引き継ぎ資料 Vol.6

全曲率の話

2016/09/??

コンセプト

ソースとテクニックを同時に話そう

目次

1. 概要
2. 軌道の3次元関数
3. 2次元化
4. S字の抽出

目次

1. 概要
2. 軌道の3次元関数
3. 2次元化
4. S字の抽出

やりたいこと

3次元軌道からS字を抽出

やりかた

- モーキャプデータをB-splineとして近似
- 軌道を投影し2次元化
- S字の抽出

やりかた

- (モーキャプデータをB-splineとして近似)
- 軌道を投影し2次元化
- S字の抽出

やりかた

- モーキャプデータをB-splineとして近似
- 軌道を投影し2次元化
- S字の抽出
    - 変曲点の計算
    - 全曲率が一定となるように取り出し

目次

1. 概要

2. 軌道の3次元関数

3. 2次元化

4. S字の抽出

B-spline曲線

B-spline曲線の詳細は省略

$$\mathbf{B}(t)(0 \leq t \leq 1) \tag{1}$$

$\mathbf{B}(t)$はtを媒介変数とする3次元曲線

実装的な話

- Bは関数名
- 3次元の座標を返す

```
>>> B(0.)
array([0., 0., 0.])
```

目次

1. 概要
2. 軌道の3次元関数
3. 2次元化
4. S字の抽出

理論

$$\mathbf{P}(t) = M \cdot \mathbf{B}(t) \tag{2}$$

Mが2×3の行列なら$\mathbf{P}(t)$は2次元軌道

実装

クラスを一つ作成

```
class SecondDimensionalize:
    def __init__(self, func):
        self.mat = np.array(
            [[1.0, 0.0, 0.0],
             [0.0, 1.0, 0.0]]
        )
        self.func = func

    def __call__(self, t):
        return np.dot(self.mat, self.func(t))
```

実装

見かけ上関数に見えるオブジェクト

```
>>> p = SecondDimensionalize(B)
>>> p(0.)
array([0., 0.])
```

なぜ見かけ上関数に見えるオブジェクトが必要?

なぜ見かけ上関数に見えるオブジェクトが必要?

S字の抽出過程で使いたい

目次

1. 概要
2. 軌道の3次元関数
3. 2次元化
4. S字の抽出

変曲点

変曲点 曲線の曲率=0となる点

曲率

曲率$\kappa(t)$の計算法

$$\kappa(t) = \frac{\begin{vmatrix} \mathbf{P'}(t) \\ \mathbf{P''}(t) \end{vmatrix}}{|\mathbf{P'}(t)|^3} \tag{3}$$

実装

```python
class Curvature:
    def __init__(self, func):
        self._func = func
        self._d = func.diff()
        self._dd = self._d.diff()

    def __call__(self, t):
        d  = self._d(t)
        dd = self._dd(t)
        return ((d[0] * dd[1] - d[1] * dd[0]) /
            ((d[0] ** 2 + d[1] ** 2) ** 1.5))
```

実装

diff関数が必要なので追記

```
class SecondDimensionalize:
.
.
.
    def diff(self):
        return (
            SecondDimensionalize(self.func.diff())
        )
```

実装

```
>>> p = SecondDimensionalize(B)
>>> p(0.)
array([0., 0.])
>>> c = Curvature(p)
>>> c(0.)
3.5
```

2次元軌道pとその曲率cがそれぞれ関数として使用可能

実装

```
>>> p = SecondDimensionalize(B)
>>> p(0.)
array([0., 0.])
>>> c = Curvature(p)
>>> c(0.)
3.5
```

2次元軌道pとその曲率cがそれぞれ関数として使用可能

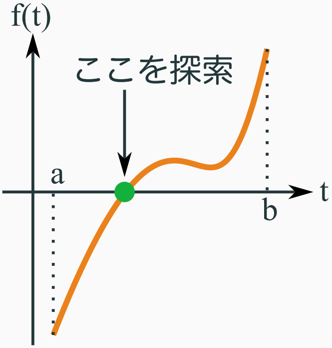
変曲点 = cが0となる点

変曲点の探し方

scipy.optimize.brentq関数は関数が0になる点を探索

brentq(f, a, b)

- **f** 関数
- **a** 探索範囲の左端
- **b** 探索範囲の右端



brentqの注意点

- a,bの間に解がなければならない
- 複数の解があった場合一つしか探索できない
- 解はわずかな誤差を含む

曲率・全曲率

曲率 軌道の曲がり具合を表す

$$\kappa(t) = \frac{1}{r} \tag{4}$$

軌道の回転が強い(rが小さいほど)大きな値

全曲率 軌道がどれだけ曲がったかを示す

$$\mu(t) = \int_a^b \kappa(s) ds \tag{5}$$

ここでsは軌道長

例: 円の場合

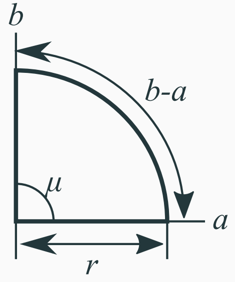
1/4円を描く軌道について

$$\kappa(t) = \frac{1}{r} = const. \qquad (6)$$

$$\mu = \int_a^b \kappa(s) ds \qquad (7)$$

$$= \frac{1}{r}\int_0^{\frac{\pi}{2}r} ds \qquad (8)$$

$$= \frac{\pi}{2} \qquad (9)$$



例: 円の場合

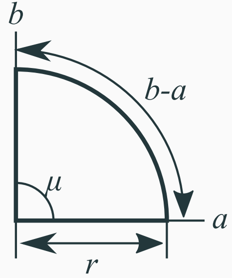
1/4円を描く軌道について

$$\kappa(t) = \frac{1}{r} = const. \tag{6}$$

$$\mu = \int_a^b \kappa(s)ds \tag{7}$$

$$= \frac{1}{r}\int_0^{\frac{\pi}{2}r} ds \tag{8}$$

$$= \frac{\pi}{2} \tag{9}$$



全曲率 = 軌道の回った角度

実装

$$\mu(s) = \int_a^b \kappa(s) ds \tag{10}$$

$$\mu(t) = \int_{t_a}^{t_b} \kappa(t) \frac{ds}{dt} dt \tag{11}$$

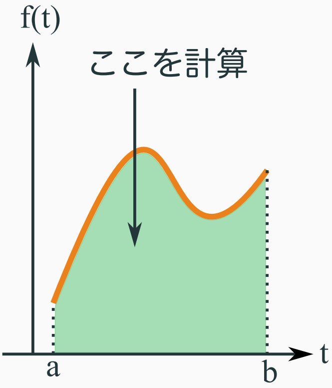
曲率を積分すればよい

積分

scipy.integrate.quad関数

quad(f, a, b)

f	関数
a, b	積分範囲
戻り値	積分値, 推定された誤差



実装

```python
class Curvature:
    def __init__(self, func):
        self._d = func.diff()
        self._c = Curvature(func)

    def _integrand(self, t):
        d = self._d(t)
        return (np.abs(self._c(t)) *
            ((d[0] ** 2 + d[1] ** 2) ** 0.5))

    def __call__(self, ta, tb):
        return si.quad(
            self._integrand, ta, tb
        )[0]
```

Questions?