ODE Solutions

Public Euler by Hand

$$\frac{dx}{dt} = xy - t + 2 \qquad x(1) = 2$$

$$\frac{dy}{dt} = \frac{3x}{y} + 5tx \qquad y(1) = -3$$

Step sizes of $h = 0.05$

<u>First step</u>

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + h_0 f(t_0, x_0, y_0)$$

$$= \begin{bmatrix} 2 \\ -3 \end{bmatrix} + 0.05 \begin{bmatrix} 2(-3) - 1 + 2 \\ \frac{3(2)}{-3} + 5(1)(2) \end{bmatrix}$$

$$= \begin{bmatrix} 2 \\ -3 \end{bmatrix} + 0.05 \begin{bmatrix} -5 \\ -2 + 10 \end{bmatrix} = \begin{bmatrix} 1.75 \\ -2.6 \end{bmatrix}$$

<u>Second step</u>

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1.75 \\ -2.6 \end{bmatrix} + 0.05 \begin{bmatrix} 1.75 \cdot (-2.6) - 1.05 + 2 \\ \frac{3(1.75)}{-2.6} + 5(1.05)(1.75) \end{bmatrix}$$

$$= \begin{bmatrix} 1.75 \\ -2.6 \end{bmatrix} + 0.05 \begin{bmatrix} -3.6 \\ 7.1683 \end{bmatrix}$$

$$= \begin{bmatrix} 1.57 \\ -2.2416 \end{bmatrix}$$

Public Golf Solutions

a)
```
function [t, y] = MyOde(f, tspan, y0, h, event)

    N = ceil( ( tspan(2) - tspan(1) ) / h ); % max number of time steps

    m = length(y0); % Number of state variables

    % Initialize output arrays (this is the largest that they could
    % need to be... we'll chop them back later if needed).
    t = zeros(N,1);
    y = zeros(N,m);

    y(1,:) = y0';
    t(1) = tspan(1);
    val = event(t(1), y(1,:));
    n = 1;

    while (n==1) || (t(n)<=tspan(2) && val>=0)

       t(n+1) = t(n) + h;

       f0 = f(t(n),y(n,:)')'; % eval RHS for Euler step
       y(n+1,:) = y(n,:) + h * f0; % take Euler step
       f1 = f(t(n+1),y(n+1,:)')'; % eval RHS at Euler step
       y(n+1,:) = y(n,:) + h/2 * ( f0 + f1 ); % modified Euler
       % Euler step
       % Modified Euler step
       % Note: The dynamics function expects the state vector to be a
       % column vector.  Its output is also a col-vect.  Hence the
       % transposes.

       n = n + 1;

       val = event(t(n), y(n,:)); % Call the events function
    end

    % Interpolate the last point if an event was flagged
    if val<0
```

```
    % Find approx crossing time
    v_prev = event(t(n-1), y(n-1,:));
    alpha = (v_prev)/(v_prev-val);
    t(n) = t(n-1) + h*alpha;  % time of zero-crossing

    % Interpolate state at last step
    y(n,:) = (1-alpha)*y(n-1,:) + alpha*y(n,:);

    %{
    % Another option is to re-do the last time step
    h_new = t(n) - t(n-1);
    f0 = f(t(n-1),y(n-1,:)')'; % eval RHS for Euler step
    y(n,:) = y(n-1,:) + h_new * f0; % take Euler step
    f1 = f(t(n),y(n,:)')'; % eval RHS at Euler step
    y(n,:) = y(n-1,:) + h_new/2 * ( f0 + f1 ); % modified Euler
    %}
end

% Extract the rows that we used (throw away extra rows)
t = t(1:n);
y = y(1:n,:);
```

b) $\begin{cases} x'' = -k\,x' \\ y'' = -g - k\,y' \end{cases}$ let $z_1 = x$, $z_2 = y$ and $z_3 = x' = z_1'$ and $z_4 = y' = z_2'$

Then our system becomes

$$\begin{cases} z_1' = z_3 \\ z_2' = z_4 \\ z_3' = -k\,z_3 \\ z_4' = -g - k\,z_4 \end{cases}$$

```
function dzdt = projectile(t, z)
   % z(1) = x(t)
   % z(2) = y(t)
   % z(3) = x'(t)
   % z(4) = y'(t)
   k = 0.1;
   dzdt = [ z(3) ; z(4) ; -k*z(3) ; -9.81-k*z(4) ];
```

c)

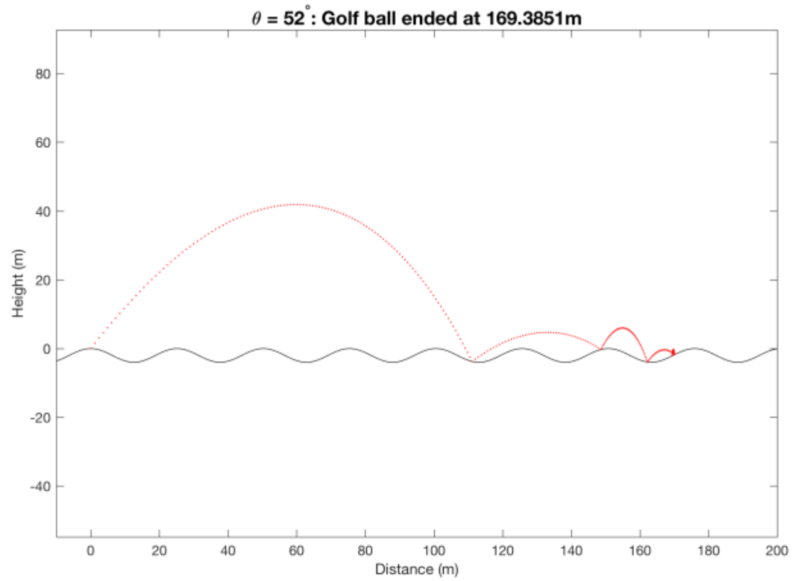
```
function value = projectile_events(t, z)

   % Event triggers when the height of the golf ball
   % (above the ground) goes from positive to
   % negative.
   value = z(2) - Ground(z(1));
```

d)

```
% golf_drive.m

theta = 52;     % Angle of initial velocity (degrees)
S = 40;         % Initial speed (m/s)
tspan = [0 30]; % Set start and end times for computation (seconds)
h = 0.05;       % time step 100ms

% Set up initial state
yStart = [0;0;                 % initial position (0,0)
        S*cos(theta/180*pi);   % x velocity
        S*sin(theta/180*pi)];  % y velocity

t_history = [];
y_history = [];

% 5 flights (in a loop, or explicitly unrolled)
```

```
for flight = 1:5
   % Call the MyOde numerical solver function

   [t,y] = MyOde(@projectile, tspan, yStart, h, @projectile_events);

   % Process first impact (bounce)
   s = GroundSlope(y(end,1));  % slope of ground

   u = [1 s];
   u = u / norm(u);  % unit tangent vector
   U = [-u(2) u(1)]; % unit normal vector
   v = y(end,3:4);   % the ball's incident velocity vector
   V = 0.75*( (v*u')*u - (v*U')*U );  % Reflected velocity vector

   % Initial state for post-bounce
   yStart = y(end,:);
   yStart(3:4) = V;
   tspan = [t(end) t(end)+30];

   t_history = [t_history ; t(1:end-1)];
   y_history = [y_history ; y(1:end-1,:)];
end

% Plot
x = linspace(-10, 200, 300);
hills = Ground(x);
plot(x, hills, 'k-');
axis equal;
hold on;

% Static plot version
plot(y_history(:,1), y_history(:,2), 'r.');
axis equal;
title(['\theta = ' num2str(theta) '^\circ: Golf ball ended at ' num2str(y(end,1)) 'm']);
xlabel('Distance (m)');
ylabel('Height (m)');

hold off;
```

e) $\theta = 52^\circ$



Public Time of Death Solution

a)

```
% time_of_death.m

% Initial Conditions [Body Temp, A, B]
T0 = [37.5 1  1];

% Try 11:30am as the time of death
tspan = [11.5 22+26/60];

% Calling the ODE solver
% Correct time of death is around 11:30am

[t, z] = ode45(@deathprocess, tspan, T0);

%% Plot solution
figure(1);
plot(t,z(:,1),'b');
xlabel('Time (h)');
ylabel('Temp (C)');

%% (c) Considering alibis

% Based on the alibis, and the fact that Robert Durst probably died
% around 11:30am, we believe that James Carver should be the main
% suspect (the only suspect that did not have a solid ailbi during
% the estimated time of the murder.




function dzdt = deathprocess(t,z)
   T = z(1);
   A = z(2);
   B = z(3);
   Agr = 0;
   if T>=29 && T<=45
      Agr = 0.0008*(T - 29)^2 * (1-exp(0.08*(T - 45)));
   end
```

```
Bgr = 0;
if T>=17 && T<=27
   Bgr = 0.001*(T - 17)^2 * (1-exp(0.05*(T - 27)));
end
dTdt = -0.2 * (T-Ta(t)) + (A + B)/100;
dKdt = Agr*A*(50-A);
dYdt = Bgr*B*(50-B);
dzdt = [dTdt; dAdt; dBdt];
```