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:	Василян Артур Размикович
Группа:	РК6-73б
Тип задания:	Лабораторная работа (дополнительное задание)
Название:	Метод конечных элементов
Вариант:	45

Студент _____ Василян А.Р.
подпись, дата Фамилия, И.О.

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

Оценка: _____

Москва, 2022

Содержание

Метод конечных элементов **3**

1 Цель выполнения лабораторной работы 3
2 Задание 3
3 Аналитическое решение 4
4 Получение локальных матрицы жесткости и вектора нагрузок 4
Линейная функция-формы КЭ 5
Кубическая функция-формы КЭ 6
5 Получение глобальных матрицы жесткости и вектора нагрузок 7
Ансамблирование 7
Учет граничных условий 8
6 Анализ результатов 8
Линейная функция-формы 8
Кубическая функция-формы 11
Нахождение количества линейных КЭ, обеспечивающих ту же точность, что и 20 кубических 14
7 Код 14
8 Вывод 20

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

$$3\frac{d^2u}{dx^2} - 4\frac{du}{dx} + 11 = 0, \tag{1}$$

при следующих граничных условиях (г. у.):

$$u(x = 0) = 4, \tag{2}$$

$$u'(x = 14) = 8. \tag{3}$$

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.

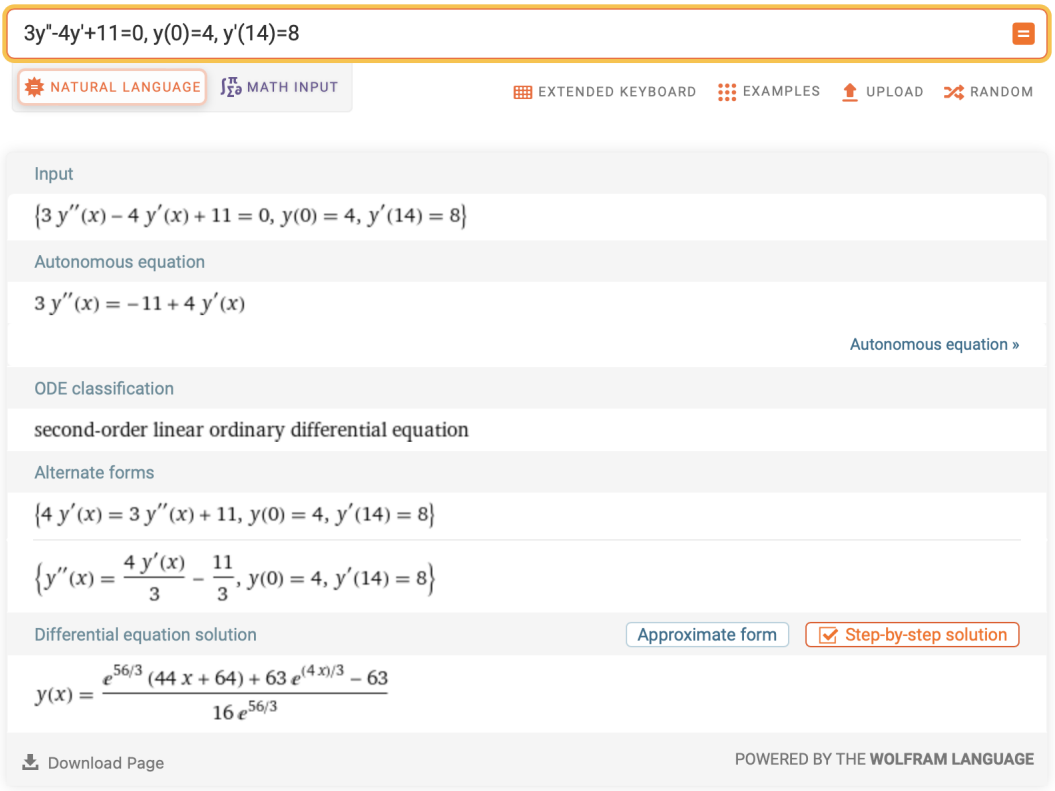

Рис. 1. Аналитическое решение

Таким образом, получаем:

$$u(x) = \frac{e^{56/3}(44x + 64) + 63e^{4x/3} - 63}{16e^{56/3}}.$$

4 Получение локальных матрицы жесткости и вектора нагрузок

Составим локальные матрицу жесткости и вектор нагрузок для уравнения 1.

Линейная функция-формы КЭ

$$\mathbf{u} = \begin{bmatrix} (1 - \frac{x}{L}); & \frac{x}{L} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} = \mathbf{N_e U},$$

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:

$$\int_0^L \mathbf{W_e}\left(3\frac{d^2\mathbf{u}}{dx^2} - 4\frac{d\mathbf{u}}{dx} + 11\right)dx = 0, \tag{4}$$

где $\mathbf{W_e} = \mathbf{N_e}^T$.

$$\int_0^L \mathbf{W_e}\left(3\frac{d^2\mathbf{u}}{dx^2} - 4\frac{du}{dx} + 11\right)dx = 3\int_0^L \mathbf{W_e}\frac{d^2\mathbf{u}}{dx^2}dx - 4\int_0^L \mathbf{W_e}\frac{d\mathbf{u}}{dx}dx + 11\int_0^L \mathbf{W_e}dx = 0$$

Распишем каждое слагаемое отдельно:

$$3\int_0^L \mathbf{W_e}\frac{d^2\mathbf{u}}{dx^2}dx = 3\int_0^L \begin{bmatrix}(1-\frac{x}{L})\\ \frac{x}{L}\end{bmatrix}\frac{d^2\mathbf{u}}{dx^2}dx = 3\begin{bmatrix}(1-\frac{x}{L})\\ \frac{x}{L}\end{bmatrix}\frac{d\mathbf{u}}{dx}|_0^L -$$

$$-3\int_0^L \frac{d}{dx}\begin{bmatrix}(1-\frac{x}{L})\\ \frac{x}{L}\end{bmatrix}\frac{d}{dx}\begin{bmatrix}(1-\frac{x}{L}); & \frac{x}{L}\end{bmatrix}\begin{bmatrix}u_i\\ u_j\end{bmatrix} = \begin{bmatrix}-3\frac{d\mathbf{u}}{dx}|_i\\ 3\frac{d\mathbf{u}}{dx}|_j\end{bmatrix} - 3\begin{bmatrix}\frac{1}{L}, & -\frac{1}{L}\\ -\frac{1}{L}, & \frac{1}{L}\end{bmatrix}\begin{bmatrix}u_i\\ u_j\end{bmatrix}$$

$$-4\int_0^L \mathbf{W_e}\frac{d\mathbf{u}}{dx}dx = -4\int_0^L \begin{bmatrix}(1-\frac{x}{L})\\ \frac{x}{L}\end{bmatrix}\frac{d\mathbf{u}}{dx}\begin{bmatrix}u_i\\ u_j\end{bmatrix}dx =$$

$$= -\frac{4}{L}\int_0^L \begin{bmatrix}(1-\frac{x}{L}) & (-1+\frac{x}{L})\\ -\frac{x}{L} & \frac{x}{L}\end{bmatrix}\begin{bmatrix}u_i\\ u_j\end{bmatrix} = -4\begin{bmatrix}-\frac{1}{2}, & \frac{1}{2}\\ -\frac{1}{2}, & \frac{1}{2}\end{bmatrix}\begin{bmatrix}u_i\\ u_j\end{bmatrix}$$

$$11\int_0^L \mathbf{W_e}dx = 11\begin{bmatrix}\frac{L}{2}\\ \frac{L}{2}\end{bmatrix}$$

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

$$\begin{bmatrix}3\frac{1}{L} - 4\frac{1}{2}, & -3\frac{1}{L} + 4\frac{1}{2}\\ -3\frac{1}{L} - 4\frac{1}{2}, & 3\frac{1}{L} + 4\frac{1}{2}\end{bmatrix}\begin{bmatrix}u_i\\ u_j\end{bmatrix} = \begin{bmatrix}-3\frac{du}{dx}|_i + 11\frac{L}{2}\\ 3\frac{du}{dx}|_j + 11\frac{L}{2}\end{bmatrix}$$

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.

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

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

$$= \begin{bmatrix} -3\frac{d\mathbf{u}}{dx}|_i \\ 0 \\ 0 \\ 3\frac{d\mathbf{u}}{dx}|_l \end{bmatrix} - 3\begin{bmatrix} \frac{37}{10L} & -\frac{189}{40} & \frac{27}{20L} & -\frac{13}{40L} \\ -\frac{189}{40} & \frac{54}{5L} & -\frac{297}{40L} & \frac{27}{20L} \\ \frac{27}{20L} & -\frac{297}{40L} & \frac{54}{5L} & -\frac{189}{40} \\ -\frac{13}{40L} & \frac{27}{20L} & -\frac{189}{40} & \frac{37}{10L} \end{bmatrix}\begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix}$$

$$-4\int_0^L \mathbf{W_e}\frac{d\mathbf{u}}{dx}dx = -4\int_0^L \begin{bmatrix} -\frac{9x^3}{2L^3} & + & \frac{18x^2}{2L^2} & - & \frac{11x}{2L} & + & 1 \\ \frac{27x^3}{2L^3} & - & \frac{45x^2}{2L^2} & + & \frac{9x}{L} & & \\ -\frac{27x^3}{2L^3} & + & \frac{36x^2}{2L^2} & - & \frac{9x}{2L} & & \\ \frac{9x^3}{2L^3} & - & \frac{9x^2}{2L^2} & - & \frac{x}{L} & & \end{bmatrix} \frac{d\mathbf{u}}{dx}dx =$$

$$= -4\begin{bmatrix} -\frac{1}{2} & \frac{57}{80} & -\frac{3}{10} & \frac{7}{80} \\ -\frac{57}{80} & 0 & \frac{81}{80} & -\frac{3}{10} \\ \frac{3}{10} & -\frac{81}{80} & 0 & -\frac{3}{10} \\ -\frac{7}{80} & \frac{3}{10} & -\frac{57}{80} & \frac{1}{2} \end{bmatrix}\begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix}$$

$$11\int_0^L \mathbf{W_e}dx = 11\begin{bmatrix} \frac{L}{8} \\ \frac{3L}{8} \\ \frac{3L}{8} \\ \frac{L}{8} \end{bmatrix}$$

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

$$\begin{bmatrix} 3\frac{37}{10L} - 4\frac{1}{2} & -3\frac{189}{40L} + 4\frac{57}{80} & 3\frac{27}{20L} - 4\frac{3}{10} & -3\frac{13}{40L} + 4\frac{7}{80} \\ -3\frac{189}{40L} - 4\frac{57}{80} & 3\frac{54}{5L} + 0 & -3\frac{297}{40L} + 4\frac{81}{80} & 3\frac{27}{20L} - 4\frac{3}{10} \\ 3\frac{27}{20L} + 4\frac{3}{10} & -3\frac{297}{40L} - 4\frac{81}{80} & 3\frac{54}{5L} + 0 & -3\frac{189}{40L} + 4\frac{57}{80} \\ -3\frac{13}{40L} - 4\frac{7}{80} & 3\frac{27}{20L} + 4\frac{3}{10} & -3\frac{189}{40L} + 4\frac{57}{80} & 3\frac{37}{10L} + 4\frac{1}{2} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix} = \begin{bmatrix} 11\frac{L}{8} - 3\frac{du}{dx}|_i \\ 11\frac{3L}{8} \\ 11\frac{3L}{8} \\ 11\frac{L}{8} + 3\frac{du}{dx}|_l \end{bmatrix} \tag{5}$$

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

$$\begin{bmatrix} a_{11} & 0 & 0 & a_{14} \\ a_{21} & a_{22} & 0 & a_{24} \\ a_{31} & 0 & a_{33} & a_{34} \\ a_{41} & 0 & 0 & a_{44} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix} = \begin{bmatrix} b_1 - 3\frac{du}{dx}|_i \\ b_2 \\ b_3 \\ b_4 + 3\frac{du}{dx}|_l \end{bmatrix}$$

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

$$\begin{bmatrix} a_{11} & a_{14} \\ a_{41} & a_{44} \end{bmatrix} \begin{bmatrix} u_i \\ u_l \end{bmatrix} = \begin{bmatrix} b_1 - 3\frac{du}{dx}|_i \\ b_4 + 3\frac{du}{dx}|_l \end{bmatrix}$$

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

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} = \begin{bmatrix} b_1 - 3\frac{du}{dx}|_i \\ b_2 + 3\frac{du}{dx}|_l \end{bmatrix},$$

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

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

Учет граничных условий

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

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ a_{21}^1 & a_{22}^1 + a_{11}^2 & a_{12}^2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_{21}^2 & a_{22}^2 + a_{11}^3 & a_{12}^3 & 0 & \cdots & 0 & 0 \\ 0 & 0 & a_{21}^3 & a_{22}^3 + \cdots & & & & 0 \\ \vdots & \vdots & \vdots & \vdots & & & & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots + a_{11}^n & a_{12}^n \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} 4 \\ b_2^1 + b_1^2 \\ b_2^2 + b_1^3 \\ b_2^3 + b_1^4 \\ \vdots \\ b_2^{n-1} + b_1^n \\ b_2^n + 3 \cdot 8 \end{bmatrix}$$

6 Анализ результатов

Проведем сравнение результатов согласно заданию.

Линейная функция-формы

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

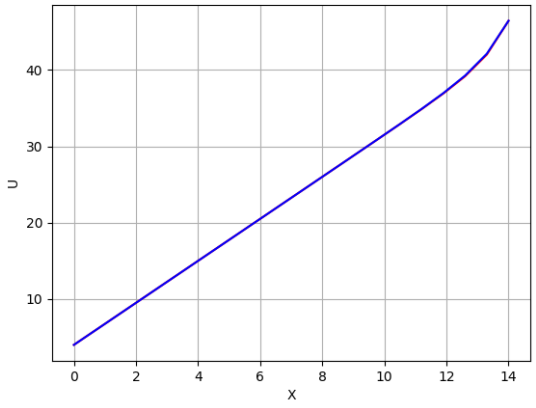

Рис. 2. Результат работы программы для 20 КЭ

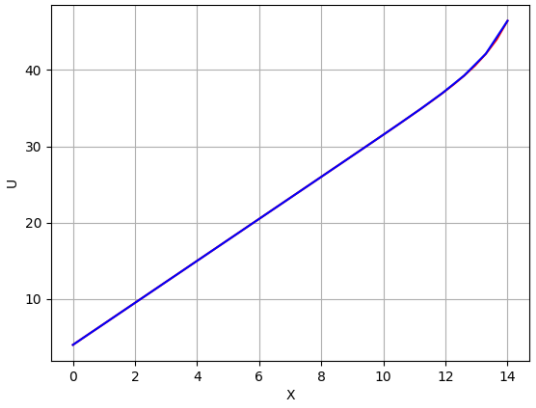

Рис. 3. Результат работы программы для 40 КЭ

X	Аналитическое решение	МКЭ-решение	Абсолютная погрешность
0.000000	4.000000	4.000000	0.000000
0.700000	5.925000	5.925000	0.000000
1.400000	7.850000	7.850000	0.000000
2.100000	9.775000	9.775000	0.000000
2.800000	11.700001	11.700000	0.000001
3.500000	13.625003	13.625001	0.000002
4.200000	15.550008	15.550003	0.000006
4.900000	17.475021	17.475008	0.000013
5.600000	19.400054	19.400021	0.000033
6.300000	21.325137	21.325058	0.000079
7.000000	23.250348	23.250159	0.000189
7.700000	25.175885	25.175438	0.000448
8.400000	27.102252	27.101204	0.001048
9.100000	29.030726	29.028310	0.002415
9.800000	30.964560	30.959104	0.005456
10.500000	32.912026	32.900036	0.011991
11.200000	34.894157	34.868848	0.025310
11.900000	36.964440	36.914331	0.050108
12.600000	39.258888	39.170661	0.088227
13.300000	42.123385	42.006818	0.116567
14.000000	46.437500	46.437500	0.000000

Таблица 1. 20 линейных КЭ

X	Аналитическое решение	МКЭ-решение	Абсолютная погрешность
0.000000	4.000000	4.000000	0.000000
0.350000	4.962500	4.962500	0.000000
0.700000	5.925000	5.925000	0.000000
1.050000	6.887500	6.887500	0.000000
1.400000	7.850000	7.850000	0.000000
1.750000	8.812500	8.812500	0.000000
2.100000	9.775000	9.775000	0.000000
2.450000	10.737501	10.737501	0.000000
2.800000	11.700001	11.700001	0.000000
3.150000	12.662502	12.662502	0.000000
3.500000	13.625003	13.625002	0.000001
3.850000	14.587505	14.587504	0.000001
4.200000	15.550008	15.550006	0.000002
4.550000	16.512513	16.512510	0.000003
4.900000	17.475021	17.475017	0.000004
5.250000	18.437534	18.437527	0.000007
5.600000	19.400054	19.400044	0.000010
5.950000	20.362586	20.362570	0.000016
6.300000	21.325137	21.325113	0.000024
6.650000	22.287718	22.287682	0.000037
7.000000	23.250348	23.250292	0.000056
7.350000	24.213055	24.212970	0.000085
7.700000	25.175885	25.175756	0.000129
8.050000	26.138912	26.138717	0.000195
8.400000	27.102252	27.101957	0.000294
8.750000	28.066091	28.065649	0.000442
9.100000	29.030726	29.030065	0.000661
9.450000	29.996631	29.995648	0.000982
9.800000	30.964560	30.963108	0.001452
10.150000	31.935719	31.933587	0.002132
10.500000	32.912026	32.908922	0.003104
10.850000	33.896545	33.892070	0.004475
11.200000	34.894157	34.887787	0.006371
11.550000	35.912650	35.903722	0.008928
11.900000	36.964440	36.952184	0.012256
12.250000	38.069327	38.052969	0.016358
12.600000	39.258888	39.237929	0.020959
12.950000	40.583476	40.558299	0.025176
13.300000	42.123385	42.096503	0.026882
13.650000	44.006663	43.985135	0.021528
14.000000	46.437500	46.437500	0.000000

Таблица 2. 40 линейных КЭ

Максимальная абсолютная погрешность 1.165671e-01 и 2.688241e-02 соответственно.

Кубическая функция-формы

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

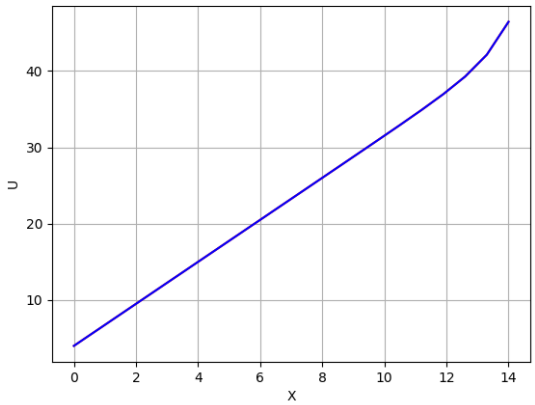

Рис. 4. Результат работы программы для 20 КЭ

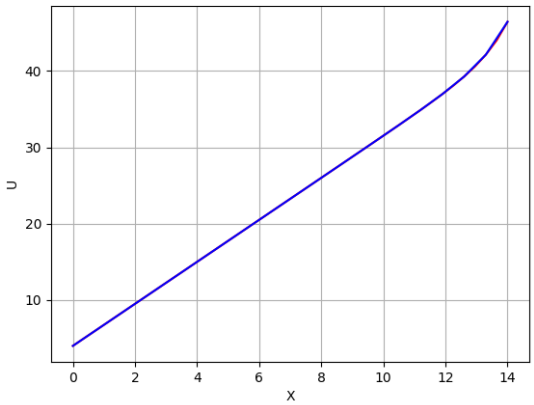

Рис. 5. Результат работы программы для 40 КЭ

X	Аналитическое решение	МКЭ-решение	Абсолютная погрешность
0.000000	4.000000	4.000000	0.000000
0.700000	5.925000	5.925000	0.000000
1.400000	7.850000	7.850000	0.000000
2.100000	9.775000	9.775000	0.000000
2.800000	11.700001	11.700001	0.000000
3.500000	13.625003	13.625003	0.000000
4.200000	15.550008	15.550008	0.000000
4.900000	17.475021	17.475021	0.000000
5.600000	19.400054	19.400054	0.000000
6.300000	21.325137	21.325137	0.000000
7.000000	23.250348	23.250348	0.000000
7.700000	25.175885	25.175885	0.000000
8.400000	27.102252	27.102251	0.000000
9.100000	29.030726	29.030725	0.000000
9.800000	30.964560	30.964560	0.000001
10.500000	32.912026	32.912025	0.000001
11.200000	34.894157	34.894155	0.000002
11.900000	36.964440	36.964435	0.000005
12.600000	39.258888	39.258880	0.000008
13.300000	42.123385	42.123376	0.000010
14.000000	46.437500	46.437500	0.000000

Таблица 3. 20 кубических КЭ

X	Аналитическое решение	МКЭ-решение	Абсолютная погрешность
0.000000	4.000000	4.000000	0.000000
0.350000	4.962500	4.962500	0.000000
0.700000	5.925000	5.925000	0.000000
1.050000	6.887500	6.887500	0.000000
1.400000	7.850000	7.850000	0.000000
1.750000	8.812500	8.812500	0.000000
2.100000	9.775000	9.775000	0.000000
2.450000	10.737501	10.737501	0.000000
2.800000	11.700001	11.700001	0.000000
3.150000	12.662502	12.662502	0.000000
3.500000	13.625003	13.625003	0.000000
3.850000	14.587505	14.587505	0.000000
4.200000	15.550008	15.550008	0.000000
4.550000	16.512513	16.512513	0.000000
4.900000	17.475021	17.475021	0.000000
5.250000	18.437534	18.437534	0.000000
5.600000	19.400054	19.400054	0.000000
5.950000	20.362586	20.362586	0.000000
6.300000	21.325137	21.325137	0.000000
6.650000	22.287718	22.287718	0.000000
7.000000	23.250348	23.250348	0.000000
7.350000	24.213055	24.213055	0.000000
7.700000	25.175885	25.175885	0.000000
8.050000	26.138912	26.138912	0.000000
8.400000	27.102252	27.102252	0.000000
8.750000	28.066091	28.066091	0.000000
9.100000	29.030726	29.030726	0.000000
9.450000	29.996631	29.996631	0.000000
9.800000	30.964560	30.964560	0.000000
10.150000	31.935719	31.935719	0.000000
10.500000	32.912026	32.912026	0.000000
10.850000	33.896545	33.896545	0.000000
11.200000	34.894157	34.894157	0.000000
11.550000	35.912650	35.912650	0.000000
11.900000	36.964440	36.964440	0.000000
12.250000	38.069327	38.069327	0.000000
12.600000	39.258888	39.258888	0.000000
12.950000	40.583476	40.583475	0.000000
13.300000	42.123385	42.123385	0.000000
13.650000	44.006663	44.006663	0.000000
14.000000	46.437500	46.437500	0.000000

Таблица 4. 40 кубических КЭ

Максимальная абсолютная погрешность 9.802044e-06 и 1.493353e-07 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.

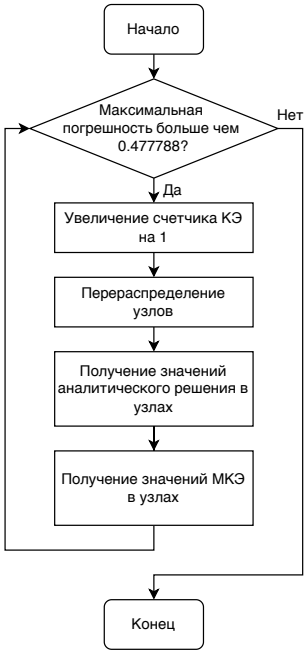

Рис. 6. Алгоритм нахождения количества КЭ, заданную точность

Реализовав данный алгоритм с начальным количеством КЭ=20 и увеличивая счетчик всегда на 1 получаем необходимое количество КЭ, равное 2073 .

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 = 0.0;
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 = 3.0, B = −4.0, C = 0.0, D = 11.0, usl_left = 4.0, usl_right = 8.0; //
       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(56./3.) * (44. * x + 64.) + 63. * exp(4 * x / 3.) − 63.)/(16. * exp(56./3.));
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) + (B / 2.), -(a / L) - (B / 2.)},
68          { -(a / L) + (B / 2.), (a / L) - (B / 2.)},
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./(10.*L) + B / 2., -a * 189./(40.*L) - B * 57./80., a * 27./(20.*L) + B
                * 3./10., -a * 13./(40.*L) - B * 7./80. },
115         { -a * 189./(40.*L)+ B * 57./80., a * 54./(5.*L) + 0., -a * 297./(40.*L) - B *
                81./80., a * 27./(20.*L) + B * 3./10. },
116         { a * 27./(20.*L) - B * 3./10., -a * 297./(40.*L) + B * 81./80., a * 54./(5.*L) -
                0., -a * 189./(40.*L) - B * 57./80.},
117         { -a * 13./(40.*L) + B * 7./80., a * 27./(20.*L) - B * 3./10., -a * 189./(40.*L)
                + B * 57./80., a * 37./(10.*L) - B * 1./2.}
118     };
119
120     // Локальный вектор нагрузок (дополнительные слагаемые для первого и последнего
            элементов учитываются далее)
121     std::vector<double> local_b = { D * L / 8.0,
122                                     D * 3.0 * L / 8.0,
123                                     D * 3.0 * L / 8.0,
124                                     D * L / 8.0 };
125
126
127     // Производим матричные преобразования для обнуления элементов локальной
            матрицы жесткости, относящихся к внутренним узлам
128     for (size_t i = 1; i < 3; i++) {
129         for (size_t j = 0; j < 4; j++) {
130             if (std::fabs(local_matrix.at(j).at(i)) > EPS && i != j) {
131                 double val = local_matrix.at(j).at(i) /local_matrix.at(i).at(i);
132                 local_b.at(j) -= val * local_b.at(i);
133                 for (size_t k = 0; k < 4; k++) {
134                     local_matrix.at(j).at(k) -= val *local_matrix.at(i).at(k);
135                 }
136             }
137             continue;
138         }
139     }
```

```
140
141
142     // Исключаем внутренние узлы из рассмотрения
143     std::vector< std::vector<double> > local_matrix_mod = { { local_matrix.at(0).at(0),
            local_matrix.at(0).at(3) },
144                                                             { local_matrix.at(3).at(0),
                                                                    local_matrix.at(3).at(3)
                                                                    } };
145     std::vector<double> local_b_mod = { local_b.at(0),
146                                         local_b.at(3)
147                                         };
148
149     // Ансамблирование и получение глобальной матрицы жесткости для кубического КЭ
150     for (size_t i = 0; i < elems_num; i++) {
151         for (size_t j = 0; j < 2; j++) {
152             for (size_t k = 0; k < 2; k++) {
153                 A.at(i + j).at(i + k) += local_matrix_mod.at(j).at(k);
154             }
155         }
156     }
157
158     for (size_t i = 0; i < elems_num; i++) {
159         b.at(i) += local_b_mod.at(0);
160         b.at(i+1) += local_b_mod.at(1);
161     }
162
163     // Учет ГУ
164     if (0 == 1 ) {
165         b.at(0) = local_b_mod.at(0) - a * usl_left;
166     } else {
167         b.at(0) = usl_left;
168         A.at(0).at(0) = 1.;
169         A.at(0).at(1) = 0.;
170     }
171
172     if (1 == 1 ) {
173         b.at(size - 1) = local_b_mod.at(1) + a * usl_right;
174     } else {
175         b.at(size - 1) = usl_right;
176         A.at(size - 1).at(size - 1) = 1.;
177         A.at(size - 1).at(size - 2) = 0.;
178     }
179
180     // Решение полученной СЛАУ методом Гаусса
181     std::vector<double> res = solve_with_gauss(A, b);
182     return res;
```

```
183 }
184
185 double calc_abs_error(const std::vector<double>& y_real, const std::vector<double>&
        y) {
186     double max_err = 0.0;
187     for (size_t i = 0; i < y_real.size(); i++) {
188         double err = std::fabs(y_real.at(i) - y.at(i));
189         if (err > max_err) {
190             max_err = err;
191         }
192     }
193     return max_err;
194 }
195
196 int main() {
197
198      std::vector<double> x(ELEMS_NUM + 1);
199      for (size_t i = 0; i < x.size(); i++) {
200          x.at(i) = X_BEGIN + i * L;
201      }
202      size_t x_size = x.size();
203
204     std::vector<double> y;
205     if (true) {
206         y = build_linear_solution(ELEMS_NUM);
207     } else {
208         y = build_cube_solution(ELEMS_NUM);
209     }
210      std::vector<double> y_real = build_analytical_solution(x);
211
212
213      FILE* gp;
214      FILE* ab;
215      FILE* pgr;
216      FILE* tab;
217      if (true) {
218         if(ELEMS_NUM == 20) {
219             gp = fopen("res/labs/text/graph/lin_20.txt", "w");
220             ab = fopen("res/labs/text/graph/abs.txt", "w");
221             for (size_t i = 0; i < x_size; i++) {
222                 fprintf(ab, "%lf %lf\n", x.at(i), y_real.at(i));
223             }
224             pgr = fopen("res/labs/text/pgr/lin_20.txt", "w");
225             tab = fopen("res/labs/text/tab/lin_20.txt", "w");
226         }
227         if(ELEMS_NUM == 40) {
```

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

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 РК6-73б, Василян А.Р.*

2022, осенний семестр