B1 Chemical Engineering - Thermal Mass in a Desert Bioreactor

1.Introduction

The purpose of this project was to investigate the effect of thermal mass on the temperature extremes within a desert bioreactor. The construction of these bioreactors are a good way to make use of the abundance of otherwise unused land available. The bioreactors sequester CO_2 by cultivating a mass of duckweed, a form of single celled algae. The duckweed absorbs CO_2 from the surrounding atmosphere to aid photosynthesis, which results in growth. It is then harvested and used as biofuel. Being a form of algae, duckweed thrives when grown on the surface of water. As with almost all organisms, it's growth is subject to being maintained within a specific temperature range:

I. If at any time the temperature of the duckweed exceeds 40˚C or falls below 10˚C for a period of one hour or more it will die
II. If at any time the temperature of the duckweed exceeds 45˚C or falls below 5˚C it will die instantly
III. In addition to these constraints, the duckweed grows most efficiently when kept between 25˚C - 30˚C

The model of the bioreactor is a simple one, consisting of a volume of water headed by an airspace. The duckweed grows throughout the water, but is mainly concentrated on the surface. This is all enclosed by a tank, as illustrated below:

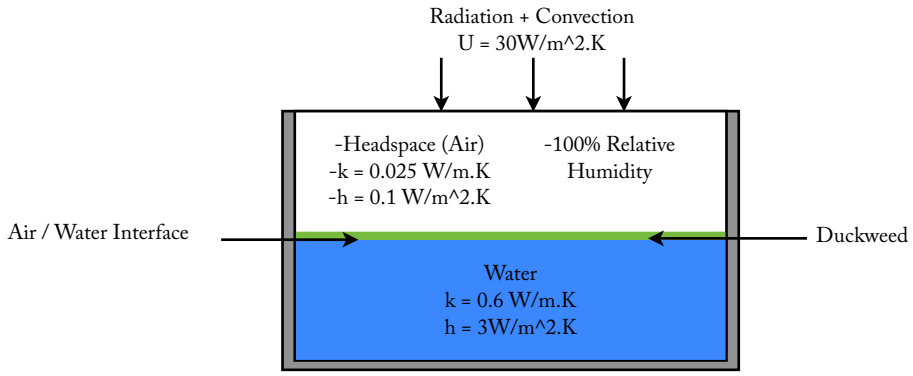


Figure 1: Schematic of the bioreactor

For the purposes of the experiment, some assumptions can be made:

I. The bottom of the reactor can be considered to be perfectly insulated
II. The sides of the reactor can be considered to be perfectly insulated, and the reactor itself can be considered wide enough to neglect any end effects
III. The Air-Water interface layer can be treated as being zero-thickness and having good thermal contact

The ultimate aim of the project is to deduce the minimum air and water spaces that are necessary to maintain the duckweed within the optimum temperature range. This approach concerns the numerical solution of the governing heat transfer equations. Convection of heat will occur through the top of the reactor from the external environment, whilst conductive heat transfer will govern the inside of the reactor.

2.Modelling the Boundary Conditions

The external temperature in the desert atmosphere changes throughout the day. This will cause a varying degree of convection in / out of the reactor throughout a 24-hour period. In addition to the convective transfer, there will be insolation in the day and radiative loss in the night. Separate functions were created to model these two conditions.

2.1.External Temperature

The external temperature varies sinusoidally throughout the day with a minimum temperature of 10˚C at 6AM and a maximum of 50˚C at 6PM. The temperature variation is also periodic, repeating every day. This lead to the following expression being implemented in Matlab:

```
function temp = tempext(t)
temp=303+(20*sin((pi*(t-12))/12));
```

-Where the units of time(t) and temperature(temp) are Hours and Kelvin respectively.

3.2.Radiative Transfer

There is constant insolation of 1000 W/m² between 0600:18:00hrs. Outside of these hours, the reactor loses heat to the night sky. Assuming the temperature of the colder body to be zero, using the Stefan-Boltzmann law the radiative losses are given by:

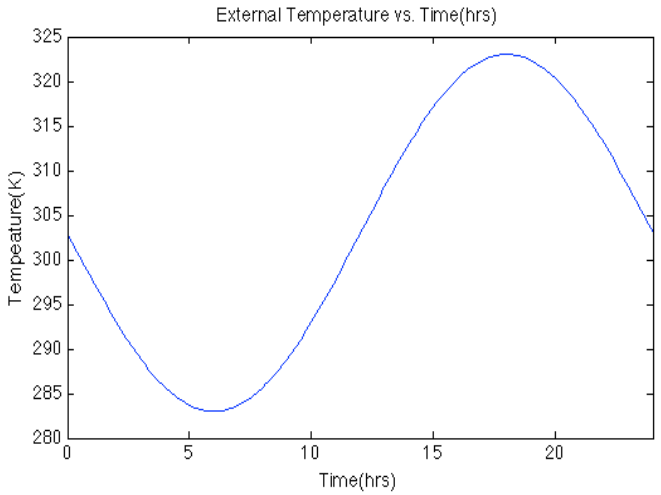
$$Q_{Radiative} = -\sigma.(T_{Surface}^4 - T_0^4) = -\sigma.T_{Surface}^4 W/m^2$$

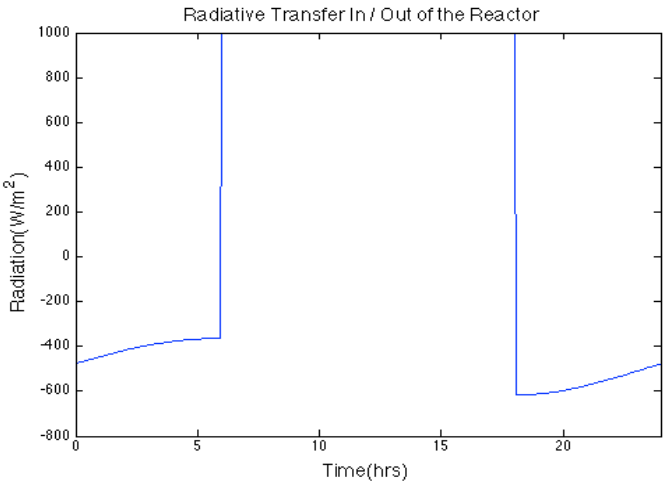
-Where $\sigma = 5.67x10^{-8}\ J/s.m^2.K^4$ *is the Stefan-Boltzmann Constant,* $T_{surface}$ *is the temperature at the surface of the airspace*

Combining the day and nighttime radiative conditions lead to the function:

```
function r = radiation(t)
if sin((pi*(t-6))/12)>=0
    r = 1000; %Units W/m^2
else r = -5.67e-08*T^4;
end
```

Plotting these two functions over a 24-hour period, using *tempext(t).m* as the surface temperature yielded:





3.Modelling Heat Transfer

For the purposes of the experiment, it was assumed that heat transfer through the water / airspace occurs only through conduction. However, heat does enter the reactor from the atmosphere via convection through the top of the reactor.

3.1.Convection Across the Boundary Layer

The change in temperature of an element of mass due to heat flow with time is given by: $Q_{in} = m.C_p.dT/dt$ (W)

The total heat transferring through the boundary layer is: $Q_{Total} = Q_{Convective} + Q_{Radiative}$ (Wm^{-2})

Where: $Q_{convective} = U.(T_{External} - T_{Surface})$ (Wm^{-2}) and $Q_{Radiative}$ = *radiation.m(t,*$T_{Surface}$*)*

Equating these expressions we arrive at the following expression for heat transfer across the boundary layer:

$$U.(T_{External} - T_{Surface}) + (radiation.m)(t,T_{Surface}) = m.C_p.dT_{Surface}/dt\ (Wm^{-2})$$

3.2. Conduction through the Mass

Heat transfers through the mass layers via conduction only. Therefore, equating the change in temperature of an element of mass due to the heat conducted in / out with time we arrive at the expression:

$$k.\partial T/\partial x = m.C_p.\partial T/\partial t$$

The problem was considered as being one dimensional. Therefore the generalised equation for transient heat transfer in one dimension becomes:

$$\partial T/\partial t = \alpha.\partial^2 T/\partial x^2$$

3.3. Solving Numerically

In order to solve these equations numerically, it is first necessary to approximate the solution domain as a set of discrete points in x=ih, t=jk; where h and k are the discrete space and time steps respectively. The above equation can then be adapted for use with discrete quantities. By using a central difference approximation in space, and a forward difference approximation in time, the equation becomes:

$$T_{i,j+1} = T_{i,j} + (k/\alpha h^2).(T_{i+1,j} - 2T_{i,j} + T_{i-1,j}) + O(k^2) + O(kh^2)$$

This equation will govern the heat transfer through the discrete layers in the body of the mass. However, the surface layer element must be modelled differently. Heat will be flowing from above through convection and radiation, but heat will also be flowing from the body layers below through conduction. The very bottom layer will also be modelled differently as conduction will only be flowing from above as the bottom of the tank is perfectly insulated.
The formulas used are *Explicit*, so each next point in time is calculated using three values calculated previously. In order to implement this scheme in MatLab, *for* loops were used:

```
for j=1:(length(t)-1) %Over sample time
T(1,j+1)=T(1,j)+((U*deltat*3600/A1)*(tempext(ti+((j-1)*deltat))-T(1,j))) %Material surface
+(A2*(T(2,j)-T(1,j)))+((deltat*3600*rad(ti+((j-1)*deltat),T(1,j)))/(A1)); %calculation
for i=2:(length(x)-1) %Incremental distances down the solution space
T(i,j+1)=T(i,j)+(A2*(T(i+1,j)-(2*T(i,j))+T(i-1,j)));  %Body calculations
end
T(length(x),j+1)=T(length(x),j)+(A2*(T((length(x)-1),j)-T(length(x),j))); %Water bottom calculation
end
```

T(i,j) is the matrix containing the temperatures in the solution space. *deltat* and *deltax* are the time and space increments with units of hours and metres respectively, U is the heat convective heat transfer coefficient across the top layer and the coefficients A1 and A2 contain the material properties.

It was necessary to multiply in some cases by 3600 as *deltat* has units of hours. However, *tempext.m* operates using hours so it was not necessary there. The modification `tempext(ti+((j-1)*deltat))` allows the *temepext.m* function to update with time as the loop progresses.

3.4. Error Analysis

Truncation errors will occur due to the finite sampling of the exact analytical expression. The local truncation errors of the explicit finite difference scheme used can be derived as follows:

$$T_h = \frac{u(x,t+h)-u(x,t)}{\Delta t} - \beta \frac{u(x+h,t)-2u(x,t)+u(x-h,t)}{h^2} - f(x,t) = u_t(x,t) + \frac{\Delta t}{2} u_{tt}(x,\eta) - \beta\left(u_{xx}(x,t) + \frac{h^2}{6} u_{xxxx}(\xi,t)\right) - f(x,t) = O(\Delta t, h^2)$$

-where h and Δt are the space and time steps respectively

It is therefore shown that the maximum local truncation error at each step will be of order $O(\Delta t, h^2)$.

4.Validating the Daytime Boundary Condition

Without proceeding further into the construction of the simulation, it was necessary to validate the above code against known data in order to ensure that it was behaving correctly. As a test, a volume of water was considered by itself without an enclosing air layer. An analytical expression for daytime transfer through a water depth of 1m is given in the notes:

$$T_{water}(t) = \frac{919}{3} - \frac{10e^{-B(t-6)}(1008B^2+13\pi^2)}{3(144B^2+\pi^2)} + \frac{30\left(-96\sin\left(\frac{\pi t}{12}\right)B^2 + 144B^2 + 8\pi\cos\left(\frac{\pi t}{12}\right)B + \pi^2\right)}{144B^2+\pi^2}$$

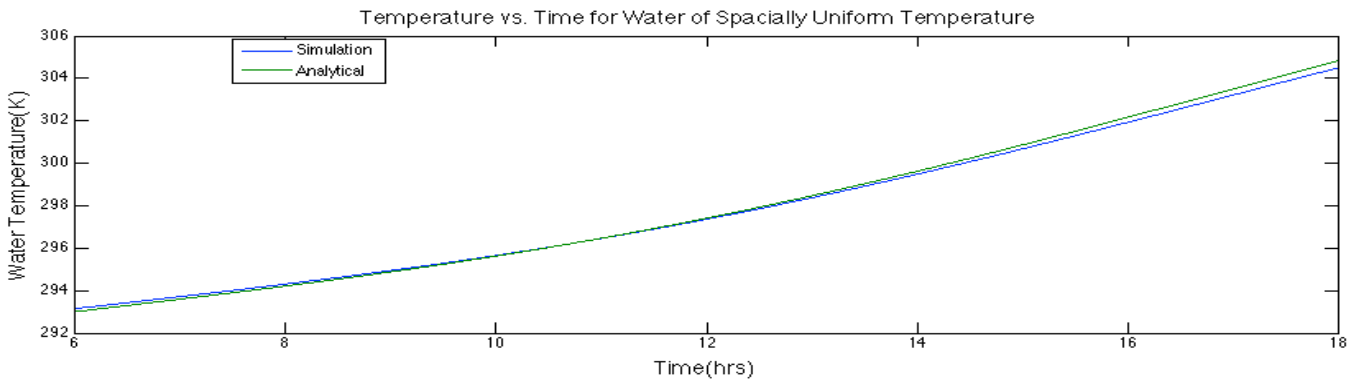
To compare the analytical expression and my simulation accurately, I ran both over one period of daytime - from 06:00-18:00 hrs, with identical material parameters. The analytical expression considers a single element of mass with uniform temperature throughout. In order to simulate this it was necessary to set the water conductivity k_{water} at a very high value. This ensured that the rate of heat transfer *through* the body would be fast relative to the transfer *into* the body, so that the body could be considered as one element. I used the following analysis based on the Biot Number (Bi) to decide a minimum value of k_{water}:

$$\mathrm{Bi} = \frac{hL_C}{k_b} \quad \text{where} \quad L_C = V_{Body} / A_{Surface}$$

For the body to be considered uniform temperature the Biot Number must be approximately <0.1. This gives an approximate minimum k_{water} of 300. I ultimately decided to use a larger value for k_{water} in order to minimise error. Setting the following values for the remaining material properties:

- C_{pwater} = 4180 $JKg^{-1}K^{-1}$
- k_{water} = 10000 $Wm^{-1}K^{-1}$
- ρ_{water} = 1000 Kgm^{-3}
- B = 0.257Khr^{-1}

The results from the two methods were then plotted. They show good agreement between the analytical expression and simulation, confirming that both the simulation method and the daytime boundary conditions were applied correctly. There is a slight discrepancy of 0.3363K at 18:00hrs, but this is likely due to finite grid sizes.



4.Validating the Effect of Grid Spacing

The choice of grid spacing is an important factor in determining both the accuracy and computational feasibility of the simulation. Running at smaller grid spacing provides more accurate results at the cost of larger computation times. A balance between these factors must be chosen before proceeding with further experiments.

4.1.Constructing the Full Code

Before proceeding, the code for the entire reactor had to be assembled. The air layer behaves identically to the water layer, with the only difference being material constants. In order to build the air layer onto the existing code for the water layer, the behaviour of the air-water interface had to be established and subsequently modelled. As stated in the introduction, the air-water interface layer can be treated as being zero-thickness and having good thermal contact.

The initial approach to this modelling this condition was to set up a convective system between the bottom layer of the air and the surface layer of the water. By setting the convective heat transfer coefficient to a large value, good thermal contact would be established. However as will be discussed in more detail later in the report, this approach posed unnecessary constraints on the grid sizes being chosen.
The subsequent approach taken was more successful. A result of two elements of mass being in infinitely good thermal contact is that they will be in a state of thermal equilibrium, therefore the approach taken was derived as follows.

Using $Q = m.C_p.\Delta T$, we equate the heat flow in and out of the two layers, and assume that the two will tend to an equilibrium temperature with infinite time T_F:

$$T_F = \frac{m_{water}C_{pwater}T_{water} + m_{air}C_{pair}T_{air}}{m_{water}C_{pwater} + m_{air}C_{pair}}$$

Therefore, the updated temperature at every iteration can be computed inexpensively. However, the code must still take into account the heat conduction in / out of the element before computing the final temperature. By substituting T_{air} in the above expression with an updated temperature accounting for conduction, the following MatLab code was implemented for the surface layer, and similarly for the water surface layer:

```
Tair = ((mair*cpair*Ta(length(xair),j)) + (mwater*cpwater*Tw(1,j)))/((mair*cpair) + (mwater*cpwater));
Ta(length(xair),j+1) = Tair + A*(Ta(length(xair)-1,j)-Tair);
```

4.2.Stability

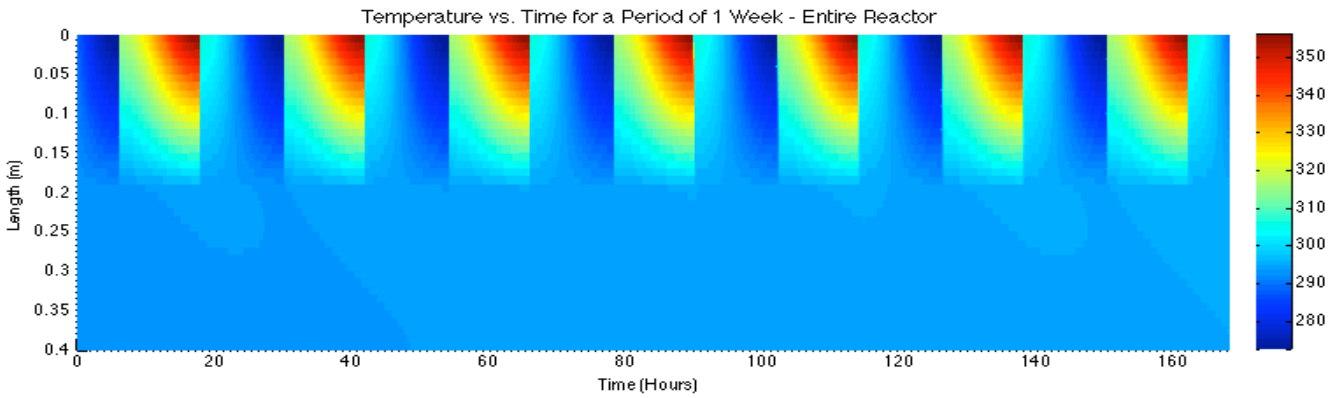
There exists a set of conditions under which the numerical solutions being used will converge. Outside of these conditions the solution will diverge, leading to large computational errors. It is therefore important that the simulation is run within these conditions.
Consider the Diffusion Equation, a parabolic Partial Differential Equation. Stability analysis[2] shows that the solution will only converge if the value of $r = k\delta t/C_p\delta x < 0.5$[2]. The simulation involves the solution of the Diffusion Equation in conjunction with other types of heat transfer, so there must also exist a limit on the coefficients set in the simulation. The *r–value* is a function of both material properties and grid spacing. It therefore follows that if the material properties are constant, judicious choices of grid spacing must be made in order for the simulation to converge.

Returning to a point made earlier (*Section 5.1*), it was decided to use an alternative method for modelling the good thermal contact at the air-water interface. With knowledge of the stability analysis stated above, it is clear that using a very large value for thermal conductivity at the interface would have caused the *'r–value'* to be very large. This would have caused the solution to diverge, and therefore a very small value would have to have been used for the time step in order to compensate. By using the subsequent method the simulation achieves perfect thermal contact whilst being able to run with a larger time step and therefore at computationally less expensive conditions. The following grid spacing experiments were performed:

No.	*deltat*	*deltaxair*	*deltaxwater*	*r_airsurface convection*	*r_airsurfacebody conduction*	*r_waterbody conduction*	*Does it Run?*	*Comments*
1	0.0001	0.1	0.1	0.107	8.96E-04	4.99E-06	Yes	Fast, but insufficient spacial resolution to be practical
2	0.0001	0.01	0.01	1.07	0.0896	4.99E-04	Yes	Good spacial resolution, slow to calculate

No.	*deltat*	*deltaxair*	*deltaxwater*	*r_airsurface convection*	*r_airsurfacebody conduction*	*r_waterbody conduction*	*Does it Run?*	*Comments*
3	0.001	0.01	0.01	10.74	0.896	4.99E-03	No	Solution diverges
4	0.0005	0.01	0.01	5.37	0.448	2.50E-03	No	Solution diverges
5	0.0002	0.01	0.01	2.15	0.179	9.99E-04	No	Solution diverges

The analysis shows that there exists an *'r-value'* for both conductive and convective heat transfer in both mediums, above which the solution will diverge. It is clear from the results that the difficulty lies in staying below the maximum *'r-value'* of the air surface's convective element. The value derived earlier for the the maximum conductive *'r-value'* of 0.5 seems here to be correct. To confirm this, I re-ran Experiment 4 but changed the value of U1 (the convective heat transfer coefficient) to be a 6th of it's original value. This in theory will keep *r_airsurface_convection* within 1, which we already know from experiment 2 to be small enough.
As predicted, the simulation converged, thus confirming that *r_airsurfacebody_conduction* = 0.448 is an acceptable value and the validity of the previous stability analysis. Therefore, the grid spacing with which I proceeded with in the further simulations were those from Experiment 2. Below is a a plot showing the temperature variation through the entire reactor over a period of one week, simulated according to the chosen grid spacing:



The temperature variations through the entire reactor can be seen clearly each day. The variations in water temperature are small compared to that of air as it has a far larger heat capacity. This result confirmed that the fully constructed code was behaving as expected.

5.Further Exploration into the Effects of Evaporation / Condensation

The airspace is maintained at 100% relative humidity. This means that the enthalpy changes associated with humidification / condensation are more significant than they would be in dry air. In order to model these effects it was therefore necessary to construct a function to equate the change of temperature of the humid air with enthalpy change before proceeding further.

5.1.Relating Enthalpy and Temperature

The data was obtained from a Psychrometric Chart[1] for air at 100% relative humidity. By inputting data from this chart, I used MatLab's *polyfit* library function to approximate an Enthalpy-Temperature curve, from which I created my *enthair.m* and *tempair.m* functions, as can be seen below.

Data was inputted between the ranges -10°C-86°C as a broad range of data was necessary in ensuring that the function continued to provide accurate interpolations at high temperature values. Using [P,S,MU] the data was scaled and centered to better accommodate the large range of input data, which was inputted with an accuracy of 2 decimal places. The large data range enabled the polynomial to be fitted to the 20th order.

```
function enthalpy = enthair(T)

dbt = (-10:2:86); %dbt = Dry Bulb Temperature
enth = [...]; %Corresponding enthalpy values

t = T-273.15; %Absolute scale correction
[P,S,MU] = polyfit(dbt,enth,10);
%Finding the coefficients of the polynomial

enthalpy = polyval(P,t,S,MU);
```

```
function temperature = tempair(h)

dbt = (-10:2:86); %dbt = Dry Bulb Temperature
enth =%Corresponding enthalpy values

[P,S,MU] = polyfit(enth,dbt,10); %Finding the
coefficients of the polynomial

temperature = polyval(P,h,S,MU)+273.15;
```

5.2.Error Analysis

The *polyfit* function uses a least-squares approach to make the interpolation. To estimate the error I constructed the following code in MatLab. The code produced the error estimates for both functions as below:

```
meanerror = ((sum(abs((enth-enthair(T))./enth)))/length(T))*100;
[maxerror,Imax] = max(abs(((enth-enthair(T))*100)./enth));
[minerror,Imin] = min(abs((((enth-enthair(T))*100)./enth)))
```

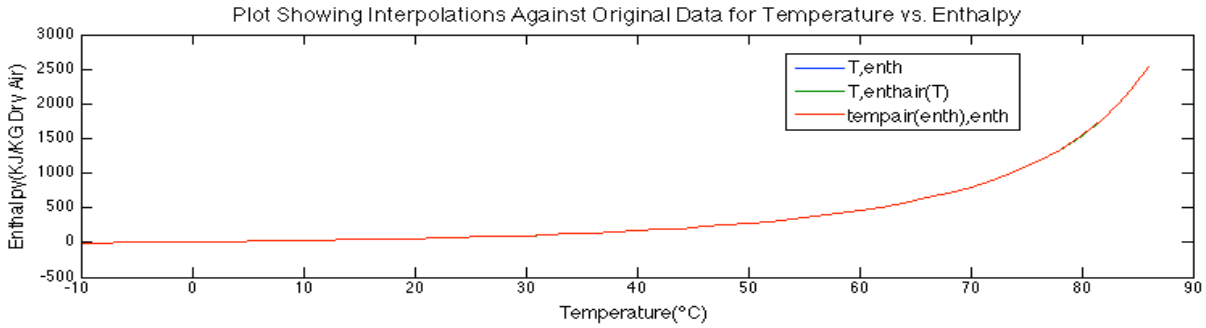
enthair.m

```
Mean Error = 0.0404372%
Max. Error = 0.949158%  at Index 3
Min. Error = 5.1131e-07% at Index 42
```

tempair.m

```
Mean Error = 0.00234272
Max. Error = 0.0198412  at Index 6
Min. Error = 3.99898e-09 at Index 46
```

Therefore the error ranges between 0.949%-5.11e-07% and 0.0198%-3.99e-09% respectively which are both tolerable errors. The plot below illustrates the good agreement between the interpolating functions:



5.3.Modelling Specific Heat Capacity in Moist Air

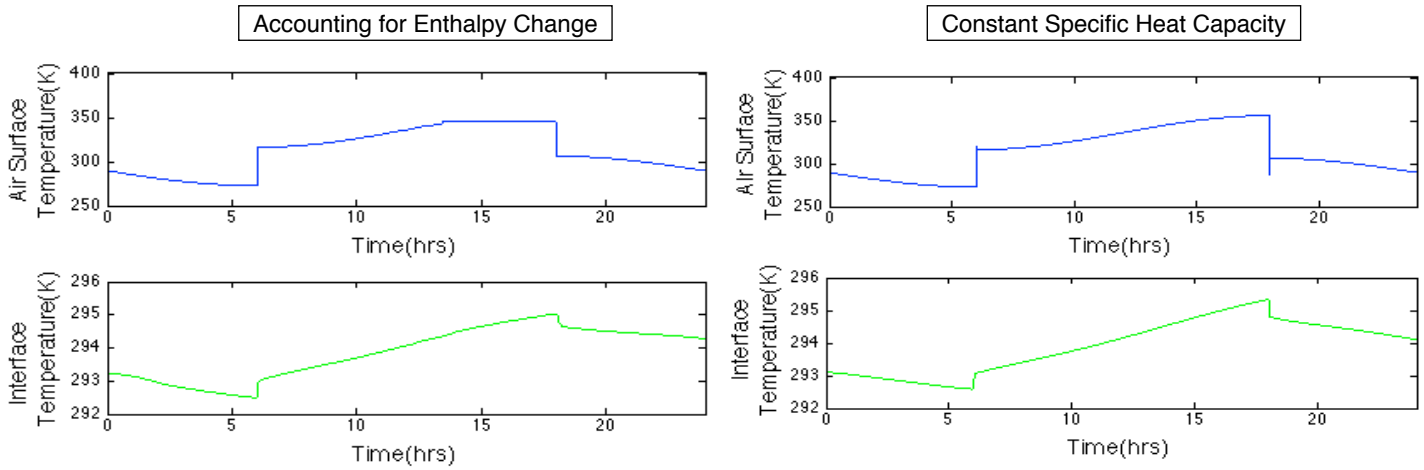
When modelling the water transfer, it was possible to use a constant value for the specific heat capacity C_p, as there is no phase change associated with heating, and the change in C_p with temperature is negligible. However, when 100% relative humidity air is heated, not all the heat energy is used to change the temperature. The amount of water that is stored in the air at a specific pressure increases with temperature. Therefore, assuming as we have that the air is maintained at a saturated (100% RH) level, there must be evaporation of water associated with temperature increases, and condensation associated with the reverse. Some of the heat flowing into the air must therefore be associated with evaporation to maintain humidity. This however is taken into account by the data from the Psychrometric Chart in HLT, measuring enthalpy (KJKg^{-1} Dry Air) against temperature at 100% relative humidity.

In order to calculate the actual temperature change resulting from heat flow, I wrote a function based on the following argument. Assuming that heat flow can be equated to enthalpy change:

$$1)\ Q = \Delta h; \qquad 2)\ h_2 = h@T_1 + \Delta h = h@T_1 + Q; \qquad 3)\ T_2 = T@h_2$$

This expression was then implemented in MatLab using the two previously constructed functions in *Section 5.1:*

```
T2 = tempair(((enthair(T1)*m)+(Q/1000))/m);
 tempchangeair = T2 - T1;
```



To compare the effects of modelling the enthalpy change, I plotted the two different simulation models over a period of 24 hours, both with 0.1m of air and 0.2m of water, as above. As can be seen above, the difference between the plots is negligible, so the effects of evaporation / condensation were minimal when compared to the use of a constant value of C_p. Further investigation was therefore continued using a constant value of C_p.

6. Temperature Variation Within the Reactor

Having settled on a suitable choice of grid spacing, it was necessary to run the simulation for an extended period of time to investigate any cumulative effects of the heat transfer occurring. The simulation was run over a period of three months, with the air and water spaces being varied.

To plot the daily maximum and minimum temperatures, a script was written using a *for* loop to extract data from the temperature matrix as follows:

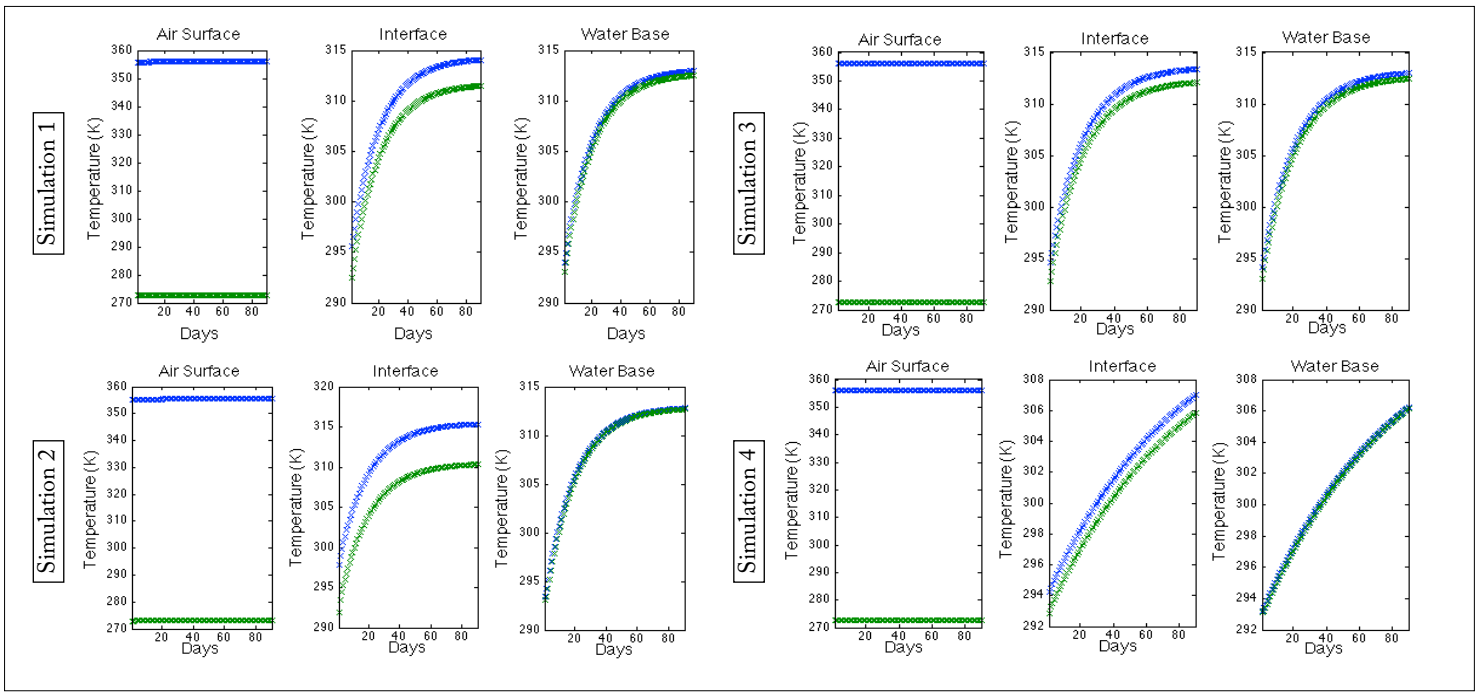
```
for j = [1,length(xair),length(xtotal)]
    for i = 0:N-1
        tempmax(j,i+1) = max(Tempreactor(j,((1+(P*i)):(P+(P*i))))); %P = Period of sampling
        tempmin(j,i+1) = min(Tempreactor(j,((1+(P*i)):(P+(P*i)))));
    end
end
```

The table below shows the variation in grid spacing at each simulation.

Simulation No.	1	2	3	4
Head Space(m)	0.1	0.2	0.05	0.2
Water Depth(m)	0.1	0.05	0.2	0.2

It can be observed that the maximum and minimum temperatures are governed by the amount of air and water space. For extended simulations, it is clear that even air and water space values of 0.2 and 0.2 respectively do not provide a sufficient thermal barrier so as to maintain the duckweed within the optimum temperature range of 25°C-30°C, however the temperature change is well within the safe survival range.
A common attribute of the plots is that the interface temperatures of all appear to tend to constant values near to 310K, which suggests that the thermal equilibrium is independent of air and water space, although these parameters do affect the time taken to approach this equilibrium.

Beneath are the plots of the daily maximum and minimum (blue and green respectively) temperatures:



7. The Minimum Design

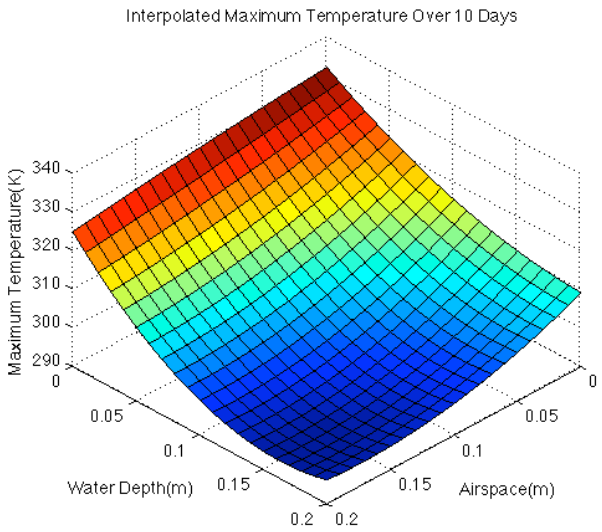
The ultimate goal of this project was to develop a method of deducing the minimum required head / water space to keep the temperature of the interface within a specific range. My solution to this was to take 9 sample points, varying both the head and water spaces between 0.02, 0.10 and 0.20m over a period of 10 days. From each simulation the maximum temperature was recorded:

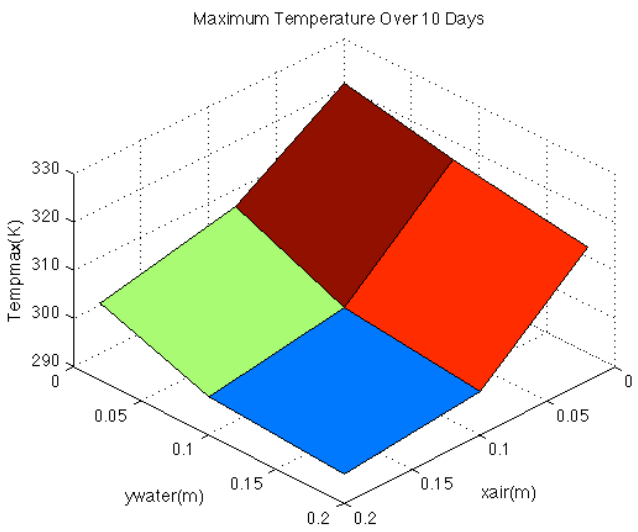
Height of Airspace	0.02	0.10	0.20	0.02	0.10	0.20	0.02	0.10	0.20
Depth of Water	0.02	0.02	0.02	0.10	0.10	0.10	0.20	0.20	0.20
Maximum Temperature(K)	326.36	312.04	306.17	321.77	302.44	298.13	317.77	298.98	296.13

From these nine points I used *polyfit* to interpolate six 2D plots, three each as functions of *xair* and *ywater* . Together these plots form a 3D mesh describing maximum temperature over 10 days against *xair* and *ywater*. The next step was to interpolate the data to form a more continuous surface plot for easier reading.

```
xair = [0.02,0.1,0.2];
yair = [0.02,0.1,0.2];
Ztemp =
[326.36,312.04,306.17;321.77,302.44,298.13;317.77,298
.98,296.13]; %Temperature matrix [x,y]

X = 0:0.01:0.2;
Y = 0:0.01:0.2;

y002 = polyfit(xair,Ztemp(1,:),2);
y10 = polyfit(xair,Ztemp(2,:),2);
y20 = polyfit(xair,Ztemp(3,:),2);

a1 = polyval(y002,X);
a2 = polyval(y10,X);
a3 = polyval(y20,X);

A = vertcat(a1,a2,a3);
A = transpose(A);
x = zeros(21,2);
b = zeros(21,21);

for i =  0:1:20

x(i+1,1:3) = polyfit(yair,A(i+1,:),2);

b(i+1,1:21) = polyval(x(i+1,1:3),Y);

end
```

Below shows a plot comparing the interpolated grid lines in one dimension against the original data. As can be seen the interpolations match the original dataset closely and produce a smooth surface plot from which data can be extrapolated easily and accurately:





8.Conclusion

The results of the project confirmed that the engineering theory that they were based on was correct. The simulation behaved as expected, and the relationship between temperature variations through the reactor and volume of thermal mass was successfully demonstrated. An increase in thermal mass will lead to smaller temperature variations within the reactor.

When explored, it was found that the enthalpy changes associated with evaporation and condensation were not significant in altering the behaviour of the system when compared to the use of a constant specific heat capacity for air (*Section 5.3*).

8.1.Further Experiments

Whilst the explicit finite difference scheme used in the simulations produced successful and accurate results, as a method it is flawed by conditional convergence. As a result, the computational expense of the simulation was governed by the heat transfer through the air, as it's properties made maintaining an *'r-value'* small enough to ensure convergence difficult (*Section 4.2*). It was necessary throughout to run the simulation at a very small time-step which made the simulations relatively time-consuming.

If I were to proceed with future simulations of this kind, I would choose to use a different method such as the Crank-Nicolson Scheme. The advantage of this implicit scheme is that it is unconditionally divergent, and so simulations could be run on any values of grid spacing and still converge. The error in the solution would still however depend on grid spacing.

Bibliography

1 - http://www.uigi.com/UIGI_SI.PDF
2 - 'Kincaid and Cheney - Mathematics of Scientific Computing'