1 常微分方程式のシンプレクティック解法

1.1 全般的注意

前項で紹介した Euler 法や Runge-Kutta 法は標準的な数値解法であるが欠点もある。特にエネルギーの保存等の拘束条件が課されている場合には見掛けの励起、減衰が起こる。ここでは吉田春夫「力学」(岩波現代の物理 1 第 2 刷) 等に習って最近の保存力学系の標準解法となっている Symplectic 数値解法を紹介してみよう。[1]

例としてハミルトニアン

$$H = \frac{1}{2}(p^2 + q^2) \tag{1}$$

で記述される調和振動子を考える (p, q:運動量、座標)。運動方程式

$$\dot{q} = p, \quad \dot{p} = -q \tag{2}$$

の厳密解は

$$\begin{pmatrix} q(\tau) \\ p(\tau) \end{pmatrix} = \begin{pmatrix} \cos\tau & \sin\tau \\ -\sin\tau & \cos\tau \end{pmatrix} \begin{pmatrix} q(0) \\ p(0) \end{pmatrix} \tag{3}$$

である。ここで τ は時間であるが同時に位相空間での回転角を表している。

Euler 法は (3) 式の $O(\tau)$ の近似であるから

$$\begin{pmatrix} q(\tau) \\ p(\tau) \end{pmatrix} = \begin{pmatrix} 1 & \tau \\ -\tau & 1 \end{pmatrix} \begin{pmatrix} q(0) \\ p(0) \end{pmatrix} \tag{4}$$

となっている。(4) 式を $(q, p) \to (q', p')$ の写像と考えると各ステップ毎にエネルギーは

$$H' = \frac{1}{2}(p'^2 + q'^2) = (1 + \tau^2)H = (1 + \tau^2)\frac{1}{2}(p^2 + q^2) \tag{5}$$

となり、単調増加を起こす。

一方、Runge-Kutta 法では 4 次精度なので

$$\begin{pmatrix} q' \\ p' \end{pmatrix} = \begin{pmatrix} 1 - \frac{\tau^2}{2} + \frac{1}{24}\tau^4 & \tau - \frac{1}{6}\tau^3 \\ -\tau + \frac{1}{6}\tau^3 & 1 - \frac{1}{2}\tau^2 + \frac{1}{24}\tau^4 \end{pmatrix} \begin{pmatrix} q \\ p \end{pmatrix} \tag{6}$$

となるのでエネルギーの単調減少

$$p'^2 + q'^2 = (1 - \frac{1}{72}\tau^6 + \cdots)(p^2 + q^2) \tag{7}$$

をもたらす。

[1] 他に Arnold, Mathematical Methods of Classical Mechanics 2nd Edition (Springer, Berlin, 1990), 山本義隆、中村孔一、解析力学 1 (朝倉 1998) 等が参考になる。特に後者は緻密ですっきりとした論理体系をなしているので参考になるであろう。

1.2 シンプレクティック数値解法

証明は省略するが正準変数の正しい変換である正準変換はシンプレクティック性を持つ事が知られている。ここでは２自由度に限定して話をする。一般の場合の証明は既に挙げた書籍を通して学習されたい。一般に正準方程式は

$$\dot{q} = \frac{\partial H}{\partial p}, \quad \dot{p} = -\frac{\partial H}{\partial q} \tag{8}$$

で表される。ここで変数をまとめて

$$\frac{\partial z}{\partial t} = \tilde{\Omega}\frac{\partial H}{\partial z} \tag{9}$$

と書いてみよう。ここで

$$z = \begin{pmatrix} q \\ p \end{pmatrix}, \quad \tilde{\Omega} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{10}$$

である。(9) 式は次の様に書き換える事も可能である。

$$\Omega\frac{\partial z}{\partial t} = \frac{\partial H}{\partial z}, \quad \Omega = \tilde{\Omega}^{-1} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \tag{11}$$

さてシンプレクティック性を持つとはある行列 M が

$$^{T}M\Omega M = \Omega \tag{12}$$

を充たす事を言う。但し ^{T}M は M の転置行列である。２次元の場合 (12) は

$$detM = 1 \tag{13}$$

と等価であることは容易に確かめられる。解析力学のどの教科書を見ても正準変換の変換行列式 $(q', p')/(q, p)$ が１であることは証明されているのでシンプレクティック変換と正準変換は等価であろうことは想像がつく。果して一般の場合に正準変換とシンプレクティック変換は必要十分であることは証明できる。

ポテンシャル場における質点の運動を記述するハミルトニアン

$$H = T(p) + V(q) \tag{14}$$

を考えよう。ここで第１項は運動エネルギー、第２項は位置のみに依るポテンシャルである。この場合の陽的なシンプレクティック法が可能であり、そのエッセンスは $H = T(p)$ と $H = V(q)$ という２つのハミルトニアンの合成系として考える事が出来ることにある。

$H = T(p)$ の厳密解は

$$q' = q + \tau\frac{\partial T}{\partial p}, \quad p' = p \tag{15}$$

で与えられ、$H = V(q)$ の厳密解は

$$q' = q, \quad p' = p - \tau\frac{\partial V}{\partial q} \tag{16}$$

で与えられ、それぞれが相空間上の直線に対応している。これらは正準変換なのでシンプレクティック性を持ち、その合成もその性質を保持している。実際、これらの式から定義される変換行列式 $M = (q', p')/(q, p)$ は $detM = 1$ と $^{T}M\Omega M = \Omega$ を充たす。またこれらを合成した

$$q' = q + \tau\frac{\partial T}{\partial p} \quad p' = p - \tau\frac{\partial V}{\partial q} \tag{17}$$

は、やはりシンプレクティック性を保つので１次のシンプレクティック解法になっている。

こういう陽的なシンプレクティック解法は Lie 代数的な定式化によって系統的に求める事が可能である。Hamilton 方程式を $z=(q,p)$ として

$$\dot{z}=\{z,H\},\qquad \{A,B\}\equiv\frac{\partial A}{\partial q}\frac{\partial B}{\partial p}-\frac{\partial A}{\partial p}\frac{\partial B}{\partial q} \tag{18}$$

と表す事が可能である。ここで $\{A,B\}$ は Poisson 括弧である。今線形微分作用素 L_H を

$$L_H\equiv\{z,H\} \tag{19}$$

で定義すると (18) の形式解は

$$z(\tau)=\exp[\tau L_H]z(0) \tag{20}$$

と書ける。特に (14) のように変数が分離しているハミルトニアンでは $L_H=L_T+L_V$ が成立しており、形式解は

$$z(\tau)=\exp[\tau(L_T+L_V)]z(0) \tag{21}$$

となる。ここで作用素 L_T,L_V は一般には可換ではない。このとき演算子をそれぞれの指数の積で近似できたら便利である。というのは個々が

$$\exp[\tau L_T]=\sum_{n=0}^{\infty}\frac{\tau^n}{n!}L_T^n,\quad \exp[\tau L_V]=\sum_{n=0}^{\infty}\frac{\tau^n}{n!}L_V^n \tag{22}$$

という式を見れば分かる通りにシンプレクティック性を持つ. 従ってその積もシンプレクレティック演算子になるからである。こうした考えに基づくと

$$\exp[\tau(L_T+L_V)]=e^{\tau L_T}e^{\tau L_V}+O(\tau^2) \tag{23}$$

となることがすぐ分かるし、次の近似では

$$\exp[\tau(L_T+L_V)]=\exp[\frac{\tau}{2}L_T]\exp[\tau L_V]\exp[\frac{\tau}{2}L_T]+O(\tau^3) \tag{24}$$

となることが分かる。

３次は飛ばして４次のシンプレクティック分解を考えてみよう。ここで $\exp[A+B]$ を $\exp[A]$,$\exp[B]$ 等の積で近似することが目的になる。但し以下では $A=\tau L_T$, $B=\tau L_V$ としよう。そのために $A+B$ の自分自身との可換性を利用し、恒等式

$$\exp[A+B]=\exp[s(A+B)]\exp[(1-2s)(A+B)]\exp[s(A+B)] \tag{25}$$

を用いてみる。ここで Taylor 展開によって

$$\exp[s(A+B)]=1+s(A+B)+\frac{s^2}{2}(A+B)^2+\frac{s^3}{6}(A+B)^3+O((A+B)^4) \tag{26}$$

と展開できる。(25) の右辺をそれぞれ展開して整理すると３次の項は

$$\frac{1}{6}(2s^3+(1-2s)^3)(A+B)^3 \tag{27}$$

となることが容易に分かる。ここで係数が 0 であれば３次の寄与がなくなる。即ち

$$2s^3+(1-2s)^3=0\Leftrightarrow s=\frac{1}{2-2^{1/3}} \tag{28}$$

が成り立つと都合が良い。このまま単純に代入しても２次までの精度が保証されないので２次までの結果 (24) と合成することを考えてみる。つまり

$$S_2[A+B] \equiv \exp[\frac{1}{2}A]\exp[B]\exp[\frac{1}{2}A] \tag{29}$$

として

$$\exp[A+B] = S_2[s(A+B)]S_2[(1-2s)(A+B)]S_2[s(A+B)] + O((A+B)^4) \tag{30}$$

とすることで４次の公式が得られる。

1.3 シンプレクティック解法の利点

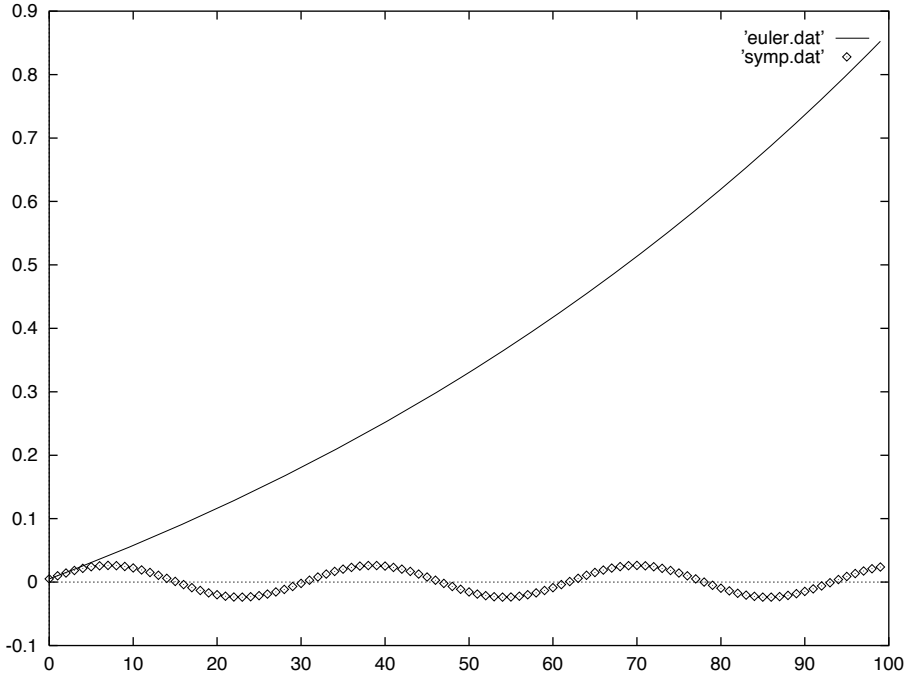

図 1: 調和振動子の Euler 法 (実線) と１次の陽的シンプレクティック法 (データ) のエネルギー誤差。ステップサイズは共に 0.1.

シンプレクティック解法の数値的な利点は Runge-Kutta 法等の様にエネルギーの単調な変化をもたらさないことにある。例えば１次のシンプレクティック法を (1) に適用すると (17) 式から容易に

$$\begin{pmatrix} q' \\ p' \end{pmatrix} = \begin{pmatrix} 1 & \tau \\ -\tau & 1-\tau^2 \end{pmatrix} \begin{pmatrix} q \\ p \end{pmatrix} \tag{31}$$

となる。この写像を繰り返す事でエネルギーは決して単調に変化しないで振動する。実はこの写像 (31) は厳密な保存量

$$\tilde{H} = H + \tau H_1 = \frac{1}{2}(p^2+q^2) + \frac{\tau}{2}pq \tag{32}$$

を持つ事は容易に確かめられる。この保存量のために数値解は (q,p) 面内の楕円上に拘束される。一般の n 次シンプレクティック解法では

$$\tilde{H} = H + \tau^n H_n + \cdots = const. \tag{33}$$

の形の保存量が存在することが分かっておりエネルギー誤差の単調な変化は起こり得ないのである。

さて実際にシンプレクティック法の利点を確かめてみよう。(31) 式を順次求めるには、例えば

```
real dt,tf,p,q,e0,ene
integer i,n

dt=0.1
tf=10.0
p=0.0
q=1.0
e0=1./2.
n=int(tf/dt)

do i=1,n
  qn=q+dt*p
  pn=-dt*q+(1.-dt*dt)*p
  q=qn
  p=pn
  ene=(q*q+p*p)/2.
  write(*,*) i,ene-e0
enddo
end
```

とでもプログラムを組めばいい。意味はいちいち書かなくても自明である。また Fortorn90 の自由形式、C では以下のように組めば良い。

```
real(8)::dt,tf,p,q,e0,ene
integer::i,n

dt=0.1_8
tf=10._8
p=0._8
q=1._8
e0=1._8/2._8
n=int(tf/dt)

do i=1,n
  qn=q+dt*p
  pn=-dt*q+(1._8-dt*dt)*p
  q=qn
  p=pn
```

```
   ene=(q*q+p*p)/2._8
   print*,i,ene-e0
 enddo
 end
```

C では、

```
#include<stdio.h>

main(){
  double dt,tf,p,q,e0,ene,qn,pn;
  int    i,n;

  dt=0.1;
  tf=10.0;
  p=0.0;
  q=1.0;
  e0=1.0/2.0;
  n=(int)(tf/dt);

  for(i=1;i<=n;i++){
    qn=q+dt*p;
    pn=-dt*q+(1.0-dt*dt)*p;
    q=qn; p=pn;
    ene=(q*q+p*p)/2.0;
    printf(" %3d  %15.10f \n",i,ene-e0);
  }
}
```

Euler 法の (4) 式は上のプログラムで dt*dt の項を落せばよい。その結果を比較したのが図１である。理論から予想される様に Euler 法では単調に誤差が大きく成る一方で Symplectic 法では誤差は振動している。

問 1 同じ問題を Runge-Kutta 法と２次、或は４次のシンプレクティック法で解き、エネルギー誤差を比較せよ。

2 多自由度系への拡張

シンプレクティック法は多自由度への拡張及びそれに伴うプログラミングも容易である。その容易さの理由は１段階の陽的解法であるがためである. 運動エネルギー $T(p)$, ポテンシャルエネルギー $V(q)$ に対して記号 $T_p \equiv \partial T/\partial p$,$V_q \equiv \partial V/\partial q$ を導入したら

$i = 1, 2, \cdots, n$ の順に

$$p(i) = p(i) - c_k dt V_q(i)$$

T_pの計算

$$q(i) = q(i) + d_k dt T_p(i)$$

V_qの計算

繰り返し

とするだけである。ここで c_k,d_k はシンプレクティック条件を充たす定数である。例えば 2 次であれば

$$c_1 = 1/2,\ c_2 = 1/2,\ d_1 = 1,\ d_2 = 0$$

であり、４次であれば

$$c_1 = v_0/2,\ c_2 = v_2,\ c_3 = v_2,\ c_4 = v_0/2,\ d_1 = v_0,\ d_2 = v_1,\ d_3 = v_0,\ d_4 = 0$$

である。但し

$$v_0 = \frac{1}{2 - 2^{1/3}}, \quad v_1 = -2^{1/3} v_0, \quad v_2 = (1 - 2^{1/3}) v_0/2$$

を充たす。また $a(i)$ は i 番目の粒子に対応する変数 a を表すこととしよう。通常の場合は力 $f(i) = -V_q(i)$ である事に注意して 64 粒子の非線形格子モデルのプログラムを組んでみよう。今ハミルトニアンを

$$H = \sum_{i=1}^{N} \frac{p(i)^2}{2} + \sum_{i=0}^{N} [\frac{1}{2}(q(i+1) - q(i))^2 + \frac{c_2}{3}(q(i+1) - q(i))^3 + \frac{c_4}{4}(q(i+1) - q(i))^4]$$

としてみよう。但し考える領域は $i = 1, 2, \cdots, n$ であって周期境界条件を課しているものとする。このときの正準方程式を充たす非線形力学系を初期条件を

$$q(i) = \sin(\frac{4\pi * i}{n}), \quad p(i) = 0$$

の下で $t = 8000$ まで解いてみた結果が図２である。そのプログラムは以下の通りで比較的簡単な構造をしている。

```
! 2nd order symplectic integrator for FPU problem
        parameter (n=100,ms=2)
        implicit real*8 (a-h,o-z)
        real*8 p(0:n+1),q(0:n+1),f(n)
        real*8 c(2),d(2)

        open(12,file='fpu.d')

        c(1)=0.5d0
        c(2)=0.5d0
        d(1)=1.d0
        d(2)=0.d0
        pi=4.d0*datan(1.d0)
```

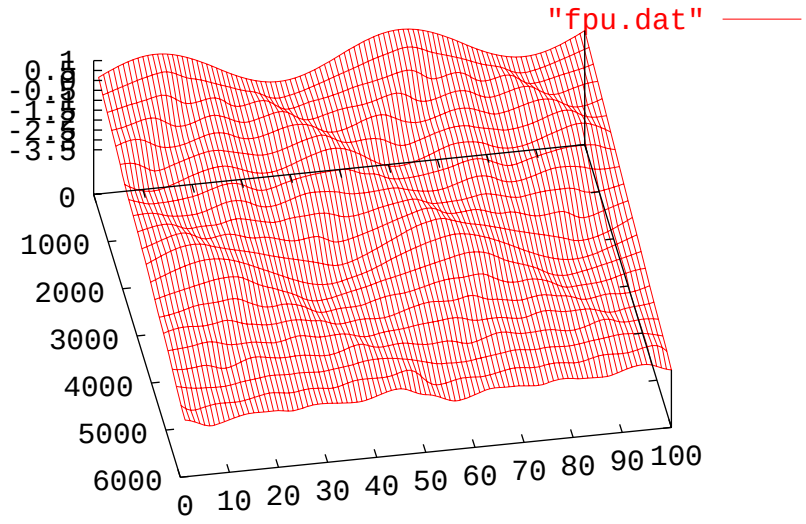

図 2: 非線形格子系のシミュレーション結果。q(i) を表示。

```
      dt=0.05d0
      t=0

      do i=0,n+1! initial condition
       q(i)=dsin(4.d0*pi/dble(n)*dble(i))
       p(i)=0.d0
      enddo
      do i=1,n ! writing data
       write(12,*) t,dble(i),q(i)
      enddo
      write(12,*)

      nt=8000*15

!      time evolution

      do it=1,nt
       do k=1,ms-1
        do i=1,n
          p(i)=p(i)+c(k)*dt*f(i)
          q(i)=q(i)+d(k)*dt*p(i)
        enddo ! end of particle-#-loop
        p(0)=p(n)
        q(0)=q(n)
```

```
          p(n+1)=p(1)
          q(n+1)=q(1)
!          periodic boundary
          call force(n,q,p,f)
         enddo ! end of SI-stage-loop

         do i=0,n
          p(i)=p(i)+c(ms)*dt*f(i)
         enddo ! end of particle-#-loop

         t=t+dt

         if (mod(it,400*15).eq.0) then ! writing data
          do i=1,n
           write(12,*) t,dble(i),q(i)
          enddo
          write(12,*)
         endif ! end of writing data

       enddo ! end of time evolution

        close(12)
        end

        subroutine force(n,q,p,f)
        real*8 q(0:n+1),p(0:n+1),f(n)
        real*8 fc2,fc3

        fc2=-0.5d0
        fc3=0.25d0

        do i=1,n
         f(i)=q(i+1)+q(i-1)-2.d0*q(i)+fc2*((q(i+1)-q(i))*(q(i+1)-q(i))
     &  -(q(i-1)-q(i))*(q(i-1)-q(i)))
     &  +fc3*((q(i+1)-q(i))* (q(i+1)-q(i))*(q(i+1)-q(i))+
     &    (q(i-1)-q(i))*(q(i-1)-q(i))*(q(i-1)-q(i)))
        enddo
        end
```

Fortran90 の自由形式で書くと、

```
integer,parameter::n=100,ms=2
integer::i,it,k
real(8)::pi,dt,t
```

```
real(8)::p(0:n+1),q(0:n+1),f(n)
real(8)::c(2),d(2)

open(12,file='fpu.dat')

 c(1)=0.5_8  ; c(2)=0.5_8
 d(1)=1._8   ; d(2)=0._8
 pi=(4._8)*datan(1.0_8)
 dt=0.05_8
 t=0._8

  do i=0,n+1 !initial condition
     q(i)=dsin(4._8*pi/dble(n)*dble(i))
     p(i)=0.0_8
  enddo

  do i=1,n ! writing data
    write(12,*) t,dble(i),q(i)
  enddo
  write(12,*)

  nt=8000*15

!---time evolution---

  do it=1,nt
    do k=1,ms-1
       do i=1,n
          p(i)=p(i)+c(k)*dt*f(i)
          q(i)=q(i)+d(k)*dt*p(i)
       enddo ! end of particle-#-loop
       p(0)=p(n)
       q(0)=q(n)
       p(n+1)=p(1)
       q(n+1)=q(1)
   !----periodic boundary---
       call force(n,q,p,f)
     enddo ! end of SI-stage-loop

     do i=0,n
       p(i)=p(i)+c(ms)*dt*f(i)
     enddo ! end of particle-#-loop
```

```
     t=t+dt

    if (mod(it,400*15).eq.0) then  ! writing data
       do i=1,n
         write(12,*) t,dble(i),q(i)
       enddo
       write(12,*)

     endif ! end of writing data
 enddo ! end of time evolution

 close(12)
 end

!-----subroutine--------------
subroutine force(n,q,p,f)
 real(8)::q(0:n+1),p(0:n+1),f(n)
 real(8)::fc2,fc3

 fc2=-0.5_8
 fc3=0.25_8

 do i=1,n
    f(i)=q(i+1)+q(i-1)-2._8*q(i)+fc2*((q(i+1)-q(i))*(q(i+1)-q(i))&
     &  -(q(i-1)-q(i))*(q(i-1)-q(i)))&
     &  +fc3*((q(i+1)-q(i))* (q(i+1)-q(i))*(q(i+1)-q(i))+&
     &    (q(i-1)-q(i))*(q(i-1)-q(i))*(q(i-1)-q(i)))
 enddo
end
```

Cで書くと、

```
#include<stdio.h>
#include<math.h>
#define n  100
#define ms  2

void force(double [],double []);
double p[n+2],q[n+2],f[n+1];

int main( ){
   FILE  *fpu;
   int    i,it,k,nt;
```

```
   double pi,dt,t;
   double c[3],d[3];

   fpu=fopen("fpuc.dat","w");

   c[1]=0.5; c[2]=0.5 ;
   d[1]=1.0; d[2]=0.0 ;
   pi=M_PI;
   dt=0.05; t=0.0;

   for(i=0;i<=n+1;i++){//initial condition
     q[i]=sin(4.0*pi/(double)(n)*(double)(i));
     p[i]=0.0;
   }

   for(i=1;i<=n;i++){//writing data
     fprintf(fpu," %f  %d  %f \n",t,i,q[i]);
   }
   fprintf(fpu," \n");

   nt=8000*15;

//---time evolution---
   for(it=1;it<=nt;it++){
     for(k=1;k<=ms-1;k++){
       for(i=1;i<=n;i++){
          p[i]=p[i]+c[k]*dt*f[i];
          q[i]=q[i]+d[k]*dt*p[i];
       }//end of particle-#-loop

   //-periodic boundary-
       p[0]=p[n]  ; q[0]=q[n]  ;
       p[n+1]=p[1]; q[n+1]=q[1];

       force(q,f);
     }//end of SI-stage-loop

     for(i=0;i<=n;i++){
       p[i]=p[i]+c[ms]*dt*f[i];
     }//end of particle-#-loop
     t=t+dt;

    if(it%(400*15)==0){// writing data
```

```
      for(i=1;i<=n;i++){
        fprintf(fpu," %f  %d  %f \n",t,i,q[i]);
      }
      fprintf(fpu,"\n");
    } //end of writing data
  }//end of time evolution
 fclose(fpu);
 return 0;
}

//-----subroutine--------------
void force(double q[],double f[]){
 double fc2,fc3;
 int i;

 fc2=-0.5; fc3=0.25;

 for(i=1;i<=n;i++){
  f[i]=q[i+1]+q[i-1]-2.0*q[i]
     +fc2*((q[i+1]-q[i])*(q[i+1]-q[i])-(q[i-1]-q[i])*(q[i-1]-q[i]))
     +fc3*((q[i+1]-q[i])*(q[i+1]-q[i])*(q[i+1]-q[i])
     +(q[i-1]-q[i])*(q[i-1]-q[i])*(q[i-1]-q[i]));
 }
}
```

問2 上のプログラムのエネルギー保存のチェックをせよ。またいろいろな格子力学系のシミュレーションをしてみよ。