Applications of Finite Element Methods

2.S976

Mohammed Aljashmi
MIT Mechanical Engineering

May 16, 2019

Abstract

Finite Element methods is a numerical tool that allows us to solve engineering and physical science problems. The method relies on the principles of energy functional minimization, an area of the calculus of variations. Finite element methods have gained popularity due to their robustness and ability to accurately represent complex geometry and utilize different materials. It relies on approximating the solution using a combination of trial functions. The process is computationally efficient, as the partial derivatives vanish outside the domain of the element. In this paper, we investigate different applications of the finite element method. First, we discuss the Rayleigh-Ritz method where we investigate steady heat transfer applications with a two different types of boundary conditions. Next, we implement the finite element method to solve time dependent partial differential equation as we explore the process of flipping burgers from a heat transfer and optimization perspectives. Finally, we implement the finite element solver for a fourth order boundary value problem with its application in beam bending and structural mechanics in general.

Chapter 1

1 The Rayleigh-Ritz Method

The Rayleigh- Ritz method is a numerical method to approximate solutions to eigenvalue problems that are difficult to solve analytically. It uses the a governing differential equation with suitable boundary conditions to derive the basis function coefficients. Here we discuss and analyze two different types of boundary conditions. Neumann-Robin (NR) and Dirichlet (D) boundary conditions. Neumann-Robin condition represents an imposed constant flux at the boundary, whereas Dirichlet boundary condition represents a constant temperature constraint at the boundary.

We can use the minimization of the energy functional Π to find the basis function coefficients. Any perturbations to the energy functional will increase and deviate from the local minimum value. Since the problems we are discussing have boundary conditions, such as fixed temperature

or flux, the basis functions are chosen to satisfy the boundary conditions and all other basis functions vanish at the boundary.
The energy functional Π minimization for a Dirichlet boundary value problem:

Model I:

$$\prod(w) = \int_0^L K(x)\left[\left(\frac{dw}{dx}\right)^2 + \mu(x)w^2\right]dx + \frac{1}{2}(\,\gamma_2 w(L)^2) - \int_0^L f(x)_\Omega\, w\, dx - w(0)f_{\gamma 1} - \; w(L)f_{\gamma 2}\; dx$$

$$-K\frac{d}{dx}\left(\pi R_0^2\left(1 + \;\beta\frac{x}{L}\right)^2\frac{du}{dx}\right) = 0$$

$$K\frac{du}{dx} = \; -q_1 \text{ on } \Omega$$

$$-K\frac{du}{dx} = \; \eta_2(u - \; u_\propto) \text{ on Gamma (2)}$$

$$\tilde{A}_{ij} = \int_0^L K\pi\mathcal{R}^2\left(1 + \;\beta\frac{x}{L}\right)^2\frac{du}{dx}\frac{dv}{dx}dx + \;\eta_2(u - \; u_\propto)(1 + \;\beta)\pi\mathcal{R}^2 u(L)v(L)$$

$$\tilde{F}_i = \int_0^L q_1\pi\mathcal{R}^2 u(0) + \;\eta_2 u_\propto(1 + \;\beta)\pi\mathcal{R}^2 u(L)$$

$$\underline{\propto^{RR}} = \underline{A}^{-1}(\underline{F})$$

$$u^{RR}(x) = \sum_{i=1}^{n^{RR}} \propto_i^{RR}\;\psi_i(x)$$

The energy functional Π minimization for a Neumann-Robin boundary value problem:

Model II:

$$\prod(w) = \int_0^L K(x)\left[\left(\frac{dw}{dx}\right)^2 + \mu(x)w^2\right]dx + \frac{1}{2}\left((\gamma_1)w(0)^2 + \;\gamma_2 w(L)^2\right) - \int_0^L f(x)_\Omega\, w dx - w(0)f_{\gamma 1} - \; w(L)f_{\gamma 2}\, dx$$

$$-KA_{cs}\frac{d^2u}{dx^2} + \eta_3 P_{cs}u - \eta_3 P_{cs}u_\infty = 0 \;\; on\ \Omega\ (1)$$

$$u = u_{gamma1}\ on\ \mathit{\Gamma}_1\ (2)$$

$$-K\frac{du}{dx} = 0 \quad on\ \mathit{\Gamma}_2\ (3)$$

$$A_{ij} = \int_0^L KA_{cs}\frac{du}{dx}\frac{dv}{dx} + \eta_3 P_{cs}uv\ dx$$

$$F_i = \int_0^L \eta_3 P_{cs}u_\infty u\, dx$$

$$u^{RR}(x) = u_{Gamma1}\psi_0(0) + \sum_{i=1}^{n^{RR}} \propto_i^{RR}\ \psi_i(x)$$

$$\propto_0^{RR} = u_{Gamma1}\psi_0(0) = u_{Gamma1}$$

$$\underline{A}\ \underline{\propto^{RR}} = \underline{F} - \underline{u_{Gamma1}b}$$

$$\underline{\propto^{RR}} = \underline{A}^{-1}\left(\underline{F} - \underline{u_{Gamma1}b}\right)$$

$$\underline{\widetilde{\propto^{RR}}} = \left[\propto_0^{RR}\ \underline{\propto^{RR}}\ \right]$$

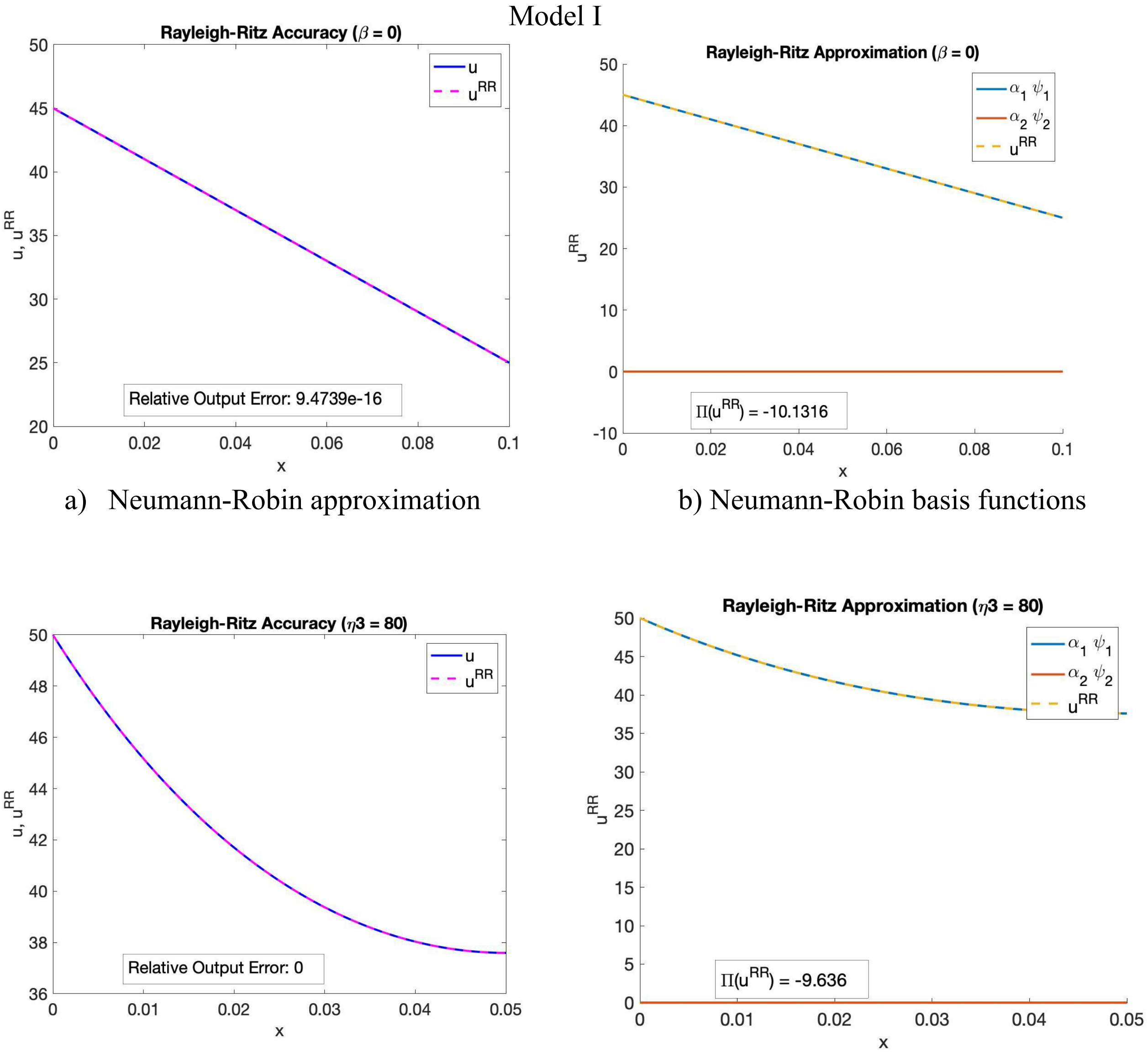
$$u^{RR}(x) = \sum_{i=1}^{n^{RR}} \widetilde{\propto_i^{RR}}\ \psi_i(x)$$

By solving for α^{RR}, the Rayleigh Ritz approximations can accurately determine the solution of the differential equation. The Rayleigh-Ritz formulation differs for the different types of boundary condition problems; however, it always consists of the summation of the product of the basis functions and its Rayleigh-Ritz coefficients.

2 Results and discussion

2.1 Using the exact solution as a base function to verify results

To verify the functionality of the MATLAB code, the exact solution is used as base function. A functioning code would yield results with error that is equivalent to the round off errors to the limited computer precision. It can be seen in the figure (1) and figure (2) for Model I and Model II respectively that $\propto_1 = 1$, $\propto_2 = 0$. This is expected from a correctly functioning code as the Rayleigh-Ritz method chooses to a coefficient of 1 of the exact solution and a coefficient of 0 for the constant base function. Additionally, the energy functional π for the exact solution is lower or equal to that yielded by ‘constlinquad’.



a) Neumann-Robin approximation

b) Neumann-Robin basis functions

c) Dirichlet approximation

d) Dirchilet basis functions

Figure 1: Rayleigh-Ritz approximation providing zero error for model I and II when the the exact function is included

2.2 The effect of increasing the number of basis functions (n^{RR}) on the accuracy of the results

Model II

By increasing the number of base functions, Rayleigh- Ritz method has more options to choose from to fit into the curve by minimizing the enrgy functional. It can be seen that the error decreases as we add more base functions into the set. Furthermore, The following graphs were generated using ‘constlinquad’ which does not use the exact solution as a base function.

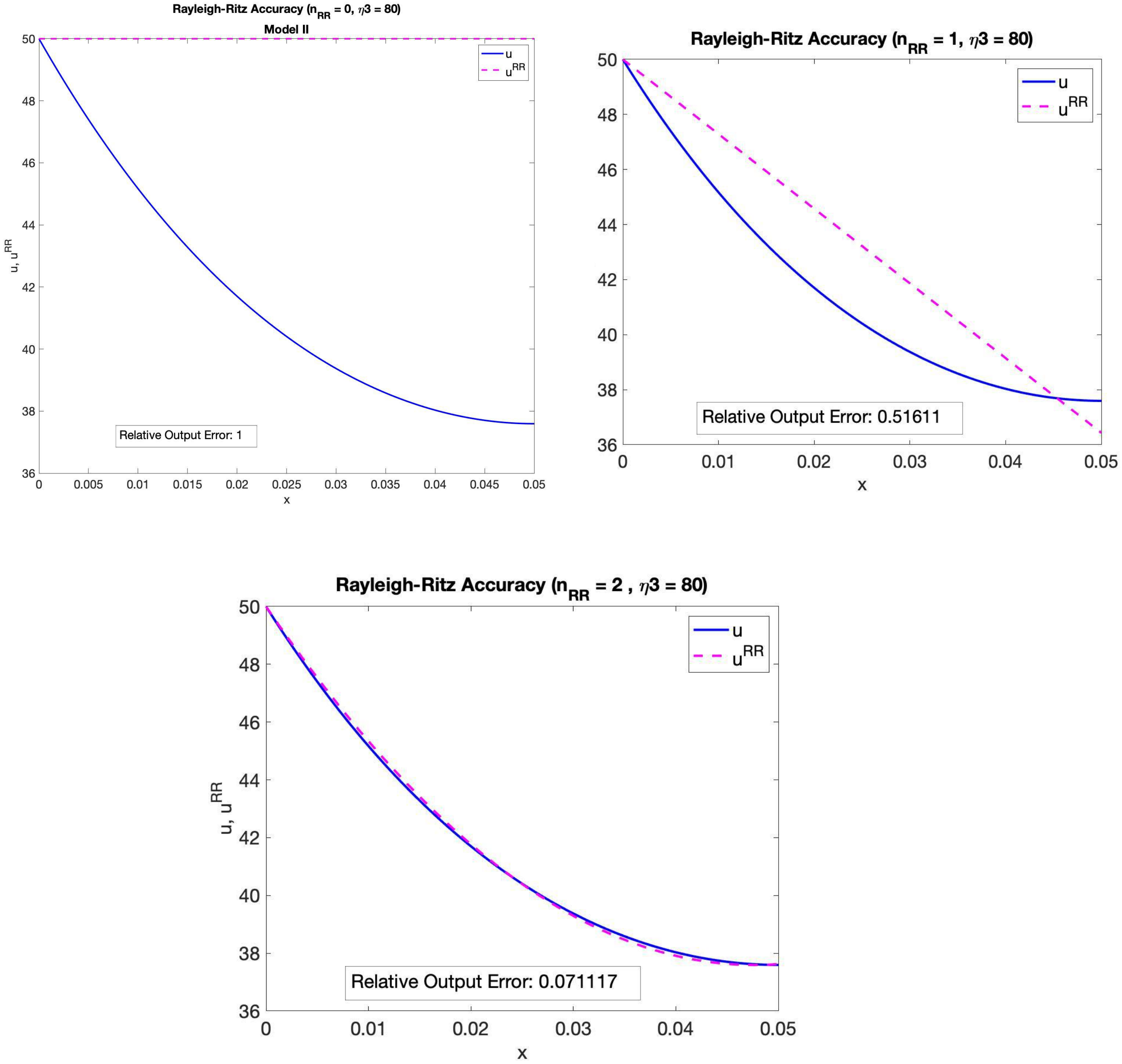
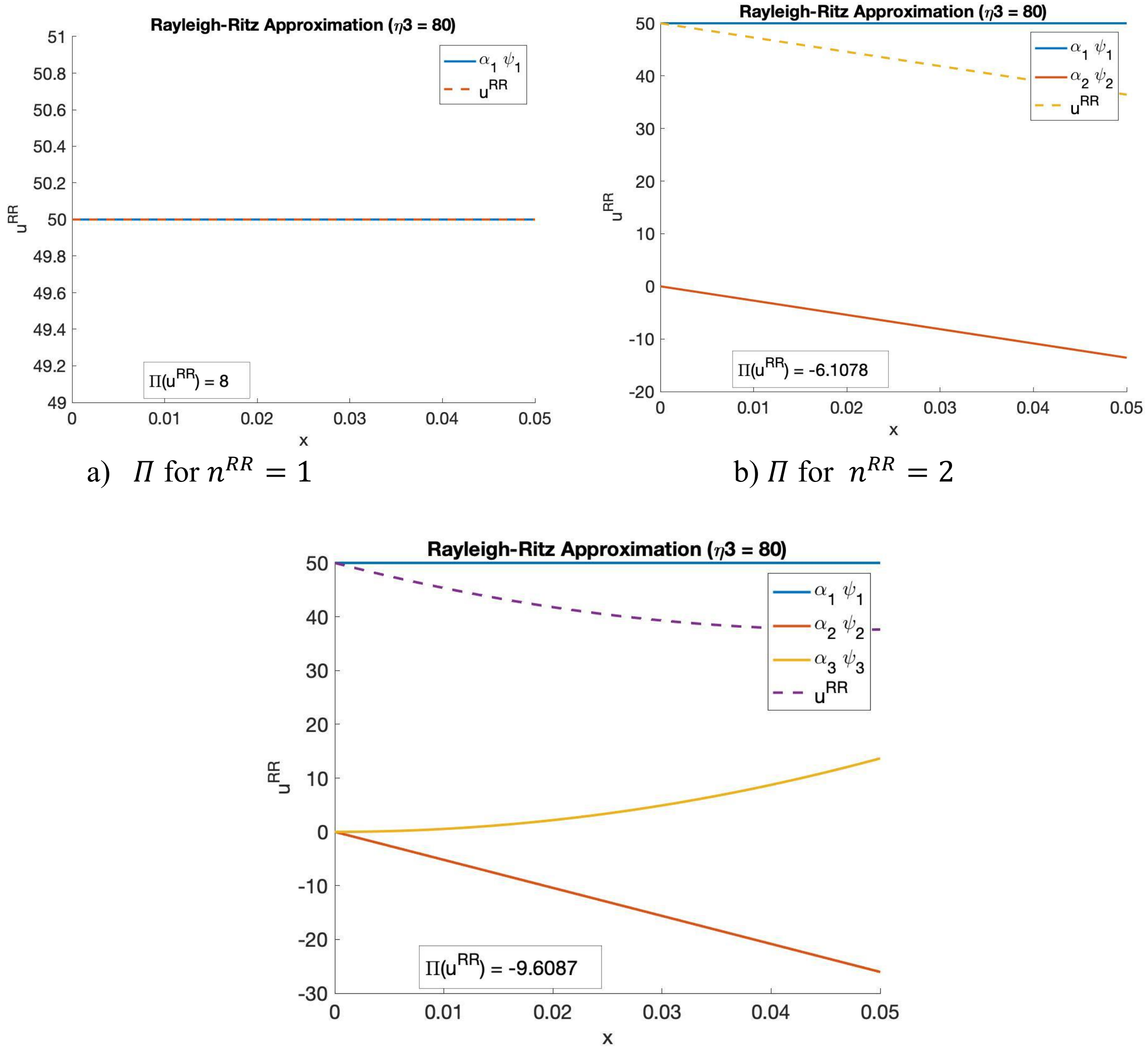


Figure 2: Rayleigh-Ritz approximations and error plots for different number of basis functions

Next, we observe the change in our Π energy functional as we increase the number of basis functions.



a) Π for $n^{RR} = 1$

b) Π for $n^{RR} = 2$

c) Π for $n^{RR} = 3$

Figure 4: The effect of changing the number of basis functions on the Π energy functional

As we can see the Rayleigh-Ritz energy functional decreases as we increase the number of base function. This makes sense as it shows that energy is minimized when we have a greater range of basis functions to choose from.

Model I:

The same is true for model I up to adding the linear base function, as adding more base functions does not necessarily improve the accuracy of the Rayleigh- Ritz method. This is due to the fact that the exact solution is linear so no higher polynomial functions are not needed to model the solution. This becomes clear as we observe that the error doesn't change between 1 and 2 base functions.

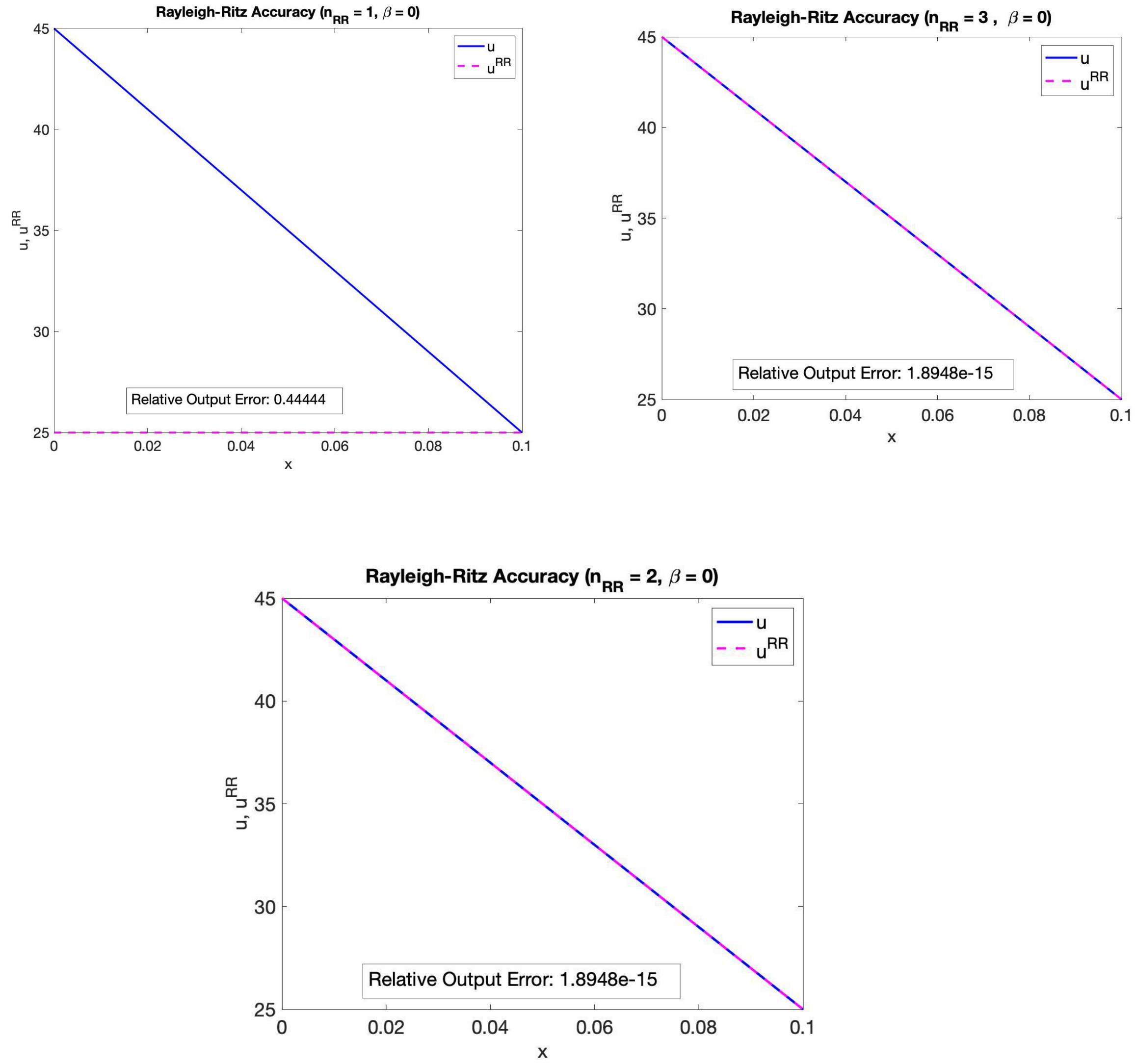
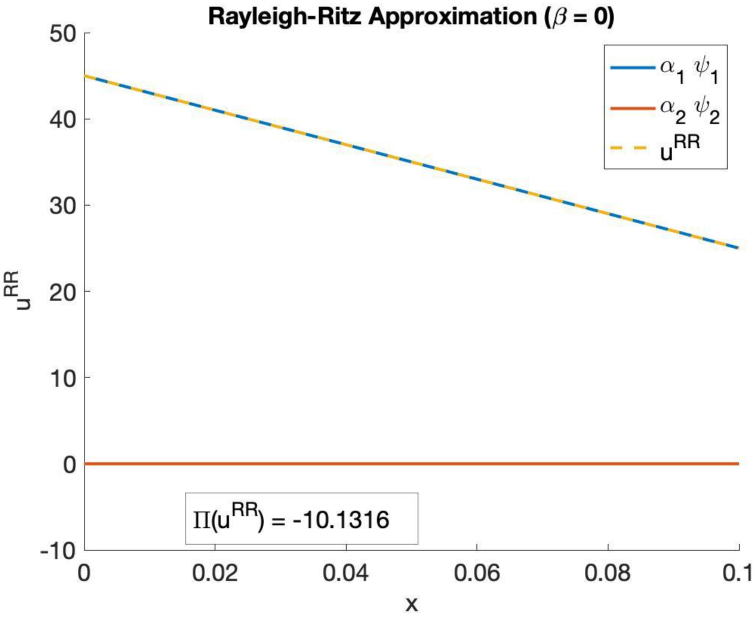


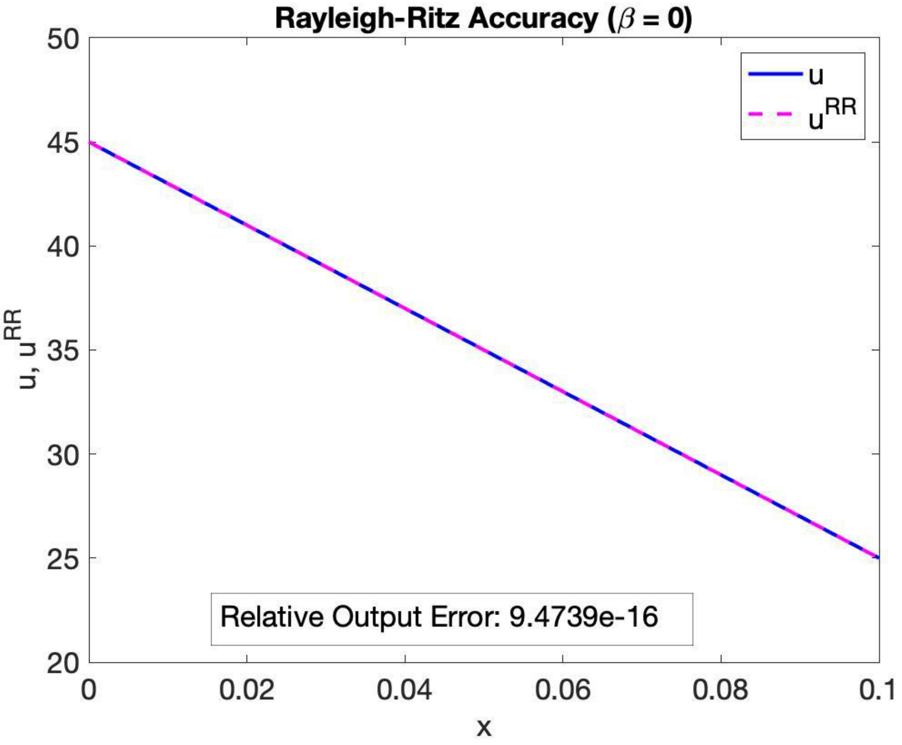
Figure 5: Model I approximated with different number of basis functions

2.3 Comparison ‘exactinlcude’ and ‘constlinquad’

Model I:

Energy Functional minimization





2.4 The effect of Beta in Model I on the accuracy of Rayleigh-Ritz method

As beta increases, the curvature of the slope increases, this makes it harder for the Rayleigh-Ritz base functions to simulate the rapid change in curvature around x = 0.0. This is only true for non-zero positive beta. At beta = 0, the solution is linear and can be solved exactly by Rayleigh-Ritz method since it has a linear base function.

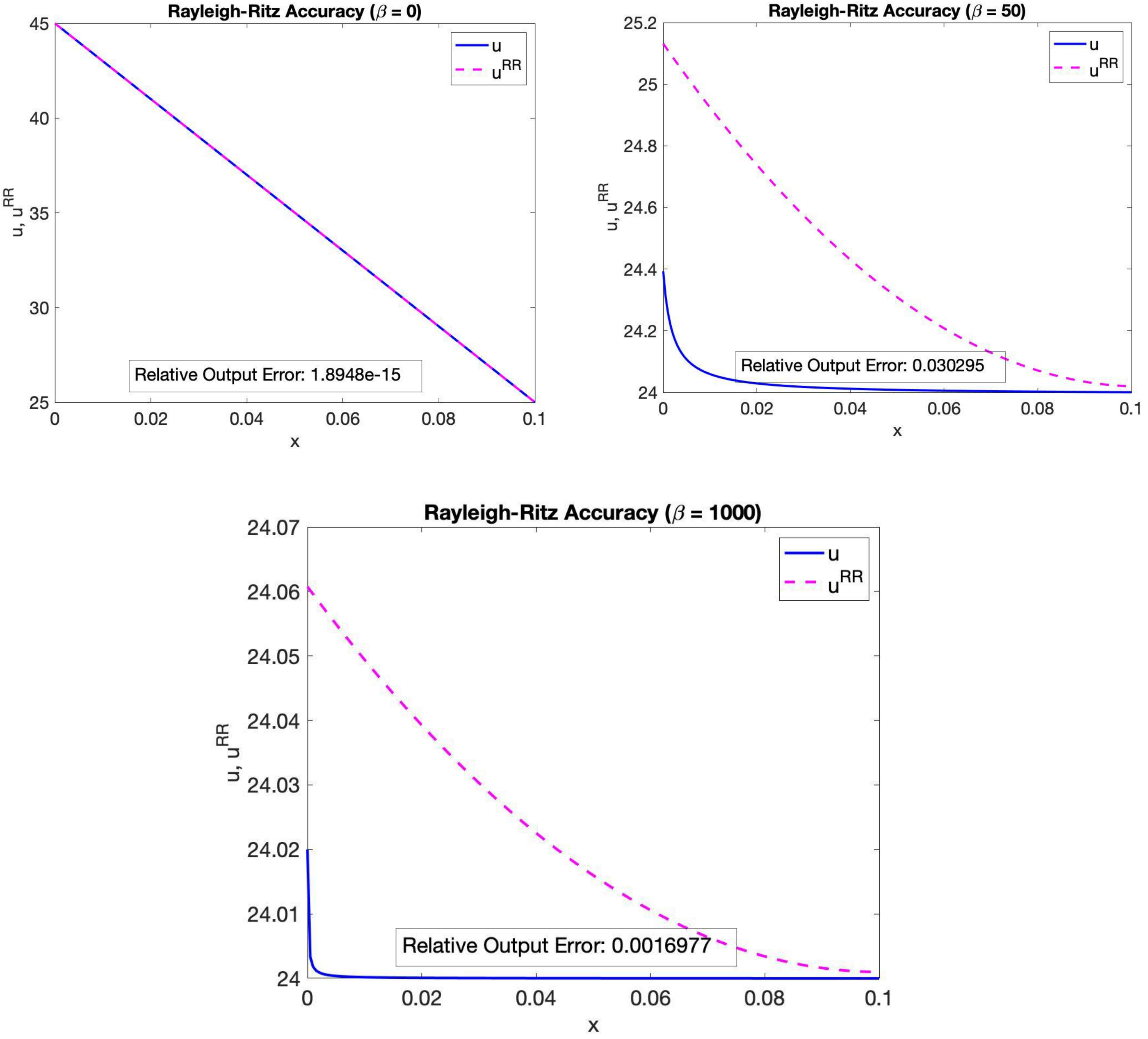
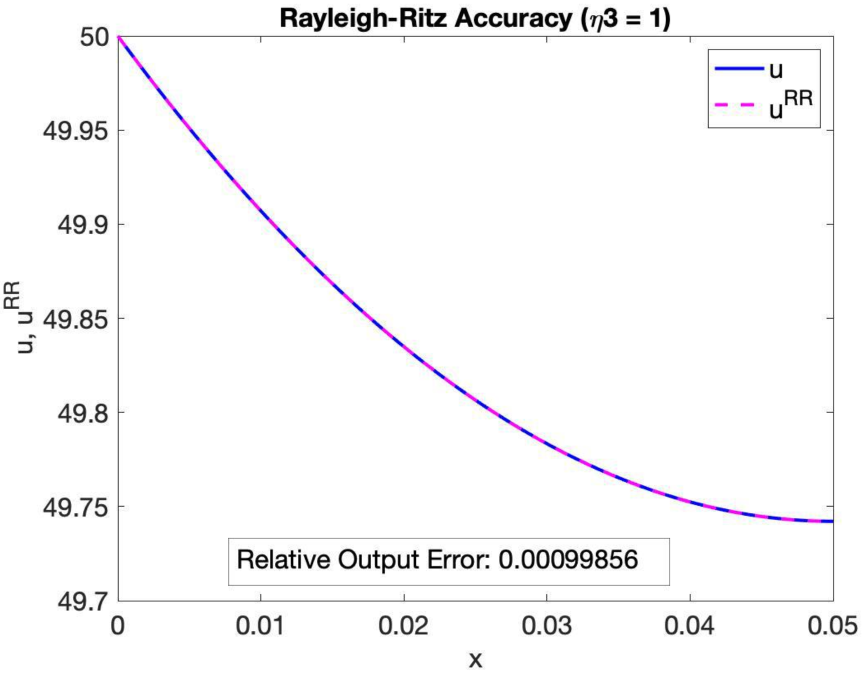


Figure 6: The effect of changing β on the accuracy of the Rayleigh-Ritz approximation

2.5 The effect of μ_0 in Model II on the accuracy of the Rayleigh-Ritz method

μ_0 represents the fin parameter that determines the rate at which the material reaches room temperature along the length of the fin. A high μ_0 indicates a rapid rate of decay, the rapid change in curvature makes it hard for Rayleigh-Ritz to solve using the set of base functions available to the method. The graphs below demonstrate the change in error for increasing η_3.

Since $\mu_0 = \frac{\eta_3 P_{cs} L^2}{K A_{cs}}$, increasing η_3 leads to an increase in μ_0. As η_3 increases from 1 to a 100, the relative output error increases from 0.0009 to 0.086. These eta3 correspond very roughly to different regimes for natural convection/radiation, forced convection, and change-of-phase.



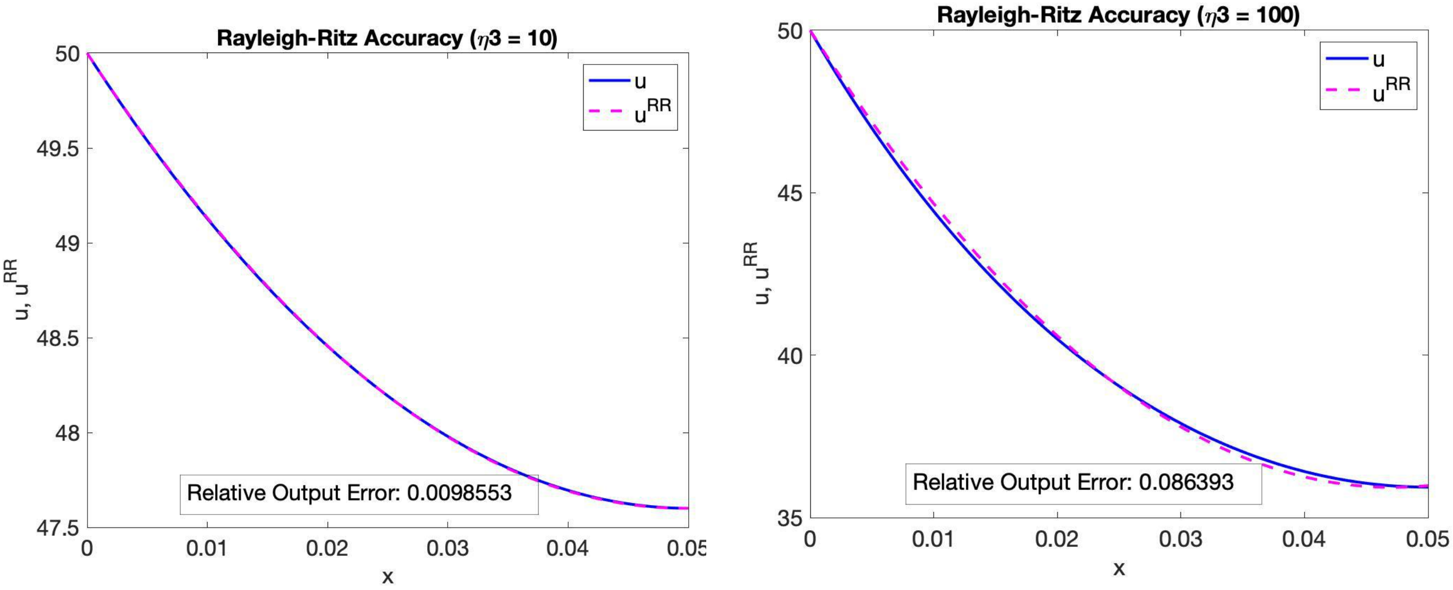


Figure 7: The effect η_3 on the accuracy of the Rayleigh-Ritz apprximation

Chapter 2

Model I corresponds to quasi-1d heat conduction in conical frustum insulated on the lateral surfaces with heat flux and heat transfer coefficient boundary conditions on the left and right surfaces, respectively. The equations and boundary conditions are given by

$$-K\tfrac{d}{dx}\left(\pi R_0^2\left(1+\beta\tfrac{x}{L}\right)^2\tfrac{du}{dx}\right) \qquad = \qquad 0 \text{ in } \Omega$$

$$K\tfrac{du}{dx} \qquad = \qquad -q_1 \text{ on } \Gamma_{\ 1}$$

$$-K\tfrac{du}{dx} \qquad = \qquad \eta_2\left(u-u_\propto\right) \text{ on } \Gamma_{\ 2}$$

where Ω = (0 , L), $\Gamma_{\ 1}$ = { 0} , $\Gamma_{\ 2}$ = { L} , k, R_0 , L, and $\eta_{\ 2}$ are positive constants, β is a non-negative constant and q_1, $u_\propto$ are constants. The exact solution to this problem is given by

$$\mathrm{u} = u_\propto + \tfrac{Lq_1}{k}\left(\tfrac{1+\beta+\frac{k}{\eta 2}}{(1+\beta)^2} - \tfrac{\left(\frac{x}{L}\right)}{1+\beta\left(\frac{x}{L}\right)}\right)$$

Model II corresponds to a right-cylinder thermal fin with temperature and zero-flux boundary conditions on the left and right surfaces, respectively. The equations and boundary conditions are given by

$$-KA_{cs}\frac{d^2u}{dx^2}+\eta_3P_{cs}u-\eta_3P_{cs}u_\infty = 0\ on\Omega$$

$$u \qquad = \qquad u_{\Gamma 1}\ on\ \Gamma_{\ 1}$$

$$-K\tfrac{du}{dx} \qquad = \qquad 0\ \ on\ \ \Gamma_{\ 2}$$

where Ω = (0 , L), $\Gamma_{\ 1}$ = { 0} , $\Gamma_{\ 2}$ = { L} , k, A_{cs} , P_{cs}, and $\eta_{\ 3}$ are positive constants, $u_{\Gamma 2}$ is constant. The exact solution to this problem is given by

$$\mathrm{u} = u_\propto + (\ u_{\Gamma 2} \text{ - } u_\propto\ )\ \frac{cosh\left(\sqrt{\mu 0\left(1-\frac{x}{L}\right)}\right)}{cosh(\sqrt{}(\mu 0)}$$

where $\mu 0 = \frac{\eta_3 PcsL^2}{kA_{cs}}$

Model Mohammed corresponds to a wall isolated laterally and heat flux and heat transfer coefficients on both left and right surfaces.

$$=$$

$$-K\frac{d}{dx}(\pi R_0^2\frac{du}{dx} = 0 \quad in\Omega$$

$$\begin{array}{rcl} K\frac{du}{dx} & = & \eta 1\left(u-u_{out}\right) \text{ on } \Gamma_{\ 1} \\ -K\frac{du}{dx} & = & \eta 2\left(u-u_{in}\right) \text{ on } \Gamma_{\ 2} \end{array}$$

where $\Omega = (0\ ,\ \mathrm{L})$, $\Gamma_{\ 1} = \{\ 0\}\ ,\Gamma_{\ 2}\ = \{\ \mathrm{L}\}$, k, $\eta_{\ 1}$ and $\eta_{\ 2}$ are positive constants. $\eta_{\ 1}$ and $\eta_{\ 2}$ represent the product of the cross-sectional area and heat transfer coefficient of the outside and inside air respectively.

This problem can be solved using a circuit diagram to yield the exact solution given by

$$T\left(x\right) = \left(b-c\right)x+c$$

where $c = \frac{\frac{\eta 2 u_{in}}{\eta 2}+\frac{\eta 2 u_{out} L}{k}}{\frac{\eta 2 L}{k}+1+\frac{\eta 2}{\eta 1}}$

$b = c + \frac{\eta_1 L(c-u_{out})}{k}$

1 Finite Element representation

$$u_h\left(x\right) = \sum_{i=1}^{node} u_{hi}\Phi_i\left(x\right)$$

$$u_{hi}\Phi_i$$

vanishes except for when i = j. This makes the FE scheme computationally efficient as we only care about neighbouring elements.

$$A_{ij} = \int_0^L k\left(x\right)\frac{d\Phi i}{dx}\frac{d\Phi j}{dx} + \mu\left(x\right)\Phi_i\Phi_j dx + \gamma_1\Phi i\left(0\right)\phi j\left(0\right) + \gamma_2\Phi i\left(L\right)\phi j\left(L\right)$$

$$F_i = \int_0^L f_\Omega\left(x\right)\Phi i\left(x\right)dx + f_{\Gamma1}\Phi i\left(0\right) + f_{\Gamma2}\Phi i\left(L\right)$$

For Model I, we have Neuman-Robin boundary condition on the right surface of the frustum and constant flux boundary condition on the left surface as indicated by Γ_2 .

$$A_{ij} = \int_0^L k\left(x\right)\frac{d\Phi i}{dx}\frac{d\Phi j}{dx} + \mu\left(x\right)\Phi_i\Phi_j dx + \gamma_2\Phi i\left(L\right)\phi j\left(L\right)$$

$$F_i = \int_0^L f_\Omega\left(x\right)\Phi i\left(x\right)dx + f_{\Gamma2}\Phi i\left(L\right)$$

$$\mathrm{A}_{ij}{=}A_{ij}\ +\ \gamma_2$$
$$F = F + f_{\Gamma2}$$

Model II
Dirichlet boundary conditions, that is a flux and temperature boundary conditions on the left and right surfaces, respectively.

$$A = \begin{bmatrix} A_{11} & b^T \\ b & \begin{bmatrix} & \cdots & \\ \vdots & A & \vdots \\ & \cdots & \end{bmatrix} \end{bmatrix} \qquad u_h = \begin{bmatrix} u_{\Gamma 1} \\ \vdots \\ u_h \\ \vdots \end{bmatrix} \qquad F = \begin{bmatrix} F_1 \\ \vdots \\ F_{nx1} \\ \vdots \end{bmatrix}$$

The implementation of this method involves the use quadrature points to perform numerical integration.In this chapter, we will discuss two different formulations, p_1 and p_2 cases. In the p_1 case, there are two linear nodal basis functions that form an X. Every nodal function spans 2 elements and reaches a maximum at the center node, while neighbouring functions are at zero. This means that the coefficient of the nodal function is a valid solution at that point. These points are connected to form a piecewise-linear , continuous function. p_2 is similar to p_1 except that there are 3 quadratic nodal base functions rather than 2 linear ones.

2 Results and discussion

As a first check of the functionality of the code, we solve for the PDE representing model II. The FEA solution is plotted along with the exact solution in the same diagram. It can be seen in the first mesh that FE solution does not an adequate job at approximating the solution's behavior. This is due to the lack of sufficient number of elements to simulate the behavior of the function. It can also be seen that the FE method does a worse job at approximating the derivative. By refining the mesh, we are able to produce a much more accurate solution to describe the solution behavior this is indicated by mesh 8 (after 8 mesh size refinements). One way to verify that the code is implemented correctly is by check the order of accuracy of Model II. The log error plot shows a convergence of order 2 which is expected.

It can be seen from model II error plot that the coarsest mesh in which

$$\|u - u_h\| <= 1$$

is mesh 7. The plot indicates an absolute error in the L-infinity norm equivalent to 0.636 however, this is true as long as higher order terms can be neglected. To be conservative, we can apply that error by a safety factor of 2. We can also see the decrease in the H1 norm error, which gives us a higher certainty in the error bounds.

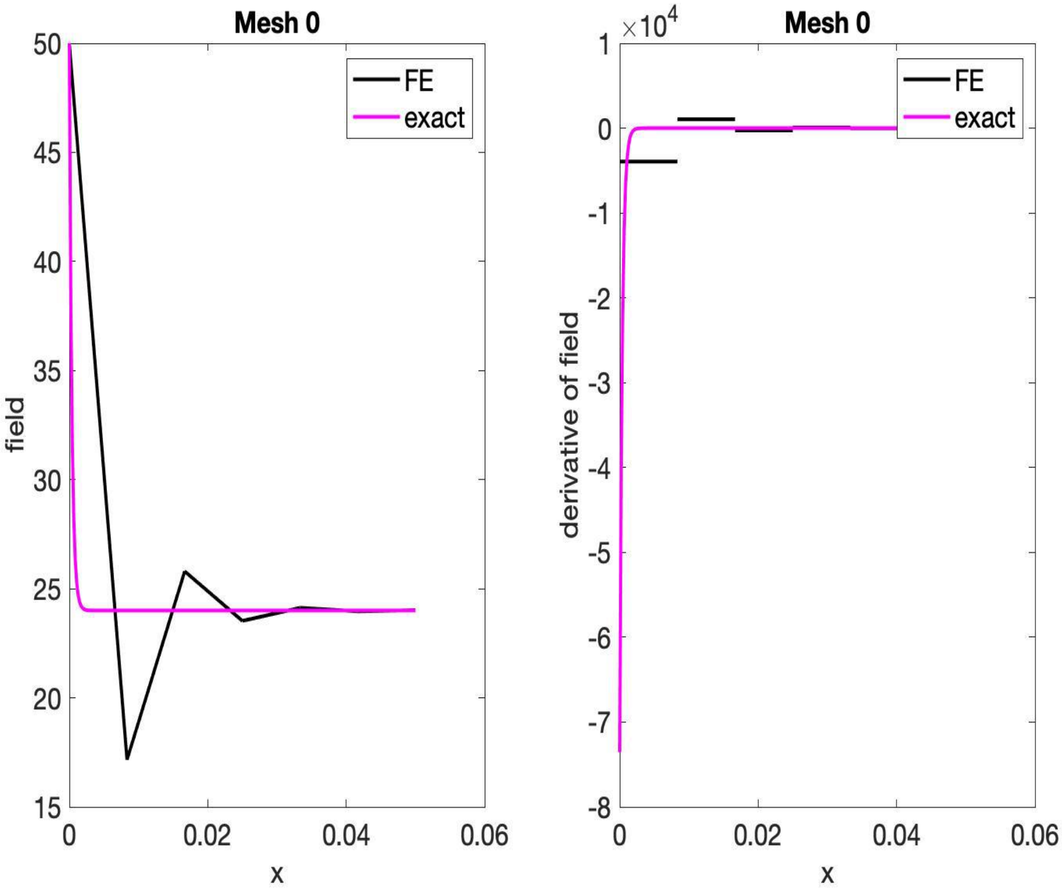


Figure 1: Model II FE plot before refinement

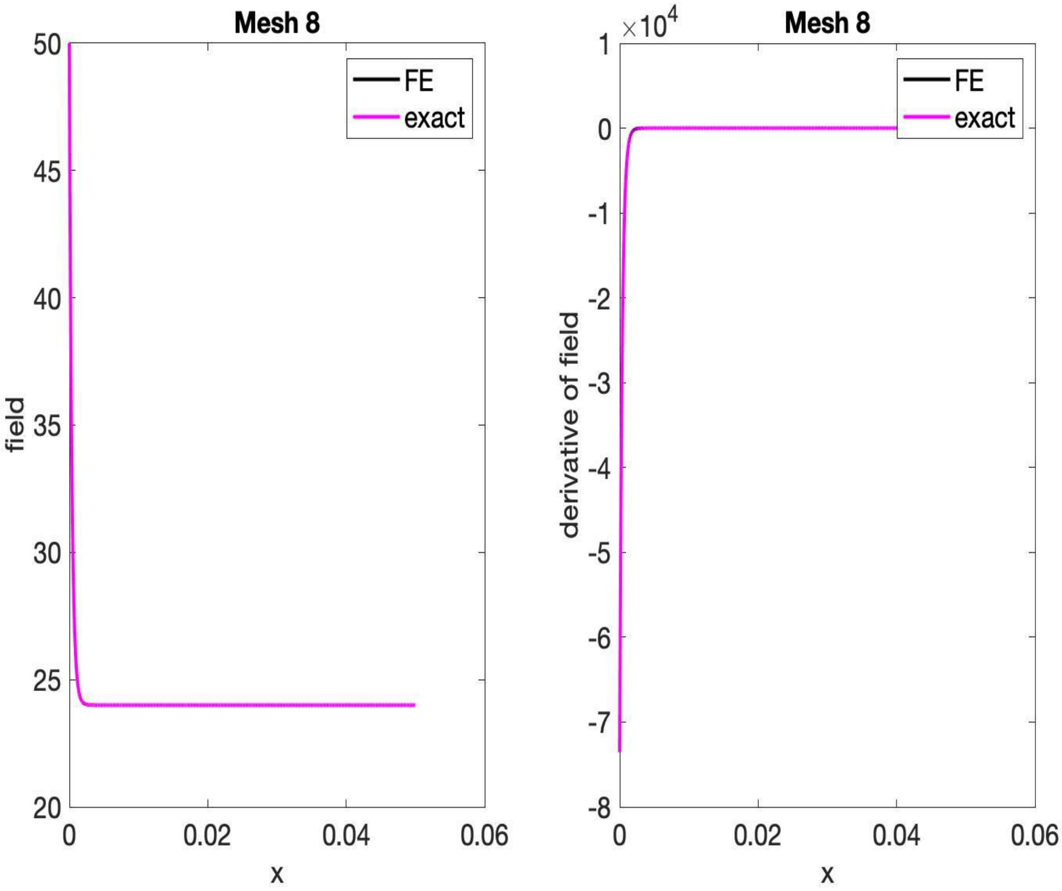


Figure 2: Model II FE plot after 8 refinement

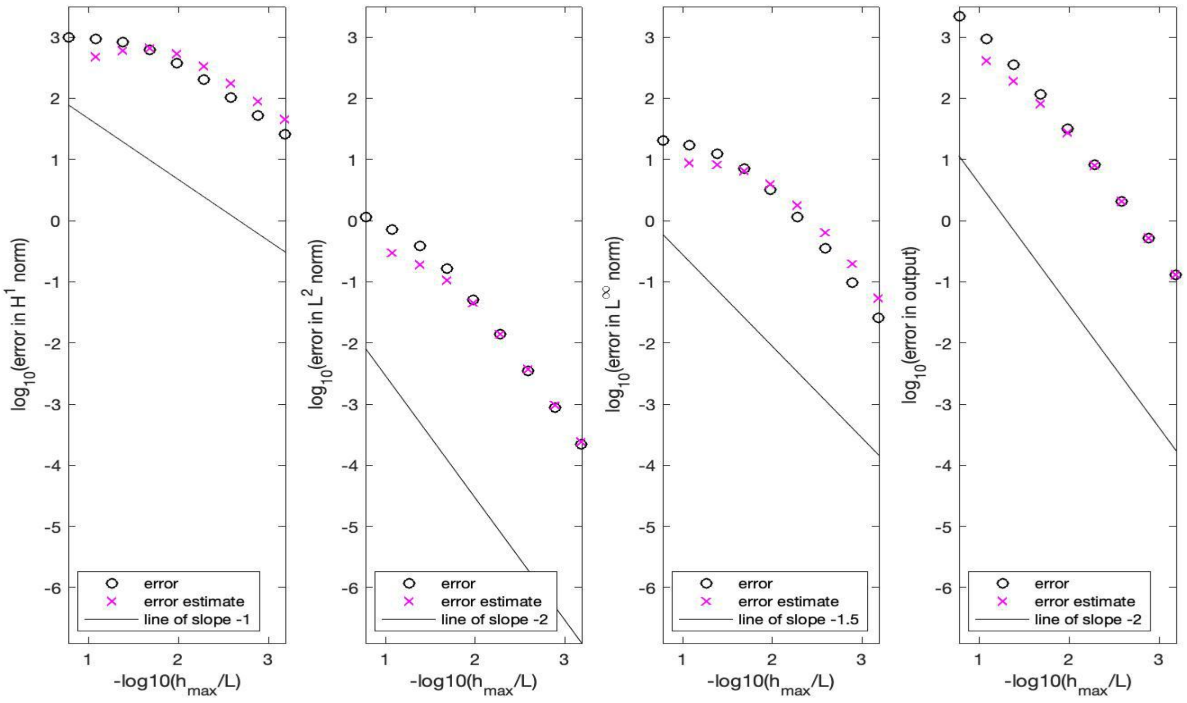


Figure 3: Model II error plot for 9 Meshes

Correct code verification means that if the discretization error goes to zero as the mesh increments decrease to zero and with the right order of convergence then the equations are solved correctly. To ensure that the code is bug-free and works under more general conditions. We need to train it where $\mu(x)$ is a non-constant. Here we propose a model similar to model II with a variable $\mu(x)$

$$\begin{aligned}
\tfrac{-d}{dx}(k(x)\tfrac{du}{dx} \quad + \quad \mu(x) \quad &= \quad f_\Omega \text{ in } \Omega \\
k(x)\tfrac{du}{dx} \quad &= \quad \gamma_1 u \quad - \quad f_{\Gamma_1} \text{ on } \Gamma_2 \\
-k(x)\tfrac{du}{dx} \quad &= \quad \gamma_2 u \quad - \quad f_{\Gamma_2} \text{ on } \Gamma_2 \\
\text{Let } \mu(x) \quad &= \quad ax \quad + \quad b
\end{aligned}$$

Now lets use Model II with this re-defined $\mu(x)$:

$$-kA_{cs}\tfrac{d^2u}{dx^2} + (ax+b)(u-u_\infty) = 0$$

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

$$-k\tfrac{du}{dx} = 0 \text{ on } \Gamma_2,$$

where $\Omega = (0, L), \Gamma_1 = 0, \Gamma_2 = L$ as defined for model II

Now lets use the method of manufactured solutions, assume u(x) = x^2

Now we can evaluate $f_\Omega(x) = -\frac{d}{dx}(k(x)\frac{du}{dx}) + \mu(x)u$

$$f_{\Gamma_1} \quad = \quad (\gamma_1 u \quad - \quad k\tfrac{du}{dx})_{(x=0)}$$

$$f_{\Gamma_2} \quad = \quad (\gamma_2 u \quad - \quad k\tfrac{du}{dx})_{(x=L)}$$

Now given $k(x), \mu(x), \gamma_1, \gamma_2, f_{\Gamma_1}, f_{\Gamma_2}$

Find u_h using the FEA formulation described earlier, then evaluate $\|u - u_h\|$ and show that it converges as the mesh size is refined and with the same order of convergence as model II.

This method of code verification helps us identify any implementation errors that might remain hidden due to lucky coefficient cancellations.
The FE scheme was also implemented on model I from chapter 1, the field and derivative are plotted side by side and super-imposed with the exact solution. It can be seen that the FE scheme produces a close fit to the exact analytical solution. The convergence plots also indicate the correct convergence rate which gives us confidence in our code. To gain more confidence in the correction operation of our code, the FE code was used to solve model Mohammed which represents conduction across a wall between a room and the ambient air outside. It can be seen that the solution to this problem is a linear function. Due to the linear nature of this function, the FE scheme is able to solve it accurately using a relatively large mesh size. The scattered black dots in the derivative plots are due to machine precision error as the plot is magnified. This can be seen by looking at the error plots, where the error starts very small and close to machine precision. However, the round off error due to machine precision are amplified as we decrease the mesh size. This is demonstrated by the increase in error.

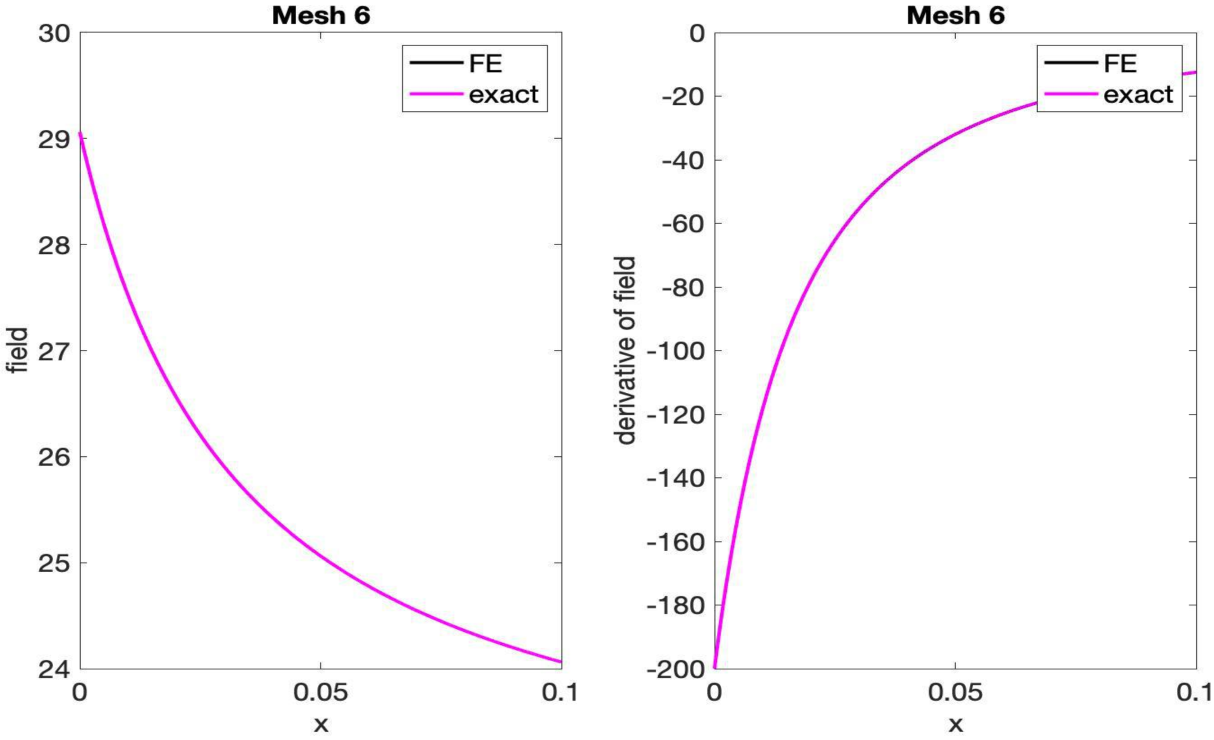


Figure 4: Model I field and derivative plots

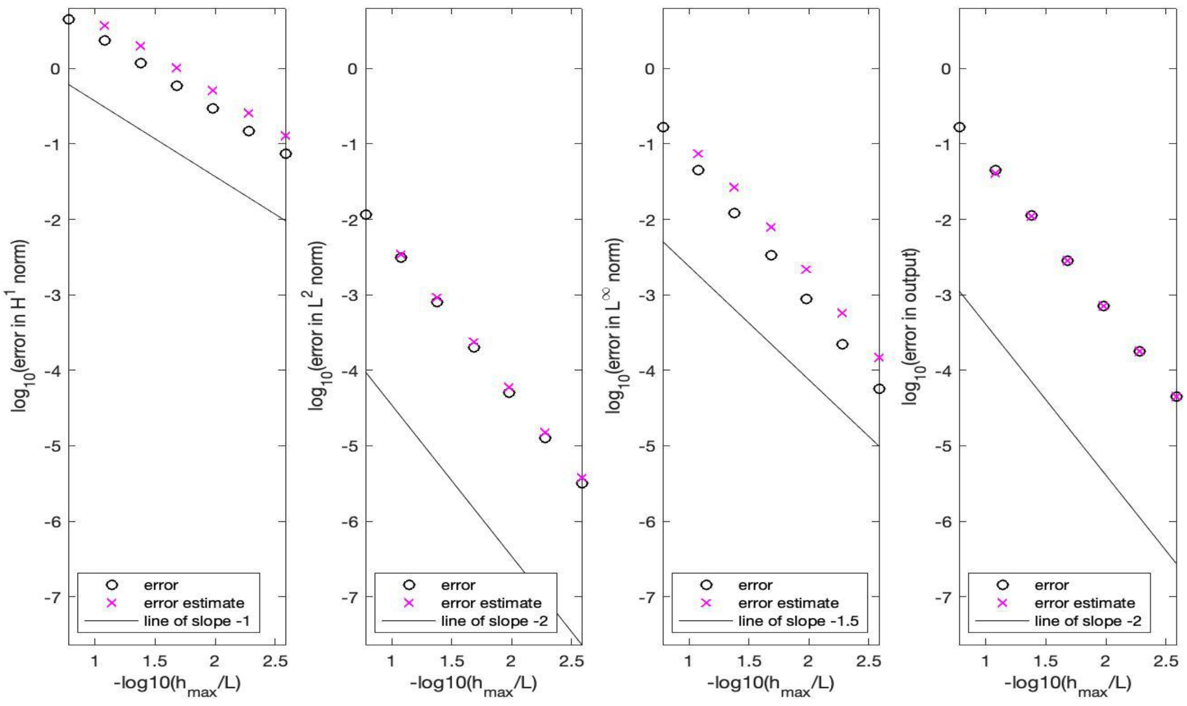


Figure 5: Model I error plots

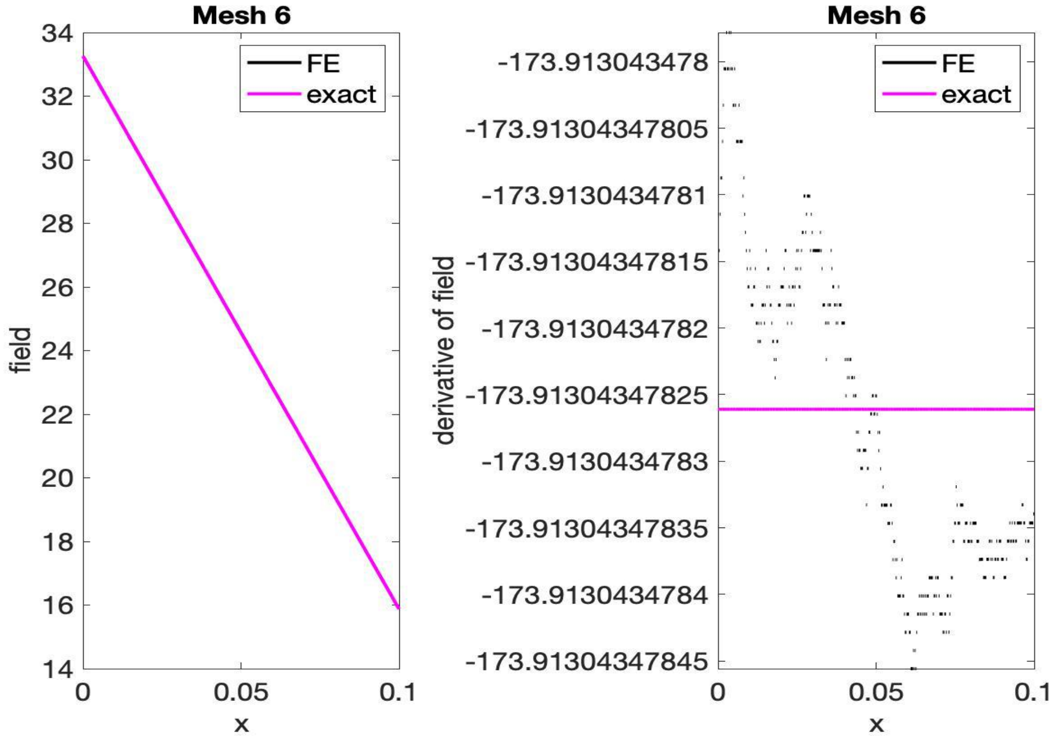


Figure 6: Model Mohammed Field and derivative plots

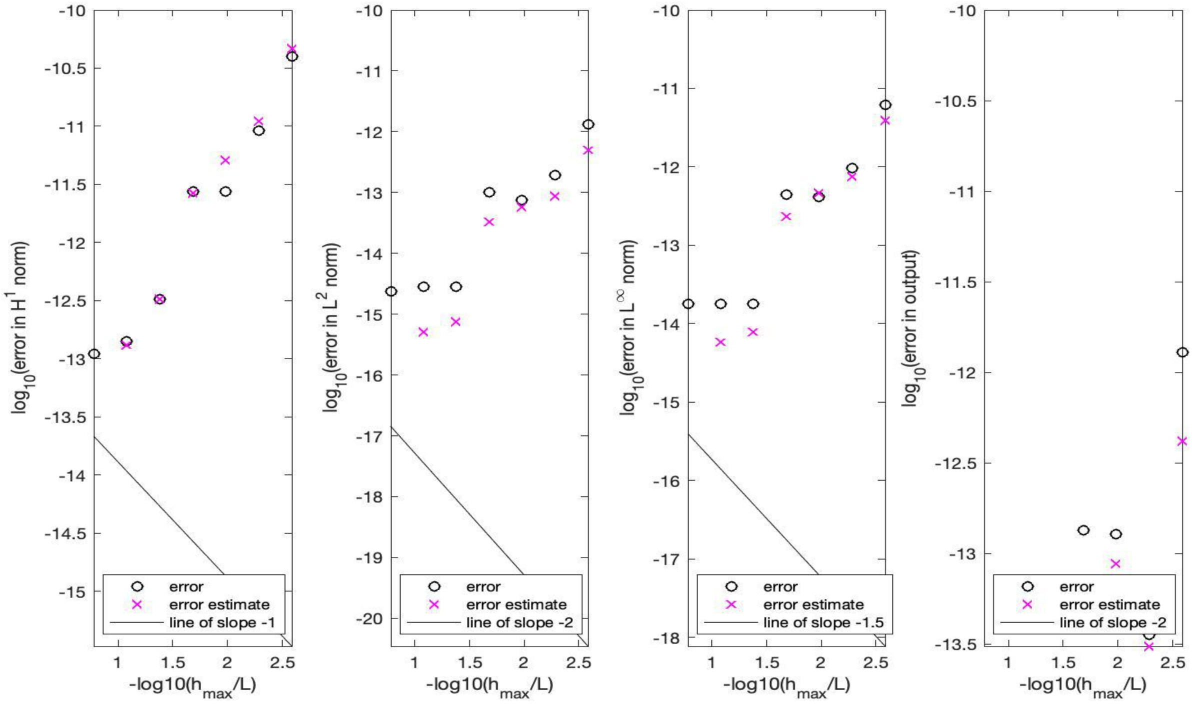


Figure 7: Model Mohammed error plots

We can also consider the model X for which the exact solution is unknown, although one might assume that by running the FE code and observing convergence at the right order that solution produced by the FE is correct this is not always true. It could be the case that the solver is solving for a different solution so we can be converging to the wrong solution as we do not know the exact solution and just using the previous refinement as an error estimator.

Chapter 3

May 17, 2019

1 Summary and Problem Formulation

In this chapter, I will discuss the formulation and implementation of the Finite Difference-Finite Element (Fd-FE) Method for the heat equation for N/R - N/R boundary conditions. This method utilizes a hybrid scheme that uses a finite element procedure in space and a finite difference procedure in time. At the end of the chapter, the results are discussed in terms of verification of correct implementation, convergence, and computational efficiency.

I. Semi-Infinite Fin
A. Dimensional Form

$$\rho c A\frac{dT}{dt} = kA\frac{d^2T}{dx^2} - \eta_{lat}P(T-T_\infty),\ 0<x<L,\ 0<t<t_f$$

$$\kappa A\frac{dT}{dx} = \eta_{bot}A(T-T_\infty),\ x=0,\ 0<t<t_f$$

$$\text{-}\kappa A\frac{dT}{dx} = 0,\ x=L,\ 0<t<t_f$$

$$\mathrm{T} = \mathrm{T}_{ic},\ t=0,\ 0<x<L$$

Assumption:
$\kappa > 0, \rho c > 0, \alpha > 0$
$Bi_{lat} = \frac{\eta_{lat}(\frac{A}{P}}{k} < 1$ For Physical relevance

Finite Element Formulation (space)
Here, the finite element formulation is similar to that of previous chapters with the addition of an extra term that accounts for the time derivative. u_h is defined as the sum of the output multiplied by the basis function vectors. In addition to u_h we have $\dot{u}_h$, the derivative of u_h with respect to time.

A. Time Derivative

$u(x,t) = u_h(x,t) = \sum_{j=1}^{n} u_{hj}(t)\phi_j(x)$, where n = n$_{node}$
$\dot{u}(x,t) = \sum_{j=1}^{n} \dot{u_{hj}}(t)\phi_j(x)$

2. Semi- Discrete Equations

The A and F matrix relationship that result from using the steady state case are not ideal: the F matrix in particular is a function of $\dot{u}$ which makes it difficult to solve numerically. This is where a new matrix called the inertial mass matrix (Minertia) is necessary. This is used to separate the $\dot{u}_h$ components from the incompatible F matrix into an isolated term after the F matrix has been expanded.

$\mathbf{Au}_h = \mathbf{F}^\dagger$

$\mathbf{A} = \leftarrow \mathbf{A^N}$

Let $\tilde{A_{ij}} = \int_0^L k(x) \frac{d\phi_i}{dx} \frac{d\phi_j}{dx} + \mu(x)\phi_i\phi_j dx + \gamma_1\phi_i(0)\phi_j(0) + \gamma_2\phi_i(L)\phi_j(L)$

$\mathbf{M}^{inertia} = \tilde{\mathbf{M}}^{inertia}$

$\tilde{\mathbf{M}_{\mathbf{ij}}}^{inertia} = \int_0^L \rho(x) d\phi_i d\phi_j dx$

$\mathbf{F}^\dagger = \tilde{\mathbf{F}^\dagger} \leftarrow \mathbf{F}^{\dagger\mathbf{N}}$

$\mathbf{F}_i^\dagger = \int_0^L f_\Omega^\dagger \phi_i dx + f_{\Gamma_1}\phi_i(0) + f_{\Gamma_2}\phi_i(L)$

$\mathbf{F}_i^\dagger = \int_0^L f_\Omega \phi_i dx + f_{\Gamma_1}\phi_i(0) + f_{\Gamma_2}\phi_i(L) - \int_0^L \rho(x) \dot{u_h} \phi_i dx$

$\mathbf{F}_i^\dagger = \tilde{F_i} - \sum_{j=1}^n (\int_0^L \rho(x)\phi_i\phi_j dx) \dot{u_h j}$

$$\downarrow$$

$\mathbf{M}^{inertia}\dot{\mathbf{u_h}} + \mathbf{Au_h} = \mathbf{F}\ \mathbf{0} < \mathbf{t} < \mathbf{t_f}$

$\mathbf{u}_h = (\mathbf{I_h u_{ic}}),\ \mathbf{t} = \mathbf{0}$

III. Finite Difference Formulation (Time)

A. θ schemes: Euler Forward, Crank-Nicolson, and Euler Backward

1. System of ODE's

$x(t)$ **: n x 1 vector;**

g : n x 1 vector

$\mathbf{x}_{ic} \quad : \quad nx1vector$

B_1**: n x n non-singular matrix**

$$B_2\textbf{: n x n matrix}$$

$$\mathbf{B}_1\dot{z} \quad + \quad B_2 x \quad = \quad g, 0 \quad < \quad t \quad < \quad t_f$$

2. Grid in Time

$$\Delta t \quad = \quad \frac{t_f}{n_{tsteps}-1}$$

$$\mathbf{t}^k \quad = \quad \Delta t(k \quad - \quad 1), \; 1 \quad < \quad k \quad < \quad n_{tsteps}$$

$$\mathbf{z}^k_{\Delta t} \quad = \quad z(t^k), \; 1 \quad < \quad k \quad < \quad n_{tsepts}$$

3. Approximation

Here we incorporate a new variable to that determines the step size. The parameter θ determines the type of discretization scheme used. Euler forward, Euler backward and Crank Nicolson have the following schemes and their corresponding values of θ:

$$] \; B_1\mathbf{z}^k_{\Delta t} - z^{k-1}_{\Delta t}\overline{{}_{\Delta t}} + \mathbf{B}_2(\theta z^k_{\Delta t} + (1 - \theta(z^{k-1}_{\Delta t})$$

$$\mathbf{z}^k_{\Delta t} = \mathbf{z_{ic}} \quad \mathbf{k = 1}$$

$$\theta = \mathbf{0}\textbf{: Euler Forward (rectangle left)}$$

$$\theta = \mathbf{1/2}\textbf{: Cran-Nicolson (trapezoidal)}$$

$$\theta = \mathbf{1}\textbf{: Euler Backward (rectangle right)}$$

B. Heat Equation 1. Discrete Equations:

$$\mathbf{u}^k_{h,\Delta t}(X) \quad \approx \quad u(x, t^k), 1 \quad < \quad k \quad < \quad n_{tsteps}n$$

$$\mathbf{M}^{inertia}\mathbf{u}^k_{h,\Delta t} \quad - \quad u^{k-1}_{h,\Delta t}\overline{{}_{\Delta t}} + \mathbf{A}(\theta\mathbf{u}^k_{h,\Delta t} \quad + \quad (1{-}\theta)\mathbf{u}^{k-1}_{h,\Delta t}$$

2 Results and Discussion

The first part of the implementation acts as a preliminary verification tool of our mathematical model. The semi infinite plat model was run using uniform refinements and and error plot was obtained. The $L^2(\Omega)$ error norm is a good indicator of correct implementation as its convergence is balanced between the spacial and temporal rates of convergence. Note that [p=1, θ =1] refers to the Euler backward scheme, whereas [p=2, $\theta = 1/2$] refers to the Crank

Nicolson scheme. The alignment of true errors with the error estimators of $\mathbf{L}^2 normindicatesthecorrectimplementationofthecodeinbothschemes.$

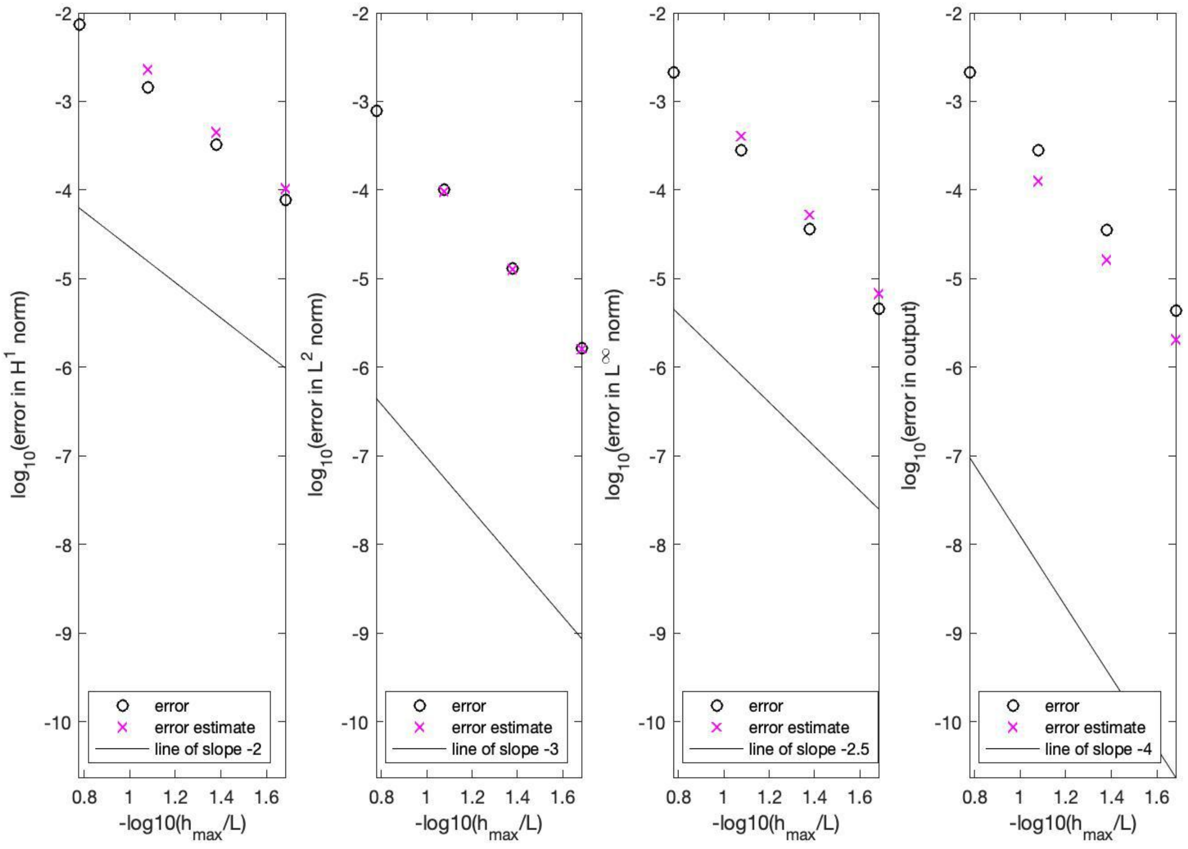


Figure 2: Semi Infinite plate model: Error Plots for first numerical scheme:[p=2, $\theta = 1/2$]

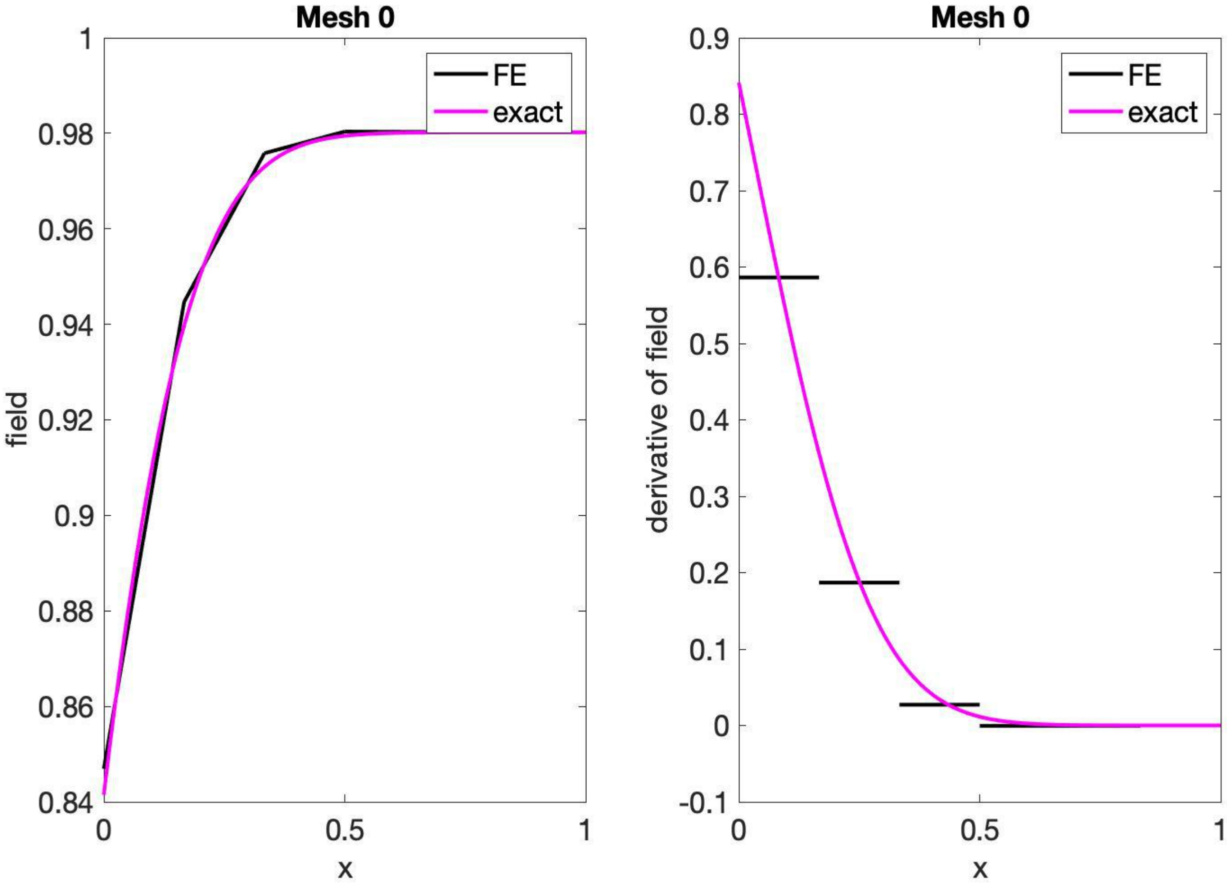


Figure 3: Semi Infinite plate model: Mesh 0 for first numerical scheme: [p=1, $\theta = 1$]

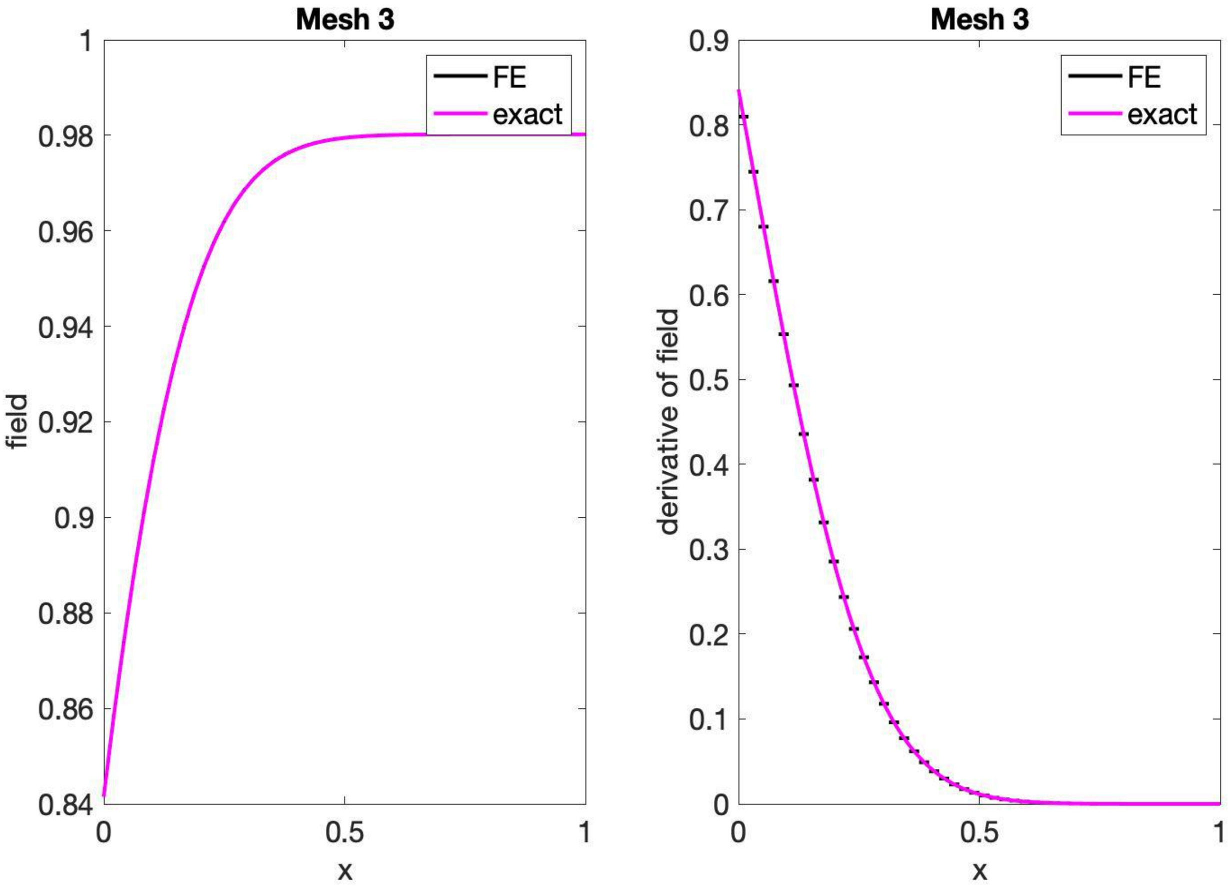


Figure 4: Semi Infinite plate model: Mesh 3 for first numerical scheme: [p=1, $\theta = 1$]

2.1 Burger cooking simulation results

Figure 4 shows the temperature distribution of different parts of the burger over time. The skillet surface is heated up to 180 °C. The burger is then cooked for 3 minutes or 180 seconds, then immediately flipped to cook on the other side. It can be seen, that the side of the pate, which is now exposed to air has a temperature decay similar to that of a semi infinite plate model. This gives us confidence in our choice of thermal models. The burger is then cooked for another 4 minutes where the heat penetrates faster to the core of the burger as indicated by the green line. Figure 6 demonstrates the error behavior of the numerical scheme and its convergence properties. The slope of the error log is -1, indicating a convergence of the first order which is expected of such a scheme.

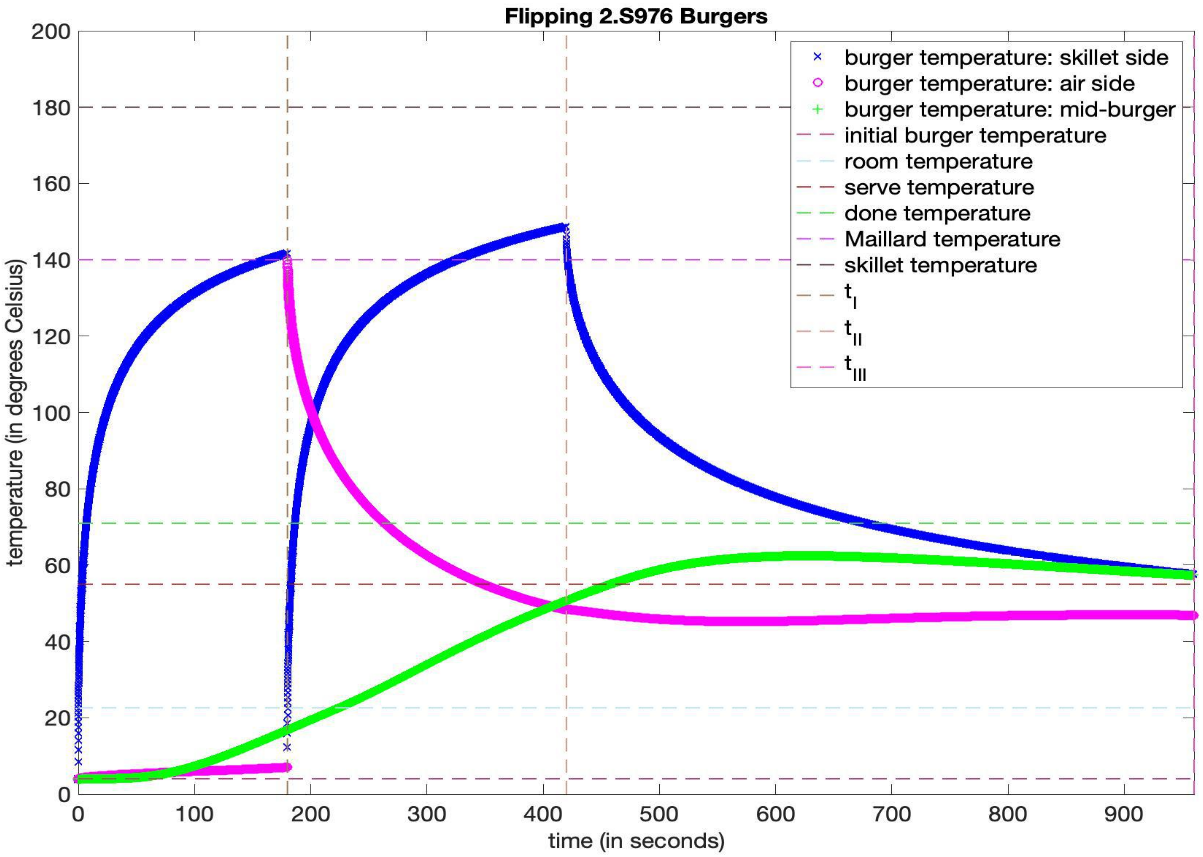


Figure 5: Temperature distribution of different parts of the burgers versus time

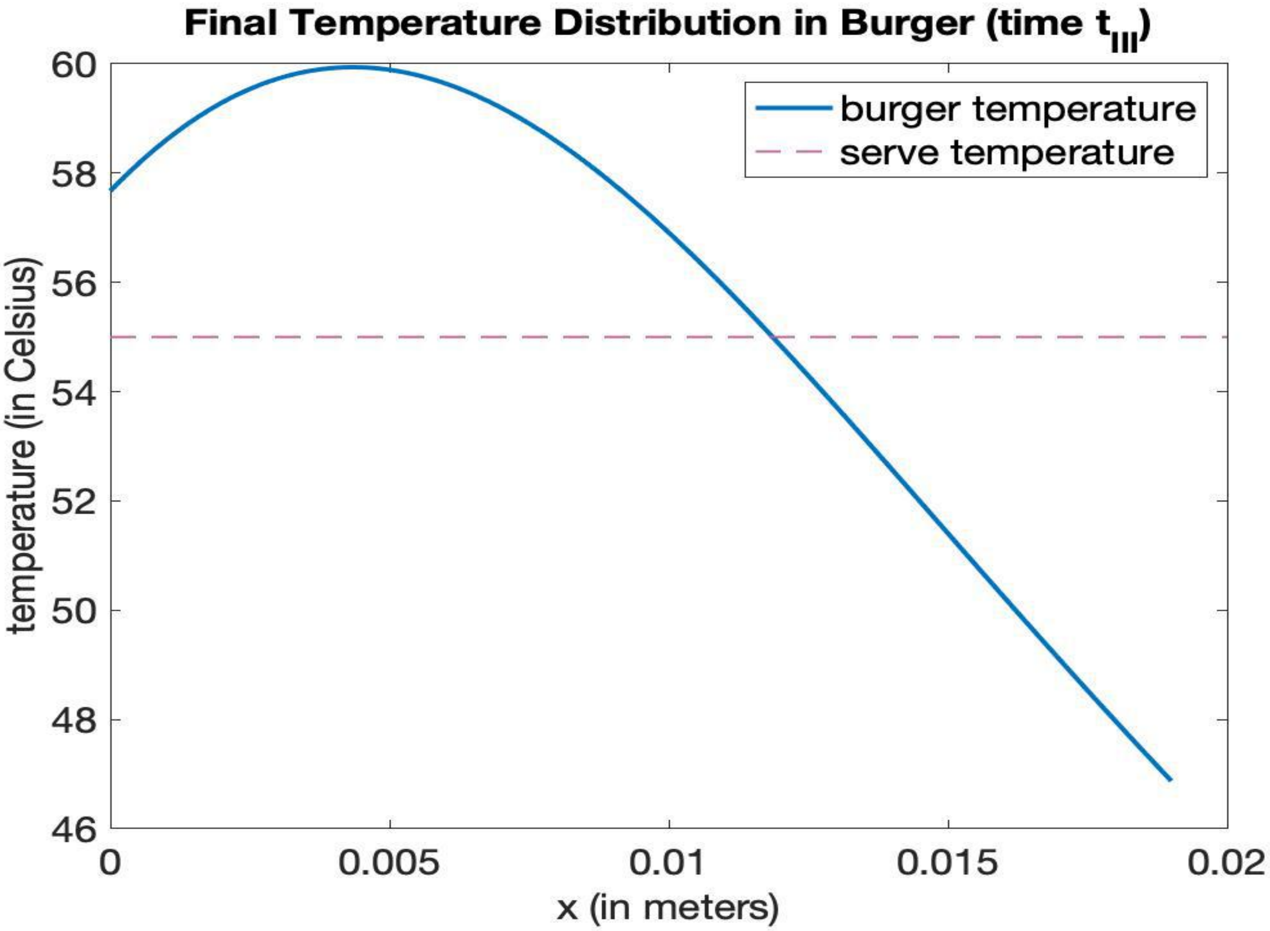


Figure 6: Final Temperature Distribution in the burger

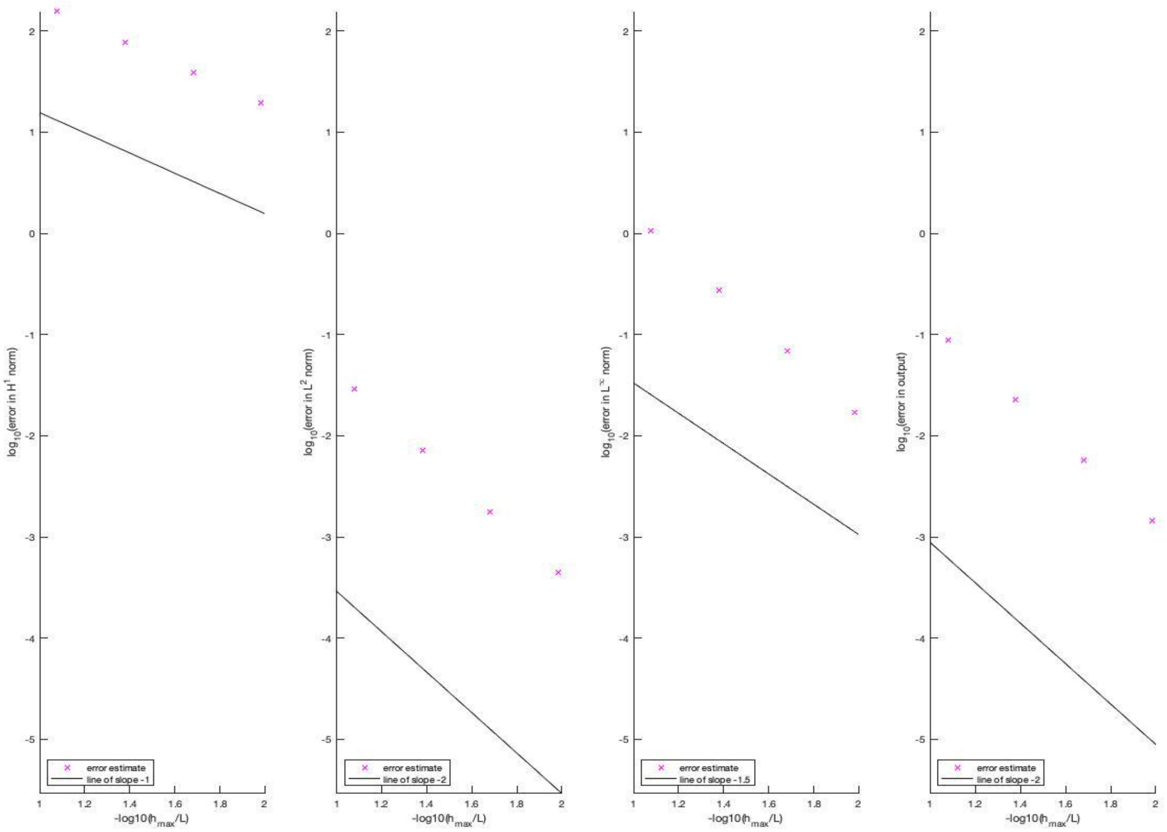


Figure 7: Burger Model: Error plots for the first numerical scheme : [p=1, $\theta = 1$]

In general, we try to obtain the coarsest mesh possible that satisfies our error tolerance and accuracy requirements. The accuracy of the solution is determined by the accuracy of the physical system and the accuracy of the numerical scheme used to solve the equations governing the physical system. A low accuracy physical model will not require a high accuracy numerical scheme as the error would be bounded by the error due to the physical model. Here, we require a tolerance of 0.001 °C. This tolerance might be too low, as burger heat transfer models are usually low fidelity and do not require such high accuracy. We will perform error analysis on 2 different schemes: [p=1, $\theta = 1$] and [p = 2, θ= 1/2]. From the error plots, it can be seen that the error error bound for the first mesh refinement of p=1 and $\theta = 1$ is 0.001. Note that we analyze the error in the output norm.

$|\mathrm{u}(:,\mathrm{t}_f) - u_{h,\Delta t}^{ntsteps}|^{(l)} \sim 2^{-rl}(C_u^1 2^{-l} + C_u^2)$
r = 1 for p = 1, r=2 for p = 2.
To evaluate the coarsest mesh For P = 1, $\theta = 1$:

$\frac{Final Error}{Initiatial Error} = \frac{\epsilon^{(l)}}{\epsilon^{(1)}}$

$\frac{0.001}{0.01} = \frac{2^{-1*(l)}}{2^{-1*1}}$

l $> 4.3 \longrightarrow l = 5$

To evaluate the coarsest mesh For P = 2, $\theta = 1/2$:
$\frac{0.001}{0.01*2} = \frac{2^{-2*l}}{2^{-2*1}}$

$8^{(l)} = 50$

l$> 1.9 \longrightarrow$ l = 2

Another area that we would like to investigate is the computational efficiency of both numerical schemes. To understand the number of computations per refinement, we need to know the number of nodes in each refinement. Given Δt_0, h_0, and σ (> 1). σ a constant used to refine the the mesh in space and time simultaneously such that the error in both space and time is decreases to be of similar order
Consider (Δt_0 , T$_{h_0}$)$\rightarrow$ ($\Delta t_0/\sigma, T_{h0/2}$) . If we refine the mesh the mesh incrementally from 1 to l.

($\frac{\Delta t_0}{\sigma^l}, T_{h0/2^l}$) $\rightarrow$ ($\frac{\Delta t_0}{\sigma^{l+1}}$), T$_{h0/2^{l+1}}$).

This leads to an operation count of O($\frac{L}{(\frac{h_0}{2^l})^2}$). The total number of operations $= n_{tsteps}n_{elements}$. For p = 1, $\theta = 1$ we get $\sigma = 4$. For p = 2, $\theta = 1/2$, we get $\sigma = 2\sqrt{2}$. The value of σ such that $2^r/\sigma^q = 1$. Lets denote the ratio of number of computations of prescribed accuracy for [p=1 and θ =1] relative to [p = 2 and $\theta = 1/2$] by k.

$$\text{k} = \frac{4^5 2^5}{(2\sqrt{2})^2 2^3} = 512$$

The quantity was doubled because it was assumed that solving a penta-diagonal matrix requires double the operational count compared to a tri-diagonal matrix. This states that the latter approach is superior in terms of computational efficiency. An assumption that has been made in calculating this ratio is that the the number of refinements needed to achieve desired accuracy is calculated apriori and the solver will only operate on the last refinement rather than iteratively refine until the prescribed error is reached.
Based on my literature review, when using the electric griddle, it is recommended to preheat the electric griddle to 190 °C. A beef patty with a diameter of 4.5 inches and a thickness of 0.75 inches is then then cooked for 4 minutes. The burger is then flipped and cooked for another 4 minutes. The lowest temperature anywhere inside the patty should be above 72°C to ensure the bacteria is killed and the burger is safe to eat. The default simulation sets the skillet temperature to 180°C and the burger is cooked for 3 minutes on one side. Then, the burger is flipped and the other side is cooked at the same skillet temperature. The center of

the burger (the region with the lowest temperature) takes about 400 second or 6.5 minutes to reach the highest temperature at around 65°C. This temperature is not high enough for safe eating. It seems that the model is predicting that a higher skillet temperature and a longer period is needed to cook the burger correctly. This discrepancy might indicate several flaws in out mathematical model. First, a Dirichlet boundary condition might be less realistic than a Neumann boundary condition in which the heat flux is imposed on the surface rather than a temperature boundary condition. Second, another source of error could be the thickness of the oil layer separating the patty and the skillet.

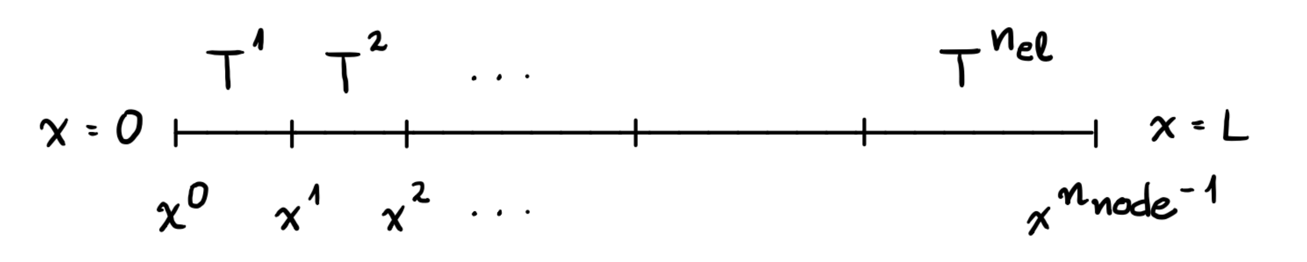
Chapter 4

FE Method For Beam Eigenproblems

1. Summary

I. Formulation

A. Mesh generation



B. Rayleigh- Ritz Basis Functions

The basis functions used for beam bending are known as hermitian approximations. A global and a local grid system is used in mapping.

Two types of basis functions are used for this method: one that is either zero or one and a zero derivative at every node, and another that is zero at every node, but the derivative is either zero or one. These basis functions are called Hermitian (C_1) basis function, and they accounts for the two degrees of freedom at every node.The basis functions has the following properties:

1. For nodes: $1 \leq$ i $\leq n_{node}$ -2

   $\Phi_{i,1}(x^i) = 1$, $\Phi_{i,1}(x^{i-1}) = \Phi_{i,1}(x^{i+1}) = 0$

   $\Phi'_{i,1}(x^i) = \Phi'_{i,1}(x^{i-1}) = \Phi'_{i,1}(x^{i+1}) = 0$

Zero for x $\notin T^i$, cubic in T^{i+1}

$\Phi_{i,2}(x^i) = \Phi_{i,2}(x^{i-1}) = \Phi_{i,2}(x^{i+1}) = 0$

$\Phi_{i,2}^{'}(x^i) = 1$, $\Phi_{i,2}^{'}(x^{i-1}) = \Phi_{i,2}^{'}(x^{i+1}) = 0$

$\Phi_{i,2} = 0$ for x $\notin T^i$ U T_{i+1}

For $1 \leq i \leq n_{node} - 2$, $1 \leq$ k ≤ 2:

q$\phi_{[i,k]}^{2\to1} = \Phi_{i,k}$

2. Basis Functions: node i $\equiv n_{node} - 1$

$\Phi_{i,1}$: cubic in T^i, zero for x $\notin T^i$

$\Phi_{i,1}(x^i)= 1, \Phi_{i,1}(x^{i-1}) = 0$

$\Phi_{i,1}^{'}(x^i)=\Phi_{i,1}^{'}(x^{i-1}) = 0$

$\Phi_{i,2}$: cubic in T^i, zero for x $\notin T^i$

$\Phi_{i,2}(x^i) = \Phi_{i,2}(x^{i-1}) = 0$

$\Phi_{i,2}^{'}(x^i) = 1$, $\Phi_{i,2}^{'}(x^{i-1}) = 0$

For i = $node_{node} - 1$, $1 \leq$ k ≤ 2:

$\phi_{[i,k]}^{2\to1} = \Phi_{i,k}$

3. Non- Basis Functions: node i $\equiv 0$

$\Phi_{i,1}$: cubic in T^i, zero for x $\notin T^i$

$\Phi_{i,1}(x^i)= 1, \Phi_{i,1}(x^{i+1}) = 0$

$\Phi_{i,1}^{'}(x^i)=\Phi_{i,1}^{'}(x^{i+1}) = 0$

$\Phi_{i,2}$: cubic in T^i, zero for x $\notin T^i$

$\Phi_{i,2}(x^i) = \Phi_{i,2}(x^{i+1}) = 0$

$\Phi_{i,2}^{'}(x^i) = 1$, $\Phi_{i,2}^{'}(x^{i+1}) = 0$

4. FE Representation n = $2 \cdot (n_{node} - 1)$

    $u_h(x) = \sum_{i=1}^{n_{node}-1} \sum_{k=1}^{2} u_{hi,k} \Phi_{i,k}(x)$

    $u_{h[i,k]}^{2\to1} = u_{hi,k}$

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

6. Nodal Interpretation

a. Function

For l = 1, …, $n_{node} - 1$

$u_h(x^l)$ = $\sum_{i=1}^{n_{node}-1} \sum_{k=1}^{2} u_{hi,k} \Phi_{i,k}(x^l)$

b. Derivative

For l = 1 ,… , $n_{node} - 1$

$u_h^{'}(x)$ = $\sum_{i=1}^{n_{node}-1} \sum_{k=1}^{2} u_{hi,k} \Phi_{i,k}^{'}(x)$

7. C^1 Continuity

$u_h(x)$ = $\sum_{i=1}^{n_{node}-1} \sum_{k=1}^{2} u_{hi,k} \Phi_{i,k}(x)$

$u_{h[i,k]}^{2\to1}$ = $u_{hi,k}$

$\sum_{j=1}^{n} u_{hj} \phi_j(x)$

$u_h(x_-^*) = u_h(x_+^*)$

$u_h^{'}(x_-^*) = u_h^{'}(x_+^*)$

C. Rayleigh - Ritz Approximation

1. **Minimization: n = 2 · ($n_{node} - 1$)**

$\Pi(\sum_{j=1}^{n} u_{hj} \phi_j(x))$ < $\Pi(\sum_{j=1}^{n} w_{hj} \phi_j(x))$

X_h : Piecewise - cubic C^1 on T_h

$w_h(0)$ = $w_h^{'}(0)$ = 0 for any $w_h \notin \mathbb{R}^n$, $w_h \neq u_h$

2. Comparison Proposition

$E_{III}(u - u_h) < E_{III}(u - w_h)$ for any $w_h \notin X_h, w_h \neq u_h$

Since $I_h^{Hermitian} u \notin X_h$

$E_{III}(u - u_h) < E_{III}(u - (I_h^{Hermitian} u))$

D. Discrete Equations

1. Linear System n = 2 · ($n_{node} - 1$)

u_{h} $\notin \mathbb{R}^n$ satisfies

<u>A</u> u_h = <u>F</u>

2. Xylophone bar problem

The tuning of the xylophone bar uses the FE-FD eigenproblem solver to determine the mode of resonant frequency based on the material and its geometry. The quadratic cut out defined by the function H(x) is used to tune the resonant frequency and adjust the placement of the holes within the physical structure. The mode of resonance corresponds to the Eigen value. The first eigen value corresponds to the mode of resonance and so on. The holes are located at the resonant frequency of the vibrating bar. By creating holes, we created fixed nodes at which the bar cannot resonate

2.1 Geometry (Upside - Down Bar)

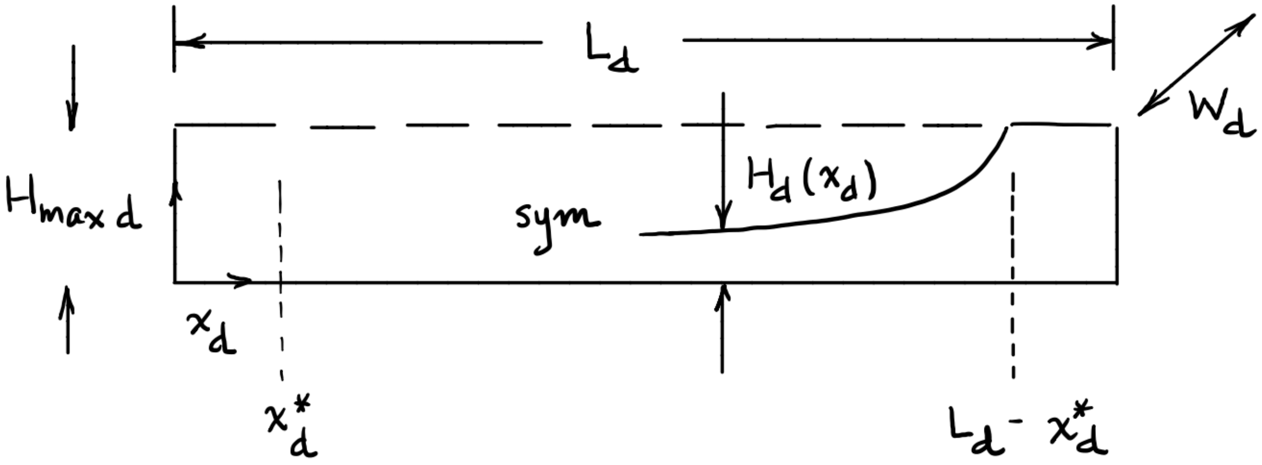


Figure 1: Geometry of the xylophone

Where $H_d(x_d) = \begin{cases} H_{maxd}[(1-p_2)(\frac{L_{d/2}-x_d}{L_{d/2}-x^*_d})] & x^*_d \leq x \leq L_d - x^*_d \\ H_{maxd} & L_d - x^*_d \leq x \leq L_d \end{cases}$

p_1 is an even integer and its set so that $p_1 = 4$, that way we have one design variable p_2 that we are going to optimize.

$p_2 \in [0.05, 1.00]$

2.2 Governing Eigenproblem

A. Dimensional Form $\Omega_d \equiv (0, L_d)$

$$\frac{d^2}{dx_d^2}(\frac{E_d W d H_d^3(x_d)}{12}\frac{d^2 u_d^{(k)}}{dx_d^2}) = \lambda_d^{(k)} \rho_d W_d H_d(x_d) u_d \; 0 < x_d < L_d$$

B. Fundamental Harmonics

$f_d^{(1)} = 0 \; u_d^{(1)} \; \alpha \; const_1 \rightarrow$ Translation $f_d^{(2)} = 0 \; u_d^{(2)} \; \alpha \; const_2 \rightarrow$ Rotation

$f_d^{(3)} > 0$ Fundamental mode

$f_d^{(k)} > 0$ $(k-3)^{th}$ harmonic

2.3 Finite Element Approximation

$[u_h^{(k)}, \lambda_h^{(k)}]$, k = 1,2, … , n = 2 $\cdot n_{node}$

$f_h^{(k)} = \sqrt{\lambda_h^{(k)}} / 2 \cdot \pi$, $f_{dh}^{(k)} = f_h^{(k)} / \tau_d$

2.4 Design Problem

A. Objective

Let R $\equiv \frac{f_d^{(4)}}{f_d^{(3)}} \equiv \frac{fundamental}{harmonic}$

R = $func(x^*, p_2)$

Given x^* and R_{target} $\equiv$ desired ratio

Find p_2^{opt} such that $\mid R(p_2^{opt})$ - $R_{target} \mid \leq tol_R$

B. FE Approximation

Let $R_h = \frac{f_{dh}^{(4)}}{f_{dh}^{(3)}}$

$R_h = func(x^*, p_2, h)$

Given x^* and R_{target} $\equiv$ desired ratio

Find p_{2h}^{opt} such that $\mid R(p_{2h}^{opt})$ - $R_{target} \mid \leq tol_R$

Root finding algorithm:

For each hole, we find where the polynomial crosses the x-axis, thats where the sign of the shape function changes. The we find the zeros in the local domain:

$$\sum_{l=1}^{4} u_{h\ lg2(l,m^*)}^{(3)} S_{lm^*}\left(x^{hole}\right) = 0$$

Thenm we need to scale to the dimensional domain:

$$x_{d\ h}^{hole} = (x^{lg(1,m^*} + h^{m^*} x^{hole})) \cdot L_d$$

The algortithm loops through all the elements and finds all elements in the domain:

```
counter = 0;
for i = 1: n_el
    if u3(lg2(1,i)).*u3(lg2(3,i)) < 0;
        counter = counter +1;
        u3_ = [u3(lg2(1,i)) u3(lg2(2,i))    u3(lg2(3,i))
u3(lg2(4,i))];
        x(counter) = fzero(@(x) u3_*hshape_fcn(x,h(i)), [0,1]);
        xhole_d(counter) = (xpts(lg(1,i)) + h(i)*x(counter))*L_d;
    end
end
```

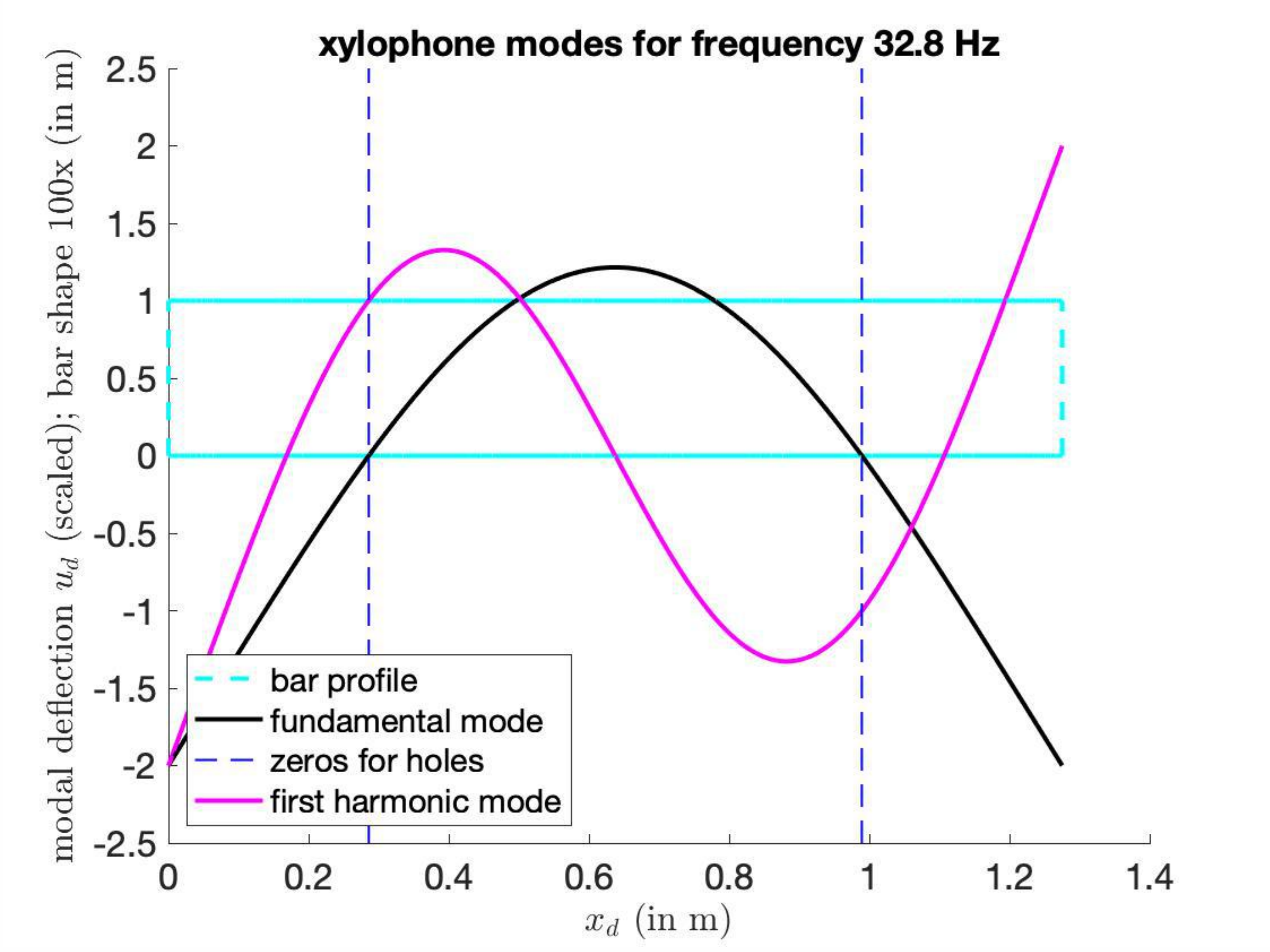


Figure 2: Fundamental and first harmonic modes superimposed on the bar profile

Figure 2 verifies the correct implementation of the software as we can see that the fundmanetal mode passes through the hole positions that we chose for that specific frequency.

2.5 Caresta Case

The caresta case is used as a verification tool for the correct imeplementation of the xylo bar design code. The benefits are 2-fold: i) the calculation of the fundamental and first harmonic frequency, and ii) the determination of the length L_d required to realize a desired (dimensional) fundamnetal frequency.

Objective:

Given $x^*, R_{target} \rightarrow p_{2h}^{opt}(x^*, R_{target})$ [from Timbre optimization] and desired funamental frequency $f_{target}^{(3)}$. We find L_d such that:

$| f_{dh}^{(3)}(L_{dh}) - f_{targetd}^{(3)}| \leq tol_{f^{(3)}}$

Once L_{dh} is found, we can place the holes by finding $x_d^{hole(1)}$, $x_d^{hole(2)}$ such that

$u_d^{(3)}(x_d^{hole(1)}) = u_d^{(3)}(x_d^{hole(2)}) = 0$

To test the code:

```
frequency4_d, err_frequency4_d, L_d, xhole_d, p2opt] =
xylo_bar_design3(32.8, 3, 0.01, 0.05, [1,1], 2.1e11, 7800, true, 1)
```

Fundamental frequency was set to 32.8 Hz. Aspect ratio of 3, $H_d = 0.01$, $x^* = 0.05$,

p_2 interval is fixed to [1,1]. This means that we are not varying p_2 and its no longer our design variable. $E_{caresta} = 2.1e11$ Pa, $\rho_{caresta} = 7800\ kg/m^3$. By running the Xylophone design code with these inputs, we get an L_{dh} value of 1.275 m. We also get a fundamental and harmonic frequencies of 32.8 Hz, and 90.44 Hz respectively. Since these values correctly match the same values in the caresta case, this provides confidence in the correct implementation of the Finite element code in both calculating the frequencies and determining the right length.

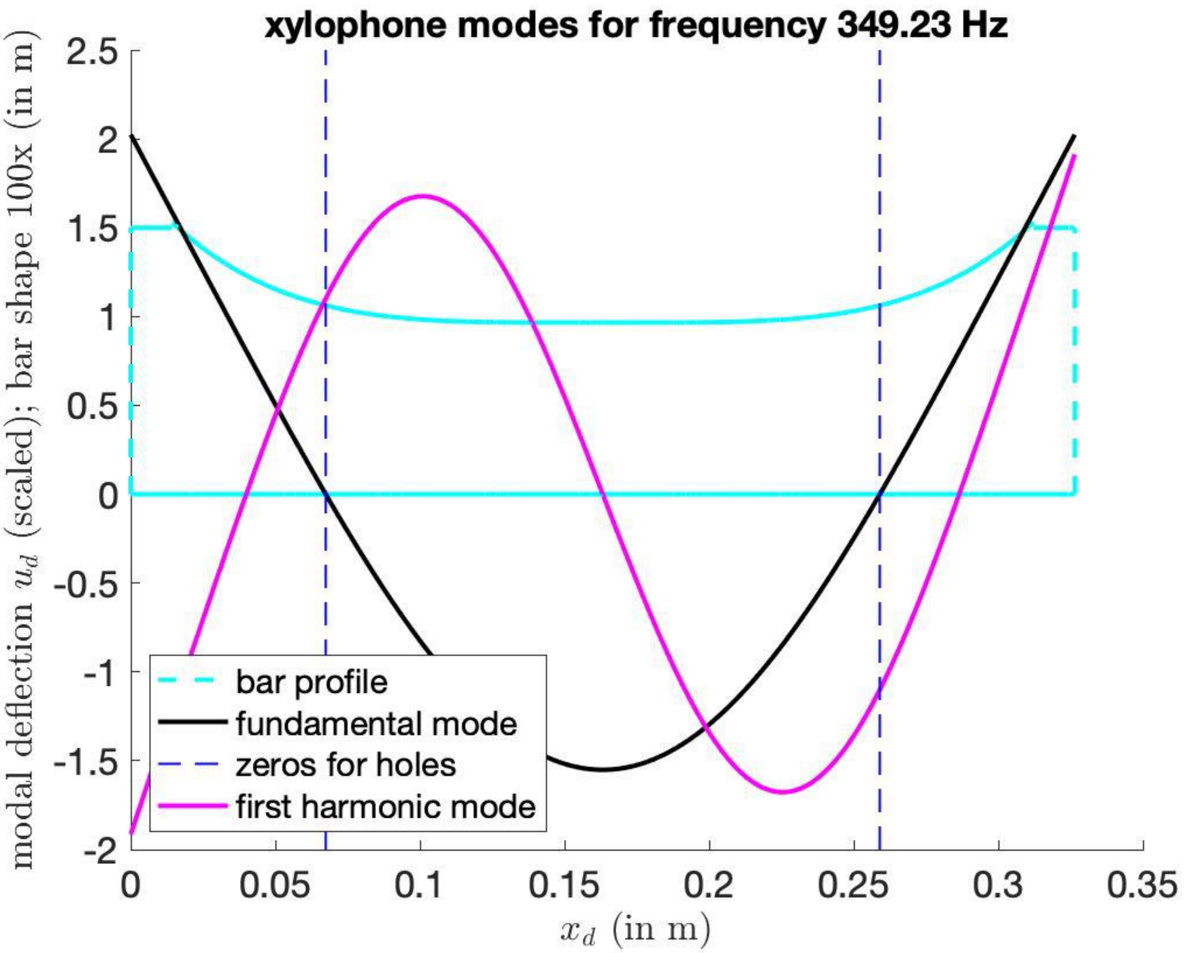
Tuning the xylophone bar

We can now tune the xylphone to produce the pitch and frequency ratio, we desire. I decided to tune it for an F4 pitch and a frequency ratio of 3 ("quint").

The inputs for tuning the xylophone are shown below:

```
frequency3target_d = 349.23; % pitch F4
R_target = 3;                % quint
suppress = false;
Hmax_d = 0.015;
xstar = 0.05;
p2_interval = [0.6438,0.6438];
Ebar_d = 1.4e10;
rhobar_d = 835;
justcalc_L_d = true;
```

By running these inputs into the xylphone design code, we get a length $L_d = 0.3265$ m and $p_2^{opt} = 0.6438$.



We already have error estimates for $f_d^{(3)}$ and $f_d^{(4)}$. One way to find an upper and a lower bound to the aspect ratio $\frac{f_d^{(4)}}{f_d^{(3)}}$ is using the upper and lower bounds of $f_d^{(4)}$ and $f_d^{(3)}$.

Upper bound of $\frac{f_d^{(4)}}{f_d^{(3)}} = \frac{\frac{max(f_4)}{min(f_3)} - \frac{f_4}{f_3}}{\frac{f_4}{f_3}} = \frac{\frac{f_4+\epsilon_4}{f_3-\epsilon_3} - \frac{f_4}{f_3}}{\frac{f_4}{f_3}}$

Lower bound of $\frac{f_d^{(4)}}{f_d^{(3)}} = abs(\frac{\frac{min(f_4)}{max(f_3)} - \frac{f_4}{f_3}}{\frac{f_4}{f_3}}) = abs(\frac{\frac{f_4-\epsilon_4}{f_3+\epsilon_3} - \frac{f_4}{f_3}}{\frac{f_4}{f_3}})$

$$f_d^{(3)} = 349.23Hz$$
$$f_d^{(4)} = 1.0446kHz$$
$$\epsilon[f_d^{(3)}] = 5.8587e-06$$
$$\epsilon[f_d^{(4)}] = 9.9133e-05$$
$$\epsilon[\frac{f_d^{(4)}}{f_d^{(3)}}] = 1.8651e-08$$

3. **Error Analysis**

Note that the error found above is way below the hearing sensitivity of human, this makes it unecessary to reach a refined level of discretization when a much coarser mesh will suffice.

3.1 A Priori Error Estimator

a. Smoothness Assumption

u_{xxxx} exists for all x in (0,L), except for jumps in u_{xx} , u_{xxx} at elemental boundaries.

Then

$$|u - u_h| \, C_u h^{r(Q)} as \; h \to 0$$
$$Q = H^2(\Omega), \; r = 2$$
$$Q = L^2(\Omega), \; r = 4$$
$$Q = L^\infty(\Omega), \; r = 4$$
$$Q = output, \; r = 4$$

3.2 A Posteriori Error Estimator

Given Q, $(u_h, \; u_{h/2})$, r $\equiv$ r(Q)

$$\Delta_h^Q \equiv |u_{h/2} - u_h|_Q / (1 - 2^{-r})$$
$$\Delta_{h/2}^Q \equiv |u_{h/2} - u_h|_Q / (2^r - 1)$$

Mode 1 and Mode 2 error plots are generated for the F4 pitch with a frequency of 349.23 Hz and "quint" frequency ratio. The error in $f_d^{(3)}$, and $f_d^{(4)}$ were found to be 5.8587e-06 and 9.9133e-05 respectively. The plots are shown below. It can be seen that the error in H^2 norm has a slope of -2, while the error in L^2, L^∞ norms and the output all have slopes of -4. This trend in error convergence matches our expectations from the theory. In general,

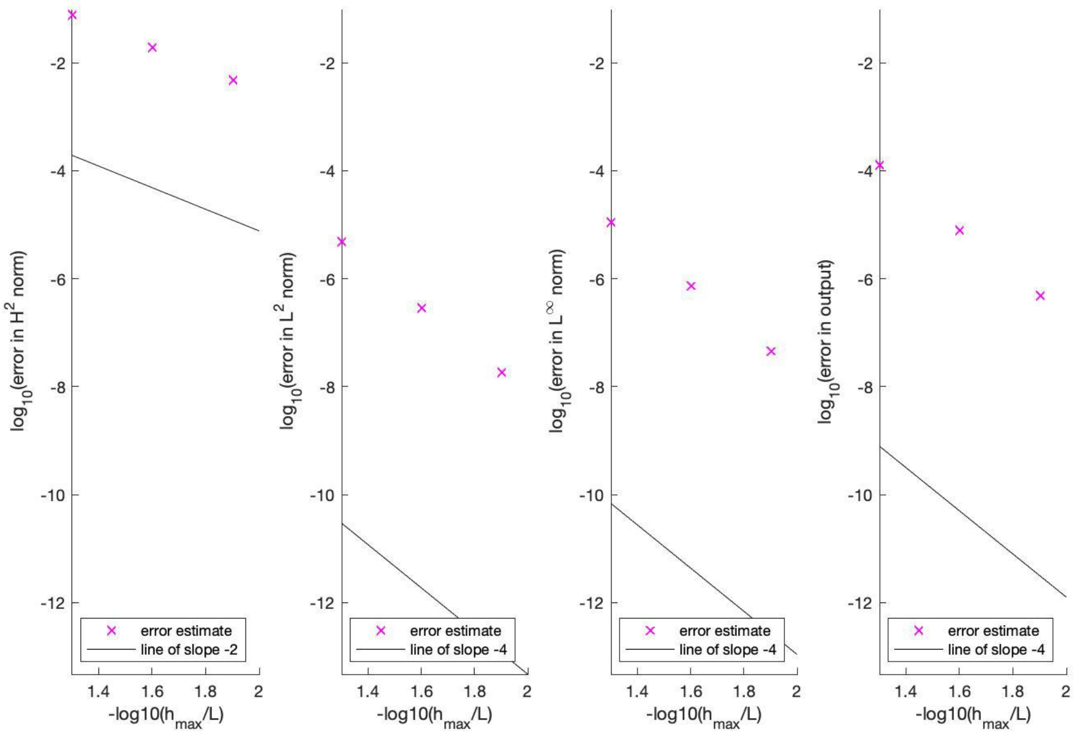


Figure 3: Error plots for the fundamental frequency mode of an F4 pitch

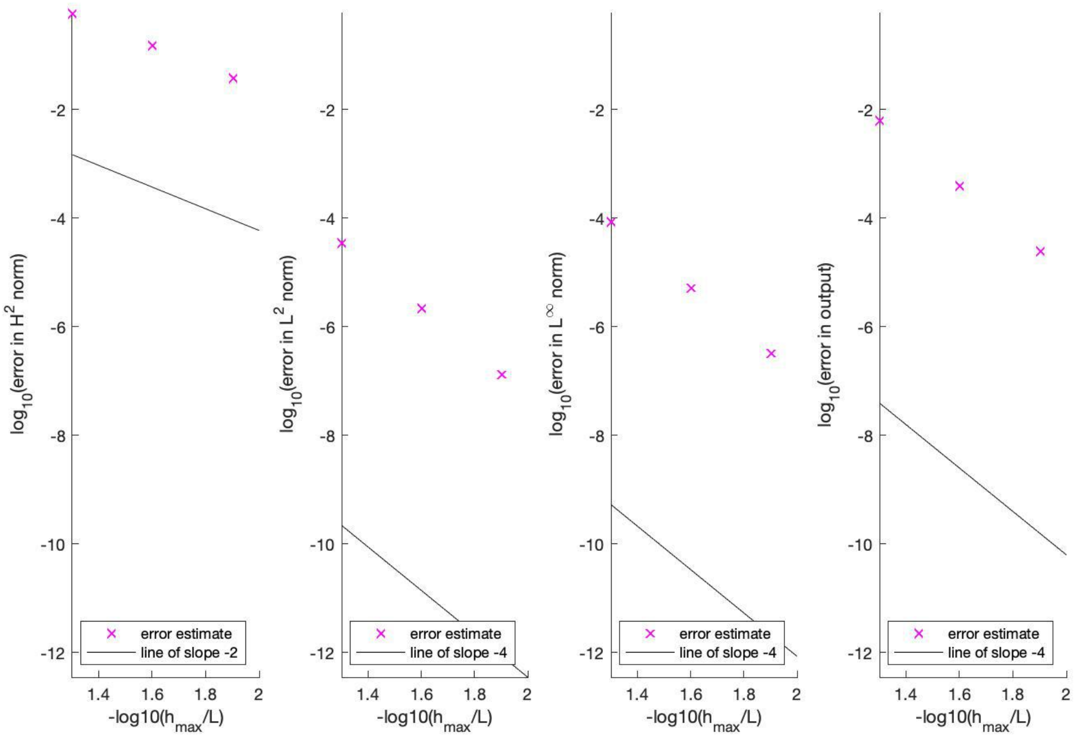


Figure 4: Error plots for the first harmonic frequency mode for an F4 pitch

Second, we analyze the error for a higher frequency, an F5 pitch with a frequency of 698.46 Hz, but the same frequency ration of 3 or "quint". The error in $f_d^{(3)}$, and $f_d^{(4)}$ were found to be 1.1717e-05 and 1.9827e-04 respectively. This is slightly higher than the error induced by the lower target frequency plot. This is expected as a higher frequency means there are steeper gradients which are harder to approximate or fit with a finite set of polynomials.

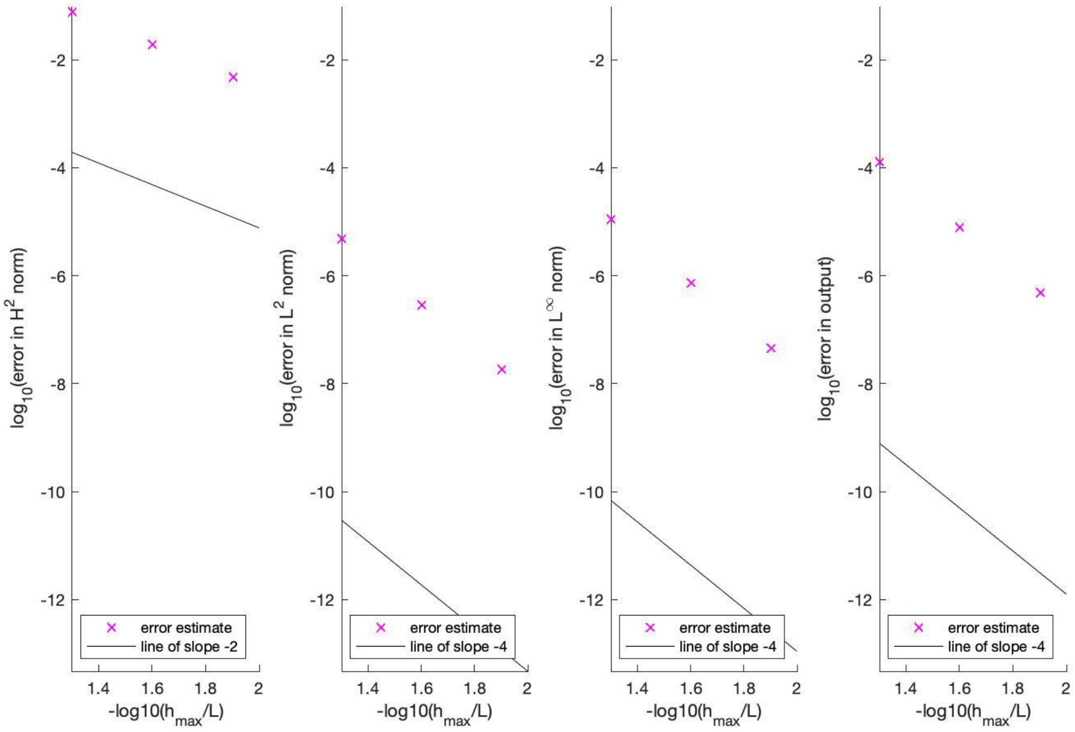


Figure 5: Error plots for the fundamental frequency mode of an F5 pitch

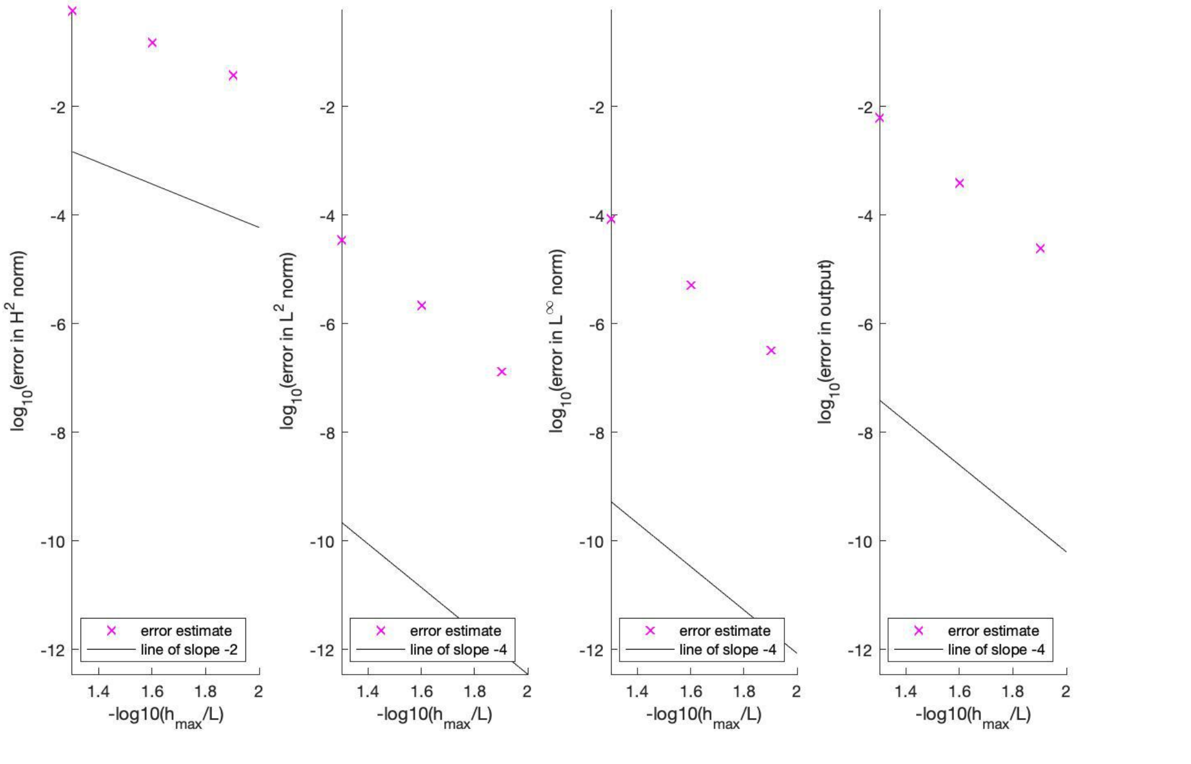


Figure 6: Error plots for the fundamental frequency mode of an F5 pitch

Second, we analyze the error for the same frequency, an F4 pitch with a frequency of 698.46 Hz, but with a different frequency ratio of 4 or "double-octave". The error in $f_d^{(3)}$, and $f_d^{(4)}$ were found to be 1.1717e-05 and 3.9173e-06 respectively. This is slightly higher than the error induced by the lower target frequency plot. This is expected as a higher frequency means there are steeper gradients which are harder to approximate or fit with polynomials.

The code is based on the slender beams model where L >> H. Since H_{maxd} is the same for all xylophone bars, the validity of the model depends on the value of L is inversely proportional to the target frequency, this means that for higher frequencies, the slender beam model might not be valid anymore.

We can incoprorate in our xylophone model the effect of the support strings which are thresaded through the two holes in each xylophone bar. The bar is modelled as a beam of length L with a lumped Hookean spring attached to the right end. The boundary conditions are the following:

$$\begin{aligned} At\ the\ left\ end\ \ x = 0,\ u_{xx} = u_{xxx} = 0\ (free) \\ At\ the\ right\ end\ \ x = L,\ u_{xx} = 0\ (momeent\ free), \\ -(EIu_{xx})_x = -k_s u\ (springforce) \end{aligned}$$

The spring term introduces a "Robin" boundary condition. We can adjust the formulation by adding the term $\frac{1}{2}k_s w^2(L)$ to the energy functional $\Pi(w)$. The new stiffnes matrix is given by:

$$\tilde{A_{i,j}} = \int_0^L EI \frac{d^2\phi_i}{dx^2}\frac{d^2\phi_i}{dx^2}dx \; + \; k_s\phi_i(L)\phi_j(L), \; 1 \le i,j \le 2 \cdot n_{node};$$

where:

$$A^N_{i,j} = \int_0^L EI \frac{d^2\phi_i}{dx^2}\frac{d^2\phi_i}{dx^2}dx$$

A^N is formed by direct stiffness summation; where $\tilde{A}$ incorporates the natural boundary conditions and A unincorporates the essential boundary conditions. In this case, where all the boundary condiitons are natural, A = $\tilde{A}$.

$$\tilde{A} = A^N, \tilde{A_{1,1}} = \tilde{A_{n_{el}+1,n_{el}+1}} + \gamma_2$$

Implementation:

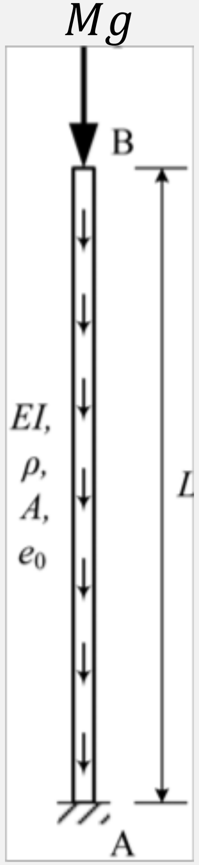
```
%% impose_boundary_cond.m
gam_Gamma2 = probdef.gam_Gamma2;
rightside_node = ttomap_fcn(n_el0+1,1);

if(Dir(1,2) == true)
    bEnodes = [bEnodes,rightside_node];
    uDir = [uDir,u_Gamma2(1)];
    n = n - 1;
else
    F(rightside_node) = F(rightside_node) + f_Gamma2(1);
    A(rightside_node) = A(rightside_node,rightside_node) + gam_Gamma2(1);
end
```

and

```
%% library_of_models.m
302  probdef.gam_Gamma2 = [+ks; 0];
```

BUCKLING EIGEN PROBLEM

SELF BUCKLING



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

$u=u_x=0$ at x = 0

$u_{xx}=(R^4u_{xx})_x=0$ at x = 1

And Normalization of u

OPTIMIZATION PROBLEM

- Optimizing for the Radius Function
  - Let R(x) = $\sqrt{1 + G(x)}$ for G(x) > -1
  - Parametrization:
    - $G^{shape\ family}(x, p_1, p_2)$
  - Constraints:

a) Fixed Volume: $V_d = \pi \int_0^{L_d} R_d^2\,(\mathrm{x_d})\mathrm{dx_d}$ → $\int_0^1 G(x)\,dx = 0$

b) Minimum Radius: $1 + G(x) \geq R_{min}$

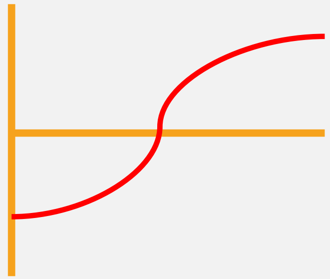
c) Smooth G(x): $|G'(x)| < S_{max}$

OPTIMIZATION PROBLEM

- Objective:
  - Maximize L_d subject to Control Volume, Material constraint, CS:
    1. Minimize γ_c over p_1: γ_c^{opt}
    2. Choose L_d^{opt} = $\left(\frac{\gamma_c^{opt} E_d\, V_d}{4\,\pi\,\rho\, g}\right)^{1/4}$
- Figure of Merit: $\frac{L_d^{opt}}{L_{d,cyl}^{opt}}$ (for a fixed volume) = $\left(\frac{\gamma_c^{opt}}{\gamma_{c,,cyl}^{opt}}\right)^{1/4}$

CHOOSING SHAPE FUNCTIONS

- Choose a set of G(x) basis functions to satisfy constraints
  - $\left\{\begin{array}{c}\text{Odd}\\ \text{Symmetric}\end{array}\right.$ By construction → $\int_0^1 G(x)\,dx = 0$
  - Choice of functions:
    - Polynomial
    - Hyperbolic
  - Control parameters:
    - p_1
    - $p_2\,, width\ of\ transition\ zone$



FE METHOD FOR SELF BUCKLING

$$\underline{A}\ \underline{u_h^0} = \lambda_h \underline{K^{ax}}\underline{u_h^0}$$

$$\underline{u_{h^0}}$$

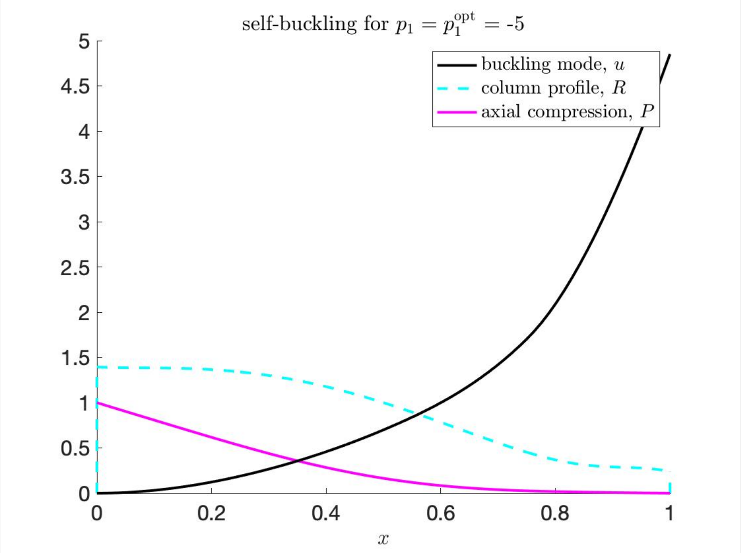
$$\backslash\underline{u_h} = [0;0;\ \cdots]$$

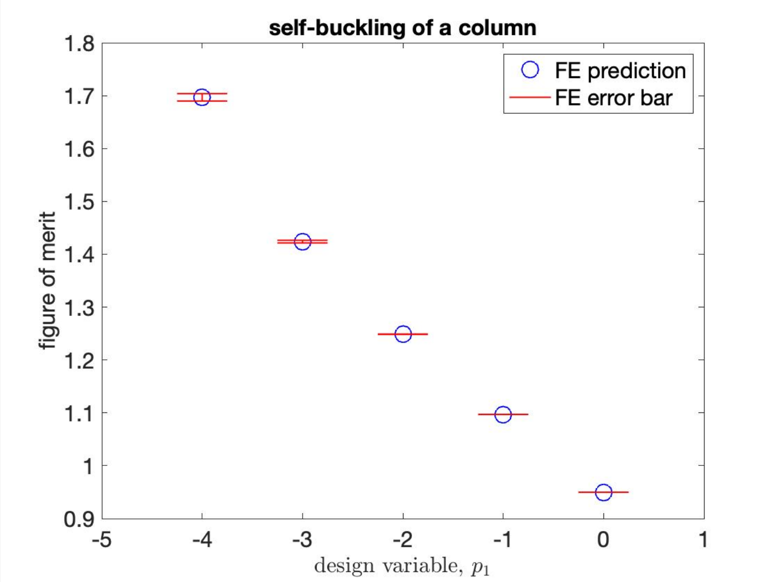
$$\begin{cases} \underline{A} = \underline{\tilde{A}} \\ \underline{K^{ax}} = \underline{\widetilde{K^{ax}}} \end{cases}$$ → Rows and columns 1 & 2 removed

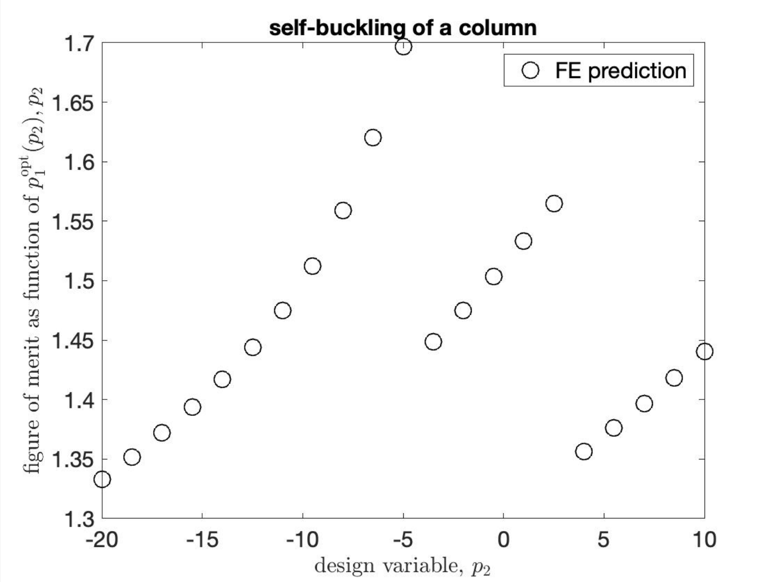
Imposing Boundary condition

$$\widetilde{A_{ij}} = A_{ij}^N = \int_0^1 R^4(x)\frac{\partial^2\phi_i}{\partial x^2}\frac{\partial^2\phi_j}{\partial x^2}dx$$

RESULTS



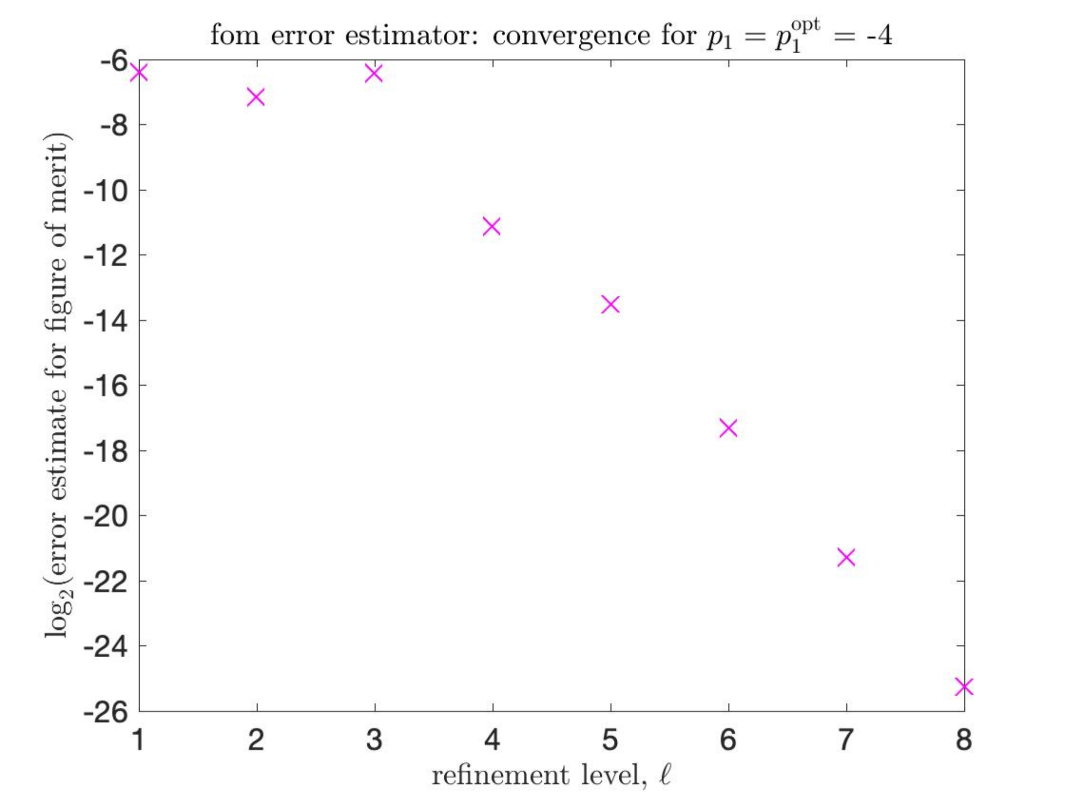




$$G(x, p_1, p_2) = p_1(3(x - 0.5)^{11} + 5(x - 0.5)^5 - 3(x - 0.5)^3 + x - 0.5) + p_2(\tanh(x - 0.5) - (x - 0.5))$$

FOM = 1.69

ERROR ESTIMATES



A priori Estimate

$$\lambda_h - \lambda \sim C_u \left(\frac{h}{h_0}\right)^4 \; as \; h \to 0$$

$$2^{-4l} = \left(\frac{1}{16}\right)^l$$

A Posteriori Estimate

$$\lambda_{h/2} - \lambda \sim \Delta_{\frac{h}{2}}^{\lambda} \; as \; h \to 0$$

$$\log_2 \Delta_h^{\lambda} \sim \log_2 C_u - 4l \; as \; h \to 0$$

Order of convergence = 4