

OCT 29, 2009

(1) See Mathematica notebook.

(2) (a) We write the solution as $T(x,t) = T_s(x) + \tilde{T}(x,t)$, where T_s is the steady-state temperature and \tilde{T} is the transient part. We solve for T_s first

$$0 = D \frac{d^2 T_s}{dx^2} + \delta, \quad 0 < x < L, \quad T_s(0) = 0, \quad T_s(L) = 0.$$

We integrate twice to get $T_s = -\frac{\delta}{2D} x^2 + Ax + B$.

Then $T_s(0) = 0 \Rightarrow B = 0$, $T_s(L) = 0 \Rightarrow A = \frac{\delta L}{2D}$, so

$$T_s(x) = \frac{\delta x}{2D} (L-x).$$

The transient part satisfies

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

$$\tilde{T}(0,t) = 0, \quad \tilde{T}(L,t) = 0, \quad \tilde{T}(x,0) = -T_s(x).$$

(We obtain this formulation by substitution)

$T = T_s + \tilde{T}$ into the original problem for T .)

This is a problem we have solved several times by separation of variables. The separated solutions which satisfy the equation and the two homogeneous boundary conditions are $e^{-n^2 \pi^2 D t / L^2}$

$$e^{-n^2 \pi^2 D t / L^2} \sin(n \pi x / L), \quad n = 1, 2, 3, \dots$$

We superpose these to satisfy the initial condition:

$$\tilde{T}(x,t) = \sum_{n=1}^{\infty} C_n e^{-n^2 \pi^2 D t / L^2} \sin(n \pi x / L),$$

$$\text{so } \tilde{T}(x,0) = -T_s(x) = -\frac{\delta x}{2D} (L-x) = \sum_{n=1}^{\infty} C_n \sin(n \pi x / L).$$

This is a Fourier sine series, so

$$C_n = \frac{2}{L} \int_0^L \left(-\frac{\delta x}{2D} (L-x) \right) \sin\left(\frac{n \pi x}{L}\right) dx = \begin{cases} 0, & n \text{ even} \\ -\frac{\delta L^2}{D n^3 \pi^3}, & n \text{ odd} \end{cases}$$

(The integral is easily done in Mathematica or by two integrations-by-parts.)

(2) (G) (continued) Our solution is

$$T(x,t) = T_s(x) + \tilde{T}(x,t)$$

$$= \frac{\delta x}{2D} (L-x) - \sum_{\substack{n=1 \\ \text{odd}}}^{\infty} \frac{4\delta L^2}{Dn^3\pi^3} e^{-n^2\pi^2Dt/L^2} \sin\left(\frac{n\pi x}{L}\right)$$

For comparison with our solution obtained in part (b), it is helpful to write

$$T_s(x) = \frac{\delta x}{2D} (L-x) = \sum_{\substack{n=1 \\ \text{odd}}}^{\infty} \frac{4\delta L^2}{Dn^3\pi^3} \sin\left(\frac{n\pi x}{L}\right)$$

So that

$$T(x,t) = \sum_{\substack{n=1 \\ \text{odd}}}^{\infty} \frac{4\delta L^2}{Dn^3\pi^3} \left[1 - e^{-n^2\pi^2Dt/L^2}\right] \sin\left(\frac{n\pi x}{L}\right)$$

(5) We start over, using the formulation for T given below. This time we will not split T into a steady-state plus transient parts. We try

$$T(x,t) = \sum_{n=1}^{\infty} c_n(t) \sin(n\pi x/L)$$

The functions $\sin(n\pi x/L)$ are eigenfunctions of the operator $\frac{d^2}{dx^2}$ which appears in the equation.

This means that when the series is substituted into the equation and differentiated termwise, it will be another series of the same type (sine series). This will enable us to balance coefficients. (How do we know we can differentiate the series termwise? As we

ME201/ME2028/ME400/ME400

SOL7 PAGE THREE

(2) (b) (continued) showed earlier, such differentiation is possible when the eigenfunctions and the function represented by the series satisfy the same boundary conditions.)

Here are the terms in the equation.

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial t} \sum_{n=1}^{\infty} (c_n t) \sin\left(\frac{n\pi x}{L}\right) = \sum_{n=1}^{\infty} \frac{dc_n}{dt} \sin\left(\frac{n\pi x}{L}\right).$$

$$D \frac{\partial^2 T}{\partial x^2} = D \frac{\partial^2}{\partial x^2} \sum_{n=1}^{\infty} (c_n t) \sin\left(\frac{n\pi x}{L}\right) = \sum_{n=1}^{\infty} -D \frac{n^2 \pi^2}{L^2} c_n \sin\left(\frac{n\pi x}{L}\right)$$

and $\delta = \sum_{n=1}^{\infty} \delta_n \sin\left(\frac{n\pi x}{L}\right)$ where

$$\delta_n = \frac{2}{L} \int_0^L \delta \sin\left(\frac{n\pi x}{L}\right) dx = \begin{cases} 0 & n \text{ even} \\ \frac{\delta}{n\pi} & n \text{ odd} \end{cases}$$

We substitute the three expansions into the equation:

$$\sum_{n=1}^{\infty} \frac{dc_n}{dt} \sin\left(\frac{n\pi x}{L}\right) = \sum_{n=1}^{\infty} \left(-D \frac{n^2 \pi^2}{L^2} c_n\right) \sin\left(\frac{n\pi x}{L}\right) + \sum_{n=1}^{\infty} \delta_n \sin\left(\frac{n\pi x}{L}\right).$$

We let $a_n = D \frac{n^2 \pi^2}{L^2}$, and we balance coefficients of $\sin\left(\frac{n\pi x}{L}\right)$:

$$\frac{dc_n}{dt} = -a_n c_n + \delta_n.$$

This is an ordinary differential equation for $c_n(t)$. We get the initial conditions for the c_n 's from the initial condition for $T(x, t)$.

(2) (b) (continued)

$$T(x, 0) = 0 = \sum_{n=1}^{\infty} C_n(0) \sin\left(\frac{n\pi x}{L}\right) \Rightarrow C_n(0) = 0$$

We have $\frac{dC_n}{dt} + \alpha_n C_n = f_n$.

We use an integrating factor $e^{\alpha_n t}$ (MTH 163 or 165)

$$e^{\alpha_n t} \frac{dC_n}{dt} + \alpha_n e^{\alpha_n t} C_n = e^{\alpha_n t} f_n$$

$$\frac{d}{dt} (e^{\alpha_n t} C_n) = e^{\alpha_n t} f_n$$

Integrate: $e^{\alpha_n t} C_n = \frac{1}{\alpha_n} e^{\alpha_n t} f_n + D_n$.

At $t=0$,

$$1 - C_n(0) = \frac{1}{\alpha_n} f_n + D_n$$

so $D_n = -\frac{f_n}{\alpha_n}$ and

$$C_n(t) = \frac{f_n}{\alpha_n} (1 - e^{-\alpha_n t}) = \begin{cases} 0, & n \text{ even} \\ \frac{4\beta L^2}{D n^3 \pi^3} (1 - e^{-\alpha_n t}), & n \text{ odd} \end{cases}$$

so

$$T(x, t) = \sum_{\substack{n=1 \\ n \text{ odd}}}^{\infty} \frac{4\beta L^2}{D n^3 \pi^3} (1 - e^{-n^2 \pi^2 \beta t / L^2}) \sin(n\pi x / L)$$

(c) The two solutions are the same - the above formula is the same as the second one derived in part (a).

(3) As shown in class, the separation solutions have the form $e^{\lambda t}$ (spatial mode). In our case this is $e^{\lambda t} \sin\left(\frac{p\pi x}{a}\right) \sin\left(\frac{q\pi y}{b}\right) \sin\left(\frac{r\pi z}{c}\right)$.

We substitute this into the equation to get

(3) (continued) $\lambda = -D \left(\frac{p^2 \pi^2}{G^2} + \frac{q^2 \pi^2}{C^2} + \frac{r^2 \pi^2}{L^2} \right) + \alpha - \beta$

The critical state corresponds to $\lambda = 0$ for one mode, and $\lambda < 0$ for all other modes. As the mode numbers p, q, r increase, λ decreases. So we want the mode with smallest mode numbers to give $\lambda = 0$. Hence

$$0 = -D \left(\frac{\pi^2}{G^2} + \frac{\pi^2}{C^2} + \frac{\pi^2}{L^2} \right) + \alpha - \beta$$

This gives us the value of G for C (critical) reactor:

$$G^2 = \frac{2}{\frac{\alpha - \beta}{D\pi^2} - \frac{1}{C^2}}$$

Because G^2 must be positive, we get

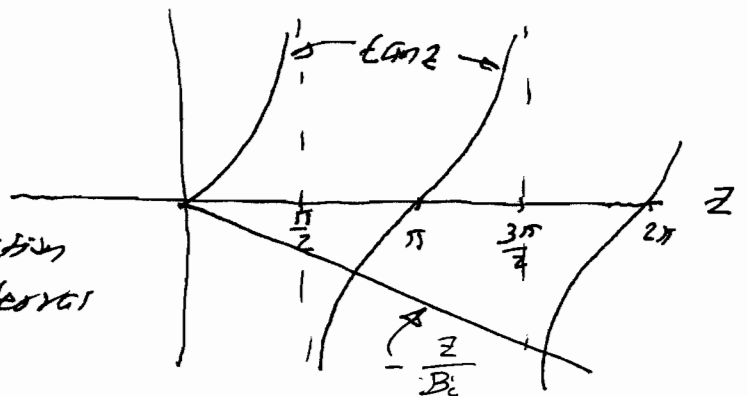
$$C^2 > \frac{D\pi^2}{\alpha - \beta}$$

If $C < \pi \sqrt{\frac{D}{\alpha - \beta}}$, the reactor is so

thin in the z dimension that neutron loss at the surfaces $z=0, C$ will be too great for a sustained reaction, no matter how large C is.

CHALLENGE PROBLEM

(a) We see from the graph that for any positive B_i , the line $-z/B_i$ will have a first intersection with $\tan z$ in the interval $(\pi/2, \pi)$.



CHALLENGE PROBLEM (c) (continued)

Note that for B_i very large (h large) the line is very shallow and the intersection is close to π . For B_i very small (h small), the line is much closer to vertical, and the intersection is close to $\pi/2$.

(b) As noted above $z \rightarrow \pi$ as $h \rightarrow \infty$.

(c) As noted in (b), $z \rightarrow \frac{\pi}{2}$ as $h \rightarrow 0$.

(d) See MATHEMATICS NOTEBOOK.

ME201/MTH281/ME400/CHE400

Assignment #7 Solutions

Problem 1 and Challenge Problem

■ Problem 1

■ 1. INTRODUCTION

This notebook uses *Mathematica* to solve problem 1 of Assignment #7. The notebook is patterned closely after the notebook handed out in class, entitled Newton's Law of Cooling. 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 fixed temperature T_L , 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 $T_0(x)$ which in this particular case is the zero function. The relevant material properties are the thermal conductivity k , the thermal diffusivity D_f , and the heat transfer coefficient h . The specific expressions for all functions and parameters for this problem are given in the next section. The mathematical formulation of the problem is given below.

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

$$\text{with } T(0, t) = T_L, \quad k \frac{\partial T}{\partial x}(L, t) + h[T(L, t) - T_A] = 0, \quad (1)$$

$$\text{and } T(x, 0) = T_0(x).$$

■ 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 values are relevant for granite.

$$h = 22.4; \quad (** \text{ W/m}^2 \cdot \text{K} **)$$

$$L = 1 / 2; \quad (** \text{ m} **)$$

$$k = 2.80; \quad (** \text{ W/m} \cdot \text{K} **)$$

$$Df = 1.37 * 10^{-6}; \quad (** \text{ m}^2/\text{s} **)$$

```

TL = 0.0; (** °C **)
TA = 10.0; (** °C **)
T0[x_] = 0.0; (** °C **)

```

■ 3. STEADY STATE SOLUTION

As in all 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

$$Ts[x_] := TL + (h * (TA - TL) / (k + h * L)) * x$$

The boundary condition at $x = 0$ is clearly satisfied. We check the boundary condition at $x = L$:

$$(k D[Ts[x], x] + h (Ts[x] - TA)) /. x -> L$$

$$-4.9738 \times 10^{-14}$$

Close enough to zero!

■ 4. FORMULATION OF PROBLEM FOR TRANSIENT

Now that we have found the steady-state solution, we decompose the full solution into a steady-state part $T_s(x)$ and a transient part $T_r(x,t)$:

$$T(x,t) = T_s(x) + T_r(x,t) . \quad (2)$$

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

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

$$\text{with } T_r(0, t) = 0, \quad k \frac{\partial T_r}{\partial x}(L, t) + h T_r(L, t) = 0, \quad (3)$$

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

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

$$F''(x) + \lambda F(x) = 0, \quad \text{with } F(0) = 0 \text{ and } kF'(L) + hF(L) = 0, \quad (4)$$

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

The general solution for the equation for $F(x)$ is $A \sin(\sqrt{\lambda} x) + B \cos(\sqrt{\lambda} x)$. Applying the boundary condition at $x = 0$, we get $B = 0$. Applying the boundary condition at $x = L$ and rejecting the trivial solution $A = 0$, we get

$$k \sqrt{\lambda} \cos(\sqrt{\lambda} L) + h \sin(\sqrt{\lambda} L) = 0. \quad (5)$$

Letting $z = \sqrt{\lambda} L$, we may put the eigenvalue equation in the following form:

$$\tan(z) = -\frac{z}{B_i}, \text{ where } B_i = \frac{hL}{k}. \quad (6)$$

Once we have found the roots $z[n]$ of this equation, we may calculate the eigenvalues $\lambda[n]$:

$$\text{lam}[n_] := (z / L)^2 /. z -> z[n]$$

We write the eigenvalue equation as $\text{eig1} = \text{eig2}$, where

$$\text{eig1} = \text{Tan}[z];$$

$$\text{eig2} = -z / B_i;$$

with

$$B_i := (h * L) / k$$

The generic eigenfunction is

$$\text{geneig}[x_] := \text{Sin}[(z * x) / L]$$

and the n th eigenfunction is given by

$$\text{Feig}[x_, n_] := \text{geneig}[x] /. z -> z[n]$$

The generic normalization integral is

$$\text{genor} = \text{Integrate}[(\text{geneig}[x])^2, \{x, 0, L\}]$$

$$\frac{1}{4} - \frac{\text{Sin}[2 z]}{8 z}$$

and the n th normalization integral is

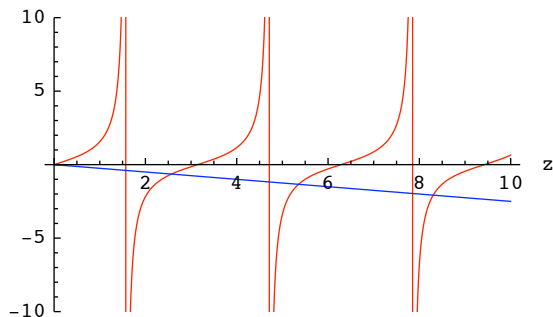
$$\text{norm}[n_] := \text{genor} /. z -> z[n]$$

■ 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.

```
Plot[{eig1, eig2}, {z, 0, 10}, PlotRange -> {-10, 10},
PlotStyle -> {RGBColor[1, 0, 0], RGBColor[0, 0, 1]},
ImageSize -> 250, AxesLabel -> {"z", "eig1 (red) and eig2 (blue)"}]
```

eig1 (red) and eig2 (blue)



We see that the first root is between $\pi/2$ and π , and each subsequent root is in an interval of length π . It is also clear that for large n , the n th root is very close to $(n-1/2)\pi$. We now use 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 $\lambda[n]$, and the approximate value of $z[n] = (n-\frac{1}{2})\pi$.

```
zroot[n_] := FindRoot[eig1 == eig2, {z, (n - 0.5) * Pi + 0.2}]
```

```
zasymp[n_] := (n - 1 / 2) * Pi
```

```
Do[z[n] = z /. zroot[n], {n, 1, 40}];
```

```
TableForm[Table[
  {n, PaddedForm[z[n], {10, 4}], PaddedForm[N[zasympt[n]], {10, 4}],
  PaddedForm[lam[n], {10, 4}]}, {n, 1, 40}], TableHeadings ->
{None, {"n", "      z[n]", "      zasympt[n]", "      lam[n]"}}]
```

n	z[n]	zasympt[n]	lam[n]
1	2.5704	1.5708	26.4285
2	5.3540	4.7124	114.6626
3	8.3029	7.8540	275.7545
4	11.3348	10.9956	513.9131
5	14.4080	14.1372	830.3585
6	17.5034	17.2788	1225.4800
7	20.6120	20.4204	1699.4233
8	23.7289	23.5619	2252.2514
9	26.8514	26.7035	2883.9946
10	29.9778	29.8451	3594.6689
11	33.1070	32.9867	4384.2834
12	36.2383	36.1283	5252.8434
13	39.3712	39.2699	6200.3523
14	42.5053	42.4115	7226.8124
15	45.6405	45.5531	8332.2252
16	48.7765	48.6947	9516.5916
17	51.9132	51.8363	10779.9124
18	55.0504	54.9779	12122.1882
19	58.1881	58.1195	13543.4193
20	61.3262	61.2611	15043.6061
21	64.4646	64.4026	16622.7487
22	67.6033	67.5442	18280.8473
23	70.7423	70.6858	20017.9021
24	73.8815	73.8274	21833.9132
25	77.0209	76.9690	23728.8807
26	80.1605	80.1106	25702.8046
27	83.3002	83.2522	27755.6850
28	86.4400	86.3938	29887.5219
29	89.5800	89.5354	32098.3155
30	92.7201	92.6770	34388.0657
31	95.8603	95.8186	36756.7725
32	99.0006	98.9602	39204.4359
33	102.1409	102.1018	41731.0561
34	105.2813	105.2434	44336.6330
35	108.4218	108.3849	47021.1666
36	111.5624	111.5265	49784.6569
37	114.7030	114.6681	52627.1040
38	117.8437	117.8097	55548.5079
39	120.9844	120.9513	58548.8685
40	124.1251	124.0929	61628.1858

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 ready finally to begin solving the boundary value for the transient $T_r(x,t)$. The form of the solution for the transient is

$$T_r(x,t) = \sum_{n=1}^{\infty} C(n) \text{Exp}(-\lambda(n) D_f t) \sin[(z(n)x/L], \quad (7)$$

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

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

We now use a Do loop to construct the first 40 coefficients, and then print them out in a table.

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

```
TableForm[Table[{n, PaddedForm[c[n], {8, 4}]}, {n, 1, 40}],  
TableHeadings -> {None, {"n", "c[n]"}]}
```

n	c[n]
1	-5.5617
2	2.0520
3	-0.9984
4	0.5714
5	-0.3648
6	0.2514
7	-0.1832
8	0.1391
9	-0.1092
10	0.0879
11	-0.0722
12	0.0604
13	-0.0512
14	0.0440
15	-0.0382
16	0.0335
17	-0.0296
18	0.0263
19	-0.0235
20	0.0212
21	-0.0192
22	0.0175
23	-0.0159
24	0.0146
25	-0.0135
26	0.0124
27	-0.0115
28	0.0107
29	-0.0100
30	0.0093
31	-0.0087
32	0.0082
33	-0.0077
34	0.0072
35	-0.0068
36	0.0064
37	-0.0061
38	0.0058
39	-0.0055
40	0.0052

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).

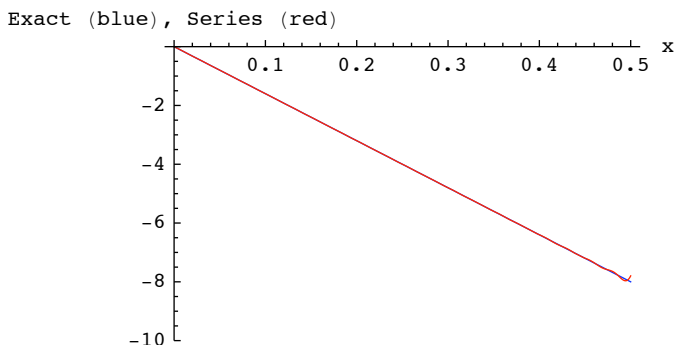
```
exactinit[x_] := T0[x] - Ts[x]
```

```

seriesinit[x_] := Sum[c[n] * Feig[x, n], {n, 1, 40}]

seriesgraph =
Plot[{exactinit[x], seriesinit[x]}, {x, 0, L}, PlotRange → {-10, 0},
PlotStyle → {RGBColor[0, 0, 1], RGBColor[1, 0, 0]},
ImageSize → 250, AxesLabel → {"x", "Exact (blue), Series (red)"}]

```

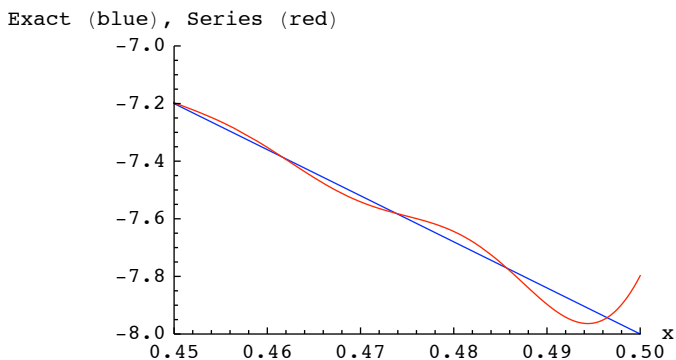


The agreement looks excellent, and is good evidence that our calculations are correct so far. To get a little closer view of the accuracy, we replot with a much narrower range in y , near the right endpoint.

```

Plot[{exactinit[x], seriesinit[x]},
{x, 0, L}, PlotRange → {{0.45, 0.5}, {-7, -8}},
PlotStyle → {RGBColor[0, 0, 1], RGBColor[1, 0, 0]},
ImageSize → 250, AxesLabel → {"x", "Exact (blue), Series (red)"}]

```



Now we can see some 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. As we will show now, the coefficients decrease like $1/n^2$ when n is large, rather than the $1/n$ decrease associated with the Gibbs overshoot. We examine the coefficients for large n by using the approximation obtained earlier for the eigenvalues when n is large -- that is, we use $z = \text{zasymp}[n] = (n - 1/2)\pi$:

Truncate

Truncate

```
largecoeff = Chop[Simplify[coeff[n] /. z[n] → zasymp[n], n ∈ Integers]]
```

$$\frac{1.62114 (-1)^n}{(0.5 - 1. n)^2}$$

Now we see that the coefficients do drop off like $1/n^2$ for large n .

■ 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 $\text{ser}[x,t,n]$ as the n th partial sum of the series solution, and we define $\text{term}[x,t,n]$ as the n th term in the series. As a special case of importance, we define $\text{firstterm}[x,t]$ to be the first term in the series.

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

```
ser[x_, t_, n_] := Sum[term[x, t, k], {k, 1, n}]
```

```
firstterm[x_, t_] := term[x, t, 1]
```

Let's look at the first term:

```
firstterm[x, t]
```

$$-5.56171 e^{-0.000036207 t} \sin[5.14086 x]$$

We may calculate the e-folding time for this mode -- call it τ_1 -- as the reciprocal of the coefficient of t in the exponent, and we express the result in units of hours:

```
 $\tau_1 = 1 / (\text{lam}[1] * \text{Df} * 3600)$ 
```

```
7.67193
```

Thus the fundamental mode has a decay time of about 7.67 hours. It is of interest to compare this with our canonical diffusion time, defined earlier in the course as

```
 $L^2 / (3600 \pi^2 \text{Df})$ 
```

```
5.13591
```

Thus the actual decay time for the first mode is somewhat longer than the diffusion time obtained earlier. This is a consequence of the thermal resistance at the boundary, as measured by the reciprocal of h . (Actually, the relevant parameter is the dimensionless ratio of the resistance in the slab, L/k , to the surface resistance $1/h$, and this ratio is just $hL/k = B_i$.)

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

```
 $\tau_2 = 1 / (\text{lam}[2] * \text{Df} * 3600)$ 
```

```
1.7683
```

```
 $\tau_3 = 1 / (\text{lam}[3] * \text{Df} * 3600)$ 
```

```
0.735283
```

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 40 decay times.

```
 $\tau[n_] := 1 / (\text{lam}[n] * \text{Df})$ 
```

```
TableForm[
  Table[{n, PaddedForm[ $\tau[n]$ , {10, 2}], PaddedForm[ $\tau[n] / 3600$ , {10, 5}]},
    {n, 1, 40}], TableHeadings ->
  {None, {"Mode", "Decay Time (s)", "Decay Time (hr)"}}]
```

Mode	Decay Time (s)	Decay Time (hr)
1	27618.96	7.67193
2	6365.87	1.76830
3	2647.02	0.73528
4	1420.33	0.39454
5	879.05	0.24418
6	595.63	0.16545
7	429.51	0.11931
8	324.09	0.09002
9	253.10	0.07030
10	203.06	0.05641
11	166.49	0.04625
12	138.96	0.03860
13	117.72	0.03270
14	101.00	0.02806
15	87.60	0.02433
16	76.70	0.02131
17	67.71	0.01881
18	60.21	0.01673
19	53.90	0.01497
20	48.52	0.01348
21	43.91	0.01220
22	39.93	0.01109
23	36.46	0.01013
24	33.43	0.00929
25	30.76	0.00854
26	28.40	0.00789
27	26.30	0.00731
28	24.42	0.00678
29	22.74	0.00632
30	21.23	0.00590
31	19.86	0.00552
32	18.62	0.00517
33	17.49	0.00486
34	16.46	0.00457
35	15.52	0.00431
36	14.66	0.00407
37	13.87	0.00385
38	13.14	0.00365
39	12.47	0.00346
40	11.84	0.00329

■ 9. GRAPHS OF THE SOLUTION SHOWING APPROACH TO STEADY STATE

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 animated to show the evolution of the system with time. The graph at time t is named `snapshot[t]`. We use the decay times to determine how many terms to keep. We agree, somewhat arbitrarily, that $\text{Exp}[-6]$ or smaller is negligible. Then we define a function `nterms[t]` which tells us how many terms to keep, with the understanding that we can't keep more than 40 (because we have calculated only the first 40 coefficients), and we will always keep at least one.

```
nterms[t_] :=
  Module[{n}, n = 1; While[ ((t /  $\tau$ [n] < 6) && (n < 40)), n = n + 1]; n]
```

Thus, for example, for times of 0.25 hr = 900 s and 2.0 hr = 7200 s the number of terms needed are

```
nterms[900]
```

```
12
```

```
nterms[7200]
```

```
5
```

Now we use this to define in a practical way the sum of the series for Tr . We call this sum `Trans`, and the temperature we call `Temp`:

```
Trans[x_, t_] := ser[x, t, nterms[t]]
```

```
Temp[x_, t_] := Trans[x, t] + 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 determine a range of variation of temperature. We call this range `plrange`:

```
plrange = {0, 10};
```

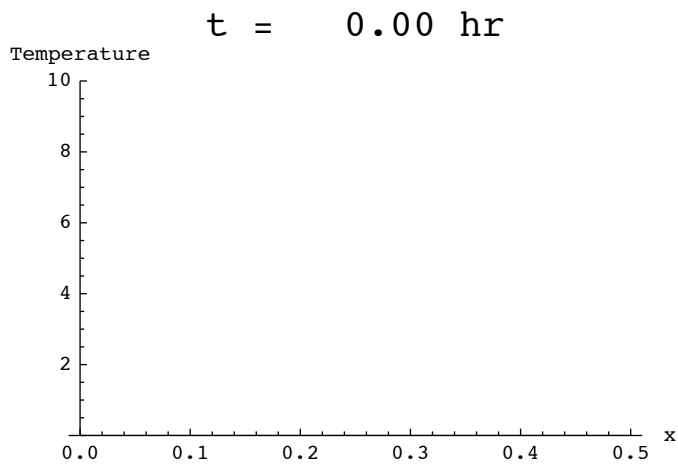
Now we define `snapshot[t]`, giving a special definition for `snapshot[0]` -- namely the initial distribution. For convenience, the function `snapshot` takes an argument in hours, but converts it internally to seconds, to be consistent with the units of the problem.

```
snapshot[t_] := Module[{time}, time = 3600 * t;
  Plot[Temp[x, time], {x, 0, L}, PlotRange -> plrange,
    AxesLabel -> {"x", "Temperature"}, ImageSize -> 250,
    PlotLabel -> Row[{"t = ", PaddedForm[t, {4, 2}], " hr"}]]]

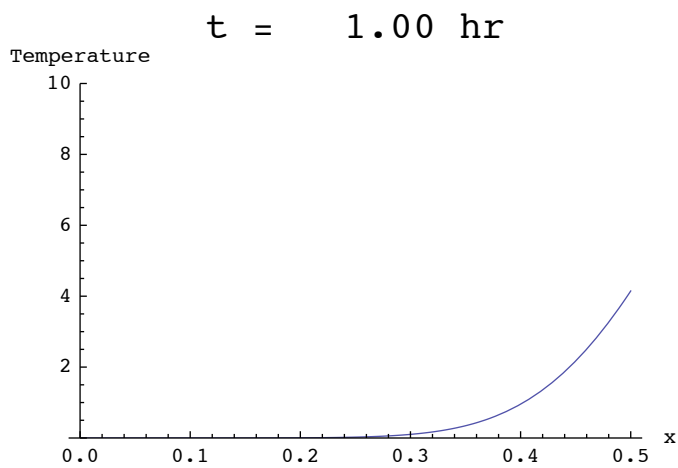
snapshot[0] := Plot[T0[x], {x, 0, L}, PlotRange -> plrange,
  AxesLabel -> {"x", "Temperature"}, ImageSize -> 250,
  PlotLabel -> Row[{"t = ", PaddedForm[0, {4, 2}], " hr"}]]]
```

We try it for the initial time and for 1 hr.

```
snapshot[0]
```

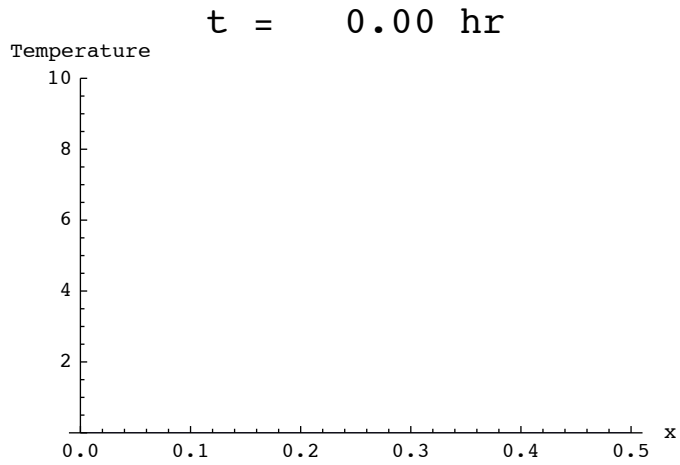


```
snapshot[1]
```



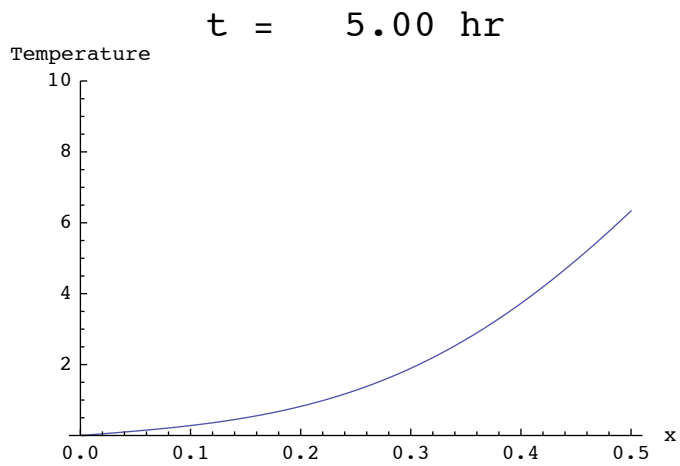
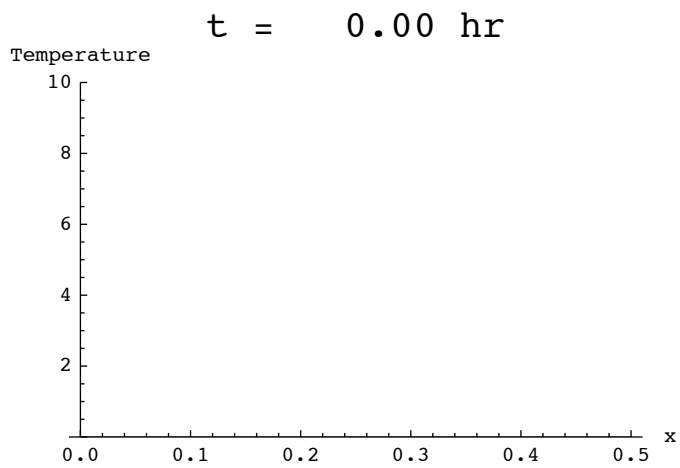
Now we produce a sequence of graphs that can be animated to show the dynamics of the heat loss process. We produce a sequence of 101 graphs at 0.20 hr intervals, running from 0 hr to 20 hr.

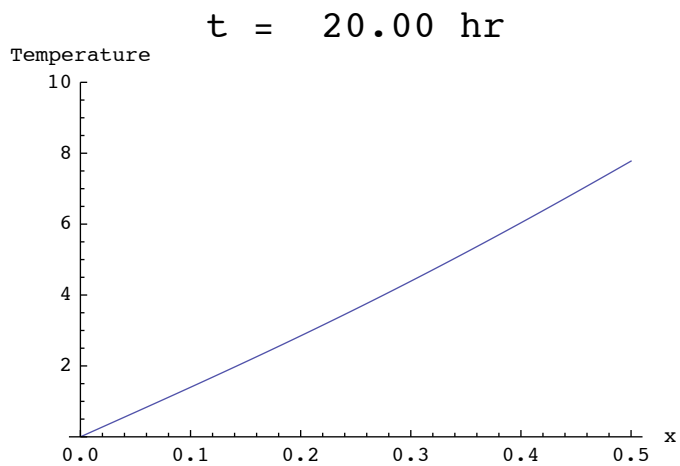
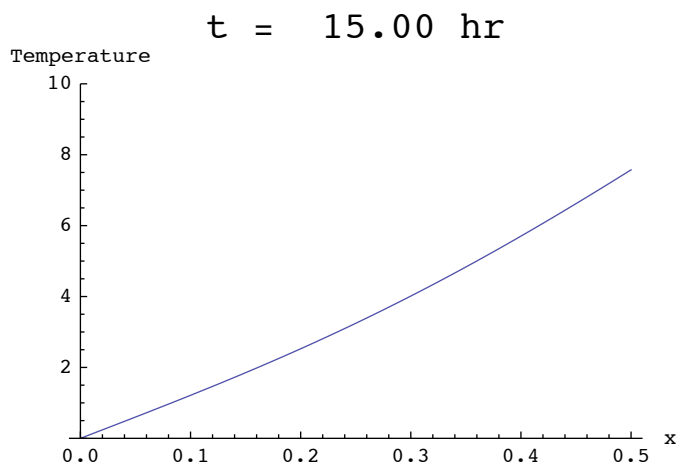
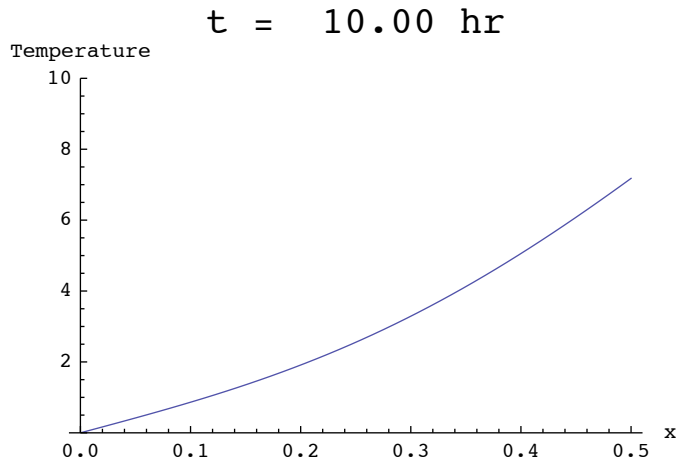
```
Do[Print[snapshot[0.2 * i]], {i, 0, 100}];
```



In the printed version, we show below only a few selected graphs of the sequence, namely at $t = 0, 5, 10, 15,$ and 20 hours.

```
Do[Print[snapshot[i]], {i, 0, 20, 5}];
```





■ 10. GRAPHS OF THE SOLUTION SHOWING THE FIRST TERM APPROXIMATION TO THE TRANSIENT SOLUTION

As our next task, we construct a series of graphs showing the full transient solution and the first term approximation to the transient. We call the graph `transhot[t]`.

```

plrange = {-10, 0};

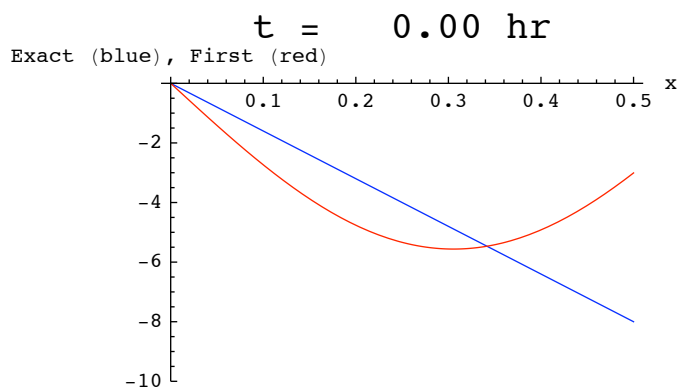
transhot[t_] := Module[{time}, time = 3600 * t;
  Plot[{Trans[x, time], firstterm[x, time]},
    {x, 0, L}, PlotRange -> plrange, ImageSize -> 250,
    PlotStyle -> {RGBColor[0, 0, 1], RGBColor[1, 0, 0]},
    AxesLabel -> {"x", "Exact (blue), First (red)"},
    PlotLabel -> Row[{"t = ", PaddedForm[t, {4, 2}], " hr"}]]]

transhot[0] :=
  Module[{time}, time = 0; Plot[{exactinit[x], firstterm[x, time]},
    {x, 0, L}, PlotRange -> plrange, ImageSize -> 250,
    PlotStyle -> {RGBColor[0, 0, 1], RGBColor[1, 0, 0]},
    AxesLabel -> {"x", "Exact (blue), First (red)"},
    PlotLabel -> Row[{"t = ", PaddedForm[0, {4, 2}], " hr"}]]]

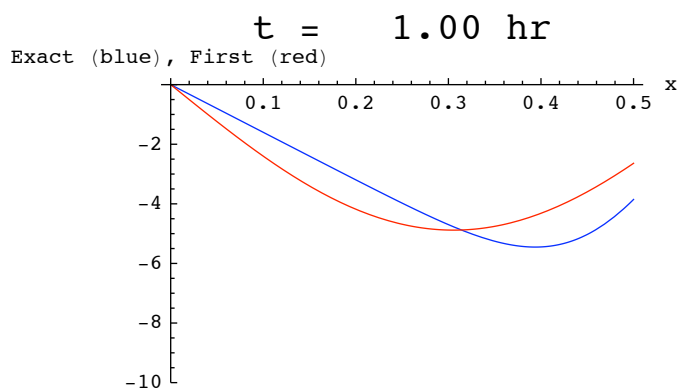
```

We try this out for the initial time and for a time of 1 hour.

`transhot[0]`

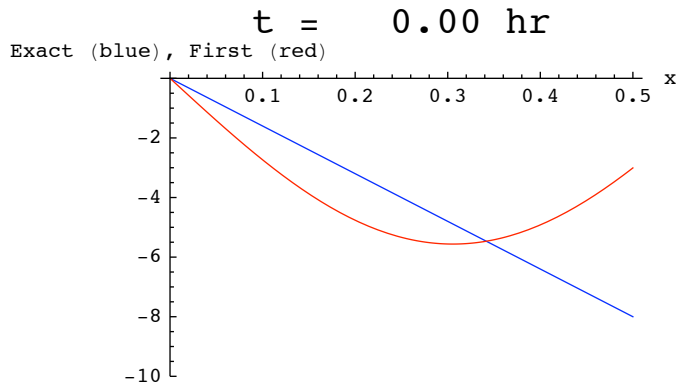


`transhot[1]`



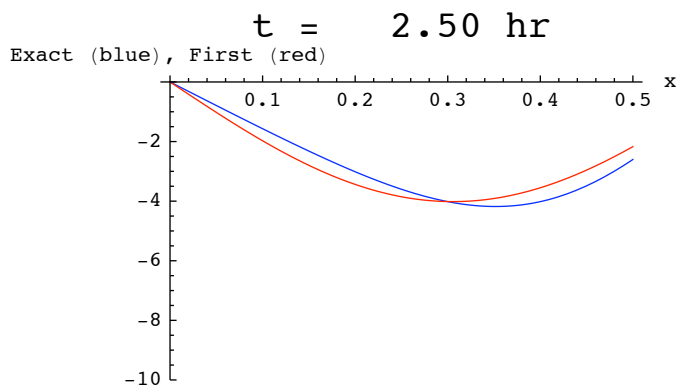
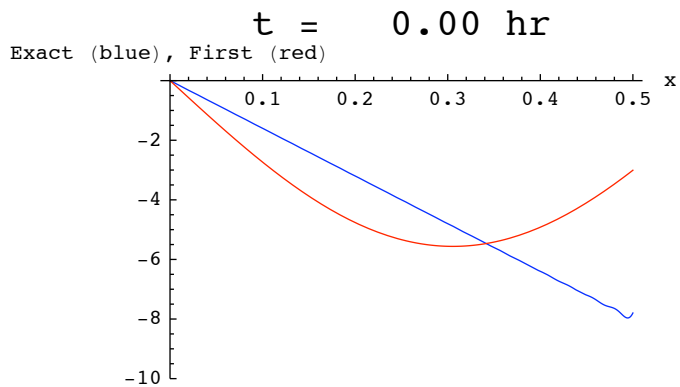
Now we construct a sequence running from 0 to 10 hr at 0.5 hr intervals.

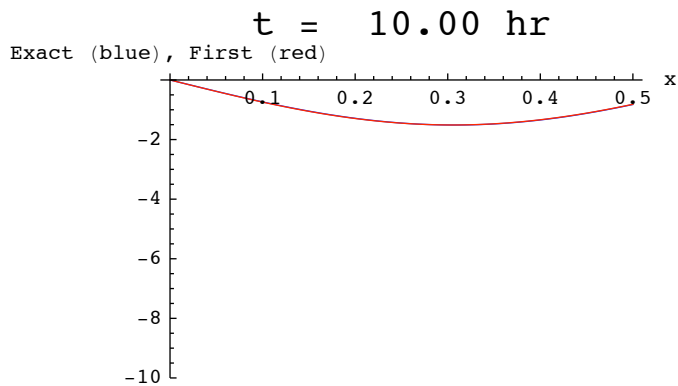
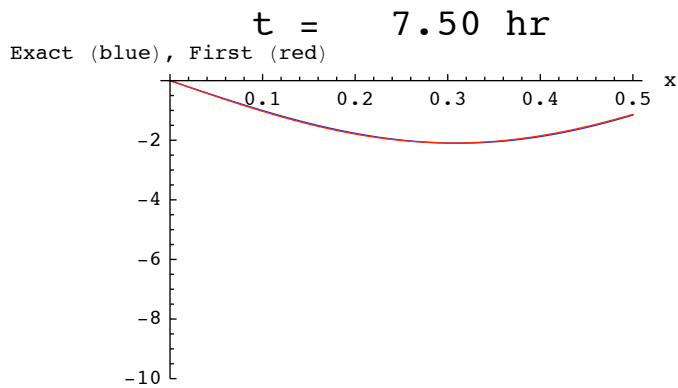
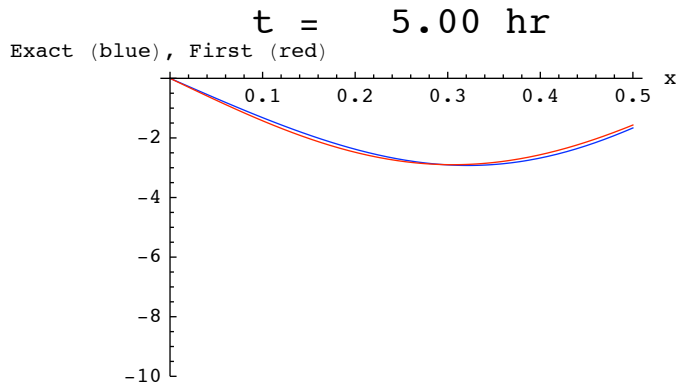
```
Do[Print[transhot[i * 0.5]], {i, 0, 20}];
```



We see that after about 6 hours, the difference between the full solution and the first term is very small. In the printed version of this notebook, we show below only a few selected graphs, namely for times of 0, 2.5, 5, 7.5, and 10 hours.

```
Do[Print[transhot[i]], {i, 0, 10, 2.5}];
```





■ Challenge Problem (d)

To get the decay time $1/\lambda D_f$ for an arbitrary h , we have to find the eigenvalue λ for an arbitrary h . As we saw earlier, we may express the eigenvalue as $\lambda = (z/L)^2$, where z is a root of

$$\tan(z) = -z/B_i .$$

In this formulation, we would proceed in the following steps: (1) calculate $B_i = (hL)/k$ from the given h ; (2) use FindRoot to calculate z_1 , that root z which is in the interval $(\pi/2, \pi)$; (3) calculate the decay time in hours from the formula below.

$$\frac{1}{(3600)\lambda D_f} = \frac{L^2}{3600 z_1^2 D_f}$$

The central part of this calculation is to find z_1 . Some experiments with that show that for the eigenvalue equation in the above form, it works well, except when h is near zero (and hence B_i is near zero). The problem is the B_i in the denominator. We can avoid this issue by simply rewriting the eigenvalue equation as

$$\cos(z) = -(B_i \sin(z))/z.$$

We use this form and write the routine to calculate z_1 given B_i . Then we write the module that goes directly from h to the decay time in hours.

```
z1[Bi_] := z /. FindRoot[Cos[z] == -(Bi Sin[z]) / z, {z, π / 2 + 0.02}]
```

We try this for $B_i = 0$, for which $z_1 = \pi/2$, and for B_i equal to

```
(h L / k)
```

```
4.
```

for which we found earlier that $z_1 = 2.5704$.

```
z1[0]
```

```
1.5708
```

```
z1[4.0]
```

```
2.57043
```

Looks like it is working. For very large B_i the value should approach π , as we discussed in the solution sheet. We try a few large values.

```
z1[100]
```

```
3.1105
```

```
z1[500]
```

```
3.13532
```

```
z1[1000]
```

```
3.13845
```

It is getting there, but slowly.

Now we write the module that gives us the decay time τ in hours from a given value of h in W/m \cdot K. First we clear the previously assigned value of h .

```
Clear[h];
```

```
tau[h_] := Module[{Bi, z, λ}, Bi = (h L) / k;
  z = z1[Bi]; λ = (z / L)2; 1 / (3600 * λ * Df)]
```

Let's check the value we know from our earlier calculations in problem 1. For $h = 22.4$, we got a decay time of 7.67193 hours.

```
tau[22.4]
```

```
7.67193
```

Looks good. Before making the plot, let's get the range of tau values for our given h range.

```
tau[0]
```

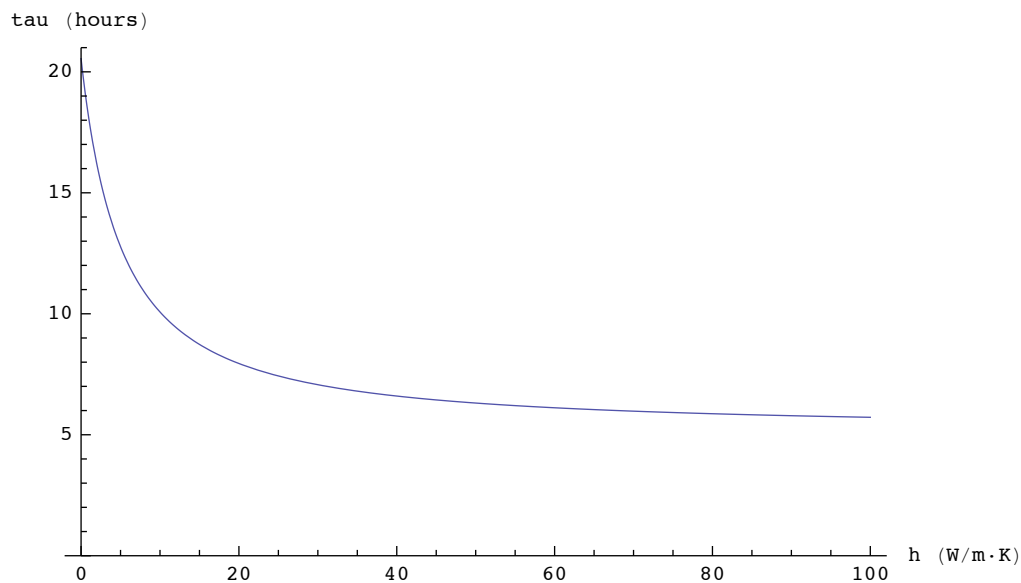
```
20.5436
```

```
tau[100]
```

```
5.7217
```

We will choose a plot range of 0 to 21 hours for tau.

```
Plot[tau[h], {h, 0, 100}, AxesLabel -> {"h (W/m·K)", "tau (hours)"},
  ImageSize -> 380, PlotRange -> {0, 21}]
```



The decay time decreases monotonically as the heat transfer coefficient increases, which we would expect. As h approaches infinity, the decay time approaches a minimum value which we can calculate, knowing that $z \rightarrow \pi$ as $h \rightarrow \infty$:

$$\frac{L^2}{3600 (\pi)^2 Df}$$

5.13591

This is about 10% lower than the value at $h = 100$.