第5回：Break-even Inflation(BEI)の定義と計算例

`(財務省が2026年1月22日に発表したBEI値での検証を追記)`

今後3回の投稿でQuantLibを使った物価連動国債の分析例を紹介する。今回は通常国債の名目利回りと物価連動国債の実質利回りの差である**Break-even Inflation**(以下`BEI`)の計算法を説明する。

1. BEIの定義

この`利回りの差`であるBEIの別の表現は「通常国債と物価連動国債(以下`物国`)の投資利回りが等しくなるような**将来のインフレ率**」で、直感的には市場が織り込んだ**期待インフレ率**である

- BEIの理想的な計算はゼロレートz、同一満期Tの債券に対し、
  - 名目利回り$z_n(T)$、実質利回り$z_r(T)$とすると、次式で算出する

$$\mathrm{BEI}(T) = z_n(T) - z_r(T)$$

  - ただ、この定義ではゼロレートの算出に補間やブートストラップが必要
- 実務的にはゼロレートを債券満期までの複利利回りYTMで置換えた次式が使われる

$$\mathrm{BEI}(T) = \mathrm{YTM}_n(T) - \mathrm{YTM}_r(T) \tag{5-1}$$

- 注意点として、BEI と期待インフレ率は異なり、両者の関係は一般的に次式となる ```math \begin{aligned} \text{BEI} &= \text{期待インフレ率} \ &\qquad \qquad
- \underbrace{\text{インフレ リスクプレミアム}}_{\text{Risk Premium}}
- \underbrace{\text{流動性プレミアム}}_{\text{Liquidity Premium}} \end{aligned} ```
- 右辺の2，3項目の計測はかなり困難で、筆者は`BEI ≒ 期待インフレ率`と見做している。ただし次の点を忘れないように！
  - 物価連動債は流動性が低く、ショック時に BEI は急低下する
  - この現象は、期待インフレ率の低下ではなく、流動性プレミアムの変動となり、`BEI ≠ 期待インフレ率`となる

2. 物国30回債のBEI計算例

- では 計算例を示そう。下図はJSDAが発表した2026年1月6日引けのマーケットデータ。(左上の2026年1月7日とは翌日に発表する日付を意味)
  - 867行目の物国30回債のクーポンが `*` 印で表示されているが、正しくは0.005%
  - 164行目がベンチマークの378回債

	A	B	C	D	E	F	G	H	I
25	**2026年1月7日 (水)　発表**								
26									
27	銘柄種別 Issue Type	銘柄コード Code	銘柄名 Issues	償還期日 Due Date	利率 Coupon Rate	平均値 Average			
28						単価 Price(Yen)	前日比(銭) Change(0.01Yen)	複利利回り(%) Compound Yield	単利利回り(%) Simple Yield
64									
164	02	003780067	長期国債 378	2035/03/20	1.4	94.56	-10	2.051	2.105
857									
867	05	000300083	物価連動国債 30	2035/03/10	*	97.55	+25	———	———

(図5-1：JSDA 2026年1月6日引けデータ)

- 第2回記事 JGBクラスの日本式複利メソッドの実装 で伝えたように、JSDAはT+0で日本式複利を計算し、2.051%

- 下のセルでは、第2回記事の復習を兼ね、T+0で378回債の価格94.56から日本式と欧米式複利を計算した

```
1  from myABBR import * ; import myUtil as mu
2
3  jb378 = mu.makeJGB(jDT(2025,9,20),jDT(2035,3,20),0.014)
4  trdDT = jDT(2026,1,5)
5  setEvDT(trdDT) ; print(f'EvDT({getEvDT().ISO()}', end=') ')
6  print(f'jpnCmpYld: {jb378.JapanCompoundYield(94.56):.5%}', end=', ')
7  print( 'stdCmpYld: {:.5%}'.format(
8                  jb378.bondYield(cP(94.56),dcA365n, cmpdCMP, freqSA)))
```

✓ 1.3s　Python

```
EvDT(2026-01-05) jpnCmpYld: 2.05190%, stdCmpYld: 2.05179%
```

  - 3行目のmakeJGB関数は第2回 4節の説明を参照
  - この関数の引数は最低3つ必要となり、`(378回債の前回利払日、満期日、クーポン)` で構成
    - 5行目 `getEvDT関数` はQuantLibのシステム日付を取得する関数として、新たにmyABBRモジュールに追加 `(これでsetter, getterが揃った)`
    - 第2回記事でも記したが、日本式複利と欧米式複利の差は大きくない。問題がない場合、筆者の投稿はQuantLib標準の複利メソッドを使用する

- T+0は邪魔なので、次のセルは通常のT+1で複利を算出させた

  - 2行目のsetEvDT関数で、取引日 `(trdDTはtrade dateの略)` を1月6日にセットし直し、複利利回り 2.052%を算出

```python
trdDT = jDT(2026,1,6) ;          stlDT = calJP.advance(trdDT,Tp1,DD)
setEvDT(trdDT) ; print(f'EvDT({getEvDT().ISO()}', end=') ')
print(f'jpnCmpYld: {jb378.JapanCompoundYield(94.56):.5%}', end=', ')
print( 'stdCmpYld: {:.5%}'.format(
                    jb378.bondYield(cP(94.56),dcA365n, cmpdCMP, freqSA)))
```

✓ 0.0s Python

```
EvDT(2026-01-06) jpnCmpYld: 2.05209%, stdCmpYld: 2.05197%
```

- これらを前提に次のセルで **BEI 1.7770%** を算出
  - 上で算出した各複利利回りは第2回記事の復習であり、本題からはズレている
  - BEIの計算は下のたった10行のコードのみで完了する

```python
# BEI計算: 初期値  (effDT:前回利払日   nmlYLD: ベンチマーク国債利回り)
effDT,               matDT,       cpnRT,   rBndPRC,  nmlYLD         =\
jDT(2025,9,10), jDT(2035,3,10), 0.005/100,  97.55,  2.052/100

# real bond object
rBndOBJ = mu.makeJGB(effDT,matDT,cpnRT)
# real yield
realYLD = rD(rBndOBJ.bondYield(cP(rBndPRC), dcA365n, cmpdCMP, freqSA),5)
beiRT   = nmlYLD -realYLD
print(f'real YLD:{realYLD:.5%}, (BEI)Break-even Infl.: {beiRT:.5%}')
```

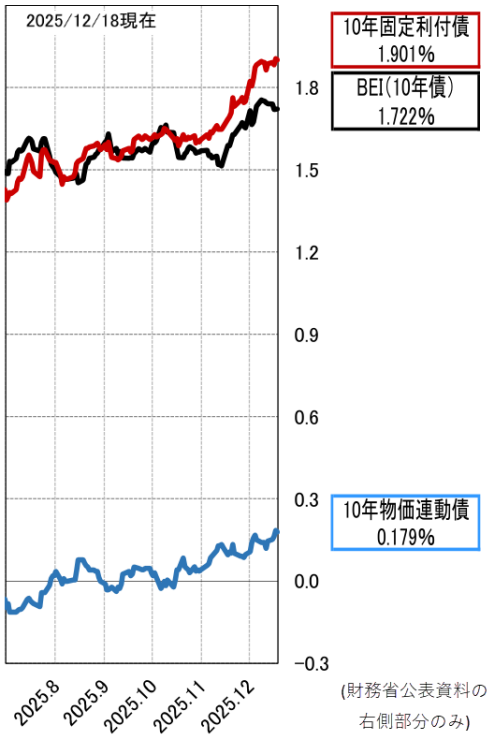
✓ 0.0s Python

```
real YLD:0.27500%, (BEI)Break-even Infl.: 1.77700%
```

(図5-2：実質利回り債オブジェクトとBEI算出)

  - 日本の財務省は「ブレーク・イーブン・インフレ率の推移」を物価連動国債のWebで発表しているが、2025/12/18現在で1.722%と公表し、ほぼ同じ水準 (注：評価日が異な

る）



- では上のコードを概観する
  - 2,3行目は初期値の設定
    - `effDT`, `matDT` は物国30回債の前回利払日と満期日
    - `rBndPRC`, `nmlYld` は それぞれreal yield bond price `（実質利回り債の価格）`, nominal yieldの略
  - 6行目は `makeJGB関数` を使い、**インフレで元本が増減しない実質利回り債**のオブジェクト `rBndOBJ` を物国30回債の条件から設定
    - `rBndOBJ` とはクーポン0.005%、償還100円の国債
  - 8行目はJSDAの価格 97.55からYTM利回り 0.2750% を算出
    - `cP(...)関数` はクリーンプライスを意味し、myABBRモジュールで定義
    - この97.55は実質利回り債の価格なので、**Real clean price** `（実質価格）` と呼ばれる
  - 9行目は YTMベースの式(1)を使い、BEI 1.777%を算出
    - このBEI値は第6回記事で利用するので、変数 `beiRT` に保存

QuantLibを使えば、BEIは驚くほど簡単に計算できることが判ったと思う。

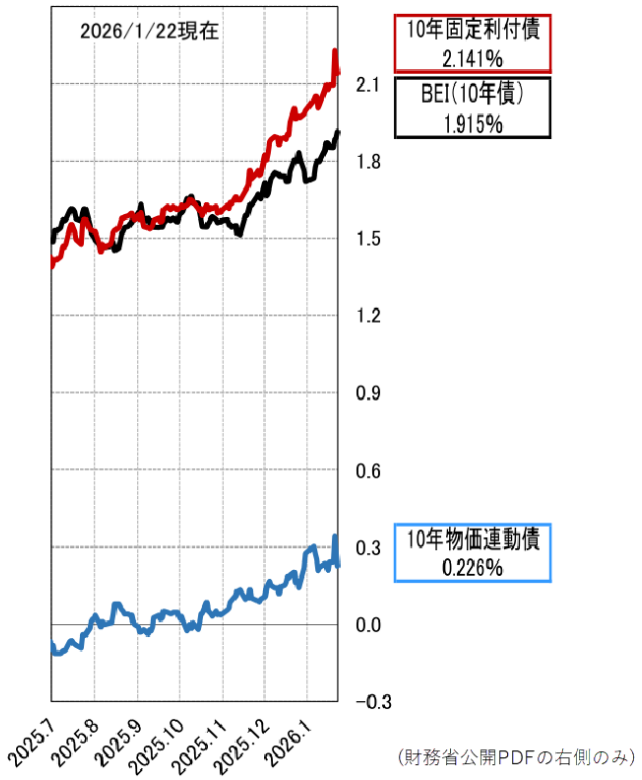
まとめ

- BEIを定義し、YTMベースのBEI算出のコード例を示した
- そのコードでは**実質利回り債**を第2回記事で作成したmakeJGB関数で作成

- 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)`
  - `(ファイル名はinfBond-JP.ipynbへ修正し、第6回コードが含まれる`

追記

- 財務省のWeb ブレーク・イーブン・インフレ率の推移は2026年1月22日付のBEI値を下図ように**1.915%**と発表



(財務省公開PDFの右側のみ)

- 次図がJSDAデータ(T+1ベース)で算出したBEI値で**1.916%**

- 0.1bp ズレている理由は、財務省の価格ソースが日本相互証券で異なるため

	A	B	C	D	E	F	G	H	I
25	2026年1月23日 (金)　発表								
26									
27	銘柄種別 Issue Type	銘柄コード Code	銘柄名 Issues	償還期日 Due Date	利率 Coupon Rate	平均値 Average			
28						単価 Price(Yen)	前日比(銭) Change(0.01Yen)	複利利回り(%) Compound Yield	単利利回り(%) Simple Yield
62									
164	02	003780067	長期国債 378	2035/03/20	1.4	93.86	+32	2.142	2.205
864									
874	05	000300083	物価連動国債 30	2035/03/10	*	98.00	+60	———	———

```python
from myABBR import * ; import myUtil as mu

jb378 = mu.makeJGB(jDT(2025,9,20),jDT(2035,3,20),0.014)

trdDT = jDT(2026,1,22) ;          stlDT = calJP.advance(trdDT,Tp1,DD)
setEvDT(trdDT) ; print(f'EvDT({getEvDT().ISO()}', end=') ')
print(f'jpnCmpYld: {jb378.JapanCompoundYield(93.86):.5%}', end=', ')
print( 'stdCmpYld: {:.5%}'.format(
                    jb378.bondYield(cP(93.86),dcA365n, cmpdCMP, freqSA)))
```

Python

```
EvDT(2026-01-22) jpnCmpYld: 2.14228%, stdCmpYld: 2.14218%
```

```python
# BEI計算: 初期値  (effDT:前回利払日    nmlYLD: ベンチマーク国債利回り)
effDT,                matDT,        cpnRT,    rBndPRC,  nmlYLD          =\
jDT(2025,9,10), jDT(2035,3,10), 0.005/100,  98.00,  2.142/100

# real bond object
rBndOBJ = mu.makeJGB(effDT,matDT,cpnRT)
# real yield
realYLD = rD(rBndOBJ.bondYield(cP(rBndPRC), dcA365n, cmpdCMP, freqSA),5)
beiRT   = nmlYLD -realYLD
print(f'real YLD:{realYLD:.5%}, (BEI)Break-even Infl.: {beiRT:.5%}')
```

Python

```
real YLD:0.22600%, (BEI)Break-even Infl.: 1.91600%
```