SIMULATION OF 1D AND 2D COLLISIONS

G. CHEN
Lab Technician, Data Analyst

L. CHEN
Lab Technician, Data Analyst

E. SHEN
Lab Technician, Data Analyst

K. WAN
Lab Technician, Data Analyst

U. YOUSAFZAI
Lab Technician, Data Analyst

(Received Oct. 15, 2019)

ABSTRACT

1D collisions are usually solved analytically by using the conservation of both linear momentum and energy. However, this problem is algebraically tedious when done for two or more dimensions, especially when rotational motion is involved. This report analyzes the collision of two spheres in 1D and 2D by simulating the event frame by frame using the spring force the balls exert on each other. Rotations, both initial and those caused by friction during the collision are also analyzed in this report.

I SPRING CONSTANT ANALYSIS

The properties of the stress ball were experimentally measured to model the properties of the spherical objects implemented in the program. Particularly, the spring constant was measured in order to determine the compression properties of the collision. The measurements were done using a Vernier LabQuest, a Vernier Motion Detector, and a Vernier Force Probe. The stress ball was compressed against a wall by a varying force, and the motion detector was used to measure the distance compressed while the force probe measured the force applied. This was repeated 10 times to obtain 10 runs with force and compression as a function of time. To ensure both measurement devices functioned, the Vernier Motion Detector was tested alongside a ruler while the Force Probe was tested with varying weights. It was determined that the Motion Detector was accurate for a range between 0.2m to 5.9m, while the Force Probe was accurate for a range from 6N to 50N.

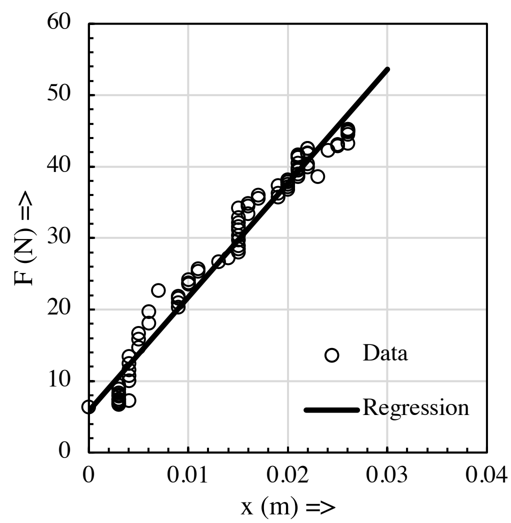


Fig 1: Plot and Regression of Force Versus Compression Data of Stress Ball. This data was collected in Run 1. The regressed equation was F = (1580±30) x + (5.8±.6). The R^2 value is 0.98.

The force was expressed as a function of compression. The collected data about the compression length and the applied force was graphed and regressed. The data for each compression resembled a linear trend, and further regression confirmed this analysis as the linear regression had the highest R^2 statistic of 0.98. Each compression run was regressed linearly and the coefficient indicating the slope was averaged, resulting in a spring constant of (1500±30) Nm^{-1}.

INDEX	SPRING CONSTANT (k) (Nm^{-1})
1	1580±30
2	1520±20
3	1510±40
4	1520±20
5	1490±50
6	1480±30
7	1500±20
8	1490±20
9	1450±40
10	1500±30
AVERAGE	1500±30

Fig 2: Spring Constant of 10 Runs and Average. The spring constants for each run was obtained through linear regression. The runs were averaged to obtain a value of (1500±30) Nm^{-1}.

The constant of the regression can be dropped as the initial reading of the Force Probe without any applied force is greater than 0.

Some error that explains the large uncertainties include the Motion Detector being unable to register a change in distance in short time changes. This is mitigated by collecting many data points to lower uncertainty.

II SAMPLE PROBLEMS

The time step used for the simulation was the largest exponent of ten that would result in residuals of total mechanical energy of less than one percent. In most applications, it was found that 10^{-4} s was enough to achieve this result. In addition, the velocities of the objects in the problems the simulator was tested on were small, so no modification was made to the time step throughout these tests.

The objects were modelled as compressible balls to simplify computations. The spring constant used was set large enough such that the objects wouldn't pass through each other. Apart from that, the actual value of the spring constant doesn't matter as it wouldn't cause energy loss.

For the one-dimensional collision, problem 61 from CPS 9 of HRW 10th edition was used. The mass of the second cart obtained from part a) was used to solve part b). The solutions manual was consulted in order to obtain the correct values.

	TEXTBOOK (v) (ms^{-1})	SIMULATOR (v) (ms^{-1})	RESIDUALS (%)
Final speed of object 1	0.66	0.65877	0.19
Final speed of object 2	1.86	1.85877	0.07

Fig 3: Values for the final speeds of the two objects obtained from the solutions manual and simulation after the 1D collision. The residuals were also calculated and displayed.

The residuals are likely the result of the textbook rounding the results to two decimal points.

For the two-dimension collision with a stationary target, problem 36 from CPS 9 of HRW 8th edition was used. The solutions manual was consulted to obtain the correct values. This problem involved a glancing collision with the constraint that the direction of the first object was altered by a certain angle. Some trigonometry was used to calculate the relative y-coordinate of the first object that would create the required collision. Then, the values were plugged into the simulator to obtain the values for the final speeds. Although the values of the masses weren't given in the problem, it was solved by setting the mass of both objects to 1 kg since the only constraint was that the objects had the same mass.

	TEXTBOOK (v) (ms^{-1})	SIMULATOR (v) (ms^{-1})	RESIDUALS (%)
Final speed of object 1	4.00	3.97223	0.69
Final speed of object 2	3.01	3.03668	0.89

Fig 4: Values for the final speeds of the two objects obtained from the solutions manual and simulation after the 2D collision with a stationary target. The residuals were also calculated and displayed.

For the two-dimensional collision with both objects in motion, problem 73 from CPS 9 of HRW 8th edition was used. The same method as the 2D-problem with one stationary target was employed to set up the simulator. Once again, the mass of the first object was set to 1 kg and the second to 3 kg as it would satisfy the constraints. The simulator was run with all initial velocities in terms of v_i. The x and y components of final velocity were used to determine the direction and magnitude of final velocity.

	TEXTBOOK (v)	SIMULATOR (v)	RESIDUALS (%)
Final speed of object 1	1.4142 v_i	1.4102 v_i	0.28
Final speed of object 2	0.8165 v_i	0.81878 v_i	0.28

Fig 5: Values for the final speeds of the two objects obtained from the solutions manual and simulation after the 2D collision. The residuals were also calculated and displayed.

While the final speeds were relatively accurate, the angle of the resulting velocity would benefit from increased accuracy. This is because even a small residual in the resulting direction could greatly affect the x and y components of the velocity.

	TEXTBOOK (degrees)	SIMULATOR (degrees)	RESIDUALS (%)
Direction of final velocity of object 2	54.7	54.963	0.48

Fig 6: Direction of the final velocity of object 2 as obtained from the solutions manual and simulation. Residuals in angle should be taken very seriously as they can have a large effect on the components of the resulting velocity.

III PROGRAM LISTING

For each simulation, the masses used were assumed to be the masses of the stress balls which were measured to be 0.0254 kg. The radii inputted were the measured radii of the stress balls used in this experiment: 0.03021 m. The rotational inertia was assumed to be that of a solid sphere for both objects.

The simulation models the collision between the two objects by modelling the deformation experienced by both spheres and the forces that this deformation exerts onto the spheres.

In each frame, it is first checked whether or not the two spheres are in contact by comparing the distance between the two centers to the radii of the circles. The compression required for each circle is also calculated with this comparison. A force is then calculated based off of this displacement as well as the spring constant that was observed from Section I.

Shown below are the two files that were used for the simulation: the driver file "Simulation.cpp" and the class "Vector2D.cpp". The initial conditions set in the program are for the bonus demo, with a nonzero initial velocity and a nonzero initial angular velocity for both objects.

Simulation.cpp:

```
//NOTE: BONUS INCLUDED
#include <stdio.h>
#include "Vector2D.cpp"
using namespace std;
const double eps=1e-6;
//epsilon: if the relative speed of the two spheres at the point of contact is
greater than epsilon, then there is friction. This prevents oscillatory behaviour
from friction due to a non-zero time step
double m1=0.0254,m2=0.0254;//masses
double r=0.03021;//radius of both spheres
double I1=0.4*m1*r*r,I2=0.4*m2*r*r;//calculated moments of inertia
double u=0;//mu: coefficient of friction
Vector2D p1,p2,p;//momentums of spheres and total momentum
double L1,L2,L;//angular momentums
double E1,E2,E;//kinetic energies
double w1=3,w2=-5;//angular velocities
Vector2D s1(0,0.01),s2(0.5,0);//displacement wrt the fiducial point 2D
Vector2D v1(1,0),v2(-2,0);//velocities 2D
Vector2D scom,vcom;
//Vector2D s1(0,0),s2(1,0);//displacement wrt the fiducial point 1D
//Vector2D v1(1,0),v2(-2,0);//velocities 1D
double k=1500;//spring constant
const double dt=0.000001;//time step
void step(){
    Vector2D a1(0,0),a2(0,0);//accelerations
    double alpha1=0,alpha2=0;//angular accelerations
    double dis=(s1-s2).mag();//distance between two centers
    double PE1=0,PE2=0;//potential energies
    if(dis<2*r){//check for contact
        double x=(2*r-dis)*0.5;//compression amount for each sphere
        double Ff=u*k*x;//magnitude of frictional force
        //acceleration calculated from force which points perpendicular from the
        intersection of the two circles
        a1=(s1-s2).norm()*k*x/m1;
        a2=(s2-s1).norm()*k*x/m2;
        //calculates potential energy of the spring
        PE1=PE2=0.5*k*x*x;
        Vector2D perp=(s1-s2).norm().perp();//unit vector along the intersection of
        the two circles
        //speed along the direction of perp (adjusted for rotation at contact point
        double ev1=v1.dot(perp)-r*w1;
        double ev2=v2.dot(perp)+r*w2;

        double v12=ev1-ev2;//relative speed of object1 with respect to object 2
        if(v12>eps){//friction in one direction
            a1=a1-perp*Ff/m1;//adds frictional forces
            a2=a2+perp*Ff/m2;
            alpha1=((s2-s1)*0.5).cross(perp*-Ff)/I1;//calculates the angular
            acceleration
            alpha2=((s1-s2)*0.5).cross(perp*Ff)/I2;
        }
        if(v12<-eps){//friction in the other direction
            //same as above
```

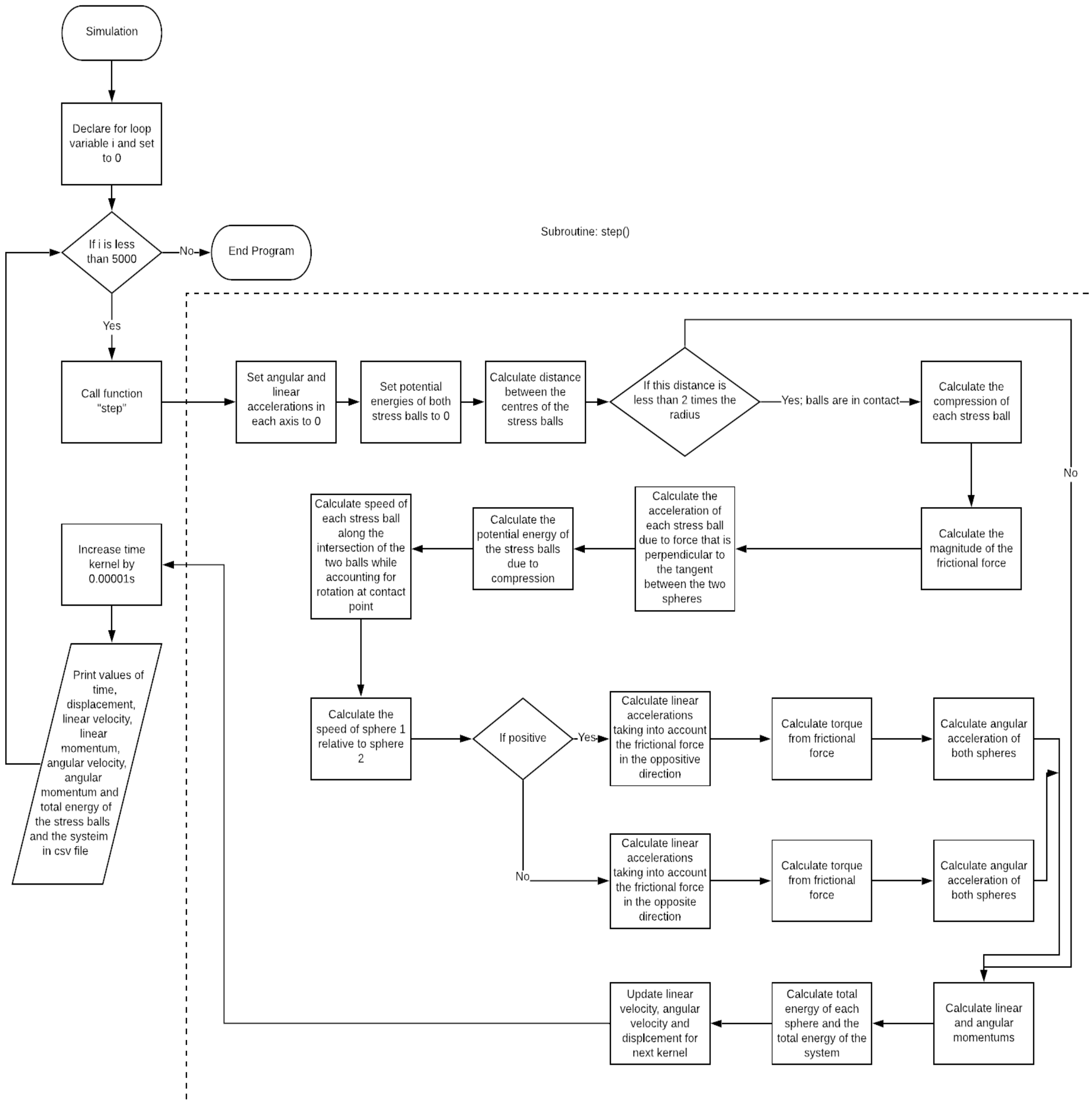
```
            a1=a1+perp*Ff/m1;
            a2=a2-perp*Ff/m2;
            alpha1=((s2-s1)*0.5).cross(perp*Ff)/I1;
            alpha2=((s1-s2)*0.5).cross(perp*-Ff)/I2;
        }
    }
    //calculates quantities related to the center of mass
    scom=(s1*m1+s2*m2)/(m1+m2);
    vcom=(v1*m1+v2*m2)/(m1+m2);
    //calculates momentums
    p1=v1*m1,p2=v2*m2,p=p1+p2;
    //calculates angular momentums
    L1=s1.cross(p1)+I1*w1;
    L2=s2.cross(p2)+I2*w2;
    L=L1+L2;
    //calculates energies
    E1=0.5*m1*v1.dot(v1)+0.5*I1*w1*w1+PE1;
    E2=0.5*m2*v2.dot(v2)+0.5*I2*w2*w2+PE2;
    E=E1+E2;
    //adjusts angular velocity, velocity and displacement for next frame
    w1+=alpha1*dt;
    w2+=alpha2*dt;
    s1=s1+v1*dt+a1*dt*dt*0.5;
    s2=s2+v2*dt+a2*dt*dt*0.5;
    v1=v1+a1*dt;
    v2=v2+a2*dt;
}
int main(){
    //freopen("//Users//kevinwan//Desktop//sim1.csv","w",stdout);
    freopen("2D-SimulationData-x-t.csv","w",stdout);//prepares the csv file

    printf("time,s1.x,s1.y,s2.x,s2.y,scom.x,scom.y,v1.x,v1.y,v2.x,v2.y,vcom.x,
    vcom.y,p1.x,p1.y,p2.x,p2.y,p.x,p.y,w1,w2,L1,L2,L,E1,E2,E\n");//writes the header
    double time = 0;
    for(int i=0;i<=100000;i++){
        step();//a single timestep
        //prints the relevant information
        if(i%100==0)
            printf("%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,
                    %lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf,%lf\n",
                    time,s1.x,s1.y,s2.x,s2.y,scom.x,scom.y,v1.x,v1.y,v2.x,v2.y,
                    vcom.x,vcom.y,p1.x,p1.y,p2.x,p2.y,p.x,p.y,w1,w2,L1,L2,L,E1,E2,E);
        time += dt;
    }
}
```

Vector2D.cpp:

```
#include<math.h>
class Vector2D{
public:
    double x,y;
    Vector2D(double x,double y){this->x=x,this->y=y;}//initializes the vector
    Vector2D(){this->x=this->y=0;}//initializes the zero vector
    Vector2D polar(double r,double t){return {r*cos(t),r*sin(t)};}//creates a vector
by polar form (radians)
    Vector2D operator+(Vector2D ot){return {this->x+ot.x,this->y+ot.y};}//adding two
vectors
    Vector2D operator-(Vector2D ot){return {this->x-ot.x,this->y-ot.y};}//subtracting
two vectors
    double dot(Vector2D ot){return this->x*ot.x+this->y*ot.y;}//returns the dot-
product
    double cross(Vector2D ot){return this->x*ot.y-this->y*ot.x;}//returns the 2D
version of cross product
    double mag(){return sqrt(this->x*this->x+this->y*this->y);}//returns the
magnitude of the vector
    double dir(){return atan2(this->y,this->x);}//returns the direction the vector is
pointing (radians)
    Vector2D norm(){double t=dir();return {cos(t),sin(t)};}//returns the norm of the
vector
    Vector2D operator*(double c){return {this->x*c,this->y*c};}//multiplies the
vector by a constant
    Vector2D operator/(double c){return {this->x/c,this->y/c};}//divides the vector
by a constant
    double proj(Vector2D ot){return this->dot(ot)/ot.mag();}//returns the length of
the projection of a vector onto another
    Vector2D perp(){return {-y,x};}//returns the vector rotated 90 degrees CCW
};
```

IV Flow Chart

The flowchart demonstrates the logic of the program. It utilizes the principles pursuant to van Bemmel's Simulation Primer.



V 1D DEMO

For this section, the energy of the system is calculated as the sum of the kinetic (both linear and rotational) and potential energies. The potential energies are defined as the spring potential energies of the sphere, evaluated in the system.

The following is a graph of a one-dimensional collision between two objects as simulated by the program:

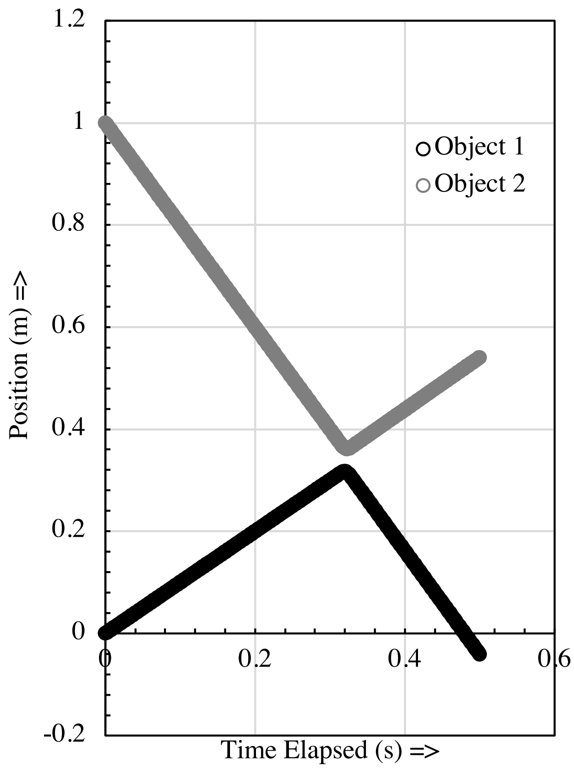


Fig 7: Position Versus Time of 1D Collision Simulation. The initial velocity of Object 1 is +1 ms^{-1}. The initial velocity of Object 2 is -2 ms^{-1}. The initial position of Object 1 is 0 m. The initial position of Object 2 is +1 m.

The following tables consist of ten lines of the simulation output:

ID	TIME (s)	OBJECT 1 X-POSITION (m)	OBJECT 2 X-POSITION (m)
1	0.312	0.31202	0.37596
2	0.314	0.314015	0.371965
3	0.316	0.315803	0.368177
4	0.318	0.316969	0.365011
5	0.32	0.317174	0.362806
6	0.322	0.31623	0.36175
7	0.324	0.314128	0.361852
8	0.326	0.311038	0.362942
9	0.328	0.307284	0.364696
10	0.33	0.303285	0.366695

Fig 8: 10 Lines of Simulation Output, Position Data. Positions indicate the centre of mass of each object, with the fiducial set at 0.

ID	OBJECT 1 X-VELOCITY (ms^{-1})	OBJECT 2 X-VELOCITY (ms^{-1})	MOMENTUM OF SYSTEM (kg ms^{-1})
1	1	-2	-0.0254
2	0.980348	-1.980348	-0.0254
3	0.771691	-1.771691	-0.0254
4	0.365141	-1.365141	-0.0254
5	-0.176247	-0.823753	-0.0254
6	-0.768399	-0.231601	-0.0254
7	-1.319286	0.319286	-0.0254
8	-1.74322	0.74322	-0.0254
9	-1.974186	0.974186	-0.0254
10	-2.004682	1.004682	-0.0254

Fig 9: 10 Lines of Simulation Output, Velocity and Momentum Data. The velocity is relative to the fiducial. The momentum of the system is calculated by adding the individual momentums of the objects. Refer to Fig 8 for the time data of each index.

ID	ENERGY OF OBJECT 1 (J)	ENERGY OF OBJECT 2 (J)	ENERGY OF SYSTEM (J)
1	0.0127	0.0508	0.0635
2	0.012955	0.05058	0.063536
3	0.015617	0.047997	0.063614
4	0.020782	0.04288	0.063662
5	0.027653	0.036024	0.063677
6	0.035173	0.028505	0.063679
7	0.042187	0.021505	0.063692
8	0.047614	0.016122	0.063736

9	0.050612	0.013199	0.063811
10	0.051038	0.012819	0.063857

Fig 10: 10 Lines of Simulation Output, Energy Data. Energy refers solely to kinetic energy of the system. Refer to Fig 8 for the time data of each index.

Object 1 initial velocity (ms^{-1})	+1
Object 1 final velocity (ms^{-1})	-2.004682
Object 2 initial velocity (ms^{-1})	-2
Object 2 final velocity (ms^{-1})	+1.004682

Fig 11: Initial and Final Conditions of Simulation. The velocities are all pointed along the x-axis. Residuals of the final velocities are below 1%.

The time step chosen for this simulation was 0.00002 s. This specific time step was chosen such that the residuals yielded were less than 1%. The program started the simulation before the moment of collision to distinguish the motion of the objects before and during the collision, to ensure that the graphs can be displayed to show when the collision takes place. The program ended after the difference between the center of masses of the two objects exceeded twice their radii, as no force acts on the balls after this criterion is met. The program continued to generate data after to ensure the graph could visually represent the change after the collision.

VI 2D DEMO

For this section and the following 2D Moving Target Demo section, the position, velocities, momentum, and energy of both objects and the entire system were evaluated by the program. Energies of the objects and system are calculated as the sum of the kinetic (both linear and rotational) and potential energies.

The following is a graph of a two-dimensional collision between two objects with one object originally stationary:

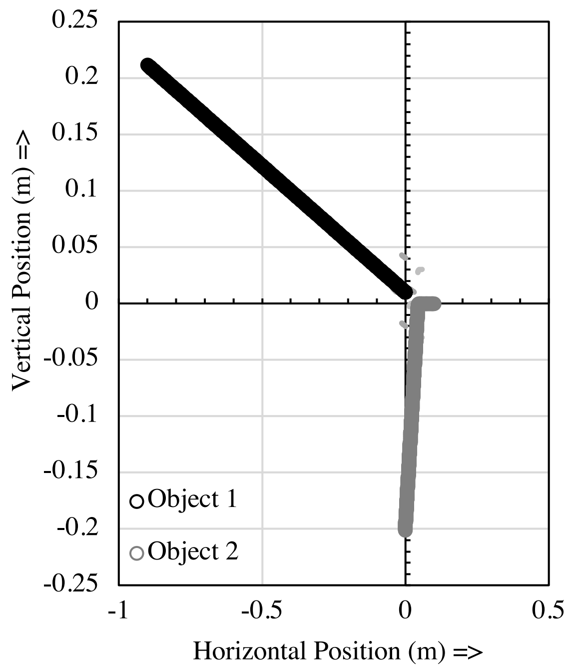


Fig 12: Vertical Versus Horizontal Position of Two Objects. Dimmed lines represent non-compressed boundaries of both objects during contact.

The following tables consist of ten lines of the simulation output:

ID	TIME (s)	OBJECT 1 X-POSITION (m)	OBJECT 1 Y-POSITION (m)
1	0.018	0	0.01
2	0.02	0	0.01
3	0.022	-0.000037	0.010006
4	0.024	-0.000338	0.010061
5	0.026	-0.001156	0.010215
6	0.028	-0.002663	0.010512
7	0.03	-0.004918	0.010977
8	0.032	-0.007867	0.011607
9	0.034	-0.011345	0.012372
10	0.036	-0.015106	0.013213

Fig 13: 10 Lines of Simulation Output, Position Data of Object 1. Positions indicate the centre of mass, with the fiducial set at 0.

ID	Time (s)	Object 2 x-position (m)	Object 2 y-position (m)
1	0.018	0.063998	0
2	0.02	0.059998	0
3	0.022	0.056035	-0.000006
4	0.024	0.052336	-0.000061
5	0.026	0.049154	-0.000215
6	0.028	0.046661	-0.000512
7	0.03	0.044916	-0.000977
8	0.032	0.043865	-0.001607
9	0.034	-0.011345	0.012372
10	0.036	-0.015106	0.013213

Fig 14: 10 Lines of Simulation Output, Position Data of Object 2. Positions indicate the centre of mass, with the fiducial set at 0.

ID	Time (s)	Object 1 x-velocity (ms^{-1})	Object 1 y-velocity (ms^{-1})
1	0.018	0	0
2	0.02	0	0
3	0.022	-0.060856	0.010641
4	0.024	-0.261768	0.047998
5	0.026	-0.57127	0.109886
6	0.028	-0.940223	0.18938
7	0.03	-1.310504	0.275136
8	0.032	-1.624242	0.352753
9	0.034	-1.832845	0.407396
10	0.036	-1.904417	0.426998

Fig 15: 10 Lines of Simulation Output, Velocity Data of Object 1. Velocity is relative to fiducial.

ID	Time (s)	Object 2 x-velocity (ms^{-1})	Object 2 y-velocity (ms^{-1})
1	0.018	-2	0
2	0.02	-2	0
3	0.022	-1.939144	-0.010641
4	0.024	-1.738232	-0.047998
5	0.026	-1.42873	-0.109886
6	0.028	-1.059777	-0.18938
7	0.03	-0.689496	-0.275136
8	0.032	-0.375758	-0.352753
9	0.034	-0.167155	-0.407396
10	0.036	-0.095583	-0.426998

Fig 16: 10 Lines of Simulation Output, Velocity Data of Object 2. Velocity is relative to fiducial.

ID	System Momentum (kg ms^{-1})	Object 1 Energy (J)	Object 2 Energy (J)	Energy of System (J)
1	-0.0508	0	0.0508	0.0508
2	-0.0508	0	0.0508	0.0508
3	-0.0508	0.001545	0.049257	0.050802
4	-0.0508	0.006647	0.044156	0.050803
5	-0.0508	0.014508	0.036296	0.050804
6	-0.0508	0.023879	0.026925	0.050804
7	-0.0508	0.033284	0.01752	0.050804
8	-0.0508	0.041255	0.00955	0.050804
9	-0.0508	0.046555	0.00425	0.050806
10	-0.0508	0.048376	0.002432	0.050808

Fig 17: 10 Lines of Simulation Output, Momentum and Energy Data. System momentum is calculated by adding the individual momentum vectors of each object. Energy refers solely to kinetic energy of the system. Refer to Fig 16 for time values of each index. The y-component of the total momentum is always 0 and is thus not shown.

Object 1 initial velocity (ms^{-1})	(0,0)
Object 1 final velocity (ms^{-1})	(-1.904417, +0.426998)
Object 2 initial velocity (ms^{-1})	(-2,0)
Object 2 final velocity (ms^{-1})	(-0.095583, -0.426998)

Fig 18: Initial and Final Conditions of Simulation. Velocity vectors are displayed in coordinate pair (x, y). Residuals of the final velocities are below 1%.

The time step chosen for this simulation was 0.00001 s. This specific time step was chosen such that the residuals yielded were less than 1%. The program started the simulation before the moment of collision to distinguish the motion of the objects before and during the collision, to ensure that the graphs can be displayed to show when the collision takes place. The program ended after the difference between the center of masses of the two objects exceeded twice their radii, as no force acts on the balls after this criterion is met. The program continued to generate data after to ensure the graph could visually represent the change after the collision.

VII 2D Moving Target Demo

The following is a graph of a two-dimensional collision between two objects with both objects in motion:

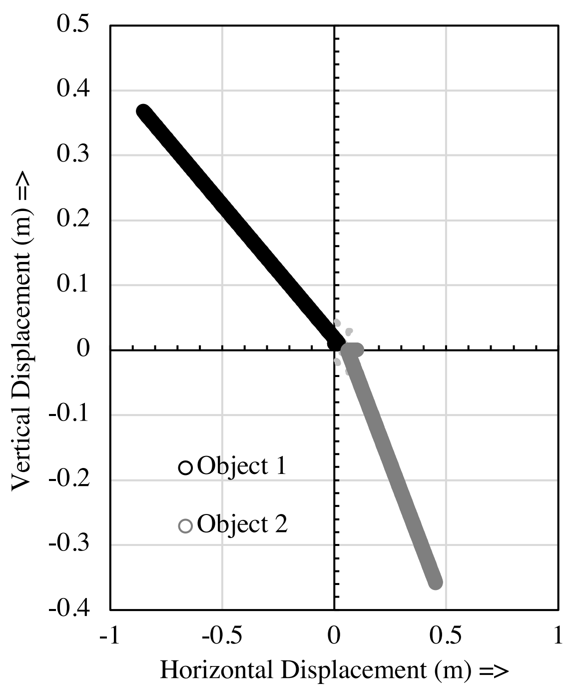


Fig 19: Vertical Versus Horizontal Position of Two Objects. Dimmed lines represent non-compressed boundaries of both objects during contact.

The following tables consist of ten lines of the simulation output:

ID	Time (s)	Object 1 x-position (m)	Object 1 y-position (m)
1	0.012	0.0121	0.01
2	0.014	0.014098	0.01
3	0.016	0.015938	0.010029
4	0.018	0.017221	0.010167
5	0.02	0.017612	0.0105
6	0.022	0.016918	0.0111
7	0.024	0.015122	0.012004
8	0.026	0.012379	0.013196
9	0.028	0.008991	0.014601
10	0.03	0.005353	0.016094

Fig 20: 10 Lines of Simulation Output, Position Data of Object 1. Positions indicate the centre of mass, with the fiducial set at 0.

ID	Time (s)	Object 2 x-position (m)	Object 2 y-position (m)
1	0.012	0.0758	0
2	0.014	0.071802	0
3	0.016	0.067962	-0.000029
4	0.018	0.064679	-0.000167
5	0.02	0.062288	-0.0005
6	0.022	0.060982	-0.0011
7	0.024	0.060778	-0.002004
8	0.026	0.061521	-0.003196
9	0.028	0.062909	-0.004601
10	0.03	0.064547	-0.006094

Fig 21: 10 Lines of Simulation Output, Position Data of Object 2. Positions indicate the centre of mass, with the fiducial set at 0.

ID	Time (s)	Object 1 x-velocity (ms^{-1})	Object 1 y-velocity (ms^{-1})
1	0.012	1	0
2	0.014	0.990404	0.001642
3	0.016	0.813598	0.034264
4	0.018	0.441064	0.110813
5	0.02	-0.066846	0.22836
6	0.022	-0.628474	0.3748
7	0.024	-1.154558	0.527828
8	0.026	-1.562844	0.65807
9	0.028	-1.790691	0.73621

10	0.03	-1.825415	0.748689

Fig 22: 10 Lines of Simulation Output, Velocity Data of Object 1. Velocity is relative to fiducial.

ID	TIME (s)	OBJECT 2 X-VELOCITY (ms^{-1})	OBJECT 2 Y-VELOCITY (ms^{-1})
1	0.012	-2	0
2	0.014	-1.990404	-0.001642
3	0.016	-1.813598	-0.034264
4	0.018	-1.441064	-0.110813
5	0.02	-0.933154	-0.22836
6	0.022	-0.371526	-0.3748
7	0.024	0.154558	-0.527828
8	0.026	0.562844	-0.65807
9	0.028	0.790691	-0.73621
10	0.03	0.825415	-0.748689

Fig 23: 10 Lines of Simulation Output, Velocity Data of Object 2. Velocity is relative to fiducial.

ID	SYSTEM MOMENTUM (kg ms^{-1})	OBJECT 1 ENERGY (J)	OBJECT 2 ENERGY (J)	ENERGY OF SYSTEM (J)
1	-0.0254	0.012706	0.050794	0.0635
2	-0.0254	0.01274	0.050761	0.063501
3	-0.0254	0.012802	0.050699	0.063501
4	-0.0254	0.012894	0.050608	0.063502
5	-0.0254	0.013014	0.050488	0.063502
6	-0.0254	0.013163	0.05034	0.063502
7	-0.0254	0.01334	0.050163	0.063503
8	-0.0254	0.013545	0.049958	0.063503
9	-0.0254	0.013778	0.049726	0.063504
10	-0.0254	0.014038	0.049466	0.063504

Fig 24: 10 Lines of Simulation Output, Momentum and Energy Data. System momentum is calculated by adding the individual momentum vectors of each object. Energy refers solely to kinetic energy of the system. Refer to Fig 23 for time values of each index. The y-component of the total momentum is always 0 and is thus not shown.

Object 1 initial velocity (ms^{-1})	(+1,0)
Object 1 final velocity (ms^{-1})	(-1.803759, +0.742209)
Object 2 initial velocity (ms^{-1})	(-2,0)
Object 2 final velocity (ms^{-1})	(+0.803759, -0.742209)

Fig 25: Initial and Final Conditions of Simulation. Velocity vectors are displayed in coordinate pair (x, y). Residuals of the final velocities are below 1%.

The time step chosen for this simulation was 0.00001 s. This specific time step was chosen such that the residuals yielded were less than 1%. The program started the simulation before the moment of collision to distinguish the motion of the objects before and during the collision, to ensure that the graphs can be displayed to show when the collision takes place. The program ended after the difference between the center of masses of the two objects exceeded twice their radii, as no force acts on the balls after this criterion is met. The program continued to generate data after to ensure the graph could visually represent the change after the collision.

VIII BONUS

For the spheres to begin rotating when they initially are not, a torque must be exerted on the spheres during the collision. The normal force that the spheres exert on each other only act perpendicular to the tangent between the two spheres and thus act parallel to the radius at the point of contact, meaning that no torque was caused by this normal force. Since the only other force that acts on the two spheres in the collision is friction, it must be what solely causes the rotation of the spheres. Thus, for this part of the simulation, it was no longer possible to assume that friction is negligible. According to Engineering Toolbox, the coefficient of friction (μ) is around 0.5. For the simulation, the velocities of the spheres at their point of contact was calculated and their components in the direction of the tangent of the two circles was found. The relative speeds along the tangents was used to determine the direction of the frictional force on each sphere. The magnitude of the frictional force is calculated as the product of the coefficient of friction and the normal force the balls exert on each other. In the simulation, collisions where both objects are initially rotating as well as those where both objects are initially moving are handled.

For each of the simulations, the angular momentum of each sphere as well as the total angular momentum were taken with respect with the origin, rather than the center of mass.

The following is a graph of a two-dimensional collision with rotation between two moving objects, as well as the angular momentum of the system:

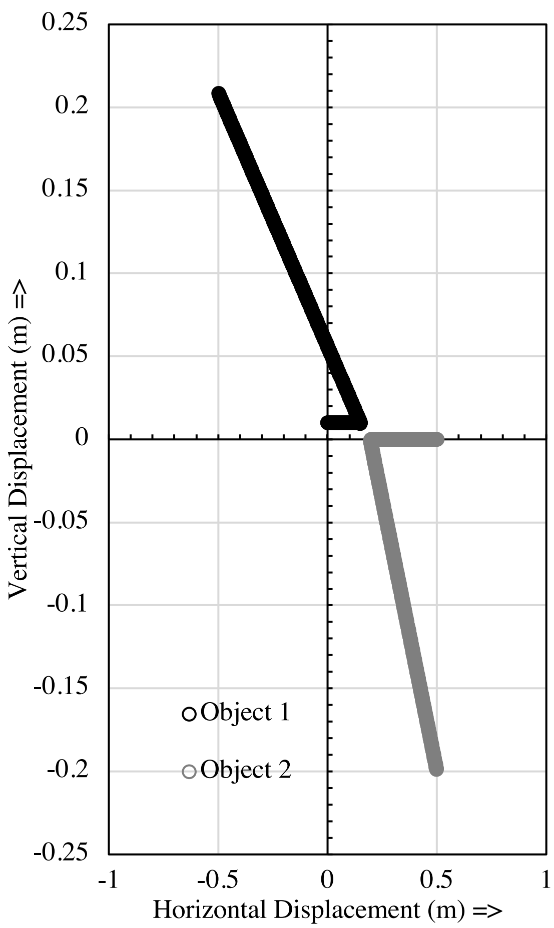


Fig 26: Vertical Versus Horizontal Position of Two Objects. Dimmed lines represent non-compressed boundaries of both objects during contact.

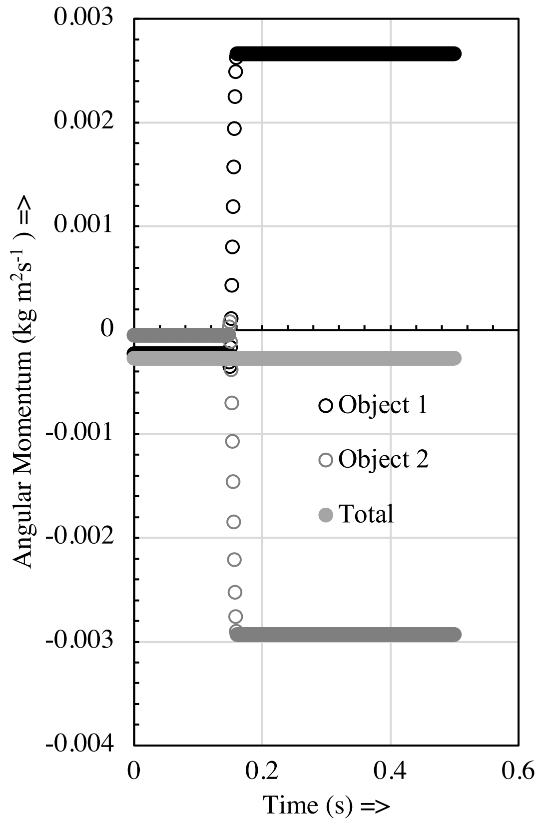


Fig 27: Angular Momentum of the System. Momentums are relative to fiducial. Total angular momentum is conserved.

The following consists of 10 lines of simulation output, including the total angular momentum as it is non-trivial.

ID	Object 1 X- position (m)	Object 1 Y-position (m)	Object 2 X-position (m)	Object 2 Y-position (m)
1	0.14501	0.01	0.20998	0
2	0.14701	0.01	0.20598	0
3	0.148847	0.009957	0.202143	0.000043
4	0.149965	0.009968	0.199025	0.000032
5	0.149906	0.010211	0.197084	-0.000211
6	0.148479	0.010755	0.196511	-0.000755
7	0.14581	0.011602	0.19718	-0.001602
8	0.142315	0.012671	0.198675	-0.002671

9	0.138564	0.013818	0.200426	-0.003818
10	0.134809	0.014966	0.202181	-0.004966

Fig 28: 10 Lines of Simulation Output, Position Data. Positions indicate the centre of mass, with the fiducial set at 0.

ID	Object 1 x-velocity (ms^{-1})	Object 1 y-velocity (ms^{-1})	Object 2 x-velocity (ms^{-1})	Object 2 y-velocity (ms^{-1})
1	1	0	-2	0
2	0.998121	-0.000573	-1.998121	0.000573
3	0.784116	-0.035149	-1.784116	0.035149
4	0.29365	0.055231	-1.29365	-0.055231
5	-0.369026	0.192099	-0.630974	-0.192099
6	-1.047578	0.351761	0.047578	-0.351761
7	-1.585188	0.49021	0.585188	-0.49021
8	-1.858349	0.567008	0.858349	-0.567008
9	-1.877703	0.574214	0.877703	-0.574214
10	-1.877703	0.574214	0.877703	-0.574214

Fig 29: 10 Lines of Simulation Output, Velocity Data. Velocity is relative to fiducial.

ID	Object 1 x-Momentum ($kg\ ms^{-1}$)	Object 1 y-Momentum ($kg\ ms^{-1}$)	Object 2 x-Momentum ($kg\ ms^{-1}$)	Object 2 y-Momentum ($kg\ ms^{-1}$)
1	0.0254	0	-0.0508	0
2	0.025357	-0.000013	-0.050757	0.000013
3	0.019966	-0.000878	-0.045366	0.000878
4	0.007542	0.001425	-0.032942	-0.001425
5	-0.009275	0.004904	-0.016125	-0.004904
6	-0.026538	0.008875	0.001138	-0.008875
7	-0.040217	0.012411	0.014817	-0.012411
8	-0.047185	0.014405	0.021785	-0.014405
9	-0.047694	0.014585	0.022294	-0.014585
10	-0.047694	0.014585	0.022294	-0.014585

Fig 30: 10 Lines of Simulation Output, Linear Momentum Data. Linear momentum of each object is established by its x and y components.

ID	System x-Momentum ($kg\ ms^{-1}$)	System y-Momentum ($kg\ ms^{-1}$)	Object 1 Angular Velocity (ω) ($rad\ s^{-1}$)	Object 2 Angular Velocity (ω) ($rad\ s^{-1}$)
1	-0.0254	0	3	-5
2	-0.0254	0	2.92779	-5.07221
3	-0.0254	0	-2.67484	-10.67483
4	-0.0254	0	-3.03146	-11.03146
5	-0.0254	0	-3.25993	-11.25992
6	-0.0254	0	-3.03040	-11.03039
7	-0.0254	0	-2.67295	-10.67294
8	-0.0254	0	-2.31202	-10.31201
9	-0.0254	0	-2.16254	-10.16254
10	-0.0254	0	-2.16254	-10.16254

Fig 31: 10 Lines of Simulation Output, System Linear Momentum and Angular Velocity Data. System linear momentum of each object is established by its x and y components. Linear momentum on each axis is conserved. Angular velocity is positive when it is rotating counterclockwise.

ID	Object 1 Angular Momentum ($kg\ m^2s^{-1}$)	Object 2 Angular Momentum ($kg\ m^2s^{-1}$)	Total Angular Momentum ($kg\ m^2s^{-1}$)
1	-0.000226	-0.000046	-0.000273
2	-0.000228	-0.000044	-0.000273
3	-0.000354	0.000081	-0.000273
4	0.000111	-0.000384	-0.000273
5	0.000801	-0.001073	-0.000272
6	0.001574	-0.001847	-0.000272
7	0.002251	-0.002523	-0.000272
8	0.002627	-0.002899	-0.000272
9	0.00266	-0.002932	-0.000272
10	0.00266	-0.002932	-0.000272

Fig 32: 10 Lines of Simulation Output, Angular Momentum Data. Angular momentum of each object and the system is calculated with respect to the fiducial.

ID	Object 1 Energy (J)	Object 2 Energy (J)	Energy of System (J)
1	0.012742	0.050916	0.063658
2	0.01276	0.050896	0.063655
3	0.015066	0.048223	0.063289

4	0.02128	0.042036	0.063316
5	0.029681	0.033636	0.063317
6	0.038314	0.025006	0.06332
7	0.045185	0.018169	0.063355
8	0.048716	0.014697	0.063414
9	0.048986	0.01445	0.063436
10	0.048986	0.01445	0.063436

Fig 33: 10 Lines of Simulation Output, Energy Data. Energy refers to both kinetic and rotational energies.

Object 1 initial velocity (ms^{-1})	(+1, 0)
Object 1 final velocity (ms^{-1})	(-1.877703, +0.574214)
Object 2 initial velocity (ms^{-1})	(-2, 0)
Object 2 final velocity (ms^{-1})	(+0.87703, -0.574214)
Object 1 initial angular velocity (rad s^{-1})	+3
Object 1 final angular velocity (rad s^{-1})	-2.162544
Object 2 initial angular velocity (rad s^{-1})	-5
Object 2 final angular velocity (rad s^{-1})	-10.162544

Fig 34: Initial and Final Conditions of Simulation. Linear velocity is represented with axial components. Angular velocity is positive when the object is rotating counterclockwise.

The time step chosen for this simulation was 0.00001 s. This specific time step was chosen such that the observed change in both angular momentum and energy are trivial (change of less than 1%). The program started the simulation before the moment of collision to distinguish the motion of the objects before and during the collision, to ensure that the graphs can be displayed to show when the collision takes place. The program ended after the difference between the center of masses of the two objects exceeded twice their radii, as no force acts on the balls after this criterion is met. The program continued to generate data after to ensure the graph could visually represent the change after the collision.

As there is no readily available simulation online that includes the rotation, spring effects, and friction of a collision between two circles, no verification of the program can be done. However, the group constructed a different, independent program that utilizes a different process to compute the collision of 2D objects using Java's Geometry library with prebuilt collision methods, and outputs were produced that had a deviation of within 1% compared to the produced outputs of the simulation.

	SIMULATION	COLLISION LIBRARY	RESIDUALS (%)
Final velocity of object 1	(-1.877703, +0.574214) ms^{-1}	(-1.882, +0.575) ms^{-1}	(0.23, 0.14)
Final velocity of object 2	(+0.87703, -0.574214) ms^{-1}	(+0.882, -0.575) ms^{-1}	(0.56, 0.14)
Final angular velocity of object 1	-2.162544 $rads^{-1}$	-2.177 $rads^{-1}$	0.69
Final angular velocity of object 2	-10.162554 $rads^{-1}$	-10.177 $rads^{-1}$	0.14

Fig 35: Linear Velocity of Simulation Compared to Collision Library. Linear velocity is represented with axial components. Angular velocity is positive when the object is rotating counterclockwise. Residuals are calculated for each component of velocity.

IX SOURCES

Engineering ToolBox, (2004). Friction and Friction Coefficients for various Materials. [online] Available at: https://www.engineeringtoolbox.com/friction-coefficients-d_778.html [Accessed 14 10. 2019].

Cutnell, J. D., and Johnson, K.W., "Physics (9th ed.)", 2000