(1) See Mathematica notebook.

(2a) The eigenfunctions we want satisfy
\[ \frac{d^2 F_n}{dx^2} = -\lambda_n F_n, \quad F_n(0) = 0, \quad F_n'(L) = 0. \]

The general solution of this equation for arbitrary \( \lambda > 0 \) (it is easy to use the Rayleigh quotient to show that \( \lambda > 0 \)) is
\[ F = A \cos \frac{\pi n}{L} x + B \sin \frac{\pi n}{L} x. \]

Then \( F(0) = 0 \Rightarrow A = 0 \), and \( F'(L) = 0 \Rightarrow \cos \left( \frac{\pi n}{L} L \right) = 0 \),
so \( \frac{\pi n}{L} L \equiv \frac{\pi (n - \frac{1}{2})}{L}, \quad n = 1, 2, 3, \ldots. \) Then
\[ \lambda_n = \left( \frac{n - \frac{1}{2}}{L} \right)^2 \pi^2, \quad F_n(x) = \sin \left[ \frac{(n - \frac{1}{2}) \pi x}{L} \right]. \]

(5) We try \( T(\alpha, t) = \sum_{n=1}^{\infty} C_n(\alpha) \sin \left[ \frac{(n - \frac{1}{2}) \pi x}{L} \right]. \)

For \( \alpha \leq 1 \), we have \( t = \cos \alpha \omega \sin \left[ \frac{\pi \alpha}{2L} \right] = \cos \alpha \sin \left[ \frac{(2\alpha - 1) \pi x}{2L} \right]. \)
Thus \( T(\alpha, t) = \sum_{n=1}^{\infty} \delta_n(\alpha) \sin \left[ \frac{(n - \frac{1}{2}) \pi x}{L} \right], \quad \) where \( \delta_n = 0 \) for \( n \neq 2 \).

And \( \delta_2(\alpha) = \cos \alpha \omega. \) We substitute the two expansions into the equation:
\[ \frac{\partial}{\partial t} \sum_{n=1}^{\infty} C_n(\alpha) \sin \left[ \frac{(n - \frac{1}{2}) \pi x}{L} \right] = D \frac{d^2}{dx^2} \sum_{n=1}^{\infty} C_n(\alpha) \sin \left[ \frac{(n - \frac{1}{2}) \pi x}{L} \right] + \sum_{n=1}^{\infty} \delta_n(\alpha) \sin \left[ \frac{(n - \frac{1}{2}) \pi x}{L} \right]. \]

We may carry out the \( x \)-differentiation termwise because \( T \) and the eigenfunctions satisfy the same boundary conditions. Then
\[ \sum_{n=1}^{\infty} \alpha \frac{\partial}{\partial t} \sin \left[ \frac{(n - \frac{1}{2}) \pi x}{L} \right] = \sum_{n=1}^{\infty} D \frac{d^2}{dx^2} C_n(\alpha) \sin \left[ \frac{(n - \frac{1}{2}) \pi x}{L} \right] + \sum_{n=1}^{\infty} \delta_n(\alpha) \sin \left[ \frac{(n - \frac{1}{2}) \pi x}{L} \right]. \]
(b) (continued). Balancing the coefficients of the sines gives us
\[ \frac{dC_n}{dt} + \beta_n C_n = \delta_n \quad n = 1, 2, \ldots \]
where \( \beta_n = D \frac{(k-\frac{1}{2})^2}{L^2} \).

From the initial condition on \( T \) we get
\[ T(x, 0) = \sum_{n=1}^{\infty} C_n (0) \sin \left( \frac{n \pi x}{L} \right) = 0 \]

Balancing coefficients gives us \( C_n (0) = 0 \).

We will get nonzero values for any of the \( C_n (0) \)'s only when there is a nonzero initial condition on \( T \).

(c) \( n \neq 2 \):
\[ \frac{dC_n}{dt} + \beta_n C_n = 0 \quad C_n (0) = 0. \]

The solution of this homogeneous equation with zero initial condition is
\[ C_n (t) = 0 \quad \forall t. \]

\( n = 2 \):
\[ \frac{dC_2}{dt} + \beta_2 (2 = \delta_2 (t) = 10 \cos \omega t. \]

We use an integrating factor to solve this equation (an alternative is to find any supersede particular and homogeneous solutions.) The integrating factor is \( e^{\beta_2 t} \).
We multiply the equation by this,
\[ e^{\beta_2 t} \frac{dC_2}{dt} + \beta_2 e^{\beta_2 t} (2 = 10 e^{\beta_2 t} \cos \omega t. \]

or \[ \frac{d}{dt} (e^{\beta_2 t} (2 (t)) = 10 e^{\beta_2 t} \cos \omega t. \]

We integrate from 0 to t using \( t \) as the variable of integration. Then,
\[ e^{\beta_2 t} (2 (t)) \bigg|_0^t = 10 \int_0^t e^{\beta_2 t} \cos \omega t \, dt. \]
Continued:

\[ e^{\beta_2 t} (c_2(t) - c_2(0)) = \int_0^t e^{\beta_2 (t-t')} \cos \omega t' \, dt' \]

So \[ c_2(t) = \int_0^t e^{-\beta_2 (t-t')} \cos \omega t' \, dt' \].

We use Mathematica to evaluate the integral to get

\[ c_2(t) = \frac{1}{\beta_2^2 + \omega^2} \left[ \beta_2 \cos(\omega t) - \omega \sin(\omega t) \right] - \frac{\beta_2}{\omega} e^{-\beta_2 t} \]

See the Mathematica notebook for problem 2. We also check our solution there. The final answer is

\[ T(x,t) = \frac{1}{\beta_2^2 + \omega^2} \left[ \beta_2 \cos(\omega t) + \omega \sin(\omega t) \right] - \frac{\beta_2}{\omega} e^{-\beta_2 t} \cdot \sin \left( \frac{3\pi x}{2l} \right) \]

where \[ \beta_2 = \sqrt{\left( \frac{3\pi}{2l} \right)^2} \].

**Challenge Problem**

In class we analyzed a rectangular parallelepiped with zero boundary conditions on all 6 faces. We found separated solutions of the form

\[ e^{-\lambda \rho t} \sin \left( \frac{\rho x}{a} \right) \sin \left( \frac{\rho y}{b} \right) \sin \left( \frac{\rho z}{c} \right), \]

where \[ -\lambda \rho = \rho^2 \left( \frac{1}{a^2} + \frac{1}{b^2} + \frac{1}{c^2} \right) \] - (a)-(b). As we discussed in class, the criteria start
is when the lowest mode has a zero \( \lambda \), so in this case when \( (p=q=r) \)
\[
D \pi^2 \left( \frac{1}{a^2} + \frac{1}{b^2} + \frac{1}{c^2} \right) = 0 - 8.
\]
For a cube, \( b = c = a \), so
\[
3 \pi^2 \frac{a^2}{a^2} = 8,
\]
and the critical side length is
\[
A_c = \pi \sqrt[3]{\frac{3D}{8}}.
\]
(9) By making the neutron flux zero on one face, we will decrease the neutron loss rate, so the critical size should be somewhat smaller. Here is more detail. Assume steady state. Then
\[
0 = D \nabla^2 N + \xi N - B N.
\]
Integrate this over the volume \( V \) of the reactor and use the divergence theorem to get
\[
- \oint S \nabla N \cdot dS = (0 - B) \int_V \int_s N dV,
\]
which says that the neutron loss rate at the surface is equal to the net neutron production. If there is now a zero flux on part of \( S \), the left-hand side will get smaller, so the right-hand side must get smaller, which can be achieved by a smaller volume.

(b) The \( x \) and \( y \) modes are unchanged because the boundary conditions are unchanged on \( x = 0, y \)
and on \( y = 0, \infty \). For the \( z \)-part, we want eigenfunctions of \( \frac{\partial^2}{\partial z^2} \) which satisfy zero function on \( z = 0 \), and zero derivative on \( z = 0 \). Let \( \psi(z) \) be the \( z \)-eigenfunction. Then

\[
\frac{\partial^2 \psi}{\partial z^2} = -\lambda \psi, \quad 0 < z < a
\]

\( \psi(0) = 0, \quad \frac{\partial \psi}{\partial z}(a) = 0. \)

The general solution is \( \psi(z) = A \cos \sqrt{\lambda} z + B \sin \sqrt{\lambda} z \). 

\( \psi(0) = 0 \Rightarrow A = 0, \quad \psi'(a) = 0 \Rightarrow \cos \sqrt{\lambda} a = 0, \quad \sin \sqrt{\lambda} a = 0. \)

\( \sqrt{\lambda} a = (n_r - \frac{1}{2})\pi, \quad n_r = 1, 3, \ldots \)

\[
\lambda_r = \frac{(n_r - \frac{1}{2})^2 \pi^2}{a^2}.
\]

The \( z \)-mode are now

\[
\sin \left( \frac{\partial_x}{a} \right) \sin \left( \frac{\partial_y}{a} \right) \sin \left[ \frac{(n_r - \frac{1}{2}) \pi z}{a} \right],
\]

\( p, q, r = 1, 3, \ldots \)

(It is easy to show with the Rayleigh quotient that all of the \( z \)-eigenvalues are positive.)

(c) Now we get separated solutions of the form

\[
e^{-\lambda_y t} \sin \left( \frac{\partial_x}{a} \right) \sin \left( \frac{\partial_y}{a} \right) + \sin \left( \frac{(r-\frac{1}{2}) \pi z}{a} \right)
\]

where \( \lambda_y = \frac{\partial_y^2}{a^2} (p^2 + q^2 + (r-\frac{1}{2})^2) - \lambda_r \).

The critical state corresponds to the smallest \( \lambda \).
being zero, so \( p = q = r = 1 \) and 

\[
\frac{Dp^2}{g^2} \left(1 + \frac{1}{4} \right) = 3L
\]

so 

\[
g^2 = \frac{Dp^2 \cdot 9}{q-3}
\]

\[
\Rightarrow L = \frac{3D}{\sqrt{q-3}} < \frac{\sqrt{2D}}{q-3}
\]

So the critical side length is reduced by a factor of 

\[
\frac{3/2}{\sqrt{5}} = 0.866
\]
Newton's Law of Cooling in Transient Conduction

1. Introduction

This notebook uses *Mathematica* to solve problem 1 of Assignment #7, in which we analyze transient heat conduction in a slab of width $L$. The boundary condition on the right face of the slab is a specified constant heat flux $F_0$, and the boundary condition on the left face of the slab is a convective flux, described by Newton's law of cooling, with an ambient temperature $T_A$. The initial temperature is constant and is equal to the ambient temperature $T_A$. The relevant parameters are the thermal conductivity $k$, the thermal diffusivity $D_f$, and the heat transfer coefficient $h$, all assumed constant. The specific numerical expressions for all functions and parameters are given in section 5. The mathematical formulation of the problem is given below.

$$\frac{\partial T}{\partial t} = D_f \frac{\partial^2 T}{\partial x^2}, \quad 0 < x < L, \quad t > 0,$$

with $k \frac{\partial T}{\partial x}(0, t) = h[T(0, t) - T_A]$, $k \frac{\partial T}{\partial x}(L, t) = F_0$, and $T(x, 0) = T_A$.

2. Steady State Solution

As in other problems of this sort, we must first find the steady-state solution $T_s(x)$, and then reformulate the problem in terms of the transient solution $T_r$. The steady-state solution is obtained by setting the time derivative to zero in the original equation for $T$. The solution of the resulting equation is a linear function of $x$, and we require that particular linear function which satisfies the boundary conditions at $x = 0$ and $L$. The result is easily shown to be

$$T_s[x_] := T_A + \frac{F_0}{h} + \frac{F_0}{k} x$$

We check this. Because it is linear, it clearly satisfies the equation. Now the boundary conditions:

$$k D[T_s[x], x] /. x \rightarrow L$$

$$F_0$$

$$k D[T_s[x], x] - h (T_s[x] - T_A) /. x \rightarrow 0$$

$$0$$

It checks.

3. Formulation of Problem for Transient

Now that we have the steady-state solution, we decompose the full solution into a steady-state part $T_s(x)$ and a transient part $T_r(x, t)$:
\[ T(x, t) = T_s(x) + T_r(x, t) . \]  

By substituting this decomposition into the original equation for \( T \), we find the following problem for the transient:

\[
\begin{align*}
\frac{\partial T_r}{\partial t} &= D_f \frac{\partial^2 T_r}{\partial x^2}, \quad 0 < x < L, \ t > 0, \\
-k \frac{\partial T_r}{\partial x}(0, t) - h T_r(0, t) &= 0, \quad \frac{\partial T_r}{\partial x}(L, t) = 0, \ \text{and} \ T_r(x, 0) = T_0(x) - T_s(x) . 
\end{align*}
\]

This is the problem that we solve by separation of variables.

### 4. Separation of Variables in Transient Problem

Following the standard approach in separation of variables, we seek functions which satisfy the equation for \( T_r \) and the homogeneous boundary conditions at \( x = 0, L \). We assume a form \( F(x)G(t) \) for such solutions. The two separated equations then have the form

\[
F''(x) + \lambda F(x) = 0, \ \text{with} \ kF'(0) - hF(0) = 0, \ F'(L) = 0 \ \text{and} \ G'(t) + \lambda G(t) = 0 .
\]

With the Rayleigh quotient, one may show that all of the eigenvalues are positive. Then the general solution of the equation for \( F(x) \) is a linear combination of \( \sin(\sqrt{\lambda} \ x) \) and \( \cos(\sqrt{\lambda} \ x) \). Applying the two homogeneous boundary conditions to the general solution, we get a solution which may be put into the form \( F(x) = \cos(\sqrt{\lambda} \ (1 - x)) \). Simplifying the resulting eigenvalue equation, we get

\[
\tan(z) = B_i/z, \ \text{where} \ z = L \sqrt{\lambda}, \ \text{and where} \ B_i = hL/k .
\]

We define the relevant quantities for Mathematica, denoting the nth value of \( z \) by \( z[n] \), and the nth eigenvalue by \( \text{lam}[n] \).

\begin{verbatim}
In[4]:= lam[n_] := (z / L) ^ 2 / . z -> z[n]
\end{verbatim}

where the values \( z[1], z[2], .. z[n] .. \) are the roots of \( \text{eig1} = \text{eig2} \), where

\begin{verbatim}
In[5]:= eig1 = Tan[z];
In[6]:= eig2 = Bi / z;
\end{verbatim}

with

\begin{verbatim}
In[7]:= Bi := (h * L) / k
\end{verbatim}

The generic eigenfunction is

\begin{verbatim}
In[8]:= geneig[x_] := Cos[z (1 - x / L)]
\end{verbatim}

and the nth eigenfunction is given by substituting for \( z \) the \( n \)th eigenvalue \( z(n) \):

\begin{verbatim}
In[9]:= Feig[x_, n_] := geneig[x] / . z -> z[n]
\end{verbatim}

The generic normalization integral is

\begin{verbatim}
In[10]:= nor = Integrate[(geneig[x]) ^ 2, {x, 0, L}]
Out[10]= L (z + Cos[z] Sin[z]) / 2 z
\end{verbatim}
and the nth normalization integral is

\[
\text{In}[11] = \text{norm}[n_] := \text{nor} /. \ z \to \ z[n]
\]

5. Parameter Values

In this section, we define the parameter values and the initial function. This is the only place in the notebook where these quantities are given specifically. This makes it possible to change a value for the entire calculation by just changing it here. The material properties are appropriate for granite. The initial temperature is taken here to be a constant. We also calculate the value of the Biot number Bi.

\[
\text{In}[12] = h = 22.4; \quad (\text{** W/m}^2\cdot\text{K} \text{ **})
\]

\[
\text{In}[13] = L = 0.75; \quad (\text{** m **})
\]

\[
\text{In}[14] = k = 16.2; \quad (\text{** W/m} \cdot\text{K **})
\]

\[
\text{In}[15] = \text{Df} = 4.1 \times 10^{-6}; (\text{** m}^3/\text{s **})
\]

\[
\text{In}[16] = \text{T}A = 10.0; \quad (\text{** °C **})
\]

\[
\text{In}[17] = \text{T}0[x_] = 10.0; \quad (\text{** °C **})
\]

\[
\text{In}[18] = \text{PO} = 170.2; \quad (\text{** W/m}^2 \text{ **})
\]

\[
\text{In}[19] = \text{Bi}
\]

\[
\text{Out}[19] = 1.03704
\]

6. Determination of Eigenvalues

We now determine numerically the eigenvalues. We begin by plotting the expressions eig1 and eig2. At each crossing of these, there is an eigenvalue.

\[
\text{In}[20] = \text{SetOption}[\text{Plot}, \text{ImageSize} \to 250];
\]

\[
\text{In}[21] = \text{Plot}\{\text{eig1, eig2}, \{z, 0, 10\}, \text{PlotRange} \to \{-10, 10\},
\text{PlotStyle} \to \{\text{RGBColor[1, 0, 0]}, \text{RGBColor[0, 0, 1]}\}, \text{AxesLabel} \to \{"z", \text{\"eig1 (red) and eig2 (blue)\"}\}, \text{Ticks} \to \{\{0, \text{Pi}/2, \text{Pi}, 3\text{Pi}/2, 2\text{Pi}, 5\text{Pi}/2, 3\text{Pi}\}\}\}
\]

\[
\text{Out}[21] =
\]
We see that the first root is between 0 and \( \pi/2 \), and each subsequent root is in an interval of length \( \pi \). It is also clear that for large \( n \), the \( n \)th root is very close to \( (n-1)\pi \). To find accurate numerical values of the \( z \)'s, we use the Mathematica command \texttt{FindRoot}. \texttt{FindRoot} requires an initial guess. The \( n \)th value of \( z \), as we can see from the graph, is between \( (n - 1)\pi \) and \( (n - 0.5)\pi \). We take as an initial guess \( (n - 0.5)\pi - 0.1 \). The command is then of the form \texttt{FindRoot[eig1 == eig2, \{z, (n - 0.5)\pi - 0.1\}]}. We now use this command in a Do loop to calculate and display in a table the first 40 roots. This will suffice for any calculation requiring up to 40 terms in the eigenfunction series. We also display the eigenvalue \( \lambda \), and the approximate value \( z = (n-1)\pi \).

\[
\text{ln}[22] := \text{zroot}[n_] := \text{FindRoot}[\text{eig1} == \text{eig2}, \{z, (n - 0.5) \pi - 0.1\}]
\]

\[
\text{ln}[23] := \text{zasymp}[n_] := \text{N}[\text{(n - 1) \pi}]
\]

\[
\text{ln}[24] := \text{Do}[z[n] = z /\ . \text{zroot}[n], \{n, 1, 40\}];
\]
We see that the asymptotic formula for $z[n]$ has an error of less than 0.5% for any $n$ equal to or greater than 6. It is possible to develop even more accurate, but still simple, asymptotic formulas.

7. Representation of the Initial Condition

We are finally ready to begin solving the boundary value for the transient $T_{r}(x,t)$. The form of the solution for the transient is
\[
T_r(x, t) = \sum_{n=1}^{\infty} C(n) \exp\left(-\lambda(n) D f t\right) \cos\left(\pi n \left(1 - \frac{x}{L}\right)\right),
\]

where the coefficients \( C(n) \) are determined by the initial conditions satisfied by \( T_r \). The formula for the \( n \)th coefficient is

\[
\text{In}[26]:= \text{coeff}[n_] := \text{Integrate}\left[\text{Feig}[x, n] \ast (T0[x] - Ts[x]), \{x, 0, L\}\right] / \text{norm}[n]
\]

We now use a Do loop to construct the first 40 coefficients, and then print them out in a table. The \( n \)th numerical value is assigned to \( c[n] \).

\[
\text{In}[27]:= \text{Do}[c[n] = \text{Re}[\text{coeff}[n]], \{n, 1, 40\}];
\]

\[
\text{In}[28]:= \text{TableForm}[
\begin{array}{ccc}
n & c[n] & n & c[n] \\
1 & -13.2501 & 21 & -0.0040 \\
2 & -1.2362 & 22 & -0.0036 \\
3 & -0.3706 & 23 & -0.0033 \\
4 & -0.1715 & 24 & -0.0030 \\
5 & -0.0979 & 25 & -0.0028 \\
6 & -0.0631 & 26 & -0.0026 \\
7 & -0.0440 & 27 & -0.0024 \\
8 & -0.0324 & 28 & -0.0022 \\
9 & -0.0248 & 29 & -0.0020 \\
10 & -0.0196 & 30 & -0.0019 \\
11 & -0.0159 & 31 & -0.0018 \\
12 & -0.0132 & 32 & -0.0017 \\
13 & -0.0111 & 33 & -0.0016 \\
14 & -0.0094 & 34 & -0.0015 \\
15 & -0.0081 & 35 & -0.0014 \\
16 & -0.0071 & 36 & -0.0013 \\
17 & -0.0062 & 37 & -0.0012 \\
18 & -0.0055 & 38 & -0.0012 \\
19 & -0.0049 & 39 & -0.0011 \\
20 & -0.0044 & 40 & -0.0010 \\
\end{array}
\]

As a check on all that we have done, we see how well our series represents the initial condition, by plotting both the initial condition (in blue) and its representation by the first 40 terms of our series (in red).

\[
\text{In}[29]:= \text{exactinit}[x_] := T0[x] - Ts[x]
\]

\[
\text{In}[30]:= \text{seriesinit}[x_] := \text{Sum}[c[n] \ast \text{Feig}[x, n], \{n, 1, 40\}]
\]
We see that the graphical agreement is excellent. We also check a few values.

\[
\begin{array}{c|c|c}
\text{x (m)} & \text{Exact IC} & \text{Series IC} \\
0 & -7.5982 & -7.5977 \\
0.05 & -8.1235 & -8.1237 \\
0.1 & -8.6488 & -8.6492 \\
0.15 & -9.1741 & -9.1736 \\
0.2 & -9.6994 & -9.6995 \\
0.25 & -10.2248 & -10.2253 \\
0.3 & -10.7501 & -10.7496 \\
0.35 & -11.2754 & -11.2752 \\
0.4 & -11.8007 & -11.8014 \\
0.45 & -12.3260 & -12.3255 \\
0.5 & -12.8513 & -12.8508 \\
0.55 & -13.3766 & -13.3779 \\
0.6 & -13.9019 & -13.9013 \\
0.65 & -14.4272 & -14.4256 \\
0.7 & -14.9525 & -14.9571 \\
0.75 & -15.4778 & -15.4374 \\
\end{array}
\]

Again excellent agreement, and we conclude that we are using enough terms (40) in the series.

8. Solution of the Initial-Boundary Value Problem

Now that we have the eigenvalues and series coefficients, we can construct the series solution of the problem. We define Tran[x,t,n] as the nth partial sum of the series solution, and we define term[x,t,k] as the kth term in the series. As a special case of importance, we define firstterm[x,t] to be the first term in the series.
Let's look at the first term:

\[ \text{In[36]} = \text{firstterm}[x, t] \]
\[ \text{Out[36]} = -13.2501 e^{-5.53938 \times 10^{-4} t} \cos[0.871766 (1 - 1.33333 x)] \]

We may calculate the e-folding time for this mode -- call it \( \tau_1 \) -- as the reciprocal of the coefficient of \( t \) in the exponent:

\[ \text{In[37]} = \frac{1}{\lambda[1] \cdot \text{Df}} \]
\[ \text{Out[37]} = 180.526. \]

This time, which is in seconds, is a little over 50 hours. We compare this with the diffusion time \( L^2/(\pi^2 D_f) \) we derived earlier for zero boundary conditions on both sides of the slab. In hours, the time is

\[ \text{In[38]} = \frac{L^2}{(3600 \ (\pi^2 \text{Df})} \]
\[ \text{Out[38]} = 3.86133 \]

This is very much less than the 50 hours we calculated for the first mode in our problem. There are two reasons for this. The first reason is that our slab is insulated on one end, and this will reduce the cooling rate. One can show that the diffusion time for a slab of width \( L \) which has zero temperature on one face and which is insulated on the other face is \( (4 \ L^2)/(\pi^2 D_f) \). For our present parameters, the value of this in hours is

\[ \text{In[39]} = \frac{4 \ L^2}{(3600 \ (\pi^2 \text{Df})} \]
\[ \text{Out[39]} = 15.4453 \]

This is somewhat closer to our calculated value of 50 hours, but still considerably smaller. There is another effect in the present problem, and that is an additional thermal resistance at the boundary (characterized by the reciprocal of the heat transfer coefficient \( h \)). This extra resistance increases the decay time to the 50 hours that we calculated. If you have studied heat transfer, you will know that because the Biot number is around 1, the thermal resistance at the boundary is comparable with that internal to the slab.

Let's look at the decay times of the second and third modes, converting them to hours as we calculate them

\[ \text{In[40]} = \frac{1}{\lambda[2] \cdot \text{Df} \cdot 3600} \]
\[ \text{Out[40]} = 3.23021 \]
\[ \text{In[41]} = \frac{1}{\lambda[3] \cdot \text{Df} \cdot 3600} \]
\[ \text{Out[41]} = 0.9181 \]

Knowledge of these decay times will help us choose time values for plotting the solution. To make this a little easier, we construct a table of the first 20 decay times.

\[ \text{In[42]} = \frac{1}{\lambda[n_] \cdot \text{Df}} \]
We function now side, snapshot[t].

9. Graphs of the Solution

We define here graphs of the solution for T versus x, for various values of time. We test the code with a few graphs, and then we construct a sequence to be shown in a Manipulate panel. The function which produces the graph at time t is named snapshot[t]. We use all the 40 terms we have calculated in the series to compute the solution, although this is somewhat extravagant in that fewer terms would be sufficient for the longer times.

\[
\begin{align*}
\text{In[43]:=} & \quad \text{TableForm[Table[{n, PaddedForm[t[n], {10, 2}], PaddedForm[t[n]/3600, {10, 5}]}, \{n, 1, 20\}], } \\
& \quad \text{TableHeadings } \rightarrow (\text{None}, (\text{"Mode"}, \text{"Decay Time (s)"}, \text{"Decay Time (hr)"}))]
\end{align*}
\]

\[
\begin{array}{ccc}
\text{Mode} & \text{Decay Time (s)} & \text{Decay Time (hr)} \\
1 & 180525.51 & 50.14598 \\
2 & 11628.76 & 3.23021 \\
3 & 3305.16 & 0.91810 \\
4 & 1509.62 & 0.41934 \\
5 & 857.60 & 0.23822 \\
6 & 551.41 & 0.15317 \\
7 & 383.90 & 0.10664 \\
8 & 282.48 & 0.07847 \\
9 & 216.49 & 0.06014 \\
10 & 171.17 & 0.04755 \\
11 & 138.72 & 0.03853 \\
12 & 114.68 & 0.03186 \\
13 & 96.39 & 0.02678 \\
14 & 82.15 & 0.02282 \\
15 & 70.85 & 0.01968 \\
16 & 61.72 & 0.01715 \\
17 & 54.26 & 0.01507 \\
18 & 48.06 & 0.01335 \\
19 & 42.88 & 0.01191 \\
20 & 38.48 & 0.01069 \\
\end{array}
\]

Now we are ready to define snapshot[t], which gives us a plot of temperature versus x at time t. From the ambient temperature and the initial distribution, we can get a range of variation of temperature. We extend this range slightly on either side, and call the extended range plrange:

\[
\begin{align*}
\text{In[44]:=} & \quad \text{Trans[x_, t_ ] := Tran[x, t, 40]} \\
\text{In[45]:=} & \quad \text{Temp[x_, t_ ] := Trans[x, t] + Ts[x]}
\end{align*}
\]

Now we are ready to define snapshot[t], which gives us a plot of temperature versus x at time t. From the ambient temperature and the initial distribution, we can get a range of variation of temperature. We extend this range slightly on either side, and call the extended range plrange:

\[
\begin{align*}
\text{In[46]:=} & \quad \text{plrange = (0, 30);}
\end{align*}
\]

Now we define snapshot[t], giving a special definition for snapshot[0] -- namely the initial distribution. For convenience, the function snapshot takes an argument in hours, but converts it internally to seconds, to be consistent with the units of the problem.

\[
\begin{align*}
\text{In[47]:=} & \quad \text{snapshot[t_] := Module[\{time\}, time = 3600 * t; \\
& \quad \text{Plot[Temp[x, time], \{x, 0, L\}, PlotStyle } \rightarrow \text{ Thickness[0.004], PlotRange } \rightarrow \text{ plrange, \\
& \quad \text{AxesLabel } \rightarrow \text{ \"x", \"Temperature\", PlotLabel } \rightarrow \text{ Row\{\"t = \", PaddedForm[t, \{4, 2\}], \" hr\"\}]}
\end{align*}
\]

\[
\begin{align*}
\text{In[48]:=} & \quad \text{snapshot[0] := Plot[20[x], \{x, 0, L\}, PlotStyle } \rightarrow \text{ Thickness[0.004], PlotRange } \rightarrow \text{ plrange, \\
& \quad \text{AxesLabel } \rightarrow \text{ \"x", \"Temperature\", PlotLabel } \rightarrow \text{ Row\{\"t = \", PaddedForm[0, \{4, 2\}], \" hr\"\}]}
\end{align*}
\]

We try it for the initial time and for 1, 10, 20, 30, 40 and 50 hr.
In[49]:= snapshot[0]

Out[49]=

In[50]:= snapshot[1]

Out[50]=

In[51]:= snapshot[10]

Out[51]=
In[52]:= snapshot[20]

Out[52]=

\[\text{Temperature} \quad t = 20.00 \text{ hr}\]

Out[52]=

\[\text{Temperature} \quad t = 20.00 \text{ hr}\]

Out[52]=

\[\text{Temperature} \quad t = 20.00 \text{ hr}\]

In[53]:= snapshot[30]

Out[53]=

\[\text{Temperature} \quad t = 30.00 \text{ hr}\]

Out[53]=

\[\text{Temperature} \quad t = 30.00 \text{ hr}\]

Out[53]=

\[\text{Temperature} \quad t = 30.00 \text{ hr}\]

In[54]:= snapshot[40]

Out[54]=

\[\text{Temperature} \quad t = 40.00 \text{ hr}\]

Out[54]=

\[\text{Temperature} \quad t = 40.00 \text{ hr}\]

Out[54]=

\[\text{Temperature} \quad t = 40.00 \text{ hr}\]
As an aid to visualizing the transient process, we produce a sequence of 201 graphs that can be animated to show the dynamics of the heat loss process. The graphs are at 0.5 hr intervals, running from 0 hr to 100 hr. The graphs are output into a Manipulate panel for ease of viewing.

Have we reached steady-state at 100 hours? Not quite. Here are the endpoint values at 100 hours and in steady-state.

<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>In[57] := Temp[0, 3600 * 100]</td>
<td>Out[57] = 16.4376</td>
</tr>
<tr>
<td>In[58] := Temp[L, 3600 * 100]</td>
<td>Out[58] = 23.6742</td>
</tr>
</tbody>
</table>
Let's look at 300 hours.

In[61]= snapshot[300]

10. The One Term Approximation to the Transient

The first term in the transient series has a decay time of over 50 hours, whereas the next term has a decay time of only 3.2 hours. Thus the first term will be dominant for times longer than several hours. We now make a graphical comparison of the series solution (in red) and the approximate solution retaining only one term of the transient series (in blue). The basic graphing command is compgraph[t], which takes a time argument in hours and plots both the series solution and the one-term approximate solution.

In[54]= Tapprox[x_, t_] := Ts[x] + firstterm[x, t]

In[65]= compgraph[t_] := Module[{time}, time = 3600 * t;
    Plot[{Temp[x, time], Tapprox[x, time]}, {x, 0, L}, PlotRange -> {5, 15},
    PlotStyle -> (Red, Blue), FrameLabel -> {"x (m)", "Temperature °C"}, Frame -> True, PlotLabel -> Row[{"Exact T (Blue), Approx T (Red); ", "t = ", PaddedForm[t, {4, 2}], " hr"}]]

We try this for 1 hour.
We look at graphs every half-hour up to 5 hours.
Exact T (Blue), Approx T (Red); \( t = 2.50 \) hr

Exact T (Blue), Approx T (Red); \( t = 3.00 \) hr

Exact T (Blue), Approx T (Red); \( t = 3.50 \) hr
The approximation is good by 5 hours. We also look at 10 hours.
Now the agreement is excellent.
We use Mathematica to evaluate the integral occurring in the solution for the coefficient $C_2[t]$.

We now use Mathematica to check our solution, which we must first define.

We now substitute this into the equation.

The solution satisfies the equation. Now we check the boundary and initial conditions.

Everything checks.