ESTIAMTION OF LAND SURFACE EVAPOTRANSPIRATION FROM POTENTIAL EVAPOTRANSPIRATION USING NON-LINEAR, LINEAR AND COMPLEMENTARY RELATIONSHIP FUNCTIONS

Report submitted to the SASTRA Deemed to be University as the requirement for the course

CSE300 – MINI PROJECT

Submitted by

VARSHITHA K
(Reg. No.: 124158095, B.Tech CSE (IOT & A))
BHAVANA MALLINENI
(Reg. No.: 124158090, B.Tech CSE (IOT & A))
HARSHA ABHINAV K
(Reg. No.: 124158027, B.Tech CSE (IOT & A))

May 2023

SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 of the UGC Act, 1956)
THINK MERIT | THINK TRANSPARENCY | THINK SASTRA
THANJAVUR | KUMBAKONAM | CHENNAI

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled "**Estimation of land surface evapotranspiration from potential evapotranspiration using Non-linear, linear and complementary Relationship functions**" submitted as a requirement for the course, CSE300 : **MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Mr. Harsha Abhinav K (Reg. No.124158027, CSE (IOT & A)), Ms.Bhavana Mallineni (Reg. No.124158090, CSE (IOT & A)), Ms.Varshitha K (Reg. No.124158095, CSE (IOT & A))** during the academic year 2022-23, in the School of Computing, under my supervision.

Signature of Project Supervisor :

Name with Affiliation :

Date :

Mini Project Viva voce held on ________________

Examiner 1 **Examiner 2**

Acknowledgements

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. A. Umamakeswari**, Dean, School of Computing,

Dr. S. Gopalakrishnan, Associate Dean, Department of Computer Application, **Dr. B.Santhi**, Associate Dean, Research**, Dr. V. S. Shankar Sriram**, Associate Dean, Department of Computer Science and Engineering, **Dr. R. Muthaiah**, Associate Dean, Department of Information Technology and Information & Communication Technology

Our guide **Dr.Ezhilarasie**, Assistant Professor, School of Computing was the driving force behind this whole idea from the start. His deep insight in the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing me an opportunity to showcase my skills through project.

Abbreviations

AET	Actual Evapotranspiration
BR	Baier-Robertson
CR	Complementary Relationship
ET	Evapotranspiration
HS	Hargreaves-Samani
FLUXNET	Flux Network
LF	Linear Function
NLF	Non-Linear Function
NSE	Nash-Sutcliffe efficiency
PET	Potential Evapotranspiration
PF	Power Function
PM	Penman-Monteith equation
PT	Priestley-Taylor equation
RMSE	Root Mean Square error
SWC	Soil Water Content
SWI	Soil Water Index

Notations

English Symbols

ETpa	Potential Evapotranspiration of a crop with a short canopy (in mm/day)
ETpw	Potential Evapotranspiration of a crop with a tall canopy (in mm/day)
ET_a	Actual Evapotranspiration
e_s	Saturation Vapor Pressure
e_a	Actual Vapor Pressure
G	Soil heat Flux density
I	Thermal Index
k	Proportionality constant
N	Daytime length
Rn	Net Radiation
Rs	Solar Radiation
T	Temperature
u	Wind speed
vpd	Vapor Pressure deficit
X_{mi}	Modelled ET time series
X_{oi}	Observed ET time series
$\overline{X}_o$	Mean of the observed ET

Greek Symbols

Δ	Gradient of saturation vapor Pressure
λ	Latent heat of vaporization (2.453)
γ	Psychometric Constant
$\sum$	Summation
ρ	Density of water
α	Surface Moisture Constant (0-1.26)

Abstract

Evapotranspiration (ET) is a fundamental factor in energy and hydrologic cycles. So, the precise simulation of actual evapotranspiration (AET) values is essential for ecological restoration, agriculture and water resources management. Evapotranspiration is equal to the fraction of potential evapotranspiration (PET) constrained by soil water. This study estimates PET in real-time. PET can be estimated from the abundant FLUXNET meteorological observations. However, the challenge lies in accurately simulating daily evapotranspiration through PET. Non-linear and linear approaches have been developed to simulate evapotranspiration through PET, based on soil moisture on a daily basis. Complementary Relationship functions are also developed. The evaluation showed that the accuracy of ET simulation using the non-linear method was better than linear relations and complementary relationship methods. In some regions, daily evapotranspiration can be simulated accurately with PET data. However, in some regions the result can be poor. Our results indicate that the accurate simulation of daily evapotranspiration can be achieved based on meteorological data. Due to the widespread availability of global meteorological observations, the proposed method for accurate simulation of the daily evapotranspiration can be applied globally.

KEY WORDS: Actual Evapotranspiration, Potential Evapotranspiration, Complementary Relationship, Nash-Sutcliffe efficiency

Table of Contents

CHAPTER 1

SUMMARY OF THE BASE PAPER

Base Paper Details
Title : Estimating land evapotranspiration from potential evapotranspiration constrained by soil water daily scale
Author : Zhaofei liu
Publication year : 2022

1.1 INTRODUCTION

Evapotranspiration is a natural process that involves the transfer of water from the surface of the earth to the atmosphere. This process occurs through two main mechanisms: Evaporation and Transpiration. Evaporation is the process of conversion of water from liquid state to gaseous state, which is then released into the atmosphere. Transpiration, on the other hand, is the release of water vapor from plants through their leaves.

The combined process of evaporation and transpiration is crucial to the movement of water and energy in the earths system. Evapotranspiration regulates the exchange of water and energy between the land and the atmosphere, and plays a key role in the water cycle and the climate. It is an important factor that acts as a bridge between the hydrological cycle, surface energy balance and carbon cycle.

Evapotranspiration is an important factor in crop Management, as it influences the water requirements of the crops. By monitoring the water usage of the crop and water lost through evapotranspiration, farmers and agriculturists can determine how much water crops need and when to irrigate them.

Evapotranspiration is also called as actual evapotranspiration (AET). Simulating actual ET values can be useful for a range of applications, including water resource management, crop yield forecasting, and climate modelling. AET values can be simulated by Potential Evapotranspiration (PET). PET is the amount of water that would be evaporated and transpired if an adequate supply of water were available in the soil. In general, AET is less than or equal to PET, as it represents the actual water use by plants, which is constrained by water availability in the soil, also called as Soil water constraint (SWC).

AET can be simulated by constructing a statistical function relationship between AET and PET under a soil water constraint. The FLUXNET global observation dataset has been widely used for evaluating ET estimations. In this proposed work, four nonlinear functions (NLF), two linear functions (LF), four complementary Relationship (CR) functions are developed to simulate AET from PET. These functions are generated by assuming relationship between AET, PET and soil water index (SWI). The accuracy of these functions has also been evaluated using Nash-Sutcliffe efficiency (NSE) coefficient.

1.2 METHODS FOR PET CALCULATION

PET is defined as the upper limit of evapotranspiration. There are 8 PET equations evaluated in this work, including 3 energy-based equations and 5 temperature-based equations. The three energy-based equations include Penman equations, Penman-Monteith (PM) equation, Priestley and Taylor (PT) equation. The five temperature based equations include Hargreaves-Samani (HS), Thronthwaite, Oudin, Hamon, and Baier-Robertson (BR) equations.

The calculation of PET values requires meteorological data such as temperature, wind speed, atmospheric pressure, vapor pressure, net radiation and soil heat flux data.

Penman Equation :

$$PET = \frac{\Delta Rn + 0.4244(1 + 0.536u)(vpd)}{(\Delta + 0.0067)(2.501 - 0.00236T)} \tag{1.2.1}$$

Penman-Monteith Equation :

$$PET = \frac{0.408\Delta(Rn - G) + \gamma \frac{900}{T + 273} u \times vpd}{\Delta + \gamma(1 + 0.34u)} \tag{1.2.2}$$

Priestley-Taylor Equation:

$$PET = \alpha \frac{\Delta}{\lambda(\Delta + \gamma)} (Rn - G) \tag{1.2.3}$$

Hargreaves-Samani Equation:

$$PET = n(T + 17.8) \frac{Rs}{\lambda} \tag{1.2.4}$$

Where n=0.0135

Oudin Equation:

$$PET = \frac{Rn}{\lambda\rho} \times \frac{T+5}{100} \quad (1.2.5)$$

Hamon Equation:

$$PET = k \times 0.165 \times 216.7 \times N \times \frac{vpd}{T + 273.3} \quad (1.2.6)$$

Thronthwaite Equation:

$$\mathrm{PET} = 16\left(\frac{\mathrm{T}}{\mathrm{I}}\right)^{a} \quad (1.2.7)$$

Where,

$$\mathrm{I} = \sum_{\mathrm{n}=1}^{12} (0.2\mathrm{T})^{1.514} \quad (1.2.8)$$

$$\mathrm{a} = 6.75 \times 10^{-7}\mathrm{I}^{3} - 7.71 \times 10^{-5}\mathrm{I}^{2} + 1.7912 \times 10^{-2}\mathrm{I} + 0.49239 \quad (1.2.9)$$

Baier-Robertson Equation:

$$PET = 0.156T + 0.158TD + 0.109Rn - 5.39 \quad (1.2.10)$$

1.3 METHODS FOR AET CALCULATION

First, Four Non-linear functions are developed between AET and PET to simulate daily AET values under soil constraint. Here we use soil water Index (SWI) which is the ratio of SWC and saturation water content. These four NLF (NLF1-4) are developed by assuming power and exponential functions between AET and PET and SWI.

AET = $a_1^{(SWI-1)}$ x PET, $a_1 \in (1,10]$ (1.3.1)

AET = exp[a_2(SWI-1)] x PET, $a_2 \in (0,10]$ (1.3.2)

AET = SWI^{a_3} x PET, $a_3 \in (1,10]$ (1.3.3)

AET = [1-(1-SWI)$^{(a_4)}$ x PET, $a_4 \in (0,10]$ (1.3.4)

The values of the parameters a_1, a_2, a_3, a_4 need to be calibrated. For the calibration of parameter a we use least square regression for non linear functions method.

Second, linear functions are developed. Here SWC is not used.

$$AET_{linear} = a_5 \text{ x } PET, \qquad a_5 \in (0,1] \tag{1.3.5}$$

$$AET_{power} = a_6 \text{ x } PET^b, \qquad a_6 \in (0,1),\ b \in (0,1] \tag{1.3.6}$$

The parameters a_5, a_6, and b needs to be calibrated

Third, the CR methods are generated. These methods involves three variables: ET_a, ET_{pa}, and ET_{pw}. ET_a is the AET, and ET_{pa} is the potential evaporation. ET_{pw} is defined as the evapotranspiration value when AET was equal to ET_{pa}. The CR methods use these three variables to calculate AET by determining some empirical relationships between them.

B1963, G1989, B2015 and M2015 equations are given below.

$$AET_{B1963} = 2ET_{pw}\text{-}ET_{pa} \tag{1.3.7}$$

$$AET_{G1989} = \frac{\Delta + \gamma}{\Delta} ET_{pw} - \frac{\gamma}{\Delta} ET_{pa} \tag{1.3.8}$$

$$AET_B = \left(\frac{ETpw}{ETpa}\right)^2 \left[(2\text{-}c)ETpa - (1\text{-}2c)ETpw - c\frac{ETpw^2}{ETpa}\right] \tag{1.3.9}$$

$$AET_M = \frac{1}{\varepsilon}[(1+\varepsilon)ETpw\text{-}ETpa] \tag{1.3.10}$$

1.4 EQUATIONS FOR EVALUATION

The six temperature-based equations are evaluated and The performance of Non-linear, linear and CR equations is evaluated. The evaluation criteria used here is the Nash and Sutcliffe efficiency coefficient (NSE) and Relative error (RE) method.

$$NSE = 1\text{-}\frac{\sum_{i=1}^{n}(Xmi - Xoi)^2}{\sum_{i=1}^{n}(Xoi\text{-}\overline{Xo})^2} \tag{1.4.1}$$

$$RE = \frac{1}{n}\sum_{i=1}^{n}\frac{Xmi - Xoi}{Xoi} \tag{1.4.2}$$

Six temperature-based equations for calculating PET values are evaluated based on the simulation results of Penman-Monteith (PM) method. They are valuated based on the NSE and RE values.

The Non-linear functions, linear functions and CR functions are also evaluated based on the observed AET value which is calculated through latent heat flux value from the dataset. They have been evaluated based on the NSE coefficient value.

CHAPTER 2

MERITS AND DEMERITS OF THE BASE PAPER

There are many existing equations which are widely used for the calculation of PET values. In this paper, PET values have been generated using three energy-based equations : Penman, Penman-Monteith (PM), Priestley-Taylor (PT) and six temperature-based equations : Hargreaves-Samani (HS), Thronthwaite, Oudin, Hamon, Baier-Robertson (BR) equations. The 6 temeprature-based equations have been evaluated and the most accurate three equations have been considered for the further estimation of AET values.

Penman-Monteith equation is widely used for estimating PET using meteorological variables such as temperature, humidity, wind speed and solar radiation. PT method is an alternative method which is based on the energy balance approach. HS and Oudin equations require fewer climate variables. The temperature based equations are effective only when the accurate temperature data is available. HS equation is the crop ET without any soil water constraint.

Peng et al. (2019) and Li et al. (2016) has evaluated the ratio of AET to PET using eddy covariance flux measurements at different cities and generated linear relationships between AET and reference crop ET. The seasonal variations of the ratio of AET to PET has also been calculated and estimated by eddy covariance measurements in different sitesThe results showed that the ratio is a function of SWC. Therefore, the ET values depend on the water availability.

The existing equations for AET calculation like PM and Shuttleworth-Wallace equatiosn, do not consider SWC or soil moisture content.In this paper four Non-linear functions, two linear functions (one linear and one power function) between AET and PET with a constraint under soil water availability have been proposed. The parameters in the equations have been calibrated. As the equations take SWI value into account, It can improve the accuracy of resulting AET values compared to other methods.

2.1 MERITS:

1. New Methods for estimating AET values have been proposed, which could potentially improve our understanding of water usage by the crops and plants, which would in turn helps us in effective water management practices.
2. Estimating accurate AET values will improve irrigation practices as AET is the important factor that affects crop productivity.
3. The proposed method calculates AET values based on both PET values and Soil water availability SWC values. So, soil water constraint has been applied which improves the accuracy of AET values which the existing methods can’t provide.

4. The proposed methods are relatively simple and computationally efficient

2.2 DEMERITS:

1. The paper doesn't compare the proposed functions with already existing functions for the calculation of AET values, which could limit our understanding the strength and weakness of proposed work and the existing work
2. The Accurate simulation of AET values is difficult, and we can't accurately conclude which function or method is accurate
3. The observed AET values are not directly given in the FLUXNET dataset. They are provided in the form of Energy (Latent heat of Flux).
4. The daily AET series depends on climatic characteristics and varies across the year, so the accuracy of daily AET simulations is still poor

CHAPTER 3

SOURCE CODE

3.1 Estimating PET values :

```
import numpy as np
import pandas as pd
import math
```

```
df=pd.read_csv("Initial_DataSet.csv")
```

```
df.dtypes
```

```
TA_F             float64
SW_IN_F          float64
LW_IN_F          float64
LW_OUT           float64
VPD_F            float64
PA_F             float64
WS_F             float64
NETRAD           float64
G_F_MDS          float64
LE_F_MDS         float64
SWC_F_MDS_1_QC   float64
SWC_F_MDS_1      float64
AET_MS           float64
SWI              float64
dtype: object
```

```
no_value=['LW_OUT']

for column in no_value:
  df[column]=df[column].replace(-9999,np.NaN)
  mean=int(df[column].mean(skipna=True))
  df[column]=df[column].replace(np.NaN,mean)
```

```
def penman(Rn,T,wind,vpd,g,pa):

  #calculate psychrometric constant
  PSY = 0.000665*pa

  #calculate slope of vapor pressure curve
  slope=4098*(0.6108*np.exp((17.27*T)/(T+237.3)))/((T+237.3)**2)

  #calculate PET
  pet=(slope*(Rn-g)+6.43*PSY*(1+0.536*wind)*vpd)/((2.501-0.00236*T)*(slope+PSY))

  return pet/28.9

df['PET_P']=penman(df['NETRAD'],df['TA_F'],df['WS_F'],df['VPD_F'],df['G_F_MDS'],df['PA_F'])
df['PET_P']
```

```
def priestley(netrad,T,wind,vpd,g,pa):
  alpha = 1.26

  slope=4098*0.6108*np.exp((17.27*T)/(T+237.3))/((T+237.3)**2)
  PSY = 0.000665*pa

  pet = alpha*(slope*(netrad-g)/2.453*(slope+PSY))

  return pet

df['PET_PT']=priestley(df['NETRAD'],df['TA_F'],df['WS_F'],df['VPD_F'],df['G_F_MDS'],df['PA_F']
df['PET_PT']
```

```python
def penmanMonteith(Rn,T,wind,vpd,g,pa):

  slope=4098*(0.6108*np.exp((17.27*T)/(T+237.3)))/((T+237.3)**2)
  PSY = 0.000665*pa

  num = 0.408*slope*(Rn-g)+PSY*(900/(T+273.15))*wind*vpd
  den = slope+PSY*(1+0.34*wind)

  pet = num/den

  return pet/28.9

df['PET_PM']=penmanMonteith(df['NETRAD'],df['TA_F'],df['WS_F'],df['VPD_F'],df['G_F_MDS'],df['PA_F'])
df['PET_PM']
```

```python
def Hargreaves(T,ters_rad,wind):

  Ra=ters_rad
  #pet = (0.0023*Rs*math.sqrt(8)*(T+b))/(2.501-0.002361*T)
  pet=0.0135*(T+17.8)*Ra*0.408*0.19*np.sqrt(6)

  return pet

df['PET_HS']=Hargreaves(df['TA_F'],df['LW_OUT']-df['LW_IN_F'],df['WS_F'])
df['PET_HS']
```

```python
def oudin(T,Rs):

  pet=0.55*0.35*0.35*4.95*np.exp(0.062*T)
  return pet

df['PET_OD']=oudin(df['TA_F'],df['NETRAD'])
df['PET_OD']
```

```python
def Hamon(T):

  PET=1.6169*math.pow(10,-3)*216.7*6.108*(np.exp(17.2693*T/(T+237.3)))*0.35*0.35
  return PET

df['PET_Ham']=Hamon(df['TA_F'])
df['PET_Ham']
```

```python
def Thornthwaite(T):
  pi=3.14
  lat=-12.4943
  lat_rad=lat*180/pi
  solar_dec=0.409*np.sin(2*pi-1.39)
  ws=np.arccos(-np.tan(lat_rad)*np.tan(solar_dec))
  N=(24/pi)*ws

  hi=12*np.power(T/5,1.514)
  a=(6.75*(1/np.power(10,7))*np.power(hi,3))-(7.751*(1/np.power(10,5))*(hi**2))+(0.01792*(hi))+(0.49239)
  PET=16*(N/360)*np.power((10*T/hi),a)

  return PET

df['PET_TH']=Thornthwaite(df['TA_F'])
df['PET_TH']
```

```
#Baier-Robertson Code

def baier_robertson(T,ters_rad):
  Ra=ters_rad
  TD=6
  PET=0.157*T+0.158*TD+0.109*Ra-5.39
  return PET

df['PET_BR']=baier_robertson(df['TA_F'],df['LW_OUT']-df['LW_IN_F'])
df['PET_BR']
```

3.2 Estimating AET values :

```
[ ] import numpy as np
    import pandas as pd
    import math
    import matplotlib.pyplot as plt
    from scipy.optimize import curve_fit
```

```
df=pd.read_csv("PET_Values_DataSet.csv")

no_zero=['SWC_F_MDS_1_QC']

for column in no_zero:
  df[column]=df[column].replace(0,np.NaN)
  mean=int(df[column].mean(skipna=True))
  df[column]=df[column].replace(np.NaN,mean)
```

```
[ ] df.dtypes
```

```
[ ] PET=df['PET_P']
    SWI=df['SWI']
    AET=df['AET_MS']
    x=SWI
    y=AET/PET
```

```
[ ] #Non Linear Function 1

    def nlf_1(x,a):
      return np.power(a,x-1)

    lower_bound=1.0
    upper_bound=10.0

    popt,pcov=curve_fit(nlf_1,x,y,bounds=(lower_bound,upper_bound))
    a1_fit=popt[0]
```

```
print('a1 : ',a1_fit)

AET_NLF1_P=df['PET_P']*np.power(a1_fit,x-1)
df['AET_NLF1_P']=AET_NLF1_P

AET_NLF1_PM=df['PET_PM']*np.power(a1_fit,x-1)
df['AET_NLF1_PM']=AET_NLF1_PM

AET_NLF1_PT=df['PET_PT']*np.power(a1_fit,x-1)
df['AET_NLF1_PT']=AET_NLF1_PT

AET_NLF1_HS=df['PET_HS']*np.power(a1_fit,x-1)
df['AET_NLF1_HS']=AET_NLF1_HS

AET_NLF1_OD=df['PET_OD']*np.power(a1_fit,x-1)
df['AET_NLF1_OD']=AET_NLF1_OD

AET_NLF1_Ham=df['PET_Ham']*np.power(a1_fit,x-1)
df['AET_NLF1_Ham']=AET_NLF1_Ham

AET_NLF1_TH=df['PET_TH']*np.power(a1_fit,x-1)
df['AET_NLF1_TH']=AET_NLF1_TH

AET_NLF1_BR=df['PET_BR']*np.power(a1_fit,x-1)
df['AET_NLF1_BR']=AET_NLF1_BR
```

```
[ ] #Non Linear Function 2

    def nlf_2(x,a):
      return np.exp(a*(x-1))

    lower_bound=0.0
    upper_bound=10.0

    popt,pcov=curve_fit(nlf_2,x,y,bounds=(lower_bound,upper_bound))
    a2_fit=popt[0]
    print(a2_fit)
```

```
[ ] AET_NLF2_P=df['PET_P']*np.exp(a2_fit*(x-1))
    df['AET_NLF2_P']=AET_NLF2_P

    AET_NLF2_PM=df['PET_PM']*np.exp(a2_fit*(x-1))
    df['AET_NLF2_PM']=AET_NLF2_PM

    AET_NLF2_PT=df['PET_PT']*np.exp(a2_fit*(x-1))
    df['AET_NLF2_PT']=AET_NLF2_PT

    AET_NLF2_HS=df['PET_HS']*np.exp(a2_fit*(x-1))
    df['AET_NLF2_HS']=AET_NLF2_HS

    AET_NLF2_OD=df['PET_OD']*np.exp(a2_fit*(x-1))
    df['AET_NLF2_OD']=AET_NLF2_OD

    AET_NLF2_Ham=df['PET_Ham']*np.exp(a2_fit*(x-1))
    df['AET_NLF2_Ham']=AET_NLF2_Ham

    AET_NLF2_TH=df['PET_TH']*np.exp(a2_fit*(x-1))
    df['AET_NLF2_TH']=AET_NLF2_TH

    AET_NLF2_BR=df['PET_BR']*np.exp(a2_fit*(x-1))
    df['AET_NLF2_BR']=AET_NLF2_BR
```

```
[ ] #Non Linear Function 3

    def nlf_3(x,a):
      return np.power(x,a)

    lower_bound=1.0
    upper_bound=10.0

    popt,pcov=curve_fit(nlf_3,x,y,bounds=(lower_bound,upper_bound))
    a3_fit=popt[0]
    print(a3_fit)

    AET_NLF3_P=df['PET_P']*np.power(x,a3_fit)
```

```
df['AET_NLF3_P']=AET_NLF3_P

AET_NLF3_PM=df['PET_PM']*np.power(x,a3_fit)
df['AET_NLF3_PM']=AET_NLF3_PM

AET_NLF3_PT=df['PET_PT']*np.power(x,a3_fit)
df['AET_NLF3_PT']=AET_NLF3_PT

AET_NLF3_HS=df['PET_HS']*np.power(x,a3_fit)
df['AET_NLF3_HS']=AET_NLF3_HS

AET_NLF3_OD=df['PET_OD']*np.power(x,a3_fit)
df['AET_NLF3_OD']=AET_NLF3_OD

AET_NLF3_Ham=df['PET_Ham']*np.power(x,a3_fit)
df['AET_NLF3_Ham']=AET_NLF3_Ham

AET_NLF3_TH=df['PET_TH']*np.power(x,a3_fit)
df['AET_NLF3_TH']=AET_NLF3_TH

AET_NLF3_BR=df['PET_BR']*np.power(x,a3_fit)
df['AET_NLF3_BR']=AET_NLF3_BR
```

```
[ ] #Non Linear Function 4

    def nlf_4(x,a):
      return 1-np.power(x,a)

    lower_bound=0.0
    upper_bound=10.0

    popt,pcov=curve_fit(nlf_4,x,y,bounds=(lower_bound,upper_bound))
    a4_fit=popt[0]
    print(a4_fit)

    df['AET_NLF4_P']=df['PET_P']*(1-np.power(1-x,a4_fit))
    df['AET_NLF4_PM']=df['PET_PM']*(1-np.power(1-x,a4_fit))
    df['AET_NLF4_PT']=df['PET_PT']*(1-np.power(1-x,a4_fit))
```

```
df['AET_NLF4_HS']=df['PET_HS']*(1-np.power(1-x,a4_fit))
df['AET_NLF4_OD']=df['PET_OD']*(1-np.power(1-x,a4_fit))
df['AET_NLF4_Ham']=df['PET_Ham']*(1-np.power(1-x,a4_fit))
df['AET_NLF4_TH']=df['PET_TH']*(1-np.power(1-x,a4_fit))
df['AET_NLF4_BR']=df['PET_BR']*(1-np.power(1-x,a4_fit))
```

```
[ ] #Linear Function 1

    def linear(AET,PET):
      mean_aet=np.mean(AET)
      mean_pet=np.mean(PET)
      a5=mean_aet/mean_pet
      return a5*PET


    df['AET_lf_P']=linear(AET,df['PET_P'])
    df['AET_lf_PM']=linear(AET,df['PET_PM'])
    df['AET_lf_PT']=linear(AET,df['PET_PT'])
    df['AET_lf_HS']=linear(AET,df['PET_HS'])
    df['AET_lf_OD']=linear(AET,df['PET_OD'])
    df['AET_lf_Ham']=linear(AET,df['PET_Ham'])
    df['AET_lf_TH']=linear(AET,df['PET_TH'])
    df['AET_lf_BR']=linear(AET,df['PET_BR'])
```

```
[ ] #Linear Function 2

    def lf_2(pet,a,b):
      return a*np.power(pet,b)

    p0=[0.01,0.1]
    lower_bound=0.001
    upper_bound=1
    popt,pcov=curve_fit(lf_2,PET,AET,p0,bounds=[lower_bound,upper_bound])
    a=popt[0]
    b=popt[1]

    print('a :',a)
    print('b :',b)

    df['AET_PF_P']=a*np.power(df['PET_P'],b)
    df['AET_PF_PM']=a*np.power(df['PET_PM'],b)
    df['AET_PF_PT']=a*np.power(df['PET_PT'],b)
    df['AET_PF_HS']=a*np.power(df['PET_HS'],b)
    df['AET_PF_OD']=a*np.power(df['PET_OD'],b)
    df['AET_PF_Ham']=a*np.power(df['PET_Ham'],b)
    df['AET_PF_TH']=a*np.power(df['PET_TH'],b)
    df['AET_PF_BR']=a*np.power(df['PET_BR'],b)
```

```
[ ] #Complementary Relationship function 1
    #B1963

    df['AET_B1963_P']=2*(df['PET_PT'])-df['PET_P'];
    df['AET_B1963_PM']=2*(df['PET_PT'])-df['PET_PM'];
    df['AET_B1963_PT']=2*(df['PET_PT'])-df['PET_PT'];
    df['AET_B1963_HS']=2*(df['PET_PT'])-df['PET_HS'];
    df['AET_B1963_OD']=2*(df['PET_PT'])-df['PET_OD'];
    df['AET_B1963_Ham']=2*(df['PET_PT'])-df['PET_Ham'];
    df['AET_B1963_TH']=2*(df['PET_PT'])-df['PET_TH'];
    df['AET_B1963_BR']=2*(df['PET_PT'])-df['PET_BR'];
```

```
[ ] #Complementary Relationship function 2
    #G1989

    #calculate psychrometric constant
    PSY = 0.000665*df['PA_F']

    T=df['TA_F']
    #calculate slope of vapor pressure curve
    slope=4098*(0.6108*np.exp((17.27*T)/(T+237.3)))/((T+237.3)**2)

    # CR METHODS
    df['AET_G1989_P']=(((PSY + slope)/PSY)*(df['PET_PT']))-(PSY/slope)*(df['PET_P']);
    df['AET_G1989_PM']=(((PSY + slope)/PSY)*(df['PET_PT']))-(PSY/slope)*(df['PET_PM']);
    df['AET_G1989_PT']=(((PSY + slope)/PSY)*(df['PET_PT']))-(PSY/slope)*(df['PET_PT']);
```

```
[ ] df['AET_G1989_HS']=(((PSY + slope)/PSY)*(df['PET_PT']))-(PSY/slope)*(df['PET_HS']);
    df['AET_G1989_OD']=(((PSY + slope)/PSY)*(df['PET_PT']))-(PSY/slope)*(df['PET_OD']);
    df['AET_G1989_Ham']=(((PSY + slope)/PSY)*(df['PET_PT']))-(PSY/slope)*(df['PET_Ham']);
    df['AET_G1989_TH']=(((PSY + slope)/PSY)*(df['PET_PT']))-(PSY/slope)*(df['PET_TH']);
    df['AET_G1989_BR']=(((PSY + slope)/PSY)*(df['PET_PT']))-(PSY/slope)*(df['PET_BR']);
```

```
[ ] #Complementary Relationship function 3
    #B2015


    c=0.12;
    df['AET_B2015_P']=(np.power(df['PET_PT']/df['PET_P'],2)*(((2-c)*df['PET_P'])-((1-2*c)*df['PET_PT'])-c*np.power(df['PET_PT'],2)/df['PET_P']));
    df['AET_B2015_PM']=(np.power(df['PET_PT']/df['PET_PM'],2)*(((2-c)*df['PET_PM'])-((1-2*c)*df['PET_PT'])-c*np.power(df['PET_PT'],2)/df['PET_PM']));
    df['AET_B2015_PT']=(np.power(df['PET_PT']/df['PET_PT'],2)*(((2-c)*df['PET_PT'])-((1-2*c)*df['PET_PT'])-c*np.power(df['PET_PT'],2)/df['PET_PT']));
    df['AET_B2015_HS']=(np.power(df['PET_PT']/df['PET_HS'],2)*(((2-c)*df['PET_HS'])-((1-2*c)*df['PET_PT'])-c*np.power(df['PET_PT'],2)/df['PET_HS']));
    df['AET_B2015_OD']=(np.power(df['PET_PT']/df['PET_OD'],2)*(((2-c)*df['PET_OD'])-((1-2*c)*df['PET_PT'])-c*np.power(df['PET_PT'],2)/df['PET_OD']));
    df['AET_B2015_Ham']=(np.power(df['PET_PT']/df['PET_Ham'],2)*(((2-c)*df['PET_Ham'])-((1-2*c)*df['PET_PT'])-c*np.power(df['PET_PT'],2)/df['PET_Ham']));
    df['AET_B2015_TH']=(np.power(df['PET_PT']/df['PET_TH'],2)*(((2-c)*df['PET_TH'])-((1-2*c)*df['PET_PT'])-c*np.power(df['PET_PT'],2)/df['PET_TH']));
    df['AET_B2015_BR']=(np.power(df['PET_PT']/df['PET_BR'],2)*(((2-c)*df['PET_BR'])-((1-2*c)*df['PET_PT'])-c*np.power(df['PET_PT'],2)/df['PET_BR']));
```

```
[ ] #Complementary Relationship function 4
    #M2015

    ε=0.12
    df['AET_M2015_P']=(1/ε)*(((1+ε)*df['PET_PT'])-df['PET_P']);
    df['AET_M2015_PM']=(1/ε)*(((1+ε)*df['PET_PT'])-df['PET_PM']);
    df['AET_M2015_PT']=(1/ε)*(((1+ε)*df['PET_PT'])-df['PET_PT']);
    df['AET_M2015_HS']=(1/ε)*(((1+ε)*df['PET_PT'])-df['PET_HS']);
    df['AET_M2015_OD']=(1/ε)*(((1+ε)*df['PET_PT'])-df['PET_OD']);
    df['AET_M2015_Ham']=(1/ε)*(((1+ε)*df['PET_PT'])-df['PET_Ham']);
    df['AET_M2015_TH']=(1/ε)*(((1+ε)*df['PET_PT'])-df['PET_TH']);
    df['AET_M2015_BR']=(1/ε)*(((1+ε)*df['PET_PT'])-df['PET_BR']);
```

3.3 Evaluating PET values:

```
# Load NSE data for each PET method into a pandas DataFrame


obs=df['AET_MS']
numerator_HS = np.sum((df['PET_HS'] -obs)**2)
numerator_TH = np.sum((df['PET_TH'] -obs)**2)
numerator_OD = np.sum((df['PET_OD'] -obs)**2)
numerator_Ham = np.sum((df['PET_Ham'] -obs)**2)
numerator_BR = np.sum((df['PET_BR'] -obs)**2)

obs_mean=np.mean(obs)

denominator=np.sum((obs-obs_mean)**2)

# Calculate the NSE value
NSE_HS = 1 - numerator_HS / denominator
NSE_TH = 1 - numerator_TH / denominator
NSE_OD = 1 - numerator_OD / denominator
NSE_Ham = 1 - numerator_Ham / denominator
NSE_BR = 1 - numerator_BR / denominator

print("NSE_HS : ",NSE_HS)
print("NSE_TH : ",NSE_TH)
print("NSE_OD : ",NSE_OD)
print("NSE_Ham : ",NSE_Ham)
print("NSE_BR : ",NSE_BR)
```

```
#Calculating Relative Error

# calculate mean observed value

mean_observed = np.mean(df['PET_PM']*28.9)

# calculate RE for each data point
residuals_HS = (df['AET_MS'])-df['PET_HS']
residuals_TH = (df['AET_MS'])-df['PET_TH']
residuals_OD = (df['AET_MS'])-df['PET_OD']
residuals_Ham = (df['AET_MS'])-df['PET_Ham']
residuals_BR = (df['AET_MS'])-df['PET_BR']

relative_errors_HS=residuals_HS/mean_observed
relative_errors_TH=residuals_TH/mean_observed
relative_errors_OD=residuals_OD/mean_observed
relative_errors_Ham=residuals_Ham/mean_observed
relative_errors_BR=residuals_BR/mean_observed

# calculate mean RE
mean_re_HS=np.mean(relative_errors_HS)
mean_re_TH=np.mean(relative_errors_TH)
mean_re_OD=np.mean(relative_errors_OD)
mean_re_Ham=np.mean(relative_errors_Ham)
mean_re_BR=np.mean(relative_errors_BR)

#print the values
print("RE_HS : ",mean_re_HS)
print("RE_TH : ",mean_re_TH)
print("RE_OD : ",mean_re_OD)
print("RE_Ham : ",mean_re_Ham)
print("RE_BR : ",mean_re_BR)
```

3.4 Estimating NLF, LF, CR between AET and PET

```
#Nash and Sutcliffe Efficieny Co-efficient
#NSE For NLF1

ET_obs = df['AET_MS']         # observed evapotranspiration
ET_pred_NLF1_PM = df['AET_NLF1_PM'] # predicted evapotranspiration
ET_pred_NLF1_PT = df['AET_NLF1_PT']
ET_pred_NLF1_P = df['AET_NLF1_P']
ET_pred_NLF1_HS = df['AET_NLF1_HS']
ET_pred_NLF1_OD = df['AET_NLF1_OD']
ET_pred_NLF1_Ham = df['AET_NLF1_Ham']

# Calculate the mean of the observed evapotranspiration values
ET_obs_mean = np.mean(ET_obs)

# Calculate the numerator and denominator of the NSE equation
numerator_NLF1_PM = np.sum((ET_obs - ET_pred_NLF1_PM)**2)
numerator_NLF1_PT = np.sum((ET_obs - ET_pred_NLF1_PT)**2)
numerator_NLF1_P = np.sum((ET_obs - ET_pred_NLF1_P)**2)
numerator_NLF1_HS = np.sum((ET_obs - ET_pred_NLF1_HS)**2)
numerator_NLF1_OD = np.sum((ET_obs - ET_pred_NLF1_OD)**2)
numerator_NLF1_Ham = np.sum((ET_obs - ET_pred_NLF1_Ham)**2)

denominator=np.sum((ET_obs-ET_obs_mean)**2)

# Calculate the NSE value
NSE_NLF1_PM = 1 - numerator_NLF1_PM / denominator
NSE_NLF1_PT = 1 - numerator_NLF1_PT / denominator
NSE_NLF1_P = 1 - numerator_NLF1_P / denominator
NSE_NLF1_HS = 1 - numerator_NLF1_HS / denominator
NSE_NLF1_OD = 1 - numerator_NLF1_OD / denominator
NSE_NLF1_Ham= 1 - numerator_NLF1_Ham / denominator

d1['NLF1']=[NSE_NLF1_PM,NSE_NLF1_PT,NSE_NLF1_P,NSE_NLF1_HS,NSE_NLF1_OD,NSE_NLF1_Ham]
```

```
#Nash and Sutcliffe Efficieny Co-efficient
#NSE For NLF2

ET_obs = df['AET_MS']         # observed evapotranspiration
ET_pred_NLF2_PM = df['AET_NLF2_PM'] # predicted evapotranspiration
ET_pred_NLF2_PT = df['AET_NLF2_PT']
ET_pred_NLF2_P = df['AET_NLF2_P']
ET_pred_NLF2_HS = df['AET_NLF2_HS']
ET_pred_NLF2_OD = df['AET_NLF2_OD']
ET_pred_NLF2_Ham = df['AET_NLF2_Ham']

# Calculate the mean of the observed evapotranspiration values
ET_obs_mean = np.mean(ET_obs)

# Calculate the numerator and denominator of the NSE equation
numerator_NLF2_PM = np.sum((ET_obs - ET_pred_NLF2_PM)**2)
numerator_NLF2_PT = np.sum((ET_obs - ET_pred_NLF2_PT)**2)
numerator_NLF2_P = np.sum((ET_obs - ET_pred_NLF2_P)**2)
numerator_NLF2_HS = np.sum((ET_obs - ET_pred_NLF2_HS)**2)
numerator_NLF2_OD = np.sum((ET_obs - ET_pred_NLF2_OD)**2)
numerator_NLF2_Ham = np.sum((ET_obs - ET_pred_NLF2_Ham)**2)

denominator=np.sum((ET_obs-ET_obs_mean)**2)

# Calculate the NSE value
NSE_NLF2_PM = 1 - numerator_NLF2_PM / denominator
NSE_NLF2_PT = 1 - numerator_NLF2_PT / denominator
NSE_NLF2_P = 1 - numerator_NLF2_P / denominator
NSE_NLF2_HS = 1 - numerator_NLF2_HS / denominator
NSE_NLF2_OD = 1 - numerator_NLF2_OD / denominator
NSE_NLF2_Ham= 1 - numerator_NLF2_Ham / denominator

d1['NLF2']=[NSE_NLF2_PM,NSE_NLF2_PT,NSE_NLF2_P,NSE_NLF2_HS,NSE_NLF2_OD,NSE_NLF2_Ham]
```

```
#Nash and Sutcliffe Efficieny Co-efficient
#NSE For NLF3

ET_obs = df['AET_MS']          # observed evapotranspiration
ET_pred_NLF3_PM = df['AET_NLF3_PM'] # predicted evapotranspiration
ET_pred_NLF3_PT = df['AET_NLF3_PT']
ET_pred_NLF3_P = df['AET_NLF3_P']
ET_pred_NLF3_HS = df['AET_NLF3_HS']
ET_pred_NLF3_OD = df['AET_NLF3_OD']
ET_pred_NLF3_Ham = df['AET_NLF3_Ham']

# Calculate the mean of the observed evapotranspiration values
ET_obs_mean = np.mean(ET_obs)

# Calculate the numerator and denominator of the NSE equation
numerator_NLF3_PM = np.sum((ET_obs - ET_pred_NLF3_PM)**2)
numerator_NLF3_PT = np.sum((ET_obs - ET_pred_NLF3_PT)**2)
numerator_NLF3_P = np.sum((ET_obs - ET_pred_NLF3_P)**2)
numerator_NLF3_HS = np.sum((ET_obs - ET_pred_NLF3_HS)**2)
numerator_NLF3_OD = np.sum((ET_obs - ET_pred_NLF3_OD)**2)
numerator_NLF3_Ham = np.sum((ET_obs - ET_pred_NLF3_Ham)**2)

denominator=np.sum((ET_obs-ET_obs_mean)**2)

# Calculate the NSE value
NSE_NLF3_PM = 1 - numerator_NLF3_PM / denominator
NSE_NLF3_PT = 1 - numerator_NLF3_PT / denominator
NSE_NLF3_P = 1 - numerator_NLF3_P / denominator
NSE_NLF3_HS = 1 - numerator_NLF3_HS / denominator
NSE_NLF3_OD = 1 - numerator_NLF3_OD / denominator
NSE_NLF3_Ham= 1 - numerator_NLF3_Ham / denominator

d1['NLF3']=[NSE_NLF3_PM,NSE_NLF3_PT,NSE_NLF3_P,NSE_NLF3_HS,NSE_NLF3_OD,NSE_NLF3_Ham]
```

```
#Nash and Sutcliffe Efficieny Co-efficient
#NSE For NLF4

ET_obs = df['AET_MS']          # observed evapotranspiration
ET_pred_NLF4_PM = df['AET_NLF4_PM'] # predicted evapotranspiration
ET_pred_NLF4_PT = df['AET_NLF4_PT']
ET_pred_NLF4_P = df['AET_NLF4_P']
ET_pred_NLF4_HS = df['AET_NLF4_HS']
ET_pred_NLF4_OD = df['AET_NLF4_OD']
ET_pred_NLF4_Ham = df['AET_NLF4_Ham']

# Calculate the mean of the observed evapotranspiration values
ET_obs_mean = np.mean(ET_obs)

# Calculate the numerator and denominator of the NSE equation
numerator_NLF4_PM = np.sum((ET_obs - ET_pred_NLF4_PM)**2)
numerator_NLF4_PT = np.sum((ET_obs - ET_pred_NLF4_PT)**2)
numerator_NLF4_P = np.sum((ET_obs - ET_pred_NLF4_P)**2)
numerator_NLF4_HS = np.sum((ET_obs - ET_pred_NLF4_HS)**2)
numerator_NLF4_OD = np.sum((ET_obs - ET_pred_NLF4_OD)**2)
numerator_NLF4_Ham = np.sum((ET_obs - ET_pred_NLF4_Ham)**2)

denominator=np.sum((ET_obs-ET_obs_mean)**2)

# Calculate the NSE value
NSE_NLF4_PM = 1 - numerator_NLF4_PM / denominator
NSE_NLF4_PT = 1 - numerator_NLF4_PT / denominator
NSE_NLF4_P = 1 - numerator_NLF4_P / denominator
NSE_NLF4_HS = 1 - numerator_NLF4_HS / denominator
NSE_NLF4_OD = 1 - numerator_NLF4_OD / denominator
NSE_NLF4_Ham= 1 - numerator_NLF4_Ham / denominator

d1['NLF4']=[NSE_NLF4_PM,NSE_NLF4_PT,NSE_NLF4_P,NSE_NLF4_HS,NSE_NLF4_OD,NSE_NLF4_Ham]
```

```
#NSE For LF1

ET_obs = df['AET_MS']          # observed evapotranspiration
ET_pred_LF1_PM = df['AET_lf_PM'] # predicted evapotranspiration
ET_pred_LF1_PT = df['AET_lf_PT']
ET_pred_LF1_P = df['AET_lf_P']
ET_pred_LF1_HS = df['AET_lf_HS']
ET_pred_LF1_OD = df['AET_lf_OD']
ET_pred_LF1_Ham = df['AET_lf_Ham']

# Calculate the mean of the observed evapotranspiration values
ET_obs_mean = np.mean(ET_obs)

# Calculate the numerator and denominator of the NSE equation
numerator_LF1_PM = np.sum((ET_obs - ET_pred_LF1_PM)**2)
numerator_LF1_PT = np.sum((ET_obs - ET_pred_LF1_PT)**2)
numerator_LF1_P = np.sum((ET_obs - ET_pred_LF1_P)**2)
numerator_LF1_HS = np.sum((ET_obs - ET_pred_LF1_HS)**2)
numerator_LF1_OD = np.sum((ET_obs - ET_pred_LF1_OD)**2)
numerator_LF1_Ham = np.sum((ET_obs - ET_pred_LF1_Ham)**2)

denominator=np.sum((ET_obs-ET_obs_mean)**2)

# Calculate the NSE value
NSE_LF1_PM = 1 - numerator_LF1_PM / denominator
NSE_LF1_PT = 1 - numerator_LF1_PT / denominator
NSE_LF1_P = 1 - numerator_LF1_P / denominator
NSE_LF1_HS = 1 - numerator_LF1_HS / denominator
NSE_LF1_OD = 1 - numerator_LF1_OD / denominator
NSE_LF1_Ham= 1 - numerator_LF1_Ham / denominator

d1['LF']=[NSE_LF1_PM,NSE_LF1_PT,NSE_LF1_P,NSE_LF1_HS,NSE_LF1_OD,NSE_LF1_Ham]
```

```
#NSE For LF2 OR POWER FUNCTION

ET_obs = df['AET_MS']          # observed evapotranspiration
ET_pred_LF2_PM = df['AET_PF_PM'] # predicted evapotranspiration
ET_pred_LF2_PT = df['AET_PF_PT']
ET_pred_LF2_P = df['AET_PF_P']
ET_pred_LF2_HS = df['AET_PF_HS']
ET_pred_LF2_OD = df['AET_PF_OD']
ET_pred_LF2_Ham = df['AET_PF_Ham']

# Calculate the mean of the observed evapotranspiration values
ET_obs_mean = np.mean(ET_obs)

# Calculate the numerator and denominator of the NSE equation
numerator_LF2_PM = np.sum((ET_obs - ET_pred_LF2_PM)**2)
numerator_LF2_PT = np.sum((ET_obs - ET_pred_LF2_PT)**2)
numerator_LF2_P = np.sum((ET_obs - ET_pred_LF2_P)**2)
numerator_LF2_HS = np.sum((ET_obs - ET_pred_LF2_HS)**2)
numerator_LF2_OD = np.sum((ET_obs - ET_pred_LF2_OD)**2)
numerator_LF2_Ham = np.sum((ET_obs - ET_pred_LF2_Ham)**2)

denominator=np.sum((ET_obs-ET_obs_mean)**2)

# Calculate the NSE value
NSE_LF2_PM = 1 - numerator_LF2_PM / denominator
NSE_LF2_PT = 1 - numerator_LF2_PT / denominator
NSE_LF2_P = 1 - numerator_LF2_P / denominator
NSE_LF2_HS = 1 - numerator_LF2_HS / denominator
NSE_LF2_OD = 1 - numerator_LF2_OD / denominator
NSE_LF2_Ham= 1 - numerator_LF2_Ham / denominator

d1['PF']=[NSE_LF2_PM,NSE_LF2_PT,NSE_LF2_P,NSE_LF2_HS,NSE_LF2_OD,NSE_LF2_Ham]
```

```
#NSE For Complementary relationship Function B1963

ET_obs = df['AET_MS']          # observed evapotranspiration
ET_pred_b1963_PM = df['AET_B1963_PM'] # predicted evapotranspiration
ET_pred_b1963_PT = df['AET_B1963_PT']
ET_pred_b1963_P = df['AET_B1963_P']
ET_pred_b1963_HS = df['AET_B1963_HS']
ET_pred_b1963_OD = df['AET_B1963_OD']
ET_pred_b1963_Ham = df['AET_B1963_Ham']

# Calculate the mean of the observed evapotranspiration values
ET_obs_mean = np.mean(ET_obs)

# Calculate the numerator and denominator of the NSE equation
numerator_b1963_PM = np.sum((ET_obs - ET_pred_b1963_PM)**2)
numerator_b1963_PT = np.sum((ET_obs - ET_pred_b1963_PT)**2)
numerator_b1963_P = np.sum((ET_obs - ET_pred_b1963_P)**2)
numerator_b1963_HS = np.sum((ET_obs - ET_pred_b1963_HS)**2)
numerator_b1963_OD = np.sum((ET_obs - ET_pred_b1963_OD)**2)
numerator_b1963_Ham = np.sum((ET_obs - ET_pred_b1963_Ham)**2)

denominator=np.sum((ET_obs-ET_obs_mean)**2)

# Calculate the NSE value
NSE_b1963_PM = 1 - numerator_b1963_PM / denominator
NSE_b1963_PT = 1 - numerator_b1963_PT / denominator
NSE_b1963_P = 1 - numerator_b1963_P / denominator
NSE_b1963_HS = 1 - numerator_b1963_HS / denominator
NSE_b1963_OD = 1 - numerator_b1963_OD / denominator
NSE_b1963_Ham= 1 - numerator_b1963_Ham / denominator

d1['LF']=[NSE_b1963_PM,NSE_b1963_PT,NSE_b1963_P,NSE_b1963_HS,NSE_b1963_OD,NSE_b1963_Ham]
```


```
# Calculate the mean of the observed evapotranspiration values
ET_obs_mean = np.mean(ET_obs)

# Calculate the numerator and denominator of the NSE equation
numerator_b1963_PM = np.sum((ET_obs - ET_pred_b1963_PM)**2)
numerator_b1963_PT = np.sum((ET_obs - ET_pred_b1963_PT)**2)
numerator_b1963_P = np.sum((ET_obs - ET_pred_b1963_P)**2)
numerator_b1963_HS = np.sum((ET_obs - ET_pred_b1963_HS)**2)
numerator_b1963_OD = np.sum((ET_obs - ET_pred_b1963_OD)**2)
numerator_b1963_Ham = np.sum((ET_obs - ET_pred_b1963_Ham)**2)

denominator=np.sum((ET_obs-ET_obs_mean)**2)

# Calculate the NSE value
NSE_b1963_PM = 1 - numerator_b1963_PM / denominator
NSE_b1963_PT = 1 - numerator_b1963_PT / denominator
NSE_b1963_P = 1 - numerator_b1963_P / denominator
NSE_b1963_HS = 1 - numerator_b1963_HS / denominator
NSE_b1963_OD = 1 - numerator_b1963_OD / denominator
NSE_b1963_Ham= 1 - numerator_b1963_Ham / denominator

d1['LF']=[NSE_b1963_PM,NSE_b1963_PT,NSE_b1963_P,NSE_b1963_HS,NSE_b1963_OD,NSE_b1963_Ham]
```

CHAPTER 4

SNAPSHOTS

Penman-Monteith Equation :

```
0       1.511162
1       1.506868
2       1.525567
3       1.475580
4       1.483668
           ...
5108    1.736031
5109    1.574631
5110    1.935188
5111    1.433266
5112    1.625606
Name: PET_PM, Length: 5113, dtype: float64
```

Priestley-Taylor Equation :

```
0       4.081680
1       3.970721
2       4.038734
3       3.722600
4       3.755355
           ...
5108    4.942757
5109    4.281594
5110    5.368785
5111    3.279681
5112    3.759829
Name: PET_PT, Length: 5113, dtype: float64
```

Penman Equation :

```
0       1.682134
1       1.677711
2       1.711343
3       1.630486
4       1.641340
           ...
5108    1.950530
5109    1.784804
5110    2.175986
5111    1.623844
5112    1.840314
Name: PET_P, Length: 5113, dtype: float64
```

Hargreaves Equation :

```
0       3.325658
1       3.400538
2       3.249600
3       3.208022
4       2.931782
          ...
5108    3.841980
5109    3.728473
5110    3.080945
5111    2.417595
5112    2.051654
Name: PET_HS, Length: 5113, dtype: float64
```

Oudin Equation :

```
0       1.758892
1       1.725516
2       1.745963
3       1.650900
4       1.660858
          ...
5108    1.864678
5109    1.825329
5110    1.816748
5111    1.562271
5112    1.528640
Name: PET_OD, Length: 5113, dtype: float64
```

Hamon Equation :

```
0       1.514026
1       1.486760
2       1.503474
3       1.425470
4       1.433677
          ...
5108    1.599857
5109    1.568034
5110    1.561079
5111    1.352064
5112    1.324034
Name: PET_Ham, Length: 5113, dtype: float64
```

Thornthwaite Equation :

```
0       4.539295
1       4.365830
2       4.471074
3       4.007886
4       4.053379
          ...
5108    5.149381
5109    4.911238
5110    4.861117
5111    3.631652
5112    3.501559
Name: PET_TH, Length: 5113, dtype: float64
```

Baier-Robertson Equation :

```
0       2.937867
1       2.983312
2       2.854983
3       2.736974
4       2.476433
          ...
5108    3.502105
5109    3.368636
5110    2.752662
5111    1.875464
5112    1.472938
Name: PET_BR, Length: 5113, dtype: float64
```

Evaluation Results of PET equations :

NSE**:**

```
NSE_HS :  -4.006396247369707
NSE_TH :  -3.842502424736545
NSE_OD :  -4.720709430859118
NSE_Ham :  -4.795499688463782
NSE_BR :  -4.097654247867864
```

RE:

```
RE_HS :  0.5718347608243542
RE_TH :  0.5761462076478854
RE_OD :  0.6349889142735553
RE_Ham :  0.6398464231381706
RE_BR :  0.5793918652365503
```

Evaluation Results of NLF, LF, CR equations:

LF1:

```
NSE_LF1_PM :  -0.008451237276017709
NSE_LF1_PT :  0.13486521741021307
NSE_LF1_P :  -0.012358209406052145
NSE_LF1_HS :  -2.957974596556938
NSE_LF1_OD :  0.15068461191402693
NSE_LF1_Ham :  0.14790797298968772
```

LF2:

```
NSE_LF2_PM :  -4.722496787861554
NSE_LF2_PT :  -3.971802240881834
NSE_LF2_P :  -4.649093998636092
NSE_LF2_HS :  -3.932001738685524
NSE_LF2_OD :  -4.720709430859119
NSE_LF2_Ham :  -4.795499688463783
```

NLF1:

```
NSE_NLF2_PM :  -4.722496787861554
NSE_NLF2_PT :  -3.9721592999768127
NSE_NLF2_P :  -4.649093998636091
NSE_NLF2_HS :  -4.006396247369707
NSE_NLF2_OD :  -4.720709430859118
NSE_NLF2_Ham :  -4.795499688463782
```

NLF2:

```
NSE_NLF1_PM :  -4.722496787861554
NSE_NLF1_PT :  -3.9721592999768127
NSE_NLF1_P :  -4.649093998636092
NSE_NLF1_HS :  -4.006396247369707
NSE_NLF1_OD :  -4.720709430859118
NSE_NLF1_Ham :  -4.795499688463782
```

NLF3:

```
NSE_NLF3_PM :  -4.99316422661766
NSE_NLF3_PT :  -4.583355959693831
NSE_NLF3_P :  -4.956101183975049
NSE_NLF3_HS :  -4.687538523547722
NSE_NLF3_OD :  -4.985945787928463
NSE_NLF3_Ham :  -5.024576383571744
```

NLF4:

```
NSE_NLF4_PM :  -4.7260759303640105
NSE_NLF4_PT :  -3.984628790589235
NSE_NLF4_P :  -4.653867559827843
NSE_NLF4_HS :  -4.024853362036466
NSE_NLF4_OD :  -4.723747225823138
NSE_NLF4_Ham :  -4.797399178413502
```

CR – B1963 :

```
NSE_b1963_PM :  -3.2870236928355396
NSE_b1963_PT :  -3.9721592999768127
NSE_b1963_P :  -3.3496533895997294
NSE_b1963_HS :  -4.071581783926463
NSE_b1963_OD :  -3.2899139502448627
NSE_b1963_Ham :  -3.2271918971653433
```

CR – G1989 :

```
NSE_g1989_PM :  -1.1122821395042441
NSE_g1989_PT :  -1.2337820034938751
NSE_g1989_P :  -1.1246364704720038
NSE_g1989_HS :  -1.2761379676202358
NSE_g1989_OD :  -1.1127480832327588
NSE_g1989_Ham :  -1.1005214120679603
```

CR – B2015

```
NSE_B2015_PM :  -9.838615017416952
NSE_B2015_PT :  -3.9721592999768127
NSE_B2015_P :  -7.364070237544144
NSE_B2015_HS :  -6978893821148.972
NSE_B2015_OD :  -11.629512563221995
NSE_B2015_Ham :  -19.409056822680476
```

CR – M2015 :

```
NSE_M2015_PM :  -0.25497559651659296
NSE_M2015_PT :  -3.972159299976812
NSE_M2015_P :  -0.4477191101264307
NSE_M2015_HS :  -8.884718179468292
NSE_M2015_OD :  -0.3127608095825478
NSE_M2015_Ham :  -0.1588277277437391
```

CHAPTER 5
CONCLUSION AND FUTURE PLANS

According the Evaluation results of six temperature-based equations for calculating PET values, the HS equation is the most accurate and the values are close to the PM calculation values followed by Oudin and Hamon equations. The later two methods achieved a certain accuracy. Hargreaves-Samani equation can replace PM equation to calculate PET values.

The AET values have been found using NLF, LF and CR equations using the resultant PET values we got from PM, PT, penman, HS, Oudin and Hamon methods. When the performance of two linear functions is compared, the NSE value of LF is higher than that of PF function. But LF is less accurate than NLF

The evaluation results of NLF functions have shown that the exponential functions (NLF3 and NLF4) perform slightly better than the power functions (NLF1 and NLF2). The combination of NLF3 with PM equation have shown greater accuracy. The evaluation of CR methods has shown that B2015 and M2015 equations performed better than other methods. The performance of CR methods is roughly equivalent to the LF methods.

The evaluation results show that PET is suitable to simulate AET values. PET values can accurately simulate AET values on a daily scale. PET values can be calculated from meteorological data, or they can be calculated using only temperature data. Therefore, an accurate simulation of a daily series of ET can be achieved using meteorological data.

The accuracy of the simulated AET values depends on the climatic characteristics of the site. The daily AET values across the year is almost irregular. Therefore, the accuracy of daily AET values is still poor.

The paper provides methods that has the potential to make users understand about plant water use. But more work is needed to accurately simulate the AET values. Additional variables regarding climate, soil dynamics need to be added and considered to simulate more accurate daily AET series.

CHAPTER 6
REFERENCES

Akuraju, V.R., Ryu, D., George, B., Ryu, Y., Dassanayake, K., 2017. Seasonal and inter-annual variability of soil moisture stress functionin dryland wheat field, Australia. Agric. For. Meteorol. 232, 489–499. https://doi.org/10.1016/j.agrformet.2016.10.007.

Baier, W., Robertson, G.W., 1965. Estimation of latent evaporation from simple weather ob servations. Can. J. Plant Sci. 45, 276–284. https://doi.org/10.4141/cjps65-051.

Brutsaert, W., 2015. A generalized complementary principle with physical constraints for land-surface evaporation. Water Resour. Res. 51, 8087–8093. https://doi.org/10.1002/2015WR017720.

Hamon, W.R., 1963. Computation of direct runoff amounts from storm rainfall. Int. Assoc. Sci. Hydrol. Publ. 63, 52–62.

Hargreaves, G.H., Samani, Z.A., 1985. Reference crop evapotranspiration from temperature. Appl. Eng. Agric. 1 (2), 96–99. https://doi.org/10.13031/2013.26773.

Monteith, J.L., 1981. Evaporation and surface temperature. Quart. J. Roy. Meteorol. Soc. 107, 1–27. https://doi.org/10.1256/smsqj.45101.

Oudin, L., Hervieu, F., Michel, C., Perrin, C., Andréassian, V., Anctil, F., Loumagne, C., 2005. Which potential evapotranspiration input for a lumped rainfall-runoff model? Part 2: to wards a simple and efficient potential evapotranspiration model for rainfall-runoff model ing. J. Hydrol. 303, 290–306. https://doi.org/10.1016/j.jhydrol.2004.08.026.

Peng, L., Zeng, Z., Wei, Z., Chen, A., Wood, E.F., Sheffield, J., 2019. Determinants of the ratio of actual to potential evapotranspiration. Glob. Change Biol. 2019 (25), 1326–1343. https://doi.org/10.1111/gcb.14577.

Priestley, C.H.B., Taylor, R.J., 1972. On the assessment of surface heat flux and evaporation using large-scale parameters. Mon. Weather Rev. 100 (2), 81–92. https://doi.org/10.1175/1520-0493(1972)1002.3.CO;2.

Thornthwaite, C.W., 1948. An approach toward a rational classification of climate. Geogr. Rev. 38 (1), 55–94. https://doi.org/10.1097/00010694-194807000-00007.

Zhang, L., Cheng, L., Brutsaert, W., 2017. Estimation of land surface evaporation using a gen eralized nonlinear complementary relationship. J. Geophys. Res. Atmos. 122, 1475–1487. https://doi.org/10.1002/2016JD025936.