Supplementary Information (SI) for Energy & Environmental Science.
This journal is © The Royal Society of Chemistry 2025

Supplemental information for

Designing next-generation all-weather and efficient atmospheric water harvesting powered by solar energy

Pengfei Wang,‡[a] Jiaxing Xu,‡[ab] Zhaoyuan Bai,‡[a] Ruzhu Wang [ab] and Tingxian Li *[ab]

[a] Institute of Refrigeration and Cryogenics, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: Litx@sjtu.edu.cn

[b] Research Center of Solar Power and Refrigeration of Ministry of Education, Shanghai Jiao Tong University, Shanghai, 200240, China.

‡ These authors contributed equally to this work.

Supplemental Information Table of Contents

Supplemental Note

Supplemental Figure

Supplemental Note

Note S1 Assessment of energy consumption for DAWH

We calculate the theoretical ideal energy consumption of the SAWH, DAWH, and HAWH systems by assessing the air treatment process as a unity criterion based on our previously reported work[1]. An index termed exergy, defined as the available energy of the input energy, is used to express the ideal energy consumption of AWH technologies. For a given AWH system with a heat input temperature higher than the ambient temperature, the exergy (Ex_{heat}) can be calculated by equation (S1), and for a given AWH system with a cold input temperature lower than the ambient temperature, the exergy (Ex_{cold}) can be described by equation (S2):

$$Ex_{heat}=\frac{Q\,(T_{in}-T_{amb})}{T_{in}} \tag{S1}$$

$$Ex_{cold}=\frac{Q\,(T_{amb}-T_{in})}{T_{in}} \tag{S2}$$

where Q, T_{in}, and T_{amb} are the heat flux, heat input temperature, and ambient temperature, respectively.

The water production per unit of exergy (the specific water production, *SWP*) is defined and can be described by equation (S3):

$$SWP=\frac{SWHR\cdot R}{Ex} \tag{S3}$$

where *SWHR* and *R* are the specific water harvesting rate and air flow rate, respectively. The *SWHR* changes with different air temperatures and RH conditions, indicating the liquefied water per kilogram of input air. An optimization problem exists to determine the most exergy-efficient operation parameters under a specific ambient condition, where the parameters for calculation are presented in Table S2.

For a given DAWH system, only a low-temperature cold source is needed to cool down the ambient air to its dew point temperature and then liquify water vapor. Firstly, we identify the relationship of each parameter of the ambient air, including temperature (*T*), humidity (*d*), pressure (*P*), and relative humidity (RH) by equations (S4)- (S6).

According to the Hyland-Wexler empirical equation, the saturated water vapor pressure ($P_{sat,amb}$, unit: Pa) can be calculated by Equation (S4)[2]:

$$P_{sat,amb}=e^{\left(\frac{c_1}{T_{amb}}+c_2-c_3\cdot T_{amb}+c_4\cdot T_{amb}^2+c_5\cdot T_{amb}^3+c_6\cdot T_{amb}^4+c_7\cdot\ln(T_{amb})\right.} \tag{S4}$$

where T_{amb} is the temperature of ambient air (unit: K). When 173.15 K < T_{amb} < 273.15 K, c_1=-5674.5359, c_2=6.3925247, c_3=-9.677843*10^{-3}, c_4=6.2215701*10^{-7}, c_5=2.0747825*10^{-19}, c_6=-9.484024*10^{-13}, c_7=4.1635019; and when 273.15 K ≤ T_{amb} < 473.15 K, c_1=-5800.2206, c_2=1.3914993, c_3=-4.860239*10^{-2}, c_4=4.1764768*10^{-5}, c_5=-1.4452093*10^{-8}, c_6=0, c_7=6.5459673.

For a given air temperature T_{amb}, we can calculate the threshold water vapor ($d_{sat,amb}$, unit: g kg_{air}^{-1}) contained in the air by equation (S5):

$$d_{\mathrm{sat,amb}} = 622 \cdot \frac{P_{\mathrm{sat,amb}}}{101325 - P_{\mathrm{sat,amb}}} \tag{S5}$$

The water vapor contained in the air (with certain temperature and relative humidity) can be calculated by equation (S6):

$$d_{\mathrm{amb}} = 622 \cdot \frac{P_{\mathrm{sat,amb}} \cdot RH_{\mathrm{amb}}}{101325 - P_{\mathrm{sat,amb}} \cdot RH_{\mathrm{amb}}} \tag{S6}$$

We further identify the basic parameters of low-temperature condenser ($T_{Cond,D}$), which is used to cool the ambient air from T_{amb} to the temperature $T_{out,D}$. The heat transfer irreversibility induces a margin between $T_{Cond,D}$ and $T_{out,D}$, which can be calculated by equation (S7).

$$\Delta T_{\mathrm{D}} = T_{\mathrm{out,D}} - T_{\mathrm{Cond,D}} = 4 + 1 \cdot \frac{SWHR_{\mathrm{D}}}{5}\ (unit: ^{\circ}\mathrm{C}) \tag{S7}$$

where $SWHR_D$ represents the specific water harvesting rate of DAWH system, indicating the liquefied water per kilogram of input air, which can be calculated by the humidity difference between the inlet air and the outlet air of the condenser (equation (S8)):

$$SWHR_{\mathrm{D}} = d_{\mathrm{amb,D}} - d_{\mathrm{out,D}}\ (unit: \mathrm{g\ kg_{air}^{-1}}) \tag{S8}$$

This DAWH system can work only when the temperature $T_{out,Cond}$ meets the following equation (S9):

$$d_{\mathrm{sat,out,Cond}} = f(T_{\mathrm{out,Cond}}) \leq d_{\mathrm{amb}} \tag{S9}$$

We assume both the sensible and latent heat of the input air are discharged by the condenser, the overall heat flux can be calculated by equations (S10) and (S11):

$$Q_{\mathrm{Cond,D}} = R_{\mathrm{D}} \cdot (i_{\mathrm{amb,D}} - i_{\mathrm{out,D}}) \tag{S10}$$

$$i = c_{p,\mathrm{air}} T + d\,(c_{p,\mathrm{vapor}} T + h_{\mathrm{fg}}) \tag{S11}$$

where i represents the specific enthalpy of air (kJ kg^{-1}), R_D represents the air flow rate in the DAWH system. $c_{p,air}$ and $c_{p,vapor}$ represent the specific heat of air and water vapor, equaling 1.01 kJ kg^{-1} $K^{-1}$and 1.84 kJ kg^{-1} K^{-1}, respectively. h_{fg} represents the latent heat of water vaporization, equaling to 2500 kJ kg^{-1}.

Finally, the exergy (Ex_D) and SWP_D can be calculated by equations (S12) and (S13).

$$Ex_D=\frac{Q_{Cond,D}\,(T_{amb}-T_{Cond,D})}{T_{Cond,D}} \tag{S12}$$

$$SWP_D=\frac{SWHR_D\cdot R_D}{Ex_D} \tag{S13}$$

Hence, the optimal $T^*_{Cond,D}$, $SWHR^*$, and SWP^*_D can be obtained by calculating the solutions of the optimization problem in equation (S14), and we calculate this process utilizing C++ software (Code S1)

$$\begin{cases} \{T^*_{Cond,D}\}=\arg\max\ \{SWP_D,\ \text{s.t. } SWHR_D(T_{Cond,D})\geq 0\} \\ T_{Cond,D}\in(-10\ ℃,\ \ T_{amb}) \\ SWHR_D{}^*=SWHR_D(T^*_{Cond,D}),\ SWP^*_D=SWP_D(T^*_{Cond,D}) \end{cases} \tag{S14}$$

where -10 ℃ is defined as the lowest limit of condensation temperature in DAWH to prevent the risk of frost.

Note S2 Assessment of energy consumption for SAWH

For a given SAWH system, the energy input can be divided into three parts, including the heat sink for sorbent sorption ($T_{\mathrm{Ad,S}}$), the heat source for sorbent desorption ($T_{\mathrm{De,S}}$), and the heat sink for condensation($T_{\mathrm{Cond,S}}$). During the water sorption process, cold energy enhances the water sorption of sorbents, which will further influence the desorption condition. During the water desorption and condensation processes, heat energy is needed to drive sorbents to release the absorbed water, while cold energy is needed to condense the released water vapor.

Considering the heat transfer irreversibility, the actual temperature of water sorbents ($T_{\mathrm{Ad,w}}$, $T_{\mathrm{De,w}}$) can be calculated by considering a temperature margin (equations (S15) and (S16)). Besides, the outlet air temperature of the condenser ($T_{\mathrm{Cond,out}}$), equaling the inlet air temperature of the desorption bed ($T_{\mathrm{De,in}}$) for a closed desorption cycle, also can be calculated (equation (S17))[1]:

$$\Delta T_{\mathrm{Ad}} = T_{\mathrm{Ad,w}} - T_{\mathrm{Ad,S}} = 4 + 1 \cdot \frac{SWHR_{\mathrm{S}}}{5}\ (unit: ^{\circ}\mathrm{C}) \tag{S15}$$

$$\Delta T_{\mathrm{De}} = T_{\mathrm{De,S}} - T_{\mathrm{De,w}} = 6 + 1 \cdot \frac{SWHR_{S}}{5}\ (unit: ^{\circ}\mathrm{C}) \tag{S16}$$

$$\Delta T_{\mathrm{Cond}} = T_{\mathrm{Cond,out}} - T_{\mathrm{Cond,S}} = T_{\mathrm{De,in}} - T_{\mathrm{Cond,S}} = 4 + 1 \cdot \frac{SWHR_{S}}{5}\ (unit: ^{\circ}\mathrm{C}) \tag{S17}$$

During the sorption process, the initiated humidity around the water sorbent is $d_{\mathrm{Ad1,w}}$, where the water content in the sorbent is W_{dry}. Thus, the humidity difference drives the moisture capture and is finally halted when $d_{\mathrm{Ad2,w}}$ is close to the humidity of inlet air $d_{\mathrm{amb,S}}$ with only a margin value of Δd_{Ad} (equation (S18)). The final water content in the sorbent is W_{sat}, and the difference between W_{dry} and W_{sat} is the cyclic water content $\Delta W_{\mathrm{cycle}}$ (assumed as 0.2 g g^{-1}, equation (S19)).

$$\Delta d_{\mathrm{Ad}} = d_{\mathrm{amb,S}} - d_{\mathrm{Ad2,w}} = 0.2\ SWHR_{\mathrm{S}}\ (unit: \mathrm{g\ kg_{air}^{-1}}) \tag{S18}$$

$$\Delta W_{\mathrm{cycle}} = W_{\mathrm{sat}} - W_{\mathrm{dry}} = 0.2\ \ (unit: g\ g_{\mathrm{sorbent}}^{-1}) \tag{S19}$$

Once the sorbent is heated, the absorbed water cannot be released immediately. A new equilibrium point with a humidity $d_{\mathrm{De1,w}}$ ($d_{\mathrm{De1,w}} < d_{\mathrm{Ad2,w}}$) is established under the desorption temperature $T_{\mathrm{De,w}}$. The humidity difference between $d_{\mathrm{De,in}}$ (the humidity of inlet desorption air) and $d_{\mathrm{De1,w}}$ drives the water release until the water sorbent is dried off to W_{dry} with a humidity $d_{\mathrm{De2,w}}$. The averaged water content in the water sorbent during the desorption process can be estimated by equation (S20) with a margin Δd_{De} between $\bar{d}_{\mathrm{De,out}}$ and $\bar{d}_{\mathrm{De,w}}$ defined by equation (S21). The desorption process proceeds when equation (S22) is satisfied and the humidity difference between the inlet and outlet air is the $SWHR_{\mathrm{S}}$ (equation (S23)).

$$\bar{d}_{\mathrm{De,w}} = \frac{\int_{W_{\mathrm{dry}}}^{W_{\mathrm{sat}}} d_{\mathrm{De}}(W)dW}{W_{\mathrm{sat}} - W_{\mathrm{dry}}} \ (unit: g\ kg_{air}^{-1}) \tag{S20}$$

$$\Delta d_{De} = \bar{d}_{\mathrm{De,w}} - \bar{d}_{\mathrm{De,out}} = 0.4\ SWHR_{\ S}\ (unit: g\ kg_{air}^{-1}) \tag{S21}$$

$$d_{\mathrm{De1,w}} > d_{De,in}, d_{De,in} = f(T_{De,in}) \tag{S22}$$

$$SWHR_{\ S} = \bar{d}_{De,w} - d_{De,in}\ (unit: g\ kg_{air}^{-1}) \tag{S23}$$

The heat flux of water sorption process, desorption process and condensation process can be calculated by the following equations (S24)-(S29):

$$Q_{\mathrm{Ad,S}}=R_{\mathrm{Ad,S}} \cdot \left(i^*_{\mathrm{Ad,out}} - i^*_{\mathrm{Ad,in}}\right) + m_{\mathrm{w}} c_{\mathrm{p,w}} \left(T_{\mathrm{De,w}} - T_{\mathrm{Ad,w}}\right) + m_{\mathrm{m}} c_{\mathrm{p,m}} (T_{\mathrm{De,m}} - T_{\mathrm{Ad,m}}) \tag{S24}$$

$$Q_{\mathrm{De,S}}=R_{\mathrm{De,S}} \cdot \left(i^*_{\mathrm{De,out}} - i^*_{\mathrm{De,in}}\right) + m_{\mathrm{w}} c_{\mathrm{p,w}} \left(T_{\mathrm{De,w}} - T_{\mathrm{Ad,w}}\right) + m_{\mathrm{m}} c_{\mathrm{p,m}} (T_{\mathrm{De,m}} - T_{\mathrm{Ad,m}}) \tag{S25}$$

$$Q_{\mathrm{Cond,S}}=R_{\mathrm{Cond,S}} \cdot \left(i_{\mathrm{Cond,out}} - i_{\mathrm{Cond,in}}\right) \tag{S26}$$

$$m_{\mathrm{w}}=\frac{R_{\mathrm{De,S}} \cdot SWHR_{\mathrm{S}}}{\Delta W_{\mathrm{cycle}}}, m_{\mathrm{m}}=r_{\mathrm{m}} \cdot m_{\mathrm{w}} \tag{S27}$$

$$i^* = c_{p,air} T_{air} + d_{air} \left(c_{p,vapor} T_{air} + h_{\mathrm{w}}\right) \tag{S28}$$

$$i = c_{p,air} T_{air} + d_{air} \left(c_{p,vapor} T_{air} + h_{fg}\right) \tag{S29}$$

where h_{w} is the water sorption/desorption enthalpy of sorbents, assumed it is equal to 2700 kJ kg^{-1}; m_{w} and m_{m} represent the mass of water sorbent and matrix of sorbent, respectively.

To reach the dynamic balance of water sorption and desorption process of two sorbent beds, the air flow rate in the sorption side is larger because low sorption kinetics is always a main obstacle (equation (S30)):

$$\frac{R_{\mathrm{Ad,S}}}{R_{\mathrm{De,S}}} = \max\left(2, \left|\frac{\bar{d}_{\mathrm{De,w}} - d_{De,in}}{\bar{d}_{\mathrm{Ad,w}} - d_{\mathrm{Ad},in}}\right|\right) \tag{S30}$$

The conditions of outlet air can be calculated by equations (S31)-(S33):

$$T_{\mathrm{De,out}} = \max\left(\eta_{\mathrm{De,T}} \left(T_{\mathrm{De,w}} - T_{\mathrm{De,in}}\right) + T_{\mathrm{De,in}}, T_{\mathrm{De,in}}\right), \eta_{\mathrm{De,T}} = 0.8 \tag{S31}$$

$$T_{\mathrm{Ad,out}} = \min\left(T_{\mathrm{Ad,in}} - \eta_{\mathrm{Ad,T}} \left(T_{\mathrm{Ad,in}} - T_{\mathrm{Ad,w}}\right), T_{\mathrm{Ad,in}}\right), \eta_{\mathrm{Ad,T}} = \frac{0.8}{\left(R_{\mathrm{Ad,S}}/R_{\mathrm{De,S}}\right)^{0.2}} \tag{S32}$$

$$d_{\mathrm{Ad,out}} = d_{\mathrm{Ad,in}} + \frac{SWHR_{\mathrm{S}} \cdot R_{\mathrm{De,S}}}{R_{\mathrm{Ad,S}}} \tag{S33}$$

Finally, exergy (*Ex*S) and *SWP*S can be calculated by equations (S34) and (S35):

$$Ex_{\mathrm{S}}=\frac{Q_{\mathrm{Ad,S}}\,(T_{\mathrm{amb}}-T_{\mathrm{Ad,S}})}{T_{\mathrm{Ad,S}}}+\frac{Q_{\mathrm{De,S}}\,(T_{\mathrm{De,S}}-T_{\mathrm{amb}})}{T_{\mathrm{De,S}}}+\frac{Q_{\mathrm{Cond,S}}\,(T_{\mathrm{amb}}-T_{\mathrm{Cond,S}})}{T_{\mathrm{Cond,S}}} \tag{S34}$$

$$SWP_{\mathrm{S}}=\frac{SWHR\cdot R_{\mathrm{air}}}{Ex_{\mathrm{S}}} \tag{S35}$$

where the subscripts Ad,S, De,S, and Cond,S represent the water sorption, desorption, and condensation processes in the SAWH system, respectively.

The optimal $T^{*}_{\mathrm{Ad,S}}$, $T^{*}_{\mathrm{De,S}}$, $T^{*}_{\mathrm{Cond,S}}$, $SWHR^{*}$, and SWP^{*}_{S} can be obtained by calculating the solutions of the optimization problem in equation (S36), and we calculate this process utilizing C++ software (Code S2)

$$\begin{cases}\left\{T^{*}_{\mathrm{Ad,S}},\,T^{*}_{\mathrm{De,S}},\,T^{*}_{\mathrm{Cond,S}}\right\}=\arg\max\ \left\{SWP_{\mathrm{S}},\ \text{s.t. } SWHR\left(T_{\mathrm{Ad,S}},\,T_{\mathrm{De,S}},\,T_{\mathrm{Cond,S}}\right)\geq 0\right\}\\ T_{\mathrm{Ad,S}},\,T_{\mathrm{Cond,S}}\in(\text{-5 °C},\,T_{\mathrm{amb}}),\,T_{\mathrm{De,S}}\in(T_{\mathrm{amb}},\,150\ \text{°C})\\ SWHR^{*}=SWHR\left(T^{*}_{\mathrm{Ad,S}},\,T^{*}_{\mathrm{De,S}},\,T^{*}_{\mathrm{Cond,S}}\right),\ SWP^{*}_{\mathrm{S}}=SWP_{\mathrm{S}}(T^{*}_{\mathrm{Ad,S}},\,T^{*}_{\mathrm{De,S}},\,T^{*}_{\mathrm{Cond,S}})\end{cases} \tag{S36}$$

where -5 °C is defined as the lowest water sorption and condensation temperature to prevent freezing and 150 °C is defined as the highest desorption temperature to avoid overheating.

Note S3 Strengthening effect of cold energy on water sorption and condensation

The cold energy supplied from the heat pump can strengthen both the water sorption and water condensation processes. We first introduced the influence of cold energy on water sorption of sorbents. Typically, the isothermal water sorption curve of the sorbent is mainly determined by the relative humidity ($P\,P_0^{-1}$), where P is the partial vapor pressure of ambient air and P_0 is the saturated vapor pressure at the temperature. Experimental tests show that the isothermal water sorption curves under different temperatures do not differ much. Here, we defined a critical parameter, sorption relative humidity (RH_{sorp}), to indicate the equilibrium capacity of water uptake at the sorption stage, expressed by the following equation:

$$RH_{sorp}=\frac{P_{sorp}}{P_{sat}(T_{sorp})} \tag{S37}$$

where P_{sorp} and $P_{sat}(T_{sorp})$ represent the water vapor pressure of the inlet air for sorption and the saturated water vapor pressure of the inlet air at T_{sorp}.

The equilibrium water uptake of the sorbent is $m_{sorp}(RH_{sorp})$ (unit: g $g_{sorbent}^{-1}$), which can be obtained from the isothermal water sorption curve. The P_{sorp} (unit: Pa) is a fixed value for these open SAWH system, always equal to the $RH_{amb}\cdot P_{sat}(T_{amb})$. The saturated water vapor pressure $P_{sat}(T_{sorp})$ can be calculated by the Hyland-Wexler empirical equation:

$$P_{sat}(T_{sorp})=e^{(\frac{c_1}{T_{sorp}}+c_2+c_3\cdot T_{sorp}+c_4\cdot T_{sorp}^2+c_5\cdot T_{sorp}^3+c_6\cdot T_{sorp}^4+c_7\cdot \ln(T_{sorp})} \tag{S38}$$

where the T_{sorp} is the temperature of the inlet air for water sorption (unit: K).

Hence, once the temperature of the inlet air is cooled by the cold energy from the heat pump, the cold energy can effectively carry away the sorption heat and cool the inlet air around the sorbent. As a result, the $P_{sat}(T_{sorp})$ is reduced, and the P_{sorp} remains constant. The RH_{sorp} is increased, inducing the increase of equilibrium water uptake of sorbents $m_{sorp}(RH_{sorp})$.

In particular, for those meso-/micro-porous materials with steep step-like S shape isothermal water sorption curves (the water uptake increases sharply at a certain and narrow range) and hygroscopic-salt-based composites with a step-like process at the deliquescence *RH*, cold energy can significantly broaden the environmental adaptability. Here we use a parameter, critical *RH* (RH_c), corresponding to the *RH* at the beginning of S shape curves for those meso-/micro-porous materials and the deliquescence RH for those hygroscopic composites. When the *RH* under ambient environment is lower than RH_c, the sorbent in the conventional SAWH system has almost no water sorption, while the sorbent with cold-assisted sorption can increase the RH_{sorp} to meet efficient water sorption.[3] We illustrated the above process in Fig. S3, using two

isothermal water sorption curves of two typical sorbents (MIL-101(Cr)[4] and LiCl@rGO-SA[5], which represent the two types of sorbents we discussed above. Hence, cold energy can both enhance the water sorption capacity and broaden the environmental adaptability during the water sorption process.

As for the influence of cold energy on the water condensation process, we defined another critical parameter, desorption relative humidity (RH_{de}), to indicate the water productivity at the desorption stage, expressed by the following equation:[6]

$$RH_{de}=\frac{P_{de}}{P_{sat}(T_{de})} \tag{S39}$$

where P_{de} and P_{sat} (T_{de}) represent the water vapor pressure of the inlet air for desorption and the saturated water vapor pressure of the inlet air at T_{de}.

For a saturated sorbent with an equilibrium water sorption capacity of $m_{sorp}(RH_{sorp})$, the total amount of desorbed water (Δm_{de}) can be calculated by the equilibrium water sorption capacity ($m_{sorp}(RH_{sorp})$) minus the residual water inside the sorbent ($m_{de}(RH_{de})$) by equation S40:

$$\Delta m_{de}=m_{sorp}\left(RH_{sorp}\right)-m_{de}(RH_{de}) \tag{S40}$$

where m_{sorp} and m_{de} can be obtained from the isothermal water sorption and desorption curves according to RH_{sorp} and RH_{de}, respectively.

To emphasize the effect of condensation strengthening, we keep the equilibrium water sorption capacity of the sorbent at a fixed value; thus, the residual water inside the sorbent ($m_{de}(RH_{de})$) influences the total amount of desorbed water. If we ignore the condensation resistance and mass transfer resistance, the water vapor pressure of the condensed air is equal to the water vapor pressure of the inlet air, expressed by the following equation S41:

$$P_{de}=P_{sat}(T_{cond})=e^{(\frac{c_1}{T_{cond}}+c_2-c_3\cdot T_{cond}+c_4\cdot T_{cond}^2+c_5\cdot T_{cond}^3+c_6\cdot T_{cond}^4+c_7\cdot\ln(T_{cond})} \tag{S41}$$

where the T_{cond} is the condensation temperature.

Compared to the T_{cond} of conventional SAWH that relies on the ambient environment for condensation, the T_{cond} of cooling-assisted condensation is lower. As a result, the P_{de} of SAWH system with cooling-assisted condensation is smaller than that of conventional SAWH. Hence, if we keep the desorption temperature constant ($P_{sat}(T_{de})$ is constant), the residual water inside the sorbent of SAWH system with cooling-assisted condensation is less, so that it can desorb more water. If we keep the total amount of desorbed water constant (RH_{de} is constant), the desorption temperature T_{de} of SAWH system with cooling-assisted condensation can

be lower; thus, this can decrease the heat grade for desorption and weaken the unnecessary heat dissipation to the environment.

Overall, the cold energy of the heat pump can synergistically reduce the sorption temperature and the condensation temperature by cascade utilization. Compared to the sorption heat release in the sorption bed, more heat is released at the condenser, including the sensible heat of the desorbed air and the latent heat of the water condensation, resulting in a higher temperature lift in the condenser. Therefore, it is a more appropriate sequence for the cold air to flow through the sorption bed first and then into the condenser. The sorption heat does not significantly raise the temperature of cold energy and thus has little impact on the subsequent condensation. Conversely, once the cold air first flows through the condenser, the temperature of the outlet air from the condenser may rise too high, affecting the subsequent enhancement of the sorption process.

Note S4 Thermodynamic analysis of HAWH

For the proposed HAWH system, the maximum *SWP* between the DAWH system and the SAWH system is selected under each specific ambient condition due to its self-adaptative dual-mode operation (S42):

$$SWP_{\mathrm{H}}^{*}= \max\left\{SWP_{\mathrm{D}}^{*}, SWP_{\mathrm{S}}^{*}\right\} \tag{S42}$$

By iteratively optimizing the optimal input temperature ($T_{\mathrm{Cond,D}}$, $T_{\mathrm{Ad,S}}$, $T_{\mathrm{De,S}}$, $T_{\mathrm{Cond,S}}$) and the optimal specific water harvesting rate, the optimal *SWP* of different AWH systems under different working conditions can be determined. By comparing the SWP_{D}^{*} and SWP_{S}^{*}, we can obtain the suitable climatic parameters for HAWH under each specific ambient condition. It is worth noting that this calculation only considers the necessary energy loss due to irreversible temperature transfer to theoretically analyze the ideal energy efficiency under each ambient condition. Exergy consumption, which is the ideal energy consumption, is difficult to achieve in state-of-the-art devices.

We further assessed the year-round operation time and average annual *SWP* in various global climatic regions of the HAWH system according to the weather data of 2022 (available at https://ldas.gsfc.nasa.gov/FLDAS/). Considering the operation time required for water collection and the weather data available, we assume that the HAWH system intelligently switches every three hours (the minimum interval of available weather data) according to the temperature and RH conditions (Code S3). Thus, the year-round operation times of the proposed HAWH system and conventional single-mode DAWH system can be obtained according to equations (S43) and (S44), and the difference between the two AWH systems is the increase in operation time per year for water harvesting from air. We selected several typical cities with arid, semi-arid, semi-humid, and humid conditions worldwide to make an intuitive comparison, demonstrating the enhanced climatic adaptability of the next-generation HAWH.

$$Time_{\mathrm{H}}= \sum_{\mathrm{i}=1}^{2920} \begin{cases} 0.125, & \text{s.t. } SWP_{\mathrm{H}}^{*}(T_{\mathrm{i}}, RH_{\mathrm{i}})>0 \\ 0, & \text{s.t. } SWP_{\mathrm{H}}^{*}(T_{\mathrm{i}}, RH_{\mathrm{i}})=0 \end{cases} \tag{S43}$$

$$Time_{\mathrm{D}}= \sum_{\mathrm{i}=1}^{2920} \begin{cases} 0.125, & \text{s.t. } SWP_{\mathrm{D}}^{*}(T_{\mathrm{i}}, RH_{\mathrm{i}})>0 \\ 0, & \text{s.t. } SWP_{\mathrm{D}}^{*}(T_{\mathrm{i}}, RH_{\mathrm{i}})=0 \end{cases} \tag{S44}$$

where i represents the number of HAWH alternations per year in each region on Earth, and the unit of operation time is days.

The water production per kilowatt-hour of exergy consumption (*SWP*, kg kWh^{-1}) is defined as the compared index to indicate the energy-saving potential of HAWH. For each region on Earth with different

temperatures and RH conditions, the average annual *SWP* of the HAWH, DAWH, and SAWH systems can be calculated by equations (S45)-(S47):

$$SWP_{\mathrm{H}}= \frac{1}{2920}\cdot \sum_{\mathrm{i=1}}^{2920} SWP_{\mathrm{H}}^{*} \tag{S45}$$

$$SWP_{\mathrm{D}}= \frac{1}{2920}\cdot \sum_{\mathrm{i=1}}^{2920} SWP_{\mathrm{D}}^{*} \tag{S46}$$

$$SWP_{\mathrm{S}}= \frac{1}{2920}\cdot \sum_{\mathrm{i=1}}^{2920} SWP_{\mathrm{S}}^{*} \tag{S47}$$

The increase in the average annual *SWP* resulting from the replacement of conventional single-mode DAWH or SAWH systems with HAWH can be calculated via equations (S48) and (S49):

$$\Delta SWP_{\mathrm{H-D}}= \frac{1}{2920}\cdot \sum_{\mathrm{i=1}}^{2920} (SWP_{\mathrm{H}}^{*} - SWP_{\mathrm{D}}^{*}) \tag{S48}$$

$$\Delta SWP_{\mathrm{H-S}}= \frac{1}{2920}\cdot \sum_{\mathrm{i=1}}^{2920} (SWP_{\mathrm{H}}^{*} - SWP_{\mathrm{S}}^{*}) \tag{S49}$$

Note S5 Energy efficiency comparison of solar-electric-driven HAWH and solar-thermal-driven SAWH

We further evaluate the influence of low solar-electric conversion efficiency on the energy efficiency of HAWH. Taking the solar energy conversion process into account, we compare the energy efficiency of the HAWH system driven by photovoltaic panels and the conventional SAWH system driven by direct solar-thermal conversion. We still use the unified index, specific water production (*SWP*, defined as water production per energy consumption, kg kWh^{-1}), for this comparison.

Specifically, three main differences exist between the HAWH system and the solar-thermal-driven SAWH system:

1) The first difference lies in the solar conversion efficiency, where we set photovoltaic conversion efficiency as 25% and photothermal conversion efficiency as 75% empirically during calculation.
2) Secondly, the HAWH system can generate cold energy to facilitate water sorption and condensation, while the conventional solar-thermal-driven SAWH system can only harvest heat energy for water desorption but perform water sorption and condensation under ambient temperature.
3) Finally, considering electricity is a high-grade energy source and heat pumps can harvest energy from ambient air, we use exergy to evaluate the minimum energy requirements (the energy input to compensate for the energy consumption) for HAWH. As for solar-thermal-driven SAWH, the energy input is equal to the energy consumption, and the desorption temperature has an upper limit (defined as 100°C empirically).

We considered the above three differences between these two systems while keeping other irreversible heat and humidity transfer losses consistent. Hence, the *SWP* of the solar-electric-driven HAWH system can be obtained by multiplying the previously calculated SWP_{H}^{*} data by photovoltaic conversion efficiency (25%), shown in equation (S50):

$$SWP_{Solar-H} = 25\% \cdot SWP_{\mathrm{H}}^{*} = 25\% \cdot \max\left\{SWP_{\mathrm{D}}^{*}, SWP_{\mathrm{S}}^{*}\right\} \quad \text{(S50)}$$

As for the conventional solar-thermal-driven SAWH system, the energy input only serves as the heat source for sorbent desorption ($T_{\mathrm{De,S}}$). During the water sorption process, we assumed the sorbents were located in the ambient environment, with the ambient air as the input airflow. Hence, the actual temperatures of both water sorbents ($T_{\mathrm{Ad,w}}$) and input air ($T_{\mathrm{Ad,in}}$) are the ambient temperature (T_{Amb}). During the water desorption and condensation processes, heat energy is needed to drive sorbents to release the absorbed water, and ambient air is applied to condense the released water vapor ($T_{\mathrm{Cond,S}}$). To keep the heat transfer irreversibility consistent with the HAWH system, the actual temperature of water sorbents ($T_{\mathrm{De,w}}$) can be calculated by

considering a temperature margin (equations (S51)). Besides, the outlet air temperature of the condenser ($T_{\mathrm{Cond,out}}$), equaling the inlet air temperature of the desorption bed ($T_{\mathrm{De,in}}$) for a closed desorption cycle, also can be calculated (equation (S52))[1]:

$$\Delta T_{\mathrm{De}} = T_{\mathrm{De,S}} - T_{\mathrm{De,w}} = 6 + 1 \cdot \frac{SWHR_S}{5}\ (unit: ^{\circ}\mathrm{C}) \tag{S51}$$

$$\Delta T_{\mathrm{Cond}} = T_{\mathrm{Cond,out}} - T_{\mathrm{Cond,S}} = T_{\mathrm{De,in}} - T_{\mathrm{Cond,S}} = 4 + 1 \cdot \frac{SWHR_S}{5}\ (unit: ^{\circ}\mathrm{C}) \tag{S52}$$

During the sorption process, the initiated humidity around the water sorbent is $d_{\mathrm{Ad1,w}}$, where the water content in the sorbent is W_{dry}. Thus, the humidity difference drives the moisture capture and is finally halted when $d_{\mathrm{Ad2,w}}$ is close to the humidity of inlet air $d_{\mathrm{amb,S}}$ with only a margin value of Δd_{Ad} (equation (S53)). The final water content in the sorbent is W_{sat}, and the difference between W_{dry} and W_{sat} is the cyclic water content $\Delta W_{\mathrm{cycle}}$ (assumed as 0.2 g g^{-1}, equation (S54)).

$$\Delta d_{\mathrm{Ad}} = d_{\mathrm{amb,S}} - d_{\mathrm{Ad2,w}} = 0.2\ SWHR_{\mathrm{S}}\ (unit: \mathrm{g\ kg_{air}^{-1}}) \tag{S53}$$

$$\Delta W_{\mathrm{cycle}} = W_{\mathrm{sat}} - W_{\mathrm{dry}} = 0.2\ \ (unit: g\ g_{\mathrm{sorbent}}^{-1}) \tag{S54}$$

Once the sorbent is heated, the absorbed water cannot be released immediately. A new equilibrium point with a humidity $d_{\mathrm{De1,w}}$ ($d_{\mathrm{De1,w}} < d_{\mathrm{Ad2,w}}$) is established under the desorption temperature $T_{\mathrm{De,w}}$. The humidity difference between $d_{\mathrm{De,in}}$ (the humidity of inlet desorption air) and $d_{\mathrm{De1,w}}$ drives the water release until the water sorbent is dried off to W_{dry} with a humidity $d_{\mathrm{De2,w}}$. The averaged water content in the water sorbent during the desorption process can be estimated by equation (S55) with a margin Δd_{De} between $\bar{d}_{\mathrm{De,out}}$ and $\bar{d}_{\mathrm{De,w}}$ defined by equation (S56). The desorption process proceeds when equation (S57) is satisfied and the humidity difference between the inlet and outlet air is the $SWHR_{\mathrm{S}}$ (equation (S58)).

$$\bar{d}_{\mathrm{De,w}} = \frac{\int_{W_{\mathrm{dry}}}^{W_{\mathrm{sat}}} d_{\mathrm{De}}(W)dW}{W_{\mathrm{sat}} - W_{\mathrm{dry}}}\ (unit: g\ kg_{air}^{-1}) \tag{S55}$$

$$\Delta d_{De} = \bar{d}_{\mathrm{De,w}} - \bar{d}_{\mathrm{De,out}} = 0.4\ SWHR_{\ S}\ (unit: g\ kg_{air}^{-1}) \tag{S56}$$

$$d_{\mathrm{De1,w}} > d_{De,in}, d_{De,in} = f(T_{De,in}) \tag{S57}$$

$$SWHR_{\ S} = \bar{d}_{De,w} - d_{De,in}\ (unit: g\ kg_{air}^{-1}) \tag{S58}$$

The heat flux of desorption process can be calculated by the following equations (S59)-(S62):

$$Q_{\mathrm{Solar-S}} = R_{\mathrm{De,S}} \cdot \left(i_{\mathrm{De,out}}^{*} - i_{\mathrm{De,in}}^{*}\right) + m_{\mathrm{w}} c_{\mathrm{p,w}} \left(T_{\mathrm{Cond,w}} - T_{\mathrm{Ad,w}}\right) \tag{S59}$$

$$m_{\mathrm{w}}=\frac{R_{\mathrm{De,S}} \cdot SWHR_{\mathrm{S}}}{\Delta W_{\mathrm{cycle}}} \tag{S60}$$

$$i^{*} = c_{p,air}T_{air} + d_{air}\left(c_{p,vapor}T_{air} + h_{\mathrm{w}}\right) \tag{S61}$$

$$i = c_{p,air}T_{air} + d_{air}\left(c_{p,vapor}T_{air} + h_{fg}\right) \tag{S62}$$

where h_{w} is the water sorption/desorption enthalpy of sorbents, assumed it is equal to 2700 kJ kg^{-1}; m_{w} represents the mass of water sorbent.

The conditions of outlet desorbed air can be calculated by equation (S63):

$$T_{\mathrm{De,out}} = \max\left(\eta_{\mathrm{De,T}}\left(T_{\mathrm{De,w}} - T_{\mathrm{De,in}}\right) + T_{\mathrm{De,in}}, T_{\mathrm{De,in}}\right), \eta_{\mathrm{De,T}} = 0.8 \tag{S63}$$

Finally, $SWP_{\mathrm{Solar\text{-}S}}$ can be calculated by equations (S64):

$$SWP_{\mathrm{Solar\text{-}S}}=75\% \cdot \frac{SWHR_{\mathrm{S}} \cdot R_{\mathrm{De,S}}}{Q_{\mathrm{Solar-S}}} \tag{S64}$$

The optimal $T^{*}_{\mathrm{De,S}}$, $SWHR^{*}$, and $SWP^{*}_{\mathrm{Solar\text{-}S}}$ can be obtained by calculating the solutions of the optimization problem in equation (S65), and we calculate this process utilizing C++ software (Code S4)

$$\begin{cases} \left\{T^{*}_{\mathrm{De,S}}\right\}=\arg\max\ \left\{SWP_{\mathrm{S}},\ \text{s.t. } SWHR\left(T_{\mathrm{De,S}}\right)\geq 0\right\} \\ T_{\mathrm{De,S}} \in (T_{\mathrm{amb}},\ 100\ ^{\circ}\mathrm{C}) \\ SWHR^{*}=SWHR\left(T^{*}_{\mathrm{De,S}}\right),\ SWP^{*}_{\mathrm{Solar\text{-}S}}=SWP_{\mathrm{S}}(T^{*}_{\mathrm{De,S}}) \end{cases} \tag{S65}$$

where 100 °C is defined as the highest desorption temperature considering the solar-thermal conversion capacity empirically.

Note S6 Theoretical annual water production potential of the HAWH system driven by a square meter PV panel

Considering the HAWH system is powered by PV panels, we evaluated the theoretical annual water productivity of HAWH system driven by one square meter of PV panels (kg m^{-2}) under various global climates. We comprehensively considered the impact of solar irradiation, ambient temperature, and local wind on the electricity generation efficiency of PV panels. The calculated process is illustrated as follows:

The solar-to-electricity conversion efficiency of PV can be calculated by equation S66[7]:

$$\eta_{PV}=\eta_{ref}\cdot[1-\beta\cdot(T_{PV}-T_{ref})] \quad (S66)$$

where η_{ref} is the reference photovoltaic efficiency at the reference temperature (T_{ref}, 25 °C), β is the temperature coefficient of the PV efficiency.

T_{PV} is expressed by equation S67[8]:

$$T_{PV}=T_{amb}+\frac{G}{U_0+u\cdot U_1} \quad (S67)$$

where T_{amb} is the ambient temperature, U_0 and U_1 are constants of the PV module, and u is the wind speed (m s^{-1}).

Hence, the annual electricity production of a square meter PV panel can be described as[7]:

$$E=\eta_{PV}\cdot G\cdot d\cdot\tau\cdot\alpha\cdot\gamma\cdot 10^{-3} \quad (S68)$$

where E is the annual electricity production of a square meter PV panel (kWh m^{-2} $year^{-1}$), G is the solar irradiation (W m^{-2}), d is the duration of solar radiation (h $year^{-1}$), τ is the transmissivity of the PV-glazing cover, α is the absorptance of the PV cells, γ is the packing factor.

Restricting by the available data, we took the solar irradiation data per 3h as the average solar irradiation during this 3-hour period, and calculated the annual electricity production of a square meter PV panel by equation S69.

$$E=\sum_{n=1}^{365}[\sum_{i=1}^{8}(3\cdot\eta_{PV}\cdot G\cdot\tau\cdot\alpha\cdot\gamma)] \quad (S69)$$

Finally, we can get the theoretical annual water production potential of the HAWH system driven by a square meter PV under various global climates by equation S70:

$$SWP_{PV} = E \cdot SWP \tag{S70}$$

where SWP_{PV} is the annual water production through a square meter PV panel (kg m^{-2}).

We listed the used parameters for calculation in Table S5, and the calculated result is listed in Fig. S12.

Note S7 Technoeconomic analysis of HAWH system

The economic feasibility of HAWH pathway is one of the main points impacting its applicability. To illustrate its commercial potential, we conducted a technoeconomic analysis of HAWH system to calculate the water production costs of HAWH system.[9] The total cost for HAWH system mainly includes the cost of PV panels, heat pumps, and sorbents, whereas the operational cost is neglected for simplicity because of its intelligent switch operation. Hence, the unit price of produced water P_w (CNY L^{-1}) can be estimated by the following equation S71.

$$P_w = P_{PV}+P_{HP}+P_{sorb} \tag{S71}$$

P_{PV} is the price of PV panels per unit water, which can be calculated by the equation S72:

$$P_{PV}=\frac{C_{PV}}{SWP} \tag{S72}$$

where C_{PV} is the levelized cost of energy generated by PV panels (CNY kWh^{-1}) and SWP is our calculated specific water production per kWh (L kWh^{-1}) by HAWH system.

P_{HP} is the price of heat pumps per unit water, which can be calculated by the equation S73:

$$P_{HP}=\frac{p_{power} \cdot C_{HP}}{t_{life,HP} \cdot m_{water}} \tag{S73}$$

where p_{power} is the heat/cold power required from the heat pump through the whole year water production process (kW) and this is the indictor for choosing suitable heat pump, C_{HP} is the cost of heat pumps according to its heat supply (CNY kW^{-1}), $t_{life,HP}$ is the lifespan of a heat pump (year), and m_{water} is the water production per year by HAWH system (L_{water} $year^{-1}$).

p_{HP} can be calculated by equation S74:

$$p_{power}=\frac{h_w \cdot m_{water}}{3600 \cdot t_{op}} \tag{S74}$$

where h_w is the desorption enthalpy (kJ kg_{water}^{-1}), t_{op} is the annual operation time of HAWH (h).

P_{sorb} is the price of sorbents per unit water, which can be calculated by the equation S75:

$$P_{sorb}=\frac{m_{sorb} \cdot C_{sorb}}{t_{life,sorb} \cdot m_{water}} \tag{S75}$$

where m_{sorb} is the mass of sorbents required per water sorption-desorption cycle (kg_{sorb}), C_{sorb} is the cost of sorbents (CNY kg_{sorb}^{-1}), $t_{life,sorb}$ is the lifespan of the sorbent (year).

m_{sorb} can be calculated by equation S76:

$$m_{sorb}=\frac{m_{water}}{\Delta W_{cycle}\cdot t_{op}} \quad (S76)$$

where ΔW_{cycle} is the cyclic water uptake per sorbent for 3h (L_{water} kg_{sorb}^{-1}), because we define the mode switch time for HAWH is 3h. We selected two representative sorbents (one is the commercial $CaCl_2$ salt with low price and the other is representative MOFs (MIL-101(Cr)) with a relatively high price) for this calculation. The data of the cost of sorbents and cyclic water uptake per sorbents in 3h are adopted from reported articles[10,11].

According to the above calculation, we can get the theoretical P_w of HAWH system under global regions (Fig. 6c and Fig. S13).

Although this technoeconomic analysis needs further optimization based on the actual system owing to the unavailable and inconsistent costs, such as the deployment and maintenance costs, it can theoretically indicate the economic feasibility of this HAWH system. Since the global drinking water costs is hard to available, we selected several typical cities to compare the P_w of HAWH system versus local drinking water costs, covering arid, semi-arid, semi-humid, and humid climate conditions. As shown in Fig. S14, based on the reported price of sorbents, the HAWH system can generate drinking water at a price lower than the current drinking water costs selling in the market in all of these cities theoretically. These results indicate the technoeconomic feasibility of HAWH, especially for these aeras without perfect infrastructures for tap water. Notably, compared to the long lifespan time for PV panels and heat pumps, the sorbents with short lifespan are the main factors affecting the water price in terms of horizontal comparison, which is the future development focus. With the low-cost and scalable production of sorbents in the near future, the unit price of water produced by HAWH system can be reduced.

The parameters used for this evaluation are presented in Table. S6.

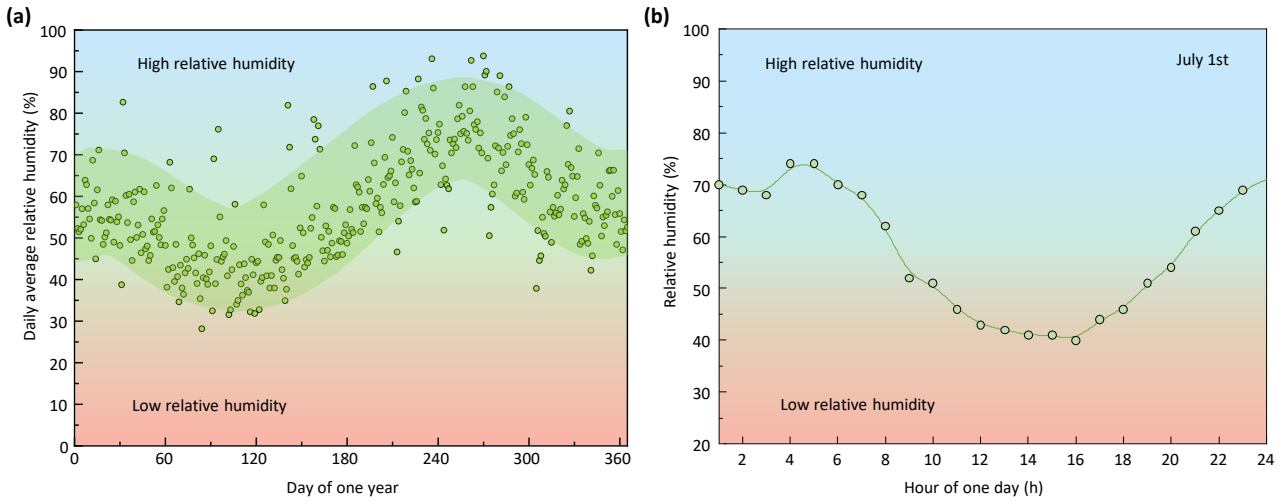

Fig S1 Actual seasonal and diurnal fluctuations of air RH in Lanzhou, CN during (a) one year and (b) one day.

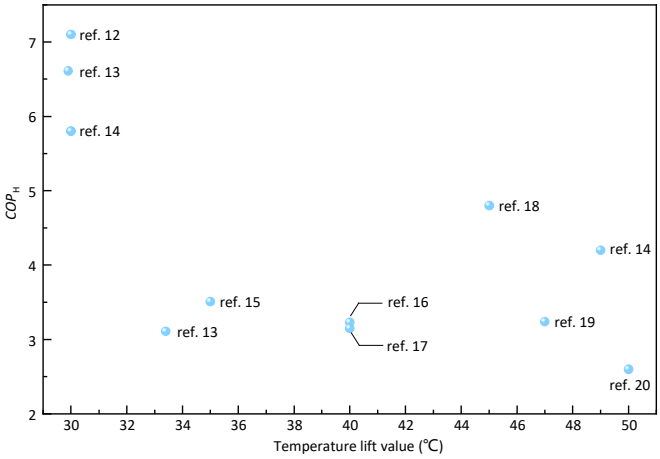

Fig S2 Coefficient of performance of heat (COP_H) versus the temperature lift of the reported state-of-the-art heat pumps.

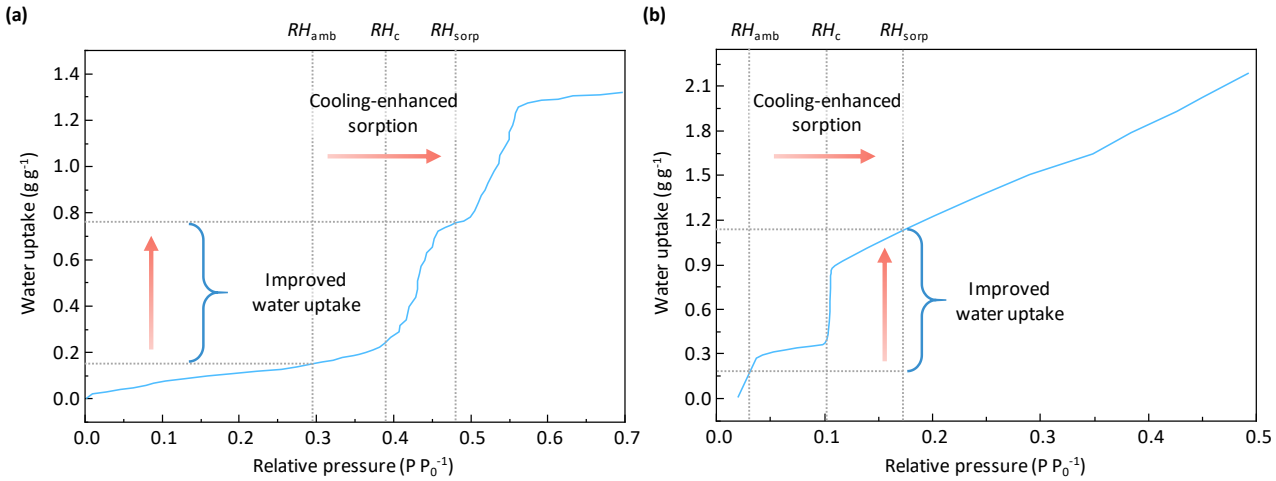

Fig S3 Schematic illustration of the enhancement of cold energy on water sorption process. (a) Water sorption enhancement of MIL-101(Cr), representing the sorbents with steep step-like S shape isothermal water sorption curves. (b) Water sorption enhancement of LiCl@rGO-SA, representing the composites sorbents mainly relies on hygroscopic salts for water sorption. The sorption isotherms data are adopted from ref. 4 and 5.

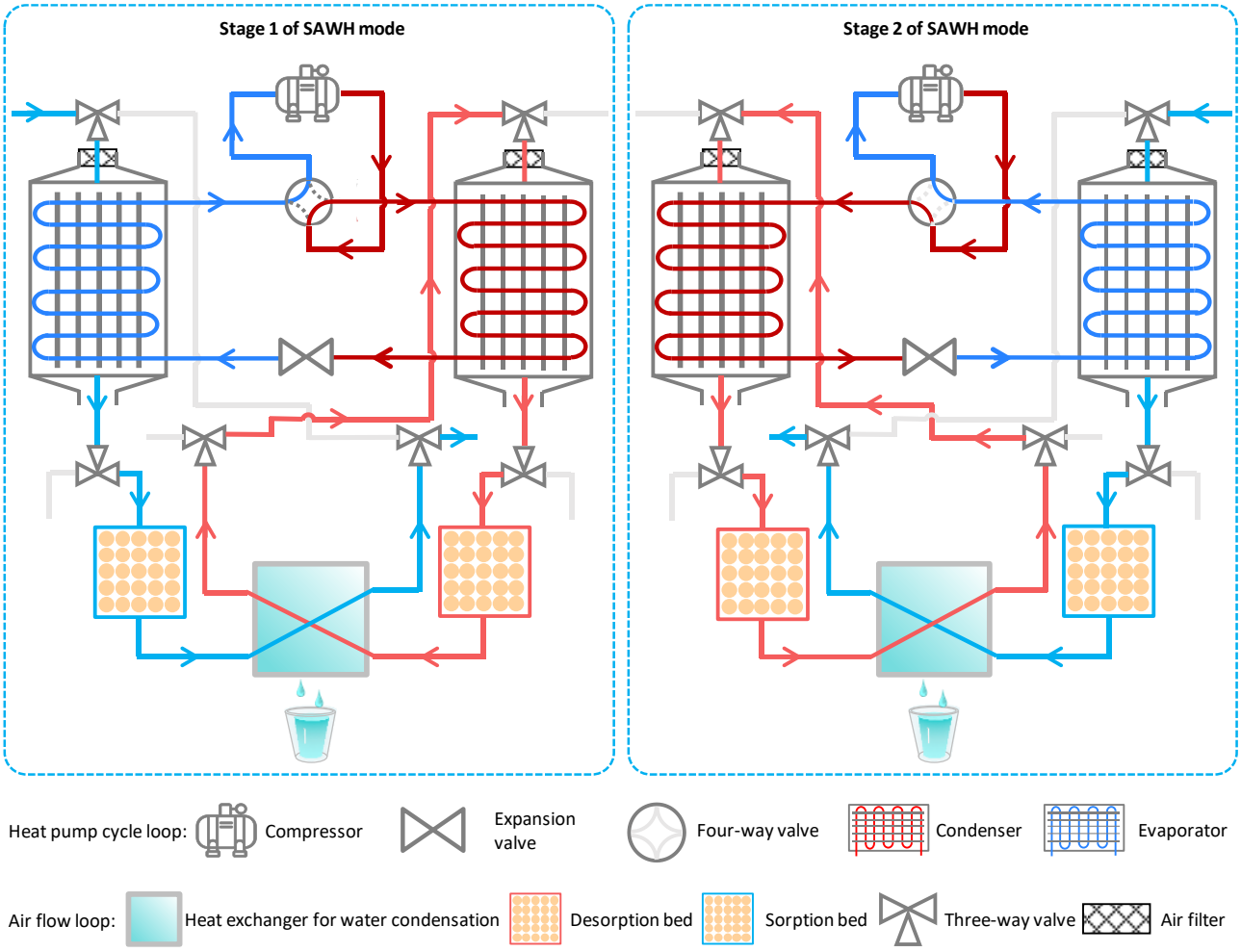

Fig. S4 Schematic diagram showing the two stages of SAWH mode for continuous water harvesting by intelligent switch.

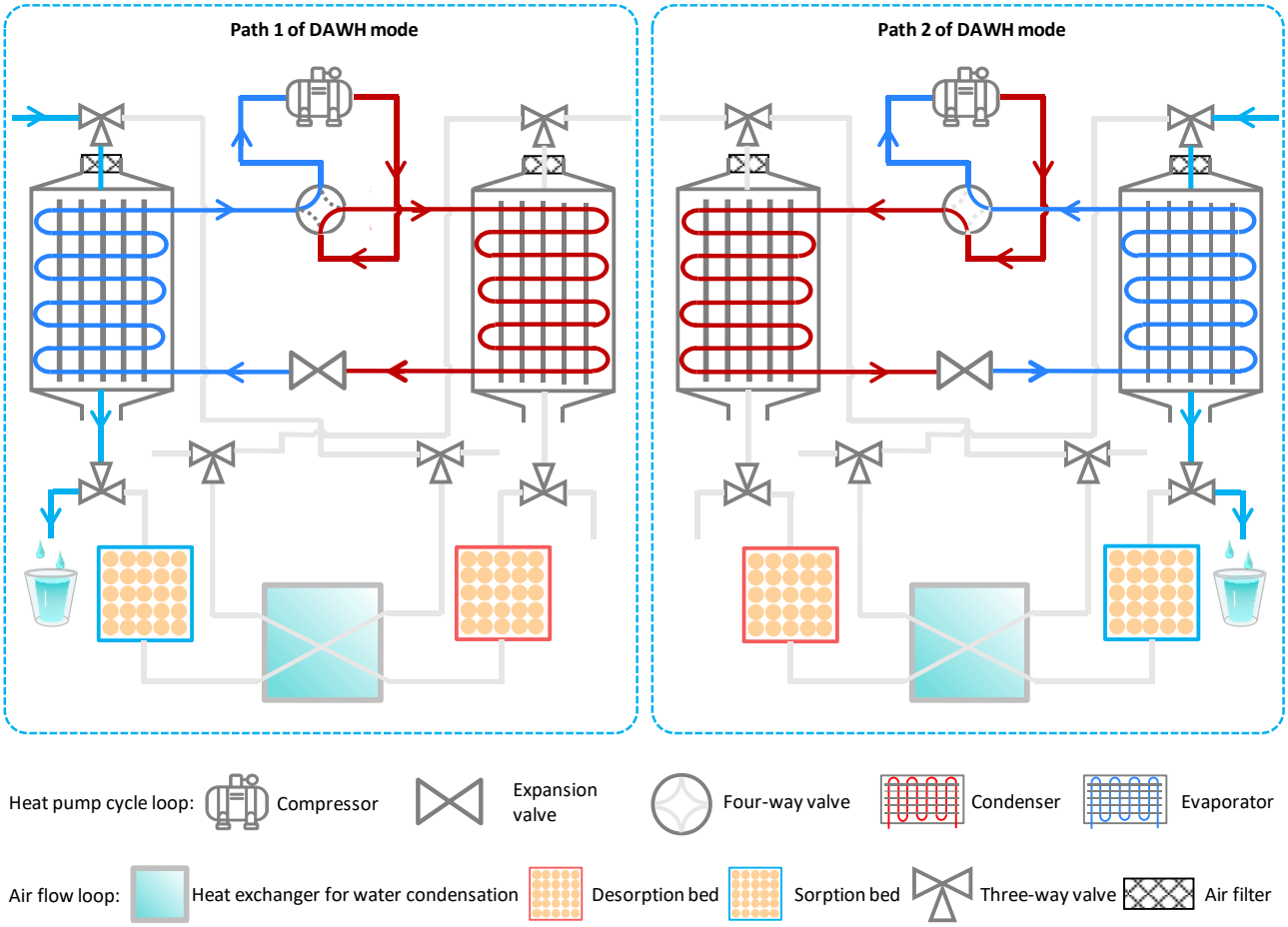

Fig. S5 Schematic diagram of the two operation pathways for DAWH mode, depending on which side is the evaporator of the heat pump.

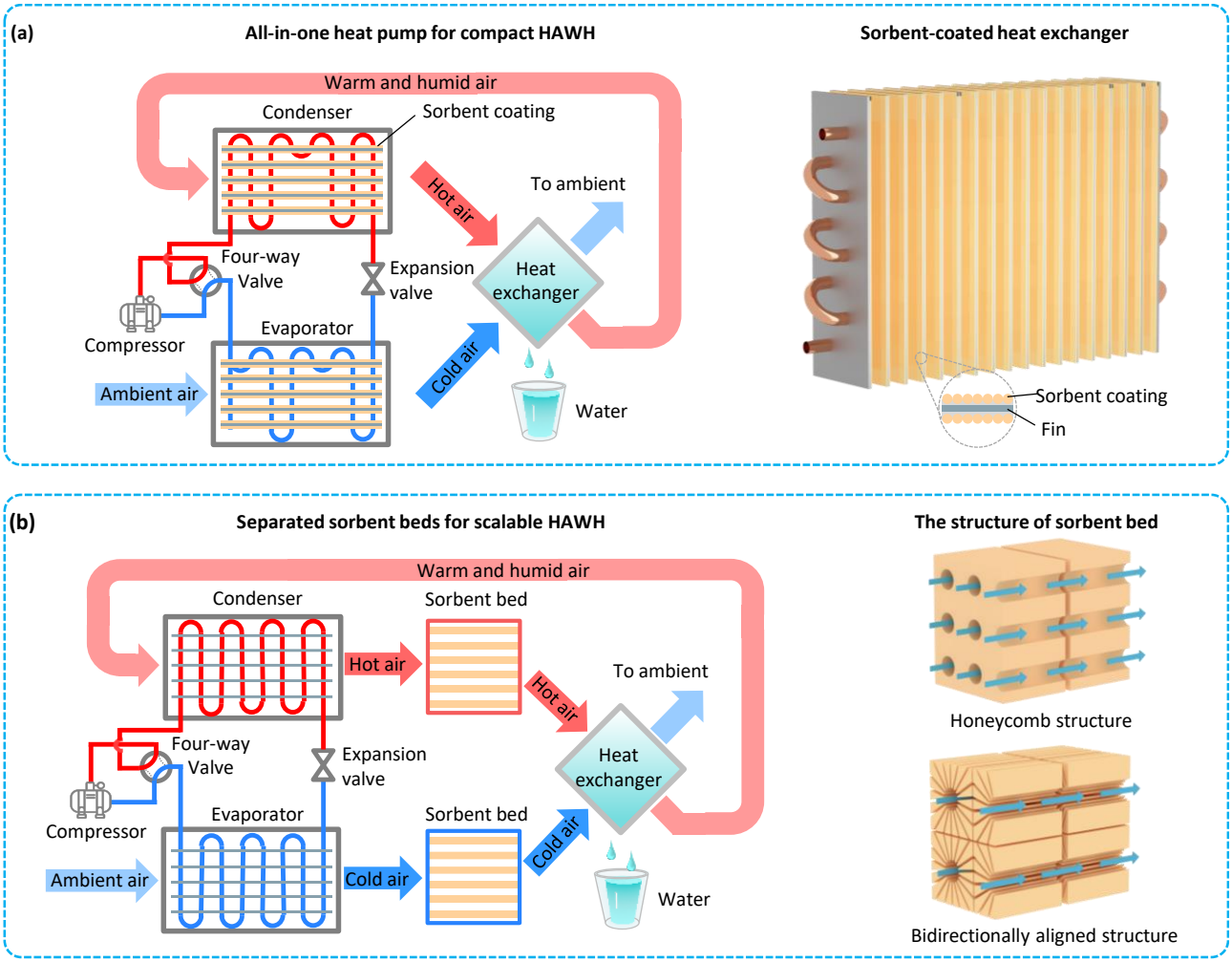

Fig. S6 Schematic illustration of two possible pathways for combining heat pumps and sorbent materials. (a) Coating sorbents on the fin surfaces of the condenser and evaporator for designing compact and portable HAWH devices. (b) Separating sorbent beds from heat pumps for developing scalable HAWH devices.

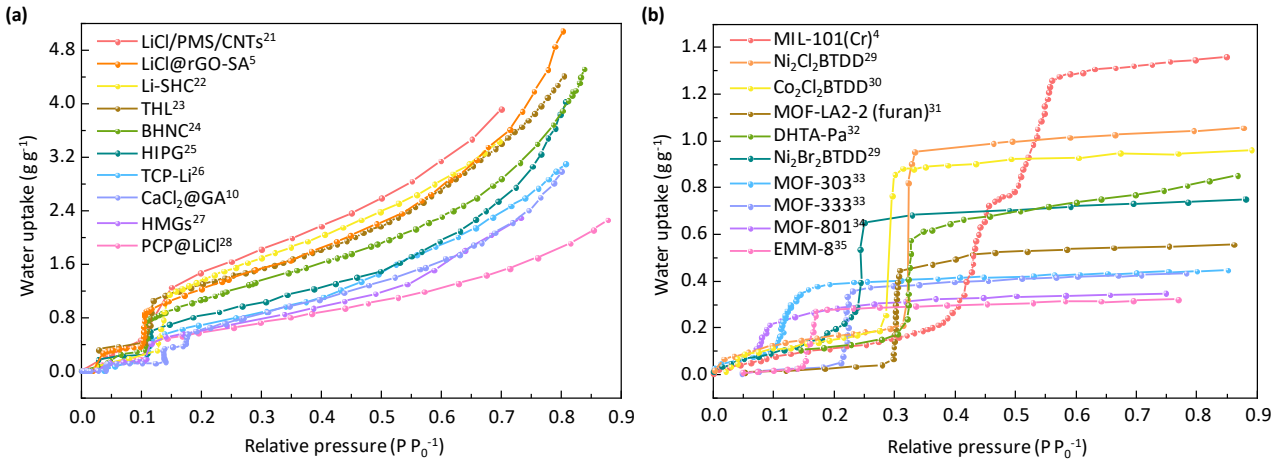

Fig. S7 Summarized water sorption isotherms of candidate sorbents for HAWH. (a) Hygroscopic salt-based composites with wide-range multi-step sorption characteristics. (b) Physical sorbents with S-shaped isothermal water sorption curves. The isotherms are adopted from Refs. 4, 5, 10, 21-35.

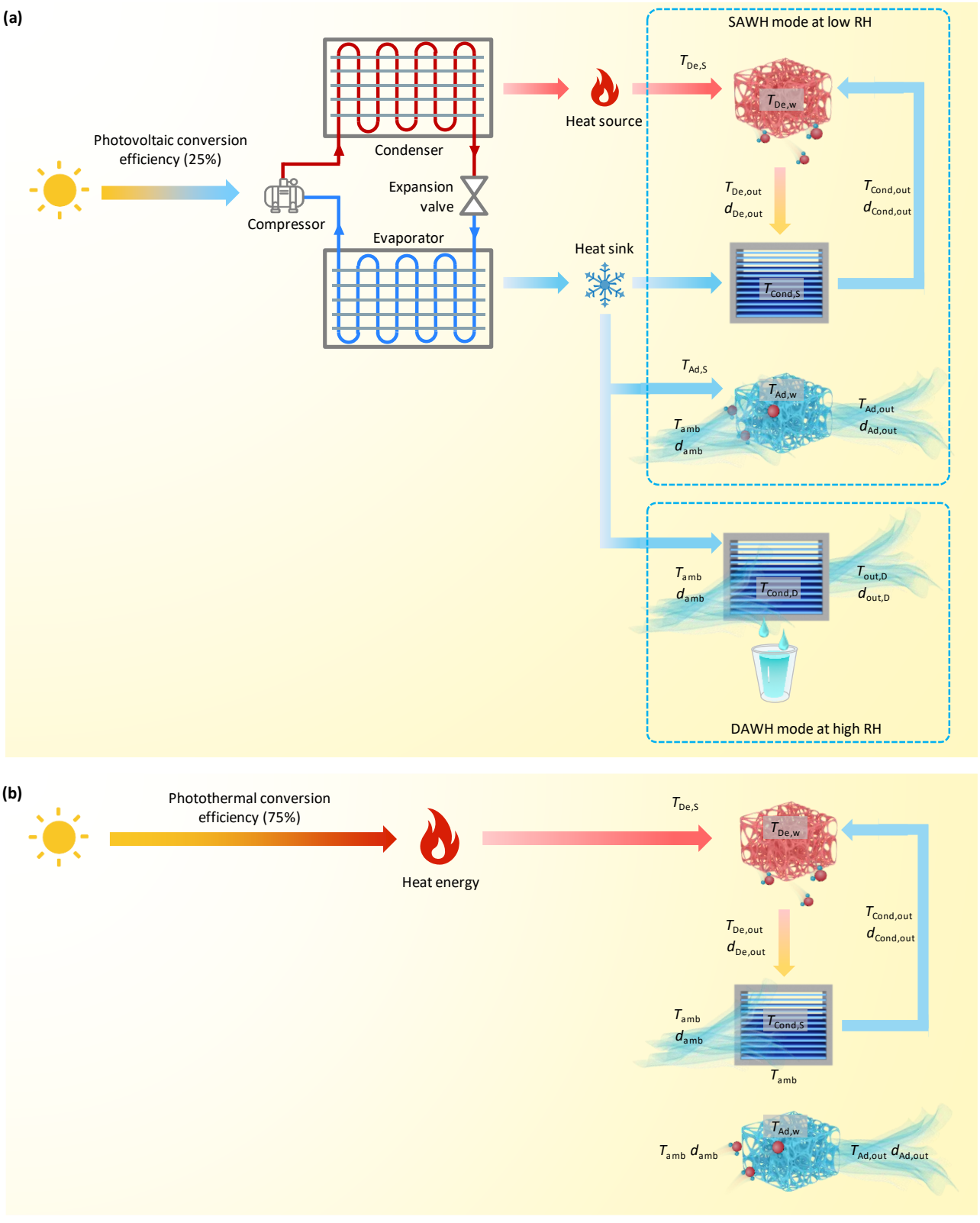

Fig. S8 The process and calculation parameters of solar-electric-driven HAWH system and conventional solar-thermal-driven SAWH system.

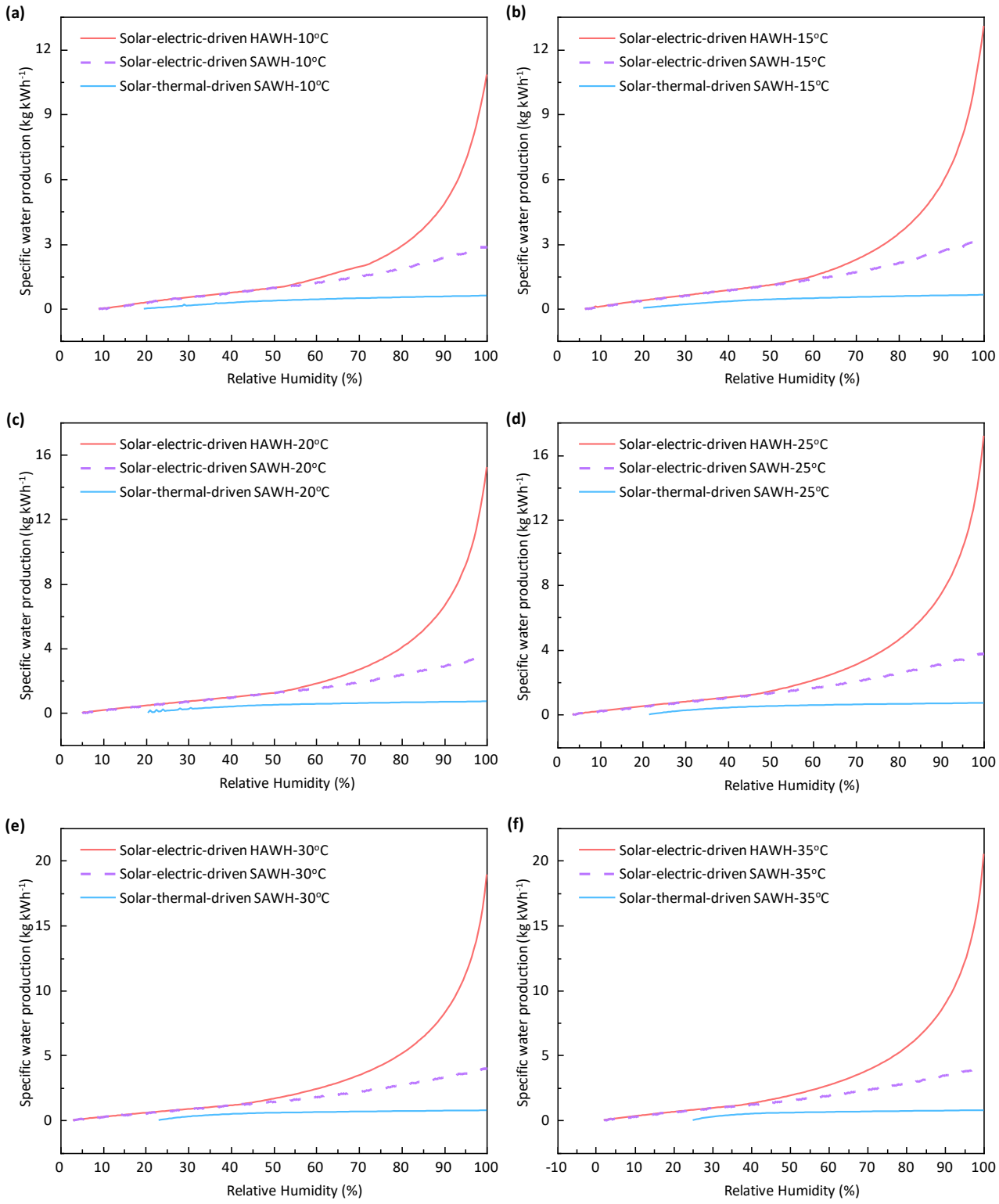

Fig. S9 Calculated specific water production of solar-electric-driven HAWH, solar-electric-driven SAWH, and solar-thermal-driven SAWH under different ambient environments. (a-f) corresponding to the ambient temperature at 10°C, 15°C, 20°C, 25°C, 30°C, 35°C.

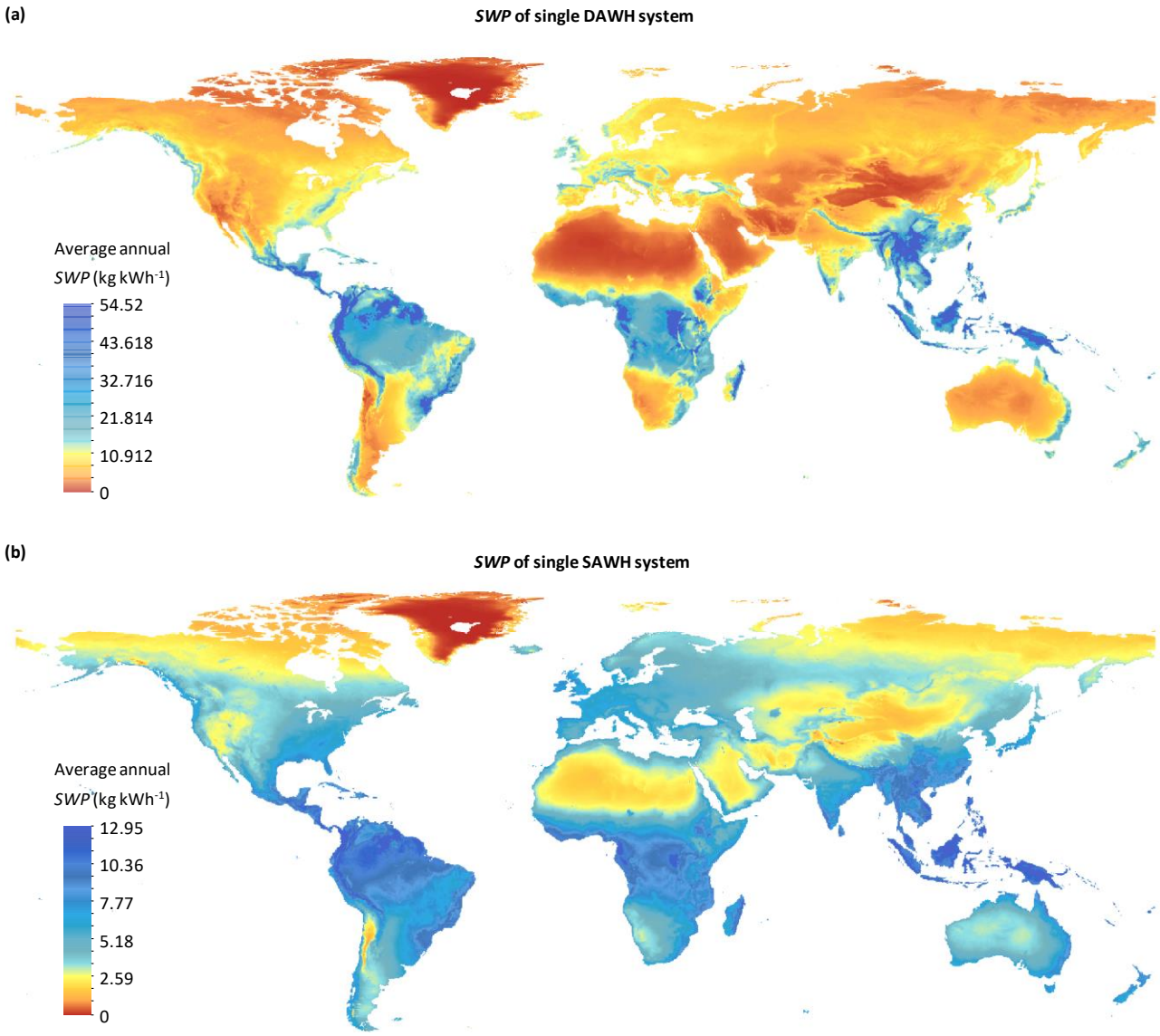

Fig. S10 Average annual *SWP* of conventional DAWH and SAWH system under various global climates. (a) Calculated results of single DAWH system. (b) Calculated results of single SAWH system.

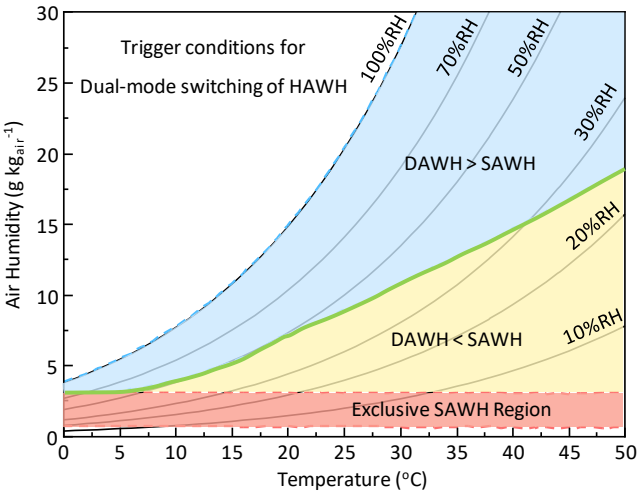

Fig. S11 Theoretical triggering conditions for dual-mode switching of HAWH.

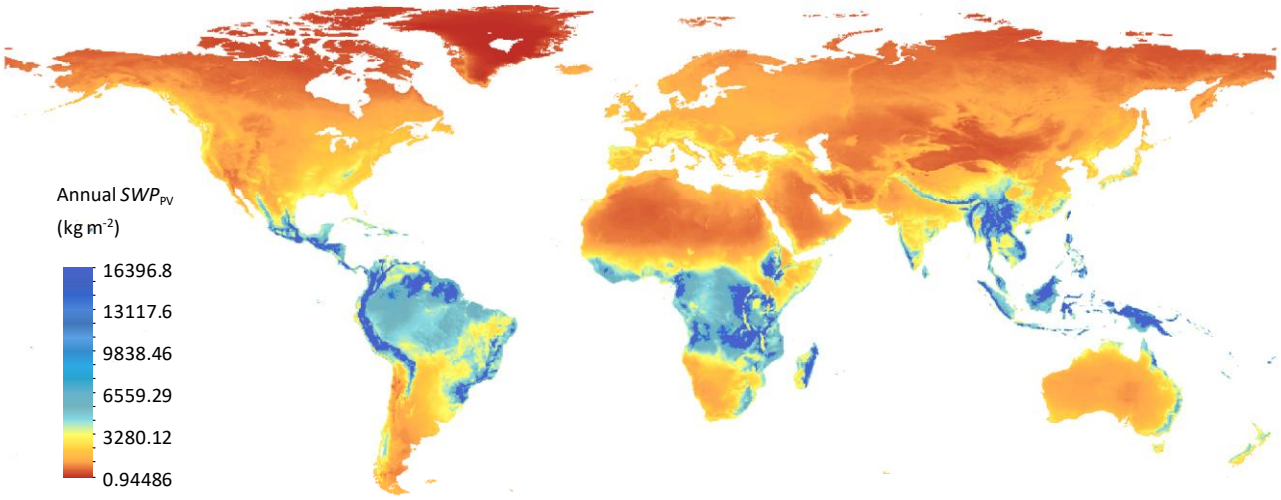

Fig. S12 Annual SWP_{PV} of HAWH system under various global climates considering the impacts of weather on PV panels.

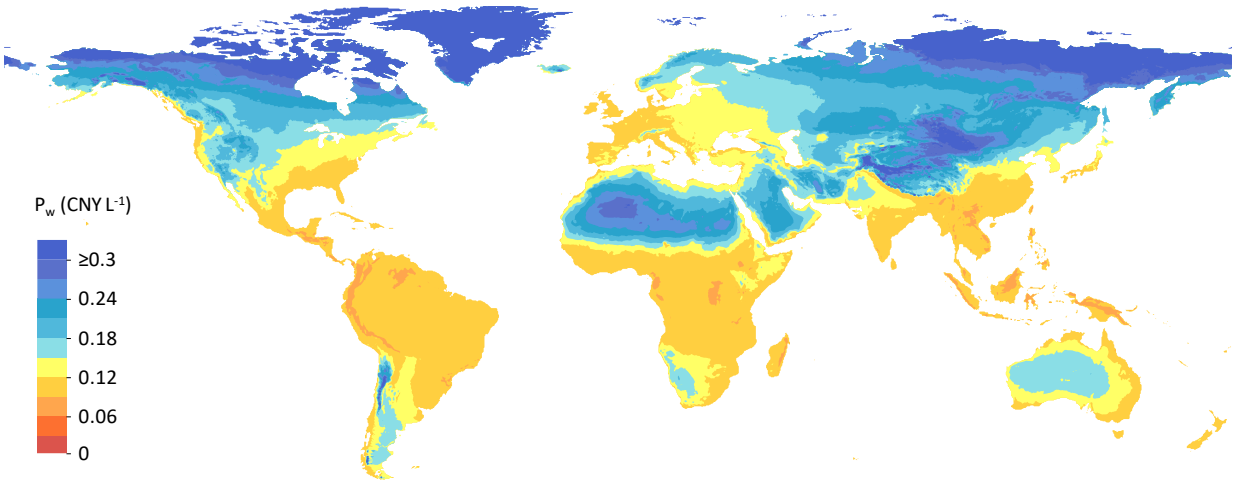

Fig. S13 Theoretical P_w of HAWH system under various global climates with MIL-101(Cr) as sorbent.

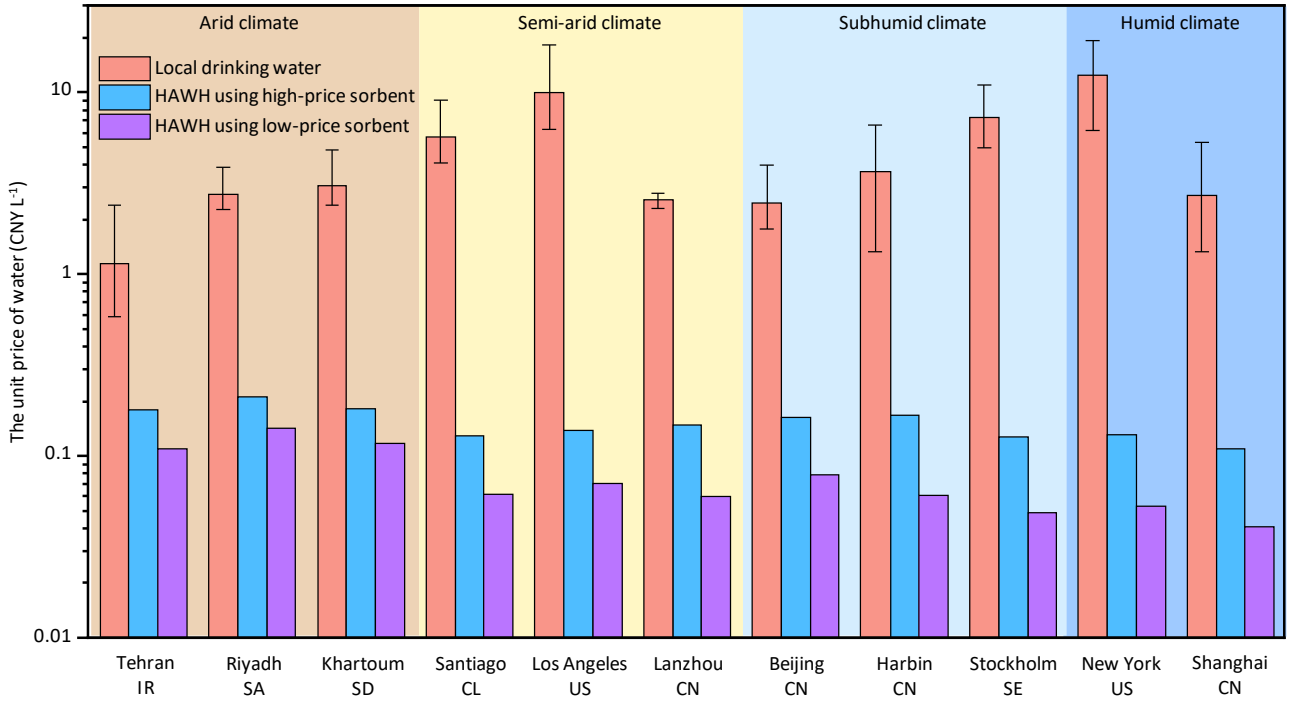

Fig. S14 Theoretical P_w of HAWH system versus local drinking water costs selling in the market in several typical cities, covering arid, semi-arid, semi-humid, and humid climate conditions. The error bars of local drinking water represent the maximum and minimum water prices of the whole statistical sample.[36]

Table S1 Summary of the water production performance of reported SAWH devices

Materials	Test site	Sorption RH (%)	Water production rate ($L\ m^{-2}\ h^{-1}$)	Energy input ($W\ m^{-2}$)	Energy efficiency (%)	Specific water production ($L\ kWh^{-1}$)	Cited literature
MOF-801	Outdoor	/	0.04	700	3.67	0.05	37
HIPG	Indoor	54-61	0.26	1000	18.33	0.26	25
HIPG	Indoor	55-60	1.86	4000	32.29	0.47	25
HIPG	Outdoor	25.6-33.5	1.20	3500-4000	22.22	0.32	25
MOF-303/Graphite	Outdoor	35	0.13	950	9.25	0.13	38
LiCl@rGO-SA	Indoor	60	/	950-1050	22.80	0.33	5
BHNC	Indoor	60	/	Electric heater	20.20	0.29	24
LiCl@rGO-SA	Outdoor	50	0.36	~970	26.10	0.38	6
MOF-801	Outdoor	20-40	/	1800	14.00	0.20	39
HCS-LiCl	Indoor	60	0.13	1000	8.90	0.13	40
AQSOA Z01	Outdoor	68	0.14	600-1000	9.00	0.13	41
MOF-801	Outdoor	19-46	/	Electric heater	13.19-41.67	0.19-0.6	42
MIL-101(Cr)@CF	Outdoor	65-70	/	800-1000	21.70	0.31	43
LiCl solution	Indoor	90	0.65	1000	45.14	0.65	44
LiCl solution	Indoor	65	0.22	1000	15.28	0.22	45
[EMIM][Ac]	Indoor	80	0.50	1000	34.72	0.50	45
[EMIM][Ac]	Indoor	80	0.70	1600	30.38	0.44	45
[EMIM][Ac]	Outdoor	70	0.40	1100	25.25	0.36	45
$CaCl_2$ solution	Indoor	/	0.63	700	62.50	0.90	46
$CaCl_2$ solution	Indoor	/	0.73	1000	50.69	0.73	46
$CaCl_2$ solution	Indoor	/	1.16	1500	53.70	0.77	46
$CaCl_2$ solution	Indoor	/	1.40	2000	48.61	0.70	46

Table S2 Parameters used for the assessment of energy consumption of AWH[1]

Nomenclature		Subscript	
T	temperature (°C or K)	amb	ambient
d	humidity (g kg_{air}^{-1})	sat	saturated
P	pressure (Pa)	out	outlet air
RH	relative humidity	in	inlet air
$SWHR$	specific water harvesting rate (g kg_{air}^{-1})	D	DAWH
Q	heat flux (kW)	S	SAWH
R	air flow rate (kg s^{-1})	Cond	Condenser
i	specific enthalpy (kJ kg^{-1})	Ad	Adsorption
c_p	specific heat (kJ kg^{-1} K^{-1})	De	Desorption
h_{fg}	latent heat of water vaporization (kJ kg^{-1})	w	water sorbent
Ex	exergy transfer rate (kW)	m	matrix of water sorbent
SWP	specific water production (kg kWh^{-1})	dry	dry water sorbent without water adsorption
W	water content in the sorbent (g $g_{sorbent}^{-1}$)	m	matrix of water sorbent
m	mass (kg)		
i^*	equivalent specific enthalpy (kJ kg^{-1})		
h_w	Adsorption/desorption enthalpy ((kJ kg^{-1})		

Parameters			
$c_{p,air}$	1.01 kJ kg^{-1} K^{-1}	h_w	2700 kJ kg^{-1}
$c_{p,vapor}$	1.84 kJ kg^{-1} K^{-1}	h_{fg}	2500 kJ kg^{-1}
$c_{p,w}$	1 kJ kg^{-1} K^{-1}	ΔW_{cycle}	0.2 g $g_{sorbent}^{-1}$
$c_{p,m}$	1 kJ kg^{-1} K^{-1}	r_m	1

Table S3 Summary of reported hygroscopic salt-based composites with wide-range multi-step sorption characteristics

Sorbent	Salt content (wt%)	Water uptake capacity ($g\ g^{-1}$)	Sorption equilibrium time (min)	Desorption equilibrium time (min)	Reference
LiCl/PMS/LiCl	91.6	1.81 (25°C, 30%RH) 3.13 (25°C, 60%RH)	160 (30%RH) 220 (60%RH)	90 (under 1 sun irradiation, 75°C)	21
LiCl@rGO-SA	78	1.52 (30°C, 30%RH) 2.75 (30°C, 60%RH)	200 (30°C, 30%RH) 450 (30°C, 60%RH)	120 (90°C, 4250Pa)	5
Li-SHC	90	1.79 (30%RH) 2.93 (60%RH)	180 (30%RH) 360 (60%RH)	>120 (110°C, ambient)	22
THL	79.4	1.54 (25°C, 30%RH) 2.86 (25°C, 60%RH)	90 (25°C, 30%RH) 80 (25°C, 60%RH)	30 (under 1 sun irradiation, 90°C)	23
BHNC	69	1.36 (25°C, 30%RH) 2.36 (25°C, 60%RH)	180 (30°C, 30%RH) 360 (30°C, 60%RH)	30 (90°C, 1.9 m s^{-1}) 60 (90°C, 1.1 m s^{-1})	24
HIPG	54	1.01 (25°C, 30%RH) 2.03 (25°C, 60%RH)	40 (25°C, 30%RH) 100 (25°C, 60%RH)	30 (under 1 sun irradiation, 55°C)	25
TCP-Li35	35	0.9 (25°C, 30%RH) 1.93 (25°C, 60%RH)	600 (30°C, 60%RH)	50 (under 1 sun irradiation, 70°C)	26
$CaCl_2$@GA	91-96	0.82 (25°C, 30%RH) 1.67 (25°C, 60%RH)	420 (30%RH) 360 (60%RH)	180 (80°C)	10
HMGs	56	0.8 (25°C, 30%RH) 1.58 (25°C, 60%RH)	75 (25°C, 15%RH) 60 (25°C, 30%RH)	30 (60°C, 3170Pa)	27
PCP@LiCl	46	0.89 (30%RH) 1.48 (60%RH)	100 (30°C, 30%RH) >250 (30°C, 60%RH)	30 (55°C, 3170 Pa)	28

Table S4 Summary of reported sorbents with S-shaped water sorption isotherms

Sorbent	Water uptake capacity (g g^{-1})	Critical RH (%)	Desorption enthalpy (kJ mol^{-1})	Desorption temperature (°C)	Reference
MIL-101(Cr)	1.36	38	46	60	4
Ni_2Cl_2BTDD	1.07	32	57	53	29
Ni_2Br_2BTDD	0.76	24	57	58	29
Co_2Cl_2BTDD	0.96	29	45.8-55	57	30
MOF-LA2-1 (furan)	0.47	14	53	65	31
MOF-LA2-2 (furan)	0.57	30	50	50	31
DHTA-Pa	0.92	17	47-50	40	32
MOF-303	0.46	15	53	60	33
MOF-333	0.43	21	50	50	33
MOF-801	0.35	9	60	85	34
EMM-8	0.31	15	46.8	65	35

Table S5 Parameters used for the calculation of operational efficiency of solar panels

Parameters	Definition	Values	Units
η_{ref}	Reference photovoltaic efficiency	0.182[47]	-
β	Temperature coefficient of the PV efficiency	0.0043[47]	$°C^{-1}$
T_{ref}	Reference temperature	25[47]	°C
U_0	Constant of PV module	30.02[8]	$W\ m^{-2}\ °C^{-1}$
U_1	Constant of PV module	6.28[8]	$J\ m^{-3}\ °C^{-1}$
τ	Transmissivity of PV-glazing cover	0.9[7]	-
α	Absorptance of PV cells	0.85[7]	-
γ	Packing factor	1[7]	-

Table S6 Parameters used for the technoeconomic analysis of HAWH system

Parameters	Definition	Values	Units
C_{PV}	Levelized cost of energy generated by PV panels	0.34[48]	CNY kWh^{-1}
C_{HP}	Cost of heat pumps according to its heat supply	1800[49]	CNY kW^{-1}
$t_{life,HP}$	Lifespan of a heat pump	15[49]	Year
$t_{life,sorb}$	Lifespan of the sorbent	1	Year
$C_{sorb,\ CaCl2}$	Cost of $CaCl_2$	0.75[50]	CNY kg_{sorb}^{-1}
$\Delta W_{cycle,\ CaCl2}$	Cyclic water uptake per $CaCl_2$ for 3h	1[10]	L_{water} kg_{sorb}^{-1}
$C_{sorb,\ MIL\text{-}101}$	Cost of MIL-101(Cr)	574.76[11]	CNY kg_{sorb}^{-1}
$\Delta W_{cycle,\ MIL\text{-}101}$	Cyclic water uptake per MIL-101(Cr) for 3h	2.865[11]	L_{water} kg_{sorb}^{-1}

Supplemental Code

Code S1 Codes of C++ for analyzing the energy consumption of conventional single DAWH

```
/**Standard Library**/
#include <iostream>
#include <cmath>
#include <fstream>

using namespace std;

/**Overall environment**/
double Patm = 1.01325e5;   /**standard atmospheric pressure**/
double hfg = 2500;         /**Latent heat of water vaporization**/
double InterP(int index, double A[], double B[], double a);

int main(){

   /**Input\Output files**/
   ifstream PTrelation("0-SaturatedPressure.txt"); /**Input table of the relationship between temperature and saturation water vapor pressure**/
   ofstream CErelation;
   CErelation.open("1-output.txt");

   ofstream Output;
   ofstream Output1, Output2, Output3;
   Output.open("1-FinalOutput.txt");
   Output1.open("2-FinalOutput1.txt");
   Output2.open("2-FinalOutput2.txt");
   Output3.open("2-FinalOutput3.txt");
```

```
int PTnum, p;
PTrelation>>PTnum;
double MT[PTnum+1], MPvs[PTnum+1];
for(p = 0; p <= PTnum+1; p++){PTrelation>>MT[p]>>MPvs[p];}

/**Paramters for calculation**/
double DeltTD;     /**Temperature difference between the outlet airflow of condenser and the cold source**/

double Tin_MIN = 0, Tin_MAX = 50, Tin_Delt = 0.5;              /**Temperature boundary and variation of the input air source**/
double RHin_Delt = 0.2;                                        /**RH variation of the input air source**/
double SWHR_MIN = 0.5, SWHR_MAX = 15,SWHR_Delt = 0.001;

/**Other intermediate quantities during calculation**/
double Tin;
double RHin;
double SWHR;  /*SWHR, g kg-1*/
double din, dout;
double Pin, Pout;
double Tout, TCond;

double Delti;     /*QCond/RD=(iamb-iout)，unit: kJ/kg*/
double SEXC;  /**1/SWP, unit:kJ/g**/
double SEXC_OP, SWHR_OP, TCond_OP;

for (RHin = 0; RHin <= 100 + 1E-3; RHin = RHin + RHin_Delt){

    cout<<RHin<<endl;
```

```
CErelation<<RHin;
Output<<RHin;
Output1<<RHin;
Output2<<RHin;
Output3<<RHin;

for (Tin = Tin_MIN; Tin <= Tin_MAX + 1E-3; Tin = Tin + Tin_Delt){

    Pin = exp(-5800.2206 / (Tin + 273.15) + 1.3914993 - 0.04860239 * (Tin + 273.15) +
0.000041764768 * pow(Tin + 273.15,2) - 0.000000014452093 * pow(Tin + 273.15,3) + 6.5459673 *
log(Tin + 273.15));

    din =  622 * RHin / 100 / (Patm / Pin - RHin / 100);

    SEXC_OP = -10;
    for (SWHR = SWHR_MAX; SWHR >= SWHR_MIN; SWHR = SWHR - SWHR_Delt){

      DeltTD = 4 + 0.2 * SWHR;
      dout = din - SWHR;
      Pout = Patm / (622 / dout + 1);

      if(Pout < 422 || dout < 0){continue;}

      Tout = InterP(PTnum, MPvs, MT, Pout / 1000);
      TCond = Tout - DeltTD;
      Delti = Tin * 1.01 + din / 1000 * (hfg + 1.84 * Tin) - Tout * 1.01 - dout / 1000 * (hfg + 1.84 *
Tout);
      SEXC = (Delti * (Tin - TCond) / (TCond + 273.15)) * 1000 / SWHR;
```

```
            if (SEXC_OP < 0 || SEXC <= SEXC_OP){SEXC_OP = SEXC,SWHR_OP = SWHR,
TCond_OP = TCond;}
        }

        if (SEXC_OP < 0){
            Output<<'\t'<<0.0/0.0;

            Output1<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;
            Output2<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;
            Output3<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;

            CErelation<<'\t'<<Tin<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;
        }
        else {
            Output<<'\t'<<3.6*1000/SEXC_OP;  /*The optimal SWP, unit: kg/kWh*/

            Output1<<'\t'<<din / 1000<<'\t'<<TCond_OP;
            Output2<<'\t'<<din / 1000<<'\t'<<SWHR_OP;
            Output3<<'\t'<<din / 1000<<'\t'<<SEXC_OP;

            CErelation<<'\t'<<Tin<<'\t'<<TCond_OP<<'\t'<<din<<'\t'<<SEXC_OP;
        }
    }
    CErelation<<endl;
    Output<<endl;
    Output1<<endl;
    Output2<<endl;
    Output3<<endl;
}
```

```
    PTrelation.close();
    CErelation.close();
    Output.close();
    Output1.close();
    Output2.close();
    Output3.close();
    return 0;
}

double InterP(int index, double A[], double B[], double a){

    double b;
    int i;
    if (a < A[0]) {b = 0.0 / 0.0;}
    else for(i = 0; i < index; i++){if(A[i] <= a && a <= A[i + 1])break;}
    if(i == index){b = 0.0 / 0.0;}
    else b = (a - A[i]) / (A[i + 1] - A[i]) * (B[i + 1] - B[i]) + B[i];
    return b;
}
```

Code S2 Codes of C++ for analyzing the energy consumption of conventional single SAWH

```
/**Standard Library**/
#include <iostream>
#include <cmath>
#include <fstream>

using namespace std;

/**Overall environment**/
double Patm = 1.01325e5;  /**standard atmospheric pressure**/
double hfg = 2500;        /**Latent heat of water vaporization**/
double InterP(int index, double A[], double B[], double a);

int main(){

    /**Input\Output files**/
    ifstream WRrelation("0-1-Isotherm-Linear.txt");     /**Input table of the linear isothermal of water sorbent**/
    ifstream PTrelation("0-2-SaturatedPressure.txt");   /**Input table of the relationship between temperature and saturation water vapor pressure**/
    ofstream CErelation;
    CErelation.open("1-output.txt");

    ofstream Output;
    Output.open("1-FinalOutput.txt");
    ofstream Output0, Output1, Output2, Output3;
    Output0.open("2-FinalOutput0.txt");
    Output1.open("2-FinalOutput1.txt");
    Output2.open("2-FinalOutput2.txt");
    Output3.open("2-FinalOutput3.txt");
```

```
    int WRnum, PTnum, p;
    WRrelation>>WRnum;
    PTrelation>>PTnum;
    double MWDa[WRnum+1], MRHa[WRnum+1];
    double MWDd[WRnum+1], MRHd[WRnum+1];
    double MT[PTnum+1], MPvs[PTnum+1];
    for(p = 0; p <= WRnum+1; p++){WRrelation>>MRHa[p]>>MWDa[p];if(p <=
WRnum+1)WRrelation>>MRHd[p]>>MWDd[p];}
    for(p = 0; p <= PTnum+1; p++){PTrelation>>MT[p]>>MPvs[p];}

    double Delt_Wcycle = 0.2;
    double rm = 1, hw = 2700, cpw = 1;
    double FlowRatio;    /**RAd/RDe**/

    /**Transfer irreversibility, initial values**/
    double Delt_TAd = 5;
    double Delt_dAd = 1;
    double Delt_TDe = 5;
    double Delt_dDe = 2;
    double Delt_TCond = 5;
    double Eta_AdT = 0.75;
    double Eta_DeT = 0.80;

    /**Define other paramters**/
    double Tin_MIN = 0, Tin_MAX = 50, Tin_Delt = 0.5;
    double SWHR_MIN = 0.5, SWHR_MAX = 25, SWHR_Delt = 0.01;
    double TAd_MIN = -5, TAd_Delt = 1;
    double TCond_MIN = -5, TCond_Delt = 1;
```

```
int WDnum = 100;
double SWHR, RH_Adin;
double T_Adin;
double T_Ad;
double T_Cond;

double W_sat, W_dry, W_Detime;
double RHW_Adend, RHW_Deend, RHW_Deave;
double T_Adout, T_Condout, TW_Adend, TW_De, TW_Decheck, T_De, T_Deout;
double P_Condout, PW_Adend, PW_Deave, PW_Deend, P_Adin;
double d_Adin, d_Condout, dW_Adend, dW_Adave, dW_Deave, d_Deout;
double Delt_iDe, Delt_iAd, Delt_iCond; /*Q/R，unit: kJ/kg*/
double SEXC;  /**1/SWP, unit:kJ/g**/;

double T_Cond_OP, T_Ad_OP, T_De_OP, SEXC_OP, SWHR_OP, FlowRatio_OP;
double T_Cond_OP1, T_Ad_OP1, T_De_OP1, SEXC_OP1, SWHR_OP1, FlowRatio_OP1;
double T_Cond_OP2, T_De_OP2, SEXC_OP2;

for (RH_Adin = 0; RH_Adin <= 100; RH_Adin = RH_Adin + 0.2){

   cout<<RH_Adin<<endl;
   CErelation<<RH_Adin;
   Output<<RH_Adin;
   Output0<<RH_Adin;
   Output1<<RH_Adin;
   Output2<<RH_Adin;
   Output3<<RH_Adin;

   for (T_Adin = Tin_MIN; T_Adin <= Tin_MAX + 1E-3; T_Adin = T_Adin + Tin_Delt){
```

```
/**Adsorption process**/
P_Adin = exp(-5800.2206 / (T_Adin + 273.15) + 1.3914993 - 0.04860239 * (T_Adin + 273.15) + 0.000041764768 * pow(T_Adin + 273.15,2) - 0.000000014452093 * pow(T_Adin + 273.15,3) + 6.5459673 * log(T_Adin + 273.15));    /**Saturated water vapor pressure of inlet air**/
d_Adin =  622 * RH_Adin / 100 / (Patm / P_Adin - RH_Adin / 100);            /**humidity of inlet air**/

/**Find the optimal temperature for cooling the adsorption process**/
SEXC_OP = -10;
for (T_Ad = TAd_MIN; T_Ad <= T_Adin + 1E-3; T_Ad = T_Ad + TAd_Delt){

   SEXC_OP1 = -10;
   for (SWHR = SWHR_MIN; SWHR <= SWHR_MAX + 1E-4; SWHR = SWHR + SWHR_Delt){

      /**Transfer irreversibility**/
      Delt_dDe = 0.4 * SWHR;
      Delt_dAd = 0.2 * SWHR;
      Delt_TDe = 6 + 0.2 * SWHR;
      Delt_TAd = 4 + 0.2 * SWHR;
      Delt_TCond = 4 + 0.2 * SWHR;

      TW_Adend = T_Ad + Delt_TAd;  /**The end temperature of water sorbent**/
      dW_Adend = d_Adin - Delt_dAd;      /**The end humidity of water sorbent**/
      PW_Adend = exp(-5800.2206 / (TW_Adend + 273.15) + 1.3914993 - 0.04860239 * (TW_Adend + 273.15) + 0.000041764768 * pow(TW_Adend + 273.15,2) - 0.000000014452093 * pow(TW_Adend + 273.15,3) + 6.5459673 * log(TW_Adend + 273.15));     /**The end saturated water vapor pressure of water sorbent**/
```

RHW_Adend = Patm * 100 / (622 * PW_Adend / dW_Adend + PW_Adend); /**The end relative humidity of water sorbent**/

if (RHW_Adend <= 0) {continue;}

W_sat = InterP(WRnum, MRHa, MWDa, RHW_Adend); /**The end water content of water sorbent**/

/**Desorption process**/

W_dry = W_sat - Delt_Wcycle; /**The end water content of water sorbent**/

if (W_dry <= 0) {continue;}

RHW_Deend = InterP(WRnum, MWDd, MRHd, W_dry); /**The end relative humidity of water sorbent**/

RHW_Deave = 0, W_Detime = W_dry;

for (p = 0; p < WDnum; p++){RHW_Deave = InterP(WRnum, MWDd, MRHd, W_Detime) / WDnum + RHW_Deave, W_Detime = W_Detime + Delt_Wcycle / WDnum;} /**The average relative humidity of water sorbent**/

dW_Adave = 622 * RHW_Deave / (Patm / PW_Adend - RHW_Deave / 100) / 100;

FlowRatio = max((SWHR + Delt_dDe) / (d_Adin - dW_Adave),2.0);

Eta_AdT = 0.8 / pow(FlowRatio,0.2);

T_Adout = (T_Adin <= TW_Adend) ? T_Adin : Eta_AdT * TW_Adend + (1 - Eta_AdT) * T_Adin;

/**Condensation process**/

SEXC_OP2 = -10;

for (T_Cond = TCond_MIN; T_Cond <= T_Adin + 1E-3; T_Cond = T_Cond + TCond_Delt){

T_Condout = T_Cond + Delt_TCond; /**The temperature of outlet air from condenser**/

P_Condout = exp(-5800.2206 / (T_Condout + 273.15) + 1.3914993 - 0.048640239 * (T_Condout + 273.15) + 0.000041764768 * pow(T_Condout + 273.15,2) - 0.000000014452093 *

```
pow(T_Condout + 273.15,3) + 6.5459673 * log(T_Condout + 273.15));   /**The saturated water vapor pressure of outlet air from condenser**/
        d_Condout =  622 / (Patm / P_Condout - 1);      /**The humidity of outlet air from condenser**/

        /**Desorption process**/
        d_Deout = d_Condout + SWHR;  /**The average humidity of outlet air from desorption bed**/
        dW_Deave = d_Deout + Delt_dDe; /**The average humidity of desorbed water sorbent**/
        PW_Deave = Patm * 100 / (622 * RHW_Deave / dW_Deave + RHW_Deave);     /**The average saturated water vapor pressure of desorbed water sorbent**/
        TW_De = InterP(PTnum, MPvs, MT, PW_Deave / 1000);   /**The temperature of desorbed water sorbent**/

        PW_Deend = Patm * 100 / (622 * RHW_Deend / d_Condout + RHW_Deend);    /**The end saturated water vapor pressure of desorbed water sorbent**/
        TW_Decheck = InterP(PTnum, MPvs, MT, PW_Deend / 1000);              /**The end temperature of desorbed water sorbent**/

        T_De = max(TW_De, TW_Decheck) + Delt_TDe; /**The temperature of heat sources for desorption**/
        if (T_De >= 150) {continue;}
        T_Deout = (T_De - Delt_TDe >= T_Condout) ? Eta_DeT * (T_De - Delt_TDe) + (1 - Eta_DeT) * T_Condout : T_Condout;  /**The temperature of outlet air from desorptionb bed**/
        /**Calculating the heat flux and SEXC of the adsorption process, desorption process, and condensation process**/
        Delt_iDe = T_Deout * 1.01 + d_Deout / 1000 * (hw + 1.84 * T_Deout) - T_Condout * 1.01 - d_Condout / 1000 * (hw + 1.84 * T_Condout) + SWHR / Delt_Wcycle / 1000 * (TW_De - TW_Adend) * cpw + SWHR / Delt_Wcycle / 1000 * (T_De - T_Ad) * cpw * rm;
        Delt_iAd = FlowRatio * (T_Adin * 1.01 + d_Adin / 1000 * (hw + 1.84 * T_Adin) - T_Adout * 1.01 - (d_Adin - SWHR / FlowRatio) / 1000 * (hw + 1.84 * T_Adout)) + SWHR /
```

```
Delt_Wcycle / 1000 * (TW_De - TW_Adend) * cpw + SWHR / Delt_Wcycle / 1000 * (T_De - T_Ad) * cpw * rm;
                Delt_iCond = T_Deout * 1.01 + d_Deout / 1000 * (hfg + 1.84 * T_Deout) - T_Condout * 1.01 - d_Condout / 1000 * (hfg + 1.84 * T_Condout);
                SEXC = (Delt_iDe * (T_De - T_Adin) / (T_De + 273.15) + Delt_iAd * (T_Adin - T_Ad) / (T_Ad + 273.15) + Delt_iCond * (T_Adin - T_Cond) / (T_Cond + 273.15)) * 1000 / SWHR;

                if (SEXC_OP2 < 0 || SEXC <= SEXC_OP2){SEXC_OP2 = SEXC, T_Cond_OP2 = T_Cond, T_De_OP2 = T_De;}
            }
            if (SEXC_OP2 > 0 && (SEXC_OP1 < 0 || SEXC_OP2 <= SEXC_OP1)){SEXC_OP1 = SEXC_OP2, T_Cond_OP1 = T_Cond_OP2, T_De_OP1 = T_De_OP2, SWHR_OP1 = SWHR; FlowRatio_OP1 = FlowRatio;}
        }
        if (SEXC_OP1 > 0 && (SEXC_OP < 0 || SEXC_OP1 <= SEXC_OP)){SEXC_OP = SEXC_OP1, T_Cond_OP = T_Cond_OP1, T_De_OP = T_De_OP1, SWHR_OP = SWHR_OP1, FlowRatio_OP = FlowRatio_OP1, SEXC_OP = SEXC_OP1, T_Ad_OP = T_Ad;}
    }

    if (SEXC_OP < 0){
        Output<<'\t'<<0.0/0.0;
        Output0<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;
        Output1<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;
        Output2<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;
        Output3<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;

CErelation<<'\t'<<T_Adin<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;
    }
    else {
        Output<<'\t'<<1/(SEXC_OP/1000/3.6); /**SWP unit: kg kWh-1**/
```

```
            Output0<<'\t'<<d_Adin / 1000<<'\t'<<SWHR_OP;
            Output1<<'\t'<<d_Adin / 1000<<'\t'<<T_Ad_OP;
            Output2<<'\t'<<d_Adin / 1000<<'\t'<<T_Cond_OP;
            Output3<<'\t'<<d_Adin / 1000<<'\t'<<T_De_OP;

CErelation<<'\t'<<T_Adin<<'\t'<<T_Ad_OP<<'\t'<<T_Cond_OP<<'\t'<<T_De_OP<<'\t'<<SWHR_OP<<'
\t'<<SEXC_OP<<'\t'<<FlowRatio;
          }
       }
       CErelation<<endl;
       Output<<endl;
       Output0<<endl;
       Output1<<endl;
       Output2<<endl;
       Output3<<endl;
    }

    WRrelation.close();
    PTrelation.close();
    CErelation.close();
    Output.close();
    Output0.close();
    Output1.close();
    Output2.close();
    Output3.close();
    return 0;
}

double InterP(int index, double A[], double B[], double a){
```

```
    double b;
    int i;
    if (a < A[0]) {b = 0.0 / 0.0;}
    else for(i = 0; i < index; i++){if(A[i] <= a && a <= A[i + 1])break;}
    if(i == index){b = B[index];}
    else b = (a - A[i]) / (A[i + 1] - A[i]) * (B[i + 1] - B[i]) + B[i];
    return b;
}
```

Code S3 Codes of Python for assessing year-round operation time, average annual SWP, and technoeconomic feasibility of AWH systems

```
import os

import os

import time

from concurrent.futures import ThreadPoolExecutor, as_completed, wait

import numpy

import pandas

import netCDF4

import threading

import matplotlib.pyplot as plt

from mpl_toolkits.basemap import Basemap


Psat_csv  = pandas.read_csv("Psat-T.csv")

Psat_form = Psat_csv.values.flatten()


SWP_sawh_csv  = pandas.read_csv("SAWH opt kg kWh-1.csv")

SWP_dawh_csv  = pandas.read_csv("DAWH opt kg kWh-1.csv")

SWP_sawh_wohp_csv = pandas.read_csv("SAWH woHP opt kg kWh-1.csv")

SWP_sawh_form = SWP_sawh_csv.values

SWP_dawh_form = SWP_dawh_csv.values

SWP_sawh_wohp_form = SWP_sawh_wohp_csv.values


SWHR_sawh_csv  = pandas.read_csv("SAWH opt g kgair-1.csv")

SWHR_dawh_csv  = pandas.read_csv("DAWH opt g kgair-1.csv")

SWHR_sawh_wohp_csv  = pandas.read_csv("SAWH woHP opt g kgair-1.csv")
```

```
SWHR_sawh_form = SWHR_sawh_csv.values

SWHR_dawh_form = SWHR_dawh_csv.values

SWHR_sawh_wohp_form = SWHR_sawh_wohp_csv.values


Result_1 =
[numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440])]

Result_2 =
[numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440])]

Result_3 =
[numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440])]

Result_4 =
[numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440])]

List_Result = [Result_1, Result_2, Result_3, Result_4]


class MultiThread (threading.Thread):
    def __init__(self, func, threadID, file):
        self.func     = func
        self.threadID = threadID
        self.file     = file
        self.result   = None
        threading.Thread.__init__(self)


    def run(self):
        print("***************" + str(self.threadID) + "Start******************")
        List_Result[self.threadID-1] = self.func(self.file)
        print("***************" + str(self.threadID) + "End******************")
```

```
def Read_NC4(file_name):

    data  = netCDF4.Dataset(file_name)
    T     = data.variables["Tair_f_inst"][:]
    Y     = data.variables["Qair_f_inst"][:]
    data.close()
    return T.data[0], Y.data[0]

def Y2RH(T, Y):

    rows, cols = T.shape
    P = numpy.zeros([rows, cols])

    for r in range(rows):
        for c in range(cols):
            P[r][c] = Psat_form[int(T[r][c]-273.15)] \
                if 323.15 > T[r][c] > 273.15 else -9999


    RH = 101325000 * numpy.multiply\
        ((1/P) ,(1/(1000+622/Y)))
    RH[RH>1] = 1

    return RH
```

```
def Calculate_SWP_SWHR (T, RH, Time_cal, Time_ha, Time_sa, Time_da, SWHR_sawh_max,
SWHR_dawh_max, SWHR_hawh_max, SWHR_sawh_wohp_max):
    rows, cols = T.shape
    SWP_sawh = numpy.zeros([rows, cols])
    SWP_dawh = numpy.zeros([rows, cols])
    SWP_hawh = numpy.zeros([rows, cols])
    SWP_sawh_wohp = numpy.zeros([rows, cols])

    SWHR_sawh = numpy.zeros([rows, cols])
    SWHR_dawh = numpy.zeros([rows, cols])
    SWHR_hawh = numpy.zeros([rows, cols])
    SWHR_sawh_wohp = numpy.zeros([rows, cols])

    for r in range(rows):
        for c in range(cols):
            rh    =       int(RH[r][c]*500)/5
            t     = int((T[r][c]-273.15)*2)/2
            SWP_sa = SWP_sawh_form[int(rh*5)][int(t*2)] if 50 >= t >= 0 else 0
            SWP_da = SWP_dawh_form[int(rh*5)][int(t*2)] if 50 >= t >= 0 else 0
            SWP_sa_wohp =  SWP_sawh_wohp_form[int(rh*5)][int(t*2)] if 50 >= t >= 0 else 0

            SWHR_sa = SWHR_sawh_form[int(rh*5)][int(t*2)] if 50 >= t >= 0 else 0
            SWHR_da = SWHR_dawh_form[int(rh*5)][int(t*2)] if 50 >= t >= 0 else 0
            SWHR_sa_wohp = SWHR_sawh_wohp_form[int(rh*5)][int(t*2)] if 50 >= t >= 0 else 0

            SWP_sawh[r][c], SWP_dawh[r][c], SWP_sawh_wohp[r][c] = SWP_sa, SWP_da, SWP_sa_wohp
```

```
SWP_hawh[r][c] = max(SWP_sa, SWP_da)
Time_cal[r][c] += 1

SWHR_sawh[r][c], SWHR_dawh[r][c], SWHR_sawh_wohp[r][c] = SWHR_sa, SWHR_da, SWHR_sa_wohp
if SWP_sa > SWP_da:
    SWHR_hawh[r][c] = SWHR_sawh[r][c]
    if 2700 * SWHR_sawh[r][c] > SWHR_hawh_max[r][c]:
        SWHR_hawh_max[r][c] = 2700 * SWHR_sawh[r][c]
else:
    SWHR_hawh[r][c] = SWHR_dawh[r][c]
    if 2500 * SWHR_dawh[r][c] > SWHR_hawh_max[r][c]:
        SWHR_hawh_max[r][c] = 2500 * SWHR_dawh[r][c]

if SWHR_sawh[r][c] > SWHR_sawh_max[r][c]:
    SWHR_sawh_max[r][c] = SWHR_sawh[r][c]

if SWHR_dawh[r][c] > SWHR_dawh_max[r][c]:
    SWHR_dawh_max[r][c] = SWHR_dawh[r][c]

# SWHR_hawh_max[r][c] = max (2700 * SWHR_sawh_max[r][c], 2500 * SWHR_dawh_max[r][c])

if SWHR_sawh_wohp[r][c] > SWHR_sawh_wohp_max[r][c]:
    SWHR_sawh_wohp_max[r][c] = SWHR_sawh_wohp[r][c]
```

```
            # if ex_sa * ex_da != 0:
            #     Ex_hawh[r][c] = min(ex_sa, ex_da)
            #     Time_cal[r][c] += 1
            #     Ex_sawh[r][c], Ex_dawh[r][c] = ex_sa, ex_da
            if SWP_da + SWP_sa != 0:
                Time_ha[r][c] += 1
            if SWP_sa != 0:
                Time_sa[r][c] += 1
            if SWP_da != 0:
                Time_da[r][c] += 1

    return SWP_sawh, SWP_dawh, SWP_hawh, SWP_sawh_wohp, SWHR_sawh, SWHR_dawh,
SWHR_hawh, SWHR_sawh_wohp, SWHR_sawh_max, SWHR_dawh_max, SWHR_hawh_max,
SWHR_sawh_wohp_max


def Calculate_Main(list_file, k):

    print("***************"+str(k)+"Start***************")

    # Initialize
    SWP_sawh     = numpy.zeros([600, 1440])
    SWP_dawh     = numpy.zeros([600, 1440])
    SWP_hawh     = numpy.zeros([600, 1440])
    SWP_sawh_wohp     = numpy.zeros([600, 1440])

    SWHR_sawh     = numpy.zeros([600, 1440])
```

```
SWHR_dawh     = numpy.zeros([600, 1440])
SWHR_hawh     = numpy.zeros([600, 1440])
SWHR_sawh_wohp     = numpy.zeros([600, 1440])

SWHR_sawh_max     = numpy.zeros([600, 1440])
SWHR_dawh_max     = numpy.zeros([600, 1440])
SWHR_hawh_max     = numpy.zeros([600, 1440])
SWHR_sawh_wohp_max     = numpy.zeros([600, 1440])

vi_time_cal = numpy.zeros([600, 1440])
vi_time_ha  = numpy.zeros([600, 1440])
vi_time_sa  = numpy.zeros([600, 1440])
vi_time_da  = numpy.zeros([600, 1440])

time_start = time.time()
n = 0
na = ""
for file_name in list_file:
    try:
        if file_name[22 : -21] != na:
            na = file_name[22:-21]
        print("course"+str(k)+"settle documents: "+str(na))
        T_air, Y_air = Read_NC4(file_path + "\\" + file_name)
        RH_air       = Y2RH(T_air, Y_air)
    except:
        print(file_name + " error !")
```

```
            continue

        SWP_sawh_temp, SWP_dawh_temp, SWP_hawh_temp, SWP_sawh_wohp_temp,
SWHR_sawh_temp, SWHR_dawh_temp, SWHR_hawh_temp, SWHR_sawh_wohp_temp,
SWHR_sawh_max, SWHR_dawh_max, SWHR_hawh_max, SWHR_sawh_wohp_max =
Calculate_SWP_SWHR\
            (T_air, RH_air, vi_time_cal, vi_time_ha, vi_time_sa, vi_time_da, SWHR_sawh_max,
SWHR_dawh_max, SWHR_hawh_max, SWHR_sawh_wohp_max)

        SWP_sawh += SWP_sawh_temp
        SWP_dawh += SWP_dawh_temp
        SWP_hawh += SWP_hawh_temp
        SWP_sawh_wohp += SWP_sawh_wohp_temp

        SWHR_sawh += SWHR_sawh_temp
        SWHR_dawh += SWHR_dawh_temp
        SWHR_hawh += SWHR_hawh_temp
        SWHR_sawh_wohp += SWHR_sawh_wohp_temp

        n += 1
    time_end = time.time()
    print("***************"+str(k)+"end"+str(n)+"files, time comsumption" + str(time_end-time_start) +
"*****************")
    return SWP_sawh, SWP_dawh, SWP_hawh, SWP_sawh_wohp, SWHR_sawh, SWHR_dawh,
SWHR_hawh, SWHR_sawh_wohp, SWHR_sawh_max, SWHR_dawh_max, SWHR_hawh_max,
SWHR_sawh_wohp_max, vi_time_cal, vi_time_ha, vi_time_sa, vi_time_da
if __name__ == "__main__":
```

```
SWP_sawh  = numpy.zeros([600, 1440])
SWP_dawh  = numpy.zeros([600, 1440])
SWP_hawh  = numpy.zeros([600, 1440])
SWP_sawh_wohp  = numpy.zeros([600, 1440])

SWHR_sawh    = numpy.zeros([600, 1440])
SWHR_dawh    = numpy.zeros([600, 1440])
SWHR_hawh    = numpy.zeros([600, 1440])
SWHR_sawh_wohp  = numpy.zeros([600, 1440])

SWHR_sawh_max     = numpy.zeros([600, 1440])
SWHR_dawh_max     = numpy.zeros([600, 1440])
SWHR_hawh_max     = numpy.zeros([600, 1440])
SWHR_sawh_wohp_max  = numpy.zeros([600, 1440])

Vil_time_cal = numpy.zeros([600, 1440])
Vil_time_ha  = numpy.zeros([600, 1440])
Vil_time_sa  = numpy.zeros([600, 1440])
Vil_time_da  = numpy.zeros([600, 1440])

file_path = r"Global data T and Y 20220101-20221231"
# file_path = r"test"
list_file_name = os.listdir(file_path)

n = int(len(list_file_name)/12)
```

```
    all_future = []
    with ThreadPoolExecutor(max_workers=12) as pool:
        for i in range(11):
            all_future.append(pool.submit(Calculate_Main, list_file_name[i*n:(i+1)*n], i+1))
        all_future.append(pool.submit(Calculate_Main, list_file_name[11*n:], 12))

        for future in as_completed(all_future):
            tmp_SWP_sawh, tmp_SWP_dawh, tmp_SWP_hawh, tmp_SWP_sawh_wohp, tmp_SWHR_sawh, tmp_SWHR_dawh, tmp_SWHR_hawh, tmp_SWHR_sawh_wohp, SWHR_sawh_max, SWHR_dawh_max, SWHR_hawh_max, SWHR_sawh_wohp_max, tmp_time_cal, tmp_time_ha, tmp_time_sa, tmp_time_da = future.result()

            SWP_sawh  += tmp_SWP_sawh
            SWP_dawh  += tmp_SWP_dawh
            SWP_hawh  += tmp_SWP_hawh
            SWP_sawh_wohp  += tmp_SWP_sawh_wohp

            SWHR_sawh  += tmp_SWHR_sawh
            SWHR_dawh  += tmp_SWHR_dawh
            SWHR_hawh  += tmp_SWHR_hawh
            SWHR_sawh_wohp  += tmp_SWHR_sawh_wohp

            Vil_time_cal += tmp_time_cal
            Vil_time_ha  += tmp_time_ha
            Vil_time_sa  += tmp_time_sa
            Vil_time_da  += tmp_time_da
```

```
print("finished")

# Vil_time_cal[Vil_time_cal == 0] = -1

# Average annual SWP
SWP_sawh = numpy.multiply(SWP_sawh, 1/Vil_time_cal)
SWP_dawh = numpy.multiply(SWP_dawh, 1/Vil_time_cal)
SWP_hawh = numpy.multiply(SWP_hawh, 1/Vil_time_cal)
SWP_sawh_wohp = numpy.multiply(SWP_sawh_wohp, 1/Vil_time_cal)

SWP_sawh[SWP_sawh == 0] = numpy.nan
SWP_dawh[SWP_dawh == 0] = numpy.nan
SWP_hawh[SWP_hawh == 0] = numpy.nan
SWP_sawh_wohp[SWP_sawh_wohp == 0] = numpy.nan

# ΔSWP
SWP_delt_sawh = SWP_hawh - SWP_sawh
SWP_delt_dawh = SWP_hawh - SWP_dawh
SWP_delt_sawh_wohp = SWP_hawh - SWP_sawh_wohp

SWP_delt_sawh[SWP_delt_sawh == 0] = numpy.nan
SWP_delt_dawh[SWP_delt_dawh == 0] = numpy.nan
SWP_delt_sawh_wohp[SWP_delt_sawh_wohp == 0] = numpy.nan
```

```
SWHR_sawh[SWHR_sawh == 0] = numpy.nan

SWHR_dawh[SWHR_dawh == 0] = numpy.nan

SWHR_hawh[SWHR_hawh == 0] = numpy.nan

SWHR_sawh_wohp[SWHR_sawh_wohp == 0] = numpy.nan


SWHR_sawh_max[SWHR_sawh_max == 0] = numpy.nan

SWHR_dawh_max[SWHR_dawh_max == 0] = numpy.nan

SWHR_hawh_max[SWHR_hawh_max == 0] = numpy.nan

SWHR_sawh_wohp_max[SWHR_sawh_wohp_max == 0] = numpy.nan


# PPV

PPV_sawh = 0.34/SWP_sawh

PPV_dawh = 0.34/SWP_dawh

PPV_hawh = 0.34/SWP_hawh

PPV_sawh_wohp = 0.34/SWP_sawh_wohp


#PHP

PHP_sawh = 2700 * 1800/15/3/3600/Vil_time_sa

PHP_dawh = 2400 * 1800/15/3/3600/Vil_time_da

PHP_hawh = 2700 * 1800/15/3/3600/Vil_time_ha

Delt_PHP_dawh = PHP_dawh - PHP_hawh


#Psorb

Psorb_sawh = 574.76 /2.865/1/Vil_time_sa

Psorb_hawh = 574.76 /2.865/1/Vil_time_ha

Psorb_sawh_wohp = 574.76 /2.865/1/Vil_time_sa_wohp
```

```
# Psorb_sawh = 0.75 /1/1/Vil_time_sa

# Psorb_hawh = 0.75 /1/1/Vil_time_ha

# Psorb_sawh_wohp = 0.75 /1/1/Vil_time_sa_wohp


# Pw

Pw_sawh = PPV_sawh + PHP_sawh + Psorb_sawh

Pw_dawh = PPV_dawh + PHP_dawh

Pw_hawh = PPV_hawh + PHP_hawh + Psorb_hawh

Pw_sawh_wohp = PPV_sawh_wohp + Psorb_sawh_wohp


# ΔPw

Pw_delt_sawh = Pw_sawh - Pw_hawh

Pw_delt_dawh = Pw_dawh - Pw_hawh

Pw_delt_sawh_wohp = Pw_sawh_wohp - Pw_hawh


Pw_delt_sawh[Pw_delt_sawh == 0] = numpy.nan

Pw_delt_dawh[Pw_delt_dawh == 0] = numpy.nan

Pw_delt_sawh_wohp[Pw_delt_sawh_wohp == 0] = numpy.nan


# Vaild time

Vil_delt_time_sawh= Vil_time_ha - Vil_time_sa

Vil_delt_time_dawh= Vil_time_ha - Vil_time_da


Vil_delt_time_sawh[Vil_time_ha == 0]=numpy.nan

Vil_delt_time_dawh[Vil_time_ha == 0]=numpy.nan
```

```
Vil_hour_ha = 3 * Vil_time_ha

Vil_hour_sa = 3 * Vil_time_sa

Vil_hour_da = 3 * Vil_time_da


Vil_day_ha = Vil_time_ha / 8

Vil_day_sa = Vil_time_sa / 8

Vil_day_da = Vil_time_da / 8


Vil_delt_hour_sawh= Vil_hour_ha - Vil_hour_sa

Vil_delt_hour_dawh= Vil_hour_ha - Vil_hour_da


Vil_delt_hour_sawh[Vil_hour_ha == 0]=numpy.nan

Vil_delt_hour_dawh[Vil_hour_ha == 0]=numpy.nan


Vil_delt_day_sawh= Vil_day_ha - Vil_day_sa

Vil_delt_day_dawh= Vil_day_ha - Vil_day_da


Vil_delt_day_sawh[Vil_day_ha == 0]=numpy.nan

Vil_delt_day_dawh[Vil_day_ha == 0]=numpy.nan


Vil_time_ha[Vil_time_ha == 0]   =numpy.nan

Vil_time_sa[Vil_time_sa == 0]   =numpy.nan

Vil_time_da[Vil_time_da == 0]   =numpy.nan


Vil_hour_ha[Vil_hour_ha == 0]   =numpy.nan

Vil_hour_sa[Vil_hour_sa == 0]   =numpy.nan
```

```
    Vil_hour_da[Vil_hour_da == 0]   =numpy.nan

    Vil_day_ha[Vil_day_ha == 0]   =numpy.nan
    Vil_day_sa[Vil_day_sa == 0]   =numpy.nan
    Vil_day_da[Vil_day_da == 0]   =numpy.nan

# Save results

    # SWP
    numpy.savetxt("Result\\SWP_sawh_opt_kg kWh-1.csv", SWP_sawh, delimiter=",")
    numpy.savetxt("Result\\SWP_dawh_opt_kg kWh-1.csv", SWP_dawh, delimiter=",")
    numpy.savetxt("Result\\SWP_hawh_opt_kg kWh-1.csv", SWP_hawh, delimiter=",")
    numpy.savetxt("Result\\SWP_sawh_wohp_opt_kg kWh-1.csv", SWP_sawh_wohp, delimiter=",")
    numpy.savetxt("Result\\SWP_delt_sawh_opt_kg kWh-1.csv", SWP_delt_sawh, delimiter=",")
    numpy.savetxt("Result\\SWP_delt_dawh_opt_kg kWh-1.csv", SWP_delt_dawh, delimiter=",")
    numpy.savetxt("Result\\SWP_delt_sawh_wohp_opt_kg kWh-1.csv", SWP_delt_sawh_wohp,
delimiter=",")

    # SWHR
    numpy.savetxt("Result\\SWHR_sawh_opt_kg kWh-1.csv", SWHR_sawh, delimiter=",")
    numpy.savetxt("Result\\SWHR_dawh_opt_kg kWh-1.csv", SWHR_dawh, delimiter=",")
    numpy.savetxt("Result\\SWHR_hawh_opt_kg kWh-1.csv", SWHR_hawh, delimiter=",")
    numpy.savetxt("Result\\SWHR_sawh_wohp_opt_kg kWh-1.csv", SWHR_sawh_wohp, delimiter=",")

    numpy.savetxt("Result\\SWHR_sawh_max_opt_kg kWh-1.csv", SWHR_sawh_max, delimiter=",")
    numpy.savetxt("Result\\SWHR_dawh_max_opt_kg kWh-1.csv", SWHR_dawh_max, delimiter=",")
```

```
    numpy.savetxt("Result\\SWHR_hawh_max_opt_kg kWh-1.csv", SWHR_hawh_max, delimiter=",")
    numpy.savetxt("Result\\SWHR_sawh_wohp_max_opt_kg kWh-1.csv", SWHR_sawh_wohp_max,
delimiter=",")

    # Pw
    numpy.savetxt("Result\\Pw_sawh_CNY L-1.csv", Pw_sawh, delimiter=",")
    numpy.savetxt("Result\\Pw_dawh_CNY L-1.csv", Pw_dawh, delimiter=",")
    numpy.savetxt("Result\\Pw_hawh_CNY L-1.csv", Pw_hawh, delimiter=",")
    numpy.savetxt("Result\\Pw_sawh_wohp_CNY L-1.csv", Pw_sawh_wohp, delimiter=",")
    numpy.savetxt("Result\\Pw_delt_sawh_CNY L-1.csv", Pw_delt_sawh, delimiter=",")
    numpy.savetxt("Result\\Pw_delt_dawh_CNY L-1.csv", Pw_delt_dawh, delimiter=",")
    numpy.savetxt("Result\\Pw_delt_sawh_wohp_CNY L-1.csv", Pw_delt_sawh_wohp, delimiter=",")

    # PPV
    numpy.savetxt("Result\\PPV_sawh_CNY L-1.csv", PPV_sawh, delimiter=",")
    numpy.savetxt("Result\\PPV_dawh_CNY L-1.csv", PPV_dawh, delimiter=",")
    numpy.savetxt("Result\\PPV_hawh_CNY L-1.csv", PPV_hawh, delimiter=",")
    numpy.savetxt("Result\\PPV_sawh_wohp_CNY L-1.csv", PPV_sawh_wohp, delimiter=",")

    # PHP
    numpy.savetxt("Result\\PHP_sawh_CNY L-1.csv", PHP_sawh, delimiter=",")
    numpy.savetxt("Result\\PHP_dawh_CNY L-1.csv", PHP_dawh, delimiter=",")
    numpy.savetxt("Result\\PHP_hawh_CNY L-1.csv", PHP_hawh, delimiter=",")

    # Psorb
    numpy.savetxt("Result\\Psorb_sawh_CNY L-1.csv", Psorb_sawh, delimiter=",")
```

```
numpy.savetxt("Result\\Psorb_hawh_CNY L-1.csv", Psorb_hawh, delimiter=",")
numpy.savetxt("Result\\Psorb_sawh_wohp_CNY L-1.csv", Psorb_sawh_wohp, delimiter=",")



# Valid time
numpy.savetxt("Result\\Vil_time_hawh_opt_kg kWh-1.csv", Vil_time_ha, delimiter=",")
numpy.savetxt("Result\\Vil_time_sawh_opt_kg kWh-1.csv", Vil_time_sa, delimiter=",")
numpy.savetxt("Result\\Vil_time_dawh_opt_kg kWh-1.csv", Vil_time_da, delimiter=",")
numpy.savetxt("Result\\Vil_delt_time_sawh_opt_kg kWh-1.csv", Vil_delt_time_sawh, delimiter=",")
numpy.savetxt("Result\\Vil_delt_time_dawh_opt_kg kWh-1.csv", Vil_delt_time_dawh, delimiter=",")
numpy.savetxt("Result\\Vil_hour_hawh_opt_kg kWh-1.csv", Vil_hour_ha, delimiter=",")
numpy.savetxt("Result\\Vil_hour_sawh_opt_kg kWh-1.csv", Vil_hour_sa, delimiter=",")
numpy.savetxt("Result\\Vil_hour_dawh_opt_kg kWh-1.csv", Vil_hour_da, delimiter=",")
numpy.savetxt("Result\\Vil_delt_hour_sawh_opt_kg kWh-1.csv", Vil_delt_hour_sawh, delimiter=",")
numpy.savetxt("Result\\Vil_delt_hour_dawh_opt_kg kWh-1.csv", Vil_delt_hour_dawh, delimiter=",")
numpy.savetxt("Result\\Vil_day_hawh_opt_kg kWh-1.csv", Vil_day_ha, delimiter=",")
numpy.savetxt("Result\\Vil_day_sawh_opt_kg kWh-1.csv", Vil_day_sa, delimiter=",")
numpy.savetxt("Result\\Vil_day_dawh_opt_kg kWh-1.csv", Vil_day_da, delimiter=",")
numpy.savetxt("Result\\Vil_day_hour_sawh_opt_kg kWh-1.csv", Vil_delt_day_sawh, delimiter=",")
numpy.savetxt("Result\\Vil_day_hour_dawh_opt_kg kWh-1.csv", Vil_delt_day_dawh, delimiter=",")
```

Code S4 Codes of C++for analyzing the energy consumption of solar-thermal-driven SAWH

```
/**Standard Library**/
#include <iostream>
#include <cmath>
#include <fstream>

using namespace std;

/**Overall environment**/
double Patm = 1.01325e5;
double hfg = 2500;
double InterP(int index, double A[], double B[], double a);

int main(){

    /**Input/output files**/
    ifstream WRrelation("0-1-Isotherm-Linear.txt");                 /**Input table of the linear isothermal of water sorbent**/
    ifstream PTrelation("0-2-SaturatedPressure.txt");               /**Input table of the relationship between temperature and saturation water vapor pressure**/
    ofstream CErelation;
    CErelation.open("1-output.txt");

    ofstream Output;
    Output.open("1-FinalOutput.txt");
    ofstream Output0, Output1, Output2, Output3;
```

```
    Output0.open("2-FinalOutput0.txt");
    Output1.open("2-FinalOutput1.txt");
    Output2.open("2-FinalOutput2.txt");
    Output3.open("2-FinalOutput3.txt");

    int WRnum, PTnum, p;
    WRrelation>>WRnum;
    PTrelation>>PTnum;
    double MWDa[WRnum+1], MRHa[WRnum+1];
    double MWDd[WRnum+1], MRHd[WRnum+1];
    double MT[PTnum+1], MPvs[PTnum+1];
    for(p = 0; p <= WRnum+1; p++){WRrelation>>MRHa[p]>>MWDa[p];if(p <=
WRnum+1)WRrelation>>MRHd[p]>>MWDd[p];}
    for(p = 0; p <= PTnum+1; p++){PTrelation>>MT[p]>>MPvs[p];}

    double Delt_Wcycle = 0.2;
    double hw = 2700, cpw = 1;
    double FlowRatio;      /*RAD/RDe*/

    /**Transfer irreversibility, initial values**/

    double Delt_dAd = 1;
    double Delt_TDe = 5;
    double Delt_dDe = 2;
    double Delt_TCond = 5;
    double Eta_AdT = 0.75;
```

```
double Eta_DeT = 0.80;

/**Define other parameters**/
double Tin_MIN = 0, Tin_MAX = 50, Tin_Delt = 0.5;
double SWHR_MIN = 0.5, SWHR_MAX = 25, SWHR_Delt = 0.01;
int WDnum = 100;
double SWHR, RH_ADin;
double T_ADin;
double T_AD;
double T_Cond;

double W_sat, W_dry, W_Detemp;
double RHW_Adend, RHW_Deend, RHW_Deave;
double T_Adout, T_Condout, TW_Adend, TW_De, TW_Decheck, T_De, T_Deout;
double P_Condout, PW_Adend, PW_Deave, PW_Deend, P_Adin;
double d_Adin, d_Condout, dW_Adend, dW_Adave, dW_Deave, dW_Deout;
double Q_SolarS;
double SEXC;

double T_Cond_OP, T_AD_OP, T_De_OP, SEXC_OP, SWHR_OP, FlowRatio_OP;
double T_Cond_OP1, T_AD_OP1, T_De_OP1, SEXC_OP1, SWHR_OP1, FlowRatio_OP1;
double T_Cond_OP2, T_De_OP2, SEXC_OP2;

for (RH_ADin = 0; RH_ADin <= 100 + 1E-3; RH_ADin = RH_ADin + 0.2)
{
```

```
cout<<RH_ADin<<endl;
CErelation<<RH_ADin;
Output<<RH_ADin;
Output0<<RH_ADin;
Output1<<RH_ADin;
Output2<<RH_ADin;
Output3<<RH_ADin;

for (T_ADin = Tin_MIN; T_ADin <= Tin_MAX + 1E-3; T_ADin = T_ADin + Tin_Delt){

    /**Adsorption process**/
    P_Adin = exp(-5800.2206 / (T_ADin + 273.15) + 1.3914993 - 0.04860239 * (T_ADin + 273.15) + 0.000041764768 * pow(T_ADin + 273.15,2) - 0.000000014452093 * pow(T_ADin + 273.15,3) + 6.5459673 * log(T_ADin + 273.15));
                                                    /**Saturated water vapor pressure of inlet air**/
    d_Adin =  622 * RH_ADin / 100 / (Patm / P_Adin - RH_ADin / 100);        /**Humidity of inlet air**/

      SEXC_OP = -10;
      for (SWHR = SWHR_MIN; SWHR <= SWHR_MAX + 1E-4; SWHR = SWHR + SWHR_Delt){

        /**Transfer irreversibility*/
        Delt_dDe = 0.4 * SWHR;
        Delt_dAd = 0.2 * SWHR;
        Delt_TDe = 6 + 0.2 * SWHR;
```

```
Delt_TCond = 4 + 0.2 * SWHR;


TW_Adend = T_ADin;                /*The end temperature of water sorbent*/
dW_Adend = d_Adin - Delt_dAd;     /**The end temperature of water sorbent**/
PW_Adend = exp(-5800.2206 / (TW_Adend + 273.15) + 1.3914993 - 0.04860239 *
(TW_Adend + 273.15) + 0.000041764768 * pow(TW_Adend + 273.15,2) - 0.000000014452093 *
pow(TW_Adend + 273.15,3) + 6.5459673 * log(TW_Adend + 273.15));
                                  /**The end saturated water vapor pressure of
water sorbent**/
RHW_Adend = Patm * 100 / (622 * PW_Adend / dW_Adend + PW_Adend);   /**The
end relative humidity of water sorbent**/
if (RHW_Adend <= 0) {continue;}
W_sat = InterP(WRnum, MRHa, MWDa, RHW_Adend);                     /**The end
water content of sorption**/


/**Desorption process**/
W_dry = W_sat - Delt_Wcycle;                          /**The end water content of
desorption**/
if (W_dry <= 0) {continue;}
RHW_Deend = InterP(WRnum, MWDd, MRHd, W_dry);                     /**The end
relative humidity of desorption of water sorbent**/
RHW_Deave = 0, W_Detemp = W_dry;
for (p = 0; p < WDnum; p++){RHW_Deave = InterP(WRnum, MWDd, MRHd, W_Detemp)
/ WDnum + RHW_Deave, W_Detemp = W_Detemp + Delt_Wcycle / WDnum;}
                                  /**The average relative humidity of
desorption of water sorbent**/
cout<<RH_ADin<<'\t'<<W_sat<<'\t'<<RHW_Deend<<'\t'<<RHW_Deave<<endl;
dW_Adave =  622 * RHW_Deave / (Patm / PW_Adend - RHW_Deave / 100) / 100;
```

```
FlowRatio = max((SWHR + Delt_dDe) / (d_Adin - dW_Adave),2.0);

Eta_AdT = 0.8 / pow(FlowRatio,0.2);

T_Adout = (T_ADin <= TW_Adend) ? T_ADin : Eta_AdT * TW_Adend + (1 - Eta_AdT) *
T_ADin;

cout<<T_ADin<<'\t'<<d_Adin<<'\t'<<dW_Adend<<'\t'<<RHW_Adend<<'\t'<<RHW_Deave
<<'\t'<<dW_Adave<<'\t'<<FlowRatio<<'\t'<<Eta_AdT<<endl;


    /*Condensation process*/

    T_Condout = T_ADin + Delt_TCond;                    /**The temperture of
outlet air from the condenser**/


    P_Condout = exp(-5800.2206 / (T_Condout + 273.15) + 1.3914993 - 0.048640239 *
(T_Condout + 273.15) + 0.000041764768 * pow(T_Condout + 273.15,2) - 0.000000014452093 *
pow(T_Condout + 273.15,3) + 6.5459673 * log(T_Condout + 273.15));
                                                    /**The saturated water vapor pressure of
outlet air from condenser**/
    d_Condout =  622 / (Patm / P_Condout - 1);       /**The humidity of outlet
air from condenser**/


    /**Desorption process**/

    dW_Deout = d_Condout + SWHR;                     /**The average humidity of
outlet air from desorption bed**/

    dW_Deave = dW_Deout + Delt_dDe;                  /**The average humidity
of desorbed water sorbent**/

    PW_Deave = Patm * 100 / (622 * RHW_Deave / dW_Deave + RHW_Deave);    /**The
average saturated water vapor pressure of desorbed water sorbent**/

    TW_De = InterP(PTnum, MPvs, MT, PW_Deave / 1000);          /**The
temperature of desorbed water sorbent**/
```

```
PW_Deend = Patm * 100 / (622 * RHW_Deend / d_Condout + RHW_Deend);      /**The end saturated water vapor pressure of desorbed water sorbent**/

TW_Decheck = InterP(PTnum, MPvs, MT, PW_Deend / 1000);                  /**The temperature of desorbed water sorbent**/


T_De = max(TW_De, TW_Decheck) + Delt_TDe;                               /**The heat source temperature for desorption**/

if (T_De >= 100) {continue;}

T_Deout = (T_De - Delt_TDe >= T_Condout) ? Eta_DeT * (T_De - Delt_TDe) + (1 - Eta_DeT) * T_Condout : T_Condout;        /**The temperature of outlet air from desorption bed**/


Q_SolarS = T_Deout * 1.01 + dW_Deout / 1000 * (hw + 1.84 * T_Deout) - T_Condout * 1.01 - d_Condout / 1000 * (hw + 1.84 * T_Condout) + SWHR / Delt_Wcycle / 1000 * (TW_De - TW_Adend) * cpw;


SEXC = Q_SolarS * 1000 / SWHR;



    if (SEXC_OP < 0 || SEXC <= SEXC_OP){SEXC_OP = SEXC, T_De_OP = T_De, SWHR_OP = SWHR, T_Cond_OP = T_Condout, T_AD_OP = T_ADin;}
  }


if (SEXC_OP < 0){
  Output<<'\t'<<0.0/0.0;
  Output0<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;
  Output1<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;
```

```
            Output2<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;
            Output3<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0;
            CErelation<<'\t'<<T_ADin<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0<<'\t'<<0.0/0.0<<'\t'<<0
.0/0.0;
        }
        else {
            Output<<'\t'<<0.75 * 1/(SEXC_OP/1000/3.6); /*SWP kg kWh-1*/ /*Solar-thermal conversion
efficiency is 75%*/
            Output0<<'\t'<<d_Adin / 1000<<'\t'<<SWHR_OP;
            Output1<<'\t'<<d_Adin / 1000<<'\t'<<T_AD_OP;
            Output2<<'\t'<<d_Adin / 1000<<'\t'<<T_Cond_OP;
            Output3<<'\t'<<d_Adin / 1000<<'\t'<<T_De_OP;
            CErelation<<'\t'<<T_ADin<<'\t'<<T_AD_OP<<'\t'<<T_Cond_OP<<'\t'<<T_De_OP<<'\t'<<SW
HR_OP<<'\t'<<SEXC_OP;
        }
    }
    CErelation<<endl;
    Output<<endl;
    Output0<<endl;
    Output1<<endl;
    Output2<<endl;
    Output3<<endl;
  }

  WRrelation.close();
  PTrelation.close();
```

```
    CErelation.close();
    Output.close();
    Output0.close();
    Output1.close();
    Output2.close();
    Output3.close();
    return 0;
}

double InterP(int index, double A[], double B[], double a){

    double b;
    int i;
    if (a < A[0]) {b = 0.0 / 0.0;}
    else for(i = 0; i < index; i++){if(A[i] <= a && a <= A[i + 1])break;}
    if(i == index){b = B[index];}
    else b = (a - A[i]) / (A[i + 1] - A[i]) * (B[i + 1] - B[i]) + B[i];
    return b;
}
```

Code S5 Codes of Python for calculating the operational efficiency and annual electricity production of a square meter PV panel considering the impacts of weather

```
import os
import time
from concurrent.futures import ThreadPoolExecutor, as_completed, wait
import numpy
import pandas
import netCDF4
import threading
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap

Result_1 =
[numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440])]
Result_2 =
[numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440])]
Result_3 =
[numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440])]
Result_4 =
[numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440]),numpy.zeros([600,1440])]
List_Result = [Result_1, Result_2, Result_3, Result_4]

class MultiThread (threading.Thread):
    def __init__(self, func, threadID, file):
        self.func     = func
        self.threadID = threadID
        self.file     = file
        self.result   = None
        threading.Thread.__init__(self)
```

```
    def run(self):
        print("***************" + str(self.threadID) + "Start******************")
        List_Result[self.threadID-1] = self.func(self.file)
        print("***************" + str(self.threadID) + "Finished******************")

def Read_NC4(file_name):
    data  = netCDF4.Dataset(file_name)
    #  T     = data.variables["Tair_f_inst"][:]
    #  Y     = data.variables["Qair_f_inst"][:]
    G_tem     = data.variables["SWdown_f_tavg"][:]
    U_tem     = data.variables["Wind_f_inst"][:]
    T_tem     = data.variables["Tair_f_inst"][:]
    data.close()

    G_data    = G_tem.data[0]
    U_data    = U_tem.data[0]
    T_data    = T_tem.data[0]

    G_data[G_data == -9999] = numpy.nan
    U_data[U_data == -9999] = numpy.nan
    T_data[T_data == -9999] = numpy.nan

    return G_data, U_data, T_data

def Calculate_yita(T, U, G):

    rows, cols = T.shape
    yita = numpy.zeros([rows, cols])
```

```
    for r in range(rows):
        for c in range(cols):
            yita[r][c] = 0.182*(1-0.0043*((T[r][c]-273.15)+G[r][c]/(30.02+6.28*U[r][c])-25))
    return yita

def Calculate_E(yita, G):
    E_array = 3 * 0.001 * 0.9 * 0.85 * numpy.multiply(yita, G)

    return E_array

def Calculate_Main(list_file, k):

    print("***************"+str(k)+"Start***************")
    E_total      = numpy.zeros([600, 1440])
    time_start = time.time()
    n = 0
    na = ""
    for file_name in list_file:
        try:
            if file_name[22 : -21] != na:
                na = file_name[22:-21]
            print("course"+str(k)+" settle documents: "+str(na))
            G_air, U_air, T_air = Read_NC4(file_path + "\\" + file_name)
            E_temp = Calculate_E(Calculate_yita(T_air, U_air, G_air), G_air)
            E_total += E_temp
            n += 1
        except:
            print(file_name + " error !")
            continue
```

```
    time_end = time.time()
    print("**************course"+str(k)+"Finished"+str(n)+"files, Time for" + str(time_end-time_start)
+ "*****************")

    return E_total
    #  return SWP_sawh, SWP_dawh, SWP_hawh, vi_time_cal, vi_time_ha, vi_time_sa, vi_time_da

if __name__ == "__main__":
    E   = numpy.zeros([600, 1440])
    file_path = r"Global data T and Y 20220101-20221231"
    # file_path = r"test"
    list_file_name = os.listdir(file_path)

    n = int(len(list_file_name)/12)
    all_future = []
    with ThreadPoolExecutor(max_workers=12) as pool:
        for i in range(11):
            all_future.append(pool.submit(Calculate_Main, list_file_name[i*n:(i+1)*n], i+1))
        all_future.append(pool.submit(Calculate_Main, list_file_name[11*n:], 12))

        for future in as_completed(all_future):

            tmp_E = future.result()
            E  += tmp_E

    print("Finished")

    # E_total
    numpy.savetxt("Result\\E_total_kWh m-2 year-1.csv", E, delimiter=",")
```

References

1. L. Hua, J. Xu and Wang, R., *Nano Energy*, 2021, **85**, 105977.
2. R. W. Hyland and A. Wexler, *ASHRAE Trans.*, 1983, **89**, 500-519.
3. Y. Feng, T. Ge, B. Chen, G. Zhan and R. Wang. *Cell Rep. Phys. Sci.*, 2021, **2**, 100561.
4. A. Khutia, H. U. Rammelberg, T. Schmidt, S. Henninger and C. Janiak, *Chem. Mater.*, 2013, **25**, 790-798.
5. J. Xu, T. Li, T. Yan, S. Wu, M. Wu, J. Chao, X. Huo, P. Wang and R. Wang, *Energy Environ. Sci.*, 2021, **14**, 5979-5994.
6. J. Xu, X. Huo, T. Yan, P. Wang, Z. Bai, J. Chao, R. Yang, R. Wang and T. Li, *Energy Environ. Sci.*, 2024, **17**, 4988-5001.
7. J. Yao, H. Xu, Y. Dai and M. Huang, *Sol. Energy*, 2020, **197**, 279–291.
8. M. Lämmle, A. Oliva, M. Hermann, K. Kramer and W. Kramer, *Sol. Energy*, 2017, **155**, 867–879.
9. Y. Zhong, L. Zhang, X. Li, B. El Fil, C.D. Díaz-Marín, A.C. Li, X. Liu, A. LaPotin and E.N. Wang, *Nat. Rev. Mater.*, 2024, **9**, 681-698.
10. T. Yan, T. Li, J. Xu, J. Chao, R. Wang, Y. I. Aristov, G. G. Larisa, D. Pradip and S. S. Murthy, *ACS Energy Lett.* 2021, **6**, 1795-1802.
11. Y. Feng, L. Ge, Y. Zhao, Q. Li, R. Wang and T. Ge, *Energy Environ. Sci.*, 2024, **17**, 1083–1094.
12. C. Jung, J. Song and Y. Kang, Y. *Energy*, 2018, **145**, 458-467.
13. Y. Li, R. Wang, J. Wu and Y. Xu, *Appl. Therm. Eng.* 2007, **27**, 2858-2868.
14. M. Hultén and T. Berntsson, *Int. J. Refrig.*, 2002, **25**, 487-497.
15. N. Dai, X. Xu, S. Li and Z. Zhang, *Appl. Sci.*, 2017, **7**, 197.
16. G. Xu, X. Zhang and S. Deng, *Appl. Therm. Eng.*, 2006, **26**, 1257-1265.
17. J. Cai, Z. Li, J. Ji and Zhou, F., *Energy*, 2019, **139**, 1133-1145.
18. Z. Li and X. Huang, *Appl. Therm. Eng.*, 2022, **200**, 117693.
19. C. Huan, S. Li, F. Wang, L. Liu, Y. Zhao, Z. Wang and P. Tao, *Energies*, 2019, **12**, 2515.
20. J. Shen, T. Guo, Y. Tian and Z. Xing, *Appl. Therm. Eng.*, 2018, **129**, 280-289.
21. H. Guo, Q. Luo, D. Liu, X. Li, C. Zhang, X. He, C. Miao, X. Zhang and X. Qin, *Adv. Mater.*, 2024, **36**, 2414285.
22. H. Shan, C. Li, Z. Chen, W. Ying, P. Poredoš, Z. Ye, Q. Pan, J. Wang and R. Wang, *Nat. Commun.* 2022, **13**, 5406.

23. C. Xiang, X. Yang, F. Deng, Z. Chen and R. Wang, *Appl. Phys. Rev.* 2023, **10**, 041413.
24. T. Li, T. Yan, P. Wang, J. Xu, X. Huo, Z. Bai, W. Shi, G. Yu and R. Wang, *Nat. Water*, 2023, **1**, 971-981.
25. X. Yang, Z. Chen, C. Xiang, H. Shan and R. Wang, *Nat. Commun.*, 2024, **15**, 7678.
26. H. Zou, X. Yang, J. Zhu, F. Wang, Z. Zeng, C. Xiang, D. Huang, J. Li and R. Wang, *Nat. Water*, 2024, **2**, 663-673
27. W. Guan, C. Lei, Y. Guo, W. Shi and Yu, G., *Adv. Mater.*, 2024, **36**, 2207786.
28. J. Wang, W. Ying, B. Lin, C. Li, C. Deng, H. Zhang, S. Wang and R. Wang, *Adv. Mater.* 2025, **37**, 2408977.
29. A. J. Rieth, A. M. Wright, G. Skorupskii, J. L. Mancuso, C. H. Hendon and M. Dincă, *J. Am. Chem. Soc.*, 2019, **141**, 13858-13866.
30. A. J. Rieth, A. M. Wright, S. Rao, H. Kim, A. D. LaPotin, E. N. Wang and M. Dincă, *J. Am. Chem. Soc.*, 2018, **140**, 17591-17596.
31. A. H. Alawadhi, S. Chheda, G. D. Stroscio, Z. Rong, D. Kurandina, H. L. Nguyen, N. Rampal, Z. Zheng, L. Gagliardi and O. M. Yaghi, *J. Am. Chem. Soc.*, 2024, **146**, 2160-2166.
32. C. Sun, Y. Zhu, P. Shao, L. Chen, X. Huang, S. Zhao, D. Ma, X. Jing, B. Wang and X. Feng, *Angew. Chem. Int. Ed.*, 2023, **62**, e202217103.
33. N. Hanikel, X. Pei, S. Chheda, H. Lyu, W. Jeong, J. Sauer, G. Laura and O. M. Yaghi, *Science*, 2021, **374**, 454-459.
34. H. Furukawa, F. Gándara, Y. B. Zhang, J. Jiang, W. L. Queen, M. R. Hudson and O. M. Yaghi, *J. Am. Chem. Soc.*, 2014, **136**, 4369−4381.
35. Z. Liu, J. Xu, M. Xu, C. Huang, R. Wang, T. Li and X. Huai, *Nat. Commun.*, 2022, **13**, 193.
36. NUMBEO: Cost of living, https://www.numbeo.com/cost-of-living, (accessed May 2025).
37. H. Kim, S. Yang, S. R. Rao, S. Narayanan, E. A. Kapustin, H. Furukawa, A. S. Umans, O. M. Yaghi and E. N. Wang, *Science*, 2017, **356**, 430-434.
38. W. Song, Z. Zheng, A.H. Alawadhi and O.M. Yaghi, *Nat. Water*, 2023, **1**, 626-634.
39. H. Kim, S.R. Rao, E.A. Kapustin, L. Zhao, S. Yang, O.M. Yaghi and E.N. Wang, *Nat. Commun.*, 2018, **9**, 1191.
40. R. Li, Y. Shi, M. Wu, S. Hong and P. Wang, *Nano Energy*, 2020, **67**, 104255.
41. A. LaPotin, Y. Zhong, L. Zhang, L. Zhao, A. Leroy, H. Kim, S.R. Rao and E.N. Wang, *Joule*, 2021, **5**, 166-182.
42. H.A. Almassad, R.I. Abaza, L. Siwwan, B. Al-Maythalony and K.E. Cordova, Nat. Commun., 2022, **13**, 4873.

43. T. Li, M. Wu, J. Xu, R. Du, T. Yan, P. Wang, Z. Bai, R. Wang and S. Wang, *Nat. Commun.*, 2022, **13**, 6771.
44. K. Yang, T. Pan, N. Ferhat, A. Felix, R. E. Waller, P. Hong, J. S. Vrouwenvelder, Q. Gan and Y. Han, *Nat. Commun.*, 2024, **15**, 6260.
45. H. Qi, T. Wei, W. Zhao, B. Zhu, G. Liu, P. Wang, Z. Lin, X. Wang, X. Li, X. Zhang and J. Zhu, *Adv. Mater.*, 2019, **31**, 1903378.
46. X. Wang, X. Li, G. Liu, J. Li, X. Hu, N. Xu, W. Zhao, B. Zhu, J. Zhu, *Angew. Chem.-Int. Edit.*, 2019, **131**, 12182–12186.
47. M. Lämmle, T. Kroyer, S. Fortuin, M. Wiese and M. Hermann, *Sol. Energy*, 2016, **130**, 161–173.
48. Dianchacha: LCOE of PV for 1000 hours in 2022, https://www.dianchacha.cn/data#/data, (accessed January 2024).
49. M. Zhou, H. Liu, L. Peng, Y. Qin, D. Chen, L. Zhang and D. L. Mauzerall, *Nat. Sustain.*, 2022, ***5***, 329–338.
50. Chemical Book, https://www.chemicalbook.com, (accessed May 2025).