シミュレーション物理6

運動方程式の方法:
惑星の軌道

出席のメール(件名に学生番号と氏名)に，中点法をサブルーチンを使って書いたプログラムを添付

今回の授業の目的

- 4次のRunge-Kutta法を用いて，惑星の軌道のシミュレーションを行う

$$m\frac{d^2\vec{r}}{dt^2} = -\frac{GMm}{r^3}\vec{r}$$

$$\rightarrow \frac{d^2\vec{r}}{dt^2} = -\frac{GM}{r^3}\vec{r} \rightarrow 2D(\text{using angular momentum cons.})$$

$$\rightarrow r/l = \tilde{r}, t/T = \tilde{t}, \frac{d^2\vec{\tilde{r}}}{d\tilde{t}^2} = -\left(\frac{GMT^2}{l^3}\right)\frac{1}{\tilde{r}^3}\vec{\tilde{r}}$$

$$\rightarrow \text{set } \frac{GMT^2}{l^3} = 1$$

$$\rightarrow \frac{d^2\vec{\tilde{r}}}{d\tilde{t}^2} = -\frac{1}{\tilde{r}^3}\vec{\tilde{r}}, \vec{\tilde{r}} = (\tilde{x}, \tilde{y})$$

rk4.f90

- 4次のルンゲークッタ法

  Δt^4まで正しい。

  - 高精度の数値計算に向いている。
  - これ以上次数を上げると，表式が複雑になりかえって計算時間がかかる。

自由落下のプログラムをrk4.f90を使うよう改良

```
program freefall
!------------------------
!  This is a program to investigate the free fall
!  2005/5/12  Written by T. Ohtsuki
!  Runge-Kutta method, subroutine used
!  derivs is used as a subprogram, using "contains".
!  This makes parameters in the procedure derivs global variables
!------------------------
implicit none  ! Always begin with this statement
integer,parameter::double=selected_real_kind(14)
real(kind=double), parameter::zero=0.0_double,one=1.0_double,half=0.5_double
real(kind=double),parameter::pi=3.1415926535897932
integer,parameter::nd=2  ! dimension of the equation
real(kind=double),dimension(nd)::y,dydx,yout  !位置，速度はベクトルで定義
real(kind=double)::deltat,t,tmax,analytic
integer::i,nmax
real(kind=double),parameter::g=9.8_double,gamma=1.0_double
external:: derivsfreefall

!---------main program---------
deltat=0.05_double

t=zero
y(1)=zero  ! initial velocity，初期値も当然変わる
y(2)=zero  ! initial position
```

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

do i=1,nmax
   call derivsfreefall(nd,t,y,dydx) ! 微分係数を求める
   call rk4(nd,y,dydx,t,deltat,yout,derivsfreefall) !微分係数を使ってΔtだけ時間発展
   y=yout
end do

tmax=deltat*nmax
analytic=-g*tmax/gamma-g/gamma**2*exp(-gamma*tmax)+g/gamma**2 !ここら辺もいらない
print *,tmax,y(2),analytic
end program freefall
```

微分を求めるサブルーチン, derivsfreefall.f90として保存

```
!------------------  Subroutines --------------------
SUBROUTINE derivsfreefall(nd,x,y,dydx)
!------------------------
!   subroutine derivs
!   nd : dimension of y
!   x : time (usually time independent)
!   dydx: first derivative w.r.t. time, not a partial derivative
!------------------------
IMPLICIT NONE
integer,parameter::double=selected_real_kind(14)
integer::nd
REAL(kind=double), INTENT(IN) :: x
REAL(kind=double), DIMENSION(nd), INTENT(IN) :: y
REAL(kind=double), DIMENSION(nd), INTENT(OUT) :: dydx
real(kind=double),parameter::g=9.8_double,gamma=1.0_double

dydx(1)=-(g+gamma*y(1)) !g, gamma are global values！ここは当然変える
dydx(2)=y(1)
END SUBROUTINE derivsfreefall
```

これは非常に汎用性が高い。
rk4.f90として保存

```
SUBROUTINE rk4(nd,y,dydx,x,h,yout,derivs)
!------------------------
!  This is a Runge-Kutta subroutine
!  2005/5/11  Written by T. Ohtsuki
!------------------------
IMPLICIT NONE
integer,parameter::double=selected_real_kind(14)
integer::nd
REAL(kind=double), DIMENSION(nd), INTENT(IN) :: y,dydx
REAL(kind=double), INTENT(IN) :: x,h
REAL(kind=double), DIMENSION(nd), INTENT(OUT) :: yout
INTERFACE
  SUBROUTINE derivs(nd,x,y,dydx)
    IMPLICIT NONE
    integer,parameter::double=selected_real_kind(14)
    integer::nd
    REAL(kind=double), INTENT(IN) :: x
    REAL(kind=double), DIMENSION(nd), INTENT(IN) :: y
    REAL(kind=double), DIMENSION(nd), INTENT(OUT) :: dydx
  END SUBROUTINE derivs
END INTERFACE
REAL(kind=double) :: h6,hh,xh
REAL(kind=double), DIMENSION(size(y)) :: dym,dyt,yt

hh=h*0.5_double
h6=h/6.0_double
xh=x+hh
yt=y+hh*dydx
call derivs(nd,xh,yt,dyt)
yt=y+hh*dyt
call derivs(nd,xh,yt,dym)
yt=y+h*dym
dym=dyt+dym
call derivs(nd,x+h,yt,dyt)
yout=y+h6*(dydx+dyt+2.0_double*dym)
END SUBROUTINE rk4
```

- dahlmanにアップロード,
- f90 –c derivsfreefall.f90
- f90 –c rk4.f90
- f90 freefallrk4.f90 derivsfreefall.o rk4.o
- a.out
- 精度が非常によいことを確認する

自由落下のプログラムを惑星の軌道を求めるプログラムに変更

```
program freefall ！Orbitとでもする
!------------------------
!  This is a program to investigate the free fall
!  2005/5/12  Written by T. Ohtsuki
!  Runge-Kutta method, subroutine used
!  derivs is used as a subprogram, using "contains".
!  This makes parameters in the procedure derivs global variables
!------------------------
implicit none  ! Always begin with this statement
integer,parameter::double=selected_real_kind(14)
real(kind=double), parameter::zero=0.0_double,one=1.0_double,half=0.5_double
real(kind=double),parameter::pi=3.1415926535897932
integer,parameter::nd=2  ! dimension of the equation!, 直線運動でなく面内の運動なのでnd=2x2=4
real(kind=double),dimension(nd)::y,dydx,yout
real(kind=double)::deltat,t,tmax,analytic
integer::i,nmax
real(kind=double),parameter::g=9.8_double,gamma=1.0_double, これはいらなくなる
External:: derivsfreefall  ！これはderivsplanetに変更。
!---------main program---------
deltat=0.05_double

t=zero
y(1)=zero  ! initial velocity, 初期値も当然変わる
y(2)=zero  ! initial position
```

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

do i=1,nmax ！ここは全く変えなくてよい！
   call derivs(nd,t,y,dydx)
   call rk4(nd,y,dydx,t,deltat,yout,derivs)
   y=yout
end do

tmax=deltat*nmax
analytic=-g*tmax/gamma-g/gamma**2*exp(-gamma*tmax)+g/gamma**2 !ここら辺もいらない
print *,tmax,y(2),analytic


end program freefall
```

プログラムのチェックは?

エネルギーの保存則: $E=\frac{m}{2}(v_x{}^2+v_y{}^2)-\frac{GmM}{\sqrt{x^2+y^2}}=\frac{ml^2}{T^2}\left(\frac{\tilde{v}_x^2+\tilde{v}_y^2}{2}-\frac{1}{\sqrt{\tilde{x}^2+\tilde{y}^2}}\right)$

よって $\frac{\tilde{v}_x^2+\tilde{v}_y^2}{2}-\frac{1}{\sqrt{\tilde{x}^2+\tilde{y}^2}}$ が一定である必要がある

角運動量保存則: $L_z=m(xv_y-yv_x)=m\frac{l^2}{T}(\tilde{\tilde{x}}\tilde{v}_y-\tilde{\tilde{y}}\tilde{v}_x)$

よって $\tilde{\tilde{x}}\tilde{v}_y-\tilde{\tilde{y}}\tilde{v}_x$ が一定である必要がある

変数

- v_x,v_y,x,yをy(1),y(2),y(3),y(4)とする。順番は別にどうでもよいが，x, v_x,y, v_yというように位置と速さが混ざると粒子が増えてきたとき，面倒になる場合がある。

入出力

これまでは，画面からパラメータを入力して，画面に結果を書かせていた。
Read *,x
Print *,x
など。これだとあとで軌道を描かせたりするのが面倒

open，close, read, writeを使う

```
open(1,file="planetorbit.txt" )
Write(1,'(e14.7)')pi**3
Write(1,'(f14.7)')pi**3
close(1)

Open(1,file="planetorbit.txt")
Read(1,*) x
Close(1)
```

！ ファイルplanetorbit.txtを1番の出力先として指定
！ 別に1でなくても2でも10でも構わない。
！複数のファイルをあけて，xは1に，yは10に書くのも可能
！ '(・・)'で書き方を指定。これは指数を使うやり方
！ これは指数を使わないやり方。
!14個の幅を使い，有効数字7桁という意味
!例えば' (i5)'なら5桁の整数を書く。答えが3桁なら2個空白となる。6桁以上だとX

できあがったプログラムを見てみよう

```
program planetorbit
!------------------------
!  This is a program to analyze the planetary motion
!  2005/5/19  Written by T. Ohtsuki
!  Runge-Kutta method, subroutine used
!  derivs is used as a subprogram, using "contains".
!------------------------
use ConservationLaw !保存則をチェックするモジュールを使う
implicit none  ! Always begin with this statement
integer,parameter::double=selected_real_kind(14)
real(kind=double),
parameter::zero=0.0_double,one=1.0_double,half=0.5_double
real(kind=double),parameter::pi=3.1415926535897932
integer,parameter::nd=4  ! dimension of the equation
real(kind=double),dimension(nd)::y,dydx,yout
real(kind=double)::deltat,t,tmax,energy,AngularMomentum
integer::i,nmax
external:: derivsplanet
!---------main program---------
deltat=0.05_double
tmax=5._double
nmax=int((tmax-deltat/2._double)/deltat)+1
```

```
t=zero
!  y(1):vx,y(2)=vy,y(3)=x,y(4)=y
y(1)=one   ! initial x-velocity
y(2)=zero  ! initial y-velocity
y(3)=zero  ! initial x-position
y(4)=one   ! initial y-position

call ConservedVariables(nd,y,energy,AngularMomentum)

open(1,file='planetorbit.txt')

write(1,'(5f14.7)')t,y(3),y(4),energy,AngularMomentum

!--------------------------------
! beginning of the time evolution
!--------------------------------
do i=1,nmax
   call derivsplanet(nd,t,y,dydx)
   call rk4(nd,y,dydx,t,deltat,yout,derivsplanet)
   y=yout
   t=t+deltat
   call ConservedVariables(nd,y,energy,AngularMomentum)
   write(1,'(5f14.7)')t,y(3),y(4),energy,AngularMomentum
end do
close(1)
end program planetorbit
```

これをderivsplanet.f90として保存

```
!------------------  Subroutines --------------------
SUBROUTINE derivsplanet(nd,x,y,dydx)
!------------------------
!   subroutine derivsplanet
!   nd : dimension of y
!   x : time (usually time independent)
!   dydx: first derivative w.r.t. time, not a partial derivative
!------------------------
IMPLICIT NONE
integer,parameter::double=selected_real_kind(14)
integer::nd
REAL(kind=double), INTENT(IN) :: x
REAL(kind=double), DIMENSION(nd), INTENT(IN) :: y
REAL(kind=double), DIMENSION(nd), INTENT(OUT) :: dydx

dydx(1)=-y(3)/sqrt(y(3)**2+y(4)**2)**3
dydx(2)=-y(4)/sqrt(y(3)**2+y(4)**2)**3
dydx(3)=y(1)
dydx(4)=y(2)
END SUBROUTINE derivsplanet
```

保存則はモジュールで計算，conservationlaw.f90というファイルを用意

```
module ConservationLaw

implicit none
contains

subroutine ConservedVariables(nd,y,energy,AngularMomentum)
!------------------------
!   subroutine for examining the conserved quantity
!------------------------
IMPLICIT NONE
integer::nd
integer,parameter::double=selected_real_kind(14)
REAL(kind=double), dimension(nd),INTENT(IN) ::y
REAL(kind=double), INTENT(OUT)::energy,AngularMomentum

energy=(y(1)**2+y(2)**2)/2._double-1._double/sqrt(y(3)**2+y(4)**2)
AngularMomentum=y(3)*y(2)-y(4)*y(1)

END subroutine ConservedVariables


end module ConservationLaw
```

実行のさせ方

- まずモジュールをコンパイル

  f90 –c conservationlaw.f90

- 次にrk4.f90，derivsplanet.f90をコンパイル（申してある場合はよい）

  f90 –c rk4.f90

  f90 –c derivsplanet.f90

- 最後に実行ファイルを作る

  f90 planetrk4.f90 conservationlaw.o rk4.o derivsplanet.o

  実行ファイル名を例えばorbit.outにしたければ

  f90 –o orbit.out planetrk4.f90 conservationlaw.o rk4.o derivsplanet.o

課題

- いろいろな初期条件で，軌道を描かせる。
- 軌道はffftpでダウンロードし，エクセルで描かせる
- 来週はフォローアップ，今週で来た人は来なくてよい。出席は取りません。