УДК 519.692

ПРОСТЕЙШИЙ МЕТОД РЕШЕНИЯ ЖЁСТКИХ КРАЕВЫХ ЗАДАЧ

© *Ю.И.Виноградов, А.Ю.Виноградов*

МГТУ им. Н.Э.Баумана, Москва, AlexeiVinogradov@yandex.ru

Предлагается простейший метод решения жёстких краевых задач. Не требуются процедуры ортонормирования, что достигается за счёт разделения интервала интегрирования на сопрягаемые участки.

THE SIMPLEST METHOD OF SOLUTION OF THE STIFF BOUNDARY VALUE PROBLEMS

Yu.I.Vinogradov, A.Yu.Vinogradov, MSTU n.a. Bauman, Moscow

The simplest method of solution of the stiff boundary value problems is offered. The method does not need orthonormalization because of dividing an integration interval into matching sectors.

1. Введение.

Изложение даётся на примере системы дифференциальных уравнений цилиндрической оболочки ракеты – системы обыкновенных дифференциальных уравнений 8-го порядка (после разделения частных производных методом Фурье).

Система линейных обыкновенных дифференциальных уравнений имеет вид:

$$\boldsymbol{Y}'(x) = A\boldsymbol{Y}(x) + \boldsymbol{F}(x),$$

где $\boldsymbol{Y}(x)$ – искомая вектор-функция задачи размерности 8x1, $\boldsymbol{Y}'(x)$ – производная искомой вектор-функции размерности 8x1, A – квадратная матрица коэффициентов дифференциального уравнения размерности 8x8, $\boldsymbol{F}(x)$ – вектор-функция внешнего воздействия на систему размерности 8x1.

Краевые условия имеют вид:

$$U\boldsymbol{Y}(0) = \boldsymbol{u},$$
$$V\boldsymbol{Y}(1) = \boldsymbol{v},$$

где $\boldsymbol{Y}(0)$ – значение искомой вектор-функции на левом крае x=0 размерности 8x1, U – прямоугольная горизонтальная матрица коэффициентов краевых условий левого края размерности 4x8, $\boldsymbol{u}$ – вектор внешних воздействий на левый край размерности 4x1,

$\boldsymbol{Y}(1)$ – значение искомой вектор-функции на правом крае x=1 размерности 8x1, V – прямоугольная горизонтальная матрица коэффициентов краевых условий правого края размерности 4x8, $\boldsymbol{v}$ – вектор внешних воздействий на правый край размерности 4x1.

В случае, когда система дифференциальных уравнений имеет матрицу с постоянными коэффициентами A=const, решение задачи Коши имеет вид [1]:

$$\boldsymbol{Y}(x) = e^{A(x-x0)}\boldsymbol{Y}(x_0) + e^{Ax}\int_{x0}^{x} e^{-At}\boldsymbol{F}(t)dt,$$

где $e^{A(x-x0)} = E + A(x-x_0) + A^2(x-x_0)^2/2! + A^3(x-x_0)^3/3! + \ldots$, где E - это единичная матрица.

Матричная экспонента ещё может называться матрицей Коши или матрициантом и может обозначаться в виде:

$$K(x \leftarrow x_0) = K(x-x_0) = e^{A(x-x0)}.$$

Тогда решение задачи Коши может быть записано в виде:

$$\boldsymbol{Y}(x) = K(x \leftarrow x_0)\boldsymbol{Y}(x_0) + \boldsymbol{Y}^*(x \leftarrow x_0),$$

где $\boldsymbol{Y}^*(x \leftarrow x_0) = e^{Ax}\int_{x0}^{x} e^{-At}\boldsymbol{F}(t)dt$ это вектор частного решения неоднородной системы дифференциальных уравнений.

Из теории матриц [1] известно свойство перемножаемости матричных экспонент (матриц Коши):

$$K(x_i \leftarrow x_0) = K(x_i \leftarrow x_{i-1}) \cdot K(x_{i-1} \leftarrow x_{i-2}) \cdot \ldots \cdot K(x_2 \leftarrow x_1) \cdot K(x_1 \leftarrow x_0).$$

В случае, когда система дифференциальных уравнений имеет матрицу с переменными коэффициентами $A = A(x)$, решение задачи Коши можно (как это известно из теории матриц) искать при помощи свойства перемножаемости матриц Коши. То есть интервал интегрирования разбивается на малые участки и на малых участках матрицы Коши приближенно вычисляются по формуле для постоянной матрицы в экспоненте. А затем матрицы Коши, вычисленные на малых участках, перемножаются:

$$K(x_i \leftarrow x_0) = K(x_i \leftarrow x_{i-1}) \cdot K(x_{i-1} \leftarrow x_{i-2}) \cdot \ldots \cdot K(x_2 \leftarrow x_1) \cdot K(x_1 \leftarrow x_0),$$

где матрицы Коши приближенно вычисляются по формуле:

$$K(x_{i+1} \leftarrow x_i) = e^{A(xi)\cdot\Delta xi} = \exp(A(x_i)\cdot\Delta x_i)\text{, где } \Delta x_i = x_{i+1} - x_i\,.$$

Вместо формулы для вычисления вектора частного решения неоднородной системы дифференциальных уравнений в виде [1]:

$$\boldsymbol{Y}^*(x \leftarrow x_0) = e^{Ax}\int_{x0}^{x} e^{-At}\boldsymbol{F}(t)dt$$

предлагается использовать следующую формулу для каждого отдельного участка интервала интегрирования:

$$\boldsymbol{Y}^*(x_j \leftarrow x_i) = \boldsymbol{Y}^*(x_j - x_i) = K(x_j - x_i)\int_{x_i}^{x_j} K(x_i - t)\boldsymbol{F}(t)dt\,.$$

Правильность приведенной формулы подтверждается следующим:

$$\boldsymbol{Y}^*(x_j - x_i) = \exp(A(x_j - x_i))\int_{x_i}^{x_j}\exp(A(x_i - t))\boldsymbol{F}(t)dt\,,$$

$$\boldsymbol{Y}^*(x_j - x_i) = \int_{x_i}^{x_j}\exp(A(x_j - x_i))\exp(A(x_i - t))\boldsymbol{F}(t)dt\,,$$

$$\boldsymbol{Y}^*(x_j - x_i) = \int_{x_i}^{x_j}\exp(A(x_j - x_i + x_i - t))\boldsymbol{F}(t)dt\,,$$

$$\boldsymbol{Y}^*(x_j - x_i) = \int_{x_i}^{x_j}\exp(A(x_j - t))\boldsymbol{F}(t)dt\,,$$

$$\boldsymbol{Y}^*(x_j - x_i) = \exp(Ax_j)\int_{x_i}^{x_j}\exp(-At)\boldsymbol{F}(t)dt\,,$$

$$\boldsymbol{Y}^*(x \leftarrow x_i) = \exp(Ax)\int_{x_i}^{x}\exp(-At)\boldsymbol{F}(t)dt\,,$$

что и требовалось подтвердить.

Вычисление вектора частного решения неоднородной системы дифференциальных уравнений производиться при помощи представления матрицы Коши под знаком интеграла в виде ряда и интегрирования этого ряда поэлементно:

$$\boldsymbol{Y}^*(x_j \leftarrow x_i) = \boldsymbol{Y}^*(x_j - x_i) = K(x_j - x_i)\int_{x_i}^{x_j} K(x_i - t)\boldsymbol{F}(t)dt =$$

$$= K(x_j - x_i)\int_{x_i}^{x_j} (E + A(x_i - t) + A^2(x_i - t)^2 / 2!+...)\boldsymbol{F}(t)dt =$$

$$= K(x_j - x_i)(E\int_{x_i}^{x_j} \boldsymbol{F}(t)dt + A\int_{x_i}^{x_j} (x_i - t)\boldsymbol{F}(t)dt + A^2/2!\int_{x_i}^{x_j} (x_i - t)^2\boldsymbol{F}(t)dt + ...).$$

Эта формула справедлива для случая системы дифференциальных уравнений с постоянной матрицей коэффициентов A=const.

Вектор $\boldsymbol{F}(t)$ может рассматриваться на участке $(x_j - x_i)$ приближенно в виде постоянной величины $\boldsymbol{F}(x_i) = \text{constant}$, что позволяет вынести его из под знака интеграла, что приводит к совсем простому ряду для вычислений на рассматриваемом участке.

Для случая дифференциальных уравнений с переменными коэффициентами в приведенной выше формуле для каждого участка может использоваться осредненная матрица $A_i = A(x_i)$ коэффициентов системы дифференциальных уравнений.

Рассмотрим вариант, когда шаги интервала интегрирования выбираются достаточно малыми, что позволяет рассматривать вектор $\boldsymbol{F}(t)$ на участке $(x_j - x_i)$ приближенно в виде постоянной величины $\boldsymbol{F}(x_i) = \text{constant}$, что позволяет вынести этот вектор из под знаков интегралов:

$$\boldsymbol{Y}^*(x_j \leftarrow x_i) = K(x_j - x_i)(E\int_{x_i}^{x_j} dt + A\int_{x_i}^{x_j} (x_i - t)dt + A^2/2!\int_{x_i}^{x_j} (x_i - t)^2 dt + ...)\boldsymbol{F}(t).$$

Известно, что при T=(at+b) имеем $\int \mathrm{T}^{\mathrm{n}} dt = \dfrac{1}{\mathrm{a}(\mathrm{n}+1)}\mathrm{T}^{\mathrm{n}+1} + \text{const}$ (при $\mathrm{n} \neq -1$).

В нашем случае имеем $\int (\mathrm{b} - \mathrm{t})^{\mathrm{n}} dt = \dfrac{1}{(-1)(\mathrm{n}+1)}(\mathrm{b} - \mathrm{t})^{\mathrm{n}+1} + \text{const}$ (при $\mathrm{n} \neq -1$).

Тогда получаем $\int_{x_i}^{x_j} (x_i - t)^n \, dt = -\frac{1}{\mathrm{n}+1}(x_i - x_j)^{n+1}$.

Тогда получаем ряд для вычисления вектора частного решения неоднородной системы дифференциальных уравнений на **малом** участке $(x_j - x_i)$:

$$\boldsymbol{Y}^*(x_j \leftarrow x_i) = K(x_j - x_i) \cdot (E + A(x_i - x_j)/2! + A^2(x_i - x_j)^2/3! + ...) \cdot (x_j - x_i) \cdot \boldsymbol{F}(x_i).$$

Если участок $(x_j - x_i)$ **не мал**, то его можно поделить на подучастки и тогда можно предложить следующие рекуррентные (итерационные) формулы для вычисления частного вектора:

Имеем $\boldsymbol{Y}(x) = K(x \leftarrow x_0)\boldsymbol{Y}(x_0) + \boldsymbol{Y}^*(x \leftarrow x_0)$.

Также имеем формулу для отдельного подучастка:

$$\boldsymbol{Y}^*(x_j \leftarrow x_i) = \boldsymbol{Y}^*(x_j - x_i) = K(x_j - x_i) \int_{x_i}^{x_j} K(x_i - t)\boldsymbol{F}(t)dt \, .$$

Можем записать:

$$\boldsymbol{Y}(x_1) = K(x_1 \leftarrow x_0)\boldsymbol{Y}(x_0) + \boldsymbol{Y}^*(x_1 \leftarrow x_0),$$

$$\boldsymbol{Y}(x_2) = K(x_2 \leftarrow x_1)\boldsymbol{Y}(x_1) + \boldsymbol{Y}^*(x_2 \leftarrow x_1).$$

Подставим $\boldsymbol{Y}(x_1)$ в $\boldsymbol{Y}(x_2)$ и получим:

$$\boldsymbol{Y}(x_2) = K(x_2 \leftarrow x_1)[K(x_1 \leftarrow x_0)\boldsymbol{Y}(x_0) + \boldsymbol{Y}^*(x_1 \leftarrow x_0)] + \boldsymbol{Y}^*(x_2 \leftarrow x_1) =$$

$$= K(x_2 \leftarrow x_1)K(x_1 \leftarrow x_0)\boldsymbol{Y}(x_0) + K(x_2 \leftarrow x_1)\boldsymbol{Y}^*(x_1 \leftarrow x_0) + \boldsymbol{Y}^*(x_2 \leftarrow x_1).$$

Сравним полученное выражение с формулой:

$$\boldsymbol{Y}(x_2) = K(x_2 \leftarrow x_0)\boldsymbol{Y}(x_0) + \boldsymbol{Y}^*(x_2 \leftarrow x_0)$$

и получим, очевидно, что:

$$K(x_2 \leftarrow x_0) = K(x_2 \leftarrow x_1)K(x_1 \leftarrow x_0)$$

и для частного вектора получаем формулу:

$$\boldsymbol{Y}^*(x_2 \leftarrow x_0) = K(x_2 \leftarrow x_1)\boldsymbol{Y}^*(x_1 \leftarrow x_0) + \boldsymbol{Y}^*(x_2 \leftarrow x_1).$$

То есть вектора подучастков $\boldsymbol{Y}^*(x_1 \leftarrow x_0), \boldsymbol{Y}^*(x_2 \leftarrow x_1)$ не просто складываются друг с другом, а с участием матрицы Коши подучастка.

Аналогично запишем $\boldsymbol{Y}(x_3) = K(x_3 \leftarrow x_2)\boldsymbol{Y}(x_2) + \boldsymbol{Y}^*(x_3 \leftarrow x_2)$ и подставим сюда формулу для $\boldsymbol{Y}(x_2)$ и получим:

$$\boldsymbol{Y}(x_3) = K(x_3 \leftarrow x_2)[K(x_2 \leftarrow x_1)K(x_1 \leftarrow x_0)\boldsymbol{Y}(x_0) + K(x_2 \leftarrow x_1)\boldsymbol{Y}^*(x_1 \leftarrow x_0) + \boldsymbol{Y}^*(x_2 \leftarrow x_1)] +$$
$$+ \boldsymbol{Y}^*(x_3 \leftarrow x_2) = K(x_3 \leftarrow x_2)K(x_2 \leftarrow x_1)K(x_1 \leftarrow x_0)\boldsymbol{Y}(x_0) +$$
$$+ K(x_3 \leftarrow x_2)K(x_2 \leftarrow x_1)\boldsymbol{Y}^*(x_1 \leftarrow x_0) + K(x_3 \leftarrow x_2)\boldsymbol{Y}^*(x_2 \leftarrow x_1) + \boldsymbol{Y}^*(x_3 \leftarrow x_2).$$

Сравнив полученное выражение с формулой:

$$\boldsymbol{Y}(x_3) = K(x_3 \leftarrow x_0)\boldsymbol{Y}(x_0) + \boldsymbol{Y}^*(x_3 \leftarrow x_0)$$

очевидно, получаем, что:

$$K(x_3 \leftarrow x_0) = K(x_3 \leftarrow x_2)K(x_2 \leftarrow x_1)K(x_1 \leftarrow x_0)$$

и вместе с этим получаем формулу для частного вектора:

$$\boldsymbol{Y}^*(x_3 \leftarrow x_0) = K(x_3 \leftarrow x_2)K(x_2 \leftarrow x_1)\boldsymbol{Y}^*(x_1 \leftarrow x_0) + K(x_3 \leftarrow x_2)\boldsymbol{Y}^*(x_2 \leftarrow x_1) + \boldsymbol{Y}^*(x_3 \leftarrow x_2).$$

То есть именно так и вычисляется частный вектор – вектор частного решения неоднородной системы дифференциальных уравнений, то есть так вычисляется, например, частный вектор $\boldsymbol{Y}^*(x_3 \leftarrow x_0)$ на рассматриваемом участке $(x_3 \leftarrow x_0)$ через вычисленные частные вектора $\boldsymbol{Y}^*(x_1 \leftarrow x_0)$, $\boldsymbol{Y}^*(x_2 \leftarrow x_1)$, $\boldsymbol{Y}^*(x_3 \leftarrow x_2)$ соответствующих подучастков $(x_1 \leftarrow x_0)$, $(x_2 \leftarrow x_1)$, $(x_3 \leftarrow x_2)$.

2. Контроль точности вычислений.

В случае использования описанной кусочно-константной аппроксимации матрицы системы обыкновенных дифференциальных уравнений (ОДУ) с переменными коэффициентами, когда на всем интервале интегрирования системы ОДУ используются матричные экспоненты от осредненных постоянных аргументов, оценка точности теоретически не дается и предлагаемый метод в этом частном случае можно считать «инженерным», который дает достаточно точные решения для уже опробованных инженерных задач.

В тоже время можно производить вычисления и иначе - с заранее известной точностью.

Для этого следует вычислять матрицы Коши не как матричные экспоненты от осредненных аргументов на своих малых участках всего интервала интегрирования, а можно использовать для вычисления векторов, входящих в матрицы Коши, методы типа методов Рунге-Кутты.

В этом случае для вычисления матриц Коши методами типа Рунге-Кутты используются стартовые (начальные) вектора, взятые из единичной матрицы для вычисления части решения, которая соответствует однородной системе ОДУ, и берется стартовый нулевой вектор для вычисления вектора частного решения неоднородной системы ОДУ.

В таком варианте предлагаемого метода - в случае применения методов типа Рунге-Кутты для вычисления матриц Коши – хорошо известны оценки точности приближенных вычислений, что означает, что вычисления можно производить с заранее известной погрешностью, так как оценки погрешностей методов Рунге-Кутты известны.

3. Простейший метод решения жестких краевых задач.

Идея преодоления трудностей решения жёстких краевых задач путём разделения интервала интегрирования на сопрягаемые участки принадлежит д.ф.-м.н. Ю.И.Виноградову, а выражение этого сопряжения через формулы теории матриц принадлежит к.ф.-м.н. А.Ю.Виноградову.

Разделим интервал интегрирования краевой задачи, например, на 3 участка. Будем иметь точки (узлы), включая края:

$$x_0, x_1, x_2, x_3 .$$

Имеем краевые условия в виде:

$$U\boldsymbol{Y}(x_0) = \boldsymbol{u},$$
$$V\boldsymbol{Y}(x_3) = \boldsymbol{v}.$$

Можем записать матричные уравнения сопряжения участков:

$$\boldsymbol{Y}(x_0) = K(x_0 \leftarrow x_1)\boldsymbol{Y}(x_1) + \boldsymbol{Y}^*(x_0 \leftarrow x_1),$$

$$\boldsymbol{Y}(x_1) = K(x_1 \leftarrow x_2)\boldsymbol{Y}(x_2) + \boldsymbol{Y}^*(x_1 \leftarrow x_2),$$

$$\boldsymbol{Y}(x_2) = K(x_2 \leftarrow x_3)\boldsymbol{Y}(x_3) + \boldsymbol{Y}^*(x_2 \leftarrow x_3).$$

Это мы можем переписать в виде, более удобном для нас далее:

$$E\boldsymbol{Y}(x_0) - K(x_0 \leftarrow x_1)\boldsymbol{Y}(x_1) = \boldsymbol{Y}^*(x_0 \leftarrow x_1),$$

$$E\boldsymbol{Y}(x_1) - K(x_1 \leftarrow x_2)\boldsymbol{Y}(x_2) = \boldsymbol{Y}^*(x_1 \leftarrow x_2),$$

$$E\boldsymbol{Y}(x_2) - K(x_2 \leftarrow x_3)\boldsymbol{Y}(x_3) = \boldsymbol{Y}^*(x_2 \leftarrow x_3).$$

где E - единичная матрица.

Тогда в объединенном матричном виде получаем систему линейных алгебраических уравнений в следующей форме:

$$\begin{Vmatrix} U & 0 & 0 & 0 \\ E & -K(x_0 \leftarrow x_1) & 0 & 0 \\ 0 & E & -K(x_1 \leftarrow x_2) & 0 \\ 0 & 0 & E & -K(x_2 \leftarrow x_3) \\ 0 & 0 & 0 & V \end{Vmatrix} \cdot \begin{Vmatrix} \boldsymbol{Y}(x_0) \\ \boldsymbol{Y}(x_1) \\ \boldsymbol{Y}(x_2) \\ \boldsymbol{Y}(x_3) \end{Vmatrix} = \begin{Vmatrix} \boldsymbol{u} \\ \boldsymbol{Y}^*(x_0 \leftarrow x_1) \\ \boldsymbol{Y}^*(x_1 \leftarrow x_2) \\ \boldsymbol{Y}^*(x_2 \leftarrow x_3) \\ \boldsymbol{v} \end{Vmatrix}.$$

Эта система решается методом Гаусса с выделением главного элемента.

В точках, расположенных между узлами, решение находиться при помощи решения задач Коши с начальными условиями в i-ом узле:

$$\boldsymbol{Y}(x) = K(x \leftarrow x_i)\boldsymbol{Y}(x_i) + \boldsymbol{Y}^*(x \leftarrow x_i).$$

Применять ортонормирование для краевых задач для жестких обыкновенных дифференциальных уравнений в рамках предложенного метода оказывается не надо.

Это объясняется тем, что на каждом отдельном участке интервала интегрирования вычисления матриц Коши ведется каждый раз отдельно (независимо от других участков) со стартом численного интегрирования (либо в виде матричной экспоненты либо методами типа Рунге-Кутты) от начальной единичной матрицы, то есть от ортонормированной (единичной) матрицы. И этот старт численного интегрирования на каждом отдельном участке ведется от ортонормированной системы векторов (от единичной матрицы) без каких-либо дополнительных операций типа операций ортонормирования векторов, которые применяются в методе С.К.Годунова.

И так как предлагаемый метод на каждом отдельном участке интервала интегрирования реализуется от единичной (ортонормированной) матрицы, то нет необходимости в программировании процедур ортонормирования, в отличие от метода С.К.Годунова, что делает программирование предлагаемого метода гораздо более простым по сравнению с методом С.К.Годунова.

4. Вычислительные эксперименты.

Вычислительные эксперименты проводились в сравнении с методом Виноградовых [2] переноса краевых условий. В этом методе используется построчное ортонормирование.

Без ортонормирования в методе переноса краевых условий Виноградовых успешно решается задача, например, нагружения цилиндрической оболочки, которая

консольно заделана по правому краю и нагружена по левому краю силой, равномерно распределенной по дуге окружности, с отношением длинны к радиусу L/R=2 и с отношением радиуса к толщине R/h=100. Для отношения R/h=200 задача без ортонормирования в методе переноса краевых условий уже не решается, так как выдаются ошибки из-за неустойчивости счета. С применением же ортонормирования в методе переноса краевых условий решаются успешно задачи и для параметров, например, R/h=300, R/h=500, R/h=1000.

Новый предлагаемый здесь метод позволяет решать все вышеуказанные тестовые задачи вовсе без применения операций ортонормирования, что значительно упрощает его программирование.

Для тестовых расчетов задач с вышеуказанными параметрами новым предлагаемым методом интервал интегрирования разделялся на 10 участков, а между узлами, как и сказано выше, решение находилось как решение задачи Коши. Для решения задач удерживалось 50 гармоник рядов Фурье, так как результат при 50 гармониках уже не отличался от случая удержания 100 гармоник.

Скорость же расчета тестовых задач новым предлагаемым методом не меньше, чем методом переноса краевых условий, так как оба метода в тестовых задачах при удержании 50 гармоник рядов Фурье выдавали готовое решение мгновенно после запуска программы на выполнение (на ноутбуке ASUS M51V CPU Duo T5800). В тоже время программирование нового предложенного здесь метода существенно проще, так как нет необходимости программировать процедуры ортонормирования.

СПИСОК ЛИТЕРАТУРЫ

1. *Гантмахер Ф.Р.*. Теория матриц. – М.: Наука, 1988. – 548 с.

2. *Виноградов А.Ю., Виноградов Ю.И.* Метод переноса краевых условий функциями Коши-Крылова для жёстких линейных обыкновенных дифференциальных уравнений. // ДАН. – М.: 2000, т.373, №4, с.474-476.

Приложение. Программа (код) на языке программирования C++.

```
// sopryazhenie.cpp: главный файл проекта.
//Решение краевой задачи - цилиндрической оболочки.
//Интервал интегрирования разбит на 10 сопрягаемых участков: левый край - точка 0 и
правый край - точка 10
//БЕЗ ОРТОНОРМИРОВАНИЯ
```

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>

using namespace std;

//Умножение матрицы A на вектор b и получаем rezult.
void mat_on_vect(double A[8][8], double b[8], double rezult[8]){
	for(int i=0;i<8;i++){
		rezult[i]=0.0;
		for(int k=0;k<8;k++){
		rezult[i]+=A[i][k]*b[k];
		}
	}
}

//Вычисление матричной экспоненты EXP=exp(A*delta_x)
void exponent(double A[8][8], double delta_x, double EXP[8][8]) {

	//n - количество членов ряда в экспоненте, m - счетчик членов ряда (m<=n)
	int n=100, m;
	double E[8][8]={0}, TMP1[8][8], TMP2[8][8];
	int i,j,k;

	//E - единичная матрица - первый член ряда экспоненты
	E[0][0]=1.0; E[1][1]=1.0; E[2][2]=1.0; E[3][3]=1.0;
	E[4][4]=1.0; E[5][5]=1.0; E[6][6]=1.0; E[7][7]=1.0;

	//первоначальное заполнение вспомогательного массива TMP1 - предыдущего члена
ряда для следующего перемножения
	//и первоначальное заполнение экспоненты первым членом ряда
	for(i=0;i<8;i++) {
		for(j=0;j<8;j++) {
			TMP1[i][j]=E[i][j];
			EXP[i][j]=E[i][j];
		}
	}

	//ряд вычисления экспоненты EXP, начиная со 2-го члена ряда (m=2;m<=n)
	for(m=2;m<=n;m++) {
		for(i=0;i<8;i++) {
			for(j=0;j<8;j++) {
				TMP2[i][j]=0;
				for(k=0;k<8;k++) {
				  //TMP2[i][j]+=TMP1[i][k]*A[k][j]*delta_x/(m-1);
					TMP2[i][j]+=TMP1[i][k]*A[k][j];
				}
				TMP2[i][j]*=delta_x;//вынесено за цикл произведения строки
на столбец
				TMP2[i][j]/=(m-1);//вынесено за цикл произведения строки на
столбец
				EXP[i][j]+=TMP2[i][j];
			}
		}

		//заполнение вспомогательного массива TMP1 для вычисления следующего члена
ряда - TMP2 в следующем шаге цикла по m
		if (m<n) {
			for(i=0;i<8;i++) {
				for(j=0;j<8;j++) {
```

```
                                        TMP1[i][j]=TMP2[i][j];
                                }
                        }
                }
        }


        }

//Вычисление матрицы MAT_ROW в виде матричного ряда для последующего использования
//при вычислении вектора partial_vector - вектора частного решения неоднородной системы
ОДУ на шаге delta_x
void mat_row_for_partial_vector(double A[8][8], double delta_x, double MAT_ROW[8][8]) {

        //n - количество членов ряда в MAT_ROW, m - счетчик членов ряда (m<=n)
        int n=100, m;
        double E[8][8]={0}, TMP1[8][8], TMP2[8][8];
        int i,j,k;

        //E - единичная матрица - первый член ряда MAT_ROW
        E[0][0]=1.0; E[1][1]=1.0; E[2][2]=1.0; E[3][3]=1.0;
        E[4][4]=1.0; E[5][5]=1.0; E[6][6]=1.0; E[7][7]=1.0;

        //первоначальное заполнение вспомогательного массива TMP1 - предыдущего члена
ряда для следующего перемножения
        //и первоначальное заполнение MAT_ROW первым членом ряда
        for(i=0;i<8;i++) {
                for(j=0;j<8;j++) {
                        TMP1[i][j]=E[i][j];
                        MAT_ROW[i][j]=E[i][j];
                }
        }

        //ряд вычисления MAT_ROW, начиная со 2-го члена ряда (m=2;m<=n)
        for(m=2;m<=n;m++) {
                for(i=0;i<8;i++) {
                        for(j=0;j<8;j++) {
                                TMP2[i][j]=0;
                                for(k=0;k<8;k++) {
                                        TMP2[i][j]+=TMP1[i][k]*A[k][j];
                                }
                                TMP2[i][j]*=delta_x;
                                TMP2[i][j]/=m;
                                MAT_ROW[i][j]+=TMP2[i][j];
                        }
                }

                //заполнение вспомогательного массива TMP1 для вычисления следующего члена
ряда - TMP2 в следующем шаге цикла по m
                if (m<n) {
                        for(i=0;i<8;i++) {
                                for(j=0;j<8;j++) {
                                        TMP1[i][j]=TMP2[i][j];
                                }
                        }
                }
        }
        }

//Задание вектора внешних воздействий в системе ОДУ - вектора POWER:
Y'(x)=A*Y(x)+POWER(x):
```

```
void power_vector_for_partial_vector(double x, double POWER[8]){
	POWER[0]=0.0;
	POWER[1]=0.0;
	POWER[2]=0.0;
	POWER[3]=0.0;
	POWER[4]=0.0;
	POWER[5]=0.0;
	POWER[6]=0.0;
	POWER[7]=0.0;
}

//Вычисление vector - НУЛЕВОГО (частный случай) вектора частного решения
//неоднородной системы дифференциальных уравнений на рассматриваемом участке:
void partial_vector(double vector[8]){
	for(int i=0;i<8;i++){
		vector[i]=0.0;
	}
}

//Вычисление vector - вектора частного решения неоднородной системы дифференциальных
уравнений на рассматриваемом участке delta_x:
void partial_vector_real(double expo_[8][8], double mat_row[8][8], double x_, double
delta_x, double vector[8]){
	double POWER_[8]={0};//Вектор внешней нагрузки на оболочку
	double REZ[8]={0};
	double REZ_2[8]={0};
	power_vector_for_partial_vector(x_, POWER_);//Расчитываем POWER_ при координате
x_
	mat_on_vect(mat_row, POWER_, REZ);//Умножение матрицы mat_row на вектор POWER_ и
получаем вектор REZ
	mat_on_vect(expo_, REZ, REZ_2);//Умножение матрицы expo_ на вектор REZ и получаем
вектор REZ_2
	for(int i=0;i<8;i++){
		vector[i]=REZ_2[i]*delta_x;
	}
}

//Решение СЛАУ размерности 88 методом Гаусса с выделением главного элемента
int GAUSS(double AA[8*11][8*11], double bb[8*11], double x[8*11]){
	double A[8*11][8*11];
	double b[8*11];
	for(int i=0;i<(8*11);i++){
		b[i]=bb[i];//Работать будем с вектором правых частей b, чтобы исходный
вектор bb не изменялся при выходе из подпрограммы
		for(int j=0;j<(8*11);j++){
			A[i][j]=AA[i][j];//Работать будем с матрицей A, чтобы исходная
матрица AA не менялась при выходе из подпрограммы
		}
	}

	int e;//номер строки, где обнаруживается главный (максимальный) коэффициент в
столбце jj
	double s, t, main;//Вспомогательная величина

	for(int jj=0;jj<((8*11)-1);jj++){//Цикл по столбцам jj преобразования матрицы A в
верхнетреугольную

		e=-1; s=0.0; main=A[jj][jj];
		for(int i=jj;i<(8*11);i++){//Находится номер e строки, где лежит главный
(максимальный) элемент в столбце jj и делается взаимозамена строк
```

```
                        if ((A[i][jj]*A[i][jj])>s) {//Вместо перемножения (удаляется
возможный знак минуса) можно было бы использовать функцию по модулю abs()
                                e=i; s=A[i][jj]*A[i][jj];
                        }
                }

                if (e<0) {
                cout<<"Mistake "<<jj<<"\n"; return 0;
                }
                if (e>jj) {//Если главный элемент не в строке с номером jj. а в строке с
номером e
                        main=A[e][jj];
                        for(int j=0;j<(8*11);j++){//Взаимная замена двух строк - с номерами
e и jj
                                t=A[jj][j]; A[jj][j]=A[e][j]; A[e][j]=t;
                        }
                        t=b[jj]; b[jj]=b[e]; b[e]=t;
                }

                for(int i=(jj+1);i<(8*11);i++){//Приведение к верхнетреугольной матрице
                        for(int j=(jj+1);j<(8*11);j++){
                                A[i][j]=A[i][j]-(1/main)*A[jj][j]*A[i][jj];//Перерасчет
коэффициентов строки i>(jj+1)
                        }
                        b[i]=b[i]-(1/main)*b[jj]*A[i][jj];
                        A[i][jj]=0.0;//Обнуляемые элементы столбца под диагональным
элементом матрицы A
                }

        }//Цикл по столбцам jj преобразования матрицы A в верхнетреугольную

        x[(8*11)-1]=b[(8*11)-1]/A[(8*11)-1][(8*11)-1];//Первоначальное определение
последнего элемента искомого решения x (87-го)
        for(int i=((8*11)-2);i>=0;i--){//Вычисление елементов решения x[i] от 86-го до 0-
го
                t=0;
                for(int j=1;j<((8*11)-i);j++){
                        t=t+A[i][i+j]*x[i+j];
                }
                x[i]=(1/A[i][i])*(b[i]-t);
        }

        return 0;
}

int main()
{
        int nn;//Номер гармоники, начиная с 1-й (без нулевой)
        int nn_last=50;//Номер последней гармоники
        double Moment[100+1]={0};//Массив физического параметра (момента), что
рассчитывается в каждой точке между краями

        double step=0.05; //step=(L/R)/100 - величина шага расчета оболочки - шага
интервала интегрирования (должна быть больше нуля, т.е. положительная)

        double h_div_R;//Величина h/R
        h_div_R=1.0/100;
        double c2;
        c2=h_div_R*h_div_R/12;//Величина h*h/R/R/12
        double nju;
        nju=0.3;
```

```
       double gamma;
       gamma=3.14159265359/4;//Угол распределения силы по левому краю

       //распечатка в файлы:
       FILE *fp;

       // Open for write
    if( (fp = fopen( "C:/test.txt", "w" )) == NULL ) // C4996
       printf( "The file 'C:/test.txt' was not opened\n" );
    else
       printf( "The file 'C:/test.txt' was opened\n" );



       for(nn=1;nn<=nn_last;nn++){ //ЦИКЛ ПО ГАРМОНИКАМ, НАЧИНАЯ С 1-ОЙ ГАРМОНИКИ (БЕЗ
НУЛЕВОЙ ГАРМОНИКИ)

       double x=0.0;//Координата от левого края - нужна для случая неоднородной системы
ОДУ для вычисления частного вектора FF
       double expo_from_minus_step[8][8]={0};//Матрица для расположения в ней экспоненты
на шаге типа (0-x1)
       double expo_from_plus_step[8][8]={0};//Матрица для расположения в ней экспоненты
на шаге типа (x1-0)
       double mat_row_for_minus_expo[8][8]={0};//вспомогательная матрица для расчета
частного вектора при движении на шаге типа (0-x1)
       double mat_row_for_plus_expo[8][8]={0};//вспомогательная матрица для расчета
частного вектора при движении на шаге типа (x1-0)

       double U[4][8]={0};//Матрица краевых условий левого края размерности 4x8
       double u_[4]={0};//Вектор размерности 4 внешнего воздействия для краевых условий
левого края
       double V[4][8]={0};//Матрица краевых условий правого края размерности 4x8
       double v_[4]={0};//Вектор размерности 4 внешнего воздействия для краевых условий
правого края
       double Y[100+1][8]={0};//Массив векторов-решений соответствующих СЛАУ (в каждой
точке интервала между краями): MATRIXS*Y=VECTORS
       double A[8][8]={0};//Матрица коэффициентов системы ОДУ
       double FF[8]={0};//Вектор частного решения неоднородной ОДУ на участке интервала
интегрирования


       double Y_many[8*11]={0};// составной вектор из векторов Y(xi) в 11-ти точках с
точки 0 (левый край Y(0)) до точки 10 (правый край Y(x10))
       double MATRIX_many[8*11][8*11]={0};//матрица СЛАУ
       double B_many[8*11]={0};// вектор правых частей СЛАУ: MATRIX_many*Y_many=B_many
       double Y_vspom[8]={0};//вспомогательный вектор
       double Y_rezult[8]={0};//вспомогательный вектор


       double nn2,nn3,nn4,nn5,nn6,nn7,nn8;//Возведенный в соответствующие степени номер
гармоники nn
       nn2=nn*nn; nn3=nn2*nn; nn4=nn2*nn2; nn5=nn4*nn; nn6=nn4*nn2; nn7=nn6*nn;
nn8=nn4*nn4;

       //Заполнение ненулевых элементов матрицы A коэффициентов системы ОДУ
       A[0][1]=1.0;
       A[1][0]=(1-nju)/2*nn2; A[1][3]=-(1+nju)/2*nn; A[1][5]=-nju;
       A[2][3]=1.0;
       A[3][1]=(1+nju)/(1-nju)*nn; A[3][2]=2*nn2/(1-nju); A[3][4]=2*nn/(1-nju);
       A[4][5]=1.0;
```

```
	A[5][6]=1.0;
	A[6][7]=1.0;
	A[7][1]=-nju/c2; A[7][2]=-nn/c2; A[7][4]=-(nn4+1/c2); A[7][6]=2*nn2;


	//Здесь надо первоначально заполнить ненулевыми значениями матрицы и вектора
краевых условий U*Y[0]=u_ (слева) и V*Y[100]=v_ (справа) :
	U[0][1]=1.0; U[0][2]=nn*nju; U[0][4]=nju; u_[0]=0.0;//Сила T1 на левом крае равна
нулю
	U[1][0]=-(1-nju)/2*nn; U[1][3]=(1-nju)/2; U[1][5]=(1-nju)*nn*c2; u_[1]=0.0;//Сила
S* на левом краю равна нулю
	U[2][4]=-nju*nn2; U[2][6]=1.0; u_[2]=0;//Момент M1 на левом краю равен нулю
	U[3][5]=(2-nju)*nn2; U[3][7]=-1.0;
	u_[3]=-sin(nn*gamma)/(nn*gamma);//Сила Q1* на левом крае распределена на угол -
gamma +gamma


	V[0][0]=1.0; v_[0]=0.0;//Перемещение u на правом крае равно нулю
	V[1][2]=1.0; v_[1]=0.0;//Перемещение v на правом крае равно нулю
	V[2][4]=1.0; v_[2]=0.0;//Перемещение w на правом крае равно нулю
	V[3][5]=1.0; v_[3]=0.0;//Угол поворота на правом крае равен нулю
	//Здесь заканчивается первоначальное заполнение U*Y[0]=u_ и V*Y[100]=v_


	exponent(A,(-step*10),expo_from_minus_step);//Шаг отрицательный (значение шага
меньше нуля из-за направления вычисления матричной экспоненты)
	//x=0.0;//начальное значение координаты - для расчета частного вектора
	//mat_row_for_partial_vector(A, step, mat_row_for_minus_expo);

	//Заполнение матрицы коэффициентов СЛАУ MATRIX_many
	for(int i=0;i<4;i++){
		for(int j=0;j<8;j++){
			MATRIX_many[i][j]=U[i][j];
			MATRIX_many[8*11-4+i][8*11-8+j]=V[i][j];
		}
		B_many[i]=u_[i];
		B_many[8*11-4+i]=v_[i];
	}

	for(int kk=0;kk<(11-1);kk++){//(11-1) единичных матриц и матриц EXPO надо
записать в MATRIX_many
		for(int i=0;i<8;i++){
			MATRIX_many[i+4+kk*8][i+kk*8]=1.0;//заполнение единичными матрицами
			for(int j=0;j<8;j++){
				MATRIX_many[i+4+kk*8][j+8+kk*8]=-
expo_from_minus_step[i][j];//заполнение матричными экспонентами
			}
		}
	}

	//Решение систем линейных алгебраических уравнений
	GAUSS(MATRIX_many,B_many,Y_many);

	//Вычисление векторов состояния в 101 точке - левая точка 0 и правая точка 100
	exponent(A,step,expo_from_plus_step);

	for(int i=0;i<11;i++){//заполнение промежуточных точек во всех 10-ти интервалах
(всего получим точки от 0 до 100) между 11 узлами
		for(int j=0;j<8;j++){
			Y[0+i*10][j]=Y_many[j+i*8];//в 11-ти узлах векторы беруться из
решения СЛАУ - из Y_many
```

```
		}
	}
	for(int i=0;i<10;i++){//заполнение промежуточных точек в 10-ти интервалах
		for(int j=0;j<8;j++){
			Y_vspom[j]=Y[0+i*10][j];//начальный вектор для i-го участка,
нулевая точка, точка старта i-го участка
		}

		mat_on_vect(expo_from_plus_step, Y_vspom, Y_rezult);
		for(int j=0;j<8;j++){
			Y[0+i*10+1][j]=Y_rezult[j];//заполнение 1-ой точки интервала
			Y_vspom[j]=Y_rezult[j];//для следующего шага
		}

		mat_on_vect(expo_from_plus_step, Y_vspom, Y_rezult);
		for(int j=0;j<8;j++){
			Y[0+i*10+2][j]=Y_rezult[j];//заполнение 2-ой точки интервала
			Y_vspom[j]=Y_rezult[j];//для следующего шага
		}

		mat_on_vect(expo_from_plus_step, Y_vspom, Y_rezult);
		for(int j=0;j<8;j++){
			Y[0+i*10+3][j]=Y_rezult[j];//заполнение 3-ой точки интервала
			Y_vspom[j]=Y_rezult[j];//для следующего шага
		}

		mat_on_vect(expo_from_plus_step, Y_vspom, Y_rezult);
		for(int j=0;j<8;j++){
			Y[0+i*10+4][j]=Y_rezult[j];//заполнение 4-ой точки интервала
			Y_vspom[j]=Y_rezult[j];//для следующего шага
		}

		mat_on_vect(expo_from_plus_step, Y_vspom, Y_rezult);
		for(int j=0;j<8;j++){
			Y[0+i*10+5][j]=Y_rezult[j];//заполнение 5-ой точки интервала
			Y_vspom[j]=Y_rezult[j];//для следующего шага
		}

		mat_on_vect(expo_from_plus_step, Y_vspom, Y_rezult);
		for(int j=0;j<8;j++){
			Y[0+i*10+6][j]=Y_rezult[j];//заполнение 6-ой точки интервала
			Y_vspom[j]=Y_rezult[j];//для следующего шага
		}

		mat_on_vect(expo_from_plus_step, Y_vspom, Y_rezult);
		for(int j=0;j<8;j++){
			Y[0+i*10+7][j]=Y_rezult[j];//заполнение 7-ой точки интервала
			Y_vspom[j]=Y_rezult[j];//для следующего шага
		}

		mat_on_vect(expo_from_plus_step, Y_vspom, Y_rezult);
		for(int j=0;j<8;j++){
			Y[0+i*10+8][j]=Y_rezult[j];//заполнение 8-ой точки интервала
			Y_vspom[j]=Y_rezult[j];//для следующего шага
		}

		mat_on_vect(expo_from_plus_step, Y_vspom, Y_rezult);
		for(int j=0;j<8;j++){
			Y[0+i*10+9][j]=Y_rezult[j];//заполнение 9-ой точки интервала
			Y_vspom[j]=Y_rezult[j];//для следующего шага
		}
```

```
	}



	//Вычисление момента во всех точках между краями
	for(int ii=0;ii<=100;ii++){
		Moment[ii]+=Y[ii][4]*(-nju*nn2)+Y[ii][6]*1.0;//Момент M1 в точке [ii]
		//U[2][4]=-nju*nn2; U[2][6]=1.0; u_[2]=0;//Момент M1
	}



	}//ЦИКЛ ПО ГАРМОНИКАМ ЗДЕСЬ ЗАКАНЧИВАЕТСЯ




	for(int ii=0;ii<=100;ii++){
		fprintf(fp,"%f\n",Moment[ii]);
	}

	fclose(fp);

	printf( "PRESS any key to continue...\n" );
	_getch();

	return 0;
}
```