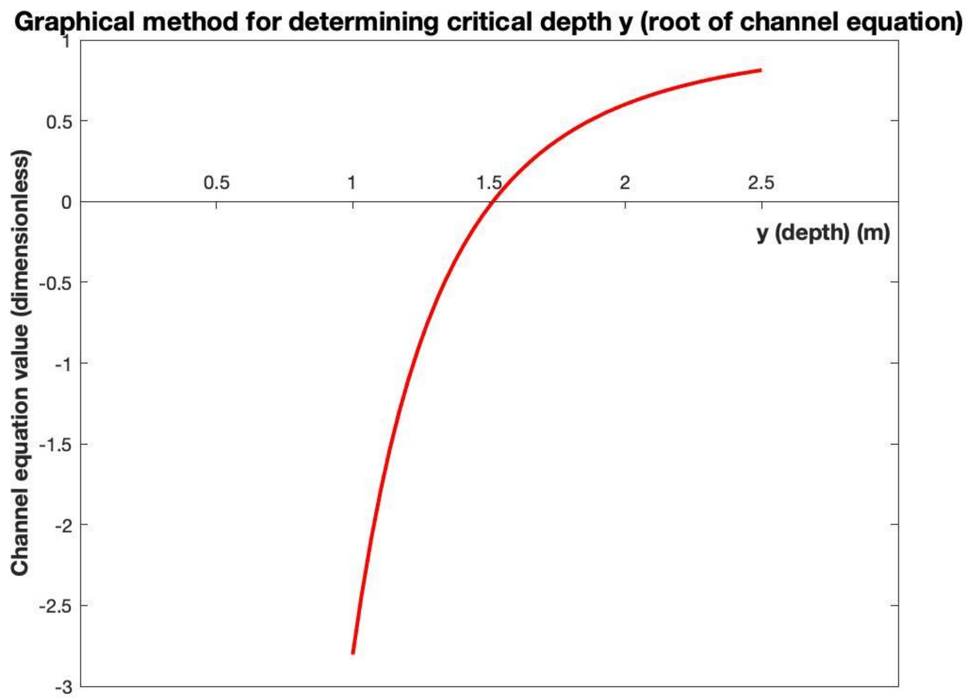
Robert Heeter
BIOE 391 Numerical Methods – Due 8 February 2022

Problem Set 3 Solutions

1. **Output**
   Bisection method output:
   c_d (kg/m) = 0.406250
   velocity (m/s) at 9s = 45.554755
   approx. relative error (%) = 4.615385
   iterations = 4

2. **Figure**

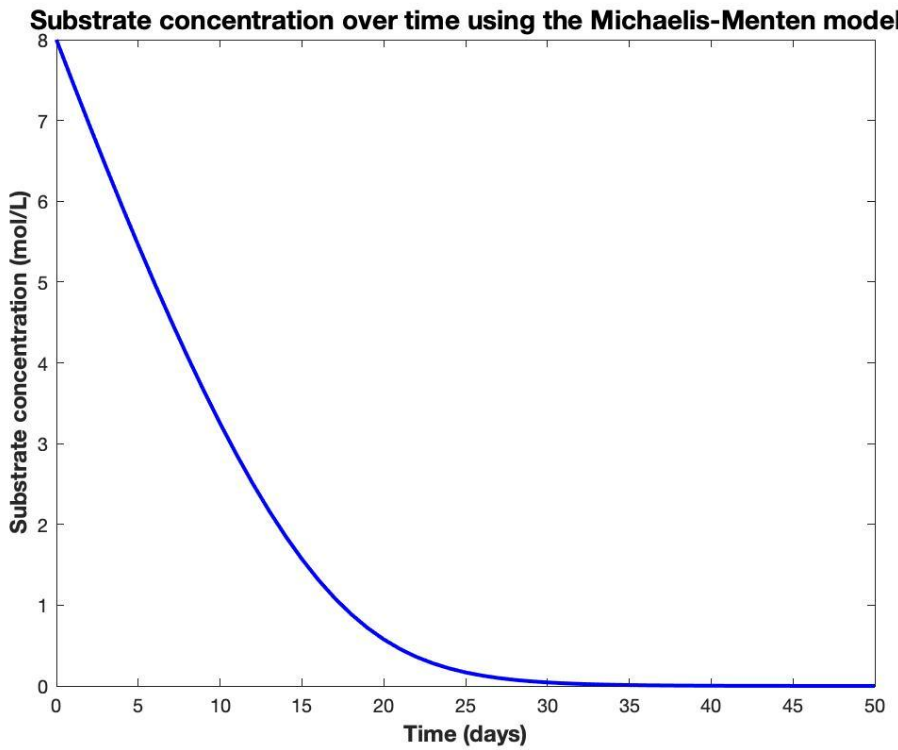


Output
Bisection method output:
y_crit (m) = 1.507812
value at y_crit = -0.013595
approx. relative error (%) = 0.518135
iterations = 8

False position method output:
y_crit (m) = 2.090766
value at y_crit = 0.656933
approx. relative error (%) = 1.591358
iterations = 10

The graphical method for determining the root is understandably the least accurate, though it does help confirm the results from the bisection and false position methods. The bisection method is more efficient than the false position method in this scenario (with an error threshold of 1% and an iteration limit of 10), with a 0.5% relative error compared to the false position method's higher 1.6% relative error. In most cases, the false position method operates better than the bisection method; however, here the function has a large curvature due to an asymptote at $x = 0$, which causes the false position method to converge more slowly as one of the bracket points must remain fixed (as shown in Figure 5.9 in the textbook). Both methods are still very effective computationally though, reaching under 2% error in 10 iterations.

3. **Figure**

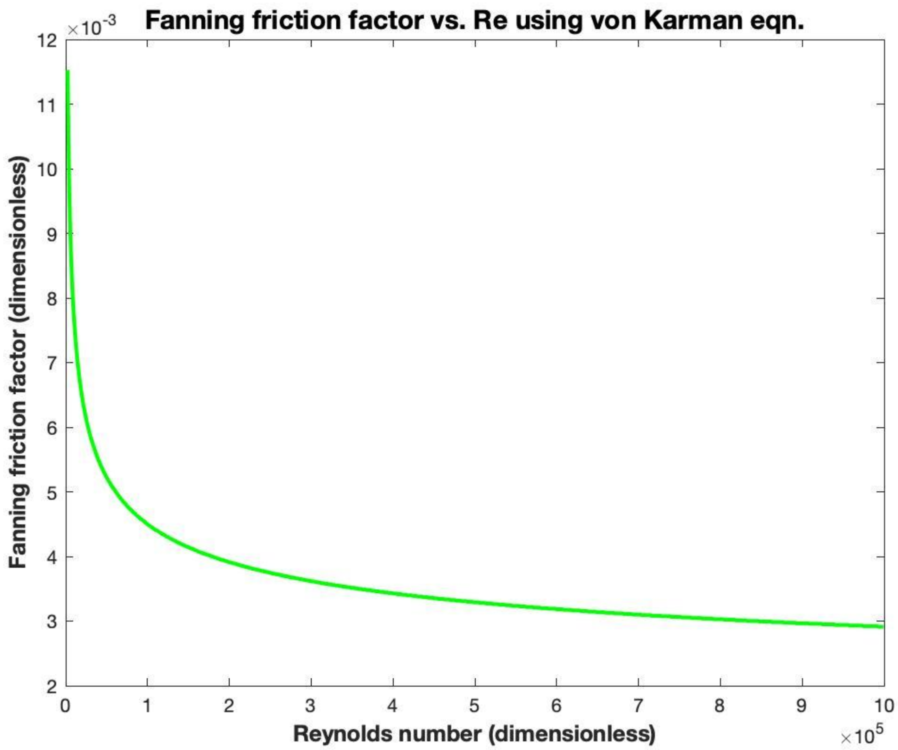


4. **Output**

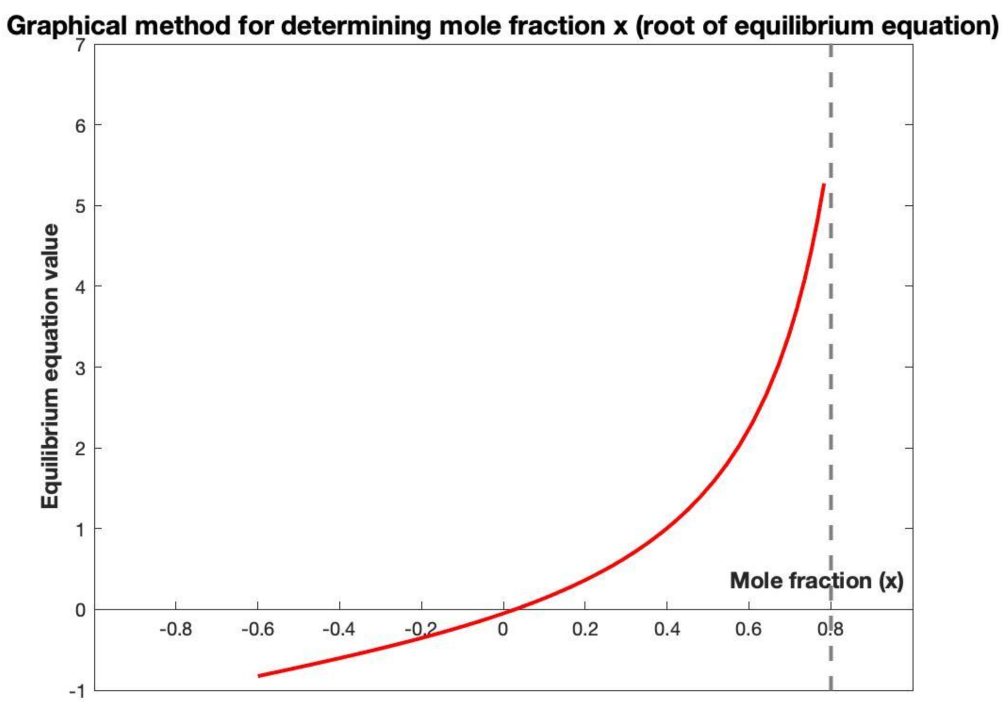
Re:	f:
2500	0.011528
3000	0.010890
10000	0.007728
30000	0.005877
100000	0.004503
300000	0.003622
1000000	0.002912

Robert Heeter
BIOE 391 Numerical Methods – Due 8 February 2022

Figure



5. **Figure**



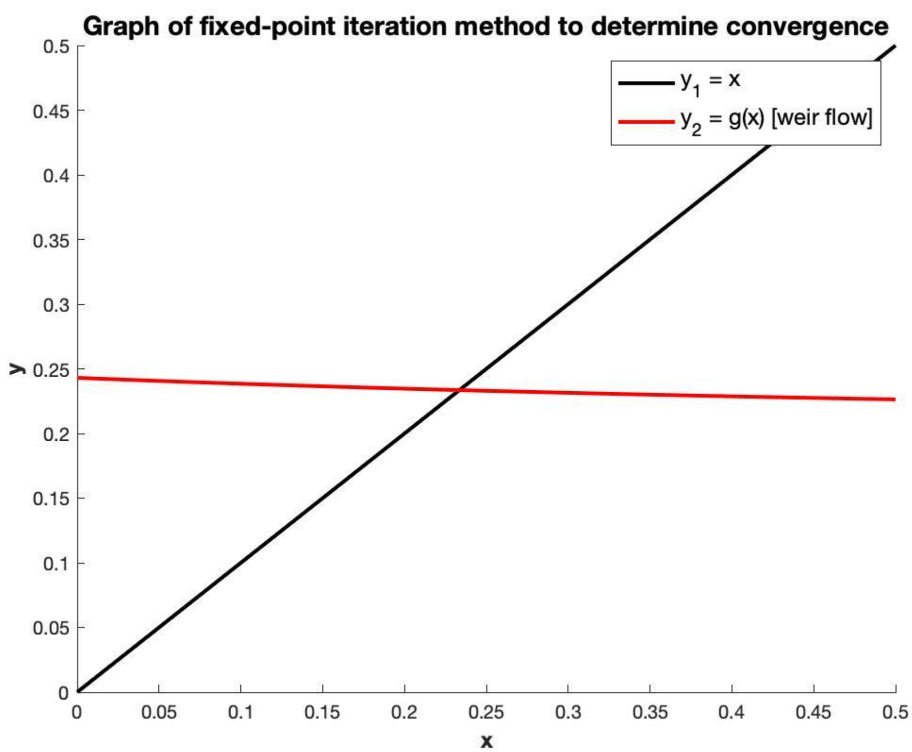
Output
Fzero output:
mole fraction (x) (dimensionless) = 0.028249

6. **Output**
Modified secant method output:
head above weir (H_h) (m) = 0.233520
upstream depth (H) (m) = 1.033520
approx. relative error (%) = 0.000000
iterations = 10

Fixed-point iteration method output:
head above weir (H_h) (m) = 0.233520
upstream depth (H) (m) = 1.033520
approx. relative error (%) = 0.000000
iterations = 10

Fzero output:
head above weir (H_h) (m) = 0.233520
upstream depth (H) (m) = 1.033520

Figure



This graph proves the convergence of the fixed-point iteration method in the case of the weir flow function, as the value of $|g'(x)| < 1$ around the root (intersection of $y=x$ and $y=g(x)$). As a result, iterations from a nearby starting value move closer to the root, as shown in Figure 6.3 in the textbook.

Robert Heeter
BIOE 391 Numerical Methods – Due 8 February 2022

7. **Output**
   Fzero output:
   smallest diameter pipe (D) (m) = 0.474178
   head loss value (h_L) (m) = 0.006000

8. **Output**
   Fzero output:
   fluid velocity (V) (m/s) = 2.631549
   volume flow rate (Q) (m^3/s) = 0.020668

Robert Heeter
BIOE 391 Numerical Methods – Due 8 February 2022

Complete MATLAB Code

```
% Robert Heeter
% BIOE 391 Numerical Methods
% HOMEWORK 3 MATLAB SCRIPT

clc, clf, clear, close all

%% P1. PROBLEM 5.1
disp('P1. PROBLEM 5.1');

m = 95; % mass (kg)
v_f = 46; % final velocity (m/s)
t = 9; % time (s)
g = 9.81; % gravitational acceleration (m/s^2)
v_rel = @(c_d) sqrt(g*m/c_d)*tanh(sqrt(g*c_d/m)*t) - v_f; % relative velocity equation (relative to
final velocity)

% Bisection method
xl = 0.2; % lower starting bound
xu = 0.5; % upper starting bound
er = 5; % relative error constraint (%)
[c_d_crit,fx,ea,iter] = bisection(v_rel,xl,xu,er); % use bisection function (below)

% Display results
disp('Bisection method output:')
fprintf('c_d (kg/m) = %f\nvelocity (m/s) at 9s = %f\napprox. relative error = %f\niterations =
%d\n\n',c_d_crit,fx+v_f,ea,iter);


%% P2. PROBLEM 5.12
disp('P2. PROBLEM 5.12');

Q = 20; % flow rate (m^3/s)
g = 9.81; % gravitational acceleration (m/s^2)
% A_c = 3*y+(y^2/2); cross-sectional area (m^2)
% B = 3+y; width of channel at surface (m)
ch = @(y) 1 - ((Q^2.*(3+y))./(g.*((3.*y)+(0.5.*y.^2)).^3)); % channel equation

% Graphical method (PART A)
figure
fplot(ch,[1, 2.5],'-r','LineWidth',2);
xlabel('y (depth)  (m)','FontSize',12,'FontWeight','bold');
ylabel('Channel equation value (dimensionless)','FontSize',12,'FontWeight','bold');
title('Graphical method for determining critical depth y (root of channel
equation)','FontSize',14,'FontWeight','bold');
ax = gca;
ax.XAxisLocation = 'origin';
axis([0 3 -3 1]);

% Bisection method (PART B)
xl = 0.5; % lower starting bound
xu = 2.5; % upper starting bound
er = 1; % relative error constraint (%)
maxit = 10; % iterations constraint
[y_crit_bi,fx_bi,ea_bi,iter_bi] = bisection(ch,xl,xu,er,maxit); % use bisection function (below)

% False position method (PART C)
[y_crit_fp,fx_fp,ea_fp,iter_fp] = falseposition(ch,xl,xu,er,maxit); % use false position function
(below) with same parameters as bisection (above)

% Compare results
disp('Bisection method output:');
fprintf('y_crit (m) = %f\nvalue at y_crit = %f\napprox. relative error = %f\niterations =
%d\n\n',y_crit_bi,fx_bi,ea_bi,iter_bi);
```

Robert Heeter

BIOE 391 Numerical Methods – Due 8 February 2022

```
disp('False position method output:');
fprintf('y_crit (m) = %f\nvalue at y_crit = %f\napprox. relative error = %f\niterations =
%d\n\n',y_crit_fp,fx_fp,ea_fp,iter_fp);


%% P3. PROBLEM 5.13
disp('P3. PROBLEM 5.13');

S_0 = 8; % initial substrate concentration (mol/L)
v_m = 0.7; % maximum uptake rate (mol/L/d)
k_s = 2.5; % half-saturation constant (mol/L)

t = (0:1:50)'; % time interval
S_vals = zeros(size(t)); % preallocate substrate concentration values
S_guess = S_0; % initial guess point for fzero

for i = 1:length(t) % iterate for each time value
    sc = @(S) S_0 - (v_m*t(i)) + (k_s*log(S_0/S)) - S; % Michaelis-Menten model
    root = fzero(sc,S_guess); % use fzero function to find substrate concentration
    S_guess = root;
    S_vals(i) = root;
end

% Plot results
figure
plot(t,S_vals,'-b','LineWidth',2);
xlabel('Time (days)','FontSize',12,'FontWeight','bold');
ylabel('Substrate concentration (mol/L)','FontSize',12,'FontWeight','bold');
title('Substrate concentration over time using the Michaelis-Menten
model','FontSize',14,'FontWeight','bold');


%% P4. PROBLEM 5.20
disp('P4. PROBLEM 5.20');

Re = 2500:1000:1000000; % interval of Reynolds numbers
f = zeros(size(Re)); % preallocate Fanning friction factor values
for i = 1:length(Re)
    f(i) = fanning(Re(i),0.0028,0.012,0.000005); % use von Karman equation to find Fanning friction
factor with function (below)
    % set lower bound = 0.0028, upper bound = 0.012, absolute error = 0.000005
end

% Plot results
figure
plot(Re,f,'-g','LineWidth',2);
xlabel('Reynolds number (dimensionless)','FontSize',12,'FontWeight','bold');
ylabel('Fanning friction factor (dimensionless)','FontSize',12,'FontWeight','bold');
title('Fanning friction factor vs. Re using von Karman eqn.','FontSize',14,'FontWeight','bold');

% Create table of values
Re = [2500 3000 10000 30000 100000 300000 1000000]';
f = zeros(size(Re));
for i = 1:length(Re)
    f(i) = fanning(Re(i),0.0028,0.012,0.000005); % use von Karman equation to find Fanning friction
factor with function (below)
    % set lower bound = 0.0028, upper bound = 0.012, absolute error = 0.000005
end
fprintf(' Re:         f: \n');
fprintf(' %7.0f     %8.6f\n',[Re(:),f(:)]');
disp(' ');


%% P5. PROBLEM 6.14
disp('P5. PROBLEM 6.14');
```

Robert Heeter
BIOE 391 Numerical Methods – Due 8 February 2022

```
p_t = 3; % total pressure of mixture (atm)
K = 0.05; % equilibrium constant

eq = @(x) ((x./(1-x)).*sqrt((2.*p_t)./(2+x))) - K; % equilibrium equation

% Graphical method
figure
fplot(eq,[-0.6, 0.8],'-r','LineWidth',2);
xlabel('Mole fraction (x)','FontSize',12,'FontWeight','bold');
ylabel('Equilibrium equation value','FontSize',12,'FontWeight','bold');
title('Graphical method for determining mole fraction x (root of equilibrium
equation)','FontSize',14,'FontWeight','bold');
ax = gca;
ax.XAxisLocation = 'origin';
axis([-1 1 -1 7]);

% Using fzero
x_root = fzero(eq,0.03); % use fzero function to find mole fraction with an initial guess based off of
graphical method

% Display results
disp('Fzero output:');
fprintf('mole fraction (x) (dimensionless) = %f\n\n',x_root);


%% P6. PROBLEM 6.31
disp('P6. PROBLEM 6.31');

g = 9.81; % gravitational acceleration (m/s^2)
H_w = 0.8; % height of weir (m)
B_w = 8; % weir width (m)
Q_w = 1.3; % flow across weir (m^3/s)

we = @(H_h) (1.125*sqrt((1+(H_h/H_w))/(2+(H_h/H_w)))*B_w*sqrt(g)*(2/3)^(3/2)*(H_h)^(3/2)) - Q_w; % weir
flow equation

iter = 10; % iterations constraint for part A and B
guess = 0.5*H_w; % initial guess for parts A-C

% Modified secant method (PART A)
pfrac = 10^-5; % perturbation fraction
[H_h_a,~,ea_a,iter_a] = modifiedsecant(we,guess,pfrac,0,iter); % use modified secant function (below)
H_a = H_w + H_h_a; % total upstream river depth

% Fixed-point iteration method with graph (PART B)
g_we = @(H_h)
(Q_w./((1.125.*sqrt((1+(H_h./H_w))./(2+(H_h./H_w))).*B_w.*sqrt(g).*(2/3).^(3/2)))).^(2/3); % weir flow
equation, solved for H_h = g(H_h)

figure
hold on
fplot(@(H_h) H_h,[0 0.5],'-k','LineWidth',2); % y_1 = x
fplot(g_we,[0 0.5],'-r','LineWidth',2); % y_2 = g(x)
xlabel('x','FontSize',12,'FontWeight','bold');
ylabel('y','FontSize',12,'FontWeight','bold');
title('Graph of fixed-point iteration method to determine
convergence','FontSize',14,'FontWeight','bold');
legend('y_1 = x','y_2 = g(x) [weir flow]','FontSize',12);
hold off

[H_h_b,~,ea_b,iter_b] = fixedpointiter(g_we,guess,0,iter); % use fixed-point iteration function (below)
H_b = H_w + H_h_b; % total upstream river depth

% Using fzero (PART C)
H_h_c = fzero(we,guess); % use fzero function
H_c = H_w + H_h_c; % total upstream river depth
```

Robert Heeter


```
% Compare results
disp('Modified secant method output:');
fprintf('head above weir (H_h) (m) = %f\nupstream depth (H) (m) = %f\napprox. relative error =
%f\niterations = %d\n\n',H_h_a,H_a,ea_a,iter_a);
disp('Fixed-point iteration method output:');
fprintf('head above weir (H_h) (m) = %f\nupstream depth (H) (m) = %f\napprox. relative error =
%f\niterations = %d\n\n',H_h_b,H_b,ea_b,iter_b);
disp('Fzero output:');
fprintf('head above weir (H_h) (m) = %f\nupstream depth (H) (m) = %f\n\n',H_h_c,H_c);


%% P7. PROBLEM 6.35
disp('P7. PROBLEM 6.35');

Q = 0.3; % volume flow rate (m^3/s)
h_L = 0.006; % head loss (m/m pipe)
v = 1.16e-6; % kinematic viscosity (m^2/s)
epsilon = 0.0004; % roughness (m)

[D,h_L_final] = headloss(Q,h_L,v,epsilon); % use headloss function (below) to find smallest diameter

% Display results
disp('Fzero output:');
fprintf('smallest diameter pipe (D) (m) = %f\nhead loss value (h_L) (m) = %f\n\n',D,h_L_final);


%% P8. PROBLEM 6.39
disp('P8. PROBLEM 6.39');

g = 9.81; % gravitational acceleration (m/s^2)
h = 24; % tower height (m)
L = 65; % horizontal pipe length (m)
d = 0.1; % pipe diameter (m)
L_eed = 30; % equivalent length for elbow (dimensionless)
L_evd = 8; % equivalent length for valve (dimensionless)
K = 0.5; % loss coefficient (dimensionless)
v = 1.2e-6; % kinematic viscosity of water (m^2/s)
epsilon = 0.005; % roughness (m)

% Equations
Re = @(V) (V*d)/v; % Reynolds number (dimensionless)
f = @(V) ((g*h)-(0.5*(V^2))-(K*0.5*(V^2)))/((((L+h)/d)+L_eed+L_evd)*0.5*(V^2)); % energy balance for
friction factor (dimensionless)
co = @(V) (1/sqrt(f(V))) + 2*log10((epsilon/(3.7*d))+(2.51/(Re(V)*sqrt(f(V))))); % Colebrook equation
for fluid velocity (m/s)

% Using fzero
V = fzero(co,1); % use fzero function to find fluid velocity (m/s)
Q = V*(pi*(d^2)*0.25); % actual fluid volume flow rate loss value (m^3/s)

% Display results
disp('Fzero output:');
fprintf('fluid velocity (V) (m/s) = %f\nvolume flow rate (Q) (m^3/s) = %f\n\n',V,Q);


%% Additional Functions

function [root,fx,ea,iter] = bisection(func,xl,xu,es,maxit,varargin)
% ABOUT: Bisection method for finding roots, adapted from textbook .m file.
% INPUTS: func = function; xl, xu = lower and upper bounds; es = desired
% relative error (as percent); maxit = maximum iterations
% OUTPUTS: root = real root; fx = function value at root; ea = approximate
% relative error (as percent); iter = number of iterations

if nargin < 3
```

```
    error('At least 3 input arguments required.')
end
test = func(xl,varargin{:})*func(xu,varargin{:});
if test > 0
    error('No sign change.')
end
if nargin < 4 || isempty(es)
    es = 0.0001;
end
if nargin < 5 || isempty(maxit)
    maxit = 50;
end

iter = 0;
xr = xl;
ea = 100;
while (1)
    xrold = xr;
    xr = (xl + xu)/2;
    iter = iter + 1;
    if xr ~= 0
        ea = abs((xr - xrold)/xr) * 100;
    end
    test = func(xl,varargin{:})*func(xr,varargin{:});
    if test < 0
        xu = xr;
    elseif test > 0
        xl = xr;
    else
        ea = 0;
    end
    if ea <= es || iter >= maxit
        break
    end
end

root = xr;
fx = func(xr, varargin{:});

end


function [root,fx,ea,iter] = falseposition(func,xl,xu,es,maxit)
% ABOUT: False position method for finding roots.
% INPUTS: func = function; xl, xu = lower and upper bounds; es = desired
% relative error (as percent); maxit = maximum iterations
% OUTPUTS: root = real root; fx = function value at root; ea = approximate
% relative error (as percent); iter = number of iterations

if nargin < 3
    error('At least 3 input arguments required/')
end
test = func(xl)*func(xu);
if test > 0
    error('No sign change.')
end
if nargin < 4 || isempty(es)
    es = 0.0001;
end
if nargin < 5 || isempty(maxit)
    maxit = 50;
end

iter = 0;
xr = xl;
ea = 100;
```

Robert Heeter


```
while (1)
    xrold = xr;
    xr = xu - (func(xu)*(xl - xu)/(func(xl)-func(xu)));
    iter = iter + 1;
    if xr ~= 0
        ea = abs((xr - xrold)/xr) * 100;
    end
    test = func(xl)*func(xr);
    if test < 0
        xu = xr;
    elseif test > 0
        xl = xr;
    else
        ea = 0;
    end
    if ea <= es || iter >= maxit
        break
    end
end

root = xr;
fx = func(xr);

end


function [fff] = fanning(re,xl,xu,ea)
% ABOUT: Fanning friction factor calculation from Reynolds number.
% INPUTS: re = Reynolds number, xl, xu = lower and upper bounds for
% calculation; ea = absolute error constraint
% OUTPUTS: fff = Fanning friction factor

iter = log2((xu-xl)/ea); % use Eqn. 5.6 to determine iterations for absolute error constraint
vk = @(f) (4*log10(re*sqrt(f)))-0.4-(1/sqrt(f)); % von Karman equation
[fff,~,~,~] = bisection(vk,xl,xu,0,iter); % use bisection function (above)

end


function [root,fx,ea,iter] = modifiedsecant(func,guess,pfrac,es,maxit)
% ABOUT: Modified secant method for finding roots (with only one point and
% a perturbation factor)
% INPUTS: func = function; guess = initial value for method; pfrac =
% perturbation factor; es = desired relative error (as percent); maxit =
% maximum iterations
% OUTPUTS: root = real root; fx = function value at root; ea = approximate
% relative error (as percent); iter = number of iterations

iter = 0;
x = guess;
while(1)
    x_i = x - ((pfrac*x*func(x))/(func(x+(pfrac*x))-func(x)));
    iter = iter + 1;
    if x_i ~= 0
        ea = abs((x_i - x)/x_i) * 100;
    end
    x = x_i;
    if ea <= es || iter >= maxit
        break
    end
end

root = x_i;
fx = func(x_i);

end
```

Robert Heeter

BIOE 391 Numerical Methods – Due 8 February 2022

```
function [root,fx,ea,iter] = fixedpointiter(gfunc,guess,es,maxit)
% ABOUT: Fixed-point iteration method for finding roots.
% INPUTS: gfunc = function in form x = g(x); guess = initial value for
% method; es = desired relative error (as percent); maxit = maximum
% iterations
% OUTPUTS: root = real root; fx = function value at root; ea = approximate
% relative error (as percent); iter = number of iterations

iter = 0;
x = guess;
while(1)
    x_i = gfunc(x);
    iter = iter + 1;
    if x_i ~= 0
        ea = abs((x_i - x)/x_i) * 100;
    end
    x = x_i;
    if ea <= es || iter >= maxit
        break
    end
end

root = x_i;
fx = gfunc(x_i);

end
```

```
function [D,h_L_final] = headloss(Q,h_L,v,epsilon)
% ABOUT: Pipe diameter calculation from provided equations, Colebrook
% equation, and Darcy-Weisbach equation given a head loss value.
% INPUTS: Q = volume flow rate; h_L = head loss; v = kinematic viscocity;
% epsilon = roughness
% OUTPUTS: D = pipe diameter; h_L_final = final head loss value

L = 1; % basis length of 1 meter
g = 9.81; % gravitational acceleration (m/s^2)
guess = 0.5; % initial guess for fzero, given head loss values range from 0 to 1

% Equations
V = @(D) Q/(pi*(D^2)*0.25); % velocity equation from flow rate and cross-sectional area (m/s)
f = @(D) (h_L*D*2*g)/(L*(V(D)^2)); % friction factor from Darcy-Weisbach equation (dimensionless)
Re = @(D) (V(D)*D)/v; % Reynolds number (dimensionless)
co = @(D) (1/sqrt(f(D))) + 2*log10((epsilon/(3.7*D))+(2.51/(Re(D)*sqrt(f(D))))); % Colebrook equation

% Using fzero
D = fzero(co,1); % use fzero function to find diameter (m)
h_L_final = (f(D)*L*(V(D)^2))/(D*2*g); % actual head loss value (m)

end
```