AME 535a Introduction to Computational Fluid Dynamics
University of Southern California – Fall 2018

Project 4: Finite Element Method
Thermal Fin

Handed Out: 15 Nov. 2018 *Due: 29 Nov. 2018*

Problem Statement

We consider the problem of a thermal fin that removes heat from a surface. The two-dimensional fin, shown in Figure 1, consists of a vertical central "post" and four horizontal "subfins". The fin conducts heat from a prescribed uniform flux "source" at the root, Γ_{root}, through the large-surface-area subfins to surrounding flowing air.

The fin is characterized by a five-component parameter vector, or "input," $\underline{\mu} = (\mu^1, \mu^2, \ldots, \mu^5)$, where $\mu^i = k^i$, $i = 1, \ldots, 4$, and $\mu^5 = \text{Bi}$; $\underline{\mu}$ may take on any value in a specified design set $\mathcal{D} \subset \mathbb{R}^5$. Here k^i is the thermal conductivity of the i^{th} subfin (normalized relative to the post conductivity $k^0 \equiv 1$); and Bi is the Biot number, a nondimensional heat transfer coefficient reflecting convective transport to the air at the fin surfaces (larger Bi means better heat transfer). For example, suppose we choose a thermal fin with $k^1 = 0.4$, $k^2 = 0.6$, $k^3 = 0.8$, $k^4 = 1.1$, and $\text{Bi} = 0.1$; for this particular configuration $\underline{\mu} = \{0.4,\ 0.6,\ 0.8,\ 1.1,\ 0.1\}$, which corresponds to a single point in the set of all possible configurations $\mathcal{D}$ (the parameter or design set). The post is of width unity and height four; the subfins are of fixed thickness $t = 0.25$ and length $L = 2.5$.

We are interested in the design of this thermal fin, and we thus need to look at certain outputs or cost-functionals of the temperature as a function of $\underline{\mu}$. We choose for our output T_{root}, the average steady-state temperature of the fin root normalized by the prescribed heat flux into the fin root. The particular output chosen relates directly to the cooling efficiency of the fin — lower values of T_{root} imply better thermal performance.

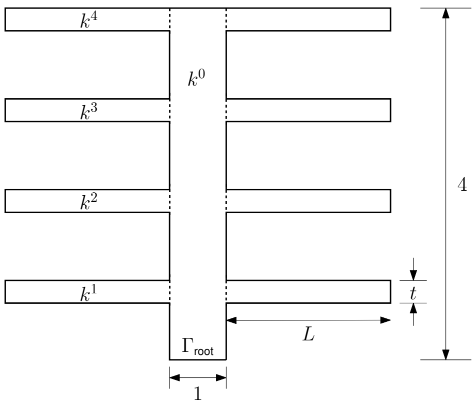

Figure 1: Thermal fin

The steady–state temperature distribution within the fin, $u(\underline{\mu})$, is governed by the elliptic equation

$$-k^i\,\nabla^2 u^i = 0 \ \text{ in } \ \Omega^i, \ \ i = 0, \ldots, 4\ , \tag{1}$$

where ∇^2 is the Laplacian operator, and u^i refers to the restriction of u to Ω^i. Here Ω^i is the region of the fin with conductivity k^i, $i = 0, \ldots, 4$: Ω^0 is thus the central post, and Ω^i, $i = 1, \ldots, 4$, corresponds to the four subfins. The entire fin domain is denoted Ω ($\Omega = \cup_{i=0}^{4}\Omega^i$), and the boundary of Ω is denoted Γ. We must also ensure continuity of temperature and heat flux at the conductivity–discontinuity interfaces $\Gamma^i_{\text{int}} \equiv \partial\Omega^0 \cap \partial\Omega^i$, $i = 1, \ldots, 4$, where $\partial\Omega^i$ denotes the boundary of Ω^i:

$$\left.\begin{array}{rcl} u^0 & = & u^i \\ -(\nabla u^0 \cdot \hat{\mathbf{n}}^i) & = & -k^i(\nabla u^i \cdot \hat{\mathbf{n}}^i) \end{array}\right\} \text{ on } \Gamma^i_{\text{int}}, \ \ i = 1, \ldots, 4;$$

here $\hat{\mathbf{n}}^i$ is the outward normal on $\partial\Omega^i$.

Finally, we introduce a Neumann flux boundary condition on the fin root

$$-(\nabla u^0 \cdot \hat{\mathbf{n}}^0) = -1 \quad \text{on } \Gamma_{\text{root}}, \tag{2}$$

to model the heat source, and a Robin boundary condition

$$-k^i(\nabla u^i \cdot \hat{\mathbf{n}}^i) = \text{Bi } u^i \ \text{ on } \Gamma^i_{\text{ext}}, \ i = 0, \ldots, 4, \tag{3}$$

to model the convective heat losses. Here Γ^i_{ext} is that part of the boundary of Ω^i exposed to the flowing fluid; note that $\cup_{i=0}^4 \Gamma^i_{\text{ext}} = \Gamma \backslash \Gamma_{\text{root}}$.

The average temperature at the root, $T_{\text{root}}(\underline{\mu})$, can then be expressed as $\ell^O(u(\underline{\mu}))$, where

$$\ell^O(v) = \int_{\Gamma_{\text{root}}} v$$

(recall Γ_{root} is of length unity). Note that $\ell(v) = \ell^O(v)$ for this problem.

Note: In addition to this assignment document, we provide MATALB finite element grids and a plotting function. See the Appendix on pages 4 and 5 for explanations.

Appendix 1 describes the grid structure and plotting function.

Appendix 2 presents good coding practices for the finite element method.

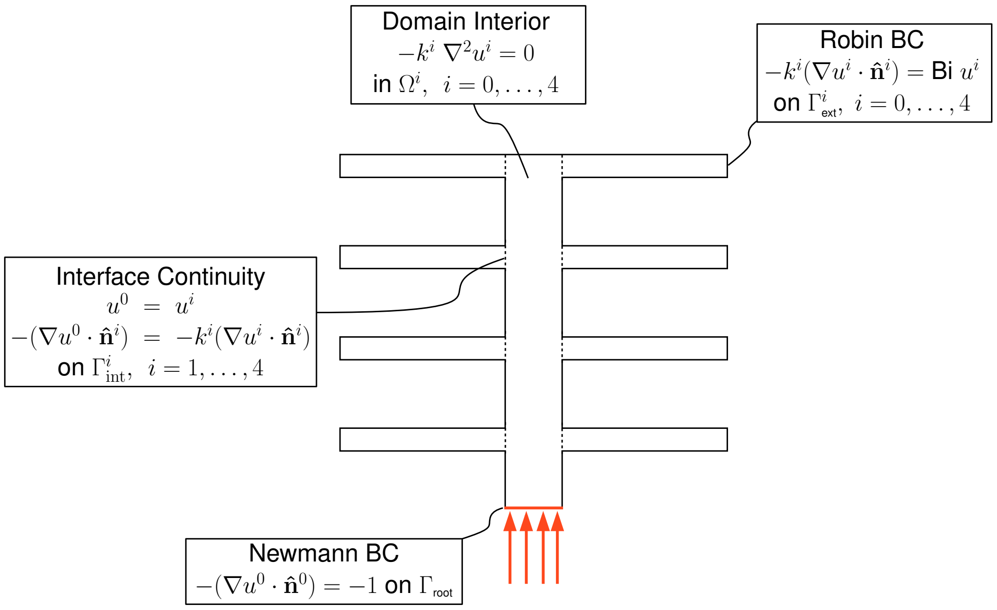
Domain Interior
$-k^i \nabla^2 u^i = 0$
in Ω^i, $i = 0, \ldots, 4$
Robin BC
$-k^i(\nabla u^i \cdot \hat{\mathbf{n}}^i) = $ Bi u^i
on Γ^i_{ext}, $i = 0, \ldots, 4$
Interface Continuity
$u^0 = u^i$
$-(\nabla u^0 \cdot \hat{\mathbf{n}}^i) = -k^i(\nabla u^i \cdot \hat{\mathbf{n}}^i)$
on Γ^i_{int}, $i = 1, \ldots, 4$
Newmann BC
$-(\nabla u^0 \cdot \hat{\mathbf{n}}^0) = -1$ on Γ_{root}

Figure 2: Governing equations and boundary conditions

Questions

1. Show that $u(\underline{\mu}) \in X \equiv H^1(\Omega)$ satisfies the weak form

$$a(u(\underline{\mu}), v; \underline{\mu}) \;=\; \ell(v), \;\; \forall v \in X, \tag{4}$$

with

$$\begin{aligned} a(w, v; \underline{\mu}) &= \sum_{i=0}^{4} k^i \int_{\Omega^i} \nabla w \cdot \nabla v \, dA \;+\; \mathrm{Bi} \int_{\Gamma \backslash \Gamma_{\mathrm{root}}} w \, v \, dS, \\ \ell(v) &= \int_{\Gamma_{\mathrm{root}}} v \, dS. \end{aligned}$$

2. Show that $u(\underline{\mu})$ is the argument that minimizes

$$J(w) = \frac{1}{2} \sum_{i=0}^{4} k^i \int_{\Omega^i} \nabla w \cdot \nabla w \, dA + \frac{\mathrm{Bi}}{2} \int_{\Gamma \backslash \Gamma_{\mathrm{root}}} w^2 \, dS - \int_{\Gamma_{\mathrm{root}}} w \, dS \tag{5}$$

over all functions w in X.

3. We now consider the linear finite element space

$$X_h = \{ v \in H^1(\Omega) \big| \; v|_{T_h} \in \mathbb{P}^1(T_h), \; \forall T_h \in \mathcal{T}_h \} \; ,$$

and look for $u_h(\underline{\mu}) \in X_h$ such that

$$a(u_h(\underline{\mu}), v; \underline{\mu}) = \ell(v), \;\; \forall v \in X_h \; . \tag{6}$$

Our output of interest is then given by

$$T_{\mathrm{root}\, h}(\underline{\mu}) = \ell^O(u_h(\underline{\mu})). \tag{7}$$

Applying our usual nodal basis, we arrive at the matrix equations

$$\begin{aligned} \underline{A}_h \; \underline{u}_h(\underline{\mu}) &= \underline{F}_h, \\ T_{\mathrm{root}\, h}(\underline{\mu}) &= (\underline{L}_h)^T \, \underline{u}_h(\underline{\mu}), \end{aligned}$$

where $\underline{A}_h \in \mathbb{R}^{n \times n}$, $\underline{u}_h \in \mathbb{R}^n$, $\underline{F}_h \in \mathbb{R}^n$, and $\underline{L}_h \in \mathbb{R}^n$; here n is the dimension of the finite element space, and is equal (given our natural boundary conditions) to the number of nodes in $\mathcal{T}_h$.

Derive the elemental matrices $\underline{A}_h^k \in \mathbb{R}^{3\times 3}$, load vectors $\underline{F}_h^k \in \mathbb{R}^3$, and output vectors, $\underline{L}_h^k \in \mathbb{R}^3$, with particular attention to those elements on Γ; and describe the procedure for creating $\underline{A}_h$, $\underline{F}_h$, and $\underline{L}_h$ from these elemental quantities.

4. Write a finite-element code that takes a configuration $\underline{\mu}$ and a triangulation $\mathcal{T}_h$ (see Appendix 1) and returns $u_h(\underline{\mu})$ and $T_{\mathrm{root}\, h}(\underline{\mu})$.

For the particular configuration $\underline{\mu}_0 = \{0.4, \; 0.6, \; 0.8, \; 1.1, \; 0.1\}$, (i) plot the solution $u_h(\underline{\mu}_0)$ (see Appendix 1), and (ii) evaluate the output $T_{\mathrm{root}\, h}(\underline{\mu}_0)$; use $\mathcal{T}_{h_{\mathrm{medium}}}$ for this calculation.

5. Show that

$$T_{\text{root}}(\underline{\mu}) - T_{\text{root}\,h}(\underline{\mu}) = a(e(\underline{\mu}), e(\underline{\mu})), \tag{8}$$

where $e(\underline{\mu}) = u(\underline{\mu}) - u_h(\underline{\mu})$ is the error in the finite element solution. If $u \in H^2(\Omega)$, how would you expect $T_{\text{root}}(\underline{\mu}) - T_{\text{root}\,h}(\underline{\mu})$ to converge as a function of h? In practice, what do you observe?

To answer the latter question, take the finest of the three triangulations given ($\mathcal{T}_{h_{\text{fine}}}$) as the "truth," and suppose that

$$\begin{aligned}(T_{\text{root}})_{h_{\text{fine}}} - (T_{\text{root}})_{2h_{\text{fine}}=h_{\text{medium}}} &= C(2h_{\text{fine}})^b \\ (T_{\text{root}})_{h_{\text{fine}}} - (T_{\text{root}})_{4h_{\text{fine}}=h_{\text{coarse}}} &= C(4h_{\text{fine}})^b;\end{aligned}$$

then find b in the obvious fashion.

Appendix 1: Finite Element Method Grids

For the implementation of the finite element method, three possible triangulations of the fin problem are provided. Save the file `grids.mat` that we provide to your hard drive, and load the triangulation in MATLAB using:

```
>> load grids
```

This creates three variables named `coarse`, `medium,` and `fine`. Each of these variables is a different triangulation $\mathcal{T}_h$ for the fin problem. More specifically

- `coarse` defines $\mathcal{T}_{h_{\text{coarse}}}$, with 1333 nodes and 2095 elements;
- `medium` defines $\mathcal{T}_{h_{\text{medium}}}$, with 4760 nodes and 8380 elements;
- `fine` defines $\mathcal{T}_{h_{\text{fine}}}$, with 17889 nodes and 33520 elements.

Each of these variables is of type struct, with four different fields:

```
>> coarse
coarse =
     nodes:  1333
     coor:  [1333x2 double]
     elements:  2095
     theta:  1x7 cell
```

Description of the fields: (assume that we are using the coarse triangulation)

- `nodes:` The number of nodes in the triangulation.
- `coor:` Two-dimensional matrix with size `nodes`$\times 2$, where each row i has the x and y coordinates for node i. For example, the location of node 49 can be determined by two coordinates: the coordinate in the x-direction would be `coarse.coor(49,1)` and in the y-direction `coarse.coor(49,2)`.
- `elements:` The number of elements in the triangulation.
- `theta:` The adjacency matrix $\theta(k, \alpha)$ which defines the local-to-global mapping required in the elemental assembly procedure. Since we have regions with different physical properties, for each region a separate adjacency matrix is provided. The regions considered are
  - Region 1: Domain Ω^1, $\theta^1(k, \alpha) =$ `coarse.theta{1}`
  - Region 2: Domain Ω^2, $\theta^2(k, \alpha) =$ `coarse.theta{2}`
  - Region 3: Domain Ω^3, $\theta^3(k, \alpha) =$ `coarse.theta{3}`
  - Region 4: Domain Ω^4, $\theta^4(k, \alpha) =$ `coarse.theta{4}`
  - Region 5: Domain Ω^0, $\theta^5(k, \alpha) =$ `coarse.theta{5}`

  For each of these regions i, the index k varies in the range $k \in \{1, \ldots, n_i\}$, where n_i are the number of elements in region i. For example element 12 in region 3 has the global nodes ν_1=`coarse.theta{3}(12,1)`, ν_2=`coarse.theta{3}(12,2)`, and ν_3=`coarse.theta{3}(12,3)`.

In addition, for the treatment of the boundary conditions, the boundary is divided into two sections. The first is $\Gamma \backslash \Gamma_{\text{root}}$, where Robin boundary conditions are applied. The second is Γ_{root}, where the incoming heat flux is applied. For each segment in these sections, the associated global nodes are listed:

- Section 1: $\Gamma \backslash \Gamma_{\text{root}}$, $\kappa^1(m, \alpha) =$ `coarse.theta{6}`
- Section 2: Γ_{root}, $\kappa^2(m, \alpha) =$ `coarse.theta{7}`

For each of the sections i, the index m varies in the range $m \in \{1, \ldots, s_i\}$, where the s_i are the number of segments in section i. As an example, to find the nodes ν_1, and ν_2 for segment 5 in the first section, we would use $\nu_1 =$ `coarse.theta{6}(5,1)`, and $\nu_2 =$ `coarse.theta{6}(5,2)`.

Plotting

To plot the temperature distribution, use the function `plotsolution.m` provided. If $z \equiv \underline{u}_h$ is the vector with the computed temperature values for each of the nodes, then a contour plot of the temperature distribution can be obtained by

```
>> plotsolution(coarse, z)
```

The first argument is the mesh used in the calculation of z, and the second is the solution vector z.

Other

For the storage of the finite element matrices, use MATLAB's sparse matrix data structure. Also, for the solution of the resulting linear systems, use the default solution methods provided in MATLAB.

Appendix 2: Good Coding Practices for the Finite Element Method

The finite element method may seem counter-intuitive, but with some good coding practices it is actually easy to program and maintain. The following skeleton gives the steps It assumes triangular elements but can modified for other element shapes.

```
-- Initializations
A = sparse(N,N);
F = sparse(N,1);

-- Domain Interior
for i = ... [ loop over domain regions ]
   for k = ... [ loop over elements: triangles ]
      N_global = local_to_global(k)
      Alocal   = ...  [ 3x3 matrix: 3 nodes per triangle ]
      A(N_grobal,N_global) = A(N_global,N_global) + Alocal
   end
end

-- Boundaries (except root)
for k = ... [ loop over elements: edges ]
   N_global = local_to_global(k)
   Alocal   = ...  [ 2x2 matrix: 2 nodes per edge ]
   A(N_grobal,N_global) = A(N_global,N_global) + Alocal
end

-- Root Boundary (Neumann)
for k = ... [ loop over elements: edges ]
   N_global = local_to_global(k)
   Flocal   = ...  [ 2x1 matrix: 2 nodes per edge ]
   F(N_grobal) = F(N_global) + Flocal
end

-- Solve the linear system A*u = F
u = A\F
```