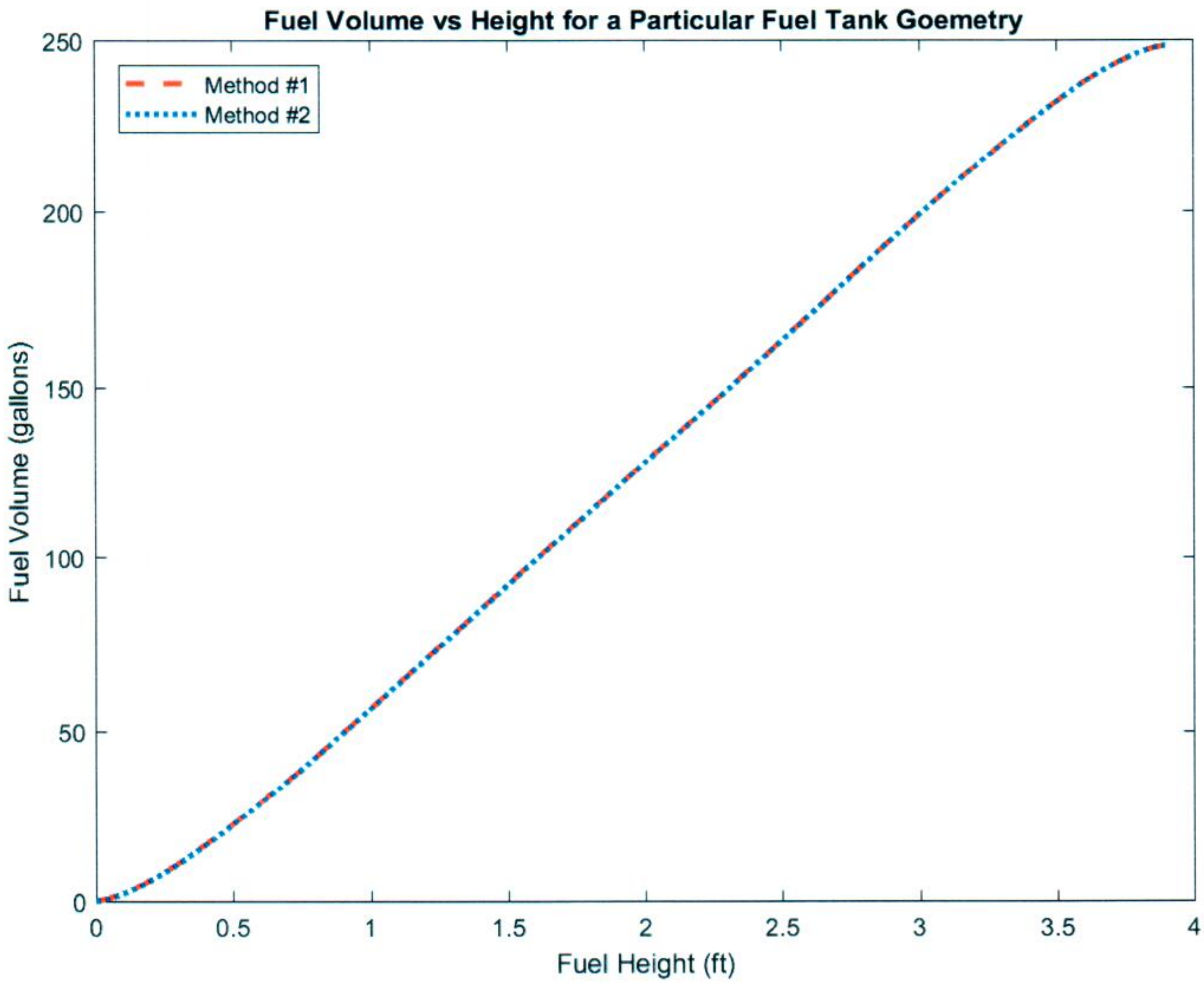
```
>> dewpoint_main


           Dewpoint Temperature vs. Relative Humidity

 Ambient Temperature: T = 20 C    T = 25 C    T = 30 C    T = 35 C
 Relative Humidity (%)          Dewpoint Temperatures (C)
         30              1.9         6.2        10.5        14.8
         40              6.0        10.4        14.9        19.4
         50              9.3        13.8        18.4        23.0
         60             12.0        16.7        21.4        26.0
         70             14.4        19.1        23.9        28.7
         80             16.4        21.3        26.2        31.0
         90             18.3        23.2        28.2        33.1
        100             20.0        25.0        30.0        35.0
>>
```

```
%
%    DEWPOINT_MAIN.M   Tabulate Dew Point Temperatures
%
%    This main program calls function DEWPOINT.M to compute the dew point temperature
%    vs. relative humidity for a single ambient temperature.  The goal here is to
%    use this routine to create a well-formatted table of results for 4 different
%    ambient temps.
%
%    File prepared by J. R. White, UMass-Lowell (last update:  Sept. 2017)
%

      clear all,  close all
%
%    compute desired dewpoint vs relative humidity for several ambient temperatures
      RH = (30:10:100)';  NRH = length(RH);  % relative humidity (%)
      T = 20:5:35;  NT = length(T);          % ambient temps (C)
      dewpt = zeros(NRH,NT);                 % storage for dewpoint temps
      for k = 1:NT
        Td = dewpoint(T(k),RH);   dewpt(:,k) = Td;
      end
%
%    print table of results
      fprintf(1,'\n\n');
      fprintf(1,'           Dewpoint Temperature vs. Relative Humidity \n');
      fprintf(1,'\n');
      fprintf(1,' Ambient Temperature: T = %2.0f C    T = %2.0f C    T = %2.0f C    T = %
2.0f C \n',T);
      fprintf(1,' Relative Humidity (%%)           Dewpoint Temperatures (C)  \n');
      for i = 1:NRH
        fprintf(1,'    %8.0f        %8.1f    %8.1f    %8.1f    %8.1f  \n',RH(i),dewpt
(i,:));
      end
%
%    end of program


%
%    DEWPOINT.M   Function file to evaluate dewpoint temperature
%            versus relative humidity for a given ambient temperature
%
%    Inputs:
%      T - ambient temperature (C) (scalar)
%      RH - relative humidity (%)  (vector)
%    Outputs:
%      Td - dewpoint temp (C)  (vector same size as RH)
%
%    File prepared by J. R. White, UMass-Lowell (last update:  Sept. 2017)
%

      function [Td] = dewpoint(T,RH)
      a = 17.27;  b = 237.7;          % constants in expression for dewpoint temp
      f = a*T/(b+T) + log(RH/100);    % intermediate function in eqn
      Td = b*f./(a-f);                % desired dewpoint temps (C)
%
%    end of function
```


Fuel Volume vs Height for a Particular Fuel Tank Goemetry
Method #1
Method #2
Fuel Volume (gallons)
Fuel Height (ft)
0
50
100
150
200
250
0.5
1
1.5
2
2.5
3
3.5
4

```
%
%   FUELVOL.M   Plots volume of fuel vs height in a particular fuel tank geometry
%
%   This program simply illustrates the use of the if ... else ... end structure and
%   the use of function files within Matlab.  Two methods are used to compute the
%   volume vs height:
%
%   Method #1 --  scalar h
%   A function file computes the fuel volume, V, given a scalar value for height, h.
%   The function is then called by this main program (within a for ... end loop) to
%   evaluate V vs h for a particular fuel tank configuration (see fuelvol1.m).
%
%   Method #2 --  vector h
%   A function file computes the fuel volume, V, given a vector of heights, h.
%   The function is then called by this main program to evaluate V vs h for a
%   particular fuel tank configuration (see fuelvol2.m).
%
%   Note that the formula for V(h) changes depending on the height...
%
%   File prepared by J. R. White, UMass-Lowell (last update: Sept. 2017)
%

      clear all,  close all
      global r H L
%
%   fix tank geometry parameters
      r = 0.95;  H = 2;  L = 5;  % radius top/bot, prism height, length of tank (ft)
      cf = 7.4805;               % conversion factor (7.4805 gal/ft^3)
      Nh = 400;  h = linspace(0,H+2*r,Nh);   % vector of fluid heights (ft)
%
%   Method #1  -- pass scalar h into function
      V1 = zeros(size(h));
      for i = 1:Nh
        V1(i) = fuelvol1(h(i));
      end
%
%   Method #2  -- pass vector h into function
      V2 = fuelvol2(h);
%
%   plot results (volume in gallons)
      plot(h,V1*cf,'r--',h,V2*cf,'b:','LineWidth',2),grid
      title('Fuel Volume vs Height for a Particular Fuel Tank Goemetry')
      xlabel('Fuel Height (ft)')
      ylabel('Fuel Volume (gallons)')
      legend('Method #1','Method #2','Location','NorthWest')
%
%   end of program
```

```
%
%   FUELVOL1.M    Function file to evaluate the volume of fuel in a
%           particular fuel tank given the fuel height (scalar form)

%   Inputs (scalar):                 Outputs (scalar):
%     h - fuel height (ft)             V - volume of fuel in tank (ft^3)

      function V = fuelvol1(h)
      global r H L
%
      if h >= 0 && h <= r                        % lower half cylinder
        V = (r^2*acos((r-h)/r) - (r-h)*sqrt(2*r*h-h^2))*L;
      elseif  h > r && h <= H+r                  % central rectangluar region
        V = pi*r^2*L/2 + 2*r*L*(h-r);
      elseif  h > H+r && h <= H+2*r              % upper half cylinder
        d = h-H;
        V = 2*r*L*H + (r^2*acos((r-d)/r) - (r-d)*sqrt(2*r*d-d^2))*L;
      else
        V = 0;
        disp('  Warning: h is outside valid range')
      end
%
%   end of function
```

```
%
%   FUELVOL2.M    Function file to evaluate the volume of fuel in a
%           particular fuel tank given the fuel height (vector form)

%   Inputs (vector):                 Outputs (vector):
%     h - fuel height (ft)             V - volume of fuel in tank (ft^3)

      function V = fuelvol2(h)
      global r H L
%
      Nh = length(h);    V = zeros(size(h));
      for i = 1:Nh
        if h(i) >= 0 && h(i) <= r                  % lower half cylinder
          V(i) = (r^2*acos((r-h(i))/r) - (r-h(i))*sqrt(2*r*h(i)-h(i)^2))*L;
        elseif  h(i) > r && h(i) <= H+r            % central rectangluar region
          V(i) = pi*r^2*L/2 + 2*r*L*(h(i)-r);
        elseif  h(i) > H+r && h(i) <= H+2*r        % upper half cylinder
          d = h(i)-H;
          V(i) = 2*r*L*H + (r^2*acos((r-d)/r) - (r-d)*sqrt(2*r*d-d^2))*L;
        else
          V(i) = 0;
          disp('  Warning: h is outside valid range')
        end
      end
%
%   end of function
```