1. Introduction

This notebook uses *Mathematica* to solve the problem presented in class, in section 5.1 of the notes. The problem is one of transient heat conduction in a slab of finite width $L$. The boundary condition on the left face of the slab is zero heat flux, and the condition on the right face of the slab is Newton’s law of cooling, with an ambient temperature $T_a$. The initial temperature is the function $T_0(x)$. The relevant material properties are the thermal conductivity $k$, the thermal diffusivity $D_f$, and the heat transfer coefficient $h$, all assumed constant. The specific expressions for all functions and parameters are given below in section 2. The mathematical formulation of the problem is given below.

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

$$\frac{\partial T}{\partial x}(0, t) = 0, \quad k \frac{\partial T}{\partial x}(L, t) + h[T(L, t) - T_a] = 0,$$

and $T(x, 0) = T_0(x)$. 

2. 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.

```
In[1]:= h = 22.4; (** W/m²·K **)
In[2]:= L = 0.5; (** m **) 
In[3]:= k = 2.80; (** W/m·K **) 
In[4]:= Df = 1.37 * 10^-6; (** m²/s **) 
In[5]:= Ta = 10.0; (** °C **) 
In[6]:= T0[x_] = 60.0; (** °C **) 
```

3. 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 $T_t$. 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 (which is intuitively obvious) is that $T_s(x)$ is constant and equal to the ambient temperature:
4. 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 $T_r(x,t)$:

$$T(x, t) = T_s(x) + T_r(x, t).$$

(2)

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

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

with

$$\frac{\partial T_r}{\partial x}(0, t) = 0, \quad k \frac{\partial T_r}{\partial x}(L, t) + h T_r(L, t) = 0,$$

$$\text{and} \quad T_r(x, 0) = T_0(x) - T_s(x).$$ (3)

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

5. 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

$$F''(x) + \lambda F(x) = 0, \quad \text{with} \quad F'(0) = 0 \quad \text{and} \quad kF'(L) + hF(L) = 0,$$

and

$$G'(t) + \lambda DG(t) = 0.$$

(4)

The general solution for 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 $F(x) = \cos(\sqrt{\lambda} x)$, and simplifying the resulting eigenvalue equation, we get

$$\tan(z) = B_i/z, \quad \text{where} \quad z = L \sqrt{\lambda}, \quad \text{and} \quad B_i = hL/k.$$ (5)

We define the relevant quantities for Mathematica, denoting the nth value of $z$ by $z[n]$, and the nth eigenvalue by lam[n].

In[8]:= lam[n_] := (z / L)^2 /. z -> z[n]

where the values $z[1], z[2], \ldots z[n] \ldots$ are the roots of eig1 = eig2, where

In[9]:= eig1 = Tan[z];

In[10]:= eig2 = Bi / z;

with

In[11]:= Bi := (h * L) / k

The generic eigenfunction is

In[12]:= geneig[x_] := Cos[(z * x) / L]

and the nth eigenfunction is given by

In[13]:= Feig[x_, n_] := geneig[x] /. z -> z[n]
The generic normalization integral is
\[
\text{nor} = \text{Integrate}[(\text{geneig}[x])^2, \{x, 0, L\}]
\]
\[
\text{Out}[14] = 0.25 + \frac{0.125 \sin(2. z)}{z}
\]
and the nth normalization integral is
\[
\text{norm}[n_] := \text{nor} /. z \rightarrow z[n]
\]

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.

\[
\text{SetOption}[\text{Plot}, \text{ImageSize} \rightarrow 250];
\]
\[
\text{Plot}[\{\text{eig1}, \text{eig2}\}, \{z, 0, 10\}, \text{PlotRange} \rightarrow \{-10, 10\},
\quad \text{PlotStyle} \rightarrow \{\text{RGBColor}[1, 0, 0], \text{RGBColor}[0, 0, 1]\},
\quad \text{AxesLabel} \rightarrow \{"z", \text{"eig1 (red) and eig2 (blue)"}\},
\quad \text{Ticks} \rightarrow \{(0, \text{Pi}/2, \text{Pi}, 3\text{Pi}/2, 2\text{Pi}, 5\text{Pi}/2, 3\text{Pi})\}]
\]

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 nth root is very close to \((n-1)\pi\). To find accurate numerical values of the \(z\)'s, we use the Mathematica command \text{FindRoot}. \text{FindRoot} requires an initial guess. The nth 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 \text{FindRoot}[\text{eig1} == \text{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 \(z = (n-1)\pi\).

\[
\text{zroot}[n_] := \text{FindRoot}[\text{eig1} == \text{eig2}, \{z, (n - 0.5) \ast \text{Pi} - 0.1\}]
\]
\[
\text{zasymp}[n_] := N[(n - 1) * \text{Pi}]
\]
\[
\text{Do}[z[n] = z /. \text{zroot}[n], \{n, 1, 40\}];
\]
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 10. 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
\[ T_r(x, t) = \sum_{n=1}^{\infty} C(n) \exp(-\lambda(n) D_T t) \cos((z(n) x)/L), \]  

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

\[ \text{coeff}[n_] := \text{Integrate}[	ext{Feig}[x, n] * (T0[x] - Ts[x]), \{x, 0, L\}]/\text{norm}[n] \]

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 \( c[n] \).

\[ \text{Do}[c[n] = \text{coeff}[n], \{n, 1, 40\}]; \]

\[ \text{TableForm}[\text{Table}[\{n, \text{PaddedForm}[c[n], \{8, 4\}], \text{PaddedForm}[n + 20, \{10\}], \text{PaddedForm}[c[n + 20], \{12, 4\}]], \{n, 1, 20\}], \text{TableHeadings} -> \{\text{None}, \{"n", "c[n]", "n", "c[n]"}\}] \]

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).

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

\[ \text{seriesinit}[x_] := \text{Sum}[c[n] * \text{Feig}[x, n], \{n, 1, 40\}] \]
In[27]:= seriesgraph = Plot[{exactinit[x], seriesinit[x]}, {x, 0, L}, 
    PlotRange -> {0, 60}, PlotStyle -> 
    {{RGBColor[0, 0, 1], Thickness[0.004]}, 
     {RGBColor[1, 0, 0], Thickness[0.004]}}, AxesLabel -> 
    {"x", "Exact (blue), Series (red)"}]

Out[27]=

The agreement looks excellent, and is good evidence that our calculations are correct so far. To get a little closer view of the accuracy, we replot with a much narrower range in y.

In[28]:= Show[seriesgraph, PlotRange -> {48, 52}]

Out[28]=

Now we can see a little overshoot at the right end of the interval. This is not the Gibbs phenomenon, however, because the magnitude of the overshoot decreases to zero as the number of terms is increased. We haven’t shown that here, but it can be shown. With some algebraic labor, it also can be shown that the coefficients \(c[n]\) diminish like \(1/n^2\), and not like the \(1/n\) that would be associated with the Gibbs overshoot.

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,n] as the nth term in the series. As a special case of importance, we define firstterm[x,t] to be the first term in the series.

In[29]:= term[x_, t_, n_] := c[n]*Feig[x, n]*Exp[-lam[n]*Df*t]

In[30]:= Tran[x_, t_, n_] := Sum[term[x, t, kk], {kk, 1, n}]

In[31]:= firstterm[x_, t_] := term[x, t, 1]
Let's look at the first term:

\[ \text{firstterm}[x, t] \]

\[
\begin{align*}
\text{Out}[32] &= 61.4354 e^{8.76357 \times 10^{-4} t} \cos[2.52918 x] \\
\end{align*}
\]

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:

\[
\text{In}[33] = \tau_1 = 1 / (\text{lam}[1] \times \text{Df})
\]

\[
\text{Out}[33] = 114.109
\]

This time, which is in seconds, equals about 31.7 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

\[
\text{In}[34] = \frac{L^2}{(3600 \pi^2 D_f)}
\]

\[
\text{Out}[34] = 5.13591
\]

It is interesting that this is less than the 31.7 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

\[
\text{In}[35] = \frac{4 L^2}{(3600 \pi^2 D_f)}
\]

\[
\text{Out}[35] = 20.5436
\]

This is much closer to our calculated value of 31.7 hours, but still 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 31.7 hours that we calculated.

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

\[
\text{In}[36] = \tau_2 = 1 / (\text{lam}[2] \times \text{Df} \times 3600)
\]

\[
\text{Out}[36] = 3.27335
\]

\[
\text{In}[37] = \tau_3 = 1 / (\text{lam}[3] \times \text{Df} \times 3600)
\]

\[
\text{Out}[37] = 1.09172
\]

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.

\[
\text{In}[38] = \tau[n_] := 1 / (\text{lam}[n] \times \text{Df})
\]
TableForm[Table[{n, PaddedForm[t[n], {10, 2}], PaddedForm[t[n] / 3600, {10, 5}]}, {n, 1, 20}], TableHeadings -> {None, {"Mode", "Decay Time (s)", "Decay Time (hr)"}}]

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.

\[ \text{Trans}[x_, t_] := \text{Tran}[x, t, 40] \]

\[ \text{Temp}[x_, t_] := \text{Trans}[x, t] + \text{Ts}[x] \]

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:

\[ \text{plrange = \{0, 70\};} \]

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.

\[ \text{snapshot}[t_] := \text{Module}[\{\text{time} = 3600 \times t\}, \text{Plot}[	ext{Temp}[x, \text{time}], \{x, 0, \text{L}\}, \text{PlotStyle} \to \text{Thickness}[0.004], \text{PlotRange} \to \text{plrange}, \text{AxesLabel} \to \{"x", "Temperature"\}, \text{PlotLabel} \to \text{Row}[\{"t = ", \text{PaddedForm}[t, \{4, 2\}], " hr"\}]]) \]

\[ \text{snapshot}[0] := \text{Plot}[	ext{T0}[x], \{x, 0, \text{L}\}, \text{PlotStyle} \to \text{Thickness}[0.004], \text{PlotRange} \to \text{plrange}, \text{AxesLabel} \to \{"x", "Temperature"\}, \text{PlotLabel} \to \text{Row}[\{"t = ", \text{PaddedForm}[0, \{4, 2\}], " hr"\}]]) \]

We try it for the initial time and for 1, 10, 20, and 30 hr.
In[45]:= `snapshot[0]`

\begin{figure}
\centering
\includegraphics[width=\textwidth]{snapshot0.png}
\caption{t = 0.00 hr}
\end{figure}

Out[45]=

In[46]:= `snapshot[1]`

\begin{figure}
\centering
\includegraphics[width=\textwidth]{snapshot1.png}
\caption{t = 1.00 hr}
\end{figure}

Out[46]=

In[47]:= `snapshot[10]`

\begin{figure}
\centering
\includegraphics[width=\textwidth]{snapshot10.png}
\caption{t = 10.00 hr}
\end{figure}

Out[47]=
As a final task, we produce a sequence of 201 graphs that can be animated to show the dynamics of the heat loss process. The graphs are at 0.25 hr intervals, running from 0 hr to 50 hr. The graphs are output into a Manipulate panel for ease of viewing.
In[60]:= DynamicModule[{mangraph, i, k}, mangraph[0] = snapshot[0];
   Do[mangraph[i] = snapshot[0.25*i], {i, 1, 200}]; Manipulate[mangraph[k], {k, 0, 200, 1}]]

Out[60]=