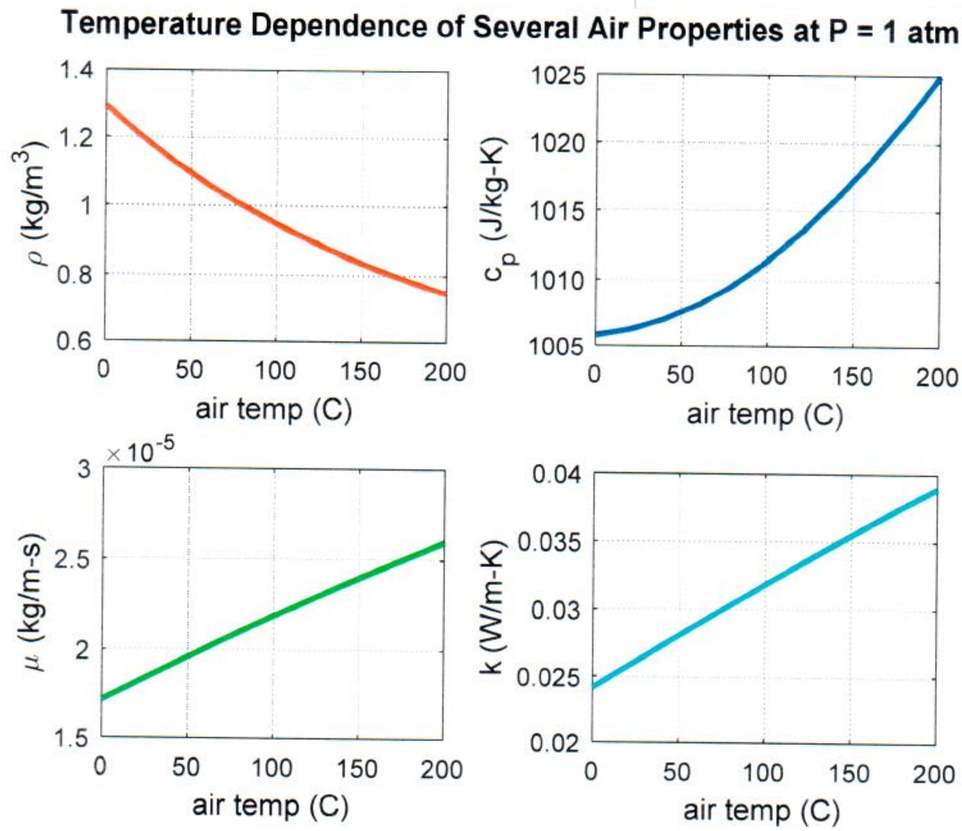



```
>> air_props_main
```

Air properties versus temperature for P = 1 atm

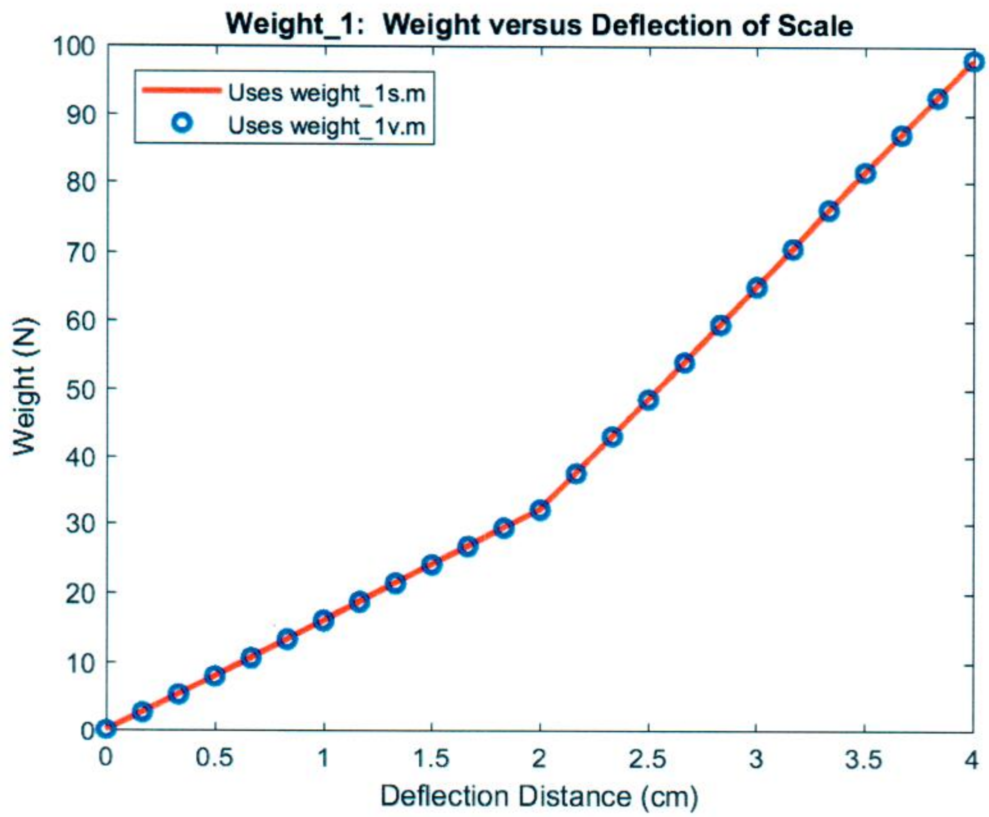
| Temperature (C) | Density (kg/m^3) | Specific Heat (J/kg-K) | Viscosity (kg/m-s) | Thermal Conductivity (W/m-K) |
|---|---|---|---|---|
| 0 | 1.292 | 1006 | 1.714e-05 | 2.413e-02 |
| 20 | 1.204 | 1006 | 1.813e-05 | 2.572e-02 |
| 40 | 1.127 | 1007 | 1.910e-05 | 2.728e-02 |
| 60 | 1.059 | 1008 | 2.004e-05 | 2.882e-02 |
| 80 | 0.999 | 1010 | 2.096e-05 | 3.033e-02 |
| 100 | 0.946 | 1011 | 2.185e-05 | 3.182e-02 |
| 120 | 0.898 | 1014 | 2.272e-05 | 3.329e-02 |
| 140 | 0.854 | 1016 | 2.357e-05 | 3.473e-02 |
| 160 | 0.815 | 1019 | 2.439e-05 | 3.614e-02 |
| 180 | 0.779 | 1022 | 2.520e-05 | 3.753e-02 |
| 200 | 0.746 | 1025 | 2.598e-05 | 3.889e-02 |

```
%
%   AIR_PROPS_MAIN.M   Evaluate and plot several properties of air
%                         vs temperature at atmospheric pressure
%
%   This file calls air_props.m to evaluate the density, specific heat, viscosity,
%   and thermal conductivity of air for a range of temperatures given a single value
%   of pressure. This main program tabulates and plots the resultant data for P =
%   atmospheric pressure.
%
%   Reference:  This problem is based on Prob. 3.47 in the text "Numerical Methods
%   with Matlab" by G. Recktenwald, Prentice Hall (2000).
%
%   File prepared by J. R. White, UMass-Lowell (last update: Sept. 2017)
%

      clear all,  close all,   nfig = 0;
%
%   set desired pressure and range of temperatures, and evaluate air properties vs T
      P = 101.3e3;           % standard atmospheric pressure (N/m^2)
      Tc = 0:20:200;         % range of temps (C)
      T = Tc+273.15;         % absolute temps (K)
      [rho,cp,mu,k] = air_props(T,P);
%
%   generate a table of values (just send this to the screen (unit #1) for now)
      fout = 1;
      fprintf(fout,'\n\n');
      fprintf(fout,'              Air properties versus temperature for P = 1 atm \n');
      fprintf(fout,'\n\n');
      fprintf(fout,' Temperature   Density   Specific Heat   Viscosity  Thermal↙
Conductivity \n');
      fprintf(fout,'      (C)       (kg/m^3)     (J/kg-K)       (kg/m-s)         (W/m-K)↙
\n');
      for i = 1:length(Tc)
        fprintf(fout,'  %6.0f        %6.3f     %6.0f        %10.3e      %10.3e \n', ...
               Tc(i),rho(i),cp(i),mu(i),k(i));
      end
      fprintf(fout,'\n');
%
%   create set of 2x2 subplots with the temperature variation of these four parameters
      nfig = nfig+1;    figure(nfig)
      subplot(2,2,1),plot(Tc,rho,'r-','LineWidth',2), grid
      title('Temperature Dependence of Several Air Properties at P = 1 atm')
      xlabel('air temp (C)'),ylabel('\rho (kg/m^3)')
      subplot(2,2,2),plot(Tc,cp,'b-','LineWidth',2), grid
      xlabel('air temp (C)'),ylabel('c_p (J/kg-K)')
      subplot(2,2,3),plot(Tc,mu,'g-','LineWidth',2), grid
      xlabel('air temp (C)'),ylabel('\mu (kg/m-s)')
      subplot(2,2,4),plot(Tc,k,'c-','LineWidth',2), grid
      xlabel('air temp (C)'),ylabel('k (W/m-K)')
%
%   end of program
```

```
%
%   AIR_PROPS.M   Evaluates the properties of air vs T & P based on
%                    ideal gas law and various curve fits
%
%   Inputs:
%      T  -  vector of absolute temperatures  (K)
%      P  -  scalar value of absolute pressure (N/m^2)
%
%   Outputs (vectors with same size as T):
%      rho - air density (kg/m^3)            cp - specific heat (J/kg-K)
%      mu  - dynamic viscosity (kg/m-s)      k  - thermal conductivity (W/m-K)
%
%   Restrictions:
%      The correlations for the material properties of air vs temperature are only
%      valid over the range 100 <= T <= 600 K.  A warning message is printed if any
%      input temperature is outside this reange.
%
%   Function prepared by J. R. White, UMass-Lowell (last update: Sept. 2017)
%


     function [rho,cp,mu,k] = air_props(T,P)
%
%   check inputs  (this part was not requested in the write-up for the students)
     if length(P) > 1
       disp('  ')
       disp('WARNING  --  this function can only handle a single input pressure')
       disp('  ')
     end
     Ilow = find(T < 100);  Ihigh = find(T > 600);   II = sum(Ilow) + sum(Ihigh);
     if II > 0
       disp('  ')
       disp('WARNING  --  input temperatures are outside the valid range 100 K <= T <=↙
600 K')
       disp('  ')
     end
%
%   set gas constant for air and coeffs for correlations
      R = 287.0;                                            % gas const for air (J/kg-K)
     a1 = -2.455322455e-7 ;  a2 =  6.701631702e-4 ;  % coeffs for cp vs T
     a3 = -2.992579643e-1 ;  a4 =  1.042503030e+3 ;
     b1 =  2.156954157e-14;  b2 = -5.332634033e-11;  % coeffs for mu vs T
     b3 =  7.477905983e-8 ;  b4 =  2.527878788e-7 ;
     c1 = -2.486402486e-12;  c2 = -2.871794872e-8 ;  % coeffs for k vs T
     c3 =  9.629059829e-5 ;  c4 =  2.060606061e-5 ;
%
%   compute properties vs T and P   (cp, mu, and k are independent of P)
     rho = P./(R*T);                        % air density (kg/m^3)
     cp = a1*T.^3 + a2*T.^2 + a3*T + a4;   % specific heat (J/kg-K)
     mu = b1*T.^3 + b2*T.^2 + b3*T + b4;   % dynamic viscosity (kg/m-s)
      k = c1*T.^3 + c2*T.^2 + c3*T + c4;   % thermal conductivity (W/m-K)
%
%   end of function
```


Weight_1: Weight versus Deflection of Scale
Uses weight_1s.m
Uses weight_1v.m
Weight (N)
Deflection Distance (cm)
0
10
20
30
40
50
60
70
80
90
100
0
0.5
1
1.5
2
2.5
3
3.5
4

```
%
%   WEIGHT_1MAIN.M   Plots Weight of an Object vs. Deflection of a Scale
%
%   This program simply illustrates the use of a function file and a conditional test
%   to make a decision within the function.  In particular, this main program calls
%   functions WEIGHT_1S.M and WEIGHT_1V.M to determine the weight of an object that
%   deflects a scale by an amount x.  The scale has an extra spring that is utilized
%   only if the deflection exceeds a distance d.  Thus, a check to determine if x > d
%   is needed to select the proper formula to use for computing the weight, W, for
%   the given x.
%
%   This main program calls WEIGHT_1S.M with a scalar x and WEIGHT_1V.M with a vector
%   of x values.  The functions simply pass back the weight, W, associated with each
%   deflection value x.  In the case of the *_1S function, a loop is needed in the
%   main program and for *_1V function, the loop is in the function file.  In both
%   cases a plot of W vs x is produced and, of course, both approaches give identical
%   results!!!
%
%   File prepared by J. R. White, UMass-Lowell (last updated: Sept. 2017)
%

      clear all,  close all,   nfig = 0;
%
%   set values of the deflection vector (m)
      x = linspace(0,0.04,25);
%
%   calc W using the function with a scalar value of x
      Ws = zeros(size(x));
      for i = 1:length(x)
        Ws(i) = weight_1s(x(i));
      end
%
%   calc W using the function with a vector of deflection values
      Wv = weight_1v(x);
%
%   now plot both results
      nfig = nfig+1;   figure(nfig)
      plot(x*100,Ws,'r-',x*100,Wv,'bo','LineWidth',2), grid
      title('Weight\_1:  Weight versus Deflection of Scale');
      xlabel('Deflection Distance (cm)'),ylabel('Weight (N)')
      legend('Uses weight\_1s.m','Uses weight\_1v.m','Location','NorthWest')
%
%   end of program
```

```
%
%   WEIGHT_1S.M   Function file to evaluate Weight vs Deflection of a Scale
%
%   Inputs:
%     x - single (scalar) deflection value (m)
%
%   Outputs:
%     W - weight of object associated with deflection x (N)
%
%   File prepared by J. R. White, UMass-Lowell (last update: Sept. 2017)
%

      function W = weight_1s(x)
%
%   data for scale of interest
      k1 = 800;       % spring constant for two of three springs (N/m)
      k2 = 1700;      % spring constant for other spring (N/m)
      d = .02;        % deflection necessary before 3rd spring is used (m)
%
%   calc weight based on input deflection
      W = 2*k1*x;     % weight of object for x <= d
      if x > d, W = W + k2*(x-d);  end  % weight of object for x > d
%
%   end of function



%
%   WEIGHT_1V.M   Function file to evaluate Weight vs Deflection of a Scale
%
%   Inputs:
%     x - vector of deflection values (m)
%
%   Outputs:
%     W - weight of object at each point associated with the deflection vector x (N)
%
%   File prepared by J. R. White, UMass-Lowell (last update: Sept. 2017)
%

      function W = weight_1v(x)
%
%   data for scale of interest
      k1 = 800;       % spring constant for two of three springs (N/m)
      k2 = 1700;      % spring constant for other spring (N/m)
      d = .02;        % deflection necessary before 3rd spring is used (m)
%
%   begin loop over each deflection value
      W = zeros(size(x));
      for i = 1:length(x)
        W(i) = 2*k1*x(i);
        if x(i) > d, W(i) = W(i) + k2*(x(i)-d);  end
      end
%
%   end of function
```