Freely Falling Ball

2022/07/07 Y. T. Huang

The content of this document contains the solution of freely falling ball as an example to explain how to use the “event” and “matlabFunction” in MATLAB.

Consider a ball freely falling from a height of "h" meters. Find the position and velocity as a function of time. In this document, this question will be solved by the principle of least action, following the steps same as solving R-SLIP model.

The first step is to define the generalized coordinates of the system. In this case, the problem is a 1-D problem. So, the coordinate is:

$$\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$$

And the corresponding codes are as following:

```matlab
y = sym('y', [2, 1], 'real'); % y = [position; velocity]
q = y(1); q_dot = y(2); % POLA generalized coordinate
```

After define the coordinates, the action L of ball can be written:

$$L = \frac{1}{2} m\dot{q}^2 - mgq$$

Derive the dynamic equation of the falling ball by Lagrangian method. Ignore air resistance, the Euler-Lagrange equation is as following:

$$\frac{\partial L}{\partial q} - \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) = 0$$

Expand the above formula with the chain row, finding the state space equation of the system:

$$\frac{\partial L}{\partial q} - \frac{\partial}{\partial q}\left(\frac{\partial L}{\partial \dot{q}}\right) q - \frac{\partial}{\partial \dot{q}}\left(\frac{\partial L}{\partial \dot{q}}\right) \ddot{q} = 0$$

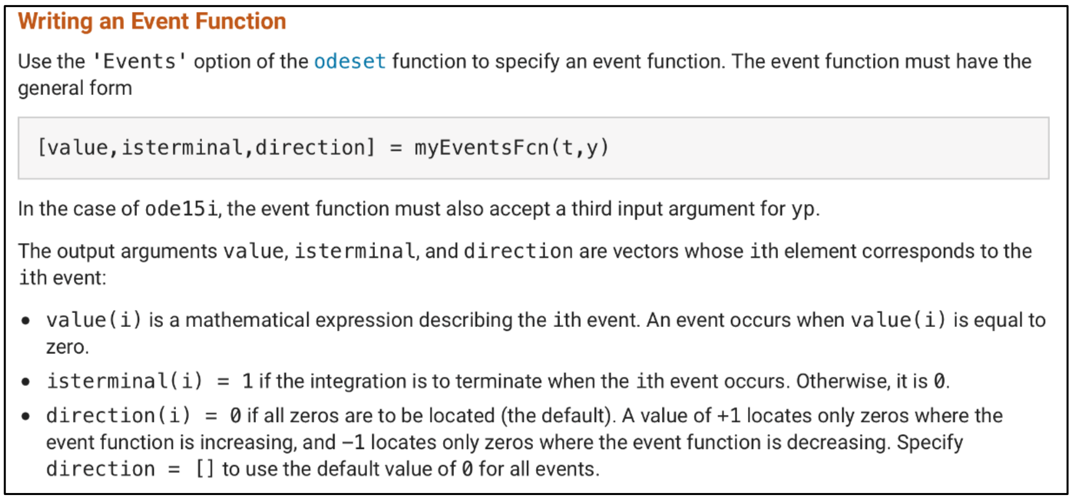
$$\ddot{q} = f(\dot{q}, \ddot{q}) = \left[\frac{\partial}{\partial \dot{q}}\left(\frac{\partial L}{\partial \dot{q}}\right)\right]^{-1} \left[\frac{\partial L}{\partial q} - \frac{\partial}{\partial q}\left(\frac{\partial L}{\partial \dot{q}}\right) \dot{q}\right]$$

$$\frac{d}{dt}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ f(\dot{q}, \ddot{q}) \end{bmatrix}$$

And the corresponding codes are as following:

```matlab
L = m*(q_dot'*q_dot)/2 - (m*g*q); % Lagrangian = T - V
E_L_eqn = [q_dot; jacobian(L, q)' - jacobian(jacobian(L, q_dot), q)*q_dot];
```

The final step to generate the needed function of this problem is using matlabFunction to generate the event function and the ODE of the system. Before start this step, lets take a look at the use of event function:



Writing an Event Function

Use the `'Events'` option of the `odeset` function to specify an event function. The event function must have the general form

```
[value,isterminal,direction] = myEventsFcn(t,y)
```

In the case of `ode15i`, the event function must also accept a third input argument for `yp`.

The output arguments `value`, `isterminal`, and `direction` are vectors whose `i`th element corresponds to the `i`th event:

- `value(i)` is a mathematical expression describing the `i`th event. An event occurs when `value(i)` is equal to zero.
- `isterminal(i) = 1` if the integration is to terminate when the `i`th event occurs. Otherwise, it is `0`.
- `direction(i) = 0` if all zeros are to be located (the default). A value of +1 locates only zeros where the event function is increasing, and −1 locates only zeros where the event function is decreasing. Specify `direction = []` to use the default value of `0` for all events.

Event function has three outputs: “value”, “isterminal”, “direction”. So, we have to generate a function with these three outputs. Value is the position in y coordinate. And this event has two variables: “t” and “y”. Has this information, we can start to generate these functions.

```
syms t real
matlabFunction(simplify(E_L_eqn), 'file', ...
 'ODE_falling_ball_matlabFunction', 'vars', {t, y});
matlabFunction(y(1), 1, -1, 'file', 'Event_TD_matlabFunction', ...
    'vars', {t, y}, 'outputs', {'value', 'isterminal', 'direction'});
```

The generated functions are shown below:

```
function out1 = ODE_falling_ball_matlabFunction(t,in2)
%ODE_falling_ball_matlabFunction
%    OUT1 = ODE_falling_ball_matlabFunction(T,IN2)

%    This function was generated by the Symbolic Math Toolbox version 9.0.
%    08-Jul-2022 03:08:08

y2 = in2(2,:);
out1 = [y2;-9.81e+2./1.0e+2];
```

```
function [value,isterminal,direction] = Event_TD_matlabFunction(t,in2)
%Event_TD_matlabFunction
%    [VALUE,ISTERMINAL,DIRECTION] = Event_TD_matlabFunction(T,IN2)

%    This function was generated by the Symbolic Math Toolbox version 9.0.
%    08-Jul-2022 03:08:08

y1 = in2(1,:);
value = y1;
if nargout > 1
    isterminal = 1.0;
end
if nargout > 2
    direction = -1.0;
end
```

The function “ODE_falling_ball” has the input “t” and “y”. By solving $f(\dot{q},\ddot{q})$, we can find the result is $\ddot{y} = g$, which is the same as the solution that matlabFunction given. The event function is an indicator, tell the ODE solver when to stop solving the equation. In this problem, the event is the ball touch the ground, or $y = 0$.

Construct these functions manually so that we can check the result given by matlabFunction is correct.

```
function dydt = ODE_falling_ball(t,y)
    g = 9.81;
    dydt = [y(2); -g]; % dy = y(2) , dy/dt = -g
end
```

```
function [value, isterminal, direction] = event_TD(t,y)
    value = y(1);
    isterminal = 1;
    direction = -1;
end
```

An event function has three outputs. Which are value, the value that we want to be zero; isterminal, decide the ode45 solver will stop or keep going; and direction, decide the solving direction (+, -), respectively.

Now, we have all needed functions. The next step is to combine them and find the solution. First, set up some parameters that influence the solving interval or precision. Add all optional conditions into “odeoption”. This will be used when solving the ODE. Shown as below:

```
tspan = [0 5];
initial_condition = [h; 0]; %[ position; velocity ]
reltol = 10^(-7);   abstol = 10^(-7);
odeoption = odeset( 'event', @Event_TD_matlabFunction, ...
        'reltol', reltol, 'abstol', abstol );
```

Solve the falling ball’s position and velocity by ode45:

```
[t,y] = ode45( @ODE_falling_ball_matlabFunction, ...
 tspan, initial_condition, odeoption );
```

The result is shown below. The blue point is the position numerical solution. The red point is the velocity numerical solution. The yellow line is the position analytic solution.

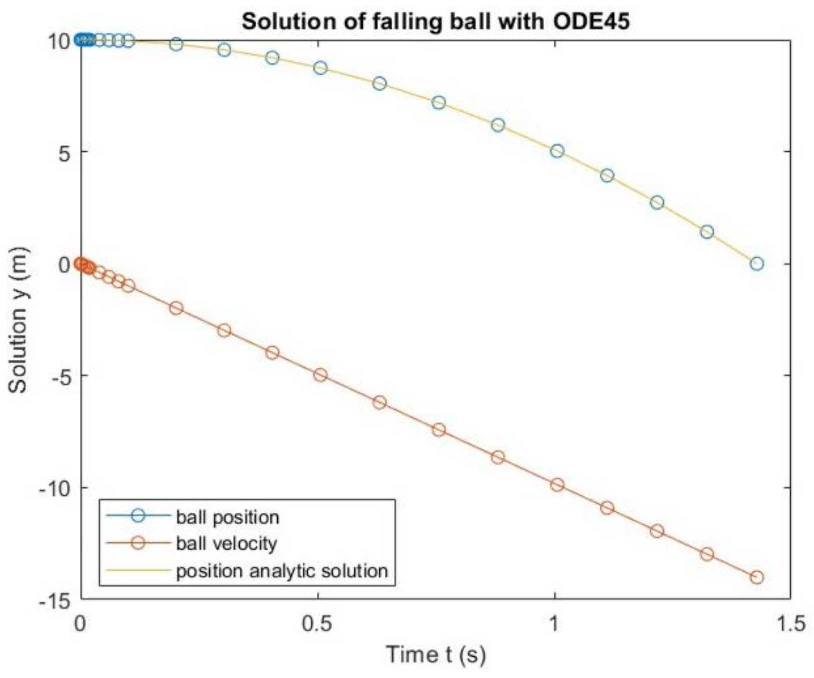


Fig.1 Solution of falling ball

The final figure is the result that use the manually written function. Compare with figure 1, the result is almost same.

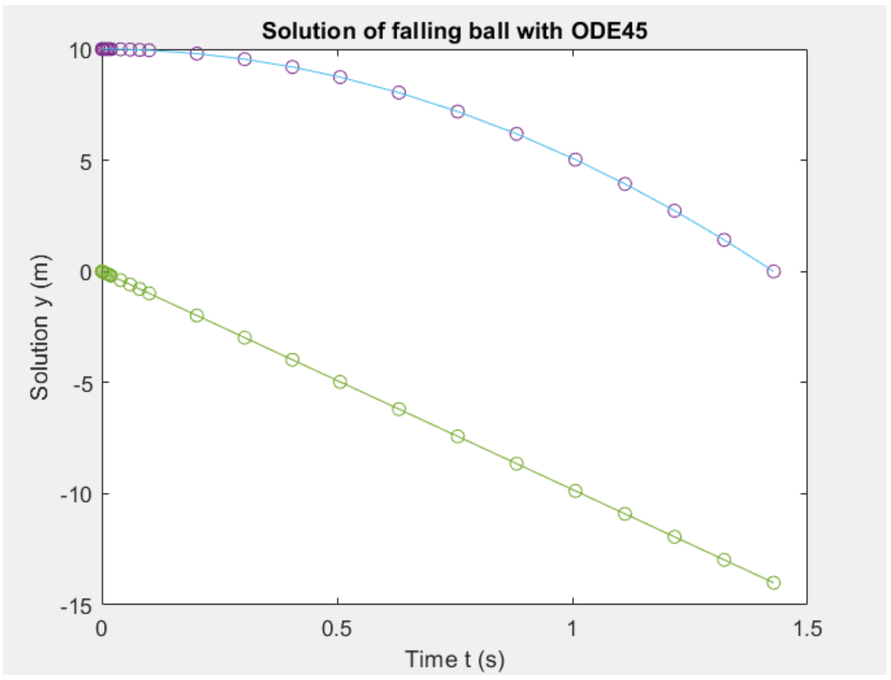


Fig.2 Solution of falling ball by manually written function