# 16.901: Project #2
# Steady and Transient Heat Transfer Simulation of a Turbine Blade Solution

## 1 Background

In the first stages of a turbine, the turbine blades are subjected to a high temperature flow due to the hot gas produced in the combustor. As a result, turbine blades are often internally cooled by pumping low temperature air through the blades in passages. In this project, we will simulate the heat transfer of an internally cooled turbine blade using a finite element discretization. The blade has three cooling passages (a leading-edge, midbody, and trailing edge passage) with the trailing edge passage connected to a trailing edge cooling slot. The blade and the coarsest mesh to be used are shown in Figure 1. The problem we are considering is representative of a first stage rotor of a turbofan.

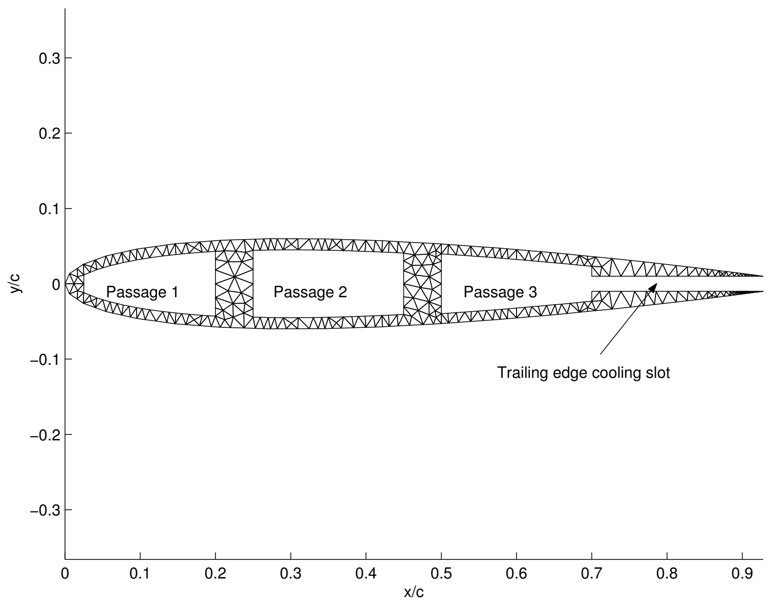


Figure 1: Blade geometry with cooling passages including coarse grid

We will be considering both steady and transient heat transfer simulation. The problem is to be modeled using the unsteady diffusion equation,

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T), \tag{1}$$

where $\rho$, $c_p$, and $k$ are the density, specific heat, and thermal conductivity of the blade. For this problem, we will assume that,

$$\rho = 8000\, kg/m^3, \qquad c_p = 400\, J/(kg\, K), \qquad k = 30\, W/(m\, K).$$

The boundary conditions on all surfaces will be modeled for convective heat transfer. Recall that the heat transfer rate is given by,

$$\vec{q} = -k\nabla T.$$

where $k$ is the thermal conductivity of the blade material. For all surfaces (both internal and external to the blade), the heat flux out of the blade will be given by,

$$\vec{q} \cdot \vec{n} = h_{ext}\left(T - T_{ext}\right),$$

where $h_{ext}$ is the convective heat transfer coefficient and $T_{ext}$ is the temperature (away from the surface). Note, $\vec{n}$ is a normal pointing out of the blade. For the internal cooling passages and trailing edge slot, we will assume that,

$$T_{ext} = 600\,K, \qquad h_{ext} = 1500\,W/(m^2K).$$

For the outer surface of the blade, the temperature will be assumed to be,

$$T_{ext} = 1500\,K.$$

The heat transfer coefficient will be larger at the leading edge, and this is modeled with the following form,

$$h_{ext}(x) = 4000\left[1 + 3e^{-4\left(\frac{x/c}{0.05}\right)^2}\right]\,W/(m^2K), \tag{2}$$

where $c$ is the chord of the blade and for this problem,

$$c = 0.04\,m.$$

A plot of the heat transfer coefficient as a function of $x/c$ is shown in Figure 2.

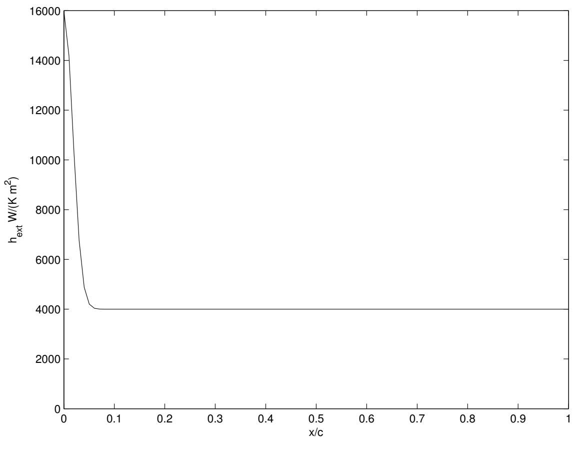


Figure 2: Heat transfer coefficient on the outer surface of the blade as a function of chordwise location.

An FEM solver has been provided which solves the steady state problem described above using linear finite elements. The main script is **bladeheat.m**. The implementation uses exact evaluation of all integrals except for the convective (Robin) boundary condition on the outer surface of the blade. On this boundary, $h_{ext}$ on the $j$-th boundary edge is approximated by $h_{ext}(\bar{x}_j)$ where $\bar{x}_j$ is the midpoint of the $j$-th edge and $h_{ext}$ is given by Equation (2).

## 2 Tasks

### 2.1 Accuracy Study for Steady Heat Transfer Analysis

The first task is to study the accuracy of FEM solver for the steady state problem. Three meshes have been provided and are stored in the MATLAB datafiles **hpblade_coarse.mat**, **hpblade_medium.mat**, and **hpblade_fine.mat**. NOTE: when you run the **bladeheat.m** script, do not include the **.mat** extension when inputting the filename. The medium and fine meshes are refinements of the coarse mesh constructed by bisecting each edge of the mesh. Thus, each triangle on the coarse mesh produces four triangles on the medium mesh, and 16 triangles on the fine mesh. Since an exact solution for this problem is not available, assume that the fine mesh results are representative of the exact solution to this problem. From the solutions on the three meshes, estimate the order of accuracy of these results (i.e. is the algorithm first-order accurate, second-order accurate, something else).

Specifically, investigate the convergence rate of the following quantities,

- the minimum temperature in the blade: $T_{\text{min}}$,
- the maximum temperature in the blade: $T_{\text{max}}$,
- the temperature at point A, $(x_A/c, y_A/c) = (0.015, 0.015)$: $T_A$,
- the temperature at point B, $(x_B/c, y_B/c) = (0.225, 0.050)$: $T_B$,
- the temperature at point C, $(x_C/c, y_C/c) = (0.475, 0.050)$: $T_C$,
- the temperature at point D, $(x_D/c, y_D/c) = (0.800, 0.020)$: $T_D$.

The minimum and maximum temperatures are already calculated in the provided scripts. To help find the temperature at the other points, first the element containing the points must be found. Then, using the basis functions and the nodal values of the temperature, the value at the point must be determined. To help in this process, the function **findloc.m** can be used to find the element containing a point (see the function for information on the calling arguments).

**Solution:** To find $\tilde{T}$ at a point $(x, y)$ in an element, the location in the reference coordinates must be found which can be done using Equation (13.8) from the notes, specifically,

$$\begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} = \begin{pmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{pmatrix}^{-1} \begin{pmatrix} x - x_1 \\ y - y_1 \end{pmatrix}$$

Then, given this location in the cell, the value of $\tilde{T}$ can be found from,

$$\tilde{T}(\xi_1, \xi_2) = \sum_{i=1}^{3} T_i \phi_i(\xi_1, \xi_2),$$

where $T_i$ are the nodal values of the temperature and $\phi_i$ are the linear nodal basis functions,

$$\begin{aligned} \phi_1 &= 1 - \xi_1 - \xi_2, \\ \phi_2 &= \xi_1, \\ \phi_3 &= \xi_2. \end{aligned}$$

| Mesh | $\tilde{T}_{\min}$ | $\tilde{T}_{\max}$ | $\tilde{T}_A$ | $\tilde{T}_B$ | $\tilde{T}_C$ | $\tilde{T}_D$ |
|---|---|---|---|---|---|---|
| Fine | 1149.13 | 1430.56 | 1404.94 | 1210.83 | 1221.48 | 1248.04 |
| Medium | 1149.99 | 1430.39 | 1404.74 | 1211.20 | 1221.74 | 1248.09 |
| Coarse | 1152.25 | 1429.85 | 1404.17 | 1211.81 | 1222.21 | 1248.22 |

Table 1: Dependence of steady-state temperature estimates on the mesh resolution.

| Error | $\tilde{T}_{\min}$ | $\tilde{T}_{\max}$ | $\tilde{T}_A$ | $\tilde{T}_B$ | $\tilde{T}_C$ | $\tilde{T}_D$ |
|---|---|---|---|---|---|---|
| Medium | 0.86 | -0.17 | -0.20 | 0.37 | 0.26 | 0.05 |
| Coarse | 3.12 | -0.71 | -0.77 | 0.98 | 0.73 | 0.18 |
| Ratio | 3.63 | 4.11 | 3.89 | 2.65 | 2.82 | 3.52 |
| Rate | 1.86 | 2.04 | 1.96 | 1.41 | 1.49 | 1.82 |

Table 2: Dependence of steady-state temperature errors on the mesh resolution.

This calculation has been implemented in the function **findval.m** which is included with the Matlab solution scripts.

The actual temperatures are shown in Table 1 and the errors on the coarse and medium meshes are shown in Table 2. The errors are with respect to the fine mesh quantities,

$$E_{coarse} = \tilde{T}_{coarse} - \tilde{T}_{fine}, \qquad E_{medium} = \tilde{T}_{medium} - \tilde{T}_{fine}.$$

The ratio is $E_{coarse}/E_{medium}$. If the method if $p$-th order accurate for a specific quantity, then the error will be,

$$E = O(h^p).$$

Thus, the ratio of the errors from coarse-to-medium mesh would be,

$$\frac{E_{coarse}}{E_{medium}} = O\left(\left(\frac{h_{coarse}}{h_{medium}}\right)^p\right).$$

Since the mesh spacing decreases by a factor of 2 with each mesh (since the edges of the mesh are bisected), then the order of accuracy can be approximated as,

$$p = \log_2 \frac{E_{coarse}}{E_{medium}}.$$

As seen from Table 2, depending on the specific value, the errors converge at a rate between first and second order. While most errors are reduced at nearly second order, the errors at points B and C are somewhat slower converging at only about $p = 1.4$.

## 2.2 Design and Implementation of FEM for Transient Diffusion Problem

The problem to be solved is the initial transient when a blade goes from a uniform (cool) temperature to a steady hot temperature environment (i.e. simulating the process of starting a turbofan). We will assume that the blade is initially at a uniform temperature,

$$T(x, y, t = 0) = 300\,K.$$

You are to develop an efficient, accurate method to solve this transient problem. It is desired to produce accurate temperature estimates over the timescale it takes for the temperature distribution in the blade to reach its (essentially) steady value. NOTE: the final solution of this transient problem after a long enough time should be the same as the steady solution you calculated in Section 2.1.

The basic approach for time dependent problems is to use the same solution basis as for steady diffusion but assume that the weights are functions of time,

$$\tilde{T}(x,y,t) = \sum_{i=1}^{N} a_i(t)\phi_i(x,y).$$

Introducing this into Equation (1), weighting by $\phi_j$ and integrating produces the following equation,

$$\rho c_p \sum_{i=1}^{N} \frac{da_i}{dt} \int_\Omega \phi_i \phi_j \, dA = \text{standard terms for } j\text{-th weighted residual}.$$

Defining the mass matrix, $M$ for which,

$$M_{j,i} = \rho c_p \int_\Omega \phi_i \phi_j \, dA = M_{i,j},$$

gives,

$$M\frac{da}{dt} = Ka + b. \tag{3}$$

Here, $K$ and $b$ are the standard stiffness matrix and boundary/forcing condition vector arising from discretization of the Laplacian. In general, $K$ and $b$ could depend on time, but in the problem we are considering, they are not time-dependent.

Answer/perform the following tasks:

1. What is the timescale over which this transient will occur? Specifically, construct an estimate of how long the transient is likely to last. You could do this estimate using a 0-D lumped model (such as you likely saw in 16.05).

   **Solution:** Using a lumped mass model, we assume that the temperature of the material is uniform. In that case, the following ODE arises,

$$A_{blade}\rho c_p \frac{dT}{dt} = s_{cool}h_{cool}(T_{cool} - T) + s_{hot}h_{hot}(T_{hot} - T), \tag{4}$$

   where $A_{blade}$ is the cross-sectional area of the blade material. $s_{cool}$ is the length of the surface exposed to the cool passages, and $h_{cool} = 1500W/(m^2K)$ and $T_{cool} = 600\,K$ are the heat transfer coefficients and temperature in the cooling passages. Similarly, $s_{hot}$ is the length of the surface exposed to the main gaspath (hot) air, and $h_{hot}$ and $T_{hot} = 1500\,K$ are the heat transfer coefficient and temperature of the main gaspath. While the external gaspath heat transfer coefficient depends on $x$, over most of the airfoil $h_{hot} \approx 4000W/(m^2K)$. So, a rough estimate will be $h_{hot} = 4000W/(m^2K)$. To determine $A_{blade}$, $s_{cool}$, and $s_{hot}$, the **bladeheat.m** script was modified to sum the area of each element, and to sum the edge lengths of the boundary edges (separated into cooling passage and gaspath edges) giving,

$$A_{blade} = 5.51 \times 10^{-5} m^2, \quad s_{cool} = 0.08m, \quad s_{hot} = 0.076m.$$

The solution to Equation (4) is,

$$T(t) = T_{final} + (T_{init} - T_{final})e^{-t/\tau}, \tag{5}$$

where

$$T_{final} = \frac{s_{cool}h_{cool}T_{cool} + s_{hot}h_{hot}T_{hot}}{s_{cool}h_{cool} + s_{hot}h_{hot}}, \quad \tau = \frac{s_{cool}h_{cool} + s_{hot}h_{hot}}{A_{blade}\rho c_p}.$$

Substituting in the assumed values gives,

$$T_{final} = 1245\,K, \quad \tau = 2.4\,\text{sec}.$$

As a measure of the timescale until the transient completes, the time at which the transient is 99% complete is used. This occurs when $\exp(-t_{steady}/\tau) = 0.01$, or $t_{steady} = 4.6\tau = 11$ seconds.

2. Given the spatial resolution of the meshes, what would the timestep limits be if an explicit time-marching method were used? You should perform this estimate using Fourier stability analysis as discussed in the finite difference portion of the lectures. For this, you can assume that the finite element method being studied will behave similar to a standard central difference finite difference discretization of diffusion. You should also perform a matrix stability analysis for this problem to find the eigenvalues. To do this, you will need to calculate the mass matrix and then you can use Matlab to calculate the eigenvalues and include plots of them for all three grids. How does the largest magnitude eigenvalue vary with grid size? Note, the eigenvalues, $\lambda$, for this problem are the roots of,

$$\det(\lambda M - K) = 0.$$

Since $K$ and $M$ are sparse matrices, the best option to calculate all of the eigenvalues in Matlab would be the command,

```
lambda = eig(full(K),full(M));
```

Compare the Fourier-based estimate with the actual (matrix-based) eigenvalues of the problem.

**Solution:** Assuming that the FEM discretization behaves like a central difference discretization of diffusion, Fourier analysis can be used to estimate the eigenvalues for the blade heat transfer problem. A one-dimensional model problem is used in which the node spacing, $\Delta x$, is based on the smallest edge size observed in the meshes. A one-dimensional diffusion problem using a central difference discretization is,

$$\rho c_p \frac{dT_j}{dt} = \frac{k}{\Delta x^2}\left(T_{j+1} - 2T_j + T_{j-1}\right).$$

To perform Fourier analysis, substitute,

$$T_j(t) = \hat{T}_m(t)e^{ij\theta_m}, \qquad -\pi \le \theta_m = k_m \Delta x \le \pi.$$

This leads to,

$$\begin{aligned} \rho c_p \frac{d\hat{T}_m}{dt} &= \frac{k}{\Delta x^2}\left(e^{i\theta_m} - 2 + e^{-i\theta_m}\right)\hat{T}_m, \\ \Rightarrow \frac{d\hat{T}_m}{dt} &= 2\frac{k}{\rho c_p \Delta x^2}(\cos\theta_m - 1)\hat{T}_m. \end{aligned}$$

Thus, the eigenvalues are,

$$\lambda_m = 2\frac{k}{\rho c_p \Delta x^2}(\cos\theta_m - 1),$$

which are non-positive real values from 0 for $\theta_m = 0$ to $-4k/(\rho c_p \Delta x^2)$ for $\theta_m = \pm\pi$. On the finest mesh, the smallest edge length is approximately 5E-06 m which gives the largest magnitude eigenvalue as,

$$\lambda_{\max} = -1.5\text{E}{+}06\, s^{-1}.$$

The eigenvalues can also be calculated directly from the $M$ and $K$ matrices as described above. This was done for the coarse and medium mesh and the resulting eigenvalues are shown in Figure 3. On the fine mesh, the eigenvalue calculation required too much time to complete and was not performed. For the coarse mesh, the eigenvalues are from $-3.9\text{E}{+}05 \le \lambda \le -1.5$. For the medium mesh, the eigenvalues are from $-1.8\text{E}{+}06 \le \lambda \le -1.5$. Thus, the largest magnitude eigenvalue is increasing by a factor of approximately 4 with a halving of the mesh spacing. As a result, for the finest mesh, the largest magnitude eigenvalue is expected to be approximately,

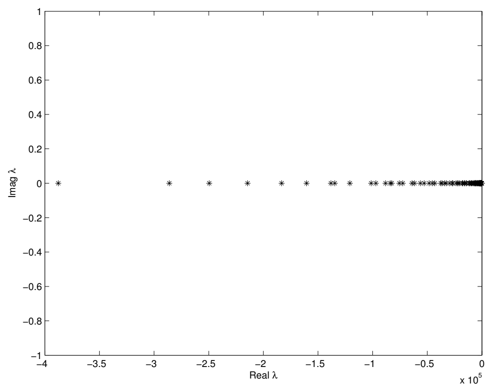
$$\lambda_{\max} \approx -7\text{E}{+}06\, s^{-1},$$

which is about four-to-five time larger than the Fourier analysis, but with the correct order of magnitude.

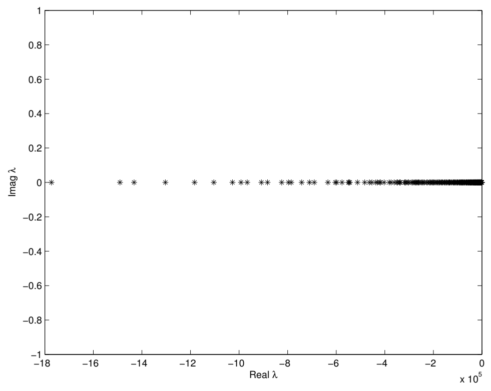
For explict schemes, the largest magnitude $\lambda\Delta t$ which remain stable are $O(1)$. For example, forward Euler requires negative real eigenvalues to be less than 2 in magnitude. Thus, for this problem, on the finest mesh, the timesteps would be limited to,

$$\begin{aligned} \lambda_{\max}\Delta t &= O(1), \\ \Delta t &= O(\lambda_{\max}^{-1}), \\ \Delta t &= O(10^{-7})\, s. \end{aligned}$$

3. Based on your estimates for the expected timescale of the transients and the timestep for stability of an explicit scheme, pick an appropriate time integration method for this application and explain your rationale.



(a) Coarse mesh



(b) Medium mesh

Figure 3: Eigenvalues for coarse and medium mesh.

**Solution:** Given the estimates for $t_{steady}$ and for $\Delta t$ given an explicit scheme, clearly an explicit scheme would require far too many iterations just to maintain stability (i.e. $t_{steady}/\Delta t \approx 10^8$ iterations). Thus, an implicit scheme is critical, in particular one with a large stability region along the negative real axis. This suggests using the backwards difference integrators. The simplest option would be backward Euler, however, this is only first-order accurate in time. Since the accuracy of the linear finite element method is second order (i.e. $O(\Delta x^2)$), then halving the grid spacing would actually require the timestep to reduce by a factor of 4 to achieve the same error reduction if backward Euler were used. Thus, a better choice would be second-order backward differencing and is the method implemented below.

Note: another option would be the trapezoidal method which is also second-order accurate and stable for any eigenvalue that has a negative real part. However, as observed in class, while the trapezoidal method is stable, it has very poor damping for large magnitude (negative real) eigenvalues. To summarize that previous in-class discussion, recall that the trapezoidal method is,

$$v^{n+1} = v^n + \frac{1}{2}\Delta t\left(f^{n+1} + f^n\right).$$

Calculating the amplification factor by substituting $f = \lambda v$ gives,

$$g_{trap}(\lambda\Delta t) = \frac{1 + \frac{1}{2}\lambda\Delta t}{1 - \frac{1}{2}\lambda\Delta t}.$$

Although the trapezoidal method is stable for any $\Delta t$ when the real part of $\lambda$ is negative, in fact the behavior of the trapezoidal method when $\lambda\Delta t \to -\infty$ is still undesirable. Specifically,

$$\begin{aligned} g_{trap}(\lambda\Delta t) &= \frac{1 + \frac{1}{2}\lambda\Delta t}{1 - \frac{1}{2}\lambda\Delta t}, \\ &= -\frac{1 + \frac{2}{\lambda\Delta t}}{1 - \frac{2}{\lambda\Delta t}}, \\ &\approx -1 - \frac{4}{\lambda\Delta t} \qquad \text{for } |\lambda\Delta t| \to \infty. \end{aligned}$$

Thus, for large negative real eigenvalues, $g_{trap}$ will oscillate every iteration with very little damping (since $g_{trap} \approx -1$). This is clearly not realistic since the analytic solution for large negative real eigenvalues is damped quickly and do not oscillate. In conclusion, for this application, trapezoidal integration will not be a good option.

4. Implement your chosen time-integration method and provide a brief description of your implementation.

   **Solution:** The second-order backward differencing algorithm for Equation (3) is,

$$\frac{1}{\Delta t}M\left(a^{n+1} - \frac{4}{3}a^n + \frac{1}{3}a^{n-1}\right) = \frac{2}{3}\left(Ka^{n+1} + b\right),$$

   which can be re-arranged into the following linear system for $a^{n+1}$,

$$\left(\frac{1}{\Delta t}M - \frac{2}{3}K\right)a^{n+1} = \frac{1}{\Delta t}M\left(\frac{4}{3}a^n - \frac{1}{3}a^{n-1}\right) + \frac{2}{3}b. \tag{6}$$

   The second-order backward differencing algorithm cannot be used for the first iteration (since the $n-1$ step is not available). Thus, on the first iteration, backward Euler will be used,

$$\frac{1}{\Delta t}M\left(a^{n+1} - a^n\right) = Ka^{n+1} + b,$$

   which can be written,

$$\left(\frac{1}{\Delta t}M - K\right)a^{n+1} = \frac{1}{\Delta t}Ma^n + b. \tag{7}$$

   In the modified **bladeheat.m** script, Equations (6) and (7) are implemented.

## 2.3 Accuracy Study for Transient Heat Transfer Analysis

Study the time evolution of the temperature distribution using your method. In particular, perform the following tasks:

1. Investigate the accuracy of the method as the timestep and grid spacing are reduced. Specifically, monitor the evolution of $T_{\min}$, $T_{\max}$, $T_A$, $T_B$, $T_C$, and $T_D$ versus time. Include plots of some of your results to demonstrate the performance of the algorithm. Be selective in your use of plots, i.e. only chose results that make a clear point about the accuracy of the method (do not simply provide a compendium of every case you had time to run). Use as your 'exact' answer the transient simulation on the finest mesh using the smallest timestep that was feasible to run.

   **Solution:** The first step taken was to perform calculations on the coarsest mesh determine what the timescale of the transient was. This was not to directly study the accuracy, but rather to ensure that the simulation was run for the actual length of time of the transient (as opposed to simulating either far too short or too long of a time interval). Only the coarse mesh was used for this part, but the timescales were monitored for finer meshes and not found to be substantialy impact by the mesh resolution. Of the temperatures monitored, $T_{\min}$ was found to have the longest timescale and the evolution of $T_{\min}$ can be seen in Figure 4 for $\Delta t = 1$, 0.1, and 0.01 seconds. The simulations were run until $t = 10$ seconds (based on the $t_{steady} \approx 11$ seconds estimated above) though the plot shows only the first 6 seconds. As can be seen, the results for $\Delta t = 0.1$ and 0.01 seconds are nearly the same, while the $\Delta t = 1$ second results have significant error. For the smaller $\Delta t$ results, the minimum temperature is within a percent of its final value beyond approximately 4 seconds. Thus, for the remaining simulations, only the timespan from $t = 0$ to 4 seconds was considered since the solutions are nearly steady at that point.

   Next, a series of calculations was performed on the finest mesh to study the time accuracy and determine a sufficient $\Delta t$ for the problem. The evolution of $T_{\max}$ was found to have amoung the shortest transients and so only the evolution of $T_{\max}$ is used to study the impact of $\Delta t$. The evolution of $T_{\max}$ can be seen in Figure 5 for $\Delta t = 0.001$, 0.01, and 0.1 seconds. A change in the temperature rate of change (i.e. $dT_{\max}/dt$) can be observed at $t \approx 0.2$ seconds for simulations on the smallest timesteps, though not $\Delta t = 0.1$. Evidently, the location of the maximum value is switching at this time such that different solution trajectories are observed and this switching is not accurately determined with $\Delta t = 0.1$. While it is not clear if this effect would be of engineering significance, a timestep of $\Delta t = 0.01$ seconds will be used for all further simulations.

   The next study was to qualitatively determine the impact of the mesh size on the temporal evolution. The coarse, medium, and fine mesh evolution of $T_{\max}$ is shown in Figure 6. As can be clearly seen, the results are virtually identical. Thus, the recommended combination of grid and timestep which provide accurate answers yet at minimal costs are the coarse mesh with $\Delta t = 0.01$ seconds.

2. For this problem and your algorithm, what would you recommend for the combination

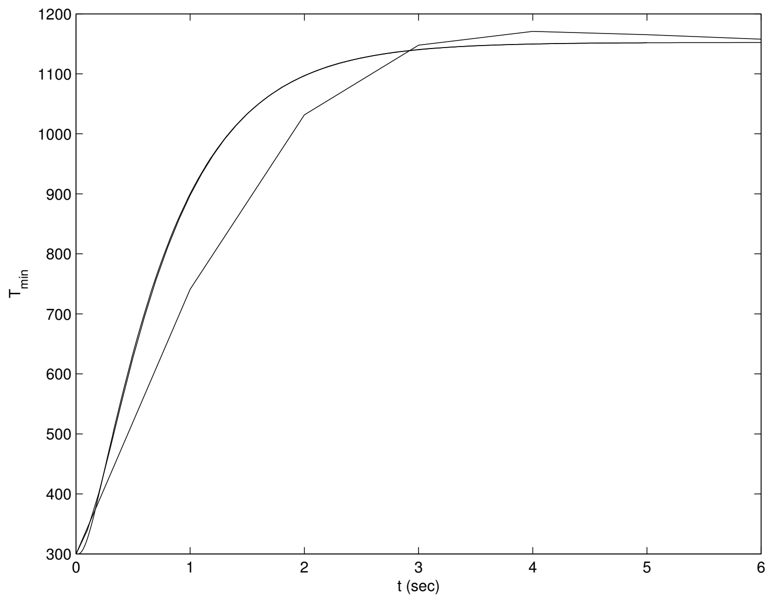


Figure 4: Minimum temperature evolution using coarsest mesh for simulations with $\Delta t = 1$ (solid), 0.1 (dash), and 0.01 (dash-dot) seconds.

of time step and mesh to provide accurate answers in a minimal time. Provide wallclock timing information for how long your recommended combination took to complete a simulation over the entire transient. Also, report the time to complete a single steady solution on the coarse mesh. You can time your algorithms using the Matlab commands: **tic** and **toc** (see the help page for how to use these). NOTE: start your timing (by issuing the tic command) after the input file has been loaded to avoid timing how long it took you to type in the filename.

**Solution:** For the recommended combination of coarse mesh and $\Delta t = 0.01$ seconds, the simulation required 5.9 seconds to run to $t = 4$ seconds. To determine the steady state solution time (i.e. with iterating in time), the original **bladeheat.m** scripts was run. For the coarse mesh, 0.4 seconds were required to complete the simulation. Thus, the transient simulation was about 14.75 times more expensive than a single steady staye solution.

3. For your chosen combination of timestep and grid, provide complete time histories for the 6 temperatures. Also, include some plots of the temperature distribution at roughly 10%, 25%, 50%, 100% through the transient. How did your initial estimate of the transient timescale compare to that observed in the simulation? If your initial estimate was not reasonably accurate, what do you think the reason was?

   **Solution:** For the recommended combination, the temperature evolutions are shown in Figure 7. Also, the temperature distributions in the blade at $t = 0.4$, 1, 2, and 4 seconds are shown in Figures 8-11. The initial estimate of the transient timescale was approximately 11 seconds. The actual timescale observed at which the temperatures

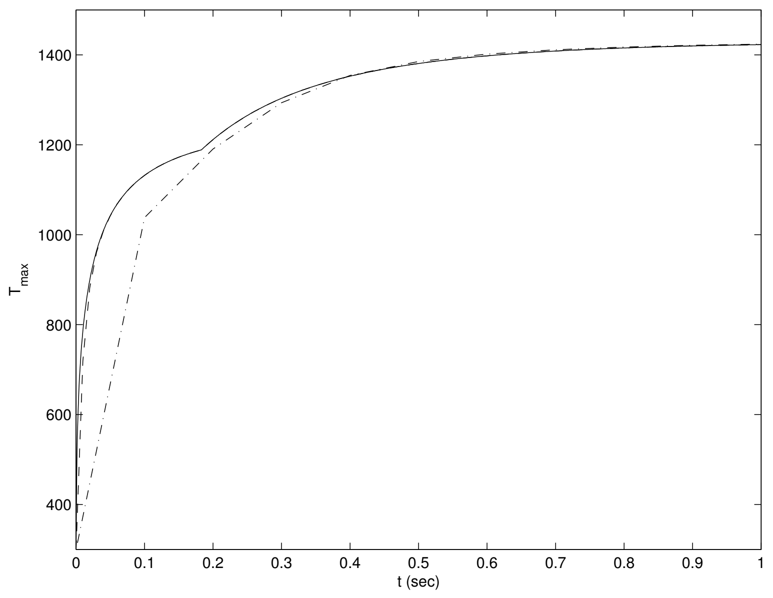


Figure 5: Maximum temperature evolution using fine mesh for simulations with $\Delta t = 0.1$ (solid), 0.01 (dash), and 0.001 seconds (dash-dot).

were within 1% of their final values was about 4 seconds. Thus, while the estimate was in the correct oder of magnitude, it was about a factor 3 too high. One possible explanation for the overestimated timescale is that the 0-D model assumes the temperature is average in the blade throughout its evolution; however, this is clearly not the case as the temperature is far from uniform throughout the entire transient (except for the initial condition). This lack of uniformity means that temperature gradients will be present which would increase the heat transfer rate and likely speed the temperature rise faster than that predicted by the lumped model.

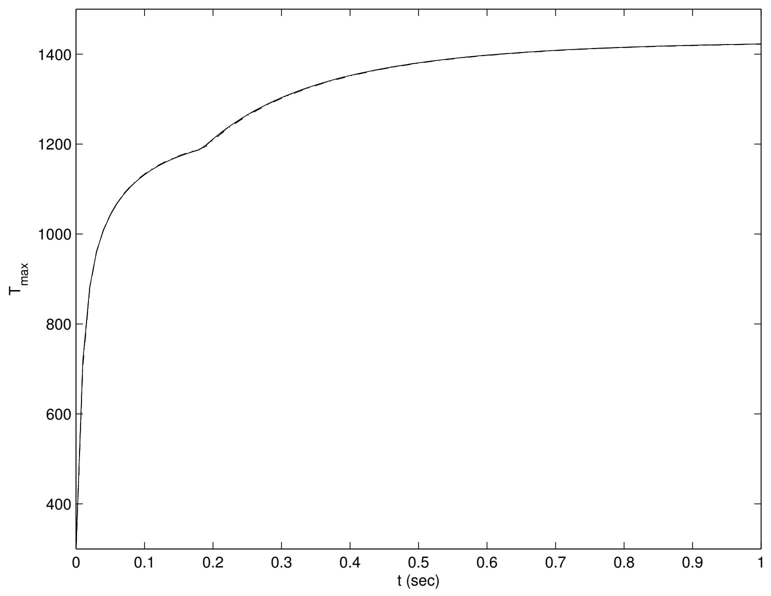


Figure 6: Maximum temperature evolution for $\Delta t = 0.01$ seconds using fine (solid), medium (dash), and coarse (dash-dot) meshes.

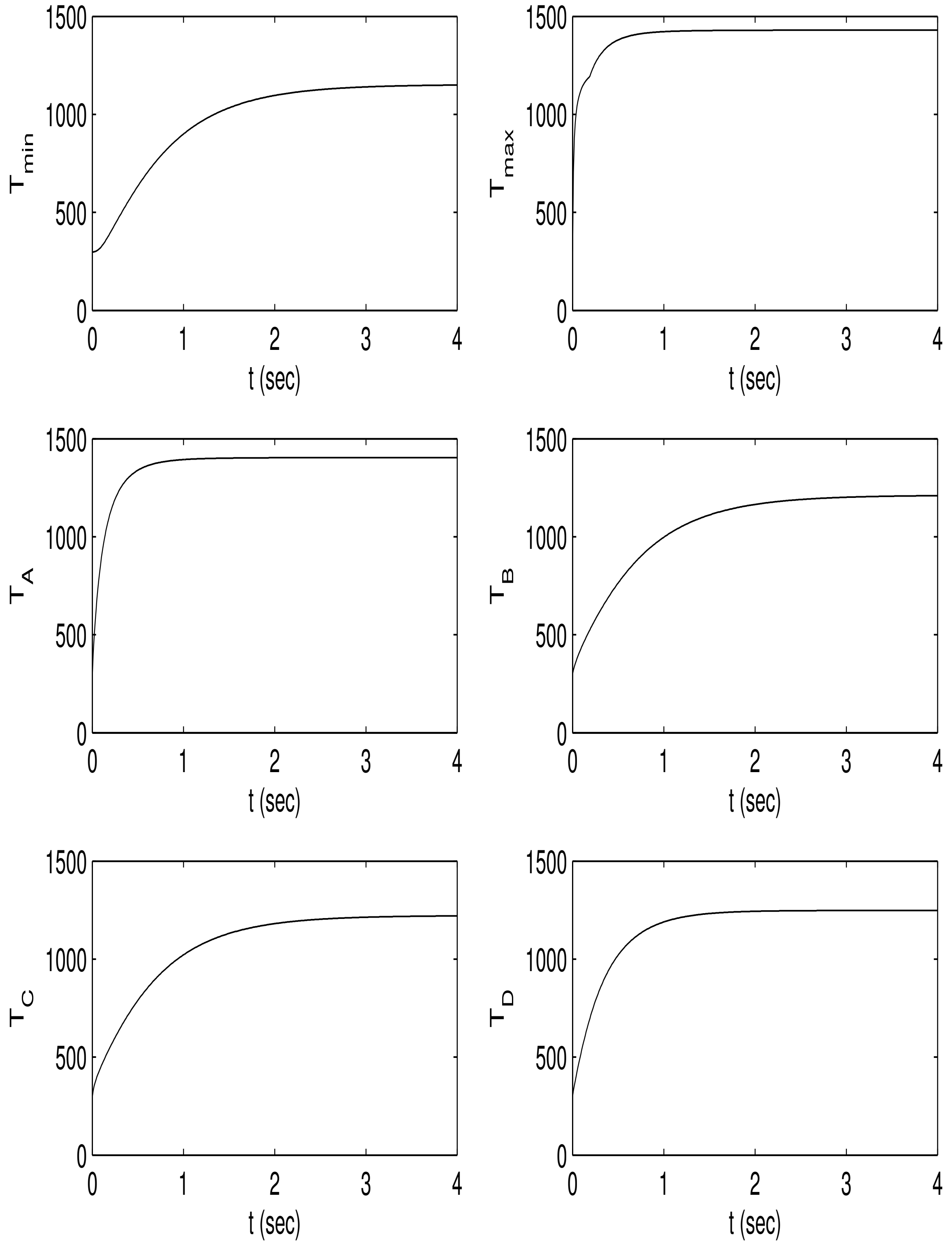


Figure 7: Temperature evolution for $\Delta t = 0.01$ seconds using coarse mesh.

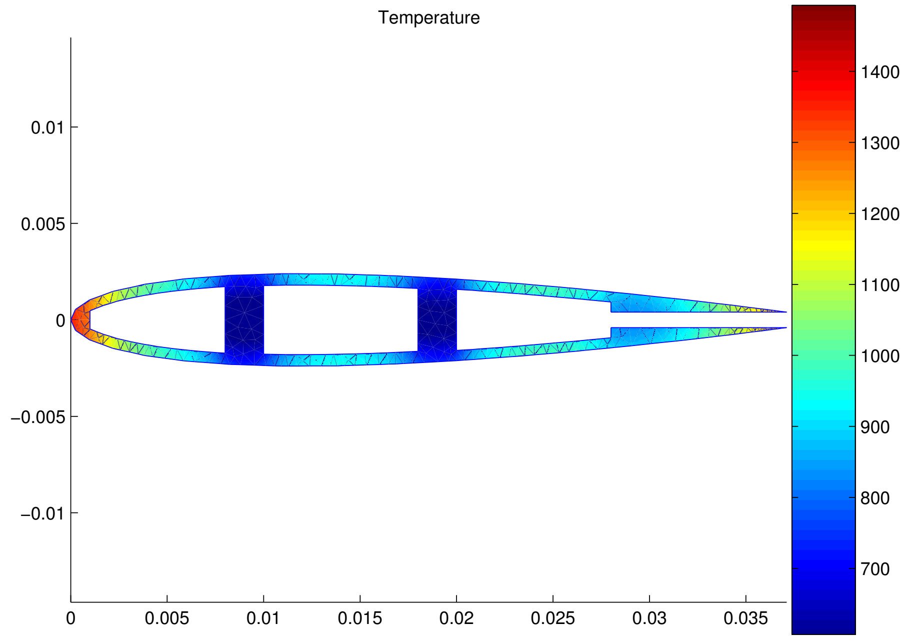


Figure 8: Temperature distribution in blade at $t = 0.4$ seconds.

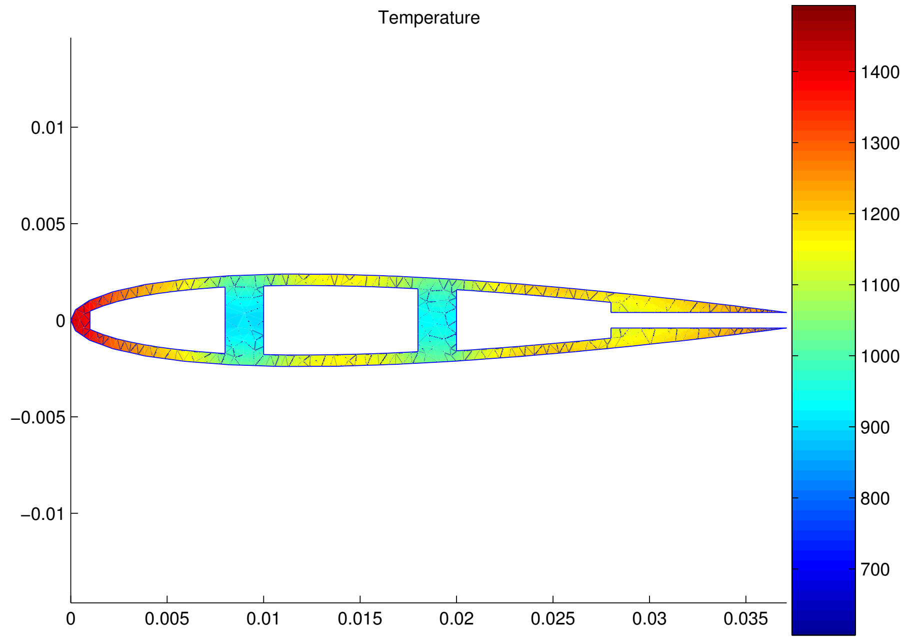


Figure 9: Temperature distribution in blade at $t = 1$ seconds.

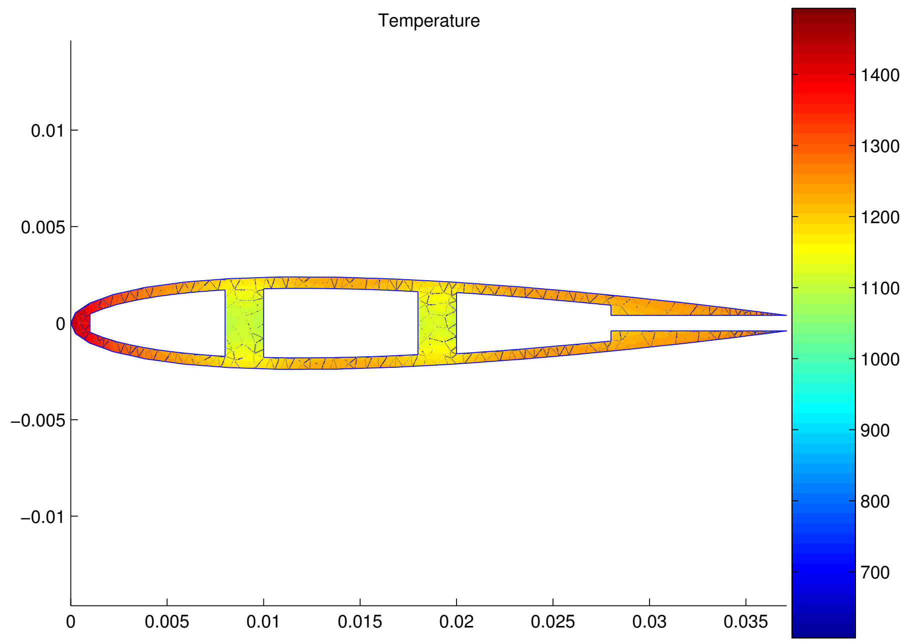


Figure 10: Temperature distribution in blade at $t = 2$ seconds.

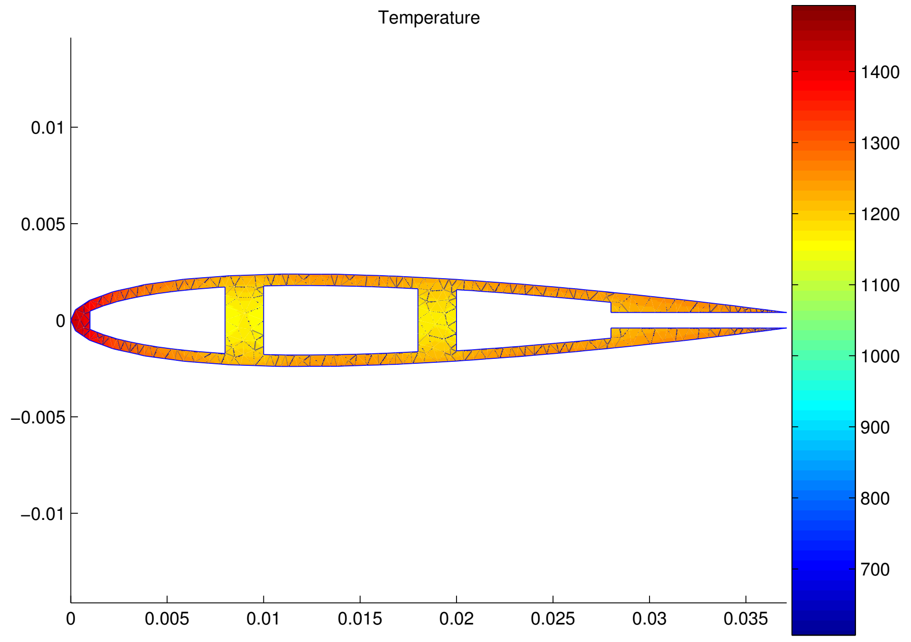


Figure 11: Temperature distribution in blade at $t = 4$ seconds.