CHEMICAL AND ENERGY ENGINEERING

SIMULATIONS OF MECHANICAL PROCESSES

PROBLEM 5.6: COLLIDING PARTICLES

Submitted by:

1. Angshuman Buragohain
   Matriculation Nr. 221552
2. Devidas Khatri
   Matriculation Nr. 221549

Chemical and Energy Engineering
Otto-von-Guericke Universität, Magdeburg

Submitted to:

Prof. Berend van Wachem
Universitätsplatz, 39106,
Otto-von-Guericke Universität,
Magdeburg, G10-236

Authors: Angshuman Buragohain, Devidas Khatri

Date: 5th July 2019

Problem 5.6: Colliding particles

Two particles at positions x1=(0,0,0)m and x2=(0,2,0)m have diameters=1m. Particle 1 is stationary and fixed at the origin and particle 2 with a mass of 0.01kg moves at a velocity of (0,-0.5,0)m/s.

Objectives:

1. Designing a function which takes positions of two particles and their diameters and returns the overlap (scalar) and the normal (vector)
2. Using initial condition of the particles, move the particle 2 with a fixed time step=1e-4s using Leapfrog algorithm (Using predictor -corrector)
3. Moving the particle with time step and evaluating the overlap. k_n=1000 and computation of the normal force based on the linear model given by Equation 1: $F = k_n \times \delta \times \vec{n}$
4. Verification of the working of the program using 40,000 time-steps
5. Plotting Kinetic energy of particle 2 as a function of time
6. Perform the objectives 3 and 4 using coefficient of restitution e=0.8

Objective 1: Function to take arbitrary position and diameter of two particles and return the overlap and the normal

The two positions of the particles are x1 and x2. The distance between the particles and their radii (r1 and r2) are used to calculated the overlap. Overlap = (r1+r2) - (Dist. betn. particles)

```
function [del, n] = Overlap(x1, x2, d1, d2)
        % Radius of the particles
        r1=d1/2;
        r2=d2/2;

        % Relative displacement between the two particles
        x12 = x1-x2;

        % Distance between the two particles
        dist_part=norm(x12);

        % Overlap
        del = r1+r2-dist_part;

        % Normal vector
        n = x12/norm(x12)
    end
```

Objective 2: Function to move the particle 2 with a fixed time step using Leapfrog mechanism.

The function described below is based on Leap frog mechanism to move particle 2 with the fixed time step of $dt = 10^{-4}$. For leap frog mechanism the velocities for calculation of new position are calculated at the mid-points of the time intervals (*dt/2*) which are assumed as *vstar* values. Using *vstar* values *x_new* is calculated from *x_old*. The correction of velocity is done by calculating the velocities. *v_new* (using *v_old* values) at the exact points of the iterations i.e at (*dt*) intervals using *vstar* values and the average value of acceleration before and after interval *dt*.

The prediction of velocity is given by the equation (1).

$$vstar = v_{old} + 0.5 \times dt \times a_{old} \tag{1}$$

$$x_{new} = x_{old} + v_{old} \times dt + 0.5 \times a_{old} \times dt^2 \tag{2}$$

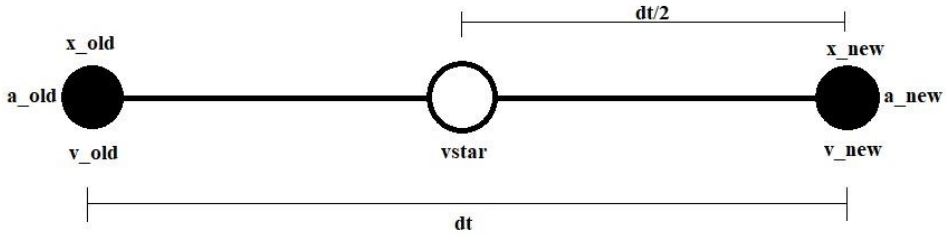
Replacing (1) in (2), we get

$$x_{new} = x_{old} + vstar \times dt \tag{3}$$

The correction of velocity after the interval *dt* is calculated using the *vstar* value and the average of the acceleration before and after the interval *dt*.

$$v_{new} = vstar + 0.5 \times dt \times (a_{old} + a_{new}) \tag{4}$$

The Leap-frog algorithm can be realised by the figure given below:



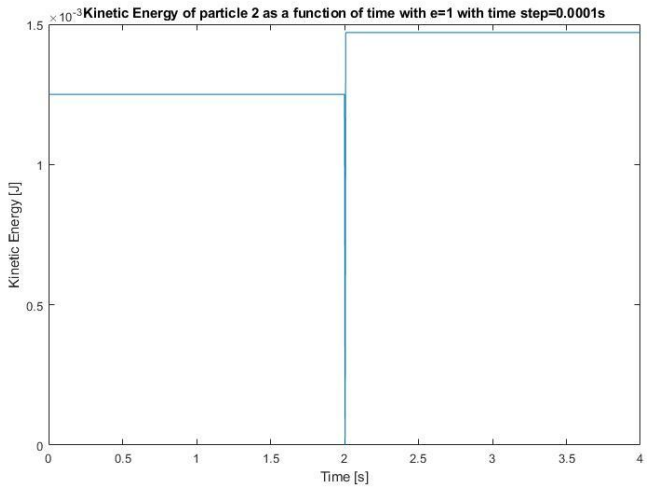

```
function [x_new,v_new] = Particle_move(x_old,v_old,F_new, F_old,m,dt)


        % Accelaration from forces before and after time interval dt
        a_new = F_new./m;
        a_old = F_old./m;

        % Prediction of velocity at mid interval using old velocity and old
          acceleration
        vstar = v_old + 0.5*dt.*a_old;
```

```
        % Position shifting calculation
        x_new= x_old + vstar*dt;

        % Correction of velocity after interval dt
        v_new = vstar + 0.5*dt.*(a_old+a_new);

    end
```

Objective 3&4: Moving the particle 2 with the time step given and calculating the force and final velocities in case of overlap

```
Force{1}=[0 0 0];
   for j=1:N
       [del, n] = Overlap(x1, x2(j,:), d1, d2);
       if (del<0)
           Force{j+1}=[0 0 0];
           F_new=Force{j+1};
           F_old=Force{j};
           [x2(j+1,:) , v2(j+1,:)] = Particle_move(x2(j,:),v2(j,:), F_new,
            F_old, m2, dt);

       elseif (del>0)
           Force{j+1}=-kn_loading*del*n;
           F_new=Force{j+1};
           F_old=Force{j};
           [x2(j+1,:) , v2(j+1,:)] = Particle_move(x2(j,:),v2(j,:), F_new,
            F_old, m2, dt);
       end
       KE(j+1) = 0.5*m2*dot(v2(j+1,:),v2(j+1,:));
   end
```

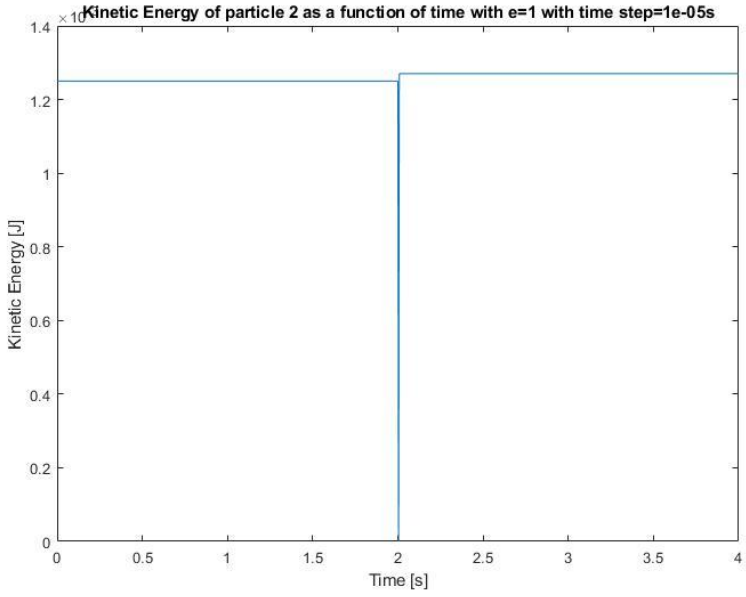
The results of running for 40000 time steps is given as follows in the figure below



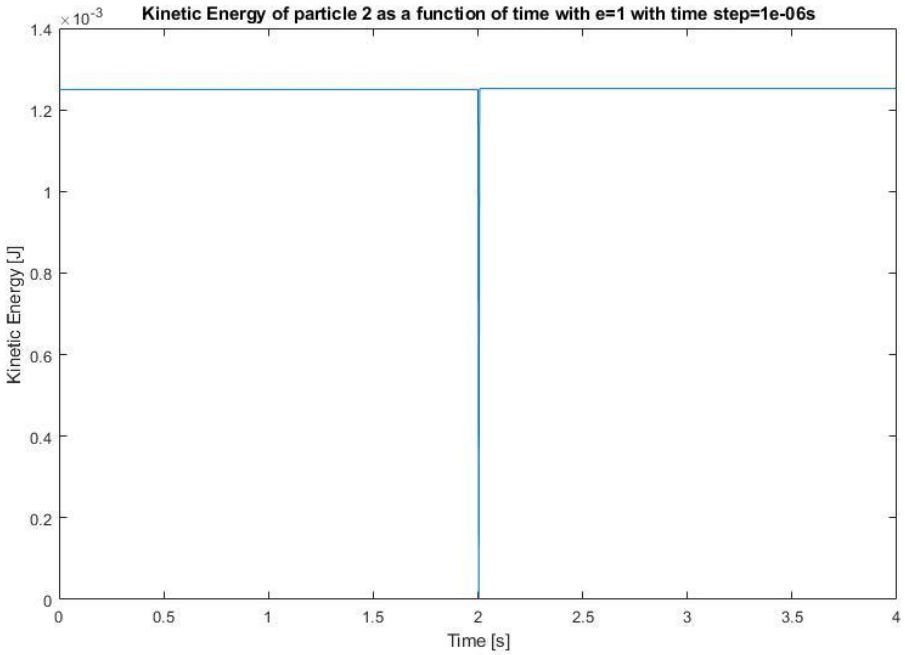
The Kinetic energy should have been equal before after the overlap is over. Hence the time step is not optimum.

Objective 5. Plotting Kinetic energy of particle 2 as a function of time

The Program was run for higher time steps with more precise time intervals and the results are as follows:



The best result was obtained for time step = 4E+6 and time interval = 1E-6 as shown below in context to conservation of energy and momentum (Equal before and after overlap).

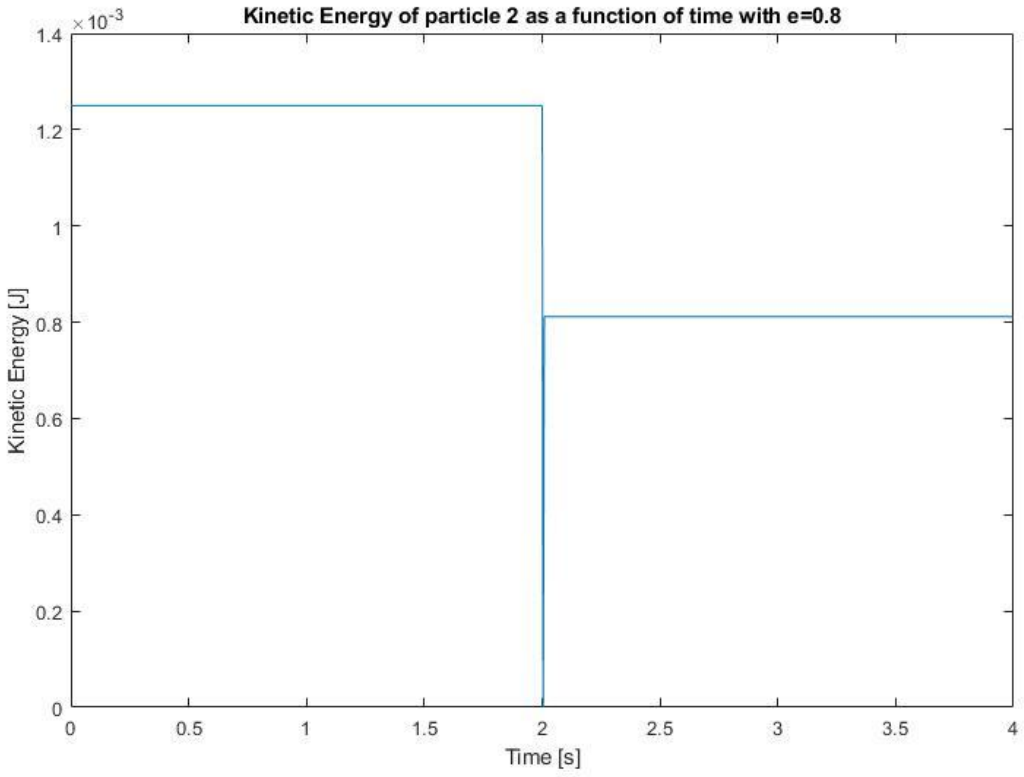


Objective 6: Performing the objectives 3 and 4 using coefficient of restitution e=0.8

```
Force{1} = [0 0 0];
   for j=1:N
       [del, n] = Overlap(x1, x2(j,:), d1, d2);

       % Before and after overlapping
       if (del<0)
           Force{j+1}=[0 0 0];
           F_new=Force{j+1};
           F_old=Force{j};
           [x2(j+1,:) , v2(j+1,:)] = Particle_move(x2(j,:),v2(j,:), F_new,
            F_old, m2, dt);

       % During overlap
       elseif (del>0)
           v21 = v2(j+1,:)-v1;

           % Loading condition
           if (dot(v21,n)>0)
                kn = kn_loading;

           % Unloading condition (e=sqrt(k_unloading/k_loading))
           elseif (dot(v21,n)<0)
```

```
            kn = kn_loading*(e_rest^2);
        end
        Force{j+1}=-kn*del*n;
        F_new=Force{j+1};
        F_old=Force{j};
        [x2(j+1,:) , v2(j+1,:)] = Particle_move(x2(j,:),v2(j,:), F_new,
         F_old, m2, dt);

    end
    KEe(j+1) = 0.5*m2*dot(v2(j+1,:),v2(j+1,:));
end
```

The kinetic energy was calculated before and after the velocity of particle 2 changes direction using the loading and the unloading conditions as given in the code above. The result is plotted as shown below



Introducing a coefficient of restitution of 0.8 reduced the Kinetic energy after unloading to 80% of the initial value. This means the dissipation of Kinetic energy due to some degree of inelasticity is 20% which is in accordance to concept of partially inelastic collision.