C12 Sphere's Collision – ACSL

Algorithmic Simulation / Time-oriented Model

Simulator: ACSL is a continuous CSSL-type simulator with textual and graphical model building features and a MATLAB-like experimental environment. In this solution, only the environment ACSL-MATH is used.

Model: As the problem allows analytical solutions, AMATH - modules simulate the time courses. The state of the system is kept in variables for the relative speed and position of the spheres. The absolute position and speed of the first sphere is also kept for reasons of completeness, but is not used for further calculations.

Special AMATH modules (like m - files) are programmed in order:

- to determine the time instance of the next collision (or reveal the fact that no more collision is going to occur); (`colltime.m`)
- to update the positions for this newly gained time instance (which is simply a linear movement, no differential equations necessary) and to update the speeds according to the rules for partially elastic collisions (`model.m`):
-

These modules are called in a loop until no more collision was going to occur (i.e. all relative speeds are positive). Parts of the self-explaining code:

```
colltime;   %next collision
  while ck>0  %collision occurs
   x=x+ct*dx;  %calc.new pos.
   y=y+ct*dy;t=t+ct; %update time
 if y(j)<0  %correct wrong neg.positions
if ck==1   %perform collision
  if y(2)<=0 & dy(2)<=0
  if y(3)<=0 & dy(3)<=0
    dx=dx+(1+e)*3/4*dy(1);
    dy(1)=-e*dy(1); cm=1;
 else
    dx=dx+(1+e)*2/3*dy(1);
```

Task a) Simulation in time domain / Final values of velocities. Fig. 1 shows the distance time functions (task a1). For Task a2, results could be calculated for e=1 (elastic case), and also e=0 (plastic case). In both cases three collisions occurred, and the final velocities were $v_1=v_2=v_3=0$, $v_4=1$ and $v_1=v_2=v_3=v_4=0.25$, resp.

In order to get results for the plastic case, it was necessary to handle numerical problems with velocities near to zero and negative velocities (which occur due numerical problems because of finite word length) separately.

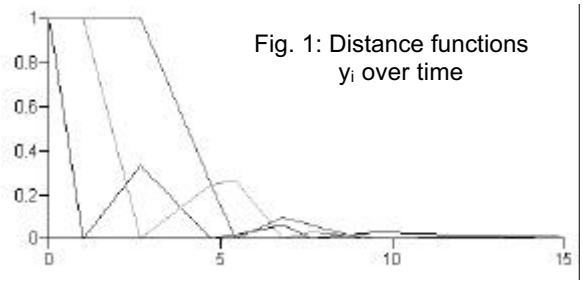

Fig. 1: Distance functions y_i over time

Task b) Variation of restitution coefficient. Because of the separately handling of small and negative velocities (numerical errors) the model calculates almost correct values down to amazingly small values of the collision coefficient (see fig. 2" for number of collisions, and fig. 3) for final velocities).

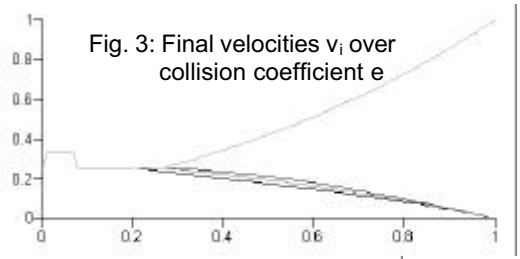

Fig. 3: Final velocities v_i over collision coefficient e

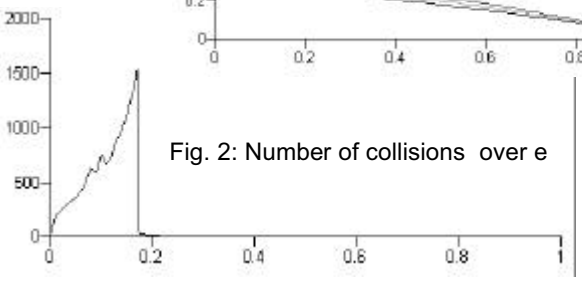

Fig. 2: Number of collisions over e

Task c) Boundary value problem / Statistical deviation of restitution coefficient. The boundary value problem (task a1) is simply solved by interpolating data calculated in task b, see fig. 3: $v_4=v_0/2$ holds, if `e = 0.5874`.

Task c2 is performed experimentally. A sample of 1000 simulation runs gives the following results:

```
mean = 0.4226, std. dev. = 0.0430
95% conf. interval:
         min = 0.3425, max = 0.5140
```

M. Lingl, F. Breitenecker, ARGESIM/SIMTECH
Dept. Simulation Technique, TU Vienna,
Wiedner Hauptstr. 8-10, A-1040 Vienna
email: Felix.Breitenecker@tuwien.ac.at