Introduction to Impact Mechanics using MATLAB

Impact mechanics is concerned with the reaction forces that develop during a collision and the dynamic response of a system to these forces. The subject has a wide range of engineering applications, from designing sports equipment to improving the crash-worthiness of automobiles. A good reference text is Impact Mechanics by W.J. Stronge.

There are many different formulations for analyzing impacts. For the purposes of this lab, rigid body impact theory will be briefly discussed.

Impact Theory

Consider two rigid bodies, A and B, that are in collision, as shown in Figure 1. The contact point on body A is C, and the contact point for body B is C’. The normal force $F$ exerted by this collision will go through $C$ and $C’$, which are coincident at the instant of collision. The way in which the forces interact with the rigid bodies is important in the understanding of impact mechanics.

![Figure 1: Collision of two rigid bodies](image)

There are two phases in the collision of these two bodies. In the compression phase of the impact, the kinetic energy of the body is converted to deformation. The velocity is decreased by the increasing normal force exerted by the other body. The compression phase ends when the normal relative velocities of the contact points vanishes due to the increasing force. The restitution phase then begins, and the strain energy of the deformed bodies pushes the bodies apart. This restores part of the relative velocity of the bodies. Unless the impact speed is very small, there is energy dissipation due to the collision. This can be due to plastic deformation, elastic vibration, viscoplasticity, or other effects. This dissipation means that the total energy absorbed during the compression phase is more than the energy released in the restitution phase. The ratio of the energies before and after collision describe a value called the energetic coefficient of restitution $e$, expressed as


e = \frac{\text{energy released}}{\text{energy absorbed}}

The value of $e$ ranges from $0 < e < 1$. In the case that $e = 0$ the two bodies do not separate after collision. In the case that $e = 1$, the collision is perfectly elastic and the departing kinetic energy is equal to before the
collision. For rigid bodies, it can be shown that the coefficient of restitution is equal to the ratio of incoming and departing velocities

\[ e = \frac{v_0}{v_f} \]

This knowledge can be used to create a simulation in a simple example.

**Example - A Bouncing Basketball**

Let us apply the theory of impact mechanics to an example of a bouncing basketball, and try to simulate the motion over time. First, the governing equations are needed. Some assumptions to simplify our analysis can be made.

1) The ball is a perfect sphere;
2) There are no defects in the material;
3) The diameter and mass can be averaged;
4) The mass of the ball acts at the center of the ball;
5) Air resistance is neglected; and
6) The ball is not rotating.

The ball can be modeled as shown in Figure 2. The ball has a mass located at the center of mass. While the ball is considered to be a rigid body, the ball has a linear stiffness element \( k \) to represent the deformation during contact. Also, when the ball hits the ground the effect of the coefficient of restitution can be replaced with a linear damper. This makes sense because, as we know, a damper element dissipates energy.

![Figure 2: Basketball model](image)

There are two cases for the governing equations that need to be considered:

Case 1) The ball is in the air;
Case 2) The ball is in contact with the ground.

The free body diagram for each case is shown in Figures 3 and 4, respectively.

In the first case, a force balance for the free-falling ball gives the following governing equation:

\[ \ddot{x} = g \]

In the second case, the force balance yields

\[ m\ddot{x} + b\dot{x} + kx = 0 \]
where \( g \) is the acceleration due to gravity, \( b \) is the damping coefficient and \( k \) is the ball spring constant. The ground acts as a displacement source and does not contribute a non-conserved force. The second equation can be written in terms of state variables in first-order form by letting 

\[
\begin{align*}
\dot{x}_1 &= x_2; \\
\dot{x}_2 &= -\frac{b}{m}x_2 - \frac{k}{m}x_1.
\end{align*}
\]

The first equation can also be written in terms of state variables. (This is left to the reader to complete as an exercise.) Now that the governing equations have been found, a propagation simulation can be found using MATLAB. Our goal is to use the ODE45 command and have something of the form:

\[
\text{[time, X_matrix]} = \text{ode45(@function_name, time_range, [initial_conditions])}
\]

The state variables have been defined as \( x_1 = \text{position} \) and \( x_2 = \text{velocity} \). The function that is created will need to describe the first-order differential equations to be integrated to yield position and velocity.

The challenge in creating the function comes from switching between Case 1) and Case 2). This can be achieved in MATLAB with the following pseudo code:

\textbf{if} \hspace{0.5em} \text{position} < \hspace{0.5em} \text{a very small value}
\[ x_2 = -\frac{b}{m} x_2 - \frac{k}{m} x_1; \]

```
else
  \[ x_2 = -g; \]
end
```

This switching can lead to numerical integration problems if the change in the acceleration is large when position is close to the contact point. There are a few ways to deal with this switching problem if you have trouble with \texttt{ode45}:

- use a (nonlinear) stiffness element that starts very soft and quickly reaches the stiffness of the deformable body;
- relax the error tolerance on \texttt{ode45};
- use a lower-order solver such as \texttt{ode23} which is more robust (at the expense of a higher truncation error); or
- use a fixed time-step size integrator (with a small step size) so that the adaptive solver does not hunt for a solution around the switching point.

These workarounds either change the model of the physics or mess with the ODE solution method to allow the simulation to be executed. For any model, verification (solving the right physics) and validation (solving the physics right) are both important, within an acceptable amount of error.

**Exercise**

Create the function in an m-file that takes time and an \( X \) matrix of two columns (position and velocity) as the input and returns \( \dot{x}_1 \) and \( \dot{x}_2 \) for \texttt{ode45} to integrate.

Plot the motion of the bouncing basketball over a duration of 5 seconds. Assume that the initial velocity is zero and the initial height is \( 0.87 \text{m} \). Use the following parameters for the ball: damping coefficient \( b = 1.526 \text{ Ns/m} \), linear spring constant \( k = 974.46 \text{ N/m} \), mass \( m = 0.596 \text{ kg} \) and the diameter \( d = 0.240 \text{ m} \).

To see the effect of solving a stiff set of equations, try the exercise again using a high value of damping coefficient (eg \( b=1526 \text{ Ns/m} \)). Does using a stiff solver or relaxing the error tolerance help?