シミュレーション物理4

運動方程式の方法

運動方程式

- 物理で最もよく出てくる。そもそも物理はものの運動を議論する学問から出発（つり合いは運動を行わないという意味で含まれる）
- 代表例
  - ニュートンの運動方程式
  - 波動方程式
  - シュレーディンガー方程式

運動方程式（微分方程式の解法）

- 高次の微分方程式を1階微分方程式に変形
  - N変数の2階微分方程式→2N変数の1階微分方程式
  - dy/dt=f(t,y)を考察（yはベクトルでもよいが説明のため，1次元で考える）

運動方程式の解き方1

- 最も原始的なとき（オイラー法）
  - 時間を離散的に0, Δt, 2Δt, 3 Δt …と刻む。
  - $y(n\Delta t)=y_n$として

$$\frac{dy}{dt} = f \rightarrow y_{n+1} = y_n + f(t_n, y_n)\Delta t$$

運動方程式の解き方2；オイラー法の改良

- オイラー法だと精度が悪い。1タイムステップでΔt^2の誤差。目標がt秒後の粒子の位置だとすると，N=t/Δt回のステップが必要。するとこの誤差が蓄積して，tΔt程度の誤差が生じる。
- オイラー法を改良してΔt^3の誤差しか生じないようにする。そのためには？

2次のRunge-Kutta法（中点法）

$$k_1 = \Delta t \; f(t_n, y_n)$$

$$k_2 = \Delta t \; f(t_n + \frac{\Delta t}{2}, y_n + \frac{k_1}{2})$$

$$y_{n+1} = y_n + k_2 + O(\Delta t^3)$$

直感的な意味；yの時間変化を決めるfがt，yに依存している。
そこでfはtとt+Δt,yとy+Δyの中点で決めるとよい。

2次のRunge-Kutta法の証明

$$y_{n+1} = y(t_n + \Delta t)$$

$$= y(t_n) + \frac{dy}{dt}\Delta t + \frac{1}{2}\frac{d^2y}{dt^2}\Delta t^2 + O(\Delta t^3)$$

←通常のテイラー展開

$$= y_n + f\Delta t + \frac{1}{2}(f_t + f_y f)\Delta t^2 + O(\Delta t^3)$$

$$(\because \frac{dy}{dt} = f(t, y(t)), \frac{d^2y}{dt^2} = f_t + f_y\frac{dy}{dt} = f_t + f_y f)$$

Runge-Kutta法→

$$y_{n+1} = y_n + \Delta t\, f(t_n + \Delta t/2, y_n + k_1/2)$$

$$= y_n + \Delta t\, f(t_n + \Delta t/2, y_n + \Delta t\, f/2)$$

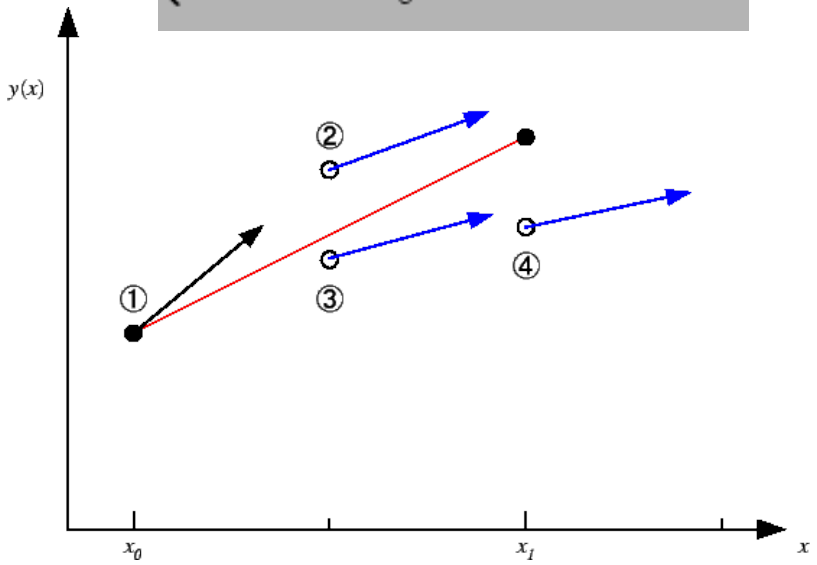
$$= y_n + \Delta t\left[f + f_t\frac{\Delta t}{2} + f_y\frac{\Delta t\, f}{2}\right] + O(\Delta t^3)$$

$$= y_n + \Delta t f + \frac{\Delta t^2}{2}(f_t + f_y f) + O(\Delta t^3)$$

4次のRunge-Kutta法

t→xとしてx，yで図で解釈

$$\begin{cases} k_1 = hf(x_n, y_n) \\ k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\ k_4 = hf(x_n + h, y_n + k_3) \\ y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases}$$



4th Order Runge-Kutta法

$$k_1 = \Delta t\ f(t_n, y_n)$$
$$k_2 = \Delta t\ f(t_n + \Delta t/2, y_n + k_1/2)$$
$$k_3 = \Delta t\ f(t_n + \Delta t/2, y_n + k_2/2)$$
$$k_4 = \Delta t\ f(t_n + \Delta t, y_n + k_3)$$
$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(\Delta t^5)$$

多くの場合，力は時間にはあらわに依存しない。

$$k_1 = \Delta t\ f(y_n)$$
$$k_2 = \Delta t\ f(y_n + k_1/2)$$
$$k_3 = \Delta t\ f(y_n + k_2/2)$$
$$k_4 = \Delta t\ f(y_n + k_3)$$
$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(\Delta t^5)$$

レポート課題：2番目の場合について，証明せよ

問題

- まずは自由落下から。t=0,y=0で初速0でものを落とした場合を，
  - Euler法
  - 中点法
  - Runge-Kutta法

  で解くこと。

  次に空気抵抗がある場合をとく。空気抵抗は速度に比例。

無次元化

$$m\frac{d^2x}{dt^2} = -mg - m\gamma\frac{dx}{dt} \rightarrow$$

$$\begin{cases} \dfrac{dv}{dt} = -g - \gamma v \\ \dfrac{dx}{dt} = v \end{cases} \rightarrow x_0, t_0, v_0 = x_0 / t_0, g_0 = x_0 / t_0^{\,2}, \gamma_0 = 1/t_0$$

$$\begin{cases} \dfrac{d\tilde{v}}{d\tilde{t}} = -\tilde{g} - \tilde{\gamma}\tilde{v} \\ \dfrac{d\tilde{x}}{d\tilde{t}} = \tilde{v} \end{cases}$$

ここでは1秒、1メートルを基本単位とする。

課題:それぞれを数値的にといて解析的な式と比較しよう

$$v = -\frac{g}{\gamma}(1 - e^{-\gamma t}), x = \frac{g}{\gamma^2}(1 - e^{-\gamma t} - t\gamma)$$

```
program euler
!------------------------
!  This is a program to investigate the free fall
! Euler method
!  2005/5/2  Written by T. Ohtsuki
!------------------------
implicit none  ! Always begin with this statement
integer,parameter::double=selected_real_kind(14) !  Type defined
real(kind=double), parameter::zero=0.0_double,one=1.0_double,&
half=0.5_double ! Parameter defined
real(kind=double), parameter::pi=3.1415926535897932 ! Parameter defined
real(kind=double)::x,vx,deltat,xnew,vxnew,t,tmax,analytic ! Real variables defined
real(kind=double), parameter::g=9.8_double,gamma=1.0_double ! Parameter defined
integer::i,nmax ! integer defined

deltat=0.05_double  ! Time interval

x=zero ! Initial position
vx=zero ! Initial velocity
tmax=5._double! Target time
nmax=int((tmax-deltat/2._double)/deltat)+1! Number of iteration required

do i=1,nmax  ! Equation of motion
   vxnew=vx-g*deltat-gamma*vx*deltat ! Vx is slightly changed
   xnew=x+vx*deltat ! X is slightly changed
   vx=vxnew  ! Set vxnew as vx
   x=xnew  ! Set xnew as x
end do

!Compare the analytic and numerical results
   tmax=deltat*nmax
   analytic=-g*tmax/gamma-g/gamma**2*exp(-gamma*tmax)+g/gamma**2
   print *,tmax,x,analytic

stop
end
```

```
program midpoint
!------------------------
!  This is a program to investigate the free fall
!  2005/5/2  Written by T. Ohtsuki
!  midpoint method
!------------------------
implicit none  ! Always begin with this statement
integer,parameter::double=selected_real_kind(1
4)
real(kind=double), parameter::zero=0.0_double,
one=1.0_double,half=0.5_double
real(kind=double), parameter::pi=3.1415926535
897932
real(kind=double), parameter::g=9.8_double,ga
mma=1._double
real(kind=double)::x,vx,deltat,xnew,vxnew,t,tma
x,analytic
real(kind=double)::xk1,xk2,vxk1,vxk2
integer::i,nmax

deltat=0.05_double
x=zero
vx=zero
tmax=5._double
nmax=int((tmax-deltat/2._double)/deltat)+1
```

```
do i=1,nmax  !ここだけEuler法とちょっと違う
   vxk1=-deltat*(g+gamma*vx)
   xk1=deltat*vx
   vxk2=-deltat*(g+gamma*(vx+half*vxk1))
   xk2=deltat*(vx+half*vxk1)
   vxnew=vx+vxk2
   xnew=x+xk2
   vx=vxnew
   x=xnew
end do

!Compare the analytic and numerical results
tmax=deltat*nmax
analytic=-g*tmax/gamma-g/gamma**2*exp(-
gamma*tmax)+g/gamma**2
print *,tmax,x,analytic

stop
end
```

- コンパイル
  - f90 filename(必ず.f90で終わるファイル)
  - a.outというファイルができるのでそれを実行（a.outと打ち込む）
  - もしa.outでなく、たとえばEuler1という名前の実行ファイル（キーボードで打ち込むと結果が出るものを実行ファイルという）がほしければ
    - f90 -o Euler1 finename