N体シミュレーションの基礎

N体シミュレーション大寒の学校

2020年1月22日

数値計算の基礎

計算機による数値の表現

整数の表現

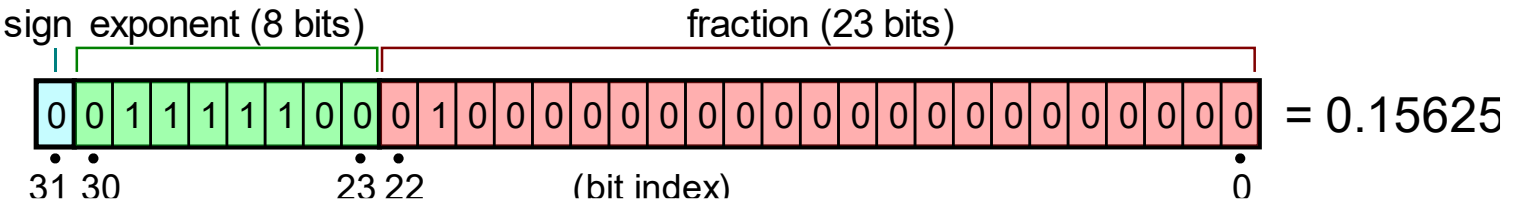
- 計算機は数値を２進数で表す
- 桁数の単位はbit
- 8桁の2進数で整数を表現したら8bit整数

10進数の13

- 8bitなら[0,255]（16進数2桁）
- 符号が付くと1bit減るので[-127,128]
- 32bitが標準，long型だと64bit
- 8bitで1byte（データ量の単位）

小数点の表現

- 浮動小数点（Floating Point）
- 2進数⇔10進数の変換において、小数点以下は有限桁数で正確に変換できない
- 指数部と仮数部に分けて表現される



https://commons.wikimedia.org/w/index.php?curid=3357169

- 単精度(32bit, FP32), 倍精度(64bit, FP64)
- 必要とする精度は問題毎に異なる

単精度実数と倍精度実数

	指数部	指数部	仮数部
単精度	1bit	8bit	23bit
倍精度	1bit	11bit	52bit

実数=(符号部)×$2^{(指数部)}$×(1+仮数部)

実数の範囲

	有効桁数	下限値	上限値
単精度	~7	1.175494×10^{-38}	3.402823×10^{38}
倍精度	~15	$2.225074 \times 10^{-308}$	1.797693×10^{308}

- 単精度は10進数7桁程度
- 倍精度は10進数16桁程度

数値計算の基礎

誤差

誤差

・丸め誤差

仮数部が有限であるので途中の桁で四捨五入(2進数では0捨1入)するため生じる誤差

・桁落ち

ほぼ等しい数どうしの減算を行ったときに有効桁数が下がること

例) 0.1234×10^{2}(4桁) - 0.1233×10^{2}(4桁)=0.1×10^{-2}(1桁)

・情報落ち

大きな数と小さな数の加減算を行うとき、小さな数からみた誤差が大きくなっている

例) $0.1234\times10^{2} + 0.1233\times10^{-1}=0.1235\times10^{2}$

常微分方程式

物理や工学の問題の多くは微分方程式で記述される

例) 天体の軌道、電気回路

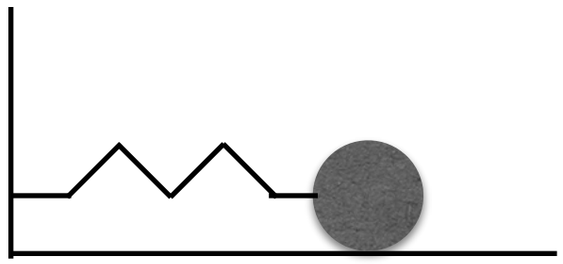
微分方程式を解析的に解く事は多くの場合不可能

数値計算で近似値を求める

数値計算を行う際は、求めたい近似値の精度に気をつける

常微分方程式

例)単振動



おもりに働く復元力
フックの法則

$$f = -kx$$

おもりの運動方程式

$$m\frac{d^2x}{dt^2} = -kx$$

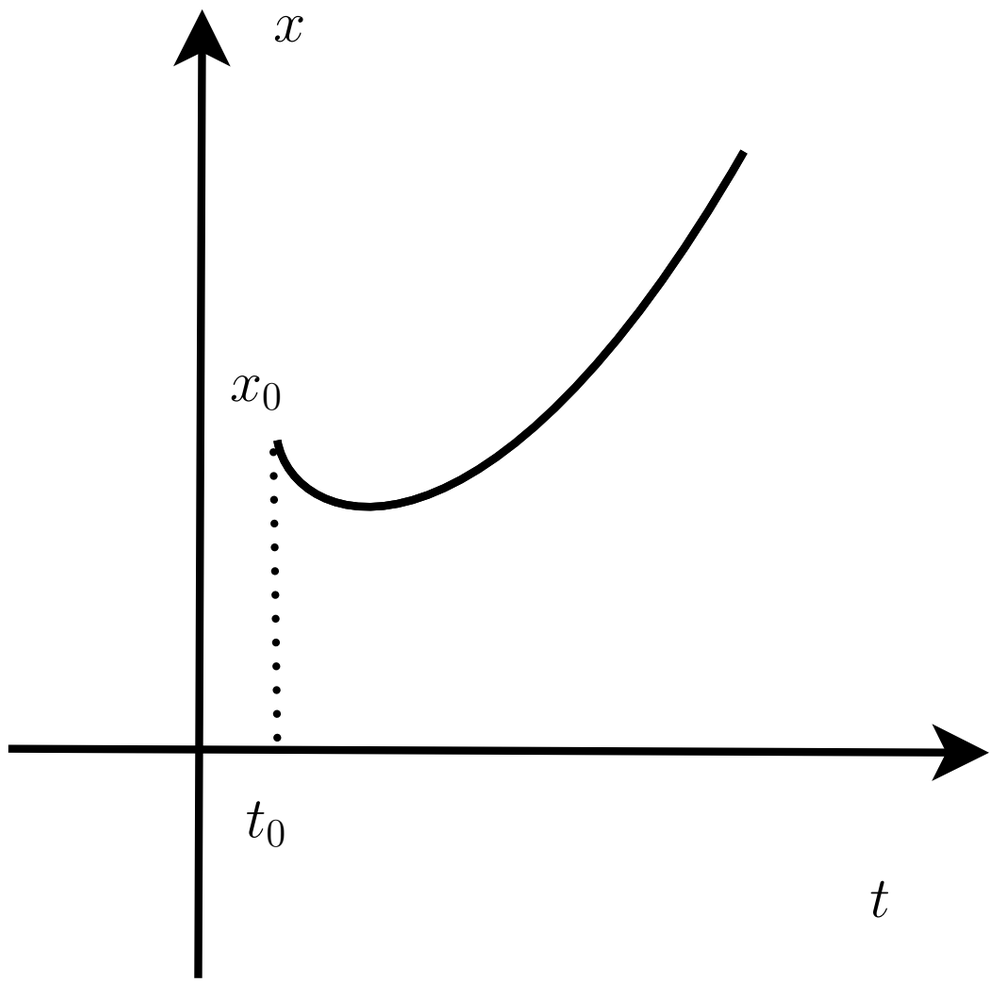
時刻tでのおもりの位置は以下のように表せる

$$x(t) = A\cos\left(\sqrt{k/m}t\right) + B\sin\left(\sqrt{k/m}t\right)$$

常微分方程式の数値解法

一階常微分方程式：　$\dfrac{dx}{dt} = f(x,t)$

初期値：　$x(t_0) = x_0$

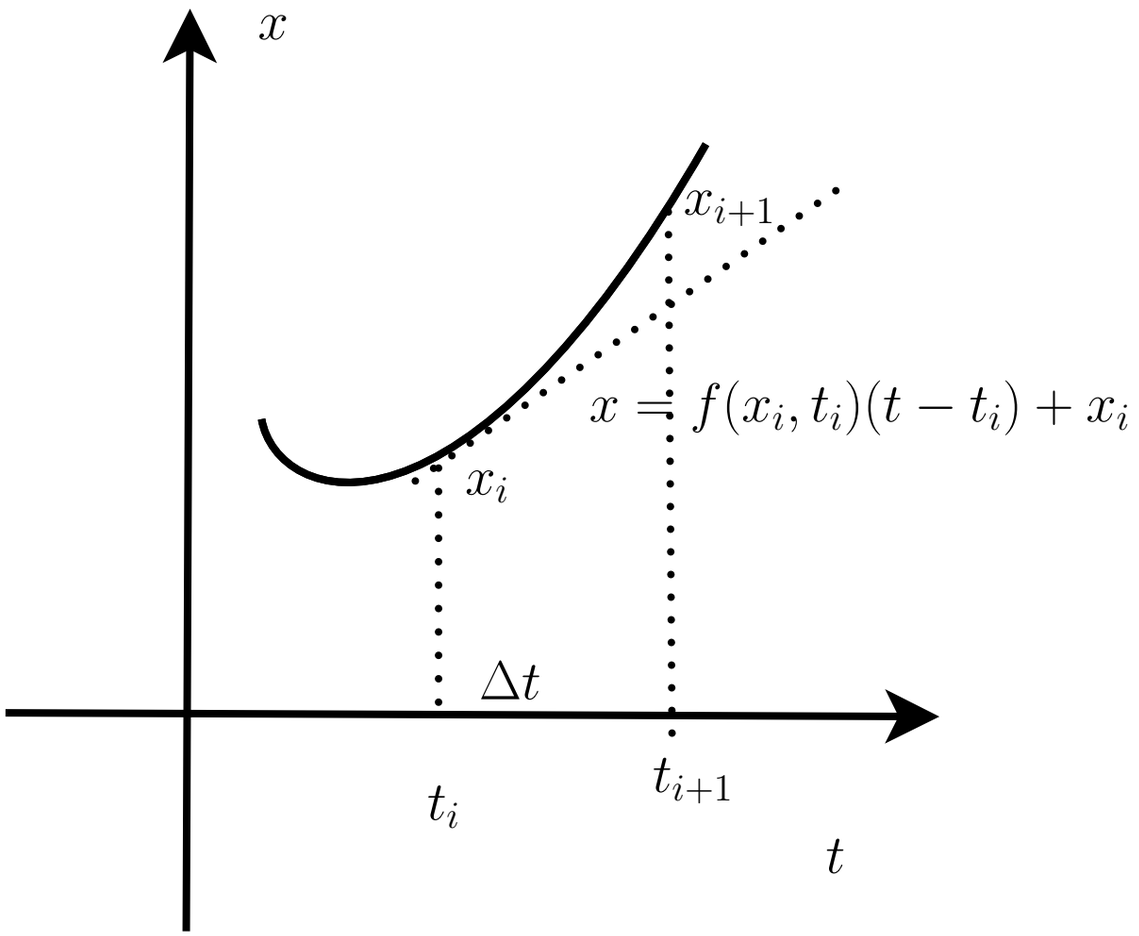


時刻 t_0 における x の値が
x_0 として (初期条件)、
任意の時刻 t における x を求める。

オイラー法

計算法

$$x_{n+1} = x_n + \Delta t f(x_n, t_n)$$

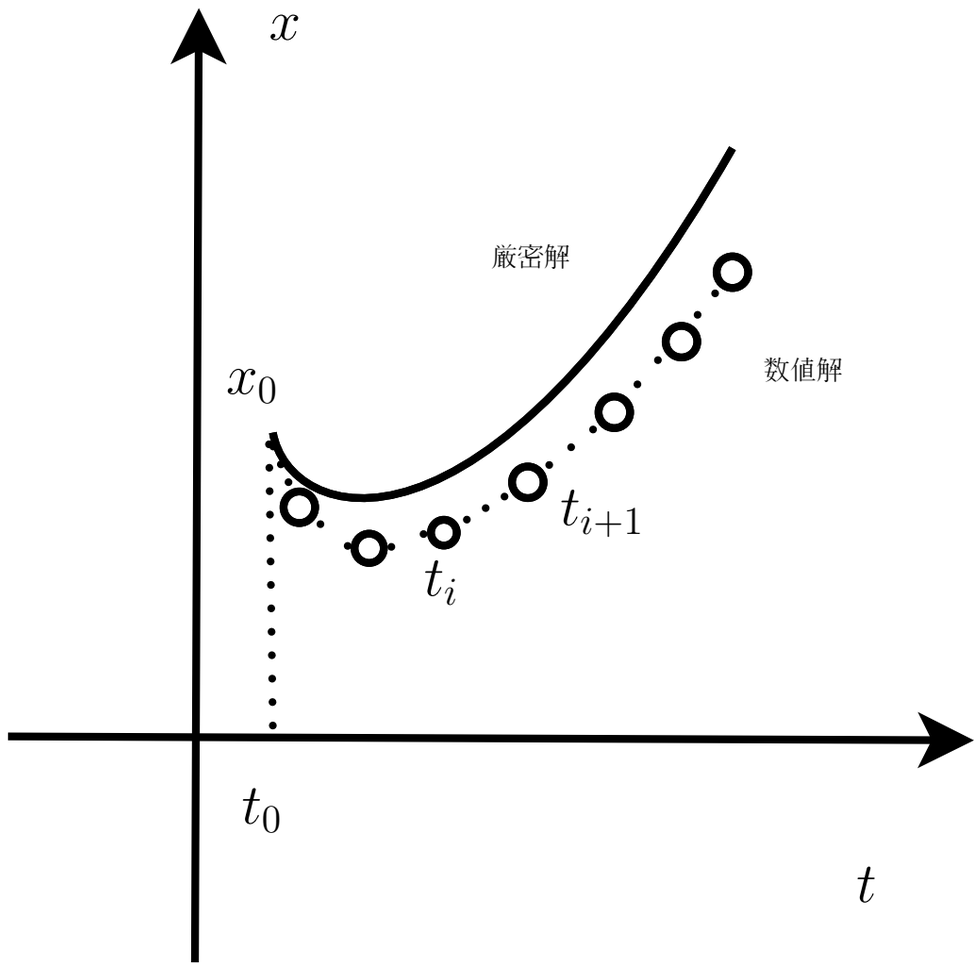


- 数値計算を行う際は常微分方程式の dt を有限の Δt とし、
Δt ずつ補正項 $\Delta t f(x_n,t_n)$ を計算
- Δtのことをタイムステップ、もしくは時間刻みとよぶ
- Δt が極小のとき
元の微分方程式に良く近似する

オイラー法

計算法

$$x_{n+1} = x_n + \Delta t f(x_n, t_n)$$



- 数値計算を行う際は常微分方程式の dt を有限の Δt とし、
Δt ずつ補正項 $\Delta t f(x_n,t_n)$ を計算
- Δtのことをタイムステップ、
もしくは時間刻みとよぶ
- Δt が極小のとき
元の微分方程式に良く近似する

オイラー法

$$\frac{dx}{dt} = f(x,t)$$

xをt_nの周りでテイラー展開する

$$x_{n+1} = x(t_n + \Delta t) = x(t_n) + \frac{dx}{dt}\Delta t + \frac{1}{2}\frac{d^2x}{dt^2}\Delta t^2 + \ldots$$

この式を微分方程式を用いて書き直すと

$$x_{n+1} = x(t_n) + \Delta t f(x,t) + O(\Delta t^2)$$

オイラー法

微分方程式

$$\frac{dx}{dt} = f(x,t)$$

$$x(t_0) = x_0$$

近似

差分方程式

$$\frac{x_{n+1} - x_n}{\Delta t} = f(x_n, t_n)$$

$$t_{n+1} = t_n + \Delta t$$

$$x(t_0) = x_0$$

プログラムの構造

① 初期値の設定

変数に初期値を入力する

$x_0=1.0$
$t_0=0.0$

② 刻み幅の計算

今回はdt=0.1を使用する

逆に数値計算を行う範囲をN分割したい場合は
以下のように刻み幅を決定する

$$dt = \frac{1}{N}$$

プログラムの構造

③　条件判定

現在の時間が終了時刻になっているか判定する

④　時刻t+Δtでの値を計算

$$x_{n+1} = x_n + f(x,t)\Delta t$$

$$t_{n+1} = t_n + \Delta t$$

新しい値を用いて次の時刻の計算を行う

プログラムの構造

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

int main(void)
{
  double t, x;
  const double dt = 0.1;
  const double t_end = 10.0;

  t = 0.0;
  x = 1.0;

  printf("%f, %f\n", t, x);

  while (t < t_end) {
    t += dt;
    x += x*dt;
    printf("%f, %f\n", t, x);
  }
  return 0;
}
```

オイラー法のプログラム例

ループの中の計算を
関数化しても良い

オイラー法の誤差

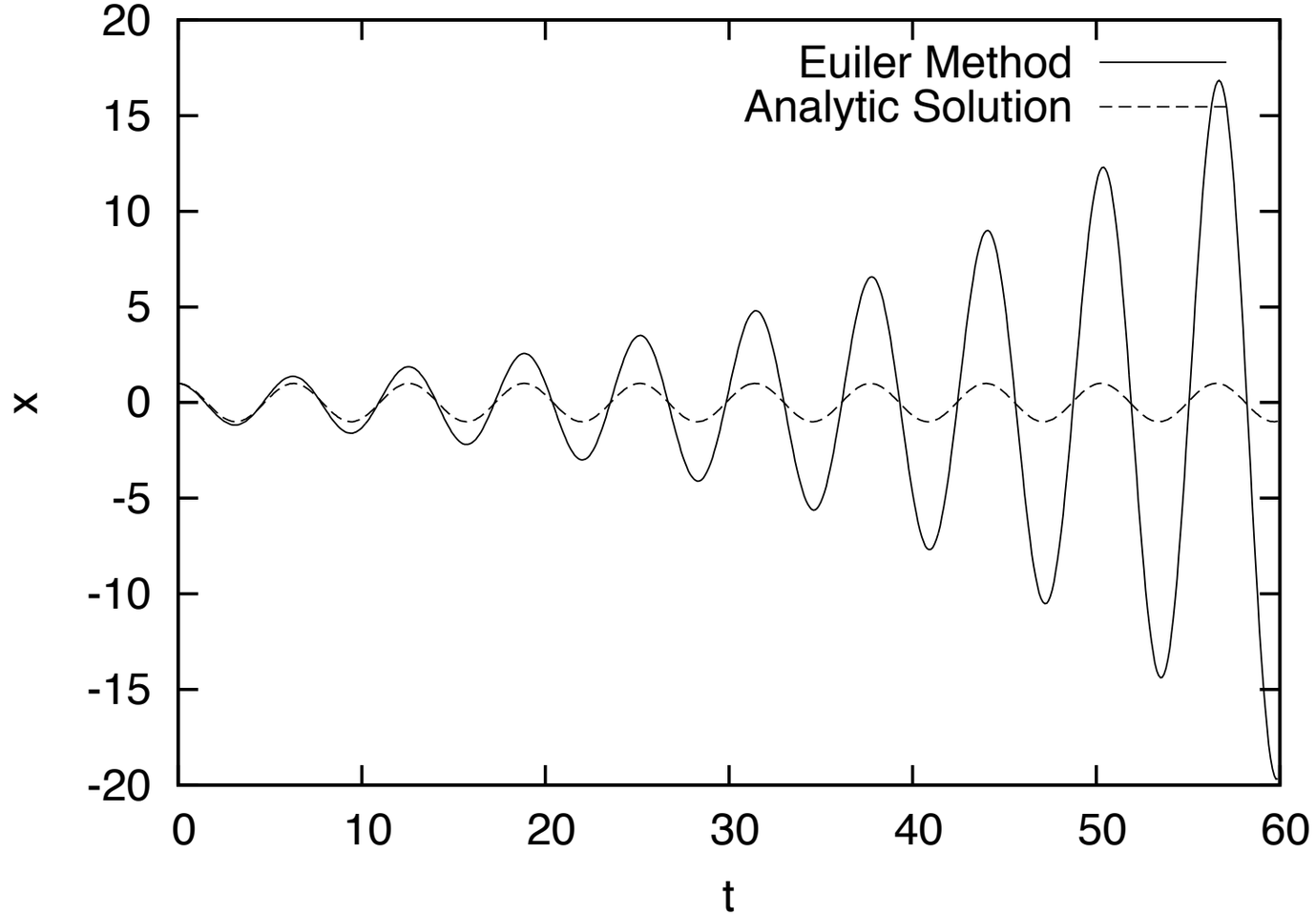
オイラー法

$$\frac{d^2x}{dt^2} = -x$$

$$v_0 = 0, x_0 = 1$$

厳密解

$$x = \cos t$$



オイラー法の誤差

- オイラー法は接線近似を仮定している
  → タイムステップを短くするほど精度が良くなる

誤差の定量的評価

$x(t_{i+1})$ を厳密解、x_{i+1} を数値解とする。このとき

誤差 $= x(t_{i+1}) - x_{i+1} = O(\Delta t^{m+1}) =$ 定数 $\times \Delta t^{m+1}$

とかけると、この数値計算法は m 次であるという

オイラー法の誤差

誤差の定量的評価

$x(t_{i+1})$ を厳密解、x_{i+1} を数値解とする。このとき

誤差 $= x(t_{i+1}) - x_{i+1} = O(\Delta t^{m+1}) =$ 定数 $\times \Delta t^{m+1}$

とかけると、この数値計算法は**m次**であるという

オイラー法は**1次**の解法

$$x_{n+1} = x(t_n) + \Delta t f(x,t) + O(\Delta t^2)$$

dtを半分にすると誤差もおよそ半分になる

オイラー法の誤差

オイラー法の1ステップ当たりの誤差は$O(\Delta t^2)$

$$x_{n+1} = x(t_n) + \Delta t f(x,t) + O(\Delta t^2)$$

一方、計算時間Tの間に行う逐次計算の回数は

$$N = T/\Delta t$$

つまり、計算全体では逐次計算の誤差が
積み上がっていく事になるので

$$O(\Delta t^2) \times N \sim O(\Delta t^2) \times O(\Delta t^{-1}) \sim O(\Delta t)$$

局所誤差と大域誤差

局所誤差

タイムステップ Δt の間に蓄積する誤差

厳密解: $x(t_{i+1})$ と数値解:x_{i+1} のとき

m次の局所誤差は $x(t_{i+1})=x_{i+1}+O(\Delta t^{m+1})$ の関係を満たす

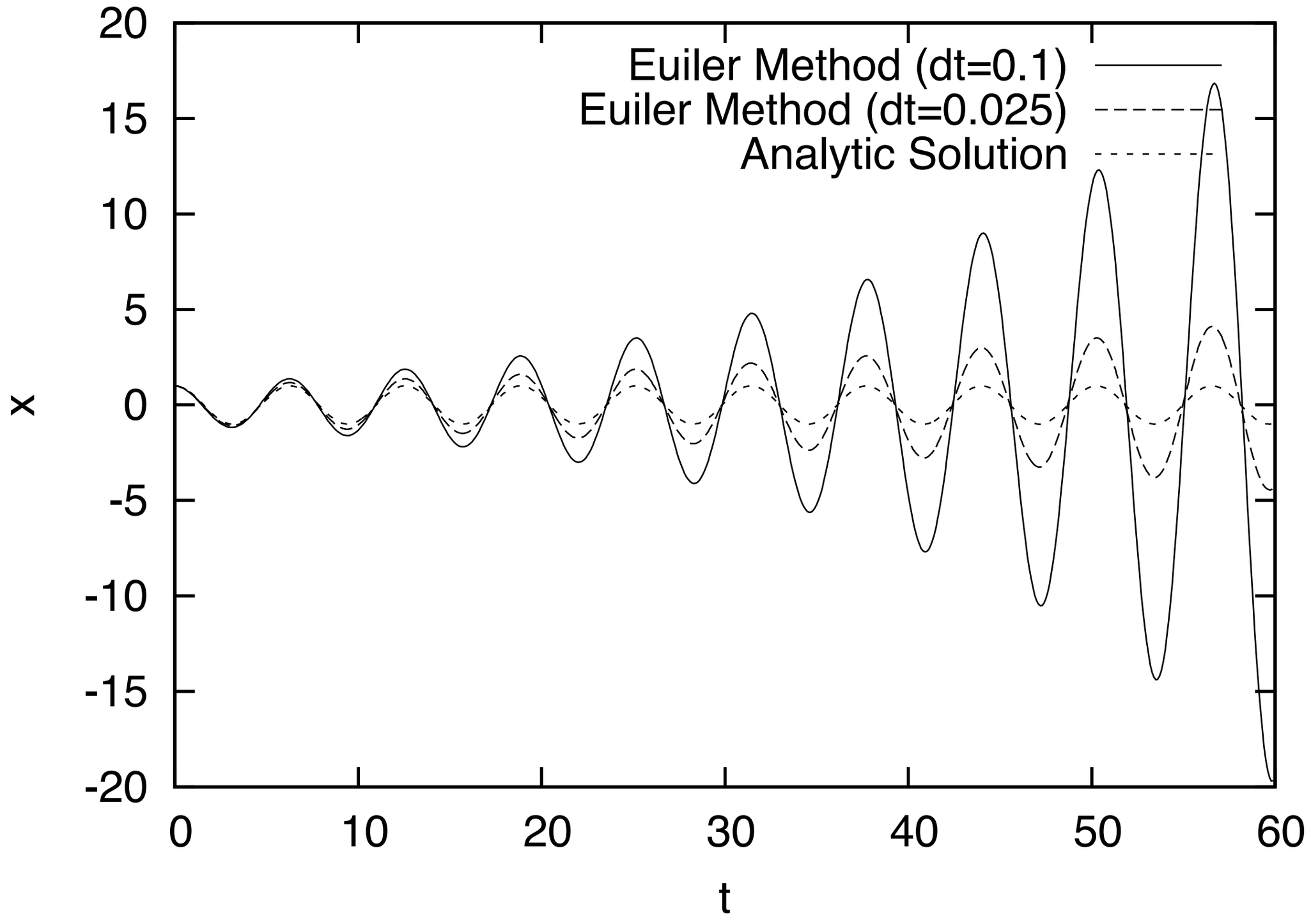
大域誤差

一定時間内で計算したときの誤差

$O(\Delta t^{m+1})\times t/\Delta t=O(\Delta t^{m})$

局所誤差の合計が
大域誤差となる

オイラー法の誤差


Euiler Method (dt=0.1)
Euiler Method (dt=0.025)
Analytic Solution
20
15
10
5
0
-5
-10
-15
-20
x
0
10
20
30
40
50
60
t

オイラー法の誤差

- 誤差を減らすにはタイムステップを小さくする
- タイムステップを小さくすると計算量が増加する
- タイムステップを小さくしていくと丸め誤差の影響が出る

より高精度な計算をするには

- 高次の解法 (ルンゲクッタ法、予測子修正子法、リチャードソン補外など)
- 考える系の構造や性質に着目する
  (後ほどハミルトン系の解法で詳しく説明)

二次のルンゲクッタ法

計算法

$$k_1 = \Delta t f(x_n, t_n),$$
$$k_2 = \Delta t f(x_n + \Delta t/2, t_n + k_1/2),$$
$$x_{n+1} = x_n + k_2$$

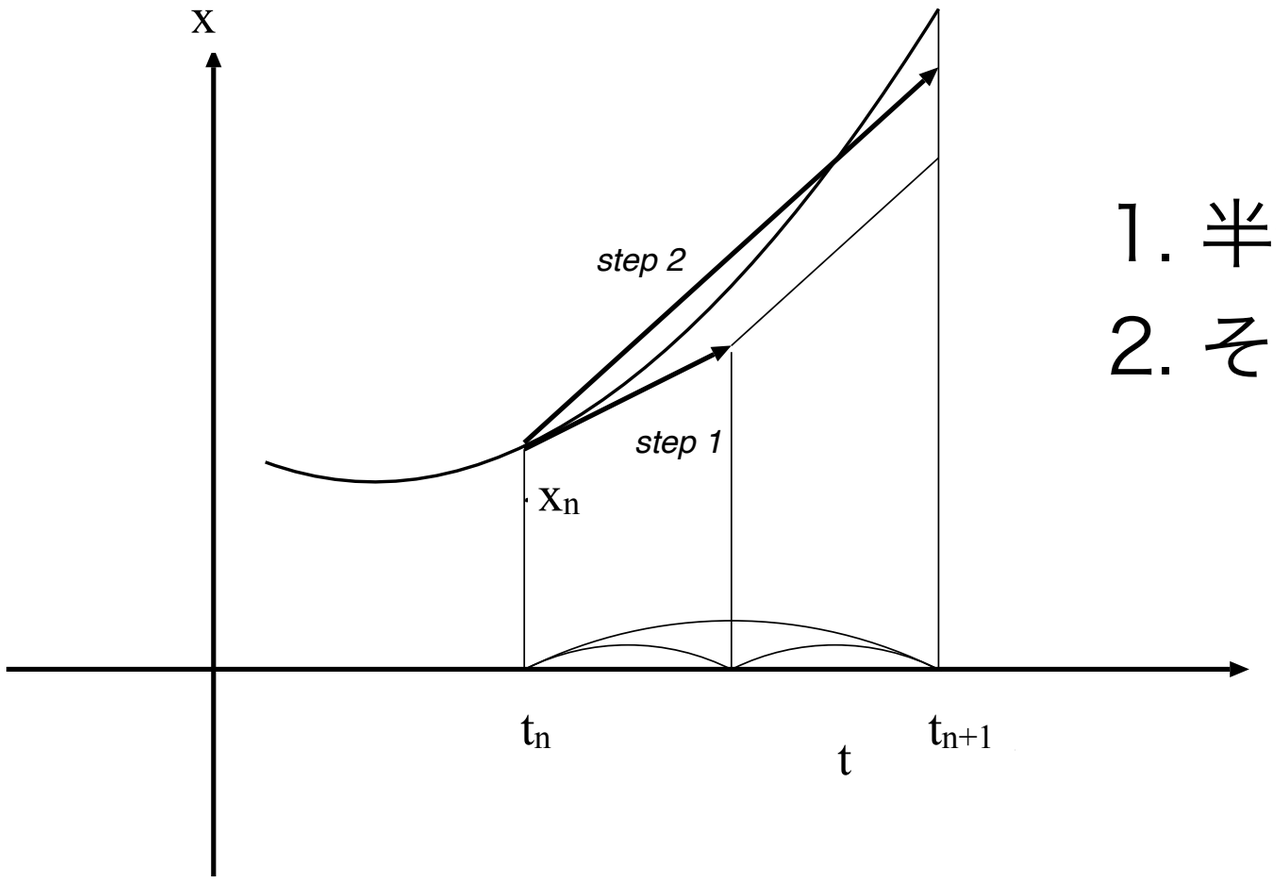
1. 半分のタイムステップまで計算を行う
2. その導関数を用いて本ステップを計算する

この計算法は二次の解法であるため、タイムステップを小さくするとオイラー法より誤差が小さくなる
一方、誤差の変化量が大きいため、タイムステップが大きいときはオイラー法より精度が悪くなることがある

二次のルンゲクッタ法

計算法

$$k_1 = \Delta t f(x_n, t_n)$$
$$k_2 = \Delta t f(x_n + k_1/2, t_n + \Delta t/2)$$
$$x_{n+1} = x_n + k_2$$



1. 半分のタイムステップで導関数を計算
2. その導関数を用いて本ステップを計算

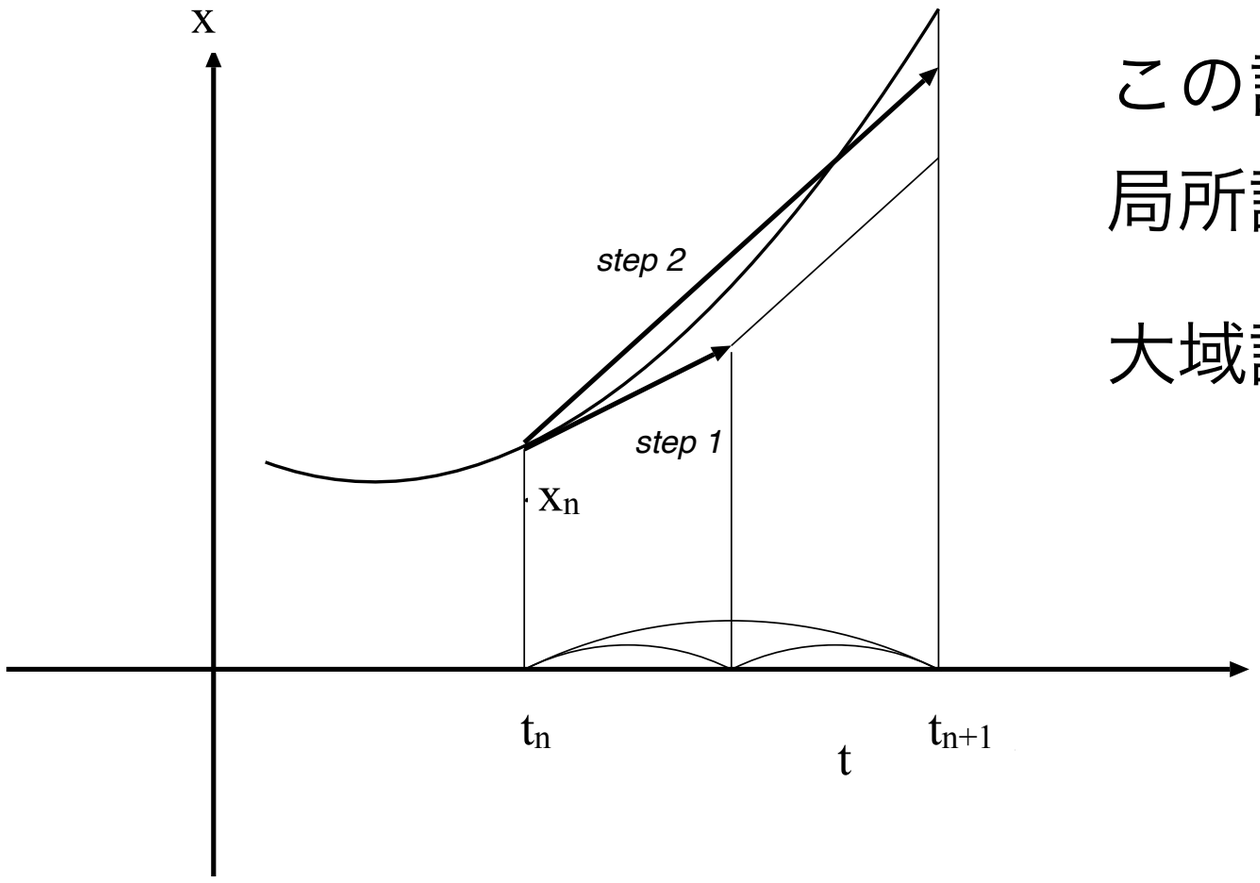
二次のルンゲクッタ法

計算法

$$k_1 = \Delta t f(x_n, t_n)$$

$$k_2 = \Delta t f(x_n + k_1/2, t_n + \Delta t/2)$$

$$x_{n+1} = x_n + k_2$$



この計算法の次数は 2 次。つまり

局所誤差 = $x(t_{i+1}) - x_{i+1} = O(\Delta t^3)$

大域誤差 = $x(t) - x_N = O(\Delta t^2)$

中点法とホイン法

2次のルンゲクッタ法には以下の2つの方法がある

中点法

$$k_1 = \Delta t f(x_n, t_n)$$
$$k_2 = \Delta t f(x_n + k_1/2, t_n + \Delta t/2)$$
$$x_{n+1} = x_n + k_2$$

ホイン法

$$k_1 = \Delta t f(x_n, t_n)$$
$$k_2 = \Delta t f(x_n + k_1, t_n + \Delta t)$$
$$x_{n+1} = x_n + \frac{1}{2}(k_1 + k_2)$$

古典的ルンゲクッタ法
(4次ルンゲクッタ法)

最も広く用いられている常微分解法

- 4段公式で到達可能な4次の公式
- 係数が簡単で計算量も少ない
- 5次以上では次数<段数となる

- オイラー法より精度が良く、タイムステップに対して誤差の減少が大きい
- テイラー展開の４次の項まで使用する

古典的ルンゲクッタ法

計算法

$$
\begin{aligned}
k_1 &= \Delta t f(x_n, t_n), \\
k_2 &= \Delta t f(x_n + \Delta k_1/2, t_n + \Delta t/2), \\
k_3 &= \Delta t f(x_n + \Delta k_2/2, t_n + \Delta t/2), \\
k_4 &= \Delta t f(x_n + \Delta k_3, t_n + \Delta t), \\
x_{n+1} &= x_n + (k_1 + 2k_2 + 2k_3 + k_4)/6
\end{aligned}
$$

古典的ルンゲクッタ法

- k_1：オイラー法の計算法と同じ
- k_2：2次ルンゲクッタ法の計算法と同じ
- k_3：k_2の情報を使用する
- k_4：k_3の情報を使用する

- k_1からk_4はx_nでの傾きを表している
- 最後に重み付けをして傾きの合成を行っている

高階の常微分方程式の解法

高階の微分方程式を計算する場合は、
1階の連立微分方程式に変更して計算を行う

$$\frac{d^2x}{dt^2} = f\left(x, \frac{dx}{dt}, t\right)$$

上記の微分方程式に対して、下記のような変数変換を行う

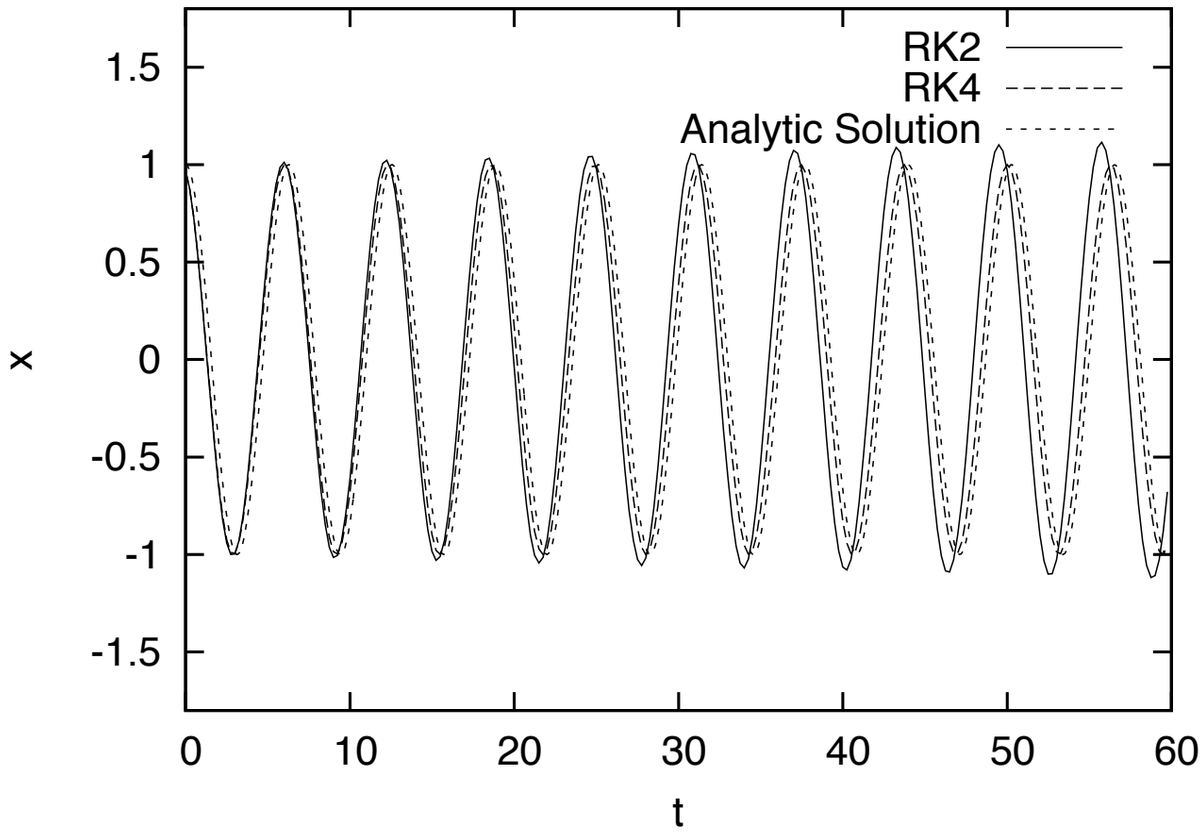
$$\begin{cases} X_0(t) = x(t) \\ X_1(t) = \frac{dx}{dt} = x'(t) \end{cases}$$

その結果、以下のような連立微分方程式が得られる

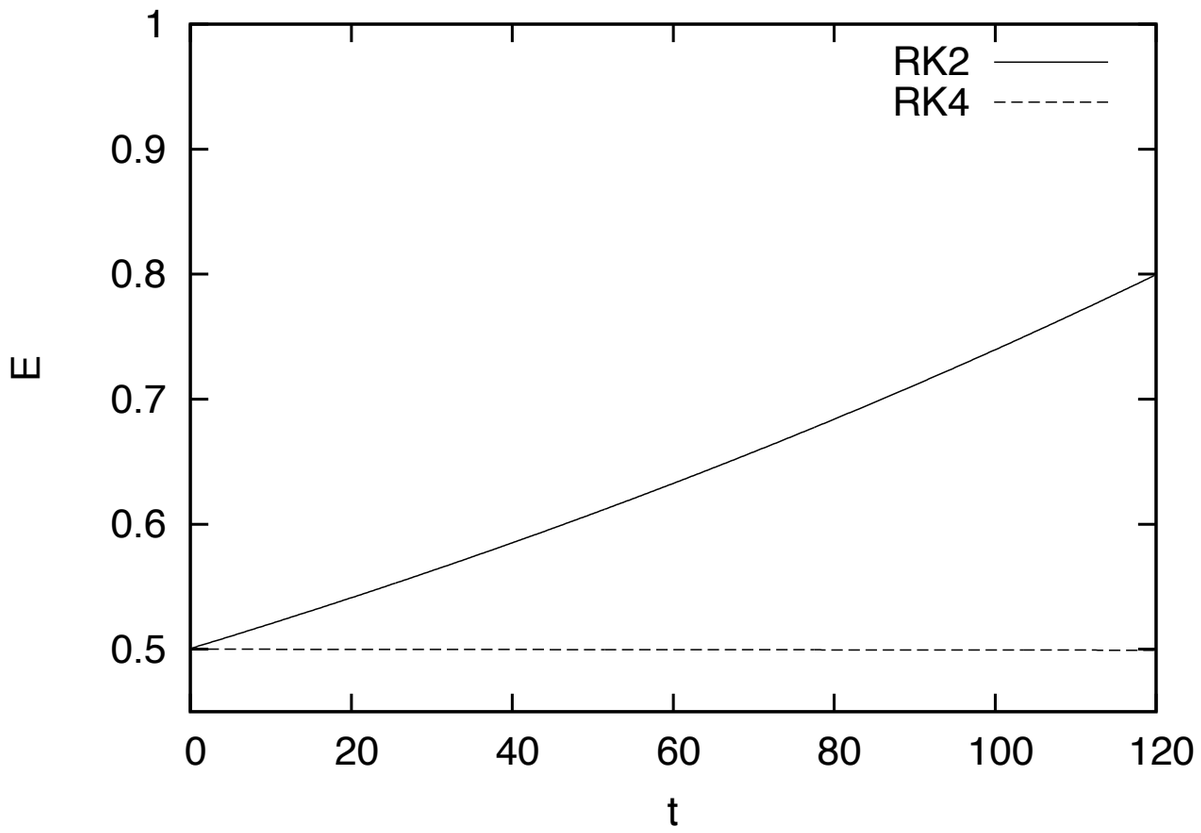
$$\begin{cases} \frac{X_0}{dt} = X_1 \\ \frac{X_1}{dt} = f(X_0, X_1, t) \end{cases}$$

ルンゲクッタ法の誤差

数値解



エネルギー誤差



N体シミュレーションの解法

基礎方程式

$$\frac{d^2x}{dt^2} = \sum_{j \neq i}^{N} G m_j \frac{x_j - x_i}{|x_j - x_i|^3}$$

N体シミュレーションでは、粒子間の重力相互作用を計算する

重力計算の量は粒子数の二乗に比例する($O(N^2)$の計算)

N体シミュレーションの解法

無衝突系：近接相互作用が系の進化に影響しない
衝突系：近接相互作用が系の進化に影響する

衝突系の場合、近接相互作用を細かく計算する必要がある
そのため、高次の数値積分法が良く使われる
粒子数が増えると計算量が急激に増加するため、
重力計算を効率よく行う必要がある(GRAPEなど)

N体シミュレーションの解法

基礎方程式

$$\frac{d^2x}{dt^2} = \sum_{j \neq i}^{N} G m_j \frac{x_j - x_i}{|x_j - x_i|^3}$$

- ルンゲクッタ法はどんな常微分方程式でも解ける
- N体シミュレーション考えるのはハミルトン系

→　ハミルトン系専用の数値解法で良い

例：調和振動子

一次元調和振動子　$$H = \frac{1}{2}(p^2 + q^2)$$

運動方程式　$$\frac{dq}{dt} = p, \frac{dp}{dt} = -q$$

この方程式の厳密解は相平面での角度τの回転で表せる

$$\begin{bmatrix} q(\tau) \\ p(\tau) \end{bmatrix} = \begin{bmatrix} \cos\tau & \sin\tau \\ -\sin\tau & \cos\tau \end{bmatrix} \begin{bmatrix} q(0) \\ p(0) \end{bmatrix}$$

この変換はJacobiの行列式が1となる

↓

この変換は相平面での面積を保存する

例：調和振動子

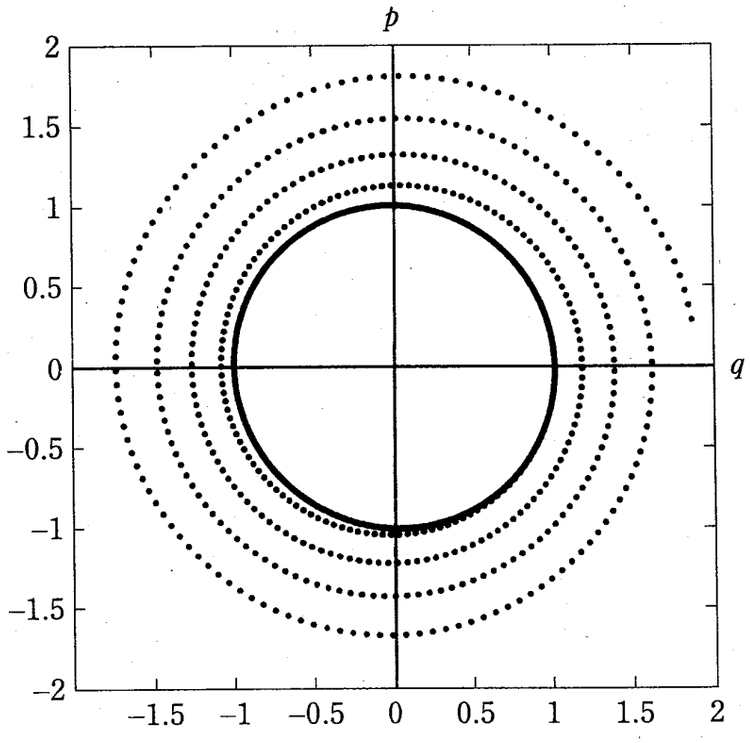
オイラー法

$$\begin{bmatrix} q' \\ p' \end{bmatrix} = \begin{bmatrix} 1 & \tau \\ -\tau & 1 \end{bmatrix} \begin{bmatrix} q \\ p \end{bmatrix}$$

シンプレクティック法

$$\begin{bmatrix} q' \\ p' \end{bmatrix} = \begin{bmatrix} 1 & \tau \\ -\tau & 1-\tau^2 \end{bmatrix} \begin{bmatrix} q \\ p \end{bmatrix}$$

- 共に厳密解に対してτの1次まで一致
- シンプレクティック法は係数行列の行列式の値が1

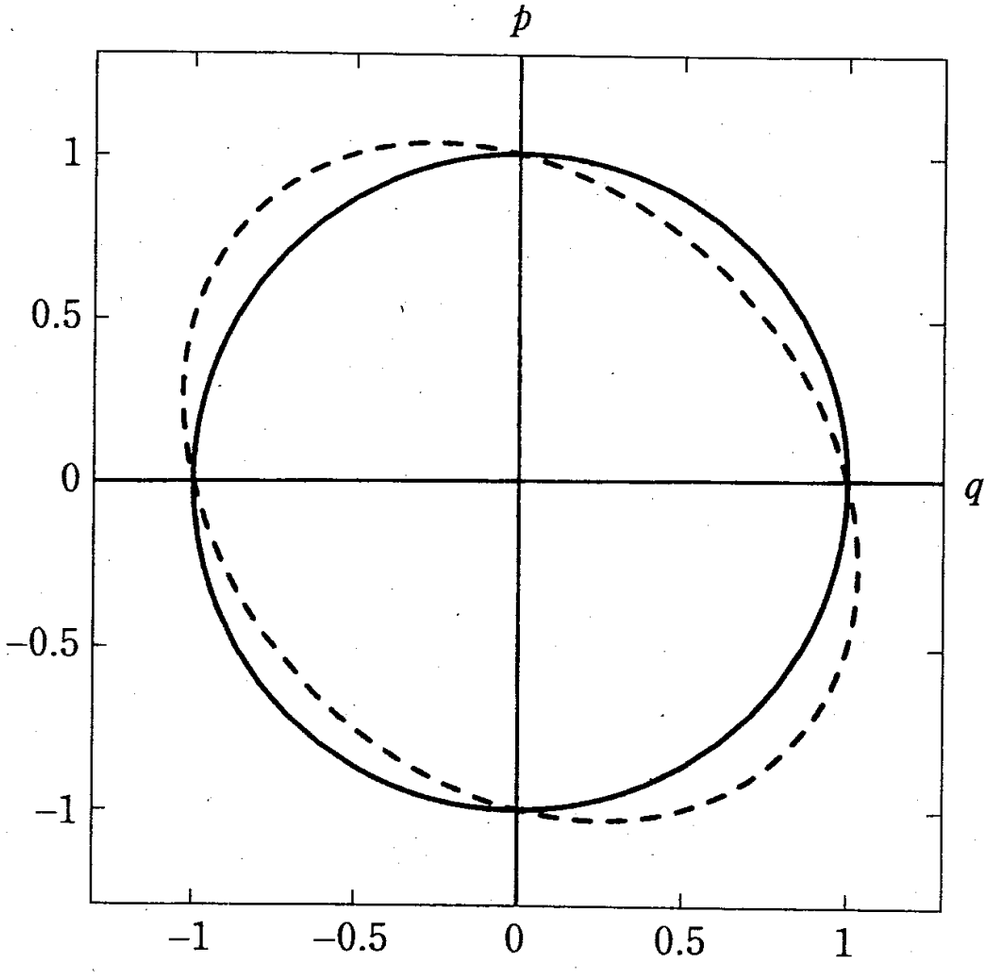


- オイラー法はエネルギーが
1ステップごとに$(1+\tau^2)$
- シンプレクティック法は厳密解に非常に近い

例：調和振動子

シンプレクティック法 $$\begin{bmatrix} q' \\ p' \end{bmatrix} = \begin{bmatrix} 1 & \tau \\ -\tau & 1-\tau^2 \end{bmatrix} \begin{bmatrix} q \\ p \end{bmatrix}$$

- シンプレクティック法に対して任意の座標変換を行う



$$\frac{1}{2}(p^2+q^2)+\frac{\tau}{2}pq = const$$

- シンプレクティック法は厳密解に対して45度傾いた楕円となる
  → 誤差が蓄積しない

リープフロッグ法

ハミルトニアン

$$H = T(p) + V(q)$$

に対するシンプレクティックな数値解法を考える

$$q' = q + \tau(\frac{\partial T}{\partial p}), p' = p - \tau(\frac{\partial V}{\partial q})$$

(q, p)→(q’, p), (q’, p)→(q’, p’) の合成として得られる

シンプレクティック性

ある種のハミルトニアンにおいて、時間発展しても保存量を持つ

リープフロッグ法

Hamilton方程式を以下の形に書き直す

$$\frac{dq}{dt} = \frac{\partial H}{\partial p}, \frac{dp}{dt} = -\frac{\partial H}{\partial q}$$

$$z = (q, p) \qquad \frac{dz}{dt} = \{z, H(z)\}$$

また、関数Fに作用する微分作用素は以下のように定義できる

$$D_{\mathrm{G}} F \equiv \{F, G\}$$

よってHmilton方程式はこの表記を使用すると

$$\frac{dz}{dt} = D_H z$$

時間0からτまでの時間発展は

$$z(\tau) = [\exp(\tau D_H)] z(0)$$

リープフロッグ法

$H = T(p) + V(q)$　の形のハミルトニアンを考慮した場合

形式解は　$z(\tau) = \exp[\tau(D_T + D_V)]z(0)$

$$D_H = D_T + D_V$$

(D_TとD_Vは非可換である事に注意)

一般的に、非可換は作用素X, Yで定義されるZについて

$$\exp X \exp Y = \exp Z$$

Baker-Campbell-Hausdorffの公式が成り立つ

$$Z = X + Y + \frac{1}{2}[X, Y] + \cdots$$

この公式を用いると、２次のシンプレクティック解法は以下のように書ける

$$\exp[\tau(D_T + D_V)] = \exp\left(\frac{\tau}{2}D_T\right)\exp(\tau D_V)\exp\left(\frac{\tau}{2}D_T\right) + O(\tau^3)$$

リープフロッグ法

計算法

$$v_{i+1/2} = v_i + a(x_i)\frac{\Delta t}{2},$$

$$x_{i+1} = x_i + v_{i+1/2}\Delta t,$$

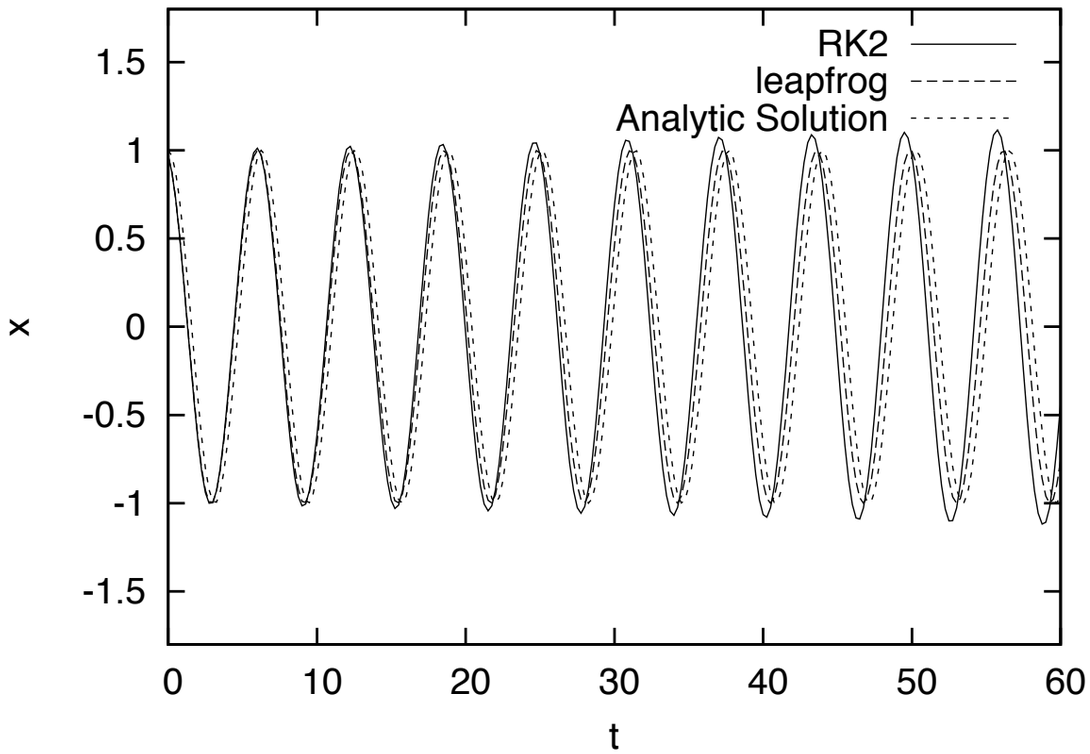
$$v_{i+1} = v_{i+1/2} + a(x_{i+1})\frac{\Delta t}{2}$$

特徴

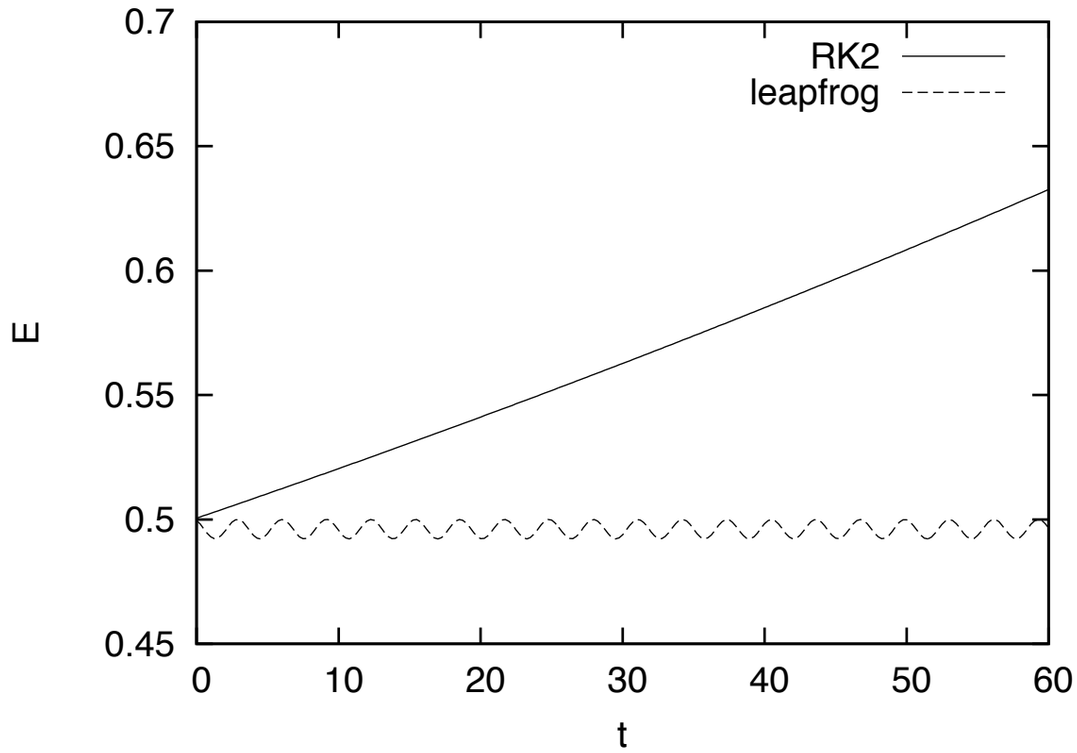
- 二次のシンプレクティック公式
- 長時間計算におけるエネルギー誤差の蓄積が無い

リープフロッグ法

数値解



エネルギー誤差



特徴

二次のシンプレクティック公式

ある種のハミルトニアンにおいてエネルギー誤差の蓄積が無い

局所誤差は古典的ルンゲクッタ法より悪い

まとめ

- 数値計算法の基礎
  - 単精度、倍精度の意味 誤差 (情報落ち、桁落ち)
- 微分方程式の解法の基本及び局所誤差と大域誤差
  - 公式の次数を定義した
    - オイラー法は1次
    - 古典ルンゲクッタは4次
- N 体シミュレーション用の解法
  - リープフロッグ法は2次
  - シンプレクティック性：**ある種のハミルトニアンにおいて誤差が蓄積されない**