Program 7 / Chapter 7

Conservative forces and potential energy

In the motion of a mass acted on by a conservative force the total energy in the system, which is the sum of the kinetic and potential energies, is conserved. In this section, this motion is computed numerically using the Euler–Cromer method.

Theory In section 7–2 of your textbook the oscillatory motion of a mass attached to a spring is described in the context of energy conservation. Specifically, if the spring is initially compressed then the system has spring potential energy. When the mass is free to move, this potential energy is converted into kinetic energy, $K = {}^1/_2 mv^2$. The spring then stretches past its equilibrium position, the potential energy increases again until it equals its initial value. This oscillatory motion is illustrated in Fig. 7–7 of your textbook.

Consider the more complicated situation in which the force on the particle is given by

$$F(x) = \kappa x - 4qx^3$$

This is a conservative force and the its potential energy is

$$U(x) = -\tfrac{1}{2}\kappa x^2 + qx^4$$

(see Fig. 7–10). From the force, we can calculate the motion using Newton's second law. The program that you will use in this section calculates this motion and demonstrates that the total energy, $E = K + U$, is conserved (i.e., E remains constant).

Given the force, F, on an object (of mass m), its position and velocity may be found by solving the two ordinary differential equations,

$$\frac{dv}{dt} = \frac{1}{m}F \qquad ; \qquad \frac{dx}{dt} = v$$

If we replace the derivatives with their right derivative approximations, we have

$$\frac{v(t+\Delta t) - v(t)}{\Delta t} = \frac{1}{m}F(t) \qquad ; \qquad \frac{x(t+\Delta t) - x(t)}{\Delta t} = v(t)$$

or

$$\frac{v_\mathrm{f} - v_\mathrm{i}}{\Delta t} = \frac{1}{m}F_\mathrm{i} \qquad ; \qquad \frac{x_\mathrm{f} - x_\mathrm{i}}{\Delta t} = v_\mathrm{i}$$

where the subscripts i and f refer to the initial (time t) and final (time t+Δt) values. Of course the approximation is only accurate when Δt is small. Solving each equation for the final values of velocity and position we have,

$$v_\mathrm{f} = v_\mathrm{i} + \frac{1}{m}F_\mathrm{i}\Delta t \qquad ; \qquad x_\mathrm{f} = x_\mathrm{i} + v_\mathrm{i}\Delta t$$

Using these equations repeatedly, we may iterate from any initial condition to compute the motion of the object. This scheme for computing the motion is called the *Euler method.*

Unfortunately, the Euler method is not always accurate; a better way to compute the motion is to use the equations,

$$v_f = v_i + \frac{1}{m} F_i \Delta t \qquad ; \qquad x_f = x_i + v_f \Delta t$$

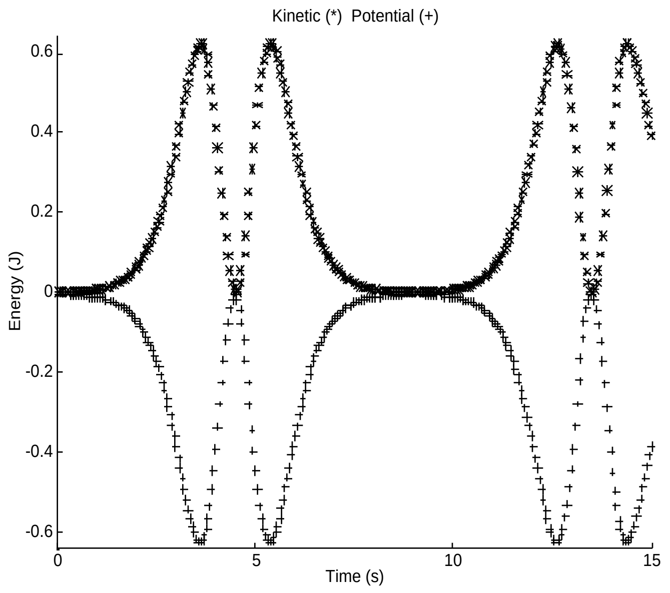
Notice the subtle difference, in the computation of the position the new velocity is used instead of the old velocity. This scheme is known as the *Euler–Cromer method*; it is a simple variant of the leap–frog method.

Program The MATLAB program `energy`, which uses the Euler–Cromer method to compute the motion of a particle acted on by a conservative force, is outlined below:

- Initialize variables (e.g., initial position, Δt, κ, q, m).
- Set up plot of kinetic and potential energy versus time.
- Loop for desired number of steps.
  - Compute kinetic and potential energy.
  - Update the plot by adding new values to the graph.
  - Compute force and acceleration.
  - Update position and velocity using Euler–Cromer.

Notice that when the position and velocity are updated, since the velocity is updated first and then used to update the position, the Euler–Cromer method is implemented. If we reverse the order of these operations (i.e., compute the position first) then we would be using the Euler method.

For κ = 1.0 N/m and q = 0.1 N/m^3, running the program with an initial displacement of 0.1 m, we obtain the plot shown below for the kinetic and potential energies versus time,



Initially the kinetic energy is zero since the initial velocity is $v = 0$. The particle speeds as it moves into the potential “well”, reaching a maximum kinetic energy when the particle reaches the minimum of the potential (near $t = 4$ s). The kinetic and potential energies mirror each other since their sum, the total energy, should remain constant.

Exercises

1. (I) Run the `energy` program and print out the resulting graphs for the following initial displacements: (a) 10^{-3} m; (b) 0.01 m; (c) 0.1 m; (d) 1.0 m; (e) 10.0 m.

2. (I) Repeat exercise 1 but edit the program to set $q=0$ and $\kappa = -1$ N/m, making the force that of an ideal spring.

3. (I) Modify the `energy` program to also plot the total energy; run your program and print the graphs using the displacements specified in exercise 1.

4. (II) Modify the `energy` program to use the Euler method instead of Euler–Cromer; run your program for the displacements given in exercise 1 and print the resulting graphs. Comment on the difference between the plots produced by the two methods.

5. (II) Modify the `energy` program to graph position and velocity versus time; run your program for the displacements given in exercise 1 and print the resulting graphs.

6. (III) The period of an oscillation is the time it takes for the mass to return to its original position. Write a program that computes the period for various displacements (from 10^{-3} m to 10.0 m). Plot the period versus displacement using a semilog scale (horizontal log scale; see the `semilogx` command).

7. (III) (a) Write a program that accepts an arbitrary expression for the force law as a function of displacement and, using the symbolic manipulation capabilities of MATLAB, obtains an expression for the potential energy. (b) Also have your program plot $U(x)$ versus x.

Listing

`energy.m`

```
% energy - Program to compute kinetic and potential energy
%          for a particle acted on by a conservative force
clear all;      % Clear memory
help energy;    % Print header

%@ Initialize variables
kCoeff = 1.0;   % Kappa coefficient (N/m)
qCoeff = 0.1;   % q coefficient (N/m^3)
x = input('Enter initial displacement (m): ');
v = 0.;         % Initial velocity (m/s)
mass = 1.0;     % Mass of particle
dt = 0.05;      % Time step (s)
KEnergy = 0.5*mass*v^2;                % Kinetic energy
UEnergy = -0.5*kCoeff*x^2 + qCoeff*x^4;  % Potential energy
TEnergy = KEnergy + UEnergy;           % Initial total energy
```

```
t = 0;                   % Time
Nstep = 300;             % Number of time steps

%@ Set up plot of kinetic and potential energy versus time
clf;                            % Clear the figure window
figure(gcf);                    % Bring figure window forward
EnergyLimit = 2*abs(TEnergy);     % Estimate y-axis limit
axis([0, Nstep*dt, -EnergyLimit, EnergyLimit]);    % Set axis limits
xlabel('Time (s)');             % x-axis label
ylabel('Energy (J)');           % y-axis label
title('Kinetic (*)  Potential (+)');
hold on;                        % Hold the plot on screen

%@ Loop for desired number of steps
for istep=1:Nstep

  %@ Compute kinetic and potential energy
  KEnergy = 0.5*mass*v^2;  % Kinetic energy
  UEnergy = -0.5*kCoeff*x^2 + qCoeff*x^4;
  t = (istep-1)*dt;   % Time

  %@ Update the plot by adding new values to the graph
  plot(t, KEnergy,'r*','EraseMode','none');
  plot(t, UEnergy,'b+','EraseMode','none');
  while( abs(KEnergy) > EnergyLimit | abs(UEnergy) > EnergyLimit )
    EnergyLimit = 2*EnergyLimit;
    axis([0, Nstep*dt, -EnergyLimit, EnergyLimit]);    % Set axis limits
  end
  drawnow;  % Update the figure now

  %@ Compute force and acceleration
  Force = kCoeff*x - 4*qCoeff*x^3;   % Force
  accel = Force/mass;                % Acceleration

  %@ Update position and velocity using Euler-Cromer
  v = v + accel*dt;    % Compute new position and velocity
  x = x + v*dt;        % using Euler-Cromer
end
```