

第4回輪講資料

4321 野秋 琳太郎

2025 年 11 月 15 日

指導教員 宮田 尚起

1. はじめに

四年生ゼミでは高周波回路の勉強が始まった。これまでの輪講では先輩方の研究内容がつかみにくかったため、しばらくはそれらを自分で再現しつつ発表していく方針とした。今回は、前回のゼミ内容とも流れが近く、シミュレーション環境を整える手間も比較的少なく済みそうだったことから、三浦先輩の研究を題材にして調べることにした。

2. やりたいこと

電信方程式は、分布定数回路のキャパシタが線形な特性を持っていると仮定している。しかし、実際のキャパシタは非線形な特性を持っており、その非線形性が超高周波 (数百 GHz とか) においては無視できず、それによって信号線にソリトン波が発生することが知られている。そのため、非線形なキャパシタを考慮した電信方程式を導出する必要がある。非線形なキャパシタのモデルは先行研究でいくつか提案されているので、それで拡張した電信方程式を導出し、シミュレーションを行うことができれば研究としては完成ということになる。また、先輩の研究では”数値計算”も重きを置いているので、その方法も調べる必要がある。

3. KdV 方程式

KdV 方程式は、非線形波動方程式の一種であり、浅水波やプラズマ波動などの現象を記述するために用いられる。その一般的な形は式 1 の通りである。この解は解析的に求めることができ、図 1 に示すようなパルス状の波 (soliton) と呼ばれる特異な波動を記述することができる。ここで、式 1 の $u(x, t)$ を 2 とし、3 の初期条件を与えている。この波は、非線形性と分散性のバランスによって形成され、他の波と衝突しても形状を保つ特性を持つ。例えば、水面に 2 つの石を別の場所に投げ込んだときに発生する波は、衝突しても元の形状を保ちながら進む。これはこのソリトン波の特性によるものである。KdV 方程式が非線形はどうの中でも広い応用範囲を持つことと、ソリトン波が確認できていることから、非線形なキャパシタを考慮した電信方程式を解ける形で導出する際に利用

できると考えられる。

$$\frac{\partial u(x, t)}{\partial t} + 6u \frac{\partial u(x, t)}{\partial x} + \frac{\partial^3 u(x, t)}{\partial x^3} = 0 \quad (1)$$

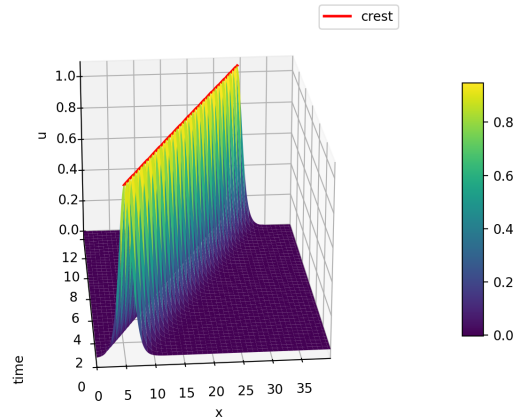


図 1. KdV 方程式の孤立波解の例

$$u(x, t) = 2k^2 \operatorname{sech}^2(k(x - 4k^2t - x_0)) \quad (2)$$

$$u(x, 0) = \frac{c}{2} \operatorname{sech}^2\left(\frac{\sqrt{c}}{2}(x - x_0)\right) \quad (3)$$

4. キャパシタモデルの改善

高周波回路、特に非線形伝送線路の解析において、キャパシタのモデルとしてバラクタダイオード (Varactor Diode) が重要な役割を果たす。これは半導体の pn 接合を利用した素子であり、逆バイアス電圧を変化させることで静電容量を制御できる特性を持つ。一般にバラクタダイオードは集中定数素子であるが、これを伝送線路に周期的に装荷し、その間隔が信号波長に対して十分に短い場合、巨視的には電圧依存性を持つ分布定数線路として扱うことが可能である。

このキャパシタの静電容量 $C(V)$ は、印加電圧 V に対して非線形な特性を持つ。この特性を表現するモデルとして、以下の式 (4) が広く用いられている。

$$C(V) = \frac{C_{J0}}{\left(1 + \frac{V}{\phi}\right)^\gamma} \quad (4)$$

ここで、 C_{J0} はバイアス電圧がゼロの時の接合容量、 ϕ は拡散電位（ビルトインポテンシャル）、 γ は接合容量係数である（例えば、シリコン階段接合では $\gamma = 0.5$ とされる）。この非線形キャパシタンスモデルを考慮して、より厳密な電信方程式を導出したい。

5. 電信方程式の拡張

図 2 のような線路を考えて、線形な電信方程式は以下の式 (5) で与えられる。ここで、 L, C, R, G はそれぞれ単位長さあたりのインダクタンス、キャパシタンス、抵抗、コンダクタンスであり、すべて定数である。

$$\frac{\partial^2 V}{\partial x^2} = LC \frac{\partial^2 V}{\partial t^2} + (RC + GL) \frac{\partial V}{\partial t} + GRV \quad (5)$$

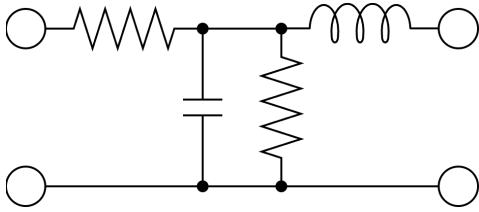


図 2. 分布定数回路 (1 ノード)

しかし、キャパシタンスが電圧の関数 $C(V)$ である場合、線形の方程式 (式 5) の定数 C を、単に $C(V)$ に書き換えるだけではよくない。電圧 V が時間変化すれば、それに応じて容量 $C(V)$ 自体も時間変化するため、時間微分の計算において積の微分と連鎖律の影響を受けるからである。非線形な電信方程式を導出するためには、完成された式 (2 階微分方程式) に代入するのではなく、その導出元となる 2 つの基礎方程式 (1 階連立偏微分方程式) に立ち返る必要がある。分布定数回路における電圧 V と電流 I の関係は、以下の 2 式で記述される。

- 電圧降下の式（線路の直列インピーダンスによる）：

$$-\frac{\partial V}{\partial x} = L \frac{\partial I}{\partial t} + RI \quad (6)$$

- 電流減少の式（線路の並列アドミタンスによる）：

$$-\frac{\partial I}{\partial x} = \frac{\partial Q}{\partial t} + GV \quad (7)$$

ここで、式 7 に含まれる変位電流項 $\frac{\partial Q}{\partial t}$ について

考える。キャパシタンスが電圧依存性 $C(V)$ を持つ場合、電荷 Q の時間変化は連鎖律により以下のように展開される。

$$\frac{\partial Q}{\partial t} = \frac{dQ}{dV} \frac{\partial V}{\partial t} = C(V) \frac{\partial V}{\partial t} \quad (8)$$

これを式 7 に代入すると、電流の変化は次式となる。

$$-\frac{\partial I}{\partial x} = C(V) \frac{\partial V}{\partial t} + GV \quad (9)$$

次に、電圧と電流を一本化して V だけの式（波動方程式）にするため、もう一方の基礎方程式である式 6 を変形する。式 6 の両辺を x で偏微分すると、以下の形になる。

$$-\frac{\partial^2 V}{\partial x^2} = L \frac{\partial}{\partial t} \left(\frac{\partial I}{\partial t} \right) + R \frac{\partial I}{\partial x} \quad (10)$$

この式の右辺にある $\frac{\partial I}{\partial x}$ の箇所に、先ほど導いた式 9 を代入することで、電流 I を消去できる。ここで特に注意すべきは、第 1 項の時間微分である。代入を行うと、積の微分と連鎖律により非線形項が現れる。

$$\frac{\partial}{\partial t} \left(C(V) \frac{\partial V}{\partial t} \right) = C(V) \frac{\partial^2 V}{\partial t^2} + \frac{dC(V)}{dV} \left(\frac{\partial V}{\partial t} \right)^2 \quad (11)$$

以上を整理することで式 12 に示す、拡張可能な電信方程式が得られる。

$$\frac{\partial^2 V}{\partial x^2} = L \left[C(V) \frac{\partial^2 V}{\partial t^2} + \frac{dC(V)}{dV} \left(\frac{\partial V}{\partial t} \right)^2 \right] + (RC(V) + GL) \frac{\partial V}{\partial t} + GRV \quad (12)$$

つぎに、キャパシタンスモデル (式 4) を用いて、 $C(V)$ とその微分 $\frac{dC(V)}{dV}$ を具体的に代入する。 $C(V)$ は 4 で与えられ、 $\frac{dC(V)}{dV}$ は以下ようになる。

$$\frac{dC(V)}{dV} = -\frac{\gamma C_{J0}}{\phi} \left(1 + \frac{V}{\phi} \right)^{-(\gamma+1)} \quad (13)$$

$$= -\frac{\gamma C_{J0}}{\phi \left(1 + \frac{V}{\phi} \right)^{\gamma+1}} \quad (14)$$

これを式 12 に代入すると、最終的な非線形電信方程式が得られる。

$$\begin{aligned} & \frac{\partial^2 V}{\partial x^2} \\ &= L \left[\frac{C_{J0}}{\left(1 + \frac{V}{\phi}\right)^\gamma} \frac{\partial^2 V}{\partial t^2} - \frac{\gamma C_{J0}}{\phi \left(1 + \frac{V}{\phi}\right)^{\gamma+1}} \left(\frac{\partial V}{\partial t}\right)^2 \right] \\ & \quad + \left(\frac{RC_{J0}}{\left(1 + \frac{V}{\phi}\right)^\gamma} + GL \right) \frac{\partial V}{\partial t} + GRV \end{aligned} \quad (15)$$

6. KdV 方程式にあてはめたい

非線形電信方程式 (式 15) から KdV 方程式を導出するためには、いくつかの近似と変数変換を行う必要がある。まず、線路は無損失であると仮定し、抵抗 R とコンダクタンス G をゼロに設定する。次に、キャパシタンス $C(V)$ とその微分 $\frac{dC(V)}{dV}$ をテイラー展開して 1 次の項まで近似する。さらに、通減摂動法を用いて、一方向に伝搬する波を抽出する。以下にその手順を示す。

6.1 基礎方程式の整理

本来、数百 GHz 帯の伝送線路は連続的な媒体ではなく、インダクタ L とキャパシタ C が間隔 h で並んだ離散的な格子の回路である。この回路の第 n 番目のノードにおける方程式は、キルヒホッフの法則より以下の差分方程式で記述される。

$$V_{n+1} + V_{n-1} - 2V_n = Lh \frac{d^2 Q_n}{dt^2} \quad (16)$$

(※左辺は隣接ノードとの電位差の変化、右辺は電流の時間変化に対応する)

ここで、空間的に連続な関数 $V(x, t)$ を仮定し、 $V_n(t) = V(x, t)$ 、 $V_{n\pm 1}(t) = V(x \pm h, t)$ と置く。 $V(x \pm h, t)$ を x の周りでテイラー展開すると以下のようになる。

$$\begin{aligned} V(x+h) &= \\ V(x) + h \frac{\partial V}{\partial x} + \frac{h^2}{2!} \frac{\partial^2 V}{\partial x^2} + \frac{h^3}{3!} \frac{\partial^3 V}{\partial x^3} + \frac{h^4}{4!} \frac{\partial^4 V}{\partial x^4} + \dots \end{aligned} \quad (17)$$

$$\begin{aligned} V(x-h) &= \\ V(x) - h \frac{\partial V}{\partial x} + \frac{h^2}{2!} \frac{\partial^2 V}{\partial x^2} - \frac{h^3}{3!} \frac{\partial^3 V}{\partial x^3} + \frac{h^4}{4!} \frac{\partial^4 V}{\partial x^4} - \dots \end{aligned} \quad (18)$$

これら 2 式の和をとると、奇数次の項が相殺し、偶数次の項が残る。

$$V(x+h) + V(x-h) = 2V(x) + h^2 \frac{\partial^2 V}{\partial x^2} + \frac{h^4}{12} \frac{\partial^4 V}{\partial x^4} + O(h^6) \quad (19)$$

これを変形して、差分方程式の左辺 (2 階中心差分) に対応させる。

$$V_{n+1} - 2V_n + V_{n-1} = h^2 \frac{\partial^2 V}{\partial x^2} + \frac{h^4}{12} \frac{\partial^4 V}{\partial x^4} + O(h^6) \quad (20)$$

これまでは $h \rightarrow 0$ とし、第 2 項以降を無視していたが、ソリトンのような分散性波動を記述する場合、 h は有限であるとして第 2 項 (h^4 の項) までを考慮する。

式 (20) を元の回路方程式 (16) に代入する。

$$h^2 \frac{\partial^2 V}{\partial x^2} + \frac{h^4}{12} \frac{\partial^4 V}{\partial x^4} \approx Lh \frac{\partial^2 Q}{\partial t^2} \quad (21)$$

両辺を h^2 で割り、整理すると次式が得られる。

$$\frac{\partial^2 V}{\partial x^2} + \frac{h^2}{12} \frac{\partial^4 V}{\partial x^4} = L \frac{\partial^2 Q}{\partial t^2} \quad (22)$$

このように、離散的な差分を近似する際にテイラー展開の高次項を残すことで、回路の分散性を表す空間 4 階微分項 $\frac{\partial^4 V}{\partial x^4}$ が自然に導かれる。

6.2 キャパシタの非線形性の近似

次に、右辺の電荷 $Q(V)$ を電圧 V でテイラー展開する。キャパシタの容量 $C(V)$ をテイラー展開して 1 次の項まで近似する。

$$C(V) \approx C_{J0} \left(1 - \gamma \frac{V}{\phi} \right) \quad (23)$$

これを電圧 V で積分し、電荷 $Q(V)$ の近似式を得る。

$$Q(V) = \int C(V) dV \approx C_{J0} V - \frac{\gamma C_{J0}}{2\phi} V^2 \quad (24)$$

式 (24) を式 (22) に代入し、線位相速度 $v_0 = 1/\sqrt{LC_{J0}}$ を用いて整理すると、以下の「分散項付き非線形波動方程式 (Boussinesq 型方程式)」が得られる。Boussinesq 型方程式は、非線形性と分散性の両方を含む波動現象を記述するために広く知られている。

$$\frac{1}{v_0^2} \frac{\partial^2 V}{\partial t^2} - \frac{\partial^2 V}{\partial x^2} - \frac{h^2}{12} \frac{\partial^4 V}{\partial x^4} - \frac{\gamma}{2\phi v_0^2} \frac{\partial^2 (V^2)}{\partial t^2} = 0 \quad (25)$$

6.3 Gardner-Morikawa 変換

式 (25) で与えられたブシネスク方程式から、一方向に伝播する波を記述する KdV 方程式を導出する。導出には、波と共に動く座標系への変換と、微小パラメータ ϵ を用いた摂動展開（通減摂動法）を用いる。

まず、空間座標 x と時間 t に対し、以下の独立変数変換を行う。

$$\xi = x - v_0 t, \quad \tau = \epsilon t \quad (26)$$

ここで、 v_0 は波の位相速度である。この変換により、微分演算子は連鎖律を用いて次のように書き換えられる。

$$\frac{\partial}{\partial x} = \frac{\partial}{\partial \xi}, \quad \frac{\partial}{\partial t} = -v_0 \frac{\partial}{\partial \xi} + \epsilon \frac{\partial}{\partial \tau} \quad (27)$$

特に、時間の 2 階微分については、 ϵ の 2 次以上の項を無視すると、以下の近似式が得られる。

$$\frac{\partial^2}{\partial t^2} \approx v_0^2 \frac{\partial^2}{\partial \xi^2} - 2v_0 \epsilon \frac{\partial^2}{\partial \xi \partial \tau} \quad (28)$$

これらの関係式を元のブシネスク方程式 (25) の各項に代入する。まず、線形項（第 1 項と第 2 項）は次のように整理される。

$$\begin{aligned} \frac{1}{v_0^2} \frac{\partial^2 V}{\partial t^2} - \frac{\partial^2 V}{\partial x^2} &\approx \frac{1}{v_0^2} \left(v_0^2 \frac{\partial^2 V}{\partial \xi^2} - 2v_0 \epsilon \frac{\partial^2 V}{\partial \xi \partial \tau} \right) - \frac{\partial^2 V}{\partial \xi^2} \\ &= -\frac{2\epsilon}{v_0} \frac{\partial^2 V}{\partial \xi \partial \tau} \end{aligned} \quad (29)$$

次に、分散項（第 3 項）と非線形項（第 4 項）については、 ϵ の最低次の項のみを考慮し、時間微分の主要項 $\partial_t^2 \approx v_0^2 \partial_\xi^2$ を用いることで次のように近似される。

$$-\frac{h^2}{12} \frac{\partial^4 V}{\partial x^4} \approx -\frac{h^2}{12} \frac{\partial^4 V}{\partial \xi^4}, \quad -\frac{\gamma}{2\phi v_0^2} \frac{\partial^2 (V^2)}{\partial t^2} \approx -\frac{\gamma}{2\phi} \frac{\partial^2 (V^2)}{\partial \xi^2} \quad (30)$$

これらを統合すると、 ξ, τ 座標系における方程式は以下ようになる。

$$-\frac{2\epsilon}{v_0} \frac{\partial^2 V}{\partial \xi \partial \tau} - \frac{h^2}{12} \frac{\partial^4 V}{\partial \xi^4} - \frac{\gamma}{2\phi} \frac{\partial^2 (V^2)}{\partial \xi^2} = 0 \quad (31)$$

この式を ξ で 1 回積分する。ここで、無限遠方 $\xi \rightarrow \pm\infty$ において V およびその導関数が 0 になるという境界条件を用いると、積分定数は 0 となる。

$$-\frac{2\epsilon}{v_0} \frac{\partial V}{\partial \tau} - \frac{h^2}{12} \frac{\partial^3 V}{\partial \xi^3} - \frac{\gamma}{2\phi} \frac{\partial (V^2)}{\partial \xi} = 0 \quad (32)$$

さらに、非線形項を $\partial_\xi (V^2) = 2V \partial_\xi V$ と変形し、全体

を $-\frac{v_0}{2\epsilon}$ 倍して整理することで、最終的に以下の KdV 方程式が得られる。

$$\frac{\partial V}{\partial \tau} + \frac{v_0 \gamma}{2\phi \epsilon} V \frac{\partial V}{\partial \xi} + \frac{v_0 h^2}{24\epsilon} \frac{\partial^3 V}{\partial \xi^3} = 0 \quad (33)$$

こうして、本当に解くべき KdV 方程式の形にたどり着いた。

7. おわりに

今回は、上記で導出した KdV 方程式を数値的に解き、ソリトン波の挙動を確認する。

7.1 参考文献

- 和達美樹『岩波講座現代の物理学 非線形波動』岩波書店 1992 年
- python で学ぶ計算物理 ドキュメント » 5. 偏微分方程式 » 5.2. 【例題】 KdV 方程式
<https://www.physics.okayama-u.ac.jp/~otsuki/lecture/CompPhys2/pde/kdv.html>
(2025/11/20)

8. 付録

8.1 KdV 方程式を解く Python コード

3 節で紹介した KdV 方程式を数値的に解く Python コードを以下に示す.

```
1 import os
2 import matplotlib.pyplot as plt
3 import numpy as np
4 from mpl_toolkits.mplot3d import Axes3D # noqa: F401 # needed for 3D
5
6 # るんげくったばらめた
7 def kdv_etdrk4(
8     Lx: float = 40.0,
9     N: int = 512,
10    dt: float = 0.005,
11    tmax: float = 12.0,
12    save_steps: int = 300,
13    c: float = 2.0,
14    x0: float = 5.0,
15 ):
16     # KdV方程式を解く!!
17     # Grid: 0 から始まる周期区間  $x$  in  $[0, Lx)$ 
18     x = np.linspace(0.0, Lx, N, endpoint=False)
19
20     # Wavenumbers for domain length Lx
21     k = 2 * np.pi * np.fft.fftfreq(N, d=Lx / N)
22     ik = 1j * k
23     L = 1j * k**3 # Linear operator in Fourier space:  $+ i k^3$ 
24
25     # Initial condition: 1-soliton for KdV (coefficient 6)
26     #  $u(x,0) = c/2 * \text{sech}^2(\text{sqrt}(c)/2 (x - x0))$ 
27     def sech(z):
28         return 1.0 / np.cosh(z)
29
30     u = 0.5 * c * sech(0.5 * np.sqrt(c) * (x - x0)) ** 2
31
32     # ETDRK4 precomputations (Kassam & Trefethen 2005)
33     E = np.exp(dt * L)
34     E2 = np.exp(dt * L / 2)
35     M = 16 # contour points for phi functions
36     r = np.exp(1j * np.pi * (np.arange(1, M + 1) - 0.5) / M)
37     LR = dt * L[:, None] + r[None, :]
38
39     Q = dt * np.mean((np.exp(LR / 2) - 1.0) / LR, axis=1)
40     f1 = dt * np.mean(
41         (-4 - LR + np.exp(LR) * (4 - 3 * LR + LR**2)) / LR**3,
42         axis=1,
43     )
44     f2 = dt * np.mean(
45         (2 + LR + np.exp(LR) * (-2 + LR)) / LR**3,
46         axis=1,
47     )
48     f3 = dt * np.mean(
```

```

49         (-4 - 3 * LR - LR**2 + np.exp(LR) * (4 - LR)) / LR**3,
50         axis=1,
51     )
52
53     # Nonlinear term N(u) = -6 u u_x (compute u_x via spectral derivative)
54     def nonlinear(u_phys: np.ndarray) -> np.ndarray:
55         U = np.fft.fft(u_phys)
56         ux = np.fft.ifft(ik * U).real
57         return -6.0 * u_phys * ux
58
59     # Time stepping & saving
60     Nt = int(np.round(tmax / dt))
61     if save_steps > Nt + 1:
62         save_steps = Nt + 1
63     save_every = max(1, Nt // (save_steps - 1))
64
65     t_save = []
66     U_save = []
67
68     def save(tn: float, u_phys: np.ndarray) -> None:
69         t_save.append(tn)
70         U_save.append(u_phys.copy())
71
72     t = 0.0
73     save(t, u)
74
75     U_hat = np.fft.fft(u)
76     for n in range(1, Nt + 1):
77         Nv = nonlinear(u)
78         a = np.fft.ifft(E2 * U_hat + Q * np.fft.fft(Nv)).real
79         Na = nonlinear(a)
80         b = np.fft.ifft(E2 * U_hat + Q * np.fft.fft(Na)).real
81         Nb = nonlinear(b)
82         c_stage = np.fft.ifft(
83             E * U_hat + Q * np.fft.fft(2 * Nb - Nv)
84         ).real
85         Nc = nonlinear(c_stage)
86
87         U_hat = (
88             E * U_hat
89             + np.fft.fft(Nv) * f1
90             + 2 * np.fft.fft(Na + Nb) * f2
91             + np.fft.fft(Nc) * f3
92         )
93         u = np.fft.ifft(U_hat).real
94         t = n * dt
95
96         if n % save_every == 0 or n == Nt:
97             save(t, u)
98
99     U_save = np.array(U_save)
100     t_save = np.array(t_save)
101

```

```

102     return x, t_save, U_save
103
104 # -----グラフ描画関数-----
105
106 def plot_surface(
107     x: np.ndarray,
108     t: np.ndarray,
109     U: np.ndarray,
110     out_path: str,
111     crest_line: bool = True,
112 ) -> None:
113     """3D サーフェスプロットを描画して保存する。"""
114     # Create mesh for surface: X: space, Y: time
115     X, Y = np.meshgrid(x, t)
116
117     fig = plt.figure(figsize=(6.3, 5.4))
118     ax = fig.add_subplot(111, projection="3d")
119
120     # 時間軸 (Y) を相対的に長く見せるためのアスペクト調整 (x:y:z)
121     try:
122         ax.set_box_aspect((1.0, 1.4, 0.9))
123     except Exception:
124         pass
125
126     surf = ax.plot_surface(
127         X,
128         Y,
129         U,
130         cmap="viridis",
131         linewidth=0,
132         antialiased=True,
133     )
134
135     # Z軸が見切れないように余白を持たせて設定
136     zmin = float(np.min(U))
137     zmax = float(np.max(U))
138     pad = 0.05 * (zmax - zmin + 1e-12)
139     ax.set_zlim(zmin - pad, zmax + pad)
140
141     # 範囲を明示
142     ax.set_xlim(float(x.min()), float(x.max()))
143     ax.set_ylim(float(t.min()), float(t.max()))
144
145     ax.set_xlabel("x")
146     # 標準の y ラベルは隠して独自配置 (余白削減)
147     ax.set_ylabel("")
148     ax.set_zlabel("u", labelpad=10)
149
150     # Z軸ラベルが視点で隠れないように画面基準で回転を固定
151     try:
152         ax.zaxis.set_rotate_label(False)
153         ax.zaxis.label.set_rotation(90)
154         ax.zaxis.label.set_color("black")

```

```

155     except Exception:
156         pass
157
158     fig.colorbar(surf, shrink=0.6, aspect=12, pad=0.1)
159
160     if crest_line:
161         idxs = np.argmax(U, axis=1)
162         x_crest = x[idxs]
163         z_crest = U[np.arange(len(t)), idxs]
164         ax.plot(
165             x_crest,
166             t,
167             z_crest,
168             color="red",
169             linewidth=2.0,
170             label="crest",
171         )
172         ax.legend(loc="upper right")
173
174     ax.view_init(elev=22, azim=-95)
175
176     fig.subplots_adjust(left=0.02, right=0.995, bottom=0.1, top=0.93)
177
178     try:
179         ax.zaxis.set_rotate_label(False)
180         ax.zaxis.label.set_rotation(90)
181     except Exception:
182         pass
183
184     # timeのラベルの位置!!
185     fig.text(
186         0.1,
187         0.2,
188         "time",
189         rotation=90,
190         va="center",
191         ha="center",
192         fontsize=10,
193     )
194
195     plt.savefig(out_path, dpi=400)
196     plt.close(fig)
197
198 def main() -> None:
199     x, t, U = kdv_etdrk4(
200         Lx=40.0,
201         N=512,
202         dt=0.005,
203         tmax=12.0,
204         save_steps=300,
205         c=2.0,
206         x0=5.0,
207     )

```

```
208     out_path = os.path.join(os.path.dirname(__file__), "KdV_surface.png")
209     plot_surface(x, t, U, out_path)
210     print(f"Saved 3D surface figure: {out_path}")
211
212
213 if __name__ == "__main__":
214     main()
```