Exploratory Report on the Finite Element Method: From Rayleigh-Ritz to 1D 4th-Order BVPs

Riley Davis

05/16/2019

Massachusetts Institute of Technology
Department of Mechanical Engineering

2.S976 Finite Element Methods for Mechanical Engineers
Professor Anthony Patera

Abstract

This report develops an exploration of the Finite Element Method from the Rayleigh-Ritz Method through more complex 1D problems. First, the Rayleigh-Ritz Method is explored along with the utilization of basis functions and the differentiation between Neumann-Robin and Dirichlet boundary conditions. In Chapter Two, 2nd Order models of heat transfer problems serve as a first exploration of the Finite Element Method in full and the considerations of error analysis. Chapter Three uses and discussion and model of flipping and cooking hamburgers to explore the class of time-dependent Finite Element problems that employ the Finite Difference-Finite Element Method. Chapter Four moves to bending cases with 4th order equations, using the specific example of determining shapes and corresponding resonant frequencies in xylophone bars. Lastly, Chapter Five applies the Finite Element Method to self-buckling problems in the context of a contest to optimize a tower shape to achieve a max height without buckling subject to certain constraints. As a note, figure and table numbers are local to each chapter, and any mention of lecture or a listing of derived equations comes directly from the MIT subject 2.S976 Lecture Notes (AT Patera, 2019) on MIT Stellar.

Chapter One – The Rayleigh-Ritz Method

1. Introduction

In this chapter, we explore the fundamentals of Finite Element Analysis via the Rayleigh-Ritz Method. This method is explored and validated through the development of code to solve two different models of problems. Both problems are from the area of thermodynamics, but are differentiated by their boundary conditions – either Neumann/Robin or Dirichlet. Through the testing of the developed code with different parameters and numbers of Rayleigh-Ritz ψ functions, a discussion of the principles of Rayleigh-Ritz and the effects of problem parameters on the Minimization Process can be presented.

1.1 The Rayleigh-Ritz Method

The Rayleigh-Ritz Method is an approximation method that given a problem and its corresponding energy functional (Π), finds an approximate solution (u^{RR}) such that the quantity $\Pi(u^{RR})$ is less than any other possible $\Pi(\omega)$. The function, ω, is any other candidate function that meets certain problem-specific conditions, and the method develops u^{RR} as a sum of unknown coefficients, ${\alpha_i}^{RR}$, multiplied by known basis functions, ψ_i. In this chapter, the problems explored involve heat transfer, and their corresponding solutions are functions that describe the temperature along a thermal fin.

Regardless of candidate function requirements and Π functional formulation, the Rayleigh-Ritz method requires the input of a set of basis functions. These functions can also be defined as shape functions, for they offer different "shapes" for the model to weight together in order to develop an approximate solution u^{RR}. The number of basis functions is denoted as n^{RR}, and a corresponding list of basis functions could look like the following: $\{\psi_1 \in X, \psi_2 \in X, \dots \psi_{RR} \in X\}$. The expansion of the Rayleigh-Ritz approximation, u^{RR}, is the sum of the weightings of the individual ψ functions: $u^{RR} = \sum_{i=1}^{n^{RR}} \alpha_i \psi_i(x) \; with \; \underline{\alpha}^{RR} = \left({\alpha_1}^{RR} \; {\alpha_2}^{RR} \dots \; {\alpha_{n^{RR}}}^{RR}\right)$. The guiding principle of the RR Method is that "lower is better," with the method seeking to find a vector $\underline{\alpha}^{RR}$such that $\Pi\left(\sum_{i=1}^{n^{RR}} {\alpha_i}^{RR} \psi_i\right) < \Pi(\sum_{i=1}^{n^{RR}} \alpha_i \psi_i)$, meaning that $\Pi(u^{RR})$ is less than the energy functional evaluated with any other combination of α_i's and ψ_i's.

The Minimization Proposition, which I will not re-prove in this paper, states that the energy functional evaluated at the exact solution to the problem plus a perturbation ($u + v$) results in three terms, of which only the third (E_{III}) is non-zero. This third term is a norm that can evaluate the accuracy of a proposed candidate function. As a result of the Minimization Proposition proof, we can state the Comparison Proposition: given two approximations to u ($\widetilde{u_1} \; and \; \widetilde{u_2}$) such that $\Pi(\tilde{u}_1) < \Pi(\tilde{u}_2)$ then $E_{III}(u - \tilde{u}_1) < E_{III}(u - \tilde{u}_2)$. This allows us to state, in general, that $\widetilde{u_1}$ is a better solution than $\widetilde{u_2}$ in the E_{III} norm if $\Pi(\tilde{u}_1) < \Pi(\tilde{u}_2)$. This is extremely valuable, as it has the practical implication that the evaluation of $\Pi(\tilde{u}_i)$ does not require knowledge of u, the exact solution to the problem. In the next section, I will describe the two models that are explored in this chapter as well as the variations of the Rayleigh-Ritz Method used for each depending on their boundary conditions.

1.2 Considered Models

1.2.1 Model I

The first considered model, Model I, describes a quasi-1D heat conduction in a conical frustum of length L and initial radius $R_{0,}$ that is insulated on the lateral surfaces and has heat flux and heat transfer coefficient boundary conditions on the left and right surfaces, respectively. These boundary conditions correspond to Neumann/Robin boundary conditions, and this type of problem can be solved using the standard Rayleigh-Ritz Method. The following equations describe the model and its boundary conditions:

$$-k\frac{d}{dx}\left(\pi {R_o}^2\left(1+\ \beta\frac{x}{L}\right)^2\frac{du}{dx}\right)=0\ in\ \Omega \tag{1}$$

$$k\frac{du}{dx}=\ -q_1\ on\ \Gamma_1 \tag{2}$$

$$-k\frac{du}{dx}=\ \eta_2(u-u_\infty)\ on\ \Gamma_2 \tag{3}$$

The value of the temperature at $x=0$ was considered as the output of the problem ($s=u(0)$). We were also supplied with an exact solution to Equation 1 with which to test our codes, which can be found in Appendix A.

For Neumann/Robin boundary conditions, the space of functions over which Π must be minimized is $H^1(\Omega)$, which includes candidate functions (ω) that satisfy the conditions that the integral of the function or its derivative squared must be finite:

$$H^1(\Omega)\left[\int_0^L\omega^2dx<\ \infty,\int_0^L\left(\frac{d\omega}{dx}\right)^2dx\ <\infty\right] \tag{4}$$

The formulation of the energy functional for this problem with respect to a valid candidate function ω can be found in Appendix A.

The Rayleigh-Ritz Approximation for this model follows the general formulation of

$$u^{RR}=\ \sum_{i=1}^{n^{RR}}\alpha_i\psi_i(x)\ with\ \underline{\alpha}^{RR}=\left({\alpha_1}^{RR}\ {\alpha_2}^{RR}\ \dots\ {\alpha_{n^{RR}}}^{RR}\right) \tag{5}$$

The derivation of the Rayleigh-Ritz Approximation of Model I can be found in Section 2.1.1 of this chapter.

1.2.2 Model II

The second considered model describes a right-cylinder thermal fin of length L, cross sectional area A_{cs}, and cross section perimeter P_{cs} that has temperature and zero-flux boundary conditions on the left and right surfaces, respectively. As a temperature is imposed as one of the boundary conditions, this problem is classified as having one Dirichlet boundary conditions (7) and one Neumann/Robin boundary condition (8). This leads to it having a slightly different formulation for finding u^{RR}via the Rayleigh-Ritz Method. The following equations describe this model and its boundary conditions:

$$-kA_{cs}\frac{d^2u}{dx^2}=\eta_3P_{cs}(u-u_\infty)=0\ in\ \Omega \tag{6}$$

$$u=\ u_{\Gamma_1}on\ \Gamma_1 \tag{7}$$

$$-k\frac{du}{dx} = 0 \; on \; \Gamma_2 \tag{8}$$

The heat flux into the frustum at $x = 0$ was considered as the output of the problem ($s = -k\frac{du}{dx}(x = 0)$). We were also supplied with an exact solution to Equation 6 with which to test our codes, which can be found in Appendix B.

For Dirichlet boundary conditions, the space of functions over which Π must be minimized is X^D, affine space. This includes candidate functions (ω) that are within $H^1(\Omega)$, but also meet the essential boundary condition of a set temperature at x = 0, $\omega|_{\Gamma_1} = u_{\Gamma_1}$.

The formulation of the energy functional for this problem with respect to a valid candidate function ω can be found in Appendix B.

The Rayleigh-Ritz Approximation for this model follows a special formulation of basis functions at the beginning of the method. In order to satisfy the affine space condition and imposed temperature boundary condition for the candidate function, $\psi_0 \in H^1(\Omega) \; AND \; \psi_0(x = 0) = 1$. The remaining basis functions must have a value of zero and x = 0 in order to uphold the temperature boundary condition. This results in a Rayleigh-Ritz expansion of

$$u^{RR}(x) = u_{\Gamma_1}\psi_0(x) + \sum_{i=1}^{n^{RR}} \alpha_i{}^{RR}\psi_i(x) \tag{9}$$

The solving for $\underline{\alpha}^{RR}$involves the formation of two sets of matrices for two steps of matrix equation operations to arrive at the final answer. The derivation of the Rayleigh-Ritz Approximation of Model II can be found in Section 2.1.2 of this chapter.

I will now move on to an explanation of the development of the code templates we were provided with to consider the Rayleigh-Ritz approximations of Model I and Model II.

2. Development of Codes

The method I used to modify the template codes given to us were to use pattern matching between the equations that govern the models and the general equation for the energy functional to determine the values of the various constants specific to each model. This allowed me to then use those constants to develop the needed $\underline{A}$ *and* $\underline{F}$ matrices. I then added to the code my derived equations for $\underline{A}, \underline{F}, \underline{\alpha}^{RR}$ *and* Π in order to ultimately calculate u^{RR}.

2.1 Mathematical Derivations

2.1.1 Model I

Based on the comparison of Equation 1 to the general form of the differential equation from lecture:

$$-\frac{d}{dx}\left(\kappa(x)\frac{du}{dx}\right) + \mu(x)u = f_\Omega(x) \; in \; \Omega \tag{10}$$

I was able to determine that for the case of Model I, $\kappa(x) = k\pi {R_0}^2(1 + \beta\frac{x}{L})^2, \mu(x) = 0, and \; f_\Omega(x) = 0$. In order to pattern match the boundary conditions outlined in Equations 2 & 3, I needed to first scale them by $\pi {R_0}^2(1 + \beta\frac{x}{L})^2$ so that I could pattern match to $\kappa(x)$, since only *k* was present in the original boundary condition equations. After scaling both B.C. and evaluating them at either x = 0 or x = L, I was able to determine via pattern matching that $\gamma_1 = 0, f_{\Gamma_1} =$

$q_1\pi {R_0}^2, \gamma_2 = \eta_2\pi {R_0}^2(1+\beta)^2, and\ f_{\Gamma_2} = \eta_2\pi {R_0}^2(1+\beta)^2 u_\infty$. This results in the following energy functional that is valid for Model I:

$$\Pi_1 = \Pi(\omega) = \frac{1}{2}\int_0^L \left[k\pi {R_0}^2(1+\beta)^2\left(\frac{d\omega}{dx}\right)^2\right]dx + \frac{1}{2}[\eta_2\pi {R_0}^2(1+\beta)^2\omega^2(L)] - q_1\pi {R_0}^2\omega(0) - \eta_2\pi {R_0}^2(1+\beta)^2 u_\infty\omega(L) \quad (11)$$

In order to solve for $\underline{\alpha}^{RR}$, it is necessary to solve the matrix equation:

$$\underline{A}\,\underline{\alpha}^{RR} = \underline{F} \quad (12)$$

The standard formulations for $A_{ij}\ and\ F_i$ can be found in Appendices A and B for the respective models. Using the values of constants that I had identified above using pattern matching, the following equations for $A_{ij}\ and\ F_i$ emerged:

$$A_{ij} = \int_0^L \left[k\pi {R_0}^2\left(1+\beta\frac{x}{L}\right)^2\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}\right]dx + \eta_2 k\pi {R_0}^2\left(1+\beta\frac{x}{L}\right)^2\psi_i(L)\,\psi_j(L) \quad (13)$$

$$F_i = q_1\pi {R_0}^2\psi_i(0) + \eta_2\pi {R_0}^2(1+\beta)^2 u_\infty\psi_i(L) \quad (14)$$

To modify RR_2S_sver in order to create rileyd_RR_2S_sver_Model1, I inputted the exact solution we were provided with as well as its derivative, and then wrote out the equations for the elements of the $\underline{A}$ and $\underline{F}$ matrices. I set all constants for the problem at the beginning of the code, except for β which is passed to the function. I also calculated $s = u(0)$, our required output for this problem, and then calculated the error between the Rayleigh-Ritz Approximation of the temperature at x = 0 to the exact solution evaluated at that point.

2.1.2 Model II

For Model II, I compared Equation 10 to the differential equation that describes the model (6) in order to determine the value of each problem specific coefficient.

I was able to determine that for Model II, $\kappa(x) = kA_{cs}, \mu(x) = \eta_3 P_{cs}, and\ f_\Omega(x) = \eta_3 P_{cs} u_\infty$. The N/R boundary condition needed to be scaled by A_{cs} in order to be consistent in our definition of $\kappa(x)$, but because the right hand side of Equation 8 equals zero, the scaling mathematically does not end up having an effect on the solution. After scaling both B.C. and evaluating them at either x = 0 or x = L, I was able to determine via pattern matching that $\gamma_2 = 0, f_{\Gamma_2} = 0, and\ u = u_{\Gamma_1}$. This results in the following energy functional that is valid for Model II:

$$\Pi_2(\omega) = \frac{1}{2}\int_0^L \left[kA_{cs}\left(\frac{d\omega}{dx}\right)^2 + \eta_3 P_{cs}\omega^2\right]dx - \int_0^L[\eta_3 P_{cs} u_\infty\omega]dx \quad (15)$$

Because of the extra condition on the ψ functions to have $\psi_0(0) = 1$ and all other $\psi_i(0) = 0$, first two matrices, $\underline{\tilde{A}}$ and $\underline{\tilde{F}}$, are formed using the identified coefficients in the same manner as in Model I:

$$\tilde{A}_{ij} = \int_0^L \left[kA_{cs}\frac{d\psi_i}{dx}\frac{d\psi_j}{dx} + \eta_3 P_{cs}\psi_i\psi_j\right]dx \quad (16)$$

$$\tilde{F}_i = \int_0^L[\eta_3 P_{cs} u_\infty\psi_i]dx \quad (17)$$

Then $\underline{A}, \underline{F}, and\ \underline{b}$ are extracted from the ~ matrices. In order to solve for $\underline{\alpha}^{RR}$, it is necessary to solve the matrix equation:

$$\underline{A}\,\underline{\alpha}^{RR} = \underline{F} - u_{\Gamma_1}\underline{b} \qquad (18)$$

Then, $\underline{\tilde{\alpha}}^{RR}$ is formed from $u_{\Gamma_1} and\ \alpha^{RR}$, and finally Π is evaluated using $\underline{\tilde{\alpha}}^{RR}$, $\underline{\tilde{A}}$ and $\underline{\tilde{F}}$. To modify RR_2S_sver in order to create rileyd_RR_2S_sver_Model2, I inputted the exact solution we were provided with as well as its derivative, and then wrote out the equations for the elements of the $\underline{\tilde{A}}$ and $\underline{\tilde{F}}$ matrices. I set all constants for the problem at the beginning of the code, except for η_3 which is passed to the function. I added code to extract $\underline{A}, \underline{F}, and\ \underline{b}$ and then form $\underline{\tilde{\alpha}}^{RR}$. I also calculated $s = -k\frac{du}{dx}(x = 0)$, our required output for this problem, and then calculated the error between the Rayleigh-Ritz Approximation of the heat flux at x = 0 to the exact solution evaluated at that point. Instead of taking the numerical derivative of u^{RR}, I simply took the derivative of the Rayleigh-Ritz Approximation (9), which only involves taking the derivatives of the individual ψ functions and multiplying them by their constant coefficients ${\alpha_i}^{RR}$.

3. Testing and Results

3.1 Exactinclude

The first check to make sure that the code that I modified was working, was to run the code using a known solution to the model to confirm that the known correct answer is returned. This was done using `exactinclude`, which passes the exact solution as $\psi_1 and\ \psi_2 = x$. Below in Table 3.1.1 are the results of running this command with varying values of β for Model I.

β	$\underline{\alpha}^{RR}$	$\Pi(u^{RR})$	Error
1	[1;0]	-37.1061	1.24E-15
2	[1;0]	-82.2905	9.81E-15
3	[1;0]	-145.5981	6.85E-15
100	[1;0]	-92297.23	0
10000	[1;0]	-904959649.7725	9.33E-15

Table 3.1.1 – Model I `exactinclude`

From this data, I can tell that the model is working because regardless of the value of β, the vector of Rayleigh-Ritz coefficients had a value of one multiplying by ψ_1 and a value of zero multiplying by ψ_2. This means that my code is returning the exact solution with a weighting of one, and not including ψ_2 at all.

For Model II the results are slightly different as seen in Table 3.1.2 below:

Table 3.1.2 – Model II `exactinclude`

In the case of Model II, the desired outcome is to have ψ_0 multiplied by u_{Γ_1} and ψ_1 multiplied by a coefficient of zero, since ψ_0 is defined as the exact solution divided by u_{Γ_1}. In my code for Model II, $u_{\Gamma_1} = 50$, and from the data in Table 3.1.2 it is clear that $u^{RR}$is calculated as $u^{RR} = 50 \times \psi_0 + 0 \times \psi_1$ which results in the correct exact solution being returned. Because of this, I can gain confidence that my codes are working correctly.

3.2 Constlinquad

Using `constlinquad`, up to three unique ψ functions can be passed to the Rayleigh-Ritz code to be multiplied by the best possible coefficients that the method determines. Figures 3.2.1 and 3.2.2 show the initial ψ functions that are passed using `constlinquad` and an example of the final weightings with coefficients that the Rayleigh-Ritz approximation calculates.

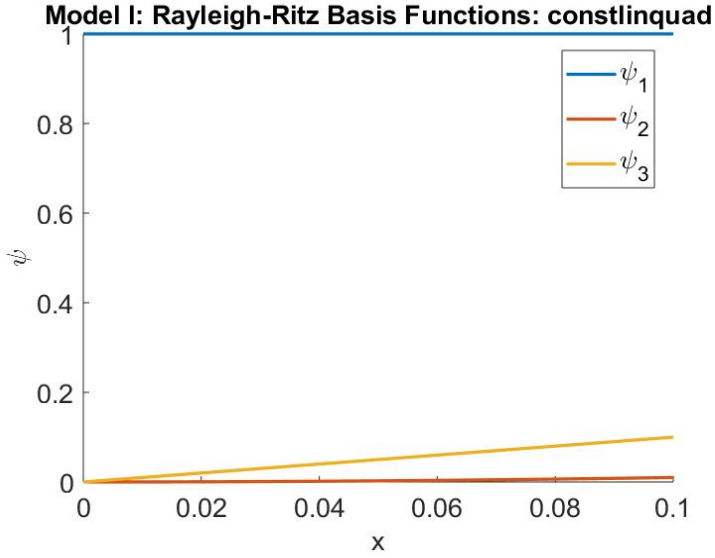


Figure 3.2.1 constlinquad Basis Functions

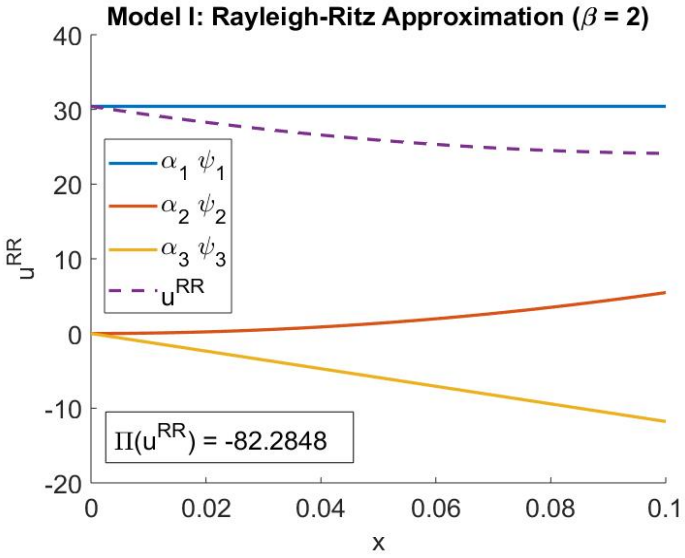


Figure 3.2.2 Example of Rayleigh-Ritz coefficient weighting

In using both Model I and Model II, the values of $\Pi(u^{RR})$ approached the value $\Pi(u_{exact})$ with increasing n^{RR} (See Table 3.3.1). This makes sense, as we discussed in class how the addition of first and

η_3	$\underline{\alpha}^{RR}$	$\Pi(u^{RR})$	Error
1	50;0	0.095529	0
80	50;0	-9.636	1.30E-16
10000	50;0	-5281.9958	1.58E-16

second order polynomial ψ functions tends to increase the accuracy with which the Rayleigh-Ritz Method can approximate the solution. We also mentioned that going further to third and higher order

polynomials does not add much value as these basis functions no longer introduce novel "shapes," but unfortunately I did not have time to test higher order ψ functions for this chapter.

3.3

	Model I		Model II	
n^{RR}	$\Pi(u^{RR})$	Error	$\Pi(u^{RR})$	Error
1	-36.9491	0.29197	0.1	1
2	-37.0251	0.15069	0.096642	0.5
3	-37.1045	0.00301	0.095529	0.000999
Exactinclude	-37.1061	1.24E-15	0.095529	0

Constlinquad vs. Exactinclude

The Rayleigh-Ritz recipe finds the coefficients, $\underline{\alpha}^{RR}$, such that

$$\Pi\left(\sum_{i=1}^{n^{RR}} \alpha_i{}^{RR}\psi_i\right) < \Pi(\sum_{i=1}^{n^{RR}} \alpha_i\psi_i)$$

meaning that $\Pi(u^{RR})$ is less than the energy functional evaluated with any other combination of α_i's and ψ_i's. In order to check this, we can compare the value of $\Pi(u^{RR})$ when using `constlinquad`, to the value of $\Pi(u^{RR})$ when using `exactinclude` (passing the exact solution as $\psi_{1\ or\ 0}$), which results in the lowest possible value of $\Pi(u^{RR})$.

Table 3.3.1 – `constlinquad` compared to `exactinclude` (β = 1, η_3 = 1)

For all values of n^{RR}, the energy functional values are close to the lowest possible value for the given problem and parameters, which is the value of the energy functional when using `exactinclude`. Increasing the number of ψ functions passed to the code, decreases the error between the RR approximation and the exact solution, which in turn brings the value of the energy functional closer to the absolute minimum. This data also leads to the conclusion that increasing the number the number of ψ functions (for this particular set of basis functions, as discussed in Section 3.2) results in a better approximation of u^{RR}. This is based on the Minimization Principle that states if $\Pi(\tilde{u}_1) < \Pi(\tilde{u}_2)$ then $\tilde{u}_1$ is a better approximation of u than $\tilde{u}_2$, using the E_{III} norm. Beyond confirming the Minimization Principle, these results also give confidence that both the codes for `exactinclude` and `constlinquad` are working well because the produced similar results for $\Pi(u^{RR})$.

3.4 Changing β and η_3

Using Model I, I varied the value of β across the following values [1, 2, 3, 100, 10000]. At above three, the values of $\Pi(u^{RR})$ became so close to $\Pi(u_{exact})$ I that I realized I should focus on varying β on the order of 1. When using one basis function, I found that the difference between the exact solution and the Rayleigh-Ritz approximation (the error of the output) decreased with increasing β. This wasn't easily observable from the graphs, but error dropped from 0.29 to 0.17 using n^{RR} =1 and moving from β =1 to β = 3. A similar drop occurs when n^{RR} = 2 (0.15 -> 0.12), but error decreases slightly when n^{RR} = 3 (0.003 -> 0.02). The parameter β affects the change in cross sectional area along the length of the frustum in Model I. The larger β is, the greater the increase in cross sectional area per length of the frustum. I am not sure how this makes the solution easier to approximate using the Rayleigh-Ritz Method.

Using Model II, I varied the value of η_3 between 1, 80, and 10000. This results in magnitudes of the parameter μ_0 that correspond to natural convection, forced convection, and change of phase, respectively. I found that increasing η_3/μ_0 corresponds with an increase in the difference between the exact solution and the Rayleigh-Ritz Approximation. These coefficients relate to the convective heat loss on the outer surface of the fin. I assume that as this term increases in magnitude, the rate of convection increases and varies more over the surface, making it harder to approximate.

Chapter Two – The FE Method for 1D 2nd-Order SPD BVPs

1. Introduction

In this chapter, we explore the fundamentals of Finite Element Analysis which builds off of the Rayleigh-Ritz Method explored in Chapter 1. The FE Analysis is explored using provided software and validated through the development of code to solve three different models of problems. All problems are from the area of thermodynamics, but are differentiated by their boundary conditions – either Neumann/Robin or Dirichlet – and the presence or absence of heat transfer coefficients. Testing of these models leads to a discussion of the principles of the Finite Element Method in 1D and the interpretation of whether solutions are converging or can be validated using error estimators.

1.2 Considered Models

1.2.1 Model I

The first considered model, Model I, describes a quasi-1D heat conduction in a conical frustum of length L and initial radius $R_{0,}$ that is insulated on the lateral surfaces and has heat flux and heat transfer coefficient boundary conditions on the left and right surfaces, respectively. These boundary conditions correspond to Neumann/Robin boundary conditions, and this type of problem can be solved using the standard Finite Element Method. The following equations from lecture notes describe the model and its boundary conditions:

$$-k\frac{d}{dx}\left(\pi {R_o}^2\left(1+\ \beta\frac{x}{L}\right)^2\frac{du}{dx}\right)=0\ in\ \Omega \tag{1}$$

$$k\frac{du}{dx}=\ -q_1\ on\ \Gamma_1 \tag{2}$$

$$-k\frac{du}{dx}=\ \eta_2(u-u_\infty)\ on\ \Gamma_2 \tag{3}$$

The value of the temperature at $x = 0$ was considered as the output of the problem ($s = u(0)$). We were also supplied with an exact solution to Equation 1 with which to test our codes, which can be found in Appendix A.

The FE Analysis solution for this model follows the general formulation of:

$$u_h(x)=\ \sum_{i=1}^{n_{node}} u_{h_i}\varphi_i(x) \tag{5}$$

Where n_{node} is the number of nodes in the finite element mesh and φ_1 are the P_1 shape functions used.

1.2.2 Model II

The second considered model describes a right-cylinder thermal fin of length L, cross sectional area A_{cs}, and cross section perimeter P_{cs} that has temperature and zero-flux boundary conditions on the left and right surfaces, respectively. As a temperature is imposed as one of the boundary conditions, this problem is classified as having one Dirichlet boundary conditions (7) and one Neumann/Robin boundary condition (8). This leads to it having a slightly different formulation for finding u_h via the Finite Element Method. The following equations describe this model and its boundary conditions:

$$-kA_{cs}\frac{d^2u}{dx^2} = \eta_3 P_{cs}(u - u_\infty) = 0 \; in \; \Omega \tag{6}$$

$$u = u_{\Gamma_1} on \; \Gamma_1 \tag{7}$$

$$-k\frac{du}{dx} = 0 \; on \; \Gamma_2 \tag{8}$$

The heat flux into the frustum at $x = 0$ was considered as the output of the problem ($s = -k\frac{du}{dx}(x = 0)$). We were also supplied with an exact solution to Equation 6 with which to test our codes, which can be found in Appendix B.

The finite element solution for this model follows a special formulation of basis functions at the beginning of the method. This results in a finite element solution formulation of:

$$u_h(x) = u_{\Gamma_1}\varphi_1(x) + \sum_{i=1}^{n_{node}} u_{h_i}\varphi_i(x) \tag{9}$$

The solving for $\underline{u_h}$ involves the formation of two sets of matrices for two steps of matrix equation operations to arrive at the final answer.

1.2.3 Ch2_Model_Mine

For our third model, we developed a simple model that imposes N/R boundary conditions at both the left and right ends of the domain, with heat transfer coefficients that are both non-zero and positive. I chose to develop a model for heat transfer through a wall with uniform conductivity and convection happening on each side. The diagram below describes the situation.

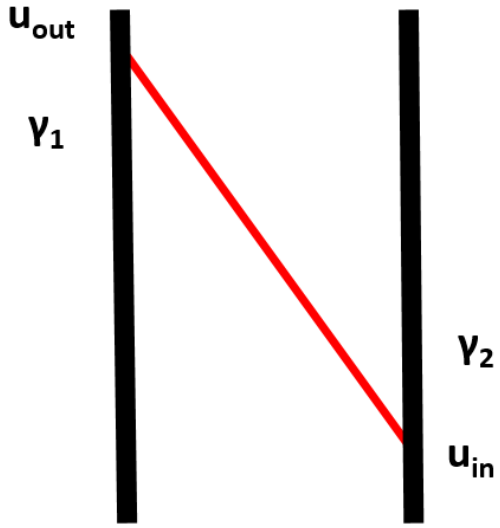


Figure 1: Visual Depiction of Ch2_Model_Mine. Temperatures on either side of a solid wall with conductivity *k*, are represented by u_{in} and u_{out}. Heat Transfer Coefficients γ_1 and γ_2 are non-zero and positive.

The equations that describe this model, as well as the assumed linear solution for *u*, are written below:

$$-kA_{cs}\frac{d^2u}{dx^2} = 0 \; in \; \Omega \tag{10}$$

$$kA_{cs}\frac{du}{dx} = \gamma_1 u - \gamma_1 u_{out} \; on \; \Gamma_1 \tag{11}$$

$$-kA_{cs}\frac{du}{dx} = \gamma_2 u - \gamma_2 u_{in} \; on \; \Gamma_2 \tag{12}$$

$$u = B\left(\frac{x}{L}\right) + C \tag{13}$$

The value of the temperature at x = 0 was considered as the output to this problem. In order to test the code, we provided `run_uniform_refinement` with the exact linear solution to the problem by solving for B and C in Equation 13, by using Equations 10, 11, and 12. This resulted in the following values:

$$B = \frac{\gamma_2(u_{in} - u_{out})}{1 + \frac{kA_{cs}}{\gamma_2 L} + \frac{kA_{cs}}{\gamma_1 L}} \tag{14}$$

$$C = \frac{kA_{cs}}{\gamma_1 L}B + u_{out} \tag{15}$$

2. Summary of Finite Element Method

The Finite Element Method uses a similar sequence of steps as the Rayleigh-Ritz Method to solve complex physical models in an elemental way. After a model is describe by a differential equation and appropriate boundary conditions, the different constants or functions that are uniform across different models can be determined via pattern matching. This then allows for the formation of elemental matrices that do not yet take the boundary conditions of the model into account. Their formulations are as follows:

$$A_{ij}{}^N = \int_0^L \left[\kappa(x)\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx} + \mu(x)\varphi_i\varphi_j\right]dx \tag{16}$$

$$F_i{}^N = \int_0^L f_\Omega(x)\varphi_i dx \tag{17}$$

Next the relevant boundary conditions are applied – the example model I will use to discuss this has a N/R boundary condition on Γ_2 and a Dirichlet boundary condition on Γ_1. First, the $\underline{A}^N and \; \underline{F}^N$ have the N/R boundary condition added:

$$\tilde{A}_{ij} = \int_0^L \left[\kappa(x)\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx} + \mu(x)\varphi_i\varphi_j\right]dx + \gamma_2\varphi_i(L)\varphi_j(L) \tag{18}$$

$$\tilde{F}_i = \int_0^L f_\Omega(x)\varphi_i dx + f_{\Gamma_2}\varphi_i(L) \tag{19}$$

If this problem did not have a Dirichlet condition, a N/R boundary condition would be added on for Γ_1 and $\underline{\tilde{A}} = \underline{A}, \underline{\tilde{F}} = \underline{F}$. The matrix equation $\underline{A}\,\underline{u}_h = \underline{F}$ could then be solved directly for the FE solution $\underline{u}_h$. Since there is a Dirichlet condition, $\underline{A} = \underline{\tilde{A}}(2{:}\,end, 2{:}\,end), \underline{u}_h{}^0 = \underline{u}_h(2{:}\,end), \; \underline{F} = \tilde{F}(2{:}\,end)$, and then $\underline{A}\,\underline{u}_h{}^0 = \underline{F} - u_{\Gamma_1}\underline{b}$ can be solved for the finite element solution. In the actual code implementation of the FE method, everything is mapped to a reference element via quadrature points.

3. Discussion of Success of Implementation

3.1 Convergence for Ch1_Model_II

After implementing `Ch1_Model_II` in `library_of_models` and calling the model in `run_uniform_refinement`, the outputted figures were analyzed to determine if u_h was converging to u which also indicated that the implementation was correct. Figures 2 and 3 show the output of Mesh 0 and Mesh 6 for `Ch1_Model_II` for both u and $\frac{du}{dx}$. With Mesh 0, it is clear that both the temperature field and its derivative are not being approximated well, as the individual shape functions (albeit weighted) can be seen in the FE Analysis plot of u_h, while the FE calculated derivative struggles to capture the large initial slope of the temperature derivative. By Mesh 6, both u and $\frac{du}{dx}$ appear to be approximated exactly by the FE method. As a second check, I looked at the error estimates and actual error calculations in different norms between the FE solution and exact solution, which can be seen in Figure 4. The error in each norm ultimately decreases in parallel to the expected trendline, which also gives confidence that the FE solution is converging successfully to the exact solution for `Ch1_Model_II`. Lastly, looking at the map of the matrix that shows the location of non-zero elements, it is clear that it is tridiagonal which is expected and also points to correct implementation.

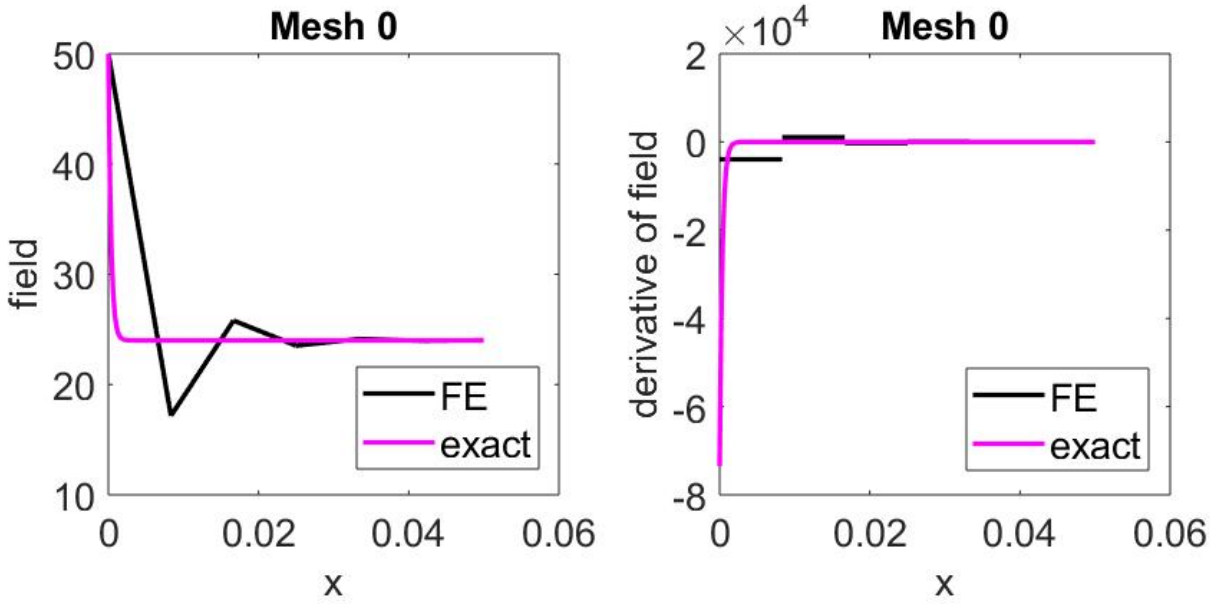


Figure 2: Mesh 0 for `Ch1_Model_II`. Temperature on left, derivative on the right.

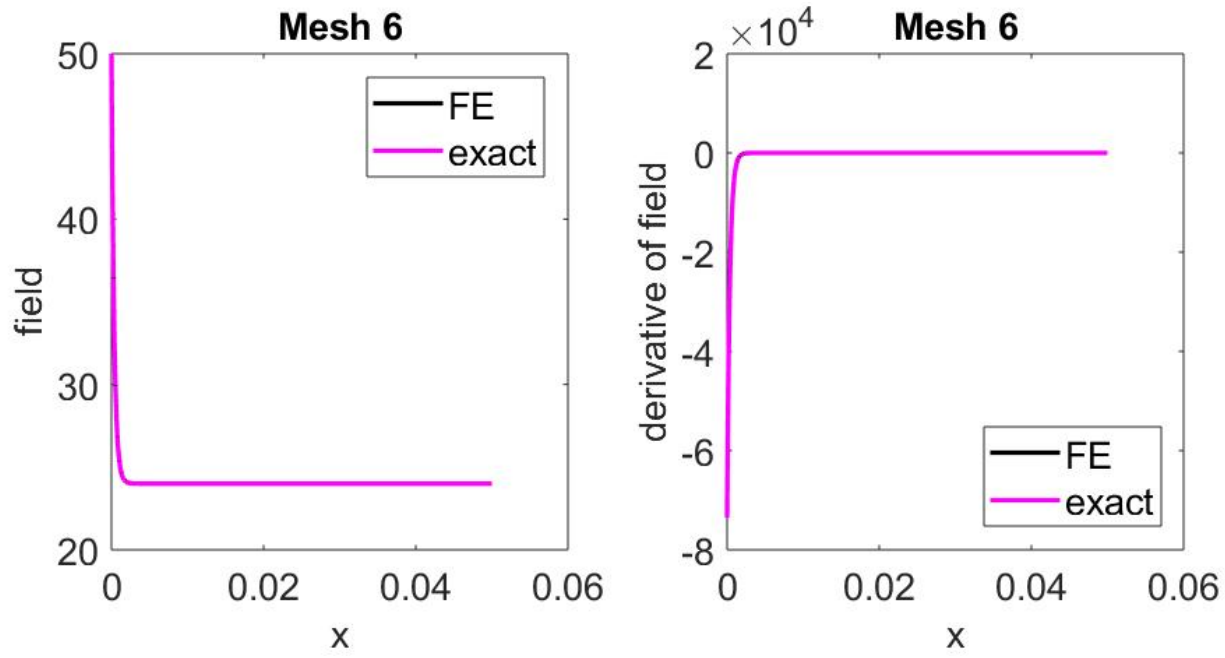


Figure 3: Mesh 6 of `Ch1_Model_II`. Temperature on left, derivative on the right.

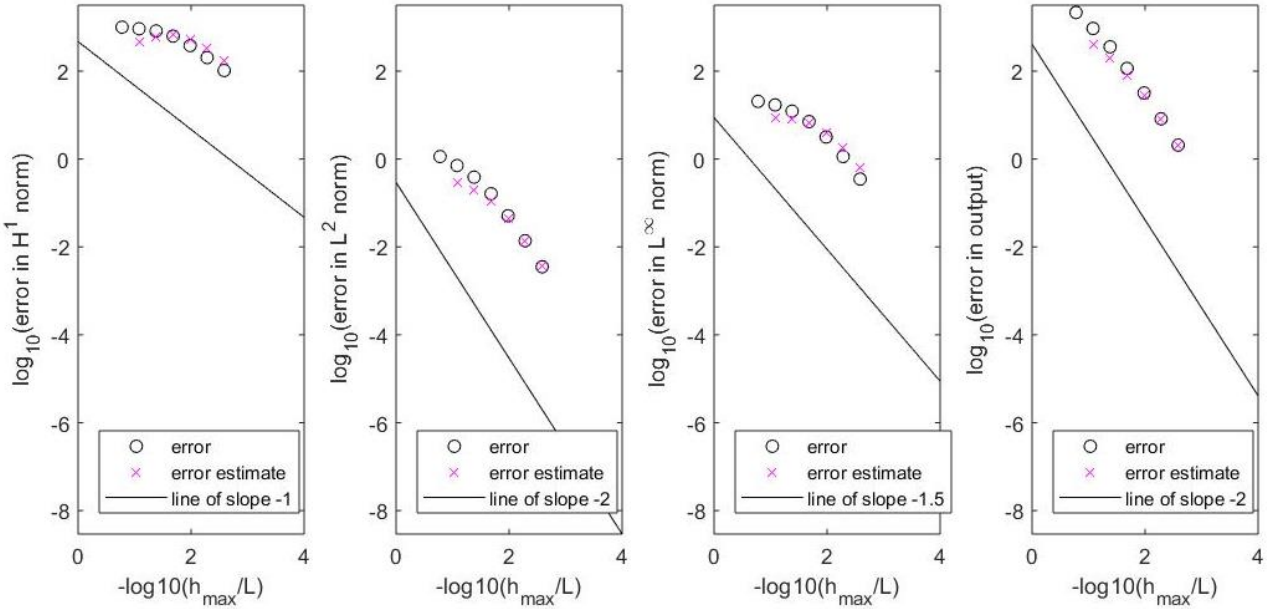


Figure 4: Estimated and Actual Error over six meshes for `Ch1_Model_II`

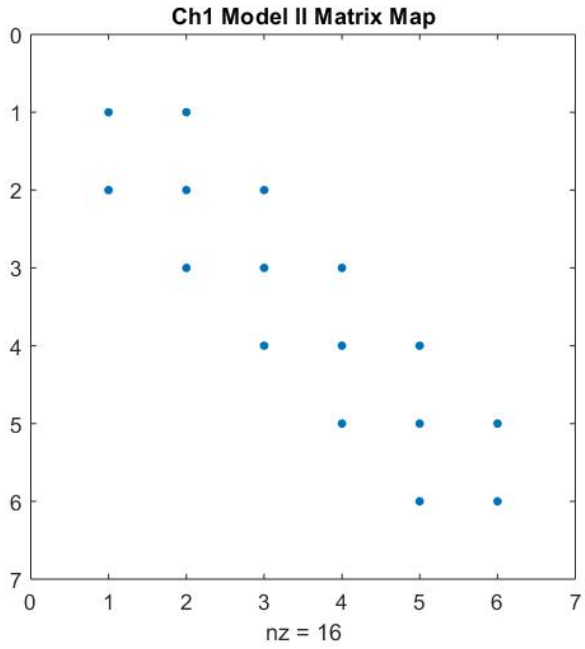


Figure 5: Matrix Element Map of `Ch1_Model_II` Matrix A

3.2 Confidence in Implementation of form_elem_mat_sver

Unfortunately, the convergence of u_h to u does not guarantee that the code I edited in `form_elem_mat_sver` is bug-free for all possible instances of $\mu(x)$. In Section 3.1, I discussed my confidence in my implementation of `form_elem_mat_sver` through testing with `Ch1_Model_II`, which has $\mu(x)$ as a constant, and therefore does not test the case where $\mu(x)$ is a function of x. Solving a problem where $\mu(x)$ is not a constant will end up being complicated and messy, therefore the "method of manufactured solutions" should be utilized. I will consider a similar formulation to `Ch1_Model_II` (see Equations 6-8), but replace the $\mu(x)$ term ($\eta_3 P_{cs}$) with a linear function of x ($\eta_3(\sigma\frac{x}{L})$):

$$\Omega \equiv (0, L),\ \ \Gamma_1 = \{0\},\ \ \Gamma_1 = \{L\}$$

$$-kA_{cs}\frac{d^2u}{dx^2} = \eta_3\sigma\frac{x}{L}(u - u_\infty) = 0\ in\ \Omega \qquad (20)$$

$$u = u_{\Gamma_1} on\ \Gamma_1 \qquad (21)$$

$$-k\frac{du}{dx} = 0\ on\ \Gamma_2 \qquad (22)$$

Then a smooth solution can be arbitrarily chosen for the temperature u – in this example, $u = x^3$ is used. After "manufacturing" a solution for u, $f_\Omega(x)\ and\ f_{\Gamma_2}$ can all be calculated directly:

$$f_\Omega(x) = -\frac{d}{dx}\left(kA_{cs}\frac{du}{dx}\right) + \eta_3\sigma\frac{x}{L}(u - u_\infty)$$

$$f_\Omega(x) = -\frac{d}{dx}(kA_{cs} * 3x^2) + \eta_3\sigma\frac{x}{L}(x^3 - u_\infty) = 6kA_{cs}x + \eta_3\sigma\frac{x}{L}(x^3 - u_\infty)$$

$$f_{\Gamma_2} = \left(-kA_{cs}\frac{du}{dx}\right)(x = L) = -k3A_{cs}L^2$$

Now using these developed equations and manufactured constants, the FE method can be used to solve this problem for u_h. Using this manufactured model with a non-constant $\mu(x)$ that is dependent on x, the robustness of `form_elem_mat_sver` can be tested using a known solution.

3.3 Confidence in Implementation of impose_boundary_cond_sver

Testing of modified code continued with `Ch1_Model_I` and `Ch2_Model_Mine` focused on `impose_boundary_cond_sver`. As can be seen in Figures 6 and 7, there is visible confirmation that the FE solution is converging for both models, as the FE solution can be seen to approach the exact solution over the iterative meshes. In `Ch2_Model_Mine`, the FE method is able to approximate the solution on the first mesh iteration because of its linear nature.

Using `Ch2_Model_Mine` provides greater implementation confidence than `Ch1_Model_I` because the `impose_boundary_cond_sver` code's job is to impose the boundary conditions of the supplied model. In `Ch1_Model_1`, γ_1 is equal to zero, while γ_2 is non-zero. If testing was only done with `Ch1_Model_I`, any errors in the code to apply the boundary condition that utilizes γ_1 would not be seen. In `Ch2_Model_Mine`, both γ values are non-zero, so any errors in code imposing the boundary conditions would be visible in the solution because both γ's are in play.

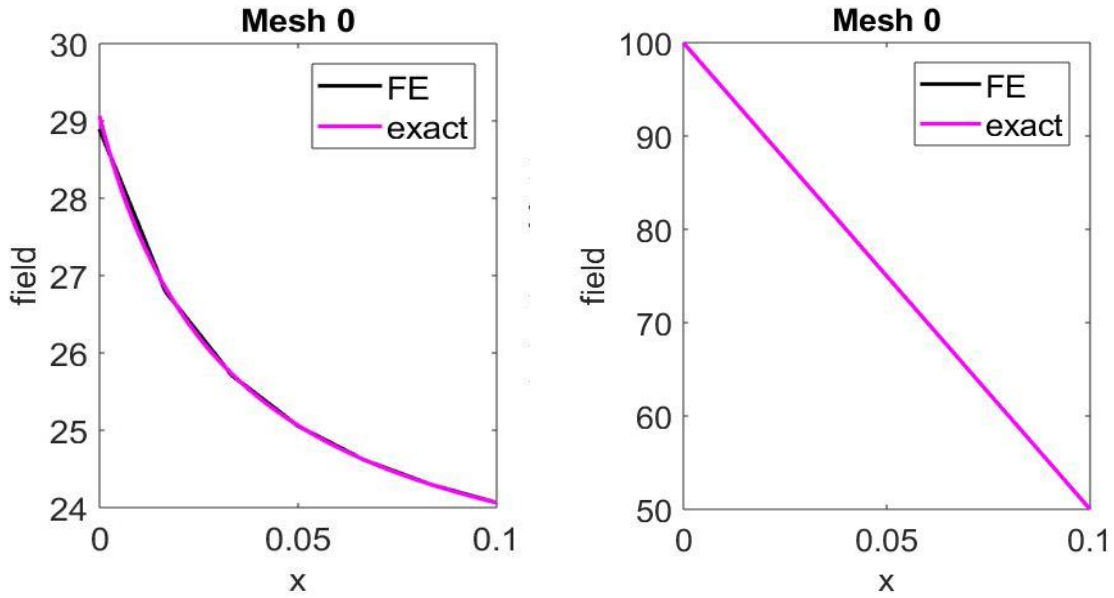


Figure 5: Mesh 0 for `Ch1_Model_I` (left) and `Ch2_Model_Mine` (right)

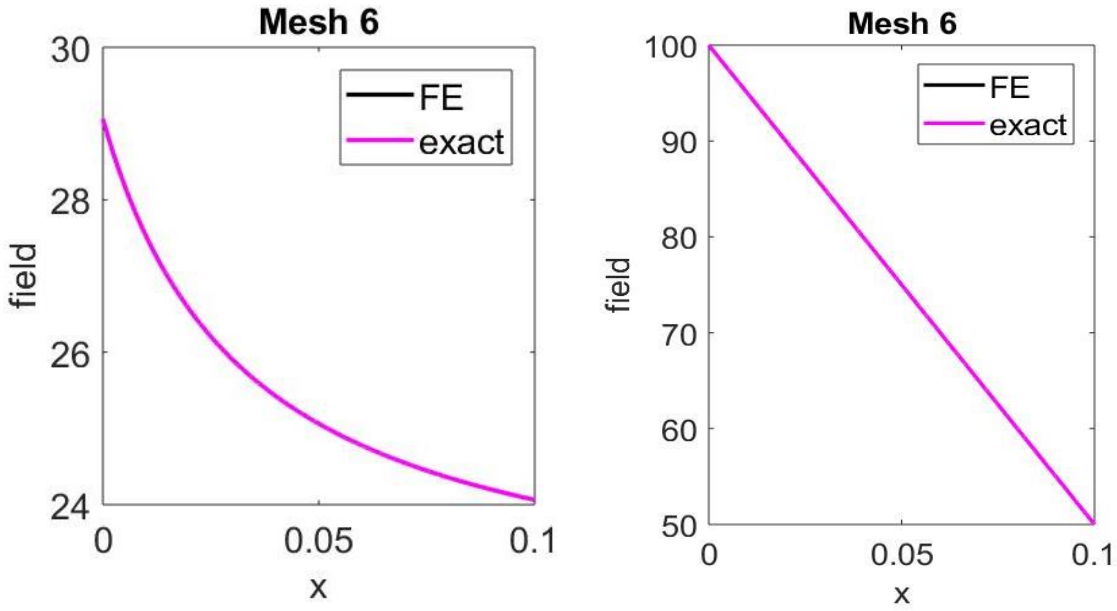


Figure 6*:* Mesh 6 for `Ch1_Model_I` (left) and `Ch2_Model_Mine` (right)

3.4 Determining Convergence for a Model_X

As a sanity check of the understanding of how one can be sure convergence is achieved, an unknown `Model_X` is considered for which the exact solution *u* is not known. When the FE code is run for `Model_X`, it is observed that for sufficiently small *h* the extrapolation error estimators converge at the anticipated rates in all norms. At first glance, it would appear that the conclusion that u_h is converging to the exact solution of `Model_X` is valid. This cannot be concluded absolutely, because there may have been an error in the model definition of `Model_X`, or it could be possible that a different model from `Model_X` was called in `run_uniform_refinement`. This would mean that the FE code is solving a different problem than the one thought to be being solved, and although the results are converging, the results are not for the correct problem!

3.5 Accuracy and Verification of Numerical Specification

`Ch1_Model_II` was considered over a sequence of 9 meshes with p = 1, and the code was run assuming no exact solution was available (i.e. changing `probdef.exact_available` from `true` to `false`). Error estimates in two norms were assessed without the use of a known solution and predictions of the upper bounds of the error estimates were made. The L^∞ norm, which is the maximum of $|u(x) - u_h(x)|$ over all x in Ω, was considered first and the coarsest mesh such that $\|u - u_h\|_{L^\infty(\Omega)} < 1.00$ was sought. First, I confirmed that the error estimate in the H^1 norm was decreasing over the relevant meshes (Meshes 4+). By looking at Figure 7, which shows the results of running an FE Analysis of `Ch1_Model_II` with no known exact solution, the coarsest mesh with a negative value for $\log_{10}(error\ in\ L^\infty norm)$ is Mesh 7 (marked with a Datatip). This corresponds to an estimated upper error bound of $10^{-0.1967} = 0.63577$ for the error in the L^∞ norm. Next, error estimates for the error in the output were

explored for Mesh 5. In the error in the output norm, Mesh 5 is also marked with at Datatip in the farthest right plot in Figure 7. The value of the estimated error in the output in Mesh 5 is $10^{1.443} = 27.7332$. I chose to check my estimations initially without using a safety factor.

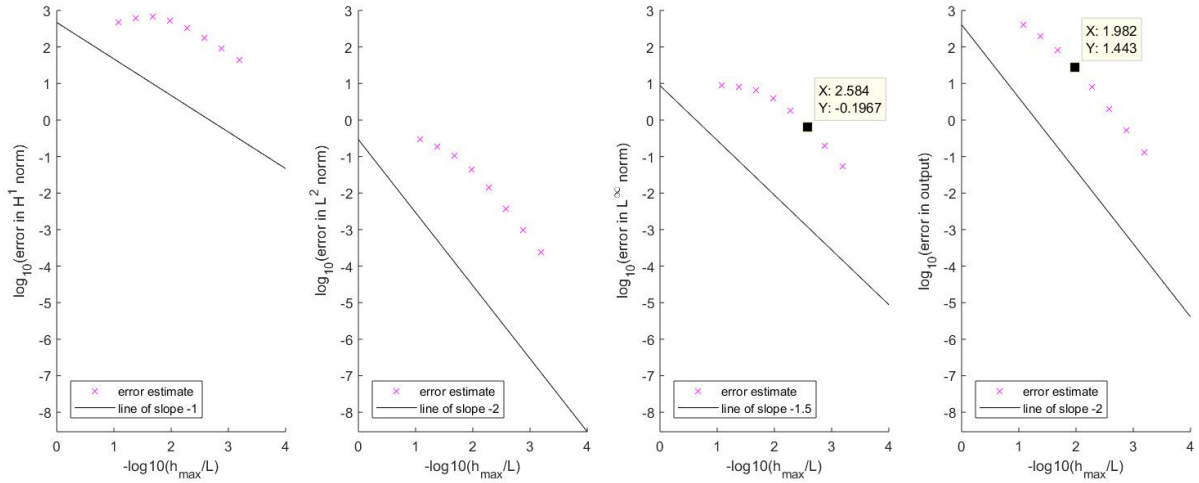


Figure 7: Error Estimates in Four Norms for `Ch1_Model_II`

After making my estimations, I then changed `probdef.exact_available` back to `true` and ran `run_uniform_refinement` again. The error norm plots from this can be seen in Figure 8. For the L^∞ norm, Mesh 7 was confirmed to be the coarsest mesh to have $\|u - u_h\|_{L^\infty(\Omega)} < 1.00$. The exact error (since now u was known) was calculated to be $10^{-0.4563} = 0.349704$. This is less than what I had estimated, so the proposed upper bound of 0.63577 is not violated. For the error in the output in Mesh 5, the exact error was found to be $10^{1.503} = 31.842$, which is larger than the predicted upper limit of 27.7332. This example shows that a safety factor is often needed when using FE analysis. If I had used a SF of 2 when predicting the upper error limit of the output, I would have predicted an upper limit of 55.4664, which is 1.7 times the actual error. In the case of the L^∞ norm, it happened that no safety factor was needed in this case to predict the upper error limit. The predicted upper limit is 1.8 times the calculated error. Regardless of this specific case, it is a sound practice to always use a SF when utilizing FE Analysis results.

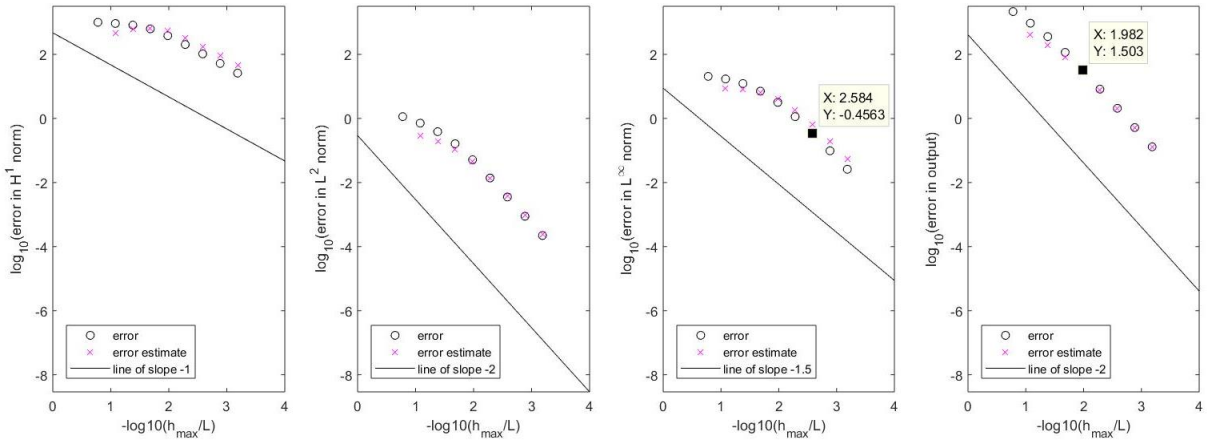


Figure 8: *Exact Errors in Four Norms for* `Ch1_Model_II`

Chapter Three – The FD-FE Method for the 1D Heat Equation: Flipping Burgers

1. Introduction

In this chapter, we explore the fundamentals of the coupled Finite Difference-Finite Element Method for the heat equation. We build off of the Rayleigh-Ritz Method and time-independent Finite Element Method to add on a finite differences piece in order to model time dependent problems. In this chapter, we focus on a more complex heat transfer model: cooking hamburgers. This involves a time dependent process, with boundary conditions that switch as the burger is flipped and ultimately taken off of the cooking skillet. Both p =1 and p = 2 elements are explored, as well as both Euler-Backward and Crank-Nicolson Finite Difference schemes. Lastly, we utilize our developed hamburger model with a classic hamburger recipe found online to explore how well our model does under these "real-life" conditions.

2. Summary of Finite Difference-Finite Element Method

The Finite Difference-Finite Element Method adds on a scheme of Finite Differences to the Finite Element Method discussed in Chapter 2 in order to handle the class of time-dependent problems. For the sake of explanation of the method, the following generic heat equation model with N/R-N/R boundary conditions and over time $0 < t < t_f$ will be considered:

$$\Omega \equiv (0, L), \Gamma_1 \equiv \{0\}, \Gamma_2 \equiv \{L\}$$

$$-\frac{\partial}{\partial x}\left(\kappa(x)\frac{\partial u}{\partial x}\right) + \mu(x)u = f_\Omega - \rho(x)\dot{u} \quad in\ \Omega, 0 < t \leq t_f \tag{1}$$

$$\kappa\frac{\partial u}{\partial x} = \gamma_1 u - f_{\Gamma_1} \quad on\ \Gamma_1, 0 < t \leq t_f \tag{2}$$

$$\kappa\frac{\partial u}{\partial x} = \gamma_2 u - f_{\Gamma_2} \quad on\ \Gamma_2, 0 < t \leq t_f \tag{3}$$

$$u = u_{ic}(x) \quad in\ \Omega, t = 0 \tag{4}$$

In order to maintain simplicity, the following assumptions about different components of these governing equations are made:

$$\kappa(x) > 0, \rho(x) > 0, \mu(x) \geq 0, \forall x \in \Omega, \gamma_1 \geq 0, and\ \gamma_2 \geq 0 \tag{5}$$

Now that a time dependence is introduced, the derivative of u with respect to time must be defined. This is done in an incredibly straightforward way. As u is defined as the sum of a set of weighting factors multiplied by defined φ functions, the derivative of u can be defined by the same sum, except now the weighting factors as their corresponding time derivatives, written as such:

$$\dot{u}(x,t) \approx \dot{u}_h(x,t) = \sum_{j=1}^{n_{node}} \dot{u}_{hj}(t)\varphi_j(x) \tag{6}$$

Once again, we are faced with solving the equation $\underline{A}\,\underline{u}_h = \underline{F}^+$. This time there is an $\underline{F}^+$ because there is an additional term in the governing ODE that contains $\dot{u}$. The $\underline{A}$ and $\underline{M}^{inertia}$matrices are formed as shown below:

$$A_{ij} = \int_0^L \left[\kappa(x)\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx} + \mu(x)\varphi_i\varphi_j\right]dx + \gamma_1\varphi_1(0)\varphi_j(0) + \gamma_2\varphi_1(L)\varphi_j(L) \tag{7}$$

$$\underline{M}^{inertia} = \int_0^L \rho(x)\varphi_i\varphi_j dx \tag{8}$$

The $\underline{F}^+$ can be formed the same way as in Equation 19 in Chapter 2, except with the addition of $f_{\Gamma_1}\varphi_i(0)$ and the replacement of f_Ω with ${f_\Omega}^+$, which equals $f_\Omega - \rho(x)\dot{u}$. Through substitution using Equation 6 and then simplifying:

$$\underline{F}^+ = \underline{F} - \underline{M}^{inertia}\dot{\underline{u}}_h \tag{9}$$

This leads to the following system of ODEs in time:

$$\underline{M}^{inertia}\dot{\underline{u}}_h + \underline{A}\,\underline{u}_h = \underline{F}, 0 < t \leq t_f \tag{10}$$

$$\underline{u}_h = \underline{(I_h u_{ic})}, t = 0 \tag{11}$$

In order to solve this time-dependent system of ODE's, we utilize a Finite Difference formulation. First, a grid in time is established, such that a time step, $\Delta t = \frac{t_f}{(n_{tsteps}-1)}$ and a discrete time $t^k = \Delta t(k-1)$ where $1 \leq k \leq n_{tsteps}$ exist. Using these defined time steps, a finite difference can be formulated to approximate the time derivatives in the ODE system we want to solve. The two schemes we focus on in this chapter are Euler Backward, also known as rectangle right and indicated by $\theta = 1$, and Crank-Nicolson, also known as trapezoidal and indicated by $\theta = 0.5$. For the heat equation example laid out in Equations 1-4, the Finite Difference-Finite Element formulation of the system of ODEs to be solved is:

$$\underline{M}^{inertia}\frac{\underline{u}^k{}_{h,\Delta t} - \underline{u}^{k-1}{}_{h,\Delta t}}{\Delta t} + \underline{A}\left(\theta\underline{u}^k{}_{h,\Delta t} + (1-\theta)\underline{u}^{k-1}{}_{h,\Delta t}\right) = \underline{F},\ 2 \leq k \leq n_{tsteps} \tag{12}$$

$$\underline{u}^k{}_{h,\Delta t} = \underline{(I_h u_{ic})}, k = 1 \tag{13}$$

Lastly, as a clear definition, the finite element approximation of *u* with mesh size *h*, and *k* time steps of size Δt is approximately equal to the exact solution of *u* over all *x* locations in space, but at discrete times t^k, such that $1 \leq k \leq n_{tsteps}$, or in mathematical terms:

$$\underline{u}^k{}_{h,\Delta t}(x) = u(x, t^k), 1 \leq k \leq n_{tsteps} \tag{14}$$

3. Model `semiinf_plus` Implementation

To verify that my implementation of `solve_fld_output_t_sver` is correct, I compared the exact and FD-FE approximation of *u* for both the [p = 1 and θ = 1] and [p = 2, θ = 0.5] cases, as well as the error plots for the various norms.

As seen in Figure 1, for both cases of *p* and θ, $\underline{u}^k{}_{h,\Delta t}(x)$ has appeared to converge to $u(x, t^k)$ by Mesh 3. Then in Figure 2, the error estimation plots for the $L^2(\Omega)$ norm are shown. I can tell that the implementation of `solve_fld_output_t_sver` is correct because the slopes of the error plots are as expected. For the $L^2(\Omega)$ norm, the negative of the slope of the error plot is equal to $r = p + 1$. Therefore, for p = 1 the slope is -2, and for p = 2, the slope is -3. This is the behavior that is seen in the plot in Figure 2.

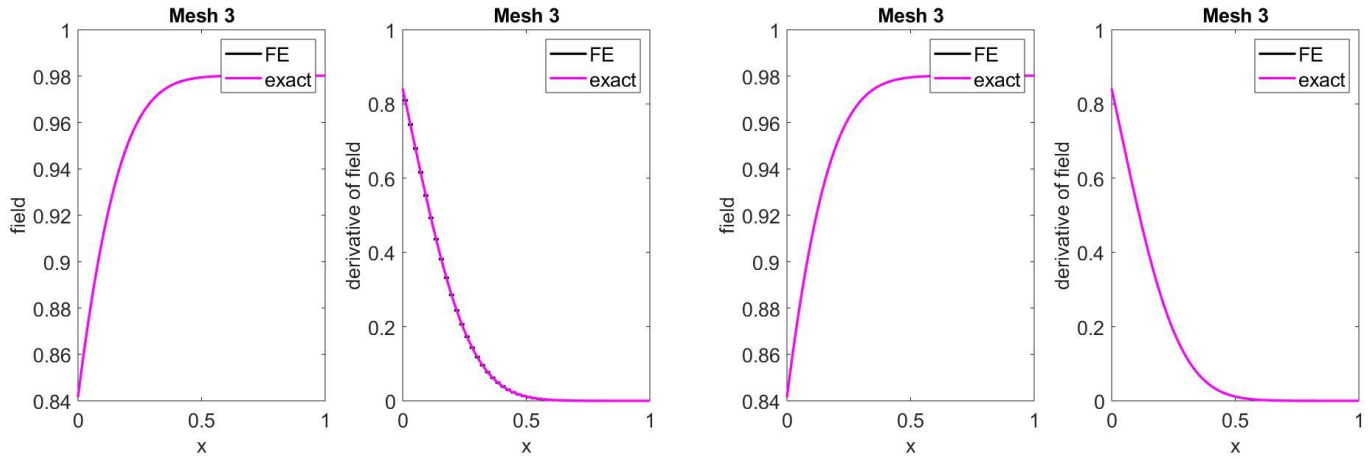


Figure 1*:* Plots of FD-FE Approximation for Semi-Infinite Fin at the 3rd Mesh refinement for p = 1, θ = 1 (top) and p = 2, θ =0.5 (bottom)

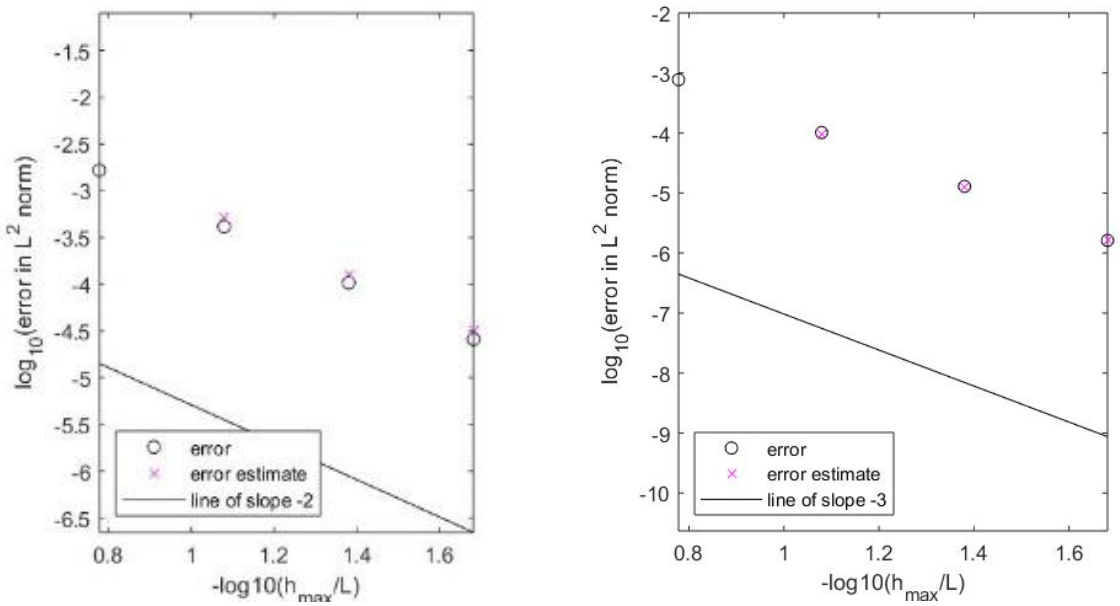


Figure 2: Error vs. Mesh Size for for Semi-Infinite Fin for p = 1, θ = 1 (left) and p = 2, θ =0.5 (right)

4. Implementation of make_probdef_burger

To confirm that my implementation of the problem definition for the burger case study described in "Heat Equation: Study Cases", I compared my results to the given figures Figure BurgerTest1 and Figure BurgerTest2. In Figure 3, the burger temperatures on the skillet side, air side, and mid-burger are plotted with the marked critical temperatures. A comparison of the instructor provided data and data created using my implementation of `make_probdef_burger` shows that they are in agreement. The same agreement can be seen in Figure 4 which compares the given and calculated internal temperature of the burger at time t^{III}.

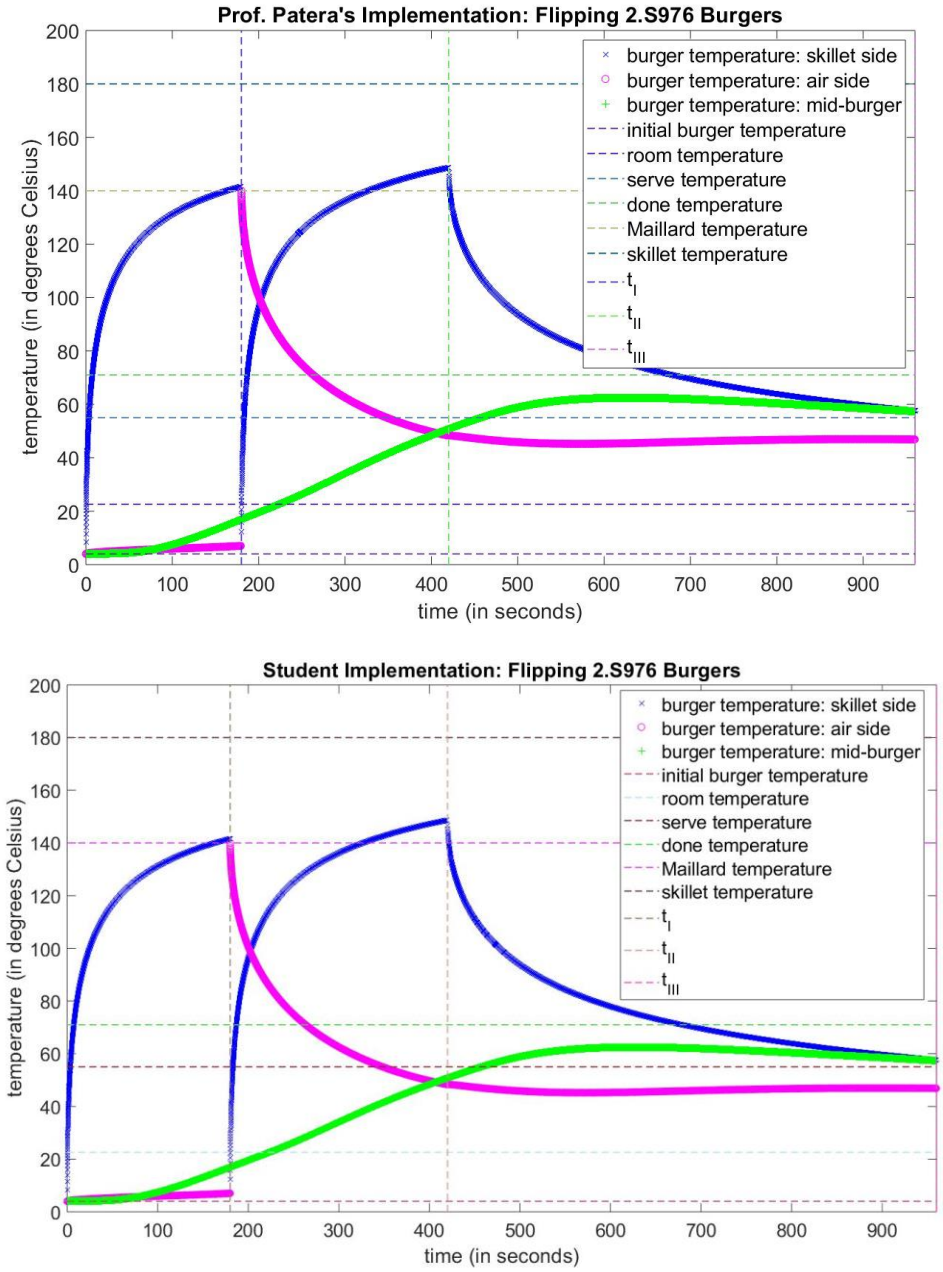


Figure 3: Comparison of Given Data and Data from Student Implementation of `make_probdef_burger` showing burger temperatures at skillet side, air side, and mid-burger

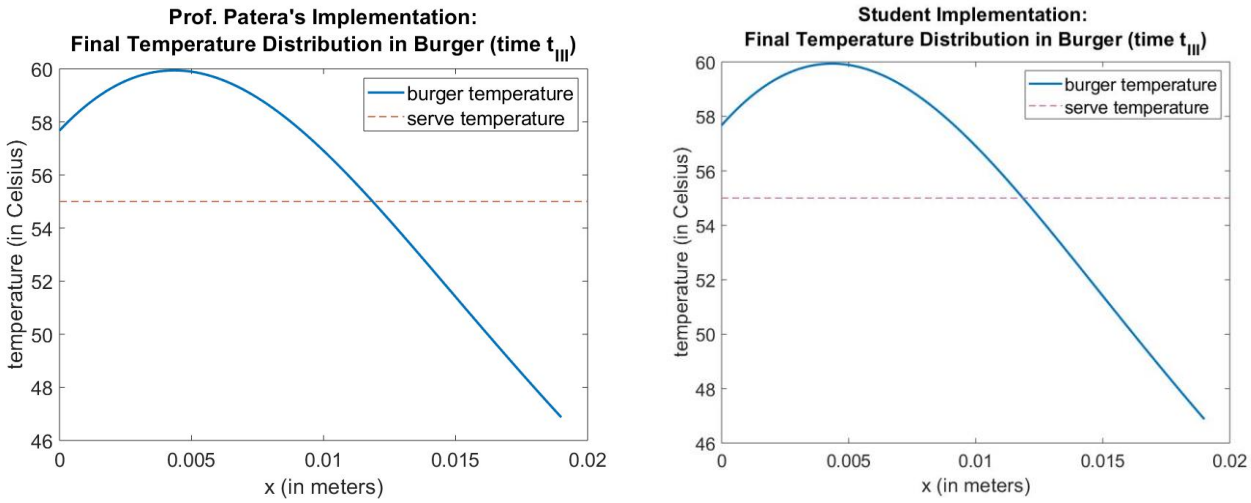


Figure 4: Comparison of Given Data and Data from Student Implementation of `make_probdef_burger` showing the temperature throughout the burger at the time at which the burger was served (t_{III})

5. Discussion of Error & Computation Time

As a next step, I explored the error in the output for [p = 1 and θ = 1] and for [p = 2, θ = 0.5]. First, I determined which mesh refinement corresponded to the coarsest FE mesh for which the error in the output (i.e. the burger temperature at the skillet side just before the flip) is less than 0.001°C. To determine the threshold criteria in the log scale of the error plots we output from MATLAB, I first solved $\log_{10} 0.001 = x$ for $x = -3$. Therefore, any y-values on the output error plot such that $x < -3$, mark a mesh with an output error less than 0.001°C. Figure 5 shows the output error plots for the cases [p = 1 and θ = 1] and for [p = 2, θ = 0.5]. The slopes of the output error log plots are equal to -2p, and in Figure 5, the p = 1 case is on the left. For p = 1, the coarsest mesh to have less than 0.001°C error is 5th refinement with an error of 0.00036°C. For the p = 2 case, the coarsest mesh with acceptable error was the 2nd refinement with an error of 0.00013°C. In the case of output error, the value calculated using $10^{\log_{10}(error\ in\ output)}$ is multiplied by 2. An error tolerance of 0.001°C is excessively small, as our mathematical model itself is not able to meet this tolerance. This small tolerance is used her to highlight the potential advantages of higher order methods, but these advantages themselves are often more apparent only at tighter error tolerances.

Next, I explored the ratio of computational time for both cases [p = 1 and θ = 1] and for [p = 2, θ = 0.5]. To calculate computation time for each case, the below equation was used:

$$t_{comp} = \Delta t_0 \sigma^l \times n_{el} 2^l (\times 2)^* \qquad (15)$$

*An extra factor of two is added in the p = 2 case because we assume that the operation count to solve a penta-diagonal system is twice the operation count to solve a tri-diagonal system. For [p = 1 and θ = 1] on the coarsest mesh with acceptable error (5th refinement),

$$\Delta t_0 = 20, \sigma = 4, n_{el} = 6, l = 5 \text{ and } t_{comp} = 20 * 4^5 \times 6 * 2^5 = 3{,}932{,}160$$

For [p = 2 and θ = 0.5] on the coarsest mesh with acceptable error (2nd refinement),

$$\Delta t_0 = 20, \sigma = 2\sqrt{2}, n_{el} = 6, l = 2 \text{ and } t_{comp} = 20 * (2\sqrt{2})^2 \times 6 * 2^2 = 3{,}840$$

Therefore, the ratio of computational time for the prescribed accuracy of 0.001 for [p = 1 and θ = 1] relative to [p = 2, θ = 0.5] is $\frac{3{,}932{,}160}{3{,}480} = 1{,}024$. This is a huge difference in computation time, with the [p = 1 and θ = 1] case being much more computationally costly than the [p = 2, θ = 0.5] case. Even with a penta-diagonal system to solve, the overall cost for the [p = 2, θ = 0.5] case is lower because it is able to reach the prescribed accuracy in fewer refinements than the [p = 1, θ = 1] case.

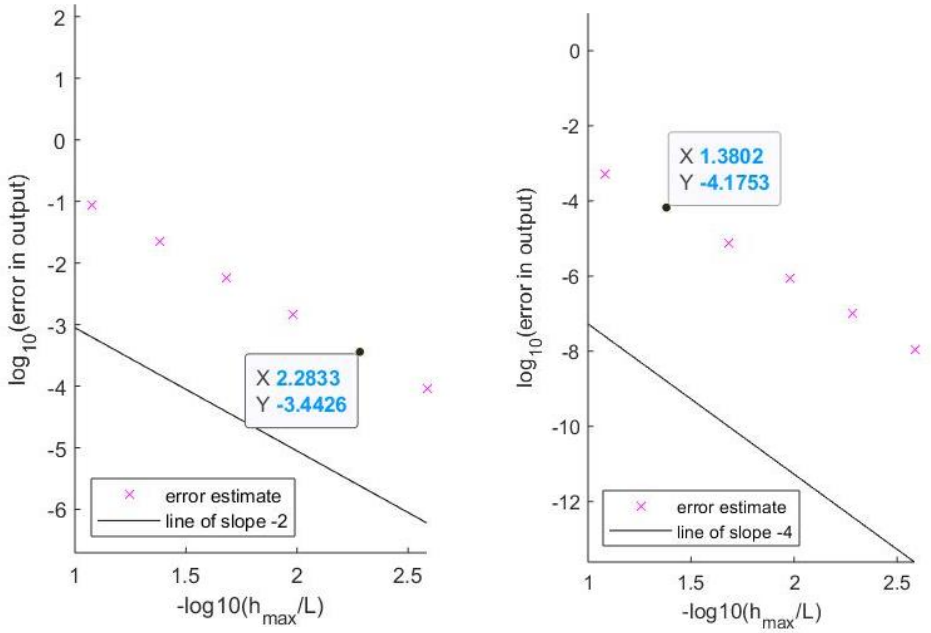


Figure 5: Output Error plots for the [p = 1 and θ = 1] (left) and [p = 2, θ = 0.5] (right) cases

6. Comparison of Model to Recipe

For a final check of the burger flipping model we developed, I found a recipe online and ran our MATLAB FD-FE Model using the parameters outlined in the recipe to see if we obtained simulation results that agreed with predicted recipe results. The recipe I used was from the *New York Times*' Cooking Section[1] and outlined the following problem parameters:

$$Diameter = 4in = 0.1016m, Thickness = 0.5in = 0.0127m, t^{I} = 120\ s, t^{II} = 60\ s$$

There was no t^{III} outlined in this recipe, so I started by keeping the t^{III} from our original simulation of 540 seconds. In Figure 6 are the initial results of temperature over time and internal temperature distribution in the burger at the serving time using the recipe parameters are shown. It is clear, that this combination of times does not appear to make for a good burger in the end, according to our simulation, as when served, the entire burger is at least 5°C below the target serve temperature. The temperature at the middle of the burger does not get above the done temperature, but this is perhaps consistent with the way in which people prefer to cook their burgers to rare. The two sides

[1] Sifton, Sam. "Deconstructing the Perfect Burger." The New York Times, The New York Times, 25 June 2014, cooking.nytimes.com/recipes/1016595-hamburgers-diner-style.

of the burger also do not reach the Maillard temperature, which leads to the nice charring of the meat and brings out flavor. This does not make sense, as one would expect a burger recipe would ensure that the correct temperatures are reached for maximum taste.

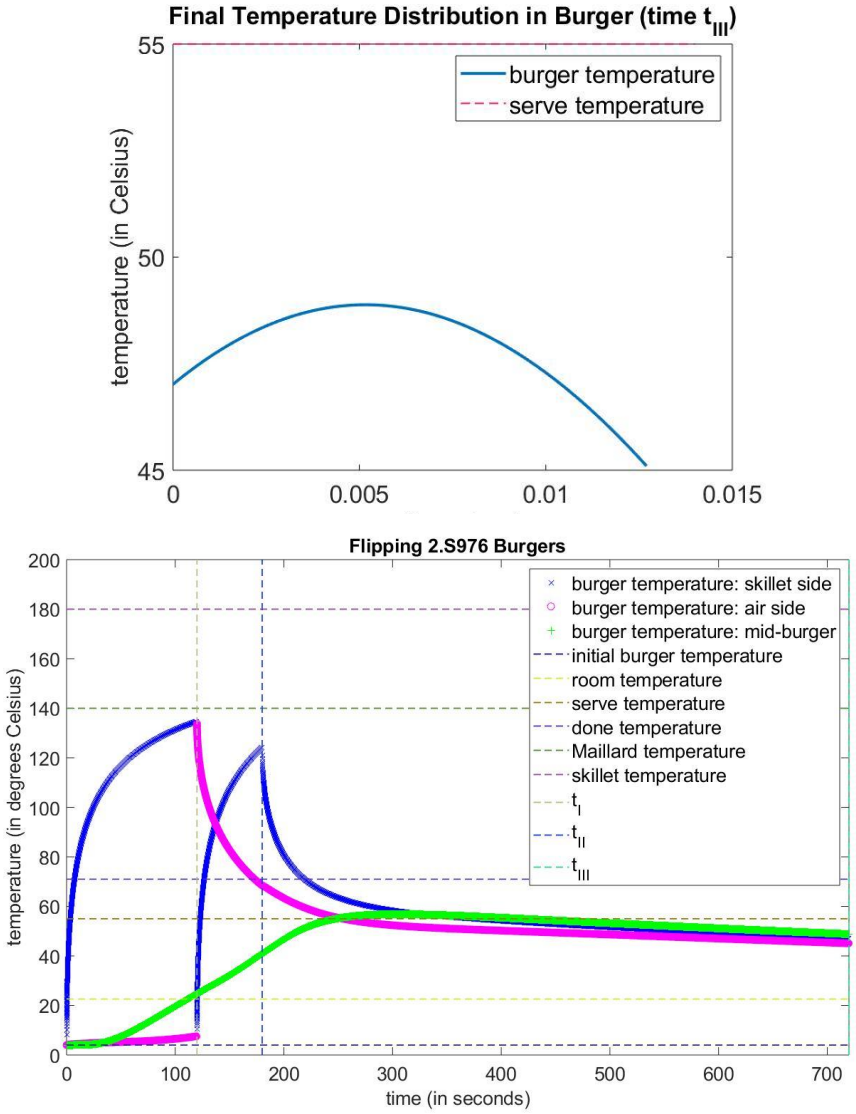


Figure 6: Final temperature distribution and temperature at different locations over time in burger using recipe parameters and t^{III} =540 seconds

In an attempt to make the serve temperature be a more desirable distribution, I changed t^{III} down to 90 seconds, since the recipe did not specify a repose time. The plots that result after this change can be seen in Figure 7. Even though the range of temperatures in the burger when it is served is now greater, more of the burger is above the desired serve temperature, which I believe is more important for burger taste and enjoyment. These shorter repose time results are more in line with the results from using our initial parameters for simulation. The mid-burger temperature is also always rising until the burger is served, instead of beginning to drop before the burger is even served.

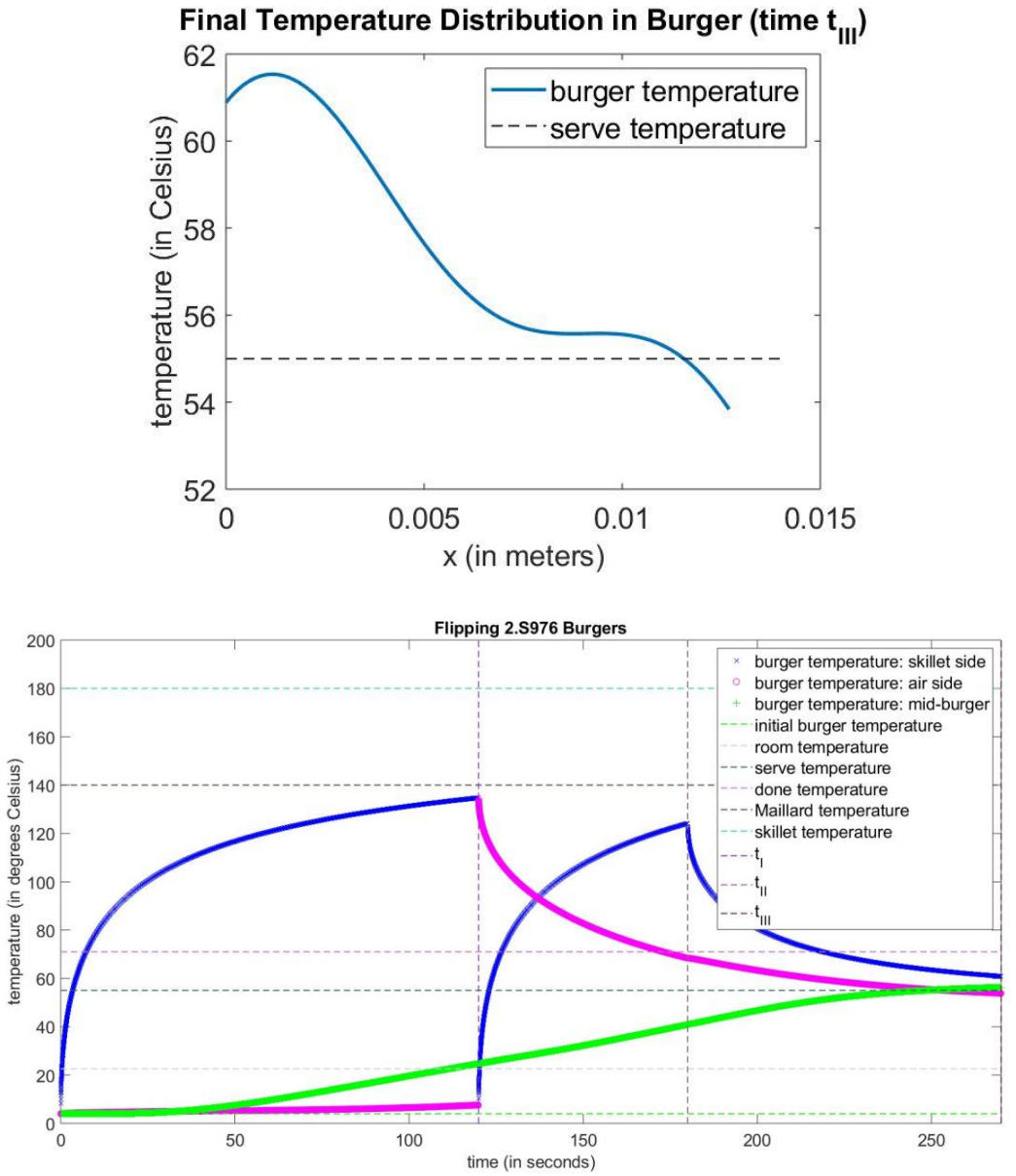


Figure 6: Final temperature distribution and temperature at different locations over time in burger using recipe parameters and t^{III} = 90 seconds

Chapter Four – The FE Method for 1D 4th-Order BVPs (Bending): Xylophone

1. Introduction

This chapter uses the application of tuning a xylophone bar to apply the FE Method for 1D 4th-Order BVPs with Bending. This method is developed with the task of finding the hole locations in a xylophone bar, such that the strings through those holes do not move when the beam is vibrating at its fundamental mode. The results of the developed method are then compared with the results of a different method of solving the same model (Caresta Pre-Print). Finally, the errors in the results of these tests are discussed, as well as potential modifications to the software in order to include the effects of the support springs.

2. Summary of Finite Element Method for Beam Eigen-problems

For Beam Eigen-problems, the equation describing the stress-strain relationship in a beam is solved using eigenvalues. This equation is as follows:

$$\frac{\partial^2}{\partial x^2}\left(\beta(x)\frac{\partial^2 u}{\partial x^2}\right) - N_0\frac{\partial^2 u}{\partial x^2} = q(x,t) - \rho A_{cs}(x)\frac{\partial^2 u}{\partial t^2}$$

Four boundary conditions and two initial conditions are needed. Using a modal representation, $q(x,t) = 0$ and $u(x,t) = \sum_{k=1}^{\infty}(c_1{}^{(k)}\cos(\omega_n{}^{(k)}t) + c_2{}^{(k)}\cos(\omega_n{}^{(k)}t))u^{(k)}(x)$. Using Hermitian Basis Functions, u_h is defined as $u_h = \sum_{j=1}^{2*n_{node}} u_{hj}\varphi_j(x)$. The $\underline{A}$ *and* $\underline{F}$ matrices are formed per page 8 of the "Bending Natural Frequencies" notes. The final Eigen-problem is solved for $\lambda_h{}^{(k)}$ after being defined as $\underline{A}\underline{u}_h{}^{(k)0} = \lambda_h{}^{(k)}\underline{M}^{inertia}\underline{u}_h{}^{(k)0}$.

3. Summary of Xylophone Bar Problem

The xylophone problem sets up a solid bar of wood that will be carved out to vibrate with a desired fundamental frequency that has a certain ratio to the first harmonic frequency of the bar. A drawing of the bar and its parameters taken from page 1 of the "Bending Study Case Xylophone" notes is shown below:

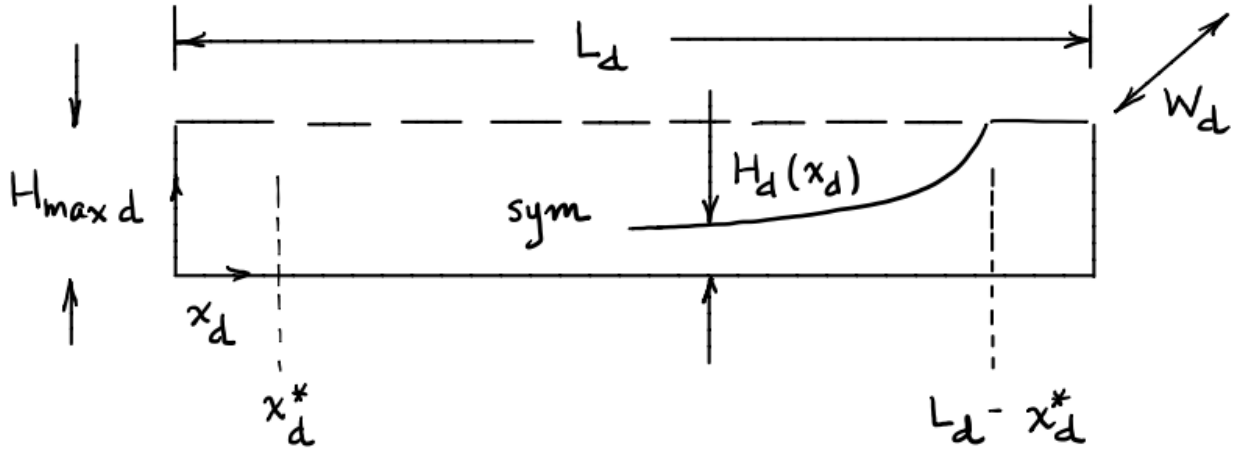


Figure 1: *Diagram of Xylophone Case Study*

The carving out of the bar begins at a point x_d^* from the edge of the bar and is governed by a function that depends on the design variable p_2 such that $p_2 \in [0.05, 1.00]$. The governing Eigen-problem for this model is:

$$\frac{d^2}{dx_d{}^2}\left(\frac{E_d W_d H_d{}^3(x_d)}{12}\frac{d^2 u_d{}^{(k)}}{dx_d{}^2}\right) = \lambda_d{}^{(k)} \rho_d W_d H_d(x_d) u_d{}^{(k)}$$

The bar is physically supported by two strings in tension that pass through holes placed on the nodes of the fundamental frequency so as to minimize the force on the bar.

The design objections are to meet a target ratio, *R*, of the first harmonic frequency to the fundamental frequency. The goal is to find an optimal value for p_2 (`p2opt`) such that the difference between the calculated ratio and the target ratio is less than a prescribed tolerance. This ratio, *R*, can either be equal to 3 for "quint" tuning, or 4 for "double-octave" tuning.

4. Explanation of Algorithm for Fundamental Node Identification

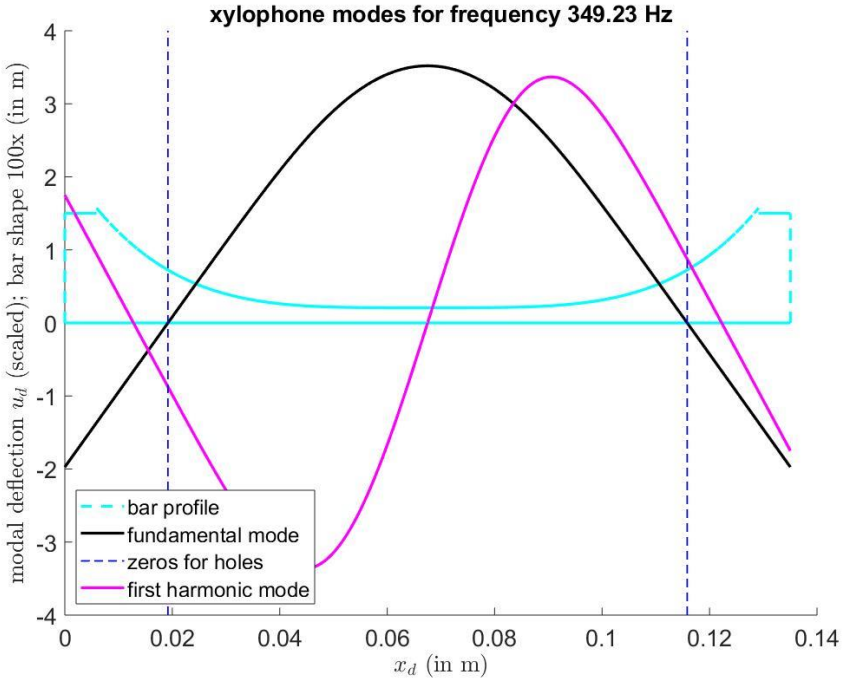


Figure 2: Plot from xylo_bar_design3 showing successful location of fundamental frequency nodes

In order to find the nodes of the fundamental frequency, the locations where the fundamental frequency has deflection of zero need to be identified. First, for each of the two holes, the elements where the fundamental mode crosses zero are found by checking if the product of the values of the mode at the two end nodes (`mcheck`) of an element is less than zero. A negative value of `mcheck` indicates that the element being checked contains a zero. Then, the summation: $\sum_{l=1}^{4} u_h{}^{(3)}{}_{lg2(l,m^*)} \hat{S}_{lm^*} \hat{x}_{hole}$) is computed by forming a 1 by 4 matrix of $u_h^{(3)}$ and then multiplying it by `hshape_fcn(x,h(mstar(hole)))`. Then the function `fzero` is used to find the zero

of the sum function. Lastly, the zeros $\hat{x}_{hole}$ are scaled to the global dimensional domain via $x^{hole} = \left(x^{\lg(1,m^*)} + h^{m^*}\hat{x}_{hole}\right) * L_d$. As confirmation that this algorithm is working correctly, I present Figure 2, which shows the plot provided by the function `xylo_bar_design3`.

5. Validation via Caresta Pre-Print

The purpose of this section is to compare the results of our analysis with those derived using a different method that included experimental results. For the Caresta Test, the same physical parameters that Caresta used were passed to `xylo_bar_design3` while keeping `justcalc_L_d` as true. This had the program determine the length of the bar that was needed to achieve the required frequency, but kept the beam completely solid and not at all hollowed out, like it is with a xylophone piece. The parameters passed for the Caresta test are as follows (using given beam data and Theoretical natural frequencies):

Parameter	Variable Name	Value
Target Fundamental Frequency	`frequency3target_d`	32.80 Hz
Target Ratio Fundamental/ First Harmonic	`R_target`	2.7573
Height of Beam (constant)	`H_max_d`	0.01m
Carving Function Parameters	`P2_interval`	[1.0, 1.0]
Young's Modulus of Beam	`Ebar_d`	7800 kg/m^3
Boolean for Optimization of Shape	`Justcalc_L_d`	true

Table 1: Parameters Passed to `xylo_bar_design3` for Caresta Test

In return, `xylo_bar_design3` returned a calculated value for the length of the bar to meet the required fundamental frequency and tuning requirements, as well as the fundamental frequency and first harmonic for the calculated length. The following were the results of running this code:

$$Fundamental\ Freq = 32.8\ Hz, 1st\ Harmonic = 90.4145\ Hz, Length = 1.2752m$$

This results in the following errors between the given Caresta results (including that the bar has length 1.275m) and the results of this code.

$$Error_{R\_target} = -7.7845 \times 10^{-4}, Error_{L_d} = 1.8704 \times 10^{-4}$$

These results allow me to confirm that `xylo_bar_design3` is successfully completing the expected calculations of fundamental frequency and first harmonic, as well as the length of a beam required to be tuned to a certain frequency. Because these results were obtained using an independent method from what was implemented in `xylo_bar_design3`, I can use them as verification that my code is computing results as expected.

6. Tuning a Xylophone Bar & the Associated Errors

To test the process of tuning a xylophone bar including the `binarychop` function (i.e. going beyond the Caresta test), the following parameters were passed to `xylo_bar_design3`:

```
frequency3target_d = 349.23; %pitch of F4
R_target = 4;
Hmax_d = 0.015;
xstar = 0.05;
p2_interval = [0.05,1.0];
Ebar_d = 1.4e10; %in Pa
rhobar_d = 835; %in kg/m^3
justcalc_L_d = false; %enables optimization and binarychop
suppress = true;
```

The results of this tuning included a `p2opt` value of 0.1391 and an `L_d` of 0.135m. The output plot can be seen below in Figure 3.

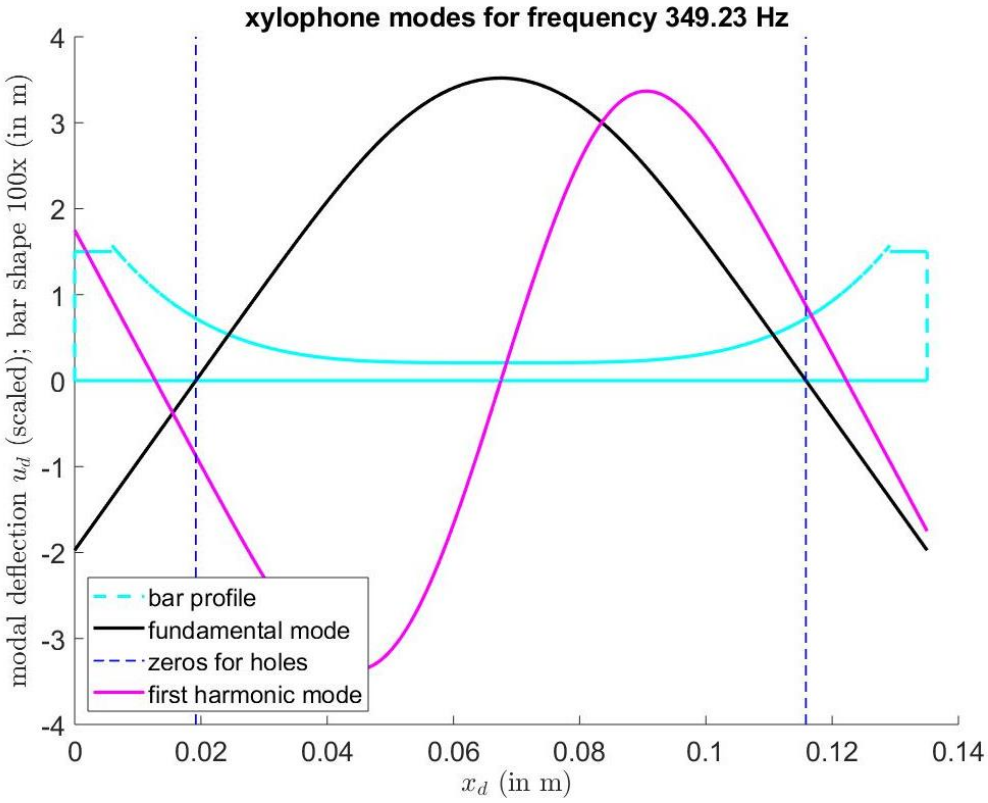


Figure 3: Results of Xylophone Tuning with `frequency3target_d` = 349.23 and `R_target` = 4

This figure verifies the correct functionality of `binarychop` because the lines indicating the zeros for holes cross exactly over the points at which the fundamental mode has zero modal

deflection. This indicates that the holes are correctly placed, as they will not hinder the vibration of the bar at its fundamental frequency.

The following list is the resulting output frequencies, their errors, the harmonic-fundamental frequency ratio, and its error:

```
frequency3_d = 349.2300
err_frequency3_d = 5.7685e-05
frequency4_d = 1.3938e+03
err_frequency4_d = 1.4453e-05
4d_3d = 3.9911
err_4d_3d* = [3.991128190712825, 3.991129097903609]
```

*in form [lower bound, upper bound]

To determine the error of the ratio of the first harmonic frequency to the fundamental frequency, a method of adding and subtracting the errors of the two frequencies are employed. A simple depiction of the ratio is:

$$\frac{frequency4\_d \pm err\_frequency4\_d}{frequency3\_d \pm err\_frequency3\_d}$$

In order to determine the error of this ratio, I chose to determine the bounds of the ratio. This ratio is smallest, when `err_frequency4_d` is subtracted from the first harmonic, and `err_frequency3_d` is added to the fundamental. The ratio is greatest when `err_frequency4_d` is added to the first harmonic and `err_frequency3_d` is subtracted from the fundamental. An expression of the lower and upper bounds of the ratio is therefore as below:

$$\left[\frac{frequency3\_d - err\_frequency3\_d}{frequency3\_d + err\_frequency3\_d} \quad \frac{frequency3\_d + err\_frequency3\_d}{frequency3\_d - err\_frequency3\_d}\right]$$

7. Discussion of Error

I do believe that the FE error estimators used in this program are reliable. There are no abnormalities seen in the error plots generated when tuning a xylophone. As can be seen in Figures 4-7, the error markers have the same slope as the predicted slope lines and there are no upward deviations in the middle of a series of mesh refinements, which would indicate a potential problem. These results are consistent across both the fundamental and first harmonic frequencies, as well as across program runs with different values of `frequency3target_d`.

I believe, that for the same mesh, the FE error will be greater for `frequency4_d` compared to `frequency3_d`, which is confirmed by the results in Figures 4-7. Errors for `frequency3_d` are only on the order of 0.0001-0.00001, while the errors of `frequency4_d` are on the order of 0.01-0.001.

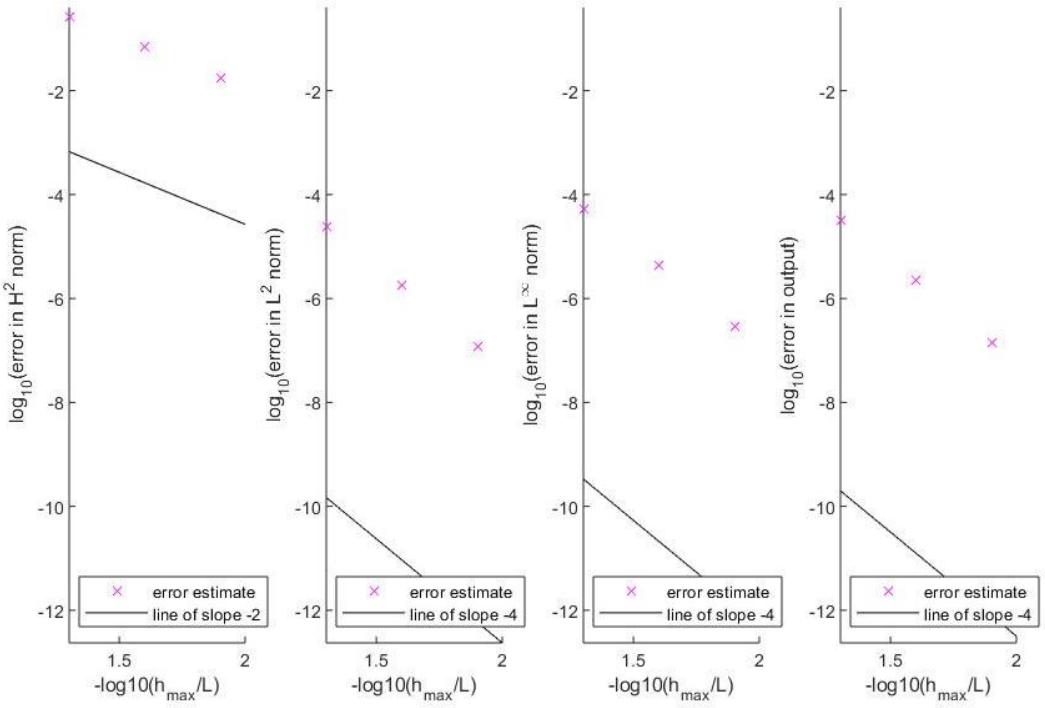


Figure 4: Error plots for Fundamental for `frequency3target_d` = 349.23Hz

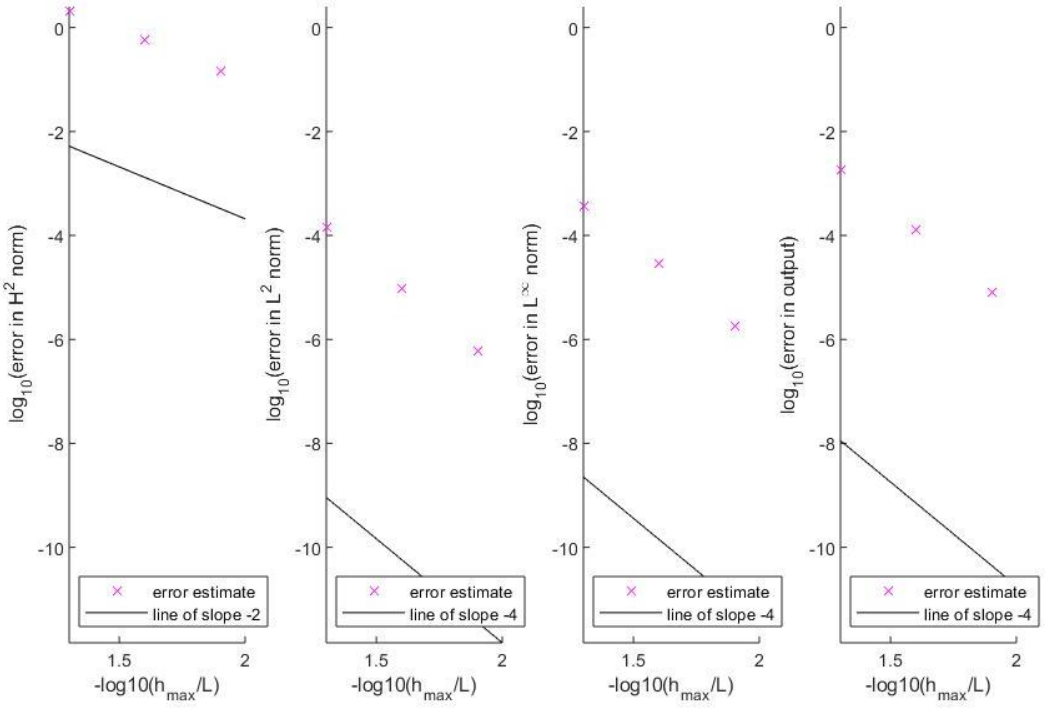


Figure 5: Error plots for First Harmonic for `frequency3target_d` = 349.23Hz

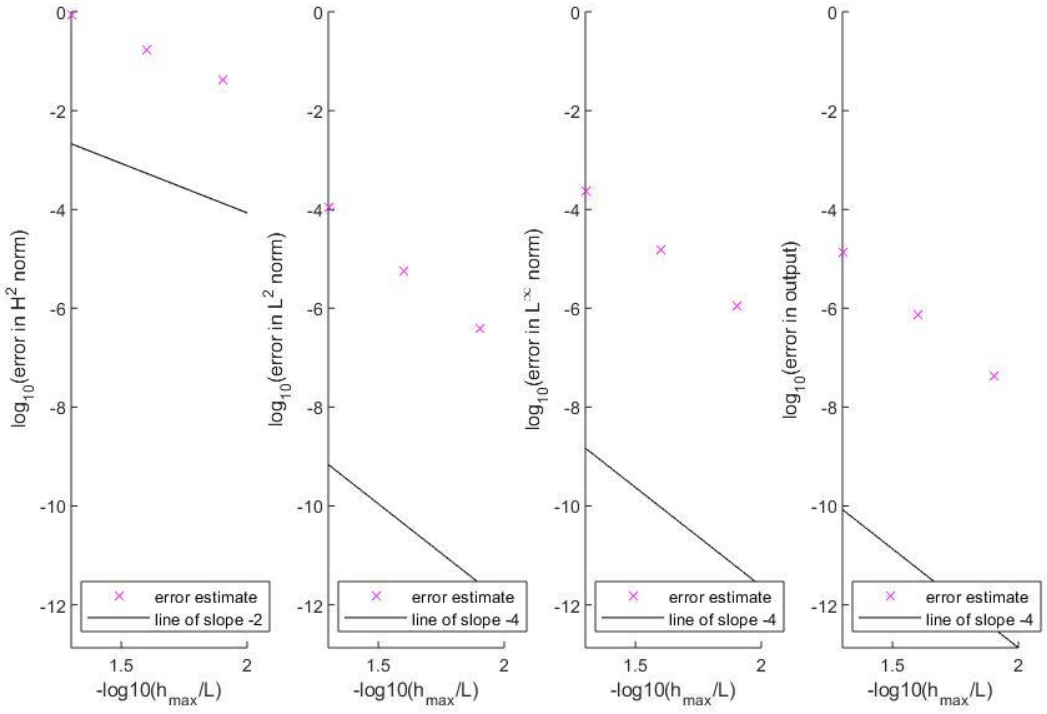


Figure 6: Error plots for Fundamental for `frequency3target_d` = 698.46Hz

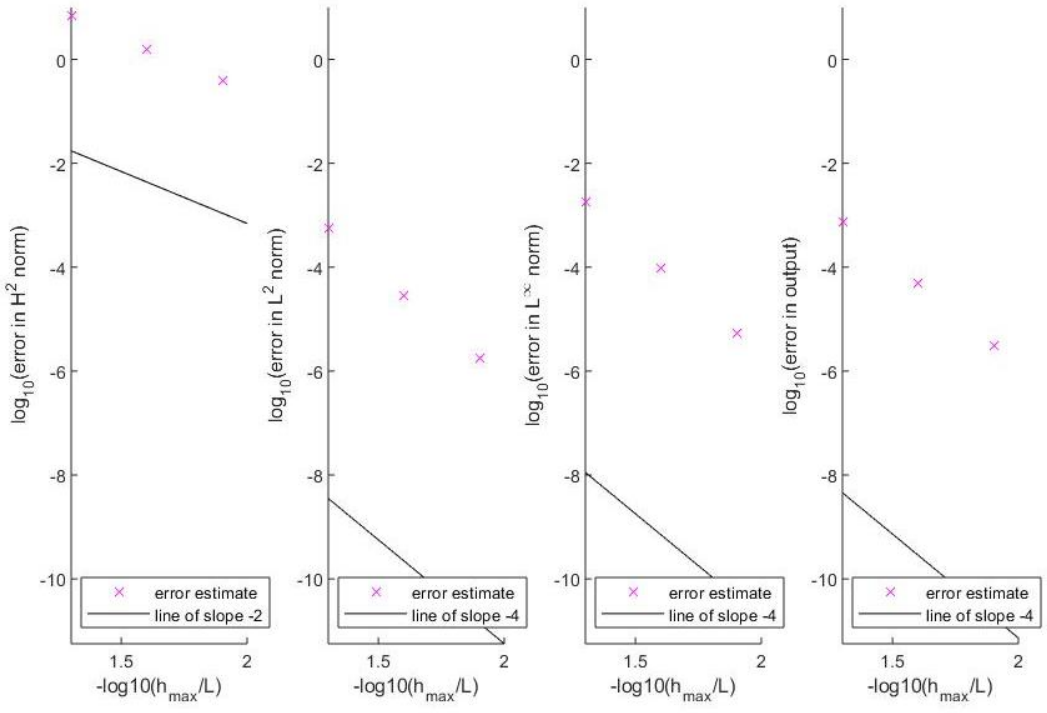


Figure 7: Error plots for First Harmonic for `frequency3target_d` = 609.46Hz

As the sensitivity of the untrained human ear is 10 Hz, I believe the mesh refinements are too fine in comparison to the sensitivity of the humans distinguishing between the different modes of the xylophone bar. This is a comparison of 10 to 0.01 (at the greatest errors in frequency) and 10 to 0.00001 (for the smallest errors in frequency).

Additional error is introduced by the treatment of the xylophone bar as an Euler-Bernoulli beam. Because of this, the predictions of `xylo_bar_design3` will be more accurate for longer beams rather than shorter beams, as `Hmax_d` is assumed to be the same for all bars. This is because Euler-Bernoulli Beam Theory assumes long and slender beams. Longer beams correspond to bars tuned to lower frequencies. This can be verified by the fact that a bar tuned to 349 Hz has a length of 0.135m, while a bar tuned to 698 Hz has a length of 0.095m.

8. Inclusion of Support Springs

In order to modify the energy functional by the addition of a $\frac{1}{2}k_s\omega^2(L)$ term, which represents a lumped Hookean spring attached to the right end of the beam, the red term below needs to be added to the $\underline{\underline{A}}$ matrix:

$$A_{ij} = \int_0^L EI\frac{d^2\varphi_i}{dx^2}\frac{d^2\varphi_j}{dx^2}dx + k_s\varphi_i(L)\varphi_j(L), 1 \le i,j \le 2 * n_{node}$$

The line of code to make this addition needs to be added in the function `impose_boundary_condition.m` in the folder `UG_FE_1d_bend_sver`. The addition should be added after line 42 and should read:

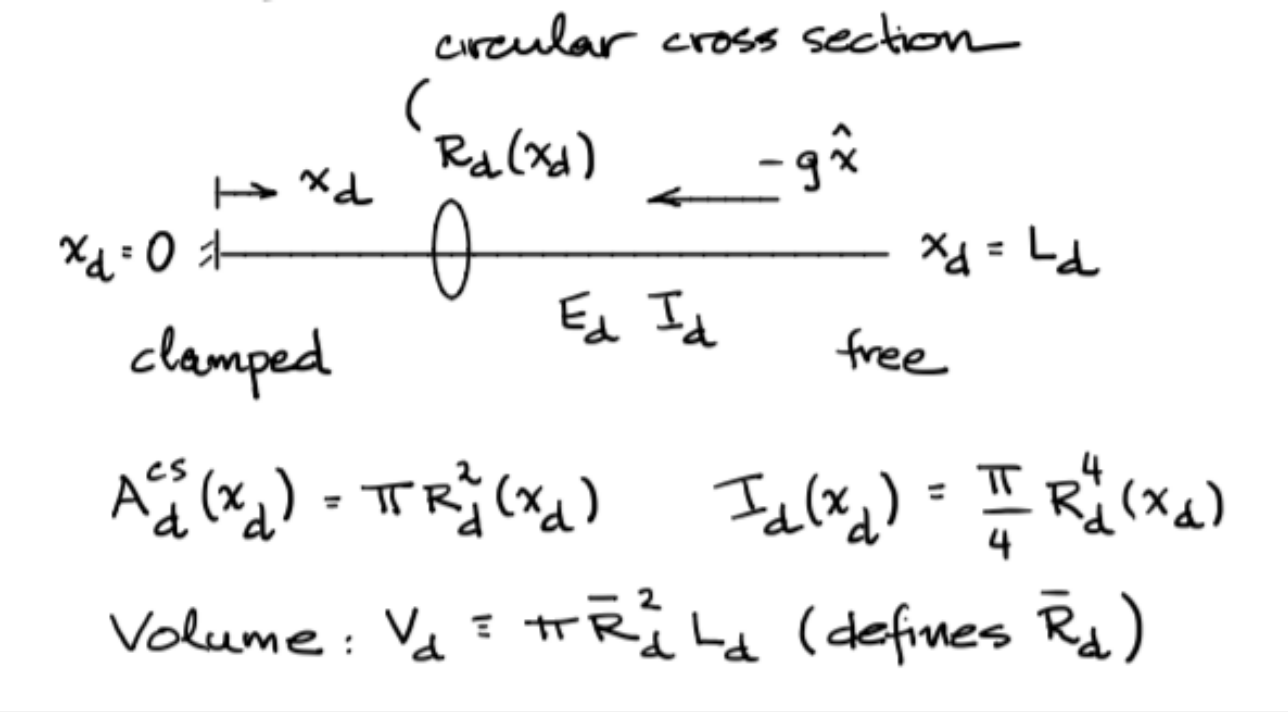
```
A(rightside_node, rightside_node) = A(rightside_node, rightside_node) + ks
```

As `rightsside_node` is equal to `ttomap_fcn(n_el0 + 1,1)`, this accesses the node at the right end of the bar and the first degree of freedom, which is deflection. This is the φ_5 function, which is the only non-zero function at the rightside_node (and is equal to 1). Therefore, the term added is only k_s rather than $k_s\varphi_i(L)\varphi_j(L)\ as\ \varphi_5(L) = 1$.

THE FE METHOD FOR 1D 4TH-ORDER BVPS (BENDING): SELF-BUCKLING

Riley Davis

SELF-BUCKLING



[1] 2.S976 Notes "Buckling Eigenproblems"

With N<0 and P defined as P = -N, the dimensional equations for our self-buckling study case can be written as:

$$\frac{d^2}{dx_d{}^2}\left(E_d I_d \frac{d^2 w_d}{dx_d{}^2}\right) + \frac{d}{dx_d}\left(P_d \frac{dw_d}{dx_d}\right) = q_d$$

Through non-dimensionalizing and applying boundary conditions, the following Self-Buckling Eigenproblem emerges to be solved:

$$\frac{d^2}{dx^2}(R^4 \frac{d^2 u}{dx^2}) = \lambda\left(-\frac{d}{dx}(P \frac{du}{dx})\right), 0 < x < 1$$

$$u(0) = u_x(0) = 0, u_x(1) = (R^4 u_{xx})_x = 0,$$

FE METHODS OF SELF-BUCKLING

To solve Self-Buckling Problems, the Finite Element method sets up the following Eigenproblem:

$$\underline{A}\,\underline{u}^0{}_h = \lambda_h \underline{K}^{ax}\,\underline{u}^0{}_h$$

That solves for $\lambda_h = \gamma_{critical}$

Such that:

$$\underline{u}_h = \left[0; 0; \underline{u}^0{}_h\right]$$

$$\underline{A} = \underline{\tilde{A}}(3{:}\,end, 3{:}\,end), \underline{K}^{ax} = \underline{\widetilde{K}}^{ax}(3{:}\,end, 3{:}\,end)$$

$$\tilde{A}_{ij} = \int_0^1 R^4(x)\frac{d^2\phi_i}{dx^2}\frac{d^2\phi_j}{dx^2}dx$$

$$\widetilde{K}^{ax}{}_{ij} = \int_0^1 P(x)\frac{d\phi_i}{dx}\frac{d\phi_j}{dx}dx$$

BUCKLING OPTIMIZATION PROBLEM

Using the finite element method to solve for $\lambda^{(1)} = \gamma_{critical}$, the objective of the optimization problem for our stated buckling problem is:

- To maximize L_d subject to constraints CV, CM, CS:

  Such that – γ_c is maximized over p_1 as γ_c^{opt}

  And L_d^{opt} is chosen such that $L_d{}^{opt} = (\frac{\gamma_c{}^{opt} E_d V_d}{4\pi\rho g})^{1/4}$

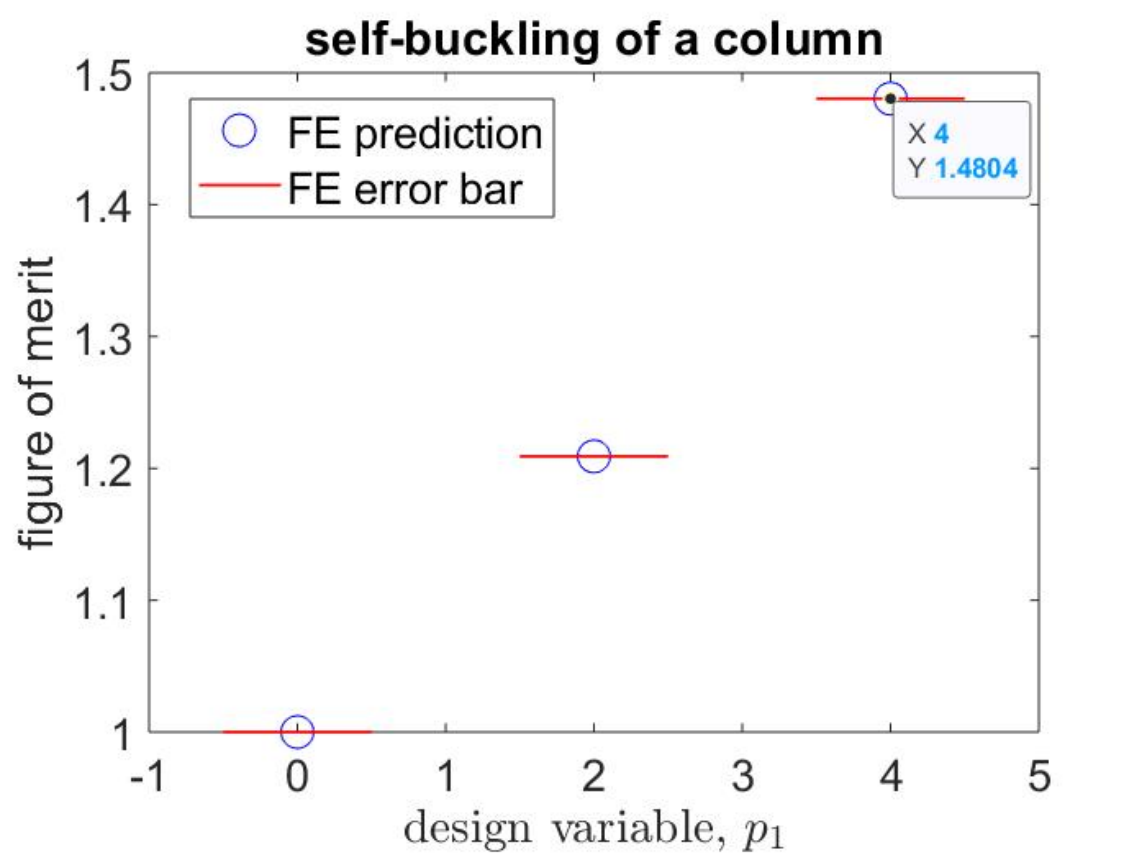
  The constraints of the optimization problem are defined as:

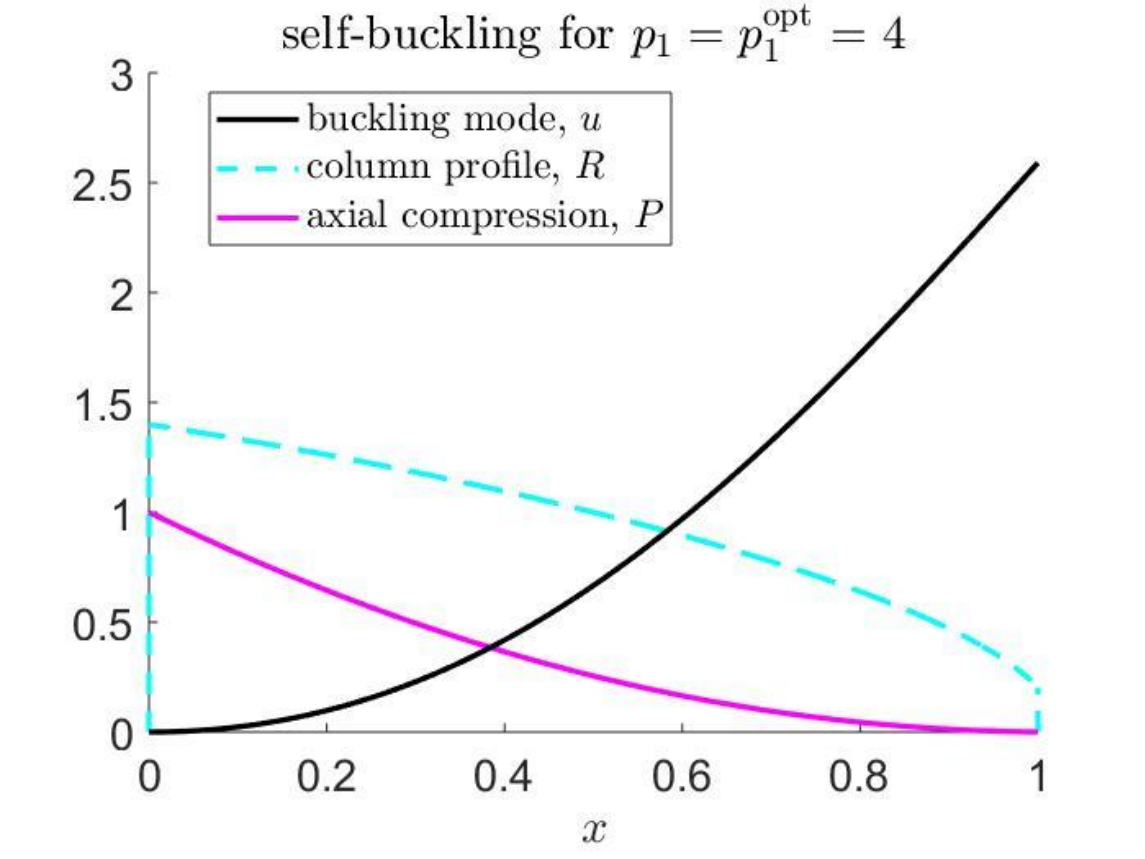
Fixed Volume (CV)	$\int_0^1 G(x)dx = 0$
Minimum Relative Radius (CM)	$R(x) \geq R_{min} = 0.2$
Gradual Variation (CS)	$\lvert G'(x)\rvert \leq S_{max} = 10.0$

CHOSEN "BEST SOLUTION"

$$G(x) = -p_1 * 0.5 * \sin(x - 0.5)$$
$$with\ p^{opt} = 4$$
$$\boldsymbol{G(x) = -2\sin(x - 0.5)}$$

$$\boldsymbol{FOM = 1.48}$$





VERIFICATION OF REQUIREMENTS

VOLUME CONSTRAINT

To satisfy the volume constraint:

$$\int_0^1 G(x)dx = 0$$

$$\int_0^1 -2\sin(x-0.5)\,dx = 2\int_0^1 \sin(0.5-x)\,dx$$

$$u = 0.5 - x, du = -dx$$

$$2\int_{-1/2}^{1/2} \sin(u)\,du = 0$$

MINIMUM RELATIVE RADIUS

To satisfy the minimum radius constraint:

$$R(x) \geq R_{min} = 0.2$$

$$G(x) \geq -1 + {R_{min}}^2 = -1 + 0.2^2 = -0.96$$

$$over\ 0 \leq x \leq 1$$

$$\max\big(G(x)\big) = 0.95885$$

$$min\big(G(x)\big) = -0.95885$$

Which is greater than -0.96!

GRADUAL VARIATION

To satisfy the gradual variation constraint:

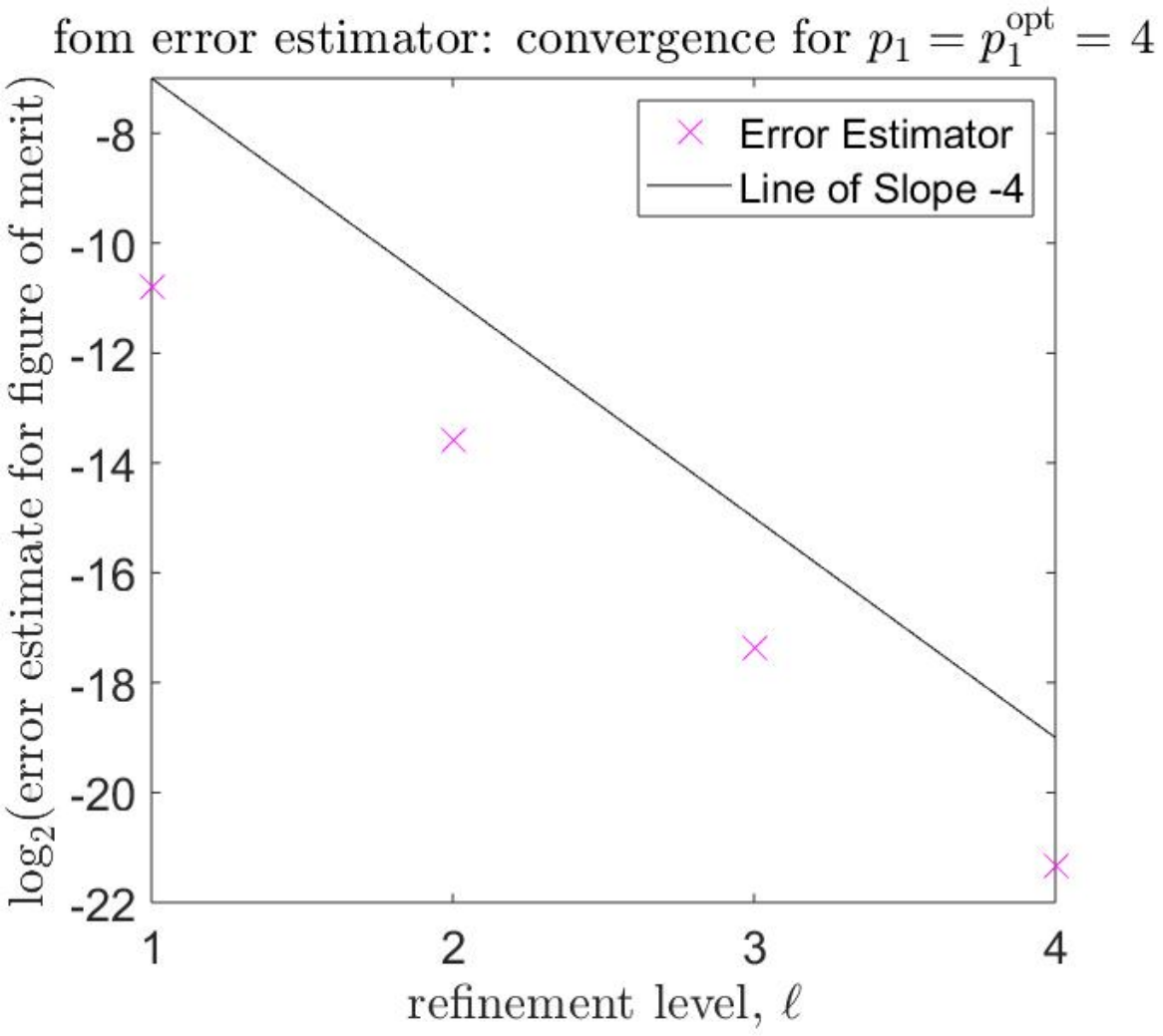
$$|G'(x)| \leq S_{max} = 10.0$$

$$G'(x) = -2\cos(x - 0.5)$$

$$|G'(x)|_{MAX} = 2$$

Which is less than 10.0!

FINITE ELEMENT ERROR



- Error appears to be converging, with no pathologies nor indications of finite-precision effects
- Maximum error is 0.000565, which is much smaller than 0.01
- To determine if error is converging at the expected slope:
  - Let $f(\lambda) = Figure\ of\ Merit$
  - The error of $f(\lambda) = f(\lambda + \epsilon) - f(\lambda)$
  - Sensitivity Analysis Indicates that the error of the Figure of Merit depends on $f'(\lambda) * error\ in\ \lambda$
  - The error in λ corresponds to the error in s_{out} from theory from lecture notes – this corresponds to a slope of the error estimator convergence of -4

Appendix A: Model I

$$-k\frac{d}{dx}\left(\pi {R_o}^2\left(1+\ \beta\frac{x}{L}\right)^2\frac{du}{dx}\right)=0\ in\ \Omega$$

$$k\frac{du}{dx}=\ -q_1\ on\ \Gamma_1$$

$$-k\frac{du}{dx}=\ \eta_2(u-u_\infty)\ on\ \Gamma_2$$

Exact Solution

$$u=\ u_\infty+\frac{q_1L}{k}\left(\frac{1+\ \beta+\frac{k}{\eta_2L}}{(1+\beta)^2}-\frac{\left(\frac{x}{L}\right)}{\left(1+\beta\frac{x}{L}\right)}\right)$$

$$\frac{du}{dx}=\ \frac{-q_1L^2}{k(\beta x+L)^2}$$

Standard Energy Functional for Neumann/Robin Boundary Conditions

$$\Pi(\omega)=\frac{1}{2}\int_0^L\left[\kappa(x)\left(\frac{d\omega}{dx}\right)^2+\ \mu(x)\omega^2\right]dx$$

$$+\frac{1}{2}(\gamma_1\omega^2(x)+\ \gamma_2\omega^2(L))-\int_0^L f_\Omega(x)\omega dx-\omega(0)f_{\Gamma_1}-\omega(L)f_{\Gamma_2}$$

Entries of $\underline{A}$ and $\underline{F}$

$$A_{ij}=\int_0^L\left[\kappa(x)\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}+\mu(x)\psi_i\psi_j\right]dx+\gamma_1\psi_i(0)\,\psi_j(0)+\gamma_2\psi_i(L)\psi_j(L)$$

$$F_i=\ \int_0^L[f_\Omega(x)\psi_i]dx+\ f_{\Gamma_1}\psi_i(0)+\ f_{\Gamma_2}\psi_i(L)$$

Appendix B: Model II

$$-kA_{cs}\frac{d^2u}{dx^2}=\eta_3P_{cs}(u-u_\infty)=0\ in\ \Omega$$

$$u=\ u_{\Gamma_1}on\ \Gamma_1$$

$$-k\frac{du}{dx}=0\ on\ \Gamma_2$$

Exact Solution

$$u = u_\infty + (u_{\Gamma_1} - u_\infty)\frac{\cosh(\sqrt{\mu_0}\left(1-\frac{x}{L}\right))}{\cosh(\sqrt{\mu_0})}$$

$$\frac{du}{dx} = \frac{-\sqrt{\mu_0}(u_{\Gamma_1} - u_\infty)\sinh(\sqrt{\mu_0}\left(1-\frac{x}{L}\right))}{L\sqrt{\mu_0}}$$

Standard Energy Functional for Dirichlet Boundary Conditions

$$\Pi(\omega) = \frac{1}{2}\int_0^L \left[\kappa(x)\left(\frac{d\omega}{dx}\right)^2 + \mu(x)\omega^2\right]dx + \frac{1}{2}\omega^2(L) - \int_0^L f_\Omega(x)\omega dx - \omega(L)f_{\Gamma_2}$$

Entries of $\underline{\tilde{A}}$ and $\underline{\tilde{F}}$

$$\tilde{A}_{ij} = \int_0^L \left[\kappa(x)\frac{d\psi_i}{dx}\frac{d\psi_j}{dx} + \mu(x)\psi_i\psi_j\right]dx + \gamma_2\psi_i(L)\psi_j(L)$$

$$\tilde{F}_i = \int_0^L [f_\Omega(x)\psi_i]dx + f_{\Gamma_2}\psi_i(L)$$