Utah State University

Free Vibration of Beams

Euler-Bernoulli Theory

Steven Proctor

MAE 5300 – Vibrations

Dr. Thomas Fronk

December 7, 2016

Problem Description

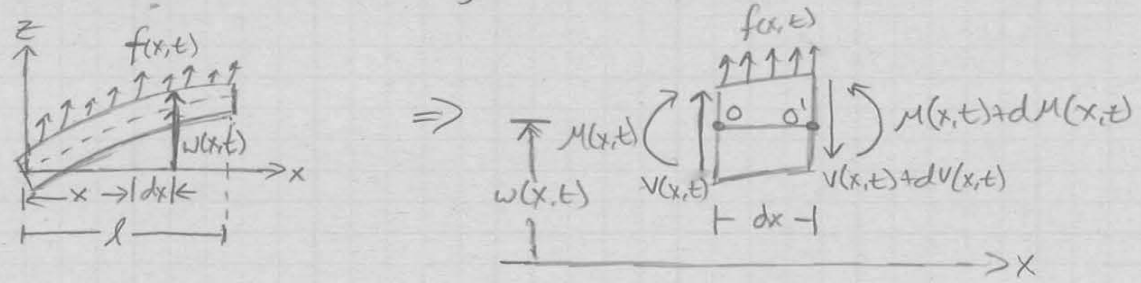
Vibrational forces are a big concern to engineers when designing structures and machines. Many people are familiar with the types of damage vibrational forces can do to a structure; and those who aren't don't need to look further than the collapse of the Tacoma Narrows bridge in 1940 due to vibrational excitation. Competent engineers will design their structure to withstand a wide range of vibrational frequencies to prevent such disasters. However, doing a vibrational analysis on something as complicated as the Tacoma Narrows bridge can be very difficult.

Fortunately, many complicated structures can be simplified as a system of beams linked together under various boundary conditions. A boundary condition simply describes how the beam is secured on each end. For example, the beam might be pinned (i.e. able to rotate), fixed (i.e. unable to move) or free (i.e. able to rotate and move). A method known as the Euler-Bernoulli Beam Theory can be used to obtain a set of equations for these various cases. This theory is derived from Newton's 2nd Law and the relationship between deflection and the moment applied to a beam. In the next section we will re-derive the Euler-Bernoulli Beam Theory to obtain the mode shapes and natural frequencies for different types of boundary conditions.

Derivation of Euler-Bernoulli Beam Theory

Governing Differential Equation

Euler-Bernoulli Beam Theory Derivation



w - displacement (position)

$\frac{d^2w}{dt^2}$ - accereation

$mass = \rho A(x)dx$

$\Sigma F = ma \Rightarrow \Sigma F = \rho A(x)dx \frac{\partial^2 w}{\partial t^2}(x,t)$

$V + f(x,t)dx - (V+dV) = \rho A(x)dx \frac{\partial^2 w}{\partial t^2}(x,t)$ (z-axis)

$(M+dM) - (V+dV)dx + f(x,t)dx\frac{dx}{2} - M = 0$ (y-axis) (Point O)

$dV = \frac{\partial V}{\partial x}dx \quad \& \quad dM = \frac{\partial M}{\partial x}dx$

$V + f(x,t)dx - (V + \frac{\partial V}{\partial x}dx) = \rho A(x)dx \frac{\partial^2 w}{\partial t^2}(x,t)$

$-\frac{\partial V}{\partial x}(x,t) + f(x,t) = \rho A(x)\frac{\partial^2 w}{\partial t^2}(x,t)$

$M + \frac{\partial M}{\partial x}dx - V + \frac{\partial V}{\partial x}dx + f(x,t)dx\frac{dx}{2} - M = 0$

$\frac{\partial M}{\partial x}(x,t) - V(x,t) = 0$ (disregard dx^2 terms)

$V(x,t) = \frac{\partial M}{\partial x}(x,t)$

$-\frac{\partial(\frac{\partial M}{\partial x})}{\partial x}(x,t) + f(x,t) = \rho A(x)\frac{\partial^2 w}{\partial t^2}(x,t) = -\frac{\partial^2 M}{\partial x^2}(x,t) + f(x,t)$

Euler-Bernoulli Theory: $M(x,t) = EI(x)\frac{\partial^2 w}{\partial x^2}(x,t)$

$\frac{\partial^2}{\partial x^2}\left[EI(x)\frac{\partial^2 w}{\partial x^2}(x,t)\right] + \rho A(x)\frac{\partial^2 w}{\partial t^2}(x,t) = f(x,t)$

$$EI\frac{\partial^4 w}{\partial x^4}(x,t) + \rho A\frac{\partial^2 w}{\partial t^2}(x,t) = 0$$ for free vibration $f = 0$

Mode Shapes and Natural Frequencies

$$EI\frac{\partial^4 w}{\partial x^4}(x,t)+\rho A\frac{\partial^2 w}{\partial t^2}(x,t)=0$$

if $c=\sqrt{\frac{EI}{\rho A}}$ then $c^2\frac{\partial^4 w}{\partial x^4}(x,t)+\frac{\partial^2 w}{\partial t^2}(x,t)=0$

assume solution of $w(x,t)=W(x)T(t)$

$$c^2\frac{d^4W(x)T(x)}{dx^4}=-\frac{d^2W(x)T(t)}{dt^2}\Rightarrow\frac{c^2}{W(x)}\cdot\frac{d^4W(x)}{dx^4}=-\frac{1}{T(t)}\cdot\frac{d^2T(t)}{dt^2}$$

$$\frac{c^2}{W(x)}\cdot\frac{d^4W(x)}{dx^4}=-\frac{1}{T(t)}\cdot\frac{d^2T(t)}{dt^2}=\text{constant}\qquad\text{let constant}=\omega^2$$

$$\frac{d^4W(x)}{dx^4}=\frac{\omega^2}{c^2}W(x)\Rightarrow\frac{d^4W(x)}{dx^4}-\beta^4W(x)=0\qquad\beta^4=\frac{\omega^2}{c^2}=\frac{\rho A\omega^2}{EI}$$

$$-\frac{d^2T(t)}{dt^2}=\omega^2T(t)\Rightarrow\frac{d^2T(t)}{dt^2}-\omega^2T(t)=0$$

$\therefore T(t)=A\cos\omega t+B\sin\omega t$ & $W(x)=Ce^{sx}$

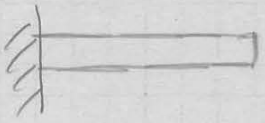
s is a constant where $s^4-\beta^4=0\Rightarrow s_{1,2}=\pm\beta,\ s_{3,4}=\pm i\beta$

$\therefore W(x)=C_1\cos\beta x+C_2\sin\beta x+C_3\cosh\beta x+C_4\sinh\beta x$

$$W(x)=C_1(\cos\beta x+\cosh\beta x)+C_2(\cos\beta x-\cosh\beta x)+C_3(\sin\beta x+\sinh\beta x)+C_4(\sin\beta x-\sinh\beta x)$$

$$\omega^2=EI\beta^4/\rho A\Rightarrow\omega=\sqrt{\beta^4EI\ell^4/\rho A\ell^4}=(\beta\ell)^2\sqrt{EI/\rho A\ell^2}$$

Fixed – Free: Mode Shapes and Frequencies



fixed-free

fixed end	free end
$\frac{\partial w}{\partial x}=0$	$\frac{\partial^2 w}{\partial x^2}=0$
$w=0$	$\frac{\partial}{\partial x}\left(\frac{\partial^2 w}{\partial x^2}\right)=0$

$$\omega=(\beta\ell)^2\sqrt{\frac{EI}{\rho A\ell^4}}$$

$W(x)=C_1\cos\beta x+C_2\sin\beta x+C_3\cosh\beta x+C_4\sinh\beta x$

$W(0)=0=C_1\cos(0)+C_2\sin(0)+C_3\cosh(0)+C_4\sinh(0)$

$0=C_1+C_3\Rightarrow C_1=-C_3$

$$dW/dx = \beta[-C_1\sin\beta x + C_2\cos\beta x + C_3\sinh\beta x + C_4\cosh\beta x]$$

$$dW/dx|_0 = 0 = \beta[-C_1(0) + C_2(1) + C_3(0) + C_4(0)]$$

$$0 = C_2 + C_4 \Rightarrow C_2 = -C_4$$

$$\therefore W(x) = C_1(\cos\beta x - \cosh\beta x) + C_2(\sin\beta x - \sinh\beta x)$$

$$\& \; W(x) = C_2\left[C_1/C_2(\cos\beta x - \cosh\beta x) + \sin\beta x - \sinh\beta x\right]$$

$$dW/dx = C_1\beta(\sin\beta x - \sinh\beta x) + C_2\beta(\cos\beta x - \cosh\beta x)$$

$$d^2W/dx = C_1\beta^2(\cos\beta x + \cosh\beta x) + C_2\beta^2(-\sin\beta x - \sinh\beta x) = 0 \quad @x = L$$

$$d^3W/dx = C_1\beta^3(\sin\beta x - \sinh\beta x) + C_3\beta^3(-\cos\beta x - \cosh\beta x) = 0 \quad @x = L$$

$$A = \begin{vmatrix} (-\cos\beta L - \cosh\beta L) & (-\sin\beta L - \sinh\beta L) \\ (\sin\beta L - \sinh\beta L) & (-\cos\beta L - \cosh\beta L) \end{vmatrix} \begin{Bmatrix} C_1 \\ C_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

$$\alpha_n = C_1/C_2 \Rightarrow (-\cos\beta L - \cosh\beta L)C_1 = (\sin\beta L + \sinh\beta L)C_2$$

$$\alpha_n = C_1/C_2 = -(\sin\beta L + \sinh\beta L)/(\cos\beta L + \cosh\beta L)$$

$$\det|A| = (-\cos\beta L - \cosh\beta L)^2 + (\sin\beta L + \sinh\beta L)(\sin\beta L - \sinh\beta L) = 0$$

$$\cos^2\beta L + 2\cos\beta L\cosh\beta L + \cosh^2\beta L + \sin^2\beta L - \sinh^2\beta L = 0$$

knowing $\cos^2 x + \sin^2 x = 1$ & $\cosh^2 x - \sinh^2 x = 1$

$$2\cos\beta L\cosh\beta L + 2 = 0 \Rightarrow \cos\beta L \cdot \cosh\beta L = -1$$

for a fixed-free beam

frequency equation: $\cos\beta_n L \cdot \cosh\beta_n L = -1$

Mode Shape where $\alpha_n = -\dfrac{(\sin\beta_n L + \sinh\beta_n L)}{(\cos\beta_n L + \cosh\beta_n L)}$

$$W_n(x) = C_n\left[\alpha_n(\cos\beta_n x - \cosh\beta x) + \sin\beta_n x - \sinh\beta_n x\right]$$

using a solver for frequency equation:

$\beta_1 L = 1.875$ $\quad$ $\beta_3 L \doteq 7.855$

$\beta_2 L = 4.694$ $\quad$ $\beta_4 L = 10.996$

Solutions to Various Beam Cases

Case 1: Pinned – Pinned Beam

Case I: Pinned-Pinned		
Young's Modulus	207	Gpa
Density	7800	kg/m^3
Length	0.5	m
Thickness	0.002	m
Width	0.02	m

Case I: Pinned-Pinned		
Mode	**Beta**	**Freq**
0	6.28319	117.419
1	12.5664	469.674
2	18.8496	1056.77
3	25.1327	1878.7

Case I: Pinned-Pinned				
x/L	**Mode**			
	1	**2**	**3**	**4**
0	0	0	0	0
0.05	0.1564	0.309	0.454	0.5878
0.1	0.309	0.5878	0.809	0.9511
0.15	0.454	0.809	0.9877	0.9511
0.2	0.5878	0.9511	0.9511	0.5878
0.25	0.7071	1	0.7071	0
0.3	0.809	0.9511	0.309	-0.5878
0.35	0.891	0.809	-0.1564	-0.9511
0.4	0.9511	0.5878	-0.5878	-0.9511
0.45	0.9877	0.309	-0.891	-0.5878
0.5	1	0	-1	0
0.55	0.9877	-0.309	-0.891	0.5878
0.6	0.9511	-0.5878	-0.5878	0.9511
0.65	0.891	-0.809	-0.1564	0.9511
0.7	0.809	-0.9511	0.309	0.5878
0.75	0.7071	-1	0.7071	0
0.8	0.5878	-0.9511	0.9511	-0.5878
0.85	0.454	-0.809	0.9877	-0.9511
0.9	0.309	-0.5878	0.809	-0.9511
0.95	0.1564	-0.309	0.454	-0.5878
1	0	0	0	0

Case 2: Free – Free Beam

Case II: Free-Free		
Young's Modulus	120	Gpa
Density	2100	kg/m^3
Length	0.5	m
Thickness	0.002	m
Width	0.02	m

Case II: Free-Free		
Mode	**Beta**	**Freq**
0	9.46008	390.58
1	15.7064	1076.65
2	21.9912	2110.66
3	28.2743	3489.03

Case II: Free-Free				
x/L	**Mode**			
	1	**2**	**3**	**4**
0	-2.036	-1.998	-2	-2
0.05	-1.563	-1.215	-0.9073	-0.6041
0.1	-1.093	-0.4545	0.1039	0.588
0.15	-0.6351	0.2348	0.8831	1.255
0.2	-0.1989	0.7939	1.286	1.201
0.25	0.2019	1.169	1.242	0.512
0.3	0.5537	1.323	0.7939	-0.4514
0.35	0.8432	1.248	0.08887	-1.213
0.4	1.059	0.9653	-0.6557	-1.4
0.45	1.192	0.5251	-1.215	-0.9198
0.5	1.237	0	-1.422	0
0.55	1.192	-0.5251	-1.215	0.9198
0.6	1.059	-0.9653	-0.6557	1.4
0.65	0.8432	-1.248	0.08887	1.213
0.7	0.5537	-1.323	0.7939	0.4514
0.75	0.2019	-1.169	1.242	-0.512
0.8	-0.1989	-0.7939	1.286	-1.201
0.85	-0.6351	-0.2348	0.8831	-1.255
0.9	-1.093	0.4545	0.1039	-0.588
0.95	-1.563	1.215	-0.9073	0.6041
1	-2.036	1.998	-2	2

Case 3: Fixed – Fixed Beam

Case III: Fixed-Fixed		
Young's Modulus	120	Gpa
Density	2100	kg/m^3
Length	0.5	m
Thickness	0.002	m
Width	0.02	m

Case III: Fixed-Fixed		
Mode	**Beta**	**Freq**
0	9.46008	390.58
1	15.7064	1076.65
2	21.9912	2110.66
3	28.2743	3489.03

Case III: Fixed-Fixed				
x/L	**Mode**			
	1	**2**	**3**	**4**
0	0	0	0	0
0.05	-0.05252	-0.1339	-0.2469	-0.3822
0.1	-0.1925	-0.4554	-0.7701	-1.074
0.15	-0.3936	-0.8479	-1.268	-1.495
0.2	-0.6304	-1.206	-1.508	-1.319
0.25	-0.8785	-1.444	-1.371	-0.5703
0.3	-1.116	-1.504	-0.8687	0.4227
0.35	-1.322	-1.364	-0.1331	1.199
0.4	-1.481	-1.034	0.6284	1.394
0.45	-1.582	-0.5568	1.196	0.9172
0.5	-1.616	0	1.406	0
0.55	-1.582	0.5568	1.196	-0.9172
0.6	-1.481	1.034	0.6284	-1.394
0.65	-1.322	1.364	-0.1331	-1.199
0.7	-1.116	1.504	-0.8687	-0.4227
0.75	-0.8785	1.444	-1.371	0.5703
0.8	-0.6304	1.206	-1.508	1.319
0.85	-0.3936	0.8479	-1.268	1.495
0.9	-0.1925	0.4554	-0.7701	1.074
0.95	-0.05252	0.1339	-0.2469	0.3822
1	0	0	0	0

Case 4: Fixed – Free Beam

Case IV: Fixed-Free		
Young's Modulus	120	Gpa
Density	2100	kg/m^3
Length	0.5	m
Thickness	0.002	m
Width	0.02	m

Case IV: Fixed-Free		
Mode	**Beta**	**Freq**
0	3.75021	61.3806
1	9.38818	384.666
2	15.7095	1077.07
3	21.9911	2110.64

Case IV: Fixed-Free				
x/L	**Mode**			
	1	**2**	**3**	**4**
0	0	0	0	0
0.05	0.0117	0.04978	0.1342	0.2469
0.1	0.0457	0.1819	0.4565	0.77
0.15	0.1004	0.3707	0.8503	1.267
0.2	0.174	0.5912	1.21	1.508
0.25	0.265	0.8194	1.45	1.37
0.3	0.3718	1.033	1.514	0.8677
0.35	0.4928	1.213	1.377	0.1315
0.4	0.6263	1.342	1.053	-0.6311
0.45	0.7709	1.408	0.5839	-1.201
0.5	0.925	1.401	0.03941	-1.414
0.55	1.087	1.318	-0.499	-1.211
0.6	1.256	1.158	-0.9483	-0.653
0.65	1.431	0.9234	-1.237	0.09042
0.7	1.61	0.6226	-1.316	0.7948
0.75	1.792	0.2651	-1.164	1.243
0.8	1.977	-0.1375	-0.7904	1.286
0.85	2.163	-0.5727	-0.2318	0.8833
0.9	2.35	-1.029	0.4574	0.1041
0.95	2.537	-1.495	1.218	-0.9072
1	2.724	-1.964	2.002	-2

Appendix

C++ Code Used to Generate Solutions

```
//
//  main.cpp
//  Beam Theory
//
//  Created by Steven Proctor on 12/4/16.
//  Copyright © 2016 Steven Proctor. All rights reserved.
//

#include <iostream>
#include "PinnedPinned.h"
#include "FreeFree.h"
#include "FixedFixed.h"
#include "FixedFree.h"

using namespace std;

int main() {

    double E, row, L, h, b;
    char choice = 'y';

    cout << "Welcome to the Euler-Bernoulli Beam Solver!\n";
    cout << "*****************************************\n\n";
    cout << "First: Enter the properites of the beam\n";
    cout << "Second: Select a beam configuration\n\n\n";

    while(choice != 'n') {

        cout << "Enter Young's Modulus: (GPa)\n";
        cin >> E;
        cout << endl;

        cout << "Enter density: (kg/m^3)\n";
        cin >> row;
        cout << endl;

        cout << "Enter length: (m)\n";
        cin >> L;
        cout << endl;

        cout << "Enter thickness: (m)\n";
        cin >> h;
        cout << endl;

        cout << "Enter width: (m)\n";
        cin >> b;
        cout << endl;

        cout << "Select a type of Beam:\n";
        cout << "a: Pinned-Pinned\n";
        cout << "b: Free-Free\n";
```

```
        cout << "c: Fixed-Fixed\n";
        cout << "d: Fixed-Free\n";
        cin >> choice;

        while (choice != 'a' && choice != 'b' && choice != 'c' && choice != 'd') {
            cout << "Please enter 'a', 'b', 'c', or 'd'\n";
            cin >> choice;
        }

        cout << endl << endl;

        if (choice == 'a') {
            PinnedPinned PP(E, row, L, h, b);
        }
        else if (choice == 'b') {
            FreeFree FF(E, row, L, h, b);
        }
        else if (choice == 'c') {
            FixedFixed XX(E, row, L, h, b);
        }
        else {
            FixedFree XF(E, row, L, h, b);
        }


        cout << "Would you like to analyze another beam? 'y' or 'n'?\n";
        cin >> choice;
        cout << endl << endl << endl;
    }


    return 0;
}
```

```
//
//  PinnedPinned.h
//  Beam Theory
//
//  Created by Steven Proctor on 12/4/16.
//  Copyright © 2016 Steven Proctor. All rights reserved.
//

#ifndef PinnedPinned_h
#define PinnedPinned_h

#include <iostream>
#include <math.h>
#include <string>
#include <sstream>
#include <iomanip>
#include <fstream>

const double pi = 3.14159265359;

using namespace std;

class PinnedPinned {
private:
    string fileName = "PinnedPinned.csv";
    double width;
    double length;
    double thickness;
    double modulus;
    double density;
    double area;
    double volume;
    double mass;
    double inertia;

    double betaL[4] = { 1 * pi, 2 * pi, 3 * pi, 4 * pi };
    double beta[4];
    double naturalFreq[4];
    double modeShape[4][21];

    void findNatFreq() {
        for (int i = 0; i < 4; i++) {
            naturalFreq[i] = pow(betaL[i], 2.0) * sqrt((modulus * inertia) / (density * area * pow(length, 4.0)));
        }
    }

    void findBeta() {
        for (int i = 0; i < 4; i++) {
            beta[i] = betaL[i] / length;
        }
    }

    void findMode() {
        double x;
        for (int i = 0; i < 4; i++) {
            double increment = 0;
```

```
        for (int j = 0; j < 21; j++) {
            x = increment * length;
            modeShape[i][j] = sin(beta[i] * x);
            if (modeShape[i][j] < .00001 && modeShape[i][j] > -.00001) { modeShape[i][j] = 0;}
            increment += .05;
        }
    }
}

string modeTable() {
    stringstream ss;
    double incr = 0;
    ss << "                                Mode\n";
    ss << " x/L    ------------------------------------------------------------\n";
    ss << "               1             2             3             4\n";
    ss << "-------------------------------------------------------------------\n";
    for (int i = 0; i < 21; i++) {
        ss << setw(4) << incr << "     ";
        for (int j = 0; j < 4; j++) {
            ss << setw(10) << setprecision(4) << modeShape[j][i] << "     ";
        }
        ss << endl;
        incr += .05;
    }
    return ss.str();
}

string freqTable() {
    stringstream ss;
    ss << "Mode      Beta        Freq\n";
    ss << "------------------------------\n";
    for (int i = 0; i < 4; i++) {
        ss << i << ":       " << setprecision(6) << beta[i] << "     " << setprecision(6) << naturalFreq[i] << endl;
    }
    return ss.str();
}

string title() {
    stringstream ss;
    ss << "Pinned - Pinned Beam Results\n";
    ss << "***************************\n";
    return ss.str();
}

void toCSV() {
    ofstream fout;
    fout.open(fileName);

    fout << title() << endl;

    fout << "Mode,Beta,Freq\n";
    for (int i = 0; i < 4; i++) {
        fout << i << "," << setprecision(6) << beta[i] << "," << setprecision(6) << naturalFreq[i] << endl;
    }

    fout << endl << endl;
```

```
        double incr = 0;
        fout << ",Mode\n";
        fout << "x/L,1,2,3,4\n";
        for (int i = 0; i < 21; i++) {
            fout << setw(4) << incr << ",";
            for (int j = 0; j < 4; j++) {
                fout << setw(10) << setprecision(4) << modeShape[j][i] << ",";
            }
            fout << endl;
            incr += .05;
        }

        fout.close();
    }

public:

    PinnedPinned(double E, double row, double L, double h, double b) {
        modulus = E * 1000000000;
        density = row;
        length = L;
        width = b;
        thickness = h;

        area = width * thickness;
        volume = area * length;
        mass = density * volume;
        inertia = width * pow(thickness, 3) / 12;

        findBeta();
        findNatFreq();
        findMode();

        cout << title() << endl;
        cout << freqTable() << endl;
        cout << modeTable() << endl;
        toCSV();
    }

};

#endif /* PinnedPinned_h */
```

```
//
//  FreeFree.h
//  Beam Theory
//
//  Created by Steven Proctor on 12/4/16.
//  Copyright © 2016 Steven Proctor. All rights reserved.
//

#ifndef FreeFree_h
#define FreeFree_h

#include <iostream>
#include <math.h>
#include <string>
#include <sstream>
#include <iomanip>
#include <fstream>

using namespace std;

class FreeFree {
private:
   string fileName = "FreeFree.csv";
   double width;
   double length;
   double thickness;
   double modulus;
   double density;
   double area;
   double volume;
   double mass;
   double inertia;

   double betaL[4] = { 1.505618812 * pi, 2.49975279 * pi, 3.500010731 * pi, 4.499999382 * pi };
   double beta[4];
   double alpha_n[4];
   double naturalFreq[4];
   double modeShape[4][21];

   void findNatFreq() {
      for (int i = 0; i < 4; i++) {
         naturalFreq[i] = pow(betaL[i], 2.0) * sqrt((modulus * inertia) / (density * area * pow(length, 4.0)));
      }
   }

   void findBeta() {
      for (int i = 0; i < 4; i++) {
         beta[i] = betaL[i] / length;
      }
   }

   void findMode() {
      double x;
      for (int i = 0; i < 4; i++) {
         double increment = 0;
         for (int j = 0; j < 21; j++) {
```

```
            x = increment * length;
            modeShape[i][j] = alpha_n[i] * (cos(beta[i] * x) + cosh(beta[i] * x)) + sin(beta[i] * x) + sinh(beta[i] * x);
            if (modeShape[i][j] < .00001 && modeShape[i][j] > -.00001) { modeShape[i][j] = 0;}
            increment += .05;
        }
    }
}

string modeTable() {
    stringstream ss;
    double incr = 0;
    ss << "                                Mode\n";
    ss << " x/L    ------------------------------------------------------------\n";
    ss << "               1             2             3             4\n";
    ss << "-------------------------------------------------------------------\n";
    for (int i = 0; i < 21; i++) {
        ss << setw(4) << incr << "     ";
        for (int j = 0; j < 4; j++) {
            ss << setw(10) << setprecision(4) << modeShape[j][i] << "     ";
        }
        ss << endl;
        incr += .05;
    }
    return ss.str();
}

void findAlpha() {
    for (int i = 0; i < 4; i++) {
        alpha_n[i] = ((sin(betaL[i]) - sinh(betaL[i])) / (cosh(betaL[i]) - cos(betaL[i])));
    }
}

string freqTable() {
    stringstream ss;
    ss << "Mode      Beta        Freq\n";
    ss << "-----------------------------\n";
    for (int i = 0; i < 4; i++) {
        ss << i << ":       " << setprecision(6) << beta[i] << "     " << setprecision(6) << naturalFreq[i] << endl;
    }
    return ss.str();
}

string title() {
    stringstream ss;
    ss << "Free - Free Beam Results\n";
    ss << "**************************\n";
    return ss.str();
}

void toCSV() {
    ofstream fout;
    fout.open(fileName);

    fout << title() << endl;

    fout << "Mode,Beta,Freq\n";
```

```
        for (int i = 0; i < 4; i++) {
            fout << i << "," << setprecision(6) << beta[i] << "," << setprecision(6) << naturalFreq[i] << endl;
        }

        fout << endl << endl;

        double incr = 0;
        fout << ",Mode\n";
        fout << "x/L,1,2,3,4\n";
        for (int i = 0; i < 21; i++) {
            fout << setw(4) << incr << ",";
            for (int j = 0; j < 4; j++) {
                fout << setw(10) << setprecision(4) << modeShape[j][i] << ",";
            }
            fout << endl;
            incr += .05;
        }

        fout.close();
    }

public:

    FreeFree(double E, double row, double L, double h, double b) {
        modulus = E * 1000000000;
        density = row;
        length = L;
        width = b;
        thickness = h;

        area = width * thickness;
        volume = area * length;
        mass = density * volume;
        inertia = width * pow(thickness, 3) / 12;

        findBeta();
        findNatFreq();
        findAlpha();
        findMode();

        cout << title() << endl;
        cout << freqTable() << endl;
        cout << modeTable() << endl;
        toCSV();
    }

};

#endif /* FreeFree_h */
```

```
//
//  FixedFixed.h
//  Beam Theory
//
//  Created by Steven Proctor on 12/4/16.
//  Copyright © 2016 Steven Proctor. All rights reserved.
//

#ifndef FixedFixed_h
#define FixedFixed_h

#include <iostream>
#include <math.h>
#include <string>
#include <sstream>
#include <iomanip>
#include <fstream>

using namespace std;

class FixedFixed {
private:
   string fileName = "FixedFixed.csv";
   double width;
   double length;
   double thickness;
   double modulus;
   double density;
   double area;
   double volume;
   double mass;
   double inertia;

   double betaL[4] = { 1.505618812 * pi, 2.49975279 * pi, 3.500010731 * pi, 4.499999382 * pi };
   double beta[4];
   double alpha_n[4];
   double naturalFreq[4];
   double modeShape[4][21];

   void findNatFreq() {
      for (int i = 0; i < 4; i++) {
         naturalFreq[i] = pow(betaL[i], 2.0) * sqrt((modulus * inertia) / (density * area * pow(length, 4.0)));
      }
   }

   void findBeta() {
      for (int i = 0; i < 4; i++) {
         beta[i] = betaL[i] / length;
      }
   }

   void findMode() {
      double x;
      for (int i = 0; i < 4; i++) {
         double increment = 0;
         for (int j = 0; j < 21; j++) {
```

```
            x = increment * length;
            modeShape[i][j] = alpha_n[i] * (cosh(beta[i] * x) - cos(beta[i] * x)) + sinh(beta[i] * x) - sin(beta[i] * x);
            if (modeShape[i][j] < .00001 && modeShape[i][j] > -.00001) { modeShape[i][j] = 0;}
            increment += .05;
        }
    }
}

string modeTable() {
    stringstream ss;
    double incr = 0;
    ss << "                              Mode\n";
    ss << " x/L    ------------------------------------------------------------\n";
    ss << "               1             2             3             4\n";
    ss << "-------------------------------------------------------------------\n";
    for (int i = 0; i < 21; i++) {
        ss << setw(4) << incr << "     ";
        for (int j = 0; j < 4; j++) {
            ss << setw(10) << setprecision(4) << modeShape[j][i] << "     ";
        }
        ss << endl;
        incr += .05;
    }
    return ss.str();
}

void findAlpha() {
    for (int i = 0; i < 4; i++) {
        alpha_n[i] = (sinh(betaL[i]) - sin(betaL[i])) / (cos(betaL[i]) - cosh(betaL[i]));
    }
}

string freqTable() {
    stringstream ss;
    ss << "Mode      Beta        Freq\n";
    ss << "-----------------------------\n";
    for (int i = 0; i < 4; i++) {
        ss << i << ":       " << setprecision(6) << beta[i] << "     " << setprecision(6) << naturalFreq[i] << endl;
    }
    return ss.str();
}

string title() {
    stringstream ss;
    ss << "Fixed - Fixed Beam Results\n";
    ss << "**************************\n";
    return ss.str();
}

void toCSV() {
    ofstream fout;
    fout.open(fileName);

    fout << title() << endl;

    fout << "Mode,Beta,Freq\n";
```

```
        for (int i = 0; i < 4; i++) {
            fout << i << "," << setprecision(6) << beta[i] << "," << setprecision(6) << naturalFreq[i] << endl;
        }

        fout << endl << endl;

        double incr = 0;
        fout << ",Mode\n";
        fout << "x/L,1,2,3,4\n";
        for (int i = 0; i < 21; i++) {
            fout << setw(4) << incr << ",";
            for (int j = 0; j < 4; j++) {
                fout << setw(10) << setprecision(4) << modeShape[j][i] << ",";
            }
            fout << endl;
            incr += .05;
        }

        fout.close();
    }

public:

    FixedFixed(double E, double row, double L, double h, double b) {
        modulus = E * 1000000000;
        density = row;
        length = L;
        width = b;
        thickness = h;

        area = width * thickness;
        volume = area * length;
        mass = density * volume;
        inertia = width * pow(thickness, 3) / 12;

        findBeta();
        findNatFreq();
        findAlpha();
        findMode();

        cout << title() << endl;
        cout << freqTable() << endl;
        cout << modeTable() << endl;
        toCSV();
    }

};

#endif /* FixedFixed_h */
```

```
//
//  FixedFree.h
//  Beam Theory
//
//  Created by Steven Proctor on 12/4/16.
//  Copyright © 2016 Steven Proctor. All rights reserved.
//

#ifndef FixedFree_h
#define FixedFree_h

#include <iostream>
#include <math.h>
#include <string>
#include <sstream>
#include <iomanip>
#include <fstream>

using namespace std;

class FixedFree {
private:
   string fileName = "FixedFree.csv";
   double width;
   double length;
   double thickness;
   double modulus;
   double density;
   double area;
   double volume;
   double mass;
   double inertia;

   double betaL[4] = { .5968641408 * pi, 1.494175572 * pi, 2.500246807 * pi, 3.499989404 * pi };
   double beta[4];
   double alpha_n[4];
   double naturalFreq[4];
   double modeShape[4][21];

   void findNatFreq() {
      for (int i = 0; i < 4; i++) {
         naturalFreq[i] = pow(betaL[i], 2.0) * sqrt((modulus * inertia) / (density * area * pow(length, 4.0)));
      }
   }

   void findBeta() {
      for (int i = 0; i < 4; i++) {
         beta[i] = betaL[i] / length;
      }
   }

   void findMode() {
      double x;
      for (int i = 0; i < 4; i++) {
         double increment = 0;
         for (int j = 0; j < 21; j++) {
```

```
            x = increment * length;
            modeShape[i][j] = - alpha_n[i] * (cos(beta[i] * x) - cosh(beta[i] * x)) + sin(beta[i] * x) - sinh(beta[i] * x);
            if (modeShape[i][j] < .00001 && modeShape[i][j] > -.00001) { modeShape[i][j] = 0;}
            increment += .05;
        }
    }
}

string modeTable() {
    stringstream ss;
    double incr = 0;
    ss << "                                Mode\n";
    ss << " x/L    ------------------------------------------------------------\n";
    ss << "               1             2             3             4\n";
    ss << "-------------------------------------------------------------------\n";
    for (int i = 0; i < 21; i++) {
        ss << setw(4) << incr << "     ";
        for (int j = 0; j < 4; j++) {
            ss << setw(10) << setprecision(4) << modeShape[j][i] << "     ";
        }
        ss << endl;
        incr += .05;
    }
    return ss.str();
}

void findAlpha() {
    for (int i = 0; i < 4; i++) {
        alpha_n[i] = (sin(betaL[i]) + sinh(betaL[i])) / (cos(betaL[i]) + cosh(betaL[i]));
    }
}

string freqTable() {
    stringstream ss;
    ss << "Mode      Beta        Freq\n";
    ss << "------------------------------\n";
    for (int i = 0; i < 4; i++) {
        ss << i << ":       " << setprecision(6) << beta[i] << "      " << setprecision(6) << naturalFreq[i] << endl;
    }
    return ss.str();
}

string title() {
    stringstream ss;
    ss << "Fixed - Free Beam Results\n";
    ss << "**************************\n";
    return ss.str();
}

void toCSV() {
    ofstream fout;
    fout.open(fileName);

    fout << title() << endl;

    fout << "Mode,Beta,Freq\n";
```

```
        for (int i = 0; i < 4; i++) {
            fout << i << "," << setprecision(6) << beta[i] << "," << setprecision(6) << naturalFreq[i] << endl;
        }

        fout << endl << endl;

        double incr = 0;
        fout << ",Mode\n";
        fout << "x/L,1,2,3,4\n";
        for (int i = 0; i < 21; i++) {
            fout << setw(4) << incr << ",";
            for (int j = 0; j < 4; j++) {
                fout << setw(10) << setprecision(4) << modeShape[j][i] << ",";
            }
            fout << endl;
            incr += .05;
        }

        fout.close();
    }

public:

    FixedFree(double E, double row, double L, double h, double b) {
        modulus = E * 1000000000;
        density = row;
        length = L;
        width = b;
        thickness = h;

        area = width * thickness;
        volume = area * length;
        mass = density * volume;
        inertia = width * pow(thickness, 3) / 12;

        findBeta();
        findNatFreq();
        findAlpha();
        findMode();

        cout << title() << endl;
        cout << freqTable() << endl;
        cout << modeTable() << endl;
        toCSV();
    }

};

#endif /* FixedFree_h */
```