MARGARET M BERTONI

FINITE ELEMENT METHODS FOR MECHANICAL ENGINEERS

MAY 16TH, 2019

MIT

MECHANICAL ENGINEERING

2.S976

TABLE OF CONTENTS

ABSTRACT

Chapter 1 provides a discussion of the Rayleigh-Ritz Method. Two examples were explored: Model I, which demonstrates Neumann/Robin conditions, and Model II, which demonstrates Dirichlet-Neumann/Robin conditions. For both models, the energy functionals were determined and the MATLAB program RR_2S_sver was modified to run the Rayleigh-Ritz approximation. Two basis functions were encoded, one with the exact function and one with constant, linear, and quadratic functions encoded. Four different tests were performed to confirm the proper implementation of the codes.

Chapter 2 provides a discussion of the Finite Element Method for 1D 2^{nd}-Order SPD Boundary Value Problems. Three examples were explored: Ch1_Model_I, which corresponds to Model I from Chapter I, Ch1_Model_II, which corresponds to Model II from Chapter II, and Ch2_Model_mine, which was developed for the chapter to demonstrate Neumann/Robin conditions on both sides of a wall. Two pieces of code had to be modified to impose the boundary conditions of the models and create the matrices needed to find the matrix of FE coefficients. The accuracy and verification of numerical specification was investigated using nine FE meshes on Ch1_Model_II.

Chapter 3 provides a discussion on the Finite Difference-Finite Element (FD-FE) Method for the 1D Heat Equation. Two examples were explored: a semi-infinite fin and a burger. The FD-FE Method utilizes the FE method discussed in previous chapters to capture changes in space in conjunction with the FD method which captures changes in time. The semi-infinite fin model, called semiinf_plus, was used to verify the proper implementation of the FD-FE Method. The burger model was more involved than the semiinf_plus model, because it involves three stages to model the flipping of a burger: the pre-flip, the post-flip, and the repose stages.

Chapter 4 provides a discussion on the Finite Element Method for 1D 4^{th}-Order SPD BVPs for beam bending. Three examples were explored: the design of a xylophone bar, a Caresta Test, and a bar with a Hookean spring attached. The FE Method finds the vibration modes of a beam. In the case of the xylophone problem, the FE Method is used to design the profile of a xylophone bar to achieve a specified note, the fundamental mode, and tuning, indicated by the ratio of the harmonic to the fundamental. The program used for the

xylophone design, xylo_bar_design3 was amended to specify the placement of the support holes to not interfere with the fundamental mode of vibration. To verify that xylo_bar_design3 was able to find the correct frequencies and bar length, its driver, xyloDesignDriver, was modified to run the Caresta Test. The calculated error estimates, modeling error, mesh coarseness, and relative error size were discussed for the xylophone bar tuning. Lastly, a potential modification was discussed to enable the study of a free beam with a Hookean spring on one end.

Chapter 5 provides a discussion of the Finite Element Method for 1D 4^{th}-Order BVPs for Self-Buckling. Chapter 5 is presented in a slide deck format. The FE Method is used to find the critical loading parameter for a given structure. A design challenge was presented: to design the tallest structure, subject to fixed volume, minimum radius, and gradual variation constraints, that would not self-buckle. Candidate G functions were devised and tested with the MATLAB program driver_selfbuckling_sver to determine the best design based on the highest Figure of Merit (FOM), a length ratio of our optimized length compared to that of the base cylinder. The design selected yielded an FOM of 1.6749. Error convergence rates were discussed to verify trust in the design.

REFERENCES AND NOTES

The following work makes several references to course notes and project guidelines from the 2.S976 course offered in the Spring 2019 by Professor Anthony Patera at MIT. Those materials are available on the 2.S976 Stellar page. In Chapter 4 a Caresta Test is referenced. That test is derived from the document “Vibrations of a Free-Free Beam” by Mauro Caresta, which is also available on the 2.S976 Stellar page. Additionally, figure numbers as referenced in the text are local to the chapter in which they reside.

CHAPTER 1 THE RAYLEIGH-RITZ METHOD

1) Chapter Summary

In Chapter 1, I discuss the Rayleigh-Ritz Method through two examples, Model I and II. Model I demonstrates Neumann/Robin conditions. Model II demonstrates Dirichlet-Neumann/Robin conditions. The first step was determining the energy functionals for both models. This was done by comparing the specific equations for Model I and Model II with the generalized Neumann/Robin and Dirichlet-Neumann/Robin equations, respectively. The next step was to determine the contents of the matrices that would determine $\underline{\alpha}^{RR}$ and $\underline{\alpha}^{\widetilde{RR}}$ for Model I and Model II, respectively.

The final step was modifying the RR_2S_sver() program to make the RR_2S_sver_Model_1() and RR_2S_sver_Model_2() programs to calculate the Rayleigh-Ritz Approximation for Model I and Model II, respectively. In these programs I implemented two basis functions: 'exactinclude' and 'constlinquad'. For both models, the constlinquad basis comprised of constant, linear, and quadratic ψ functions. For both models, exactinclude was made of two ψ functions. For Model I first ψ encodes the exact solution, and the second ψ encodes a linear function. For Model II first ψ encodes the exact solution normalized by $u_{\Gamma 1}$, the imposed temperature at x=0, and the second ψ encodes a linear function. For both models, I discuss how four different tests, combined with my understanding of the Rayleigh-Ritz Method, give me confidence that I have properly implemented my codes.

2) Method I

a) Energy Functional

Upon inspecting the equations and boundary conditions of Model I, I determined that Model I had Neumann/Robin conditions. By comparing side-by-side the generalized Neumann/Robin equations and the Model I equations, I was able to replace the generalized terms in the energy functional formula with those specific to Model I. This resulted in the following energy functional:

$$\pi(w) = \frac{1}{2}\int_0^L k\pi R_0^2 \left(1+\frac{\beta x}{L}\right)^2 \left(\frac{dw}{dx}\right)^2 dx + \frac{1}{2}\pi R_0^2(1+\beta)^2\eta_2 w^2(L) - w(0)q_1\pi R_0^2 - w(L)\pi R_0^2(1+\beta)^2\eta_2 u_\infty$$

The energy functional $\pi(w)$ must be minimized over the space X which is characterized by the following:

$\int_0^L w^2 dx \ < \infty$ and $\int_0^L \left(\frac{dw}{dx}\right)^2 dx \ < \infty$.

b) System of Equations $\underline{\boldsymbol{B}}\underline{\boldsymbol{\alpha}}^{\boldsymbol{RR}} = \underline{\boldsymbol{G}}$

$$B_{ij} = \int_0^L k\pi R_0^2 \left(1 + \frac{\beta x}{L}\right)^2 \frac{d\psi_i}{dx}\frac{d\psi_j}{dx} dx + \ \pi R_0^2 (1+\beta)^2 \eta_2 \psi_i(L)\psi_j(L)$$

$$G_i = q_i \pi R_0^2 \psi_i(0) + \pi R_0^2 (1+\beta)^2 \eta_2 u_\infty \psi_i(L)$$

c) Discussion

i) Exactinclude

For a given set of ψ functions, the Rayleigh-Ritz formula finds a linear combination of those functions where the value of the energy functional π is lower than that of any other linear combination of those functions. This ensures that the selected combination has the smallest error in the E_{III} norm. Exactinclude has two basis functions: ψ_1 and ψ_2. ψ_1 is equal to the exact solution to the equations and boundary conditions that govern Model I. ψ_2 is a linear function of x. If the code has been implemented properly, RR_2S_sver_Model_1('exactinclude',beta) should produce a linear combination of 1* ψ_1+0* ψ_2, because any contribution from ψ_2 would increase the error between the Rayleigh-Ritz approximation and the exact solution. Figures 1 and 2 show the results of the Rayleigh-Ritz Approximation using the exactinclude basis functions and two different values of β. Both indicate that there is no contribution from ψ_2. The only contribution is from ψ_1 which corresponds to the exact solution.

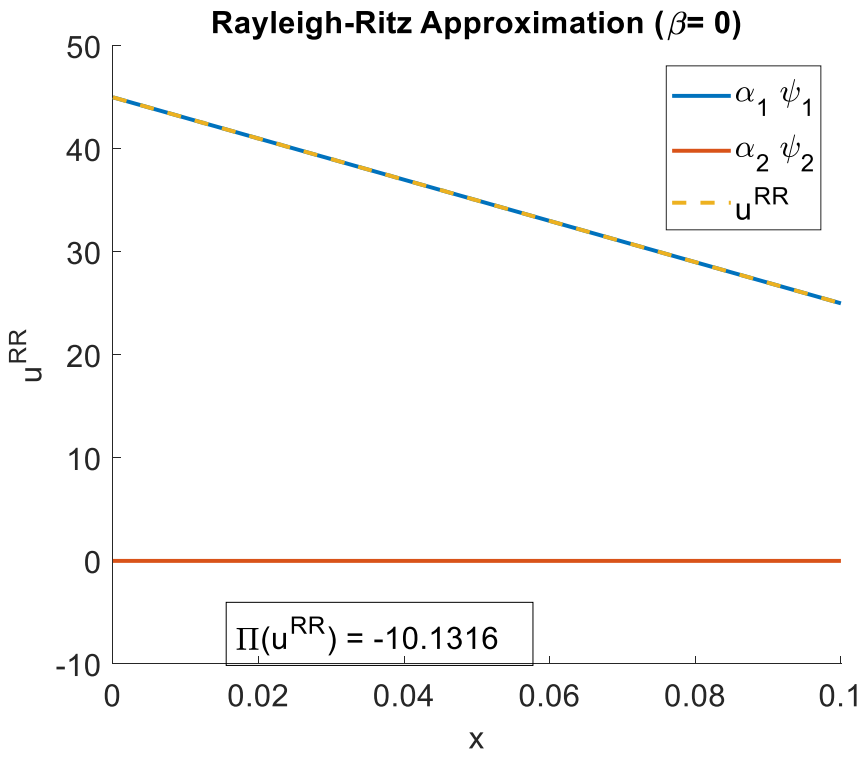


Figure 1: Model I Rayleigh-Ritz Approximation using exactinclude basis functions and β=0

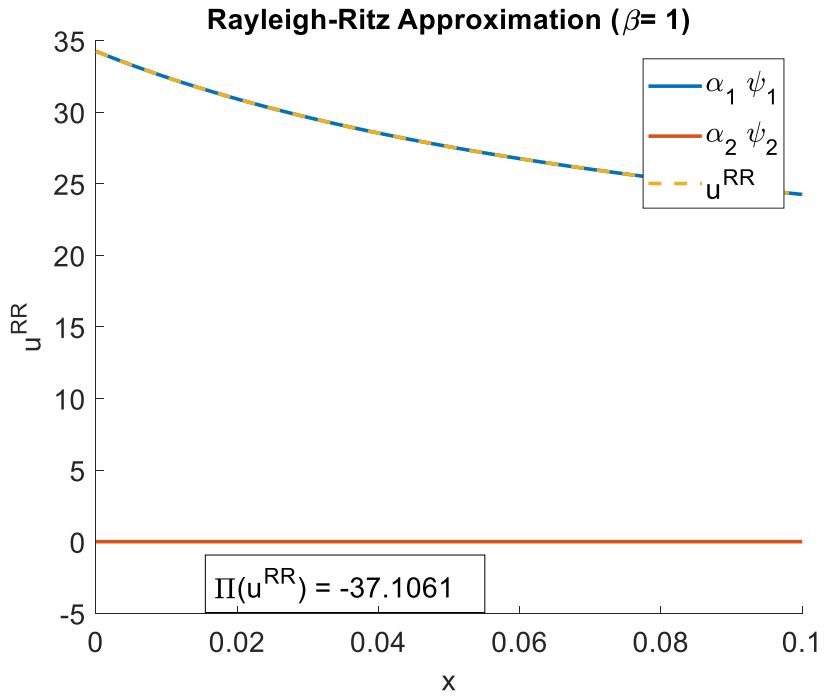


Figure 2: Model I Rayleigh-Ritz Approximation using exactinclude basis functions and β=1

ii) Constlinquad for different n^{RR} values

To confirm that my codes were working for the constlinquad basis functions, I tested RR_2S_sver_Model_1() using β=0 with two different values for n^{RR}. Based on the exact solution to Model I given β=0, I knew that the exact solution u would be a linear function with a non-zero y-intercept. Therefore, with the constlinquad basis, I knew I should be able to approximate the exact solution with good accuracy with only n^{RR}=2. Even if the number of ψs was increased beyond two, the Rayleigh-Ritz recipe should churn out the same linear combination as when n^{RR}=2 because no additional curvature is needed to capture the exact solution. This is demonstrated in Figures 3 and 4. For β=0, the solutions with n^{RR}=2 and n^{RR}=3 have the same value of the energy functional, meaning that they have the same amount of error in the E_{III} norm. Figure 4 also shows that ψ_3 has no contribution to the Rayleigh-Ritz Approximation. This confirms my belief that the code is properly functioning to find the best linear combination of basis functions.

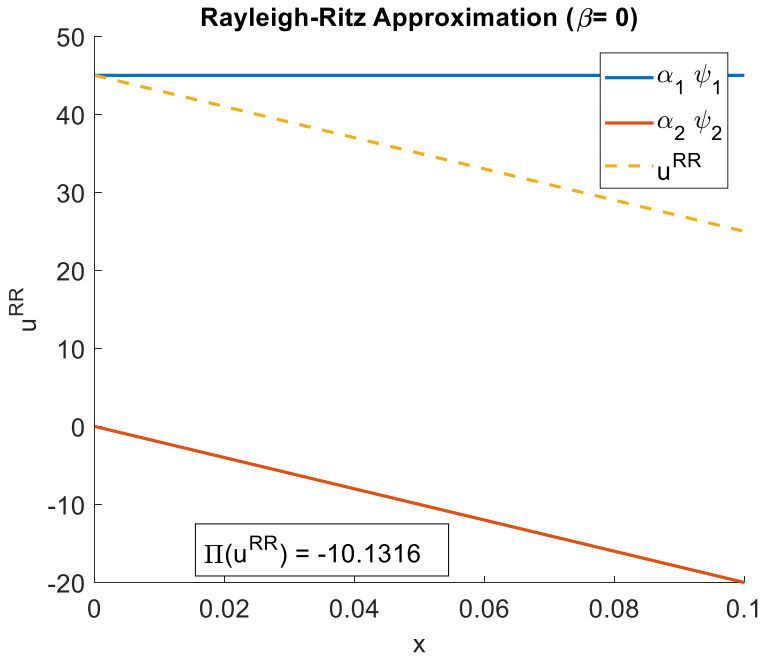


Figure 3: Model I Rayleigh-Ritz Approximation using constlinquad basis functions, β=0, and n^{RR}=2

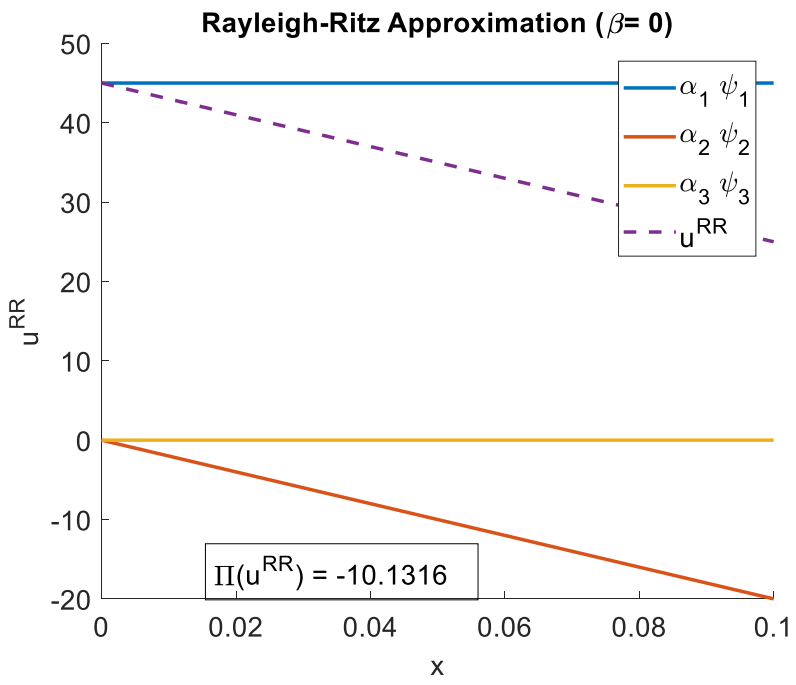


Figure 4: Model I Rayleigh-Ritz Approximation using constlinquad basis functions, β=0, and n^{RR}=3

iii) Comparison of Exactinclude and Constlinquad

By the Minimization Proposition the Rayleigh-Ritz Approximation should find the linear combination of Ψs with the lowest value of the energy functional π. Based on the exact solution to Model I given β=0, I knew that the exact solution u would be a linear function with a non-zero y-intercept. From Figure 1, I knew that exactinclude would be able to find the exact solution, and as shown in Figure 3, I knew that constlinquad with n^{RR}=2 would also be able to find the exact solution. If the code is working correctly, the π value from the exactinclude solution should be equivalent to the value from the constlinquad with n^{RR}=2 solution given that they find the same solution. As shown in Figure 1 and Figure 5, both solutions have a π value of -10.1316. As an additional check, I compared this π value with that of a known worse solution: constlinquad with n^{RR}=1. I knew this would return a less accurate solution because constlinquad with n^{RR}=1 only contains ψ_1 which only has a constant term and does not capture the linearity of the exact solution. If the code is working correctly, then the π value for the solution of constlinquad with n^{RR}=1 would be higher than that

of exactinclude because it is less accurate. Figure 6 shows that the π value for the solution of constlinquad with n^{RR}=1 is -9.8175 which is greater than that of the solution for the exactinclude basis functions. These observations taken together give me confidence that the code is working properly for both basis functions.

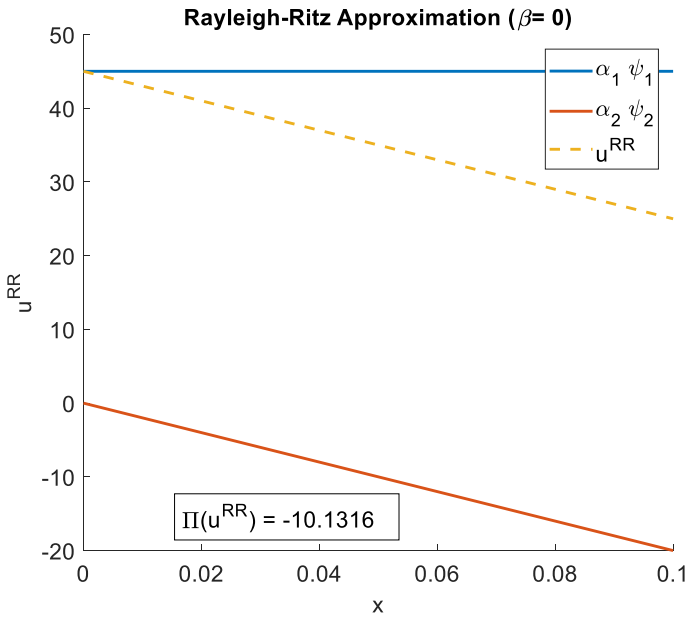


Figure 5: Model I Rayleigh-Ritz Approximation using constlinquad basis functions, β=0, and n^{RR}=2

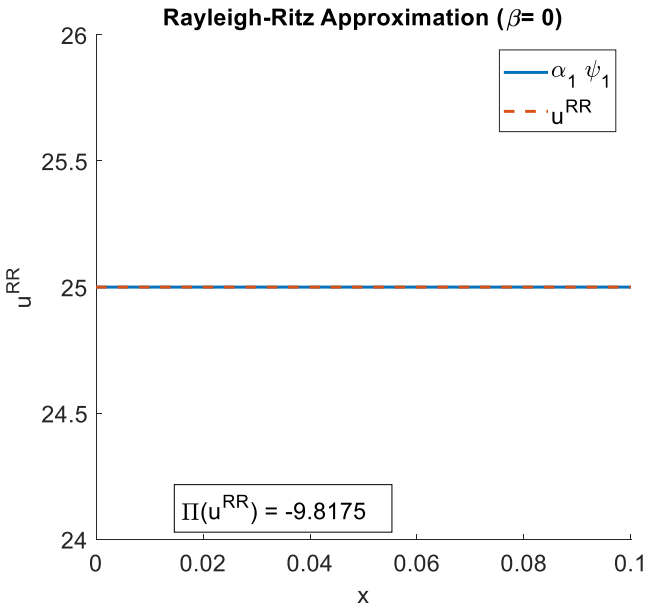


Figure 6: Model I Rayleigh-Ritz Approximation using constlinquad basis functions, β=0, and n^{RR}=1

iv) Constlinquad for different values of β

The constlinquad basis functions are constructed to allow one to capture more curvature with the more Ψs used. Therefore, the trend in relative output error with changing values of β depends on the n^{RR} value. When n^{RR} =2, we can capture linear behavior and non-zero intercepts. That describes the behavior of the exact solution when β=0 as shown in Figure 7 where the relative error is on the order of 10^{-15}. As β is increased, the exact solution becomes more quadratic. Therefore, if you hold n^{RR}=2, as you increase β the Rayleigh-Ritz Approximation cannot capture that curvature, so the error increases as shown in Figures 8 and 9. However when n^{RR} =1 and β is increased, the relative error decreases as shown in Figures 10 and 11.

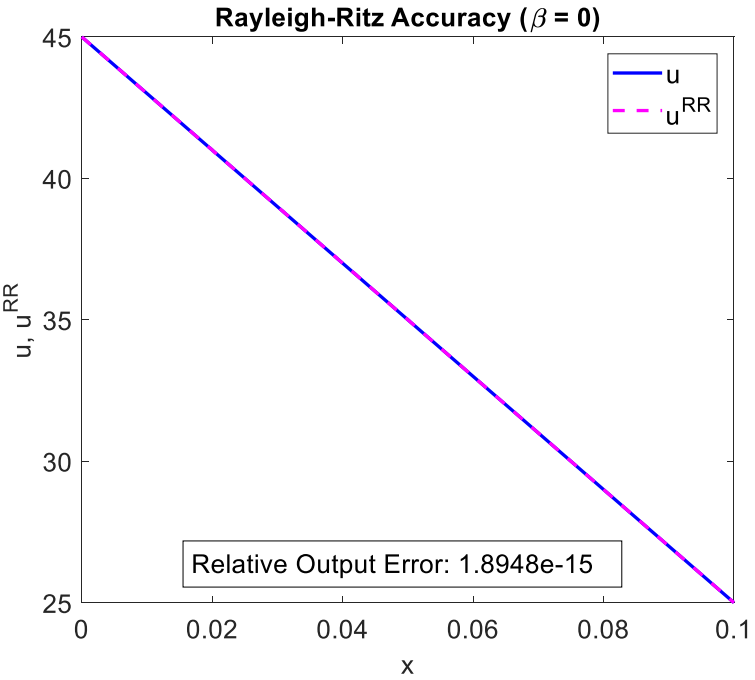


Figure 7: Model I Rayleigh-Ritz Approximation using constlinquad basis functions, β=0, and n^{RR}=2

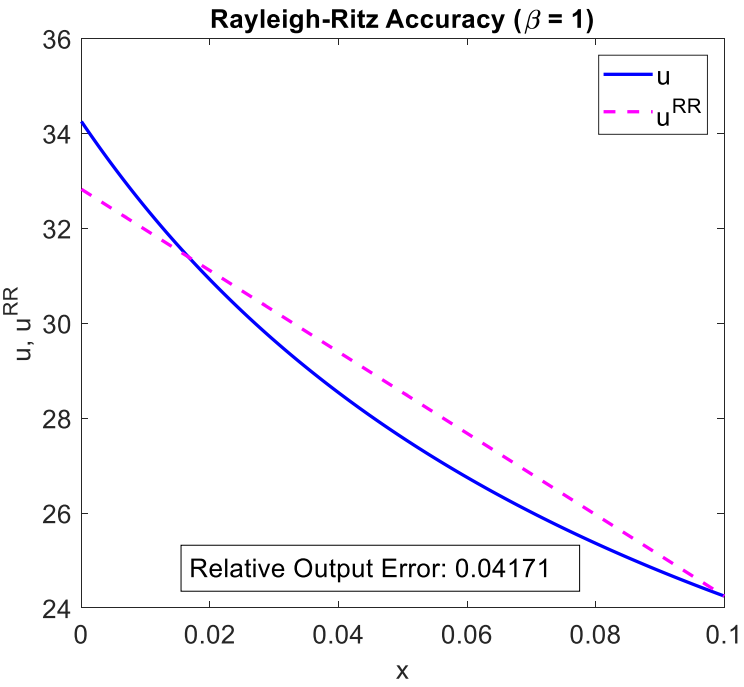


Figure 8: Model I Rayleigh-Ritz Approximation using constlinquad basis functions, β=1, and n^{RR}=2

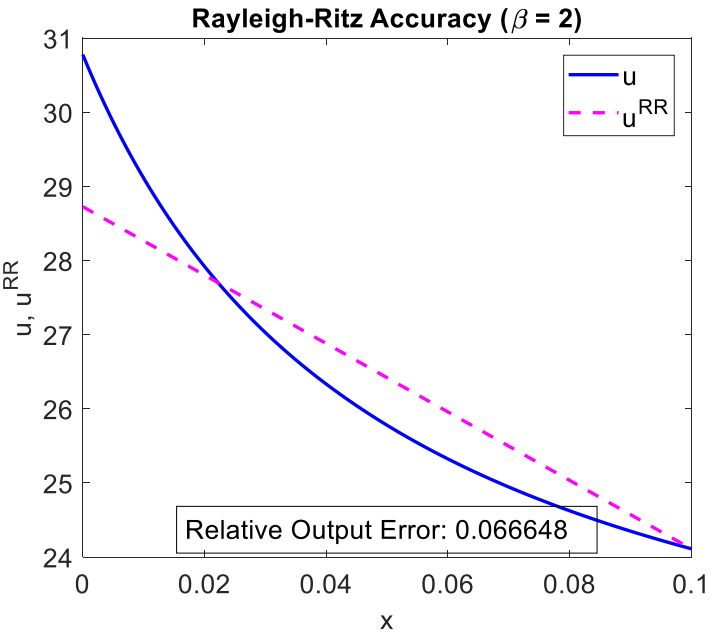


Figure 9: Model I Rayleigh-Ritz Approximation using constlinquad basis functions, β=2, and n^{RR}=2

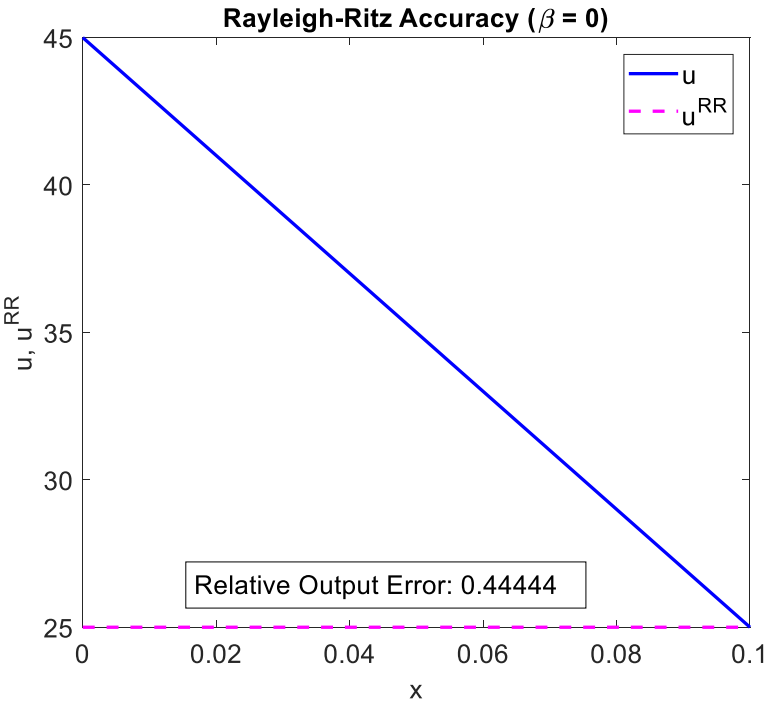


Figure 10: Model I Rayleigh-Ritz Approximation using constlinquad basis functions, β=0, and n^{RR}=1

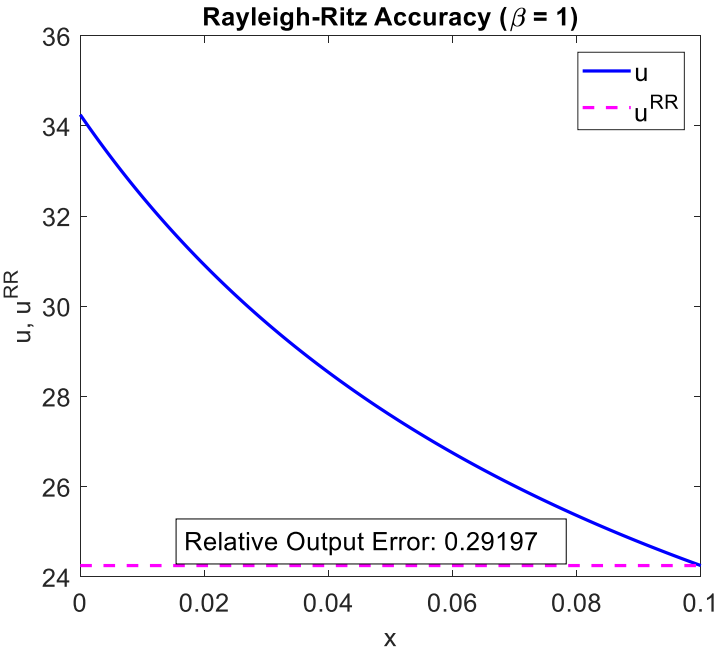


Figure 11: Model I Rayleigh-Ritz Approximation using constlinquad basis functions, β=1, and n^{RR}=1

3) Method II

a) Energy Functional

Upon inspecting the equations and boundary conditions of Model II, I determined that Model II had Dirichlet-Neumann/Robin conditions. By comparing side-by-side the generalized Dirichlet-Neumann/Robin equations and the Model II equations, I was able to replace the generalized terms in the energy functional formula with those specific to Model II. This resulted in the following energy functional:

$$\pi(w) = \frac{1}{2}\int_0^L \left[kA_{cs}\left(\frac{dw}{dx}\right)^2 + \eta_3 P_{cs} w^2\right] dx - \int_0^L \eta_3 P_{cs} u_\infty w dx$$

The energy functional $\pi(w)$ must be minimized over the space X which is characterized by the following:

$w(0) = u_{\Gamma_1}$, $\int_0^L w^2 dx \ < \infty$, and $\int_0^L \left(\frac{dw}{dx}\right)^2 dx \ < \infty$.

b) System of Equations $\widetilde{\underline{\boldsymbol{B}}}\widetilde{\underline{\boldsymbol{\alpha}}}^{\boldsymbol{RR}} = \widetilde{\underline{\boldsymbol{G}}}$

$$\underline{\tilde{B}}_{ij} = \int_0^L \left[kA_{cs}\frac{d\psi_i}{dx}\frac{d\psi_j}{dx} + \eta_3 P_{cs}\psi_i\psi_j\right] dx$$

$$\underline{\tilde{G}_i} = \int_0^L \eta_3 P_{cs} u_\infty \psi_i dx$$

c) Discussion

i) Exactinclude

For a given set of ψ functions, the Rayleigh-Ritz formula finds a linear combination of those functions where the value of the energy functional π is lower than that of any other linear combination of those functions. This ensures that the selected combination has the smallest error in the E_{III} norm. Exactinclude has two basis functions: ψ_1 and ψ_2 . ψ_1 is equal to the exact solution to the equations governing Model II divided by $u_{\Gamma 1}$, the temperature at x=00. ψ_2 is a linear function of x. If the code has been implemented properly, RR_2S_sver_Model_2('exactinclude', η_3) should produce a linear combination of $u_{\Gamma 1}$* ψ_1+0* ψ_2, because any contribution from ψ_2 would increase the error between the Rayleigh-Ritz approximation and the exact solution. Figures 12 and 13 show the results of the Rayleigh-Ritz Approximation using the exactinclude basis functions and two different values of η_3. Both indicate that there is no contribution from ψ_2. The only non-zero element of the $\underline{\tilde{\alpha}}^{RR}$ matrix is

the one corresponding to the ψ_1 function with a value of 50, which is the given value for $u_{\Gamma 1}$. This gives me confidence that the Model II code has been properly implemented for the exactinclude basis functions.

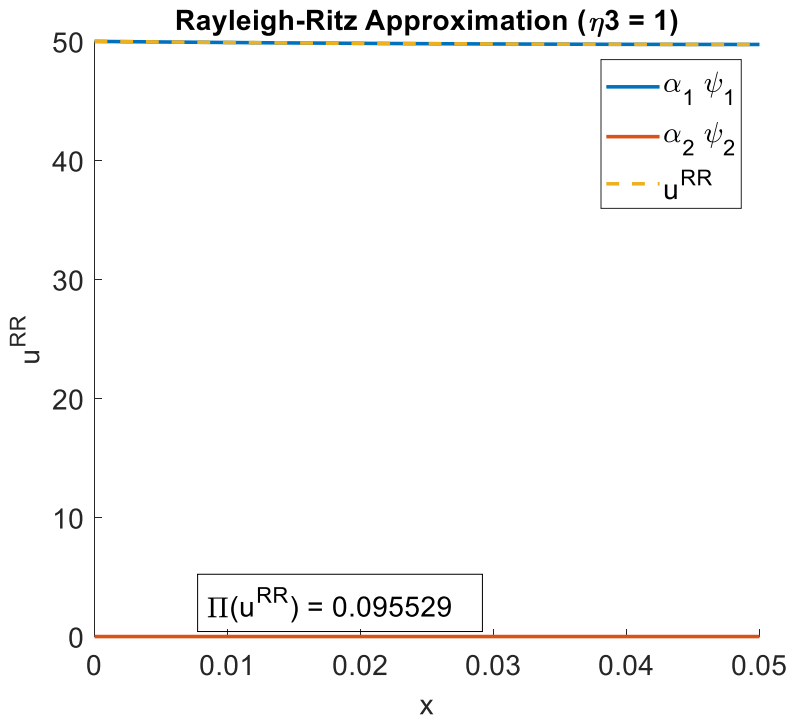


Figure 12: Model II Rayleigh-Ritz Approximation using exactinclude basis functions and η_3=1

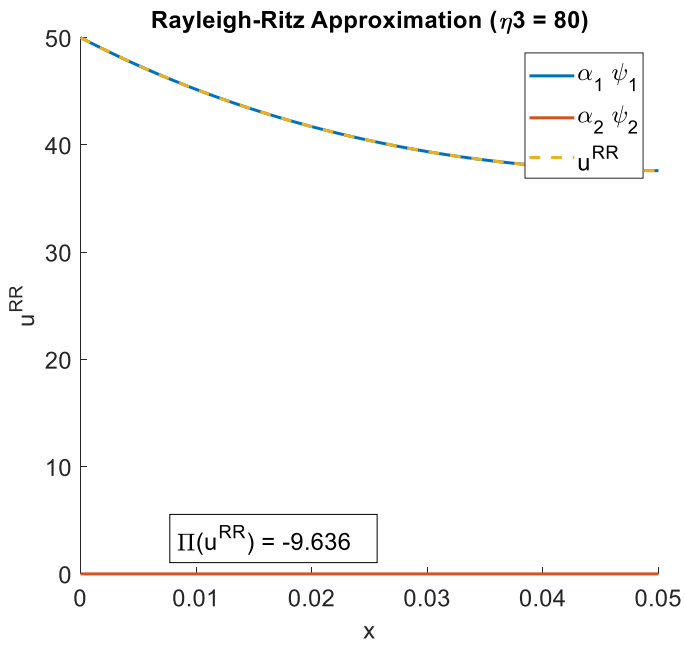


Figure 13: Model II Rayleigh-Ritz Approximation using exactinclude basis functions and η_3=80

ii) Constlinquad for different n^{RR} values

To test if the code for the constlinquad basis was working, I tested RR_2S_sver_Model_2() with η_3=80 and two different values for n^{RR}. As shown in Figure 14, when η_3=80, the exact solution u involves the cosh function which has somewhat quadratic behavior on the interval of interest. The constlinquad basis functions are constructed so that when n^{RR}=2, the basis functions can capture linear behavior with non-zero y-intercepts. When n^{RR}=3, the basis functions can capture quadratic behavior. Therefore, if the code is working, the constlinquad basis function with n^{RR}=3 should produce a more accurate approximation than the constlinquad basis function with n^{RR}=2. Figure 14 shows that when n^{RR}=2 the relative output error is 0.51611 and Figure 15 shows that when n^{RR}=3 the relative output error is 0.071117, confirming my expectation.

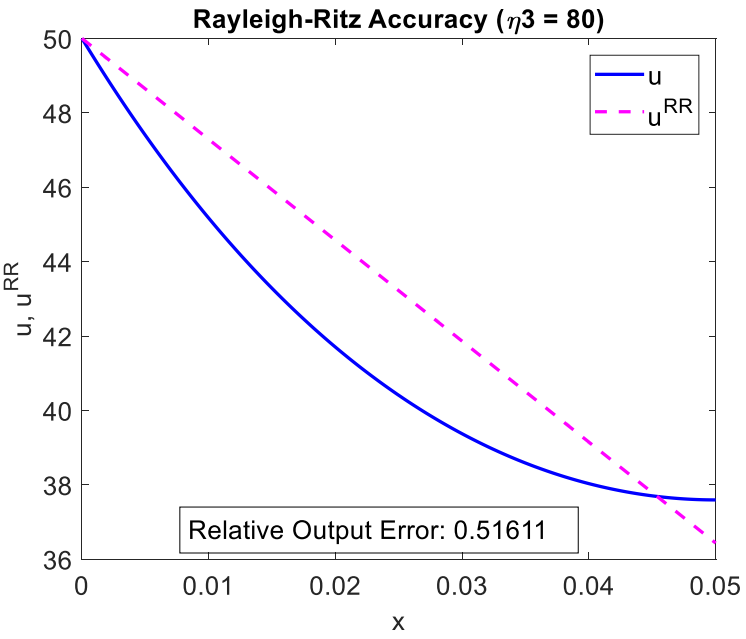


Figure 14: Model II Rayleigh-Ritz Accuracy using constlinquad basis functions, η_3=80 and n^{RR}=2

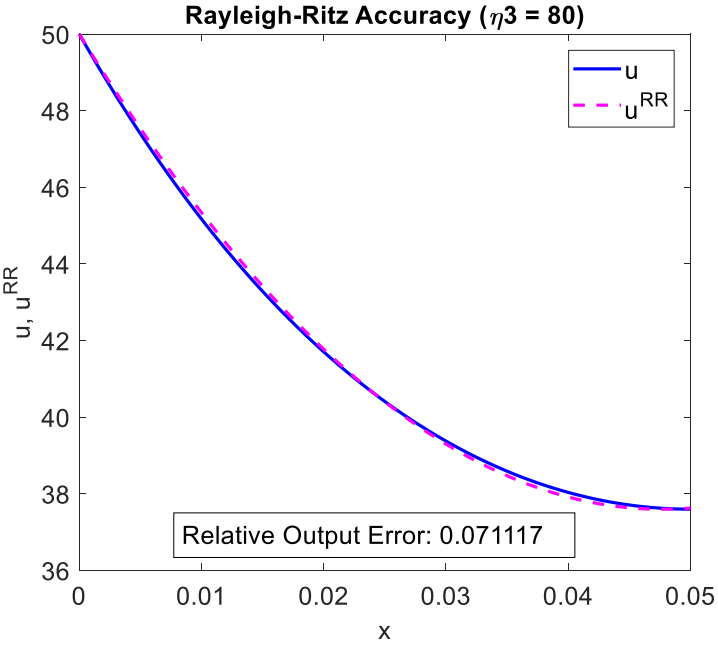


Figure 15: Model II Rayleigh-Ritz Accuracy using constlinquad basis functions, η_3=80 and n^{RR}=3

iii) Comparison of Exactinclude and Constlinquad

By the Minimization Proposition the Rayleigh-Ritz Approximation should find the linear combination of Ψs with the lowest value of the energy functional π. The exact

solution to Model II involves cosh which as shown in Figure 15 has somewhat, but not exactly, quadratic behavior on our interval of interest. Based on this behavior, if the code is working, I expect that the exactinclude Rayleigh-Ritz approximation should be better than the constlinquad Rayleigh-Ritz approximation. Figure 13 shows that the π value for the exactinclude approximation is -9.636. Figure 16 shows that the that the π value for the constlinquad approximation with n^{RR}=3 is -9.6087. The π values are close, with that of exactinclude being slightly lower meaning that it has less error in the E_{III} norm; this is consistent with my expectation.

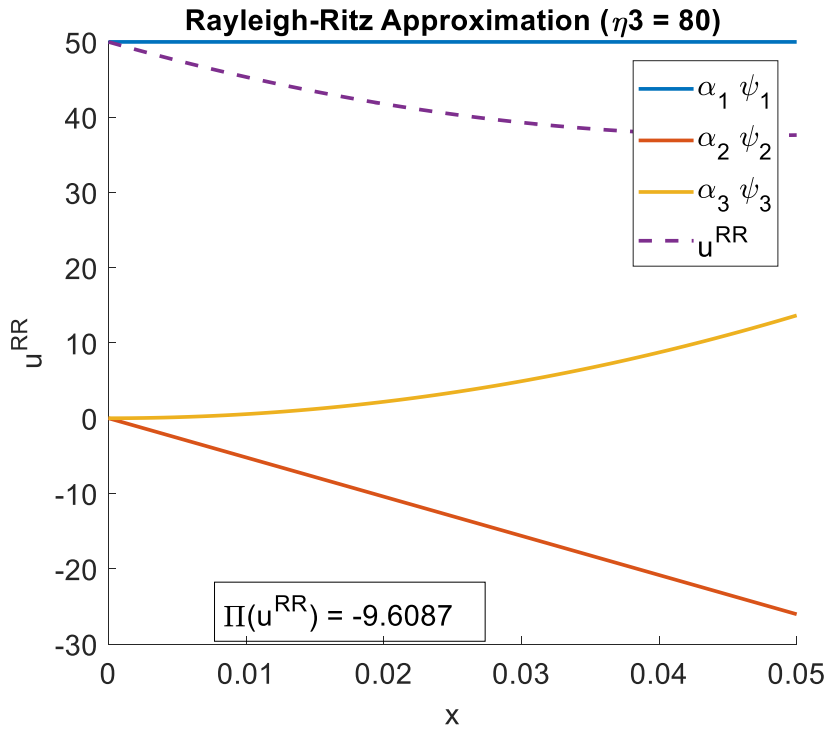


Figure 16: Model II Rayleigh-Ritz Approximation using constlinquad basis functions, η_3=80 and n^{RR}=3

iv) Constlinquad for different values of μ_0

When n^{RR} does not equal 1, as μ_0 is increased, achieved by increasing η_3, the relative error increases as shown in Figures 15 and 17. This occurs because as μ_0 increases, the exact solution becomes harder to approximate using only quadratic terms. When n^{RR}=1, the constlinquad basis can only approximate constant functions. This approximation is poor, and in this case as μ_0 is increased, the relative error stays constant as 1 as shown in Figures 18 and 19.

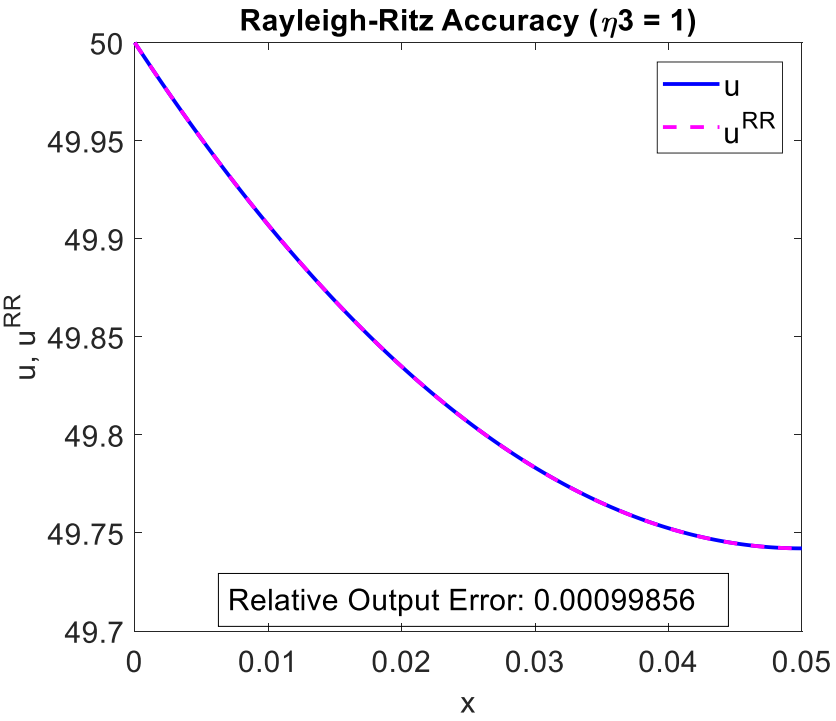


Figure 17: Model II Rayleigh-Ritz Accuracy using constlinquad basis functions, η_3=1 and n^{RR}=3

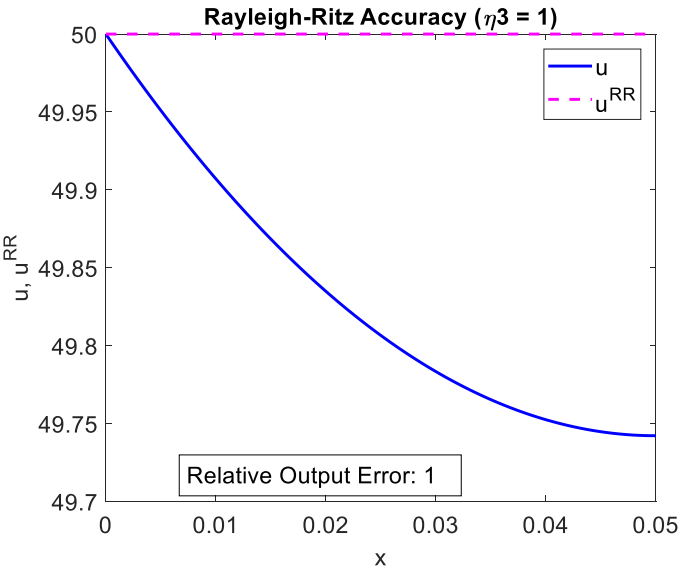


Figure 18: Model II Rayleigh-Ritz Accuracy using constlinquad basis functions, η_3=1 and n^{RR}=1

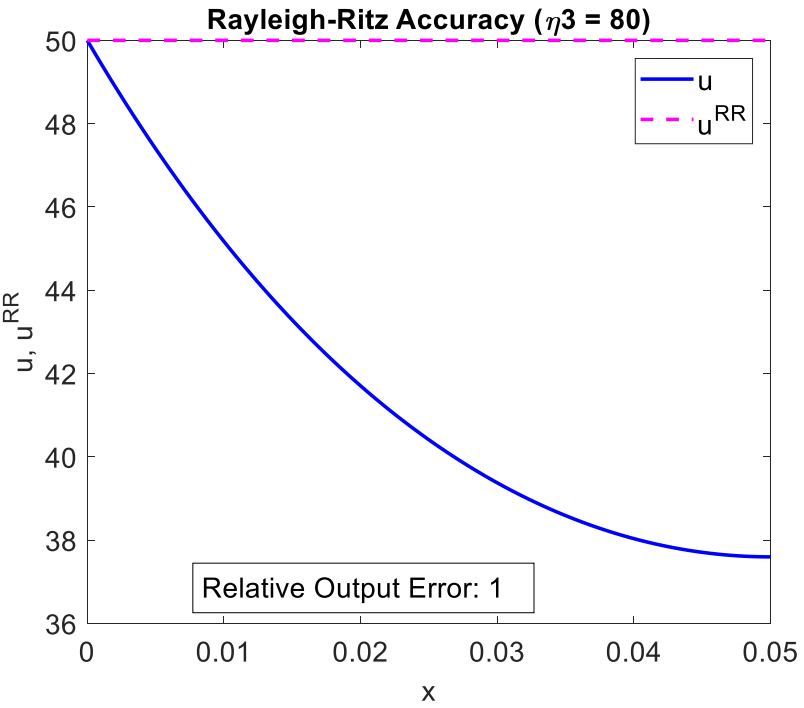


Figure 19: Model II Rayleigh-Ritz Accuracy using constlinquad basis functions, η_3=80 and n^{RR}=1

CHAPTER 2 THE FE METHOD FOR 1D 2ND-ORDER SPD BVPS

1) Chapter Summary

In Chapter 2, I discuss the Finite Element method for 1D 2nd-Order SPD Boundary Value Problems through three examples, Ch1_Model_I, Ch1_Model_II, and Ch2_Model_Mine Ch1_Model_I corresponds to the Model I discussed in Chapter I and demonstrates Neumann/Robin conditions. Ch1_Model_II corresponds to the Model II discussed in Chapter 1 and demonstrates Dirichlet-Neumann/Robin conditions. Ch2_Model_Mine was developed for this chapter. It demonstrates Neumann/Robin conditions on both sides of a wall.

The first step was developing Ch2_Model_Mine and adding it to library_of_models. Run_uniform_refinement also had to be modified in order to run the FE method on this new case. Then, I had to modify the code for form_elem_mat_sver and impose_boundary_cond_sver in order to replace the instructor's code to impose the boundary conditions of the models and create the matrices needed to find the matrix of FE coefficients.

Next, I verified that my modifications were successful. I did this by checking the convergence of u, the exact solution, and u_h, the finite element method solution through convergence plots and visual inspection of the solution plots. I also inspected the matrix of finite element coefficients to insure that it was tri-diagonal as expected. Finally, I investigated accuracy and verification of numerical specification using nine meshes on Ch1_Model_II.

2) Model Summaries

a) Ch1_Model_I

Ch1_Model_I is a model of quasi-1D heat conduction in a conical frustum insulated on the lateral surfaces. This model demonstrates Neumann/Robin conditions. The model is governed by the following equation: $-k\frac{d}{dx}\left(\pi R_0^2\left(1+\beta\frac{x}{L}\right)^2\frac{du}{dx}\right)=0\ in\ \Omega$, and boundary conditions: $k\frac{du}{dx}=-q_1\ on\ \Gamma_1$ and $-k\frac{du}{dx}=\eta_2(u-u_\infty)\ on\ \Gamma_2$. The output is: $s\equiv u(0)$, the temperature on Γ_1.

b) Ch1_Model_II

Ch1_Model_II is a model of a right-cylinder thermal fin with Dirichlet-Neumann/Robin conditions imposed. The model is governed by the following equation:

$-kA_{cs}\frac{d^2u}{dx^2} + \eta_3 P_{cs}(u - u_\infty) = 0\ in\ \Omega$, and boundary conditions: $u = u_{\Gamma_1} on\ \Gamma_1$ and $-k\frac{du}{dx} = 0\ on\ \Gamma_2$. The output is: $s \equiv -k\frac{du}{dx}(x = 0)$, the flux into the fin at the base of the fin.

c) Ch2_Model_Mine

Ch2_Model_Mine is a model of a wall with a constant cross-section and Neumann/Robin conditions on both sides. It is visualized in Figure 1. The model is governed by the following equation: $-\frac{d}{dx}\left(kA_{cs}\frac{du}{dx}\right) = 0\ in\ \Omega$, and boundary conditions: $kA_{cs}\frac{du}{dx} = \gamma_1(u - u_{out})\ on\ \Gamma_1$ and $-kA_{cs}\frac{du}{dx} = \gamma_2(u - u_{in})\ on\ \Gamma_2$. The output is: $s \equiv u(0)$, the temperature on Γ_1.

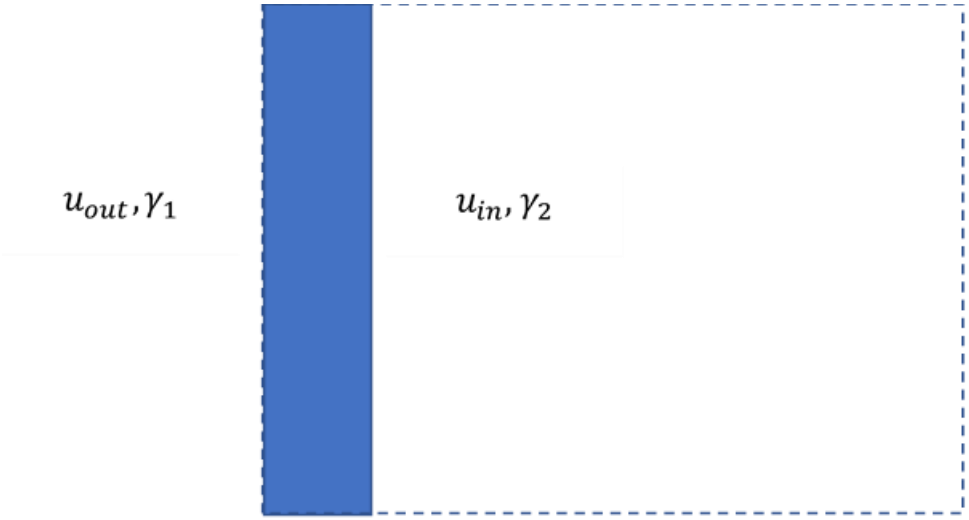


Figure 1 Diagram of Ch2_Model_Mine

3) Summary of the FE Method

a) Ch1_Model_I

For Ch1_Model_I the finite element solution is formulated as: $u_h(x) = \sum_{i=1}^{n} u_{hi}\varphi_i(x)$. The finite element coefficients, represented by matrix $\underline{u_h}$, is found with the following linear system: $\underline{Au_h} = \underline{F}$. Where

$A_{ij} = \int_0^L k\pi R_0^2\left(1 + \frac{\beta x}{L}\right)^2 \frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx}dx + \pi R_0^2(1+\beta)^2\eta_2\,\varphi_i(L)\varphi_j(L),\ 1 \le i,j \le n$ and $F_i = q_1\pi R_0^2\varphi_i(0) + \pi R_0^2(1+\beta)^2\eta_2 u_\infty\varphi_i(L), 1 \le i \le n.$

b) Ch1_Model_II

For Ch1_Model_II the finite element solution is formulated as: $u_h(x) = u_{\Gamma_1}\varphi_1(x) + \sum_{i=2}^{n_{node}} u_{hi}\varphi_i(x)$. The finite element coefficients, represented by matrix $\underline{u}_h$, is found with the following linear system: $\underline{A}\underline{u}^0{}_h = \underline{F} - u_{\Gamma_1}\underline{b}$. Where

$\underline{A} = \underline{\tilde{A}}(2{:}\,end, 2{:}\,end)$, $\underline{b} = \underline{\tilde{A}}(2{:}\,end, 1)$ and $\underline{u}_h = \left[u_{\Gamma_1}; \underline{u}^0{}_h\right]$. The matrices are defined by $\widetilde{A_{iJ}} = \int_0^L kA_{cs}\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx} + \eta_3 P_{cs}\varphi_i\varphi_j dx,\ 1 \le i,j \le n$ and

$\widetilde{F_i} = \int_0^L \eta_3 P_{cs} u_\infty \varphi_i dx,\ 1 \le i \le n.$

c) Ch2_Model_Mine

For Ch2_Model_Mine the finite element solution is formulated as:

$u_h(x) = \sum_{i=1}^{n} u_{hi}\varphi_i(x)$. The finite element coefficients, represented by matrix $\underline{u}_h$, is found with the following linear system: $\underline{A}\underline{u}_h = \underline{F}$. Where

$A_{ij} = \int_0^L kA_{cs}\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx}dx + \gamma_1\varphi_i(0)\varphi_j(0) + \gamma_2\varphi_i(L)\varphi_j(L),\ 1 \le i,j \le n$ and

$F_i = \gamma_1 u_{out}\varphi_i(0) + \gamma_2 u_{in}\varphi_i(L), 1 \le i \le n.$

4) Form_elem_mat_sver

To confirm that I correctly modified the function form_elem_mat_sver, I replaced the lines calling form_elem_mat.p in the FE1d_uniform_refinement code with the lines calling my form_elem_mat_sver. Then I ran run_uniform_refinement on Ch1_Model_II. Figures 2 and 3 show that over the course of eight refinements, nine meshes total, the error converges at the expected rate in all three norms and in the output. This is confirmed visually in Figure 4 by the plots showing both the Finite Element and exact solution for the first and last mesh. A final check is performed by inspected Figure 5, the visualization of the u_h matrix. This confirms that the matrix is tri-diagonal as expected if the code was properly working. With these checks, I am confident that the function form_elem_mat_sver is functioning properly with Ch1_Model_II.

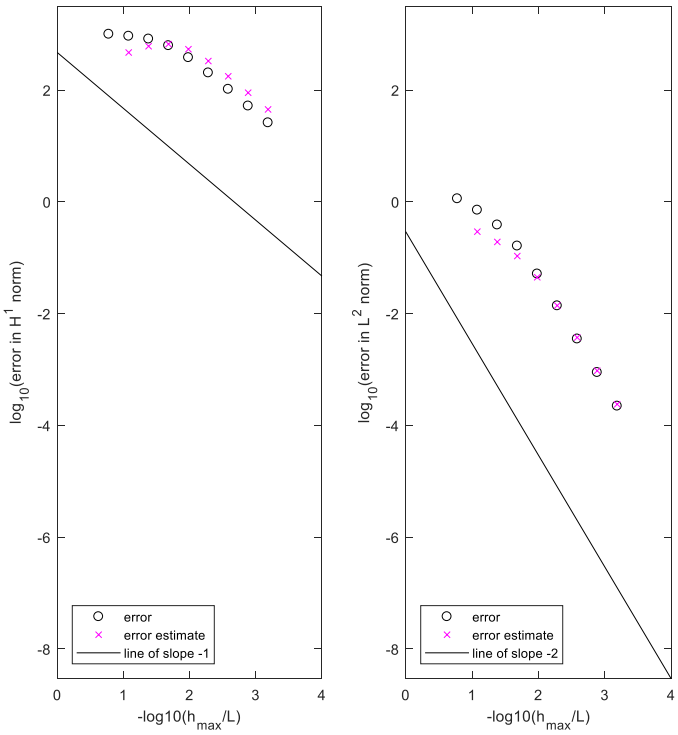


Figure 2 Ch1_Model_II Log-Log Convergence Plots for $H^1(\Omega)$ and $L^2(\Omega)$ Norm Error

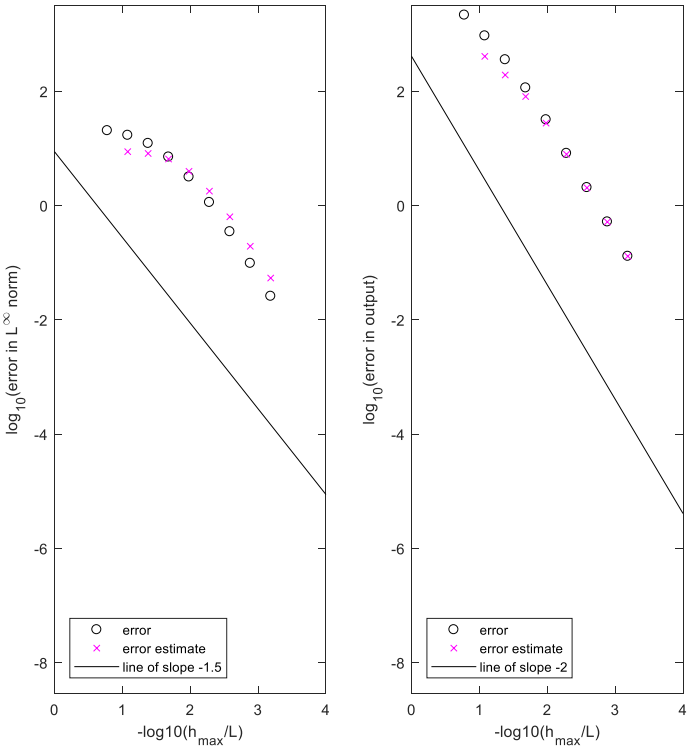


Figure 3 Ch1_Model_II Log-Log Convergence Plots for $L^{inf}(\Omega)$ and Output Error

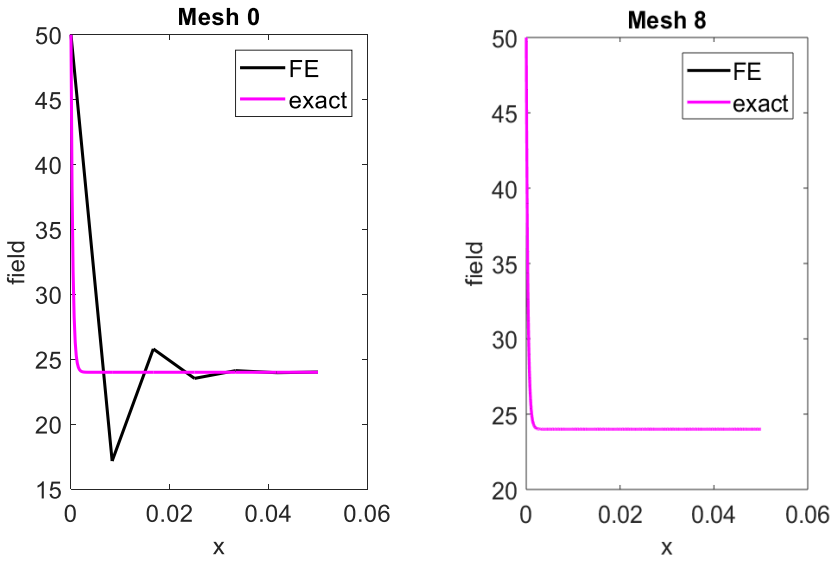


Figure 4 Plot of the Finite Element Method and Exact Solution for Ch1_Model_II for Mesh 0 and Mesh 8

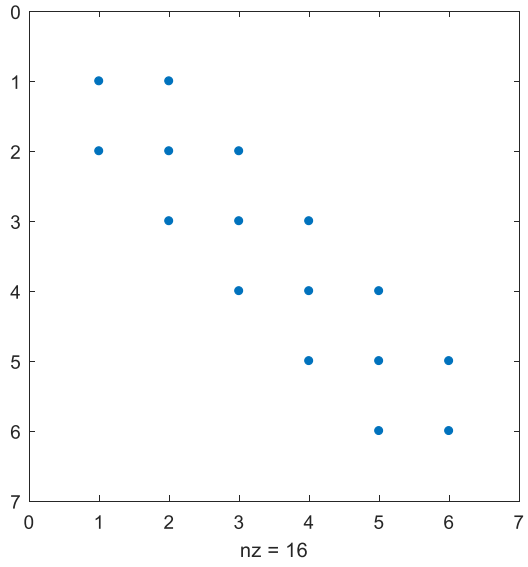


Figure 5 Visualization of the u_h matrix for Ch1_Model_II

5) Additional Model

For the three models previously discussed, each had variables which were zero. For Ch1_Model_1 $\mu(x), f_\Omega \; and \; \gamma_1 = 0$. For Ch1_Model_2 $f_{\Gamma_2} \; and \; \gamma_2 = 0$. For Ch2_Model_Mine $\mu(x) \; and \; f_\Omega = 0$. Therefore, I would want to test an additional model with all non-zero terms to ensure that form_element_mat_sver is fully-functioning. This

model would be defined by N/R-N/R conditions. It would be governed by the equation: $-\frac{d}{dx}\left(\kappa(x)\frac{du}{dx}\right)+\mu(x)u=f_{\Omega}(x) in\ \Omega\ with\ \kappa(x),\mu(x)>0$, and the boundary conditions: $\kappa(x)\frac{du}{dx}=\gamma_1 u-f_{\Gamma_1}\ on\ \Gamma_1$ and $-\kappa(x)\frac{du}{dx}=\gamma_2 u-f_{\Gamma_2}\ on\ \Gamma_1\ with\ \gamma_1,\gamma_2\geq 0$.

6) **Impose_boundary_cond_sver**

To confirm that I correctly modified the function impose_boundary_cond_sver, I replaced the lines calling impose_boundary_cond.p in the FE1d_uniform_refinement code with the lines calling my impose_boundary_cond_sver. Then I ran run_uniform_refinement on Ch1_Model_I and Ch2_Model_Mine. Figures 6 and 7 show that for Ch1_Model_I the error on all norms and the output converges at the expected rate. Figure 8 shows the plots of the Finite Element and exact solutions for the first and last mesh. By visual inspection I can see that there are no erroneous offsets from the exact solution, which leads me to believe that impose_boundary_cond_sver functions properly on Ch1_Model_I. Figures 9 and 10 show that for Ch2_Model_Mine, the error does not converge as expected on any of the norms or the output. This may be caused by error stack up as the mesh is refined. Figure 11 shows that with no refinements the finite element method can find the exact solution, because the exact solution to Ch2_Model_Mine is a linear profile. Despite the lack of convergence, Ch2_Model_Mine provides greater confidence in my implementation of impose_boundary_cond_sver than Ch1_Model_I, because Ch2_Model_Mine has two non-zero gammas, whereas Ch1_Model_I has only on non-zero gamma. With Ch2_Model_Mine I confirm that the code for incorporating both gammas is working.

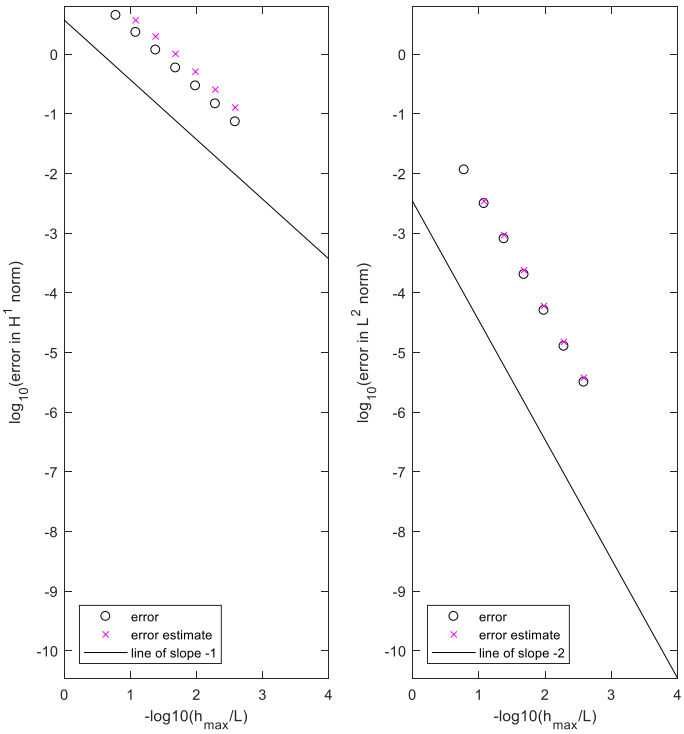


Figure 6 Ch1_Model_I Log-Log Convergence Plots for $H^1(\Omega)$ and $L^2(\Omega)$ Norm Error

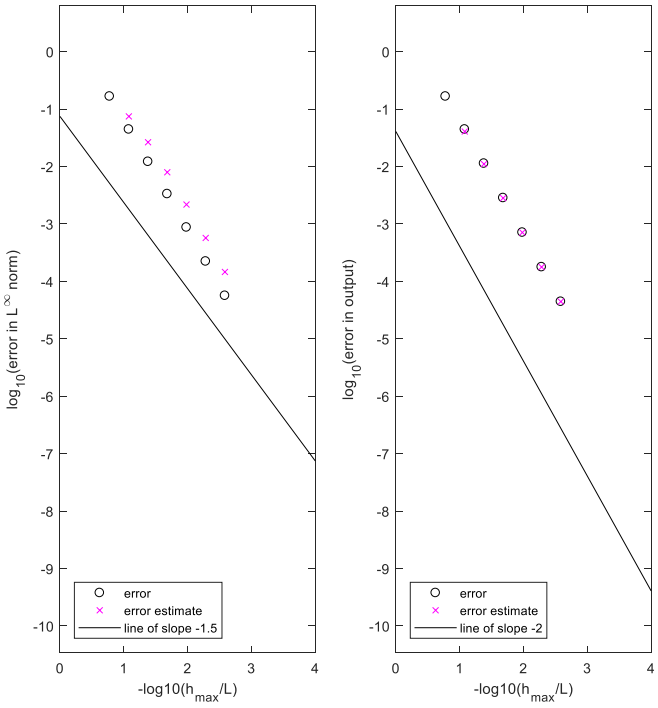


Figure 7 Ch1_Model_I Log-Log Convergence Plots for $L^{inf}(\Omega)$ and Output Error

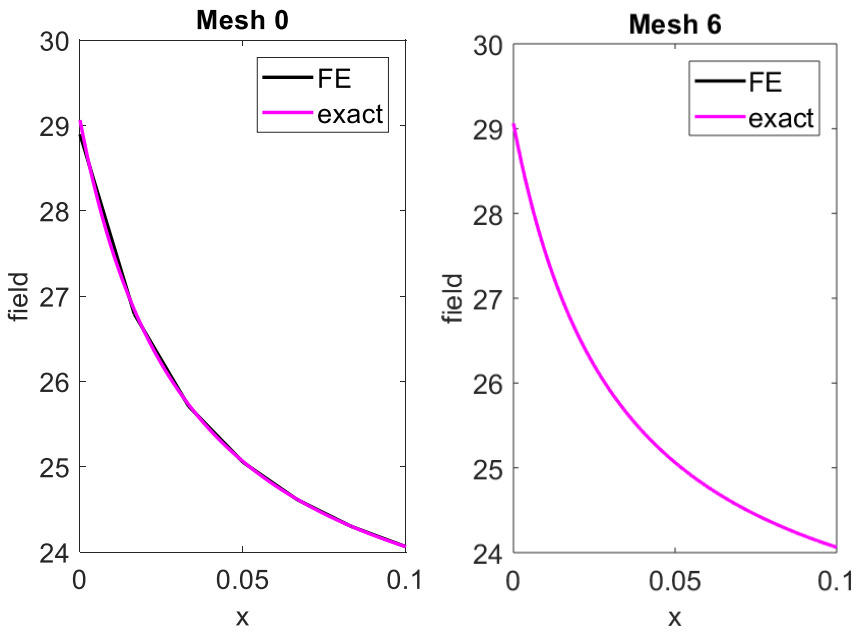


Figure 8 Plot of the Finite Element Method and Exact Solution for Ch1_Model_I for Mesh 0 and Mesh 6

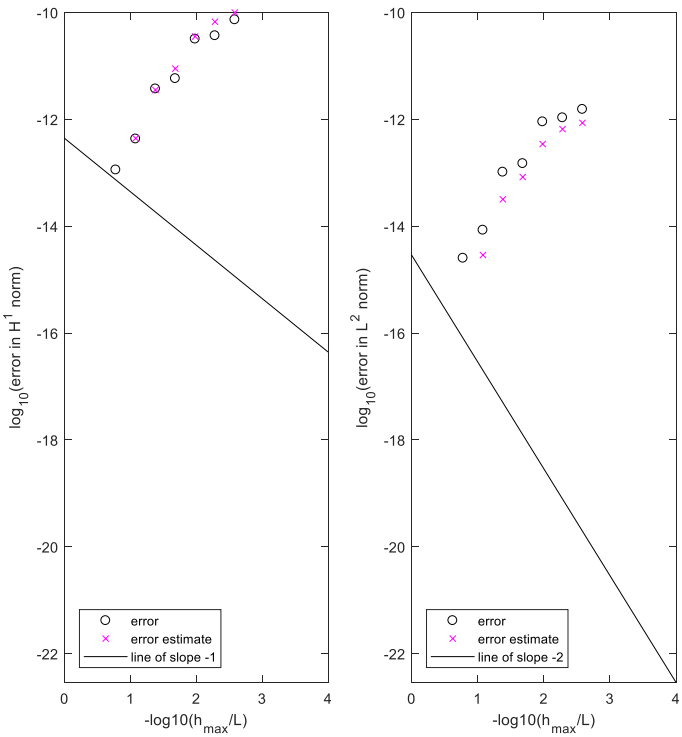


Figure 9 Ch2_Model_Mine Log-Log Convergence Plots for $H^1(\Omega)$ and $L^2(\Omega)$ Norm Error

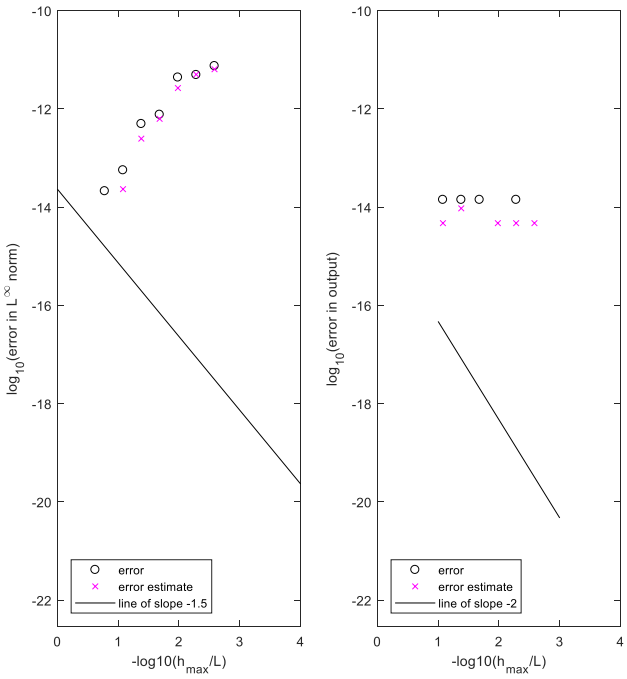


Figure 10 Ch2_Model_Mine Log-Log Convergence Plots for $L^{inf}(\Omega)$ and Output Error

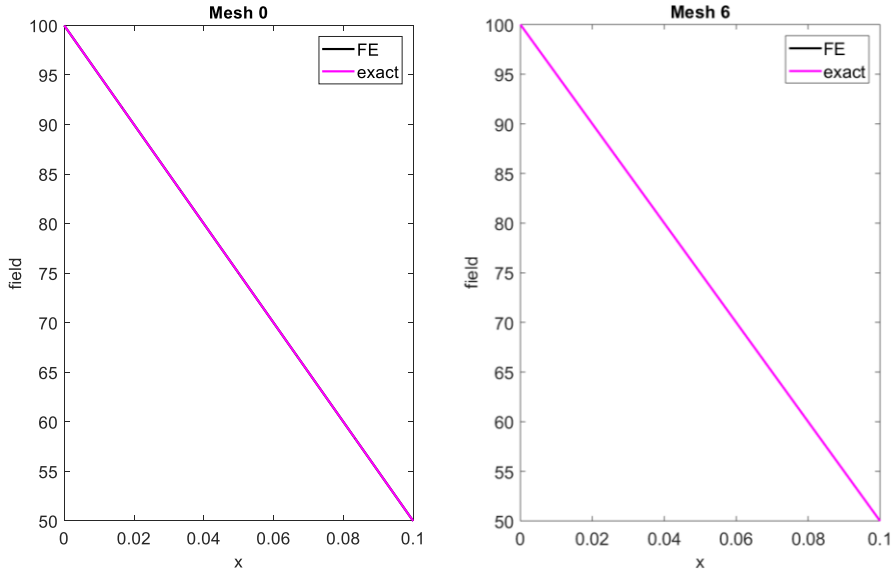


Figure 11 Plot of the Finite Element Method and Exact Solution for Ch2_Model_Mine for Mesh 0 and Mesh 6

7) Model_X

Now I consider a model, Model_X for which the exact solution is unknown. I observe that for sufficiently small *h*, the extrapolation error estimators converge at the anticipated rates in

all norms. However, you can not conclude that u_h is converging to the exact solution u. If I entered in the governing equation of Model_X correctly, but incorrectly entered the boundary conditions, u_h would converge to something but not the exact solution u that I was looking for.

8) Task 4

For Task 4 I looked at Ch1_Model_II on a sequence of nine meshes for p=1. Initially, I modified the Ch1_Model_II in library_of_models to simulate not knowing the exact answer. Then I found the coarsest mesh such that $\|u - u_h\|_{L^\infty(\Omega)}$ was less than 1. I determined it to be Mesh 7 with $\Delta_h^{L^\infty(\Omega)} = 0.636$. I propose an upper bound of $\|u - u_h\|_{L^\infty(\Omega)}$ as 0.636 given that $\|u - u_h\|_{L^\infty(\Omega)} \leq \Delta_h^{L^\infty(\Omega)}$ if $\|u - u_{\bar{h}}\|_{L^\infty(\Omega)} = C_u \bar{h}^{1.5}$. Next, I propose that the upper bound for the error in the output on Mesh 5 is 27.73 given that $|s - s_h| \leq \Delta_h^s$ if $|s - s_{\bar{h}}| = C_u \bar{h}^2$. Then I reverted the library_of_models code to include the exact solution. I compared the exact results to my predicted results. For Mesh 7 in the $L^\infty(\Omega)$ norm, the error was 0.3497, which was within my predicted bounds. For Mesh 5 in the output, the error was 31.84, which is about 14.8% higher than my predicted bound. Figure 12, which shows the log-log convergence of the error, provides some insight. The magenta crosses show the calculated Δ's and the circles show the actual error. For the L^{inf} norm on Mesh 7, the circle is below the cross, meaning the error is less than the predicted bound of Δ. For the output on Mesh 5, the circle is close to the cross, but slightly above, meaning the error is greater than the predicted bound of Δ.

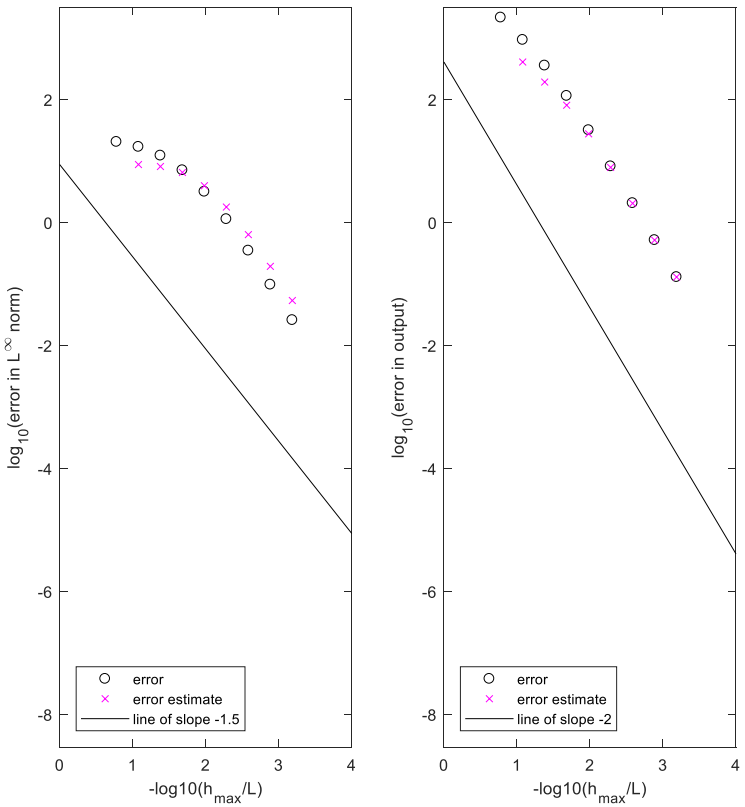


Figure 12 Ch1_Model_II Log-Log Convergence Plots for $L^{inf}(\Omega)$ and Output Error

CHAPTER 3 THE FD-FE METHOD FOR THE 1D HEAT EQUATION: FLIPPING BURGERS

1) Chapter Summary

In this chapter I discuss the Finite Difference-Finite Element (FD-FE) Method for the Heat Equation. Two models were explored: a semi-infinite fin and a burger. Both models have Neumann-Robin conditions on both ends, denoted as N/R-N/R. The FD-FE Method utilizes the FE method discussed in previous chapters to capture changes in space in conjunction with the FD method which captures changes in time. The semi-infinite fin model, called semiinf_plus, was used to verify the proper implementation of the FD-FE Method. Based on the convergence rates of the L^2 norm, I was able to determine that the FD-FE Method was properly implemented.

The burger model is more involved than the semiinf_plus model, because it involves three stages to model the flipping of a burger: the pre-flip, the post-flip, and the repose stages. The first step in verifying my implementation of the burger model was comparing my temperature graphs with those of Professor Patera. Both by visual inspection and by using the DataTip tool, I was able to confirm that my implementation achieved the same results as Professor Patera's implementation.

After verifying my correct implementation, I did some further verification and validation. I wanted to find the coarsest mesh with less than 0.001 °C error in output for the [p=1 and θ=1] and [p=2 and θ=1/2] schemes. I determined that 5 refinements were needed for the [p=1 and θ=1] scheme and 1 refinement for the [p=2 and θ=1/2] scheme, with the later being the most operationally efficient. Lastly, I found instructions for making a stovetop burger from The Kitchn and compared its predicted results with the results predicted by my make_uniform_refinement_burger_sver code. My code appeared to confirm the recipe's prediction of a pink center with charred outsides.

2) FD-FE Method

In this chapter we explore two applications of the Heat Equation. Because the Heat Equation is time-dependent, the Finite Element (FE) Method for 1D 2nd-Order SPD BVPs as developed in Chapter 2 is insufficient for solving these applications. The FD-FE Method builds upon the FE Method to enable approximations of time-dependent solutions. The FD formulation allows us to capture the change in $u^k_{h,\Delta t}$ over time. This is coupled with the FE

Method which allows us to capture the change in $u_{h,\Delta t}^{k}$ over space. I will briefly describe how this is set up in the following paragraph.

Both of our applications have Neumann/Robin conditions on both ends. This case has the following general form:

$$\frac{-d}{dx}\left(k(x)\frac{du}{dx}\right) + \mu(x)u = f_{\Omega} - \rho(x)\dot{u}\ in\ \Omega, 0 < t \leq t_f$$

$$k(x)\frac{du}{dx} = \gamma_1 u - f_{\Gamma_1}\ on\ \Gamma_1, 0 < t \leq t_f$$

$$-k(x)\frac{du}{dx} = \gamma_2 u - f_{\Gamma_2}\ on\ \Gamma_2, 0 < t \leq t_f$$

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

For the FE formulation, the governing matrix equations are:

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

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

Where $M_{ij}^{inertia} = \int_0^L \rho(x)\varphi_i\,\varphi_j dx, 1 \leq i,j \leq n_{node}$,

$A_{ij} = \int_0^L k(x)\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx} + \mu(x)\varphi_i\,\varphi_j dx + \gamma_1\varphi_i(0)\varphi_j(0) + \gamma_2\varphi_i(L)\varphi_j(L), 1 \leq i,j \leq n_{node}$, $and\ F_{ij} = \int_0^L f_{\Omega}\,\varphi_i dx + f_{\Gamma_1}\varphi_i(0) + f_{\Gamma_2}\varphi_i(L)$. To approximate $\underline{\dot{u}}_h$ and $\underline{u}_h$, the FD approximation is used. $\underline{\dot{u}}_h = \frac{u_{h,\Delta t}^{k} - u_{h,\Delta t}^{k-1}}{\Delta t}$, $\underline{u}_h = \theta u_{h,\Delta t}^{k} + (1-\theta)u_{h,\Delta t}^{k-1}, 2 \leq k \leq n_{tsteps}$ and $\underline{u}_h = \underline{(I_h u_{ic})}$, k=1. Where $\Delta t = \frac{t_f}{n_{tsteps}-1}$ and θ indicates which FD scheme is being used. If $\theta = 0$ the scheme is Euler Forward. If $\theta = 0.5$ the scheme is Crank-Nicolson. If $\theta = 1$ the scheme is Euler Backward. When you plug in the FD approximations for $\underline{\dot{u}}_h$ and $\underline{u}_h$, into the FE formulation, you get

$\left(\frac{\underline{\underline{M}}^{inertia}}{\Delta t} + \theta\underline{\underline{A}}\right)u_{h,\Delta t}^{k} = \left(\frac{\underline{\underline{M}}^{inertia}}{\Delta t} - (1-\theta)\underline{\underline{A}}\right)u_{h,\Delta t}^{k-1} + \underline{F},$ for $k = 2{:}\,n_{tsteps}$. To tie it all back together, $u_{h,\Delta t}^{k}(x) \approx u(x,t^k), 1 \leq k \leq n_{tsteps}$ giving us our approximation of the temperature distribution in time and space.

3) Verification with semiinf_plus

I needed to implement solve_fld_output_t_sver to calculate the temperature field over time. I replaced the instructor's call to solve_fld_output_t with my own

solve_fld_output_t_sver in run_uniform_refinement. To verify that I properly coded solve_fld_output_t_sver, I ran run_uniform_refinement on the semiinf_plus model. Then I checked the plots and convergence rates to confirm they were behaving as theory predicts. Figure 1 is a plot of the non-dimensional temperature Θ over non-dimensional time F_0. The theory for a semi-infinite fin predicts that $\Theta \sim 1 - Const.\sqrt{F_0}$. From Figure 1, we can confirm that Θ=1 when F_0=0. As F_0 increases, Θ decreases as expected. Then I checked the error convergence for both p=1, θ=1 and p=2, θ=1/2. The error for the L^2 norm $\sim C_{u,L^2}\left(\frac{L}{h_0}\right)^r$. Therefore, the log error for the L^2 norm over the log of $\frac{h_0}{L}$ should have a slope of -r. For [p=1 and θ=1], r=2, and for [p=2 and θ=1/2], r=3. Figures 2 and 3 show that for [p=1 and θ=1] and [p=2 and θ=1/2] the error estimates converge with slopes of -2 and -3 respectively. This result makes me confident that I correctly implemented solve_fld_output_t_sver.

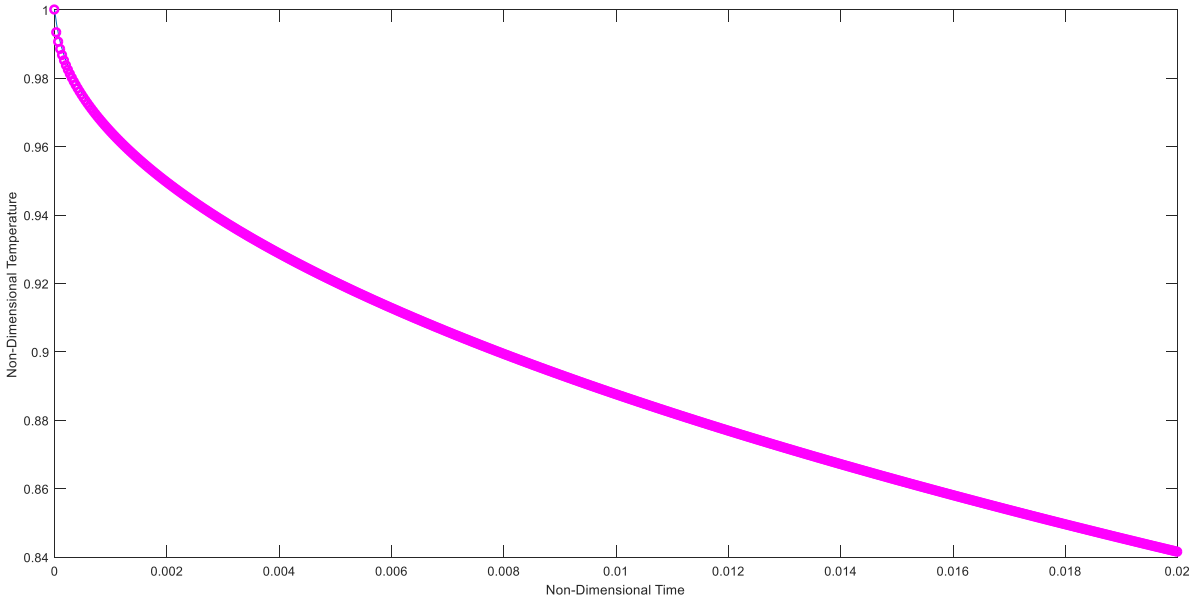


Figure 1 Non-Dimensional Temperature vs. Non-Dimensional Time for p=1 and θ=1

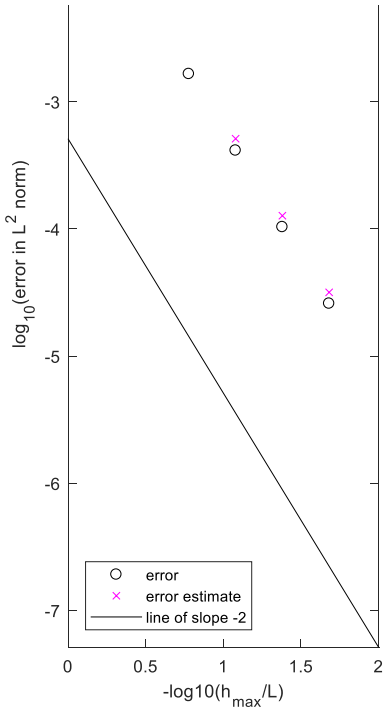


Figure 2 Log Error Plot for L^2 norm for p=1 and θ=1

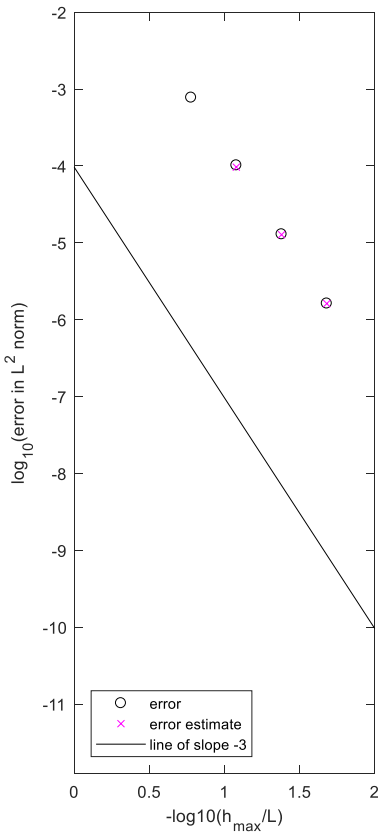


Figure 3 Log Error Plot for L^2 norm for p=2 and θ=1/2

4) Verification of make_probdef_burger

In order to use the FD-FE method to flip burgers with run_uniform_refinement_burger_sver, I had to implement make_probdef_burger. To verify that I implemented make_probdef_burger correctly, I ran run_uniform_refinement_burer_sver and compared my results with that of Professor Patera's results. Figure 4 and 5 show graphs of the final temperature distribution in the burger for Professor Patera's and my implementation, respectively. By visual inspection, the shape of these curves appears similar. I also confirmed with the data tip tool that the temperatures at x=0 and x=L are the same for both plots, 57.67°C and 46.87°C respectively. Using the same methods as described for Figures 4 and 5, I confirmed that Figures 6 and 7, the temperature over time graphs for Professor Patera and me respectively, produced the same results. These results make me confident that I correctly implemented make_probdef_burger.

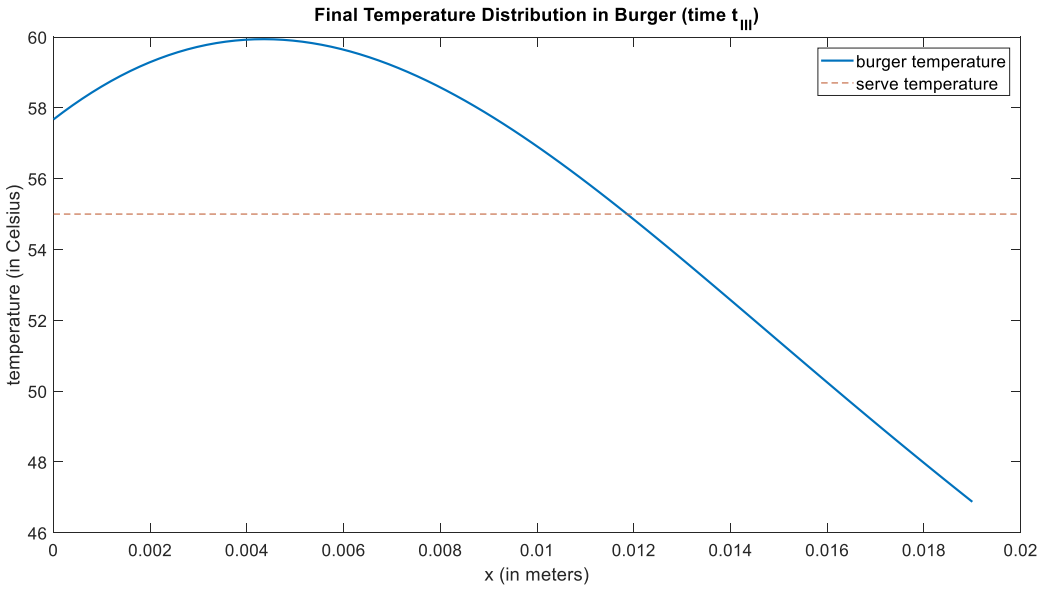


Figure 4 Professor Patera's Final Temperature Distribution in Burger

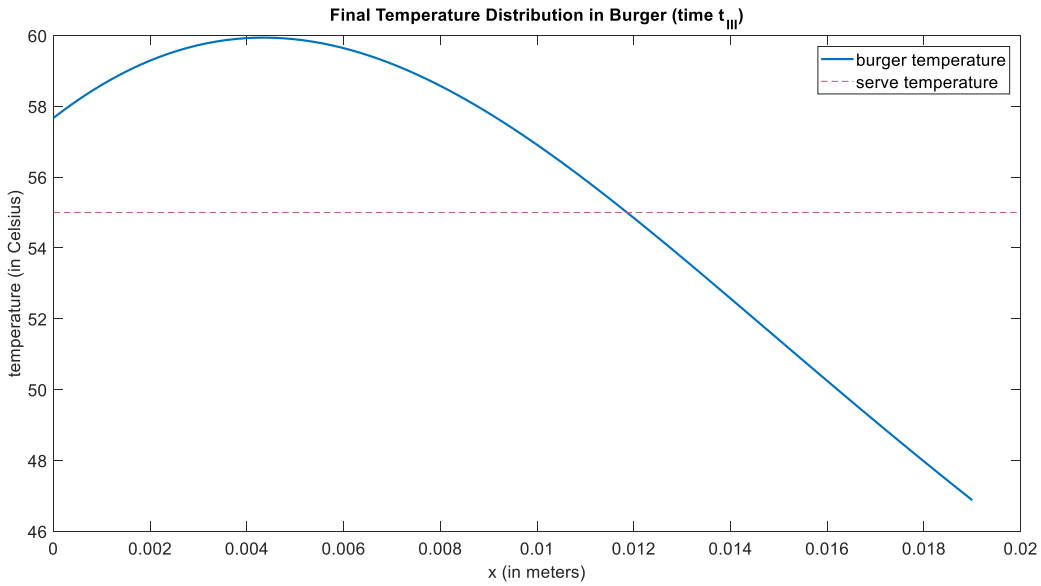


Figure 5 Margaret's Final Temperature Distribution in Burger

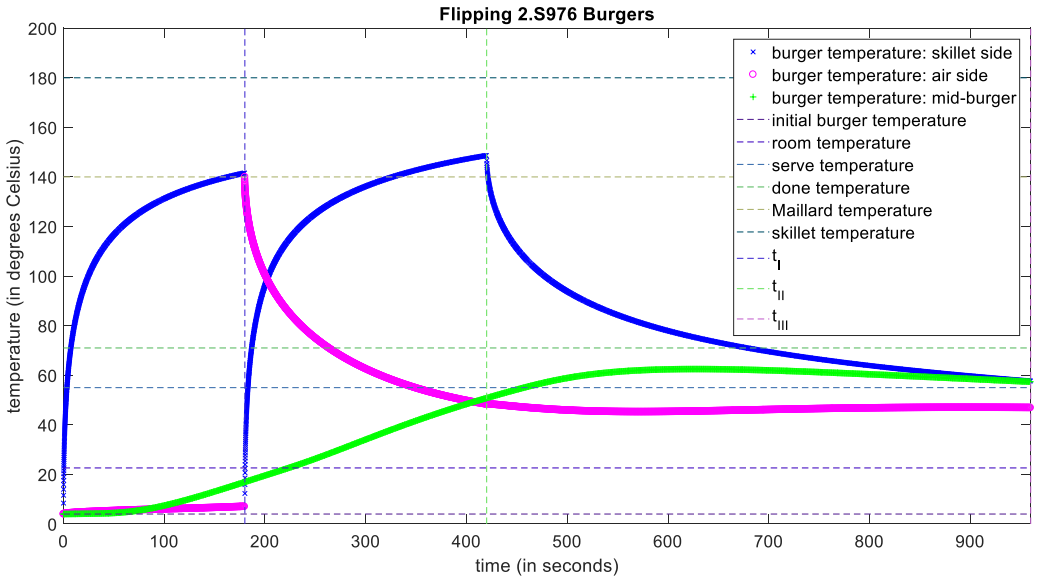


Figure 6 Professor Patera's Temperature Over Time Graph

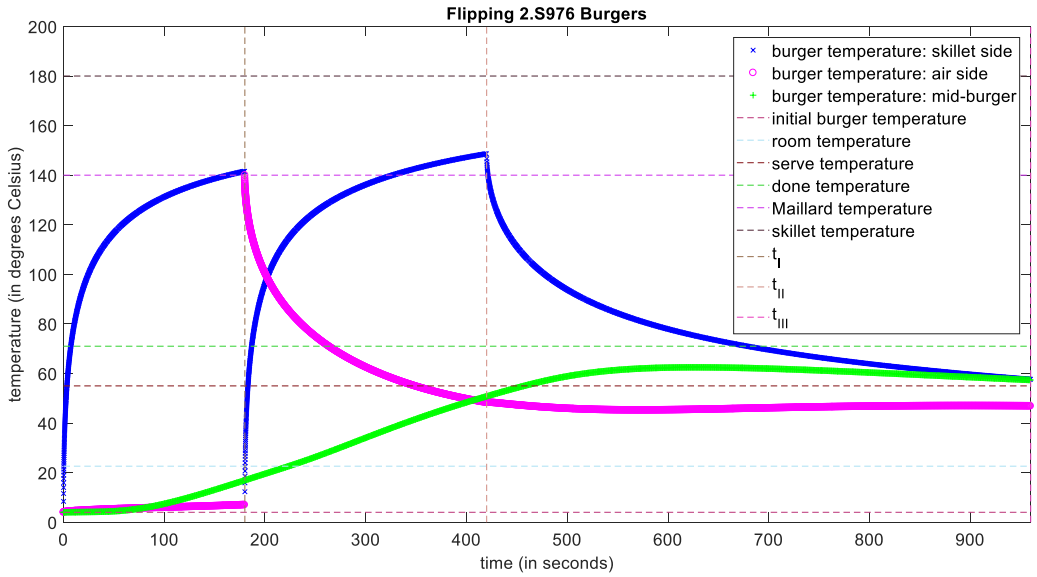


Figure 7 Margaret's Temperature Over Time Graph

5) Verification of Numerical Specifications

Next, I wanted to verify a numerical specification on the output error for both the [p=1 and θ=1] and [p=2 and θ=1/2] schemes. I wanted to find the coarsest mesh with less than 0.001°C error in the output. Since the error graphs are log plots, I looked for the mesh with a log error less than -3. When p=1 and θ=1, Figure 8 shows that the coarsest mesh that fits the specification is mesh 6 which corresponds to the fifth refinement. The error is 0.0004°C. When p=2 and θ=1/2, Figure 9 shows that the coarsest mesh that fits the specification is mesh 2 which corresponds to the first refinement. The error is 0.0005°C. Then I wanted to determine whether the [p=1 and θ=1] or [p=2 and θ=1/2] scheme was more operationally efficient when the specification was reached. For the FD-FE Method, the total cost is $O\left(\frac{t_f}{\Delta t/_{\sigma^l}}\right) \times O\left(\frac{L}{h/_{2^l}} p\right)$ for l^{th} refinement. For the p=1 and θ=1 scheme, the fifth refinement met the specification and σ=4. For the p=2 and θ=1/2 scheme, the first refinement met the specification and σ=2√2. For the p=1 and θ=1 scheme, the operational cost was on the order of 3,932,160, and for the p=2 and θ=1/2 scheme, the operational cost was on the order of 1,357.645. The operational cost for the p=2 and θ=1/2 scheme must be multiplied by two, since a pentadiagonal matrix costs two times as much as a tridiagonal matrix. The adjusted

cost for the p=2 and θ=1/2 scheme is 2715.3. Therefore, the p=2 and θ=1/2 scheme is the most operationally efficient scheme to meet the 0.001°C error specification.

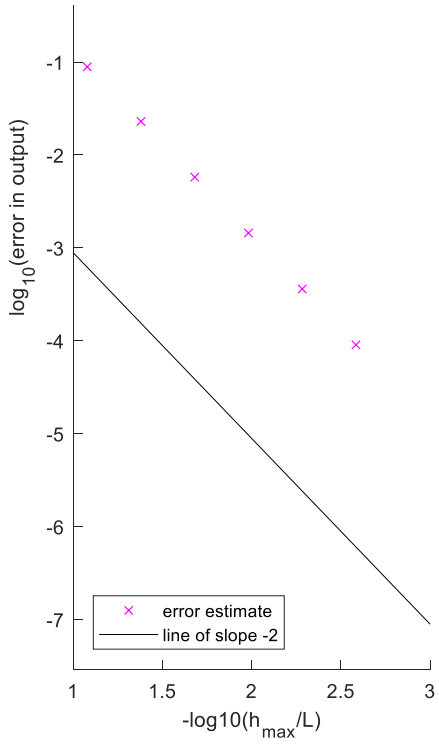


Figure 8 Log Error in Output for p=1 and θ=1

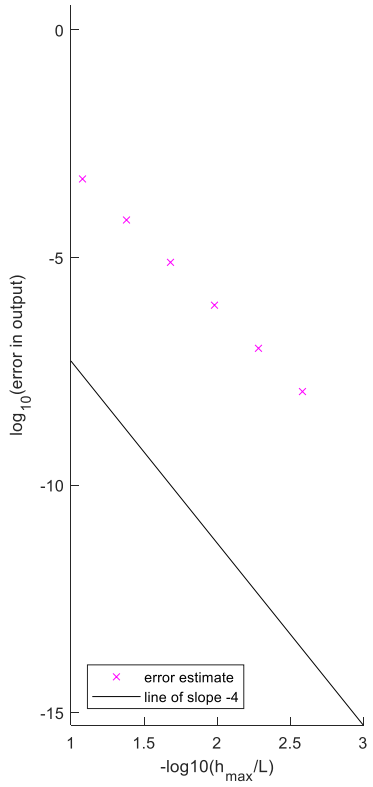


Figure 9 Log Error in Output for p=2 and θ=1/2

6) Internet Hamburger Instructions

Finally, I plugged in the parameters taken from internet instructions for a burger recipe to the run_uniform_refinement_burger_sver code to see if it would generate the same results as the recipe. I followed recommendations from The Kitchn on "How to Make Burgers on the Stovetop" found at the following url: https://www.thekitchn.com/how-to-make-burgers-on-the-stovetop-cooking-lessons-from-the-kitchn-217722. Some of the parameters were presented directly and others I had to approximate. From the recipe I gathered that the burger thickness was 1 inch and the duration of the preflip and postflip stages were 4 minutes each. I estimated that the skillet temperature was 205 °C (medium-high heat), the burger diameter was 4 inches, and the repose time was estimated as 30 seconds. The directions indicated that the burger should be medium, which corresponds to a pink center. I inspected Figures 10 and 11 to determine if the model was in line with the recipe's predicted outcome. Figure 10 shows that both sides of the burger get charred, but the middle of the burger does not reach the "done temperature" which could signify a pink center. Figure 11 shows that the outsides of the burger will be hot when

served, but the middle of the burger is colder than the serve temperature. This is consistent with a burger coming hot off the skillet with a pink center.

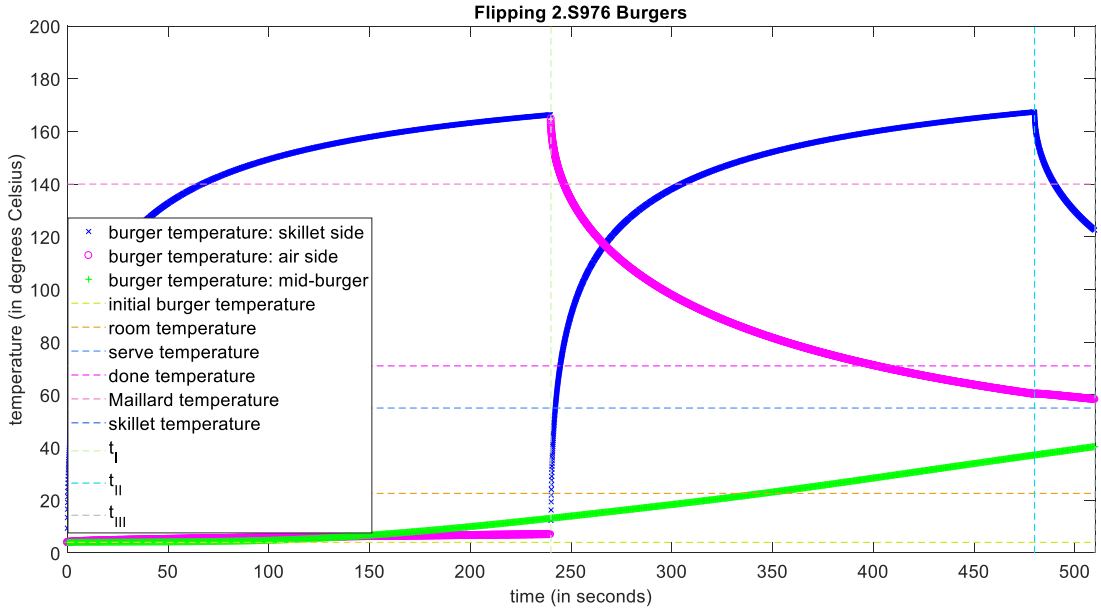


Figure 10 Temperature Over Time Graph for The Kitchn Recipe

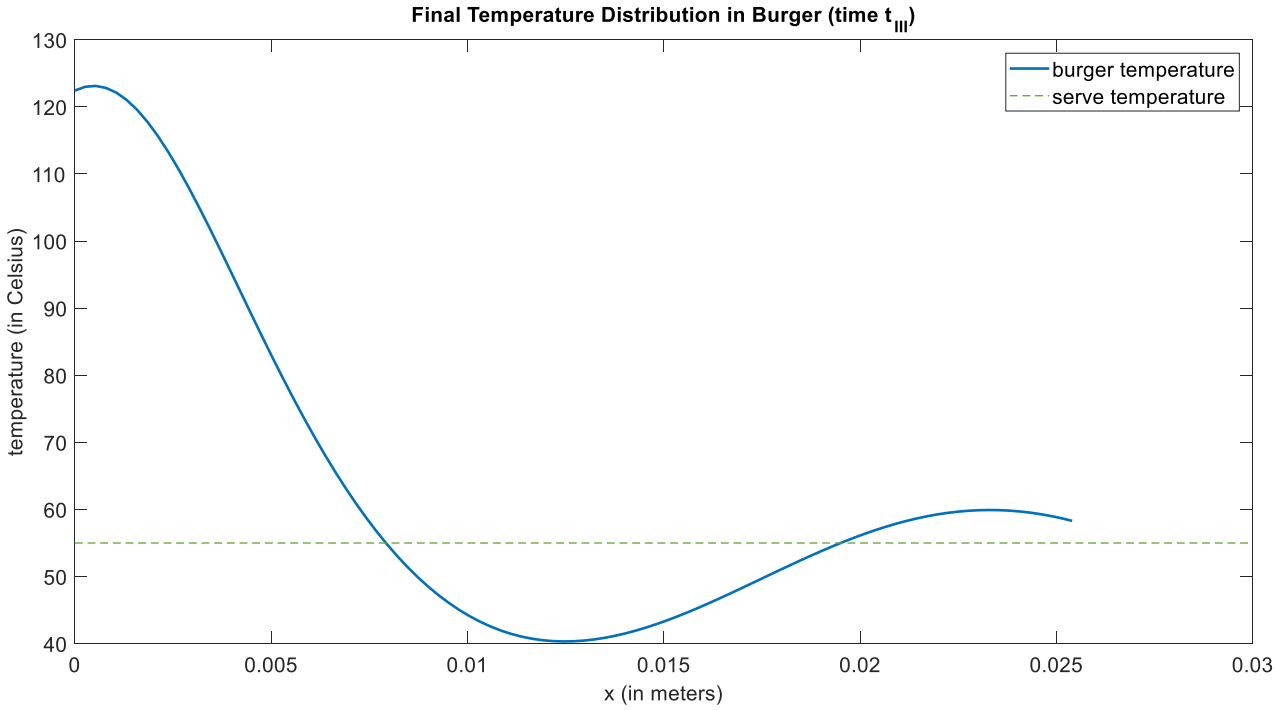


Figure 11 Final Temperature Distribution in the Burger for the Kitchn Recipe

CHAPTER 4 THE FE METHOD FOR 1D 4TH-ORDER BVPS (BENDING):XYLOPHONE

1) Chapter Summary

In this chapter I discuss the Finite Element Method for Beam Eigenproblems. Three applications were explored: the design and tuning of a xylophone bar, a Caresta Test, and a bar with a Hookean spring attached. The FE Method finds the vibration modes of a beam. In the case of the xylophone problem, the FE Method is used to design the profile of a xylophone bar to achieve a specified note, the fundamental mode, and tuning, indicated by the ratio of the harmonic to the fundamental. The program used for the xylophone design, xylo_bar_design3, had to be amended to specify the placement of the support holes. The holes had to be placed to not interfere with the fundamental mode's vibration and maintain the "free" support condition. A hole finder algorithm was coded to first find the elements where zeros occurred on the fundamental frequency, then find where in the element the zero occurred, and lastly scale that location to correspond with a dimensional location on the beam.

To verify that xylo_bar_design3 was able to find the correct frequencies and bar length, its driver, xyloDesignDriver, was modified to run the Caresta Test. The Caresta Test identifies the vibrational modes of a beam with uniform cross-section. The xylo_bar_design3 Caresta Test results were compared with that of the Caresta experiment's theory. The xylo_bar_design3 results had less than 0.03% error for the first two frequencies and the bar length. With the verification that the xylo_bar_design3 program was working properly, a xylophone bar was designed for the fundamental note C5=523.25 Hz with quint tuning. Another bar was designed for the fundamental note F4=349.23 Hz with quint tuning to discuss the calculated error estimates, modeling error, mesh coarseness, and relative error size. Lastly, I discussed a potential modification to the UG_FE_1d_bend_sver folder to enable the study of another model: a free beam with a Hookean spring on one end. As it is like adding a "Robin" condition, I posited that adding in the condition in the program impose_boundary_cond would enable me to add the spring condition on the end of the bar.

2) Summary of the FE Method for Beam Eigenproblems

A beam is defined in space, with respect to its length L, by the following:

$$\Omega \equiv (0, L);\ \bar{\Gamma} = \bar{\Gamma_1} \cup \bar{\Gamma_2}, \Gamma_1 = \{0\}, \Gamma_2 = \{L\}$$

The beam eigenproblem is governed by the following equation:

$$\frac{d^2}{dx^2}\left(\beta(x)\frac{d^2u}{dx^2}\right) - N_0\frac{d^2u}{dx^2} = q(x,t) - \rho A_{cs}(x)\frac{d^2u}{dt^2}, for\ 0 < x < L, 0 < t \leq t_f$$

Where $\beta(x)$ is the effective stiffness of the beam, *u(x,t)* is the beam displacement profile, N_0 is the constant axial force, and $q(x,t)$ is the force per unit length being applied to the beam. Depending on the way the beam is supported, different essential boundary conditions will be applied to the ends of the beam. Table 1 shows how homogenous boundary conditions are determined for $0 < t \leq t_f$.

BC	**Clamped**	**Free**	**Simply Supported**
$u(0) = 0$	X		X
$u_x(0) = 0$	X		
$u(0) = L$	X		X
$u_x(L) = 0$	X		

Table 1 Essential Homogenous Boundary Conditions and Support Types for $0 < t \leq t_f$

Initial conditions for displacement and velocity are indicated by:

$$u(x, t = 0) = u_{ic}\ and\ \dot{u}(x, t = 0) = \dot{u_{ic}}\ for\ x\ in\ \Omega$$

When represented modally, where *q(x,t)*=0, the beam eigenproblem becomes:

$$\frac{d^2}{dx^2}\left(\beta(x)\frac{d^2u^{(k)}}{dx^2}\right) - N_0\frac{d^2u^{(k)}}{dx^2} = \lambda^{(k)}\rho A_{cs}u^{(k)}, for\ 0 < x < L$$

Where k is the mode number and λ is the square of the natural frequency of the mode. To implement the finite element method, the beam is divided into n_{el} elements with n_e+1 nodes. Hermitian basis functions are used, and each node is given two degrees of freedom to capture the displacement and derivative of the displacement. Using a double-mapping, the FE representation is:

$$u_h(x) = \sum_{j=1}^{2*n_{node}} u_{hj}\varphi_j(x)$$

The eigenproblem is solved with the following:

$$\underline{A}\underline{u}_h^{(k)0} = \lambda_h^{(k)}\underline{M}^{inertia}\underline{u}_h^{(k)0}$$

$$\left(\underline{u}_h^{(k)0}\right)^T\underline{M}^{inertia}\left(\underline{u}_h^{(k)0}\right) = 1$$

Where $\underline{A}$ and $\underline{M}^{inertia}$ are obtained by

$$\widetilde{M}_{ij}^{inertia} = \int_0^L \rho A_{cs}(x)\varphi_i\,\varphi_j dx,\ 1 \le i \le 2 * n_{node},$$

$$\tilde{A}_{ij} = \int_0^L \beta(x)\frac{d^2\varphi_i}{dx^2}\frac{d^2\varphi_j}{dx^2} + N_0\frac{d\varphi_i}{dx}\frac{d\varphi_j}{dx}dx\ , 1 \le i,j \le 2 * n_{node}$$

And $\underline{A}$ and $\underline{M}^{inertia}$ are obtained from $\tilde{A}$ and $\widetilde{M}$ by imposing the essential boundary conditions.

3) Summary of Xylophone Bar Problem

I explored the beam eigenproblem application of the design of a xylophone bar. The profile of a xylophone bar is optimized so that when hit, the bar's fundamental frequency is that of the desired note and the first harmonic is either quint-tuned, a multiple of three, or double-octave tuned, a multiple of four. Diagram 1, taken from the April 23rd 2019 notes from Professor Patera, shows the geometry of the xylophone bar.

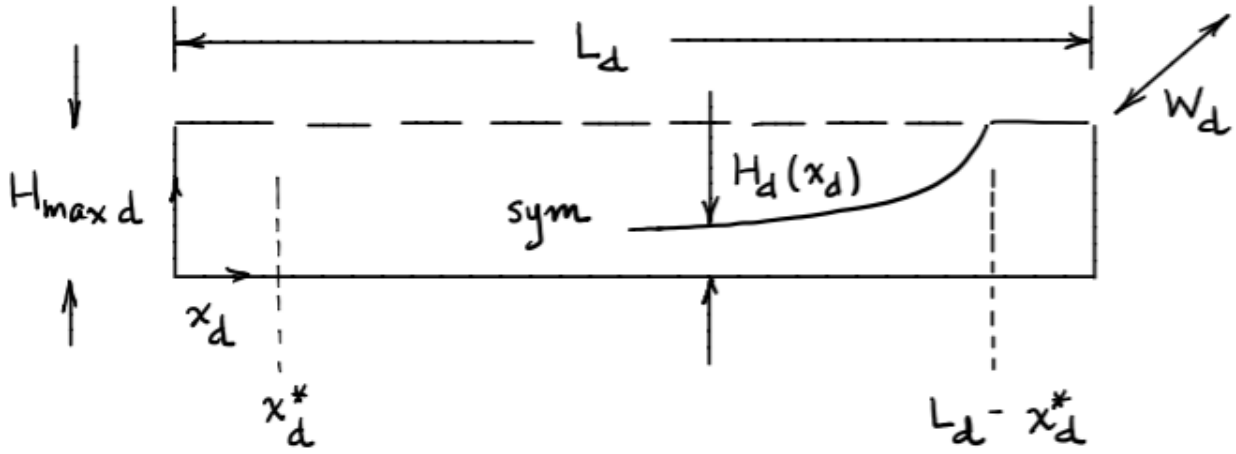


Diagram 1 Geometry of the Xylophone Bar Taken from the April 23rd 2019 Notes

The profile of the beam, and thus the tuning, is determined by $H_d(x)$. $H_d(x)$ is defined by:

$$H_d(x_d) = H_{maxd}\left[(1-p_2)\left(\frac{L_{d/2} - x_d}{L_{d/2} - x_d^{star}}\right)^{p_1} + p_2\right], for\ x_d^{star} \le x_d \le L_d - x_d^{star}$$

$$H_d(x_d) = H_{maxd}, for\ 0 \le x_d < x_d^{star}\ and\ L_d - x_d^{star} < x_d \le L_d$$

P_1 is set to equal four, and P_2 is the design variable to achieve the desired tuning. The governing eigenproblem for the xylophone bar is:

$$\frac{d^2}{dx^2}\left(\frac{E_d W_d H_d^3(x_d)}{12}\frac{d^2 u_d{}^{(k)}}{dx_d{}^2}\right) = \lambda_d{}^{(k)}\rho_d W_d H_d(x_d) u_d{}^{(k)}, for\ 0 < x_d < L_d$$

Because the xylophone bar is supported by two strings in tension, the support of the beam is approximately free.

4) Algorithm for Hole Placement

I implemented an algorithm to properly place the support holes for the xylophone bar, indicated by x^{star}_d in Diagram 1. I want to place the holes where the fundamental mode has zero deflection to ensure that the hole placement does not affect the vibration, and to maintain the "free" support condition. The algorithm is run after the tone and harmonic modes are determined. The algorithm has three steps. First, I want to identify which element, m^{star}, contains a zero for the tone mode. This is done by finding a sign change over an element in the tone mode. Then I want to find where the zero occurs in that element. This is done by finding the point, x^{hat}, where the sum of all the FE coefficients and their corresponding shape functions for the tone mode equal zero. The final step is converting from the non-dimensionalized x^{hat} to a dimensionalized coordinate x^{hole}_d. For simplicity, I described how to find one zero. However, in the MATLAB code, the algorithm is coded to find both zeros. This is done by making m^{star}, x^{hat}, and x^{hole}_d 2x1 vectors and creating loops within the three steps to find both zeros. Figure 1 shows the results of running the xylo_bar_design3 program for a quint-tuned C5 bar with the whole finder algorithm implemented. In this figure, the dark blue dashed-lines indicate where the holes will be placed. Visual inspection confirms that the holes are placed when the modal deflection is zero for the fundamental mode.

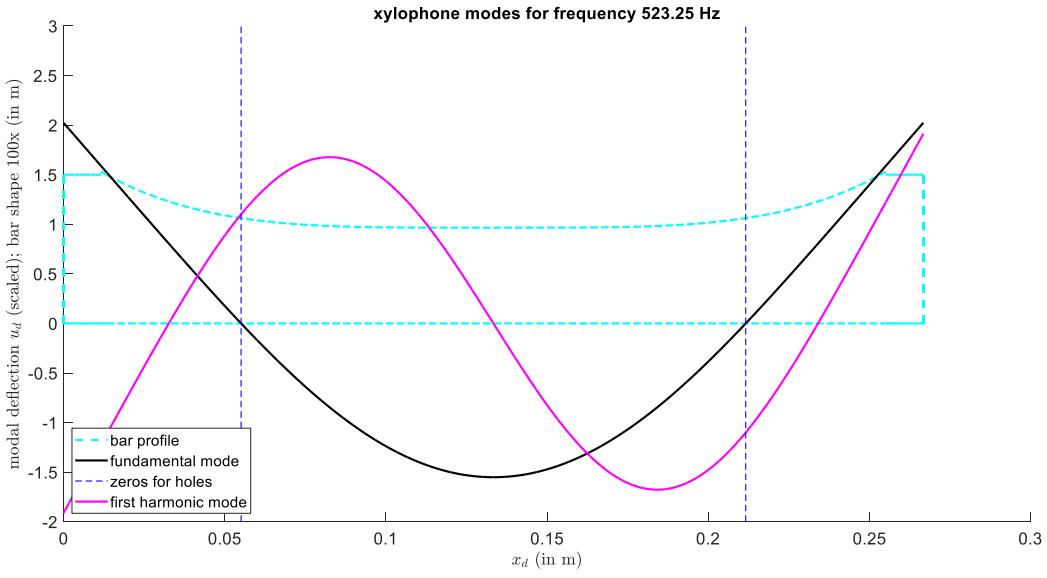


Figure 1 Xylophone Modes for C5=523.25 Hz with Hole Finding Implemented

5) Caresta Test Verification

The Caresta Test serves as a verification test for two aspects of the xylo_bar_design3 code. It confirms the code's ability to find the fundamental and harmonic frequencies and the ability to calculate the required beam length for those frequencies. The Caresta Test looks at the vibrations of a free beam with a uniform cross section. It provides both theoretical and experimental mode frequencies. I modified the xylo_bar_design3 driver, called xyloDesignDriver, to run the Caresta Test. Table 2 shows the inputs that were used in the xyloDesignDriver for the Caresta Test. Table 3 compares the results of the xylo_bar_design3 Caresta Test with the actual values from the Caresta experiment. The modified xylo_bar_design3 has less than 0.03% error for the three outputs of interest. Figure 2 shows the first two modes calculated by the xylo_bar_design3 Caresta Test. The mode shapes are consistent with those reported by Caresta when the diagram is flipped (this is necessary because the xylo_bar_design3 program is designed for an upside-down beam).

Input	**Description**	**Value**
frequency3target_d	Target frequency for the fundamental	32.80 Hz
R_target	Ratio of harmonic to target	90.44/32.80
Hmax_d	Max thickness	0.01 m
x^{star}	Beginning of profile	0.05 m
p2_interval	Tuning interval	[1,1]
Ebar_d	Young's Modulus	$2.1x10^{11}$ Pa
rhobar_d	Bar density	7800 kg/m^3
justcalc_L_d	Calculate only the length	true
suppress	Suppress graphs	true

Table 2 Inputs to xyloDesignDriver for Caresta Test

Output	**Caresta Experiment**	**xylo_bar_design3**	**Percent Error**
frequency3_d	32.80	32. 799999999999997	~0
frequency4_d	90.44	90.414466828282841	0.028

L_d	1.275	1.275187035433650	0.015

Table 3 Comparison of xylo_bar_design3 Caresta Test output and Caresta Experiment

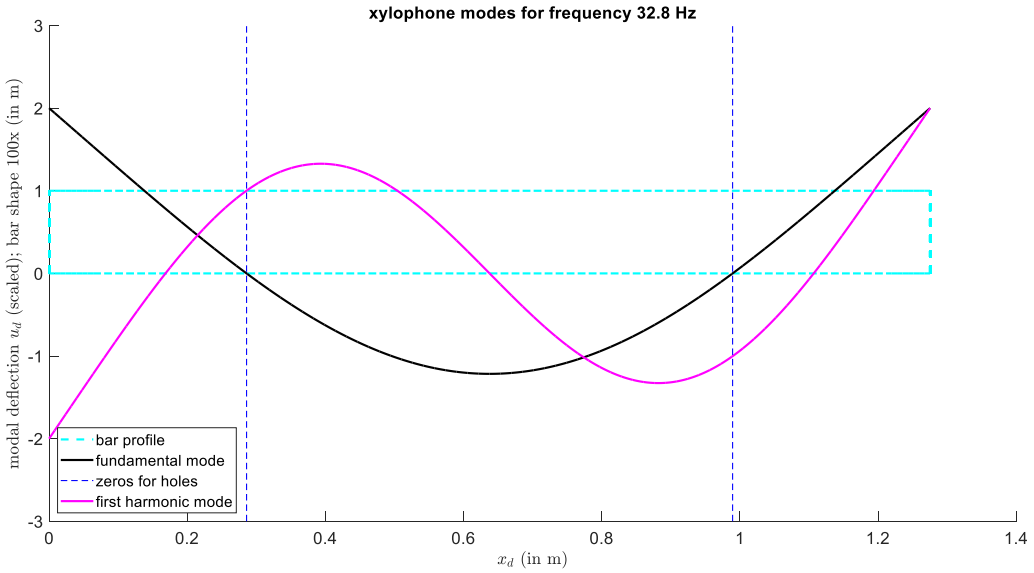


Figure 2 First Two Modes for xylo_bar_design3 Caresta Test

6) Tuning Results

I tuned a xylophone for the note C5 with a frequency of 523.25 Hz and quint-tuning, R=3.Table 4 describes the inputs used for the xyloDesignDriver. The xylo_bar_design3 program found that the optimal tuning design variable p2opt=0.64375. The length of the xylophone bar was found to be 0.26669 m. Figure 1 shows the tone and harmonic modes and the optimal placement of the support holes. Table 5 shows the calculated frequencies, frequency ratios, and error estimates. To calculate the value bounds for the frequency ratio, the errors from both the fundamental and harmonic frequency had to be considered. The lower value bound is where the numerator (the harmonic) is minimized and the denominator (the fundamental) is maximized. The upper value bound is where the harmonic is maximized and the fundamental is minimized. In symbolic form that translates to:

$$\left[\frac{f^4 - error_{f^4}}{f^3 + error_{f^3}}, \frac{f^4 + error_{f^4}}{f^3 - error_{f^3}}\right]$$

Input	**Description**	**Value**

frequency3target_d	Target frequency for the fundamental	523.25
R_target	Ratio of harmonic to target	3
Hmax_d	Max thickness	0.015 m
x^{star}	Beginning of profile	0.05 m
p2_interval	Tuning interval	[0.05,1]
Ebar_d	Young's Modulus	$1.4x10^{10}$ Pa
rhobar_d	Bar density	835 kg/m^3
justcalc_L_d	Calculate only the length	false
suppress	Suppress graphs	true

Table 4 Inputs to xyloDesignDriver for C5 Quint Tuning

Quantity	**Value**	**Error Estimate or Value Interval**
frequency3_d	523.25 Hz	$8.764x10^{-6}$
frequency4_d	1565.18 Hz	$2.930\ x10^{-6}$
frequency4_d/frequency3_d	2.99	[2.9912632,2.9912633]

Table 5 Results of xylo_bar_design3 C5 Quint Tuning and Error Estimates

7) Discussion of Errors

To explore the errors in FE predictions, a second bar was quint tuned for F4=349.23 Hz. Figures 3 and 4 show the log error convergence rates over the course of four refinements on the fundamental and harmonic mode, respectively. The log errors are converging with the expected slope for the four error norms. The calculated error estimates for the fundamental (f3) and harmonic (f4) were $5.85x10^{-6}$ and $1.96x10^{-6}$ respectively. This, combined with the log 10 errors on the output reported in Table 6, suggests the calculated error estimate is not accurate. The error on the output, λ^k, for Mesh 3 is on the order of 10^{-6} and 10^{-5} for the fundamental and harmonic respectively. At first glance, this appears close to the calculated estimates, but the estimates

calculate the error on the frequency. To go from the error on the output to the error in frequency requires the following adjustment:

$$\left|f^{(k)} - f_{h/2}^{(k)}\right| \approx< \frac{1}{2(2\pi)^2} \frac{1}{f_{h/2}^{(k)}} \Delta_{h/2}^{output}$$

When the adjustment is made, Figures 3 and 4 suggest that the error on the fundamental frequency should be approximately less than 1.76×10^{-11} and the error on the harmonic frequency should be approximately less than 2.98×10^{-10}. Because the fundamental frequency is the main frequency of interest, as it is the dominant tone, I want the error on the fundamental to be less than the error on the harmonic. The results support this hypothesis. Table 6 shows that for a given mesh, the log error on the output is larger for the fundamental then the harmonic. This means the error on the output of the fundamental is less than that of the harmonic.

Now I turn to how appropriate the number of meshes is and how appropriate the model is. The untrained human ear has a sensitivity of 10 Hz. On Mesh 1, the first refinement, the error on the output of the fundamental is 1.26×10^{-4}. Using the adjustment factor, the error on the fundamental frequency is approximately less than 4.57×10^{-9}, which is significantly less than 10 Hz. Therefore, for the untrained ear, even one refinement is too fine to make a significant difference. The xylophone bar is modeled as a Euler-Bernoulli beam. An assumption of the Euler-Bernoulli model is that the beam is thin. Therefore, I predict that the modeling error will be greater for higher frequencies than lower frequencies, because as the frequency becomes higher, the bar becomes shorter. The shorter the bar becomes, the less like a thin beam the xylophone bar becomes.

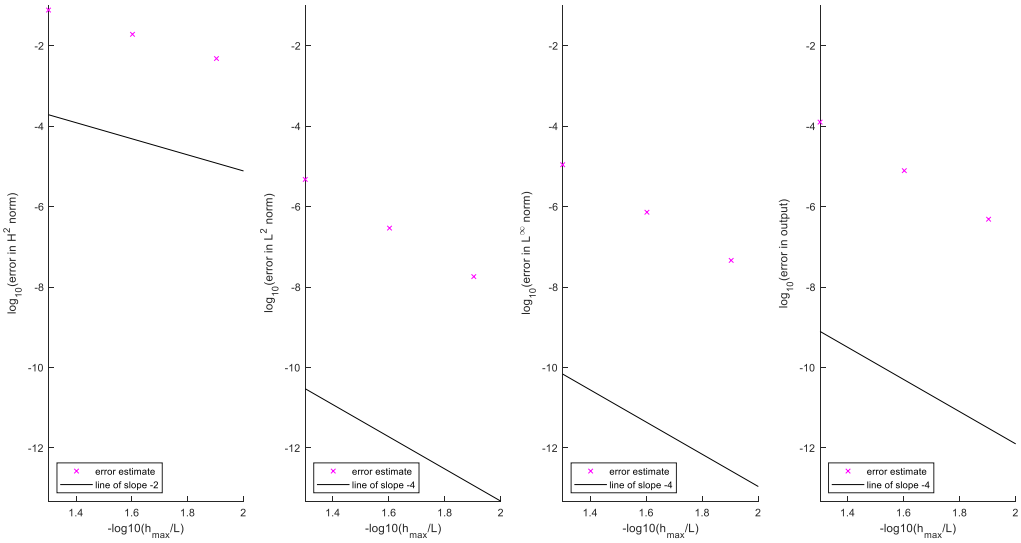


Figure 3 Error Convergence on the Fundamental for F4 Quint Tuning

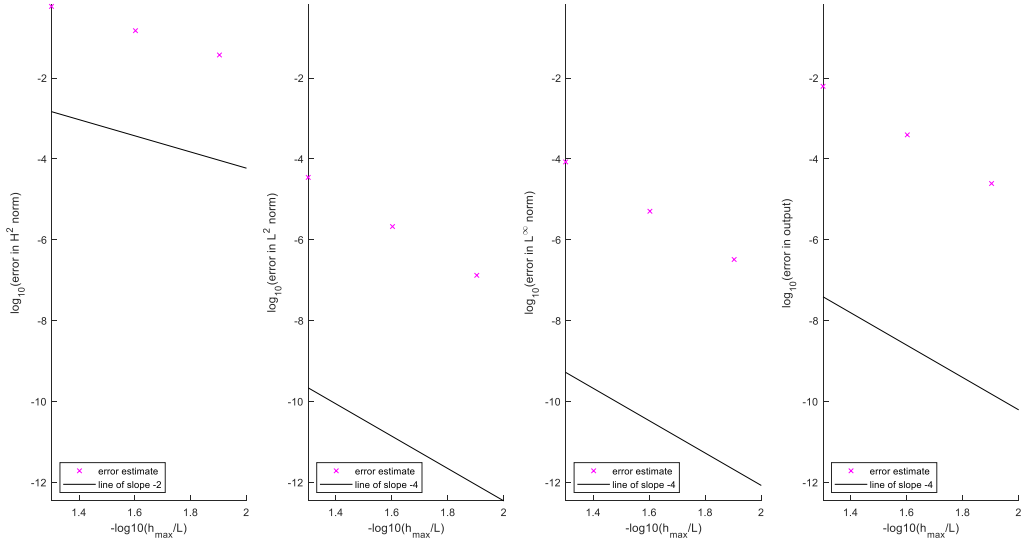


Figure 4 Error Convergence on the Harmonic for F4 Quint Tuning

Mesh	Fundamental Log Error in Output	Harmonic Log Error in Output
1	-3.8995	-2.2057
2	-5.1063	-3.4061
3	-6.3138	-4.6088

Table 6 Log 10 Error on the Fundamental and Harmonic Output for F4 Quint Tuning for 3 Meshes

8) Spring Modification Code

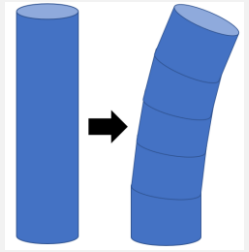
The folder UG_FE_1d_bend_sver can be modified to model a free beam on one side with Hookean spring on the other side. This is like a "Robin" boundary condition. Due to the similarities with a task in Chapter 2 I decided that the modification should occur in the impose_boundary_cond program. The modification would occur after A is assigned to A_N (line 18) and look like the following:

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

The modification is inspired by the following rationale. The spring is only attached at the end of the bar. The only shape function that is non-zero at the end of the beam is φ_5. Because of the double-mapping, the quantity of interest, the displacement at the end of the bar, occurs at the second to last node number, where the node number is equal to 2*(n_el0+1) where n_el0+1 is equal to the number of nodes in the double-mapping scheme. Therefore, the A entry that corresponds to that node should have ks, the spring constant, added to it.

CHAPTER 5 SELF-BUCKLING

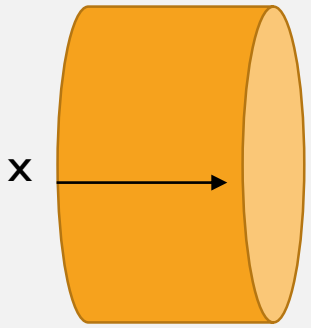
By Margaret Bertoni



SELF-BUCKLING

Governing Equations (Non-Dimensionalized)

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



$$u=u_x=0\ at\ x=0$$

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

R(x) is the radius

P(x) is the axial load due to the weight of the structure above x

u(x) is the deflection

Lambda is the load parameter

FE METHODS

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

$$\tilde{A}_{ij} = \int_0^1 R^4(x) \frac{d^2\varphi_i}{dx^2} \frac{d^2\varphi_j}{dx^2} dx, \qquad 1 \le i,j \le 2 * n_{node}$$

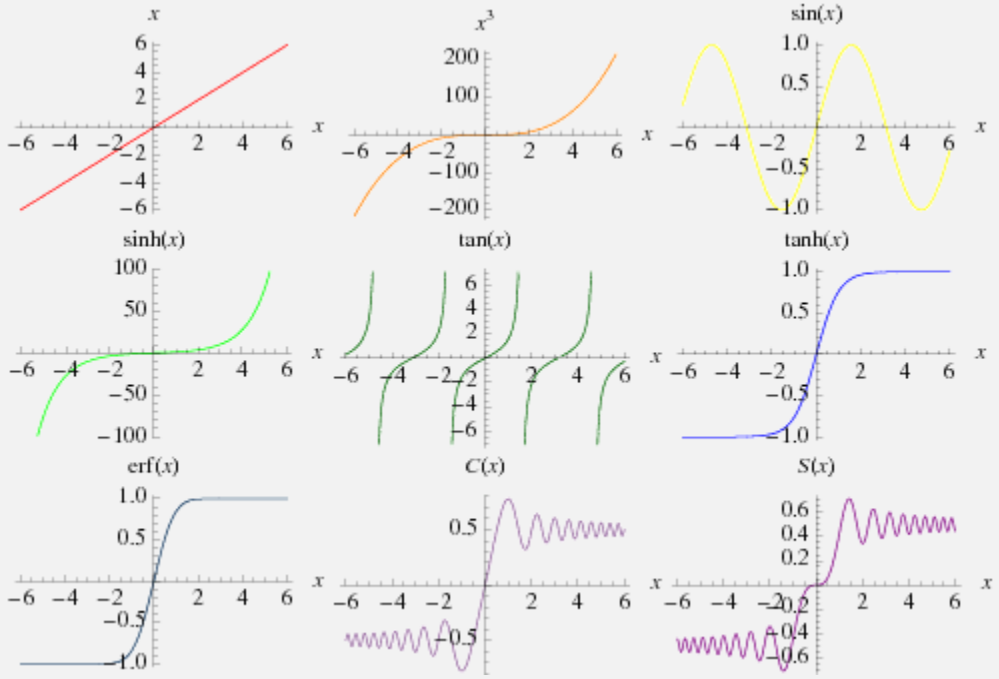
$$\tilde{K}_{ij}^{ax} = \int_0^1 P(x) \frac{d\varphi_i}{dx} \frac{d\varphi_j}{dx} dx \,, 1 \le i,j \le 2 * n_{node}$$

$\underline{A}$ and $\underline{K}^{ax}$ made by removing rows and columns 1 and 2 from $\tilde{A}$ and $\tilde{K}^{ax}$

OPTIMIZATION

- $R(x) = \sqrt{1 + G(x)}\ \ for\ G(x) \succ 1 \quad\quad Choose\ G(x)\ subject\ to$:
- Fixed Volume Constraint: $\int_0^1 G(x)dx = 0$
- Minimum Relative Radius Constraint: $G(x) \geq -1 + R_{min}^2 \equiv G_{min} (\succ -1)$
- Gradual Variation Constraint: $|G'(x)| \leq S_{max}$
- Goal: Maximize L by: Maximizing γ_c

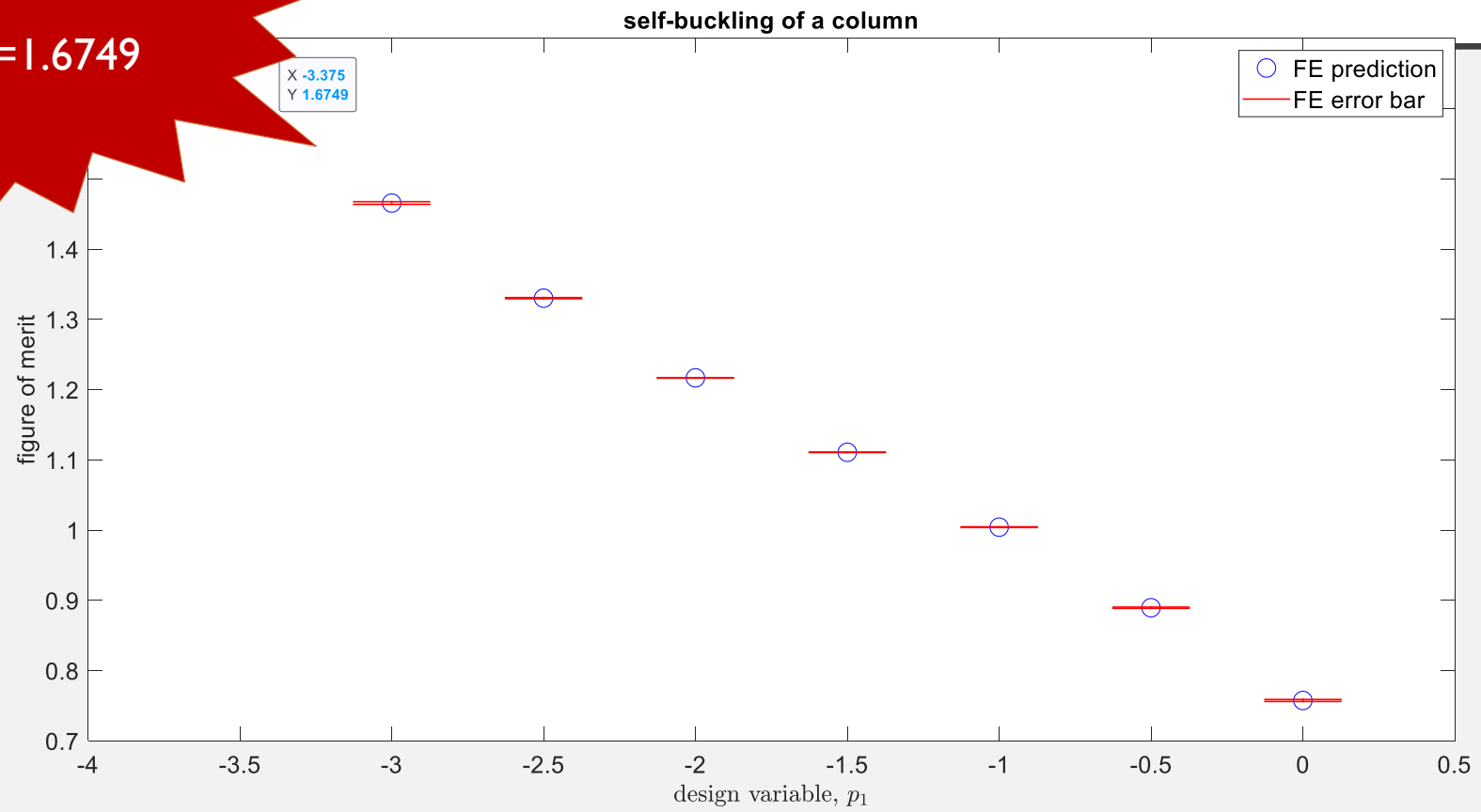
APPROACH



- Visual from Wolfram Alpha http://mathworld.wolfram.com/OddFunction.html
- Designed various odd candidate G functions

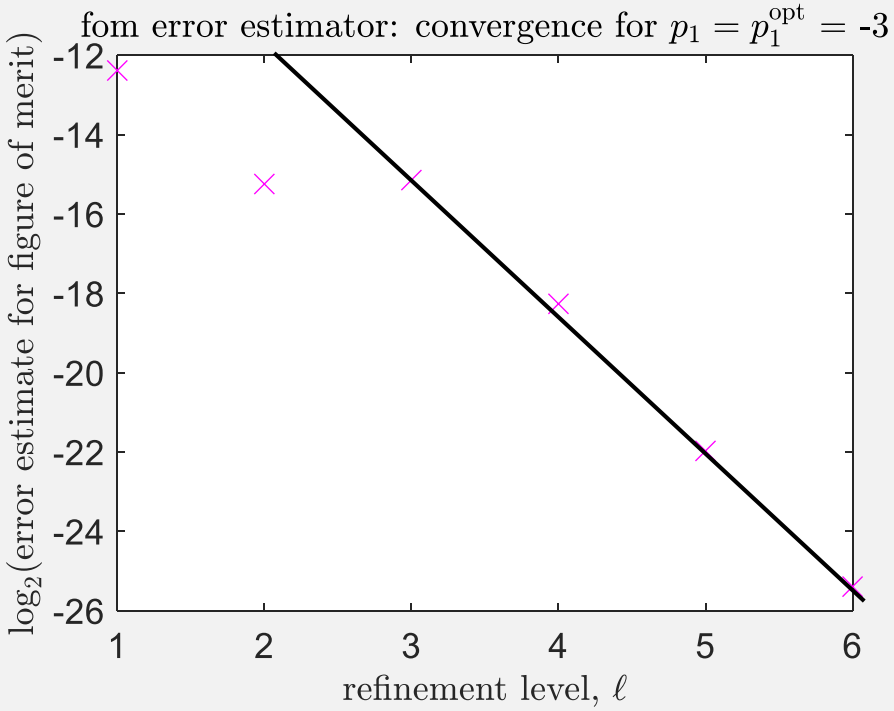
PROPOSED SOLUTION

FOM=1.6749

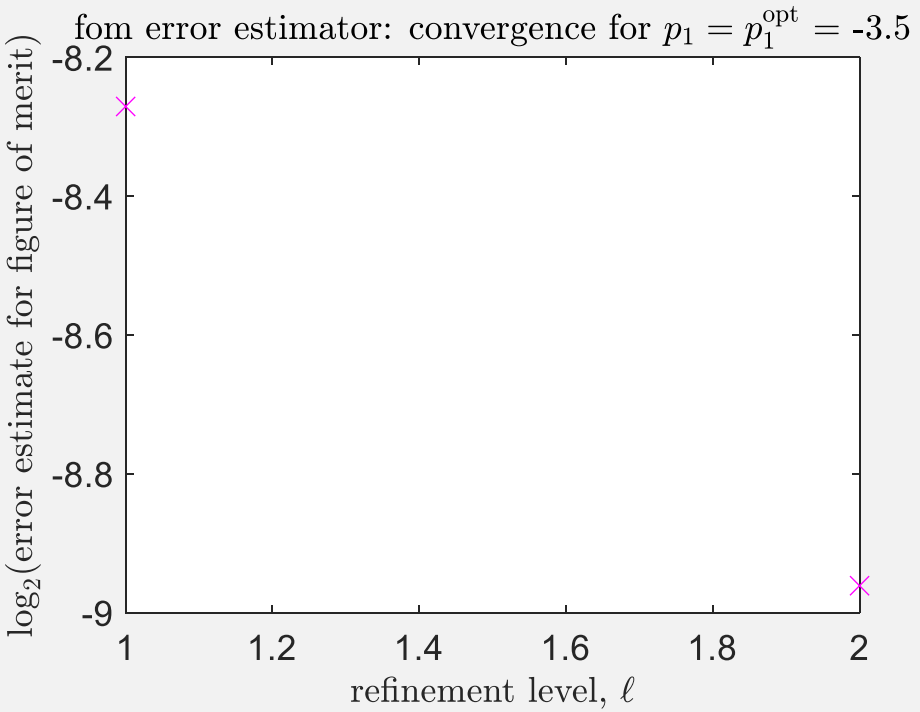


G=p1*tanh(x-0.5)+p2*(x-0.5)^3 p1=-3.5 and p2=6

ERROR

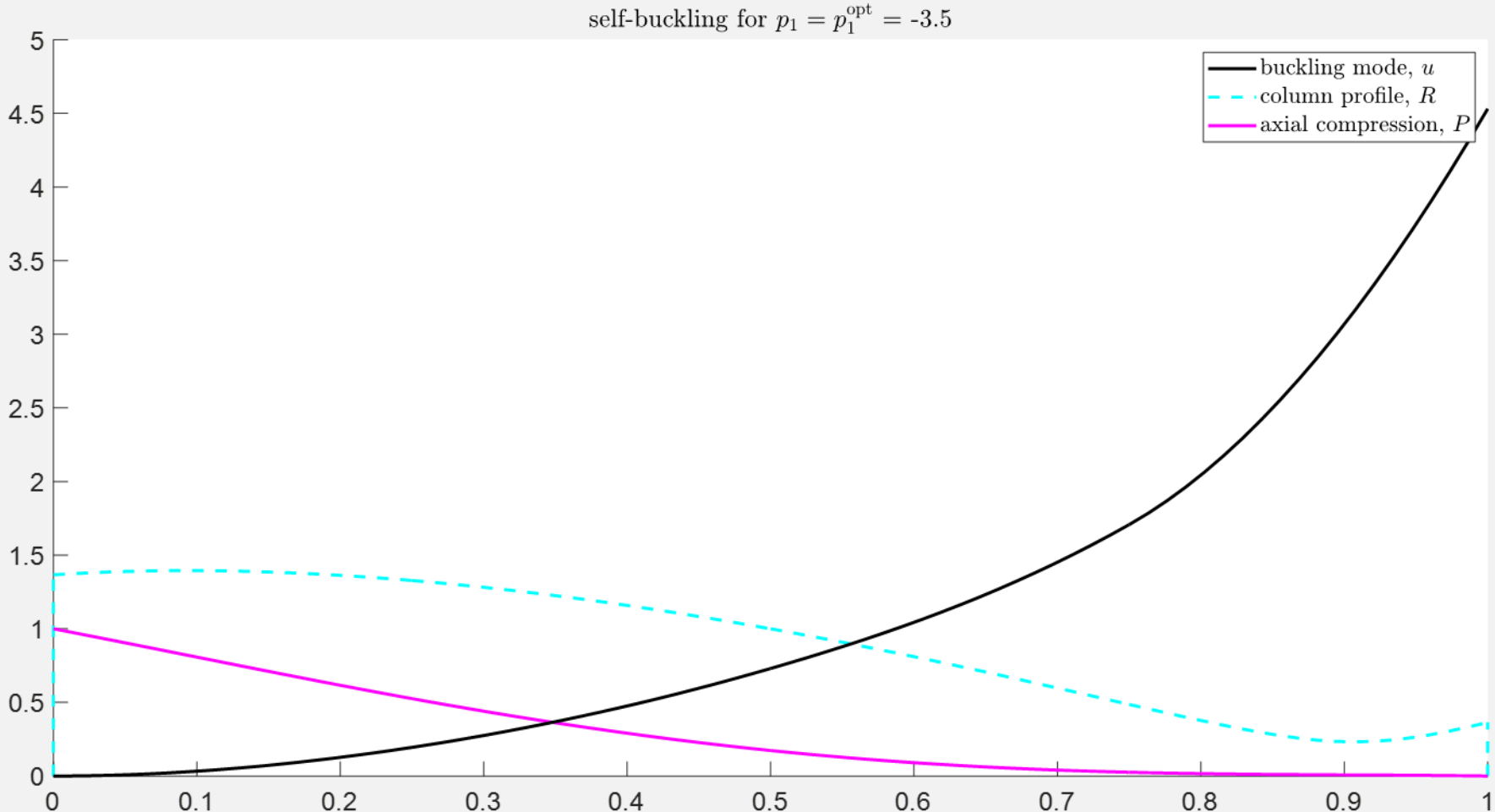


Convergence Chart

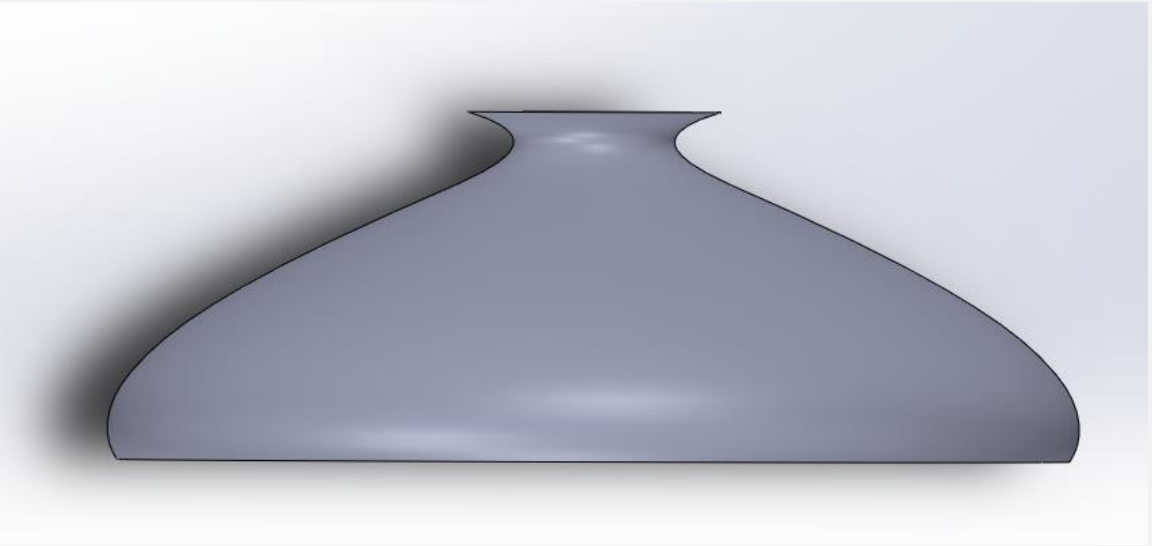


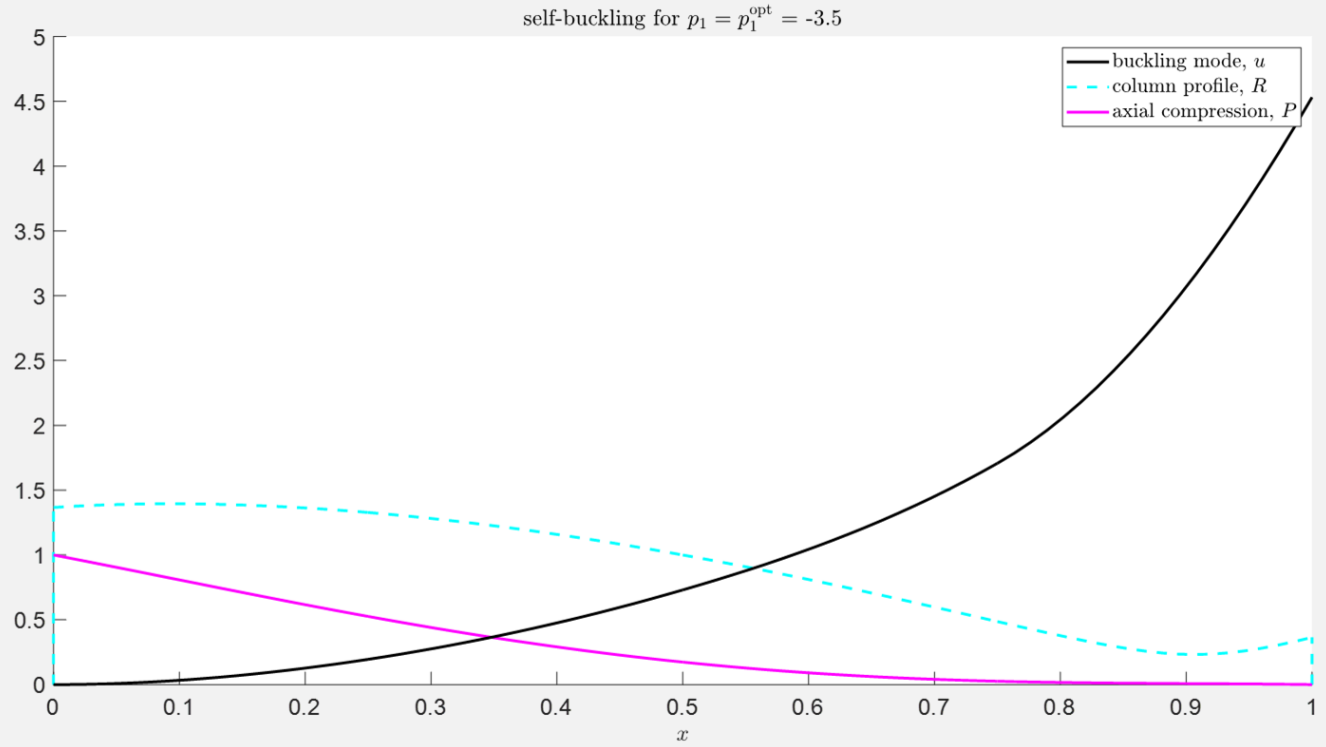
Convergence with 2 Refinements

PROFILE


self-buckling for $p_1 = p_1^{\text{opt}}$ = -3.5
buckling mode, u
column profile, R
axial compression, P
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

DESIGN




self-buckling for $p_1 = p_1^{opt}$ = -3.5
buckling mode, u
column profile, R
axial compression, P
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x