Event Location

ballode models the motion of a bouncing ball. This example illustrates the event location capabilities of the ODE solvers.

The equations for the bouncing ball are:

y'1 = y2
y'2 = -9.8

In this example, the event function is coded in a subfunction events
[value,isterminal,direction] = events(t,y)

which returns

A value of the event function

The information whether or not the integration should stop when value = 0 (isterminal = 1 or 0, respectively)

The desired directionality of the zero crossings:

-1

Detect zero crossings in the negative direction only

0

Detect all zero crossings

1

Detect zero crossings in the positive direction only

The length of value, isterminal, and direction is the same as the number of event functions. The ith element of each vector, corresponds to the ith event function. For an example of more advanced event location, see orbitode (Advanced Event Location).

In ballode, setting the Events property to @events causes the solver to stop the integration (isterminal = 1) when the ball hits the ground (the height y(1) is 0) during its fall (direction = -1). The example then restarts the integration with initial conditions corresponding to a ball that bounced.

To run this example, click on the example name, or type ballode at the command line.

```
function ballode
%BALLODE  Run a demo of a bouncing ball.

tstart = 0;
tfinal = 30;
y0 = [0; 20];
refine = 4;
options = odeset('Events',@events,'OutputFcn', @odeplot,...
                 'OutputSel',1,'Refine',refine);

set(gca,'xlim',[0 30],'ylim',[0 25]);
box on
hold on;

tout = tstart;
yout = y0.';
```

```
teout = [];
yeout = [];
ieout = [];
for i = 1:10
  % Solve until the first terminal event.
  [t,y,te,ye,ie] = ode23(@f,[tstart tfinal],y0,options);
  if ~ishold
    hold on
  end
  % Accumulate output.
  nt = length(t);
  tout = [tout; t(2:nt)];
  yout = [yout; y(2:nt,:)];
  teout = [teout; te];    % Events at tstart are never reported.
  yeout = [yeout; ye];
  ieout = [ieout; ie];

  ud = get(gcf,'UserData');
  if ud.stop
    break;
  end

  % Set the new initial conditions, with .9 attenuation.
  y0(1) = 0;
  y0(2) = -.9*y(nt,2);

  % A good guess of a valid first time step is the length of
  % the last valid time step, so use it for faster computation.
  options = odeset(options,'InitialStep',t(nt)-t(nt-refine),...
                           'MaxStep',t(nt)-t(1));
  tstart = t(nt);
end

plot(teout,yeout(:,1),'ro')
xlabel('time');
ylabel('height');
title('Ball trajectory and the events');
hold off
odeplot([],[],'done');
% --------------------------------------------------------------
function dydt = f(t,y)
dydt = [y(2); -9.8];
% --------------------------------------------------------------
function [value,isterminal,direction] = events(t,y)
% Locate the time when height passes through zero in a
% decreasing direction and stop integration.
value = y(1);     % Detect height = 0
isterminal = 1;   % Stop the integration
direction = -1;   % Negative direction only
```