AC 2008-933: MICROSOFT EXCEL HEAT TRANSFER ADD-IN FOR ENGINEERING COURSES

Troy Dent, University of Alabama

Keith Woodbury, University of Alabama

Robert Taylor, University of Alabama

Excel Heat Transfer Add-in for Engineering Courses

Abstract

The application of computer software is central in modern engineering instruction. Software has been specifically designed for courses and some textbooks are packaged with specialized versions of popular software. However, a survey of University of Alabama alumni shows that the majority primarily uses MS Excel for engineering computations compared to those that commonly use engineering software packages. The wide availability of MS Excel contributes to its use. With this in mind, an Add-in for MS Excel is being developed to provide a useful package of engineering functions for heat transfer instruction. The initial Add-in was developed to calculate the dimensionless temperature for 1-D transient heat conduction in a solid. The Add-in includes 4 functions which handle a plane wall, infinite cylinder, sphere and semi-infinite solid. Additional modules have been developed to calculate the local or average Nusselt number for internal or external flows and the view factors for different 3-D radiation heat transfer set-ups. Currently, these three modules are presented separately as part of a Heat Transfer course, but could be combined into a single Add-in.

The paper discusses the development, testing and application of these Add-in modules. A major part of the effort was the development of algorithms to compute the transient conduction solutions in a timely manner. After some refinement, sufficiently efficient functions were developed. The Add-in provides a tool for engineers for the widely-used MS Excel. As a learning tool, the Add-in provides a demonstration of the underlying formulas for a heat transfer course.

Introduction

As more computer programs and software become available for engineering, it has become necessary for universities to include instruction involving the usage of these software packages. Software has been specifically designed for courses and some textbooks are packaged with specialized versions of popular software. However, a survey of University of Alabama alumni shows that the majority primarily uses MS Excel for engineering computations, with a fewer number that commonly use engineering software packages. The wide availability of MS Excel contributes to its popularity. With this in mind, an Add-in for MS Excel is being developed that would provide a useful package of engineering functions. The initial Add-in was developed to calculate the dimensionless temperature for 1-D transient heat conduction in a solid. The Add-in includes 4 functions which handle a plane wall, infinite cylinder, sphere and semi-infinite solid. The Add-in needed to provide results with accuracy comparable to other software packages and have an efficient run-time. Short run-times for the individual calculations are necessary for the multiple calculations to generate tables and charts. Additional modules have been developed to calculate the local or average Nusselt number for internal or external flows and the view factors for different 3-D radiation heat transfer set-ups. Currently, these three modules are presented separately as part of a Heat Transfer course, but could be combined into a single Add-in.

For the 1-D transient heat conduction functions, the calculations for the plane wall, infinite cylinder and sphere involve finding eigenvalues and performing an infinite series summation; so, they were developed together with changes only for the different equations related to each shape.

The semi-infinite solid does not involve eigenvalues or summations but has equations for 3 different boundary conditions. Its development will be described separately for the boundary conditions of constant temperature, constant heat flux and convection. The Nusselt number functions and view factor functions are not as complicated in their development but do show the advantage of providing simplified functions to replace the usually convoluted calculations.

The Add-in modules can be viewed and downloaded from the website www.me.ua.edu/excelinme.

Microsoft Excel Add-ins

The Add-ins for Microsoft Excel can provide additional functionality for Excel by loading custom functions and macros. Engineers familiar with Excel are probably already aware of the Solver and Analysis ToolPak Add-ins that are packaged with Excel. Excel Add-in files can include macros or custom functions written in the Visual Basic Editor. Excel Add-in files are created by saving the workbook as an .xla (Excel 2003 and earlier) or .xlam (Excel 2007) file type. The files are loaded into Excel 2003 by selecting Tools in the main toolbar then clicking Add-ins in the drop-down menu. In order to load an Add-in that is not packaged with Excel, it is necessary to click the Browse button and locate the file on your computer. Getting to the Add-ins window in Excel 2007 is a slightly more complicated process. The user needs to click the Office Button in the top left corner of Excel, and then click the Excel Options button at the bottom of the drop-down. On the new window, select Add-ins from the list on the left, then click Go for Manage Add-ins at the bottom of the window. The list of available Add-ins should appear and works the same as Excel 2003.

Simple program, 1-D Transient Heat Conduction

Rather than immediately trying to construct the best program possible, a simple program is first written that performs the calculation for 1-D transient heat conduction correctly, but is not very efficient or user-friendly. Then improvements are made to the simple program for better efficiency and features. By creating the program in increments, it is possible to confirm that the program functions correctly and locate errors more readily since only a small portion of the code has changed since the last correct version. The first three functions that were created are 1-D transient heat conduction in a plane wall, infinite cylinder and sphere [1]. These functions take the form of Tslab(xstar, Fo, Bi), Tcyl(rstar, Fo, Bi) and Tsph(rstar, Fo, Bi). The inputs are the dimensionless lengths (xstar and rstar), the Fourier number and the Biot number as defined below. The functions return the dimensionless temperature, θ^*, based on the following equations for each shape.

The plane wall temperature profile is

$$\theta^* = \frac{T - T_\infty}{T_i - T_\infty} = \sum_{n=1}^{\infty} C_n \exp(-\zeta_n^2 Fo) \cos(\zeta_n x^*) \qquad (1)$$

where T_i is the initial temperature of the solid, T_∞ is the ambient fluid temperature, $Fo = \alpha t / L^2$ is the Fourier number, x* is the dimensionless length from the center equal to x / L, where L is

half of the thickness of the wall with the same convection boundary condition on both sides. The coefficient C_n is

$$C_n = \frac{4\sin(\zeta_n)}{2\zeta_n + \sin(2\zeta_n)} \tag{2}$$

and ζ_n are the eigenvalues that are the positive roots of the equation

$$\zeta_n \tan\zeta_n = Bi \tag{3}$$

where Bi = h L / k is the Biot number. There are similar equations for an infinite cylinder involving the Bessel functions $J_0(x)$ and $J_1(x)$ and dimensionless radius r* = r / R where R is the radius of the cylinder, with Bi and Fo defined as above except R is the length scale.

$$\theta^* = \sum_{n=1}^{\infty} C_n \exp(-\zeta_n^2 Fo) J_0(\zeta_n r^*) \tag{4}$$

$$C_n = \frac{2}{\zeta_n} \frac{J_1(\zeta_n)}{J_0^2(\zeta_n) + J_1^2(\zeta_n)} \tag{5}$$

$$\zeta_n \frac{J_1(\zeta_n)}{J_0(\zeta_n)} = Bi \tag{6}$$

Finally, the equations for a solid sphere are

$$\theta^* = \sum_{n=1}^{\infty} C_n \exp(-\zeta_n^2 Fo) \frac{\sin(\zeta_n r^*)}{\zeta_n r^*} \tag{7}$$

$$C_n = \frac{4\left[\sin(\zeta_n) - \zeta_n \cos(\zeta_n)\right]}{2\zeta_n - \sin(2\zeta_n)} \tag{8}$$

$$1 - \zeta_n \cot(\zeta_n) = Bi \tag{9}$$

Although the equations for each shape are different, the code will be similar since all three include finding ζ_n and performing the summations. The equations for ζ_n were rewritten so the left-hand side of the equation equals 0. For each n of the summation, ζ_n is the root of the equations,

$$f_{PW}(\zeta_n) = \zeta_n \sin\zeta_n - Bi \cdot \cos\zeta_n = 0 \tag{10}$$

$$f_{IC}(\zeta_n) = \zeta_n \cdot J_1(\zeta_n) - Bi \cdot J_0(\zeta_n) = 0 \tag{11}$$

$$f_{sp}(\zeta_n) = \sin\zeta_n - \zeta_n \cos(\zeta_n) - Bi \cdot \sin\zeta_n = 0 \tag{12}$$

for the plane wall, infinite cylinder and sphere, respectively. The equations have also been multiplied by the denominator that was present in each of them. Before multiplying by denominator, each equation had singularities, which caused problems with the program initially. Multiplying by the denominator changes the equations so they do not go to infinity but does not affect the value of ζ_n.

For the simple program, ζ_n was found by interval halving. The first step was determining an upper and lower limit for ζ_n in terms of n. The ζ_n's for the plane wall lie in the range of the positive values of tan(x). This gives a range in terms of n of $(n-1)\,\pi \le \zeta_n \le (n-0.5)\,\pi$. Although the range for the positive values of cot(x) is $(n-0.5)\,\pi \le x \le n\,\pi$, the range for the spherical eigenvalues is $(n-1)\,\pi \le \zeta_n \le n\,\pi$ because ζ_n approaches 0 as Bi approaches 0 due to the $\zeta_n \cot(\zeta_n)$ term. This is not a problem for the plane wall range because it already includes 0 in the range. The range for the cylindrical eigenvalues was determined by experimentation. As Bi approaches 0, ζ_1 goes to 0 which gives a lower limit of $(n-1)\,\pi$. For the upper limit, ζ_1 was calculated for increasing Bi values. For Bi values greater than 10^7, ζ_1 converged to approximately 0.76548π. The eigenvalues for the cylinder equation were calculated for higher values of n to determine the period for the eigenvalues and if the range expanded or converged. The calculations, shown in Figure 1, converged towards a value of $(n-0.75)\,\pi$ as n went to infinity. Therefore, the range for the cylindrical eigenvalues is $(n-1)\,\pi \le \zeta_n \le (n-0.23452)\,\pi$ based on the maximum possible range of ζ_1. This was simplified to a range of $(n-1)\,\pi \le \zeta_n \le (n-0.2)\,\pi$ since it neither excludes any eigenvalues nor causes an overlap of the ranges for different values of n.

For interval halving, it is necessary to test the midpoint value, between the upper and lower limit, to determine if it is less than or greater than the ζ_n. If the midpoint value is less than the ζ_n, it replaces the lower limit and a new midpoint value is tested. If it is higher, then it replaces the upper limit. Since the ζ_n is the value where $f(\zeta_n)$ equals zero, determining if the midpoint value is high or low just involves comparing the sign of $f(x_m)$ of the midpoint value with the sign of $f(x_L)$ of one of the limits. If the signs match, then both points are on the same side of the desired ζ_n. The search is terminated when the percentage of change from the last value checked and the new value is less than the desired tolerance. The tolerance was set to $(1 - x_i / x_{i-1}) < 10^{-9}$. The functions for finding ζ_n were separate from the main functions for calculating the dimensionless temperature, which allowed these functions to be verified independently.

The main function for each shape would calculate C_n and the summations and call the eigenvalue function for each value of n. The calculation of C_n and the summations was performed with a for-loop to calculate a specified number of terms. The spherical equation had to be changed for $r^* = 0$, which caused $\sin(\zeta_n r^*) / (\zeta_n r^*)$ in the summation to include division by zero. The limit of $\sin(\zeta_n r^*) / (\zeta_n r^*)$ goes to 1 as r^* goes to 0, so $\sin(\zeta_n r^*) / (\zeta_n r^*)$ is replaced by 1 if r^* is 0. The functions began by calculating the first four terms of the summation. Since the summation can be approximated by the first term for $Fo > 0.2$, the main function would return the summation of these four terms for $Fo > 0.2$. If the value of $Fo \le 0.2$, additional terms need to be calculated to get a reasonably accurate value for θ^*. For the initial program, it was arbitrarily decided to calculate 25 more terms of the summation. If the difference due to these 25 terms was less than the tolerance of 10^{-9}, then the summation was terminated. Otherwise, additional groups of 25 terms are calculated until the difference falls below 10^{-9}. Since low Fourier numbers

would require a significant number of iterations to meet the criteria, an upper limit of 100,000 iterations was set for the initial testing of the program.

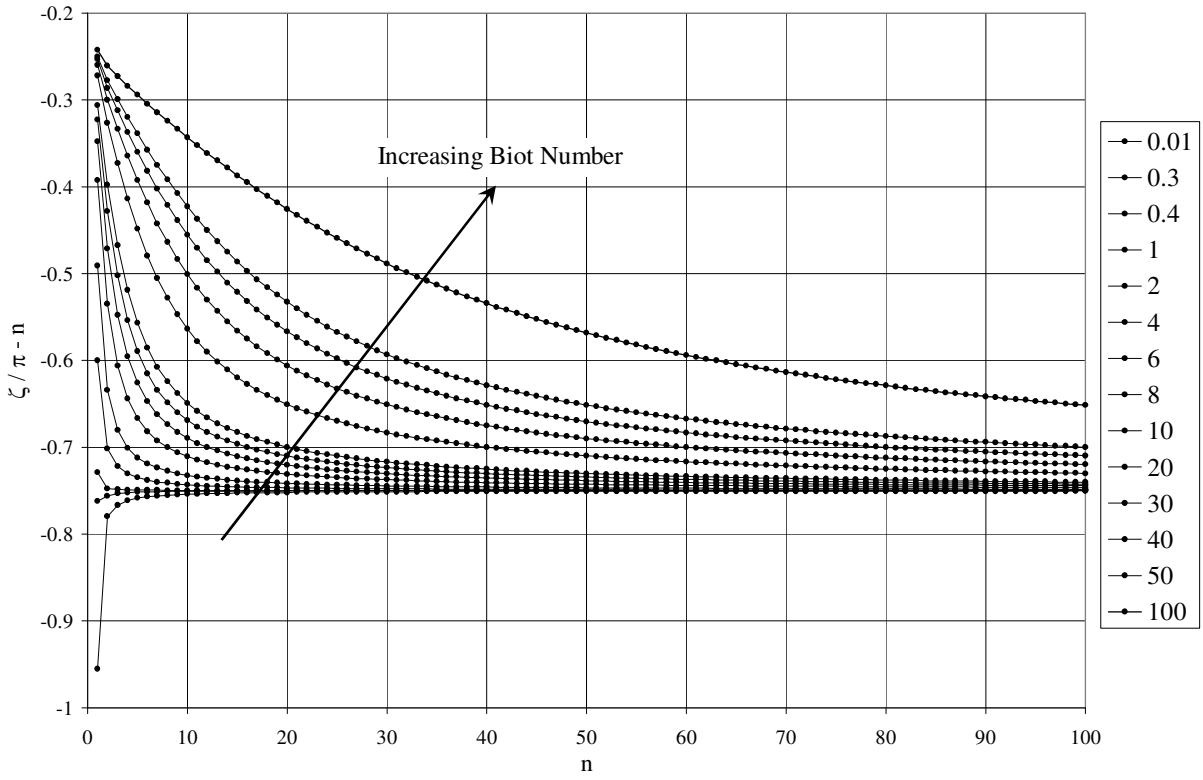

Figure 1. ζ_n/π-n for cylinder equation for $0.01 \leq Bi \leq 100$

Initial evaluation

The first verification of the Add-in was to confirm the eigenvalues found by the program. Schneider provides tables of the first 5 eigenvalues to 4 decimal places over a range of Biot numbers [2]. When rounded to 4 decimal places, the calculated eigenvalues for the plane wall and cylindrical equations were equal to Schneider's values. The majority of the spherical eigenvalues were equal when rounded; however 10 of the 190 eigenvalues disagreed on the final digit, but the calculated values were within ±0.0002% of Schneider's values. For additional verification, the calculated eigenvalues were put back into the equations for the eigenvalues to check for equality. The remaining differences for the plane wall and spherical equations were less than 10^{-11}. For the cylindrical eigenvalues, the remaining difference was less than 2×10^{-4}, but the differences decreased with decreases in the Biot number. The high remaining difference for the cylindrical eigenvalues was for a Biot number of 100, which would equate to a minor difference from the true eigenvalue due to a larger value for $f'(\zeta_n)$.

The dimensionless temperatures were compared to the values calculated with the Interactive Heat Transfer (IHT) [1] software to verify the final calculation of the temperatures. The IHT

software has built-in models for 1D conduction in a plane wall, cylinder and sphere; however they differ from the functions in this Add-in. Rather than returning the dimensionless temperature, the IHT models return the temperature based on the free-stream and initial temperatures that are input into the model, but it was possible to calculate the dimensionless temperature from Equation 1. For Fo > 0.00125, the results from the Add-in matched the 4 digit result from the IHT software. For Fo < 0.00125, there was an increasing amount of error between the two results. The results from the IHT software were greater than 1 for some r* < 1 and became larger as Fo decreased, while the results from the Add-in tended to converge to 1 more readily. While information about the calculations for IHT was not readily available, the likely explanation for this difference is that the IHT limits the calculations to a smaller number of iterations, which helps explain the agreement for the higher Fo values that require fewer iterations. Based on the comparisons of the eigenvalues and dimensionless temperature, the Excel Add-in provides reasonable results up to a minimum of 4 significant digits. The Add-in is set to return 8 significant digits but this level of accuracy has not been confirmed.

The next step is to improve the efficiency of the calculations. If the functions require a long time to calculate a result, it will lower its utility especially in generating tables and charts. A baseline of the run-time for each function is necessary in order to determine if subsequent changes in the functions are beneficial. The time trials were performed on a Dell Inspiron 2200 with an Intel® Pentium® M processor 1.60 GHz with 590 MHz, 1.24 GB of RAM. An initial test was performed to determine the time required to perform 100 calculations. Each set of calculations was performed for a specific Fo and x* or r*, but a range of Bi from 0.1 to 10 at a 0.1 intervals. The time calculation was repeated over a range of Fo from 0.00001 to 0.02 and x* (r*) from 0 to 1 at a 0.1 interval. From Figures 2 and 3, it can be seen that the dimensionless length had a minor effect on the run time compared to the Fo value, which is reasonable as the presence of Fo in the exponential term has a significant effect on the decrease in the value of the terms as the number of iterations increases. Excel's Timer function provided the start and stop times for each set of calculations. The drawback of the Timer function is that it measures "wall clock" time with a resolution of $1/64^{th}$ of a second. Other processes were turned off including the screensaver and power management processes in order to minimize the amount of error.

The result of the initial test showed that the cylinder function required a run-time approximately 100 times longer to finish the calculations compared to the plane wall and sphere calculations. The main difference in the functions is that the plane wall and spherical functions only use the built-in trigonometry function where the cylinder function uses Excel's BESSELJ(x, n) function from the Analysis Toolpak – VBA Add-in. The BESSELJ(x, n) function is a generalized function designed to calculate $J_n(x)$ for any value of n. For the cylinder function, only $J_0(x)$ and $J_1(x)$ are needed. The functions bessj0(x) and bessj1(x) were taken from *Numerical Recipes in C* [3] to try to improve the run-time and remove the need to include the Analysis ToolPak – VBA Add-in. The result was a 97% reduction in the average time required, though it still required about 4 times longer than the plane wall and sphere functions. For a baseline, the number of calculations was increased to 10000 for the plane wall and sphere functions to decrease the error due to the 0.01-second resolution of the Timer function. Only 2500 calculations were performed for the cylinder function to reduce the total run-time needed for the test and the functions are not being compared to each other for now. For each change to improve the efficiency, the dimensionless temperature returned by the functions was compared against the original value to ensure it had not changed.

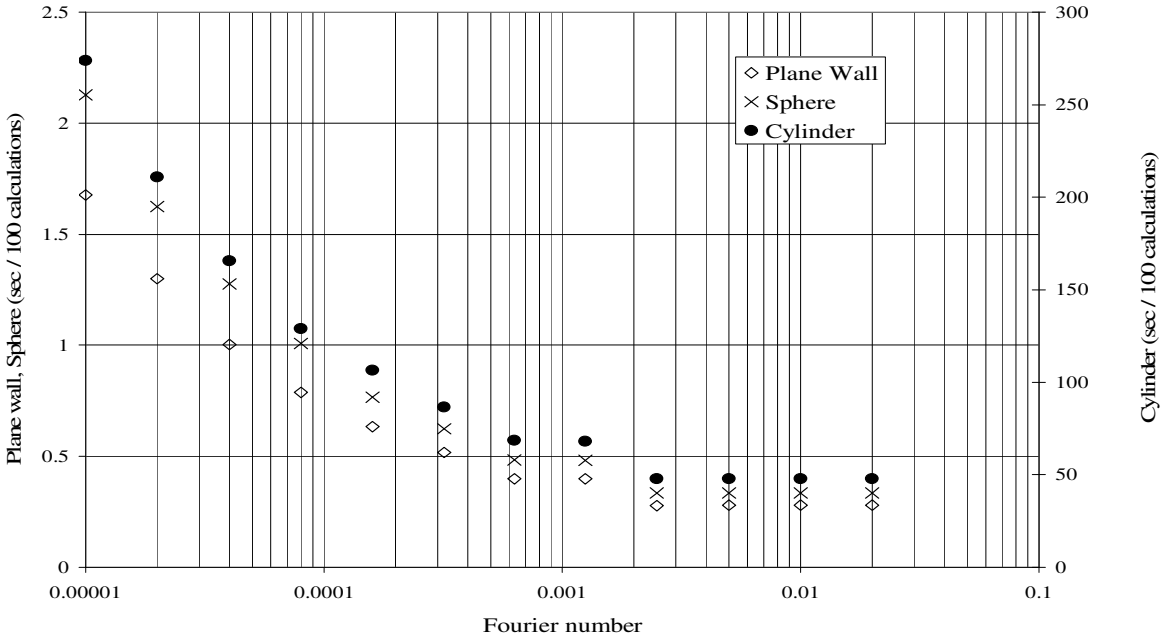

Figure 2. Time for 100 calculations v. Fourier number

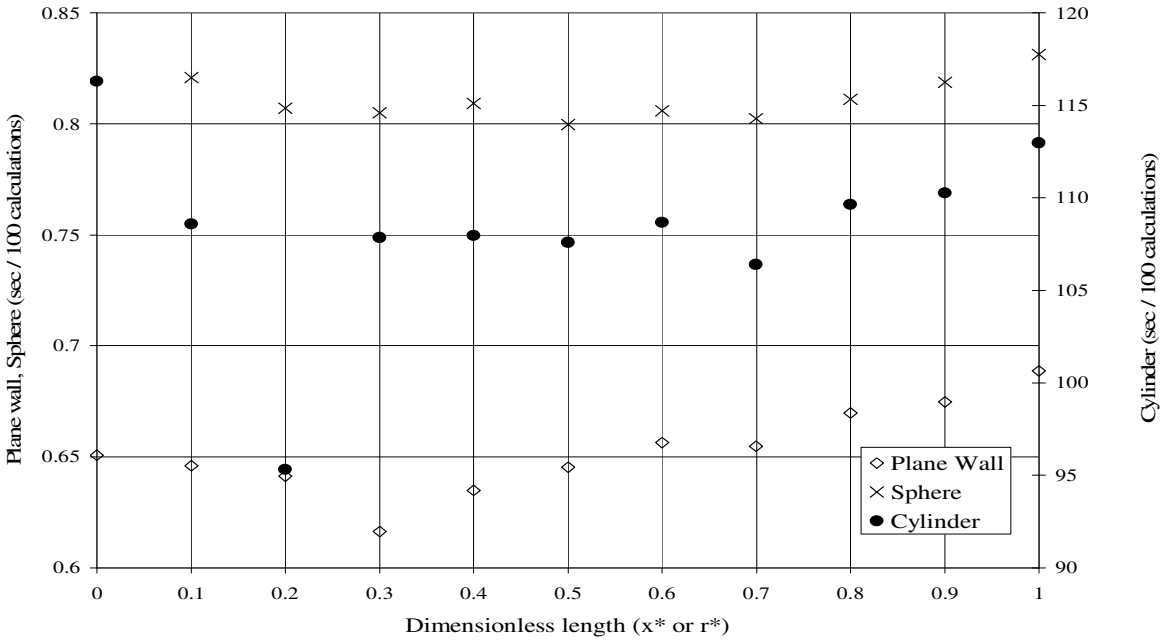

Figure 3. Time for 100 calculations v. dimensionless length

Improved efficiency

While interval halving proved to be an effective root-finding method, it is not the most efficient requiring $\log_2(N)$ iterations where N is the range being searched divided by the desired resolution. More efficient methods include the Newton-Rhapson method and the secant method. The secant method was tried first since the equations for the eigenvalues were already written. One problem that was encountered was that the equation for the spherical eigenvalues had a peak in the middle of the range. Using the midpoint of the range caused an incorrect value to be returned for some Biot numbers. Calculation of the sphere eigenvalues for different Biot numbers showed that the eigenvalues for Bi = 1 were $(n-½)\,\pi$. For Bi > 1, the eigenvalues were greater than $(n-½)\,\pi$ and less than $(n-½)\,\pi$ for Bi < 1. The range for the eigenvalues was changed to $(n-1)\,\pi \le \zeta_n \le (n-0.4)\,\pi$ for Bi < 1 and $(n-0.6)\,\pi \le \zeta_n \le n\,\pi$ for Bi > 1. The original attempt utilized the $(n-½)\,\pi$ value instead of $(n-0.4)\,\pi$ and $(n-0.6)\,\pi$ for the cut-off, but this caused errors to occur for Bi values near 1. The plane wall and sphere functions had a 55% reduction of their average run-time, while the cylinder function had a slightly better improvement with a 60% reduction in its average run-time. For the Newton-Rhapson method, it was necessary to determine the derivative of each of the functions. With only trigonometry functions, it was a simple, direct differentiation of the plane wall and sphere equations. The differentiation of the cylinder equation was a little more complicated with the inclusion of the Bessel function, but was accomplished through Laplace transforms. The derivatives were compared to values calculated with the secant method to check both methods. Both the plane wall and sphere functions had a 20% reduction in run-time with Newton-Rhapson method compared to the secant method. The cylinder function was slightly slower with the Newton-Rhapson method with a 5% increase in run-time. This is likely the result of the longer run-time for the Bessel functions compared to trigonometric functions. For the secant method, only one addition calculation using the Bessel function was necessary in order to determine the initial slope. The Newton-Rhapson method requires two calculations involving the Bessel function for each new value, one for $f(\zeta_n)$ and one for $f'(\zeta_n)$. The additional run-time for the second calculation offset any advantage of the Newton-Rhapson method for the cylinder function. The cylinder function utilizes the secant method due to the slightly better performance, where the Newton-Rhapson method is utilized by the plane wall and sphere functions.

The next change arose from the observation that as the index n increases, the spacing between ζ_n and ζ_{n+1} approaches π for all three functions. So far, the initial guess for the eigenvalue was the midpoint of the range to be searched. The midpoint is still the initial guess for the ζ_1, but a better starting point for subsequent ζ_n is $\zeta_{n-1} + \pi$. Both the plane wall and sphere had a 10% improvement in the run-time with a final average time of 21.5s and 25.4s per 10000 calculations respectively. The cylinder function had a 20% improvement with a final average run-time of 22.1s per 2500 calculations. The big advantage of this last improvement is the reduction in the run-time for low Fourier numbers since the spacing approaches π as n increases. For Fo = 0.00001, the cylinder function had a 30% improvement with the plane wall and sphere functions having a 20% and 15% improvement, respectively.

The final change to the functions was to change the maximum number of iterations, so each function had an equal maximum run-time. The plane wall function is the fastest of the functions and kept the 100,000 iteration limit. By setting the Fo = 0 to ensure the maximum number of

iterations, the three functions were timed over a range of Bi and x* (r*) values to get a comparison. Based on the average times, the sphere function was set to a maximum of 60,000 iterations and 20,000 iterations for the cylinder function. This did result in a slight change of 10^{-8} in some of the calculated dimensionless temperatures. However, since the functions have only been confirmed to 4 digits and a very low Fourier number is required to reach this limit, it was decided this change was acceptable in order to reduce the run-time.

Semi-Infinite Solid

The Semi-Infinite Solid function performs calculations for the three different boundary conditions of constant surface temperature, constant surface heat flux and convection. The function takes the form TsemiInf(BCtype, x, alpha, t, k, BC). The function could be separated into three separate functions, but it was decided to combine them into one function and include an input, BCType, to specify which boundary condition. This was decided because each boundary condition only requires a single formula and did not required extensive programming. BCType accepts a value of 1, 2, or 3 to specify constant surface temperature, constant surface heat flux or convection respectively. Care needs to be taken to understand the value being returned and the requirements of the inputs. The returned value is on the left-hand side of the equations below.

Constant Surface Temperature,

$$\frac{T(x,t)-T_s}{T_i-T_s}=erf\left(\frac{x}{2\sqrt{\alpha\cdot t}}\right) \tag{13}$$

Constant Surface Heat Flux,

$$T(x,t)-T_i=\frac{2\cdot q''_0\sqrt{\frac{\alpha\cdot t}{\pi}}}{k}\exp\left(\frac{-x^2}{4\cdot\alpha\cdot t}\right)-\frac{q''_0\cdot x}{k}erfc\left(\frac{x}{2\sqrt{\alpha\cdot t}}\right) \tag{14}$$

and convection boundary condition

$$\frac{T(x,t)-T_i}{T_\infty-T_i}=erfc\left(\frac{x}{2\sqrt{\alpha\cdot t}}\right)-\left[\exp\left(\frac{h\cdot x}{k}+\frac{h^2\cdot\alpha\cdot t}{k^2}\right)\right]\left[erfc\left(\frac{x}{2\sqrt{\alpha\cdot t}}+\frac{h\sqrt{\alpha\cdot t}}{k}\right)\right] \tag{15}$$

The three inputs that are consistent for all the boundary conditions are x, the distance from the surface; alpha, $\alpha = k / \rho c$; and t, time. Since these are not dimensionless numbers, it is important that the units of these inputs are consistent with the result in the proper calculations. It is recommended that standard units of m, m^2/s and s are used to avoid errors. It is possible to use other units provided they cancel properly or a correction is made to the answer to cancel out remaining factors. The conduction coefficient, k, is an optional input which is not necessary for the constant surface temperature formula but required for the other two formulas. A second optional input is BC which is the heat flux for constant heat flux formula and the convection coefficient, h, for the convection formula. The BC input is not necessary for the constant

temperature formula. There is a check for these inputs for the last two boundary conditions and a warning box will appear if they are left blank to notify the user of the mistake. It is recommended that the units of W/m-K, W/m^2, and W/m^2-K are used for k, q''_0 and h, respectively, to avoid problems with the units not canceling properly.

Equations 13, 14, and 15 show that the error function (erf) and complimentary error function (erfc) are needed for the calculations. These functions are available in the Analysis ToolPak - VBA Add-in and worked properly in Excel 2003 which was used to develop the module. However, problems were encountered when module was tried with Excel 2007, which appeared to be linked with using the functions from the Analysis ToolPak - VBA Add-in. It seemed that two versions of the module would be necessary to have one work with Excel 2003 and one with Excel 2007. Rather than having two modules, it was decided to replace the erf(x) and erfc(x) functions from the Analysis ToolPak - VBA Add-in with errf(x) and errfc(x) functions that were programmed according to functions available in *Numerical Recipes in C* [3]. In addition to allowing the same module to work with Excel 2003 and Excel 2007, the 1-D transient heat conduction module does not require any other Add-ins.

Nusselt Number Functions

A total of 13 different formulas were taken from *Introduction of Heat Transfer* [1] for both external and internal flow over different solid shapes. The formulas were grouped, where appropriate, to yield the 9 functions shown in Table 1. The Nusselt number calculations involve a single formula and it is possible for to simply type the formula into an Excel worksheet. However, some of the formulas are fairly long and these functions provide a simple solution that can help avoid mistakes. The formulas were grouped by shape and whether the local or average Nusselt number was calculated. Table 1 also shows the reference temperature for the inputs and the ranges for each formula. The reference temperature is the temperature utilized to determine the inputs such as Reynolds or Prandtl number. The ranges specified are the values over which the results of the formula have been determined to be valid. For each formula, the inputs are tested to see if they are within the range. If the values are outside the range, a warning message will notify the user, but the function will still calculate an answer based on the inputs. At times, engineers may need an approximate Nusselt number for values that are outside the range. It would be up to the user to decide if the result is sufficiently accurate for their purpose. The warning message appears anytime the function is calculated which can cause multiple warning messages to appear if a table is being created or a warning message to appear for the same values when the sheet is recalculated. An optional input, *Quiet*, is part of each function to allow the user to turn off the warning messages by setting it equal to true. By default, *Quiet* is set to false to provide the user with warnings.

The majority of the functions involve direct calculation of the formulas, but there are variations in the inputs and differences that are specific to each function. NuxPlate and NuBarPlate initially had the inputs of Re, Reynolds number; Pr, Prandtl number and Laminar, a Boolean input that is true for laminar flow or false for turbulent flow. The NuBarPlate function was based on a critical Reynolds number of 500,000 for the separation point between the laminar and turbulent flow. However, the NuBarPlate function was expanded to accommodate an optional *Rexc* input for different critical Reynolds numbers. This input has a default value of 500,000 and makes the Laminar input redundant. The *Rexc* input replaced the *Laminar* input in both

functions to maintain consistency although the specific value does not affect the calculation for NuxPlate beyond determining if the point is in the laminar or turbulent region.

For crossflow over a cylinder and turbulent flow through a circular tube, the only inputs required are Re, Reynolds number and Pr, Prandtl number. There is one additional input, mu_mus, for flow over a sphere. The input is equal to μ, the viscosity at the free-stream temperature, over μ_s, the viscosity at the surface temperature. Likewise, NuDLiqMetals has one additional input, *UniformT*, with Re and Pr. The *UniformT* input is an optional Boolean input that is true for uniform surface temperature or false to specify uniform heat flux.

Table 1. Nusselt number functions

Function	Description	Reference Temperature	Ranges
NuxPlate	Calculates local Nusselt number for flow over a flat plate	Film Temperature	Laminar flow $Pr \geq 0.6$ Turbulent flow $Rex \leq 10^8$ $0.6 \leq Pr \leq 60$
NuBarPlate	Calculates average Nusselt number for flow over a flat plate	Film Temperature	Laminar flow, $Pr \geq 0.6$ Mixed flow $Rex \leq 10^8$ $0.6 \leq Pr \leq 60$
NuDBarCyl	Calculates average Nusselt number for crossflow over a cylinder	Film Temperature	$Re_D*Pr \geq 0.2$
NuDBarSphere	Calculates average Nusselt number for flow over a sphere	Free-stream Temperature	$0.71 \leq Pr \leq 380$ $3.5 \leq ReD \leq 7.6E4$
NuDBarTubes	Calculates average Nusselt number for crossflow over a bank of tubes	Mean Film Temperature	$Pr \geq 0.7$ $2E3 \leq Re_{D,\,max} \leq 4E4$
NuDBarZTubes	Calculates average Nusselt number for crossflow over a bank of tubes with correlation developed by Zukauskas	Average of entrance and exit temperature	$0.7 \leq Pr \leq 500$ $1000 \leq Re_{D,\,max} \leq 2E6$
NuDBarLamTube	Calculates average Nusselt number for laminar flow through a circular tube	Mean of inlet and outlet temperature	Thermal entry or combined entry > 5 Uniform T_s Combined entry < 5 Uniform T_s $Pr \geq 0.6$ $0.0044 \leq (\mu/\mu_s) \leq 9.75$
NuDTurbTube	Calculates Nusselt number for turbulent flow through a circular tube	Mean of inlet and outlet temperature	$0.5 \leq Pr \leq 2000$ $3000 \leq Re_D \leq 5E6$ $L/D \geq 10$
NuDLiqMetals	Calculates Nusselt number for liquid metal flowing through a circular tube	Mean of inlet and outlet temperature	Uniform surface temperature $Pe_D = Re_D*Pr \geq 100$ Uniform surface heat flux $3.6E3 \leq Re_D \leq 9.05E5$ $10^2 \leq Pe_D \leq 10^4$

NuDBarLamTube requires three additional inputs to calculate the average Nusselt number for laminar flow through a circular tube. The first is D_L which is the dimensionless ratio of the inside diameter over the length. The second two are the optional inputs, *Thermal* and *mu_mus*. The *Thermal* input is a Boolean input that is true for calculation based on thermal entry length. This means the velocity profile is fully developed prior to the heat transfer section of the tube. False for the *Thermal* input length specifies a combined entry length problem where both the temperature and velocity profile are developing simultaneously. There is one additional check involved since the combined entry length problem for high Prandtl numbers, $Pr > 5$, is calculated with the same formula for the thermal entry length problem. If $Pr > 5$, the calculation is performed with the formula for the thermal entry length regardless of the value set for *Thermal*. The *mu_mus* input is the same viscosity ratio described before but is only necessary for the combined entry length problem with $Pr \leq 5$. The default value for *mu_mus* is 0 which will return an answer of 0, but there is a check for the *mu_mus* range that will warn the user if *Quiet* is not set to true.

The two functions for the banks of tubes involve additional inputs as well as constants that need to be obtained from tables. NuDBarTubes function is based on a traditional correlation, where the NuDBarZTubes function is based on a newer correlation developed by Zukauskas [1]. The NuDBarTubes function requires the inputs St_D and Sl_D and NuDBarZTubes requires the inputs Prs and St_Sl. St_D is the transverse spacing, perpendicular to the flow, divided by the tube diameter. Sl_D is the longitudinal spacing, parallel to the flow, divided by the tube diameter. St_Sl is the ratio of transverse spacing to the longitudinal spacing. Prs is the Prandtl number based on the average of the inlet and outlet temperatures where the other inputs are based on the film temperature. Both functions have the optional inputs *Aligned* and *Nl*. The *Aligned* input is true by default specifying aligned tube geometry. Setting *Aligned* to false specifies a staggered geometry. The *Nl* input is the number of rows along the path of the flow. It allows for correction for a low number of rows. *Nl* is 10 by default for NuDBarTubes, 20 for NuDBarZTubes and is only necessary if the number of rows is less than these values. The inputs St_D, Sl_D, *Aligned*, and *Nl* determine constants from associated tables. The *Aligned* input is used by an If-Then statement in the first part of the table lookup. The constants for NuDBarTubes are interpolated based on the inputs St_D and Sl_D. A Select Case statement selects two arrays based on the value of St_D with another Select Case statement to pick the specific elements used in the interpolation. For both functions, an If-Then statement with *Aligned* determines the array for the correction coefficient for the number of rows with the *Nl* input used to select the specific value.

View Factor Functions

The three view factor functions include two aligned parallel plates that are the same size, two coaxial disks that can have different diameters, and two perpendicular plates that share a common edge. The only requirement for the inputs is that they have the same units of length. The FijAlignPlates function for the two aligned parallel plates requires three inputs, X, Y, and L. X and Y specify the dimensions of the plates and L is the distance between the plates. The calculation of Fij is long, so it was broken down into smaller steps to avoid errors. Each step was checked against a separate calculation. Once the formula was entered, it was combined into a single calculation to simplify the code.

The FijAlignDisks function for the coaxial disks has three inputs, ri, rj, L. The inputs ri and rj are the radii of the source and intercepting disks, respectively and L is the distance between the disks similar to the FijAlignPlates function. Intermediate calculations are required to determine coefficients needed for the calculation.

The FijPerpPlates function for the perpendicular plates requires three inputs of X, Y, Z. X is the length of the common edge. The Y and Z inputs are the length of the remaining edge of the source and intercepting plate respectively.

There is a graph of each function available in *Introduction to Heat Transfer* [1]. These graphs were compared with graphs, shown in Figures 4, 5, and 6, generated in Excel using the different functions. The functions were able to generate similar graphs. The comparison of the graphs, along with hand calculation of the formulas, verified the calculations performed by the functions.

Similar to the Nusselt number functions, the view factor functions only involve a single calculation. However, by creating functions to perform these calculations, it reduces the possibility of errors and saves time by having a verified function available.

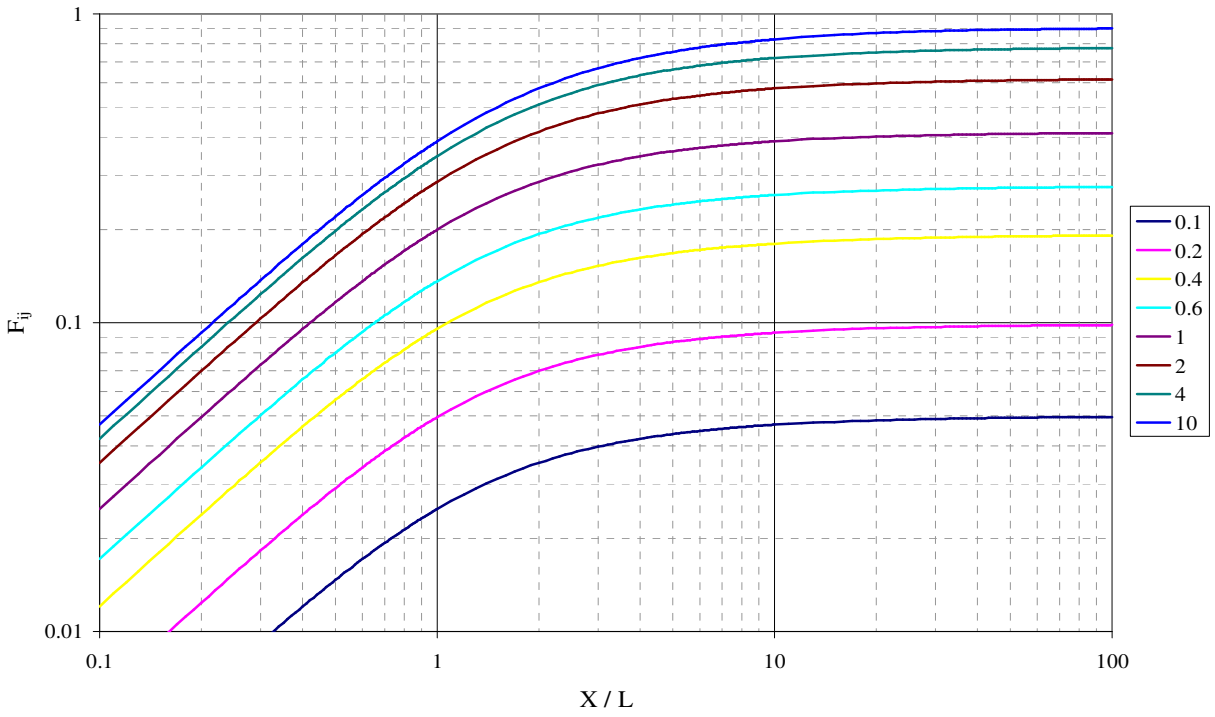

Figure 4. View factor, F_{ij}, v. X / L for different values of Y / L

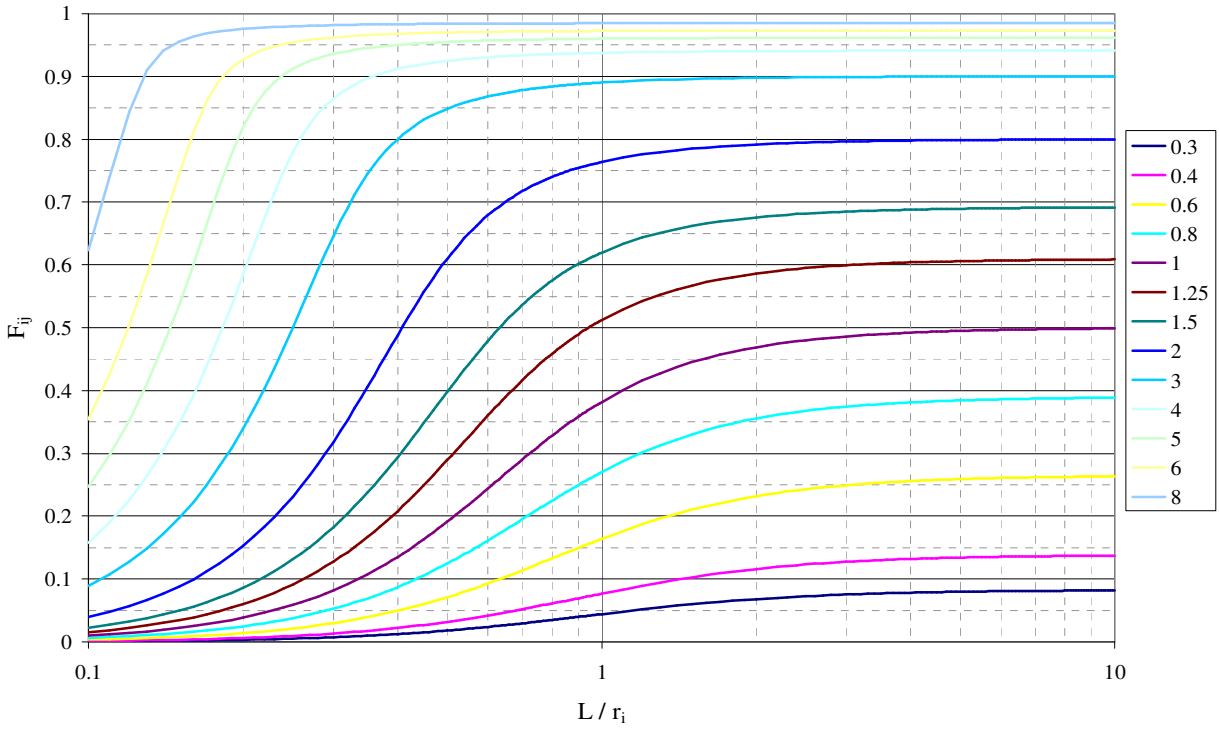

Figure 5. View factor, F_{ij}, v. L / r_i for different values of r_j / L

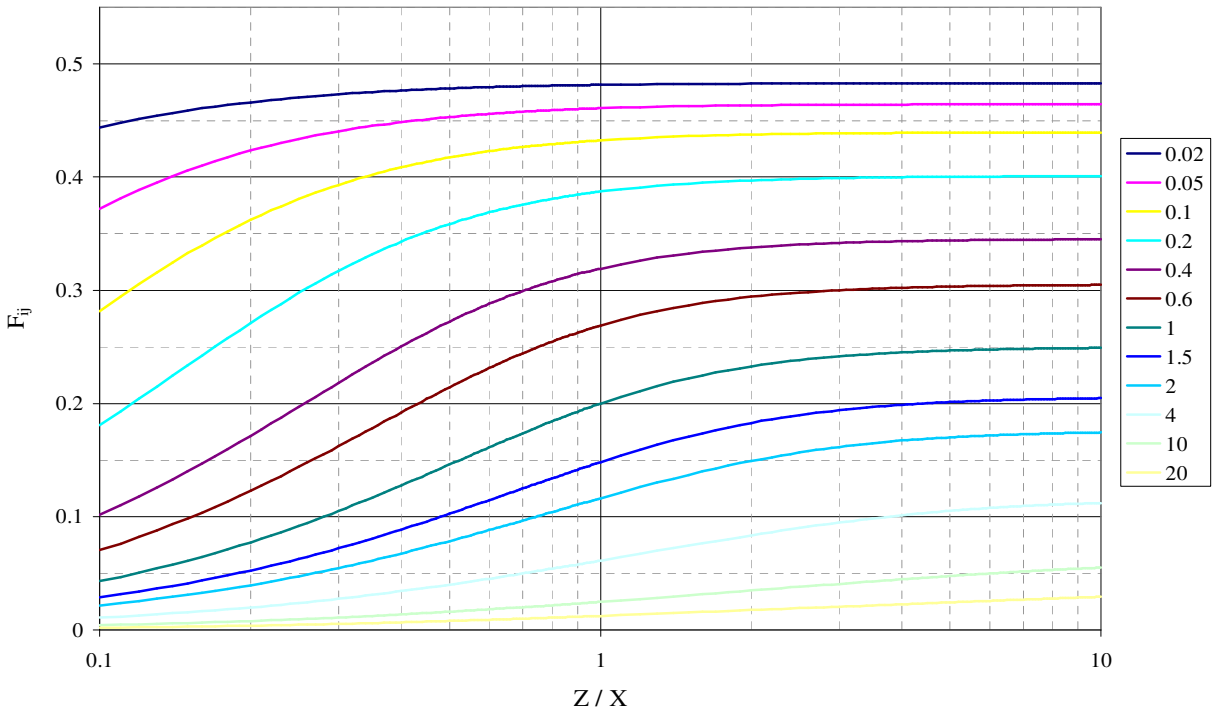

Figure 6. View factor, F_{ij}, v. Z / X for different values of Y / X

Examples

The Add-ins provided a teaching aid for undergraduate heat transfer courses. For 1-D transient heat conduction without the Add-in, students are only able to look up the constant for the first term of the summation from a table. This provides an approximate dimensionless temperature for Fo > 0.2. For smaller Fo, charts are provided, but the students can only estimate the dimensionless temperature based on Bi, Fo and the dimensionless length. By providing these functions, more complicated examples and problems can be solved. One example was determining the time required for the center of a slab to reach a given temperature with equal convection on both sides. The Biot number (Bi) is calculated from information given in the problem, xstar is 0 for the center of the slab and a dimensionless temperature, θ_G, is calculated from the given temperature. The student is then able to use Tslab(xstar, Fo, Bi) along with either GoalSeek or Solver in Excel to determine the Fourier number for Tslab = θ_G or Tslab - θ_G = 0. From the Fourier number, the student can calculate the time.

It is also possible to demonstrate to the students how the 1D transient heat conduction formulas can solve other geometries. The Tslab function can solve for a 2D bar or 3D cube by multiplying the results for the heat transfer from each surface. Figure 7 shows the intersection of 2 plane walls to form an infinite 2D bar. The dimensionless temperature can be calculated for the point shown from the product of Tslab for each Plane Wall.

$$\theta = Tslab(xstar_1, Fo, Bi_1) \cdot Tslab(xstar_2, Fo, Bi_2) \tag{16}$$

The same calculation can be performed for a box by adding a 3rd plane wall that is perpendicular to the first two or a finite cylinder from the intersection of an infinite cylinder and plane wall.

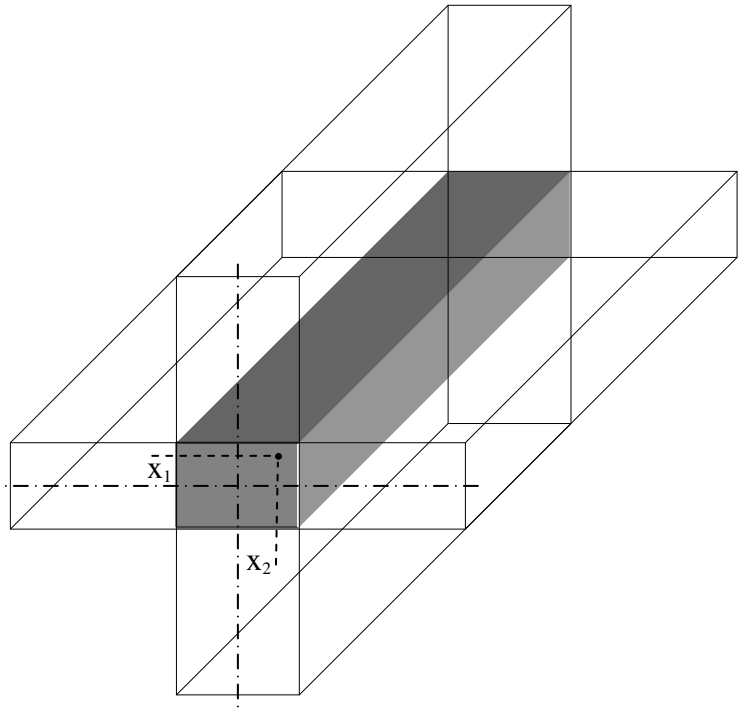

Figure 7. Intersection of 2 plane walls to form a 2D bar

Classroom Experiences

These Excel Add-in functions and the Excel problem solving format has been introduced into the junior-level course in heat transfer, and the follow-on course, Energy Systems Design. The students took easily to the Excel format. This is not surprising since Excel is used widely both in the high schools and in the University. About one-third of the homework problems required the Excel format and the other two-thirds were open for the students to choose the format. About half the time for these open assignments, the students would choose to use the Excel format. Two or three students would always choose to use paper and pencil, and three or four would always choose to use the Excel format. The other twenty would choose a mixture of paper and pencil and Excel format.

When using the Excel Add-in functions the students are not blindly plugging data into boxes on the spreadsheet and getting answers highlighted in green at the bottom. The students start with a blank spreadsheet. They still have to solve the problems. They have to make the same decisions and follow the same reasoning paths as when using paper and pencil. The Add-in functions relieve the tedium of evaluating complicated functions or replace graphical or tabular lookups.

The students had low resistance to adopting the already familiar Excel for heat transfer problem solving. The typical student could attack a larger number of problems within the time limits of a traditional three-hour junior course. He or she appeared more confident to attack problems that involve tedious calculations. Numerical accuracy was improved. The students still have to solve the problems; so, problem solving skills improved through the opportunity to attack a larger number of problems.

Conclusion

By designing the functions step-by-step, it was possible to verify that each part of the function worked properly rather than attempting to debug the functions after they had been completed. These functions created for this Excel Add-in provide a useful tool as both a teaching tool and engineering tool. While there are more sophisticated software packages available, they are neither as prevalent nor as widely used as Microsoft Excel. By creating an Engineering Add-in with functions that perform the more common calculations in engineering, it is possible for the students to solve more complicated problems for a more in-depth learning experience as well as provide them with a useful tool for their engineering career after graduation.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. DUE-0633330. The authors gratefully acknowledge support from this NSF award.

Troy Dent was supported in part by a fellowship from a grant under the U.S. Department of Education's Graduate Assistance in Areas of National Need (GAANN) program. A final thank you goes to the undergraduate students in the ME 309, Heat Transfer course and ME 415, Energy Systems Design course for their assistance in evaluating the usefulness and usability of the Add-in.

Disclaimer

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the U.S. Department of Education.

Bibliography

1. Incopera, F. P., DeWitt, D. P., Bergman, T. L., Lavine, A. S., *Introduction to Heat Transfer*, John Wiley & Sons, Inc., Hoboken, NJ, 2007.
2. Schneider, P. J., *Conduction Heat Transfer*, Addison-Wesley Publishing Company, Inc., Reading, MA, 1974.
3. Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T., *Numerical Recipes in C*, Cambridge University Press, New York, 2002.