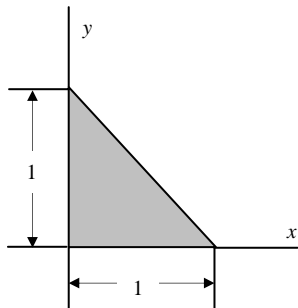


NEEP 602 -- Engineering Problem Solving II Exercise 3

Event Location in Matlab

Sometimes in the course of solving a particular problem we need to change what we're doing or stop what we're doing, because certain conditions have been reached. We might need to or want to terminate the solution, or perhaps we need to change the IVP that we are solving for the next part of the process we are simulating. Matlab (as well as other ODE solvers) offers a capability called *event location* for identifying the time t^* at which an event occurs (i.e., the specified conditions are met), and also the value of the solutions $w(t^*)$ at that time. This is done by defining one or more event functions whose values go to zero when the event occurs. Let's look at the example of a ball bouncing down a ramp as shown. We'll start by dropping the ball from a height of 1 meter above the top of the ramp.



The equations of motion for the ball in free fall are

$$\frac{d^2 x}{dt^2} = 0$$

$$\frac{d^2 y}{dt^2} = -g$$

The first event that we are interested in is when the ball hits the ramp and bounces. If $y(t)$ is the height of the ball, then when it hits the ramp we'll have

$$y(t^*) = 1 - x(t^*)$$

(note that this is the equation of the line that defines the upper surface of the ramp.) So our first event function will be

$$g_1 = y(t^*) - (1 - x(t^*)) = 0$$

After the ball hits the ramp, we'll have the same equations of motion, but a different IVP, with new initial conditions, which will take into account the effects of the bounce. For a coefficient of restitution k , the new initial conditions will be $[x(t^*) \quad -ky'(t^*) \quad y(t^*) \quad kx'(t^*)]$. One thing to consider for these new initial conditions is that the ball is in contact with the ramp, so g_1 will be 0 here. We can use a vector called *direction* to tell the solver whether or not it matters if the event function is increasing or decreasing at the event. Setting *direction* to -1 means that the solver will only report an event if the event function is decreasing. For our example, g_1 will be increasing if the ball is bouncing. Therefore, the solver will not report another event and will begin solving the new IVP.

The other event we need to locate is when the ball reaches the end of the ramp, where $x(t^*) = 1$. Thus, our second event function is

$$g_2 = x(t^*) - 1 = 0$$

This event will terminate the integration.

Depending on the value of k and the height from which the ball is dropped, the bounces may cluster until the ball stops and rolls down the ramp. The equations we have don't model that motion (how could we do that?), so we stop the solution when the time between successive bounces differs by less than 1%. We also want to stop when the ball bounces very close to the end of the ramp, so we'll stop when the ball bounces at a spot within 1% of the length of the ramp from the end.

```
%Bouncing Ball
%ball at x,y
%w(1) = x, w(2) = x', w(3) = y, w(4) = y'
global k
winit = [0 0 2 0];
% Draw the ramp
fill([0 0 1],[0 1 0], [0.8,0.8,0.8]);
axis([-0.1 1.1 0 winit(3)])
hold on
plot(0,winit(3),'ro'); % mark the drop point with a red circle

options = odeset('Events',@events1);
% store the points on the path in xplot and yplot
xplot = [];
yplot = [];
tstar = 0;
while 1
    % use enough points for a smooth plot
    tspan = linspace(tstar, tstar+1,20);
    [t,w,te,we,ie] = ode45(@ballodes,tspan,winit,options);
    xplot = [xplot; w(:,1)];
    yplot = [yplot; w(:,3)];
    if isempty(ie) %if no events, no bounce - extend the interval
        tstar = t(end); %new start time
        winit = w(end,:); %new initial conditions
    elseif ie(end)==1 %event 1 - ball bounced
        % mark the bounce point with a red *
        plot(we(end,1),we(end,3),'r*');
        if (te(end)-tstar)<0.01*tstar %bounces cluster,ball rolls down
            text(.6,1.4,'Bounces cluster') %print message at (0.6,1.4)
            break
        end
        if abs(we(end,1)-1)<0.01 % ball is close to bottom of ramp
            break
        end
        tstar = te(end);
        winit = [we(end,1) -k*we(end,4) we(end,3) k*we(end,2)];
    elseif ie(end)==2 % event 2, ball is at bottom of ramp
        break
    end
end
end
plot(xplot,yplot)

function f = ballodes(t,w)
```

```
f =[w(2); 0; w(4); -9.81];
```

```
function [value,isterminal,direction] = events1(t,y)
value = [y(3)-(1-y(1)); y(1)-1]; %event functions g1 and g2
isterminal = [1; 1]; %both events are terminal
%report event1 only if g1 is decreasing at new initial point
direction = [-1; 0];
```

Try this out for different values of k , different drop heights, a different ramp.