Санкт-Петербургский политехнический университет

Высшая школа теоретической механики, ФизМех

Направление подготовки

«01.03.03 Механика и математическое моделирование»

Отчет по индивидуальной работе №**05**

**тема "** **Метод конечных элементов "**

**дисциплина "Вычислительная механика"**

Выполнил студент гр. 90301 **М. А.Бенюх**

Преподаватель: Е.Ю. Витохин

Санкт-Петербург

**2021**

Содержание:

[1. Формулировка задачи 3](#_Toc43323906)

[2. Алгоритм метода 3](#_Toc43323908)

[3. Результаты 4](#_Toc43323909)  
[4. Заключение 8](#_Toc43323909)

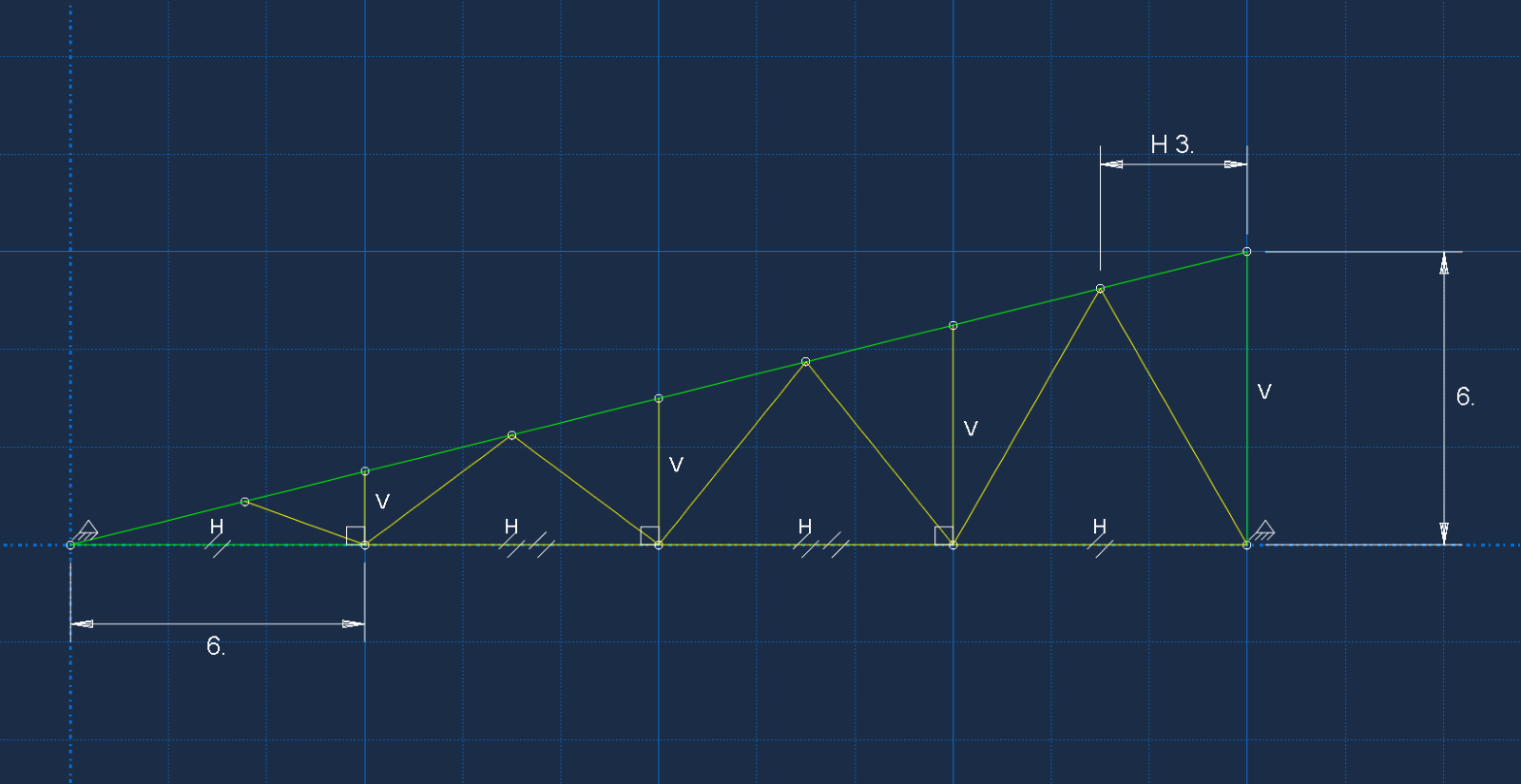
[5. Код программы 9](#_Toc43323909)

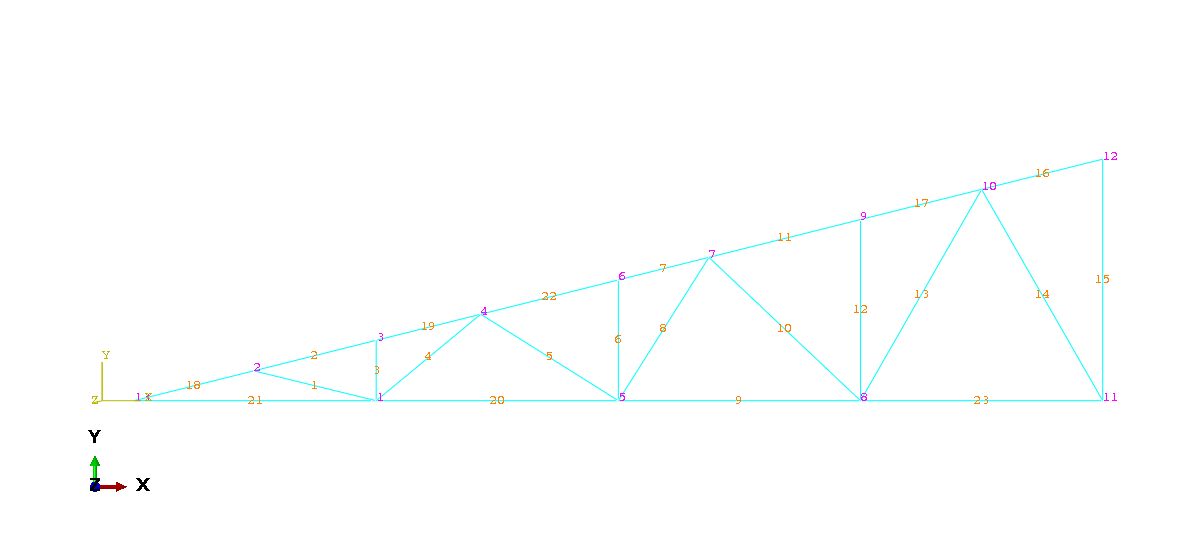
1. Формулировка задачи.  
   Произвести расчет плоских фермы под действием нагрузки **F**. В вариантах 1 – 4, 9, 11 – 13, 15 – 17, 19, 20 нагрузку следует прикладывать на верхний пояс. В остальных случаях нагрузка прикладывается на нижний пояс. Во всех случаях, кроме 12 и закрепляется крайне левый и правый нижний угол. Закрепление производится по горизонтальным и вертикальным степеням свободы. Требуется определить перемещения узлов фермы и усилия в стержнях.

Таблица 1. Параметры задачи

|  |  |
| --- | --- |
| Параметр | Значение |
| Площадь сечения **S** (м2) | 1 10-4 |
| Модуль Юнга **E** (Па) | 2 1011 |
| Сила **F** (Н) | 1 103 |

 Вариант 3





|  |
| --- |
|  |
| Нумерация узлов и элементов |

Получить:

1. поле ,
2. Таблицу сравнения и
3. Поле усилий
4. Таблицу сравнений
5. Алгоритм метода.

Для каждого элемента получим локальную матрицу жесткости:

Для каждого элемента соберем матрицу перехода:

Переедем к матрице жесткости для каждого элемента в глобальной системе координат при помощи матрицы перехода:

Соберем  матрицу жесткости системы:

Также обозначим вектор сил для системы:

Применим граничные условия:

Например, 1-й элемент закреплен:

Тогда решением основанного уравнения МКЭ: будет вектор перемещений

1. Результаты.

Результаты данной задачи представлены в сравнительной форме Abacus и алгоритма на MATLAB.



|  |
| --- |
|  |
| Таб. 1. Перемещения в узлах |

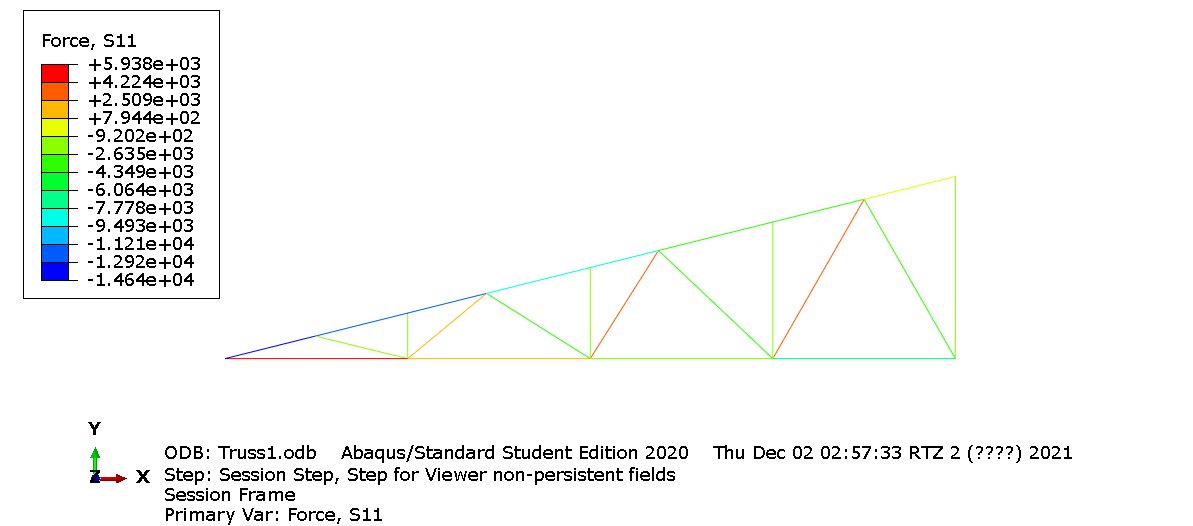
|  |
| --- |
|  |
| Таб. 2. Таблица сравнения |

|  |
| --- |
|  |
| Таб. 3. Таблица сравнения |

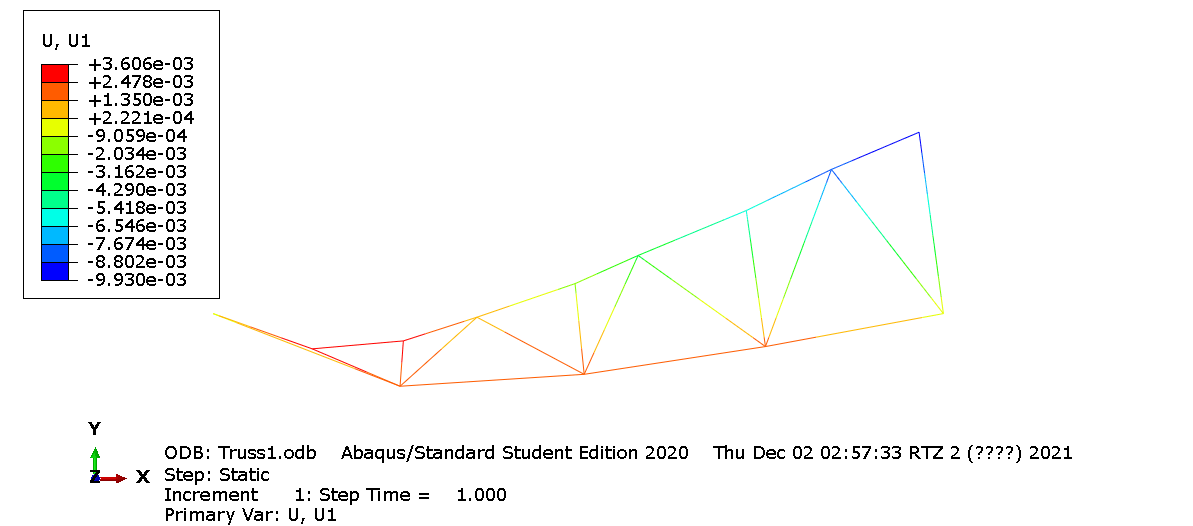


|  |
| --- |
|  |
| Таб. 4. Силы в узлах |

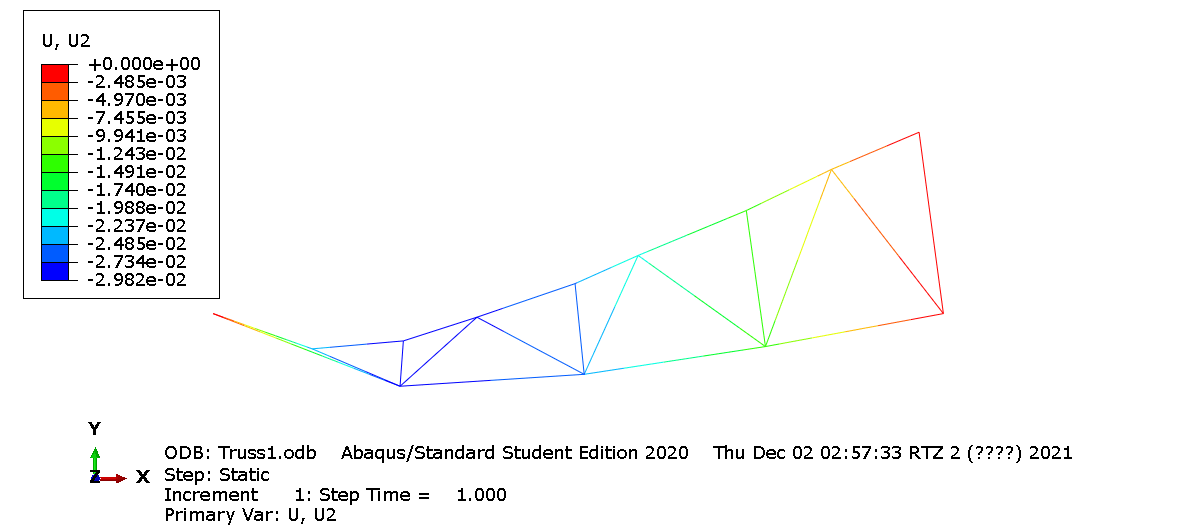
АБАКУС:



|  |
| --- |
|  |
| Рис. 2. Поле Сил [Н] |

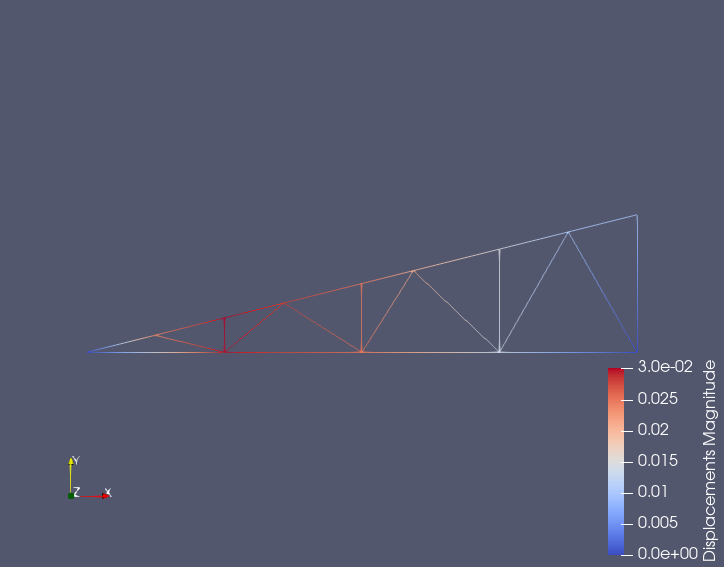


|  |
| --- |
|  |
| Рис. 2. Поле перемещений по X [м] |

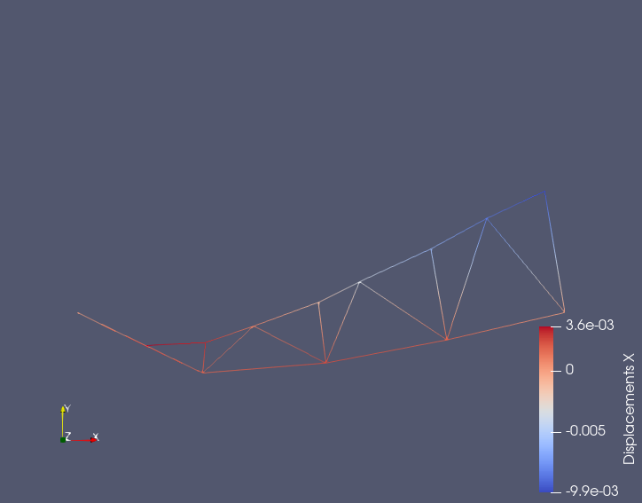
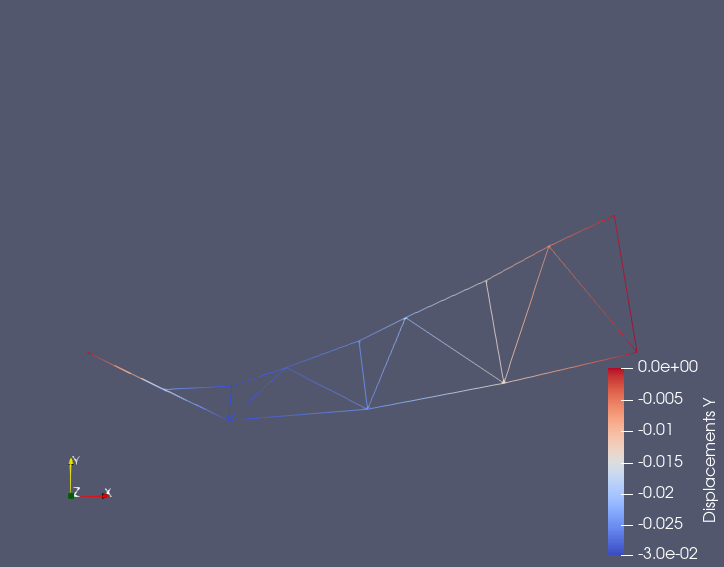


|  |
| --- |
|  |
| Рис. 3. Поле перемещений по Y [м] |

MATLAB:

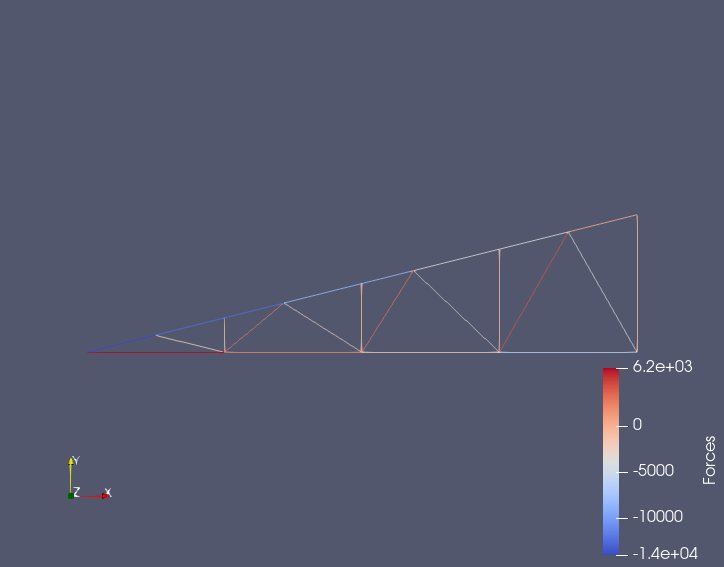
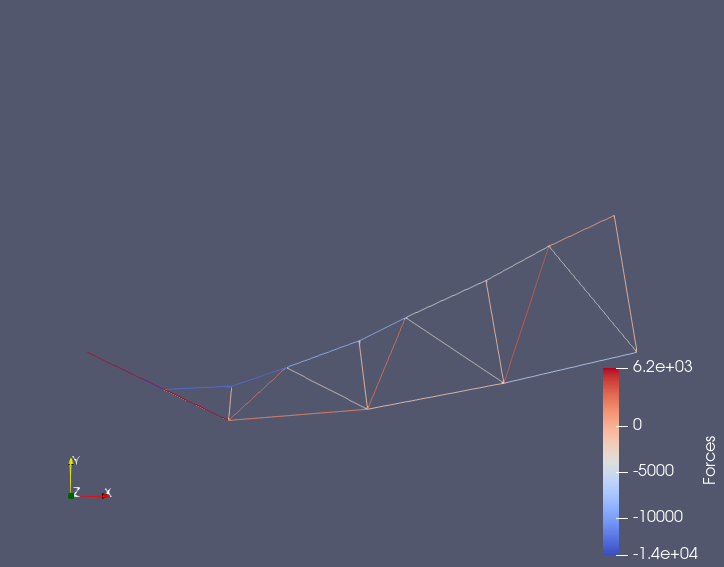


|  |
| --- |
|  |
| Рис. 5. Поле перемещения по X и Y [м] |

|  |
| --- |
|  |
| Рис. 7. Поле перемещения по Y [м] |

|  |
| --- |
|  |
| Рис. 6. Поле перемещения по X [м] |

|  |
| --- |
|  |
| Рис. 9. Поле Сил [Н] |

|  |
| --- |
|  |
| Рис. 8. Поле Сил [Н] |

1. Заключение

В рамках данной задачи с помощью метода конечных элементов были получены усилия и деформации, возникающие при растяжении-сжатии упругого стержня.

В ходе работы проведен расчет плоских ферм под действием нагрузки F, которая прикладывалась на верхний пояс, а закреплены – левый и правый нижний угол.

Далее был проверен МКЭ в Abaqus и MATLAB. Были получены результаты с точностью до 10 знака, которые оказались равны.

Использование Abaqus позволяет строить ферму и получать решение и визуализацию сразу в программном пакете, тогда как для визуализации в Matlab приходилось использовать стороннее ПО и задавать координаты узлов вручную.

Также не маловажно, что алгоритм на MatLab гибок для решения не стандартных задач, в отличие от Abaqus(не возможно изменить исходный код) и стоимость разработки на MatLab значительно ниже, а в случае портирования кода например на Python – зависит только от стоимости оплаты труда сотрудника.

1. Код программы

Mass\_node=[6.78988647, 0.;

3.75697184, 0.741771281;

6.78988647, 1.5;

9.38103199, 2.14778638;

12.7898865, 0.;

12.7898865, 3.;

15.0334682, 3.56089544;

18.7898865, 0.;

18.7898865, 4.5;

21.7898865, 5.25;

24.7898865, 0.;

24.7898865, 6.;

0.789886594, 0.;

]

Mass\_Element=[

1, 2;

3, 2;

1, 3;

4, 1;

5, 4;

5, 6;

7, 6;

7, 5;

5, 8;

8, 7;

9, 7;

8, 9;

10, 8;

11, 10;

11, 12;

12, 10;

10, 9;

2, 13;

4, 3;

1, 5;

13, 1;

6, 4;

8, 11;

]

GU\_el=[13,11] %закреп

N\_no=length(Mass\_node);

N\_el=length(Mass\_Element);

E=2e11; %юнг

A=0.0001; % площадь

l\_e=getLenEls(Mass\_Element,Mass\_node) %длина каждого элемента

massK\_loc=getK\_loc(E,A,l\_e) %\*[1,-1;-1,1] не забвть

%сосредоточеная сила

% F=zeros(length(Mass\_node),2);

% F(2,2)=1000 ; F(3,2)= 1000; F(5,2)=1000 ; F(9,2) = 1000; F(11,2) = 1000;

% F1\_gl = getF\_gl(F,Mass\_Element,Mass\_node, l\_e )

F=zeros(2\*length(Mass\_node),1);

% F(2\*2)=1000 ;

% F(3\*2)= -1000; F(5\*2)=-1000 ; F(9\*2) = -1000; %F(11\*2) = 1000;

F(2\*2)= -1000; F(3\*2)=-1000 ; F(4\*2)=-1000 ; F(6\*2)=-1000 ; F(7\*2)=-1000 ; F(9\*2)=-1000 ; F(10\*2)=-1000 ; F(12\*2)=-1000 ;

F1\_gl=F

%

%матрица жесткости

K\_gl=getK\_gl(Mass\_Element,massK\_loc,Mass\_node, l\_e )

%вектор сил

% F1\_gl= getF\_gl(F,Mass\_Element,Mass\_node, l\_e )

length(F1\_gl)

length(K\_gl(1,:))

%задние ГУ

[guF\_gl,guK\_gl]=setGuToFK(F1\_gl,K\_gl , GU\_el)

det(guK\_gl)

% guF\_gl=guF\_gl(1:length(guF\_gl)-1)

% guK\_gl=guK\_gl(1:length(guK\_gl)-1,:)

% guF\_gl=[guF\_gl(1:11),guF\_gl(13:length(guK\_gl)-1)]

% guK\_gl=[guK\_gl(1:11,:),guK\_gl(13:length(guK\_gl)-1,:)]

%перемещение

U=guK\_gl\guF\_gl

%деформации

new\_len=zeros(length(Mass\_Element),1);

deffs=zeros(length(Mass\_Element),1);

for i=1:length(Mass\_Element)

nodes=Mass\_Element(i,:);

X1=Mass\_node(nodes(1),1)+U(2\*nodes(1)-1);

X2=Mass\_node(nodes(2),1)+U(2\*nodes(2)-1);

Y1=Mass\_node(nodes(1),2)+U(2\*nodes(1));

Y2=Mass\_node(nodes(2),2)+U(2\*nodes(2));

new\_len(i)=sqrt((X2-X1)^2 +(Y2-Y1)^2);

deffs(i)=(new\_len(i)-l\_e(i))/l\_e(i);

end

stesses=deffs\*E

Forces=stesses\*A

function [F1\_gl,K\_gl]=setGuToFK(F1\_gl,K\_gl , GU\_el)

for i=1:length(F1\_gl)

for j=1:length(GU\_el)

if(i == 2\*(GU\_el(j)-1)+1)

F1\_gl(i) = 0;

F1\_gl(i+1) = 0;

K\_gl(i,:) = 0;

K\_gl(:,i) = 0;

K\_gl(i+1,:) = 0;

K\_gl(:,i+1) = 0;

K\_gl(i,i) = 1;

K\_gl(i+1,i+1) = 1;

end

end

end

end

function mass\_k\_loc=getK\_loc(E,A,l\_e)

mass\_k\_loc=zeros(length(l\_e),1);

for i=1: length(l\_e)

mass\_k\_loc(i)=E\*A/l\_e(i);

end

end

function mass\_l=getLenEls(Mass\_Element,Mass\_node)

mass\_l=zeros(length(Mass\_Element),1);

for i=1:length(Mass\_Element)

nodes=Mass\_Element(i,:)

mass\_l(i)=sqrt((Mass\_node(nodes(2),1)-Mass\_node(nodes(1),1))^2 +(Mass\_node(nodes(2),2)-Mass\_node(nodes(1),2))^2);

end

mass\_l

end

function T=getT(x\_loc,y\_loc,l)

l\_loc=(x\_loc(2)-x\_loc(1))/l;

m\_loc=(y\_loc(2)-y\_loc(1))/l;

T=[l\_loc, m\_loc, 0 0;

0, 0, l\_loc, m\_loc]

end

function K\_gl = getK\_gl(Mass\_Element,massK\_loc,Mass\_node, l\_e )

K\_gl=zeros(2\*length(Mass\_node));

for i =1:length(Mass\_Element)

nodes=Mass\_Element(i,:)

T=getT([Mass\_node(nodes(1),1),Mass\_node(nodes(2),1)],[Mass\_node(nodes(1),2),Mass\_node(nodes(2),2)], l\_e(i));

k\_loc =[1,-1;-1,1].\*massK\_loc(i)

T

k\_gl=T'\*k\_loc\*T

for k=1:2

for j=1:2

K\_gl(2\*(nodes(k)-1)+1+0,2\*(nodes(j)-1)+1+0) =K\_gl(2\*(nodes(k)-1)+1+0,2\*(nodes(j)-1)+1+0) +k\_gl(2\*(k-1)+1+0,2\*(j-1)+1+0);

K\_gl(2\*(nodes(k)-1)+1+0,2\*(nodes(j)-1)+1+1) =K\_gl(2\*(nodes(k)-1)+1+0,2\*(nodes(j)-1)+1+1) +k\_gl(2\*(k-1)+1+0,2\*(j-1)+1+1);

K\_gl(2\*(nodes(k)-1)+1+1,2\*(nodes(j)-1)+1+0) =K\_gl(2\*(nodes(k)-1)+1+1,2\*(nodes(j)-1)+1+0) +k\_gl(2\*(k-1)+1+1,2\*(j-1)+1+0);

K\_gl(2\*(nodes(k)-1)+1+1,2\*(nodes(j)-1)+1+1) =K\_gl(2\*(nodes(k)-1)+1+1,2\*(nodes(j)-1)+1+1) +k\_gl(2\*(k-1)+1+1,2\*(j-1)+1+1);

end

end

end

K\_gl

end