**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**“Московский политехнический университет”**

**КАФЕДРА ИНФОКОГНИТИВНЫХ ТЕХНОЛОГИЙ**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине

**ИНЖЕНЕРНОЕ ПРОЕКТИРОВАНИЕ**

Задание № 1

**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ АНАЛИЗА НАПРЯЖЕННО-ДЕФОРМИРОВАННОГО СОСТОЯНИЯ БАЛОК**

Студент: Аверьянов А.А.

Группа: 171-333

Преподаватель: к.т.н., доцент Лянг В.Ф.

Москва - 2019

**Задание**

1. Разработать программное обеспечение на языке программирования C# по анализу напряженно-деформированного состояние двух статически определимых балок с разными поперечными сечениями.
2. Программное обеспечение должно включать одно или несколько диалоговых окон или окно с вкладками для размещения в них расчетной схемы, поперечных сечений балок и результатов численного моделирования напряженно-деформированного состояния. Окна снабдить элементами управления.

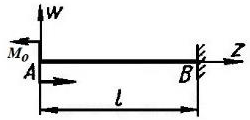
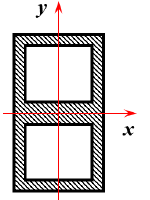
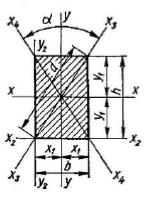


Рис. 1. Виды сечений и расчетная схема

**Содержание**

[Введение 4](#_Toc534642168)

[1. Интерфейс пользователя 5](#_Toc534642169)

[1.1. Вкладка «Эпюры» 5](#_Toc534642170)

[1.2. Вкладки «w(z) и θ(z)» 5](#_Toc534642171)

[1.3. Вкладка «f(Mo) и σ max(Mo)» 7](#_Toc534642172)

[1.4. Вкладка «Масса и оптимальный профиль» 9](#_Toc534642173)

[2. Расчеты и построение графиков и эпюр 10](#_Toc534642174)

[2.1. Эпюры 10](#_Toc534642175)

[2.1.1. Эпюр Q 10](#_Toc534642176)

[2.1.2. Эпюр M 11](#_Toc534642177)

[2.2. Графики w(z) и θ(z) 11](#_Toc534642178)

[2.2.1. Сплошное сечение 12](#_Toc534642179)

[2.2.2. Полое сечение 13](#_Toc534642180)

[2.2.3. Момент инерции Jx 14](#_Toc534642181)

[2.3. Графики f(Mo) и σ max(Mo) 14](#_Toc534642182)

[2.3.1. График f(Mo) 14](#_Toc534642183)

[2.3.2. График σ max(Mo) 15](#_Toc534642184)

[2.4. Расчет массы и поиск оптимального профиля по массе 16](#_Toc534642185)

[3. Дополнительный функционал программы 18](#_Toc534642186)

[Заключение 19](#_Toc534642187)

[Список используемой литературы 20](#_Toc534642188)

[Приложение 1 21](#_Toc534642189)

# Введение

**Цель**

Целью проекта является разработка программного обеспечения на языке программирования C# в среде Microsoft Visual Studio, способного проанализировать напряженно-деформированные состояния балок.

**Поставленные задачи**

1. Разработка интерфейса программного продукта, состоящего из расчетной схемы, чертежей сечения балок и их настраиваемыми размерами.
2. Программное построение эпюр и графиков на основе вводных данных.

# 1. Интерфейс пользователя

Основное окно состоит из вкладок. В каждой вкладке пользователю предложены рисунки, схема и настройки размеров для построения эпюр и графиков

## 1.1. Вкладка «Эпюры»

Здесь расположена расчетная схема балки, параметры её длины, а также неравномерно-распределенной нагрузки с её величиной и длиной. Так же можно построить эпюры поперечной силы Q и изгибающего момента M.

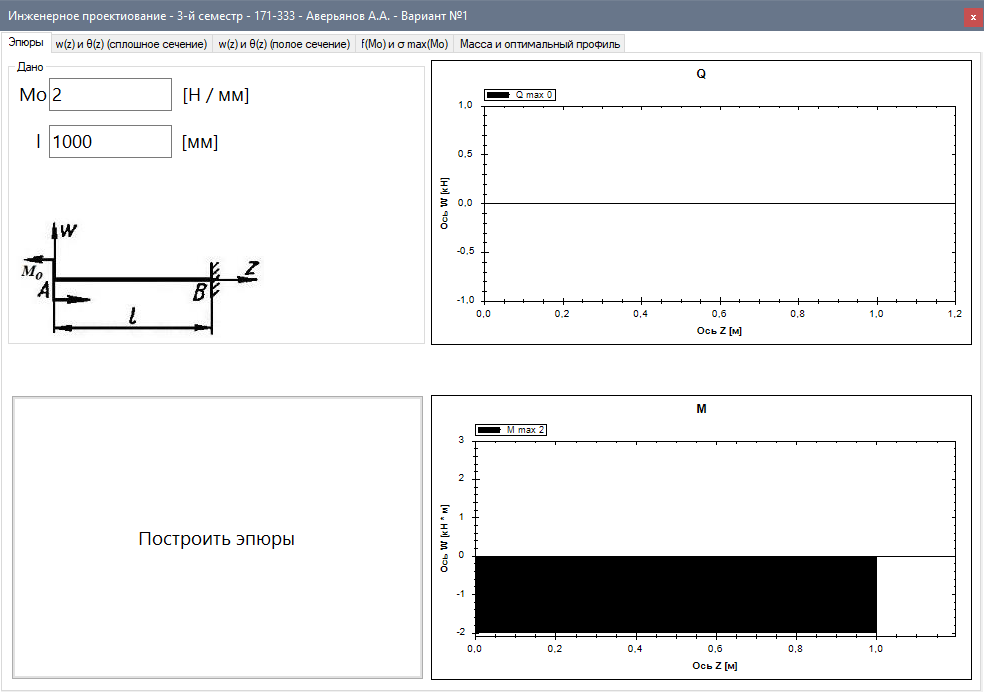


Рис. 1.1. Вкладка «Эпюры»

## 

## 1.2. Вкладки «w(z) и θ(z)»

На этих двух вкладках расположены чертежи поперечных сечений балки с расчетной схемой, а также поля настройки размеров этих сечений, графики соответствующих функций и поля вывода максимального значения каждого графика.

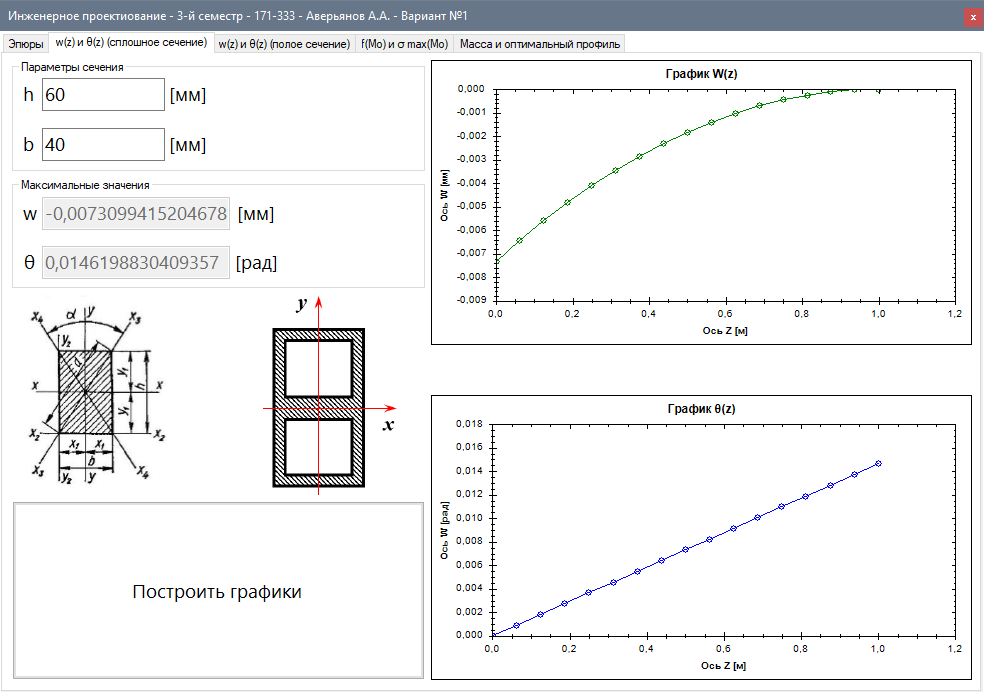


Рис. 1.2. Вкладка «w(z) и θ(z) (сплошное сечение)»

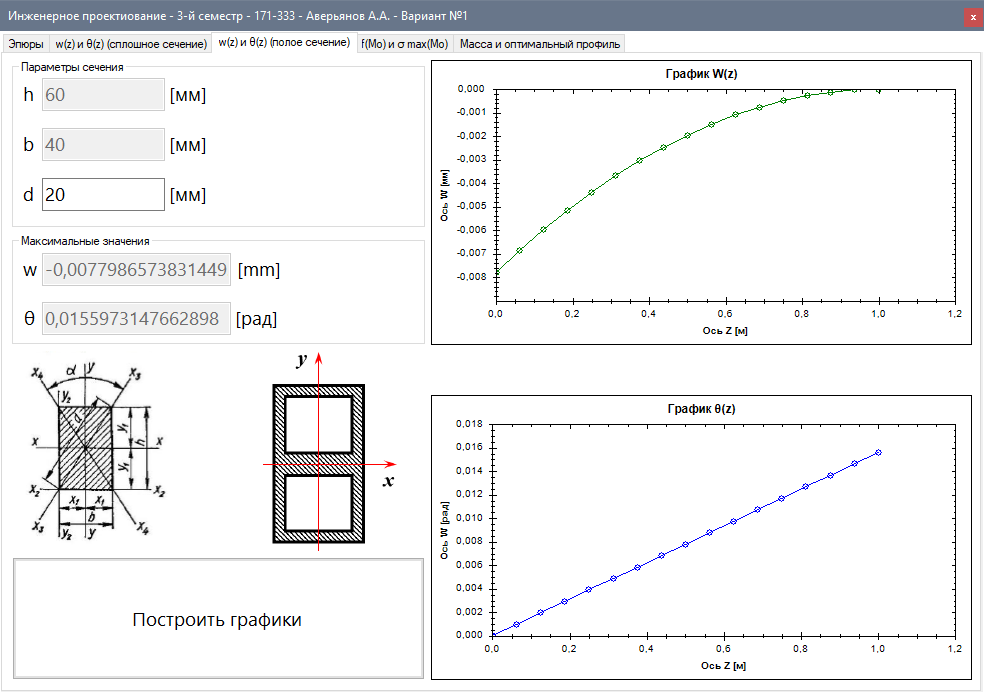


Рис. 1.3. Вкладка «w(z) и θ(z) (полое сечение)»

## 1.3. Вкладка «f(q) и σ max(q)»

На вкладке расположены графики зависимость максимального смещения f и максимального напряжения σ от нагрузки q с возможность настраивать диапазон q.

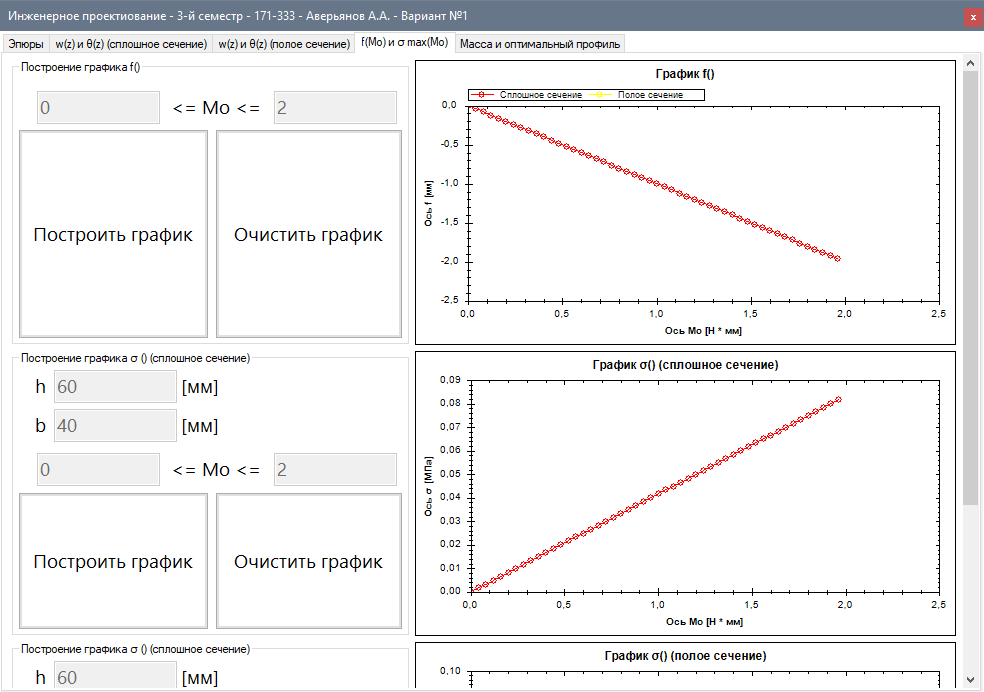


Рис. 1.4. Вкладка «f(q) и σ max(q)(сплошное сечение)»

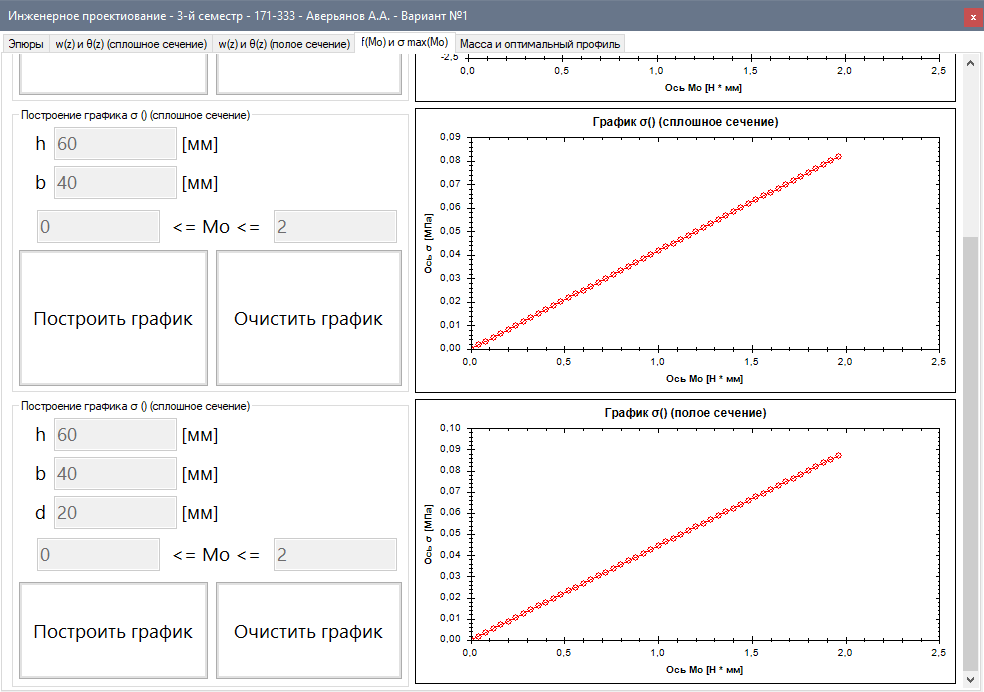


Рис. 1.5. Вкладка «σ max(q) (сплошное сечение) и «σ max(q) (полое сечение)»

## 1.4. Вкладка «Масса и оптимальный профиль»

На вкладке расположены настройки профилей и график зависимости B(H). Также здесь можно найти оптимальный по массе профиль.

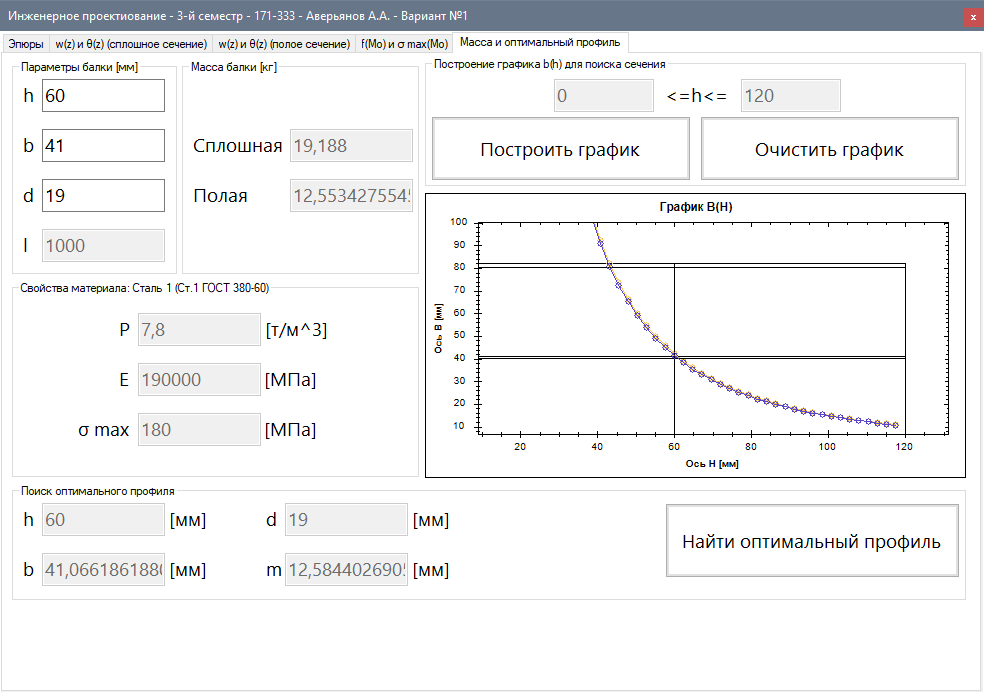


Рис. 1.6. Вкладка «Масса и оптимальный профиль»

# 2. Расчеты и построение графиков и эпюр

Для создания графиков и эпюр была подключена графическая библиотека ZedGraph, которая позволяет делать графики очень легко и просто. Каждый график или эпюр строится при помощи цикла, в теле которого функция, рассчитывающая точки для графика. Эти точки подставляются в соответствующий метод графика.

Пример (прямая линия):

PointPairList list = new PointPairList();

for (int i = 0; i <= 2; i ++) list.Add(i, 2 \* i + 7);

## 

## 2.1. Эпюры

### 2.1.1. Эпюр Q

По заданию на балку не было приложено внешних сил за исключением момента, поэтому Q равен нулю.

В результате получается соответствующий эпюр.

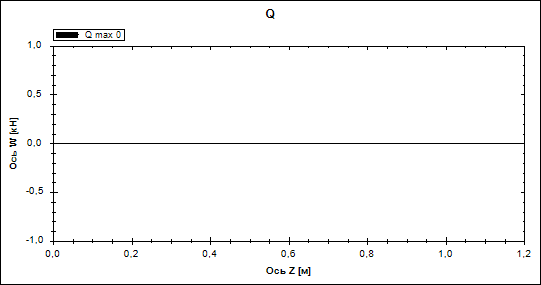


Рис. 2.1. Эпюр Q

### 2.1.2. Эпюр M

Для построения данного эпюра использовались формулы, отображенные на рис. 2.2.

M = - Mo \* l

Рис. 2.2. Формулы для расчета M

В результате получается соответствующий эпюр.

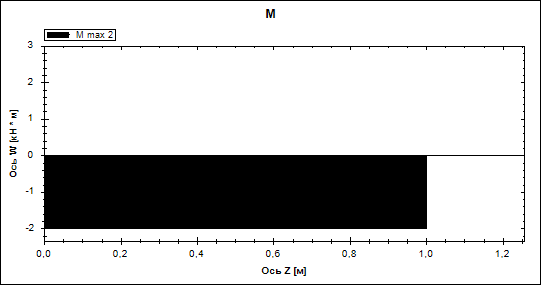


Рис. 2.3. Эпюр M

## 2.2. Графики w(z) и θ(z)

Данные графики строились при помощи формул, отображенных на рис. 2.4 и рис. 2.5.

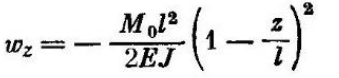


Рис. 2.4. Формулы для расчета w(z)

θ(z) = (Mo \* z) / (E \* Jx)

Рис. 2.5. Формулы для расчета θ(z)

### 2.2.1. Сплошное сечение

В результате получаются соответствующие графики.

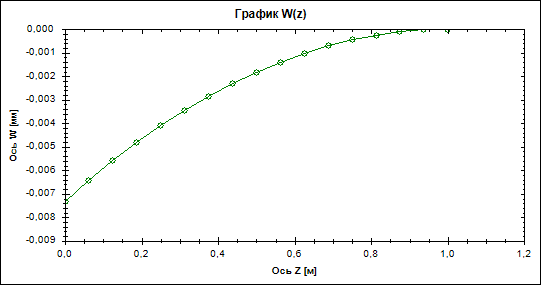


Рис. 2.6. График w(z) для сплошного сечения

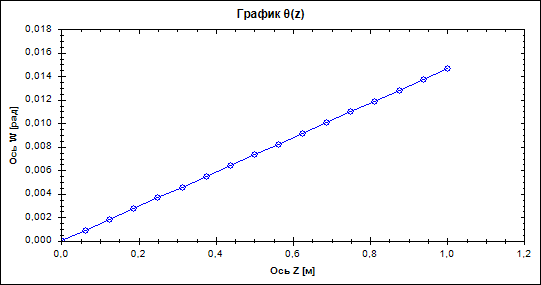


Рис. 2.7. График θ(z) для сплошного сечения

### 2.2.2. Полое сечение

В результате получаются соответствующие графики.

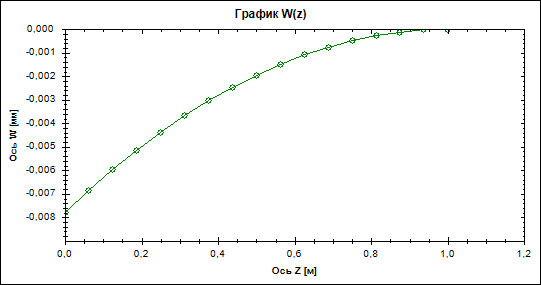


Рис. 2.8. Графики w(z) и θ(z) для полого сечения

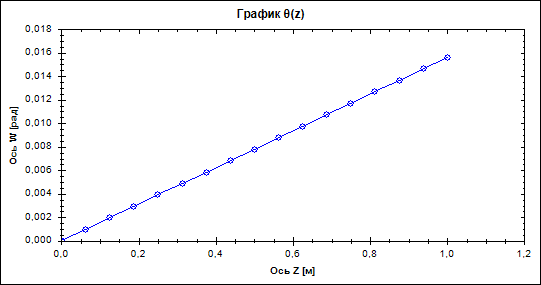


Рис. 2.9. Графики w(z) и θ(z) для полого сечения

### 2.2.3. Момент инерции Jx

Для предыдущих формул, а также для последующих, была необходима переменная, которая определяла момент инерции относительно оси x.

Для данных сечений этот момент рассчитывался таким образом (далее выдержка из программы):

public double Jx(double b, double h, double d = 0) =>

b \* Math.Pow(h, 3) / 12 - 2 \* 0.141 \* Math.Pow(d, 4);

## 2.3. Графики f(Mo) и σ max(q)

Данные графики нужны для просмотра прямолинейной зависимости параметров максимального прогиба и напряжений при изменении нагрузки и увидеть разницу при полом и сплошном сечениях.

### 2.3.1. График f(q)

Для построения данного графика использовалась формула, ранее отображенная на рис. 2.4., однако вместо z изменялся параметр Mo, а в качестве z была выбрана точка, где прогиб максимальный. В результате получился соответствующих график.

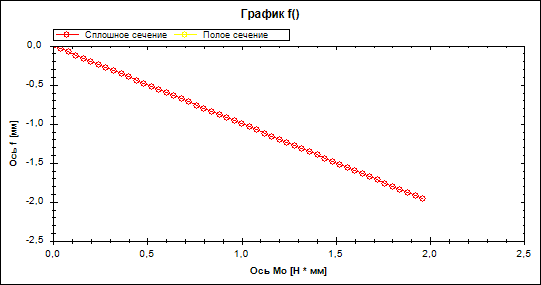


Рис. 2.10. График f(Mo) для сплошного и полого сечений

### 2.3.2. График σ max(q)

Для построения данного графика использовалась формула, приведенная ниже (далее выдержка из программы):

public double stress(double M\_max, double y, double Jx) =>

M\_max \* y / Jx;

В результате было построено 2 графика, для сравнения максимальных напряжений при сплошном и полом сечениях.

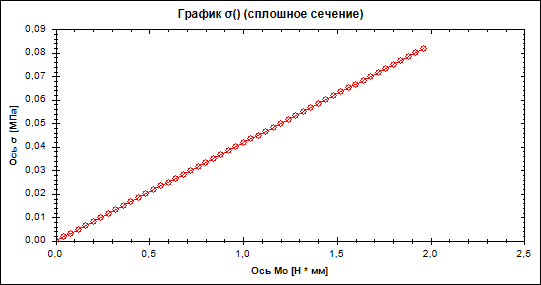


Рис. 2.11. График σ max(Mo) для сплошного сечения

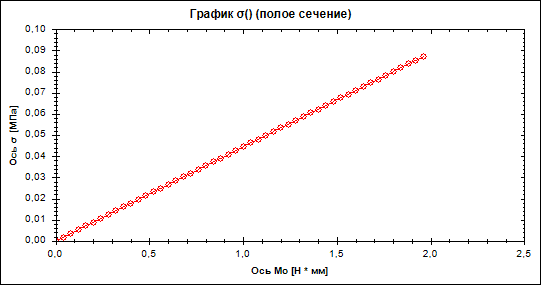


Рис. 2.12. График σ max(Mo) для полого сечения

## 2.4. Расчет массы и поиск оптимального профиля по массе

Для выполнения данного расчета использовались определенные формулы. Для начала, для расчета массы использовалась формула, приведенная ниже (далее выдержка из программы). Результат можно увидеть на рис. 1.5.

public double mass(double l, double b, double h, double p, double d = 0) =>

(b \* h - 3 \* Math.PI \* Math.Pow(d / 2, 2)) \* l \* p;

Для поиска оптимального профиля, использовалось условие равенства максимальных напряжений. Из этого условия была выведена формула, которая позволяет вывести ширину сечения B при изменении H и D и которая будет использована для поиска оптимального напряжения (далее выдержка из программы):

public double b(double step, double b\_0, double h, double d = 0) =>

12 \* (step / h \* Jx(b\_0, h) + 2 \* 0.141\* Math.Pow(d, 4)) / Math.Pow(step, 3);

step – изменяющийся H

b\_0 и h – параметры неизменного сплошного сечения

d – изменяющийся диаметр

В результате получился график зависимости B(H).

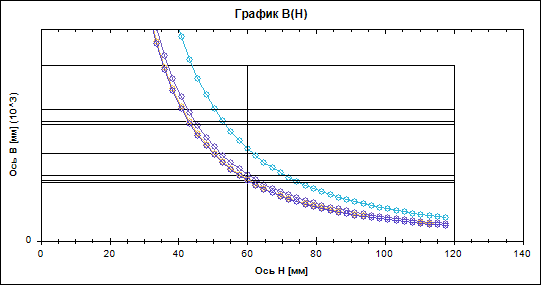


Рис. 2.13. График зависимости B(H)

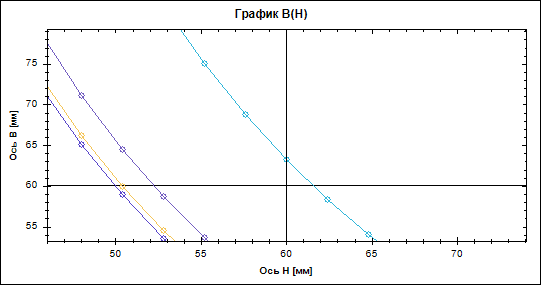


Рис. 2.14. График зависимости B(H) с приближением.

В итоге при использовании этих формулы происходил поиск оптимальной массы. Перебирались все высоты и диаметры (ширина считалась по формуле) и в итоге сравнивались значения массы при всех возможных вариантов. В итоге пользователю выводятся размеры наиболее подходящего по массе профиля, включая саму массу.

Часть кода, отвечающая за поиск:

for (double h = h\_0; h <= h\_0 \* 2; h++)

for (double d = 0; d < h / 3 && d < b; d++) {

b = formula.b(h, b\_0, h\_0, d);

if (b < b\_0) break;

mass = formula.mass(l / 1000, b / 1000, h / 1000, density \* 1000, d / 1000);

if (bestMass > mass)

{

bestH = h;

bestB = b;

bestD = d;

bestMass = mass;

}

}

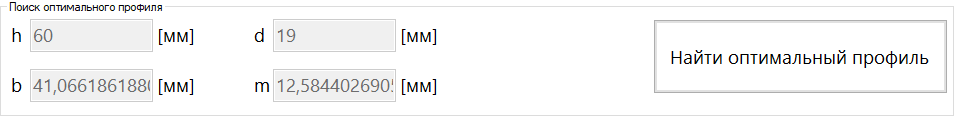


Рис. 2.14. Вывод результатов поиска

# 3. Дополнительный функционал программы

Программа выводит сообщение об ошибке, если в поле, в котором не должно быть чего-либо кроме чисел, ввести не число. В этом случае программа сообщит о данной ошибке, а в поле введет заранее созданное значение.

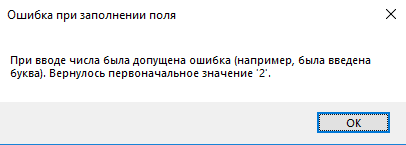
\*

Рис. 3. Сообщение об ошибке

Программа не предусматривает иного дополнительного функционала, кроме упомянутого выше.

# Заключение

В результате проделанной работы было создано приложение на языке программирования C#, которое выполняет анализ напряженного-деформированного состояние балок, выполняет необходимые расчеты, строит соответствующие эпюры и графики.

# Список используемой литературы

* В. Ф. Лянг. Учебное пособие по ИП – 2017. – 85 с.

# Приложение 1

using System;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

using ZedGraph;

namespace AveryanovCoursework3

{

class Formules

{

public double R\_a(double a, double q, double l) =>

q \* a / 6 \* (3 - 2 \* a / l);

public double R\_b(double a, double q, double l) =>

q \* Math.Pow(a, 2) / 3 / l;

public double getGreaterR(double a, double M0, double l) =>

R\_a(a, M0, l) > R\_b(a, M0, l) ? R\_a(a, M0, l) : R\_b(a, M0, l);

public double Q()=>

0;

public double Q\_max() =>

0;

public double M(double M0) =>

-M0;

public double M\_max(double M0) =>

M0;

public double f() =>

1;

public double Jx(double b, double h, double d = 0) =>

b \* Math.Pow(h, 3) / 12 - 2 \* 0.141 \* Math.Pow(d, 4);

public double w( double M0, double l, double E, double Jx, double z) =>

-(M0 \* Math.Pow(l, 2) / (2 \* E \* Jx)) \* Math.Pow((1 - (z / l)),2);

public double o(double M0, double l, double E, double Jx, double z) =>

( M0 \* z ) / ( E \* Jx );

public double stress(double M\_max, double y, double Jx) =>

M\_max \* y / Jx;

public double mass(double l, double b, double h, double p, double d = 0) =>

(b \* h - 3 \* Math.PI \* Math.Pow(d / 2, 2)) \* l \* p;

public double b(double step, double b\_0, double h, double d = 0) =>

12 \* (step / h \* Jx(b\_0, h) + 3 \* Math.PI \* Math.Pow(d, 4) / 64) / Math.Pow(step, 3);

}

public partial class Main : Form

{

private Formules formules = new Formules();

private double M0 = 2, l = 1000, E = 190000, yieldStrengthMax = 180, density = 7.8, h = 60, b = 40, d = 20, M\_example\_min = 0, M\_example\_max = 2, h\_0 =60, b\_0 = 40, d\_0 = 20, mass\_full, mass\_empty;

public Main()

{

InitializeComponent();

tb\_Mo.Text = tb\_Mo\_3\_max.Text = Convert.ToString(M0);

tb\_Mo\_2\_min.Text = tb\_Mo\_3\_min.Text = tb\_Mo\_4\_min.Text = Convert.ToString(M\_example\_min);

tb\_Mo\_2\_max.Text = tb\_Mo\_3\_max.Text = tb\_Mo\_4\_max.Text = Convert.ToString(M\_example\_max);

tb\_l\_0.Text = tb\_l.Text = Convert.ToString(l);

tb\_E.Text = Convert.ToString(E);

tb\_yieldStrengthMax.Text = Convert.ToString(yieldStrengthMax);

tb\_density.Text = Convert.ToString(density);

tb\_h\_0.Text = tb\_h\_1.Text = tb\_h\_3.Text = tb\_h\_5.Text = tb\_h.Text = Convert.ToString(h);

tb\_b\_0.Text = tb\_b\_1.Text = tb\_b\_3.Text = tb\_b\_4.Text = tb\_b.Text = Convert.ToString(b);

tb\_d\_0.Text = tb\_d\_4.Text = tb\_d.Text = Convert.ToString(d);

tb\_h\_min.Text = Convert.ToString(0);

tb\_h\_max.Text = Convert.ToString(h\_0 \* 2);

tb\_h\_result.Text = tb\_b\_result.Text = tb\_d\_result.Text = tb\_m\_result.Text = "";

zedGraphControl1.GraphPane.Title.Text = "Q";

zedGraphControl1.GraphPane.XAxis.Title.Text = "Ось Z [м]";

zedGraphControl1.GraphPane.YAxis.Title.Text = "Ось W [кН]";

zedGraphControl2.GraphPane.Title.Text = "M";

zedGraphControl2.GraphPane.XAxis.Title.Text = "Ось Z [м]";

zedGraphControl2.GraphPane.YAxis.Title.Text = "Ось W [кН \* м]";

zedGraphControl1.GraphPane.Title.Text = "Q";

zedGraphControl1.GraphPane.XAxis.Title.Text = "Ось Z [м]";

zedGraphControl1.GraphPane.YAxis.Title.Text = "Ось W [кН]";

zedGraphControl2.GraphPane.Title.Text = "M";

zedGraphControl2.GraphPane.XAxis.Title.Text = "Ось Z [м]";

zedGraphControl2.GraphPane.YAxis.Title.Text = "Ось W [кН \* м]";

zedGraphControl3.GraphPane.Title.Text = "График W(z)";

zedGraphControl3.GraphPane.XAxis.Title.Text = "Ось Z [м]";

zedGraphControl3.GraphPane.YAxis.Title.Text = "Ось W [мм]";

zedGraphControl4.GraphPane.Title.Text = "График θ(z)";

zedGraphControl4.GraphPane.XAxis.Title.Text = "Ось Z [м]";

zedGraphControl4.GraphPane.YAxis.Title.Text = "Ось W [рад]";

zedGraphControl5.GraphPane.Title.Text = "График W(z)";

zedGraphControl5.GraphPane.XAxis.Title.Text = "Ось Z [м]";

zedGraphControl5.GraphPane.YAxis.Title.Text = "Ось W [мм]";

zedGraphControl6.GraphPane.Title.Text = "График θ(z)";

zedGraphControl6.GraphPane.XAxis.Title.Text = "Ось Z [м]";

zedGraphControl6.GraphPane.YAxis.Title.Text = "Ось W [рад]";

zedGraphControl7.GraphPane.Title.Text = "График f()";

zedGraphControl7.GraphPane.XAxis.Title.Text = "Ось Mo [Н \* мм]";

zedGraphControl7.GraphPane.YAxis.Title.Text = "Ось f [мм]";

zedGraphControl8.GraphPane.Title.Text = "График σ() (сплошное сечение)";

zedGraphControl8.GraphPane.XAxis.Title.Text = "Ось Mo [Н \* мм]";

zedGraphControl8.GraphPane.YAxis.Title.Text = "Ось σ [МПа]";

zedGraphControl9.GraphPane.Title.Text = "График σ() (полое сечение)";

zedGraphControl9.GraphPane.XAxis.Title.Text = "Ось Mo [Н \* мм]";

zedGraphControl9.GraphPane.YAxis.Title.Text = "Ось σ [МПа]";

zedGraphControl10.GraphPane.Title.Text = "График B(H)";

zedGraphControl10.GraphPane.XAxis.Title.Text = "Ось H [мм]";

zedGraphControl10.GraphPane.YAxis.Title.Text = "Ось B [мм]";

}

private void draw\_1\_Click(object sender, EventArgs e)

{

zedGraphControl1.GraphPane.CurveList.Clear();

zedGraphControl2.GraphPane.CurveList.Clear();

PointPairList list = new PointPairList(), list1 = new PointPairList();

for (double z = 0; z <= l; z += l / 50)

{

list.Add(z / 1000, formules.Q());

list1.Add(z / 1000, formules.M(M0));

}

LineItem MyLine = zedGraphControl1.GraphPane.AddCurve("Q max " + Convert.ToString(formules.Q\_max()), list, Color.Black, SymbolType.None);

LineItem MyLine1 = zedGraphControl2.GraphPane.AddCurve("M max " + Convert.ToString(formules.M\_max(M0)), list1, Color.Black, SymbolType.None);

MyLine.Line.Fill = MyLine1.Line.Fill = new Fill(Color.Black);

MyLine.Symbol.IsVisible = MyLine1.Symbol.IsVisible = false;

zedGraphControl1.RestoreScale(zedGraphControl1.GraphPane);

zedGraphControl2.RestoreScale(zedGraphControl2.GraphPane);

}

private void zedGraphControl2\_ZoomEvent(ZedGraphControl sender, ZoomState oldState, ZoomState newState)

{

GraphPane pane = sender.GraphPane;

pane.XAxis.Scale.Min = 0;

pane.YAxis.Scale.Max = M0 \* 1.5;

}

private void draw\_2\_Click(object sender, EventArgs e)

{

zedGraphControl3.GraphPane.CurveList.Clear();

zedGraphControl4.GraphPane.CurveList.Clear();

PointPairList list = new PointPairList(), list1 = new PointPairList();

double w\_max = 0, w = 0, o\_max = 0, o = 0;

for (double z = 0; z <= l; z += l / 16)

{

w = formules.w(M0 , l / 1000, E \* Math.Pow(10, 6), formules.Jx(b / 1000, h / 1000), z / 1000) \* 1000;

o = formules.o(M0 \* 1000, l / 1000, E \* Math.Pow(10, 6), formules.Jx(b / 1000, h / 1000), z / 1000);

list.Add(z / 1000, w);

list1.Add(z / 1000, o);

if (w\_max > w) w\_max = w;

if (o > o\_max) o\_max = o;

}

tb\_w\_max.Text = Convert.ToString(w\_max);

tb\_O\_max.Text = Convert.ToString(o\_max);

LineItem MyLine = zedGraphControl3.GraphPane.AddCurve("", list, Color.Green, SymbolType.Circle);

LineItem MyLine1 = zedGraphControl4.GraphPane.AddCurve("", list1, Color.Blue, SymbolType.Circle);

zedGraphControl3.RestoreScale(zedGraphControl3.GraphPane);

zedGraphControl4.RestoreScale(zedGraphControl4.GraphPane);

}

private void draw\_3\_Click(object sender, EventArgs e)

{

zedGraphControl5.GraphPane.CurveList.Clear();

zedGraphControl6.GraphPane.CurveList.Clear();

PointPairList list = new PointPairList(), list1 = new PointPairList();

double w\_max = 0, w = 0, o\_max = 0, o = 0;

for (double z = 0; z <= l; z += l / 16)

{

w = formules.w(M0, l / 1000, E \* Math.Pow(10, 6), formules.Jx(b / 1000, h / 1000, d / 1000), z / 1000) \* 1000;

o = formules.o(M0 \* 1000, l / 1000, E \* Math.Pow(10, 6), formules.Jx(b / 1000, h / 1000, d / 1000), z / 1000);

list.Add(z / 1000, w);

list1.Add(z / 1000, o);

if (w\_max > w) w\_max = w;

if (o > o\_max) o\_max = o;

}

tb\_w\_max\_1.Text = Convert.ToString(w\_max);

tb\_O\_max\_1.Text = Convert.ToString(o\_max);

LineItem MyLine = zedGraphControl5.GraphPane.AddCurve("", list, Color.Green, SymbolType.Circle);

LineItem MyLine1 = zedGraphControl6.GraphPane.AddCurve("", list1, Color.Blue, SymbolType.Circle);

zedGraphControl5.RestoreScale(zedGraphControl5.GraphPane);

zedGraphControl6.RestoreScale(zedGraphControl6.GraphPane);

}

private void draw\_4\_Click(object sender, EventArgs e)

{

PointPairList list = new PointPairList(), list1 = new PointPairList();

for (double i = M\_example\_min; i <= M\_example\_max; i += M\_example\_max / 50)

{

list.Add(i, - formules.f() \* i);

list1.Add(i, - formules.f() \* i);

}

LineItem MyLine = zedGraphControl7.GraphPane.AddCurve("Сплошное сечение", list, Color.Red, SymbolType.Circle);

LineItem MyLine1 = zedGraphControl7.GraphPane.AddCurve("Полое сечение", list1, Color.Yellow, SymbolType.Circle);

zedGraphControl7.RestoreScale(zedGraphControl7.GraphPane);

}

private void clear\_4\_Click(object sender, EventArgs e)

{

zedGraphControl7.GraphPane.CurveList.Clear();

zedGraphControl7.RestoreScale(zedGraphControl7.GraphPane);

}

private void draw\_5\_Click(object sender, EventArgs e)

{

PointPairList list = new PointPairList();

for (double M0 = M\_example\_min; M0 <= M\_example\_max; M0 += M\_example\_max / 50) list.Add(M0, formules.stress(formules.M\_max(M0), h / 2 / 1000, formules.Jx(b / 1000, h / 1000)) \* Math.Pow(10, -6));

LineItem MyLine = zedGraphControl8.GraphPane.AddCurve("", list, Color.Red, SymbolType.Circle);

zedGraphControl8.RestoreScale(zedGraphControl8.GraphPane);

}

private void clear\_5\_Click(object sender, EventArgs e)

{

zedGraphControl8.GraphPane.CurveList.Clear();

zedGraphControl8.RestoreScale(zedGraphControl8.GraphPane);

}

private void draw\_6\_Click(object sender, EventArgs e)

{

PointPairList list = new PointPairList();

for (double M0 = M\_example\_min; M0 <= M\_example\_max; M0 += M\_example\_max / 50) list.Add(M0, formules.stress(formules.M\_max(M0), h / 2 / 1000, formules.Jx(b / 1000, h / 1000, d / 1000)) \* Math.Pow(10, -6));

LineItem MyLine = zedGraphControl9.GraphPane.AddCurve("", list, Color.Red, SymbolType.Circle);

zedGraphControl9.RestoreScale(zedGraphControl9.GraphPane);

}

private void clear\_6\_Click(object sender, EventArgs e)

{

zedGraphControl9.GraphPane.CurveList.Clear();

zedGraphControl9.RestoreScale(zedGraphControl9.GraphPane);

}

private void draw\_7\_Click(object sender, EventArgs e)

{

Random random = new Random();

Color color = Color.FromArgb(random.Next(256), random.Next(256), random.Next(256));

PointPairList list = new PointPairList(), list1 = new PointPairList(), list2 = new PointPairList(), list3 = new PointPairList();

for (double i = 0; i <= h\_0 \* 2; i += h\_0 / 25) list.Add(i, formules.b(i, b\_0, h\_0, d\_0));

list1.Add(0, b\_0 \* 2);

list1.Add(h\_0 \* 2, b\_0 \* 2);

list1.Add(h\_0 \* 2, 0);

list2.Add(h\_0, 0);

list2.Add(h\_0, b\_0 \* 2);

list3.Add(0, b\_0);

list3.Add(h\_0 \* 2, b\_0);

LineItem line = zedGraphControl10.GraphPane.AddCurve("", list, color, SymbolType.Circle);

LineItem line1 = zedGraphControl10.GraphPane.AddCurve("", list1, Color.Black, SymbolType.None);

LineItem line2 = zedGraphControl10.GraphPane.AddCurve("", list2, Color.Black, SymbolType.None);

LineItem line3 = zedGraphControl10.GraphPane.AddCurve("", list3, Color.Black, SymbolType.None);

zedGraphControl10.RestoreScale(zedGraphControl10.GraphPane);

}

private void clear\_7\_Click(object sender, EventArgs e)

{

zedGraphControl10.GraphPane.CurveList.Clear();

zedGraphControl10.RestoreScale(zedGraphControl10.GraphPane);

}

private void search\_Click(object sender, EventArgs e)

{

double b = b\_0, mass = formules.mass(l / 1000, b\_0 / 1000, h\_0 / 1000, density \* 1000), bestH, bestB, bestD, bestMass;

bestH = h\_0;

bestB = b\_0;

bestD = d\_0;

bestMass = mass;

for (double h = h\_0; h <= h\_0 \* 2; h++)

for (double d = 0; d < h / 3 && d < b; d++)

{

b = formules.b(h, b\_0, h\_0, d);

if (b < b\_0) break;

mass = formules.mass(l / 1000, b / 1000, h / 1000, density \* 1000, d / 1000);

if (bestMass > mass)

{

bestH = h;

bestB = b;

bestD = d;

bestMass = mass;

}

}

tb\_h\_result.Text = Convert.ToString(bestH);

tb\_b\_result.Text = Convert.ToString(bestB);

tb\_d\_result.Text = Convert.ToString(bestD);

tb\_m\_result.Text = Convert.ToString(bestMass);

}

private void mass()

{

mass\_full = formules.mass(l / 1000, b\_0 / 1000, h\_0 / 1000, density \* 1000);

tb\_mass\_full.Text = Convert.ToString(mass\_full);

mass\_empty = formules.mass(l / 1000, b\_0 / 1000, h\_0 / 1000, density \* 1000, d\_0 / 1000);

tb\_mass\_empty.Text = Convert.ToString(mass\_empty);

}

private double checkTextBoxChange(TextBox textBox, double defaultDouble) // Предотвращает ввод не double

{

try

{

return Convert.ToDouble(textBox.Text);

}

catch

{

textBox.Text = Convert.ToString(defaultDouble);

MessageBox.Show("При вводе числа была допущена ошибка (например, была введена буква). Вернулось первоначальное значение '" + defaultDouble + "'.", "Ошибка при заполнении поля");

return defaultDouble;

}

}

private void tb\_Mo\_TextChanged(object sender, EventArgs e)

{

M0 = M\_example\_max = checkTextBoxChange(tb\_Mo, 2);

}

private void tb\_l\_TextChanged(object sender, EventArgs e)

{

l = checkTextBoxChange(tb\_l, 4500);

tb\_l\_0.Text = tb\_l.Text;

}

private void tb\_h\_TextChanged(object sender, EventArgs e)

{

h = checkTextBoxChange(tb\_h, 175);

tb\_h\_1.Text = tb\_h\_3.Text = tb\_h\_4.Text = tb\_h.Text;

}

private void tb\_b\_TextChanged(object sender, EventArgs e)

{

b = checkTextBoxChange(tb\_b, 100);

tb\_b\_1.Text = tb\_b\_3.Text = tb\_b\_4.Text = tb\_b.Text;

}

private void tb\_d\_TextChanged(object sender, EventArgs e)

{

d = checkTextBoxChange(tb\_d, 58);

tb\_d\_4.Text = tb\_d.Text;

}

private void tb\_h\_0\_TextChanged(object sender, EventArgs e)

{

h\_0 = checkTextBoxChange(tb\_h\_0, 175);

mass();

tb\_h\_max.Text = Convert.ToString(h\_0 \* 2);

}

private void tb\_b\_0\_TextChanged(object sender, EventArgs e)

{

b\_0 = checkTextBoxChange(tb\_b\_0, 100);

mass();

}

private void tb\_d\_0\_TextChanged(object sender, EventArgs e)

{

d\_0 = checkTextBoxChange(tb\_d\_0, 0);

mass();

}

private void zedGraphControl10\_ZoomEvent(ZedGraphControl sender, ZoomState oldState, ZoomState newState)

{

GraphPane pane = sender.GraphPane;

if (pane.XAxis.Scale.Min <= -25) pane.XAxis.Scale.Min = -25;

if (pane.XAxis.Scale.Max >= h\_0 \* 2 + 25) pane.XAxis.Scale.Max = h\_0 \* 2 + 25;

if (pane.YAxis.Scale.Min <= -25) pane.YAxis.Scale.Min = -5;

if (pane.YAxis.Scale.Max >= b\_0 \* 2 + 25) pane.YAxis.Scale.Max = b\_0 \* 2 + 25;

}

}

}