Appendix

The code can also be found in this Python Notebook.

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from CoolProp.CoolProp import PropsSI
```

Input Data

[ ]:
```
T_now_C = 11.07 # Initial Temperature of Pool in °C
A = 40          # Surface Area of Pool in sq.m
H = 1.5         # Depth of Pool in m

T_now = T_now_C + 273.15
V = A*H
m = PropsSI('D','P',101325,'T',T_now,'Water')*V
```

Control Variables

1 for on, 0 for off

[ ]:
```
convection = 1
evaporation = 1
solar = 1
control = 1
ctrend = 1      # Correlation for convection
etrend = 1      # Correlation for evaporation

eff = 0.2       # Solar Panel Efficiency

hVal = 20       # Convective Heat Transfer Coefficient in W/m2K
eVal = 10       # Volumetric Rate of Evaporation in liters/day
```

External Heating Control

Specify the required temperature

[ ]:
```
T_user_C = 18
T_user = T_user_C + 273.15
```

Solar Panel Control

[ ]:
```
a = 10          # Area of solar panel in m^2
```

Convection Control

[ ]:
```
def h_conv(w):
    if ctrend==1:
      y = 10 + (30)*(w-1)/(9)
      return y
    else:
      return hVal
```

Evaporation Control

[ ]:
```
q_ev = PropsSI('H','P',101325,'Q',1,'Water') -␣
 ↪PropsSI('H','P',101325,'Q',0,'Water')

def ev1(w,T):
  if etrend==1:
    y = 4 + (16-4)*(w)/(3)
    D = PropsSI('D','P',101325,'T',T,'Water')
    return 0.25*y/D
  else:
    return eVal*1e-3/24
```

KNMI Weather Data

[ ]:
```
url = 'https://raw.githubusercontent.com/shyam97/modelling1/master/knmi.csv'
array = pd.read_csv(url, sep=',',header=None)
temp = 0.1*array.values[:,0]
shine = 1e4*array.values[:,1]
wind = 0.1*array.values[:,2]
hours = np.linspace(1,31,num=len(temp))
```

Plot of Local Temperature

[ ]:
```
plt.figure(figsize=(20,5))
plt.plot(hours,temp,'b')
plt.xlim([1,30])
plt.xlabel("Day in April")
plt.ylabel("Temperature in °C")
plt.xticks(np.arange(min(hours), max(hours)+1, 1.0))
plt.title("Air Temperature Variation")
```

Plot of Sunshine

[ ]:
```
plt.figure(figsize=(20,5))
plt.plot(hours,1e-4*shine,'b')
plt.xlim([1,30])
plt.xlabel("Day in April")
```

```
plt.ylabel("Irradiation in J/sq.cm/hr")
plt.xticks(np.arange(min(hours), max(hours)+1, 1.0))
plt.title("Sunshine Variation")
```

Plot of Wind Speed

[ ]:
```
plt.figure(figsize=(20,5))
plt.plot(hours,wind,'b')
plt.xlim([1,30])
plt.xlabel("Day in April")
plt.ylabel("Wind speed in m/s")
plt.xticks(np.arange(min(hours), max(hours)+1, 1.0))
plt.title("Wind Variation")
```

Time Integration of System of Equations

Initialisation of Matrices

[ ]:
```
T = np.zeros((len(temp)+1,1))
Q_CM = np.zeros((len(temp),1))
Q_EM = np.zeros((len(temp),1))
Q_SM = np.zeros((len(temp),1))
Q_M = np.zeros((len(temp),1))
m_M = np.zeros((len(temp)+1,1))
```

Initial Values

[ ]:
```
T[0] = T_now
m_M[0] = m
```

Start of Loop

[ ]:
```
for i in range(0,len(temp)):

    # break if pool goes below 0°C
    if T_now < 273.15:
        print("The pool froze!")
        m_M = m_M[0:i]
        T = T[0:i]
        Q_M = Q_M[0:i]
        Q_CM = Q_CM[0:i]
        Q_EM = Q_EM[0:i]
        Q_SM = Q_SM[0:i]
        break

    # fetch values for density and specific heat
    D = PropsSI('D','P',101325,'T',T_now,'Water')
```

```
    Cp = PropsSI('Cpmass','P',101325,'T',T_now,'Water')

    # convection term
    if convection==1:
        h_c = h_conv(wind[i])
        Q_C = - h_c * A * (T_now - temp[i] - 273.15) * 3600
        Q_CM[i] = Q_C

    # evaporation term
    if evaporation==1:
        V_ev = ev1(wind[i],T_now)
        m1 = m
        m2 = m - D*V_ev
        m3 = 0.5*(m2+m1)
        m = m - V_ev*D
        Q_E = -q_ev*ev1(wind[i],T_now)*D
        Q_EM[i] = Q_E
    else:
        m3 = m

    # solar term
    if solar==1:
        if control ==1:
            Q_S = shine[i]*eff*a
        elif control==0.5:
            if T_now<T_user:
                Q_S = shine[i]*eff*a
            else:
                Q_S = 0
        else:
            Q_S = 0
        Q_SM[i] = Q_S

    # energy balance
    Q = Q_S + Q_E + Q_C
    Q_M[i] = Q

    # temperature at next time
    delT = Q/(m3*Cp)
    T_next = T_now + delT
    T[i+1] = T_next
    T_now = T_next
    m_M[i+1] = m
```

Plot of Temperature of Pool Water

[ ]:
```
if convection or evaporation or solar:
    hour = np.linspace(1,len(T)/24+1,num=len(T))
    plt.figure(figsize=(20,5))
    plt.plot(hour,T-273.15,'r')
    plt.xlim([1,len(T)/24+1])
    plt.xlabel("Day in April")
    plt.ylabel("Temperature of Pool in °C")
    plt.xticks(np.arange(min(hour), max(hour), 1.0))
    plt.title("Temperature vs Time")
```

Plot of Heat Flux

[ ]:
```
if convection or evaporation or solar:
    hours = np.linspace(1,len(Q_M)/24 + 1,num=len(Q_M))
    plt.figure(figsize=(20,5))
    plt.plot(hours,Q_M/3600,'k',linewidth=2,label="Total Heat Flux")
    if convection==1:
        plt.plot(hours,Q_CM/3600,'b',linewidth=0.75,label="Convection")
    if evaporation==1:
        plt.plot(hours,Q_EM/3600,'g',linewidth=0.75,label="Evaporation")
    if solar==1:
        plt.plot(hours,Q_SM/3600,'r',linewidth=0.75,label="Solar Panel")
    plt.ticklabel_format(axis='y', style='sci', scilimits=(2,4))
    plt.xticks(np.arange(min(hours), max(hours)+1, 1.0))
    plt.xlim([1,len(Q_M)/24 + 1])
    plt.xlabel("Day in April")
    plt.ylabel("Heat Flux in W")
    plt.title("Heat Flux vs Time")
    plt.legend()
```

Plot of Mass of Pool Water

[ ]:
```
if convection or evaporation or solar:
    hours = np.linspace(1,len(m_M)/24 + 1,num=len(m_M))
    plt.figure(figsize=(20,5))
    plt.plot(hours,m_M,'g')
    plt.xlim([1,len(m_M)/24+1])
    plt.ylim([56000,60000])
    plt.xlabel("Day in April")
    plt.ylabel("Mass of Pool Water in kg")
    plt.xticks(np.arange(min(hours), max(hours), 1.0))
    plt.title("Mass of Pool Water vs Time")
```

Calculation of Heating Costs

[ ]:
```
Cp1 = PropsSI('Cpmass','P',101325,'T',T_now,'Water')
W = (m*Cp1*T_user - m*Cp1*T_now)/3600000
P = W*0.25

if solar == 0:
  print("Solar panel heating is turned off. It will take %.2f EUR to heat\
        to %.2f°C" %(P,T_user-273.15))

if solar == 1:
  if T_now > T_user:
    print("Solar panel heating is turned on.\nTemperature of the pool is",\
          "%.2f°C and the user-required temperature is %.2f°C." \
          %(T_now - 273.15, T_user_C),"\nHence, no heating required.")
  else:
    print("Solar panel heating is turned on.\nTemperature of the pool is",\
    "%.2f°C and the user-required temperature is %.2f°C.\nIt will take"\
    %(T_now-273.15,T_user_C),"%.2f EUR to heat to %.2f°C." %(P,T_user_C))

  T_now = T_now_C + 273.15; m = m_M[0]

  for i in range(0,len(temp)):
    D = PropsSI('D','P',101325,'T',T_now,'Water')
    Cp = PropsSI('Cpmass','P',101325,'T',T_now,'Water')

    if convection==1:
        h_c = h_conv(wind[i])
        Q_C = - h_c * A * (T_now - temp[i] - 273.15) * 3600

    if evaporation==1:
        V_ev = ev1(wind[i],T_now)
        m1 = m; m2 = m - D*V_ev; m3 = 0.5*(m2+m1)
        m = m - V_ev*D
        Q_E = -q_ev*ev1(wind[i],T_now)*D
    else:
        m3 = m

    Q = Q_E + Q_C
    delT = Q/(m3*Cp)
    T_next = T_now + delT
    T_now = T_next

  Cp1 = PropsSI('Cpmass','P',101325,'T',T_now,'Water')
  W = (m*Cp1*T_user - m*Cp1*T_now)/3600000
  P = W*0.25
  print("Without solar panel heating, it will take",\
        "%.2f EUR to heat to %.2f°C" %(P,T_user-273.15))
```