# Curved Line Function

```
% this function plots points between a starting point and an ending point
% the starting point is defined by x1, y1
% the ending point is defined by x2, y2
% 1/ptsPerRadian (points per radian) is the increment used to plot the
% points along the circumference of a circle

function xy = CurvedLine(x1, y1, x2, y2, h, ptsPerRadian)

  % if reverse is true, then reverse the curvature of the line
  % the standard curvature is up for Case I and right for Case II (see graphics)
  % if reverse is false the curvature is up for case I and right for Case II
  % if reverse is true the curvature is down for case I and left for Case II

  reverse = false;       % set reverse to false as the default

  if h < 0               % if h is negative
    h = -h;              % set h to the positive value of h
    reverse = true;      % set revese to true
  end


  % calcuate the length of the chord
  if x1 == x2                   % if x1 = x2, the chord is horizontal
    chord = abs(y2 - y1);       % and the length of the chord is abs(y2 - y1)
  elseif y1 == y2               % if y1 = y2, the chord is vertical
    chord = abs(x2 - x1);       % and the length of the chord is abs(x2 - x1)
  else                          % otherwise
    chord = 0;                  % set the value of the chord to zero
  end

  x = [0 0 0];                  % default values for x
  y = [0 0 0];                  % default values for y

  if x1 < 0 || y1 < 0 || x2 < 0 || y2 < 0          % starting and ending points must be in Quardrant I
    disp("""CurvedLine"" function arguments ""x1, y1, x2 and y2"" must all be >= 0")

    if x1 < 0                       % if x1 < 0
      fprintf('x1 = %d\n', x1);     % display "X1 = (value of x1)"
    end
    if y1 < 0                       % if y1 < 0
      fprintf('y1 = %d\n', y1);     % display "y1 = (value of y1)"
    end

    if x2 < 0                       % if x2 < 0
      fprintf('x2 = %d\n', x2);     % display "X2 = (value of x2)"
    end

    if y2 < 0                       % if y2 < 0
      fprintf('y2 = %d\n', y2);     % display "y2 = (value of y2)"
    end

  elseif ptsPerRadian <= 0                          % points per meter must be greater than zero
    disp("""CurvedLine"" function argument ""ptsPerRadian"" must be > 0")
    fprintf('ptsPerRadian = %d\n', ptsPerRadian);   % display "ptsPerRadian = (value of ptsPerRadian)"
  elseif (x1 == x2) && (y1 == y2)                   % starting and ending points can not be the same point
    disp("""CurvedLine"" function arguments ""x1 = x2 and y1 = y2""")
    disp("Starting point and ending point can not be the same point")
    fprintf('x1 = x2 = %d\n', x1);                  % display "x1 = x2 = (value of x1)"
    fprintf('y1 = y2 = %d\n', y1);                  % display "y1 = y2 = (value of y1)"
  elseif ((x1 ~= x2) && (y1 ~= y2))                 % this function only works for horizontal or vertical chords
    disp("Either (x1 must equal x2) or (y1 must equal y2)")
    disp("This function only works for horizontal or vertical chords")
    fprintf('x1 = %d\n', x1);     % display "X1 = (value of x1)"
    fprintf('y1 = %d\n', y1);     % display "y1 = (value of y1)"
    fprintf('x2 = %d\n', x2);     % display "X2 = (value of x2)"
    fprintf('y2 = %d\n', y2);     % display "y2 = (value of y2)"
  elseif h == 0                   % if h were zero then the radius of the circle would be infinite
    disp("h cannot be 0")
    fprintf('h = %d\n', h);       % display "h = 0"
  elseif (abs(h) > (chord/2))
    disp("the absolute value of h can not be greater than half the chord length")
    fprintf('the absolute value of h = %d\n', h);       % display "the absolute value of h = (value of h)"
    fprintf('  1/2 the chord length = %d\n', chord/2);  % display "1/2 the chord length = (half the value of the
chord)"
    fprintf('  %d > %d\n', h, chord/2);
  else
    increment = 1/ptsPerRadian;                  % calculate the increment
```

# Curved Line Function

```
radius = (chord^2 + (4*h^2)) / (8*h);        % calculate the radius
alpha = asin((radius - h) / radius);         % calculate the angle alpha

% CASE I - the chord is horizontal
if y1 == y2
% develop a vector of angles (theta)
% we want the last angle in the vector to be (pi - alpha)
% initially the last value in the theta vector may or may not be equal to (pi - alpha)
% adding (pi - alpha) to the theta vector will assure ourselves that the last value is (pi - alpha)
% if the last value is already (pi - alpha),
% adding one more (pi - alpha) to the vector will not hurt anything

  theta = alpha:increment:(pi - alpha);      % theta goes from alpha to (pi - alpha)
  theta = [theta, (pi - alpha)];             % see the notes above

  x = radius * cos(theta);                   % re-develop the x vector
  y = radius * sin(theta);                   % re-develop the y vector
  y = y - (radius - h);                      % move the arc down to the x-axis

  % the above code assumes we want to draw the arc from right to left
  if (x2 > x1)                  % if x2 if greater than x1, then
    x = flip(x);                % we want to draw the arc
    y = flip(y);                % from left to right
  end

  % the above code assumes we want to draw the arc up
  if reverse == true             % if reverse is true
    y = -y;                      % draw the arc down
  end

  x = x + ((x1 + x2) / 2);       % move the arc to the right
  y = y + y1;                    % move the arc up (remember y1 = y2)

else
% CASE II - the chord is vertical
% develop a vector of angles (theta)
% we want the last angle in the vector to be ((pi/2) - alpha)
% initially the last value in the theta vector may or may notbe equal to ((pi/2) - alpha)
% adding ((pi/2) - alpha) to the theta vector will assure ourselves that the last value is ((pi/2) - alpha)
```

# Curved Line Function

```
    % if the last value is already ((pi/2) - alpha),
    % adding one more ((pi/2) - alpha) to the vector will not hurt anything

      theta = ((-pi/2) + alpha):increment:((pi/2) - alpha);  % theta goes from (-pi/2) + alpha) to ((pi/2) - alpha)
      theta = [theta, ((pi/2) - alpha)];                     % see the notes above

      x = radius * cos(theta);           % re-develop the x vector
      y = radius * sin(theta);           % re-develop the y vector
      x = x - (radius - h);              % move the arc left to the x-axis

      % the above code assumes we want to draw the arc from bottom to top
      if (y1 > y2)              % if y1 if greater than y2, then
        x = flip(x);            % we want to draw the arc
        y = flip(y);            % from top to bottom
      end

      % the above code assumes we want to draw the arc to the right
      if reverse == true        % if reverse is true
        x = -x;                 % draw the arc to the left
      end

      x = x + x1;               % move the arc to the right (remember x1 = x2)
      y = y + ((y1 + y2) / 2);  % move the arc up

    end
  end

  xy = [x(:) -y(:)];            % concatenate x and y into a matrix called xy

end
```