2018年航空宇宙情報システム学第２
第2部「プログラミングと数値計算」

Numpyによる数値計算入門２
常微分方程式の数値解法

2018年7月3日

（復習）小課題3（5月29日の課題）

減衰振動の運動方程式を表す微分方程式

$$\ddot{x} + 2\gamma\dot{x} + {\omega_0}^2 x = 0$$

の挙動を、オイラー法による数値計算で調べよ。

与えられた微分方程式を変形すると、

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ -{\omega_0}^2 x - 2\gamma\dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -{\omega_0}^2 & -2\gamma \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

$\mathbf{x} \equiv [x, \dot{x}]^T \quad A = \begin{bmatrix} 0 & 1 \\ -{\omega_0}^2 & -2\gamma \end{bmatrix}$ と置けば、$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$ と書ける

解析解は、$\boldsymbol{x}(t) = e^{tA}\,\boldsymbol{x}(0)$ となる

行列指数関数

オイラー法の手順

（1）微分方程式を次の形式で表現する

$$\dot{\boldsymbol{x}} = f(t, \boldsymbol{x}) \quad \Rightarrow \quad \boxed{\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}}$$

（2）十分小さいΔt に対して、1次のTaylor展開で近似

$$\boldsymbol{x}(t+\Delta t) \approx \boldsymbol{x}(t) + \Delta t \cdot f(t, \boldsymbol{x})$$

$$\Rightarrow \quad \boxed{\boldsymbol{x}(t+\Delta t) \simeq \boldsymbol{x}(t) + \Delta t \cdot \boldsymbol{A}\boldsymbol{x}(t)}$$

（3） $t_n = n\Delta t$, $\boldsymbol{x}_n = \boldsymbol{x}(t_n) \quad (n = 0, 1, \cdots, N)$

と置けば、下の漸化式（差分方程式）が得られる

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \Delta t \cdot f(t_n, \boldsymbol{x}_n)$$

$$\Rightarrow \quad \boxed{\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \Delta t \cdot \boldsymbol{A}\boldsymbol{x}_n = (\boldsymbol{I} + \Delta t \cdot \boldsymbol{A})\boldsymbol{x}_n}$$

Python+NumPyによるオイラー法（1）

euler_damped_oscillation.py

```
from numpy import *
from pylab import *

# Parameters
gam = 0.3
ome2 = 1.0

# Time step size
dlt = 0.1

# Array of time steps
T = arange(0.0,10.0,dlt)
# Number of steps
N = T.shape[0]
# Array of state vectors
X = zeros((N,2))
```

#NumPyとMatplotlibをインポート

#パラメータγ, ω_0^2の値（変えて挙動を見てみよう）

#計算時刻の刻み幅。大き過ぎると精度が悪化

#開始時刻0.0から終了時刻10.0までを分割

#時刻ステップの数

#各時刻で計算した状態ベクトルを格納するarray

次ページに続く

Python+NumPy によるオイラー法（2）

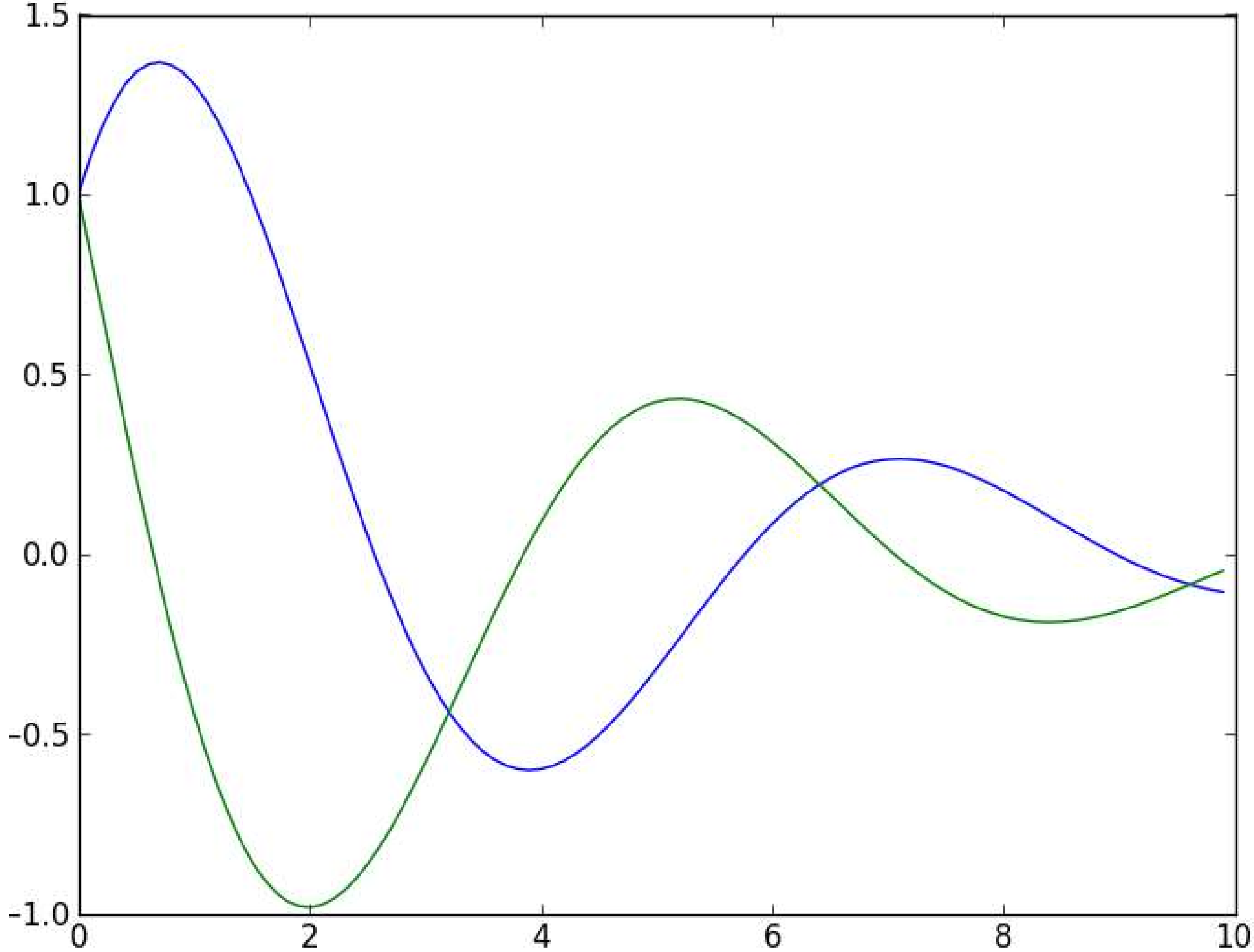
前ページから続く

```
# Initial state
X[0,:] = [1.0, 1.0]    #時刻t=0での状態を指定（変更してみよう）

# Matrix A
A = array([[0.0,1.0],[-ome2,-2.0*gam]])

# Loop
for n in range(N-1):
    X[n+1,:] = X[n,:] + dlt * dot(A,X[n,])

# Plot by matplotlib
plot(T,X[:,0])
plot(T,X[:,1])
show()
```

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -\omega_0{}^2 & -2\gamma \end{bmatrix}$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \Delta t \ \boldsymbol{A}\boldsymbol{x}_n$$

Euler法による数値解（$\Delta t = 0.1$としたとき）



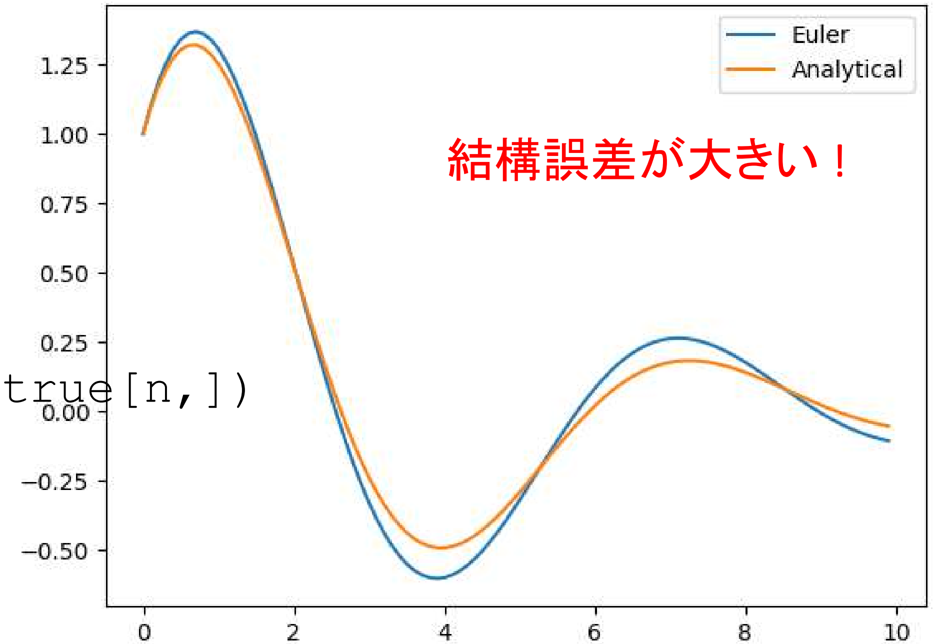
一見、良さそうに見えるが、どれくらい正確なのか？

解析解との比較

解析解 $\boldsymbol{x}(t) = e^{tA}\,\boldsymbol{x}(0)$ より、漸化式 $\boldsymbol{x}_{n+1} = e^{\Delta tA}\boldsymbol{x}_n$

```
import scipy.linalg as scla
expdtA = scla.expm(dlt*A)
Xtrue = zeros((N,2))
Xtrue[0,:] = [1.0, 1.0]
for n in range(N-1):
    Xtrue[n+1,:] = dot(expdtA,Xtrue[n,])
figure()
plot(T,X[:,0])
plot(T,Xtrue[:,0])
legend(['Euler','Analytical'])
```

解析解（橙）とオイラー法（ ⊿t=0.1 ）の解（青）



結構誤差が大きい！

Euler 法と解析解（減衰振動の場合）

両者の漸化式を比較してみると、

- Euler法の漸化式：

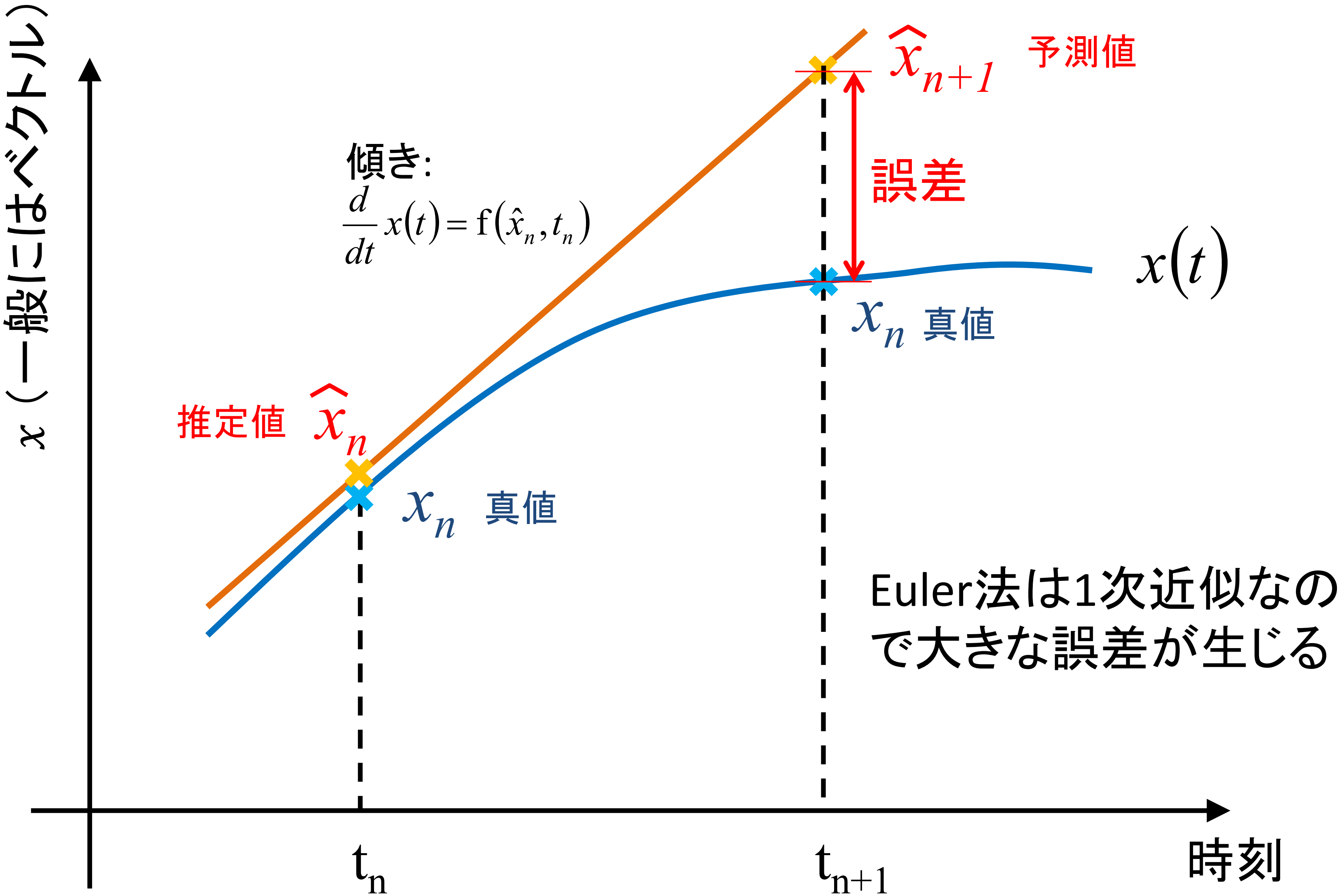
$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \Delta t \cdot \boldsymbol{A}\boldsymbol{x}_n = \underbrace{(\boldsymbol{I} + \Delta t\,\boldsymbol{A})}\boldsymbol{x}_n$$

- 解析解の漸化式：

$$\boldsymbol{x}_{n+1} = e^{\Delta t \boldsymbol{A}}\boldsymbol{x}_n = \left(\overbrace{\boldsymbol{I} + \Delta t\,\boldsymbol{A}} + \underbrace{\frac{\Delta t^2}{2}\boldsymbol{A}^2 + \cdots + \frac{\Delta t^k}{k!}\boldsymbol{A}^k + \cdots}_{\text{Euler 法で無視される項}}\right)\boldsymbol{x}_n$$

誤差の要因

Euler 法を視覚的に解釈すると



2次近似

- 2次のTaylor展開をしてみると

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}(t_n + \Delta t) \approx \boldsymbol{x}(t_n) + \Delta t \cdot \frac{d\boldsymbol{x}}{dt}\Big|_{t=t_n} + \frac{(\Delta t)^2}{2} \cdot \frac{d^2\boldsymbol{x}}{dt^2}\Big|_{t=t_n}$$

ここで、

$$\frac{d\boldsymbol{x}}{dt}\Big|_{t=t_n} = \dot{\boldsymbol{x}}(t_n) = f(\boldsymbol{x}_n, t_n)$$

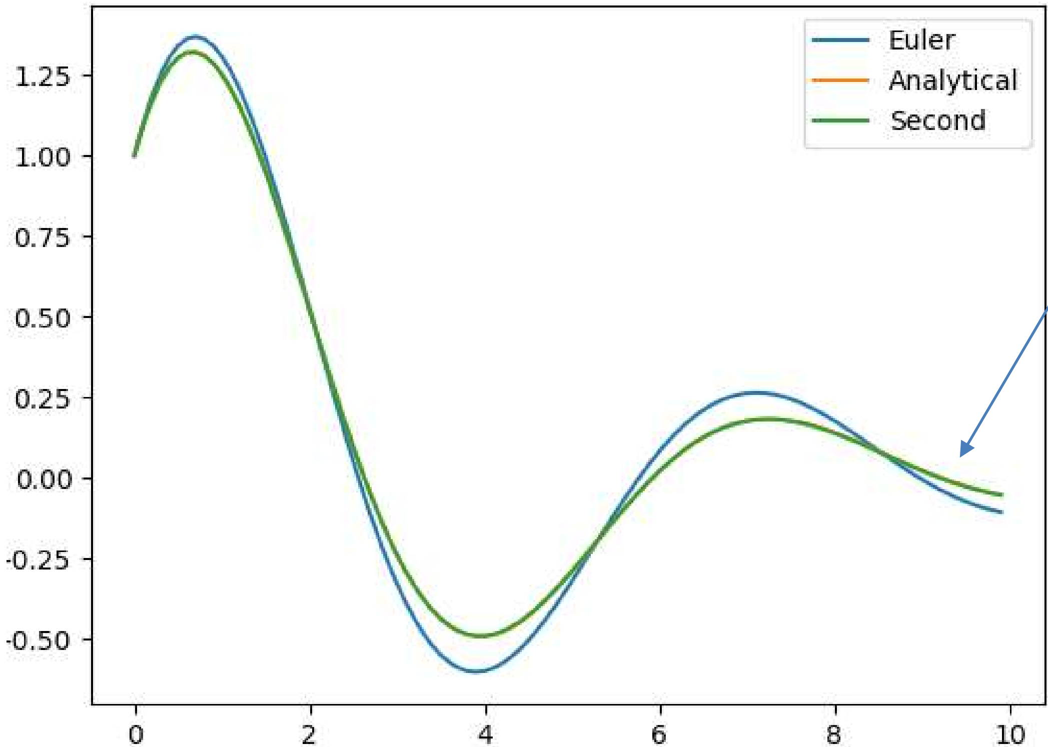
$$\frac{d^2\boldsymbol{x}}{dt^2} = \frac{df(t,\boldsymbol{x})}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial \boldsymbol{x}} \cdot \frac{d\boldsymbol{x}}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial \boldsymbol{x}} \cdot f$$

$$\boxed{\boldsymbol{x}_{n+1} \approx \boldsymbol{x}_n + \Delta t \cdot f(t_n, \boldsymbol{x}_n) + \frac{(\Delta t)^2}{2} \cdot \left[\frac{\partial f}{\partial t} + \frac{\partial f}{\partial \boldsymbol{x}} \cdot f\right]_{t=t_n, \boldsymbol{x}=\boldsymbol{x}_n}}$$ (Eq.1)

例題に適用してみると

この場合、$\dot{\boldsymbol{x}} = f(t, \boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ なので、2次まで考えると、

$$\boldsymbol{x}_{n+1} \approx \boldsymbol{x}_n + \Delta t\, \boldsymbol{A}\boldsymbol{x}_n + \frac{\Delta t^2}{2}\boldsymbol{A}^2\boldsymbol{x}_n = \left(\boldsymbol{I} + \Delta t\, \boldsymbol{A} + \frac{\Delta t^2}{2}\boldsymbol{A}^2\right)\boldsymbol{x}_n$$



解析解と2次近似解とは重なっていて目視では区別がつかない

このような2次近似解は、「テイラー級数3項法」と呼ばれる

しかし、高次微分を求めるのは一般には容易ではない

修正オイラー法(1)

$$\boldsymbol{x}_{n+1} \approx \boldsymbol{x}_n + \Delta t \cdot f(t_n, \boldsymbol{x}_n) + \frac{(\Delta t)^2}{2} \cdot \underbrace{\left[\frac{\partial f}{\partial t} + \frac{\partial f}{\partial \boldsymbol{x}} \cdot f \right]_{t=t_n, \boldsymbol{x}=\boldsymbol{x}_n}} \qquad \text{(Eq.1)}$$

この偏微分を直接計算せずに近似する方法がある！

(ステップ1)以下の2つのベクトル $\boldsymbol{k}_1, \boldsymbol{k}_2$ を計算する

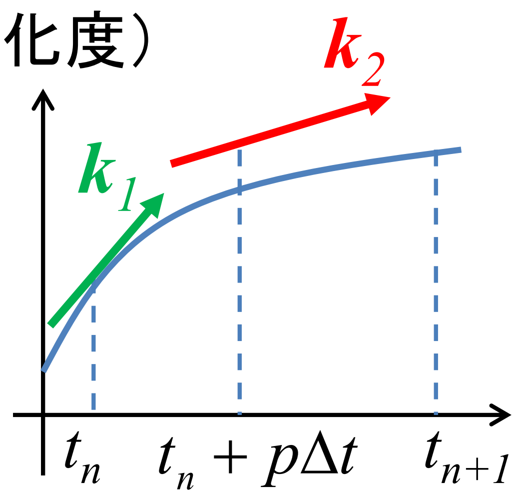
$$\mathbf{k}_1 = f(t_n, \mathbf{x}_n)$$

$(t_n, \mathbf{x}_n)$ での傾き(変化度)

$$\mathbf{k}_2 = f(t_n + \underbrace{p \cdot \Delta t}, \ \mathbf{x}_n + \underbrace{p \cdot \Delta t \cdot \mathbf{k}_1})$$

$p \cdot \Delta t$ だけ進んだ時刻での傾き $(0 < p \leq 1)$

(少しずらした)2時点で傾きを計算するということ



修正オイラー法（２）

（ステップ2） $\boldsymbol{k}_2$ を $t = t_n, \boldsymbol{x} = \boldsymbol{x}_n$ のまわりで１次近似してみる

$$\mathbf{k}_2 = f\left(t_n + p \cdot \Delta t,\ \ \mathbf{x}_n + p \cdot \Delta t \cdot \mathbf{k}_1\right)$$

$$\approx f\left(t_n,\ \ \mathbf{x}_n\right) + \frac{\partial f}{\partial t} p \cdot \Delta t + \frac{\partial f}{\partial \mathbf{x}} p \cdot \Delta t \cdot \mathbf{k}_1$$

$$= f\left(t_n,\ \ \mathbf{x}_n\right) + \frac{\partial f}{\partial t} p \cdot \Delta t + \frac{\partial f}{\partial \mathbf{x}} p \cdot \Delta t \cdot f\left(t_n,\ \ \mathbf{x}_n\right)$$

修正オイラー法(3)

(ステップ3)2ベクトル $\boldsymbol{k}_1, \boldsymbol{k}_2$ の重み和によって $\boldsymbol{x}_{n+1}$ を近似することを考える

$$\mathbf{x}_{n+1} \approx \mathbf{x}_n + \Delta t \cdot \left(\alpha \cdot \mathbf{k}_1 + \beta \cdot \mathbf{k}_2 \right)$$

$\boldsymbol{k}_1, \boldsymbol{k}_2$ を代入して整理すると

$$\mathbf{x}_{n+1} \approx \mathbf{x}_n + \Delta t \cdot (\alpha + \beta) f(t_n,\ \mathbf{x}_n) + \Delta t^2 \cdot \beta \cdot p \cdot \left[\frac{\partial f}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \cdot f \right]_{t=t_n,\ \mathbf{x}=\mathbf{x}_n}$$

$\Updownarrow$ 係数比較

$$\boldsymbol{x}_{n+1} \approx \boldsymbol{x}_n + \Delta t \cdot f(t_n, \boldsymbol{x}_n) + \frac{(\Delta t)^2}{2} \cdot \left[\frac{\partial f}{\partial t} + \frac{\partial f}{\partial \boldsymbol{x}} \cdot f \right]_{t=t_n, \boldsymbol{x}=\boldsymbol{x}_n} \quad \text{(Eq.1)}$$

$$\boxed{\alpha + \beta = 1 \quad , \quad \beta \cdot p = \tfrac{1}{2}}$$

一致条件

修正Euler法とHeun法

２次近似に一致する条件

$$\boxed{\alpha + \beta = 1 \quad , \quad \beta \cdot p = 1/2}$$

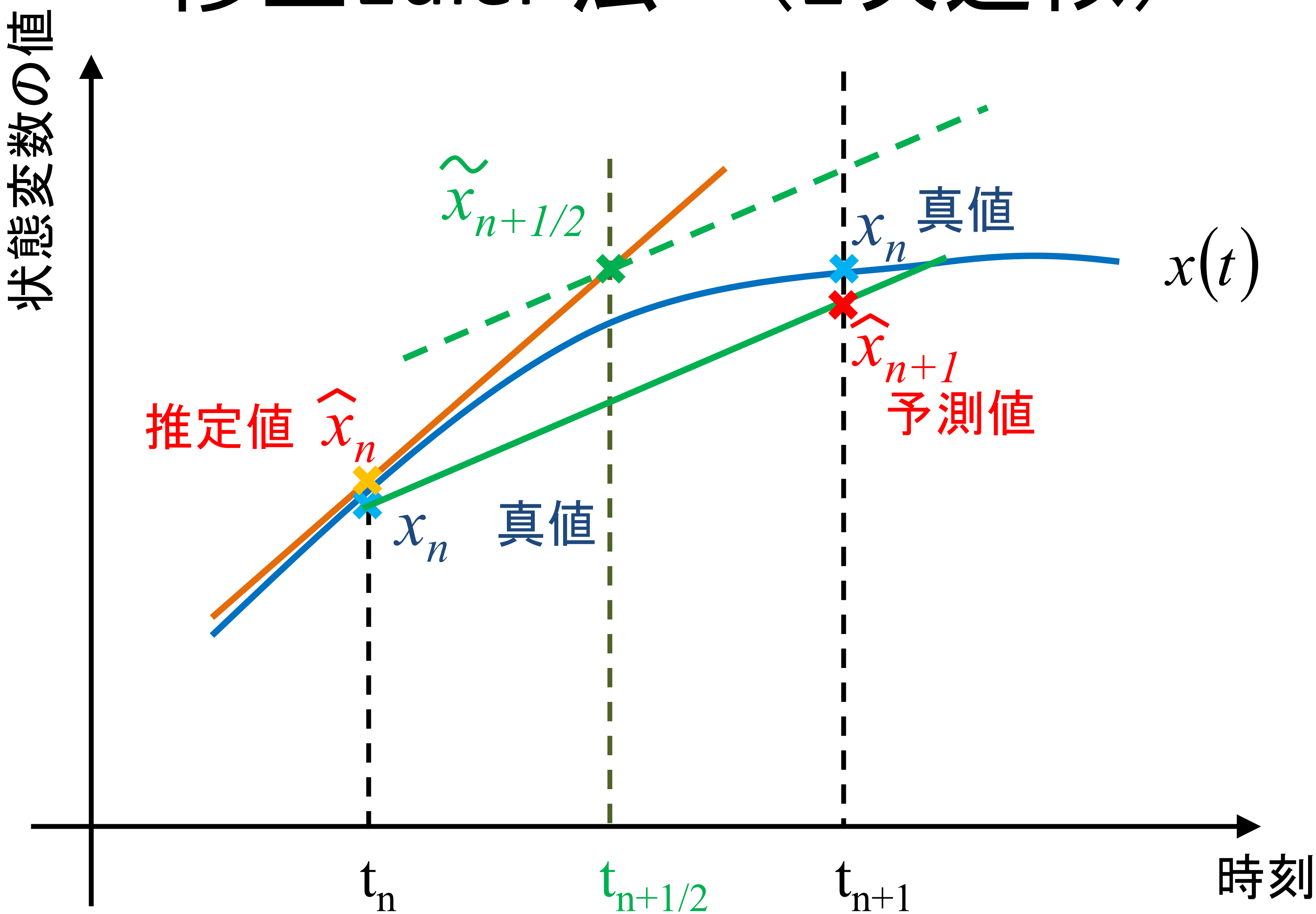
を満たす (α, β, p)の組み合わせは無数にあるが、

特に以下の場合が有名

- 修正Euler法 : $\alpha = 0, \beta = 1, p = 1/2$
- Heun法： $\alpha = 1/2, \beta = 1/2, p = 1$

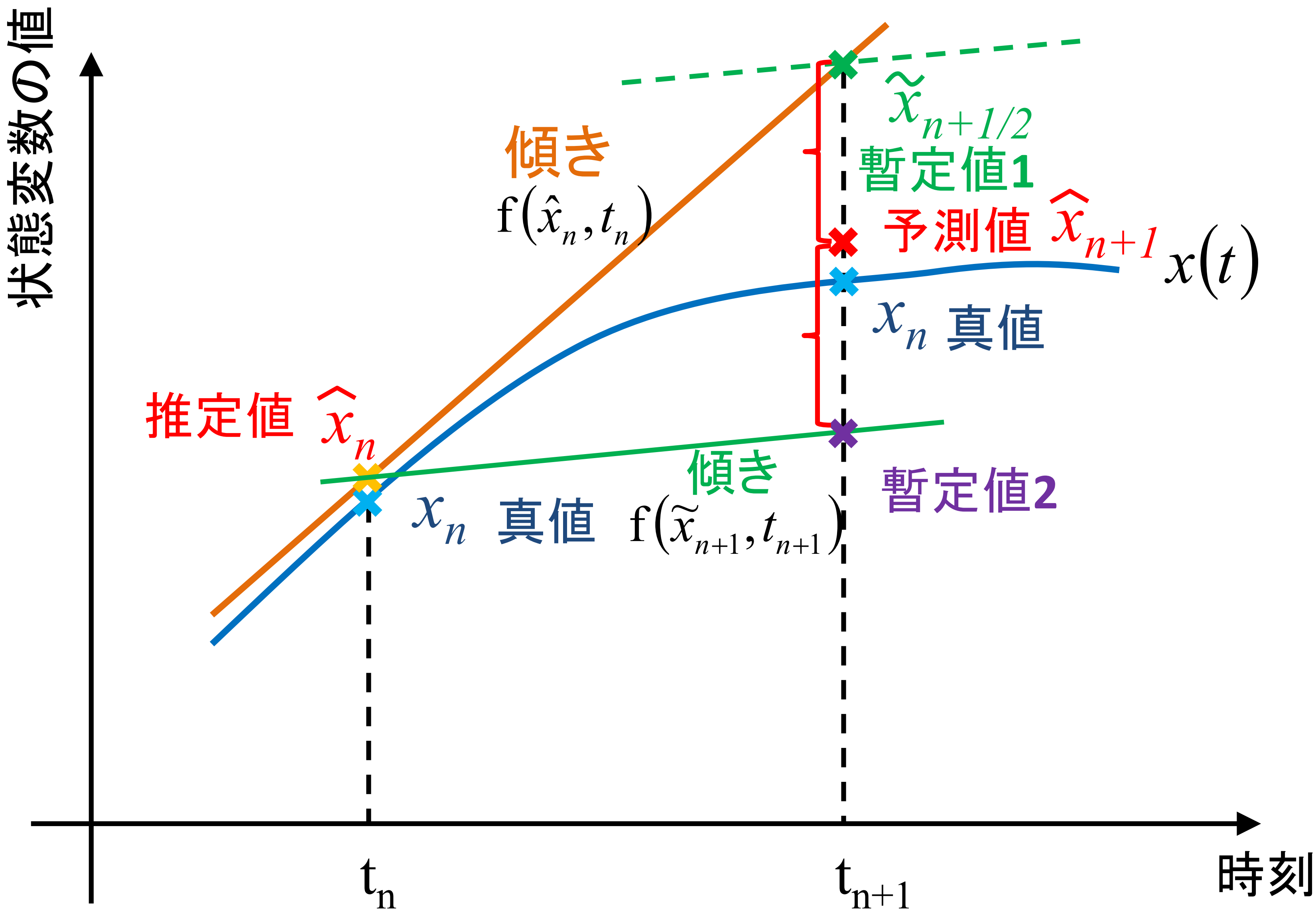
「２次のRunge-Kutta法」とも呼ばれる

修正Euler 法(*) （2次近似）



(*)数力演習の教科書では、「改良オイラー法」と呼んでいる

Heun 法(*)（2次近似）



(**)数力演習の教科書では、こちらを「修正オイラー法」と呼んでいる

修正Euler法・Heun法のプログラム（重要な部分のみ抜粋）

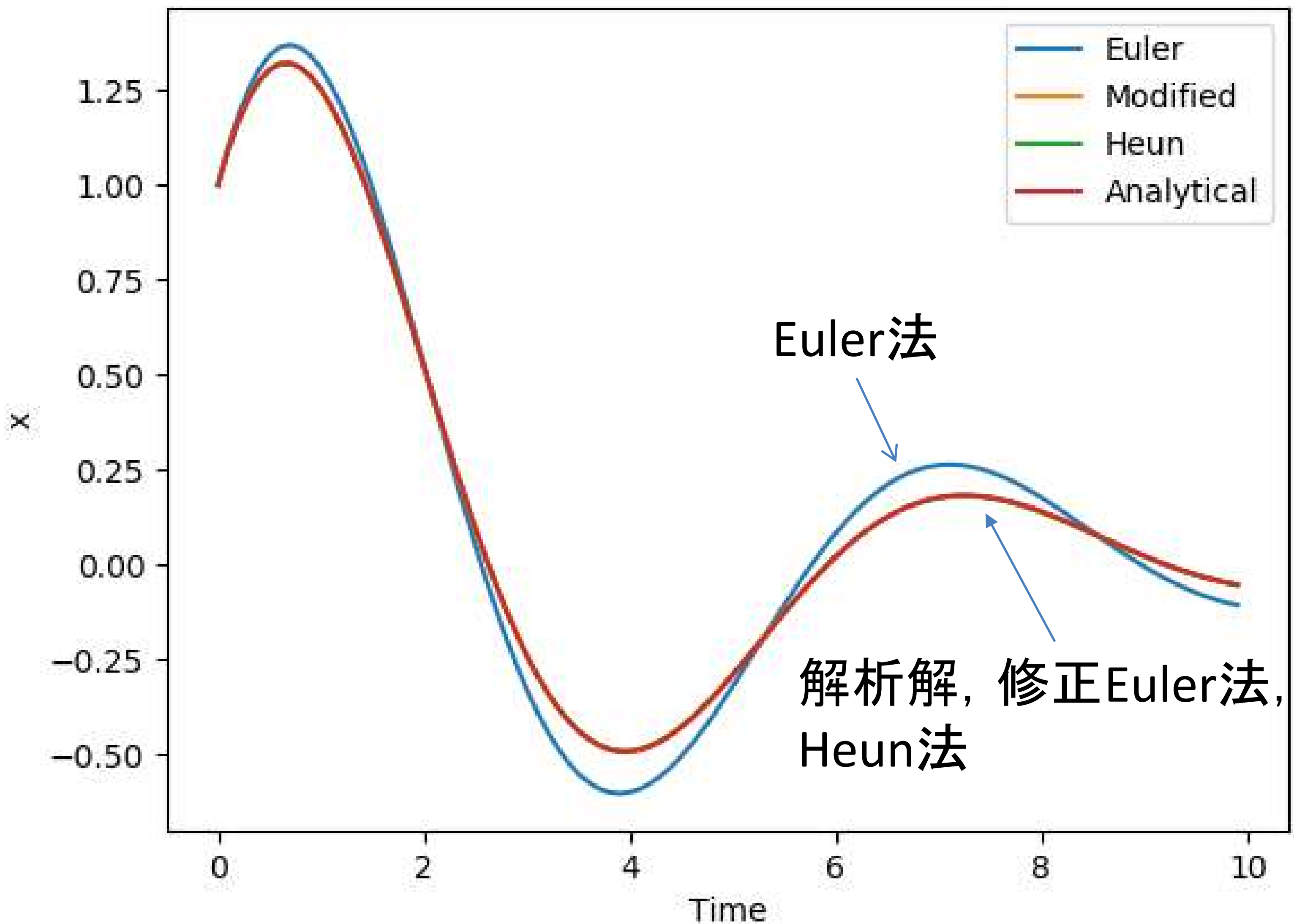
修正Euler法

```
for n in range(N-1):
    K1 = dot(A,Xmo[n,:])
    K2 = dot(A,Xmo[n,:]+0.5*dlt*K1)
    Xmo[n+1,:] = Xmo[n,:] + dlt*K2
```

Heun法

```
for n in range(N-1):
    K1 = dot(A,Xhe[n,:])
    K2 = dot(A,Xhe[n,:]+dlt*K1)
    Xhe[n+1,:] = Xhe[n,:] + 0.5*dlt*(K1+K2)
```

修正Euler法、Euler法、解析解の比較



Runge-Kutta法

- 修正Euler法、Heun法は、2か所で傾きを計算することで2次近似が行えた
- では、より多くの時点で傾きを計算し、重み付き足し算を行えば高次の近似ができるのか？
- 答えは.. **Yes**
- 性能と計算コストの観点から、4次のRunge-Kutta法がよく用いられる
  - 単に "Runge-Kutta法" と言った場合、暗に 4次の Runge-Kutta 法を指すことが多い
  - 2次の場合と同様、無数の組み合わせがあるが、1/6公式、1/8公式と呼ばれるものが有名

4次のRunge-Kutta法（1/6公式）

- 以下の$\mathbf{k}_1,\mathbf{k}_2,\mathbf{k}_3,\mathbf{k}_4$を計算

$$\mathbf{k}_1 = f\left(t_n, \mathbf{x}_n\right)$$

$$\mathbf{k}_2 = f\left(t_n + \Delta t/2,\ \mathbf{x}_n + \Delta t/2 \cdot \mathbf{k}_1\right)$$

$$\mathbf{k}_3 = f\left(t_n + \Delta t/2,\ \mathbf{x}_n + \Delta t/2 \cdot \mathbf{k}_2\right)$$

$$\boldsymbol{k}_4 = f\left(t_n + \Delta t,\ \boldsymbol{x}_n + \Delta t \cdot \boldsymbol{k}_3\right)$$

- 次式の重み付き足し算により$\boldsymbol{x}_{n+1}$を計算

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{\Delta t}{6} \cdot \left(\mathbf{k}_1 + 2 \cdot \mathbf{k}_2 + 2 \cdot \mathbf{k}_3 + \mathbf{k}_4\right)$$

減衰振動の場合の4次のRunge-Kutta法

- 主な変更部分のみ

```
for n in range(N-1):
    K1 = dot(A,Xrk[n,:])
    K2 = dot(A,Xrk[n,:]+0.5*dlt*K1)
    K3 = dot(A,Xrk[n,:]+0.5*dlt*K2)
    K4 = dot(A,Xrk[n,:]+dlt*K3)
    Xrk[n+1,:]=Xrk[n,:]+dlt/6*(K1+2*K2+2*K3+K4)
```

非線形な常微分方程式の例

下の常微分方程式（Lorenz方程式）を数値的に解くプログラムを作成してみよう。

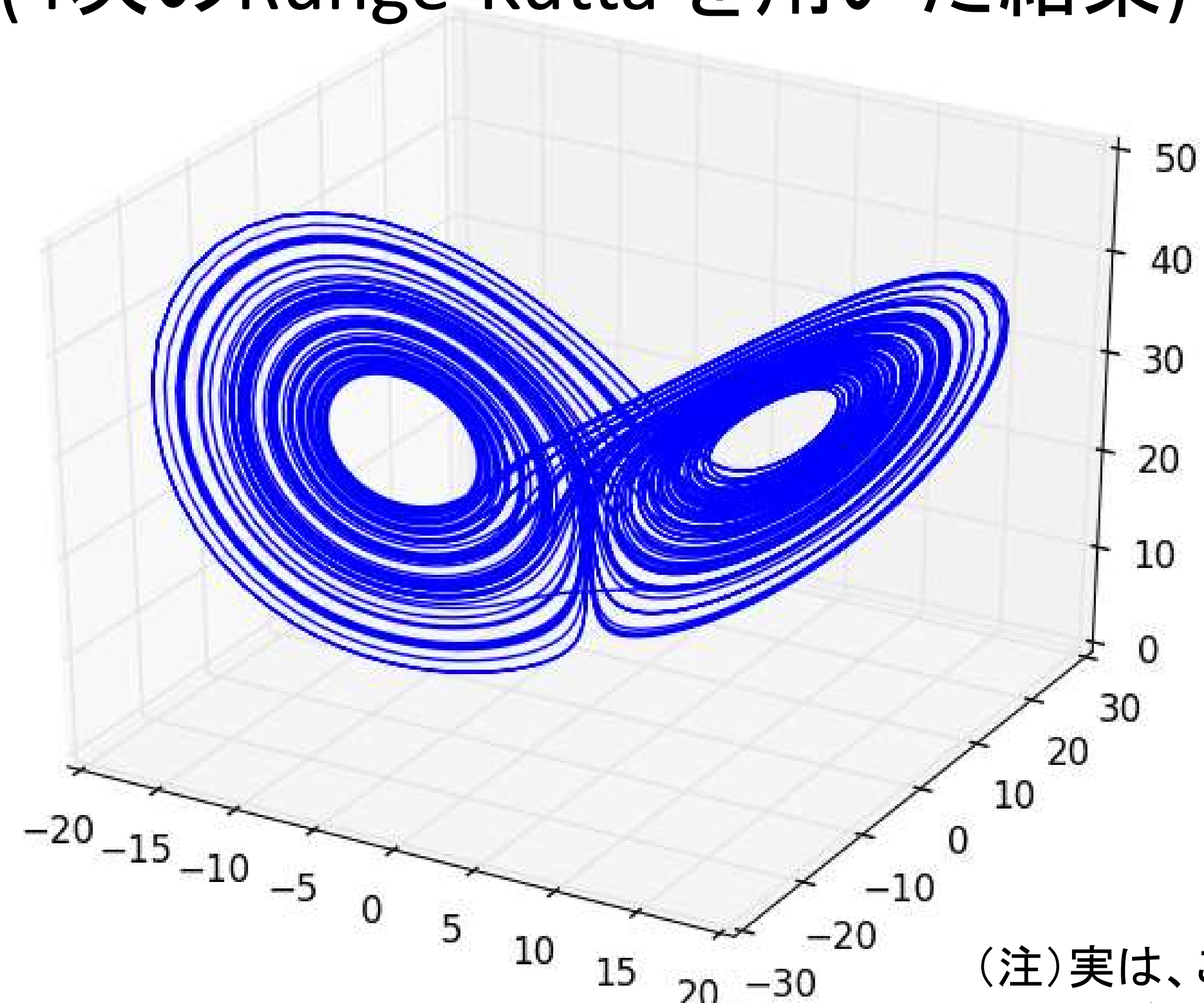
$$
\begin{cases}
\dfrac{dx_1}{dt} = \sigma \cdot (x_2 - x_1) \\[2ex]
\dfrac{dx_2}{dt} = x_1 \cdot (\rho - x_3) - x_2 \\[2ex]
\dfrac{dx_3}{dt} = x_1 \cdot x_2 - \beta \cdot x_3
\end{cases}
$$

パラメータの値は、σ=10 , β=8/3, ρ=28 とする。状態変数 x_1 , x_2 , x_3 の初期値は適当に与えてよい。

Lorenz方程式のRunge-Kutta法による数値計算（主要部分）

```
# パラメータの値をグローバル変数として設定
s = 10.0
b = 8.0/3.0
r = 28.0

# 状態ベクトルの時間微分を計算する関数
def deriv_lorenz(x):
    dx = s*(x[1]-x[0])
    dy = x[0]*(r-x[2])-x[1]
    dz = x[0]*x[1]-b*x[2]
    dxdydz = array([dx,dy,dz])
    return dxdydz

# Runge-Kutta 法のメインループ
for i in range(numt-1):
    k1 = deriv_lorenz(X[i,:])
    k2 = deriv_lorenz(X[i,:]+0.5*dt*k1)
    k3 = deriv_lorenz(X[i,:]+0.5*dt*k2)
    k4 = deriv_lorenz(X[i,:]+dt*k3)
    X[i+1,:] = X[i,:] + dt/6 * (k1+2*k2+2*k3+k4)
```

Lorenz System
(4次のRunge-Kutta を用いた結果)



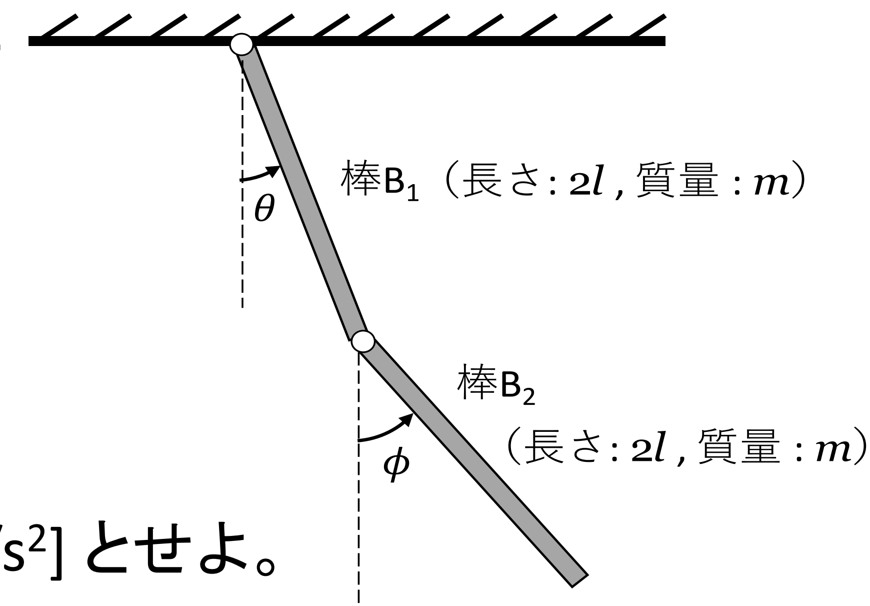
（注）実は、これでも精度が不十分だったりするが、系の定性的挙動はわかる

$\sigma=10$, $\beta=8/3$, $\rho=28$

今日の課題：数力演習期末テスト第5問

ともに長さ$2l$, 質量m の2本の剛体棒B_1, B_2を連結した二重振り子を考える。天井とB_1, およびB_1とB_2とは摩擦の無いヒンジで連結されており、棒の長さは無視でき密度は一定であるとする。2つの振り子は同一鉛直面内で振動し、各振り子の鉛直軸からの変位角をそれぞれ、θ, ϕ として以下の問いに答えよ。重力加速度はgとする。

（２）ラグランジュ方程式を用いてこの系の運動方程式を求めよ。



（５）設問（２）で求めた運動方程式をRunge-Kuttta法を用いて数値的に解け。

$m = 1$[kg], $l = 1$[m], $g = 9.8$[m/s^2] とせよ。

課題補足

元の運動方程式:

$$\frac{16}{3}l\ddot{\theta} + 2l\ddot{\phi}\cos(\theta-\phi) + 2l\dot{\phi}^2\sin(\theta-\phi) + 3g\sin\theta = 0$$

$$\frac{4}{3}l\ddot{\phi} + 2l\ddot{\theta}\cos(\theta-\phi) - 2l\dot{\theta}^2\sin(\theta-\phi) + g\sin\phi = 0$$

を、$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x})$ の形式に変更する。ただし、$\boldsymbol{x} = \left[\theta, \phi, \dot{\theta}, \dot{\phi}\right]^T$ とする

（1）頑張って手計算で、$\frac{d}{dt}\begin{bmatrix}\theta \\ \phi \\ \dot{\theta} \\ \dot{\phi}\end{bmatrix} = \begin{bmatrix}\dot{\theta} \\ \dot{\phi} \\ \cdots \\ \cdots\end{bmatrix}$ の形式に直せば、右辺が $\boldsymbol{f}(t, \boldsymbol{x})$ になる。

（2）元の式を整理すると、

$$\begin{bmatrix}\frac{16}{3} & 2\cos(\theta-\phi) \\ 2\cos(\theta-\phi) & \frac{4}{3}\end{bmatrix}\begin{bmatrix}\ddot{\theta} \\ \ddot{\phi}\end{bmatrix} = -\frac{g}{l}\begin{bmatrix}3\sin\theta \\ \sin\phi\end{bmatrix} + 2\sin(\theta-\phi)\begin{bmatrix}-\dot{\phi}^2 \\ \dot{\theta}^2\end{bmatrix}$$

となるはずなので、$\ddot{\theta}$ と $\ddot{\phi}$ は、（毎回）この1次方程式を解く。

今回の課題（続き）

- homework07.py という名前を付けて、ITC-LMSから提出
- 締切: 7月17日午前8時（1週間延長）
- 発展課題:
  - どのような初期条件でカオス的な挙動になるか。
  - 逆に、どんな条件で線形近似が可能か。
  - 位置・運動エネルギーの変化
  - アニメーション表示