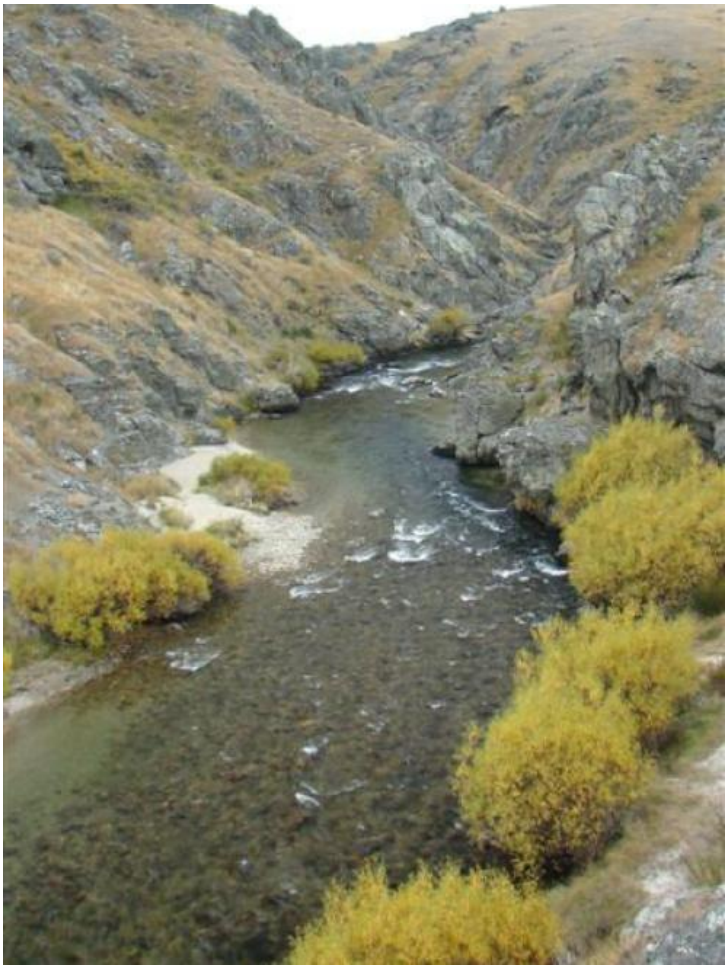
CE 5362 Surface Water Modeling
Lesson 6

by
Theodore G. Cleveland
Department of Civil, Environmental, and Construction Engineering
Texas Tech University



Contents

1 Steady Water Surface Profiles

1.1 Steady Uniform Flow

In the previous chapter uniform flow was defines at the situation where friction slope and channel slope are the same. In this kind of flow the profile grade line (channel bottom), the hydraulic grade line (water surface), and the energy grade line are all parallel. The following would be implied

$$S_f = \frac{\Delta h_L}{\Delta x} = S_0 \tag{1}$$

Suppose we apply the Darcy-Weisbach head loss model (and adapt it for non-circular conduit)

$$h_L = f \frac{L}{4R_h} \frac{V^2}{2g} \tag{2}$$

1.2 Steady Gradually Varied Flow

Need a brief theory

1.2.1 Finite Difference Methods — Fixed Depth Change Step Method

The fixed-step refers to fixed changes in depth for which we solve to find the variable spatial steps. The method is a very simple method for computing water surface profiles in prismatic channels. A prismatic channel is a channel of uniform cross sectional geometry[1] with constant bed (topographic) slope.

In such channels with smooth (non-jump) steady flow the continunity and momentum equations are

$$Q = AV \tag{3}$$

where Q is volumetric discharge, A is cross sectional flow area, and V is the mean section velocity.

and

$$\frac{V}{g}\frac{dV}{dx} + \frac{dh}{dx} = S_o - S_f \tag{4}$$

where h is the flow depth (above the bottom), and x is horizontal the distance along the channel.

[1] Channel geometry is same at any section, thus rectangular, trapezoidal, and circular channels if the characteristic width dimension is constant would be prisimatic.

For the variable step method, the momentum equation is rewritten as a difference equation (after application of calculus to gather terms) then rearranged to solve for the spatial step dimension [2].

$$\frac{\frac{V_{i+1}^2}{2g} - \frac{V_i^2}{2g}}{\Delta x} + \frac{h_{i+1} - h_i}{\Delta x} = S_o - \bar{S}_f \tag{5}$$

where $\bar{S}_f$ is the average slope of the energy grade line between two sections (along a reach of length Δx, the unknown value).

Rearrangement to isolate Δx produces an explicit update equation that can be evaluated to fond the different values of Δx associated with different flow depths. The plot of the accumulated spatial changes versus the sum of the flow depth and bottom elevation is the water surface profile.

$$\frac{(h_{i+1} + \frac{V_{i+1}^2}{2g}) - (h_i + \frac{V_i^2}{2g})}{S_o - \bar{S}_f} = \Delta x \tag{6}$$

The distance between two sections with known discharges is computed using the equation, all the terms on the left hand side are known values. The mean energy gradient ($\bar{S}_f$) is computed from the mean of the velocity, depth, area, and hydraulic radius for the two sections.

The friction slope can be computed using Manning's, Chezy, or the Darcy-Weisbach friction equations adapted for non-circular, free-surface conduits.

1.2.2 Coding the algorithm in R

Here the method is illustrated in **R** to illustrate the tool as a programming environment[3].

First we build a set of utility functions, these will be used later in the `backwater` function:

Listing 1 are utility functions for rectangular channels for flow area given channel depth and width for rectangular and wetted perimeter given depth and width. Different geometries will need different functions (probably by numerical methods rather than actual functional relationships).

[2] The equation here is written moving upstream, direction matters for indexing. Thus position $i+1$ is assumed upstream of position i in this essay. This directional convention is not generally true in numerical methods and analysts need to use care when developing their own tools or using other tools. A clever analyst need not rewrite code, but simple interchange of upstream and downstream depths will handle both backwater and front-water curves.

[3] The **R** code in this essay is broken into parts and some unnecessary line feeds are included to fit the page.

Listing 1. R Code for prototype hydraulic support functions (for rectangular channels)

```
# Depth-Area Function
area<-function(depth,width){
        area<-depth*width;
        return(area)
        }
# Wetted perimeter function for rectangular channel
perimeter<-function(depth,width){
        perimeter<-2*depth+width;
        return(perimeter)
        }
# 2-point average - useful with friction slopes
avg2point<-function(x1,x2){
        avg2point<-0.5*(x1+x2);
        return(avg2point)
        }
```

Listing 2 is a listing of the code for the hydraulic radius (ratio of the above results), this is a generic function, it does not need to know the flow geometry

Listing 2. R Code for prototype hydraulic radius function

```
# Hydraulic radius function
radius<-function(area,perimeter){
        radius<-area/perimeter;
        return(radius)
        }
```

Listing 3 is a listing of code for the friction slope given Manning's n, discharge, hydraulic radius, and flow area. Notice that this function implicitly assumes SI units (the 1.49 constant in U.S. Customary units is not present). For U.S. Customary units either add the constant or convert the US units into equivalent SI units.

Listing 3. R Code for prototype friction slope function)

```
# Friction slope function
slope_f<-function(discharge,mannings_n,area,radius){
        slope_f<-(discharge^2)*(mannings_n^2)/( (radius^(4/3))*(area^2) );
        return(slope_f)
        }
```

The semi-colons in the functions are probably unnecessary, but have value because it forces the expression to its left to be evaluated and helps prevent ambiguous[4] code. Also notice the use of the scope { } delimiter, the delimiter is required, however there is no requirement to line feed — I simply find it easier to read my own code in this fashion (and count delimiters).

At this point, we would have 5 useful, testable functions (and we should test before the next step.

Listing 4 is the step-backwater method implemented as a function. This function computes the space steps, changes in depth, etc. as per the algorithm. The function is a `FORTRAN` port, so it is not a terribly efficient use of **R**, but it illustrates count controlled repetition (for loops), array indexing, and use of the utility functions to make the code readable as well as ensure that the parts work before the whole program is assembled. This concept is really crucial, if you can build a tool of parts that are known to work, it helps keep logic errors contained to known locations.

[4]To the human operator; the computer will either accept the code or throw an error.

Listing 4. R Code for prototype friction slope function)

.

```
# Backwater curve function
backwater<-function(begin_depth,end_depth,how_many,discharge,width,mannings_n,slope){
#
## Example function call will be shown in class and on movie.
#
# Other functions must exist otherwise will spawn errors
#
# Prepare space for vectors
 depth<-numeric(0)
 bse<-numeric(0)
 wse<-numeric(0)
# change in depth for finding spatial steps
 delta_depth<-(begin_depth-end_depth)/(how_many)
# assign downstream value
 depth[1]<-begin_depth
# depth values to evaluate
 for (i in 2:how_many){depth[i]<-depth[1]-i*delta_depth}
#velocity for each depth
 velocity<-discharge/area(depth,width)
# numeric vector for space steps (destination space)
 deltax<-numeric(0)
# next for loop is very FORTRANesque!
for (i in 1:(how_many-1)){
#compute average depth in reach
    depth_bar<-avg2point(depth[i],depth[i+1]);
#compute average area in reach
    area_bar<-area(depth_bar,width);
#compute average wetted perimeter
    perimeter_bar<-perimeter(depth_bar,width);
 #compute average hydraulic radius
    radius_bar<-radius(area_bar,perimeter_bar);
 #compute friction slope
    friction<-slope_f(discharge,mannings_n,area_bar,radius_bar)
 # compute change in distance for each change in depth
    deltax[i]<-( (depth[i+1]+(velocity[i+1]^2)/(2*9.8)
                    - (depth[i] + (velocity[i]^2)/(2*9.8)) )  /  (slope-friction);
}
# space for computing cumulative distances
 distance<-numeric(0);
 distance[1]<-0;
 bse[1]<-0; # bottom elevation at origin
 for (i in 2:(how_many)){
        distance[i]<-distance[i-1]+deltax[i-1]; # spatial distances
        bse[i]<-bse[i-1]-deltax[i-1]*slope; # bottom elevations
        }
 wse<-bse+depth # water surface elevations
# output
 z<-cbind(distance,depth,bse,wse) # bind output into 4 columns
return(z)
#
}
```

1.2.3 Example 1 — Backwater curve

Figure 1 is a backwater curve[5] for a rectangular channelwith discharge over a weir (on the right hand side — not depicted). The channel width is 5 meters, bottom slope 0.001, Manning's $n = 0.02$ and discharge $Q = 55.4\frac{m^3}{sec}$.

Using the `backwater` function and some plot calls in **R** we can duplicate the figure (assuming the figure is essentially correct).

[5]Page 85. Koutitas, C.G. (1983). Elements of Computational Hydraulics. Pentech Press, London 138p. ISBN 0-7273-0503-4

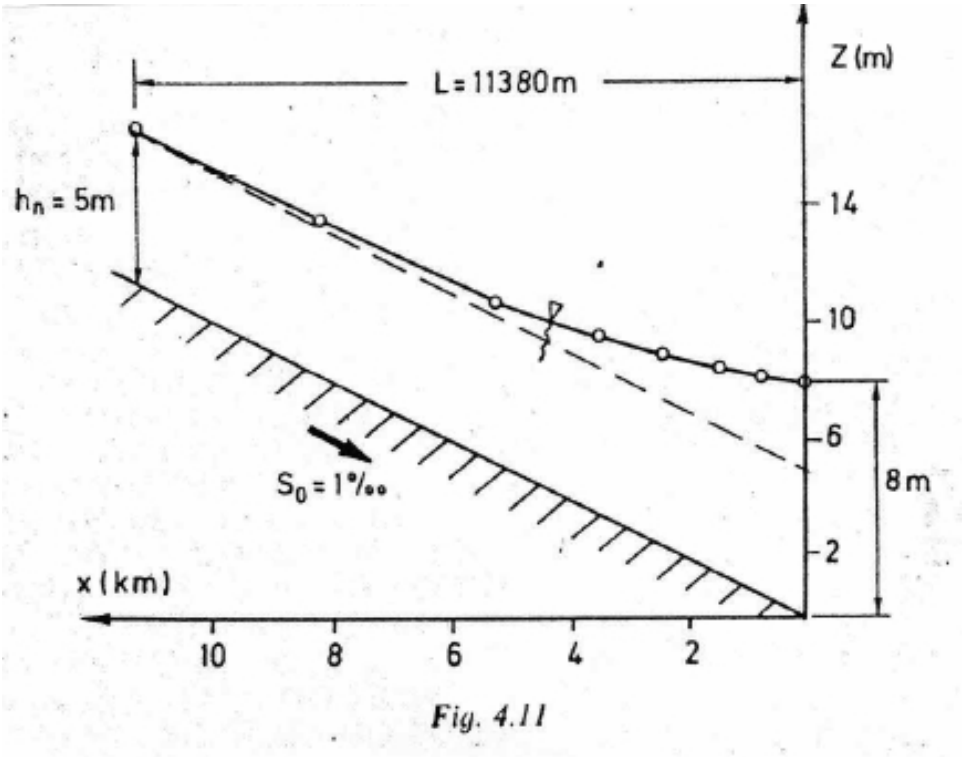


Figure 1. Example backwater curve.

Listing 5. R Console Output when script is run

```
> # here is the function call
> backwater(begin_depth=8,end_depth=5,how_many=31,
    + discharge=55.4,width=5,mannings_n=0.02,slope=0.001)
         distance    depth         bse        wse
 [1,]      0.0000 8.000000  0.0000000  8.000000
 [2,]   -283.3504 7.806452  0.2833504  8.089802
 [3,]   -428.0112 7.709677  0.4280112  8.137689
 [4,]   -574.8516 7.612903  0.5748516  8.187755
 [5,]   -724.0433 7.516129  0.7240433  8.240172
 [6,]   -875.7785 7.419355  0.8757785  8.295133
... Many Rows ...
[29,]  -7186.0943 5.193548  7.1860943 12.379643
[30,]  -8389.5396 5.096774  8.3895396 13.486314
[31,] -11393.2010 5.000000 11.3932010 16.393201
```

Figure 2 is the same situation computed and plotted using the script in this essay[6].

[6]With some additions that will be demonstrated in class!

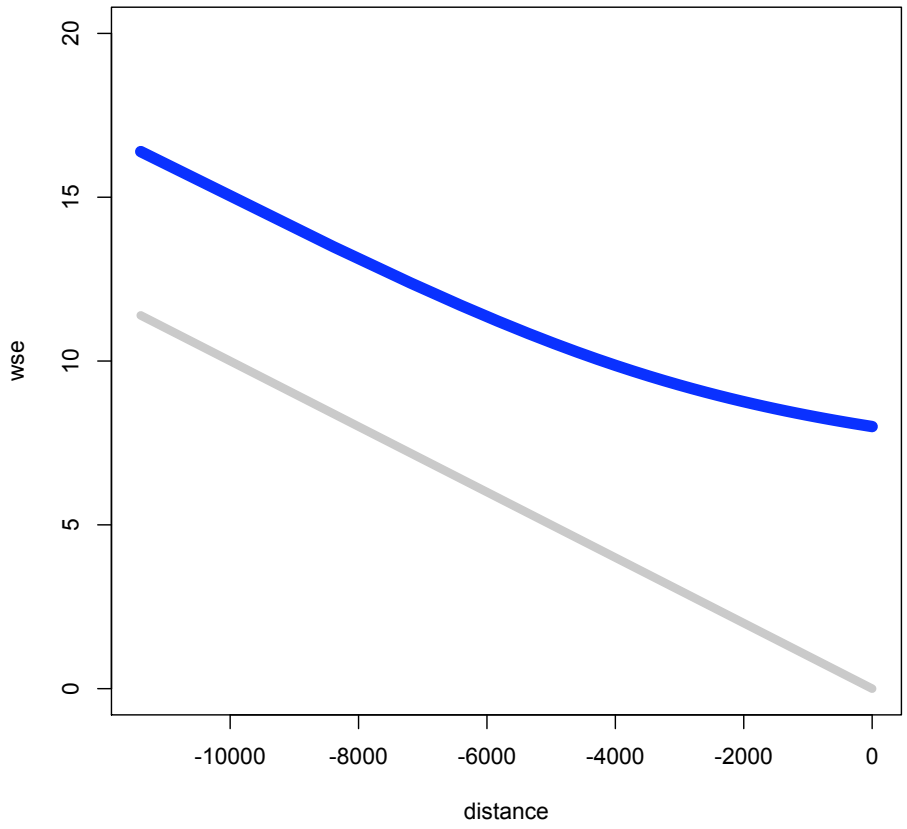


Figure 2. Backwater curve computed and plotted using **R**.

References

Sturm T.W (2001). *Open Channel Hydraulics, 1ed.*. McGraw-Hill, New York.

Cunge, J.A., Holly, F.M., Verwey, A. (1980). Practical Aspects of Computational River Hydraulics. Pittman Publishing Inc. , Boston, MA. pp. 7-50