

計算物理学B

第11回

常微分方程式 Part II

藤本 悠輝、野垣 康介
(藤本担当回)

質問等あればメールでも受け付けます：
yuki.fujimoto.phys__at__niigata-u.ac.jp
(__at__ を @ に変えてください)

講義予定

10/07 両名:四則演算

10/14 野垣:制御文(for, if)

10/21 野垣:関数

10/28 藤本:配列(numpy)

11/04 藤本:可視化(matplotlib)

11/11 野垣:数値微分

11/18 藤本:数値積分

～中間レポート～

12/09 野垣:モンテカルロ1

12/16 野垣:モンテカルロ2

12/23 藤本:常微分方程式1

01/13 藤本:常微分方程式2

01/20 藤本:偏微分方程式1

01/27 藤本:偏微分方程式2

02/03 野垣:最適化

～期末レポート～

あくまで予定なので変更の可能性あり

授業で用いるURL

Google Colab:

<https://colab.research.google.com/?hl=ja>

GitHub上の講義サイト:

https://github.com/nogaki/Computational_Physics_B

今週の教材:

https://github.com/nogaki/Computational_Physics_B/blob/main/week11

第11回 常微分方程式その2

- scipyライブラリのODEソルバーの使い方
- 分子動力学シミュレーション

この講義は以下を参考に準備されています:

<https://github.com/vlvovch/PHYS6350-ComputationalPhysics>

「実践計算物理学」野本拓也、是常隆、有田亮太郎 著 (共立出版)

「理工学のための数値計算法」水島二郎、柳瀬眞一郎 著 (数理工学社)

常微分方程式 (ODE) ソルバー

1階の常微分方程式 (ODE):

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

この形の微分方程式を与えられた初期条件

$$x(t = 0) = x_0$$

のもとで解くことを考える。

前回の講義では、基本的なオイラー法やルンゲ・クッタ法などの解法を紹介した。

ODEソルバー

前回の講義ノートでは、オイラー法、中点則(RK2)、RK4などによるODEソルバー関数を自前で定義した。

しかし、実際に研究で用いる場合は `scipy` ライブラリに含まれる `solve_ivp` を使うのが良い:

```
from scipy.integrate import solve_ivp
```

- デフォルトで適応刻み幅制御
 - 高精度のアルゴリズム(5次のRunge-Kutta)
 - 陰解法
- などを実装

solve_ivpの使い方

```
from scipy.integrate import solve_ivp
```

solve_ivp で次の形の微分方程式を解くことを考える:

$$\frac{dy}{dt} = f(t, \mathbf{y}; \{c_i\}), \quad \mathbf{y}(t_0) = \mathbf{y}_0$$

```
solution = solve_ivp(fun, # 関数 fun(t, y, c0, c1, ...)
                    [t0, t_end], # tの初期値と終端値
                    y0, # y(t) の t = t0での初期値
                    method='RK45', # ODEの解き方
                    t_eval=time_points, # tを見積もる点
                    args=(c0, c1)) # tとy以外の関数fのパラメタ
```

```
t = solution.t # ODEの解y(t)を見積もるtの点:要素nの配列
y = solution.y # 解y(t):yが長さmのベクトルのとき、サイズ(m,n)の配列
```

solve_ivpの応用例(1)

Lorenz attractor:

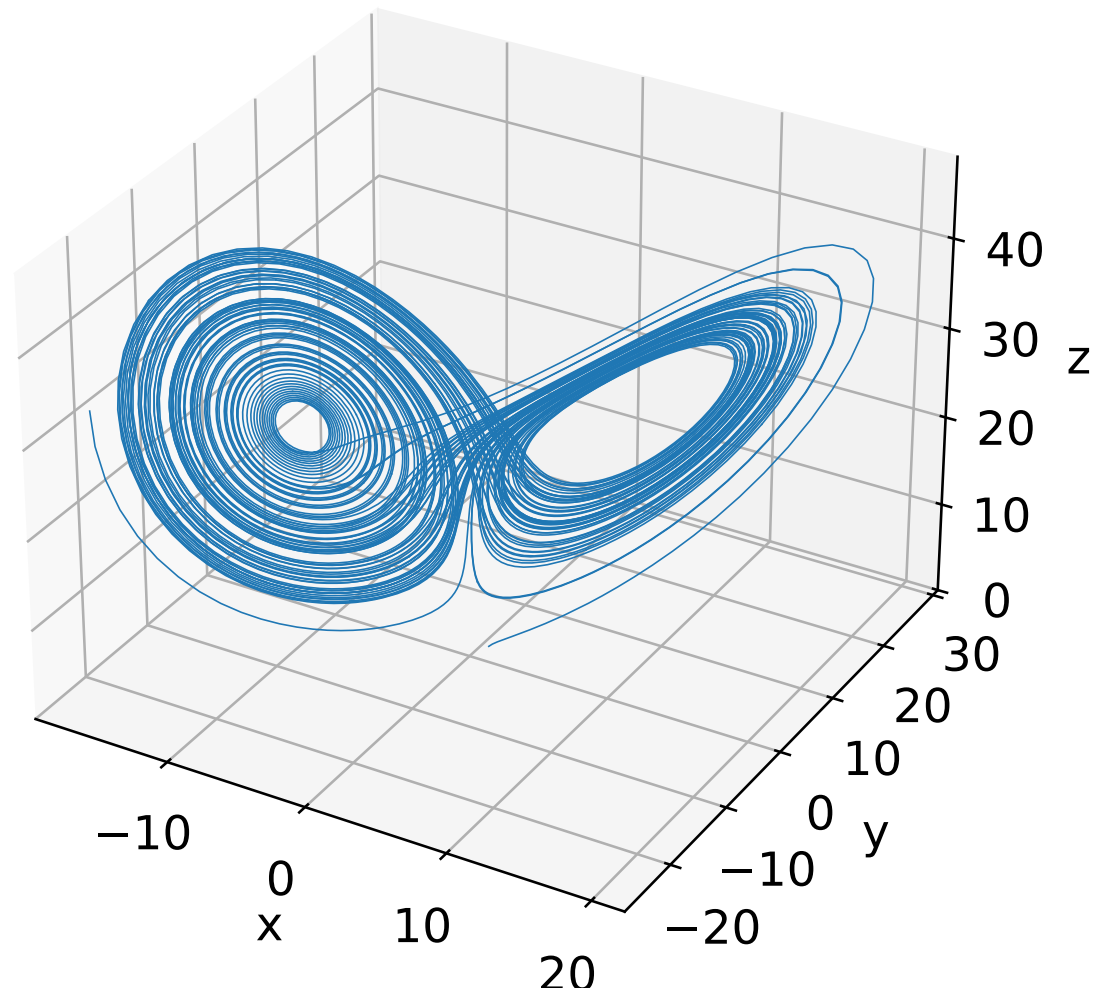
$$\frac{dx}{dt} = \sigma(y - x),$$

$$\frac{dy}{dt} = x(\rho - z) - y,$$

$$\frac{dz}{dt} = xy - \beta z,$$

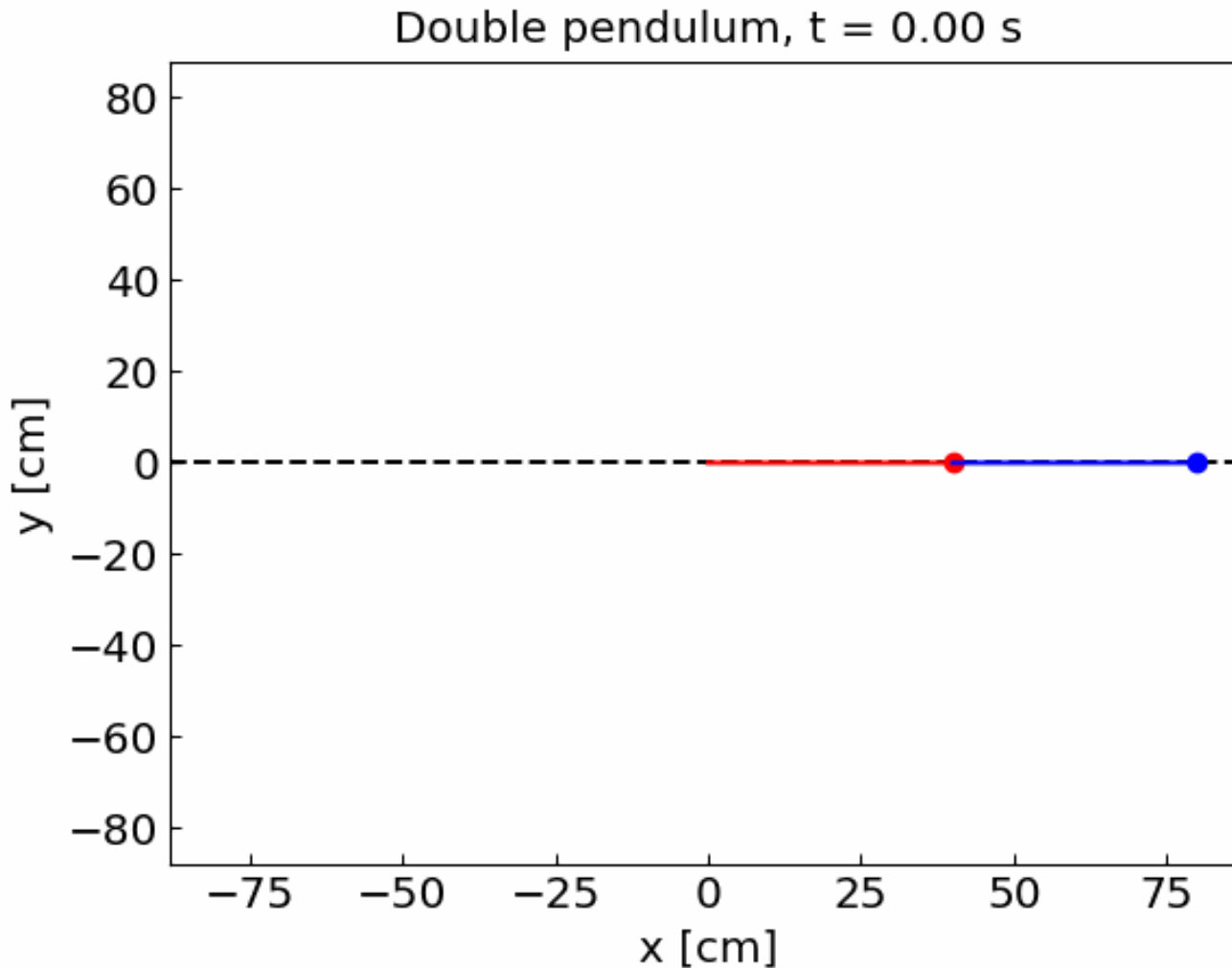
カオスの例

Lorenz Attractor



solve_ivpの応用例(2)

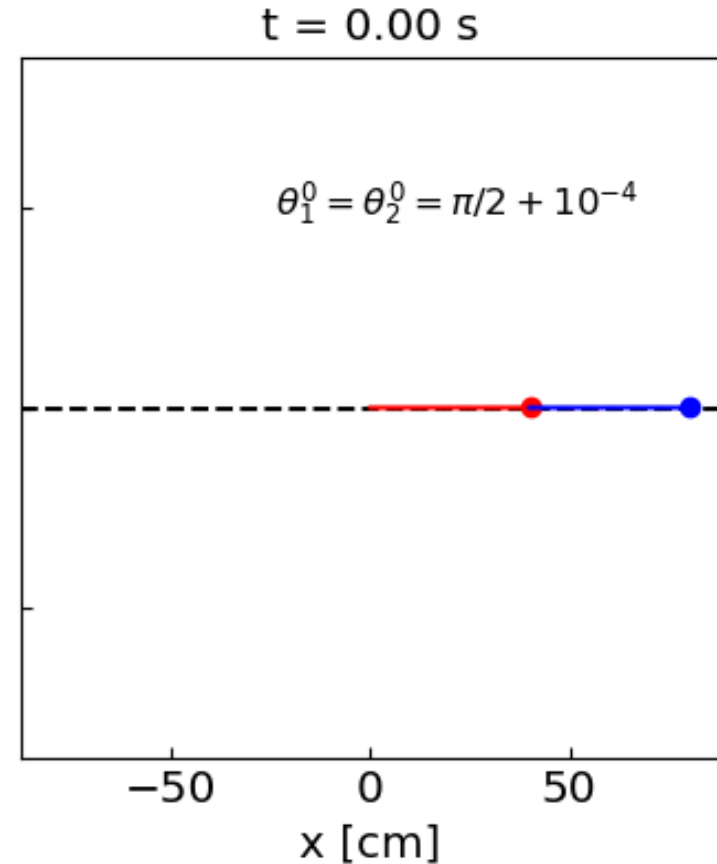
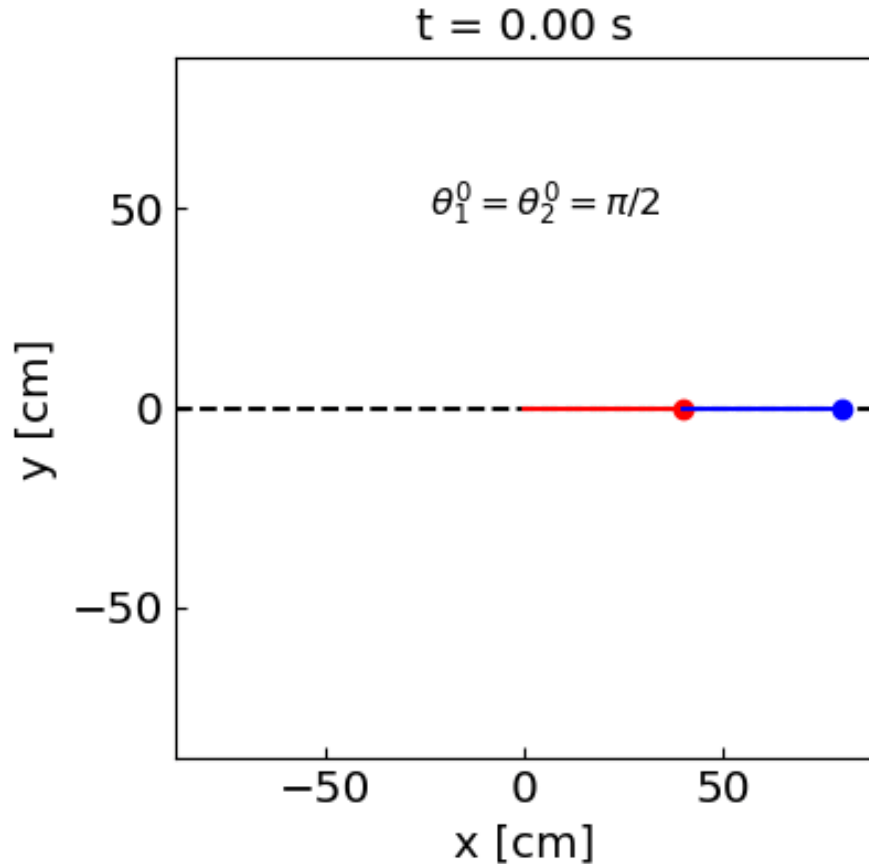
二重振り子



solve_ivpの応用例(2)

二重振り子

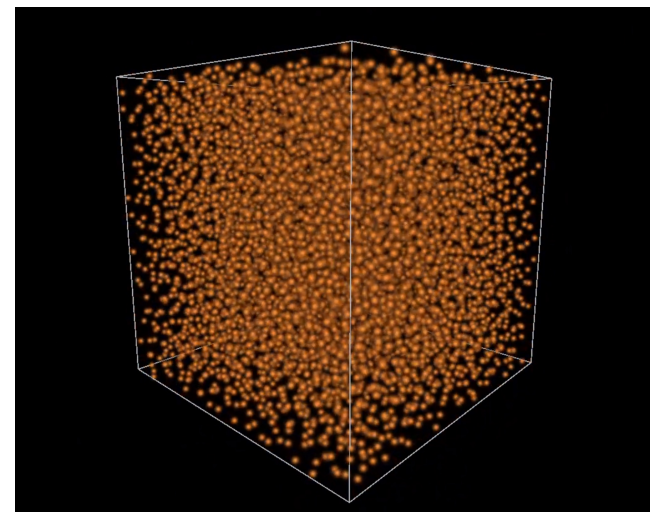
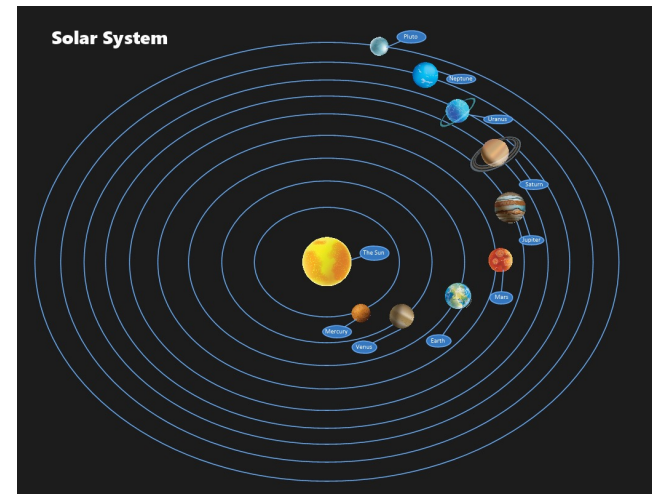
Double pendulum



初期値に鋭敏→カオス的振る舞いの例

分子動力学法(Molecular Dynamics)

- 2体力の下でのN粒子系
- 運動方程式(古典N体問題):
$$m\ddot{\mathbf{r}}_i = - \sum_j \nabla_i V_{ij}(|\mathbf{r}_i - \mathbf{r}_j|)$$
- 惑星の運動
 - 太陽系のシミュレーション
- 統計力学と状態方程式
 - 有限体積の箱と周期境界条件



分子動力学法の方程式

運動方程式 (古典 N 体問題): $m_i \ddot{\mathbf{r}}_i = - \sum_{j \neq i} \nabla_i V(\mathbf{r}_i, \mathbf{r}_j)$

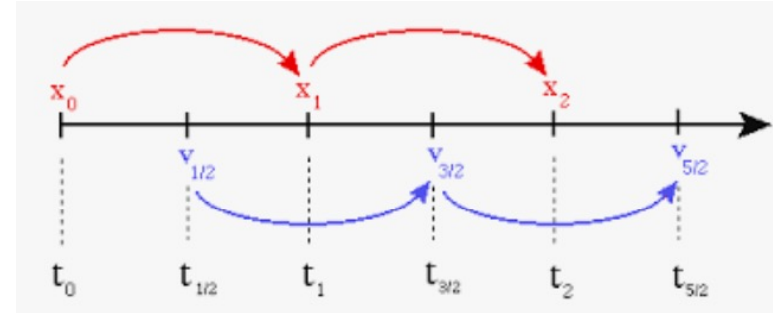
- その際、解法が満たしてほしい性質:
 - 安定性 (長く走らせるので)
 - エネルギー保存
 - 時間反転対称性
- 運動方程式を1階のODEの連立系で書き直す:

$$\begin{aligned} \dot{\mathbf{r}}_i &= \mathbf{v}_i, \\ \dot{\mathbf{v}}_i &= -(m_i)^{-1} \sum_{j \neq i} \nabla_i V(\mathbf{r}_i, \mathbf{r}_j), \end{aligned}$$

そして、リープ・フロッグ (蛙飛び) 法を使って解く

リープ・フロッグ法

連立微分方程式: $\frac{dx}{dt} = v,$
 $\frac{dv}{dt} = f(x, t).$



Leap frog (蛙飛び) 法:

$$x(t + h) = x(t) + hv(t + h/2),$$

$$v(t + 3h/2) = v(t + h/2) + hf[x(t + h), t + h],$$

- 位置は整数ステップで計算
- 速度は半整数ステップで計算

誤差: 全体で $O(h^2)$ ← 中点法(RK2)と同じ

利点: (1) 時間反転対称性 (2) エネルギー保存

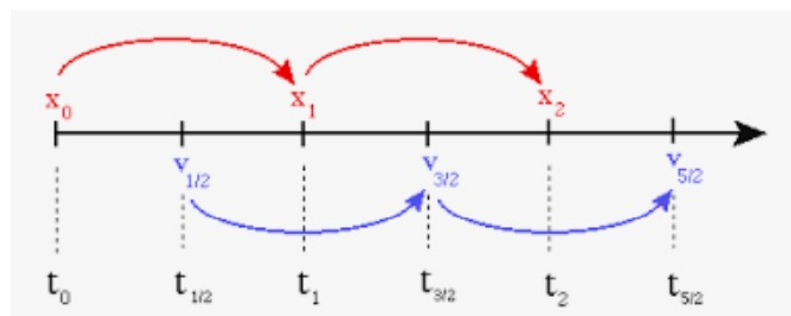
速度Verlet法

分子動力学の文脈ではleap frog法は速度Verlet法という：

$$v(t + h/2) = v(t) + \frac{h}{2} f[x(t), t],$$

$$x(t + h) = x(t) + hv(t + h/2),$$

$$v(t + h) = v(t + h/2) + \frac{h}{2} f[x(t + h), t + h].$$



ボックスシミュレーションと統計力学

- 統計力学: 多数の粒子の系
- 微視的にはニュートンの運動方程式に従う
- 有限体積でのシミュレーションで無限系を再現
 - 周期境界条件
 - 最小イメージ規約 (最短距離の他粒子との相互作用)
- N が十分大きければ系の性質はマクロな量で指定可
 - U, V, N 表示: 一定エネルギーのミクロカノニカル分布
 - T, V, N 表示: 一定温度のカノニカル分布
- MD法は状態方程式を計算できる

2体力

- 以降、質量は $m=1$ と仮定。
- 2体ポテンシャルは相対距離にのみ依存：

$$\ddot{\mathbf{r}}_i = - \sum_{j \neq i} \frac{dV(r_{ij})}{dr_{ij}} \frac{\mathbf{r}_i - \mathbf{r}_j}{r_{ij}} .$$

Lennard-Jonesポテンシャル

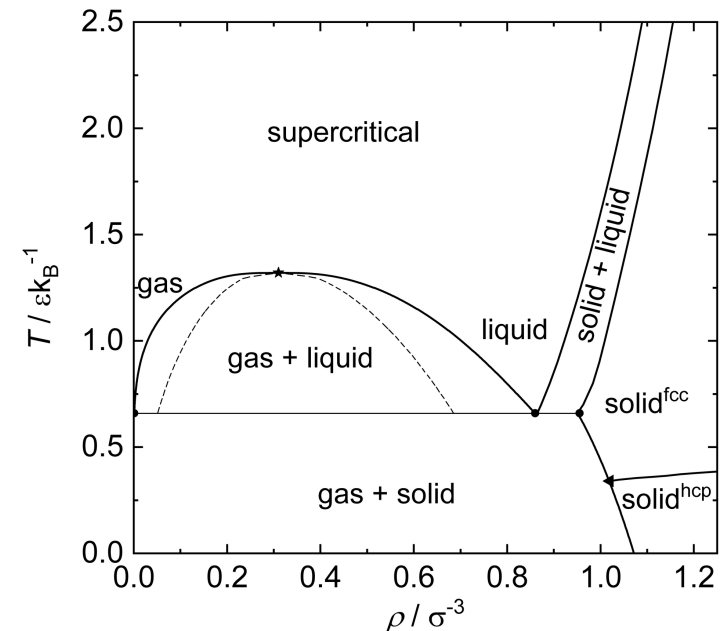
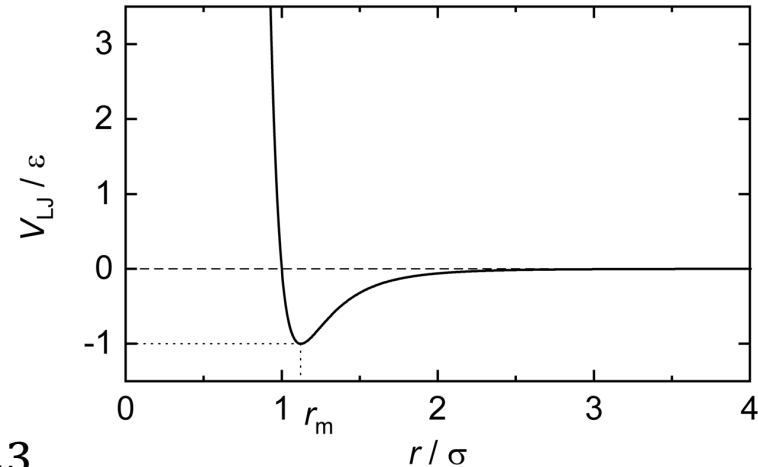
- Lennard-Jonesポテンシャル:

$$V_{\text{LJ}}(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right]$$

- 無次元量:

$$\tilde{r} = r/\sigma \quad \tilde{T} = T/(k_B\epsilon) \quad \tilde{n} = n\sigma^3$$

- 性質:
 - 複数の相転移、臨界点
 - 解析的には解けないが、MDシミュレーション可



シミュレーション

系の初期化:

- 空間座標
 - 格子上に粒子を配置
 - 粒子の重なりを防ぐ (r^{-12} の項に注意)
- 速度
 - 各粒子の速度を Gaussian (= Maxwell-Boltzmann) 分布からサンプルする。

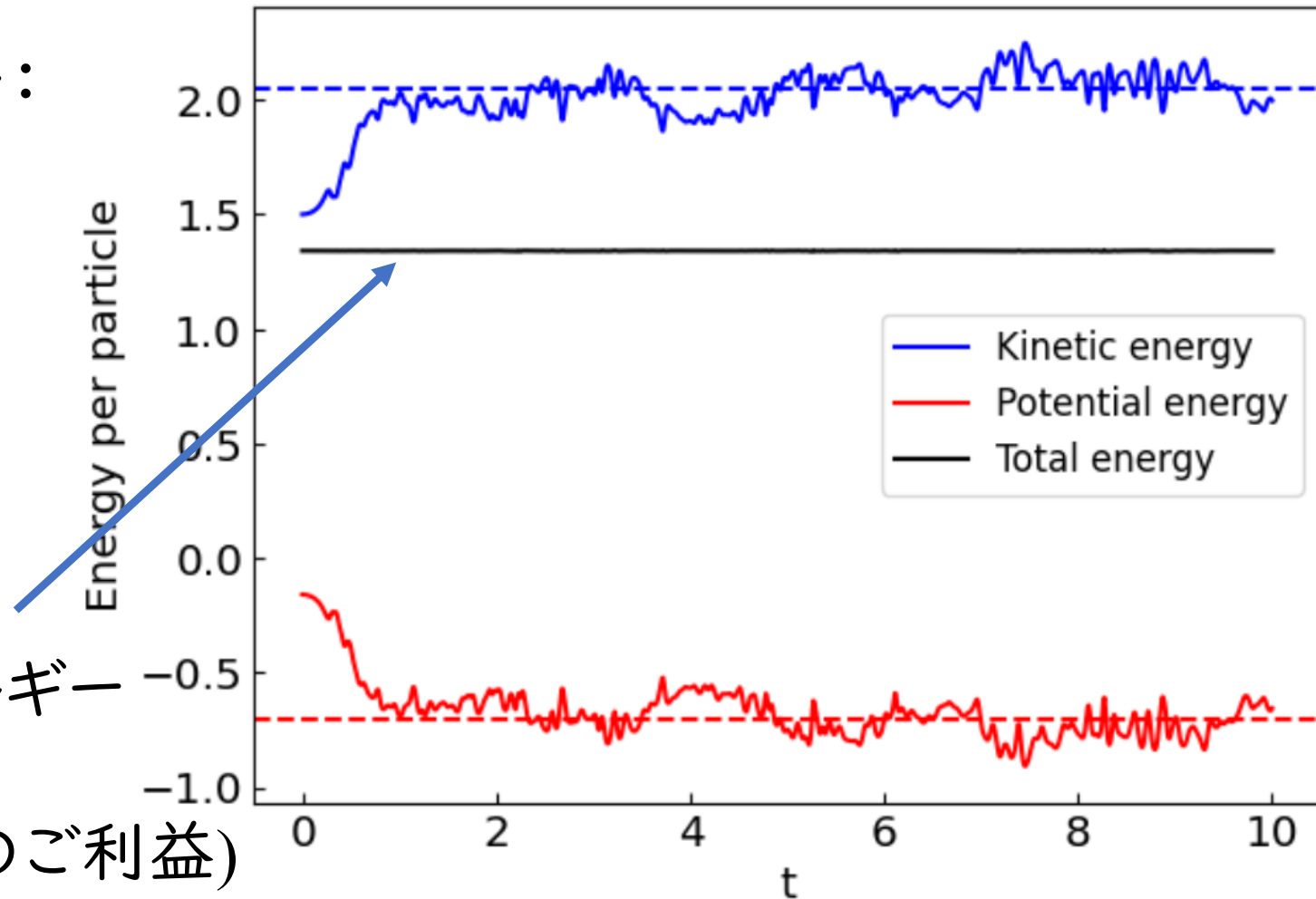
$$f(\mathbf{v}) d^3 \mathbf{v} = \left[\frac{m}{2\pi k_B T} \right]^{3/2} \exp \left(-\frac{mv^2}{2k_B T} \right) d^3 \mathbf{v},$$

シミュレーション結果

$$T = 1, \rho = 0.1, N = 64$$

LJ fluid, $T_0 = 1.0, \rho = 0.1$

エネルギー:



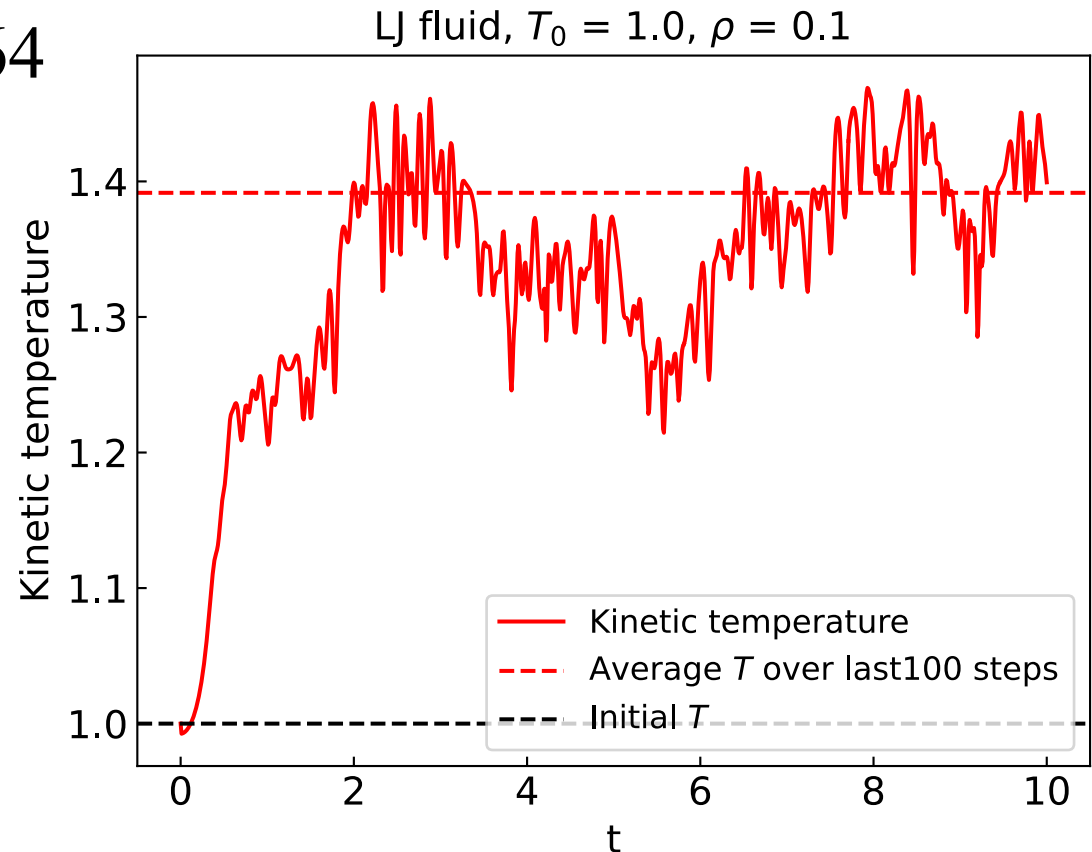
系の全エネルギー
が保存
(leap-frog法のご利益)

シミュレーション結果

$$T = 1, \rho = 0.1, N = 64$$

温度:

$$T_{\text{kin}} = \langle \mathbf{v}_i^2 \rangle / 3$$



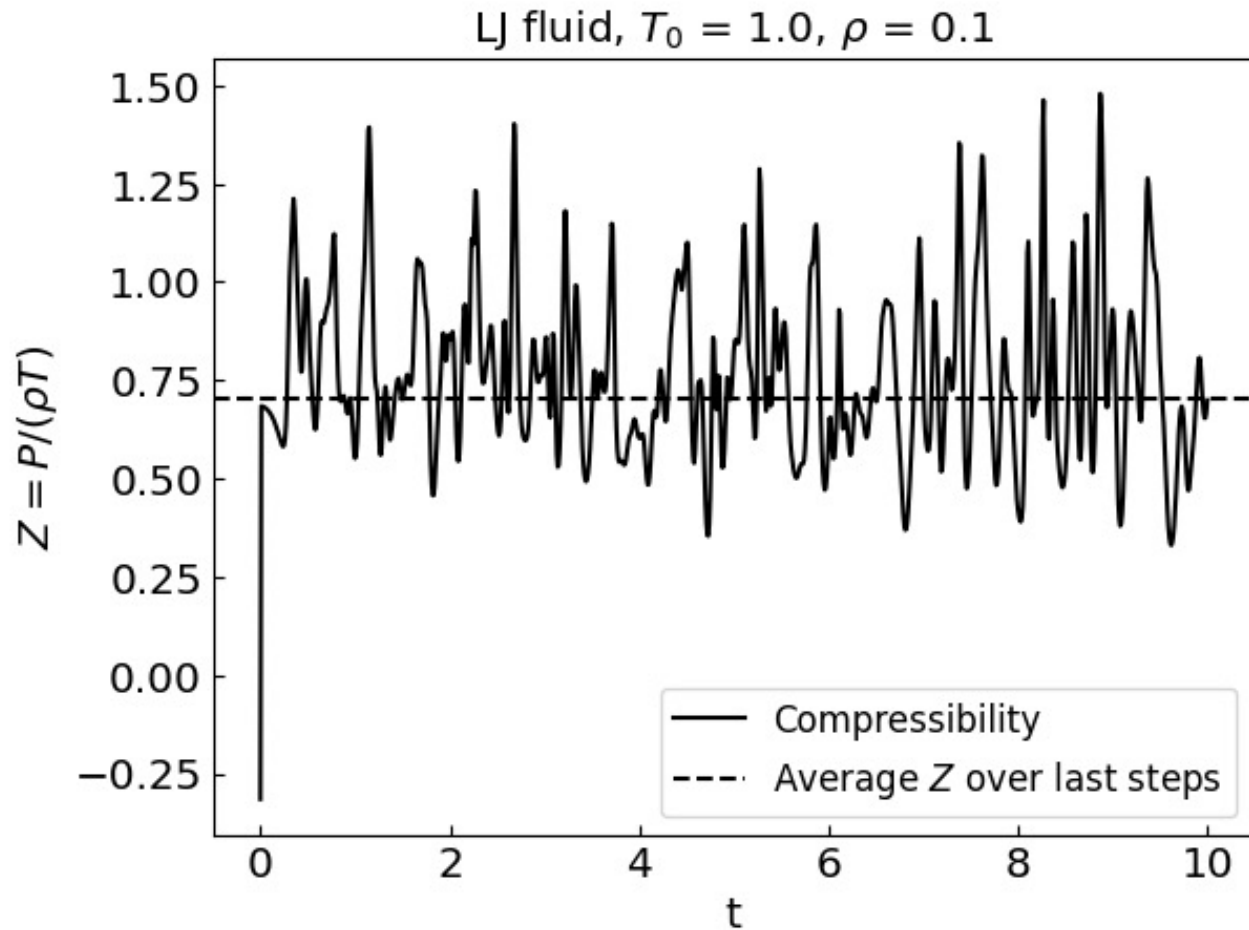
気体分子運動論から読み出した温度 T_{kin} は初期値 $T = 1.0$ から乖離する。なぜなら、系は時間とともに平衡化し、ミクロカノニカルでは温度は保存しないから

シミュレーション結果

$$T = 1, \rho = 0.1, N = 64$$

圧縮率
(実在気体の
理想気体から
のずれを表す):
 $Z = p/(\rho T)$

$Z=1$ が理想気体



実習タイム

今週のGitHubにあげたファイルのうち

- Lorenz方程式
- 二重振り子
- 分子動力学シミュレーション

のコードをよく読んで理解せよ。

また、パラメタを変えてコードを走らせてみよ。

特に、Lorenz方程式と二重振り子は初期値を変えて、カオス的な振る舞いを観察してみよう。