VICTORIA UNIVERSITY | A NEW SCHOOL OF THOUGHT

The design of hydrocoolers using MATLAB®

An addendum to Thorpe, G. R. (2007) Postharvest Biology and Technology, **42**, pp 280-289

Graham Thorpe
Institute of Sustainability and Innovation
PO Box 14428, Victoria University
Melbourne 8001

May 2008

PREFATORY NOTES

Very little research has been published on the design of hydrocoolers used to cool fruits and vegetables. Thorpe (2007) recently employed correlations used primarily for the design of packed bed chemical process equipment to analyse the design and operation of hydrocoolers. It is recognised that the approach adopted by Thorpe (2007) is by no means exclusive of others, and there is no doubt that his approach will be either refined or jettisoned in favour of more sophisticated approaches. Furthermore, as the speed and memory capacities of computers increase we can be certain that the heat, mass and momentum transport processes that occur in hydrocoolers will be quantified in much more detail. The ability to consider increasing detail is one of the underlying features of contemporary engineering science. However, there is no doubt that correlations will remain an indispensable weapon in the design engineer's armoury for the foreseeable future.

In this brief work I present the MATLAB® scripts used to obtain the results reported by Thorpe (2007). The work is motivated by a desire to:

1. Provide readers with an insight how to translate the equations presented in the paper into MATLAB®. This is essentially a didactic aim.
2. Save readers' time if they wish to implement the analysis.
3. Provide a springboard for people to carry out further research on hydrocoolers.
4. Enable the work to be scrutinised for its accuracy.
5. Provide space for an agonistic discourse on the design and operation of hydrocoolers.

This work is not a user-manual for the MATLAB® scripts which are best regarded as works in progress.

This work was prompted by a grant received by the Smart Water Fund of Victoria to investigate the design of water-efficient hydrocoolers.

GR Thorpe
Institute of Sustainability and Innovation
Victoria University

CONTENTS

1. THE COMPUTATIONAL SCHEME

The idea of this brief work is to associate the equations presented in Thorpe (2007) with the MATLAB® script and function m-files used to obtain the results. This is achieved by referencing the equations in the paper from the m-files.

The relevant papers cited in Thorpe (2007) are referenced in the m-files and listed in the bibliography of this report. The components of the program are listed in Table I. The computational scheme is controlled by VU_Hydrocooler.m which invokes a number of function m-files that calculate the thermal diffusivity of the bed of produce, the dynamic and total hold-up of water in the bed, the degree of wetting of the produce and the heat transfer coefficient between the produce and the cooling water. Because the computer program evolved as a research task it does not have a well structured procedure for inputting variables such as the temperature and flow rate of the cooling water and so on. However, the process becomes quite obvious after the user has gained a little familiarisation with the comprehensively annotated scripts.

Table I. Components of the MATLAB® program used to estimate the performance of hydrocoolers.

m-file	Tasks
VU_Hydrocooler.m	Sets up the physical configuration, i.e. the height of the hydrocooler, the flow rate and temperature of the cooling water and the duration of operation. Data on the produce such as its size and initial temperature are also supplied by the user.
Thermal_diffusivity_idealf,m	A function that estimates the thermal diffusivity of a bed of produce irrigated with water.
Dynamic_hold_upf.m	Calculates the dynamic hold-up of water as it flows through a bed of produce.
Wetting_efficiencyf.m	Estimates the fraction of the surface of the produce that is wetted with water.
Total_hold_upf.m	This function calculates the total hold-up of water in the bed of produce.
Heat_transfer_coefficientf.m	Estimates the heat transfer coefficient between the produce and the water.

2. THE THERMAL CONTINUITY EQUATION

Equation 30 in Thorpe (2007) is solved by a well known and simple explicit method that is not outlined in the paper, yet it is used in VU_Hydrocooler.m. This lacuna is remedied by first considering equation 30, namely

$$\rho_w \varepsilon_w c_w \frac{\partial T_w}{\partial t} + \rho_w c_w u_w \frac{\partial T_w}{\partial x} = D_w \frac{\partial^2 T_w}{\partial x^2} + h_w A f \left(T_{sw} - T_w\right) \qquad (1)$$

where the symbols are ascribed meanings given to them by Thorpe (2007). It will be convenient to divide equation 1 by the coefficient of the first term on the left hand side to obtain

$$\frac{\partial T_w}{\partial t} + \frac{\rho_w c_w u_w}{\rho_w \varepsilon_w c_w} \frac{\partial T_w}{\partial x} = \frac{D_w}{\rho_w \varepsilon_w c_w} \frac{\partial^2 T_w}{\partial x^2} + \frac{h_w A f}{\rho_w \varepsilon_w c_w} \left(T_{sw} - T_w\right) \qquad (2)$$

The water flows axially along the bed, hence to calculate the temperature of the water at discrete times and locations we define the following finite difference approximations:

$$\frac{\partial T_{w,j}}{\partial t} \approx \frac{T_{w,j}^{p+1} - T_{w,j}^{p}}{\Delta t} \qquad (3a)$$

$$\frac{\partial T_{w,j}}{\partial x} \approx \frac{T_{w,j}^{p} - T_{w,j-1}^{p}}{\Delta x} \qquad (3b)$$

$$\frac{\partial T_{w,j}^{2}}{\partial x} \approx \frac{T_{w,j+1}^{p} + 2T_{w,j}^{p} - T_{w,j-1}^{p}}{\Delta x^2} \qquad (3c)$$

In the finite difference approximations, equations 3a to 3c, the subscript j refers to the jth node in the axial direction along the bed where $j = 1$ refers to the location of the water inlet. In the MATLAB® scripts the total number of nodes in the direction of the water flow is `nx`. The superscript p refers to the p^{th} time step.

In the MATLAB® VU_hydrocooler.m file we have defined the variable `premult` thus

```
premult=dt/(Dynamic_hold_up*rhow*cpw*dx)    (4)
```

which is the equivalent of

$$premult = \frac{\Delta t}{\varepsilon_w \rho_w c_w \Delta x} \qquad (5)$$

When we substitute the finite difference approximations presented in equations 3a, 3b, and 3c in equation 2 we obtain

$$\frac{T_{w,j}^{p+1} - T_{w,j}^{p}}{\Delta t} + \frac{\rho_w c_w u_w}{\rho_w \varepsilon_w c_w} \frac{T_{w,j}^{p} - T_{w,j-1}^{p}}{\Delta x} = \frac{D_w}{\rho_w \varepsilon_w c_w} \frac{T_{w,j+1}^{p} + 2T_{w,j}^{p} - T_{w,j-1}^{p}}{\Delta x^2} + \frac{h_w A f}{\rho_w \varepsilon_w c_w} \left(T_{sw,j}^{p} - T_{w,j}^{p}\right) \qquad (6)$$

In the MATLAB® script we have made use of the fact that the mass flow rate of water per unit area of bed, f_w, can be expressed as

$$f_w = \rho_w u_w \tag{7}$$

The updated temperatures of the water, $T_{w,j}^{p+1}$, after each time step are found by inserting equation 5 into equation 6 and rearranging the result to obtain

$$\begin{aligned} T_{w,j}^{p+1} &= T_{w,j}^{p} - f_w c_w \times premult \times \left(T_{w,j}^{p} - T_{w,j-1}^{p}\right) \\ &+ \frac{premult}{\Delta x} D_w \left(T_{w,j+1}^{p} + 2T_{w,j}^{p} - T_{w,j-1}^{p}\right) \\ &- h_w fA_f \times premult \times \Delta x \times \left(T_{w,j}^{p} - T_{sw}^{j}\right) \end{aligned} \tag{8}$$

There are slight differences between equation 8 and its transliteration into the MATLAB® script. These arise because the area, A_f, in equation 8 is defined as the surface area of the produce per unit volume of irrigated bed, whereas in the script it refers to the area per volume of bed between the finite difference nodes, *j*. The degree of wetting does not appear in the MATLAB® script because it has been subsumed in the heat transfer coefficient, `htc`.

3. USER-DEFINED VARIABLES

It has already been noted that VU_Hydrocooler is essentially a research tool, and as such it has many of the idiosyncrasies of a work in progress. An aim of the MATLAB® program is to allow a range of design and operating conditions of hydrocoolers to be explored. The would-be user of the program can change any variable at will, but it seems to me that the most obvious ones users will initially want to change are:

`de`	Diameter of the produce, m
`lx`	Length of bed of produce, m
`fw`	Mass flow rate of water per unit cross-sectional area of the bed, kg/(m^{2}s)
`tinitial`	Initial temperature of the produce, °C
`twater(1)`	Temperature of water used to cool the produce, °C

4. POSTSCRIPTAL NOTE

The reader is now in a position to explore the work published by Thorpe (2007) and use it, criticise it, extend it or develop some quite different approach.

If the reader needs any assistance please do not hesitate to contact the author.

5. REFERENCE

Thorpe, G. R., 2007, Towards a semi-continuum approach to the design of hydrocoolers for horticultural produce. Postharvest Biology and Technology. Volume 42, pp 280-289.

6. BIBLIOGRAPHY

de Castro, L. R., Vignault, C. and Cortez, L. A. B. (2005) Effect of container openings and air flow rate on energy required for forced air cooling of horticultural produce. Canadian Biosystems Engineering, **47**, pp3.3-3.9

Larachi, F., Alix, C., Grandjean, B. P. A. and Bernis, A. (2003) Nu/Sh correlation for particle-liquid heat and mass transfer coefficients in trickle beds based on Péclet similarity. Chem. Eng. Res. Dev. (Trans IChemE part A), 81, pp 689-694.

Larachi, F., Belfares, L. and Grandjean, B. P. A. (2001) Prediction of liquid-solid wetting efficiency in trickle flow reactors. Int. Comm. Heat and Mass Transfer, **28**, pp 595-603.

Larachi, F., Belfares, L., Iluta, I. and Grandjean, B. P. A. (2004) Liquid hold-up correlations for trickle beds without gas flow. Chem. Engng and Processing, **43**, pp 85-90.

Saez, A. E. and Carbonell, R. G. (1986) Hydrodynamic Parameters for Gas-Liquid Cocurrent Flow in Packed Beds, AIChE Journal, **31**, pp 52-62.

Appendix I VU_Hydrocooler.m

```
%   To explore the performance of a hydrocooler in which there
%   is a finite rate of heat transfer between the cooling water
%   and the produce, and there is heat transfer by thermal
%   conduction within the produce.
%
%   The equation numbers in this MATLAB script and its
%   associated functions refer to those in Thorpe, G. R. (2007),
%   Towards a semi-continuum approach to the design of
%   hydrocoolers for horticultural produce.
%   Postharvest Biology and Technology. Volume 42, pp 280-289.

%   Graham Thorpe
%   Institute of Sustainability and Innovation
%   Victoria University, Melbourne, Australia
%   Revised May 2008

clear all

%%%%  Physical properties of the produce   %%%%

%   k       Thermal conductivity, W/(m C)
%   rho     Density, kg/m^3
%   cp      Specific heat, J/(kg C)
%   alpha   Thermal, diffusivity, m^2/s
%   de      Equivalent diameter of the produce, m
%   radius  Equivalent radius of the produce, m
%   phi     Sphericity of the produce defined implicitly
%           by equation 27
%   eps     Volumetric fraction voids between the pieces
%           of produce in the hydrocooler
%   epsolid Volumetric fraction of voids between the
%           pieces of produce in the hydrocooler

    k=.6;
    rho=1000;
    cp=4000;
    alpha=k/(rho*cp);
    de=0.025;   %   Set by user
    radius=de/2;
    phi=1;
    eps=0.4;
    epssolid=1-eps;

%%%%%%%%   Physical properties of water    %%%%%%%

%   rhow    Density, kg/m3
%   cpw     Specific heat, J/(kg C)
%   kwater  Thermal conductivity, W/(m C)
%   sigma   Surface tension of water, N/m
%   viscw   Viscosity of water, Pa s

    rhow=1000;
    cpw=4180;
    kwater=0.6;
    sigma=0.072;
```

```
    viscw=0.001


%%%%%%%%   Physical properties of air    %%%%%%%

%   rhoa    Density, kg/m3
%   kair    Thermal conductivity, W/(m C)
%   visca   Viscosity of air, Pa s

    visca=18e-6;
    rhoa=1.2;
    kair=0.025;

%%%%   Physical characteristics of the hydrocooler   %%%%

    lx = 0.5;           %   Length of the bed of produce, m.
                        %       Set by user.
    dc=1.0;             %   A length scale that represents the
                        %   diameter of the bed of produce, m
    fw=16               %   Mass flow rate of water per unit
                        %   cross-sectional area of the bed of
                        %   produce, kg/(s.m^2).  Set by user.
    uDarcian=fw/rhow    %   Superficial velocity of water
                        %   through the bed of produce, m/s
%%%%   Calculated charactersitics of the bed of produce   %%%%

    Dispersion_tensor=Thermal_diffusivity_idealf(de,kair,k,kwater,...
        viscw,rhow,cpw,uDarcian,epssolid)
    Dynamic_hold_up=Dynamic_hold_upf(viscw,eps,phi,de,sigma,rhow,...
        uDarcian)
    Eta_wetting = Wetting_efficiencyf(viscw,rhow,eps,phi,de,dc,...
        sigma,uDarcian)
    Total_hold_up = Total_hold_upf(viscw,rhow,eps,phi,de,uDarcian,...
        Eta_wetting)
    htc=Heat_transfer_coefficientf(viscw,visca,rhow,rhoa,sigma,cpw,...
        kwater,eps,de,uDarcian)

%    Set time of operation, minutes
    Time_of_operation = 1.6;
    dt=0.04;             %   Integration time step, s.
%   Number of times steps over which the calculations are executed
    nsteps=Time_of_operation*60/dt;

%%%%%   Numerical parameters   %%%%%%%

    nr=21;              %   Number of finite difference nodes in
                        %   the individua pieces of produce

    nx=11;              %   Number of finite difference nodes
                        %   in the bed of produce
    dx=lx/(nx-1);       %   Distance between equidistant nodes
    Atotal=1.8*dx/radius;%  Total area of spherical produce in
                        %   each finite volume
                        %   This is equivalent to equation 27.

%%%%%%% Set up finite difference coefficients %%%%%%

        nu=0.9          %   Ratio of the radii of successive nodes,
                        %   equation  32.
```

```
        r(1)=0;         %   Radius of the central node, i.e. zero.
        dr(1)=1         %   An arbitrary radiual distance of the
                        %   first node. This will be corrected to
                        %   account for its true value.  See below.
%   Equation 53:
        for i=2:nr-1
            dr(i)=dr(i-1)*nu;  %   Set up distances between the nodes
        end
%   Calculate the relative positions of the radii of the nodes given
%   dr(1)=1
        for i=2:nr
            r(i)=r(i-1)+dr(i-1);
        end
%   Correct values of dr(i) and r(i) to account for the fact that
%   r(1) must be chosen so that r(nr)=radius

        for i=1:nr-1
            dr(i)=dr(i)*radius/r(nr);
        end

        for i=1:nr
            r(i)=r(i)*radius/r(nr);
        end


%%%%%%  Coefficients for the first differentials  %%%%%

%   A slight variation of equations 34a, 34b and 34c
        for i=2:nr-1
            c1(i)=-dr(i)/(dr(i-1)*(dr(i)+dr(i-1)));
            c2(i)=-(dr(i-1)-dr(i))/(dr(i)*dr(i-1));
            c3(i)=dr(i-1)/(dr(i)*(dr(i)+dr(i-1)));
        end

%%%%%%  Coefficients for the second differentials  %%%%%

%   Equations 36a, 36b and 36c
        for i=2:nr-1
            c11(i)=2/(dr(i-1)*(dr(i)+dr(i-1)));
            c12(i)=-2/(dr(i)*dr(i-1));
            c13(i)=2/(dr(i)*(dr(i)+dr(i-1)));
        end

%%%%%%    Coefficients for zero temperature    %%%%%%
%%%%%%  gradient in the centre of the produce  %%%%%%
%%%%%%        equations 38a and 38b            %%%%%%

        denom=2*dr(1)*dr(2)+dr(2)^2;
        cc1 = (dr(1)+dr(2))^2/denom;
        cc2 = -dr(1)^2/denom;

%%%%%%    Coefficients for temperature gradient    %%%%%%
%%%%%%     at the edge of the produce given by     %%%%%%
%%%%%%        equations 40a, 40b and 40c           %%%%%%

        denom1=dr(nr-1)*dr(nr-2)*(dr(nr-1)+dr(nr-2));
        cen=(2*dr(nr-1)*dr(nr-2)+dr(nr-2)^2)/denom1;
        cenm1=-(dr(nr-1)+dr(nr-2))^2/denom1;
        cenm2=dr(nr-1)^2/denom1;
```

```
%%%%%%% Set up initial conditions %%%%%%%
%   Initial temperature of the produce, deg C:
        tinitial=25.0
        for j=1:nx
            for i=1:nr
                t(i,j)=tinitial;
                tnew(i,j)=t(i,j);
                twater(j)=0.0;
                twaternew(j)=0.0;
        end
    tparti(j)=0;
    tmean(j)=0;

end
%   Temperature of water entering the hydrocooler
    twater(1)=2.5;  %   Set by the user
%   twaternew refers to the temperature of the water
%   at the end of a time step.  It is the updated
%   value of twater.
    twaternew(1)=2.5;
    time=0;

%   ttrack(1,j) refers to the mean temperatures
%   of the produce at each node within the hydrocooler.
%   The values of ttrack are updated and stored after
%   each time step.
    for j=1:nx
           ttrack(1,j)=tinitial;
    end

    timecount(1)=0; %   timecount stores elapsed times, minutes.
                    %   It is updated every time step.

%   Marching through time begins here

for jjj=2:nsteps;

%   Demonstrate that the script is working
    jinterval=100;
    jint=fix(jjj/jinterval);
     if jjj==jint*jinterval
         fprintf('Working %i\n',jjj)
     end
%   End of demonstration

%   Increment the current time by dt
    time=time+dt;

%   Marching through the length of the hydrocooler begins here

for j=1:nx
%   Calculate the surface temperature of the produce
%   using equation 42.

    t(nr,j)=(htc*twater(j)-k*cenm1*t(nr-1,j)...
        -k*cenm2*t(nr-2,j))/(k*cen+htc);
    tnew(nr,j)=t(nr,j);
```

```
%   premult is defined by equations 5 and 6 in this Addendum
    if j>1
    premult=dt/(Dynamic_hold_up*rhow*cpw*dx);
%   The updated temperatures of the water, twaternew(j), are
%   calculated from equation 6 of the Addendum.
    %   Arising from Advection term in equation 30
    twaternew(j)=twater(j)+fw*cpw*(twater(j-1)...
        -twater(j))*premult;

    %   Surface heat transfer to the produce
    twaternew(j)=twaternew(j)-htc*Atotal*premult...
        *(twater(j)-t(nr,j));

    %   Arising from dispersion term in equation 30
    if j<nx
       twaternew(j)=twaternew(j)+premult/dx*...
           Dispersion_tensor*(twater(j-1)...
           -2*twater(j)+twater(j+1));
    else
       twaternew(j)=twaternew(j)+premult/dx*...
           Dispersion_tensor*(twater(j-1)-twater(j));
    end

    end

%   Computing the temperature distribution in the produce begins here

    for i=2:nr-1
        %   Account for the heating effects of respiration using
        %   the expression presented in de Castro, L. R., Vignault,
        %   C. and Cortez, L. A. B. (2005) Effect of container openings
        %   and air flow rate on energy required for forced air cooling
        %   of horticultural produce. Canadian Biosystems Engineering,
        %   47, pp 3.3-3.9


         Heat_source=rho*0.087*exp(0.1197*t(i,j));  %   W/(m^3 of produce)

%   Finite difference forms of first and second spatial derivatives
%   of temperature in the produce, equations 33 and 35.
        dtdr=c1(i)*t(i-1,j)+c2(i)*t(i,j)+c3(i)*t(i+1,j);
        d2tdr2=c11(i)*t(i-1,j)+c12(i)*t(i,j)+c13(i)*t(i+1,j);
%   An explicit expression for the temperature of the produce
%   which is given using a distretyised form of equation 3.
        tnew(i,j)=t(i,j)+alpha*dt*(2*dtdr/r(i)+...
            d2tdr2)+Heat_source*dt/(rho*cp);
    end

%   Calculate the temperature at the centre of the produce using
%   equation 37.
    tnew(1,j)=cc1*t(2,j)+cc2*t(3,j);

    %   Update temperatures at the end of the time step
for i=1:nr
    t(i,j)=tnew(i,j);
end %   Marching through conduction in the produce ends here

%   Calculate the mass weighted average temperature using equation 43
```

```
    tparti(j)=0;    %   Used to sum mass weighted average temperature
    for i=2:nr
        tparti(j)=tparti(j)+(t(i,j)+t(i-1,j))*(r(i)^3-r(i-1)^3)/2;
    end

%   tmean is calculated from equation 43 solved numnerically
%   by equation 44.
    tmean(j)=tparti(j)/radius^3;

%   the following variables enable one to record the values
%   of the mean and centre temperatures as one marches through
%   time
    ttrack(jjj,j)=tmean(j);
    tcentre(jjj,j)=tnew(1,j);
%   Elapsed time of operation in minutes
    timecount(jjj)=timecount(jjj-1)+dt/60;


end %   End of marching along the hydrocooler

%   Set the values of the water temperatures at the
%   start of the next time step to those at the
%   end of the time step just completed.
for j=1:nx
    twater(j)=twaternew(j);
end

end %   End of the marching through time loop

for j=1:nx
    hold on
        plot(timecount,ttrack(:,j),'b')
        title('Temperatures of produce at different bed depths')
        xlabel('Elapsed time, minutes')
        ylabel('Mean temperature of produce, ^oC')

    hold on
end
```

Appendix II Thermal_diffusivity_idealf.m

```
function Thermal_diffusivity=Thermal_diffusivity_idealf(de,tca,tcs,...
    tcw,viscw,rhow,cpw,uDarcian,epssolid)

%   To make an order of magnitude estimate of the axial
%   component of the thermal dispersion tensor in an idealised
%   spatially periodic porous medium using the analysis proposed
%   by Saez, A. E., Carbonell, R. G. and Levec, J. (1986) The
%   hydrodynamics of trickling flow in packed beds, Part I:
%   Conduit models. AIChEJ, 31, pp 52-62.

%   The equation numbers in this function refer to those in
%   Thorpe G.R. (2007), Towards a semi-continuum approach to
%   the design of hydrocoolers for horticultural produce.
%   Postharvest Biology and Technology. Volume 42, pp 280-289.


%   Graham Thorpe
%   Institute of Sustainability and Innovation
%   Victoria University
%   Melbourne
%   Revised May 2008

%   qcell     Volume flow rate of water per metre
%             width of a unit cell, m^3/s
%   deltas    Thickness of solid layer, m
%
%   uAverage  Average velocity of water film, Volume flow
%             rate per meter width divided by thickness
%             of the water film, m/s
%   alphaw    Thermal diffusivity, equation 13
%   eps       Volume fraction of solid phase
%   s         Volume fraction of the liquid film
%   Pe        Peclet number, equation 12

qcell = uDarcian*de;
deltal=(qcell*3*viscw/(rhow*9.81))^0.33333; %   Equation 17
liquidholdup=deltal/de;                     %   Equation 18
deltas=de*epssolid;
deltag=de-(deltal+deltas);                  %   From equation 15
uAverage=qcell/deltal;
alphaw=tcw/(rhow*cpw);
eps=(deltal+deltag)/de;
s=deltal/de;
Pe=uAverage*6*s*deltal/alphaw;

%   Calculation of variables equations 20, 21, 22 (a1 is used
%   for beta1 and so on)
a1=-3*eps*s/20 - 14*tcw*eps*(1-s)/(15*tca);
a2=-9*eps*s/10 - 7*tcw*eps*(1-s)/(30*tcs);
a3=-9*eps^2*s^2 - 4*tcw*eps*s*(1-eps)/tcs-4*tcw*eps^2*s*(1-s)/tca...
    -4*tcw^2*(1-eps)*eps*(1-s)/(tcs*tca);

%   Estimation of thermal dispersivity, equation 11
Thermal_diffusivity = Pe^2*eps*s/24*(37/180+5*eps*s/12+(a1-a2)/a3);
```

Appendix III Dynamic_hold_up.m

```
function Dynamic_hold_up = Dynamic_hold_upf(viscw,eps,phi,...
    de,sigma,rhow,uDarcian)

%   To calculate the dynamic hold-up of water in
%   a bed of horticultural produce that is being
%   cooled in a hydrocooler.  The correlation is
%   obtained from Larachi, F., Belfares, L., Iluta, I.
%   and Grandjean, B. P. A. (2004)  Liquid hold-up
%   correlations for trickle beds without gas flow.
%   Chem. Engng and Processing, 43, pp 85-90.

%   Graham Thorpe
%   Institute of Sustainability and Innovation
%   Victoria University
%   Melbourne
%
%   Revised May 2008

g=9.81;

%%%%%%%%%   Dimensionless groups    %%%%%%%%%

    Re = rhow*uDarcian*de/(viscw*(1-eps));
    Fr = uDarcian^2/(g*de);
    We = rhow*uDarcian^2*de/sigma;
    Eo = rhow*g*de^2*phi^2*eps^2/(sigma*(1-eps)^2);

%%%%%%%%%    Normalised inputs    %%%%%%%%%%

    Omega(1) = log10(Fr/(7.14e-8))/5.9829;
    Omega(2) = log10(Re/0.174)/4.171768;
    Omega(3) = log10(We/4.2e-8)/7.1308;
    Omega(4) = log10(Eo/0.06)/3.82732;
    Omega(5) = 1;

%%%%  Neural network connectivity weights  %%%%

    omega1(1,1)=  13.3463;
    omega1(1,2)=  10.5363;
    omega1(1,3)= -12.5596;
    omega1(1,4)=   4.63111;
    omega1(1,5)=  23.2651;
    omega1(1,6)=   3.75811;
    omega1(1,7)=  13.818;
    omega1(1,8)=  -7.64755;

    omega1(2,1)= -65.2407;
    omega1(2,2)=   9.9268;
    omega1(2,3)=  20.2003;
    omega1(2,4)=  -1.55236;
    omega1(2,5)= -86.6907;
    omega1(2,6)=  -1.55038;
    omega1(2,7)= -19.1719;
    omega1(2,8)=   8.29131;
```

```
    omega1(3,1)=  38.9399;
    omega1(3,2)= -26.6538;
    omega1(3,3)=  -2.06595;
    omega1(3,4)=   0.25875;
    omega1(3,5)=  46.0733;
    omega1(3,6)=   2.2269;
    omega1(3,7)=  -2.46715;
    omega1(3,8)= -13.5451;

    omega1(4,1)=  64.6021;
    omega1(4,2)= -51.9727;
    omega1(4,3)= -14.279;
    omega1(4,4)=  -0.905499;
    omega1(4,5)=  69.0901;
    omega1(4,6)=  -1.19279;
    omega1(4,7)=  -3.4826;
    omega1(4,8)= -16.3938;

    omega1(5,1)= -15.8737;
    omega1(5,2)=  59.445;
    omega1(5,3)= -26.6731;
    omega1(5,4)=   2.31463;
    omega1(5,5)= -18.3042;
    omega1(5,6)= -5.45437;
    omega1(5,7)= -5.70309;
    omega1(5,8)= -18.2135;

    omega2(1)=   5.44266;
    omega2(2)=  10.9984;
    omega2(3)= -15.5088;
    omega2(4)=  19.3656;
    omega2(5)=  -4.57547;
    omega2(6)=  15.4822;
    omega2(7)=  48.7554;
    omega2(8)=  -8.70918;
    omega2(9)= -30.9751;

%%%%%%  Normalised output   %%%%%%

    for j=1:8
        sum1 = 0
            for i=1:5
                sum1 = sum1+omega1(i,j)*Omega(i);
            end
        Gamma(j) = 1/(1+exp(-sum1));
    end
    Gamma(9) = 1;

    sum2=0
    for j=1:9
        sum2 = sum2 + omega2(j)*Gamma(j);
    end

    Psi= 1/(1 + exp(-sum2));

%%%%%  Calculate the dynamic hold-up in a hydrocooler  %%%%%%%

    Dynamic_hold_up = 4.5e-3*10^(1.9497*Psi);
```

Appendix IV Wetting_efficiencyf.m

```
function Eta_wetting = Wetting_efficiencyf(viscw,rhow,eps,phi,dv,...
    dc,sigma,uDarcian)

%   To calculate the wetting efficiency in
%   a bed of horticultural produce that is being
%   cooled in a hydrocooler.  The correlation is taken
%   from Larachi, F., Belfares, L. and Grandjean, B. P. A. (2001)
%   Prediction of liquid-solid wetting efficiency in trickle flow
%   reactors.  Int. Comm. Heat and Mass Transfer, 28, pp 595-603.


%   Graham Thorpe
%   Victoria University
%   Institute of Sustainability and Innovation
%   Melbourne, Australia

%   Revised May 2008


g=9.81;

dh = dv*(16*eps^3/(9*pi*(1-eps)^2))^(1/3);
as = 6*(1-eps)/(phi*dv)+4.0/dc;

%%% Let the gas velocity be much lower than that of the liquid %%%

uGasDarcian=uDarcian/30;

%%%%%%%%%   Dimensionless groups    %%%%%%%%%

    Refg = rhow*(uDarcian + uGasDarcian)*dv/(viscw*(1-eps))
    Fr = uDarcian^2/(g*dv)
    St = uDarcian*viscw/(rhow*g*dv^2)
    Ga = dv^3*g*rhow^2*eps^3/((1-eps)^3*viscw^2)
    Sb = as*dh/(1-eps)


%%%%%%%%%    Normalised inputs    %%%%%%%%%%


%   Equations 7a:

    Omega(1) = (log10(Refg)-0.271842)/3.950878;
    Omega(2) = (log10(St)+5.84164)/3.57484;
    Omega(3) = (log10(Fr)+6.5986)/5.52955;
    Omega(4) = (log10(Ga)-2.43297)/3.58464;
    Omega(5) = (log10(Sb)-0.369216)/0.332352;

%    Equation 7b:
    Omega(6) = 1;

%%%%  Neural network connectivity weights  %%%%
```

```
omega1(1,1)=   4.19968;
omega1(1,2)=  -0.259888;
omega1(1,3)=  -0.481944;
omega1(1,4)=  11.4991;
omega1(1,5)=  -2.02498;
omega1(1,6)=   3.10936;
omega1(1,7)=  -2.13749;

omega1(2,1)= -10.0386;
omega1(2,2)=  -5.88365;
omega1(2,3)=  -0.393829;
omega1(2,4)=   7.51315;
omega1(2,5)= -12.4709;
omega1(2,6)=  -3.0791;
omega1(2,7)=   3.12087;

omega1(3,1)=  9.61655;
omega1(3,2)= 10.5134 ;
omega1(3,3)= -6.64832;
omega1(3,4)= -8.71024;
omega1(3,5)= 10.9642;
omega1(3,6)=  6.14818;
omega1(3,7)= -1.32697;

omega1(4,1)=  0.87737;
omega1(4,2)= -3.9468;
omega1(4,3)= -6.38033;
omega1(4,4)= -8.66964;
omega1(4,5)=  2.62594;
omega1(4,6)=-15.556;
omega1(4,7)=  8.72525;


omega1(5,1)= -4.44327;
omega1(5,2)=  3.68511;
omega1(5,3)= -0.238838;
omega1(5,4)= 11.9850;
omega1(5,5)=  6.96582;
omega1(5,6)=-24.3462;
omega1(5,7)= -6.71659;

omega1(6,1)= -5.38237;
omega1(6,2)= -1.25289;
omega1(6,3)=-12.8234;
omega1(6,4)= -0.087999;
omega1(6,5)= -4.02972;
omega1(6,6)= 17.7918;
omega1(6,7)= -3.91923;

omega2(1) =   8.18548;
omega2(2) =   5.34465;
omega2(3) =  -2.80463;
omega2(4) =  -6.65745;
omega2(5) =   4.78822;
omega2(6) =   5.02297;
omega2(7) = -10.459;
omega2(8) =  -1.03008;
```

```
%%%%%%  Normalised output   %%%%%%

%   Equation 8a:
    for j=1:7
        sum1 = 0;
            for i=1:6
                sum1 = sum1+omega1(i,j)*Omega(i);
            end
        H(j) = 1/(1+exp(-sum1));
    end

%   Equation 8b:
    H(8) = 1;


%   Equation 9:
    sum2=0;
    for j=1:8
        sum2 = sum2 + omega2(j)*H(j);
    end

    S = 1/(1 + exp(-sum2))

%%%%  Calculate the wetting efficiency in a hydrocooler %%%%

%   Equation 10:
Eta_wetting = 0.83*S+0.17
```

Appendix V Total_hold_upf.m

```
function Total_hold_up = Total_hold_upf(viscw,rhow,eps,...
    phi,dv,uDarcian,etawet)

%   To calculate the total hold-up of water in
%   a bed of horticultural produce that is being
%   cooled in a hydrocooler

%   The correletion for total hold-up is taken from
%   Larachi, F., Belfares, L., Iluta, I.
%   and Grandjean, B. P. A. (2004)  Liquid hold-up
%   correlations for trickle beds without gas flow.
%   Chem. Engng and Processing, 43, pp 85-90.

%   The equation number refers to the work of Thorpe, GR
%   2007, Towards a semi-continuum approach to the design
%   of hydrocoolers for horticultural produce.  Postharvest
%   Biology and Technology. Volume 42, pp 280-289.

%   Graham Thorpe
%   Institute of Sustainability and Innovation
%   Victoria University
%   Melbourne, Australia
%
%   Revised March 2008


g=9.81;
%   Coefficients in Ergun's equation:
E1=150;
E2=1.75;

Re=rhow*uDarcian*dv/(viscw*(1-eps));        %   Reynolds number
Ga=dv^3*g*rhow^2*eps^3/((1-eps)^3*viscw^2); %   Galileo number

term1=E1*etawet^2*Re/(phi^2*Ga);
term2=E2*etawet*Re^2/(phi*Ga);
%   Equation 6
Total_hold_up=eps*(term1+term2)^(1/3);
```

Appendix VI Heat_transfer_coefficientf.m

```
function htc=Heat_transfer_coefficientf(viscw,visca,rhow,rhoa,...
    sigma,cpw,kwater,eps,dv,uDarcian)

%   To calculate heat and mass transfer coefficients in
%   beds of horticultural produce that are being
%   cooled in a hydrocooler.  The heat transfer coefficent
%   is calculated from Larachi, F., Alix, C., Grandjean, B. P. A.
%   and Bernis, A. (2003)  Nu/Sh correlation for particle-liquid
%   heat and mass transfer coefficients in trickle beds based
%   on Péclet similarity.  Chem. Eng. Res. Dev.
%   (Trans IChemE part A), 81, pp 689-694.
%   Note that errata to some the published coefficients
%   have kindly been supplied by the authors.

%   The equation number refers to that in Thorpe GR
%   2007, Towards a semi-continuum approach to the design
%   of hydrocoolers for horticultural produce.  Postharvest
%   Biology and Technology. Volume 42, pp 280-289.

%   Graham Thorpe
%   Victoria University
%   Institute of Sustainability and Innovation
%   Melbourne, Australia
%
%   Modified May 2008


    Mass_diffusivity = 1.0e-5   %   For completeness only
    Thermal_diffusivity=kwater/(rhow*cpw)

%%%%%   Physical properties of the bed of produce   %%%%%

    phi=1       %   Sphericity of horticultural produce
    dv=0.025
    dc=dv*20    %   Diameter of container of the produce
    ag=6/dv     %   Area of sphere per unit volume of solid
    ap=ag*(1-eps)%  Ares of pieces of produce per unit volume of bed

%%%%%   Fluid flow rates   %%%%%%%%%

    uGasDarcian =uDarcian/30

    g=9.81

%%%%%%%%%   Dimensionless groups    %%%%%%%%%

    Re_l_by_g = (rhow*uDarcian*visca)/(rhoa*uGasDarcian*viscw)
    Re_hybrid = (rhow*dv*uGasDarcian)/viscw
    St = uDarcian*viscw/(rhow*g*dv^2)
    Ca = viscw*uDarcian/sigma
    Sb = (1+4/(dc*ap))*(eps/phi)*(384/pi)^(1/3)/(1-eps)^(2/3)
    Pe_mass = dv*uDarcian/Mass_diffusivity
    Pe_heat = dv*uDarcian/Thermal_diffusivity
```

```
%%%%%%%%%   Examine the limits of the dimensionless groups  %%%%%%

    if Re_l_by_g <= 3.43e-3 | Re_l_by_g >= 572
        fprintf(1,'Liquid-gas Reynolds numbers ratio is out of
limits.\n')
        fprintf(1,'Re_l_by_g = %5.3g\n\n',Re_l_by_g)
    end

    if Re_hybrid <= 2 | Re_hybrid >= 1.56e4
        fprintf(1,'Hybrid Reynolds number is out of limits.\n')
        fprintf(1,'Re_hybrid = %5.3g\n\n',Re_hybrid)
    end

    if St <= 3.75e-7 | St >= 5.41e-3
        fprintf(1,'Liquid Stokes number is out of limits.\n')
        fprintf(1,'St = %5.3g\n\n',St)
    end

    if Ca <= 3.8e-6 | Ca >= 6.19e-3
        fprintf(1,'Capillary number ratio is out of limits.\n')
        fprintf(1,'Ca = %5.3g\n\n',Ca)
    end

    if Sb <= 1.35 | Sb >= 9.27
        fprintf(1,'Bed correction factor is out of limits.\n')
        fprintf(1,'Sb = %5.3g\n\n',Sb)
    end

    if Pe_mass <= 27.7 | Pe_mass >= 5.28e5
        fprintf(1,'Mass Peclet number is out of limits.\n')
        fprintf(1,'Pe_mass = %5.3g\n\n',Pe_mass)
    end

    if Pe_heat <= 27.7 | Pe_heat >= 5.28e5
        fprintf(1,'Heat Peclet number is out of limits.\n')
        fprintf(1,'Pe_heat = %5.3g\n\n',Pe_heat)
    end


%%%%%%%%%    Normalised inputs    %%%%%%%%%%

    U(1) = log(Re_l_by_g/(3.43e-3))/(5.22*log(10));
    U(2) = log(Re_hybrid/2)/(3.89*log(10));
    U(3) = log(St/3.75e-7)/(4.16*log(10));
    U(4) = log(Ca/3.8e-6)/(3.21*log(10));
    U(5) = log(Sb/1.35)/(0.837*log(10));
%   Included here for completeness:
    U(6) = log(Pe_mass/27.7)/(4.28*log(10));
    U(6) = log(Pe_heat/27.7)/(4.28*log(10));
    U(7) = 1;

%%%%  Neural network connectivity weights  %%%%

    omega1(1,1)=   7.24694;
    omega1(1,2)=  12.3823;
    omega1(1,3)= -22.6901;
    omega1(1,4)=   2.13869;
    omega1(1,5)= -14.6415;
    omega1(1,6)=   7.43942;
```

```
    omega1(2,1)=   8.51099;
    omega1(2,2)=  32.6348;
    omega1(2,3)= -21.6824;
    omega1(2,4)=   2.03463;
    omega1(2,5)=  -7.84882;
    omega1(2,6)=   2.58508;

    omega1(3,1)= -45.4274;
    omega1(3,2)=   0.558753;
    omega1(3,3)=  24.4283 ;
    omega1(3,4)=   1.07939;
    omega1(3,5)= -39.3043;
    omega1(3,6)= -32.1894;

    omega1(4,1)=  15.5508;
    omega1(4,2)=  32.9063;
    omega1(4,3)=   4.87814;
    omega1(4,4)=  -1.63867;
    omega1(4,5)=  -5.3102;
    omega1(4,6)=  18.7931;

    omega1(5,1)= -50.7208;
    omega1(5,2)= -19.9724;
    omega1(5,3)= -40.9165;
    omega1(5,4)=   0.27428;
    omega1(5,5)= -31.9197;
    omega1(5,6)= -67.3021;

    omega1(6,1)=  29.8193;
    omega1(6,2)=  -5.26075;
    omega1(6,3)= -12.0626;
    omega1(6,4)=   1.54517;
    omega1(6,5)=  68.7597;
    omega1(6,6)=  32.1737;

    omega1(7,1)= -12.931;
    omega1(7,2)= -40.5313;
    omega1(7,3)=   0.384958;
    omega1(7,4)=  -2.76777;
    omega1(7,5)= -17.3021;
    omega1(7,6)= -10.9662;


    omega2(1)=  -6.69033;
    omega2(2)=   0.767127;
    omega2(3)=  -6.94293;
    omega2(4)=   4.90085;
    omega2(5)=   6.4788;
    omega2(6)=   0.591789;
    omega2(7)=  -2.19871;


%%%% Discriminate between Nusselt and Sherwood number %%%%

for ijk = 1:2

    if ijk == 1
        U(6) = log(Pe_mass/27.7)/(4.28*log(10));
```

```
    else
        U(6) = log(Pe_heat/27.7)/(4.28*log(10));
    end


%%%%%%  Normalised output   %%%%%%

    for j=1:6
        sum1 = 0;
            for i=1:7
                sum1 = sum1+omega1(i,j)*U(i);
            end
        H(j) = 1/(1+exp(-sum1));
    end
    H(7) = 1.0;


    sum2=0;

    for j=1:7
        sum2 = sum2 + omega2(j)*H(j);
    end

    S = 1/(1 + exp(-sum2));

%%%%%  Calculate the mass transfer in a hydrocooler  %%%%%

    if ijk == 1
        Sh=0.43495*10^(3.4849*S);
        kls_eta = Mass_diffusivity * Sh/dv;

%%%%%  Calculate the Nusselt number in a hydrocooler  %%%%%

%   Equation 24
    else
        Nu=0.43495*10^(3.4849*S);
        htc = kwater * Nu/dv;
    end

end
```