Solution 5.1

The function to evaluate is

$$f(c_d)=\sqrt{\frac{gm}{c_d}}\tanh\left(\sqrt{\frac{gc_d}{m}}t\right)-v(t)$$

or substituting the given values

$$f(c_d)=\sqrt{\frac{9.81(95)}{c_d}}\tanh\left(\sqrt{\frac{9.81c_d}{95}}9\right)-46$$

The first iteration is

$$x_r=\frac{0.2+0.5}{2}=0.35$$
$$f(0.2)f(0.35)=12.70647(2.3387193)=29.71688$$

Therefore, the root is in the second interval and the lower guess is redefined as x_u = 0.35. The second iteration is

$$x_r=\frac{0.35+0.5}{2}=0.425$$
$$\varepsilon_a=\left|\frac{0.425-0.35}{0.425}\right|\times 100\%=17.65\%$$
$$f(0.35)f(0.425)=2.3387193(-1.2809449)=-2.99577$$

Therefore, the root is in the first interval and the upper guess is redefined as x_u = 0.425. The remainder of the iterations are displayed in the following table:

i	x_l	$f(x_l)$	x_u	$f(x_u)$	x_r	$f(x_r)$	$\lvert\varepsilon_a\rvert$	$f(x_l)\times f(x_r)$
1	0.2	12.70647	0.5	-4.2485678	0.35	2.3387193		29.71688
2	0.35	2.338719	0.5	-4.2485678	0.425	-1.2809449	17.65%	-2.99577
3	0.35	2.338719	0.425	-1.2809449	0.3875	0.4340883	9.68%	1.015211
4	0.3875	0.434088	0.425	-1.2809449	0.40625	-0.4452446	4.62%	-0.19328

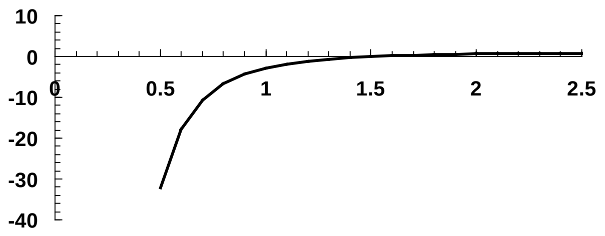
Thus, after four iterations, we obtain a root estimate of **0.40625** with an approximate error of 4.62%.

Solution 5.12

(a) The function to be evaluated is

$$f(y) = 1 - \frac{400}{9.81(3y + y^2/2)^3}(3 + y)$$

A graph of the function indicates a positive real root at approximately 1.5.



(b) Using bisection, the first iteration is

$$x_r = \frac{0.5 + 2.5}{2} = 1.5$$
$$f(0.5)f(1.5) = -32.2582(-0.030946) = 0.998263$$

Therefore, the root is in the second interval and the lower guess is redefined as x_l = 1.5. The second iteration is

$$x_r = \frac{1.5 + 2.5}{2} = 2 \qquad \varepsilon_a = \left|\frac{2 - 1.5}{2}\right| 100\% = 25\%$$

$$f(1.5)f(2) = -0.030946(0.601809) = -0.018624$$

Therefore, the root is in the first interval and the upper guess is redefined as x_u = 2. All the iterations are displayed in the following table:

i	x_l	$f(x_l)$	x_u	$f(x_u)$	x_r	$f(x_r)$	ε_a
1	0.5	-32.2582	2.5	0.813032	1.5	-0.030946	
2	1.5	-0.03095	2.5	0.813032	2	0.601809	25.00%
3	1.5	-0.03095	2	0.601809	1.75	0.378909	14.29%
4	1.5	-0.03095	1.75	0.378909	1.625	0.206927	7.69%
5	1.5	-0.03095	1.625	0.206927	1.5625	0.097956	4.00%
6	1.5	-0.03095	1.5625	0.097956	1.53125	0.036261	2.04%
7	1.5	-0.03095	1.53125	0.036261	1.515625	0.003383	1.03%
8	1.5	-0.03095	1.515625	0.003383	1.5078125	-0.013595	0.52%

Solution continued on next page...

After eight iterations, we obtain a root estimate of **1.5078125** with an approximate error of 0.52%.

(c) Using false position, the first iteration is

$$x_r = 2.5 - \frac{0.81303(0.5-2.5)}{-32.2582-0.81303} = 2.45083$$

$$f(0.5)f(2.45083) = -32.25821(0.79987) = -25.80248$$

Therefore, the root is in the first interval and the upper guess is redefined as x_u = 2.45083. The second iteration is

$$x_r = 2.45083 - \frac{0.79987(0.5-2.45083)}{-32.25821-0.79987} = 2.40363 \qquad \varepsilon_a = \left|\frac{2.40363-2.45083}{2.40363}\right| 100\% = 1.96\%$$

$$f(0.5)f(2.40363) = -32.2582(0.78612) = -25.35893$$

The root is in the first interval and the upper guess is redefined as x_u = 2.40363. All the iterations are displayed in the following table:

i	x_l	$f(x_l)$	x_u	$f(x_u)$	x_r	$f(x_r)$	ε_a
1	0.5	-32.2582	2.50000	0.81303	2.45083	0.79987	
2	0.5	-32.2582	2.45083	0.79987	2.40363	0.78612	1.96%
3	0.5	-32.2582	2.40363	0.78612	2.35834	0.77179	1.92%
4	0.5	-32.2582	2.35834	0.77179	2.31492	0.75689	1.88%
5	0.5	-32.2582	2.31492	0.75689	2.27331	0.74145	1.83%
6	0.5	-32.2582	2.27331	0.74145	2.23347	0.72547	1.78%
7	0.5	-32.2582	2.23347	0.72547	2.19534	0.70900	1.74%
8	0.5	-32.2582	2.19534	0.70900	2.15888	0.69206	1.69%
9	0.5	-32.2582	2.15888	0.69206	2.12404	0.67469	1.64%
10	0.5	-32.2582	2.12404	0.67469	2.09077	0.65693	1.59%

After ten iterations we obtain a root estimate of **2.09077** with an approximate error of 1.59%. Thus, after ten iterations, the false position method is converging at a very slow pace and is still far from the root in the vicinity of 1.5 that we detected graphically.

Discussion: This is a classic example of a case where false position performs poorly and is inferior to bisection. Insight into these results can be gained by examining the plot that was developed in part **(a)**. This function violates the premise upon which false position was based–that is, if $f(x_u)$ is much closer to zero than $f(x_l)$, then the root is closer to x_u than to x_l (recall Figs. 5.8 and 5.9). Because of the shape of the present function, the opposite is true.

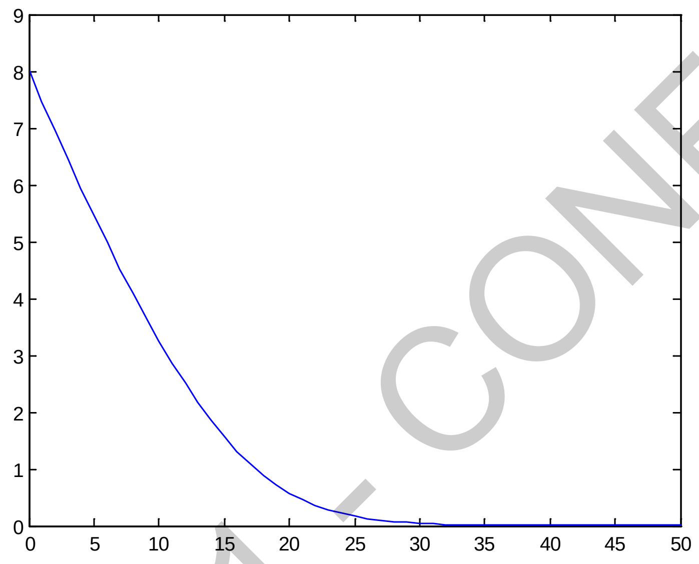
Solution 5.13

The problem amounts to determining the root of

$$f(S) = S_0 - v_m t + k_s \ln(S_0 / S) - S$$

The following script calls the `bisect` function (Fig. 5.7) for various values of *t* in order to generate the solution.

```
S0=8;vm=0.7;ks=2.5;
f = @(S,t) S0 - vm * t + ks * log(S0 / S) - S;
t=0:50;S=0:50;
n=length(t);
for i = 1:n
  S(i)=bisect(f,0.00001,10.01,1e-6,100,t(i));
end
plot(t,S)
```



Solution 5.20

The solution can be formulated as

$$f(f) = 4\log_{10}(\mathrm{Re}\sqrt{f}) - 0.4 - \frac{1}{\sqrt{f}}$$

We want our program to work for Reynolds numbers between 2,500 and 1,000,000. Therefore, we must determine the friction factors corresponding to these limits. This can be done with any root location method to yield 0.011525 and 0.002913. Therefore, we can set our initial guesses as x_l = 0.0028 and x_u = 0.012. Equation (5.6) can be used to determine the number of bisection iterations required to attain an absolute error less than 0.000005,

$$n = \log_2\left(\frac{\Delta x^0}{E_{a,d}}\right) = \log_2\left(\frac{0.012 - 0.0028}{0.000005}\right) = 10.8454$$

which can be rounded up to 11 iterations. Here is a MATLAB function that is set up to implement 11 iterations of bisection to solve the problem.

```
function f=Fanning(func,xl,xu,varargin)
test = func(xl,varargin{:})*func(xu,varargin{:});
if test>0,error('no sign change'),end
for i = 1:11
  xr = (xl + xu)/2;
  test = func(xl,varargin{:})*func(xr,varargin{:});
  if test < 0
    xu = xr;
  elseif test > 0
    xl = xr;
  else
    break
  end
end
f=xr;
```

This can be implemented in MATLAB. For example,

```
>> vk=@(f,Re) 4*log10(Re*sqrt(f))-0.4-1/sqrt(f);
>> format long
>> f=Fanning(vk,0.0028,0.012,2500)

f =
   0.01152832031250
```

Solution continued on next page...

Here are additional results for a number of values within the desired range. We have included the true value and the resulting error to verify that the results are within the desired error criterion of $E_a < 5\times10^{-6}$.

Re	Root	Truth	E_t
2500	0.0115283203125	0.0115247638118	3.56×10^{-6}
3000	0.0108904296875	0.0108902285840	2.01×10^{-7}
10000	0.0077279296875	0.0077271274071	8.02×10^{-7}
30000	0.0058771484375	0.0058750482511	2.10×10^{-6}
100000	0.0045025390625	0.0045003757287	2.16×10^{-6}
300000	0.0036220703125	0.0036178949673	4.18×10^{-6}
1000000	0.0029123046875	0.0029128191460	5.14×10^{-7}

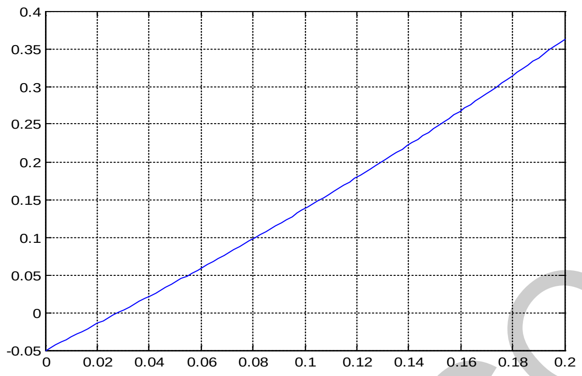
Solution 6.14

The solution involves determining the root of

$$f(x) = \frac{x}{1-x}\sqrt{\frac{6}{2+x}} - 0.05$$

MATLAB can be used to develop a plot that indicates that a root occurs in the vicinity of $x = 0.03$.

```
f=@(x) x./(1-x).*sqrt(6./(2+x))-0.05;
x = linspace(0,.2);
y = f(x);
plot(x,y),grid
```



The `fzero` function can then be used to find the root

```
format long
fzero(f,0.03)

ans =
   0.028249441148471
```

Solution 6.31

(a) Substituting Eq. (2) into Eq. (1) and collecting terms gives

$$Q_w = 1.125\sqrt{\frac{1+H_h/H_w}{2+H_h/H_w}}B_w\sqrt{g}\left(\frac{2}{3}\right)^{3/2}H_h^{3/2}$$

Equation (3) can be expressed as a roots problem

$$f(H_h) = 1.125\sqrt{\frac{1+H_h/H_w}{2+H_h/H_w}}B_w\sqrt{g}\left(\frac{2}{3}\right)^{3/2}H_h^{3/2} - Q_w$$

Substituting parameters

$$f(H_h) = 15.34405\sqrt{\frac{1+H_h/0.8}{2+H_h/0.8}}H_h^{3/2} - 1.3$$

<u>First iteration:</u>

$x_0 = 0.4$	$f(x_0) = 1.706808275$
$x_0 + \delta x_0 = 0.4000040$	$f(x_0 + \delta x_0) = 1.706855381$

$$x_1 = 0.4 - \frac{10^{-5}(0.4)1.706808275}{1.706855381 - 1.706808275} = 0.4 - \frac{1.706808275}{11.7767} = 0.255069$$

$$\varepsilon_a = \left|\frac{0.255069 - 0.4}{0.255069}\right| \times 100\% = 56.82\%$$

The computation can be continued as shown in the following table:

iteration	H_h	$f(H_h)$	$H_h + \delta H_h$	$f(H_h + \delta H_h)$	$f'(H_h)$	$\lvert\varepsilon_a\rvert$
0	0.4	1.706808275	0.4000040	1.706855381	11.776700	
1	0.255069	0.190689500	0.2550716	0.190712637	9.071067	56.82%
2	0.234047	0.004549614	0.2340497	0.004569827	8.635981	8.98%
3	0.233521	0.000002932	0.2335228	0.000023073	8.624897	0.23%

Solution continued on next page...

(b) The problem can be solved with a fixed-point or successive substitution approach. This is done by rearranging it so that the last H_h is brought to the left-hand side,

$$H_h = \left(\frac{Q_w}{1.125\sqrt{\dfrac{1+H_h/H_w}{2+H_h/H_w}}B_w\sqrt{g}\left(\dfrac{2}{3}\right)^{3/2}} \right)^{2/3}$$

Substituting the values from this problem gives

$$H_h = \left(\frac{0.084723}{\sqrt{\dfrac{1+H_h/0.8}{2+H_h/0.8}}} \right)^{2/3}$$

First iteration:

$$H_h = \left(\frac{0.084723}{\sqrt{\dfrac{1+0.4/0.8}{2+0.4/0.8}}} \right)^{2/3} = 0.22871$$

$$\varepsilon_a = \left| \frac{0.22871-0.4}{0.255069} \right| \times 100\% = 74.89\%$$

The computation can be continued as shown in the following table:

i	x_i	$\lvert\varepsilon_a\rvert$
0	0.4	
1	0.22871	74.89%
2	0.233679	2.13%
3	0.233515	0.07%

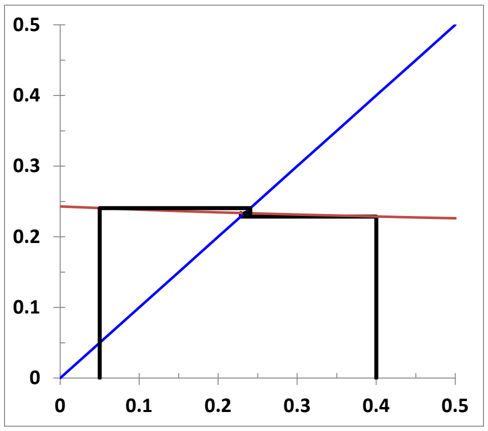
Solution continued on next page...

We can prove that the method will be convergent for all positive initial guesses by graphing the following versus H_h,

$$y_1 = H_h$$

$$y_2 = \left(\frac{0.084723}{\sqrt{\dfrac{1 + H_h / 0.8}{2 + H_h / 0.8}}} \right)^{2/3}$$

As in the resulting plot, the second function is really flat, so not only is the method convergent, but it does so rapidly.



(c) A script and function can be developed to use `fzero` to solve this problem:

Script:

```
function f = fHh(Hh,Hw,Bw,Qw)
g=9.81;
f = 1.125 * sqrt((1 + Hh / Hw) / (2 + Hh / Hw)) * ...
       Bw * sqrt(g) * (2 / 3) ^ (3 / 2) * Hh ^ (3 / 2) - Qw;
```

Function:

```
clear,clc
Hw=0.8; Bw=8; Qw=1.3; Hh0=0.4;
format long
[x,fx] = fzero(@fHh,Hh0,[],Hw,Bw,Qw)
```

Results:

```
x =
   0.233520169358115
fx =
     4.440892098500626e-16
```

Solution 6.35

For a circular pipe, Eq. (P6.35.2) can be solved for

$$V = \frac{4Q}{\pi D^2}$$

This result can be substituted into Eq. (P6.35.1) and the result solved for

$$f = h_L \frac{gD^5\pi^2}{8LQ^2}$$

This result can be substituted into the Colebrook equation which can then be formulated as a roots problem

$$f(D) = \frac{1}{\sqrt{h_L \frac{gD^5\pi^2}{8LQ^2}}} + 2\log\left(\frac{\varepsilon/D}{3.7} + \frac{2.51}{\mathrm{Re}\sqrt{h_L \frac{gD^5\pi^2}{8LQ^2}}}\right)$$

Script:

```
clear,clc
format compact
Q=0.3; hL=0.006; L=1; vw=1.16e-06;eps=0.4; eps=eps/1e3;d0=1;
[D,fD] = fzero(@fpipeD,d0,[],Q,hL,L,vw,eps);
D*1e3,fD
```

Function:

```
function fd = fpipeD(d,Q,hL,L,vw,eps)
g=9.81; V=Q/(pi*d^2/4);
ed=eps/d; Re=V*d/vw;
f=d*hL*2*g/(L*V^2);
fd=2*log10(ed/3.7+2.51/Re/sqrt(f))+1/sqrt(f);
```

Results:

```
ans =
     4.741778364172691e+02
fD =
    -1.776356839400251e-15
```

Therefore, the inside diameter is 474.1778 mm. You would, therefore, pick the next larger standard diameter pipe to achieve the prescribed behavior.

Solution 6.39

```
clear,clc
global g
format compact
g=9.81;
h = 24; L = 65; d = .1; Lee_d = 30; Lev_d = 8;
K = 0.5; nu = 1.2e-6; e_d=0.005;
v=fzero(@ftank,1,[],h,L,d,Lee_d,Lev_d,K,nu,e_d)
Q=v*pi*d^2/4
f=Colebrook(v,d,nu,e_d)

function ft = ftank(v,h,L,d,Lee_d,Lev_d,K,nu,e_d)
global g
f = Colebrook(v,d,nu,e_d);
ft=g*h-v^2/2-f*((L+h)/d+Lee_d+Lev_d)*v^2/2-K*v^2/2;
end


function f = Colebrook(v,d,nu,e_d)
global g
Re=v*d/nu;
ff=@(f) 1/sqrt(f)+2*log10(e_d/3.7+2.51/(Re*sqrt(f)));
f=fzero(ff,.05);
end

v =
    3.9650
Q =
    0.0311
f =
    0.0307
```