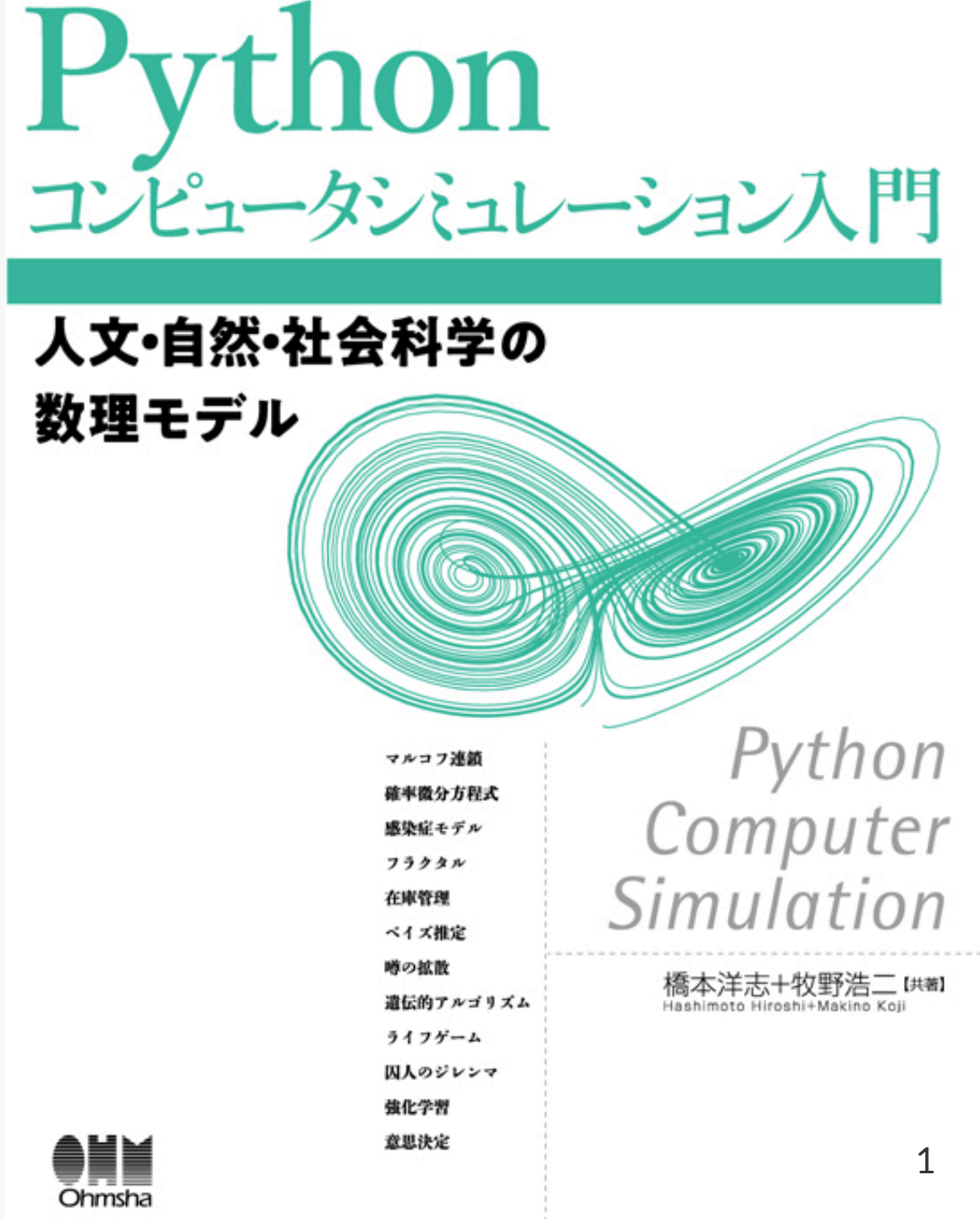
Pythonコンピュータシミュレーション入門

Python Computer Simulation



第3章 アニメーション

Chapter 3 Animation

--

3.1 アニメーションの意義

3.1 Meaning of animation

アニメーションの意義 / Meaning of animation

- アニメーションの動きを見ることで、シミュレーションで用いるモデルの本質をより理解できる、または気づきが得られる場合がある

--

- The animated movements may provide a better understanding or awareness of the nature of the model used in the simulation

アニメーションの意義 / Meaning of animation

今回扱うのは以下の2点：

- **Matplotlib** : 2次元のアニメーション
- **VPython** : 3次元のアニメーション、3次元座標系や空間内の幾何を視覚化し、さらに動きを与えることで数学やCG論の理解の助けになる

--

The following two items will be handled this time:

- **Matplotlib** : 2D animation
- **VPython** : 3D animation, visualization of geometry in 3D coordinate systems and space, and motion to help understand mathematics and computer graphics theory.

物理演算をしたいときは / If you want to do physical calculation

- **ODE (Open Dynamics Engine)** : 剛体が対象。比較的扱いが容易。C++。
- **Bullet Physics** : 剛体、ソフトボディ、流体を扱える。C, C++。これを応用したものにPanda3Dがあり、こちらはPythonが使用可能。

--

- **ODE (Open Dynamics Engine)**:.
  Rigid bodies are targeted. Relatively easy to handle. C++.
- **Bullet Physics**: We can handle rigid bodies, soft bodies, and fluids. C, C++.Panda3D is an application of Panda3D, which can use Python.

3.2 Matplotlibを用いたアニメーション

Animation used Matplotlib

- ドキュメント : https://matplotlib.org/ 内で"Animation"と検索
- Matplotlibは次の2つの方法を提供している
  - **ArtistAnimation** : あらかじめ作成した複数のグラフを切り替える方式
  - **FuncAnimation** : グラフ更新用の関数を用いて逐次的にアニメーションを表示する方式

--

- Documentation : search for “Animation” in https://matplotlib.org/
- Matplotlib offers two methods
  - **ArtistAnimation**: switching between multiple pre-created graphs
  - **FuncAnimation**: sequential animation using a function to update the graph

Matplotlibのポイント / Point of Matplotlib

- マジックコマンド **nbagg** : Notebook上でインタラクティブな画像表示を実現する機能
- `plt.cla()`, `plt.clf()`, `plt.close()` の機能
  - `plt.cla()` : Clear Axes, 現在のFigureの現在のアクティブなAxesをクリアする
  - `plt.clf()` : Clear Figure, 現在の図形全体をクリアする
  - `plt.close()` : Close a Figure window, 現在のグラフ用ウインドウをクローズする

--

- Magic command **nbagg** : functions for interactive image display on Notebook
- Functions of `plt.cla()`, `plt.clf()`, `plt.close()`
  - `plt.cla()` : Clear Axes, clears the current active Axes of the current Figure
  - `plt.clf()` : Clear Figure, clears the entire current Figure
  - `plt.close()` : Close a Figure window, closes the window for the current figure.

Matplotlibのポイント / Point of Matplotlib

- 引数（ArtistAnimationとFuncAnimationで共通なもの）
  - `interval` : 描画に関する手続き間のインターバル時間[ms]
  - `frames` : 描画するフレームの数
  - `repeat` : Trueはフレームの最後を描画した後に繰り返す。Falseは繰り返さない。
  - `blit` : いくつかの要件があるが、Trueにすると描画が高速になるらしい。
- アニメーションの保存 : matplotlib.animation https://matplotlib.org/ 内で"animation", "save"で検索
- Arguments (common to ArtistAnimation and FuncAnimation)
  - `interval` : interval time between drawing procedures [ms].
  - `frames` : number of frames to draw.
  - `repeat` : True means repeat after drawing the last of the frames; false means do not repeat; false means repeat after drawing the last of the frames.
  - `blit` : some requirements, but it is said to be faster if set to True.
- Save animation : search for “animation”, “save” in matplotlib.animation https://matplotlib.org/

3.2.1 ArtistAnimation

"AN_MatplotlibArtist.ipynb"を参照, 以下解説

- `num`個のランダムな位置(x, y)にマークを描いたグラフを`img`に格納し、これをリスト`imgBuffer`に順に追加する。ただし、標準正規分布を用いているので、(x, y) のレンジは予測できない
- このリストに格納されたグラフをArtistAnimationが`ts`[ms]ごとに、順に切り替えて描写する。
- `repeat=False`は一連のグラフを表示したら終了する。`True`ならば無限に繰り返す。この場合のアニメーションの収量は電源マークのアイコンをおせばよい
  --
  Please refer "AN_MatplotlibArtist.ipynb"
- A graph with `num` marks drawn at `num` random positions (x, y) is stored in `img`, which is added in turn to the list `imgBuffer`. However, the range of (x, y) is not predictable because of the standard normal distribution
- ArtistAnimation switches and depicts the graphs stored in this list every `ts`[ms] in order.
- If `repeat=False`, ArtistAnimation terminates after displaying a series of graphs. If `True`, it repeats indefinitely. In this case, you can check the yield of the animation by pressing the power icon.

3.2.2 FuncAnimation

"AN_MatplotlibFunc.ipynb"を参照, 以下解説

- 一つ目のプログラムは軸が固定され、二つ目は軸が移動する。この違いは、関数`Update()`が呼び出され、描画が更新される時に`xlim()`を固定値とするか、変化させるかによって生じている。

--

Please refer "AN_MatplotlibFunc.ipynb"

- In the first program, the axes are fixed, and in the second, the axes are moved. The difference is caused by whether `xlim()` is fixed or varied when the function `Update()` is called and the drawing is updated.

3.2.3 地球儀の回転 / Rotating Globe

"AN_Matplotartist.ipynb"を参照, 以下解説

- Cartopy：地図やその他の地理空間データ分析を作成するために地理空間データ処理用に設計された Python パッケージ（公式HPより）
- いくつかの投影法、東映における点、線、ベクトル、ポリゴン、画像などを変換する機能を持つ
- `ccrs.Orthographic()`：正射図法を指定。引数は中心となる経度と緯度の指定
- `coasltines()`：沿岸線を描く関数

  --

  Please refer "AN_Matplotartist.ipynb"
- Cartopy: Python package designed for geospatial data processing to create maps and other geospatial data analysis (from [official website] (https://scitools.org.uk/cartopy))
- Several projection methods, with the ability to convert points, lines, vectors, polygons, images, etc. in Toei
- `ccrs.Orthographic()` : Specifies the orthographic method. The argument specifies the longitude and latitude of the center.
- `coasltines()` : Functions to draw coastlines.

3.3 VPythonを用いたアニメーション

3.3 Animation by VPython

--

3.3.1 簡単な使い方

3.3.1 Simple usase

基本情報 / ****

- 透視投影を用いており、右手系を採用している
- 回転の正方向は、右ねじの法則に従う
- 詳しい情報：公式サイト, ドキュメント
- ちなみに、初期設定では、視野に映るシーンは画面右へ向くベクトルが＋xである

--

- Uses perspective projection and a right-handed system
- Positive direction of rotation obeys the right-hand thread law
- More information: official website, documentation
- Note that, by default, the scene in the field of view has a vector +x pointing to the right of the screen

boxの引数 / ****

```
vp.box(pos=vp.vector(x0,y0,z0),size=vp.vector(L,H,W),
color=vp.vector(R,G,B),axis=vp.vector(x,y,z))
```

- VPythonのベクトル表現： `vp.vector()`
- `pos` : boxの中心位置。表す位置は物体により異なる(例えばcylinderは端にとる)
- `size` ： `Length`, `Height`, `Width` の順番で与え、それぞれxyz軸方向に対応
- `color` : オブジェクト表面の色。[0, 1]である
- `axis` : Length方向の方向ベクトル。ローカル座標を指定している

--

- VPython vector representation: `vp.vector()`
- `pos` : center position of the box. The representation position depends on the object (e.g., cylinder takes the edge).
- `size` : given in the order of `Length`, `Height`, `Width`, each corresponding to the direction of the xyz axis.
- `color` : color of the object surface. [0, 1].
- `axis` : Direction vector in the Length direction. Local coordinates are specified.

バネの伸縮アニメーション / Animation of spring expansion and contraction

"AN_VPythonCubeSpring.ipynb"を参照, 以下解説

- `vp.canvas()` : 描画のためのキャンバスを作成する
- `vp.helix()` : ばねの描画。ばねの左端を壁の表面、右端を立方体の左面位置とし、方向ベクトルをこれらの差としている。
  --

Please refer "AN_VPythonCubeSpring.ipynb"

- `vp.canvas()` : creates a canvas for drawing.
- `vp.helix()` : draws a spring. The left end of the spring is the surface of the wall, the right end is the position of the left face of the cube, and the direction vector is the difference between them.

3.3.2 ビリヤードの球の衝突問題 / Billiard ball collision problem

- ビリヤード球が衝突したとき、その後の２つのビリヤード球の速度ベクトルはどうなるか考える
- 下図のような状況で、次式が成り立つ

$$x = 2r\sin\theta_2$$
$$x = 2r - d$$

xを消去すると、

$$d = 2r(1 - \sin\theta_2)$$

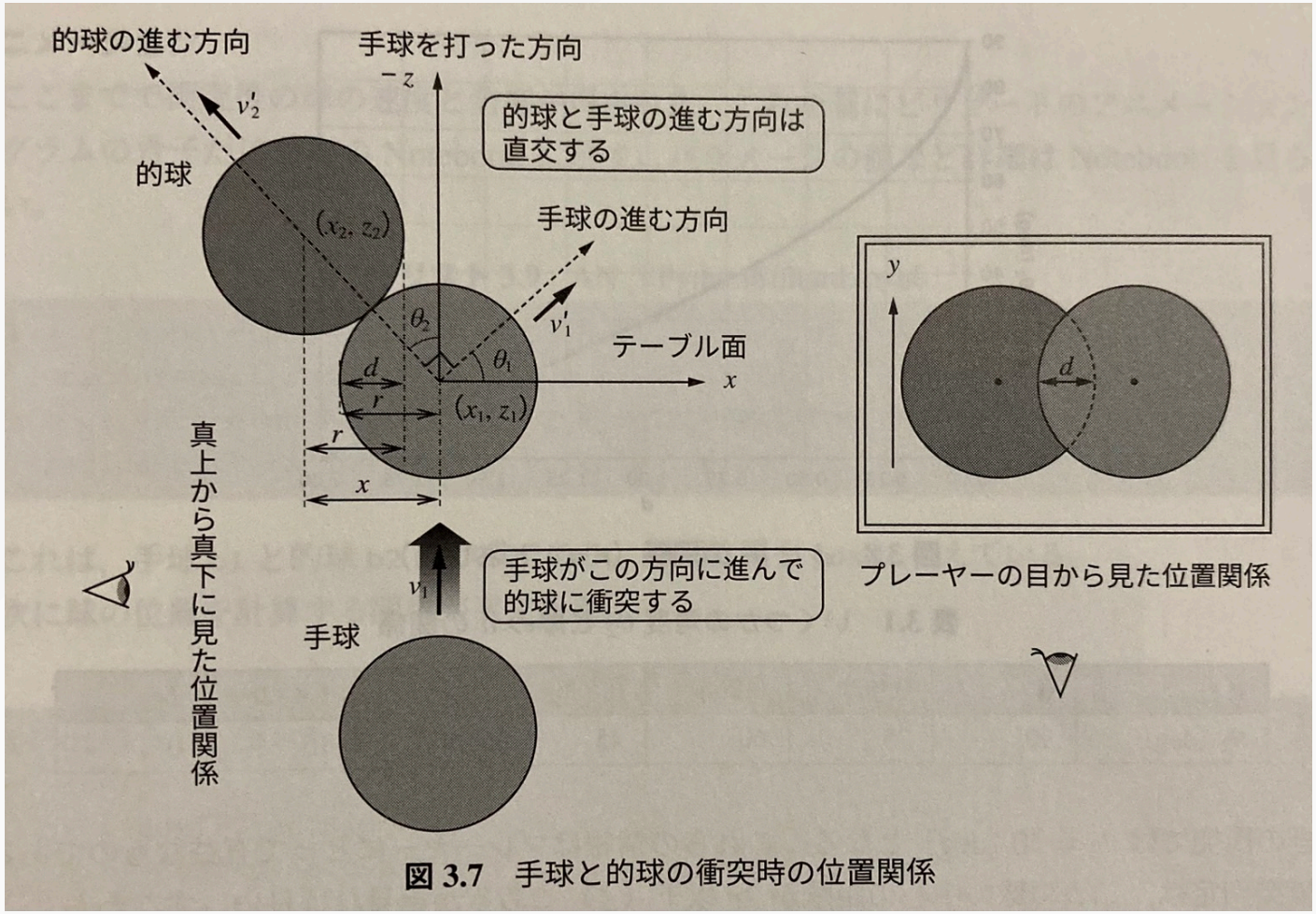


図 3.7　手球と的球の衝突時の位置関係

3.3.2 ビリヤードの球の衝突問題 / Billiard ball collision problem

- Consider what happens to the velocity vectors of two billiard balls after they collide
- In the situation shown below, the following equation holds
  ! [bg right:55% w:700](. /fig/chapter3/billiard.jpg)

$$x = 2r \sin \theta_2$$
$$x = 2r - d$$

Eliminating x,

$$d = 2r(1 - \sin \theta_2)$$

3.3.2 ビリヤードの球の衝突問題 / Billiard ball collision problem

$d = 2r(1 - \sin\theta_2)$について、$r = 1$としたときのdとθ_2の関係は以下のようになる

- $0.5 \leq d \leq 2$の範囲では、dとθ_2はおよそ線形関係にあるように見える。
- $d = 1$のとき、$\theta_2 = 30$となる。

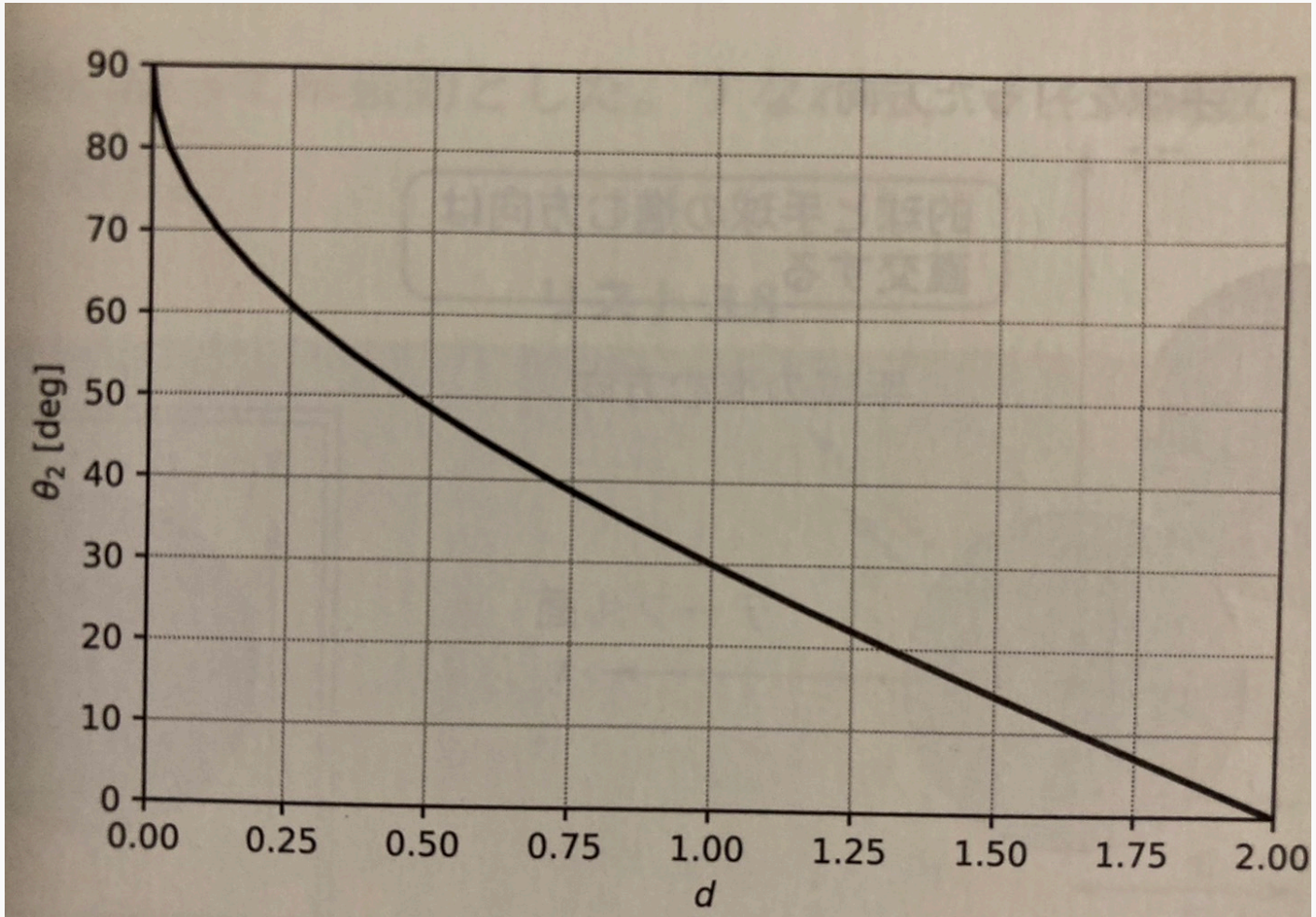


図 3.8　d と θ_2 の関係（$r = 1$ とおいた）

3.3.2 ビリヤードの球の衝突問題 / Billiard ball collision problem

For $d = 2r(1 - \sin\theta_2)$, the relation between d and θ_2 when $r = 1$ is as follows

- In the range $0.5 \leq d \leq 2$, d and θ_2 appear to have an approximately linear relationship.
- When $d = 1$, $\theta_2 = 30$.

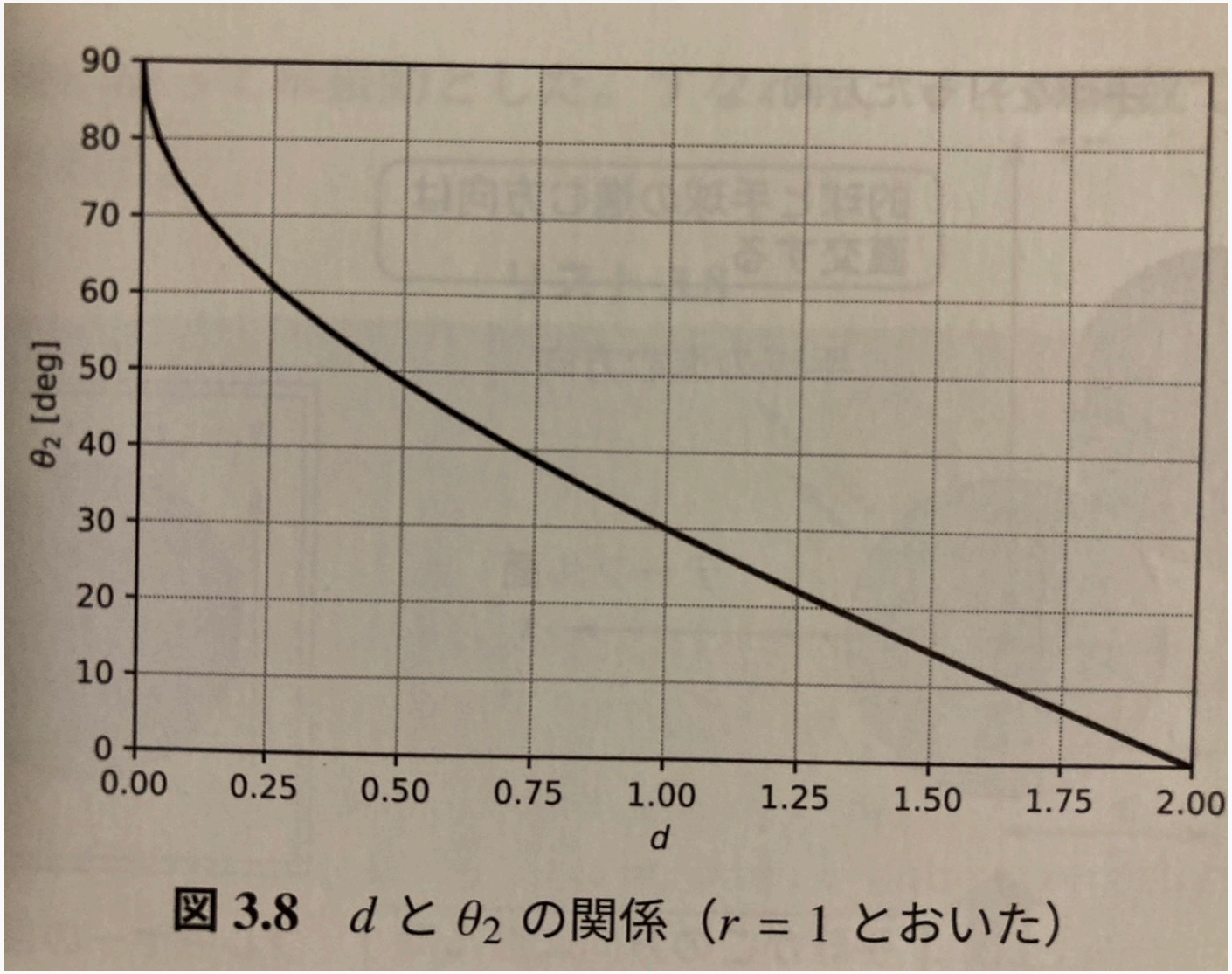


図 3.8　d と θ_2 の関係（$r = 1$ とおいた）

3.3.2 ビリヤードの球の衝突問題 / Billiard ball collision problem

- 衝突判定：$\sqrt{((x_1(t)-x_2(t))^2+(z_1(t)-z_2(t))^2)} \leq 2r$
- θ_2と衝突後の速度 v_1', v_2'は次のように求める
  - 運動量保存則：$mv_1+mv_2=mv_1'+mv_2'$
  - 反発係数：$e=-\frac{v_1'-v_2'}{v_1-v_2}$
  - 条件より、$v_2=0$。v_1', v_2'について解くと,

$$v_1'=\frac{1}{2}v_1(1-e)$$
$$v_2'=\frac{1}{2}v_1(1+e)$$

- 衝突後の角度：
  - $x=2r\sin\theta_2$を変形して、$\theta_2=\sin^{-1}\left(\frac{x}{2r}\right)$
  - 分離角度は90度になるから、$\theta_1=\theta_2-\frac{\pi}{2}$

3.3.2 ビリヤードの球の衝突問題 / Billiard ball collision problem

- Collision determination: $\sqrt{((x_1(t) - x_2(t))^2 + (z_1(t) - z_2(t))^2)} \leq 2r$
- The velocities v_1', v_2' after collision with θ_2 are obtained as follows
  - Momentum conservation law: $mv_1 + mv_2 = mv_1' + mv_2'$.
  - Coefficient of repulsion: $e = -\frac{v_1' - v_2'}{v_1 - v_2}$
  - From the condition $v_2 = 0$. Solving for v_1', v_2', we obtain.

$$v_1' = \frac{1}{2}v_1(1 - e)$$
$$v_2' = \frac{1}{2}v_1(1 + e)$$

- Angle after collision:
  - Transform $x = 2r\sin\theta_2$ to $\theta_2 = \sin^{-1}\left(\frac{x}{2r}\right)$
  - Since the separation angle is 90 degrees, $\theta_1 = \theta_2 - \frac{\pi}{2}$

3.3.2 ビリヤードの球の衝突問題 / Billiard ball collision problem

- 以上を基にアニメーションを実装する。"AN_VPythonbilliard.ipynb"を参照
  - `coef`は速度、`c_rest`は反発係数
- 本アニメーションは、厚みと球の進行方向を動体視力によって視認識するための良いトレーニングになるだろう、と著者は述べている
  --
- Implement the animation based on the above.” See `AN_VPythonbilliard.ipynb
  - coef`is the velocity and`c_rest` is the rebound coefficient.
- The author states that this animation is a good training for visual recognition of the thickness and the direction of the sphere's motion by dynamic vision.

3.3.3 振り子の等時性は成り立つのか（練習問題） / Does isochrony of the pendulum hold?（Exercise）

真空中に存在する、質量m, 糸の長さlの振り子を考えます。重力加速度をg, 地面からの垂線を基準とする振れ角θ[rad]とします。
(1)この振り子に関する連立方程式を書いてください。
(2)$\theta(t)$が十分に小さいとき、$\theta(t)$の一般解を求めてください。
(3)(1)で求めた式を、連立の一階微分方程式に書き直すことを考えます。式の１つが$\frac{d}{dt}\theta(t) = \omega(t)$であるとき、もう1つの式を書いてください。
(4)"AN_VPythonPendulum.ipynb"を実行し、$\theta $の初期値が振り子の等時性に与える影響を考察してください。

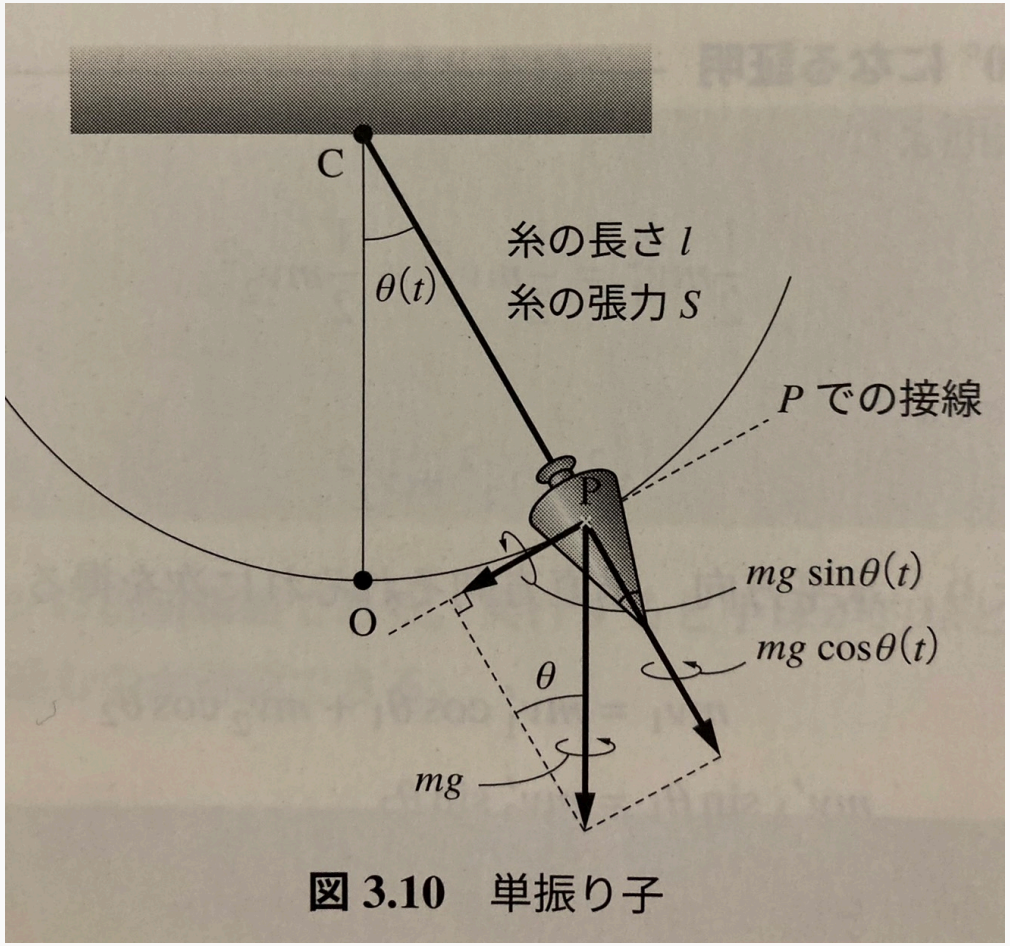


図 3.10　単振り子

Consider a pendulum with mass m and thread length l existing in a vacuum. Let the acceleration of gravity be g and the swing angle θ [rad] with respect to the perpendicular line from the ground.

(1)Write a simultaneous equation for this pendulum.

(2)When $\theta(t)$ is sufficiently small, find a general solution of $\theta(t)$.

(3) Consider rewriting the equation obtained in (1) as a simultaneous first-order differential equation. When one of the equations is $\frac{d}{dt}\theta(t) = \omega(t)$, write another one.

(4) Run “AN_VPythonPendulum.ipynb” and consider the effect of the initial value of θ on the isochronism of the pendulum.

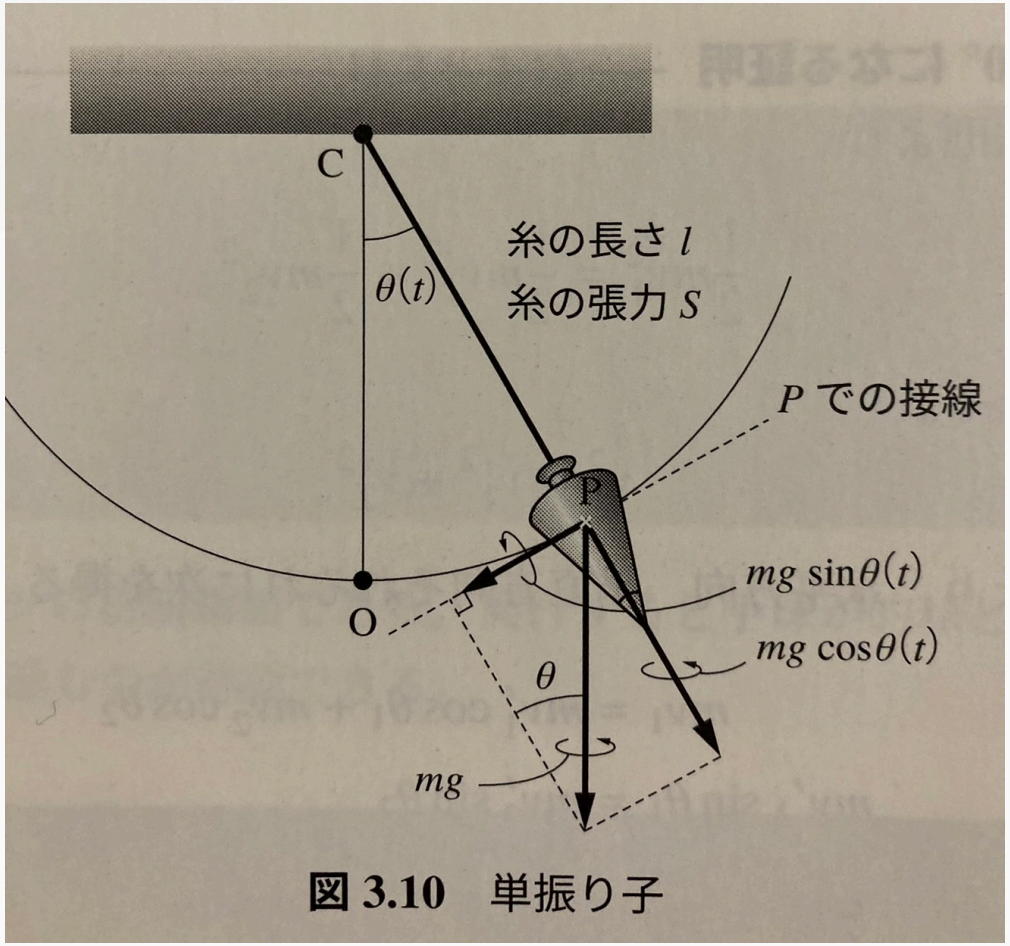


図 3.10　単振り子

Answer

(1) $m\frac{d^2 l\theta(t)}{dt^2} = -mg\sin\theta(t)$

(2) $\theta(t) = A\sin(\sqrt{(}L/g)t + B)$ （A, B are Arbitrary constants）

(3) $\frac{d}{dt}\omega(t) = -\frac{g}{l}\sin\theta(t)$