```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MATH_151_Lab6
%-------------------------------------------------------------------------
% C Rocheleau, Colorado State University
% 10/10/2023
%-------------------------------------------------------------------------
% Answer key for MATH-151 Lab 7 for the Fall 2023 semester
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all; clear all; clc;
```

Task 1: Reaching the Bottom

First, lets look at this function so we can estimate where the minimum may be

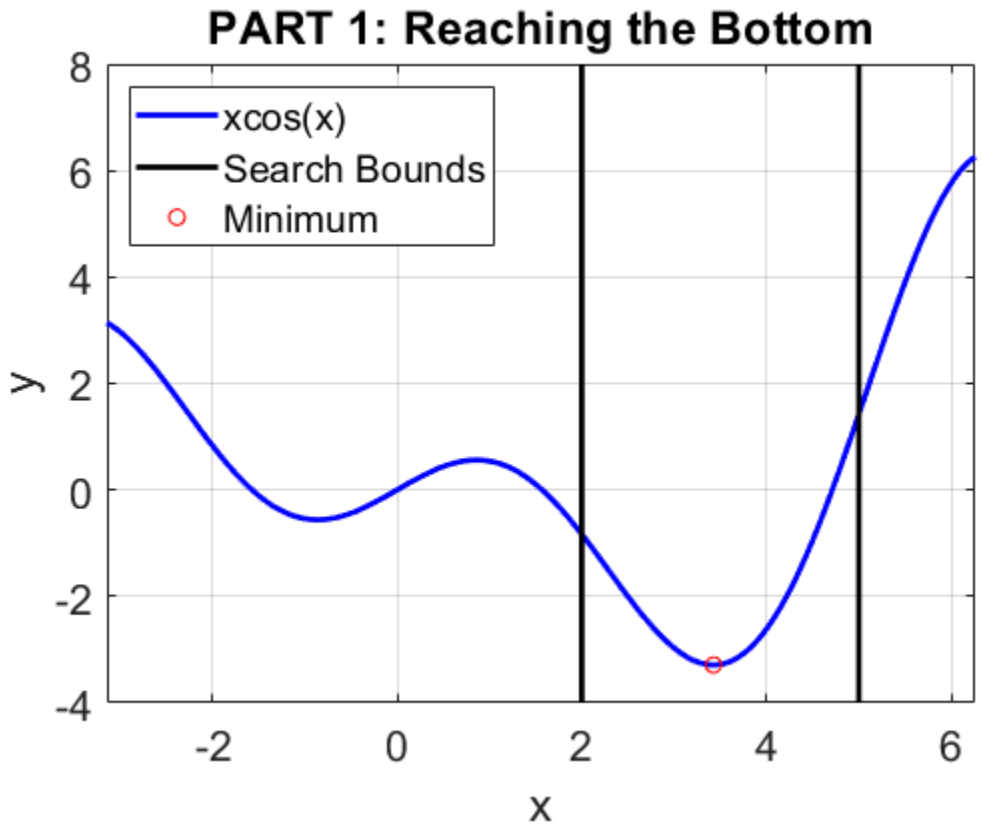
```
x  = -pi:0.05:2*pi;
fx = x.*cos(x);
figure();
plot(x, fx,'b-','linewidth',2);
grid on; hold on;

% Let's say that the minimum is between 2 and 5
xL = 2; xR = 5;
fprime_L = cos(xL) - xL*sin(xL);
fprime_R = cos(xR) - xR*sin(xR);
plot([xL xL NaN xR xR], [ylim NaN ylim], 'k-','linewidth',2)
% Lets use Bisection method to find our minimum
while (xR - xL) > 0.001
    % Compute midpoint and function at midpoint
    xM = (xR + xL)/2;
    fprime_M = cos(xM) - xM*sin(xM);

    % Check if this becomes our new left or right side
    if fprime_M*fprime_L > 0
        % Same sign as our left means it becomes our new left point
        xL = xM;
        fprime_L = fprime_M;
    else
        % Otherwise, it is our new right point
        xR = xM;
        fprime_R = fprime_M;
    end
end

% When we are done, whatever the final midpoint was, we'll use as our root
x_star = xM;
f_star = x_star*cos(x_star);
plot(x_star, f_star, 'ro');
legend('xcos(x)','Search Bounds','Minimum','location','NorthWest');
axis tight; set(gca,'fontsize',16);
title('PART 1: Reaching the Bottom','fontweight','b');
```

```
xlabel('x'); ylabel('y')
```



Task 2: See You Next Fall

```
% a) I guess it'll take 30 seconds

% Initialize our constants
g = 9.8; k = 0.0034;
% Let's find our initial locations
t_old_sec = 0; t_new_sec = 30;
y_old_sec = log( cosh( t_old_sec*sqrt(g*k) ) ) / k - 2000;
y_new_sec = log( cosh( t_new_sec*sqrt(g*k) ) ) / k - 2000;

% Lets do Secant method, and count our iterations
nSecantIts = 0;
while abs(t_new_sec - t_old_sec) > 0.001
    % Calculate new secant
    temp = t_new_sec - y_new_sec*( (t_new_sec - t_old_sec) / (y_new_sec -
 y_old_sec) );
    t_old_sec = t_new_sec; y_old_sec = y_new_sec;
    t_new_sec = temp;
    y_new_sec = log( cosh( t_new_sec*sqrt(g*k) ) ) / k - 2000;

    nSecantIts = nSecantIts + 1;
end
```

```
fprintf(['Secant Method finds a solution of ', num2str(t_new_sec), ...
        's after ', num2str(nSecantIts), ' iterations \n']);

% Now let's see what Newton's Method does
t_old_newt = 0; t_new_newt = 30;
y_new_newt = log( cosh( t_new_newt*sqrt(g*k) ) ) / k - 2000;

% Lets do Newton's method, and count our iterations
nNewtonIts = 0;
while abs(t_new_newt - t_old_newt) > 0.001
    % Calculate new secant
    y_prime = sqrt(g/k)*tanh(t_new_newt*sqrt(g*k));
    temp    = t_new_newt - y_new_newt / y_prime;
    t_old_newt   = t_new_newt;
    t_new_newt   = temp;
    y_new_newt   = log( cosh( t_new_newt*sqrt(g*k) ) ) / k - 2000;

    nNewtonIts = nNewtonIts + 1;
end

fprintf(['Newton''s Method finds a solution of ', num2str(t_new_newt), ...
        's after ', num2str(nNewtonIts), ' iterations \n']);
```

```
Secant Method finds a solution of 41.0499s after 3 iterations
Newton's Method finds a solution of 41.0499s after 2 iterations
```

Published with MATLAB® R2022b