情報科学概論 11 回目

今回の目的

- 数値計算法 –常微分方程式の解法–

1 常微分方程式

- 常微分方程式の問題は、いつも 1 階の微分方程式の組みに置き換えて考えることができる。
(例) $\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} + r(x) = 0$ は以下の連立 1 階微分方程式に書き直せる。

$$\begin{aligned} \frac{dy}{dx} &= z(x) \\ \frac{dz}{dx} &= -r(x) - q(x)z(x) \end{aligned}$$

- したがって、連立 1 階微分方程式を数値的に解ければ、一般的な常微分方程式を解けることになる

1.1 連立 1 階常微分方程式の解法

連立 1 階常微分方程式

$$\frac{d\boldsymbol{y}(x)}{dx} = \boldsymbol{f}(x, \boldsymbol{y}) \tag{1}$$

を考える。ここで、$\boldsymbol{y}(x) = \{y_0(x), y_1(x), \cdots\}$ や $\boldsymbol{f}(x, \boldsymbol{y}) = \{f_0(x, \boldsymbol{y}), f_1(x, \boldsymbol{y}), \cdots\}$ は多成分から成るベクトルである (ここでの添え字はベクトルの成分を表わしており、以下の座標の離散化とは区別する)。

この解を数値的に求める際に、まず、座標 x について刻み幅 h で、$x_0, x_1 = x_0+h, x_2 = x_0+2h, \cdots, x_n = x_0+nh$ と離散化する。もしある位置 x_n での値 $\boldsymbol{y}(x_n) \equiv \boldsymbol{y}_n$ が既知の場合には、式 (1) の微分を刻み幅 h での差分で近似してやることで、隣の点 $x_{n+1} = x_n + h$ での値 $\boldsymbol{y}(x_{n+1}) \equiv \boldsymbol{y}_{n+1}$ を近似的に求めることができる。この差分化の方法として以下のものが代表的である。

- (1 次精度) オイラー (Euler) 法

$$\begin{aligned} \boldsymbol{k} &= \boldsymbol{f}(x_n, \boldsymbol{y}_n), \\ \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{k} \end{aligned} \tag{2}$$

  - 刻み幅 h で離散化された座標 $x_n = x_0 + nh$ に対し、上記を $(x_0, \boldsymbol{y}_0)$ から逐次的に繰り返すことによって解 $\boldsymbol{y}_n$ を数値的に求める。
  - オイラー法は、テイラー展開の 1 次までを残した近似となっている。そのため、時間積分を逐次的に繰り返していくと、$\mathcal{O}(h^2)$ の誤差が累積していく。
  - オイラー法で（物理現象の時間スケールで決まる）単位時刻まで時間積分した場合の打ち切り誤差は $\mathcal{O}(h)$ である。単位時刻までを N 等分した小さな刻み幅 $h \sim 1/N$ を用いて、1 ステップ更新する度に h^2 程度の誤差が生じるので、N ステップ分だけ誤差が累積したとすると、全体で $h^2N \sim h$ 程度の誤差が生じる。

- 2 次精度ルンゲクッタ (Runge Kutta) 法 (中点法)

$$\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{f}(x_n, \boldsymbol{y}_n), \\ \boldsymbol{k}_2 &= \boldsymbol{f}\left(x_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_1\right), \\ \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h\boldsymbol{k}_2 \end{aligned} \tag{3}$$

  - 出発点での微分 $\boldsymbol{k}_1$ を用いて、区間の中点位置での微分近似値 $\boldsymbol{k}_2$ を求め、それを用いて値を更新する。
  - 誤差評価を行うと、テイラー展開の 2 次までを残した近似となっている。そのため、1 ステップ当たり $\mathcal{O}(h^3)$ の誤差が累積し、単位時間当たり $\mathcal{O}(h^2)$ の打ち切り誤差を生じる。

- 4 次精度ルンゲクッタ法

$$
\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{f}(x_n, \boldsymbol{y}_n), \\
\boldsymbol{k}_2 &= \boldsymbol{f}\left(x_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_1\right), \\
\boldsymbol{k}_3 &= \boldsymbol{f}\left(x_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_2\right), \\
\boldsymbol{k}_4 &= \boldsymbol{f}\left(x_n + h, \boldsymbol{y}_n + h\boldsymbol{k}_3\right), \\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + \frac{h}{6}\left(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4\right)
\end{aligned} \tag{4}
$$

  - 誤差評価を行うと、テイラー展開の 4 次までを残した近似となっている。そのため、1 ステップ当たり $\mathcal{O}(h^5)$ の誤差が累積し、単位時間当たり $\mathcal{O}(h^4)$ の打ち切り誤差を生じる。

1.2 2 次精度ルンゲクッタ法の導出

オイラー法が 1 次近似となっていることは自明であろうから、2 次精度ルンゲクッタ法の導出を行い、2 次近似となっていることを確認しよう。簡単のため 1 変数 $\boldsymbol{y} = y$ とする。まず、以下のような形の差分方程式系を考える。

$$
\begin{aligned}
k_1 &= f(x_n, y_n), \\
k_2 &= f\left(x_n + ch, y_n + ahk_1\right), \\
y_{n+1} &= y_n + b_1 h k_1 + b_2 h k_2
\end{aligned} \tag{5}
$$

ここで、a, c, b_1, b_2 は定数である。上記の形式に制限している理由は、既知の値 x_n, y_n を用いて k_1 を、それを用いて k_2 を、というように順次値を計算できるようにするためで、このような差分化の方法を陽的解法と言う。式 (5) から k_1, k_2 を消去すると

$$
\begin{aligned}
y_{n+1} &= y_n + b_1 h f(x_n, y_n) + b_2 h f\left(x_n + ch, y_n + ahf\right) \\
&= y_n + (b_1 + b_2) h f(x_n, y_n) + b_2 c h^2 \frac{\partial f}{\partial x} + a b_2 h^2 f \frac{\partial f}{\partial y} + \mathcal{O}(h^3)
\end{aligned}
$$

と表わせる。一方、$y_{n+1} = y(x_{n+1})$ について x_n 周りでのテイラー展開を行うと、

$$
\begin{aligned}
y(x_{n+1}) &= y(x_n) + h\frac{dy}{dx} + \frac{h^2}{2}\frac{d^2y}{dx^2} + \mathcal{O}(h^3) \\
&= y(x_n) + h f(x_n, y_n) + \frac{h^2}{2}\frac{df}{dx} + \mathcal{O}(h^3) \\
&= y(x_n) + h f(x_n, y_n) + \frac{h^2}{2}\left(\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial y}\right) + \mathcal{O}(h^3)
\end{aligned} \tag{6}
$$

より、差分式 (5) がテイラー展開 (6) の 2 次までの近似となるように定数 a, c, b_1, b_2 を決定すればよい。すなわち、

$$
\begin{aligned}
b_1 + b_2 &= 1 \\
b_2 c &= \frac{1}{2} \\
a b_2 &= \frac{1}{2}
\end{aligned}
$$

このような選び方は複数あり、中点法 $a = c = 1/2, b_1 = 0, b_2 = 1$ はその一つとなっている。その他には、例えば、ホイン (Heun) 法 $a = c = 1, b_1 = b_2 = 1/2$ などが知られる。

2 初期値問題 –ボールの軌道–

z 方向に一定の重力加速度 $-g$ を受けるボールの運動を考える。ボールの位置 $\boldsymbol{r}$ の運動方程式は

$$
m\frac{d^2\boldsymbol{r}}{dt^2} = -mg\hat{\boldsymbol{z}} \tag{7}
$$

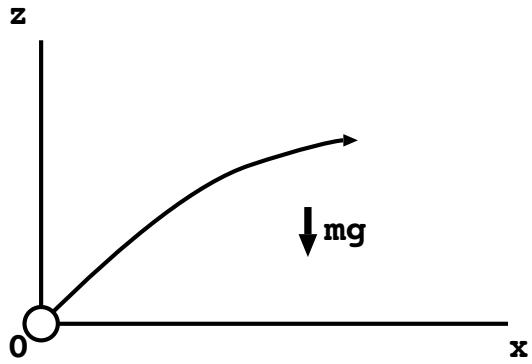

Figure 1: 一定重力加速度中のボールの軌道

但し、$\hat{z}$ は z 方向の単位ベクトルである。

初期にある速度で投げ上げた場合のボールの運動を考えよう。初期に $v_y = 0$ であれば以降もずっと $v_y = 0$ であるので、運動は $x - z$ 面内に限定できる。するとこの運動は、以下のような 1 階の連立微分方程式とすることができる。

$$\frac{dx}{dt} = v_x \tag{8}$$

$$\frac{dz}{dt} = v_z \tag{9}$$

$$\frac{dv_x}{dt} = 0 \tag{10}$$

$$\frac{dv_z}{dt} = -g \tag{11}$$

質量を $m = 1\mathrm{kg}$、重力加速度を $g = 9.8\mathrm{m/s^2}$ とし、初期に原点から仰角 30 度、初速 $v(t = 0) = 20\mathrm{m/s}$ で投げ上げたボールの運動をオイラー法により求めるプログラムは、例えば以下のようになる。

```
#include <stdio.h>
#include <math.h>
#define G_ACC 9.8 /* 重力加速度 */
void calc_derivative(double xv[], double dxvdt[]);

int main(void){
  double dt = 0.01; /* 時間積分の刻み幅 dt (=h) */
  double t;         /* 時間 t */
  double xv[4];     /* 粒子の位置と速度を xv[0]=x,xv[1]=z,xv[2]=vx,xv[3]=vz とまとめた配列 */
  double k1[4];     /* 同様に dx/dt, dz/dt, dvx/dt, dvz/dt をまとめた配列 */
  int i;
  /* 初期値の設定 */
  t = 0.0;
  xv[0] = 0.0; /* x(t=0) */
  xv[1] = 0.0; /* z(t=0) */
  xv[2] = 20.0 * cos(2.0*M_PI*30.0/360.0); /* vx(t=0) */
  xv[3] = 20.0 * sin(2.0*M_PI*30.0/360.0); /* vz(t=0) */
  printf("#  t,  x,  z,  vx,  vz\n");
  printf("%f %f %f %f %f\n", t, xv[0], xv[1], xv[2], xv[3]);

  /* 時間積分 */
  for (t = dt; t <= 2.0; t=t+dt){
    calc_derivative(xv,k1);
    for (i=0;i<4;i++) {
      xv[i] = xv[i] + dt * k1[i];
    }
```

```
    printf("%f %f %f %f %f\n", t, xv[0], xv[1], xv[2], xv[3]);
  }
  return 0;
}

void calc_derivative(double xv[], double dxvdt[]){
  dxvdt[0] = xv[2];  /* dx/dt=vx */
  dxvdt[1] = xv[3];  /* dz/dt=vz */
  dxvdt[2] = 0.0;    /* dvx/dt=0 */
  dxvdt[3] = -G_ACC; /* dvz/dt=-g */
}
```

このプログラムを実行すると、各時刻 t における t, x, y, v_x, v_y の値が標準出力に書き出される。実行結果を保存するには、プログラム実行時にリダイレクトしてファイルに書き出せばよい。

2.1 刻み幅の選び方

刻み幅 h はどの程度の大きさに選ぶべきだろうか？オイラー法の場合、$\mathcal{O}(h^2)$ の誤差が累積するというのだから、h は小さい方が良いのであろうが、いくつくらいなら”小さい”と思えるだろう。今、式 (1) に基づく時間積分を考えているので、テイラー展開より (簡単のため、1 変数 $\boldsymbol{y} = y$ とする)、

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + h\frac{dy}{dx} + \frac{h^2}{2}\frac{d^2y}{dx^2} + \mathcal{O}(h^3) \\ &= y(x_n) + hf(x_n, y_n) + \frac{h^2}{2}\frac{df}{dx} + \mathcal{O}(h^3) \\ &= y(x_n) + hf(x_n, y_n) + \frac{h^2}{2}\left(\frac{\partial f}{\partial x} + f\frac{\partial f}{\partial y}\right) + \mathcal{O}(h^3) \end{aligned}$$

となる。オイラー法が良い近似となるためには、第 2 項 ($\sim hf$) に比べて、第 3 項 ($\sim h^2 d_x f$) が小さく、つまり、$h \ll |f/d_x f|$ でなければならない。右辺は支配方程式が記述する物理現象の変化する時間スケールを表わしているので、それよりは小さな時間刻み幅を取らなければならない。直感的にも納得できるだろう。これはあくまで最低限の制約で、実用上は、求めたい精度で計算できているかどうか、時間刻み幅を小さくしていったときにどの程度数値解が変化しうるかという収束性を確認することになる。

2.2 得られた数値解は果たして正しいか

求根アルゴリズムの場合には、得られた数値解が $f(x) = 0$ をどの程度満たしているか、グラフなどから解のおおよその値が分かっている場合はそれと遜色ないかなどの検証が行えたが、この常微分方程式の場合はどうだろう。

- まず、時間刻み幅が粗すぎて真の解とは異なる軌道となっているかもしれない。これに関しては、前述の通り、時間刻み幅を小さくして収束性を確認すればよい。とはいえ、時間刻み幅に対して数値解が収束していたとしても、結果が正しい保証はない。
- 今回の例では $x(t), z(t)$ など解析的に一般解が分かるが、解析解が分からないので数値計算をする場合がほとんどであろう。そうした場合でも、解が物理的に満たすべき性質、例えばエネルギー保存則などが分かっている場合には、計算結果の検証の一つとして用いることができる。

数値シミュレーションプログラムを実行して終わり、ではなく、計算結果が収束しているか、物理的に正しそうか、多面的に検証してみる姿勢が重要である。

3 境界値問題

前節の斜方投射の問題では、ある投射速度と角度に対しボールがどのような軌跡を描き、飛距離がどうなるかを調べる初期値問題であった。今度は、ある飛距離を出すために必要な初速度や投げ上げ角度を計算することを考える。すなわち、境界値問題として微分方程式を解く。ここでは簡単のため、投げ上げ角度を 30 度に固定し、ある目標飛距離 x_{t} を出すための初速度を求めてみよう (角度も変更する場合は、演習課題で取り組む)。

まず初速度の推測値を v とし、その時の着地点 $z=0$ までの飛距離を $x(v)$ とする。当然この $x(v)$ は一般的には $x_{\rm t}$ とは異なるので、ここから解を改良し求める真の解 $v^\star$ に逐次的に近付けていく。

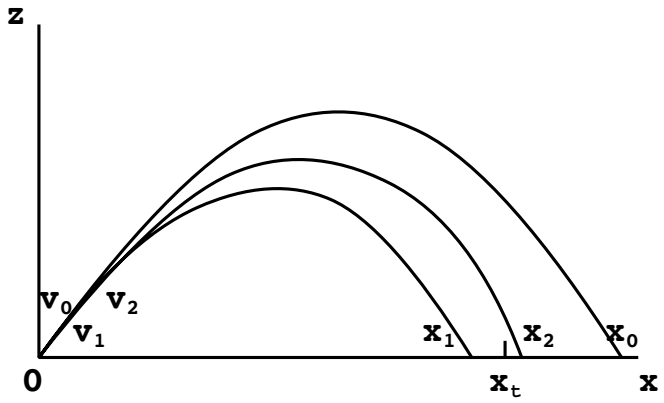

Figure 2: 目標飛距離 $x_{\rm t}$ を決めた際に、与えるべき初速度をシューティング法で求める。

推測値 v と真の解 $v^\star$ の誤差を δv とすると (つまり、$v^\star = v + \delta v$)、Taylor 展開より、

$$x_{\rm t} = x(v^\star) = x(v+\delta v) = x(v) + \delta v \frac{\partial x}{\partial v} + \frac{\delta v^2}{2}\frac{\partial^2 x}{\partial v^2} + \cdots \tag{12}$$

となり, Newton-Raphson 法のときと同じ考えで δv の 1 次までの近似を適用すれば

$$x(v) - x_{\rm t} + \delta v \frac{\partial x}{\partial v} \simeq 0 \tag{13}$$

から δv を求めれば良いことが分かる。ここで問題となるのが $\frac{\partial x}{\partial v}$ であるが、ここでは以下のように差分近似により数値的に求める。n 回の改良後の初期速度を v_n とし、その時の飛距離を $x_n (\equiv x(v_n))$ とすると

$$\frac{\partial x}{\partial v} = \frac{x_n - x_{n-1}}{v_n - v_{n-1}}. \tag{14}$$

すなわち、(13) 式に (14) 式を代入し, 次の解の改良値を

$$v_{n+1} = v_n + \delta v = v_n - \frac{v_n - v_{n-1}}{x_n - x_{n-1}}(x_n - x_{\rm t}) \tag{15}$$

とすれば良い。以上のような方法をシューティング法と呼ぶ。例えば、以下のようなプログラムを作成すれば良い。

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define G_ACC 9.8 /* 重力加速度 */
#define ACC 1.0e-12
void calc_derivative(double xv[], double dxvdt[]);

/* シューティング法により目標とする飛距離を出すために必要な初速度を推定する */
int main(){
  double x_target;
  double x, v, x_prev, v_prev, dv;
  char buf[30];
  int i;
```

```
  double landing_point(double v0);
  /*目標飛距離の入力*/
  printf("distance ?");
  fgets(buf,sizeof(buf),stdin);
  if (sscanf(buf,"%lf",&x_target) != 1) {
    fputs("input error\n", stderr);
    exit(-1);
  }

  v = 20.0; /* 初期推定値 */
  x = landing_point(v);
  v_prev = 1.01 * v; /* 微分の差分近似のために初期推定値を少しずらしたもの */
  x_prev = landing_point(v_prev);

  i = 0;
  do {
    i++;
    dv = - (v - v_prev) * (x - x_target) / (x - x_prev); /* 解の改良 */
    v_prev = v;
    x_prev = x;
    v = v + dv;
    x = landing_point(v);
      printf("i=%d, v=%25.16f: x=%25.16f\n", i, v, x);
    } while(fabs(x-x_target) >= ACC); /* 収束判定 */
  return 0;
}

/* 初速度 v0 を入力として、着地点 (z=0) の x 座標を返り値とする関数 */
/* 初期値問題の main 関数をほぼそのまま利用している */
double landing_point(double v0){
  double dt = 0.01; /* 時間積分の刻み幅 dt (=h) */
  double t;         /* 時間 t */
  double xv[4];     /* 粒子の位置と速度を xv[0]=x,xv[1]=z,xv[2]=vx,xv[3]=vz とまとめた配列 */
  double k1[4];     /* 同様に dx/dt, dz/dt, dvx/dt, dvz/dt をまとめた配列 */
  double x0, z0, x1, z1, x_landing;
  int i;
  /* 初期値の設定 */
  t = 0.0;
  xv[0] = 0.0;
  xv[1] = 0.0;
  xv[2] = v0 * cos(2.0*M_PI*30.0/360.0);
  xv[3] = v0 * sin(2.0*M_PI*30.0/360.0);

  /* 時間積分 */
  for (t = dt; xv[1] >= 0.0; t=t+dt){
    calc_derivative(xv,k1);
    for (i=0;i<4;i++) {
      xv[i] = xv[i] + dt * k1[i];
    }
  }

  /* 着地点を (x0,z0>0) と (x1,z1<0) の 2 点の内挿で求める */
  x0 = xv[0] - dt * k1[0];
  z0 = xv[1] - dt * k1[1];
```

```
  x1 = xv[0];
  z1 = xv[1];
  x_landing = (x1*z0 - x0*z1)/(z0-z1);
  return x_landing;
}

void calc_derivative(double xv[], double dxvdt[]){
  dxvdt[0] = xv[2];  /* dx/dt=vx */
  dxvdt[1] = xv[3];  /* dz/dt=vz */
  dxvdt[2] = 0.0;    /* dvx/dt=0 */
  dxvdt[3] = -G_ACC; /* dvz/dt=-g/m */
}
```

注) 内挿の意味：着地点 $(x, z = 0)$ を求める際に、時間積分した値がちょうど $z = 0$ となることは稀であるので、下図のように着地点通過前 $(x_0, z_0 > 0)$ と通過後 $(x_1, z_1 < 0)$ の線形内挿により着地点を $x = (x_1 z_0 - x_0 z_1)/(z_0 - z_1)$ として求めている。

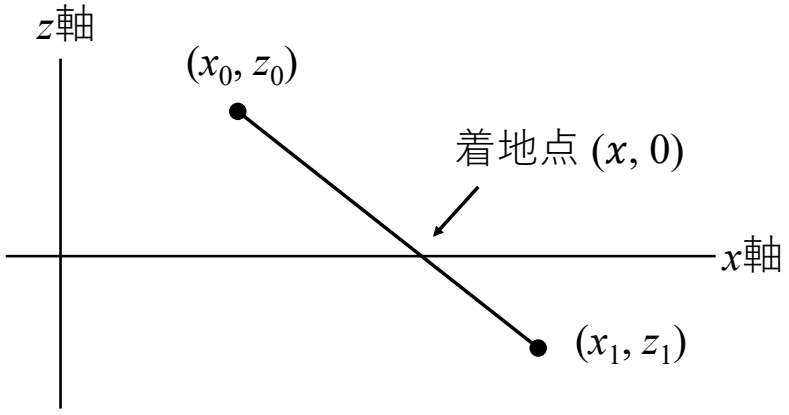

Figure 3: 着地点の線形内挿

閑話：その他の初期値問題の解法

- 第 1.2 節で導出した方法で、3 次精度や 4 次精度などの高次精度ルンゲクッタ法も導出できる。2 次精度や 4 次精度ルンゲクッタ法では、それぞれ 2 段 (k_1, k_2)、4 段 (k_1, k_2, k_3, k_4) の計算が必要であった。一方、5 次精度以上に高次精度化していくと、導出が煩雑になるだけではなく、精度より多くの段数が必要となるという問題が生じる（表 1)。段数が増大しない範囲で精度を確保できるという点で、4 次精度ルンゲクッタ法が広く用いられる。加えて、1 次や 2 次精度ルンゲクッタ法では、単振動のような線形常微分方程式の固有値が純虚数となる場合に、時間とともに数値解の振幅が増大していく数値不安定性の問題がある。固有値が純虚数の場合にも安定に計算できるためには、3 次精度以上のルンゲクッタ法を用いる必要がある。

次数 p	2	3	4	5	6	7	8	10
段数 s	2	3	4	6	7	9	11	≤ 17 [Hairer(1978)]

Table 1: ルンゲクッタ法の段数 s と近似精度の次数 p [J. C. Butcher, "A history of Runge-Kutta methods", Applied Numerical Mathematics 20, 247 (1996) より]

- また、第 1.2 節では、既知の値を支配方程式に代入することで各段を順次計算できる陽的解法に限って説明

を行った。例えば、陽的オイラー法は

$$\frac{d\boldsymbol{y}}{dx} \simeq \frac{\boldsymbol{y}_{n+1} - \boldsymbol{y}_n}{h} = \boldsymbol{f}(x_n, \boldsymbol{y}_n)$$

と近似したことに相当する。一方、右辺を x_{n+1} での値を用いて、

$$\frac{d\boldsymbol{y}}{dx} \simeq \frac{\boldsymbol{y}_{n+1} - \boldsymbol{y}_n}{h} = \boldsymbol{f}(x_{n+1}, \boldsymbol{y}_{n+1})$$

と近似することも可能である。これを陰的オイラー法と言う。$\boldsymbol{y}_{n+1}$ は求めたい未知の値であったので、このままでは右辺は計算できない。方程式が線形 $\boldsymbol{f}(x_{n+1}, \boldsymbol{y}_{n+1}) = A\boldsymbol{y}_{n+1}$ の場合には、$\boldsymbol{y}_{n+1} = (I - hA)^{-1}\boldsymbol{y}_n$ などとして求めることができる（I は恒等行列)。方程式が非線形の場合、$\boldsymbol{y}_{n+1} - \boldsymbol{y}_n - h\boldsymbol{f}(x_{n+1}, \boldsymbol{y}_{n+1}) = 0$ を満たす $\boldsymbol{y}_{n+1}$ の値を Newton Raphson 法などの反復法により探す必要がある。煩雑に思えるかもしれないが、拡散方程式を空間格子上で離散化して解く場合などに、数値安定性の観点から陰的解法が用いられることがある。

- ルンゲクッタ法は、ある座標での値 $(x_n, \boldsymbol{y}_n)$ のみを用いて、逐次的に隣の座標での値を求めていく方法であった。これとは別に、複数の座標での値 $(x_n, \boldsymbol{y}_n), (x_{n-1}, \boldsymbol{y}_{n-1}), \cdots$ が既知であった場合に、隣の座標の値 $(x_{n+1}, \boldsymbol{y}_{n+1})$ を求める方法を多段解法と呼ぶ。例えば、線形多段解法である 2 次精度アダムスバッシュフォース (Adams Bashforth) 法では、

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h}{2}\left[3\boldsymbol{f}(x_n, \boldsymbol{y}_n) - \boldsymbol{f}(x_{n-1}, \boldsymbol{y}_{n-1})\right]$$

として解を更新していく。