8　微分方程式の解法

物理の法則の多くは微分方程式の形で記述されている。例えば、Newton 方程式、Maxwell 方程式、Schrödinger 方程式等すべて微分方程式である。従って、それらを実際にどのように解くかは極めて重要な問題である。

物理学において、数値として解が得られるという事は本質的意味を持つ。例えば、Newton 力学により惑星の運動が理解されたといっても、定性的な理解のみでは不十分である。実際に Newton 方程式を解いて惑星の位置を計算し、それを観測と比較する事によってのみ、Newton 力学が正しいか否かが確かめられるのである。水星の運動については、どうしても説明の出来ない部分が残り、それが一般相対性理論により説明された事実は、いかに定量的に現象を理解する事が重要かを如実に示している。

8.1　例題

図 8.1 で表される簡単な RC 回路を考える。はじめコンデンサ C に電荷はたくわえられていないとする。時刻 $x = 0$ でスイッチ S を入れる時、時刻 x でのコンデンサ C の両端の電圧 y を求めよ。

8.1.1　微分方程式

回路を流れる電流は $C(dy/dx)$ で与えられるので、キルヒホッフの法則により y の時間変化を表す微分方程式は

$$\begin{aligned} CR\frac{dy}{dx} &= E - y \\ y(0) &= 0 \end{aligned} \tag{8.1}$$

と表される。ここで $C = 1\,\mu\mathrm{F}$、$R = 1\,\mathrm{k}\Omega$、$E = 1\,\mathrm{V}$ とすると

$$\begin{aligned} \frac{dy}{dx} &= 1 - y \\ y(0) &= 0 \end{aligned} \tag{8.2}$$

となる。ただし、時定数 $\tau = RC = 1\,\mathrm{ms}$ であるので、時刻 x は ms で表す (電圧 y は V で表す)。

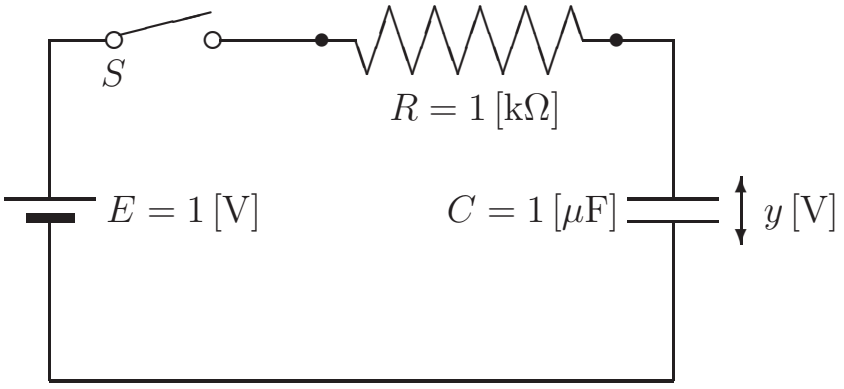

図 8.1: RC 回路の例。スイッチ S を閉じた後、電圧 y はどう変化するか?

8.1.2 メインプログラム

数値計算では変数を離散的に扱う為、時刻 x の区間 $[a,b]$ を適当なステップ h によって分割し、$x_n = a + nh(n = 0, 1, 2, \cdots)$ における解 y_n を順次求めれば良い。

メインプログラム rc_circuit は例えば以下のように書ける。

```
      program rc_circuit
      implicit none
c constants:
      real TIME_INIT,TIME_LAST            ! 計算する時間範囲
      parameter(TIME_INIT=0.0,TIME_LAST=4.0)
      real Y_INIT                         ! 電圧の初期値 (V)
      parameter(Y_INIT=0.0)
      real STEP                           ! 時間の刻み (ms)
      parameter(STEP=0.25)
c local:
      real x,y,y_next                     ! 時刻、電圧
c begin:
      y    = Y_INIT                       ! 電圧を初期化
      x    = TIME_INIT                    ! 時間を初期化
      write(*,'(a,F5.2,a,F8.5)')          ! 電圧の初期値を出力
     &     'Time: ',x,' Voltage: ',y
      do while (x.lt.TIME_LAST)
c 時間 x での電圧を y とし、時間が STEP だけ経過した時の
c 電圧 y_next を微分方程式を解いて求める
         call calc_next_step(x,y,STEP,y_next)
         x    = x + STEP                 ! STEP 後の時間
         y    = y_next                   ! STEP 後の電圧
         write(*,'(a,F5.2,a,F8.5)')      ! STEP 後の時間と
     &     'Time: ',x,' Voltage: ',y      ! 電圧を出力
      end do
      stop
      end
```

プログラムでは、時間 x 及び電圧 y を表す変数 x,y に初期値 (TIME_INIT,Y_INIT) を代入した後、x があらかじめ定めた時間 TIME_LAST に達するまで calc_next_step を呼び、ある時刻 x_n における (x_n, y_n) を用いて次のステップ (x_{n+1}, y_{n+1}) の値を計算する。微分方程式を実際に解く作業は、サブルーチン calc_next_step に下請けさせる。

微分方程式の記述は、以下のように diff_eq に押し込める。

```
      real function diff_eq(x,y)
      implicit none
c constant:
      real MAX_VOLT                       ! 最大電圧（電池電圧）
```

```
      parameter(MAX_VOLT=1.0)
c inputs:
      real x,y                              ! 時刻 x, 電圧 y
c begin:
      diff_eq = MAX_VOLT - y                ! dy/dx
      return
      end
```

8.2 微分方程式の解法

8.2.1 Euler 法

微分の定義に立ち戻って考えると、

$$\left.\frac{dy}{dx}\right|_{x=x_n} = \lim_{h\to 0}\frac{y(x_n+h)-y(x_n)}{h}(\equiv f(x_n, y(x_n))) \tag{8.3}$$

であるので、一番簡単な微分方程式の数値解法は、

$$\begin{aligned} y(x_1) &= y(x_0) + hf(x_0, y(x_0)) \\ y(x_2) &= y(x_1) + hf(x_1, y(x_1)) \\ &\vdots \\ y(x_m) &= y(x_{m-1}) + hf(x_{m-1}, y(x_{m-1})) \end{aligned} \tag{8.4}$$

と、前の点における傾き $f(x_{m-1}, y(x_{m-1}))$ をたどっていく方法である。これを**Euler** 法という。

この解法では、微分方程式を x_i から x_{i+1} まで積分する時、右辺の $f(x, y)$ がその区間で一定であると近似しており、数値積分法では**区分求積法**に対応している。

サブルーチンの例

サブルーチン calc_next_step は Euler 法では以下のように書くことができる。

```
      subroutine calc_next_step(x,y,step,y_next) ! Euler 法による解法
      implicit none
c inputs:
      real x,y,step                         ! 時刻 x, 電圧 y, 刻み幅 step
c output:
      real y_next                           ! 刻み幅後の電圧
c function:
      real diff_eq                          ! dy/dx
c begin:
      y_next = y + step * diff_eq(x,y)      ! Euler 法
      return
      end
```

Euler 法の誤差

微分方程式 $dy/dx = f(x, y)$ の解が x の解析関数 $y(x)$ であるとして、区分点 x_i の近傍で $y(x)$ を Taylor 展開すると

$$y(x_i + h) = y(x_i) + h \left.\frac{dy}{dx}\right|_{x=x_i} + \frac{1}{2}h^2 \left.\frac{d^2y}{dx^2}\right|_{x=x_i} + \mathcal{O}(h^3) \tag{8.5}$$

と書ける。最後の $\mathcal{O}(h^3)$ は h^3 以上の高い次数の項を表す。Euler 法は h^2 以上の項を無視している事に相当するので、誤差の程度は大雑把にいって h^2 程度である。しかし解法を見れば分かるように、微分方程式を解くという事は変化を追いかける作業を多数回行う事に相当する。回数は $(b-a)/h$ で、大雑把には h^{-1} である。従って多数回繰り返すと、誤差は積もり積もって容易に h の 1 次自体に近付いてしまう。

以上から、Euler 法では刻み h を細かくして精度を上げようとしても、必然的に増加する繰り返し数のため、ある限界以上の精度は不可能である事が分かる。

練習

上の例題のサンプルプログラムは、

```
/home/teacher/z6wt01in/SAMPLE/
```

に、

```
rc_circuit.f diff_eq.f euler.f
```

として置いてある。各自コピーして、実行ファイル `rc_circuit.f` を

```
frt -o rc_circuit rc_circuit.f diff_eq.f euler.f
```

により生成し、実行せよ (コンパイル時に、「"diff_eq.f", line 1: この仮引数'x' は, 副プログラム中で使用されていません.」といったメッセージがでるが、実行には影響は無い)。

刻み幅 `STEP` を色々変えて実行し、厳密解 $y(x) = 1 - e^{-x}$ と比較せよ。ある程度刻み幅を小さくすると計算機内部の丸め誤差が無視できなくなる事を確かめよ。

8.3 Runge-Kutta 法

微分方程式の解法として勧められるのが Runge-Kutta 法[1]である。この方法は以下に示すように、プログラムが容易であるにも関わらず誤差の積み重なりを抑える効果が絶大である。

基本的な方針は図 8.2 に示すように、変域を分け、途中で勾配を修正しながら進むというものである。ここで修正の方針は、そのまま先に進んだ場合に得られる勾配を、先回りして使うというものである。微分方程式、

$$\frac{dy}{dx} = f(x, y) \tag{8.6}$$

を例にとり具体的に説明する。

Runge-Kutta 法では、x の変域 h を 4 つの領域、

$$\frac{h}{6}, \frac{h}{3}, \frac{h}{3}, \frac{h}{6} \tag{8.7}$$

に分け、ステップ 1–4 で各変域を進んで行く。各ステップでの処理は以下の通り。

[1] 1895 年に Runge が発表したものを、Kutta が 1901 年に改良したもの。

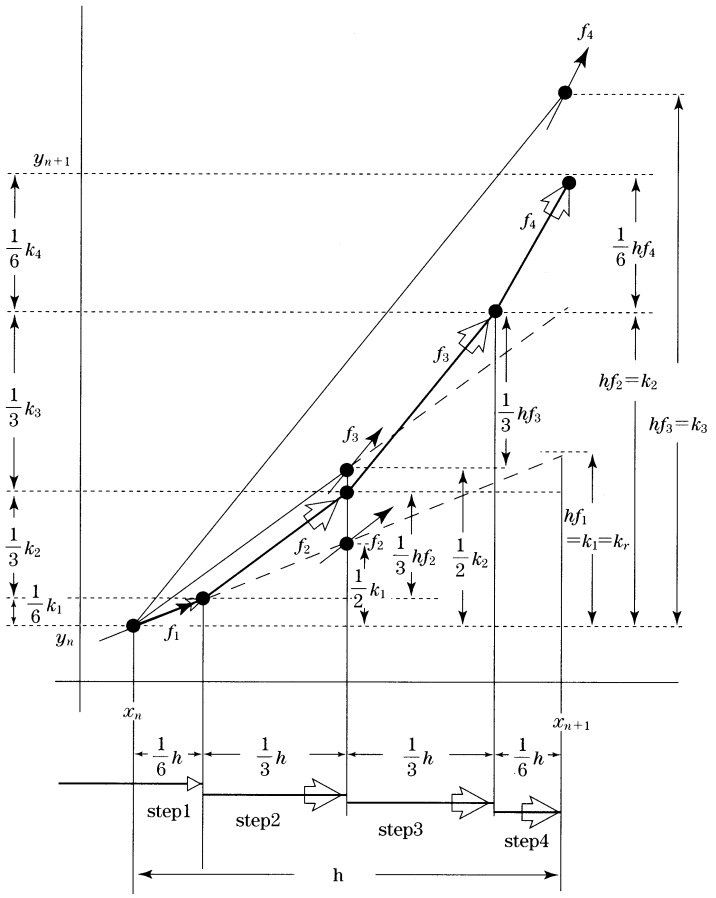

図 8.2: Runge-Kutta 法の概念図。計算は白抜きの矢印に従って進んでいく。ここで $f_1 \sim f_4$ は矢印の勾配を示す。「「計算物理 I」夏目雄平・小川建吾 著 (朝倉書店) の図 7.3 より転載。

ステップ 1 Euler 法と同じく、(x_n, y_n) での勾配で $h/6$ 進む。この勾配を f_1 とし、この勾配で h 進んだ場合の y の変化を $k_1(\equiv hf_1)$ とおく。

ステップ 2 先を予想し、ステップ 1 での勾配でそのまま更に $h/3$(全体では $h/2$) 進んだ地点の勾配

$$f\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) = f_2 \tag{8.8}$$

を用いて、実際に $h/3$(全体では $h/2$) 進む。

ステップ 3 始めの点 (x_n, y_n) から勾配 f_2 でステップ 1 と 2 を実行した ($h/2$ 進んだ) 場合の点 $(x_n + h/2, y_n + k_2/2)$ での勾配 $(k_2 \equiv hf_2)$

$$f\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) = f\left(x_n + \frac{h}{2}, y_n + \frac{hf_2}{2}\right) = f_3 \tag{8.9}$$

を用いて、$h/3$ 進む。

ステップ 4 勾配 f_3 で (x_n, y_n) から (x_{n+1}, y_{n+1}) へ進んだ場合の (x_{n+1}, y_{n+1}) での勾配

$$f(x_n + h, y_n + k_3) = f(x_n + h, y_n + hf_3) = f_4 \tag{8.10}$$

で最後の $h/6$ 進む。ここで $k_3 \equiv hf_3$ である。

以上説明は繁雑だが、式で整理すると、

$$y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 \tag{8.11}$$

である。ここで $k_4 \equiv hf_4$ である。$k_i = hf_i$ $(i = 1 - 4)$ を用いると

$$y_{n+1} = y_n + \left\{\frac{1}{6}f_1 + \frac{1}{3}f_2 + \frac{1}{3}f_3 + \frac{1}{6}f_4\right\} h \tag{8.12}$$

と書ける。

式 (8.12) は、関数 f の Tayler 展開の 4 次までを正しく扱っており、誤差は大雑把にいって h^5 程度である。式 (8.12) による解法を**4 次の Runge-Kutta 法**という。単に Runge-Kutta 法という場合には、この 4 次のものを指す。

サブルーチンの例

サブルーチン `calc_next_step` は Runge-Kutta 法では以下のように書くことができる。

```
      subroutine calc_next_step(x,y,step,y_next) ! Runge-Kutta 法
      implicit none
c inputs:
      real x,y,step                          ! 時間 x, 電圧 y, 刻み幅 step
c output:
      real y_next                            ! 刻み幅後の電圧
c function:
      real diff_eq                           ! dy/dx
c local:
      real k1,k2,k3,k4
c begin:
      k1 = step * diff_eq(x           , y         )
      k2 = step * diff_eq(x + 0.5*step, y + 0.5*k1)
      k3 = step * diff_eq(x + 0.5*step, y + 0.5*k2)
      k4 = step * diff_eq(x +     step, y +     k3)
      y_next = y + (k1 +2.0*k2 +2.0*k3 +k4)/6.0
      return
      end
```

Runge-Kutta 法の誤差

練習

式 (8.12) が、h の 4 次の項まで正しく扱っている事を確かめよ。

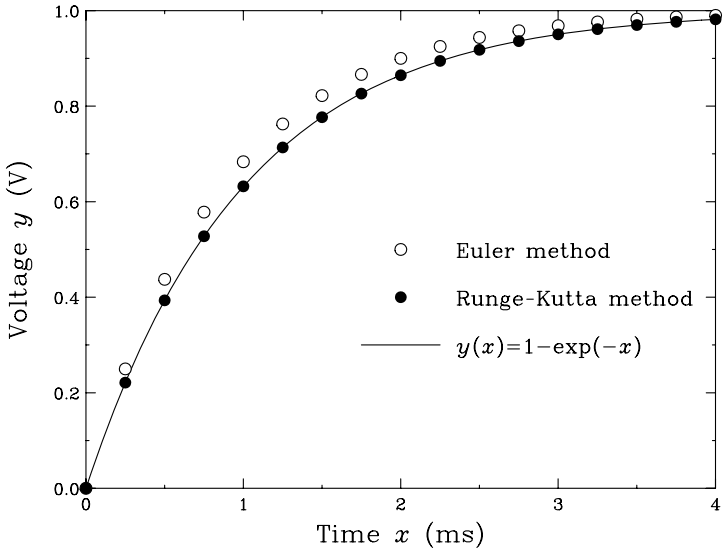

図 8.3: 解析解 (実線) と、Euler 法及び Runge-Kutta 法を用いた数値解との比較。

8.4 厳密解との比較

微分方程式 (8.2) の厳密解は

$$y(t) = 1 - e^{-x} \tag{8.13}$$

で与えられる。

ステップを `step=0.25` にした時の Euler 法と Runge-Kutta 法による数値解と解析解を比較したのが図 8.3 である。Runge-Kutta 法が Euler 法と比べて解析解をよく再現することが分かる。

練習

Runge-Kutta 法のサンプルプログラムは、

```
/home/teacher/z6wt01in/SAMPLE/runge_kutta.f
```

として置いてある。各自コピーして実行せよ (コンパイル時に、「"diff_eq.f", line 1: この仮引数'x' は, 副プログラム中で使用されていません.」といったメッセージがでるが、実行には影響は無い)。

また、刻み幅 `STEP` を色々変えて実行し、Euler 法を用いた場合の結果及び厳密解 $y(x) = 1 - e^{-x}$ と比較せよ。

8.5 2 階常微分方程式の解法

物理学で多く使われる 2 階常微分方程式も、以下のように Runge-Kutta 法により解くことが出来る。

2 階常微分方程式は一般に、

$$\frac{d^2y}{dx^2} = g\left(x, y, \frac{dy}{dx}\right) \tag{8.14}$$

と書くことが出来る。g は 3 つの引数で決まる関数である。ここで、dy/dx を新たな変数 z とおくことにより、

$$\begin{aligned}\frac{dy}{dx} &= z \\ \frac{dz}{dx} &= g(x, y, z)\end{aligned} \tag{8.15}$$

と、連立の 1 階常微分方程式とみなせる。これは一般的に

$$\begin{aligned}\frac{dy}{dx} &= g_1(x, y, z) \\ \frac{dz}{dx} &= g_2(x, y, z)\end{aligned} \tag{8.16}$$

と表す事ができる。Runge-Kutta 法を y と z の両方の変化に対して適用すると、

$$\begin{aligned}k_1 &= hg_1(x_n, y_n, z_n) \\ q_1 &= hg_2(x_n, y_n, z_n) \\ k_2 &= hg_1\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}, z_n + \frac{q_1}{2}\right) \\ q_2 &= hg_2\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}, z_n + \frac{q_1}{2}\right) \\ k_3 &= hg_1\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}, z_n + \frac{q_2}{2}\right) \\ q_3 &= hg_2\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}, z_n + \frac{q_2}{2}\right) \\ k_4 &= hg_1(x_n + h, y_n + k_3, z_n + q_3) \\ q_4 &= hg_2(x_n + h, y_n + k_3, z_n + q_3)\end{aligned} \tag{8.17}$$

となり、最終的に次式を得る。

$$\begin{aligned}y(x_n + h) &= y(x_n) + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 \\ z(x_n + h) &= z(x_n) + \frac{1}{6}q_1 + \frac{1}{3}q_2 + \frac{1}{3}q_3 + \frac{1}{6}q_4\end{aligned} \tag{8.18}$$

これにより、2 階常微分方程式が Runge-Kutta 法により解ける事が分かる。

課題：RLC 直列回路のシミュレーション

図 8.1 にコイルが直列に入った RLC 直列回路を考える。コンデンサ両端の電圧 y は 2 階常微分方程式

$$LC\frac{d^2y}{dx^2} + RC\frac{dy}{dx} + y = 0 \tag{8.19}$$

を満たす。この方程式を、適当な条件下で数値的に解け。y の時間依存性をグラフに示せ。