

APPENDIX 1: EXPERIMENTAL EVIDENCE SHOWING THE ARS WARM-WATER PONDS ARE WELL MIXED

The aerators located approximately 2 m from the inlet were continuously operating, ensuring that the temperature within the pond was uniform. The assumption that the geothermal ponds were well mixed is a necessary condition for using Equation 3.4 and Fishfry. Between 2002 and 2003, the pond temperature was measured at various locations and depths. The temperature, measured with type T thermocouples and recorded with a 21X Campbell Scientific data logger, varied little with respect to position or depth within the pond. For instance, between 04/07/02 and 04/14/02, the temperature in Pond 8 was measured at the aerator, at the discharge pipe and midway between the aerator and the discharge pipe. The temperature, at these locations, was measured 1 foot above the pond bottom and 1 foot below the water surface. The greatest temperature difference measured between any 2 of the 6 thermocouples was 1.6°C. The greatest average temperature difference between any 2 of the 6 thermocouples for the one week period was 0.3°C.

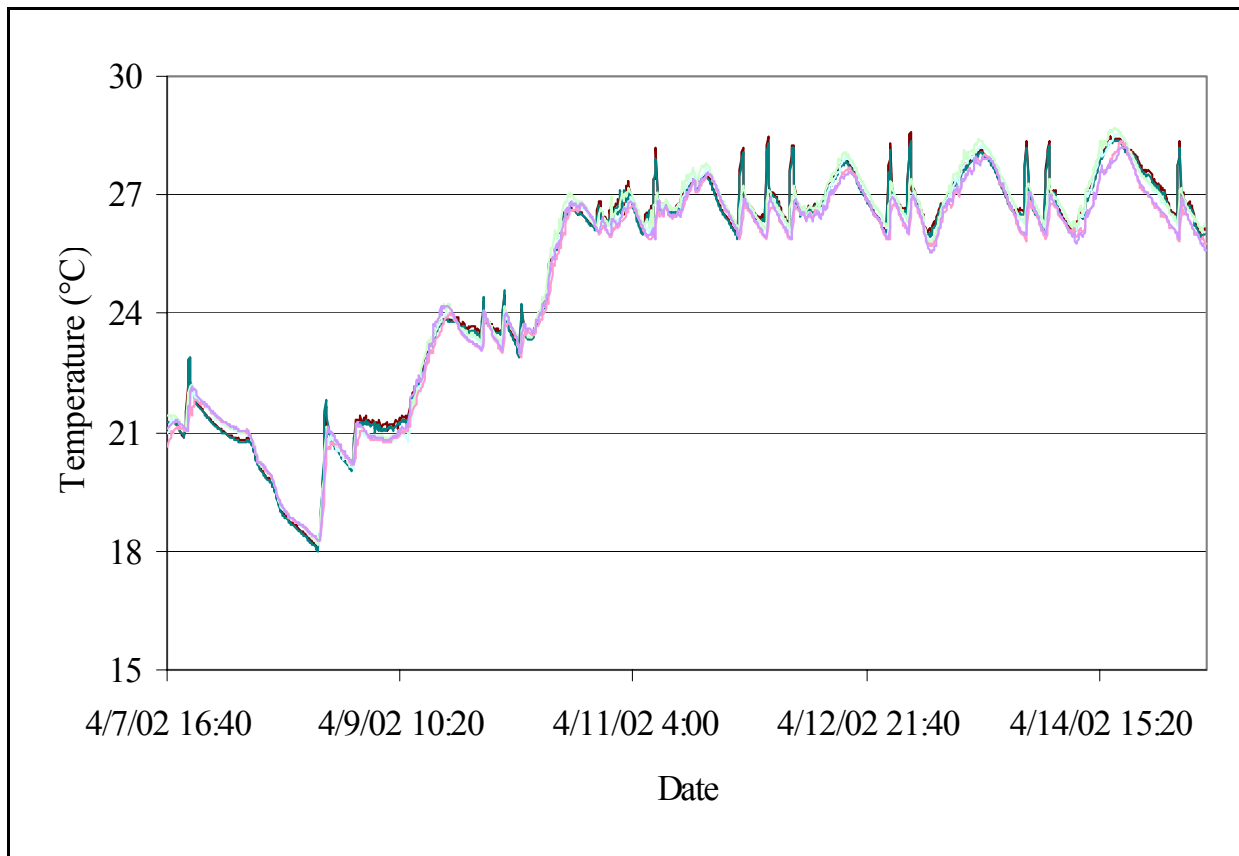


Figure A1.1: The pond temperature was recorded between April 7 and April 14, 2002 in Pond 8. Temperature was measured at the aerator, 13 meters from the aerator and at the pond discharge. At each of location, measurements were taken at 1 foot above the soil surface - pond bottom and 1 foot below the water surface. A total of 6 temperature profiles are shown here. Before April 9, Pond 8 was unheated. The maximum recorded difference between two curves for this time period was 1.6 °C. The maximum average difference between two curves was 0.3°C.

APPENDIX 2: CALCULATING THE SOLAR ZENITH (EXAMPLE)

Example: Determine the solar zenith on the 14th of July at 10:30 A.M. for Baton Rouge, LA (30.53°N, 91.15°W).

The 14th of July is 195th day of the year. Therefore, the solar declination is:

$$\delta = 23.45 \sin \left[\frac{360}{365} (284 + 195) \right] \quad (\text{A2.1})$$

$$\delta = 21.67^\circ \quad (\text{A2.2})$$

Baton Rouge is in the Central Time Zone, where the standard median is at 90° W. If the standard time is 9:30 (remember, 10:30 in July is actually 9:30 standard time), then

$$\omega_{\text{time}} = 9.5 + (90 - 91.15) \div 15 \quad (\text{A2.3})$$

$$\omega_{\text{time}} = 9.42 = 9:25 \text{ A.M.} \quad (\text{A2.4})$$

from which

$$\omega = (12 - 9.42) \times 15^\circ \quad (\text{A2.5})$$

$$\omega = 38.7^\circ \quad (\text{A2.6})$$

and

$$\cos \theta_z = \sin 30.52^\circ \sin 21.67^\circ + \cos 30.52^\circ \cos 21.67^\circ \cos 38.7^\circ \quad (\text{A2.7})$$

$$\cos \theta_z = 0.8123 \quad (\text{A2.8})$$

$$\theta_z = 35.7^\circ \quad (\text{A2.9})$$

APPENDIX 3: CALCULATING THE EMISSIVE WAVELENGTH SPECTRUM OF WATER AT 300 K

The emissive power of a black body is the sum of all the power carried by all radiation emissions at all wavelengths. The emissive power for a given wavelength can be calculated using Planck's Distributive Equation:

$$\frac{E_{b\lambda}}{T^5} = \frac{C_1}{(\lambda T)^5 \left(e^{\left(\frac{C_2}{\lambda T} \right) - 1} \right)} \quad (\text{A3.1})$$

where $E_{b\lambda}$ is the emissive power of a black body for a specific wavelength ($\text{W}/\text{m}^2/\mu\text{m}$)

T is the temperature of the black body (Kelvin)

λ is the wavelength (μm)

$C_1 = 3.743 \times 10^8 \text{ W} \cdot \mu\text{m}^4/\text{m}^2$

$C_2 = 1.4387 \times 10^4 \mu\text{m} \cdot \text{K}$

Integrating this equation for all wavelengths, one obtains the familiar:

$$E_b = \frac{q}{A} = \sigma T^4 \quad (\text{A3.2})$$

which represents the total emitted radiation from a black body.

The Planck distributive equation is a function of the λT term and its solution for different λT is presented in tabulated form (Holman, 1997). The table also shows, for given λT terms, the percentage of the total black body emitted radiation found within that bandwidth. For 1%, $\lambda T = 1444 \mu\text{m} \cdot \text{K}$. For 99%, $\lambda T = 22\,222 \mu\text{m} \cdot \text{K}$. If the temperature of the pond is 27°C , or 300 K, then $\lambda_{1\%} = 4.8 \mu\text{m}$ and $\lambda_{99\%} = 74 \mu\text{m}$, the bandwidth of the radiation emitted by a black body at 300 K.

**APPENDIX 4: JUSTIFYING THE USE OF CONSTANT SURFACE TEMPERATURE
AS AN APPROPRIATE BOUNDARY CONDITION IN DETERMINING THE SOIL
HEAT TRANSFER RATE**

The soil was treated as a semi-infinite homogenous material when solving the heat diffusion equation:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial z^2} \quad (\text{A4.1})$$

where T is the temperature at any time and at any position within the soil,
 α is the soil's thermal diffusivity (a property of the soil) and
 z is depth below the soil surface.

One initial and two boundary conditions were required in solving the heat diffusion equation.
They were:

$$T_{soil}(z, t) = T_{initial} \quad (\text{A4.2})$$

(initial condition)

$$\lim_{z \rightarrow \infty} T_{soil}(z, t) = T_{z=\infty} \quad (\text{A4.3})$$

(first boundary condition)

$$-k_{soil} A \left. \frac{\partial T}{\partial z} \right|_{z=0} = -hA(T_{soil} - T_{pond}) \quad (\text{A4.4})$$

(second boundary condition)

where k is the soil's thermal conductivity,
 A is the area of the soil surface and
 h is the convective heat transfer coefficient.

However, the second boundary condition required the heat transfer coefficient (h) to be known.
Because of this, it was desirable to replace the second boundary condition with the following
boundary condition:

$$T(z = 0, t) \approx T_{pond} \quad (\text{A4.5})$$

(alternate second boundary condition)

This approximation could be made if the heat transfer rate between the pond and the soil was
mainly a function of the thermal properties of the soil. The Biot number (Bi) is a convenient
dimensionless number which compares the resistance to conduction and the resistance to
convection for the soil.

$$Bi = \frac{R_{cond}}{R_{conv}} = \frac{hL_{char}}{k_{soil}} \quad (A4.6)$$

where R_{cond} is the thermal resistance to conduction
 R_{conv} is the thermal resistance to convection
 h is the heat transfer coefficient of the water (W/m²/K)
 L_{char} is the characteristic length of the soil (m)
 k_{soil} is the thermal conductivity of the soil (W/m/K)

The characteristic length of the soil was chosen to be the dampening depth (D) with $\omega = 1.157 \times 10^{-5} \text{ s}^{-1}$ (equation 3.39). The dampening depth for clay is 0.122 m (Van Wijk and De Vries, 1966). From Table 3.7, the thermal conductivity of the soil was estimated as 2.92 W/m/K. If the source of resistance to heat transfer is the soil's conductance, the Biot number will be large (30). So the heat transfer coefficient is:

$$h = \frac{100 \times (2.92 \text{ W} / \text{m} / \text{K})}{0.122 \text{ m}} = 2393 \text{ W} / \text{m}^2 / \text{K} \quad (A4.7)$$

To relate this heat transfer coefficient as a velocity, Nusselt, Prandtl and Reynolds numbers were used. The Nusselt number (Nu) is (equation 3.48):

$$Nu = \frac{hL_{char}}{k_{water}} = \frac{(2393 \text{ W} / \text{m}^2 / \text{K})(3.75 \text{ m})}{0.614 \text{ W} / \text{m} / \text{K}} = 14615 \quad (A4.8)$$

Note here that the characteristic length (L_{char}) is different, and was calculated from Equation 3.49. Assuming mixed laminar and turbulent conditions were present, Equation 3.54 was used to determine the Reynolds number (Re) (the Prandtl number (Pr) is 5.85 when the water temperature is at 26°C; Holman, 1997):

$$Re = \left(\frac{Nu}{\frac{Pr^{1/3}}{0.037} + 871} \right)^{5/4} = 5388535 \quad (A4.9)$$

Finding the velocity using Equation 3.55 yielded:

$$V = \frac{Re \mu}{\rho L_{char}} = \frac{5388535 (8.6 \times 10^{-4} \text{ kg} / \text{m} / \text{s})}{(995.8 \text{ kg} / \text{m}^3)(3.75 \text{ m})} = 1.24 \text{ m} / \text{s} \quad (A4.10)$$

If the velocity of the water at the soil surface is faster than 0.36 m/s, then the Biot number is large enough to assume that resistance to conduction is limiting the overall heat transfer rate.

APPENDIX 5: DETERMINING THE DAILY PHASE ANGLE AT THE SOIL SURFACE

The second boundary condition in Equation 3.37 gives the soil temperature profile at the surface with respect to time. Daily variations in soil temperature were represented by the second term:

$$T_{amp-day} \sin(\omega_{day}t) \quad (A5.1)$$

which can be rewritten as

$$T_{amp-day} \sin\left(2\pi \frac{t}{86400}\right) \quad (A5.2)$$

where $T_{amp-day}$ is the amplitude of the sinusoidal curve describing soil temperature at the surface ($^{\circ}\text{C}$),
 t is time in seconds.

The more general form of this equation is:

$$T_{amp-day} \sin\left(2\pi \frac{t}{86400} + \phi\right) \quad (A5.3)$$

where ϕ is the daily phase angle.

Equation 3.37 assumed that the phase angle is 0 and Fishfray assumes the phase angle is $\pi/2$, both of which are not true for the case of a soil surface submerged under 1.2 meters of water. Assuming that the maximum daily pond temperature occurred at $t = 16:00$ ($t = 57600$ seconds), and that the angle inside the sine function must be $\pi/2$ at 16:00, then the daily phase angle must be $-5\pi/6$.

APPENDIX 6: THEORY BEHIND THE DETERMINATION OF TRANSPORT COEFFICIENTS - ANALYTICAL METHOD

(Source: Incropera and De Witt, 1985. Fundamentals of Heat and Mass Transfer 2nd edition , page 284.)

The theory relating the convective heat transfer coefficient (h) and the convective mass transfer coefficient (h_m) relies on boundary layer theory. Both the concentration and thermal boundary layers are similar in shape and are determined by the velocity boundary layer. Both use the diffusion equation and both can be calculated using analogous empirical equations.

$$Nu = f_1(Re, Pr) \quad (A6.1a)$$

$$Sh = f_2(Re, Sc) \quad (A6.1b)$$

Normally, Pr and Sc are related to Nu and Sh by powers of some quantity n .

$$Nu = f_1(Re)Pr^n \quad (A6.2a)$$

$$Sh = f_2(Re)Sc^n \quad (A6.2b)$$

The Reynolds number (Re) is the same for both functions because both functions are subjected to the same geometry,

$$f_1(Re) = f_2(Re) \quad (A6.3)$$

so...

$$\frac{Nu}{Pr^n} = \frac{Sh}{Sc^n} \Rightarrow \frac{Nu}{Sh} = \frac{Pr^n}{Sc^n} = Le^{-n} \quad (A6.4)$$

and...

$$\frac{Nu}{Sh} = \frac{\frac{hL_c}{k_{fluid}}}{\frac{h_m L_c}{D_{AB}}} = \frac{h}{h_m} \frac{D_{AB}}{k_{fluid}} = Le^{-n} \quad (A6.5)$$

By definition, the Lewis number is thermal diffusivity divided by the mass diffusivity.

$$Le = \frac{\alpha}{D_{AB}} = \frac{k_{fluid}}{\rho_{fluid} c_{p-fluid} D_{AB}} \quad (A6.6)$$

Rearranging the terms gives:

$$\frac{D_{AB}}{k_{fluid}} = (\rho_{fluid} c_{p-fluid} Le)^{-1} \quad (A6.7)$$

$$h = h_m (\rho c_p) Le^{1-n}$$

isolating for h_m gives Equation 5.1:

Substituting A6.7 into equation A6.5, and

$$(A6.8)$$

APPENDIX 7: DERIVATION OF EQUATION 5.3

Given Equation 3.4:

$$\left(\frac{dE}{dt}\right)_{pond} = q_{solar} - q_{back} + q_{sky} \pm q_{conv} - q_{evap} \pm q_{soil} + q_{rain} - q_{seep} + q_{well} - q_{out} \quad (A7.1)$$

and given Equation 5.1

$$h = h_m (\rho c_p) Le^{1-n} \quad (A7.2)$$

one can solve for h_m . Consider Fick's law of diffusion:

$$\dot{m}_{water} = h_m A (C_{water} - C_{air}) \quad (A7.3)$$

where \dot{m}_{water} is the evaporation rate,
 h_m is the convective mass transfer coefficient,
 A is the area,
 C_{water} is the concentration of water vapour at the pond surface and
 C_{air} is the concentration of water vapour in the air.

Using the perfect gas law, Equation A7.3 can be rewritten as

$$\dot{m}_{water} = h_m A \left(\frac{MM}{R}\right) \left(\left(\frac{P}{T}\right)_{water} - \left(\frac{P}{T}\right)_{air} \right) \quad (A7.4)$$

where MM is the molar mass of water,
 R is the universal gas constant,
 P is the pressure of the water vapour and
 T is the absolute temperature.

The energy lost to evaporation is:

$$q_{evap} = h_{fg} \dot{m} = h_{fg} h_m A \left(\frac{MM}{R}\right) \left(\left(\frac{P}{T}\right)_{water} - \left(\frac{P}{T}\right)_{air} \right) \quad (A7.5)$$

Newton's law of cooling is used to predict the heat lost by convection.

$$q_{conv} = hA(T_{air} - T_{pond}) \quad (A7.6)$$

Using Equation A7.2, Equation A.6 becomes

$$q_{conv} = h_m (\rho c_p)_{air} L e^{1-n} A (T_{air} - T_{water}) \quad (A7.7)$$

Substituting A7.5 and A7.7 into A7.1, and then isolating h_m , one gets Equation 5.3

$$h_m = \frac{\left(\frac{dE}{dt} \right)_{pond} - (q_{solar} - q_{back} + q_{sky} \pm q_{soil} - q_{seep} + q_{rain} + q_{well} - q_{out})}{A \left(L e^{2/3} \rho_{air} c_{p-air} (T_{air} - T_{pond}) - h_{fg} \frac{MM}{R} \left(\left(\frac{P}{T} \right)_{surface} - \left(\frac{P}{T} \right)_{air} \right) \right)} \quad (A7.8)$$

APPENDIX 8: PYRANOMETER INFORMATION

A pyranometer sensor (LI-COR LI-200SZ) was used to measure the solar radiation below the water surface. The sensor, shown below, had a coaxial cable with the inner cable being the positive lead and the outer cable being the negative lead. Because the pyranometer produced a current signal, it was necessary to use a 147-Ohm resistor to convert this signal into a voltage signal (schematic in Figure A8.2).

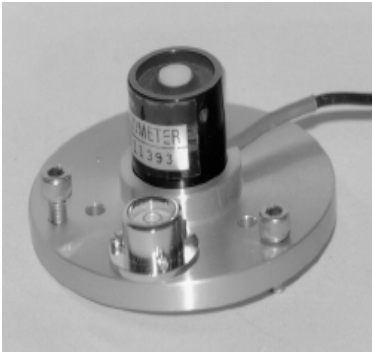


Figure A8.1: A pyranometer, similar to this one, was used to measure solar radiation under water.

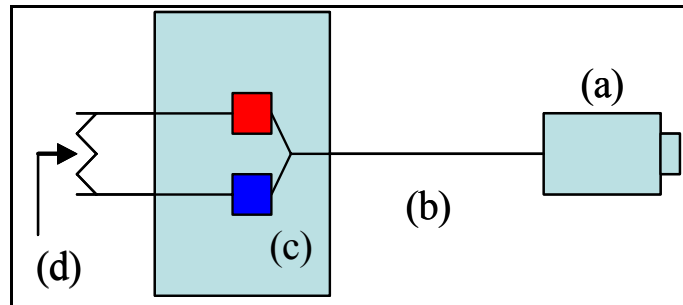


Figure A8.2: The pyranometer (a) was connected to the data logger (c) with the use of a coaxial wire (b). The inner wire was connected to the high lead while the outer shield was connected to the low lead. A 147 Ohm resistor (d) was also connected across the data logger terminal.

A data logger (Campbell Scientific 21X, Campbell Scientific Inc., North Logan, UT) was used as an output device so that measurements could be read. The following program was entered into the data logger. Measurements were recorded so that the LI-COR pyranometer could be calibrated to an Eppley Radiometer (model PSP).

```

;{21X}
;
*Table 1 Program
01: 1      Execution Interval (seconds)

1: Volt (Diff) (P2)
1: 1      Reps
2: 3      50 mV Slow Range
3: 2      DIFF Channel
4: 1      Loc [ small ]
5: 99.8   Mult
6: 0.0    Offset
    
```

2: Volt (Diff) (P2)

1: 1 Reps
2: 3 50 mV Slow Range
3: 3 DIFF Channel
4: 2 Loc [big]
5: 115.7 Mult
6: 0.0 Offset

3: If time is (P92)

1: 0 Minutes into a
2: 1 Minute Interval
3: 10 Set Output Flag High

4: Real Time (P77)

1: 1220 Year,Day,Hour/Minute (midnight = 2400)

5: Sample (P70)

1: 2 Reps
2: 1 Loc [small]

6: Batt Voltage (P10)

1: 3 Loc [batery]

*Table 2 Program

02: 0 Execution Interval (seconds)

*Table 3 Subroutines

End Program

APPENDIX 9: EXAMPLE CALCULATIONS SHOWING THE NEED FOR SENSITIVE TEMPERATURE SENSING DEVICES IN CHAPTER 5.5.1

Consider a pond 400 m³ in volume, subjected to a temperature change of 0.01°C over 10 minutes (600 seconds). Such a change in temperature represents a change in energy of

$$\begin{aligned} \left(\frac{\Delta E}{\Delta t} \right)_{pond} &= \rho \forall c_p \frac{\Delta T}{\Delta t} \\ &= (990 \text{ kg} / \text{m}^3)(400 \text{ m}^3)(4180 \text{ J} / \text{kg} / \text{K}) \left(\frac{0.01^\circ \text{C}}{600 \text{ s}} \right) \\ &= 27588 \text{ W} \end{aligned}$$

The average long wave sky radiation between the 13th of February and the 23rd of March, 2003 for unheated Pond 3 was 110 000W. Therefore, an error of 0.04°C for a difference of two successive pond temperature readings is equivalent in magnitude to long wave sky radiation.

APPENDIX 10: DATA LOGGER PROGRAMS

The following basic program was used to measure the pond temperature a data logger (Campbell Scientific 21X, Campbell Scientific Inc., North Logan, UT). The multiplier in Instruction P:14(7) and the offset in Instruction P:14(8) were used to calibrate the thermocouple readings, so as to avoid additional manipulations during the analysis of data.

```

;{21X}
;
*Table 1 Program
  01: 60      Execution Interval (seconds)
  1: Internal Temperature (P17)
    1: 1      Loc [ ref_temp ]

  2: Thermocouple Temp (DIFF) (P14)
    1: 1      Reps
    2: 1      5 mV Slow Range
    3: 1      DIFF Channel
    4: 1      Type T (Copper-Constantan)
    5: 1      Ref Temp (Deg. C) Loc [ ref_temp ]
    6: 2      Loc [ soil_far ]
    7: 1.008  Mult
    8: -.016  Offset

  3: Thermocouple Temp (DIFF) (P14)
    1: 1      Reps
    2: 1      5 mV Slow Range
    3: 2      DIFF Channel
    4: 1      Type T (Copper-Constantan)
    5: 1      Ref Temp (Deg. C) Loc [ ref_temp ]
    6: 3      Loc [ bot_far ]
    7: 1.008  Mult
    8: -.016  Offset

  4: Thermocouple Temp (DIFF) (P14)
    1: 1      Reps
    2: 1      5 mV Slow Range
    3: 3      DIFF Channel
    4: 1      Type T (Copper-Constantan)
    5: 1      Ref Temp (Deg. C) Loc [ ref_temp ]
    6: 4      Loc [ high_far ]
    7: 1.020  Mult
    8: -.136  Offset
```

5: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 4 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 5 Loc [sur_far]
7: 1.008 Mult
8: -.016 Offset

6: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 5 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 6 Loc [soil_near]
7: 1.005 Mult
8: .034 Offset

7: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 6 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 7 Loc [bot_near]
7: 1.010 Mult
8: 0 Offset

8: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 7 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 8 Loc [high_near]
7: 1.008 Mult
8: 0.017 Offset

9: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 8 DIFF Channel

```

4: 1    Type T (Copper-Constantan)
5: 1    Ref Temp (Deg. C) Loc [ ref_temp ]
6: 9    Loc [ sur_near ]

7: 1.005  Mult
8: 0.101  Offset

10: Batt Voltage (P10)
1: 10    Loc [ battery ]

11: If time is (P92)
1: 0     Minutes into a
2: 10    Minute Interval
3: 10    Set Output Flag High

12: Real Time (P77)
1: 1220  Year,Day,Hour/Minute (midnight = 2400)

13: Sample (P70)
1: 10    Reps
2: 1     Loc [ ref_temp ]

*Table 2 Program
02: 0.0000  Execution Interval (seconds)

*Table 3 Subroutines

End Program

```

If in addition to 8 thermocouples an anemometer is being used, the following program should be entered into the data logger.

```

*Table 1 Program
01: 60     Execution Interval (seconds)
1: Internal Temperature (P17)
1: 1     Loc [ ref_temp ]

2: Thermocouple Temp (DIFF) (P14)
1: 1     Reps
2: 1     5 mV Slow Range
3: 1     DIFF Channel
4: 1     Type T (Copper-Constantan)
5: 1     Ref Temp (Deg. C) Loc [ ref_temp ]

```

6: 2 Loc [soil_far]
7: 0.988 Mult
8: .544 Offset

3: Thermocouple Temp (DIFF) (P14)
1: 1 Reps
2: 1 5 mV Slow Range
3: 2 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 3 Loc [bot_far]
7: 1.002 Mult
8: .154 Offset

4: Thermocouple Temp (DIFF) (P14)
1: 1 Reps
2: 1 5 mV Slow Range
3: 3 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 4 Loc [high_far]
7: 1.004 Mult
8: .385 Offset

5: Thermocouple Temp (DIFF) (P14)
1: 1 Reps
2: 1 5 mV Slow Range
3: 4 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 5 Loc [sur_far]
7: .997 Mult
8: .25 Offset

6: Thermocouple Temp (DIFF) (P14)
1: 1 Reps
2: 1 5 mV Slow Range
3: 5 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 6 Loc [soil_near]
7: 1.033 Mult
8: -.905 Offset

7: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 6 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 7 Loc [bot_near]

7: 1.028 Mult
8: -.834 Offset

8: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 7 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 8 Loc [high_near]
7: 1.013 Mult
8: -.218 Offset

9: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 8 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 9 Loc [sur_near]
7: .988 Mult
8: 1.265 Offset

10: Batt Voltage (P10)
1: 10 Loc [battery]

11: Pulse (P3)
1: 1 Repts
2: 2 Pulse Input Channel
3: 21 Low Level AC, Output Hz
4: 11 Loc [wind]
5: 0.75 Mult
6: .2 Offset

12: If time is (P92)
1: 0 Minutes into a

```

2: 10    Minute Interval
3: 10    Set Output Flag High

13: Real Time (P77)
1: 1220   Year,Day,Hour/Minute (midnight = 2400)

14: Sample (P70)
1: 10    Reps
2: 1     Loc [ ref_temp ]

15: Average (P71)
1: 1     Reps
2: 11    Loc [ wind    ]

*Table 2 Program
02: 0.0000 Execution Interval (seconds)

*Table 3 Subroutines

End Program

```

If, in addition to 8 thermocouples a weather vane is being use, the following program should be entered into the datalogger:

```

;{21X}
;
*Table 1 Program
01: 60    Execution Interval (seconds)
1: Internal Temperature (P17)
1: 1     Loc [ ref_temp ]

2: Thermocouple Temp (DIFF) (P14)
1: 1     Reps
2: 1     5 mV Slow Range
3: 1     DIFF Channel
4: 1     Type T (Copper-Constantan)
5: 1     Ref Temp (Deg. C) Loc [ ref_temp ]
6: 2     Loc [ soil_far ]
7: 1.01  Mult
8: -.269 Offset

3: Thermocouple Temp (DIFF) (P14)
1: 1     Reps
2: 1     5 mV Slow Range

```

3: 2 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 3 Loc [bot_far]
7: 1.008 Mult
8: -.185 Offset

4: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 3 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 4 Loc [high_far]
7: 1.027 Mult
8: -.635 Offset

5: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 4 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 5 Loc [sur_far]
7: 1.01 Mult
8: -.236 Offset

6: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 5 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 6 Loc [soil_near]
7: 1.01 Mult
8: -.202 Offset

7: Thermocouple Temp (DIFF) (P14)
1: 1 Repts
2: 1 5 mV Slow Range
3: 6 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 7 Loc [bot_near]

7: 1.013 Mult
8: -.219 Offset

8: Thermocouple Temp (DIFF) (P14)

1: 1 Repts
2: 1 5 mV Slow Range
3: 7 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 8 Loc [high_near]
7: 1.01 Mult
8: -.202 Offset

9: Thermocouple Temp (DIFF) (P14)

1: 1 Repts
2: 1 5 mV Slow Range
3: 8 DIFF Channel
4: 1 Type T (Copper-Constantan)
5: 1 Ref Temp (Deg. C) Loc [ref_temp]
6: 9 Loc [sur_near]
7: 1.046 Mult
8: -.508 Offset

10: Batt Voltage (P10)

1: 10 Loc [battery]

11: Excite Delay Volt (SE) (P4)

1: 1 Repts
2: 5 5000 mV Slow Range
3: 1 SE Channel
4: 1 Excite all reps w/Exchan 1
5: 2 Delay (units 0.01 sec)
6: 5000 mV Excitation
7: 11 Loc [winddir]
8: 0.071 Mult
9: 0.0 Offset

12: If time is (P92)

1: 0 Minutes into a
2: 10 Minute Interval
3: 10 Set Output Flag High

13: Real Time (P77)

1: 1220 Year,Day,Hour/Minute (midnight = 2400)

14: Sample (P70)

1: 10 Reps

2: 1 Loc [ref_temp]

15: Average (P71)

1: 1 Reps

2: 11 Loc [winddir]

*Table 2 Program

02: 0.0000 Execution Interval (seconds)

*Table 3 Subroutines

End Program

APPENDIX 11: HOW TO USE FISHFRY

To use Fishfry, the following 4 ASCII data files must be available to the executive program in the same directory:

- **information.dat:** lists all the soil and pond parameters as well as the initial conditions. The file must have the following layout, although the numeric values can change.

3600.0000000000	step size (s)
31104000.0000000000	Initial time (s)
17.0000000000	Pond temperature (°C)
33609600.0000000000	Upper time limit (s)
480.0000000000	Pond Volume (m ³)
394.0000000000	Pond area (m ²)
990.0000000000	Water density (kg/m ³)
4180.0000000000	Specific heat of water (J/kg°C)
0.9599999785	Water emissivity (decimal)
0.0000005500	thermal diffusivity (m ² /s)
1.0000000000	soil density (kg/m ³)
2900000.0953674316	specific heat (J/kg°C)
5.6999998093	soiltempamp_year (°C)
5.0000000000	soiltempamp_day (°C)
1.5707963705	phase_year (radians)
1.5707963705	phase_day (radians)

Figure A11.1: The required layout for the information.dat file.

- **weather.dat:** lists all the weather data for a certain time step specified in information.dat. This ASCII file contains 8 columns. The first 3 are arbitrary numbers (usually the year, day of the year and time). The fourth column is for the air temperature (°C), the fifth is for relative humidity (a number between 1 and 100), the sixth is for solar radiation (kW/m²), the seventh is for wind speed (m/s), the eighth is for rainfall (inches/hr).
- **flow.dat:** this is a 5 column ASCII file. The first 3 are arbitrary numbers (usually the year, day of the year and time). The fourth column lists the flow rate of warm water into the pond. The fifth column lists the flow rate of cold water into the pond.
- **temp.dat:** this is a 1 column ASCII file which lists the pond temperature (°C). This is for cases when the user is interested in knowing the flow rate of water

needed to maintain his pond temperature at a certain value (used only for option 1).

Once these files are available in the home directory, the user can run Fishfry.exe. After starting the program, the following prompt will appear:

Welcome to FISHFRY, the easy way to know how much heat you need to supply to the pond so that your fish won't freeze or cook.

Please remember to include the following files in the FISHFRY directory/folder before running the program:

- information.dat
- weather.dat
- flow.dat
- temp.dat

The output file will be called output.dat, where columns are space separated, and easily viewed in your favorite spreadsheet.

Press any number to continue

What do you want FISHFRY to do?
Please enter the appropriate number.

- (1) Determine the flow rate of water into the pond to maintain a desired pond temperature.
- (2) Determine the temperature of a pond given the flow rate and weather.
- (0) End FISHFRY, the best program in the whole wide world.

Figure A11.2: The Fishfry User Prompt is accessed from DOS.

Once the user has selected one of these choice, the model will prompt the user for a name for the output file, the initial pond temperature (°C), the temperature of the warm water at the inlet (°C), the temperature of the cold water at the inlet (°C), the name of the weather file and the name of the pond temperature file. If no warm or cold water is being used, enter 0 when asked for the temperature.

Fishfry will then compile results. Sometimes, Fishfry will give out an error statement at the end of a model run. This happened because there was not enough input information for the defined

time period (defined in information.dat). The user must make sure the number of input data in a file is equal to the number of time steps.

Once the program is finished, the user can open the output file in a spreadsheet program as a ASCII file with comma separated variables.

APPENDIX 12: PROGRAM CODE FOR FISHFRY

```
PROGRAM fishfry
IMPLICIT NONE
```

```
=====
! INTRODUCTION
!-----
!
! Fishfry v 1.2 is the master version of Fishfry which includes some
! some of the suggested modifications made in chapter 4. Although
! equations for the effects of soil heat transfer are present, they
! have been made idle due to their small contribution to the overall
! pond energy balance.
!
! This version of Fishfry has two modes. The first is more practice
! for managing the supply of cool and warm water. For a given pond
! temperature, which can be made to vary in the temp.dat input file,
! Fishfry will calculate the amount of cold and warm water required to
! keep the pond at that given temperature.
!
! The second Fishfry mode is more useful for validating the model,
! although it can be used as a management tool. Fishfry, in mode 2,
! will solve for the pond temperature given all the energy fluxes.
!
!-----
! FUNCTIONS
!-----
! dTemp - Differential Equation
!
! The following interface evaluates dTemp for a given temp and time, using
! the 4th order Runge Kutta numerical method
!
Interface
FUNCTION dTemp(Temp,time,coefficient,thermal_mass, airtemp) !step)
IMPLICIT NONE
    REAL (8), INTENT (IN):: Temp, time, thermal_mass, airtemp !, step
    REAL (8), dimension (10), INTENT (IN):: coefficient
    REAL (8):: dTemp
END function dTemp
END interface

!-----
! emissivity_sky
```

```
!  
! The following interface evaluates the emissivity of a clear sky.
```

```
Interface  
Function emissivity_sky(airtemp,rh)  
Implicit NONE
```

```
REAL (8), INTENT(IN):: airtemp, rh  
rEAL (8):: emissivity_sky
```

```
END function emissivity_sky  
end interface
```

```
!-----  
! vapour pressure  
!  
! The following interface is for a function which evaluates the vapour  
! pressure in Pascals.  
!
```

```
Interface  
FUNCTION vp(T, rh)  
IMPLICIT NONE  
REAL (8), INTENT (IN):: T, rh  
REAL (8):: vp  
eND function vp  
end Interface
```

```
!-----  
! Wind speed correction function - REQUIRES AN INPUT!  
!  
! The following interface is for a function which corrects wind speed for  
! the Lake Hefner equation. It also converts units from m/s to mph.  
!
```

```
Interface  
function ws(windspeed)  
implicit none  
REAL (8), INTENT (IN):: windspeed  
REAL (8):: ws  
end function ws  
END interface
```

```
!
```

```

=====
! VARIABLE DECLARATIONS
!-----
!      Runge Kutta variables

REAL (8):: F1, F2, F3, F4
REAL (8):: Temp, time, step, timemax

!      Temp: pond temperature (C)
!      time: time (sec)
!      step: time step (sec)
!      timemax: upper integration limit (sec)
!      F1, F2, F3, F4: integration formulae for Runge-Kutta
!
!-----
!      Pond parameters

REAL(8):: volume, area, density, specheat, Thermal_mass, emissivity_water

!      Volume: pond volume (m^3)
!      Area: pond surface area (m^2)
!      density: density of pond water (kg/m^3)
!      specheat: specific heat of water (J/kg/C)
!      Thermal_mass: Volume*density*specific heat (J/C)
!      emissivity_water: emissivity of the water
!
!-----
!      Flux parameters

REAL(8):: latent_energy
REAL(8):: flow_rate_hot, flow_rate_cold, gpm_hot, gpm_cold, hot, cold

!      latent_energy: energy for the vapourization of water
!      hot: the temperature of the warm influent (C)
!      cold: the temperature of the cold influent (C)
!      flow_rate_hot: the flow rate of the warm effluent (m^3/s)
!      flow_rate_cold: the flow rate of the cold effluent (m^3/s)
!      gpm_hot:: the flow rate of the warm effluent (gpm)
!      gpm_cold: the flow rate of the cold effluent (gpm)
!
!-----
!      Atmospheric parameters

```

REAL (8):: airtemp, rh, windspeed, solar_radiation, x, rain!, winddirection

! emissivity_sky: emissivity in the sky (decimal)
! airtemp: air temperature (C)
! rh: relative humidity (demical)
! solar_radiation: solar_radiation (J/m^2/s)
! windspeed: wind speed (mph)

! wind direction: degrees, 0 = North
! x: relative humidity of 100% (1)
!

!-----

! Soil parameters

!

!Real(8):: thermal_conductivity, thermal_diffusivity, soil_density

!REAL(8):: cp_soil, Dday, Dyear, temp_initial, PI

!REAL(8):: Soiltempamp_year, Soiltempamp_day, phase_year, phase_day

!

! thermal conductivity: J/m/C/s (Fourier's heat conduction law)

! thermal diffusivity: m^2/s (Heat diffusion equation)

! Soil_density: kg/m^3

! cp_soil: Soil specific heat (J/kg°C)

! Dday: dampening depth for daily variations (m)

! Dyear: dampening depth for yearly variations (m)

! temp_initial: initial soil temperature (°C)

! Soiltempamp_year: The yearly amplitude for soil temperature (°C)

! Soiltempamp_day: The daily amplitude for soil temperature (°C)

! phase_year: phase angle for the year

! phase_day: phase angle for the day

! PI

!

!-----

! Coefficient arrays

!

REAL(8), dimension (10):: coefficient, vector

! Coefficient (1): coefficient for pond longwave radiation
! Coefficient (2): coefficient for atmospheric longwave radiation
! Coefficient (3): coefficient for soil heat exchange
! Coefficient (4): coefficient for surface evaporation
! Coefficient (5): coefficient for surface convection
! Coefficient (6): coefficient for penetrated solar radiation

```

!   Coefficient (7): coefficient for warm water inlet

!   Coefficient (8): coefficient for cold water inlet
!   Coefficient (9): coefficient for discharge
!
!   Vector (1): Power due to pond longwave radiation
!   Vector (2): Power due to atmospheric longwave radiation
!   Vector (3): Power due to soil heat exchange
!   Vector (4): Power due to surface evaporation
!   Vector (5): Power due to surface convection
!   Vector (6): Power due to solar radiation
!   Vector (7): Energy flow in the hot water
!   Vector (8): Energy flow in the cold water
!   Vector (9): Energy flow in the discharge
!
!   All vector terms are in Joules/second (Watts)
!
!-----
!       Other parameters
!

```

```

integer:: choice
CHARACTER(LEN=12):: weather_file, temp_file, pond
!
!=====
! FILES AND FORMAT
!-----

```

```

OPEN (99, FILE="information.dat")  ! Information input data file
!OPEN (100, FILE="output.dat")     ! Output data file
!OPEN (200, FILE="weather.dat")    ! Input air temperature data file
OPEN (300, FILE="flow.dat")        ! Flow input
!OPEN (400, FILE="temp.dat")       ! Pond Temperature input

```

```

10  FORMAT (20(f30.6, ","), f30.6)
20  FORMAT (20(a30, ","), a30)

```

```

!=====
! SETTING OF PARAMETERS (USER REQUIRED INPUT HERE)
!-----
!   Initial conditions

```

```

READ (99,*) step      ! Step size (s)
Read (99,*) time     ! Initial time(s)
Read (99,*) temp     ! Initial pond water temperature (C)
Read (99,*) timemax  ! Upper limit for time (s)

```

```
!
```

```
!-----
```

```
!   Properties of the pond
```

```

Read (99,*) Volume   ! in m^3
Read (99,*) Area     ! in m^2

Read (99,*) Density  ! in kg/m^3
Read (99,*) Speheat  ! in J/kg/C

```

```
Thermal_mass = Volume*Density*Speheat
```

```

Read (99,*) Emissivity_water ! decimal
x = 1.0          ! constant (no input here)
!PI=ACOS(-1.0)   ! the value of pi

```

```
!-----
```

```
!   Properties of the liner (soil)
```

```
!
```

```

!read (99,*) thermal_diffusivity ! soil thermal diffusivity (m^2/s)
!read (99,*) soil_density         ! bulk soil density (kg/m^3)
!read (99,*) cp_soil              ! soil specific heat (J/kg°C)
!read (99,*) Soiltempamp_year     ! Yearly soil amplitude (°C)
!read (99,*) Soiltempamp_day      ! Daily soil amplitude (°C)
!read (99,*) phase_year           ! Phase angle, year (radians)
!read (99,*) phase_day            ! Phase angle, day (radians)

```

```

!temp_initial=temp
!Thermal_conductivity=thermal_diffusivity*(soil_density*cp_soil)
!Dday=SQRT(2.0*thermal_conductivity/soil_density/cp_soil*86400.0/2/PI)
!Dyear=SQRT(2.0*thermal_conductivity/soil_density/cp_soil*31536000.0/2/PI)

```

```
!
```

```
!=====
```

```
!   User prompt to determine if the pond is at steady state or transient.
```

```

!
write (*,*) "Welcome to FISHFRY, the easy way to know how much heat you need"
write (*,*) "to supply to the pond so that your fish won't freeze or cook."
write (*,*) " "
write (*,*) "Please remember to include the following files in the FISHFRY"
write (*,*) "directory/folder before running the program: "
write (*,*) " "
write (*,*) "          - information.dat "
write (*,*) "          - weather.dat"
write (*,*) "          - flow.dat"
write (*,*) "          - temp.dat"
write (*,*) " "
write (*,*) "The output file will be called output.dat, where columns are space"
write (*,*) "seperated, and easily viewed in your favorite spreadsheet."
write (*,*) " "
write (*,*) "Press any number to continue"
write (*,*) " "
READ (*,*) choice

```

```

111 write (*,*) "What do you want FISHFRY to do?"
write (*,*) "Please enter the appropriate number."
write (*,*) " "
write (*,*) "(1) Determine the flow rate of water into the pond to maintain"
write (*,*) "    a desired pond temperature."
write (*,*) " "
write (*,*) "(2) Determine the temperature of a pond given the flow rate"
write (*,*) "    and weather."
write (*,*) " "
write (*,*) "(0) End FISHFRY, the best program in the whole wide world."
write (*,*) " "
read (*,*) choice

```

```

IF (choice>2) THEN
write (*,*) "You have made an invalid choice."
GO TO 111
END IF

```

```

!=====
!   SETTING INITIAL OR BOUNDARY CONDITIONS
!

```

```

write (*,*) "Which pond is being considered?"
write (*,*) " "
read (*,*) pond

```

```
write (*,*) "What is the temperature of the warm influent? (C)"
write (*,*) " "
read (*,*) hot
```

```
write (*,*) "What is the temperature of the cool influent? (C)"
write (*,*) " "
read (*,*) cold
```

```
IF (choice==2) THEN
```

```
write (*,*) "What is the initial pond temperature?"
write (*,*) " "
read (*,*) temp
```

```
END IF
```

```
IF (choice==0) GO TO 9
```

```
WRITE (*,*) "What is the name of the weather file? "
READ (*,*) weather_file
WRITE (*,*) "What is the name of the temp file? "
READ (*,*) temp_file
```

```
OPEN (200, FILE=weather_file)    ! Input air temperature data file
OPEN (400, FILE=temp_file)      ! Flow input
open (100, FILE=pond)
```

```
!=====
```

```
! HEADINGS FOR OUTPUT FILES
```

```
!
```

```
!
```

```
WRITE (100, *) "Results from Fishfry; OUTPUT.DAT"
WRITE (100, *) " "
WRITE (100, *) "Pond number: ", pond
WRITE (100, *) " "
WRITE (100, *) "Initial day of trial: ", time/24/3600+1
WRITE (100, *) "Final day of trial: ", timemax/24/3600+1
WRITE (100, *) " "
```

```
WRITE (100,*) "Influent hot temperature (C): ", hot, "Type of year:", weather_file
WRITE (100,*) "Influent cold temperature (C): ", cold, "Pond Temperature:", temp_file
WRITE (100,*) " "
```

!
! Case 1: Determination of flowrates
!

IF (choice==1) THEN

WRITE (100,20) "Time", " ", "Pond Radiation", "Sky Radiation", "Soil Conduction", &
& "Evaporation", "Convection", "Solar Radiation", "Rain", "Rain out",&
& "Surplus/deficit", "Warm Water Flow", "Warm Water Flow",&
& "Cool Water Flow", "Cool Water Flow"

WRITE (100,20) "(seconds)", "(days)", "(Watts)", "(Watts)", "(Watts)", "(Watts)", "(Watts)",&
& "(Watts)", "(Watts)", "(Watts)", "(Whr)", "(m^3/s)", "(GPM)",&
& "(m^3/s)", "(GPM)"

END IF

! Case 2: Determination of pond temperature

IF (choice==2) THEN

WRITE (100,20) "Time", " ", "Pond Radiation", "Sky Radiation", "Soil Conduction", &
& "Evaporation", "Convection", "Solar Radiation", &
& "Warm inlet", "Cold inlet", "Discharge", "Rain", "Pond Temperature"

WRITE (100,20) "(seconds)", "(days)", "(Watts)", "(Watts)", "(Watts)", "(Watts)", &
& "(Watts)", "(Watts)", "(Watts)", "(Watts)", "(Watts)", "(Watts)", "(°C)"

END IF

=====
! CALCULATIONS AND INTEGRATION

DO

!-----
! Determination of the atmospheric variables
!
! Because the user may wish to either use recorded data or generated
! weather data, all atmospheric variables will be called up from *.dat
! files. The user, however, must make sure the time steps in the data
! file corresponds to the time step in this program. For instance, if the
! model is to simulate pond heat transfer as of March 10th, every 10

```

!      minutes, then data for every 10 minutes as of March 10th are to be in
!      the respective *.dat file.
!
!      In the absence of real data, the user may generate his/her own data
!      with the use of sinusoidal functions. For temperature approximations for
!      Baton Rouge, LA, the program temperature.exe, generated from the
!      temperature.f90 code, can be used.
!
!      Also, MAKE SURE THAT THE DATA FILES ARE IN THE SAME DIRECTORY AS
! FISHFRY!
!

```

```

read (200, *) airtemp, airtemp, airtemp, airtemp, rh, solar_radiation, windspeed, rain

```

```

!      Conversions

```

```

rh=rh*0.01
solar_radiation=solar_radiation*1000.0
windspeed=windspeed*3600.0/1609.0 ! converts m/s to mph
rain=rain/3600.0 ! converts m/hr into m/s

```

```

if (choice==1) read (400,*) temp
if (choice==2) read (300,*) flow_rate_hot, flow_rate_hot, flow_rate_hot,&
                & flow_rate_hot, flow_rate_cold

```

```

!
!      From these variables, all other atmospheric variables can be determined.
!
!      Energy of vaporization

```

```

latent_energy= 2502535.259 -212.56384*temp ! J/kg

```

```

!-----

```

```

!
!      Determination of the coefficients
!

```

```

coefficient(1)=emissivity_water*area*5.67*(10.0**(-8))
coefficient(2)=emissivity_sky(airtemp, rh)*area*5.67*(10.0**(-8))*(airtemp+273)**4

```

```

!if (choice==3) then
!      ! For the case when the temperature is the driving force

```

```

!coefficient(3) = -thermal_conductivity * Area&
!      & * (-soiltempamp_year/Dyear&

```

```

!      & *(sin(2*acos(-1.0)/31536000.0*time+phase_year)&
!      & +cos(2*acos(-1.0)/31536000.0*time+phase_year))&
!      & -soiltempamp_day/Dday&
!      & *(sin(2*acos(-1.0)/86400.0*time+phase_day)&
!      & +cos(2*acos(-1.0)/86400.0*time+phase_day)))
!
!
!
!else      ! Steady state constant temperature conditions

!coefficient(3)=thermal_conductivity*area/sqrt(PI*thermal_diffusivity)
!END if
!
!
!

coefficient(3)=0.0
coefficient(4)=(0.068+0.059*ws(windspeed))*(vp(Temp, x)-vp(airtemp,rh))&
      & *area*density/24.0/3600.0*0.0254& ! rate of evaporation (kg/s)
      & *latent_energy

coefficient(5)=(2.8+3.0*windspeed*1609.0/3600.0)*area
coefficient(6)=solar_radiation*area
coefficient(7)=density*specheat*hot*flow_rate_hot
coefficient(8)=density*specheat*cold*flow_rate_cold
coefficient(9)=density*specheat*airtemp*area*rain

if (choice==1) then
coefficient(10)=density*specheat*(area*rain)
else
coefficient(10)=density*specheat*(flow_rate_hot+flow_rate_cold+area*rain)
end if

!
! "If" statements to make sure no evaporation occurs when the air is saturated.
!
!
if (rh>=1.0) coefficient(4)=0
if (vp(airtemp,rh)>vp(Temp,x)) coefficient(4) =0
!
!=====
!   DETERMINATION OF VECTORS
!

```

```
vector(1)= - coefficient(1)*(273.0+temp)**4 ! Pond radiation (J/s)
vector(2)= coefficient(2) ! Sky radiation
```

```
!if (choice==3) then ! Soil conduction (weather forcing)
! vector(3)= coefficient(3)
!ELSE ! Soil conduction (temperature constant)
```

```
! vector(3)=coefficient(3)*(Temp_initial-temp)/SQRT(time)
```

```
!end if
```

```
vector(3)=0.0
```

```
vector(4)= - coefficient(4) ! Evaporation
```

```
vector(5)= coefficient(5)*(airtemp - temp) ! Surface convection
```

```
vector(6)= coefficient(6) ! Solar Radiation
```

```
vector(9)= coefficient(9) ! Rain
```

```
vector(10)=-coefficient(10)*temp
```

```
if (choice==1) THEN
```

```
vector(7)=0.0
```

```
vector(8)=0.0
```

```
if (SUM(vector)<0) then
```

```
flow_rate_hot=SUM(vector)/(specheat*density*(temp-hot))
```

```
gpm_hot=flow_rate_hot*1000*60/3.78
```

```
flow_rate_cold=0.0
```

```
gpm_cold=0.0
```

```
end if
```

```
if (SUM(vector)>0) then
```

```
flow_rate_cold=SUM(vector)/(specheat*density*(temp-cold))
```

```
gpm_cold=flow_rate_cold*1000*60/3.78
```

```
flow_rate_hot=0.0
```

```
gpm_hot=0.0
```

```
end if
```

```
END if
```

```
if (choice==2) then
```

```
vector(7)= coefficient(7)
```

```
vector(8)= coefficient(8)
```

```
vector(9)= coefficient(9)
```

```
vector(10)= -coefficient(10)*temp
```

end if

!=====

! Output of results

!

! Case 1

IF (choice==1) THEN

vector(7)=0

vector(8)=0

write (100,10) Time, time/3600.0/24.0 + 1, vector(1), vector(2), &
& vector(3), vector(4), vector(5), vector(6), vector(9),&
& vector(10), SUM(vector), flow_rate_hot, gpm_hot, &
& flow_rate_cold, gpm_cold

END IF

!

! Case 2

!

IF (choice==2) THEN

write (100,10) Time, time/3600.0/24.0+1, vector, temp

END IF

!

!=====

! Runge-Kutta Formulae

!

! Remember, the Runge-Kutta method predicts the temperature for the
! following step.

!

F1=step*dTemp(Temp,time,coefficient,thermal_mass, airtemp)!step)

F2=step*dtemp(Temp+0.5*F1, time+0.5*step,coefficient,thermal_mass, airtemp)!step)

F3=step*dTemp(Temp+0.5*F2, time+0.5*step,coefficient,thermal_mass, airtemp)!step)

F4=step*dTemp(Temp+F3, time+step,coefficient,thermal_mass, airtemp)!step)

time=time+step

Temp=temp+(F1+2*F2+2*F3+F4)/6

IF (time>timemax) GO TO 9

```
END do
```

```
9 write (*,*) "Thank you for using FISHFRY, the best program in the whole,"  
write (*,*) "wide world. (Quite frankly, I don't know why you would want to"  
write (*,*) "end your session. I mean, do you really have anything better"  
WRITE (*,*) "to do?"  
WRITE (*,*) " "  
WRITE (*,*) "Results are in the output.dat file."  
write (*,*) " "
```

```
stop  
end program fishfry
```

```
!=====!  
!  FUNCTIONS  
!=====!
```

```
!-----!  
!  Function to evaluate dtemp  
!-----!
```

```
Function dtemp (temp, time,coefficient,thermal_mass, airtemp)  
IMPLICIT NONE
```

```
    REAL (8), INTENT (IN):: temp, time, thermal_mass, airtemp  
    REAL (8), dimension (10), INTENT (IN):: coefficient  
    REAL (8):: dtemp
```

```
dtemp= (&  
    & - coefficient(1) * ((Temp+273.0)**4)&      ! Backradiation  
    & + coefficient(2) &                        ! Sky longwave rad  
    & + coefficient(3) &  
!    & + coefficient(3)*(Tsoil-temp)/SQRT(timereset)& ! Soil heat exchange  
    & - coefficient(4)&                          ! Evaporation  
    & + coefficient(5)*(airtemp-temp)&          ! Convection  
    & + coefficient(6)&                          ! Solar radiation  
    & + coefficient(7)&  
    & + coefficient(8)+coefficient(9) &  
    & - coefficient(10)*temp&  
    & )/thermal_mass - time+ time                ! Thermal mass
```

```
Return  
END function dtemp
```

```
!-----
!   Function to evaluate emissivity_sky
!-----
```

```
Function emissivity_sky(airtemp,rh)
IMPLICIT NONE
```

```
!   This function calculates the emissivity of the sky based on a graph
!   produced by Bliss,1961, Atmospheric Radiation Near the Surface of
!   the Ground: a Summary for Engineers. Solar Energy 5:103-120
!
```

```
REAL (8), INTENT(IN):: airtemp, rh
REAL (8):: emissivity_sky, dewT
```

```
dewT = 1/((1/(airtemp+273))-1.846*(10.0**(-4))*LOG(rh)) - 273
emissivity_sky = 1/(1.2488219 -0.0060896701*dewT+4.8502935e-005*dewT**2)
```

```
! dewT = dew temperature (C)
! emissivity_sky = the sky's emissivity (decimal)
```

```
Return
END function emissivity_sky
```

```
!-----
!   Function to evaluate the vapour pressure
!-----
```

```
!
!   This function was taken from ASHRAE Fundamentals Handbook, 1985, SI
!   edition. All temperatures must be in Kelvin and all pressures are in
!   Pascals.
```

```
FUNCTION vp(t, rh)
IMPLICIT NONE
REAL (8), INTENT (IN):: T, rh
REAL (8):: vp_saturated, vp
```

```
vp_saturated= exp ( -5800.2206/(t+273) &
& +1.3914993 &
& -0.04860239 * (t+273) &
& +0.41764768 * (10.0**(-4)) * ((t+273)**2)&
& -0.14452093 * (10.0**(-7)) * ((t+273)**3)&
& +6.5459673 * log(t+273))
```

```
vp=rh*vp_saturated
```

```
vp=vp/3387 ! conversion from Pascals to inches of mercury (I HATE IMPIRIAL UNITS!!)
```

```
return
```

```
END function vp
```

```
!-----  
!   Function to evaluate the corrected wind speed (REQUIRES INPUT!)  
!-----
```

```
function ws(windspeed)
```

```
implicit none
```

```
REAL (8), INTENT (IN):: windspeed
```

```
REAL (8):: ws, height_anemometer
```

```
!   This function corrects the windspeed for a height of 13 feet above  
!   the ground. The height of the anemometer is in feet and the windspeed  
!   in m/s. The function then converts from m/s to mph.  
!
```

```
height_anemometer=10.0
```

```
ws=windspeed*LOG(13.0)/LOG(height_anemometer)! conversion due to anemometer height  
!ws=ws*3600.0/1609.0           ! converts m/s to mph
```

```
return
```

```
end function ws
```

APPENDIX 13: PROGRAM CODE FOR WEATHER GENERATOR(A)

```
program weather_generator
implicit none

interface
function solar(i, month, x)
implicit none

INTEGER, INTENT (IN):: i, month, x
REAL:: solar
END function solar
END interface

CHARACTER (LEN=20):: filename, garbage
REAL, dimension (8760):: airtemp, rh, windspeed
REAL, DIMENSION (8760):: airtemp_max, airtemp_min, airtemp_avg
REAL, DIMENSION (8760):: rh_avg, windspeed_avg
REAL, DIMENSION (8760):: solar_radiation_max, solar_radiation_min
REAL, DIMENSION (8760):: solar_radiation_avg

INTEGER:: i, year, month, day, hour, j, n, choice, x

88 FORMAT (f5.1, ",", f5.0, ",", f10.5, ",", f5.1 )

!=====
open (300, FILE="hot.dat")
open (301, FILE="cold.dat")
open (302, FILE="avg.dat")

!=====
airtemp_max=-1000.0
airtemp_min=1000.0

n=0

!=====

1 n=n+1

write (*,*) "What is the name of the file?"
read (*,*) filename
open (10, FILE=filename)
```

```
read (10,*) garbage
read (10,*) garbage
open (20, file = "garbage")
write (20, *) garbage
```

```
Do i=1,8760
```

```
! years before 1964 do not have wind speed data
```

```
if (year<1964) then
```

```
    read (10, *) year, month, day, hour, airtemp(i), rh(i)
```

```
    if (month==2.and.day==29) then
```

```
        Do j=1,24
```

```
        read (10,*) year, month, day, hour, airtemp(i), rh(i)
```

```
        END do
```

```
    end if
```

```
    if (airtemp_max(i)<airtemp(i)) airtemp_max(i)=airtemp(i)
```

```
    if (airtemp_min(i)>airtemp(i)) airtemp_min(i)=airtemp(i)
```

```
    airtemp_avg(i)=(airtemp_avg(i)*(n-1.0)+airtemp(i))/n
```

```
    rh_avg(i)=(rh_avg(i)*(n-1.0)+rh(i))/n
```

```
    windspeed_avg(i)=(windspeed_avg(i)*(n-1.0)+windspeed(i))/n
```

```
        else ! ALTERNATE CASES
```

```
! First alternate case: between years 1965 and 1972, the records are
```

```
! every 3 hours
```

```
IF (year>=1965.and.year<=1972) THEN
```

```
if (MOD(i, 3)==1) then
```

```
    read (10, *) year, month, day, hour, airtemp(i), rh(i), windspeed(i)
```

```
    if (month==2.and.day==29) then
```

```
        Do j=1,23
```

```
        read (10,*) year
```

```
        END do
```

```
    read (10, *) year, month, day, hour, airtemp(i), rh(i), windspeed(i)
```

```
    end if
```

```

if (airtemp_max(i)<airtemp(i)) airtemp_max(i)=airtemp(i)
if (airtemp_min(i)>airtemp(i)) airtemp_min(i)=airtemp(i)
airtemp_avg(i)=(airtemp_avg(i)*(n-1.0)+airtemp(i))/n
rh_avg(i)=(rh_avg(i)*(n-1.0)+rh(i))/n
windspeed_avg(i)=(windspeed_avg(i)*(n-1.0)+windspeed(i))/n

```

```

else

```

```

  read (10,*) year, month, day, hour
  airtemp(i)=airtemp(i-1)
  rh(i) =rh(i-1)
  windspeed(i) = windspeed(i-1)
  if (airtemp_max(i)<airtemp(i)) airtemp_max(i)=airtemp(i)
  if (airtemp_min(i)>airtemp(i)) airtemp_min(i)=airtemp(i)
  airtemp_avg(i)=(airtemp_avg(i)*(n-1.0)+airtemp(i))/n
  rh_avg(i)=(rh_avg(i)*(n-1.0)+rh(i))/n
  windspeed_avg(i)=(windspeed_avg(i)*(n-1.0)+windspeed(i))/n

```

```

END if

```

```

! Second alternate case: 1964, and 1973 onward

```

```

  Else

```

```

read (10,*) year, month, day, hour, airtemp(i), rh(i), windspeed(i)

```

```

if (month==2.and.day==29) then

```

```

  Do j=1,24
  read (10,*) year, month, day, hour, airtemp(i), rh(i), windspeed(i)
  END do
end if

```

```

if (airtemp_max(i)<airtemp(i)) airtemp_max(i)=airtemp(i)
if (airtemp_min(i)>airtemp(i)) airtemp_min(i)=airtemp(i)
airtemp_avg(i)=(airtemp_avg(i)*(n-1.0)+airtemp(i))/n
rh_avg(i)=(rh_avg(i)*(n-1.0)+rh(i))/n
windspeed_avg(i)=(windspeed_avg(i)*(n-1.0)+windspeed(i))/n

```

```

END IF

```

```

END if

```

```

airtemp(i) = (airtemp(i)-32)*5.0/9.0
airtemp_max(i) = (airtemp_max(i)-32)*5.0/9.0
airtemp_min(i) = (airtemp_min(i)-32)*5.0/9.0

```

```
airtemp_avg(i) = (airtemp_avg(i)-32)*5.0/9.0
windspeed_avg(i) = windspeed_avg(i) * 0.51444
```

```
! Solar radiation calculations
```

```
x=0
solar_radiation_avg(i)=solar(i, month, x)
```

```
x=1
solar_radiation_max(i)=solar(i, month, x)
```

```
x=2
solar_radiation_min(i)=solar(i, month, x)
```

```
END do
```

```
WRITE (*,*) "Do you want to run an other year? (1 for yes, 2 for no) "
read (*,*) choice
if (choice==1) GO TO 1
```

```
!=====
```

```
DO i=1,8760
    write (300,88) airtemp_max(i), rh_avg(i), solar_radiation_max(i), windspeed_avg(i)
    write (301,88) airtemp_min(i), rh_avg(i), solar_radiation_min(i), windspeed_avg(i)
    write (302,88) airtemp_avg(i), rh_avg(i), solar_radiation_avg(i), windspeed_avg(i)
```

```
END DO
```

```
stop
end program weather_generator
```

```
!
=====
```

```
function solar(i, month, x)
implicit none
```

```
INTEGER, INTENT (IN):: i, month, x
```

```
REAL:: solar
```

```
REAL:: dy, a, pi
```

```
REAL:: ET, decl, rr
```

REAL:: TST, LST, Lnt, Lng, Lat
 REAL:: HR, Elev, Directsun, etr, PETR
 REAL:: sunup, sundown

Lat = 30.32
 Lng = 91.15
 Lnt = 90.00
 pi=ACOS(-1.0)

Dy=i/24
 LST=1.0*MOD(i,24)

! Position of the sun

a=2*pi*dy/365.0
 rr=1.000110+0.034221*COS(a)+0.001280*SIN(a)+& ! ratio
 & 0.000719*COS(2*a)+0.000077*SIN(2*a)

ET= 0.0172+0.4281*COS(a)-7.3515*SIN(a)& ! time (min)
 & -3.3495*COS(2*a)-9.3619*sin(2*a)

Decl= 0.39637 -22.91326*COS(a)+4.02543*SIN(a)& ! degrees
 & -0.38720*COS(2*a)+0.05197*SIN(2*a)&
 & -0.15453*COS(3*a)+0.08479*SIN(3*a)

TST = LST +(Lnt-Lng)/15.0+ET/60.0

HR = (12.0-tst)*15 ! degrees

Elev = ASIN(SIN(pi/180*lat)*SIN(pi/180*Decl)& ! radians
 & +COS(pi/180*lat)*COS(pi/180*decl)*COS(pi/180*HR))

! Daylength

sundown=((180/PI*ACOS(-TAN(Lat*Pi/180)*TAN(decl*pi/180)))/15+12)-(Lnt-Lng)/15-et/60
 sunup =(12-(180/PI*ACOS(-TAN(Lat*Pi/180)*TAN(decl*pi/180)))/15)-(Lnt-Lng)/15-et/60
 ! \-----hour angle-----/
 ! \-----true solar time-----/
 ! \-----local standard time-----/

! Extra-terrestrial solar radiation

```
Directsun=1377 * rr      ! Solar constant taken as 1377 W/m^2  
etr=Directsun/1000*COS(pi/2-elev)  ! kW/m^2
```

```
if (LST<sunup.or.lst>sundown) etr=0
```

```
if (month==1) petr= 0.46  
if (month==2) petr= 0.50  
if (month==3) petr= 0.56  
if (month==4) petr= 0.62  
if (month==5) petr= 0.62  
if (month==6) petr= 0.63  
if (month==7) petr= 0.58  
if (month==8) petr= 0.61  
if (month==9) petr= 0.61  
if (month==10) petr= 0.64  
if (month==11) petr= 0.54  
if (month==12) petr= 0.48  
if (x==1) petr=0.85  
if (x==2) petr=0.25
```

```
solar=petr*etr
```

```
Return  
END function solar
```