfBND.copy()
for cc in ['payDate', 'accruStart', 'accruEnd'] :
    dfBND2[cc] = pd.to_datetime(dfBND2[cc]) + pd.Timedelta(days=ds9)
    dfBND2[cc] = dfBND2[cc].dt.strftime("%Y-%m-%d")
dfBND2.style.format(fmtFUT)
```

- index20は物国30回債の償還価格が 120.2160 と算出
  - この償還価格は `Index Ratio(連動係数)` として、 `償還日CPI 131.756719 ÷ 発行時CPI 109.60` から計算

    - 各利払い日の増大した元本額は、cpi列の各利払い日CPIから `Index Ratio` を計算すれば済む
    - 各利払い額は `増大した元本額 x クーポン(0.005%)` で計算
- 図6-4で出力された `名目利回り(nominalYLD) 2.0520%` はpayDate列とamount列から計算した年2回複利利回り `(この手計算例は次回記事で説明)`

この第6回記事はかなり長くなったので、この辺りで一旦休憩し、残りは第7回記事で説明しよう。

- 上記コードはテキストのサポートページ より、取得可能。 `(ファイル名はinfBond-JP.ipynb, myABBR.py, myUtil.py)`