

Pythonと実例で学ぶ 微分方程式

— はりの方程式から感染症の数理モデルまで —

博士(理学) 神永 正博 著

コロナ社

まえがき

本書は IT 活用型の微分方程式の教科書です。おもに、微分方程式を現実問題に応用したい学生・エンジニアの方々へ向けて書かれています。微分方程式の標準的な解法や、微分方程式が現実問題にどのように応用されるかを理解するとともに、Python を活用して現実問題を解けるようになることが目的です。

微分方程式は、自然科学、工学、医学などの共通言語です。多様な分野における現象を微分方程式の言葉によって記述し、現象の背後にあるメカニズムを解明し、未来を予測します。機械工学や土木工学を学ぶ方は、機械や建物の振動を解析するために微分方程式の応用としてモード解析を学ぶでしょうし、医学や疫学を学ぶ方は、感染症の拡大を予測するために微分方程式で記述された感染症の数理モデルを学ぶでしょう。本書では、こうした諸分野の入り口から少し内側にまで立ち入って、微分方程式がどのように活用されているのかまで説明します。さまざまな分野に興味を持ち、あちこち寄り道しながら微分方程式を学びたい方に向いています。

こうした多様な分野の入門的解説に加え、その多くに Python コードが付属しており、実際に動かして学ぶことができます (サンプルコードはコロナ社の書籍紹介ページ[†]からダウンロードできます)。Matplotlib ライブラリを使ったグラフの描画はもちろんですが、`scipy.optimize` を用いた実データへの当てはめ方を学んだり、気象予報士の試験問題や、電験三種 (第三種電気主任技術者試験) の問題を解いたりもします。その他、電気工学で使われるインピーダンスや、ローパスフィルタ回路がどのように雑音を除去するかなども学べます。微分方程式を数式処理ライブラリ SymPy で解く方法を知り、`scipy.odeint` ライブラリによる数値計算ソルバを使えるようになります。

興味がある方のために、数値解析の理論も紹介しました。例えば、`scipy.odeint` ライブラリは内部で数値計算アルゴリズムを切り替えています。なぜ切り替えなければならないのか、なぜ標準解法として知られるルンゲ・クッタ法だけでは済まないのかなどについて、疑問に思う方がいらっしゃるかもしれません。これらについても説明し、数値計算ソルバを自分で実装できるところまで案内いたします。

事前に必要なのは、大学初年次に学ぶ微分積分学と線形代数学の知識、それに Python の実行環境がインストールされている PC だけです。理解を深めるための章末問題も 100 題用意しました (解答は、コロナ社の書籍紹介ページからダウンロードできます)。楽しんでいただけたら幸いです。

[†] <https://www.coronasha.co.jp/np/isbn/9784339061239/>

本書執筆にあたり，感染症の数理モデルについては東京大学大学院数理科学研究科教授の稲葉寿先生，電気回路については東北学院大学工学部教授の吉川英機先生，Python プログラムについては同じく東北学院大学工学部講師の森島佑先生からご助言いただきました。記して感謝申し上げます。

2021 年 8 月

神永 正博

本文中に記載している会社名，製品名は，それぞれ各社の商標または登録商標です。本書では®やTMは省略しています。

目 次

1. 変数分離形の微分方程式

1.1 微分方程式とは何か	1
1.1.1 微分方程式を解くということ	1
1.1.2 常微分方程式と偏微分方程式	2
1.1.3 NumPy と Matplotlib の基本的な使い方	4
1.1.4 はりはどうたわむのか	10
1.2 変数分離形の方程式	15
1.2.1 放射性炭素年代測定	15
1.2.2 酵母菌の増殖, SciPy による実データへの当てはめ	18
1.2.3 電 気 伝 導	21
1.2.4 雨滴の落下速度	23
1.2.5 懸 垂 線	27
1.2.6 宇宙空間から月への自由落下	31
章 末 問 題	32

2. 変数分離形以外の 1 階微分方程式

2.1 同次系の方程式	36
2.2 1 階線形方程式	39
2.3 ベルヌーイの微分方程式	42
2.4 完全微分方程式, <code>contour</code> 関数による陰関数の表示	44
2.5 解の存在と一意性	47
2.5.1 変数分離形の解法で感じる違和感	47
2.5.2 解の存在と一意性の定理	49
2.5.3 逐次近似解の例	54
2.5.4 関数列の収束について	55
章 末 問 題	58

3. 定数係数線形方程式

3.1 典型的な運動方程式	60
---------------	----

3.2	斉次方程式を解く	62
3.3	特性方程式の解が複素数の場合	64
3.4	特性方程式が重解を持つ場合	69
3.5	非斉次方程式をどう解くか	71
3.5.1	非斉次項が指数関数の場合	71
3.5.2	$P(r) = 0$ となる場合	72
3.6	非斉次項が三角関数の場合	75
3.7	非斉次項が多項式の場合	78
3.7.1	振り子時計の原理	79
3.7.2	サスペンション	80
3.7.3	振動工学 (モード解析)	83
3.7.4	電気回路	90
3.7.5	インピーダンス	96
章 末 問 題		98

4. ラプラス変換, Pythonで厳密解・流れの可視化

4.1	ラプラス変換	100
4.2	SymPyでシンボリックに微分方程式を解く	105
4.3	連立微分方程式	107
4.3.1	1階連立微分方程式	107
4.3.2	streamplot関数による微分方程式の定める流れの可視化	108
4.3.3	微分方程式の定める流れの局所理論	110
4.3.4	行列の指数関数	116
章 末 問 題		123

5. Pythonで微分方程式を解く

5.1	微分方程式ソルバの使い方	125
5.1.1	極限周期軌道 (リミットサイクル)	126
5.1.2	トンネルダイオードとファン・デル・ポル方程式	128
5.1.3	ローレンツ方程式とカオス・数値計算の誤差	136
5.2	感染症の数理モデルを解く	143
5.2.1	SIRモデル	143
5.2.2	PythonでSIRモデルを解いてみよう	146
5.2.3	SEIRモデル	150
5.2.4	現実データへの当てはめ	153
章 末 問 題		156

6. Pythonで数値解析

6.1 基本的な数値計算アルゴリズム	159
6.1.1 オイラー法	159
6.1.2 素朴な数値解法がうまくいかない場合	164
6.1.3 ルンゲ・クッタ法	166
6.2 odeint ライブラリで使われている数値解法と硬い方程式	171
6.2.1 アダムス・バッシュフォース法の考え方	171
6.2.2 ルンゲ現象	173
6.2.3 アダムス・バッシュフォース・モルトン法	175
6.2.4 硬い方程式と数値的安定性	177
6.2.5 後退微分法	183
章末問題	186
引用・参考文献	187
索引	189

1

変数分離形の微分方程式

本章では、最初に微分方程式とは何なのか、微分方程式を解くとはどういうことかを説明し、積分するだけで解ける微分方程式と変数分離形の微分方程式について多くの具体例とともに説明していきます。1階の微分方程式はそれこそ無数にあるのですが、特に基本的で重要なものは変数分離形の方程式です。じつのところ、実用面で最も役立つタイプの微分方程式といっても過言ではありません。微分方程式の説明とともに、関数のグラフの描画や科学技術計算処理に利用されるライブラリ NumPy と Matplotlib についても説明します。

1.1 微分方程式とは何か

最初に準備運動を兼ねて、1階に限らず、微分方程式とは何かを簡単に説明してから、ただ積分すれば解が求まる微分方程式について説明します。

1.1.1 微分方程式を解くということ

定数 C_1 , C_2 を含む関数

$$x(t) = C_1 \cos t + C_2 \sin t \quad (1.1)$$

を考えます。つぎの x と、 x の微分を使って、定数を消去してみましょう。

$$x(t) = C_1 \cos t + C_2 \sin t$$

$$x'(t) = -C_1 \sin t + C_2 \cos t$$

$$x''(t) = -C_1 \cos t - C_2 \sin t$$

となりますので

$$x''(t) = -x(t) \quad (1.2)$$

とすることで定数を消去できます。式 (1.2) のような、関数とその微分からなる方程式を**微分方程式** (differential equation) といい、これを満たす式 (1.1) のような関数を求めることを式 (1.2) を解くといいます。式 (1.2) は2階の微分を含んでおり、2階よりも階数の高い微分は含まれていないので、2階の微分方程式といいます。一般にその微分方程式が含む微分の最大の

階数が k のとき、 k 階の微分方程式といいます。現象の解析に使われる微分方程式は、1 階か 2 階のものが多いのですが、3 階、4 階の微分方程式もあります。

後ほどさまざまな例を見ていきますが、自然現象は微分方程式の形で記述されることが多く、その振舞いを調べるために微分方程式を解くわけです。

定数を消去する方法は 1 通りとは限りません。 $x''''(t) = x(t)$ は、4 階の微分方程式ですが、明らかに式 (1.1) を含んでいますし、定数を 1 つだけ消去した関係式として

$$x^2 + (x')^2 = C \quad (1.3)$$

も微分方程式 ($C \geq 0$ は定数) です。これは 1 階の微分方程式です。

1.1.2 常微分方程式と偏微分方程式

いま述べた微分方程式は、正確には、**常微分方程式** (ordinary differential equation) と呼ばれます。これは変数が 1 つだけ、例えば時刻 t のみの微分方程式という意味です。変数が 2 つ以上の微分方程式を**偏微分方程式** (partial differential equation) といいます。本書では偏微分方程式を扱いませんが、常微分方程式との違いと両者の関係について簡単に触れておきます。

例えば

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \quad (1.4)$$

は、時刻 t 、(直線上の) 位置 x という 2 つの変数を持つ関数 $u = u(t, x)$ が満たす方程式で、偏微分方程式の 1 つです。この方程式は、**波動方程式** (wave equation) と呼ばれている重要な方程式で、速度 c ($\neq 0$) の波の運動を記述しています。常微分方程式と偏微分方程式を区別する理由の 1 つは、常微分方程式では定まらない定数 (式 (1.1) で出てきた C_1, C_2 のようなもの) の数が有限であるのに対し、偏微分方程式では、(標語的にいえば、ということですが) 解が無数の定数を含むということです。有限と無限という点で扱いが根本的に違ってくるのです。常微分方程式よりも偏微分方程式のほうが数学的には難しい対象ですが、偏微分方程式を解く際には常微分方程式が基礎となります。歴史的には、フーリエ (Jean Baptiste Joseph Fourier)[†] が、熱方程式 (拡散方程式ということもあります) の初期値・境界値問題

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} \\ u(0, x) &= f(x) \\ u(t, a) &= u(t, b) = 0 \end{aligned}$$

を解く際に、次節で扱う変数分離形の微分方程式と、3 章で扱う定数係数の線形微分方程式が

[†] ジャン・バティスト・ジョゼフ・フーリエは、18 世紀後半から 19 世紀前半にかけて活躍したフランスの数学者・物理学者です。熱伝導に関する研究から**熱方程式** (heat equation) を導出して、これを解くためにフーリエ級数の理論を展開しました。

現れました¹¹。これらの常微分方程式を解くことで、熱方程式を解くことができたのです。このように、解の構成を行う際に常微分方程式を解くことがありますが、特殊な形の解を求める際に常微分方程式を解く必要が出てくることがあります。例えば、光ファイバ内の光パルスの運動は、空間1次元の**非線形シュレディンガー方程式** (nonlinear Schrödinger equation)

$$i \frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} + |u|^{p-1} u = 0 \quad (1.5)$$

で記述されます。 i は虚数単位です。 $p > 1$ は定数ですが、応用上は、 $p = 3$ に取ることが多いようです。 u の正確な意味づけは難しいですが、だいたい光波の包絡線関数と考えればよいでしょう。工学的には、大容量の光通信を実現する際に出会う方程式です。この方程式の**定在波解** (standing wave solution) は、 $u(t, x) = e^{i\omega t} \varphi(x)$ という形の解ですが、これを式(1.5)に代入すると

$$i(i\omega)e^{i\omega t}\varphi + e^{i\omega t}\varphi'' + |\varphi|^{p-1}e^{i\omega t}\varphi = 0$$

となります。両辺を $e^{i\omega t}$ で割ると、つぎの常微分方程式が得られます。

$$\varphi'' - \omega\varphi + |\varphi|^{p-1}\varphi = 0 \quad (1.6)$$

詳しい話は多くの準備が必要なので立ち入りませんが、式(1.6)の解を調べることで、光ファイバ中の光波の動きがわかるのです。このように、偏微分方程式の特別な解を調べる際に常微分方程式が現れることがあります。

ここで、非線形という言葉が出てきました。波の重ね合わせができる(解の1次結合もまた同じ方程式の解になっている)方程式を線形方程式、できないものを非線形方程式といいます(線形方程式の正確な定義については、2.2節をご参照ください)。波動方程式と熱方程式は線形方程式で、非線形シュレディンガー方程式は名前のとおり非線形方程式です。

先ほど、常微分方程式は偏微分方程式より簡単だといいましたが、これは数学的な取扱いが入門段階で比較的簡単ということであって、最先端の研究において常微分方程式のほうが簡単というわけではありません。式(1.6)でもわからないことがいろいろとありますし¹²、昔から研究されている難問として、**三体問題** (the three-body problem) があります。三体問題とは、3つの天体の運動を記述する常微分方程式の性質を調べる問題ですが、その解の複雑さは驚くべきものです。ポアンカレ (Henri Poincaré) は、「三体問題がいかに複雑なものか、またこの

¹¹ 波動方程式との違いは、時間微分が1階になっただけですが、解の性質は随分違います。例えば、熱方程式では、時間がわずかでも経過すれば、初期値(ここでいえば $f(x)$) が滑らかでなくても解は非常に滑らかになる(平滑化効果がある)のに対し、波動方程式ではそうならず、解の特異性(滑らかでない部分)が伝播する(解の特異性の伝播)ことが知られています。

¹² 昔、大学時代の友人と2人で、反発的な非線形項を持つ非線形シュレディンガー方程式に、吸引的なデルタポテンシャルが入った場合の定在波解の安定性を巻末の文献1)のように調べたことがあります。論文を書くにあたりいろいろと調べて仰天したのは、このようにごく基本的な問題があまりわかっていないことでした。非線形の微分方程式は非常に難しいのです。

問題を解くためには、われわれの既知のすべての知識とは異なった超越的な知識が、いかに必要であるかもわかってくる]²⁾†とっています。

1992年、三体問題について、3つの星が8の字型の軌道を描く、いわゆる8の字解が存在することが証明されました³⁾。これは数値解が発見され、後に数学的な存在証明がなされたものですが、このような解があるなんてほとんど信じられないような話です。宇宙は広いので、どこかに本当に8の字軌道の天体があるのかもしれませんが。三体問題にはいまでも多くの謎が残されています。

本書では偏微分方程式は扱わないので、以下、特に断らない限り、微分方程式と書けば、常微分方程式を意味するものとします。

1.1.3 NumPy と Matplotlib の基本的な使い方

本書では、特に NumPy と Matplotlib というライブラリを頻繁に使用しますので、ここで、必要な知識をまとめておきます。ご存じの方は読み飛ばしていただいて結構です。

以降、Python 3.6 (以降) と NumPy, Matplotlib, SciPy, SymPy というライブラリがインストールされていることを前提に説明しています (互換性のない Python 2.x については対応していませんのでご注意ください)。Python のインストールについては、いくつかの方法がありますが、ウェブ上に豊富な情報がありますので、そちらを見ていただいたほうがよいでしょう。本書では、Anaconda をインストールして標準で使える Spyder を使っています。Anaconda をインストールした場合は、すでに上記の4つのライブラリがインストールされているはずです。

SciPy は、NumPy ベースの科学技術計算ライブラリで、使いやすいインタフェースにラッピングされています。SymPy は数式処理ライブラリで、代数計算、微分・積分、ラプラス変換などを行うことができます。SymPy と SciPy の使い方については、適宜説明することにして、ここではより頻繁に利用する NumPy と Matplotlib について説明します。

[1] **NumPy の n 次元配列 (ndarray)** Python で配列を扱う場合、リストにする方法と ndarray にする方法の2つがあります。リスト化は手軽な方法ですが、処理が遅く、高速な処理が必要な科学技術計算には向いていません。Python で高速な計算を行う場合には、ndarray というデータ型にすることで、NumPy という高速なライブラリが使えるようになります。以下、単に配列といえば ndarray のことを指します。ndarray は、 n -dimensional array (n 次元配列) という意味です。1次元の配列はリストと似ていますが、リストでは異なる型が混ざっていてもよいのに対し、ndarray ではそれが許されず、すべての要素が同じ型である必要があります。IPython で少し様子を見てみましょう。IPython は、対話型の Python インタフェースで、プログラムを組むほどでないちょっとしたことを試すのに便利です。例えば、 $\cos \frac{\pi}{2}$, $\sin \frac{\pi}{3}$, $\sin \frac{\pi}{4}$ を一度に計算させることを考えます。NumPy を使う場合は、このようにします。

† 肩つき数字は巻末の引用・参考文献を示します。

```
In [1]: import numpy as np
In [2]: PI = np.pi
In [3]: x = np.array([PI/2,PI/3,PI/4])
In [4]: np.cos(x)
Out[4]: array([6.12323400e-17, 5.00000000e-01, 7.07106781e-01])
```

1行目でNumpyをnpという名前をつけてインポートしています。これは広く使われている略記法ですので、本書でもこれになります。2行目でNumPy用の π にPIという名前をつけました。3行目では、 $(\pi/2, \pi/3, \pi/4)$ という1次元配列(ndarray型のデータ)をつくっています。4行目では、この $x=(x[0], x[1], x[2])$ という配列に対して、 $[\cos(x[0]), \cos(x[1]), \cos(x[2])]$ の値を計算しています。配列の要素ごとに正弦の値を求めて並べたものが出力されていることがわかるでしょう。 $\cos \frac{\pi}{2}$ は0にならなければなりません、ここでは非常に小さい値ではあるものの0にはなっていません。これは数値計算の誤差によります。続けて、xの要素をまとめて2倍してみましょう。つぎのように書くだけです。ベクトルのスカラー倍と同じです。

```
In [5]: 2*x
Out[5]: array([3.14159265, 2.0943951, 1.57079633])
```

同じ型の配列なら、ベクトルのように足すこともできます。

```
In [6]: y = np.array([1, 2, 3])
In [7]: z = np.array([3, 8, 9])
In [8]: y+z
Out[8]: array([ 4, 10, 12])
```

掛け算やべき乗の計算もできます。例えば、 t をndarray型にして $t^2 + 2t + 3$ とするとつぎのようになります。Pythonでは、 a^b は、 $a**b$ で表します。

```
In [9]: t = np.array([1, 2, 3, 4])
In [10]: t**2+2*t+3
Out[10]: array([ 6, 11, 18, 27])
```

このように成分ごとに計算してくれるわけです。

ndarrayでは複素数を扱うこともできます。微分方程式を解くために必要となる特性方程式という代数方程式がありますが、その解が複素数のとき、オイラーの公式($e^{i\theta} = \cos\theta + i\sin\theta$)を經由して複素数成分を持つ配列を扱う場合がありますので、ここで簡単に説明しておきましょう。Pythonでは虚数単位をjで表現します。これは電気工学などで一般に使われる記法です。電気工学では、電流を*i*で表現するため、*i*を虚数単位としては使わずに*j*を使うことになったのです。本書では、数学の慣習に従って虚数単位を*i*で表しますが、Pythonでの表現はjになるので混同しないようご注意ください。

例えば、 $z = (-1 + 5i, 1.4 + 7.6i, -1.7 + 0.5i)$ というベクトル(配列)はつぎのように表現します[†]。

[†] dtypeを参照してz.dtypeとするとその型(プラットフォームに依存)が表示されます。筆者の環境では、dtype('complex128')と出ます。

索引

【あ】		共振周波数	96	終端速度	23
アダムス・バッシュフォース法	171	競争排除則	18	状態ベースインタフェース	7
アダムス・バッシュフォース・モールトン法	171, 176	共鳴	76	常微分方程式	2
アダムス・モールトン法	171, 175	行列の指数関数	116	初期条件	15
アノード	129	極限周期軌道	126, 128	初期値に対する鋭敏性	140
アール・ゼロ	145	局所線形化	111	初期値問題	15
アール・ノート	145	局所離散化誤差	166	自励系	107
安定	111	【く】		振動応答	89
安定領域	178	空乏層	129	振動工学	83
【い】		グロンウォールの不等式	53	振幅応答	88
一樣収束	55	【け】		振幅倍率	81
陰的公式	175	減衰行列	85	【す】	
インピーダンス	97	懸垂線	27	ストークスの法則	24
【え】		【こ】		ストークス方程式	40
エアリー方程式	40	剛性行列	85	ストレンジアトラクタ	142
江崎ダイオード	131	合成積	103	【せ】	
【お】		後退オイラー法	181	制御理論	98
オイラー法	160	後退微分法	171, 183	正孔	128
【か】		降伏	129	斉次方程式	61
回復率	144	降伏電圧	129	整流作用	128
改良オイラー法	170	固定点	110	積分ステップ	138
カオス	140	固有振動数	83	絶対許容誤差	138
各点収束	55	固有振動モード	89	漸近安定	111
隔離者	143	【さ】		線形化行列	111
加速器質量分析法	17	最終規模方程式	150	線形化方程式	111
カソード	129	最小二乗法	19	線形性	39
硬い方程式	171, 177	サスペンション	80	せん断形構造物	84
過渡現象	90	残差平方和	19	【そ】	
感受性保持者	143	三体問題	3	相図	110
感染者	143	サンプリングステップ	138	相対許容誤差	138
完全微分方程式	44	【し】		阻止帯域	92
緩和時間	21	シグモイド	18	【た】	
【き】		シグモイド曲線	18	大域漸近安定	141
構造的なモデル	156	自己インダクタンス	94	大域離散化誤差	166
刻み幅	160	実ジョルダン標準形	121	ダイオード	128
基本再生産数	145, 152	質量行列	85	第1種エアリー関数	40
境界値問題	12	時定数	91	畳込み	103
共振	76	遮断周波数	92	多段法	160
		自由振動特性	85	単純固定ばり	13
		修正子	176	単振動の方程式	60
		収束座標	100	弾性2次モーメント	11

	【ち】	非線形シュレディンガー方程式	3	有効接触率	144
		微分演算子	61	【よ】	
地 動	84	微分多項式	61	陽的公式	175
中点公式	164	微分方程式	1	予測子	176
調和振動子の方程式	60			予測子・修正子法	176
	【つ】	【ふ】		【ら】	
		ファン・デル・ポル振動子	133	ラグランジュの補間多項式	172
通過帯域	92	ファン・デル・ポル方程式	132	ラプラス逆変換	101
	【て】	負性抵抗	130	ラプラス変換	100
定義域	61	フックの法則	60		
定在波解	3	不動点	110	【り】	
定常状態	81	プラントル数	136	リアプノフ安定	111
定常振動	81	分離線	114	リエナール方程式	135
適応刻み幅制御	139			リカッチの微分方程式	43
電気伝導度	22	【へ】		力学系	108
伝達関数	98	平均自由時間	22	力学系理論	135
	【と】	平均的な接触率	158	リプシッツ条件	50
特殊関数	41	平衡点	110	流動速度	22
特性方程式	62	ベッセル方程式	59	両端固定ばり	12
ドルーデの公式	22	ベルヌーイの微分方程式	42	臨界減衰	70
トンネル効果	130	偏微分方程式	2	臨界免疫化割合	146
トンネルダイオード	130	【ほ】		【る】	
	【な】	ポアンカレ・ベンディクソンの		ルンゲ・クッタ型の公式	167
		定理	135	ルンゲ・クッタ法	167
流 れ	108	ホイン法	170	ルンゲ現象	174
	【ね】	放射性炭素年代測定	17	【れ】	
		補 外	172	レイノルズ数	24
熱電圧	129	ホール	128	レイリー数	136
熱方程式	2			レスラー方程式	157
	【の】	【ま】		【ろ】	
ノルム	116	曲げ剛性	11	ロジスティック曲線	18
	【は】	丸め誤差	164	ロジスティック方程式	18
爆 発	57	免疫保持者	143	ローパスフィルタ	92
爆発時刻	57	【も】		ローレンツアトラクタ	142
波動方程式	2	モード解析	89	ローレンツ方程式	136
ばね・マス・ダンパモデル	80	【や】		【わ】	
は り	10	ヤコビ行列	111	ワイエルシュトラスの優級数	
半減期	16	ヤング率	11	判定法	55, 56
	【ひ】	【ゆ】			
ピカールの逐次近似法	50	有限要素法	89		
非斉次方程式	61				

【A】

A 安定 181

【B】

BDF 171

【L】LSODA 171
LSODE 171

	【N】			【S】			
n 形半導体		128	SEIR モデル		150	1 階線形方程式	39
N 段法		160	SIR モデル		143	1 段法	172
						2 段法	173
				【その他】			
p 形半導体		128	ω 極限集合		142		

—— 著者略歴 ——

1991年 東京理科大学理学部数学科卒業
1993年 京都大学大学院理学研究科修士課程修了（数学専攻）
1994年 京都大学大学院理学研究科博士課程中退（数学専攻）
1994年 東京電機大学助手
1998年 株式会社日立製作所勤務
2003年 博士（理学）（大阪大学）
2004年 東北学院大学講師
2005年 東北学院大学助教授
2007年 東北学院大学准教授
2011年 東北学院大学教授
現在に至る

Python と実例で学ぶ微分方程式

— はりの方程式から感染症の数理モデルまで —

Ordinary Differential Equation with Python

— Learning via Many Real-World Examples, including Beam Equations,

Mathematical Models of Infectious Diseases — © Masahiro Kaminaga 2021

2021年10月22日 初版第1刷発行

★

検印省略

著者 神 永 正 博
発行者 株式会社 コロナ社
代表者 牛来真也
印刷所 三美印刷株式会社
製本所 有限会社 愛千製本所

112-0011 東京都文京区千石 4-46-10

発行所 株式会社 コロナ社
CORONA PUBLISHING CO., LTD.

Tokyo Japan

振替 00140-8-14844 · 電話 (03) 3941-3131(代)

ホームページ <https://www.coronasha.co.jp>

ISBN 978-4-339-06123-9 C3041 Printed in Japan

(齋藤)



JCOPY <出版者著作権管理機構 委託出版物>

本書の無断複製は著作権法上での例外を除き禁じられています。複製される場合は、そのつど事前に、出版者著作権管理機構（電話 03-5244-5088, FAX 03-5244-5089, e-mail: info@jcopy.or.jp）の許諾を得てください。

本書のコピー、スキャン、デジタル化等の無断複製・転載は著作権法上での例外を除き禁じられています。購入者以外の第三者による本書の電子データ化及び電子書籍化は、いかなる場合も認めていません。落丁・乱丁はお取替えいたします。