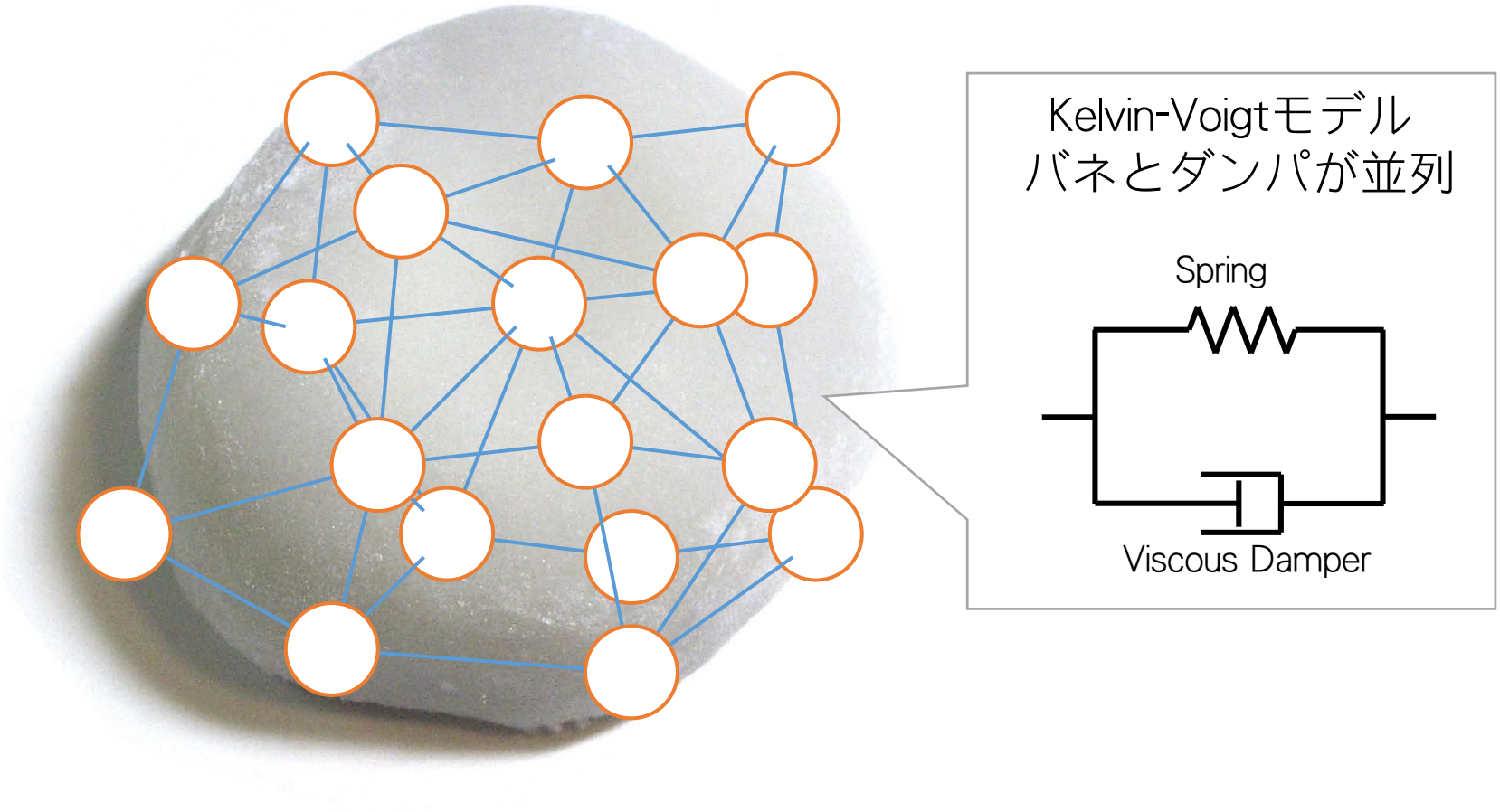
エルゴノミクスコンピューティング実習

物理シミュレーション

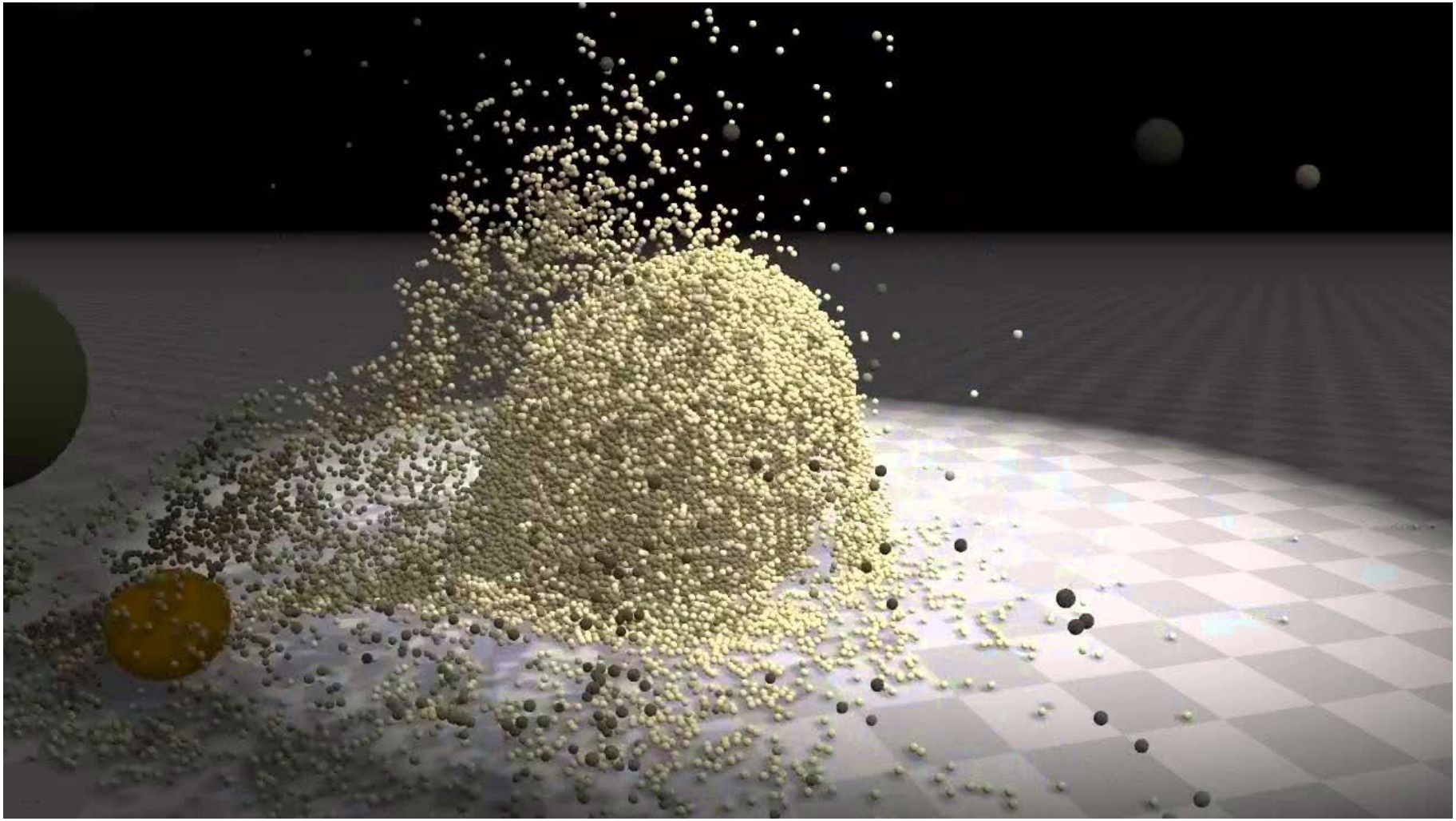
人間システム工学科 井村 誠孝
m.imura@kwansei.ac.jp

バネ-ダンパ-質点モデル

- 対象物を質点群とそれらを結ぶバネ(およびダンパ)で表す.



Particle Physics の現在の到達点



Unified Particle Physics for Real-Time Applications, Miles Macklin, Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, ACM Transactions on Graphics (SIGGRAPH 2014), 33(4)

個々の質点は，運動方程式に従って運動する

- Newtonの運動方程式

$$m\frac{d^2\mathbf{x}}{dt^2} = \mathbf{F}$$

- バネ-ダンパ-質点モデルの場合，質点に働く力
  - バネの弾性力 – バネの伸びに比例
  - ダンパの粘性力 – 質点間の相対速度に比例
  - 外力 – 各質点のみに依存(あるいは一様)
    - 例: 重力

運動のシミュレーション

- まず物体に働く力を全て考える.

- 運動方程式(微分方程式になる)を書き下せる.

- ほとんどの場合，解析的な解を求めることはできないので，微分の部分を差分に置き換えて，離散的な時刻の状態を次々に計算していく.

常微分方程式の解法

常微分方程式とは

- t の関数 $y(t)$ を考える.
  - 変数名は任意. x の関数 $y(x)$ でも, t の関数 $x(t)$ でも考え方は同じ.
- $t, y(t), y'(t)$ (yの1階導関数) を含む方程式を考える.
  - 例えば $y' = (1-t)y$

$$y'(t) = \frac{dy(t)}{dt}$$

(t)はしばしば省略される.
簡潔だが, わかりにくくなる.

- 微分方程式とは
  - 従属変数(この場合はy)の導関数(この場合はy')を含む方程式
- 常微分方程式とは
  - 独立変数(この場合はt)が一つである微分方程式

解析解

- 微分方程式の解析解が得られることは稀である.
  - この例では解析解が得られる.

$$y' = (1-t)y$$

$$\Leftrightarrow \frac{dy}{dt} = (1-t)y$$

$$\Leftrightarrow \frac{dy}{y} = (1-t)dt$$

  - 積分して積分定数を整理すると

$$y(t) = Ae^{-(1-t)^2/2}$$

初期値問題

- 任意定数 A はどう決まるか?
  → 初期値によって決まる.
  - 例　$y' = (1-t)y$

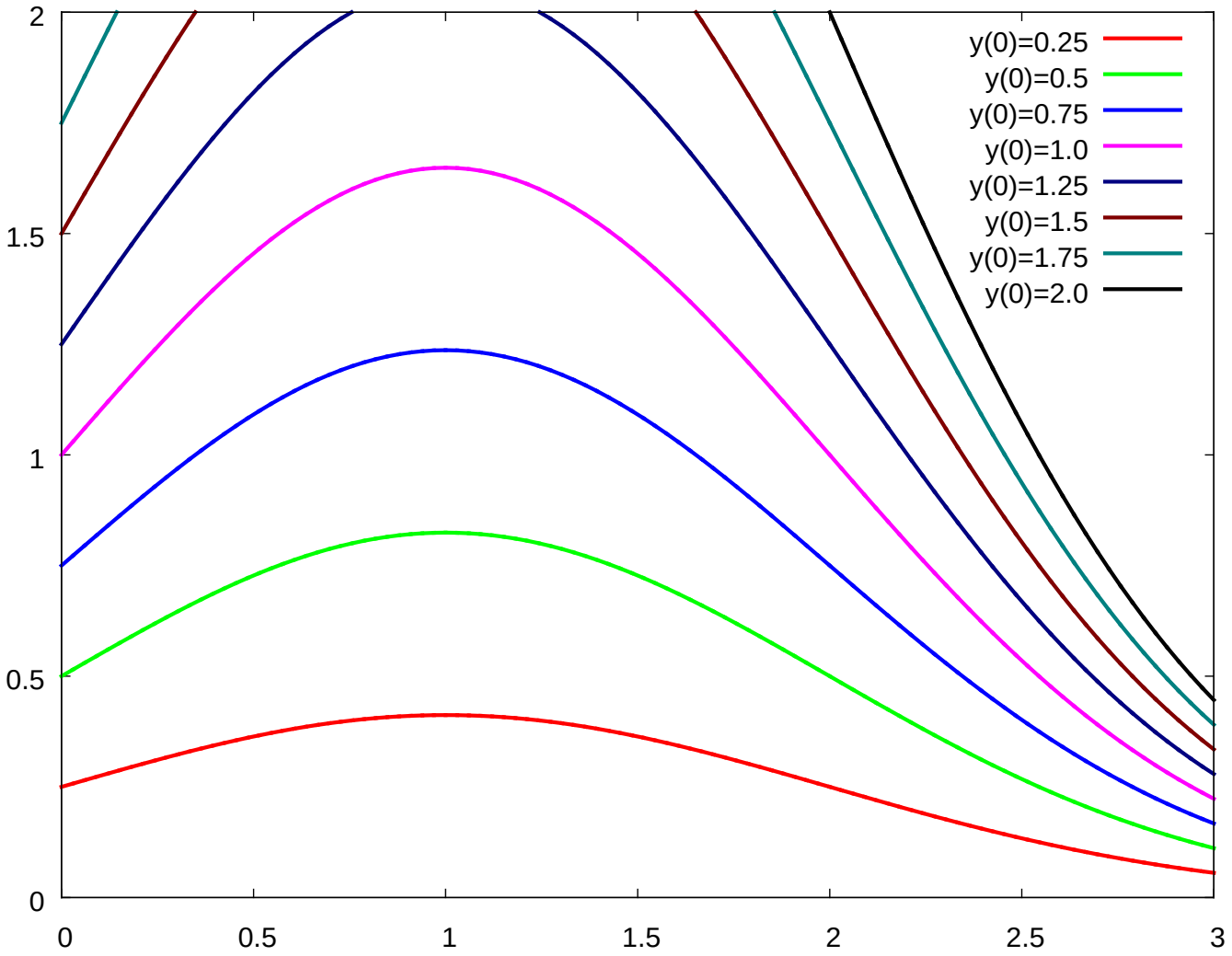
    $y(t) = Ae^{-(1-t)^2/2}$

    $y(0) = 1 \Leftrightarrow Ae^{-1/2} = 1$

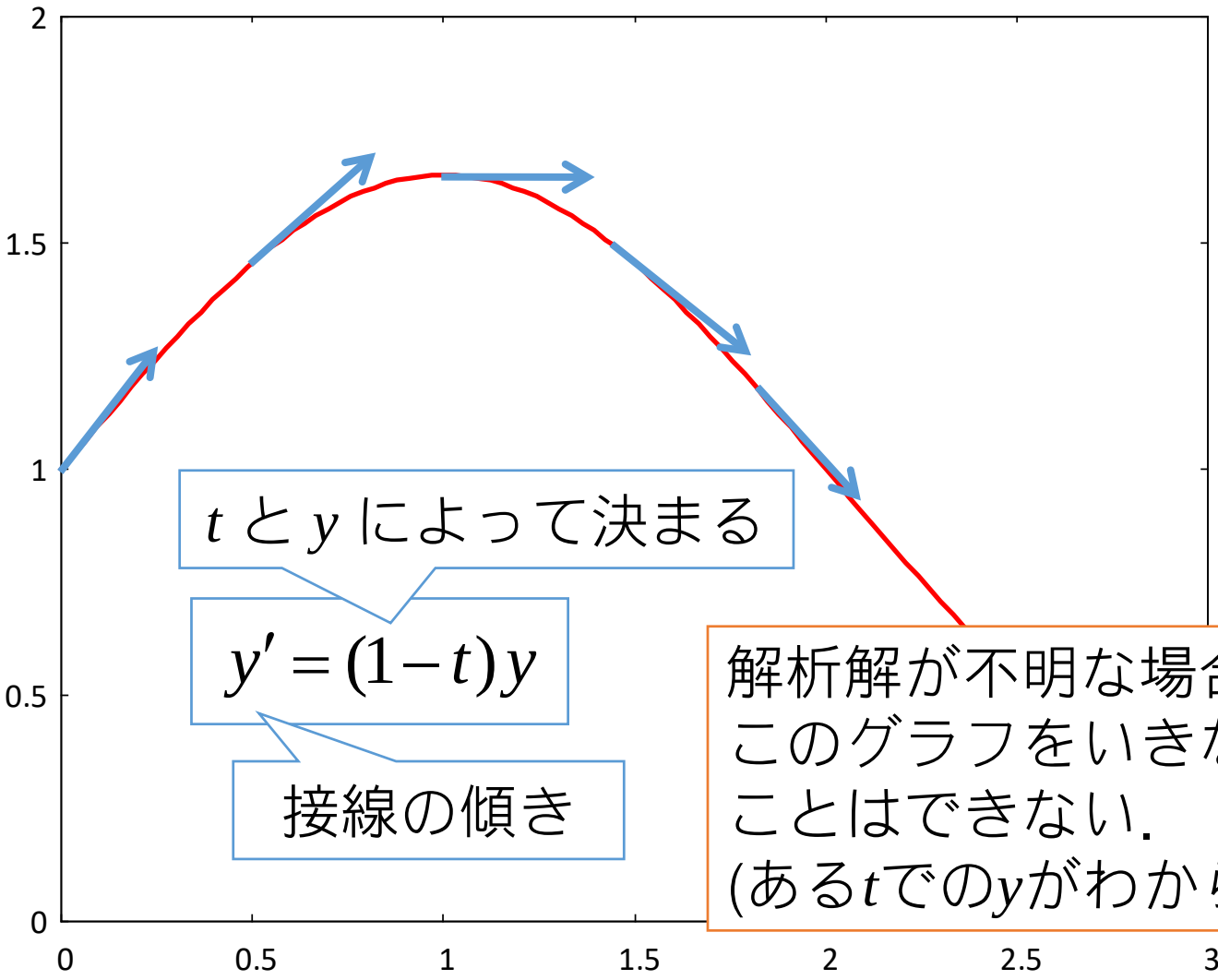
    よって，$A = e^{1/2}$
- 初期条件: ある t での y の値を定める条件
- 初期値問題: 常微分方程式を初期条件の下で解く

初期条件が決まらないと，解は一意に決まらない.

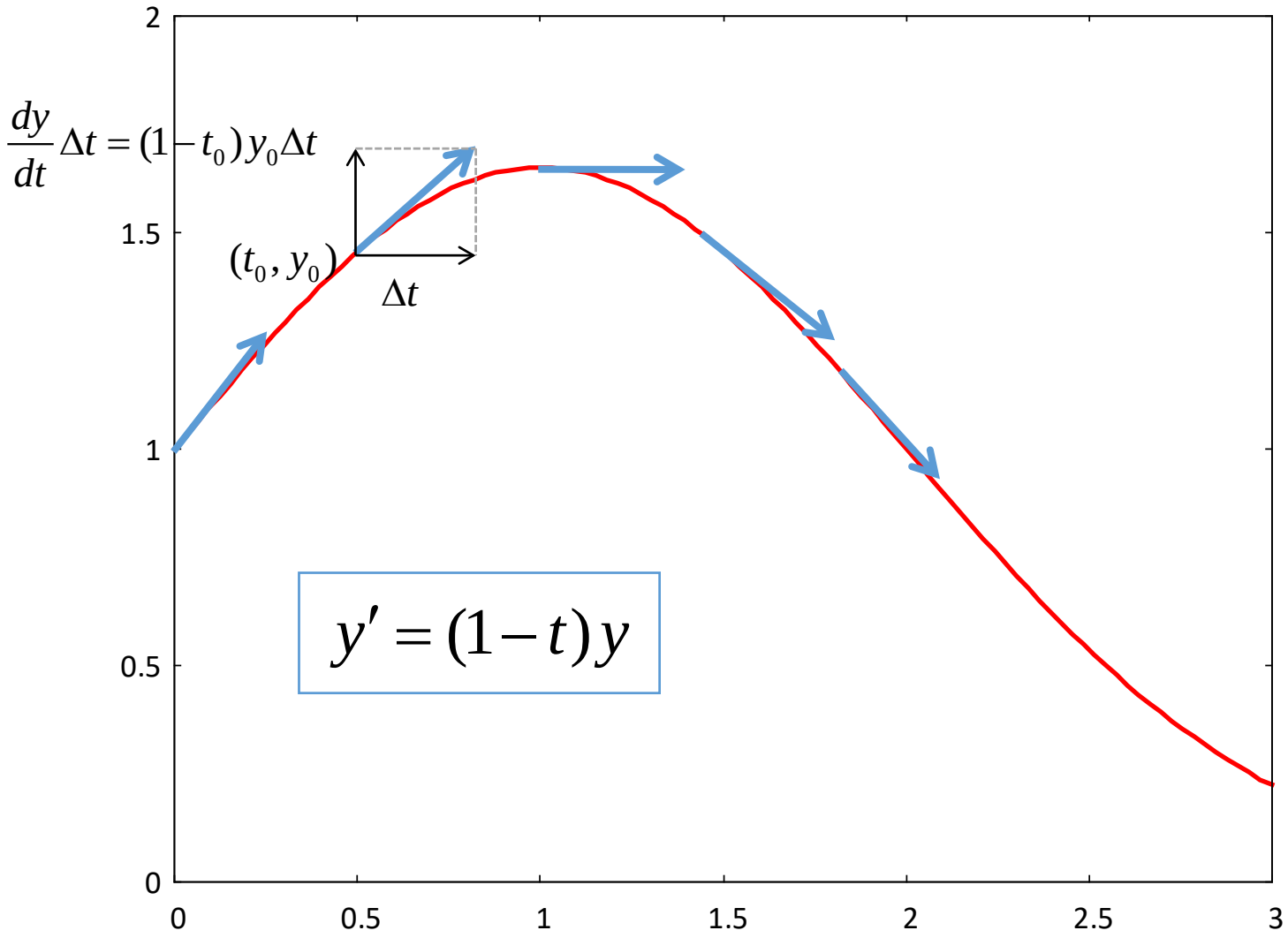
初期値問題の解の，初期値依存性



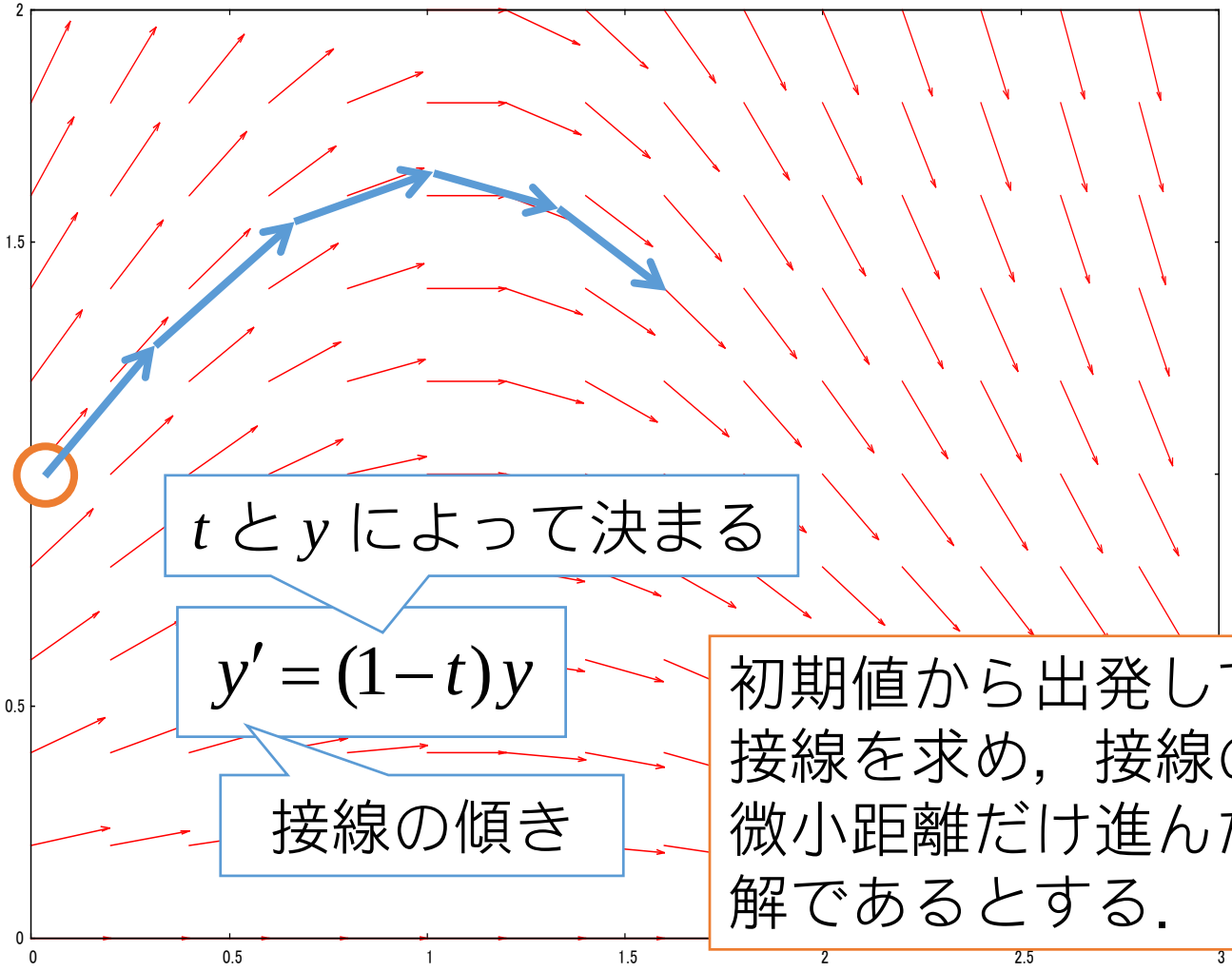
微分方程式の意味するところ



接ベクトルの成分


$\frac{dy}{dt}\Delta t = (1-t_0)y_0\Delta t$
(t_0, y_0)
Δt
$y' = (1-t)y$
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3

解析解不明の場合，どうするか?



初期値から出発して，
接線を求め，接線の方向に
微小距離だけ進んだ点が
解であるとする.

ここまでのまとめ

- 常微分方程式とは，独立変数が一つである微分方程式.
  - 独立変数が複数ある微分方程式は，偏微分方程式.
- 常微分方程式の解を一意に定めるためには，初期値が必要.
- 数値解は，初期値から出発して，微分方程式から得られる接線方向に少しずつ進んでいくことで，求められる.

差分法

微分を差分で近似するとは

- $f(t)$をtの関数とする.
- $f(t)$の微分係数の定義

$$f'(t) = \lim_{h \to 0} \frac{f(t+h) - f(t)}{h}$$

- 数値計算では h を無限に小さくできないので，差分で代用する.

$$f'(t) \to \frac{f(t+\Delta t) - f(t)}{\Delta t}$$

差分にもいろいろ

- 前進差分

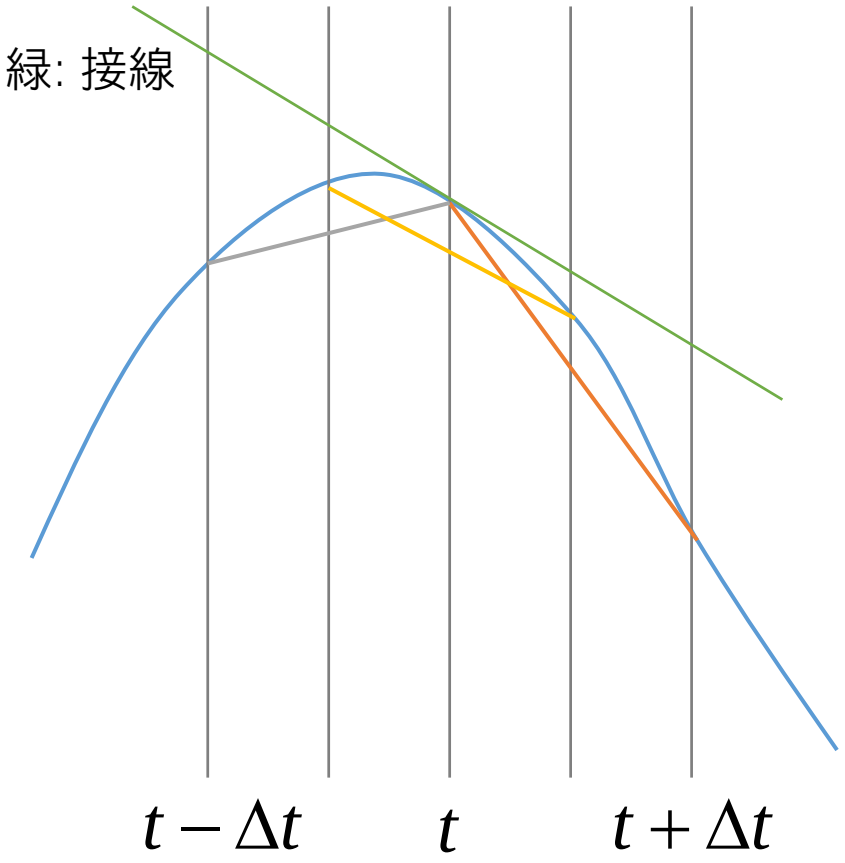
$$\frac{f(t+\Delta t)-f(t)}{\Delta t}$$

- 後退差分

$$\frac{f(t)-f(t-\Delta t)}{\Delta t}$$

- 中心差分

$$\frac{f(t+\Delta t/2)-f(t-\Delta t/2)}{\Delta t}$$



常微分方程式の差分解

- 一般的な1階常微分方程式を考える.

$$y' = f(t, y)$$
$$y(0) = a$$

- 微分を差分で置き換える→差分方程式

$$\frac{y(t+\Delta t) - y(t)}{\Delta t} = f(t, y(t))$$ さしあたり前進差分

- ある時刻 t までの解がわかっているなら,

$$y(t+\Delta t) = y(t) + f(t, y(t))\Delta t$$ 上式を変形しただけ

未知の値　　既知の値

tおよびyの離散化

- 微分を差分で置き換えた場合，とびとびな時刻でのみ値が決まる．
- 時刻の離散化: $t \to i\Delta t \equiv t_i$
- 時刻 t_i における解:

$$y(t) \to y(i\Delta t) = y(t_i) \equiv y_i$$

- 差分方程式:

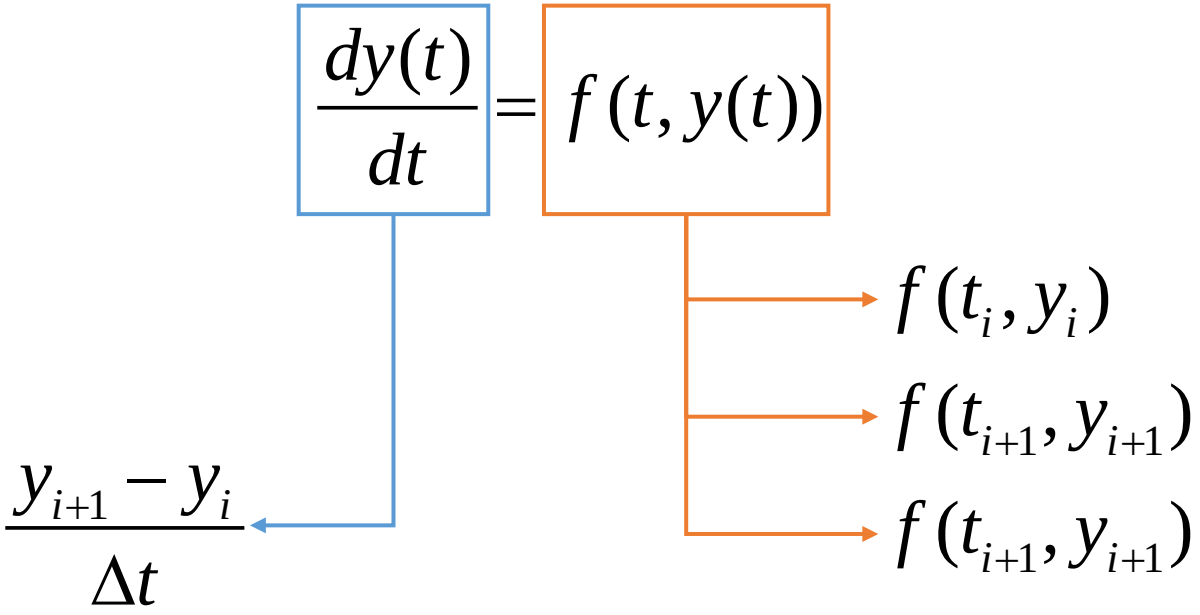
$$y_{i+1} = y_i + f(t_i, y_i)\Delta t$$

$$y_0 = a \quad \text{(初期値)}$$

※ただしこの差分方程式は安定性が低いので注意

差分方程式を解く

- 様々な離散化手法がある.



時刻 t_i から t_{i+1} までの変化を考えるときに

どの時刻の微分係数を用いるか

様々な数値積分法

- オイラー法　陽解法

$$\frac{y_{i+1}-y_i}{\Delta t}=f(t_i,y_i)$$

右辺に未知の値が入っていると陰解法

- 完全陰解法　陰解法

$$\frac{y_{i+1}-y_i}{\Delta t}=f(t_{i+1},y_{i+1})$$

- クランク・ニコルソン法　陰解法

$$\frac{y_{i+1}-y_i}{\Delta t}=\frac{1}{2}\left\{f(t_i,y_i)+f(t_{i+1},y_{i+1})\right\}$$

ここまでのまとめ

- 運動は時間に関する微分を含む微分方程式で表される.
- 微分を差分に置き換える(差分法).
- 独立変数(ここではt)を離散化→従属変数(ここではy)も離散的に求まる.
  - 微分方程式を近似する漸化式が得られる.
- 初期値問題の場合，初期値から順に漸化式に従って値を求めていく.

2階常微分方程式

2階常微分方程式

- 式中に含まれる導関数の最高階数が2階

$$y'' = f(t, y, y')$$
$$y(0) = a_1, y'(0) = a_2$$

- $y \to y_1, y' \to y_2$と置くことで，連立1階常微分方程式に帰着される．

$$y_1' = y_2$$
$$y_2' = f(t, y_1, y_2)$$
$$y_1(0) = a_1, y_2(0) = a_2$$

単振動

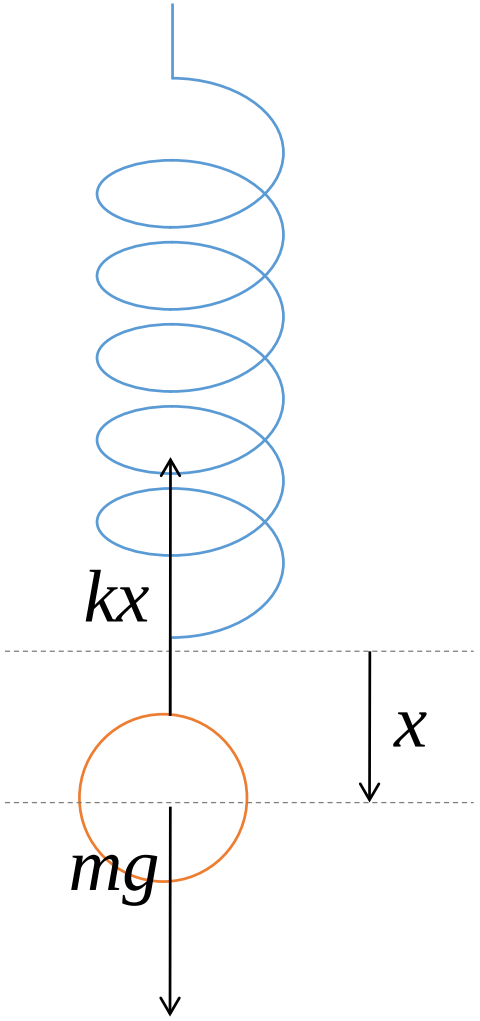
- バネの単振動
  - 自然長からの変位: x

$$m\frac{d^2x}{dt^2} = -kx + mg = -k\left(x - \frac{mg}{k}\right)$$

  - つりあいの位置からの変位: y

$$y \equiv x - \frac{mg}{k}$$

$$m\frac{d^2y}{dt^2} = -ky$$



2階常微分方程式の例: 単振動

- ニュートンの運動方程式(係数＝1とする)

$$y'' = -y$$

$$y(0) = 1, y'(0) = 0$$

  - 解析解: $y = \cos(t)$

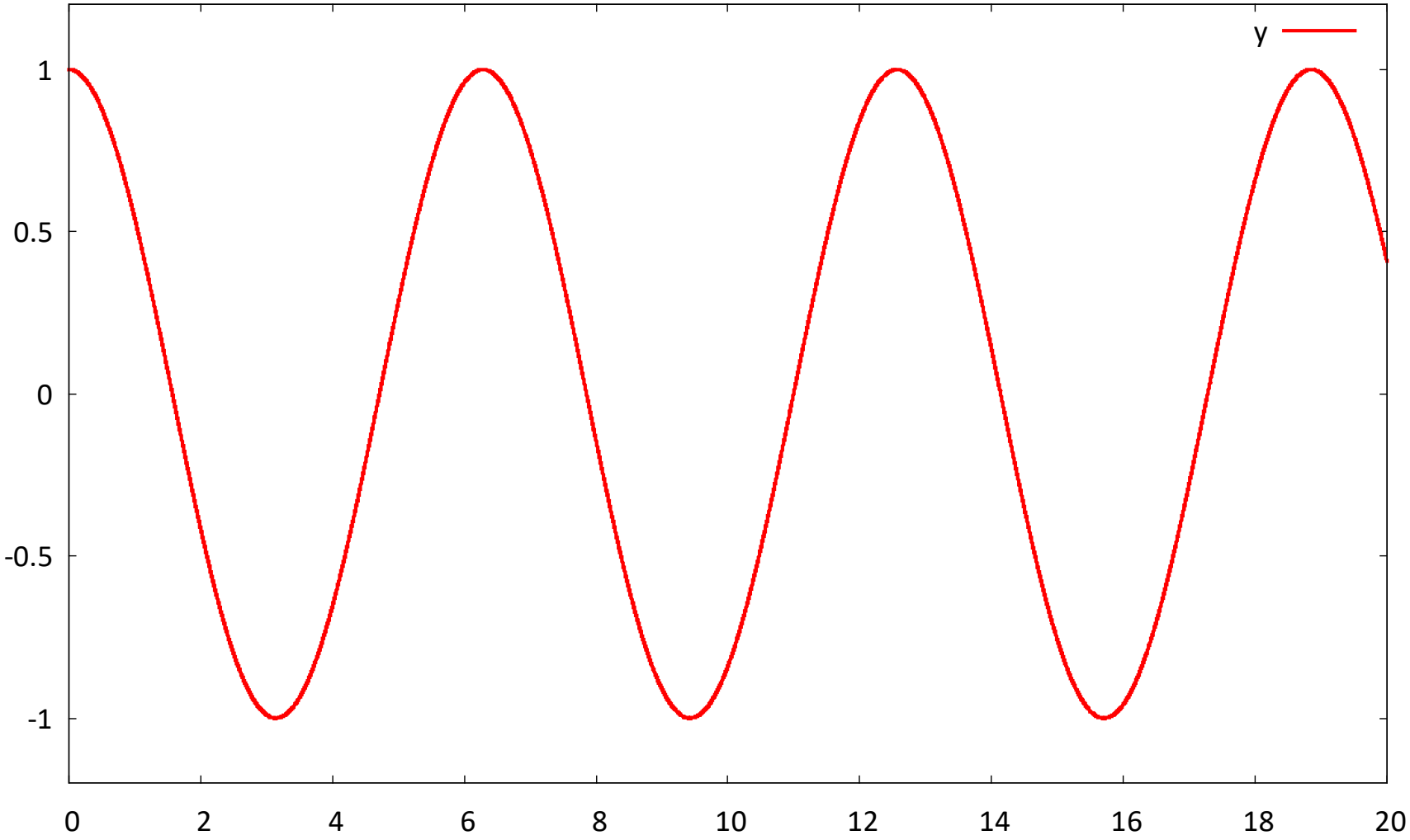
- 連立1階常微分方程式に変換: $y \to y_1, y' \to y_2$

$$y_1' = y_2$$

$$y_2' = -y_1$$

$$y_1(0) = 1, y_2(0) = 0$$

単振動の例



クラスを作る

バラバラの変数と関数をまとめる

変数

m

x

y

vx

vy

etc.

(Cで言うところの)
構造体

```
class Particle {
  float m;
  float x, y;
  float vx, vy;
  :
};
```

クラス

```
class Particle {
  float m;
  float x, y;
  float vx, vy;
  :

  void init() {
    :
  }

  void move() {
    :
  }
};
```

関数

〜を初期化する

〜を移動する

〜を描画する

etc.

オブジェクト指向特有の用語

変数の型

変数名
(実体)

```
int i;
```

```
Particle p;
```

クラス

インスタンス
(or オブジェクト)

フィールド
(or メンバ)
＝変数

メソッド
＝関数

```
class Particle {
  float m;
  float x, y;
  float vx, vy;
  :

  void init() {
    :
  }

  void move() {
    :
  }
};
```

物体を沢山にしたい

- クラスのないとき

```
int nParticle = 100;
float[] m = new float[nParticle];
float[] x = new float[nParticle];
float[] y = new float[nParticle];
:
```

- クラスのあるとき

```
int nParticle = 100;
Particle[] particles = new Particle[nParticle];
```