Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет имени
Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехника и комплексная автоматизация»
КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине «Модели и методы анализа проектных решений»

Студент:	Шашко Олег Владимирович
Группа:	РК6-71б
Тип задания:	Лабораторная работа
Название:	Метод конечных элементов
Вариант:	23

Студент _____ Шашко О. В.
подпись, дата Фамилия, И.О.

Преподаватель _____ Трудоношин В. А.
подпись, дата Фамилия, И.О.

Оценка: _____

Москва, 2023

Содержание

Метод конечных элементов

1 Цель выполнения лабораторной работы

Цель выполнения лабораторной работы – решение дифференциального уравнения методом конечных элементов (МКЭ), используя линейную и кубическую функции формы, и анализ точности относительной аналитического способа решения

2 Задание

Решить с помощью МКЭ уравнение 1

$$1\frac{d^2u}{dx^2} - 15u + 4 = 0, \tag{1}$$

при следующих граничных условиях (г. у.):

$$u(x = 3) = 10, \tag{2}$$

$$u'(x = 14) = 1. \tag{3}$$

Количество конечных элементов

- для первого расчета – 20,
- для второго – 40.

Также необходимо:

1. Сравнить результаты с аналитическим решением. Оценить максимальную погрешность.
2. Определить количество линейных КЭ, обеспечивающих такую же точность как и кубические.

3 Аналитическое решение

На рисунке 1 представлено аналитическое решение поставленной задачи.

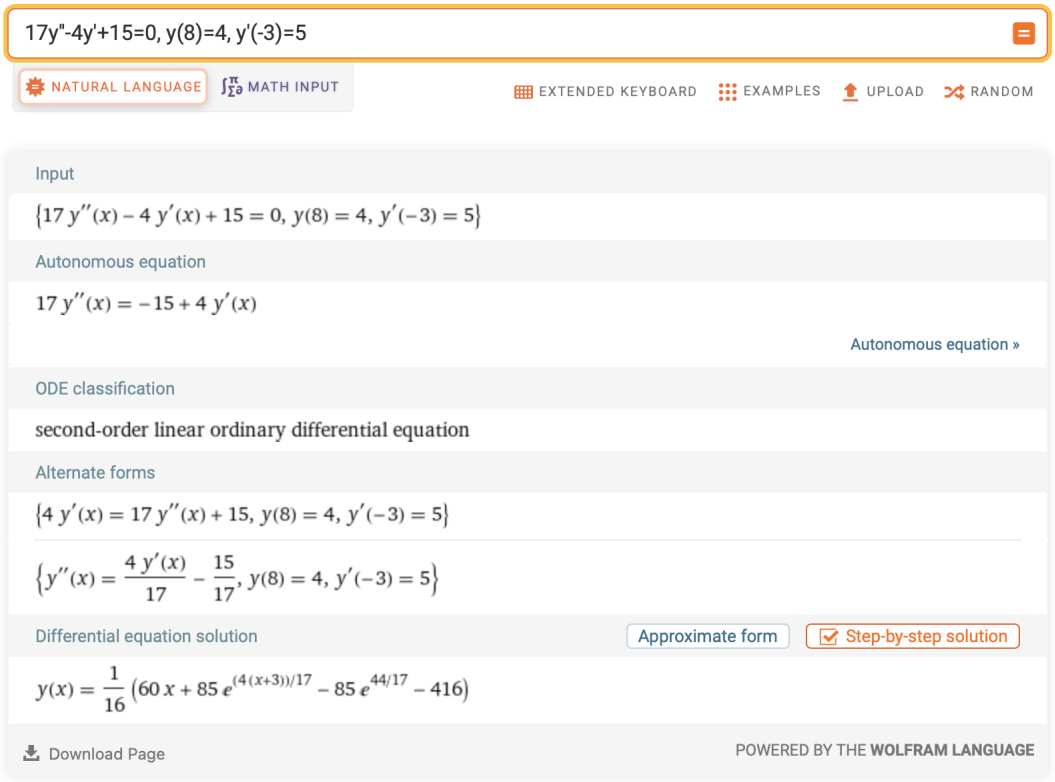

Рис. 1. Аналитическое решение

Таким образом, получаем:

$$u(x) = .$$

4 Получение локальных матрицы жесткости и вектора нагрузок

Составим локальные матрицу жесткости и вектор нагрузок для уравнения 1.

Линейная функция-формы КЭ

$$\mathbf{u} = \left[(1 - \tfrac{x}{L});\quad \tfrac{x}{L}\right]\begin{bmatrix} u_i \\ u_j \end{bmatrix} = \mathbf{N_e U},$$

где $\mathbf{N_e}$ – вектор функции формы конечного элемента (в данном случае линейной), его составляющие элементы - глобальные базисные функции, отличные от нуля в пределах этого элемента, L – длина КЭ.

В соответствии с методом Галеркина для уравнения 1:

$$\int_0^L \mathbf{W_e}\left(1\frac{d^2\mathbf{u}}{dx^2}\frac{d\mathbf{u}}{dx}+4\right)dx=0, \tag{4}$$

где $\mathbf{W_e}=\mathbf{N_e}^T$.

$$\int_0^L \mathbf{W_e}\left(1\frac{d^2\mathbf{u}}{dx^2}\frac{du}{dx}+4\right)dx=1\int_0^L \mathbf{W_e}\frac{d^2\mathbf{u}}{dx^2}dx\int_0^L \mathbf{W_e}\frac{d\mathbf{u}}{dx}dx+4\int_0^L \mathbf{W_e}dx=0$$

Распишем каждое слагаемое отдельно:

$$1\int_0^L \mathbf{W_e}\frac{d^2\mathbf{u}}{dx^2}dx=1\int_0^L\begin{bmatrix}(1-\frac{x}{L})\\ \frac{x}{L}\end{bmatrix}\frac{d^2\mathbf{u}}{dx^2}dx=1\begin{bmatrix}(1-\frac{x}{L})\\ \frac{x}{L}\end{bmatrix}\frac{d\mathbf{u}}{dx}|_0^L-$$

$$-1\int_0^L\frac{d}{dx}\begin{bmatrix}(1-\frac{x}{L})\\ \frac{x}{L}\end{bmatrix}\frac{d}{dx}\begin{bmatrix}(1-\frac{x}{L}); & \frac{x}{L}\end{bmatrix}\begin{bmatrix}u_i\\ u_j\end{bmatrix}=\begin{bmatrix}-1\frac{d\mathbf{u}}{dx}|_i\\ 1\frac{d\mathbf{u}}{dx}|_j\end{bmatrix}-1\begin{bmatrix}\frac{1}{L}, & -\frac{1}{L}\\ -\frac{1}{L}, & \frac{1}{L}\end{bmatrix}\begin{bmatrix}u_i\\ u_j\end{bmatrix}$$

$$\int_0^L \mathbf{W_e}\frac{d\mathbf{u}}{dx}dx=\int_0^L\begin{bmatrix}(1-\frac{x}{L})\\ \frac{x}{L}\end{bmatrix}\frac{d\mathbf{u}}{dx}\begin{bmatrix}u_i\\ u_j\end{bmatrix}dx=$$

$$=$$

$$\int_0^L\begin{bmatrix}(1-\frac{x}{L}) & (-1+\frac{x}{L})\\ -\frac{x}{L} & \frac{x}{L}\end{bmatrix}\begin{bmatrix}u_i\\ u_j\end{bmatrix}=$$

$$\begin{bmatrix}-\frac{1}{2}, & \frac{1}{2}\\ -\frac{1}{2}, & \frac{1}{2}\end{bmatrix}\begin{bmatrix}u_i\\ u_j\end{bmatrix}$$

$$4\int_0^L \mathbf{W_e}dx=4\begin{bmatrix}\frac{L}{2}\\ \frac{L}{2}\end{bmatrix}$$

Таким образом, для уравнения 4, при использовании линейной функции-формы, получаем (матмодель линейного КЭ):

$$\begin{bmatrix}1\frac{1}{L}\frac{1}{2}, & -1\frac{1}{L}\frac{1}{2}\\ -1\frac{1}{L}\frac{1}{2}, & 1\frac{1}{L}\frac{1}{2}\end{bmatrix}\begin{bmatrix}u_i\\ u_j\end{bmatrix}=\begin{bmatrix}-1\frac{du}{dx}|_i+4\frac{L}{2}\\ 1\frac{du}{dx}|_j+4\frac{L}{2}\end{bmatrix}$$

Кубическая функция-формы КЭ

$$\mathbf{u}=\left[-\frac{9x^3}{2L^3}+\frac{18x^2}{2L^2}-\frac{11x}{2L}+1;\frac{27x^3}{2L^3}-\frac{45x^2}{2L^2}+\frac{9x}{L};-\frac{27x^3}{2L^3}+\frac{36x^2}{2L^2}-\frac{9x}{2L};\frac{9x^3}{2L^3}-\frac{9x^2}{2L^2}-\frac{x}{L};\right]\begin{bmatrix}u_i\\ u_j\\ u_k\\ u_l\end{bmatrix}=\mathbf{N_e}\mathbf{U},$$

Как и для линейной функции-формы применим метод Галеркина (см. уравнение 4) и рассмотрим каждое слагаемое отдельно.

$$1\int_0^L \mathbf{W_e}\frac{d^2\mathbf{u}}{dx^2}dx = 1\int_0^L \begin{bmatrix} -\frac{9x^3}{2L^3} & + & \frac{18x^2}{2L^2} & - & \frac{11x}{2L} & + & 1 \\ \frac{27x^3}{2L^3} & - & \frac{45x^2}{2L^2} & + & \frac{9x}{L} & & \\ -\frac{27x^3}{2L^3} & + & \frac{36x^2}{2L^2} & - & \frac{9x}{2L} & & \\ \frac{9x^3}{2L^3} & - & \frac{9x^2}{2L^2} & - & \frac{x}{L} & & \end{bmatrix}\frac{d^2\mathbf{u}}{dx^2}dx =$$

$$= 1\begin{bmatrix} -\frac{9x^3}{2L^3} & + & \frac{18x^2}{2L^2} & - & \frac{11x}{2L} & + & 1 \\ \frac{27x^3}{2L^3} & - & \frac{45x^2}{2L^2} & + & \frac{9x}{L} & & \\ -\frac{27x^3}{2L^3} & + & \frac{36x^2}{2L^2} & - & \frac{9x}{2L} & & \\ \frac{9x^3}{2L^3} & - & \frac{9x^2}{2L^2} & - & \frac{x}{L} & & \end{bmatrix}\frac{d\mathbf{u}}{dx}|_0^L - 1\int_0^L \frac{d}{dx}\begin{bmatrix} -\frac{9x^3}{2L^3} & + & \frac{18x^2}{2L^2} & - & \frac{11x}{2L} & + & 1 \\ \frac{27x^3}{2L^3} & - & \frac{45x^2}{2L^2} & + & \frac{9x}{L} & & \\ -\frac{27x^3}{2L^3} & + & \frac{36x^2}{2L^2} & - & \frac{9x}{2L} & & \\ \frac{9x^3}{2L^3} & - & \frac{9x^2}{2L^2} & - & \frac{x}{L} & & \end{bmatrix}\frac{d}{dx}\mathbf{u} =$$

$$= \begin{bmatrix} -1\frac{d\mathbf{u}}{dx}|_i \\ 0 \\ 0 \\ 1\frac{d\mathbf{u}}{dx}|_l \end{bmatrix} - 1\begin{bmatrix} \frac{37}{10L} & -\frac{189}{40} & \frac{27}{20L} & -\frac{13}{40L} \\ -\frac{189}{40} & \frac{54}{5L} & -\frac{297}{40L} & \frac{27}{20L} \\ \frac{27}{20L} & -\frac{297}{40L} & \frac{54}{5L} & -\frac{189}{40} \\ -\frac{13}{40L} & \frac{27}{20L} & -\frac{189}{40} & \frac{37}{10L} \end{bmatrix}\begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix}$$

$$\int_0^L \mathbf{W_e}\frac{d\mathbf{u}}{dx}dx = \int_0^L \begin{bmatrix} -\frac{9x^3}{2L^3} & + & \frac{18x^2}{2L^2} & - & \frac{11x}{2L} & + & 1 \\ \frac{27x^3}{2L^3} & - & \frac{45x^2}{2L^2} & + & \frac{9x}{L} & & \\ -\frac{27x^3}{2L^3} & + & \frac{36x^2}{2L^2} & - & \frac{9x}{2L} & & \\ \frac{9x^3}{2L^3} & - & \frac{9x^2}{2L^2} & - & \frac{x}{L} & & \end{bmatrix}\frac{d\mathbf{u}}{dx}dx =$$

$$=$$

$$\begin{bmatrix} -\frac{1}{2} & \frac{57}{80} & -\frac{3}{10} & \frac{7}{80} \\ -\frac{57}{80} & 0 & \frac{81}{80} & -\frac{3}{10} \\ \frac{3}{10} & -\frac{81}{80} & 0 & -\frac{3}{10} \\ -\frac{7}{80} & \frac{3}{10} & -\frac{57}{80} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix}$$

$$4\int_0^L \mathbf{W_e}dx = 4\begin{bmatrix} \frac{L}{8} \\ \frac{3L}{8} \\ \frac{3L}{8} \\ \frac{L}{8} \end{bmatrix}$$

Таким образом, для уравнения 4, при использовании кубической функции-формы, получаем:

$$\begin{bmatrix} 1\frac{37}{10L}\frac{1}{2} & -1\frac{189}{40L}\frac{57}{80} & 1\frac{27}{20L}\frac{3}{10} & -1\frac{13}{40L}\frac{7}{80} \\ -1\frac{189}{40L}\frac{57}{80} & 1\frac{54}{5L}+0 & -1\frac{297}{40L}\frac{81}{80} & 1\frac{27}{20L}\frac{3}{10} \\ 1\frac{27}{20L}\frac{3}{10} & -1\frac{297}{40L}\frac{81}{80} & 1\frac{54}{5L}+0 & -1\frac{189}{40L}\frac{57}{80} \\ -1\frac{13}{40L}\frac{7}{80} & 1\frac{27}{20L}\frac{3}{10} & -1\frac{189}{40L}\frac{57}{80} & 1\frac{37}{10L}\frac{1}{2} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix} = \begin{bmatrix} 4\frac{L}{8} - 1\frac{du}{dx}|_i \\ 4\frac{3L}{8} \\ 4\frac{3L}{8} \\ 4\frac{L}{8} + 1\frac{du}{dx}|_l \end{bmatrix} \tag{5}$$

Локальные матрицу жесткости и вектор нагрузок из уравнения 5 с помощью матричных преобразований приведем к следующему виду:

$$\begin{bmatrix} a_{11} & 0 & 0 & a_{14} \\ a_{21} & a_{22} & 0 & a_{24} \\ a_{31} & 0 & a_{33} & a_{34} \\ a_{41} & 0 & 0 & a_{44} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix} = \begin{bmatrix} b_1 - 1\frac{du}{dx}|_i \\ b_2 \\ b_3 \\ b_4 + 1\frac{du}{dx}|_l \end{bmatrix}$$

Для упрощения расчетов преобразуем систему выше, исключив внутренние узлы. Таким образом СЛАУ (математическая модель кубического КЭ):

$$\begin{bmatrix} a_{11} & a_{14} \\ a_{41} & a_{44} \end{bmatrix} \begin{bmatrix} u_i \\ u_l \end{bmatrix} = \begin{bmatrix} b_1 - 1\frac{du}{dx}|_i \\ b_4 + 1\frac{du}{dx}|_l \end{bmatrix}$$

5 Получение глобальных матрицы жесткости и вектора нагрузок

Проведем процедуры ансамблирования и учет граничных условий для формирования итоговой математической модели.

Ансамблирование

Пусть локальные матрица жесткости и вектор неизвестных заданы следующим образом

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} = \begin{bmatrix} b_1 - 1\frac{du}{dx}|_i \\ b_2 + 1\frac{du}{dx}|_l \end{bmatrix},$$

тогда, при разбитие области на n КЭ, глобальная матрица жесткости будет иметь размерность $(n+1)\cdot(n+1)$:

$$\begin{bmatrix} a_{11}^1 & a_{12}^1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ a_{21}^1 & a_{22}^1 + a_{11}^2 & a_{12}^2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_{21}^2 & a_{22}^2 + a_{11}^3 & a_{12}^3 & 0 & \cdots & 0 & 0 \\ 0 & 0 & a_{21}^3 & a_{22}^3 + \cdots & & & & 0 \\ \vdots & \vdots & \vdots & \vdots & & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots + a_{11}^n & a_{12}^n \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{21}^n & a_{22}^n \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} b_1^1 - 1\frac{du}{dx}|_0 \\ b_2^1 + b_1^2 \\ b_2^2 + b_1^3 \\ b_2^3 + b_1^4 \\ \vdots \\ b_2^{n-1} + b_1^n \\ b_2^n + 1\frac{du}{dx}|_L \end{bmatrix}$$

Учет граничных условий

Применим граничные условия первого (см. 3) и второго рода (см. 2) к выведенной выше системе.

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ a_{21}^1 & a_{22}^1 + a_{11}^2 & a_{12}^2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_{21}^2 & a_{22}^2 + a_{11}^3 & a_{12}^3 & 0 & \cdots & 0 & 0 \\ 0 & 0 & a_{21}^3 & a_{22}^3 + \cdots & & & & 0 \\ \vdots & \vdots & \vdots & \vdots & & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots + a_{11}^n & a_{12}^n \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} 10 \\ b_2^1 + b_1^2 \\ b_2^2 + b_1^3 \\ b_2^3 + b_1^4 \\ \vdots \\ b_2^{n-1} + b_1^n \\ b_2^n + 1 \cdot 1 \end{bmatrix}$$

6 Анализ результатов

Проведем сравнение результатов согласно заданию.

Линейная функция-формы

На рисунках 2, 3 представлены графики полученные с помощью МКЭ (линейная функция-формы).

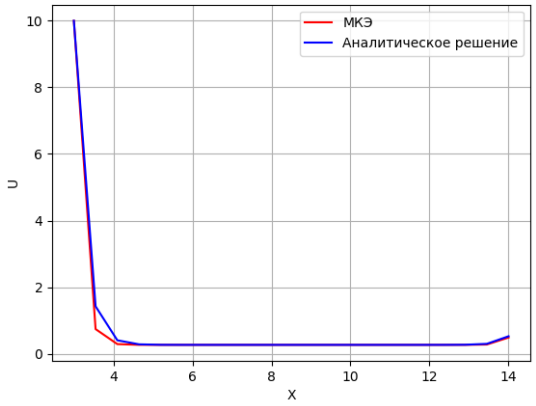

Рис. 2. Результат работы программы для 20 КЭ

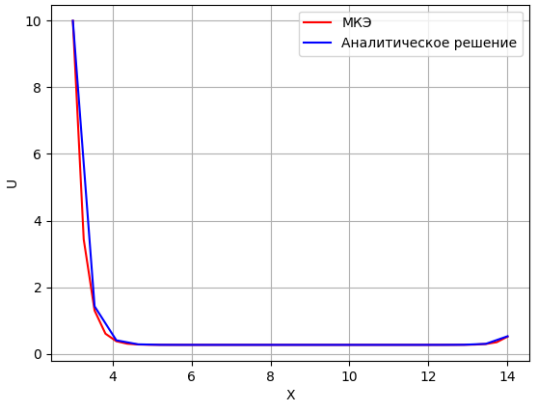

Рис. 3. Результат работы программы для 40 КЭ

X	Аналитическое решение	МКЭ-решение	Абсолютная погрешность
3.000000	10.000000	10.000000	0.000000
3.550000	1.423187	0.739922	0.683265
4.100000	0.404085	0.289677	0.114408
4.650000	0.282995	0.267785	0.015209
5.200000	0.268607	0.266721	0.001886
5.750000	0.266897	0.266669	0.000228
6.300000	0.266694	0.266667	0.000027
6.850000	0.266670	0.266667	0.000003
7.400000	0.266667	0.266667	0.000000
7.950000	0.266667	0.266667	0.000000
8.500000	0.266667	0.266667	0.000000
9.050000	0.266667	0.266667	0.000000
9.600000	0.266667	0.266667	0.000000
10.150000	0.266667	0.266667	0.000000
10.700000	0.266667	0.266667	0.000001
11.250000	0.266673	0.266667	0.000006
11.800000	0.266718	0.266668	0.000050
12.350000	0.267100	0.266692	0.000408
12.900000	0.270312	0.267187	0.003125
13.450000	0.297346	0.277361	0.019985
14.000000	0.524866	0.486610	0.038256

Таблица 1. 20 линейных КЭ

X	Аналитическое решение	МКЭ-решение	Абсолютная погрешность
3.000000	10.000000	10.000000	0.000000
3.275000	3.621782	3.433544	0.188238
3.550000	1.423187	1.297055	0.126132
3.825000	0.665323	0.601918	0.063405
4.100000	0.404085	0.375745	0.028340
4.375000	0.314035	0.302157	0.011878
4.650000	0.282995	0.278214	0.004781
4.925000	0.272295	0.270424	0.001871
5.200000	0.268607	0.267889	0.000718
5.475000	0.267335	0.267064	0.000271
5.750000	0.266897	0.266796	0.000101
6.025000	0.266746	0.266709	0.000037
6.300000	0.266694	0.266680	0.000014
6.575000	0.266676	0.266671	0.000005
6.850000	0.266670	0.266668	0.000002
7.125000	0.266668	0.266667	0.000001
7.400000	0.266667	0.266667	0.000000
7.675000	0.266667	0.266667	0.000000
7.950000	0.266667	0.266667	0.000000
8.225000	0.266667	0.266667	0.000000
8.500000	0.266667	0.266667	0.000000
8.775000	0.266667	0.266667	0.000000
9.050000	0.266667	0.266667	0.000000
9.325000	0.266667	0.266667	0.000000
9.600000	0.266667	0.266667	0.000000
9.875000	0.266667	0.266667	0.000000
10.150000	0.266667	0.266667	0.000000
10.425000	0.266667	0.266667	0.000000
10.700000	0.266667	0.266667	0.000000
10.975000	0.266669	0.266668	0.000001
11.250000	0.266673	0.266670	0.000003
11.525000	0.266684	0.266677	0.000008
11.800000	0.266718	0.266698	0.000020
12.075000	0.266816	0.266762	0.000054
12.350000	0.267100	0.266959	0.000140
12.625000	0.267923	0.267567	0.000357
12.900000	0.270312	0.269432	0.000880
13.175000	0.277242	0.275167	0.002075
13.450000	0.297346	0.292793	0.004553
13.725000	0.355669	0.346966	0.008703
14.000000	0.524866	0.513464	0.011402

Таблица 2. 40 линейных КЭ

Максимальная абсолютная погрешность 6.832646e-01 и 1.882376e-01 соответственно.

Кубическая функция-формы

На рисунках 4, 5 представлены графики полученные с помощью МКЭ (кубическая функция-формы).

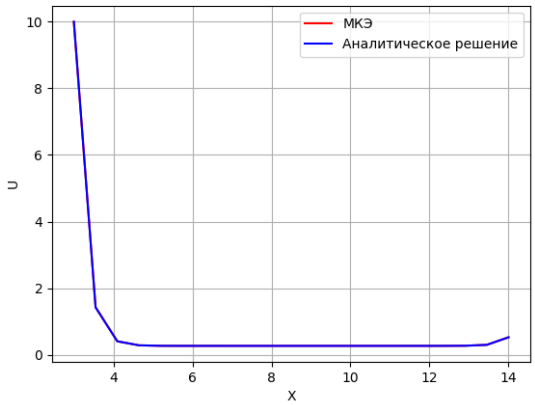

Рис. 4. Результат работы программы для 20 КЭ

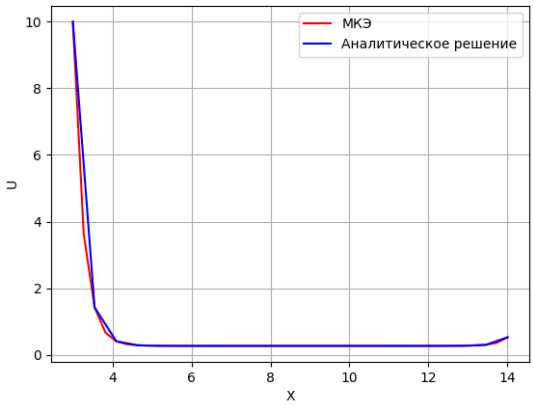

Рис. 5. Результат работы программы для 40 КЭ

X	Аналитическое решение	МКЭ-решение	Абсолютная погрешность
3.000000	10.000000	10.000000	0.000000
3.550000	1.423187	1.421854	0.001333
4.100000	0.404085	0.403768	0.000317
4.650000	0.282995	0.282938	0.000056
5.200000	0.268607	0.268598	0.000009
5.750000	0.266897	0.266896	0.000001
6.300000	0.266694	0.266694	0.000000
6.850000	0.266670	0.266670	0.000000
7.400000	0.266667	0.266667	0.000000
7.950000	0.266667	0.266667	0.000000
8.500000	0.266667	0.266667	0.000000
9.050000	0.266667	0.266667	0.000000
9.600000	0.266667	0.266667	0.000000
10.150000	0.266667	0.266667	0.000000
10.700000	0.266667	0.266667	0.000000
11.250000	0.266673	0.266673	0.000000
11.800000	0.266718	0.266718	0.000000
12.350000	0.267100	0.267098	0.000002
12.900000	0.270312	0.270303	0.000009
13.450000	0.297346	0.297302	0.000044
14.000000	0.524866	0.524791	0.000074

Таблица 3. 20 кубических КЭ

X	Аналитическое решение	МКЭ-решение	Абсолютная погрешность
3.000000	10.000000	10.000000	0.000000
3.275000	3.621782	3.621755	0.000027
3.550000	1.423187	1.423168	0.000019
3.825000	0.665323	0.665314	0.000010
4.100000	0.404085	0.404081	0.000004
4.375000	0.314035	0.314033	0.000002
4.650000	0.282995	0.282994	0.000001
4.925000	0.272295	0.272295	0.000000
5.200000	0.268607	0.268607	0.000000
5.475000	0.267335	0.267335	0.000000
5.750000	0.266897	0.266897	0.000000
6.025000	0.266746	0.266746	0.000000
6.300000	0.266694	0.266694	0.000000
6.575000	0.266676	0.266676	0.000000
6.850000	0.266670	0.266670	0.000000
7.125000	0.266668	0.266668	0.000000
7.400000	0.266667	0.266667	0.000000
7.675000	0.266667	0.266667	0.000000
7.950000	0.266667	0.266667	0.000000
8.225000	0.266667	0.266667	0.000000
8.500000	0.266667	0.266667	0.000000
8.775000	0.266667	0.266667	0.000000
9.050000	0.266667	0.266667	0.000000
9.325000	0.266667	0.266667	0.000000
9.600000	0.266667	0.266667	0.000000
9.875000	0.266667	0.266667	0.000000
10.150000	0.266667	0.266667	0.000000
10.425000	0.266667	0.266667	0.000000
10.700000	0.266667	0.266667	0.000000
10.975000	0.266669	0.266669	0.000000
11.250000	0.266673	0.266673	0.000000
11.525000	0.266684	0.266684	0.000000
11.800000	0.266718	0.266718	0.000000
12.075000	0.266816	0.266816	0.000000
12.350000	0.267100	0.267100	0.000000
12.625000	0.267923	0.267923	0.000000
12.900000	0.270312	0.270312	0.000000
13.175000	0.277242	0.277242	0.000000
13.450000	0.297346	0.297345	0.000001
13.725000	0.355669	0.355667	0.000001
14.000000	0.524866	0.524864	0.000002

Таблица 4. 40 кубических КЭ

Максимальная абсолютная погрешность 1.333048e-03 и 2.691318e-05 соответственно.

Нахождение количества линейных КЭ, обеспечивающих ту же точность, что и 20 кубических

Так как очевидно, что при увлечении числа КЭ точность растет, найдем искомое следуя алгоритму, представленному на рисунке 6.

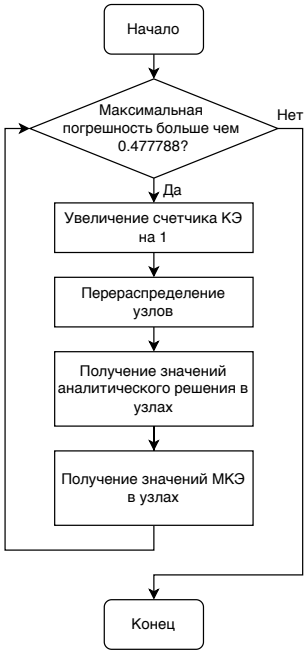

Рис. 6. Алгоритм нахождения количества КЭ, заданную точность

Реализовав данный алгоритм с начальным количеством КЭ=20 и увеличивая счетчик всегда на 1 получаем необходимое количество КЭ, равное 45034 .

7 Код

Листинг 1. *Реализация МКЭ*

```
1
2   #include <iostream>
3 #include <vector>
4 #include <cmath>
5
6 double EPS = 1e-16;
7 double X_BEGIN = 3.0;
8 double X_END = 14.0;
9 size_t ELEMS_NUM = 20;
```

```
10 double L = (X_END − X_BEGIN) / ELEMS_NUM;
11
12 double a = 1.0, B = 0.0, C = −15.0, D = 4.0, usl_left = 10.0, usl_right = 1.0; //
       au"+Bu'+Cu+D=0
13
14 std::vector<double> solve_with_gauss(std::vector<std::vector<double>>& A,
       std::vector<double>& b){
15     size_t row_size = A.size();
16     size_t col_size = A.back().size();
17
18     // Прямой ход Гаусса
19     double pivot = 0.;
20     for (size_t i = 0; i < row_size; i++) {
21         for (size_t j = i + 1; j < col_size; j++) {
22             if (std::abs(A.at(j).at(i)) < EPS) {
23                 continue;
24             }
25             pivot = A.at(j).at(i) / A.at(i).at(i);
26             b.at(j) −= pivot * b.at(i);
27             for (size_t k = 0; k < row_size; k++) {
28                 A.at(j).at(k) −= pivot * A.at(i).at(k) ;
29             }
30         }
31     }
32
33     // Обратный ход Гаусса
34     std::vector<double> x(row_size);
35     for (int i = row_size − 1.; i >= 0; i−−) {
36         x.at(i) = b.at(i);
37         for (size_t j = i + 1; j < row_size; j++) {
38             x.at(i) −= x.at(j) * A.at(i).at(j);
39         }
40         x.at(i) /= A.at(i).at(i);
41     }
42
43     return x;
44 }
45
46 double analytical_solution(double x) {
47     return (exp(−sqrt(15.) * (x + 3.)) * (146. * exp(2. * sqrt(15.) * x) + 4. * exp(sqrt(15.)
           * (x + 3.)) + 4. * exp(sqrt(15.) * (x + 25.)) + sqrt(15) * exp(sqrt(15.) * (2. * x +
           11.)) − sqrt(15.) * exp(17. * sqrt(15.)) + 146. * exp(28. * sqrt(15.)))) / (15. * (1.
           + exp(22. * sqrt(15.))));
48 }
49
50 std::vector<double> build_analytical_solution(std::vector<double>& x_vec) {
```

```
51      size_t x_vec_size = x_vec.size();
52      std::vector<double> y_vec = std::vector<double>(x_vec_size);
53      for (size_t i = 0; i < x_vec_size; i++) {
54          y_vec.at(i) = analytical_solution(x_vec.at(i));
55      }
56      return y_vec;
57  }
58
59  std::vector<double> build_linear_solution(size_t elems_num) {
60      double L = (X_END - X_BEGIN) / elems_num;
61      size_t size = elems_num + 1;
62      std::vector< std::vector<double> > A(size, std::vector<double>(size));
63      std::vector<double> b(size);
64
65      // Локальная матрица жесткости для линейного КЭ
66      std::vector< std::vector<double> > local_matrix = {
67          { a/L - C * L/3.0 + B*1.0/2.0, -a/L - C * L/6.0 - B*1.0/2.0},
68          { -a/L - C * L/6.0 + B*1.0/2.0, a/L - C*L/3.0 - B*1.0/2.0},
69      };
70
71      // Ансамблирование и получение глобальной матрицы жесткости для линейного КЭ
72      for (size_t i = 0; i < elems_num; i++) {
73          for (size_t j = 0; j < 2; j++) {
74              for (size_t k = 0; k < 2; k++) {
75                  A.at(i + j).at(i + k) += local_matrix.at(j).at(k);
76              }
77          }
78      }
79
80      for (size_t i = 0; i < size ; i++) {
81          b.at(i) = D * L;
82      }
83
84      // Учет ГУ
85      if ( 0 == 1 ) {
86          b.at(0) = D * L /2. - a*usl_left;
87      } else {
88          b.at(0) = usl_left;
89          A.at(0).at(0) = 1;
90          A.at(0).at(1) = 0;
91      }
92
93      if ( 1 == 1 ) {
94          b.at(size - 1) = D * L /2. + a*usl_right;
95      } else {
96          b.at(size - 1) = usl_right;
```

```
97          A.at(size - 1).at(size - 1) = 1;
98          A.at(size - 1).at(size - 2) = 0;
99      }
100
101     // Решение полученной СЛАУ методом Гаусса
102     std::vector<double> res = solve_with_gauss(A, b);
103     return res;
104 }
105
106 std::vector<double> build_cube_solution(size_t elems_num) {
107     double L = (X_END - X_BEGIN) / elems_num;
108     size_t size = elems_num + 1;
109     std::vector< std::vector<double> > A(size,std::vector<double>(size));
110     std::vector<double> b(size);
111
112     // Локальная матрица жесткости для кубического КЭ
113     std::vector< std::vector<double> > local_matrix = {
114         { a*37.0/(10.0*L) - C*8*L/105.0 +B*1.0/2.0, -a*189.0/(40.0*L) - C*33*L/560.0
                - B*57/80.0, a*27.0/(20.0*L) + C*3*L/140.0 + B*3.0/10.0,
                -a*13.0/(40.0*L) - C*19.0*L/1680.0 - B*7/80.0},
115         { -a*189.0/(40.0*L) - C*33*L/560.0 + B*57/80.0, a*54.0/(5.0*L)-C*27*L/70.0,
                -a*297.0/(40*L) + C*27*L/560.0 - B*81.0/80.0, a*27.0/(20.0*L) +
                C*3*L/140.0 + B*3.0/10.0},
116         { a*27.0/(20.0*L) + C*3*L/140.0 - B*3.0/10.0, -a*297.0/(40.0*L) +
                C*27*L/560.0 + B*81.0/80.0, a*54.0/(5.0*L) - C*27*L/70.0,
                -a*189.0/(40.0*L) - C*33*L/560.0 - B*57/80.0},
117         { -a*13.0/(40.0*L) - C*19.0*L/1680.0 + B*7/80.0, a*27.0/(20.0*L) +
                C*3*L/140.0 - B*3.0/10.0 , -a*189.0/(40.0*L) - C*33*L/560.0 +
                B*57/80.0, a*37.0/(10.0*L) - C*8*L/105.0 - B*1.0/2.0}
118
119     };
120
121
122     // Локальный вектор нагрузок (дополнительные слагаемые для первого и последнего
            элементов учитываются далее)
123     std::vector<double> local_b = { D * L / 8.0,
124                                     D * 3.0 * L / 8.0,
125                                     D * 3.0 * L / 8.0,
126                                     D * L / 8.0 };
127
128
129     // Производим матричные преобразования для обнуления элементов локальной
            матрицы жесткости, относящихся к внутренним узлам
130     for (size_t i = 1; i < 3; i++) {
131         for (size_t j = 0; j < 4; j++) {
132             if (std::fabs(local_matrix.at(j).at(i)) > EPS && i != j) {
```

```
133                 double val = local_matrix.at(j).at(i) /local_matrix.at(i).at(i);
134                 local_b.at(j) -= val * local_b.at(i);
135                 for (size_t k = 0; k < 4; k++) {
136                     local_matrix.at(j).at(k) -= val *local_matrix.at(i).at(k);
137                 }
138             }
139             continue;
140         }
141     }
142
143
144     // Исключаем внутренние узлы из рассмотрения
145     std::vector< std::vector<double> > local_matrix_mod = { { local_matrix.at(0).at(0),
            local_matrix.at(0).at(3) },
146                                                             { local_matrix.at(3).at(0),
                                                                 local_matrix.at(3).at(3)
                                                                 } };
147     std::vector<double> local_b_mod = { local_b.at(0),
148                                         local_b.at(3)
149                                         };
150
151     // Ансамблирование и получение глобальной матрицы жесткости для кубического КЭ
152     for (size_t i = 0; i < elems_num; i++) {
153         for (size_t j = 0; j < 2; j++) {
154             for (size_t k = 0; k < 2; k++) {
155                 A.at(i + j).at(i + k) += local_matrix_mod.at(j).at(k);
156             }
157         }
158     }
159
160     for (size_t i = 0; i < elems_num; i++) {
161         b.at(i) += local_b_mod.at(0);
162         b.at(i+1) += local_b_mod.at(1);
163     }
164
165     // Учет ГУ
166     if (0 == 1 ) {
167         b.at(0) = local_b_mod.at(0) - a * usl_left;
168     } else {
169         b.at(0) = usl_left;
170         A.at(0).at(0) = 1.;
171         A.at(0).at(1) = 0.;
172     }
173
174     if (1 == 1 ) {
175         b.at(size - 1) = local_b_mod.at(1) + a * usl_right;
```

```
176     } else {
177         b.at(size - 1) = usl_right;
178         A.at(size - 1).at(size - 1) = 1.;
179         A.at(size - 1).at(size - 2) = 0.;
180     }
181
182     // Решение полученной СЛАУ методом Гаусса
183     std::vector<double> res = solve_with_gauss(A, b);
184     return res;
185 }
186
187 double calc_abs_error(const std::vector<double>& y_real, const std::vector<double>&
        y) {
188     double max_err = 0.0;
189     for (size_t i = 0; i < y_real.size(); i++) {
190         double err = std::fabs(y_real.at(i) - y.at(i));
191         if (err > max_err) {
192             max_err = err;
193         }
194     }
195     return max_err;
196 }
197
198 int main() {
199
200      std::vector<double> x(ELEMS_NUM + 1);
201      for (size_t i = 0; i < x.size(); i++) {
202          x.at(i) = X_BEGIN + i * L;
203      }
204      size_t x_size = x.size();
205
206     std::vector<double> y;
207     if (true) {
208         y = build_linear_solution(ELEMS_NUM);
209     } else {
210         y = build_cube_solution(ELEMS_NUM);
211     }
212      std::vector<double> y_real = build_analytical_solution(x);
213
214
215      FILE* gp;
216      FILE* ab;
217      FILE* pgr;
218      FILE* tab;
219      if (true) {
220         if(ELEMS_NUM == 20) {
```

```
221             gp = fopen("res/labs/text/graph/lin_20.txt", "w");
222             ab = fopen("res/labs/text/graph/abs.txt", "w");
223             for (size_t i = 0; i < x_size; i++) {
224                 fprintf(ab, "%lf %lf\n", x.at(i), y_real.at(i));
225             }
226             pgr = fopen("res/labs/text/pgr/lin_20.txt", "w");
227             tab = fopen("res/labs/text/tab/lin_20.txt", "w");
228         }
229         if(ELEMS_NUM == 40) {
230             gp = fopen("res/labs/text/graph/lin_40.txt", "w");
231             pgr = fopen("res/labs/text/pgr/lin_40.txt", "w");
232             tab = fopen("res/labs/text/tab/lin_40.txt", "w");
233         }
234     } else {
235         if(ELEMS_NUM == 20) {
236             gp = fopen("res/labs/text/graph/cub_20.txt", "w");
237             pgr = fopen("res/labs/text/pgr/cub_20.txt", "w");
238             tab = fopen("res/labs/text/tab/cub_20.txt", "w");
239         }
240         if(ELEMS_NUM == 40) {
241             gp = fopen("res/labs/text/graph/cub_40.txt", "w");
242             pgr = fopen("res/labs/text/pgr/cub_40.txt", "w");
243             tab = fopen("res/labs/text/tab/cub_40.txt", "w");
244         }
245     }
246
247     for (size_t i = 0; i < x.size()-1; i++) {
248         fprintf(tab, "%lf & %lf & %lf & %lf \\\\\n", x.at(i), y_real.at(i), y.at(i),
                std::fabs(y_real.at(i) - y.at(i)));
249     }
250     fprintf(tab, "%lf & %lf & %lf & %lf", x.at(x.size()-1), y_real.at(x.size()-1),
            y.at(x.size()-1), std::fabs(y_real.at(x.size()-1) - y.at(x.size()-1)));
251
252     for (size_t i = 0; i < x_size; i++) {
253         fprintf(gp, "%lf %lf\n", x.at(i), y.at(i));
254     }
255
256     fprintf(pgr, "%e", calc_abs_error(y_real, y));
257     fclose(gp);
258     fclose(ab);
259     fclose(pgr);
260     fclose(tab);
261
262     return 0;
263 }
```

8 Вывод

В ходе выполнения лабораторной работы был реализован МКЭ для различных функций форм, а также найдено количество линейных КЭ обеспечивающих точность 20ти кубических КЭ.

Постановка: © *доцент кафедры РК-6, кандидат технических наук, доцент, Трудоношин В.А.*

Решение и вёрстка: © *студент группы РК6-71б, Шашко О. В.*

2023, осенний семестр