Kip A. Stahlecker

Finite Element Methods in Mechanical Engineering

MIT Mechanical Engineering, Course 2.S976

May 16, 2019

Abstract

In order to explore finite element methods in a mechanical engineering context, this document will explore three different scenarios. Each of these scenarios are designed to be highly physically relevant in order to build and allow for intuitive understanding in order to check the results of the finite element analysis. In the first scenario, finite element analysis is used to calculate the heat transfer through a hamburger throughout the preparation process. The second scenario explores the tuning of a xylophone bar through the use of finite element analysis. Finally, the third scenario explores the optimization of the profile of a tower in order to create the tallest possible tower that will not buckle under its own weight

References

- Course notes and materials for MIT Course 2.S976 created by Prof. A. Patera are referenced extensively throughout this document and are available on stellar.mit.edu
- The website at URL: https://www.americastestkitchen.com/recipes/9182-cast-iron-ultimate-indoor-burgers burger recipe was referenced to obtain a hamburger recipe for analysis in chapter 3

Chapter 3

Throughout the process of exploring finite element analysis over time, two primary models were used in the analysis: semiinf_plus and burger. Detailed descriptions of each of the models can be found in the lecture notes named “Heat Equation: Study Cases”. In order to properly analyze the cooking hamburger, we applied Newmann/Robin boundary conditions on both of the ends of the hamburger. This allowed for application of the theoretical equations in a way that most closely resembled the physical setup of the experiment. Newmann/Robin boundary conditions allow for us to supply the code with a designated heat flux due to the skillet as well as the heat transfer coefficients between the burger and the oil layer as well as between the burger of the air.

Throughout the course of the analysis, the final product is a term $u_{h,\Delta t}^{k}(x)$, evaluated over $1 \leq k \leq n_{tsteps}$. This term stores all of the temperature values at all of the finite element nodes within the burger at each individual timestep. For each change in time (Δt), the values of temperature (u) at each different node location (h) are calculated and stored in the index k.

The model semiinf_plus was analyzed in order to confirm that the $u_{h,\Delta t}^{k}(x)$ term exhibits a proper convergence in error as the number or timesteps and nodes increases. Various analysis schemes were tested for error convergence using different combinations of p and θ values and the results can be seen in Figures 3.1-3.3. In the first case, where p = 1, the error in the L^2 norm should converge to a slope of -2. For the other two cases, where p = 2, the error in the L^2 norm should converge to a slope of -3. These slopes were calculated using the relationship that error in the L^2 norm is

proportional to $2^{-rl}\left(c_1\left(\frac{2^r}{\sigma^q}\right)^l + c_2\right)$. (Within this error proportionality, the 2^r term represents the refinement of equation in space and the σ^q term represents the refinement in time. In order to ensure that both parameters are optimized at the same rate, r and q are chosen such that the $\frac{2^r}{\sigma^q}$ term cancels out.) In each case, the error estimate in the L^2 norm tracked very closely to the actual error, and they exhibited the expected error convergence slope based on the particular scheme.

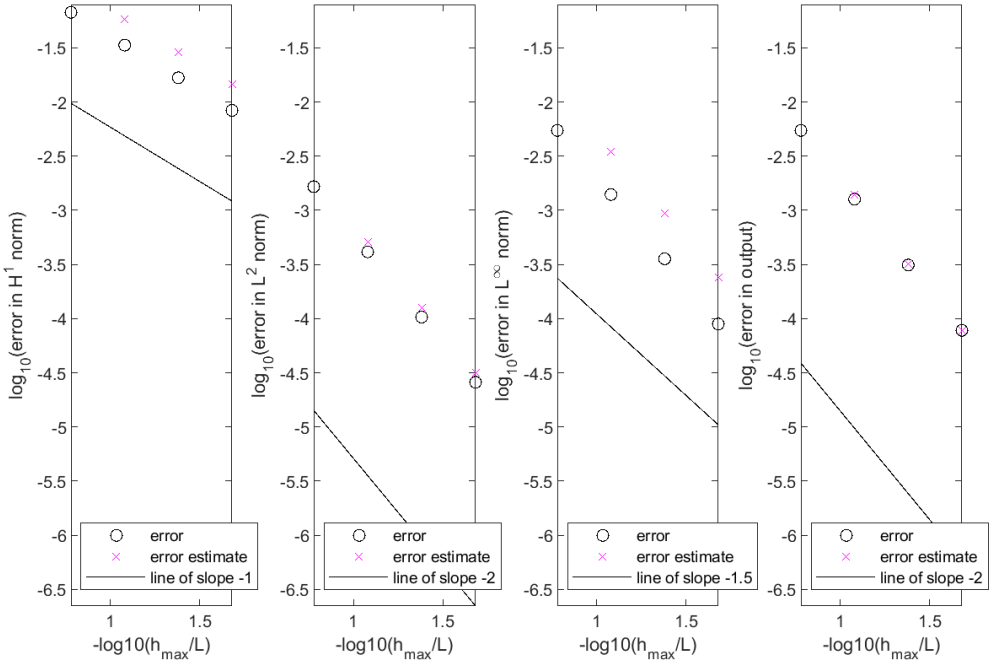


Figure 3.1: Error convergence for p=1, θ=1

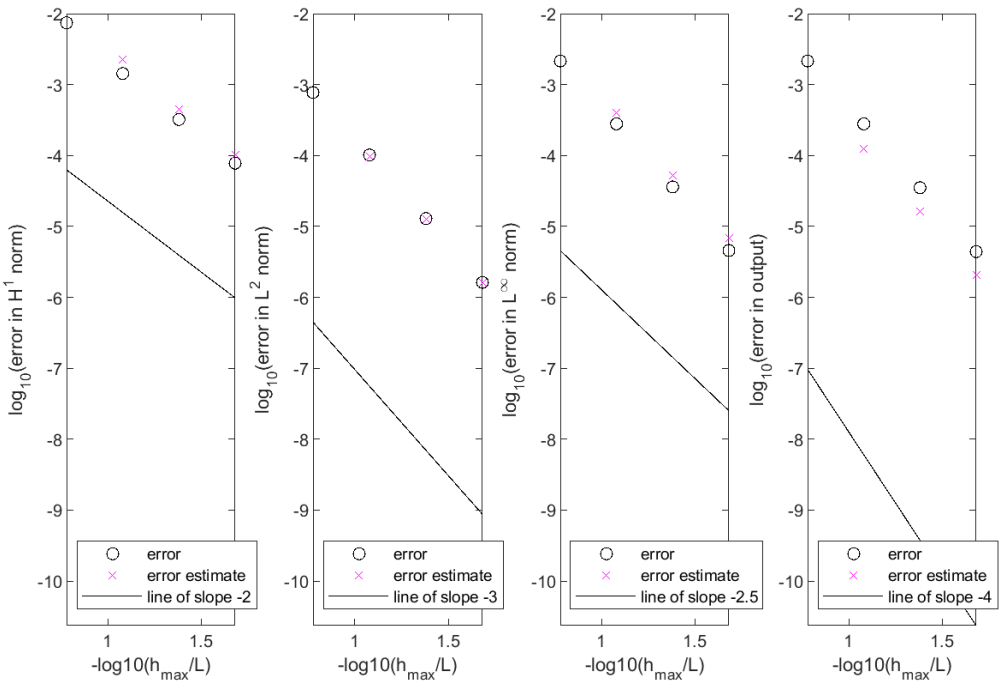


Figure 3.2: Error convergence for p=2, θ=2

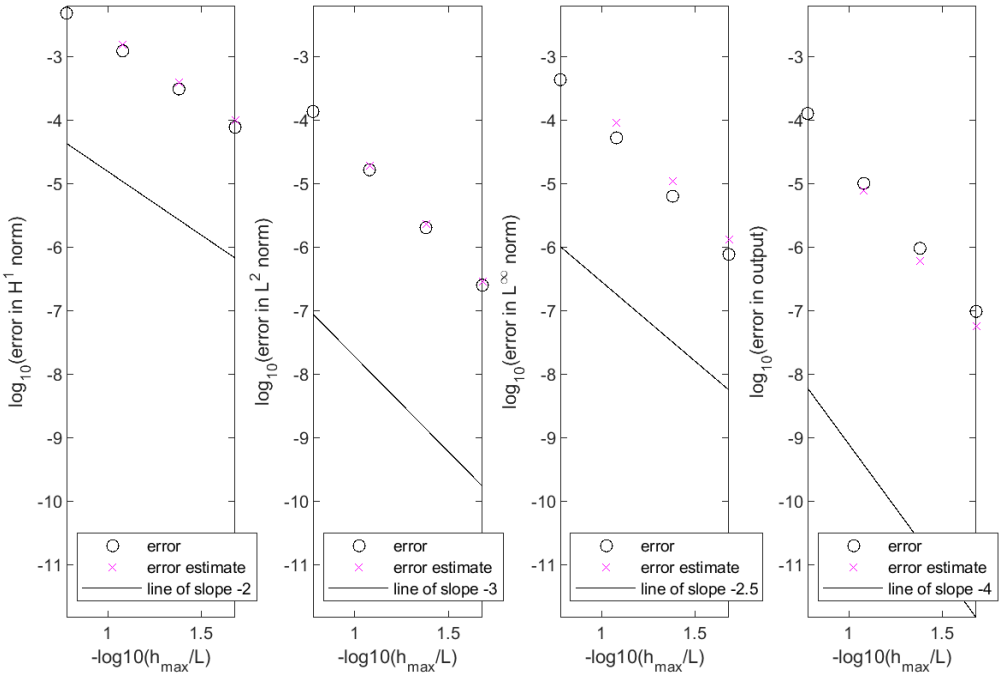


Figure 3.3: Error convergence for p=2, θ=0.5

In order to allow for visual confirmation of proper hamburger analysis, the code produces a graph displaying the temperature of each side of the hamburger as well as the center, overlaid with several reference lines representing specific target values when prepared to the specifications of professional chef Bobby Flay (see Figure 3.4). Initially, the hamburger will start at a temperature below room temperature as it is recommended that raw hamburger is refrigerated prior to preparation of a burger. Next, the side of the burger closest to the skillet can be observed to greatly increase in temperature until it surpasses the Maillard temperature temperature for ideal flavor as the side exposed to air gradually climbs. When the burger is flipped, the side previously exposed to the air rapidly begins to increase in temperature while the already hot side now exposed to air initially rapidly loses heat. Throughout this process, the internal temperature in the center of the burger steadily rises until the burger is removed and placed to cool. During the cooling period, all of the different temperatures gradually and steadily decrease, converging toward the ideal serving temperature.

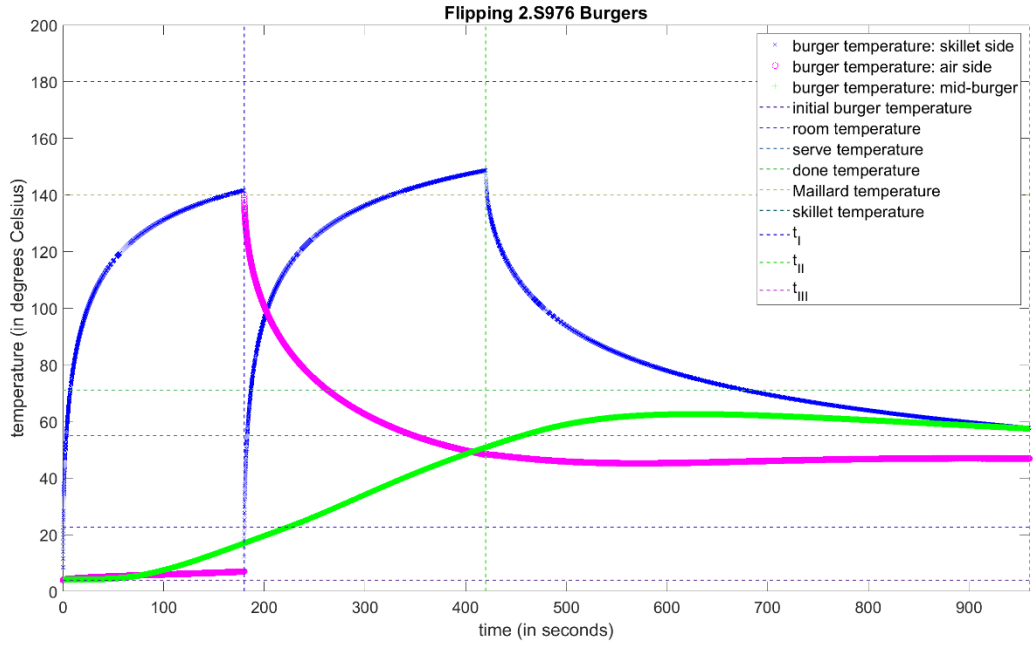


Figure 3.4: Burger temperature over time following Bobby Flay's Recipe

As finite element meshes and the accompanying timesteps over which they are evaluated become more refined, the computational load required in order to solve the problem increases. This is due to the increase in the total number of operations required to solve the problem brought about by each iteration of refinement on the mesh and timestep. Ultimately, the goal of refining the timestep and mesh that a problem is evaluated at is to achieve some desired error tolerance with the final answer. Different computational schemes and parameters produce different amounts of error at different computational cost due to their differing total number of operations.

In a small problem like the one demonstrated here with a relatively small number of total operations and a generous tolerance (when compared to typical engineering applications like those in nuclear reactors or jet engines), even computing to tolerances well beyond what is required, the total computational time is quite short. Using a

reasonably powerful modern laptop, computing the analysis with both analysis schemes [p=1, θ=1 and p=2, θ=0.5] took on the order of 1-3 minutes. However, in the aforementioned engineering applications, as well as most other real-world implementations of finite element analysis, the total number of operations, and thus the total computational time, becomes significant. Choosing an analysis scheme carefully has the potential to save large amounts of time and computational resources on an analysis while still meeting the same pre-determined error tolerances. Table 3.1 shows the process for calculating the computational cost for two different analysis schemes.

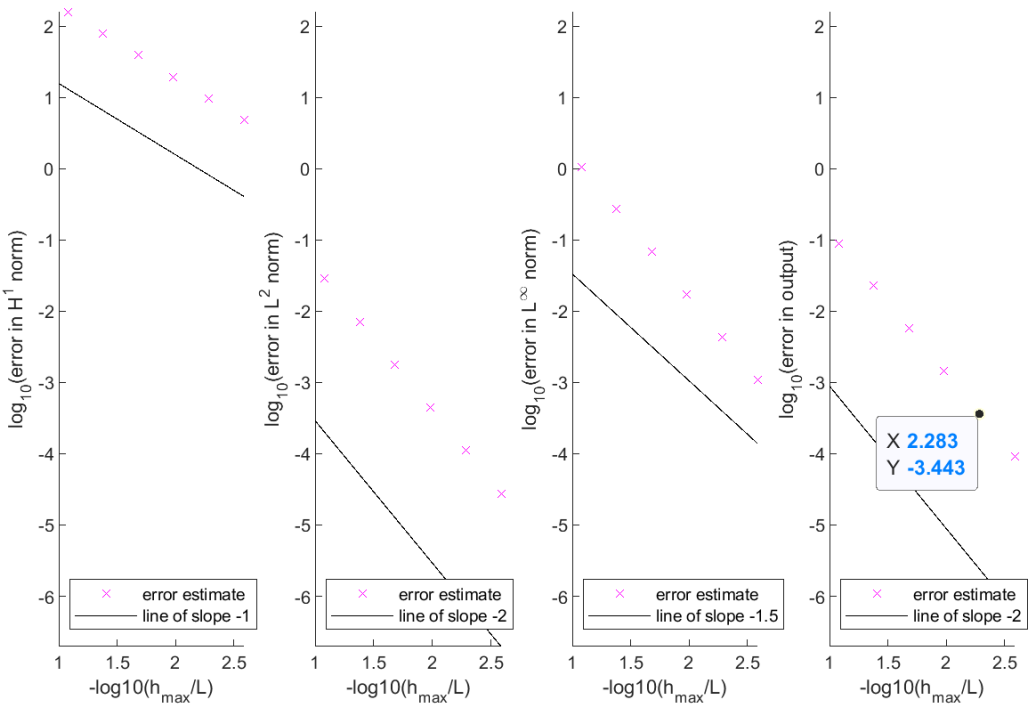


Figure 3.5: Error convergence for p=1, θ=1 highlighting the point where the error dips below the desired threshold

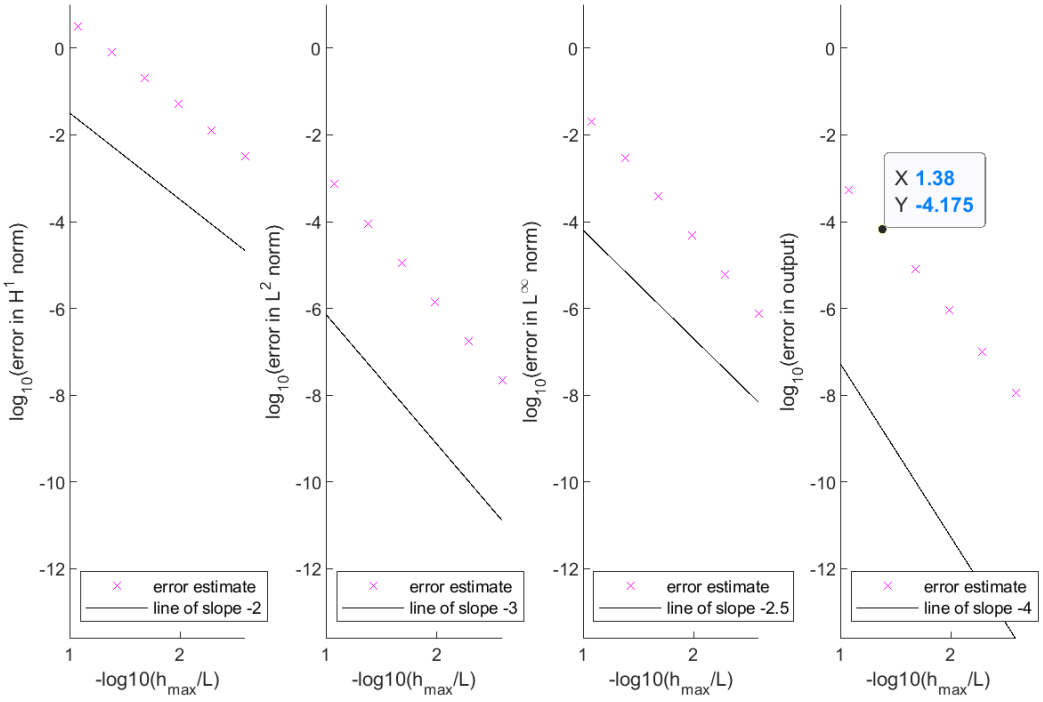


Figure 3.6: Error convergence for p=2, θ=0.5 highlighting the point where the error dips below the desired threshold

[P = 1, θ = 1]	**[P = 2, θ = 1/2]**
6 initial elements, 20 initial timesteps	6 initial elements, 20 initial timesteps
5 refinements required	2 refinements required
Final elements: $6 \times 2^5 = 192$	Final elements: $6 \times 2^2 = 24$
Final timestep: $20 \times 4^5 = 20480$	Final timestep: $20 \times (2\sqrt{2})^2 = 362039$
Overall operations: $192 \times 20480 = \underline{3.93 \times 10^6 \text{ operations}}$	Overall operations: $24 \times 3620.39 = 8.69 \times 10^4$ (multiply total operations by 2 to account for the pentadiagonal structure of the calculation matrix) $2 \times 8.69 \times 10^4 = \underline{1.74 \times 10^5 \text{ operations}}$

Table 3.1: Computational cost calculation examples

In order to further explore and understand the finite element code, I tested the analysis on a hamburger recipe found on the internet. The results are shown in Figures 3.7 and 3.8. Much like the recipe from professional chef Bobby Flay used in the previous hamburger analysis, this recipe manages to achieve many of the targets for hamburger preparation. In particular, each side is brought comfortably above the Maillard temperature, and the five minute repose time is sufficient to allow the hamburger to come very close to the recommended serving temperature. However, much like the Bobby Flay recipe, the center temperature also does not reach the USDA recommended temperature at any time throughout the cooking process, although the recipe that I found from searching on the internet does bring the hamburger's center to a temperature closer to the recommended temperature more consistently and for a longer time.

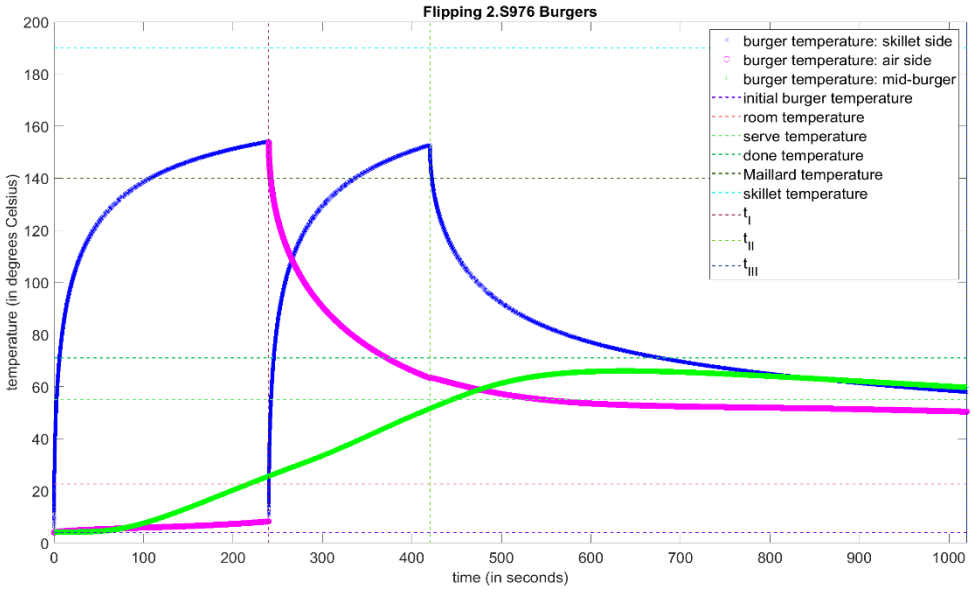


Figure 3.7: Burger recipe temperature profile over time

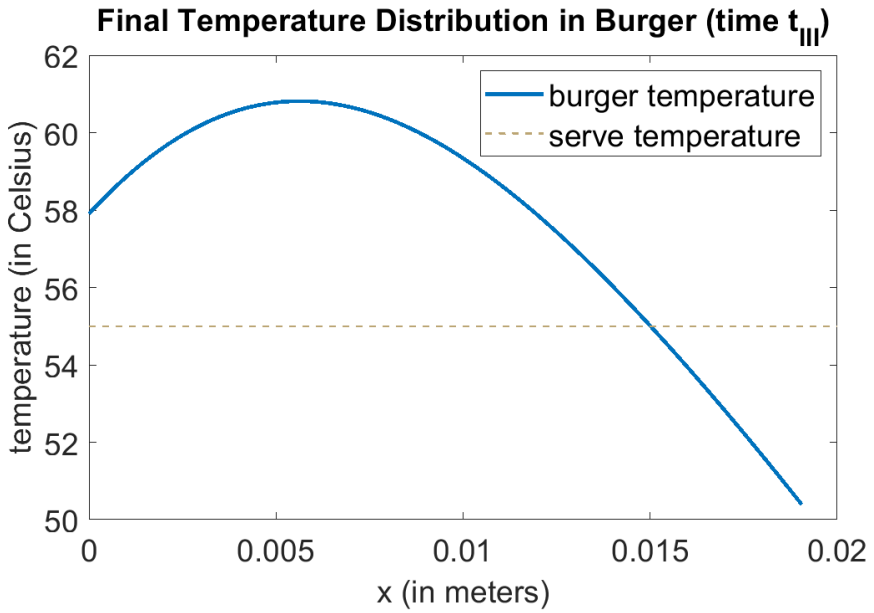


Figure 3.8: Burger recipe cross-section temperature distribution

Chapter 4

Beam eigenproblems are constructed and solved similarly to other finite element problems that discussed in previous chapters. These problems are separated into a finite element mesh and described by a combination of phi functions as before, however, in the case of beam eigenproblems, each node relates to two different phi functions relating to two different degrees of freedom: displacement and slope. At each node, all phi functions have a value of zero except one which is normalized to a value of one. Similarly, all phi functions have a derivative of zero except for one which has a normalized value of one (see Figure 4.1)

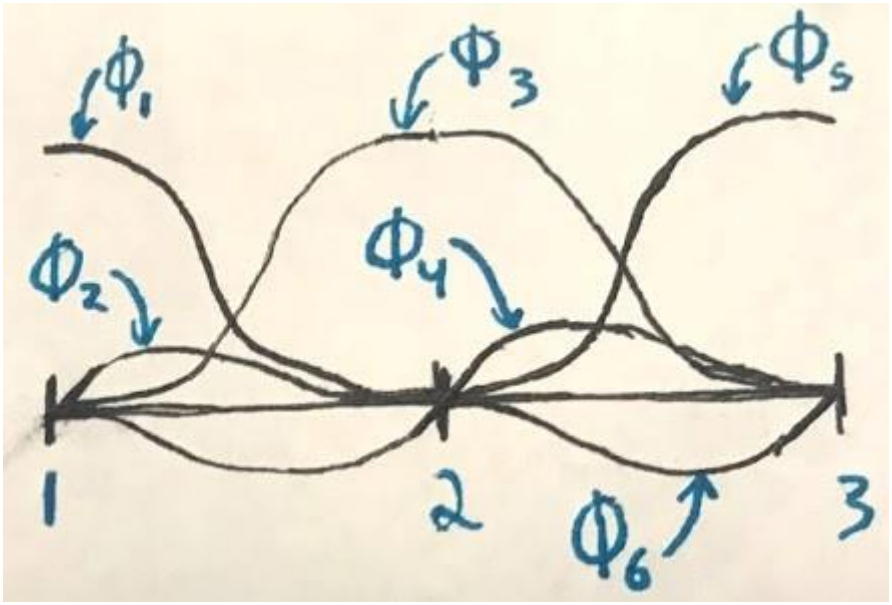


Figure 4.1: Example showing the phi functions for a bar divided into two elements.

Another important consideration in this finite element implementation of beam eigenproblems is the nondimensionalization of both the bar as a whole as well as an individual finite element. This allows for not only easier calculation within the code, but more importantly it allows for any analysis and calculation to be universally applicable

for given input parameters, regardless of scaling. Once a particular analysis is performed, the relations between different parameters can be observed and modified without requiring the analysis to be performed again. Once all desired analysis has been performed, the nondimensionalization can be applied again in reverse in order to undue those operations and scale the result back to the desired dimensional scale.

In this chapter, the task is designed around optimizing the design parameters of a xylophone bar in order to tune it for its desired use case given particular parameters. The first and most immediately noticeable of the design objectives of a xylophone bar is the fundamental frequency with which the bar will vibrate when struck. This frequency determines what musical note the xylophone bar will most closely resemble. While other factors can impact the fundamental frequency of a bar, most factors outside of the length of the bar are held constant across all of the different bars in a particular xylophone. In particular, considerations such as the cross-sectional profile of the bar as well as the material the bar is constructed from are held constant throughout all of the bars on a xylophone, leaving only the length of a bar to determine its fundamental frequency. This is the case for the problem explored in this chapter. The non-dimensionalized length of the bar is optimized to produce the desired fundamental frequency.

Along with the fundamental frequency of the xylophone bar, the harmonic frequencies that a bar vibrates at will also have a great impact on the musical quality of the instrument and how it is perceived by listeners. For example, a frequency ratio of 4 (“double-octave”) is very typical of other orchestral instruments whereas a frequency ratio of 3 (“quint”) has a different auditory quality, allowing the xylophone to sound

distinct and noticeable even when combined with an orchestra. This parameter is tuned by removing material from the underside of a bar. Since the overall length of the bar does not change through the process of removing material from the underside of the bar, the fundamental frequency remains the same but the first harmonic frequency will be impacted as mentioned before.

A final consideration for xylophone bar design is that of physically holding the bar in place so that it may be played. Holding a xylophone bar at an arbitrary location will cause damping effects on the vibration of the bar, impacting sound quality. In order to avoid these negative impacts, the nodes of the bar must be identified. These nodes are the locations along the bar where, due to the characteristics of the bar's vibration, there is no displacement of the bar when vibrating. If holes are drilled through the bar at these nodes, it can be mounted without impacting the vibration due to the fundamental frequency (and thus the sound quality) of the bar.

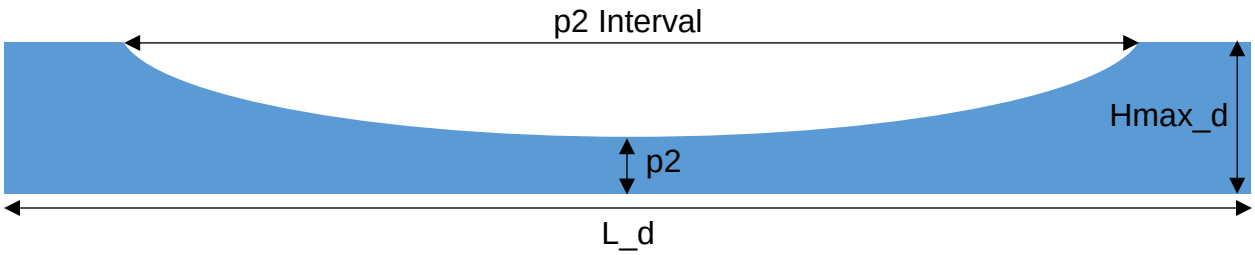


Figure 4.2: Xylophone Bar Dimensions. Width of the bar (not used in calculations) is measured into the page

Finding the nodes in a xylophone bar due to its fundamental frequency is one of the primary functions of this finite element code. In the case of this problem, the nodes on a xylophone bar are physically represented as the points on the bar where the displacement of the bar due to vibration at its fundamental frequency is zero. Computationally, this was accomplished by solving the beam eigenproblem:

$\frac{d^2}{dx^2}\left(\frac{H^3(x)}{12}\frac{d^2u^{(k)}}{dx^2}\right) = \lambda^{(k)}H(x)u^{(k)}$. The values of λ are the nodes of xylophone bar and can also be represented graphically through observing the plot of displacement of the bar in the y axis along the length of the x axis when it is vibrating at its fundamental frequency. As a part of the finite element analysis, the bar is separated into a mesh of distinct finite elements. Since these elements span the entire length of the bar, there will be two distinct elements that contain the nodes, represented graphically as the displacement graph crossing through y=0. In order to identify these elements, I wrote an algorithm to check each element by evaluating the product of the displacement at its left node with its right node, accomplished computationally through the formula: $u_h^{(3)}lg2(1,m^*) \times u_h^{(3)}lg2(3,m^*) < 0$. This inequality will only be true in the cases where the displacement graph crosses zero within the element because only then will the value at the left node ($u_h^{(3)}lg2(1,m^*)$) have a different sign than the value at the right node ($u_h^{(3)}lg2(3,m^*)$). (While it is true that this code is susceptible to producing inaccurate results in the case where the displacement graph crosses y=0 twice within a single element, that is not a concern in this particular problem because the fundamental frequency is a gradual enough curve and even the initial mesh is sufficiently fine in order to ensure this inaccuracy is avoided.) Once the elements where the displacement graph crosses zero are identified, my code then uses the built-in matlab function fzero over only those two elements in order to identify the x values of the two nodes along the individual finite element within which they occur. Using this value of distance along the particular finite element as well as the location of the start of that element, I then calculated the distance along the non-dimensionalized bar where each node occurred. This could then be re-dimensionalized in order to find out the locations of the nodes in

the final bar. This final placement calculation and re-dimensionalization was accomplished through the equation: $x_{dh}^{hole} = \left(x^{lg(1,m^*)} + h^{m^*}\hat{x}^{hole}\right)L_d$ where $x^{lg(1,m^*)}$ is the location of the start of the finite element that contains the node, h^{m^*} is the length of an individual finite element, $\hat{x}^{hole}$ is the distance along the particular finite element where the node occurs, and L_d is the length of the xylophone bar used to re-dimensionalize the problem.

Once the nodes are identified, the remaining code is able to continue and complete the optimization of the xylophone bar and present those results graphically (see Figures 4.3 and 4.4). Overlaid on top of the cross-sectional profile of the xylophone bar is the graph of the displacement due to the fundamental frequency (as well as the first harmonic frequency) and two lines representing the identified node locations. The identified node locations can be seen on the plot to align with the points where the displacement graph crosses zero. This is exactly what was intended from the code, lending credibility to the argument that my algorithm is implemented correctly.

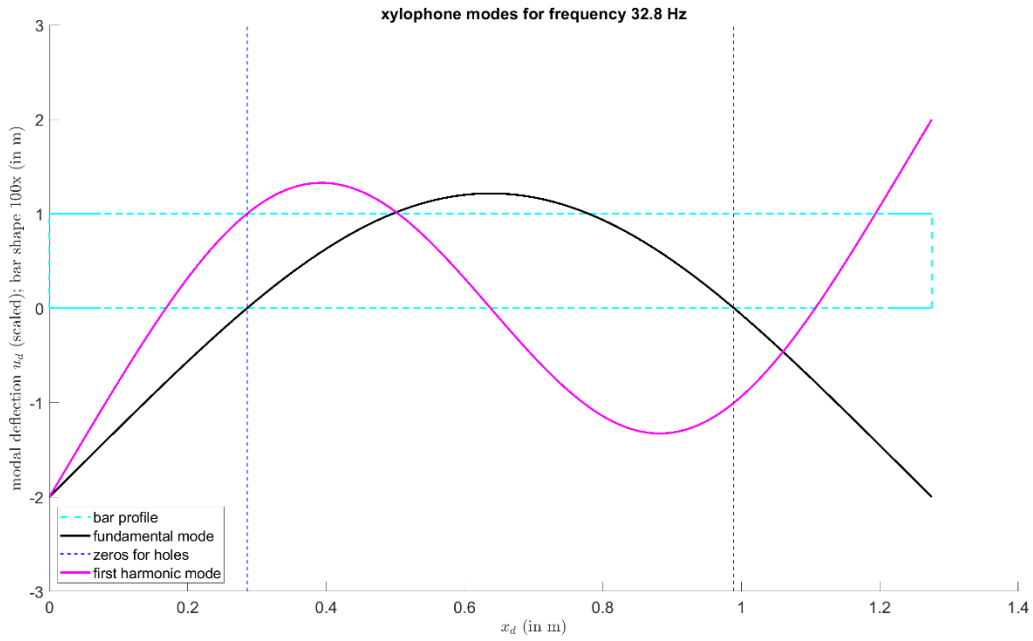


Figure 4.3: Optimized node locations using Caresta's experimental setup

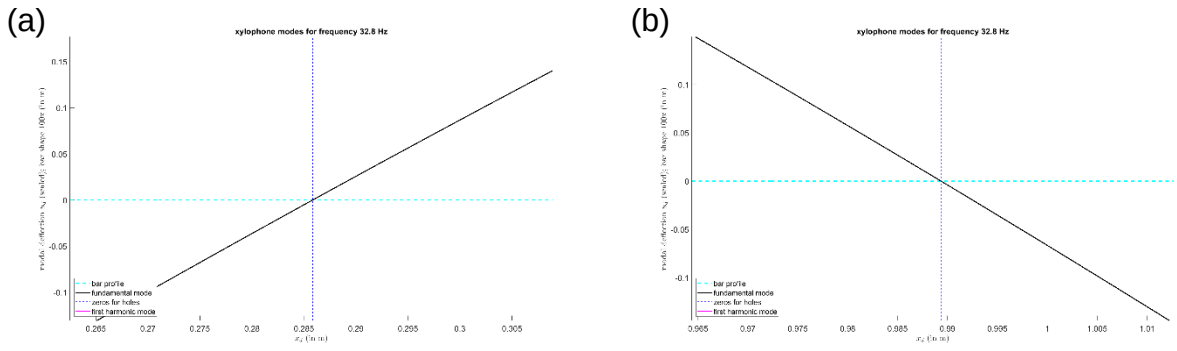


Figure 4.4: Zoomed in views of the left (a) and right (b) nodes identified previously verifying that the fundamental frequency crosses zero at each node.

In order to verify correct implementation and operation of the analysis algorithm, experimental data gathered by Mauro Caresta was used. Caresta experimentally and theoretically determined the fundamental and first four harmonic frequencies of a bar of steel with a given length, density, cross-section, and Young's modulus. The finite

element code, if all parts are correctly implemented, will determine the first harmonic frequency and optimal length in order to achieve a target harmonic frequency (given the cross-section, density, and Young's modulus). Caresta calculated that his bar (of length L = 1.275m, thickness Hmax_d = 0.01m, density rhobar_d = 7800 kg/m^3, and p2 = 1 indicating that the bar had no material removed from it) had a theoretical fundamental frequency of 32.80Hz and a first harmonic frequency of 90.44Hz. Experimentally, Caresta observed a fundamental frequency of 32.25Hz and a first harmonic frequency of 88.50. After running the code with the given input values taken directly from Caresta's experimental setup (aiming for a fundamental frequency of 32.80Hz), the code calculated that the optimal length would be 1.2752m with a first harmonic frequency of 90.4145Hz. Both of these values (and the length in particular) are very close to the theoretically calculated values determined by Caresta lending a large amount of credibility to the fact that the code is implemented correctly.

After building confidence that the finite element code has been implemented properly, I used the code to optimize the parameters for a xylophone bar tuned to C5 (frequency3target_d = 523.25Hz) with "quint" tuning (R_target = 3). Given these inputs, my code has determined the following optimized values (displayed in table 4.1): (also see Figure 4.5)

Quantity	Variable Name	Value
Fundamental Frequency	frequency3_d	8.7756×10^{-6}Hz
First Harmonic Frequency	frequency4_d	1.4854×10^{-4}Hz
Optimized Bar Length	L_d	0.2667m
Optimal p2 Length	p2opt	0.6438
Frequency Ratio	ratio_calculated	2.9913

Table 4.1: Optimized xylophone bar parameters

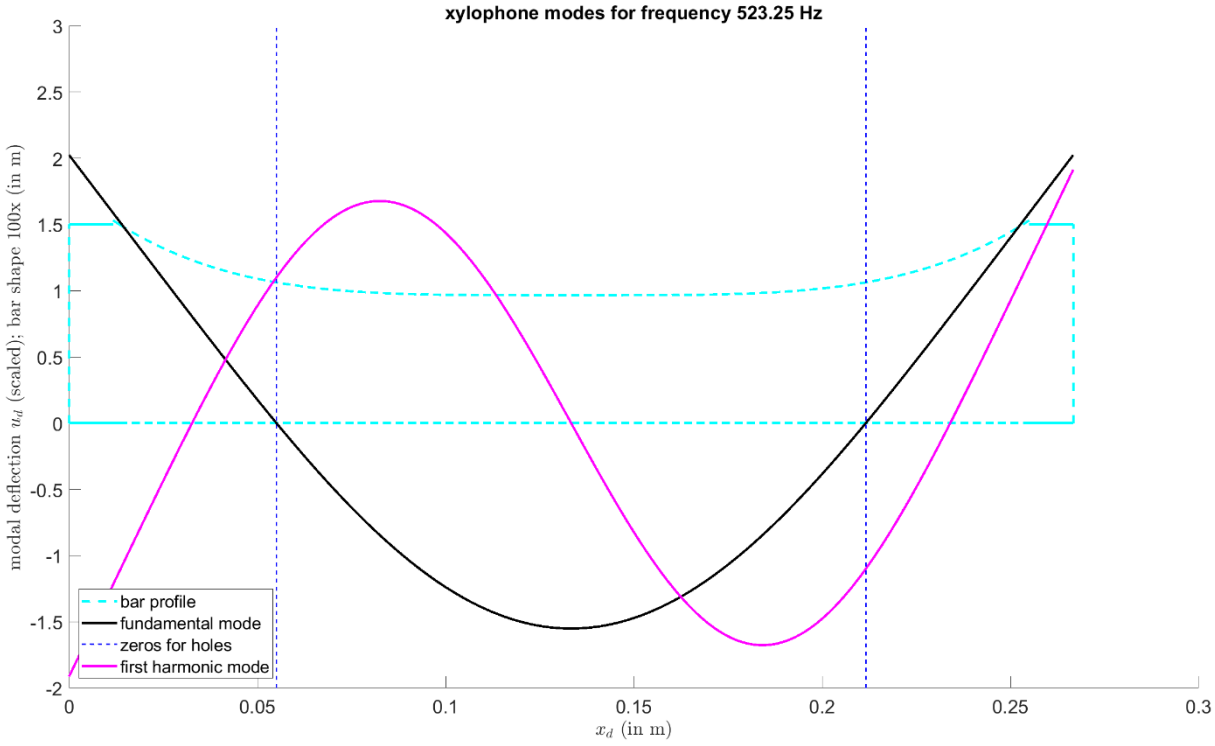


Figure 4.5: Computed xylophone bar optimization for desired parameters

In terms of accuracy, both the fundamental frequency and first harmonic frequency experience very little estimated error as shown above. These values are

calculated directly in the code along with the rest of the finite element analysis that is performed. A secondary step was required in order to calculate the error estimate for the frequency ratio. With no direct way to estimate the frequency ratio error, the estimate listed above was derived from the previously calculated error estimates for the fundamental and first harmonic frequencies. Using the highest possible value for the fundamental frequency allowed by the error estimate and the lowest possible value for the first harmonic frequency allowed by the error estimate, the largest possible value for the frequency ratio (with the given error bounds) can be calculated. A similar strategy was used to calculate the smallest possible value for the frequency ratio by using the smallest allowed fundamental frequency and largest allowed first harmonic frequency. Comparing these two values to the calculated frequency ratio, the largest difference between the calculated value and the maximum and minimum values was used in order to determine the final error estimation for the frequency ratio.

In cases where the exact solution is not known, it is impossible to determine with certainty if the FE error estimators are reliable. However, a number of factors can lend credibility to the accuracy of these estimators, the primary of which is the plot of error estimations produced by the code. Observing the error estimation plots (see Figures 4.6 and 4.7) reveals that the error is consistently decreasing with the increase in refinement of the finite element mesh, and this decrease in estimated error follows the expected slope. This, combined with the fact that the code performed admirably in the test against the Caresta experimental data and the lack of any other obvious red flags, lends great credibility to the accuracy of the FE error estimators.

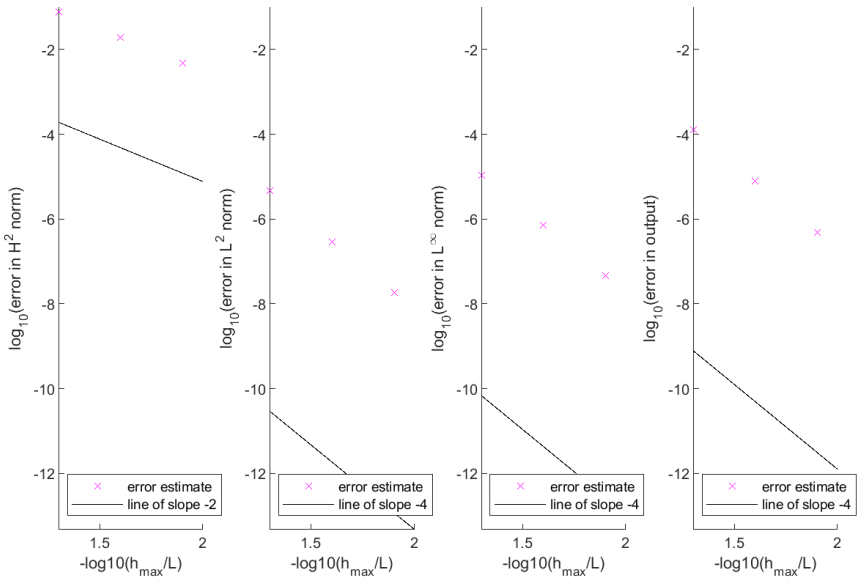


Figure 4.6: Error estimation plots for the fundamental frequency

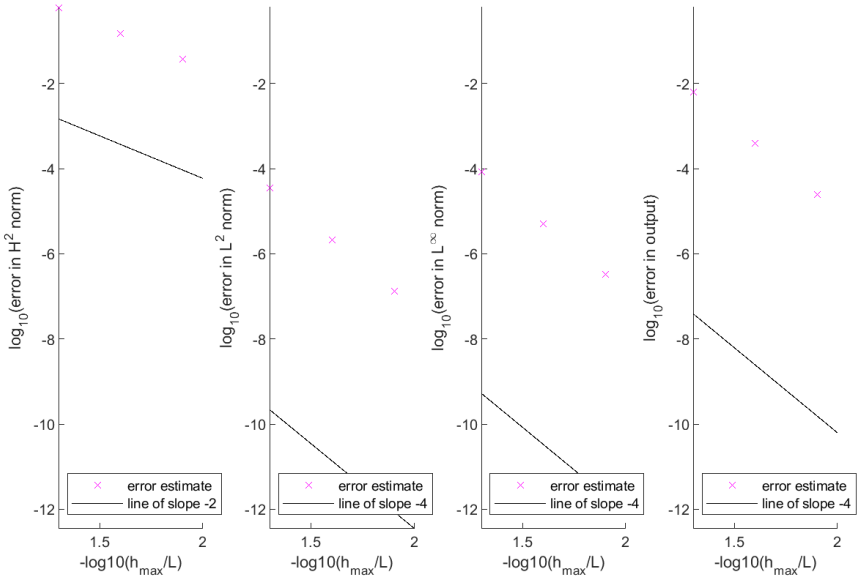


Figure 4.7: Error estimation plots for the first harmonic frequency

Given the same mesh for this xylophone bar optimization problem, the FE error for the fundamental frequency (denoted in the code as frequency3_d) is expected to be smaller than FE error for the first harmonic frequency (denoted in the code as

frequency4_d). My primary reasoning supporting this claim is the fact that, by definition, the fundamental frequency will be physically represented in the bar (through the displacement of different parts of the bar) by a curve that is “less complex” than the curve representing the physical displacements due to the first harmonic frequency. In this case, the term “less complex” refers to the fact that the curve for the fundamental frequency over the length of the bar is an overall smoother curve (primarily seen through the presence of fewer inflection points) than that of the first harmonic frequency. (Figures 4.3 and 4.5 show these curves overlaid on top of a xylophone bar profile.) The type of finite element analysis that has been implemented can be thought of as analogous to linear interpolation in this particular case with regards to error estimation. Given the two different functions, a linear interpolation will be better able to much more accurately approximate the simpler function when evaluated at any different precision of mesh. Analogously, the first harmonic frequency (represented by the more complex curve) will experience a larger FE error than the fundamental frequency.

Despite these small differences in accuracy, both the fundamental and first harmonic frequencies are determined with incredible precision. The error for both of the frequencies is on the order of 10^{-7}Hz. Untrained human ears, however, are only able to perceive differences in frequency that are around 10Hz or greater. Essentially, this physical human limitation renders the calculated frequencies 7-8 orders of magnitude more precise than is strictly necessary in order to ensure that the xylophone sounds like it is designed to.

The aforementioned error is only due to the computational methods used. Another source of error is the theoretical model used to describe the xylophone bars

themselves. In the code, the xylophone bars are modeled as Euler-Bernouli beams. Principle among the requirements for a bar to be accurately modeled as an Euler-Bernouli beam is that the bar is long and slender. How long and slender a bar is can be determined quantitatively by examining the ratio between the width or height of the bar compared to its length. The Euler-Bernouli beam theorem requires that the width and height be much less than the length. The larger the difference between these values, the more accurately the tenants of the Euler-Bernouli beam theorem describe a particular beam. The height and width of all of the bars on a particular xylophone are consistent; only the length changes from one bar to another. Thus, those bars that are the longest will be the most accurately represented by the Euler-Bernouli beam theorem. All other factors held constant, longer xylophone bars will exhibit a lower fundamental frequency than shorter bars, meaning that bars tuned to lower frequencies will most closely mimic Euler-Bernouli beams and will lead to the most accurate predictions.

In order to model the impact that support strings have on each of the xylophone bars, a small section of a bar terminating at the point where the first string connects is analyzed (see Figure 4.8). The impact of the string on the vibration of the xylophone bar can be simplified and represented simply as a spring. At all other points, the xylophone bar portion is modeled as being free, analogous to earlier portions of this chapter. Under this model, the support string only imparts a force on the bar portion at the point where it connects. Thus, when evaluating the stiffness matrix (A) including the effects of the support string, the only entries that will be changed are those that correspond to the displacement of the node at the end of the bar segment. Within the finite element

matlab code, calling the function ttomap_fcn(n_el0+1,1) will return the index of where the displacement data for the node n_ele0+1. The node at the end of the bar segment (which is connected to the spring) will always be numbered n_ele0+1 due to the pre-determined naming scheme established throughout the rest of the code. When the initial finite element mesh is established, the nodes are numbered sequentially, leaving the last node to be numbered one greater than the number of elements. This label stays consistent because as the mesh is refined and new nodes are added between existing nodes, all previous nodes retain their previously defined names and the new nodes are numbered sequentially starting after the last assigned label. Furthermore, at each node, there are two different pieces of information stored: the displacement of the bar at that node and the slope of the bar at that node. The second input of the aforementioned function indicates to the function that the displacement (as opposed to the slope) is requested.

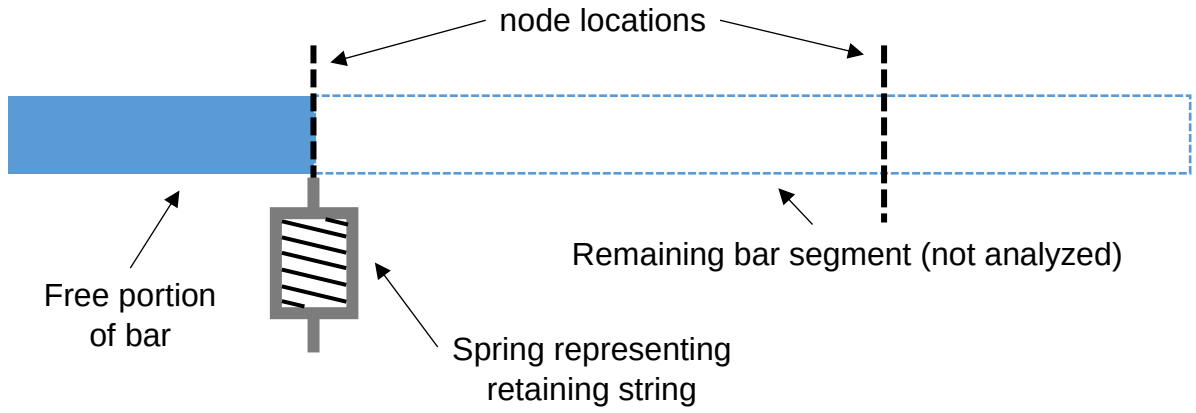


Figure 4.8: Analysis problem setup

Once the index of where the value of displacement at the end of the bar is stored is determined, that will allow the code to index correctly into the stiffness matrix in order

to modify only that particular value. Within the code, a potential location of where to make this modification to account for the support string is within the file impose_boundary_cond.m after line 67 when the A matrix is defined. At this point in the code, it is a simple matter to add the spring constant (k_s) to the entry in the A matrix at the index i and j, which are both equal to the index determined above. This has the desired effect because at this point in the code, the stiffness matrix is normalized, meaning that at the desired node, there is only one phi function representing the displacement and one phi function representing the slope of the bar that are not zero for their respective values (see Figure 4.1 above). Furthermore, the phi function representing the displacement is at a value of 1, as is the derivative of the phi function representing the slope. When the spring constant is added to the stiffness matrix at this point in the code, it will eventually fully represent the force imparted on the bar by the spring when the stiffness matrix is fully unpacked and realized. Also, due to the fact that of all of the phi functions representing the bar at the desired node have zero displacement except for one, this term is the only term in the entire matrix that needs to be modified by the spring constant in order to capture all of the effects of the support string.

Kip A. Stahlecker

2.S976 Chapter 5: Buckling Eigneproblems and Self-Buckling Beams

Self-Buckling Summary

- Condition where beam placed in gravitational field buckles under its own weight
- Distinct from deflection
  - Deflection has one unique solution, buckling has infinite
- Characterized by:

$$\begin{cases} \frac{d^2}{dx^2}\left(R^4\frac{d^2u}{dx^2}\right) = \lambda\left(-\frac{d}{dx}\left(P\frac{du}{dx}\right)\right) & 0 < x < 1 \\ u = u_x = 0 & at\ x = 0 \\ u_{xx} = (R^4 u_{xx})_x = 0 & at\ x = 1 \\ and\ normalization\ of\ u \end{cases}$$

Where P(x) is the axial force due to the weight of material above position x along the beam

Summary of FE Methods for Self-Buckling

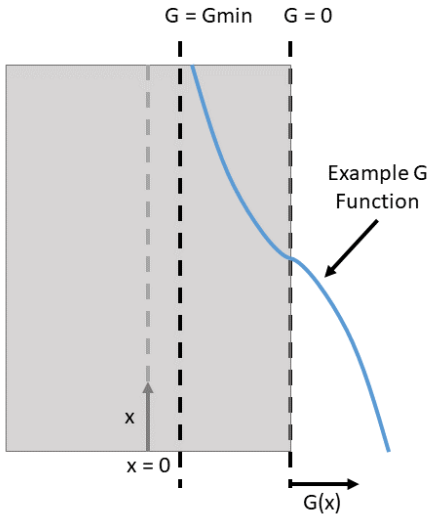
- Solved using similar methods explored in Chapter 4
- Characterized by the eigenvalue problem: $\underline{A}\,\underline{u_h^0} = \lambda_h\,\underline{K^{ax}}\,\underline{u_h^0}$
  - $\underline{A}$ is the matrix containing the 4th derivative terms
  - $\underline{K^{ax}}$ is the matrix containing the 2nd derivative terms
  - λ_h is the eigenvalue, representing the force at which the beam buckles
  - $\underline{u_h^0}$ is the function representing the buckling mode along the beam
- The 4th derivative terms represent the shear force (3rd derivative terms) over differential elements along the length of the beam
- The 2nd derivative terms represent the moment produced by forces offset from the axis of symetry

Optimization Problem Summary

- Define a non-dimensionalized radius that will allow creation of a G function used to modify the radius along x
  - The G function depends on x and one or more additional parameters used in order to vary the function and facilitate optimization
- Figure of merit (FoM):
  - Defined as $\mathrm{FoM} = \frac{\text{length obtained using given profile}}{\text{length obtained using cylindrical profile}}$
  - Larger is better
  - FoM for cylindrical profile is 1

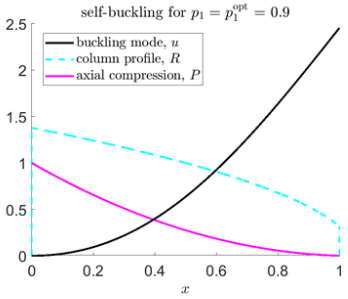
Design Constraints

- Pre-defined material volume
  - Maintained by ensuring that the integral of the G function over the profile is zero
- Maximum allowed beam profile derivative (Smax)
- Minimum required beam radius (Gmin)
- Circular cross-section

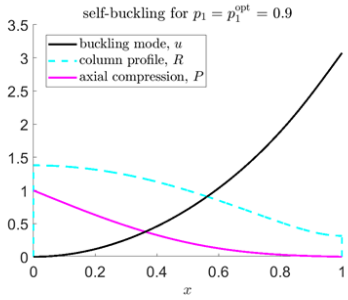


Obtaining G Function Profile

- Combined two G functions with highest figure of merit found when examining simple odd functions

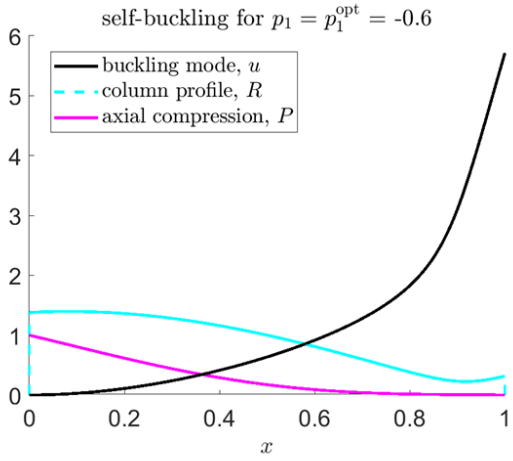


- $G = (-2p1x + p1), p1 = 0.9$
- FoM = 1.4283



- $G = \left(p1 \sin\left(\pi x + \frac{\pi}{2}\right)\right), p1 = 0.9$
- FoM = 1.5736

Final G Function Profile and Figure of Merit



- G Function:
  - $G = (-2p1x + p1) + \left(p2\sin\left(\pi x + \frac{\pi}{2}\right)\right)$
  - $p1 = -0.6$
  - $p2 = 1.5$
- Figure of Merit Value: 1.6877

Error Estimate

- Error should converge to a slope of -4
- After 5 levels of refinement, converges to desired slope
  - Additional levels of refinement to verify convergence were only calculated for the calculated optimal values for p1 and p2