Transient, One-Dimensional Heat Conduction in a Convectively Cooled Sphere

Gerald Recktenwald*

March 16, 2006†

1 Overview

This article documents the numerical evaluation of a well-known analytical model for transient, one-dimensional heat conduction. The physical situation is depicted in Figure 1. A sphere of uniform material is initially at a uniform temperature T_i. At time $t = 0$ the sphere is immersed in a stream of moving fluid at some different temperature T_∞. The external surface of the sphere exchanges heat by convection. The local heat flux from the sphere to the fluid is

$$q = h(T_s - T_\infty) \tag{1}$$

where h is the heat transfer coefficient, and T_s is the local surface temperature.

The problem is greatly simplified by assuming that the heat flux on the surface is uniform. Under this condition, $T_s = T(r_0)$ is also uniform and the temperature inside the sphere depends only on the radius, r, and time t, i.e.,

*Mechanical and Materials Engineering Department, Portland State University, Portland, OR, 97201, gerry@me.pdx.edu

†Corrections made 10 September 2011 and 18 July 2013

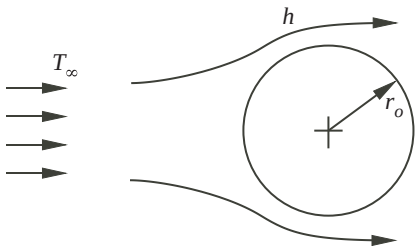

Figure 1: A sphere immersed in and exchanging heat with stream of fluid.

$T = T(r, t)$. The temperature field is governed by the heat equation in spherical coordinates

$$\frac{\partial T}{\partial t} = \frac{\alpha}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial T}{\partial r}\right) \tag{2}$$

where $\alpha = k/(\rho c)$ is the thermal diffusivity of the sphere material, k is the thermal conductivity, ρ is the density, and c is the specific heat. The boundary condition on the surface is

$$k\left.\frac{\partial T}{\partial r}\right|_{r=r_0} = h(T_\infty - T_s). \tag{3}$$

The initial condition is

$$T(r, 0) = T_i. \tag{4}$$

The remaining condition is that the temperature at all points in the sphere is bounded. These three conditions are sufficient to obtain a solution to Equation (2).

2 Analytical Solution

The analytical solution for Equation (2), subject to Equation (3), Equation (4), and the condition of bounded $T(r, t)$ is given in several heat transfer textbooks, e.g. [1, 2]. A universal solution is obtained in terms of the dimensionless variables

$$\theta^* = \frac{T - T_\infty}{T_i - T_\infty}, \qquad r^* = \frac{r}{r_o}, \qquad \mathrm{Fo} = \frac{\alpha t}{r_o^2}. \tag{5}$$

The dimensionless form of the boundary condition in Equation (3) is

$$\left.\frac{\partial \theta^*}{\partial r^*}\right|_{r^*=1} = \mathrm{Bi}\,\theta_s^* \tag{6}$$

where the *Biot* number is

$$\mathrm{Bi} = \frac{h r_o}{k} \tag{7}$$

and $\theta_s^* = (T_s - T_\infty)/(T_i - T_\infty)$. If $\mathrm{Bi} \ll 1$ the internal temperature gradient is small compared to the scaled difference between the surface temperature and the fluid. In that case the temperature in the sphere is spatially uniform.

The analytical solution is the infinite series

$$\theta^* = \sum_{n=1}^{\infty} C_n \exp(-\zeta_n^2 \mathrm{Fo})\frac{1}{\zeta_n r^*}\sin(\zeta_n r^*) \tag{8}$$

where

$$C_n = \frac{4\left[\sin(\zeta_n) - \zeta_n\cos(\zeta_n)\right]}{2\zeta_n - \sin(2\zeta_n)}. \tag{9}$$

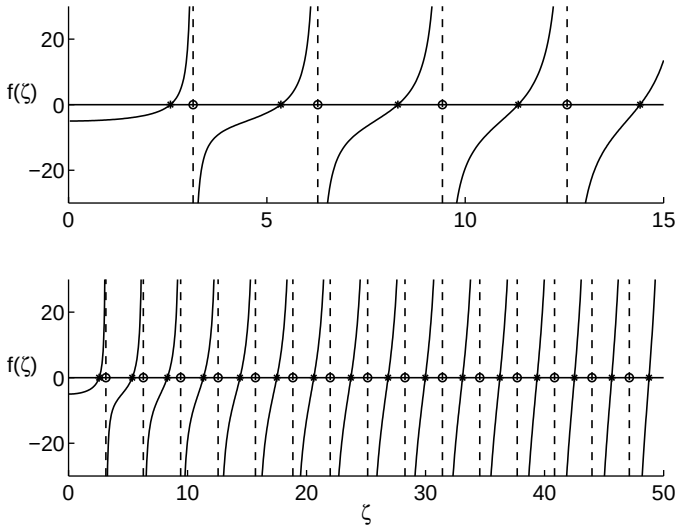

Figure 2: Plots of $f(\zeta) = 1 - \zeta \cot(\zeta) - \mathrm{Bi}$ for $\mathrm{Bi} = 5$ showing roots ($*$) and singularities ($\circ$) of Equation (10). Upper plot shows the range $0 \leq \zeta \leq 15$. Lower plot shows the range $0 \leq \zeta \leq 50$.

The ζ_n are the positive roots of

$$1 - \zeta_n \cot(\zeta_n) = \mathrm{Bi}. \tag{10}$$

Equations (8) through (10) provide a compact representation of the solution. Obtaining numerical values from these formulas is a nontrivial effort except in the case where $\mathrm{Fo} \gg 1$ when a one term approximation is sufficient. In general, at least the first few terms terms in the series are needed.

3 Evaluating the Solution in MATLAB

In this section, a procedure for evaluating Equation (8) is presented. The MATLAB programs, or (for long programs) the function prologues, are listed in the appendix.

3.1 Finding the Roots of Equation (10)

The first task is to find the roots, ζ_n, of Equation (10) for a given Bi. This is a standard root-finding problem, with two important complications. First, there are an infinite number of ζ_n on the positive real line. To evaluate Equation (8) a large number of roots must be available and sorted in order of increasing ζ_n. A missing root will cause erroneous evaluation of the series. The second complication is that Equation (10) has a singularity between successive roots.

Figure 2 shows the function $f(\zeta) = 1 - \zeta\cot(\zeta) - \mathrm{Bi}$ for $\mathrm{Bi} = 5$ in the range $0 \le \zeta \le 50$. Note that $f(\zeta) = 0$ when ζ is a root of Equation (10). The top half of Figure 2 shows $f(\zeta)$ (for $\mathrm{Bi} = 5$) in the range $0 \le \zeta \le 15$. The roots are identified with the `*` symbol, and the singularities are identified by vertical dashed lines and the `o` symbol. The bottom half of Figure 2 shows $f(\zeta)$ over the larger range $0 < \zeta \le 50$. As ζ increases the spacing between roots approaches a constant, and the roots are located midway between the singularities. At small ζ the spacing between the roots is not equal.

The `zRoots` function in Listing 1 uses the `zfun` function (Listing 2), the `bracket` function (Listing 3), and the built-in `fzero` function to find the roots of Equation (10). `fzero` is a very efficient root-finding routine, but it converges equally well to roots and singularities. The strategy used in `zRoots` is to first use `fzero` find all the roots *and* singularities in order of increasing ζ, and then discard the singularities. A singularity $\hat{\zeta}$ is easily identified because $f(\hat{\zeta})$ is large.

The `zRoots` function can be called with two optional input arguments. The first input, `Bi`, is required. The first optional input, `zmax`, specifies the upper limit of the range of ζ to search for roots. The second optional input, `verbose`, specifies whether the roots and singularities are printed, and whether a plot of $f(\zeta)$ is created. For example, the top plot in Figure 2 is obtained with

```
>> z = zRoots(5,15,true);
5 good zeta values found in zRoots()
4 potential zeros are suspected to be singularities

All roots found by fzero
   k     zleft       z        zright       f(z)
   1    2.5301     2.5704     2.5904     3.128e-08
   2    3.1325     3.1416     3.1928     6.740e+07
   3    5.3012     5.3540     5.3614     2.485e-11
   4    6.2651     6.2832     6.3253    -1.015e+08
   5    8.2530     8.3029     8.3133     3.864e-12
   6    9.3976     9.4248     9.4578     9.273e+07
   7   11.3253    11.3348    11.3855    -1.298e-07
   8   12.5301    12.5664    12.5904    -1.450e+08
   9   14.3976    14.4080    14.4578    -2.699e-07

Good roots retained by zRoots
   n      z         f(z)
   1    2.5704     3.128e-08
   3    5.3540     2.485e-11
   5    8.3029     3.864e-12
   7   11.3348    -1.298e-07
   9   14.4080    -2.699e-07

Singularities eliminated from list of roots
   n     zbad       f(zbad)
   2    3.1416     6.740e+07
   4    6.2832    -1.015e+08
   6    9.4248     9.273e+07
   8   12.5664    -1.450e+08
```

Note that the roots and singularities are interleaved, and that `zRoots` correctly identifies the roots.

In practice, a plot of the roots is not always necessary or desirable. To obtain only the ζ_n for $\mathrm{Bi} = 5$ in the range $0 \le \zeta \le 50$ use

```
>> z = zRoots(5,50);
16 good zeta values found in zRoots()
15 potential zeros are suspected to be singularities
```

To facilitate manual calculation, Incropera and DeWitt [2] provide a list of the first roots of Equation (10) for a range of Bi values[1]. The `zRootFirst` function produces a similar list of ζ_1 values. There are some discrepancies in the ζ_1 values created by `zRootFirst` and the list given by Incropera and DeWitt. The first few rows of output from `zRootFirst` are

```
>> zRootFirst
      Bi       z1       f(z1)
    0.01    0.1730   -2.064e-14
    0.02    0.2445   -9.368e-17
    0.03    0.2991   -7.494e-16    (0.2989)
    0.04    0.3450   -4.517e-15
    0.05    0.3854   -2.179e-15    (0.3852)
    0.06    0.4217    4.857e-17
    0.07    0.4551   -1.665e-16    (0.4550)
    0.08    0.4860   -8.042e-14
     ⋮        ⋮         ⋮
```

The third column gives $f(z_1) = 1 - \zeta_n \cot(\zeta_n) - \mathrm{Bi}$ as a check on whether z_1 satisfies Equation (10). The numbers in parenthesis to the right of the third column are ζ_1 values from Incropera and DeWitt that differ from those created by `zRootFirst`. (Those values are not printed by the MATLAB function.) The discrepancies are small (all less than 0.07 percent) and are likely due to rounding errors in the calculations used to produce the table in the textbook by Incropera and DeWitt. The MATLAB calculations with `zRootFirst` are all in double precision[2]. The `z1` values produced by `zRootFirst` all give very small residuals, and therefore are accurate.

3.2 Evaluating $\theta^*(\mathrm{Bi}, Fo)$

Once the ζ_n are known for a given Bi, the dimensionless temperature is computed from Equation (8). The `Tsphere` function in Listing 5 calls `zRoots` to find the required ζ_n, and then evaluates Equation (8). In other words, the user only needs to call `Tsphere` to obtain numerical values of θ^*.

`Tsphere` has a number of optional input and output arguments so that it can be called in the following ways.

```
T = Tsphere(Bi,Fo)
T = Tsphere(Bi,Fo,nr)
T = Tsphere(Bi,Fo,nr,verbose)
[T,r] = Tsphere(...)
```

[1] For large t a one-term approximation to Equation (8) is sufficient.
[2] Roughly fifteen decimal digits of accuracy.

The required inputs are `Bi`, the Biot number, and `Fo`, the Fourier number. The `Fo` input can be a scalar or vector. The third input `nr` is optional and specifies the number of radial positions at which θ^* is to be evaluated. If `nr`$= 1$ then only θ^* at $r^* = 0$ is evaluated. If `nr`> 1 then θ^* is evaluated at `nr` equally spaced values on the interval $0 \le r^* \le 1$.

The default output `T` is a matrix of θ^* values. If `Fo` is a vector, each column of `T` corresponds to an element in `Fo`. If `nr`> 1 the rows of `T` correspond to the `nr` equally spaced r^* values.

For example, to evaluate $\theta^*(r^*, \mathrm{Fo})$ at 11 uniformly spaced radial locations for $\mathrm{Fo} = [0.1, 0.15, 0.2]$ and $\mathrm{Bi} = 5$, use

```
>> [T,r] = Tsphere(5,[0.1 0.15 0.2],11,true);
16 good zeta values found in zRoots()
15 potential zeros are suspected to be singularities

Dimensionless Temperature Profile
    dTmax = 9.65003e-105

         |  Fo =
    r*   |  0.1000  0.1500  0.2000
-----------------------------------
  0.0000 |  0.8459  0.6447  0.4722
  0.1000 |  0.8393  0.6383  0.4672
  0.2000 |  0.8194  0.6192  0.4523
  0.3000 |  0.7858  0.5879  0.4281
  0.4000 |  0.7380  0.5451  0.3953
  0.5000 |  0.6758  0.4921  0.3551
  0.6000 |  0.5997  0.4303  0.3091
  0.7000 |  0.5114  0.3619  0.2587
  0.8000 |  0.4136  0.2892  0.2060
  0.9000 |  0.3102  0.2151  0.1527
  1.0000 |  0.2059  0.1423  0.1009
```

3.3 Evaluating $T(r,t)$

The MATLAB programs presented in this article produce dimensionless results. To solve a problem in dimensional units follow these steps

1. Convert the dimensional inputs to Bi and Fo.
2. Obtain the dimensionless solution $\theta^*(\mathrm{Bi}, \mathrm{Fo})$.
3. Compute the temperature from the definition of θ^*, viz., $T(r,t) = T_\infty + \theta^*(T_i - T_\infty)$.

3.3.1 Example

(From Incropera and DeWitt) A 5 mm radius metal sphere initially at 400 °C is plunged into a water bath at 50 °C. Assume that the heat transfer coefficient is $6000\,\mathrm{W/(m^2\,°C)}$, the thermal conductivity of the metal is $k = 20\,\mathrm{W/m\cdot K}$, and the thermal diffusivity of the metal is $\alpha = 6.66 \times 10^{-6}\,\mathrm{m^2/s}$. Plot the variation of the centerline temperature with time for the first five seconds after

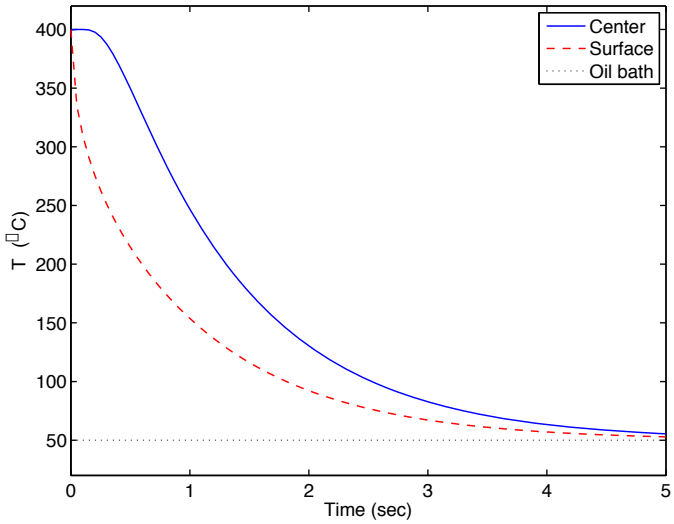

Figure 3: Center and surface temperatures versus time for Example 3.3.1

the sphere is plunged into the water bath. Plot the radial temperature variation at $t = 0.5, 1$, and 5 seconds.

The solution is implemented in the `sphereExample` function in Listing 6. Figure 3 shows the variation of the center temperature with time. Figure 4 shows the radial temperature variation at $t = 0.5, 1$, and 5 seconds.

References

[1] E. Eckert and J. Drake, Robert M. *Analysis of Heat and Mass Transfer*. McGraw-Hill, New York, 1972.

[2] F. P. Incropera and D. P. DeWitt. *Introduction to Heat Transfer*. Wiley, New York, fifth edition, 2002.

Appendix: Listings of MATLAB Codes

The MATLAB source code for the mfiles described in the article are listed on the following pages. Two of the mfiles (`zRoots` and `Tsphere`) are too long to fit on a single page. For these functions the last lines of code are omitted. The reader is encouraged to download the complete code from `http://web.cecs.pdx.edu/~gerry/epub/`. The code omitted from `zRoots` and `Tsphere` deals with optional printing of results, so the listings provided here allow the reader to study the essential parts of the calculations.

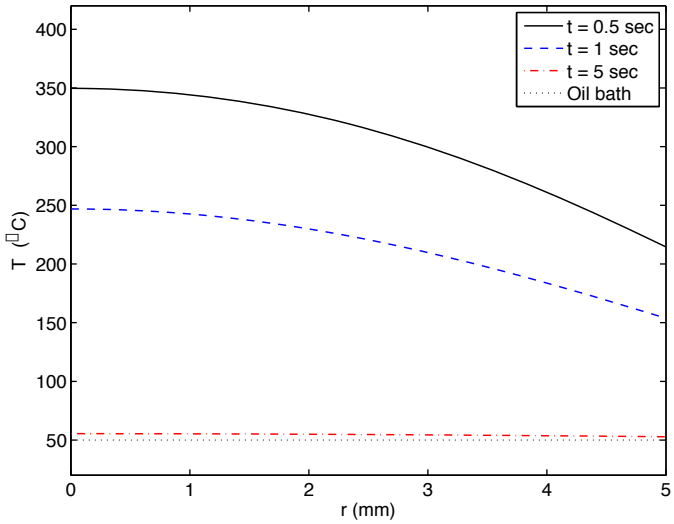

Figure 4: Radial temperature variation at three times for Example 3.3.1

```
function z = zRoots(Bi,zmax,verbose)
% zRoots  Find all roots to 1 - z*cot(z) - Bi over a range of z
%
% Synopsis:  z = zRoots
%            z = zRoots(Bi)
%            z = zRoots(Bi,zmax)
%            z = zRoots(Bi,zmax,verbose)
%
% Input:     Bi = Biot number.  Default:  Bi = 10
%            zmax = upper limit of a range.  Roots are sought in the
%                   range 0 < zeta <= zmax.  Default: zmax=50
%            verbose = flag to control print out
%                      Default:  verbose = 0  (no extra print out)
%
% Output:    z = Vector of roots in the interval 0 < z <= zmax

if nargin<1  isempty(Bi),       Bi = 10;      end
if nargin<2  isempty(zmax),     zmax = 50;    end
if nargin<3  isempty(verbose),  verbose = 0;  end

% --- Find brackets for zeros of 1 - z*cot(z) - Bi
zb = bracket(@zfun,10*eps,zmax,250,Bi);   %  zb is a 2 column matrix

% --- Find the zero (or singularity) contained in each bracket pair
mb,nb = size(zb);  zall = zeros(mb,1);  %  Preallocate array for roots
%  Call optimset to create the data structure that controls fzero
%  Use no messages ('Display','Off') and tight tolerance ('TolX',5e-9)
fzopts = optimset('Display','Off','TolX',5e-9);
for k=1:mb
  zall(k) = fzero(@zfun,zb(k,:),fzopts,Bi);
end

% --- Sort out roots and singularities.  Singularites are "roots"
%     returned from fzero that have f(z) greater than a tolerance.
fall = zfun(zall,Bi);          %  evaluate f(z) at all potential roots
igood = find(abs(fall)<5e-4);     %  vector of indices of good roots
ngood = length(igood);
z = zall(igood);   f = fall(igood);
zbad = zall(:);    ibad = (1:length(zbad))';  %  First copy all data
zbad(igood) = ;  ibad(igood) = ;          %  then throw away good parts
nbad = length(ibad);
fprintf('%d good zeta values found in zSphereRoots()\n',ngood);
if nbad>0
  fprintf('%d potential zeros are suspected to be singularities\n',nbad);
end
.
.
.
%  Omitted code for printing and plotting of roots
```

Listing 1: Partial listing of the `zRoots` function.

```
function f = zfun(z,Bi)
% ztest  Evaluate f = 1 - z*cot(z) - Bi for root-finding algorithm
f = 1 - z.*cot(z) - Bi;
```

Listing 2: The `zfun` function evaluates $f(z) = 1 - z\cot(z) - \mathrm{Bi}$. This formula is used with the built-in `fzero` function to find the roots (zeros) of Equation (10).

```
function xb = bracket(fun,xmin,xmax,nx,varargin)
% brackPlot  Find brackets for roots of a function.
%
% Synopsis:  xb = bracket(fun,xmin,xmax)
%            xb = bracket(fun,xmin,xmax,nx)
%
% Input:  fun = (string) name of function for which roots are sought
%         xmin,xmax = endpoints of interval to subdivide into brackets.
%         nx = (optional) number of subintervals.  Default:  nx = 20.
%
% Output:  xb = 2-column matrix of bracket limits.  xb(k,1) is the left
%               bracket and xb(k,2) is the right bracket for the kth
%               potential root.  If no brackets are found, xb = [].

if nargin<4, nx=20; end

x = linspace(xmin,xmax,nx);    %  Test f(x) at these x values
f = feval(fun,x,varargin{:});
nb = 0;
xbtemp = zeros(nx,2);   %  Temporary storage for brackets as they are found

for k = 1:length(f)-1
  if sign(f(k))~=sign(f(k+1))  %  True if f(x) changes sign in interval
    nb = nb + 1;
    xbtemp(nb,:) = x(k:k+1);
  end
end

% -- Return nb-by-2 matrix of brackets.
if nb == 0
  warning('bracket:NoSignChange','No brackets found. Change [xmin,xmax] or nx');
  xb = [];
else
  xb = xbtemp(1:nb,:);
end
```

Listing 3: The `bracket` function returns a list of intervals in which a function changes sign. This utility function is used to obtain initial guesses for root-finding algorithms.

```
function [zout,Biout] = zRootFirst
% zRootFirst  List smallest roots of 1 - z*cot(z) = Bi for a range of Bi.
%             Compare to Table 5.1 in _Introduction to Heat Transfer_,
%             2nd ed., F.P. Incropera & D.P. De Witt, 1990, Wiley
%
% Synopsis:  zRootFirst
%            z1 = zRootFirst
%            [z1,Bi] = zRootFirst
%
% Input:  none
%
% Output:  z1 = vector of first roots, i.e. roots closest to z=0,
%              of 1 - z*cot(z) = Bi for a range of Bi
%          Bi = vector of Bi values corresponding to z1 values

% --- Use Bi values from Table 5.1 by Incropera and Dewitt
Bi = [0.01:0.01:0.1, 0.15:0.05:0.30, 0.4:0.1:1, 2:1:10, 20:10:50, 100]';
z = zeros(size(Bi));

%  Call optimset to create the data structure that controls fzero
%  Use no messages ('Display','Off') and tight tolerance ('TolX',5e-9)
fzopts = optimset('Display','Off','TolX',5e-12);
zmax = 4;
fprintf('      Bi       z1       f(z1)\n');
for i=1:length(Bi)
  zb = bracket(@zfun,10*eps,zmax,100,Bi(i));  %  2 column matrix
  z(i) = fzero(@zfun,zb(1,:),fzopts,Bi(i));   %  Find root in 1st bracket
  fprintf(' %8.2f  %8.4f  %11.3e\n',Bi(i),z(i),zfun(z(i),Bi(i)));
end
if nargout>0, zout=z(:);    end   %  Optional return arguments
if nargout>1, Biout=Bi(:);  end   %  insure that both are column vectors
```

Listing 4: The `zRootFirst` function finds the first root of Equation (10) for a set of Bi in the range $0.01 \leq \text{Bi} \leq 100$.

```
function T,r = Tsphere(Bi,Fo,nr,verbose)
% Tsphere  Dimensionless T(r,t) for convective cooling of a sphere
%
% Synopsis:  T = Tsphere(Bi,Fo)
%            T = Tsphere(Bi,Fo,nr)
%            T = Tsphere(Bi,Fo,nr,verbose)
%            T,r = Tsphere(...)
%
% Input:  Bi = scalar, Biot number for the sphere
%         Fo = scalar or vector of Fourier numbers (dimensionless time)
%         nr = number of r values at which to evaluate T(r,t).  If nr=1
%              only r=0 is used.  Default: nr=1
%         verbose = flag to control printing.  Default: verbose = false
%
% Output:  T = matrix of dimensionless temperatures (theta).  Column j
%              T(:,j) is a vector of theta values at the nr dimensionless
%              radial locations uniformly spaced in 0 <= rstar <= 1.
%              T(:,j) is the profile at dimensionless time Fo(j).
%          r = dimensionless radial locations: 0 <= r <= 1
%          verbose = flag to control printing.  Default: verbose = false

if prod(size(Bi))>1,  error('Bi must be a scalar');  end
if nargin<3,          nr=1;           end
if nargin<4,          verbose=false;  end

% --- Find zeta in range 0<zeta<=50, and compute coefficients of series
zeta = zRoots(Bi,50);
c = 4*(sin(zeta) - zeta.*cos(zeta))./(2*zeta - sin(2*zeta));

% --- Special handling for nr=1 to avoid creation of a vector
if nr==1
  rstar = eps;                %  Use eps instead of zero to avoid
else                          %  division by zero in formula for T
  rstar = linspace(eps,1,nr)'; %  rstar must be a columnn vector
end
% --- Vectorized loop to evaluate theta = f(Bi,Fo).  T is nr-by-nf
%     matrix, where nf is number of radial locations and nf is number
%     of Fo values.  Construct T by summing outer products of column
%     vector (sin(zeta1*rstar)./(zeta1*rstar)) with row vector
%     exp(-zeta1^2*Fo).
rstar = linspace(2*eps,1,nr)'; %  Avoid rstar exactly zero to avoid
                               %  division by zero in formula for T
Fov = Fo(:)';   %  Local copy of Fo, guaranteed to be a row vector
T = c(1)*(sin(zeta(1)*rstar)./(zeta(1)*rstar))*exp(-zeta(1)^2*Fov);
for k=2:length(zeta)
  dT = c(k)*(sin(zeta(k)*rstar)./(zeta(k)*rstar))*exp(-zeta(k)^2*Fov);
  dTmax = max(max(abs(dT)));
  T = T + dT;
end
if dTmax > 0.005
  warning(sprintf('Series not converged: dTmax = %g\n',dTmax));
end
⋮
%  Omitted code for printing of T(r,t) table
```

Listing 5: Partial listing of the `Tsphere` function.

```
function sphereExample
% sphereExample  T(r,t) for a metal sphere plunged into a water bath

% --- Specify constants
h = 6000;        %  heat transfer coefficent,  W/m^2/C
k = 20;          %  thermal conductivity,  W/m/K
alfa = 6.66e-6;  %  thermal diffusivity,  m^2/s
ro = 5e-3;       %  radius of sphere,  m
tmax = 5;        %  stop time, s

r = linspace(0,ro);
t = linspace(0,tmax);

% --- Compute theta at center and at the surface of the sphere.
%     Characteristic length is r0 for the exact solution.
Bi = h*ro/k
Fo = alfa*t/ro^2;
theta = Tsphere(Bi,Fo,2);  %  nr = 2 for r*=1 and r*=1

% --- Convert to temperature and plot
Ti = 400;   Tinf = 50;
T = Tinf + theta*(Ti-Tinf);
plot(t,T(1,:),'b-',t,T(2,:),'r--',[0 max(t)],[Tinf Tinf],'k:');
axis([0 max(t) 20 Ti+20]);
xlabel('Time (sec)');    ylabel('T  ({}^\circ C)');
legend('Center','Surface','Oil bath','Location','NorthEast');

% --- Evaluate and plot the radial temperatures at t=0.5, 1, 5
t = [0.5 1 5]
Fo = alfa*t/ro^2;
[theta,rstar] = Tsphere(Bi,Fo,50);
T = Tinf + theta*(Ti-Tinf);
rmm = 1000*rstar*ro;
figure
plot(rmm,T(:,1),'k-',rmm,T(:,2),'b--',rmm,T(:,3),'r-.',...
     [0 max(t)],[Tinf Tinf],'k:');
legend('t = 0.5 sec','t = 1 sec','t = 5 sec','Oil bath','Location','NorthEast');
xlabel('r (mm)');    ylabel('T  ({}^\circ C)');
axis([0 max(rmm) 20 Ti+20]);
```

Listing 6: The `sphereExample` function performs the sample calculations described in § 3.3.1.