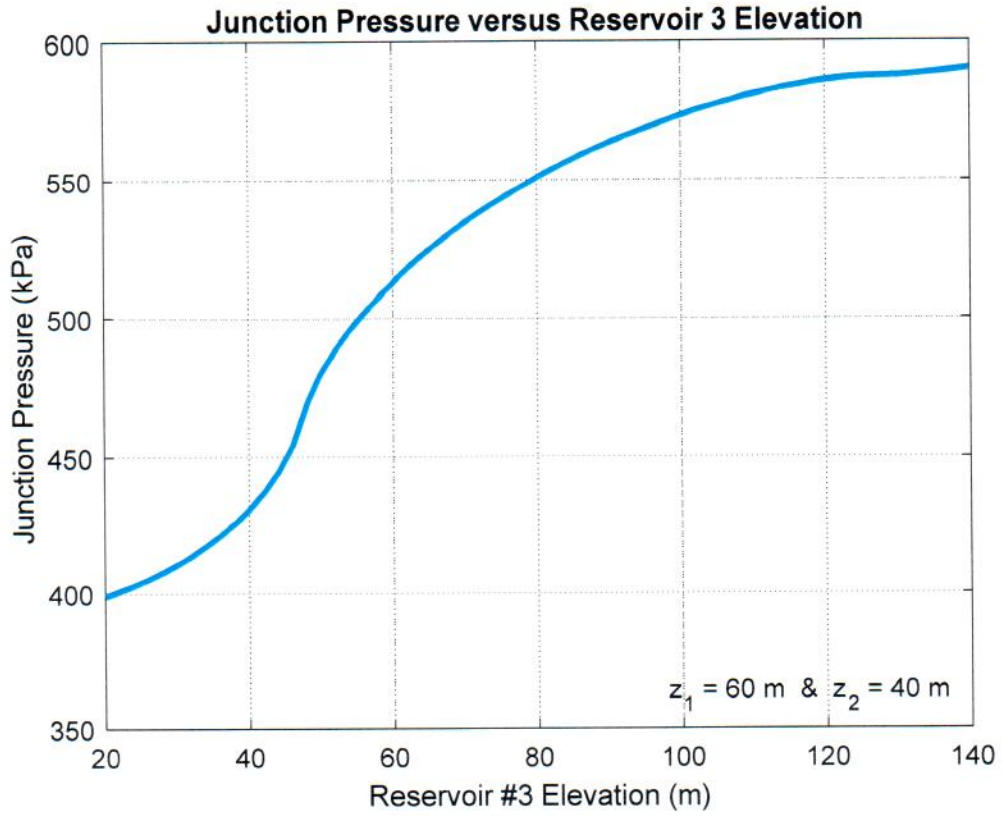
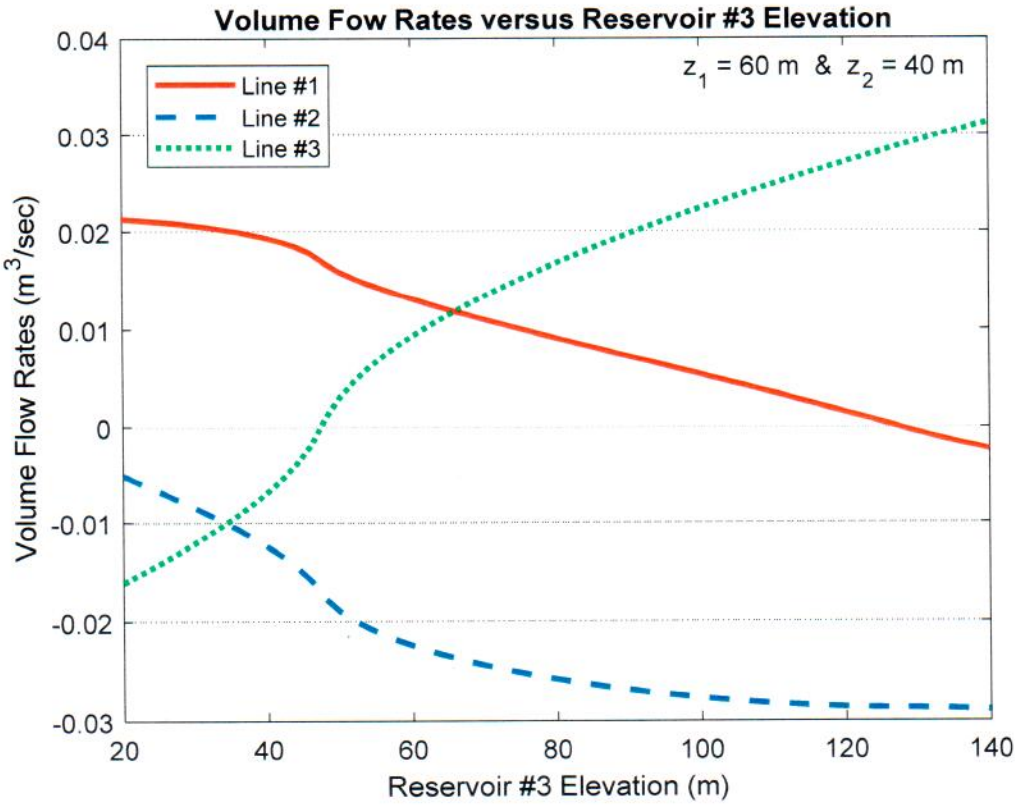
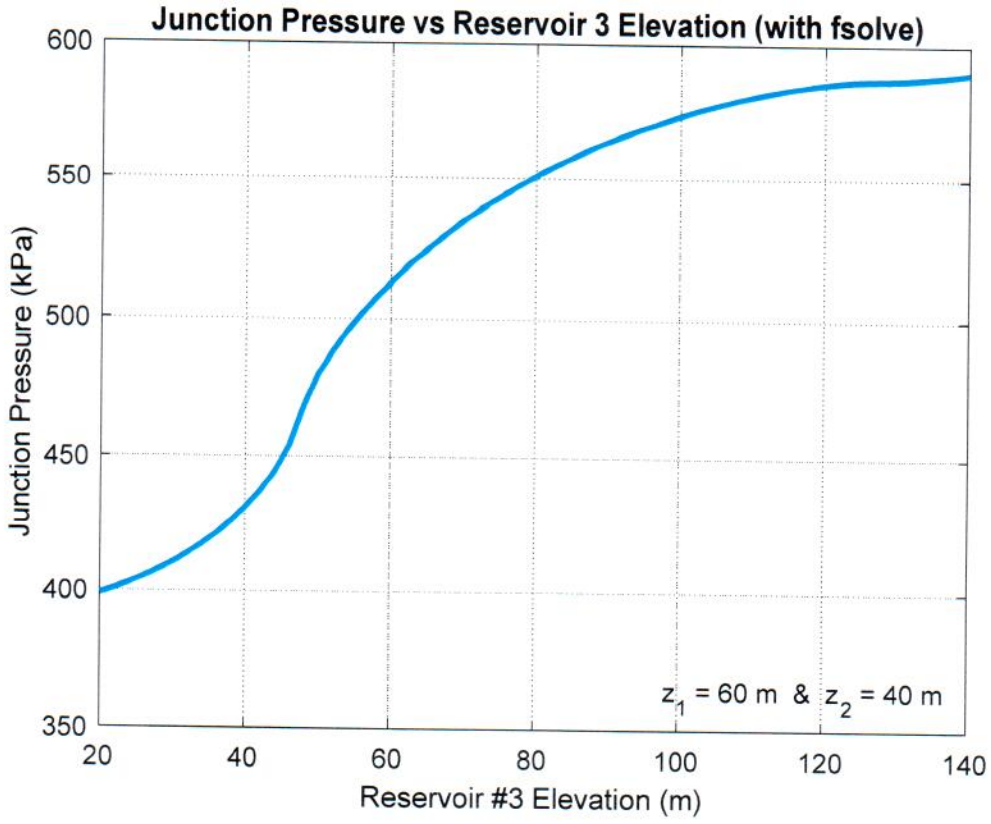
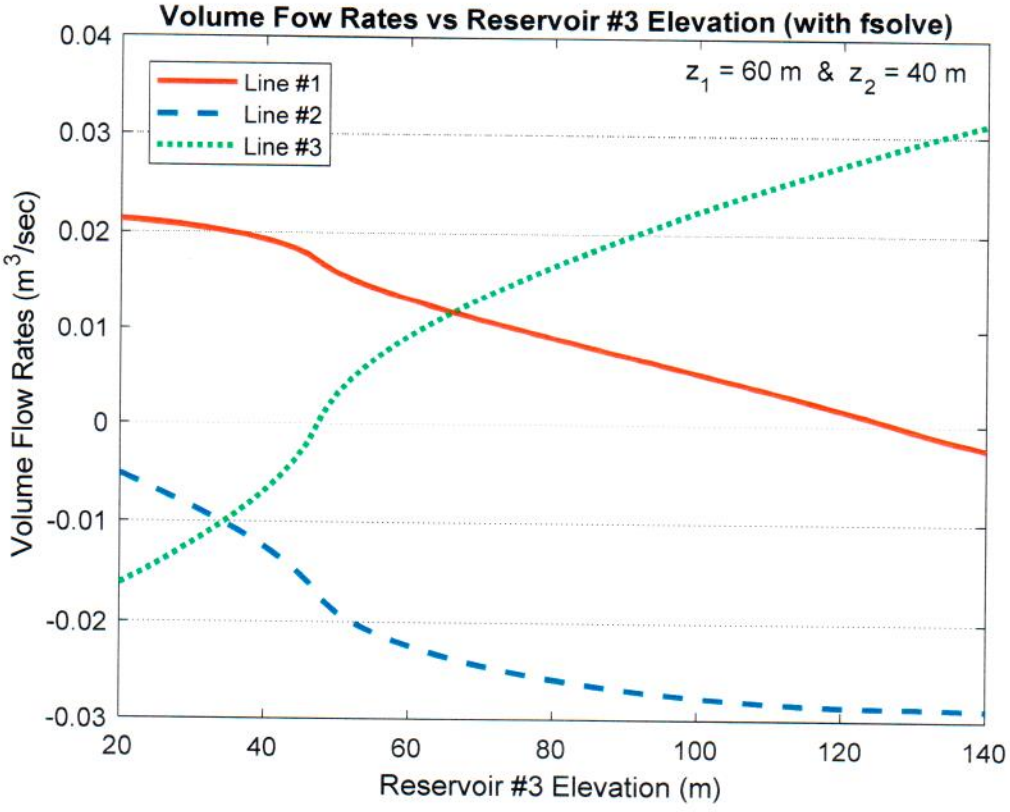
```
>> three_reservoirs_2

 Intermediate edit for Linearized Iteration Method -- Case A
  it =   1     max error = 6.96e-01
       xnew          xold
     4.425e+02       3.916e+02
     3.288e-02       1.000e-02
    -1.690e-02       1.000e-02
    -1.598e-02      -2.000e-02
  it =   2     max error = 4.68e-02
       xnew          xold
     3.984e+02       4.170e+02
     2.112e-02       2.144e-02
    -6.636e-03      -3.450e-03
    -1.449e-02      -1.799e-02
  it =   3     max error = 2.19e-02
       xnew          xold
     3.990e+02       4.077e+02
     2.121e-02       2.128e-02
    -5.226e-03      -5.043e-03
    -1.598e-02      -1.624e-02
  it =   4     max error = 1.10e-02
       xnew          xold
     3.990e+02       4.033e+02
     2.124e-02       2.124e-02
    -5.143e-03      -5.135e-03
    -1.610e-02      -1.611e-02
  it =   5     max error = 5.49e-03
       xnew          xold
     3.990e+02       4.012e+02
     2.124e-02       2.124e-02
    -5.139e-03      -5.139e-03
    -1.610e-02      -1.610e-02
  it =   6     max error = 2.74e-03
       xnew          xold
     3.990e+02       4.001e+02
     2.124e-02       2.124e-02
    -5.139e-03      -5.139e-03
    -1.610e-02      -1.610e-02
  it =   7     max error = 1.37e-03
       xnew          xold
     3.990e+02       3.995e+02
     2.124e-02       2.124e-02
    -5.139e-03      -5.139e-03
    -1.610e-02      -1.610e-02
  it =   8     max error = 6.86e-04
       xnew          xold
     3.990e+02       3.992e+02
     2.124e-02       2.124e-02
    -5.139e-03      -5.139e-03
    -1.610e-02      -1.610e-02
  it =   9     max error = 3.43e-04
       xnew          xold
     3.990e+02       3.991e+02
     2.124e-02       2.124e-02
    -5.139e-03      -5.139e-03
    -1.610e-02      -1.610e-02
  it =  10     max error = 1.72e-04
       xnew          xold
     3.990e+02       3.990e+02
     2.124e-02       2.124e-02
    -5.139e-03      -5.139e-03
    -1.610e-02      -1.610e-02
  it =  11     max error = 8.58e-05
       xnew          xold
     3.990e+02       3.990e+02
     2.124e-02       2.124e-02
```

```
   -5.139e-03      -5.139e-03
   -1.610e-02      -1.610e-02

fsolve solution edit --  Case A
    initial guess          solution vector     function vector at solution
xo(1) =    3.916e+02   x(1) =    3.990e+02    F(1) =   3.4694e-18
xo(2) =    1.000e-02   x(2) =    2.124e-02    F(2) =   2.1316e-14
xo(3) =    1.000e-02   x(3) =   -5.139e-03    F(3) =  -7.1054e-15
xo(4) =   -2.000e-02   x(4) =   -1.610e-02    F(4) =   1.7764e-14


  Results for Three Reservoirs Problem Part A (ver 2)

    Geometry (3" Sch 40 steel pipe):
      pipe dia (m):              7.790e-02
      flow area (m^2):           4.766e-03
      relative roughness:        5.905e-04

    Fluid Properties:
      sp. wt. (N/m^3):           9.790e+00
      density (kg/m^3):          9.980e+02
      viscosity (N-s/m^2):       1.020e-03

       *** Part A Results from Linearized Iteration method ***
    Pressure at junction (kPa):      3.990e+02
    Pressure head at junction (m):   4.075e+01

    Calculated Parameters for each Pipe Segment (Case A):
Pipe# Height Length Flow Rate Ave Vel    Re#  FricFactor Kcoeff Head Loss  Balance
        (m)    (m)   (m^3/s)   (m/s)                                (m)
   1    60.0   75.0   0.02124    4.46  339708    0.0187    42655   19.25    6.04e-13
   2    40.0   50.0  -0.00514   -1.08   82186    0.0213   -28484   -0.75   -2.91e-13
   3    20.0  150.0  -0.01610   -3.38  257522    0.0190   -80028  -20.75    2.05e-12


       *** Part A Results when using fsolve method ***
    Pressure at junction (kPa):      3.990e+02
    Pressure head at junction (m):   4.075e+01

    Calculated Parameters for each Pipe Segment (Case A):
Pipe# Height Length Flow Rate Ave Vel    Re#  FricFactor Kcoeff Head Loss  Balance
        (m)    (m)   (m^3/s)   (m/s)                                (m)
   1    60.0   75.0   0.02124    4.46  339708    0.0187    42655   19.25    2.84e-14
   2    40.0   50.0  -0.00514   -1.08   82186    0.0213   -28484   -0.75   -7.11e-15
   3    20.0  150.0  -0.01610   -3.38  257522    0.0190   -80028  -20.75    1.78e-14
>>
```


Junction Pressure versus Reservoir 3 Elevation
Junction Pressure (kPa)
600
550
500
450
400
350
20
40
60
80
100
120
140
Reservoir #3 Elevation (m)
z1 = 60 m & z2 = 40 m


Volume Fow Rates versus Reservoir #3 Elevation
Volume Flow Rates (m3/sec)
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
Line #1
Line #2
Line #3
z1 = 60 m & z2 = 40 m
20
40
60
80
100
120
140
Reservoir #3 Elevation (m)


Junction Pressure vs Reservoir 3 Elevation (with fsolve)
Junction Pressure (kPa)
Reservoir #3 Elevation (m)
600
550
500
450
400
350
20
40
60
80
100
120
140
z_1 = 60 m & z_2 = 40 m


Volume Fow Rates vs Reservoir #3 Elevation (with fsolve)
Volume Flow Rates (m^3/sec)
Reservoir #3 Elevation (m)
Line #1
Line #2
Line #3
z_1 = 60 m & z_2 = 40 m
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
20
40
60
80
100
120
140

```
%
% THREE_RESERVOIRS_2.M  Solve for the Flow Rates for the Classical
%                       Three-Reservoir Problem
%
% This file computes the flow rates and junction pressure for the classical
% three-reservoirs problem.  This problem is a great example of a nonlinear
% system of equations, and it is used to illustrate the "linearized iteration"
% approach for solving mildly nonlinear problems.  This file also compares this
% simple method to the more robust "fsolve" routine in Matlab for solving general
% unconstrained nonlinear problems.
%
% Linearized Iteration Method:
% In this method, the coefficient matrix is a function of the solution vector.
% However, with a given guess for x, A(x) can be determined, and the solution of
% a linear system, A(x) * x = b, can be obtained.  This process is continued
% until convergence (or we hit the max number of iterations).
%
% Note that one often simply lets the next estimate of the solution be the
% current best guess, xold = xnew.  However, for this problem, the solution
% tended to oscillate around the real solution.  When this happens, we often
% use the next guess as the average of the previous two solution estimates,
% xold = (xnew + xold)/2.  This is the approach taken here.
%
% Use of fsolve:
% "fsolve" is one of Matlab's main routines for solving unconstrained nonlinear
% problems.  It is used in a fashion simliar to "fzero", but there is a vector of
% x values to determine and each call to the function file returns a vector of
% F values (instead of one value as in fzero).  The goal here is to find vector x
% such that vector F(x) = 0.
%
% Part A
% The given problem has two parts.  The first part (Part A) looks at the detailed
% solution to a single case with fixed elevations for the three reservoirs.  This
% case is studied in detail to make sure everything is working properly and that
% the solution makes sense physically.  BOTH the linearized iteration method and
% fsolve are used to make sure we get the same solution...
%
% Part B
% Then, once things make sense, we solve the problem with a varying height for
% Reservoir #3 to see how the junction pressure and flow rates vary with this
% parameter.  The problem specification only requests one method for this part of
% the problem --  I originally selected to use the Linearized Iteration Method
% (with no intermediate edit).  However, since some students were having difficulty
% with the use of fsolve for the parametric study, I decided to also use this
% method just to see what was happening here.  Thus, I now have both methods
% implemented for the parametric analysis...
%
% Final Note:  Everything is contained in one m file for ease in keeping all the
% subfunctions together.  This requires that the "main" program be a function!!!
%
% File prepared by J. R. White, UMass-Lowell (last update:  Nov. 2017)
%

function three_reservoirs_2    % this is the "main" function (i.e. main program)
close all;   nfig = 0;
%
```

```
%   set the problem parameters
      g = 9.8;                        % gravitational constant (m/s^2)
      D =  0.0779;                    % pipe diameter (all 3" Sch 40) (m) [Mott App F]
      e = 4.6e-5;                     % surface roughness (m)  [Mott Table 9.1]
      eoD = e/D;                      % relative roughness
      A =  pi*D*D/4;                  % flow area (m^2)
      L = [75 50 150]';               % pipe lengths (m)
      z = [60 40 20]';                % reservoir elevations relative to junction (m)
%
%   set fluid properties  (water at 20 C)   [Mott App A]
      gam = 9.79;                     % specific weight (kN/m^3)
      rho = 998;                      % fluid density (kg/m^3)
      mu = 1.02e-3;                   % fluid dynamic viscosity (N-s/m^2)
%
%   set reasonable initial guess for the solution vector
%    -- actually I really have no idea here
%    -- let's say that Pj is about gam*(sum(z)/3)  --  where ave height is sum(z)/3
%    -- just set Qi to [0.01 0.01 -0.02] m^3/s (for starters anyway)
%
      xguess = [gam*sum(z)/3 0.01 0.01 -0.02]';  N = length(xguess);
%
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   %   Part A -- Linearized Iteration Method    %
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
      iedit = 1;
      if iedit == 1
        fprintf(1,'\n Intermediate edit for Linearized Iteration Method -- Case A\n');
      end
      itmax = 20;  it = 0;  tol = 1e-4;  emax = 1;   xold = xguess;
      while emax > tol  &&  it <= itmax
        [AA,bb] = abmatrix(xold,g,D,A,L,eoD,gam,rho,mu,z); % determine coeff matrices
        xnew = AA\bb;                                       % find solution vector
        emax = max(abs(xnew-xold)./xnew);                   % calculate max error
        it = it+1;                                          % increment counter
        if iedit == 1                                       % intermediate edit
          fprintf(1,'  it = %3d    max error = %8.2e \n',it,emax);
          fprintf(1,'        xnew          xold       \n');
          for j = 1:N
            fprintf(1,'  %12.3e   %12.3e    \n',xnew(j),xold(j));
          end
        end
        xold = (xnew+xold)/2;                               % guess for next iteration
      end
      if it > itmax   % print warning and max relative error if we hit max iterations
        fprintf(1,'\n *** WARNING ***   Not Converged -- hit max number of
iterations\n');
        fprintf(1,'\n    Max number of iterations  = %3d\n',itmax);
        fprintf(1,'    Max relative error when stopped =  %8.2e\n',emax);
      end
      x_liniter = xnew;   % solution vector from Linearized Iteration Method
%
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   %   Part A -- Use of fsolve    %
%   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
```

```
      xo = xguess;
      Fx = @(x) Feqns(x,g,D,A,L,eoD,gam,rho,mu,z);
      options = optimoptions('fsolve','Display','none');
      [x,F] = fsolve(Fx,xo,options);
      fprintf(1,'\n fsolve solution edit --  Case A \n');
      fprintf(1,'     initial guess         solution vector    function vector at↙
 solution \n');
      for j = 1:length(xo)
        fprintf(1,'  xo(%1i) = %12.3e   x(%1i) = %12.3e    F(%1i) = %12.4e\n', ...
                j,xo(j),j,x(j),j,F(j));
      end
      x_fsolve = x;    % solution vector from Matlab's fsolve routine
 %
 %   finally, edit some detailed results for Part A  (for both methods)
      [P,Q] = editA(x_liniter,x_fsolve,g,D,A,L,eoD,gam,rho,mu,z);
      Pref = P;   Qref = Q;     % save reference results for later use
 %
 %   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %   %  Part B -- vary elevation of reservoir #3 (linearized iteration method)  %
 %   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %
      z3 = 20:2:140;   Nz3 = length(z3);      % define vector of z3 elevations
      PB = zeros(Nz3,1);  QB = zeros(Nz3,3);  % initialize matrices for results
      PB(1) = Pref;  QB(1,:) = Qref;          % store results from above
 %
 %   loop over number z3 values
      for n = 2:Nz3
        z = [60 40 z3(n)]';               % height relative to junction (m)
        xold = [PB(n-1) QB(n-1,:)]';      % use previous values as guess for current z3
        N = length(xold);
        itmax = 20;   it = 0;   tol = 1e-4;  emax = 1;
        while emax > tol  &&  it <= itmax
          [AA,bb] = abmatrix(xold,g,D,A,L,eoD,gam,rho,mu,z);% determine coeff matrices
          xnew = AA\bb;                                     % find solution vector
          emax = max(abs(xnew-xold)./xnew);                 % calculate max error
          it = it+1;                                        % increment counter
          xold = (xnew+xold)/2;                             % guess for next iteration
        end
        if it > itmax  % print warning and max relative error if we hit max iterations
          fprintf(1,'\n *** WARNING ***   Not Converged -- hit max number of↙
iterations\n');
          fprintf(1,'\n    Max number of iterations  = %3d\n',itmax);
          fprintf(1,'    Max relative error when stopped =  %8.2e\n',emax);
        end
        PB(n) = xnew(1);   QB(n,:) = xnew(2:N)';  % save results for nth value of z3
      end
 %
 %   plot results
      nfig = nfig+1;   figure(nfig)
      plot(z3,PB,'LineWidth',2),grid
      title('Junction Pressure versus Reservoir 3 Elevation')
      xlabel('Reservoir #3 Elevation (m)'),ylabel('Junction Pressure (kPa)')
      rr = axis;   % gets axis limits
      xloc = rr(1) + 0.65*(rr(2)-rr(1));  yloc = rr(3) + 0.05*(rr(4)-rr(3));
      text(xloc,yloc,['z_1 = ',num2str(z(1)),' m  &  z_2 = ',num2str(z(2)),' m'])
 %
```

```
  nfig = nfig+1;   figure(nfig)
  plot(z3,QB(:,1),'r-',z3,QB(:,2),'b--',z3,QB(:,3),'g:','LineWidth',2),grid
  title('Volume Fow Rates versus Reservoir #3 Elevation')
  xlabel('Reservoir #3 Elevation (m)'),ylabel('Volume Flow Rates (m^3/sec)')
  legend('Line #1','Line #2','Line #3','Location','NorthWest')
  rr = axis;   % gets axis limits
  xloc = rr(1) + 0.65*(rr(2)-rr(1));  yloc = rr(3) + 0.95*(rr(4)-rr(3));
  text(xloc,yloc,['z_1 = ',num2str(z(1)),' m  &  z_2 = ',num2str(z(2)),' m'])
%
%
%  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  %  Part B -- vary elevation of reservoir #3 (use of fsolve)  %
%  %      (just repeats coding from above where possible)       %
%  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
  z3 = 20:2:140;   Nz3 = length(z3);      % define vector of z3 elevations
  PB = zeros(Nz3,1);  QB = zeros(Nz3,3);  % initialize matrices for results
  PB(1) = Pref;  QB(1,:) = Qref;          % store results from above
%
%  loop over number z3 values
  options = optimoptions('fsolve','Display','none');
  for n = 2:Nz3
    z = [60 40 z3(n)]';            % height relative to junction (m)
    xold = [PB(n-1) QB(n-1,:)]';   % use previous values as guess for current z3
    Fx = @(x) Feqns(x,g,D,A,L,eoD,gam,rho,mu,z);
    [xnew,F] = fsolve(Fx,xold,options);
    PB(n) = xnew(1);   QB(n,:) = xnew(2:N)';  % save results for nth value of z3
  end
%
%  plot results
  nfig = nfig+1;   figure(nfig)
  plot(z3,PB,'LineWidth',2),grid
  title('Junction Pressure vs Reservoir 3 Elevation (with fsolve)')
  xlabel('Reservoir #3 Elevation (m)'),ylabel('Junction Pressure (kPa)')
  rr = axis;   % gets axis limits
  xloc = rr(1) + 0.65*(rr(2)-rr(1));  yloc = rr(3) + 0.05*(rr(4)-rr(3));
  text(xloc,yloc,['z_1 = ',num2str(z(1)),' m  &  z_2 = ',num2str(z(2)),' m'])
%
  nfig = nfig+1;   figure(nfig)
  plot(z3,QB(:,1),'r-',z3,QB(:,2),'b--',z3,QB(:,3),'g:','LineWidth',2),grid
  title('Volume Fow Rates vs Reservoir #3 Elevation (with fsolve)')
  xlabel('Reservoir #3 Elevation (m)'),ylabel('Volume Flow Rates (m^3/sec)')
  legend('Line #1','Line #2','Line #3','Location','NorthWest')
  rr = axis;   % gets axis limits
  xloc = rr(1) + 0.65*(rr(2)-rr(1));  yloc = rr(3) + 0.95*(rr(4)-rr(3));
  text(xloc,yloc,['z_1 = ',num2str(z(1)),' m  &  z_2 = ',num2str(z(2)),' m'])

  end   %  end of main function file



%
%  abmatrix  Function file to compute the coefficient matrices for three_reservoir_2
%
  function [a,b] = abmatrix(x,g,D,A,L,eoD,gam,rho,mu,z)
%
```

```
  N = length(x);  a = zeros(N,N);    % initialize variables
  Q = x(2:N);                        % extract the flow rates (for convenience)
%
% compute resistance coefficients (note: there are N-1 flow rates)
  K = zeros(N-1,1);
  for i = 1:N-1
    Re = (rho*D/(mu*A))*abs(Q(i));
    if Re <= 2000
      f = 64/Re;
    else
      f = 0.25/(log10(eoD/3.7 + 5.74/Re^0.9))^2;
    end
    K(i) = (1 + sign(Q(i))*f*L(i)/D)/(2*g*A*A);
  end
%
% compute desired coefficient matrices
  a(1,2:N) = 1;                                    % set first row
  a(2:N,1) = 1/gam;                                % set remainder of 1st column
  for i = 2:N,  a(i,i) = K(i-1)*Q(i-1);  end       % set remaining nonzero terms
  b = [0; z];                                      % set b vector
%
  end  % end of abmatrix function



%
% Feqns   Nonlinear function for use in fsolve  (called from three_reservoirs_2)
%
  function F = Feqns(x,g,D,A,L,eoD,gam,rho,mu,z)
  [A,b] = abmatrix(x,g,D,A,L,eoD,gam,rho,mu,z); % determine coeff matrices
  F = A*x-b;
%
  end   % end of Feqn function



%
% editA   edit some detailed results from Part A (pass back P and Q from fsolve)
%
  function [P,Q] = editA(x_liniter,x_fsolve,g,D,A,L,eoD,gam,rho,mu,z)
%
% print general problem specifications
  fid = 1;
  fprintf(fid,' \n\n');
  fprintf(fid,'    Results for Three Reservoirs Problem Part A (ver 2) \n');
  fprintf(fid,' \n');
  fprintf(fid,'      Geometry (3" Sch 40 steel pipe): \n');
  fprintf(fid,'        pipe dia (m):            %10.3e \n',D);
  fprintf(fid,'        flow area (m^2):         %10.3e \n',A);
  fprintf(fid,'        relative roughness:      %10.3e \n',eoD);
  fprintf(fid,' \n');
  fprintf(fid,'      Fluid Properties: \n');
  fprintf(fid,'        sp. wt. (N/m^3):         %10.3e \n',gam);
  fprintf(fid,'        density (kg/m^3):        %10.3e \n',rho);
  fprintf(fid,'        viscosity (N-s/m^2):     %10.3e \n',mu);
  fprintf(fid,' \n');
```

```
%
%   now compute and edit result from Linearized Iteration Method
      [P,Q,v,Re,f,K,hL,bal] = results(x_liniter,g,D,A,L,eoD,gam,rho,mu,z);
      fprintf(fid,'         *** Part A Results from Linearized Iteration method *** \n');
      fprintf(fid,'      Pressure at junction (kPa):    %10.3e \n',P);
      fprintf(fid,'      Pressure head at junction (m): %10.3e \n',P/gam);
      fprintf(fid,' \n');
      fprintf(fid,'      Calculated Parameters for each Pipe Segment (Case A): \n');
      fprintf(fid,'   Pipe# Height Length Flow Rate Ave Vel   Re#  FricFactor Kcoeff Head↙
Loss  Balance \n');
      fprintf(fid,'           (m)      (m)    (m^3/s)    (m/s)↙
(m) \n');
      for n = 1:length(x_liniter)-1
        fprintf(fid,'%6i %7.1f %6.1f  %8.5f %7.2f %7.0f %8.4f %8.0f %7.2f  %10.2e \n',↙
...
                    n,z(n),L(n),Q(n),v(n),Re(n),f(n),K(n),hL(n), bal(n));
      end
      fprintf(fid,' \n\n');
%
%   now compute and edit result from Matlab's fsolve routine
      [P,Q,v,Re,f,K,hL,bal] = results(x_fsolve,g,D,A,L,eoD,gam,rho,mu,z);
      fprintf(fid,'         *** Part A Results when using fsolve method *** \n');
      fprintf(fid,'      Pressure at junction (kPa):    %10.3e \n',P);
      fprintf(fid,'      Pressure head at junction (m): %10.3e \n',P/gam);
      fprintf(fid,' \n');
      fprintf(fid,'      Calculated Parameters for each Pipe Segment (Case A): \n');
      fprintf(fid,'   Pipe# Height Length Flow Rate Ave Vel   Re#  FricFactor Kcoeff Head↙
Loss  Balance \n');
      fprintf(fid,'           (m)      (m)    (m^3/s)    (m/s)↙
(m) \n');
      for n = 1:length(x_fsolve)-1
        fprintf(fid,'%6i %7.1f %6.1f  %8.5f %7.2f %7.0f %8.4f %8.0f %7.2f  %10.2e \n',↙
...
                    n,z(n),L(n),Q(n),v(n),Re(n),f(n),K(n),hL(n), bal(n));
      end
%
      end   % end of editA function


%
%   results   calculate some intermediate results
%
      function [P,Q,v,Re,f,K,hL,bal] = results(x,g,D,A,L,eoD,gam,rho,mu,z)
      N = length(x);                                  % number of unknowns
      P = x(1);  Q = x(2:N);                          % extract pressure & flow rates
      v = Q/A;                                        % average flow velocity
      vm = abs(v);                                    % magnitude of average velocity
      Re = rho*vm*D/mu;                               % Reynolds number
      f = 0.25./(log10(eoD/3.7 + 5.74./Re.^0.9)).^2;% Darcy friction factor
      K = (1+sign(Q).*f.*(L/D))/(2*g*A*A);            % resistance coeff
      hL = K.*Q.^2;                                   % head loss (includes vel head)
      bal = P/gam + hL - z;                           % energy balance (should be zero)

      end   % end of results function
```