Lab 5: Post Processing and Solving Conduction Problems

**Objective:**

The objective of this lab is to use the tools we have developed in MATLAB and SolidWorks to solve conduction heat transfer problems that a thermal analyst might encounter. This will include additional post-processing of your data.

**Background:**

A typical calculation that is part of post-processing a numerical solution is calculating the heat flux at a boundary. From Fourier’s law, the heat flux in the $y$-direction is defined as:

$$q_y^* = -k \frac{\partial T}{\partial y}_{\text{boundary}}$$  \hspace{1cm} (1)

where $k$ is thermal conductivity and the spatial derivative for temperature must be approximated numerically from your discrete numerical results.

For the finite difference method, the heat flux can be approximated using either the *Taylor Series* method or the *control volume approach*. The latter will account for any internal heat generation within the element. In either case, one-sided finite difference approximations will be needed because of the presence of the boundary. Note that the finite difference approximations should have the same order of accuracy as the internal nodes (typically 2nd order). The appropriate 2nd order accurate forward difference approximation derived using the Taylor Series approach for a node located at the $y = 0$ boundary (or for $j = 1$) as shown in Figure 1 is

$$\frac{\partial T}{\partial y}_{i,j} \approx \frac{-3T_{i,1} + 4T_{i,2} - T_{i,3}}{2 dy}. \hspace{1cm} (2)$$

![Figure 1. Schematic diagram of nodal numbering for heat flux boundary condition.](image-url)
For the finite element method, the heat flux can be calculated from the numerical solution for the temperature distribution as follows:

\[ q^* = -(k \nabla \tilde{T}) \cdot \hat{n} = - \left[ k \nabla \left( \sum_{i=1}^{nne} p^{(e)}_i T^{(e)}_i \right) \right] \cdot \hat{n} \]  

(3)

where \( \hat{n} \) is the boundary outward unit normal, \( \tilde{T} \) is the approximate solution for temperature, \( T_i \) is the temperature at the \( i \)th node, \( p^{(e)}_i \) is the interpolation function at the \( i \)th node, and \( nne \) is the number of nodes in an element. For 3-node triangles where the temperature varies linearly over the triangle the interpolation functions are defined as:

\[
p^{(e)}_i = \frac{1}{2\Delta} \left( a_i + b_i x + c_i y \right), \quad a_i = x_j y_k - x_k y_j, \quad b_i = y_j - y_k, \quad c_i = x_k - x_j \\
p^{(e)}_j = \frac{1}{2\Delta} \left( a_j + b_j x + c_j y \right), \quad a_j = x_k y_i - x_i y_k, \quad b_j = y_k - y_i, \quad c_j = x_i - x_k \\
p^{(e)}_k = \frac{1}{2\Delta} \left( a_k + b_k x + c_k y \right), \quad a_k = x_i y_j - x_j y_i, \quad b_k = y_i - y_j, \quad c_k = x_j - x_i
\]  

(4)

where \( \Delta = \frac{1}{2} \left( x_j y_k + x_k y_i - x_i y_j - x_j y_i - x_k y_j + x_i y_k \right) \) is the area of the triangle. For heat flux in the \( y \)-direction through a boundary with \( \hat{n} = -\hat{j} \) this becomes:

\[
q^*_y = k \left( \frac{\partial p^{(e)}_i}{\partial y} T^{(e)}_i + \frac{\partial p^{(e)}_j}{\partial y} T^{(e)}_j + \frac{\partial p^{(e)}_k}{\partial y} T^{(e)}_k \right) = \frac{k}{2\Delta} \left( c_i T^{(e)}_i + c_j T^{(e)}_j + c_k T^{(e)}_k \right)
\]  

(5)

For the PDE Toolbox in MATLAB there are several functions that can be used to obtain temperatures, gradients, and heat fluxes. For data exported from the PDE Modeler App, you can use the following where \( p \) is the point matrix, \( t \) is the triangle matrix, and \( u \) is solution vector:

```matlab
interp = tri2grid(p,t,u,x,y);
[gradux, graduy] = pdegrad(p,t,u);
```

For the `tri2grid()` function, `interp` is the interpolated temperature evaluated at the locations specified by \( x \) and \( y \). For the `pdegrad()` function, `gradux` and `graduy` are the gradients of \( u \) with respect to \( x \) and \( y \) evaluated at the center of each element.

Alternatively, for command line implementation of the PDE Toolbox, you can use the following where `thermalresults` is the `object` that contains all the solution information for your heat transfer PDE problem (in particular the geometry, mesh, properties, and solution):

```matlab
Tinterp = evaluateTemperature(thermalresults,x,y);
[gradTx, gradTy] = evaluateTemperatureGradient(thermalresults,x,y);
```
\[ [qx, qy] = \text{evaluateHeatFlux(thermalresults, } x, y) \];

\[ Qn = \text{evaluateHeatRate(thermalresults, RegionType, RegionID)} \];

For the `evaluateTemperature()` function, \( T_{\text{interp}} \) is the interpolated temperature evaluated at the locations specified by \( x \) and \( y \). For the `evaluateTemperatureGradient()` function, \( \text{grad}Tx \) and \( \text{grad}Ty \) are the gradients of temperature in each direction evaluated at the locations specified by \( x \) and \( y \). For the `evaluateHeatFlux()` function, \( qx \) and \( qy \) are the heat fluxes in each direction at the locations specified by \( x \) and \( y \). For the `evaluateHeatRate()` function, \( Qn \) is the total heat rate through either an edge or face as specified by `RegionType` and `RegionID`.

Finally, to calculate total heat transfer rate, integrate Equation (1) over the boundary area.

\[
q_y = \int q_y' \, dA = -\int k \frac{\partial T}{\partial y} \bigg|_{\text{boundary}} \, dA \quad (6)
\]

For discrete data this must be numerically integrated. The simplest method is to replace the integral with a summation. For the bottom boundary and for our symmetric 2-D fin

\[
q_y = -2k \left( \frac{\partial T}{\partial y} \bigg|_{y=0} \frac{dx}{2} + \sum_{i=2}^{N_x} \frac{\partial T}{\partial y} \bigg|_{y=y_i} \frac{dx}{2} + \frac{\partial T}{\partial y} \bigg|_{y=N_y} \frac{dx}{2} \right)w. \quad (7)
\]
Assignment: Do NOT include any code with your assignment.

1. We have studied the temperature distribution in a straight fin for 3 thermal conductivities, $k$, using the finite difference method and the finite element method (Labs 3 and 4, Part 3). We will now use post-processing of the temperature to calculate heat transfer rates.

(a) Calculate the fin heat transfer per unit width, $q' = q/w$, for $k = 5$, 50, and 200 W/m•K using the finite difference method. Recall that for an accurate solution the total error (TE), which is the sum of the discretization error (DE), iteration error (IE), and round-off error (RO), must be minimized. This requires the following: (1) minimizing DE by using a mesh that is sufficiently refined, (2) minimizing IE by using a tolerance for the convergence test for the Gauss Seidel iteration loop that is sufficiently low, and (3) minimizing RO by using high enough precision. For these calculations, we will use a very low iteration loop tolerance ($10^{-9}$) and double-precision (the default for MATLAB floating point calculations). Thus, DE for this case will be much greater than IE and RO. To check that DE decreases with mesh size, calculate $q'$ on a series of meshes ($N_x = 5, 10, 20$) and use Richardson Extrapolation \(^1\) to estimate the apparent order, extrapolated value for heat flux, and the extrapolated relative error. Make a plot of $q'$ versus $N_x$ for each case. Note that the number of nodes needed for convergence for $q'$ is typically more than that required for temperature to converge. For $k = 50$ W/m•K you should get an apparent order of approximately 2 and an extrapolated $q'$ of 4,353 W/m.

(b) Compare the two-dimensional numerical results to calculations made using a simple model that assumes one-dimensional conduction in the $y$-direction. The exact analytical solution to the model is given in Chapter 3 of Bergman, et al.\(^2\). Our conditions correspond to Case A in Table 3.4 for fins of uniform cross section. Equation (3.72) in this table can be used to solve for the fin heat transfer rate:

$$
q_f = M \frac{\sinh m L_y + (h/m k) \cosh m L_y}{\cosh m L_y + (h/m k) \sinh m L_y}
$$

$$
m = \sqrt{\frac{h P}{k A_c}} = \sqrt{\frac{h (2 w)}{k (2 L_x w)}} = \sqrt{\frac{h}{k L_x}}
$$

$$
M = \sqrt{h P k A_c (T_b - T_x)} = \sqrt{h (2 w) k (2 L_x w) (T_b - T_x)} = 2 w \sqrt{h k L_x (T_b - T_x)}
$$


For \( k = 50 \text{ W/m} \cdot \text{K} \):

\[
m = \sqrt{\frac{h}{k L_x}} = \sqrt{\frac{500 \text{ W/m}^2 \cdot \text{K}}{50 \text{ W/m} \cdot \text{K} \times 0.02 \text{ m}}} = 22.36 / \text{m}
\]

\[
M/w = 2 \sqrt{h k L_x} (T_b - T_\infty) = 2 \sqrt{500 \text{ W/m}^2 \cdot \text{K} \times 50 \text{ W/m} \cdot \text{K} \times 0.02 \text{ m} (200 - 100) \text{ K}}
\]

\[
M/w = 4472.5 \text{ W/m}
\]

Substituting into (3.72) for heat flux per unit width:

\[
q'_f = q_f = \frac{M}{w} \frac{\sinh [22.36 / \text{m} (0.2 \text{ m})] + \frac{500 \text{ W/m}^2 \cdot \text{K}}{22.36 / \text{m} \times 50 \text{ W/m} \cdot \text{K}} \cosh [22.36 / \text{m} (0.2 \text{ m})]}{\cosh [22.36 / \text{m} (0.2 \text{ m})] + \frac{500 \text{ W/m}^2 \cdot \text{K}}{22.36 / \text{m} \times 50 \text{ W/m} \cdot \text{K}} \sinh [22.36 / \text{m} (0.2 \text{ m})]}
\]

\[
q' = 4472.5 \text{ W/m} \times 0.9999 = 4471 \text{ W/m}
\]

Repeat these calculations for \( k = 5 \text{ W/m} \cdot \text{K} \) and \( k = 200 \text{ W/m} \cdot \text{K} \) for your comparison.

(c) Calculate the Biot number for each case defined as follows:

\[
Bi = \frac{h L_x}{k}
\]

where \( L_x \) is the characteristic length for conduction heat transfer in the \( x \)-direction. Note that this is NOT the computational Biot number that was based on mesh spacing. For each \( k \) value show your results for \( q' \) from Parts (a) and (b), the percent difference between these results, and the Biot number in one table. Discuss the physical significance of the Biot number and the range of Biot numbers for which assuming one-dimensional conduction in the fin is reasonable.
2. Simplified schematics for two micro-electronics cooling configurations are shown in Figures 2 and 3. The top surface is cooled convectively while the remaining surfaces are all well insulated from the surroundings. For steady-state operation, the electric power dissipation results in a uniform volumetric heating rate of \(10^7\) W/m\(^3\). The required cooling rate is determined by the maximum allowable chip temperature of 85 °C based on industry standards.

![Figure 2](image1.png)

**Figure 2.** Schematic of a chip array embedded in a substrate to be modeled as two-dimensional heat transfer.

![Figure 3](image2.png)

**Figure 3.** Schematic of single chip embedded in a substrate to be modeled as three-dimensional heat transfer.

![Figure 4](image3.png)

**Figure 4.** Cross-section of a chip and substrate for both the two-dimensional and three-dimensional cases.

(a) The representation shown in Figure 2 can be approximated as two-dimensional heat transfer by assuming a very long row of chips, thus neglecting end effects. Make a temperature contour plot for the chip and substrate using the PDE toolbox based on the geometry and specified operating conditions in Figure 4. Make sure to take advantage of thermal symmetry when defining your geometry. For command line implementation, you now need to create two rectangles, R1 and R2, and use the set formula, \(s_{\text{f}} = R_1 + R_2\), to add the shapes together. Next, plot out the geometry with the options ‘EdgeLabels’ set to ‘on’ and ‘FaceLabels’ set to ‘on’. The edge labels are used to set the boundary conditions correctly. The face labels are used to set the conductivity for each face and to only include internal heat generation in the chip using...
internalHeatSource(thermalmodel,qdot,’Face’,RegionID);

where \( qdot \) is the value for the internal heat source and the \( \text{RegionID} \) is the number for the face that corresponds to the chip.

Verify that the maximum chip temperature does not exceed the maximum allowable value. Discuss if the temperatures shown in your contour plot are consistent with the imposed boundary conditions and volumetric heating.

(b) Determine the minimum convection coefficient possible for the coolant to meet industry standards. Does this convection coefficient correspond to natural and/or forced convection with a gas and/or liquid (see Chapter 1 of Introduction to Heat Transfer by Bergman, et al.)?

(c) Make a similar temperature contour plot for the chip and substrate using SolidWorks Simulation for the two-dimensional case shown in Figure 2 and verify that you can get the same results as for Part (a). Next, make a new temperature contour plot for the three-dimensional case shown in Figure 3 for a square chip and a square substrate based on the same geometry and operating conditions in Figure 4. Discuss if the changes in your calculated temperatures between the two-dimensional and three-dimensional cases make sense.

Notes for SolidWorks Simulation:

1. To take advantage of symmetry, use a \( \frac{1}{4} \) of the domain shown in Figure 3. Begin by making separate \( \text{Part} \) drawings for the substrate and chip with the material properties specified for each one separately. For the substrate, make the bottom left hand corner at the origin and make an \( \text{Extruded Cut} \) for the \( \frac{1}{4} \) chip above the origin. For the chip, make the bottom left hand corner offset \( \frac{3}{4} H \) upwards from the origin. Next make an “Assembly” by inserting the two parts at the origin so they will be fixed at the correct relative locations with the \( x-y \) coordinates defined such that it will be easy to compare these results with your MATLAB results. (Alternatively, you can properly align the two parts by defining \( \text{Mates} \) and then move the origin.)

2. For an \( \text{Assembly} \) with multiple \( \text{Parts} \), in the \( \text{Simulation Tree} \) under \( \text{Connections} \) you must define the contact between components to control how heat is conducted across interior boundaries (for our case this corresponds to the faces between the substrate and chip). By default under \( \text{Connections} \) and \( \text{Component Contacts} \) there will be a \( \text{Global Contact} \) (corresponding to the entire \( \text{Assembly} \)) that is set to \( \text{Bonded} \) (corresponding to no thermal resistance). If you right click on this you can change it to \( \text{Insulated} \) (corresponding to infinite thermal resistance). For this assignment you will just use the default \( \text{Bonded} \) connections. For future work, if you want to set a thermal resistance for three-dimensional cases, in the \( \text{Simulation Tree} \) right click on \( \text{Connections} \) and select \( \text{Contact Sets} \) from the pull-down menu to open the \( \text{Connections} \) window and enter the desired settings.

3. To specify the uniform internal heat generation in the chip, in the \( \text{Simulation Tree} \) right click on \( \text{Thermal Loads} \) and select \( \text{Heat Power} \) from the pull-down menu. This corresponds to the volumetric heating rate above multiplied by the chip volume. For the two-dimensional case the volume is calculated using the width that was set for the two-dimensional simplification.
Notes for PDE Toolbox:

Alternatively, you can perform the calculations in 3D using the PDE Toolbox, but for this particular geometry MATLAB does not allow you to create separate cells for the chip and the substrate. Thus, you need to use the following “work around” for this problem. First, create a single cube for both the chip and substrate using

```matlab
  gm = multicuboid(W,D,H);  % create a cube W x D x H
  thermalmodel.Geometry = gm;  % add geometry to your model object
```

Then, use a function to set the properties to different values based on location such as

```matlab
  function kt = k_fun(location,state)  % Note, you can use a tilde for state
    global kc ks D W H d w h
    xmax = D/2; xlim = (D-d)/2;
    xc = ceil((location.x - xlim)/xmax);
    ymax = W/2; ylim = (W-w)/2;
    yc = ceil((location.y - ylim)/ymax);
    zmax = H; zlim = H-h;
    zc = ceil((location.z - zlim)/zmax);
    kt = xc.*yc.*zc.*(kc - ks) + ks;
  end
```

which sets the conductivity \( kt \) to either \( kc \) or \( ks \) based on location, where \( D, W, \) and \( H \) are the overall dimensions for the substrate, and \( d, w, \) and \( h \) are the dimensions for the chip.

(d) For most engineering problems you should use simple analytical models to insure that your simulation results have the right order of magnitude. For this case, use a simple one-dimensional analytical model of the system to estimate the maximum chip temperature (at base where it is in contact with the substrate) to insure that your answer to part (a) is reasonable. For your model list all of your assumptions (for example, you can assume heat loss is only from the chip surface and not through the substrate which is required to make the problem one-dimensional) and show all of your work.
3. An automobile door panel is fabricated by a plastic hot-extrusion process resulting in the ribbed cross-section shown in Figure 5. Following a process involving air-cooling, painting, and baking, the panel is ready for assembly on a vehicle. However, upon visual inspection, the rib pattern is evident on the outer surface. In regions over the ribs, the paint has an “orange peel” appearance, making the door panel unacceptable for use. The apparent reason for this defect is differential cooling rates at the panel surface that affects adherence of the paint.

![Figure 5. Schematic of ribbed cross-section of an automotive door panel.](image)

The panel is ejected from the extrusion process at a uniform temperature of $T_i = 275 ^\circ C$ and is allowed to cool on a transport table where the air temperature is $T_\infty = 25 ^\circ C$ and the convection coefficient is $h = 10 \text{ W/m}^2\cdot\text{K}$ (corresponding to natural convection to air for slightly warm conditions). The material properties of the extruded plastic are approximately $\rho = 1050 \text{ kg/m}^3$, $c = 800 \text{ J/kg}\cdot\text{K}$, and $k = 0.50 \text{ W/m}\cdot\text{K}$. Use the PDE toolbox or SolidWorks Simulation to obtain the temperature distribution as a function of time. Take advantage of symmetry for your geometry. If you use the PDE toolbox, use the union and intersection of several basic shapes to approximate the panel geometry. To get more details on how to do this search “Three Ways to Create 2-D Geometry” in the Help window. For example, under the “2-D Geometry Creation at the command line you can find instructions for creating a polygon, ‘P1’, for most of the door panel and then use a circle, ‘C1’, to cut out the radius using the set formula, ‘sf = P1 − C1’. Note that you will have to pad C1 with extra zeros so it is the same length as P1.

Answer the following questions:

(a) Make a temperature contour plot at 60 seconds and discuss if it is consistent with the boundary conditions. Calculate how long it takes the temperature to drop by 99.9% of $(T_i - T_\infty)$, or below $T_\infty + 0.001(T_i - T_\infty)$, at all locations within the domain.
(b) In the MATLAB environment manipulate the data to extract temperatures defined as follows (and shown in Figure 5):

\[ T_1 \] point above the rib center on the outer surface of the panel
\[ T_2 \] point 5 mm from \( T_1 \) along the outer surface
\[ T_3 \] point 13 mm from \( T_1 \) along the outer surface

To validate your results, compare your data to thermocouple measurements that indicate a maximum temperature difference of \((T_1 - T_3) = 18.5 \pm 0.2 \, ^\circ\text{C}\). Plot temperature differences \((T_1 - T_2)\) and \((T_1 - T_3)\) versus time on one figure. Is there a noticeable differential cooling in the region above the rib? If so, explain what is the cause?

(c) View an animation of the temperature contour plots until steady state is reached. Describe major features of the cooling process. Print out 3 temperature contour plots at different times that illustrate some of the important features described. How would you redesign the ribbed panel to reduce this thermally induced defect, while still retaining the stiffening function required by the ribs? Test your proposed design with an additional simulation and present at least one plot of your results along with a discussion.