第三の科学
シミュレーションの世界

理工学群工学システム学類

Q(@ITF_QSYS)

資料の置き場

Google drive に保存

目次

シミュレーションとは

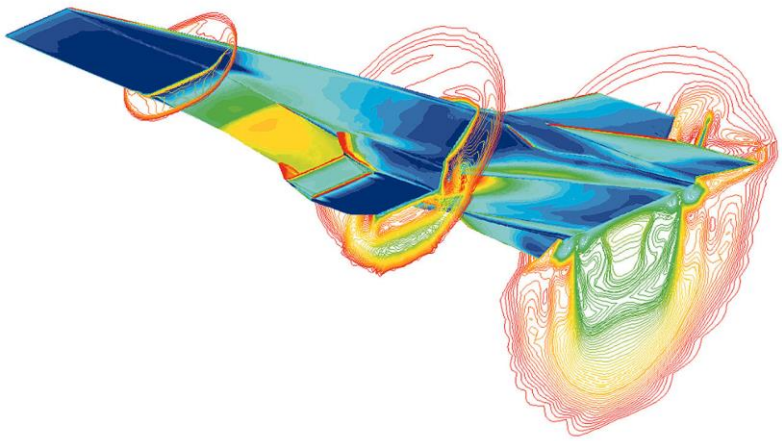
シミュレーションとは

シミュレーションとは、
コンピューターまたは模型を用いて実システムと同等のモデルを作り上げ（モデリング）、モデルによって評価をすることである。

現実では行うことのできない実験には非常に効果的である（例：宇宙の形成）。

物理学とシミュレーション

- 物理学はシミュレーションによって発展してきた。
- シミュレーションなしでは解くことのできなかった方程式を数値的に解くことができるようになり、物理学が発展した。
- 近年、コンピューターの発展により性能が向上した



https://ja.wikipedia.org/wiki/%E6%95%B0%E5%80%A4%E6%B5%81%E4%BD%93%E5%8A%9B%E5%AD%A6
より引用

シミュレーションの手順

1. 目的の明確化
   なにをどのくらいの精度でシミュレーションするのかを決める
2. データの収集
   対象とする問題の範囲を決め、パラメータなどを明確にする
3. モデリング
   変数の関係を数式やアルゴリズムなどコンピューターに記述できる形にする
4. シミュレーターの作成と実験
   プログラミング、模型の作成を行って、実験する。
5. 結果の評価、分析

シミュレーションの注意点

- シミュレーションはどの程度の精度で行うかを注意する必要がある。
- 精度が不足していたら結果は利用できないし、十分すぎる精度ではシミュレーションにかかるコストが増大してしまう。
- また、モデリングにおいて、近似、離散化をしているときには誤差があるので注意が必要。

シミュレーションは
万能ではない！

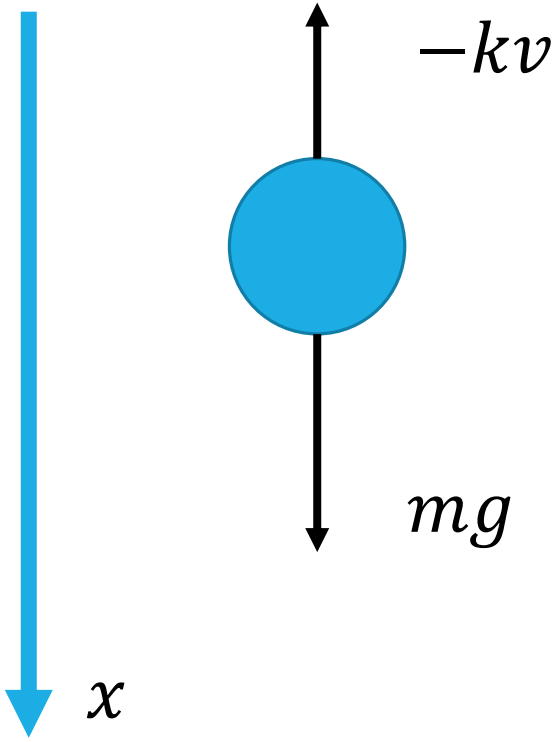
シミュレーションの実例　その1

抵抗のある落下運動

抵抗のある落下運動

- 目的
  - 抵抗のある落下運動を調べる
- データの収集
  - パラメータは、m, g, k, v
- モデリング
  - 運動方程式は、

$$m\frac{dx}{dt} = mg - kv$$



常微分方程式の数値解法

- 運動方程式を数値的に解くこととする。
- 常微分方程式の数値解法にはいくつかの方法がある
  - オイラー法
  - ルンゲクッタ法

Euler法

微分方程式を$\frac{dx}{dt} = f(x,t)$とし、初期値を$x(0) = x_0$とする。

時刻tにおける微分を微小時間Δtにおいて以下のように近似する。

$$\frac{dx}{dt} = \frac{x(t+\Delta t) - x(x,t)}{\Delta t}$$

したがって、時刻t_iにおいて以下のように変形できる

$$x(t_{i+1}) = x(t_i) + \Delta t \times f(x,t_i)$$

なお、計算精度のオーダーはO(h)である。

Runge-Kutta法

Runge-Kutta法では、xの変化分Δxを以下のように定義して、Euler法と同様な計算を繰り返す。

$$\Delta x = \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}$$

$$k_1 = \Delta t \times f(x(t_i), t_i)$$
$$k_2 = \Delta t \times f(x(t_i) + k_1/2 +, t_i + h/2)$$
$$k_3 = \Delta t \times f(x(t_i) + k_2/2 +, t_i + h/2)$$
$$k_4 = \Delta t \times f(x(t_i) + k_3, t_i + h)$$

なお、計算精度のオーダーは$O(h^4)$である。

実験

- 以下の条件数値計算の実験を行う
- 抵抗のある落下運動
  - M=0.5
  - G=9.8
  - K=0.5
  - H=1,0.5,0.2,0.1
  - T=0のときx=0
- 実装はpythonを用いて行った。

結果

解析解は、$v = v_1 + v_2$として、

$$v_1 = -\frac{k}{m}v$$

$$v_2 = -\frac{k}{m}v + g$$

それぞれ解くことで、Cを積分定数とすると、

$$v_1 = Cexp(-\frac{k}{m}t)$$

$$v_2 = \frac{mg}{k}$$

結果

解は、

$$v = Cexp\left(-\frac{k}{m}t\right) + \frac{mg}{k}$$

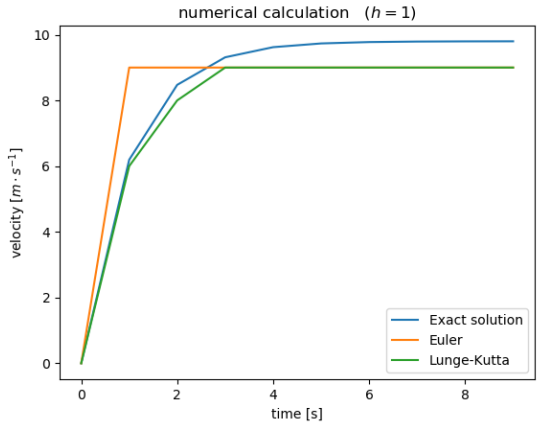
初期条件より、厳密解は

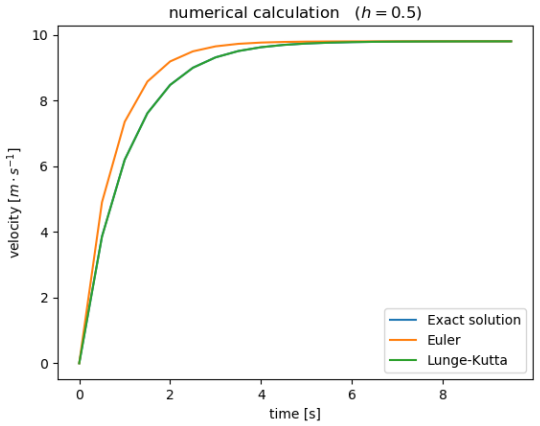
$$v = \frac{mg}{k}\left[1 - exp\left(-\frac{k}{m}t\right)\right]$$

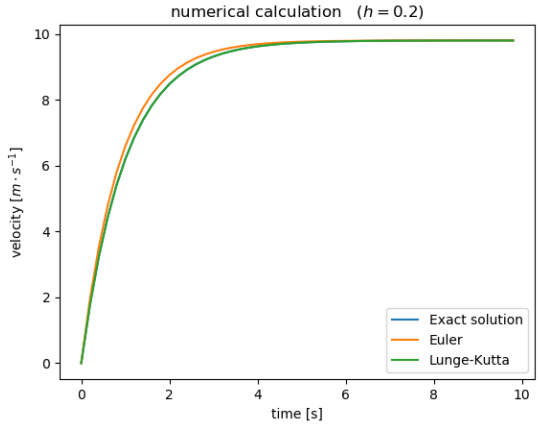
となる。数値を代入すると、

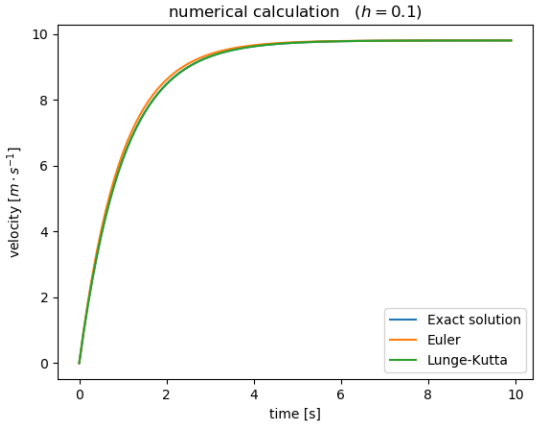
$$v = 9.8\left[1 - exp(-t)\right]$$

結果


numerical calculation $(h = 1)$
velocity $[m \cdot s^{-1}]$
time [s]
Exact solution
Euler
Lunge-Kutta


numerical calculation $(h = 0.5)$
velocity $[m \cdot s^{-1}]$
time [s]
Exact solution
Euler
Lunge-Kutta


numerical calculation $(h = 0.2)$
velocity $[m \cdot s^{-1}]$
time [s]
Exact solution
Euler
Lunge-Kutta


numerical calculation $(h = 0.1)$
velocity $[m \cdot s^{-1}]$
time [s]
Exact solution
Euler
Lunge-Kutta

考察と結論

- Runge-Kutta法のほうが精度がよい
  - 精度のオーダーが$O(h^4)$だから
  - 一方、Euler法の精度は$\mathrm{O(h)}$だから精度が悪い
- 1ステップあたりの計算はEuler法が簡単で、Runge-Kutta法は面倒
  - Eulear法は近似がより簡単だから

- Euler法は計算量は少ないが、精度は悪い
- Runge-Kutta法は計算量は多いが、精度は良い。

参考文献

1. 山田新一など, システム工学通論, コロナ社, 2001
2. 筑波大学, 物理学実験テキスト

シミュレーションの実例　その2

バネマスダンパー系のシミュレーション

2 階の常微分方程式

- 運動方程式は一般的に2 階の線形微分方程式である
- Euler法、Runge-Kutta法は1階の微分方程式の階法である

どうやって計算するのか？

計算方法

2階の常微分方程式

$$x'' = f(x, x', t)$$

があったとき、$x' = y$とおくと、

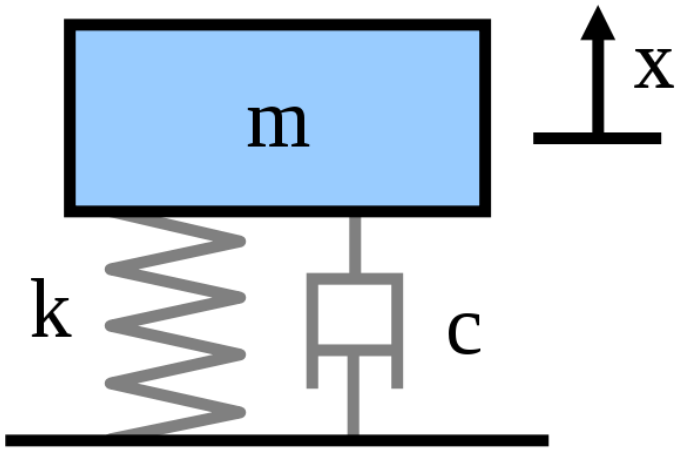
$$y' = f(x, y, t)$$

$$y = g(x, x', t)$$

連立1階の微分方程式になる

目的の明確化、データの収集

- バネマスダンパー系のシミュレーションを行う
- ダンパーの抵抗力は速度に比例する
- 変数は$x, \dot{x}$の2つ



https://ja.wikipedia.org/wiki/%E6%B8%9B%E8%A1%B0%E6%8C%AF%E5%8B%95#/media/File:Mass_spring_damper.svg

モデリング

運動方程式は、以下の通りである

$$m\ddot{x} + 2c\dot{x} + kx = 0$$

$$\boldsymbol{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

とすると、以下のように表すことができる

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & -2c/m \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

この連立微分方程式をRunge-Kutta法で計算する。

モデリング

$$A = \begin{bmatrix} 0 & 1 \\ -k/m & -2c/m \end{bmatrix}$$

とすると、

$$\dot{\boldsymbol{x}} = A\boldsymbol{x}$$

となるので

この連立微分方程式をRunge-Kutta法で計算する。

実験

- 以下の条件で実験を行った。
  - C=1
  - K=25
  - T=0のときx=0
  - T=0のときv=2
- Runge-Kutta法はpythonでプログラミング
  - h=0.1, 0.05で実行

結果

解を$e^{\rho t}$とすると、

$$e^{\rho t}(\rho^2 + 2\rho + 25) = 0$$

$$\rho = -1 \pm 2\sqrt{6}i$$

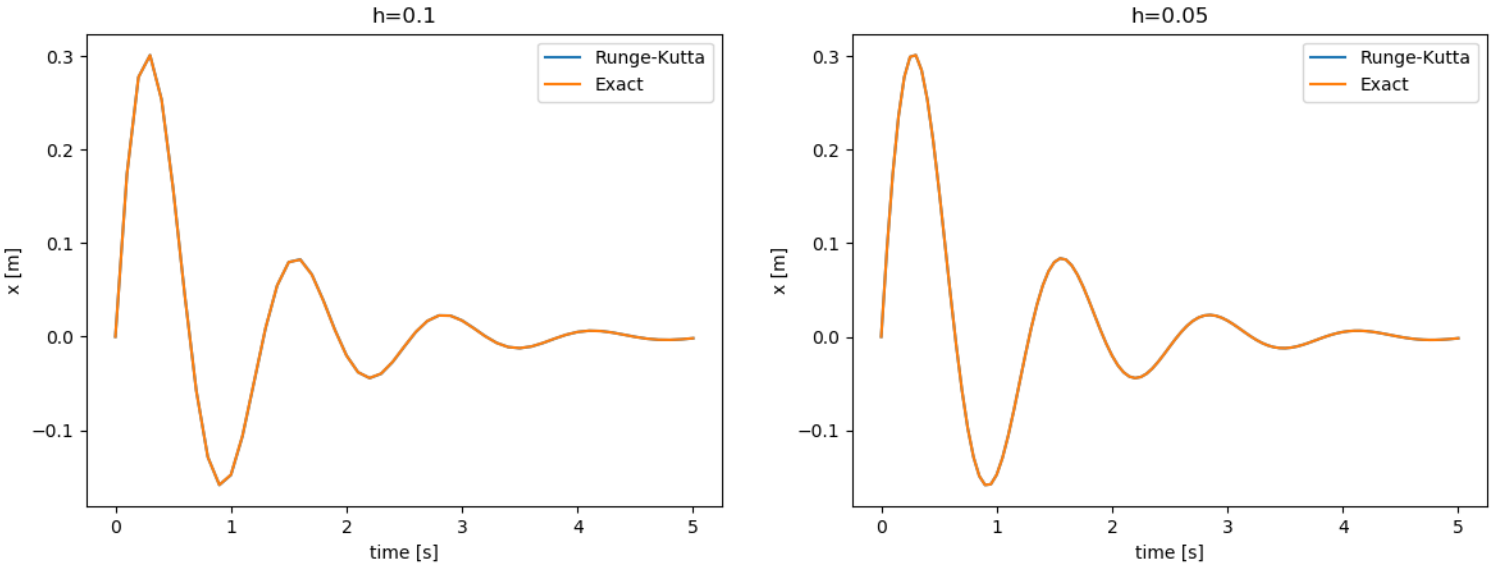
オイラーの公式$e^{ix} = cosx + isinx$より、

$$x = e^{-t}(C_1 cos2\sqrt{6}t + C_2 sin2\sqrt{6}t)$$

$x(0) = 0$より、C_1=0

$\dot{x(0)} = 0$より、$C_2 = \frac{1}{\sqrt{6}}$

結果


h=0.1
Runge-Kutta
Exact
x [m]
time [s]
h=0.05
Runge-Kutta
Exact
x [m]
time [s]

考察、余談

- 階数が高くなっても、連立微分方程式にすることで、数値的に解くことができるということが確認できた。

- 方程式が線形であって、行列Aが対角化可能ならば、解を簡単に求めることができます

シミュレーションの実例　その3

3次元での流体のシミュレーション

流れを支配する方程式

- ナビエ・ストークス方程式と連続方程式によって流体の動きは支配されている。
- 非常に複雑なので圧力をかけても密度は変化しないとみなす

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\nabla \frac{p}{\rho} + \upsilon \Delta v$$

$$\nabla \cdot u = 0$$

twitterより

ナビエ･ストークス方程式
半端ないって！

- アイツ半端ないって！
- 非線形の偏微分方程式だもん…。
- そんなん解けんやん普通…。

そもそも解が存在するかどうかさえわかっていない…。

某国の王妃様
Wikipediaより

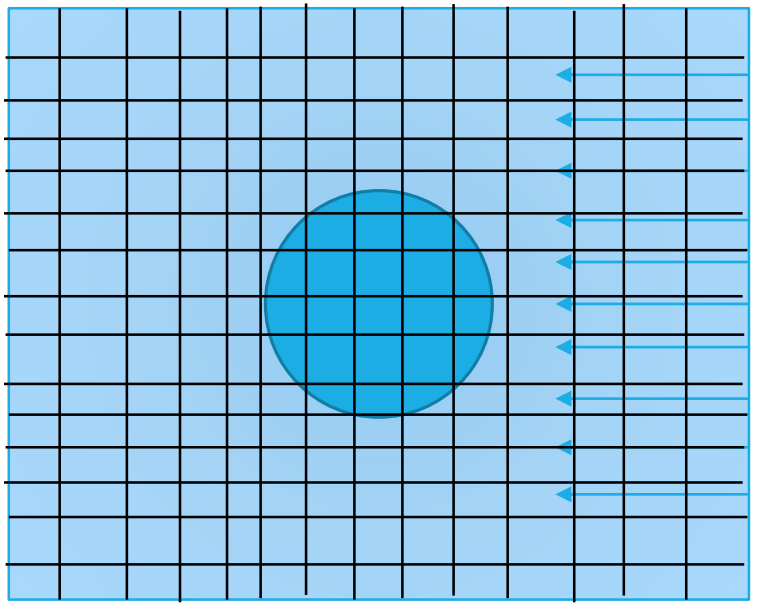
厳密解がわからなければ、
数値計算すればいいじゃない

目的を明確化,データの収集

- 完全でなくてもいいから流体の流れが知りたい!
- 空間は十分に細ければ連続でなくてもよい
- 空間と時間を離散化して、近似すればよい

モデリング

- 計算を離散化するために、空間をメッシュに分割する
- メッシュは細ければよいが、細かすぎると解析に時間がかかる
- メッシュはひずみがなく、直行しているとよいとされる



実行

- 複雑なので自前でプログラミングするのが難しい
- そして、計算量が多いので最適化しないと厳しい
- 一般的には既成品のソフトが多く用いられている。

From ANSYS社

OpenFOAM

The Open Source CFD Toolbox

From OpenFOAM.com

参考文献

1. 大槻義彦,大場一郎,新・物理学辞典,講談社,2018
2. 渡辺尚貴,流体力学,http://www-cms.phys.s.u-tokyo.ac.jp/~naoki/CIPINTRO/CIP/fluid.html, 2018/7/7閲覧
3. CYBERNET,はじめてみよう！流体解析（入門編）[Ⅰ], http://www.cybernet.co.jp/ansys/case/lesson/003.html , 2018/7/7閲覧

ご清聴ありがとうございました

Q