COMPUTER ASSISTED ANALYSIS OF FLOW OVER SPILLWAY CHUTES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MERİÇ KIRANOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CIVIL ENGINEERING

JUNE 2019

Approval of the thesis:

COMPUTER ASSISTED ANALYSIS OF FLOW OVER SPILLWAY CHUTES

submitted by **MERİÇ KIRANOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Civil Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Ahmet Türer
Head of Department, **Civil Engineering** _____

Prof. Dr. A. Melih Yanmaz
Supervisor, **Civil Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Nuri Merzi
Civil Engineering Dept., METU _____

Prof. Dr. A. Melih Yanmaz
Civil Engineering, METU _____

Prof. Dr. İsmail Yücel
Civil Engineering Dept., METU _____

Assist. Prof. Dr. Müsteyde Baduna Koçyiğit
Civil Engineering Dept., Gazi University _____

Assist. Prof. Dr. Meriç Yılmaz
Civil Engineering Dept., Atılım University _____

Date: 17.06.2019

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: MERİÇ KIRANOĞLU

Signature:

ABSTRACT

COMPUTER ASSISTED ANALYSIS OF FLOW OVER SPILLWAY CHUTES

KIRANOĞLU, MERİÇ
Master of Science, Civil Engineering
Supervisor: Prof. Dr. A. Melih Yanmaz

June 2019, 243 pages

In this study, a literature review is conducted in terms of assumptions, approaches, and various calculation methods used for the analysis of flow over conventional and stepped type spillways. In guidance of this review, a software is developed to perform such analyses with the set of given data by the user, and display results with some tables and figures. To this end, water surface profile over the spillway face is determined considering air entrainment process, pressure distribution over the crest under various flow conditions are checked, and the possibility of cavitation along the spillway chute is searched. Besides, hydraulic jump over the stilling basin is determined and necessary stilling basin level is calculated according to the given downstream energy level. Similar computations are also conducted for stepped spillways such that a designer can perform quick successive analyses for various alternatives. The software is applied to an example to illustrate its use and discuss the results of the analyses.

Keywords: Overflow Spillway, Stepped Spillway, Chute, Flow, Aeration

ÖZ

DOLUSAVAK DÜŞÜLERİ ÜZERİNDE OLUŞAN AKIMIN BİLGİSAYAR DESTEKLİ ANALİZİ

KIRANOĞLU, MERİÇ
Yüksek Lisans, İnşaat Mühendisliği
Tez Danışmanı: Prof. Dr. A. Melih Yanmaz

Haziran 2019, 243 sayfa

Bu çalışmada, klasik ve basamaklı tipteki dolusavakların tasarımı ve hidrolik analizi konusunda kabuller, yaklaşımlar ve çeşitli hesaplama yöntemlerini kapsayan bir literatür taraması yapılmıştır. Bu araştırmanın rehberliğinde, kullanıcının girdiği veriler doğrultusunda dolusavağın tasarımını ve akım analizini gerçekleştiren, sonuçları tablo ve şekillerle sunan bir yazılım geliştirilmiştir. Bu bağlamda, hava girişi, çeşitli akım koşullarında kret üzerinde oluşacak basınç dağılımı, dolusavak boyunca yüzeyde kavitasyon oluşma olasılığı gibi konular dikkate alınarak dolusavak yüzeyindeki su yüzü profili belirlenmiştir. Ayrıca, enerji kırıcı havuzda oluşan hidrolik sıçrama hesaplanarak mansap enerji çizgisine göre gerekli enerji kırıcı havuz taban kotu belirlenmiştir. Kullanıcının çeşitli seçenekler için kısaca analiz yapabilmesi için benzer hesaplamalar basamaklı dolusavaklar için de yapılmıştır. Yazılım, kullanımın görülmesi ve analiz sonuçlarının tartışılması amacıyla bir örnek üzerinde uygulanmıştır.

Anahtar Kelimeler: Dolusavak, Basamaklı Savak, Şut Kanalı, Akım, Havalanma

To my dear aunt Emel ÜNLÜOĞLU…

ACKNOWLEDGEMENTS

The author would like to express his gratitude to his supervisor Prof. Dr. A. Melih YANMAZ for his assistance, support, and patience throughout the conduction of study.

The author would like to thank his former co-supervisor Assoc. Prof. Dr. Melih ÇALAMAK for his advices.

Special thanks are extended to the author's mother Ümit KIRANOĞLU and his lifetime companion Esin KIRANOĞLU for their endless love, patience, and support during his life.

Encouragement and guidance of Alper SUNGUR during the preparation of this thesis is acknowledged.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

B	Spillway length
B_p	Pier thickness
C_d	Discharge coefficient
C_{ent}	Air concentration
D_H	Hydraulic depth
d	Flow depth
d_0	Uniform flow depth
d_c	Critical flow depth
d_i	Flow depth at the inception point
d_m	Modified flow depth due to bulking
$(d_c)_{onset}$	Characteristic critical flow depth for stepped spillways
E_d	Downstream bed elevation
EGL_d	Downstream energy grade elevation
F_r	Froude number
F_{r1}	Froude number at upstream of the hydraulic jump
g	Gravitational acceleration
H	Total head
H_{crest}	Total head on the spillway crest
H_d	Design head
H_e	Existing head on the crest
$H_{sequent}$	Total head on the sequent depth section
H_{toe}	Total head on the spillway toe
h_s	Step height
h_L	Friction headloss
$E_{\%C-1}$	Headloss between the spillway crest and the spillway toe
$E_{\%1-2}$	Headloss between the toe and the sequent depth location
K_a	Abutment coefficient

K_c	Spillway upstream face slope parameter
K_p	Pier coefficient
K_s	Spillway crest elevation
k_s	Roughness height
L_B	Length of the stilling basin
L_e	Effective length of the crest
L_i	Length of inception point
L_{net}	Net length of the crest
l_s	Step length
N_p	Number of piers
n	Manning's roughness coefficient of concrete
n_c	Spillway upstream face slope parameter
Q_d	Design discharge
Q_e	Existing discharge
q	Discharge per unit width
y_1	Initial depth of hydraulic jump
y_2	Sequent depth of hydraulic jump
P	Reference pressure
P_v	Vapor pressure of water
p	Local pressure
U	Reference velocity
u	Local velocity
R	Radius of crest upstream quadrant curvatures
r	Abutment rounding radius
S^*	Dimensionless free surface elevation over crest
s	Water depth measured in vertical axis
V_0	Uniform flow velocity
X	Distance in horizontal axis normalized to design head
X^*	Transformed distance in horizontal axis
x	Distance in horizontal axis

x_i	Streamwise distance between crest and inception point
x_{ix}	Distance to the inception point from any node
x_s	Distance in the direction of channel
y_{aw}	Flow depth with air-water mixture
Z	Distance in vertical axis normalized to design head
Z^*	Transformed distance in vertical axis
z	Distance in vertical axis
α	Channel slope angle with horizontal
γ	Specific weight of water
δ	Turbulent boundary layer thickness
θ	Downstream channel angle
ρ	Density of water
σ	Cavitation parameter
χ	Head ratio
χ_s	Generalized streamwise coordinate

CHAPTER 1

INTRODUCTION

1.1. General

A spillway is a very important component of a dam that releases excess water from reservoir to downstream area. Its capacity should be selected with great care such that the possibility of overtopping of the dam crest is eliminated. Hydraulic conformity of this component needs to be checked with respect to the development of various flow conditions along the crest and the chute. Furthermore, energy dissipating facility is to be designed in such a way that it leads to dissipation of excessive energy at the toe to maintain foundation safety.

Hydraulic design of a spillway and its energy dissipating basin has been one of the most studied subjects in hydraulic engineering (Savage and Johnson 2001). Design guidelines have been recommended according to comprehensive laboratory experimentations. Most of the approaches and assumptions during the process of design and analysis of the components are quite certain. However, it may require repetitive calculations to end up a complete spillway design. With the advancement in computer technology and numerical solution techniques of governing equations, computer assisted analysis techniques are now available. Under these circumstances, it is possible to benefit from computer processing speed in design and analysis of a spillway.

The United States Army Corps of Engineers Waterways Experiment Station (USACE-WES) has conducted many studies to investigate the flow characteristics over spillways. They published several design manuals and charts. These design manuals and charts help engineers in the design and analysis of a spillway for a given flow condition (Chatila and Tabbara 2004).

Recently, Computational Fluid Dynamics (CFD) has been started to be used in the analysis of flow over spillways. The results of these numerical models were in good agreement with the experimental results, i.e. bottom pressures over the crest, water surface profiles and the discharge coefficient are determined precisely by these techniques. The foundation of CFD is built based on the conservation laws of mass, momentum, and energy, i.e. the Navier-Stokes equations which are a group of partial differential equations that compute fluid flow (Hashimi 2018).

FLUENT (ANSYS 2009) and FLOW-3D (Flow Science 2019) are the most common softwares which deal with CFD. These softwares require great care in the preparation of model, i.e., engineers should be competent in fluid mechanics. In addition to this, input data of these softwares are more complicated when compared to the use of the design manuals and charts. Therefore, they may not be practical for engineers dealing with a preliminary design of a spillway.

1.2. The Aim and Scope of the Study

The objective of this study is to develop a software which provides information for various design alternatives of an overflow spillway in a short time. The outputs of the software have a wide spectrum and satisfy all requirements for a spillway design, such as definition of spillway crest and face profiles, bottom pressure profiles and cavitation checks along the spillway, water surface profile starting from the spillway crest to the beginning of the stilling basin with the use of standard step method and recently proposed methods in the literature, and determination of the type and calculation of necessary length of the stilling basin. Besides providing these data in both tabular and visual forms, the software also informs the user if any design requirement is not satisfied. Furthermore, the software suggests adjusting the stilling basin elevation considering the given downstream energy grade level.

Spillways are classified as controlled and uncontrolled depending on whether the gates are located on the crest or not. Furthermore, the upstream face of the spillway can be either inclined or vertical. In this research, existence of piers on the crest are

considered. The gates are assumed fully open such that the spillway acts as an uncontrolled structure having a vertical upstream face. Moreover, the effects of submergence of the flow and the level of the apron on the discharge coefficient are ignored. This is a realistic approach for medium and high spillways. Hence, spillway crest is considered to be the only control section in the system.

This thesis is composed of five chapters. Introduction, aim and the scope of the study are given in Chapter 1. Hydraulics of overflow spillways are discussed in Chapter 2. Chapter 3 gives the methodology followed in the thesis together with the description of the developmental steps of the computer code. An application is presented in Chapter 4; whereas conclusions of the thesis are given in Chapter 5. The computer code of the software is given in Appendix A.

CHAPTER 2

HYDRAULICS OF OVERFLOW SPILLWAYS

2.1. Ogee Shaped Spillways

Ogee shaped crest is the most common control structure for overflow spillways. Due to its hydraulic performance, it has received much attention from engineers.

2.1.1. Crest Geometry

Ogee shape is derived from the lower surface of an aerated nappe flowing over a sharp-crested weir when design discharge passes over the spillway. This particular shape results in near-atmospheric pressure along the crest for a certain head which is called "design head". Therefore, ideal flow conditions are expected for the design head (Savage and Johnson 2001).

Ogee shaped spillway results in near-atmospheric pressure over the crest for design head. For higher heads, pressure values below atmospheric pressure cause a suction effect which increases the discharge over the spillway. For lower heads, due to the crest resistance, the discharge is less (Kim and Park 2005), and pressures over the crest profile are greater than the atmospheric pressure.

For the discharge at design head (design discharge), the flow glides over the crest with no loss of contact from the boundary surface and attains near-maximum discharge efficiency (USBR 1987).

According to Waterway Experiment Station, the upstream quadrant is subdivided into three domains. When the spillway crest is chosen as the origin (0, 0) and all quantities are normalized with the design head, US Corps of Engineers proposed following equations for crest bottom profile (Kiamanesh 1996; USBR 1987):

$$(X + 0.2418)^2 + (Z + 0.136)^2 = (0.04)^2 \quad \text{for} -0.2818 \leq X \leq -0.276 \qquad (2.1)$$

$$(X + 0.105)^2 - (Z + 0.219)^2 = (0.2)^2 \quad \text{for} -0.276 \leq X < -0.175 \qquad (2.2)$$

$$X^2 + (Z + 0.5)^2 = (0.5)^2 \quad \text{for} -0.175 \leq X \leq 0 \qquad (2.3)$$

where X and Z represent the relative distances in the horizontal axis and vertical axis, respectively, i.e., $X = x/H_d$ and $Z = z/H_d$ where H_d is the design head, x is the distance in horizontal axis, and z is the distance in vertical axis. As shown in Figure 2.1, the profile has discontinuities at the connection points X = -0.276 and X = -0.175 (Kiamanesh 1996) where R represents the radius of curvatures defined for upstream quadrant.

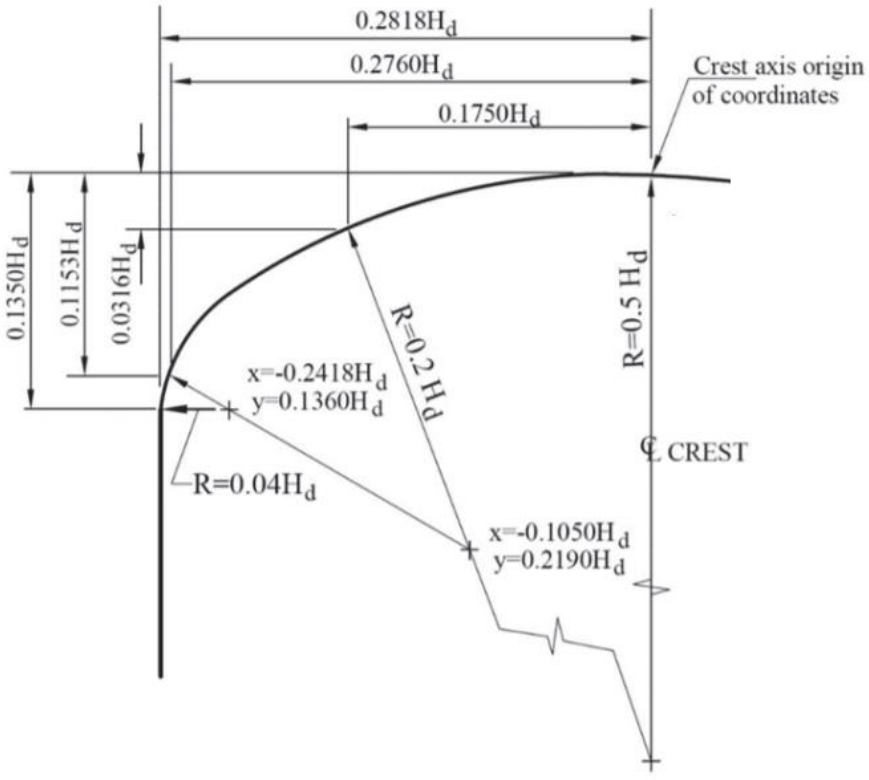


Figure 2.1. Spillway Upstream Quadrant Curvatures (Kiamanesh 1996)

Such a crest geometry is complicated to use for computational approaches due to the discontinuities on the curvature. An alternative approach with a smooth curvature was provided by Hager (1987):

$$Z^* = -X^* \ln X^* \qquad \text{for } X \geq -0.2818 \qquad (2.4)$$

where (X^*, Z^*) are transformed coordinates based on USACE shape as (Hager 1987):

$$X^* = 1.3055\,(X + 0.2818) \qquad (2.5)$$

$$Z^* = 2.705\,(Z + 0.136) \qquad (2.6)$$

The difference between these two profiles is negligible. Therefore, the second approach is used in the computations.

On the other hand, downstream quadrant profile, which is originally proposed by Creager (1917) as:

$$Z = -K_c (X)^{n_c} \qquad (2.7)$$

where K_c and n_c are parameters depending on the slope of the upstream face (See Table 2.1).

Table 2.1. Spillway Upstream Face Slope Parameters

Slope of the upstream face	K_c	n_c
Vertical	2	1.85
3V:1H	1.936	1.836
3V:2H	1.939	1.81
3V:3H	1.873	1.776

2.1.2. Discharge Calculations

Discharge over an ogee crest is given by the equation (Hager 1991):

$$Q = C_d L_e H \sqrt{2gH} \tag{2.8}$$

where Q is the discharge, L_e is the effective length of the crest, C_d is the discharge coefficient, H is the total head over the spillway crest.

Effective length varies with the total head over the spillway,

$$L_e = L_{net} - 2H(N_p K_p + K_a) \tag{2.9}$$

$$L_{net} = B - (B_p N_p) \tag{2.10}$$

where L_{net} is the net length, B_p is the pier thickness, N_p is the number of piers, K_p is the pier coefficient and K_a is the abutment coefficient.

All spillways have abutments on the crest, and many of them have intermediate piers. These structures may partially disturb the flow over the spillway crest. Hence, their geometric properties should be taken into account. The pier contraction coefficient K_p varies with the shape, thickness of the pier, design head, and approach flow velocity. Average pier contraction coefficients may be taken as follows (USBR 1987):

For square-nosed piers with corners rounded on a radius equal to about 10% of the pier thickness: $K_p = 0.02$

For round-nosed piers: $K_p = 0.01$

For pointed-nose piers: $K_p = 0$.

The abutment contraction coefficient K_a varies with the shape of the abutment, the angle between the upstream wall and the flow direction, existing head in relation to

design head and the approach flow velocity. Average abutment contraction coefficients may be taken as follows:

For square abutments with headwall perpendicular to the direction of flow $K_a = 0.20$,

For rounded abutments with headwall perpendicular to the direction of flow, when $0.5\ H \geq r \geq 0.15\ H$, $K_a = 0.10$,

For rounded abutments where $r \geq 0.5H$, and the headwall is placed not more than 45º to the direction of flow, $K_a = 0$, in which r is the radius of abutment rounding.

Discharge coefficient needs to be modified with reference to the effects of inclined upstream face, level of apron, and downstream submergence (USBR 1987). However, the effects of apron level and submergence may be considered for low spillways, which is not in the scope of this thesis.

As a more recent approach, Vischer and Hager (1998) provided a discharge coefficient, C_d, which varies with the head ratio (χ) exclusively.

$$C_d = \frac{2}{3\sqrt{3}}\left(1 + \frac{4\chi}{9 + 5\chi}\right) \qquad (2.11)$$

where χ is the head ratio H_e/H_d, and H_e is the existing head over the spillway crest.

In order to define the crest geometry, design head should be calculated. Since effective length varies with the total head (design head including the approach velocity head) over the spillway crest, calculation of total head requires an iterative solution.

When the existing head over the crest is different from design head, discharge coefficient varies. As mentioned before, smaller existing discharge will result in positive pressures along the contact surface, hence reduce the discharge. On the other hand, greater existing discharge causes negative pressures along the contact surface which will result in an increased discharge.

2.1.3. Water Surface Profile over the Crest

Determination of water surface profile is an important step for a spillway design. As shown in Figure 2.2, the water surface profile decreases almost linearly. In Figure 2.2, s is the flow depth perpendicular to the horizontal and α is the chute angle with the horizontal.

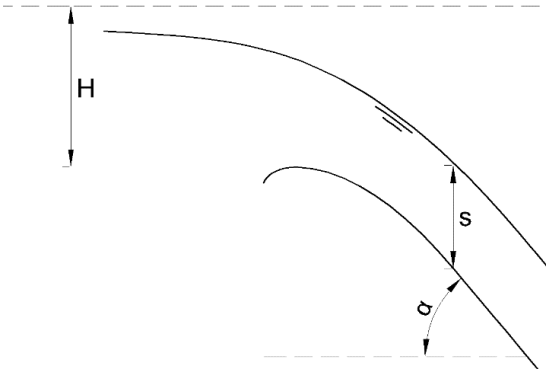


Figure 2.2. Water Surface Profile over the Crest (Hager, 1991)

A generalized approach for the flow over the standard-shaped overflow crest was provided recently by Hager (1991).

$$S^* = 0.75\left(\chi^{1.1} - \frac{X^*}{6}\right) \qquad \text{for} -2 < X/\chi^{1.1} < 2 \tag{2.12}$$

where S^* is the dimensionless free surface profile. The vertical distance between the crest and the water surface profile, s, is given by the equation (Vischer and Hager 1998):

$$s = H_d\, S^* \tag{2.13}$$

2.1.4. Crest Bottom Pressures

As the spillway may be operated under heads other than the design head, the pressure will increase under lower heads and decrease under higher heads with respect to the atmospheric pressure.

Existence of a pier is critical for bottom pressures since it causes local changes in pressure distribution. This case is handled with the use of some charts. Each of these charts contains the pressure profiles along the crest for three different head ratios.

In Figure 2.3, bottom pressures on a spillway configuration without piers are given. In Figure 2.4, bottom pressures along the centerline of the pier on a spillway configuration with piers are given, whereas relative bottom pressures adjacent to the pier are presented in Figure 2.5.

These pressure distribution charts are obtained from the CW-801 experiments (Melsheimer et al. 1970). Pressures for intermediate head ratios can be obtained by interpolation (Chow 1959).

Static energy in the reservoir is greater than that at the spillway section. The reason for the sudden drop in the spillway upstream quadrant is the conversion of static energy to kinetic energy as the water flows (Chow 1959). However, for the heads less than design head, this pressure hardly becomes negative. As shown in these figures, head ratio and pressure distribution have an inversely proportional relationship. Pressure differences become more significant as the head ratio increases. Furthermore, pressure ratio decreases with the flow direction and almost tend to the atmospheric pressure at a distance approximately $1.4H_d$.

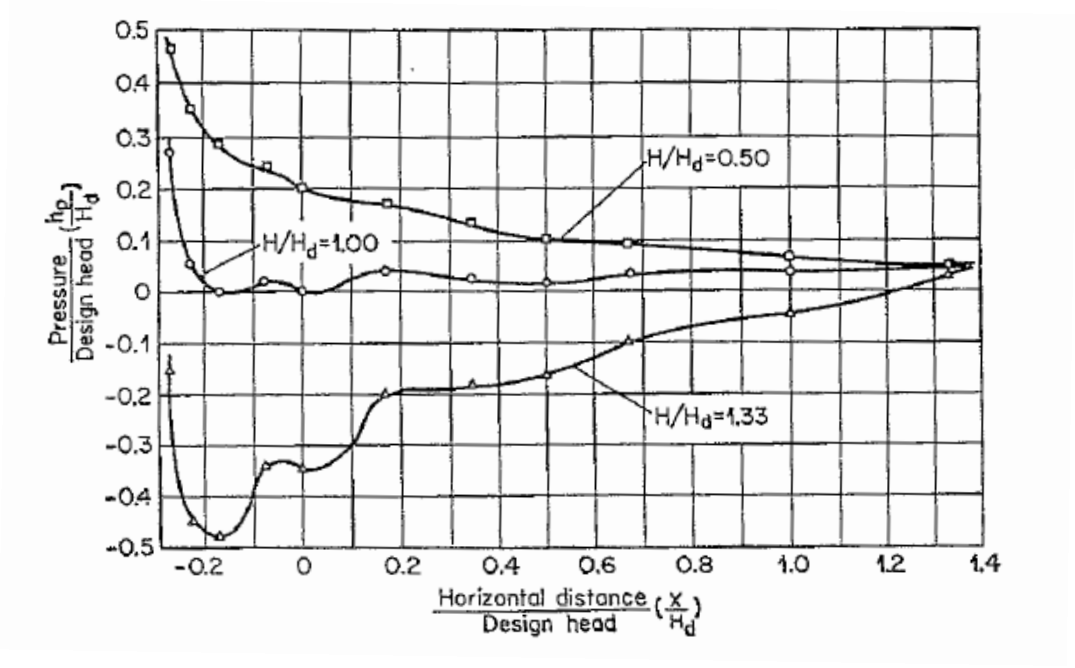


Figure 2.3. Crest Pressures for High Overflow Spillways (a) No Piers (Chow 1959)

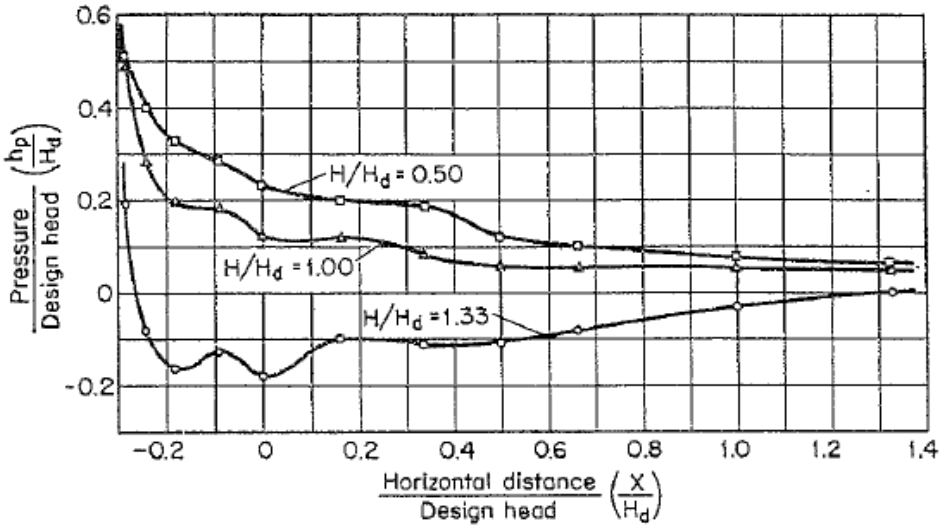


Figure 2.4. Crest Pressures for High Overflow Spillways (b) Along centerline of pier bay (Chow 1959)

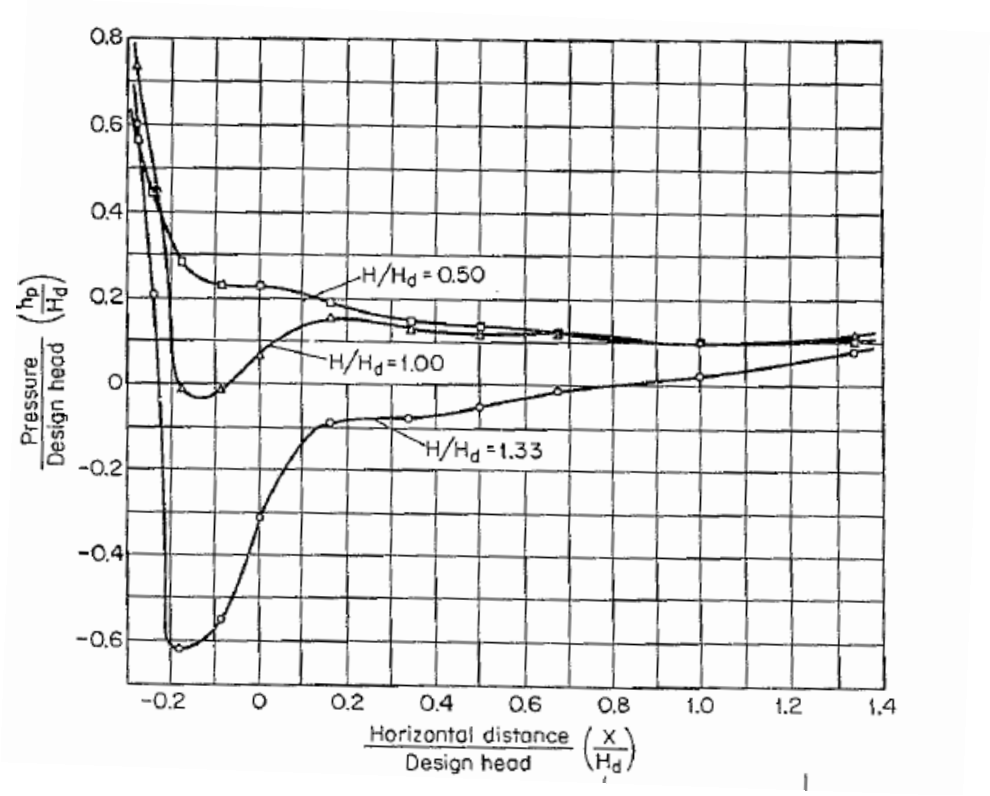


Figure 2.5. Crest Pressures for High Overflow Spillways (c) Along piers (Chow 1959)

2.1.5. Cavitation Concept

Cavitation is a significant problem that may occur when the local pressures reach the vapor pressure of the fluid (Molland 2011).

Theoretically, the minimum pressure on the spillway is atmospheric at the design head and it becomes negative for increasing heads. When local velocities exceed a critical value (e.g. 25 m/s (Vischer and Hager 1998)), possibility of cavitation is more likely. With further increase in velocity and hence decrease in local pressure, the liquid boils and air bubbles are formed in the flow that are carried away by the flow. When the flow reaches to a high pressure zone, air bubbles collapse and apply very big pressures on the spillway surface. These pressures may be as high as 800 Mpa and causes significant deformation on the spillway surface (Yanmaz 2018).

Origin of cavitation can be defined as the ratio of the difference in pressure to total dynamic pressure (Jagtap and Shrivastava 2014). Cavitation index, σ, normalizes this pressure drop to the dynamic pressure. Therefore, it can be written as (Graham et al. 1998),

$$\sigma = \frac{P - P_v}{\frac{1}{2}\rho U^2} \tag{2.14}$$

where P is a reference pressure which is usually taken as atmospheric pressure, P_v is the vapor pressure which varies with temperature, ρ is the density of the fluid. U is the reference velocity taken in the undisturbed conditions under atmospheric pressure.

A common assumption is that cavitation is expected when the flow velocity exceeds a threshold value (Falvey 1990). Cavitation problems can arise when velocity reaches 15 m/s. Serious damage is expected above 25 m/s (May 1987). It is suggested that head ratios should be selected for design so that the minimum pressure head cannot be less than -6.10 m (Melsheimer et al. 1970).

Cavitation index, σ, evaluates the occurrence potential of cavitation damage. If the index is greater than 0.5, significant cavitation damage is not expected for a typical spillway chute. For cavitation indices between 0.5 and 0.2, depending on the surface irregularities, cavitation damage can occur. If the cavitation index is below 0.2, in order to prevent cavitation damage, air entrainment is the only reliable method (USBR 2015).

2.2. Downstream Channels

Downstream channels convey the flow that passes through the control structure down to the stream below the dam through either a steep channel or a stepped channel. Flow velocity over the steep sloped channel (also called chute) is greater than the critical

velocity. Therefore, flow will not be disturbed by the downstream and the only control condition will be the upstream.

2.2.1. Chute Channel

Flow on a chute channel is gradually varied as shown in Figure 2.6, in which d_0 is the uniform depth, d_c is the critical depth and S_2 is the longitudinal gradually varied flow profile. As a boundary condition, spillway crest is located at the upstream. Starting from this point, flow depth along a chute can be calculated by using the methods listed below (Chow 1959);

-The graphical-integration method,

-The direct-integration method,

-Step methods,

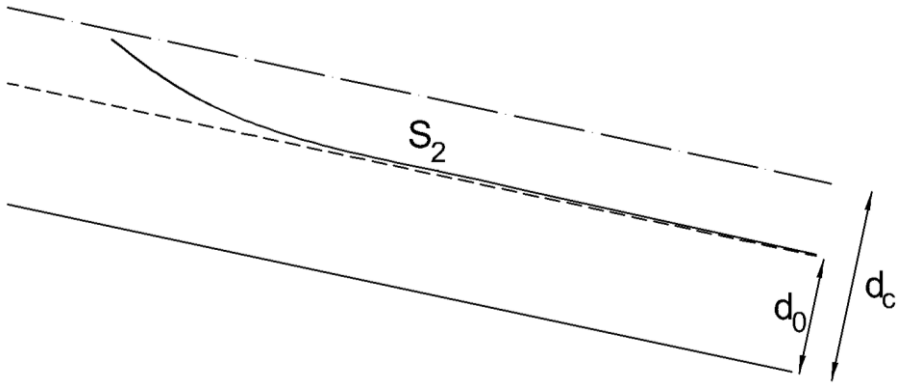


Figure 2.6. Water Surface Profile over the Chute Channel

The standard step method is the most common method to calculate the water surface profile along a chute. Therefore, it will also be used in this study.

2.2.1.1. Aeration on Chute

Chute conveys a turbulent (air-water mixture) flow with a high velocity. Close to the crest, the flow is non-aerated. Incipient aeration occurs when the turbulent boundary layer has reached the free surface (Falvey 1990).

The water surface may be slightly higher due to the entrainment of air. When air is entrained, it causes an increase in volume, also called bulking. For flows with Froude number below 1.5, this effect is negligible (USACE 1992).

Air entrainment must be considered in the design of a chute. Flow depth of air-water mixture, d_m, can be obtained with the use of Figure 2.7 (USACE 1992). In Figure 2.7, F_r stands for the Froude number.

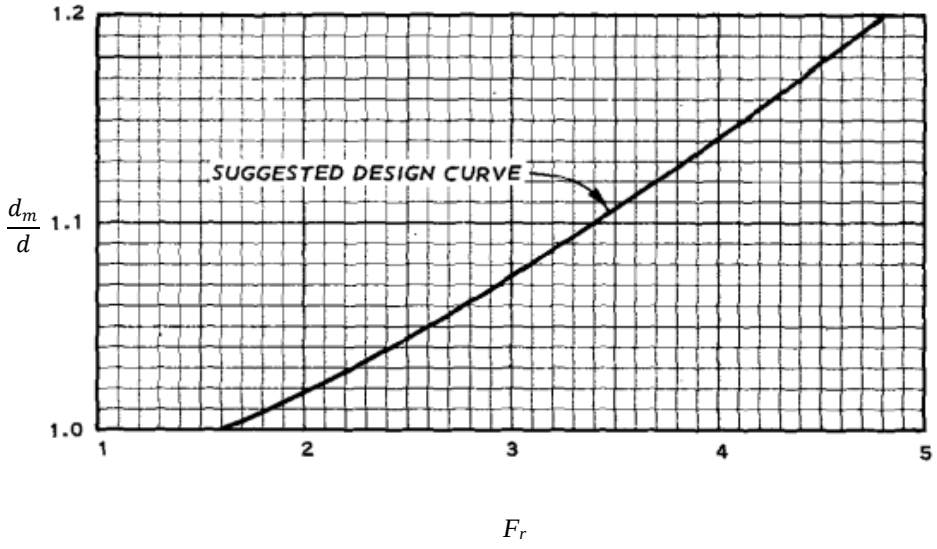


Figure 2.7. Modification of Flow Depth due to Bulking (USACE 1992)

Another approach provided by Hager and Blaser (1998) is given below.

Turbulent boundary layer begins to grow at the top of the crest. At the point where turbulent boundary layer reaches the free surface, incipient aeration occurs. Before

this point, flow is non-aerated. Turbulent boundary layer thickness, δ, can be computed using the following equation given by Hager and Blaser (1998):

$$\frac{\delta}{x_s} = 0.029\left[\sin^2\alpha\left(\frac{k_s}{x_s}\right)\right]^{1/7} \quad (2.15)$$

where x_s is the distance along the chute starting from the crest point and k_s is Nikuradse's equivalent roughness height, which is 0.0015 m for a typical concrete (Hager and Blaser 1998).

Water surface profile over the chute can be computed from Equation (2.16) in which d stands fort he water depth. (Hager and Blaser 1998),

$$\frac{d}{d_0} = \left[1 - \left(1 - \frac{d_0}{d_c}\right)\exp\left(-\frac{10}{3}\chi_s\right)\right]^{-1} \quad (2.16)$$

where χ_s is the generalized streamwise coordinate which can be computed with Equation (2.17) (Hager and Blaser 1998),

$$\chi_s = \sin\alpha\left(\frac{d_0}{d_c}\right)^3\left(\frac{x}{d_0}\right) \quad (2.17)$$

In Figure 2.8, I stands for the inception point of aeration.

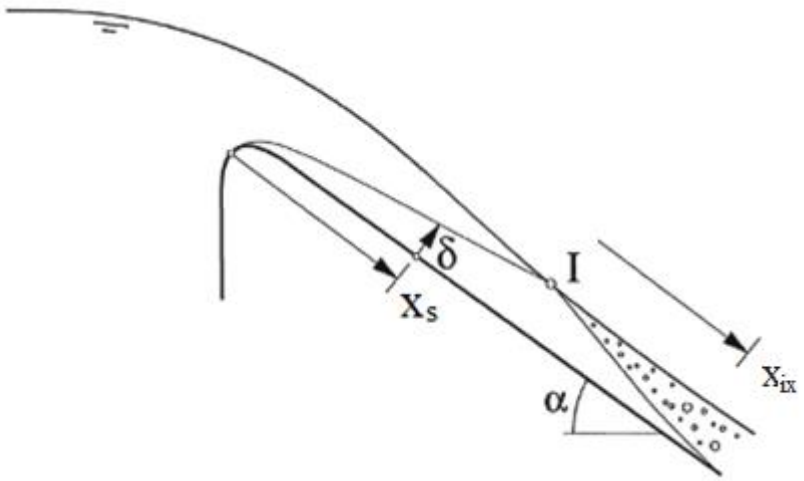


Figure 2.8. Notation for Incipient Aeration for Chute Flow (Hager and Blaser 1998)

Uniform flow depth can be computed from Equation (2.18) (Sinniger and Hager 1989),

$$d_0 = \left(\frac{Qk_s^{1/6}}{6.75\sqrt{g}L\sin^{1/2}\alpha}\right)^{3/5} \tag{2.18}$$

The critical depth for rectangular channels can be computed from Equation (2.19) (Chow 1959),

$$d_c = \left(\frac{Q^2}{gL^2}\right)^{1/3} \tag{2.19}$$

The location of air inception, x_i, and the depth at that point, d_i, can be determined from the following equations:

$$x_i = 16d_c(\sin\alpha)^{-0.60}\left(\frac{k_s}{d_c}\right)^{-0.08} \tag{2.20}$$

$$d_i = \frac{0.258\, d_c}{\left[\frac{1}{0.64}\left(\frac{d_c \sin^3 \alpha}{k_s}\right)^{1/10} - 1\right]^{1/2}} \tag{2.21}$$

Equation (2.21) is only valid for $d_i / d_0 < 0.25$. If $d_i / d_0 > 0.25$, d_i should be assumed to be equal to d_0.

Further from this point, flow is aerated. Therefore, air concentration should be taken into account. Flow depth with the air-water mixture is calculated from Equation (2.22) (Falvey 1990):

$$\frac{y_{aw}}{d} = \frac{1}{1 - C_{ent}} \tag{2.22}$$

where y_{aw} is the flow depth with air-water mixture. Air concentration, C_{ent}, is calculated using Equation (2.23) (Hager and Blaser 1998). In Equation (2.23), x_{ix} represents the distance measured from the inception point (Figure 2.8).

$$C_{ent} = 0.48 - 0.48 \exp\left(-0.01061 \frac{x_{ix}}{d_0}\right) \tag{2.23}$$

Flow aeration is important in design of sidewalls. It affects the water surface profile on the chute channel significantly. Hence, it should be taken into account.

2.2.2. Stepped Channel

History of stepped spillways go a long way back to B.C. 694, Khosr River Dams in Iraq. Much later, the Romans and then the Muslim engineers built many stepped spillways. Thereafter, Spanish engineers designed dams with overflow stepped spillways. In 1791, they built the largest dam with a stepped spillway named Puentes

Dam. Unfortunately, this dam collapsed in 1802 due to a foundation failure. Before 1850, Spanish engineers were quite experienced on dam engineering. After the conquest of America, this exceptional knowledge exported there. French and English engineers also designed many stepped spillways (Chanson 2008).

Stepped (cascade) channels may be characterized by two types of flow, which are skimming and nappe flows. At low flow rates, water flows as free-falling nappes following each other as shown in Figure 2.9. At larger flow rates, the water flows over the step edges while forming recirculating vortices between the main stream and step corners (See Figure 2.10).

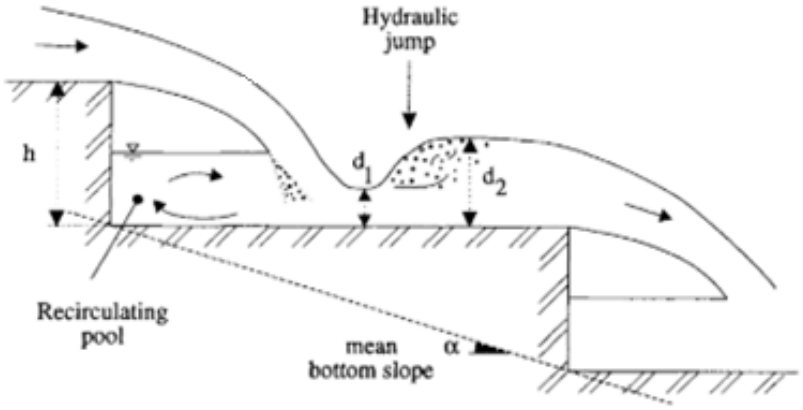


Figure 2.9. Nappe Flow (Chanson 1996a)

As the flow rate increases, flow generally experiences transitions from nappe flow to skimming flow. In the perspective of the designer, it is important to know where this transition occurs.

Chamani and Rajaratnam (1994) showed that nappe flow would be observed up to $d_c/h_s = 0.8$ where h_s is the step height, for a limited range of stepped channel angle ranging between 20° and 40°).

Chanson (1994) proposed Equation (2.24) for the onset of skimming flow. This equation is only valid for the h_s/l_s values ranging from 0.2 to 1.3.

$$\frac{(d_c)_{onset}}{h_s} = 1.057 - 0.465\,\frac{h_s}{l_s} \tag{2.24}$$

where, $(d_c)_{onset}$ is the characteristic critical flow depth, and l_s is the step length. For greater critical flow depth values than characteristic critical flow depth, skimming flow will occur.

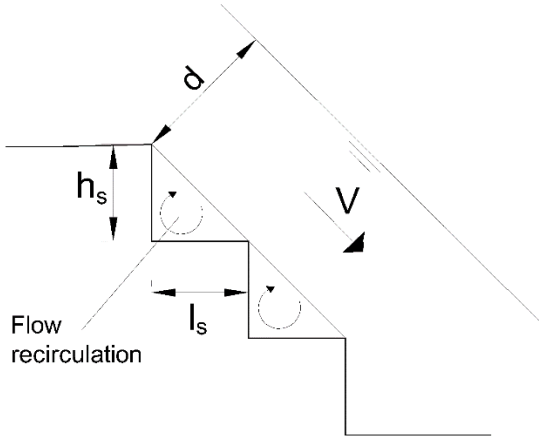


Figure 2.10. Skimming Flow

The main difference between these two types of flow is the pressure distribution across the flow. The change in pressure distribution is associated with a change of streamline directions. In skimming flow, the streamlines are nearly parallel to the pseudo-bottom formed by the step edges while they follow a path guided by the step geometry in nappe flow (Chanson 1996b).

Recent stepped spillways are designed for the skimming flow regime. This type of flow is highly turbulent. At the upstream, flow is smooth and turbulent, when the outer edge of the developing bottom boundary layer reaches the free surface, turbulence induces vigorous aeration. At the downstream, turbulence next to the free surface becomes large enough to initiate natural free-surface aeration (Gonzalez and Chanson 2007).

As shown in Figure 2.11, there are four different skimming flow regions which are,

(1) Non-aerated flow region
(2) Partially developed aerated flow region
(3) Fully developed aerated flow region
(4) Uniform flow region

In the upstream region, the turbulent boundary layer grows from the spillway surface. No air entrainment occurs in this region. The location where the turbulent boundary layer reaches the free surface is called the inception point. In region (2), air penetrates the fluid. This partially developed aerated flow region ends when the entrained air reaches the pseudo-bottom. In region (3), flow is fully aerated. However, equilibrium conditions are not satisfied until the uniform flow region where the flow characteristics become constant along the channel (Renna 2005).

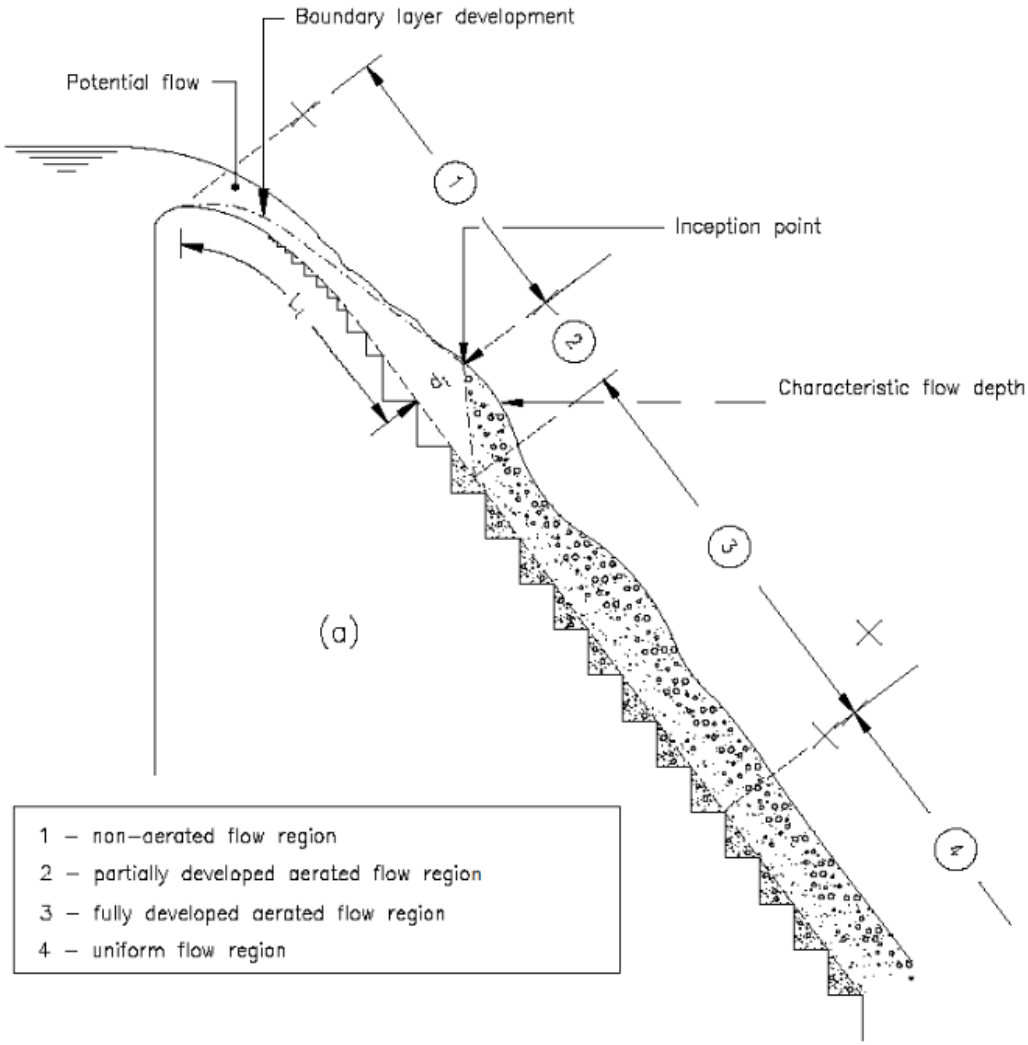


Figure 2.11. Skimming Flow Regions (Boes and Minor 2000)

2.2.2.1. Design Criterion

When designing a stepped spillway, the height of the spillway and the design discharge is known. Therefore, water surface elevation is computed at the upstream.

The step height should be selected to ensure that the stepped channel will operate with skimming flow conditions. Fully developed flow conditions must be achieved before the toe of the stepped channel. The point of inception should be reached before the toe

to ensure that happens. Its coordinates might be calculated by the following equations (Gonzalez and Chanson 2007),

$$\frac{L_i}{h_s \cos\theta} = 9.72\,(\sin\theta)^{0.08}\left(\frac{q}{\sqrt{g\sin\theta(h_s\cos\theta)^3}}\right)^{0.71} \tag{2.25}$$

$$\frac{d_i}{h_s \cos\theta} = \frac{0.4034}{(\sin\theta)^{0.04}}\left(\frac{q}{\sqrt{g\sin\theta(h_s\cos\theta)^3}}\right)^{0.592} \tag{2.26}$$

where L_i is the length of inception point, d_i is the flow depth at the inception point, θ is the angle between the horizontal and the pseudo-bottom formed by the step edges, q is the discharge per unit width and g is the gravitational acceleration.

Fully developed flow condition can be checked by the following equation (Gonzalez and Chanson 2007),

$$\frac{d_c}{h_s} < \frac{1}{0.1193\cos\theta\sin\theta^{0.259}\left(\frac{L}{h_s\cos\theta}\right)^{0.935}} \tag{2.27}$$

It is possible for steps to become too small and no longer act as a large roughness. Therefore (Chanson 1994) suggested a maximum step height which can be obtained from Equation (2.28),

$$h_s \leq 15 d_c \cos\theta \tag{2.28}$$

If the channel is long enough for the flow, uniform flow conditions prevail.

Uniform flow velocity, V_0, can be computed from Equation (2.29).

$$V_0 = \sqrt{\frac{8g \sin\theta}{f} \frac{D_H}{4}} \tag{2.29}$$

where f is the friction factor which is typically 0.20 for skimming flow (Gonzalez and Chanson 2007), D_H is the hydraulic depth, which is a function of uniform flow depth. Therefore, uniform flow depth should be calculated with the help of continuity equation:

$$d_0 = \frac{q}{V_0} \tag{2.30}$$

2.3. Terminal Structures

2.3.1. Hydraulic Jump Concept

As velocity increases in the downstream channel, the risk of scouring increases as well. Terminal structures are energy dissipating zones which are built to convey the flow to the river without any scouring or erosion problem. Hydraulic jump is an essential feature of energy dissipation. For Froude numbers between 1 and 1.7, the flow velocity is slightly greater than critical velocity. As flow regime changes from supercritical to subcritical, slightly ruffled water surface is observed. As the Froude number approaches to 1.7, small rollers begin to develop on the surface and these rollers become more intense with increasing Froude number until it reaches the value of 2.5. This form of hydraulic jump is called Form A, shown in Figure 2.12.

Figure 2.12. Form A – Prejump Stage (Weak Jump) (Chow 1959)

For Froude numbers between 2.5 and 4.5, an oscillating form of jump occurs. This type of jump is called Form B, shown in Figure 2.13.

Figure 2.13. Form B – Transition Stage (Oscillating Jump) (Chow 1959)

For Froude numbers between 4.5 and 9, a stable and well-balanced hydraulic jump occurs. This type of hydraulic jump is called Form C, shown in Figure 2.14.

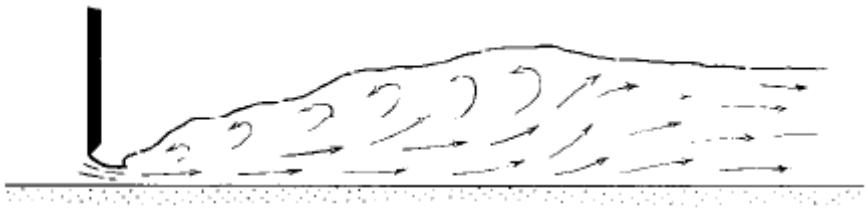


Figure 2.14. Form C – Well-balanced Jump (Steady Jump) (Chow 1959)

For Froude numbers greater than 9, high turbulence in the jump causes rough water surface with strong water waves downstream of the hydraulic jump. Figure 2.15 represents this type of hydraulic jump.

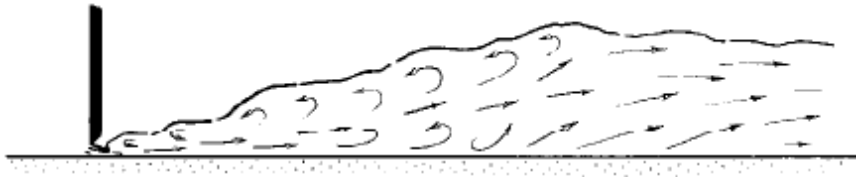


Figure 2.15. Form D – Effective Jump but Rough Surface Downstream (Strong Jump) (Chow 1959)

Momentum equation can be used to predict the sequent depth of a classical hydraulic jump. With the uniform velocity distribution and hydrostatic pressure distribution assumptions, sequent depth of the hydraulic jump in a rectangular channel can be calculated from (Yanmaz, 2018):

$$\frac{y_2}{y_1} = \frac{1}{2}\left[\left(1 + 8\,F_{r_1}^{2}\right)^{0.5} - 1\right] \tag{2.31}$$

where y_1 is the initial depth of the hydraulic jump, y_2 is the sequent depth of the hydraulic jump and F_{r_1} is the initial Froude number.

2.3.2. Standard Stilling Basin Design

Three standard types of stilling basin are developed by the U.S Bureau of Reclamation. Chute blocks and end sills are located at the entrance and exit of the stilling basin, respectively, to reduce the length of jump and stabilize it. Baffle piers are located at an intermediate position across the stilling basin to increase the dissipation by impact action. Approaching Froude number and velocity are two factors for selecting the stilling basin type. For Froude numbers less than 2.5, dissipation devices are unnecessary, but apron length should be greater than about five times y_2 (USBR 1987). For Froude numbers between 2.5 and 4.5, USBR Type IV basin is recommended (See Figure 2.16).

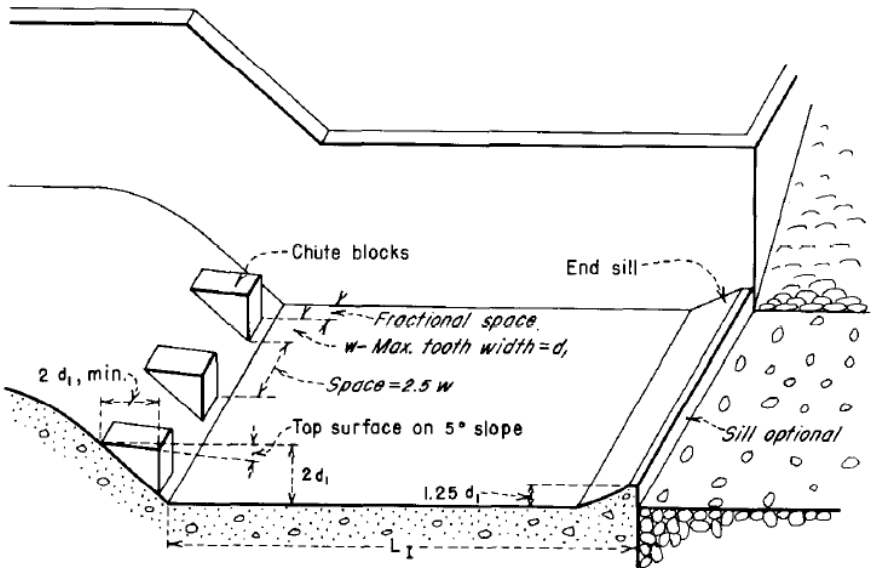


Figure 2.16. Type IV Stilling Basin (USBR 1987)

For Froude numbers greater than 4.5, basin type is selected regarding approaching velocity. If the velocity before the hydraulic jump is greater than 18 m/s, Type II basin is recommended, else Type III Basin is suitable (USBR 1987) (See Figure 2.17 and Figure 2.18).

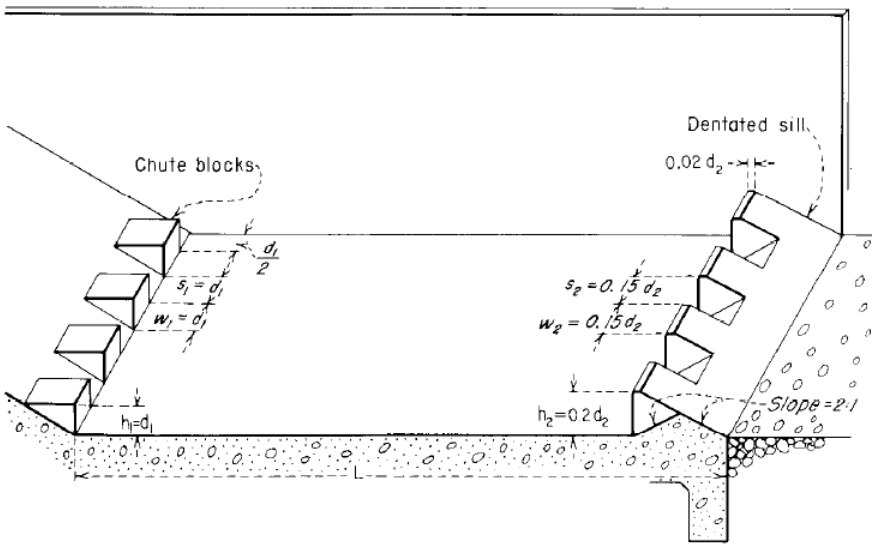


Figure 2.17. Type II Stilling Basin (USBR 1987)

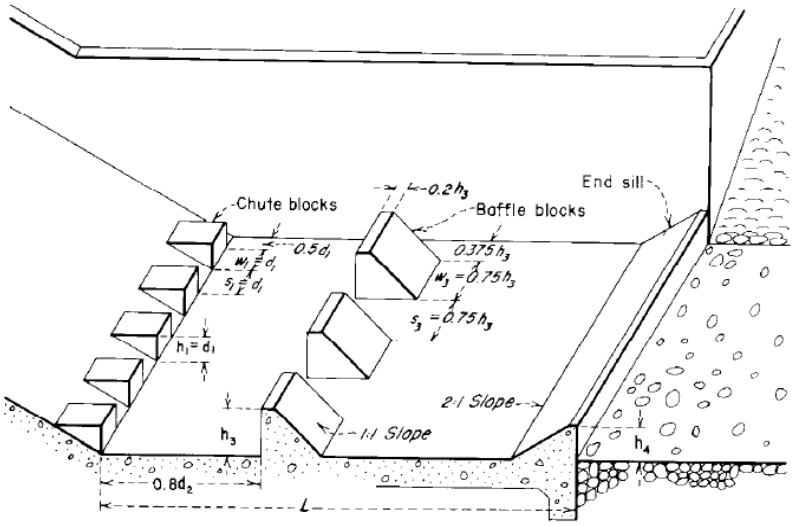


Figure 2.18. Type III Stilling Basin (USBR 1987)

Length of the stilling basin, L_B, is obtained from Figure 2.19.

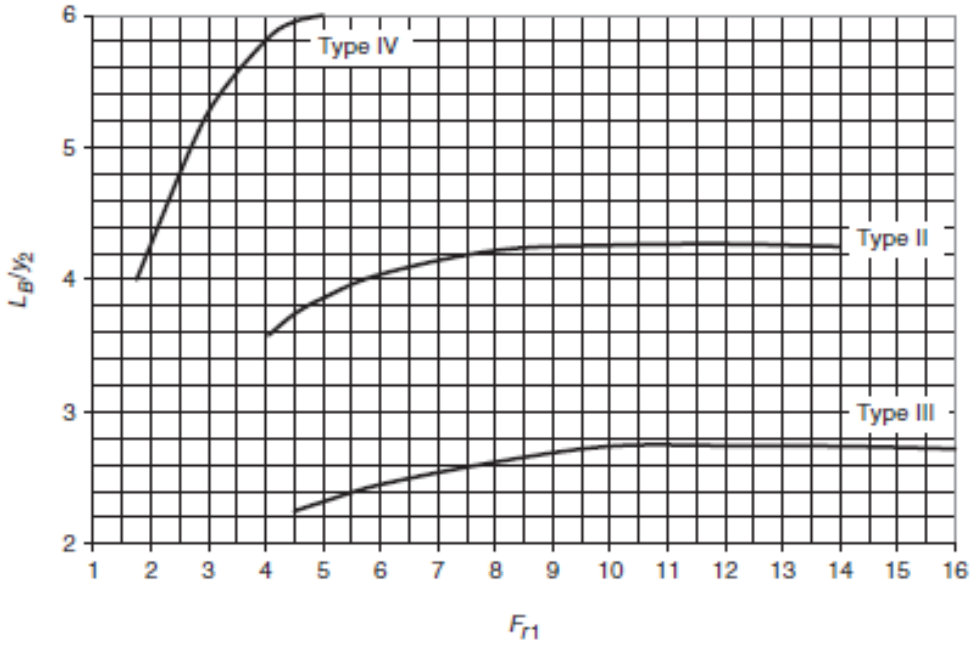


Figure 2.19. Stilling Basin Length (USBR 1987)

The information presented in this chapter will be used in the code developed in this thesis as explained in Chapter 3.

CHAPTER 3

DEVELOPMENT OF THE COMPUTER CODE

3.1. General Information

As summarized in Chapter 2, there are many items to consider while designing a spillway. Even for a small weir structure, the design process is time-consuming. In this thesis, a software named “SP-DAS” is developed to design a spillway structure in a short while. The interface of this software is designed in Windows Forms which is a graphical (GUI) class library included as a part of the “Microsoft .NET” framework. In the formation of the user interfaces used in software, an integrated development environment named “Microsoft Visual Studio” is used. The language of the code is C#, which is an object-oriented language. It can be used to build applications that run on the .NET framework. It has been recognized as one of the most powerful programming languages.

In the selection of code language, visualization has been the most important criteria, since the purpose of this study is to develop a user-friendly software for design engineers. On the other hand, performance is also somehow important, but not crucial, because the design may not require excessive computations. Besides, object-oriented programming languages, such as C# and Java performs better in terms of management and readability than procedure-oriented programs like C and FORTRAN (Biswa et al. 2016).

The software “SP-DAS” contains two different windows form objects which are the main screen and a tables window. Tables window provides all calculated data in tabular forms where the main screen provides a user-friendly design opportunity (See Figure 3.1 and Figure 3.2)

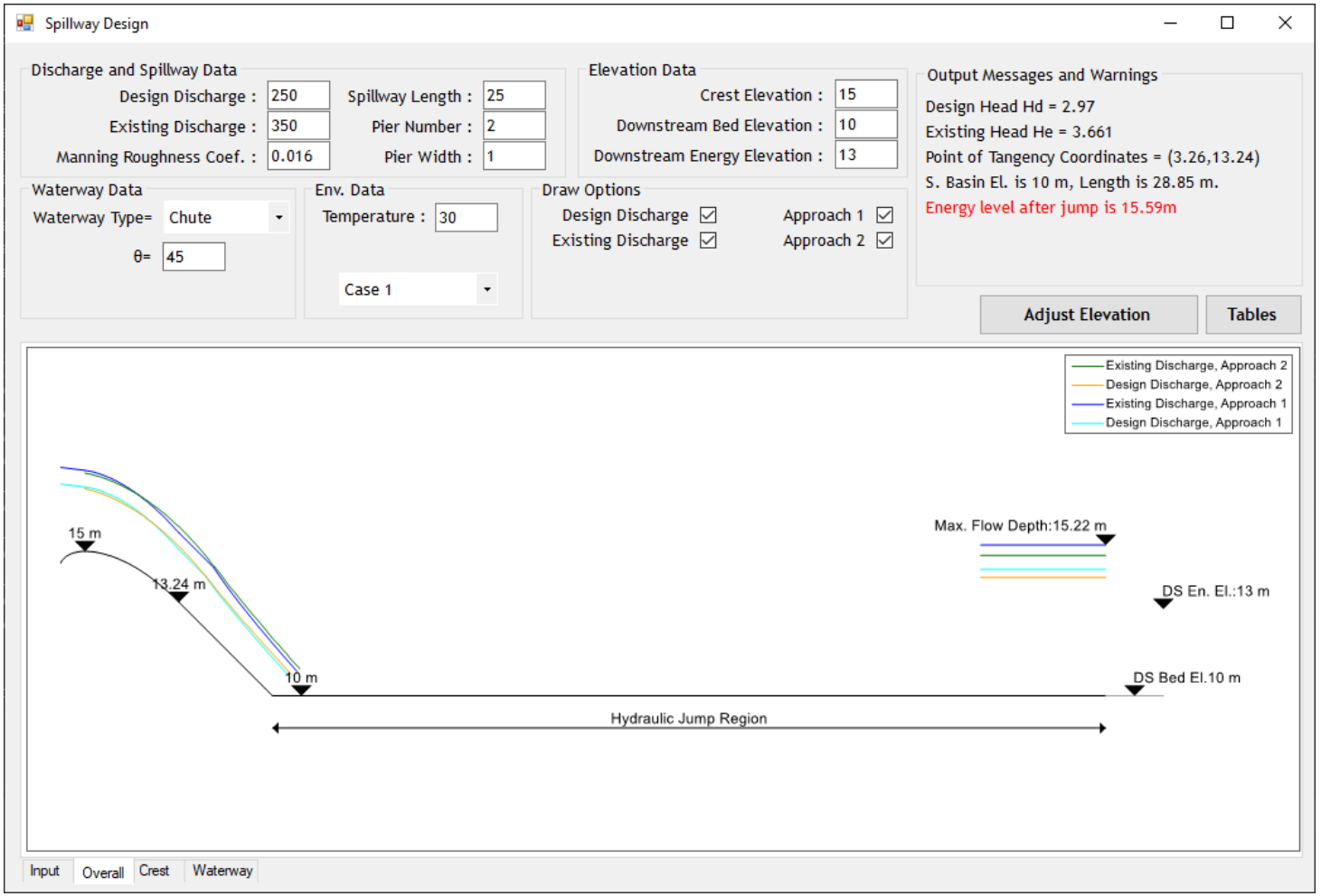
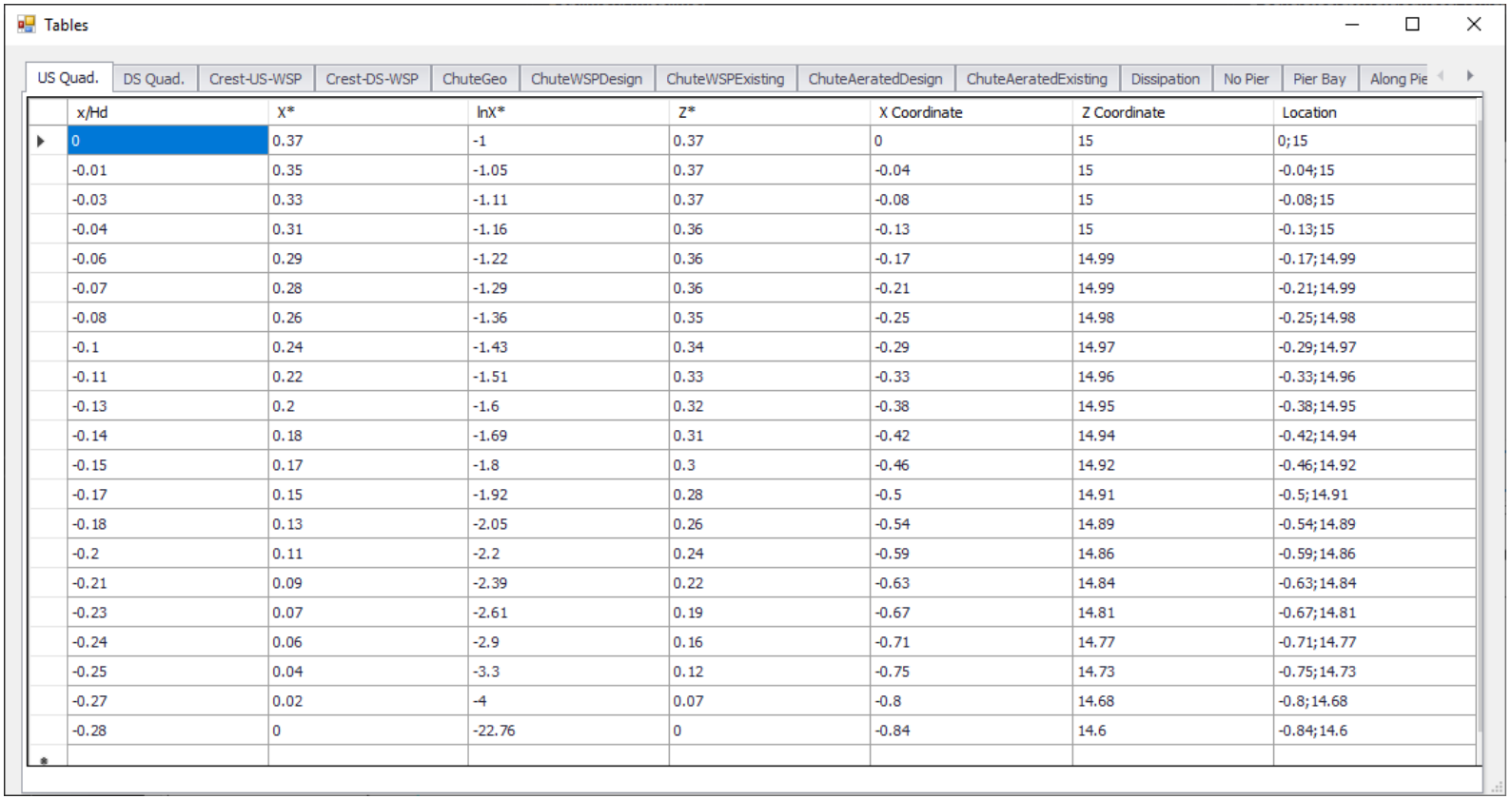


Figure 3.1. Software Main Window



x/Hd	X*	lnX*	Z*	X Coordinate	Z Coordinate	Location
0	0.37	-1	0.37	0	15	0;15
-0.01	0.35	-1.05	0.37	-0.04	15	-0.04;15
-0.03	0.33	-1.11	0.37	-0.08	15	-0.08;15
-0.04	0.31	-1.16	0.36	-0.13	15	-0.13;15
-0.06	0.29	-1.22	0.36	-0.17	14.99	-0.17;14.99
-0.07	0.28	-1.29	0.36	-0.21	14.99	-0.21;14.99
-0.08	0.26	-1.36	0.35	-0.25	14.98	-0.25;14.98
-0.1	0.24	-1.43	0.34	-0.29	14.97	-0.29;14.97
-0.11	0.22	-1.51	0.33	-0.33	14.96	-0.33;14.96
-0.13	0.2	-1.6	0.32	-0.38	14.95	-0.38;14.95
-0.14	0.18	-1.69	0.31	-0.42	14.94	-0.42;14.94
-0.15	0.17	-1.8	0.3	-0.46	14.92	-0.46;14.92
-0.17	0.15	-1.92	0.28	-0.5	14.91	-0.5;14.91
-0.18	0.13	-2.05	0.26	-0.54	14.89	-0.54;14.89
-0.2	0.11	-2.2	0.24	-0.59	14.86	-0.59;14.86
-0.21	0.09	-2.39	0.22	-0.63	14.84	-0.63;14.84
-0.23	0.07	-2.61	0.19	-0.67	14.81	-0.67;14.81
-0.24	0.06	-2.9	0.16	-0.71	14.77	-0.71;14.77
-0.25	0.04	-3.3	0.12	-0.75	14.73	-0.75;14.73
-0.27	0.02	-4	0.07	-0.8	14.68	-0.8;14.68
-0.28	0	-22.76	0	-0.84	14.6	-0.84;14.6

Figure 3.2. Software Tables Window

3.2. Inputs of the Software

A user will provide the following data for the design and flow analysis of spillway.

- Discharges,
- Spillway geometry (spillway length, pier, abutment data)
- Spillway crest elevation,
- Downstream channel type,
- Downstream channel geometry (chute angle or stepped channel geometry data)
- Tailwater bed elevation,
- Tailwater energy elevation,
- Manning's roughness coefficient of flow surface.

Some of these inputs are stated visually on the Inputs tab page of the main window as shown in Figure 3.3. During the planning stage, tailwater rating curves are obtained from water surface profile computations. Therefore, user would know the tailwater surface elevation and the corresponding energy grade line elevation for a particular discharge to be considered as a design discharge.

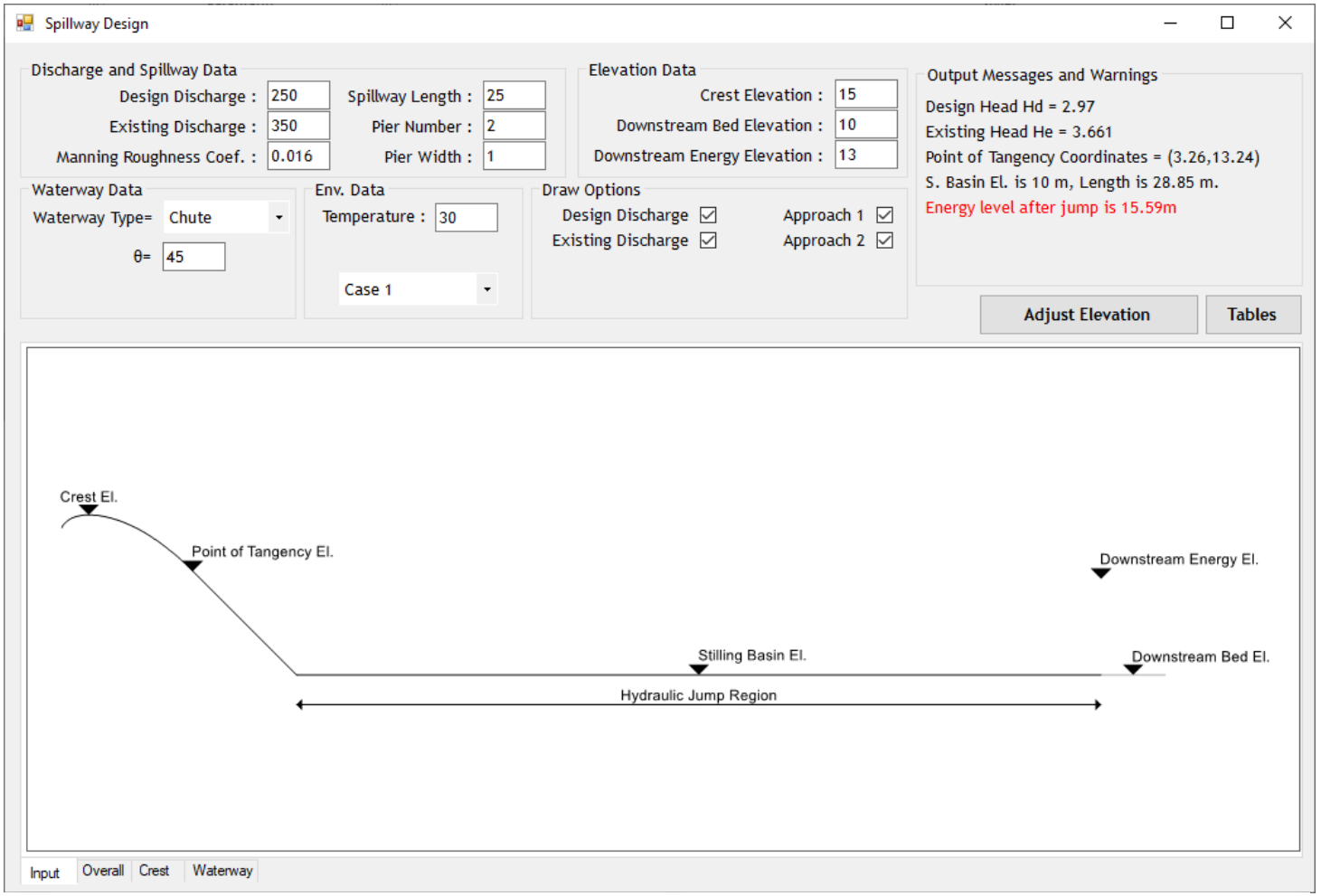


Figure 3.3. Software Input Window

3.3. The Flowchart

Computational steps of the design are presented as a flowchart in Figure 3.4. As shown in the flowchart, the initial step is the computation of design and existing heads over the spillway crest using the weir discharge equation. Then spillway crest bed profile is determined using the computed design head. At this point, crest bottom pressures which are influenced by the configuration of the spillway, design head and the head ratio can be investigated.

At the downstream of the ogee profile, conventional chute or stepped downstream channel is generated depending on the user selection. Then stilling basin is generated at the downstream channel elevation.

The following computational step describes computation of water surface profile over the crest and the downstream channel. At this point, the designer can select two different approaches. From the crest of the spillway to the end of the downstream channel, the software generates a water surface profile for each selected discharge and approach. Afterwards, energy dissipation calculations are performed for each of these flows.

The results are displayed on the main window of the software (See Figure 3.1). Flow characteristics including the downstream energy level are determined all along the spillway. Also, they are compared with the tailwater energy level on the main screen of the software. At this point, the designer may consider adjusting the stilling basin level, such that energy level at the section where sequent depth of the jump occurs coincides with the energy level at the tailwater section.

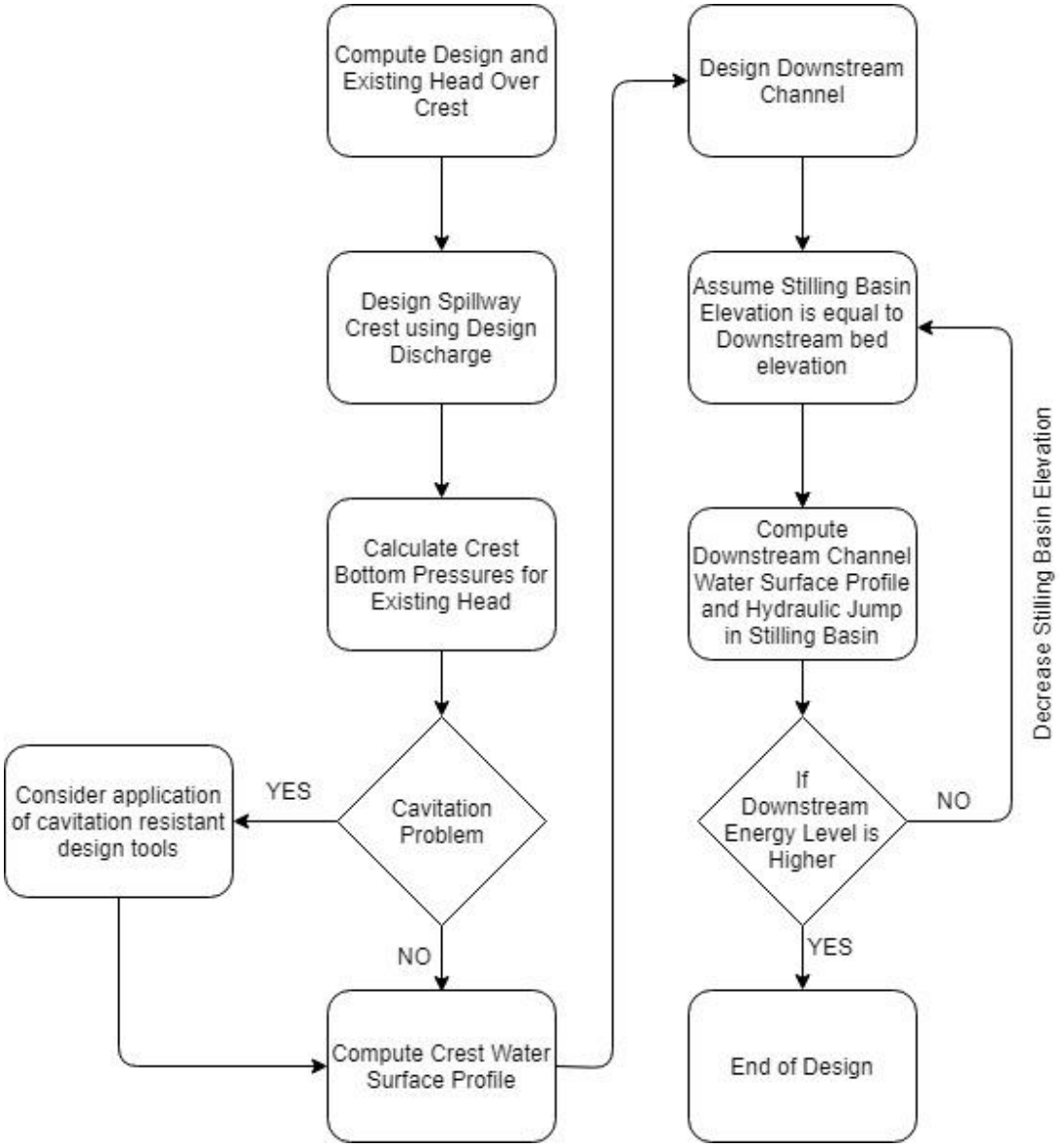


Figure 3.4. Spillway Design Flowchart

3.4. The Methodology

For demonstration purposes, the results of an arbitrary application are presented through Figure 3.1 to Figure 3.16 and Table 3.1 to Table 3.12.

3.4.1. Calculation of Design Head

Discharge equation (See Equation (2.8)) requires an iterative solution. Starting from 0.01 m, design head is increased with incrementation of 0.001 m until given design discharge is reached. Since the effective spillway length varies with the design head, it is updated in each step where the discharge coefficient remains constant.

3.4.2. Calculation of Existing Head

Starting from 0.01 m, the existing head is increased with incrementation of 0.001 m until the given existing discharge is reached. Discharge coefficient and effective spillway length are calculated at each step.

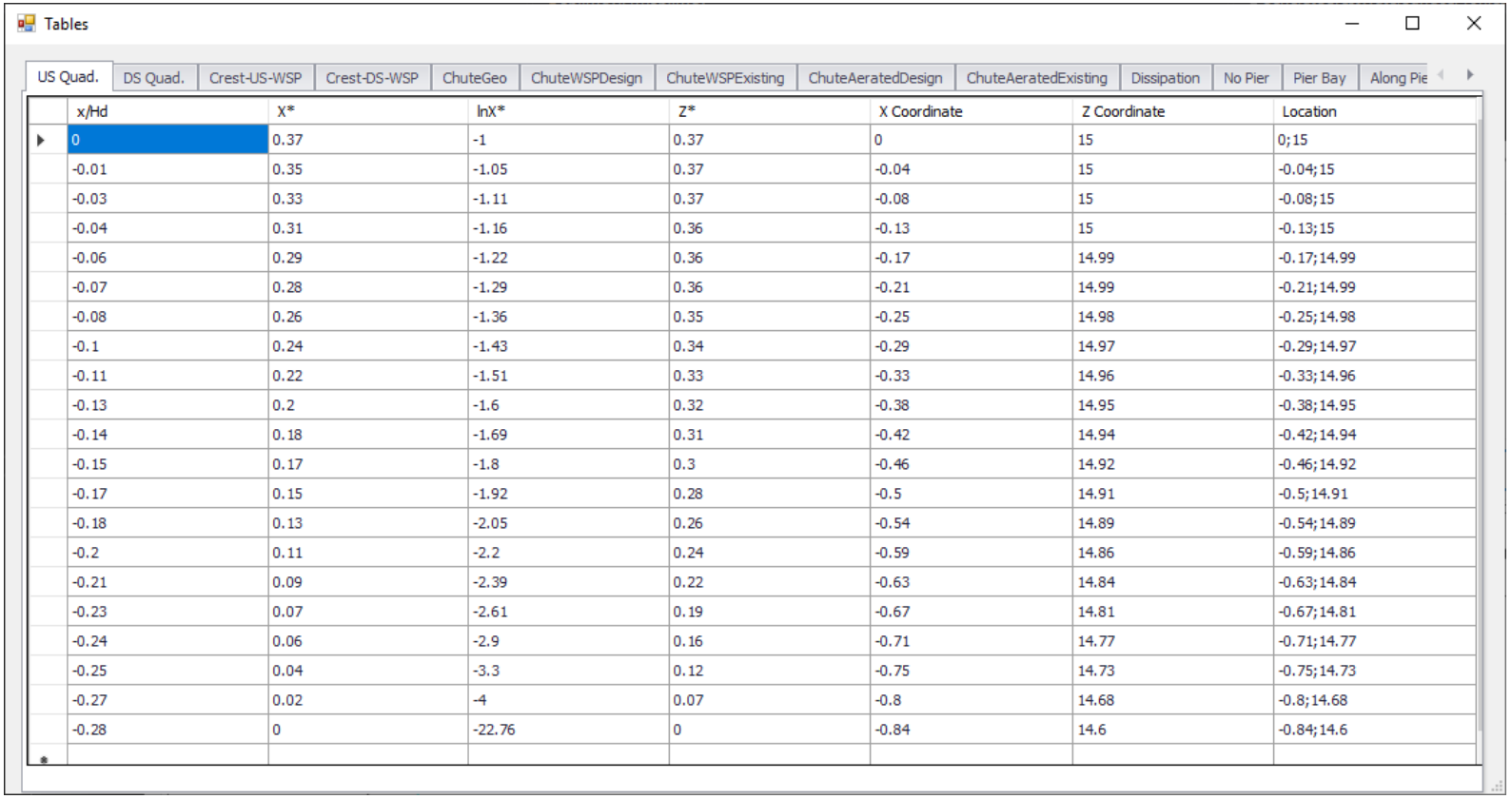
3.4.3. Crest Bed Profile

In order to obtain a smooth bed profile, upstream and downstream quadrants of the crest are divided into 20 and 40 segments, respectively. For each upstream quadrant node, Equations (2.3), (2.4) and (2.5) are applied. For each downstream quadrant node, Equation (2.6) is applied. Output data of an example obtained from the software are given in Table 3.1 and Table 3.2.

3.4.4. Crest Water Surface Profile

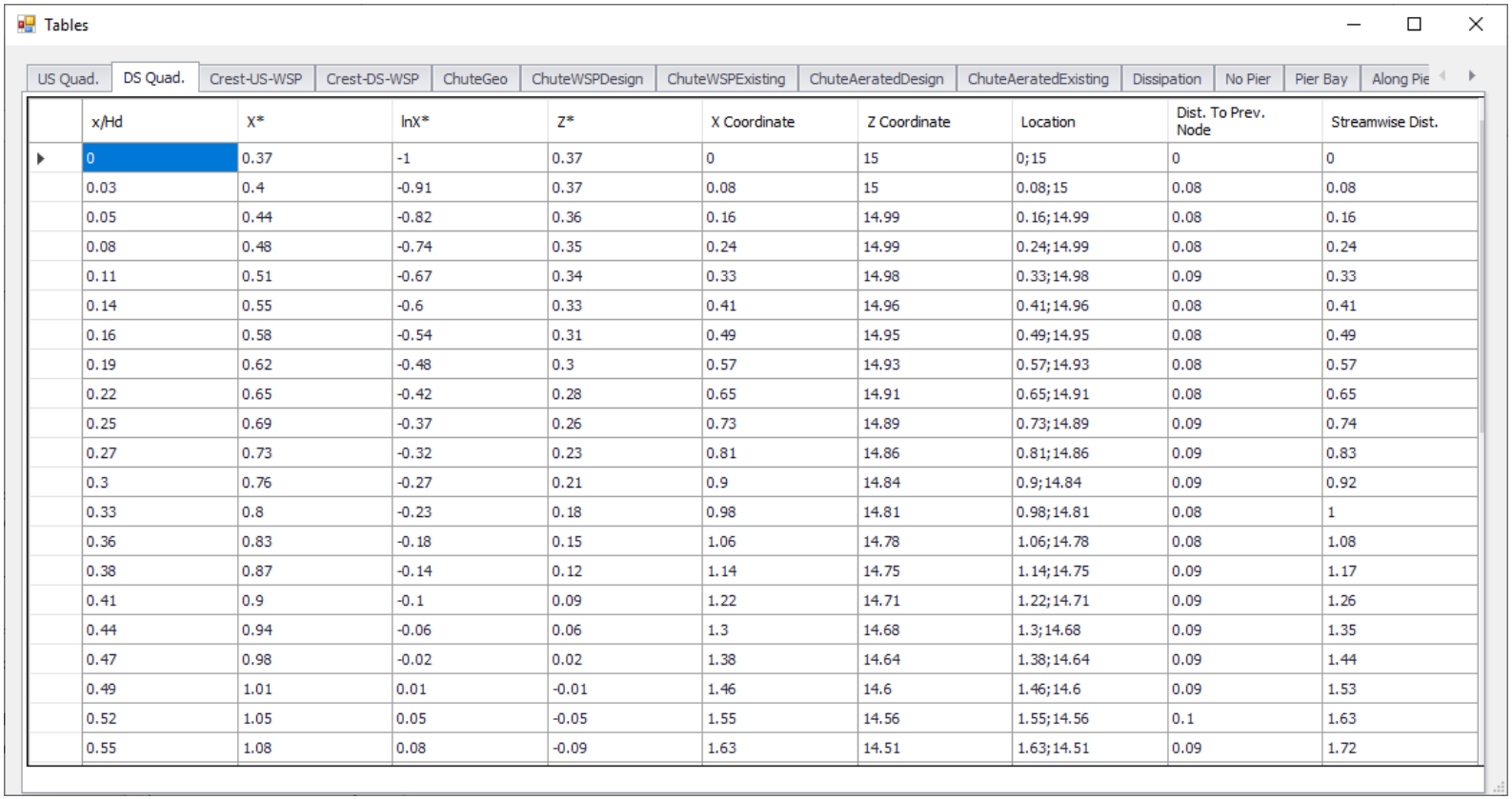
For each node which is previously defined before, water surface elevation is computed by applying Equations (2.12) and (2.13). Output data of an example obtained from the software are given in Table 3.3 and Table 3.4. Visual output is given in Figure 3.5.

Table 3.1. Upstream Quadrant Bed Profile (Software)



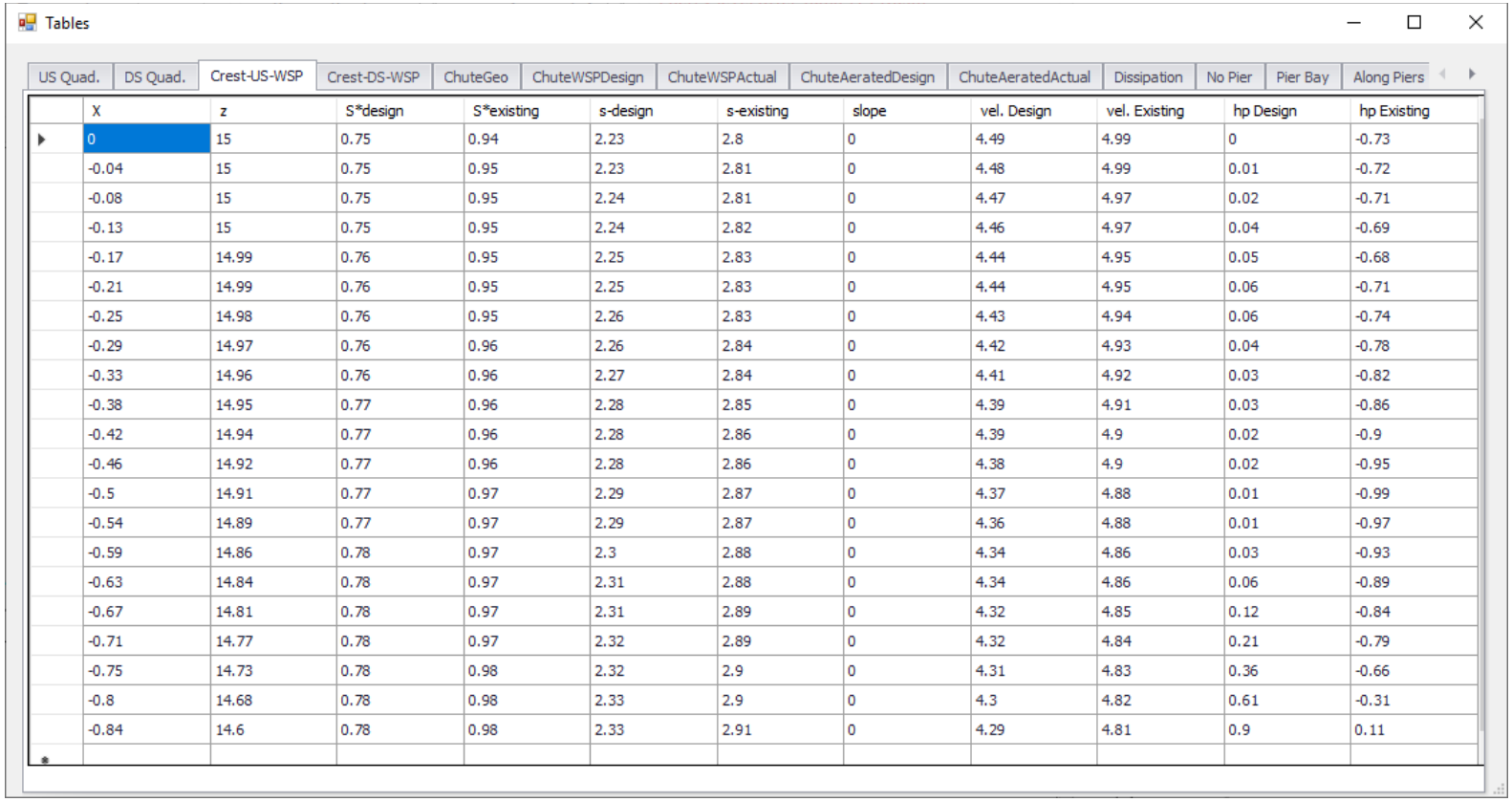
x/Hd	X*	lnX*	Z*	X Coordinate	Z Coordinate	Location
0	0.37	-1	0.37	0	15	0;15
-0.01	0.35	-1.05	0.37	-0.04	15	-0.04;15
-0.03	0.33	-1.11	0.37	-0.08	15	-0.08;15
-0.04	0.31	-1.16	0.36	-0.13	15	-0.13;15
-0.06	0.29	-1.22	0.36	-0.17	14.99	-0.17;14.99
-0.07	0.28	-1.29	0.36	-0.21	14.99	-0.21;14.99
-0.08	0.26	-1.36	0.35	-0.25	14.98	-0.25;14.98
-0.1	0.24	-1.43	0.34	-0.29	14.97	-0.29;14.97
-0.11	0.22	-1.51	0.33	-0.33	14.96	-0.33;14.96
-0.13	0.2	-1.6	0.32	-0.38	14.95	-0.38;14.95
-0.14	0.18	-1.69	0.31	-0.42	14.94	-0.42;14.94
-0.15	0.17	-1.8	0.3	-0.46	14.92	-0.46;14.92
-0.17	0.15	-1.92	0.28	-0.5	14.91	-0.5;14.91
-0.18	0.13	-2.05	0.26	-0.54	14.89	-0.54;14.89
-0.2	0.11	-2.2	0.24	-0.59	14.86	-0.59;14.86
-0.21	0.09	-2.39	0.22	-0.63	14.84	-0.63;14.84
-0.23	0.07	-2.61	0.19	-0.67	14.81	-0.67;14.81
-0.24	0.06	-2.9	0.16	-0.71	14.77	-0.71;14.77
-0.25	0.04	-3.3	0.12	-0.75	14.73	-0.75;14.73
-0.27	0.02	-4	0.07	-0.8	14.68	-0.8;14.68
-0.28	0	-22.76	0	-0.84	14.6	-0.84;14.6

Table 3.2. Downstream Quadrant Bed Profile (Software)



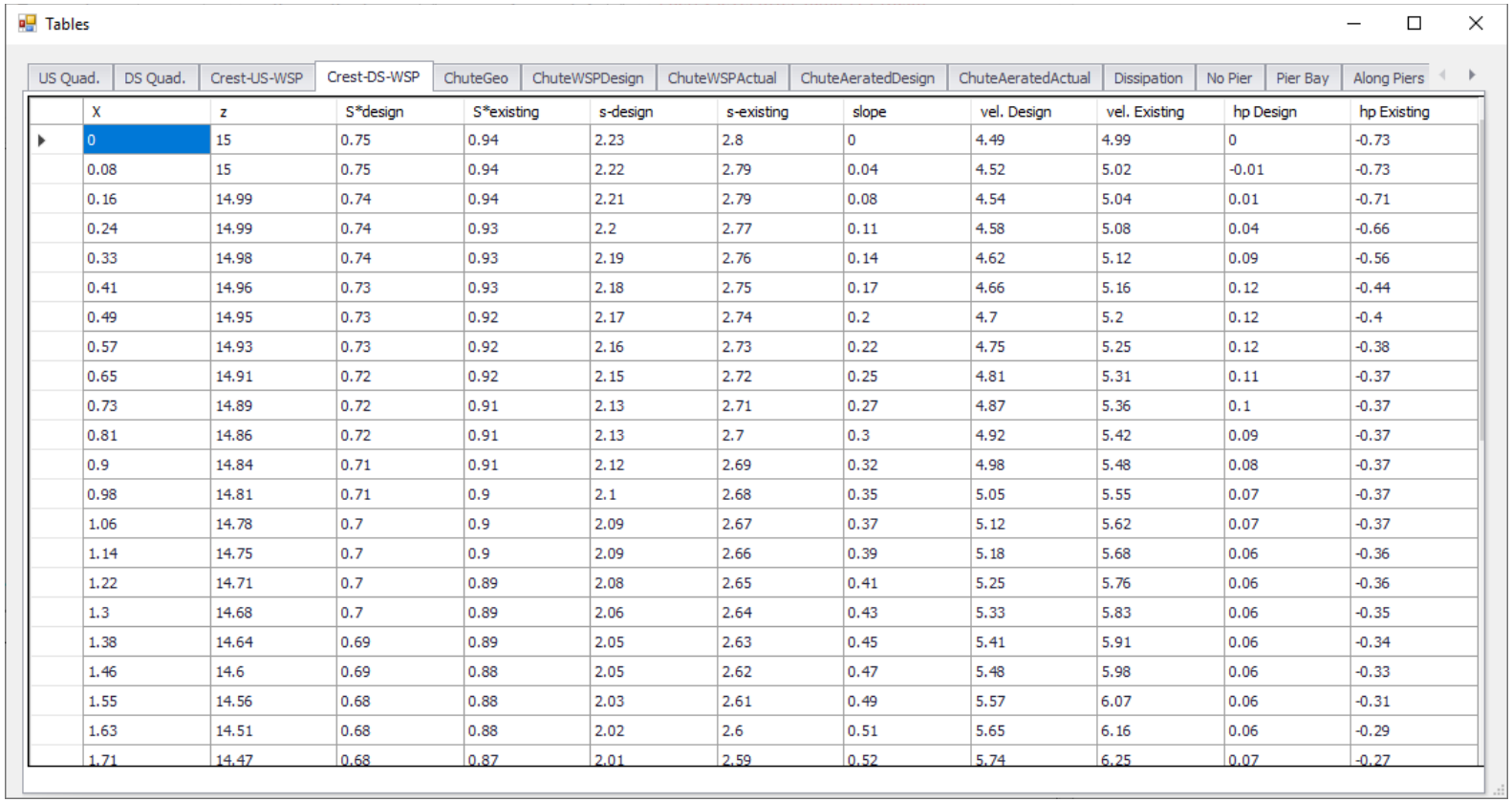
x/Hd	X*	lnX*	Z*	X Coordinate	Z Coordinate	Location	Dist. To Prev. Node	Streamwise Dist.
0	0.37	-1	0.37	0	15	0;15	0	0
0.03	0.4	-0.91	0.37	0.08	15	0.08;15	0.08	0.08
0.05	0.44	-0.82	0.36	0.16	14.99	0.16;14.99	0.08	0.16
0.08	0.48	-0.74	0.35	0.24	14.99	0.24;14.99	0.08	0.24
0.11	0.51	-0.67	0.34	0.33	14.98	0.33;14.98	0.09	0.33
0.14	0.55	-0.6	0.33	0.41	14.96	0.41;14.96	0.08	0.41
0.16	0.58	-0.54	0.31	0.49	14.95	0.49;14.95	0.08	0.49
0.19	0.62	-0.48	0.3	0.57	14.93	0.57;14.93	0.08	0.57
0.22	0.65	-0.42	0.28	0.65	14.91	0.65;14.91	0.08	0.65
0.25	0.69	-0.37	0.26	0.73	14.89	0.73;14.89	0.09	0.74
0.27	0.73	-0.32	0.23	0.81	14.86	0.81;14.86	0.09	0.83
0.3	0.76	-0.27	0.21	0.9	14.84	0.9;14.84	0.09	0.92
0.33	0.8	-0.23	0.18	0.98	14.81	0.98;14.81	0.08	1
0.36	0.83	-0.18	0.15	1.06	14.78	1.06;14.78	0.08	1.08
0.38	0.87	-0.14	0.12	1.14	14.75	1.14;14.75	0.09	1.17
0.41	0.9	-0.1	0.09	1.22	14.71	1.22;14.71	0.09	1.26
0.44	0.94	-0.06	0.06	1.3	14.68	1.3;14.68	0.09	1.35
0.47	0.98	-0.02	0.02	1.38	14.64	1.38;14.64	0.09	1.44
0.49	1.01	0.01	-0.01	1.46	14.6	1.46;14.6	0.09	1.53
0.52	1.05	0.05	-0.05	1.55	14.56	1.55;14.56	0.1	1.63
0.55	1.08	0.08	-0.09	1.63	14.51	1.63;14.51	0.09	1.72

Table 3.3. Crest Upstream Water Surface Profile (Software)



X	z	S*design	S*existing	s-design	s-existing	slope	vel. Design	vel. Existing	hp Design	hp Existing
0	15	0.75	0.94	2.23	2.8	0	4.49	4.99	0	-0.73
-0.04	15	0.75	0.95	2.23	2.81	0	4.48	4.99	0.01	-0.72
-0.08	15	0.75	0.95	2.24	2.81	0	4.47	4.97	0.02	-0.71
-0.13	15	0.75	0.95	2.24	2.82	0	4.46	4.97	0.04	-0.69
-0.17	14.99	0.76	0.95	2.25	2.83	0	4.44	4.95	0.05	-0.68
-0.21	14.99	0.76	0.95	2.25	2.83	0	4.44	4.95	0.06	-0.71
-0.25	14.98	0.76	0.95	2.26	2.83	0	4.43	4.94	0.06	-0.74
-0.29	14.97	0.76	0.96	2.26	2.84	0	4.42	4.93	0.04	-0.78
-0.33	14.96	0.76	0.96	2.27	2.84	0	4.41	4.92	0.03	-0.82
-0.38	14.95	0.77	0.96	2.28	2.85	0	4.39	4.91	0.03	-0.86
-0.42	14.94	0.77	0.96	2.28	2.86	0	4.39	4.9	0.02	-0.9
-0.46	14.92	0.77	0.96	2.28	2.86	0	4.38	4.9	0.02	-0.95
-0.5	14.91	0.77	0.97	2.29	2.87	0	4.37	4.88	0.01	-0.99
-0.54	14.89	0.77	0.97	2.29	2.87	0	4.36	4.88	0.01	-0.97
-0.59	14.86	0.78	0.97	2.3	2.88	0	4.34	4.86	0.03	-0.93
-0.63	14.84	0.78	0.97	2.31	2.88	0	4.34	4.86	0.06	-0.89
-0.67	14.81	0.78	0.97	2.31	2.89	0	4.32	4.85	0.12	-0.84
-0.71	14.77	0.78	0.97	2.32	2.89	0	4.32	4.84	0.21	-0.79
-0.75	14.73	0.78	0.98	2.32	2.9	0	4.31	4.83	0.36	-0.66
-0.8	14.68	0.78	0.98	2.33	2.9	0	4.3	4.82	0.61	-0.31
-0.84	14.6	0.78	0.98	2.33	2.91	0	4.29	4.81	0.9	0.11

Table 3.4. Crest Downstream Water Surface Profile (Software)



X	z	S*design	S*existing	s-design	s-existing	slope	vel. Design	vel. Existing	hp Design	hp Existing
0	15	0.75	0.94	2.23	2.8	0	4.49	4.99	0	-0.73
0.08	15	0.75	0.94	2.22	2.79	0.04	4.52	5.02	-0.01	-0.73
0.16	14.99	0.74	0.94	2.21	2.79	0.08	4.54	5.04	0.01	-0.71
0.24	14.99	0.74	0.93	2.2	2.77	0.11	4.58	5.08	0.04	-0.66
0.33	14.98	0.74	0.93	2.19	2.76	0.14	4.62	5.12	0.09	-0.56
0.41	14.96	0.73	0.93	2.18	2.75	0.17	4.66	5.16	0.12	-0.44
0.49	14.95	0.73	0.92	2.17	2.74	0.2	4.7	5.2	0.12	-0.4
0.57	14.93	0.73	0.92	2.16	2.73	0.22	4.75	5.25	0.12	-0.38
0.65	14.91	0.72	0.92	2.15	2.72	0.25	4.81	5.31	0.11	-0.37
0.73	14.89	0.72	0.91	2.13	2.71	0.27	4.87	5.36	0.1	-0.37
0.81	14.86	0.72	0.91	2.13	2.7	0.3	4.92	5.42	0.09	-0.37
0.9	14.84	0.71	0.91	2.12	2.69	0.32	4.98	5.48	0.08	-0.37
0.98	14.81	0.71	0.9	2.1	2.68	0.35	5.05	5.55	0.07	-0.37
1.06	14.78	0.7	0.9	2.09	2.67	0.37	5.12	5.62	0.07	-0.37
1.14	14.75	0.7	0.9	2.09	2.66	0.39	5.18	5.68	0.06	-0.36
1.22	14.71	0.7	0.89	2.08	2.65	0.41	5.25	5.76	0.06	-0.36
1.3	14.68	0.7	0.89	2.06	2.64	0.43	5.33	5.83	0.06	-0.35
1.38	14.64	0.69	0.89	2.05	2.63	0.45	5.41	5.91	0.06	-0.34
1.46	14.6	0.69	0.88	2.05	2.62	0.47	5.48	5.98	0.06	-0.33
1.55	14.56	0.68	0.88	2.03	2.61	0.49	5.57	6.07	0.06	-0.31
1.63	14.51	0.68	0.88	2.02	2.6	0.51	5.65	6.16	0.06	-0.29
1.71	14.47	0.68	0.87	2.01	2.59	0.52	5.74	6.25	0.07	-0.27

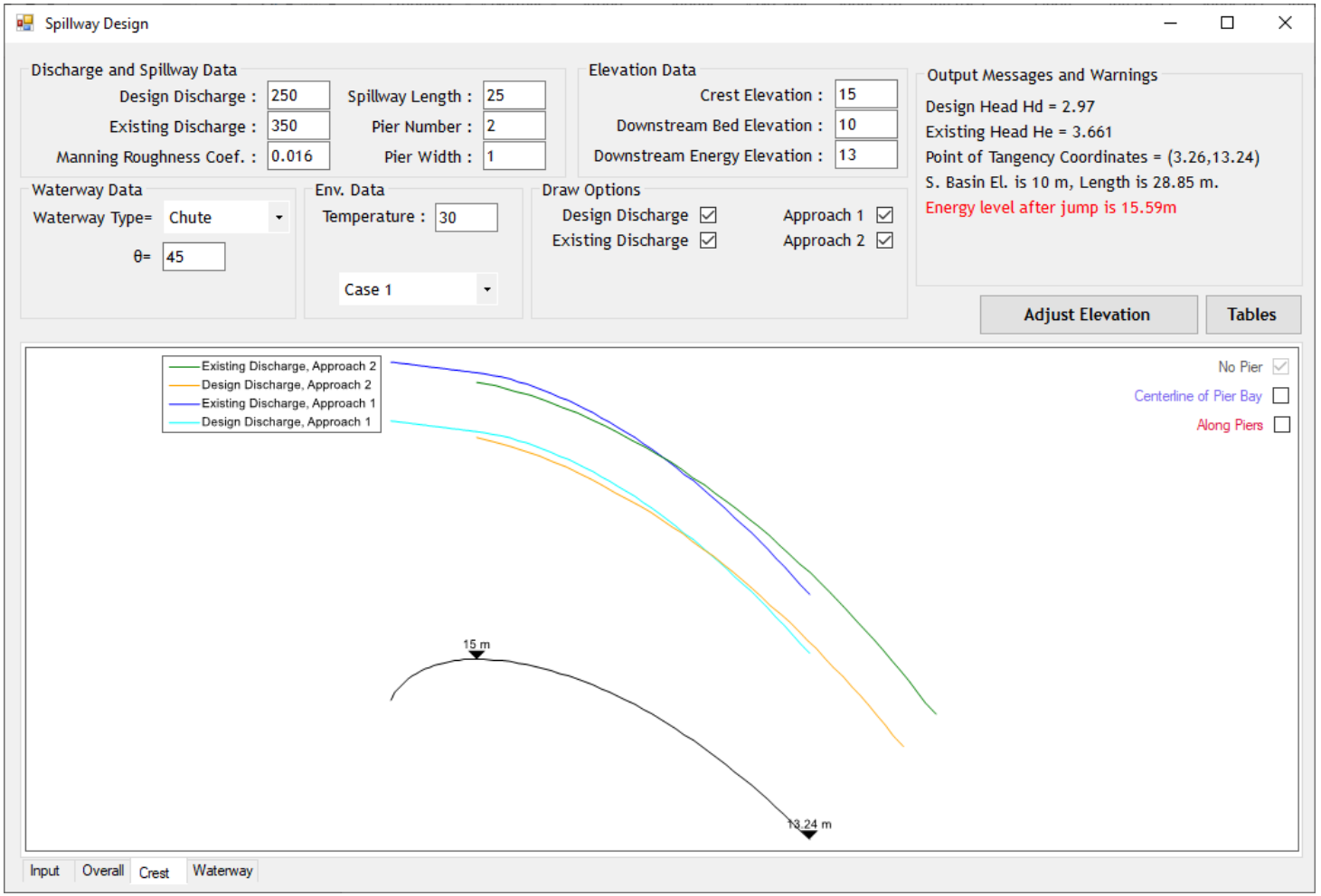


Figure 3.5. Crest Water Surface Profile

3.4.5. Crest Bottom Pressures

In order to determine the pressure head along the crest under a specific head ratio, pressure distribution charts which are shown in Figure 2.3, Figure 2.4 and Figure 2.5 are digitized by dividing the profile into small segments. Equations for each segment (3.1) through (3.33) are implemented into the software. Therefore, it is possible to calculate a pressure at any location as long as the head ratio is between 0.5 and 1.33.

In the following equations, h is the relative pressure, $h = h_p/H_d$, where h_p is the pressure head, i.e., water depth. Bottom pressure profiles, when there is no pier on the crest, are divided into 2, 3, 5 sub-profiles for head ratio of 0.5, 1.0 and 1.33, respectively.

For $\chi = 0.50$,

$$h = 536.84\,X^6 - 12.172\,X^5 - 34.376\,X^4 - 0.217\,X^3 + 1.4794\,X^2 - 0.3638\,X + 0.1999 \quad \text{for } X < 0.20 \qquad (3.1)$$

$$h = 0.9572\,X^5 - 3.4197\,X^4 + 4.3\,X^3 - 2.193\,X^2 + 0.2682\,X + 0.1662 \quad \text{for } X \geq 0.20 \qquad (3.2)$$

For $\chi = 1.00$,

$$h = 537.29\,X^4 + 258\,X^3 + 43.623\,X^2 + 3.0826\,X + 0.0861 \quad \text{for } X < -0.10 \qquad (3.3)$$

$$h = 449.2\,X^5 - 271.05\,X^4 + 22.06\,X^3 + 4.415\,X^2 - 0.1955\,X \quad \text{for } -0.10 \leq X \leq 0.17 \qquad (3.4)$$

$$h = 0.6563\,X^5 - 2.2952\,X^4 + 2.7051\,X^3 - 1.1496\,X^2 + 0.0862\,X + 0.0486 \quad \text{for } X \geq 0.17 \qquad (3.5)$$

For $\chi = 1.33$,

$$h = 130.92\,X^2 + 60.483\,X + 6.5576 \quad \text{for } X < -0.245 \qquad (3.6)$$

$$h = -X - 0.645 \quad \text{for } -0.245 \leq X < -0.17 \qquad (3.7)$$

$$h = -1.9418\,X^2 + 0.8433\,X - 0.28 \qquad \text{for } -0.17 > X \leq -0.05 \tag{3.8}$$

$$h = 23.479\,X^3 + 4.5254\,X^2 - 0.1841\,X - 0.3481 \quad \text{for } -0.05 > X \leq 0.14 \tag{3.9}$$

$$h = -4.5413\,X^6 + 21.475\,X^5 - 39.338\,X^4 + 35.031\,X^3 - 15.547\,X^2 + 3.3403\,X - 0.465 \qquad \text{for } X > 0.14 \tag{3.10}$$

Bottom pressure profiles along the centerline of pier bay, are divided into 3, 4, 5 sub-profiles for head ratio of 0.5, 1.0 and 1.33, respectively.

For $\chi = 0.50$,

$$h = 31.037\,X^2 + 12.078\,X + 1.5164 \qquad \text{for } X \leq -0.20 \tag{3.11}$$

$$h = -2.0404\,X^4 - 0.4116\,X^3 + 0.9517\,X^2 - 0.3469\,X + 0.2343 \qquad \text{for } -0.20 < X \leq 0.50 \tag{3.12}$$

$$h = 0.0422\,X^2 - 0.1392\,X + 0.177 \qquad \text{for } 0.50 < X \tag{3.13}$$

For $\chi = 1.00$,

$$h = 39.738\,X^2 + 15.569\,X + 1.7342 \qquad \text{for } X \leq -0.20 \tag{3.14}$$

$$h = -30.078\,X^3 - 10.529\,X^2 - 1.3526\,X + 0.12 \qquad \text{for } -0.20 < X \leq 0 \tag{3.15}$$

$$h = 0.0422\,X^2 - 0.1392\,X + 0.177 \qquad \text{for } 0.50 < X \tag{3.16}$$

$$h = -0.0065\,X + 0.0588 \qquad \text{for } X > 0.5 \tag{3.17}$$

For $\chi = 1.33$,

$$h = -9.625\,X - 2.41 \qquad \text{for } X \leq -0.24 \tag{3.18}$$

$$h = -1.25\,X - 0.40 \qquad \text{for } -0.24 < X \leq -0.20 \tag{3.19}$$

$$h = -4.5X^2 - 1.1\,X - 0.19 \qquad \text{for } -0.20 < X \leq 0 \tag{3.20}$$

$$h = -2.0455\,X^2 + 0.9\,X - 0.19 \qquad \text{for } 0 < X \leq 0.32 \tag{3.21}$$

$$h = 0.2356\,X^4 - 1.0135\,X^3 + 1.5085\,X^2 - 0.7762\,X + 0.0146 \qquad \text{for } X > 0.32 \tag{3.22}$$

Bottom pressure profiles along piers are divided into 3, 4, 4 sub-profiles for head ratio of 0.5, 1.0 and 1.33, respectively.

For $\chi = 0.50$,

$$h = 8.89\,X^2 + 1.213\,X + 0.2331 \qquad \text{for } X \leq -0.15 \tag{3.23}$$

$$h = 17.555\,X^4 - 4.8743\,X^3 - 0.6634\,X^2 - 0.664\,X + 0.2292 \qquad \text{for } -0.15 < X \leq 0.30 \tag{3.24}$$

$$h = 0.1185\,X^2 - 0.2411\,X + 0.2268 \qquad \text{for } X > 0.30 \tag{3.25}$$

For $\chi = 1.00$,

$$h = -27.983\,X^2 - 20.263\,X - 2.7446 \qquad \text{for } X < -0.18 \tag{3.26}$$

$$h = -0.4\,X - 0.082 \qquad \text{for } -0.18 \leq X < -0.13 \tag{3.27}$$

$$h = 0.6515\,X + 0.0549 \qquad \text{for } -0.13 \leq X < 0.10 \tag{3.28}$$

$$h = 0.1123\,X^3 - 0.2344\,X^2 + 0.1093\,X + 0.1105 \qquad \text{for } X \geq 0.10 \tag{3.29}$$

For $\chi = 1.33$,

$$h = -14.598\,X - 3.4743 \qquad \text{for } X < -0.20 \tag{3.30}$$

$$h = 11.043\,X^2 + 3.7034\,X - 0.3105 \qquad \text{for } -0.20 \leq X < 0 \tag{3.31}$$

$$h = -33.333\,X^5 + 33.333\,X^4 - 2.5\,X^3 - 6.5833\,X^2 + 2.3533\,X - 0.31 \qquad \text{for } 0 \leq X < 0.50 \tag{3.32}$$

$$h = 0.4272\,X^3 - 1.1635\,X^2 + 1.1331\,X - 0.3806 \qquad \text{for } X \geq 0.50 \qquad (3.33)$$

Digitized pressure distribution charts are given in Figure 3.6, Figure 3.7, and Figure 3.8 in which relative distance is x/H_d. Data of the charts are given in tabular form in Table 3.5, Table 3.6, and Table 3.7. On Crest tab page, bottom pressure distributions are visualized. Depending on the pier number input, different pressure distribution profiles are obtained. The user can plot different variations using the checkboxes.

In Figure 3.9 and Figure 3.10, crest pressures are displayed considering a case which has an existing head greater than design head with and without piers, respectively. In Figure 3.11 and Figure 3.12, crest pressures are displayed considering a case in which design head is greater than existing head with and without piers, respectively.

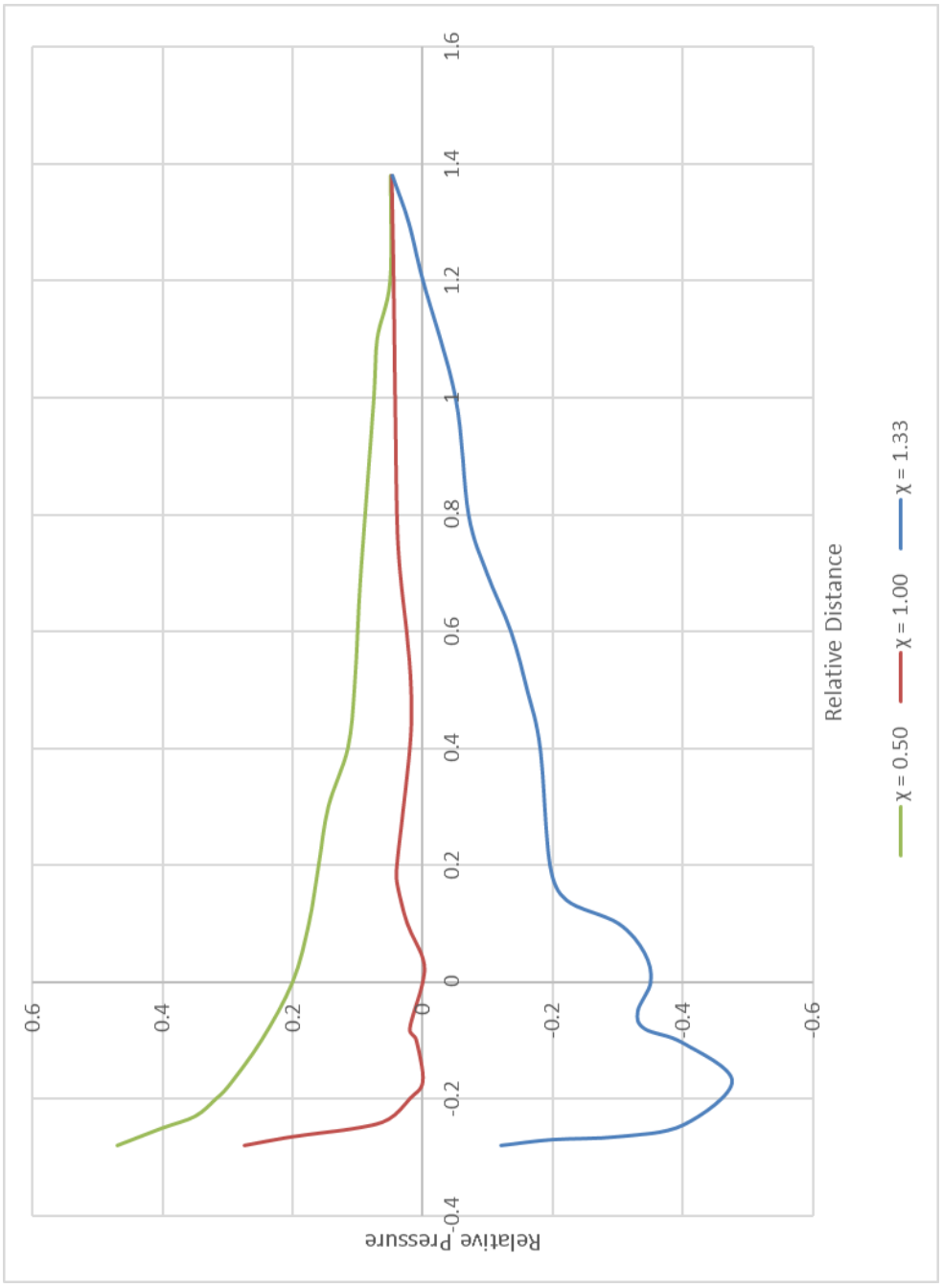


Figure 3.6. Digitized Pressure Table for High Overflow Spillways (a) No Pier

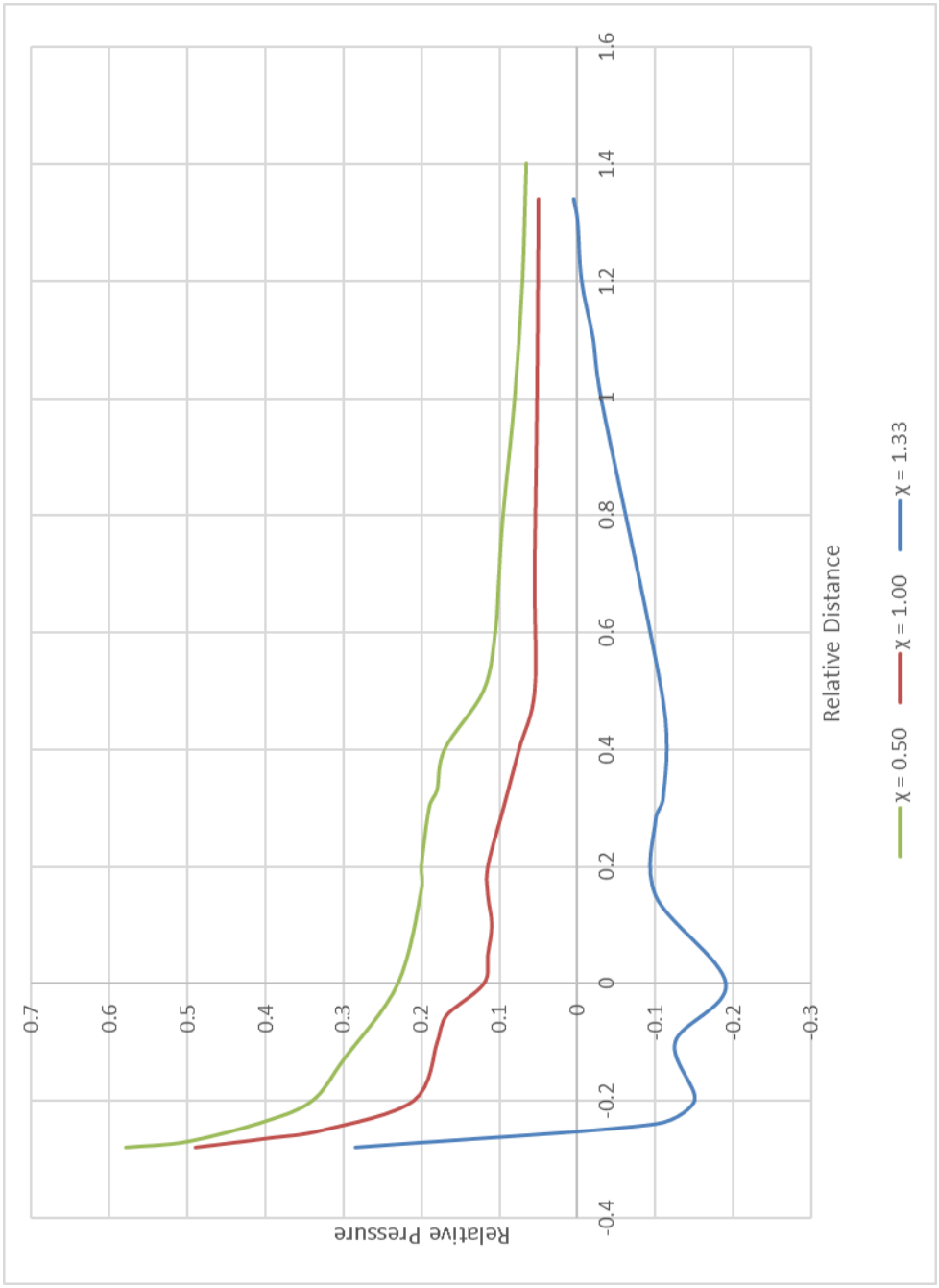


Figure 3.7. Digitized Pressure Table for High Overflow Spillways (b) Along centerline of pier bay

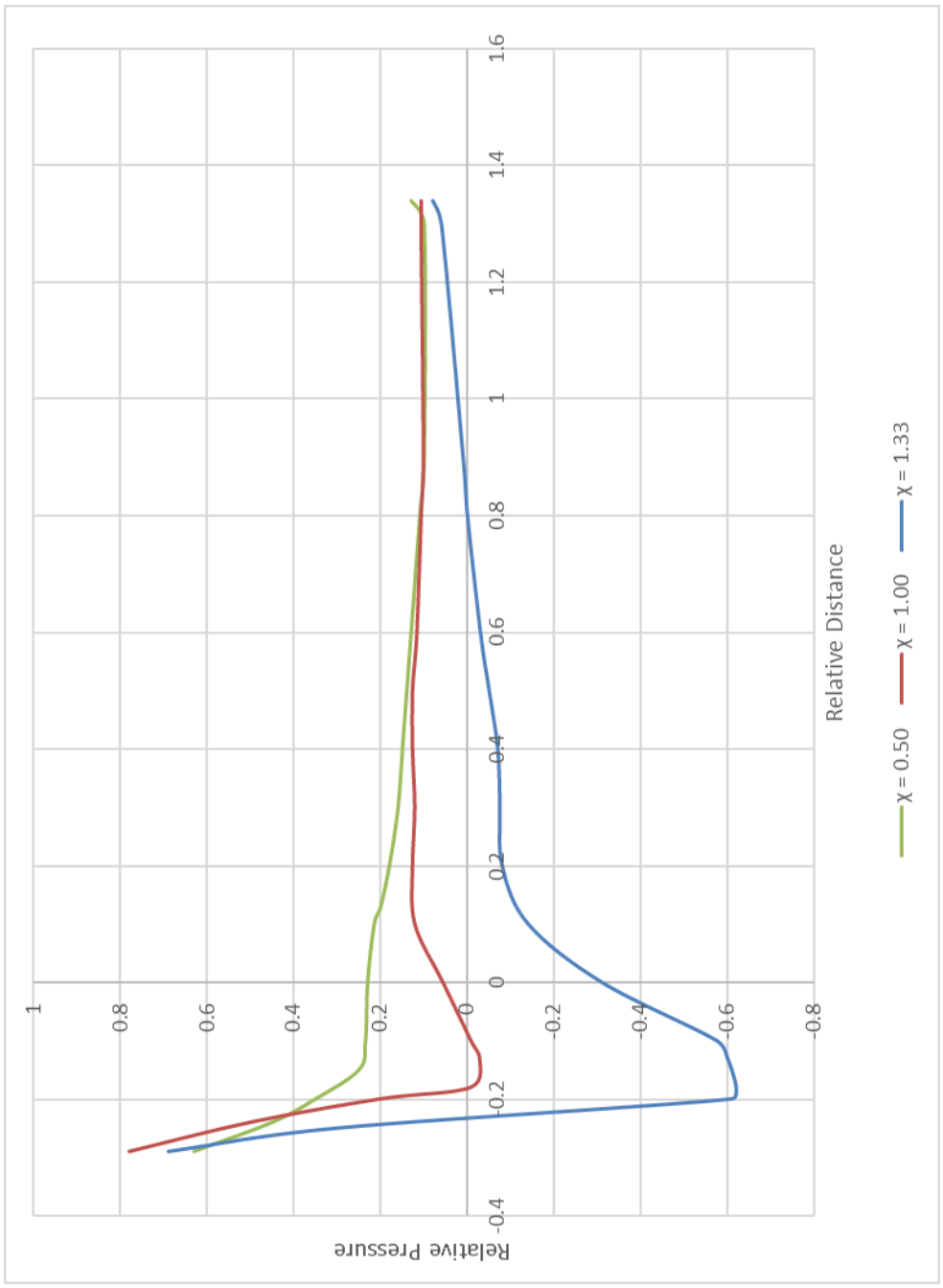
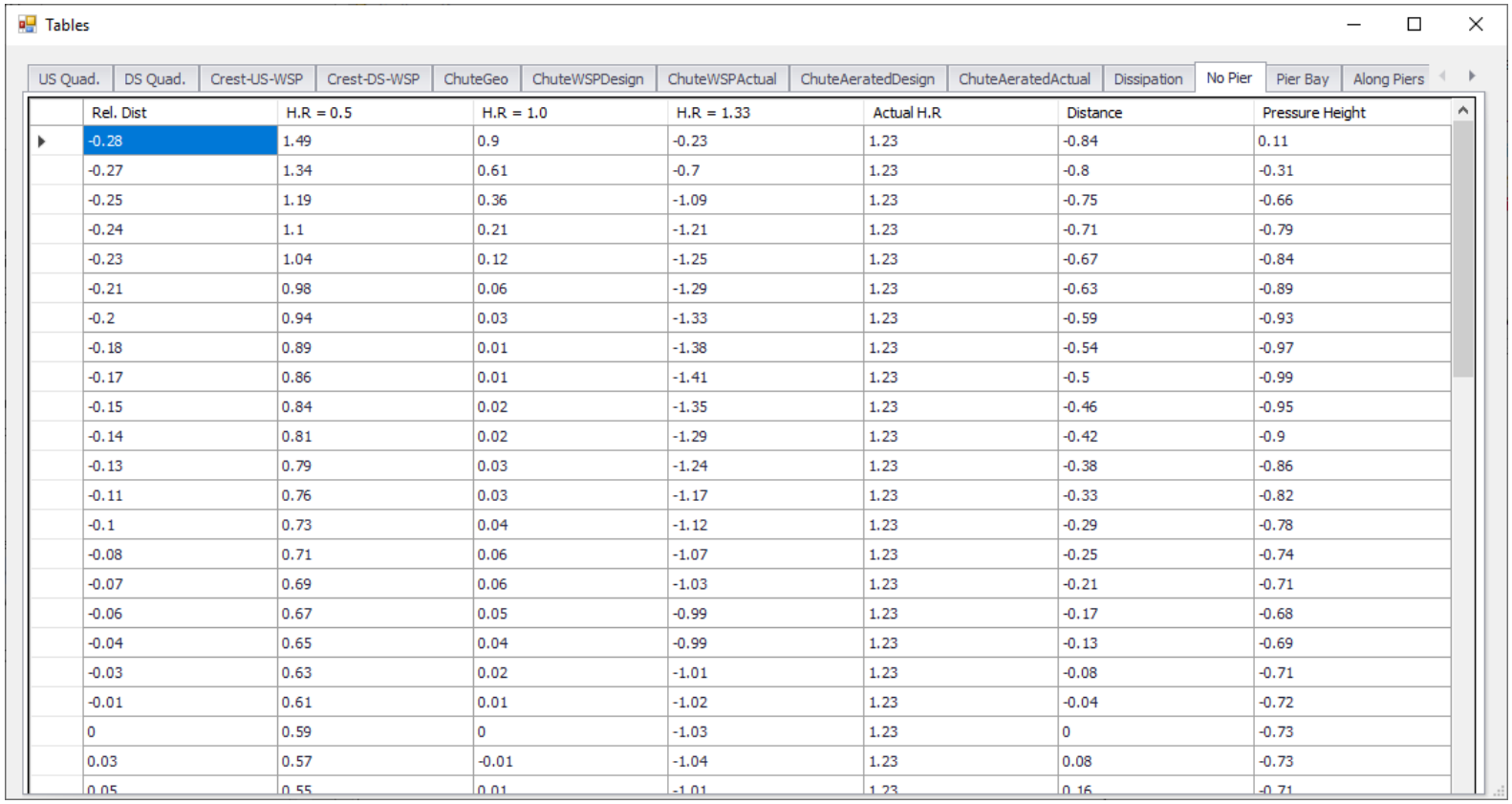


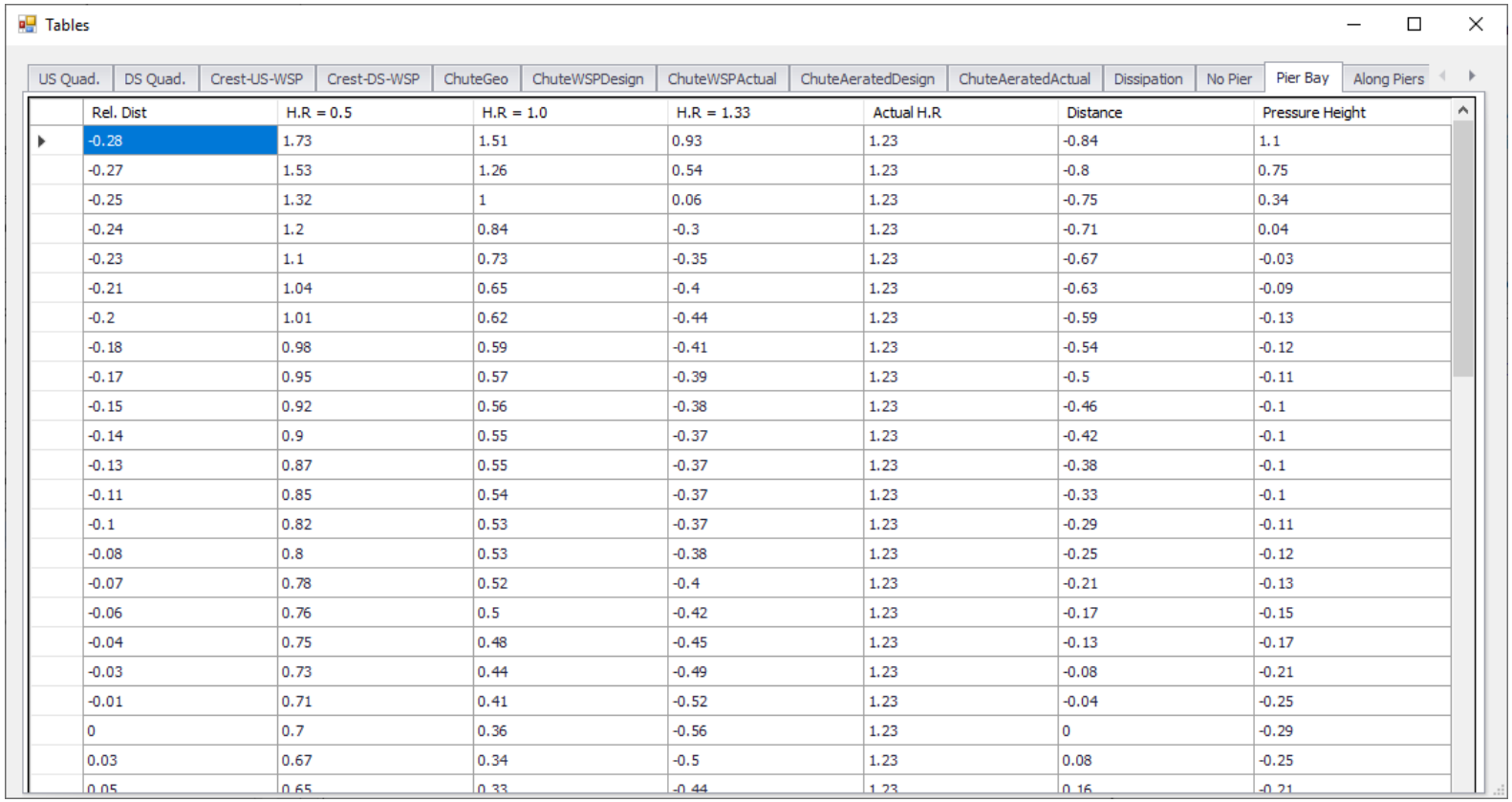
Figure 3.8. Digitized Pressure Table for High Overflow Spillways (c) Along piers

Table 3.5. Crest Pressures Tabular Output (a) No Pier



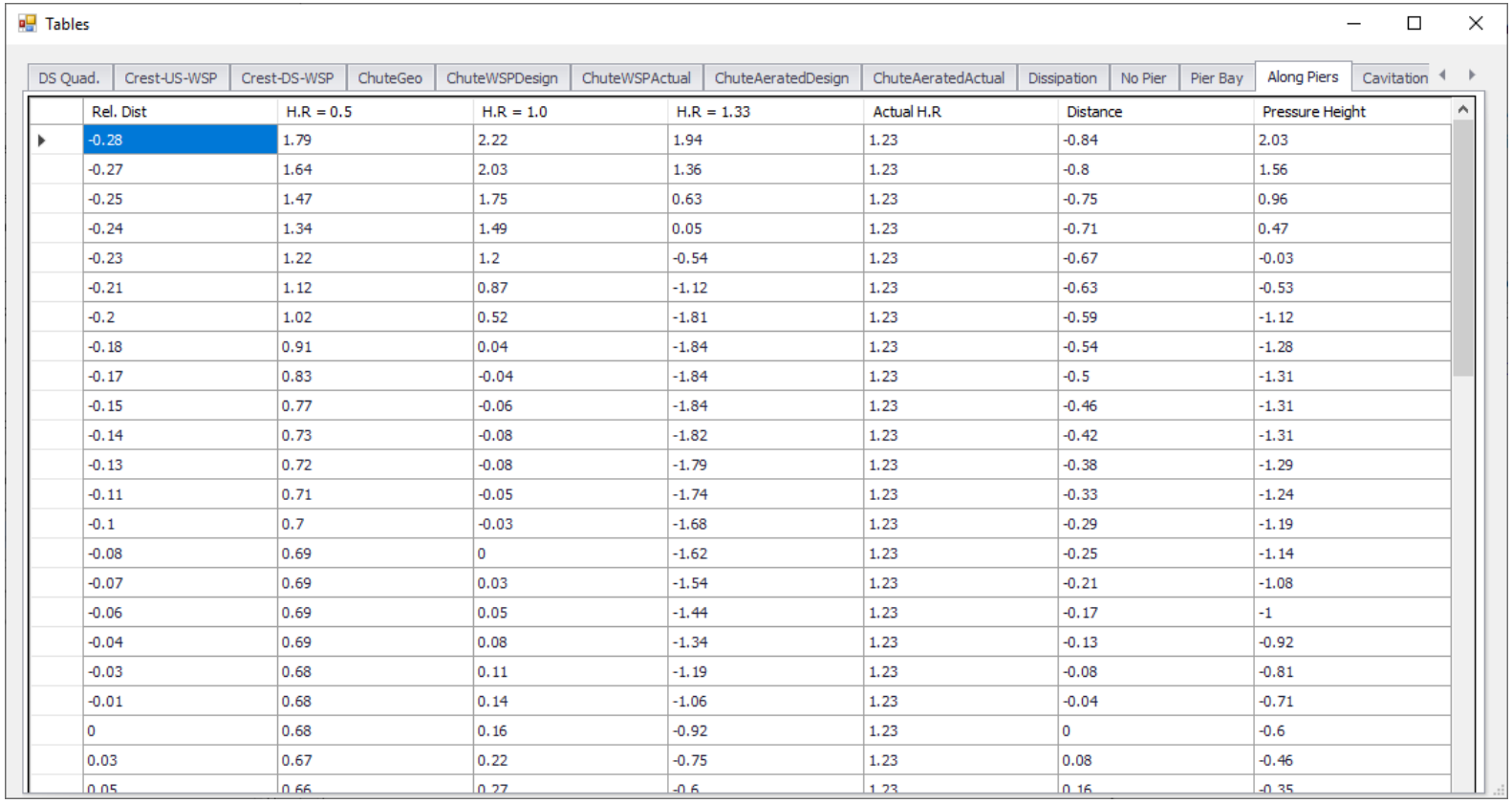
Rel. Dist	H.R = 0.5	H.R = 1.0	H.R = 1.33	Actual H.R	Distance	Pressure Height
-0.28	1.49	0.9	-0.23	1.23	-0.84	0.11
-0.27	1.34	0.61	-0.7	1.23	-0.8	-0.31
-0.25	1.19	0.36	-1.09	1.23	-0.75	-0.66
-0.24	1.1	0.21	-1.21	1.23	-0.71	-0.79
-0.23	1.04	0.12	-1.25	1.23	-0.67	-0.84
-0.21	0.98	0.06	-1.29	1.23	-0.63	-0.89
-0.2	0.94	0.03	-1.33	1.23	-0.59	-0.93
-0.18	0.89	0.01	-1.38	1.23	-0.54	-0.97
-0.17	0.86	0.01	-1.41	1.23	-0.5	-0.99
-0.15	0.84	0.02	-1.35	1.23	-0.46	-0.95
-0.14	0.81	0.02	-1.29	1.23	-0.42	-0.9
-0.13	0.79	0.03	-1.24	1.23	-0.38	-0.86
-0.11	0.76	0.03	-1.17	1.23	-0.33	-0.82
-0.1	0.73	0.04	-1.12	1.23	-0.29	-0.78
-0.08	0.71	0.06	-1.07	1.23	-0.25	-0.74
-0.07	0.69	0.06	-1.03	1.23	-0.21	-0.71
-0.06	0.67	0.05	-0.99	1.23	-0.17	-0.68
-0.04	0.65	0.04	-0.99	1.23	-0.13	-0.69
-0.03	0.63	0.02	-1.01	1.23	-0.08	-0.71
-0.01	0.61	0.01	-1.02	1.23	-0.04	-0.72
0	0.59	0	-1.03	1.23	0	-0.73
0.03	0.57	-0.01	-1.04	1.23	0.08	-0.73
0.05	0.55	0.01	-1.01	1.23	0.16	-0.71

Table 3.6. Crest Pressures Tabular Output (b) Along Centerline of Pier Bay



Rel. Dist	H.R = 0.5	H.R = 1.0	H.R = 1.33	Actual H.R	Distance	Pressure Height
-0.28	1.73	1.51	0.93	1.23	-0.84	1.1
-0.27	1.53	1.26	0.54	1.23	-0.8	0.75
-0.25	1.32	1	0.06	1.23	-0.75	0.34
-0.24	1.2	0.84	-0.3	1.23	-0.71	0.04
-0.23	1.1	0.73	-0.35	1.23	-0.67	-0.03
-0.21	1.04	0.65	-0.4	1.23	-0.63	-0.09
-0.2	1.01	0.62	-0.44	1.23	-0.59	-0.13
-0.18	0.98	0.59	-0.41	1.23	-0.54	-0.12
-0.17	0.95	0.57	-0.39	1.23	-0.5	-0.11
-0.15	0.92	0.56	-0.38	1.23	-0.46	-0.1
-0.14	0.9	0.55	-0.37	1.23	-0.42	-0.1
-0.13	0.87	0.55	-0.37	1.23	-0.38	-0.1
-0.11	0.85	0.54	-0.37	1.23	-0.33	-0.1
-0.1	0.82	0.53	-0.37	1.23	-0.29	-0.11
-0.08	0.8	0.53	-0.38	1.23	-0.25	-0.12
-0.07	0.78	0.52	-0.4	1.23	-0.21	-0.13
-0.06	0.76	0.5	-0.42	1.23	-0.17	-0.15
-0.04	0.75	0.48	-0.45	1.23	-0.13	-0.17
-0.03	0.73	0.44	-0.49	1.23	-0.08	-0.21
-0.01	0.71	0.41	-0.52	1.23	-0.04	-0.25
0	0.7	0.36	-0.56	1.23	0	-0.29
0.03	0.67	0.34	-0.5	1.23	0.08	-0.25
0.05	0.65	0.33	-0.44	1.23	0.16	-0.21

Table 3.7. Crest Pressures Tabular Output (c) Along Piers



Rel. Dist	H.R = 0.5	H.R = 1.0	H.R = 1.33	Actual H.R	Distance	Pressure Height
-0.28	1.79	2.22	1.94	1.23	-0.84	2.03
-0.27	1.64	2.03	1.36	1.23	-0.8	1.56
-0.25	1.47	1.75	0.63	1.23	-0.75	0.96
-0.24	1.34	1.49	0.05	1.23	-0.71	0.47
-0.23	1.22	1.2	-0.54	1.23	-0.67	-0.03
-0.21	1.12	0.87	-1.12	1.23	-0.63	-0.53
-0.2	1.02	0.52	-1.81	1.23	-0.59	-1.12
-0.18	0.91	0.04	-1.84	1.23	-0.54	-1.28
-0.17	0.83	-0.04	-1.84	1.23	-0.5	-1.31
-0.15	0.77	-0.06	-1.84	1.23	-0.46	-1.31
-0.14	0.73	-0.08	-1.82	1.23	-0.42	-1.31
-0.13	0.72	-0.08	-1.79	1.23	-0.38	-1.29
-0.11	0.71	-0.05	-1.74	1.23	-0.33	-1.24
-0.1	0.7	-0.03	-1.68	1.23	-0.29	-1.19
-0.08	0.69	0	-1.62	1.23	-0.25	-1.14
-0.07	0.69	0.03	-1.54	1.23	-0.21	-1.08
-0.06	0.69	0.05	-1.44	1.23	-0.17	-1
-0.04	0.69	0.08	-1.34	1.23	-0.13	-0.92
-0.03	0.68	0.11	-1.19	1.23	-0.08	-0.81
-0.01	0.68	0.14	-1.06	1.23	-0.04	-0.71
0	0.68	0.16	-0.92	1.23	0	-0.6
0.03	0.67	0.22	-0.75	1.23	0.08	-0.46
0.05	0.66	0.27	-0.6	1.23	0.16	-0.35

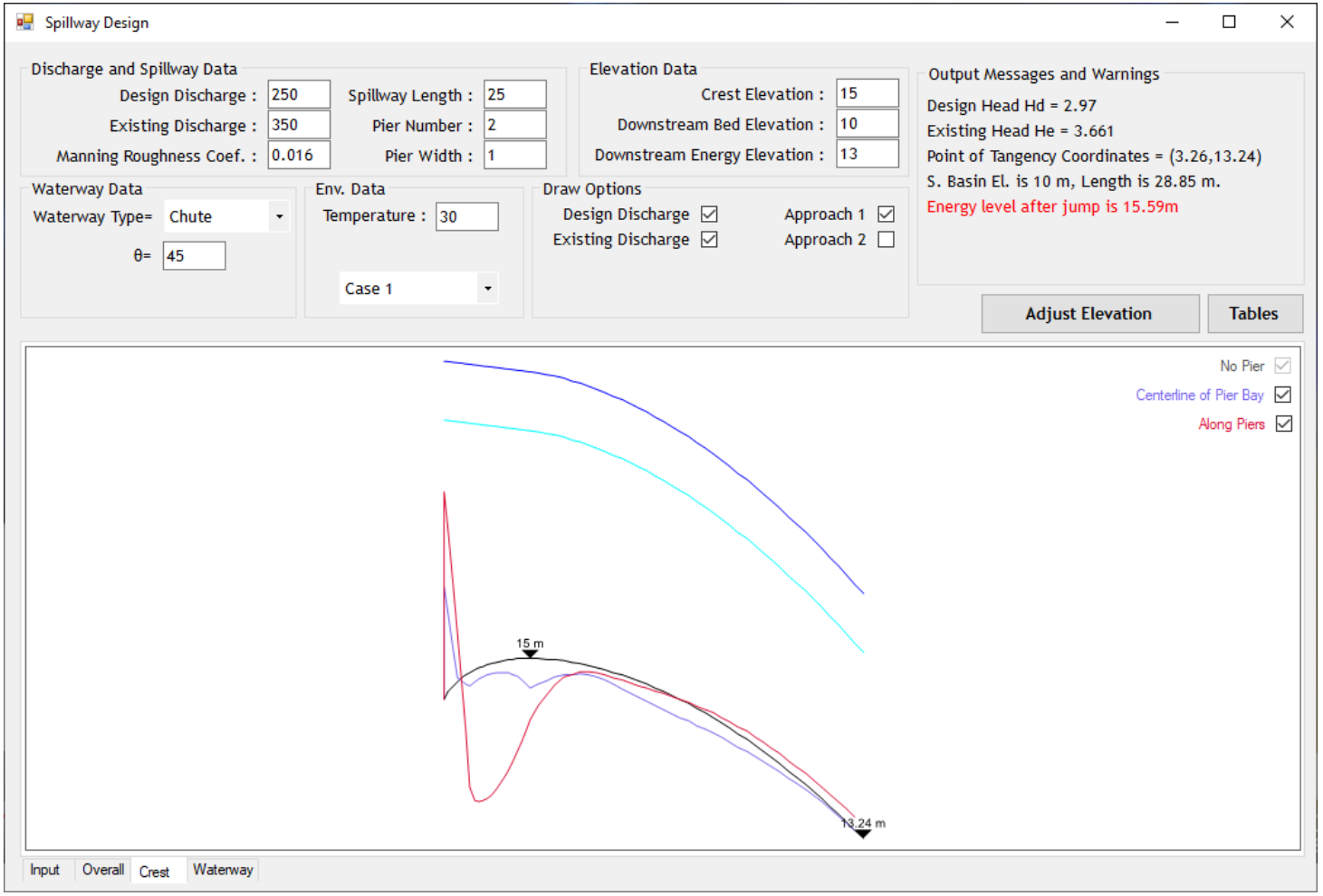


Figure 3.9. Crest Pressures with Piers, $\chi > 1.00$

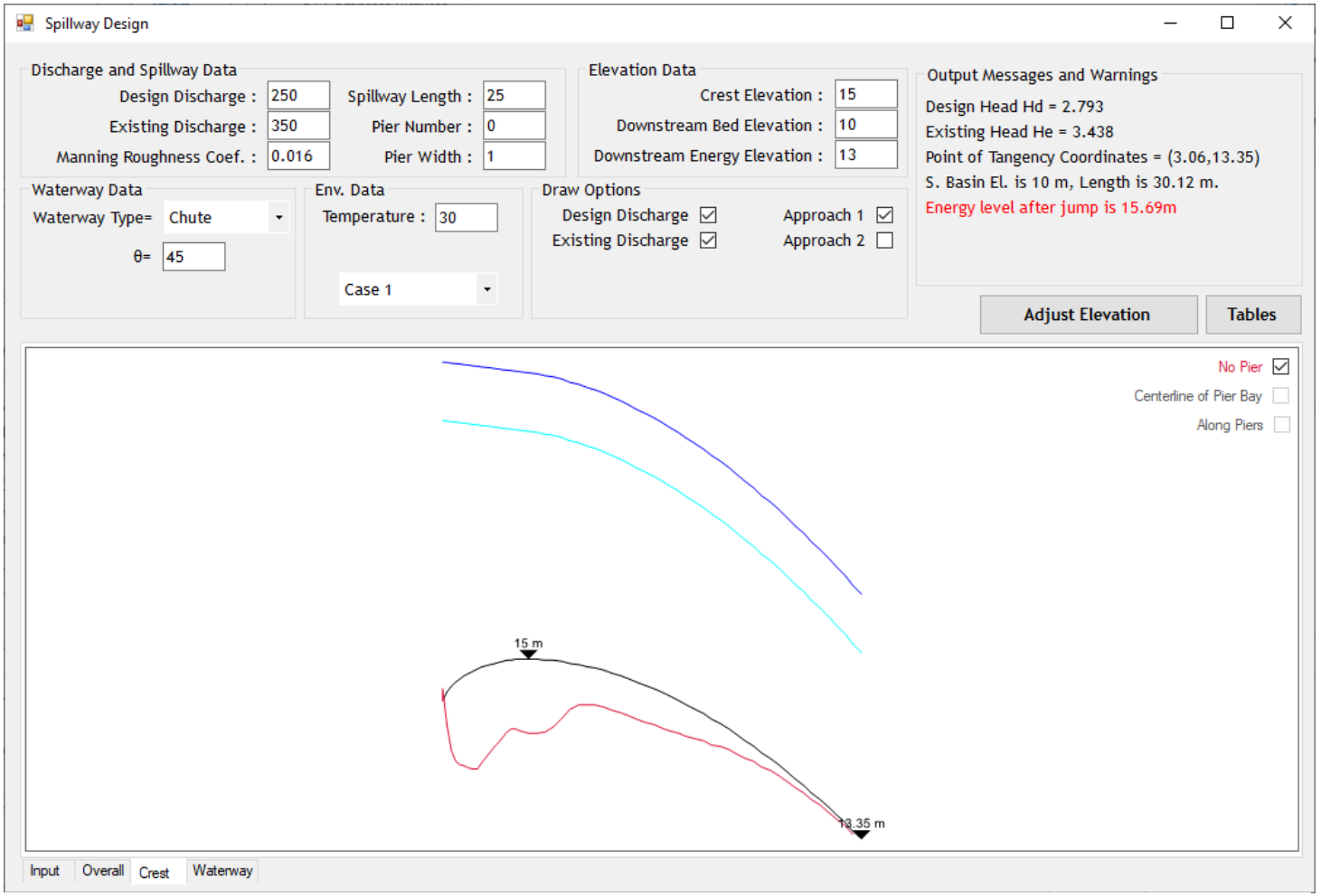


Figure 3.10. Crest Pressures without Piers, $\chi > 1.00$

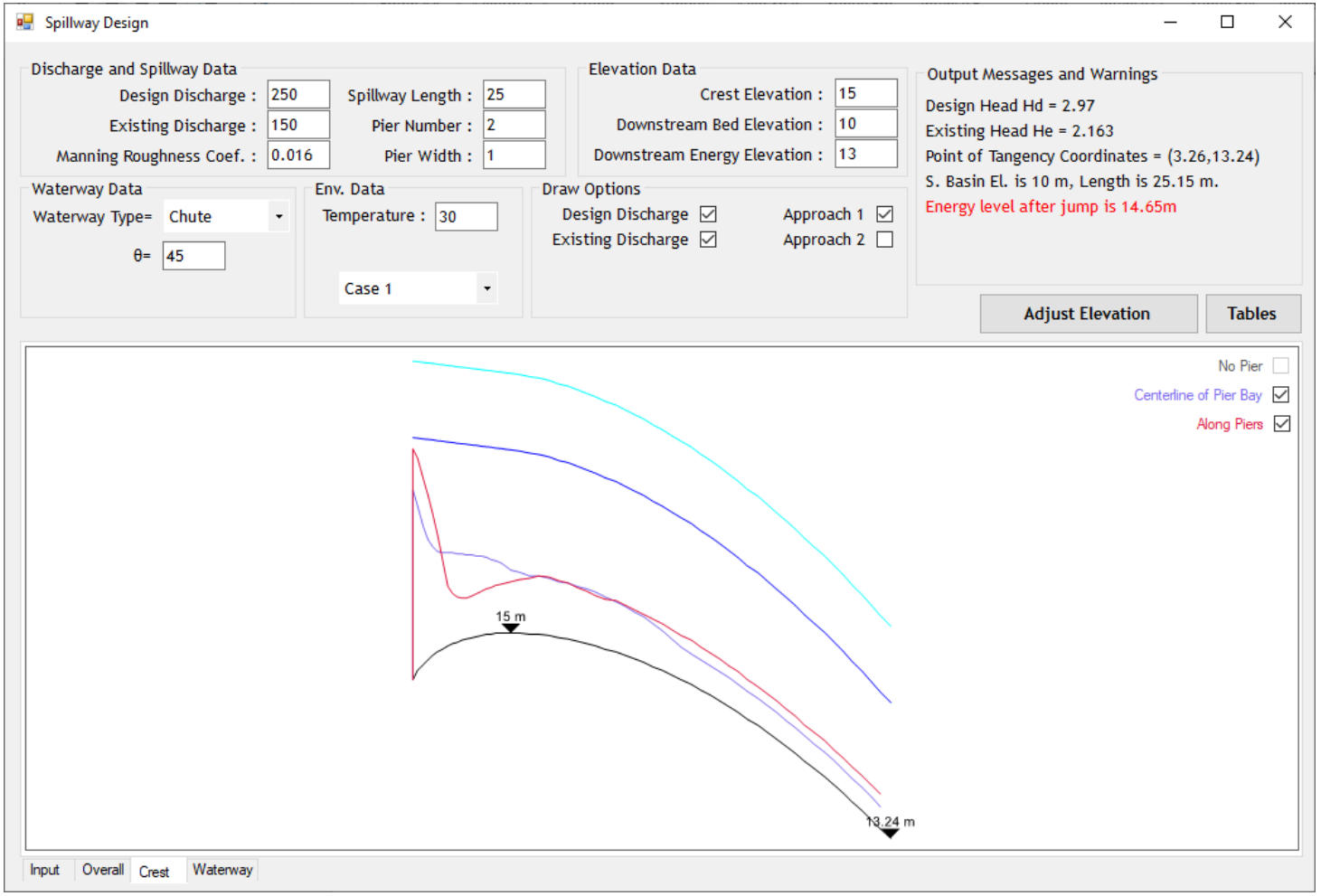


Figure 3.11. Crest Pressures with Piers, $\chi < 1.00$

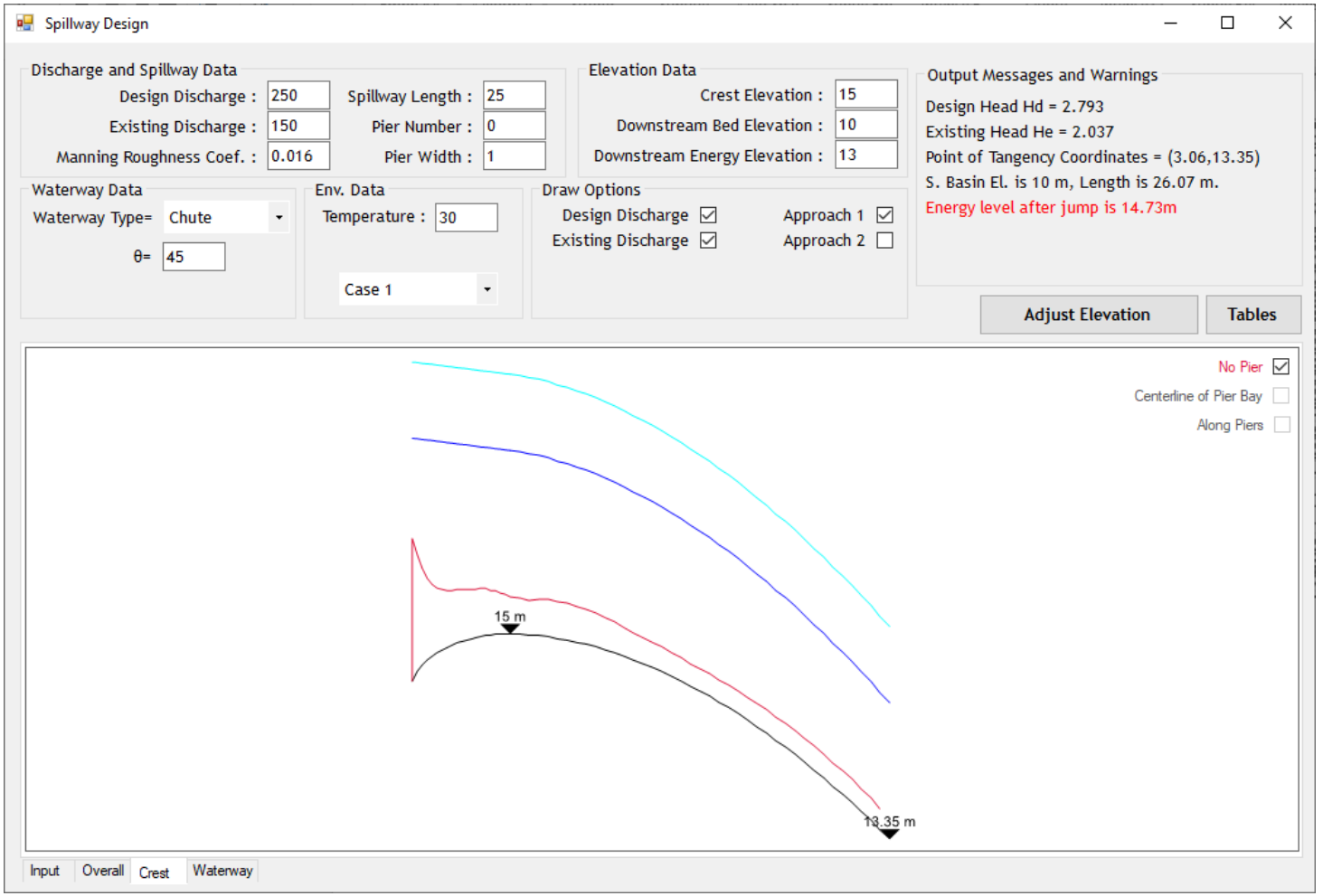


Figure 3.12. Crest Pressures without Piers, $\chi < 1.00$

3.4.6. Cavitation Calculations

Between the crest and the toe of the spillway, the cavitation parameter is calculated for both design and existing discharges at each node. In chute region, values of approach 1 are used in the calculation of cavitation parameters. Results are given in Table 3.8.

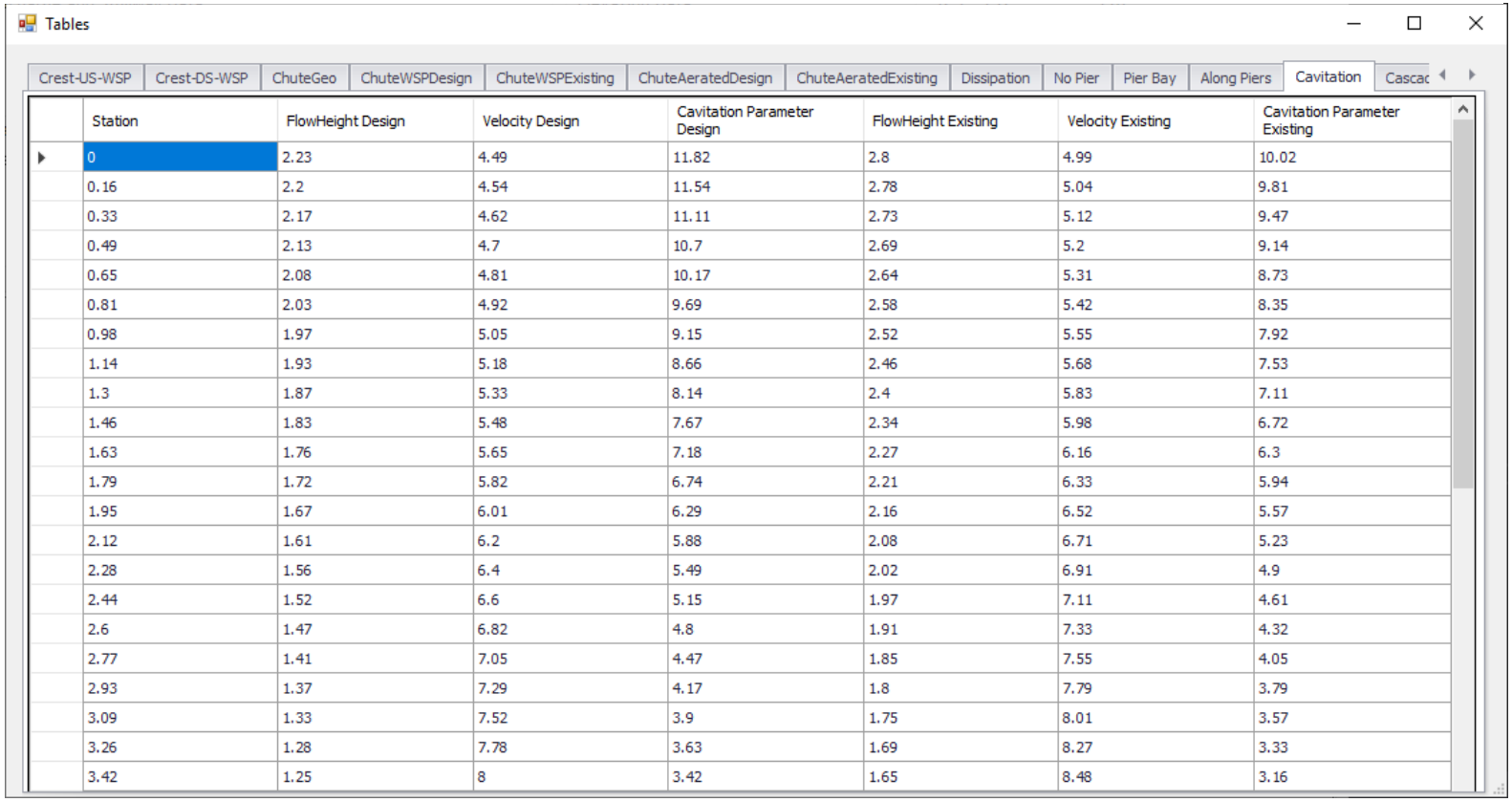
3.4.7. Chute Water Surface Profile

Two different approaches i.e. the standard step method without flow aeration and the approach with flow aeration are incorporated into the software to compute the water surface profile along the chute. For both approaches, the chute is divided into 20 segments and calculations are performed for each point.

Non-aerated flow is assumed in the first approach. Since the bed slope of the chute channel is steep, supercritical flow will occur. S2 gradually varied flow profile will be observed on the chute. Standard step method is used to determine the variation of flow depths along the chute. Results of approach 1 in tabular form are given in Table 3.9.

Aerated flow is considered in the second approach. Uniform flow depth, critical flow depth and distance to the inception point of aeration are calculated from Equations (2.18), (2.19), and (2.20), respectively. Flow depths are calculated from Equation (2.16). For the nodes located beyond inception point, air concentration is calculated from Equation (2.23), and the effect of the aeration is applied to the flow depth using Equation (2.22). For demonstration purposes, results of an arbitrary values are given in Figure 3.13.

Table 3.8. Cavitation Calculations



Station	FlowHeight Design	Velocity Design	Cavitation Parameter Design	FlowHeight Existing	Velocity Existing	Cavitation Parameter Existing
0	2.23	4.49	11.82	2.8	4.99	10.02
0.16	2.2	4.54	11.54	2.78	5.04	9.81
0.33	2.17	4.62	11.11	2.73	5.12	9.47
0.49	2.13	4.7	10.7	2.69	5.2	9.14
0.65	2.08	4.81	10.17	2.64	5.31	8.73
0.81	2.03	4.92	9.69	2.58	5.42	8.35
0.98	1.97	5.05	9.15	2.52	5.55	7.92
1.14	1.93	5.18	8.66	2.46	5.68	7.53
1.3	1.87	5.33	8.14	2.4	5.83	7.11
1.46	1.83	5.48	7.67	2.34	5.98	6.72
1.63	1.76	5.65	7.18	2.27	6.16	6.3
1.79	1.72	5.82	6.74	2.21	6.33	5.94
1.95	1.67	6.01	6.29	2.16	6.52	5.57
2.12	1.61	6.2	5.88	2.08	6.71	5.23
2.28	1.56	6.4	5.49	2.02	6.91	4.9
2.44	1.52	6.6	5.15	1.97	7.11	4.61
2.6	1.47	6.82	4.8	1.91	7.33	4.32
2.77	1.41	7.05	4.47	1.85	7.55	4.05
2.93	1.37	7.29	4.17	1.8	7.79	3.79
3.09	1.33	7.52	3.9	1.75	8.01	3.57
3.26	1.28	7.78	3.63	1.69	8.27	3.33
3.42	1.25	8	3.42	1.65	8.48	3.16

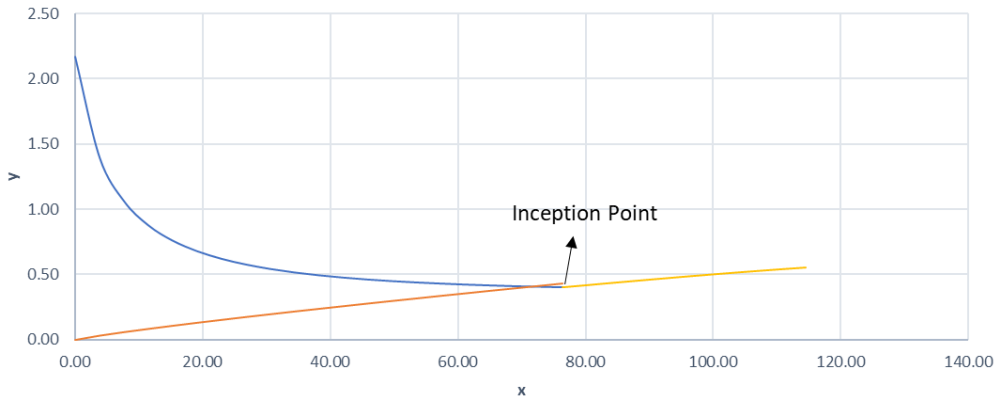


Figure 3.13. Effect of Aeration

Results of approach 2 in tabular form are given in Table 3.10. As shown in Figure 3.14, the results of the approaches are almost parallel to each other. As can also be seen from Figure 3.14, the aerated water surface profile is higher than the non-aerated flow due to flow bulking.

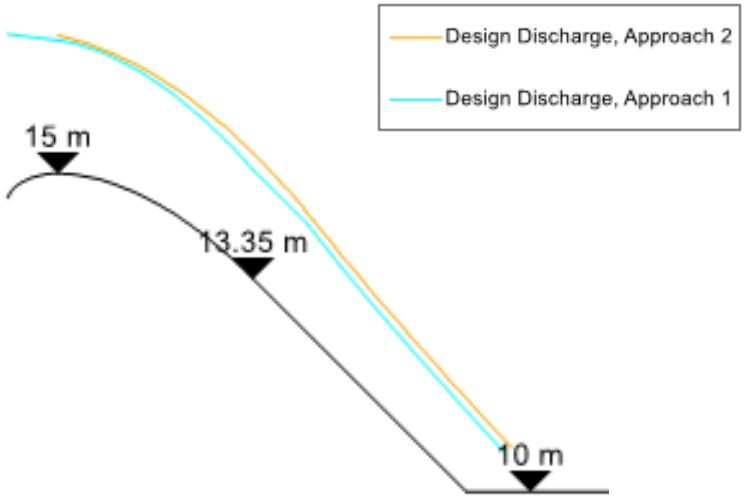


Figure 3.14. Comparison of Approach 1 and Approach 2

Table 3.9. Water Surface Profile over Chute, Approach 1

Tables

US Quad. | DS Quad. | Crest-US-WSP | Crest-DS-WSP | ChuteGeo | ChuteWSPDesign | ChuteWSPActual | ChuteAeratedDesign | ChuteAeratedActual | Dissipation | No Pier | Pier Bay | Along Piers

Index	ZCoord	Flow Depth	Area	Wetted Perimeter	Hydraulic Radius	Velocity	Froude Number	Friction Slope	Total Head	Averaged Friction Slope	Previous Total Head	Hydraulic Loss	Error Term
0	13.345	1.209	30.229	27.418	1.103	8.27	2.401	0.015	17.686	0	0	0	0
1	13.178	1.178	29.449	27.356	1.077	8.489	2.497	0.017	17.684	0.016	17.686	0.003	0
2	13.011	1.149	28.734	27.299	1.053	8.701	2.591	0.018	17.682	0.017	17.684	0.003	-0.001
3	12.843	1.123	28.076	27.246	1.03	8.904	2.683	0.02	17.679	0.019	17.682	0.003	0
4	12.676	1.099	27.466	27.197	1.01	9.102	2.773	0.021	17.676	0.02	17.679	0.003	0
5	12.509	1.076	26.899	27.152	0.991	9.294	2.861	0.022	17.672	0.022	17.676	0.004	0
6	12.342	1.055	26.369	27.11	0.973	9.481	2.947	0.024	17.669	0.023	17.672	0.004	0
7	12.174	1.035	25.871	27.07	0.956	9.663	3.033	0.025	17.665	0.025	17.669	0.004	-0.001
8	12.007	1.016	25.404	27.032	0.94	9.841	3.117	0.027	17.662	0.026	17.665	0.004	-0.001
9	11.84	0.998	24.964	26.997	0.925	10.014	3.2	0.028	17.658	0.028	17.662	0.005	0
10	11.673	0.982	24.549	26.964	0.91	10.184	3.281	0.03	17.653	0.029	17.658	0.005	0
11	11.505	0.966	24.154	26.932	0.897	10.35	3.362	0.032	17.649	0.031	17.653	0.005	-0.001
12	11.338	0.951	23.781	26.903	0.884	10.512	3.441	0.033	17.643	0.033	17.649	0.005	0
13	11.171	0.937	23.426	26.874	0.872	10.672	3.52	0.035	17.638	0.034	17.643	0.006	0
14	11.004	0.923	23.089	26.847	0.86	10.828	3.597	0.037	17.632	0.036	17.638	0.006	0
15	10.836	0.911	22.766	26.821	0.849	10.981	3.674	0.038	17.626	0.038	17.632	0.006	0
16	10.669	0.898	22.459	26.797	0.838	11.131	3.75	0.04	17.62	0.039	17.626	0.007	0
17	10.502	0.886	22.164	26.773	0.828	11.28	3.825	0.042	17.613	0.041	17.62	0.007	0
18	10.335	0.875	21.881	26.751	0.818	11.425	3.899	0.044	17.607	0.043	17.613	0.007	0
19	10.167	0.864	21.609	26.729	0.808	11.569	3.973	0.045	17.601	0.045	17.607	0.007	-0.001

Table 3.10. Water Surface Profile over Chute, Approach 2

Tables

US Quad. | DS Quad. | Crest-US-WSP | Crest-DS-WSP | ChuteGeo | ChuteWSPDesign | ChuteWSPActual | ChuteAeratedDesign | ChuteAeratedActual | Dissipation | No Pier | Pier Bay | Along Piers

Crest Station	Streamwise Distance	Crest Elevation	Normalized Streamwise Distance	Aerated Water Depth	Boundary Layer Thickness	Angle (rad)	Relative Head
0	0	15	0	2.17	0	0	3.25
0.08	0.08	15	0	2.13	0	0.05	3.25
0.15	0.15	14.99	0.01	2.1	0	0.08	3.25
0.23	0.23	14.99	0.01	2.07	0	0.11	3.25
0.31	0.31	14.98	0.01	2.03	0	0.14	3.25
0.38	0.38	14.96	0.01	2.01	0	0.17	3.24
0.46	0.46	14.95	0.02	1.98	0.01	0.2	3.24
0.54	0.54	14.93	0.02	1.95	0.01	0.22	3.24
0.61	0.61	14.92	0.02	1.92	0.01	0.25	3.24
0.69	0.69	14.9	0.02	1.9	0.01	0.27	3.24
0.77	0.77	14.87	0.03	1.87	0.01	0.3	3.24
0.84	0.85	14.85	0.03	1.85	0.01	0.32	3.25
0.92	0.93	14.82	0.03	1.82	0.01	0.35	3.25
0.99	1.01	14.79	0.04	1.8	0.01	0.37	3.26
1.07	1.1	14.76	0.04	1.77	0.01	0.39	3.26
1.15	1.18	14.73	0.04	1.75	0.01	0.41	3.27
1.22	1.26	14.7	0.04	1.73	0.01	0.43	3.28
1.3	1.35	14.66	0.05	1.71	0.01	0.45	3.29
1.38	1.44	14.62	0.05	1.68	0.01	0.47	3.3
1.45	1.52	14.58	0.05	1.66	0.01	0.49	3.31
1.53	1.61	14.54	0.06	1.64	0.02	0.51	3.32

3.4.8. Stepped Channel Water Surface Profile

Stepped spillway equations covered in Chapter 2 are implemented. Water surface profile consists of two straight lines. The starting point of the first line is the flow depth at the point of tangency of the bed profile and the end of the first line is the inception point of aeration. Aerated flow depth is calculated at that point and the water surface elevations corresponding to the subsequent flow depths until the toe are joined to finalize the water surface profile over the chute. Results of stepped channel calculation are given in Table 3.11.

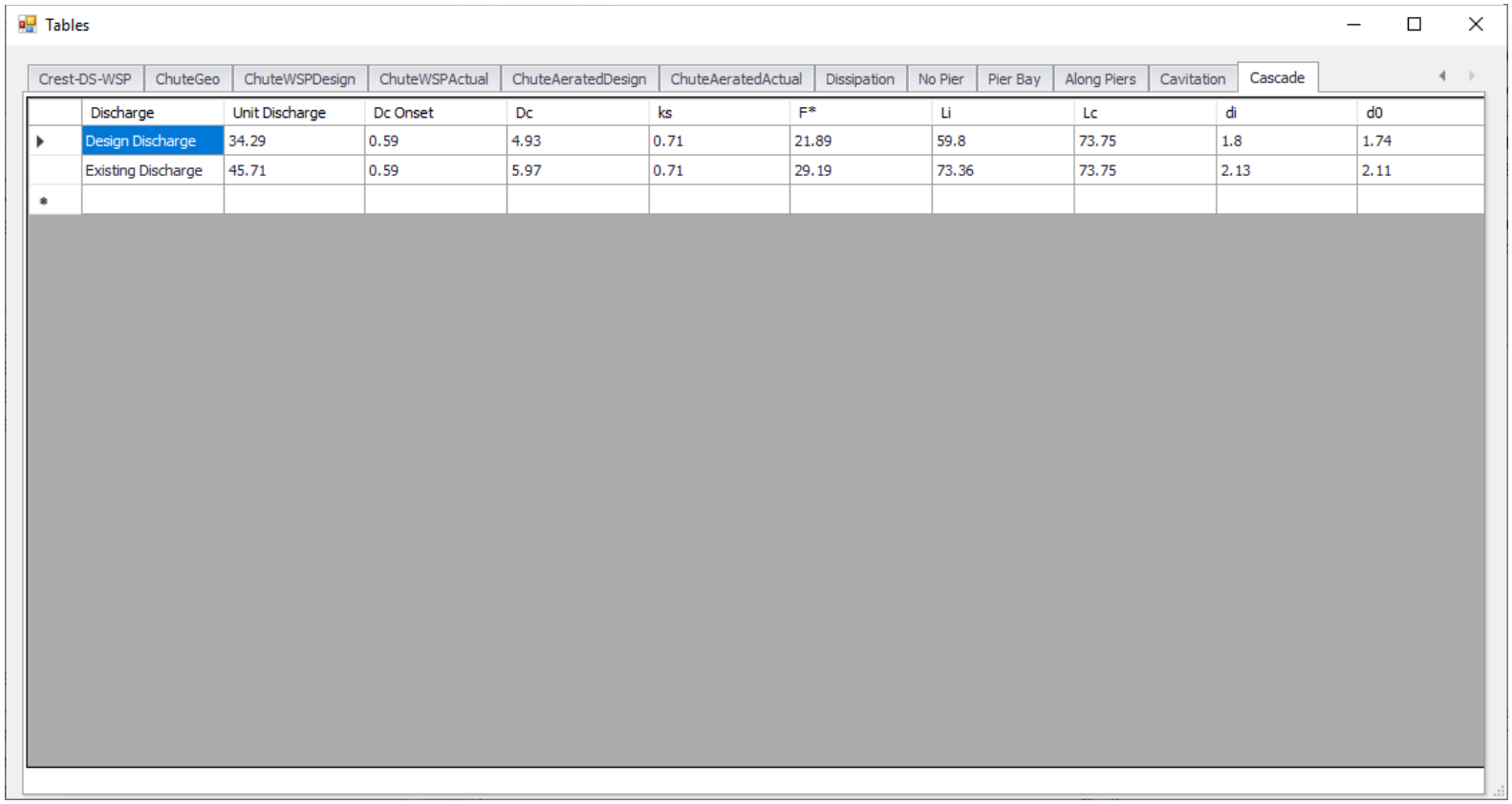
3.4.9. Energy Dissipation Basin Design

For each approach and discharge, stilling basin design is performed based on the flow characteristics at the end of the waterway. Flow depth, Froude number, velocity and energy level at the end of the stilling basin are computed by using Equation (2.31). Results of hydraulic jump calculations are given in Table 3.12. Stilling basin type is then decided and proper basin length is computed.

The energy level at the end of the stilling basin should not exceed the given downstream energy grade line which is an input data.

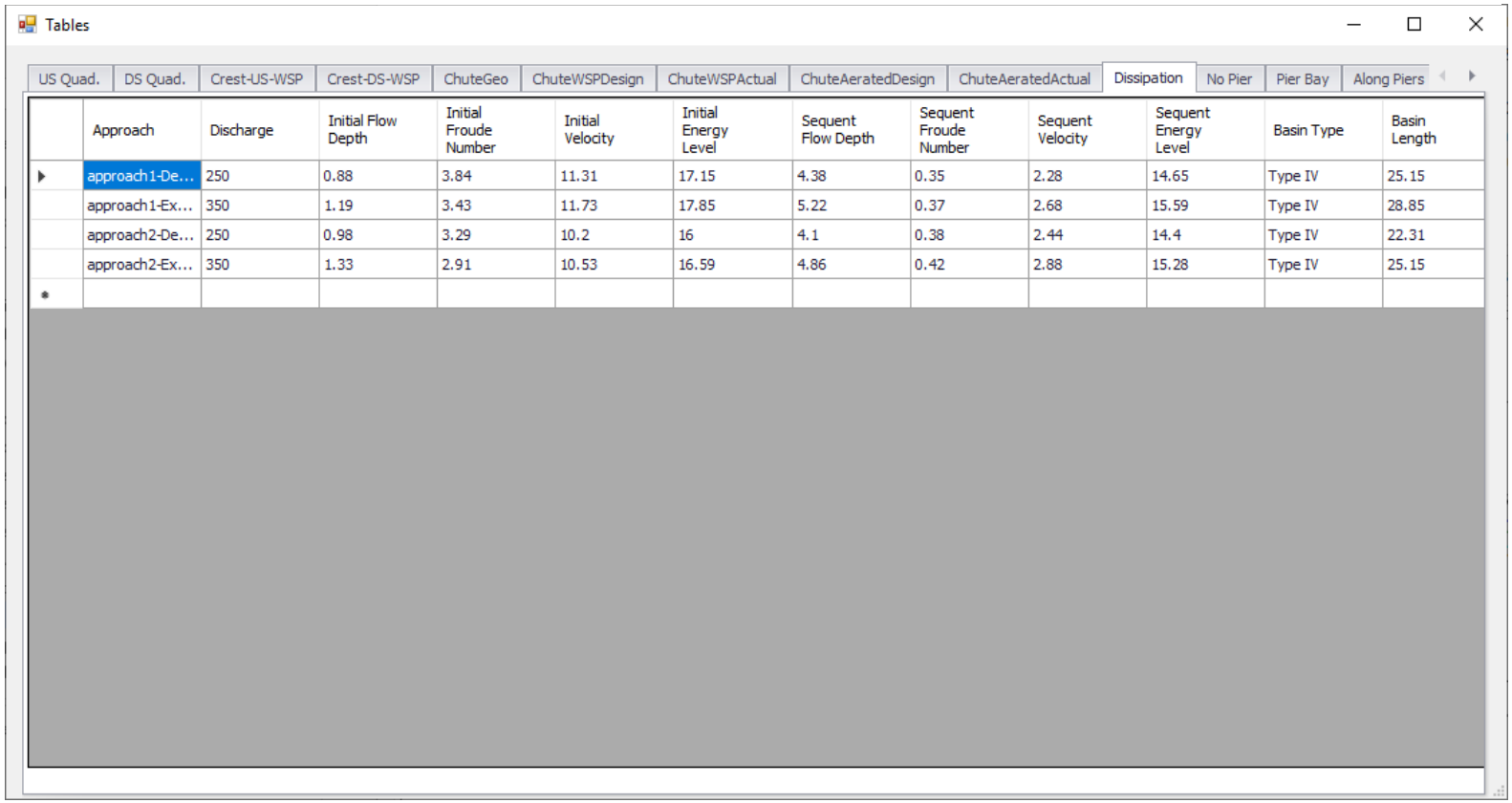
When the “Adjust Stilling Basin” button is clicked, the software computes the necessary stilling basin surface elevation by modifying it in small decrements. Calculations are performed until the energy levels at the sequent depth section and tailwater section are the same. In Figure 3.15 and Figure 3.16, stilling basin designs are displayed before and after the adjustment, respectively.

Table 3.11. Stepped Channel Calculations



	Discharge	Unit Discharge	Dc Onset	Dc	ks	F*	Li	Lc	di	d0
▶	Design Discharge	34.29	0.59	4.93	0.71	21.89	59.8	73.75	1.8	1.74
	Existing Discharge	45.71	0.59	5.97	0.71	29.19	73.36	73.75	2.13	2.11
*										

Table 3.12. Energy Dissipation Calculations



Approach	Discharge	Initial Flow Depth	Initial Froude Number	Initial Velocity	Initial Energy Level	Sequent Flow Depth	Sequent Froude Number	Sequent Velocity	Sequent Energy Level	Basin Type	Basin Length
approach1-De...	250	0.88	3.84	11.31	17.15	4.38	0.35	2.28	14.65	Type IV	25.15
approach1-Ex...	350	1.19	3.43	11.73	17.85	5.22	0.37	2.68	15.59	Type IV	28.85
approach2-De...	250	0.98	3.29	10.2	16	4.1	0.38	2.44	14.4	Type IV	22.31
approach2-Ex...	350	1.33	2.91	10.53	16.59	4.86	0.42	2.88	15.28	Type IV	25.15

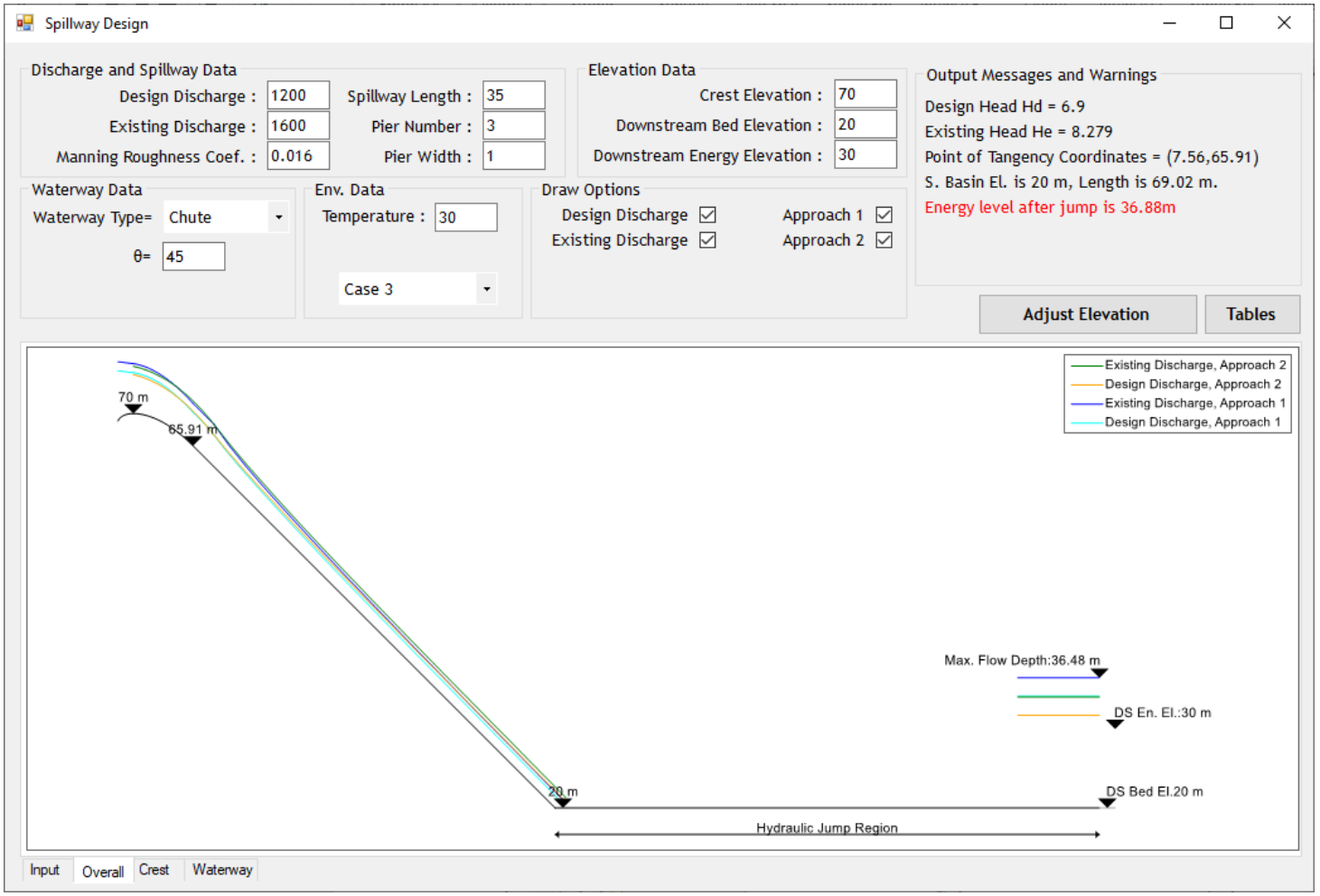


Figure 3.15. Stilling Basin Before Adjustment

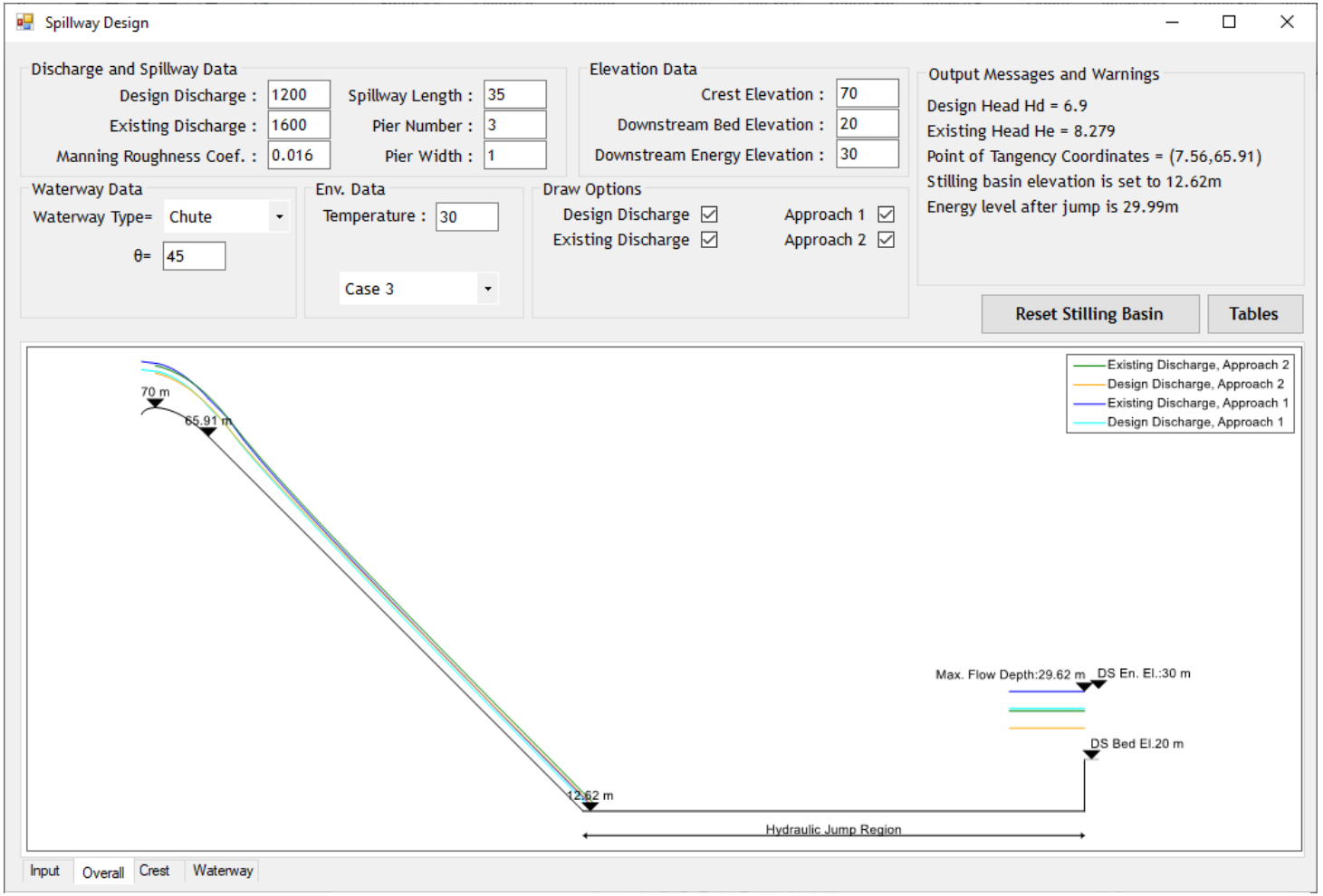


Figure 3.16. Stilling Basin After Adjustment

3.5. Outputs of the Software

The software will perform the flow analysis for given input data. The user will get the following outputs:

- Crest bottom profile,
- Water surface profiles on the crest,
- Pressure profiles on the crest,
- Waterway channel profile,
- Water surface profiles on the waterway channel,
- Necessary stilling basin elevation,
- Hydraulic jump information
- Stilling basin information

With the execution of the software, a designer can evaluate the results in terms of hydraulic conformity and cost in an implicit manner. The application of the software is demonstrated in the next chapter for various flow conditions and spillway types considered in this study.

CHAPTER 4

APPLICATION

4.1. Introduction

An application is presented to highlight the use of the software developed in this study. In the application, design and flow analysis of spillways having various properties are performed. Low head and high head spillways with chute and stepped waterways will be investigated. The scope of the applications is outlined in Figure 4.1 in which Q_d and Q_e stand for design discharge and existing discharge, respectively. Computational approaches for different spillway cases will be discussed in the following sections.

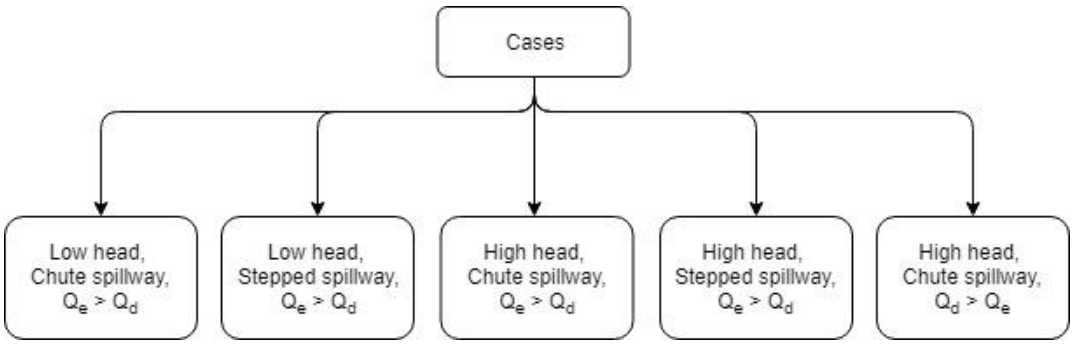


Figure 4.1. Cases Considered in the Study

4.2. Data Selection for Cases

Input data used in this application are hypothetical. Arbitrary values are used to show the differences among the cases.

Since the main objective of this application is to demonstrate the results of different variations for a spillway design, five different cases are investigated and results are discussed. Input data of these cases are listed in Table 4.1, in which K_s, E_d, and EGL_d

stand for spillway crest elevation, downstream bed elevation and downstream energy grade elevation, respectively.

Table 4.1. Input Data

Data	Case 1	Case 2	Case 3	Case 4	Case 5
Q_d (m^3/s)	250	250	1200	1200	1200
Q_e (m^3/s)	350	350	1600	1600	800
L_{net} (m)	25	25	35	35	35
N_p	2	2	3	3	3
B_p (m)	1	1	1	1	1
K_s (m)	15	15	70	70	70
E_d (m)	10	10	20	20	20
EGL_d (m)	13	13	30	30	30
Waterway Type	Chute	Stepped	Chute	Stepped	Chute
α (°)	45°	-	45°	-	45°
h_s (m)	-	0.4	-	1.0	-
l_s (m)	-	0.4	-	1.0	-
Temperature (°C)	30	30	30	30	30

Comparison of Case 1 and Case 2 demonstrates the effect of different waterway types on low head spillways whereas comparison of Case 3 and Case 4 demonstrates the effect of different waterway type on high head spillways.

Comparison of Case 1 and Case 3 demonstrates the influence of total head on chute spillways whereas comparison of Case 2 and Case 4 demonstrates the influence of total head on stepped spillways.

Comparison of Case 3 and Case 5 demonstrates the influence of head ratio on chute spillways. The main purpose of this comparison is to show how crest pressures vary with the head ratio.

4.3. Results of Case 1

Case 1 represents a low head chute spillway. With the given data, design head is calculated as 2.97 m whereas the existing head is 3.66 m. Existing discharge is greater than design discharge. Therefore, negative pressure on the crest is expected as seen in Figure 4.2. Crest bed profile is determined for the calculated design head, and the results are given for upstream and downstream quadrants in Table 4.2 and Table 4.3, respectively. Point of tangency coordinates are (3.26, 13.24). Water surface profile along the crest for the design head and the existing head are computed. The results are given in Table 4.4 and Table 4.5, respectively. When approach 1 is used, vertical flow depth at the beginning of the chute channel is 1.82 m for the design discharge and 2.40 m for the existing discharge. Flow depth perpendicular to the channel bottom is calculated 1.287 m for the design discharge and 1.697 m for the existing discharge. Standard step method is applied from this point to the end of the chute. Therefore, 0.876 m flow depth is computed at the end of the chute for the design discharge, whereas 1.187 m flow depth is computed for the existing discharge. Results for design discharge and existing discharge are given in Table 4.6 and Table 4.7, respectively.

In approach 2, 1.29 m and 1.74 m flow depths are computed for the design discharge and the existing discharge at the point of tangency, respectively. With this approach, 0.98 m and 1.33 m flow depths are computed at the end of the chute for the design and the existing discharge, respectively. Inception point of aeration is located 76.44 m and 97.39 m for design discharge and existing discharge, respectively. Turbulent boundary layer does not reach the free surface for both discharges. Results for design discharge and existing discharge are given in Table 4.8 and Table 4.9, respectively.

Cavitation parameters along the crest are calculated. The results are given in Table 4.10.

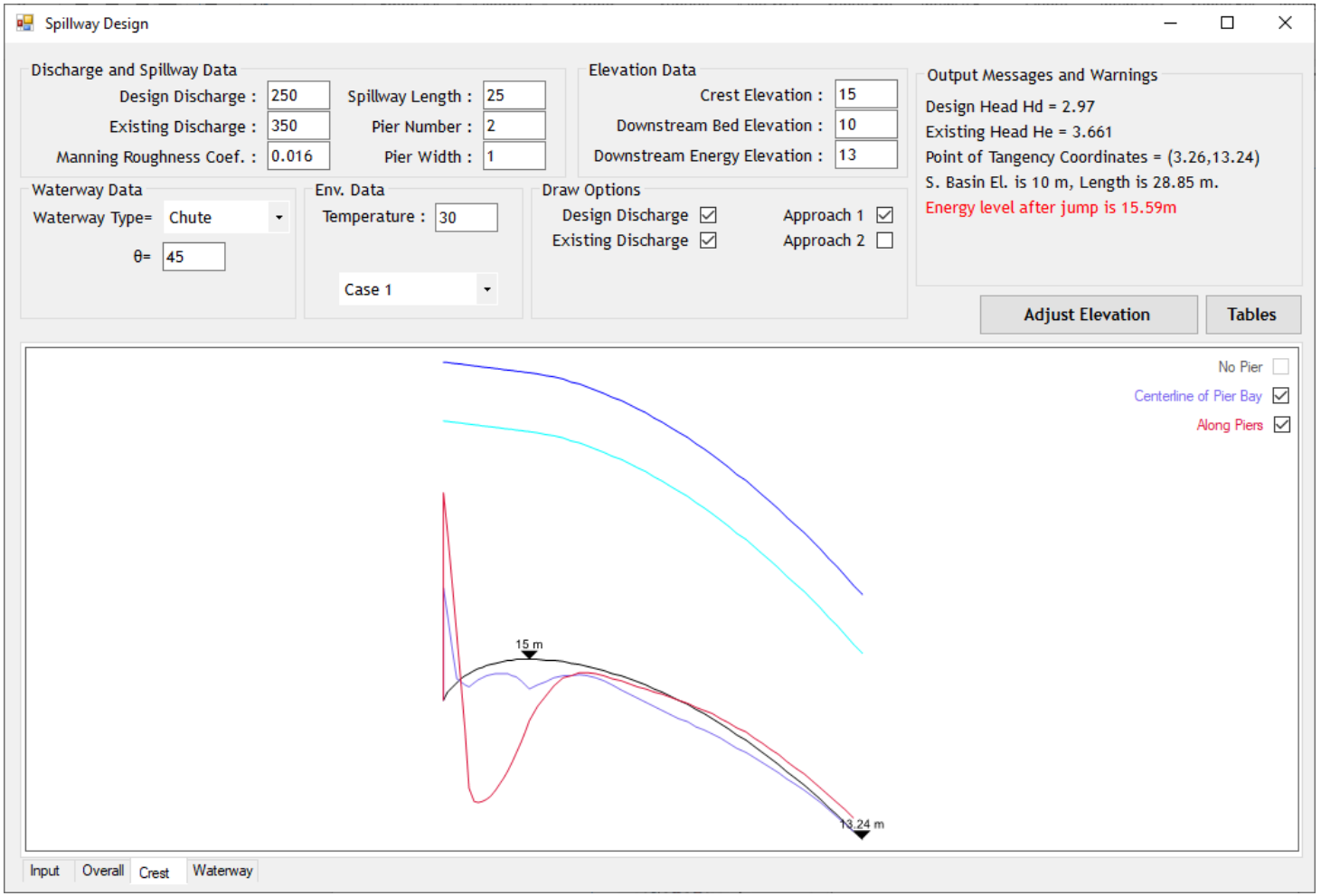


Figure 4.2. Case 1 – Crest Pressures

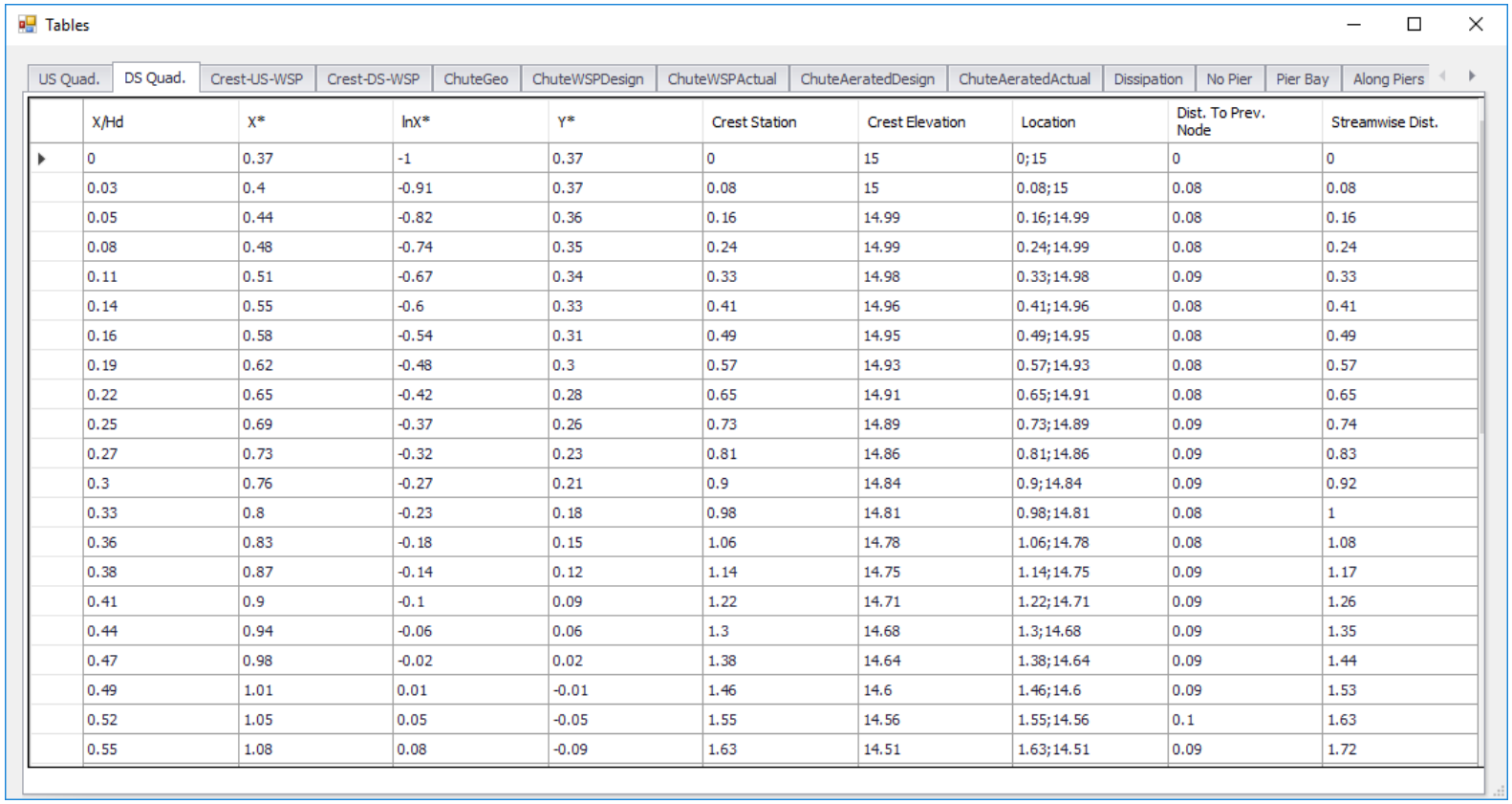
Table 4.2. Case 1 Crest Upstream Quadrant Bed Profile

Tables

US Quad. | DS Quad. | Crest-US-WSP | Crest-DS-WSP | ChuteGeo | ChuteWSPDesign | ChuteWSPActual | ChuteAeratedDesign | ChuteAeratedActual | Dissipation | No Pier | Pier Bay | Along Piers

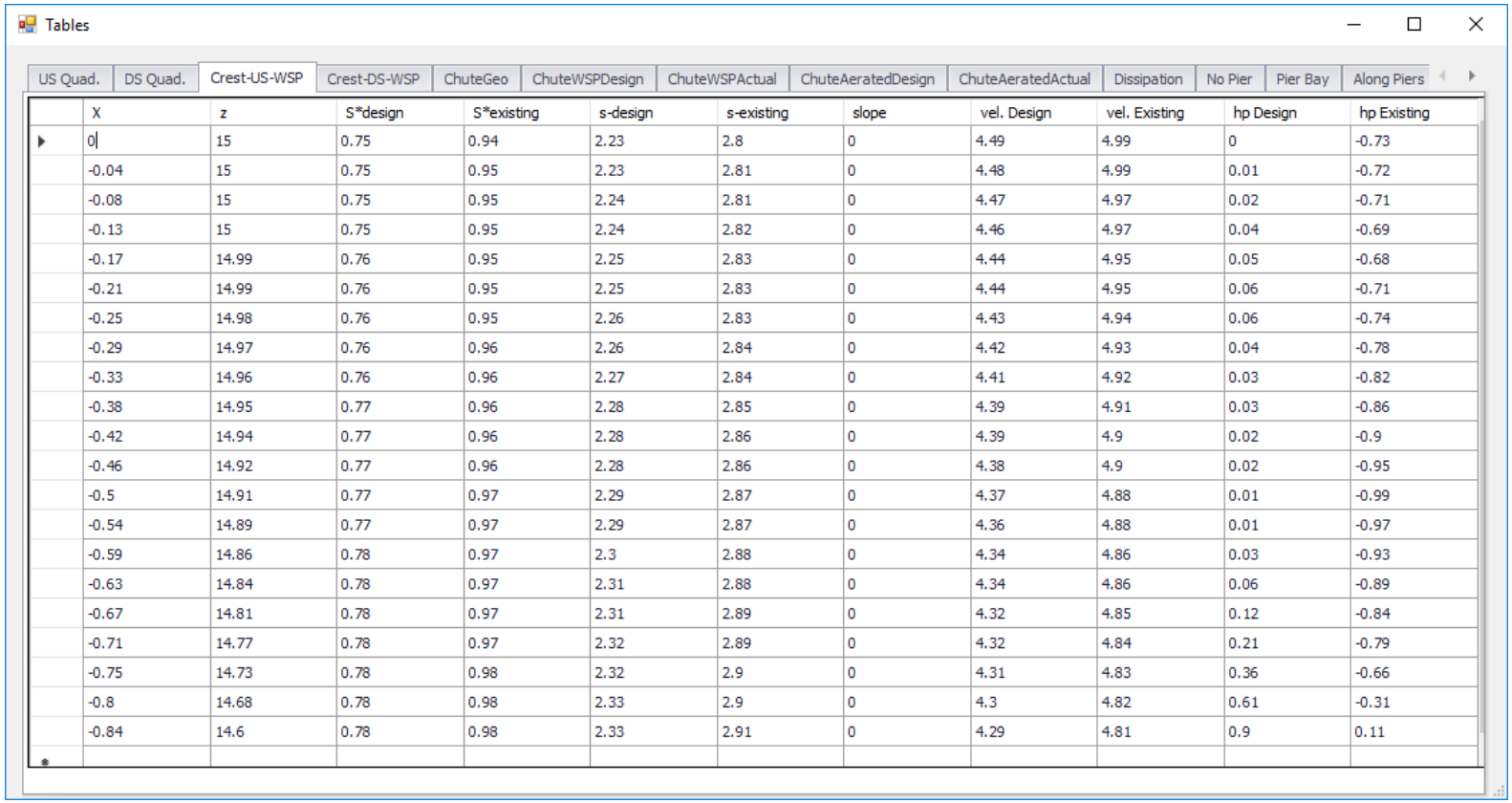
X/Hd	X*	lnX*	Y*	Crest Station	Crest Elevation	Location
0	0.37	-1	0.37	0	15	0;15
-0.01	0.35	-1.05	0.37	-0.04	15	-0.04;15
-0.03	0.33	-1.11	0.37	-0.08	15	-0.08;15
-0.04	0.31	-1.16	0.36	-0.13	15	-0.13;15
-0.06	0.29	-1.22	0.36	-0.17	14.99	-0.17;14.99
-0.07	0.28	-1.29	0.36	-0.21	14.99	-0.21;14.99
-0.08	0.26	-1.36	0.35	-0.25	14.98	-0.25;14.98
-0.1	0.24	-1.43	0.34	-0.29	14.97	-0.29;14.97
-0.11	0.22	-1.51	0.33	-0.33	14.96	-0.33;14.96
-0.13	0.2	-1.6	0.32	-0.38	14.95	-0.38;14.95
-0.14	0.18	-1.69	0.31	-0.42	14.94	-0.42;14.94
-0.15	0.17	-1.8	0.3	-0.46	14.92	-0.46;14.92
-0.17	0.15	-1.92	0.28	-0.5	14.91	-0.5;14.91
-0.18	0.13	-2.05	0.26	-0.54	14.89	-0.54;14.89
-0.2	0.11	-2.2	0.24	-0.59	14.86	-0.59;14.86
-0.21	0.09	-2.39	0.22	-0.63	14.84	-0.63;14.84
-0.23	0.07	-2.61	0.19	-0.67	14.81	-0.67;14.81
-0.24	0.06	-2.9	0.16	-0.71	14.77	-0.71;14.77
-0.25	0.04	-3.3	0.12	-0.75	14.73	-0.75;14.73
-0.27	0.02	-4	0.07	-0.8	14.68	-0.8;14.68
-0.28	0	-22.76	0	-0.84	14.6	-0.84;14.6

Table 4.3. Case 1 Crest Downstream Quadrant Bed Profile



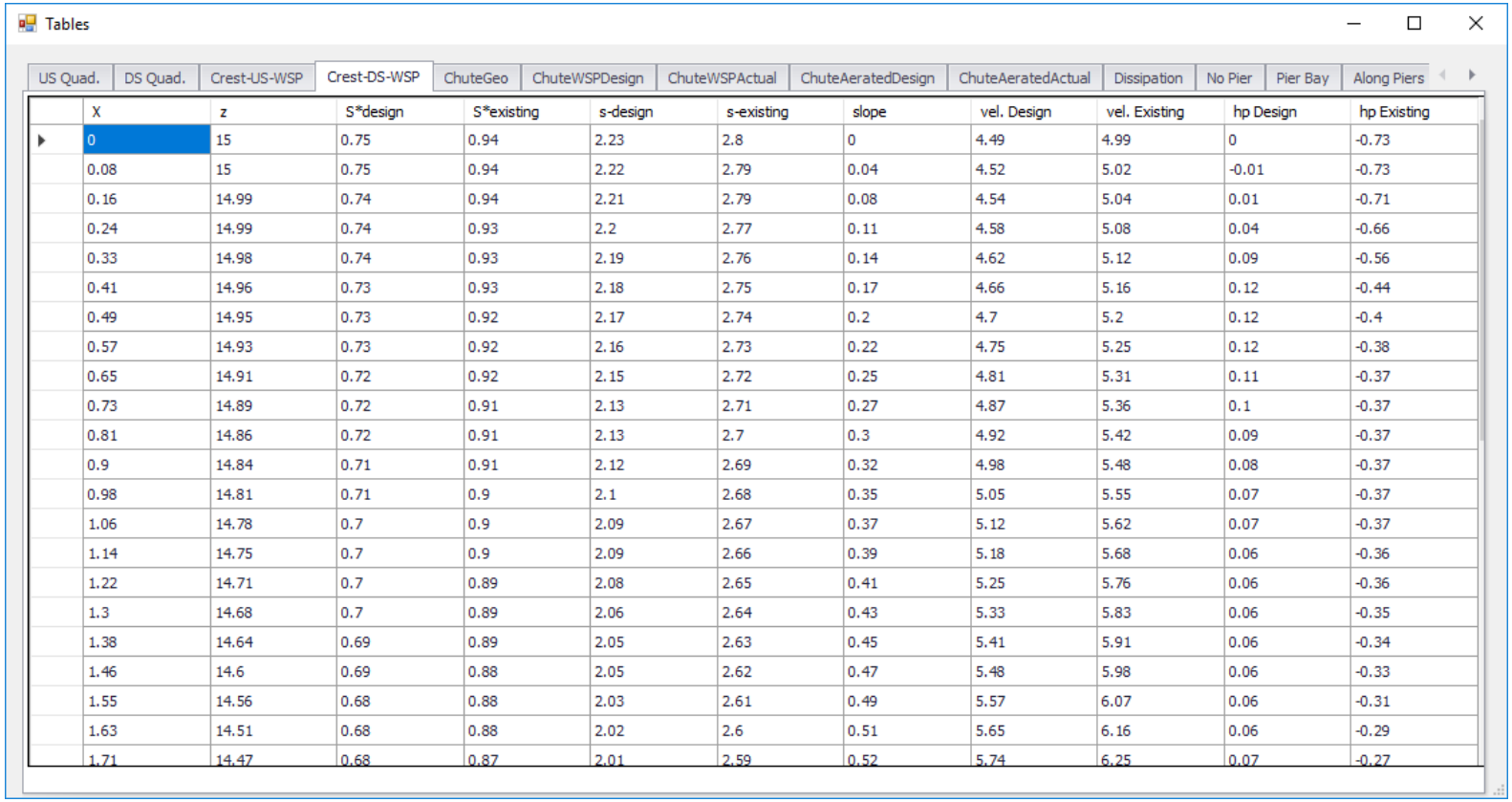
X/Hd	X*	lnX*	Y*	Crest Station	Crest Elevation	Location	Dist. To Prev. Node	Streamwise Dist.
0	0.37	-1	0.37	0	15	0;15	0	0
0.03	0.4	-0.91	0.37	0.08	15	0.08;15	0.08	0.08
0.05	0.44	-0.82	0.36	0.16	14.99	0.16;14.99	0.08	0.16
0.08	0.48	-0.74	0.35	0.24	14.99	0.24;14.99	0.08	0.24
0.11	0.51	-0.67	0.34	0.33	14.98	0.33;14.98	0.09	0.33
0.14	0.55	-0.6	0.33	0.41	14.96	0.41;14.96	0.08	0.41
0.16	0.58	-0.54	0.31	0.49	14.95	0.49;14.95	0.08	0.49
0.19	0.62	-0.48	0.3	0.57	14.93	0.57;14.93	0.08	0.57
0.22	0.65	-0.42	0.28	0.65	14.91	0.65;14.91	0.08	0.65
0.25	0.69	-0.37	0.26	0.73	14.89	0.73;14.89	0.09	0.74
0.27	0.73	-0.32	0.23	0.81	14.86	0.81;14.86	0.09	0.83
0.3	0.76	-0.27	0.21	0.9	14.84	0.9;14.84	0.09	0.92
0.33	0.8	-0.23	0.18	0.98	14.81	0.98;14.81	0.08	1
0.36	0.83	-0.18	0.15	1.06	14.78	1.06;14.78	0.08	1.08
0.38	0.87	-0.14	0.12	1.14	14.75	1.14;14.75	0.09	1.17
0.41	0.9	-0.1	0.09	1.22	14.71	1.22;14.71	0.09	1.26
0.44	0.94	-0.06	0.06	1.3	14.68	1.3;14.68	0.09	1.35
0.47	0.98	-0.02	0.02	1.38	14.64	1.38;14.64	0.09	1.44
0.49	1.01	0.01	-0.01	1.46	14.6	1.46;14.6	0.09	1.53
0.52	1.05	0.05	-0.05	1.55	14.56	1.55;14.56	0.1	1.63
0.55	1.08	0.08	-0.09	1.63	14.51	1.63;14.51	0.09	1.72

Table 4.4. Case 1 Water Surface Profile Along Crest Upstream Quadrant



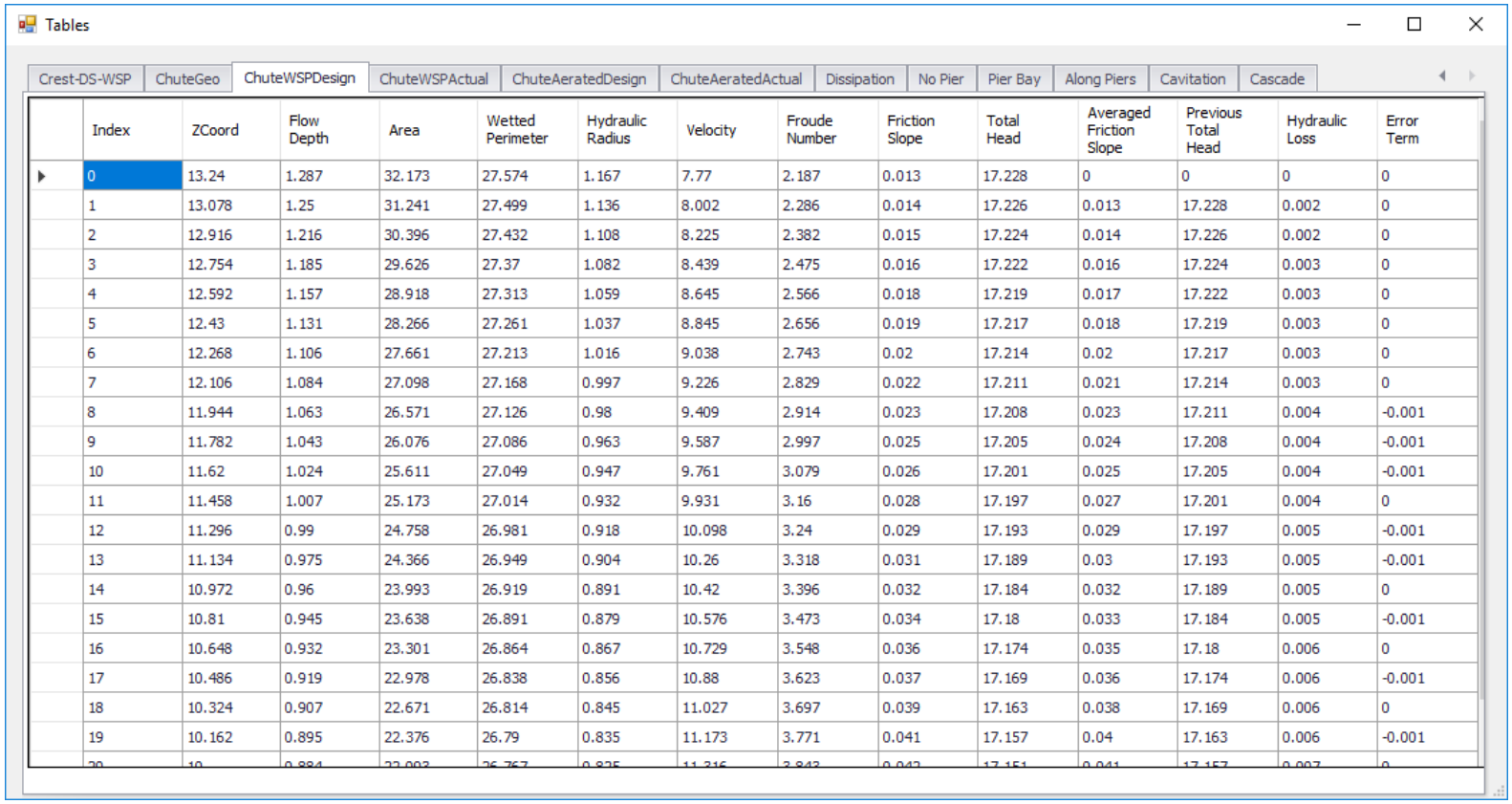
X	z	S*design	S*existing	s-design	s-existing	slope	vel. Design	vel. Existing	hp Design	hp Existing
0	15	0.75	0.94	2.23	2.8	0	4.49	4.99	0	-0.73
-0.04	15	0.75	0.95	2.23	2.81	0	4.48	4.99	0.01	-0.72
-0.08	15	0.75	0.95	2.24	2.81	0	4.47	4.97	0.02	-0.71
-0.13	15	0.75	0.95	2.24	2.82	0	4.46	4.97	0.04	-0.69
-0.17	14.99	0.76	0.95	2.25	2.83	0	4.44	4.95	0.05	-0.68
-0.21	14.99	0.76	0.95	2.25	2.83	0	4.44	4.95	0.06	-0.71
-0.25	14.98	0.76	0.95	2.26	2.83	0	4.43	4.94	0.06	-0.74
-0.29	14.97	0.76	0.96	2.26	2.84	0	4.42	4.93	0.04	-0.78
-0.33	14.96	0.76	0.96	2.27	2.84	0	4.41	4.92	0.03	-0.82
-0.38	14.95	0.77	0.96	2.28	2.85	0	4.39	4.91	0.03	-0.86
-0.42	14.94	0.77	0.96	2.28	2.86	0	4.39	4.9	0.02	-0.9
-0.46	14.92	0.77	0.96	2.28	2.86	0	4.38	4.9	0.02	-0.95
-0.5	14.91	0.77	0.97	2.29	2.87	0	4.37	4.88	0.01	-0.99
-0.54	14.89	0.77	0.97	2.29	2.87	0	4.36	4.88	0.01	-0.97
-0.59	14.86	0.78	0.97	2.3	2.88	0	4.34	4.86	0.03	-0.93
-0.63	14.84	0.78	0.97	2.31	2.88	0	4.34	4.86	0.06	-0.89
-0.67	14.81	0.78	0.97	2.31	2.89	0	4.32	4.85	0.12	-0.84
-0.71	14.77	0.78	0.97	2.32	2.89	0	4.32	4.84	0.21	-0.79
-0.75	14.73	0.78	0.98	2.32	2.9	0	4.31	4.83	0.36	-0.66
-0.8	14.68	0.78	0.98	2.33	2.9	0	4.3	4.82	0.61	-0.31
-0.84	14.6	0.78	0.98	2.33	2.91	0	4.29	4.81	0.9	0.11

Table 4.5. Case 1 Water Surface Profile Along Crest Downstream Quadrant



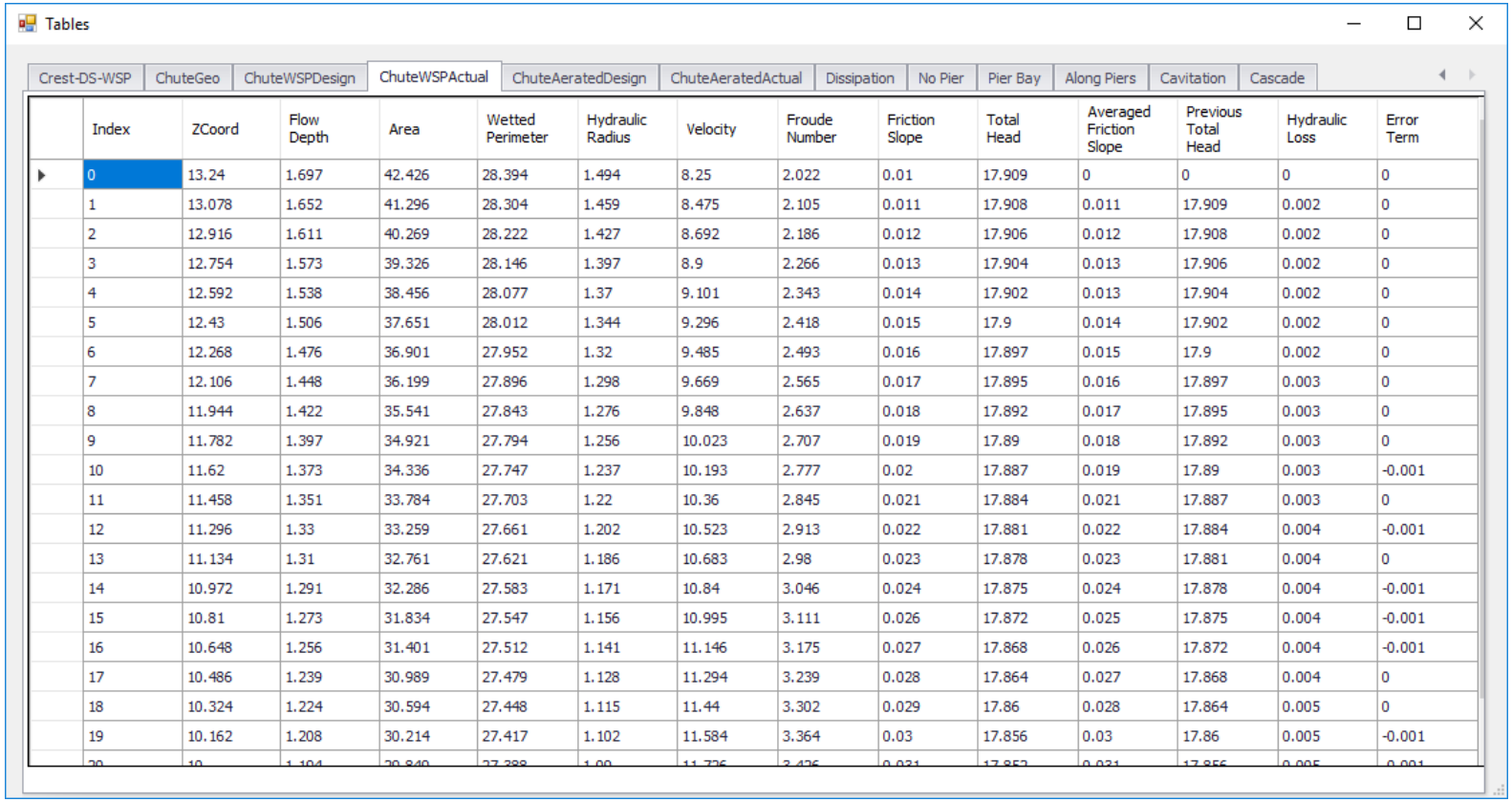
X	z	S*design	S*existing	s-design	s-existing	slope	vel. Design	vel. Existing	hp Design	hp Existing
0	15	0.75	0.94	2.23	2.8	0	4.49	4.99	0	-0.73
0.08	15	0.75	0.94	2.22	2.79	0.04	4.52	5.02	-0.01	-0.73
0.16	14.99	0.74	0.94	2.21	2.79	0.08	4.54	5.04	0.01	-0.71
0.24	14.99	0.74	0.93	2.2	2.77	0.11	4.58	5.08	0.04	-0.66
0.33	14.98	0.74	0.93	2.19	2.76	0.14	4.62	5.12	0.09	-0.56
0.41	14.96	0.73	0.93	2.18	2.75	0.17	4.66	5.16	0.12	-0.44
0.49	14.95	0.73	0.92	2.17	2.74	0.2	4.7	5.2	0.12	-0.4
0.57	14.93	0.73	0.92	2.16	2.73	0.22	4.75	5.25	0.12	-0.38
0.65	14.91	0.72	0.92	2.15	2.72	0.25	4.81	5.31	0.11	-0.37
0.73	14.89	0.72	0.91	2.13	2.71	0.27	4.87	5.36	0.1	-0.37
0.81	14.86	0.72	0.91	2.13	2.7	0.3	4.92	5.42	0.09	-0.37
0.9	14.84	0.71	0.91	2.12	2.69	0.32	4.98	5.48	0.08	-0.37
0.98	14.81	0.71	0.9	2.1	2.68	0.35	5.05	5.55	0.07	-0.37
1.06	14.78	0.7	0.9	2.09	2.67	0.37	5.12	5.62	0.07	-0.37
1.14	14.75	0.7	0.9	2.09	2.66	0.39	5.18	5.68	0.06	-0.36
1.22	14.71	0.7	0.89	2.08	2.65	0.41	5.25	5.76	0.06	-0.36
1.3	14.68	0.7	0.89	2.06	2.64	0.43	5.33	5.83	0.06	-0.35
1.38	14.64	0.69	0.89	2.05	2.63	0.45	5.41	5.91	0.06	-0.34
1.46	14.6	0.69	0.88	2.05	2.62	0.47	5.48	5.98	0.06	-0.33
1.55	14.56	0.68	0.88	2.03	2.61	0.49	5.57	6.07	0.06	-0.31
1.63	14.51	0.68	0.88	2.02	2.6	0.51	5.65	6.16	0.06	-0.29
1.71	14.47	0.68	0.87	2.01	2.59	0.52	5.74	6.25	0.07	-0.27

Table 4.6. Case 1 Water Surface Profile Along Chute Channel (Design Discharge, Approach 1)



Index	ZCoord	Flow Depth	Area	Wetted Perimeter	Hydraulic Radius	Velocity	Froude Number	Friction Slope	Total Head	Averaged Friction Slope	Previous Total Head	Hydraulic Loss	Error Term
0	13.24	1.287	32.173	27.574	1.167	7.77	2.187	0.013	17.228	0	0	0	0
1	13.078	1.25	31.241	27.499	1.136	8.002	2.286	0.014	17.226	0.013	17.228	0.002	0
2	12.916	1.216	30.396	27.432	1.108	8.225	2.382	0.015	17.224	0.014	17.226	0.002	0
3	12.754	1.185	29.626	27.37	1.082	8.439	2.475	0.016	17.222	0.016	17.224	0.003	0
4	12.592	1.157	28.918	27.313	1.059	8.645	2.566	0.018	17.219	0.017	17.222	0.003	0
5	12.43	1.131	28.266	27.261	1.037	8.845	2.656	0.019	17.217	0.018	17.219	0.003	0
6	12.268	1.106	27.661	27.213	1.016	9.038	2.743	0.02	17.214	0.02	17.217	0.003	0
7	12.106	1.084	27.098	27.168	0.997	9.226	2.829	0.022	17.211	0.021	17.214	0.003	0
8	11.944	1.063	26.571	27.126	0.98	9.409	2.914	0.023	17.208	0.023	17.211	0.004	-0.001
9	11.782	1.043	26.076	27.086	0.963	9.587	2.997	0.025	17.205	0.024	17.208	0.004	-0.001
10	11.62	1.024	25.611	27.049	0.947	9.761	3.079	0.026	17.201	0.025	17.205	0.004	-0.001
11	11.458	1.007	25.173	27.014	0.932	9.931	3.16	0.028	17.197	0.027	17.201	0.004	0
12	11.296	0.99	24.758	26.981	0.918	10.098	3.24	0.029	17.193	0.029	17.197	0.005	-0.001
13	11.134	0.975	24.366	26.949	0.904	10.26	3.318	0.031	17.189	0.03	17.193	0.005	-0.001
14	10.972	0.96	23.993	26.919	0.891	10.42	3.396	0.032	17.184	0.032	17.189	0.005	0
15	10.81	0.945	23.638	26.891	0.879	10.576	3.473	0.034	17.18	0.033	17.184	0.005	-0.001
16	10.648	0.932	23.301	26.864	0.867	10.729	3.548	0.036	17.174	0.035	17.18	0.006	0
17	10.486	0.919	22.978	26.838	0.856	10.88	3.623	0.037	17.169	0.036	17.174	0.006	-0.001
18	10.324	0.907	22.671	26.814	0.845	11.027	3.697	0.039	17.163	0.038	17.169	0.006	0
19	10.162	0.895	22.376	26.79	0.835	11.173	3.771	0.041	17.157	0.04	17.163	0.006	-0.001
20	10	0.884	22.093	26.767	0.825	11.316	3.843	0.042	17.151	0.041	17.157	0.007	0

Table 4.7. Case 1 Water Surface Profile Along Chute Channel (Existing Discharge, Approach 1)



Index	ZCoord	Flow Depth	Area	Wetted Perimeter	Hydraulic Radius	Velocity	Froude Number	Friction Slope	Total Head	Averaged Friction Slope	Previous Total Head	Hydraulic Loss	Error Term
0	13.24	1.697	42.426	28.394	1.494	8.25	2.022	0.01	17.909	0	0	0	0
1	13.078	1.652	41.296	28.304	1.459	8.475	2.105	0.011	17.908	0.011	17.909	0.002	0
2	12.916	1.611	40.269	28.222	1.427	8.692	2.186	0.012	17.906	0.012	17.908	0.002	0
3	12.754	1.573	39.326	28.146	1.397	8.9	2.266	0.013	17.904	0.013	17.906	0.002	0
4	12.592	1.538	38.456	28.077	1.37	9.101	2.343	0.014	17.902	0.013	17.904	0.002	0
5	12.43	1.506	37.651	28.012	1.344	9.296	2.418	0.015	17.9	0.014	17.902	0.002	0
6	12.268	1.476	36.901	27.952	1.32	9.485	2.493	0.016	17.897	0.015	17.9	0.002	0
7	12.106	1.448	36.199	27.896	1.298	9.669	2.565	0.017	17.895	0.016	17.897	0.003	0
8	11.944	1.422	35.541	27.843	1.276	9.848	2.637	0.018	17.892	0.017	17.895	0.003	0
9	11.782	1.397	34.921	27.794	1.256	10.023	2.707	0.019	17.89	0.018	17.892	0.003	0
10	11.62	1.373	34.336	27.747	1.237	10.193	2.777	0.02	17.887	0.019	17.89	0.003	-0.001
11	11.458	1.351	33.784	27.703	1.22	10.36	2.845	0.021	17.884	0.021	17.887	0.003	0
12	11.296	1.33	33.259	27.661	1.202	10.523	2.913	0.022	17.881	0.022	17.884	0.004	-0.001
13	11.134	1.31	32.761	27.621	1.186	10.683	2.98	0.023	17.878	0.023	17.881	0.004	0
14	10.972	1.291	32.286	27.583	1.171	10.84	3.046	0.024	17.875	0.024	17.878	0.004	-0.001
15	10.81	1.273	31.834	27.547	1.156	10.995	3.111	0.026	17.872	0.025	17.875	0.004	-0.001
16	10.648	1.256	31.401	27.512	1.141	11.146	3.175	0.027	17.868	0.026	17.872	0.004	-0.001
17	10.486	1.239	30.989	27.479	1.128	11.294	3.239	0.028	17.864	0.027	17.868	0.004	0
18	10.324	1.224	30.594	27.448	1.115	11.44	3.302	0.029	17.86	0.028	17.864	0.005	0
19	10.162	1.208	30.214	27.417	1.102	11.584	3.364	0.03	17.856	0.03	17.86	0.005	-0.001

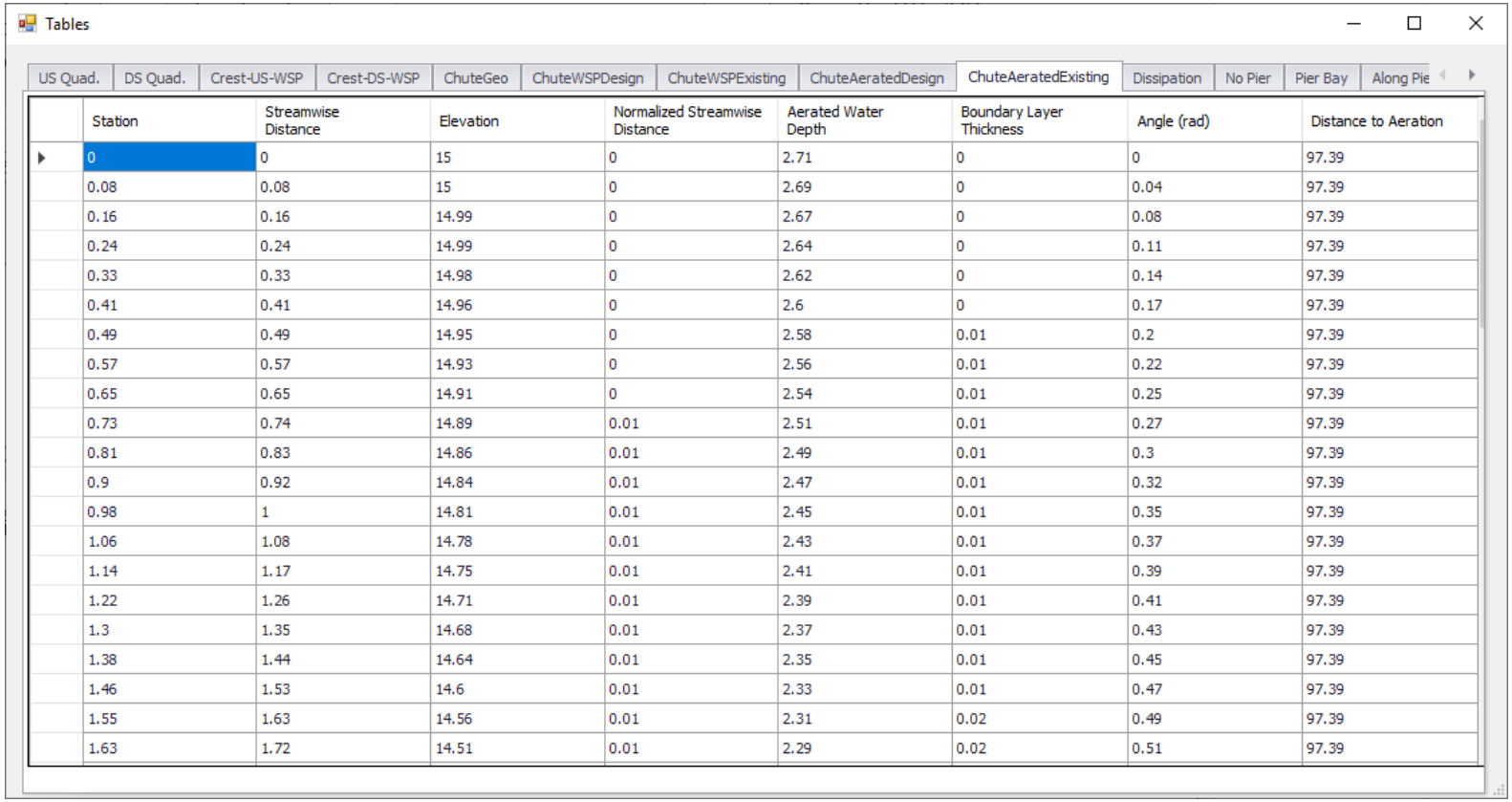
Table 4.8. Case 1 Water Surface Profile Along Chute Channel (Design Discharge, Approach 2)

Tables

US Quad. | DS Quad. | Crest-US-WSP | Crest-DS-WSP | ChuteGeo | ChuteWSPDesign | ChuteWSPExisting | ChuteAeratedDesign | ChuteAeratedExisting | Dissipation | No Pier | Pier Bay | Along Pie

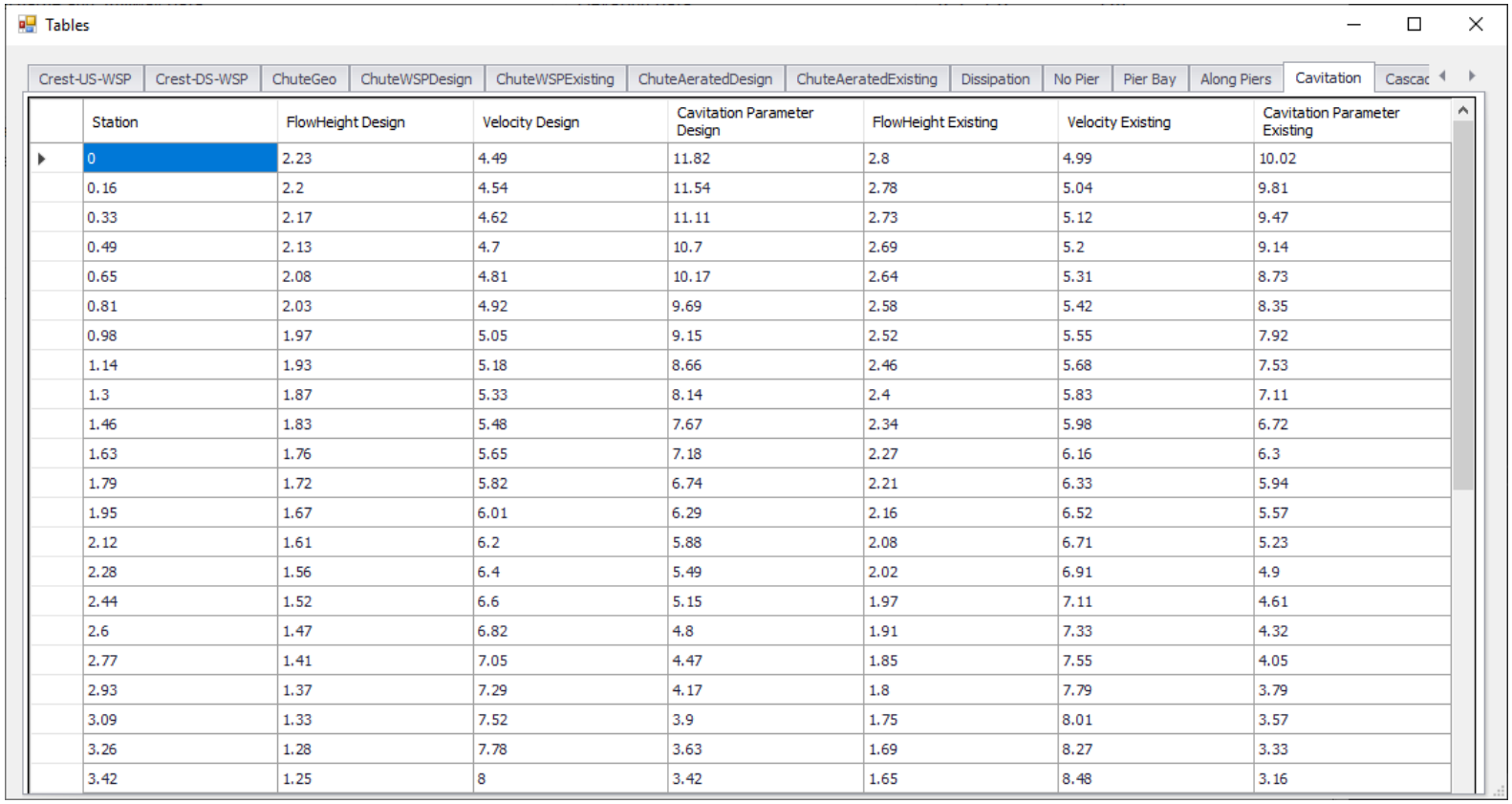
Station	Streamwise Distance	Elevation	Normalized Streamwise Distance	Aerated Water Depth	Boundary Layer Thickness	Angle (rad)	Distance to Aeration
0	0	15	0	2.17	0	0	76.44
0.08	0.08	15	0	2.14	0	0.04	76.44
0.16	0.16	14.99	0	2.12	0	0.08	76.44
0.24	0.24	14.99	0	2.1	0	0.11	76.44
0.33	0.33	14.98	0	2.07	0	0.14	76.44
0.41	0.41	14.96	0	2.05	0	0.17	76.44
0.49	0.49	14.95	0	2.03	0.01	0.2	76.44
0.57	0.57	14.93	0.01	2.01	0.01	0.22	76.44
0.65	0.65	14.91	0.01	1.99	0.01	0.25	76.44
0.73	0.74	14.89	0.01	1.97	0.01	0.27	76.44
0.81	0.83	14.86	0.01	1.95	0.01	0.3	76.44
0.9	0.92	14.84	0.01	1.93	0.01	0.32	76.44
0.98	1	14.81	0.01	1.91	0.01	0.35	76.44
1.06	1.08	14.78	0.01	1.89	0.01	0.37	76.44
1.14	1.17	14.75	0.01	1.87	0.01	0.39	76.44
1.22	1.26	14.71	0.01	1.85	0.01	0.41	76.44
1.3	1.35	14.68	0.01	1.83	0.01	0.43	76.44
1.38	1.44	14.64	0.01	1.81	0.01	0.45	76.44
1.46	1.53	14.6	0.01	1.8	0.01	0.47	76.44
1.55	1.63	14.56	0.02	1.78	0.02	0.49	76.44
1.63	1.72	14.51	0.02	1.76	0.02	0.51	76.44

Table 4.9. Case 1 Water Surface Profile Along Chute Channel (Existing Discharge, Approach 2)



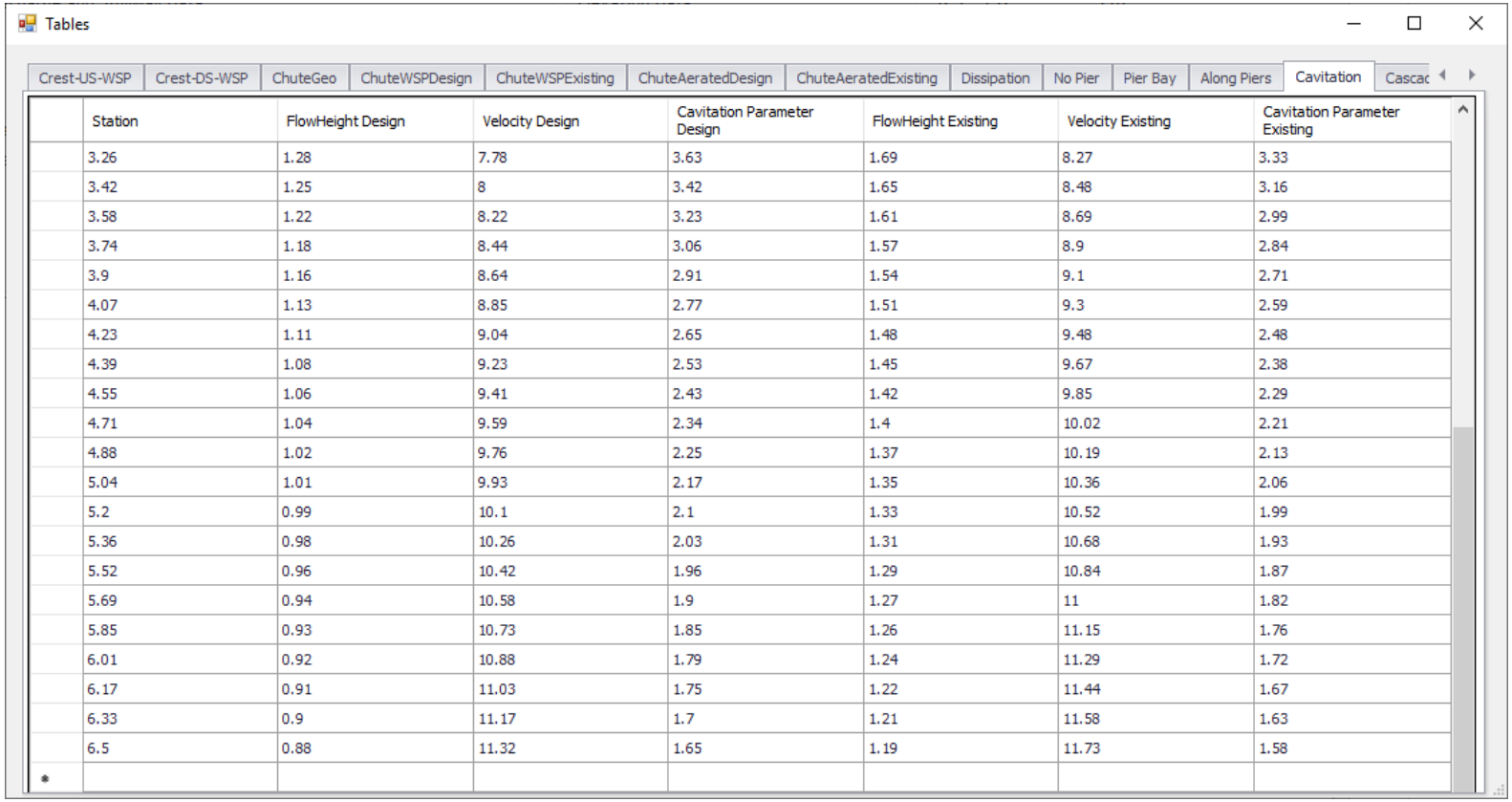
Station	Streamwise Distance	Elevation	Normalized Streamwise Distance	Aerated Water Depth	Boundary Layer Thickness	Angle (rad)	Distance to Aeration
0	0	15	0	2.71	0	0	97.39
0.08	0.08	15	0	2.69	0	0.04	97.39
0.16	0.16	14.99	0	2.67	0	0.08	97.39
0.24	0.24	14.99	0	2.64	0	0.11	97.39
0.33	0.33	14.98	0	2.62	0	0.14	97.39
0.41	0.41	14.96	0	2.6	0	0.17	97.39
0.49	0.49	14.95	0	2.58	0.01	0.2	97.39
0.57	0.57	14.93	0	2.56	0.01	0.22	97.39
0.65	0.65	14.91	0	2.54	0.01	0.25	97.39
0.73	0.74	14.89	0.01	2.51	0.01	0.27	97.39
0.81	0.83	14.86	0.01	2.49	0.01	0.3	97.39
0.9	0.92	14.84	0.01	2.47	0.01	0.32	97.39
0.98	1	14.81	0.01	2.45	0.01	0.35	97.39
1.06	1.08	14.78	0.01	2.43	0.01	0.37	97.39
1.14	1.17	14.75	0.01	2.41	0.01	0.39	97.39
1.22	1.26	14.71	0.01	2.39	0.01	0.41	97.39
1.3	1.35	14.68	0.01	2.37	0.01	0.43	97.39
1.38	1.44	14.64	0.01	2.35	0.01	0.45	97.39
1.46	1.53	14.6	0.01	2.33	0.01	0.47	97.39
1.55	1.63	14.56	0.01	2.31	0.02	0.49	97.39
1.63	1.72	14.51	0.01	2.29	0.02	0.51	97.39

Table 4.10. Case 1 Cavitation Parameters



Station	FlowHeight Design	Velocity Design	Cavitation Parameter Design	FlowHeight Existing	Velocity Existing	Cavitation Parameter Existing
0	2.23	4.49	11.82	2.8	4.99	10.02
0.16	2.2	4.54	11.54	2.78	5.04	9.81
0.33	2.17	4.62	11.11	2.73	5.12	9.47
0.49	2.13	4.7	10.7	2.69	5.2	9.14
0.65	2.08	4.81	10.17	2.64	5.31	8.73
0.81	2.03	4.92	9.69	2.58	5.42	8.35
0.98	1.97	5.05	9.15	2.52	5.55	7.92
1.14	1.93	5.18	8.66	2.46	5.68	7.53
1.3	1.87	5.33	8.14	2.4	5.83	7.11
1.46	1.83	5.48	7.67	2.34	5.98	6.72
1.63	1.76	5.65	7.18	2.27	6.16	6.3
1.79	1.72	5.82	6.74	2.21	6.33	5.94
1.95	1.67	6.01	6.29	2.16	6.52	5.57
2.12	1.61	6.2	5.88	2.08	6.71	5.23
2.28	1.56	6.4	5.49	2.02	6.91	4.9
2.44	1.52	6.6	5.15	1.97	7.11	4.61
2.6	1.47	6.82	4.8	1.91	7.33	4.32
2.77	1.41	7.05	4.47	1.85	7.55	4.05
2.93	1.37	7.29	4.17	1.8	7.79	3.79
3.09	1.33	7.52	3.9	1.75	8.01	3.57
3.26	1.28	7.78	3.63	1.69	8.27	3.33
3.42	1.25	8	3.42	1.65	8.48	3.16

Table 4.10. Case 1 Cavitation Parameters, Continued



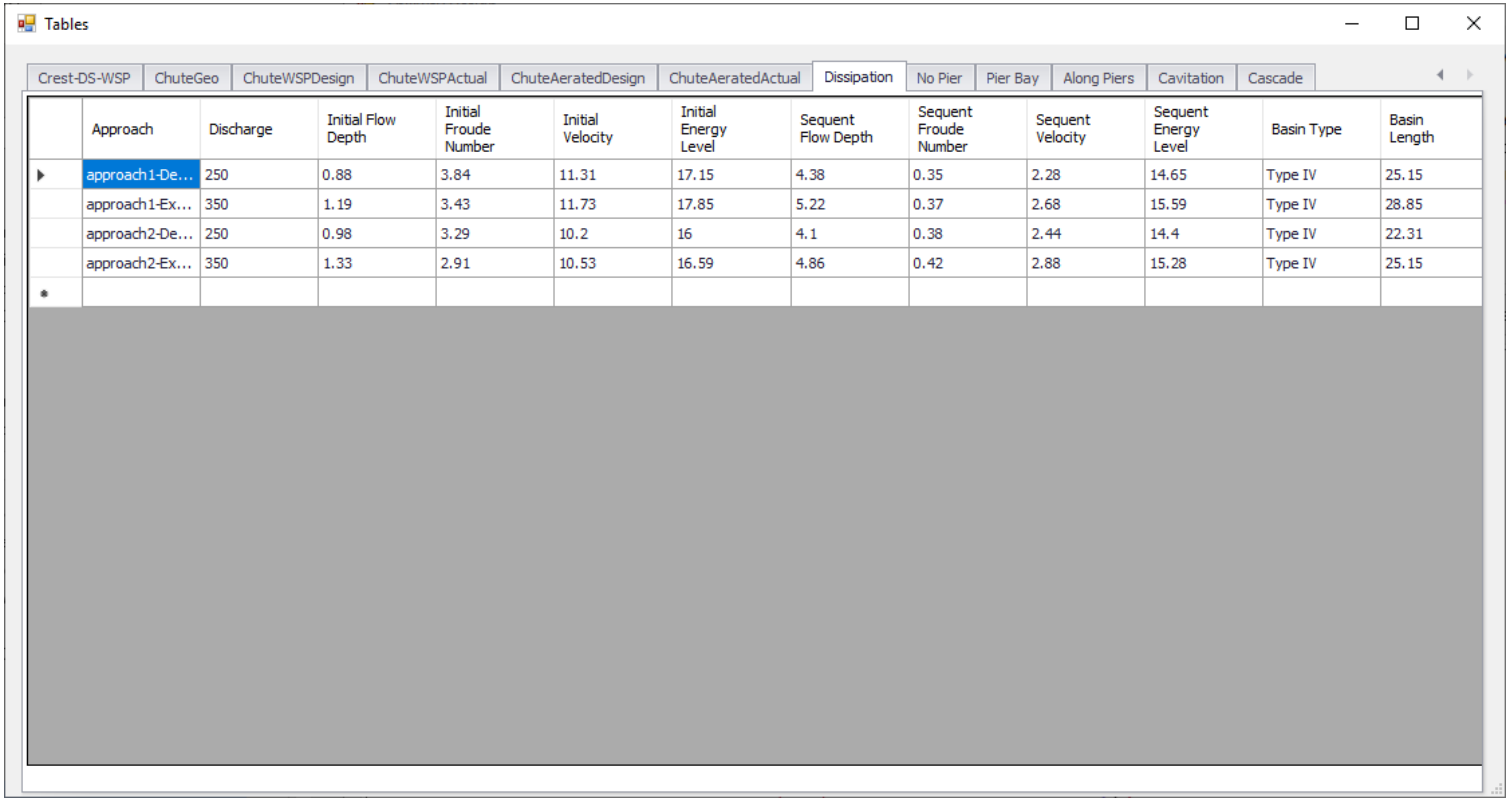
Station	FlowHeight Design	Velocity Design	Cavitation Parameter Design	FlowHeight Existing	Velocity Existing	Cavitation Parameter Existing
3.26	1.28	7.78	3.63	1.69	8.27	3.33
3.42	1.25	8	3.42	1.65	8.48	3.16
3.58	1.22	8.22	3.23	1.61	8.69	2.99
3.74	1.18	8.44	3.06	1.57	8.9	2.84
3.9	1.16	8.64	2.91	1.54	9.1	2.71
4.07	1.13	8.85	2.77	1.51	9.3	2.59
4.23	1.11	9.04	2.65	1.48	9.48	2.48
4.39	1.08	9.23	2.53	1.45	9.67	2.38
4.55	1.06	9.41	2.43	1.42	9.85	2.29
4.71	1.04	9.59	2.34	1.4	10.02	2.21
4.88	1.02	9.76	2.25	1.37	10.19	2.13
5.04	1.01	9.93	2.17	1.35	10.36	2.06
5.2	0.99	10.1	2.1	1.33	10.52	1.99
5.36	0.98	10.26	2.03	1.31	10.68	1.93
5.52	0.96	10.42	1.96	1.29	10.84	1.87
5.69	0.94	10.58	1.9	1.27	11	1.82
5.85	0.93	10.73	1.85	1.26	11.15	1.76
6.01	0.92	10.88	1.79	1.24	11.29	1.72
6.17	0.91	11.03	1.75	1.22	11.44	1.67
6.33	0.9	11.17	1.7	1.21	11.58	1.63
6.5	0.88	11.32	1.65	1.19	11.73	1.58

Froude number at the end of the chute is 3.89 for the design discharge and 3.46 for the existing discharge in approach 1. In approach 2, Froude number at the end of the chute is 3.29 for the design discharge and 2.91 for the existing discharge. The sequent depth and energy level at downstream of the hydraulic jump will be different for each approach and discharge. USBR Type IV stilling basin is proper for both discharges and approaches.

As shown in Table 4.11 the energy level at the sequent depth section is calculated as 15.60 m in approach 1, and 15.28 m in approach 2 for existing discharge. Tailwater energy level is 13 m for this case. Therefore, dissipated energy level is still higher than the tailwater level. Adjust stilling basin elevation feature should be used to determine the right stilling basin elevation.

As shown in Figure 4.3, stilling basin initial elevation was equal to the downstream bed elevation, which is 10 m. In order to keep the downstream energy level lower than the tailwater level, stilling basin elevation is decreased to 6.80 m (See Figure 4.4).

Table 4.11. Hydraulic Jump Table for Case 1



Approach	Discharge	Initial Flow Depth	Initial Froude Number	Initial Velocity	Initial Energy Level	Sequent Flow Depth	Sequent Froude Number	Sequent Velocity	Sequent Energy Level	Basin Type	Basin Length
approach1-De...	250	0.88	3.84	11.31	17.15	4.38	0.35	2.28	14.65	Type IV	25.15
approach1-Ex...	350	1.19	3.43	11.73	17.85	5.22	0.37	2.68	15.59	Type IV	28.85
approach2-De...	250	0.98	3.29	10.2	16	4.1	0.38	2.44	14.4	Type IV	22.31
approach2-Ex...	350	1.33	2.91	10.53	16.59	4.86	0.42	2.88	15.28	Type IV	25.15

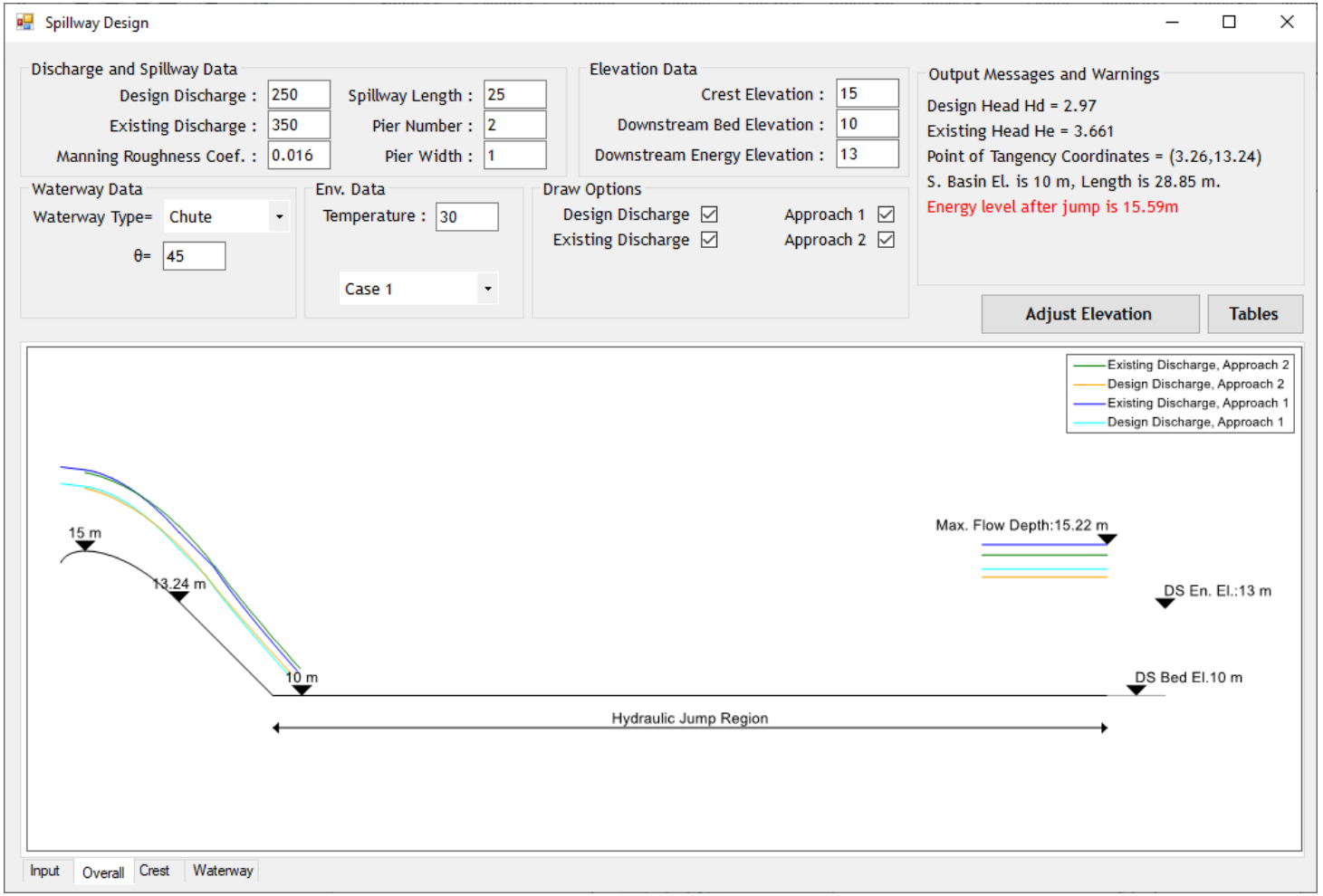


Figure 4.3. Case 1 – Initial Design

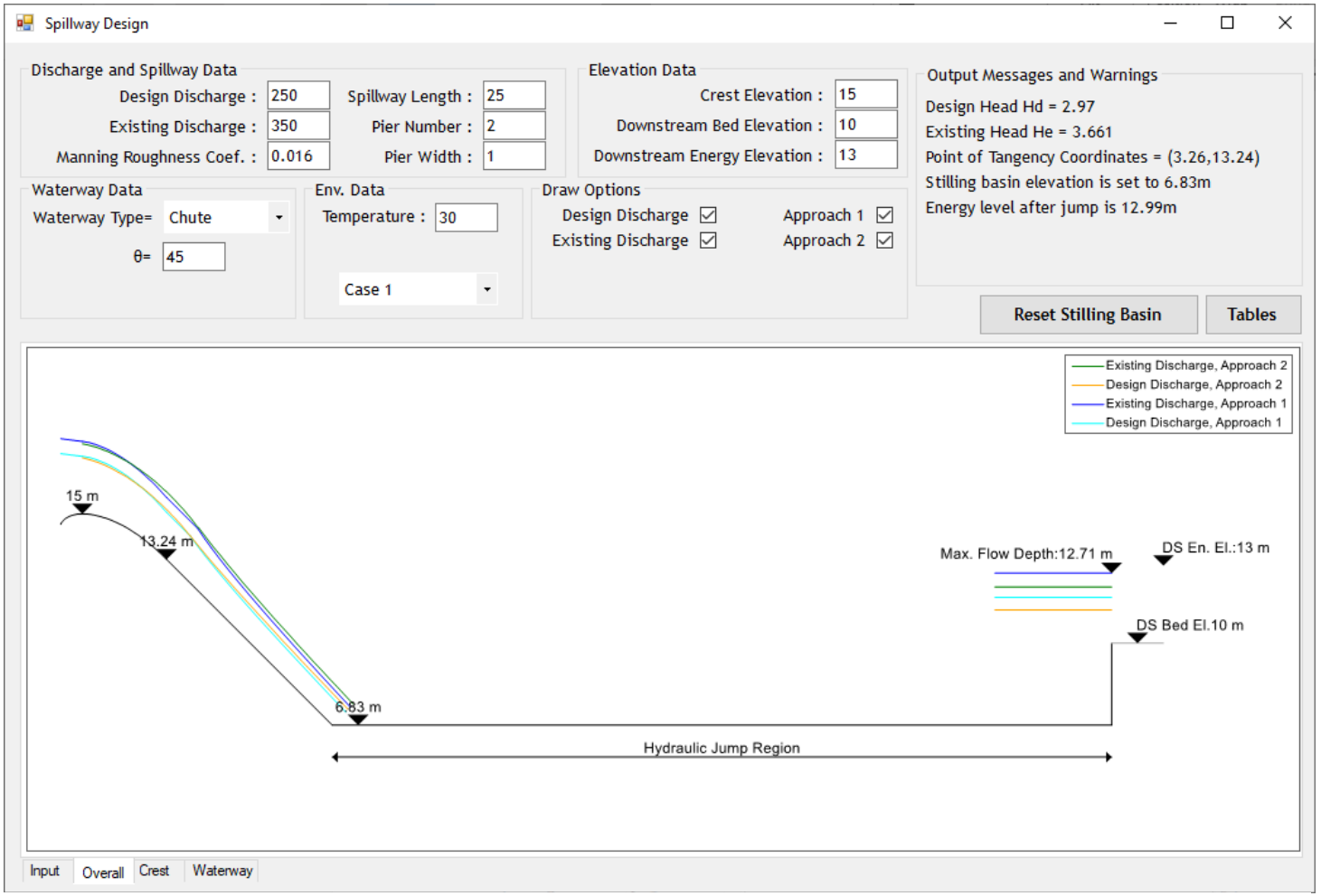


Figure 4.4. Case 1 – Adjusted Design

4.4. Results of Case 2

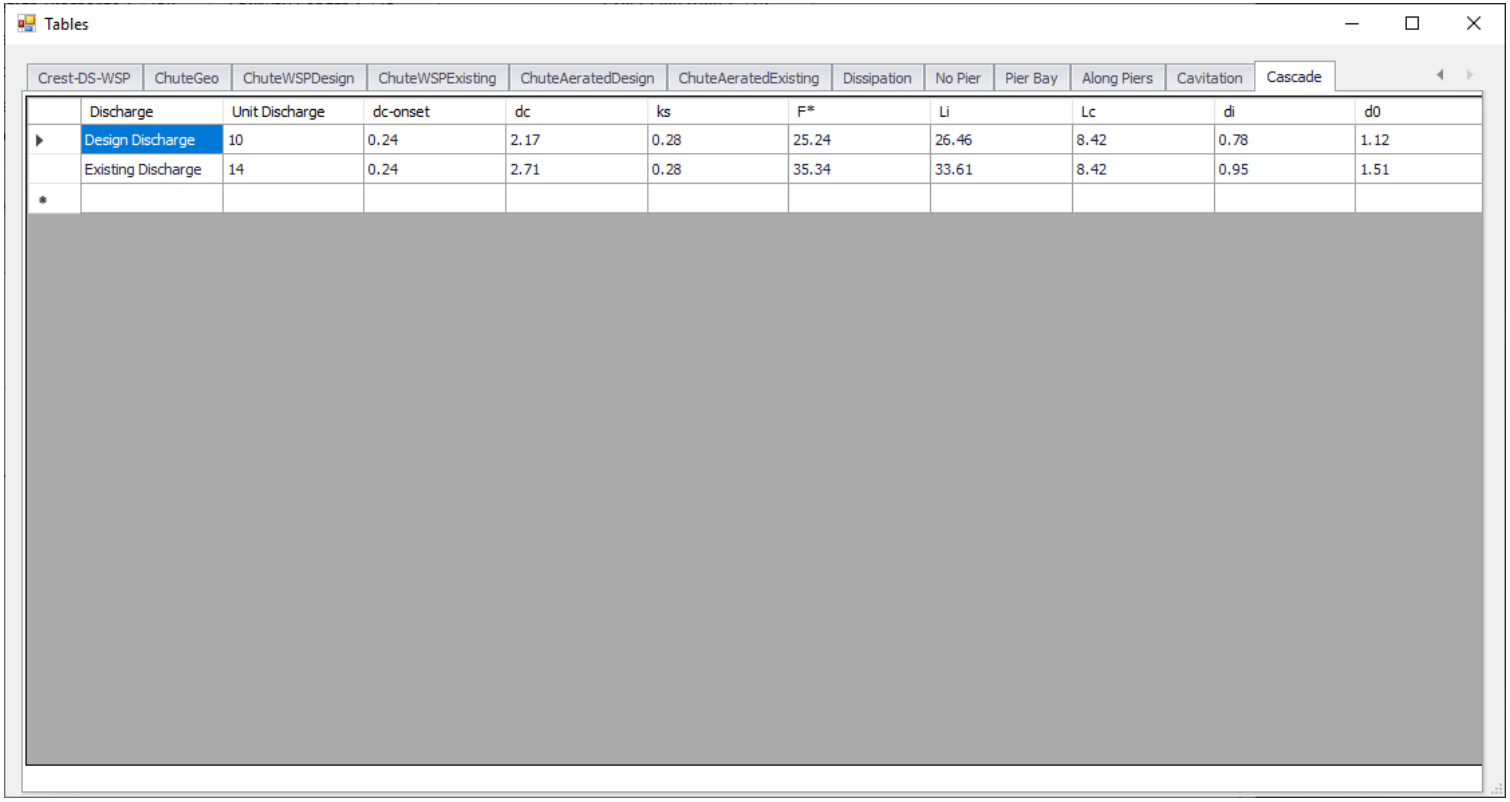
Case 2 represents a low head stepped spillway. It has only one difference than Case 1, which is the type of waterway. Crest bed profile and water surface profile over the crest is the same as it is in Case 1. Vertical flow depth at the beginning of the chute channel is 1.82 m for the design discharge and 2.40 m for the existing discharge. Flow depth perpendicular to channel bottom is calculated as 1.287 m for the design discharge and 1.697 m for the existing discharge.

In order to specify the flow type over the stepped channel, characteristic critical flow depth, $(d_c)_{onset}$, is computed as 0.24 m. Critical flow depth, (d_c) is 2.17 m and 2.71 m for the design and the existing discharge, respectively. Therefore, skimming flow is occurred. Inception point of aeration is found 26.46 m and 33.61 m away from the crest for the design and the existing discharge, respectively. From the crest to the toe of the spillway, there is only 8.42 m length available. Therefore, the spillway is not long enough to start aeration. As shown in Figure 4.6 the user is warned about this situation.

As shown in Table 4.13, Froude number at the end of the chute is 2.69 for the design discharge and 2.40 for the existing discharge. The sequent depth and the energy level at downstream of the the hydraulic jump will be different for each approach and discharge. USBR Type IV stilling basin is proper for design discharge, where USBR Type I is proper for existing discharge.

Another warning is given for downstream energy level. The stilling basin should be placed on a lower elevation. Figure 4.7 shows the adjusted design for Case 2. As shown in Figure 4.6, stilling basin initial elevation was equal to downstream bed elevation which is 10 m. In order to keep downstream energy level lower than the tailwater level, stilling basin elevation is decreased to 7.94 m (See Figure 4.7).

Table 4.12. Case 2 Stepped Spillway Calculations



	Discharge	Unit Discharge	dc-onset	dc	ks	F*	Li	Lc	di	d0
▶	Design Discharge	10	0.24	2.17	0.28	25.24	26.46	8.42	0.78	1.12
	Existing Discharge	14	0.24	2.71	0.28	35.34	33.61	8.42	0.95	1.51
*										

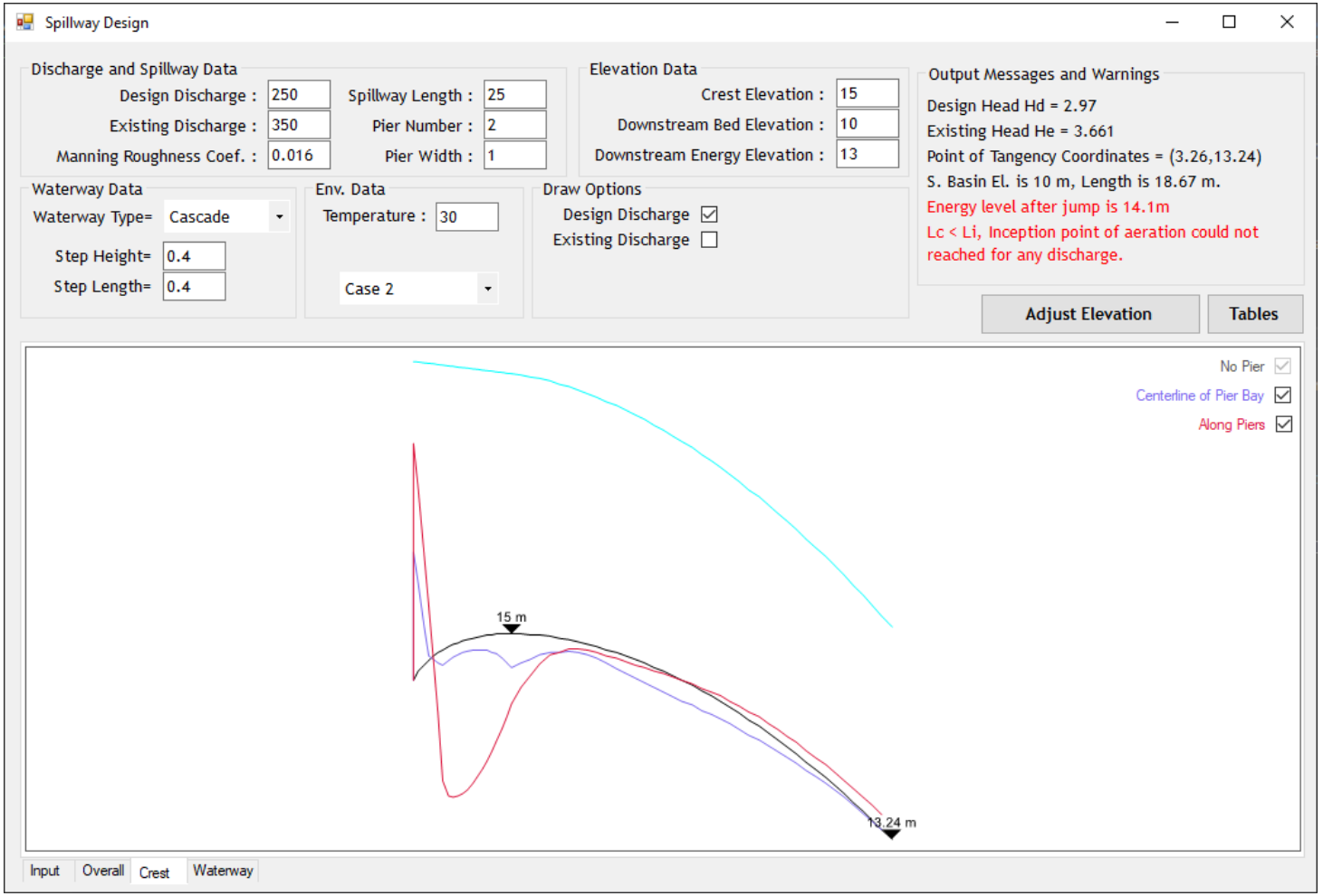
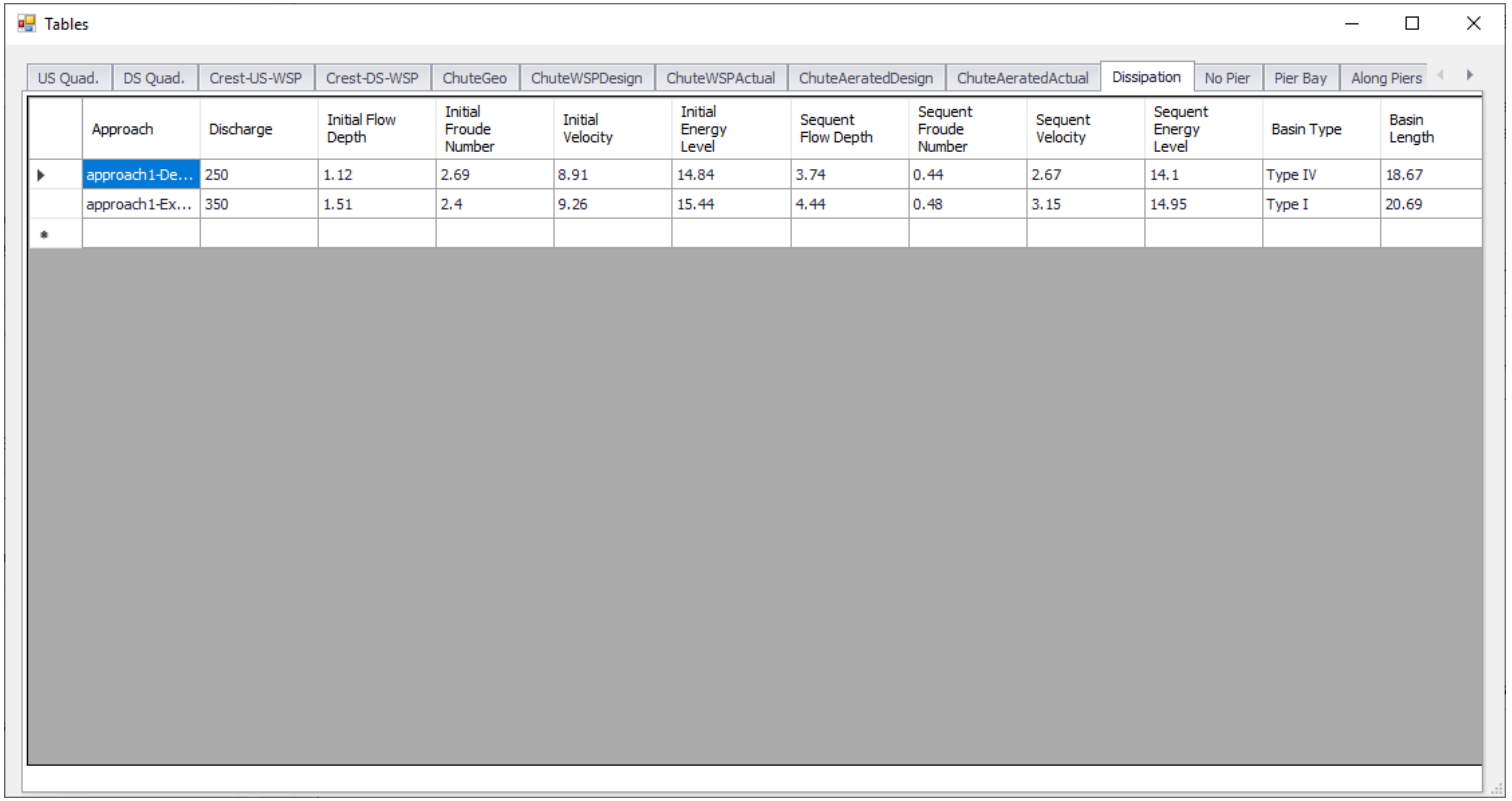


Figure 4.5. Case 2 – Crest Pressures

Table 4.13. Hydraulic Jump Table for Case 2



Approach	Discharge	Initial Flow Depth	Initial Froude Number	Initial Velocity	Initial Energy Level	Sequent Flow Depth	Sequent Froude Number	Sequent Velocity	Sequent Energy Level	Basin Type	Basin Length
approach1-De...	250	1.12	2.69	8.91	14.84	3.74	0.44	2.67	14.1	Type IV	18.67
approach1-Ex...	350	1.51	2.4	9.26	15.44	4.44	0.48	3.15	14.95	Type I	20.69

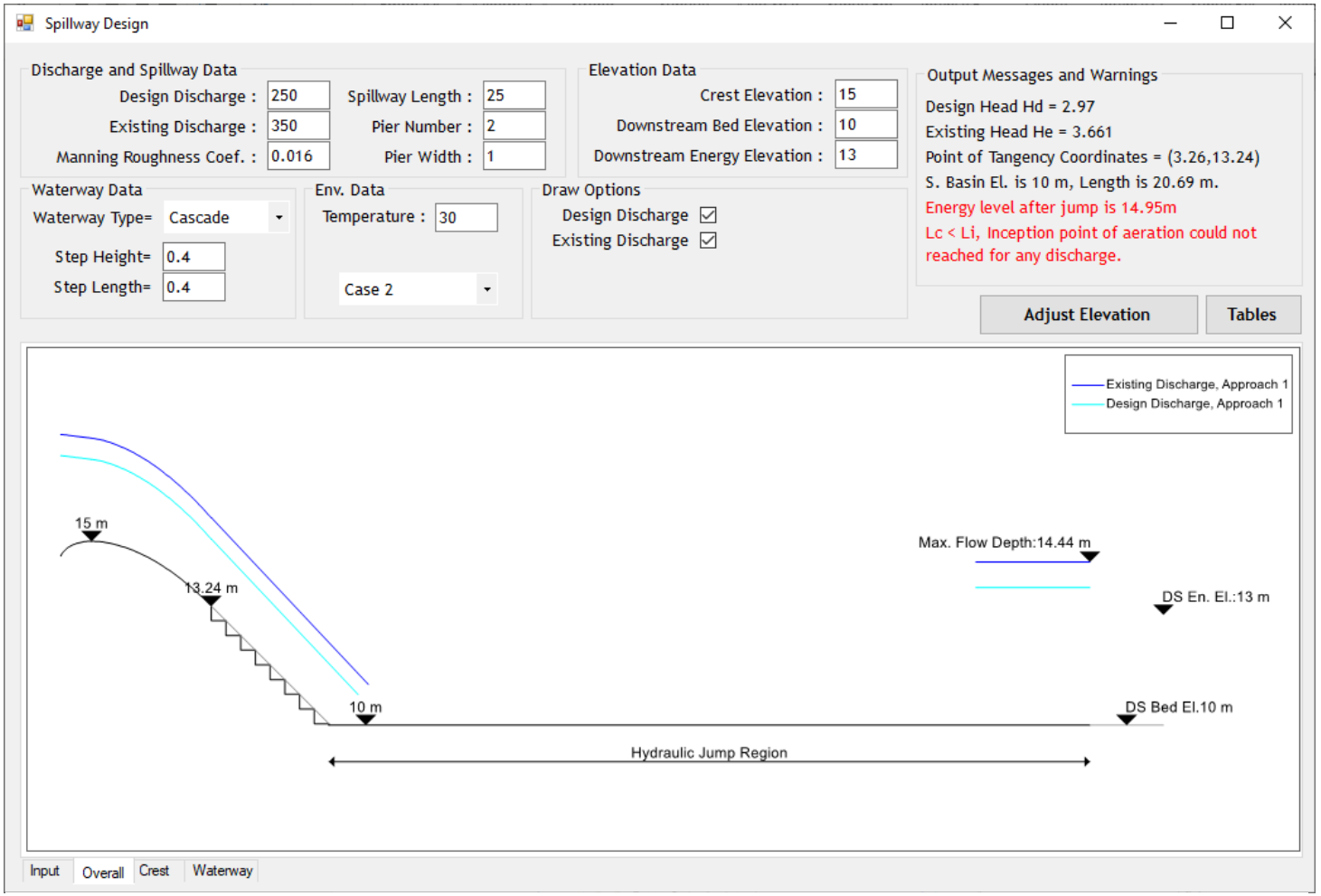


Figure 4.6. Case 2 – Initial Design

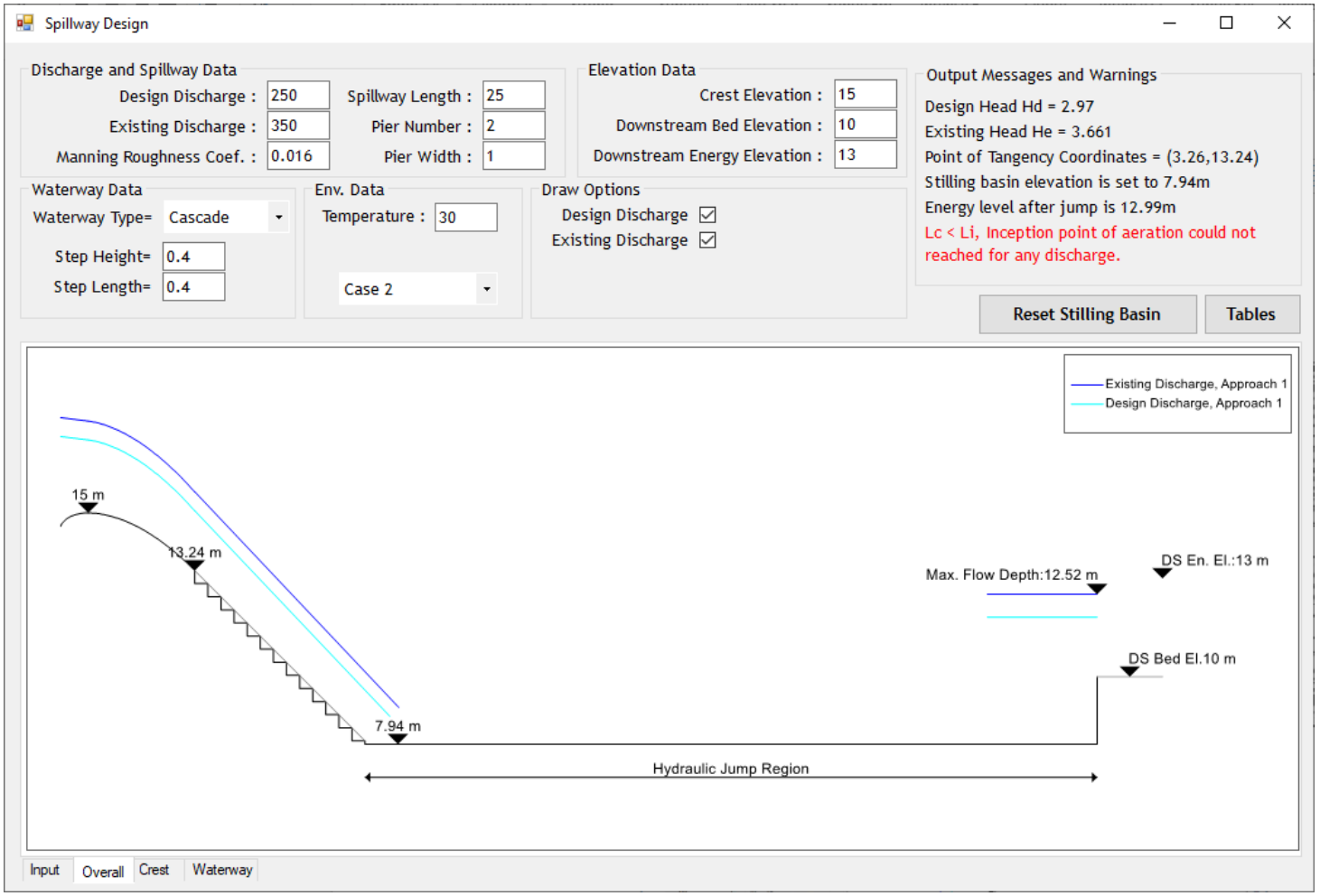


Figure 4.7. Case 2 – Adjusted Design

4.5. Results of Case 3

Case 3 represents a high head chute spillway. With the given data, design head is calculated as 6.90 m whereas the existing head is 8.28 m. As such in Case 1, the existing discharge is greater than the design discharge. Hence, negative pressure on the crest is expected. Crest pressures are shown in Figure 4.8. Crest bed profile is determined for the calculated design head, and the results are given for upstream and downstream quadrants in Table 4.14 and Table 4.15, respectively. Point of tangency coordinates are (7.56, 65.91). Water surface profile along the crest for the design head and the existing head are computed. The results are given in Table 4.16 and Table 4.17, respectively. When approach 1 is used, vertical flow depth at the beginning of the chute channel is 4.23 m for the design discharge and 5.37 m for the existing discharge. Flow depth perpendicular to channel bottom is calculated 2.99 m for the design discharge and 3.80 m for the existing discharge. Standard step method is applied from this point to the end of the chute. Therefore, 1.10 m flow depth is computed at the end of chute for the design discharge and 1.44 m flow depth is computed for the existing discharge. Results for design discharge and existing discharge are given in Table 4.18 and Table 4.19, respectively.

In approach 2, 2.94 m and 3.80 m flow depths are computed for the design discharge and the existing discharge at the point of tangency, respectively. With this approach, 1.54 m and 1.90 m flow depths are computed at the end of the chute for the design and the existing discharge, respectively. Inception point of aeration is located at 185.59 m and 228.29 m for design discharge and existing discharge, respectively. Turbulent boundary layer does not reach the free surface for both discharges. Results for design discharge and existing discharge are given in Table 4.20 and Table 4.21, respectively.

Cavitation parameters along the crest are calculated. The results are given in Table 4.22. As shown in Figure 4.9, stilling basin initial elevation was equal to downstream bed elevation which is 20 m. In order to keep downstream energy level lower than the tailwater level, stilling basin elevation is decreased to 12.62 m (See Figure 4.10).

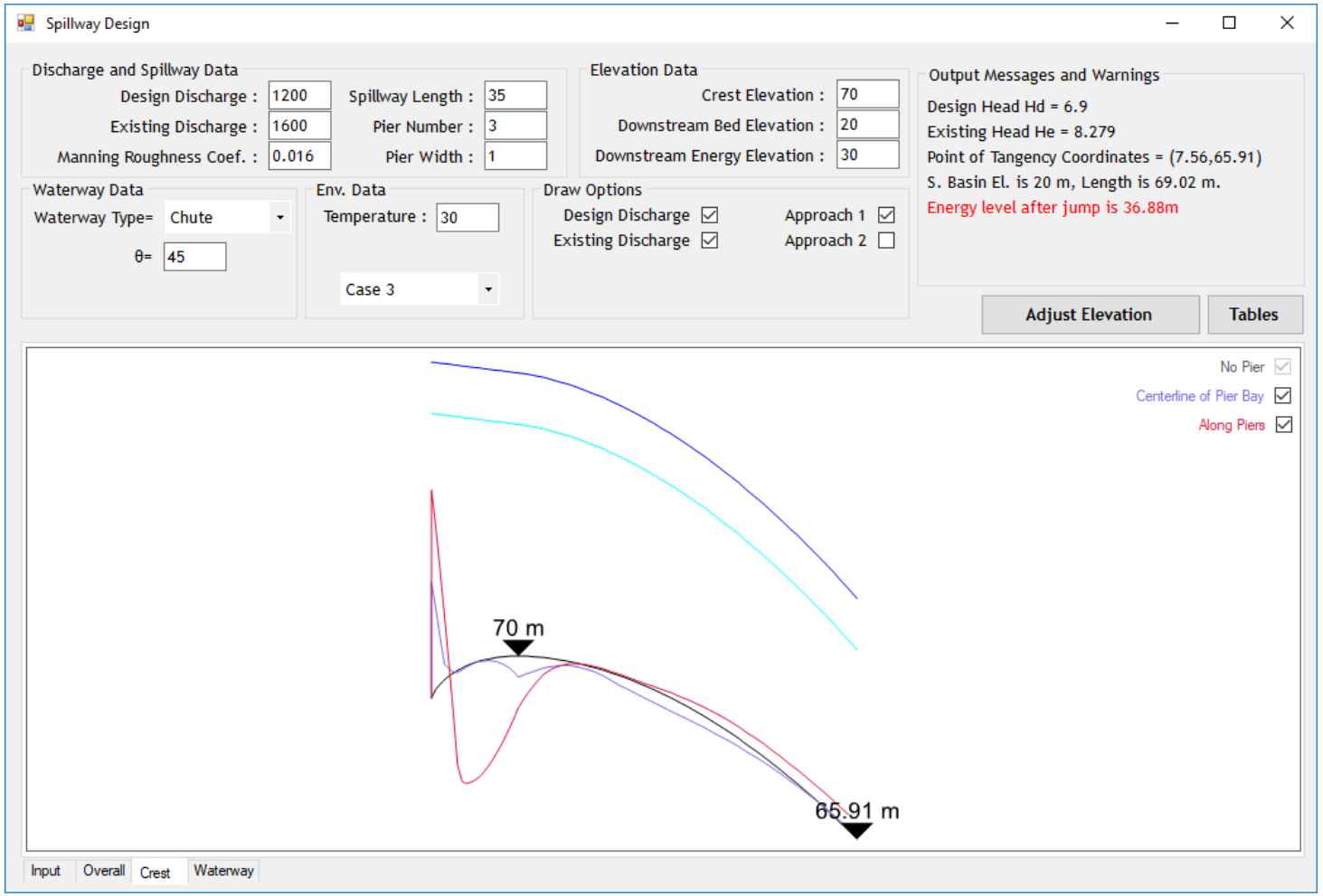


Figure 4.8. Case 3 – Crest Pressures

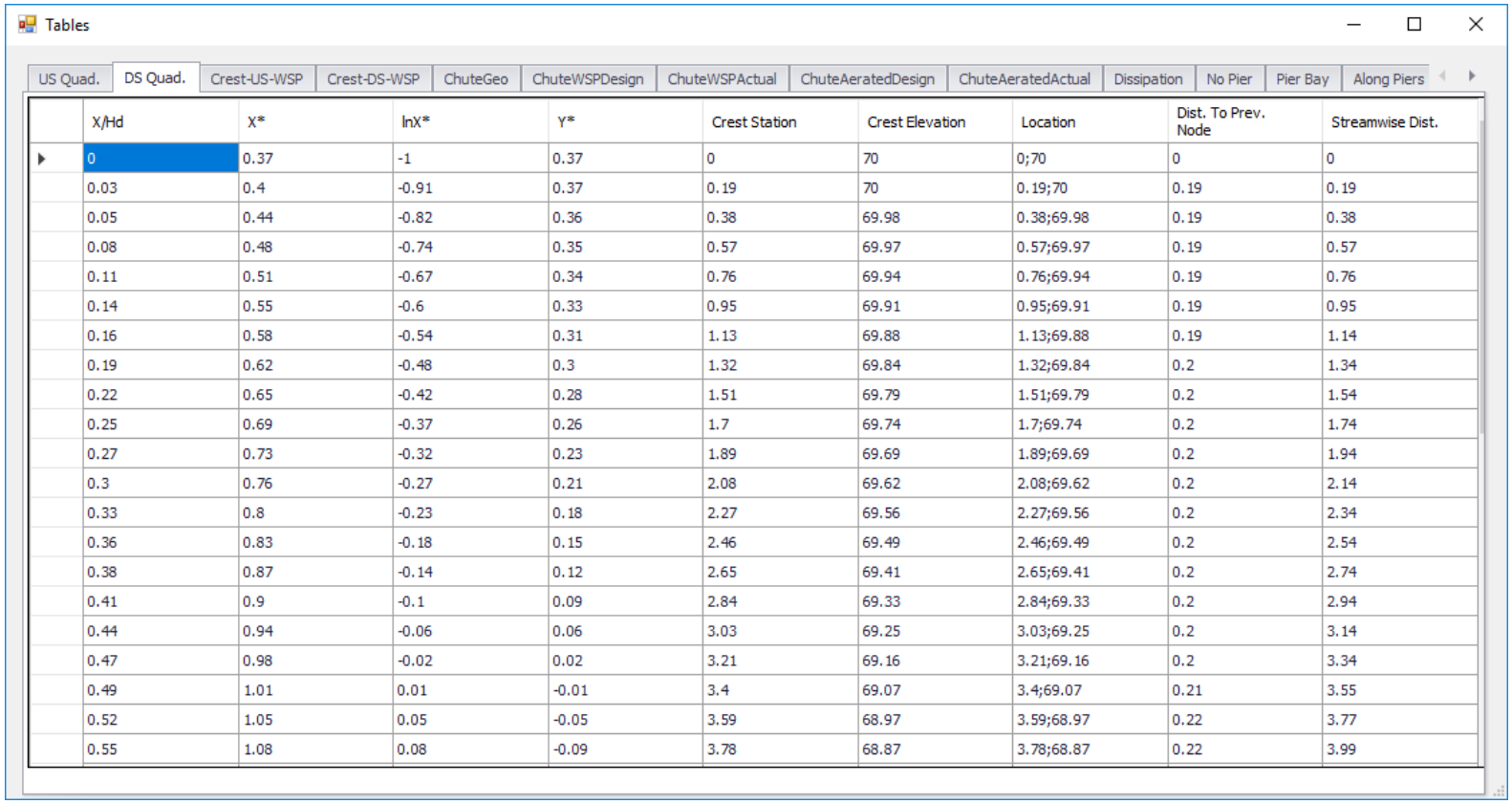
Table 4.14. Case 3 Crest Upstream Quadrant Bed Profile

Tables

US Quad. | DS Quad. | Crest-US-WSP | Crest-DS-WSP | ChuteGeo | ChuteWSPDesign | ChuteWSPActual | ChuteAeratedDesign | ChuteAeratedActual | Dissipation | No Pier | Pier Bay | Along Piers

X/Hd	X*	lnX*	Y*	Crest Station	Crest Elevation	Location
0	0.37	-1	0.37	0	70	0;70
-0.01	0.35	-1.05	0.37	-0.1	70	-0.1;70
-0.03	0.33	-1.11	0.37	-0.19	70	-0.19;70
-0.04	0.31	-1.16	0.36	-0.29	69.99	-0.29;69.99
-0.06	0.29	-1.22	0.36	-0.39	69.98	-0.39;69.98
-0.07	0.28	-1.29	0.36	-0.49	69.97	-0.49;69.97
-0.08	0.26	-1.36	0.35	-0.58	69.95	-0.58;69.95
-0.1	0.24	-1.43	0.34	-0.68	69.93	-0.68;69.93
-0.11	0.22	-1.51	0.33	-0.78	69.91	-0.78;69.91
-0.13	0.2	-1.6	0.32	-0.87	69.89	-0.87;69.89
-0.14	0.18	-1.69	0.31	-0.97	69.86	-0.97;69.86
-0.15	0.17	-1.8	0.3	-1.07	69.82	-1.07;69.82
-0.17	0.15	-1.92	0.28	-1.17	69.78	-1.17;69.78
-0.18	0.13	-2.05	0.26	-1.26	69.73	-1.26;69.73
-0.2	0.11	-2.2	0.24	-1.36	69.68	-1.36;69.68
-0.21	0.09	-2.39	0.22	-1.46	69.62	-1.46;69.62
-0.23	0.07	-2.61	0.19	-1.56	69.55	-1.56;69.55
-0.24	0.06	-2.9	0.16	-1.65	69.47	-1.65;69.47
-0.25	0.04	-3.3	0.12	-1.75	69.37	-1.75;69.37
-0.27	0.02	-4	0.07	-1.85	69.25	-1.85;69.25
-0.28	0	-22.76	0	-1.94	69.06	-1.94;69.06

Table 4.15. Case 3 Crest Downstream Quadrant Bed Profile



X/Hd	X*	lnX*	Y*	Crest Station	Crest Elevation	Location	Dist. To Prev. Node	Streamwise Dist.
0	0.37	-1	0.37	0	70	0;70	0	0
0.03	0.4	-0.91	0.37	0.19	70	0.19;70	0.19	0.19
0.05	0.44	-0.82	0.36	0.38	69.98	0.38;69.98	0.19	0.38
0.08	0.48	-0.74	0.35	0.57	69.97	0.57;69.97	0.19	0.57
0.11	0.51	-0.67	0.34	0.76	69.94	0.76;69.94	0.19	0.76
0.14	0.55	-0.6	0.33	0.95	69.91	0.95;69.91	0.19	0.95
0.16	0.58	-0.54	0.31	1.13	69.88	1.13;69.88	0.19	1.14
0.19	0.62	-0.48	0.3	1.32	69.84	1.32;69.84	0.2	1.34
0.22	0.65	-0.42	0.28	1.51	69.79	1.51;69.79	0.2	1.54
0.25	0.69	-0.37	0.26	1.7	69.74	1.7;69.74	0.2	1.74
0.27	0.73	-0.32	0.23	1.89	69.69	1.89;69.69	0.2	1.94
0.3	0.76	-0.27	0.21	2.08	69.62	2.08;69.62	0.2	2.14
0.33	0.8	-0.23	0.18	2.27	69.56	2.27;69.56	0.2	2.34
0.36	0.83	-0.18	0.15	2.46	69.49	2.46;69.49	0.2	2.54
0.38	0.87	-0.14	0.12	2.65	69.41	2.65;69.41	0.2	2.74
0.41	0.9	-0.1	0.09	2.84	69.33	2.84;69.33	0.2	2.94
0.44	0.94	-0.06	0.06	3.03	69.25	3.03;69.25	0.2	3.14
0.47	0.98	-0.02	0.02	3.21	69.16	3.21;69.16	0.2	3.34
0.49	1.01	0.01	-0.01	3.4	69.07	3.4;69.07	0.21	3.55
0.52	1.05	0.05	-0.05	3.59	68.97	3.59;68.97	0.22	3.77
0.55	1.08	0.08	-0.09	3.78	68.87	3.78;68.87	0.22	3.99

Table 4.16. Case 3 Water Surface Profile Along Crest Upstream Quadrant

Tables

US Quad. | DS Quad. | Crest-US-WSP | Crest-DS-WSP | ChuteGeo | ChuteWSPDesign | ChuteWSPActual | ChuteAeratedDesign | ChuteAeratedActual | Dissipation | No Pier | Pier Bay | Along Piers

X	z	S*design	S*existing	s-design	s-existing	slope	vel. Design	vel. Existing	hp Design	hp Existing
0	70	0.75	0.92	5.18	6.32	0	6.63	7.23	0	-1.45
-0.1	70	0.75	0.92	5.18	6.33	0	6.61	7.22	0.03	-1.43
-0.19	70	0.75	0.92	5.2	6.35	0	6.59	7.2	0.06	-1.4
-0.29	69.99	0.75	0.92	5.21	6.36	0	6.58	7.19	0.09	-1.36
-0.39	69.98	0.76	0.92	5.23	6.38	0	6.56	7.17	0.13	-1.33
-0.49	69.97	0.76	0.93	5.24	6.38	0	6.55	7.16	0.14	-1.39
-0.58	69.95	0.76	0.93	5.24	6.39	0	6.54	7.15	0.13	-1.46
-0.68	69.93	0.76	0.93	5.26	6.41	0	6.52	7.13	0.08	-1.55
-0.78	69.91	0.76	0.93	5.27	6.42	0	6.51	7.12	0.07	-1.63
-0.87	69.89	0.77	0.93	5.29	6.44	0	6.48	7.1	0.07	-1.71
-0.97	69.86	0.77	0.93	5.3	6.44	0	6.47	7.09	0.05	-1.79
-1.07	69.82	0.77	0.94	5.3	6.45	0	6.46	7.08	0.04	-1.88
-1.17	69.78	0.77	0.94	5.32	6.47	0	6.44	7.07	0.03	-1.98
-1.26	69.73	0.77	0.94	5.33	6.48	0	6.43	7.06	0.03	-1.92
-1.36	69.68	0.78	0.94	5.35	6.5	0	6.41	7.04	0.06	-1.85
-1.46	69.62	0.78	0.94	5.36	6.5	0	6.4	7.03	0.14	-1.76
-1.56	69.55	0.78	0.95	5.37	6.52	0	6.38	7.01	0.28	-1.64
-1.65	69.47	0.78	0.95	5.38	6.53	0	6.37	7	0.5	-1.5
-1.75	69.37	0.78	0.95	5.39	6.54	0	6.36	6.99	0.86	-1.17
-1.85	69.25	0.78	0.95	5.41	6.56	0	6.34	6.97	1.38	-0.49
-1.94	69.06	0.78	0.95	5.42	6.56	0	6.33	6.96	2.01	0.38

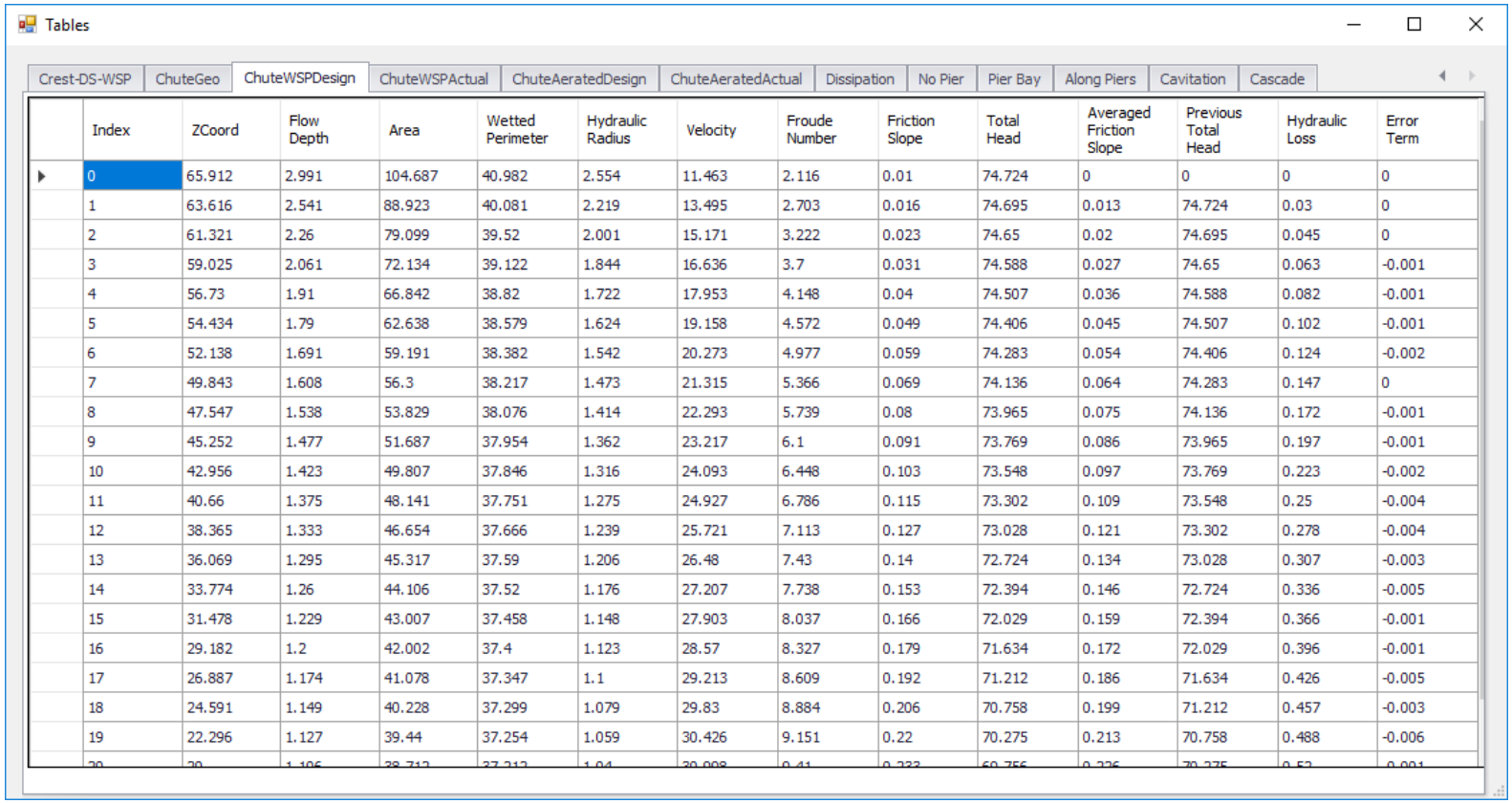
Table 4.17. Case 3 Water Surface Profile Along Crest Downstream Quadrant

Tables

US Quad. | DS Quad. | Crest-US-WSP | Crest-DS-WSP | ChuteGeo | ChuteWSPDesign | ChuteWSPActual | ChuteAeratedDesign | ChuteAeratedActual | Dissipation | No Pier | Pier Bay | Along Piers

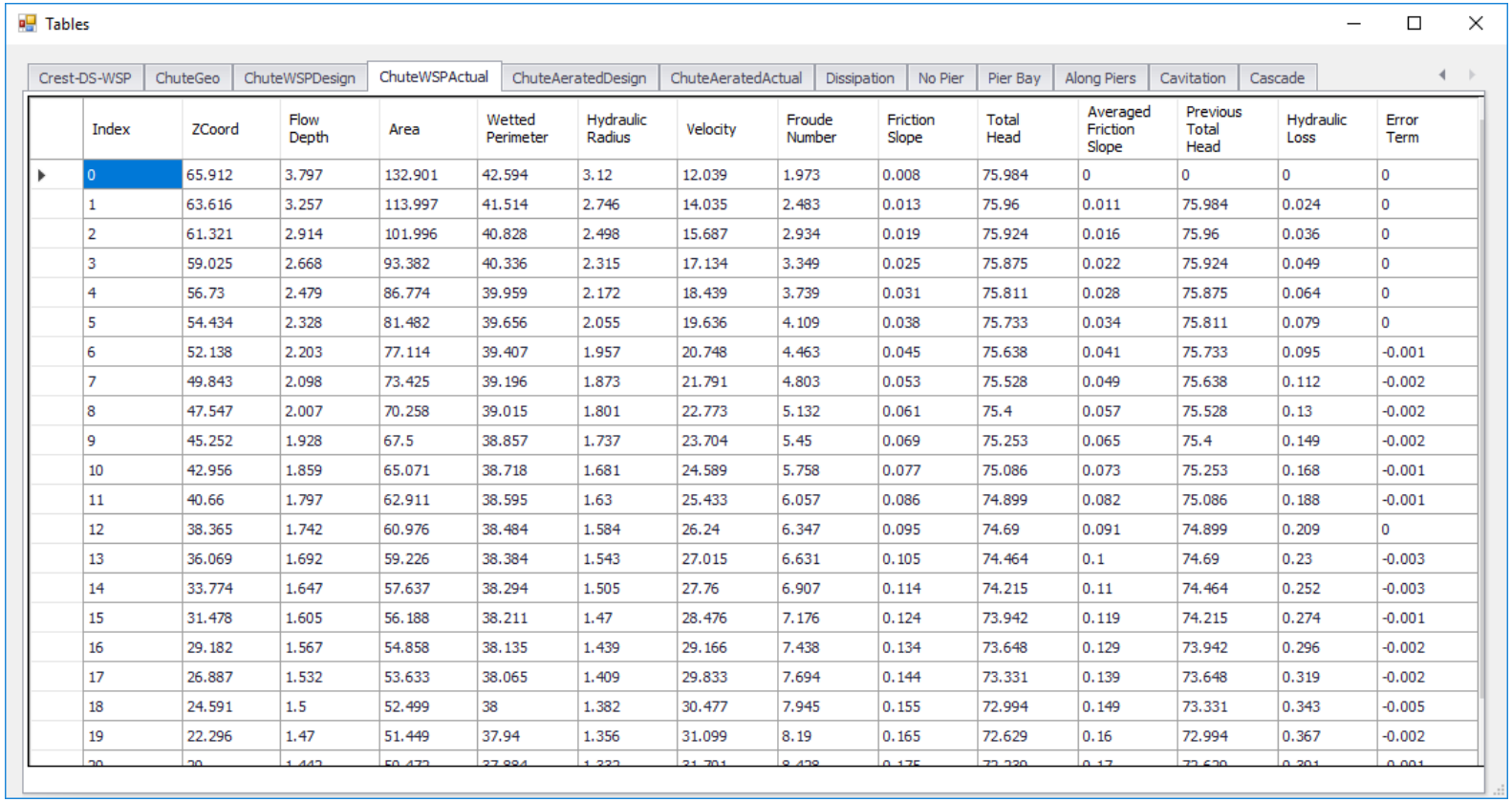
X	z	S*design	S*existing	s-design	s-existing	slope	vel. Design	vel. Existing	hp Design	hp Existing
0	70	0.75	0.92	5.18	6.32	0	6.63	7.23	0	-1.45
0.19	70	0.75	0.91	5.15	6.3	0.04	6.66	7.27	-0.01	-1.46
0.38	69.98	0.74	0.91	5.13	6.28	0.08	6.7	7.3	0.03	-1.41
0.57	69.97	0.74	0.91	5.11	6.25	0.11	6.76	7.35	0.11	-1.29
0.76	69.94	0.74	0.9	5.08	6.23	0.14	6.82	7.41	0.2	-1.1
0.95	69.91	0.73	0.9	5.05	6.2	0.17	6.88	7.48	0.27	-0.84
1.13	69.88	0.73	0.9	5.04	6.19	0.2	6.94	7.54	0.28	-0.75
1.32	69.84	0.73	0.89	5.01	6.16	0.22	7.02	7.61	0.27	-0.72
1.51	69.79	0.72	0.89	4.99	6.13	0.25	7.1	7.69	0.25	-0.7
1.7	69.74	0.72	0.89	4.96	6.11	0.27	7.18	7.77	0.23	-0.7
1.89	69.69	0.72	0.88	4.94	6.09	0.3	7.26	7.85	0.2	-0.7
2.08	69.62	0.71	0.88	4.92	6.06	0.32	7.35	7.95	0.19	-0.7
2.27	69.56	0.71	0.88	4.89	6.04	0.35	7.45	8.04	0.17	-0.71
2.46	69.49	0.7	0.87	4.86	6.01	0.37	7.55	8.15	0.16	-0.71
2.65	69.41	0.7	0.87	4.85	6	0.39	7.64	8.24	0.14	-0.7
2.84	69.33	0.7	0.87	4.82	5.97	0.41	7.75	8.35	0.14	-0.69
3.03	69.25	0.7	0.86	4.8	5.94	0.43	7.87	8.46	0.13	-0.68
3.21	69.16	0.69	0.86	4.77	5.92	0.45	7.98	8.58	0.13	-0.66
3.4	69.07	0.69	0.86	4.75	5.9	0.47	8.09	8.69	0.13	-0.63
3.59	68.97	0.68	0.85	4.73	5.87	0.49	8.21	8.81	0.14	-0.6
3.78	68.87	0.68	0.85	4.7	5.85	0.51	8.34	8.94	0.15	-0.56
3.97	68.76	0.68	0.84	4.67	5.82	0.52	8.47	9.07	0.16	-0.52

Table 4.18. Case 3 Water Surface Profile Along Chute Channel (Design Discharge, Approach 1)



Index	ZCoord	Flow Depth	Area	Wetted Perimeter	Hydraulic Radius	Velocity	Froude Number	Friction Slope	Total Head	Averaged Friction Slope	Previous Total Head	Hydraulic Loss	Error Term
0	65.912	2.991	104.687	40.982	2.554	11.463	2.116	0.01	74.724	0	0	0	0
1	63.616	2.541	88.923	40.081	2.219	13.495	2.703	0.016	74.695	0.013	74.724	0.03	0
2	61.321	2.26	79.099	39.52	2.001	15.171	3.222	0.023	74.65	0.02	74.695	0.045	0
3	59.025	2.061	72.134	39.122	1.844	16.636	3.7	0.031	74.588	0.027	74.65	0.063	-0.001
4	56.73	1.91	66.842	38.82	1.722	17.953	4.148	0.04	74.507	0.036	74.588	0.082	-0.001
5	54.434	1.79	62.638	38.579	1.624	19.158	4.572	0.049	74.406	0.045	74.507	0.102	-0.001
6	52.138	1.691	59.191	38.382	1.542	20.273	4.977	0.059	74.283	0.054	74.406	0.124	-0.002
7	49.843	1.608	56.3	38.217	1.473	21.315	5.366	0.069	74.136	0.064	74.283	0.147	0
8	47.547	1.538	53.829	38.076	1.414	22.293	5.739	0.08	73.965	0.075	74.136	0.172	-0.001
9	45.252	1.477	51.687	37.954	1.362	23.217	6.1	0.091	73.769	0.086	73.965	0.197	-0.001
10	42.956	1.423	49.807	37.846	1.316	24.093	6.448	0.103	73.548	0.097	73.769	0.223	-0.002
11	40.66	1.375	48.141	37.751	1.275	24.927	6.786	0.115	73.302	0.109	73.548	0.25	-0.004
12	38.365	1.333	46.654	37.666	1.239	25.721	7.113	0.127	73.028	0.121	73.302	0.278	-0.004
13	36.069	1.295	45.317	37.59	1.206	26.48	7.43	0.14	72.724	0.134	73.028	0.307	-0.003
14	33.774	1.26	44.106	37.52	1.176	27.207	7.738	0.153	72.394	0.146	72.724	0.336	-0.005
15	31.478	1.229	43.007	37.458	1.148	27.903	8.037	0.166	72.029	0.159	72.394	0.366	-0.001
16	29.182	1.2	42.002	37.4	1.123	28.57	8.327	0.179	71.634	0.172	72.029	0.396	-0.001
17	26.887	1.174	41.078	37.347	1.1	29.213	8.609	0.192	71.212	0.186	71.634	0.426	-0.005
18	24.591	1.149	40.228	37.299	1.079	29.83	8.884	0.206	70.758	0.199	71.212	0.457	-0.003
19	22.296	1.127	39.44	37.254	1.059	30.426	9.151	0.22	70.275	0.213	70.758	0.488	-0.006
20	20	1.106	38.712	37.212	1.04	30.998	9.41	0.233	69.756	0.226	70.275	0.52	-0.001

Table 4.19. Case 3 Water Surface Profile Along Chute Channel (Existing Discharge, Approach 1)



Index	ZCoord	Flow Depth	Area	Wetted Perimeter	Hydraulic Radius	Velocity	Froude Number	Friction Slope	Total Head	Averaged Friction Slope	Previous Total Head	Hydraulic Loss	Error Term
0	65.912	3.797	132.901	42.594	3.12	12.039	1.973	0.008	75.984	0	0	0	0
1	63.616	3.257	113.997	41.514	2.746	14.035	2.483	0.013	75.96	0.011	75.984	0.024	0
2	61.321	2.914	101.996	40.828	2.498	15.687	2.934	0.019	75.924	0.016	75.96	0.036	0
3	59.025	2.668	93.382	40.336	2.315	17.134	3.349	0.025	75.875	0.022	75.924	0.049	0
4	56.73	2.479	86.774	39.959	2.172	18.439	3.739	0.031	75.811	0.028	75.875	0.064	0
5	54.434	2.328	81.482	39.656	2.055	19.636	4.109	0.038	75.733	0.034	75.811	0.079	0
6	52.138	2.203	77.114	39.407	1.957	20.748	4.463	0.045	75.638	0.041	75.733	0.095	-0.001
7	49.843	2.098	73.425	39.196	1.873	21.791	4.803	0.053	75.528	0.049	75.638	0.112	-0.002
8	47.547	2.007	70.258	39.015	1.801	22.773	5.132	0.061	75.4	0.057	75.528	0.13	-0.002
9	45.252	1.928	67.5	38.857	1.737	23.704	5.45	0.069	75.253	0.065	75.4	0.149	-0.002
10	42.956	1.859	65.071	38.718	1.681	24.589	5.758	0.077	75.086	0.073	75.253	0.168	-0.001
11	40.66	1.797	62.911	38.595	1.63	25.433	6.057	0.086	74.899	0.082	75.086	0.188	-0.001
12	38.365	1.742	60.976	38.484	1.584	26.24	6.347	0.095	74.69	0.091	74.899	0.209	0
13	36.069	1.692	59.226	38.384	1.543	27.015	6.631	0.105	74.464	0.1	74.69	0.23	-0.003
14	33.774	1.647	57.637	38.294	1.505	27.76	6.907	0.114	74.215	0.11	74.464	0.252	-0.003
15	31.478	1.605	56.188	38.211	1.47	28.476	7.176	0.124	73.942	0.119	74.215	0.274	-0.001
16	29.182	1.567	54.858	38.135	1.439	29.166	7.438	0.134	73.648	0.129	73.942	0.296	-0.002
17	26.887	1.532	53.633	38.065	1.409	29.833	7.694	0.144	73.331	0.139	73.648	0.319	-0.002
18	24.591	1.5	52.499	38	1.382	30.477	7.945	0.155	72.994	0.149	73.331	0.343	-0.005
19	22.296	1.47	51.449	37.94	1.356	31.099	8.19	0.165	72.629	0.16	72.994	0.367	-0.002

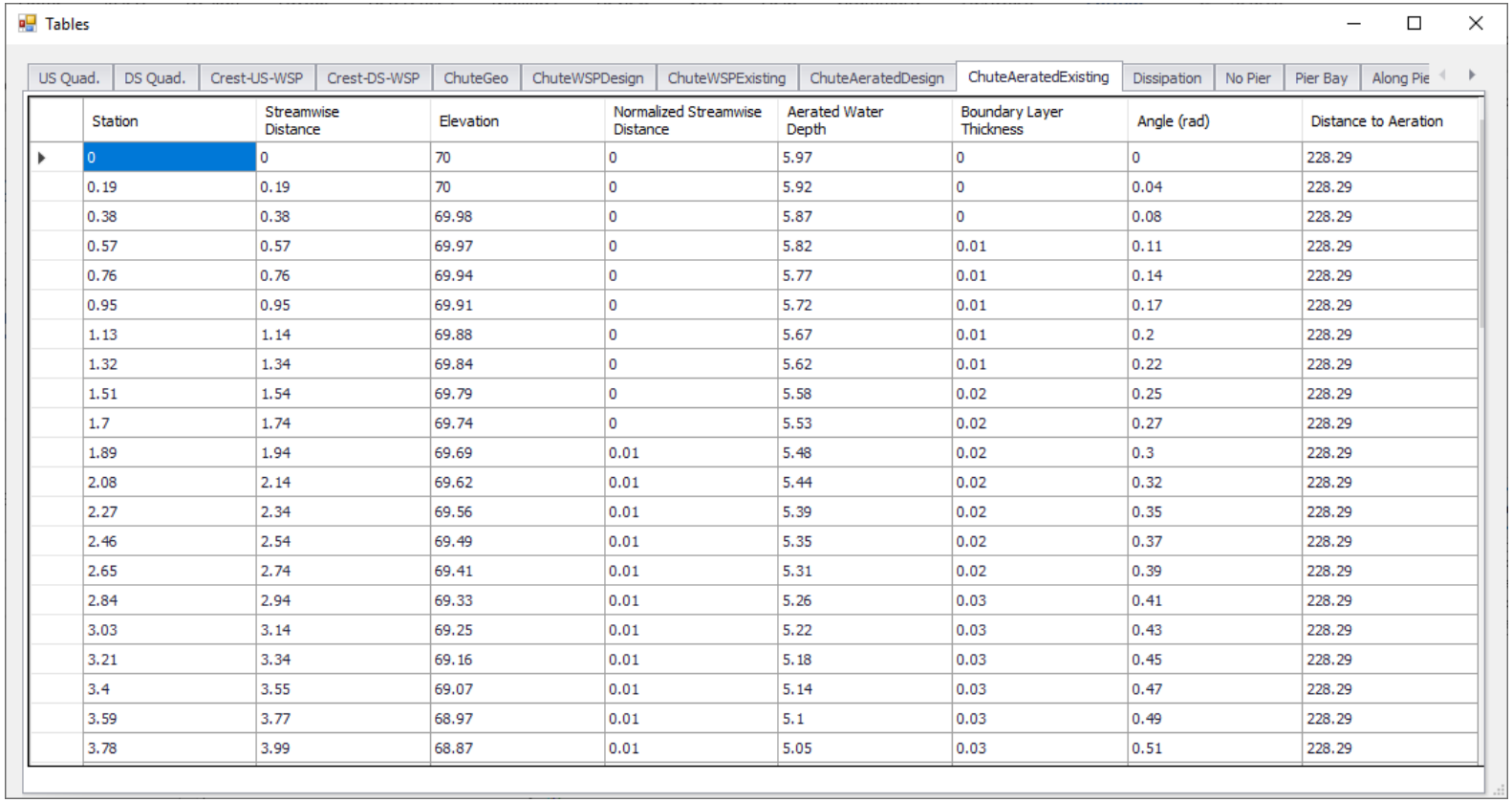
Table 4.20. Case 3 Water Surface Profile Along Chute Channel (Design Discharge, Approach 2)

Tables

US Quad. | DS Quad. | Crest-US-WSP | Crest-DS-WSP | ChuteGeo | ChuteWSPDesign | ChuteWSPExisting | ChuteAeratedDesign | ChuteAeratedExisting | Dissipation | No Pier | Pier Bay | Along Pie

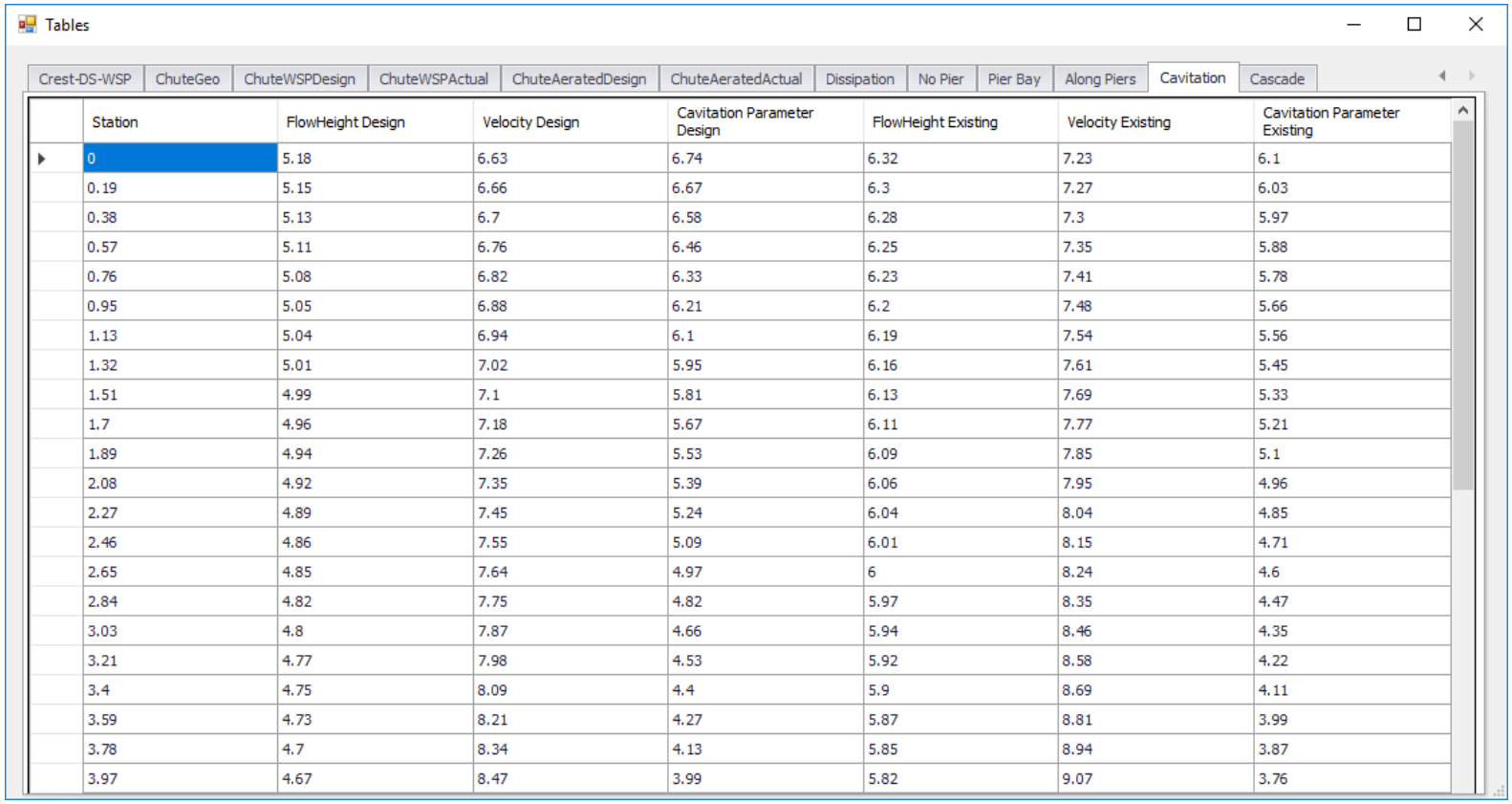
Station	Streamwise Distance	Elevation	Normalized Streamwise Distance	Aerated Water Depth	Boundary Layer Thickness	Angle (rad)	Distance to Aeration
0	0	70	0	4.93	0	0	185.59
0.19	0.19	70	0	4.88	0	0.04	185.59
0.38	0.38	69.98	0	4.83	0	0.08	185.59
0.57	0.57	69.97	0	4.77	0.01	0.11	185.59
0.76	0.76	69.94	0	4.73	0.01	0.14	185.59
0.95	0.95	69.91	0	4.68	0.01	0.17	185.59
1.13	1.14	69.88	0	4.63	0.01	0.2	185.59
1.32	1.34	69.84	0	4.58	0.01	0.22	185.59
1.51	1.54	69.79	0.01	4.53	0.02	0.25	185.59
1.7	1.74	69.74	0.01	4.49	0.02	0.27	185.59
1.89	1.94	69.69	0.01	4.44	0.02	0.3	185.59
2.08	2.14	69.62	0.01	4.4	0.02	0.32	185.59
2.27	2.34	69.56	0.01	4.36	0.02	0.35	185.59
2.46	2.54	69.49	0.01	4.31	0.02	0.37	185.59
2.65	2.74	69.41	0.01	4.27	0.02	0.39	185.59
2.84	2.94	69.33	0.01	4.23	0.03	0.41	185.59
3.03	3.14	69.25	0.01	4.19	0.03	0.43	185.59
3.21	3.34	69.16	0.01	4.15	0.03	0.45	185.59
3.4	3.55	69.07	0.01	4.11	0.03	0.47	185.59
3.59	3.77	68.97	0.01	4.07	0.03	0.49	185.59
3.78	3.99	68.87	0.01	4.03	0.03	0.51	185.59

Table 4.21. Case 3 Water Surface Profile Along Chute Channel (Existing Discharge, Approach 2)



Station	Streamwise Distance	Elevation	Normalized Streamwise Distance	Aerated Water Depth	Boundary Layer Thickness	Angle (rad)	Distance to Aeration
0	0	70	0	5.97	0	0	228.29
0.19	0.19	70	0	5.92	0	0.04	228.29
0.38	0.38	69.98	0	5.87	0	0.08	228.29
0.57	0.57	69.97	0	5.82	0.01	0.11	228.29
0.76	0.76	69.94	0	5.77	0.01	0.14	228.29
0.95	0.95	69.91	0	5.72	0.01	0.17	228.29
1.13	1.14	69.88	0	5.67	0.01	0.2	228.29
1.32	1.34	69.84	0	5.62	0.01	0.22	228.29
1.51	1.54	69.79	0	5.58	0.02	0.25	228.29
1.7	1.74	69.74	0	5.53	0.02	0.27	228.29
1.89	1.94	69.69	0.01	5.48	0.02	0.3	228.29
2.08	2.14	69.62	0.01	5.44	0.02	0.32	228.29
2.27	2.34	69.56	0.01	5.39	0.02	0.35	228.29
2.46	2.54	69.49	0.01	5.35	0.02	0.37	228.29
2.65	2.74	69.41	0.01	5.31	0.02	0.39	228.29
2.84	2.94	69.33	0.01	5.26	0.03	0.41	228.29
3.03	3.14	69.25	0.01	5.22	0.03	0.43	228.29
3.21	3.34	69.16	0.01	5.18	0.03	0.45	228.29
3.4	3.55	69.07	0.01	5.14	0.03	0.47	228.29
3.59	3.77	68.97	0.01	5.1	0.03	0.49	228.29
3.78	3.99	68.87	0.01	5.05	0.03	0.51	228.29

Table 4.22. Case 3 Cavitation Parameters



Station	FlowHeight Design	Velocity Design	Cavitation Parameter Design	FlowHeight Existing	Velocity Existing	Cavitation Parameter Existing
0	5.18	6.63	6.74	6.32	7.23	6.1
0.19	5.15	6.66	6.67	6.3	7.27	6.03
0.38	5.13	6.7	6.58	6.28	7.3	5.97
0.57	5.11	6.76	6.46	6.25	7.35	5.88
0.76	5.08	6.82	6.33	6.23	7.41	5.78
0.95	5.05	6.88	6.21	6.2	7.48	5.66
1.13	5.04	6.94	6.1	6.19	7.54	5.56
1.32	5.01	7.02	5.95	6.16	7.61	5.45
1.51	4.99	7.1	5.81	6.13	7.69	5.33
1.7	4.96	7.18	5.67	6.11	7.77	5.21
1.89	4.94	7.26	5.53	6.09	7.85	5.1
2.08	4.92	7.35	5.39	6.06	7.95	4.96
2.27	4.89	7.45	5.24	6.04	8.04	4.85
2.46	4.86	7.55	5.09	6.01	8.15	4.71
2.65	4.85	7.64	4.97	6	8.24	4.6
2.84	4.82	7.75	4.82	5.97	8.35	4.47
3.03	4.8	7.87	4.66	5.94	8.46	4.35
3.21	4.77	7.98	4.53	5.92	8.58	4.22
3.4	4.75	8.09	4.4	5.9	8.69	4.11
3.59	4.73	8.21	4.27	5.87	8.81	3.99
3.78	4.7	8.34	4.13	5.85	8.94	3.87
3.97	4.67	8.47	3.99	5.82	9.07	3.76

Table 4.22. Case 3 Cavitation Parameters, Continued

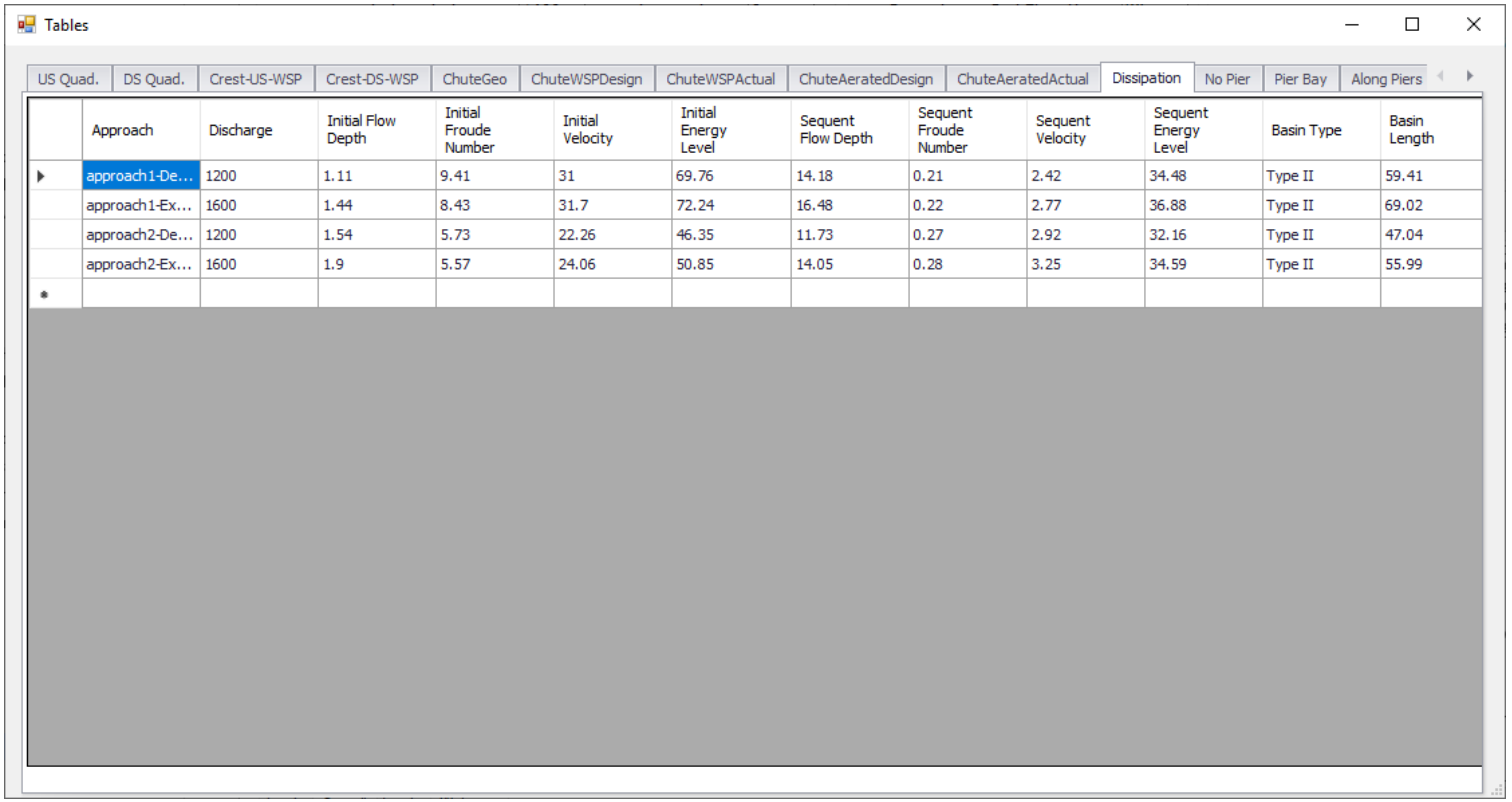
Station	FlowHeight Design	Velocity Design	Cavitation Parameter Design	FlowHeight Existing	Velocity Existing	Cavitation Parameter Existing
0	5.18	6.63	6.74	6.32	7.23	6.1
0.19	5.15	6.66	6.67	6.3	7.27	6.03
0.38	5.13	6.7	6.58	6.28	7.3	5.97
0.57	5.11	6.76	6.46	6.25	7.35	5.88
0.76	5.08	6.82	6.33	6.23	7.41	5.78
0.95	5.05	6.88	6.21	6.2	7.48	5.66
1.13	5.04	6.94	6.1	6.19	7.54	5.56
1.32	5.01	7.02	5.95	6.16	7.61	5.45
1.51	4.99	7.1	5.81	6.13	7.69	5.33
1.7	4.96	7.18	5.67	6.11	7.77	5.21
1.89	4.94	7.26	5.53	6.09	7.85	5.1
2.08	4.92	7.35	5.39	6.06	7.95	4.96
2.27	4.89	7.45	5.24	6.04	8.04	4.85
2.46	4.86	7.55	5.09	6.01	8.15	4.71
2.65	4.85	7.64	4.97	6	8.24	4.6
2.84	4.82	7.75	4.82	5.97	8.35	4.47
3.03	4.8	7.87	4.66	5.94	8.46	4.35
3.21	4.77	7.98	4.53	5.92	8.58	4.22
3.4	4.75	8.09	4.4	5.9	8.69	4.11
3.59	4.73	8.21	4.27	5.87	8.81	3.99
3.78	4.7	8.34	4.13	5.85	8.94	3.87
3.97	4.67	8.47	3.99	5.82	9.07	3.76

Froude number at the end of the chute is 9.41 for the design discharge and 8.43 for the existing discharge in approach 1. In approach 2, Froude number at the end of the chute is 5.73 for the design discharge and 5.57 for the existing discharge. The sequent depth and energy level at the downstream of the hydraulic jump will be different for each approach and discharge. USBR Type II stilling basin is proper for both discharges and approaches.

As shown in Table 4.23, energy level is calculated as 36.88 m in approach 1, and 34.59 m in approach 2 for existing discharge. Tailwater energy level is 30 m for this case. Therefore, the dissipated energy level is still higher than the tailwater level. Adjust stilling basin elevation feature should be used to determine the right stilling basin elevation.

As shown in Figure 4.9, stilling basin initial elevation was equal to downstream bed elevation which is 20 m. In order to keep downstream energy level lower than the tailwater level, stilling basin elevation is decreased to 12.62 m (See Figure 4.10).

Table 4.23. Hydraulic Jump Table for Case 3



Approach	Discharge	Initial Flow Depth	Initial Froude Number	Initial Velocity	Initial Energy Level	Sequent Flow Depth	Sequent Froude Number	Sequent Velocity	Sequent Energy Level	Basin Type	Basin Length
approach1-De...	1200	1.11	9.41	31	69.76	14.18	0.21	2.42	34.48	Type II	59.41
approach1-Ex...	1600	1.44	8.43	31.7	72.24	16.48	0.22	2.77	36.88	Type II	69.02
approach2-De...	1200	1.54	5.73	22.26	46.35	11.73	0.27	2.92	32.16	Type II	47.04
approach2-Ex...	1600	1.9	5.57	24.06	50.85	14.05	0.28	3.25	34.59	Type II	55.99

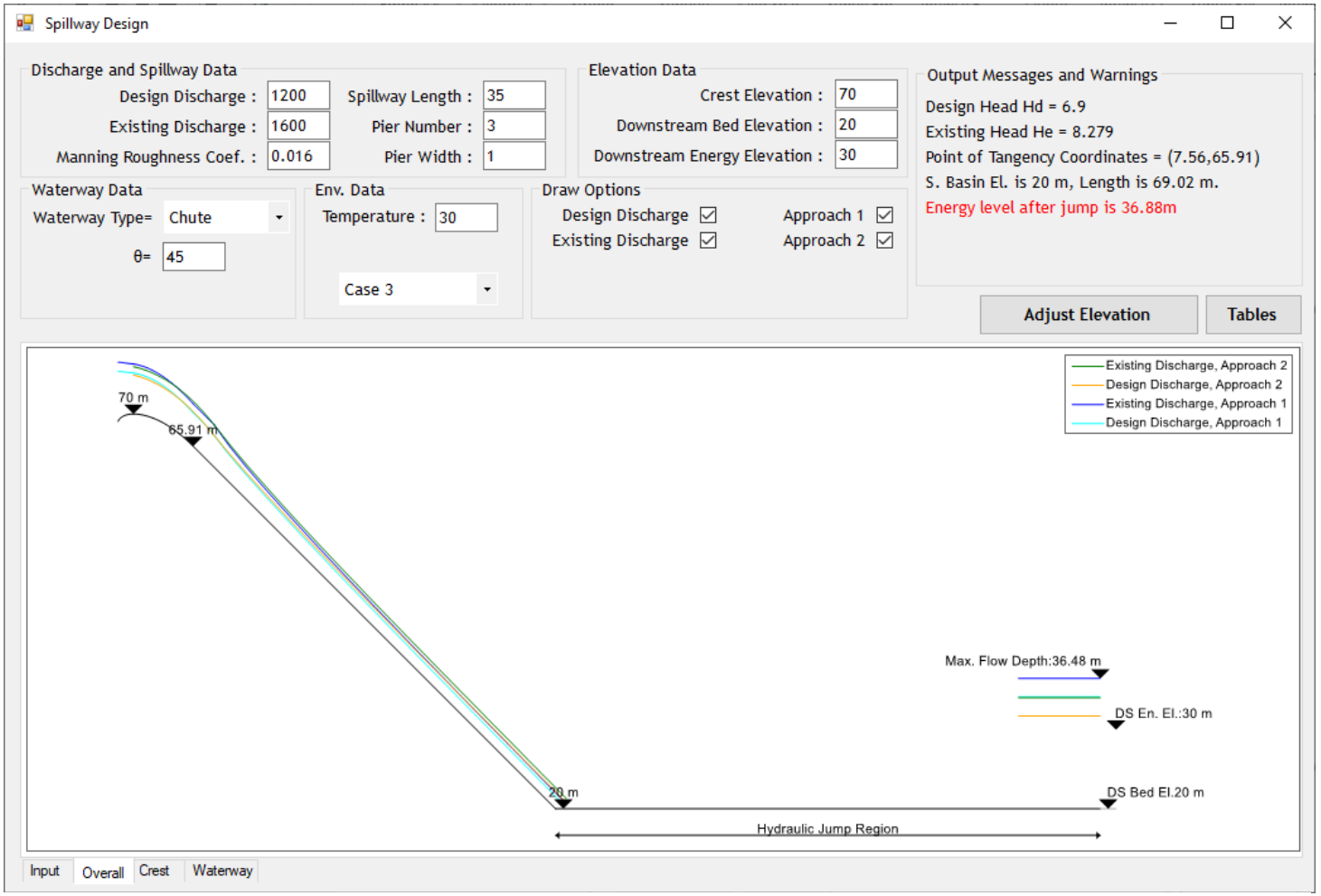


Figure 4.9. Case 3 – Initial Design

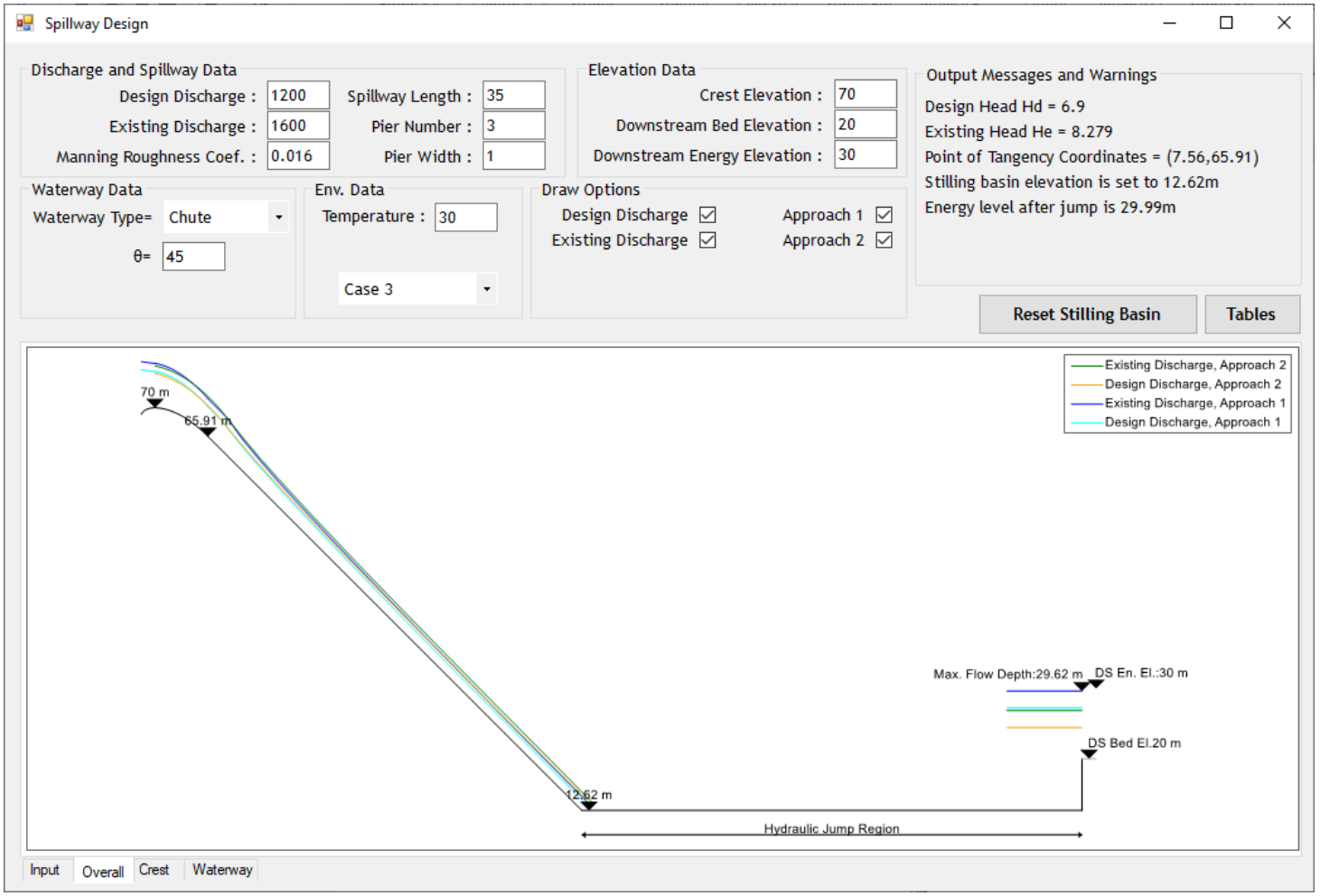


Figure 4.10. Case 3 – Adjusted Design

4.6. Results of Case 4

Case 4 has only one difference than Case 3 which is the type of waterway. Stepped spillway is used in Case 4. Water surface profile over the crest is same as it is in Case 3. Vertical flow depth at the beginning of the chute channel is 4.23 m for design discharge and 5.37 m for existing discharge. Flow depth perpendicular to channel bottom is calculated 2.99 m for design discharge and 3.80 m for existing discharge.

In order to specify the flow type over the stepped channel, characteristic critical flow depth, $(d_c)_{onset}$, is computed as 0.59 m. Critical flow depth, (d_c), is 4.93 m and 5.97 m for design and existing discharge, respectively. Inception point of aeration is found 59.80 m and 73.36 m away from crest for design and existing discharge, respectively. From crest to toe of the spillway, available crest length is 73.75 m. Therefore, flow will be fully aerated before it reaches to the toe of the spillway.

As shown in Table 4.25, Froude number at the end of the chute is 2.69 for design discharge, 2.40 for existing discharge. Sequent depth and energy level at the downstream of the hydraulic jump will be different for each approach and discharge. USBR Type II stilling basin is proper for both discharges.

As shown in Figure 4.12, stilling basin initial elevation was equal to downstream bed elevation which is 20 m. In order to keep downstream energy level lower than the tailwater level, stilling basin elevation is decreased to 16.2 m (See Figure 4.13).

4.7. Results of Case 5

Case 5 has only one difference than Case 3 which is the existing discharge. In this case, existing discharge is smaller than design discharge. To this end, positive pressure on the crest is expected as seen in Figure 4.14

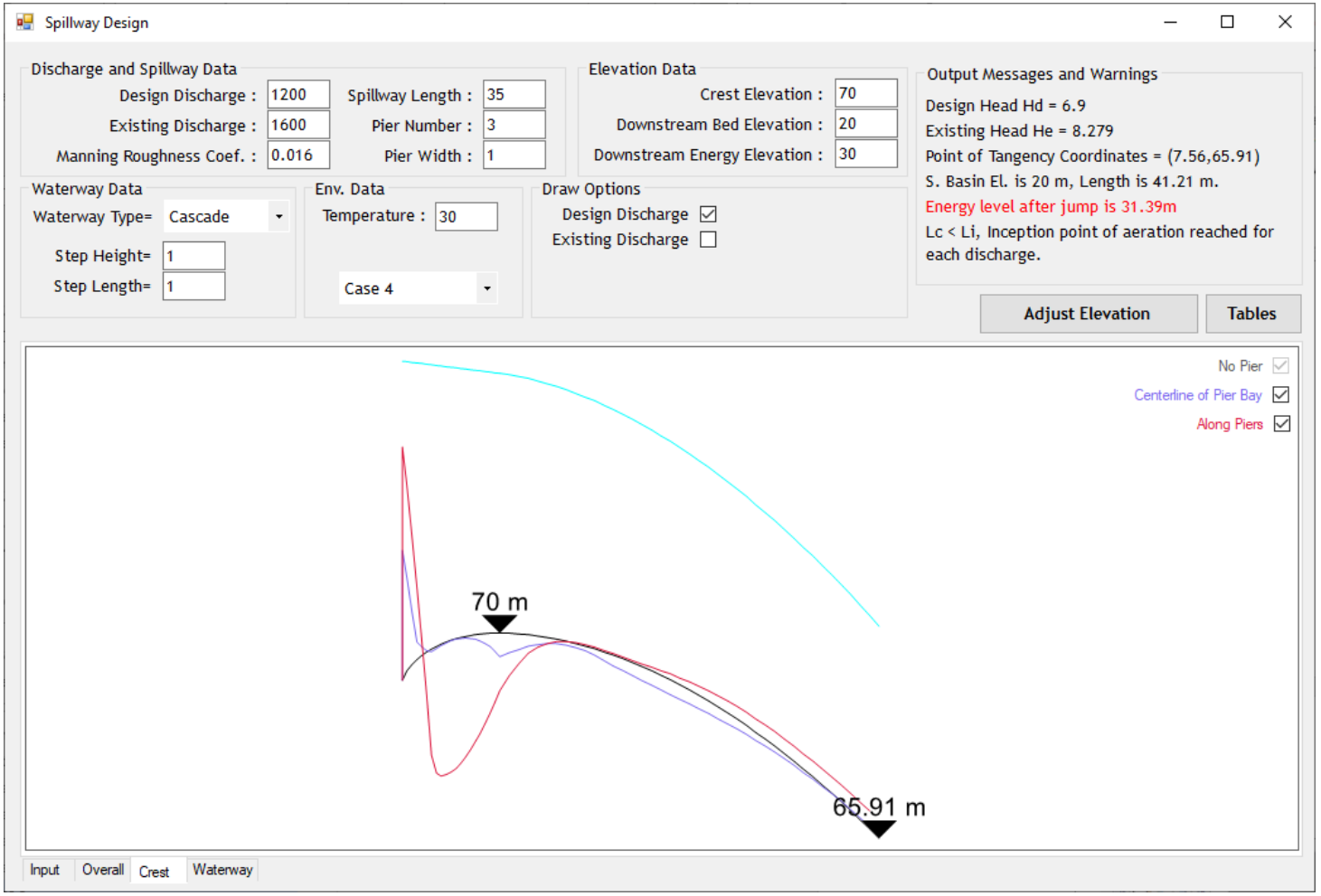
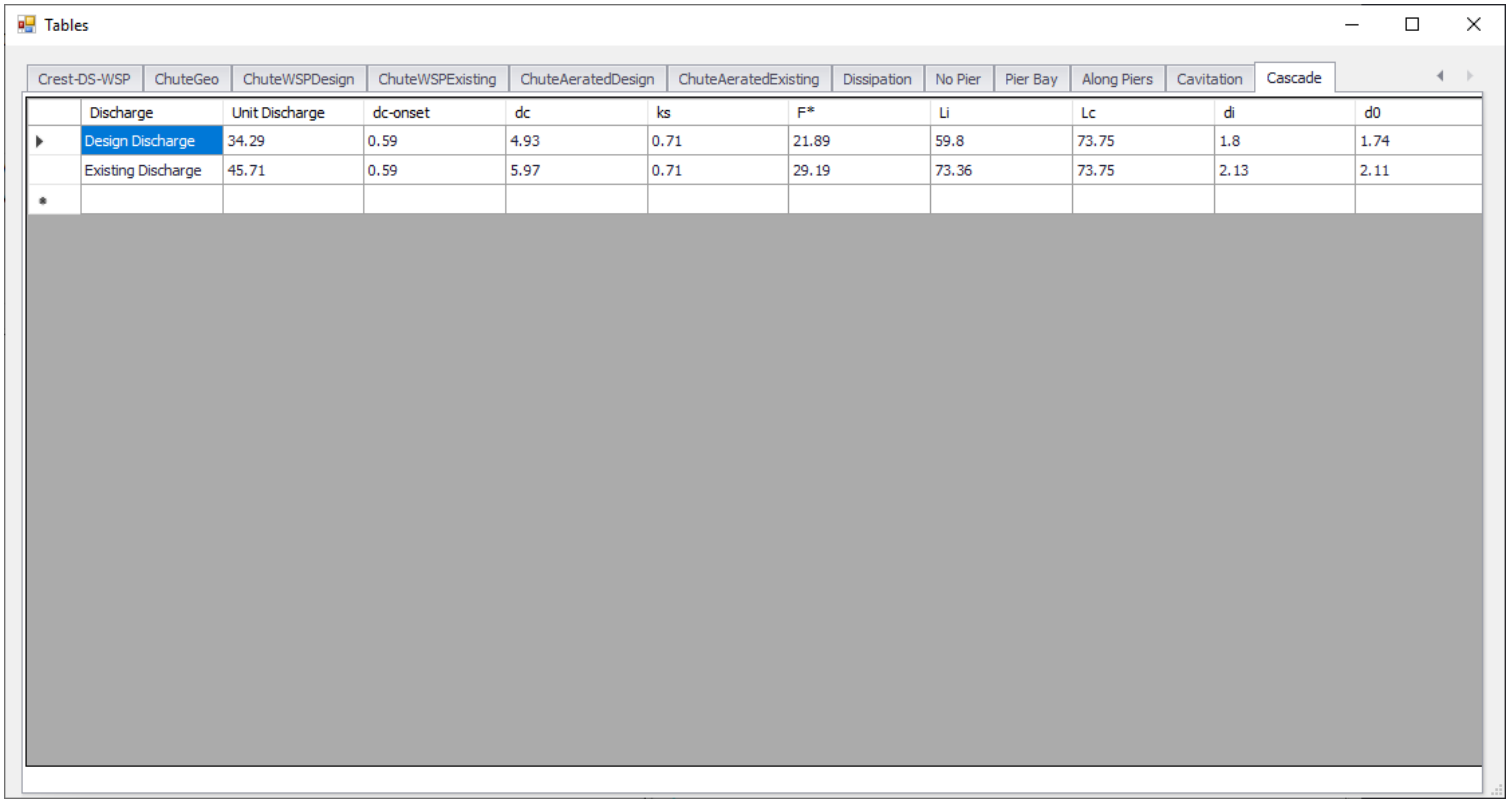


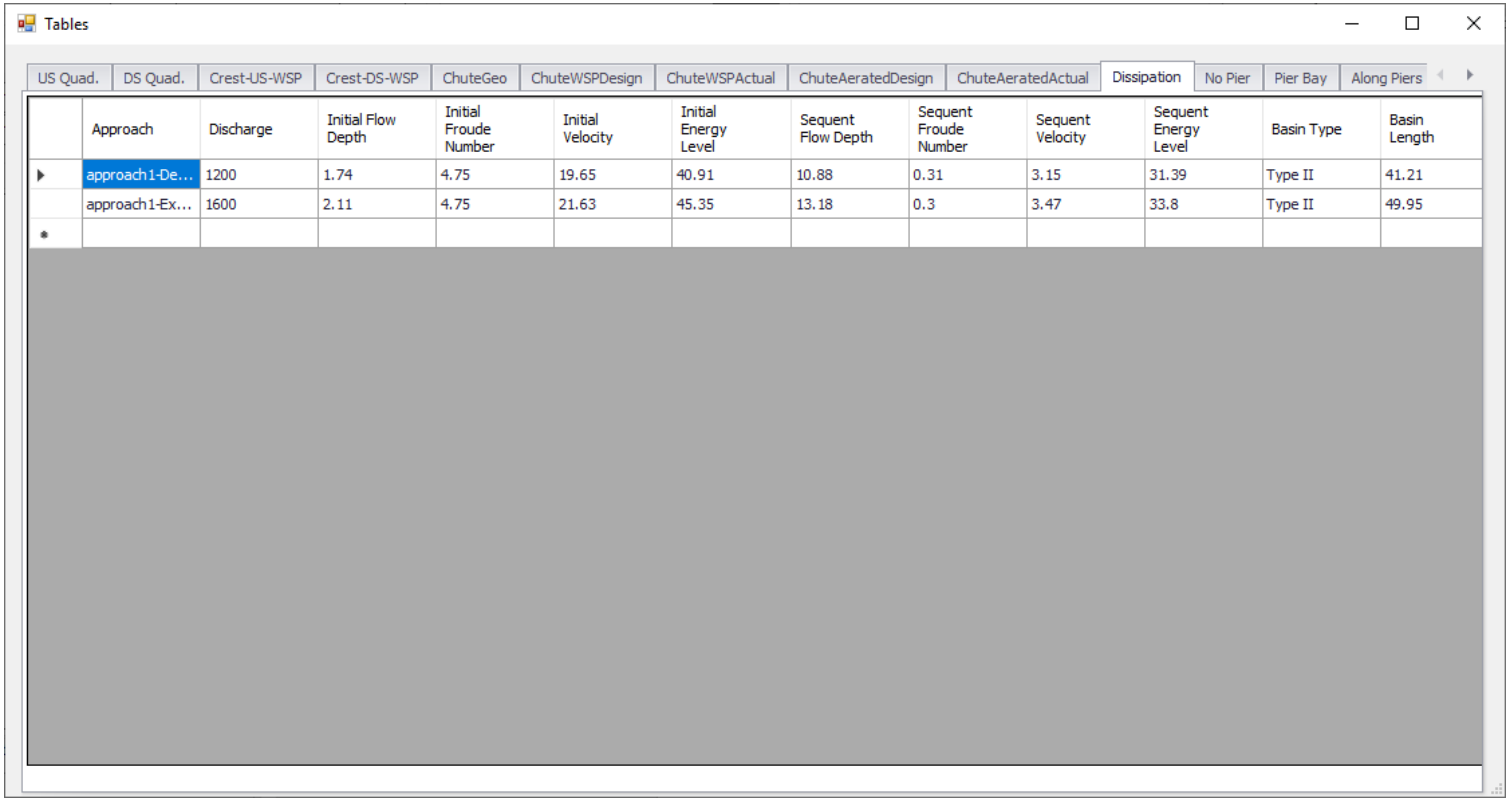
Figure 4.11. Case 4 – Crest Pressures

Table 4.24. Case 4 Stepped Spillway Calculations



	Discharge	Unit Discharge	dc-onset	dc	ks	F*	Li	Lc	di	d0
▶	Design Discharge	34.29	0.59	4.93	0.71	21.89	59.8	73.75	1.8	1.74
	Existing Discharge	45.71	0.59	5.97	0.71	29.19	73.36	73.75	2.13	2.11
*										

Table 4.25. Hydraulic Jump Table for Case 4



Approach	Discharge	Initial Flow Depth	Initial Froude Number	Initial Velocity	Initial Energy Level	Sequent Flow Depth	Sequent Froude Number	Sequent Velocity	Sequent Energy Level	Basin Type	Basin Length
approach1-De...	1200	1.74	4.75	19.65	40.91	10.88	0.31	3.15	31.39	Type II	41.21
approach1-Ex...	1600	2.11	4.75	21.63	45.35	13.18	0.3	3.47	33.8	Type II	49.95

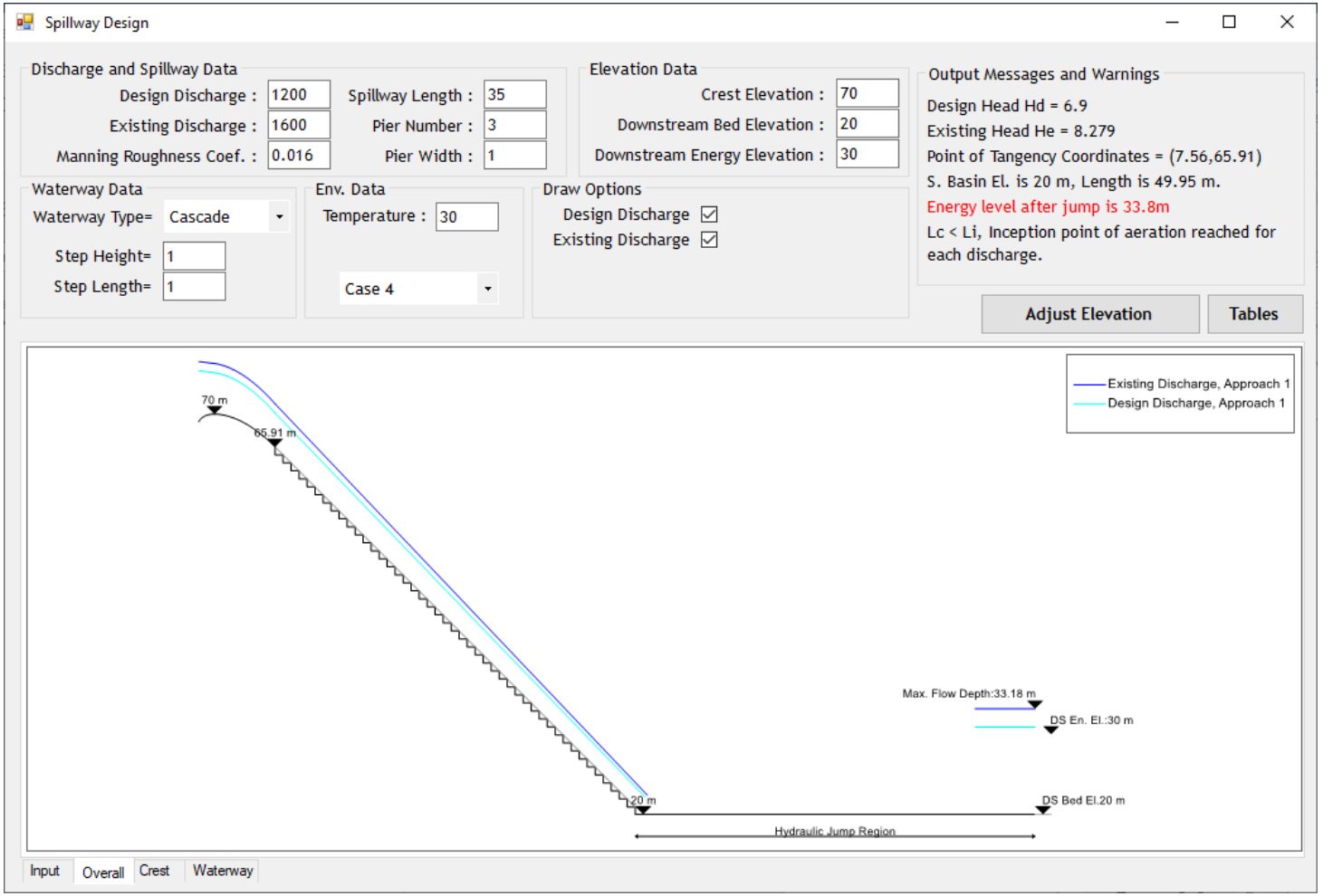


Figure 4.12. Case 4 – Initial Design

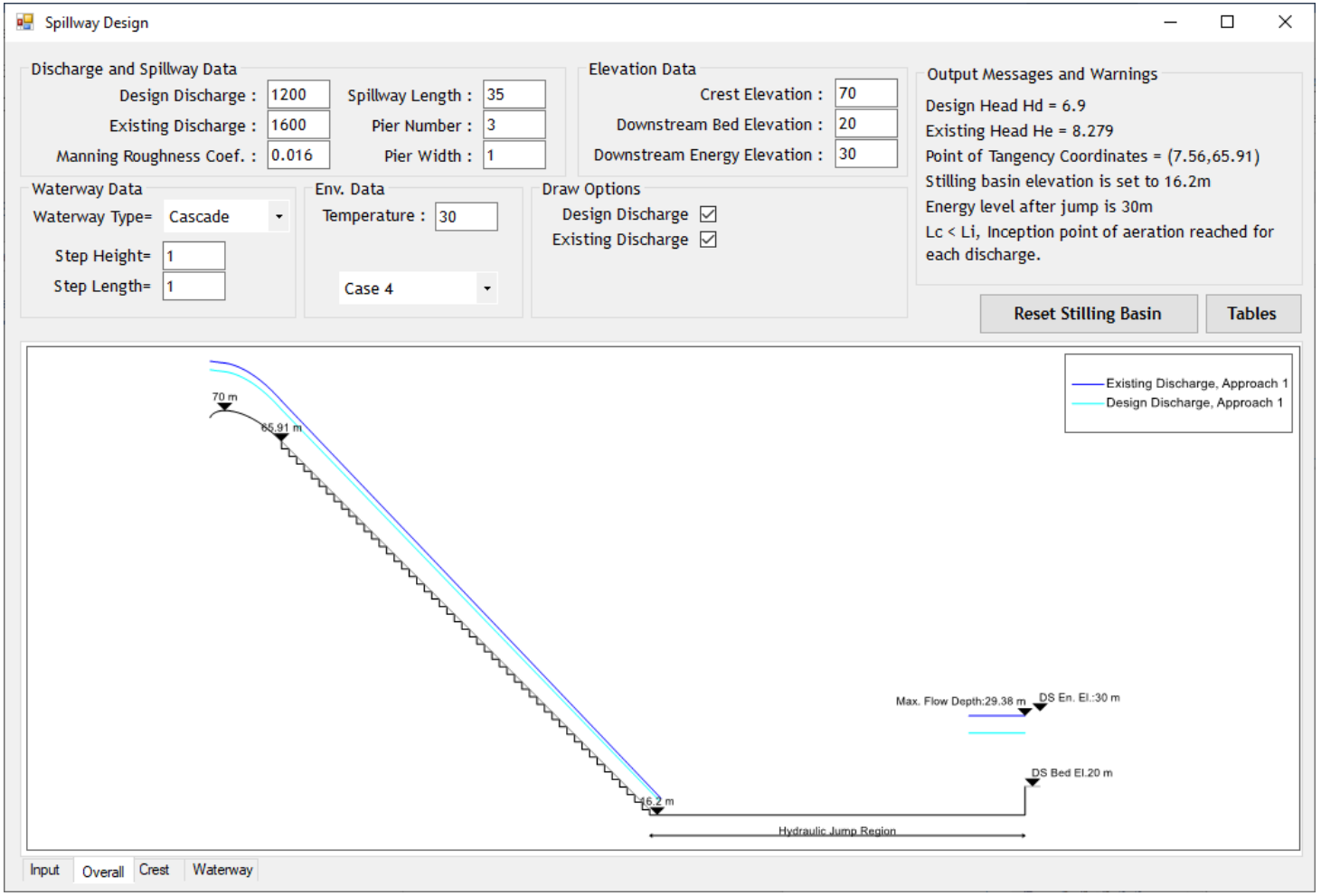


Figure 4.13. Case 4 – Adjusted Design

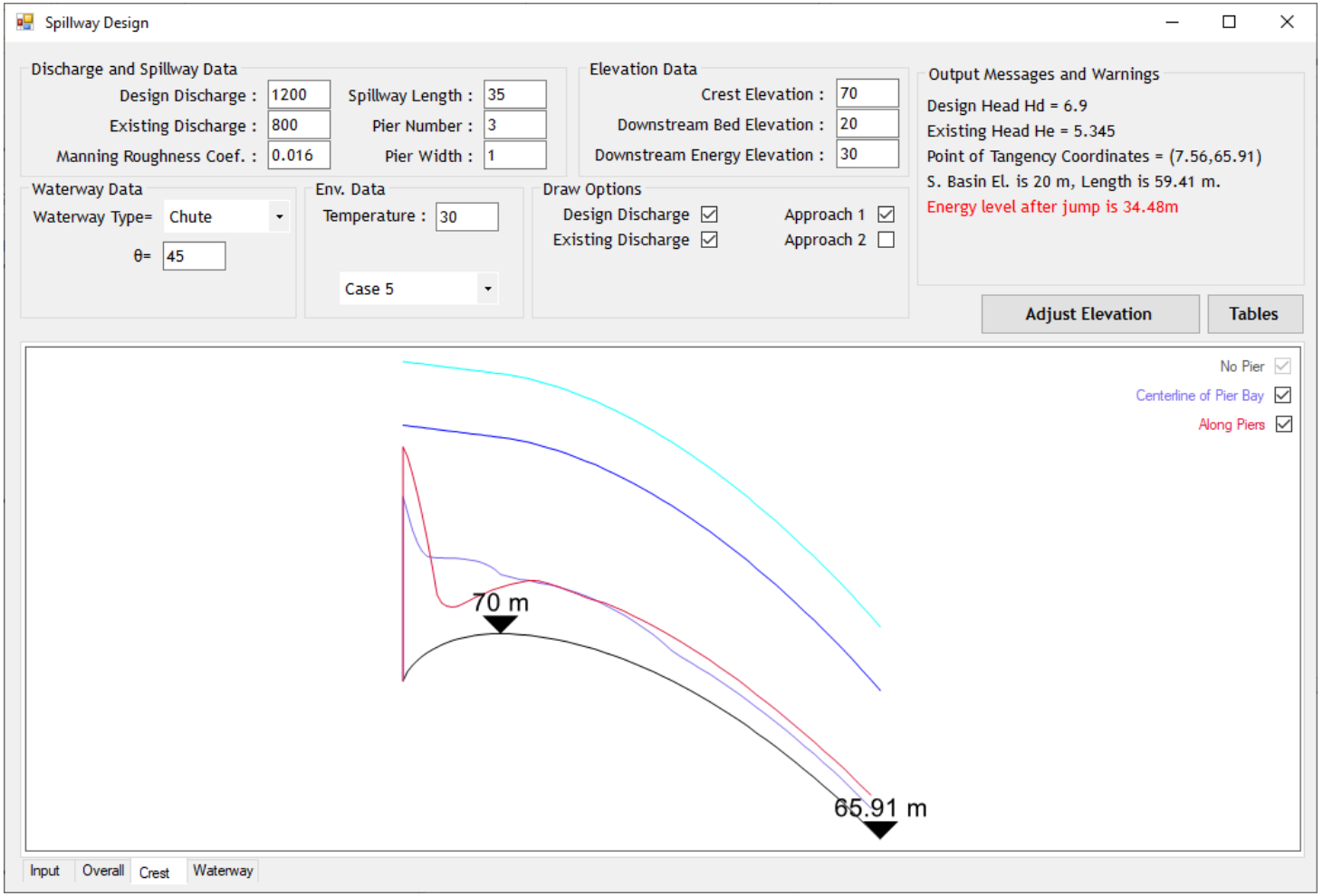


Figure 4.14. Case 5 – Crest Pressures

4.8. Discussion of the Application

The results of the analyses are summarized for each case in Table 4.26 which contains the information of discharge, Q_d, downstream channel type, Froude number at the spillway toe, F_{r1}, total heads on the spillway crest, H_{crest}, the spillway toe, H_{toe} , and the sequent depth location, $H_{sequent}$, percent energy loss between the crest and the toe, $E_{\%C-1}$, percent energy loss in the stilling basin, $E_{\%1-2}$

Since the flow is disturbed more on stepped spillways when compared to conventional chutes, the headloss is greater (See Table 4.26). $E_{\%1-2}$ is the headloss between the toe and the sequent depth location, which is also called the percent energy loss through the hydraulic jump. As initial Froude number increases, turbulence in flow becomes greater, hence stronger hydraulic jump is observed (See Table 4.26). In Table 4.26, L_B stands for the stilling basin length. As can be seen from Table 4.26, stepped spillways require relatively shorter stilling basins.

Table 4.26. Output Data

Data	Case 1	Case 2	Case 3	Case 4	Case 5
Q_d (m^3/s)	250	250	1200	1200	1200
Type	Chute	Stepped Spillway	Chute	Stepped Spillway	Chute
F_{r1} (approach 1)	3.84	2.69	9.41	4.75	9.41
F_{r1} (approach 2)	3.29	NA	5.73	NA	5.73
H_{crest} (m)	18.44	18.44	77.85	77.85	77.85
H_{toe} (m)	17.15	14.84	69.76	41.53	69.76
$H_{sequent}$ (m)	14.65	14.10	34.48	31.39	34.48
$E_{\%C-1}$ (%)	7%	20%	10%	47%	10%
$E_{\%1-2}$ (%)	15%	5%	51%	24%	51%
Basin Type (approach 1)	Type IV	Type IV	Type II	Type II	Type II
L_B (approach 1) (m)	25.15	18.67	59.41	41.21	59.41
Basin Type (approach 2)	Type IV	NA	Type II	NA	Type II
L_B (approach 2) (m)	22.31	NA	47.04	NA	47.04

* NA: Not applicable.

CHAPTER 5

CONCLUSION

A spillway is a complicated structure and its design is a time-consuming process. The design and flow analysis of a spillway have been conducted within the scope of this study. At the beginning of the study, a literature survey is performed. Recent assumptions, approaches, and various calculation methods used in the design and analysis for each component of the spillway are explored. Safety of the spillway in terms of released total energy to the downstream area, cavitation possibility, and bottom pressure distributions are concerned in this study. A visual, user-friendly software named "SP-DAS" is developed to make the whole process faster and easier. This software is able to respond to user's requests dynamically. With the execution of the software, a designer can evaluate the results in terms of hydraulic conformity and cost in an implicit manner. It is possible to observe the influence of any input data by just changing it. All output data are updated when any input data change. In order to demonstrate the software, an application study with five alternatives is performed. As a summary of that application, despite the challenge in their construction, stepped spillways are found to be more efficient in terms of energy dissipation, because of considerable air entrainment due to increased turbulence intensity over steps.

In a future study, effect of spillway gates and their possible operations can be incorporated into the software. Furthermore, cavitation resistant design process can be incorporated into the software. Cost estimation of conventional and stepped spillways can also be added to the software.

REFERENCES

ANSYS. (2009). "ANSYS Fluent 12.0 Getting Started Guide." *Knowledge Creation Diffusion Utilization*, 15317(April), 724–746.

Biswa, K., Jamatia, B., Choudhury, D., and Borah, P. (2016). "Comparative analysis of C, Fortran, C# and Java Programming Languages." *International Journal of Computer Science and Information Technologies (IJCSIT)*, 7(2), 1004–1007.

Boes, Robert M. and Hans-erwin Minor. 2000. "Guidelines for the Hydraulic Design of Stepped Spillways." *Proc. Int. Workshop on Hydraulics of Stepped Spillways* (September 2015):163–70.

Chamani, M. R., and Rajaratnam, N. (1994). "Jet flow on stepped spillways." J. Hydraul. Eng., 10.1061/(ASCE)0733-9429(1994)120:2(254), 254–259.

Chanson, H. (1994). "Hydraulics of skimming flows over stepped channels and spillways." *Journal of Hydraulic Research*, Taylor and Francis, 32(3), 445–460.

Chanson, H. (1996a). *Air Bubble Entrainment in Free-Surface Turbulent Shear Flows*. Elsevier Science.

Chanson, H. (1996b). "Prediction of the transition nappe/skimming flow on a stepped channel." *Journal of Hydraulic Research*, Taylor and Francis, 34(3), 421–429.

Chanson, H. (2008). "History of stepped channels and spillways: a rediscovery of the 'wheel.'" *Canadian Journal of Civil Engineering*, 22(2), 247–259.

Chatila, J., and Tabbara, M. (2004). *Computational modeling of flow over an ogee spillway*, *Computers and structures, 82, 1805-1812*. *Computers and Structures*.

Chow, V. (1959). *Open-Channel Hydraulics*. *New York: McGraw-Hill Book Company, Inc.*, McGraw-Hill, New York.

Creager, W. P. (1917). *Engineering for Masonry Dams*. J. Wiley & sons, Incorporated.

Falvey, H. T. (1990). Cavitation in chutes and spillways. Engineering Monograph 42. *U.S. Department of the Interior, U.S. Government Printing Office, Denver, Colo.*

Flow Science, I. (2019). "FLOW-3D, Version 12.0 ." Santa Fe, NM.

Gonzalez, C. A., and Chanson, H. (2007). "Hydraulic Design of Stepped Spillways and Downstream Energy Dissipators for Embankment Dams." *Dam Engineering*, 17(4), 223–244.

Graham, J. R., Creegan, P. J., Hamilton, W. S., Kaden, R. A., McDonald, J. E., Noble, G. E., and Schrader, E. K. (1998). “Reported by ACI Committee 211.” *Most*, 93(Reapproved), 1–18.

Hager. (1991). “Experiments on Standard Spillway Flow (Including Appendix).” *Proceedings of the Institution of Civil Engineers*, 91(3), 399–416.

Hager. (1987). “Continuous Crest Profile for Standard Spillway.” *Journal of Hydraulic Engineering*, American Society of Civil Engineers, 113(11), 1453–1457.

Hager, W.H., and Blaser, F. 1998. “Drawdown curve and incipient aeration for chute flow. ” *Canadian Journal of Civil Engineering, 25: 467–473.*

Hashimi, S. Al. (2018). "CFD Modeling of Flow over Ogee Spillway by Using Different Turbulence Models.” *International Journal of Scientific and Technology Research (November 2013).*

Jagtap, M. M., and Shrivastava, B. B. (2014). “Source Equation Analysis Of Cavitation In Fluid Flow Over A Surface.” 3(3), 55–57.

Kiamanesh, H. (1996). “An Investigation to Predict the Sub-Atmospheric Pressure on High Spillways.” *PhD Thesis, Concordia University.*

Kim, D. G., and Park, J. H. (2005). “Analysis of flow structure over ogee-spillway in consideration of scale and roughness effects by using CFD model.” *J. Civil Eng. KSCE*, 9(2), 161–169.

May, R. W. P. (1987). “Cavitation in hydraulic structures: occurrence and prevention.” *Technical Report. Hydraulics Research Wallingford.*

Melsheimer, E. S., Murphy, T. E., United States., & Waterways Experiment Station (U.S.). (1970). *Investigations of various shapes of the upstream quadrant of the crest of a high spillway: Hydraulic laboratory investigation*. Vicksburg, Miss: Waterways Experiment Station

Molland, A. F. (2011). “The Maritime Engineering Reference Book : a Guide to Ship Design, Construction and Operation.” Butterworth-Heinemann.

Renna, F. (2005). *Air water flow features in skimming regime in stepped spillways*.

Savage, B. M., and Johnson M. C. (2001). “Flow over Ogee Spillway: Physical and Numerical Model Case Study.” *Journal of Hydraulic Engineering*, American Society of Civil Engineers, 127(8), 640–649.

Sinniger, R. O., and Hager, W. H. (1989). *Constructions hydrauliques: ecoulements stationnaires*. Traite de Genie Civil, Presses Polytechniques Romandes.

USACE. (1992). “Hydraulic Design of Spillways, EM 1110-2-1603”, *U.S. Army Corps of Engineers.*

USBR. (1987). *Design of small dams.* U.S. Dept. of the Interior, Bureau of Reclamation, Denver.

USBR. (2015). *"Cavitation Damage Induced Failure of Spillways Key Concepts Description of Potential Failure Mode."* Bureau of Reclamation, Denver.

Vischer, D., and Hager, W. H. (1998). *Dam hydraulics*. Wiley series in water resources engineering, Wiley.

Yanmaz, A. M. (2018). *Applied water resources engineering*. METU Press, Ankara.

APPENDICES

A. SOURCE CODE OF THE SOFTWARE

A.1. Form Fields

```
private double _DesignDischarge;
private double _ExistingDischarge;
private double _Manning;
private double _SpillwayRoughLength;
private int _PierNumber;
private double _PierWidth;
private double _CrestElevation;
private double _DownstreamBedElevation;
private double _DownstreamEnergyElevation;
private double _StillingBasinElevation;
private double _Temperature;
private double _StepHeight;
private double _StepLength;
private double _ChuteAngleDegree;
private double _ChuteAngleRadian;
private double _DesignHead;
private double _ExistingHead;
Crest _Crest = new Crest();
HydraulicSolver _HydraulicSolver = new HydraulicSolver();
private double _CrestGeometryUpstreamNodes;
private double _CrestGeometryDownstreamNodes;
private double _CrestGeometryUpstreamSegment;
private double _CrestGeometryDownstreamSegment;
private double _CrestWaterSurfaceProfileNodes;
```

```
private double _CrestWaterSurfaceProfileSegment;
DataTable _dtCrestGeometryUpstream;
DataTable _dtCrestGeometryDownstream;
DataTable _dtCrestUSWSP;
DataTable _dtCrestDSWSP;
DataTable _dtChuteGeometry;
DataTable _dtChuteWSPDesign;
DataTable _dtChuteWSPExisting;
DataTable _dtAeratedChuteDesign;
DataTable _dtAeratedChuteExisting;
DataTable _dtPreDissipationTable;
DataTable _dtPressureNoPier;
DataTable _dtPressurePierBay;
DataTable _dtPressureAlongPiers;
DataTable _dtCavitation;
DataTable _dtCascade;
WSPTable _ChuteWSPDesign;
WSPTable _ChuteWSPExisting;
Polyline _plUpstreamBottomProfile;
Polyline _plDownstreamBottomProfile;
Polyline _plWSPDesignUpstream;
Polyline _plWSPDesignDownstream;
Polyline _plWSPExistingUpstream;
Polyline _plWSPExistingDownstream;
Polyline _plWaterwayBottomProfile;
Polyline _plWSPDesignStandardStep;
Polyline _plWSPExistingStandardStep;
Polyline _plDesignWSP;
Polyline _plExistingWSP;
Polyline _plCascadeDesignWSP;
Polyline _plCascadeExistingWSP;
```

```
Polyline _plStillingBasinGeometry;
Polyline _plDownstreamBed;
Polyline _plDissipationStep;
Polyline _linkDesign = new Polyline();
Polyline _linkExisting = new Polyline();
Polyline _plcrestpressuresNoPier;
Polyline _plcrestpressurePierBay;
Polyline _plcrestpressureAlongPiers;
Polyline _plPseudoBottom;
double _PTStreamWiseDistance;
double _InitialFlowDepthDesign;
double _InitialFlowDepthExisting;
double _WaterwayHeight;
double _WaterwayHorizontalLength;
double _ChuteCalculationStepNumber;
double _ChuteHorizontalCalculationInterval;
double _ChuteVerticalCalculationInterval;
double _FlowDepthBeforeJumpApproach1Design;
double _FlowDepthBeforeJumpApproach1Existing;
double _FlowDepthBeforeJumpApproach2Design;
double _FlowDepthBeforeJumpApproach2Existing;
double _MaximumStillingBasinLength;
double _MaxEnergy;
double _MaxFlowDepthAfterJump;
double _BasinIndex;
List<DissipationRow> dissipationrowlist;
Polyline plhjdesignapp1;
Polyline plhjExistingapp1;
Polyline plhjdesignapp2;
Polyline plhjExistingapp2;
Cascade _CascadeDesign = new Cascade();
```

```
Cascade _CascadeExisting = new Cascade();
System.Windows.Forms.GroupBox groupBoxdischargeandspillwaydata;
System.Windows.Forms.TextBox txtDesignDischarge;
System.Windows.Forms.TextBox txtActualDischarge;
System.Windows.Forms.Label labelQe;
System.Windows.Forms.Label labelB;
System.Windows.Forms.TextBox txtPierNumber;
System.Windows.Forms.TextBox txtPierWidth;
System.Windows.Forms.Label labelpierno;
System.Windows.Forms.Label labeln;
System.Windows.Forms.TextBox txtManning;
System.Windows.Forms.TextBox txtSpillwayLength;
System.Windows.Forms.Label labelpierwidth;
System.Windows.Forms.GroupBox groupBoxelevationdata;
System.Windows.Forms.TextBox txtDownstreamBedElevation;
System.Windows.Forms.TextBox txtDownstreamEnergyElevation;
System.Windows.Forms.TextBox txtCrestElevation;
System.Windows.Forms.Label labeldsenergyel;
System.Windows.Forms.Label labeldsbedel;
System.Windows.Forms.Label label5;
System.Windows.Forms.GroupBox groupBoxwaterwaydata;
System.Windows.Forms.Label label7;
System.Windows.Forms.ComboBox cmbWaterwayType;
System.Windows.Forms.Label lblStepHeight;
System.Windows.Forms.TextBox txtStepLength;
System.Windows.Forms.Label lblStepLength;
System.Windows.Forms.TextBox txtStepHeight;
System.Windows.Forms.Label lblChuteAngleDegree;
System.Windows.Forms.TextBox txtChuteAngleDegree;
System.Windows.Forms.Button btnCalculate;
System.Windows.Forms.GroupBox groupBoxoutputdata;
```

```
System.Windows.Forms.Label lblOutputActualHead;
System.Windows.Forms.Label lblOutputDesignHead;
System.Windows.Forms.Label lblOutputPointOfTangency;
System.Windows.Forms.TabControl tcVisual;
System.Windows.Forms.TabPage tabpgOverall;
System.Windows.Forms.TabPage tabpgCrest;
System.Windows.Forms.TabPage tabpgWaterway;
System.Windows.Forms.TabPage tabpgInput;
System.Windows.Forms.Button btnTables;
System.Windows.Forms.CheckBox chkbxActualDischarge;
System.Windows.Forms.CheckBox chkbxDesignDischarge;
System.Windows.Forms.GroupBox groupBox1;
System.Windows.Forms.CheckBox chkApproach1;
System.Windows.Forms.Button button1;
System.Windows.Forms.CheckBox chkApproach2;
System.Windows.Forms.Label lblOutputStillingBasinElevation;
System.Windows.Forms.Label lblOutputMaxEnergy;
System.Windows.Forms.Button btnAdjust;
System.Windows.Forms.Button btnResetStillingBasin;
System.Windows.Forms.CheckBox chkbxAlongPiers;
System.Windows.Forms.CheckBox chkbxPierBay;
System.Windows.Forms.CheckBox chkbxNoPier;
System.Windows.Forms.GroupBox grpBxEnvData;
System.Windows.Forms.TextBox txtTemperature;
System.Windows.Forms.Label lblTemperature;
System.Windows.Forms.Button button2;
System.Windows.Forms.ComboBox cmbCase;
System.Windows.Forms.Label lblOutputCascade;
```

A.2. Form Design Code

```
private void InitializeComponent()
{
```

```
System.Windows.Forms.Label labelQd;
this.groupBoxdischargeandspillwaydata = new
System.Windows.Forms.GroupBox();
this.txtDesignDischarge = new System.Windows.Forms.TextBox();
this.txtActualDischarge = new System.Windows.Forms.TextBox();
this.labelQe = new System.Windows.Forms.Label();
this.labelB = new System.Windows.Forms.Label();
this.txtPierNumber = new System.Windows.Forms.TextBox();
this.txtPierWidth = new System.Windows.Forms.TextBox();
this.labelpierno = new System.Windows.Forms.Label();
this.labeln = new System.Windows.Forms.Label();
this.txtManning = new System.Windows.Forms.TextBox();
this.txtSpillwayLength = new System.Windows.Forms.TextBox();
this.labelpierwidth = new System.Windows.Forms.Label();
this.groupBoxelevationdata = new System.Windows.Forms.GroupBox();
this.txtDownstreamBedElevation = new System.Windows.Forms.TextBox();
this.txtDownstreamEnergyElevation = new System.Windows.Forms.TextBox();
this.txtCrestElevation = new System.Windows.Forms.TextBox();
this.labeldsenergyel = new System.Windows.Forms.Label();
this.labeldsbedel = new System.Windows.Forms.Label();
this.label5 = new System.Windows.Forms.Label();
this.groupBoxwaterwaydata = new System.Windows.Forms.GroupBox();
this.label7 = new System.Windows.Forms.Label();
this.cmbWaterwayType = new System.Windows.Forms.ComboBox();
this.txtStepLength = new System.Windows.Forms.TextBox();
this.lblStepLength = new System.Windows.Forms.Label();
this.txtStepHeight = new System.Windows.Forms.TextBox();
this.txtChuteAngleDegree = new System.Windows.Forms.TextBox();
this.lblStepHeight = new System.Windows.Forms.Label();
this.lblChuteAngleDegree = new System.Windows.Forms.Label();
this.btnCalculate = new System.Windows.Forms.Button();
this.groupBoxoutputdata = new System.Windows.Forms.GroupBox();
```

```
this.lblOutputCascade = new System.Windows.Forms.Label();
this.lblOutputMaxEnergy = new System.Windows.Forms.Label();
this.lblOutputStillingBasinElevation = new System.Windows.Forms.Label();
this.lblOutputActualHead = new System.Windows.Forms.Label();
this.lblOutputDesignHead = new System.Windows.Forms.Label();
this.lblOutputPointOfTangency = new System.Windows.Forms.Label();
this.btnTables = new System.Windows.Forms.Button();
this.btnResetStillingBasin = new System.Windows.Forms.Button();
this.btnAdjust = new System.Windows.Forms.Button();
this.tcVisual = new System.Windows.Forms.TabControl();
this.tabpgInput = new System.Windows.Forms.TabPage();
this.tabpgOverall = new System.Windows.Forms.TabPage();
this.tabpgCrest = new System.Windows.Forms.TabPage();
this.chkbxNoPier = new System.Windows.Forms.CheckBox();
this.chkbxAlongPiers = new System.Windows.Forms.CheckBox();
this.chkbxPierBay = new System.Windows.Forms.CheckBox();
this.tabpgWaterway = new System.Windows.Forms.TabPage();
this.chkbxActualDischarge = new System.Windows.Forms.CheckBox();
this.chkbxDesignDischarge = new System.Windows.Forms.CheckBox();
this.groupBox1 = new System.Windows.Forms.GroupBox();
this.chkApproach2 = new System.Windows.Forms.CheckBox();
this.chkApproach1 = new System.Windows.Forms.CheckBox();
this.button1 = new System.Windows.Forms.Button();
this.grpBxEnvData = new System.Windows.Forms.GroupBox();
this.cmbCase = new System.Windows.Forms.ComboBox();
this.txtTemperature = new System.Windows.Forms.TextBox();
this.lblTemperature = new System.Windows.Forms.Label();
this.button2 = new System.Windows.Forms.Button();
labelQd = new System.Windows.Forms.Label();
this.groupBoxdischargeandspillwaydata.SuspendLayout();
this.groupBoxelevationdata.SuspendLayout();
```

```
this.groupBoxwaterwaydata.SuspendLayout();

this.groupBoxoutputdata.SuspendLayout();

this.tcVisual.SuspendLayout();

this.tabpgInput.SuspendLayout();

this.tabpgOverall.SuspendLayout();

this.tabpgCrest.SuspendLayout();

this.tabpgWaterway.SuspendLayout();

this.groupBox1.SuspendLayout();

this.grpBxEnvData.SuspendLayout();

this.SuspendLayout();

//

// labelQd

//

labelQd.Cursor = System.Windows.Forms.Cursors.Default;

labelQd.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

labelQd.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;

labelQd.Location = new System.Drawing.Point(50, 19);

labelQd.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

labelQd.Name = "labelQd";

labelQd.RightToLeft = System.Windows.Forms.RightToLeft.No;

labelQd.Size = new System.Drawing.Size(143, 23);

labelQd.TabIndex = 0;

labelQd.Text = "Design Discharge :";

labelQd.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

//

// groupBoxdischargeandspillwaydata

//

this.groupBoxdischargeandspillwaydata.Controls.Add(labelQd);

this.groupBoxdischargeandspillwaydata.Controls.Add(this.txtDesignDischarge
);
```

```
this.groupBoxdischargeandspillwaydata.Controls.Add(this.txtActualDischarge
);

this.groupBoxdischargeandspillwaydata.Controls.Add(this.labelQe);

this.groupBoxdischargeandspillwaydata.Controls.Add(this.labelB);

this.groupBoxdischargeandspillwaydata.Controls.Add(this.txtPierNumber);

this.groupBoxdischargeandspillwaydata.Controls.Add(this.txtPierWidth);

this.groupBoxdischargeandspillwaydata.Controls.Add(this.labelpierno);

this.groupBoxdischargeandspillwaydata.Controls.Add(this.labeln);

this.groupBoxdischargeandspillwaydata.Controls.Add(this.txtManning);

this.groupBoxdischargeandspillwaydata.Controls.Add(this.txtSpillwayLength)
;

this.groupBoxdischargeandspillwaydata.Controls.Add(this.labelpierwidth);

this.groupBoxdischargeandspillwaydata.Font = new
System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.groupBoxdischargeandspillwaydata.Location = new
System.Drawing.Point(12, 12);

this.groupBoxdischargeandspillwaydata.Name =
"groupBoxdischargeandspillwaydata";

this.groupBoxdischargeandspillwaydata.Size = new System.Drawing.Size(433,
96);

this.groupBoxdischargeandspillwaydata.TabIndex = 60;

this.groupBoxdischargeandspillwaydata.TabStop = false;

this.groupBoxdischargeandspillwaydata.Text = "Discharge and Spillway
Data";

//

// txtDesignDischarge

//

this.txtDesignDischarge.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.txtDesignDischarge.Location = new System.Drawing.Point(196, 18);

this.txtDesignDischarge.Margin = new System.Windows.Forms.Padding(2);

this.txtDesignDischarge.Name = "txtDesignDischarge";

this.txtDesignDischarge.Size = new System.Drawing.Size(50, 23);
```

```
this.txtDesignDischarge.TabIndex = 4;

//

// txtActualDischarge

//

this.txtActualDischarge.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.txtActualDischarge.Location = new System.Drawing.Point(196, 42);

this.txtActualDischarge.Margin = new System.Windows.Forms.Padding(2);

this.txtActualDischarge.Name = "txtActualDischarge";

this.txtActualDischarge.Size = new System.Drawing.Size(50, 23);

this.txtActualDischarge.TabIndex = 5;

//

// labelQe

//

this.labelQe.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.labelQe.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;

this.labelQe.Location = new System.Drawing.Point(50, 43);

this.labelQe.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.labelQe.Name = "labelQe";

this.labelQe.Size = new System.Drawing.Size(143, 23);

this.labelQe.TabIndex = 1;

this.labelQe.Text = "Existing Discharge :";

this.labelQe.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

//

// labelB

//

this.labelB.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.labelB.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;

this.labelB.Location = new System.Drawing.Point(243, 19);
```

```
this.labelB.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.labelB.Name = "labelB";

this.labelB.Size = new System.Drawing.Size(121, 23);

this.labelB.TabIndex = 20;

this.labelB.Text = "Spillway Length :";

this.labelB.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

//

// txtPierNumber

//

this.txtPierNumber.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.txtPierNumber.Location = new System.Drawing.Point(367, 42);

this.txtPierNumber.Margin = new System.Windows.Forms.Padding(2);

this.txtPierNumber.Name = "txtPierNumber";

this.txtPierNumber.Size = new System.Drawing.Size(50, 23);

this.txtPierNumber.TabIndex = 10;

//

// txtPierWidth

//

this.txtPierWidth.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.txtPierWidth.Location = new System.Drawing.Point(367, 66);

this.txtPierWidth.Margin = new System.Windows.Forms.Padding(2);

this.txtPierWidth.Name = "txtPierWidth";

this.txtPierWidth.Size = new System.Drawing.Size(50, 23);

this.txtPierWidth.TabIndex = 11;

//

// labelpierno

//

this.labelpierno.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
```

```
this.labelpierno.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;

this.labelpierno.Location = new System.Drawing.Point(246, 43);

this.labelpierno.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.labelpierno.Name = "labelpierno";

this.labelpierno.Size = new System.Drawing.Size(118, 23);

this.labelpierno.TabIndex = 13;

this.labelpierno.Text = "Pier Number :";

this.labelpierno.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

//

// labeln

//

this.labeln.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.labeln.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;

this.labeln.Location = new System.Drawing.Point(2, 67);

this.labeln.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.labeln.Name = "labeln";

this.labeln.Size = new System.Drawing.Size(191, 23);

this.labeln.TabIndex = 15;

this.labeln.Text = "Manning Roughness Coef. :";

this.labeln.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

//

// txtManning

//

this.txtManning.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.txtManning.Location = new System.Drawing.Point(196, 66);

this.txtManning.Margin = new System.Windows.Forms.Padding(2);

this.txtManning.Name = "txtManning";

this.txtManning.Size = new System.Drawing.Size(50, 23);

this.txtManning.TabIndex = 12;
```

```
//
// txtSpillwayLength
//
this.txtSpillwayLength.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.txtSpillwayLength.Location = new System.Drawing.Point(367, 18);
this.txtSpillwayLength.Margin = new System.Windows.Forms.Padding(2);
this.txtSpillwayLength.Name = "txtSpillwayLength";
this.txtSpillwayLength.Size = new System.Drawing.Size(50, 23);
this.txtSpillwayLength.TabIndex = 21;
//
// labelpierwidth
//
this.labelpierwidth.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
this.labelpierwidth.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;
this.labelpierwidth.Location = new System.Drawing.Point(278, 67);
this.labelpierwidth.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
this.labelpierwidth.Name = "labelpierwidth";
this.labelpierwidth.Size = new System.Drawing.Size(86, 23);
this.labelpierwidth.TabIndex = 14;
this.labelpierwidth.Text = "Pier Width :";
this.labelpierwidth.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
//
// groupBoxelevationdata
//

this.groupBoxelevationdata.Controls.Add(this.txtDownstreamBedElevation);

this.groupBoxelevationdata.Controls.Add(this.txtDownstreamEnergyElevation)
;
```

```
this.groupBoxelevationdata.Controls.Add(this.txtCrestElevation);

            this.groupBoxelevationdata.Controls.Add(this.labeldsenergyel);

this.groupBoxelevationdata.Controls.Add(this.labeldsbedel);

this.groupBoxelevationdata.Controls.Add(this.label5);

this.groupBoxelevationdata.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.groupBoxelevationdata.Location = new System.Drawing.Point(454, 12);

this.groupBoxelevationdata.Name = "groupBoxelevationdata";

this.groupBoxelevationdata.Size = new System.Drawing.Size(262, 96);

this.groupBoxelevationdata.TabIndex = 61;

this.groupBoxelevationdata.TabStop = false;

this.groupBoxelevationdata.Text = "Elevation Data";

//

// txtDownstreamBedElevation

//

this.txtDownstreamBedElevation.Font = new System.Drawing.Font("Trebuchet
MS", 9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.txtDownstreamBedElevation.Location = new System.Drawing.Point(204,
41);

this.txtDownstreamBedElevation.Margin = new
System.Windows.Forms.Padding(2);

this.txtDownstreamBedElevation.Name = "txtDownstreamBedElevation";

this.txtDownstreamBedElevation.Size = new System.Drawing.Size(50, 23);

this.txtDownstreamBedElevation.TabIndex = 18;

//

// txtDownstreamEnergyElevation

//

this.txtDownstreamEnergyElevation.Font = new
System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.txtDownstreamEnergyElevation.Location = new System.Drawing.Point(204,
65);
```

```
this.txtDownstreamEnergyElevation.Margin = new
System.Windows.Forms.Padding(2);

this.txtDownstreamEnergyElevation.Name = "txtDownstreamEnergyElevation";

this.txtDownstreamEnergyElevation.Size = new System.Drawing.Size(50, 23);

this.txtDownstreamEnergyElevation.TabIndex = 19;

//

// txtCrestElevation

//

this.txtCrestElevation.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));

this.txtCrestElevation.Location = new System.Drawing.Point(204, 17);

this.txtCrestElevation.Margin = new System.Windows.Forms.Padding(2);

this.txtCrestElevation.Name = "txtCrestElevation";

this.txtCrestElevation.Size = new System.Drawing.Size(50, 23);

this.txtCrestElevation.TabIndex = 40;

//

// labeldsenergyel

//

this.labeldsenergyel.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.labeldsenergyel.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;

this.labeldsenergyel.Location = new System.Drawing.Point(9, 66);

this.labeldsenergyel.Margin = new System.Windows.Forms.Padding(2, 0, 2,
0);

this.labeldsenergyel.Name = "labeldsenergyel";

this.labeldsenergyel.Size = new System.Drawing.Size(191, 23);

this.labeldsenergyel.TabIndex = 17;

this.labeldsenergyel.Text = "Downstream Energy Elevation :";

this.labeldsenergyel.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;

//
```

```
// labeldsbedel

//

this.labeldsbedel.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.labeldsbedel.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;

this.labeldsbedel.Location = new System.Drawing.Point(24, 42);

this.labeldsbedel.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.labeldsbedel.Name = "labeldsbedel";

this.labeldsbedel.Size = new System.Drawing.Size(176, 23);

this.labeldsbedel.TabIndex = 16;

this.labeldsbedel.Text = "Downstream Bed Elevation :";

this.labeldsbedel.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

//

// label5

//

this.label5.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.label5.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;

this.label5.Location = new System.Drawing.Point(71, 18);

this.label5.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(129, 23);

this.label5.TabIndex = 39;

this.label5.Text = "Crest Elevation :";

this.label5.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

//

// groupBoxwaterwaydata

//

this.groupBoxwaterwaydata.Controls.Add(this.label7);

            this.groupBoxwaterwaydata.Controls.Add(this.cmbWaterwayType);
```

```
this.groupBoxwaterwaydata.Controls.Add(this.txtStepLength);

this.groupBoxwaterwaydata.Controls.Add(this.lblStepLength);

this.groupBoxwaterwaydata.Controls.Add(this.txtStepHeight);

this.groupBoxwaterwaydata.Controls.Add(this.txtChuteAngleDegree);

this.groupBoxwaterwaydata.Controls.Add(this.lblStepHeight);


this.groupBoxwaterwaydata.Controls.Add(this.lblChuteAngleDegree);

this.groupBoxwaterwaydata.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(162)));

this.groupBoxwaterwaydata.Location = new System.Drawing.Point(12, 107);

this.groupBoxwaterwaydata.Name = "groupBoxwaterwaydata";

this.groupBoxwaterwaydata.Size = new System.Drawing.Size(219, 113);

this.groupBoxwaterwaydata.TabIndex = 63;

this.groupBoxwaterwaydata.TabStop = false;

this.groupBoxwaterwaydata.Text = "Waterway Data";

//

// label7

//

this.label7.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.label7.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;

this.label7.Location = new System.Drawing.Point(2, 20);

this.label7.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.label7.Name = "label7";

this.label7.Size = new System.Drawing.Size(108, 23);

this.label7.TabIndex = 44;

this.label7.Text = "Waterway Type=";

this.label7.TextAlign = System.Drawing.ContentAlignment.MiddleRight;

            //

// cmbWaterwayType

//
```

```
this.cmbWaterwayType.BackColor = System.Drawing.Color.White;

this.cmbWaterwayType.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList;

this.cmbWaterwayType.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.cmbWaterwayType.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.cmbWaterwayType.FormattingEnabled = true;

this.cmbWaterwayType.Items.AddRange(new object[] {

"Chute",

"Cascade"});

this.cmbWaterwayType.Location = new System.Drawing.Point(114, 18);

this.cmbWaterwayType.Margin = new System.Windows.Forms.Padding(2);

this.cmbWaterwayType.Name = "cmbWaterwayType";

this.cmbWaterwayType.Size = new System.Drawing.Size(100, 26);

this.cmbWaterwayType.TabIndex = 45;

this.cmbWaterwayType.Tag = "";

//

// txtStepLength

//

this.txtStepLength.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.txtStepLength.Location = new System.Drawing.Point(113, 75);

this.txtStepLength.Margin = new System.Windows.Forms.Padding(2);

this.txtStepLength.Name = "txtStepLength";

this.txtStepLength.Size = new System.Drawing.Size(50, 23);

this.txtStepLength.TabIndex = 53;

//

// lblStepLength

//

this.lblStepLength.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
```

```
this.lblStepLength.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;

this.lblStepLength.Location = new System.Drawing.Point(5, 75);

this.lblStepLength.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.lblStepLength.Name = "lblStepLength";

this.lblStepLength.Size = new System.Drawing.Size(104, 23);

this.lblStepLength.TabIndex = 52;

this.lblStepLength.Text = "Step Length=";

this.lblStepLength.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;

//

// txtStepHeight

//

this.txtStepHeight.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.txtStepHeight.Location = new System.Drawing.Point(113, 51);

this.txtStepHeight.Margin = new System.Windows.Forms.Padding(2);

this.txtStepHeight.Name = "txtStepHeight";

this.txtStepHeight.Size = new System.Drawing.Size(50, 23);

this.txtStepHeight.TabIndex = 55;

//

// txtChuteAngleDegree

//

this.txtChuteAngleDegree.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(162)));

this.txtChuteAngleDegree.Location = new System.Drawing.Point(113, 51);

this.txtChuteAngleDegree.Margin = new System.Windows.Forms.Padding(2);

this.txtChuteAngleDegree.Name = "txtChuteAngleDegree";

this.txtChuteAngleDegree.Size = new System.Drawing.Size(50, 23);

this.txtChuteAngleDegree.TabIndex = 9;

//

// lblStepHeight
```

```
//
this.lblStepHeight.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
this.lblStepHeight.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;
this.lblStepHeight.Location = new System.Drawing.Point(5, 51);
this.lblStepHeight.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
this.lblStepHeight.Name = "lblStepHeight";
this.lblStepHeight.Size = new System.Drawing.Size(104, 23);
this.lblStepHeight.TabIndex = 54;
this.lblStepHeight.Text = "Step Height=";
this.lblStepHeight.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
//
// lblChuteAngleDegree
//
this.lblChuteAngleDegree.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(162)));
this.lblChuteAngleDegree.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;
this.lblChuteAngleDegree.Location = new System.Drawing.Point(69, 51);
this.lblChuteAngleDegree.Margin = new System.Windows.Forms.Padding(2, 0,
2, 0);
this.lblChuteAngleDegree.Name = "lblChuteAngleDegree";
this.lblChuteAngleDegree.Size = new System.Drawing.Size(40, 23);
this.lblChuteAngleDegree.TabIndex = 8;
this.lblChuteAngleDegree.Text = "θ=";
this.lblChuteAngleDegree.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
//
// btnCalculate
//
this.btnCalculate.Cursor = System.Windows.Forms.Cursors.Default;
```

```
this.btnCalculate.Font = new System.Drawing.Font("Trebuchet MS", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.btnCalculate.Location = new System.Drawing.Point(917, 340);

this.btnCalculate.Margin = new System.Windows.Forms.Padding(2);

this.btnCalculate.Name = "btnCalculate";

this.btnCalculate.Size = new System.Drawing.Size(92, 29);

this.btnCalculate.TabIndex = 64;

this.btnCalculate.Text = "Calculate";

this.btnCalculate.UseVisualStyleBackColor = true;

this.btnCalculate.Visible = false;

//

// groupBoxoutputdata

//

this.groupBoxoutputdata.Controls.Add(this.lblOutputCascade);

this.groupBoxoutputdata.Controls.Add(this.lblOutputMaxEnergy);

this.groupBoxoutputdata.Controls.Add(this.lblOutputStillingBasinElevation)
;

this.groupBoxoutputdata.Controls.Add(this.lblOutputActualHead);

this.groupBoxoutputdata.Controls.Add(this.lblOutputDesignHead);

this.groupBoxoutputdata.Controls.Add(this.lblOutputPointOfTangency);

this.groupBoxoutputdata.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(162)));

this.groupBoxoutputdata.Location = new System.Drawing.Point(722, 16);

this.groupBoxoutputdata.Name = "groupBoxoutputdata";

this.groupBoxoutputdata.Size = new System.Drawing.Size(307, 178);

this.groupBoxoutputdata.TabIndex = 65;

this.groupBoxoutputdata.TabStop = false;

this.groupBoxoutputdata.Text = "Output Messages and Warnings";

//

// lblOutputCascade

//
```

```
this.lblOutputCascade.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(162)));

this.lblOutputCascade.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;

this.lblOutputCascade.Location = new System.Drawing.Point(5, 125);

this.lblOutputCascade.Margin = new System.Windows.Forms.Padding(2, 0, 2,
0);

this.lblOutputCascade.Name = "lblOutputCascade";

this.lblOutputCascade.Size = new System.Drawing.Size(292, 50);

this.lblOutputCascade.TabIndex = 71;

this.lblOutputCascade.Text = "lblOutputCascade";

//

// lblOutputMaxEnergy

//

this.lblOutputMaxEnergy.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(162)));

this.lblOutputMaxEnergy.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;

this.lblOutputMaxEnergy.Location = new System.Drawing.Point(5, 105);

this.lblOutputMaxEnergy.Margin = new System.Windows.Forms.Padding(2, 0, 2,
0);

this.lblOutputMaxEnergy.Name = "lblOutputMaxEnergy";

this.lblOutputMaxEnergy.Size = new System.Drawing.Size(292, 23);

this.lblOutputMaxEnergy.TabIndex = 67;

this.lblOutputMaxEnergy.Text = "lblOutputEnergy";

//

// lblOutputStillingBasinElevation

//

this.lblOutputStillingBasinElevation.Font = new
System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.lblOutputStillingBasinElevation.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;
```

```
this.lblOutputStillingBasinElevation.Location = new
System.Drawing.Point(5, 85);

this.lblOutputStillingBasinElevation.Margin = new
System.Windows.Forms.Padding(2, 0, 2, 0);

this.lblOutputStillingBasinElevation.Name =
"lblOutputStillingBasinElevation";

this.lblOutputStillingBasinElevation.Size = new System.Drawing.Size(292,
23);

this.lblOutputStillingBasinElevation.TabIndex = 66;

this.lblOutputStillingBasinElevation.Text =
"lblOutputStillingBasinElevation";

//

// lblOutputActualHead

//

this.lblOutputActualHead.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(162)));

this.lblOutputActualHead.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;

this.lblOutputActualHead.Location = new System.Drawing.Point(5, 45);

this.lblOutputActualHead.Margin = new System.Windows.Forms.Padding(2, 0,
2, 0);

this.lblOutputActualHead.Name = "lblOutputActualHead";

this.lblOutputActualHead.Size = new System.Drawing.Size(292, 23);

this.lblOutputActualHead.TabIndex = 39;

this.lblOutputActualHead.Text = "lblOutputExistingHead";

//

// lblOutputDesignHead

//

this.lblOutputDesignHead.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(162)));

this.lblOutputDesignHead.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;

this.lblOutputDesignHead.Location = new System.Drawing.Point(5, 25);
```

```
this.lblOutputDesignHead.Margin = new System.Windows.Forms.Padding(2, 0,
2, 0);

this.lblOutputDesignHead.Name = "lblOutputDesignHead";

this.lblOutputDesignHead.Size = new System.Drawing.Size(292, 23);

this.lblOutputDesignHead.TabIndex = 22;

this.lblOutputDesignHead.Text = "lblOutputDesignHead";

//

// lblOutputPointOfTangency

//

this.lblOutputPointOfTangency.Font = new System.Drawing.Font("Trebuchet
MS", 9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(162)));

this.lblOutputPointOfTangency.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;

this.lblOutputPointOfTangency.Location = new System.Drawing.Point(5, 65);

this.lblOutputPointOfTangency.Margin = new System.Windows.Forms.Padding(2,
0, 2, 0);

this.lblOutputPointOfTangency.Name = "lblOutputPointOfTangency";

this.lblOutputPointOfTangency.Size = new System.Drawing.Size(292, 23);

this.lblOutputPointOfTangency.TabIndex = 35;

this.lblOutputPointOfTangency.Text = "lblOutputPointOfTangency";

//

// btnTables

//

this.btnTables.Cursor = System.Windows.Forms.Cursors.Default;

this.btnTables.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.btnTables.Location = new System.Drawing.Point(951, 199);

this.btnTables.Margin = new System.Windows.Forms.Padding(2);

this.btnTables.Name = "btnTables";

this.btnTables.Size = new System.Drawing.Size(78, 33);

this.btnTables.TabIndex = 65;

this.btnTables.Text = "Tables";
```

```
this.btnTables.UseVisualStyleBackColor = true;

this.btnTables.Click += new System.EventHandler(this.btnTables_Click_1);

//

// btnResetStillingBasin

//

this.btnResetStillingBasin.Cursor = System.Windows.Forms.Cursors.Default;

this.btnResetStillingBasin.Font = new System.Drawing.Font("Trebuchet MS",
9.75F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.btnResetStillingBasin.Location = new System.Drawing.Point(772, 199);

this.btnResetStillingBasin.Margin = new System.Windows.Forms.Padding(2);

this.btnResetStillingBasin.Name = "btnResetStillingBasin";

this.btnResetStillingBasin.Size = new System.Drawing.Size(175, 33);

this.btnResetStillingBasin.TabIndex = 69;

this.btnResetStillingBasin.Text = "Reset Stilling Basin";

this.btnResetStillingBasin.UseVisualStyleBackColor = true;

//

// btnAdjust

//

this.btnAdjust.Cursor = System.Windows.Forms.Cursors.Default;

this.btnAdjust.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.btnAdjust.Location = new System.Drawing.Point(772, 199);

this.btnAdjust.Margin = new System.Windows.Forms.Padding(2);

this.btnAdjust.Name = "btnAdjust";

this.btnAdjust.Size = new System.Drawing.Size(175, 33);

this.btnAdjust.TabIndex = 68;

this.btnAdjust.Text = "Adjust Elevation";

this.btnAdjust.UseVisualStyleBackColor = true;

//

// tcVisual

//
```

```
this.tcVisual.Alignment = System.Windows.Forms.TabAlignment.Bottom;
this.tcVisual.Controls.Add(this.tabpgInput);
this.tcVisual.Controls.Add(this.tabpgOverall);
this.tcVisual.Controls.Add(this.tabpgCrest);
this.tcVisual.Controls.Add(this.tabpgWaterway);
this.tcVisual.Location = new System.Drawing.Point(12, 237);
this.tcVisual.Name = "tcVisual";
this.tcVisual.SelectedIndex = 0;
this.tcVisual.Size = new System.Drawing.Size(1019, 557);
this.tcVisual.TabIndex = 68;
//
// tabpgInput
//
this.tabpgInput.Controls.Add(this.canvasInput);
this.tabpgInput.Location = new System.Drawing.Point(4, 4);
this.tabpgInput.Name = "tabpgInput";
this.tabpgInput.Padding = new System.Windows.Forms.Padding(3);
this.tabpgInput.Size = new System.Drawing.Size(1011, 531);
this.tabpgInput.TabIndex = 4;
this.tabpgInput.Text = "Input";
this.tabpgInput.UseVisualStyleBackColor = true;
//
// canvasInput
//
this.canvasInput.BackColor = System.Drawing.SystemColors.Window;
this.canvasInput.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
this.canvasInput.Location = new System.Drawing.Point(1, 0);
this.canvasInput.Name = "canvasInput";
this.canvasInput.Size = new System.Drawing.Size(1010, 531);
this.canvasInput.TabIndex = 70;
//
```

```
// tabpgOverall
//
this.tabpgOverall.Controls.Add(this.canvasdrawoptions);
this.tabpgOverall.Controls.Add(this.canvasoverall);
this.tabpgOverall.Location = new System.Drawing.Point(4, 4);
this.tabpgOverall.Name = "tabpgOverall";
this.tabpgOverall.Padding = new System.Windows.Forms.Padding(3);
this.tabpgOverall.Size = new System.Drawing.Size(1011, 531);
this.tabpgOverall.TabIndex = 0;
this.tabpgOverall.Text = "Overall";
this.tabpgOverall.UseVisualStyleBackColor = true;
//
// canvasdrawoptions
//
this.canvasdrawoptions.BackColor = System.Drawing.SystemColors.Window;
this.canvasdrawoptions.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
this.canvasdrawoptions.Location = new System.Drawing.Point(824, 6);
this.canvasdrawoptions.Name = "canvasdrawoptions";
this.canvasdrawoptions.Size = new System.Drawing.Size(181, 63);
this.canvasdrawoptions.TabIndex = 70;
//
// canvasoverall
//
this.canvasoverall.BackColor = System.Drawing.SystemColors.Window;
this.canvasoverall.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
this.canvasoverall.Location = new System.Drawing.Point(1, 0);
this.canvasoverall.Name = "canvasoverall";
this.canvasoverall.Size = new System.Drawing.Size(1010, 531);
this.canvasoverall.TabIndex = 69;
//
```

```
// tabpgCrest
//
this.tabpgCrest.Controls.Add(this.chkbxNoPier);
this.tabpgCrest.Controls.Add(this.chkbxAlongPiers);
this.tabpgCrest.Controls.Add(this.chkbxPierBay);
this.tabpgCrest.Controls.Add(this.canvasCrest);
this.tabpgCrest.Location = new System.Drawing.Point(4, 4);
this.tabpgCrest.Name = "tabpgCrest";
this.tabpgCrest.Padding = new System.Windows.Forms.Padding(3);
this.tabpgCrest.Size = new System.Drawing.Size(1011, 531);
this.tabpgCrest.TabIndex = 1;
this.tabpgCrest.Text = "Crest";
this.tabpgCrest.UseVisualStyleBackColor = true;
//
// chkbxNoPier
//
this.chkbxNoPier.AutoSize = true;
this.chkbxNoPier.Checked = true;
this.chkbxNoPier.CheckState = System.Windows.Forms.CheckState.Checked;
this.chkbxNoPier.Location = new System.Drawing.Point(942, 8);
this.chkbxNoPier.Name = "chkbxNoPier";
this.chkbxNoPier.RightToLeft = System.Windows.Forms.RightToLeft.Yes;
this.chkbxNoPier.Size = new System.Drawing.Size(61, 17);
this.chkbxNoPier.TabIndex = 76;
this.chkbxNoPier.Text = "No Pier";
this.chkbxNoPier.TextImageRelation =
System.Windows.Forms.TextImageRelation.TextAboveImage;
this.chkbxNoPier.UseVisualStyleBackColor = true;
//
// chkbxAlongPiers
//
this.chkbxAlongPiers.AutoSize = true;
```

```
this.chkbxAlongPiers.Checked = true;
this.chkbxAlongPiers.CheckState = System.Windows.Forms.CheckState.Checked;
this.chkbxAlongPiers.Location = new System.Drawing.Point(925, 54);
this.chkbxAlongPiers.Name = "chkbxAlongPiers";
this.chkbxAlongPiers.RightToLeft = System.Windows.Forms.RightToLeft.Yes;
this.chkbxAlongPiers.Size = new System.Drawing.Size(79, 17);
this.chkbxAlongPiers.TabIndex = 75;
this.chkbxAlongPiers.Text = "Along Piers";
this.chkbxAlongPiers.TextImageRelation =
System.Windows.Forms.TextImageRelation.TextAboveImage;
this.chkbxAlongPiers.UseVisualStyleBackColor = true;
//
// chkbxPierBay
//
this.chkbxPierBay.AutoSize = true;
this.chkbxPierBay.Checked = true;
this.chkbxPierBay.CheckState = System.Windows.Forms.CheckState.Checked;
this.chkbxPierBay.Location = new System.Drawing.Point(876, 31);
this.chkbxPierBay.Name = "chkbxPierBay";
this.chkbxPierBay.RightToLeft = System.Windows.Forms.RightToLeft.Yes;
this.chkbxPierBay.Size = new System.Drawing.Size(127, 17);
this.chkbxPierBay.TabIndex = 74;
this.chkbxPierBay.Text = "Centerline of Pier Bay";
this.chkbxPierBay.TextImageRelation =
System.Windows.Forms.TextImageRelation.TextAboveImage;
this.chkbxPierBay.UseVisualStyleBackColor = true;
//
// canvasCrest
//
this.canvasCrest.BackColor = System.Drawing.SystemColors.Window;
this.canvasCrest.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
```

```
this.canvasCrest.Location = new System.Drawing.Point(0, 0);
this.canvasCrest.Name = "canvasCrest";
this.canvasCrest.Size = new System.Drawing.Size(1011, 531);
this.canvasCrest.TabIndex = 68;
//
// tabpgWaterway
//
this.tabpgWaterway.Controls.Add(this.canvaswaterway);
this.tabpgWaterway.Controls.Add(this.btnCalculate);
this.tabpgWaterway.Location = new System.Drawing.Point(4, 4);
this.tabpgWaterway.Name = "tabpgWaterway";
this.tabpgWaterway.Size = new System.Drawing.Size(1011, 531);
this.tabpgWaterway.TabIndex = 2;
this.tabpgWaterway.Text = "Waterway";
this.tabpgWaterway.UseVisualStyleBackColor = true;
//
// canvaswaterway
//
this.canvaswaterway.BackColor = System.Drawing.SystemColors.Window;
this.canvaswaterway.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
this.canvaswaterway.Location = new System.Drawing.Point(1, 0);
this.canvaswaterway.Name = "canvaswaterway";
this.canvaswaterway.Size = new System.Drawing.Size(1010, 531);
this.canvaswaterway.TabIndex = 69;
//
// chkbxActualDischarge
//
this.chkbxActualDischarge.AutoSize = true;
this.chkbxActualDischarge.Location = new System.Drawing.Point(13, 39);
this.chkbxActualDischarge.Name = "chkbxActualDischarge";
```

```
this.chkbxActualDischarge.RightToLeft =
System.Windows.Forms.RightToLeft.Yes;

this.chkbxActualDischarge.Size = new System.Drawing.Size(135, 22);

this.chkbxActualDischarge.TabIndex = 73;

this.chkbxActualDischarge.Text = "Existing Discharge";

this.chkbxActualDischarge.TextImageRelation =
System.Windows.Forms.TextImageRelation.TextAboveImage;

this.chkbxActualDischarge.UseVisualStyleBackColor = true;

//

// chkbxDesignDischarge

//

this.chkbxDesignDischarge.AutoSize = true;

this.chkbxDesignDischarge.Checked = true;

this.chkbxDesignDischarge.CheckState =
System.Windows.Forms.CheckState.Checked;

this.chkbxDesignDischarge.Location = new System.Drawing.Point(21, 19);

this.chkbxDesignDischarge.Name = "chkbxDesignDischarge";

this.chkbxDesignDischarge.RightToLeft =
System.Windows.Forms.RightToLeft.Yes;

this.chkbxDesignDischarge.Size = new System.Drawing.Size(127, 22);

this.chkbxDesignDischarge.TabIndex = 72;

this.chkbxDesignDischarge.Text = "Design Discharge";

this.chkbxDesignDischarge.TextImageRelation =
System.Windows.Forms.TextImageRelation.TextAboveImage;

this.chkbxDesignDischarge.UseVisualStyleBackColor = true;

//

// groupBox1

//

this.groupBox1.Controls.Add(this.chkApproach2);

this.groupBox1.Controls.Add(this.chkApproach1);

this.groupBox1.Controls.Add(this.chkbxActualDischarge);

this.groupBox1.Controls.Add(this.button1);

this.groupBox1.Controls.Add(this.chkbxDesignDischarge);
```

```
this.groupBox1.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.groupBox1.Location = new System.Drawing.Point(417, 107);

this.groupBox1.Name = "groupBox1";

this.groupBox1.Size = new System.Drawing.Size(299, 113);

this.groupBox1.TabIndex = 69;

this.groupBox1.TabStop = false;

this.groupBox1.Text = "Draw Options";

//

// chkApproach2

//

this.chkApproach2.Location = new System.Drawing.Point(154, 39);

this.chkApproach2.Name = "chkApproach2";

this.chkApproach2.RightToLeft = System.Windows.Forms.RightToLeft.Yes;

this.chkApproach2.Size = new System.Drawing.Size(134, 22);

this.chkApproach2.TabIndex = 75;

this.chkApproach2.Text = "Approach 2";

this.chkApproach2.TextImageRelation =
System.Windows.Forms.TextImageRelation.TextAboveImage;

this.chkApproach2.UseVisualStyleBackColor = true;

//

// chkApproach1

//

this.chkApproach1.Checked = true;

this.chkApproach1.CheckState = System.Windows.Forms.CheckState.Checked;

this.chkApproach1.Location = new System.Drawing.Point(154, 19);

this.chkApproach1.Name = "chkApproach1";

this.chkApproach1.RightToLeft = System.Windows.Forms.RightToLeft.Yes;

this.chkApproach1.Size = new System.Drawing.Size(134, 22);

this.chkApproach1.TabIndex = 74;

this.chkApproach1.Text = "Approach 1";
```

```
this.chkApproach1.TextImageRelation =
System.Windows.Forms.TextImageRelation.TextAboveImage;

this.chkApproach1.UseVisualStyleBackColor = true;

//

// button1

//

this.button1.Cursor = System.Windows.Forms.Cursors.Default;

this.button1.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.button1.Location = new System.Drawing.Point(423, 149);

this.button1.Margin = new System.Windows.Forms.Padding(2);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(61, 30);

this.button1.TabIndex = 65;

this.button1.Text = "Tables";

this.button1.UseVisualStyleBackColor = true;

//

// grpBxEnvData

//

this.grpBxEnvData.Controls.Add(this.cmbCase);

this.grpBxEnvData.Controls.Add(this.txtTemperature);

this.grpBxEnvData.Controls.Add(this.lblTemperature);

this.grpBxEnvData.Controls.Add(this.button2);

this.grpBxEnvData.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.grpBxEnvData.Location = new System.Drawing.Point(237, 107);

this.grpBxEnvData.Name = "grpBxEnvData";

this.grpBxEnvData.Size = new System.Drawing.Size(174, 113);

this.grpBxEnvData.TabIndex = 71;

this.grpBxEnvData.TabStop = false;

this.grpBxEnvData.Text = "Env. Data";
```

```
//
// cmbCase
//
this.cmbCase.AutoCompleteCustomSource.AddRange(new string[] {
"Case 1",
"Case 2",
"Case 3",
"Case 4"});
this.cmbCase.BackColor = System.Drawing.Color.White;
this.cmbCase.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList;
this.cmbCase.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.cmbCase.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(162)));
this.cmbCase.FormattingEnabled = true;
this.cmbCase.Items.AddRange(new object[] {
"Case 1",
"Case 2",
"Case 3",
"Case 4",
"Case 5"});
this.cmbCase.Location = new System.Drawing.Point(28, 75);
this.cmbCase.Margin = new System.Windows.Forms.Padding(2);
this.cmbCase.Name = "cmbCase";
this.cmbCase.Size = new System.Drawing.Size(126, 26);
this.cmbCase.TabIndex = 72;
this.cmbCase.Tag = "";
//
// txtTemperature
//
```

```
this.txtTemperature.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.txtTemperature.Location = new System.Drawing.Point(104, 20);

this.txtTemperature.Margin = new System.Windows.Forms.Padding(2);

this.txtTemperature.Name = "txtTemperature";

this.txtTemperature.Size = new System.Drawing.Size(50, 23);

this.txtTemperature.TabIndex = 67;

//

// lblTemperature

//

this.lblTemperature.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
((byte)(0)));

this.lblTemperature.ImageAlign =
System.Drawing.ContentAlignment.MiddleRight;

this.lblTemperature.Location = new System.Drawing.Point(5, 19);

this.lblTemperature.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);

this.lblTemperature.Name = "lblTemperature";

this.lblTemperature.Size = new System.Drawing.Size(97, 23);

this.lblTemperature.TabIndex = 66;

this.lblTemperature.Text = "Temperature :";

this.lblTemperature.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;

//

// button2

//

this.button2.Cursor = System.Windows.Forms.Cursors.Default;

this.button2.Font = new System.Drawing.Font("Trebuchet MS", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(162)));

this.button2.Location = new System.Drawing.Point(423, 149);

this.button2.Margin = new System.Windows.Forms.Padding(2);

this.button2.Name = "button2";
```

```
this.button2.Size = new System.Drawing.Size(61, 30);
this.button2.TabIndex = 65;
this.button2.Text = "Tables";
this.button2.UseVisualStyleBackColor = true;
//
// FrmSpillway
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(1038, 797);
this.Controls.Add(this.grpBxEnvData);
this.Controls.Add(this.groupBox1);
this.Controls.Add(this.groupBoxoutputdata);
this.Controls.Add(this.tcVisual);
this.Controls.Add(this.groupBoxwaterwaydata);
this.Controls.Add(this.groupBoxelevationdata);
this.Controls.Add(this.groupBoxdischargeandspillwaydata);
this.Controls.Add(this.btnTables);
this.Controls.Add(this.btnAdjust);
this.Controls.Add(this.btnResetStillingBasin);
this.Name = "FrmSpillway";
this.Text = " Spillway Design";
this.groupBoxdischargeandspillwaydata.ResumeLayout(false);
this.groupBoxdischargeandspillwaydata.PerformLayout();
this.groupBoxelevationdata.ResumeLayout(false);
this.groupBoxelevationdata.PerformLayout();
this.groupBoxwaterwaydata.ResumeLayout(false);
this.groupBoxwaterwaydata.PerformLayout();
this.groupBoxoutputdata.ResumeLayout(false);
this.tcVisual.ResumeLayout(false);
this.tabpgInput.ResumeLayout(false);
```

```
this.tabpgOverall.ResumeLayout(false);
this.tabpgCrest.ResumeLayout(false);
this.tabpgCrest.PerformLayout();
this.tabpgWaterway.ResumeLayout(false);
this.groupBox1.ResumeLayout(false);
this.groupBox1.PerformLayout();
this.grpBxEnvData.ResumeLayout(false);
this.grpBxEnvData.PerformLayout();
this.ResumeLayout(false);
}
```

A.3. Data Types

```
class AeratedChuteRow
{
public AeratedChuteRow()
{
}
public double HorizontalStation;
public double StreamwiseDistance;
public double BottomElevation;
public double NormalizedStreamwiseDistance;
public double AeratedWaterDepth;
public double BoundaryLayerThickness;
public double AngleRadian;
public double RelativeHead;
}
class Cascade
{
public Cascade()
{
}
public double UnitDischarge;
```

```
public double CascadeAngleRadian;
public double CascadeAngle;
public double dcOnset;
public double dc;
public double ks;
public double Fstar;
public double Li;
public double Lc;
public double di;
public double fe;
public double UniformDepth;
public double Area;
public double WettedPerimeter;
public double HydraulicRadius;
public double HydraulicDepth;
public double ErrorTerm;
public double Velocity;
public double TotalHeight;
public bool IsFlow
class Crest
{
public Crest()
{
}
public XYPT InsPt;
public XYPT PointOfTangency;
}
class CrestGeometryRow
{
public CrestGeometryRow()
{
```

```
}
public double XoverHd;
public double Xstar;
public double lnXstar;
public double YStar;
public double BottomStation;
public double BottomElevation;
public double DistanceToPreviousNode;
public double StreamwiseDistance;
public XYPT Location;
}
class CrestWSPRow
{
public CrestWSPRow()
{
}
public double Station;
public double SstarDesign;
public double SstarExisting;
public double VerticalHeightDesign;
public double VerticalHeightExisting;
public double YStar;
public double BottomStation;
public double BottomElevation;
public XYPT Location;
public double Slope;
public double VelocityDesign;
public double VelocityExisting;
public double PressureHeadDesign;
public double PressureHeadExisting;
public double TotalHeadDesign;
```

```
public double TotalHeadExisting;
public double StreamwiseDistance { get; set; }
}
class DissipationRow
{
public DissipationRow()
{
}

public enum eApproachDesignIndex
{
approach1Design = 0,
approach1Existing = 1,
approach2Design = 2,
approach2Existing = 3
}

public double StillingBasinElevation;
public string Approach;
public double Discharge;
public double SpillwayRoughLength;
public double FlowDepthBeforeJump;
public double FroudeNumberBeforeJump;
public double VelocityBeforeJump;
public double EnergyLevelBeforeJump;
public double FlowDepthAfterJump;
public double FroudeNumberAfterJump;
public double VelocityAfterJump;
public double EnergyLevelAfterJump;
public double StillingBasinLength;
public string BasinType;
```

```
public eApproachDesignIndex approachdesignindex;
}
class PressureRow
{
public enum dtPressureType
{
NoPier,
PierBay,
AlongPiers
}
public PressureRow()
{
}
public double DesignHead;
public double ExistingHead;
public double HeadRatio;
}
class WaterwayGeometryRow
{
public WaterwayGeometryRow()
{
}
public double BottomStation;
public double BottomElevation;
public double DistanceToPreviousNode;
public double StreamwiseDistance;
public XYPT Location;
}

public class WSPTable
{
```

```
public WSPTable()
{
}

private object _WSPTable;
public List<WSPTableRow> RowList = new List<WSPTableRow>();
public double KnownFlowDepth;
public double CalculationInterval;
public int CalculationStepNumber;
}
public class WSPTableRow
{
public WSPTableRow()
{
}

public double Index { get; set; }
public double ZCoord { get; set; }
public double FlowDepth { get; set; }
public double Area { get; set; }
public double WettedPerimeter { get; set; }
public double HydraulicRadius { get; set; }
public double HydraulicDepth { get; set; }
public double Velocity { get; set; }
public double FroudeNumber { get; set; }
public double FrictionSlope { get; set; }
public double TotalHead { get; set; }
public double AveragedFrictionSlopeWithPreviousSection { get; set; }
public double HydraulicLoss { get; set; }
public double PreviousTotalHead { get; set; }
public double ErrorTerm { get; set; }
```

```
}
```

A.4. Functions

```
class HydraulicSolver

{

public HydraulicSolver()

{}

public WSPTableRow StandardStepMethodInitialRowCalculations(double
FlowDepth, double ChuteAngle, double Datum, double Width, double
Discharge, double Manning)

{

WSPTableRow initXS = new WSPTableRow();

initXS.Index = 0;

initXS.ZCoord = Datum;

initXS.FlowDepth = FlowDepth;

initXS.Area = Width * FlowDepth;

initXS.WettedPerimeter = Width + 2 * FlowDepth;

initXS.HydraulicRadius = initXS.Area / initXS.WettedPerimeter;

initXS.HydraulicDepth = FlowDepth;

initXS.Velocity = Discharge / initXS.Area;

initXS.FroudeNumber = initXS.Velocity / Math.Sqrt(9.81 * FlowDepth);

initXS.FrictionSlope = Math.Pow(initXS.Velocity, 2) * Math.Pow(Manning, 2)
/ Math.Pow(initXS.HydraulicRadius, 1.333);

initXS.TotalHead = initXS.ZCoord + FlowDepth * Math.Cos(ChuteAngle *
Math.PI / 180) + Math.Pow(initXS.Velocity, 2) / 19.62;

return initXS;

}

public WSPTableRow StandardStepMethodRowCalculations(WSPTableRow prevXS,
double Datum, double CalculationInterval, double FlowDepth, double
ChuteAngle, double Discharge, double Manning, double Width)

{

double ChannelSlope = Math.Tan(ChuteAngle * Math.PI / 180);

WSPTableRow XS = new WSPTableRow();

XS.Index = prevXS.Index + 1;
```

```
XS.ZCoord = Datum - ChannelSlope * CalculationInterval * XS.Index;

XS.FlowDepth = FlowDepth;

XS.Area = Width * FlowDepth;

XS.WettedPerimeter = Width + FlowDepth * 2;

XS.HydraulicDepth = FlowDepth;

XS.HydraulicRadius = XS.Area / XS.WettedPerimeter;

XS.Velocity = Discharge / XS.Area;

XS.FroudeNumber = XS.Velocity / Math.Sqrt(9.81 * FlowDepth);

XS.FrictionSlope = Math.Pow(XS.Velocity, 2) * Math.Pow(Manning, 2) /
Math.Pow(XS.HydraulicRadius, 4.0 / 3.0);

XS.TotalHead = XS.ZCoord + FlowDepth * Math.Cos(ChuteAngle * Math.PI /
180) + Math.Pow(XS.Velocity, 2) / 19.62;

XS.AveragedFrictionSlopeWithPreviousSection = (XS.FrictionSlope +
prevXS.FrictionSlope) / 2.0;

XS.PreviousTotalHead = prevXS.TotalHead;

XS.HydraulicLoss = XS.AveragedFrictionSlopeWithPreviousSection *
CalculationInterval;

XS.ErrorTerm = (XS.PreviousTotalHead - XS.TotalHead) - XS.HydraulicLoss;

return XS;

}

public DissipationRow CreateDissipationRow(string Approach, double
Discharge, double FlowDepthBeforeJump, double SpillwayRoughLength, double
StillingBasinElevation, double BasinIndex, double AngleRadian)

{

DissipationRow row = new DissipationRow();

row.Approach = Approach;

row.Discharge = Discharge;

row.StillingBasinElevation = StillingBasinElevation;

row.SpillwayRoughLength = SpillwayRoughLength;

row.FlowDepthBeforeJump = FlowDepthBeforeJump;

row.VelocityBeforeJump =
DissipationRow.CalculateVelocity(row.FlowDepthBeforeJump, row.Discharge,
row.SpillwayRoughLength);
```

```
row.FroudeNumberBeforeJump =
DissipationRow.CalculateFroudeNumber(row.FlowDepthBeforeJump,
row.VelocityBeforeJump);

row.EnergyLevelBeforeJump =
DissipationRow.CalculateEnergyLevel(row.StillingBasinElevation,
row.FlowDepthBeforeJump, row.VelocityBeforeJump, AngleRadian);

row.FlowDepthAfterJump =
DissipationRow.CalculateFlowDepthAfterJump(row.FlowDepthBeforeJump,
row.FroudeNumberBeforeJump);

row.VelocityAfterJump =
DissipationRow.CalculateVelocity(row.FlowDepthAfterJump, row.Discharge,
row.SpillwayRoughLength);

row.FroudeNumberAfterJump =
DissipationRow.CalculateFroudeNumber(row.FlowDepthAfterJump,
row.VelocityAfterJump);

row.EnergyLevelAfterJump =
DissipationRow.CalculateEnergyLevel(row.StillingBasinElevation,
row.FlowDepthAfterJump, row.VelocityAfterJump, 0);

row.BasinType =
DissipationRow.DetermineStillingBasinType(row.FroudeNumberBeforeJump,
row.VelocityBeforeJump);

row.StillingBasinLength =
DissipationRow.CalculateStillingBasinLength(row.FroudeNumberBeforeJump,
row.VelocityBeforeJump, row.FlowDepthAfterJump);

return row;

}

public double CalculateDesignHead(double DesignDischarge, double
SpillwayRoughLength, double PierNumber, double PierWidth)

{

double DesignHead = 0.01;

double DischargeCoefficient = CalculateDischargeCoefficient(1);

double SpillwayEffectiveLength =
CalculateSpillwayEffectiveLength(SpillwayRoughLength, DesignHead,
PierNumber, PierWidth);

double CalculatedDesignDischarge = DischargeCoefficient *
SpillwayEffectiveLength * Math.Sqrt(2.0 * 9.81 * Math.Pow(DesignHead, 3));


while (CalculatedDesignDischarge < DesignDischarge)

{
```

```
DesignHead += 0.001;

SpillwayEffectiveLength =
CalculateSpillwayEffectiveLength(SpillwayRoughLength, DesignHead,
PierNumber, PierWidth);

if (SpillwayEffectiveLength <= 0)

{

return 0;

}

CalculatedDesignDischarge = DischargeCoefficient * SpillwayEffectiveLength
* Math.Sqrt(2.0 * 9.81 * Math.Pow(DesignHead, 3));

}

return DesignHead;

}

public double CalculateActualHead(double ActualDischarge, double
DesignHead, double SpillwayRoughLength, double PierNumber, double
PierWidth)

{

double ActualHead = 0.01;

double ActualHeadOverDesignHead = ActualHead / DesignHead;

double DischargeCoefficient =
CalculateDischargeCoefficient(ActualHeadOverDesignHead);

double SpillwayEffectiveLength =
CalculateSpillwayEffectiveLength(SpillwayRoughLength, ActualHead,
PierNumber, PierWidth);

double CalculatedActualDischarge = DischargeCoefficient *
SpillwayEffectiveLength * Math.Sqrt(2.0 * 9.81 * Math.Pow(ActualHead, 3));

while (CalculatedActualDischarge < ActualDischarge)

{

ActualHead += 0.001;

ActualHeadOverDesignHead = ActualHead / DesignHead;

DischargeCoefficient =
CalculateDischargeCoefficient(ActualHeadOverDesignHead);

SpillwayEffectiveLength =
CalculateSpillwayEffectiveLength(SpillwayRoughLength, ActualHead,
PierNumber, PierWidth);

if (SpillwayEffectiveLength <= 0)
```

```
{
return 0;
}
CalculatedActualDischarge = DischargeCoefficient * SpillwayEffectiveLength
* Math.Sqrt(2.0 * 9.81 * Math.Pow(ActualHead, 3));
}
return ActualHead;
}


public double CalculateDischargeCoefficient(double
ActualHeadOverDesignHead)
{
double Cd = 2.0 / 3.0 / Math.Sqrt(3) * (1.0 + (4.0 *
ActualHeadOverDesignHead / (9.0 + 5.0 * ActualHeadOverDesignHead)));
return Cd;
}
public double CalculateSpillwayEffectiveLength(double SpillwayRoughLength,
double FlowHead, double PierNumber, double PierWidth)
{
double Ka = 0.1;
double Kp = 0.01;
double Ln = SpillwayRoughLength - PierNumber * PierWidth;
double Le = Ln - ((Ka + Kp * PierNumber) * 2 * FlowHead);
return Le;
}
public double CalculateNormalizedStreamwiseDistance(double
StreamwiseDistance, double Width, double ChuteAngleRadian, double
RoughnessHeightAsMeter, double Discharge)
{
double CriticalDepth = Math.Pow((Math.Pow(Discharge, 2) / 9.81 /
Math.Pow(Width, 2)), 0.3333);
double UniformDepth = Math.Pow(Discharge *
Math.Pow(RoughnessHeightAsMeter, 1.0 / 6.0) / 6.75 / Math.Sqrt(9.81) /
Width / Math.Sqrt(Math.Sin(ChuteAngleRadian)), 0.6);
```

```
double NormalizedStreamWiseDistance = Math.Sin(ChuteAngleRadian) *
Math.Pow((UniformDepth / CriticalDepth), 3) * (StreamwiseDistance /
UniformDepth);

return NormalizedStreamWiseDistance;

}

public double CalculateAeratedWaterDepth(double StreamwiseDistance, double
NormalizedStreamWiseDistance, double Width, double ChuteAngleRadian,
double RoughnessHeightAsMeter, double Discharge)

{

double CriticalDepth = Math.Pow((Math.Pow(Discharge, 2) / 9.81 /
Math.Pow(Width, 2)), 0.3333);

double UniformDepth = Math.Pow(Discharge *
Math.Pow(RoughnessHeightAsMeter, 1.0 / 6.0) / 6.75 / Math.Sqrt(9.81) /
Width / Math.Sqrt(Math.Sin(ChuteAngleRadian)), 0.6);


double AerationDistance = CalculateAerationDistance(CriticalDepth,
RoughnessHeightAsMeter, ChuteAngleRadian);

if (NormalizedStreamWiseDistance <= AerationDistance)

{

double AeratedWaterDepth = UniformDepth / (1 - (Math.Exp(-10 / 3 *
NormalizedStreamWiseDistance) * (1 - UniformDepth / CriticalDepth)));

return AeratedWaterDepth;

}

else

{

double AeratedWaterDepth = UniformDepth / (1 - (Math.Exp(-10 / 3 *
NormalizedStreamWiseDistance) * (1 - UniformDepth / CriticalDepth)));

double AirConcentration = CalculateAirConcentration(UniformDepth,
StreamwiseDistance - AerationDistance);

double ModifiedWaterDepth = AeratedWaterDepth / (1 - AirConcentration);

return ModifiedWaterDepth;

}


}

public double CalculateCriticalDepth(double Discharge, double Width)

{
```

```
double CriticalDepth = Math.Pow((Math.Pow(Discharge, 2) / 9.81 /
Math.Pow(Width, 2)), 0.3333);

return CriticalDepth;

}

public double CalculateAerationDistance(double CriticalDepth, double
RoughnessHeightAsMeter, double ChuteAngleRadian)

{

double AerationDistance = Math.Pow(Math.Sin(ChuteAngleRadian), -3.0 / 5.0)
* Math.Pow((RoughnessHeightAsMeter / CriticalDepth), -0.08) * 16 *
CriticalDepth;

return AerationDistance;

}

public double CalculateAirConcentration(double UniformDepth, double
DistanceToInceptionPoint)

{

double AirConcentration = 0.48 - 0.48 * Math.Exp(-0.01061 *
DistanceToInceptionPoint / UniformDepth);

return AirConcentration;

}

public double CalculateBoundaryLayerThickness2(double StreamwiseDistance,
double ChuteAngleRadian, double RoughnessHeightAsMeter, double Discharge)

{

var a = Math.Pow(Math.Sin(ChuteAngleRadian), 2);

double BoundaryLayerThickness = Math.Pow((RoughnessHeightAsMeter /
(StreamwiseDistance+0.01) * Math.Pow(Math.Sin(ChuteAngleRadian), 2)), 1.0
/ 7.0) * 0.029 * StreamwiseDistance;

return BoundaryLayerThickness;

}


public double CalculateAngleOfPoint(double BottomStation, double
DesignHead)

{

double Angle = Math.Atan(Math.Pow(BottomStation / DesignHead, 0.85) * 1.85
* 0.5);

return Angle;

}
```

```
public double CalculateVaporPressure(double temperature)
{
double VaporPressureKpa;
double A, B, C;
if (temperature < 100)
{
A = 8.07131;
B = 1730.63;
C = 233.426;
}
else
{
A = 8.14019;
B = 1810.94;
C = 244.485;
}
VaporPressureKpa = Math.Pow(10, A - (B / (C + temperature))) * 0.133322;
//kPa
return VaporPressureKpa;
}
public double CalculateFluidDensity(double temperature)
{
double A = 0.14395;
double B = 0.0112;
double C = 649.727;
double D = 0.05107;
double tKelvin = temperature + 273.15;
double Density = A / Math.Pow(B, (Math.Pow((1 - tKelvin / C), D) + 1));
return Density;
}
public double CalculateCavitationParameter(double temperature, double
flowheight, double velocity)
```

```
{
double VaporPressure = CalculateVaporPressure(temperature);
double density = CalculateFluidDensity(temperature);
double CavitationParameter = (101 + (flowheight * 9.81) - VaporPressure) *
1000 / density / (Math.Pow(velocity, 2) / 2);
return CavitationParameter;
}
}
public static double CalculateVelocity(double FlowDepth, double Discharge,
double RoughSpillwayLength)
{
double Velocity = Discharge / FlowDepth / RoughSpillwayLength;
return Velocity;
}
public static double CalculateFroudeNumber(double FlowDepth, double
Velocity)
{
double FroudeNumber = Velocity / Math.Sqrt(9.81 * FlowDepth);
return FroudeNumber;
}
public static double CalculateEnergyLevel(double DownstreamBedElevation,
double FlowDepth, double Velocity, double ChuteAngleRadian)
{
double EnergyLevel = DownstreamBedElevation + FlowDepth *
Math.Cos(ChuteAngleRadian) + Math.Pow(Velocity, 2) / 19.62;
return EnergyLevel;
}
public static double CalculateFlowDepthAfterJump(double
FlowDepthBeforeJump, double FroudeNumberBeforeJump)
{
double FlowDepthAfterJump = (Math.Pow((Math.Pow(FroudeNumberBeforeJump, 2)
* 8 + 1), 0.5) - 1) / 2 * FlowDepthBeforeJump;
return FlowDepthAfterJump;
}
```

```
public static string DetermineStillingBasinType(double
FroudeNumberBeforeJump, double VelocityBeforeJump)

{

string BasinType = "";

if (FroudeNumberBeforeJump <= 2.5)

{

BasinType = "Type I";

}

else if (FroudeNumberBeforeJump > 4.5 & VelocityBeforeJump > 15)

{

BasinType = "Type II";

}

else if (FroudeNumberBeforeJump > 4.5 & VelocityBeforeJump <= 15)

{

BasinType = "Type III";

}

else if (FroudeNumberBeforeJump > 2.5 & FroudeNumberBeforeJump <= 4.5)

{

BasinType = "Type IV";

}

return BasinType;

}

public static double CalculateStillingBasinLength(double
FroudeNumberBeforeJump, double VelocityBeforeJump, double
FlowDepthAfterJump)

{

double StillingBasinLength = 0;

double BasinIndex = 0;

if (FroudeNumberBeforeJump <= 2.5)

{

BasinIndex = 0; //Type1

}
```

```
else if (FroudeNumberBeforeJump > 4.5 & VelocityBeforeJump > 15)
{
BasinIndex = 1; //Type2
}
else if (FroudeNumberBeforeJump > 4.5 & VelocityBeforeJump <= 15)
{
BasinIndex = 2; //Type3
}
else if (FroudeNumberBeforeJump > 2.5 & FroudeNumberBeforeJump <= 4.5)
{
BasinIndex = 3; //Type4
}
if (BasinIndex == 0)
{
//Type1
StillingBasinLength = (-0.1228 * Math.Pow(FroudeNumberBeforeJump, 2) +
1.4973 * FroudeNumberBeforeJump + 1.7727) * FlowDepthAfterJump;
}
else if (BasinIndex == 1)
{
//Type2
StillingBasinLength = (-0.0006 * Math.Pow(FroudeNumberBeforeJump, 4) +
0.0228 * Math.Pow(FroudeNumberBeforeJump, 3) - 0.3298 *
Math.Pow(FroudeNumberBeforeJump, 2) + 2.1478 * FroudeNumberBeforeJump -
1.1103) * FlowDepthAfterJump;
}
else if (BasinIndex == 2)
{
//Type3
StillingBasinLength = (-0.00003 * Math.Pow(FroudeNumberBeforeJump, 4) +
0.0015 * Math.Pow(FroudeNumberBeforeJump, 3) - 0.0371 *
Math.Pow(FroudeNumberBeforeJump, 2) + 0.4113 * FroudeNumberBeforeJump +
1.0556) * FlowDepthAfterJump;
}
```

```
else if (BasinIndex == 3)

{

//Type4

StillingBasinLength = (-0.185 * Math.Pow(FroudeNumberBeforeJump, 2) +
1.857 * FroudeNumberBeforeJump + 1.338) * FlowDepthAfterJump;

}

return StillingBasinLength;

}

public static double Calculatehp050NoPier(double reldist, double
designhead)

{

double hp050NoPier, relpressure;

if (reldist < 0.20)

{

relpressure = 536.84 * Math.Pow(reldist, 6) - 12.172 * Math.Pow(reldist,
5) - 34.378 * Math.Pow(reldist, 4) - 0.217 * Math.Pow(reldist, 3) + 1.4794
* Math.Pow(reldist, 2) - 0.3638 * reldist + 0.1999;

}

else

{

relpressure = 0.9572 * Math.Pow(reldist, 5) - 3.4197 * Math.Pow(reldist,
4) + 4.3 * Math.Pow(reldist, 3) - 2.193 * Math.Pow(reldist, 2) + 0.2682 *
reldist + 0.1662;

}

hp050NoPier = relpressure * designhead;

return hp050NoPier;

}

public static double Calculatehp100NoPier(double reldist, double
designhead)

{

double hp100NoPier, relpressure;

if (reldist < -0.1)

{
```

```
relpressure = 537.29 * Math.Pow(reldist, 4) + 258 * Math.Pow(reldist, 3) +
43.623 * Math.Pow(reldist, 2) + 3.0826 * (reldist) + 0.0861;

}

else if (reldist >= 0.17)

{

relpressure = 0.6563 * Math.Pow(reldist, 5) - 2.2952 * Math.Pow(reldist,
4) + 2.7051 * Math.Pow(reldist, 3) - 1.1496 * Math.Pow(reldist, 2) +
0.0862 * reldist + 0.0486;

}

else

{

relpressure = 449.2 * Math.Pow(reldist, 5) - 271.05 * Math.Pow(reldist, 4)
+ 22.06 * Math.Pow(reldist, 3) + 4.4105 * Math.Pow(reldist, 2) - 0.1955 *
reldist;

}

hp100NoPier = relpressure * designhead;

return hp100NoPier;

}

public static double Calculatehp133NoPier(double reldist, double
designhead)

{

double hp133NoPier, relpressure;

if (reldist < -0.245)

{

relpressure = 130.92 * Math.Pow(reldist, 2) + 60.483 * Math.Pow(reldist,
1) + 6.5576;

}

else if (reldist >= -0.245 & reldist <= -0.17)

{

relpressure = -Math.Pow(reldist, 1) - 0.645;

}

else if (reldist > -0.17 & reldist <= -0.05)

{
```

```
relpressure = -1.9418 * Math.Pow(reldist, 2) + 0.8433 * Math.Pow(reldist,
1) - 0.2772;

}

else if (reldist >= -0.05 & reldist <= 0.14)

{

relpressure = 23.479 * Math.Pow(reldist, 3) + 4.5254 * Math.Pow(reldist,
2) - 0.1841 * Math.Pow(reldist, 1) - 0.3481;

}

else

{

relpressure = -4.5413 * Math.Pow(reldist, 6) + 21.475 * Math.Pow(reldist,
5) - 39.338 * Math.Pow(reldist, 4) + 35.031 * Math.Pow(reldist, 3) -
15.547 * Math.Pow(reldist, 2) + 3.3403 * Math.Pow(reldist, 1) - 0.465;

}

hp133NoPier = relpressure * designhead;

return hp133NoPier;

}

public static double Calculatehp050PierBay(double reldist, double
designhead)

{

double hp050PierBay, relpressure;

if (reldist <= -0.20)

{

relpressure = 31.037 * Math.Pow(reldist, 2) + 12.078 * Math.Pow(reldist,
1) + 1.5164;

}

else if (reldist > -0.20 & reldist <= 0.5)

{

relpressure = -2.0404 * Math.Pow(reldist, 4) - 0.4116 * Math.Pow(reldist,
3) + 0.9517 * Math.Pow(reldist, 2) - 0.3469 * reldist + 0.2343;

}

else

{
```

```
relpressure = 0.0422 * Math.Pow(reldist, 2) - 0.1392 * Math.Pow(reldist,
1) + 0.177;

}

hp050PierBay = relpressure * designhead;

return hp050PierBay;

}

public static double Calculatehp100PierBay(double reldist, double
designhead)

{

double hp100PierBay, relpressure;

if (reldist <= -0.20)

{

relpressure = 39.738 * Math.Pow(reldist, 2) + 15.569 * Math.Pow(reldist,
1) + 1.7342;

}

else if (reldist > -0.20 & reldist <= 0)

{

relpressure = -30.078 * Math.Pow(reldist, 3) - 10.529 * Math.Pow(reldist,
2) - 1.3526 * Math.Pow(reldist, 1) + 0.12;

}

else if (reldist > 0 & reldist <= 0.5)

{

relpressure = -31.282 * Math.Pow(reldist, 5) + 45.468 * Math.Pow(reldist,
4) - 23.298 * Math.Pow(reldist, 3) + 4.6207 * Math.Pow(reldist, 2) -
0.3448 * Math.Pow(reldist, 1) + 0.1207;

}

else

{

relpressure = -0.0065 * Math.Pow(reldist, 1) + 0.0588;

}

hp100PierBay = relpressure * designhead;

return hp100PierBay;

}
```

```
public static double Calculatehp133PierBay(double reldist, double
designhead)

{

double hp133PierBay, relpressure;

if (reldist <= -0.24)

{

relpressure = -9.625 * reldist - 2.41;

}

else if (reldist > -0.24 & reldist <= -0.20)

{

relpressure = -1.25 * reldist - 0.4;

}

else if (reldist > -0.20 & reldist <= 0)

{

relpressure = -4.5 * Math.Pow(reldist, 2) - 1.1 * reldist - 0.19;

}

else if (reldist > 0 & reldist <= 0.32)

{

relpressure = -2.0455 * Math.Pow(reldist, 2) + 0.9001 * Math.Pow(reldist,
1) - 0.1898;

}

else

{

relpressure = 0.2356 * Math.Pow(reldist, 4) - 1.0135 * Math.Pow(reldist,
3) + 1.5085 * Math.Pow(reldist, 2) - 0.7762 * Math.Pow(reldist, 1) +
0.0146;

}

hp133PierBay = relpressure * designhead;

return hp133PierBay;

}

public static double Calculatehp050AlongPiers(double reldist, double
designhead)

{
```

```
double hp050AlongPiers, relpressure;

if (reldist < -0.15)

{

relpressure = 8.8889 * Math.Pow(reldist, 2) + 1.213 * reldist + 0.2331;

}

else if (reldist >= -0.15 & reldist <= 0.3)

{

relpressure = 17.555 * Math.Pow(reldist, 4) - 4.8743 * Math.Pow(reldist,
3) - 0.6634 * Math.Pow(reldist, 2) - 0.0664 * Math.Pow(reldist, 1) +
0.2292;

}

else

{

relpressure = 0.1185 * Math.Pow(reldist, 2) - 0.2411 * Math.Pow(reldist,
1) + 0.2268;

}

hp050AlongPiers = relpressure * designhead; ;

return hp050AlongPiers;

}

public static double Calculatehp100AlongPiers(double reldist, double
designhead)

{

double hp100AlongPiers, relpressure;

if (reldist < -0.18)

{

relpressure = -27.983 * Math.Pow(reldist, 2) - 20.263 * Math.Pow(reldist,
1) - 2.7446;

}

else if (reldist >= -0.18 & reldist <= -0.13)

{

relpressure = -0.4 * Math.Pow(reldist, 1) - 0.082;

}

else if (reldist >= -0.13 & reldist <= 0.10)
```

```
{
relpressure = 0.6515 * Math.Pow(reldist, 1) + 0.0549;
}
else
{
relpressure = 0.1123 * Math.Pow(reldist, 3) - 0.2344 * Math.Pow(reldist,
2) + 0.1093 * Math.Pow(reldist, 1) + 0.1105;
}
hp100AlongPiers = relpressure * designhead;
return hp100AlongPiers;
}
public static double Calculatehp133AlongPiers(double reldist, double
designhead)
{
double hp133AlongPiers, relpressure;
if (reldist <= -0.20)
{
relpressure = -14.598 * Math.Pow(reldist, 1) - 3.4743;
}
else if (reldist > -0.20 & reldist <= 0)
{
relpressure = 11.043 * Math.Pow(reldist, 2) + 3.7034 * Math.Pow(reldist,
1) - 0.3105;
}
else if (reldist > 0 & reldist <= 0.50)
{
relpressure = -33.333 * Math.Pow(reldist, 5) + 33.333 * Math.Pow(reldist,
4) - 2.5 * Math.Pow(reldist, 3) - 6.5833 * Math.Pow(reldist, 2) + 2.3533 *
Math.Pow(reldist, 1) - 0.31;
}
else
{
relpressure = 0.4272 * Math.Pow(reldist, 3) - 1.1635 * Math.Pow(reldist,
2) + 1.1331 * Math.Pow(reldist, 1) - 0.3806;
```

```
}
hp133AlongPiers = relpressure * designhead;
return hp133AlongPiers;
}
public static double Calculatehp(double headratio, double hp050, double
hp100, double hp133)
{
double hp;
if (headratio >= 0.50 && headratio <= 1.00)
{
hp = hp050 + (hp100 - hp050) / (1 - 0.50) * (headratio - 0.50);
}
else if (headratio > 1.00 && headratio <= 1.33)
{
hp = hp100 + (hp133 - hp100) / (1.33 - 1.00) * (headratio - 1.00);
}
else
{
hp = 0;
}
return hp;
}
```

A.5. Generation Methods

```
private void GenerateCrestGeometry()
{
_dtCrestGeometryUpstream = new DataTable();
_dtCrestGeometryUpstream.Columns.Add("XoverHd", typeof(double));
_dtCrestGeometryUpstream.Columns.Add("X*", typeof(double));
_dtCrestGeometryUpstream.Columns.Add("lnX*", typeof(double));
_dtCrestGeometryUpstream.Columns.Add("Y*", typeof(double));
_dtCrestGeometryUpstream.Columns.Add("CrestStation", typeof(double));
```

```
_dtCrestGeometryUpstream.Columns.Add("CrestElevation", typeof(double));
_dtCrestGeometryUpstream.Columns.Add("Location", typeof(XYPT));
for (int i = 0; i < _CrestGeometryUpstreamNodes; i++)
{
CrestGeometryRow row = new CrestGeometryRow();
row.XoverHd = -0.2818 * i / (_CrestGeometryUpstreamNodes - 1);
row.Xstar = (row.XoverHd + 0.2818000001) * 1.3055;
row.lnXstar = Math.Log(row.Xstar);
row.YStar = -1 * row.lnXstar * row.Xstar;
row.BottomStation = row.XoverHd * _DesignHead;
row.BottomElevation = _Crest.InsPt.y + ((row.YStar / 2.705) - 0.136) *
_DesignHead;
row.Location = new XYPT(Math.Round(row.BottomStation,2),
Math.Round(row.BottomElevation,2));
_dtCrestGeometryUpstream.Rows.Add(new object[] {
Math.Round(row.XoverHd,2),
Math.Round(row.Xstar,2),
Math.Round(row.lnXstar,2),
Math.Round(row.YStar,2),
Math.Round(row.BottomStation,2),
Math.Round(row.BottomElevation,2),
row.Location });
}
_dtCrestGeometryDownstream = new DataTable();
_dtCrestGeometryDownstream.Columns.Add("XoverHd", typeof(double));
_dtCrestGeometryDownstream.Columns.Add("X*", typeof(double));
_dtCrestGeometryDownstream.Columns.Add("lnX*", typeof(double));
_dtCrestGeometryDownstream.Columns.Add("Y*", typeof(double));
_dtCrestGeometryDownstream.Columns.Add("CrestStation", typeof(double));
_dtCrestGeometryDownstream.Columns.Add("CrestElevation", typeof(double));
_dtCrestGeometryDownstream.Columns.Add("Location", typeof(XYPT));
_dtCrestGeometryDownstream.Columns.Add("Dist2PrevNode", typeof(double));
```

```
_dtCrestGeometryDownstream.Columns.Add("StreamwiseDist", typeof(double));

double craegerk = 0.5;

double craegern = 1.85;

double Xpt = Math.Pow(Math.Tan(_ChuteAngleRadian) / craegerk / craegern, 1
/ (craegern - 1)) * _DesignHead;

double Ypt = _Crest.InsPt.y + Math.Pow(Xpt / _DesignHead, craegern) *
craegerk * _DesignHead * (-1);

_Crest.PointOfTangency = new XYPT(Xpt, Ypt);

for (int i = 0; i < _CrestGeometryDownstreamNodes; i++)

{

CrestGeometryRow row = new CrestGeometryRow();

row.XoverHd = Xpt / _DesignHead / _CrestGeometryDownstreamSegment * i;

row.Xstar = (row.XoverHd + 0.2818000001) * 1.3055;

row.lnXstar = Math.Log(row.Xstar);

row.YStar = -1 * row.lnXstar * row.Xstar;

row.BottomStation = _Crest.PointOfTangency.x /
_CrestGeometryDownstreamSegment * i;

row.BottomElevation = _Crest.InsPt.y - Math.Pow(row.BottomStation /
_DesignHead, craegern) * craegerk * _DesignHead;

row.Location = new XYPT(Math.Round(row.BottomStation,2),
Math.Round(row.BottomElevation,2));

if (i == 0)

{

row.DistanceToPreviousNode = 0;

row.StreamwiseDistance = 0;

}

else

{

double prevx = Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i -
1][4]);

double prevz = Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i -
1][5]);

double prevStreamwiseDistance =
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i - 1][8]);
```

```
row.DistanceToPreviousNode = Math.Sqrt(Math.Pow(row.BottomStation - prevx,
2) + Math.Pow(row.BottomElevation - prevz, 2));

row.StreamwiseDistance = prevStreamwiseDistance +
row.DistanceToPreviousNode;

}

_dtCrestGeometryDownstream.Rows.Add(new object[] {

Math.Round(row.XoverHd,2),

Math.Round(row.Xstar,2),

Math.Round(row.lnXstar,2),

Math.Round(row.YStar,2),

Math.Round(row.BottomStation,2),

Math.Round(row.BottomElevation,2),

row.Location,

Math.Round(row.DistanceToPreviousNode,2),

Math.Round(row.StreamwiseDistance,2), });

}

}

private void GenerateCrestWaterSurfaceProfiles()

{

_dtCrestUSWSP = new DataTable();

_dtCrestUSWSP.Columns.Add("X", typeof(double));

_dtCrestUSWSP.Columns.Add("z", typeof(double));

_dtCrestUSWSP.Columns.Add("S*design", typeof(double));

_dtCrestUSWSP.Columns.Add("S*existing", typeof(double));

_dtCrestUSWSP.Columns.Add("s-design", typeof(double));

_dtCrestUSWSP.Columns.Add("s-existing", typeof(double));

_dtCrestUSWSP.Columns.Add("slope", typeof(double));

_dtCrestUSWSP.Columns.Add("vel. Design", typeof(double));

_dtCrestUSWSP.Columns.Add("vel. Existing", typeof(double));

_dtCrestUSWSP.Columns.Add("hp Design", typeof(double));

_dtCrestUSWSP.Columns.Add("hp Existing", typeof(double));

_dtCrestDSWSP = new DataTable();
```

```
_dtCrestDSWSP.Columns.Add("X", typeof(double));
_dtCrestDSWSP.Columns.Add("z", typeof(double));
_dtCrestDSWSP.Columns.Add("S*design", typeof(double));
_dtCrestDSWSP.Columns.Add("S*existing", typeof(double));
_dtCrestDSWSP.Columns.Add("s-design", typeof(double));
_dtCrestDSWSP.Columns.Add("s-existing", typeof(double));
_dtCrestDSWSP.Columns.Add("slope", typeof(double));
_dtCrestDSWSP.Columns.Add("vel. Design", typeof(double));
_dtCrestDSWSP.Columns.Add("vel. Existing", typeof(double));
_dtCrestDSWSP.Columns.Add("hp Design", typeof(double));
_dtCrestDSWSP.Columns.Add("hp Existing", typeof(double));
_plWSPDesignUpstream = new Polyline();
_plWSPDesignDownstream = new Polyline();
_plWSPExistingUpstream = new Polyline();
_plWSPExistingDownstream = new Polyline();
PolylineVertex wspvertexDesign = new PolylineVertex();
PolylineVertex wspvertexExisting = new PolylineVertex();
double DesignHeadRatio = 1;
double ExistingHeadRatio = _ExistingHead / _DesignHead;
for (int i = 0; i < _dtCrestGeometryUpstream.Rows.Count; i++)
{
CrestWSPRow row = new CrestWSPRow();
row.Station = Convert.ToDouble(_dtCrestGeometryUpstream.Rows[i][4]);
row.BottomElevation =
Convert.ToDouble(_dtCrestGeometryUpstream.Rows[i][5]);
row.SstarDesign = (Math.Pow(DesignHeadRatio, 1.1) -
(Convert.ToDouble(_dtCrestGeometryUpstream.Rows[i][0]) / 6)) * 0.75;
row.SstarExisting = (Math.Pow(ExistingHeadRatio, 1.1) -
(Convert.ToDouble(_dtCrestGeometryUpstream.Rows[i][0]) / 6)) * 0.75;
row.VerticalHeightDesign = row.SstarDesign * _DesignHead;
row.VerticalHeightExisting = row.SstarExisting * _DesignHead;
row.Slope = 0;
```

```
row.VelocityDesign = _DesignDischarge / (_SpillwayRoughLength) /
(row.VerticalHeightDesign * Math.Cos(row.Slope));

row.VelocityExisting = _ExistingDischarge / (_SpillwayRoughLength) /
(row.VerticalHeightExisting * Math.Cos(row.Slope));

row.PressureHeadDesign = Convert.ToDouble(_dtPressureNoPier.Rows[20-
i][2]);

row.PressureHeadExisting = Convert.ToDouble(_dtPressureNoPier.Rows[20-
i][6]);

row.TotalHeadDesign = row.BottomElevation + row.VerticalHeightDesign +
Math.Pow(row.VelocityDesign, 2) / 19.62;

row.TotalHeadExisting = row.BottomElevation + row.VerticalHeightExisting +
Math.Pow(row.VelocityExisting, 2) / 19.62;

_dtCrestUSWSP.Rows.Add(new object[] {

Math.Round(row.Station,2),

Math.Round(row.BottomElevation,2),

Math.Round(row.SstarDesign,2),

Math.Round(row.SstarExisting,2),

Math.Round(row.VerticalHeightDesign,2),

Math.Round(row.VerticalHeightExisting,2),

Math.Round(row.Slope,2),

Math.Round(row.VelocityDesign,2),

Math.Round(row.VelocityExisting,2),

Math.Round(row.PressureHeadDesign,2),

Math.Round(row.PressureHeadExisting,2) });

wspvertexDesign = new PolylineVertex(new XYPT(row.Station, _Crest.InsPt.y
+ row.VerticalHeightDesign));

_plWSPDesignUpstream.Vertexes.Add(wspvertexDesign);

wspvertexExisting = new PolylineVertex(new XYPT(row.Station,
_Crest.InsPt.y + row.VerticalHeightExisting));

_plWSPExistingUpstream.Vertexes.Add(wspvertexExisting);

}

for (int i = 0; i < _dtCrestGeometryDownstream.Rows.Count; i++)

{

CrestWSPRow row = new CrestWSPRow();
```

```
row.Station = Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][4]);

row.BottomElevation =
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][5]);

row.SstarDesign = (Math.Pow(DesignHeadRatio, 1.1) -
(Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][0]) / 6)) * 0.75;

row.SstarExisting = (Math.Pow(ExistingHeadRatio, 1.1) -
(Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][0]) / 6)) * 0.75;

row.VerticalHeightDesign = row.SstarDesign * _DesignHead;

row.VerticalHeightExisting = row.SstarExisting * _DesignHead;

row.Slope = Math.Atan((Math.Pow(row.Station / _DesignHead, 0.85) * 1.85 *
0.5));

row.VelocityDesign = _DesignDischarge / (_SpillwayRoughLength) /
(row.VerticalHeightDesign * Math.Cos(row.Slope));

row.VelocityExisting = _ExistingDischarge / (_SpillwayRoughLength) /
(row.VerticalHeightExisting * Math.Cos(row.Slope));

row.PressureHeadDesign =
Convert.ToDouble(_dtPressureNoPier.Rows[20+i][2]);

row.PressureHeadExisting =
Convert.ToDouble(_dtPressureNoPier.Rows[20+i][6]);

row.TotalHeadDesign = row.BottomElevation + row.VerticalHeightDesign *
Math.Pow(Math.Cos(_ChuteAngleRadian), 2) + Math.Pow(row.VelocityDesign, 2)
/ 19.62;

row.TotalHeadExisting = row.BottomElevation + row.VerticalHeightExisting *
Math.Pow(Math.Cos(_ChuteAngleRadian), 2) + Math.Pow(row.VelocityExisting,
2) / 19.62;

_dtCrestDSWSP.Rows.Add(new object[] {

Math.Round(row.Station,2),

Math.Round(row.BottomElevation,2),

Math.Round(row.SstarDesign,2),

Math.Round(row.SstarExisting,2),

Math.Round(row.VerticalHeightDesign,2),

Math.Round(row.VerticalHeightExisting,2),

Math.Round(row.Slope,2),

Math.Round(row.VelocityDesign,2),

Math.Round(row.VelocityExisting,2),

Math.Round(row.PressureHeadDesign,2),
```

```
Math.Round(row.PressureHeadExisting,2)});

wspvertexDesign = new PolylineVertex(new XYPT(row.Station,
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][5]) +
row.VerticalHeightDesign));

_plWSPDesignDownstream.Vertexes.Add(wspvertexDesign);

wspvertexExisting = new PolylineVertex(new XYPT(row.Station,
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][5]) +
row.VerticalHeightExisting));

_plWSPExistingDownstream.Vertexes.Add(wspvertexExisting);

}

_InitialFlowDepthDesign =
Convert.ToDouble(_dtCrestDSWSP.Rows[_dtCrestDSWSP.Rows.Count - 1][4]);

_InitialFlowDepthExisting =
Convert.ToDouble(_dtCrestDSWSP.Rows[_dtCrestDSWSP.Rows.Count - 1][5]);

}

private void GenerateWaterwayGeometry(double StillingBasinElevation)

{

switch (cmbWaterwayType.SelectedIndex)

{

case 0: //chute

_dtChuteGeometry = new DataTable();

_dtChuteGeometry.Columns.Add("ChuteStation", typeof(double));

_dtChuteGeometry.Columns.Add("ChuteElevation", typeof(double));

_dtChuteGeometry.Columns.Add("Location", typeof(XYPT));

_dtChuteGeometry.Columns.Add("Dist2PrevNode", typeof(double));

_dtChuteGeometry.Columns.Add("StreamwiseDistance", typeof(double));

_WaterwayHeight = _Crest.PointOfTangency.y - StillingBasinElevation;

_WaterwayHorizontalLength = _WaterwayHeight / Math.Tan(_ChuteAngleRadian);

_ChuteCalculationStepNumber = 20;

_ChuteHorizontalCalculationInterval = _WaterwayHorizontalLength /
_ChuteCalculationStepNumber;

_ChuteVerticalCalculationInterval = _WaterwayHeight /
_ChuteCalculationStepNumber;

for (int i = 0; i <= _ChuteCalculationStepNumber; i++)
```

```
{
WaterwayGeometryRow row = new WaterwayGeometryRow();
row.BottomStation = _Crest.PointOfTangency.x + i *
_ChuteHorizontalCalculationInterval;
row.BottomElevation = _Crest.PointOfTangency.y - i *
_ChuteVerticalCalculationInterval;
row.Location = new XYPT(Math.Round(row.BottomStation,2),
Math.Round(row.BottomElevation,2));
if (i == 0)
{
row.DistanceToPreviousNode = 0;
row.StreamwiseDistance =
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[Convert.ToInt32(_CrestGeo
metryDownstreamNodes) - 1][8]);
}
else
{
double prevx = Convert.ToDouble(_dtChuteGeometry.Rows[i - 1][0]);
double prevz = Convert.ToDouble(_dtChuteGeometry.Rows[i - 1][1]);
double prevStreamwiseDistance = Convert.ToDouble(_dtChuteGeometry.Rows[i -
1][4]);
row.DistanceToPreviousNode = Math.Sqrt(Math.Pow(row.BottomStation - prevx,
2) + Math.Pow(row.BottomElevation - prevz, 2));
row.StreamwiseDistance = prevStreamwiseDistance +
row.DistanceToPreviousNode;
}
_dtChuteGeometry.Rows.Add(new object[] {
Math.Round(row.BottomStation,2),
Math.Round(row.BottomElevation,2),
row.Location,
Math.Round(row.DistanceToPreviousNode,2),
Math.Round(row.StreamwiseDistance,2) });
}
_plWaterwayBottomProfile = new Polyline();
```

```
_plWaterwayBottomProfile.Vertexes.Add(new
PolylineVertex(_Crest.PointOfTangency));

_plWaterwayBottomProfile.Vertexes.Add(new
PolylineVertex(_Crest.PointOfTangency + new
XYPT(_WaterwayHorizontalLength, -1 * _WaterwayHeight)));

break;

case 1: //cascade

int cascadestepnumber =
Convert.ToInt32(Math.Ceiling((_Crest.PointOfTangency.y -
StillingBasinElevation) / _StepHeight));

_plWaterwayBottomProfile = new Polyline();

_plWaterwayBottomProfile.Vertexes.Add(new
PolylineVertex(_Crest.PointOfTangency));

for (int i = 1; i < cascadestepnumber; i++)

{

_plWaterwayBottomProfile.Vertexes.Add(new PolylineVertex(new
XYPT(_plWaterwayBottomProfile.Vertexes[0].Point.x + (i - 1) * _StepLength,
_plWaterwayBottomProfile.Vertexes[0].Point.y - i * _StepHeight)));

_plWaterwayBottomProfile.Vertexes.Add(new PolylineVertex(new
XYPT(_plWaterwayBottomProfile.Vertexes[0].Point.x + i * _StepLength,
_plWaterwayBottomProfile.Vertexes[0].Point.y - (i) * _StepHeight)));

}

_plWaterwayBottomProfile.Vertexes.Add(new PolylineVertex(new
XYPT(_plWaterwayBottomProfile.Vertexes[2 * (cascadestepnumber - 1) -
1].Point.x + _StepLength, _plWaterwayBottomProfile.Vertexes[2 *
(cascadestepnumber - 1)].Point.y)));

_plWaterwayBottomProfile.Vertexes.Add(new PolylineVertex(new
XYPT(_plWaterwayBottomProfile.Vertexes[2 * cascadestepnumber - 1].Point.x,
_StillingBasinElevation)));

_plPseudoBottom = new Polyline();

_plPseudoBottom.OverrideColor = new PBUI.Colour.PDColor(Color.Gray);

_plPseudoBottom.Vertexes.Add(_plWaterwayBottomProfile.Vertexes[0]);

_plPseudoBottom.Vertexes.Add(new PolylineVertex(new
XYPT(_plPseudoBottom.Vertexes[0].Point + new XYPT(_StepLength *
(cascadestepnumber-1), (-1) * _StepHeight * (cascadestepnumber-1)))));

break;

}

}
```

```
private Cascade GenerateCascadeProfile(double Discharge, double
StillingBasinElevation)

{

Cascade _Cascade = new Cascade();

_Cascade.UnitDischarge = Discharge / _SpillwayRoughLength;

_Cascade.CascadeAngleRadian = Math.Atan(_StepHeight / _StepLength);

_Cascade.CascadeAngle = _Cascade.CascadeAngleRadian * 180 / Math.PI;

_Cascade.dcOnset = (1.057 - 0.465 * _StepHeight / _StepLength) *
_StepHeight;

_Cascade.dc = Math.Pow(Math.Pow(_Cascade.UnitDischarge, 2) / 9.81,
0.3333);

_Cascade.ks = _StepLength * Math.Sin(_Cascade.CascadeAngleRadian);

_Cascade.Fstar = _Cascade.UnitDischarge / Math.Pow(9.81 *
Math.Sin(_Cascade.CascadeAngleRadian) * Math.Pow(_Cascade.ks, 3), 0.5);

_Cascade.Li = 9.72 * Math.Pow(Math.Sin(_Cascade.CascadeAngleRadian), 0.08)
* Math.Pow(_Cascade.Fstar, 0.71) * _Cascade.ks;

_PTStreamWiseDistance =
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[_dtCrestGeometryDownstrea
m.Rows.Count - 1][8]);

_Cascade.Lc = _PTStreamWiseDistance + (_Crest.PointOfTangency.y -
_StillingBasinElevation) / Math.Sin(_Cascade.CascadeAngleRadian);

_Cascade.di = Math.Pow(_Cascade.Fstar, 0.592) * 0.4034 /
Math.Pow(Math.Sin(_Cascade.CascadeAngleRadian), 0.04) * _Cascade.ks;

_Cascade.fe = 0.25;

double dd = _Cascade.dc * Math.Pow(_Cascade.fe / 8 /
Math.Sin(_Cascade.CascadeAngleRadian), 0.3333); //control

for (_Cascade.UniformDepth = 0.001; _Cascade.ErrorTerm >= 0;
_Cascade.UniformDepth += 0.001)

{

_Cascade.Area = _Cascade.UniformDepth * _SpillwayRoughLength;

_Cascade.WettedPerimeter = _Cascade.UniformDepth * 2 +
_SpillwayRoughLength;

_Cascade.HydraulicRadius = _Cascade.Area / _Cascade.WettedPerimeter;

_Cascade.HydraulicDepth =_Cascade.UniformDepth * 4;

_Cascade.ErrorTerm = (_Cascade.UnitDischarge / _Cascade.UniformDepth) -
Math.Pow(8 * 9.81 * Math.Sin(_Cascade.CascadeAngleRadian) *
_Cascade.HydraulicDepth / _Cascade.fe / 4, 0.5);
```

```
}
_Cascade.Velocity = _Cascade.UnitDischarge / _Cascade.UniformDepth;
_Cascade.TotalHeight = _Crest.PointOfTangency.y - _StillingBasinElevation;
if (_Cascade.Lc >= _Cascade.Li)
{
_Cascade.IsFlowFullyAerated = true;
}
else
{
_Cascade.IsFlowFullyAerated = false;
}
return _Cascade;
}
private void FixCascadeDepth(double initialflowdepthDesign, double
initialflowdepthExisting)
{
if (_CascadeDesign.IsFlowFullyAerated == false)
{
double fixeddepthDesign = initialflowdepthDesign - ((-
_CascadeDesign.UniformDepth + initialflowdepthDesign) / _CascadeDesign.Li
* _CascadeDesign.Lc);
_CascadeDesign.UniformDepth = fixeddepthDesign;
}
if (_CascadeExisting.IsFlowFullyAerated == false)
{
double fixeddepthExisting = initialflowdepthExisting - ((-
_CascadeExisting.UniformDepth + initialflowdepthExisting) /
_CascadeExisting.Li * _CascadeExisting.Lc);
_CascadeExisting.UniformDepth = fixeddepthExisting;
}
}
private void GenerateWaterwayWaterSurfaceProfiles(double
initialFlowDepthDesign, double initialFlowDepthExisting)
{
```

```
if (cmbWaterwayType.SelectedIndex == 0)
{
_dtChuteWSPDesign = new DataTable();
_dtChuteWSPExisting = new DataTable();
_plWSPDesignStandardStep = new Polyline();
_plWSPExistingStandardStep = new Polyline();

_dtChuteWSPDesign.Rows.Clear();
_dtChuteWSPDesign.Columns.Add("Index", typeof(int));
_dtChuteWSPDesign.Columns.Add("ZCoord", typeof(double));
_dtChuteWSPDesign.Columns.Add("FlowDepth", typeof(double));
_dtChuteWSPDesign.Columns.Add("Area", typeof(double));
_dtChuteWSPDesign.Columns.Add("WettedPerimeter", typeof(double));
_dtChuteWSPDesign.Columns.Add("HydraulicRadius", typeof(double));
_dtChuteWSPDesign.Columns.Add("Velocity", typeof(double));
_dtChuteWSPDesign.Columns.Add("FroudeNumber", typeof(double));
_dtChuteWSPDesign.Columns.Add("FrictionSlope", typeof(double));
_dtChuteWSPDesign.Columns.Add("TotalHead", typeof(double));
_dtChuteWSPDesign.Columns.Add("AveragedFrictionSlope", typeof(double));
_dtChuteWSPDesign.Columns.Add("PreviousTotalHead", typeof(double));
_dtChuteWSPDesign.Columns.Add("HydraulicLoss", typeof(double));
_dtChuteWSPDesign.Columns.Add("ErrorTerm", typeof(double));

_dtChuteWSPExisting.Rows.Clear();
_dtChuteWSPExisting.Columns.Add("Index", typeof(int));
_dtChuteWSPExisting.Columns.Add("ZCoord", typeof(double));
_dtChuteWSPExisting.Columns.Add("FlowDepth", typeof(double));
_dtChuteWSPExisting.Columns.Add("Area", typeof(double));
_dtChuteWSPExisting.Columns.Add("WettedPerimeter", typeof(double));
_dtChuteWSPExisting.Columns.Add("HydraulicRadius", typeof(double));
_dtChuteWSPExisting.Columns.Add("Velocity", typeof(double));
```

```
_dtChuteWSPExisting.Columns.Add("FroudeNumber", typeof(double));

_dtChuteWSPExisting.Columns.Add("FrictionSlope", typeof(double));

_dtChuteWSPExisting.Columns.Add("TotalHead", typeof(double));

_dtChuteWSPExisting.Columns.Add("AveragedFrictionSlope", typeof(double));

_dtChuteWSPExisting.Columns.Add("PreviousTotalHead", typeof(double));

_dtChuteWSPExisting.Columns.Add("HydraulicLoss", typeof(double));

_dtChuteWSPExisting.Columns.Add("ErrorTerm", typeof(double));

_ChuteWSPDesign = new WSPTable();

_ChuteWSPExisting = new WSPTable();

WSPTableRow initialrowdesign =
_HydraulicSolver.StandardStepMethodInitialRowCalculations(initialFlowDepth
Design, _ChuteAngleDegree, _Crest.PointOfTangency.y, _SpillwayRoughLength,
_DesignDischarge, _Manning);

_ChuteWSPDesign.RowList.Add(initialrowdesign);

WSPTableRow initialrowExisting =
_HydraulicSolver.StandardStepMethodInitialRowCalculations(initialFlowDepth
Existing, _ChuteAngleDegree, _Crest.PointOfTangency.y,
_SpillwayRoughLength, _ExistingDischarge, _Manning);

_ChuteWSPExisting.RowList.Add(initialrowExisting);

for (int i = 1; i < _ChuteCalculationStepNumber + 1; i++)

{

WSPTableRow rowdesign = new WSPTableRow();

WSPTableRow rowExisting = new WSPTableRow();

WSPTableRow previousrowdesign = _ChuteWSPDesign.RowList[i - 1];

WSPTableRow previousrowExisting = _ChuteWSPExisting.RowList[i - 1];

rowdesign =
_HydraulicSolver.StandardStepMethodRowCalculations(previousrowdesign,
_Crest.PointOfTangency.y, _ChuteHorizontalCalculationInterval,
previousrowdesign.FlowDepth, _ChuteAngleDegree, _DesignDischarge,
_Manning, _SpillwayRoughLength);

rowExisting =
_HydraulicSolver.StandardStepMethodRowCalculations(previousrowExisting,
_Crest.PointOfTangency.y, _ChuteHorizontalCalculationInterval,
previousrowExisting.FlowDepth, _ChuteAngleDegree, _ExistingDischarge,
_Manning, _SpillwayRoughLength);

for (rowdesign.FlowDepth = previousrowdesign.FlowDepth;
rowdesign.ErrorTerm > 0; rowdesign.FlowDepth -= 0.0001)
```

```
{
rowdesign =
_HydraulicSolver.StandardStepMethodRowCalculations(previousrowdesign,
_Crest.PointOfTangency.y, _ChuteHorizontalCalculationInterval,
rowdesign.FlowDepth, _ChuteAngleDegree, _DesignDischarge, _Manning,
_SpillwayRoughLength);
}
_ChuteWSPDesign.RowList.Add(rowdesign);
for (rowExisting.FlowDepth = previousrowExisting.FlowDepth;
rowExisting.ErrorTerm > 0; rowExisting.FlowDepth -= 0.0001)
{
rowExisting =
_HydraulicSolver.StandardStepMethodRowCalculations(previousrowExisting,
_Crest.PointOfTangency.y, _ChuteHorizontalCalculationInterval,
rowExisting.FlowDepth, _ChuteAngleDegree, _ExistingDischarge, _Manning,
_SpillwayRoughLength);
}
_ChuteWSPExisting.RowList.Add(rowExisting);
}
foreach (WSPTableRow item in _ChuteWSPDesign.RowList)
{
_dtChuteWSPDesign.Rows.Add(new object[] { item.Index,
Math.Round(item.ZCoord, 3), Math.Round(item.FlowDepth, 3),
Math.Round(item.Area, 3), Math.Round(item.WettedPerimeter, 3),
Math.Round(item.HydraulicRadius, 3), Math.Round(item.Velocity, 3),
Math.Round(item.FroudeNumber, 3), Math.Round(item.FrictionSlope, 3),
Math.Round(item.TotalHead, 3),
Math.Round(item.AveragedFrictionSlopeWithPreviousSection, 3),
Math.Round(item.PreviousTotalHead, 3), Math.Round(item.HydraulicLoss, 3),
Math.Round(item.ErrorTerm, 3) });
_plWSPDesignStandardStep.Vertexes.Add(new PolylineVertex(new
XYPT(_Crest.PointOfTangency.x + item.Index *
_ChuteHorizontalCalculationInterval + item.FlowDepth *
Math.Sin(_ChuteAngleRadian), item.ZCoord + item.FlowDepth *
Math.Cos(_ChuteAngleRadian))));
}
foreach (WSPTableRow item in _ChuteWSPExisting.RowList)
{
_dtChuteWSPExisting.Rows.Add(new object[] { item.Index,
Math.Round(item.ZCoord, 3), Math.Round(item.FlowDepth, 3),
Math.Round(item.Area, 3), Math.Round(item.WettedPerimeter, 3),
```

```
Math.Round(item.HydraulicRadius, 3), Math.Round(item.Velocity, 3),
Math.Round(item.FroudeNumber, 3), Math.Round(item.FrictionSlope, 3),
Math.Round(item.TotalHead, 3),
Math.Round(item.AveragedFrictionSlopeWithPreviousSection, 3),
Math.Round(item.PreviousTotalHead, 3), Math.Round(item.HydraulicLoss, 3),
Math.Round(item.ErrorTerm, 3) });

_plWSPExistingStandardStep.Vertexes.Add(new PolylineVertex(new
XYPT(_Crest.PointOfTangency.x + item.Index *
_ChuteHorizontalCalculationInterval + item.FlowDepth *
Math.Sin(_ChuteAngleRadian), item.ZCoord + item.FlowDepth *
Math.Cos(_ChuteAngleRadian))));

}

} //chute

else // cascade

{

_CascadeDesign = GenerateCascadeProfile(_DesignDischarge,
_StillingBasinElevation);

_CascadeExisting = GenerateCascadeProfile(_ExistingDischarge,
_StillingBasinElevation);

double cascadeangleradian = Math.Atan(_StepHeight / _StepLength);

FixCascadeDepth(_InitialFlowDepthDesign * Math.Cos(cascadeangleradian),
_InitialFlowDepthExisting * Math.Cos(cascadeangleradian));

_dtCascade = new DataTable();

_dtCascade.Rows.Clear(); _dtCascade = new DataTable();

_dtCascade.Rows.Clear();

_dtCascade.Columns.Add("Discharge", typeof(string));

_dtCascade.Columns.Add("Unit Discharge", typeof(double));

_dtCascade.Columns.Add("Dc Onset", typeof(double));

_dtCascade.Columns.Add("Dc", typeof(double));

_dtCascade.Columns.Add("ks", typeof(double));

_dtCascade.Columns.Add("F*", typeof(double));

_dtCascade.Columns.Add("Li", typeof(double));

_dtCascade.Columns.Add("Lc", typeof(double));

_dtCascade.Columns.Add("di", typeof(double));

_dtCascade.Columns.Add("d0", typeof(double));

_dtCascade.Rows.Add(new object[] { "Design Discharge",
```

```
Math.Round(_CascadeDesign.UnitDischarge, 2),
Math.Round(_CascadeDesign.dcOnset, 2),
Math.Round(_CascadeDesign.dc, 2),
Math.Round(_CascadeDesign.ks, 2),
Math.Round(_CascadeDesign.Fstar, 2),
Math.Round(_CascadeDesign.Li,2),
Math.Round(_CascadeDesign.Lc,2),
Math.Round(_CascadeDesign.di,2),
Math.Round(_CascadeDesign.UniformDepth,2)
});

_dtCascade.Rows.Add(new object[] { "Existing Discharge",
Math.Round(_CascadeExisting.UnitDischarge, 2),
Math.Round(_CascadeExisting.dcOnset, 2),
Math.Round(_CascadeExisting.dc, 2),
Math.Round(_CascadeExisting.ks, 2),
Math.Round(_CascadeExisting.Fstar, 2),
Math.Round(_CascadeExisting.Li,2),
Math.Round(_CascadeExisting.Lc,2),
Math.Round(_CascadeExisting.di,2),
Math.Round(_CascadeExisting.UniformDepth,2)
});
_plCascadeDesignWSP = new Polyline();
_plCascadeExistingWSP = new Polyline();
_plCascadeDesignWSP.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBottomProfile.Vertexes[_plDownstreamBottomProfile.Vertex
es.Count - 1].Point.x,
_plDownstreamBottomProfile.Vertexes[_plDownstreamBottomProfile.Vertexes.Co
unt-1].Point.y + _InitialFlowDepthDesign)));
_plCascadeExistingWSP.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBottomProfile.Vertexes[_plDownstreamBottomProfile.Vertex
es.Count - 1].Point.x,
_plDownstreamBottomProfile.Vertexes[_plDownstreamBottomProfile.Vertexes.Co
unt - 1].Point.y + _InitialFlowDepthExisting)));
```

```
_plCascadeDesignWSP.Vertexes.Add(new PolylineVertex(new
XYPT(_plPseudoBottom.Vertexes[_plPseudoBottom.Vertexes.Count-1].Point.x +
_CascadeDesign.UniformDepth * Math.Sin(_CascadeDesign.CascadeAngleRadian),
_plPseudoBottom.Vertexes[_plPseudoBottom.Vertexes.Count - 1].Point.y +
_CascadeDesign.UniformDepth *
Math.Cos(_CascadeDesign.CascadeAngleRadian))));

_plCascadeExistingWSP.Vertexes.Add(new PolylineVertex(new
XYPT(_plPseudoBottom.Vertexes[_plPseudoBottom.Vertexes.Count - 1].Point.x
+ _CascadeExisting.UniformDepth *
Math.Sin(_CascadeExisting.CascadeAngleRadian),
_plPseudoBottom.Vertexes[_plPseudoBottom.Vertexes.Count - 1].Point.y +
_CascadeExisting.UniformDepth *
Math.Cos(_CascadeExisting.CascadeAngleRadian))));

}

}

private void GenerateLinkBetweenCrestAndChute()

{

_linkDesign = new Polyline();

_linkExisting = new Polyline();

_linkDesign.Vertexes.Add(new PolylineVertex(_Crest.PointOfTangency + new
XYPT(0, _InitialFlowDepthDesign)));

_linkDesign.Vertexes.Add(new
PolylineVertex(_plWSPDesignStandardStep.Vertexes[0].Point));

_linkExisting.Vertexes.Add(new PolylineVertex(_Crest.PointOfTangency + new
XYPT(0, _InitialFlowDepthExisting)));

_linkExisting.Vertexes.Add(new
PolylineVertex(_plWSPExistingStandardStep.Vertexes[0].Point));

}

private void AeratedChuteCalculations()

{

if (chkbxDesignDischarge.Checked)

{

_dtAeratedChuteDesign = new DataTable();

_dtAeratedChuteDesign.Columns.Add("CrestStation", typeof(double));

_dtAeratedChuteDesign.Columns.Add("StreamwiseDistance", typeof(double));

_dtAeratedChuteDesign.Columns.Add("CrestElevation", typeof(double));

_dtAeratedChuteDesign.Columns.Add("NormalizedStreamwiseDistance",
typeof(double));
```

```
_dtAeratedChuteDesign.Columns.Add("AeratedWaterDepth", typeof(double));

_dtAeratedChuteDesign.Columns.Add("BoundaryLayerThickness",
typeof(double));

_dtAeratedChuteDesign.Columns.Add("AngleRadian", typeof(double));

_dtAeratedChuteDesign.Columns.Add("Relative Head", typeof(double));

for (int i = 0; i < _dtCrestGeometryDownstream.Rows.Count; i++)

{

AeratedChuteRow row = new AeratedChuteRow();

row.HorizontalStation =
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][4]);

row.StreamwiseDistance =
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][8]);

row.BottomElevation =
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][5]);

row.NormalizedStreamwiseDistance =
_HydraulicSolver.CalculateNormalizedStreamwiseDistance(row.StreamwiseDista
nce, _SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _DesignDischarge);

row.AeratedWaterDepth =
_HydraulicSolver.CalculateAeratedWaterDepth(row.NormalizedStreamwiseDistan
ce, _SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _DesignDischarge);

row.BoundaryLayerThickness =
_HydraulicSolver.CalculateBoundaryLayerThickness(row.StreamwiseDistance,
_SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _DesignDischarge);

row.AngleRadian =
_HydraulicSolver.CalculateAngleOfPoint(row.HorizontalStation,
_DesignHead);

row.RelativeHead = row.AeratedWaterDepth * Math.Cos(row.AngleRadian) +
Math.Pow((_DesignDischarge / _SpillwayRoughLength /
row.AeratedWaterDepth), 2) / 19.62;

_dtAeratedChuteDesign.Rows.Add(new object[] {

Math.Round(row.HorizontalStation,2),

Math.Round(row.StreamwiseDistance,2),

Math.Round(row.BottomElevation,2),

Math.Round(row.NormalizedStreamwiseDistance,2),

Math.Round(row.AeratedWaterDepth,2),

Math.Round(row.BoundaryLayerThickness,2),

Math.Round(row.AngleRadian,2),
```

```
Math.Round(row.RelativeHead,2) });

}

for (int i = 0; i < _dtChuteGeometry.Rows.Count; i++)

{

AeratedChuteRow row = new AeratedChuteRow();

row.HorizontalStation = Convert.ToDouble(_dtChuteGeometry.Rows[i][0]);

row.StreamwiseDistance = Convert.ToDouble(_dtChuteGeometry.Rows[i][4]);

row.BottomElevation = Convert.ToDouble(_dtChuteGeometry.Rows[i][1]);

row.NormalizedStreamwiseDistance =
_HydraulicSolver.CalculateNormalizedStreamwiseDistance(row.StreamwiseDista
nce, _SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _DesignDischarge);

row.AeratedWaterDepth =
_HydraulicSolver.CalculateAeratedWaterDepth(row.NormalizedStreamwiseDistan
ce, _SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _DesignDischarge);

row.BoundaryLayerThickness =
_HydraulicSolver.CalculateBoundaryLayerThickness(row.StreamwiseDistance,
_SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _DesignDischarge);

row.AngleRadian = _ChuteAngleRadian;

row.RelativeHead = row.AeratedWaterDepth * Math.Cos(row.AngleRadian) +
Math.Pow((_DesignDischarge / _SpillwayRoughLength /
row.AeratedWaterDepth), 2) / 19.62;

_dtAeratedChuteDesign.Rows.Add(new object[] {

Math.Round(row.HorizontalStation,2),

Math.Round(row.StreamwiseDistance,2),

Math.Round(row.BottomElevation,2),

Math.Round(row.NormalizedStreamwiseDistance,2),

Math.Round(row.AeratedWaterDepth,2),

Math.Round(row.BoundaryLayerThickness,2),

Math.Round(row.AngleRadian,2),

Math.Round(row.RelativeHead,2) });

}

}

if (chkbxExistingDischarge.Checked)

{
```

```
_dtAeratedChuteExisting = new DataTable();

_dtAeratedChuteExisting.Columns.Add("CrestStation", typeof(double));

_dtAeratedChuteExisting.Columns.Add("StreamwiseDistance", typeof(double));

_dtAeratedChuteExisting.Columns.Add("CrestElevation", typeof(double));

_dtAeratedChuteExisting.Columns.Add("NormalizedStreamwiseDistance",
typeof(double));

_dtAeratedChuteExisting.Columns.Add("AeratedWaterDepth", typeof(double));

_dtAeratedChuteExisting.Columns.Add("BoundaryLayerThickness",
typeof(double));

_dtAeratedChuteExisting.Columns.Add("AngleRadian", typeof(double));

_dtAeratedChuteExisting.Columns.Add("Relative Head", typeof(double));

for (int i = 0; i < _dtCrestGeometryDownstream.Rows.Count; i++)

{

AeratedChuteRow row = new AeratedChuteRow();

row.HorizontalStation =
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][4]);

row.StreamwiseDistance =
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][8]);

row.BottomElevation =
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][5]);

row.NormalizedStreamwiseDistance =
_HydraulicSolver.CalculateNormalizedStreamwiseDistance(row.StreamwiseDista
nce, _SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _ExistingDischarge);

row.AeratedWaterDepth =
_HydraulicSolver.CalculateAeratedWaterDepth(row.NormalizedStreamwiseDistan
ce, _SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _ExistingDischarge);

row.BoundaryLayerThickness =
_HydraulicSolver.CalculateBoundaryLayerThickness(row.StreamwiseDistance,
_SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _ExistingDischarge);

row.AngleRadian =
_HydraulicSolver.CalculateAngleOfPoint(row.HorizontalStation,
_DesignHead);

row.RelativeHead = row.AeratedWaterDepth * Math.Cos(row.AngleRadian) +
Math.Pow((_ExistingDischarge / _SpillwayRoughLength /
row.AeratedWaterDepth), 2) / 19.62;


_dtAeratedChuteExisting.Rows.Add(new object[] {
```

```
Math.Round(row.HorizontalStation,2),
Math.Round(row.StreamwiseDistance,2),
Math.Round(row.BottomElevation,2),
Math.Round(row.NormalizedStreamwiseDistance,2),
Math.Round(row.AeratedWaterDepth,2),
Math.Round(row.BoundaryLayerThickness,2),
Math.Round(row.AngleRadian,2),
Math.Round(row.RelativeHead,2) });
}
for (int i = 0; i < _dtChuteGeometry.Rows.Count; i++)
{
AeratedChuteRow row = new AeratedChuteRow();
row.HorizontalStation = Convert.ToDouble(_dtChuteGeometry.Rows[i][0]);
row.StreamwiseDistance = Convert.ToDouble(_dtChuteGeometry.Rows[i][4]);
row.BottomElevation = Convert.ToDouble(_dtChuteGeometry.Rows[i][1]);
row.NormalizedStreamwiseDistance =
_HydraulicSolver.CalculateNormalizedStreamwiseDistance(row.StreamwiseDista
nce, _SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _ExistingDischarge);
row.AeratedWaterDepth =
_HydraulicSolver.CalculateAeratedWaterDepth(row.NormalizedStreamwiseDistan
ce, _SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _ExistingDischarge);
row.BoundaryLayerThickness =
_HydraulicSolver.CalculateBoundaryLayerThickness(row.StreamwiseDistance,
_SpillwayRoughLength, _ChuteAngleRadian, 0.0015, _ExistingDischarge);
row.AngleRadian = _ChuteAngleRadian;
row.RelativeHead = row.AeratedWaterDepth * Math.Cos(row.AngleRadian) +
Math.Pow((_ExistingDischarge / _SpillwayRoughLength /
row.AeratedWaterDepth), 2) / 19.62;
_dtAeratedChuteExisting.Rows.Add(new object[] {
Math.Round(row.HorizontalStation,2),
Math.Round(row.StreamwiseDistance,2),
Math.Round(row.BottomElevation,2),
Math.Round(row.NormalizedStreamwiseDistance,2),
Math.Round(row.AeratedWaterDepth,2),
```

```
Math.Round(row.BoundaryLayerThickness,2),

Math.Round(row.AngleRadian,2),

Math.Round(row.RelativeHead,2) });

}

}

}

private void PressureCalculations()

{

_dtPressureNoPier =
PressureRow.CreatePressureTable(PressureRow.dtPressureType.NoPier,
_DesignHead, _ExistingHead, _Temperature, _plUpstreamBottomProfile,
_plDownstreamBottomProfile,_ExistingDischarge,_SpillwayRoughLength);

_dtPressurePierBay =
PressureRow.CreatePressureTable(PressureRow.dtPressureType.PierBay,
_DesignHead, _ExistingHead, _Temperature, _plUpstreamBottomProfile,
_plDownstreamBottomProfile , _ExistingDischarge, _SpillwayRoughLength);

_dtPressureAlongPiers =
PressureRow.CreatePressureTable(PressureRow.dtPressureType.AlongPiers,
_DesignHead, _ExistingHead, _Temperature, _plUpstreamBottomProfile,
_plDownstreamBottomProfile, _ExistingDischarge,_SpillwayRoughLength);

}

private void CavitationCalculations()

{

_dtCavitation = new DataTable();

_dtCavitation.Rows.Clear();

_dtCavitation.Columns.Add("Station", typeof(double));

_dtCavitation.Columns.Add("FlowHeight Design", typeof(double));

_dtCavitation.Columns.Add("Velocity Design", typeof(double));

_dtCavitation.Columns.Add("Cavitation Parameter Design", typeof(double));

_dtCavitation.Columns.Add("FlowHeight Existing", typeof(double));

_dtCavitation.Columns.Add("Velocity Existing", typeof(double));

_dtCavitation.Columns.Add("Cavitation Parameter Existing",
typeof(double));


for (int i = 0; i < _dtCrestGeometryDownstream.Rows.Count; i++)
```

```
{
var row = _dtCrestDSWSP.Rows[i];
double station = Convert.ToDouble(row[0]);
double flowheightdesign = Convert.ToDouble(row[4]);
double velocitydesign = Convert.ToDouble(row[7]);
double cavparamdesign =
_HydraulicSolver.CalculateCavitationParameter(_Temperature,
flowheightdesign, velocitydesign);
double flowheightexisting = Convert.ToDouble(row[5]);
double velocityexisting = Convert.ToDouble(row[8]);
double cavparamexisting =
_HydraulicSolver.CalculateCavitationParameter(_Temperature,
flowheightexisting, velocityexisting);


_dtCavitation.Rows.Add(new object[] {
Math.Round(station,2),
Math.Round(flowheightdesign,2),
Math.Round(velocitydesign,2),
Math.Round(cavparamdesign,2),
Math.Round(flowheightexisting,2),
Math.Round(velocityexisting,2),
Math.Round(cavparamexisting,2) });
}
for (int i = 0; i < _dtChuteGeometry.Rows.Count; i++)
{
var chutegeorow = _dtChuteGeometry.Rows[i];
var designrow = _dtChuteWSPDesign.Rows[i];
var existingrow = _dtChuteWSPActual.Rows[i];


double station = Convert.ToDouble(chutegeorow[0]);
double flowheightdesign = Convert.ToDouble(designrow[2]);
double velocitydesign = Convert.ToDouble(designrow[6]);
```

```
double cavparamdesign =
_HydraulicSolver.CalculateCavitationParameter(_Temperature,
flowheightdesign, velocitydesign);

double flowheightexisting = Convert.ToDouble(existingrow[2]);

double velocityexisting = Convert.ToDouble(existingrow[6]);

double cavparamexisting =
_HydraulicSolver.CalculateCavitationParameter(_Temperature,
flowheightexisting, velocityexisting);


_dtCavitation.Rows.Add(new object[] {

Math.Round(station,2),

Math.Round(flowheightdesign,2),

Math.Round(velocitydesign,2),

Math.Round(cavparamdesign,2),

Math.Round(flowheightexisting,2),

Math.Round(velocityexisting,2),

Math.Round(cavparamexisting,2) });

}}

private void DissipationCalculations()

{

_dtPreDissipationTable = new DataTable();

_dtPreDissipationTable.Rows.Clear();

_dtPreDissipationTable.Columns.Add("Approach", typeof(string));

_dtPreDissipationTable.Columns.Add("Discharge", typeof(double));

_dtPreDissipationTable.Columns.Add("FlowDepthBeforeJump", typeof(double));

_dtPreDissipationTable.Columns.Add("FroudeNumberBeforeJump",
typeof(double));

_dtPreDissipationTable.Columns.Add("VelocityBeforeJump", typeof(double));

_dtPreDissipationTable.Columns.Add("EnergyLevelBeforeJump",
typeof(double));

_dtPreDissipationTable.Columns.Add("FlowDepthAfterJump", typeof(double));

_dtPreDissipationTable.Columns.Add("FroudeNumberAfterJump",
typeof(double));

_dtPreDissipationTable.Columns.Add("VelocityAfterJump", typeof(double));
```

```
_dtPreDissipationTable.Columns.Add("EnergyLevelAfterJump",
typeof(double));

_dtPreDissipationTable.Columns.Add("Basin Type", typeof(string));

_dtPreDissipationTable.Columns.Add("StillingBasinLength", typeof(double));

dissipationrowlist = new List<DissipationRow>();

if (chkbxDesignDischarge.Checked && chkApproach1.Checked)

{

_FlowDepthBeforeJumpApproach1Design =
Convert.ToDouble(_dtChuteWSPDesign.Rows[_dtChuteWSPDesign.Rows.Count -
1][2]);

DissipationRow row = new DissipationRow();

row.approachdesignindex =
DissipationRow.eApproachDesignIndex.approach1Design;

if (cmbWaterwayType.SelectedIndex == 0)

{

row = _HydraulicSolver.CreateDissipationRow("approach1-Design",
_DesignDischarge, _FlowDepthBeforeJumpApproach1Design,
_SpillwayRoughLength, _StillingBasinElevation, _BasinIndex,
_ChuteAngleRadian);

}

else

{

row = _HydraulicSolver.CreateDissipationRow("approach1-Design",
_DesignDischarge, _CascadeDesign.UniformDepth, _SpillwayRoughLength,
_StillingBasinElevation, _BasinIndex, _ChuteAngleRadian);

}

dissipationrowlist.Add(row);

_dtPreDissipationTable.Rows.Add(new object[] {

row.Approach,

Math.Round(row.Discharge,2),

Math.Round(row.FlowDepthBeforeJump,2),

Math.Round(row.FroudeNumberBeforeJump,2),

Math.Round(row.VelocityBeforeJump,2),

Math.Round(row.EnergyLevelBeforeJump,2),

Math.Round(row.FlowDepthAfterJump,2),
```

```
Math.Round(row.FroudeNumberAfterJump,2),

Math.Round(row.VelocityAfterJump,2),

Math.Round(row.EnergyLevelAfterJump,2),

row.BasinType,

Math.Round(row.StillingBasinLength,2) });

}

if (chkbxExistingDischarge.Checked && chkApproach1.Checked)

{

_FlowDepthBeforeJumpApproach1Existing =
Convert.ToDouble(_dtChuteWSPExisting.Rows[_dtChuteWSPExisting.Rows.Count -
1][2]);

DissipationRow row = new DissipationRow();

row.approachdesignindex =
DissipationRow.eApproachDesignIndex.approach1Existing;

if (cmbWaterwayType.SelectedIndex ==0)

{

row = _HydraulicSolver.CreateDissipationRow("approach1-Existing",
_ExistingDischarge, _FlowDepthBeforeJumpApproach1Existing,
_SpillwayRoughLength, _StillingBasinElevation, _BasinIndex,
_ChuteAngleRadian);

}

else

{

row = _HydraulicSolver.CreateDissipationRow("approach1-Existing",
_ExistingDischarge, _CascadeExisting.UniformDepth, _SpillwayRoughLength,
_StillingBasinElevation, _BasinIndex, _ChuteAngleRadian);

}

dissipationrowlist.Add(row);

_dtPreDissipationTable.Rows.Add(new object[] {

row.Approach,

Math.Round(row.Discharge,2),

Math.Round(row.FlowDepthBeforeJump,2),

Math.Round(row.FroudeNumberBeforeJump,2),

Math.Round(row.VelocityBeforeJump,2),
```

```
Math.Round(row.EnergyLevelBeforeJump,2),

Math.Round(row.FlowDepthAfterJump,2),

Math.Round(row.FroudeNumberAfterJump,2),

Math.Round(row.VelocityAfterJump,2),

Math.Round(row.EnergyLevelAfterJump,2),

row.BasinType,

Math.Round(row.StillingBasinLength,2) });

}

if (chkbxDesignDischarge.Checked && chkApproach2.Checked)

{

_FlowDepthBeforeJumpApproach2Design =
Convert.ToDouble(_dtAeratedChuteDesign.Rows[_dtAeratedChuteDesign.Rows.Cou
nt - 1][4]);

DissipationRow row = new DissipationRow();

row.approachdesignindex =
DissipationRow.eApproachDesignIndex.approach2Design;

row = _HydraulicSolver.CreateDissipationRow("approach2-Design",
_DesignDischarge, _FlowDepthBeforeJumpApproach2Design,
_SpillwayRoughLength, _StillingBasinElevation, _BasinIndex,
_ChuteAngleRadian);

dissipationrowlist.Add(row);

_dtPreDissipationTable.Rows.Add(new object[] {

row.Approach,

Math.Round(row.Discharge,2),

Math.Round(row.FlowDepthBeforeJump,2),

Math.Round(row.FroudeNumberBeforeJump,2),

Math.Round(row.VelocityBeforeJump,2),

Math.Round(row.EnergyLevelBeforeJump,2),

Math.Round(row.FlowDepthAfterJump,2),

Math.Round(row.FroudeNumberAfterJump,2),

Math.Round(row.VelocityAfterJump,2),

Math.Round(row.EnergyLevelAfterJump,2),

row.BasinType,

Math.Round(row.StillingBasinLength,2) });
```

```
}

if (chkbxExistingDischarge.Checked && chkApproach2.Checked)

{

_FlowDepthBeforeJumpApproach2Existing =
Convert.ToDouble(_dtAeratedChuteExisting.Rows[_dtAeratedChuteExisting.Rows
.Count - 1][4]);

DissipationRow row = new DissipationRow();

row.approachdesignindex =
DissipationRow.eApproachDesignIndex.approach2Existing;

row = _HydraulicSolver.CreateDissipationRow("approach2-Existing",
_ExistingDischarge, _FlowDepthBeforeJumpApproach2Existing,
_SpillwayRoughLength, _StillingBasinElevation, _BasinIndex,
_ChuteAngleRadian);

dissipationrowlist.Add(row);

_dtPreDissipationTable.Rows.Add(new object[] {

row.Approach,

Math.Round(row.Discharge,2),

Math.Round(row.FlowDepthBeforeJump,2),

Math.Round(row.FroudeNumberBeforeJump,2),

Math.Round(row.VelocityBeforeJump,2),

Math.Round(row.EnergyLevelBeforeJump,2),

Math.Round(row.FlowDepthAfterJump,2),

Math.Round(row.FroudeNumberAfterJump,2),

Math.Round(row.VelocityAfterJump,2),

Math.Round(row.EnergyLevelAfterJump,2),

row.BasinType,

Math.Round(row.StillingBasinLength,2) });

}

_MaxEnergy = 0;

foreach (DissipationRow row in dissipationrowlist)

{

if (row.EnergyLevelAfterJump > _MaxEnergy)

{

_MaxEnergy = row.EnergyLevelAfterJump;
```

```
}
}


_MaxFlowDepthAfterJump = 0;
foreach (DissipationRow row in dissipationrowlist)
{
if (row.FlowDepthAfterJump > _MaxFlowDepthAfterJump)
{
_MaxFlowDepthAfterJump = row.FlowDepthAfterJump;
}
}
}
private void StillingBasinElevationAdjustment(double maxenergy)
{
for (_StillingBasinElevation = _DownstreamBedElevation; maxenergy >=
_DownstreamEnergyElevation; _StillingBasinElevation -= 0.01)
{
GenerateWaterwayGeometry(_StillingBasinElevation);
DissipationCalculations();
maxenergy = _MaxEnergy;
lblOutputStillingBasinElevation.Text = "Stilling basin elevation is set to
" + Math.Round(_StillingBasinElevation, 2) + "m";
lblOutputMaxEnergy.Text = "Energy level after jump is " +
Math.Round(maxenergy, 2) + "m";
lblOutputMaxEnergy.ForeColor = Color.Black;
if (maxenergy > _DownstreamEnergyElevation)
{
lblOutputMaxEnergy.ForeColor = Color.Red;
}


RemoveAllEntities();
DrawOverall();
```

```
DrawCrest();
DrawWaterway();
InsertAllElevationTexts();
Regen();
}
}


public static DataTable CreatePressureTable(dtPressureType dt, double
designhead, double Existinghead, double temperature, Polyline plUS,
Polyline plDS, double ExistingDischarge, double SpillwayRoughLength)
{
DataTable dtPressure = new DataTable();
dtPressure.Rows.Clear();
dtPressure.Columns.Add("Rel. Dist", typeof(double));
dtPressure.Columns.Add("H.R = 0.5", typeof(double));
dtPressure.Columns.Add("H.R = 1.0", typeof(double));
dtPressure.Columns.Add("H.R = 1.33", typeof(double));
dtPressure.Columns.Add("Existing H.R", typeof(double));
dtPressure.Columns.Add("Distance", typeof(double));
dtPressure.Columns.Add("Pressure Height", typeof(double));
List<double> ListOfRelDist = new List<double>();
for (int i = plUS.Vertexes.Count - 1; i > 0; i--)
{
ListOfRelDist.Add(plUS.Vertexes[i].Point.x / designhead);
}
double headratio = Existinghead / designhead;
switch (dt)
{
case dtPressureType.NoPier:
foreach (double item in ListOfRelDist)
{
dtPressure.Rows.Add(new object[] {
```

```
Math.Round(item,2),
Math.Round(Calculatehp050NoPier(item, designhead),2),
Math.Round(Calculatehp100NoPier(item, designhead),2),
Math.Round(Calculatehp133NoPier(item, designhead),2),
Math.Round(headratio,2),
Math.Round(item * designhead,2),
Math.Round(Calculatehp(headratio, Calculatehp050NoPier(item, designhead),
Calculatehp100NoPier(item, designhead), Calculatehp133NoPier(item,
designhead)),2),
});
}
break;
case dtPressureType.PierBay:
foreach (double item in ListOfRelDist)
{
dtPressure.Rows.Add(new object[] {
Math.Round(item,2),
Math.Round(Calculatehp050PierBay(item, designhead),2),
Math.Round(Calculatehp100PierBay(item, designhead),2),
Math.Round(Calculatehp133PierBay(item, designhead),2),
Math.Round(headratio,2),
Math.Round(item * designhead,2),
Math.Round(Calculatehp(headratio, Calculatehp050PierBay(item, designhead),
Calculatehp100PierBay(item, designhead), Calculatehp133PierBay(item,
designhead)),2),
});
}
break;
case dtPressureType.AlongPiers:
foreach (double item in ListOfRelDist)
{
dtPressure.Rows.Add(new object[] {
Math.Round(item,2),
```

```
Math.Round(Calculatehp050AlongPiers(item, designhead),2),

Math.Round(Calculatehp100AlongPiers(item, designhead),2),

Math.Round(Calculatehp133AlongPiers(item, designhead),2),

Math.Round(headratio,2),

Math.Round(item * designhead,2),

Math.Round(Calculatehp(headratio, Calculatehp050AlongPiers(item,
designhead), Calculatehp100AlongPiers(item, designhead),
Calculatehp133AlongPiers(item, designhead)),2),

});

}

break;

}

return dtPressure;

}
```

A.6. Drawing Methods

```
private void DrawCrestGeometry(Canvas winform)

{

_plUpstreamBottomProfile = new Polyline();

for (int i = 0; i < _CrestGeometryUpstreamNodes; i++)

{

PolylineVertex vertex = new PolylineVertex(new
XYPT(Convert.ToDouble(_dtCrestGeometryUpstream.Rows[i][4]),
Convert.ToDouble(_dtCrestGeometryUpstream.Rows[i][5])));

_plUpstreamBottomProfile.Vertexes.Add(vertex);

}

_plUpstreamBottomProfile.OverrideColor = new
PBUI.Colour.PDColor(Color.Black);

_plDownstreamBottomProfile = new Polyline();

for (int i = 0; i < _CrestGeometryDownstreamNodes; i++)

{

PolylineVertex vertex = new PolylineVertex(new
XYPT(Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][4]),
Convert.ToDouble(_dtCrestGeometryDownstream.Rows[i][5])));
```

```
_plDownstreamBottomProfile.Vertexes.Add(vertex);
}
winform.Document.Entities.Add(_plUpstreamBottomProfile);
winform.Document.Entities.Add(_plDownstreamBottomProfile);
}
private void DrawCrestWaterSurfaceProfiles(Canvas winform)
{
winform.Document.Entities.Remove(_plWSPDesignUpstream);
winform.Document.Entities.Remove(_plWSPDesignDownstream);
winform.Document.Entities.Remove(_plWSPExistingUpstream);
winform.Document.Entities.Remove(_plWSPExistingDownstream);
if (chkbxDesignDischarge.Checked)
{
winform.Document.Entities.Add(_plWSPDesignUpstream);
_plWSPDesignUpstream.OverrideColor = new PBUI.Colour.PDColor(Color.Cyan);
winform.Document.Entities.Add(_plWSPDesignDownstream);
_plWSPDesignDownstream.OverrideColor = new
PBUI.Colour.PDColor(Color.Cyan);
}
if (chkbxExistingDischarge.Checked)
{
winform.Document.Entities.Add(_plWSPExistingUpstream);
_plWSPExistingUpstream.OverrideColor = new
PBUI.Colour.PDColor(Color.Blue);
winform.Document.Entities.Add(_plWSPExistingDownstream);
_plWSPExistingDownstream.OverrideColor = new
PBUI.Colour.PDColor(Color.Blue);
}
}
private void DrawWaterwayGeometry(Canvas winform)
{
winform.Document.Entities.Add(_plWaterwayBottomProfile);
```

```
if (cmbWaterwayType.SelectedIndex == 1)
{
winform.Document.Entities.Add(_plPseudoBottom);
}
}
private void DrawLinkBetweenCrestAndChute(Canvas winform)
{
if (chkbxDesignDischarge.Checked && cmbWaterwayType.SelectedIndex == 0)
{
winform.Document.Entities.Add(_linkDesign);
_linkDesign.OverrideColor = new PBUI.Colour.PDColor(Color.Cyan);
}
if (chkbxExistingDischarge.Checked && cmbWaterwayType.SelectedIndex == 0)
{
winform.Document.Entities.Add(_linkExisting);
_linkExisting.OverrideColor = new PBUI.Colour.PDColor(Color.Blue);
}
}
private void DrawWaterwayWaterSurfaceProfiles(Canvas winform)
{
if (chkbxDesignDischarge.Checked && cmbWaterwayType.SelectedIndex == 0)
{
winform.Document.Entities.Add(_plWSPDesignStandardStep);
_plWSPDesignStandardStep.OverrideColor = new
PBUI.Colour.PDColor(Color.Cyan);
}
if (chkbxExistingDischarge.Checked && cmbWaterwayType.SelectedIndex == 0)
{
winform.Document.Entities.Add(_plWSPExistingStandardStep);
_plWSPExistingStandardStep.OverrideColor = new
PBUI.Colour.PDColor(Color.Blue);
}
```

```
if (chkbxDesignDischarge.Checked && cmbWaterwayType.SelectedIndex == 1)
{
winform.Document.Entities.Add(_plCascadeDesignWSP);
_plCascadeDesignWSP.OverrideColor = new PBUI.Colour.PDColor(Color.Cyan);
}
if (chkbxExistingDischarge.Checked && cmbWaterwayType.SelectedIndex == 1)
{
winform.Document.Entities.Add(_plCascadeExistingWSP);
_plCascadeExistingWSP.OverrideColor = new PBUI.Colour.PDColor(Color.Blue);
}
}
private void DrawAeratedCrestWSP(Canvas winform)
{
winform.Document.Entities.Remove(_plDesignWSP);
winform.Document.Entities.Remove(_plExistingWSP);
if (chkbxDesignDischarge.Checked)
{
Polyline plDesignWSP = new Polyline();
for (int i = 0; i < _dtCrestGeometryDownstream.Rows.Count; i++)
{
PolylineVertex vertexdesign = new PolylineVertex(new
XYPT(Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][0]) +
Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][4]) *
Math.Sin(Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][6])),
Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][2]) +
Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][4]) *
Math.Cos(Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][6]))));
plDesignWSP.Vertexes.Add(vertexdesign);
}
winform.Document.Entities.Add(plDesignWSP);
plDesignWSP.OverrideColor = new PBUI.Colour.PDColor(Color.Orange);
}
if (chkbxExistingDischarge.Checked)
{
```

```
Polyline plExistingWSP = new Polyline();

for (int i = 0; i < _dtCrestGeometryDownstream.Rows.Count; i++)

{

PolylineVertex vertexExisting = new PolylineVertex(new
XYPT(Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][0]) +
Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][4]) *
Math.Sin(Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][6])),
Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][2]) +
Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][4]) *
Math.Cos(Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][6]))));

plExistingWSP.Vertexes.Add(vertexExisting);

}

winform.Document.Entities.Add(plExistingWSP);

plExistingWSP.OverrideColor = new PBUI.Colour.PDColor(Color.Green);

}

}

private void DrawAeratedChuteWSP(Canvas winform)

{

if (chkbxDesignDischarge.Checked && cmbWaterwayType.SelectedIndex == 0)

{

winform.Document.Entities.Remove(_plDesignWSP);

_plDesignWSP = new Polyline();

for (int i = _dtCrestGeometryDownstream.Rows.Count; i <
_dtCrestGeometryDownstream.Rows.Count + _dtChuteGeometry.Rows.Count; i++)

{

PolylineVertex vertexdesign = new PolylineVertex(new
XYPT(Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][0]) +
Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][4]) *
Math.Sin(Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][6])),
Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][2]) +
Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][4]) *
Math.Cos(Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][6]))));

_plDesignWSP.Vertexes.Add(vertexdesign);

}

winform.Document.Entities.Add(_plDesignWSP);

_plDesignWSP.OverrideColor = new PBUI.Colour.PDColor(Color.Orange);
```

```
}

if (chkbxExistingDischarge.Checked && cmbWaterwayType.SelectedIndex == 0)

{

winform.Document.Entities.Remove(_plExistingWSP);

_plExistingWSP = new Polyline();

for (int i = _dtCrestGeometryDownstream.Rows.Count; i <
_dtCrestGeometryDownstream.Rows.Count + _dtChuteGeometry.Rows.Count; i++)

{

PolylineVertex vertexExisting = new PolylineVertex(new
XYPT(Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][0]) +
Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][4]) *
Math.Sin(Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][6])),
Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][2]) +
Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][4]) *
Math.Cos(Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][6]))));

_plExistingWSP.Vertexes.Add(vertexExisting);

}

winform.Document.Entities.Add(_plExistingWSP);

_plExistingWSP.OverrideColor = new PBUI.Colour.PDColor(Color.Green);

}

}

private void DrawCrestPressures()

{

_plcrestpressuresNoPier = new Polyline();

_plcrestpressurePierBay = new Polyline();

_plcrestpressureAlongPiers = new Polyline();

if (txtPierNumber.Text == "0")

{

_plcrestpressuresNoPier.Vertexes.Add(new
PolylineVertex(_plUpstreamBottomProfile.Vertexes[_plUpstreamBottomProfile.
Vertexes.Count - 1].Point));

for (int i = 0; i < _plUpstreamBottomProfile.Vertexes.Count - 1; i++)

{

_plcrestpressuresNoPier.Vertexes.Add(new
PolylineVertex(_plUpstreamBottomProfile.Vertexes[20 - i].Point + new
XYPT(0, Convert.ToDouble(_dtPressureNoPier.Rows[i][6]))));
```

```
}

for (int i = 0; i < _plDownstreamBottomProfile.Vertexes.Count - 1; i++)

{

_plcrestpressuresNoPier.Vertexes.Add(new
PolylineVertex(_plDownstreamBottomProfile.Vertexes[i].Point + new XYPT(0,
Convert.ToDouble(_dtPressureNoPier.Rows[i + 20][6])));

}

_plcrestpressuresNoPier.OverrideColor = new
PBUI.Colour.PDColor(Color.Crimson);

chkbxNoPier.ForeColor = Color.Crimson;

canvasCrest.Document.Entities.Add(_plcrestpressuresNoPier);

}

else

{

if (chkbxPierBay.Checked)

{

_plcrestpressurePierBay.Vertexes.Add(new
PolylineVertex(_plUpstreamBottomProfile.Vertexes[_plUpstreamBottomProfile.
Vertexes.Count - 1].Point));

for (int i = 0; i < _plUpstreamBottomProfile.Vertexes.Count - 1; i++)

{

_plcrestpressurePierBay.Vertexes.Add(new
PolylineVertex(_plUpstreamBottomProfile.Vertexes[20 - i].Point + new
XYPT(0, Convert.ToDouble(_dtPressurePierBay.Rows[i][6])));

}

for (int i = 0; i < _plDownstreamBottomProfile.Vertexes.Count - 1; i++)

{

_plcrestpressurePierBay.Vertexes.Add(new
PolylineVertex(_plDownstreamBottomProfile.Vertexes[i].Point + new XYPT(0,
Convert.ToDouble(_dtPressurePierBay.Rows[i + 20][6])));

}

_plcrestpressurePierBay.OverrideColor = new
PBUI.Colour.PDColor(Color.MediumSlateBlue);

chkbxPierBay.ForeColor = Color.MediumSlateBlue;

canvasCrest.Document.Entities.Add(_plcrestpressurePierBay);
```

```
}
if (chkbxAlongPiers.Checked)
{
_plcrestpressureAlongPiers.Vertexes.Add(new
PolylineVertex(_plUpstreamBottomProfile.Vertexes[_plUpstreamBottomProfile.
Vertexes.Count - 1].Point));
for (int i = 0; i < _plUpstreamBottomProfile.Vertexes.Count - 1; i++)
{
_plcrestpressureAlongPiers.Vertexes.Add(new
PolylineVertex(_plUpstreamBottomProfile.Vertexes[20 - i].Point + new
XYPT(0, Convert.ToDouble(_dtPressureAlongPiers.Rows[i][6]))));
}
for (int i = 0; i < _plDownstreamBottomProfile.Vertexes.Count - 1; i++)
{
_plcrestpressureAlongPiers.Vertexes.Add(new
PolylineVertex(_plDownstreamBottomProfile.Vertexes[i].Point + new XYPT(0,
Convert.ToDouble(_dtPressureAlongPiers.Rows[i + 20][6]))));
}
_plcrestpressureAlongPiers.OverrideColor = new
PBUI.Colour.PDColor(Color.Crimson);
chkbxAlongPiers.ForeColor = Color.Crimson;
canvasCrest.Document.Entities.Add(_plcrestpressureAlongPiers);
}
}
}
private void DrawDissipationGeometry(Canvas winform)
{
_MaximumStillingBasinLength = 0;
foreach (DissipationRow row in dissipationrowlist)
{
if (row.StillingBasinLength > _MaximumStillingBasinLength)
{
_MaximumStillingBasinLength = row.StillingBasinLength;
}
```

```
}

_plStillingBasinGeometry = new Polyline();

_plStillingBasinGeometry.Vertexes.Add(new PolylineVertex(new
XYPT(_plWaterwayBottomProfile.Vertexes[_plWaterwayBottomProfile.Vertexes.C
ount - 1].Point)));

_plStillingBasinGeometry.Vertexes.Add(new PolylineVertex(new
XYPT(_plStillingBasinGeometry.Vertexes[0].Point) + new
XYPT(_MaximumStillingBasinLength, 0)));

winform.Document.Entities.Add(_plStillingBasinGeometry);


_plDissipationStep = new Polyline();

_plDownstreamBed = new Polyline();


_plDissipationStep.Vertexes.Add(_plStillingBasinGeometry.Vertexes[1]);

_plDissipationStep.Vertexes.Add(new PolylineVertex(new
XYPT(_plDissipationStep.Vertexes[0].Point.x, _DownstreamBedElevation)));


_plDownstreamBed.Vertexes.Add(_plDissipationStep.Vertexes[1]);

_plDownstreamBed.Vertexes.Add(new PolylineVertex(new
XYPT(_plDissipationStep.Vertexes[1].Point) + new XYPT(2, 0)));


if (Math.Abs(_StillingBasinElevation - _DownstreamBedElevation) <= 0.001)
{ }

else { winform.Document.Entities.Add(_plDissipationStep); }

winform.Document.Entities.Add(_plDownstreamBed);

_plDownstreamBed.OverrideColor = new PBUI.Colour.PDColor(Color.DarkGray);


Polyline _dimJumpRegion = new Polyline();

_dimJumpRegion.Vertexes.Add(new PolylineVertex(new
XYPT(_plStillingBasinGeometry.Vertexes[0].Point.x, (_MaxEnergy -
_StillingBasinElevation) * -0.2 + _StillingBasinElevation)));

_dimJumpRegion.Vertexes.Add(new PolylineVertex(new
XYPT(_plStillingBasinGeometry.Vertexes[1].Point.x, (_MaxEnergy -
_StillingBasinElevation) * -0.2 + _StillingBasinElevation)));

winform.Document.Entities.Add(_dimJumpRegion);
```

```
double arrowsize = _MaximumStillingBasinLength / 100;

winform.Document.Entities.Add(DrawingHelper.DrawArrowHead(arrowsize, 0,
_dimJumpRegion.Vertexes[0].Point));

winform.Document.Entities.Add(DrawingHelper.DrawArrowHead(arrowsize, 180,
_dimJumpRegion.Vertexes[1].Point));


double fontsize = GetAppropriateTextSize(winform);

Font deffont2 = new Font("Arial", Convert.ToSingle(fontsize));

Text txtJumpRegion = new Text(indicators);

txtJumpRegion.InsPt = new XYPT((_dimJumpRegion.Vertexes[0].Point.x +
_dimJumpRegion.Vertexes[1].Point.x) / 2,
_dimJumpRegion.Vertexes[0].Point.y + arrowsize);

txtJumpRegion.Text = "Hydraulic Jump Region";

txtJumpRegion.OverrideAlingment = eText_Alignment.MiddleCenter;

txtJumpRegion.OverrideFont = deffont2;

winform.Document.Entities.Add(txtJumpRegion);

}

private void DrawHydraulicJumpVisuals(Canvas winform)

{

foreach (DissipationRow row in dissipationrowlist)

switch (row.Approach)

{

case "approach1-Design":

plhjdesignapp1 = new Polyline();

plhjdesignapp1.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBed.Vertexes[0].Point.x, _StillingBasinElevation +
row.FlowDepthAfterJump)));

plhjdesignapp1.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBed.Vertexes[0].Point.x - _MaximumStillingBasinLength *
0.15, _StillingBasinElevation + row.FlowDepthAfterJump)));


plhjdesignapp1.OverrideColor = new PBUI.Colour.PDColor(Color.Cyan);

winform.Document.Entities.Add(plhjdesignapp1);

break;
```

```
case "approach1-Existing":

plhjExistingapp1 = new Polyline();

plhjExistingapp1.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBed.Vertexes[0].Point.x, _StillingBasinElevation +
row.FlowDepthAfterJump)));

plhjExistingapp1.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBed.Vertexes[0].Point.x - _MaximumStillingBasinLength *
0.15, _StillingBasinElevation + row.FlowDepthAfterJump)));


plhjExistingapp1.OverrideColor = new PBUI.Colour.PDColor(Color.Blue);

winform.Document.Entities.Add(plhjExistingapp1);

break;

case "approach2-Design":

plhjdesignapp2 = new Polyline();

plhjdesignapp2.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBed.Vertexes[0].Point.x, _StillingBasinElevation +
row.FlowDepthAfterJump)));

plhjdesignapp2.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBed.Vertexes[0].Point.x - _MaximumStillingBasinLength *
0.15, _StillingBasinElevation + row.FlowDepthAfterJump)));


plhjdesignapp2.OverrideColor = new PBUI.Colour.PDColor(Color.Orange);

winform.Document.Entities.Add(plhjdesignapp2);

break;

case "approach2-Existing":

plhjExistingapp2 = new Polyline();

plhjExistingapp2.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBed.Vertexes[0].Point.x, _StillingBasinElevation +
row.FlowDepthAfterJump)));

plhjExistingapp2.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBed.Vertexes[0].Point.x - _MaximumStillingBasinLength *
0.15, _StillingBasinElevation + row.FlowDepthAfterJump)));


plhjExistingapp2.OverrideColor = new PBUI.Colour.PDColor(Color.Green);

winform.Document.Entities.Add(plhjExistingapp2);

break;
```

```
default:
break;
}
}
private void DrawCrest()
{
DrawCrestGeometry(canvasCrest);
PressureCalculations();
if (chkApproach1.Checked)
{
GenerateCrestWaterSurfaceProfiles();
DrawCrestWaterSurfaceProfiles(canvasCrest);
}
if (chkApproach2.Checked)
{
AeratedChuteCalculations();
DrawAeratedCrestWSP(canvasCrest);
}
DrawCrestPressures();
DrawHorizontalAxis(canvasCrest);
}
private void DrawWaterway()
{
DrawWaterwayGeometry(canvaswaterway);
if (chkApproach1.Checked)
{
DrawWaterwayWaterSurfaceProfiles(canvaswaterway);
}
if (chkApproach2.Checked)
{
DrawAeratedChuteWSP(canvaswaterway);
```

```
}
DrawDissipationGeometry(canvaswaterway);
DrawHydraulicJumpVisuals(canvaswaterway);
DrawHorizontalAxis(canvaswaterway);
}
private void DrawOverall()
{
canvasoverall.Document.ClearEntities();
GenerateCrestGeometry();
DrawCrestGeometry(canvasoverall);
PressureCalculations();
GenerateWaterwayGeometry(_StillingBasinElevation);
DrawWaterwayGeometry(canvasoverall);
if (chkApproach1.Checked)
{
GenerateCrestWaterSurfaceProfiles();
DrawCrestWaterSurfaceProfiles(canvasoverall);
GenerateWaterwayWaterSurfaceProfiles(_InitialFlowDepthDesign *
Math.Cos(_ChuteAngleRadian), _InitialFlowDepthExisting *
Math.Cos(_ChuteAngleRadian));
DrawWaterwayWaterSurfaceProfiles(canvasoverall);
GenerateLinkBetweenCrestAndChute();
DrawLinkBetweenCrestAndChute(canvasoverall);
}
if (chkApproach2.Checked)
{
DrawAeratedCrestWSP(canvasoverall);
DrawAeratedChuteWSP(canvasoverall);
}
DissipationCalculations();
DrawDissipationGeometry(canvasoverall);
DrawHydraulicJumpVisuals(canvasoverall);
```

```
DrawHorizontalAxis(canvasoverall);

}

private void DrawEnergyGradeLine(Canvas winform)

{

Polyline pldsbedegl = new Polyline();

pldsbedegl.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBed.Vertexes[0].Point.x, _DownstreamEnergyElevation)));

pldsbedegl.Vertexes.Add(new PolylineVertex(new
XYPT(_plDownstreamBed.Vertexes[1].Point.x, _DownstreamEnergyElevation)));

pldsbedegl.OverrideColor = new PBUI.Colour.PDColor(Color.DarkRed);

winform.Document.Entities.Add(pldsbedegl);

foreach (DissipationRow row in dissipationrowlist)

{

Polyline pldissipationegl = new Polyline();

pldissipationegl.Vertexes.Add(new PolylineVertex(new
XYPT(_plStillingBasinGeometry.Vertexes[0].Point.x,
row.EnergyLevelBeforeJump)));

pldissipationegl.Vertexes.Add(new PolylineVertex(new
XYPT(_plStillingBasinGeometry.Vertexes[1].Point.x,
row.EnergyLevelAfterJump)));

pldissipationegl.OverrideColor = new PBUI.Colour.PDColor(Color.Red);

winform.Document.Entities.Add(pldissipationegl);

}

// waterway

double waterwaystepdistance =
(_plWaterwayBottomProfile.Vertexes[1].Point.x -
_plWaterwayBottomProfile.Vertexes[0].Point.x) / 20;

if (chkApproach1.Checked && chkbxDesignDischarge.Checked)

{

Polyline plwaterwayeglapproach1design = new Polyline();

for (int i = 0; i < _dtChuteWSPDesign.Rows.Count; i++)

{

plwaterwayeglapproach1design.Vertexes.Add(new PolylineVertex(new
XYPT(Convert.ToDouble(_dtChuteWSPDesign.Rows[i][0]) * waterwaystepdistance
+_Crest.PointOfTangency.x,
Convert.ToDouble(_dtChuteWSPDesign.Rows[i][9]))));
```

```
}

winform.Document.Entities.Add(plwaterwayeglapproach1design);

}

if (chkApproach2.Checked && chkbxDesignDischarge.Checked)

{

Polyline plwaterwayeglapproach2design = new Polyline();

for (int i = 0; i < _dtAeratedChuteDesign.Rows.Count; i++)

{

plwaterwayeglapproach2design.Vertexes.Add(new PolylineVertex(new
XYPT(Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][0]),
Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][2]) +
Convert.ToDouble(_dtAeratedChuteDesign.Rows[i][7]))));

}

winform.Document.Entities.Add(plwaterwayeglapproach2design);

}

if (chkApproach2.Checked && chkbxExistingDischarge.Checked)

{

Polyline plwaterwayeglapproach2Existing = new Polyline();

for (int i = 0; i < _dtAeratedChuteExisting.Rows.Count; i++)

{

plwaterwayeglapproach2Existing.Vertexes.Add(new PolylineVertex(new
XYPT(Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][0]),
Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][2]) +
Convert.ToDouble(_dtAeratedChuteExisting.Rows[i][7]))));

}

winform.Document.Entities.Add(plwaterwayeglapproach2Existing);

}

if (chkApproach1.Checked && chkbxExistingDischarge.Checked)

{

Polyline plwaterwayeglapproach1Existing = new Polyline();

for (int i = 0; i < _dtChuteWSPExisting.Rows.Count; i++)

{

plwaterwayeglapproach1Existing.Vertexes.Add(new PolylineVertex(new
XYPT(Convert.ToDouble(_dtChuteWSPExisting.Rows[i][0]) *
```

```
waterwaystepdistance + _Crest.PointOfTangency.x,
Convert.ToDouble(_dtChuteWSPExisting.Rows[i][9])));

}

winform.Document.Entities.Add(plwaterwayeglapproach1Existing);

}

//crest

if (chkbxDesignDischarge.Checked && chkApproach1.Checked)

{

Polyline plcrestdesign = new Polyline();

for (int i = 0; i < _dtCrestDSWSP.Rows.Count; i++)

{

plcrestdesign.Vertexes.Add(new PolylineVertex(new
XYPT(Convert.ToDouble(_dtCrestDSWSP.Rows[i][0]),
Convert.ToDouble(_dtCrestDSWSP.Rows[i][11])));

}

winform.Document.Entities.Add(plcrestdesign);

}

if (chkbxExistingDischarge.Checked && chkApproach1.Checked)

{

Polyline plcrestExisting = new Polyline();

for (int i = 0; i < _dtCrestDSWSP.Rows.Count; i++)

{

plcrestExisting.Vertexes.Add(new PolylineVertex(new
XYPT(Convert.ToDouble(_dtCrestDSWSP.Rows[i][0]),
Convert.ToDouble(_dtCrestDSWSP.Rows[i][12])));

}

winform.Document.Entities.Add(plcrestExisting);

}

}

private void InsertAllElevationTexts()

{

double sizeoverall = GetAppropriateTextSize(canvasoverall);

double sizewaterway = GetAppropriateTextSize(canvaswaterway);
```

```
double sizecrest = GetAppropriateTextSize(canvasCrest);

InsertElevation(canvasCrest, _Crest.InsPt, _Crest.InsPt.y.ToString() + "
m",sizecrest/2,eText_Alignment.MiddleCenter);

InsertElevation(canvasCrest, _Crest.PointOfTangency,
Math.Round(_Crest.PointOfTangency.y, 2).ToString() + " m", sizecrest/2,
eText_Alignment.MiddleCenter);

InsertElevation(canvaswaterway, _Crest.PointOfTangency,
Math.Round(_Crest.PointOfTangency.y,2).ToString() + " m", sizewaterway,
eText_Alignment.MiddleCenter);

InsertElevation(canvaswaterway,
_plWaterwayBottomProfile.Vertexes[_plWaterwayBottomProfile.Vertexes.Count
- 1].Point + new XYPT(1, 0),
Math.Round(_plWaterwayBottomProfile.Vertexes[_plWaterwayBottomProfile.Vert
exes.Count - 1].Point.y, 2).ToString() + " m", sizewaterway,
eText_Alignment.MiddleCenter);

InsertElevation(canvaswaterway,
_plDownstreamBed.Vertexes[0].Point.Midpt(_plDownstreamBed.Vertexes[1].Poin
t), Math.Round(_DownstreamBedElevation, 2).ToString() + " m",
sizewaterway, eText_Alignment.MiddleCenter);

InsertElevation(canvaswaterway, new
XYPT(_plDownstreamBed.Vertexes[0].Point.Midpt(_plDownstreamBed.Vertexes[1]
.Point).x, _DownstreamEnergyElevation),
Math.Round(_DownstreamEnergyElevation, 2).ToString() + " m", sizewaterway,
eText_Alignment.MiddleCenter);

InsertElevation(canvasoverall, _Crest.InsPt, Math.Round(_Crest.InsPt.y,
2).ToString() + " m", sizeoverall, eText_Alignment.MiddleCenter);

InsertElevation(canvasoverall, _Crest.PointOfTangency,
Math.Round(_Crest.PointOfTangency.y, 2).ToString() + " m", sizeoverall,
eText_Alignment.MiddleCenter);

InsertElevation(canvasoverall,
_plWaterwayBottomProfile.Vertexes[_plWaterwayBottomProfile.Vertexes.Count
- 1].Point + new XYPT(1, 0),
Math.Round(_plWaterwayBottomProfile.Vertexes[_plWaterwayBottomProfile.Vert
exes.Count - 1].Point.y, 2).ToString() + " m", sizeoverall,
eText_Alignment.MiddleCenter);

InsertElevation(canvasoverall,
_plDownstreamBed.Vertexes[0].Point.Midpt(_plDownstreamBed.Vertexes[1].Poin
t), "DS Bed El." + Math.Round(_DownstreamBedElevation, 2).ToString() + "
m", sizeoverall, eText_Alignment.MiddleLeft);

InsertElevation(canvasoverall, new
XYPT(_plDownstreamBed.Vertexes[1].Point.x, _DownstreamEnergyElevation),"DS
En. El.:" + Math.Round(_DownstreamEnergyElevation, 2).ToString() + " m",
sizeoverall, eText_Alignment.MiddleLeft);
```

```
InsertElevation(canvasoverall, new
XYPT(_plDownstreamBed.Vertexes[0].Point.x, _StillingBasinElevation +
_MaxFlowDepthAfterJump), "Max. Flow Depth:" +
Math.Round(_StillingBasinElevation + _MaxFlowDepthAfterJump, 2).ToString()
+ " m", sizeoverall, eText_Alignment.MiddleRight);

}

private void InsertElevation(Canvas winform, XYPT inspt, string
txtcontent, double size, eText_Alignment txtalignment)

{

Font inputdrawingfont = new Font("Arial", Convert.ToSingle(size));

Text_Style InputDrawingTextStyle = new Text_Style("inputdrawingtextstyle",
inputdrawingfont, txtalignment);

Polyline arrowhead = DrawingHelper.DrawArrowHead(size, 90, inspt);

winform.Document.Entities.Add(arrowhead);

Text txt = DrawingHelper.InsertText(InputDrawingTextStyle, inspt + new
XYPT(0, size * 1.25), txtcontent, txtalignment);

winform.Document.Entities.Add(txt);

winform.Invalidate();

}

private void GenerateInputDrawingTexts()

{

double size = GetAppropriateTextSize(canvasInput);

Font inputdrawingfont = new Font("Arial",Convert.ToSingle(size));

Text_Style InputDrawingTextStyle = new Text_Style("inputdrawingtextstyle",
inputdrawingfont, eText_Alignment.MiddleLeft);


Polyline txtCrestElevationAH = DrawingHelper.DrawArrowHead(size, 90,
_Crest.InsPt);

canvasInput.Document.Entities.Add(txtCrestElevationAH);

Text txtCrestElevation = DrawingHelper.InsertText(InputDrawingTextStyle,
_Crest.InsPt + new XYPT(0, size*1.25), "Crest El.",
eText_Alignment.MiddleCenter);

canvasInput.Document.Entities.Add(txtCrestElevation);

Polyline txtPointOfTangencyAH = DrawingHelper.DrawArrowHead(size, 90,
_Crest.PointOfTangency);

canvasInput.Document.Entities.Add(txtPointOfTangencyAH);
```

```
Text txtPointOfTangency = DrawingHelper.InsertText(InputDrawingTextStyle,
_Crest.PointOfTangency + new XYPT(0, size*1.25), "Point of Tangency El.",
eText_Alignment.MiddleLeft);

canvasInput.Document.Entities.Add(txtPointOfTangency);

Polyline txtStillingBasinAH = DrawingHelper.DrawArrowHead(size, 90,
_plStillingBasinGeometry.Vertexes[0].Point + new
XYPT(_MaximumStillingBasinLength / 2,0));

canvasInput.Document.Entities.Add(txtStillingBasinAH);

Text txtStillingBasin = DrawingHelper.InsertText(InputDrawingTextStyle,
_plStillingBasinGeometry.Vertexes[0].Point + new
XYPT(_MaximumStillingBasinLength / 2, size * 1.25), "Stilling Basin El.",
eText_Alignment.MiddleLeft);

canvasInput.Document.Entities.Add(txtStillingBasin);

Polyline txtDownstreamBedLevelAH = DrawingHelper.DrawArrowHead(size, 90,
new XYPT(_plStillingBasinGeometry.Vertexes[1].Point.x +1,
_DownstreamBedElevation));

canvasInput.Document.Entities.Add(txtDownstreamBedLevelAH);

Text txtDownstreamBedLevel =
DrawingHelper.InsertText(InputDrawingTextStyle, new
XYPT(_plStillingBasinGeometry.Vertexes[1].Point.x +1,
_DownstreamBedElevation + size * 1.25), "Downstream Bed El.",
eText_Alignment.MiddleLeft);

canvasInput.Document.Entities.Add(txtDownstreamBedLevel);

Polyline txtDownstreamEnergyAH = DrawingHelper.DrawArrowHead(size, 90, new
XYPT(_plStillingBasinGeometry.Vertexes[1].Point.x,_DownstreamEnergyElevati
on));

canvasInput.Document.Entities.Add(txtDownstreamEnergyAH);

Text txtDownstreamEnergy = DrawingHelper.InsertText(InputDrawingTextStyle,
new XYPT(_plStillingBasinGeometry.Vertexes[1].Point.x,
_DownstreamEnergyElevation + size * 1.25), "Downstream Energy El.",
eText_Alignment.MiddleLeft);

canvasInput.Document.Entities.Add(txtDownstreamEnergy);

}

public static Polyline DrawArrowHead(double arrowsize, double
angle_degree, XYPT InsPt)

{

Polyline arrowhead = new Polyline();

arrowhead.Vertexes.Add(new PolylineVertex(InsPt));
```

```
arrowhead.Vertexes.Add(new PolylineVertex(new XYPT(InsPt.x + arrowsize *
Math.Cos((45 + angle_degree) * Math.PI / 180), InsPt.y + arrowsize *
Math.Sin((angle_degree + 45) * Math.PI / 180))));

arrowhead.Vertexes.Add(new PolylineVertex(new XYPT(InsPt.x + arrowsize *
Math.Cos((45 - angle_degree) * Math.PI / 180), InsPt.y + arrowsize *
Math.Sin((angle_degree - 45) * Math.PI / 180))));

arrowhead.Vertexes.Add(new PolylineVertex(InsPt));

arrowhead.IsFilled = true;

return arrowhead;

}

public static Text InsertText(Text_Style txtstyle, XYPT InsPT, string
textcontent, eText_Alignment alignment)

{

Text txt = new Text(txtstyle);

txt.InsPt = InsPT;

txt.Text = textcontent;

txt.OverrideAlingment = alignment;

return txt;

}
```

A.7. Events

```
private void BtnCalculate_Click(object sender, EventArgs e)

{

UnsubscribeEvents();

_ShowResults = true;

RemoveAllEntities();

Form2Rec();

GenerateIndicators();

GenerateCrestGeometry();

DrawCrestGeometry(canvasoverall);

PressureCalculations();

CavitationCalculations();

GenerateWaterwayGeometry(_StillingBasinElevation);
```

```
if(chkApproach1.Checked)
{
GenerateCrestWaterSurfaceProfiles();
GenerateWaterwayWaterSurfaceProfiles(_InitialFlowDepthDesign *
Math.Cos(_ChuteAngleRadian), _InitialFlowDepthExisting *
Math.Cos(_ChuteAngleRadian));
GenerateLinkBetweenCrestAndChute();
}
if (chkApproach2.Checked)
{
AeratedChuteCalculations();
}
DissipationCalculations();
DrawCrest();
DrawWaterway();
DrawOverall();
if (tcVisual.SelectedTab == tabpgInput)
{
tcVisual.SelectTab(1);
}
Rec2Form();
InsertAllElevationTexts();
Regen();
SubscribeEvents();
}
private void BtnAdjust_Click(object sender, EventArgs e)
{
UnsubscribeEvents();
StillingBasinElevationAdjustment(_MaxEnergy);
Regen();
SubscribeEvents();
}
```

```
private void BtnResetStillingBasin_Click(object sender, EventArgs e)
{
UnsubscribeEvents();
_StillingBasinElevation = _DownstreamBedElevation;
BtnCalculate_Click(sender, e);
SubscribeEvents();
}
private void cmbCaseSelection_SelectedIndexChanged(object sender,
EventArgs e)
{
UnsubscribeEvents();
switch (cmbCase.SelectedIndex)
{
case 0: //case 1
_DesignDischarge = 250;
_ExistingDischarge = 350;
_SpillwayRoughLength = 25;
_PierNumber = 2;
_PierWidth = 1;
_CrestElevation = 15;
_DownstreamBedElevation = 10;
_DownstreamEnergyElevation = 13;
cmbWaterwayType.SelectedIndex = 0; //chute
_ChuteAngleDegree = 45;
_ChuteAngleRadian = _ChuteAngleDegree * Math.PI / 180;
_Temperature = 30;
break;
case 1: //case 2
_DesignDischarge = 250;
_ExistingDischarge = 350;
_SpillwayRoughLength = 25;
_PierNumber = 2;
```

```
_PierWidth = 1;
_CrestElevation = 15;
_DownstreamBedElevation = 10;
_DownstreamEnergyElevation = 13;
cmbWaterwayType.SelectedIndex = 1; //cascade
_StepHeight = 0.4;
_StepLength = 0.4;
_Temperature = 30;
break;
case 2: //case 3
_DesignDischarge = 1200;
_ExistingDischarge = 1600;
_SpillwayRoughLength = 35;
_PierNumber = 3;
_PierWidth = 1;
_CrestElevation = 70;
_DownstreamBedElevation = 20;
_DownstreamEnergyElevation = 30;
cmbWaterwayType.SelectedIndex = 0; //chute
_ChuteAngleDegree = 45;
_ChuteAngleRadian = _ChuteAngleDegree * Math.PI / 180;
_Temperature = 30;
break;
case 3: //case 4
_DesignDischarge = 1200;
_ExistingDischarge = 1600;
_SpillwayRoughLength = 35;
_PierNumber = 3;
_PierWidth = 1;
_CrestElevation = 70;
_DownstreamBedElevation = 20;
```

```
_DownstreamEnergyElevation = 30;
cmbWaterwayType.SelectedIndex = 1; //cascade
_StepHeight = 1.00;
_StepLength = 1.00;
_Temperature = 30;
break;
case 4: //case 5
_DesignDischarge = 1200;
_ExistingDischarge = 800;
_SpillwayRoughLength = 35;
_PierNumber = 3;
_PierWidth = 1;
_CrestElevation = 70;
_DownstreamBedElevation = 20;
_DownstreamEnergyElevation = 30;
cmbWaterwayType.SelectedIndex = 0; //chute
_ChuteAngleDegree = 45;
_ChuteAngleRadian = _ChuteAngleDegree * Math.PI / 180;
_Temperature = 30;
break;
default:
break;
}
Rec2Form();
BtnCalculate_Click(sender, e);
SubscribeEvents();
}
private void CmbWaterwayType_SelectedIndexChanged(object sender, EventArgs
e)
{
UnsubscribeEvents();
if (cmbWaterwayType.SelectedIndex == 1)
```

```
{
chkApproach1.Checked = true;
chkApproach2.Checked = false;
lblOutputCascade.Visible = true;
}
else
{
lblOutputCascade.Visible = false;
}
SetWaterwayTypeInputVisual();
BtnCalculate_Click(sender, e);
SubscribeEvents();
}
private void TxtBox_ValueChanged(object sender, EventArgs e)
{
UnsubscribeEvents();
BtnCalculate_Click(sender, e);
SubscribeEvents();
}
private void TxtBoxPierNumber_ValueChanged(object sender, EventArgs e)
{
UnsubscribeEvents();
if (txtPierNumber.Text == "0")
{
chkbxNoPier.Checked = true;
chkbxPierBay.Checked = false;
chkbxAlongPiers.Checked = false;
chkbxNoPier.Enabled = true;
chkbxPierBay.Enabled = false;
chkbxAlongPiers.Enabled = false;
BtnCalculate_Click(sender, e);
```

```
}
else
{
chkbxNoPier.Checked = false;
chkbxPierBay.Checked = true;
chkbxAlongPiers.Checked = true;
chkbxNoPier.Enabled = false;
chkbxPierBay.Enabled = true;
chkbxAlongPiers.Enabled = true;
}
SubscribeEvents();
}
private void DischargeChkbox_CheckStateChanged(object sender, EventArgs e)
{
CheckBox chk = sender as CheckBox;
if (chkbxExistingDischarge.Checked == false &&
chkbxDesignDischarge.Checked == false)
{ chk.Checked = true; }
UnsubscribeEvents();
BtnCalculate_Click(sender, e);
SubscribeEvents();
}
private void ApproachChkbox_CheckStateChanged(object sender, EventArgs e)
{
CheckBox chk = sender as CheckBox;
if (chkApproach1.Checked == false && chkApproach2.Checked == false)
{ chk.Checked = true; }
UnsubscribeEvents();
BtnCalculate_Click(sender, e);
SubscribeEvents();
}
private void PressureChkbox_CheckStateChanged(object sender, EventArgs e)
```

```
{
UnsubscribeEvents();
BtnCalculate_Click(sender, e);
tcVisual.SelectTab(2);
SubscribeEvents();
}
private void btnTables_Click_1(object sender, EventArgs e)
{
frmTables f = new frmTables(_dtCrestGeometryUpstream,
_dtCrestGeometryDownstream, _dtCrestUSWSP, _dtCrestDSWSP,
_dtChuteGeometry, _dtChuteWSPDesign, _dtChuteWSPExisting,
_dtAeratedChuteDesign, _dtAeratedChuteExisting, _dtPreDissipationTable,
_dtPressureNoPier, _dtPressurePierBay, _dtPressureAlongPiers,
_dtCavitation, _dtCascade);
f.ShowDialog();
}
```