X. Analytical Solution of PDEs

Introduction

The topic of PDEs is probably the most important subject in applied engineering math. This is true from the physical viewpoint because PDEs result from the mathematical modeling of real engineering/physical systems. From the pure math perspective, the solution of PDEs brings together much of one's knowledge of advanced mathematics -- ODEs, eigenvalue problems, orthogonality, Laplace and Fourier Transforms, linear algebra, numerical methods, etc.. In addition, there is a rich variety of interesting applications from which to select examples to illustrate the various methods. Thus, not only is this subject of fundamental importance, it can also be quite rewarding when all your mathematics background comes together to yield interesting solutions to challenging real-world problems.

In our relatively brief overview of PDE solution methods (note that one could easily spend a full semester or more on this subject), we will find that certain techniques are appropriate for certain classes of problems (this is especially true for the numerical solution techniques discussed in Section XI). Thus, it is often useful to classify various PDEs into one or more categories, and then discuss solution methods appropriate for the various classes that have been defined. The most common classification scheme in use deals with 2nd order *Quasi-Linear* systems that are defined by

$$A(x,y)\frac{\partial^2}{\partial x^2}u + B(x,y)\frac{\partial^2}{\partial x \partial y}u + C(x,y)\frac{\partial^2}{\partial y^2}u = \phi\left(u, \frac{\partial}{\partial x}u, \frac{\partial}{\partial y}u, x, y\right) \qquad (10.1)$$

where u(x,y) is the dependent variable, x and y are the two independent variables, the system can have variable coefficients, A, B, and C, and the RHS function, ϕ, can be just about any function of interest (linear or nonlinear). The term quasi-linear is used because the left hand side (LHS) of eqn. (10.1) in linear in the dependent variable, but the RHS function may not be.

Equation (10.1) is often written using a shorthand notation for the partial derivatives, or

$$Au_{xx} + Bu_{xy} + Cu_{yy} = \phi(u, u_x, u_y, x, y) \qquad (10.2)$$

where, in general, the coefficients A, B, and C are functions of x and y, and the u_{xx}, u_{xy}, and u_{yy} second-order partial derivatives are defined explicitly by

$$u_{xx} = \frac{\partial^2}{\partial x^2}u \qquad u_{xy} = \frac{\partial^2}{\partial x \partial y}u \qquad \text{and} \qquad u_{yy} = \frac{\partial^2}{\partial y^2}u$$

We will use this shorthand notation throughout our treatment of PDE solution methods.

The most common PDE classification scheme identifies the PDE as either a *hyperbolic*, *parabolic*, or *elliptic* equation depending on the sign of the term B^2 - 4AC (which can vary with x and y). In particular, we have the following classification scheme:

$$B^2 - 4AC \begin{cases} > 0 & \text{hyperbolic} \\ = 0 & \text{parabolic} \\ < 0 & \text{elliptic} \end{cases}$$

These types of systems give rise to significantly different characteristic behavior and, as mentioned above, the solution scheme for each method can also differ. An example of each type of PDE is summarized below:

Application	Differential Equation	Value of Coefficients	Sign of $B^2 - 4AC$	PDE Class
Wave Equation	$u_{tt} = \alpha^2 u_{xx}$	$A = 1, B = 0, C = -\alpha^2$	positive	hyperbolic
Diffusion Equation	$u_t = \alpha^2 u_{xx}$	$A = 0, B = 0, C = -\alpha^2$	zero	parabolic
Poisson's Equation	$u_{xx} + u_{yy} = f(x, y)$	$A = 1, B = 0, C = 1$	negative	elliptic

Methods for solution of PDEs are best demonstrated by examples. This statement is probably applicable for most subjects in science, math, and engineering courses, but it is particularly true here. Thus, our approach, after a very brief overview of the theory, will be to simply do, in detail, as many problems as possible.

With this overall direction in mind, the organization for the remainder of this section of notes is given below:

The Separation of Variables Method

- The Classical Separation of Variables Method

- Treatment of Inhomogeneous Equations and Boundary Conditions

- The Eigenfunction Expansion Method

Analytical Solutions using Separation of Variables

- **Example 10.1 -- Heat Transfer in a Finite Bar with Homogeneous BCs**

- **Example 10.2 -- Heat Transfer in a Finite Bar with Fixed End Temperatures**

- **Example 10.3 -- Heat Transfer in a Finite Bar with Time Dependent BCs**

- **Example 10.4 -- Heat Transfer in a Cylindrical Bar with Homogeneous BCs**

- **Example 10.5 -- Heat Transfer in a 2-D Block with One Convective Surface**

Overview of Integral Transform Methods (not available)

Analytical Solutions using Integral Transforms (not available)

The Separation of Variables Method

The Classical Separation of Variables Method

The Separation of Variables (SOV) method represents one of the most powerful and most used analytical techniques for solving a variety of PDEs. In its traditional form, however, *it can only be applied directly to linear homogeneous problems with homogeneous boundary conditions*. The basic idea is to assume that the original function of two variables can be written as a product of two functions, each of which is only dependent upon a single independent variable -- thus the name *Separation of Variables* identifies precisely the technique for initiating the problem solution. The separated form of the solution is inserted into the original linear PDE and, after some manipulation, one obtains two homogeneous ODEs that can be solved by traditional means. If the original boundary conditions (BCs) for the problem are homogeneous, one of the ODEs will give a Sturm-Liouville type problem, which leads to a set of orthogonal eigenfunctions as solutions. Because the original PDE is linear, its final solution is formed as a linear combination of the individual solutions -- which gives rise to a solution written in the form of an infinite series. A final condition imposed on the problem (either an initial condition or a remaining BC that has not yet been used) is used to determine the unknown expansion coefficients in the infinite series solution. Since the basis functions are orthogonal, these coefficients are readily determined as illustrated previously (see Section VIII and/or Section IX). The analytical solution is complete once the coefficients have been determined. However, since the solution is still written in the form of an infinite series expansion, it is often evaluated and plotted using computer techniques -- thus completing the overall problem.

This basic procedure for the SOV technique is best illustrated by example. In particular, Example 10.1 solves a 1-D transient heat transfer problem by following the above outline for the classical Separation of Variables method, and the final infinite series solution is evaluated and plotted within Matlab. It represents a fundamental, straightforward demonstration of the basic technique.

A similar problem is given in Example 10.4. This situation also models transient heat transfer in a 1-D configuration but, this time, the geometry involves cylindrical coordinates. As seen before, cylindrical geometry problems often lead to Bessel function solutions, and this is indeed the situation here. Thus, Example 10.4, in addition to illustrating the classical SOV scheme, also gives another example of the importance of Bessel functions in typical engineering applications.

Examples 10.1 and 10.4 both involve transient heat transfer in a 1-D geometry, and this situation is described by a diffusion-type equation, which falls into the parabolic PDE classification. As an example of the solution to an elliptic PDE, Example 10.5 addresses the steady state heat transfer process in a 2-D rectangular block. An energy balance for this situation gives Laplace's equation since there is no internal heat source in the problem (Laplace's equation is the same as Poisson's equation with f(x,y) = 0). The common thread here is that this problem is linear, homogeneous, and one of the dimensions has homogeneous BCs -- thus, the classical SOV method is applicable. These three problems, Examples 10.1, 10.4, and 10.5, should give the reader a good overview of how to apply the classical SOV method to a variety of problems.

Treatment of Inhomogeneous Equations and Boundary Conditions

If the either the original linear PDE or the BCs have inhomogeneous terms, the classical Separation of Variables (SOV) method will not work. However, one technique that is often successful is to break the desired solution into two components -- one that is a function of both independent variables and one that is only a function of one of the variables. More specifically, one can often convert the original PDE into two problems; a homogeneous PDE with homogeneous boundary conditions (BCs) that can be solved by the SOV method, and an ODE that can be solved by traditional methods.

As a particular example, consider the mathematical description of transient heat conduction in a 1-D laterally insulated bar. This situation is governed by

$$u_t = \alpha u_{xx} + Q(x) \tag{10.3}$$

where $Q(x)$ could account for an internal heat generation term in the energy balance (this term makes the parabolic PDE inhomogeneous). If, in addition, fixed endpoint temperatures are imposed (i.e. inhomogeneous BCs), we have BCs of the form

$$u(0,t) = u_L \qquad \text{and} \qquad u(L,t) = u_R \tag{10.4}$$

where L represents the length of the bar and u_L and u_R are the left and right endpoint temperatures, respectively. Finally, specifying some initial temperature distribution,

$$u(x,0) = f(x) \tag{10.5}$$

completes the mathematical description of a particular heat transfer problem.

Now, following the above guidelines, we can try to solve this problem by separating the desired solution, u(x,t), into two components -- the *transient solution, v(x,t)*, and the *steady state solution, w(x)*, or

$$u(x,t) = v(x,t) + w(x) \tag{10.6}$$

With this assumed solution, we have

$$u_t = v_t \qquad \text{and} \qquad u_{xx} = v_{xx} + w''$$

and substitution into eqn. (10.3) gives

$$v_t = \alpha v_{xx} + \alpha w'' + Q$$

However, since we desire a homogeneous PDE, we let

$$\alpha w'' + Q = 0 \qquad \text{or} \qquad w'' = -\frac{1}{\alpha} Q$$

and, with this condition, the remaining PDE for v(x,t) is homogeneous, or

$$v_t = \alpha v_{xx}$$

We also desire homogeneous boundary conditions for the PDE. Performing similar operations as above, we substitute eqn. (10.6) into the original statement for the BCs [eqn. (10.4)] giving

$$u(0,t) = v(0,t) + w(0) = u_L \qquad \text{and with } w(0) = u_L, \text{ we have } v(0,t) = 0$$

and

$$u(L,t) = v(L,t) + w(L) = u_R \quad \text{and with } w(L) = u_R, \text{ we have } v(L,t) = 0$$

as desired.

As a last step, eqn. (10.6) is also inserted into eqn. (10.5) which describes the initial temperature distribution, or

$$u(x,0) = v(x,0) + w(x) = f(x)$$

which gives the initial condition for the v(x,t) problem as

$$v(x,0) = f(x) - w(x)$$

Thus, the conversion process is now complete -- the original problem which included an inhomogeneous PDE with inhomogeneous BCs has been converted into two problems, and each of these is straightforward to solve.

The *transient solution, v(x,t)*, is completely defined by

$$v_t = \alpha v_{xx} \quad \text{with} \quad v(0,t) = 0, \quad v(L,t) = 0, \quad \text{and} \quad v(x,0) = f(x) - w(x) \tag{10.7}$$

This problem includes a homogeneous PDE with homogeneous boundary conditions. Therefore, we can use the classical Separation of Variables method to determine v(x,t).

The *steady state problem* is completely defined by

$$w'' = -\frac{1}{\alpha}Q \quad \text{with} \quad w(0) = u_L \quad \text{and} \quad w(L) = u_R \tag{10.8}$$

which is a relatively simple ODE that can be solved by standard methods.

Finally, the solutions to the two separate problems given by eqns. (10.7) and (10.8) are substituted into eqn. (10.6) to give the desired space-time temperature distribution, u(x,t); thus completing the original problem of interest.

The basic technique outlined here is often quite useful. Example 10.2 illustrates this method for a particular 1-D transient heat transfer problem (a laterally insulated bar with fixed endpoint temperatures), including solution of the two sub-problems with implementation and plotting of the solutions within Matlab. It serves as a good illustration of the basic approach to solving problems of this type. Note that, since a preconditioning step to account for the inhomogeneous source term and/or the inhomogeneous BCs is often required for most realistic problems, this example includes all the steps needed in most situations where the SOV method can be used.

The Eigenfunction Expansion Method

Unfortunately, one cannot always remove the inhomogeneous terms from the PDE or BCs as illustrated in the above discussion. When this occurs, we often have difficulty finding an exact analytical solution to the given problem. However, if we combine the basic ideas of the SOV method and our knowledge and understanding of Generalized Fourier Series, we can always generate approximate solutions -- which are sometimes quite good depending on our ability to find a set of suitable expansion functions. The dilemma, of course, is that, without a Sturm-Liouville type sub-problem (which requires a homogeneous ODE and homogeneous BCs), we

can not generate an orthogonal set of eigenfunctions that are fully consistent with the desired problem specifications. One approach to this problem is to *select a reasonable set of functions* for the infinite series expansion and then proceed systematically to determine the unknown coefficients within the series, hoping to get a *reasonable approximation* to the real solution. This general technique is often referred to as the *Eigenfunction Expansion Method*.

As before, the best way to explain the method is via example. Let's again choose a parabolic PDE, the Diffusion Equation, for our example, but this time the BCs will be time dependent. This problem can be described mathematically as follows:

$$u_t = \alpha u_{xx} \quad \text{with} \quad u(0,t) = g_1(t) \quad u(L,t) = g_2(t) \quad \text{and} \quad u(x,0) = f(x) \tag{10.9}$$

where $g_1(t)$ and $g_2(t)$ are known time-dependent boundary values that are imposed on the left side and right side of the interval of interest.

Note that the technique used in the previous subsection will not work for this problem. However, based on experience with the Separation of Variables (SOV) method, let's assume that the solution to eqn. (10.9) can be written as

$$u(x,t) = \sum_n a_n(t)\phi_n(x) \tag{10.10}$$

where the $\phi_n(x)$ functions are a set of **known** functions defined over the domain of interest and the $a_n(t)$ functions are **unknown** coefficients to be determined as part of the solution process. Equation (10.10) is an assumed solution, which may or may not lead to an exact result for a given problem. In general, if the expansion functions do not exactly satisfy the BCs for the problem, no analytical solution is possible -- this is the situation in this problem, since the imposed BCs are time dependent. However, as we have seen in Sections VIII and IX, we can still use a Generalized Fourier Series to approximate the given function and, away from the boundaries, the series approximation usually gives a very good result.

Now, proceeding systematically from the assumed solution given in eqn. (10.10), we shall attempt to develop an expression for the unknown coefficients, $a_n(t)$. To do this, multiply eqn. (10.10) by $\phi_m(x)$ and integrate over the spatial domain to give

$$\int_0^L u(x,t)\phi_m(x)dx = \sum_n a_n(t)\int_0^L \phi_m(x)\phi_n(x)dx \tag{10.11}$$

This equation is rather complex for the general case for arbitrary $\phi_n(x)$. However, if the spatial expansion functions, $\phi_n(x)$, are solutions to an *"associated"* homogeneous ODE with homogeneous boundary conditions, then an explicit equation for $a_n(t)$ results. The term *associated* implies that the best choice for the basis functions should be the solution set from a Sturm-Liouville problem that closely resembles the problem being addressed. This will give a set of functions that are orthogonal over the domain defined by the original problem, and significantly reduce the complexity associated with eqn. (10.11).

To see this, let's choose $\phi_n(x)$ be a solution to

$$\phi''(x) + \lambda^2\phi(x) = 0 \quad \text{with} \quad \phi(0) = 0 \quad \text{and} \quad \phi(L) = 0$$

The general solution to this ODE is

$$\phi(x) = A_1 \sin \lambda x + A_2 \cos \lambda x$$

Applying the BC at $x = 0$ gives

$$\phi(0) = 0 = A_1(0) + A_2 \qquad \text{or} \qquad A_2 = 0$$

Similarly at $x = L$, we have

$$\phi(L) = 0 = A_1 \sin \lambda L$$

which leads to the eigencondition

$$\sin \lambda L = \sin n\pi \qquad \text{for} \quad n = 1, 2, \cdots$$

Thus, the eigenvalues and eigenfunctions for this "associated" problem are

$$\lambda_n = \frac{n\pi}{L} \qquad \text{and} \qquad \phi_n(x) = \sin \lambda_n x \qquad \text{for} \quad n = 1, 2, \cdots \tag{10.12}$$

Since these functions are solutions to a Sturm-Liouville problem, we have

$$\int_0^L \phi_m(x)\phi_n(x)dx = \frac{L}{2}\delta_{mn}$$

as the formal statement of orthogonality for these functions.

Substituting this orthogonality relation into eqn. (10.11) gives

$$a_n(t) = \frac{2}{L}\int_0^L u(x,t)\sin \lambda_n x \, dx \tag{10.13}$$

which is a considerable simplification of the original expression -- this is a direct result of the use of orthogonal functions. However, we still have some work to do, since eqn. (10.13) has $a_n(t)$ written in terms of the solution $u(x,t)$. We can resolve this situation, by differentiating eqn. (10.13) with respect to time to give

$$\frac{d}{dt}a_n(t) = \frac{2}{L}\int_0^L u_t(x,t)\sin \lambda_n x \, dx$$

But the original PDE allows us to replace $u_t(x,t)$ with $\alpha u_{xx}(x,t)$, or

$$\frac{d}{dt}a_n(t) = \frac{2\alpha}{L}\int_0^L u_{xx}(x,t)\sin \lambda_n x \, dx$$

Now integrating the RHS of this expression by parts, using the expressions

$$\int w \, dz = wz - \int z \, dw \qquad \text{with} \qquad w = \sin \lambda_n x \qquad dz = \frac{\partial}{\partial x}\{u_x(x,t)\}dx$$

$$dw = \lambda_n \cos \lambda_n x \, dx \qquad z = u_x(x,t)$$

gives

$$\frac{d}{dt}a_n(t) = \frac{2\alpha}{L}\left\{u_x(x,t)\sin\lambda_n x\Big|_0^L - \lambda_n\int_0^L u_x(x,t)\cos\lambda_n x\,dx\right\}$$

or

$$\frac{d}{dt}a_n(t) = -\frac{2\alpha\lambda_n}{L}\int_0^L u_x(x,t)\cos\lambda_n x\,dx$$

Integrating by parts one more time with

$$\int w\,dz = wz - \int z\,dw \qquad \text{and} \qquad w = \cos\lambda_n x \qquad dz = \frac{\partial}{\partial x}\{u(x,t)\}dx$$

$$dw = -\lambda_n\sin\lambda_n x\,dx \qquad z = u(x,t)$$

gives

$$\frac{d}{dt}a_n(t) = -\frac{2\alpha\lambda_n}{L}\left\{u(x,t)\cos\lambda_n x\Big|_0^L + \lambda_n\int_0^L u(x,t)\sin\lambda_n x\,dx\right\}$$

$$= -\frac{2\alpha\lambda_n}{L}\left\{u(L,t)\cos n\pi - u(0,t) + \lambda_n a_n(t)\frac{L}{2}\right\}$$

$$= -\frac{2\alpha\lambda_n}{L}\left\{g_2(t)\cos n\pi - g_1(t) + \lambda_n a_n(t)\frac{L}{2}\right\}$$

where, in the last equality, we have used the imposed boundary conditions to replace u(0,t) and u(L,t). With one final algebraic simplification, one has

$$\frac{d}{dt}a_n(t) = -\frac{2\alpha\lambda_n}{L}\left[(-1)^n g_2(t) - g_1(t)\right] - \alpha\lambda_n^2 a_n(t)$$

or

$$\frac{d}{dt}a_n(t) + \alpha\lambda_n^2 a_n(t) = \beta_n g_n(t) \tag{10.14}$$

with β_n and $g_n(t)$ defined by

$$\beta_n = -\frac{2\alpha\lambda_n}{L} \qquad \text{and} \qquad g_n(t) = (-1)^n g_2(t) - g_1(t) \tag{10.15}$$

Finally, note that the initial condition, u(x,0), is needed to specify the initial condition for $a_n(0)$. Inserting $t = 0$ into eqn. (10.13) gives the desired relationship,

$$a_n(0) = \frac{2}{L}\int_0^L u(x,0)\sin\lambda_n x\,dx = \frac{2}{L}\int_0^L f(x)\sin\lambda_n x\,dx \tag{10.16}$$

Equations (10.14)-(10.16) completely define a set of first order linear ODEs for the $a_n(t)$ coefficients. Given $g_1(t)$ and $g_2(t)$ for a particular problem, this system can be solved for the specific time-dependent coefficients of interest. These coefficients, coupled with the spatial functions defined in eqn. (10.12), give the final solution to this problem via eqn. (10.10).

This solution scheme is certainly more complicated than the classical Separation of Variables method, and it usually only gives ***approximate solutions***, depending upon how well the BCs are satisfied. However, if the original problem cannot be treated with the classical method, then one can at least get a reasonable idea of the general result for the problem.

Example 10.3 illustrates this method. It treats a 1-D transient heat conduction problem with a time-dependent right boundary condition. The solution is evaluated in Matlab and the results are plotted and discussed. It is clear from the plotted solutions that the right boundary condition is not satisfied during the early transient; thus, the space-time temperature distribution throughout the whole domain can only be treated as an approximation to reality.

Note: This same problem is also solved again in Example 11.4 in the next section using a numerical technique. Comparison of the two solutions shows that the analytical method is indeed approximate, but overall it does quite well in establishing the general space-time temperature profile for this system (near the left end of the bar the predictions are very good).

Although the Eigenfunction Expansion Method is useful in a variety of situations, it is quite tedious and it only gives approximate solutions to the problem of interest. In practical engineering analysis applications, when a problem can not be solved using exact analytical means, a numerical solution scheme is used. Thus, the current method often serves as a bridge between a pure analytical technique and a numerical approach to a particular problem. When the analytical methods become this tedious and/or only give approximate solutions, then it is clearly time to look to the computer for assistance -- and this is exactly the approach taken in Section XI on the Numerical Solution of PDEs.

The SOV and general Eigenfunction Expansion methods are quite powerful for the analytical solution of many classical PDE problems, and a series of detailed examples are available in the next subsection to give the reader some experience with the details of the methods outlined here. The first three problems illustrate, respectively, the three subjects discussed here -- the classical SOV method, the treatment of inhomogeneous source terms and inhomogeneous BCs, and the general Eigenfunction Expansion method. These three problems should be studied in detail to get a good overview of the basic solution techniques. Beyond the basics, however, one is often faced with a number of subtle differences from the classical examples presented here and the problems you are asked to solve in practical applications. Thus, a set of additional worked-out examples are also provided to give the reader further experience beyond the basics -- such as working with cylindrical coordinates which often lead to solutions in the form of a generalized Fourier-Bessel series, the treatment of elliptic PDEs (relative to the parabolic PDEs addressed in the first three examples), etc. etc.. Each example has something new to offer, and you are encouraged to study each of these in some detail -- since, as stated before, the best way to understand various PDE solution methods is via example!!!

Example 10.1 -- Heat Transfer in a Finite Bar with Homogeneous BCs

Problem Description:

A laterally insulated bar of length L initially has some temperature distribution given by f(x). The ends at x = 0 and x = L are held fixed at 0 ^{0}C for all time. Determine the temperature profile versus time, u(x,t), given that

$$f(x) = \begin{cases} x & \text{if} & 0 < x < L/2 \\ L - x & \text{if} & L/2 < x < L \end{cases}$$

Use $\alpha = 1$ cm^2/s and L = 50 cm for numerical evaluation and plotting of the solution within Matlab.

Problem Solution:

Theoretical Development

This problem calls for a classic Separation of Variables solution scheme since the defining heat conduction equation (i.e. the Diffusion Equation) and the boundary conditions are both homogeneous. Formally, this problem can be stated mathematically as

$$u_t(x,t) = \alpha u_{xx}(x,t) \qquad \text{with} \quad u(0,t) = 0, \quad u(L,t) = 0, \quad \text{and} \quad u(x,0) = f(x)$$

Following the outline discussed previously, let's first assume a solution of the form

$$u(x,t) = F(x)G(t)$$

Substitution of this expression into the PDE gives

$$F\dot{G} = \alpha F''G$$

where the "dot" notation implies a time derivative and the "primes" indicate spatial derivatives.

This equation is separable, or

$$\frac{F''}{F} = \frac{\dot{G}}{\alpha G} = k$$

where k is referred to as a *separation constant*. The implication here is that, since the LHS is only a function of x and the RHS is only a function of t, for these terms to be equal to each other for all x and t, they must both be constants -- which is indicated by the separation constant, k. This observation is very important because it effectively breaks the original PDE into two separate ODEs; one for the G(t) function and another for F(x). The two equations are

$$\dot{G} - \alpha kG = 0 \qquad \text{and} \qquad F'' - kF = 0$$

where the specification of the F-problem becomes complete with the conversion of the original BCs to the new notation. Formally, we have

$$u(0,t) = 0 \quad \Rightarrow \quad G(t)F(0) = 0 \qquad \text{or} \qquad F(0) = 0$$

$$u(L,t) = 0 \quad \Rightarrow \quad G(t)F(L) = 0 \qquad \text{or} \qquad F(L) = 0$$

Thus, the G-problem is a simple first order ODE with solution

$$G(t) = Ce^{\alpha k t}$$

and the F-problem is a Sturm-Liouville problem (homogeneous ODE with homogeneous BCs). From our experience with eigenvalue problems of this type, we recognize that $k \geq 0$ leads to trivial solutions. Thus, we choose k to be negative, or $k = -\lambda^2$, which gives

$$F'' + \lambda^2 F = 0 \qquad \text{with solution} \qquad F(x) = C_1 \sin \lambda x + C_2 \cos \lambda x$$

Now let's apply the BCs to the general solution for the F-problem, or

$$\text{at } x = 0, \quad F(0) = 0 = C_1(0) + C_2(1) \qquad \text{or} \qquad C_2 = 0$$

$$\text{at } x = L, \quad F(L) = 0 = C_1 \sin \lambda L \qquad \text{or} \qquad \sin \lambda L = \sin n\pi = 0 \quad \text{for} \quad n = 1, 2, \cdots$$

where this last condition is the ***eigencondition*** for this problem. Thus, the ***eigenvalues*** and ***eigenfunction solutions*** are given by

$$\lambda_n = \frac{n\pi}{L} \qquad \text{and} \qquad F_n(x) = \sin \frac{n\pi x}{L} \qquad \text{for} \quad n = 1, 2, \cdots$$

(note that this solution was already obtained as part of Example 9.1)

Now, since we have multiple values of λ_n that satisfy the spatial problem, we also have multiple solutions for the G-problem. Rewriting the above G(t) solution with $k_n = -\lambda_n^2$ gives

$$G_n(t) = C_n e^{-\alpha \lambda_n^2 t}$$

With known distribution functions for the spatial and temporal profiles, the desired temperature distribution can be written as

$$u(x, t) = \sum_n u_n(x, t) = \sum_n F_n(x) G_n(t) = \sum_n C_n \sin \lambda_n x \, e^{-\alpha \lambda_n^2 t}$$

where we have used the fact that linear equations have general solutions that are formed from a linear combination of the individual solutions.

The only remaining unknown left to be determined is the coefficient, C_n. However, we still have an initial condition for this problem that has not yet been used. In particular, letting $t = 0$ in the final expression for u(x,t) gives

$$u(x, 0) = f(x) = \sum_n C_n \sin \frac{n\pi x}{L}$$

However, this is just a Fourier series expansion for f(x) in terms of a set of orthogonal basis functions. The unknown expansion coefficients, C_n, can be determined, as usual, by using the orthogonality property of the expansion functions. In particular, if we multiply both sides of the last expression by $\sin \lambda_m x$ and integrate over $0 < x < L$, we have

$$\int_0^L f(x) \sin \frac{m\pi x}{L} dx = \sum_n C_n \int_0^L \sin \frac{n\pi x}{L} \sin \frac{m\pi x}{L} dx$$

and since the norm for these functions is L/2, we get

$$C_m = \frac{2}{L} \int_0^L f(x) \sin \frac{m\pi x}{L} dx$$

Now we can evaluate the desired C_n's for the specific f(x) given in the problem specification, or

$$C_n = \frac{2}{L} \left[\int_0^{\frac{L}{2}} x \sin \frac{n\pi x}{L} dx + \int_{\frac{L}{2}}^L (L-x) \sin \frac{n\pi x}{L} dx \right]$$

$$= \frac{2}{L} \left[\left(\frac{L}{n\pi} \right)^2 \sin \frac{n\pi x}{L} - \frac{L}{n\pi} x \cos \frac{n\pi x}{L} \right]_0^{\frac{L}{2}}$$

$$+ \frac{2}{L} \left[L \left(\frac{L}{n\pi} \right) \left(-\cos \frac{n\pi x}{L} \right) - \left(\frac{L}{n\pi} \right)^2 \sin \frac{n\pi x}{L} + \frac{L}{n\pi} x \cos \frac{n\pi x}{L} \right]_{\frac{L}{2}}^L$$

or

$$C_n = \frac{2}{L} \left[\left(\frac{L}{n\pi} \right)^2 \sin \frac{n\pi}{2} + 0 - 0 + 0 \right]$$

$$+ \frac{2}{L} \left[-\frac{L^2}{n\pi} \cos n\pi - 0 + \frac{L^2}{n\pi} \cos n\pi + 0 + \left(\frac{L}{n\pi} \right)^2 \sin \frac{n\pi}{2} - 0 \right]$$

Thus, the coefficients simplify to

$$C_n = \frac{4L}{(n\pi)^2} \sin \frac{n\pi}{2}$$

Finally we note that $\sin n\pi/2$ is zero for even n. Thus, we can consider only the nonzero coefficients by letting $n = 2m - 1$ for $m = 1, 2, \cdots$.

We have finally completed this development using the classical Separation of Variables method. In summary, the pertinent equations for the space-time temperature profiles for this example are

$$u(x,t) = \sum_m C_m \sin \lambda_m x e^{-\alpha \lambda_m^2 t}$$

where

$$C_m = \frac{4L}{(2m-1)^2 \pi^2} \sin \frac{(2m-1)\pi}{2} \qquad \text{and} \qquad \lambda_m = \frac{(2m-1)\pi}{L} \qquad \text{for } m = 1, 2, \cdots$$

Matlab Implementation

These final expressions have been implemented and evaluated in Matlab file **ht1d_sov1.m**, and the resultant temperature profiles for various times have been plotted. The Matlab file is listed in Table 10.1. The expansion coefficients and eigenvalues are first evaluated for some maximum number of nonzero terms and stored for later use. Then, an outer loop over Nt discrete time points is initiated and, within this loop, a ***while loop*** over the nonzero components of the series expansion is made as long as the next term continues to contribute to the partial sum. The spatial variable is simply treated using the vector arithmetic capability built into the Matlab syntax. Finally, the discrete time profiles are plotted as separate curves in a single Matlab figure.

The final solution profiles for this problem are given in Fig. 10.1. At t = 0, the temperature profile has a triangular shape as specified in the problem description. Since there is no energy generation within the bar and the endpoint temperatures are fixed at zero, we expect that, over time, the energy within the bar will move along the bar toward the ends. Eventually, all the energy initially stored in the bar will dissipate, leaving the bar at a constant temperature of zero degrees. As seen in Fig. 10.1, this is exactly what is observed from the simulations. The profiles are always symmetric, as expected, and the temperatures gradually do approach the endpoint temperatures.

This example represents a good illustration of the Separation of Variables method and the Matlab file **ht1d_sov1.m** shows, in general, how to numerically evaluate infinite series expansions for visualization and analysis of the analytical solutions. This same scheme can also be applied to other problems of this type.

Finally, we note that the reader should also see Examples 10.4 and 10.5 for additional demonstrations of the classical SOV method. Example 10.4, in particular, is very similar to the problem solved here, except that the heat transfer flows radially outward from the center of a cylindrical bar. Because of the cylindrical geometry, the expansion functions for Example 10.4 turn out to be Bessel functions. Thus, in addition to providing further insight into the SOV method, Example 10.4 also gives some additional experience with the use of Bessel functions for solving some real problems of interest.

Example 10.5, in contrast, solves a steady state heat transfer problem in 2-D geometry. Although this problem also uses the classical SOV method, the base mathematical problem is an elliptic PDE (rather than a parabolic PDE as for the current example), and the solution scheme is somewhat different. Thus, Example 10.5 is a good companion problem to the current example.

Table 10.1 Listing of Matlab file ht1d_sov1.m.

```
%
%   HT1D_SOV1.M   Heat Transfer in a 1-D Finite Bar (Example 10.1 in Class Notes)
%
%   Analytical Solutionn using Separation of Variables (SOV) method for the
%   following problem:
%     ut(x,t) = alf*uxx(x,t)   with u(0,t) = 0   u(L,t) = 0   u(x,0) = f(x)
%     where    u(x,0) = f(x) = x    if 0 < x <L/2
%                              L-x  if L/2 < x < L
%
%   The goal here is to evaluate and plot the temperature profile in the
%   1-D finite bar of length L at various time points.  This is an easy
%   way to visualize the full space-time solution, u(x,t).
%
```

```
%    File prepared by J. R. White, UMass-Lowell (Aug. 2003)
%

%
%    getting started
      clear all,   close all,   nfig = 0;
%
%    problem data
      alf = 1.0;                % cm^2/s thermal diffusivity
      L = 50;                   % cm     length of bar
      Nx = 101;                 % number of x values
      x = linspace(0,L,Nx)';    % vector of points to evaluate function
      nmax = 50;                % max number of nonzero terms
      tol = 0.001;              % tolerance to stop series evaluation
%
%    calc ln = n*pi/L  and   bn = (4*L/n*pi)*sin(n*pi/2)  for n = 1,3,5,...
      ln = zeros(1,nmax);    bn =  zeros(1,nmax);
      dd = pi/L;    cc = 4*L/(pi^2);
      for m = 1:nmax
        n = 2*m-1;      ln(m) = n*dd;    cn(m) = cc*sin(n*pi/2)/(n^2);
      end
%
%    now evaluate series expansion for several different times
      tt = [0 25 100 250 500];   Nt = length(tt);    ut = zeros(Nx,Nt);
%
      for i = 1:Nt
        t = tt(i);    cc = -alf*t;   mrerr = 1.0;   n = 0;   u = zeros(size(x));
        while mrerr > tol   &    n < nmax
          n = n+1;    un = cn(n)*exp(cc*ln(n)*ln(n))*sin(ln(n)*x);   u = u + un;
          j = find(u);                   % finds indices of nonzero values of u(x)
          mrerr = max(abs(un(j)./u(j)));  % compute max relative error
        end
        ut(:,i) = u;
        disp([' Needed ',num2str(n),' terms for convergence at t = ',num2str(t),' s'])
      end
%
%    plot curves of u(x,t) for various times
      nfig = nfig+1;  figure(nfig)
      v = [0 L 0 25];
      plot(x,ut,'LineWidth',2),axis(v)
      title('HT1D\_SOV1:  Bar Temperature Profile at Various Times (Example 10.1)')
      grid,xlabel('Distance (cm)'),ylabel('Temperature (C)')
      for i = 1:Nt,  gtext(['t = ',num2str(tt(i)),' s']),   end
%
%    end of example%
```
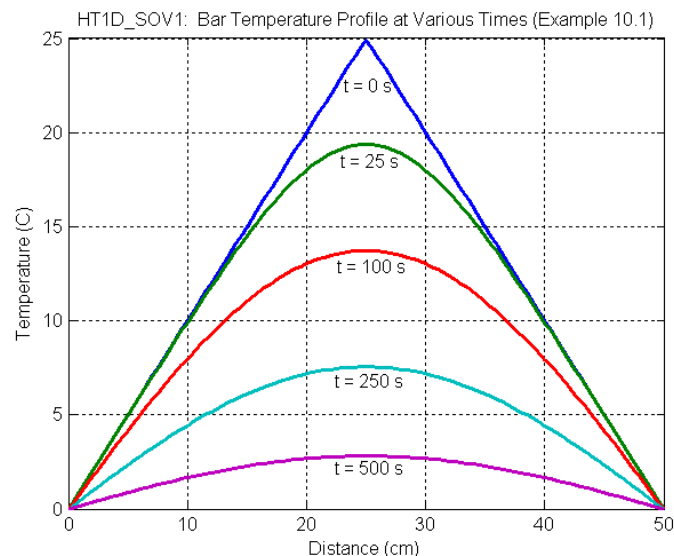

Fig. 10.1 Temperature profiles from Example 10.1.

Example 10.2 – Heat Transfer in a Finite Bar with Fixed End Temperatures

Problem Description:

A laterally insulated bar of length L initially has some temperature distribution given by f(x). At time zero, the left and right sides are subjected to fixed temperatures, u_L and u_R, respectively, and these are held constant for all time. Determine the temperature profile versus time, u(x,t), given the following specific conditions:

$$u_L = 100 \; ^\circ C \qquad u_R = 400 \; ^\circ C \qquad f(x) = 70 \; ^\circ C \text{ (i.e. initial profile is constant)}$$

Also use $\alpha = k/\rho c_p = 0.0001 \, m^2/s$ and L = 1 m for numerical evaluation and plotting of the solution within Matlab.

Problem Solution:

Theoretical Development

Formally, this problem can be stated mathematically as

$$u_t(x,t) = \alpha u_{xx}(x,t) \qquad \text{with} \quad u(0,t) = u_L, \quad u(L,t) = u_R, \quad \text{and} \quad u(x,0) = f(x) = u_0$$

Since the BCs are ***not*** homogeneous, the standard Separation of Variables solution scheme will not work. However, as discussed previously, let's first assume a solution of the form

$$u(x,t) = v(x,t) + w(x)$$

Upon substitution, the original PDE is converted into two problems: a homogeneous PDE with homogeneous BCs that describes the ***transient solution***, and an ODE that represents the final ***steady state solution*** for this situation. The ***transient problem*** is described by

$$v_t = \alpha v_{xx} \qquad \text{with} \quad v(0,t) = 0, \quad v(L,t) = 0, \quad \text{and} \quad v(x,0) = f(x) - w(x)$$

and the ***steady state problem*** is completely defined by

$$w'' = 0 \qquad \text{with} \quad w(0) = u_L \quad \text{and} \quad w(L) = u_R$$

Both these problems are straightforward to solve. Working first on the steady state solution, we have after two integrations

$$w(x) = C_1 x + C_2$$

and applying the boundary conditions gives

$$\text{at } x = 0, \quad w(0) = u_L = C_1(0) + C_2 \qquad \text{or} \qquad C_2 = u_L$$

$$\text{at } x = L, \quad w(L) = u_R = C_1 L + u_L \qquad \text{or} \qquad C_1 = \frac{u_R - u_L}{L}$$

Thus, the steady state solution is simply a linear profile given by

$$w(x) = \left(\frac{u_R - u_L}{L} \right) x + u_L$$

Now turning to the transient part, we can use a classical Separation of Variables approach for this homogeneous PDE with homogeneous BCs. Following the same procedure as used in Example 10.1, let's first assume a solution of the form

$$v(x,t) = F(x)G(t)$$

Substitution of this expression into the PDE gives

$$F\dot{G} = \alpha F''G$$

and since this is separable, we have

$$\frac{F''}{F} = \frac{\dot{G}}{\alpha G} = k$$

where k is the ***separation constant***. The form of this expression implies that each term must be a constant for this equality to be true for all x and t. This observation is essential because it effectively breaks the original PDE into two separate ODEs; one for the G(t) function and another for F(x). The two equations are

$$\dot{G} - \alpha k G = 0 \qquad \text{and} \qquad F'' - kF = 0 \quad \text{with} \quad F(0) = 0 \quad F(L) = 0$$

Thus, the G-problem is a simple first order ODE with solution $G(t) = Ce^{\alpha kt}$, and the F-problem is a Sturm-Liouville problem (homogeneous ODE with homogeneous BCs). Based on prior experience with eigenvalue problems of this type, we recognize that $k \geq 0$ leads to trivial solutions. Thus, we choose k to be negative, or $k = -\lambda^2$, which gives

$$F'' + \lambda^2 F = 0 \qquad \text{with solution} \qquad F(x) = C_1 \sin\lambda x + C_2 \cos\lambda x$$

Now applying the BCs to the general solution for the F-problem gives

$$\text{at } x = 0, \quad F(0) = 0 = C_1(0) + C_2(1) \qquad \text{or} \qquad C_2 = 0$$

$$\text{at } x = L, \quad F(L) = 0 = C_1 \sin\lambda L \qquad \text{or} \qquad \sin\lambda L = \sin n\pi = 0 \quad \text{for} \quad n = 1, 2, \cdots$$

where the last condition is the ***eigencondition*** for this problem. Thus, the ***eigenvalues*** and ***eigenfunction solutions*** are given by

$$\lambda_n = \frac{n\pi}{L} \qquad \text{and} \qquad F_n(x) = \sin\frac{n\pi x}{L} \qquad \text{for} \quad n = 1, 2, \cdots$$

The multiple values of λ_n that satisfy the spatial problem lead to multiple solutions for the G-problem. Rewriting the above G(t) solution with $k_n = -\lambda_n^2$ gives

$$G_n(t) = C_n e^{-\alpha \lambda_n^2 t}$$

Now, we write the desired solution to the transient problem in terms of a linear combination of the individual solutions, or

$$v(x,t) = \sum_n v_n(x,t) = \sum_n F_n(x)G_n(t) = \sum_n C_n \sin\lambda_n x\, e^{-\alpha \lambda_n^2 t}$$

As before, the initial condition for this problem can be used to determine the remaining unknown coefficients, C_n. In particular, letting $t = 0$ in the expression for $v(x,t)$ gives

$$v(x,0) = f(x) - w(x) = g(x) = \sum_n C_n \sin \frac{n\pi x}{L}$$

The unknown expansion coefficients, C_n, in this Fourier series expansion for $g(x)$ can be determined by using the orthogonality property of the expansion functions. In particular, if we multiply both sides of the last expression by $\sin \lambda_m x$ and integrate over $0 < x < L$, we have

$$\int_0^L g(x) \sin \frac{m\pi x}{L} dx = \sum_n C_n \int_0^L \sin \frac{n\pi x}{L} \sin \frac{m\pi x}{L} dx$$

and since the norm for these functions is L/2, we get

$$C_m = \frac{2}{L} \int_0^L g(x) \sin \frac{m\pi x}{L} dx$$

Now all we are left with is the application of specific values for this problem. With $L = 1$ m and fixed endpoint temperatures, $u_L = 100\ ^\circ C$ and $u_R = 400\ ^\circ C$, the steady state solution becomes

$$w(x) = \left(\frac{u_R - u_L}{L} \right) x + u_L = 300x + 100$$

Also, with an initially constant profile, $f(x) = u_0 = 70\ ^0C$, the initial condition for $v(x,t)$ becomes $v(x,0) = g(x) = f(x) - w(x)$, or

$$g(x) = u_0 - \left(\frac{u_r - u_L}{L} \right) x - u_L = 70 - 300x - 100 = -30(10x + 1)$$

With this result, we can calculate the expansion coefficients as

$$C_n = \frac{2}{L} \int_0^L g(x) \sin \frac{n\pi x}{L} dx = -60 \int_0^1 (10x + 1) \sin n\pi x dx$$

$$= -60 \left\{ 10 \left[\frac{1}{n^2\pi^2} \sin n\pi x - \frac{1}{n\pi} x \cos n\pi x \right]_0^1 + \left[\frac{-1}{n\pi} \cos n\pi x \right]_0^1 \right\}$$

$$= \frac{600}{n\pi} \cos n\pi + \frac{60}{n\pi} (\cos n\pi - 1)$$

or

$$C_n = \frac{600}{n\pi} (-1)^n + \frac{60}{n\pi} \left[(-1)^n - 1 \right]$$

Finally, with the C_n's known, we can form the complete solutions as $u(x,t) = v(x,t) + w(x)$, or

$$u(x,t) = \sum_n C_n \sin(\lambda_n x) e^{-\alpha\lambda_n^2 t} + \left(\frac{u_R - u_L}{L} \right) x + u_L$$

with all the constants completely specified.

Matlab Implementation

The solution for this problem has been implemented and evaluated in Matlab file **ht1d_sov2.m**, and the resultant temperature profiles for various times have been plotted in Fig. 10.2. The Matlab file is listed in Table 10.2. The coding algorithm used here is similar to that discussed for Example 10.1 (see listing of **ht1d_sov1.m** in Table 10.1) except, for the current problem, the steady state solution has been added to the solution of the transient problem.

As shown in Fig. 10.2, at t = 0 the constant temperature distribution is represented only roughly by the Fourier series expansion. This was expected, since the endpoints associated with the expansion functions do not match the function being represented. Thus, there is no way this function can be reproduced exactly with the basis functions for this problem -- even with an infinite number of terms (recall that this situation was observed previously in Example 9.1). Normally, since the initial temperature profile is known, the exact f(x) distribution is plotted. Here we chose to evaluate and plot the series representation of u(x,t) at t = 0 just to re-emphasize that the series expansion does not always converge to the exact profile. The time evolution of the temperature profiles, however, follows expected behavior, eventually approaching the steady state linear temperature distribution at long times (t = 5000 s in Fig. 10.2). The series representation is exact for t > 0, since the eigenfunctions match the actual BCs for all t > 0.

Note that an option was also implemented within in the Matlab file to allow the user to create a movie that represents the time evolution of the temperature profiles via animation. The student is invited to actually run **ht1d_sov2.m** to observe this animation. It simply represents another way to help visualize the physical behavior of this system.

The primary purpose of this example was to illustrate how to treat problems with inhomogeneous components. This was achieved by breaking the problem into two parts, and by solving each of these using standard methods. The Matlab file for this problem, **ht1d_sov2.m**, simply represents another example of the evaluation and plotting of Fourier series solutions for a particular PDE. It also introduces the use of animation as a tool that might aid in the visualization and understanding of the physical processes occurring in the system of interest.
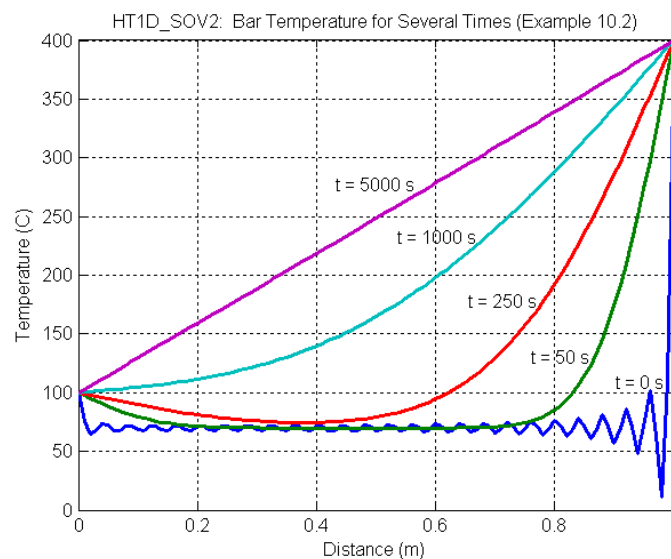

Fig. 10.2 Temperature profiles from Example 10.2.

Table 10.2 Listing of Matlab file ht1d_sov2.m.

```
%
%   HT1D_SOV2.M   Heat Transfer in a 1-D Finite Bar  (Example 10.2 in Class Notes)
%
%   Analytical Solution using Separation of Variables (SOV) to the following problem:
%     ut(x,t) = alf*uxx(x,t)   with  u(0,t) = ul    u(L,t) = ur    u(x,0) = u0
%   has solution given by
%     u(x,t) = v(x,t) + w(x)
%   where v(x,t) is the transient solution and w(x) is the steady state solution
%
%   The goal here is to evaluate and visualize the temperature profile in the
%   1-D finite bar of length L.  Two options are provided:
%     1. Create snapshots of u(x) for several times.  This is an easy
%        way to visualize the full space-time solution, u(x,t).
%     2. Use Matlab's animation capability to create a movie that
%        shows the spatial temperature profile changing in time.
%
%   File prepared by J. R. White, UMass-Lowell  (Aug. 2003)
%

%
%   getting started
      clear all,   close all,   nfig = 0;
%
%   problem data
      alf = 0.0001;    % m^2/s  thermal diffusivity
      ul = 100;        % C      fixed temp at left end
      ur = 400;        % C      fixed temp at right end
      u0 = 70;         % C      initial uniform temp of bar
      L = 1;           % m      length of bar
      maxt = 50;       % max number of terms in expansion
      tol = 0.001;     % tolerance used to stop series expansion
%
%   the steady state solution w(x)
      Nx = 101;   x = linspace(0,L,Nx)';   w = (ur-ul)*x/L + ul;
%
%   now v(x,t) is given as an infinite series
%   calc terms for n = 1,2,3,4,5,...,max
      cc1 = 600*L/pi;   cc2 = 60/pi;
      for n = 1:maxt
        lam(n) = n*pi/L;   c(n) = (1/n)*(cc1*(-1)^n + cc2*((-1)^n-1));
      end
%
%   define time points and initialize space-time distribution
      opt = menu('Output plot option?','Create profiles at 5 time points', ...
                 'Make a movie with Nt frames');
      switch opt
        case 1     % use for plotting snapshots
          Nt = 5;    tt = [0 50 250 1000 5000];
        case 2     % use for making smooth movie
          Nt = input(' Input number of frames for movie (5-100): ');
          if Nt < 5,    Nt = 5;    end
          if Nt > 100,  Nt = 100;  end
          tt = linspace(0,5000,Nt);
      end
%
%   start loop over time points
      u = zeros(Nx,Nt);
      for i = 1:Nt
        t = tt(i);   cc = -alf*t;   mrerr = 1.0;   n = 0;   v = zeros(size(x));
          while mrerr > tol   &   n < maxt
            n = n+1; vn = c(n)*sin(lam(n)*x).*exp(cc*lam(n)*lam(n));  v = v + vn;
            j = find(v);                      % finds indices of nonzero values of v(x)
            mrerr = max(abs(vn(j)./v(j)));  % compute max relative error
          end
        u(:,i) = v + w;
        disp([' Needed ',num2str(n),' terms for convergence at t = ',num2str(t),' s'])
      end
%
%   plot curves of u for various times (if opt = 1)
      switch opt
        case 1
          nfig = nfig+1;   figure(nfig)
```

```
            plot(x,u,'LineWidth',2)
            axis([0 1 0 400]);
            title('HT1D\_SOV2:  Bar Temperature for Several Times (Example 10.2)')
            grid,xlabel('Distance (m)'),ylabel('Temperature (C)')
            for i = 1:Nt,   gtext(['t = ',num2str(tt(i)),' s']),   end
%
%   let's make a movie of the time-dependent profile (if opt = 2)
        case 2
            nfig = nfig+1;      figure(nfig)
            plot(x,u(:,1),'LineWidth',2);    axis([0 L 0 400]);
            M = moviein(Nt);
            for i = 1:Nt
              plot(x,u(:,i),'LineWidth',2);  axis([0 L 0 400]);
              M(:,i) = getframe;
            end
%
%   play the movie one more time and show initial and final profiles
            movie(M,0);   hold on;
            plot(x,u(:,1),'LineWidth',2);    grid on;
            title('HT1D\_SOV2:  Bar Temperature for Initial and Final Times (Example 10.2)');
            xlabel('Distance (m)'),ylabel('Temperature (C)');
            gtext('initial'),gtext('final')
            hold off
     end
%
%   end of problem
```

Example 10.3 -- Heat Transfer in a Finite Bar with Time Dependent BCs

Problem Description:

A laterally insulated bar of length L has some temperature distribution, f(x). At time $t = 0$, the right side is cooled in such a way that the temperature drops to zero in an exponential fashion. The left endpoint temperature is held fixed at u_L. Find the temperature distribution, u(x,t), for this system given the following specific conditions:

$$u_L = 100\ ^{\circ}C, \qquad u_R(t) = u_{R_0}e^{-\mu t} \qquad and \qquad f(x) = \left(\frac{u_{R_0} - u_L}{L}\right)x + u_L$$

where $u_{R_0} = 400\ ^{\circ}C$. Also let $\alpha = k/\rho c_p = 0.0001\ m^2/s$ and $L = 1$ m for numerical evaluation and plotting of the solution within Matlab, and determine the sensitivity of the result to the exponential decay rate by using two different values of μ (0.001 s^{-1} for slow decay and 0.005 s^{-1} for rapid decay).

Problem Solution:

Theoretical Development

This problem can be stated formally in mathematical terms as

$$u_t(x,t) = \alpha u_{xx}(x,t) \qquad with \quad u(0,t) = u_L \quad and \quad u(L,t) = u_{R_0}e^{-\mu t}$$

and

$$u(x,0) = f(x) = \left(\frac{u_{R_0} - u_L}{L}\right)x + u_L$$

Since the BCs are not homogeneous and one of the inhomogeneous terms is time dependent, the techniques used in Example 10.1 or Example 10.2 are not sufficient, by themselves, to solve this problem. Instead, we will employ the Eigenfunction Expansion Method to obtain an approximate solution for this situation.

To remove the constant boundary term on the left hand side of the bar, let's break the overall temperature distribution into the sum of a transient solution and a steady state solution, or

$$u(x,t) = v(x,t) + w(x)$$

Following the same steps as performed in Example 10.2, substitution of this expression into the defining PDE and BCs gives the ***transient problem*** as

$$v_t = \alpha v_{xx} \qquad with \quad v(0,t) = 0, \quad v(L,t) = u_{R_0}e^{-\mu t}, \quad and \quad v(x,0) = f(x) - w(x)$$

and the ***steady state problem*** as

$$w'' = 0 \qquad with \quad w(0) = u_L \quad and \quad w(L) = 0$$

Note that the transient problem still has inhomogeneous BCs; thus, the standard Separation of Variables (SOV) method is not suitable.

Focusing first on the steady state solution, we have after two integrations

$$w(x) = C_1 x + C_2$$

and applying the boundary conditions gives

at $x = 0$, $\quad w(0) = u_L = C_1(0) + C_2 \qquad$ or $\qquad C_2 = u_L$

at $x = L$, $\quad w(L) = 0 = C_1 L + u_L \qquad$ or $\qquad C_1 = -\dfrac{u_L}{L}$

Thus, the steady state solution is simply a linear profile given by

$$w(x) = \frac{u_L}{L}(L - x)$$

Now addressing the transient problem, we see we were not able to completely remove the inhomogeneous BCs. In fact, in this case, the right BC is time dependent. This is a specific example of a linear homogeneous PDE with an inhomogeneous BC, and the Eigenfunction Expansion Method (as discussed previously) should be appropriate for this problem.

Following the previous development, we let

$$v(x,t) = \sum_n a_n(t)\phi_n(x)$$

where the $\phi_n(x)$ are the orthogonal eigenfunctions of the "associated" Sturm-Liouville problem with **homogeneous** BCs which, for this case, is given by

$$\phi''(x) + \lambda^2 \phi(x) = 0 \qquad \text{with} \qquad \phi(0) = 0 \quad \text{and} \quad \phi(L) = 0$$

with solution

$$\phi_n(x) = \sin \lambda_n x \qquad \text{and} \qquad \lambda_n = \frac{n\pi}{L} \qquad \text{for} \quad n = 1, 2, \cdots$$

Therefore, the eigenfunction expansion for the transient solution becomes

$$v(x,t) = \sum_n a_n(t)\sin \lambda_n x \qquad \text{with} \qquad \lambda_n = \frac{n\pi}{L} \qquad \text{for} \quad n = 1, 2, \cdots$$

and the boundary and initial conditions are

$$v(0,t) = g_1(t) = 0, \qquad v(L,t) = g_2(t) = u_{R_0}e^{-\mu t}$$

and

$$v(x,0) = h(x) = f(x) - w(x) == \left(\frac{u_{R_0} - u_L}{L}\right)x + u_L - \left(\frac{u_L}{L}(L-x)\right) = \frac{u_{R_0}}{L}x$$

From our previous development, the $a_n(t)$ expansion coefficients satisfy

$$\frac{d}{dt}a_n(t) + \alpha \lambda_n^2 a_n(t) = \beta_n g_n(t)$$

with β_n and $g_n(t)$ defined by

$$\beta_n = -\frac{2\alpha\lambda_n}{L} \qquad \text{and} \qquad g_n(t) = (-1)^n g_2(t) - g_1(t) = (-1)^n u_{R_0} e^{-\mu t}$$

and the initial condition for $a_n(t)$ is given by

$$a_n(0) = a_{n_0} = \frac{2}{L}\int_0^L v(x,0)\sin\lambda_n x dx = \frac{2}{L}\int_0^L h(x)\sin\lambda_n x dx$$

This is a first order linear ODE with integrating factor $e^{\int \alpha\lambda_n^2 dt} = e^{\alpha\lambda_n^2 t}$. Therefore, we have

$$\frac{d}{dt}\left\{e^{\alpha\lambda_n^2 t}a_n(t)\right\} = e^{\alpha\lambda_n^2 t}\frac{d}{dt}a_n(t) + \alpha\lambda_n^2 e^{\alpha\lambda_n^2 t}a_n(t) = C_n\left[e^{(\alpha\lambda_n^2-\mu)t}\right]$$

where

$$C_n = \beta_n(-1)^n u_{R_0}$$

Integration of this equation over the interval 0 to t gives

$$e^{\alpha\lambda_n^2 t}a_n(t) - a_{n_0} = C_n \left.\frac{e^{(\alpha\lambda_n^2-\mu)t}}{\alpha\lambda_n^2 - \mu}\right|_0^t = C_n\frac{e^{(\alpha\lambda_n^2-\mu)t}-1}{\alpha\lambda_n^2 - \mu}$$

or

$$a_n(t) = a_{n_0}e^{-\alpha\lambda_n^2 t} + C_n\left[\frac{e^{-\mu t}-e^{-\alpha\lambda_n^2 t}}{\alpha\lambda_n^2 - \mu}\right]$$

The only remaining unknown is the value for a_{n_0}. Inserting the expression for $h(x)$ into the above expression for a_{n_0} gives

$$a_{n_0} = \frac{2}{L}\int_0^L h(x)\sin\frac{n\pi x}{L}dx = \frac{2}{L}\frac{u_{R_0}}{L}\int_0^L x\sin\frac{n\pi x}{L}dx$$

$$= \frac{2u_{R_0}}{L^2}\left[\left(\frac{L}{n\pi}\right)^2\sin\frac{n\pi x}{L} - \left(\frac{L}{n\pi}\right)x\cos\frac{n\pi x}{L}\right]_0^L = \frac{2u_{R_0}}{L^2}\left[-\frac{L^2}{n\pi}(-1)^n\right]$$

or

$$a_{n_0} = \frac{2u_{R_0}}{\pi}\frac{(-1)^{n+1}}{n}$$

With this result, we have a complete set of expressions that define the expansion coefficients for the $v(x,t)$ problem. This transient solution, coupled with the steady state solution, $w(x)$, gives the final temperature profile, $u(x,t)$, for this system. For ease of implementation, a summary of the pertinent equations is listed below:

$$u(x,t) = v(x,t) + w(x)$$

with

$$v(x,t) = \sum_{n} \left\{ a_{n_0} e^{-\alpha\lambda_n^2 t} + C_n \left[\frac{e^{-\mu t} - e^{-\alpha\lambda_n^2 t}}{\alpha\lambda_n^2 - \mu} \right] \right\} \sin \lambda_n x$$

$$w(x) = \frac{u_L}{L}(L - x)$$

where

$$a_{n_0} = \frac{2u_{R_0}}{\pi} \frac{(-1)^{n+1}}{n} \qquad C_n = \frac{2\alpha\lambda_n u_{R_0}}{L}(-1)^{n+1} \qquad \lambda_n = \frac{n\pi}{L} \qquad \text{for} \quad n = 1, 2, \cdots$$

Matlab Implementation

The solution for this problem has been implemented and evaluated in Matlab file **ht1d_sov3.m**, and the resultant temperature profiles for various times have been plotted in Figs. 10.3 and 10.4, where the two sets of profiles are associated with the different values of the decay rate used in the simulations. The coding algorithm used here is similar to that discussed for Example 10.2 (see listing of **ht1d_sov2.m** in Table 10.2), with the listing of the Matlab simulation file for the current problem given in Table 10.3.

In the simulations shown here, we again see that the series expansion is not convergent because the nonzero right boundary condition cannot be achieved with the expansion functions used in this problem. The induced error is especially apparent at the initial time and early in the transient, when the difference between the actual boundary temperature and the predicted zero temperature is large. As the transient proceeds and the real endpoint temperature approaches zero, the spatial oscillations observed near the right boundary tend to subside, and the overall spatial temperature profile approaches expected conditions. The case with the rapidly decaying boundary temperature shows better agreement with expected behavior earlier in the transient, simply because the zero boundary value imposed by the expansion functions is reached more rapidly -- comparing Figs. 10.3 and 10.4 shows this quite clearly. Even with the obvious inaccuracy at the right boundary point, the time evolution of the temperature profiles and the actual temperature values near the left boundary follow expected behavior, with the temperature profile eventually approaching the steady state linear temperature distribution at long times (t = 80 min in Figs. 10.3 and 10.4).

This example represents a good illustration of the Eigenfunction Expansion Method and the Matlab file **ht1d_sov3.m**, once again, demonstrates how to numerically evaluate a relatively complicated infinite series expansion to assist in the visualization and analysis of the analytical solution to a given problem. Understanding this Matlab implementation is important since this same scheme can be applied to other similar problems of interest to the individual student.

Note: Finally, the reader should also see Example 11.4 in the next section of notes for another solution approach for this particular problem, where a numerical technique referred to as the *State Space Method* is used. Comparison of the two solutions shows that the analytical method is indeed approximate, but overall it does quite well in establishing the general space-time temperature profile for this system (near the left end of the bar the predictions are very good).
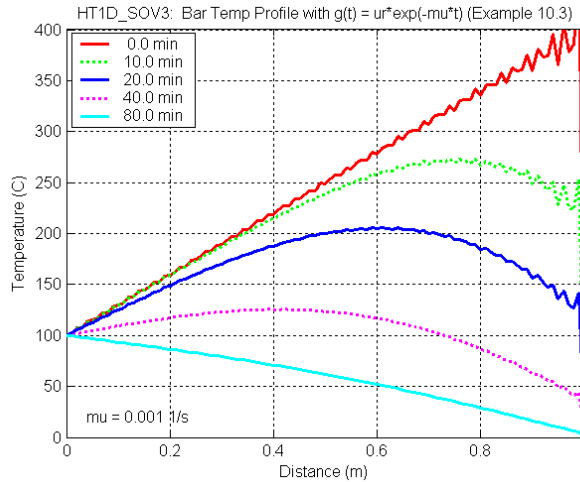
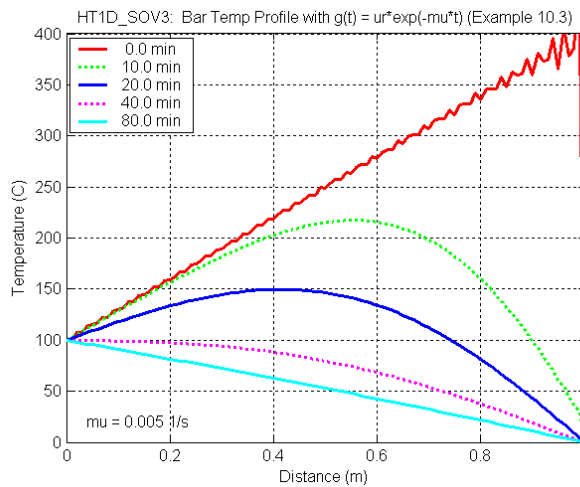**Fig. 10.3 Temperature profiles from Example 10.3 with slowly decaying $u_R(t)$.**

Fig. 10.4 Temperature profiles from Example 10.3 with rapidly decaying $u_R(t)$.

Table 10.3 Listing of Matlab file ht1d_sov3.m.

```
%
%   HT1D_SOV3.M    Heat Transfer in a 1-D Finite Bar (Example 10.3 in Class Notes)
%
%   Analytical Solution using Separation of Variables (SOV) to the following problem:
%      ut(x,t) = alf*uxx(x,t)
%   where
%      u(L,t) = g(t) = ur*exp(-mu*t)    --- decaying temperature on right side
%      u(0,t) = ul                      --- fixed temp on left
%      u(x,0) = f(x) = (ur-ul)x/L + ul  --- linear profile
%   has solution given by
%     u(x,t) = v(x,t) + w(x)
%   where v(x,t) is the transient solution and w(x) is the steady state solution.
%
%   The goal here is to evaluate and visualize the temperature profile in the
%   1-D finite bar of length L where the RHS endpoint temperature is time
%   dependent.  The Eigenfunction Expansion Method is used to approximate the
%   v(x,t) solution (see class notes).
%
%   File generated by J. R. White, UMass-Lowell (Aug. 2003)
```

```
%
%   getting started
      clear all,   close all,   nfig = 0;
%
%   problem data
      alf = 0.0001;     % m^2/s  thermal diffusivity
      L = 1;            % m      length of bar
      ul = 100;         % C      left (fixed) temp of bar
      ur = 400;         % C      initial temp on right
      maxt =  80;       % max terms in expansion
      tol = 0.001;      % tolerance used to stop series expansion
      imu = menu('Select decay rate of right side temp', ...
                 'Use slow decay rate (mu = 0.001 1/s)', ...
                 'Use rapid decay rate (mu = 0.005 1/s)');
      if imu == 1,   mu = .001;   end
      if imu == 2,   mu = .005;   end
%
%   vector of points to evaluate function
      Nx = 101;   x = linspace(0,L,Nx)';
%
%   w(x) is the steady state solution
      w = ul*(L-x)/L;
%
%   calc ln, ano and cn
      for n = 1:maxt
        ln(n) = n*pi/L;     ano(n) = (2*ur/(pi*n))*(-1)^(n+1);
        cn(n) = (2*alf*ur*ln(n)/L)*(-1)^(n+1);
      end
%
%   define time points and initialize space-time distribution
      Nt = 5;    tt = [0 600 1200 2400 4800];   % time in seconds
      u = zeros(Nx,Nt);
%
%   start loop over time points
      for i = 1:Nt
        t = tt(i);   bb = -mu*t;  cc = -alf*t;  v = zeros(Nx,1);
        n = 0;   mrerr = 1.0;
          while mrerr > tol  &   n < maxt
            n = n+1;  k1 = exp(bb);  k2 = exp(cc*ln(n)*ln(n));  k3 = alf*ln(n)*ln(n);
            vn = (ano(n)*k2 + cn(n)*(k1-k2)/(k3-mu))*sin(ln(n)*x);
            v = v + vn;
            j = find(v);                    % finds indices of nonzero values of v(x)
            mrerr = max(abs(vn(j)./v(j)));  % compute max relative error
          end
        u(:,i) = v + w;
        disp([' Needed ',num2str(n),' terms for convergence at t = ',num2str(t),' s'])
      end
%
%   set color and marker code for creating plots
      Ncm = 6;
      scm = ['r-';      % red solid
             'g:';      % green dotted
             'b-';      % blue solid
             'm:';      % magenta dotted
             'c-';      % cyan solid
             'y:'];     % yellow dotted
%
%   plot curves of u for various times
      st = zeros(Nt,9);
      if Nt < 7
        nfig = nfig+1;   figure(nfig)
        for i = 1:Nt
          h(i) = plot(x,u(:,i),scm(i,:),'LineWidth',2); hold on
          st(i,:) = sprintf('%5.1f min',tt(i)/60);
        end
        hold off
        axis([0 1 0 400]);
        title('HT1D\_SOV3:  Bar Temp Profile with g(t) = ur*exp(-mu*t) (Example 10.3)')
        grid,xlabel('Distance (m)'),ylabel('Temperature (C)')
        legend(h,char(st))
        text(0.04,20,['mu = ',num2str(mu),' 1/s'])
      end
%
%   end of problem
```

Example 10.4 – Heat Transfer in a Cylindrical Bar with Homogeneous BCs

Problem Description:

A long solid cylinder has an initial temperature profile given by f(r). If the rim temperature is zero for $t > 0$, find the temperature distribution, u(r,t), for this system.

Problem Solution:

This problem calls for a classic Separation of Variables (SOV) solution scheme since the defining heat conduction equation (i.e. the Diffusion Equation) and the boundary conditions are both homogeneous. Formally, this problem can be stated mathematically as

$$u_t(r,t) = \alpha\nabla^2 u(r,t) \qquad \text{with} \quad u_r(0,t) = 0, \quad u(R,t) = 0, \quad \text{and} \quad u(r,0) = f(r)$$

where the boundary condition at $r = 0$ says that the temperature gradient at this point is zero (symmetry condition). However, in 1-D cylindrical geometry the Laplacian operator can be expanded as

$$\nabla^2 = \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} \qquad (1-D \text{ cylindrical geometry})$$

and the defining PDE for this heat conduction problem with no internal energy generation becomes

$$u_t(r,t) = \alpha\nabla^2 u(r,t) = \alpha\left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r}\right)u(r,t) = \alpha\left(u_{rr} + \frac{1}{r}u_r\right)$$

Now, following the general outline for the classical SOV method, let's first assume a solution of the form

$$u(r,t) = F(r)G(t)$$

Substitution of this expression into the PDE gives

$$F\dot{G} = \alpha\left(F'' + \frac{1}{r}F'\right)G$$

where, as before, the "dot" notation implies a time derivative and the "primes" indicate spatial derivatives.

This equation is separable, or

$$\frac{F'' + \frac{1}{r}F'}{F} = \frac{\dot{G}}{\alpha G} = k = -\lambda^2$$

where k is referred to as a **separation constant**, and we have immediately forced this constant to be negative, since positive or zero values lead to trivial results (the reader should prove this latter statement if any doubt exists). As discussed for Example 10.1, the implication here is that, since the LHS is only a function of r and the RHS is only a function of t, for these terms to be equal to each other for all r and t, they must both be constants - which is indicated by the separation

constant, $k = -\lambda^2$. This process breaks the original PDE into two separate ODEs; one for the G(t) function and another for F(r). The two equations are

$$\dot{G} + \alpha\lambda^2 G = 0 \qquad \text{and} \qquad F'' + \frac{1}{r}F' + \lambda^2 F = 0$$

where the specification of the F-problem is completed with the conversion of the original BCs to the new notation. Formally, we have

$$u_r(0,t) = 0 \quad \Rightarrow \quad G(t)F'(0) = 0 \qquad \text{or} \qquad F'(0) = 0$$

$$u(R,t) = 0 \quad \Rightarrow \quad G(t)F(R) = 0 \qquad \text{or} \qquad F(R) = 0$$

Thus, the G-problem is a simple first order ODE with solution

$$G(t) = Ce^{-\alpha\lambda^2 t}$$

and the F-problem is a Sturm-Liouville problem (homogeneous ODE with homogeneous BCs). We also recognize the spatial problem as a special case of an ordinary Bessel equation of order zero, or

$$r^2 F'' + rF' + (\lambda^2 r^2 - 0)F = 0$$

Thus, the general solution for the spatial part of the problem can be written as

$$F(r) = C_1 J_0(\lambda r) + C_2 Y_0(\lambda r)$$

where J_0 and Y_0 are zero-order ordinary Bessel functions of the first and second kind, respectively. Also, we can compute the gradient, $F'(r)$, as

$$F'(r) = -C_1\lambda J_1(\lambda r) - C_2\lambda Y_1(\lambda r)$$

where the standard derivative formulas for the ordinary Bessel functions have been used, or

$$J_0'(\lambda r) = -\lambda J_1(\lambda r) \qquad \text{and} \qquad Y_0'(\lambda r) = -\lambda Y_1(\lambda r)$$

Now applying the boundary conditions to the general solution of the spatial problem, we have at the center of the solid rod,

$$F'(0) = 0 = -C_1\lambda \times (0) - C_2\lambda \times (-\infty)$$

where we have used the facts that $J_1(r)$ approaches zero as $r \to 0$ and $Y_1(r)$ approaches negative infinity as $r \to 0$. Thus, the only way to satisfy this condition is to let $C_2 = 0$. With this condition the spatial solution profile reduces to

$$F(r) = C_1 J_0(\lambda r)$$

Now, the second boundary condition at $r = R$ gives

$$F(R) = 0 = C_1 J_0(\lambda R)$$

Since $C_1 = 0$ would give a trivial solution, we must let $J_0(\lambda R) = 0$ to satisfy this condition. As we have seen previously, $J_n(r)$ is an oscillatory function and it has an infinite number of zeros, which we denote as β_{mn} for $m = 1, 2, \cdots$. β_{mn} represents the mth value of r for which $J_n(r) = 0$.

Therefore, from the above discussion, we have

$$J_0(\lambda R) = J_0(\beta_{m0}) = 0$$

as the *eigencondition* for this problem. This leads to

$$\lambda_n = \frac{\beta_{n0}}{R} \qquad \text{and} \qquad F_n(r) = J_0(\lambda_n r) \qquad \text{for} \quad n = 1, 2, \cdots$$

as the allowed *eigenvalues* and *eigenfunctions* for this problem.

Since we have multiple values of λ_n that satisfy the spatial problem, we also have multiple solutions for the G-problem. Rewriting the above G(t) solution, using the subscript n to denote the nth solution, gives

$$G_n(t) = C_n e^{-\alpha \lambda_n^2 t}$$

With known distribution functions for the spatial and temporal profiles, the desired temperature distribution can be written as

$$u(r,t) = \sum_n u_n(r,t) = \sum_n F_n(r)G_n(t) = \sum_n C_n J_0(\lambda_n r)e^{-\alpha \lambda_n^2 t}$$

where we have written the general solution in terms of a linear combination of the individual solutions.

The only remaining unknown left to be determined is the expansion coefficient, C_n. However, we still have an initial condition for this problem that has not yet been used. In particular, for $t = 0$ in the final expression for u(r,t), we have

$$u(r,0) = f(r) = \sum_n C_n J_0\left(\frac{\beta_{n0}}{R}r\right)$$

However, this is just a Fourier-Bessel series expansion for f(r) in terms of a set of orthogonal basis functions. The unknown expansion coefficients, C_n, can be determined by using the orthogonality property of the expansion functions. In particular, if we multiply both sides of the last expression by $rJ_0(\lambda_m r)$ and integrate over $0 < r < R$ [recall that the weight function for this situation is $p(r) = r$], we have

$$\int_0^R f(r)rJ_0\left(\frac{\beta_{m0}r}{R}\right)dr = \sum_n C_n \int_0^R rJ_0\left(\frac{\beta_{m0}}{R}r\right)J_0\left(\frac{\beta_{n0}}{R}r\right)dr$$

or

$$C_n = \frac{1}{\frac{R^2}{2}J_1^2(\beta_{n0})}\int_0^R f(r)rJ_0\left(\frac{\beta_{n0}}{R}r\right)dr$$

where the denominator represents the norm of the J_0 Bessel functions, or

$$\int_0^R rJ_0\left(\frac{\beta_{m0}}{R}r\right)J_0\left(\frac{\beta_{n0}}{R}r\right)dr = \frac{R^2}{2}J_1^2(\beta_{n0})\delta_{mn}$$

Recall that this normalization result was developed formally as part of Example 9.3. Finally, given a specific initial distribution, $f(r)$, the final expression for C_n can be evaluated for the desired expansion coefficients, thus completing the problem.

Since no specific initial temperature profile was given and no numerical data were specified for this problem, we cannot formally carry this solution any further (one needs numerical data for implementation into Matlab, for example). However, we can summarize the problem by gathering the formulas of interest for this case, giving

$$u(r,t) = \sum_n C_n J_0(\lambda_n r)\, e^{-\alpha \lambda_n^2 t}$$

with

$$\lambda_n = \frac{\beta_{n0}}{R} \qquad \text{and} \qquad C_n = \frac{2}{R^2 J_1^2(\beta_{n0})} \int_0^R f(r) r J_0(\lambda_n r)\, dr$$

where β_{n0} are the zeros of the $J_0(r)$ Bessel function.

With specific values for α and R, and a given initial temperature profile, f(r), these expressions lead to an exact analytical solution, u(r,t), for this problem.

Example 10.5 -- Steady State Heat Transfer in a 2-D Block

Problem Description:

Consider a long rectangular bar with a thermal conductivity of 1 W/m-C. As shown in the sketch, the top surface is exposed to a convective environment with h = 100 W/m²-C and T_∞ = 100 °C. The other three sides are held at a fixed temperature of T_w = 50 °C.
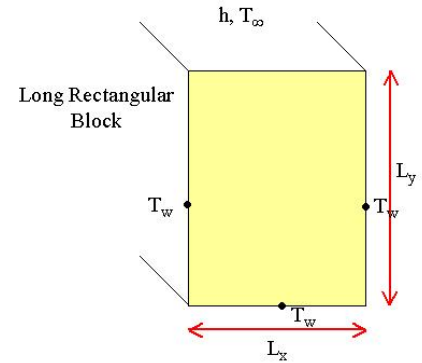
For this situation, compute and plot the 2-D temperature distribution, T(x,y), and determine the heat transfer on each face of this bar. We will actually solve this problem using several techniques, but here we will focus on an analytical solution using the separation of variables (SOV) method.
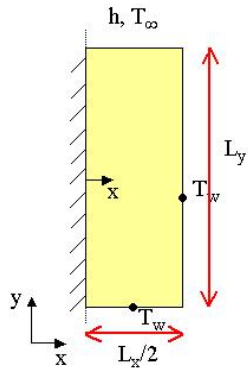
From symmetry, we can draw the geometry of interest as shown to the left. Within the block, heat transfer is via conduction and, for steady state, the defining balance equation becomes (with no internal energy generation)

$$\nabla^2 T = 0$$

which, for Cartesian geometry, can be written as

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) T(x,y) = 0$$

For convenience, we will define u = T – T_w. With this definition the complete mathematical description of the problem can be written as

$$u_{xx} + u_{yy} = 0$$

with the four BCs written as

    left BC: $u_x(0,y) = 0$ right BC: $u(L_x/2,y) = 0$

  bottom BC: $u(x,0) = 0$ top BC: $-ku_y(x,L_y) = h\left(u(x,L_y) - T_e \right)$

where we have defined $T_e = T_\infty - T_w$.

This is the problem we want to solve, with the following parameters:

      L_x = 0.06 m L_y = 0.09 m

      k = 1 W/m-C   h = 100 W/m²-C

      T_w = 50 °C T_∞ = 100 °C T_e = 50 °C

Problem Solution:

Theoretical Development

We first note that the PDE and the BCs in the x-direction are homogeneous. Thus, a classical solution via the SOV method should work. As such, we can follow the traditional outline as given previously. First we let

$$u(x,y) = F(x)G(y)$$

and then, upon substitution into the defining PDE, we have

$$F''G + FG'' = 0$$

and dividing by FG gives

$$\frac{F''}{F} = -\frac{G''}{G} = -\lambda^2$$

where λ is the separation constant (note that the choice of the sign distribution was made based on the fact that the x-direction has homogeneous boundary conditions). This expression now leads to two separate ODEs, as follows:

x-direction

The defining ODE in the x-direction is

$$F'' + \lambda^2 F = 0$$

with general solution

$$F(x) = A_1 \sin \lambda x + A_2 \cos \lambda x$$

The left boundary for this problem (at x = 0) is a line of symmetry, which gives

$$u_x(0,y) = 0 \qquad \text{which leads to} \qquad F'(0) = 0$$

Taking the indicated derivative

$$F'(x) = A_1 \lambda \cos \lambda x - A_2 \lambda \sin \lambda x$$

and substitution into the BC gives

$$F'(0) = 0 = A_1 \lambda (1) - 0 \qquad \text{or} \qquad A_1 = 0$$

Thus, the solution to the F-problem reduces to

$$F(x) = A_2 \cos \lambda x$$

Now applying the right BC, we have

$$u(L_x/2, y) = 0 \qquad \text{which leads to} \qquad F(L_x/2) = 0$$

or

$$F\left(L_x/2\right) = 0 = A_2 \cos \lambda L_x/2$$

But we know that the cosine function is periodic with an infinite number of zero crossings. Also, since the zeros occur at odd integer multiples of $\pi/2$, we have

$$\cos \lambda L_x / 2 = \cos(2n - 1) \pi / 2 = 0 \qquad \text{for} \quad n = 1, 2, 3, \cdots$$

which gives the following eigenvalues for this problem

$$\lambda_n = \frac{(2n - 1)\pi}{L_x}$$

With an infinite number of eigenvalues, we can rewrite F(x) with an n subscript as

$$F_n(x) = \cos \lambda_n x$$

where we have arbitrarily set the coefficient to unity.

y-direction

Now, focusing on the y direction, the ODE of interest is

$$G'' - \lambda^2 G = 0$$

which has the general solution

$$G(y) = C_1 \sinh \lambda y + C_2 \cosh \lambda y$$

The bottom BC (at y = 0) has a fixed temperature

$$u(x, 0) = 0 \qquad \text{which leads to} \qquad G(0) = 0$$

and, upon substitution, we have

$$G(0) = 0 + C_2(1) \qquad \text{or} \qquad C_2 = 0$$

This leaves

$$G_n(y) = \sinh \lambda_n y$$

where again we have set the normalization to unity and we have inserted the subscript n to imply that there is a different solution for each λ_n for all integer $n > 0$.

With explicit expressions for F(x) and G(y), we can write the nth solution as

$$u_n(x, y) = F_n(x) G_n(y) = B_n \cos \lambda_n x \sinh \lambda_n y$$

and, since the original PDE is linear, then we can form a general solution as the linear combination of the individual solutions, or

$$u(x, y) = \sum_{n=1}^{\infty} u_n(x, y) = \sum_{n=1}^{\infty} B_n \cos \lambda_n x \sinh \lambda_n y$$

Now, we still need to satisfy the top BC, which says that

$$-k u_y(x, L_y) = h u(x, L_y) - h T_e$$

Written out in detail, this becomes

$$-k\sum_{n=1}^{\infty} B_n \lambda_n \cos\lambda_n x \cosh\lambda_n L_y = h\sum_{n=1}^{\infty} B_n \cos\lambda_n x \sinh\lambda_n L_y - hT_e$$

To evaluate this expression for the unknown B_n coefficients, we multiply both sides by $\cos\lambda_m x$, perform a little algebra to collect similar terms, and integrate over the x domain,

$$\sum_{n=1}^{\infty} B_n \left(h\sinh\lambda_n L_y + k\lambda_n \cosh\lambda_n L_y\right) \int_0^{L_x/2} \cos\lambda_m x \cos\lambda_n x dx = \int_0^{L_x/2} hT_e \cos\lambda_m x dx$$

To actually do the integral on the LHS, we note that

$$\int_0^{L_x/2} \cos\lambda_m x \cos\lambda_n x dx = \left[\frac{\sin(\lambda_m - \lambda_n)x}{2(\lambda_m - \lambda_n)} + \frac{\sin(\lambda_m + \lambda_n)x}{2(\lambda_m + \lambda_n)}\right]_0^{\frac{L_x}{2}} \qquad \text{for } m \neq n$$

But, with $\lambda_n = (2n-1)\pi/L_x$, we have the following relationships:

$$(\lambda_m - \lambda_n)\frac{L_x}{2} = (2m-1)\frac{\pi}{2} - (2n-1)\frac{\pi}{2} = 2(m-n)\frac{\pi}{2} = (m-n)\pi$$

$$(\lambda_m + \lambda_n)\frac{L_x}{2} = (2m-1)\frac{\pi}{2} + (2n-1)\frac{\pi}{2} = \left(2(m+n)-2\right)\frac{\pi}{2} = (m+n-1)\pi$$

and $\sin(p\pi) = 0$, where p = integer. Thus, the LHS is zero for $m \neq n$.

When, $m = n$, we have

$$\int_0^{L_x/2} \cos^2\lambda_m x dx = \left[\frac{x}{2} + \frac{\sin 2\lambda_m x}{4\lambda_m}\right]_0^{\frac{L_x}{2}} = \frac{L_x}{4} + \frac{\sin(2m-1)\pi}{4\lambda_m} = \frac{L_x}{4}$$

Thus, we see that the orthogonality relationship can be summarized with the following expression:

$$\int_0^{L_x/2} \cos\lambda_m x \cos\lambda_n x dx = \frac{L_x}{4}\delta_{mn} = \begin{cases} 0 & m \neq n \\ \dfrac{L_x}{4} & m = n \end{cases}$$

which greatly simplifies the LHS of the above equation.

Now, focusing on the RHS, we have

$$hT_e \int_0^{L_x/2} \cos\frac{(2m-1)\pi x}{L_x} dx = \frac{hT_e}{\lambda_m}\sin\frac{(2m-1)\pi x}{L_x}\Big|_0^{\frac{L_x}{2}} = \frac{hT_e}{\lambda_m}\sin(2m-1)\frac{\pi}{2} = (-1)^{m+1}\frac{hT_e}{\lambda_m}$$

Now, equating the LHS and RHS, we have

$$B_m(h\sinh\lambda_m L_y + k\lambda_m \cosh\lambda_m L_y)\left(\frac{L_x}{4}\right) = (-1)^{m+1}\frac{hT_e}{\lambda_m}$$

or

$$B_m = \frac{(-1)^{m+1} 4hT_e}{L_x \lambda_m (h \sinh \lambda_m L_y + k \lambda_m \cosh \lambda_m L_y)}$$

which completes our derivation. The expressions for λ_n and B_n, substituted into the series representation for u(x,y), gives the formal SOV solution for this problem.

The original problem specification also requested an evaluation of the heat loss at the boundaries of the block. With the above analytical solution, we can also compute the heat flux everywhere as follows:

$$\bar{q} = -k\bar{\nabla}T = -k\frac{\partial}{\partial x}T(x,y)\hat{i} - k\frac{\partial}{\partial y}T(x,y)\hat{j}$$

and, since $T = u + T_w$, we have

$$\bar{q} = -k\frac{\partial}{\partial x}u(x,y)\hat{i} - k\frac{\partial}{\partial y}u(x,y)\hat{j}$$

Writing this vector quantity in component form, we have

$$\bar{q} = q_x\hat{i} + q_y\hat{j}$$

where

$$q_x = k\sum_{n=1}^{\infty}B_n\lambda_n \sin\lambda_n x \sinh\lambda_n y \quad \text{and} \quad q_y = -k\sum_{n=1}^{\infty}B_n\lambda_n \cos\lambda_n x \cosh\lambda_n y$$

Note that this vector function can be plotted with the *quiver* command in Matlab.

We can also compute the total rate at which energy crosses a boundary, as follows:

$$q_{top} = \int_0^{L_x/2} \bar{q}(x,L_y)\cdot\hat{n}dx = \int_0^{L_x/2} q_y(x,L_y)dx = -k\sum_{n=1}^{\infty}B_n \sin\lambda_n L_x/2\left(\cosh\lambda_n L_y\right)$$

$$q_{right} = \int_0^{L_x/2} q_x(L_x/2,y)dy = k\sum_{n=1}^{\infty}B_n \sin\lambda_n L_x/2\left(\cosh\lambda_n L_y - 1\right)$$

$$q_{bottom} = \int_0^{L_x/2} q_y(x,0)dx = -k\sum_{n=1}^{\infty}B_n \sin\lambda_n L_x/2$$

$$q_{left} = \int_0^{L_y} q_x(0,y)dy = 0$$

Here, q has units of watts per unit length into the z direction, where for z = 1 m, we get q in W.

Note also that we can compute the heat transfer along the top surface with Newton's law of cooling, or

$$q_{top}\Big|_{conv} = \int_0^{L_x/2} h\left(u(x,L_y) - T_e\right)dx = h\sum_{n=1}^{\infty}B_n \frac{1}{\lambda_n}\sin\lambda_n L_x/2\left(\cosh\lambda_n L_y\right)$$

Matlab Implementation

The above expressions for the temperature profile, heat flux distribution, and heat loss rates were implemented into Matlab file **block2d_sov1.m** as listed in Table 10.4, and a series of graphical results from this program are plotted in Figs. 10.5 and 10.6. Matlab's *meshgrid* command was used to lay out the x,y spatial grid for evaluation of the analytical expressions for T(x,y) and the heat fluxes, and the series expansion was continued until the maximum change in temperature and heat flux was less than 0.01 (with units of C and W/m^2, respectively) or the maximum number of terms was reached.

Note: It turns out that this problem has a very large temperature gradient and corresponding heat flux in the upper right corner of the model, and there were some additional subtle numerical considerations (associated with possible round off errors) that were not handled as effectively as possible inside **block2d_sov1.m**. Thus, in all cases, convergence to the desired level of 0.01 was not achieved -- that is, we hit the **maxt** limit each time. However, the qualitative results given here should be just fine…
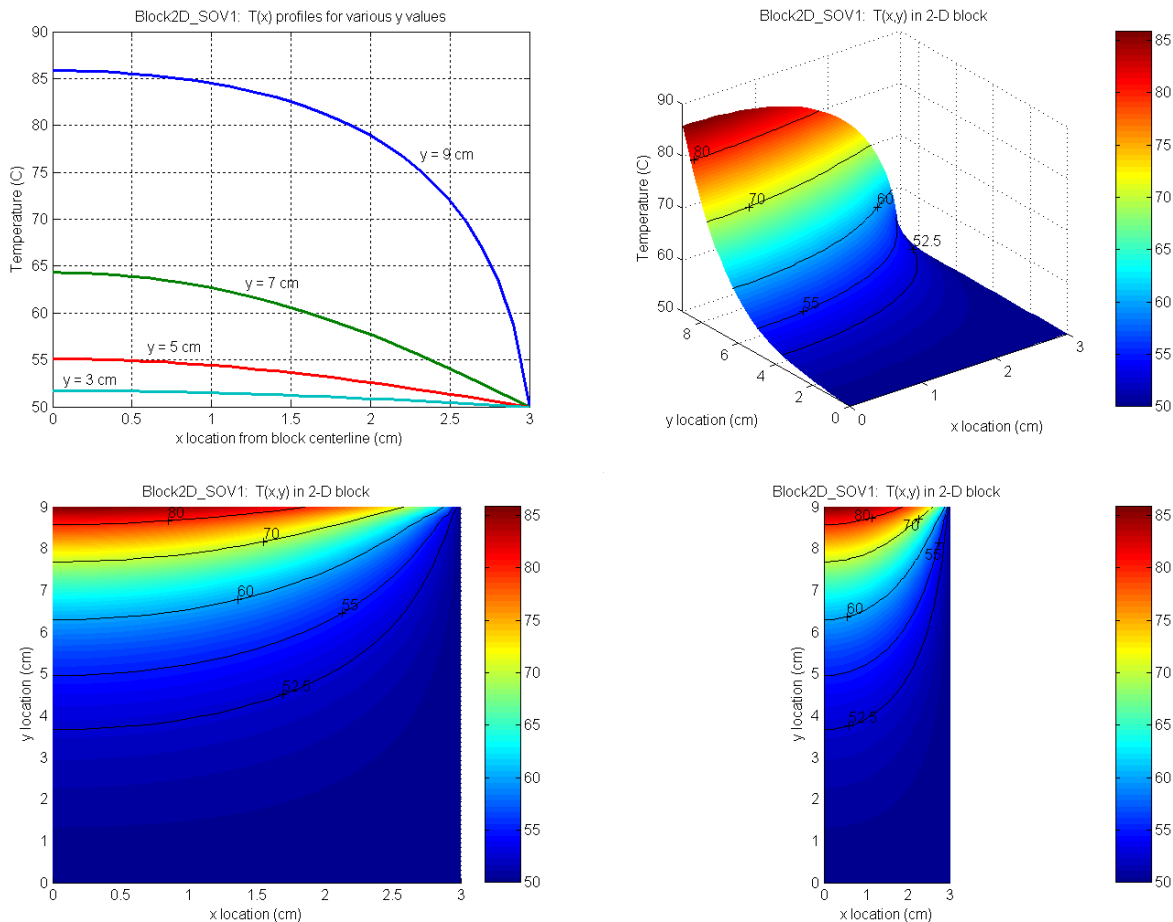

Fig. 10.5 Several views for T(x,y) in Example 10.5.

Focusing on the base results, we have plotted four different views of the temperature distribution in Fig. 10.5 and a 3-D surface plot of the magnitude of the heat flux, $|q| = \sqrt{q_x^2 + q_y^2}$, in Fig. 10.6. All the temperature profiles are as expected, with a peak temperature near 86 C in the center of the block along the upper boundary. The temperature drops off towards 50 C as one moves away from the peak, either towards the right boundary at $x = L_x/2$ or the bottom boundary at $y = 0$. Because we forced a fixed temperature of 50 C along the bottom and right boundaries, this behavior was expected.
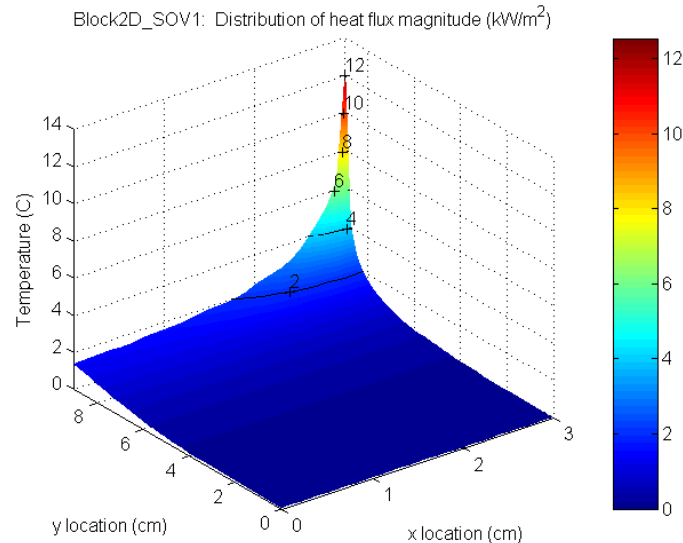

Fig. 10.6 Distribution of the heat flux magnitude for the 2-D block example.

There is, however, some interesting behavior in the upper right corner of the model. Recall that, on the right face, we have forced a fixed temperature of $T_w = 50$ C, and on the top surface (just an infinitesimal distance away), we have a convective condition with $T_\infty = 100$ C. These constraints force a very large temperature gradient in the area where the two surfaces intersect. This is especially apparent in Fig. 10.6 where the surface plot of the heat flux magnitude is given -- and clearly we can see the large peak in the upper right corner. Note that, even though this behavior is not overly realistic, it is consistent with the imposed BCs for the given problem (in a realistic illustration, the right side temperature would be a function of y).

Quantitative results for the heat loss or gain from each side of the 2-D block were also computed and the actual output from **block2d_sov1.m** is reproduced below:

```
Block2D_SOV1 -- Heat transfer (W) across boundaries for the right half of a 2-D block
   in top boundary (conduction):     61.70
   in top boundary (convection):     62.30
   out right boundary:               60.94
   out bottom boundary:               0.75
   out left boundary:                 0.00
```

Clearly, all the energy enters the block through the top surface and it leaves from the right and bottom surfaces -- with the lion's share of the overall energy transfer taking place near the upper right corner of the block. Since the left boundary in the half-block model is a line of symmetry, there is no heat transfer along this surface. Also note that both the convective and conductive heat gains along the top boundary were computed. Of course, the two values should be identical, but because of numerical difficulties and the large gradients seen here, a small difference is observed. This observation is fairly typical, where usually the convective result, which involves integration (rather than differentiation), is somewhat more accurate. Here, the observed difference is nearly negligible…

Finally, we note that this same problem is also solved again in Section XI of these notes using the Finite Difference (FD) method and in the Appendix using FEMLAB and Matlab's PDE Toolbox. Thus, you should be sure that you have a good handle on this problem, since it will be used again for comparison purposes as we discuss other techniques for solution of PDEs of this type (using numerical schemes).

Table 10.4 Listing of Matlab file block2d_sov1.m.

```
%
%   BLOCK2D_SOV1      2-D Heat Conduction in a Rectangular Block
%
%   Analytical Soln using Separation of Variables (SOV) to the following heat
%   conduction problem:
%     uxx(x,y) + uyy(x,y)  = 0     with ux(0,y) = 0   u(Lx/2,y) = 0
%                                       u(x,0) = 0   -kuy(x,Ly) = h(u - Te)
%     where  u(x,y) = T(x,y) - Tw   and   Te = Tinf - Tw
%
%   The goal here is to plot the analytical solution a variety of ways to
%   help visualize the 2-D temperature distribution.  We also want to compute
%   the heat loss or gain across each face of the rectangular block.  Only the
%   right half of the block is modeled  --  a symmetry condition at x = 0
%   handles this (note that the heat flows given are only for the portion of
%   the block modeled).
%
%   File prepared by J. R. White, UMass-Lowell (Aug. 2003)
%


%
%   getting started
      clear all,   close all,   nfig = 0;
%
%   problem data
      Tw = 50;           % wall temperature on left, bottom, and right sides (C)
      Tinf = 100;        % air temperature on top of block (C)
      Te = Tinf-Tw;      % difference between air and wall temps (C)
      h = 100;           % heat transfer coeff (W/m^2-C)
      k = 1;             % thermal conductivity of block (W/m-C)
      Lx = 0.06;         % width of block in x direction (m)
      Ly = 0.09;         % height of block in y direction (m)
      maxt = 50;         % max number of nonzero terms
      tol = 0.01;        % convergence limit for series expansion (absolute temp diff)
%
%   points to evaluate function (domain of interest is a rectangle)
      Nx = 31;    x = linspace(0,Lx/2,Nx);
      Ny = 37;    y = linspace(0,Ly,Ny);
      [xx,yy] = meshgrid(x,y);
%
%   calc eigenvalues and expansion coeffs
%   Note: Since the bn values become so small so quickly, we can expect that
%         numerical errors may become a problem (small # * large # in series expansion).
%         Because of this I really should implement the series expansion differently
%         than given here --- use recursive form.  However, our focus here is
```

```
%           simply to get a reasonable visualization of the temperature and heat flux
%           distributions, so we will keep the implementation on the simple side (and
%           just be careful to make sure that the numerical errors do not affect the
%           qualitative observations)
      dd = pi/Lx;      cc = 4*h*Te/Lx;
      for n = 1:maxt
        ln(n) = (2*n-1)*dd;
        bn(n) = (-1)^(n+1)*cc/ln(n)/(h*sinh(ln(n)*Ly) + k*ln(n)*cosh(ln(n)*Ly));
      end
%
%    determine spatial temperature distribution
      n = 0;    merr = 1.0;    u = zeros(Ny,Nx);
      while merr > tol   &   n < maxt
        n = n+1;    un = bn(n)*cos(ln(n)*xx).*sinh(ln(n)*yy);
        u = u + un;     merr = max(max(abs(un)));
      end
      fprintf(1,'\n  Needed %3i terms for convergence of u(x,y) \n\n',n)
%
%    now form actual temperature distribution of interest  --  T(x,y)
      T = u + Tw;
%
%    plot T(x,y) for a few y values
      nfig = nfig+1;  figure(nfig)
      plot(x*100,T(Ny:-8:13,:),'LineWidth',2),grid
      title('Block2D\_SOV1:  T(x) profiles for various y values')
      xlabel('x location from block centerline (cm)');
      ylabel('Temperature (C)')
      for j = Ny:-8:13,  gtext(['y = ',num2str(y(j)*100),' cm']);    end
%
%    various 3-D plots
      nfig = nfig+1;  figure(nfig)
      surf(xx*100,yy*100,T), shading interp, colorbar
      axis([0 Lx*100/2 0 Ly*100 50 90])
      title('Block2D\_SOV1:  T(x,y) in 2-D block')
      xlabel('x location (cm)');   ylabel('y location (cm)')
      zlabel('Temperature (C)')
%
      nfig = nfig+1;  figure(nfig)
      surf(xx*100,yy*100,T), shading interp, colorbar, hold on
      [cc,hh] = contour3(x*100,y*100,T, [80 70 60 55 52.5]);
      clabel(cc), set(hh,'EdgeColor','k')
      axis([0 Lx*100/2 0 Ly*100 50 90])
      title('Block2D\_SOV1:  T(x,y) in 2-D block')
      xlabel('x location (cm)');   ylabel('y location (cm)')
      zlabel('Temperature (C)'),   hold off
%
      nfig = nfig+1;  figure(nfig)
      surf(xx*100,yy*100,T), shading interp, view(2), colorbar, hold on
      [cc,hh] = contour3(x*100,y*100,T, [80 70 60 55 52.5]);
      clabel(cc), set(hh,'EdgeColor','k')
      axis([0 Lx*100/2 0 Ly*100 50 90])
      title('Block2D\_SOV1:  T(x,y) in 2-D block')
      xlabel('x location (cm)');   ylabel('y location (cm)')
      zlabel('Temperature (C)'),   hold off
%
      nfig = nfig+1;  figure(nfig)
      surf(xx*100,yy*100,T), shading interp, view(2), colorbar, hold on
      [cc,hh] = contour3(x*100,y*100,T, [80 70 60 55 52.5]);
      clabel(cc), set(hh,'EdgeColor','k')
      axis equal
      axis([0 Lx*100/2 0 Ly*100 50 90])
      title('Block2D\_SOV1:  T(x,y) in 2-D block')
      xlabel('x location (cm)');   ylabel('y location (cm)')
      zlabel('Temperature (C)'),   hold off
%
%    determine spatial heat flux distributions
%    x-directed heat flux
      n = 0;    merr = 1.0;    qx = zeros(Ny,Nx);
      while merr > tol    &    n < maxt
        n = n+1;    qxn = k*bn(n)*ln(n)*sin(ln(n)*xx).*sinh(ln(n)*yy);
        qx = qx + qxn;    merr = max(max(abs(qxn)));
      end
      fprintf(1,'\n  Needed %3i terms for convergence of qx(x,y) \n\n',n)
%    y-directed heat flux
      n = 0;    merr = 1.0;    qy = zeros(Ny,Nx);
```

```
        while merr > tol   &   n < maxt
          n = n+1;    qyn = -k*bn(n)*ln(n)*cos(ln(n)*xx).*cosh(ln(n)*yy);
          qy = qy + qyn;   merr = max(max(abs(qyn)));
        end
        fprintf(1,'\n  Needed %3i terms for convergence of qy(x,y) \n\n',n)
%
%   plot the heat flux as a vector plot
        nfig = nfig+1;  figure(nfig)
        quiver(xx*100,yy*100,qx,qy),grid
        axis([0 Lx*100/2 0 Ly*100])
        title('Block2D\_SOV1:  Heat flux vector plot for 2-D block')
        xlabel('x location (cm)');
        ylabel('y location (cm)')
%
%   plot magnitude of the heat flux
        qmag = sqrt(qx.*qx + qy.*qy);
        nfig = nfig+1;  figure(nfig)
        surf(xx*100,yy*100,qmag/1000), shading interp, colorbar, hold on
        [cc,hh] = contour3(x*100,y*100,qmag/1000);
        clabel(cc), set(hh,'EdgeColor','k')
        axis([0 Lx*100/2 0 Ly*100 0 14])
        title('Block2D\_SOV1:  Distribution of heat flux magnitude (kW/m^2)')
        xlabel('x location (cm)');   ylabel('y location (cm)')
        zlabel('Temperature (C)'),   hold off
%
%   now compute the heat flows across each surface (use all terms)
        qtop1 = -k*sum(bn.*sin(ln*Lx/2).*cosh(ln*Ly));
        qtop2 = h*sum(bn.*sin(ln*Lx/2).*sinh(ln*Ly)./ln) - h*Te*Lx/2;
        qrt = k*sum(bn.*sin(ln*Lx/2).*(cosh(ln*Ly)-1));
        qbot = -k*sum(bn.*sin(ln*Lx/2));
        qlt = 0;
%
        fprintf(1,'\n  Block2D_SOV1 -- Heat transfer (W) across boundaries for the right half of a
2-D block \n')
        fprintf(1,'    in top boundary (conduction):   %8.2f  \n',-qtop1)
        fprintf(1,'    in top boundary (convection):   %8.2f  \n',-qtop2)
        fprintf(1,'    out right boundary:             %8.2f  \n',qrt)
        fprintf(1,'    out bottom boundary:            %8.2f  \n',-qbot)
        fprintf(1,'    out left boundary:              %8.2f  \n',-qlt)
%
%   end of problem
```