ISSN: 2449 - 0539
BAYERO JOURNAL OF ENGINEERING AND TECHNOLOGY VOL 18 NO. 3 AUGUST 2023 PP. 20-39

Programme development and simulation studies of a single slope solar still

Mohammed Tajudeen Jimoh

Department of Mechanical Engineering, Bayero University, Kano, Nigeria.

ABSTRACT

This study develops a computer programme using MATLAB® software for simulating a single slope solar still, based on published models for the solar still and its major inputs variable, the ambient temperature and the solar irradiance. Then the solar still is digitally studied for the effects of some variables on its performance. The effects of variables such as depth of water in the basin slope of glass cover, insulation thickness, solar radiation and wind velocity on factors such as the distillate yield, water temperature and efficiency of the still are investigated using derived models. The simulated trends show that the distillate yield, water temperature and efficiency increase with increase in insulation thickness, wind velocity and solar radiation. But decreasing water depth increases the above-mentioned parameters. Changes in glass cover slope have minimal effect on the still efficiency.

Keywords: Solar still; Solar still model; solar irradiance model; ambient temperature model; heat transfer coefficients.

1. INTRODUCTION

The global demand for water by many competing needs (drinking, washing, farming, construction, etc,), and the increasing global population, place enormous pressure on the little available sources of portable water. This is particular so in Africa where the greater majority reside in rural areas where access to treated water is limited. In most places in Africa, the available water sources are polluted or briny. There is therefore the need to expand the portable water sources for the population. One means by which the brackish or contaminated water can be made suitable for drinking is by distillation. Distillation is however, an energy intensive process. The common energy sources for distillation are electricity, fossil fuel (oil, gas), biofuel (wood). Solar is proving to be a viable alternative energy source for distillation. It is renewable, and less expensive when compared with conventional sources. Solar distillation technology is well suited for the rural regions, due to its simplicity and ease of maintenance (Jimoh, 2023).

Practically, solar distillation has been a low yield process. The practical distillate yield is between 2-5 litres/m^2 day (Mkhize & Msomi, 2023), though theoretical yield is reported to be higher (Hafs, et al., 2021). There have been many researches at investing the factors that affect the performance of solar still so as to understand how its efficiency can be improved. Both empirical studies, and analytical and numerical studies, based on mathematical models, have been employed in the investigations. Some of the recent works that employed analytical studies are (Dwivedi & Tiwari, 2009), (El-Sebaiia & Al-Dossarib, 2011), (Gupta, et al., 2013), (Yeo, et al., 2014), and (Torchia-Núñez, et al., 2014), (Ebaid & Ammari, 2015) (2015), (Johnson, et al., 2019) etc.

Though more detailed models of solar still are now available, some authors that have carried out its simulation studies make assumptions that simplify the analysis. For example, some assume that water physical properties remain constant throughout the simulation (Johnson, et al., 2019), some use inputs that makes use of average daily ambient temperature and solar irradiance (Tiwari, 2002), (Cooper, 1969), some assume that the water mass in the basin is constant while evaporation is taking place (Yeo, et al., 2014), other use ambient temperature and solar irradiance models that are over simplified (Torchia-Núñez, et al., 2014). Though these assumptions help to reduce the computational requirements of the simulation studies, they however detract from the transient nature of the distillation procedure, in relation to its primary inputs: the solar irradiance and ambient temperature.

(Jimoh, 2023), leveraging on the works of many researchers, presented a comprehensive mathematical model for single slope solar still. In addition, the work presented the models for estimating the solar irradiance and ambient temperature, both are vital inputs to the solar distillation process. The model provides a platform for comprehensive numerical studies of single slop solar still as a mainly transient phenomena. The models presented accounted for change in water mass in the still with time, variation in the specific heat capacities of the glass cover and the still basin, the difference in areas of cover glass and still and water surface, variation of latent heat of vaporisation of water with temperature, and some more.

This work develops the programme for simulating a single basin solar still, the associated ambient temperature and

solar irradiance, based on the models presented in (Jimoh, 2023), and then carries out a simulation study of the still. The simulation studies involve investigating the effects of parameters and variables such as ambient temperature, solar irradiance, water depth can be studied. The programme is based on MATLAB® software.

2. MATERIALS AND METHODS

2. 1 Models of single slope solar still and incident solar radiation

The heat and mass transfer process in a solar still is governed by its interaction with the solar irradiance and its environment (ambient temperature and air). Figure 1 illustrates the relationships between the solar irradiance incident on a single basin solar still, and of energy transfer within it.

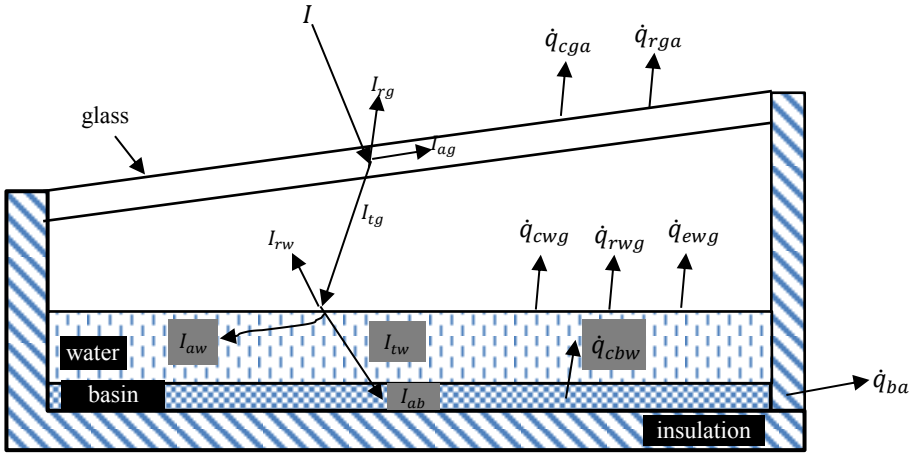

Figure 1: Heat transfer components in a typical single slope solar still **(Jimoh, 2023)**

The energy balance for glass cover, water mass in the basing, and still basin are given respectively as:

$$\frac{dT_g}{dt} = \frac{1}{m_g C_{pg}}\left[A_g I_{ag} + A_w h_{wg}(T_w - T_g) - A_g h_{ga}(T_g - T_a)\right] \quad (1)$$

$$\frac{dT_w}{dt} = \frac{1}{m_g C_{pb}}\left[A_w I_{aw} + A_b h_{bw}(T_b - T_w) - A_w h_{wg}(T_w - T_g)\right] \quad (2)$$

$$\frac{dT_b}{dt} = \frac{1}{m_b C_{pb}}\left[A_b I_{ab} - A_b h_{cwb}(T_b - T_w) - A_b h_{ba}(T_b - T_a)\right] \quad (3)$$

All the terms of these equations and the associated tables of parameters and expressions are contained in (Jimoh, 2023).

The beam and diffuse irradiance models, and hence the global irradiance models, are given respectively as:

$$I_B = I_{NE}\exp(-\tau_b m_a^{ab}) \quad (4)$$

$$I_D = I_{NE}\exp(-\tau_d m_a^{ad}) \quad (5)$$

$$I_G = I_b + I_d \quad (6)$$

Again the relevant terms of these equations and the associated tables of parameters and expressions are contained in (Jimoh, 2023).

The empirical relation for estimating the ambient temperature is:

$$T_a = T_{max} - f_h(T_{max} - T_{min}) \quad (7)$$

The relevant terms of these equations and the associated tables of parameters and expressions are contained in (Jimoh, 2023).

The instantaneous distillate yield and efficiency of the solar still are given receptively as:

$$\dot{m}_{ew} = \frac{\dot{q}_{ewg} L_s B_s}{L} = \frac{h_{ewg} L_s B_s (T_w - T_g)}{L} \quad (8)$$

$$\eta_i = \frac{\dot{q}_{ewg}}{I_{THG}} = \frac{h_{ewg}(T_w - T_g)}{I_{THG}} \quad (9)$$

The relevant terms of these equations and the associated tables of parameters and expressions are contained in (Jimoh, 2023)

2.2 Algorithm for solving Still ODEs

At every time step, the three ODE of the solar still have to be solved. The numerical method for solving the ODEs is

the fourth order Runge-Kutta method. For an ODE defined as $T'(t) = f(T)$ can be written as (Yang, et al., 2005):

$$T_{k+1} = T_k + \frac{h}{6}(f_{k1} + 2f_{k2} + 2f_{k3} + f_{k4}) \quad (10)$$

where:

$$f_{k1} = f(T_k)$$
$$f_{k2} = f(T_k + 0.5hf_{k1})$$
$$f_{k3} = f(T_k + 0.5hf_{k2})$$
$$f_{k4} = f(T_k + hf_{k3})$$

2.3. Programming codes for the models coding and simulation procedures

The computer simulation of the solar still model is carried out using MATLAB software. Basically, the simulation involves the calculation, at every second, the solar still performance indicators – the instantaneous temperatures of glass (T_g), water (T_w), and basin (T_b), as well as the distillate yield (dy) and efficiency (η), for given solar still configuration and within a given period during the day. These calculations require information about the solar still primary inputs - the ambient temperature, T_a and the global solar irradiance, I_G - as well as its many solar parameters, some of which are day dependent, day and location dependent, and time dependent.

In order to aid fast computations and to avoid repetition of calculations, a compartmentalised computer programme approach involving seven MATLAB functions and a script, is adopted. The flowchart showing the outline of the program codes for the functions is shown in figure 2. The sub-Sections in this Section describe the programme compartments and the functions, and the simulation procedures for the model. The codes for the functions are given in the Appendix.

2.3.1 Simulating day-dependent parameters

Some solar parameters, such as declination angle, are constant for all time in a given day, and can be calculated separately. The MATLAB function, `day_solarparams`, based on global solar irradiance models in (Jimoh, 2023), is created to simulate such parameters. The syntax for the function is:

```
[delta,eqts,NE,taubn,taudn] =
day_solarparams(Isc,ndate,smdats)
```

The input arguments of the function, their descriptions, as well as nominal values used in the simulation, are described in Table 1. Likewise, the output arguments of the function, their descriptions, as well as samples of the outputs, are described in Table 2.

2.3.2 Simulating location (latitude) dependent parameters

Some solar parameters, such as sunrise and sunset times, depend on a location's latitude angles for a given day. And information about the sunrise and sunset times can be used to simulate a vector of air mass values for every second between sunrise and sunset for that day. To simulate such parameters, we do not have to recompute daily parameters such as declination angle. The MATLAB function `loc_solarparams`, based on global solar irradiance models in (Jimoh, 2023), is created and used to simulate vectors of sunrise to sunset time (in seconds), and the corresponding air mass values. The syntax for the function is:

```
[tsrss,masrss] =
loc_solarparams(delta,phi)
```

The input arguments of the function, their descriptions, as well as nominal values used in the simulation, are described in Table 3. Likewise, the output arguments of the function, their descriptions, as well as samples of the outputs, are described in Table 4.

Table 1: Input arguments for function **`day_solarparams`**

Input Arguments	Model symbol	Description	nominal value/ source
`Isc`	I_{SC}	Solar constant	$1366\ W/m^2$
`ndate`		Date for which simulation is done	$'15: APR: 2018'$
`smdats`		Stored monthly data (for no of days in a month and beam and diffuse optical depths	Similar to Table 6

Table 2: Output arguments for function **`day_solarparams`**

Output Arguments	Model symbol	Description	Output value	Target function
`delta`	δ	Declination angle	9.4149^o	`loc_solarparams`
`eqts`	E_{ts}	Equation of time in seconds	$-14.1846s$	`prd_solarparams`
`NE`	I_{NE}	Extra-terrestrial beam irradiance	$1356.4\ W/m^2$	`prd_irrads`
`taubn`	τ_b	beam optical depth	0.3795	`prd_irrads`
`taudn`	τ_d	diffuse optical depth	2.3455	`prd_irrads`

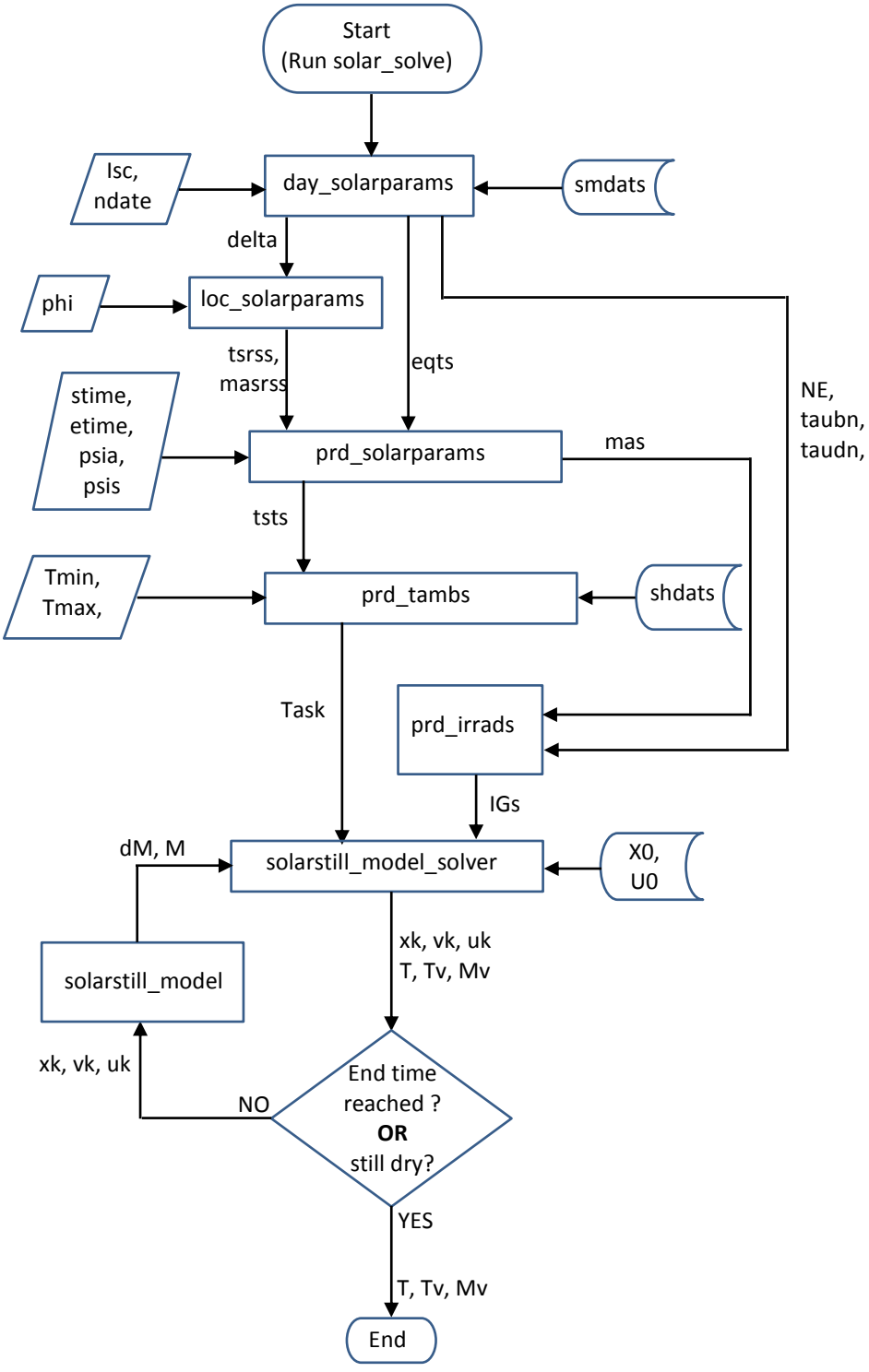

Figure 2: Flow chart of the programme flow for the solar still simulation

Table 3: Input arguments for function **`loc_solarparams`**

Input Arguments	**Model symbol**	**Description**	**nominal value (source function)**
`delta`	δ	Declination angle	`delta` (output of `loc_solarparams`
`phi`	ϕ	Latitude angle	12.0^{o}

Table 4: Output arguments for function **`loc_solarparams`**

Output Arguments	Description	Target function
`tsrss`	Solar time vector, in seconds, between sunrise and sunset	`prd_solarparams`
`masrss`	Vector of air mass values for every second, between sunrise and sunset	`prd_solarparams`

2.3.3 Simulating time period dependent parameters

On feature of the codes for simulating solar still in this paper is that the user must specify the start time and end time (for a day) of the simulation. Such times must be within the sunrise and sunset times for that location. For given local time period, its equivalent solar time period, which depends on the locations longitudes, must be calculated. The MATLAB function `prd_solarparams`, based on global solar irradiance models in (Jimoh, 2023), is used so simulate the vector of solar time (period between sunrise time and sunset time), in seconds) and the equivalent air mas vector, for the period.

Table 5: Input arguments for function `prd_solarparams`

Input Arguments	Model symbol	Description	Nominal value (source function)
`stime`		Start time	`'07:00:00'`
`etime`		End time	`'18:00:00'`
`eqts`	E_{ts}	Equation of time	`eqts` (output of `day_solarparams`
`psis`	ψ_s	Standard longitude for the location	`15`
`psia`	ψ_a	Actual longitude of the location	`8.592`
`tsrss`	t	Vector of solar time, in seconds, between sunrise and sunset	`tsrss` (output of `loc_solarparams`
`masrss`	m_a	Vector of air mass values for every second between sunrise and sunset	`masrss` (output of `loc_solarparams`

Table 6: Output arguments for function `prd_solarparams`

Output Arguments	Model symbol	Description	Target function
`tsts`	t	Solar time vector, in seconds, for the specified period	`prd_tambs`
`mas`	m_a	Vector of air mass values for every second for the specified period	`prd_irrads`

Table 7: Input arguments for function **`prd_tambs`**

Input Arguments	Model Symbol	Description	Value (source function)
`Fhh`	f_h	Stored data of the hourly fraction of the daily temperature range	Similar to Table 8
`Tsts`	t	Solar time vector, in seconds, for the specified period	`tsts` (output of `prd_solarparams`
`Tmin`	T_{min}	Minimum temperature, in Celsius, for the specified day	27.3^oC
`Tmax`	T_{max}	Maximum temperature, in Celsius, for the specified day	39.9^oC

Table 8: Output arguments for function **`prd_tambs`**

Output Arguments	Description	Target function
`Tasc`	Vector of ambient temperature values for specified period, in degrees Celsius	
`Task`	Vector of ambient temperature values for specified period, in Kelvin	`solarstill_model_solver`

The syntax for the function is:

```
[tsts,mas] =
prd_solarparams(stime,etime,eqts,psis,ps
ia,tsrss,masrss)
```

The input and output arguments of the function, as well as nominal values used in the simulation, are described in Tables 5 and 6. Figure 3 shows the plot of air mass against time period specified for the chosen location and day. The value of air mass is lowest at solar noon (when the sun is directly overhead) and have its highest values close to sunrise and sunset.

2.3.4 Simulating the ambient temperature

The ambient temperature is required in the calculation of the evaporative and radiative heat transfer from the glass cover to the ambient. The data can be computed a priori, for every second within a specified time period, and be used during the actual still simulation. Based on equation (7) and associated equations and Table of global solar irradiance models in (Jimoh, 2023), a MATLAB function, `prd_tambs`, is developed. The syntax for the function is:

```
[Tasc,Task] =
p_prd_tambs(fhh,tsts,Tmin,Tmax)
```

The input and output arguments of the function are described in Tables 7 and 8.

The trends of the ambient temperature with respect to time for a chosen day (15th April, 2018, where maximum and minimum temperatures are 39.9°C and 27.3°C respectively, are shown figure 4.

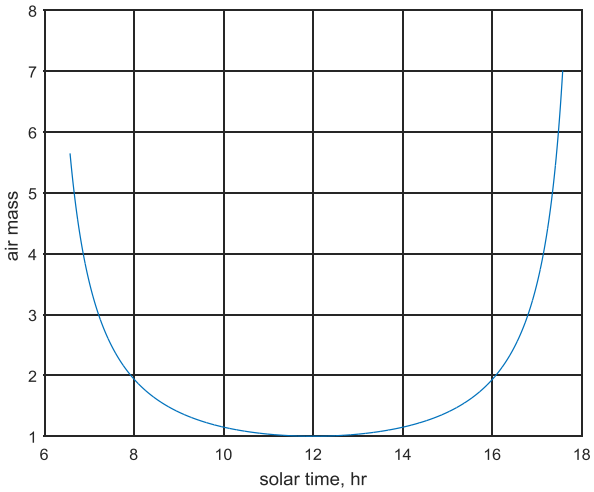

Figure 3: plot of air mass against solar time

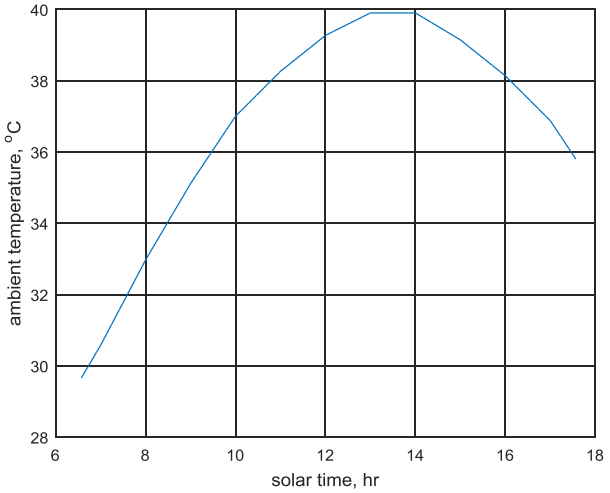

Figure 4: simulated trend of ambient temperature vs solar time

2.3.5 Simulating solar irradiance

Solar irradiance is the primary energy source for the still. Again, the irradiance data can be computed a priori, for every second within a specified day and time period, and be used during the actual still simulation. Based on equations (4, 5 6) and other equation associated Equations and Tables in (Jimoh, 2023), a MATLAB function, `prd_irrads`, is developed. The syntax for the function is:

```
IGs = prd_irrads(taubn,taudn,EN,mas)
```

The output and input arguments of the function are described in Tables 9 and 10. The trends of the solar irradiance with respect to time for a chosen day (24th April, 2018, are shown in figure 5

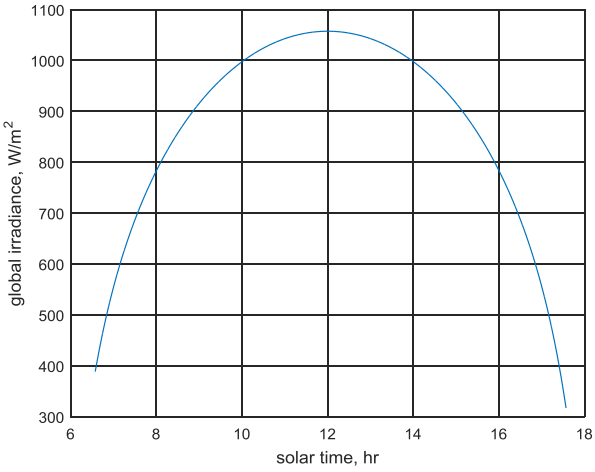

Figure 5: Simulated trend of solar irradiance vs solar time

Table 9: Output arguments for function `prd_irrads`

Output Argument	Model symbol	Description	Target function
`IGs`	I_G	Global Irradiance vector, in W/m^2, for every second between start time and end time	`solarstill_model_solver`

Table 10: Input arguments for function `prd_irrads`

Input Arguments	Description	Value (source function
`taubn`	Beam optical depth value for the day	`taubn` (output of `day_solarparams`)
`taudn`	Diffuse optical depth value for the day	`taudn` (output of `day_solarparams`)
`NE`	Extra-terrestrial normal irradiance for the day	`NE` (output of `day_solarparams`)
`mas`	Air mass vector for each second of the specified period	`mas` (output of `prd_solarparams`)

2.3.6 Simulating the still performance

Simulation of the still performance involves the prediction of the every-second values of the glass, water and basin temperature values, as well as the still yield, depth and efficiency. To achieve this, two MATLAB functions, `solarstill_model`, and `solarstill_model_solver`, and a MATLAB script, `solarstill_solve`, are developed.

The `solarstill_model` function is used to code the differential equations 1, 2 and 3, equations 8 and 9 governing the yield and efficiency of the still, as well as all other equations governing the operation of the still as contained in ,. The function `solarstill_model_solver` is to code the 4th order Runge-Kutta ODE solver (based on equation 10) for solving the differential equations of the `solarstill_model` function, as well as evaluating the instantaneous yield and efficiency of the solar still as contained in the function.

The syntaxes for the two functions are:

```
[dT,M] = solarstill_model(xk,vk,uk)
[t,Tv,Mv] = solarstill_model_solver(f,tsts,vv,x0,u0)
```

Tables 11 to 14 show a description the input and output arguments of the two functions.

Table 11: Input arguments for function `solarstill_model`

Input Arguments	Description	Source function
`xk`	Row vector of ODEs state values for glass, water and basin ([Tg, Tw, Tb]) at an instant	`solarstill_model_solver`
`vk`	Row vector of ambient temperature and solar irradiance ([Ta, Ig]) at an instant	`solarstill_model_solver`
`uk`	Row vector of still depth and distillate yield ([dw, dy]) at an instant	`solarstill_model_solver`

Table 12: Output arguments for function `solarstill_model`

Output Arguments	Description	Target function
`dT`	Row vector of the expression evaluation the three still differential equations 1, 2 and 3, at an instant	`solarstill_model_solver`
`M`	Row vector of expression evaluation water depth, cumulative distillate yield, instantaneous distillate yield, heat of evaporation, and efficiency ([dw, dy, idy, qew, eff]), at an instant	`solarstill_model_solver`

Table 13: Input arguments for function **`solarstill_model_solver`**

Input Arguments	Description	Value or source function
`f`	Refence to the function `solarstill_model`	`solarstill_model`
`tsts`	Solar time vector, in seconds, for the specified period	`tsts` (output of `prd_solarparams`
`vv`	Matrix of ambient temperature (first column) and irradiance (second column),	`[Task, IGs]` (outputs of `prd_tamb` and `prd_irrads`)
`x0`	Row vector of initial temperature of glass, water and basin	[293.15,303.15,313.15]
`u0`	Initial depth of water (metres)	0.01

Table 14: Output arguments for function **`solarstill_model_solver`**

Output Arguments	Description
`t`	Solar time vector, in seconds during the period of operation of the still (may have a length less than or equal to `tsts`)
`Tv`	Matrix of per second values of temperatures of the glass (first column), water (second column) and basin (third column)
`Mv`	Matrix of per second values of water depth (first column), cumulative distillate still yield (second column), distillate yield (third column), heat of evaporation (fourth column) and efficiency (fifth column)

The script `solarstill_solve` serves as the parent script for the functions, the simulation nominal values, and the solar still parameters. All the other functions created are called and implemented when the script is run. The description and values of other fixed parameters used for the simulation are contained in (Jimoh, 2023),.

3. RESULTS AND DISCUSSIONS

There are many variables that are considered to have effect on the performance of solar still in terms of its instantaneous water temperature, distillate yield, efficiency, etc. These variables may be grouped into disturbance variables (inputs variables that are not directly amenable to manipulation, like wind velocity, or season of the year), or manipulaple variables (inputs variables that can be directly manipulated, like the initial depth (or initial mass) of water, the water -glass spacing, the materials of construction - the types of glass cover, the type of absorber and insulation, etc. While this simulation is capable of studying the effect of these and more variables, only four - the season, the wind velocity, the initial depth of water and the water-glass spacing - are investigated here. The observed trends of the effects of the variables on the performance of solar still are presented and discussions.

First, the model is simulated using the nominal parameter values of Table 15 and other input values indicated in the relevant tables above. The observed trends of the performance indices (water temperature, instantaneous depth of water, cumulative distillate yield and instantaneous efficiency) are as shown in Figure 6. The observed trends show that for a system operated between sunrise and sunset, the water temperature may reach 80^{o}C (Figure 6a), the distillate yield may be up to 8 kg/m^2 (Figure 6b) and that instantaneous efficiency may be more than 60% (Figure 6d). The spike shown by the efficiency plot of Figure 6(d) indicates the fact that even towards sunset, when the irradiance is significantly low, distillate production continues because of the already elevated temperature of water, and its increased thermal capacitance, the heat of evaporation and hence distillate yield is very high relative to the irradiance. These trends depend on the specifics of the solar still in terms of inputs and materials of construction.

The trends largely follow the pattern of those presented in some previous works (Afrand, et al., 2017), (Yeo, et al., 2014), (Ahsan, et al., 2011), (Cooper, 1969), (Johnson, et al., 2019), (Bao, 2019), (Ebaid & Ammari, 2015), though there are specifics that will be pointed out in the sub-Sections below

The fact that the observed trends follow which indicates that the computer programme can be used for further simulation studies. Now the effect of other variables on the performance of the still investigated by the simulation studies, as presented in the following sub-Sections.

3.1 Effect of Season of the year

One way to study the effect of ambient temperature (and irradiance) is to simulate for different seasons of the year. One feature of this programme is that when a date is given, a representative data of ambient temperature and irradiance for that date can be simulated. For this study, the effect of the cold dry weather of January, the hot and humid weathers of April and July, and the moderately warm weather of October, on solar still performance, are studied. For the chosen dates, their maximum and minimum temperatures are required. The input data for the chosen dates are based on weather data of (NIMET, 2018), given in Table 16.

Table 16: Min and Max temperature data for selected days **(NIMET, 2018)**

Date	T_{min} (^0C)	T_{max} (^0C)
'15-JAN-2018'	11.8	28.4
'15-APR-2018'	27.3	39.9
'15-JUL-2018'	22.7	30.2
'15-OCT-2018'	22.2	35.8

The trends of simulated water temperature, instantaneous depth of water, distillate yield and heat of evaporation for different seasons of the year are shown in Figure 7, while the simulated day available irradiation and still efficiency are shown in Figure 8.

Figure 7 shows that the month of April, representing a very hot and dry season provides the best condition for solar still performance, while the month of July, moderately hot and humid season, provided the least favourable condition for solar still performance. The distillate yield in this day in April, 7.94 kg, is 17.86% more than that of July, which is

6.52 kg. This is supported by the fact that the available irradiation in April, 34.59 MJ (Figure 8), is 12.69% more than the month of July, which is 30.20 MJ.

In (Cooper, 1969), their simulation predicted that increase in ambient temperature will result in minimal increase in the yield of solar still. But it is seen from the current simulation that it is difficult to separate the effect of ambient temperature from solar irradiance, since both occur simultaneously. Here though, it is observed that though July has relatively higher ambient temperature that January (ambient temperature trend not shown), the distillate yield in JANUARY 15th (Figure 8) is higher than that of July 15th. It is observed from the figure that January has a higher irradiation value than that of July. So, it may be deducted that irradiance (or irradiation) has more dominant effect on yield than ambient temperature.

3.2 Effect of Wind Velocity

To simulate the effect of ambient air velocity, air velocities of 3, 5, 7.5 and 10 m/s are simulated for the nominal month of April. The trends of simulated water temperature, instantaneous depth of water, distillate yield and heat of evaporation for these air velocities are shown in Figure 9, while the simulated day still efficiency is shown in Figure 10. Figure 9a shows that though velocity seems to have noticeable effect on water temperature, its effect on the distillate yield and instantaneous efficiency (Figure 9b, c) is minimal. Increase in wind velocity increases instantaneous efficiency lightly at the beginning but have a reverse effect as the day progresses (Figure 9d). But the day efficiency increases minimally as the wind velocity increases (Figure 10). The trends are in conformity with the predictions from earlier simulations. This is largely in agreement with the observations of (Yeo, et al., 2014), (Cooper, 1969).

3.3 Effect of Initial water depth

For the nominal month of April, the effect of water depths was simulated with water depth values of 1 cm, 2 cm, 3 cm and 4 cm. The trends of simulated water temperature, instantaneous depth of water, distillate yield and efficiency for these water depths are shown in Figure 11, while the corresponding simulated day still efficiency is shown in Figure 12. Figure 11 shows that water depth have significant effect on all the solar still performance indices. The major factor for this significant effect can be attributed to the water capacitance.

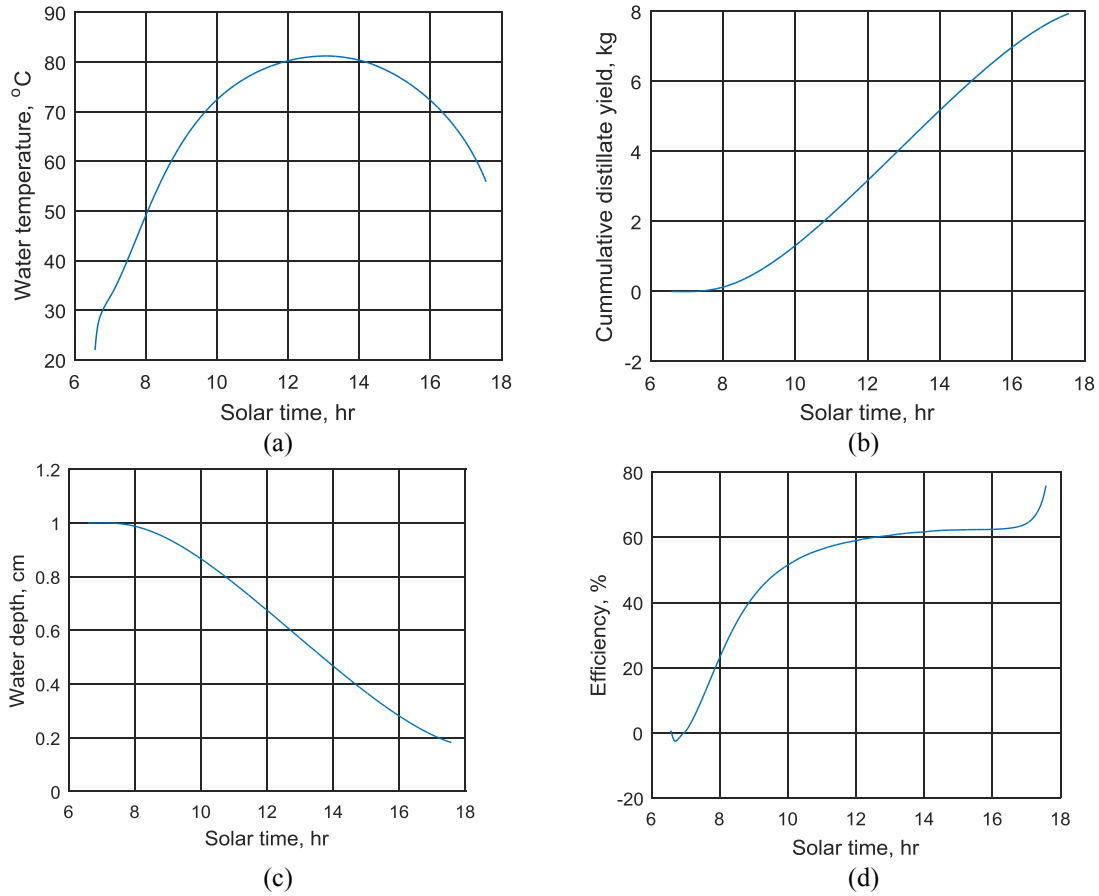

Figure 6: Trends of simulated solar still performance indices

Because of increasing water capacitance with increased in the depth (or mass) of water, the water temperature rise time is much longer for 4 cm water depth than it is for 1 cm water depth. For example, it takes about two hours longer

for the water to reach its peak temperature for a 4 cm water depth compared with a 1 cm water depth (Figure 11a). Likewise, the day distillate yield of 1 cm water depth, about 8 kg, is about 25% higher than for a depth of 4 cm, which is about 6 kg (Figure 11c). At 1 cm water depth (10 kg water), only about 0.18 cm depth (1.8 kg water, or 18% of initial water mass) is left un-distilled, while for a 4 cm water depth (40 kg water), 3.4 cm depth is un-distilled (34 kg water, or about 85% of initial water), is un-distilled. Figure 12 shows that the daily efficiency decreases with increase in water depth. This is in line with observations of (Johnson, et al., 2019), (Yeo, et al., 2014) and (Cooper, 1969).

3.4 Effect of Water-glass spacing

In literature, evaporation correlations for water-glass spacing between 0.15 m to 0.25 are reported So for this study the effect of glass-water spacing for 0.25 m, 0.20 m and 0.15 m on the solar still performance indices, are simulated, since for any value outside this range, new correlations may have to be developed. The trends of the simulated effect of water-glass spacing on water temperature, instantaneous depth of water, distillate yield and efficiency are shown in Figure 13. The figure shows minimal effect on performance for different water-glass spacings.

Likewise, Figure 14 shows that daily efficiency is minimally affected by water-glass spacing. So for the range where evaporation correlations exist, there is insignificant difference in performance when any of these values is used

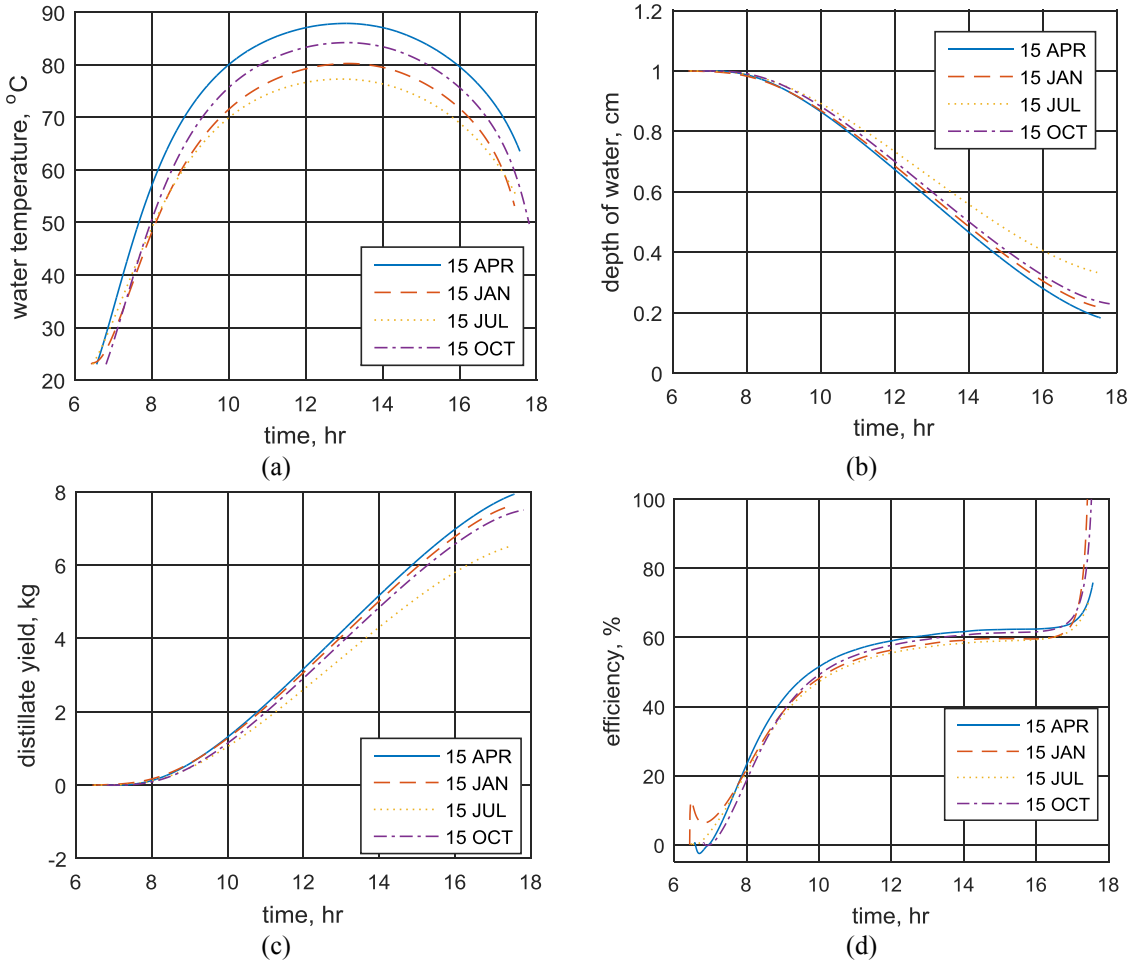

Figure 7: Simulated trends of solar still performance variables for different seasons of the year

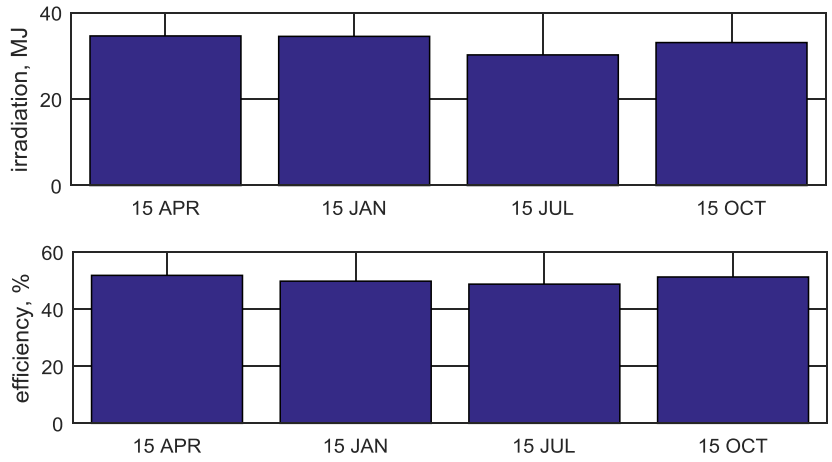

Figure 8: Simulated trends of available day irradiation and efficiency for different seasons of the year

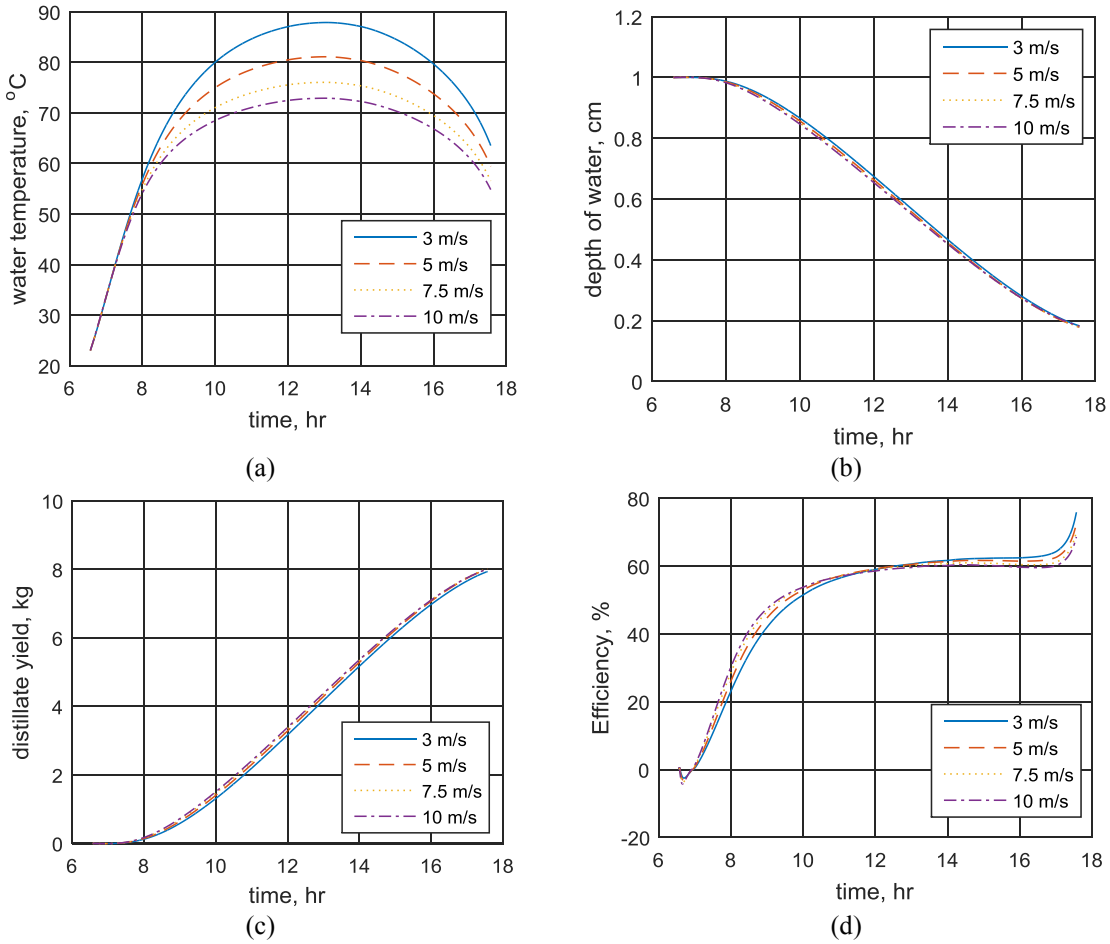

Figure 9: Simulated trends of solar still performance variables for different wind speeds

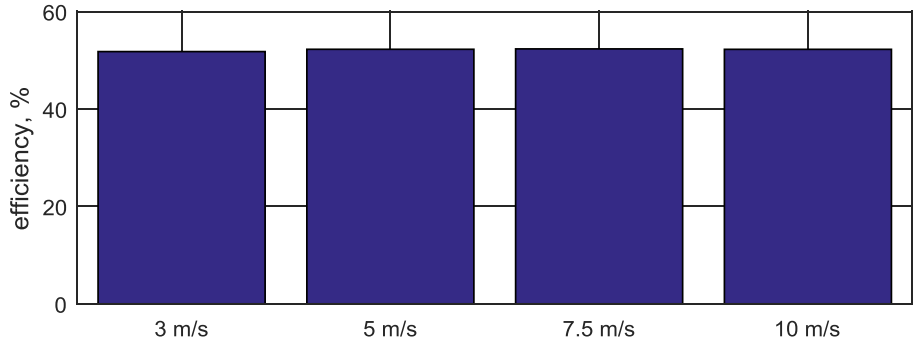

Figure 10: Simulated trends of solar still day efficiency for different wind speeds

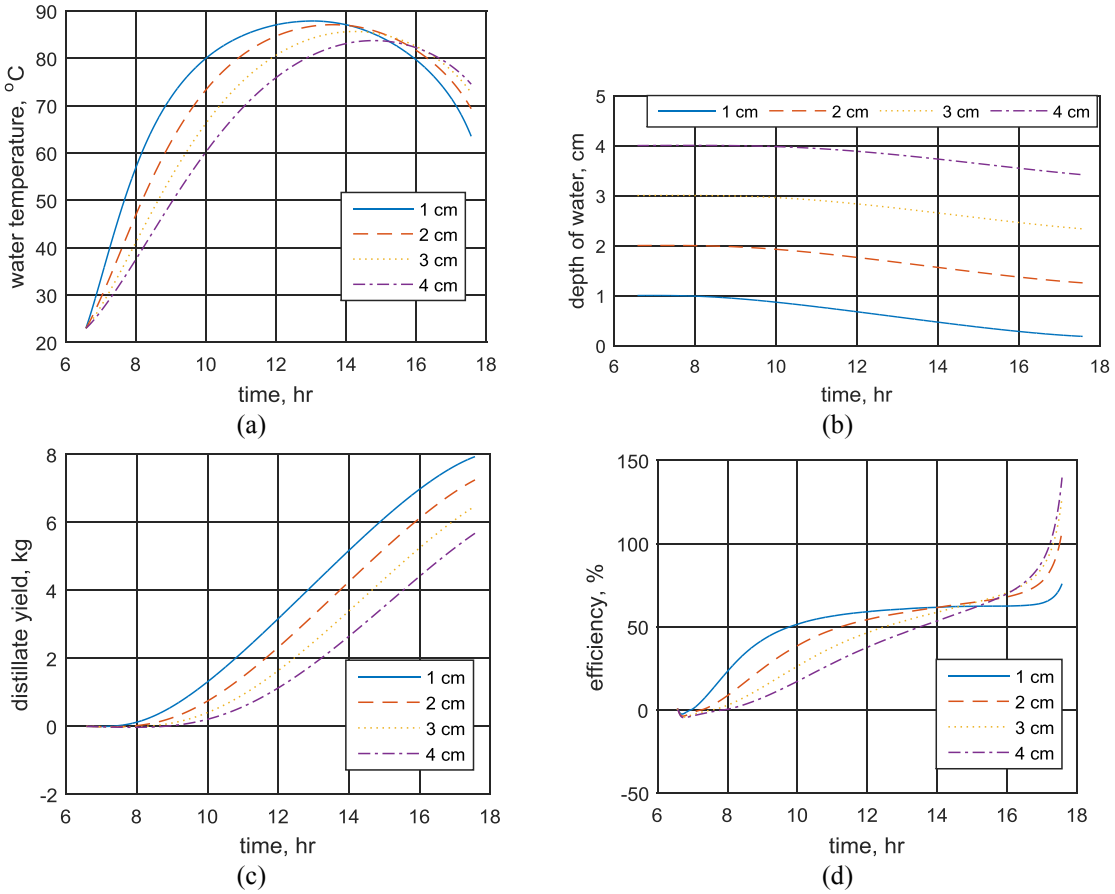

Figure 11: Simulated trends of solar still performance variables for different water depths

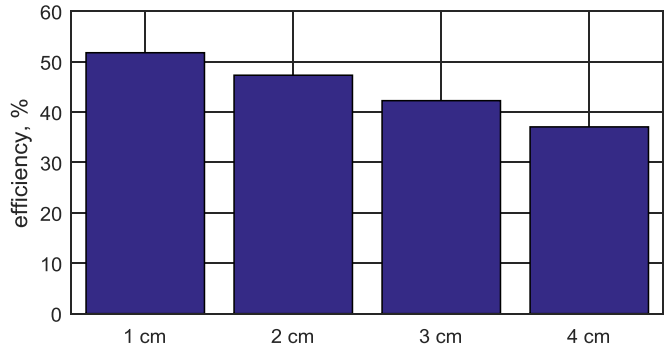

Figure 12: Simulated trends of solar still day efficiency for different water depths

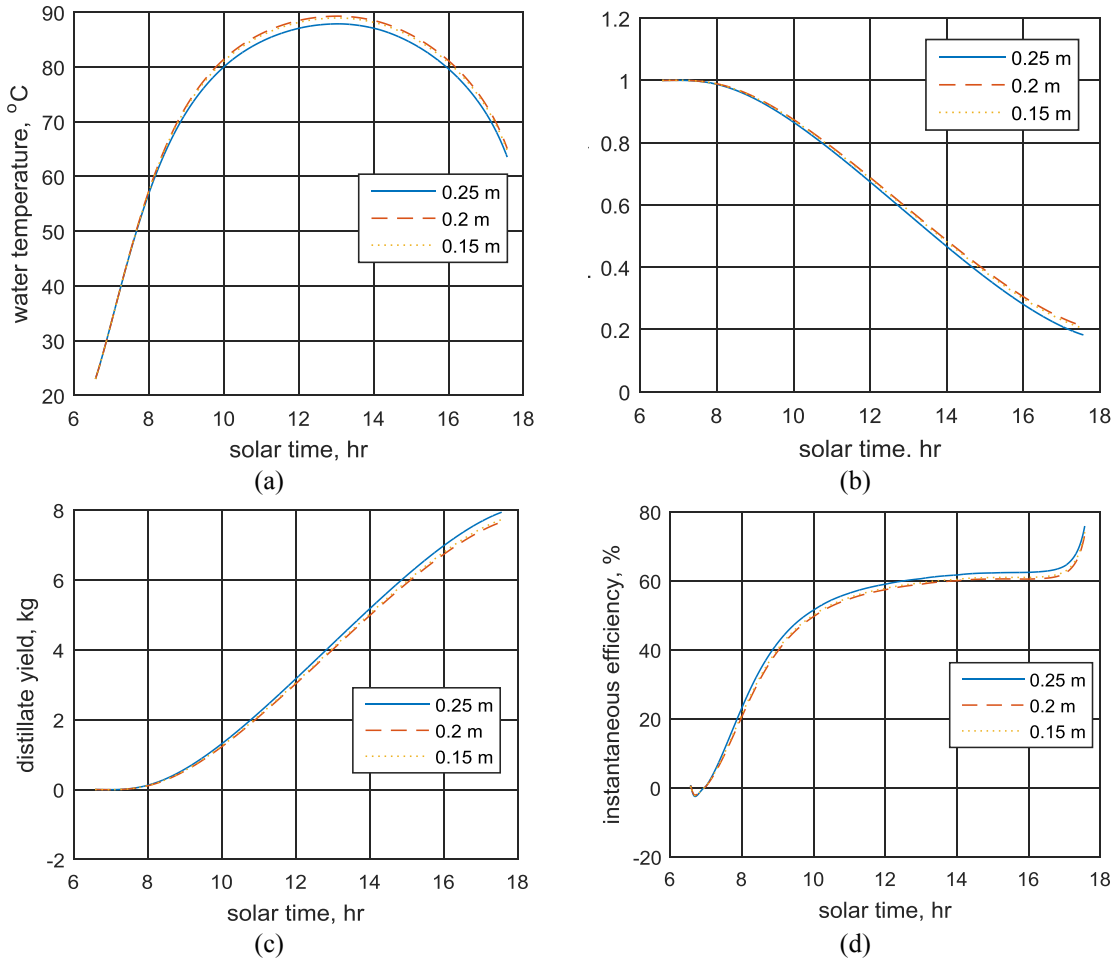

Figure 13: Simulated trends of solar still performance variables for different water-glass spacing

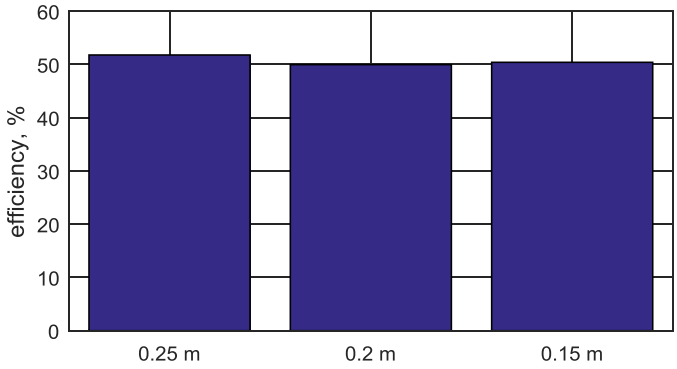

Figure 14: Simulated trends of solar still heat of evaporation and day efficiency for different water-glass spacing

.

4. CONCLUSIONS

MATLAB programmes for simulating the models were developed and implementation. With these, the effect of four variables, the season of the year, the wind velocity, the water depth and the water-glass spacing have been simulated and studied. The trends from the simulations show that water depth has the most significant effect on the performance of solar still, followed by the season of the year. Wind velocity and water-glass spacing have minimal effects on the performance of solar still. The developed MATLAB codes provides a platform for the investigation of the effects of other solar still parameters such as the glass cover type (reflectivity, absorptivity, transmissivity), basin types, bottom loss coefficients, and so on.

REFERENCES

Afrand, M., Kalbasi, R., Karimipour, A. & Wongwises, S., 2017. Experimental investigation on a thermal model for basin solar still with an external reflector. *Energies,* 10(18), pp. 1-16.

Ahsan, A. et al., 2011. Evaporation Phenomenon Inside a Solar Still: From Water Surface to Humid Air.. In: *Evaporation, condensation and heat transfer.* s.l.:IntechOpen, pp. 1-22.

Bao, N. T., 2019. The Mathematical Model of Basin-Type Solar Distillation Systems. In: *Distillation - Modelling, Simulation and Optimization.* s.l.:IntechOpen, pp. 1-17.

Cooper, P. I., 1969. Digital Simulation of Transient Solar Still Processes. *Solar Energy,* Volume 12, pp. 313-333.

Dwivedi, V. K. & Tiwari, G. N., 2009. Comparison of internal heat transfer coefficients in passive solar still by different thermal models: an experimental validation.. *Desalination,* Volume 246, pp. 304-318.

Ebaid, M. S. Y. & Ammari, H., 2015. Modelling and analysis of unsteady-state thermal performance of a single-slope tilted solar still. *Renewables,* pp. 2-19.

El-Sebaiia, A. & Al-Dossarib, M., 2011. A mathematical model of single basin solar still with an external reflector. *Desalination and Water Treatment,* Volume 26, p. 250–259.

Gupta, B., Mandraha, T. K., Edla, P. J. & Pandya, M., 2013. Thermal Modelling and Efficiency of Solar Water Distillation: A Review.. *American Journal of Engineering Research (AJER),* 02(12), pp. 203-213.

Hafs, H. et al., 2021. Numerical simulation of the performance of passive and active solar still with corrugated absorber surface as heat storage medium for sustainable solar desalination technology. *Groundwater for Sustainable Development,* Volume 14, pp. 1-17.

Jimoh, M. T., 2023. Mathematical modelling of single slope solar still with ambient temperature and global irradiance. *Bayero Journal of Engineering and Technology,* 18(2), pp. 104-112.

Johnson, A. et al., 2019. A thermal model for predicting the performance of a solar still with Fresnel Lense. *Water,* pp. 1-20.

Johnson, A. et al., 2019. A Thermal Model for Predicting the Performance of a Solar Still with Fresnel Lens. *Water,* pp. 1-20.

Mkhize, M. M. & Msomi, V., 2023. Year-Round Experimental Analysis of the Productivity of Vapour-Based Multistage Solar Still: A Developmental Study. *Hindawi Journal of Renewable Energy,* Volume 2023, pp. 1-15.

NIMET, 2018. *Nigerian Meteorolological Agency.* s.l.:s.n.

Tiwari, G. N., 2002. *Solar Energy: Fundamentals, Design, Modelling and Applications.*. New Delhi: Narosa Publishing House.

Torchia-Núñez, J. C., Cervantes-de-Gortari, J. & Porta-Gándara, M. A., 2014. Thermodynamics of a Shallow Solar Still,. *Energy and Power Engineering,* Volume 6, pp. 246-265.

Yang, W. Y., Cao, W., Chung, T.-S. & Morris, J., 2005. *Applied Numerical Methods Using MATLAB.*. Hoboken, New Jersey: John Wiley & Sons, Inc., .

Yeo, K. B., Ong, C. M. & Teo, T. K., 2014. Heat Transfer Energy Balance Model of Sigle Slope Solar Still. *Journal of Applied Sciences,* 14(23), pp. 3344-3348.

Appendix

MATLAB codes for simulating the solar still

A1: Function `day_solarparams` code

```
function [delta,eqts,NE,taubn,taudn] = day_solarparams(Isc,ndate,smdats)
%This function calculates daily solar parameters

%calculation of declination angle and nth day of the year
YM = smdats(:,2);
taub = smdats(:,3);
taud = smdats(:,4);
ndy = sum(YM);
dm = day(ndate);% day of the  month
my = month(ndate);%month of the year
y = year(ndate);%year
if rem(y,4) == 0  %leap y ear
    YM(2,1) = 29;
    ndy = sum(YM);
end
n = ones(1,my)*YM(1:my,1) - YM(my,1) + dm;
delta = 23.45*sind((360/ndy)*(284+n));%eqn(28)

%calculating equation of time in seconds
gam = 360*(n-1)/365;%eqn(32)
eqts = (432/pi)*(0.0075 + 0.1868*cosd(gam) - 3.2077*sind(gam)...
     - 1.4615*cosd(2*gam) - 4.089*sind(2*gam));% eqn (31)

% calculating air mass and irradiance
NE = Isc*(1 + 0.033*cosd(360*n/365));%eqn (23)

%calculating beam and diffuse optical depth for all days of the year
%(from Table 6)
taubd = zeros(ndy,1);
taudd = taubd;
taubd(1:20,1) = linspace(taub(12),taub(1),20);
taudd(1:20,1) = linspace(taud(12),taud(1),20);
a = 21;
b = 51;
for k = 1:11
    taubd(a:b,1) = linspace(taub(k),taub(k+1),YM(k));
    taudd(a:b,1) = linspace(taud(k),taud(k+1),YM(k));
    a = a+YM(k);
    b = b+YM(k+1);
end
c = ndy-10;
taubd(c:ndy,1) = linspace(taub(12),taub(1),11);
taudd(c:ndy,1) = linspace(taud(12),taud(1),11);
taubn = taubd(n); % beam optical depthfor the nth day
taudn = taudd(n);% diffuse optical depth for the nth day
```

A2: Function `loc_solarparams` code

```
function [tsrss,masrss] = loc_solarparams(delta,phi)
% This function returns  vectors of location (latitude) specific solar
% parameters for a day

 %calculating sunrise and sunset angles of the day
w1 = tand(phi)*tand(delta);
omgsr = -acosd(-w1);
```

```
omgss = -omgsr;
if w1 >=1
    omgsr = -180;
    omgss = 0;
elseif w1 <=-1
    omgsr = 0;
    omgss = 180;
end

%calculating solar times equivalents for  sunrise and sunset times, second
srs = ceil((omgsr/15 + 12)*3600); %sunrise time, seconds
sss = floor((omgss/15 + 12)*3600); %sunset time, seconds
tsrss = (srs:sss)';

%time vectors, hour angles and air mass for solar times
omgs = 15*((tsrss/3600) - 12);%hour angles
czags = cosd(phi)*cosd(delta)*cosd(omgs) + sind(delta)*sind(phi);
zags = acosd(czags);
masrss = czags.^-1 + 0.50572*(96.07995-zags).^(-1.6364);  %air mass
```

A3: Function `prd_solarparams` code

```
function [tsts,mas] = ...
                  prd_solarparams(stime,etime,eqts,psis,psia,tsrss,masrss)
% This function returns the vectors of solar parameters within a period

srs = tsrss(1);
sss = tsrss(end);
%calculating seconds equivalent of given local start time and end time
lsts = hour(stime)*3600 + minute(stime)*60 + second(stime);
lets = hour(etime)*3600 + minute(etime)*60 + second(etime);

%calculating solar seconds equivalent of start and end times
ssts = ceil(lsts + 240*(psia-psis) + eqts);% solar start time, seconds
sets = floor(lets + 240*(psia-psis) + eqts); %solar end times seconds

%testing for the correctness of the given start and end times
if ssts > sets
    error('start time must be less that end time')
elseif ssts < srs || ssts > sss
    error('start time not within sunrise and sunset period')
elseif sets < srs || sets > sss
     error('end time not within sunrise and sunset period')
end

%time vectors, hour angles and air mass for solar times
tsts = (ssts:sets)';% solar time vector, seconds

y1 =  find(tsrss==ssts);
y2 =  find(tsrss==sets);
y = (y1:y2)';
mas = masrss(y);
```

A4: Function `prd_tambs` code

```
function [Tasc,Task] = prd_tambs(fhh,tsts,Tmin,Tmax)
% This function returns  the vector  of per second ambient  temperature
% for a given period in a day

fh = fhh(:,2);
tds = (1:24*3600)';% daily time vector, seconds
fh = [fh;fh(1)];
fhs = zeros(size(tds));% to store per second values
for i = 1:24
    c = linspace(fh(i,1),fh(i+1,1),3600)';
    n1 = (i-1)*3600+1;
    n2 = i*3600;
```

```
    fhs(n1:n2,1) = c;
end
%calculating per second values of daily ambient temperature
Tads = Tmax-fhs*(Tmax-Tmin);%
td = (1:24*3600)';
t1 = find(td == tsts(1));
t2 = find(td == tsts(end));
tah = (t1:t2)';
Tasc = Tads(tah);% ambient teperature, celcius
Task = Tasc + 273.15;% ambient temperature, Kelvin
```

A5: Function `prd_irrads` code

```
Function IGs = prd_irrads(taubn,taudn,NE,mas)
% This function returns the vector of per second irradiance for a given
% period in a day

ab = 1.454 - 0.406*taubn - 0.268*taudn + 0.021*taubn*taudn;
ad = 0.507 + 0.205*taudn - 0.08*taudn - 0.190*taubn*taudn;

IBs = NE*exp(-taubn*mas.^ab);
IDs = NE*exp(-taudn*mas.^ad);
IGs = IBs + IDs
```

A6: Function `solarstill_model` code

```
function [dT,M] = solarstill_model(xk,vk,uk)
%Function to evaluate the mathematical model of the single-slope solar still

global Ls Bs theta rhog rhob dwg Cf nf ti tg tb g ki hi va Cpg Cpb ...
       epsilong epsilonw sigmma rg alfg rw alfw alfb L

% ode states
Tgk = xk(1); %Glass temperature, Kelvin
Twk = xk(2); %Water temperature, Kelvin
Tbk = xk(3); %Basin temperature, Kelvin

%other transient inputs
Tak = vk(1); %ambient temperature, Kelvin
Ik = vk(2);% global irradiance)  , W/m2
dwk = uk(1);% depth of water, m
dyk = uk(2);% distillate yield, kg
%
%calculating temperature dependednt properties of moist air (Table 3)
%calculating temperature dependednt properties of moist air (Table 3)
Tvk = 0.5*(Twk + Tgk);%mean water-glass temp, K
Tvc = Tvk - 273.15;%mean water-glass temp, oC
muv = 1.718e-5 + 4.62e-8*Tvc;%dynamic viscosity
% muv = 1.7344e-5 + 4.62e-8*Tvc;%dynamic viscosity

Cpv = 999.2+0.1434*Tvc+1.101e-4*(Tvc^2)-6.7581e-8*(Tvc^3);%specific heat
kv = 0.0244 + 7.673e-5*Tvc;%thermal conductivity
betav = 1/Tvk;%volumetric expansion coefficient
rhov = 353.44/Tvk;%density, kg/m3

% obtaining values of Cv and nv (Table 2)
if dwg > 0 && dwg <= 0.2
    Cv = 0.21;
    nv = 0.25;
elseif dwg > 0.2 && dwg <= 0.25
     Cv = 0.075;
    nv = 1/3;
end

%calculating partial pressure of water vapour at Tgk and Twk
Pg = exp(25.317 - 5144/Tgk); %eqn(6)
```

```
Pw = exp(25.317 - 5144/Twk); %eqn(6)

Pwg = Pw - Pg;
DTP = abs((Twk - Tgk) + Pwg*Twk/(268.9e3 - Pw)); %eqn(4)

%calculating Grashof and Prandtl numbers (Table 1)
Grv = (DTP*g*betav)*(rhov^2)*(dwg^3)/(muv^2);%Grashof number
Prv = muv*Cpv/kv;%Prandtl number

hcwg = (Grv*Prv)^nv*(kv*Cv)/dwg;% eqn(3)
if abs(Twk-Tgk) == 0
    hewg = 0;
else
    hewg = 1.6273e-2*hcwg*Pwg/(Twk - Tgk); %eqn(5)
end

epsilonwg = (1/epsilong + 1/epsilonw - 1)^(-1); %effective emissivity
hrwg = epsilonwg*sigmma*(Twk^2 + Tgk^2)*(Twk + Tgk); %eqn(7)
hcga = 2.8 + 3*va; %eqn(8)
Tskyk = Tak-6;%eqn(10)
% Tskyk = Tak-12;%eqn(10)
hrga = epsilong*sigmma*(Tgk^2 + Tskyk^2)*(Tgk + Tskyk);% eqn(9)
% hrga = epsilong*sigmma*(Tgk^4 - Tskyk^4)/(Tgk - Tak);% eqn(9)
% hrga = epsilong*sigmma*(Tgk^4 - Tskyk^4)/(Tak - Tgk);% eqn(9)
%calculating temperature dependednt properties of water (Table 5)
Tfk = 0.5*(Tbk + Twk);%mean water-basin temp, K
Tfc = Tfk - 273.15;%mean water-glass temp, oC
muf = 1.7e-3 -4.6535e-5*Tfc + 5.7066e-7*(Tfc^2)-2.5438e-9*(Tfc^3); %dynamic
      %viscosity
Cpf = 4.2171e3 - 2.9971*Tfc + 0.0775*(Tfc^2)- 8.0286e-4*(Tfc^3)...
      + 3.2887e-6*(Tfc^4);%specific heat capacity, J/kg
kf = 0.56 + 0.0021*Tfc - 9.4839e-6*(Tfc^2);%thermal conductivity
rhof = 1.0006e3 - 0.071*Tfc - 0.0036*(Tfc^2);%density
betaf = 1/Tfk;%thermal expansivity
% betaf = -4.1454e-5 + 1.1852e-5*Tfc - 4.0623e-8*(Tfc^2);
%calculating Grashof & Prandtl numbers, and characteristic length (Table 4)
xf = 0.5*(Ls+Bs);%chracteristic length
Grf = abs(((Tbk - Twk)*g*betaf)*(rhof^2)*(xf^3)/(muf^2));%Grashof number
Prf = muf*Cpf/kf;%Prandtl number

hcwb = ((Grf*Prf)^nf)*(kf*Cf/xf);%eqn(14)

%basin - air heat transfer coeficients
hba = (ti/ki + 1/hi)^(-1);
%
Twc = Twk-273.15;
rhow = 1.0006e3 - 0.071*Twc - 0.0036*(Twc^2);%density of water
Cpw = 4.2171e3 - 2.9971*Twc + 0.0775*(Twc^2)- 8.0286e-4*(Twc^3)...
      + 3.2887e-6*(Twc^4);%specific heat capacity, J/kg
Ag = Ls*Bs*secd(theta); %eqn(2)
Aw = Ls*Bs; %eqn(2)
Ab = Aw; %eqn(2)
mg = Ag*rhog*tg; %eqn(2)
mw = Aw*rhow*dwk; %eqn(13)
mb = Ab*rhob*tb; %eqn(17)

%calculating irradiance components of glass, water and basin (Table 7)
cg = (1-rg)*alfg;
cw = (1-alfg)*(1-rg)*(1-rw)*alfw;
cb = (1-alfg)*(1-alfw)*(1-rg)*(1-rw)*alfb;
%
qewg = hewg*(Twk - Tgk);
%
hwg = hcwg + hrwg + hewg;
hga = hcga + hrga;
```

```
Iag = cg*Ik;
Iaw = cw*Ik;
Iab = cb*Ik;

%Heat balance derivative equations
dT1 = (1/(mg*Cpg))*(Ag*Iag + Aw*hwg*(Twk-Tgk)-Ag*hga*(Tgk-Tak));%eqn(
dT2 = (1/(mw*Cpw))*(Aw*Iaw + Aw*hcwb*(Tbk-Twk)-Aw*hwg*(Twk-Tgk));%eqn(
dT3 = (1/(mb*Cpb))*(Ab*Iab - Aw*hcwb*(Tbk-Twk)-Ab*hba*(Tbk-Tak));%eqn(
dT = [dT1,dT2,dT3];

%instantaneous yield, depth and efficiency of still, %
idy = qewg*Aw/L; %yield, Kg/s
dy = dyk + idy; %yield, Kg/s
dw = dwk - idy/(rhow*Aw);
eff = (qewg/Ik)*100; %efficiency, %
M = [dw,dy,idy,qewg,eff];
```

A7: Function `solarstill_model_solver` code

```
function [t,Tv,Mv] = solarstill_model_solver(f,tsts,vv,x0,u0)
% This function obtains the numerical solution of solar still ODEs using
% Runge Kutta method

xk = x0;
vk = vv(1,:);
uk = [u0,0];

t(1,1) = tsts(1);
Tv(1,:) = xk;
Mv(1,:) = [uk,0,0,0];

n = length(tsts);
h = tsts(2)-tsts(1);
for k = 2:n
    if uk(1,1) < 2.5e-6;
        break
    else

    [f11,f12] = feval(f,xk,vk,uk);
    f11 = h*f11;
    [f21,~] = feval(f,xk+f11/2,vk,uk);
     f21 = h*f21;
    [f31,~] = feval(f,xk+f21/2,vk,uk);
     f31 = h*f31;
    [f41,~] = feval(f,xk+f31,vk,uk);
      f41 = h*f41;
    ff = xk + (f11+2*(f21+f31)+ f41)/6;

    xk = ff;
    vk = vv(k,:);
    uk = f12(1:2);

 %updating...
    t(k,1) = tsts(k);
    Tv(k,:) = xk;
    Mv(k,:) = f12;
    end
end
```

A8: `solarstill_solve` script

```
clear
%Fixed parameters
global Ls Bs theta rhog rhob dwg Cf nf ti tg tb g ki hi va Cpg Cpb ...
       epsilong epsilonw sigmma rg alfg rw alfw alfb L
%%
```

```
%Fixed parameter values
Ls = 1; %length of still, m
Bs = 1; % breadth of still, m
theta = 11;% slope of glass cover degrees
Cf = 0.54;% Basin-water convective heat transfer constant (lamina flow
nf = 0.25;% Basin-water convective heat transfer constant (lamina flow
ti = 0.01;% insulator thickness, m
tg = 0.003;% glass thickness, m
tb = 0.005; %absorber thickness, m
g = 9.81;%acceleration due to gravity, m/s2
ki = 0.042;%Thermal conductivity of insulation W/m°K
hi = 5.7; % combined heat transfer coefficient from insulator W/m2°K
L = 2256700; %Latent heat of vapourisation of water J/Kg
epsilong = 0.9; %Emmissivity of glass;
epsilonw = 0.95; %Emmissivity of water;
sigmma = 5.67e-8;%Stephan-Boltzmann constant
Cpg = 750; % specific heat capacity of glass;
Cpb = 385; % specific heat capacity of basin liner (absorber)
rhog = 2800; % density of glass, kg/m3
rhob = 8933; % density of basin liner, kg/m3
rg = 0.05;% reflectivity of glass
alfg = 0.1;% absobvity of glass
rw = 0.05;% reflectivity of water
alfw = 0.29;% absobvity of water
alfb = 0.97;% absobvity of basin
va = 3; %velocity of air
dwg = 0.25;%water-glass spacing, m
%%
load smdat
load shdat
smdats = smdat;
fhh = shdat;
Isc = 1367;% solar constant
ndate = '15-APR-2018';%
stime = '07:00:00';%
etime = '18:00:00';%
phi = 12;%
psis = 15;%
psia = 8.5920;
Tmin = 27.3;
Tmax = 39.9;
%%
[delta,eqts,EN,taubn,taudn] = day_solarparams(Isc,ndate,smdats);%no 1
[tsrss,masrss] = loc_solarparams(delta,phi);
[tsts,mas] = ...
          prd_solarparams(stime,etime,eqts,psis,psia,tsrss,masrss);
IG = prd_irrads(taubn,taudn,EN,mas);%no 4
[Tasc,Task] = prd_tambs(fhh,tsts,Tmin,Tmax);%no 5

f = 'solarstill_model';

u0 = 0.01;
% tv = tsts;
vv = [Task,IG];
x0 = [22,23,24] + 273.15;

[ts,T,M] = solarstill_model_solver(f,tsts,vv,x0,u0);
ts = ts/3600;
T = T-273.15;

figure(1)
plot(ts,T(:,1))
xlabel('Solar time, hr')
ylabel('Water temperature, ^oC')
grid
```