数値計算と精度保証付き数値計算

柏木 雅英
kashi@waseda.jp
http://verifiedby.me/

早稲田大学 基幹理工学部 応用数理学科

概要

- いろいろな計算
- コンピュータによる数値計算
- 数値計算の誤差の話

足し算

足し算

$$1+1=2$$

$$8+5=13$$

引き算

引き算

$$13 - 5 = 8$$

$$2 - 5 = -3$$

(負の数)

掛け算

掛け算

$$2 \times 3 = 6$$

$$12 \times 5 = 60$$

割り算

割り算

$$6 \div 2 = 3$$

$$17 \div 5 = 3 \ldots 2$$

$$17 \div 5 = 3.4$$

$$17 \div 5 = \frac{17}{5}$$

(分数、小数)

方程式

方程式

$$2x + 3 = 7$$

(答え) $x = 2$

(両辺がちょうど釣り合うような数を求める)

連立方程式

連立方程式

$$
\begin{aligned}
2x + 4y &= 16 \\
x + y &= 5 \\
\text{(答え)} \qquad & x = 2,\ y = 3
\end{aligned}
$$

(鶴亀算、連立一次方程式)

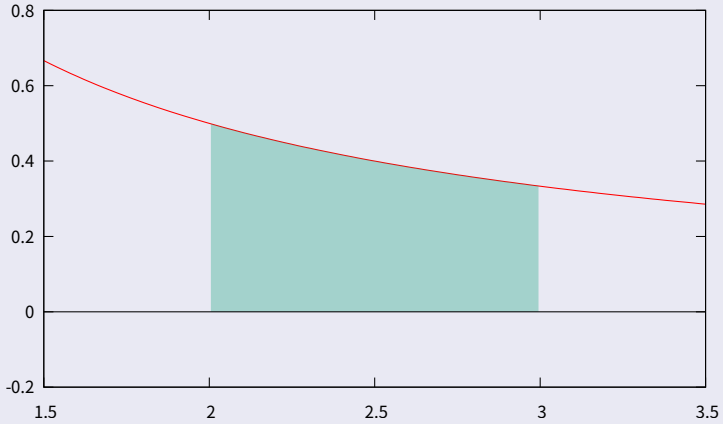
2 次方程式

$$
\begin{aligned}
x^2 - 3x + 2 &= 0 \\
\text{(答え)} \quad & x = 1, 2 \\
x^2 + x - 1 &= 0 \\
\text{(答え)} \quad & x = \frac{-1 \pm \sqrt{5}}{2} \\
x^2 + x + 1 &= 0 \\
\text{(答え)} \quad & x = \frac{-1 \pm \sqrt{3}i}{2}
\end{aligned}
$$

(平方根、複素数)

積分

問題



$y = 0$, $y = \frac{1}{x}$, $x = 2$, $x = 3$ で囲まれた部分の面積を求めよ。

答え

$$\int_2^3 \frac{1}{x} dx = [\log x]_2^3 = \log \frac{3}{2}$$

微分方程式 (1)

問題

時刻 0 で 温度 x_0 度の物体を 0 度の空間に放置したとき、t 秒後の温度は? ただし、単位時間あたりの冷却は、その時刻での周囲との温度差に比例する (比例定数 a) ものとする。

方程式

$$\begin{aligned} \frac{d}{dt}x(t) &= -ax(t) \\ x(0) &= x_0 \end{aligned}$$

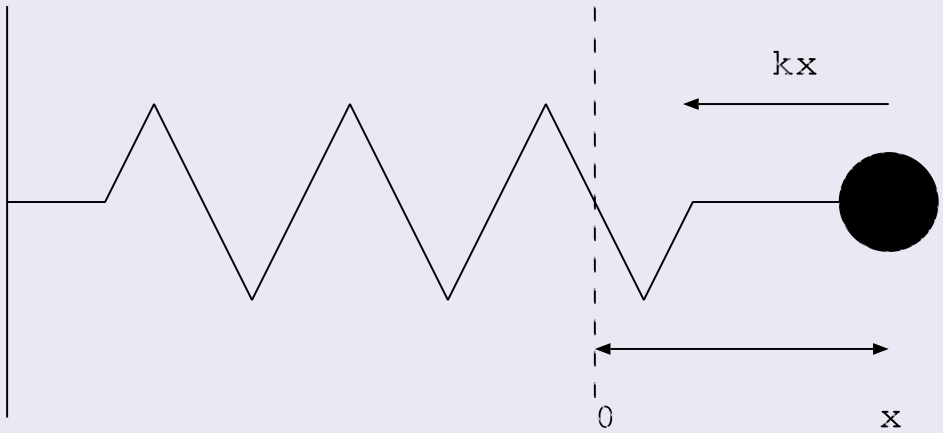
答え

$$x(t) = x_0 e^{-at}$$

(微分方程式、指数関数)

微分方程式 (2)

問題



図のようにバネ定数 k (バネの引っ張る力がバネの (伸びた/縮んだ) 量に比例する) のバネに重さ m の重りをつけて長さ 1 だけ引っ張って、手を離すとどんな運動をするか?

微分方程式 (3)

方程式

$$\begin{aligned} m\frac{d^2}{dt^2}x(t) &= -kx(t) \\ x(0) &= 1 \\ x'(0) &= 0 \end{aligned}$$

答え

$$x(t) = \cos(\sqrt{\frac{k}{m}}t)$$

(三角関数)

人類は方程式で世界を理解した

- ニュートン力学

$$F = ma$$

- 電磁気学 (マクスウェル方程式)

$$\begin{cases} \nabla \cdot \boldsymbol{B}(t, \boldsymbol{x}) & = 0 \\ \nabla \times \boldsymbol{E}(t, \boldsymbol{x}) + \dfrac{\partial \boldsymbol{B}(t, \boldsymbol{x})}{\partial t} & = 0 \\ \nabla \cdot \boldsymbol{D}(t, \boldsymbol{x}) & = \rho(t, \boldsymbol{x}) \\ \nabla \times \boldsymbol{H}(t, \boldsymbol{x}) - \dfrac{\partial \boldsymbol{D}(t, \boldsymbol{x})}{\partial t} & = \boldsymbol{j}(t, \boldsymbol{x}) \end{cases}$$

- 量子力学 (シュレーディンガー方程式)

$$i\hbar \frac{d}{dt}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle$$

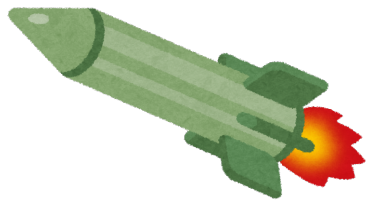
- 流体力学 (ナビエストークス方程式)

$$\frac{D\boldsymbol{v}}{Dt} = \frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \nabla)\boldsymbol{v} = -\frac{1}{\rho} \operatorname{grad} p + \nu \triangle \boldsymbol{v} + \boldsymbol{g}$$

方程式が解ければ予測ができる

- 天体の動き
- 天気予報
- 地球の気候変動 (温暖化)
- ビルや橋などの建造物の強度
- 自動車や飛行機の空気抵抗や強度
- ミサイル、核兵器など、軍事応用



解けない問題 (1)

5 次方程式

$$ax^5 + bx^4 + cx^3 + dx^2 + ex + f = 0$$

3 次方程式はカルダノの公式、4 次方程式はフェラリの公式が知られているが、5 次以上の方程式に解の公式は存在しない。

解けない問題 (2)

非線形方程式

$$\cos(x) = x$$

cos, log などを含む方程式は一般に解けない。

解けない問題 (3)

積分

$$\int_0^{10} \frac{\sin(x)}{\cos(x^2) + 1 + 2^{-10}} dx$$

特定の形を除いて積分は求まらないことが多い。

解けない問題 (4)

微分方程式

$$\begin{aligned} \frac{dx}{dt} &= 10(y-x) \\ \frac{dy}{dt} &= 28x - y - xz \\ \frac{dz}{dt} &= -\frac{8}{3}z + xy \end{aligned}$$

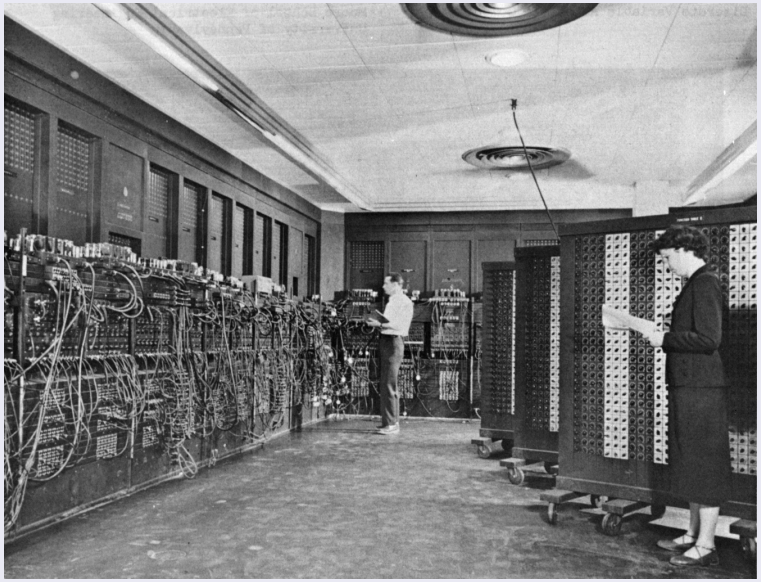
(ローレンツ方程式)

ほとんどの微分方程式は解けない!

コンピュータの出現

世界最初のコンピュータ ENIAC

- 1946 年 2 月、アメリカ
- ミサイルの弾道計算のために開発された



数値計算

- 解けない方程式をコンピュータを使って数値的に解く。
- コンピュータはプログラムに書かれた通りにしか動かないが、とても高速で人間には不可能な大量の計算ができる。

連立一次方程式

ガウスの消去法

- 連立一次方程式をコンピュータで組織的に解く方法
- 普通のパソコンで 10000 変数、スパコンなら 100 万変数でも解ける。

$$\begin{aligned} 2x &- y &+ z &= -1 \\ 3x &+ y &+ z &= 4 \\ x &+ y &+ z &= 2 \end{aligned}$$

$$\Downarrow$$

$$\begin{aligned} 2x &- y &+ z &= -1 \\ &\tfrac{5}{2}y &- \tfrac{1}{2}z &= \tfrac{11}{2} \\ &\tfrac{3}{2}y &+ \tfrac{1}{2}z &= \tfrac{5}{2} \end{aligned}$$

$$\Downarrow$$

$$\begin{aligned} 2x &- y &+ z &= -1 \\ &\tfrac{5}{2}y &- \tfrac{1}{2}z &= \tfrac{11}{2} \\ & &\tfrac{4}{5}z &= -\tfrac{4}{5} \end{aligned}$$

$$\Downarrow$$

$$\begin{aligned} x &= 1 \\ y &= 2 \\ z &= -1 \end{aligned}$$

ガウスの消去法のプログラム

```
void gauss(double **a, double *b, int n)
{
    int i, j, k;
    double tmp;

    for (k=0; k<n-1; k++) {
        for (i=k+1; i<n; i++) {
            tmp = a[i][k] / a[k][k];
            for (j=k; j<n; j++) {
                a[i][j] -= a[k][j] * tmp;
            }
            b[i] -= b[k] * tmp;
        }
    }

    for (i = n-1; i>=0; i--) {
        for (j = i+1; j<n; j++) {
            b[i] -= a[i][j] * b[j];
        }
        b[i] /= a[i][i];
    }
}
```

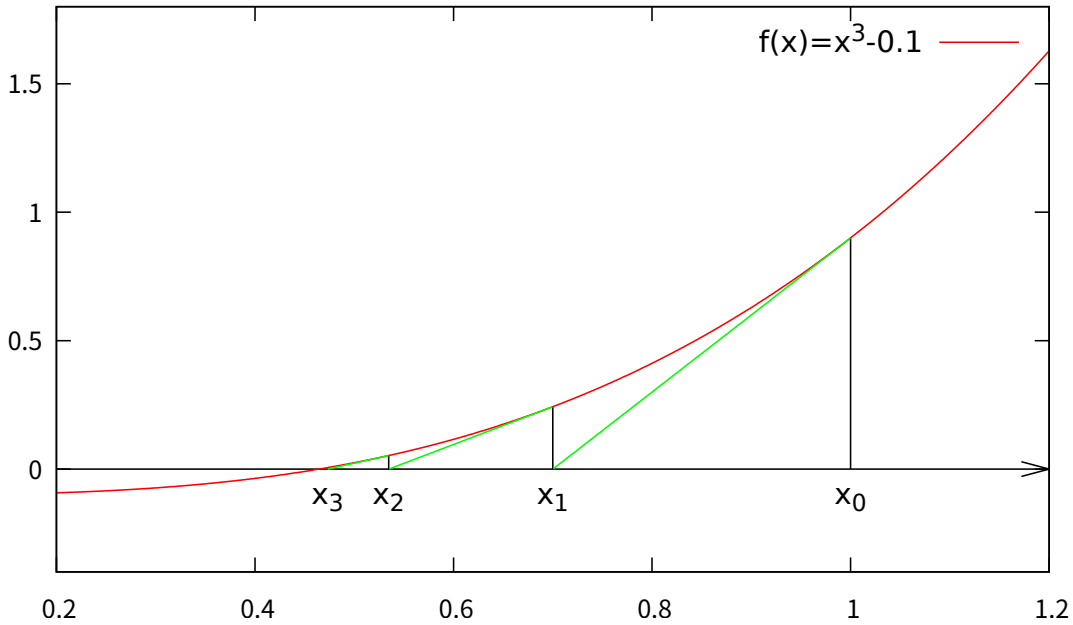
ニュートン法

ニュートン法

非線形方程式 $f(x) = 0$ に対して、x_0 を適当な数として

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

のような数列を作って解に近づけていく。



ニュートン法の例

例

$\cos(x) - x = 0$ に対して、

$$x_{n+1} = x_n - \frac{\cos(x_n) - x_n}{-\sin(x_n) - 1}$$

を繰り返す。

```
#include <stdio.h>
#include <math.h>

int main()
{
    double x;
    int i;

    x = 1.;
    for (i=0; i<=10; i++) {
        printf("x%d: %.15g\n", i, x);
        x = x - (cos(x) - x) / (-sin(x)-1);
    }
}
```

```
x0: 1
x1: 0.750363867840244
x2: 0.739112890911362
x3: 0.739085133385284
x4: 0.739085133215161
x5: 0.739085133215161
x6: 0.739085133215161
x7: 0.739085133215161
x8: 0.739085133215161
x9: 0.739085133215161
x10: 0.739085133215161
```

5 次以上の多項式

デュランカーナー法

1 変数 n 次多項式

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0$$

の n 個全ての解を同時に求める。
$x_1^{(0)}, \cdots, x_n^{(0)}$ を n 個の近似解として、

$$x_k^{(m+1)} = x_k^{(m)} - \frac{f(x_k^{(m)})}{a_n \prod_{j \neq k} (x_k^{(m)} - x_j^{(m)})}$$

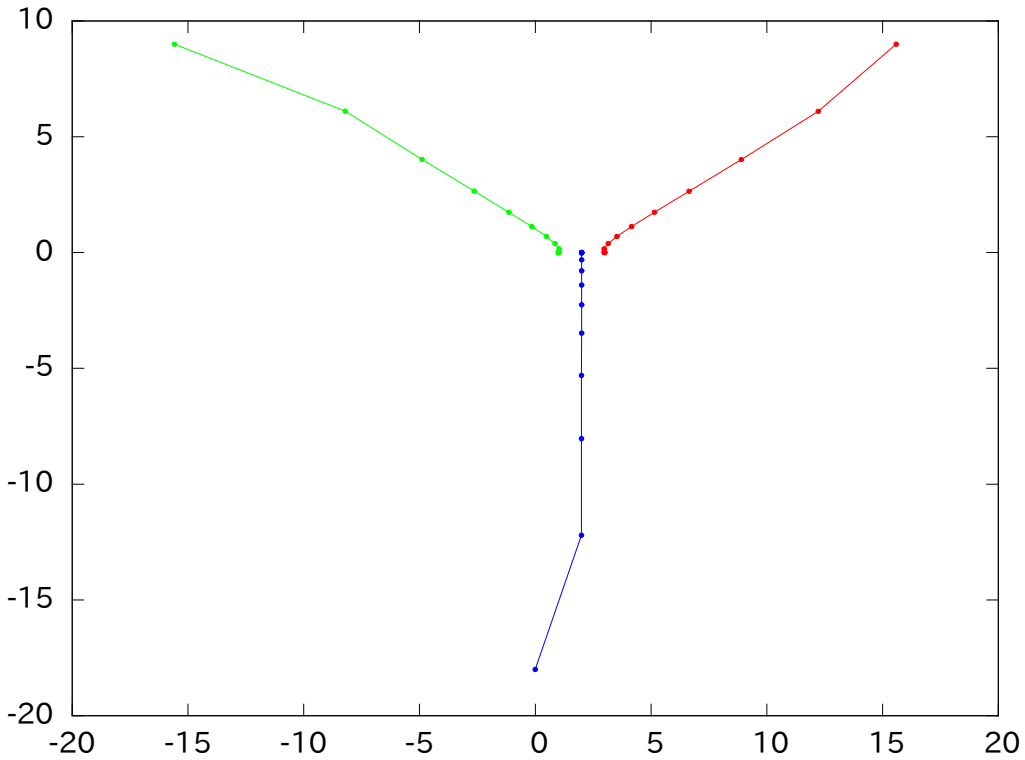
のような反復を行う。

デュランカーナー法の計算例

デュランカーナー法の計算例

$$x^3 - 6x^2 + 11x - 6 = 0$$

の 3 つの解 $x = 1, 2, 3$ を複素平面内で同時に求めている。

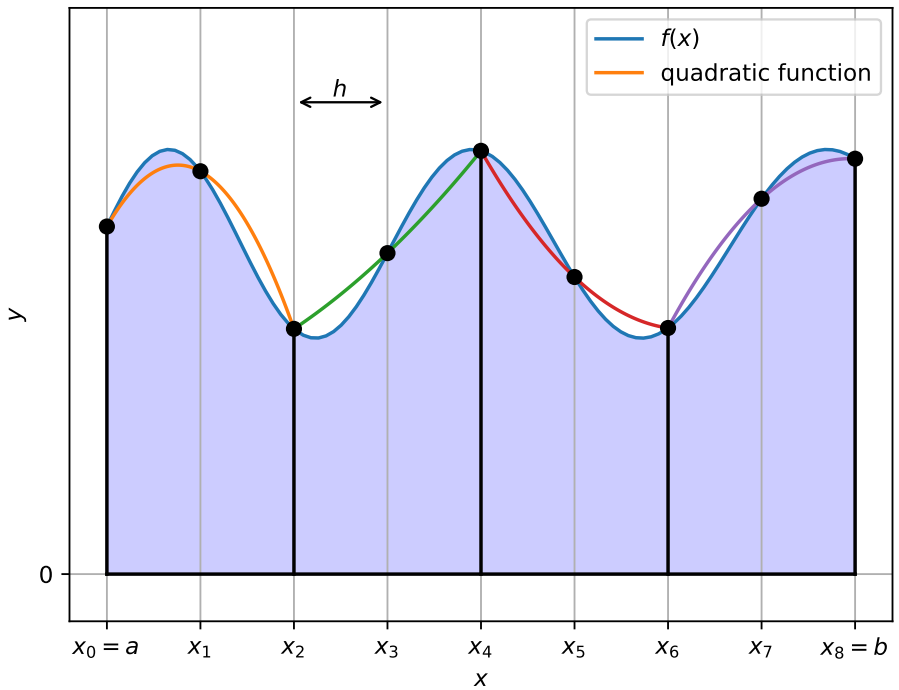


数値積分

シンプソンの公式

n を偶数、$h = (b-a)/n$、$x_k = x_0 + kh$ として

$$\int_a^b f(x)dx \simeq \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots 4f(x_{n-1}) + f(x_n)]$$



本当の面積の代わりに、3 点を通る 2 次関数で囲まれた領域の面積を求めている。

微分方程式の数値計算

オイラー法

微分方程式

$$\begin{aligned} \frac{d}{dt}x(t) &= f(x(t),t) \\ x(t_0) &= x_0 \end{aligned}$$

に対して、Δt を刻み幅、$t_n = t_0 + n\Delta t$ として

$$x(t_{n+1}) = x(t_n) + f(x(t_n), t_n)\Delta t$$

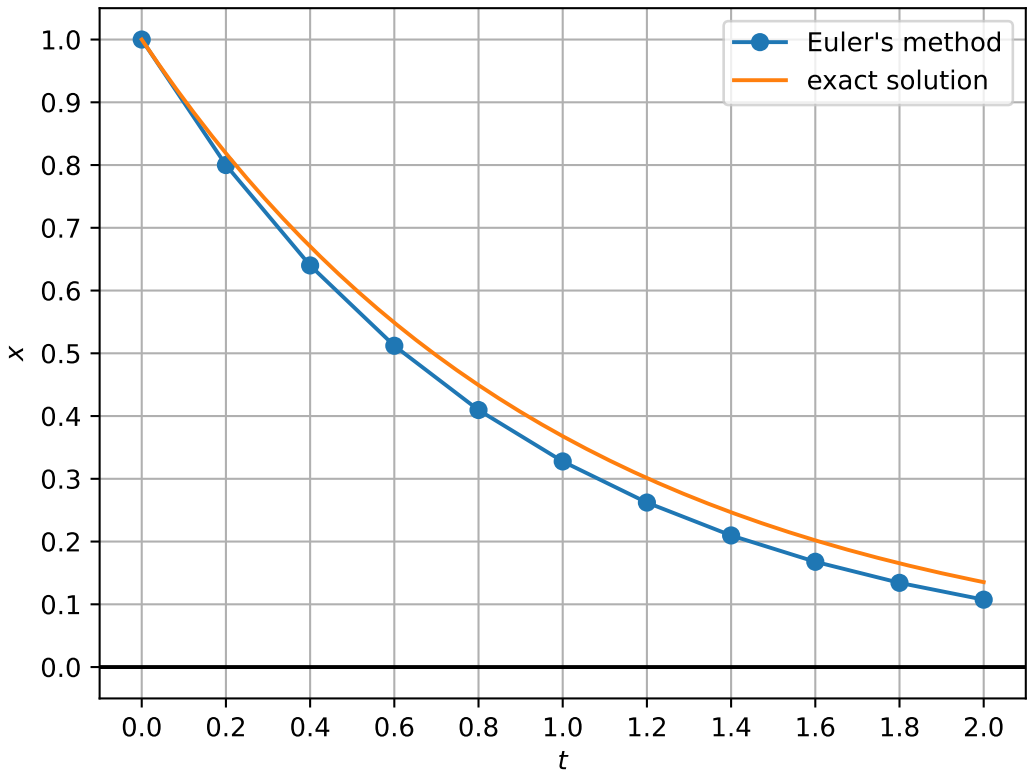
のように飛び飛びの時刻に対する値を順に計算していく。

(このように連続の問題を離散の問題に置き換えることを離散化という)

オイラー法の例

方程式 (厳密解は $x(t) = \exp(-t)$)

$$\frac{dx}{dt} = -x, \quad x(0) = 1, \quad t \in [0, 2], \quad \Delta t = 0.2$$



数値解析

代数的に解を得ることが不可能な解析学上の問題を、数値を用いて近似的に解く。

自然現象

$\Downarrow$ モデル化

微分方程式

$\Downarrow$ 離散化　　(離散化誤差が混入)

(有限次元) 方程式

$\Downarrow$ 数値計算　　(丸め誤差, 打ち切り誤差が混入)

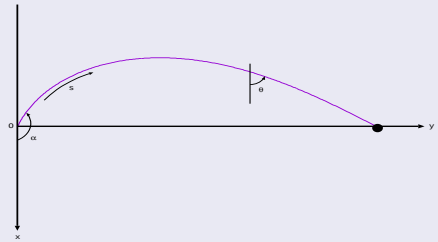
数値解

数値解析の誤差の例

自然現象 (釣り竿の撓み)

⇓ モデル化

微分方程式 $\frac{d^2\theta}{ds^2} = K\sin\theta, \quad \theta(0) = \alpha, \quad \frac{d\theta}{ds}(1) = 0$



⇓ 離散化 (離散化誤差が混入)

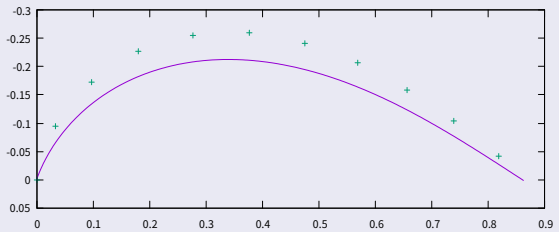
(有限次元) 方程式

$$\theta_{k+1} = \theta_k + \phi_k h \qquad \theta_0 = \alpha$$
$$\phi_{k+1} = \phi_k + K\sin\theta_k h \qquad \phi_n = 0$$

⇓ 数値計算 (丸め誤差, 打ち切り誤差が混入)

数値解

k	θ	ϕ
0	2.7925268031909272	-3.272746776503749
1	2.4652521255405522	-3.1359387191734815
⋮		
10	0.88043670714284783	1.6653345369377348e-16



コンピュータの数値表現

- 電気 (電荷) で記憶するため、0(電荷なし) と 1(電荷あり) の組み合わせで全てのデータを表現する。
- 基本的に 2 進法を使う。

整数の表現

- 0 または 1 を記憶できるメモリを 32 個束ねて使う。(32bit)
- メモリ $b_0 \sim b_{31}$ と実際の数値との対応は、

$$x = -b_{31}2^{31} + \sum_{i=0}^{30} b_i 2^i$$

の通り。下位から順に $1, 2, 4, 8, \ldots$ の重みを持つ (2 進法の基本どおり) が、最上位 bit だけ -2^{31} と負の重みを持たせている。

整数の表現

整数の対応表

b_{31}	b_{30}	b_{29}	$\cdots$	b_2	b_1	b_0	x
0	1	1	$\cdots$	1	1	1	$2147483647 = 2^{31} - 1$
				$\vdots$			
0	0	0	$\cdots$	0	1	0	2
0	0	0	$\cdots$	0	0	1	1
0	0	0	$\cdots$	0	0	0	0
1	1	1	$\cdots$	1	1	1	-1
1	1	1	$\cdots$	1	1	0	-2
1	1	1	$\cdots$	1	0	1	-3
				$\vdots$			
1	0	0	$\cdots$	0	0	0	$-2147483648 = -2^{31}$

(表現できる数の大きさに限界がある)

計算機の実数の表現 (IEEE 754 倍精度浮動小数点数)

実数の表現

- 6.02×10^{23} のような、「1 に近い数 $\times$ ベキ数」の形で記憶する。浮動小数点形式という。
- 2 進数で記憶し、64bit=8byte を使う。
- 64bit を以下のように分割し、

s	e_{10}	e_9	$\cdots$	e_0	m_{51}	m_{50}	$\cdots$	m_0

e を 2 進整数 ($e = \sum_{i=0}^{10} e_i 2^i$)、$m$ を 2 進小数 ($m = \sum_{i=0}^{51} m_i 2^{i-52}$) として

$$(-1)^s \times (1+m) \times 2^{e-1023}$$

で計算される数値と対応する。

- 計算精度 (有効数字) は、$2^{-52} \simeq 2.22 \times 10^{-16} \simeq 10^{-15.65}$ なのでおよそ 16 桁。

丸め誤差

丸め誤差

- 有効数字がおよそ 16 桁なので、例えば $1 \div 3$ を計算すると

$$1 \div 3 = 0.3333333333333333148\ldots$$

のように少し誤差が入る。これを丸め誤差という。

- 2 進法を使っているので、$1 \div 10 = 0.1$ でも

$$1 \div 10 = 0.10000000000000005551\ldots$$

のように誤差が入る。

- 1 回だけの計算なら大したことないが、誤差が積み重なると想像以上に大きく誤差が拡大することがある。

2 次方程式の丸め誤差

2 次方程式

2 次方程式 $ax^2 + bx + c = 0$ の解の公式は、

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

大きな誤差が見られる例

$$a = 1,\ b = 10^{15},\ c = 10^{14}$$

のときの大きい方の解を計算する。

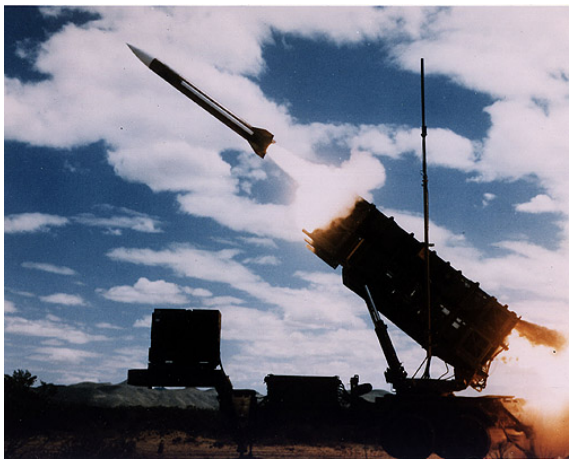
- (解の公式) $\dfrac{-b + \sqrt{b^2 - 4ac}}{2a} = -0.125$
- (解の公式の分子を有理化)
  $\dfrac{2c}{-b - \sqrt{b^2 - 4ac}} = -0.10000000000000002$

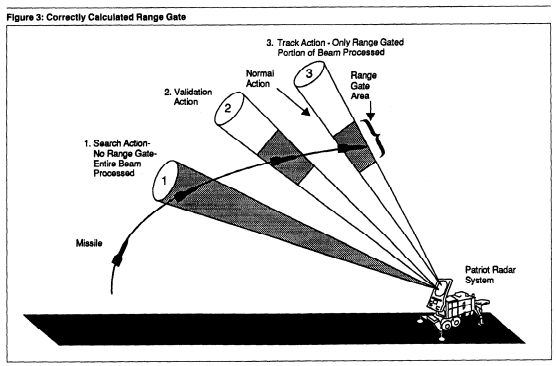
湾岸戦争におけるパトリオットミサイルの事故

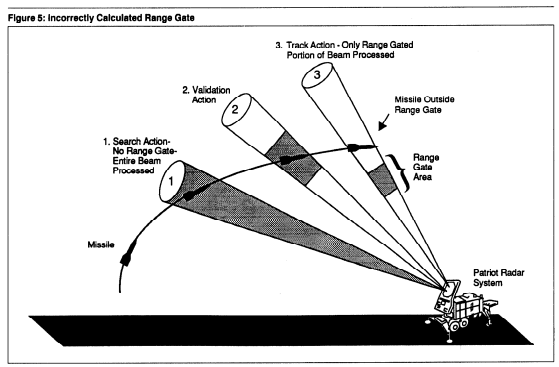
http://www.sydrose.com/case100/298/
https://www.gao.gov/assets/220/215614.pdf

湾岸戦争中、イラク軍のスカッド・ミサイル (Scud Missile) の迎撃のため、アメリカ軍のパトリオット・ミサイル (Patriot Missile) が発射されたが、その内部計算機のわずかな誤差 (0.1 秒を正確に表現できなかったことによる) からスカッド・ミサイルを捉えることが出来ず迎撃は失敗した。

- 発生日時 1991 年 2 月 25 日
- 発生場所 サウジアラビア、Dharan
- 死者 28 名、負傷者約 100 名



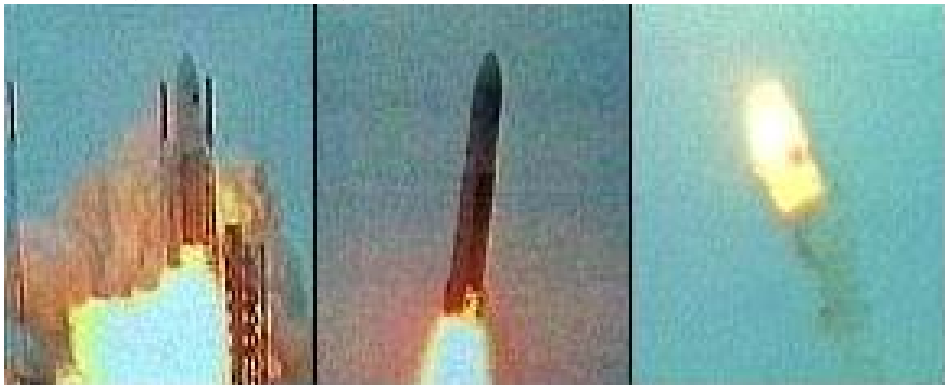




アリアン5型ロケット爆発事故

http://www.sydrose.com/case100/284/

http://www.math.umn.edu/~arnold/disasters/ariane.html



爆発事故

1996 年 6 月 4 日、欧州宇宙機関 (European Space Agency) が新たに開発した無人ロケット、アリアン 5 (Ariane 5) の最初の打ち上げがフランス領ギアナのクルーで行われたが、発射のわずか 40 秒後に進路を大きく逸れ、空中で爆発した。アリアン 5 の開発には 10 年の歳月と 70 億ドルの費用が投じられていた。

原因

慣性システムのソフトウェアに問題があった。ロケットの水平速度に関する数値を 64bit の浮動小数点数から 16bit の符号付き整数に変換する部分で 16bit 符号付き整数の最大値 32767 を超えてしまい変換に失敗、正確なロケットの姿勢データを得ることが出来なくなってしまった。

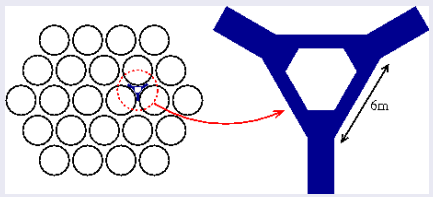
シュライプナー A 海上プラットフォーム沈没事故

http://www.shippai.org/fkd/cf/CA0000299.html
http://www.math.umn.edu/~arnold/disasters/sleipner.html

沈没事故

北海で石油とガスを産出するための海上プラットフォーム Sleipner A は、1991 年 8 月 23 日、ノルウェー沖での建設中に水中の基礎構造物に破損が生じて浸水し、崩壊、沈没した。

原因



崩壊の原因は、基礎構造物の 24 本の柱の間にある、tricell と呼ばれるコンクリートの構造物が、強度の不足により破断したためであった。有限要素法での計算の精度に問題があって圧力が実際の 47%と低く計算され、それが設計の強度不足を招いてしまった。

精度保証付き数値計算

精度保証付き数値計算

近似解だけでなく、その解の数学的に厳密な誤差評価をも計算する数値計算法。

必要な技術

- 区間演算 (切り捨てと切り上げを併用して計算に混入した誤差を把握する。関数の像の評価も行う。)
- 不動点定理 (不動点定理の成立の十分条件を区間演算で確認することによって、方程式の解の存在と存在範囲を保証する)
- 自動微分も重要。

区間演算 (1/5)

区間演算

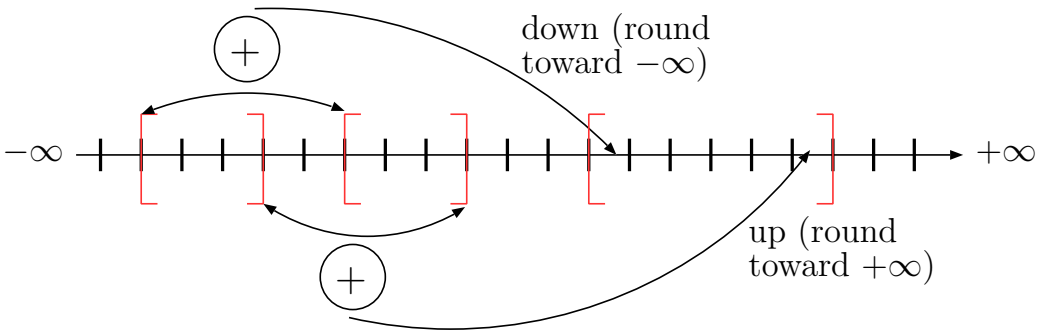
- 精度保証付き数値計算の基本技術。
- 例えば、円周率を $\pi \simeq 3.14$ とするとそれは近似だが、$\pi \in [3.14, 3.15]$ というのは正しい情報。
- 数値を、計算機で表現可能な浮動小数点数を両端に持つ閉区間 $X = [a, b]$ で表現する。
- 区間同士の演算は、集合値演算として計算結果として有り得る値を包含するように行う。
- IEEE754 標準で定義されている「方向付き丸め」を利用する。
- 「丸め誤差の把握」と「関数の値域の評価」の 2 つの役割がある。

区間演算 (2/5)

IEEE754 の丸めモード

- round to nearest: デフォルトの丸めモード。最も近い浮動小数点数に丸める。
- round toward $+\infty$: 上向き丸め
- round toward $-\infty$: 下向き丸め
- round toward 0: 絶対値が大きくならない方向への丸め

区間演算では、下端を下向き丸め、上端を上向き丸めで計算することによって、計算結果が外側に広がるように計算する。



区間演算 (3/5)

- $X = [a, b], Y = [c, d]$
- $\underline{\cdot}, \overline{\cdot}$ はそれぞれ下向き丸め、上向き丸め

- 加算 $X + Y = [a \underline{+} c, b \overline{+} d]$
- 減算 $X - Y = [a \underline{-} d, b \overline{-} c]$
- 乗算 $X \times Y =$

	$d < 0$	$c \le 0, d \ge 0$	$c > 0$
$b < 0$	$[b \underline{\times} d, a \overline{\times} c]$	$[a \underline{\times} d, a \overline{\times} c]$	$[a \underline{\times} d, b \overline{\times} c]$
$a \le 0, b \ge 0$	$[b \underline{\times} c, a \overline{\times} c]$	$[\min(a \underline{\times} d, b \underline{\times} c), \max(a \overline{\times} c, b \overline{\times} d)]$	$[a \underline{\times} d, b \overline{\times} d]$
$a > 0$	$[b \underline{\times} c, a \overline{\times} d]$	$[b \underline{\times} c, b \overline{\times} d]$	$[a \underline{\times} c, b \overline{\times} d]$

- 除算 $X/Y =$

	$d < 0$	$c > 0$
$b < 0$	$[b \underline{/} c, a \overline{/} d]$	$[a \underline{/} c, b \overline{/} d]$
$a \le 0, b \ge 0$	$[b \underline{/} d, a \overline{/} d]$	$[a \underline{/} c, b \overline{/} c]$
$a > 0$	$[b \underline{/} d, a \overline{/} c]$	$[a \underline{/} d, b \overline{/} c]$

($Y \not\ni 0$ の場合のみ定義される)

- 平方根 $\sqrt{X} = [\underline{\sqrt{a}}, \overline{\sqrt{b}}]$

区間演算 (4/5)

10 進数有効数字 3 桁で $(1 \div 3) \times 3$ を計算すると $\cdots$

- 普通の数値計算

$$1 \div 3 = 0.333$$
$$0.333 \times 3 = 0.999$$

四捨五入による誤差がでる。

- 区間演算

最初は幅のない区間からスタート。

$$[1, 1] \div [3, 3] = [0.333, 0.334]$$
$$[0.333, 0.334] \times [3, 3] = [0.999, 1.01]$$

常に区間内に真の値を含みながら計算が進む。

2 次方程式の例を区間演算で計算する

2 次方程式

2 次方程式 $ax^2 + bx + c = 0$ の解の公式は、

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

大きな誤差が見られる例

$$a = 1,\ b = 10^{15},\ c = 10^{14}$$

のときの大きい方の解を計算する。

- (解の公式) $\dfrac{-b + \sqrt{b^2 - 4ac}}{2a} = [-0.1875, -0.0625]$
- (解の公式の分子を有理化) $\dfrac{2c}{-b - \sqrt{b^2 - 4ac}} =$
  $[-0.10000000000000004, -0.09999999999999991]$

区間演算 (5/5)

丸めの向きの変更方法

- IEEE754 で丸めの向きを変更可能にすることが要請されているが、その方法は CPU やコンパイラによって違う。
- X86 だと FPU と SSE2 の 2 種類の演算器で丸めの向きの変更方法が異なるなど複雑。
- 最近は C99 準拠のコンパイラが増えたので、`fenv.h` と `fesetround` を使えばハードウェアの違いを吸収してくれる。

```
#include <iostream>
#include <fenv.h>
int main()
{
    double x=1, y=10, z;
    std::cout.precision(17);

    fesetround(FE_TONEAREST);
    z = x / y;
    std::cout << z << std::endl;
    fesetround(FE_DOWNWARD);
    z = x / y;
    std::cout << z << std::endl;
    fesetround(FE_UPWARD);
    z = x / y;
    std::cout << z << std::endl;
}
```

```
0.10000000000000001
0.099999999999999991
0.10000000000000001
```

kv ライブラリ

- http://verifiedby.me/kv/ で公開中。
- 作成開始は 2007 年秋頃。公開開始は 2013 年 9 月 18 日。最新版は version 0.4.55。
- 言語は C++。boost C++ Libraries (http://www.boost.org/) も必要。
- 全てヘッダファイルで記述されており、インストールはヘッダファイルをどこかに配置するだけ。
- オープンソースである。精度保証付き数値計算の結果が「証明」であると主張するならば、計算に使われたプログラムは必ず公開されているべき。
- 計算に使う数値の型は double に制限されていない。C++のテンプレート機能を用いて容易に変更することが出来る。

2021年ノーベル物理学賞

THE NOBEL PRIZE

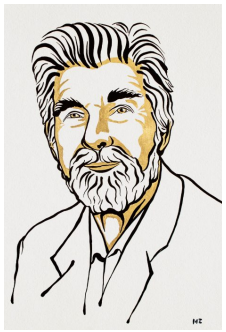
The Nobel Prize in Physics 2021 | Explore

The Nobel Prize in Physics 2021

Ill. Niklas Elmehed © Nobel Prize Outreach

Syukuro Manabe

Prize share: 1/4



Ill. Niklas Elmehed © Nobel Prize Outreach

Klaus Hasselmann

Prize share: 1/4

Ill. Niklas Elmehed © Nobel Prize Outreach

Giorgio Parisi

Prize share: 1/2

KUNGL. VETENSKAPS-AKADEMIEN
THE ROYAL SWEDISH ACADEMY OF SCIENCES

5 OCTOBER 2021

Scientific Background on the Nobel Prize in Physics 2021

"FOR GROUNDBREAKING CONTRIBUTIONS TO OUR UNDERSTANDING OF COMPLEX PHYSICAL SYSTEMS"

The Nobel Committee for Physics

THE ROYAL SWEDISH ACADEMY OF SCIENCES has as its aim to promote the sciences and strengthen their influence in society.

BOX 50005 (LILLA FRESCATIVÄGEN 4 A), SE-104 05 STOCKHOLM, SWEDEN
TEL +46 8 673 95 00, KVA@KVA.SE • WWW.KVA.SE

Scientific Backgroud 文書は · · ·

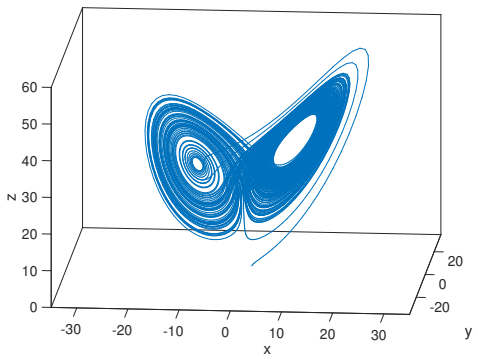
複雑系の元祖として、Lorenz 方程式の初期値鋭敏性の説明から始まる。

upper and lower boundaries [97]. The model is

$$\frac{dX}{dt} = \sigma(Y - X),$$
$$\frac{dY}{dt} = X(Ra - Z) - Y \qquad \text{and}$$
$$\frac{dZ}{dt} = XY - \beta Z,$$

where X describes the intensity of convective motion, Y is the temperature difference between ascending and descending flow and Z is the deviation from linearity of the vertical temperature profile. The control parameters are the Prandtl Number, σ, which is a property of the fluid, the Rayleigh Number, Ra, which is the dimensionless buoyancy driving vertical fluid motions, and a constant factor β, characterizing the domain geometry.

The Lorenz system acts as a rich toy model of low-dimensional chaos. Since its origin the breadth and extension of studies has been so broad [e.g., 103] it would be difficult to enumerate them all. Key here are the facts that the solutions are bounded, (Fig. 1) and yet exhibit *sensitive dependence on initial conditions* (Fig. 2).



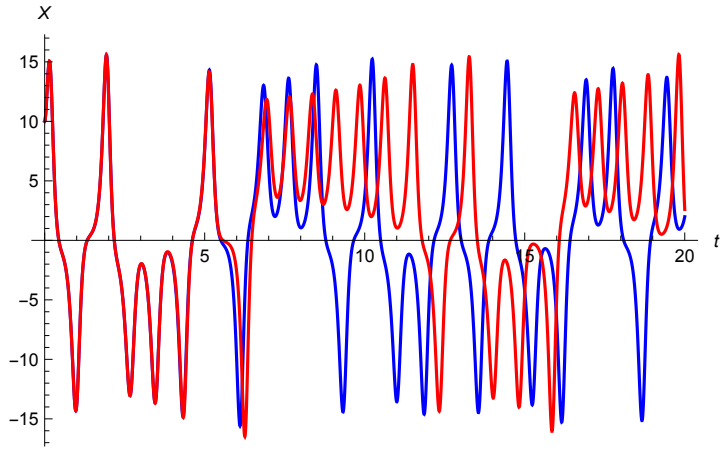
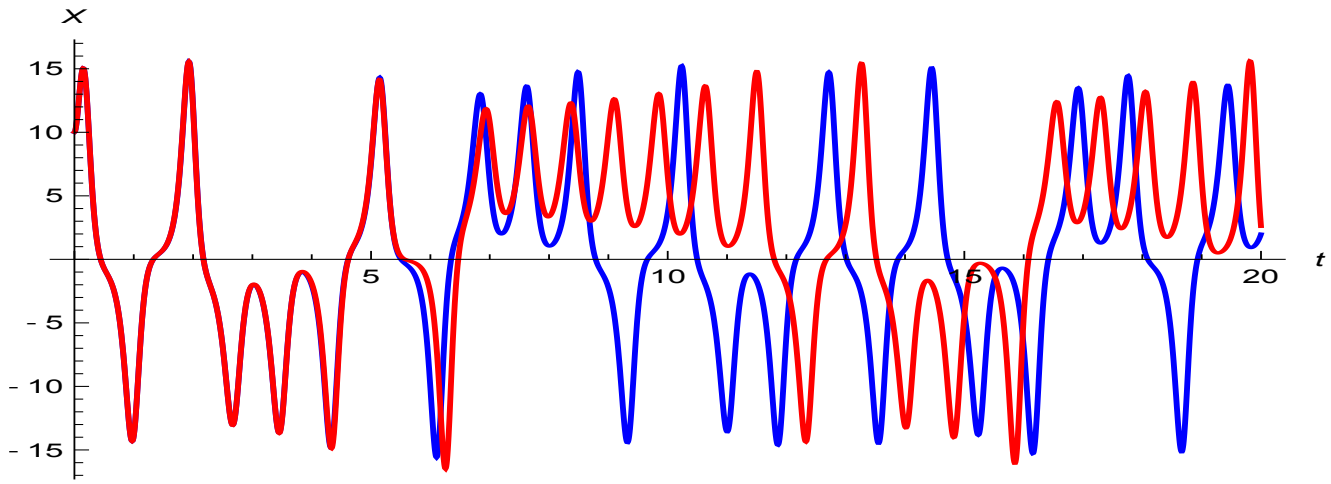


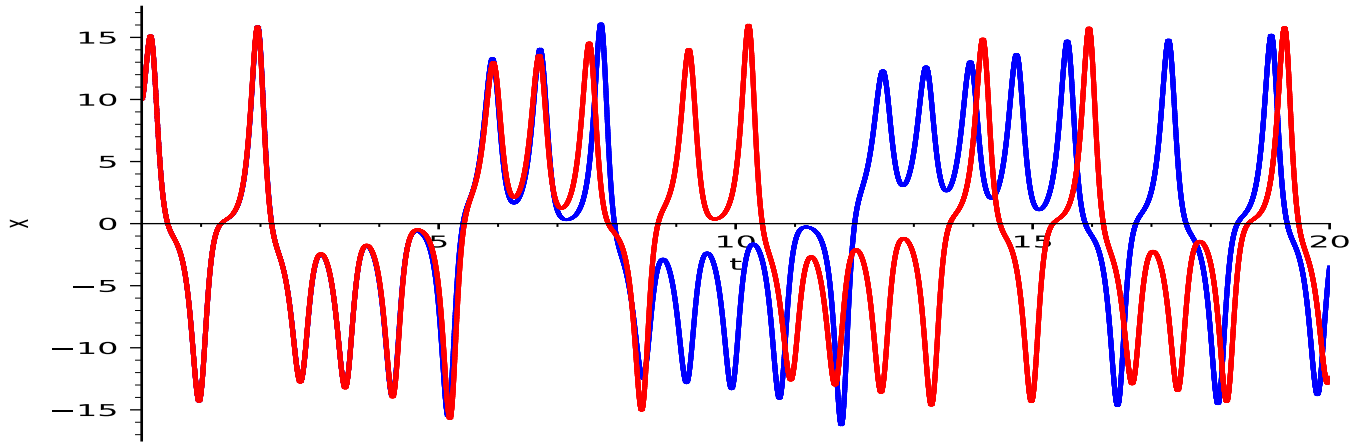
FIG. 2. Plot of $X(t)$ of the Lorenz system with $(\sigma, \beta, Ra) = (10, 8/3, 24.9)$ in which the initial data for all three variables are 10 (blue) or 10.01 (red). The divergence of the two solutions with slightly different initial conditions begins at $t = 5.5$; this is sensitive dependence on initial conditions, often whimsically referred to as the "Butterfly Effect".

初期値 10 の場合の青いグラフと、初期値 10.01 の場合の赤いグラフで途中から全然違った軌道になる。いわゆる Buttefly Effect。

試しに精度保証付きで再計算してみたら・・・



(FIG.2 of "Scientific background on the Nobel Prize in physics 2021")



(Verified Solution of Lorenz Equation calculated by kv library)

初期値の違いによる誤差よりも数値計算の誤差の方がずっと大きい！

精度保証付き数値積分の例

$$\int_0^{10} \frac{\sin(x)}{\cos(x^2)+1+2^{-10}}dx$$

kv-0.4.41	[38.38352626453522 7,38.38352626464969]
intlab 9	[38.34845927756175, 38.41859325162576]
octave 3.8.1	38.3837105761501
Mathematica 10.1.0	0.0608979
matlab 2007b	38.383519835854528
keisan (Romberg)	38.324147930794
keisan (Tanh-Sinh)	38.248589488377546779 84
keisan (Gauss-Legendre)	116.44815670772585127 3
intde2 by ooura	32.4641
python + scipy	36.48985372847387
CASIO fx-5800P	38.38352669