数理科学コース 模擬授業

~~ 力学系における予測不可能性 ~~

花田 康高（理工学部 数理科学コース）

授業資料

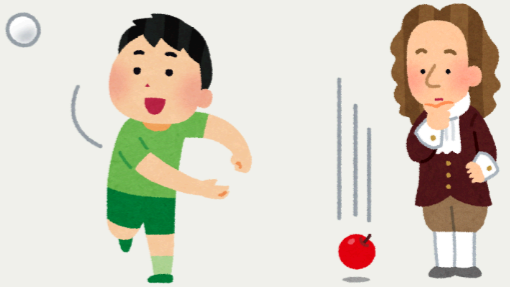
Menu

力学系とは？

力学系とは，状態の時間発展に関する力学法則に確率論的事象を含まない系

(例) 投擲（ニュートン力学）

反射, etc



力学法則（注:ただし空気抵抗はないものとする）

$$(1) \qquad m\frac{\mathrm{d}^2x}{\mathrm{d}t^2} = 0, \qquad m\frac{\mathrm{d}^2y}{\mathrm{d}t^2} = -mg$$

解

$$(2) \qquad x(t) = |\vec{v}_0|t\cos\theta, \qquad y(t) = |\vec{v}_0|t\sin\theta - \frac{1}{2}gt^2$$

力学系の特徴：

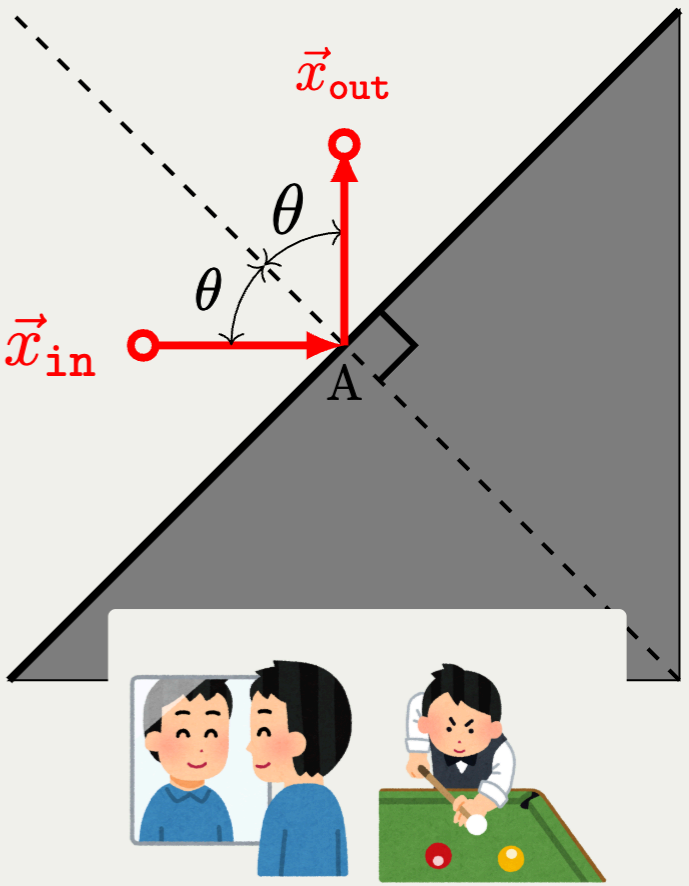
初期条件が決まれば任意の時刻tにおける座標が一意的に定まる（決定論）

反射の法則

力学系とは，状態の時間発展に関する力学法則に確率論的事象を含まない系

(例) 投擲（ニュートン力学）

反射, etc



■ 反射の法則

1. $\vec{x}_{\text{in}}$を延長して壁との交点Aを求める
2. 壁と直交する法線を引く
3. 法線を軸として$\vec{x}_{\text{in}}$を折り返す（鏡像）

■ 反射の特徴

入射角θ = 反射角θ　　注意:ベクトルの長さは変わらない（摩擦は無いもの）とする

初期条件$\vec{x}_{\text{in}}$を定めれば任意の時刻tにおける座標が一意的に定まる（決定論）

反射の法則

曲線$y=f(x)$に当たった場合は？

（$f(x)$は微分可能とする）

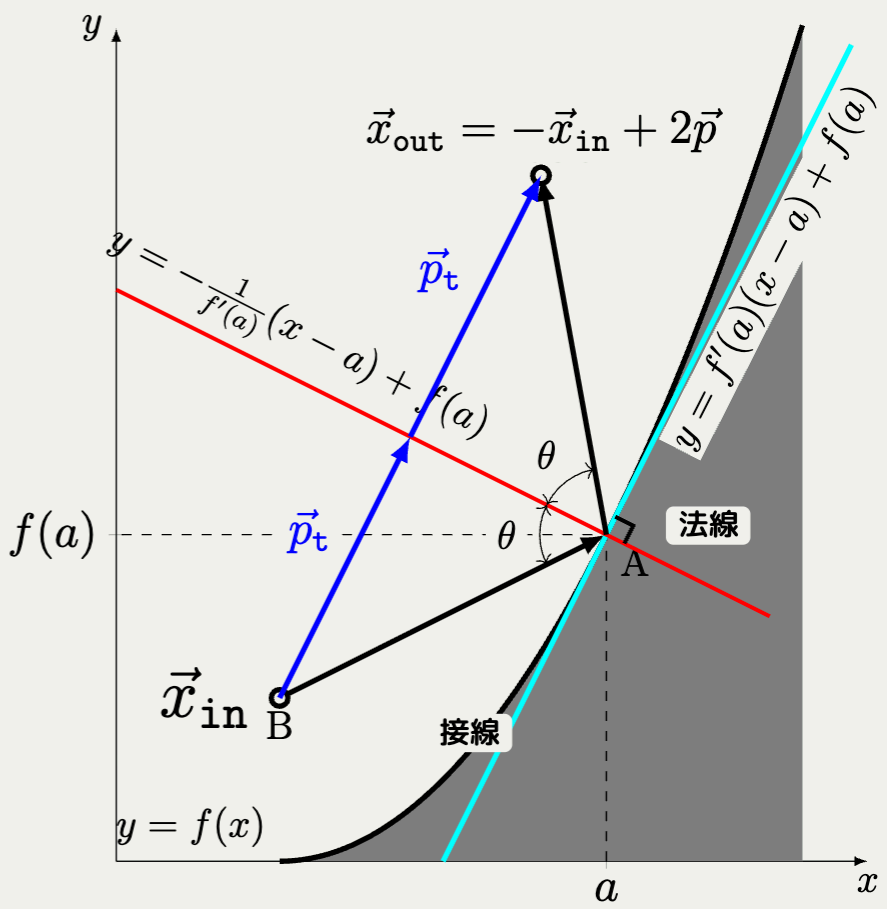
1. $\vec{x}_{\text{in}}$の延長と$f(x)$との交点Aを求める
2. Aでの$f(x)$の接線（数II）を引く
3. Aでの$f(x)$の法線（数III）を引く
4. $\vec{x}_{\text{in}}$を法線を軸として折り返した点が$\vec{x}_{\text{out}}$
5. $\vec{x}_{\text{in}}$の接線への射影ベクトル$\vec{p}_t$を用いて

$$\vec{x}_{\text{out}} = -\vec{x}_{\text{in}} + 2\vec{p}_t$$

注意： 微分可能な関数$f(x)$は点aの近傍で

$$f(x) \simeq f'(a)(x-a) + f(a)$$

と近似できる（1次近似，数III）．



射影ベクトル

射影ベクトル$\vec{p}_t$を求める． 内積の定義（数C）より

$$(3) \qquad \cos\theta = \frac{(-\vec{x}_{\mathrm{in}})\cdot\vec{n}}{|\vec{x}_{\mathrm{in}}||\vec{n}|} = -\frac{(\vec{x}_{\mathrm{in}}\cdot\vec{n})}{|\vec{x}_{\mathrm{in}}||\vec{n}|}.$$

△ABCは直角三角形なので$\cos\theta$の定義（数II）より

$$(4) \qquad \cos\theta = \frac{|\vec{c}|}{|\vec{x}_{\mathrm{in}}|}, \qquad \vec{c} := \overrightarrow{\mathrm{AC}}.$$

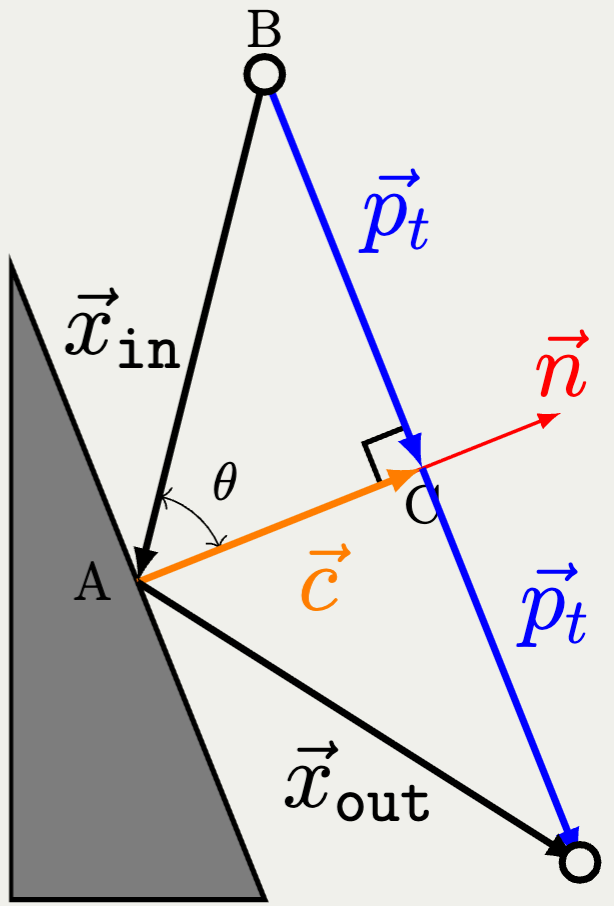
(3) = (4)より$|\vec{c}| = -\dfrac{\vec{x}_{\mathrm{in}}\cdot\vec{n}}{|\vec{n}|}$． 単位法線ベクトル$\hat{\vec{n}} = \dfrac{\vec{n}}{|\vec{n}|}$と

すると

$$\vec{c} = |\vec{c}|\hat{\vec{n}} = -\frac{(\vec{x}_{\mathrm{in}}\cdot\vec{n})}{(\vec{n}\cdot\vec{n})}\vec{n},$$

$$\vec{p}_t = \vec{x}_{\mathrm{in}} + \vec{c} = \vec{x}_{\mathrm{in}} - \frac{(\vec{x}_{\mathrm{in}}\cdot\vec{n})}{(\vec{n}\cdot\vec{n})}\vec{n}$$

$$\therefore \vec{x}_{\mathrm{out}} = -\vec{x}_{\mathrm{in}} + 2\vec{p}_t = \vec{x}_{\mathrm{in}} - 2\frac{(\vec{x}_{\mathrm{in}}\cdot\vec{n})}{(\vec{n}\cdot\vec{n})}\vec{n}$$



残りは，法線ベクトル$\vec{n}$が定まれば良い

法線ベクトル

反射の法則

$$\vec{x}_{\mathrm{out}} = \vec{x}_{\mathrm{in}} - 2\frac{(\vec{x}_{\mathrm{in}} \cdot \vec{n})}{(\vec{n} \cdot \vec{n})}\vec{n}$$

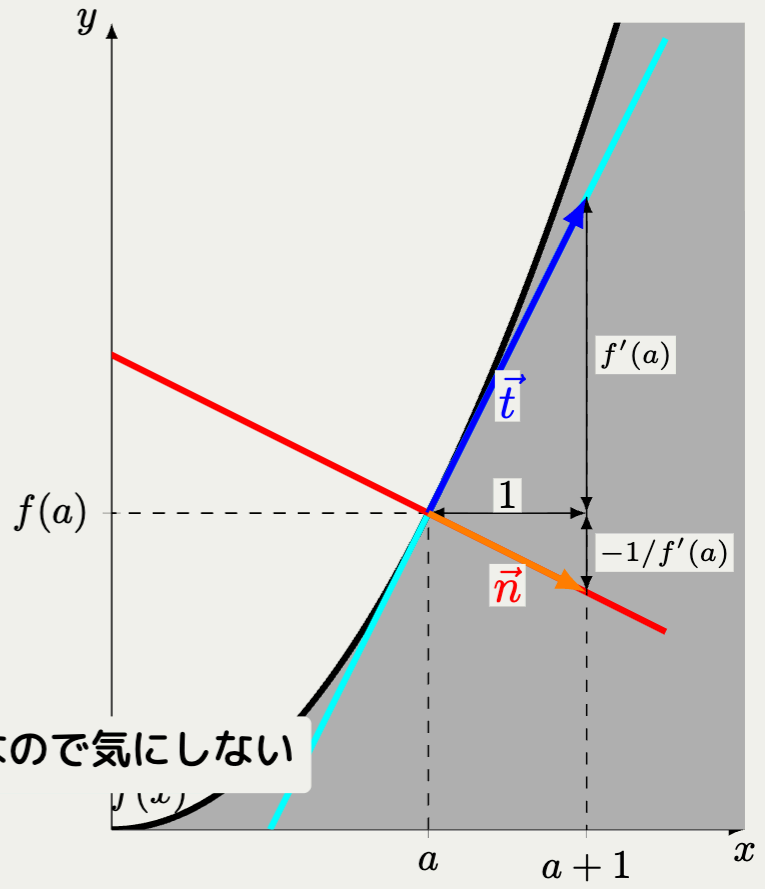
$x = a$での接線ベクトル$\vec{t}$と法線ベクトル$\vec{n}$

$$y = f'(a)(x-a) + f(a), \qquad \vec{t} = \begin{pmatrix} 1 \\ f'(a) \end{pmatrix}$$

$$y = -\frac{1}{f'(a)}(x-a) + f(a), \qquad (f'(a) \neq 0)$$

$$\vec{n} = \begin{pmatrix} 1 \\ -1/f'(a) \end{pmatrix} = -\frac{1}{f'(a)}\begin{pmatrix} -f'(a) \\ 1 \end{pmatrix}$$

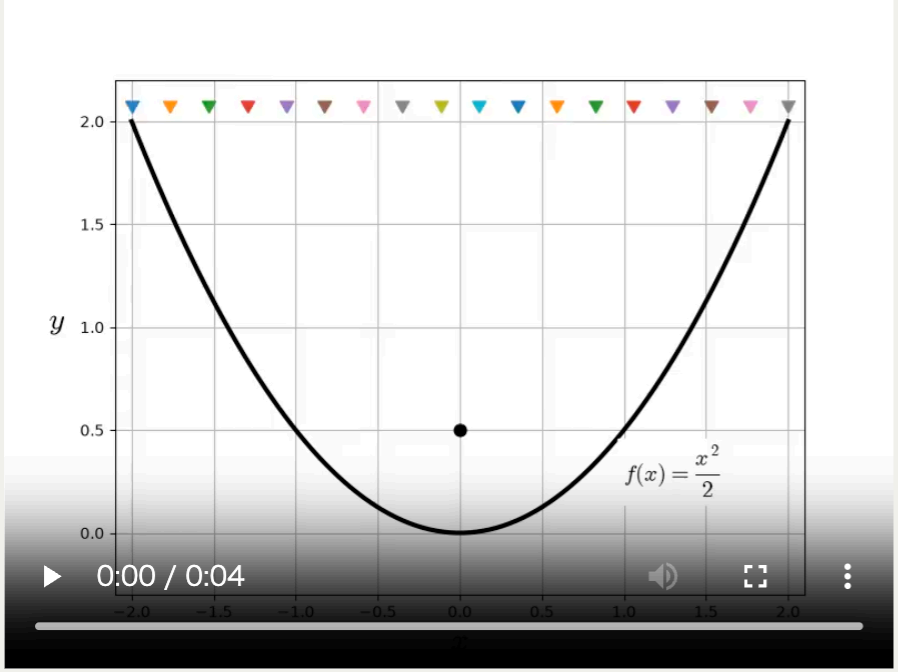
↑ $-\frac{1}{f'(a)}$はスカラー倍（定数）なので気にしない

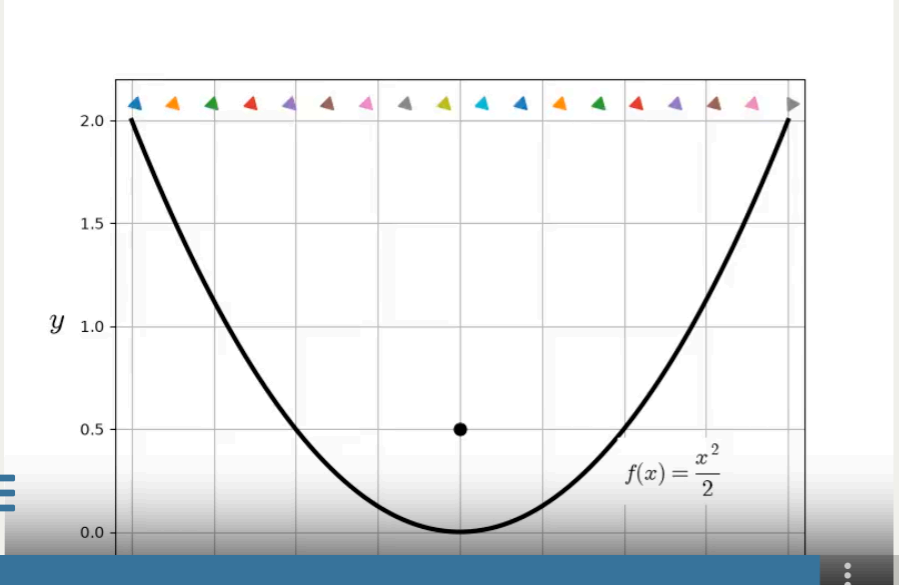


実際$\vec{t}$と$\vec{n}$の内積（数C）は

$$\vec{t} \cdot \vec{n} = 1 \times [-f'(a)] + 1 \times f'(a) = 0 \qquad \therefore \vec{t} \perp \vec{n}$$

パラボラアンテナ

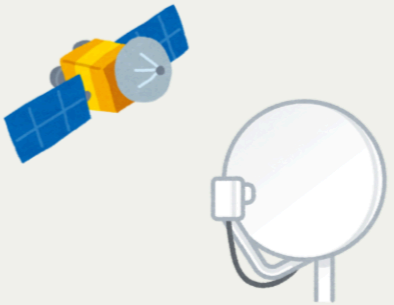




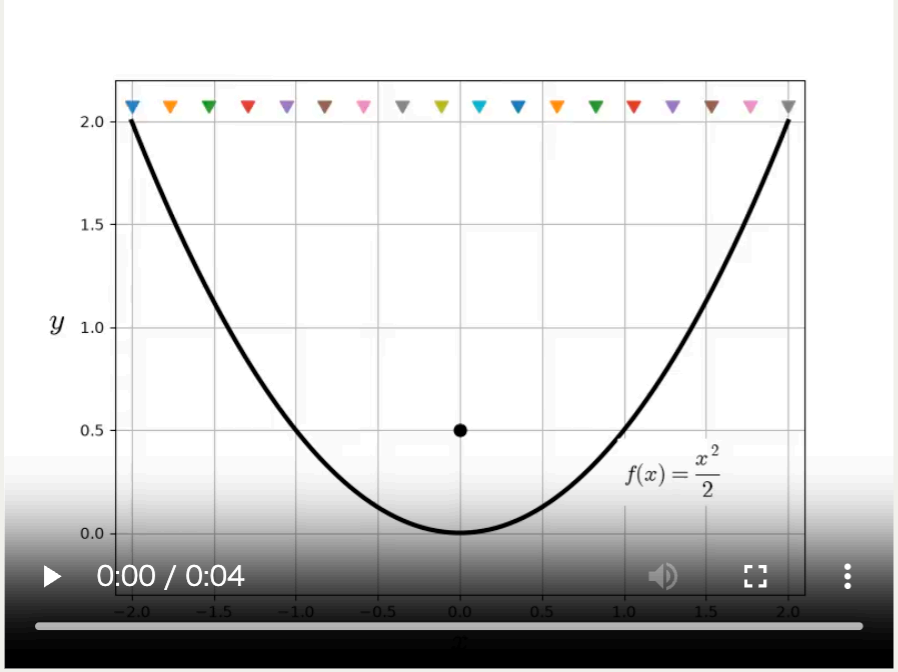
$$f(x) = \frac{x^2}{2} \implies \vec{n} = \begin{pmatrix} -x \\ 1 \end{pmatrix}$$

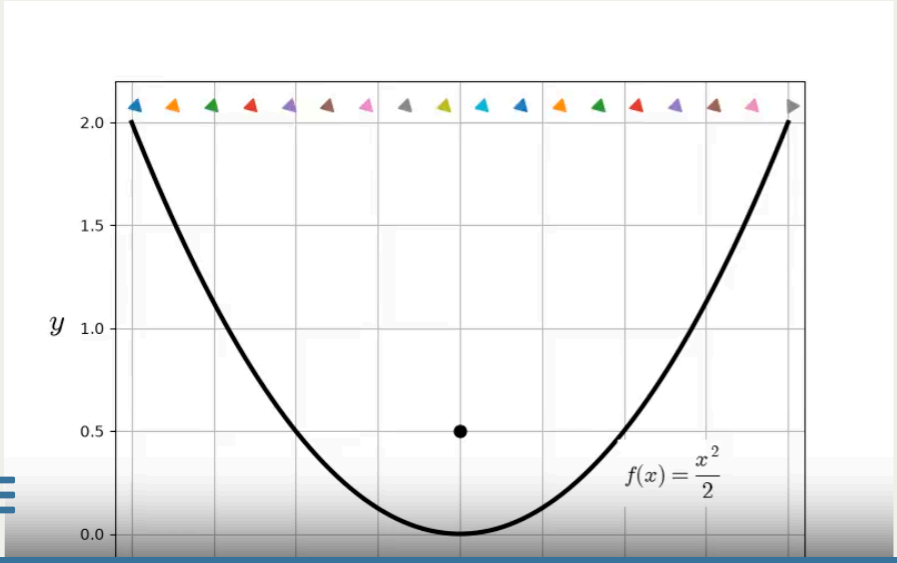
$\vec{x}_{\mathrm{in}}$が鉛直成分のみの場合，$\vec{x}_{\mathrm{out}}$は焦点（数C）に集まる

$\vec{x}_{\mathrm{in}}$が水平成分を含む場合はその限りではない（静止衛星からのデータ通信はアンテナの位置合わせが重要）



パラボラアンテナ





parabolic-reflector.py

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

sample = 18  # サンプル数
dt = 0.02    # 時間刻み
nsteps = 200 # 総ステップ数

'''
# google colab で実行する場合は以下の3行を実行して，最下部近傍のplt.sho
import matplotlib
%matplotlib nbagg
matplotlib.rc('animation', html='jshtml')
'''

def f(x):
    return x**2 / 2

# ---------- 全軌道を先に計算 ----------

traj = np.zeros((sample, nsteps, 2))                # (サンプルの数，フレ
init_x = np.linspace(-2, 2, sample)

for i, x0 in enumerate(init_x): # i番目のサンプル
    xy = np.array([x0, 2.1]) # 初期位置
    vec = np.array([0.0, -1.0]) # 初期向きのベクトル（下向き）
    #vec = np.array([1.0, -1.0]) # 初期向きのベクトル（右下向き）

    for s in range(nsteps): # 時間発展
        xy = xy + dt * vec    # xy をvecの方向にdtだけ進める
        if xy[1] < f(xy[0]):  # 面下に潜ったらvecの向きを変える
            xy[1] = f(xy[0])
            n = np.array([-xy[0], 1.0]) # 法線ベクトル（x軸方向の成分∂F
            vec = vec - 2 * (vec @ n) / (n @ n) * n # x@yは内積 xとyの

        if  np.allclose(xy, [0, 0.5], atol=dt):   # ¦ (xy[0], xy[1]) -
```

法線ベクトル

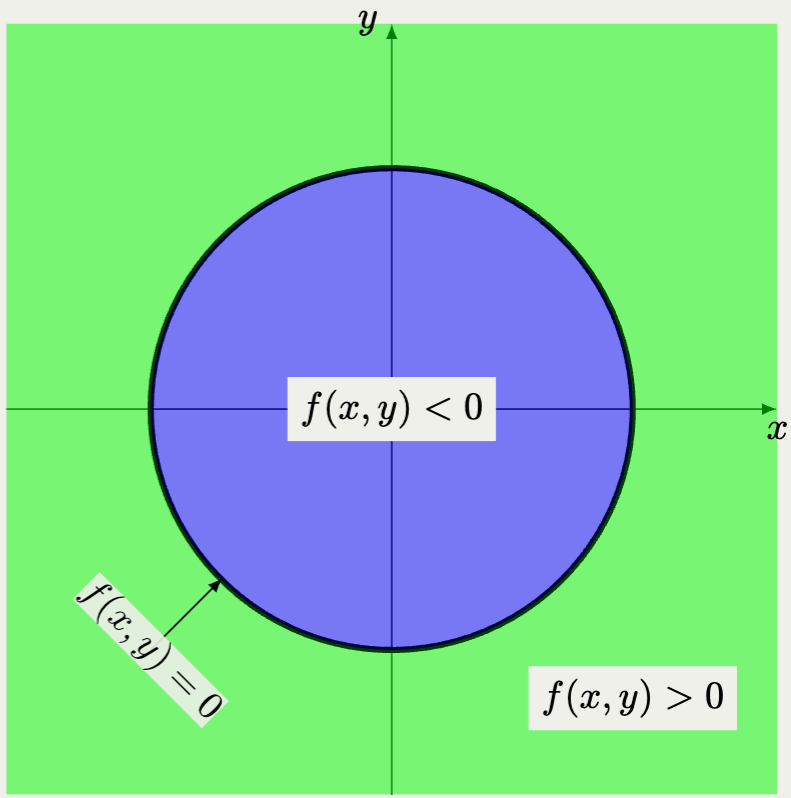
一般の曲線$f(x,y)=0$の法線ベクトル$\vec{n}$は

$$\vec{n}=\begin{pmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\end{pmatrix}$$

で与えられる．ここで偏微分（大学1年）は

$$\frac{\partial f}{\partial x}:=\lim_{h\to 0}\frac{f(x+h,y)-f(x,y)}{h},$$
$$\frac{\partial f}{\partial y}:=\lim_{h\to 0}\frac{f(x,y+h)-f(x,y)}{h}.$$

$\frac{\partial f}{\partial x}$は$y$を固定（定数とみなして）$x$で微分

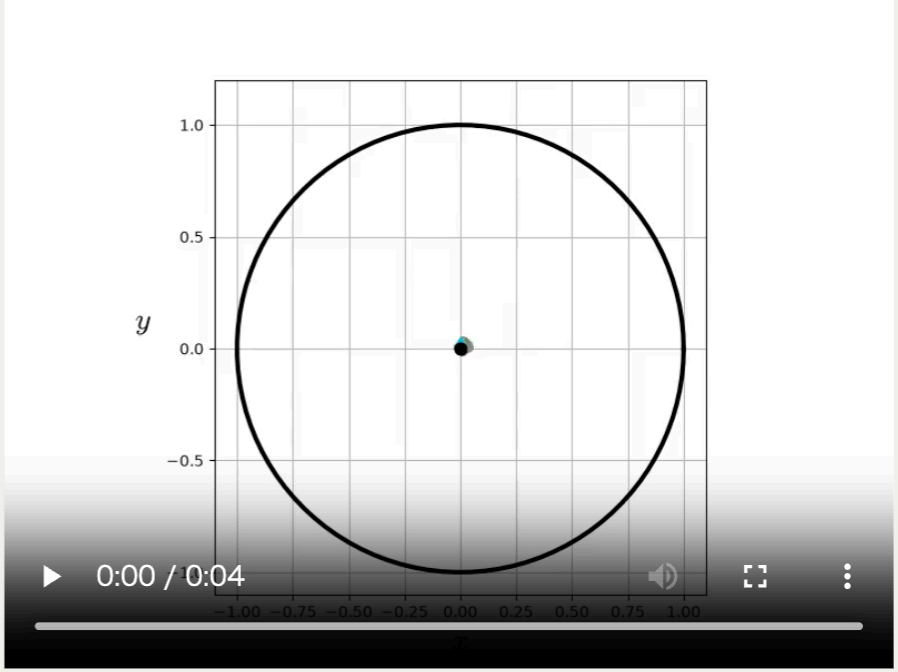


$$\frac{x^2}{2}+\frac{y^2}{2}=\frac{1}{2}\iff f(x,y):=\frac{x^2}{2}+\frac{y^2}{2}-\frac{1}{2}=0\implies\vec{n}=\begin{pmatrix}x\\ y\end{pmatrix}$$

円ビリヤード

circular-reflection.py

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

'''
# google colab で実行する場合は以下の3行も実行して，最下部近傍のplt.sho
import matplotlib
%matplotlib nbagg
matplotlib.rc('animation', html='jshtml')
'''

sample = 30  # サンプル数
dt = 0.02    # 時間刻み
nsteps = 200 # 総ステップ数

def f(x,y):
    return x**2 + y**2 - 1.0

# ---------- 全軌道を先に計算 ----------

traj = np.zeros((sample, nsteps, 2))               # (サンプルの数，フ
init_theta = np.linspace(0, np.pi/2, sample, endpoint=False)


for i, theta in enumerate(init_theta): # i番目のサンプル
    xy = np.array([0.0, 0.0], dtype=float)
    vec = np.array([np.cos(theta), np.sin(theta)]) # 初期向きのベクトル

    for s in range(nsteps): # 時間発展
        xy = xy + dt * vec    # xy をvecの方向にdtだけ進める
#        r = np.sqrt(xy[0]**2 + xy[1]**2)
        if f(xy[0], xy[1]) >= 0:  # xyがf(x,y)=0の外に出たらvecの向きを
            n = np.array([xy[0], xy[1]])  # 法線ベクトル (x軸方向の成分
            vec = vec - 2 * (vec @ n) / (n @ n) * n # x@yは内積 xとyの

        if (s > 1) and np.allclose(xy, [0, 0], atol=dt):
```



単位円

$$f(x,y) = \frac{x^2}{2} + \frac{y^2}{2} - \frac{1}{2} = 0$$

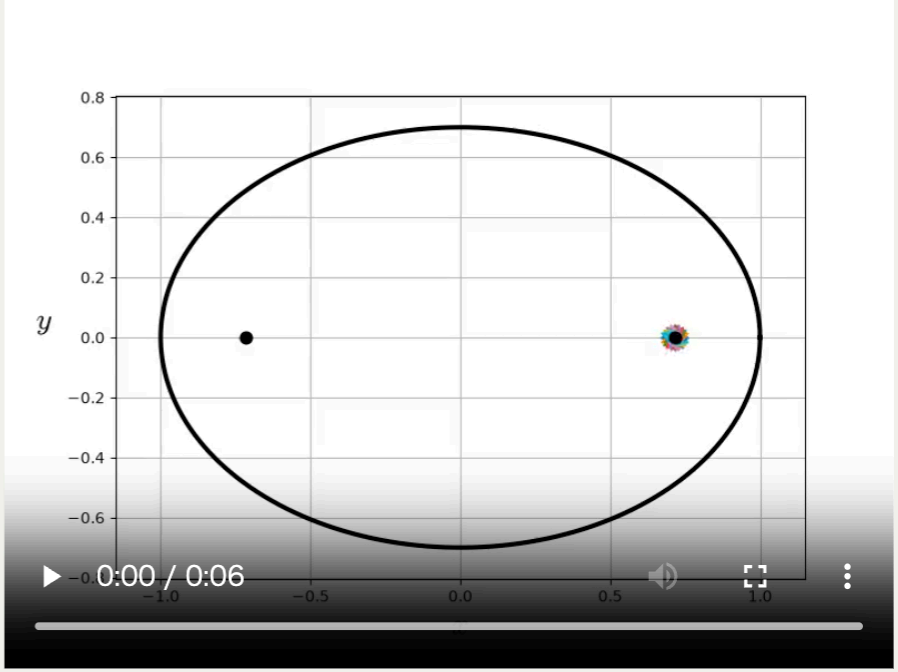
$$\vec{n} = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}$$

注意：境界との当たり判定が雑なため反射のタイミングがズレている

楕円ビリヤード

elliptic-reflection.py

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

'''
# google colab で実行する場合は以下の3行も実行して，最下部近傍のplt.sho
import matplotlib
%matplotlib nbagg
matplotlib.rc('animation', html='jshtml')
'''


sample = 20  # サンプル数
dt = 0.02    # 時間刻み
nsteps = 300 # 総ステップ数

a = 1.0
b = 0.7

def f(x,y ,a, b):
    return (x/a)**2 + (y/b)**2 - 1.0

def normal_vec(x, y, a, b):
    return np.array([2*x/(a**2), 2*y/(b**2)])

def focus_pt(a, b):
    c = np.sqrt(abs(a**2 - b**2))
    if a >= b:
        return np.array([c, 0.0]), np.array([-c, 0.0])
    else:
        return np.array([0.0, c]), np.array([0.0, -c])

# ---------- 全軌道を先に計算 ----------

traj = np.zeros((sample, nsteps, 2))              # (サンプルの数，フレ
init_theta = np.linspace(-np.pi, np.pi, sample, endpoint=False)
```



楕円（数学C）

$$f(x,y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0$$

$$\vec{n} = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2x/a^2 \\ 2y/b^2 \end{pmatrix}$$

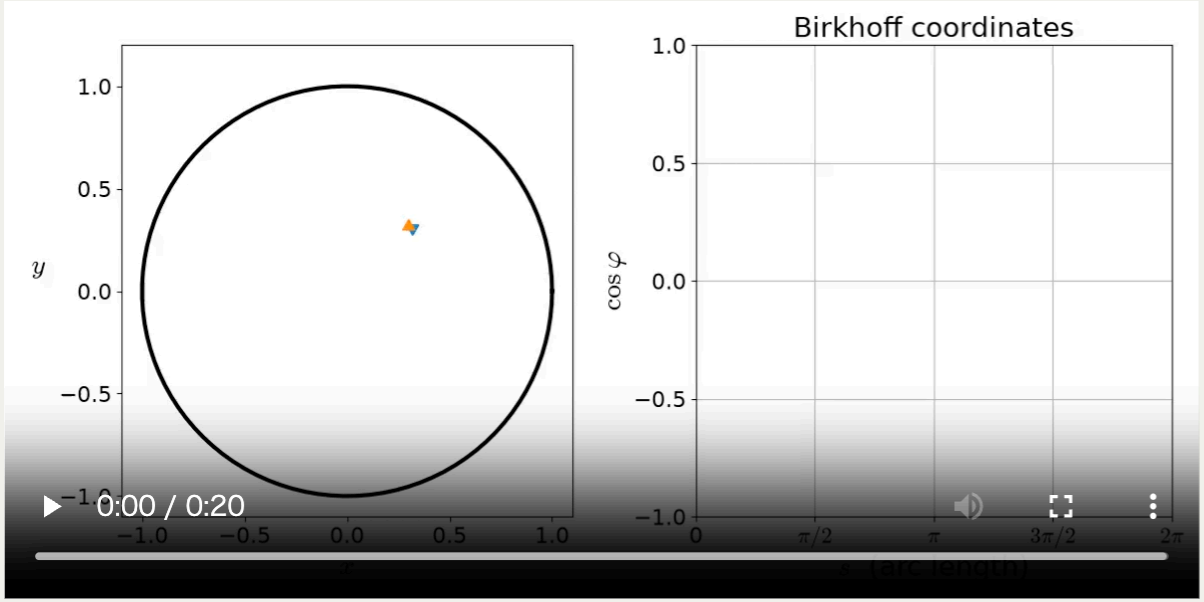
注意：境界との当たり判定が雑なため条件によっては焦点で止まらない場合もある

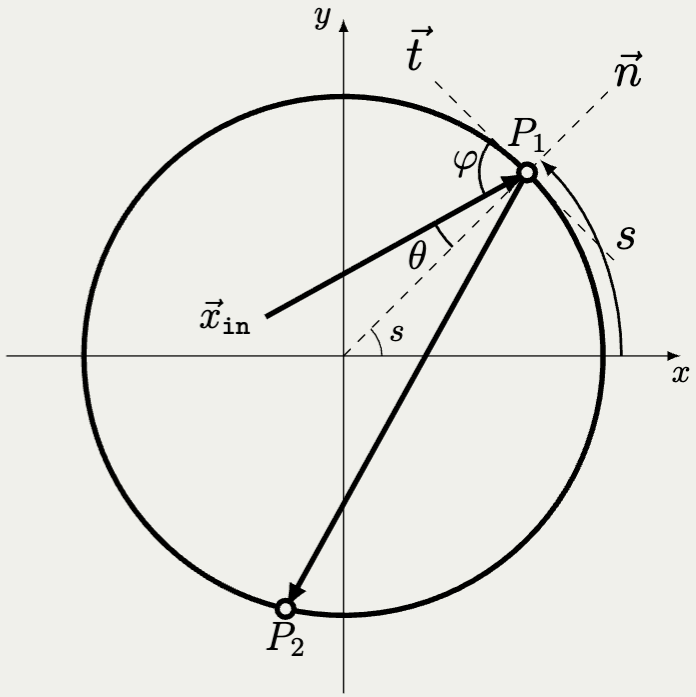
バーコフ座標

$\vec{x}_{\mathrm{in}}$の延長と円弧の交点の弧長sと，入射角θが決まれば自動的に次に当たる場所が確定する（決定論）

φをθの余角とする． n回目の衝突点$P_n(s_n, \cos\varphi_n)$を記録し$(s, \cos\varphi)$座標系（バーコフ座標）に軌跡を示す

注: $\cos\varphi = \dfrac{(\vec{x}_{\mathrm{in}} \cdot \vec{t})}{|\vec{x}_{\mathrm{in}}||\vec{t}|}$





円ビリヤードの$\cos\varphi$は一定

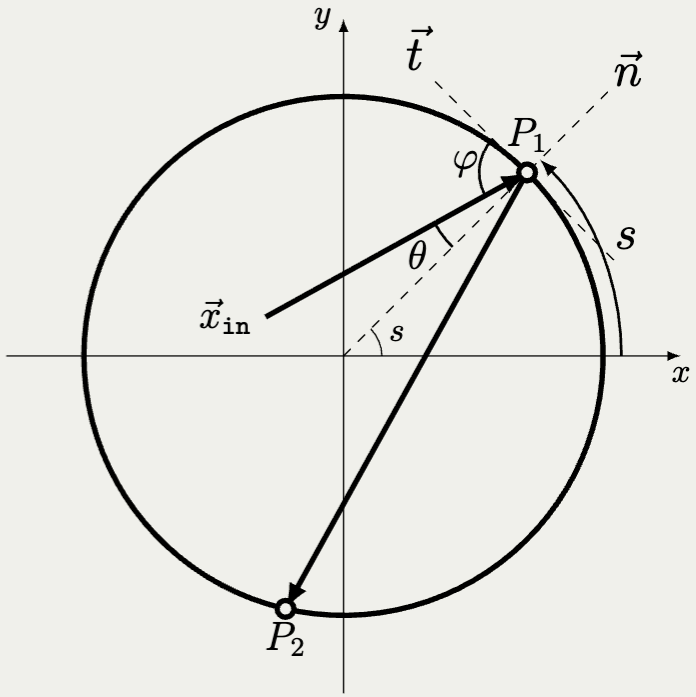
（規則的で予測が容易）

証明は反射の法則+中学数学幾何で可能

バーコフ座標

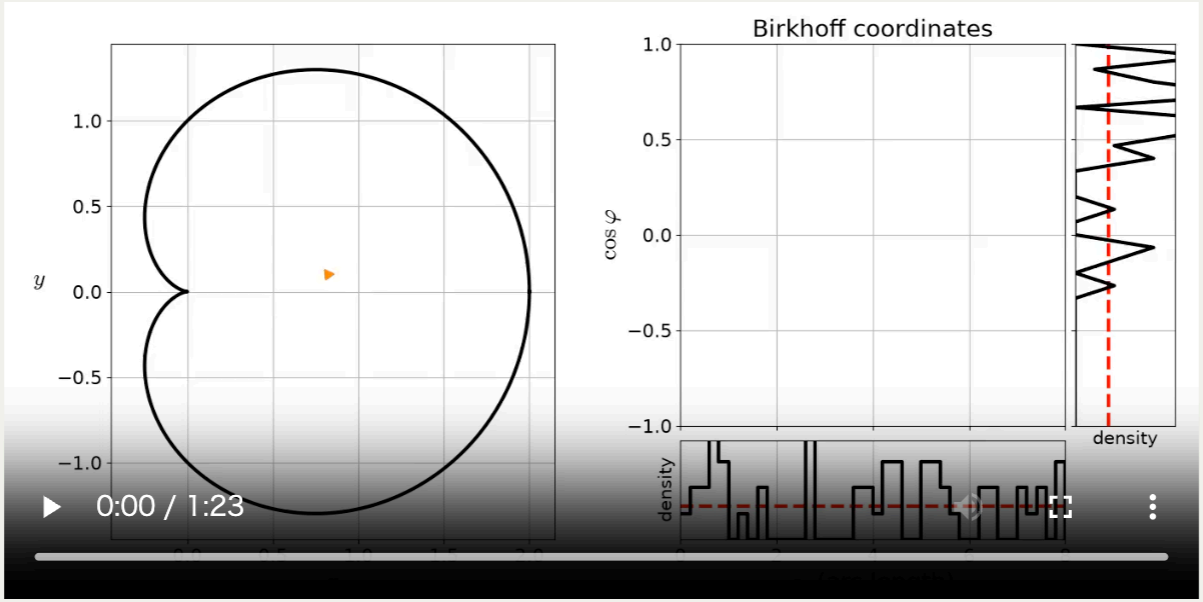
circular-billiard.py

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

'''
# google colab で実行する場合は以下の3行も実行して，最下部近傍のplt.sho
import matplotlib
%matplotlib nbagg
matplotlib.rc('animation', html='jshtml')
'''

sample = 2  # サンプル数
dt = 0.02    # 時間刻み
nsteps = 5000 # 総ステップ数
# 1コマで何ステップ進めるか。大きくすると速く・短くなる。
stride = 2

def f(x,y):
    return x**2 + y**2 - 1.0

# ---------- 全軌道を先に計算 ----------

traj = np.zeros((sample, nsteps, 2))               # (サンプルの数, フ
hits = [[] for _ in range(sample)]                 # (何ステップ目, 弧
init_theta = np.linspace(np.pi/6, np.pi/2, sample)


for i, theta0 in enumerate(init_theta): # i番目のサンプル
    xy = np.array([0.3, 0.3], dtype=float)
    vec = np.array([np.cos(theta0), np.sin(theta0)]) # 初期向きの単位ベ

    for t in range(nsteps): # 時間発展
        xy = xy + dt * vec    # xy をqvecの方向にdtだけ進める
#        r = np.sqrt(xy[0]**2 + xy[1]**2)
        if f(xy[0], xy[1]) >= 0:  # xyがf(x,y)=0の外に出たらvecの向きを
            n = np.array([xy[0], xy[1]])  # 法線ベクトル（x軸方向の成分
```



カージオイド（数C）

$$f(x,y) = (x^2+y^2)(x^2+y^2-2ax) - a^2y^2 = 0$$



初期に僅かな差（誤差）があると時間発展とともに全く異なる軌道に（カオス）

sおよび$\cos\varphi$のヒストグラム（密度プロット）は$n \to \infty$で一様分布（数B）に近づく

決定論的な力学系であっても長時間の振る舞いはルーレット回して$(s, \cos\varphi)$を定めたものと区別できない（決定論的予測不可能性 → 確率論の基礎）