DEVELOPMENT OF A 1-D SALT INTRUSION MODELLING PROGRAMME USING PYTHON PROGRAMMING LANGUAGE

LOH RI JIAN

B. ENG (HONS.) CIVIL ENGINEERING

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : LOH RI JIAN

Date of Birth : 31ST JULY 1993

Title : DEVELOPMENT OF A 1-D SALT INTRUSION MODELLING PROGRAMME USING PYTHON PROGRAMING LANGUAGE

Academic Session : SEMESTER II 2016/2017

I declare that this thesis is classified as:

☐ CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)*

☐ RESTRICTED (Contains restricted information as specified by the organization where research was done)*

☑ OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

_______________________	_______________________
(Student's Signature)	(Supervisor's Signature)
(930731-08-5927)	Dr. Jacqueline Isabella Anak Gisen
New IC/Passport Number	Name of Supervisor
Date:	Date:

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of Bachelor of Engineering with Honours Civil Engineering.

____

(Supervisor's Signature)

Full Name : Dr. Jacqueline Isabella Anak Gisen

Position : Senior Lecturer

Date :

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

____

(Student's Signature)

Full Name : Loh Ri Jian

ID Number : AA13230

Date :

DEVELOPMENT OF A 1-D SALT INTRUSION MODELLING PROGRAMME USING PYTHON PROGRAMMING LANGUAGE

LOH RI JIAN

Thesis submitted in fulfillment of the requirements
for the award of the
Bachelor Degree in Civil Engineering

Faculty of Civil Engineering and Earth Resources
UNIVERSITI MALAYSIA PAHANG

JUNE 2017

ACKNOWLEDGEMENTS

First of all, I am grateful to the God for the good health and well-being throughout the entire study.

I would like to express my greatest gratitude to my supervisor Dr. Jacqueline Isabella Anak Gisen for the greatest effort in guiding, giving advices and supports and encouraged me thorough the entire journey of my studies in final year project. Her patience in teaching the theories of this study has eased my development of *SALT* modelling programme.

Next, I wish to thank Dr. Nurul Nadrah Aqilah binti Tukimat for providing me with necessary knowledge in writing thesis. Throughout this final year project, I wish to place my sincere appreciations to the lectures for giving encouragement, support and attention directly and indirectly in Universiti Malaysia Pahang.

I would also like to take this opportunity to thank my family, partner and friends who encouraged me with their best wishes and support. Also, I am extremely thankful to my friend Kevin Wong Liang Ji for the sharing some of his programming expertise.

Last but not least, I would like to thank all who is involved directly and indirectly in ensuring the smoothness of this study. Thank you very much.

ABSTRACT

Salt intrusion model mostly does not standalone and is integrated into 2-D and 3-D hydraulics modelling tools such as *Mike21* and *Delft-3D*. Although the integration looks convenient, the models are very expensive due to its costly license procurement. As an alternative, 1-D salt intrusion model generates a simpler and economical platform for the user to conduct study on salt intrusion in estuaries. The objective of this study are 1) to develop a 1-D analytical salt intrusion modelling programme using *Python* programming language: *SALT* 2) to simulate the longitudinal salinity curve using developed programme and 3) to validate the applicability and reliability of the model. In this study, the *SALT* modelling programme has been developed using *Python* programming language to protect the formulas that can be easily changed in traditional spreadsheet. The core concept of this model adopts the analytical 1-D salt intrusion model developed by Savenije (2005) and Gisen (2015).The existing salt intrusion data of Malaysia estuaries were divided into two sets, one for model validation (Kurau, Perak, Bernam, Selangor, Muar and Endau) from Gisen (2015) and another set for model testing on the newly surveyed Belat Estuary. Based on the comparison between the result obtained from the conventional spreadsheet and *SALT*, the *SALT* modelling programme is indeed reliable to be used for salt intrusion study application as the model performance analyses show high accuracy with average *RMSE* of 1.31 and average *NSE* of 0.98.

ABSTRAK

Kebanyakan model intrusi air masin tidak berdiri sendiri dan disepadukan ke dalam alat pemodelan hidraulik 2-D dan 3-D seperti *Mike21* dan *Delft-3D*. Walaupun integrasi kelihatan selesa untuk digunakan, harga model adalah sangat membebankan disebabkan oleh perolehan lesen yang mahal. Sebagai alternatif, model intrusi air masin 1-D menyediakan platform yang lebih mudah dan jimat untuk menjalankan kajian mengenai intrusi air masin di muara. Objektif kajian ini adalah 1) untuk menghasilkan program model intrusi air masin 1-D dengan menggunakan bahasa pengaturcaraan *Python*: *SALT* 2) untuk mensimulasikan graf kemasinan bujur dengan menggunakan program yang dihasilkan dan 3) untuk mengesahkan kesesuaian dan kebolehpercayaan model. Dalam kajian ini, program pemodelan *SALT* telah dihasilkan dengan menggunakan bahasa pengaturcaraan *Python* bagi melindungi formula yang mudah diubah suai dalam hamparan tradisional. Konsep utama model ini menggunakan model analisis intrusi air masin 1-D yang diperkenalkan oleh Savenije (2005) dan Gisen (2015). Data intrusi air masin yang sedia ada bagi muara Malaysia telah dibahagikan kepada dua set, satu untuk pengesahan model (Kurau, Perak, Bernam, Selangor, Muar dan Endau) dari Gisen (2015) dan satu lagi untuk kajian model di Muara Belat yang baru ditinjau. Berdasarkan perbandingan antara keputusan yang diperolehi daripada hamparan tradisional dan *SALT*, Kerpercayaan program pemodelan *SALT* telah dikesahkan sebagi aplikasi intrusi air masin atas sebab purata *RMSE* dan *NSE* yang bernilai 1.31 dan 0.98 menunjukkan ketepatan tinggi analisis prestasi model.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

kg/m^3	kilogram per cubic metre
km	kilometre
m^{-1}	per metre
m	metre
m^2/s	square metre per second
m^3	cubic metre
ppt	parts per thousand

LIST OF ABBREVIATIONS

a	Cross-sectional convergence length
a_1	Cross-sectional convergence length of the seaward reach of estuary
a_2	Cross-sectional convergence length of the landward reach of estuary
A	Cross-sectional area
A_0	Cross-sectional area at the estuary mouth
A_1	Cross-sectional area at the inflection point
ARI	Average Recurrence Interval
b	Width convergence length
b_1	Width convergence length of the seaward reach of estuary
b_2	Width convergence length of the landward reach of estuary
B	Estuary width
B_0	Estuary width at the estuary mouth
B_1	Estuary width at the inflection point
D	Longitudinal dispersion
D_0	Longitudinal dispersion at the estuary mouth
D_1	Longitudinal dispersion at the inflection point
dx	Step length
E	Tidal Excursion
E_0	Tidal Excursion starting from the estuary mouth
E_1	Tidal Excursion starting from the inflection point
EFDC	*Environmental Fluid Dynamic Code*
GUI	*Graphical User Interface*
h	Estuary depth
h_0	Estuary depth at the estuary mouth
h_1	Estuary depth at the inflection point
H	Tidal range
HW	High Water
HWS	High Water Slack
IDLE	*Integrated Development Learning Environment*
K	Van der Burgh's coefficient
L	Salt intrusion length
LW	Low Water
LWS	Low Water Slack
N	Number of measurement data
NSE	Nash-Sutcliffe Efficiency
Obs	Observed data
Omean	Average observed Data
pip	*Pip Install Package*
PNG	*Portable Network Graphic*
Q_f	Fresh water discharge
RMSE	Root Mean Square Error
S	Steady state salinity
S_0	Steady state salinity at the estuary mouth
S_1	Steady state salinity at the inflection point
S_i	Steady state salinity at HWS, TA and LWS
S_f	Fresh water salinity
SALT	*Salinity AnaLysis Technique*
Sim	Simulated data
TA	Tidal Average
ver.	version
x	Distance

x_1	Inflection point
α	Mixing coefficient
α_0	Mixing coefficient at the estuary mouth
α_1	Mixing coefficient at the inflection point
β	Dispersion reduction rate
β_0	Dispersion reduction rate at the estuary mouth
β_1	Dispersion reduction rate at the inflection point
ε	Phase lag between HW and HWS or LW and LWS

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Estuary is a transition medium of river and ocean which exhibit the characteristics of brackish ecosystem. Existence of this physical productive ecosystem provides excellent potential for habitat of flora and fauna as well as providing flood barrier and pollutant filter in the area (Ibrahim *et al.*, 2008; Savenije, 2012). Besides, rich characteristics of its fertile soil, flat surface terrain, fresh water availability, sources of food and accessible transportation medium have encouraged cities to be built in the estuarine region (Gay and O'Donnell, 2009; Savenije, 2012; Gisen, 2015). In the tropical countries, most of the estuaries are classified as alluvial estuaries; estuaries with movable beds which can be eroded or deposited by sediments (Savenije, 1993a).

However, the estuarine region is often prone to salt intrusion problem which can greatly interfere water resources system in an area. Human interference in estuaries for improvement of own various needs, such as navigation and irrigation, can change the natural river flow and salinity distribution (da Silva Dias *et al.*, 2011). Liu *et al.* (2004) found that the reduction of river discharge after construction of the Feitsui Reservoir in China has increased the annual mean salinity to 4.3 ppt near the Kuan-Du wetland. This explained that decreasing river flow will result the increase of salinity increase in estuary. Savenije (1993a) mentioned that dredging works can directly induce salt intrusion. This influence can be seen in the Pulai River estuary in Johore, where dredging works of 18m deep and excessive shoreline development for navigation purpose have affected the salinity pattern and subsequently affected the estuarine ecosystem (Ibrahim *et al.*, 2008). In the Sg. Selangor, salt intrusion problem has significantly deteriorated the ecosystem such as "*fading fireflies*" issue due to the destruction of the mangrove trees (Hassan and

Hashim., 2011). Hence, da Silva Dias *et al.* (2011) stated that it is necessary to maintain an acceptable salinity gradient to ensure that the estuarine ecosystem is protected.

Salt intrusion study is essential to determine the sufficient amount of river discharge to flush out saline water up to acceptable saline level for water intake purpose. Maximum allowable human consumption of chloride ion content is 0.2 ppt thus it will be a problem if salt intrusion occurs at the water intake zone (SMHB *et al.*, 2000). For this reason, salt intrusion study should be done preliminary before any proposal on the construction of the water intake station to prevent the need of station migration and inefficiency of fresh water pumping.

Different types of 2-D and 3-D modelling software have been developed to study salt intrusion in estuaries. However, these application software are very costly and the modelling process required substantial amount of data. As an alternative, selection of one-dimensional modelling allows users to apply a simpler and economical salt intrusion modelling application in alluvial estuaries (Savenije, 1992). Nguyen and Savenije's (2006) one dimensional analytical model showed good performance in simulating salinity distribution model even in stratified neap tide condition with low relative error of 4.6% to 5.2% .This reliability of the analytical salt intrusion model is also high as it was tested in a real estuary (Mekong Estuary) rather than laboratory set-up (Shaha and Cho, 2009).

1.2 PROBLEM STATEMENT

Salt intrusion is a common phenomenon in estuaries. High tide in conjunction of low fresh water discharge causing salt water to intrude further into the river system. For example in 1977 the Kobat water intake station in Kuantan has encountered salt intrusion problem during the drought period at spring tide. The high saline concentration has reached the maximum acceptable consumption of chloride ion content of 0.2 ppt (SMHB *et al*., 2000). Besides water pumping problem, sudden change of salinity concentration can also lead to the destruction of mangroves and aquatic lives as they are sensitive to the change of salinity level (Hassan and Hashim, 2011).

There are several modelling method is being done to determine the salinity concentration at the estuaries. To perform a comprehensive salinity model, long term data are often needed. However, this requires substantial amount of funding. Moreover, readily available software such as *Mike21* and *Delft3D* modelling systems are very expensive in terms of the license procurement. Hence, one dimensional modelling is a good value-for-money approach to model salt intrusion in estuaries (Savenije, 1993a). This approach can be easily done by performing the computation using spreadsheets. Nevertheless, the formulas generated in the spreadsheet can be easily erased accidentally by the modeller as shown in Figure 1.1. Hence a hidden coding is essential to prevent this mistake.

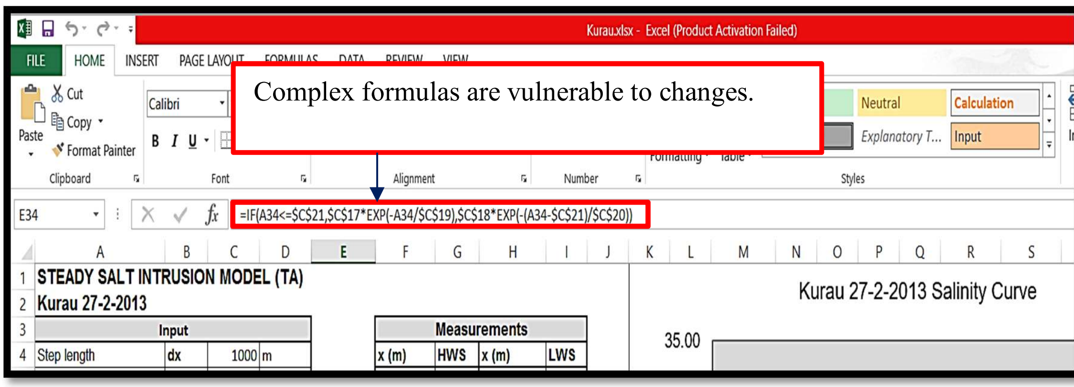


Figure 1.1 Complex formulas are vulnerable to changes

Sources: Savenije (2005)

1.3 OBJECTIVES

This study was meant to achieve the following objectives:

1. To develop a one-dimensional analytical salt intrusion modelling programme using *Python* programming language: *SALT*.

2. To simulate the longitudinal salinity distribution using developed programme.

3. To validate the applicability and reliability of the model.

1.4 SCOPE OF STUDY

This study focused on the development of an open access one-dimensional salt intrusion modelling programme: *SALT*. This modelling programme was created using *Python* programming language. The theory behind this study is based on the steady state one-dimensional analytical salt intrusion model at tidal average (*TA*) condition introduced by Savenije (2005) and Gisen (2015). This modelling programme had been validated by using salinity data for the Malaysian Estuaries by Gisen (2015) to ensure its applicability and validity and also to determine possible bugs and error.

The simulated longitudinal salinity distribution curve produced by the developed programme was fitted against the observation data by calibrating the related parameters such as the initial salinity (S_0), tidal excursion (*E*), Van der Burgh's coefficient (*K*) and dispersion coefficient (*D*). To test the reliability of the data, the results produced in the programme had been validated by comparing it with the previous studies in Malaysia by Gisen (2015). Besides secondary data, new salinity study had be carried out for the Belat Estuary and these new data will also be used for application of this newly developed modelling programme.

1.5 SIGNIFICANCE OF STUDY

Modelling software with fully equipped 2-D and 3-D programming are expensive due to its license procurement which are required to be updated for every few years. Though there are cheaper approach of conducting salt intrusion modelling by using merely excel spreadsheet, the non-encrypted and accessible formulas in the spreadsheet can be accidentally deleted.

Hence, this study converts the formula from the previous studies of steady state one-dimensional analytical salt intrusion model at tidal average condition by Savenije (2005) and Gisen (2015) into encrypted coding script by using *Python* programming language to create an open access modelling programme: *SALT*. This modelling programme has the possibility of integrating with *Graphical User Interface* (*GUI*) to create a user friendly interface for the end-user

CHAPTER 2

LITERATURE REVIEW

2.1 ESTUARY

Estuary can be described as transition medium between sea and river that function as storing and transporting water and sediments (Gisen *et al.*, 2015). It contains its own hydraulic, morphology and biological characteristics such as mixed type tidal waves, funnel shape and a brackish environment (Savenije, 1993a). Due to this unique estuarine habitat, estuary plays a very important role in the flora and fauna's life cycle (Savenije, 2005). Comparison of estuary's characteristics with river and sea is shown in Table 2.1.

Table 2.1 Characteristics of estuary compared to sea and river.

	Sea	***Estuary***	***River***
Shape	Basin	Funnel	Prismatic
Main hydraulic function	Storage	Storage and Transport	Transport of water and sediments
Flow Direction	No dominant direction	Dual direction	Single downstream direction
Bottom slope	No slope	No slope	Downward slope
Salinity	Salt	Brackish	Fresh
Wave Type	Standing	Mixed	Progressive
Ecosystem	Nutrient poor, marine	High biomass productivity, high biodiversity	Nutrient rich, riverine

Source: Savenije (2005).

Combination of driving forces such as tidal influence, wave action, river discharge, littoral sediment transport and gravitational circulation due to salinity and density between sea and river make the classification of estuary based on its shape (Pittaluga *et al.,* 2015; Gisen, 2015). Savenije (2005) summarized the classification of the estuary as in Table 2.2.

Table **Error! Use the Home tab to apply 1ofs to the text that you want to appear here.**.2 Summary on estuary classification.

Shape	***Tidal wave type***	***River influence***	***Geology***	***Salinity***	***Estuarine Richardson number***
Bay	Standing wave	No river discharge	-	Sea salinity	Zero
Ria	Mixed wave	Small river discharge	Drowned drainage system	High salinity, often hypersaline	Small
Fjord	Mixed wave	Modest river discharge	Drowned glacier valley	Partially mixed to stratified	High
Funnel	Mixed wave; large tidal range	Seasonal discharge	Alluvial in coastal plain	Well mixed	Low
Delta	Mixed wave; small tidal range	Seasonal discharge	Alluvial in coastal plain	Partially mixed	Medium
Infinite Prismatic Channel	Progressive wave	Seasonal discharge	Man-made	Partially mixed to stratified	High

Source: Savenije (2005).

2.1.1 Alluvial Estuary

Alluvial estuary is the estuary with movable bed which made up of sediments of riverine and marine origin (Savenije, 1993a). The term alluvium indicates that the estuary bed can be eroded (widening and deepening of the estuary bed) and deposited with sediments (narrower and shallower). Dynamic equilibrium is achieved when the predominance of erosion and deposition occurred consecutively, and it is vital for engineers to derive a universal relationship between estuary's geometry with the hydraulics (Savenije, 1993a; Zhou *et al*., 2012; Pittaluga *et al*., 2015)

On the other hand, estuary with fixed bed (e.g. fjords), where the shape of the estuary cannot be changed by the tidal flow (Savenije, 2005). In this estuary, there is no equilibrium in terms of alluvium as the erosion exceed the deposition rate. Hence, it is not possible to derive relationship between the estuary geometry with the hydraulics (Savenije, 1993a). Savenije (1993b) stated that as long as the estuary is alluvial that agree on the dynamic equilibrium, his analytical one-dimensional model works perfectly on any shape with either density driven or tide driven mixing. Also, it is stated that most estuaries, especially in tropical countries, are alluvial estuaries. Table 2.3 illustrates the interaction between shape and flow in estuaries.

Table 2.3 Interaction between shape and flow in estuaries.

	Shape determine discharge	***Shape does not determine discharge***
Discharge determine shape	Alluvial estuaries	Alluvial rivers
Discharge does not determine shape	Fjords and Rias	Canals and non-alluvial rivers

Sources: Savenije (2005).

2.1.2 Geometry

The shape of tide-dominated estuary is funnel while discharge-dominated estuary is long and narrow (Gisen, 2015). The funnel shape in the seaward part is caused by the wave impact on the estuary mouth, creating an exponential law on the estuary mouth (Pittaluga *et al.*, 2015). Friedrich and Armburst (1998) illustrated the conceptual view of the ideal estuary with decreasing width exponentially as shown in Figure 2.1. Gisen (2015) illustrated the location of inflection point for two reaches estuary in Figure 2.2.

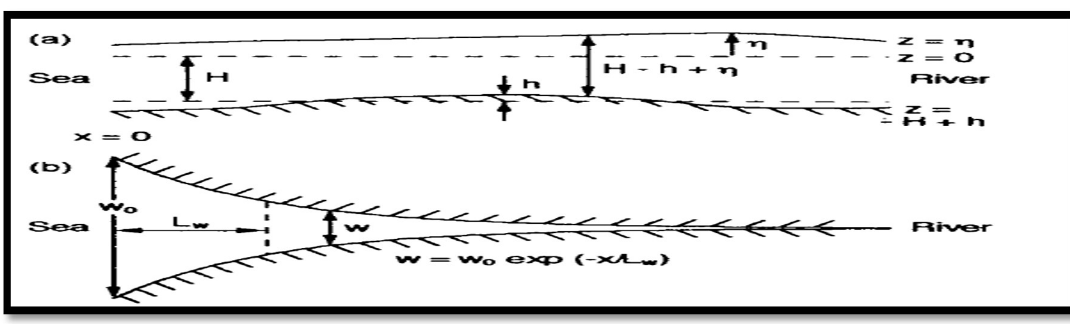


Figure 2.1 Geometry of an idealized tidal estuary: a) side view b) plan view
Sources: Friedrich and Armburst (1998).

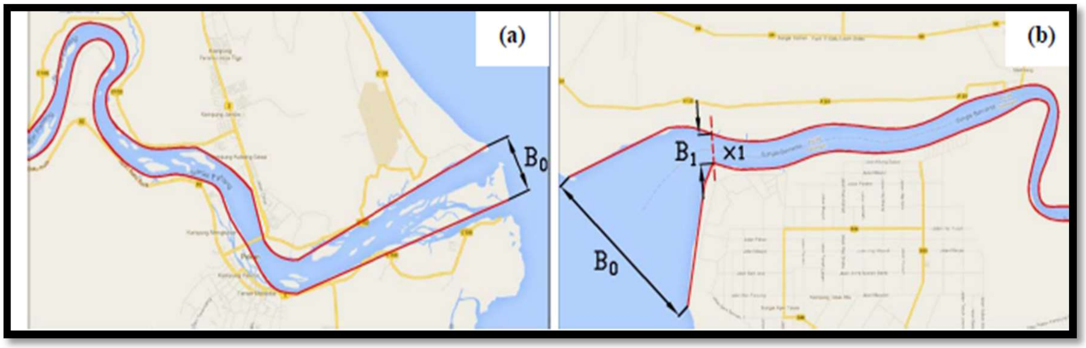


Figure 2.2 (a) shows single reach estuary that do not experience strong wave action. Figure 2.2 (b) represents estuary resulted from strong wave action near the mouth and separate it into two reaches with inflection point x_1.

Sources: Gisen (2015).

Geometry of the alluvial estuaries can be expressed in exponential function (Savenije, 1989, 1993a, 1993b; Grass and Savenije 2008; Nguyen *et al.*, 2012; Gisen *et al.*, 2015). The geometric analysis of the model can be applicable to multi-channel and multi reach estuaries as in Mekong Delta and Yangtze Estuary (Nguyen and Savenije, 2006; Zhang *et al.*, 2011, Nguyen *et al.*, 2012). Estuaries that experience strong tidal waves near the estuary mouth generally have two reaches with two convergence lengths, where the short reaches with short convergence length near the sea and the longer one approaches upstream (Gisen , 2015). Estuaries that does not experience strong tidal waves near the estuary mouth usually are in the form of single reach with one convergence length.

Based on 17 estuaries' studies worldwide, Savenije (2005) claimed that the linear formula that derived from prismatic channel (on the basis of laboratory) works very poorly in natural estuaries, while his proposed model which take account for exponential function in geometry works very well on the alluvial estuaries (Nguyen and Savenije, 2006; Parsa and Etemad-Shahidi, 2011; Zhang *et al.*, 2011; Gisen *et al.*, 2015). Savenije (1993a) claimed the geometric analysis formulas are as follow:

$$A = A_0 e^{-\frac{x}{a_1}} \qquad \text{for } 0 < x \leq x_1 \qquad 2.1$$

$$A = A_1 e^{-\frac{x-x_1}{a_2}} \qquad \text{for } \quad x > x_1 \qquad 2.2$$

$$B = B_0 e^{-\frac{x}{b_1}} \qquad \text{for } 0 < x \leq x_1 \qquad 2.3$$

$$B = B_1 e^{-\frac{x-x_1}{b_2}} \qquad \text{for} \qquad x > x_1 \tag{2.4}$$

$$h = h_0 e^{\frac{x(a_1-b_1)}{a_1 b_1}} \qquad \text{for } 0 < x \leq x_1 \tag{2.5}$$

$$h = h_1 e^{\frac{(x-x_1)(a_2-b_2)}{a_2 b_2}} \qquad \text{for} \qquad x > x_1 \tag{2.6}$$

where A, B and h represent cross-sectional area, width and average depth at distance x, A_0, B_0, and h_0 are the cross-sectional area, width and average depth of estuary mouth, A_1, B_1, and h_1 are the cross-sectional area, width and average depth at the inflection point, a_1 and b_1 are the cross sectional and width convergence length at the estuary mouth.

After the inflection point x_1, the formulas used for the analysis are Equations 2.2, 2.4 and 2.6 with values of A_1, B_1, and h_1 and convergence lengths of a_2 and b_2. In alluvial estuary, convergence lengths of $a_{1,2}$ and $b_{1,2}$ are approximately equal with near constant depth (Savenije, 2005).

2.2 TIDES

Shape of the estuary mouth has a strong dependency with the tidal impact. Depending on the predominant impact (erosion and deposition dominance) along with the magnitude of tidal flows, the existence of the sand bar, spits and barrier islands can be formed (Savenije, 2005; Gisen, 2015). In salt intrusion studies, it became more concerned when there is a small river discharge and tide dominated.

On the other hand, Davies (1964) classified the estuary based on tidal range (H) as follows:

- Micro-tidal estuary: $H < 2$ m; formation of the delta, spits, barrier islands and bar-built estuary, has short convergence (long convergence length a and b).
- Meso-tidal estuary: 2 m $< H <$ 4 m; formation of flood-tide and ebb-tide deltas upstream and downstream of estuary mouth.
- Macro-tidal estuary: $H > 4$ m; formation of funnel shape estuary with strong convergence (short convergence length a and b), does not possess flood-tide and ebb-tide deltas.

Tides can be classified based on tidal period (diurnal, mixed and semi-diurnal). Semi-diurnal consist of two nearly identical tidal cycles and diurnal tides consist of a single tidal cycle in a day (Gisen, 2015). Mixed type tides consisted of one small tidal cycle and a large tidal cycle. The comparison of the tidal range between these two tidal cycles in a day in mixed type is high due to insignificant comparison of the effect of the smaller types with the larger one (Gisen, 2015). Figure 2.3 illustrated the water level oscillation during the tidal cycle for 24 hours.

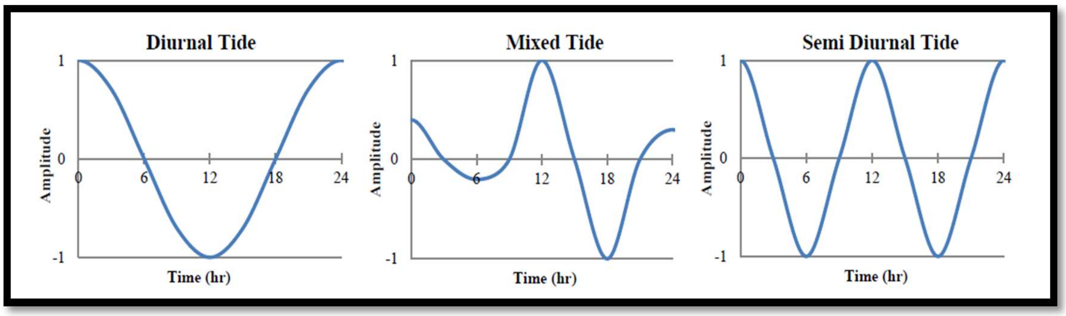


Figure 2.3 Tidal oscillation of diurnal tide (left), mixed tide (middle) and semi diurnal tide (right).
Sources: Gisen (2015).

Other than these classifications, type of waves takes into account for the derivation of the equation. Standing waves only occur in semi-enclosed water bodies. This type of waves reached its maximum water level when the velocity is zero at phase lag of $\pi/2$, as illustrated in Figure 2.4 (a) (Savenije, 2005; Gisen, 2015). Meanwhile, a progressive wave is the wave only occur in prismatic channel with infinite channel length. This type of wave possesses zero phase lag between the water level and velocity of flow as displayed in Figure 2.4 (b). However, these two types of waves do not occur in funnel-shaped alluvial estuary (Savenije 2005; Gisen and Savenije, 2014; Gisen, 2015).

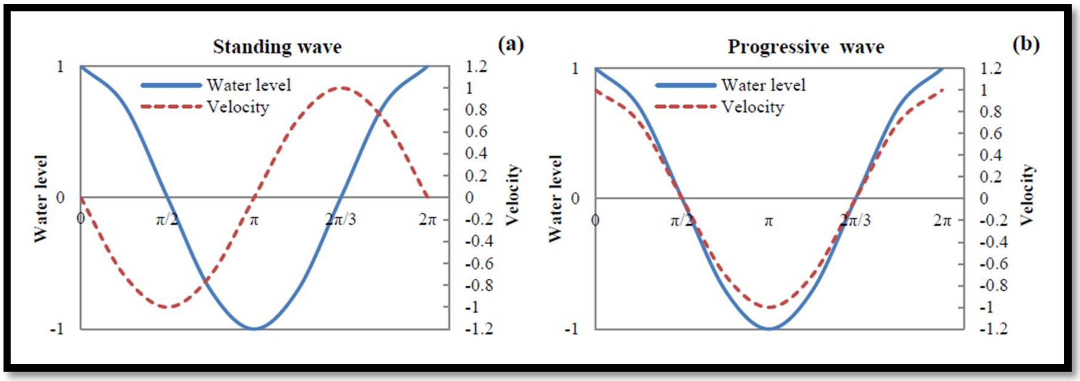


Figure 2.4 Types of tidal waves: a) Standing Wave b) Progressive Wave
Source: Gisen (2015).

Alluvial estuary has a mixed type estuary, where the phase lag is between *0* to *π/2* based on channel geometry and channel bed roughness as shown in Figure 2.5 (Savenije, 2005). In this alluvial estuary, the water level is always reaches highest or lowest point before tidal velocity reach zero (slack moment) (Gisen, 2015). This is a crucial parameter in the tidal dynamics analysis and deriving the analytical one-dimensional salt intrusion model of Savenije (2005) and Gisen (2015).

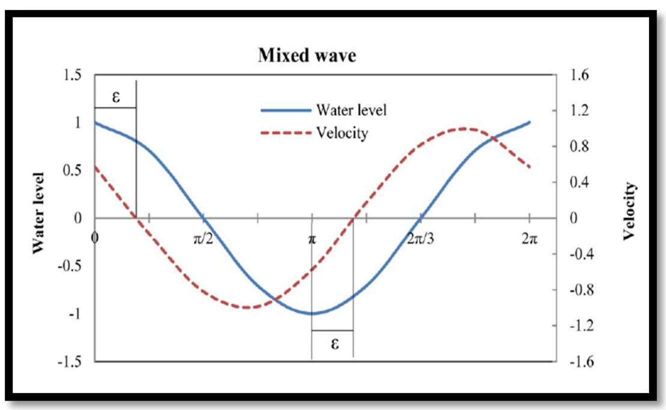


Figure 2.5 Mixed type wave in converging estuary with phase lag ε between HW and HWS together with LW and LWS.
Source: Gisen (2015).

2.3 MIXING

Savenije (2005) explained that the mixing types can be categorized as mixing by turbulence, mixing by tidal shear, mixing by residual circulation, mixing by trapping and mixing by density driven. These mixing mechanisms that constructed longitudinal salinity dispersion concept can be later decomposed into several small constituting fluxes. However, this approach is not practical friendly and hence, Savenije (2005) used the effective longitudinal dispersion as the predictive model of this one-dimensional equation.

Savenije (1993a) found that the tide driven and density driven mixing occurs in the estuary simultaneously. This is true because the alluvial estuary has a strong convergence geometry seaward and prismatic channel landward as illustrated in Figure 2.6. In addition, Gisen (2015) stated that tide driven mixing at the estuary mouth induce small density gradient, but for the region with strong salinity gradient, the density driven characteristic becomes more apparent.

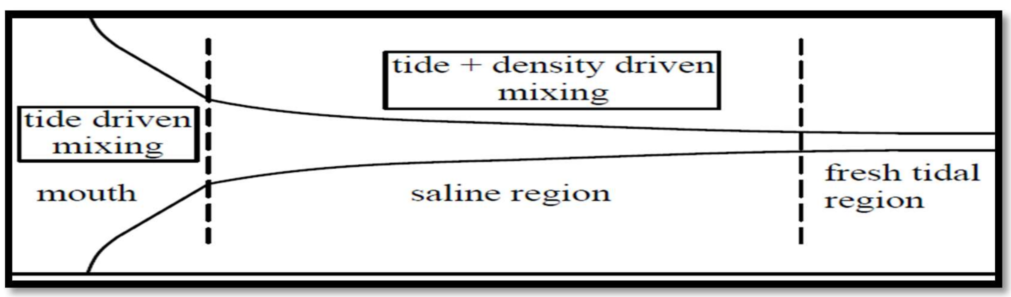


Figure 2.6 Region that dominated by tide and density driven mixing.
Sources: Gisen *et al.* (2015).

2.4 SALT INTRUSION

2.4.1 Type of Salt Intrusion

Salt intrusion mechanism can be classified into three types, which are the salt wedge type, partially mixed type and well mixed type. With the increment of tidal flow along with the decreasing river flow, the type of estuary will make transition along the sequence from highly stratified salt wedge estuary, through the partially mixed estuary to well mixed type estuary (Pritchard, 1967). He stated that the flow in the estuaries shows the Coriolis Effect operating laterally and normal to the direction of flow.

Salt wedge is also known as stratified type (Savenije, 2012). Salt wedge occurred if the river discharge into an estuary, which connected to nearly tide-less sea, such as Sea of Japan or the Mediterranean. The fresh water overrides the layer of salt water and allows the salt water to intrude it underneath in the form of a wedge and only occurred close to the mouth of estuary with high river discharge (Fischer *et al.*, 1979; Savenije, 2012).

A well-mixed type salt intrusion occurred when the river discharge is small compared to the tidal flows, especially during dry periods, where the availability of water is the lowest (Savenije, 1993a). However the difference between partially mixed estuary and well mixed estuary is arbitrary (Savenije, 1993a). He stated that the salt intrusion can classified as well mixed when the stratification is less than 10%. In addition, he also mentioned that this value is arbitrary since there are more stratification when salt intrusion is near towards the sea. From this statement, this implied that there is no serious drawback when applying well mixed theory unless the stratification reached 20% to 30%.

Limitation of this analytical approach of one-dimensional salt intrusion model of Savenije (2005) and Gisen (2015) only applicable to partially mixed and well mixed estuaries at steady state condition. Figure 2.7 shows the longitudinal distribution of the salinity for salt wedge, partially mixed and well mixed estuary. Figure 2.8 shows the variation of the salinity over the depth in salt wedge estuary, partially mixed estuary and well mixed estuary.

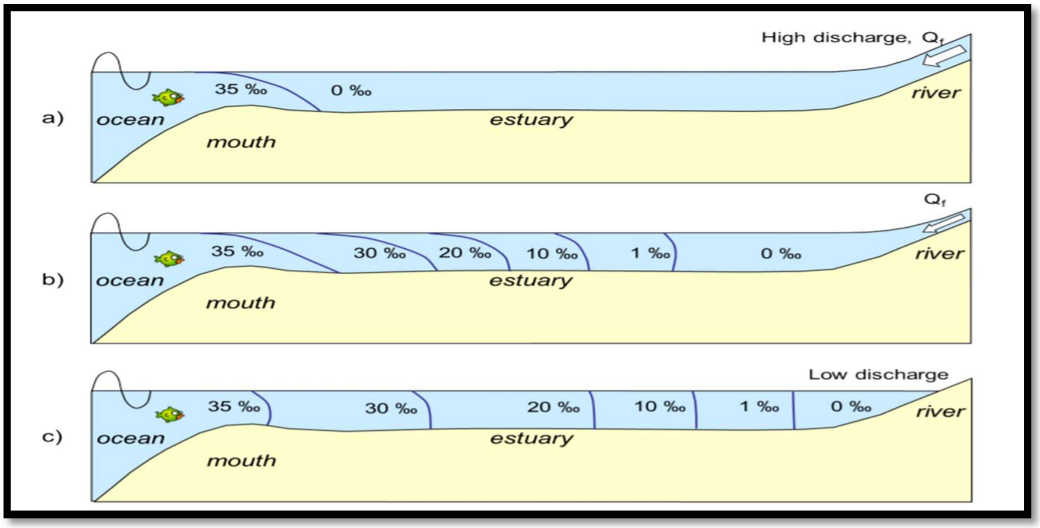


Figure 2.7 Longitudinal distribution of the salinity for (a) salt wedge estuary, (b) partially mixed estuary and (c) well mixed estuary.

Source: Savenije (2015).

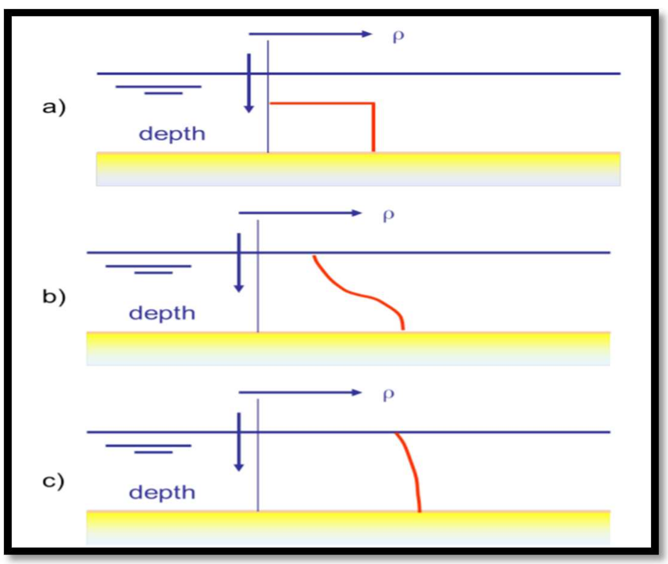


Figure 2.8 Variation of the salinity over the depth in (a) salt wedge estuary, (b) partially mixed estuary and (c) well mixed estuary.

Source: Savenije (2005).

2.4.2 Shape of Longitudinal Salinity Distribution

In well mixed estuary, the longitudinal salinity distribution shows a gradual declining trend in salinity. Smooth curve can be fitted through the observed cross-sectional averaged salinity if a continuous survey is conducted (Savenije, 1993a). However, the shape of this salinity curve can be in various forms depending on the type of estuary. Figure 2.9 is the classification which helps in identifying certain types of salt intrusion based on the longitudinal salinity distribution.

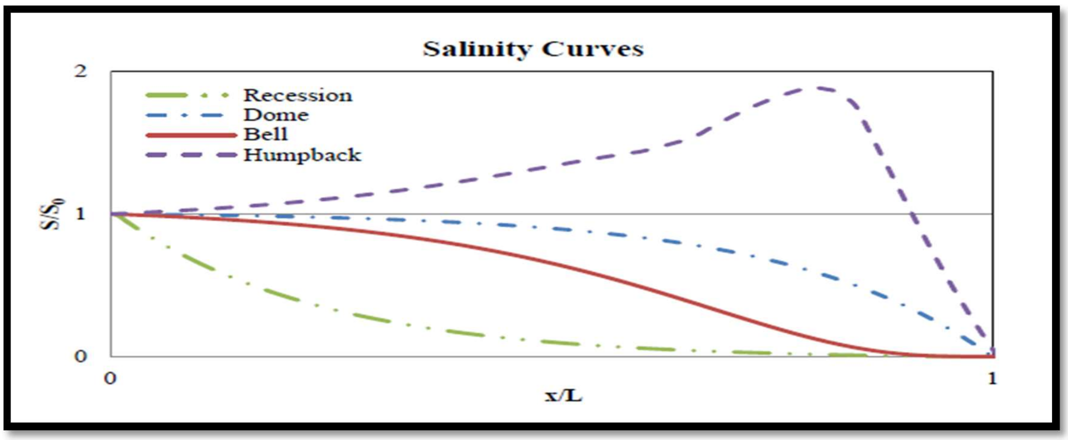


Figure 2.9 Four types of salt intrusion curves
Source: Gisen (2015)

Type 1 salt intrusion curve is a recession curve. The steep salinity gradient at the mouth of estuary indicate the steep and narrow estuary's geometry or a very high volume of river discharge is being received by the estuary (Savenije, 1993a; Gisen *et al.*, 2015). Meanwhile, Type 3 bell curves indicate that the estuary consisted of strongly converged estuary mouth or trumpet estuary's shape while the Type 2 dome curve commonly exist in the strongly funnelled estuary with narrow upstream.

However, Type 4 salt intrusion curve is a special exceptional case in the shape of salinity profile. The geometry of it does not affect the shape of humpback in Type 4 salt intrusion curve, where the salinity ratio (S/S_0) shows increasing value instead of decreasing. This exceptional case resulted from the rainfall deficit or evaporation excess occurred in the estuary. An evaporation can transform a bell shape salt intrusion curve into a dome shape and soon after become humpback hypersaline curve (Savenije, 1993a). However, this salt intrusion curve is not the concern of this study.

2.4.3 Factors of Salt Intrusion

There are several natural phenomena that can cause salt intrusion such as deficit rainfall during the dry season, topography, sea level rising and wind inducing wave (Tran and Tran, 2011). Nevertheless, salt intrusion can also be enhanced by human activities. The primary cause of inducing salt intrusion is the over-pumping of the fresh water, thus depleting the fresh water table and as a result lowering down the fresh water discharge (EPA, 1973). This over-pumping of freshwater results in backwater effect, where the fresh water is not sufficient enough to counteract the tidal flow inward to the estuary (Md. Mahmuduzzaman *et al.*, 2014). Besides, Savenije (1993a) mentioned that dredging works for the channel can be a cause of salt intrusion event. Freshwater Bayou Channel, a 12 foot deep and bottom width of 125 foot channel in Mexico, constructed in 1968 had caused salt intrusion which later resulted in construction of Freshwater Bayou Lock to prevent the salt water impact that erode further inland (Good *et al*., 1995).

Other than the effect of changes of fresh water discharge and the channel depth influences, tidal effect and the diffusion aspect can result in rapid change in salinity in estuary (Ippen and Harleman, 1961). Human activities that release carbon dioxide and greenhouse gases result in rising temperature that expand the volume of sea water by ice melting and thermal expansion of water (Md. Mahmuduzzaman *et al.*, 2014). The increasing volume of sea water can lead to salt intrusion of the estuary. Savenije (1993a) stated that the characteristics of salt intrusion can directly link to geometry of the estuary. It is because the geometry is affected by the tidal impact at the estuary mouth.

2.5 PREVIOUS CASE STUDIES

Salt intrusion analysis can be performed by many available software worldwide, either one, two or three-dimensional analysis. To select the conceptual model of choices, evaluation of the model is needed.

Liu *et al.* (2004) used laterally integrated two-dimensional numerical model of *LINPACK* for salt intrusion study in the Tanshui River Estuary in Taiwan. The study proved that the construction of both Feitshui and Shihman reservoirs at the upstream reach of Tanshui and Tahan Stream have decreased the river discharge that resulted in salt intrusion in the area. However, Liu's model is only applicable to narrow and partially mixed estuary.

In Shanghai, Fu *et al.* (2008) used numerical method of two and three-dimensional incompressible Reynold average Navier-Stroke equation with the assumption of Boussinesq constant and hydrostatic pressure with *MIKE 21 Flow Model FM* of the saline intrusion investigation in the Yangtze River Estuary. The outcome of the study proved that the changes of upstream discharge, tide effect, rising of sea level and bathymetry are the factors causing the salt intrusion while typhoon does not significantly affect the salinity in the area. Gong *et al.* (2012) used three-dimensional baroclinic model *Environmental Fluid Dynamic Code* (*EFDC*) to simulate water level, current and salinity as well as solving free-surface and three-dimensional continuity motion equation for the Modaomen Estuary, one of the eight outlets of Pearl River Delta. This study discovered that the closure of the Hongwan and Hezhou Waterway can reduce salt intrusion by 17% and 19% during spring tides and neap tides respectively. Gong *et al.* (2011) stated that this model required complex geometry and bathymetric information of the estuary, similar to Yangtze Estuary (Fu *et al.*, 2008) and Mekong Delta (Nguyen *et al.*, 2008).

In Malaysia, Van Breemen (2008) applied 3-dimensional numerical model of *Delft3D* to analyze the water extraction effect on salt intrusion in the Selangor Estuary. To obtain the precise and accurate data for simulation in *Delft3D*, several programmes such as *TIDE, RGFGRID, QUICKIN, FLOW, QUICKPLOT, TRIANA, NESTHD1* and *NESTHD2* have to be incorporated and the output needed to be processed by using *MATLAB*. Van Breemen (2008) stated that this method can provide very accurate and promising result, but is time consuming, which is not suitable for small scale estuary. In addition, user is required to have sufficient knowledge and experience in the field in order to simulate the tidal model and waves for the boundary condition of the tidal flow model.

Applying either *Mike 21* or *Delft3D* will be very expensive for short-term salinity studies. Current version of *Delft3D* requires annual subscription of 3375 Euro (about 16000 Ringgit Malaysia) and annual renewal fee of 33750 Euro (about 160000 Ringgit Malaysia). Furthermore, to execute one-dimensional salt intrusion simulation, adopting on one of these software will be a burden on budget.

Waite (1980) used one-dimensional coupled model of salt intrusion and reservoir operation system to conduct salt intrusion analysis on the River Abary, Guyana. This model successfully assisted the reservoir operator to release water in the river for salt intrusion control. This study indicates that the maximum salinity of 1 ppt can be achieved at 40 km from the sea with minimum reservoir storage of 305 x 10^6 m^3 due to unlimited discharge. On the other hand, one-dimensional finite difference explicit scheme numerical method by Lepage and Ingram (1986) showed that there is good agreement between the simulated values with the actual observation and explained that the salinity at Eastmain River estuary changes rapidly due to wind and tidal force at low river discharge. Correspondingly, both case studies that required less field data and act as economic analysis for long term period made one-dimensional salt intrusion model a major advantage.

Zhang *et al.* (2011) tested the applicability of the analytical salt intrusion model by Savenije (2005) in the Yangtze Estuary after the successful application on the Mekong Estuary in Vietnam (Nguyen and Savenije, 2006; Nguyen *et al*., 2008). Yangtze Estuary is one of the biggest multichannel estuary with complex topography. The predictive equation of Savenije (2005) proved the theory derived from single reach channel estuary can be used in multichannel estuary as all the simulated salinity profiles have shown very good fit to the measured data. Furthermore, the geometry results that fulfil the exponential function proved the estuaries have alluvial characteristics. Additionally, this model can use to estimate the distribution of river discharge over a separate channel in the Yangtze Estuary.

In Malaysia, Gisen *et al.* (2015) applied the improvised predictive model of Savenije (2005) in various Malaysia estuaries. The surveyed estuaries in Malaysia are Kurau, Perak, Bernam, Selangor, Muar and Endau. It is claimed that from the site observation, both Selangor and Perak estuaries are partially mixed estuaries while others are the well mixed estuaries. Some of the estuaries exhibit some special characteristics, such as the existence of three drainage sluices in Selangor Estuary, Perak Estuary with sand bars and Endau Estuary with one estuary tributary. These special characteristics of the estuaries can affect the salt intrusion simulation. Gisen (2015) claimed that the one-dimensional analytical model has a very good fit on all the estuaries but due to underestimation of discharge from some parts of drainage basin, dispersion (D_0), mixing number (α_0) and intrusion length (L^{HWS}) did not tally to the observed data. Nevertheless, these six Malaysian estuaries are used as troubleshooting and validation purpose for the development of *SALT* modelling programme. The salinity profiles of all the Malaysian estuaries are shown as in Figure 2.10.

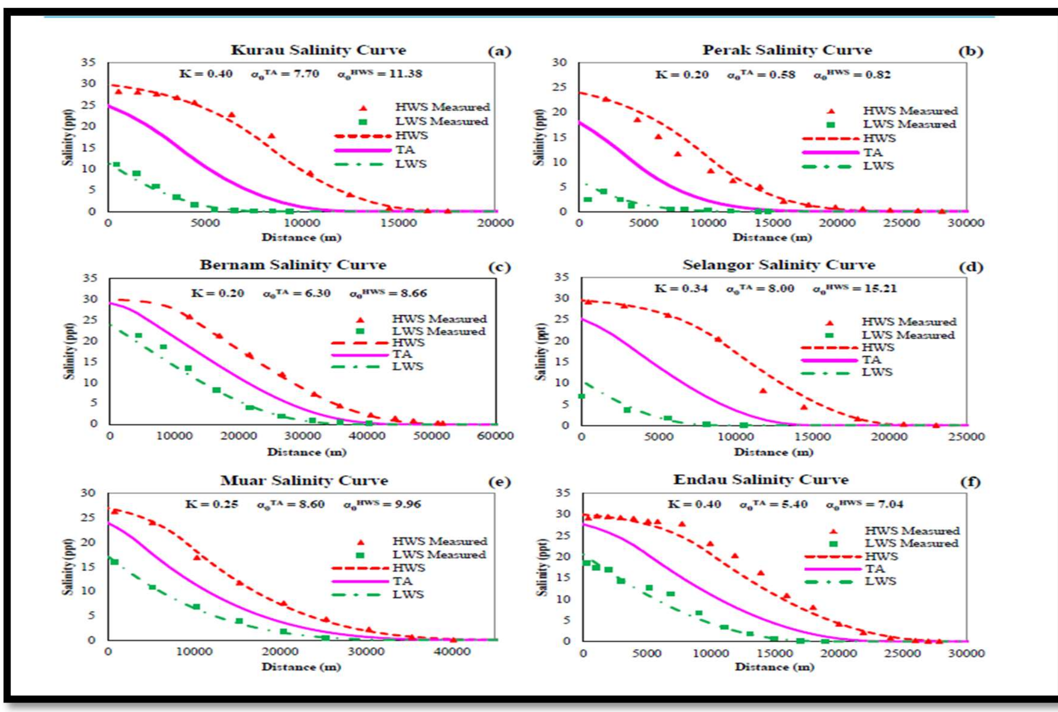


Figure 2.10 Longitudinal salinity distributions in Malaysian estuaries.
Source: Gisen (2015)

Farleigh (1978) performed simulation in the Kuantan Estuary for the proposal of Kobat barrage at the water intake station. The simulation was made by using predicted salinity distribution for 5, 20, and 50 ARI in conjunction with agriculture use and municipal water supply. He used one-dimensional analytical method of Waite (1976) and successfully plotted longitudinal salinity profile as shown in Figure 2.11. However, it is believed that rapid changes by municipal development lead to salt intrusion in April 2016. Salt intrusion occurred during high tide in April 2016 forced the water supply operator to release sufficient water from Chereh Dam (Star, 2016).

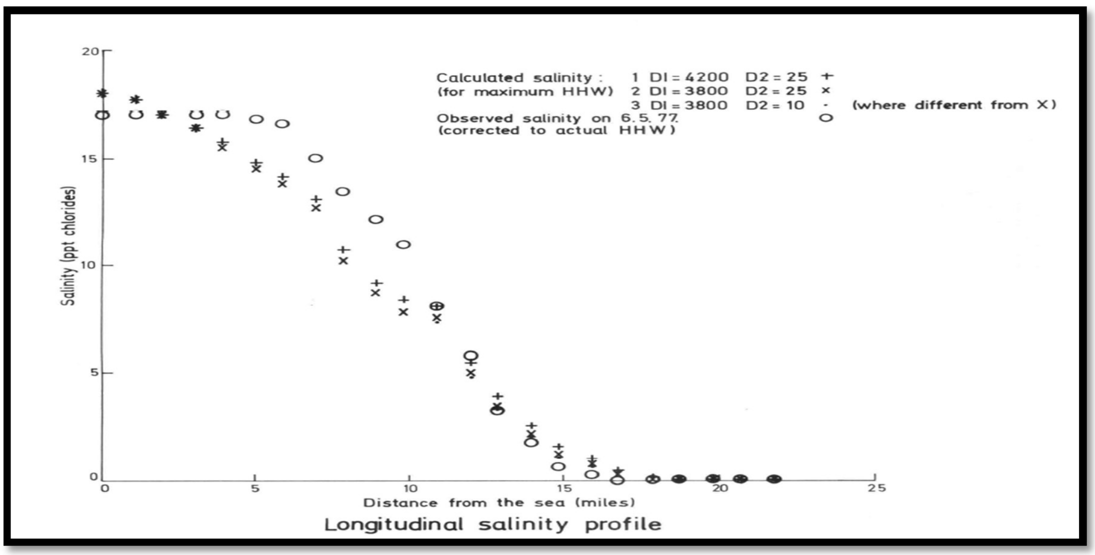


Figure 2.11 Longitudinal salinity distribution of Kuantan estuary in 1978
Source: Farleigh (1978).

2.6 THEORETICAL BACKGROUND OF THE MODEL

Savenije (2005) and Gisen (2015) steady state analytical one-dimensional salt intrusion models at tidal average condition are the basis of this entire study. This model is capable to compute salinity (*S*) at Tidal Average (*TA*), High Water Slack (*HWS*) and Low Water Slack (*LWS*) conditions (Savenije, 2012). This theory involves in three components: geometric analysis, simulating the salinity distribution and calibration process.

As this model is capable to transform from tidal average (*TA*) condition to low water slack (*LWS*) or high water slack (*HWS*) condition, the salinity distribution is simulated by *TA* condition (Gisen, 2015). By integrating Savenije's (2005) model with Van der Burgh's (1972) model, the salinity distribution and the dispersion equation for tidal average (*TA*) condition at steady state condition becomes:

$$\frac{S^{TA}-{S_f}^{TA}}{{S_0}^{TA}-{S_f}^{TA}} = \left(\frac{D^{TA}}{D_0^{TA}}\right)^{\frac{1}{K}} \quad \text{for } 0 < x \leq x_1 \qquad 2.7$$

$$\frac{S^{TA}-{S_f}^{TA}}{{S_1}^{TA}-{S_f}^{TA}} = \left(\frac{D^{TA}}{D_1^{TA}}\right)^{\frac{1}{K}} \quad \text{for } \quad x > x_1 \qquad 2.8$$

$$\frac{D^{TA}}{D_0^{TA}} = 1 - {\beta_0}^{TA}(\exp^{\left(\frac{x}{a_1}\right)} - 1) \text{ for } 0 < x \leq x_1 \qquad 2.9$$

$$\frac{D^{TA}}{D_1^{TA}} = 1 - {\beta_1}^{TA}(\exp^{\left(\frac{x}{a_2}\right)} - 1) \text{ for } \quad x > x_1 \qquad 2.10$$

where S^{TA} and D^{TA} represent the salinity and dispersion coefficient at tidal average (*TA*) condition at a specific location, S_0^{TA} and D_0^{TA} represent the salinity and dispersion coefficient at tidal average condition at estuary mouth, S_1^{TA} and D_1^{TA} represent the salinity and dispersion coefficient at tidal average condition (*TA*) at inflection point (x_1), S_f^{TA} represent fresh water salinity, which is normally close to zero value.

The dispersion reduction rate, β_0 and β_1 are the dispersion rate at the estuary mouth and inflection point (x_1) respectively. This reduction rate is used to calculate dispersion ratio for Equation 2.9 and 2.10. It can be calculated by using the following equation based on the boundary condition:

$$\beta_0{}^{TA} = \frac{Ka_1}{\propto_0{}^{TA} A_0} \qquad \text{for } 0 < x \le x_1 \qquad 2.11$$

$$\beta_1{}^{TA} = \frac{Ka_2}{\propto_1{}^{TA} A_1} \qquad \text{for } \quad x > x_1 \qquad 2.12$$

Calibration is needed for the simulation in order to achieve correct prediction on the salt intrusion analysis. In this steady state analytical one-dimensional salt intrusion model, the calibration factor is Van der Burgh's coefficient (*K*) and the dispersion coefficient (*D*). The coefficient *K* is known as "shape factor" for the tail of the longitudinal salinity distribution with a strong dependency on the geometry (Savenije, 1993a). *K* value ranges from 0 to 1 and is essential for Equation 2.7, 2.8, **Error! Reference source not found.** and 2.12. Also, different estuary has its own Van der Burgh's coefficient (*K*). Dispersion (*D*) estimation and discharge rate of fresh water (Q_f) are very difficult to be obtained on site. Gisen (2015) mentioned that predictive measures of determining fresh water discharge makes one-dimensional approach to be more advantage. Due to this, mixing number (α_0) was introduced as the calibration parameter instead of dispersion (*D*) to relate the relationship between dispersion (*D*) and fresh water discharge (Q_f) (Savenije, 2005):

$$\alpha_0{}^{TA} = \frac{D_0{}^{TA}}{|Q_f|} \qquad \text{for } 0 < x \le x_1 \qquad 2.13$$

$$\alpha_1{}^{TA} = \frac{D_1{}^{TA}}{|Q_f|} \qquad \text{for } \quad x > x_1 \qquad 2.14$$

With the following relation, salt intrusion length *L* can be computed under condition of *D* = 0, yielding the following equations:

$$L^{TA} = a_1 \ln\left(\frac{1}{\beta_0{}^{TA}} + 1\right) \qquad \text{for } \quad 0 < x \le x_1 \qquad 2.15$$

$$L^{TA} = x_1 + a_2 \ln\left(\frac{1}{\beta_1{}^{TA}} + 1\right) \quad \text{for } \qquad x > x_1 \qquad 2.16$$

Since the final objective in this model is obtaining the maximum salt intrusion length in the estuary, the salinity *HWS* condition has to be obtained. In order to calculate the salinity at *LWS* or *HWS* condition, the longitudinal salinity distribution has to be shifted horizontally over x-axis by half of tidal excursion *E/2* or *–E/2* respectively which can be best demonstrated by Figure 2.12 (Savenije, 2005; Deynoot, 2011; Gisen *et al.*, 2015) by the following equations:

$$S^{HWS}(x) = S^{TA}\left(x + \frac{E}{2}\right) \tag{2.17}$$

$$S^{LWS}(x) = S^{TA}\left(x - \frac{E}{2}\right) \tag{2.18}$$

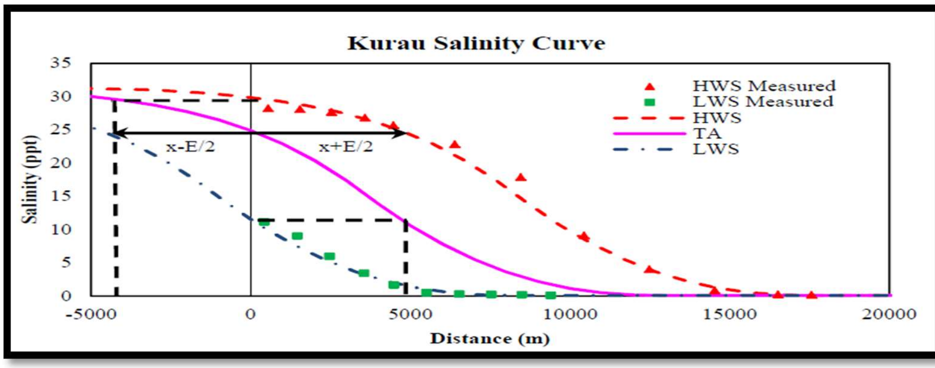


Figure 2.12 Demonstration on the shifting process to obtain the salinity profile for LWS and HWS for better understanding.

Source: Gisen (2015).

From the salinity profile at HWS condition, maximum salt intrusion can be calculated using the following equation:

$$L^{HWS} = a_1 \ln\left(\frac{1}{\beta_0{}^{HWS}} + 1\right) \tag{2.19}$$

2.7 PYTHON PROGRAMMING

Python Programming is a multi-paradigm high-level programming language equipped with a wide range of open-source modules and online database provided by the online communities. Since the creation of the programming language in 1989 by Guido van Rossum, it is widely utilized by the communities to accomplish tasks such as web development, scientific computation, scripting and also *Graphical User Interface* (GUI) development since the language can be interpreted and expressed it out easily (Fritz, 2011). Besides, it is free and can be downloadable without any license issue. Nowadays, there are many famous software utilized *Python Programming* scripting for development such as *ArcGIS*, *FreeCAD*, *ABAQUS*, *Dropbox* and *MODFLOW*. In this study, we use *Python ver. 2.7.13* instead of the latest version of *Python ver. 3.0* due to its version stability and sufficient relevant references.

In comparison with other programming language such as *FORTRAN* and *C++*, it is indeed *FOTRAN* wins in terms of processing speed among all despite as the oldest programming language. Nevertheless, the major advantages of *Python* as an interpreter language with availability of simple development environment and a large open source library supported by the online communities resulting the coding to be corrected and tested easily (Georgatos, 2002). This simple development environment of *Python* also served as its default *Graphical User Interface* (*GUI*), named as *Integrated Development and Learning Environment* (*IDLE*). It also widely used due to its compatibility on many internet protocols and is able to integrate with other languages such as *CPython* (*C++* with *Python*) and *Jython* (*Java* with *Python*). Figure 2.13 shows a very simple but functional *Python* programme typed in *IDLE* editor and Figure 2.14 shows the process of which simple programme is being run by *Python*'s default *GUI* - *IDLE*.

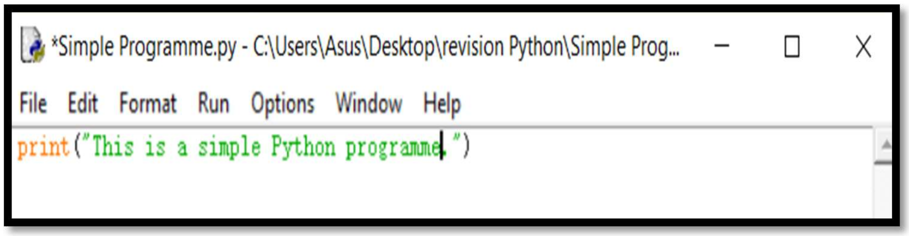


Figure 2.13 A very simple but functional *Python* programme entered in *IDLE* editor

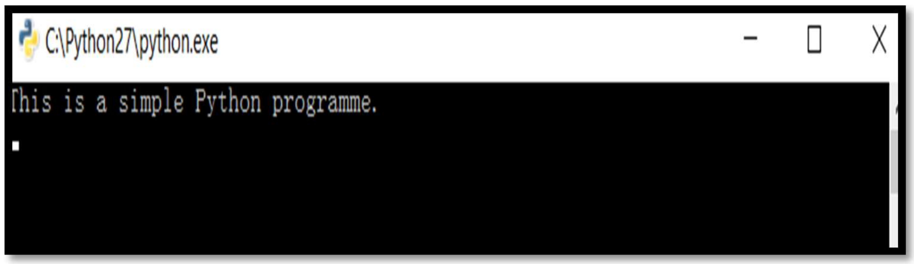


Figure 2.14 Coding from Figure 2.13 is being run by *Python*'s default *GUI* - *IDLE*

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

This chapter describes the methodologies in developing an open access one-dimensional salt intrusion modelling programme named *SALT* (*Salinity AnaLysis Technique*).

First, the overall summary of the entire study is discussed follows by the description on the development of the *SALT* modelling programme conceptual model. Supporting modules that implemented into *SALT* modelling programme is presented in detail. *Python*'s modules are the series of runnable code which define its functions, classes and variables, provided by the *Python*'s online communities.

Next, the concept of 1-D salt intrusion model which is the core of the *SALT* modelling programme is discussed. In order to ensure the programme is able to work appropriately, repetitive trial and error of testing were being done for troubleshooting purpose to identify potential bugs and errors.

After the programme is able to run without error, the simulated result was validated against the existing salt intrusion study in the Malaysian estuaries by Gisen (2015). Also, the effectiveness of *SALT* was evaluated by comparing the simulated output of *SALT* for the Belat Estuary with the conventional spreadsheet method.

Lastly, the reliability and the performance of the model were evaluated by performing the Root Mean Square Error (RMSE) and Nash-Sutcliffe Efficiency (NSE) analyses.

3.2 FLOW CHART OF METHODOLOGY

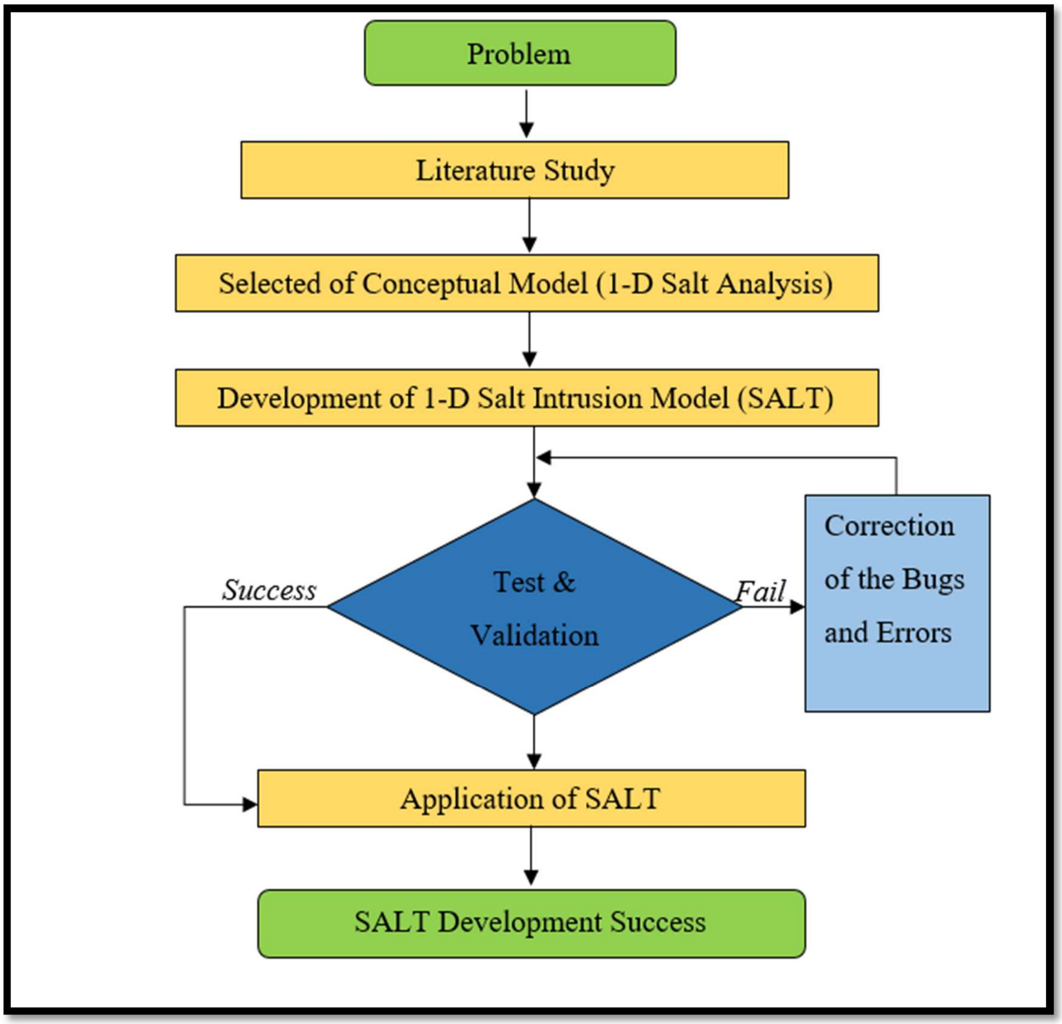


Figure 3.1 Summary of the study for developing *SALT* modelling programme.

Figure 3.1 shows the flow chart for the development of *SALT* modelling programme. With sufficient literature study, suitable conceptual model was decided to be implemented into the development of *SALT*. The theory selected in this study is the steady state one dimensional analytical salt intrusion model at tidal average condition by Savenije (2005) and Gisen (2015). Coding of Python Programming is encrypted to protect the formulas from being altered.

To test the function and reliability of the developed modelling programme, *SALT* was run by using secondary data from previous salt intrusion study done by Gisen (2015) in the Muar Estuary (surveyed on 3rd August 2012). At this stage, correction was done by locating incorrect algorithm in the coding and formulas. Obtaining similar longitudinal salinity distribution indicate no flaws in *SALT* modelling programme.

In order to ensure the developed programme can be used smoothly, *SALT* simulation was executed by running the data from different salt intrusion studies in Malaysian estuaries and the results are compared to the output generated using the conventional spreadsheet. This included the application on the new case study at Belat Estuary.

3.3 DEVELOPMENT PROCESS OF *SALT*

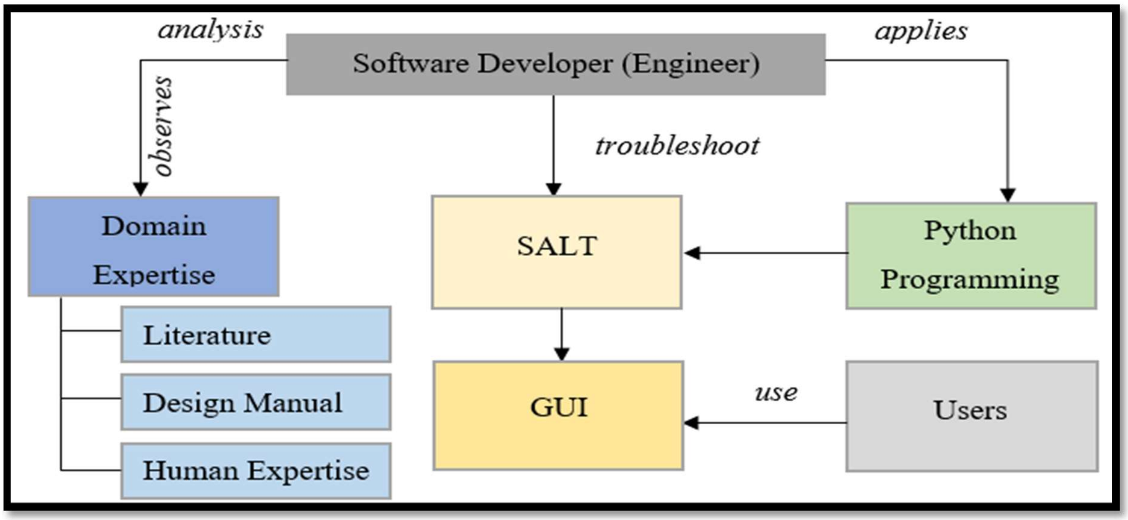


Figure 3.2 Architecture structure of *SALT* modelling programme development.

The development of *SALT* model began with the architecture structure design of the model as shown in Figure 3.2. Software developer or engineer observed the potential theories and methods from literature, existing studies and human expertise. From the knowledge obtained, the *SALT* model was built by selecting suitable mathematical model and programming language. The theory selected to develop the *SALT* model is the steady state one-dimensional analytical salt intrusion model introduced by Savenije (2005) and in reference to the tidal average (*TA*) condition improvised by Gisen (2015). Meanwhile, for the coding, Python Programming was chosen. *SALT* modelling programme was developed by using *Python ver. 2.7.13* to prevent user from accessing the script without prior permission and protect the formulas to be altered.

Additionally, *SALT* modelling programme also utilized different supporting modules to serve for various purposes for better usability. Modules are the series of runnable code which define its functions, classes and variables. It allows easier binding and referencing for software developer with the grouping of related codes. Assuming *foo.py* module that contained variables is needed to be declared, it can be called upon with the coding “import *foo*” (Fritz, 2011).

The four modules that integrated into *SALT* modelling programme are *Math*, *Numpy*, *Plotly* and *PrettyTable*. The *Math* and *Numpy* modules are the *Python*'s built-in modules but *Plotly* and *PrettyTable* modules are the modules provided by *Python*'s online contributor. Function of each module used by *SALT* modelling programme is shown in Table 3.1.

Table 3.1 Modules used by *SALT*

Modules	**Purpose**
Math	- Python's built-in modules that enable formulas inside *SALT* to function correctly. - Example: natural logarithms (log_e) and exponential function (e^x)
Numpy	Python's built-in modules that allow saving array in proper manner that allows *SALT* to read data more easily.
Plotly	Online graphing modules but can be possibly plot in offline mode.
PrettyTable	Enable table to be constructed in well-organised manner.

**Plotly* and *PrettyTable* modules are not *Python*'s built-in modules and have to be installed by using "*pip_install"* in *Command Prompt*.

Since *Plotly* and *PrettyTable* modules are not the built-in modules of *Python*, they had to be installed by "*pip_install"* command in *Command Prompt*. *Pip* (recursive acronym of *Pip Install Packages*) is a package management system that install and manage *Python*'s written modules. First, the module *get-pip.py* that obtained from online sources is needed to be downloaded and executed as shown in Figure 3.3. Then *Path* for the *Python Script* is being added by adding variable value "*C:\Python27\Scripts*" in *Environmental Variables* section in *Computer Properties* of *Windows* shown in Figure 3.4 and Figure Figure 3.5. This allows *pip* to conduct any installation for the other modules without having reference to its full installation path name in *Command Prompt*. Finally, the "*pip_install*" command can only be done after installation of *pip* module along with altering environmental variable for *Python Script*'s *Path* as demonstrated in Figure 3.6.

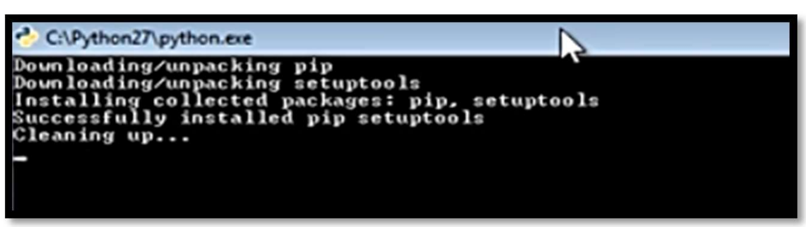


Figure 3.3 Running *get-pip.py* modules to install *pip*

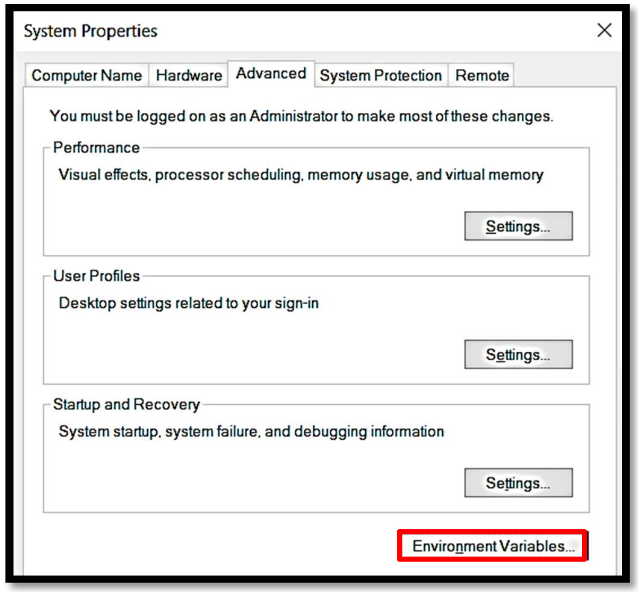


Figure 3.4 *Environment Variables* in *Computer Properties* of *Windows*.

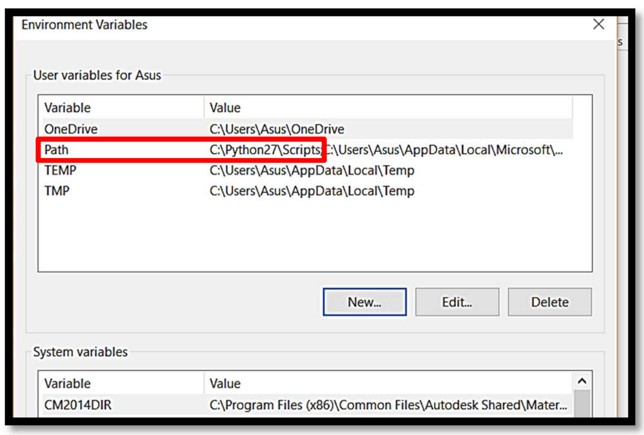


Figure 3.5 Adding *Path* for *Python Script*.

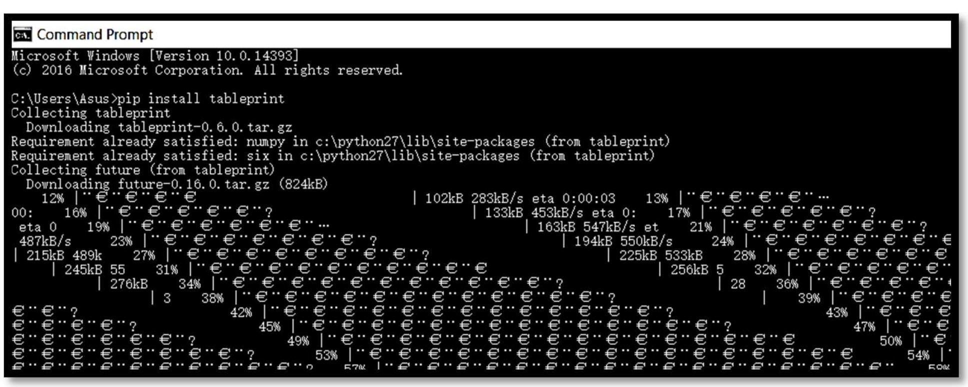


Figure 3.6 Example of execution of "*pip_install*" command.

During the computation process, sometimes there are bugs and errors occurred because of the incorrect algorithm. In order to locate and solve the problems, troubleshooting process were done by repeated checking on bugs in the coding until the computation process works perfectly. To ensure the calculation is performed correctly, a previous salt intrusion study done by Gisen (2015) in the Muar Estuary (surveyed on 3rd August 2012) was taken as reference. The result produced by the *SALT* modelling programme must be the same as the result of Gisen (2015) to confirm that the coding and the formula encrypted works properly.

SALT modelling programme is currently run by using *Python*'s default *Graphical User Interface* (*GUI*) named *Integrated Development Learning Environment* (*IDLE*). Hence, it is possible to be integrated with other *GUI* such as *Tkinter* and *wxPython* for a more user friendly interface for the end-user.

3.4 COMPUTATION PROCESS

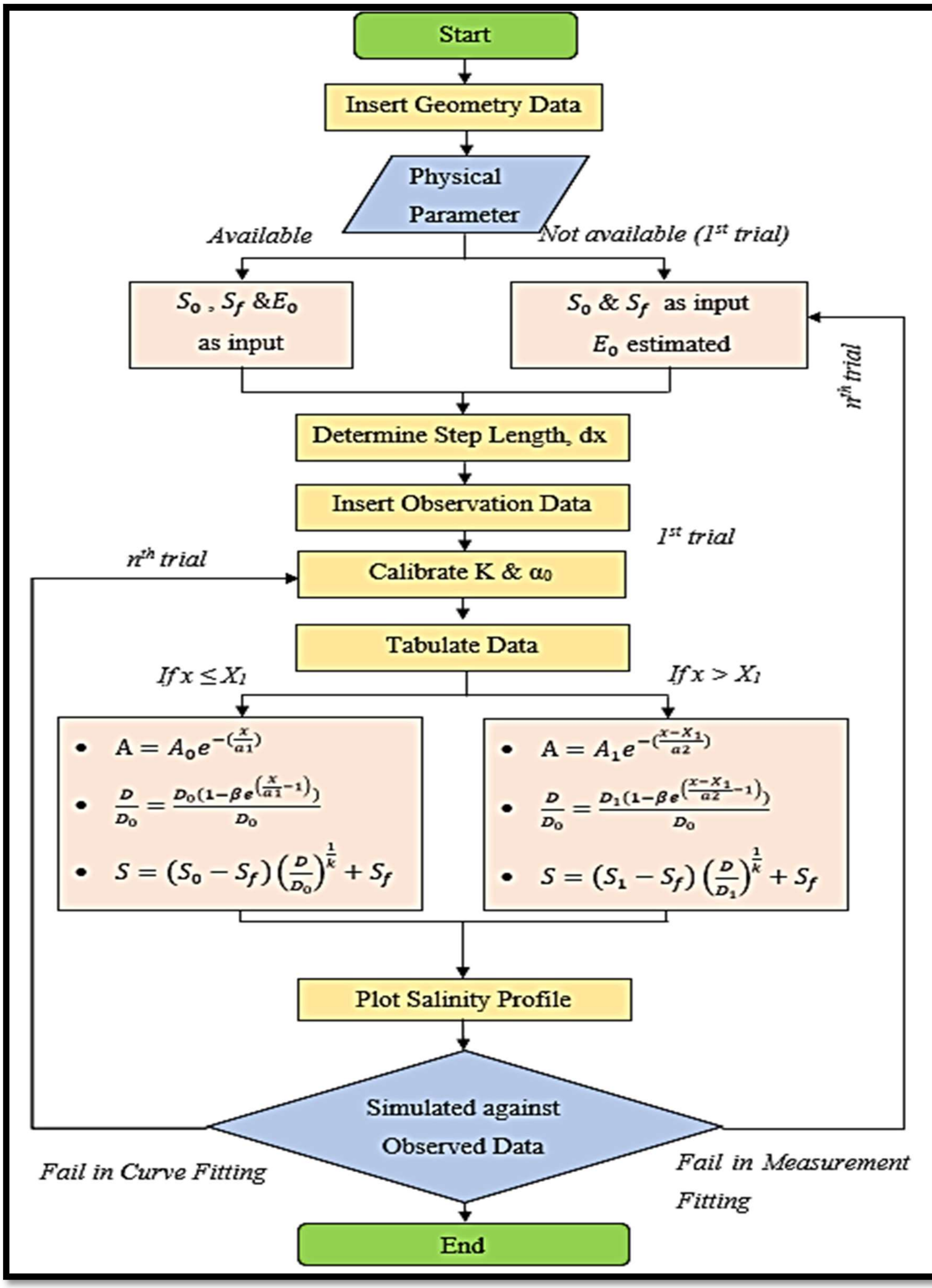


Figure 3.7 Computation Process of the SALT modelling programme.

The first step in developing the core structure of the *SALT* modelling programme is to ensure the formulas or algorithm applied in this programme are hidden.

Figure 3.7 shows the analytical computation process of *SALT* modelling programme. To start this modelling programme, some input data are required for the boundary condition. The first input data required are the geometry parameters. These were predetermined by a separated analysis called the shape analysis. The geometry data required are namely the inflection point (x_1), cross-sectional area at the estuary mouth and inflection point (A_0 and A_1), and area convergence length before and after inflection point (a_1 and a_2), width at the estuary mouth and inflection point (B_0 and B_1), width convergence length before and after inflection point (b_1 and b_2) and average depth h_1.

Next, the physical parameters known as the sea salinity (S_0), fresh water salinity (S_f) and tidal excursion (E_0) were determined. Usually, the sea salinity has the value near to 30 ppt., while fresh water salinity is about 0.1 ppt. In case where the tidal velocity amplitude is not measured, the E_0 has to be calibrated based on tidal envelop.

A suitable step length (*dx*) of the longitudinal salinity profile was defined at the beginning. This information is used to tabulate the data utilized to simulate the longitudinal salinity distribution. The final input data needed is the observed longitudinal salinity along the estuary which were collected during the field survey. This observed data was used to aid the calibration process.

In the salt intrusion model, there are two parameters that cannot be directly measured on site. Thus, they have to be calibrated to fit the simulated salinity curves to the observed data. These parameters are the Van Der Burgh's coefficient (*K*) and dispersion coefficient (*D*). The coefficient *K* also known as "shape factor" controlling the tail of the salinity curve, indicating a strong dependent on the geometry of the estuary (Savenije, 1993a). Savenije (1993a) also explained that *K* ranges from 0 to 1, and is time-dependent. For a start, the first trial for *K* value is generally taken as 0.5. Dispersion on the other hand is a product of mixing salinity of river and sea due to residual circulation induced by gravitational circulation and tidal movement (Gisen, 2015). Since the dispersion is a mathematical artefact and it is always difficult to measure the fresh water discharge, a mixing number (α_0) was introduced as the calibration parameter (Savenije, 2005). The mixing number is the ratio of dispersion (D) over the fresh water discharge (Q_f).

Generally, the salinity analysis can be done for two types of estuary based on the geometry: single convergence and multiple convergence length with inflection point. For that reason, this application is developed to cater both types. The calculation result for the entire process are listed in a table consisting the longitudinal distance from the mouth (x), cross-sectional area at the certain point (A), dispersion (D) and tidal average salinity (S^{TA}). From S^{TA}, the high water slack salinity (S^{HWS}) and low water slack salinity (S^{LWS}) can be obtained by shifting the salinity curve at tidal average for half of the tidal excursion ($\pm E_0/2$) landward and seaward, respectively.

After the computation process are completed, the simulated longitudinal salinity distribution is produced. The result is then compared with the observed data to determine its degree of fitness. If the simulated result deviates from the observed, the calibration parameters have to be adjusted. This process continues until the result fits the observed up to an acceptable level.

3.5 APPLICATION OF THE MODEL

Salt intrusion has become an issue in the Kuantan Estuary when the water intake station located at Kg. Kobat area was affected by saline water. Due to this problem, a salt intrusion study was conducted in 1977 and a barrage is built. However, during the extreme dry season in the early 2016 due to El-Nino phenomenon, saline water was pumped out into the water supply system despite the existence of the barrage.

Based on the problem, review on the performance and reliability of the barrage in accordance to the current salinity condition can be done. However, alternative approach by identifying new location of future water intake stations can be proposed. Since Belat Estuary is the biggest sub-catchment in the Kuantan River Basin, it can become the alternative water sources. Hence, salt water intrusion study is essential to be carried out in the Belat Estuary to identify the intrusion limit. New salinity measurement conducted in April 2017 in the Belat Estuary was taken as observed data for the *SALT* modelling programme.

This also serves as an application to simulate the longitudinal salinity distribution of Belat Estuary. Then, the output longitudinal salinity distribution was compared to conventional spreadsheet for validation process to test the applicability of the developed programme. Error analyses were performed to evaluate the reliability of the developed programme.

3.6 MODEL PERFORMANCE

The performance of *SALT* application model was evaluated by determining the Root Mean Square Error (*RMSE*) and Nash-Sutcliffe Efficiency (*NSE*).

RMSE, also known as root mean square deviation, is the comparison between closeness of the observed value with the simulated one. Lower *RMSE* value indicate desirable closeness of the predicted model to the observed data. *RMSE* is calculated using Equation 3.1:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Obs - Sim)^2} \qquad 3.1$$

where, Obs is the observed discharge and Sim is the simulated discharge.

The *NSE* are used to evaluate the predictive power of this salt intrusion model. Nash and Sutcliffe (1970) suggested that it is necessary to find R^2 value to determine efficiency of the model where this value can determine the linear agreement or disagreement between observed and measured data. The value of *NSE* ranges from negative infinity to 1, where 1 is perfect match of the measured and observed data. Efficiency of 0 indicate the prediction of model equal to mean of the data observation. Negative value of *NSE* indicate mean observed data is a better predictor than the simulated data. Moraisi *et al.* (2007) stated that the accepted values of the *NSE* are in between 0 to 1. NSE is calculated by Equation 3.2:

$$NSE = 1 - \left[\frac{\sum(Obs - Sim)^2}{\sum(Obs - Omean)^2}\right] \qquad 3.2$$

where Obs is the observed discharge, Sim is the simulated discharge and $Omean$ is the mean observed discharge.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 DEVELOPMENT OF *SALT*

SALT modelling programme had been successfully developed using *Python* programming language with integration of several external modules. Repetitive trial and error checking was performed to ensure the programme is able to function properly. The summary of *SALT* modelling programme is shown in Table 4.1.

Table 4.1 Summary of *SALT* modelling programme.

Name	*Salinity AnaLysis Technique* (*SALT*)
Model	Steady State Analytical 1-D Salt Intrusion Model at Tidal Average (TA) Condition by Savenije (2005) and Gisen (2015)
GUI	*Python*'s *Integrated Development Learning Environment* (*IDLE*)
Modules	*Math, Numpy, Plotly, PrettyTable*
Menu	New File, Open File, Save File, List and Edit Input, Generate Result, Help, Exit
Capabilities	Determine salinity intrusion length at HWS Tabulation of data at TA condition Simulate longitudinal salinity distribution Generate Model Performance Analyses

The developed programme is executed by using *Python*'s default *Graphical User Interface* (*GUI*) – *Integrated Development Learning Environment* (*IDLE*) as shown in Figure 4.1. With this base coding of *SALT* modelling programme, it is compatible with any Python's GUI to produce a user-friendly interface to the end user. Menus of *SALT* modelling programme include New File, Open File, Measurement Data, Save File, List and Edit Input, Generate Result, Help and Exit functions. The list of menus in *SALT* are summarized in Table 4.2.

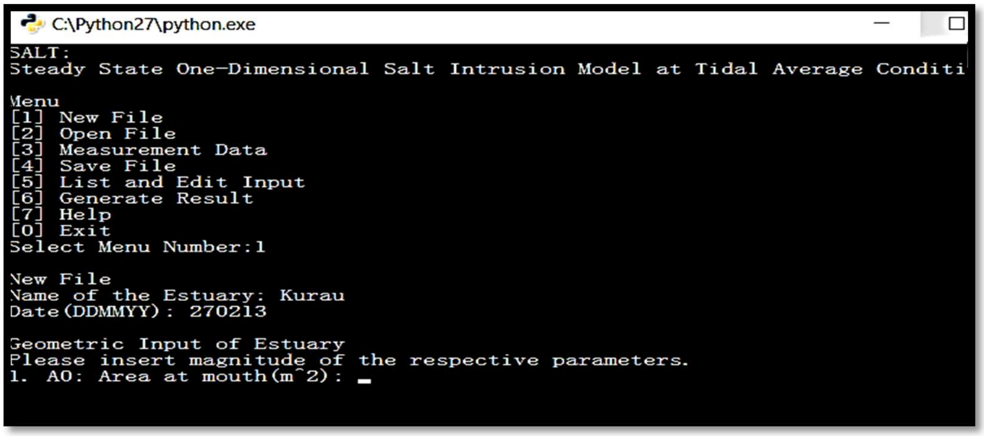


Figure 4.1 *SALT* modelling programme interface using *Python*'s default *GUI – IDLE*

Table 4.2 Functions of Menu in *SALT* modelling programme.

MENU	FUNCTION
New File	To start a new file. Required to insert geometry data, physical parameters and calibration parameters.
Open File	To open a file saved by *SALT* modelling programme with geometry data, physical parameters and calibration parameters.
Measurement Data	To insert observation data of salinity measurement on site for calibration purpose and model performance analyses.
Save File	To save data of estuary studies for Open File in text format.
List and Edit Input	To list all the input including measurement data for edit purpose or calibration purpose.
Generate Result	To generate tabulation and simulate longitudinal salinity distribution. Longitudinal salinity profile open in another window of internet protocol.
Help	To provide help for using *SALT* modelling programme.
Exit	To exit from *SALT* modelling programme.

By performing repetitive of trial and error in checking for the bugs and error in formulas and coding, *SALT* modelling programme now yields the similar longitudinal salinity distribution as in conventional spreadsheet as shown in Figure 4.2. The formulas utilized are based on the analytical 1-D salt intrusion theory by Savenije (2005) and Gisen (2015) and the checking was done by repeated insertion of secondary data of Muar Estuary (surveyed on 3rd August 2012) by Gisen (2015).

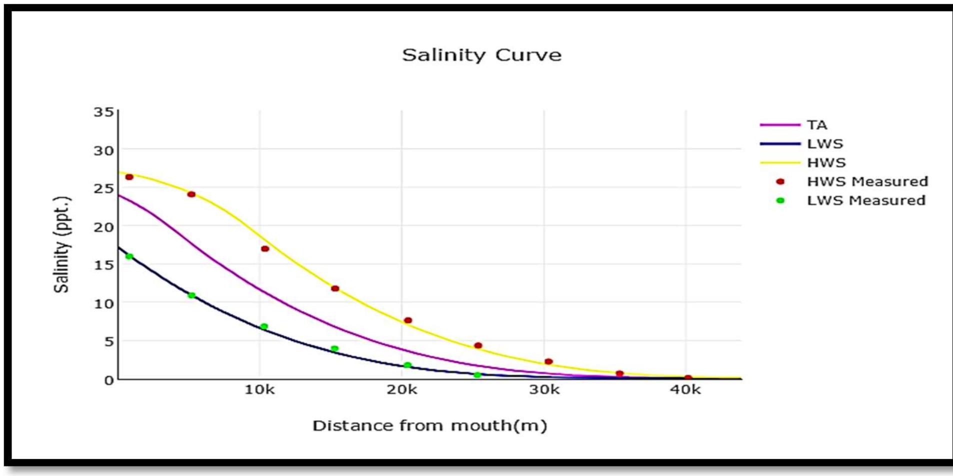


Figure 4.2 Longitudinal Salinity Distribution generated by *SALT* using Internet Protocol

4.2 SIMULATION OF *SALT*

To demonstrate the simulation process of *SALT* modelling programme, the procedures are clearly explained step by step. This allows the end-user to become familiar with the interface of *SALT* modelling programme.

Firstly, *SALT* programme file was executed and New File option was selected to start a new project as shown in Figure 4.3 Starting of the New File option. In this study, secondary data of the Muar Estuary (surveyed on 3rd August 2012) by Gisen (2015) was selected. For a new project, the estuary name and date of measurement has to be addressed in the beginning. Then, the geometry data obtained from the shape analysis was inserted as boundary data. As shown in Figure 4.4, each time after the input was inserted, *SALT* will enquire for the confirmation of input. To re-enter the input, selection [1] is to be chosen, else, by pressing any button, the procedure will proceed to the input parameters and calibration parameters.

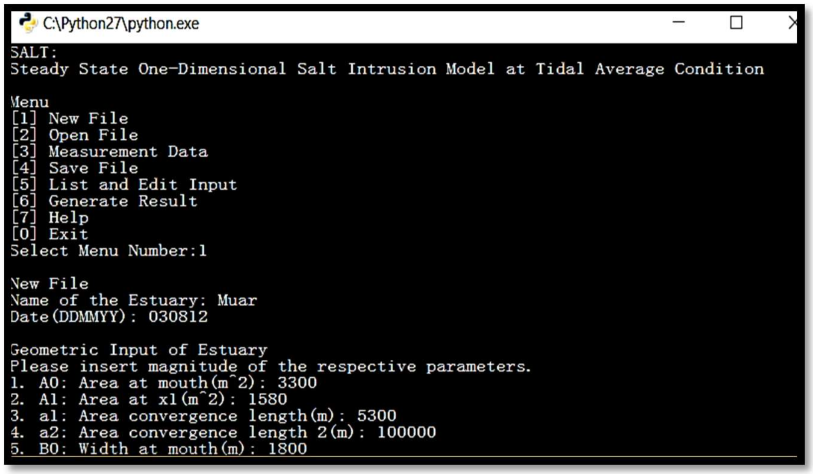


Figure 4.3 Starting of the New File option.

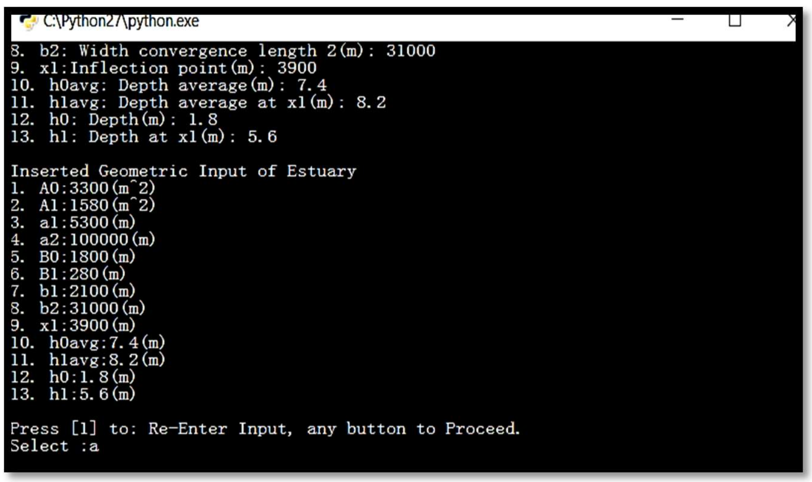


Figure 4.4 Confirmation of the Input.

Then, the magnitude of each input parameter and calibration parameter were declared as explained in Section 3.4. After the calibration, the Tidal Excursion (E_0), Van der Burgh coefficient (K) and mixing coefficient (α_0) obtained are 11000m, 0.25 and 8.6m^{-1} respectively. Sea salinity (S_0) and fresh water salinity (S_f) is set at 24 ppt. and 0.1 ppt. The step length is by default taken as 1000 m and the dispersion (D) to fresh water discharge (Q_f) ratio is determined by calibrating the mixing coefficient (α_0). This end the process of inserting input for New File.

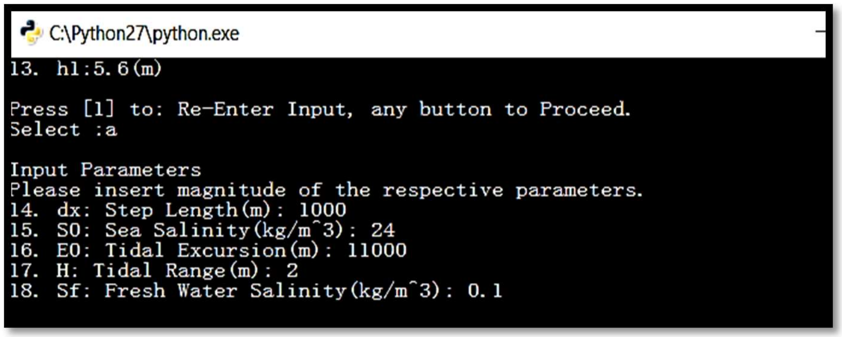


Figure 4.5 Input Parameters of the Muar Estuary.

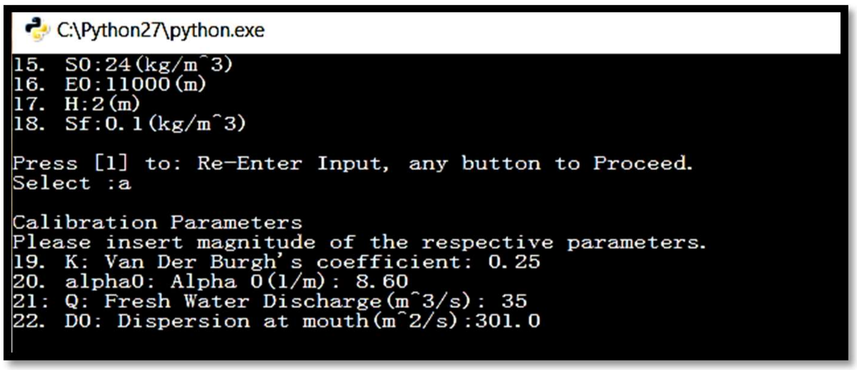


Figure 4.6 Calibration Parameters of the Muar Estuary.

Next, user is required to insert the observed longitudinal salinity measurement for both *HWS* and *LWS* to allow the calibration process as well as computing the model performance analyses. Measurement Data [3] was selected in the *SALT* modelling programme for the observed data input as shown in Figure 4.7. All the inserted data is tabulated in a well-organised manner as shown in Figure 4.8.

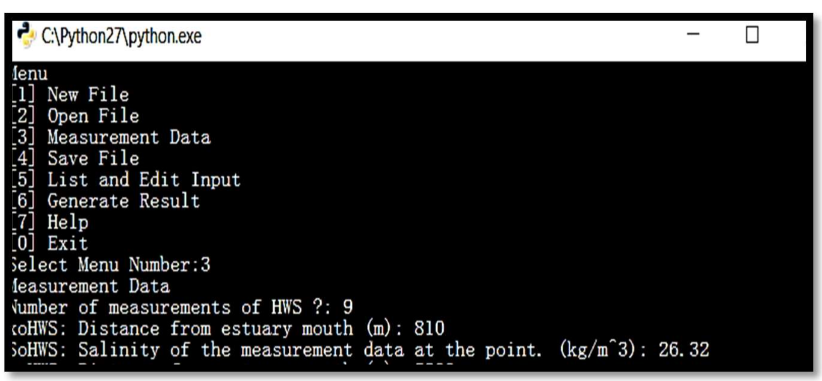
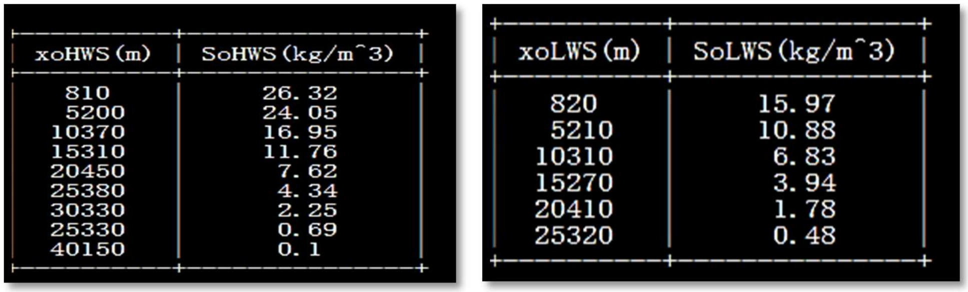


Figure 4.7 Measurement Data is selected and values of observation data of HWS is being inserted.

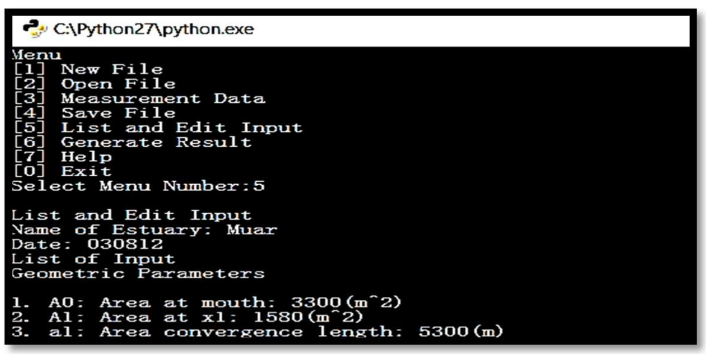


xoHWS (m)	SoHWS (kg/m^3)
810	26.32
5200	24.05
10370	16.95
15310	11.76
20450	7.62
25380	4.34
30330	2.25
25330	0.69
40150	0.1

xoLWS (m)	SoLWS (kg/m^3)
820	15.97
5210	10.88
10310	6.83
15270	3.94
20410	1.78
25320	0.48

Figure 4.8 Tabulation of distance from estuary mouth (x) and salinity (S) at *HWS* (left) and *LWS* (right).

If the user has inserted any parameters wrongly including the observation data, they can edited by reselecting the Measurement Data [3] in the menu. This function is important to ensure all the input data are correct before the simulation is performed. To change the input, user can choose either the abbreviation of the parameters or the coded parameter number. Figure 4.9 displays the interface for the data input editor.

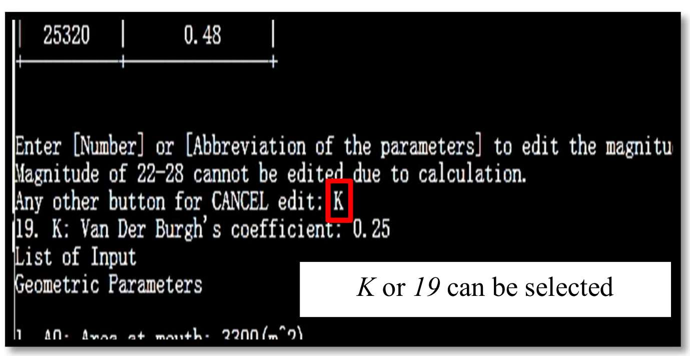


```
C:\Python27\python.exe
Menu
[1] New File
[2] Open File
[3] Measurement Data
[4] Save File
[5] List and Edit Input
[6] Generate Result
[7] Help
[0] Exit
Select Menu Number:5

List and Edit Input
Name of Estuary: Muar
Date: 030812
List of Input
Geometric Parameters

1. A0: Area at mouth: 3300(m^2)
2. A1: Area at x1: 1580(m^2)
3. a1: Area convergence length: 5300(m)
```

Figure 4.9 List and Edit Input.



```
| 25320 | 0.48 |

Enter [Number] or [Abbreviation of the parameters] to edit the magnitu
Magnitude of 22-28 cannot be edited due to calculation.
Any other button for CANCEL edit: K
19. K: Van Der Burgh's coefficient: 0.25
List of Input
Geometric Parameters
```


Figure 4.10 Example of changing *K* value for calibration.

In Figure 4.10, the function to adjust the calibration parameter of *K* value, is presented. User can either select the symbol *K* or *19* in this section for alteration of the calibration parameters to fit the simulated curved to the observed salinity. For "Generate Result" selection, the default iteration is 200, tabulation and simulation were done based on the inserted parameters shown in Figure 4.11. Salinity curve was plotted by opening internet protocol in offline mode shown in Figure 4.12. This generated longitudinal salinity distribution can be save in *Portable Network Graphic* (*PNG*) format .Also, *SALT* generate *RMSE* and *NSE* analyses automatically based on the simulated data and observation data in Figure 4.13 and Figure 4.14.

C:\Python27\python.exe

−1000	3985. 25664829	303. 416389531	24. 7767558032
0	3300. 0	301. 0	24. 0
1000	2732. 57181684	298. 081838653	23. 086560075
2000	2262. 71173763	294. 557710803	22. 0186348945
3000	1873. 64312845	290. 301785364	20. 7790577293
4000	1578. 42078974	285. 166720048	19. 3542798784
5000	1562. 71524047	279. 595394419	17. 8931144303
6000	1547. 16596402	273. 968076037	16. 5033200282
7000	1531. 77140546	268. 284202165	15. 1838539635
8000	1516. 53002532	262. 543204411	13. 9335946233
9000	1501. 44029944	256. 74450867	12. 7513413215
10000	1486. 50071885	250. 887535068	11. 6358143232
11000	1471. 70978956	244. 971697903	10. 5856550815
12000	1457. 06603248	238. 996405586	9. 59942670226
13000	1442. 56798322	232. 961060582	8. 675614656
14000	1428. 21419196	226. 865059353	7. 81262775753
15000	1414. 00322331	220. 707792293	7. 00879943252
16000	1399. 93365616	214. 488643671	6. 26238929416
17000	1386. 00408354	208. 206991565	5. 57158505271
18000	1372. 21311248	201. 862207807	4. 93450478264
19000	1358. 55936388	195. 453657911	4. 34919957335
20000	1345. 04147235	188. 980701019	3. 813656591
21000	1331. 65808609	182. 442689827	3. 32580258029
22000	1318. 40786674	175. 838970532	2. 8835078371
23000	1305. 28948928	169. 168882753	2. 48459068405
24000	1292. 30164186	162. 431759478	2. 12682248325
25000	1279. 44302567	155. 626926989	1. 80793322223
26000	1266. 71235486	148. 753704796	1. 52561771096
27000	1254. 10835634	141. 811405572	1. 27754243003
28000	1241. 62976969	134. 79933508	1. 06135307211
29000	1229. 27534706	127. 716792109	0. 874682821244
30000	1217. 04385299	120. 563068397	0. 715161416797
31000	1204. 93406432	113. 337448566	0. 580425051288

Figure 4.11 Tabulation of the simulated result.

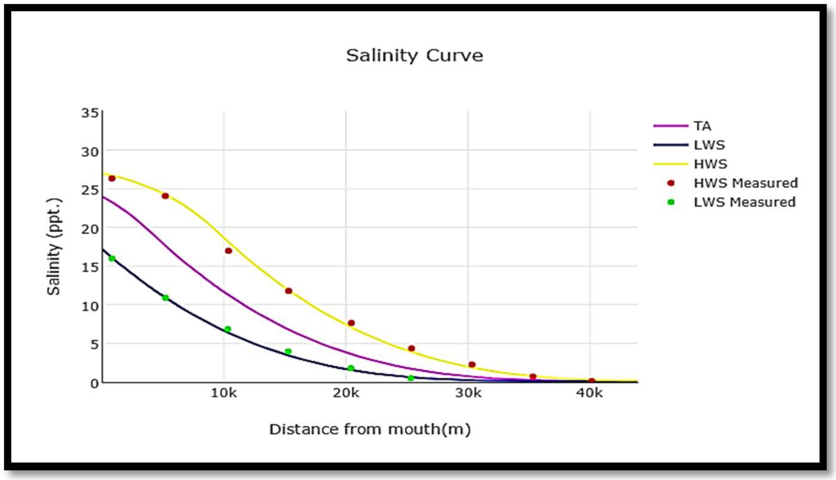


Figure 4.12 Longitudinal salinity distribution of the Muar Estuary.

####Root Mean Square Error, RMSE (HWS)####

n	xoHWS(m)	SmHWS(ppt)	SoHWS(ppt)	(SmHWS-SoHWS)^2
1	810	26.7308695709	26.32	0.168813804296
2	5200	24.2465737161	24.05	0.038641225844
3	10370	18.0790085885	16.95	1.27466039295
4	15310	11.8426918508	11.76	0.00683794218297
5	20450	7.04761240504	7.62	0.327627558867
6	25380	3.87536236366	4.34	0.215888133099
7	30330	1.85950029665	2.25	0.152490018316
8	25330	3.90127666293	0.69	10.3122978059
9	40150	0.261816946476	0.1	0.0261847241668
9			Total Error^2	12.5234416056
			RMSE	1.17961583227

####Nash-Sucliffe Efficiency, NSE (HWS)####

n	xoHWS(m)	SmHWS(ppt)	SoHWS(ppt)	(SmHWS-SoHWS)^2	(SoHWS-SoHWSmean)^2
1	810	26.7308695709	26.32	0.168813804296	251.751111111
2	5200	24.2465737161	24.05	0.038641225844	184.869344444
3	10370	18.0790085885	16.95	1.27466039295	42.2066777778
4	15310	11.8426918508	11.76	0.00683794218297	1.70737777778
5	20450	7.04761240504	7.62	0.327627558867	8.02777777778
6	25380	3.87536236366	4.34	0.215888133099	37.3728444444
7	30330	1.85950029665	2.25	0.152490018316	67.2946777778
8	25330	3.90127666293	0.69	10.3122978059	95.3226777778
9	40150	0.261816946476	0.1	0.0261847241668	107.191511111
9			Total	12.5234416056	795.744
			NSE	0.984261971682	

Figure 4.13 *RMSE* and *NSE* analyses at *HWS* condition of the Muar Estuary.

####Root Mean Square Error, RMSE (LWS)####

n	xoLWS(m)	SmLWS(ppt)	SoLWS(ppt)	(SmLWS-SoLWS)^2
1	820	16.0734912694	15.97	0.0107104428516
2	5210	10.8835544938	10.88	1.26344260128e-05
3	10310	6.39987205268	6.83	0.185010051064
4	15270	3.43389654089	3.94	0.256140711323
5	20410	1.54958907916	1.78	0.0530891924422
6	25320	0.60295383423	0.48	0.0151176453518
6			Total Error^2	0.520080677459
			RMSE	0.2944148653

####Nash-Sucliffe Efficiency, NSE (LWS)####

n	xoLWS(m)	SmLWS(ppt)	SoLWS(ppt)	(SmLWS-SoLWS)^2	(SoLWS-SoLWSmean)^2
1	820	16.0734912694	15.97	0.0107104428516	86.9245444444
2	5210	10.8835544938	10.88	1.26344260128e-05	17.9211111111
3	10310	6.39987205268	6.83	0.185010051064	0.0336111111111
4	15270	3.43389654089	3.94	0.256140711323	7.32604444444
5	20410	1.54958907916	1.78	0.0530891924422	23.6844444444
6	25320	0.60295383423	0.48	0.0151176453518	38.0277777778
6			Total	0.520080677459	173.917533333
			NSE	0.997009613306	

Figure 4.14 *RMSE* and *NSE* analyses at *LWS* condition of the Muar Estuary.

4.3 VALIDATION OF *SALT*

Validation process was done by comparing the simulated result of *SALT* modelling programme with the result obtained from the conventional spreadsheet for the six Malaysian estuaries (Bernam, Endau, Kurau, Muar, Perak and Selangor) by Gisen (2015). The comparison of salinity of TA condition generated by *SALT* against the spreadsheet result was plotted in reference to a perfect agreement line.

Perfect agreement line, also named as line of equality, is the $y = x$ line through the origin at 45 degrees to both axes (Bland and Altman, 2003; Watson and Petrie, 2010). If the results plotted fall on the perfect agreement line, it means that the output of *SALT* model is similar to the conventional spreadsheet and thus certified the correctness of the formulas as well as the result of the entire modelling programme. The validation for all the six Malaysian estuaries are shown below in Figure 4.15 to Figure 4.20.

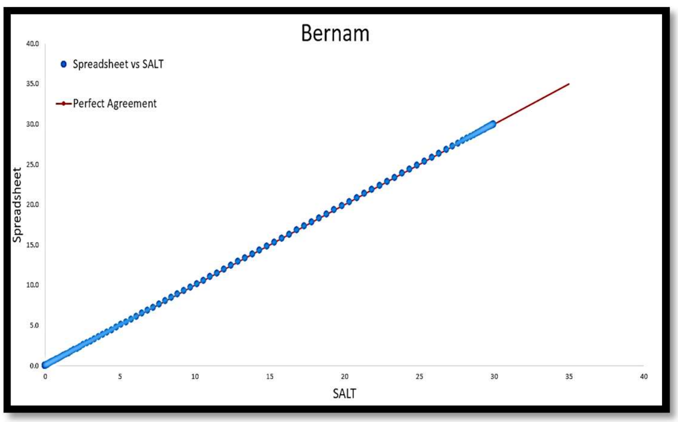


Figure 4.15 Validation of the Bernam Estuary

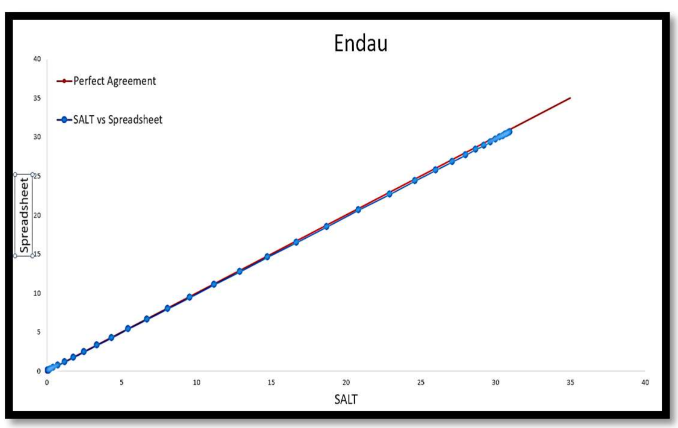


Figure 4.16 Validation of the Endau Estuary

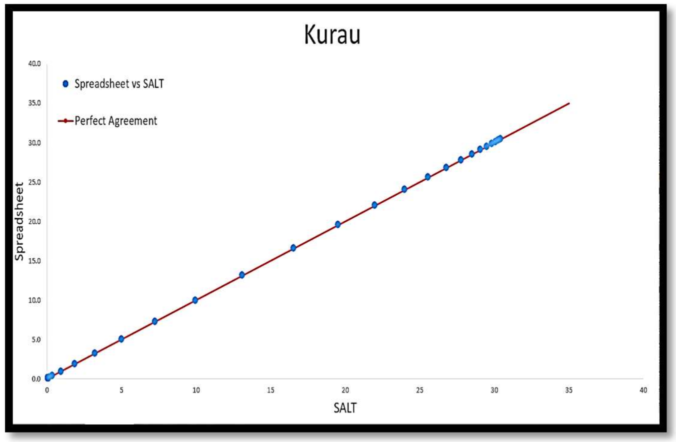


Figure 4.17 Validation of the Kurau Estuary

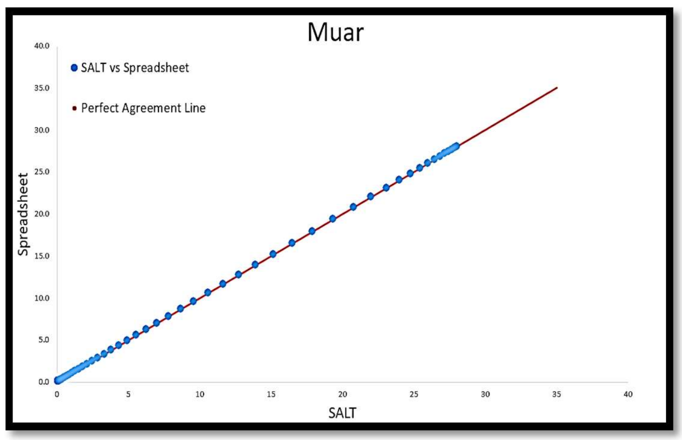


Figure 4.18 Validation of the Muar Estuary

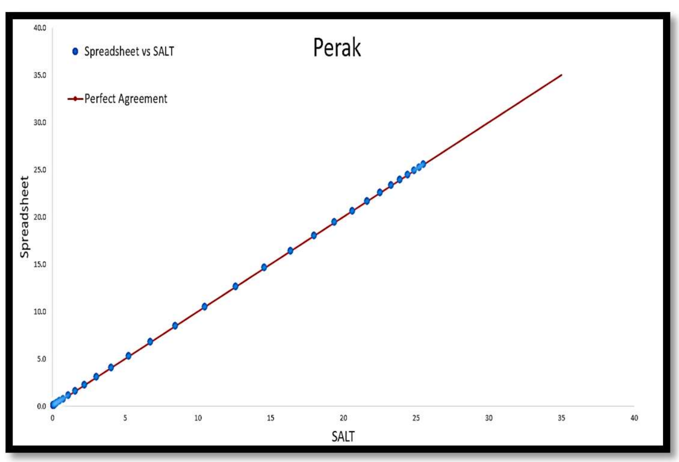


Figure 4.19 Validation of the Perak Estuary

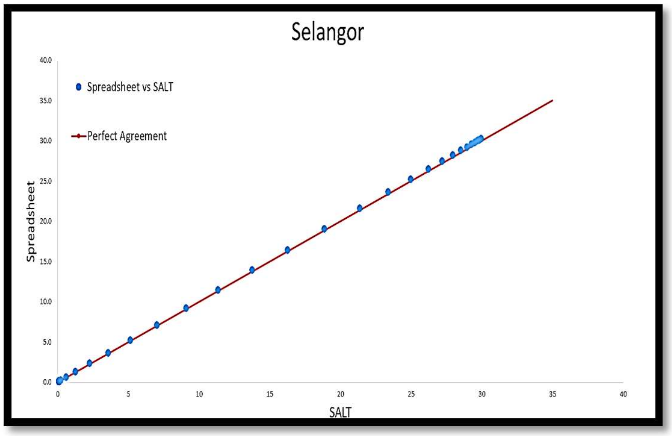


Figure 4.20 Validation of the Selangor Estuary

From all the validation results, there are slight deviations at the end of the perfect agreement line. These deviation values represents the salinity before estuary mouth at tidal average (*TA*) condition. The deviations occured due to the difference in the number of decimal points selected. Nevertheless, the validation of *SALT* against spreadsheet on the perfect agreement line showed a very good fit for the six Malaysian estuary that certify the formulas and the output of *SALT* modelling programme.

4.4 APPLICATION OF *SALT*

The *SALT* modelling programme was applied in the salt intrusion study for the Belat Estuary. Data from the salinity field measurement conducted on 28th April 2017 were used as the observed data in the *SALT* modelling programme. The geometry input and the observations data of Belat Estuary were as shown in Table 4.3 and

Table 4.4.

Table 4.3 Geometry input of Belat Estuary.

Geometry Data	Abbreviation	Magnitude	Units
Area at mouth	A_0	1200	m^2
Area at x1	A_1	1200	m^2
Area convergence length	a_1	20000	m
Area convergence length 2	a_2	20000	m
Width of mouth	B_0	280	m
Width at x1	B_1	280	m
Width convergence length	b_1	20000	m
Width convergence length 2	b_2	20000	m
Inflection point	x_1	0	m
Depth average	$\overline{h_0}$	4.1	m
Depth average at x1	$\overline{h_1}$	4.1	m
Depth at mouth	h_0	4.1	m
Depth at x1	h_1	4.1	m

Table 4.4 Observation data of Belat Estuary.

Measurement			
x_{HWS} (m)	S_{HWS} (ppt)	x_{LWS} (m)	S_{LWS} (ppt)
140	29.99	159	18.920
1349	29.94	1010	15.300
2867	29.73	2741	15.189
463	28.51	4493	9.657
6649	22.36	6689	7.580
8707	20.81	8619	5.580
10685	13.98	9914	4.014

After the calibration process, the calibration parameters, result output as well as the model performance of the Belat Estuary were as shown in Table 4.5. From the simulation, *SALT* modelling programme showed that the salinity intrusion length at *HWS* condition at Belat Estuary is 18 km. Also, the longitudinal salinity distribution of the Belat Estuary generated from *SALT* modelling programme is shown in Figure 4.21. The comparison of the results between *SALT* and from spreadsheet is shown in Figure 4.22 with minor difference where the ratio is close to unity.

Table 4.5 Calibration parameters and output data of Belat Estuary.

Parameters	Abbreviation	Magnitude	Units
Sea salinity	S_0	28	ppt
Tidal Excursion	E_0	6500	m
Van Der Burgh's coefficient	K	0.65	
Mixing Coefficient	α_0	12	m^{-1}
Salinity intrusion length at HWS	L_{HWS}	18161.87	m
RMSE at HWS	$RMSE_{HWS}$	2.17	
RMSE at LWS	$RMSE_{LWS}$	1.88	
NSE at HWS	NSE_{HWS}	0.96	
NSE at LWS	NSE_{LWS}	0.92	

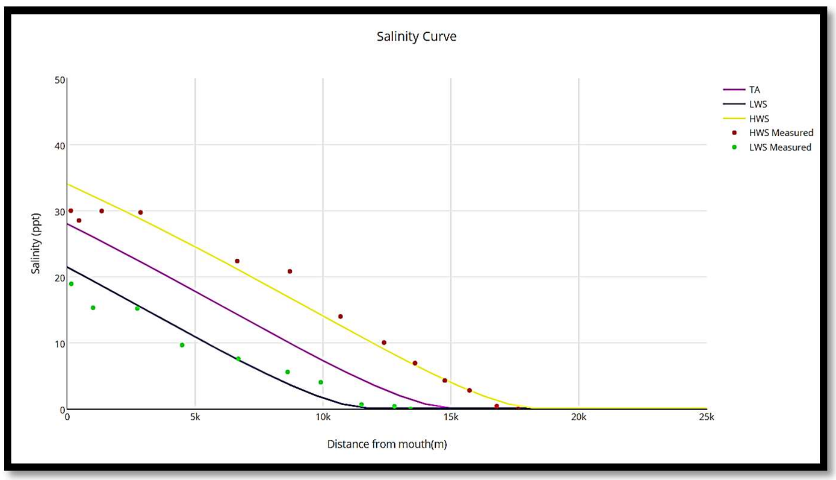


Figure 4.21 Longitudinal salinity distribution of the Belat Estuary.

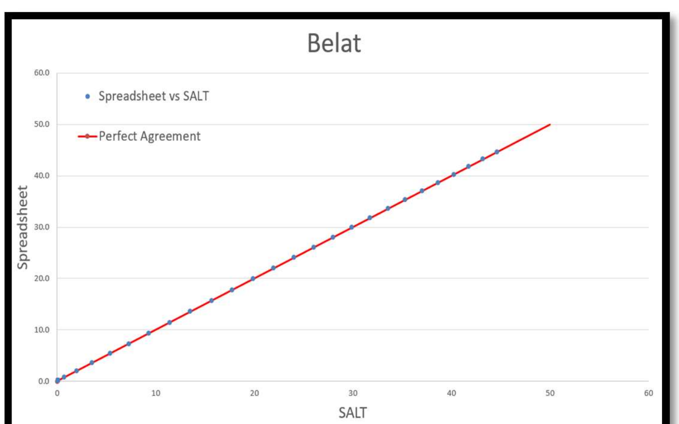


Figure 4.22 Comparison of *SALT*'s output against conventional spreadsheet.

4.5 MODEL PERFORMANCE

Performance of the one-dimensional analytical salt intrusion model in *SALT* modelling programme was evaluated by using Root Mean Square Error (*RMSE*) and Nash-Sutcliffe Efficiency (*NSE*) for assessment of model accuracy and efficiency. The major advantage of the *SALT* modelling programme is its capability to generate the model performance analyses automatically for the *RMSE* and *NSE* values. The summary of all the model performances for the Malaysian estuaries including the Belat Estuary were shown in Table 4.6.

Table 4.6 *RMSE* and *NSE* for all the Malaysian estuaries.

Estuary	***RMSE***		***NSE***	
	HWS	***LWS***	***HWS***	***LWS***
Bernam	0.49	1.34	1.00	0.97
Endau	1.47	1.27	0.96	0.97
Kurau	1.32	0.64	0.99	0.98
Muar	0.50	0.29	1.00	1.00
Perak	1.58	1.14	0.95	0.34
Selangor	1.66	1.69	0.98	0.55
Belat	2.17	1.88	0.96	0.92

In overall, the model performance of Malaysian estuaries shows acceptable *RMSE* values in both *HWS* and *LWS* condition and a very ideal *NSE* value of approximate to 1.00 for both condition. The relatively high *RMSE* values of 2.17 and 1.88 for the Belat Estuary indicated that the simulated result is not as close to the observed data as the other Malaysian estuaries. For the *LWS* in Perak and Selangor, the low *NSE* of 0.34 and 0.55 are still considered as acceptable because the values showed that the simulated value is the better predictor than observed values.

CHAPTER 5

CONCLUSION

5.1 INTRODUCTION

SALT modelling programme has been successfully developed by using *Python* programming language adopting a steady state theory of an analytical salt intrusion model at tidal average condition introduced by Savenije (2005) and improvised by Gisen (2015). Repetitive testing has been done to troubleshoot and eliminate possible bugs and error. The developed programme has been validated by using salinity data of the six Malaysian Estuaries by Gisen (2015) to ensure its applicability and reliability. Then, the model has been applied in the Belat Estuary to test its applicability to simulate salt intrusion cuve in the region.

5.2 CONCLUSION

The objectives of this study have been achieved and are described in the followings:

i) *SALT* modelling programme has been successfully developed using *Python* programming language. The formulas and coding are encrypted to prevent any changes or accidental amendments by end users on the formula. *SALT* modelling programme adopts the *Python*'s default *GUI* - *IDLE* for simulation of salt intrusion profile. The programme comes with the menu of New File, Open File, Save File, Measurement Data, List and Edit Input, Generate Result, Help and Exit.

ii) *SALT* modelling programme has been proved applicable in simulating the longitudinal salinity distribution in all the studied estuaries. The developed programme is able to generate graphical output by displaying the results with internet protocol in offline mode with the aid of *Plotly* module. This graphical output includes the salinity at *TA*, *HWS* and *LWS* condition over a distance *x* from the mouth of estuary.

iii) Validation of *SALT* modelling programme has been done by comparing the output of *SALT* with the conventional spreadsheet method utilizing the salinity data from six Malaysian Estuaries (Gisen, 2015). The output of *SALT* against the spreadsheet were plotted in reference to a perfect agreement line for all the estuaries including the new applied Belat Estuary. Insignificant deviation at the end of the perfect agreement line was examined and this is due to the difference in decimal points. This concludes that *SALT* modelling programme can be used without error.

iv) The model outcomes show that the salinity intrusion length at *HWS* condition for the Belat Estuary is 18 km. Also, the sea salinity, tidal excursion, Van der Burgh's coefficient, mixing coefficient are 28 ppt, 6500 m, 0.65 and 12 m^{-1} respectively. For the model performance, the *RMSE* were 2.17 and 1.88 and *NSE* were 0.96 and 0.92 respectively for *HWS* and *LWS* condition.

v) Based on the model performance analyses, *SALT* is able to simulate the salinity profile accurately with average *RMSE* of 1.31 and average *NSE* of 0.98. The low *RMSE* and high *NSE* value indicated that this model is suitable for Malaysian Estuaries.

5.3 RECOMMENDATION

There are some aspect that have to consider to improve this modelling programme. The followings are the recommendation listed for the future enhancement to this *SALT* modelling programme:

i) This model used Python's default *GUI*. Hence, it does not create a user-friendly interface for the end user to use for salt intrusion simulation. Further integration of this *SALT* model with Python *GUI* such as *Tkinter* and *wxPython* is needed for generation of user-friendly *Graphical User Interface*.

ii) Since the development of *SALT* modelling programme is currently at the early stage of development. Some of the functions are not completely working and have to be decoded. Detailed features that can enhance the function of this modelling programme is encouraged.

iii) This model can be improved by taking into consideration of the integration of 2-D or 3-D salt intrusion model for complex and detailed simulation. If the integration can be done, this model is able to compete with any other simulation programme.

REFERENCES

Bland, J. M., & Altman, D. G. (2003). Applying the right statistics: analyses of measurement studies. *Ultrasound in obstetrics & gynecology*, *22*(1), 85-93.

da Silva Dias, F. J., Lacerda, L. D., Marins, R. V., and de Paula, F. C. F. (2011). Comparative analysis of rating curve and ADP estimates of instantaneous water discharge through estuaries in two contrasting Brazilian rivers. *Hydrological Processes*, *25*(14), 2188-2201.

Davies, L. J. (1964). A morphogenic approach to the worlds' shorelines. *Zeitschrift Geomorphologie* 8:127-142

Deynoot, F. G. (2011). Analytical modeling of Salt Intrusion in the Kapuas Estuary. *Delft University of Technology, Delft.*

Environment Protection Act (1973). *Environmental Protection Section No*. 34 of 1973

Farleigh, D. R. P. (1978). Kuantan River, Malaysia: Prediction of salinity intrusion. *HR Wallingford*.

Fischer, H. B., List, J. E., Koh, C. R., Imberger, J., and Brooks, N. H. (2013). Mixing in inland and coastal waters. *Elsevier.*

Friedrichs, C. T., Armbrust, B. D., and De Swart, H. E. (1998). Hydrodynamics and equilibrium sediment dynamics of shallow, funnel-shaped tidal estuaries. *Physics of estuaries and coastal seas*, 315-327.

Fritz, L. (2011) Balancing cost and Precision of Approximate Type Inference in Python. *Master, Universiteit Utrecht.*

Fu, G., Chen, J. and Jiang, W. (2008). Scenario Studies on the Salinity Intrusion in the Yangtze Estuary.

Gay, P. S., and O'Donnell, J. (2009). Buffering of the salinity intrusion in estuaries by channel convergence. *Hydrology and Earth System Sciences Discussions*, *6*(5), 6007-6033.

Georgatos, F. (2002). How applicable is Python as first computer language for teaching in a pre-university education environment, from a teacher's point of view. *Master, Universiteit van Amsterdam.*

Gisen, J. I. A., and Savenije, H. H. (2015). Estimating bankfull discharge and depth in ungauged estuaries. *Water Resources Research*, *51*(4), 2298-2316.

Gisen, J. I. A. (2015). Prediction in ungauged estuaries. *Delft University of Technology, Delft.*

Gisen, J. I. A., Savenije, H. H. G., Nijzink, R. C., and Abd. Wahab, A. K. (2015). Testing a 1-D analytical salt intrusion model and its predictive equations in Malaysian estuaries. *Hydrological Sciences Journal*, 60(1), 156-172.

Gong, W., Wang, Y. and Jia, J. (2012). The effect of interacting downstream branches on saltwater intrusion in the Modaomen Estuary, China. *Journal of Asian Earth Sciences*, 45, 223-238.

Good, B., Buchtel, J., Meffert, D., Radford, J., Rhinehart, K., and Wilson, R. (1995). Louisiana's major coastal navigation channels. *Unpublished report. Baton Rouge: Louisiana Department of Natural Resources, Coastal Restoration Division*.

Graas, S., and Savenije, H. H. G. (2008). Salt intrusion in the Pungue estuary, Mozambique: effect of sand banks as a natural temporary salt intrusion barrier. *Hydrology and Earth System Sciences Discussions*, *5*(4), 2523-2542.

Hassan, A. J. and Hashim, N. (2011). Salinity intrusion modeling for Sungai Selangor.

Ibrahim, Z., Abdul Latiff A. A., Ab Halim A. H., Abu Bakar N., and Subramaniam S. (2008). Experimental Studies on Mixing in a Salt Wedge Estuary. *Malaysian Journal of Civil Engineering* 20 (2): 188 - 199

Ippen, A.T. and Harleman, D. R. F. (1961). One dimensional analysis of salinity intrusion in estuaries, US Army Corps Eng., Waterways Experiment Station, Visksburg, *Miss. Tech. Bull. No.* 5.

Lepage, S., and Ingram, R. G. (1986). Salinity intrusion in the Eastmain River estuary following a major reduction of freshwater input. *Journal of Geophysical Research*, *91*(C1), 909-915.

Liu, W. C., Hsu, M. H., Wu, C. R., Wang, C. F., and Kuo, A. Y. (2004). Modeling salt water intrusion in Tanshui River estuarine system—case-study contrasting now and then. *Journal of hydraulic engineering*, *130*(9), 849-859.

Mahmuduzzaman, M., Ahmed, Z. U., Nuruzzaman, A. K. M. and Ahmed, F. R. S. (2014). Causes of salinity intrusion in coastal belt of Bangladesh. *International Journal of Plant Research*, *4*(4A), 8-13.

Moriasi, D. N., Arnold, J. G., Van Liew, M. W., Bingner, R. L., Harmel, R. D., and Veith, T. L. (2007). Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. *Trans. Asabe*, *50*(3), 885-900.

Nash, J. E., and Sutcliffe, J. V. (1970). River flow forecasting through conceptual models part I—A discussion of principles. *Journal of hydrology*, *10*(3), 282-290.

Nguyen, A. D., and Savenije, H. H. G. (2006). Salt intrusion in multi-channel estuaries: a case study in the Mekong Delta, Vietnam. *Hydrology and Earth System Sciences Discussions*, *10*(5), 743-754.

Nguyen, A. D., Savenije, H. H. G., Pham, D. N., and Tang, D. T. (2008). Using salt intrusion measurements to determine the freshwater discharge distribution over the branches of a multi-channel estuary: The Mekong Delta case. *Estuarine, Coastal and Shelf Science*, *77*(3), 433-445.

Nguyen, D. H., Umeyama, M., and Shintani, T. (2012). Importance of geometric characteristics for salinity distribution in convergent estuaries. *Journal of hydrology*, *448*, 1-13.

Parsa, J., and Etemad-Shahidi, A. (2011). An empirical model for salinity intrusion in alluvial estuaries. *Ocean Dynamics*, *61*(10), 1619-1628.

Pittaluga, M. B., Tambroni, N., Canestrelli, A., Slingerland, R., Lanzoni, S., and Seminara, G. (2015). Where river and tide meet: The morphodynamic equilibrium of alluvial estuaries. *Journal of Geophysical Research: Earth Surface*, 120(1), 75-94.

Pritchard, D. W. (1967). Observations of circulation in coastal plain estuaries.

Savenije, H. H. G. (1989). Salt intrusion model for high-water slack, low-water slack, and mean tide on spread sheet. *Journal of Hydrology*, *107*(1-4), 9-18.

Savenije, H. H. G. (1992). Rapid assessment technique for salt intrusion in alluvial estuaries.

Savenije, H. H. G. (1993a). Predictive model for salt intrusion in estuaries. *Journal of Hydrology*, *148*(1), 203-218.

Savenije, H. H. G. (1993b). Composition and driving mechanisms of longitudinal tidal average salinity dispersion in estuaries. *Journal of Hydrology*, *144*(1), 127-141.

Savenije, H. H. G. (2005). Salinity and Tides in Alluvial Estuaries. *Elsevier*, New York.

Savenije, H. H. G. (2012). Salinity and Tides in Alluvial Estuaries, http://salinityandtides.com/

Shaha, D. C., and Cho, Y. K. (2009). Comparison of empirical models with intensively observed data for prediction of salt intrusion in the Sumjin River estuary, Korea. *Hydrology and Earth System Sciences*, *13*(6), 923-933.

SMHB, Ranhill and Zaaba (2000). Salinity studies for Sg. Rompin and Sg. Sedili Besar, vol 5 of *Malaysia National Water Resources Study*, DID Malaysia, Malaysia.

Star Malaysia (2016). A little salty, but treated water still potable. Retrieved from: http://www.thestar.com.my/news/nation/2016/04/30/a-little-salty-but-treated-water-still-potable/#WAOjuq8GcVcW2hvW.99

Tran, H. T., and Tran, T. V. (2011). Assessment of climate change impacts on salinity intrusion in Hong-Thai Binh and Dong Nai river basins.

Waite P. J. (1980). Control of salt water intrusion in estuaries by means of a dual purpose reservoir. *IAHS-AISH Publication* no. 129

Watson, P. F. and Petrie, A. (2010). Method agreement analysis: a review of correct methodology. *Theriogenology*, 73(9), 1167-1179.

van Breemen, M. T. J. (2008). Salt intrusion in the Selangor Estuary in Malaysia model—study with Delft3D. *Master*, *University of Twente*.

Zhang, E., Savenije, H. H. G., Wu, H., Kong, Y., & Zhu, J. (2011). Analytical solution for salt intrusion in the Yangtze Estuary, China. *Estuarine, Coastal and Shelf Science*, *91*(4), 492-501.

Zhang, E. F., Savenije, H. H. G., Chen, S. L., and Mao, X. H. (2012). An analytical solution for tidal propagation in the Yangtze Estuary, China. *Hydrology and Earth System Sciences*, *16*(9), 3327.

APPENDIX A
LONGITUDINAL SALINITY DISTRIBUTION OF MALAYSIAN ESTUARIES

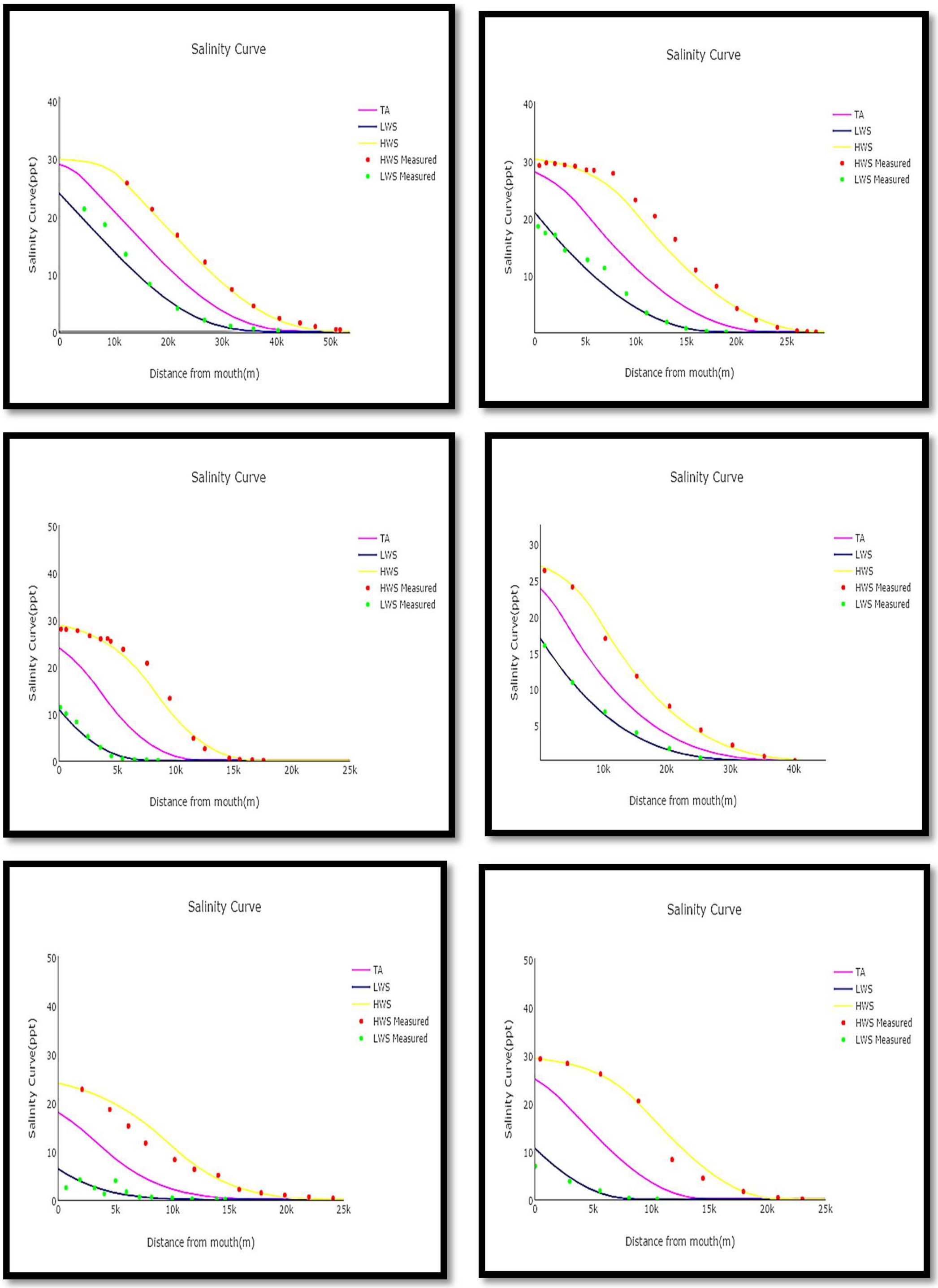


Figure A1 Longitudinal Salinity Distribution of a) Belat b) Endau c) Kurau d) Muar e) Perak f) Selangor

APPENDIX B
***SALT*'S PYTHON CODING**

```
#SALT:Steady State One-Dimensional Salt Intrusion Model at Tidal Average Condition

print("SALT: \nSteady State One-Dimensional Salt Intrusion Model at Tidal Average Condition")

#---------------------------------------------------------------------------------------------------

#Menu

SALT=1 #Looping Purpose

while SALT==1:

    menuselect=input("\n"+"Menu"+"\n"+"[1] New File"+"\n"+"[2] Open File"+"\n"+"[3] Measurement Data"+"\n"+"[4] Save File"+"\n"+"[5] List and Edit Input"+"\n"+"[6] Generate Result"+"\n"+"[7] Help"+"\n"+"[0] Exit"+"\n"+"Select Menu Number:")

    if menuselect==1: #New File

#---------------------------------------------------------------------------------------------------

        print #Spacing

        print("New File")

        #Survey Details

        name=raw_input("Name of the Estuary: ")

        date=raw_input("Date(DDMMYY): ")

        print #Spacing

#---------------------------------------------------------------------------------------------------

        #Geometric Input of Estuary
```

```
print("Geometric Input of Estuary"+"\n"+"Please insert magnitude of the
respective parameters.")

#Input

vargeo=1 #Looping purpose.

while vargeo==1:

	A0=input("1. A0: Area at mouth(m^2): ")

	A1=input("2. A1: Area at x1(m^2): ")

	a1=input("3. a1: Area convergence length(m): ")

	a2=input("4. a2: Area convergence length 2(m): ")

	B0=input("5. B0: Width at mouth(m): ")

	B1=input("6. B1: Width at x1(m): ")

	b1=input("7. b1: Width convergence length(m): ")

	b2=input("8. b2: Width convergence length 2(m): ")

	x1=input("9. x1:Inflection point(m): ")

	h0avg=input("10. h0avg: Depth average(m): ")

	h1avg=input("11. h1avg: Depth average at x1(m): ")

	h0=input("12. h0: Depth(m): ")

	h1=input("13. h1: Depth at x1(m): ")

	#For checking & confirmation purpose.

	print #Spacing

	print("Inserted Geometric Input of Estuary")

	print("1. A0:"+str(A0)+"(m^2)"+"\n"+
```

```
"2. A1:"+str(A1)+"(m^2)"+"\n"+
"3. a1:"+str(a1)+"(m)"+"\n"+
"4. a2:"+str(a2)+"(m)"+"\n"+
"5. B0:"+str(B0)+"(m)"+"\n"+
"6. B1:"+str(B1)+"(m)"+"\n"+
"7. b1:"+str(b1)+"(m)"+"\n"+
"8. b2:"+str(b2)+"(m)"+"\n"+
"9. x1:"+str(x1)+"(m)"+"\n"+
"10. h0avg:"+str(h0avg)+"(m)"+"\n"+
"11. h1avg:"+str(h1avg)+"(m)"+"\n"+
"12. h0:"+str(h0)+"(m)"+"\n"+
"13. h1:"+str(h1)+"(m)"+"\n")
redo=raw_input("Press [1] to: Re-Enter Input, any button to Proceed."+"\n"+"Select :")
if redo=="1":
    vargeo==1 #vargeo = 1 indicate Looping.
    print #Spacing
else:
    vargeo=0 #vargeo=0 indicate end Looping.
    print #Spacing
#----------------------------------------------------------------------------------------------------
    #Input Parameters
```

```
print("Input Parameters"+"\n"+"Please insert magnitude of the respective parameters.")

#Input

varin=1 #Looping purpose.

while varin==1:

    dx=input("14. dx: Step Length(m): ")

    S0=input("15. S0: Sea Salinity(kg/m^3): ")

    E0=input("16. E0: Tidal Excursion(m): ")

    H=input("17. H: Tidal Range(m): ")

    Sf=input("18. Sf: Fresh Water Salinity(kg/m^3): ")

    #For checking & confirmation purpose.

    print #Spacing

    print("Inserted Input Parameters")

    print("14. dx:"+str(dx)+"(m)"+"\n"+

    "15. S0:"+str(S0)+"(kg/m^3)"+"\n"+

    "16. E0:"+str(E0)+"(m)"+"\n"+

     "17. H:"+str(H)+"(m)"+"\n"+

     "18. Sf:"+str(Sf)+"(kg/m^3)"+"\n")

    redo=raw_input("Press [1] to: Re-Enter Input, any button to Proceed."+"\n"+"Select :")

    if redo=="1":

        varin=1 #varin=1 indicate Looping.
```

```
else:
        varin=0 #varin=0 indicate end Looping.
        print #Spacing
#-----------------------------------------------------------------------------------------------------
    #Calibration Parameters
    print("Calibration Parameters"+"\n"+"Please insert magnitude of the respective parameters.")
    Input
    varc=1 #Looping purpose.
    while varc==1:
        K=input("19. K: Van Der Burgh's coefficient: ")
        alpha0=input("20. alpha0: Alpha 0(1/m): ")
        Q=input("21: Q: Fresh Water Discharge(m^3/s): ")
        #Calculation
        D0=Q*alpha0 #Dispersion at mouth
        print("22. D0: Dispersion at mouth(m^2/s):"+str(D0))
        #For checking & confirmation purpose.
        print #Spacing
        print("19. K:"+str(K)+"\n"+
        "20. alpha0:"+str(alpha0)+"(1/m)"+"\n"+
        "21. Q:"+str(Q)+"(m^3/s)"+"\n"+
         "22. D0:"+str(D0)+"(m^2/s)"+"\n")
```

```
        redo=raw_input("Press [1] to: Re-Enter Input, any button to Proceed."+"\n"+"Select :")
        if redo=="1":
            varc=1 #varc=1 indicate Looping.
        else:
            varc=0 #varc=0 indicate end Looping.
    print #Spacing
#---------------------------------------------------------------------------------------------------
    #Calculations
    import math
    def exp(n):
      n=math.exp(n)
      return n
    def loge(n):
      n=math.log(n)
      return n
    beta=(K*a1)/(alpha0*float(A0)) #Beta
    D1=D0*(1-beta*(exp(x1/float(a1))-1)) #Dispersion at x1
    alpha1=D1/float(Q) #Alpha 1
    beta1=(K*a2)/(alpha1*float(A1)) #Beta 1
    S1=(S0-Sf)*((D1/D0)**(1/float(K)))+Sf #Salinity at x1
```

```
    LHWS=x1+a2*loge((A1*alpha1)/(K*float(a2))+1)+(E0/2)        #Salinity
    Length at HWS

  print("Calculations:")

  print("23. beta: Beta: "+str(beta)+"\n"+

      "24. D1: Dispersion at x1: "+str(D1)+"(m^2/s)"+"\n"+

      "25. alpha1: Alpha 1: "+str(alpha1)+"\n"+

      "26. beta1: Beta 1: "+str(beta1)+"\n"+

      "27. S1: Salinity at x1: "+str(S1)+"(kg/m^3)"+"\n"+

      "28. LHWS: Salinity Length at HWS: "+str(LHWS)+"(m)"+"\n")

  print #Spacing

  #Prevent Error in Listing

  countHWS=0

  countLWS=0

  xoHWS=[] #Distance from mouth at LWS

  SoHWS=[] #Salinity of Measurement Data at HWS

  xoLWS=[] #Distance from mouth at LWS

  SoLWS=[] #Salinity of Measurement Data at LWS

#-----------------------------------------------------------------------------------------------------

 if menuselect==3: #Measurement Data

    print("Measurement Data")

    #Array for Multiple Data in Single Variable.

     xoHWS=[] #Distance from mouth at LWS
```

```
SoHWS=[] #Salinity of Measurement Data at HWS

xoLWS=[] #Distance from mouth at LWS

SoLWS=[] #Salinity of Measurement Data at LWS

#InputHWS

varHWS=1 #Looping purpose.

while varHWS==1:

    countHWS=input("Number of measurements of HWS ?: ")

    for n in range(countHWS):

        xmHWS=input("xoHWS: Distance from estuary mouth (m): ") #Cannot have same naming before putting into array.

        xoHWS.append(xmHWS) #Save into array.

        SmHWS=input("SoHWS: Salinity of the measurement data at the point. (kg/m^3): ") #Cannot have same naming before putting into array.

        SoHWS.append(SmHWS) #Save into array.

    #For checking & confirmation purpose.

    print #Spacing

    ####PrettyTable plot#####

    from prettytable import PrettyTable as ptable

    pt=ptable()

    pt.add_column("xoHWS(m)",xoHWS)

    pt.add_column("SoHWS(kg/m^3)",SoHWS)
```

```
            print pt

            ####PrettyTable plot#####

            redo=raw_input("Press [1] to: Re-Enter Input, any button to Proceed."+"\n"+"Select :")

            if redo=="1":

                  #Reset Data

                  xoHWS=[]

                  SoHWS=[]

                  varHWS=1 #varHWS=1 indicate Looping.

            else:

                varHWS=0 #varHWS=0 indicate end Looping.

#InputLWS

varLWS=1 #Looping purpose.

while varLWS==1:

  countLWS=input("Number of measurements of LWS ?: ")

  for n in range(countLWS):

            xmLWS=input("xoLWS: Distance from estuary mouth (m): ") #Cannot have same naming before putting into array.

            xoLWS.append(xmLWS) #Save into array.

            SmLWS=input("SoLWS: Salinity of the measurement data at the point. (kg/m^3): ") #Cannot have same naming before putting into array.

            SoLWS.append(SmLWS) #Save into array.
```

```
                #For checking & confirmation purpose.
                print #Spacing
                ####PrettyTable plot#####
                from prettytable import PrettyTable as ptable
                pt=ptable()
                pt.add_column("xoLWS(m)",xoLWS)
                pt.add_column("SoLWS(kg/m^3)",SoLWS)
                print pt
                ####PrettyTable plot#####
                redo=raw_input("Press [1] to: Re-Enter Input, any button to Proceed."+"\n"+"Select :")
                if redo=="1":
                    #Reset Data
                    xoLWS=[]
                    SoLWS=[]
                    varLWS=1 #varLWS=1 indicate Looping.
                else:
                    varLWS=0 #varLWS=0 indicate end Looping.
#-----------------------------------------------------------------------------------------------------
        if menuselect==4:
            #Save File
```

```
print("Saving...")

filename=name+date+"_data.txt"

save=open(filename,"w")

#All Input Data                                                          #line

save.write("SALT alpha ver. 1.0: Steady State One-Dimensional Salt
Intrusion Model at Tidal Average Condition"+"\n"+                        #1

"(Please return this data text file to Desktop to let SALT application
programme to Open the file)"+"\n"+                                       #2

"(To Open the file,press Open File in SALT menu and enter ONLY the
Name of Estuary and Date of Estuary)"+"\n"+"\n"+                         #3-4

"NameOfTheEstuary: "+name+"\n"+                                          #5

"Date: "+date+"\n"+"\n"+                                                 #6-7

"Geometric Parameters"+"\n"+                                             #8

"1.A0:Area_at_mouth(m^2): "+str(A0)+"\n"+                                #9

"2.A1:Area_x1(m^2):"+str(A1)+"\n"+                                       #10

"3.a1:Area_convergence_length(m):"+str(a1)+"\n"+                         #11

"4.a2:Area_convergence_length_2:"+str(a2)+"\n"+                          #12

"5.B0:Width_at_mouth(m):"+str(B0)+"\n"+                                  #13

"6.B1:Width_at_x1(m):"+str(B1)+"\n"+                                     #14

"7.b1:Width_convergence_length(m):"+str(b1)+"\n"+                        #15

"8.b2:Width_convergence_length_2(m):"+str(b2)+"\n"+                      #16

"9.x1:Inflection_point(m):"+str(x1)+"\n"+                                #17

"10.h0avg:Depth_average(m):"+str(h0avg)+"\n"+                            #18
```

```
"11.h1avg:Depth_average_at_x1(m):"+str(h1avg)+"\n"+                #19
"12.h0:Depth(m): "+str(h0)+"\n"+                                   #20
"13.h1:Depth_at_x1(m):"+str(h1)+"\n"+"\n"+                         #21-22
"Input Parameters"+"\n"+                                           #23
"14.dx:Step_Length(m):"+str(dx)+"\n"+                              #24
"15.S0:Sea_Salinity(kg/m^3):"+str(S0)+"\n"+                        #25
"16.E0:Tidal_Excursion(m):"+str(E0)+"\n"+                          #26
"17.H:Tidal_Range(m):"+str(H)+"\n"+                                #27
"18.Sf:Fresh_Water_Salinity(kg/m^3):"+str(Sf)+"\n"+"\n"+           #28-29
"Calibration Parameters"+"\n"+                                     #30
"19.K:Van_Der_Burgh's_coefficient:"+str(K)+"\n"+                   #31
"20.alpha0:Alpha_0(1/m):"+str(alpha0)+"\n"+                        #32
"21.Q:Fresh_Water_Discharge(m^3/s):"+str(Q)+"\n"+                  #33
"22.D0:Dispersion_at_mouth(m^2/s):"+str(D0)+"\n"+"\n"+             #34
#This part will be calculate back while reading#
"Calculation"+"\n"+                                                #35
"23.beta:Beta: "+str(beta)+"\n"+                                   #36
"24.D1:Dispersion_at_x1(m^2/s):"+str(D1)+"\n"                      #37
"25.alpha1:Alpha_1:"+str(alpha1)+"\n"+                             #38
"26.beta1:Beta_1:"+str(beta1)+"\n"+                                #39
"27.S1:Salinity_at_x1(kg/m^3):"+str(S1)+"\n"+                      #40
```

```
        "28.LHWS:Salinity_Length_at_HWS(m):"+str(LHWS)+"\n"+"\n")
                                                                        #41-42
        #Measurement Data
        save.write("29.MHWS:Measurement_Data,HWS("+str(countHWS)+")\n
        "+                                                              #43
        "xoHWS(m)"+""+"SoHWS(kg/m^3)"+"\n")                             #44
        import numpy as np
        np.savetxt(save,(xoHWS,SoHWS),fmt="%d")                         #45-46
        save.write("30. MLWS: Measurement Data, LWS
        ("+str(countLWS)+")\n"+                                         #47
        "xoLWS(m)"+""+"SoLWS(kg/m^3)"+"\n")                             #48
        np.savetxt(save,(xoLWS,SoLWS),fmt="%d")                         #49-50
        save.close()
        print ("Saved.")
#----------------------------------------------------------------------------------------------------
    if menuselect==5:
      #List and Edit Input
      print("List and Edit Input")
      print("Name of Estuary: "+ name)
      print("Date: "+date)
#----------------------------------------------------------------------------------------------------
      var=1 #Looping purpose.
```

```
#Listing Purpose

while var==1:

    #Recalculation in case of edited

    #Calculations

    import math

    def exp(n):

        n=math.exp(n)

        return n

    def loge(n):

        n=math.log(n)

        return n

    D0=Q*alpha0 #Dispersion at mouth

    beta=(K*a1)/(alpha0*float(A0)) #Beta

    D1=D0*(1-beta*(exp(x1/float(a1))-1)) #Dispersion at x1

    alpha1=D1/float(Q) #Alpha 1

    beta1=(K*a2)/(alpha1*float(A1)) #Beta 1

    S1=(S0-Sf)*((D1/D0)**(1/float(K)))+Sf #Salinity at x1

    LHWS=x1+a2*loge((A1*alpha1)/(K*float(a2))+1)+(E0/2)        #Salinity
    Length at HWS

    print("List of Input")
```

```
print("Geometric Parameters"+"\n"+"\n"+
    "1. A0: Area at mouth: "+str(A0)+"(m^2)"+"\n"+
    "2. A1: Area at x1: "+str(A1)+"(m^2)"+"\n"+
    "3. a1: Area convergence length: "+str(a1)+"(m)"+"\n"+
    "4. a2: Area convergence length 2: "+str(a2)+"(m)"+"\n"+
    "5. B0: Width at mouth: "+str(B0)+"(m)"+"\n"+
    "6. B1: Width at x1: "+str(B1)+"(m)"+"\n"+
    "7. b1: Width convergence length: "+str(b1)+"(m)"+"\n"+
    "8. b2: Width convergence length 2: "+str(b2)+"(m)"+"\n"+
    "9. x1: Inflection point:"+str(x1)+"(m)"+"\n"+
    "10. h0avg: Depth average: "+str(h0avg)+"(m)"+"\n"+
    "11. h1avg: Depth average at x1: "+str(h1avg)+"(m)"+"\n"+
    "12. h0: Depth: "+str(h0)+"(m)"+"\n"+
    "13. h1: Depth at x1: "+str(h1)+"(m)"+"\n"+"\n"+
    "Input Parameters"+"\n"+"\n"+
    "14. dx: Step Length: "+str(dx)+"(m)"+"\n"+
    "15. S0: Sea Salinity: "+str(S0)+"(kg/m^3)"+"\n"+
    "16. E0:  Tidal Excursion: "+str(E0)+"(m)"+"\n"+
    "17. H: Tidal Range: "+str(H)+"(m)"+"\n"+
    "18. Sf: Fresh Water Salinity: "+str(Sf)+"(kg/m^3)"+"\n"+"\n"+
```

```
"Calibration Parameters"+"\n"+"\n"+
"19. K: Van Der Burgh's coefficient: "+str(K)+"\n"+
"20. alpha0: Alpha 0: "+str(alpha0)+"(1/m)"+"\n"+
"21. Q: Fresh Water Discharge: "+str(Q)+"(m^3/s)"+"\n"+
"22. D0: Dispersion at mouth: "+str(D0)+"(m^2/s)"+"\n"+"\n"+
"Calculation"+"\n"+"\n"+
"23. beta: Beta: "+str(beta)+"\n"+
"24. D1: Dispersion at x1: "+str(D1)+"(m^2/s)"+"\n"+
"25. alpha1: Alpha 1: "+str(alpha1)+"\n"+
"26. beta1: Beta 1: "+str(beta1)+"\n"+
"27. S1: Salinity at x1: "+str(S1)+"(kg/m^3)"+"\n"+
"28. LHWS: Salinity Length at HWS: "+str(LHWS)+"(m)"+"\n")
print#Spacing
print("29.MHWS:Measurement_Data_HWS")
print #Spacing
####PrettyTable plot#####
from prettytable import PrettyTable as ptable
pt=ptable()
pt.add_column("xoHWS(m)",xoHWS)
pt.add_column("SoHWS(kg/m^3)",SoHWS)
print pt
```

```
####PrettyTable plot#####

print#Spacing

print("30.MLWS:Measurement_Data_LWS")

print #Spacing

####PrettyTable plot#####

from prettytable import PrettyTable as ptable

pt=ptable()

pt.add_column("xoLWS(m)",xoLWS)

pt.add_column("SoLWS(kg/m^3)",SoLWS)

print pt

####PrettyTable plot#####

print#Spacing

print

#---------------------------------------------------------------------------------------------------------

#Editing Purpose

edit=raw_input("Enter [Number] or [Abbreviation of the parameters] to edit the magnitude,\n"+

"Magnitude of 22-28 cannot be edited due to calculation.\n"+"Any other button for CANCEL edit: ")

if edit=="1" or edit=="A0":

    A0=input("1. A0: Area at mouth(m^2): ")

    var=1
```

```
elif edit=="2" or edit=="A1":
    A1=input("2. A1: Area at x1(m^2): ")
    var=1
elif edit=="3" or edit=="a1":
    a1=input("3. a1: Area convergence length(m): ")
    var=1
elif edit=="4" or edit=="a2":
    a2=input("4. a2: Area convergence length 2(m): ")
    var=1
elif edit=="5" or edit=="B0":
    B0=input("5. B0: Width at mouth(m): ")
    var=1
elif edit=="6" or edit=="B1":
    B1=input("6. B1: Width at x1(m): ")
    var=1
elif edit=="7" or edit=="b1":
    b1=input("7. b1: Width convergence length(m): ")
    var=1
elif edit=="8" or edit=="b2":
    b2=input("8. b2: Width convergence length 2(m): ")
    var=1
```

```
elif edit=="9" or edit=="x1":

        x1=input("9. x1:Inflection point(m): ")

        var=1

elif edit=="10" or edit=="h0avg":

        h0avg=input("10. h0avg: Depth average(m): ")

        var=1

elif edit=="11" or edit=="h1avg":

        h1avg=input("11. h1avg: Depth average at x1(m): ")

        var=1

elif edit=="12" or edit=="h0":

        h0=input("12. h0: Depth(m): ")

        var=1

elif edit=="13" or edit=="h1":

        h1=input("13. h1: Depth at x1(m): ")

        var=1

elif edit=="14" or edit=="dx":

        dx=input("14. dx: Step Length(m): ")

        var=1

elif edit=="15" or edit=="S0":

        S0=input("15. S0: Sea Salinity(kg/m^3): ")

        var=1
```

```
elif edit=="16" or edit=="E0":

        E0=input("16. E0: Tidal Excursion(m): ")

        var=1

elif edit=="17" or edit=="H":

        H=input("17. H: Tidal Range(m): ")

        var=1

elif edit=="18" or edit=="Sf":

        Sf=input("18. Sf: Fresh Water Salinity(kg/m^3): ")

        var=1

elif edit=="19" or edit=="K":

        K=input("19. K: Van Der Burgh's coefficient: ")

        var=1

elif edit=="20" or edit=="alpha0":

        alpha0=input("20. alpha0: Alpha 0(1/m): ")

        var=1

elif edit=="21" or edit=="Q":

        Q=input("21: Q: Fresh Water Discharge(m^3/s): ")

        var=1

elif edit=="29" or edit=="MHWS":

        #Reset Input

        xoHWS=[]
```

```
SoHWS=[]

varHWS=1 #Looping purpose.

while varHWS==1:

    countHWS=input("29. MHWS: Number of measurements of HWS ?: ")

    for n in range(countHWS):

        xmHWS=input("xoHWS: Distance from estuary mouth (m): ") #Cannot have same naming before putting into array.

        xoHWS.append(xmHWS) #Save into array.

        SmHWS=input("SoHWS: Salinity of the measurement data at the point. (kg/m^3): ") #Cannot have same naming before putting into array.

        SoHWS.append(SmHWS) #Save into array.

        #For checking & confirmation purpose.

        print #Spacing

        ####PrettyTable plot#####

        from prettytable import PrettyTable as ptable

        pt=ptable()

        pt.add_column("xoHWS(m)",xoHWS)

         pt.add_column("SoHWS(kg/m^3)",SoHWS)

        print pt

        ####PrettyTable plot#####
```

```
        redo=raw_input("Press [1] to: Re-Enter Input, any button to Proceed."+"\n"+"Select :")
        if redo=="1":
            #Reset Data
             xoHWS=[]
            SoHWS=[]
            varHWS=1 #varHWS=1 indicate Looping.
        else:
            varHWS=0 #varHWS=0 indicate end Looping.
elif edit=="30" or edit=="MLWS":
    #Reset Input
    xoLWS=[]
    SoLWS=[]
    varLWS=1 #Looping purpose.
    while varLWS==1:
        countLWS=input("30. MLWS: Number of measurements of LWS ?:")
        for n in range(countLWS):
            xmLWS=input("xoLWS: Distance from estuary mouth (m): ") #Cannot have same naming before putting into array.
            xoLWS.append(xmLWS) #Save into array.
```

```
        SmLWS=input("SoLWS: Salinity of the measurement data at the point. (kg/m^3): ") #Cannot have same naming before putting into array.

        SoLWS.append(SmLWS) #Save into array

        #For checking & confirmation purpose.

        print #Spacing

        ####PrettyTable plot#####

        from prettytable import PrettyTable as ptable

        pt=ptable()

        pt.add_column("xoLWS(m)",xoLWS)

        pt.add_column("SoLWS(kg/m^3)",SoLWS)

        print pt

        ####PrettyTable plot#####

redo=raw_input("Press [1] to: Re-Enter Input, any button to Proceed."+"\n"+"Select :")

if redo=="1":

        #Reset Data

        xoLWS=[]

        SoLWS=[]

        varLWS=1 #varLWS=1 indicate Looping.

else:

        varLWS=0 #varLWS=0 indicate end Looping.
```

```
        else:
            #var=0 indicate end EDIT Looping.
            var=0
#-----------------------------------------------------------------------------------------------------
    if menuselect==6:
        #Generate Result, Table and Graph
         print("Generate Result, Table and Graph")
#-----------------------------------------------------------------------------------------------------
        iteration=200 #Number of Simulated Data
        variteration=0 #Looping purpose
         while variteration==0:
            changeiteration=raw_input("Iteration = "+str(iteration)+"\n Change value ? Press [1] for Change.Any button to cancel. Default value = 200 \n Select =")
            if changeiteration==1:
                iteration=input("New Iteration Value (Default =200) =")
                variteration=0
            else:
                variteration=1
#-----------------------------------------------------------------------------------------------------
        import math
        def exp(n):
```

```
        n=math.exp(n)
        return n
def loge(n):
        n=math.log(n)
        return n
count=0 #Start
#Array of table
xTA=[]
ATA=[]
DTA=[]
STA=[]
xLWS=[]
xHWS=[]
#Starting Value
xstart=-10000
for n in range(iteration):
        if count==0: #1st Value
                count=count+1
                xsim=xstart
                if xsim<=x1:
                        Asim=A0*exp(-(xsim/float(a1))) #Enable division
                        done correctly
```

```
                if D0*(1-beta*(exp(xsim/float(a1))-1))>0:
                    Dsim=D0*(1-beta*(exp(xsim/float(a1))-
                    1))
                else:
                    Dsim=0
                Ssim=(S0-
                Sf)*((Dsim/float(D0))**(1/float(K)))+Sf
            else:
                Asim=A1*exp(-((xsim-x1)/float(a2)))
                if D1*(1-beta1*(exp((xsim-x1)/float(a2))-1))>0:
                    Dsim=D1*(1-beta1*(exp((xsim-
                    x1)/float(a2))-1))
                else:
                    Dsim=0
                Ssim=((S1-
                Sf)*((Dsim/float(D0))**(1/float(K))))+Sf
        else:
            count=count+1
            xsim=xsim+dx #Subsequent value
            if xsim<=x1:
                Asim=A0*exp(-(xsim/float(a1))) #Enable division
                done correctly
```

```
            if D0*(1-beta*(exp(xsim/float(a1))-1))>0:
                Dsim=D0*(1-beta*(exp(xsim/float(a1))-
                1))
            else:
                Dsim=0
            Ssim=(S0-
            Sf)*((Dsim/float(D0))**(1/float(K)))+Sf
        else:
            Asim=A1*exp(-((xsim-x1)/float(a2)))
            if D1*(1-beta1*(exp((xsim-x1)/float(a2))-1))>0:
                Dsim=D1*(1-beta1*(exp((xsim-
                x1)/float(a2))-1))
            else:
                Dsim=0
                Ssim=((S1-
                Sf)*((Dsim/float(D1))**(1/float(K))))+Sf
    #LWS and HWS calculation
    LWS=xsim-E0/2
    HWS=xsim+E0/2
    #Save data into array
    xTA.append(xsim)
    ATA.append(Asim)
```

```
    DTA.append(Dsim)

    STA.append(Ssim)

    xLWS.append(LWS)

    xHWS.append(HWS)

#--------------------------------------------------------------------------------------------------

  #Tabulation of Data

  print #Spacing

  ####PrettyTable plot#####

  from prettytable import PrettyTable as ptable

  pt=ptable()

  pt.add_column("x(m)",xTA)

  pt.add_column("A(m^2)",ATA)

  pt.add_column("D(m^2/s)",DTA)

  pt.add_column("S(kg/m^3)",STA)

  print pt

  ####PrettyTable plot#####

  #Plotting Graph

  import plotly as py

  import plotly.graph_objs as go

  #TA Simulated #Magenta

  trace1=go.Scatter(
```

```
        name='TA',
        x=xTA,
        y=STA,
        mode='line',
        line=dict(color='rgb(255,0,255)'))
#LWS Simulated #Blue
trace2=go.Scatter(
        name='LWS',
        x=xLWS,
        y=STA,
        mode= 'line',
        line=dict(color='rgb(0,0,204)'))
#HWS Simulated #Yellow
        trace3=go.Scatter(
        name='HWS',
        x=xHWS,
        y=STA,
        mode= 'line',
        line=dict(color='rgb(255,255,0)'))
#HWS Measured #Red
        trace4=go.Scatter(
```

```
        name='HWS Measured',
        x=xoHWS,
        y=SoHWS,
        mode= 'markers',
        marker=dict(color='rgb(255,0,0)'))
#LWS Measured #Green
trace5=go.Scatter(
        name='LWS Measured',
        x=xoLWS,
        y=SoLWS,
        mode= 'markers',
        marker=dict(color='rgb(0,255,0)'))
data=[trace1, trace2, trace3, trace4,trace5]
layout=go.Layout(
        title='Salinity Curve',
        xaxis=dict(
        title='Distance from mouth(m)',
        range=[0,25000],
        showgrid=True,
        zeroline=True,
        showline=True,
```

```
            autotick=True,
            ticks='',
            showticklabels=True),
            yaxis=dict(
            title="Salinity (ppt.)",
            range=[0,50.00],
            showgrid=True,
            zeroline=True,
            showline=True,
            autotick=True,
            ticks='',
            showticklabels=True))
        fig = go.Figure(data=data, layout=layout)
        py.offline.plot(fig,filename="Salinity Curve.html")
#-----------------------------------------------------------------------------------------------------------
        #Model Performance
        print #Spacing
        print("Model Performance")
        def exp(n):
            n=math.exp(n)
            return n
```

```
def loge(n):
	n=math.log(n)
	return n
nHWS=[]
xmHWS=[]
SmHWS=[]
diff2HWS=[]
obsdiff2HWS=[]
NHWS=0
totalSoHWS=0
totaldiffsqrHWS=0
totalobsdiffsqrHWS=0
nLWS=[]
xmLWS=[]
SmLWS=[]
diff2LWS=[]
obsdiff2LWS=[]
NLWS=0
totalSoLWS=0
totaldiffsqrLWS=0
totalobsdiffsqrLWS=0
```

```
#Get HWS Salinity by using TA condition

for n in range (countHWS):

        xmea=xoHWS[n]-E0/2

        if xmea<=x1:

                Amea=A0*exp(-(xmea/float(a1))) #Enable division done correctly

                if D0*(1-beta*(exp(xmea/float(a1))-1))>0:

                        Dmea=D0*(1-beta*(exp(xmea/float(a1))-1))

                else:

                        Dmea=0

                Smea=(S0-Sf)*((Dmea/float(D0))**(1/float(K)))+Sf

        else:

                Amea=A1*exp(-((xmea-x1)/float(a2)))

                if D1*(1-beta1*(exp((xmea-x1)/float(a2))-1))>0:

                        Dmea=D1*(1-beta1*(exp((xmea-x1)/float(a2))-1))

                else:

                        Dmea=0

                Smea=((S1-Sf)*((Dmea/float(D1))**(1/float(K))))+Sf

        xmHWS.append(xoHWS[n]) #Distance of mouth of HWS at TA condition (xoHWS-E0/2)

        SmHWS.append(Smea) #Salinity of simulated HWS at the specific point
```

```
        totalSoHWS=totalSoHWS+SoHWS[n] #Total Observed Salinity for obtain Average of Observed Values

        SoHWSmean=totalSoHWS/countHWS #Average of Observed Values (NSE)

for n in range (countHWS):

        NHWS=NHWS+1

        diffsqrHWS=(SmHWS[n]-SoHWS[n])**2 #(Smea-Sobs)^2

        totaldiffsqrHWS=totaldiffsqrHWS+diffsqrHWS #Summation of (Smea-Sobs)^2 (NSE/RMSE)

        obsdiffsqrHWS=(SoHWS[n]-SoHWSmean)**2 #(Sobs-Sobs(avg))^2 (NSE)

        totalobsdiffsqrHWS=totalobsdiffsqrHWS+obsdiffsqrHWS # Summation of (Sobs-Sobs(avg))^2 (NSE)

        nHWS.append(NHWS)

        diff2HWS.append(diffsqrHWS)

        obsdiff2HWS.append(obsdiffsqrHWS)

#Get LWS Salinity by using TA condition

for n in range (countLWS):

        xmea=xoLWS[n]+E0/2

        if xmea<=x1:

                Amea=A0*exp(-(xmea/float(a1))) #Enable division done correctly
```

```
                if D0*(1-beta*(exp(xmea/float(a1))-1))>0:
                    Dmea=D0*(1-beta*(exp(xmea/float(a1))-1))
                else:
                    Dmea=0
                Smea=(S0-Sf)*((Dmea/float(D0))**(1/float(K)))+Sf
            else:
                Amea=A1*exp(-((xmea-x1)/float(a2)))
                if D1*(1-beta1*(exp((xmea-x1)/float(a2))-1))>0:
                    Dmea=D1*(1-beta1*(exp((xmea-x1)/float(a2))-1))
                else:
                    Dmea=0
                Smea=((S1-Sf)*((Dmea/float(D1))**(1/float(K))))+Sf
                xmLWS.append(xoLWS[n]) #Distance of mouth of LWS at TA condition (xoHWS+E0/2)
                SmLWS.append(Smea) #Salinity of simulated LWS at the specific point
                totalSoLWS=totalSoLWS+SoLWS[n] #Total Observed Salinity for obtain Average of Observed Values
                SoLWSmean=totalSoLWS/countLWS #Average of Observed Values (NSE)
        for n in range (countLWS):
            NLWS=NLWS+1
            diffsqrLWS=(SmLWS[n]-SoLWS[n])**2 #(Smea-Sobs)^2
```

```
            totaldiffsqrLWS=totaldiffsqrLWS+diffsqrLWS  #Summation of (Smea-Sobs)^2 (NSE/RMSE)

            obsdiffsqrLWS=(SoLWS[n]-SoLWSmean)**2  #(Sobs-Sobs(avg))^2 (NSE)

            totalobsdiffsqrLWS=totalobsdiffsqrLWS+obsdiffsqrLWS  # Summation of (Sobs-Sobs(avg))^2 (NSE)

        nLWS.append(NLWS)

        diff2LWS.append(diffsqrLWS)

        obsdiff2LWS.append(obsdiffsqrLWS)

#----------------------------------------------------------------------------------------------------

    #RMSE (HWS)

    RMSEHWS=((totaldiffsqrHWS)/float(countHWS))**0.5

    print("RMSE (HWS) ="+str(RMSEHWS))

    #NSE (HWS)

    NSEHWS=1-((totaldiffsqrHWS)/float(totalobsdiffsqrHWS))

    print("NSE (HWS) ="+str(NSEHWS))

    #RMSE (LWS)

    RMSELWS=((totaldiffsqrLWS)/float(countLWS))**0.5

    print("RMSE (LWS) ="+str(RMSELWS))

    #NSE (LWS)

    NSELWS=1-((totaldiffsqrLWS)/float(totalobsdiffsqrLWS))

    print("NSE (LWS) ="+str(NSELWS))
```

```
#-----------------------------------------------------------------------------------------------------------

        #Tabulation HWS

        #Tabulation RMSE (HWS)

        print #Spacing

        ####PrettyTable plot#####

        print ("####Root Mean Square Error, RMSE (HWS)####")

        total=[countHWS," "," ","Total Error^2",totaldiffsqrHWS]

        rmse=[" "," "," ","RMSE",RMSEHWS]

        from prettytable import PrettyTable as ptable

        pt=ptable()

        pt.add_column("n",nHWS)

        pt.add_column("xoHWS(m)",xmHWS)

        pt.add_column("SmHWS(ppt)",SmHWS)

        pt.add_column("SoHWS(ppt)",SoHWS)

        pt.add_column("(SmHWS-SoHWS)^2",diff2HWS)

        pt.add_row(total)

        pt.add_row(rmse)

        print pt

        ####PrettyTable plot#####

        print #Spacing
```

```
#Tabulation NSE(HWS)

####PrettyTable plot#####

print ("####Nash-Sucliffe Efficiency, NSE (HWS)####")

total=[countHWS," "," ","Total",totaldiffsqrHWS,totalobsdiffsqrHWS]

nse=[" "," "," ","NSE",NSEHWS," "]

from prettytable import PrettyTable as ptable

pt=ptable()

pt.add_column("n",nHWS)

pt.add_column("xoHWS(m)",xmHWS)

pt.add_column("SmHWS(ppt)",SmHWS)

pt.add_column("SoHWS(ppt)",SoHWS)

pt.add_column("(SmHWS-SoHWS)^2",diff2HWS)

pt.add_column("(SoHWS-SoHWSmean)^2",obsdiff2HWS)

pt.add_row(total)

pt.add_row(nse)

print pt

####PrettyTable plot#####

print #Spacing

#----------------------------------------------------------------------------------------------------

#Tabulation LWS

#Tabulation RMSE (LWS)
```

```
####PrettyTable plot#####

print ("####Root Mean Square Error, RMSE (LWS)####")

total=[countLWS," "," ","Total Error^2",totaldiffsqrLWS]

rmse=[" "," "," ","RMSE",RMSELWS]

from prettytable import PrettyTable as ptable

pt=ptable()

pt.add_column("n",nLWS)

pt.add_column("xoLWS(m)",xmLWS)

pt.add_column("SmLWS(ppt)",SmLWS)

pt.add_column("SoLWS(ppt)",SoLWS)

pt.add_column("(SmLWS-SoLWS)^2",diff2LWS)

pt.add_row(total)

pt.add_row(rmse)

print pt

####PrettyTable plot#####

#Tabulation NSE(LWS)

####PrettyTable plot#####

print ("####Nash-Sucliffe Efficiency, NSE (LWS)####")

total=[countLWS," "," ","Total",totaldiffsqrLWS,totalobsdiffsqrLWS]

nse=[" "," "," ","NSE",NSELWS," "]

from prettytable import PrettyTable as ptable
```

```
            pt=ptable()
            pt.add_column("n",nLWS)
            pt.add_column("xoLWS(m)",xmLWS)
            pt.add_column("SmLWS(ppt)",SmLWS)
            pt.add_column("SoLWS(ppt)",SoLWS)
            pt.add_column("(SmLWS-SoLWS)^2",diff2LWS)
            pt.add_column("(SoLWS-SoLWSmean)^2",obsdiff2LWS)
            pt.add_row(total)
            pt.add_row(nse)
            print pt
            ####PrettyTable plot#####
#------------------------------------------------------------------------------------------------------
        if menuselect==7:#Help
            print("This SALT application programme uses the theory of Savenije(2005) and Gisen et al.(2015) model. Please refer to Salinity and Tides website (https://salinityandtides.com/).")
#------------------------------------------------------------------------------------------------------
        if menuselect==0:#Exit
            SALT=0 #No Looping
            break
```