Finite Element Methods Project

Ali Daher

16^{th} May, 2019

Massachusetts Institute of Technology (MIT)
Department of Mechanical Engineering
Report Submitted in Fulfillment of the Requirement for Finite Element Methods for Mechanical Engineers Course (2.S976) Spring 2019.

Name of Student: Ali Daher
Name of Supervisor: Prof. Anthony Patera

Acknowledgements

I would like to thank Professor Anthony for his invaluable support and guidance over the last semester, without which this report would not have been possible. I am grateful to him for providing me the chance to learn how to utilize and build upon many of the theoretical concepts taught in the undergraduate mechanical engineering subjects, and apply them within the framework of the project to solve real world problems of engineering and mathematical physics. His enthusiasm and passion for teaching, words of encouragement, and constructive feedback has helped me develop a broad perspective on the topic of the finite element method and mathematical modeling in general.

Abstract

The finite element method is a numerical technique used to compute approximations of the real solutions to differential equations that often cannot be solved via analytical methods. The project report examines the methodology behind the finite element method, through illustrative applications in the thermal and structural systems.

The first chapter examines the Rayleigh Ritz method, a preamble to the finite element method, through two thermal mathematical models. The first model corresponds to a quasi-1D heat conduction in a conical frustum, and the second model corresponds to a 1D conduction through right-cylinder thermal fin. Each model is subject to a particular combination of boundary conditions (Dirichlet and Neumann/Robin) at its left and right side. In the Rayleigh Ritz method, we are provided with a set of base functions, and the approximated solution is the weighted sum of these base functions. The lower the energy function of a solution approximation, the closer to the actual solution the approximation is. Hence, given a set of base functions, the weights of the base functions is obtained from solving a linear system of equations that would carry out the energy functional minimization. For both models, we examine how the energy functionals, the accuracy of the approximated solution to the actual solution, and the relative output error, change with different choice of base functions, as well as the model parameters (which in turn change the actual solution). In general, as we provide more base functions (from constant to linear to quadratic base functions), the Rayleigh Ritz method more accurately approximates the mathematical model. Furthermore, as the actual solution starts having steeper curvature (as a result of parameter modification), the Rayleigh Ritz method becomes less capable of capturing the actual solution using the the limited set of base functions provided.

For chapter two, we first start using the finite element method for approximating the solution for 1D second order symmetric positive definite (SPD) boundary value problems, also corresponding to a thermal system. A finite element method can be viewed a specific case of the RR method for approximation of a function over a domain. In this case, our basis functions are actually piece wise functions, either linear (p=1) or quadratic (p=2), that are only non-zero at specific regions in the domain, and are thus more capable of capturing local effects of the function. To solve for the unknown function over the domain, the method subdivides the large domain into smaller, simpler parts that are called finite elements (in the one dimensional case, finite segments). The simple base functions that model these finite elements are then assembled into a larger system of equations that model the entire problem, and are weighted in such a way to minimize the energy functional, just as in the general RR case. The choice of the nature of the base functions (linear for p=1 or quadratic for p=2), would in turn affect the structure of the arrays (matrices and vectors) used in the linear system of equations to solve for the weighs of the base functions. In both cases, the analysis is conducted over a generic reference element, before being mapped to and applied to to each of the global elements. Analyzing the convergence of the error estimates with mesh refinement can be used to either provide confidence, or invalidate correct numerical implementation. If the posteriori error estimates for a particular error norm converge at the correct theoretical rate that corresponds to the error norm, or if their behavior agrees with that which is expected (for instance, super convergence at the nodes), then we are able to obtain better confidence of correct numerical implementation of the finite element method for the model in the code.

For chapter three, we extend our analysis beyond the steady state case for the models in chapters one and two, and examine a spatiotemporal mathematical model represented by a partial differential equation (PDE). We use the study case of flipping burgers over multiple stages while cooking to study how temperature varies with both space and time. The finite difference-finite element method required for spatiotemporal modeling, can be viewed as a discretization process conducted over two stages. We first discretize only in space, leaving the problem continuous in time. This leads to a system of ordinary differential equations in time, called the semidiscrete equations," which are similar to what we obtained for the FE steady state case. Afterwards, we use the finite difference method, to approximate the time derivatives, which converts each ODE to a system of algebraic equations (in our case a linear system), ultimately transforming the original PDE into equations solved by matrix/algebra. In general, implicit schemes, while more expensive per timestep, are, as opposed to explicit schemes, usually unconditionally stable, thus do not necessarily require smaller time steps. This means that to achieve a prescribed modeling accuracy, implicit schemes would require less time steps and would thus be less expensive than explicit levels, especially for one space dimensions in which matrix solves required for implicit schemes are not significantly more expensive than matrix multiplications required for explicit schemes. Successive discretization of the model involves both space and time simultaneous refinements chosen to balance the spatial and temporal error convergence, which in turn depends on the finite difference scheme used, the finite element nature of the base function used, and the error norm being analyzed. As for chapter two, convergence of errors at the right rates provides confidence of correct numerical implementation. Finally, we plug in as input to the code hamburger cooking guidelines, and evaluate to what extent does our code

support the recommendations provided for the particular recipes chosen.

In Chapter four, we move away from thermal systems into structural systems, in which we use the finite element method for one dimensional fourth order bending boundary value problem. The finite element method is used to approximate the deflection of an Euler Bernoulli bar (a sufficiently slender bar). However, for the bending case, we require approximations for both the displacement (deflection), and it's first derivative in terms of x (Hermitian approximations). Hence both these pieces of information need to be captured in the 'weight' vector when solving our linear system of equations that achieves energy functional minimization, via a double index to single index mapping. We employ two sets of base functions, one set captures the the function values, and the other set captures the value of the derivatives. We translate our bar PDE into an eigenproblem, since displacement of the bar along the length of the bar/beam is produced from a combination of the bar's (infinite) vibrational modes. This means we would first find the eigenvalue corresponding to a particular mode, then we solve for the displacement along the beam (the eigenfunction) corresponding to the eigenvalue. We consider the particular case study of a xylophone bar, following the Euler Bernoulli model. We optimize the length of the bar such that it is tuned to have a particular fundamental frequency (the playing note or pitch), and we also manipulate the shape of the bar to produce a particular frequency ratio R, the frequency of the first harmonic frequency to the fundamental frequency. Finally, we find the position of the holes in the bar where the string will pass through such that the holes pass through the nodes where there is zero bar displacement to minimize the force of the strings on the bar. We were also capable of verifying correct numerical implementation of our code by comparing our optimized bar properties and frequencies with those obtained from an external source (Caresta's paper). Also, just as in the previous chapters, we show how boundary conditions could be incorporated into our system we are solving for.

Finally, for chapter 5, we consider another one dimensional fourth order bending boundary value problem, which is self-buckling. Self-buckling is failure mode in which a column buckles due to its own weight with no other direct forces acting on it. Again, we translate the BVP into an eigenproblem. The critical load at which buckling load occurs corresponds to a particular eigenvalue. The finite element method can be used to find the eigenvalue (critical load) for a particular column and the corresponding deflection due to buckling. For task 5, we find the radius profile for a column of circular cross section, subject to certain constraints that maximizes the self-buckling critical load, thereby maximizing the critical height before self-buckling occurs.

"Lecture notes" shall refer in general to the MIT subject 2.S976 Lecture Notes (AT Patera, 2019) on MIT Stellar.

1 Chapter One: The Rayleigh-Ritz Method

1.1 Model One

Model One corresponds to quasi-1D heat conduction in a conical frustrum insulated on the lateral surfaces with heat flux and heat transfer coefficient boundary conditions on the left and right surfaces, respectively. Since we are provided with heat flux at the first boundary (derivative of required temperature solution), and heat transfer coefficient at the second boundary, we can consider those as our Neumann/Robin (N/R) boundary conditions imposed on our PDE and we approach the approximation to the second order boundary value problem (BVP) via such case accordingly. The BVP for the 1-D temperature profile across the conical frustrum is given by:

$$-k\frac{d}{dx}(\pi R_0^2(1+\beta\frac{x}{L})^2\frac{du}{dx}) = 0 \qquad in\ \Omega \tag{1}$$

where $\Omega = (0, L)$. Equation 73 could be rearranged to the more familiar form encountered in class for (N/R) BVPs. Taking πR_0^2 out of the derivative, inserting k inside the derivative (since both are constants) and dividing both LHS and RHS by πR_0^2:

$$\frac{-d}{dx}\Big(\underbrace{k(1+\beta\frac{x}{L})^2}_{k(x)}\frac{du}{dx}\Big) + \underbrace{0}_{\mu(x)}u = \underbrace{0}_{f_\Omega} \tag{2}$$

The first boundary condition at the left surface, in which the flux is imposed, is given by:

$$k\frac{du}{dx} = -q_1 \qquad on \quad \tau_1 \tag{3}$$

Multiply both sides of equation 75 by $(1+\beta\frac{x}{L})^2$ to get:

$$\underbrace{k(1+\beta\frac{x}{L})^2}_{k(x)}\frac{du}{dx} = -(1+\beta\frac{x}{L})^2 q_1 \qquad on \quad \tau_1 \tag{4}$$

However, given that surface τ_1 occurs at x=0, equation 76 can be rewritten as:

$$\underbrace{k(1+\beta\frac{x}{L})^2\Big|_{x=0}}_{k(x=0)}\frac{du}{dx} = \underbrace{0}_{\gamma_1} - \underbrace{q_1}_{f_{\tau_1}} \qquad (on \quad \tau_1) \tag{5}$$

Similarly, the boundary condition at the second surface τ_2 to the right which is:

$$-k\frac{du}{dx} = \eta_2(u - u_\infty) \quad on \ \ \tau_2 \tag{6}$$

can be multiplied by $(1+\beta\frac{x}{L}^2)$ to get:

$$\underbrace{-k(1+\beta\frac{x}{L})^2\Big|_{x=L}}_{k(x=L)}\frac{du}{dx} = \underbrace{(1+\beta)^2\eta_2}_{\gamma_2}u - \underbrace{\eta(1+\beta)^2u_\infty}_{f_{r_2}} \qquad (on\ \ \tau_2) \tag{7}$$

Table 1 provides the values of the parameters of the (N/R) second order BVP, in terms given constants of the problem in model one

2^{nd} Order BVP Parameter	Expression in given constants	Significance
$\mu(x)$	0	heat loss through lateral surface
$f_\Omega(x)$	0	heat source/generation term
f_{r_1}	q_1	heat flux at τ_1
f_{r_2}	$\eta_2(1+\beta)^2u_\infty$	heat flux at τ_2
γ_1	0	heat transfer coefficient at τ_1
γ_2	$\eta_2(1+\beta)^2$	heat transfer coefficient at τ_2
k(x)	$k(1+\beta\frac{x}{L})^2$	heat conductivity

Table 1: Values of Second Order BVP parameters in terms of model one problem constants

To provide the energy functional minimization of which yields the solution u, let us examine a candidate function, w(x), where $x \in \Omega$, which should meet the following conditions, or, in other words, fall within the sample space $H(\omega)$ which contains all candidate functions meeting the following conditions appropriate for a (N/R) boundary value problem:

$$H(\Omega): \begin{cases} \int_0^L w^2\,dx < \infty \\ \int_0^L (\frac{dw}{dx})^2\,dx < \infty \end{cases} \tag{8}$$

The constraints of the sample space within which the candidate function lies seem to be pretty intuitive; they do impose some sort a finite limit for the value of the candidate function integral throughout the system (bearing in mind that the candidate function represents the 1-D temperature profile T(x)), as well as impose a limit on the possible heat flux through the system by imposing a limit on $\frac{dw}{dx}$. Now, we can write an energy functional $\pi(w)$ in terms of the candidate function w(x) that satisfies the sample space H(Ω). This energy functional 'describes' the total energy of the state of our system (the total heat energy of our frustum), and it is something we need to minimize (i.e find the function w(x) that minimizes $\pi(w)$) in order to arrive to our steady state solution. We use an appropriate energy functional form for a (N/R) second order BVP provided in the course notes:

$$\begin{aligned}
\pi(w) &= \frac{1}{2}\int_0^L k(x)(\frac{dw}{dx})^2 + \cancel{\mu(x)}w^2\,dx + \frac{1}{2}\Big(\gamma_1\cancel{w(0)} + \gamma_2w^2(L)\Big) - \cancel{\int_0^L f_\Omega w\,dx} - w(0)f_{r_1} - w(L)f_{r_2} \\
\pi(w) &= \frac{1}{2}\int_0^L k(x)(\frac{dw}{dx})^2\,dx + \frac{1}{2}\gamma_2w^2(L) - w(0)f_{r_1} - w(L)f_{r_2} \\
&= \frac{1}{2}\int_0^L k(1+\beta\frac{x}{L})^2(\frac{dw}{dx})^2\,dx + \frac{1}{2}\eta_2(1+\beta)^2w^2(L) - w(0)q_1 - w(L)\eta_2(1+\beta)^2u_\infty
\end{aligned} \tag{9}$$

Consider the energy functional of the actual solution $\pi(u)$; u(x) is the candidate function that actually minimizes the energy functional in equation 9:

$$\begin{aligned}
u = argmin\ \pi(w) \qquad & w \in H(\Omega) \\
or\ \ \pi(w) > \pi(u) \qquad & \forall w \in H(\Omega), w \neq u
\end{aligned} \tag{10}$$

Why is this so? Consider the energy functional of the desired solution with a slight perturbation: $\pi(u+v)$. such that $v \in H(\Omega)$ and $(u+v) \in H(\Omega)$. Then evaluating the energy functional of (u+v) by plugging in (u+v) in place of w in equation 9 using the given constants in the problem:

$$\pi(u+v) = \frac{1}{2}\int_0^L k(1+\beta\frac{x}{L})^2\Big(\frac{d(u+v)}{dx})^2\Big)\,dx + \frac{1}{2}\gamma_2\Big(u(L)+v(L)\Big)^2 - \Big(u(0)+v(0)\Big)f_{r_1} - \Big(u(L)+v(L)\Big)f_{r_2} \quad (11)$$

$$= \frac{1}{2}\int_0^L k(1+\beta\frac{x}{L})^2(\frac{du}{dx})^2\,dx + \frac{1}{2}\gamma_2 u^2(L) - u(0)f_{r_1} - u(L)f_{r_2} \quad (12)$$

$$+ \int_0^L k(1+\beta\frac{x}{L})^2\frac{du}{dx}\frac{dv}{dx}\,dx + \gamma_{-2}u(L)v(L) - fr_1v(0) - f_{r_2}v(L) \quad (13)$$

$$+ \frac{1}{2}\Big(\int_0^L k(1+\beta\frac{x}{L})^2(\frac{dv}{dx})^2\,dx + \gamma_2 v^2(L)\Big) \quad (14)$$

Equation 12 is basically $\pi(u)$, referred to as E_I. Equation 13 is refered to as E_{II}, and can be shown to equal zero. We use integration by parts, to find the integral in E_{II} , where we take the derivative of $k(1+\beta\frac{x}{L})^2\frac{du}{dx}$, and integrate $\frac{dv}{dx}$:

$$\int_0^L k(1+\beta\frac{x}{L})^2\frac{du}{dx}\frac{dv}{dx}\,dx = \Big[k(1+\beta xL)^2\frac{du}{dx}v(x)\Big]\Big|_{x=0}^{L} + \int_0^L \cancel{-\frac{d}{dx}\Big(k(1+\beta\frac{x}{L})^2\frac{du}{dx}\Big)v(x)\,dx} \quad (15)$$

Where the term crossed in the integral in equation 15 is simply the equation of our BVP system in equation 74, which is equal to zero. Putting what is left from equation 15 into equation 13 and rearranging we get:

$$E_{II} = v(L)\cancel{\Big(\gamma_2 u(L) - f_{r_2} + \Big[k(1+\beta\frac{x}{L})^2\frac{du}{dx}\Big]\Big|_{x=L}\Big)} + v(0)\cancel{\Big(-k(1+\beta\frac{x}{L})^2\Big|_{x=0} - f_{r_1}\Big)} = 0 \quad (16)$$

Where the first cancelled term in equation 16 is the boundary condition at τ_2 in equation 79, and the second cancelled is the boundary condition on τ_1 in equation 76. Hence E_{II} vanishes to zero. We can refer to equation 14 as $\frac{1}{2}E_{III}$, which, by inspection, is greater than zero since it involves all squared terms and $v(x) > 0$.

All this boils down to the following:

$$\begin{aligned} &\pi(w) = \pi(u+v) = \pi(u) + \frac{1}{2}E_{III} > \pi(u) \\ &\forall w, v \in H(\Omega), w \neq u, v \neq 0 \end{aligned} \quad (17)$$

Hence, the exact candidate function (i.e exact temperature profile solution) is the one that minimizes the energy functional given in equation 9. However, we do not know (we actually do but that is not the point) the exact candidate function, and so we need to come up with best approximation to the exact solution without solving for the exact solution. For this, consider the interpretation of E_{III}(u-w):

$$E_{III}(u-w) = \int_0^L k(1+\beta\frac{x}{L})^2(\frac{d}{dx}(u-w))^2\,dx + \gamma_2\big(u(L)-w(L)\big)^2 \quad (18)$$

The term in equation 18 contains the two weighted 'root square mean' terms squared over the interval (0,L), the first signifying the error in the derivative of w, or we could think about it as the error in the flux, and the second is the error of the candidate function itself evaluated at x=L. Hence, by the 'error' norm defined in E_{III}, the greater the error in the approximation, the greater the E_{III} of that candidate function. Looking at equation 17, and bearing in mind that $E_{III} > 0$, we arrive to the following: the greater the error in the candidate function from the true solution, the greater the value of E_{III} for that function, and hence the greater the energy functional for the candidate function. Via this logic, if we want to better approximate the exact function solution, we then try as best as we can to come up with the candidate function that most minimizes the energy functional. The exact math for the proof is in course notes but it is along the same lines.

From the previous discussion, the lower the energy functional of the candidate function we arrive to, the better the candidate function is as an approximation to the exact solution. This is where the Rayleigh Ritz method comes

in. We can view a function as the weighted summation of other known functions. Ofcourse, if we don't know the original function in most cases it could take us an infinite number of terms (functions) to exactly evaluate the original function. However, with the right choice of those functions (which I do not do here), a limited number of functions, n^{RR}, we could get a fairly decent approximation to the solution. In the RR method, an approximate form of the solution is assumed in terms of series containing functions and unknown coefficients. We substitute these weighted functions in the summation formulation in place of the candidate function in the energy functional term, and we get a set of algabraic equations in terms of the unknown 'weight' coefficient, that we solve to obtain the set of coefficients that minimizes the energy functional and thus best approximates the solution given the predetermined set of functions used. Of course, these functions should be linearly independent, otherwise we could get multiple sets of coefficients that basically give the same candidate function and MATLAB could crash.

We now proceed to write our RR candidate function, u^{RR}, in terms of n^{RR} basis functions:

$$\begin{aligned} & u^{RR} = \Sigma_{i=1}^{n^{RR}} \alpha_i^{RR} \Psi_i(x) \\ & \Big\{ \Psi_1, \Psi_2, ..., \Psi_{n^{RR}} \Big\} \in H(\Omega) \\ & \underline{\alpha}^{RR} = (\alpha_1^{RR}\ \alpha_2^{RR}\ ...\ \alpha_{n^{RR}}^{RR}) \in \mathbb{R}^{\mathrm{n^{RR}}} \end{aligned} \tag{19}$$

And now we find $\underline{\alpha}^{RR}$ such that:

$$\pi(\Sigma_{i=1}^{n^{RR}} \alpha_i^{RR} \Psi_i) < \pi(\Sigma_{i=1}^{n^{RR}} \alpha_i \Psi_i) \tag{20}$$

For any possible combination of the LHS. Now consider the general algebraic form of energy for any size of n^{RR}:

$$\begin{aligned} \pi(\Sigma_{i=1}^{n^{RR}} \alpha_i \Psi_i) &= \frac{1}{2} \int_0^L k(x) (\Sigma_{i=1}^{n^{RR}} \frac{d}{dx} \alpha_i \Psi_i)^2 \, dx + \frac{1}{2} \Sigma_{i=1}^{n^{RR}} \gamma_2 \alpha_i^2 \Psi_i^2(L) - \Sigma_{i=1}^{n^{RR}} \alpha_i \Psi_i(0) f_{r_1} - \Sigma_{i=1}^{n^{RR}} \alpha_i \Psi_i(L) f_{r_2} \\ &= \frac{1}{2} \int_0^L k(x) (\Sigma_{i=1}^{n^{RR}} \frac{d}{dx} \alpha_i \Psi_i) (\Sigma_{j=1}^{n^{RR}} \frac{d}{dx} \alpha_j \Psi_j) \, dx + \frac{1}{2} \gamma_2 \Big(\Sigma_{i=1}^{n^{RR}} \alpha_i \Psi_i(L) \Big) \Big(\Sigma_{j=1}^{n^{RR}} \alpha_j \Psi_j(L) \Big) - \Sigma_{i=1}^{n^{RR}} \alpha_i \Psi_i(0) f_{r_1} \\ &- \Sigma_{i=1}^{n^{RR}} \alpha_i \Psi_i(L) f_{r_2} \\ &= \frac{1}{2} \Sigma_{i=1}^{n^{RR}} \Sigma_{j=1}^{n^{RR}} \alpha_i \Big(\underbrace{\int_0^L k(x) \frac{d\Psi_i}{dx} \frac{d\Psi_j}{dx} \, dx + \gamma_2 \Psi_i(L) \Psi_j(L)}_{A_{ij} \in R^{n^{RR} x n^{RR}}} \Big) \alpha_j - \Sigma_{i=1}^{n^{RR}} \alpha_i \Big(\underbrace{\Psi_i(0) f_{r_1} - \Psi_i(L) f_{r_2}}_{F_i \in R^{n^{RR}}} \Big) \\ &= \frac{1}{2} \alpha^T A \alpha - \alpha^T F = Q(\pi) \end{aligned} \tag{21}$$

Matrix a is an SPD matrix (proof in course notes) of size $n^{RR} \times n^{RR}$, while vector F is of size n^{RR}. Now we need to minimize $Q(\pi)$, so we first find its derivative in terms of the general α_k:

$$\begin{aligned} \frac{\partial Q_\pi}{\partial \alpha_k} &= \frac{\partial}{\partial \alpha_k} (\frac{1}{2} \Sigma_{i=1}^{n^{RR}} \Sigma_{j=1}^{n^{RR}} \alpha_i \alpha_j A_{ij} - \Sigma_{i=1}^{n^{RR}} \alpha_i F_i) \\ &= \frac{1}{2} \Sigma_{i=1}^{n^{RR}} \Sigma_{j=1}^{n^{RR}} (\alpha_i \frac{\partial \alpha_j}{\partial \alpha_k} + \frac{\partial \alpha_i}{\partial \alpha_k} \alpha_j) A_{ij} - \Sigma_{i=1}^{n^{RR}} \frac{\partial \alpha_i}{\partial \alpha_k} F_i \\ &= \frac{1}{2} \Sigma_{i=1}^{n^{RR}} \alpha_i A_{ik} + \frac{1}{2} \Sigma_{j=1}^{n^{RR}} \alpha_j A_{kj} - F_k \end{aligned} \tag{22}$$

Now using the fact that the matrix is SPD such that $A_{ij} = A_{ji}$, we can write equation 22 in the following:

$$\begin{aligned}\frac{\partial Q_\pi}{\partial \alpha_k} &= \frac{1}{2}\Sigma_{i=1}^{n^{RR}} A_{ki}\alpha_i + \frac{1}{2}\Sigma_{j=1}^{n^{RR}} A_{kj}\alpha_j - F_k \\ &= \Sigma_{i=1}^{n^{RR}} A_{ki}\alpha_i - F_k \\ &= \left(\underline{A}\alpha - \underline{F}\right)_k \quad 1 \le k \le n^{RR}\end{aligned} \tag{23}$$

Setting the result of equation 23 to zero to find the coefficient vector that minimizes the system of linear equations:

$$\begin{aligned}\frac{\partial Q_\pi}{\partial \alpha_k}(\alpha^{RR}) &= 0 \\ \underline{A}\alpha_{RR} &= \underline{F}\end{aligned} \tag{24}$$

And the fact that the second derivative provides a positive definite verifies that solving the system of equations in equation 106 provides a minimum for $Q(\pi)$:

$$\begin{aligned}\frac{\partial^2 Q(\pi)}{\partial \alpha_l \partial \alpha_k} &= \frac{\partial}{\partial \alpha_l}\left(\Sigma_{i=1}^{n^{RR}} A_{ki}\alpha_i - F_k\right) \\ &= \Sigma_{i=1}^{n^{RR}} A_{ki}\frac{\partial \alpha_i}{\partial \alpha_l} \\ &= A_{kl} = \left(\underline{A}\right)_{kl} \quad 1 \le k, l \le n^{RR}\end{aligned} \tag{25}$$

So all of the RR method boils down to solving for α_{RR} in 106 for particular function basis, in which the values of A_{ij} and F_i are given by (after substituting in the parameter values in table 1:

$$\begin{aligned}A_{ij} &= \int_0^L k(1+\beta\frac{x}{L})^2 \frac{d\Psi_i}{dx}\frac{d\Psi_j}{dx}\,dx + \eta_2(1+\beta)^2\Psi_i(L)\Psi_j(L) \\ F_i &= \Psi_i(0)q_1 - \Psi_i(L)\eta_2(1+\beta)^2 u_\infty\end{aligned} \tag{26}$$

Lets us now write the values of the entries of matrix A and vector F in equation 26 in terms of the non-dimensional variable $\zeta = \frac{x}{L}$. Few things to bear in mind:

$$\begin{aligned}dx &= L d\zeta \\ \frac{d\Psi}{dx} &= \frac{d\Psi}{d\zeta}\cdot\frac{d\zeta}{dx} = \frac{d\Psi}{d\zeta}\cdot\frac{1}{L}\end{aligned}$$

Accordingly, the expressions in equation 26 become:

$$\begin{aligned}A_{ij} &= \int_0^1 \frac{k}{L}(1+\beta\zeta)^2 \frac{d\Psi_i}{d\zeta}\frac{d\Psi_j}{d\zeta}\,d\zeta + \eta_2(1+\beta)^2\Psi_i(1)\Psi_j(1) \\ F_i &= \Psi_i(0)q_1 - \Psi_i(1)\eta_2(1+\beta)^2 u_\infty\end{aligned} \tag{27}$$

The exact solution for model 1 is given by:

$$u = u_\infty + \frac{q_1 L}{k}\left(\frac{1+\beta+\frac{k}{\eta_2 L}}{(1+\beta)^2} - \frac{\frac{x}{L}}{1+\beta\frac{x}{L}}\right) \tag{28}$$

substituting $\zeta = \frac{x}{L}$ we get for the exact solution:

$$u_\infty + \frac{q_1 L}{k}\left(\frac{1+\beta+\frac{k}{\eta_2 L}}{(1+\beta)^2} - \frac{\zeta}{1+\beta\zeta}\right) \tag{29}$$

And the derivative to the simplified version of the exact solution in equation 29 is given by:

$$\frac{du}{d\zeta} = -\frac{q_1 L}{k(1+\mu\zeta)^2} \tag{30}$$

First, we use two basis functions ($n_{RR} = 2$) in the MATLAB code provided; the first one is the exact solution in 29, $\Psi_1 = u$, and the other is the linear function $\Psi_2 = x$. Table 2 summarizes the values of the parameters used in the code (with the exception of β, which is varied in the code):

Parameter	Value	Unit
k	0.5	$\frac{W}{m.C}$
q_1	100	$\frac{W}{m^2}$
η_2	100	$\frac{W}{m^2.C}$
L	0.1	m
u_∞	24	C

Table 2: Values of model 1 constants

If our work so far is actually right, then applying the RR method should give us $u^{RR} = u$, where all the weight is on the first basis function ($\alpha_1^{RR} = 1$), and no weight is given to the second basis function ($\alpha_2^{RR} = 0$). And indeed, when we run the code for $\mu = 100$, we do get:

$$\begin{aligned} \alpha_1^{RR} &= 1 \\ \alpha_2^{RR} &= 0 \end{aligned} \tag{31}$$

Which at least means our method and derivation are right. This places all of the weight on the exact solution basis function Ψ_1, and no weight on the linear function Ψ_2, as shown in figure 1:

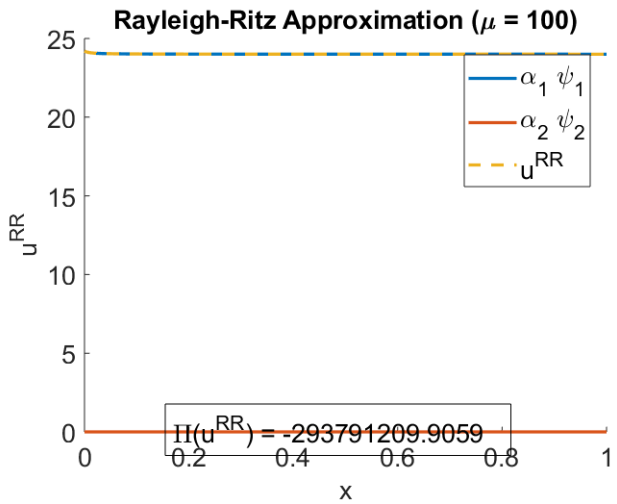


Figure 1: RR approximation for model 1 using: $\Psi_1 = u$ and $\Psi_2 = x$, $\beta = 100$

And Hence, we expect the calculated u^{RR} to match the exact solution, as shown in figure 2. The output was defined to be the temperature at the left boundary (i.e when λ_1 =1 and $\lambda_2 = 0$ in equation 80).

$$
\begin{aligned}
s &= \lambda_1 u\Big|_{\Gamma_1} + \Gamma_2 u\Big|_{\Omega_2} \\
&= u(0)
\end{aligned}
\tag{32}
$$

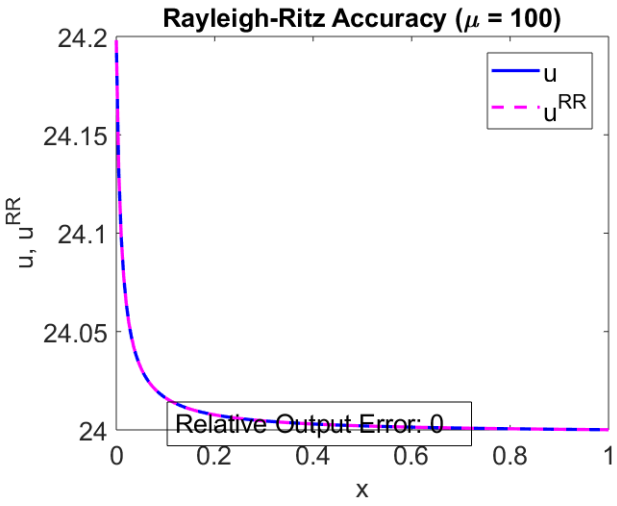


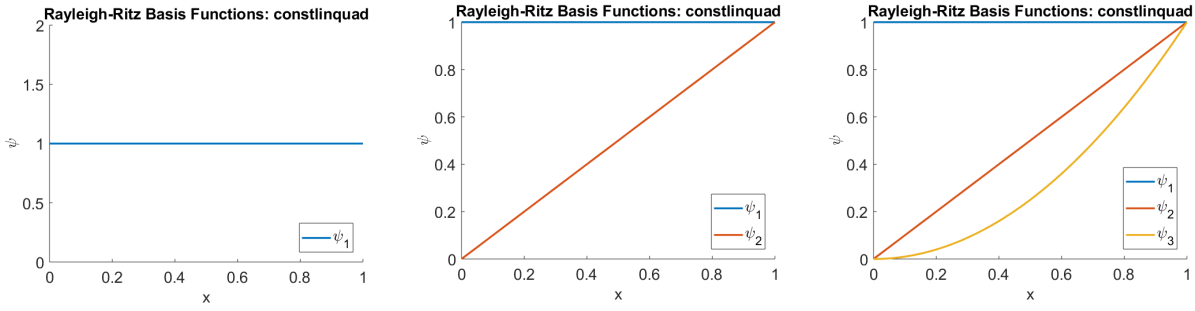
Figure 2: RR accuracy for model 1 using: $\Psi_1 = u$ and $\Psi_2 = x$, $\beta = 100$

Examining the values of the RR coefficients (also presented in figure 1), we could see that indeed the values are in agreement with what we expected. Examining figure 2, our derived u_{RR} is in agreement with the exact solution.

Well this is not very interesting since we plugged the actual solution as one of the basis functions. It only verifies our derivations and code are accurate. Lets us try using more general base function:

$$
\begin{aligned}
n^{RR} = 1: &\quad \Psi_1 = 1 \\
n^{RR} = 2: &\quad \Psi_1 = 1, \Psi_2 = x \\
n^{RR} = 3: &\quad \Psi_1 = 1, \Psi_2 = x, \Psi_3 = x^2
\end{aligned}
\tag{33}
$$

The Basis functions for model one are shown in figure 3. Now we use the MATLAB code to come up with our own RR function approximation u^{RR} as shown in figure 4 (note that in figure 4c, $\alpha_2\Psi_2$ and $\alpha_3\Psi_3$ overlap hence only appear as one color). Finally, we plot our function approximation u^{RR} along with the exact solution in figure 5 to evaluate the accuracy of our approximations.



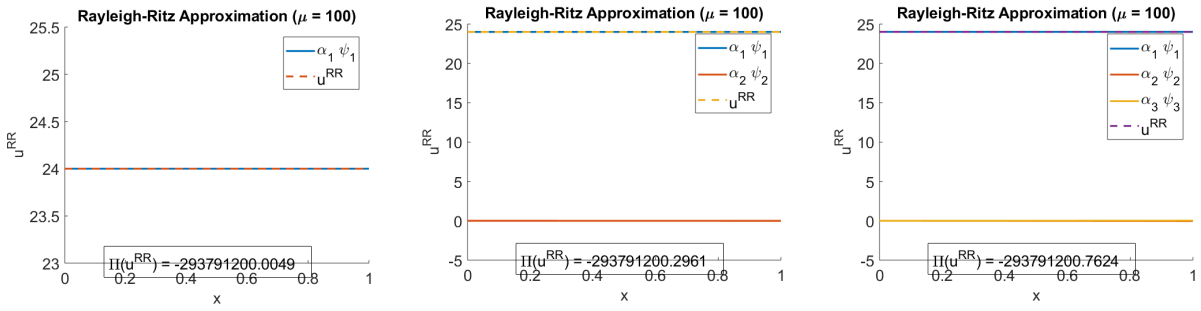
(a) Basis Functions: $\Psi_1 = 1$

(b) Basis functions $\Psi_1 = 1$ and $\Psi_2 = x$

(c) Basis functions $\Psi_1 = 1$ and $\Psi_2 = x$ and $\Psi_3 = x^2$

.

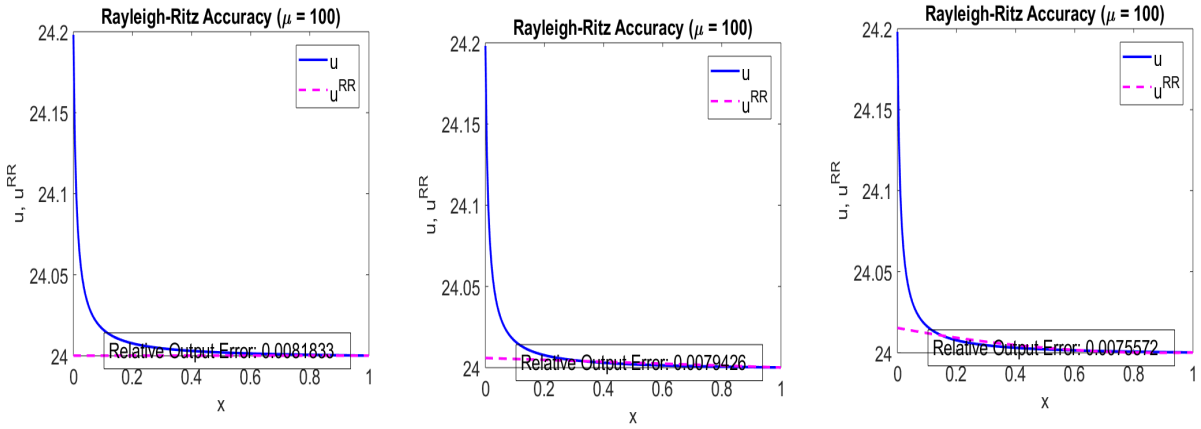
Figure 3: Basis functions for model 1 ($n^{RR} = 1, 2, 3$) $\beta = 100$



(a) u_{RR} approximation: $\Psi_1 = 1$ (b) u_{RR} approximation $\Psi_1 = 1$ and $\Psi_2 = x$ (c) u_{RR} approximation $\Psi_1 = 1$ and $\Psi_2 = x$ and $\Psi_3 = x^2$

.

Figure 4: u_{RR} approximations for model 1 ($n^{RR} = 1, 2, 3$) $\beta = 100$



(a) u_{RR} accuracy: $\Psi_1 = 1$ (b) u_{RR} accuracy $\Psi_1 = 1$ and $\Psi_2 = x$ (c) u_{RR} accuracy $\Psi_1 = 1$ and $\Psi_2 = x$ and $\Psi_3 = x^2$

.

Figure 5: u_{RR} Accuracies for model 1 ($n^{RR} = 1, 2, 3$) $\beta = 100$

Table 3 summarizes the values of the energy functionals, as well as the relative output error when compared to the actual solution:

u^{RR}	n^{RR}	$\pi(u^{RR})$	Relative Output Error
$\Psi_1 = u, \Psi_2 = x$	2	-293791209.9059	0
$\Psi_1 = 1$	1	-293791200.0049	0.0081833
$\Psi_1 = 1, \Psi_2 = x$	2	-293791200.2961	0.0.0079426
$\Psi_1 = 1, \Psi_2 = x, \Psi_3 = x^2$	3	-293791200.7624	0.0075572

Table 3: Values of the functional energy and relative output error for the RR functions for model 1, β=100.

The lower the value of $\pi(u^{RR})$, the less the relative output error, and the more accurate our RR approximation is. From table 3, we can see that using a RR approximation that includes the exact solution as a base function provides a very accurate approximation to the actual solution, and hence it exhibits the lowest value of the energy functional as well as the lowest error. For our RR approximations that do not include the exact solution as a basis function, the more basis functions (n^{RR}) that we add to our approximation, the more accurate our approximation is, as seen from the functional energy and error output values in table 3. This makes sense because the RR minimization method would not assign any weight (zero α^{RR}) to any added base functions if these base functions would result in a higher energy functional and thus less accurate approximation. In other words, adding extra (linearly independent) functions to the original basis functions would either, at worse, maintain the accuracy of the original approximation, or, ideally, improve it.

Another interesting question is: how does the value of β affect the difference between the exact solution and the Rayleigh-Ritz approximation? To investigate this, we vary the value of β between 0,0.5,0.7, 1,100, and 1600, across

the different number of base functions n^{RR}. Table 4 summarizes the values of the energy functional and the relative output error when varying the values of both β and n^{RR} (note, while the analysis for trends and conclusions included β values of 0.5 and 0.7 too, I did not include them in table so as not to make it unnecessarily crowded). We notice that while maintaining the same value of β, the approximation error still decreases (less energy functional and relative output error) as the number of basis functions increase, as expected. However, at higher values of β (100 and 1600), the difference in accuracy as we use more base functions becomes more and more neglegible, as we see in figure 8b; this is because at higher values of β the solution is almost effectively constant, and thus any extra base functions apart from the constant base function Ψ_1 don't improve the accuracy of the RR approximation by much.

u^{RR}	n^{RR}	β	$\pi(u^{RR})$	Relative Output Error
$\Psi_1 = u, \Psi_2 = x$	2	0	-32250	1.1053×10^{-15}
$\Psi_1 = 1$	1	0	-31250	0.44444
$\Psi_1 = 1, \Psi_2 = x$	2	0	-32250	1.579×10^{-16}
$\Psi_1 = 1, \Psi_2 = x, \Psi_3 = x^2$	3	0	-32250	1.579×10^{-16}
$\Psi_1 = u, \Psi_2 = x$	2	1	-118112.5	2.0746×10^{-16}
$\Psi_1 = 1$	1	1	-117612.5	0.29197
$\Psi_1 = 1, \Psi_2 = x$	2	1	-118041.0714	0.04171
$\Psi_1 = 1, \Psi_2 = x, \Psi_3 = x^2$	3	1	-118107.3454	0.00301
$\Psi_1 = u, \Psi_2 = x$	2	100	-293791209.9059	0
$\Psi_1 = 1$	1	100	-293791200.0049	0.0081833
$\Psi_1 = 1, \Psi_2 = x$	2	100	-293791200.2961	0.0079426
$\Psi_1 = 1, \Psi_2 = x, \Psi_3 = x^2$	3	100	-293791200.7624	0.0075572
$\Psi_1 = u, \Psi_2 = x$	2	1600	-73820191200.6247	2.9591×10^{-16}
$\Psi_1 = 1$	1	1600	-73820191200.0000	0.00052024
$\Psi_1 = 1, \Psi_2 = x$	2	1600	-73820191200.0012	0.0005199
$\Psi_1 = 1, \Psi_2 = x, \Psi_3 = x^2$	3	1600	-73820191200.0031	0.00051764

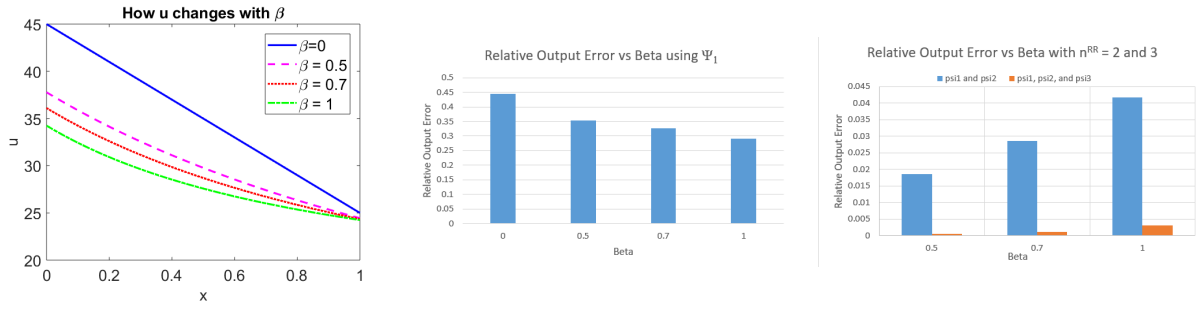
Table 4: Values of the functional energy and relative output error for the RR functions for model 1

What I believe is that the effect of the value of β on accuracy really depends on the range of β values you are investigating. For instance, for $\beta = 0$, the exact solution essentially becomes a linear function, hence, we expect that when incorporating $\Psi_2 = x$ into the base functions (as with $n^{RR} = 2$ and $n^{RR} = 3$), we will essentially get the exact solution with the same energy functional value and zero error, as seen in table 4 (i.e all weight will be assigned to α_2).

We could also see that between β values of 0 and 1 (excluding $n^{RR} = 2, n^{RR} = 3$ for $\beta = 0$ due to the reason explained above), the behavior also depends on what basis functions we are using. When using only Ψ_1, for instance, the RR approximation of the solution is basically a constant, so we would expect that for the decaying exact solution behavior, as the difference between the maximum and minimum values of exact solution increases, the approximation error would also increase because we are now approximating a solution that spans a greater range with a constant. As β increases within this range, the difference between maximum and minimum decreases as seen in figure 7a. Hence, error would decrease as β increases for this particular range as seen in figure 7b.

However, once we are no longer approximating our function using only a constant function, the accuracy of the RR approximation decreases as β increases for β from 0 to 1. This is because as β increases, the decay of the exact solution exhibits greater curvature, making it harder to approximate such curvature using the monomial constant, linear and quadratic basis functions, as shown in figures 6 a and c.

However, at high values of β, as mentioned, the solution basically approaches a constant value, as shown in figure 8a, which is basically easier to approximate using our choice of functions. Hence, at high values of β such as 100 and 1600, the higher the value of β, the more the exact solution approaches a constant function behavior, and the better approximated it is using the RR method (also the less added value of using Ψ_2 and Ψ_3 as mentioned), as shown in figure 7b.

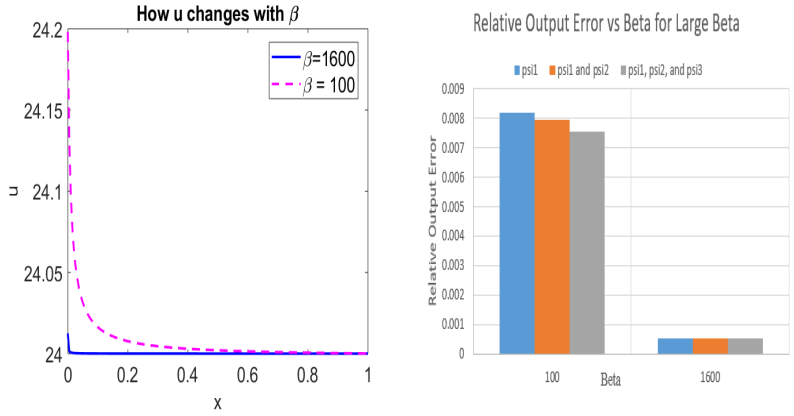


(a) Exact solution with different β

(b) Error vs β for Ψ_1

(c) Error vs β for $n^{RR} = 2, 3$

Figure 6: Behavior of exact solution function (subfigure a), as well as the relative output error across the different base functions, as β changes from 0 to 1 for model 1.



.

Figure 7: Behavior of exact solution and approximated solution functions, as well as relative error for Large β for model 1

1.2 Model Two

Model Two corresponds to a right-cylinder thermal fin with temperature and zero-flux boundary conditions on the left and right surfaces, respectively. Since we are provided with a temperature boundary condition at the first boundary, and the (effective) flux at the second boundary, we consider those as Dirichlet-N/R boundary conditions imposed on our ODE and we approach the approximation to the second order BVP via such case.

The BVP for the right-cylinder thermal fin is given by equation 82

$$-KA_{cs}\frac{d^2u}{dx^2} + \eta_3 P_{cs}(u - u_\infty) = 0 \qquad in\ \Omega \tag{34}$$

Where, just like model one, $\Omega = (0, L)$, and just like in model one, the BVP equation could be rearranged to the more familiar form encountered in class for Dirichlet-(N/R) BVPs, as in equation 83

$$-\frac{d}{dx}\underbrace{(kA_{cs})}_{k(x)}\frac{du}{dx} + \underbrace{\eta_3 P_{cs}}_{\mu(x)}u = \underbrace{\eta_3 P_{cs} u_\infty}_{f_\Omega(x)} \tag{35}$$

And just like in model 1, we could rewrite the boundary conditions as the following:

$$u\Big|_{x=0} = u_{\Gamma_1} \qquad on\ \Gamma_1 \tag{36}$$

$$\underbrace{-kA_{cs}}_{k(x)} \frac{du}{dx}\Big|_{x=L} = \underbrace{0}_{\gamma_2 u - f_{\gamma_2}} \qquad on\ \Gamma_2 \tag{37}$$

We impose the simple condition satisfying equation 85 that $\gamma_2 = f_{\Gamma_2} = 0$. Table 5 provides the values of the parameters to be used in terms of given constants in the problem.

2^{nd} Order BVP Parameter	Expression in given constants	Significance
$\mu(x)$	$\eta_3 P_{cs}$	heat loss through lateral surface
$f_\Omega(x)$	$\eta_3 P_{cs} u_\infty$	heat source/generation term
f_{r_2}	0	heat flux at τ_2
γ_2	0	heat transfer coefficient at τ_2
k(x)	kA_{cs}	heat conductivity
μ_0	$\frac{\eta_3 P_{cs} L^2}{kA_{cs}}$	convenience constant

Table 5: Parameter values in terms of problem constants model 2

Just as in model 1, we need to find the candidate function w(x) that minimizes the energy functional (as much as we can). This candidate function must lie in the sample space H(w) for which all candidate function constraints hold true. Some of the constraints are reasonably the same as those of model 1 in equation 8, as well an extra requirement satisfying the given temperature (Dirichlet) boundary condition, as seen in equation 38.

$$\begin{aligned} H(w) &: \begin{cases} \int_0^L w^2\, dx < \infty \\ \int_0^L (\frac{dw}{dx})^2\, dx < \infty \end{cases} \\ u(0) &= u_{\Gamma_1} \end{aligned} \tag{38}$$

Furthermore, apart from the candidate functions, there is a sample space that includes constraints for the admissible function v(x):

$$\begin{aligned} H(\Omega) &: \begin{cases} \int_0^L v^2\, dx < \infty \\ \int_0^L (\frac{dv}{dx})^2\, dx < \infty \end{cases} \\ v(0) &= 0 \quad \text{on } \Gamma_1 \end{aligned} \tag{39}$$

The energy functional corresponding to a Dirichlet-(N/R) boundary condition (as given in the course notes) is given by:

$$\pi(w) = \frac{1}{2}\int_0^L k(x)(\frac{dw}{dx})^2 + \mu(x)w^2(x)\, dx + \frac{1}{2}\gamma_2 w^2(L) - \int_0^L f_\Omega(x)w(x)\, dx - w(L)f_{\Gamma_2} \tag{40}$$

We could rewrite the energy functional in terms of our problem parameters just as we did for model 1, substituting x for $\zeta = \frac{x}{L}$ and plugging in the parameters of the problem:

$$\pi(w) = \frac{1}{2}\int_0^1 \Big[\frac{kA_{cs}}{L}(\frac{dw}{d\zeta})^2 + \eta_3 P_{cs} L w^2(\zeta)\Big]\, d\zeta - \int_0^l \eta_3 P_{cs} u_\infty L w(\zeta)\, d\zeta \tag{41}$$

As in model 1, the candidate function that gives the solution is the one that minimizes the energy functional given in equation 41:

$$\begin{aligned} u &= argmin\ \pi(w) \qquad w \in H(\Omega) \\ or\ \ \pi(w) &> \pi(u) \qquad \forall w \in H(\Omega), w \neq u \end{aligned} \tag{42}$$

The reason for this is similar to the reason in model one; consider the energy functional of the desired solution with a slight perturbation: $\pi(u+v)$. such that $v \in H(\Omega)$ and $(u+v) \in H(\Omega)$. Ofcourse, this function $w(x) = u(x) + v(x)$ must also satisfy the given boundary condition at the left boundary:

$$w(0) = \underbrace{u(0)}_{u_{\Gamma_1}} + \underbrace{v(0)}_{0} \tag{43}$$

Then evaluating the energy functional of (u+v) by plugging in (u+v) in place of w in equation 41:

$$\pi(u+v) = \frac{1}{2}\int_0^1 \frac{kA_{cs}}{L}\Big(\frac{d(u+v)}{d\zeta})^2\Big) + \eta_3 P_{cs} L(u+v)^2\, d\zeta - \int_0^1 \eta_3 P_{cs} u_\infty L(u+v)^2\, d\zeta \tag{44}$$

$$= \frac{1}{2}\int_0^1 \Big[\frac{kA_{cs}}{L}(\frac{du}{d\zeta})^2 + \eta_3 P_{cs} L u^2\Big]\, d\zeta - \int_0^l \eta_3 P_{cs} u_\infty L u^2(\zeta)\, d\zeta \tag{45}$$

$$+ \int_0^1 \big[\frac{kA_{cs}}{L}\frac{du}{d\zeta}\frac{dv}{d\zeta} + \eta_3 P_{cs} L uv\big]\, d\zeta + - \int_0^1 \eta_3 P_{cs} u_\infty L v\, d\zeta \tag{46}$$

$$+ \frac{1}{2}\Big(\int_0^1 \frac{kA_{cs}}{L}(\frac{dv}{d\zeta})^2 + \eta_3 P_{cs} L v^2\ d\zeta\Big) \tag{47}$$

Equation 45 is basically $\pi(u)$, referred to as E_I. Equation 46 is referred to as E_{II}, and can be shown to equal zero. We use integration by parts, to find the integral in E_{II} , where we take the derivative of $\frac{kA_{cs}}{L}\frac{du}{d\zeta}$, and integrate $\frac{dv}{d\zeta}$:

$$\int_0^1 \Big[\frac{kA_{cs}}{L}\frac{du}{d\zeta}\Big]\frac{dv}{d\zeta}\, d\zeta = \Big[\frac{kA_{cs}}{L}\frac{du}{d\zeta}v(\zeta)\Big]\Big|_{\zeta=0}^1 + \int_0^1 -\frac{d}{d\zeta}\Big(\frac{KA_{cs}}{L}\frac{du}{d\zeta}\Big)v(\zeta)\, d\zeta \tag{48}$$

Perhaps it would be easier to relate equation 48 to our original problem by reconverting back in terms of x instead of ζ:

$$\begin{aligned} \int_0^1 \Big[\frac{kA_{cs}}{L}\frac{du}{d\zeta}\Big]\frac{dv}{d\zeta}\, d\zeta &= \Big[\frac{kA_{cs}}{\cancel{L}}\frac{du}{dx}v(x)\cancel{L}\Big]\Big|_{x=0}^L + \int_0^L -\frac{d}{dx}\Big(\frac{KA_{cs}}{\cancel{L}}\frac{du}{dx}\Big)v(x)\cancel{L}\frac{1}{\cancel{L}}\, dx \\ &= \Big[kA_{cs}\frac{du}{dx}v(x)\Big]\Big|_{x=0}^L + \int_0^L -\frac{d}{dx}\Big(KA_{cs}\frac{du}{dx}\Big)v(x)\, dx \end{aligned} \tag{49}$$

Where we have used the following simplifications:

$$d\zeta = \frac{1}{L}dx \tag{50}$$

$$\frac{d}{d\zeta}(\frac{du}{d\zeta}) = L^2\frac{d}{dx}(\frac{du}{dx}) \tag{51}$$

Plugging in equation 49 into 46 and applying the simplifications we get:

$$E_{II} = \int_0^L \cancel{\Big[\frac{-d}{dx}(kA_{cs}\frac{du}{dx}) + \eta_3 P_{cs}(u-u_\infty)\Big]} - \cancel{(-kA_{cs}\frac{du}{dx}\Big|_{x=L})}v(L) - (kA_{cs}\frac{du}{dx}\Big|_{x=0})\cancel{v(x=0)} = 0 \tag{52}$$

Where the first term cancelled in equation 52 is the given ODE for the problem, the second term cancelled is the second boundary condition stating the effective flux at $\Gamma_2 = 0$, and the third term cancelled is due to the constraint that v(x=0) = 0. Hence E_{II} vanishes to zero. We can refer to the expression in 47 as $\frac{1}{2}E_{III}$ which, by inspection, is greater than zero since it involves all squared terms and $v(\zeta)^2 > 0$ for all $\zeta \neq 0$, and we permit v $\neq 0$ for $\zeta > 0$

Exactly as in model one, all this boils down to the following:

$$\begin{aligned} &\pi(w) = \pi(u+v) = \pi(u) + \frac{1}{2}E_{III} > \pi(u) \\ &\forall w, v \in H(\Omega), w \neq u, v \neq 0 \end{aligned} \tag{53}$$

Where, just exactly as in model one, E_{III} refers to an error norm for the approximation. Hence, as in model one, the function that minimizes the given energy function in equation 40 is the one that gives the best approximation to our solution, the temperature profile u(x) for x $\in$ (x,L).

We now proceed to write our RR candidate function, u^{RR}, in terms of n^{RR} basis functions, just like in model one. However, given the Dirichlet boundary condition such that we actually do know the value of the exact solution on Γ_1, we add one extra basis function and its coefficient, which correspond to the known value of the solution at boundary Γ_1:

$$\begin{aligned} u^{RR} &= u_{\Gamma_1}\Psi_0(x) + \Sigma_{i=1}^{n^{RR}} \alpha_i^{RR}\Psi_i(x) \\ &= \Sigma_{i=0}^{n^{RR}} \tilde{\alpha}_i^{RR}\Psi_i(x) \\ &\left\{\Psi_0, \Psi_1, ..., \Psi_{n^{RR}}\right\} \in H(\Omega) \\ &\underline{\tilde{\alpha}}^{RR} = (\tilde{\alpha}_0^{RR}\ \tilde{\alpha}_1^{RR}\ ...\ \tilde{\alpha}_{n^{RR}}^{RR}) \in \mathbb{R}^{\mathrm{n^{RR}}+1} \end{aligned} \tag{54}$$

Such that by the introduction of the extra function and coefficient, the conditions in equation 55, are met:

$$\begin{aligned} &\tilde{\alpha}_0 = u_{\Gamma_1} \\ &\Psi_0(x=0) = 1 \\ &\textit{hence}\ \ u^{RR}(0) = \underbrace{\tilde{\alpha}_0\Psi_0(0)}_{u_{\Gamma_1}} + \underbrace{\Sigma_{i=1}^{n^{RR}} \alpha_i^{RR}\Psi_i(0)}_{0} \end{aligned} \tag{55}$$

And now we find $\underline{\tilde{\alpha}}^{RR}$ such that:

$$\pi(\Sigma_{i=0}^{n^{RR}} \tilde{\alpha}_i^{RR}\Psi_i) < \pi(\Sigma_{i=0}^{n^{RR}} \tilde{\alpha}_i\Psi_i) \tag{56}$$

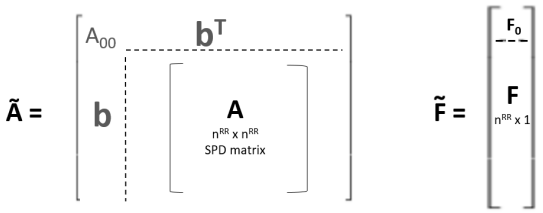
For any possible combination of the LHS. Just like in model 1, we consider the general algebraic form of energy for any size of n^{RR}:

$$\begin{aligned} \pi(\Sigma_{i=0}^{n^{RR}} \tilde{\alpha}_i\Psi_i) &= \frac{1}{2}\int_0^1 \frac{kA_{cs}}{L}(\Sigma_{i=0}^{n^{RR}} \frac{d}{d\zeta}\tilde{\alpha}_i\Psi_i)^2 + \eta_3 P_{cs} L(\Sigma_{i=0}^{n^{RR}} \tilde{\alpha}_i\Psi_i)^2\, d\zeta - \int_0^1 \eta_3 P_{cs} u_\infty L(\Sigma_{i=0}^{n^{RR}} \tilde{\alpha}_i\Psi_i)\, d\zeta \\ &= \frac{1}{2}\int_0^1 \Big[\frac{kA_{cs}}{L}(\Sigma_{i=o}^{n^{RR}} \frac{d}{d\zeta}\tilde{\alpha}_i\Psi_i)(\Sigma_{j=0}^{n^{RR}} \frac{d}{d\zeta}\tilde{\alpha}_j\Psi_j) + \eta_3 P_{cs} L(\Sigma_{i=o}^{n^{RR}} \tilde{\alpha}_i\Psi_i)(\Sigma_{j=0}^{n^{RR}} \tilde{\alpha}_j\Psi_j)\Big]\, d\zeta \\ &- \int_0^1 \eta_3 P_{cs} u_\infty L(\Sigma_{i=0}^{n^{RR}} \tilde{\alpha}_i\Psi_i)\, d\zeta \\ &= \frac{1}{2}\Sigma_{i=0}^{n^{RR}}\Sigma_{j=0}^{n^{RR}} \tilde{\alpha}_i\Big(\underbrace{\int_0^1 \frac{kA_{cs}}{L}\frac{d\Psi_i}{d\zeta}\frac{d\Psi_j}{d\zeta} + \eta_3 P_{cs} L\Psi_i\Psi_j)\, d\zeta}_{\tilde{A}_{ij}\in R^{(n^{RR}+1)\times(n^{RR}+1)}}\Big)\alpha_j - \Sigma_{i=0}^{n^{RR}} \alpha_i\Big(\underbrace{\int_0^1 \eta_3 P_{cs} u_\infty L\Psi_i\, d\zeta}_{\tilde{F}_i\in R^{n^{RR}+1}}\Big) \\ &= \frac{1}{2}\tilde{\alpha}^T\tilde{A}\tilde{\alpha} - \tilde{\alpha}^T\tilde{F} = \tilde{Q}(\pi) \end{aligned} \tag{57}$$

This is the exact system of equations we got from model 1, and Matrix $\tilde{A}$ is an SPD matrix too. However, the only difference is that now the matrix is now of size $(n^{RR}+1)\times(n^{RR}+1)$, while vector $\tilde{F}$ and $\tilde{\alpha}$ are of size $n^{RR}+1$. the values of $\tilde{A}_{ij}$ and $\tilde{F}_i$ are given by

$$\begin{aligned}\tilde{A}_{ij} &= \int_0^1 \frac{kA_{cs}}{L}\frac{d\Psi_i}{d\zeta}\frac{d\Psi_j}{d\zeta} + \eta_3 P_{cs} L \Psi_i \Psi_j)\, d\zeta \\ \tilde{F}_{ij} &= \int_0^1 \eta_3 P_{cs} u_\infty L \Psi_i \, d\zeta \\ 0 &\le i,j \le n^{RR}\end{aligned} \tag{58}$$

After construction of matrix $\tilde{A}$ and vector $\tilde{F}$, we need to extract the matrix A, and vector b from the former, and vector F from the latter, as shown below:



The following lines of psuedo code in MATLAB enable carrying out such extraction:

$\tilde{A} = A$ — *assign the constructed matrix to* $\tilde{A}$

$\tilde{F} = F$ — *assign the constructed vector to* $\tilde{F}$

$b = \tilde{A}(:,1)$ — *construct vector b from the first column of* $\tilde{A}$ *excluding the first element*

$b(1) = []$

$A(1,:) = []$ — *construct the matrix A from* $\tilde{A}$ *by deleting first row and column*

$A(:,1) = []$

$F(1) = []$ — *construct the vector F by deleting first entry from* $\tilde{F}$

$\alpha = A\backslash(F - u_{\Gamma_1} * b)$ — *solve for* α

$\tilde{\alpha} = [u_{\Gamma_1}; \alpha]$ — *construct* $\tilde{\alpha}$ *by adding* u_{Γ_1} *as first entry*

(59)

We evaluate α as $A(F - u_{\Gamma_1} * b)$ since $\tilde{Q}(\pi)$ can be written in terms of $n^{RR}\times n^{RR}$ matrix A and $n^{RR}\times 1$ vectors F and b, which is then minimized to solve for α^{RR}, as seen in equations 60, 61, 62, and 63.

$$\begin{aligned}\tilde{Q}(\pi) &= \underbrace{\tilde{\alpha}_0 A_{00}\tilde{\alpha}_0 - \tilde{\alpha}_0 F_0}_{constant} + \frac{1}{2}\alpha^T A\alpha + \alpha^T b\tilde{\alpha}_0 - \alpha^T F \\ &= C + \frac{1}{2}\alpha^T A\alpha - \alpha^T(F - u_{\Gamma_1} b)\end{aligned} \tag{60}$$

Minimizing $\tilde{Q}(\pi)$ to find α^{RR} by finding its derivative in terms of general α_k:

$$\begin{aligned}\frac{\partial Q_\pi}{\partial \alpha_k} &= \frac{\partial}{\cancel{\partial \alpha_k}} \cancel{\Sigma_{i=1}^{n^{RR}} \Sigma_{j=1}^{n^{RR}} C} + \frac{\partial}{\partial \alpha_k}(\frac{1}{2}\Sigma_{i=1}^{n^{RR}}\Sigma_{j=1}^{n^{RR}}\alpha_i\alpha_j A_{ij} - \Sigma_{i=1}^{n^{RR}}\alpha_i(F_i - u_{\Gamma_1}b_i) \\ &= \frac{1}{2}\Sigma_{i=1}^{n^{RR}}\Sigma_{j=1}^{n^{RR}}(\alpha_i\frac{\partial \alpha_j}{\partial \alpha_k} + \frac{\partial \alpha_i}{\partial \alpha_k}\alpha_j)A_{ij} - \Sigma_{i=1}^{n^{RR}}\frac{\partial \alpha_i}{\partial \alpha_k}(F_i - u_{\Gamma_1}b_i) \\ &= \frac{1}{2}\Sigma_{i=1}^{n^{RR}}\alpha_i A_{ik} + \frac{1}{2}\Sigma_{j=1}^{n^{RR}}\alpha_j A_{kj} - (F_k - u_{\Gamma_1}b_k)\end{aligned} \tag{61}$$

Just as in model one, using the fact that the matrix is SPD such that $A_{ij} = A_{ji}$, we can write equation 22 in the following:

$$\begin{aligned}\frac{\partial Q_\pi}{\partial \alpha_k} &= \frac{1}{2}\Sigma_{i=1}^{n^{RR}}A_{ki}\alpha_i + \frac{1}{2}\Sigma_{j=1}^{n^{RR}}A_{kj}\alpha_j - (F_k - u_{\Gamma_1}b_k) \\ &= \Sigma_{i=1}^{n^{RR}}A_{ki}\alpha_i - (F_k - u_{\Gamma_1}b_k) \\ &= \Big(\underline{A}\alpha - (\underline{F} - u_{\Gamma_1}\underline{b})\Big)_k \quad 1 \le k \le n^{RR}\end{aligned} \tag{62}$$

Setting the result of equation 62 to zero to find the coefficient vector that minimizes the system of linear equations:

$$\begin{aligned}\frac{\partial Q_\pi}{\partial \alpha_k}(\alpha^{RR}) &= 0 \\ \underline{A}\alpha_{RR} &= \underline{F} - u_{\Gamma_1}\underline{b}\end{aligned} \tag{63}$$

Just as in model one, the second derivative provides a positive definite, meaning that solving the system of equations in equation 63 provides a minimum for $\tilde{Q}(\pi)$.

Afterwards, we simply construct $\tilde{\alpha}^{RR}$ from α^{RR} as indicated in the pseudo code, and solve for $u^{RR} = \Sigma_{i=0}^{n^{RR}}\tilde{\alpha}_i\Psi_i$ using the chosen basis functions. However, the way we reduce the dimensions of our matrix A and vector F to be one less than the number of basis functions means we need a special case when the our base functions consist only of Ψ_0, in which case:

$$u^{RR}(\zeta) = u_{\Gamma_1}\Psi_0(\zeta) \tag{64}$$

for which there are two equivalent ways to calculate $\pi(u^{RR})$:

$$\begin{aligned}\pi(u^{RR}) &= \frac{1}{2}\int_0^1 \Big[\frac{kA_{cs}}{L}\cancel{(\frac{dw}{d\zeta})^2} + \eta_3 P_{cs} L \underbrace{w^2(\zeta)}_{1}\Big]\, d\zeta - \int_0^l \eta_3 P_{cs} u_\infty L \underbrace{w(\zeta)}_{1}\, d\zeta \\ &= \frac{1}{2}L\eta_3 P_{cs}u_{\Gamma_1}^2 - L\eta_3 P_{cs}u_\infty u_{\Gamma_1} = L\eta_3 P_{cs}(\frac{u_{\Gamma_1}}{2} - u_\infty)\end{aligned} \tag{65}$$

Or equivalently:

$$\begin{aligned}\pi(u^{RR}) &= \frac{1}{2}\underbrace{\tilde{A}}_{1\times 1}\tilde{\alpha_0}^2 - \underbrace{\tilde{F}}_{1\times 1}\tilde{\alpha}_0 \\ &= \frac{1}{2}A_{00}\tilde{u}_{\Gamma_1}^2 - F_0\tilde{u}_{\Gamma_1}\end{aligned} \tag{66}$$

The exact solution to the right cylinder fin is given by equation:

$$u(\zeta) = u_\infty + (u_{\Gamma_1} - u_\infty)\frac{cosh(\sqrt{\mu(1-\zeta)})}{cosh\sqrt{\mu}}; \tag{67}$$

Whose derivative is given by:

$$\frac{du}{d\zeta} = (u_{\Gamma_1} - u_\infty)\frac{sinh(\sqrt{\mu(1-\zeta)}\sqrt{\mu}}{cosh\sqrt{\mu}} \tag{68}$$

Which is found using the following for derivative of $cosh(\zeta)$:

$$\frac{dcosh(\zeta)}{d\zeta} = \frac{d}{d\zeta}\Big(\frac{1}{2}(e^{\zeta} + e^{-\zeta})\Big) = \frac{1}{2}(e^{\zeta} - e^{-\zeta}) = sinh(\zeta) \tag{69}$$

Table 6 summarizes the values of the parameters used in the code (with the exception of η_3, which is varied in the code:

Parameter	Value	Unit
k	50	$\frac{W}{m.C}$
A_{cs}	0.0001	m^2
u_{Γ_1}	50	C
L	0.05	m
u_∞	24	C
P_{cs}	0.04	m

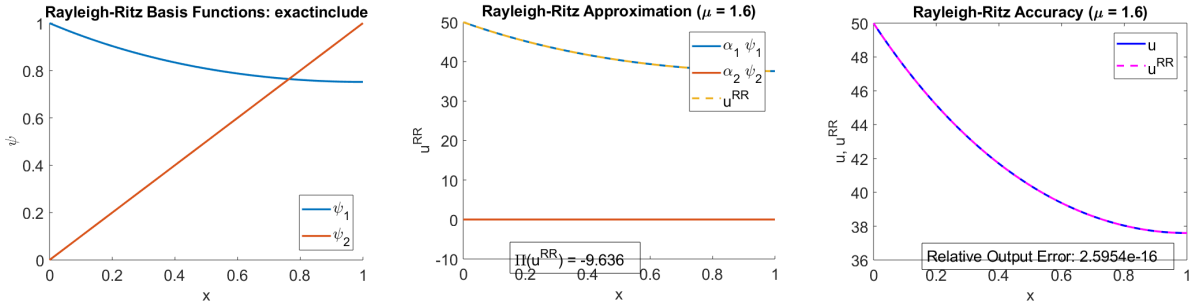
Table 6: Values of model 2 constants

First, we use two basis functions provided; the first one is the exact solution in 87 divided by boundary condition at Γ_1, $\Psi_0 = \frac{u(x)}{u_{\Gamma_1}}$, and the other is the linear function $\Psi_1 = x$. When we run the code, we find that:

$$\begin{aligned} \tilde{\alpha}_0 &= 50 = u_{\Gamma_1} \\ \tilde{\alpha}_0 &= -4.3444 \times 10^{-15} \approx 0 \end{aligned} \tag{70}$$

In which the RR method assigns all the weight to Ψ_0, in agreement with equation 55. We define the output to be the heat flux at the left boundary condition Γ_1 (the root of the fin):

$$s = -k\frac{du}{dx}\Big|_{x=0} = \frac{1}{L}(-k\frac{du}{d\zeta}\Big|_{\zeta=0}) \tag{71}$$



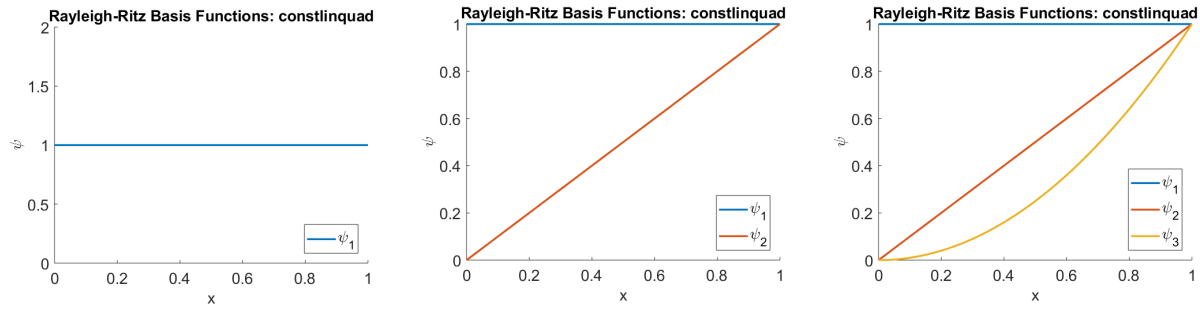
(a) Basis functions $\Psi_1 = \frac{u}{u_{\Gamma_1}}, \Psi_2 = x$ $\mu_0 = 1.6$

(b) u_{RR} approximation $\Psi_1 = \frac{u}{u_{\Gamma_1}}, \Psi_2 = x$ $\mu_0 = 1.6$

(c) u_{RR} accuracy $\Psi_1 = \frac{u}{u_{\Gamma_1}}, \Psi_2 = x$ $\mu_0 = 1.6$

Figure 8: u_{RR} base functions, approximations, and accuracy for model 2 exact $\mu_0 = 1.6$, $\Psi_1 = \frac{u}{u_{\Gamma_1}}, \Psi_2 = x$

Lets us try using more general base function:



(a) Basis Functions: $\Psi_0 = 1$

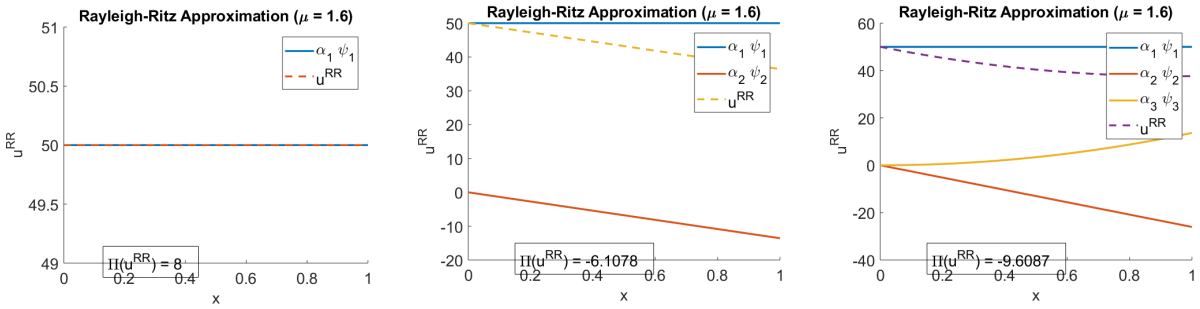
(b) Basis functions $\Psi_1 = 1$ and $\Psi_2 = x$

(c) Basis functions $\Psi_0 = 1$ and $\Psi_1 = x$ and $\Psi_2 = x^2$

Figure 9: Basis functions for model 2

$$
\begin{aligned}
n^{RR} = 0: &\quad \Psi_0 = 1 \\
n^{RR} = 1: &\quad \Psi_0 = 1, \Psi_1 = x \\
n^{RR} = 2: &\quad \Psi_0 = 1, \Psi_1 = x, \Psi_2 = x^2
\end{aligned}
\tag{72}
$$

The Basis functions for model one are shown in figure 9. Now we use the MATLAB code to come up with our own RR function approximation u^{RR} as shown in figure 10. Finally, we plot our function approximation u^{RR} along with the exact solution in figure 11 to evaluate the accuracy of our approximations.



(a) u_{RR} approximation: $\Psi_0 = 1$

(b) u_{RR} approximation $\Psi_0 = 1$ and $\Psi_1 = x$

(c) u_{RR} approximation $\Psi_0 = 1$ and $\Psi_1 = x$ and $\Psi_2 = x^2$

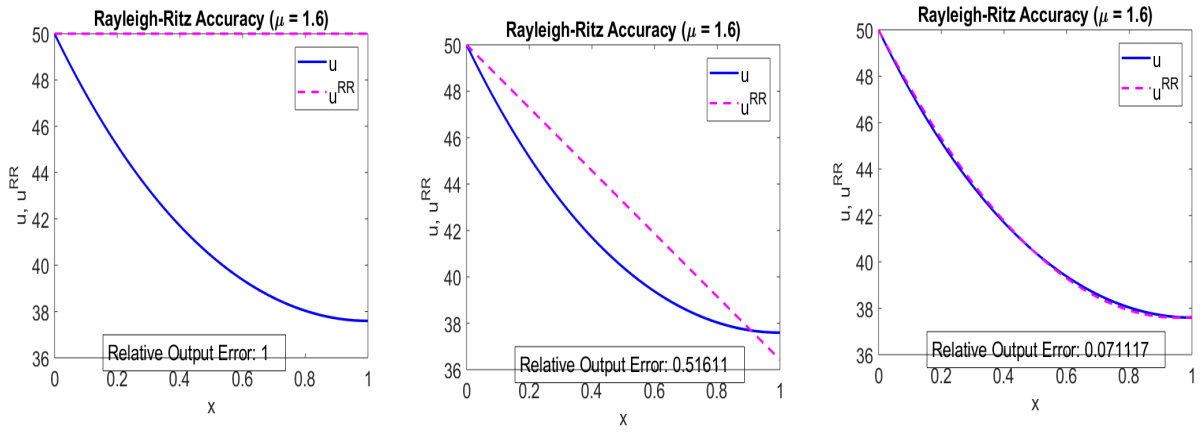
Figure 10: u_{RR} approximations for model 2, $\mu_0 = 1.6$

Table 7 summarizes the values of $\pi(u^{RR})$ and the relative output error when using the different basis functions, for $\mu_0 = 0.02$, 1 and 200 (μ was changed via changing η_3).

It would make sense for the relative output error to be 1 in case of using only Ψ_0 as our basis function, since the derivative of a constant function is zero and hence the relative output error, by definition, would be 1 for all values of μ_0. As in model 1, we could see in how $\pi(u^{RR})$ is lowest when incorporating the exact solution as a function giving zero relative output error. The energy functional decreases, as the number of basis functions used increases, yielding less relative output error and more accurate RR approximation, as also seen in figure 12

Also, by comparing the values of the relative output errors for the last two cases (case one being using Ψ_0 and Ψ_1 and case two being using Ψ_0, Ψ_1 and Ψ_2), we could see that the relative output error (output being defined as the derivative or flux at the boundary) increases as μ_0 increases as in figure 12. We could also visually see how less accurate the approximation solution is in terms of tracing the actual solution, as μ increases, as seen in figure 13.

This means that as μ increases, the error between the approximation and exact solution increases and the approximation becomes less accurate. This could perhaps be attributed to the functional form of the actual solution and its



(a) u_{RR} accuracy: $\Psi_0 = 1$

(b) u_{RR} accuracy $\Psi_0 = 1$ and $\Psi_1 = x$

(c) u_{RR} accuracy $\Psi_0 = 1$ and $\Psi_1 = x$ and $\Psi_2 = x^2$

.

Figure 11: u_{RR} Accuracies for model 2, $\mu_0 = 1.6$

u^{RR}	n^{RR}	μ_0	$\pi(u^{RR})$	Relative Output Error
$\Psi_0 = \frac{u}{u(0)}, \Psi_1 = x$	1	0.02	0.095529097	0
$\Psi_0 = 1$	0	0.02	0.1	1
$\Psi_0 = 1, \Psi_1 = x$	1	0.02	0.096642	0.5
$\Psi_0 = 1, \Psi_1 = x, \Psi_2 = x^2$	2	0.02	0.095529099	0.00099856
$\Psi_0 = \frac{u}{u(0)}, \Psi_1 = x$	1	1.6	-9.636	2.5954×10^{-16}
$\Psi_0 = 1$	0	1.6	8	1
$\Psi_0 = 1, \Psi_1 = x$	1	1.6	-6.1078	0.51611
$\Psi_0 = 1, \Psi_1 = x, \Psi_2 = x^2$	2	1.6	-9.6087	0.071117
$\Psi_0 = \frac{u}{u(0)}, \Psi_1 = x$	1	200	-5281.9958	0
$\Psi_0 = 1$	0	200	1000	1
$\Psi_0 = 1, \Psi_1 = x$	1	200	-3995.0739	0.8955
$\Psi_0 = 1, \Psi_1 = x, \Psi_2 = x^2$	2	200	-4879.3723	0.73529

Table 7: Values of the functional energy and relative output error for the RR functions for model 2

dependency on μ. To understand why, it is worth it to pay some attention to what μ means. μ is some fin parameter that depends on the fin properties (length, perimeter, area, conductivity), as seen in table 5. As μ increases, the function decays more steeply along the length of the fin before it stabilises at value of around u_∞. Hence, this parameter apparently determines how rapidly the fin reaches the surrounding temperature along the length of the fin, or, equivalently, much much heat is lost through the fin to the surrounding. This makes sense; for instance, as the perimeter/Area ratio (and thus μ) increases, more of the fin material is in contact to outside air and thus loses temperature and equilibriates with room temperature earlier along the fin length. Similarly as length (and thus μ) increases, you get more length, or fin material exposed to surrounding air for same fraction of fin length, meaning faster decay per fractional length of the fin. Finally, as the conductivity increases, more heat would flow via conduction through the fin cross section rather than to the outside surrounding via convection (heat prefers traveling path of less resistance), thereby meaning slower decay of temperature (equivalently, as PL increases, surface area exposed to surrounding increases and heat transfer via convection increases, and as $\frac{L}{KA}$ increases conductive resistance increases also resulting in greater heat transfer via convection and thus faster temperature drop). All in all, the greater the value of μ, the faster the decay of temperature along the fin length until temperature reaches room temperature, the steeper the curvature , and the longer the 'tail' of the function, as shown in figure 13. All these features make it harder to approximate the solution via RR method using the basic constant, linear and quadratic monomials that we have used as base functions, which could potentially explain why error increases as μ_0 increases.

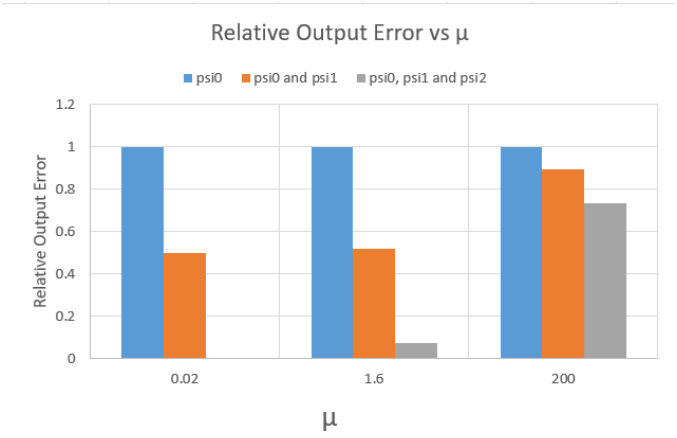


Figure 12: Relative Output Error Model 2

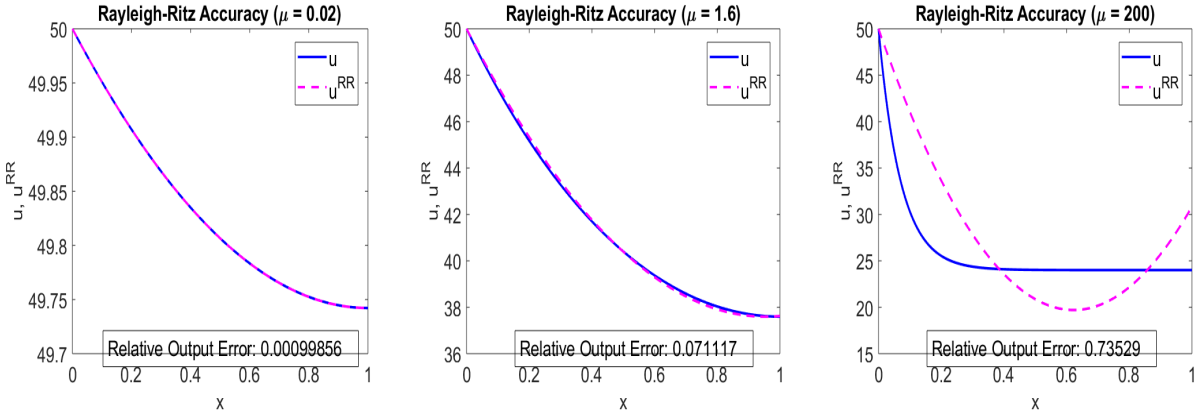


Figure 13: u_{RR} accuracies for model 2, across different μ_0, using : $\Psi_0 = 1$, $\Psi_1 = x$, and $\Psi_2 = x^2$

2 Chapter Two: The Finite Element Method for One Dimensional Second Order Symmetric Positive Definite Boundary Value Problems

2.1 Summaries of Models

2.1.1 Model One

Model One corresponds to quasi-1D heat conduction in a conical frustum insulated on the lateral surfaces with heat flux and heat transfer coefficient boundary conditions on the left and right surfaces, respectively. We impose the heat flux at the first boundary (derivative of required temperature solution), and heat transfer coefficient at the second boundary, to get our Neumann/Robin (N/R) boundary conditions. The BVP for the 1-D temperature profile across the conical frustum is given by:

$$-k\frac{d}{dx}(\pi R_0^2(1+\beta\frac{x}{L})^2\frac{du}{dx}) = 0 \qquad in\ \Omega \tag{73}$$

where $\Omega = (0, L)$. Equation 73 could be rearranged to the more familiar form encountered in class for (N/R) BVPs. Taking πR_0^2 out of the derivative, inserting k inside the derivative (since both are constants) and dividing both LHS and RHS by πR_0^2:

$$\frac{-d}{dx}\Big(\underbrace{k(1+\beta\frac{x}{L})^2}_{k(x)}\frac{du}{dx}\Big)+\underbrace{0}_{\mu(x)}u=\underbrace{0}_{f_\Omega} \tag{74}$$

The first boundary condition at the left surface, in which the flux is imposed, is given by:

$$k\frac{du}{dx}=-q_1 \qquad on \quad \tau_1 \tag{75}$$

Multiply both sides of equation 75 by $(1+\beta\frac{x}{L})^2$ to get:

$$\underbrace{k(1+\beta\frac{x}{L})^2}_{k(x)}\frac{du}{dx}=-(1+\beta\frac{x}{L})^2q_1 \qquad on \quad \tau_1 \tag{76}$$

However, given that surface τ_1 occurs at x=0, equation 76 can be rewritten as:

$$\underbrace{k(1+\beta\frac{x}{L})^2\Big|_{x=0}}_{k(x=0)}\frac{du}{dx}=\underbrace{0}_{\gamma_1}-\underbrace{q_1}_{f_{r_1}} \qquad (on \ \ \tau_1) \tag{77}$$

Similarly, the boundary condition at the second surface τ_2 to the right which is:

$$-k\frac{du}{dx}=\eta_2(u-u_\infty) \quad on \ \ \tau_2 \tag{78}$$

can be multiplied by $(1+\beta\frac{x}{L}^2)$ to get:

$$-\underbrace{k(1+\beta\frac{x}{L})^2\Big|_{x=L}}_{k(x=L)}\frac{du}{dx}=\underbrace{(1+\beta)^2\eta_2}_{\gamma_2}u-\underbrace{\eta(1+\beta)^2u_\infty}_{f_{r_2}} \qquad (on \ \ \tau_2) \tag{79}$$

The output was defined to be the temperature at the left boundary (i.e when λ_1 =1 and $\lambda_2 = 0$ in equation 80).

$$\begin{aligned} s &= \lambda_1 u\Big|_{\Gamma_1}+\lambda_2 u\Big|_{\Gamma_2} \\ &= u(0) \end{aligned} \tag{80}$$

The exact solution for model 1 is given by:

$$u=u_\infty+\frac{q_1L}{k}\Big(\frac{1+\beta+\frac{k}{\eta_2L}}{(1+\beta)^2}-\frac{\frac{x}{L}}{1+\beta\frac{x}{L}}\Big) \tag{81}$$

2.1.2 Model Two

Model two corresponds to a right-cylinder thermal fin with temperature and zero-flux boundary conditions on the left and right surfaces, respectively. Since we are provided with a temperature boundary condition at the first boundary, and the (effective) flux at the second boundary, we consider those as Dirichlet-N/R boundary conditions imposed on our ODE and we approach the approximation to the second order BVP via such case.

The BVP for the right-cylinder thermal fin is given by equation 82

$$-KA_{cs}\frac{d^2u}{dx^2}+\eta_3 P_{cs}(u-u_\infty)=0 \qquad in\ \Omega \tag{82}$$

Where, just like model one, $\Omega=(0,L)$, and just like in model one, the BVP equation could be rearranged to the more familiar form encountered in class for Dirichlet-(N/R) BVPs, as in equation 83:

$$-\frac{d}{dx}\underbrace{(kA_{cs}}_{k(x)}\frac{du}{dx})+\underbrace{\eta_3 P_{cs}}_{\mu(x)}u=\underbrace{\eta_3 P_{cs}u_\infty}_{f_\Omega(x)} \tag{83}$$

And just like in model 1, we could rewrite the boundary conditions as the following:

$$u\Big|_{x=0}=u_{\Gamma_1} \qquad on\ \Gamma_1 \tag{84}$$

$$\underbrace{-kA_{cs}}_{k(x)}\frac{du}{dx}\Big|_{x=L}=\underbrace{0}_{\gamma_2 u-f_{\gamma_2}} \qquad on\ \Gamma_2 \tag{85}$$

We impose the simple condition satisfying equation 85 that $\gamma_2=f_{\Gamma_2}=0$.

We define the output to be the heat flux at the left boundary condition Γ_1 (the root of the fin):

$$s=-k\frac{du}{dx}\Big|_{x=0} \tag{86}$$

The exact solution to the right cylinder fin is given by equation:

$$\begin{aligned} u(x)&=u_\infty+(u_{\Gamma_1}-u_\infty)\frac{cosh(\sqrt{\mu_0(1-\frac{x}{L})}}{cosh\sqrt{\mu_0}}\\ \mu_0&=\frac{\eta_3 PcsL^2}{kAcs}\end{aligned} \tag{87}$$

2.1.3 Model Mine

For the third model, we examine the steady state heat transfer across a wall (of uniform cross sectional area) of thickness L, with effective heat flux (heat transfer via convection) imposed on both ends of the wall, thereby imposing Neumann-Robin boundary conditions at both the left end and right end of the domain. Furthermore, we assume that the wall is perfectly insulated such that there is no heat loss to the surrounding across the lateral surfaces of the wall, meaning that the only method of heat transfer through the wall is via conduction. We will assume constant material conductivity, and a heat source/generation term per unit cross sectional area f_Ω equal to 1. Putting all this together we obtain the following BVP in equation 88:

$$-kA_{cs}\frac{d^2u}{dx^2}=A_{cs}\underbrace{f_\Omega}_{1} \quad in\ \Omega \tag{88}$$

Equation 88 can be written as:

$$-\frac{d}{dx}(k\frac{du}{dx})+\underbrace{0}_{\mu(x)}=\underbrace{1}_{f_\Omega} \qquad in\ \Omega \tag{89}$$

With the following boundary conditions:

$$k\frac{du}{dx} = \eta_1(u - u_{\infty_1}) \quad (on\ \Gamma_1) \tag{90}$$

$$-k\frac{du}{dx} = \eta_2(u - u_{\infty_2}) \quad (on\ \Gamma_2) \tag{91}$$

Which can be written for convenience as:

$$k\frac{du}{dx} = \underbrace{\eta_1}_{\gamma_1} u - \underbrace{\eta_1 u_{\infty_1}}_{f_{\Gamma_1}} \quad (on\ \Gamma_1) \tag{92}$$

$$-k\frac{du}{dx} = \underbrace{\eta_2}_{\gamma_2} u - \underbrace{\eta_2 u_{\infty_2}}_{f_{\Gamma_2}} \quad (on\ \Gamma_2) \tag{93}$$

The BVP ODE can be simply analytically solved to arrive to the exact solution, which is a quadratic variation in terms of x in equation 94:

$$\begin{aligned} u(x) &= -\frac{1}{2k}x^2 + Ax + B \\ B &= \frac{\frac{f_{\Gamma_1}}{k} + \frac{L+\Gamma_2 L^2 + f_{\Gamma_2}}{\gamma_2 L + k}}{\frac{\gamma_1}{k} + \frac{\gamma_2}{\gamma_2 L + K}} \\ A &= \frac{\gamma_1 B - f_{\Gamma_1}}{k} \end{aligned} \tag{94}$$

With the following derivative:

$$u'(x) = -\frac{x}{k} + A \tag{95}$$

The output is defined to be the temperature at the left boundary (via setting $\lambda_1 = 1$ and $\lambda_2 = 0$:

$$\begin{aligned} s &= \lambda_1 u\Big|_{\Gamma_1} + \lambda_2 u\Big|_{\Gamma_2} \\ &= u(0) \end{aligned} \tag{96}$$

Table **??** shows the values of the parameters used for model mine the finite element method code:

Problem Parameter	Value	Units
k	0.5	$\frac{W}{mC}$
L	10	m
f_Ω	1	$\frac{W}{m^3 C}$
η_1	150	$\frac{W}{m^2 C}$
η_2	100	$\frac{W}{m^2 C}$
u_{∞_1}	20	C
u_{∞_2}	25	C

Table 8: Model mine parameter values

2.2 Finite Element Method

We could think of the finite element method as a specific case of the RR method for approximation of a function over a domain. In this case, our basis functions are actually piecewise functions that are only non-zero at specific regions in the domain, and are thus more capable of capturing local effects of the function as in chapter 1 model 2.

To solve for the unknown function over the domain, the method subdivides the large domain into smaller, simpler parts that are called finite element (in the one dimensional case, finite segments). The simple equations that model these finite element are then assembled into a larger system of equations that models the entire problem, just as the basis functions were weighted then assembled in the RR method.

We refer to the discritization of the domain as meshing (or 'triangulation' in 2D), and we divide the domain into element that span the whole domain:

$$\begin{aligned} Elements : T^m,\ 1 \leq m \leq n_{el} \\ \overline{\Omega} = \bigcup_{m=1}^{n_{el}} T^m \end{aligned} \tag{97}$$

Where each element's starting and ending point is defined by a node:

$$\begin{aligned} &Node : x^i,\ 1 \leq i \leq n_{node} \\ &x^1 = 0 < x^2 ... < x^{n_{node}} = L \\ &n_{node} = n_{el} + 1 \end{aligned} \tag{98}$$

We define the element length, h^m (which could be uniform across elements or not):

$$h^m = x^{m+1} - x^m,\ 1 \leq m \leq n_{el} \tag{99}$$

And then we use our special piecewise functions as our basis functions for RR approximation:

$$\psi_i(x) = \varphi_i(x),\ 1 \leq i \leq n_{node} \tag{100}$$

The nature of each basis function is that each function is continuous, and is equal to 1 on only one corresponding node, and evaluates to 0 on all other nodes:

$$\varphi_i(x) = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases} \tag{101}$$

This means that $\psi_i(x)$ would be non zero only over at most two elements (its corresponding element and preceeding element), whether we choose to define $\varphi_i(x)$ as linear (p=1) or quadratic (p=2).

Following our RR representation, we could write the approximate solution to our problem over our entire domain as a 'weighted' sum of our new basis (piecewise) functions:

$$u_h(x) = \Sigma_{i=1}^{n} u_{hi} \varphi_i(x) \tag{102}$$

However, given that each basis function is non zero only at its corresponding node, we get the very interesting and useful result that the evaluation of the approximation function at a particular node is the the value of the 'weight' assigned to the basis function corresponding to that node:

$$\begin{aligned} u_h(x^j) &= \Sigma_{i=1}^{n} u_{hi}\varphi(x^j) \\ &= u_{hj}\varphi_j(x^j) \\ &= u_{hj} \quad 1 \le j \le n \end{aligned} \tag{103}$$

Since the FE method could be viewed as a form of RR approximation, the the minimization and comparison propositions for RR method (discussed in chapter 1) also hold true in our FE case. One interesting proposition of which is that RR approximations are always better than (or equal to) any linear combination of basis functions:

$$\pi(\Sigma_{i=1}^{n^{RR}} \alpha_i^{RR}\Psi_i) < \pi(\Sigma_{i=1}^{n^{RR}} \alpha_i\Psi_i) \tag{104}$$

When equation 104 is applied in the FE case, we get the following inequality in equation 105:

$$\pi \underbrace{(\Sigma_{i=1}^{n} u_{hi}\varphi_i(x))}_{u_h(x)} \le \pi \underbrace{(\Sigma_{i=1}^{n} w_{hi}\varphi_i(x))}_{any\ other\ function\ w_h} \tag{105}$$

Equation 105 means that, for instance, our FE approximation using linear piecewise functions is better (according to the energy functional minimization norm) than any other piecewise linear function w_h continuous on our mesh, including the interpolant function where the values at the nodes are actually known!

Proceeding as we would proceed with the RR method, the FE method would eventually require solving a system of algebraic equations:

$$\begin{aligned} \frac{\partial Q_\pi}{\partial \alpha_k}(\alpha^{RR}) &= 0 \\ \underline{A}\alpha_{RR} &= \underline{F} \end{aligned} \tag{106}$$

Where just in the RR formulation in chapter 1, matrix A is an SPD matrix (proof in course notes) of size $n^{RR} \times n^{RR}$, while vector F is of size n^{RR}. And just as in the RR method, for our definition of the energy functional in chapter one, the entries in matrix A and vector F are given by:

$$A_{ij} = \int_0^L k(x)\frac{d\Psi_i}{dx}\frac{d\varphi_j}{dx} + \mu(x)\varphi_i(x)\varphi_j(x)\,dx + \gamma_1\varphi_i(0)\varphi_j(0) + \gamma_2\varphi_i(L)\varphi_j(L) \tag{107}$$

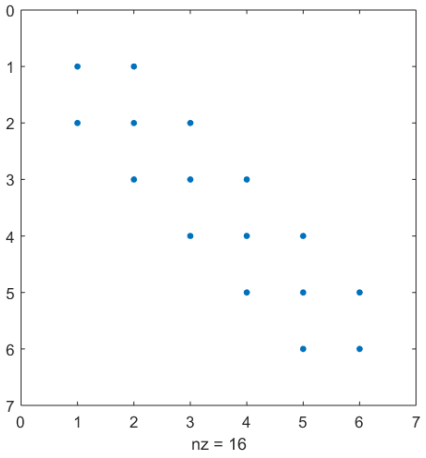
$$F_i = \int_0^L f_\Omega(x)\varphi_i(x)\,dx + f_{\Gamma_1}\varphi_i(0) + f_{\Gamma_2}\varphi_i(L) \tag{108}$$

$$1 \le i \le n_{node}$$

Notice that the entry A_{ij} in only non-zero when $\varphi_i(x)$ overlaps with $\varphi_j(x)$ (i.e both are non-zero). For linear base functions (p=1), a function could overlap with a maximum of 3 functions (including itself), while for quadratic base functions (p=2), a function could overlap with at most five functions (including itself). Accordingly, we could easily envision matrix A to be a sparse matrix that is tridiagonal in case of using linear base functions (p=1), and having at most 5 terms in a row (after reordering (thus pentadiagonal after reordering) when using quadratic base functions (p=2). Note that when using quadratic base functions (p=2), we introduce an additional node halfway between the nodes that define the boundary of the element, with its corresponding basis function that, just as the other node, takes a value of 1 only at its corresponding node and 0 at the other nodes in the domain (you need 3 points to define a quadratic function).

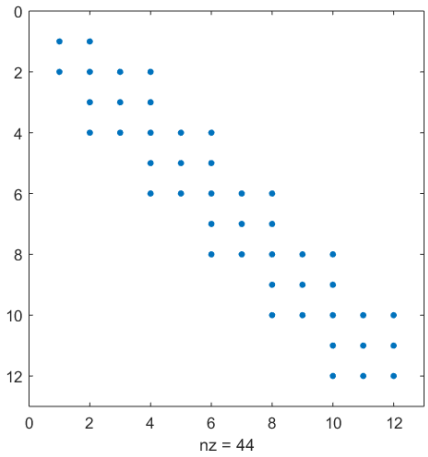
For instance, for p=1, we envision the tridiagonal matrix A to be something like this:

$$\begin{pmatrix} A_{11} & A_{12} & 0 & 0 & \cdots & 0 \\ A_{21} & A_{22} & A_{23} & 0 & \cdots & 0 \\ 0 & A_{32} & A_{33} & A_{34} & \dots & \vdots \\ & & & \vdots & & \\ 0 & \cdots & 0 & 0 & A_{n(n-1)} & A_{nn} \end{pmatrix}$$



(a) triadiagonal matrix A for p=1

.



(b) pentadiagonal matrix (after rearrangement) for p=2

Figure 14: Sparse Matrix structures for matrix A

For implementation, we create a generic 'reference element' that is representative of all potential elements in the domain. We define the domain of this reference element to be:

$$\hat{T} = [0, 1] \tag{109}$$

such that the left and right boundary nodes of our domain are defined by:

$$\begin{aligned} \hat{x}^1 &= 0 \\ \hat{x}^2 &= 1 \end{aligned} \tag{110}$$

And then we add our overlapping generic base functions onto our reference element: two linear functions for p=1 and 3 quadratic functions for p=2 (third function corresponding to a third node halfway through the domain of the reference element at $\hat{x} = 0.5$), such that, as before, each shape function takes the value of 1 at only its corresponding node, and a value of zero at the other node(s) in the reference element.

Given these sets of conditions, the generic shape functions ($\hat{S}_1(\hat{x})$ and $\hat{S}_2(\hat{x})$ for p=1, and $\hat{S}_1(\hat{x})$, $\hat{S}_2(\hat{x})$, and $\hat{S}_3(\hat{x})$ for p=2) can be constructed over the reference element (as in course notes). As mentioned, this reference element could represent any actual (global) element in the domain after some mapping. We could think of the mapping of the reference element to an actual element in the domain as a two stage process which includes 'translation' of the reference element such that the left node (boundary) of the reference element ($\hat{x}^0$) corresponds to the left node (boundary) of our global element m ($x^{lg(1,m)}$), and a scaling (or stretching component) such that each point on the reference element of length 1 maps onto a point on our global element m of length h^m. Hence the mapping from a reference element to a global element would take the form of

$$G_m(\hat{x}) = x^{lg(1,m)} + h^m \hat{x} \tag{111}$$

And the inverse mapping (from a global element to the reference element) would take the form of:

$$G^{-1}(x) = \frac{x - x^{lg(1,m)}}{h^m} \tag{112}$$

And we can see that applying the inverse mapping to the forward mapping would get us back to the value of $\hat{x}$ in the reference element:

$$G^{-1}(G_m(\hat{x})) = \frac{(x^{lg(1,m)} + h^m\hat{x}) - x^{lg(1,m)}}{h^m} = \hat{x} \tag{113}$$

Now to construct the matrix A and vector F in equations 107 and 108, we divide each into an 'integral' component that does not include the information about the boundary conditions, as well as a part for boundary condition information. We call the arrays that do not incorporate information about the boundary conditions elemental matrices and vectors, respectively.

$$A_{ij}^N = \int_0^L k(x)\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx} + \mu(x)\varphi_i(x)\varphi_j(x)\,dx \tag{114}$$

$$F_i^N = \int_0^L f_\Omega(x)\varphi_i(x)\,dx \tag{115}$$

$$1 \le i, j \le n_{node}$$

Furthermore, we divide the entries elemental matrix A^N into two components:

$$k_{ij}^N = \int_0^L k(x)\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx}\,dx \tag{116}$$

$$H_{ij}^N = \int_0^L \mu(x)\varphi_i(x)\varphi_j(x)\,dx \tag{117}$$

$$A^N ij = H^N ij + k^N ij \tag{118}$$

$$\tag{119}$$

We then decompose the integral over the length of the domain into constituent integrals over each element. for instance, for k_{ij}^N:

$$\begin{aligned} k_{ij}^N &= \int_0^L k(x)\frac{d\varphi_i}{dx}\frac{\varphi_j}{dx}dx \\ &= \Sigma_{m=1}^{n_{el}} \int_{x^{lg(1,m)}}^{x^{lg(2,m)}} k(x)\frac{d\varphi_i}{dx}\frac{\varphi_j}{dx}dx \end{aligned} \tag{120}$$

Basis functions that are zero over the element do not contribute to the integral over that element. Only the basis functions that have non-zero components over the element (at most 2 for p=1 and atmost 3 for p=2) contribute to the integral. Hence the integral over the element term for an element m in equation 120 is only non-zero for nodal entries:

1. row i = lg(1,m) , i = lg(2,m) , $\underbrace{i = lg(3,m)}_{(for\ p{=}2\ only)}$

2. column j = lg(1,m) , j = lg(2,m) , $\underbrace{j = lg(3,m)}_{(for\ p{=}2\ only)}$

Therefore, the only terms that could contribute to the integral over an element m in equation 120, and thus to the particular nodal entries in the elemental matrix, are the multiplications of $\varphi_{lg(\alpha,m)}$ and $\varphi_{lg(\beta,m)}$, where α and β represent the different base functions, or nodes that are present over the element. Notice that this also means that the integral over the element in equation 120 actually affects multiple nodal entry values in our elemental matrix:

$$k^N_{lg(\alpha,m)\ lg(\beta,m)} \tag{121}$$

contributes to

$$\underbrace{\int_{x^{lg(1,m)}}^{x^{lg(2,m)}} k(x) \frac{d\varphi_{lg(\alpha,m)}}{dx} \frac{d\varphi_{lg(\beta,m)}}{dx}\, dx}_{k^{elm}_{\alpha\beta}} \tag{122}$$

for each of α, β spanning from 1 to p+1, $1 \leq m \leq n_{el}$.

Notice that the expression in equation 122, $k^{elm}_{\alpha\beta}$ refers to the contribution of integrating over the particular element m, to the nodal entry values determined by α and β. The same thing could be done for the nodal entry values of the elemental H^N matrix, and after evaluating the effect of the integral over a particular element in equation 120 over all possible nodal combinations of α and β for that particular element, K^{elm} and H^{elm} we simply add the two together to get A^{elm}, as seen in equation 118. Similarly, for the elemental vector F, the contribution of the integral over an element, F^{elm} , could be evaluated over all nodes affected (dictated by only α in this case):

$$\begin{aligned} F^{elm}_{\alpha} &= \int_{x^{lg(1,m}}^{x^{lg(2,m}} f_\Omega(x) \varphi_{lg(\alpha,m)}\, dx \\ &1 \leq \alpha \leq p+1 \\ &1 \leq m \leq n_{el} \end{aligned} \tag{123}$$

To be able to modularly and thereby efficiently evaluate these integral expressions for each element, we map those expressions onto our reference element, using the mapping scheme in equations 111 and 112. A few things to bear in mind regarding scaling as we map the integral expression over to the reference element:

$$dx = h^m d\hat{x} \tag{124}$$

$$\frac{d}{dx} = \frac{d}{d\hat{x}} \frac{d\hat{x}}{dx} = \frac{1}{h^m} \frac{d}{d\hat{x}} \tag{125}$$

$$\tag{126}$$

Our basis functions over the particular global element become our generic shape functions; for instance:

$$\varphi_{lg(\beta,m)}(x) = \hat{S}_\beta(\hat{x}) \tag{127}$$

Applying these mapping to the entries in our elemental arrays, we get for the contribution of integration over an element to the nodal entry value in the elemental matrix:

$$k_{\alpha\beta}^{elm} = \frac{1}{h^m}\int_0^1 k(x^{lg(1,m)} + h^m x)\hat{S}'_\alpha(\hat{x})\hat{S}'_\beta(\hat{x})\,d\hat{x} \qquad 1 \le \alpha,\beta \le 2 \tag{128}$$

$$H_{\alpha\beta}^{elm} = h^m\int_0^1 \mu(x^{lg(1,m)} + h^m x)\hat{S}_\alpha(\hat{x})\hat{S}_\beta(\hat{x})\,d\hat{x} \qquad 1 \le \alpha,\beta \le 2 \tag{129}$$

$$A^{elm} = k^{elm} + H^{elm} \qquad 1 \le m \le n_{el} \tag{130}$$

$$F_\alpha^{elm} = h^m\int_0^1 f_\Omega(x^{lg(1,m)} + h^m\hat{x})\hat{S}_\alpha(\hat{x})\,d\hat{x} \qquad 1 \le \alpha \le 2, \quad 1 \le m \le n_{el} \tag{131}$$

For the evaluation of the integrals containing the piecewise functions on the reference element, we use a Gauss-Legendre numerical quadrature method, in which we discretize the domain of the integral into quadrature points, and approximate the integral of the function over the integral domain to be the weighted sum of the function values evaluated at the quadrature points:

$$\begin{aligned} &I = \int_0^1 g(\hat{x})\,d\hat{x} \approx I^{app} = \Sigma_{q=1}^{n^{quad}}\hat{w}_q^{quad}g(\hat{x}_q^{quad}) \\ &\hat{x}_q^{quad}, 1 \le q \le n^{quad} : \ quadrature\ points \\ &\hat{w}_q^{quad}, 1 \le q \le n^{quad} : \ quadrature\ weights \end{aligned} \tag{132}$$

The quadrature scheme would exactly evaluate the polynomial integral if the order of the polynomial inside the integral is less than or equal to $2 \times n^{quad} - 1$. This means that choosing two quadrature points over the reference element $n^{quad} = 2$ for p=1 would give the exact value of the integral of the quadratic function inside since the integral is evaluated exactly up to a third order polynomial function. Similarly, 3 quadrature points for p=2 would evaluate the integral of the fourth order polynomial function exactly since the integral is exactly accurate up to a fifth order polynomial function. The conversion of the integral to quadrature scheme provides those respective definitions for our elemental matrices, as provided by the lecture notes:

$$k_{\alpha\beta}^{elm} = \frac{1}{h^m}\Sigma_{q=1}^{n^{quad}}(\hat{w})_q^{quad}\Big(k(x^{lg(1,m)} + h^m x)\hat{S}'_\alpha(\hat{x})\hat{S}'_\beta(\hat{x})\,d\hat{x}\Big) \qquad 1 \le \alpha,\beta \le 2 \tag{133}$$

$$H_{\alpha\beta}^{elm} = h^m\Sigma_{q=1}^{n^{quad}}(\hat{w})_q^{quad}\Big(\mu(x^{lg(1,m)} + h^m x)\hat{S}_\alpha(\hat{x})\hat{S}_\beta(\hat{x})\,d\hat{x}\Big) \qquad 1 \le \alpha,\beta \le 2 \tag{134}$$

$$A^{elm} = k^{elm} + H^{elm} \qquad 1 \le m \le n_{el} \tag{135}$$

$$F_\alpha^{elm} = h^m\Sigma_{q=1}^{n^{quad}}(\hat{w})_q^{quad}\Big(f_\Omega(x^{lg(1,m)} + h^m\hat{x})\hat{S}_\alpha(\hat{x})\,d\hat{x}\Big) \qquad 1 \le \alpha \le 2, \quad 1 \le m \le n_{el} \tag{136}$$

$$\tag{137}$$

To demonstrate how we do that in the code, let us view how k^{elm} is constructed:

First we construct an $n^{quad} \times n^{quad}$ diagonal matrix (call matrix D) whose diagonal values correspond to $k(\hat{x}_q^{quad})\hat{w}_q^{quad}$ (element by element multiplication).

$$\begin{pmatrix} D_{11} & 0 & 0 & \cdots \\ 0 & D_{22} & 0 & \cdots \\ 0 & 0 & D_{33} & \cdots \\ 0 & 0 & \ddots & \\ 0 & 0 & \cdots & D_{nn} \end{pmatrix}$$

We then construct a $(p+1) \times n^{quad}$ matrix (call matrix S') corresponding to the shape function derivatives over the element evaluated at the quadrature points:

$$\begin{pmatrix} S_1'(\hat{x}_1^{quad}) & \cdots & S_1'(\hat{x}_n^{quad}) \\ \vdots & \cdots & \vdots \\ S_3'(\hat{x}_1^{quad}) & \cdots & S_3'(\hat{x}_n^{quad}) \end{pmatrix}$$

We then multiply the matrices in the following order: $S' \times D \times S'^T$ to get a $(p+1) \times (p+1)$ matrix corresponding to k^{elm}, which outlines, as discussed, the contribution of integrating over a particular element m for nodal entries α and β corresponding to that element:

$k^{elm} =$

$$\begin{pmatrix} k_{11} & \cdots & k_{1(p+1)} \\ \vdots & \cdots & \vdots \\ k_{(p+1)1} & \cdots & k_{(p+1)(p+1)} \end{pmatrix}$$

Such that the value of each entry in the matrix above is given by:

$$\begin{aligned} &k_{\alpha\beta}^{elm} = \Sigma_{q=1}^{nquad} \hat{S}_\alpha'(\hat{x}_q^{quad}) D_q^m \hat{S}_\beta'(\hat{x}_q^{quad}) \\ &1 \leq \alpha\beta \leq p+1 \end{aligned} \tag{138}$$

And of course repeating this for the contribution of every element would give us a 3 dimensional matrix. The same is repeated for the other arrays too.

```
A_el = zeros(nperel,nperel,n_el);
M_el = zeros(nperel,nperel,n_el);
Minertia_el = zeros(nperel,nperel,n_el);
F_el = zeros(nperel,n_el);

for i = 1:n_el
    xgq_el = xpts(lg(1,i)) + xq*h(i);
    kappa_el = kappa_fcn(xgq_el);
    mu_el = mu_fcn(xgq_el);
    rho_el = rho_fcn(xgq_el);
    f_Omega_el = f_Omega_fcn(xgq_el);


    % modification: three lines to be completed below
    A_el(:,:,i) = (shapeder_xq*diag(kappa_el.*wq)*shapeder_xq')/h(i) ...
        +  (shape_xq*diag(mu_el.*wq)*shape_xq')*h(i);
    M_el(:,:,i) =  h(i) * shape_xq * diag(wq) * shape_xq';
    Minertia_el(:,:,i) = h(i) * shape_xq * diag(rho_el.*wq) * shape_xq';
    X_el(:,:,i) = (shapeder_xq*(diag(wq))*shapeder_xq')/h(i) + c_L2*M_el(:,:,i);
    F_el(:,i) = (f_Omega_el'*diag(wq)*shape_xq')*h(i);
    % end modification
end
```

All the necessary values are stored in the arrays above, from which I could now construct my elemental matrices via direct stiffness summation:

$$
\begin{aligned}
&A^N = zeros(n,n)\\
&F^N = zeros(n,1)\\
&for\ m = 1 : n_{el}\\
&\ for\ \alpha = 1 : p+1\\
&\ \ for\ \beta = 1 : p+1\\
&\ \ \ A^N_{lg(\alpha,m)lg(\beta,m)} = A^N_{lg(\alpha,m)lg(\beta,m)} + A^{elm}_{\alpha\beta}\\
&\ \ end\\
&\ \ F^N_{lg(\alpha,m)} = F^N_{lg(\alpha,m)} + F^{elm}_{\alpha\beta}\\
&\ end\\
&end
\end{aligned}
$$

Finally, we add our 'boundary information' from equations 107 and 108 to our elemental matrices to construct our arrays, noting that:

$$\varphi_i(0)\varphi_j(0) = \begin{cases} 1 & i = j = 0 \\ 0 & otherwise \end{cases} \tag{139}$$

Similarly

$$\varphi_i(L)\varphi_j(L) = \begin{cases} 1 & i = j = n_{el0} + 1 \\ 0 & otherwise \end{cases} \tag{140}$$

Where $n_{el0} + 1$ refers to the node at x=L. Hence for our N/R boundary conditions on both boundaries, we have:

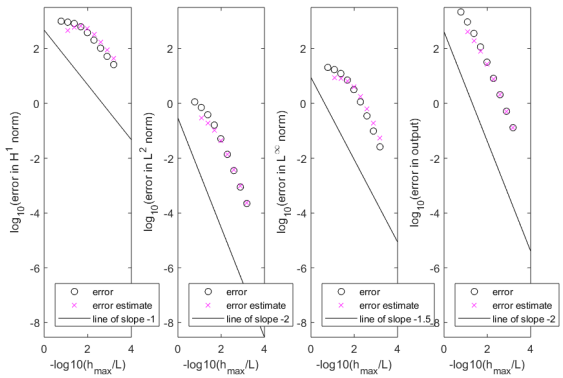
```
F(1) = F(1) + f_Gamma1;
    % NOTE: nodes are numbered (for p = 1 and p = 2, and under uniform and
    % adaptive refinement) such that node 1 is always the node at x
    % = xleft.
A(1,1) = A(1,1) + gam_Gamma1 ;

F(n_el0 + 1) = F(n_el0 + 1) + f_Gamma2;
    % NOTE: nodes are numbered (for p = 1 and p = 2, and under uniform and
    % adaptive refinement) such that node n_el0 + 1 is always the node at x
    % = xright.
A(n_el0 +1, n_el0 +1) = A(n_el0 +1, n_el0 +1) + gam_Gamma2;
```

2.3 Convergence

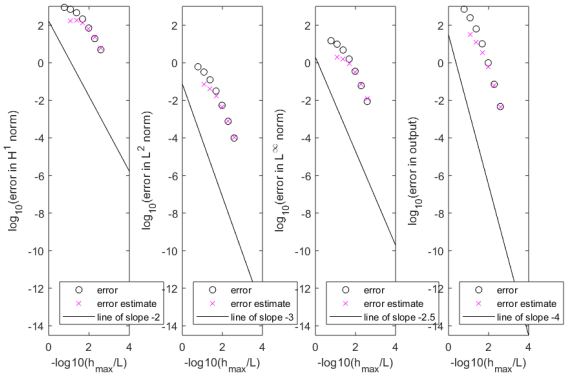
We are asked to provide evidence of convergence of the error as meshing (number of elements) increases for chapter one model 2. We could see via figure 15, that for all error norms, using both linear (p=1 in sub-figure a) and quadratic (p=2 in sub-figure b) base functions, not only does the error converge as the number of elements increases, but it also converges at the rate (right slope for the corresponding error norm) in the asymptotic convergence regime, meaning after the domain is meshed to a particular number of elements (i.e when the log(error) starts varying asymptotically with log(number of elements) and not for coarser meshes whose points do not lie on the asymptotic line). And indeed, for both p=1 (linear piecewise base functions) and p=2 (quadratic piecewise base functions), we could see that our FE approximation more accurately approximates the actual solution throughout the domain, as the degree of meshing increases, as in figure 16.

While this does not provide absolute evidence, it does provide some confidence of correct implementation. When looking at the error convergence rate results for model mine in figure 17, we see the same trends in approximation



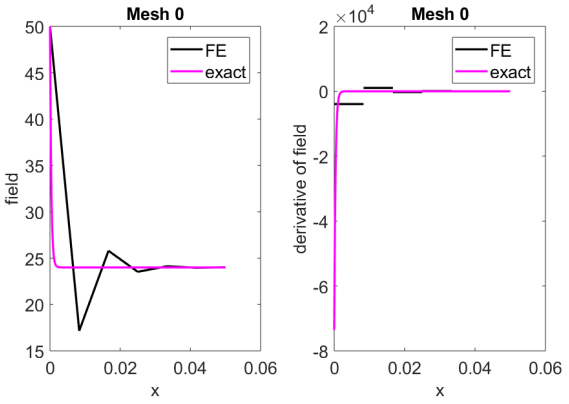
(a) log(error) vs log(element number) across the different error norms for p=1

.

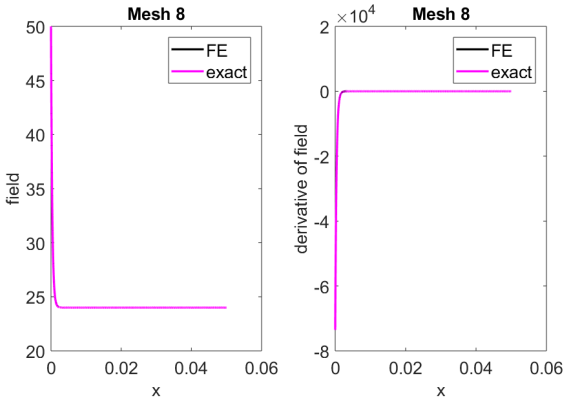


(b) log(error) vs log(element number) across the different error norms for p=2

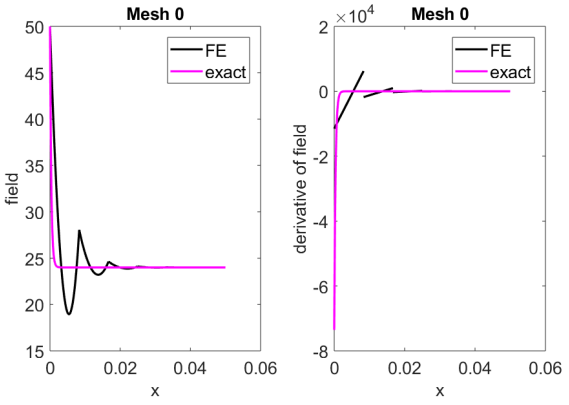
Figure 15: Convergence Rates across different norms for Chapter 1 model 2. In both cases of using either linear (p=1) and quadratic (p=2) base functions, the error estimates converge at the right rate (appropriate slope) across all norms after a particular degree of mesh refinement.



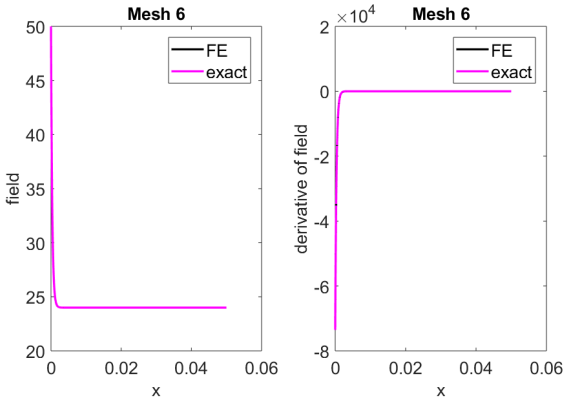
(a) Fe approximation before extra meshing model 2 p=1.



(b) Fe approximation after extra meshing model 2 p=1.



(c) Fe approximation before extra meshing model 2 p=2.

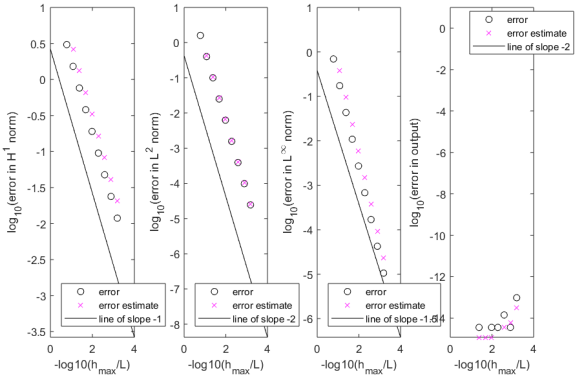


(d) Fe approximation before extra meshing model 2 p=2.

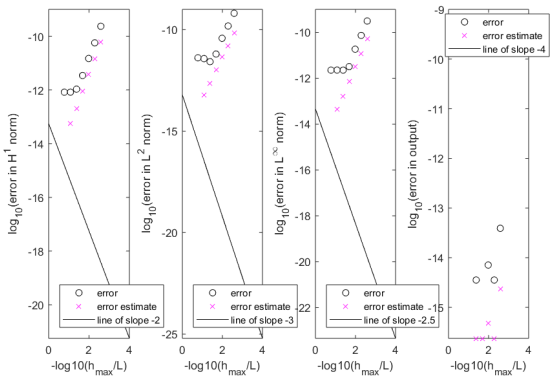
Figure 16: FE approximation for chapter 1 model 2, p=1,2. For both cases of using linear (p=1) and quadratic (p=2) base functions, the accuracy of the FE approximation, as infered from the degree of resemblance of the approximated solution to the exact solution, increases with increased mesh refinement.

error as for model 2 chapter 1, except for a couple of things. First, for p=2, the error across all norms is effectively zero (very small in magnitude due to finite machine precision). This is because the actual solution to our model is quadratic, and in our approximation we approximate the solution as a linear combination of quadratic functions over

each element, thus the FE method is able to come up with the exact solution to the BVP. And indeed, when looking at the FE approximation in figure 18, mesh 0 is accurately able to match the exact solution (and its derivative) across the length of the domain. In fact, we could see the small-magnitude errors actually increasing with meshing, and this is because as you decrease h and increase number of elements, the corresponding size of matrix A increases, which, upon inversion to solve for vector u_h, amplifies the finite machine precision errors. To examine potential convergence rates for p=2 it might be worth it to define a linear heat generation f_Ω term such that our actual solution would be cubic in terms of x instead of quadratic. Second of all, it is worth noting that our FE approximation provides exact agreement at the nodes due to the phenomenon of super-convergence, and indeed if we look at figure 17, the output error for p=1, which evaluates an output at particular node(s) (in our case the temperature at the left boundary), the error is effectively zero, meanwhile in the L_∞ norm, the error still converges (at the correct rate) since even with super-convergence our approximation is not exact for non-nodal values of x across the domain. Indeed, when examining the accuracy of the FE approximation in figure 18, the approximation exactly matches the exact solution values at the nodes. The observance of the superconvergence phenomenon in both the error converngence plot (figure 17) and the FE approximation plot (figure 18) for model mine, only provides more (by no means absolute) confidence that our implementation is correct.

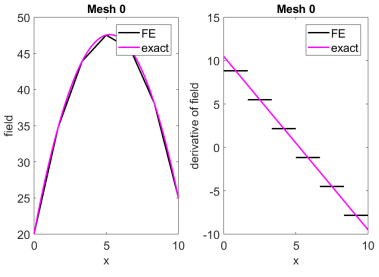


(a) log(error) vs log(element number) across the different error norms mode mine for p=1

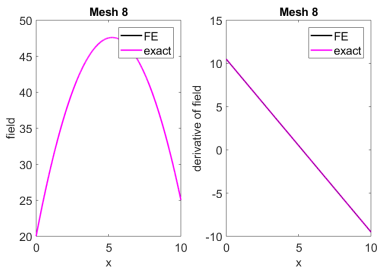


(b) log(error) vs log(element number) across the different error norms for model mine p=2

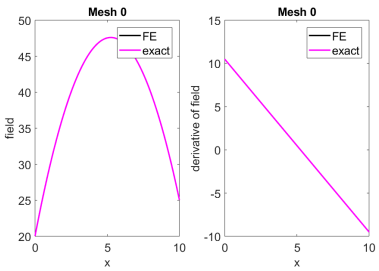
Figure 17: Convergence Rates for model mine, p=1,2 across error norms. Using quadratic piecewise base functions (p=2) in sub-figure b fully captures the exact solution which is quadratic, hence the approximation error for the case of p=2 is effectively zero as inferred from the low error values in sub-figure b that are due to finite machine precision (and hence increase with mesh refinement). Due to super convergence, the FE method exactly approximates the function at the nodes, which explains the effectively zero error in output for p=1. The agreement between the expected and observed error behavior in these case, as well as the correct convergence rates for the other error norms, provides confidence of correct numerical implementation.



(a) FE approximation before extra meshing for model mine for p=1



(b) FE approximation after extra meshing for model mine p=1



(c) FE approximation before extra meshing for model mine p=2

Figure 18: FE approximations for model mine for p=1,2. Using quadratic piecewise base functions (p=2) in sub-figure c fully captures the exact solution which is quadratic, hence the approximation solution consummately traces the exact solution. For both p=1 and p=2, the approximated solution exactly matches the exact solution at the nodes due to superconvergence.

Convergence provides some confidence, but not absolute evidence, of correct implementation. However, there could be errors in implementation that have not been detected for our particular problem model mine (or other models) due to the way parameters have been defined. For instance, for our model mine, we have set the heat transfer through the lateral through the lateral sides of the wall to be zero, and hence $\mu = 0$. Also for Chapter 1 model 2, μ is a constant. It could be the case that there is an implementation error that depends on μ, but that implementation error could have not been detected since we have set $\mu = 0$ as in model mine and model 1 chapter 1 or to a constant as in model 2 chapter 1. Therefore it would be useful to test an additional model, for instance a model similar to model 2 chapter 1 but now with a $\mu(x)$ term that varies linearly with x, to provide additional confidence that we have carried out correct implementation of the desired mathematical operations. Ideally we would test for an array of μ functions, such as polynomials, exponents... etc.

Let us examine a problem similar to chapter 1 model 2 where conductivity is still constant, but this time μ varies linearly with x All other parameters are same as in chapter 1 model 2.

$$-kA_{cs}\frac{d^2u}{dx^2} + \eta_3 P_0(1 + \frac{x}{L})(u - u_\infty) = 0 \;\; in \;\; \Omega \tag{141}$$

With the following Dirichlet-(N/R) boundary conditions:

$$u = u_{\Gamma_1} \quad (on\ \Gamma_1) \tag{142}$$

$$-k\frac{du}{dx} = \gamma_2 u - f_{\Gamma_2} \quad (on\ \Gamma_2) \tag{143}$$

In which γ_2 is known (the heat transfer coefficient at Γ_2). The BVP equation could be rearranged to the more familiar form encountered in class for Dirichlet-(N/R) BVPs, as in equation 144:

$$-\frac{d}{dx}\underbrace{(kA_{cs}}_{k(x)}\frac{du}{dx}) + \underbrace{\eta_3 P_0(1 - \frac{x}{L})}_{\mu(x)}u = \underbrace{\eta_3 u_\infty P_0(1 - \frac{x}{L})}_{f_\Omega(x)} \tag{144}$$

To apply the manufactured solution method, first consider as a smooth function, the decaying function u(x) = e^{-x} (preferably any function that would not be exactly evaluated using the FE method so that I could look at error convergence). On the LHS I would get:

$$-kA_{cs}e^{-x} + \eta_3 P_0(1 - \frac{x}{L})e^{-x} \tag{145}$$

The next step is to adjust the forcing term to arrive to a new BVP. In my case the forcing function is f_Ω(x), which now I equate to LHS I got in equation 145. I also use my smooth function to evaluate my unknown boundary parameters:

$$\begin{aligned} f_\Omega(x) &= -kA_{cs}e^{-x} + \eta_3 P_0(1 - \frac{x}{L})e^{-x} \\ u_{\Gamma_1} &= -ke^{-x}\Big|_{x=0} = -k \\ f_{\Gamma_2} &= (\gamma_2 - k)e^{-x} \end{aligned} \tag{146}$$

What I have now is a similar (albeit new and forced) BVP with an expression for all relevant parameters, one for which I know the exact solution (since I have fit the forcing term effectively to the exact solution), and also has a non-constant μ term. Now that I have the BVP, and the exact solution, I am now capable of evaluating the error norms and output error and see if the error convergences at the expected rate (slope) with successive spatial discretizations; if it does, this provides confidence of correct implementation. On the other hand, if the convergence is not at the right rate (slope), then our implementation is not correct.

In general, I could keep refining my mesh until:

$$\begin{aligned} \Delta_{h/2}^{H^1\Omega} &\leq \Delta_h^{H^1\Omega} \\ and\ \Delta_{h/2}^{norm}\ or\ \Delta_{h/2}^{s} &\leq \epsilon_{tolerance} \end{aligned} \tag{147}$$

For the first condition in equation 147, I use the error estimator in the H^1 norm since it is most sensitive measure of convergence, as it also reflects derivatives revealed by refinement. In general, I would only trust the error estimates for meshes when they are in the asymptotic region (decreasing) over the H^1 norm, after which I start looking at the other norms. For the second condition I set the error defined by any error norm or error output to be less than a particular error tolerance, which I could set to be the tolerance of the finite precision of the machine (for instance, 10^{-16}), which I could multiply by a safety factor like 2.

If, after refinement, both conditions are met, then, perhaps, I could get, using the exact solution:

$$\begin{aligned} ||u - u_{\frac{h}{2}}||^{norm} &\leq \epsilon_{tol} \\ or\ |s - s_{\frac{h}{2}}| &\leq \epsilon_{tol} \end{aligned} \tag{148}$$

Where the tolerance here could also be set to the machine finite precision tolerance. However, some stuff can prevent me from getting to this result even if my implementation is correct. For instance, we are assuming that

$$|s - s_h| \sim C_u h^{\tilde{r}} \tag{149}$$

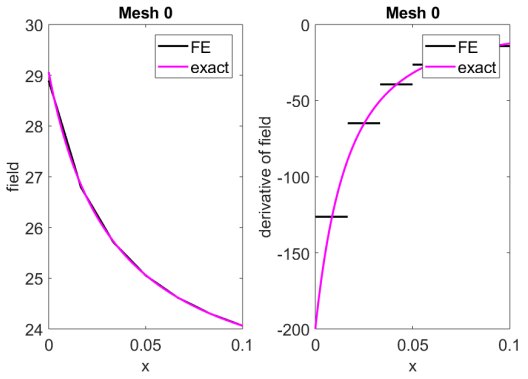
However, small higher order contributions could add up and put me on the wrong side of the error. This could be potentially avoided by assigning a safety factor to our tolerance, such as a factor of 2. Furthermore, there could be non-smooth effects present, which could be mitigated using a more 'conservative' exponent (for instance, $\tilde{r} = \frac{1}{2}$). What is important is to make sure that the error is going down as the mesh is refined (and at the right convergence rate for error estimates), which, combined with the fulfillment of the conditions mentioned, provides a strong case for the correct implementation since we used a non-constant μ term that would reveal any implementation errors that go unnoticed when using a constant μ term.

The fact that the error norms for both chapter 1 model 2 and chapter 2 model mine converge at the right rates, both of which have N/R boundary conditions that must be applied, provides kind of an indication for correct implementation and function modification of impose_boundary_cond_sver.

As for model 1 chapter 1, figure 7 demonstrates convergence of the error with mesh refinement at the expected rates (slope) across all error norms, and figure 19 demonstrates how the approximation appropriately approximates the solution at the boundaries once the boundary conditions are incorporated, and how such approximation improves as the meshing improves. The correct convergence rates, and the improved approximation with mesh refinement for model 1, provide confidence of correct numerical implementation. Chapter 2 model mine provides greater implementation confidence than chapter 1 model 1 because model mine has a non-zero γ_1 term that must be incorporated as a boundary condition, and so if there was an error in implementation of γ_1 that would have affected our approximation/convergence rates it would have more probably appeared during model line run, meanwhile for chapter 1 model 1 $\gamma_1 = 0$, and hence an error in implementation of the term could have gone unnoticed.

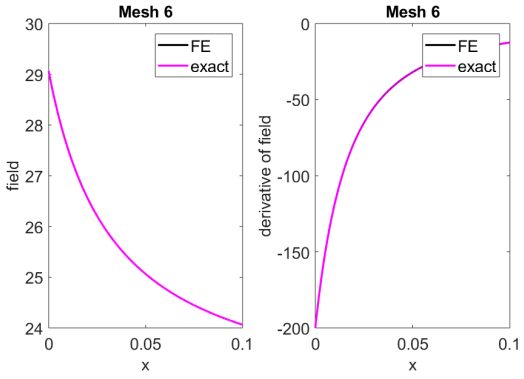
Notice that in general, we cannot assume that just because the error estimates for a particular model converge at the anticipated rates (even in all norms), our actual FE approximation is converging to the actual solution. Notice that no where in the error estimator definition do we use the actual (exact) solution for error estimation:

$$\begin{aligned} \Delta_h^{norm} &= \frac{||u_{\frac{h}{2}} - u_h||_{norm}}{1 - 2^{-\tilde{r}}} \\ \Delta_{\frac{h}{2}}^{norm} &= \frac{||u_{\frac{h}{2}} - u_h||_{norm}}{2^{\tilde{r}} - 1} \end{aligned} \tag{150}$$

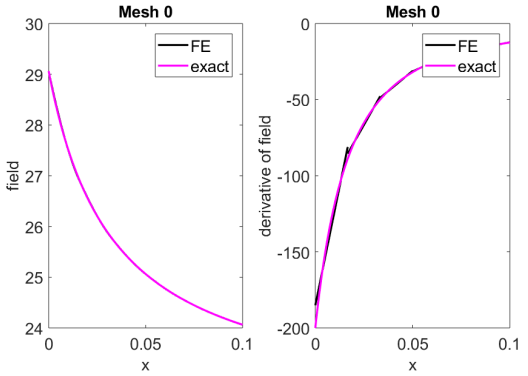


(a) FE approximation before extra meshing for model 1 for p=1

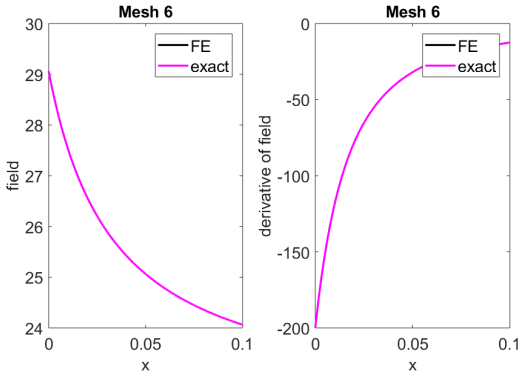
.



(b) FE approximation after extra meshing for model 1 p=1

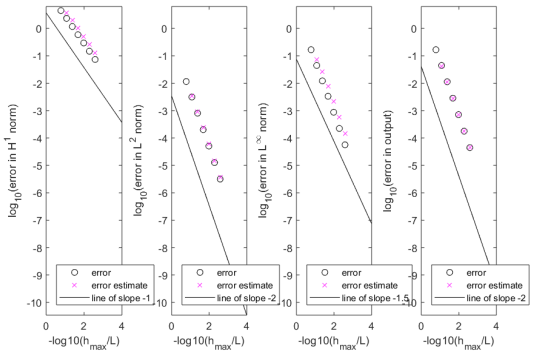


(c) FE approximation before extra meshing for model 1 p=2



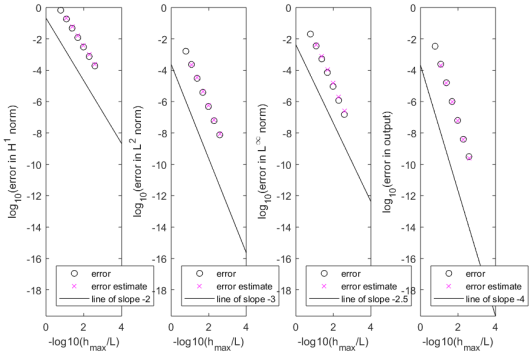
(d) FE approximation after extra meshing for model 1 p=2

Figure 19: FE approximations for model 1 for p=1,2. The approximated solution better traces the exact solution as meshing is refined for both linear (p=1) and quadratic (p=2) piecewise base functions.



(a) Convergence rates model 1 p=1

.



(b) Convergence rates model 1 p=2

Figure 20: Error convergence rates for model 1 using linear (p=1) and quadratic (p=2) piecewise functions. Across all error norms, the error converges at the expected rate (slope), providing confidence of correct numerical implementation.

where $\tilde{r}$ is some function of p that depends on which error norm we are using. Hence it could be very plausible that for instant, the user actually commits an error into inputting into the code or running his specific BVP. This means that the wrong problem_def structure is being run, for instance, by plugging in or running the code using a different differential equation, boundary conditions, or problem parameters. If the error norms do what they should be doing,

and otherwise the other implementations are correct, then indeed the error norms would be converging at the right rate but they are estimating the error for the wrong problem and hence the FE approximation would by no means necessarily converge to the actual exact solution, but rather converging to the wrong boundary value problem.

For task 4, estimating the error in our approximation across the different error norms for chapter 1 model 2 (for p=1) is done using the error estimator method without regards to the actual solution (which is often the case in real life). Problem 2 asks for the coursest mesh such that the error in the L_∞ norm is less than 1, meaning log(error in L_∞ norm) is less than zero. This occurs for mesh 7, for which the base 10 logarithm of estimated error in the norm is -0.1967, and hence the error is in the L_∞ norm is 0.63577. It is worth examining also the H^1 norm, since it is the most sensitive norm given that it incorporate the error in the derivative in its definition:

$$\begin{aligned} ||v||^2_{H^1(\Omega)} &= \int_0^L (\frac{dv}{dx})^2 + \underbrace{c_0}_{\frac{1}{L^2}} v^2 \, dx \\ v(x) &= u - u_h \end{aligned} \tag{151}$$

It seems that the mesh points start falling into the asymptotic regime (convergence of the error with increased mesh refinements with a slope appropriate to the error norm) for points including and after mesh 5 in the H^1 norm. For all the other error norms for chapter 1 model 2, p=1 in figure 15, mesh 5 falls within the asymptotic convergence region which means we could use the theoretical posteriori error estimates for meshs 5 and after as approximations to the actual error across all norms.

To better ensure that the error estimate calculated for the L_∞ for mesh 7 is conservative, we introduce a safety factor of 2, hence an upper bound for $||u - u_h||_{L_\infty}$ could be $2(0.63577) = 1.27154$. Similarly, for the output error, for Mesh 5 we get that the log(error in output) = 1.443 and hence output error estimate = 27.73. We introduce a safety factor of 2 such that an upper bound for the output error for mesh 5 is 55.47. When comparing our error estimates with the exact error in the L_∞ norm and output error, for both mesh 5 and 7, our error estimates with the safety factor of 2 (hence our upper bounds for the error) are higher than the actual error (in the asymptotic convergence region where the error estimates are valid), as seen in table

Mesh number	Upper bound for L_∞ error	Actual L_∞ error	Upper bound for output error	Actual output error
5	7.96	3.17	55.47	31.84
7	1.27	0.35	4.10	2.05

Table 9: Upper bound estimates and actual errors in the L_∞ norm and output error, for model 2 chapter 1 using linear piecewise base functions (p=1), as obtained from the error convergence rate plots in figure 15. The upper bounds for both the L_∞ norm and output error were obtained by multiplying the calculated posteriori error estimate with a safety factor of 2. The upper bounds obtained from the error estimates are always greater than the actual error, which ensures our error estimates (with a safety factor) are conservative.

3 Chapter Three: The Finite Difference-Finite Element Method for the One Dimensional Heat Equation

3.1 Task One

We now consider a spatiotemporal (varying in both space and time) model for heat transfer, as opposed to the steady state heat transfer models we examined earlier. The heat equation for the unsteady state is very similar to the one with steady state heat transfer, and could also be seen as a result of the basic application of the first law of thermodynamics to a body:

$$\frac{dU}{dt}_{body} = Q_{in} + Q_{Source} - \dot{W} \tag{152}$$

Where U is the internal energy, Q_{in} is the (net) heat transfer rate into our body of interest, Q_{source} is the heat generation inside the body, and $\dot{W}$ is the rate of the work that our body does on the surrounding. Applying equation 152 to a control volume of thickness Δx, and assuming there is neither heat generation inside our control volume nor work done by our control volume, equation 152 simplifies to:

$$\frac{dU}{dt}_{body} = Q_{in} \tag{153}$$

We now simply use Fourier's Law of heat transfer to find an expression for the heat transfer flux:

$$\overrightarrow{q} = -k\overrightarrow{\nabla}T \tag{154}$$

And applying Fourier's law for a 1D case of heat conduction, we get for the :

$$\overrightarrow{q} = -k\frac{\partial T}{\partial x}\hat{x} \tag{155}$$

And now integrating over the heat flux over a surface to get Q(x):

$$Q(x) = \int_S q_n \, dA = \int_S -k\frac{\partial T}{\partial x}\hat{x}\cdot\hat{x}\, dA = -kA\frac{\partial T}{\partial x} \tag{156}$$

For a homogeneous body with uniform cross sectional area. Referring to our control volume, heat transfer rate due to heat conduction through the body is heat flux into the control volume through surface at x minus heat flux out of the control volume through surface at $x + \Delta x$.

$$\begin{aligned} Q_{conduction} &= Q(x) - Q(x+\Delta x) \\ &= -kA\frac{\partial T}{\partial x}\Big|_x + kA\frac{\partial T}{\partial x}\Big|_{x+\Delta x} \end{aligned} \tag{157}$$

Using Taylor series to re-express the second term in equation 157:

$$kA\frac{\partial T}{\partial x}\Big|_{x+\Delta x} = kA\frac{\partial T}{\partial x}\Big|_x + \frac{\partial}{\partial x}\Big(kA\frac{\partial T}{\partial x}\Big)\Big|_x \Delta x + O(\Delta x^2) \tag{158}$$

We plug in equation 158 into equation 157 to get an expression for heat transfer rate due to conduction through the control volume:

$$
\begin{aligned}
Q_{conduction} &= Q(x) - Q(x+\Delta x) \\
&= \cancel{-kA\frac{\partial T}{\partial x}\Big|_x} + \cancel{kA\frac{\partial T}{\partial x}\Big|_x} + \frac{\partial}{\partial x}\Big(kA\frac{\partial T}{\partial x}\Big)\Big|_x \Delta x \\
&= \frac{\partial}{\partial x}\Big(kA\frac{\partial T}{\partial x}\Big)\Big|_x \Delta x
\end{aligned} \tag{159}
$$

Another factor affecting the heat transfer rate Q_{in} is the heat loss to the surrounding fluid via convection through the lateral surface. The lateral surface area of the control volume is given by $P_{cs}\Delta x$. Hence, according to Newton's law of cooling, the heat transfer from the control volume to the fluid is given by:

$$
Q_{convection} = \eta P_{cs}\Delta x(T_{cv} - T_{\text{inf}}) \tag{160}
$$

Where T_{inf} is the fluid temperature and T_{cv} is the control volume temperature, and η is the heat transfer coefficient at the control volume/fluid boundary. Hence the net heat transfer rate into the control volume is given by adding the contributions from both conduction and convection:

$$
\begin{aligned}
Q_{in} &= Q_{conduction} - Q_{convection} \\
&= \Delta x\Big(\frac{\partial}{\partial x}(kA\frac{\partial T}{\partial x})\Big|_x - \eta P_{cs}(T_{cv} - T_\infty)\Big)
\end{aligned} \tag{161}
$$

Next, we find an expression for the internal energy rate of change inside our control volume, assuming that internal energy is purely a function of temerature:

$$
\begin{aligned}
\frac{dU}{dt} &= mc\frac{T}{dt} \\
&= \rho Ac\frac{dT}{dt}\Delta x
\end{aligned} \tag{162}
$$

Finally we plug in the expressions we have in equations 161 and 162 into equation 153 , and cancel Δx in each term to find our heat equation for our whole body in terms of both time and space, instead of only for our control volume of a particular x value (assuming no heat generation):

$$
\rho cA\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\Big(kA\frac{\partial T}{\partial x}\Big) - \eta P_{cs}(T - T_\infty) \tag{163}
$$

Equation 163 belongs to the more general class of parabolic PDEs with the particular form discussed in class:

$$
-\frac{\partial}{\partial x}\Big(\underbrace{kA}_{k(x)}\frac{\partial u}{\partial x}\Big) + \underbrace{\eta P_{cs}}_{\mu(x)}T = \underbrace{\eta P_{cs}T_\infty}_{f_\Omega} - \underbrace{\rho cA}_{\rho(x)}\frac{\partial u}{\partial t} \tag{164}
$$

Where equation 164 characterizes the spatiotemporal variance of temperature in the spatial space Ω, over the time interval $t_i < t \leq t_f$. As we can see, equation 164 bares vey similar resemblance to the previous steady state equation we handled, except for the new time dependent term. We could group the two terms on the left hand side of equation 164 together to arrive to the following general form in equation 165.

$$
-\frac{\partial}{\partial x}\Big(k(x)\frac{\partial u}{\partial x}\Big) + \mu(x)u = \underbrace{f_\Omega - \rho(x)\frac{\partial u}{\partial t}}_{f_\Omega^t} \tag{165}
$$

The boundary conditions imposed would be, just like the steady state case, problem dependent, but here we would also need an initial condition characterizing the temperature distribution at the initial time. For N/R N/R boundary conditions imposed at both boundaries, the boundary and initial conditions would look something like this:

$$k(x)\frac{\partial u}{\partial x} = \gamma_1 u - f_{\Gamma_1} \qquad \text{on } \Gamma_1, \quad t_i < t \leq t_f \tag{166}$$

$$-k(x)\frac{\partial u}{\partial x} = \gamma_2 u - f_{\Gamma 2} \qquad \text{on } \Gamma_2, \quad t_i < t \leq t_f \tag{167}$$

$$u = u_{ic}(x) \qquad \text{in } \Omega, \quad t = t_i \tag{168}$$

Let us assume that each of the functions k(x), $\rho(x)$, and $\mu(x)$ are greater than or equal to zero, and that our boundary heat transfer coefficient terms corresponding to γ_1 and γ_2 are also greater than or equal to zero. Now, just as for the steady state case, we proceed with our finite element approximation:

$$u(x,t) \approx u_h(x,t) = \sum_{j=1}^{n} u_j^h(t)\varphi_j(x,t) \tag{169}$$

Conducting the FE approximation with the base functions φ_js as both time and space varying would be very complex. Instead, we decouple such that the only the values of the coefficients are time dependent, while the base functions are only space dependent. Recall from the previous chapter that the values of the coefficients corresponds to the values of the function at the nodes obtained from the meshing; hence these values at the nodes are changing (increasing or decreasing) with time evolution, while the inherent shape of the base function over elements is not changing with time (perhaps only scaled according the value at its corresponding node). Hence, the finite element approximation in equation 169 could be simplified:

$$u(x,t) \approx u_h(x,t) = \sum_{j=1}^{n} u_j^h(t)\varphi_j(x) \tag{170}$$

Therefore, when calculating the time derivative of our approximation, we only need to worry about the time varying behavior of our coefficients (value at nodes):

$$\dot{u}(x,t) \approx \dot{u}_h(x,t) = \sum_{j=1}^{n} \dot{u}_j^h(t)\varphi_j(x) \tag{171}$$

It is useful to consider the discretization process in two stages, first discretizing only in space, leaving the problem continuous in time, before we discretize in time using any standard numerical method for systems of ordinary differential equations. The first step leads to a system of ordinary differential equations in time, called the semi-discrete equations," which are similar to what we obtained for the FE steady state case:

$$Au^h = F^t \tag{172}$$

Where F^t contains the time variation factor in our approximation. The construction of the matrix A, as discussed in chapter 2, would first involve running the direct stiffness summation scheme to construct A^N, for which the boundary conditions are imposed via the addition of the γ terms to get $\tilde{A}$, which, for the N/R N/R boundary case, is the same as our desired matrix A in equation 172. The construction of the vector F^t is almost similar, it is just we need to factor in that our f_Ω^t has a time varying term.

We first construct F^{tN} via the direct stiffness summation scheme discussed in chapter 2, and then we impose the boundary conditions to get $\tilde{F}^t$, which, for the N/R N/R happens to be our desired F^t vector. The expression for each entry in the vector is similar to the one in chapter 2, except now we have f_Ω^t instead of f_Ω:

$$\begin{aligned}\tilde{F}_i^t &= \int_0^L f_\Omega^t \varphi_i \, dx + f_{\Gamma_1}\varphi_i(0) + f_{\Gamma_2}\varphi_i(L) \\ &= \int_0^L f_\Omega \varphi_i \, dx + f_{\Gamma_1}\varphi_i(0) + f_{\Gamma_2}\varphi_i(L) - \int_0^L \rho(x)()_h \varphi_i \, dx\end{aligned} \tag{173}$$

Before we proceed, let us introduce the inertia mass matrix, M. The inertia matrix is, just like matrix A, sparse (tridiagonal for p=1 and pentadiagonal for p=2). The entries of the matrix are described by:

$$\tilde{M}_{ij} = \int_0^L \rho(x)\varphi_i\varphi_j \, dx \quad 1 \leq i,j \leq n_{node} \tag{174}$$

Just like what we did for matrix A and vector F, the integral, or contribution of each element to the nodal entry values of Matrix M, dictated by α and β (which describe the base functions over the element), can be characterized, and is given by:

$$M_{\alpha\beta}^{elm} = h^m \sum_{q=1}^{n^{quad}} \hat{w}_q^{quad}[\rho(x^{lg(1,m)} + h^m \hat{x}_q^{quad})\hat{S}_\alpha(\hat{x}_q^{quad})\hat{S}_\beta(\hat{x}_q^{quad})] \quad 1 \leq \alpha,\beta,\leq p+1 \tag{175}$$

And just like what we did for matrix A and F, matrix M could be constructed via direct stiffness summation. In the N/R N/R boundary condition case, the matrix constructed from the direct summation is the same as our desired inertia matrix M.

Now going back to equation 173, we can the rewrite the entry value for matrix F to be:

$$\begin{aligned}\tilde{F}_i^t = \tilde{F}_i &- \int_0^L \rho(x)\varphi_i \sum_{j=1}^{n} \dot{u}_{hj}\varphi_j \, dx \\ &+ \tilde{F}_i - \underbrace{\sum_{j=1}^{n}\Big(\int_0^L \rho(x)\varphi_i\varphi_j \, dx\Big)\dot{u}_{hj}}_{M\dot{u}_h}\end{aligned} \tag{176}$$

The term under the bracket represents the multiplication of Matrix M with our coefficient time derivative vector (in which each entry value in the resultant vector is the dot product of the vector corresponding to the row from matrix M, with the column vector for the coefficients time derivative). This means that we can write F^t as:

$$F^t = F + M\dot{u}_h \tag{177}$$

Plugging in equation 177 into equation 172, we arrive to our system of n ODE equations in time:

$$\begin{aligned}M^{inertia}\dot{u}_h + Au_h = F \qquad & t_i < t \leq t_f \\ u_h = I_h U_{ic} \qquad & t = t_i\end{aligned} \tag{178}$$

Where, for time initial, we take the interpolant of our initial condition function due to our discretization of the spatial domain. Of course, if we had Dirichlet boundary conditions, the procedure would be similar, but would require the extraction of arrays A,F,M and b from $\tilde{A}$, $\tilde{F}$, and $\tilde{M}$ first.

Looking at our system of ODEs in equation 178, matrices M and A are of size (n × n), while vectors F, u_h and $\dot{u}_h$ are of size (n × 1). For the temporal discretization (grid in time), define

1. Δt to be our time discretization.
2. n_{tsteps} to be the number of points in time for which we are solving our system (the finite difference method approximates the solution to the ODE only at particular points in time, or timesteps, according to the time discretization).
3. t^k to be the time at the particular timestep k.
4. $u^k_{\Delta t}$ to be our approximation of the solution at a particular timestep k for the particular discretization of Δt.

Hence, for the grid in time, we get:

$$\Delta t = \frac{t_f - t_i}{n_{tsteps} - 1} \tag{179}$$

$$t^k = \Delta t(k-1), \qquad 1 \leq k \leq n_{tsteps} \tag{180}$$

$$z^k_{\Delta t} \approx z(t^k), \qquad 1 \leq k \leq n_{tsteps} \tag{181}$$

We use finite difference method to approximate the derivatives. The finite difference method converts each ODE to a system of algebraic equations (in our case a linear system), and hence can be applied to our system of ODEs obtained from spatial discretization to ultimately transform the PDE into equations solved by matrix/algebra. There are multiple discretization schemes that could be used, each with their advantages and disadvantages. In summary, explicit schemes calculate the state of the system at later time t^k from the state of the system at the current time t^{k-1}, while implicit schemes find a solution for the current state of the system by solving an equation involving the current state of the system and later one (thereby requiring some form of a matrix or a reiterative technique.

Explicit schemes are thus simpler, meaning they are less computationally expensive per time step, and easier to implement. However, they are only conditionally stable (for small enough Δt), and so to ensure stability small time steps are usually required. Implicit schemes are in general more robust in the sense that they are stable (since they use later state of the system for evaluation) over a wide range of timesteps, sometimes unconditionally, yet on the other hand they have a high cost per time step and could be harder to implement.

However, in most cases the approximate solution is desired to be within a prescribed accuracy. Most often, for some prescribed accuracy, the implicit (Euler Backward and Crank Nicolson) schemes will be less expensive than explicit (Euler Forward) schemes since for explicit schemes we are typically forced for stability reasons to take a time step much smaller than the time step actually required for the designated accuracy (given the conditional stability). Furthermore, for a case of one space dimension such as ours, the difference between a tridiagonal matrix multiplication required for an Euler Forward scheme, and the utilization of a tridiagonal solver tool, namely Matlab's fast backslash, for Euler Backward and Crank Nicolson schemes, is not very large (that is definitely not true for higher space dimensions, but is true for our model of one-dimensional heat transfer). Ultimately, implicit schemes, given their unconditional stability, can be much less expensive than explicit schemes in terms of achieving a prescribed level of accuracy since they do not require smaller time steps.

Table 10 includes the time stepping methods discussed in class, as well as their order (the asymptotic rate at which the error is reduced as Δt approaches zero), which depends on the truncation error from the Taylor expansion. We can model these different schemes via equation 182, where each scheme corresponds to a different value of θ, which can be thought of as the implicitness parameter.

θ	Scheme	Implicit/Explicit	Order
0	Euler Forward	Explicit	$O(\Delta t)$
$\frac{1}{2}$	Crank Nicholson	Implicit	$O(\Delta t)^2$
1	Euler Backward	Implicit	$O(\Delta t)$

Table 10: Finite Difference Schemes

$$
\begin{aligned}
&M\frac{u^k_{h,\Delta t}-u^{k-1}_{h,\Delta t}}{\Delta t}+A(\theta u^k_{h,\Delta t}+(1-\theta)u^{k-1}_{h,\Delta t})=F \qquad 2\le k\le n_{tsteps}\\
&u^k_{h,\delta t}=I_hU_{ic}, \qquad\qquad k=1
\end{aligned}
\tag{182}
$$

Where $u^k_{h,\Delta t}(x)$ corresponds to the approximated solution for a particular mesh h and time discretization Δt at timestep k. For the first point in time (k=1), we use the initial conditions provided (taking the interpolation of the initial condition over the spatial domain due to spatial discretization). We then solve our system (only) at the other timesteps, starting from the second timesteps until the last timestep corresponding to the final time t_f, using the timestepping method of our choice, dictated by the value of θ. Rearranging equation 182, we get the system of equations that we could implement in the code:

$$
\begin{aligned}
&(\frac{M}{\Delta t}+\theta A)u^k_{h,\Delta t}=\frac{M}{\Delta t}-(1-\theta)A)u^{k-1}_{h,\Delta t}+F \qquad 2\le k\le n_{tsteps}\\
&u^k_{h,\delta t}=I_hU_{ic}, \qquad\qquad k=1
\end{aligned}
\tag{183}
$$

3.2 Evaluating Correct Implementation: Tasks 2-4

To evaluate correct implementation for the Model semiinf plus, we first compare the approximated solution at the final time, to the actual solution at the final time, across different spatial meshings, timestepping schemes, and the behavior of the base functions over the elements (dictated by the value of p).

Examining figure 21, and looking at its left column, for each combination of p and θ in figures 21 a,c,e,and g, the initial mesh does indeed accurately trace the exact solution, which gives confidence of correct implementation. Furthermore, after 3 uniform refinements, the approximation at the final time more accurately traces the exact solution for all combinations of p and θ compared to the initial mesh (figures 1 b,d,f, and h respectively), and such improved accuracy with further refinement gives further confidence of correct implementation.

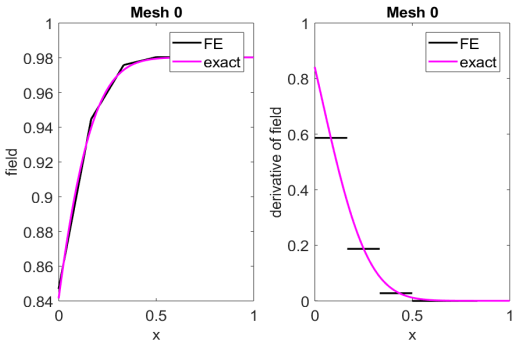
We not that each level of refinement includes both space and time uniform refinements such that with each level the size of each element h is halved, while the size of the time discretization Δt is scaled down by a particular value of α that depends on the finite difference scheme used (value of θ), the value of r, which in turn depends on the finite elements scheme used (value of p), and the error norm of interest Q. These values of α are chosen to balance the spatial and temporal errors convergence with uniform refinement (or at least for the L_2 norm; you cannot please every single norm). In other words, with the careful value of α, for the L_2 norm, we acheive the optimal balance for which the error due to temporal discretization decreases by the same factor as the error due to spatial discretization, as opposed to having one of those errors dominating over the other, in which case the discretization of the latter will not aid in refining the accuracy of the approximation by much. This is perhaps best demonstrated by looking at the equation for the convergence rate of the error. The error at a particular level of refinement at the final time (the final time norm) is approximated to be dependent on both the initial mesh/time discretization and number of refinements:

$$
\begin{aligned}
||u(.,t_f)-h^{n_{tsteps}}_{h,\Delta t}||^{(l)}_Q &\sim C^1_{u,Q}\alpha^{-ql}+C^2_{u,Q}2^{-rl}\\
&=2^{-rl}[C^1_{u,Q}(\frac{2^r}{\alpha^q})^l+C^2_{u,Q}]
\end{aligned}
\tag{184}
$$

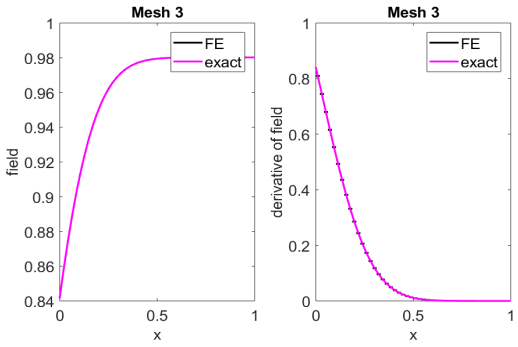
where

$$
q=\begin{cases}1 & \theta=1\\ 2 & \theta=0.5\end{cases}
\tag{185}
$$

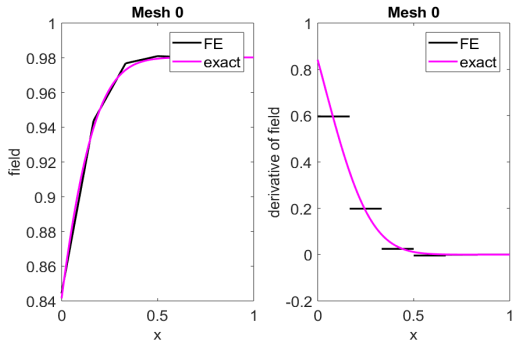
We could see that the error for a particular error norm depends on both the space and time discretizations, and we want to maintain a balance between both as we iteratively refine so that one does not dominate over the other, in which case, as mentioned, the discretization of the latter becomes almost futile. First, we chose our initial mesh and time discretization, h and Δt such that $C^1_{u,Q}\approx C^2_{u,Q}$. Then, looking at equation 184, we could see that if we could



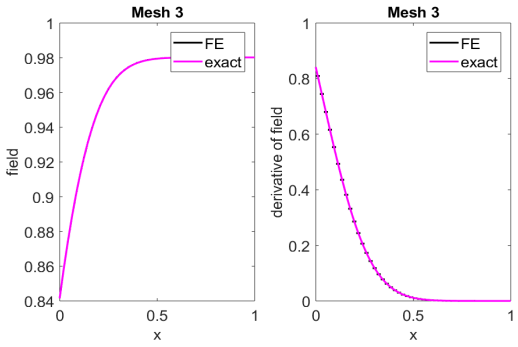
(a) First mesh, p=1, $\theta = 1$



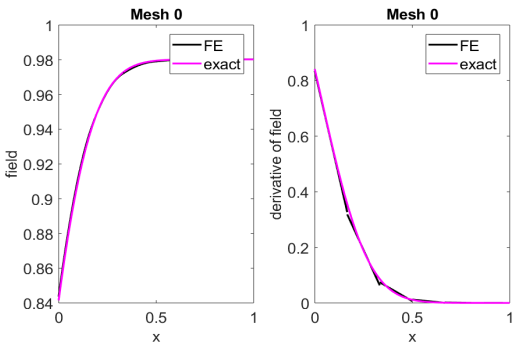
(b) Fourth mesh, p=1, $\theta = 1$



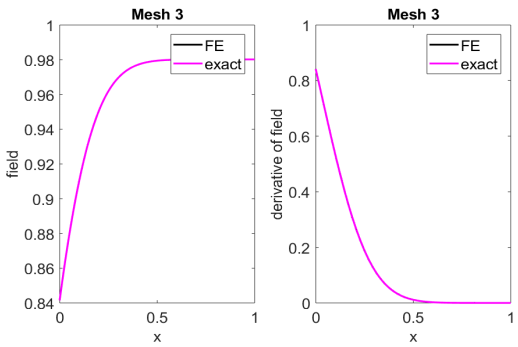
(c) First mesh, p=1, $\theta = 0.5$



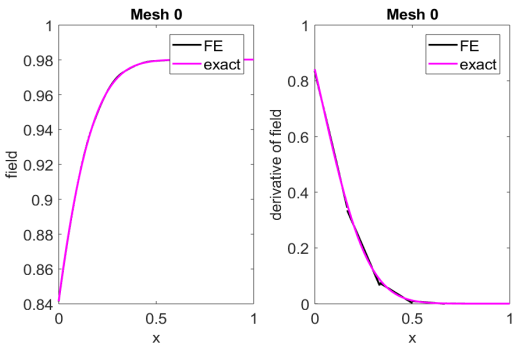
(d) Fourth mesh, p=1, $\theta = 0.5$



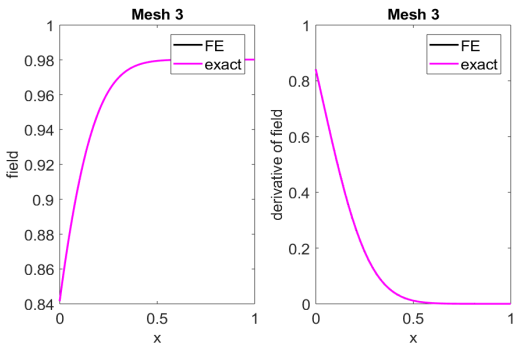
(e) First mesh, p=2, $\theta = 1$



(f) Fourth mesh, p=2, $\theta = 1$



(g) First mesh, p=2, $\theta = 0.5$



(h) Fourth mesh, p=2, $\theta = 0.5$

Figure 21: Approximate Solution compared to accurate solution at the final time for the semi-infinite model for an initial mesh of 6 elements and initial number of $n_{tsteps} = 10$.

somehow manipulate the weighing factor $(\frac{2^r}{\alpha^q})^l$, we could achieve a balance between the space and time convergence rates. Three factors go into this weighing factor, which are α, r (dependent on p and on the error norm Q), and q (dependent on θ). This means that for each combination of r and and θ, we could choose α such that this scaling factor is exactly 1 in the L_2 error norm, thus achieving the desired convergence balance in time and space. Indeed, the values of α for each case are given in the table in the lecture notes.

Thus, for the appropriate α values, we get:

$$\begin{aligned} ||u(.,t_f) - u_{h,\Delta t}^{n_{tsteps}}||_Q^{(l)} &\sim C_{u,Q} 2^{-rl} \\ &\sim C_{u,Q}(2^{-l})^r \end{aligned} \tag{186}$$

However, the size of the element at a particular refinement, could be found from the original element size (the one without refinements), and the number of refinements:

$$h^{(l)} = 2^{-l} h_0 \tag{187}$$

Equivalently, we could write equation 187 as:

$$2^{-l} = \frac{h^{(l)}}{h_0} \tag{188}$$

And so, we arrive to the convergence behavior of the error estimate in the L_2 in terms of the number of elements by plugging 187 into 186:

$$\begin{aligned} ||u(.,t_f) - u_{h,\Delta t}^{n_{tsteps}}||_Q^{(l)} &\sim C_{u,Q}(\frac{h^{(l)}}{h_0})^r \\ &\sim C_{u,Q}(\frac{h_0}{h^{(l)}})^{-r} \end{aligned} \tag{189}$$

Meaning that log(error) vs log(number of elements) should show the asymptotic convergence behavior with a slope of -r.

Figure 22 shows the convergence rates for the different error norms for the semi model across different values of p and θ. Looking at the error norms, we first observe that the actual error as well as the error estimate decreases with increased refinement, among all error norms:

$$\Delta_Q^{l+1} \leq \Delta_Q^l \tag{190}$$

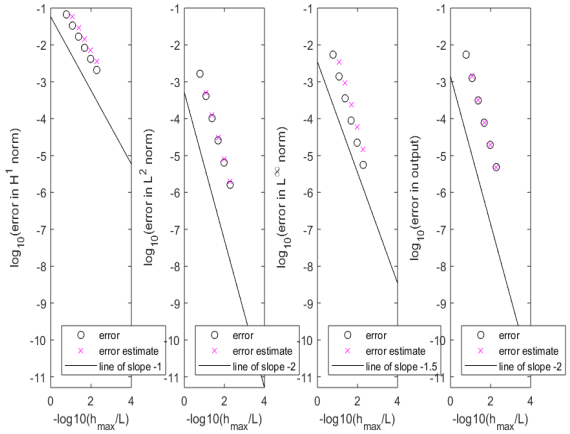
More importantly, and when specifically looking at the L_2 error norm, the error estimates converge at the correct rate (at the correct slope of -r), as required, which gives confidence of proper implementation. Furthermore, according to the reverse triangle inequality proof, the error estimates should be greater than or equal to (provide an upper bound) to the actual error:

$$||u(.,t_f) - u^{l+1}||_Q \leq \Delta_Q^{l+1} \tag{191}$$

Which is indeed the case when looking at the L_2 norm across the different combinations of θ and p in figure 22. All these features provide confidence of correct code implementation.

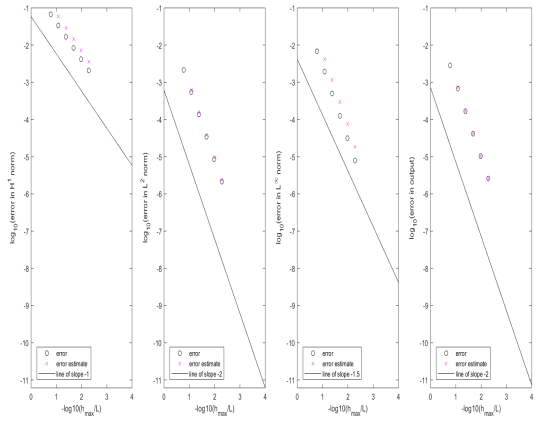
Note that while the tabulated values of α give me a value for $(\frac{2^r}{\alpha^q})^l$ =1 for the L_2 norm, this is not the case for the other norms due to different dependencies of r on p. This results in the uniform refinement being sub-optimal, either in time or space. For the H^1 and L_∞ norms, the error converges faster with space discretization than time

discretization (sub-optimal in time), where $(\frac{2^r}{\alpha^q})^l < 1$ (for both p=1 and 2). On the other hand, the output norm is sub-optimal in space (discretization error converges faster for time compared to spatial discretization) for p=2, in which $(\frac{2^r}{\alpha^q})^l > 1$. While the suboptimal in time case generally cannot be detected in the log(error) vs log(number of elements), the slope in the log(error) vs log(number of elements) for the sub-optimal in space case would be less steep than anticipated (in fact would have an absolute value of r-1 instead of r), as seen for the output error in figures 22 a,b,c and d. Also note that, when looking at the output error, for p=2, the actual error in the output error appears to be greater than the error estimate, but that is because there is a factor of ≈ 2 (that would emerge from the mathematical proof) to be multiplied to the error output estimate to ensure it is an upper bound, and it is not incorporated into the code. In fact if error estimates where multiplied by such factor then they would place an upper bound to the actual output error. Finally, looking at the p=1 case for subfigures a and b in figure 22, the L_∞ norm convergence is unrelated to the choice of σ, and converges at a faster rate (probably with $h^2 log(h)$ as opposed to our more conservative convergence rate of $h^{\frac{3}{2}}$).

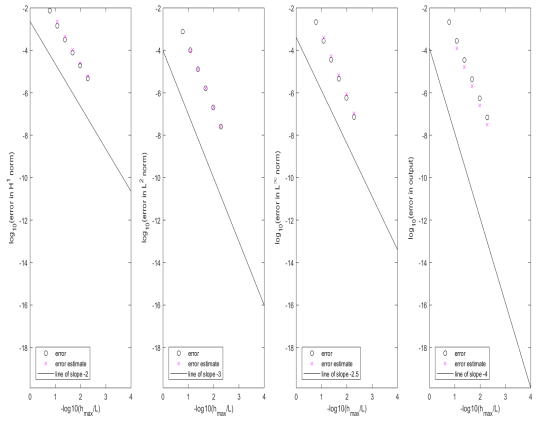


(a) p=1, $\theta = 1$

.

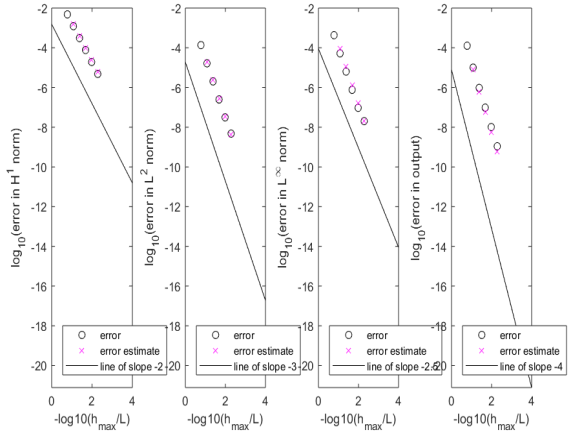


(b) p=1, $\theta = 0.5$



(c) p=2, $\theta = 1$

.



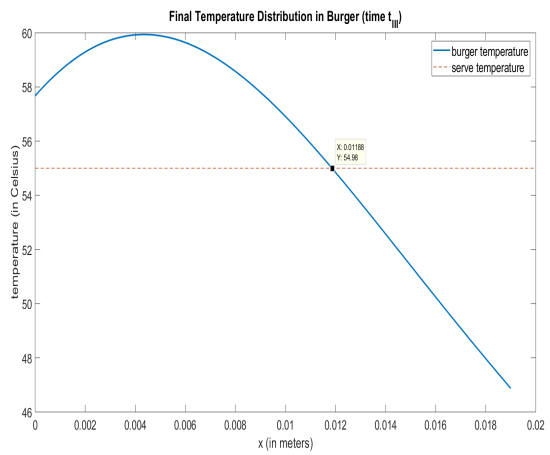
(d) p=2, $\theta = 0.5$

.

Figure 22: convergence rates for the semiinf model across different values of p and θ for an initial mesh of 6 elements and initial number of $n_{tsteps} = 10$, up to 5 uniform refinements

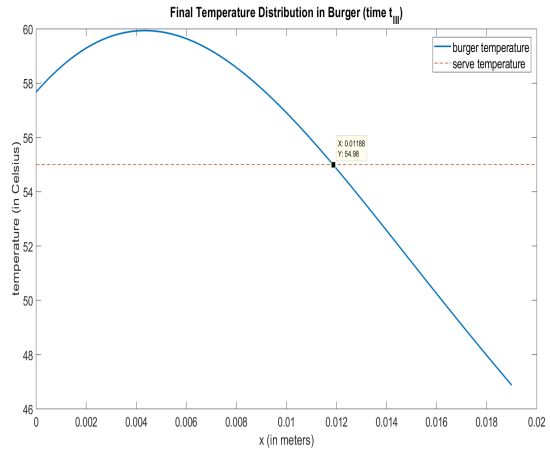
For evaluating correct numerical implementation of the burger code, we first note that we have available the approximation results, at relatively tight error tolerances, from a reference code (Professor Patera's), albeit both codes use similar numerical approximation techniques. Hence, a first step of evaluation of correct implementation would be direct comparison of results of the two codes:

Using MATLAB's Datatip to perform precise comparisons for a few selected points (i.e comparing between subfigures a and b and between subfigures c and d in figure 23 verifies that the results of my code and Professor Patera's code

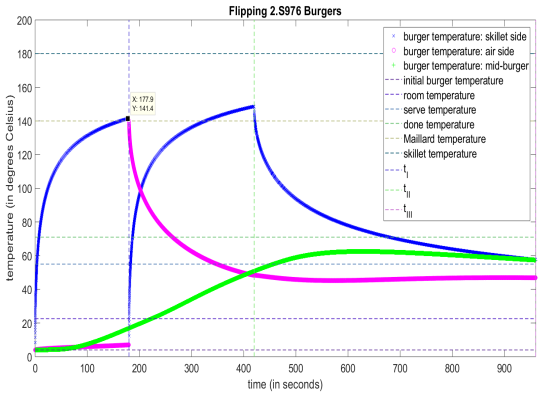


(a) Ali's code's result for final temperature distribution

.

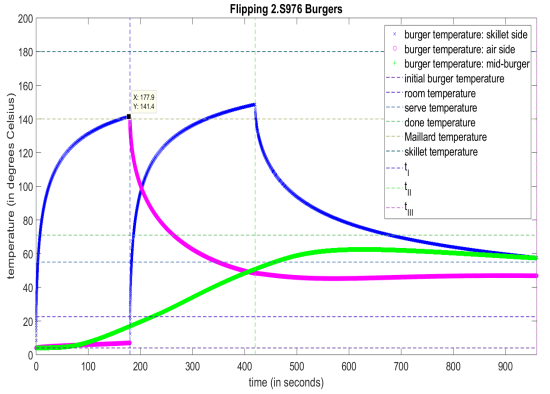


(b) Professor Patera's code's result for final temperature distribution



(c) Ali's code's results for evolution of temperatures

.



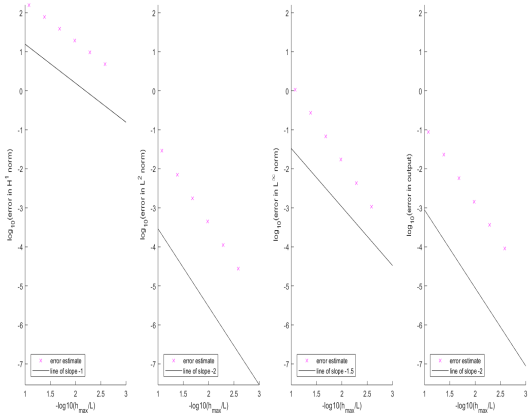
(d) Professor Patera's code's results for evolution of temperatures

.

Figure 23: Comparison of the code results between Ali's code and Professor Patera's code for burger flipping, for p=1, θ=1, starting with 6 elements in the first mesh and with 20 timesteps and conducting four refinements (hence 96 elements and 5120 timesteps). The results are identical: both burgers, as given by the Datatips on the figures, have a final temperature of 54.98C at 0.01188m (sub-figures a and b), and both have a skillet temperature of 141.4C at 177.9s (sub-figures c and d). The correspondence between the results of the two codes verifies Ali's code's correct numerical implementation (assuming Professor Patera's code implementation is correct).

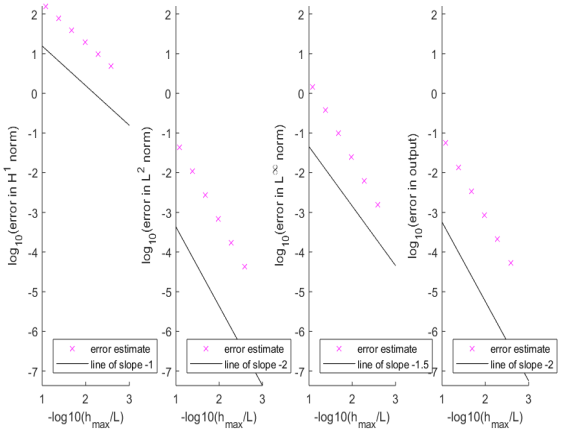
are the same, thereby providing confidence of correct implementation of my code. Next, we further verify correct implementation by looking at the convergence rates across the different values of p and θ.

Examining figure 24, indeed, the error estimates do converge as the discretization increases, and more specifically, they do converge at the right rate (slope) equivalent to -r, with the expected exception for the output error for case p=2, where the slope is slightly less steep (due to the refinement being suboptimal in space as explained for the semiinf model), as seen in figures 24 c and d. It is worth noting that there is some deviation from the slope for the L_∞ norm in case of p=1, as seen in figures 24 a and b. This is because for this error norm, for p=1, the error probably converges with $h^2 log(h)$ as opposed to the more conservative convergence rate with $h^{\frac{3}{2}}$, just as in the semiinf model. The agreement of the error convergence rates (in general), along with the agreement in the results with the reference code, provide confidence of correct implementation.

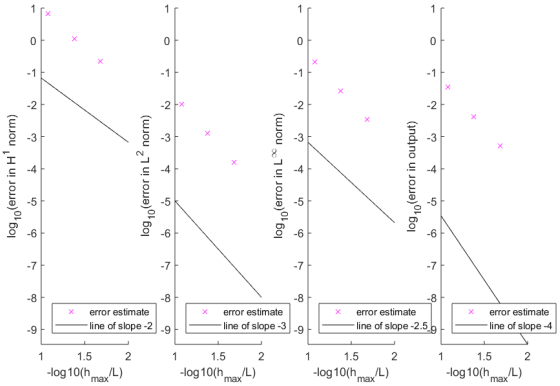


(a) p=1, θ =1, 6 uniform refinements

.

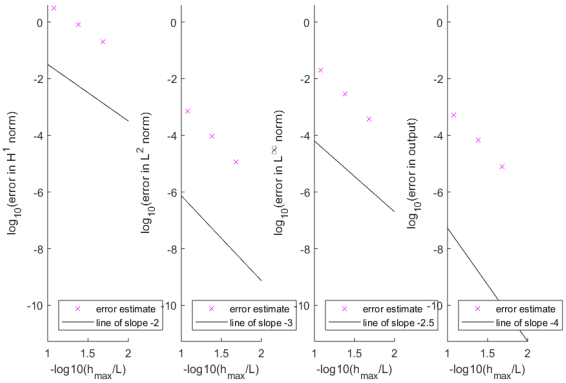


(b) p=1, θ =0.5, 6 uniform refinements



(c) p=2 θ=1, 3 uniform refinements

.



(d) p=2 θ=0.5, 3 uniform refinements

.

Figure 24: Convergence Rates for the burger flipping model for each combination of p and θ, starting from 6 elements for the initial mesh and 20 timesteps.

3.3 Task 5

For task 5, we want the coarsest mesh for which we believe the error in the output for the burger temperature at the skillet side at time t_I (just before the flip), is less than 0.001 C. For p=1, the error estimate is an upper bound for the actual output error, hence we are looking for the coarsest mesh for which:

$$\begin{aligned} log(error) &< log(0.001) \\ &= -3 \end{aligned} \tag{192}$$

For the case of p=1, $\theta = 1$ (in figure 24), the coarsest mesh with an estimated output error of less than 0.001C is mesh 6 (after 5 refinements), with log(output error) = -3.443 (hence an estimated output error of 0.00036C. This corresponds to $6 \times 2^5 = 192$ elements.

For the case of p=2, we note that we need to multiply the estimated output error by a factor of $\frac{2^r-1}{2^{r-1}+1}$ which is ≈2 for p=2, in order for the output error estimate to provide an upper bound for the actual output error. Hence for a mesh that we believe would give an error of less than 0.001C, we are looking for a mesh that, on the plotted output error figure, would give:

$$\begin{aligned} log(error) &< log(0.0005) \\ &= -3.3 \end{aligned} \tag{193}$$

This would correspond to the third mesh (after 2 refinements), with a log(error) value of -4.175, and a number of elements of $6 \times 2^2 = 24$ elements.

To find an order of magnitude approximation for how the computational cost for the case p=1, θ=1 compares to the case for which p=2, θ= 0.5, we note that for each ODE in time after conducting our spatial discretization, the computational cost scales with the number of timesteps $[O(n_{tsteps})]$.

Furthermore, for the initial spatial discretization required to arrive to the time dependent ODEs, the computational cost for forming the RHS and solving for the LHS in equation 183 scales with our number of nodes, [O(n)], which, for p=1, would be (almost) the same as the number of elements (plus 1), and hence an equivalent computational cost in Big O notation of O(m), where m is the number of elements.

Meanwhile, for p=2, we are solving a pentadiagonal matrix as opposed to a tridiagonal one with more (almost twice) number of nodes than elements. We approximate the cost of solving the pentadiagonal matrix to be twice that of solving for a similar tridiagonal matrix of the same number of elements (since we have almost twice the number of nodes), giving a cost of 2 $\times$ O(m) for p=2.

Hence, to find the ratio of computational time, we compare the product of the number of elements and the timesteps between both cases of p=1, θ=1 and p=2, $\theta = 0.5$, and factor in the fact that we are solving a pentadioagonal matrix for the latter.

The number of elements m_{new} after a particular number of refinements l, starting from an initial mesh with m_0 number of elements, is given by:

$$m_{new} = m_0 \times 2^l \tag{194}$$

Meanwhile, the number of timesteps n_{new} after a particular number of refinements 1 starting from an initial time discretization of n_0 timesteps, is given by

$$n_{new} = n_0 \times \alpha^l \tag{195}$$

For the prescribed choice of α, which is given by:

$$\alpha = \begin{cases} 4 & p = 1, \theta = 1 \\ 2\sqrt{2} & p = 2, \theta = 0.5 \end{cases} \tag{196}$$

We start from an initial mesh of 6 elements and an initial time discretization of 20 timesteps. Putting all that in, the ratio of computational cost for the prescribed accuracy of 0.001C for [p=1, θ =1] relative to [p=2, θ = 0.5] is given by:

$$\frac{O(p = 1, \theta = 1)}{O(p = 2, \theta = 0.5)} = \frac{6 \times 2^5}{6 \times 2^2} \times \frac{20 \times 4^5}{20 \times (2\sqrt{2})^2} \times \frac{1}{2} = 512. \tag{197}$$

We could see that using p=2, θ 0.5 for our finite element/finite differences scheme is much more computationally efficient compared to using p=1, θ 1. . While it is true that in general using the more elaborate scheme, which in our case is p=2, θ 0.5, would be more computationally expensive (for instance, for the same level of refinement) than less elaborate schemes, it is much more accurate that, for a prescribed level of accuracy, it requires significantly less number of meshes such that the overall computational cost ends up being lower.

3.4 Task 6

The quest for the perfect burger has bee innitiated by many seekers of knowledge, and below I mention some the guidelines that help in choosing the paramter values for the burger code. According to Jamielyn on the food and lifestyle blog Iheartnap[1] I (attempted to) follow during the summer:

1. The burger patty should be 3/4 inches thick, which is about 0.019m burger thickness.
2. The burger should be placed on the grill and must cooked for 4-5 minutes (before flipping). Hence we assign the duration for the first stage to be 270s.
3. The burger should then be flipped and cooked for an additional 4-5 minutes (or until juices clear), hence we assign the duration of the second stage to be also 270s.

According to Amazing ribs:[2]

1. The burger should be large enough to overhang the edges of the patty, usually 4.5 inches, which translates to 0.114m burger diameter.
2. The cast iron griddle or heavy frying pan should be heated to about at least 350 F, hence $T_{skillet}$ is 176 C. This is to ensure that the burger surface reaches the temperature of Maillard browning of 285F (hece $T_{Maillard} = 140.5$ C), the temperature at which the meat's sugar and protein react with each other, favoring an unstable structure, which breaks down into hundreds of flavorful compounds that make meat taste more savory, caramelized and delicious.[1]

Other factors to note:

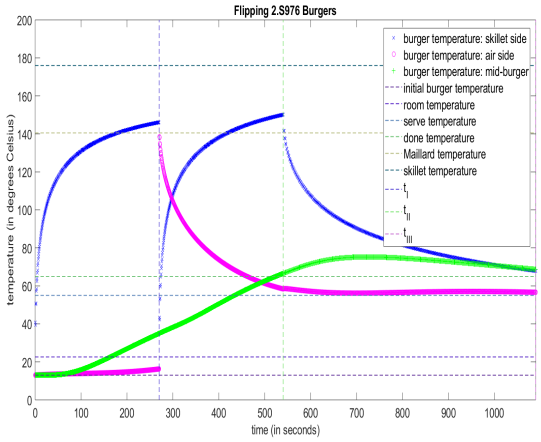
1. Bacteria is killed at a temperature of about 140 F.[3] Hence the meat throughout the whole burger (in particular, the mid burger section since it would have the lowest maximum temperature reached) should reach 145F at some point. Since I like my burger to be medium done, We assign T_{done} to be 65C.
2. I usually leave my burgers at room temperature for a while before cooking/grilling them, so we will consider an initial temperature equivalent to the room temperature T_∞.
3. While recipes did suggest indirect grilling (moving burgers from high to low heat zone) which corresponds to stage 3 repose, I could not find specific guidelines for the duration, hence we will leave stage 3 duration almost as is.

Table 11 summarizes the chosen parameter values, and figure 25 shows the run results.

Parameter	Value	Units
Burger Thickness	0.019	m
Burger Diameter	0.114	m
$T_{skillet}$	176	C
$T_{Malliard}$	140.4	C
T_{done}	65	C
T_{serve}	55	C
Stage One Duration	270	s
Stage Two Duration	270	s
Stage Three Duration	550	s

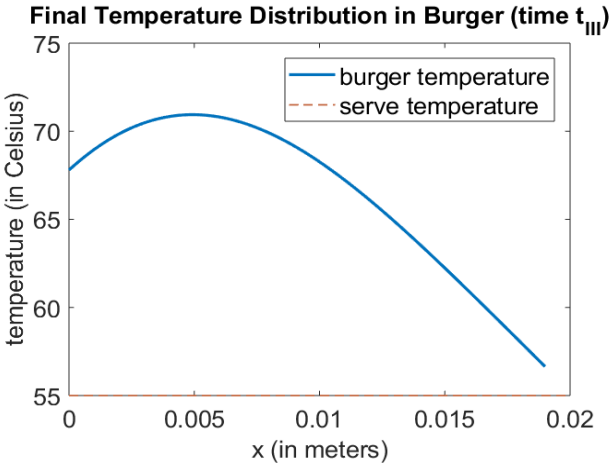
Table 11: Parameter values for try 1 of my burger flipping scheme

Looking at figure 25a, the temperature of the two surfaces do indeed reach the Maillard temperature required for the desired browning, and the mid burger (along with the temperature of the two burger surfaces) does reach T_{done}.



(a) Temperature Evolution through time for burger flipping

.

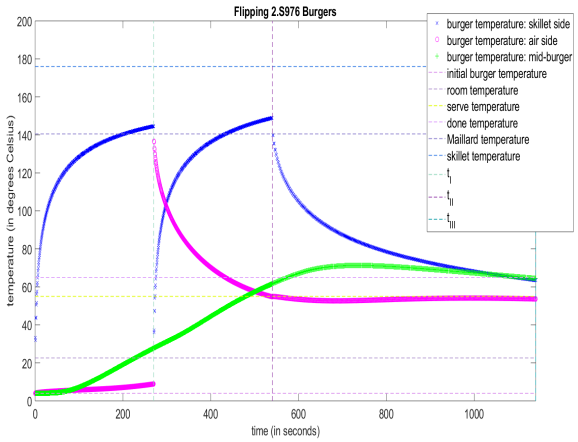


(b) Temperature profile at the end time

Figure 25: Try 1 Simulation Outputs using parameter values in table 11 using p=2 and $\theta = 0.5$.

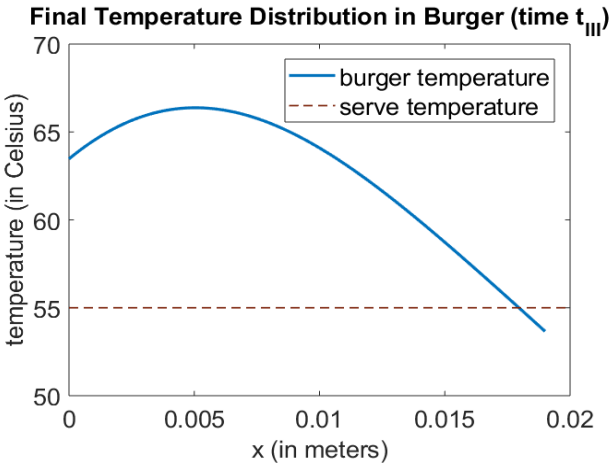
However, looking at subfigure b, unfortunately the burger is still too hot as the temperature at the final time is greater than the serve temperature throughout the whole burger, and the temperature distribution is not witihin 5C of the serving temperature.

Afterall, perhaps my idea of leaving the burgers at room temperature before cooking is not so good afterall. Hence, this time I do decide to keep the initial temperature at 4C, and perhaps extend the duration of stage 3 during the which the burger is cooling down to 600s in efforts to cool the burger to the serve temperature. If such effects do indeed lower the final temperature distribution then this also (somehow) provides some confidence of correct implementation.



(a) Temperature Evolution through time for burger flipping

.



(b) Temperature profile at the end time

Figure 26: Try 2 Simulation run outputs using parameter values in table 11 but with $T_{in} = 4$C and Stage three duration of 600s, using p=2 and $\theta = 0.5$

Looking at figure 26 a, we still reach the desired malliard temperature at the surfaces and T_{done} throughout the burger, but now we also got a lower the final temperature of the burger (as one would theoretically expect from increasing stage 3 duration and lowering initial temperature) such that now (albeit only part of) the burger has a temperature of T_{serve} at some point in space. Yet most of the burger is still not within 5C of serving temperature. Perhaps my cooking strategies need revising, and thankfully Professor Patera's code saved me from burning everyone's burger during the CPW grill. Well this might not be the case because even though the implementation might be correct (and there is plenty of evidence supporting correct numerical implementation), the inherent mathematical model,

whether the modeling PDE or the parameter values and behavior, might not fully capture what is happening in the physical world. For instance, as mentioned in the chapter deliverables document:

1. The effects of moisture are ignored.
2. The skillet temperature is assumed constant in time.
3. The thermal properties from the literature are not overly well characterized, and may depend on the type/ any extra additions to meat material especially if it was not commercially grounded.
4. The natural convection and radiation boundary conditions are linearized.
5. The lateral Biot number is a bit above the limit for comfortable application of quasi one-dimensional heat transfer (i.e you cannot necessarily assume the temperature to be uniform over the cross sectional area for a particular value of x).

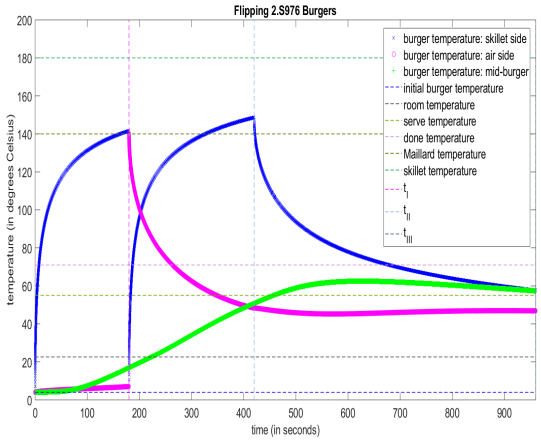
Other trials worth investigating include

1. Flipping the burger every 20/30s to "drive much more temperature throughout the meat",[3] which would require multiple stages and less than trivial modifications to the code.
2. Since the Malliard crust is desired, it was recommended to increase the surface area (that is prone to crusting) by splitting the patty into two thinner patties, 'smash into hot griddle and flip'[2] browning each of the four sides before combining together. In our code, this could be simply accounted for by halving the thickness of the burger.

Although the diameter is unreasonable for me, I thought of conducting one more trial for what a 'Gastrophysicist' (food scientist) from University of Oxford claimed to be the perfect dimensions for a burger.[4] It is also worth mentioning that the scientist mentions that it is imperative for the burger "to be eaten with music, given a name, and should feel as good as it tastes," (these factors are a bit trickier to implement in the code). The dimensions recommended are:

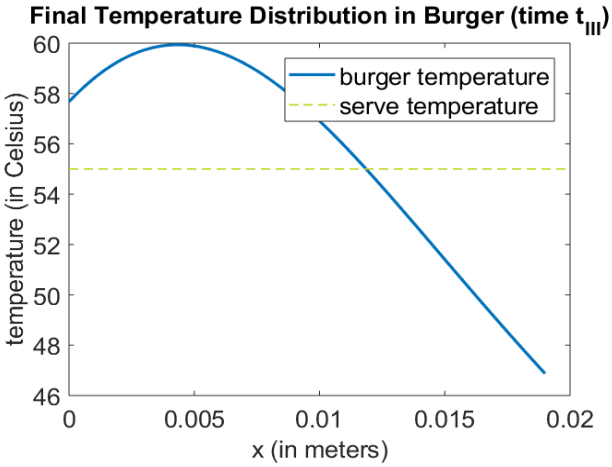
1. The burger patty should be 7cm tall (with 2cm height made of fluffy bread that can be squeezed between both hands).
2. The burger patty should be 5cm wide.

Hence we run the simulation with burger thickness = 0.07m, and diameter = 0.05m, using the default values for the other parameters. It does indeed look like the scientists recommendations hit all the benchmarks; from figure a, both surfaces reach the maillard temperature at some point, the mid burger (as well as the surfaces) reach T_{done}, and from figure b, the burger temperature distribution is more clustered around the serving temperature compared to my trials, with the maximum temperature being within the 5C of the serve temperature.



(a) Temperature Evolution through time for burger flipping

.



(b) Temperature profile at the end time

Figure 27: Try 3 Simulation Outputs using parameter values in table 11 but with burger diameter of 5 cm and burger thickness of 7 cm, approximated using p=2 and $\theta = 0.5$

4 Chapter Four: The Finite Element Method for One Dimensional Fourth Order Boundary Value Problems (Bending): Xylophone

4.1 Finite Element Method for Beam Eigenvalues

As mentioned in chapter 2, we could think of the finite element method as a specific case of the RR method for approximation of a function over a domain. In this case, our basis functions are actually piece wise functions that are only non-zero at specific regions in the domain, and are thus more capable of capturing local effects of the function. In our case, the PDE for our bar of length L, following the Euler Bernoulli model (as long as it sufficiently slender, meaning L >> H such that shear strain could be neglected) with axial loading N_0 is given in equation 198.

$$\frac{d}{dx^2}(\beta(x)\frac{d^2u}{dx^2}) - N_0\frac{d^2u}{dx^2} = q(x) \qquad (198)$$
$$0 \leq x \leq L$$

For boundary conditions, while essential boundary conditions are imposed imposed on the primary variables (deflection/displacement and its first derivative), natural boundary conditions are imposed on the secondary variables, or higher derivatives like shear forces and moments and will automatically be satisfied after approximating the solution of the problem (given sufficient accuracy of the FE method).

And in the case of all natural boundary conditions, our boundary conditions corresponding to a free bar that need to be provided would be:

1. M(0) = $\beta(0)u_{xx}(0)$
2. V(0) = -$(\beta u_{xx})_x(0)$
3. M(L) = $\beta(L)u_{xx}(L)$
4. V(L) = -$(\beta u_{xx})_x(L)$

Just as we handled any RR method, basically there is an energy functional for our system that we wish to minimize. For an Euler Bernoulli bar, it is worth noting that the energy functional being minimized actually has physical

relevance; it captures the elastic energy of the bar. Equation 199 shows the generic energy functional expression for an Euler-Bernoulli bar under no axial load, subject to natural boundary conditions. The candidate function ω, in this case, represents the vertical displacement/deflection along the length of the bar. Notice how the energy functional depends on both the candidate function, or the displacement, and the second derivative of the candidate function with respect to x, or the moment ($\beta(x)\omega_{xx}^2$), both of which are physically relevant parameters when considering the energy functional of the bar.

$$\pi(\omega) = \frac{1}{2}\int_0^L \beta(x)\omega_{xx}^2 + N_0\omega_x^2\,dx - \int_0^L q(x)\omega\,dx \underbrace{-M(0)\omega_x(0) - V(0)\omega(0) - M(L)\omega_x(L) - V(L)\omega_x(L)}_{\text{natural boundary condition terms}} \tag{199}$$

As mentioned in the previous chapters, the best approximation of the exact solution in equation 198 is the one that minimizes the energy functional, which is found by solving the system of equations:

$$\underline{A}u_h = \underline{F} \tag{200}$$

In the bending case, we require approximations for both the displacement (deflection), and it's first derivative in terms of x (Hermitian approximations). Just as before, for spatial discretization, we divide the domain into elements that span the whole domain:

$$\begin{aligned} Elements : T^m,\ 1 \le m \le n_{el} \\ \overline{\Omega} = \bigcup_{m=1}^{n_{el}} T^m \end{aligned} \tag{201}$$

Where each element's starting and ending point is defined by a node:

$$\begin{aligned} &Node : x^i,\ 1 \le i \le n_{node} \\ &x^1 = 0 < x^2 ... < x^{n_{node}} = L \\ &n_{node} = n_{el} + 1 \end{aligned} \tag{202}$$

We define the element length, h^m (which could be uniform across elements or not):

$$h^m = x^{m+1} - x^m,\ 1 \le m \le n_{el} \tag{203}$$

And, just as before, we use our special piecewise functions as our basis functions for RR approximation:

$$\psi_i(x) = \varphi_i(x),\ 1 \le i \le n_{node} \tag{204}$$

However, for the Hermitian approximation, we need to approximate both the actual solution and its derivative, at the nodes defined according to the spatial discretization. Hence both these pieces of information need to be captured in the vector u_h we are solving for. In other words, we need a method to map 2 pieces of information for each node (the approximation to the function value at the node and the approximation to the function derivative value at the node), into a single vector that we could solve for. This requires some form a mapping from double index to single

index. Let us represent the node number by i, and the particular piece of information (function value or derivative value) by k. Hence k would take on the values:

$$k = \begin{cases} 1 & \text{function value} \\ 2 & \text{function derivative} \end{cases} \tag{205}$$

Now we could map both the approximation of the function value and its derivative for all nodes into a single vector. In the vector, the index (position of element) of the vector would give information about both the node number and whether the value of the index represents function value or derivative, according to the following double index to single index mapping:

$$[i,k]^{2\leftarrow 1} = 2(i-1)+k \tag{206}$$

Such mapping is unique, in the sense that no two double index values give rise to the same single index value. For each entry in our 'weighing' vector u_h, we need a corresponding base function. Remember that the approximation to the function is the weighted sum of all base functions:

$$u_h = \sum_{j=1}^{n} u_{hj}\varphi_j(x) \tag{207}$$

In brief terms, there are two sets of base functions, one set captures the the function values, and the other set captures the value of the derivatives. In both these sets of functions, each individual base function is cubic over both the element corresponding to its node, and the next element. For the first set of base functions capturing the function value, each individual base function has a value of 1 only at its corresponding node, and zero elsewhere. As for the second set of functions, each individual base function has a derivative value of 1 at its corresponding node, and zero elsewhere. Furthermore, each individual function in set 2 has a zero function value at all nodes. and each individual function in set 1 has a zero derivative value (critical point) at all nodes (hence if we wanted to evaluate the function derivative value at the nodes we would use the second set of base functions).

Let us represent the base function corresponding to a particular node i, belonging to the first set as $\phi_{i,1}$. From the previous discussion we would get

$$\phi_{i,1}(x) = \begin{cases} 1 & x = x_i \\ 0 & \text{for all other nodes} \end{cases} \tag{208}$$

and

$$\phi'_{i,1}(x) = 0 \quad \text{at all nodes} \tag{209}$$

Similarly,let us represent the base function corresponding to a particular node i, belonging to the second set as $\phi_{i,2}$. Hence we would get

$$\phi'_{i,2}(x) = \begin{cases} 1 & x = x_i \\ 0 & \text{for all other nodes} \end{cases} \tag{210}$$

and

$$\phi_{i,2}(x) = 0 \quad \text{at all nodes} \tag{211}$$

Notice that now the size of of our vector u_h (and number of base functions) is now 2 $\times n_{node}$ since we have two entries for each node. Also note if we wanted to evaluate the function value any nodes by the weighted sum of our base functions we would use only be factoring in first set of base functions, and in particular the 'first weight' of the base function at that node:

$$u_h(x^l) = \sum_{i=1}^{n_{node}} \sum_{k=1}^{2} u_{h\ i,k} \underbrace{\phi_{i,k}(x^l)}_{\text{non-zero for only} i=l \text{ and } k=1} = u_{h\ l,1} \tag{212}$$

Similarly, to evaluate the value of the function derivative at a particular node, only the derivative of the second set of base functions factors in, and in particular, only the 'second weight' of the base function at that node.

$$u_h'(x^l) = \sum_{i=1}^{n_{node}} \sum_{k=1}^{2} u_{h\ i,k} \underbrace{\phi_{i,k}'(x^l)}_{\text{non-zero for only} i=2 \text{ and } k=1} = u_{h\ l,2} \tag{213}$$

Just as explained in chapter 3, for implementation (and in brief summary), we create a generic 'reference element' that is representative of all potential elements in the domain. We define the domain of this reference element to be:

$$\hat{T} = [0, 1] \tag{214}$$

such that the left and right boundary nodes of our domain are defined by:

$$\begin{aligned} \hat{x}^1 &= 0 \\ \hat{x}^2 &= 1 \end{aligned} \tag{215}$$

And then we add our overlapping generic base functions onto our reference element. Four generic base functions could overlap in a single element (two pairs of functions that capture the value, and its derivative, one pair corresponding to each of the start and end node). This means we have four degrees of freedom over our reference element. The mapping, evaluating of integrals via quadrature,...etc is discussed thoroughly in lecture notes and is very similar to what has been explained previously in chapter 3. One interesting thing worth noting is that before we actually do the mapping, we scale the functions belonging to the second set (that have to do with the derivative) with the length of the element being mapped to, such that ultimately the derivatives are 1 at the appropriate nodes (before being scaled by the appropriate element in vector u_h to retrieve the final derivative value).

Now for beam eigenvalue problems, as a result of moments being in equilibrium and applying Newton's third law, we arrive to our PDE that has a time-varying component:

$$\frac{\partial^2}{\partial x^2}(\beta(x)\frac{\partial^2 u}{\partial x^2}) - N_0\frac{\partial^2 u}{\partial x^2} = q(x,t) - A_{cs}\rho\frac{\partial^2 u}{\partial t^2} \tag{216}$$

As discussed, the PDE could have multiple combinations of (four) essential and natural bounary conditions. As for the inital conditions, we need information about both the initial displacement and velocity at time 0 over the domain:

$$\begin{aligned} u(x, t=0) &= u_{ic} \ \ x\ in\ \Omega \\ \dot{u}(x, t=0) &= \dot{u}_{ic} \ \ x\ in\ \Omega \end{aligned} \tag{217}$$

The displacement of the bar along the length of the bar/beam is produced from a combination of the bar's (infinite) vibrational modes. Assuming q(x,t) = 0, then the contribution of the modes to the displacement along the beam could be represented as:

$$u(x,t) = \sum_{k=1}^{\infty} \Big(c_1^{(k)} cos(\omega^{(k)} t) + c_2^{(k)} sin(\omega^{(k)} t) \Big) \tag{218}$$

In a natural vibration, when you excite the object once, it will vibrate for a while. Those are the natural vibrations of the object, represented by $w^{(k)}$. Where c_1 and c_2 for each mode are determined from the initial conditions. Usually the fundamental (lowest) frequency is perceived as the loudest (highest amplitude).

To translate this into an eigenproblem, we note that when taking the second derivative of the sinusoidal functions in equation 218, we get the sinusoidal functions scaled to the (negative) square of the natural frequency corresponding to the mode, or what we might call the eigenvalue:

$$\begin{bmatrix} cos(\omega t) \\ sin(\omega t) \end{bmatrix} = - \underbrace{\omega^2}_{\lambda} \begin{bmatrix} cos(\omega t) \\ sin(\omega t) \end{bmatrix} \tag{219}$$

Hence we could substitute the second derivative, $\frac{\partial^2 u}{\partial t^2}$ in equation 216 with -λu to arrive to our now ordinary (only varying in space) differential equation system (for each mode k):

$$\frac{d^2}{dx^2}(\beta(x)\frac{d^2 u^{(k)}}{dx^2}) - N_0 \frac{\partial^2 u^{(k)}}{dx^2} = \lambda^{(k)} A_{cs} \rho u^{(k)} \tag{220}$$

In which, for the eigenvalue corresponding to a particular mode, we solve for the displacement along the beam corresponding to the eigenvalue. For the xylophone bar problem, our beam is basically a xylophone bar that is modeled to be free on both ends (natural boundary conditions corresponding to the zero moment and shear force at both ends). The governing eigenproblem (in dimensional form) for each mode is given by equation 221:

$$\begin{aligned} &\frac{d^2}{dx^2}(\frac{E_d W_d H_d(x_d)}{12}\frac{d^2 u^{(k)}}{dx^2}) - N_0 \frac{\partial^2 u^{(k)}}{dx^2} = \lambda^{(k)} W_d H_d(x_d) \rho_d u^{(k)} \\ &0 \leq x_d \leq L_d \end{aligned} \tag{221}$$

Subject to the free-free (zero moment and shear force) at both ends x=0 and x=$L - d$:

$$u_{xx}^{(k)}(0) = u_{xxx}^{(k)}(0) = u_{xx}^{(k)}(L) = u_{xxx}^{(k)}(L) = 0 \tag{222}$$

Strings are inserted into hole positions that correspond to the nodes of the fundamental node where the displacement is zero, to minimize the force on the bar. In the case with no strings, the first and second eigenvalues are zero, due to translational and rotational freedom. The third eigenvalue would correspond to the fundamental frequency, in which the relationship between the dimensional frequency and eigenvalue is given in equation 223. However, in the case in which there are strings, or the bar is not allowed to translate or rotate, then the two zero eigenvalues disappear, and the fundamental mode is then the first eigenvalue.

$$\begin{aligned} &\omega_d^{(k)} = \sqrt{\lambda_d^{(k)}} \text{ angular frequency in rad/s} \\ &f_d^{(k)}) = \frac{\omega_d^{(k)}}{2\pi} \end{aligned} \tag{223}$$

In summary, what we do is we convert our system into a non-dimensional form and solve it using the prescribed FE method. When we design the bar, we want to tune it to have a particular fundamental (corresponding to the third eigenvalue) frequency which is the playing note or pitch, and we also manipulate the shape of the bar to produce a particular frequency ratio R, which the frequency of the first harmonic frequency (corresponding to the fourth eigenvalue) to the fundamental frequency. Theoretically, there are many possible bar shapes that would emit the

desired frequencies, so it is necessary to mathematically constrain the problem of determining the exact bar shape in order to make it solvable. Fourth order polynomials (p_1=4) are chosen to describe the undercut shape of the bar, because it is simple.

First, the frequency ratio R, would depend on the shape/carving of the bar. We constraint certain parameters of the carving, such as the maximum height of the bar (before carving), the beginning and end of the carving along the length of the bar (symmetrical from both ends), and the polynomial order that describes the shape of the carving (p_1 =4). Then, what we optimize in the carving is the value of p_2, the ratio of the minimal to maximal height of the bar along its length, in such a way that we obtain our desired frequency ratio R (within the prescribed tolerance).

Now we have the value of p_2 corresponding to the carving that would give the desired R frequency ratio, and thus we also have to height profile along the length of the bar. The next step is to tune the bar to the desired fundamental frequency such that the actual fundamental frequency is within the prescribed tolerance of the desired fundamental frequency. The fundamental frequency of the bar depends on its length, as given by the equation:

$$L_{d\,h} = \left(\frac{f_h^{(3)}}{f_{target\ d}^{(3)}}\right)^{\frac{1}{2}} \left(\frac{E_d H_{max\ d}^2}{\rho_d}\right)^{\frac{1}{4}} \tag{224}$$

Finally, as mentioned and will be discussed later, we place the holes in the bar where the string will pass through such that the holes pass through the nodes where there is zero bar displacement to minimize the force of the strings on the bar.

4.2 Task 1

Care must be taken to determine the hole locations, because it is important to minimize the damping effect of the strings on the vibrational modes. The nodes of the fundamental mode, are the ones that experience zero modal displacement (deflection). The optimal place to string the bars would logically be the nodes, because holes placed at the undeflected nodes would minimize the effects on the bars vibration and frequencies. Hence before we 'drill' or 'insert' the holes we need to determine the location of the bar nodes:

1. Find the elements over which the deflection, or displacement, changes sign: when the displacement of the bar goes from being above the neutral axis to below the neutral axis in a particular element, this means at some point in between there is no displacement of the bar. I do that (perhaps not the most efficient way) by iterating over each element, and if the product of the displacement at the left and right end nodes of an element has a negative sign, it means somewhere along the element the displacment is zero:

```
      element_ids = [];
for m = 1: n_el
    if u3(lg2(1,m))*u3(lg2(3,m)) < 0
        element_ids(end+1) = m;
    end
end
```

2. A zero displacement node means the (weighted) sum of all our basis functions evaluated at the node is equal to zero. Hence we first construct a function for each of the two elements with a node (the function is in terms of the dummy variable $\hat{x}$ which represents the scaled position along each element from 0 to 1). To create such function, first we construct the 4 by 1 vector in which each entry corresponds the the weight of each of the four shape functions, and we take the dot product of this vector with the vector corresponding the four base generic base functions (as a function of our dummy variable $\hat{x}$) scaled to the reference element. We obtain two functions, one for each element where the hole should be placed, and each function represents the displacement along the element as a function of the scaled position over that element (from 0 to 1) constructed by summing the contributions of our base functions over the element:

```
u_vec_1 = [];
```

```
u_vec_2 = [];
for i=1:4
  u_vec_1(end+1) =  u3(lg2(i,element_ids(1)));
  u_vec_2(end+1) =  u3(lg2(i,element_ids(2)));
end

function_1 = @(xhat) u_vec_1 * hshape_fcn(xhat, h(element_ids(1)));
function_2 = @(xhat) u_vec_2 * hshape_fcn(xhat, h(element_ids(2)));
```

2. Now we simply use the fzero function to find the scaled position over each element where our weighted sum function, corresponding to the displacement, is zero:

```
first_zero = fzero(function_1, [0,1]);
second_zero = fzero(function_2, [0,1]);
```

3. Finally we find the (dimensional) position of the nodes where the zero occurs, by finding the scaled (non-dimensional) position along the bar where such nodes occur, and multiplying that by the length of the bar to get the dimensional position (length from start of the bar) of the holes:

```
xhole_d = [(xpts(lg(1,element_ids(1))) +h(element_ids(1))*first_zero) * L_d,...
    (xpts(lg(1,element_ids(2))) +h(element_ids(2))*second_zero)*L_d];
```

For a particular run implementation of a rosewood xylophone bar, the length of the bar required to achieve the desired fundamental frequency is 0.2667m. We can see from figure 28 that the dimensional location of the holes where the strings will pass through, provided as output from our xylo-design function, which are 0.0551m and 0.2116m, match the position of the nodes, the points along the bar of zero modal displacement for the fundamental mode.

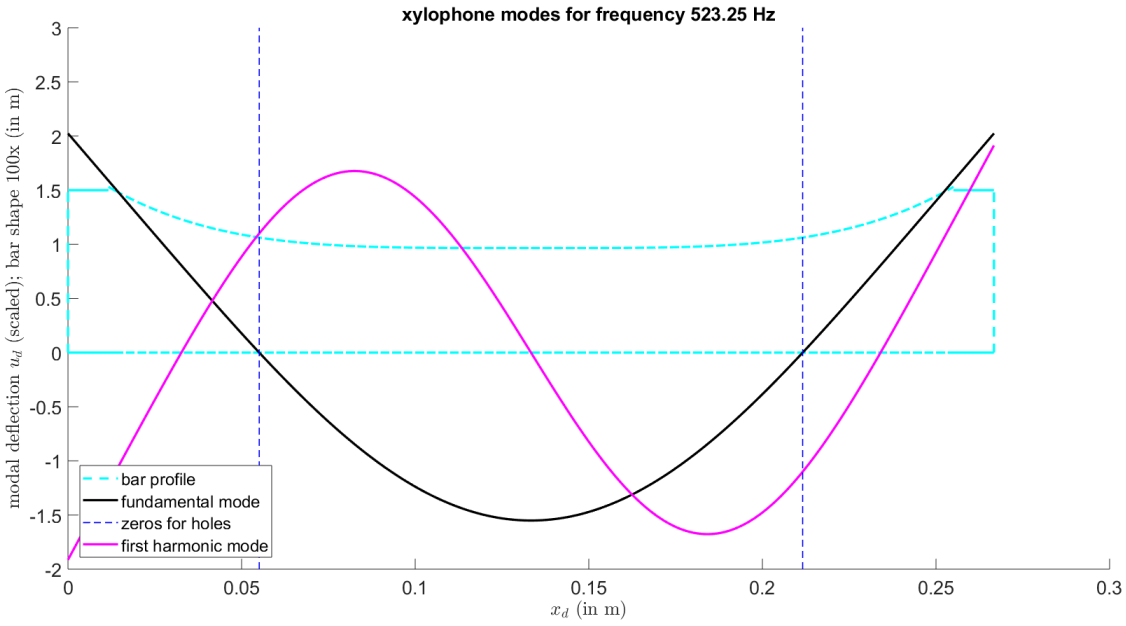


Figure 28: The modal deflection of a rosewood xylophone bar tuned at 523.25 HZ fundamental frequency (C5 note), with a frequency ratio (first harmonic to fundamental) 0f 3. The length of the bar (required to achieve the desired fundamental frequency) is 0.2667m. The position of the holes at 0.0551m and 0.2116m, match the position of the nodes of zero modal deflection for the fundamental mode.

4.3 Task 2

In the Caresta paper, we are provided with an example of a slender (L >> H) beam (with no chopping hence p_2=1), and are provided with the properties of the beam, including its material properties (Young's modulus and density), beam height, shape, and beam length. The paper, just our code, also treats the beam of interest as a an Euler-Bernoulli beam that has free boundary conditions at both ends, and numerically solves the corresponding derived equations for a beam of a particular length to arrive to the first theoretical natural frequencies of the beam when it is in bending vibration. The author also experimentally verified those frequencies, which match with the theoretically derived frequencies for the beam to a great extent.

This beam and the information can provide a very insightful example to assess certain aspects of the xylo bar design3 code. Since both the paper and the code treat the beam as an Euler-Bernoulli beam, we could use the information from the paper to verify our code's correct numerical implementation of the Euler Bernoulli method.

What we do is we run our code with the material properties of the beam of interest, and the desired fundamental frequency of the beam as inputs to our code, which are summarized in table 12. We initially 'disregard' the other information (the first harmonic frequency and the length of the bar under vibration), which are also the outputs to our code. After running our code using the xylophone_design function, we compare the outputs of code, namely the fundamental frequency of the resultant virtual bar (to see if our 'produced' bar maintains the target fundamental frequency), the first harmonic frequency (given no carving), and length of the bar required to produce the desired fundamental frequency, to the actual natural frequencies obtained by Caresta, and the length of beam subject to vibration. An agreement (within the error tolerance) between our code output and Caresta's information about the bar provides (very) strong confidence that our function is indeed correctly implementing the Euler-Bernoulli method for handling beams. Note that since the information given is for beam of rectangular cross section (with no carving), the results can not serve to test the carving part (binary chop) of function xylo bar design3 which determines p_{2op}t for a given frequency ratio R target between the first harmonic and fundamental frequency. To match the beam in Caresta's paper, we run the code with no carving (p_2 interval of [1.0, 1.0], which enables us to compare our outputs of frequencies and length to the frequencies and length of Caresta's bar (hence here we are fixing p_2 more as an input than an output). This would allow us to assess the function's ability to calculate the the fundamental and first harmonic frequency, and also its ability to calculate the length of the bar required to realize the desired dimensional fundamental frequency (all outputs of our function), though it will not allow us to assess the function's ability to optimize the carving (determining p_{2opt}) to achieve the desired R ratio. The output results compared to Caresta's beam information are summarized in table 13.

Input	Value	Unit
Young's Modulus	2.1 $\times 10^1 1$	N/m^2
Beam Height	0.01	m
Density	7800	kg/m^3
p_2	1	unitless

Table 12: Inputs to xylophone design function according to Caresta's study case.

Beam Property	Xylo Design Function Output	Caresta's Atual Beam Property Value	Unit
Fundamental Frequency	32.8000	32.8	Hz
First Harmonic Frequency	90.4145	90.44	Hz
Beam Length	1.2752	1.275	m

Table 13: Comparison between xylo design function output and Caresta's beam actual properties. The great agreement among values provides confidence of the the function's ability to create a bar with a length that has the desired fundamental frequency, and its ability to calculate the first harmonic frequency

The excellent agreement between the outputed values and the beam actual properties means that the beam our function outputed has the desired fundamental frequency, has the length that produces the desired fundamental frequency, and correctly calculates the first harmonic frequency of the beam, all in all providing strong confidence of the code's correct numeical implementation of the Euler Bernoulli method, at least in those aspects tested.

4.4 Task 3

For task 3, we 'tune' in our xylophone bar, meaning we provide a particular set of inputs of our choice relating to the properties of the bar bar and its frequencies, and we obtain the rest of the bar properties that would allow the xylophone bar to satisfy the inputted properties. Table 14 summarizes the inputs to the xylo bar design code for our run.

Input	Value	Units
Desired Fundamental Frequency	349.23	Hz
Desired frequency ratio (first harmonic/fundamental)	3	unitless
Maximum Height of Bar	0.05	m
Non-dimensional location of the start of bar carving (symmetric from both ends)	0.05	unitless
p_2 Interval: Allowed interval for the minimal to maximal bar height ratio	[0.05,1]	unitless
Young's Modulus	1.4×10^{10}	$\frac{N}{m^2}$
Density	835	$\frac{kg}{m^3}$

Table 14: Inputs into Xylo-design function for task 3. The material properties (Young's modulus and density) correspond to a rosewood material.

We run our xylo-design function, and obtain the following outputs summarized in table 15

Output	Value	Unit
Non-dimensional hole positions (zero deflection)	0.0674, 0.2590	unitless
Optimal p_2 Value	0.6438	unitless
Length of bar (required to achieve desired fundamental frequency)	0.3264	m
Actual Fundamental Frequency	349.23	Hz
First Harmonic Frequency	1.0446×10^3	Hz
Error estimate for fundamental frequency	5.8525×10^{-6}	Hz
Error estimate for first harmonic frequency	9.9132×10^{-5}	Hz

Table 15: Task3 Output for a Xylophone bar tuned at an F4 note corresponding to 349.23 fundamental frequency, and a frequency ratio (first harmonic to fundamental of 3.

Our ratio for the output first harmonic frequency to output fundamental frequency is given by:

$$R = \frac{f^{(4)}}{f^{(3)}} = \frac{1.0446 \times 10^3}{349.23} = 2.991151963 \tag{225}$$

To get an upper bound for the error estimate in the frequency ratio (fundamental frequency/ first harmonic frequency), we find the maximum and minimum possible ratio given the error estimates for the fundamental and first harmonic frequencies:

$$\begin{aligned} R_{max} &= \frac{f^{(4)}_{max}}{f^{(3)}_{min}} = \frac{f^{(4)} + \epsilon_4}{f^{(3)} - \epsilon_3} = \frac{1.0446 \times 10^3 + 9.9132 \times 10^{-5}}{349.23 - 5.8525 \times 10^{-6}} = 2.99152297 Hz \\ R_{min} &= \frac{f^{(4)}_{min}}{f^{(3)}_{max}} = \frac{f^{(4)} - \epsilon_4}{f^{(3)} + \epsilon_3} = \frac{1.0446 \times 10^3 - 9.9132 \times 10^{-5}}{349.23 + 5.8525 \times 10^{-6}} = 2.99151629 Hz \end{aligned} \tag{226}$$

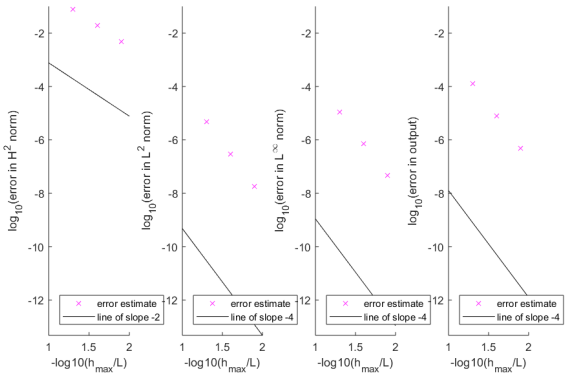
Upper bound for frequency error estimate can then be given by:

$$\begin{aligned} \text{Upper Bound} &= max(R_{max} - R, R - R_{min}) \\ &= max(3.34 \times 10^{-7},\ 3.34 \times 10^{-7}) = 3.4 \times 10^{-7}\ Hz \end{aligned} \tag{227}$$

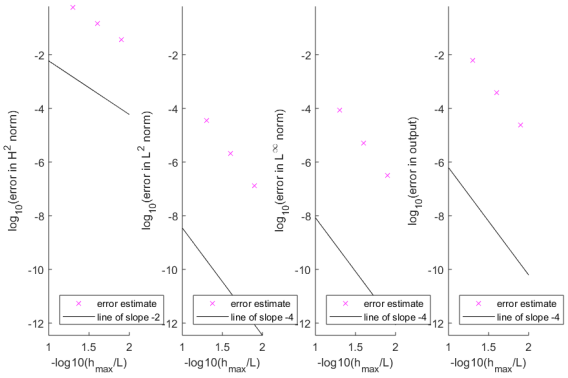
Similarly a lower bound for the error estimate in the frequency ratio could be formulated:

$$
\begin{aligned}
&|\frac{f^{(4)}}{f^{(3)}} - \frac{f^{(4)} - \epsilon_4}{f^{(3)} - \epsilon_3}| = |2.991151963 - \frac{1.0446 \times 10^3 - 9.9132 \times 10^{-5}}{349.23 + 5.8525 \times 10^{-6}}| = 2.338424 \times 10^{-7} \\
&|\frac{f^{(4)}}{f^{(3)}} - \frac{f^{(4)} + \epsilon_4}{f^{(3)} + \epsilon_3}| = |2.991151963 - \frac{1.0446 \times 10^3 + 9.9132 \times 10^{-5}}{349.23 + 5.8525 \times 10^{-6}}| = 2.336219 \times 10^{-7} \\
&\text{Minimum Error Bound} = \min\left(2.338424 \times 10^{-7},\ 2.336219 \times 10^{-7}\right) = 2.336219 \times 10^{-7}
\end{aligned}
\tag{228}
$$

Now let us tune our xylophone bar to a C5 note, corresponding to a fundamental frequency of 523.25 Hz. Looking at the error estimate graphs in figure 29, we could see that across all error norms for both the first mode (third eigenvalue) and second mode(fourth eigenvalue), the error decreases with refined meshing, and it converges at the right rate (slope in log(error) vs log(number of elements)), without any abnormal topologies, providing confidence in regards to the reliability of the error estimates.



(a) Error estimate convergence rates for the fundamental mode corresponding to the third eigenvalue



(b) Error estimate convergence rates for the first harmonic mode corresponding to the fourth eigenvalue

Figure 29: Error estimate convergence rates for the fundamental and first harmonic frequencies when tuning the rosewood bar for a C5 note of 523.5 Hz fundamental frequency and a frequency ratio (first harmonic to fundamental) of 3. The errors across all norms for both the fundamental and first harmonic modes converge at the right rates, providing confidence that the error estimates are reliable

.

Since the first harmonic mode has a higher frequency than the fundamental mode, it is going to be more wiggly compared to the fundamental mode for the same mesh, meaning it will have greater number and more sharp changes in the curvature of the function within the length domain of the xylophone bar, and I believe that the Finite Element method would less accurately capture the sharper changes in the curvature of the higher frequency function . Hence, I believe the actual FE error will be larger for frequency4_d compared to frequency3_d. Indeed, when we look at the frequency error estimates generated as outputs, the error estimate for the fundamental frequency ($8.7645 \times 10^{-06} Hz$) is less than the error estimate for the first harmonic frequency ($1.4855 \times 10^{-04} Hz$), which supports the theoretical claim.

I believe that our mesh is in fact too fine; the error in the dimensional frequencies, whether the fundamental frequency or the first harmonic frequency, according to our error estimates, are on the order of 10^{-6} Hz, which is significantly and futilely lower than the 10Hz sensitivity of the (un-trained) human ear. In other words, why refine the mesh too much to achieve an error in the frequency on the order of 10^{-6} Hz when a non-musician human can not distinguish between two pitches that differ by less than 10 Hz? Well, for one, the error describes the discrepancy between the approximated solution via the FE method to the Euler Bernoulli equation, and the actual solution to the equation. The inherent mathematical model (Euler Bernoulli equation) probably does not fully capture what is happening in real life, hence at some point (aka when the error estimate between the approximation and the exact solution is on the order of 10^{-6} Hz), tightening the error in the numerical specification via mesh refinement wouldn't be as valuable when probably sources of error in the mathematical model itself are larger.

When tuning the xylophone bar, the bar, for a particular value of $x*$ is first carved in such way to optimize the value of p_2 that would result in achieving the frequency ratio R within the error tolerance. The fundamental frequency

is tuned by optimizing the length of the bar to achieve the desired fundamental frequency. Equation 229 shows the explicit dependence of the fundamental frequency on the length of the bar

$$f_d^{(k)} = f^{(k)}\sqrt{\frac{E_d H_{max\ d}^2}{\rho_d L_d^4}} \tag{229}$$

Where $f^{(k)}$ in equation 229 is solely dependent on the shape of the bar (function of x* and p_2 only). We could see the inverse relationship between the frequency at which the bar is tuned and the length of the bar required to achieve such frequency. Hence, as the desired frequency increases, the length of the bar required to achieve such frequency decreases. As the length of the bar decreases (with constant H_{max}, the bar slenderness assumption (L >> H_{max}) to ensure the bar follows the Euler Bernoulli model becomes less valid because the shear strain/deformation, which is not accounted for in the Euler/Bernoulli model, becomes more prominent as the distance between opposing shear forces decreases with decreased length of bar. The reason why transverse shear stress is not taken into account in Euler - Bernoulli beams is bending is assumed to behave in such a way that cross section normal to the neutral axis remain normal to the neutral axis after bending. All in all, this means the predictions would be more accurate for bars tuned at lower frequencies because these bars would be longer.

4.5 Task 4

For this task we consider a beam of length L. For the boundary conditions, the beam is free at the left end, meaning that both the moment and the shear force at the left end (x=0) are zero:

$$\begin{aligned} M(x) &= \beta(x)\underbrace{u_{xx}(x)}_{0} = 0 \\ V(x) &= -(\beta(x)u_{xx}(x))_x = -\beta\underbrace{u_{xxx}}_{0} = 0 \\ &\text{on} \quad \Omega_1 \quad \text{(x=0)} \end{aligned} \tag{230}$$

Where we assumed that our task 4 beam is of uniform material and cross sectional area, hence:

$$\beta(x) = EI_{eff}(x) = \int_{\mathcal{D}} E(x,y,z)y^2\,dA = \text{constant.} \tag{231}$$

And hence we could take $\beta(x)$ out of the derivative for the shear force expression in equation 230. Even if this was not the case (changing cross sectional area as a result of beam carving, for instance, or change in material), then the shear force would still be zero according to our problem boundary inputs, :

$$\begin{aligned} V(x) = -(\beta(x)u_{xx}(x))_x = -\beta_x(x)\underbrace{u_{xx}(x)}_{0} - \beta(x)\underbrace{u_{xxx}}_{0} = 0 \\ \text{on} \quad \Omega_1 \quad \text{(x=0)} \end{aligned} \tag{232}$$

On the right end (Ω_2 at x=L) we add to our beam a lumped (massless) Hookean spring, such that the vertical force from the spring enforces, or imposes a shear force on the right end. The moment at the right end is also zero (moment free):

$$\begin{aligned} M(x) &= \beta(x)u_{xx} = 0 \\ V(x) &= -(\beta(x)u_{xx}(x))_x = -k_s u(x) \\ &\text{on} \quad \Omega_2 \quad \text{(x=L)} \end{aligned} \tag{233}$$

We can see that all imposed boundary conditions, including the shear force at the right end due to the spring, are natural (imposed on the 'secondary' variables, or higher derivatives like shear forces and moments and will automatically be satisfied after solution of the problem given sufficient accuracy of the FE method), as opposed to essential (imposed on primary variables which are the deflection/displacement and its first derivative).

As a result of our energy minimization proposition, we are solving the equation for α^{RR}, which, for the finite element method, corresponds to $\underline{u}_h$, the displacement at the nodes:

$$\underline{A}\,\underline{\alpha}^{RR} = \underline{F} \tag{234}$$

To construct each of of the $\underline{A}$ and $\underline{F}$, we first construct $\underline{A}^N$ and $\underline{F}^N$ from direct stiffness summation.

$$\begin{aligned} A_{ij}^N &= \int_0^L \beta(x) \frac{d^2\varphi_i}{dx^2} \frac{d^2\varphi_j}{dx^2}\, dx \\ F_i^N &= \int_0^L q(x)\varphi_i\, dx \end{aligned} \tag{235}$$

The direct stiffness summation matrix and vector do not have the natural boundary conditions incorporated. We construct $\tilde{A}$ and $\tilde{F}$ to incorporate the boundary conditions by adding the appropriate factors arising from the natural boundary conditions to the appropriate entries in the matrix and vector, respectively.

$$\begin{aligned} \tilde{A}_{ij} &= \underbrace{\int_0^L \beta(x) \frac{d^2\varphi_i}{dx^2} \frac{d^2\varphi_j}{dx^2}\, dx}_{A_{ij}^N} + k_s\varphi_i(L)\varphi_j(L) \\ \tilde{F}_i &= \underbrace{\int_0^L q(x)\varphi_i\, dx}_{F_{ij}^N} - M^L \frac{d\varphi_i}{dx}(L) - V^L\varphi(L) \end{aligned} \tag{236}$$

Since we have no Dirichlet boundary conditions, we do not need to remove any imposed essential boundary conditions from $\tilde{A}$ and $\tilde{F}$ before we solve the equation arising from the energy minimization proposition. In other words, $\tilde{A}$ and $\tilde{F}$ after incorporation of the natural boundary conditions are indeed matrix $\underline{A}$ and vector $\underline{F}$ in equation 234 that are used to solve for α^{RR}, or $\underline{u}_h$, the approximation for displacement at the nodes, since we are using FE method (which also happens to be the exact solution for the displacement due to nodal superconvergence assuming no quadrature errors and uniform $\beta(x)$) .

Given our FE formulation (the piece-wise base functions), the additional term for the matrix entries in equation 236 would only affect one entry value in the matrix. We use two pieces of information in our finite difference formulation: first, each of the cubic base functions corresponds to the approximation of the function value itself, $\Phi_{i,1}$, has a value of 1 at its corresponding node, and zero at the other nodes:

$$\Phi_{node,1}(x^i) = \begin{cases} 1 & node = i \\ 0 & node \neq i \end{cases} \tag{237}$$

Second, the other set of cubic base functions whose derivatives at the nodes approximate our solution derivative values at the nodes, $\Phi_{i,2}$ has a value of zero at all nodes:

$$\Phi_{node,2}(x^i) = 0 \quad \text{for all nodes} \tag{238}$$

This means that the only node whose base function contributes to $\varphi(L)$ is the node on the rightmost end, node* and in particular only the first of our cubic functions associated with the node, $\Phi_{node*,1}$ has a non-zero value of 1, while the second cubic function associated with the node, $\Phi_{node*,2}$, is zero.

In our FE mapping we combine both sets of base functions (the ones corresponding the actual value $\Phi_{node,1}$, and the ones whose derivative corresponds the the approximation derivative, $\Phi_{node,2}$), to come up with a single set of functions of $\varphi(x)$. The function that arises from the weighted sum of these new base functions (incorporating information from both the approximation value and the approximation derivative over the domain) is the one that gives the minimal energy functional:

$$u_h(x) = \sum_{i=1}^{n_{node}} \sum_{l=1}^{2} u_{h\ i,l} \Phi_{i,l}(x) = \sum_{j=1}^{2n_{node}} u_{h\ j} \varphi_j(x) \tag{239}$$

This means that the only function that contributes to $\varphi(L)$ would be the first of the two φ functions at the rightmost node. The node at the rightmost end, node*, is given by $\text{num\_el}_0 + 1$, where num_el_0 is the original number of elements before meshing. With meshing, any extra nodes introduced are sequentially numbered according to the order of their introduction such that the nodes before the meshing continue to have their same node ids. This means that even with meshing, the last node continues to have an id of $\text{num\_el}_0 + 1$. Since now we have two base functions for each node, and are interested in the first of the two base functions corresponding to the rightmost node, the id of the non-zero base function of interest (where $\varphi(L) \neq 0$) is given by:

$$2 \times (num\_el_0 + 1) - 1 = 2 \times (num\_el_0) + 1 \tag{240}$$

Combining all this together we get:

$$\varphi_i(L)\varphi_j(L) = \begin{cases} 1 & i = j = 2 \times (num\_el_0) + 1 \\ 0 & \text{otherwise} \end{cases} \tag{241}$$

Hence, what we would do is that after the construction of the direct stiffness summation matrix, A^N, using the direct stiffness summation scheme, we add the natural boundary condition corresponding to the spring effect to the appropriate entry in the matrix to get $\tilde{A}$ which is the same as our matrix $\underline{A}$ in this case:

```
1 A(2*n_el0 +1, 2*n_el0 +1) = A(2*n_el0 +1, 2*n_el0 +1) + ks;
```

And now we use this new matrix A, along with vector F with the natural boundary conditions incorporated, to solve for u_h according to equation 234.

4.6 Task 5

Now we consider a whole xylophone instrument, with N_{bar} bars, which share a common set of two strings for support. The vibration of the string, I assume, would slightly affect the mode shapes for each bar, given the fact that the bars are now coupled via the strings inserting at the holes.

Consider an eigenproblem for all the bars simultaneously. The vector of unknowns would be constructed as the unknowns for the first bar followed by the unknowns for the second bar. . . followed by the unknowns for the Nth bar bar; $2 \times n_{node} \times N_{bar}$ degrees of freedom in total.

We note that there are no direct forces affecting the bar other than at the location of the holes where the vertical spring force could be imposed at the corresponding nodes of the holes. What makes this problem feasible to solve is that the stiffness matrix associated with this eigenvalue problem will still be sparse.

We can consider the vertical spring force applied at the holes to be natural interface conditions that could hence be incorporated into the matrix after direct stiffness summation scheme.

Our matrix A is going to be of size $2 \times n_{node} \times N_{bar}$ by $2 \times n_{node} \times N_{bar}$. For matrix A obtained from the direct stiffness assembly, we do not yet account for the coupling between bars at the holes, hence each bar could be viewed

more or less as an independent system, and there is no 'overlap' in a base function in one bar with the base function of another bar. This means that:

$$\left(\frac{d^2\varphi_i}{dx^2}\right)_{\text{bar a}}\left(\frac{d^2\varphi_j}{dx^2}\right)_{\text{bar b}} = \left(\frac{d\varphi_i}{dx}_{\text{bar a}}\frac{d\varphi_j}{dx}_{\text{bar b}}\right)^2 = 0 \tag{242}$$

For a $\neq$ b. Hence the entries in the matrix attained from the direct stiffness matrix are going to be zero for the nodes that lie on different bars:

$$A_{ij}^N = \int_0^L \beta(x)\frac{d^2\varphi_i}{dx^2}\frac{d^2\varphi_j}{dx^2}\,dx = 0. \tag{243}$$

For nodes i,j lying on different bars. Given the order of which the vector of unknowns was constructed, for each bar we first have the unknowns arranged in a similar fashion as if our system is composed of the bar alone, after which we have a similar vector of unknowns for the following bar concatenated underneath and so on. For each bar, we know that the corresponding matrix is a hexadiagonal matrix (a base function overlapping with a maximum of 6 base functions including itself). Putting all these pieces of information together, our big matrix obtained from the direct stiffness summation is going to be composed of independent (not yet couple) series of hexadiagonal matices, arranged along the diagonal of the big matrix, as can be seen in figure 30.

Now we apply our natural interface conditions to our matrix to get matrix $\tilde{A}_{ij}$ (which is the same as matrix $\underline{A}$ in case of only natural boundary conditions). The separate blocks are now 'coupled' due to the vertical spring force (that depends on the displacements of the nodes at the holes) acting at the holes.

Hence, a bar with hole 1 at node of let's say i, node i is now is coupled with node j of the adjacent bar where hole 1 is present. This means that the entries into our matrix (i,j) and (j,i) are no longer zero due to the vertical spring force at the hole locations. The same is true for hole 2. Also note that for the bars at the beginning and end of xylophone, there is only one (as opposed to two) adjacent bars and hence each node corresponding to the hole location would be coupled to one (as opposed to two) other nodes on other bars where the holes are. Finally, we note that given the contribution of the hole node itself to the spring force according to the spring force equation, we also need to modify the diagonal entry in the block for each bar at the hole node: Combining all this together, our new matrix with the natural interface conditions incorporated would look like something similar to what is in figure 31

I assume the effect of vibration will slightly affect the shape of the modes, especially that now the displacement/deflection at the hole nodes is no longer zero. Furthermore, now that the bars are coupled via a string, a bar may 'pick up' the vibration and resonate at one of its natural frequencies when other bars are sounded (sympathetic resonance). For instance, perhaps hitting a bar tuned for a particular fundamental frequency could now drive a harmonic mode of a higher amplitude adjacent bar, because both could share an overtone.

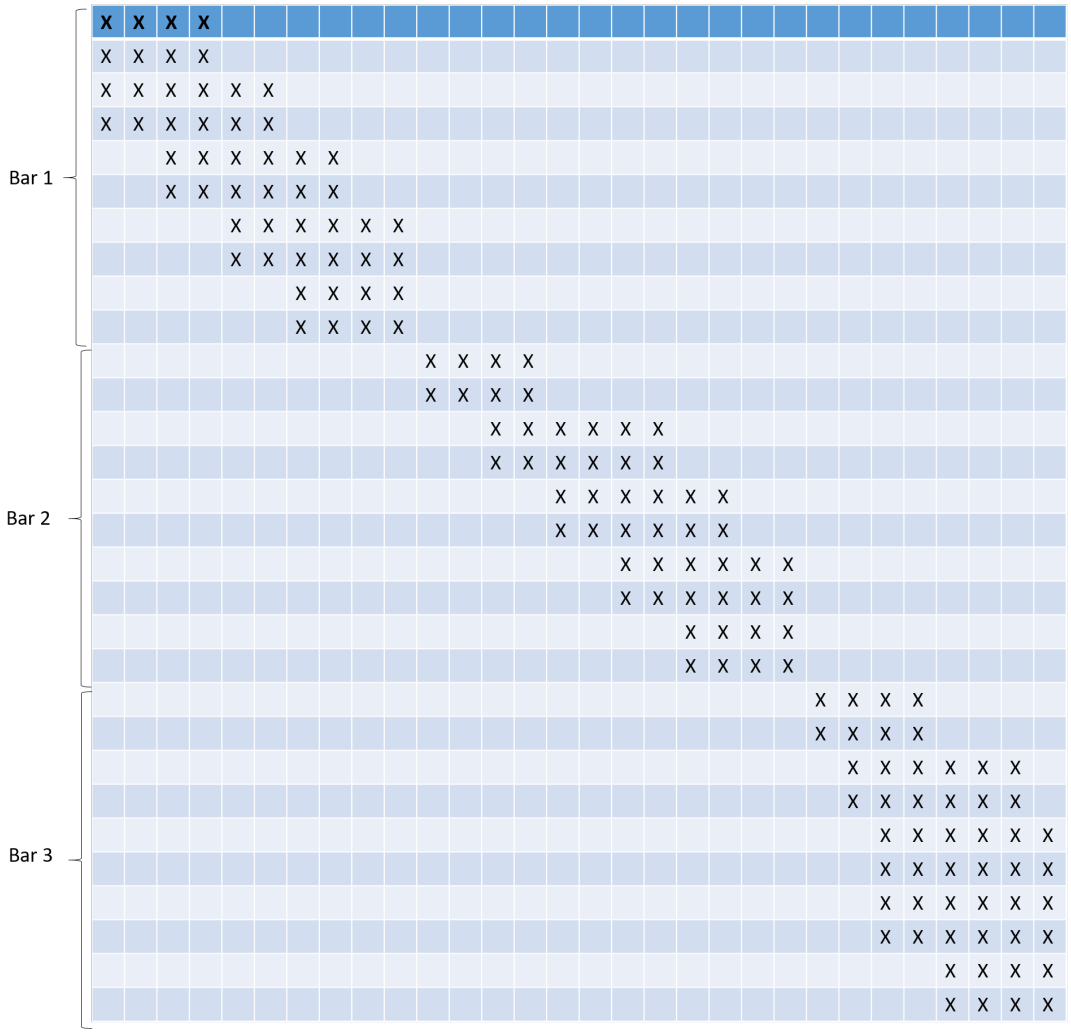


Figure 30: Task 5: Matrix A^N obtained from direct stiffness summation for multiple bars. X represents non-zero values in the matrix. Here $n_{node} = 5$ and N_{bar}=3 hence the matrix is of size 30 by 30. As it can be seen, the matrix is sparse. The minimum number of entries on any row is 4. For an element with a node on the boundary (x = 0, or x = 1) each basis function overlaps with the other three basis functions that are non-zero in that element. (This then preserves symmetry even at the "edges".)

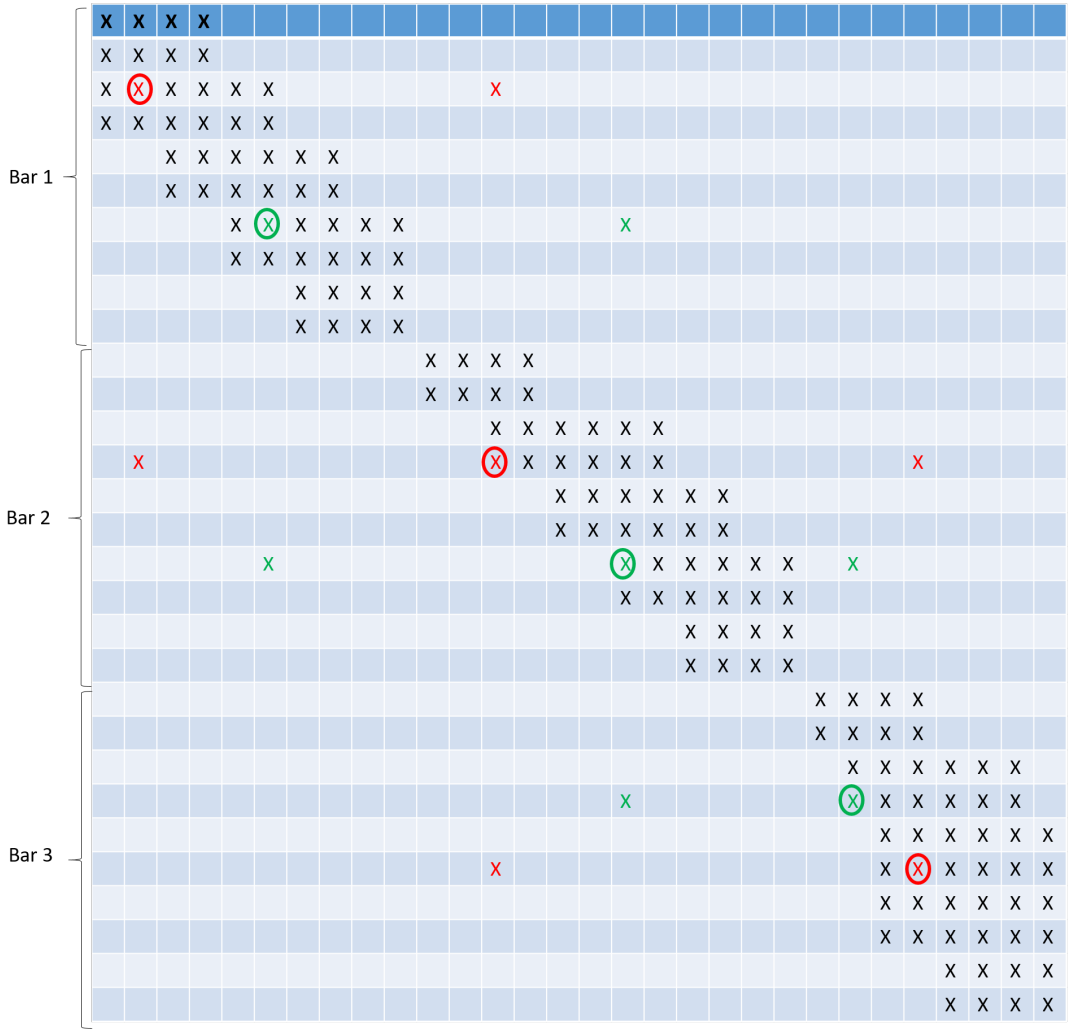


Figure 31: Task 5: Matrix $\tilde{A}_{ij}$ obtained from direct stiffness summation for multiple bars. X represents non-zero values in the matrix. Here $n_{node} = 5$ (for each bar for simplicity) and N_{bar}=3 hence the matrix is of size 30 by 30. Red circles correspond to nodes where hole 1 is located, and green circles represent nodes where hole 2 is located. The colored X represents affected nodal entries as a result of the coupling between hole nodes due to the vertical spring force natural interface conditions. As it can be seen, the matrix is still sparse.

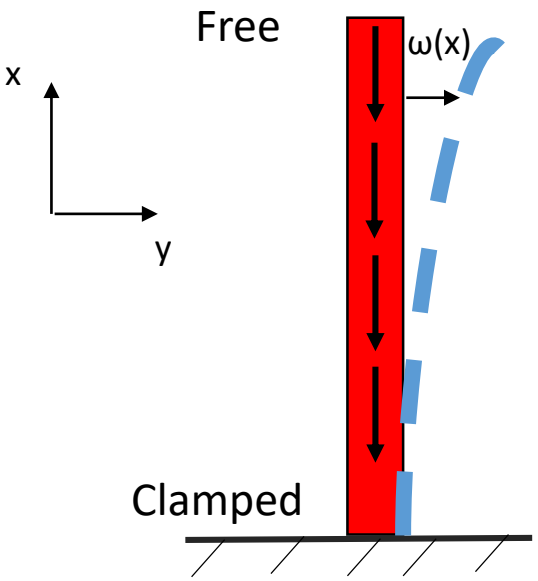
.

Chapter 5: The FE Method for 1D 4th-Order BVPs (Bending): Self-Buckling

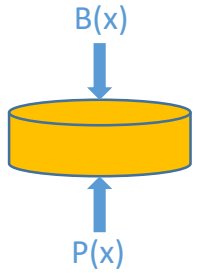
Ali Daher

Self-buckling

A column can buckle due to its own weight with no other direct forces acting on it, in a failure mode called **self-buckling**.



B(x): Section supports the weight of all material above it.



Equilibrium in x direction: $\Sigma F_x = 0$

$$P(x) = B(x) = \int_x^{L_d} \pi \rho g R_d^2(x'_d)\, dx'_d > 0$$

Apply Balance of Moments to arrive to BVP:

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

$$0 < x < L$$

With the following boundary conditions:

$$w_d(0) = w_{dx_d}(0) = 0 \quad \text{free}$$

$$M(L) = (E_d I_d w_{d_{x_d x_d}}(L) = 0$$

$$V(L) = (E_d I_d w_{dx_d x_d})(L) = 0 \quad \text{clamped}$$

Non-dimensionalize to get:

$$\frac{d^2}{dx^2}\left(R^4(x)\frac{d^2 w}{dx^2}\right) + \gamma \frac{d}{dx}\left(P(x)\frac{dw}{dx}\right) = q(x) \quad 0 < x < 1$$

$$w(0) = w_x(0) = 0$$

$$w_{xx}(1) = (R^4 w_{xx})_x(1) = 0$$

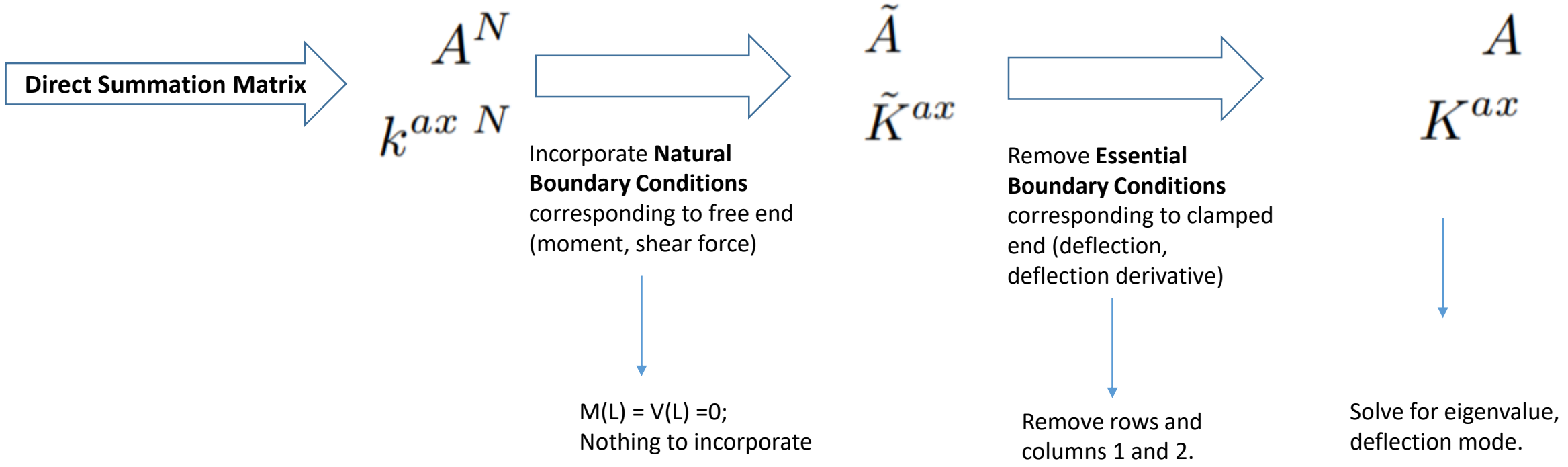
Self-Buckling Eigenproblem

- Eigenvalues give non-trivial solutions for the homogeneous BVP (or, say q(x) odd about x = $\frac{1}{2}$).

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

  - The eigenvalues ($\gamma^{(n)}$) correspond to the (non-dimensional) loads P.
  - Define the first eigenvalue $\gamma^{(1)}$ to be the critical load (P_c) that produces the first buckling mode, with corresponding eigenfunction $u^{(1)}(x)$.
  - For loads (eigenvalues) satisfying the BVP:
    - P < Pc : u(x) is unique
    - P > Pc : u(x) is not unique

FE Method


Direct Summation Matrix
A^N
$k^{ax\ N}$
Incorporate **Natural Boundary Conditions** corresponding to free end (moment, shear force)
M(L) = V(L) =0;
Nothing to incorporate
$\tilde{A}$
$\tilde{K}^{ax}$
Remove **Essential Boundary Conditions** corresponding to clamped end (deflection, deflection derivative)
Remove rows and columns 1 and 2.
A
K^{ax}
Solve for eigenvalue, deflection mode.

Task

- Find radius profile R(x) for a column of circular cross section that maximizes the self-buckling height:

- 1. Find R(x) that maximizes $\gamma^{(1)} = P_c$.
- Length of column would accordingly increase as the critical load corresponding to the first mode increases:

$$L_d^{opt} = \left(\frac{\lambda_c^{opt} E_d V_d}{4\pi\rho g}\right)^{\frac{1}{4}}$$

Cross Sectional Radius Constraint: Minimal Radius R_{min}

- No Cross Section along the length of the structure is allowed to have less than a prescribed radius:

$$\mathrm{R}_d(x_d) \geq R_{min} \times \overline{R_d} \longrightarrow \mathrm{R(x)} \geq R_{min}$$

Radius Gradual Variation Constraint:

- R(x) [and hence G(x)] must be continuous and weakly differentiable:

$$|G'(x)| \leq S_{max}$$
$$|R'(x)| \leq \sqrt{1 + S_{max}}$$

Material Constraint: Constant Volume V_d

- Optimized structure should have the same volume as a cylinder of the same length of radius $\overline{R_d}$:

$$\int_0^L \pi R_d^2(x_d)\, dx_d = \pi \overline{R}_d^2$$

$$\int_0^L R_d^2(x_d)\, dx_d = \overline{R}_d^2 \int_0^1 R^2(x)\, dx$$

$$\overline{R}_d^2 \int_0^1 R^2(x)\, dx = \overline{R}_d^2$$

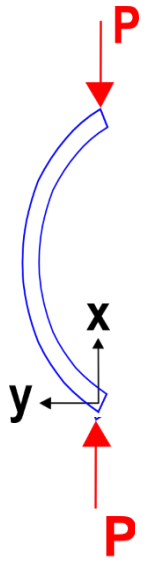
$$\int_0^1 R^2(x)\, dx = 1$$

- Non-dimensionalization:

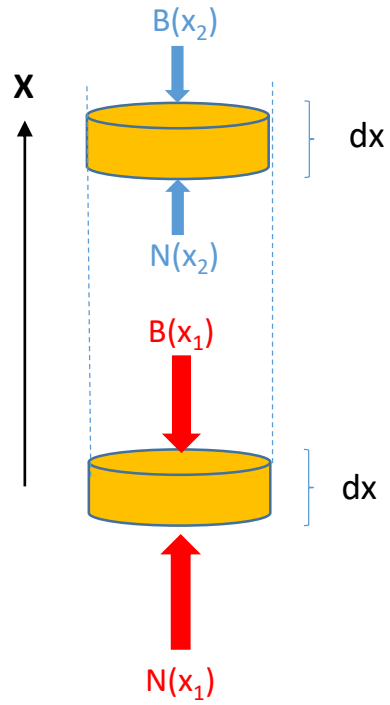
$$\text{let } x_d = \mathrm{L} \times \mathrm{x}$$
$$0 \leq \mathrm{x} \leq 1$$
$$\mathrm{d}x_d = \mathrm{Ldx}$$

$$\text{let } R_d(x_d) = \overline{R}_d\ \mathrm{R(x)}$$

Motivation behind Optimization Choice

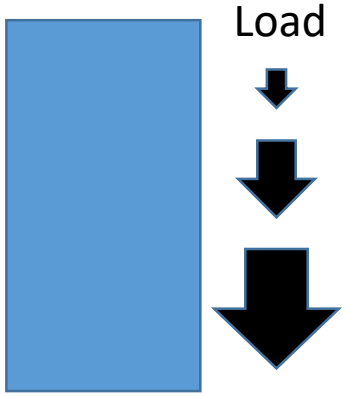


Buckling is the result of compressive stress ($\frac{Force}{Area}$)



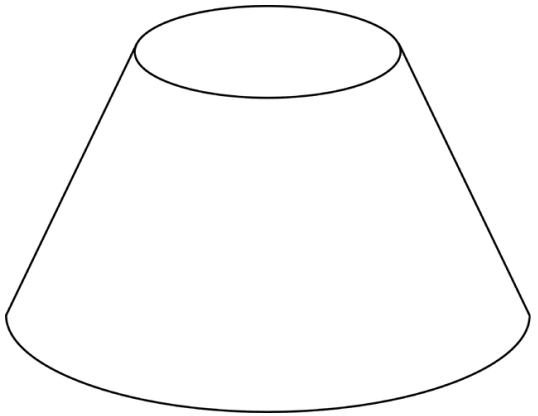
Column tends to buckle due to high load at the bottom

$$|B(x_1)| = |N(x_1)| > |B(x_2)| = |N(x_2)|$$



$$P_d(x_d) = \int_{x_d}^{L_d} \rho g \pi R^2(x'd)\, dx'_d$$
$$P(x) = \int_x^1 R^2(x')\, dx'$$

The normal and weight forces (equal in magnitude) increase as we go down the structure



Choose a radius profile along the length of structure that 'allocates' more material (A_{cs}) to lower regions experiencing higher load to 'even out' the compressive stress:

- $R'(x) < 0$
- $R(1) = R_{min}$

Functional Form for R(x)

$$R(x) = asin(b(x-c)) + d$$

- Four parameters (a,b,c,d) to optimize?

→ Use 'introduced' constraint of R(1) = R_{min} to dispense d.

$$R(x) = asin(b(x-c)) + d$$

$$\mathrm{R}(1) = \mathrm{R}_{\min}$$

$$\mathrm{d} = \mathrm{R}_{\min} - asin(b(1-c))$$

$$\mathrm{R}(\mathrm{x}) = \mathrm{a}\ \sin(\mathrm{b}(\mathrm{x}\text{-}\mathrm{c})) + \underbrace{\mathrm{R}_{\min} - asin(b(1-c))}_{d}$$

→ Use Volume constraint to dispense a.

$$\int_0^1 R^2(x)\,dx = 1$$

$$\mathrm{R}^2(x) = a^2 sin^2(b(x-c)) \quad \text{I}$$

$$+\ 2\mathrm{asin}(\mathrm{b}(\mathrm{x}\text{-}\mathrm{c}))[\mathrm{R}_{min} - asin(b(1-c))] \quad \text{II}$$

$$+[\mathrm{R}_{min} - asin(b(1-c))]^2 \quad \text{III}$$

Evaluation of terms

I

$$\int_0^1 a^2 sin^2(b(x-c))\,dx = a^2 \int_0^1 \left(1 - cos^2(b(x-c))\right) dx$$
$$= a^2 \int_0^1 1\,dx - a^2 \tfrac{1}{2} \int_0^1 [1 + cos(2b(x-c))]\,dx$$
$$= a^2 [\tfrac{1}{2} - \tfrac{1}{4b} sin(2b(x-c))\Big|_{x=0}^{1}$$
$$= a^2 [\tfrac{1}{2} - \tfrac{1}{4b}\left(sin(2b(1-c)) + sin(2bc)\right)]$$

II

$$2a\left(R_{min} - asin(b(1-c))\right) \int_0^1 sin(b(x-c))\,dx$$
$$= \tfrac{2a}{b}\left(asin(b(1-c)) - R_{min}\right) cos(b(x-c))\Big|_{x=0}^{1}$$
$$= \tfrac{2a}{b}\left(cos(b(1-c)) - cos(bc)\right)\left(asin(b(1-c)) - R_{min}\right)$$

III

$$\int_0^1 \left(R_{min} - asin(b(1-c))\right)^2 dx$$
$$= R_{min}^2 - 2R_{min} asin(b(1-c)) + a^2 sin^2(b(1-c))$$

$$\int_0^1 R^2(x)\,dx = 1$$

$$a^2\left(\tfrac{1}{2} - \tfrac{1}{4b}\left(sin(2b(1-c)) + sin(2bc)\right) + \tfrac{2}{b}\left(cos(b(1-c)) - cos(bc)\right) sin(b(1-c)) + sin^2(b(1-c))\right)$$ **A**

$$+a\left(\tfrac{-2R_{min}}{b}\left(cos(b(1-c)) - cos(bc)\right) - 2R_{min} sin(b(1-c))\right)$$ **B**

$$+\left(R_{min}^2 - 1\right) = 0$$ **C**

$$Aa^2 + Ba + C = 0$$

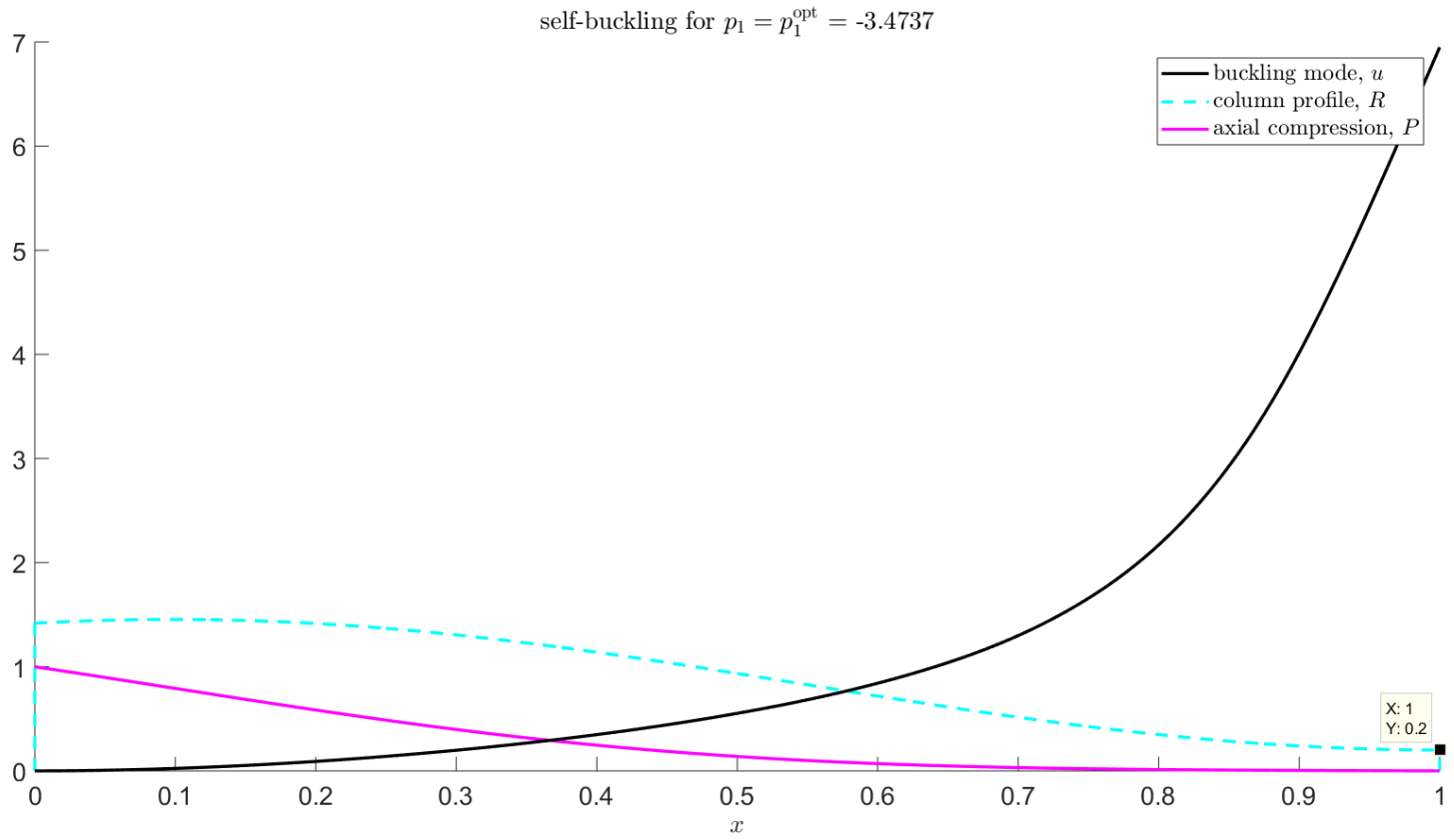
$$a = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

$$d = R_{min} - asin(b(1-c))$$

Optimize over two parameters only:

- **b**
- **c**

A ‘Quick’ Glance at Numerical Implementation



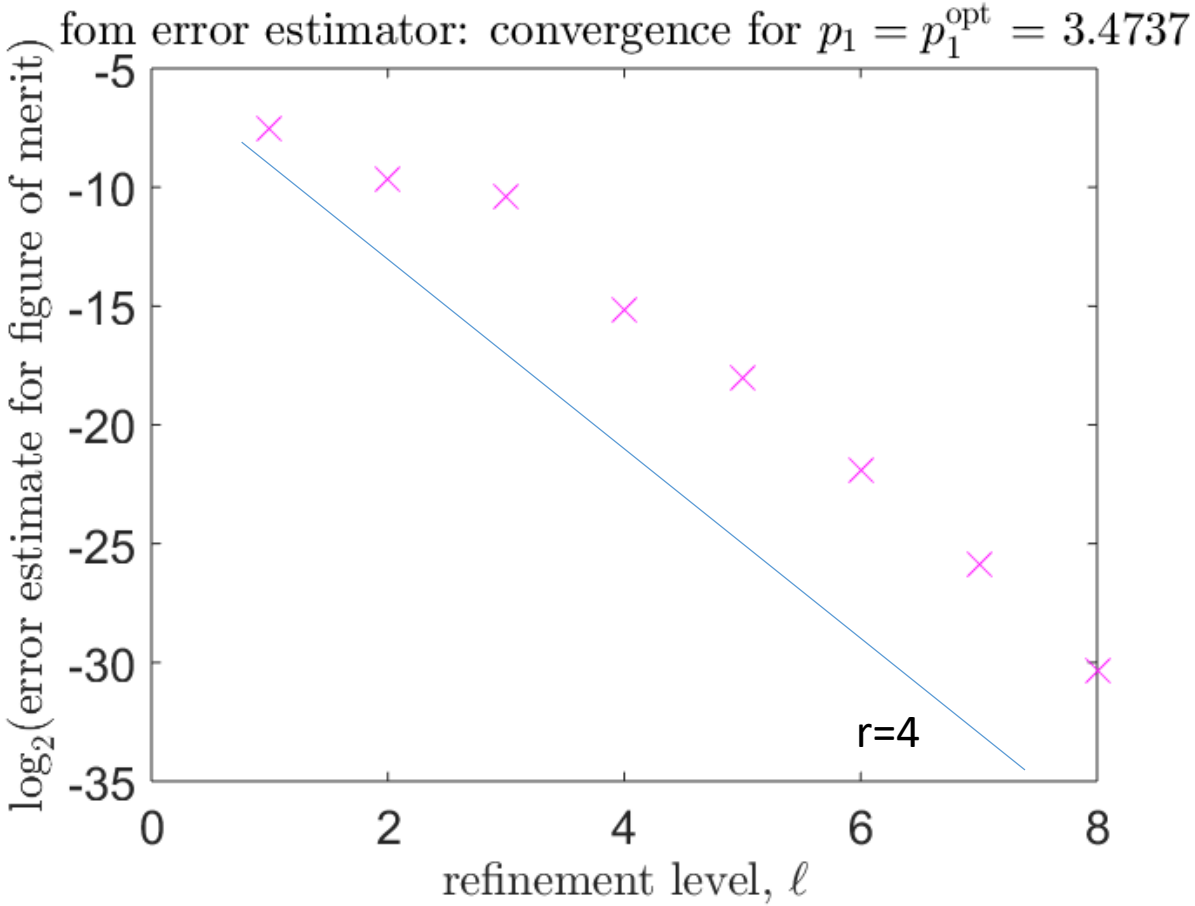
Optimized column radius profile R(x), buckling mode, and axial compression after 3 refinements, starting from an initial mesh consisting of only a single (C1, Hermitian) finite element. Radial profile is given by R(x) = Asin(b(x-c)) + d. b_{opt} = -3.4737 , c_{opt} = 3.2632.

- Column profile looks sinusoidal.
- Axial compression decreases with x faster than a linear decrease (corresponding to a cylinder), meaning cross sectional area is decreasing with x.
- R(1) = R_{min} = 0.2 as desired.

Convergence Rates

Number of Refinements	1	2	3	4	5
Figure of Merit	1.7587	1.7462	1.7370	1.7364	1.7364
Error	0.0125	0.00284	0.00287	9.315×10^{-5}	1.597×10^{-5}

- Choose a mesh that lies in the asymptotic convergence region (with a slope of -4) such that the posteriori error estimates are reliable.
- Choosing 3rd of 4th refinement: FOM ≈ 1.74.



Error estimate Convergence rates for figure of merit (FOM) across different mesh refinements. The error estimate, and hence its reliability, starts converging around 3 or 4 refinements, with the desired convergence slope of -4.

References

[1] Jamielyn Nye. The best ever hamburger recipe - i heart naptime, Nov 2018.

[2] Meathead Goldwyn. The science of hamburger buns.

[3] David Joachim and Andrew Schloss. The science of grilling burgers - how-to, May 2017.

[4] Agency. How to make the perfect burger: Oxford food scientist claims to have answer, Aug 2015.