Санкт-Петербургский политехнический университет Петра Великого

Высшая школа теоретической механики и математической физики

Отчёт по лабораторной работе №6 (2 семестр)

**Тема «Расчёт статического прогиба балки Бернулли-Эйлера»**

По дисциплине "Вычислительная механика"

Вариант 15

Выполнила студентка гр. 5030103/00101: А.Д.Работинский

Преподаватель: Е. Ю. Витохин

Санкт-Петербург

2022

**Содержание**

[Постановка задачи 3](#_Toc101746859)

[Метод решения 4](#_Toc101746860)

[Результаты 7](#_Toc101746861)

[Выводы 11](#_Toc101746862)

[Код программы 12](#_Toc101746863)

# Постановка задачи

Требуется рассчитать прогибы и изгибающие моменты стальной балки Бернулли-Эйлера под собственным весом. Сечение балки представляет угол с высотой h = 20 мм, шириной b = 20 мм, толщиной t = 4 мм.

Изображение выглядит как текст

Автоматически созданное описание

Для расчетов понадобятся физические постоянные, такие как плотность стали

(ρ= 8050 кг/м^3), модуль Юнга ( Н/м).

# Метод решения

Поле перемещений балки:

где угол поворота сечения.

Выражение для деформаций запишем в виде:

где кривизна балки, а – момент инерции

Как и в предыдущей работе воспользуемся подходом минимизации функционала потенциальной энергии.

Потенциальная энергия:

где – потенциальная энергия деформаций,а – работа внешних сил. В общем случае в данной лабораторной работе могут присутствовать и объёмные, и поверхностные, и сосредоточенные силы:

,

В данном случае нет ни поверхностных, ни сосредоточенных сил, а сила тяжести – объёмная сила.

– работа объёмных сил.

Мы будем проводить расчёты в изопараметрической системе координат для каждого элемента, после переходить обратно в глобальную СК.

Уравнение метода конечных элементов для одного элемента:

Введём вектор-столбец узловых перемещений:

Прогиб элемента в любой точке можно получить с помощью функций форм:

, где

Зная, что в общем случае кривизна – это вторая производная от прогибов по иксу, можем найти кривизну следующим образом:

матрица градиентов.

Имея кривизну, можем в тех же терминах задать потенциальную энергию деформации:

Подставим все вышеперечисленное в данный интеграл и получим конечные выражения для энергии и для локальной матрицы жесткости элемента:

Посчитаем интеграл, чтобы получить выражение для работы сил:

Далее, переходим к минимизации функционала потенциальной энергии: его минимум достигается, когда первая вариация потенциальной энергии по перемещениям равна нулю, в итоге получиим:

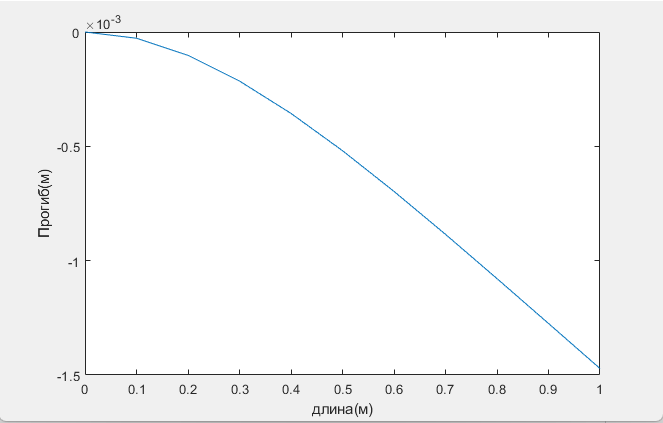
# Для нахождения основного уравнения МКЭ для всей системы будем складывать локальные матрицы жесткостей, однако конечная матрица имеет размерность 2n на 2n (n – количество узлов), а локальные матрицы 4 на 4, поэтому придётся прибегнуть к тому же построению, что и в прошлой лабораторной работе: будем строить матрицу А имеющую размерность 4 на 2n такую, чтобы перенести матрицу локальных жесткостей на определенное место в глобальную матрицу жесткостей, а далее будем складывать матрицы полученные умножением транспонированной матрицы А слева на локальную матрицу жесткостей и справа на саму матрицу А.

# Результаты

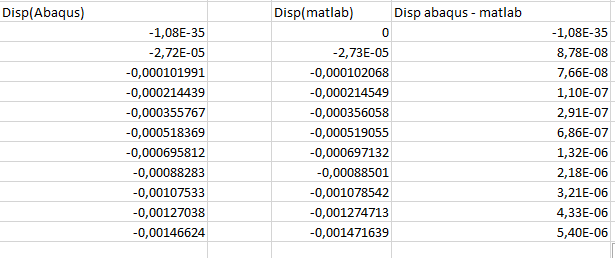
Проведем решение в конечно – элементном пакете Abaqus и непосредственно вручную в MatLab.

Прогибы:

Прогибы балки по расчетам в Abaqus



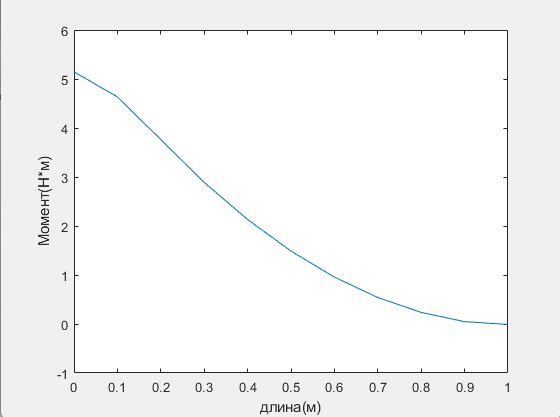
Прогибы балки по расчетам в MatLab



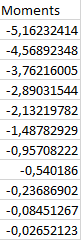
Численные результаты перемещений, полученные в Abaqus и MatLab

Моменты:

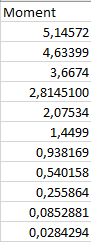
Изгибающие моменты, возникающие в балке Abaqus



Изгибающие моменты, возникающие в балке MatLab



Моменты полученные в MatLab

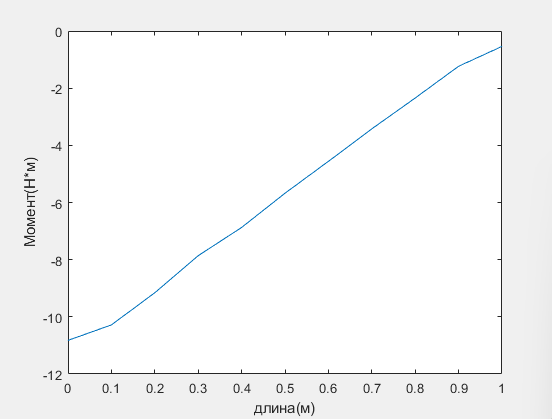


Моменты полученные в Abaqus

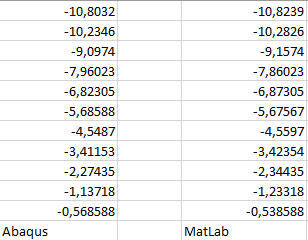
Стоит отметить, что противоположные значения не несут никакого физического смысла: т.к он зависит от точки отсчёта, то есть достаточно просто инвертировать график моментов полученный в MatLab и будет получен график из Abaqus.

Силы:

Cилы полученные в Abaqus



Силы полученные в MatLab



Cилы полученные в Abaqus и MatLab

# Выводы

Результаты, полученные в ходе работы в Abaqus и MatLab дали очень близкие результаты, что говорит о правильности приведенных построений.

# Код программы(выполнен в MatLab)

clc;

clear all;

E=2\*10^11;

nodes=11;

elems=10;

gravity=9.81;

% площадь ниже

square = 0.000144 ;

lengtht=[];

force\_gravity = [];

density = 8050;

nodesx=[0,0.100000001,0.200000003, 0.300000012,0.400000006, 0.5 , 0.600000024, 0.699999988 , 0.800000012 , 0.899999976, 1];

nodesy=[0, 0 ,0 ,0, 0, 0, 0 ,0 ,0 ,0, 0];

assoc = [[1,2];[2,3];[3,4];[4,5];[5,6];[6,7];[7,8];[8,9];[9,10];[10,11]];

inertion\_moment = 5.027555555555557e-9;

result = zeros(22,1);

result1 = zeros(22,1);

for i=1:nodes-1

lengtht = [lengtht nodesx(i+1)-nodesx(i)];

end

forces\_e = [[]]

k\_e = zeros(4);

K=zeros(2\*nodes);

K1=zeros(2\*nodes);

lengtht(1)

f\_e\_tmp =0.1/2\*(-gravity\*density\*square)\*[1;0.1/6;1;-0.1/6]

% fe = [f\_e\_tmp(1);f\_e\_tmp(2);f\_e\_tmp(3)+f\_e\_tmp(1);f\_e\_tmp(4)+f\_e\_tmp(2);f\_e\_tmp(3)+f\_e\_tmp(1);f\_e\_tmp(4)+f\_e\_tmp(2);f\_e\_tmp(3)+f\_e\_tmp(1);f\_e\_tmp(4)+f\_e\_tmp(2);f\_e\_tmp(3)+f\_e\_tmp(1);f\_e\_tmp(4)+f\_e\_tmp(2);f\_e\_tmp(3)+f\_e\_tmp(1);f\_e\_tmp(4)+f\_e\_tmp(2);f\_e\_tmp(3)+f\_e\_tmp(1);f\_e\_tmp(4)+f\_e\_tmp(2);f\_e\_tmp(3)+f\_e\_tmp(1);f\_e\_tmp(4)+f\_e\_tmp(2);f\_e\_tmp(3)+f\_e\_tmp(1);f\_e\_tmp(4)+f\_e\_tmp(2);f\_e\_tmp(3)+f\_e\_tmp(1);f\_e\_tmp(4)+f\_e\_tmp(2);f\_e\_tmp(1);f\_e\_tmp(2)];

fe(1)=0;

fe(2)=0;

Loads=zeros(2\*nodes,1);

for i=2:nodes-1

fe(2\*i-1)=f\_e\_tmp(1)+f\_e\_tmp(3);

fe(2\*i)=f\_e\_tmp(2)-f\_e\_tmp(4);

end

fe(2\*nodes-1)=f\_e\_tmp(3);

fe(2\*nodes)=f\_e\_tmp(4);

for i=1:10

Loads(2\*i-1:2\*i+2,1)=Loads(2\*i-1:2\*i+2,1) + f\_e\_tmp;

end

% fe = [f\_e\_tmp(1);f\_e\_tmp(2);];

Elements = zeros(elems, 4);

for el=1:elems

matrix\_for\_k\_e = zeros(4);

matrix\_for\_k\_e(1,1)=12;

matrix\_for\_k\_e(1,2)=6\*lengtht(el);

matrix\_for\_k\_e(1,3)=-12;

matrix\_for\_k\_e(1,4)=6\*lengtht(el);

matrix\_for\_k\_e(2,1)=6\*lengtht(el);

matrix\_for\_k\_e(2,2)=4\*lengtht(el)\*lengtht(el);

matrix\_for\_k\_e(2,3)=-6\*lengtht(el);

matrix\_for\_k\_e(2,4)=2\*lengtht(el)\*lengtht(el);

matrix\_for\_k\_e(3,1)=-12;

matrix\_for\_k\_e(3,2)=-6\*lengtht(el);

matrix\_for\_k\_e(3,3)=12;

matrix\_for\_k\_e(3,4)=-6\*lengtht(el);

matrix\_for\_k\_e(4,1)=6\*lengtht(el);

matrix\_for\_k\_e(4,2)=2\*lengtht(el)\*lengtht(el);

matrix\_for\_k\_e(4,3)=-6\*lengtht(el);

matrix\_for\_k\_e(4,4)=4\*lengtht(el)\*lengtht(el);

k\_e = E\*inertion\_moment/(0.1^3)\*matrix\_for\_k\_e;

A = zeros(4,22);

k\_v = 2\*assoc(el,1)-1;

k\_theta = 2\*assoc(el,1);

m\_v = 2\*assoc(el,2)-1;

m\_theta = 2\*assoc(el,2);

A(1,k\_v)=1;

A(2,k\_theta)=1;

A(3,m\_v)=1;

A(4,m\_theta)=1;

K\_temp = A'\*k\_e\*A;

K = K+K\_temp;

end

K(1,:)=0;

K(:,1)=0;

K(1,1)=1;

K(2,:)=0;

K(:,2)=0;

K(2,2)=1;

result = linsolve(K,fe');

Displacement = zeros(nodes,1);

Rotation = zeros(nodes,1);

for i = 2:nodes

Displacement(i)=result(2\*i-1);

Rotation(i)=result(2\*i);

end