微分方程式・偏微分方程式

田浦健次朗
電子情報工学科

東京大学

Contents

Contents

微分方程式とは

- 未知の関数 (例えば x に関する未知の関数 $y(x)$) に関する方程式で, x, $y(x)$, $y'(x)$, $y''(x)$, $\dots$ などを含んだ式
  - しばしば $y(x)$ の (x) を省略して単に y と書く
  - y が 1 変数のとき, 「常」微分方程式
  - 方程式の「階数」は y の最大の微分の回数 ($'$ の個数)
  - 例 (1 階の常微分方程式)

$$y' - xy = 0$$

- 微分方程式を「解く」とは?
  - y を x の関数として求めること
  - 例 (上記の解は実は)

$$y = Ce^{\frac{1}{2}x^2} \quad (C \text{ は定数})$$

  - どう求めたかは重要でないのでスキップ

初期値

- 通常,

$$y' - xy = 0$$

のような式だけでは y は一意に決まらない

- この解は以下 (C は任意の定数) だった

$$y = Ce^{\frac{1}{2}x^2} \quad (C \text{ は定数})$$

- そこで通常, ある $x = a$ での y の値 $y(a) = c$ (初期値) を一緒に与えて解く
- 例

$$y(0) = 1$$

という初期値を与えると,

$$y = e^{\frac{1}{2}x^2}$$

記号的な「解く」と数値計算での「解く」

- 記号的な求解 $= y(x)$ を x の「式」として表す
  - 通常, 数学の問題で解くといえばこちら
- 数値的な求解 $= y(x)$ の近似値を任意の x に対して求める
  - 通常, コンピュータで解くといえばこちら

Contents

常微分方程式の数値的な求解

- 以下の形 $(y' = \dots)$ の 1 階の常微分方程式

$$\begin{aligned} \frac{dy}{dx} &= f(x, y), \\ y(a) &= c \end{aligned}$$

  - y : 未知の, x の関数 $(y(x))$
  - a, c : 与えられた定数 (初期値)
  - f : 与えられた関数

常微分方程式の数値的な求解

- 以下の形 ($y' = \ldots$) の 1 階の常微分方程式

$$\begin{aligned} \frac{dy}{dx} &= f(x, y), \\ y(a) &= c \end{aligned}$$

  - y : 未知の, x の関数 ($y(x)$)
  - a, c : 与えられた定数 (初期値)
  - f : 与えられた関数
- 基本アイデア:
  - ある x における y の値がわかれば右辺が計算できる
  - $\rightarrow$ その時点での $\frac{dy}{dx}$ がわかる
  - $\rightarrow$ 少し異なる x に対する y (つまり $y(x + h)$) がわかる

常微分方程式の数値的な求解

- 以下の形 ($y' = \ldots$) の 1 階の常微分方程式

$$\begin{aligned} \frac{dy}{dx} &= f(x, y), \\ y(a) &= c \end{aligned}$$

  - y : 未知の, x の関数 ($y(x)$)
  - a, c : 与えられた定数 (初期値)
  - f : 与えられた関数
- 基本アイデア:
  - ある x における y の値がわかれば右辺が計算できる
  - $\rightarrow$ その時点での $\frac{dy}{dx}$ がわかる
  - $\rightarrow$ 少し異なる x に対する y (つまり $y(x+h)$) がわかる
- 上記を形式的に実行する方法 $\rightarrow$ 離散化

離散化

- 数式で書けば: h を十分小さく取れば,

$$\frac{y(x+h)-y(x)}{h} \approx f(x, y(x)) \tag{1}$$

$$\therefore y(x+h) \approx y(x) + f(x, y(x))h$$

$$y(a) \to y(a+h) \to y(a+2h) \to \cdots$$

- キーは式 (1) で，元の微分方程式の「離散化」という

プログラム化

$$
\begin{aligned}
y(a+h) &\approx y(a) + f(a, y(a))h \\
y(a+2h) &\approx y(a+h) + f(a+h, y(a+h))h \\
y(a+3h) &\approx y(a+2h) + f(a+2h, y(a+2h))h \\
&\cdots
\end{aligned}
$$

- 大雑把には,

```
繰り返す:
    y = y + f(x, y) * h
    x = x + h
```

ちゃんとプログラム化

```
# 初期値
x = a
y = c
h = (b - a) / n_steps    # 刻み幅
for i in range(n_steps):
    y = y + f(x, y) * h
    x = x + h
# この時点で y ≈ y(b)
```

- 誤差の積もりにくい計算の仕方 (離散化の方法) とかありますが，基本はこれだけ!

シミュレーションでよくある状況: x が時刻で y が位置

- 時刻は t, 位置は u で表すことにする
- 微分方程式：時刻と位置 $\to$ 速度

$$\frac{du}{dt} = f(t, u)$$

- 離散化：ある時刻における位置 $\to$ 一瞬後の位置

$$u(t+\Delta t) \approx u(t)+f(t,u)\Delta t$$

```
t = a
u = c
dt = (b - a) / n_steps
for i in range(n_steps):
    # 位置+速度× 微小時間
    u = u + f(t, u) * dt
    t = t + dt
# u ≈ 時刻b における位置
```

2階だったら?

- 力学の多くの微分方程式 ($ma = F$) は,

$$\begin{aligned} \text{加速度} &= f(\text{時刻}, \text{位置}, \text{速度}) \\ \ddot{x} &= f(t, x, \dot{x}) \end{aligned}$$

- これのプログラムは

```
t = a
u = .. # 初期位置
v = .. # 初速度
dt = (b - a) / n_steps
for i in range(n_steps):
    u = u + v * dt
    v = v + f(t, u, v) * dt
    t = t + dt
```

感染拡大の数理モデル

- 人口を以下に分類 (t：時刻)
  - $S(t)$(useptible)：これから感染するかもしれない人 (まだ陰性)
  - $I(t)$(nfected)：感染中
  - $R(t)$(ecovered)：治った
- 方程式

$$\begin{array}{rcll} S'(t) & = & -\beta SI & S \text{ と } I \text{ が出会うと一定確率で感染} \\ I'(t) & = & \beta SI - \gamma I & \\ R'(t) & = & \gamma I & \text{感染者は一定時間に一定確率で治る} \end{array}$$

- 注: $S' + I' + R' = 0$, すなわち $S + I + R =$ 一定

感染拡大の数理モデル

- プログラムも簡単
- ちなみに,

$$I'(t) = (\beta S - \gamma)I$$

なので,

$$\beta S - \gamma > 0$$

つまり

$$\frac{\beta S}{\gamma} > 1$$

だと感染拡大 (左辺 = 基本再生産数)

Contents

まだできないこと

- これでシミュレートできるのはせいぜい数個の質点 (または数個のパラメータ (e.g., 角度) だけで決まる系) の動き
- 物理の多くの問題で求めたいものは,「時刻」と「場所」の関数 (「場」)
  - 電場，磁場，重力場，鉄板表面の温度，速度の分布，圧力の分布，力の分布，電子の波動関数 (存在確率), ...
  - 記号で書けば（$u(t)$ ではなく），$u(x, y, z, t)$
- 別の言い方をすると，時間の関数が無数に (x, y, z ごとに) ある

場を記述する方程式

- 「時間 (t)」と「場所 (x, y, z)」の関数 $u(x, y, z, t)$ の方程式
- または定常状態を求める問題では場所だけの関数 $u(x, y, z)$ のこともある
- 式としては $\frac{\partial u}{\partial t}$, $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial y}$, などが入り乱れた式 (偏微分方程式) になる

物理で目にする例

- 熱伝導: $u(x, y, z, t) =$ 場所 (x, y, z), 時刻 t における温度

$$c\frac{\partial u}{\partial t} = \lambda\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$$

- 右辺の $(\cdots)$ の部分は他の方程式でも何故か非常によく現れるので，特別な記号 (Δ または ∇^2) が用意されている．

$$\Delta u \equiv \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

- これを使えば,

$$c\frac{\partial u}{\partial t} = \Delta u$$

物理で目にする例

- マックスウェルの方程式 (真空中)

$$
\begin{aligned}
\nabla \cdot B &= 0 \\
\nabla \times E &= -\frac{\partial B}{\partial t} \\
\nabla \cdot E &= \frac{1}{\varepsilon}\rho \\
\nabla \times B &= \mu j + \mu\varepsilon E
\end{aligned}
$$

注: B, E はベクトル場

$$
\begin{aligned}
\nabla \cdot u &\equiv \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} \\
\nabla \times u &\equiv \ldots \text{教科書を見ま笑} \ldots
\end{aligned}
$$

物理で目にする例

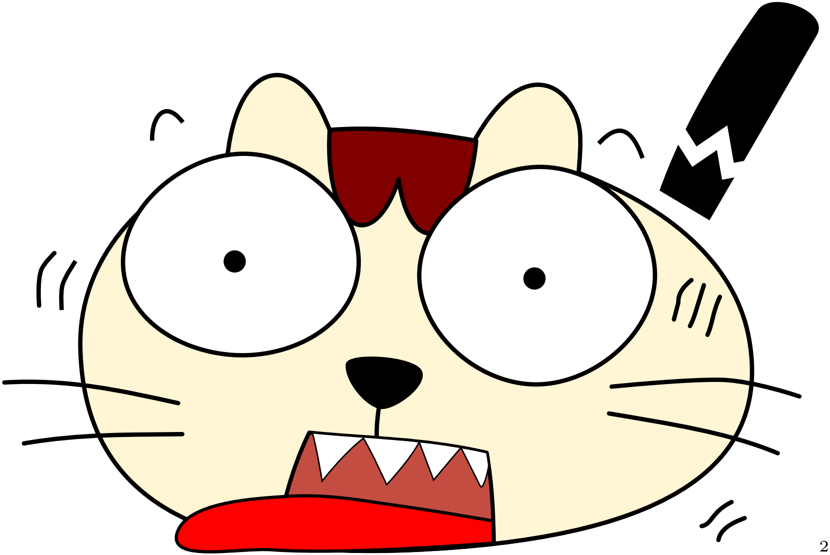
- 流体の速度場の方程式 (ナビエストークス)

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\frac{1}{\rho}\nabla p + \nu\nabla^2 u + F$$

- 波動方程式 (音とか光とか)

$$\frac{1}{s^2}\frac{\partial^2 u}{\partial t^2} = \Delta u$$

コワイ??



しかし恐れる必要はない

- 根本的な考えは同じ; (常) 微分方程式がたくさん並んでいるだけと思えば良い
- 例えば 1 次元の場 $u(x,t)$ の方程式が,

$$\frac{\partial u}{\partial t} = \cdots$$

と書けているならこれを用いて,

$$\begin{pmatrix} u(0,t) \\ u(0.1,t) \\ u(0.2,t) \\ u(0.3,t) \\ u(0.4,t) \\ \vdots \end{pmatrix} \to \begin{pmatrix} u(0,t+\Delta t) \\ u(0.1,t+\Delta t) \\ u(0.2,t+\Delta t) \\ u(0.3,t+\Delta t) \\ u(0.4,t+\Delta t) \\ \vdots \end{pmatrix} \to \begin{pmatrix} u(0,t+2\Delta t) \\ u(0.1,t+2\Delta t) \\ u(0.2,t+2\Delta t) \\ u(0.3,t+2\Delta t) \\ u(0.4,t+2\Delta t) \\ \vdots \end{pmatrix} \to \cdots$$

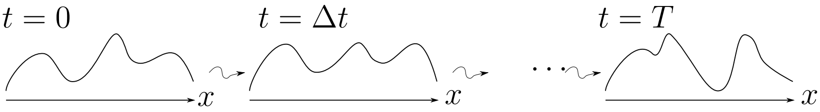
という計算をしていくだけ (リストや配列が必須かつ有用)

ともかく例を見る

簡単のため 1 次元空間の例題 (x は単なる実数):

$$
\begin{cases}
\dfrac{\partial u}{\partial t} &= 3\,\dfrac{\partial u}{\partial x} \quad (0 < x < 1), \\
u(0,t) &= 0 \\
u(1,t) &= 0 \\
u(x,0) &= f(x)
\end{cases}
$$

- u は「場所 (x)」と「時刻 (t)」の関数
- 1 次元 (x が単なる実数) だから視覚的には，弦が時間につれどう変化していくかを求める問題



離散化

- 微分方程式:

$$\frac{\partial u}{\partial t} = 3\,\frac{\partial u}{\partial x}$$

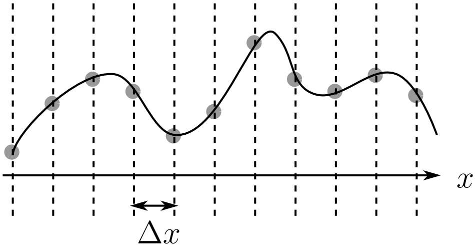
- 離散化:

$$\frac{u(x, t+\Delta t) - u(x,t)}{\Delta t} \approx 3\,\frac{\partial u(x+\Delta x, t) - u(x,t)}{\Delta x}$$

$$\therefore u(x, t+\Delta t) \approx u(x,t) + (u(x+\Delta x, t) - u(x,t))\frac{3\Delta t}{\Delta x}$$

離散化

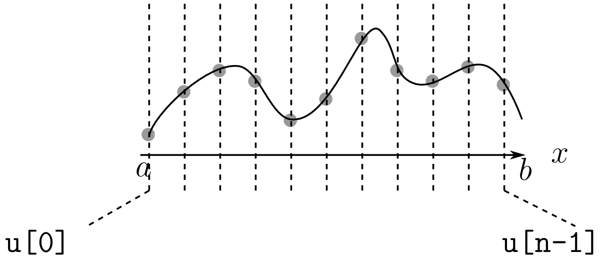
$$\therefore u(x, t+\Delta t) \approx u(x,t) + (u(x+\Delta x, t) - u(x,t))\frac{3\Delta t}{\Delta x}$$



- ある時点における自分と，右隣 $(+\Delta x)$ の人の値がわかれば，自分の一瞬 $(+\Delta t)$ 後の値がわかる

プログラム化

- 用意する変数: 先の●たちを格納した一次元の配列 `u`
- `u` の要素数を `n` とする



- これを踏まえ，更新式

$$u(x, t+\Delta t) = u(x,t) + (u(x+\Delta x, t) - u(x,t))\frac{3\Delta t}{\Delta x}$$

に相当する Python のプログラムは?

プログラム化

$$u(x, t + \Delta t) = u(x, t) + (u(x + \Delta x, t) - u(x, t))\frac{3\Delta t}{\Delta x}$$

```
for i in range(1, n-1):
  u[i] = u[i] + (u[i+1] - u[i]) * (3*dt/dx)
```

- ポイント: 端の点 (u[0], u[n-1]) はこの式では更新しない．何故か?

端の点の扱い

- 一般に端の点の扱いはよく考えて慎重に．いい加減にやるとおかしな結果になる
- まずプログラム上も，右端の点 (u[n-1]) を更新しようとすると困ったことになる

```
u[n-1] = u[n-1] + (u[n] - u[n-1]) * (3*dt/dx)
```

- u は n 要素．u[n] などという要素はない
- ただしそれが u[n-1] を更新しない理由ではない
- 微分方程式に立ち返る

$$\frac{\partial u}{\partial t} = 3\frac{\partial u}{\partial x} \quad (0<x<1) \tag{2}$$

そもそも式 (2) は，端点において成り立つとは言われていない ($0 < x < 1$ でしか成り立たない式を $x = 0, 1$ に適用するのは，純粋に間違い)

端の点の扱い

- 端の点は，別途与えられた条件で求める
- 本問題の場合，$u(0,t) = u(1,t) = 0$ だった
- よって，

```
for i in range(1, n-1):
  u[i] = u[i] + (u[i+1] - u[i]) * (3*dt/dx)
u[0] = u[n-1] = 0
```

が与えられた条件に忠実な，正しい更新式

- 本問においては, `u[0]`, `u[n-1]` は常に 0 なのだから，実は毎回更新する必要もない

```
for i in range(1, n-1):
  u[i] += (u[i+1] - u[i]) * (3*dt/dx)
```

(注: `A += B` は，`A = A + B` と同じ意味)

numpy の配列機能を使ってスッキリと

- for 文を使った更新式:

```
for i in range(1, n-1):
  u[i] += (u[i+1] - u[i]) * (3*dt/dx)
```

- 同じことは numpy の配列に対する`+`, `*` などを使ってより簡潔に書ける

```
u[1:n-1] += (u[2:n] - u[1:n-1]) * (3*dt/dx)
```

- 簡潔なだけでなく，圧倒的に ($>$ 100 倍) 速い
- 10000 要素の更新に必要な時間
  - for 文: 2×10^{-2} 秒程度
  - 配列で: 8×10^{-5} 秒程度