Applications of Non-linear Equations with SCILAB

By

Gilberto E. Urroz, Ph.D., P.E.

Distributed by

InfoClearinghouse.com

A "zip" file containing all of the programs in this document (and other SCILAB documents at InfoClearinghouse.com) can be downloaded at the following site:

http://www.engineering.usu.edu/cee/faculty/gurro/Software_Calculators/Scilab_Docs/ScilabBookFunctions.zip

The author's SCILAB web page can be accessed at:

http://www.engineering.usu.edu/cee/faculty/gurro/Scilab.html

Please report any errors in this document to: gurro@cc.usu.edu

Applications of non-linear equations

In this section we present solutions of non-linear equations that arise from applications to the physical sciences.

Projectile motion

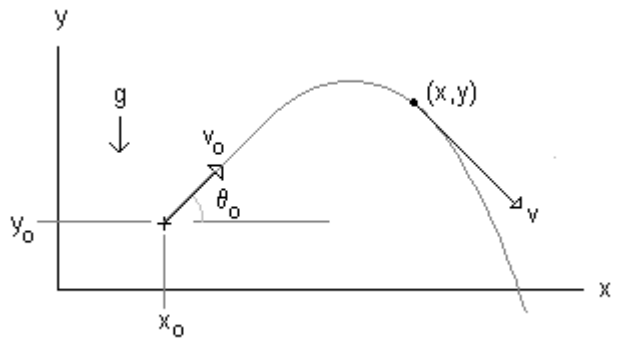
The motion of a projectile in a Cartesian coordinate system as shown in the figure below is described by the equations:

$$a_x = 0,\ a_y = -g,$$

$$v_x = v_0 \cdot \cos(\theta_0),\ v_y = v_0 \cdot \sin(\theta_0) - gt,$$

$$x = x_0 + v_0 \cdot \cos(\theta_0) \cdot t,\ y = y_0 + v_0 \cdot \sin(\theta_0) \cdot t - g \cdot t^2/2,$$

where *a* stands for acceleration, *v* for velocity, and (*x*,*y*) are the positions of the projectile at time *t*. It is implied that the projectile was launched from point (x_0,y_0) at time $t = 0$ with an initial velocity of magnitude $v = v_0$ at an angle above the positive x-axis of $\theta = \theta_0$. The variable *g* represents the acceleration of gravity ($g = 9.806\ m/s^2 = 32.2\ ft/s^2$).



The velocity vector, illustrated in the figure, can be written as $\mathbf{v} = v_x\mathbf{i} + v_y\mathbf{j}$. The equation of the trajectory can be obtained by replacing $t = (x-x_o)/(v_o \cdot \cos(\theta_o))$, into the equation for y, resulting in

$$y = y_0 + \tan(\theta_0) \cdot (x - x_0) - \frac{g}{2 \cdot v_0^2 \cdot \cos^2(\theta_o)} \cdot (x - x_0)^2.$$

Example 1. A projectile is launched from point $(x_0,y_0) = (0,0)$ with a velocity $v_0 = 25$ m/s at an unknown angle θ_0. If the projectile passes through point (x,y) = (2,3), determine the angle θ_0.

Replacing the values of the data given in the problem statement together with the appropriate value of *g* in the equation of the trajectory, we can form a single non-linear equation on θ_0 defined by

$$f(\theta_0) = y_0 + \tan(\theta_0)\cdot(x - x_0) - \frac{g}{2\cdot v_0^2\cdot\cos^2(\theta_o)}\cdot(x - x_0)^2 - y = 0.$$

The solution using SCILAB can be found as follows:

First, we enter the known data:

```
-->x0 = 0, y0 = 0, v0 = 25, x = 2, y = 3, g = 9.806
 x0  =

    0.
 y0  =

    0.
 v0  =

    25.
 x  =

    2.
 y  =

    3.
 g  =

    9.806
```

Next, we define the function $f(\theta_o)$.

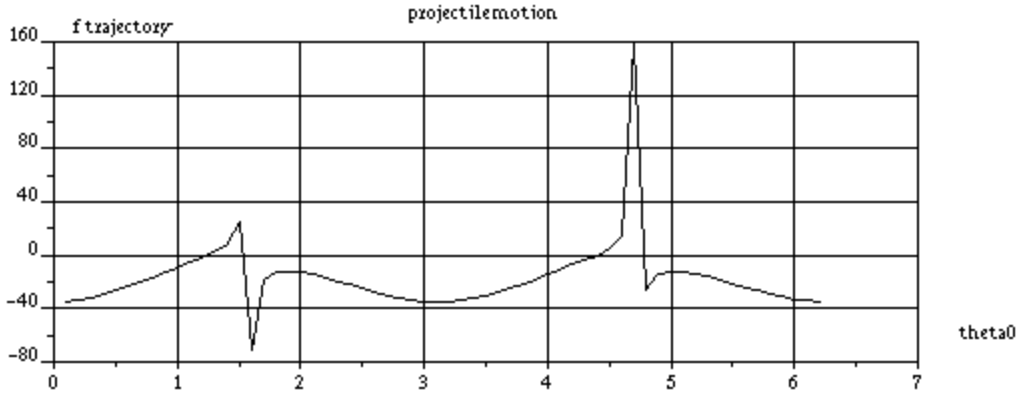
```
-->deff('[f]=traj(theta0)',...
-->'f = y0+tan(theta0).*(x-x0)-g.*(x-x0)^2.\(2.*v0^2.*cos(theta0)^2)-y')
```

Notice that in the definition we used the left division (\) in the third term of the function. We do this in order to be able to plot the function given an array of values of θ_0 as will be shown next. The variable *th* is an array of values of θ_0, starting at *0.1* and ending at $6.28 \approx 2\pi$. The variable *fth* stores the values of the function *traj*(*theta0*), representing $f(\theta_0)$, and corresponding to *th*.

```
-->th = [0.1:0.1:6.28]'; fth = traj(th);
```

We use those variables, *th* and *fth*, to plot the function $f(\theta_0)$ with the purpose of identifying possible solutions.

```
-->plot(th,fth,'theta0','f_trajectory','projectile motion')

-->xgrid()
```

The figure is shown above. From the figure we notice that there are two possible solutions, one near 1.0 and one near 4.0. To obtain the solutions we use SCILAB'S function *fsolve.* The two solutions are shown next. The solutions are given in radians, the natural unit of angular measurement. We also show the equivalent values in degrees, through the formula,

$$\theta^o = 180 \cdot \theta^r / \pi:$$

```
-->th0 = fsolve(1,traj)
 th0  =

    1.2538009

-->180*th0/%pi
 ans  =

    71.8375

-->th0 = fsolve(4,traj)
 th0  =

    4.3953936

-->180*th0/%pi
 ans  =
    251.8375
```

Next, we produce a plot of the projectile trajectory corresponding to $\theta_0 = 1.2538009$ rad. We start by clearing the variables and defining the equation of the trajectory:

```
-->clear

-->deff('[y]=f(x)',...
-->'y=y0+tan(theta0)*(x-x0)-g*(x-x0)^2/(2*v0^2*cos(theta0)^2)')
```

Next, we load the constant values:

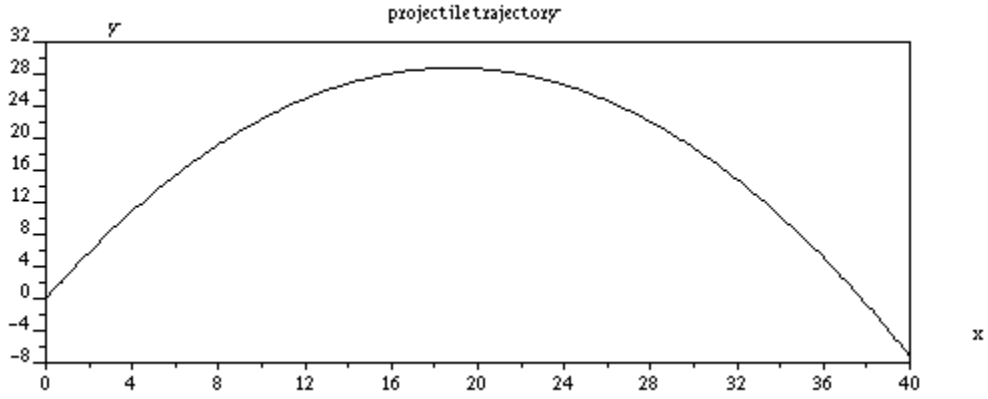
```
-->x0 = 0; y0 = 0; v0 = 25; x = 2; y = 3; g = 9.806; theta0 = 1.2538009;
```

The plot is produced by the following statements:

```
-->xx = [0:0.1:40]';
```



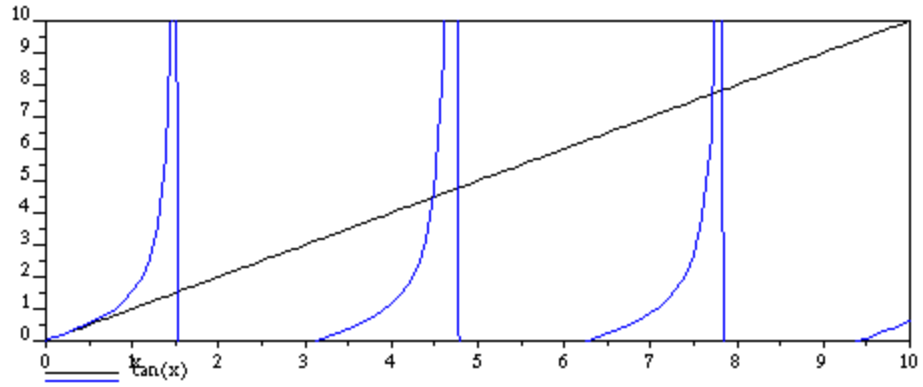
```
-->yy = f(xx);

-->plot(xx,yy,'x','y','projectile trajectory')
```



Solving the equation *tan(x) = x*

The equation *tan(x) = x* shows up in problems of differential equations when determining so-called eigenvalues and eigenfunctions. In this example we solve the equation for values of x > 0. To see possible solutions we plot the functions $f_1(x) = x$ and $f_2(x) = tan(x)$ in the same set of axes by using:

```
-->deff('[y]=f1(x)','y=x')

-->deff('[y]=f2(x)','y=tan(x)')

-->xx = [0:0.1:20]'; yy1 = f1(xx); yy2 = f2(xx);

-->plot2d([xx xx],[yy1 yy2],[1,2],'111','x@tan(x)',[0 0 10 10])
```



We can see at least three roots one is zero, the other two are near 4 and 7.5. To solve the corresponding equation we define the function $f_x(x) = tan(x)-x$, and solve it using SCILAB's function *fsolve*:

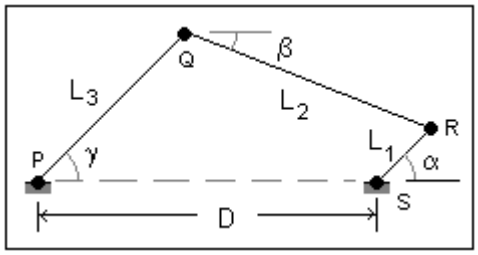
```
-->deff('[y]=fx(x)','y=tan(x)-x')

-->x1 = fsolve(0.5,fx)
 x1  =

    9.218E-09

-->x2 = fsolve(4.1,fx)
 x2  =

    4.4934095

-->x3 = fsolve(7.5,fx)
 x3  =

    7.7252518
```

To verify that the solutions satisfy the equation $f_x(x) = 0$, try:

```
-->fx(x1), fx(x2), fx(x3)
 ans  =

    0.
 ans  =

    8.882E-16
 ans  =

  - 2.309E-14
```

Analysis of a simple three-bar mechanism

Consider the mechanism shown in the figure below. Bar SR has a fixed hinge at S, while bar PQ has a fixed hinge at P. Bar SR is animated by a rotational motion about point S that drives the mechanism forcing bar PQ to rotate about point P. The analysis in this case consists in determining the angle γ (output angle) given the angle α (input angle).



You can verify from the figure that:

$$L_3 \sin\gamma = L_1 \sin\alpha + L_2 \sin\beta$$
$$L_3 \cos\gamma + L_2 \cos\beta - L_1 \cos\alpha = D$$

This is a system of two non-linear equations in two unknowns, β and γ, for a given value of α. The system can be written as follows:

$$f_1(\beta,\gamma) = -L_2 \sin\beta + L_3 \sin\gamma - L_1 \sin\alpha = 0$$
$$f_2(\beta,\gamma) = L_2 \cos\beta + L_3 \cos\gamma - L_1 \cos\alpha - D = 0$$

These system of non-linear equations can be entered into SCILAB as the following file function:

```
function [f] = fmech(angle)

//evaluates f1(beta,gama) & f2(beta,gamma)
//for the case of a three-bar mechanism
//angle(1) = beta, angle(2) = gamma

f = zeros(2,1);
f(1) = -L2*sin(angle(1))+L3*sin(angle(2))-L1*sin(alpha);
f(2) =  L2*cos(angle(1))+L3*cos(angle(2))-L1*cos(alpha)-D;

//end function
```

Next, we will use SCILAB function *fsolve* to obtain a table of values of γ given a set of values of α. We will use the values D = 4, L_1 = 6, L_2 = 3, L_3 = 7, and α = 0, $\pi/20$, $2\pi/20$, …, $19\pi/20$. First, we load the function fmech:

```
-->getf('fmech')
```

Next, we define the constant values:

```
-->L1=6;L2=3;L3=7;D=4;

-->alphav = [0:%pi/20:19*%pi/20];
```

The size of *alphav* (the *v* stands for vector) will be used later in calculating the angles β and γ:

```
-->[m n] = size(alphav)
 n  =

    20.
 m  =

    1.
```

The angles β and γ will be obtained as rows in the matrix angles, which is first defined as an empty array:

```
-->angles = [];
```

The next *for* loop will solve for values of angles β and γ for each value of *alphav*, which is temporarily stored in variable *alpha*:

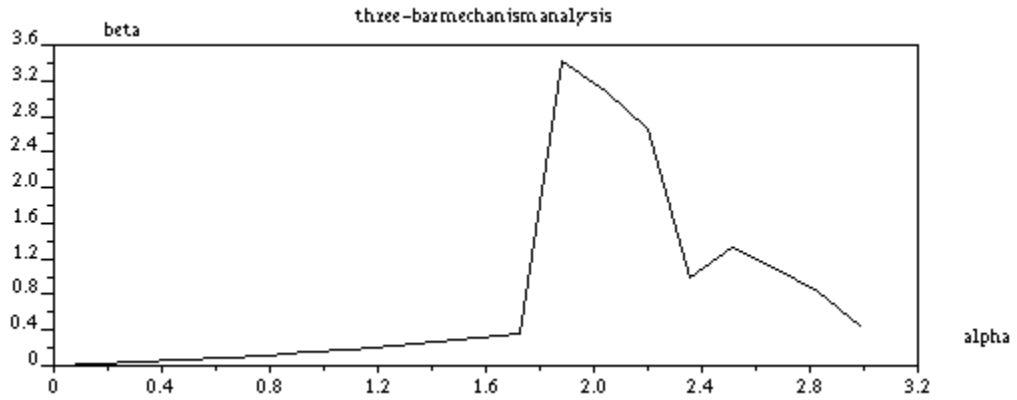

```
-->for j = 1:m
-->    alpha = alphav(j);
-->    angles = [angles, fsolve([alpha;alpha],fmech)];
-->end;
```

The following assignment statements load the values of the angles β and γ in separate vectors:

```
-->beta = angles(1,:);

-->gama = angles(2,:);
```

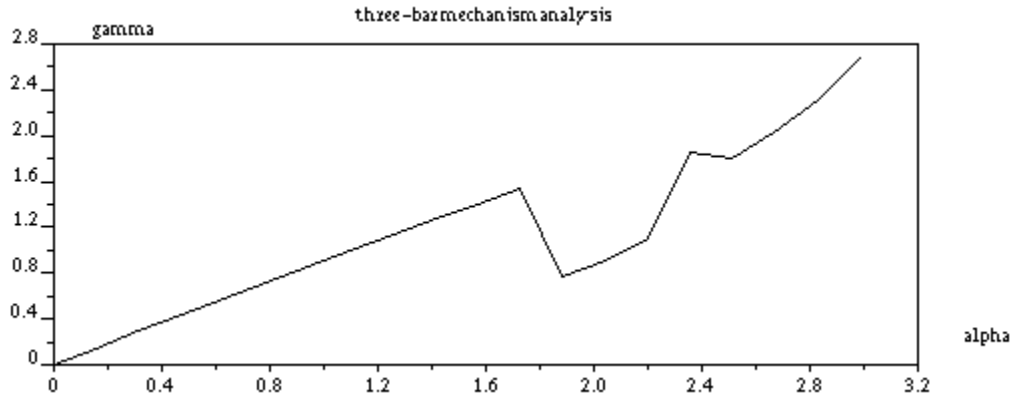
Next, we produce a plot of β-vs-α (both in radians):

```
-->plot(alphav,beta,'alpha','beta','three-bar mechanism analysis')
```



The following is a plot of γ-vs-α (both in radians):

```
-->plot(alphav,gama,'alpha','gamma','three-bar mechanism analysis')
```



Solving the Darcy-Weisbach and Coolebrook-White equations for pipeline flow

The figure below shows the components of the energy equation for turbulent flow in a pipe. The datum is a horizontal reference level from which the elevation of the pipe centerline is measured. The energy line (E.L.) is a graphical representation of the total head along the pipe. The hydraulic grade line (H.G.L.) represents the piezometric head along the pipe.

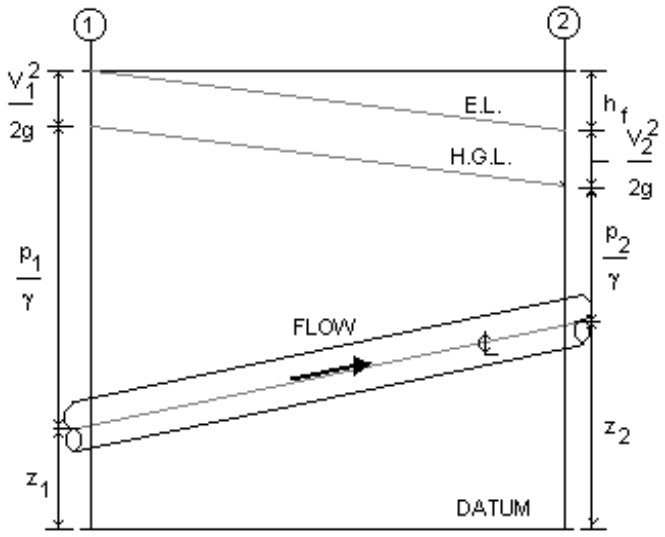
Let the pipe length between cross-sections 1 and 2 be L. The slope of the energy line is defined as $S_f = h_f/L$, where h_f is the energy losses due to friction on the length L.

The energy equation for the figure above can be written by simply adding the lengths of the different energy heads shown, i.e.,

$$z_1 + \frac{p_1}{\gamma} + \frac{V_1^2}{2g} = z_2 + \frac{p_2}{\gamma} + \frac{V_2^2}{2g} + h_f$$

For a constant-diameter pipeline $V_1 = V_2$, and using the *piezometric head*, $h = z + p/\gamma$, the energy loss is equal to the difference in piezometric heads, i.e.,

$$h_f = h_1 - h_2 = (z_1 + p_1/\gamma) - (z_2 + p_2/\gamma).$$



The Darcy-Weisbach equation provides a way to calculate the friction head loss, h_f, based of fluid properties and flow characteristics:

$$h_f = f \cdot \frac{L}{D} \cdot \frac{V^2}{2g},$$

in which, L and D are the length and diameter of the pipe, V is the mean flow velocity, g is the acceleration of gravity, and f is known as the Darcy-Weisbach friction factor. The friction factor, *f = f(e/D, Re),* is a function of the parameters e/D or *relative roughness*, and the *Reynolds number*, Re. The parameter *e*, known as the absolute roughness of the pipeline is a measure of the roughness heights of the pipeline's inner wall, and the Reynolds number is defined as

$$\mathrm{Re} = \frac{\rho VD}{\mu} = \frac{VD}{\nu},$$

where ν is the fluid's kinematic viscosity.

An expression describing the variation of *f = f(e/D, Re)* for turbulent flow is the Coolebrook-White equation given by:

$$\frac{1}{\sqrt{f}} = -2 \cdot \log\left(\frac{e}{3.7D} + \frac{2.51}{\mathrm{Re}\sqrt{f}}\right)$$

Here, *log* represents the logarithm of base 10. In SCILAB, the function *log* represents the natural logarithm, i.e., the logarithm of base *e* = 2.718281828, which is typically written in paper as *ln*. Using natural logarithms, we re-write the Coolebrook-White equation to read:

$$\frac{1}{\sqrt{f}} = -0.8686 \cdot \ln\left(\frac{e}{3.7 \cdot D} + \frac{2.51}{\mathrm{Re}\sqrt{f}}\right)$$

This function is implicit on *f*, therefore, suitable for solution through the methods of non-linear equations presented in this chapter.

Example 1. Determining the friction factor.

As an example, try using the following values e = 0.00001m, D = 0.25 m, Re = 1×10^6 to determine the corresponding friction factor. Here is the solution using SCILAB: First, we define the function for the Coolebrook-White equation:

```
-->deff('[P]=CW(f)','...
-->P=1/sqrt(f)+0.8686*log(e/(3.7*D)+2.51/(Re*sqrt(f)))')
```

Next, we enter the constant values:

```
-->e = 0.00001; D = 0.25; Re = 1e6;
```

The corresponding friction factor is calculated as:

```
-->f = fsolve(0.02,CW)
 f  =

     .0124687
```


Example 2. Plotting the Moody diagram.

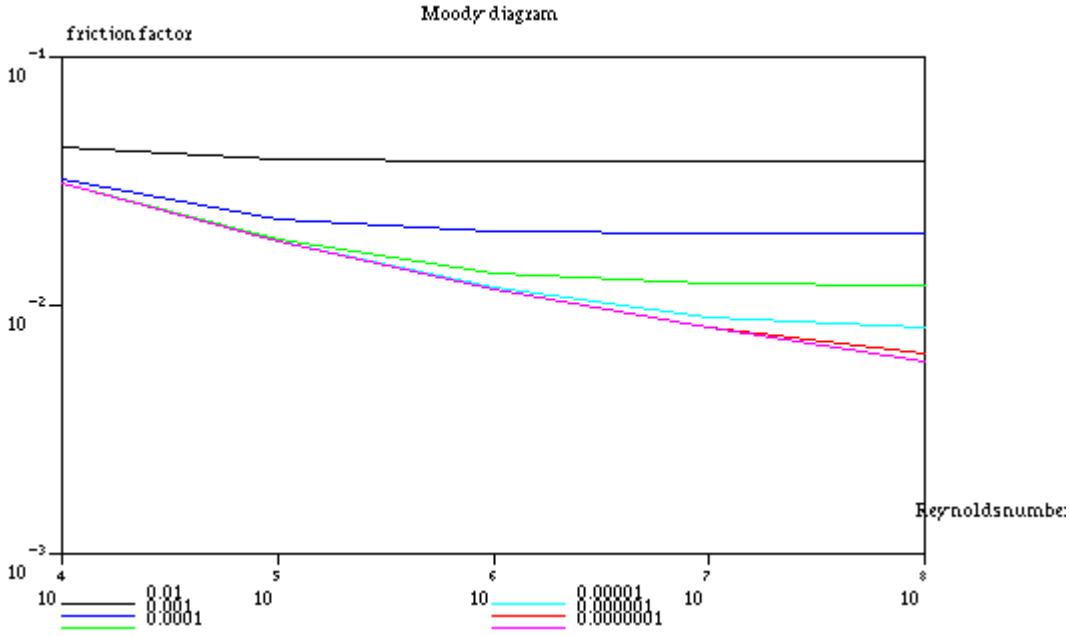
The function described by the Coolebrook-White equation is typically plotted in log-log scale with the Reynolds number, Re, in the x-axis and the friction factor, f, as curves corresponding to different values of the relative roughness, e/D. The following SCILAB script is used to plot a simplified version of the Moody diagram for selected values of the relative roughness. The script is called *PlotMoody* and it is stored in SCILAB's working directory:

```
//Script to plot Moody diagram

//First we define the function for the Coolebrook-
//White equation:

deff('[P]=CW(f)','...
P=1/sqrt(f)+0.8686*log(e/(3.7*D)+2.51/(Re*sqrt(f)))')

D = 0.025; //Diameter of 2.5 cm, approx. 1.0 inch

//The next two vectors contain values of the relative
//roughness and the Reynolds number
e_v = [0.01 0.001 0.0001 0.00001 0.000001 0.0000001];
Re_v = [1e4 1e5 1e6 1e7 1e8];

[ne me] = size(e_v);    //size of vectors
[nR mR] = size(Re_v);   //e_v and Re_v

fMoody = zeros(mR,me);  //Create matrix for f values

//Calculating friction factors for combinations of values
//of e_v(i) and Re_v(j):
for j = 1:me
    for i = 1:mR
        Re = Re_v(i); e = D*e_v(j);
       if e < 1e-5 then
           f0 = 0.01;
        else
           f0 = 0.02;
        end;
        fMoody(i,j) = fsolve(f0,CW);
    end;
end;

//Plotting the Moody diagram
plot2d1('oll',Re_v',[fMoody(:,1),fMoody(:,2),...
fMoody(:,3),fMoody(:,4), fMoody(:,5), fMoody(:,6)],...
[1:6],'121','0.01@0.001@0.0001@0.00001@0.000001@0.0000001');

xtitle('Moody diagram','Reynolds number','friction factor');

//end script
```

To execute the script use:

```
-->exec('PlotMoody')
```

SCILAB produces a listing of the script before producing the following graph:



Example 3. Determining pipe diameter given the flow velocity or discharge

Eliminating the friction factor from the Darcy-Weisbach and Coolebrook-White equation we get the following equation relating velocity (V) to pipe length (L), diameter (D), and absolute roughness (e), energy loss (h_f), and kinematic viscosity(ν):

$$V = -1.2283 \cdot \sqrt{\frac{g \cdot h_f \cdot D}{L}} \cdot \ln\left(\frac{e}{3.7 \cdot D} + \frac{1.77 \cdot \nu}{D^{3/2} \cdot \sqrt{\frac{g \cdot h_f}{L}}} \right)$$

Suppose that we are given the values g = 9.806 m/s^2, h_f = 1.0 m, e = 0.0001 m, L = 2000 m, ν = 1x10^{-5} m^2/s, V = 1.1 m/s, and asked to determine the value of D. We can define the following SCILAB function:

```
-->deff('[P]=VCW(D)',...
-->'P=V+1.2283*sqrt(g*D*hf/L)*log(e/(3.7*D)+1.77*Nu/(D^1.5*sqrt(g*hf/L)))')
```

The problem data is entered next:

```
-->V=1.1;g=9.806;hf=1;L=2000;e=0.0001;Nu=1e-5;
```

The solution is found by using the following call to SCILAB function *fsolve* with an initial guess D = 0.5:

```
-->D = fsolve(0.5,VCW)
```



```
 D  = 1.9543027
```

Using continuity, i.e., Q = AV = $\pi D^2 V/4$, an equation can be written in terms of Q as,

$$Q = -0.9648 \cdot D^{5/2} \cdot \sqrt{\frac{g \cdot h_f}{L}} \cdot \ln\left(\frac{e}{3.7 \cdot D} + \frac{1.78 \cdot \nu}{D^{3/2} \cdot \sqrt{\frac{g \cdot h_f}{L}}} \right)$$

For design problems, this equation is often more useful than the previous one given in terms of the velocity because more often than not the discharge rather than the velocity is given. To solve for the diameter given the discharge, we present the following SCILAB example. Given g = 9.806 m/s^2, h_f = 1.0 m, e = 0.0001 m, L = 2000 m, ν = 1x10^{-5} m^2/s, Q = 2.2 m^3/s, find the corresponding pipe diameter. We start by defining the function:

```
-->deff('[P]=QCW(D)',...
--
>'P=Q+0.9648*D^2.5*sqrt(g*hf/L)*log(e/(3.7*D)+1.78*Nu/(D^1.5*sqrt(g*hf/L)))');
```

Next, we enter the problem data:

```
-->Q=2.2;g=9.806;hf=1;L=2000;e=0.0001;Nu=1e-5;
```

Finally, we solve for D:

```
-->D = fsolve(0.5,QCW)
 D  =  1.6782603
```

Solving pipe flow with the Swamee-Jain equation

To avoid the implicit nature of *f* in the Coolebrook-White equation we can use the following explicit approximation for *f*, referred to as the Swamee-Jain equation:

$$f = \frac{1.3254}{\left[\ln\left(\frac{e}{3.75D} + \frac{5.74}{\mathrm{Re}^{0.9}} \right) \right]^2}.$$

With the definition of the Reynolds number *Re = VD/ν,* the resulting expression for the friction factor is

$$f = \frac{1.3254}{\left[\ln\left(\frac{e}{3.75D} + \frac{5.74\nu^{0.9}}{V^{0.9} D^{0.9}} \right) \right]^2}.$$

In design problems it is more convenient to work with the discharge Q. Replacing the velocity *V = 4Q/(πD^2)* into the Swamee-Jain equation we get

$$f = \frac{1.3254}{\left[\ln\left(\frac{e}{3.75 \cdot D} + 4.618 \cdot \left(\frac{D \cdot \nu}{Q}\right)^{0.9}\right)\right]^2}.$$

If we write the Darcy-Weisbach equation in terms of the discharge, i.e.,

$$h_f = f \cdot (L/D) \cdot V^2/(2g) = 8fLQ^2/(\pi^2 g D^5),$$

and then introduce the latest version of the Swamee-Jain equation, we will get the following equation for the friction losses in a pipe:

$$h_f = \frac{1.074 \cdot L \cdot Q^2}{g \cdot D^5 \left[\ln\left(\frac{e}{3.75 \cdot D} + 4.618 \cdot \left(\frac{D \cdot \nu}{Q}\right)^{0.9}\right)\right]^2}.$$

A SCILAB function to solve the Darcy-Weisbach equation with the Swamee-Jain equation

The following function, *DWSJ* (*D*arcy-*W*eisbach equation with *S*wamee-*J*ain friction factor) can be used to solve for any of the variables in the previous equation given appropriate values of the remaining variables. The function also requires that the user provide an initial value of the variable that he or she is solving for. Function *DWSJ* uses SCILAB function *fsolve* to solve for the particular unknown of interest. A listing of the function follows:

```
function [result] = DWSJ(index,gindex,L,D,e,nu,hf,Q)

//This function solves for one of the variables
//in the Darcy-Weisbach equation with the friction
//factor approximated through the Swamee-Jain
//equation.   The string variable 'index' determines
//which variable to solve for.  Possible values of
//index are:
//      'L'   - to solve for the length of the pipe
//      'D'   - to solve for the diameter
//      'e'   - to solve for the absolute wall roughness
//      'nu'  - to solve for the kinematic viscosity
//      'hf'  - to solve for the friction losses
//      'Q'   - to solve for the discharge
//The variable 'gindex' can take the values 'SI' or
//'ES' corresponding to the system of units to be used:
//      'SI'  - for the Systeme International
//      'ES'  - for the English (or Imperial) System
//Make sure that the values of the variables are given
//in consistent units, i.e.,
// L(m or ft), D(m or ft), e(m or ft),
```



```
// nu(m^2/s or ft^2/s), hf(m or ft), Q(m^3/s or ft^3/s)
// The order of the variables does not change in the call
// to the function.  The user needs to provide an initial
// guess for the variable he/she is solving for in the
// appropriate position in the function call.

if gindex == 'SI' then
       g = 9.806;
elseif gindex == 'ES' then
       g = 32.2;
else
       error('DWSJ - wrong index for unit system');
       abort;
end;

if index == 'L' then
       deff('[P] = DWSJEq(LL)',...
         'P=1.074*LL*Q^2/(g*D^5*(log(e/(3.75*D)+4.618*(D*nu/Q)^0.9))^2)-hf');
        result = fsolve(L,DWSJEq);
elseif index == 'D' then
       deff('[P] = DWSJEq(DD)',...
       'P=1.074*L*Q^2/(g*DD^5*(log(e/(3.75*DD)+4.618*(DD*nu/Q)^0.9))^2)-hf');
        result = fsolve(D,DWSJEq);
elseif index == 'e' then
       deff('[P] = DWSJEq(ee)',...
       'P=1.074*L*Q^2/(g*D^5*(log(ee/(3.75*D)+4.618*(D*nu/Q)^0.9))^2)-hf');
        result = fsolve(e,DWSJEq);
elseif index == 'nu' then
       deff('[P] = DWSJEq(nnu)',...
       'P=1.074*L*Q^2/(g*D^5*(log(e/(3.75*D)+4.618*(D*nnu/Q)^0.9))^2)-hf');
        result = fsolve(nu,DWSJEq);
elseif index == 'hf' then
       deff('[P] = DWSJEq(hhf)',...
       'P=1.074*L*Q^2/(g*D^5*(log(e/(3.75*D)+4.618*(D*nu/Q)^0.9))^2)-hhf');
        result = fsolve(L,DWSJEq);
elseif index == 'Q' then
       deff('[P] = DWSJEq(QQ)',...
       'P=1.074*L*QQ^2/(g*D^5*(log(e/(3.75*D)+4.618*(D*nu/QQ)^0.9))^2)-hf');
        result = fsolve(L,DWSJEq);
else
       error('DWSJ - index is L, D, e, nu, hf, or Q enclosed in quotes');
       abort;
end;
```

The function may provide an error message if the initial value for the unknown variable is too far away from the solution, particularly for those unknowns whose values are typically very small, such as the viscosity (of the order of 10^{-6} to 10^{-5}) or the absolute roughness (anywhere from 0.000001to 0.1). Whenever you receive an error message in a solution, try using different initial values of the unknown to improve the convergence of the function.

Applications of function *DWSJ* to pipe flow

Examples of applications of the function *DWSJ* follow. Recall that the general call to the function is

[result] = DWSJ(index,gindex,L,D,e,nu,hf,Q)

With *index* determining the unknown to solve for *('L','D','e','nu','hf','Q')*, and *gindex* determining the system of units *('SI', 'ES')*.

```
-->getf('DWSJ')                              //Load the function

-->DWSJ('L','SI',100,0.5,0.0001,1e-5,2,0.5)  //Solve for pipe length
 ans  =

    165.70807

-->DWSJ('D','SI',100,0.5,0.0001,1e-5,2,0.5)  //Solve for pipe diameter
 ans  =

     .4511914

-->DWSJ('e','SI',100,0.5,0.0001,1e-5,2,0.5)  //Solve for pipe roughness
 ans  =

     .0022291

-->DWSJ('nu','SI',100,0.5,0.0001,1e-5,2,0.5)  //Solve for kinematic viscosity
 ans  =

     .0001117

-->DWSJ('hf','SI',100,0.5,0.0001,1e-5,2,0.5)  //Solve for friction loss
 ans  =

    1.2069418

-->DWSJ('Q','SI',100,0.5,0.0001,1e-5,2,0.5)  //Solve for discharge
 ans  =

     .6567565
```

Solving for discharge and head for a pipe-pump system

Consider the flow through a horizontal pipe between two reservoirs as shown in the figure below. A pump is necessary to overcome the difference in water surface levels between the two reservoirs, $\Delta h = z_2 - z_1$. The pump inserts an energy head H into the system. If we neglect minor losses (entrance into the pipe, exit out of the pipe, valves, etc.), the energy equation for this system is given by

$$z_1 + 0 + 0 + H = z_2 + 0 + 0 + h_f.$$

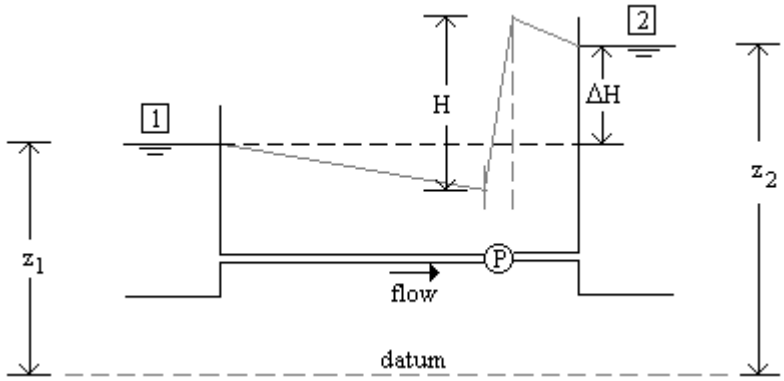
Using the equation presented earlier for h_f that combines the Darcy-Weisbach and Swamee-Jain equations, we can write the so-called *system equation*:

$$H = \Delta h + \frac{1.074 \cdot L \cdot Q^2}{g \cdot D^5 \left[\ln\left(\frac{e}{3.75 \cdot D} + 4.618 \cdot \left(\frac{D \cdot \nu}{Q} \right)^{0.9} \right) \right]^2}.$$

The discharge through the pump *Q* and the energy head *H* that the pump provides to the system are related by the *pump rating curve*, typically described by a quadratic equation, i.e.,

$$H = a + bQ + cQ^2.$$

For a given system, the pipe characteristics, i.e., length(L), diameter (D), and roughness (e), as well as the kinematic viscosity of the liquid (ν) are known. For a given value of Δh, the two equations listed above are solved for the unknowns H and Q. The solution can be accomplished through graphical means by plotting the two equations in the same set of axis and determining their point of intersection. This approach is shown next, using SCILAB.



Graphical solution to the pump-pipeline system

To obtain the graphical solution we define the following functions that represent, respectively, the system equation and the pump rating curve:

```
-->deff('[HH]=H1(Q)',...
-->'HH = Dh+1.074*L*Q^2/(g*D^5*log(e/(3.75*D)+4.618*(D*nu/Q)^0.9)^2)')

-->deff('[HH]=H2(Q)','HH = a+b*Q+c*Q^2')
```

Next, we enter the constant values:

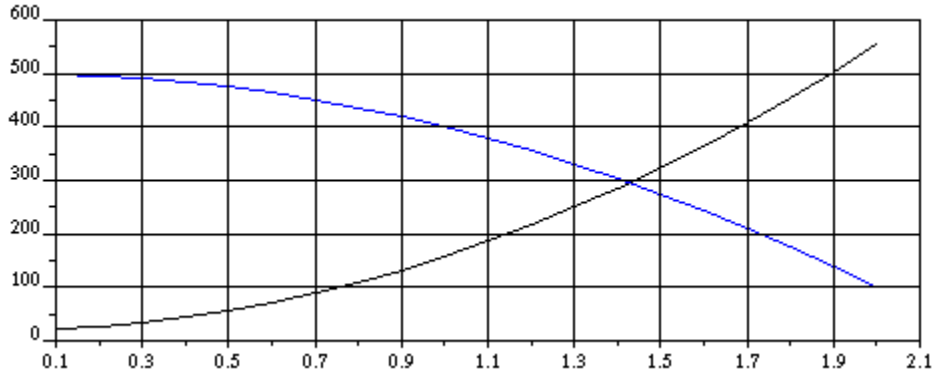
```
-->Qp = [0.1:0.1:2.0]; Hp1 = feval(Qp,H1); Hp2 = feval(Qp,H2);
```

The next step is to define a vector of values of Q, which we call Qp, and evaluate the two functions for those values of Q using function *feval* (*f*unction *eval*uation). Thus, vectors Hp1 and Hp2 store, respectively, the values of H corresponding to the system equation and the pump rating curve.


```
-->Dh = 20; L = 150; g = 9.806; e = 0.00001;

-->D = 0.25; nu = 1e-6; Q = 0.20;

-->a = 500; b = 0; c = -100;
```

The plot is obtained by using:

```
->plot2d([Qp',Qp'],[Hp1',Hp2'],[1,2])

-->xtitle('Pump-pipeline system','Q(m^3/s)','H(m)')
```



We can estimate the solution from the graphics as Q = 1.42 m^3/s and H = 290 m.

A function to solve a pipe-pump system

The following SCILAB function, *PipePump*, is used to code the functions

$$f_1(Q,H) = H - \Delta h - \frac{1.074 \cdot L \cdot Q^2}{g \cdot D^5 \left[\ln\left(\frac{e}{3.75 \cdot D} + 4.618 \cdot \left(\frac{D \cdot \nu}{Q} \right)^{0.9} \right) \right]^2},$$

$$f_2(Q,H) = H - a - b \cdot Q - c \cdot Q^2.$$

This is the listing of the function:

```
function [P] = PipePump(X)

//This function codes the system equation
//and the pump rating curve for a pipe-pump
//system representing pumping between two
//reservoirs.  X(1) = Q and X(2) = H.
```



```
P = zeros(2,1);

P(1)=X(2)-Dh-1.074*L*X(1)^2/(g*D^5*(log(e/(3.75*D)+4.618*(D*nu/X(1))^0.9))^2);

P(2)=X(2)-a-b*X(1)-c*X(1)^2;

//end function
```

To load the function we use:

```
-->getf('PipePump')
```

The following call to function *fsolve* produces the solution:

```
-->fsolve([1.4;300],PipePump)
 ans  =

!   1.4275835 !
!   296.20053 !
```

To verify the solution use:

```
-->PipePump(ans)
 ans  =

   1.0E-11 *

!    .5798029 !
! -  .4490630 !
```

Notice that the results found through the use of *fsolve* and *PipePump*, i.e., $Q = 1.4275835\ m^3/s$ and $H = 296.20053\ m$, are very close to the estimates we obtained from the graphical solution.

Exercises

In problems [1] through [6], use the following definitions:

$$z_1 = -3 + 2i,\ z_2 = 5 - i,\ z_3 = -4+3i,\ z_4 = -2-4i$$

[1]. Determine the following magnitudes and arguments:

(a) $|z_1|$ (b) $\text{Arg}(z_1)$ (c) $|z_2|$ (d) $\text{Arg}(z_2)$
(e) $|z_3|$ (f) $\text{Arg}(z_3)$ (g) $|z_4|$ (h) $\text{Arg}(z_4)$

[2]. Write the following complex numbers in polar form:

(a) 3-5i (b) 4-4i (c) 3-5i (d) -2-6i
(e) -5+6i (f) $-\pi+3i$ (g) (5-i)/2 (h) 7+5i

[3]. N/A.

[4]. Determine the result of the following complex number operations:

(a) $z_1+3\cdot z_2$ (b) $z_1-z_2+4\cdot z_3$ (c) $(z_1-2)\cdot(z_4-z_3)$ (d) $z_4/z_2 + z_3/z_1$
(e) $z_1\cdot z_2\cdot z_3$ (f) $(z_1-2z_2)\cdot(z_3/\pi)$ (g) $z_1\cdot z_2 -1/z_3$ (h) $(z_2+z_3)/(z_1+3\cdot z_2)$

[5]. Determine the result of the following complex number operations:

(a) $\bar{z}_1+z_1$ (b) $\bar{z}_2\cdot z_2$ (c) $z_3/\bar{z}_3$ (d) $(\bar{z}_1+\bar{z}_2)/(z_3-5\,\bar{z}_4)$
(e) $\bar{z}_1-z_1$ (f) $2(\bar{z}_1-z_1)$ (g) $1/\bar{z}_2+1/z_2$ (h) $|z_1|(\bar{z}_2+z_3)$

[6]. Determine the result of the following complex number operations:

(a) z_3^3 (b) $z_2(z_1-2z_3)^2$ (c) z_2/z_3^4 (d) z_3^2/z_1^3
(e) $z_1+1/z_2+1/z_3^2$ (f) $(1+1/z_2)^3$ (g) $z_1^3+(z_2-z_3)^2$ (h) $(z_2+1)^2/z_3$

[7]. Solve for z in the following equations:

(a) $z^2+3-2i = 0$ (b) $(z+1)^2 = 3^{1/2}$ (c) $z^3-i = 0$ (d) $z^4+(i-2)^3 = 0$
(e) $z^2+2z=4$ (f) $z^3=-1$ (g) $1/(z+1)^3 = i$ (h) $z(z-1)=2$

[8]. Write a SCILAB function to calculate the roots of the quadratic equation: $ax^2+bx+c = 0$. Use the function to solve the quadratic equations whose coefficients are:

(a) $a = 2,\ b = -5,\ c = 3$ (b) $a = 12,\ b = 22,\ c = -3$
(c) $a = 10,\ b = 25,\ c = 33$ (d) $a = -5,\ b = -5,\ c = 2$

[9]. Write a SCILAB function to calculate the roots of the cubic equation: $ax^3+bx^2+cx+d=0$. (a)

(a) $a = 2,\ b = -5,\ c = 3,\ d = 23$ (b) $a = 12,\ b = 22,\ c = -3,\ d = 1$
(c) $a = 10,\ b = 25,\ c = 33,\ d = -10$ (d) $a = -5,\ b = -5,\ c = 2,\ d = 10$

[10]. Use function *roots* to solve problems [8] and [9].

[11]. Use function *roots* to solve the following polynomial equations:

(a) $s^5 + 6s^3 - 27 = 0$ (b) $y^4 - 23y^3 + 5y^2 - 3y + 2 = 0$
(c) $r^6 + r^2 + r - 2345 = 0$ (d) $(s+1)^7 - s^2 + 2 = 0$

In problems [12] through [16], you are required to solve the equation $f(x) = 0$. Plot the corresponding function $y = f(x)$ to obtain guesses of the solution(s), then solve for as many solutions as possible using:

(a) the bisection method
(b) the Newton-Raphson method
(c) the secant method
(d) SCILAB function *fsolve*

[12] $f(x) = x^3 + 18x^2 - 22x + \sin(2x^2+x) - 255$

[13] $f(x) = x^4 + \exp(x-2) + x^3 - 200$

[14] $f(x) = \exp(-0.1x)\cos(2x-\pi/2)$

[15] $f(x) = x^5 + 5x^2 - 23x - 19$

[16] $f(x) = \ln((x^2+x+2)/(\sin x + 1) - 5$, for $x>0$

In problems [17] through [21], solve the system of equations shown using (a) SCILAB function *fsolve*, and (b) the "Secant" method for multiple equations

[17] $x^2 + y^2 - 2xy = 1,\ (x-2)(y-3) + e^x + y = 10.39$

[18] $x^2 + y^2 + z^2 = 17,\ xy + yz + zx = -4,\ (x+1)(y+1)(z-3) + xyz = -8$

[19] $\tan(x-1/2) + 5\cos y + z = 0.75,\ xyz + \sin(y) = 4.05,\ \exp(-xyz) + \cos(z) = 0.58$

[20] $ab + bc^2 + abc = 26,\ \sin(a+b) + \exp(c) - \ln(a) = -2.05,\ a + b + c = 0$

[21] $x\ln(y+z) = -3.83,\ xy^2 + yz^2 + zx^2 = -27.20,\ (x+y)^{1/2} + x = 0.60$

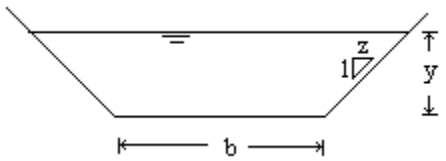
[22]. Manning's equation is used to calculate the discharge on an open channel of slope S_0 (typically a relatively small value, i.e., $0.00000001 < S_0 < 0.001$), whose surface roughness is characterized by a parameter known as the Manning's coefficient, n (typical values between 0.001 and 0.3. The larger the value of n, the rougher the surface. For example, concrete surfaces have $n = 0.012$). The equation is written as

$$Q = \frac{C_u}{n} \cdot \frac{A^{5/3}}{P^{2/3}} \cdot \sqrt{S_0},$$

where C_u is a coefficient that depends on the system of units used, with $C_u = 1.0$ for the Systeme Internationale (S.I.) and $C_u = 1.486$ for the English (or Imperial) System of units (E.S.), A is the cross-sectional area, P is the wetted perimeter of the cross-section (i.e., the length of the cross-sectional boundary in contact with the water), and Q is the discharge.
For a symmetric trapezoidal cross section, as shown below, the area and wetted perimeter are given by

$$A = (b + zy)y, \quad P = b + 2y\sqrt{1+z^2},$$

where *b* is the bottom width of the cross-section, *y* is the cross-sectional depth, and *z* is the (dimensionless) side slope.



Write a function, along the lines of function DWSJ, that allows the user to select which element to solve for out of the Manning's equation for a trapezoidal cross-section. Use the function thus developed to solve for the missing terms in each line of the following table:

Case	System of Units	b	y	z	n	So	Q
(a)	S.I.	***b=?***	*0.6*	*1.5*	*0.012*	*0.0001*	*0.2*
(b)	S.I.	*1.5*	***y=?***	*0.5*	*0.023*	*0.00001*	*0.15*
(c)	S.I.	*0.5*	*0.25*	***z=?***	*0.01*	*0.001*	*0.35*
(d)	S.I.	*1.4*	*0.4*	*1*	***n=?***	*0.0001*	*0.6*
(e)	S.I.	*1.2*	*0.6*	*0.5*	*0.018*	***S=?***	*0.8*
(f)	S.I.	*0.6*	*0.3*	*0.75*	*0.015*	*0.0001*	*Q=?*
(g)	E.S.	***b=?***	*0.6*	*1.5*	*0.012*	*0.0001*	*3.5*
(h)	E.S.	*3*	***y=?***	*0.5*	*0.023*	*0.00001*	*7.2*
(i)	E.S.	*2*	*1.2*	***z=?***	*0.01*	*0.001*	*4.5*
(j)	E.S.	*3.5*	*0.75*	*1*	***n=?***	*0.0001*	*10*
(k)	E.S.	*4.25*	*2.1*	*0.5*	*0.018*	***S=?***	*25*
(l)	E.S.	*5*	*2.5*	*0.75*	*0.015*	*0.0001*	***Q=?***

[23]. Critical flow conditions in open channel flow are given by the equation

$$\frac{Q^2 T}{gA^3} = 1,$$

where *Q* is the flow discharge, *T* is the top width of the cross-section (for a trapezoidal channel, $T = b+2zy$), g is the acceleration of gravity ($g = 9.806\ m/s^2$ in the S.I. and $g = 32.2\ ft/s^2$ in the E.S.), and A is the cross-sectional area. Use SCILAB function *fsolve* to obtain the critical depth, y_c, (i.e., the water depth at critical conditions) for the data shown in the following table:

Case	System of units	b	z	*Q*
(a)	S.I.	*0.6*	*1.5*	*0.2*
(b)	S.I.	*1.5*	*0.5*	*0.15*
(c)	S.I.	*0.5*	*1*	*0.35*
(d)	E.S.	*1.4*	*1*	*0.6*
(e)	E.S.	*1.2*	*0.5*	*0.8*
(f)	E.S.	*0.6*	*0.75*	*1*

[24]. The conditions of open channel flow at the entrance from a reservoir are determined by the simultaneous solution of the energy equation and Manning's equation. The energy equation, for an available head of H_o at the reservoir, is written as

$$H_0 = y + \frac{Q^2}{2g[A(y)]^2},$$

while the Manning's equation is written as

$$Q = \frac{C_u}{n} \cdot \frac{[A(y)]^{5/3}}{[P(y)]^{2/3}} \cdot \sqrt{S_0},$$

where C_u is a coefficient that depends on the system of units used, with $C_u = 1.0$ for the Systeme Internationale (S.I.) and $C_u = 1.486$ for the English (or Imperial) System of units (E.S.), n is the Manning's coefficient (typically, between 0.001 and 0.3), $A(y)$ is the cross-sectional area, $P(y)$ is the wetted perimeter of the cross-section (i.e., the length of the cross-sectional boundary in contact with the water), and Q is the discharge.

Typically the values of n, g, C_u, S_0, and the geometry of the cross-section are known. The simultaneous solution of these two equations produces as a result the values of the water depth, y, and the flow discharge, Q.

For a trapezoidal cross-section of bottom width b and side slope z, the area and wetted perimeter are given by

$$A(y) = (b + zy)y, \quad P(y) = b + 2y\sqrt{1 + z^2}\ .$$

Use SCILAB function *fsolve* to obtain the simultaneous solution of the energy and Manning's equation at the entrance from a reservoir into an open channel using the following data values:

Case	System of units	*Ho*	b	z	n	So
(a)	S.I.	*4.5*	*1.2*	*1.5*	*0.012*	*0.0001*
(b)	S.I.	*3.5*	*0.8*	*0.5*	*0.023*	*0.00001*
(c)	S.I.	*2.5*	*0.4*	*1*	*0.01*	*0.001*
(d)	E.S.	*6*	*5*	*1*	*0.012*	*0.0001*
(e)	E.S.	*3*	*3*	*0.5*	*0.018*	*0.00001*
(f)	E.S.	*10*	*7.5*	*0.75*	*0.015*	*0.0001*

REFERENCES (for all SCILAB documents at InfoClearinghouse.com)

Abramowitz, M. and I.A. Stegun (editors), 1965,"*Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*," Dover Publications, Inc., New York.

Arora, J.S., 1985, "*Introduction to Optimum Design*," Class notes, The University of Iowa, Iowa City, Iowa.

Asian Institute of Technology, 1969, "*Hydraulic Laboratory Manual*," AIT - Bangkok, Thailand.

Berge, P., Y. Pomeau, and C. Vidal, 1984,"*Order within chaos - Towards a deterministic approach to turbulence*," John Wiley & Sons, New York.

Bras, R.L. and I. Rodriguez-Iturbe, 1985,"*Random Functions and Hydrology*," Addison-Wesley Publishing Company, Reading, Massachussetts.

Brogan, W.L., 1974,"*Modern Control Theory*," QPI series, Quantum Publisher Incorporated, New York.

Browne, M., 1999, "*Schaum's Outline of Theory and Problems of Physics for Engineering and Science*," Schaum's outlines, McGraw-Hill, New York.

Farlow, Stanley J., 1982, "*Partial Differential Equations for Scientists and Engineers*," Dover Publications Inc., New York.

Friedman, B., 1956 (reissued 1990), "*Principles and Techniques of Applied Mathematics*," Dover Publications Inc., New York.

Gomez, C. (editor), 1999, "*Engineering and Scientific Computing with Scilab*," Birkhäuser, Boston.

Gullberg, J., 1997, "*Mathematics - From the Birth of Numbers*," W. W. Norton & Company, New York.

Harman, T.L., J. Dabney, and N. Richert, 2000, "*Advanced Engineering Mathematics with MATLAB® - Second edition*," Brooks/Cole - Thompson Learning, Australia.

Harris, J.W., and H. Stocker, 1998, "*Handbook of Mathematics and Computational Science*," Springer, New York.

Hsu, H.P., 1984, "*Applied Fourier Analysis*," Harcourt Brace Jovanovich College Outline Series, Harcourt Brace Jovanovich, Publishers, San Diego.

Journel, A.G., 1989, "*Fundamentals of Geostatistics in Five Lessons*," Short Course Presented at the 28th International Geological Congress, Washington, D.C., American Geophysical Union, Washington, D.C.

Julien, P.Y., 1998,"*Erosion and Sedimentation*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Keener, J.P., 1988, "*Principles of Applied Mathematics - Transformation and Approximation*," Addison-Wesley Publishing Company, Redwood City, California.

Kitanidis, P.K., 1997,"*Introduction to Geostatistics - Applications in Hydogeology*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Koch, G.S., Jr., and R. F. Link, 1971, "*Statistical Analysis of Geological Data - Volumes I and II*," Dover Publications, Inc., New York.

Korn, G.A. and T.M. Korn, 1968, "*Mathematical Handbook for Scientists and Engineers*," Dover Publications, Inc., New York.

Kottegoda, N. T., and R. Rosso, 1997, "*Probability, Statistics, and Reliability for Civil and Environmental Engineers*," The Mc-Graw Hill Companies, Inc., New York.

Kreysig, E., 1983, "*Advanced Engineering Mathematics - Fifth Edition*," John Wiley & Sons, New York.

Lindfield, G. and J. Penny, 2000, "*Numerical Methods Using Matlab®,"* Prentice Hall, Upper Saddle River, New Jersey.

Magrab, E.B., S. Azarm, B. Balachandran, J. Duncan, K. Herold, and G. Walsh, 2000, "*An Engineer's Guide to MATLAB®*", Prentice Hall, Upper Saddle River, N.J., U.S.A.

McCuen, R.H., 1989,"*Hydrologic Analysis and Design - second edition*," Prentice Hall, Upper Saddle River, New Jersey.

Middleton, G.V., 2000, "*Data Analysis in the Earth Sciences Using Matlab®,"* Prentice Hall, Upper Saddle River, New Jersey.

Montgomery, D.C., G.C. Runger, and N.F. Hubele, 1998, "*Engineering Statistics*," John Wiley & Sons, Inc.

Newland, D.E., 1993, "*An Introduction to Random Vibrations, Spectral & Wavelet Analysis - Third Edition*," Longman Scientific and Technical, New York.

Nicols, G., 1995, "*Introduction to Nonlinear Science*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Parker, T.S. and L.O. Chua, , "*Practical Numerical Algorithms for Chaotic Systems*," 1989, Springer-Verlag, New York.

Peitgen, H-O. and D. Saupe (editors), 1988, "*The Science of Fractal Images*," Springer-Verlag, New York.

Peitgen, H-O., H. Jürgens, and D. Saupe, 1992, "*Chaos and Fractals - New Frontiers of Science*," Springer-Verlag, New York.

Press, W.H., B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, 1989, "*Numerical Recipes - The Art of Scientific Computing (FORTRAN version)*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Raghunath, H.M., 1985, "*Hydrology - Principles, Analysis and Design*," Wiley Eastern Limited, New Delhi, India.

Recktenwald, G., 2000, "*Numerical Methods with Matlab - Implementation and Application*," Prentice Hall, Upper Saddle River, N.J., U.S.A.

Rothenberg, R.I., 1991, "*Probability and Statistics*," Harcourt Brace Jovanovich College Outline Series, Harcourt Brace Jovanovich, Publishers, San Diego, CA.

Sagan, H., 1961,"*Boundary and Eigenvalue Problems in Mathematical Physics*," Dover Publications, Inc., New York.

Spanos, A., 1999,"*Probability Theory and Statistical Inference - Econometric Modeling with Observational Data*," Cambridge University Press, Cambridge CB2 2RU, U.K.

Spiegel, M. R., 1971 (second printing, 1999), "*Schaum's Outline of Theory and Problems of Advanced Mathematics for Engineers and Scientists*," Schaum's Outline Series, McGraw-Hill, New York.

Tanis, E.A., 1987, "*Statistics II - Estimation and Tests of Hypotheses*," Harcourt Brace Jovanovich College Outline Series, Harcourt Brace Jovanovich, Publishers, Fort Worth, TX.

Tinker, M. and R. Lambourne, 2000, "*Further Mathematics for the Physical Sciences*," John Wiley & Sons, LTD., Chichester, U.K.

Tolstov, G.P., 1962, "*Fourier Series*," (Translated from the Russian by R. A. Silverman), Dover Publications, New York.

Tveito, A. and R. Winther, 1998, "*Introduction to Partial Differential Equations - A Computational Approach*," Texts in Applied Mathematics 29, Springer, New York.

Urroz, G., 2000, "*Science and Engineering Mathematics with the HP 49 G - Volumes I & II*", www.greatunpublished.com, Charleston, S.C.

Urroz, G., 2001, "*Applied Engineering Mathematics with Maple*", www.greatunpublished.com, Charleston, S.C.

Winnick, J., , "*Chemical Engineering Thermodynamics - An Introduction to Thermodynamics for Undergraduate Engineering Students*," John Wiley & Sons, Inc., New York.

