Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего 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	Мудриченко Михаил Николаевич
Группа	РК6-72Б
Тема лабораторной работы	Метод конечных элементов
Вариант	35

Студент _____ **Мудриченко М.Н.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Трудоношин В.А.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2022 г.

Оглавление

Задание

Методом конечных элементов решить уравнение:

$$5\frac{d^2u}{dx^2} + 2\frac{du}{dx} - 5 = 0 \qquad (1)$$

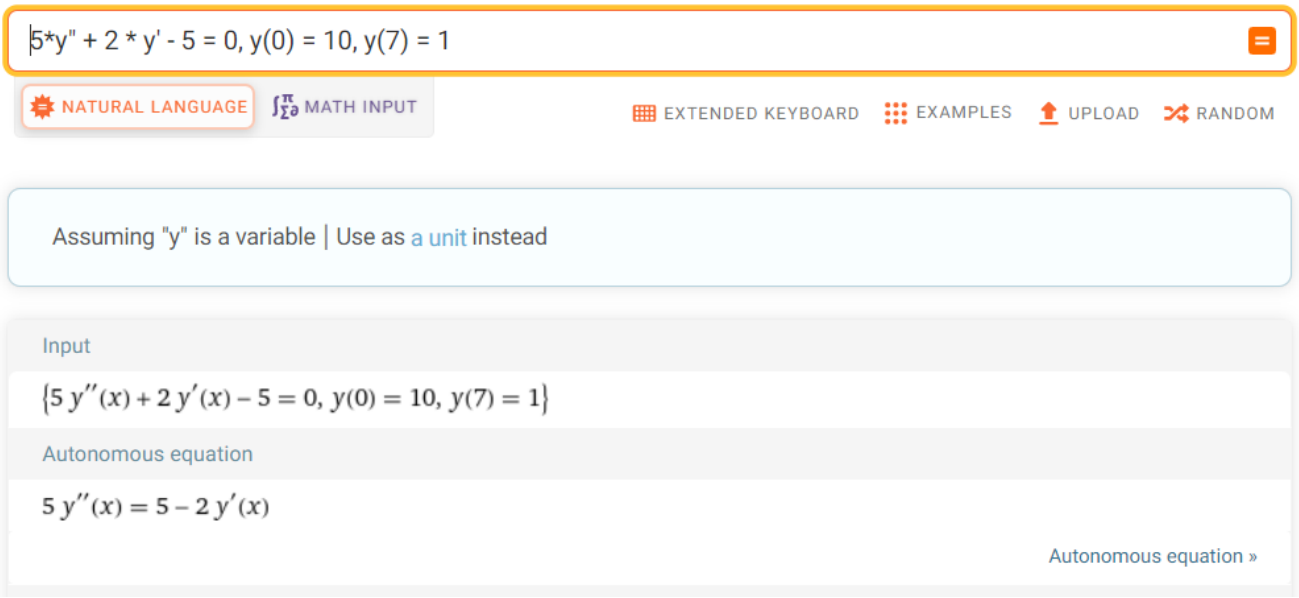
при следующих граничных условиях:

$$u(0) = 10 \quad u(7) = 1$$

Количество конечных элементов для первого расчета — 20, количество конечных элементов для второго расчета — 40.

Сравнить результат с аналитическим решением, оценить максимальную погрешность.

Аналитическое решение



Alternate forms

$$\left\{y''(x) = 1 - \frac{2\,y'(x)}{5},\ y(0) = 10,\ y(7) = 1\right\}$$

$$\left\{5\,y''(x) + 2\,y'(x) = 5,\ y(0) = 10,\ y(7) = 1\right\}$$

Differential equation solution

$$y(x) = \frac{e^{14/5}\,(33 - 5\,x) - 53\,e^{-2/5\,(x-7)} + 5\,(x+4)}{2 - 2\,e^{14/5}}$$

Локальная матрица жесткости и вектор нагрузок для линейной функции формы

Пусть длина конечного элемента (КЭ) равна L, u внутри КЭ изменяется линейно, а значения в узлах равны $u_i(x = 0)$ и $u_j(x = L)$, тогда с помощью линейной интерполяции получим:

$$u = \left(1 - \frac{x}{L}\right) u_i + \frac{x}{L} u_j = \boldsymbol{N_e u},$$

где $\boldsymbol{N_e}$ — вектор функции формы.

Получим локальную матрицу жесткости с помощью метода Галеркина:

$$\int_0^L \boldsymbol{N_e}^T \left(5\frac{d^2u}{dx^2} + 2\frac{du}{dx} - 5\right) dx = \int_0^L \begin{bmatrix} 1 - \frac{x}{L} \\ \frac{x}{L} \end{bmatrix} \left(5\frac{d^2u}{dx^2} + 2\frac{du}{dx} - 5\right) dx$$

$$= 5 \begin{bmatrix} -\frac{du}{dx}\big|_i \\ \frac{du}{dx}\big|_j \end{bmatrix} - 5 \begin{bmatrix} \frac{1}{L} & -\frac{1}{L} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} + 2 \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} - 5 \begin{bmatrix} \frac{L}{2} \\ \frac{L}{2} \end{bmatrix} = 0$$

$$\begin{bmatrix} -1 - \frac{5}{L} & 1 + \frac{5}{L} \\ -1 + \frac{5}{L} & 1 - \frac{5}{L} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} = \begin{bmatrix} \frac{5L}{2} + 5\frac{du}{dx}\big|_i \\ \frac{5L}{2} - 5\frac{du}{dx}\big|_j \end{bmatrix}$$

Локальная матрица жесткости и вектор нагрузок для кубической функции формы

Для получения функции формы воспользуемся интерполяцией Лагранжа, проходящей через узлы $u_i(x=0)$, $u_j(x=\frac{L}{3})$, $u_k(x=\frac{2L}{3})$, $u_l(x=L)$. В результате интерполяции получим следующую функцию формы:

$$u=\left(-\frac{9x^3}{2L^3}+\frac{9x^2}{L^2}-\frac{11x}{2L}+1\right)u_i+\left(\frac{27x^3}{2L^3}-\frac{45x^2}{2L^2}+\frac{9x}{L}\right)u_j+$$

$$+\left(-\frac{27x^3}{2L^3}+\frac{18x^2}{L^2}-\frac{9x}{2L}\right)u_k+\left(\frac{9x^3}{2L^3}-\frac{9x^2}{2L^2}+\frac{x}{L}\right)u_l=\boldsymbol{N_e u}$$

Получим локальную матрицу жесткости с помощью метода Галеркина:

$$\int_0^L \boldsymbol{N_e}^T\left(5\frac{d^2u}{dx^2}+2\frac{du}{dx}-5\right)dx=5\int_0^L \boldsymbol{N_e}^T\frac{d^2u}{dx^2}dx+2\int_0^L \boldsymbol{N_e}^T\frac{du}{dx}dx-5\int_0^L \boldsymbol{N_e}^T dx$$

$$\int_0^L \boldsymbol{N_e}^T\frac{d^2u}{dx^2}dx=\int_0^L\begin{bmatrix}-\frac{9x^3}{2L^3}+\frac{9x^2}{L^2}-\frac{11x}{2L}+1\\ \frac{27x^3}{2L^3}-\frac{45x^2}{2L^2}+\frac{9x}{L}\\ -\frac{27x^3}{2L^3}+\frac{18x^2}{L^2}-\frac{9x}{2L}\\ \frac{9x^3}{2L^3}-\frac{9x^2}{2L^2}+\frac{x}{L}\end{bmatrix}\frac{d^2u}{dx^2}dx=$$

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

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

$$\int_0^L \boldsymbol{N_e}^T \frac{du}{dx} dx = \int_0^L \begin{bmatrix} -\frac{9x^3}{2L^3} + \frac{9x^2}{L^2} - \frac{11x}{2L} + 1 \\ \frac{27x^3}{2L^3} - \frac{45x^2}{2L^2} + \frac{9x}{L} \\ -\frac{27x^3}{2L^3} + \frac{18x^2}{L^2} - \frac{9x}{2L} \\ \frac{9x^3}{2L^3} - \frac{9x^2}{2L^2} + \frac{x}{L} \end{bmatrix} \frac{du}{dx} dx =$$

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

$$\int_0^L \boldsymbol{N_e}^T = \int_0^L \begin{bmatrix} -\frac{9x^3}{2L^3} + \frac{9x^2}{L^2} - \frac{11x}{2L} + 1 \\ \frac{27x^3}{2L^3} - \frac{45x^2}{2L^2} + \frac{9x}{L} \\ -\frac{27x^3}{2L^3} + \frac{18x^2}{L^2} - \frac{9x}{2L} \\ \frac{9x^3}{2L^3} - \frac{9x^2}{2L^2} + \frac{x}{L} \end{bmatrix} dx = \begin{bmatrix} \frac{L}{8} \\ \frac{3L}{8} \\ \frac{3L}{8} \\ \frac{L}{8} \end{bmatrix}$$

Итоговое значение интеграла:

$$\int_0^L \boldsymbol{N_e}^T \left(5\frac{d^2u}{dx^2} + 2\frac{du}{dx} - 5 \right) dx = 5 \begin{bmatrix} -\frac{du}{dx}|_i \\ 0 \\ 0 \\ \frac{du}{dx}|_l \end{bmatrix} - 5 \begin{bmatrix} \frac{37}{10L} & -\frac{189}{40L} & \frac{27}{20L} & -\frac{13}{40L} \\ -\frac{189}{40L} & \frac{54}{5L} & -\frac{297}{40L} & \frac{27}{20L} \\ \frac{27}{20L} & -\frac{297}{40L} & \frac{54}{5L} & -\frac{189}{40L} \\ -\frac{13}{40L} & \frac{27}{20L} & -\frac{189}{40L} & \frac{37}{10L} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix}$$

$$+2 \begin{bmatrix} -\frac{1}{2} & \frac{57}{80} & -\frac{3}{10} & \frac{7}{80} \\ -\frac{57}{80} & 0 & \frac{81}{80} & -\frac{3}{10} \\ \frac{3}{10} & -\frac{81}{80} & 0 & \frac{57}{80} \\ -\frac{7}{80} & \frac{3}{10} & -\frac{57}{80} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix} - 5 \begin{bmatrix} \frac{L}{8} \\ \frac{3L}{8} \\ \frac{3L}{8} \\ \frac{L}{8} \end{bmatrix} = 0$$

$$\begin{bmatrix} -\frac{37}{2L}-1 & \frac{189}{8L}+\frac{57}{40} & -\frac{27}{4L}-\frac{3}{5} & \frac{13}{8L}+\frac{7}{40} \\ \frac{189}{8L}-\frac{57}{40} & -\frac{54}{L} & \frac{297}{8L}+\frac{81}{40} & -\frac{27}{4L}-\frac{3}{5} \\ -\frac{27}{4L}+\frac{3}{5} & \frac{297}{8L}-\frac{81}{40} & -\frac{54}{L} & \frac{189}{8L}+\frac{57}{40} \\ \frac{13}{8L}-\frac{7}{40} & -\frac{27}{4L}+\frac{3}{5} & \frac{189}{8L}-\frac{57}{40} & -\frac{37}{2L}+1 \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix} = \begin{bmatrix} \frac{5L}{8}+5\frac{du}{dx}|_i \\ \frac{15L}{8} \\ \frac{15L}{8} \\ \frac{5L}{8}-5\frac{du}{dx}|_l \end{bmatrix}$$

Так как нас интересуют только крайние узлы, преобразуем систему к виду

$$\begin{bmatrix} \tilde{a}_{11} & 0 & 0 & \tilde{a}_{14} \\ \tilde{a}_{21} & \tilde{a}_{22} & 0 & \tilde{a}_{24} \\ \tilde{a}_{31} & 0 & \tilde{a}_{33} & \tilde{a}_{34} \\ \tilde{a}_{41} & 0 & 0 & \tilde{a}_{44} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_k \\ u_l \end{bmatrix} = \begin{bmatrix} \tilde{b}_1+5\frac{du}{dx}|_i \\ \tilde{b}_2 \\ \tilde{b}_3 \\ \tilde{b}_4-5\frac{du}{dx}|_l \end{bmatrix}$$

После чего можно перейти к системе из двух уравнений:

$$\begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{14} \\ \tilde{a}_{41} & \tilde{a}_{44} \end{bmatrix} \begin{bmatrix} u_i \\ u_l \end{bmatrix} = \begin{bmatrix} \tilde{b}_1+5\frac{du}{dx}|_i \\ \tilde{b}_4-5\frac{du}{dx}|_l \end{bmatrix}$$

Ансамблирование

Пусть локальные матрицы жесткости и вектора нагрузок имеют вид

$$\begin{bmatrix} a_{11} & a_{11} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} u_i \\ u_l \end{bmatrix} = \begin{bmatrix} b_1+5\frac{du}{dx}|_i \\ b_2-5\frac{du}{dx}|_l \end{bmatrix}$$

Тогда после ансамблирования будет получена следующая СЛАУ:

$$\begin{bmatrix} a_{11} & a_{12} & 0 & \dots & 0 \\ a_{21} & a_{22}+a_{11} & a_{12} & \dots & 0 \\ 0 & a_{21} & a_{22}+a_{11} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & a_{12} \\ 0 & 0 & 0 & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} b_1+5\frac{du}{dx}|_1 \\ b_2+b_1 \\ \vdots \\ b_2+b_1 \\ b_2-5\frac{du}{dx}|_n \end{bmatrix}$$

По условию u_1 и u_n известны, что позволяет переписать СЛАУ:

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ a_{21} & a_{22}+a_{11} & a_{12} & \dots & 0 \\ 0 & a_{21} & a_{22}+a_{11} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & a_{12} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \begin{bmatrix} 10 \\ b_2+b_1 \\ \vdots \\ b_2+b_1 \\ 1 \end{bmatrix}$$

Результат работы программы

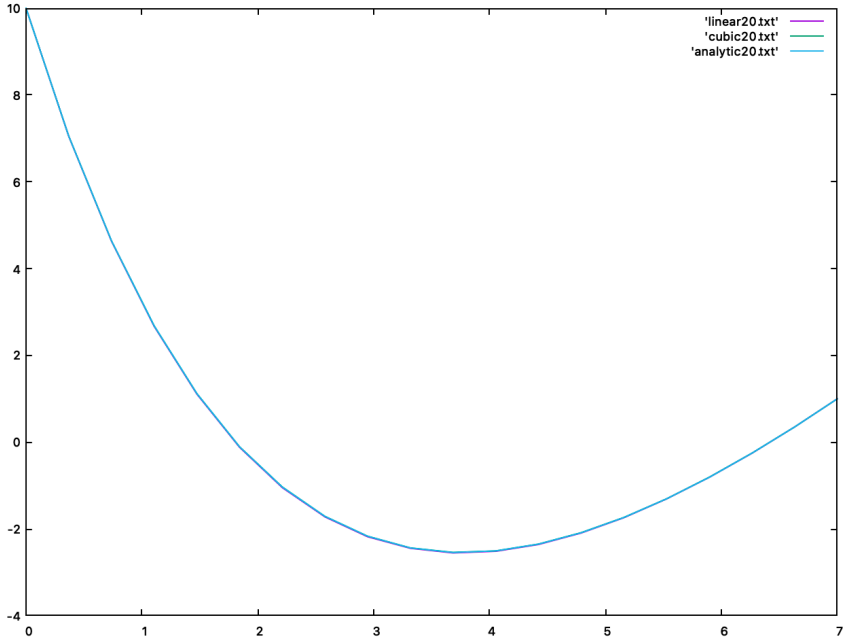


Рисунок 1 — Результат работы программы для 20 КЭ.

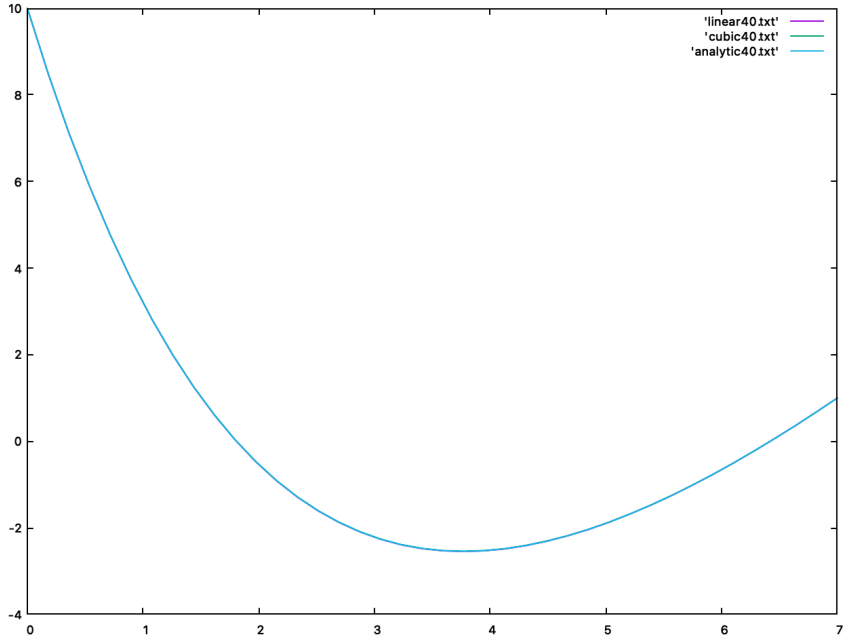


Рисунок 2 — Результат работы программы для 40 КЭ.

Данные для 20 узлов

Аналитический	Линейный	Кубический	Ошибка линейный	Ошибка кубический
10	10	10	1.42109e-14	1.42109e-14
7.05481	7.04957	7.05481	0.00524337	-7.89904e-09
4.63939	4.63051	4.63939	0.00887442	3.80418e-09
2.68114	2.66991	2.68114	0.0112367	2.64254e-08
1.11743	1.10482	1.11743	0.0126101	5.35854e-08
-0.105809	-0.119031	-0.105809	0.0132212	8.07419e-08
-1.03523	-1.04848	-1.03523	0.0132534	1.04805e-07
-1.71109	-1.72394	-1.71109	0.0128539	1.23826e-07
-2.16814	-2.18028	-2.16814	0.0121405	1.36741e-07
-2.43635	-2.44756	-2.43635	0.0112069	1.43165e-07
-2.5416	-2.55173	-2.5416	0.0101268	1.43228e-07
-2.50623	-2.51519	-2.50623	0.00895808	1.37438e-07
-2.3495	-2.35724	-2.3495	0.00774524	1.26579e-07
-2.08803	-2.09456	-2.08803	0.0065224	1.11621e-07
-1.73619	-1.74151	-1.73619	0.00531511	9.3657e-08
-1.30635	-1.31049	-1.30635	0.00414212	7.38524e-08
-0.809206	-0.812222	-0.809206	0.0030167	5.34016e-08
-0.253974	-0.255922	-0.253974	0.00194789	3.35002e-08
0.351384	0.350443	0.351384	0.000941308	1.53214e-08
1	1	1	1.15119e-15	1.15119e-15

Данные для 40 узлов

Аналитический	Линейный	Кубический	Ошибка линейный	Ошибка кубический
10	10	10	1.42109e-14	1.42109e-14
8.49398	8.49332	8.49398	0.000658178	-2.06437e-08
7.12338	7.12216	7.12338	0.00121459	-1.81847e-08
5.87881	5.87713	5.87881	0.00168018	2.36391e-09
4.75156	4.74949	4.75156	0.00206487	3.66476e-08
3.73348	3.73111	3.73348	0.00237762	8.09033e-08
2.81703	2.8144	2.81703	0.00262657	1.31899e-07
1.99515	1.99233	1.99515	0.00281905	1.86876e-07
1.26129	1.25833	1.26129	0.00296169	2.43503e-07
0.609361	0.606301	0.609361	0.00306048	2.99825e-07
0.0336827	0.0305619	0.0336824	0.0031208	3.54227e-07
-0.471027	-0.474175	-0.471028	0.00314752	4.05394e-07
-0.909686	-0.912831	-0.909686	0.00314501	4.52275e-07
-1.28687	-1.28999	-1.28687	0.00311717	4.94058e-07
-1.60683	-1.6099	-1.60683	0.00306754	5.30138e-07
-1.87355	-1.87655	-1.87355	0.00299928	5.60094e-07
-2.09069	-2.09361	-2.0907	0.00291519	5.83664e-07
-2.26171	-2.26453	-2.26171	0.0028178	6.00727e-07
-2.3898	-2.39251	-2.3898	0.00270937	6.11287e-07
-2.47792	-2.48051	-2.47792	0.00259188	6.15451e-07

-2.52886	-2.53132	-2.52886	0.00246711	6.13417e-07
-2.54517	-2.54751	-2.54517	0.00233664	6.05464e-07
-2.52928	-2.53148	-2.52928	0.00220185	5.91935e-07
-2.48339	-2.48546	-2.48339	0.00206397	5.73229e-07
-2.4096	-2.41152	-2.4096	0.00192408	5.49792e-07
-2.30984	-2.31162	-2.30984	0.00178311	5.22108e-07
-2.1859	-2.18754	-2.1859	0.00164189	4.90691e-07
-2.03945	-2.04096	-2.03946	0.00150113	4.5608e-07
-1.87207	-1.87343	-1.87207	0.00136144	4.18832e-07
-1.6852	-1.68642	-1.6852	0.00122335	3.79517e-07
-1.48019	-1.48128	-1.48019	0.00108731	3.38714e-07
-1.25829	-1.25925	-1.25829	0.000953699	2.97005e-07
-1.02068	-1.02151	-1.02068	0.000822833	2.54977e-07
-0.768449	-0.769144	-0.768449	0.000694972	2.13211e-07
-0.502602	-0.503172	-0.502602	0.00057033	1.72287e-07
-0.224086	-0.224535	-0.224086	0.000449073	1.32777e-07
0.0662207	0.0658894	0.0662206	0.000331331	9.52453e-08
0.367502	0.367285	0.367502	0.000217199	6.02453e-08
0.678998	0.678891	0.678998	0.000106743	2.832e-08
1	1	1	1.15119e-15	1.15119e-15

Одинаковая погрешность для кубической формы при 20 узлах и линейной достигается при ~6000 узлах.

Код программы

```
#include <math.h>
#include <vector>
#include <string>
#include <cassert>

typedef std::vector<std::vector<double>> Matrix;
typedef std::vector<double> Vector;

Matrix create_matrix(int rows, int cols) {
    Matrix res;
    res.resize(rows);
```

```
    for (int i = 0; i < rows; ++i) {
        res[i].resize(cols, 0);
    }
    return res;
}

Vector create_vector(int size) {
    Vector res;
    res.resize(size, 0);
    return res;
}

int gauss(Matrix& m, Vector& b) {
    for (int k = 0; k < m.size(); ++k) {
        if (fabs(m[k][k]) < 1e-17) {
            return 1;
        }
        double diagonal = m[k][k];
        for (int i = k; i < m.size(); ++i) {
            m[k][i] /= diagonal;
        }
        b[k] /= diagonal;

        for (int i = k+1; i < m.size(); ++i) {
            double elem = m[i][k];
            for (int j = k; j < m.size(); ++j) {
                m[i][j] -= elem * m[k][j];
            }
            b[i] -= elem * b[k];
        }
    }
    for (int i = m.size()-2; i >= 0; --i) {
        for (int j = i + 1; j < m.size(); ++j) {
            b[i] -= m[i][j] * b[j];
        }
    }
    return 0;
}

Vector solve(const Matrix& local_matrix, const Vector& local_vector, int n) {
    auto matrix = create_matrix(n+1, n+1);
    for (int i = 0; i < n; ++i) {
```

```
        matrix[i][i] += local_matrix[0][0];
        matrix[i][i+1] += local_matrix[0][1];
        matrix[i+1][i] += local_matrix[1][0];
        matrix[i+1][i+1] += local_matrix[1][1];
    }

    auto vector = create_vector(n+1);
    for (int i = 0; i < n; ++i) {
        vector[i] += local_vector[0];
        vector[i+1] += local_vector[1];
    }
    vector[0] = 10;
    matrix[0][0] = 1;
    matrix[0][1] = 0;
    vector[n] = 1;
    matrix[n][n] = 1;
    matrix[n][n - 1] = 0;

    if (gauss(matrix, vector) != 0) {
        puts("Error: degraded matrix");
        exit(-1);
    }
    return vector;
}

void save(const Vector& vector, const std::string& name) {
    FILE *f = fopen(name.c_str(), "w");
    assert(f);
    double L = 7.0 / (vector.size() - 1);
    for (int i = 0; i < vector.size(); ++i) {
        fprintf(f, "%lf %lf\n", -1.0 + i * L, vector[i]);
    }
    fclose(f);
}

void compress(Matrix& matrix, Vector& vector) {
    for (int i = 1; i < 3; ++i) {
        for (int j = 0; j < 4; ++j) {
            if (fabs(matrix[j][i]) < 1e-10 || i == j) {
                continue;
            }
            double val = matrix[j][i]/matrix[i][i];
```

```
            vector[j] -= val * vector[i];
            for (int k = 0; k < 4; ++k) {
                matrix[j][k] -= val * matrix[i][k];
            }
        }
    }
}

double max_different(const Vector& a, const Vector& b) {
    double max_dif = a[0] - b[0];
    for (int i = 1; i < a.size(); ++i) {
        auto d = fabs(a[i] - b[i]);
        if (d > max_dif) {
            max_dif = d;
        }
    }
    return max_dif;
}

Vector analytical(int node_count) {
    Vector result;
    result.resize(node_count);
   double l = 7.0 / (node_count - 1);
    for (int i = 0; i < node_count; ++i) {
        double x = l * i;
   //double q = 0.0000001;
        result[i] = (exp(14./5) * (33. - 5 * x) - 53 * exp(-2. / 5 * (x - 7.)) + 5. * (x + 4)) / (2. - 2 * exp(14./5));
    }
    return result;
  //  double t = exp(14.0 / 5.0);
     //   double L = 7.0 / (node_count - 1);
     //   for (int i = 0; i < node_count; ++i) {
      //          double x = L * i;
       //         double a = (20.0 + 33.0 * t) / (2.0 - 2.0 * t);
      //          double b = 53.0 * t / (2.0 * t - 2.0);
      //         result[i] = a + b * exp(-0.4 * x) + 2.5 * x;
      //  }
      //  return result;
}

Vector linear(int node_count) {
    double L = 7.0 / (node_count-1);
```

```
    Matrix local_matrix = {
        {
          -1 - 5 / L, 1 + 5 / L
        },
        {
        -1 + 5 / L,
                1 - 5 / L
        },
    };
    Vector local_vector = {
        5. * L / 2.,
        5. * L / 2.
    };
    return solve(local_matrix, local_vector, node_count - 1);
}

Vector cubic(int node_count) {
    double L = 7.0 / (node_count-1);
          Matrix temp_matrix = {
           {
                -37.0 / (2 * L) - 1.0, 189.0 / (8 * L) + 57.0 / 40, -27.0 / (4 * L) - 3.0 / 5, 13.0 / (8 * L) + 7.0 / 40
           },
           {
                189.0000001 / (8 * L) - 57.0 / 40,
                -54.0 / L,
                297.0 / (8 * L) + 81.0 / 40,
                -27.0 / (4 * L) - 3.0 / 5
           },
           {
                -27.0 / (4 * L) + 3.0 / 5,
                297.0 / (8 * L) - 81.0 / 40,
                -54.0 / L,
                189 / (8 * L) + 57.0 / 40
           },
           {
                13.0 / (8 * L) - 7.0 / 40,
                -27.0 / (4 * L) + 3.0 / 5,
                189.0 / (8 * L) - 57.0 / 40,
                -37.0 / (2 * L) + 1
           }
      };
    Vector temp_vector = {
```

```
            5. * L / 8.0,
            5. * 3.0 * L / 8.0,
            5. * 3.0 * L / 8.0,
            5. * L / 8.0,
        };
        compress(temp_matrix, temp_vector);
        Matrix local_matrix = {
            {temp_matrix[0][0], temp_matrix[0][3]},
            {temp_matrix[3][0], temp_matrix[3][3]}
        };
        Vector local_vector = {
            temp_vector[0],
            temp_vector[3]
        };
        return solve(local_matrix, local_vector, node_count - 1);
}

int main() {
    int node_count_1 = 20;
    int node_count_2 = 40;

    auto analytic20 = analytical(node_count_1);
    save(analytic20, "analytic20.txt");

    auto analytic40 = analytical(node_count_2);
    save(analytic40, "analytic40.txt");

    auto linear20 = linear(node_count_1);
    save(linear20, "linear20.txt");

    auto linear40 = linear(node_count_2);
    save(linear40, "linear40.txt");

    auto cubic20 = cubic(node_count_1);
    save(cubic20, "cubic20.txt");

    auto cubic40 = cubic(node_count_2);
    save(cubic40, "cubic40.txt");

    printf("analytic20/linear20/cubic20/error_linear20/error_cubic20\n");
    for (int i = 0; i < analytic20.size(); ++i) {
        printf("%g ", analytic20[i]);
```

```
        printf("%g ", linear20[i]);
        printf("%g ", cubic20[i]);
        printf("%g ",(double)  (analytic20[i] - linear20[i]));
        printf("%g\n", analytic20[i] - cubic20[i]);
    }

    printf("\nanalytic40/linear40/cubic40/error_linear40/error_cubic40\n");
    for (int i = 0; i < analytic40.size(); ++i) {
        printf("%g ", analytic40[i]);
        printf("%g ", linear40[i]);
        printf("%g ", cubic40[i]);
        printf("%g ", analytic40[i] - linear40[i]);
        printf("%g\n", analytic40[i] - cubic40[i]);
    }
    printf("\n");

    int node_count = 6000;
    auto linearn = linear(node_count);
    save(linearn, "linearn.txt");
    auto analyticn = analytical(node_count);
    save(analyticn, "linearn.txt");
    printf("Error linear (%d nodes): %g\n", node_count, max_different(analyticn, linearn));
    printf("Error cubic (20 nodes): %g\n", max_different(analytic20, cubic20));
    return 0;
}
```