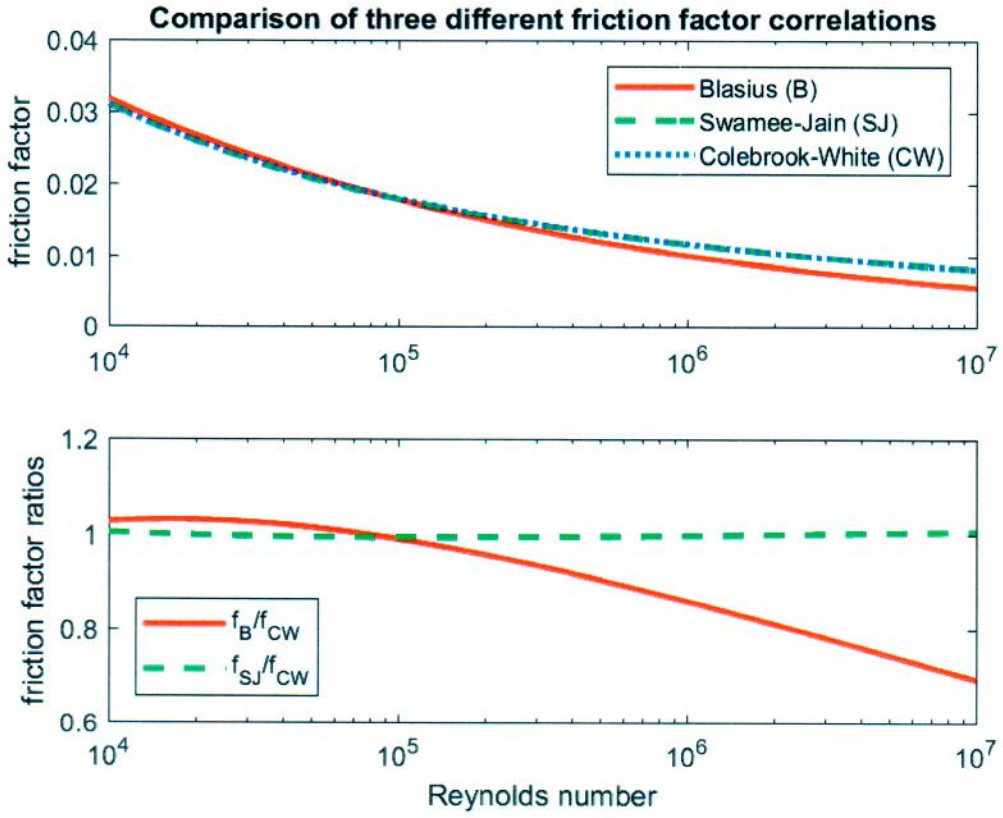
```
>> compare_methods_4

   Part a:
    Given vertical deflection =        0.200 m
    Computed weight of object =      640.743 N


   Part b:
    Given object's weight =          250.000 N

Results for Scale Deflection  ==>  Bisection     Secant      Matlab's FZERO

A zero occurs at x =               0.1099892    0.1099892    0.1099892 m
The value of f(x) is =             0.0000001   -0.0000000    0.0000000 N
# of function evaluations  =           31            8            10
```

```
%
%   COMPARE_METHODS_4.M   Compare three root finding methods for relative efficiency
%
%   This project involves comparing the efficiency of the Bisection and Secant
%   Methods relative to Matlab's built-in FZERO function for finding the real roots
%   of nonlinear equations.
%
%   This code also illustrates the difference between explicit and implicit equations.
%   An expression that relates the weight, W, and vertical deflection, x, of a
%   particular scale arrangement was derived in the problem description.  Given x,
%   W can be easily determined via an explicit evaluation of the given equation.
%   However, if W is given, we need to use a root finding routine to evaluate x,
%   since the relationship, x(W), is implicit  --  that is, we can't easily write
%   the equation in the form of x = f(W).  This implicit equation is used to
%   illustrate the relative efficiency of the various root finding methods:
%     Bisection Method              -- bisection.m
%     Secant Method                 -- secant.m
%     Matlab's built-in function    -- fzero.m
%
%   The function evaluations are done using anonymous functions since the
%   relationships are relatively simple.
%
%   File prepared by J. R. White, UMass-Lowell  (last update: Nov. 2017)
%


    clear all,  close all,   nfig = 0;
%
%   base design parameters
    a = 0.15;   b = 0.05;           % geometry parameters (m)
    K = 2800;                       % spring constants (N/m)
    Lo = sqrt(a*a + b*b);           % unstretched spring length
%
%   anonymous functions (only scalar arithmetic is used here --  so no dots needed)
    Lx = @(x) sqrt(a*a + (b + x)^2);              % explicit relation for L(x)
    Wx = @(x) (2*K/Lx(x))*(Lx(x) - Lo)*(b + x);   % explicit relation for W(x)
%
%   Part a:  Find weight of object given deflection, x  (explicit equation)
    xa = 0.2;   Wa = Wx(xa);  % given deflection and computed weight
    fprintf('\n    Part a: \n')
    fprintf('     Given vertical deflection =     %8.3f m  \n ',xa)
    fprintf('    Computed weight of object =     %8.3f N  \n ',Wa)
%
%   Part b:  Find deflection of scale given the object's weight, W  (implicit equation)
    Wb = 250;                         % object's weight (N)
    fx = @(x) Wb - Wx(x);             % implicit relation for f(x) = W - W(x) = 0
    fprintf('\n\n    Part b: \n')
    fprintf('     Given object''s weight =         %8.3f N  \n ',Wb)
%
%   call the various root finding methods and tabulate the results
    M = 50;   tol = 1e-6;   display = 0;  lb = 0;   ub = 0.2;
    [xb1,k1] = bisection(fx,lb,ub,tol,M,display);   f1 = fx(xb1);
    [xb2,k2] = secant(fx,lb,ub,tol,M,display);      f2 = fx(xb2);
    [xb3,f3,flag,output] = fzero(fx,[lb ub]);
    fprintf('\n  Results for Scale Deflection  ==>  Bisection     Secant      Matlab''s↙
FZERO \n\n')
    fprintf('  A zero occurs at x =           %12.7f %12.7f %12.7f m\n', ...
```

```
        xb1,xb2,xb3)
  fprintf('  The value of f(x) is =          %12.7f %12.7f %12.7f N\n',f1,f2,f3)
  fprintf('  # of function evaluations  =       %6i %12i  %12i\n', ...
        k1+2,k2+2,output.funcCount)

%
%  end of problem
```


Comparison of three different friction factor correlations
friction factor
0.04
0.03
0.02
0.01
0
Blasius (B)
Swamee-Jain (SJ)
Colebrook-White (CW)
10^4
10^5
10^6
10^7
friction factor ratios
1.2
1
0.8
0.6
f_B/f_{CW}
f_{SJ}/f_{CW}
Reynolds number

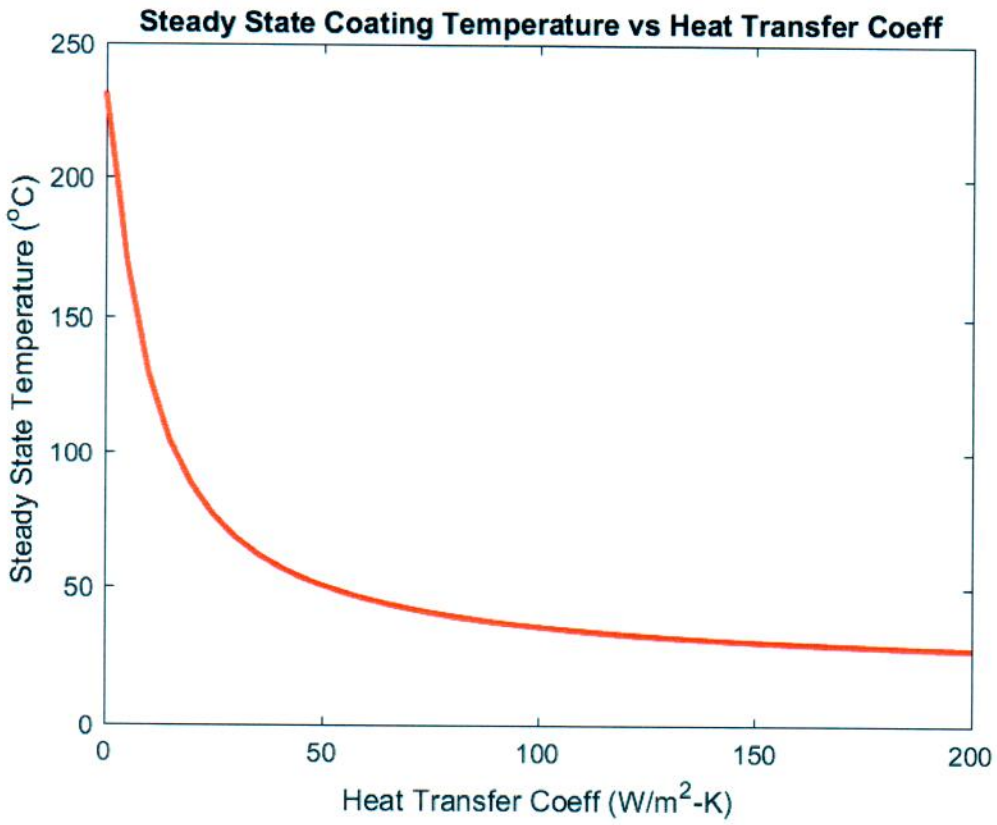
```
>> fricf_1
```

Comparison of Three Friction Factor Correlations for Smooth Pipes
(fB -> Blasius eqn, fSJ -> Swamee-Jain, & fCW -> Colebrook-White correlation)

Re	fCW	fB	fB/fCW	fSJ	fSJ/fCW
1.00e+04	0.03089	0.03164	1.024	0.03097	1.003
1.26e+04	0.02908	0.02987	1.027	0.02910	1.001
1.58e+04	0.02743	0.02820	1.028	0.02739	0.999
2.00e+04	0.02590	0.02662	1.028	0.02583	0.997
2.51e+04	0.02450	0.02513	1.026	0.02440	0.996
3.16e+04	0.02320	0.02373	1.023	0.02308	0.995
3.98e+04	0.02200	0.02240	1.018	0.02187	0.994
5.01e+04	0.02088	0.02115	1.013	0.02075	0.994
6.31e+04	0.01985	0.01996	1.006	0.01971	0.993
7.94e+04	0.01889	0.01885	0.998	0.01875	0.993
1.00e+05	0.01799	0.01779	0.989	0.01786	0.993
1.26e+05	0.01716	0.01680	0.979	0.01703	0.993
1.58e+05	0.01638	0.01586	0.968	0.01626	0.993
2.00e+05	0.01565	0.01497	0.957	0.01554	0.993
2.51e+05	0.01496	0.01413	0.945	0.01486	0.993
3.16e+05	0.01432	0.01334	0.932	0.01423	0.994
3.98e+05	0.01372	0.01260	0.918	0.01364	0.994
5.01e+05	0.01315	0.01189	0.904	0.01308	0.995
6.31e+05	0.01262	0.01123	0.889	0.01256	0.995
7.94e+05	0.01212	0.01060	0.874	0.01207	0.996
1.00e+06	0.01165	0.01001	0.859	0.01161	0.997
1.26e+06	0.01120	0.00945	0.843	0.01117	0.997
1.58e+06	0.01078	0.00892	0.827	0.01076	0.998
2.00e+06	0.01038	0.00842	0.811	0.01037	0.999
2.51e+06	0.01000	0.00795	0.795	0.01000	1.000
3.16e+06	0.00964	0.00750	0.778	0.00965	1.000
3.98e+06	0.00930	0.00708	0.762	0.00931	1.001
5.01e+06	0.00898	0.00669	0.745	0.00900	1.002
6.31e+06	0.00867	0.00631	0.728	0.00870	1.003
7.94e+06	0.00838	0.00596	0.711	0.00841	1.004
1.00e+07	0.00810	0.00563	0.694	0.00814	1.005

```
%
%   FRICF_1.M   Compare three different Friction Factor Correlations
%                     for turbulent flow in smooth pipes
%       (An example that requires finding roots of nonlinear expressions)

%   This project involves comparing the friction factor using three different
%   correlations for a range of Reynolds numbers.  One implicit expression for f is
%   given by the Colebrook-White equation:
%        1/sqrt(f) = 2.0*log10(Re*sqrt(f))-0.8
%
%   Two other commonly used explicit relationships are:
%      Blasius equation:          f = 0.3164/Re^(1/4)
%      Swamee-Jain correlation:   f = 0.25/(log10(5.74/Re^0.9))^2
%
%   where all three correlations are for turbulent flow in smooth pipes. This
%   program computes and compares these correlations for a wide range of Reynolds
%   numbers.
%
%   Matlab's FZERO root finding routine is used with the Colebrook-White equation
%   since it is an implicit relationship for f as a function of Re.  The FZERO
%   function calls a user-defined function to evaluate the particular function of
%   interest.  In this case the implicit function, F(ff), using the Colebrook-White
%   relationship is coded in an anonymous function Fff (see below).
%
%   File prepared by J. R. White, UMass-Lowell (last update: Nov. 2017)
%


    clear all;   close all;   nfig = 0;
%
%   define Re and evaluate the Blasius & Swamee_Jain equations (explicit)
    Re = logspace(4,7,61);
    fB = 0.3164./Re.^(1/4);
    fSJ = 0.25./(log10(5.74./Re.^0.9)).^2;
%
%   now evaluate the Colebrook-White equation (implicit)
%   --> use the Swamee-Jain result as an estimate of the root for FZERO.
    NRe = length(Re);  fCW = zeros(1,NRe);
    for n = 1:NRe
      RE = Re(n);
      Fff = @(ff) 1.0./sqrt(ff) + 0.8 - 2*log10(RE*sqrt(ff));
      fCW(n) = fzero(Fff,fSJ(n));
    end
%
%   create a table of results
    ratio1 = fB./fCW;   ratio2 = fSJ./fCW;
    fprintf(1,' \n     Comparison of Three Friction Factor Correlations for Smooth↙
Pipes \n');
    fprintf(1,'(fB -> Blasius eqn, fSJ -> Swamee-Jain,  &  fCW -> Colebrook-White↙
correlation)\n\n');
    fprintf(1,'         Re          fCW          fB        fB/fCW       fSJ↙
fSJ/fCW\n');
    for n = 1:2:NRe
      fprintf(1,'    %10.2e    %8.5f    %8.5f    %8.3f    %8.5f    %8.3f\n', ...
        Re(n),fCW(n),fB(n),ratio1(n),fSJ(n),ratio2(n));
    end
%
```

```
%   now let's plot the results for easy visualization
      nfig = nfig+1;    figure(nfig)
      subplot(2,1,1),semilogx(Re,fB,'r-',Re,fSJ,'g--',Re,fCW,'b:','LineWidth',2),grid
      title('Comparison of three different friction factor correlations')
      ylabel('friction factor')
      legend('Blasius (B)','Swamee-Jain (SJ)','Colebrook-White (CW)')
      subplot(2,1,2),semilogx(Re,ratio1,'r-',Re,ratio2,'g--','LineWidth',2),grid
      ylabel('friction factor ratios'),xlabel('Reynolds number')
      legend('f_B/f_{CW}','f_{SJ}/f_{CW}','Location','SouthWest')
%
%   end of program
```


Steady State Coating Temperature vs Heat Transfer Coeff
Steady State Temperature (°C)
250
200
150
100
50
0
0
50
100
150
200
Heat Transfer Coeff (W/m2-K)

```
>> coating_temp_1
```

Summary Results from the COATING_TEMP_1.M Program

Basic Design Parameters

Emissivity (dimensionless):	0.50
Stefan-Boltzmann Constant (W/m^2-K^4):	5.67e-08
Bulk Fluid Temperature (C):	20.00
Temperature of Surroundings (C):	30.00
Fraction of Incident Energy Absorbed in Coating:	0.80
Incident Irradiation from Lamps (W/m^2):	2000.00

Components of Energy Balance vs. Heat Transfer Coeff.

h (W/m^2-K)	Generation (W/m^2)	Convection (W/m^2)	Radiation (W/m^2)	Balance (W/m^2)	SS Temp (C)
0.0	1600.00	0.00	1600.00	0.000	231.55
10.0	1600.00	1094.57	505.43	0.000	129.46
20.0	1600.00	1357.75	242.25	0.000	87.89
30.0	1600.00	1453.41	146.59	0.000	68.45
40.0	1600.00	1500.51	99.49	-0.000	57.51
50.0	1600.00	1528.12	71.88	0.000	50.56
60.0	1600.00	1546.16	53.84	0.000	45.77
70.0	1600.00	1558.82	41.18	0.000	42.27
80.0	1600.00	1568.19	31.81	-0.000	39.60
90.0	1600.00	1575.40	24.60	-0.000	37.50
100.0	1600.00	1581.11	18.89	0.000	35.81
110.0	1600.00	1585.74	14.26	0.000	34.42
120.0	1600.00	1589.58	10.42	0.000	33.25
130.0	1600.00	1592.80	7.20	-0.000	32.25
140.0	1600.00	1595.56	4.44	0.000	31.40
150.0	1600.00	1597.93	2.07	0.000	30.65
160.0	1600.00	1600.00	0.00	0.000	30.00
170.0	1600.00	1601.82	-1.82	0.000	29.42
180.0	1600.00	1603.43	-3.43	0.000	28.91
190.0	1600.00	1604.87	-4.87	0.000	28.45
200.0	1600.00	1606.16	-6.16	0.000	28.03

```
%
%  COATING_TEMP_1.M      A Root Finding Example for Steady State Heat Transfer
%    Steady State Coating Temperature on a Plate Exposed to Infrared Irradiation
%
%  This file solves for the steady state surface temperature of a flat plate that is
%  irradiated by infrared lamps and that loses energy by both convective and
%  radiation heat transfer.  A range of heat transfer coefficients is considered so
%  that a plot of Tss vs h can be made.  In this way, one could effectively control
%  the curing temperature of the coating.
%
%  The goal of this problem is to illustrate how to solve nonlinear equations using
%  the built-in fzero routine in Matlab.  It also demonstates how, with a simple for
%  loop, to do a parametric study for a single variable -- in this case, we use a
%  range of values of the heat transfer coefficient, h, to show how the steady
%  state curing temperature varies with h.  For each value of h, a nonlinear energy
%  balance equation must be solved (here we use fzero to do this) to determine the
%  correct steady state temperature.
%
%  A table of results is also produced that shows the various components of the
%  energy balance eqn for each value of h.
%
%  Note: When including radiation heat transfer, the computations need to be done
%  using absolute temperatures.  Here, however, the final plot and table display
%  temperature in C  --  the units conversion is done in the plot and fprintf
%  commands.
%
%  The basic idea for this problem comes from Example 1.6 on pgs. 25-27
%  in the text, Fundamentals of Heat and Mass Transfer, by Incropera and DeWitt
%  5th Ed (2002, John Wiley & Sons).
%
%  File prepared by J. R. White, UMass-Lowell (last update: Nov. 2017)
%


  clear all;   close all;   nfig = 0;
%
%  set parameters for the problem
  emiss = 0.5;           % emissivity (dimensionless)
  sig = 5.67e-8;         % Stefan-Boltzmann constant (W/m^2-K^4)
  Tinf = 20+273.15;      % bulk fluid temperature (K)
  Tsur = 30+273.15;      % temperature of surroundings for radiation HT (K)
  alf = 0.8;             % fraction of incident energy absorbed in coating
  G = 2000;              % incident radiation from lamps (W/m^2)
%
  Nh = 41;                          % number of h values to study
  htc = linspace(0,200,Nh);         % set range for heat transfer coeff (W/m^2-K)
  Tss = zeros(size(htc));           % allocate space for storage of SS temperatures
  h = htc(1);                       % h for first value of heat transfer coeff
  Tg = Tsur+50;                     % initial guess of Tss for first value of htc
  fT = @(T) alf*G - h*(T - Tinf) - emiss*sig*(T^4 - Tsur^4);
  Tss(1) = fzero(fT,Tg);            % find actual Tss for first value of htc
%
%  find SS temperautre for each heat transfer coeff (use previous T as guess)
  for n = 2:Nh
    h = htc(n);
    fT = @(T) alf*G - h*(T - Tinf) - emiss*sig*(T^4 - Tsur^4);
    Tss(n) = fzero(fT,Tss(n-1));
```

```
    end
%
%   plot Tss vs htc
    nfig = nfig+1;   figure(nfig)
    plot(htc,Tss-273.15,'r-','LineWidth',2), grid on
    rr = axis;   rr(3) = 0;   axis(rr);
    title('Steady State Coating Temperature vs Heat Transfer Coeff')
    xlabel('Heat Transfer Coeff (W/m^2-K)'),ylabel('Steady State Temperature (^oC)')
%
%   evaluate components of balance equation (use 'dot arithmetic' where appropriate)
    gen = alf*G;                          % energy rate in by irradiation
    conv = htc.*(Tss - Tinf);             % loss rate by convection
    rad = emiss*sig*(Tss.^4 - Tsur^4);    % loss rate via radiation
    bal = gen - conv - rad;               % SS energy balance eqn
%
%   write summary table of results
    fprintf(1,'\n\n');
    fprintf(1,'          Summary Results from the COATING_TEMP_1.M Program \n');
    fprintf(1,'\n');
    fprintf(1,'    Basic Design Parameters      \n');
    fprintf(1,'      Emissivity (dimensionless):                    %8.2f    \n',↙
emiss);
    fprintf(1,'      Stefan-Boltzmann Constant (W/m^2-K^4):         %9.2e    \n',↙
sig);
    fprintf(1,'      Bulk Fluid Temperature (C):                    %8.2f    \n',↙
Tinf-273.15);
    fprintf(1,'      Temperature of Surroundings (C):               %8.2f    \n',↙
Tsur-273.15);
    fprintf(1,'      Fraction of Incident Energy Absorbed in Coating: %8.2f    \n',↙
alf);
    fprintf(1,'      Incident Irradiation from Lamps (W/m^2):       %8.2f    \n',G);
    fprintf(1,'\n');
    fprintf(1,'    Components of Energy Balance vs. Heat Transfer Coeff.  \n');
    fprintf(1,'           h      Generation    Convection   Radiation     Balance↙
SS Temp  \n');
    fprintf(1,'       (W/m^2-K)     (W/m^2)       (W/m^2)     (W/m^2)       (W/m^2)↙
(C)    \n');
    for n = 1:2:Nh
      fprintf(1,'          %5.1f     %8.2f      %8.2f    %8.2f    %8.3f     %8.2f↙
\n', ...
              htc(n),gen,conv(n),rad(n),bal(n),Tss(n)-273.15);
    end
%
%   end of problem
```