Исправленные методы А.Ю.Виноградова решения краевых задач, в том числе жестких краевых задач

к.ф.-м.н. Алексей Юрьевич Виноградов

Мои сайты по методам решения краевых задач в интернете:

www.vinogradov-design.narod.ru/math.html

www.vinogradov-best.narod.ru

www.AlexeiVinogradov.narod.ru

www.VinogradovAlexei.narod.ru

www.Vinogradov-Alexei.narod.ru

www.Vinogradov-Math.narod.ru

Москва, июль 2013

Содержание:

1. Введение.

На примере системы дифференциальных уравнений цилиндрической оболочки ракеты – системы обыкновенных дифференциальных уравнений 8-го порядка (после разделения частных производных).

Система линейных обыкновенных дифференциальных уравнений имеет вид:

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

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

Здесь и далее вектора обозначаем **жирным** шрифтом вместо черточек над буквами

Краевые условия имеют вид:

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

где

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

$\mathbf{Y}(1)$ – значение искомой вектор-функции на правом крае x=1 размерности 8x1, V – прямоугольная горизонтальная матрица коэффициентов краевых условий правого края размерности 4x8, **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 [Гантмахер]:

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

где

$$e^{A(x-x_0)} = E + A(x-x_0) + A^2 (x-x_0)^2/2! + A^3 (x-x_0)^3/3! + \ldots,$$

где E это единичная матрица.

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

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

Тогда решение задачи Коши может быть записано в виде:

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

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

2. Случай переменных коэффициентов.

Этот вариант рассмотрения переменных коэффициентов проверялся в кандидатской диссертации.

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

$$e^{A(x_i - x_0)} = e^{A(x_i - x_{i-1})} \cdot e^{A(x_{i-1} - x_{i-2})} \cdot \ldots \cdot e^{A(x_2 - x_1)} \cdot e^{A(x_1 - x_0)},$$

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

В случае, когда система дифференциальных уравнений имеет матрицу с переменными коэффициентами A=A(x), решение задачи Коши предлагается 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:

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

3. Формула для вычисления вектора частного решения неоднородной системы дифференциальных уравнений.

Эта очень простая формула еще не обсчитана на компьютерах. Вместо неё обсчитывалась (а может быть и не обсчитывалась, не знаю) значительно ранее выведенная (выведенная моим отцом) и гораздо более сложная формула, приведенная в:

Численный метод переноса краевых условий для жестких дифференциальных уравнений строительной механики

Журнал "ММ", Том: 14 (2002), Номер: 9, 3 стр. 1409-003r.pdf

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

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

предлагается использовать следующую формулу для каждого отдельного участка интервала интегрирования:

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

Правильность приведенной формулы подтверждается следующим:

$$\mathbf{Y^*}(x_j - x_i) = e^{A(x_j - x_i)} \cdot \int_{x_i}^{x_j} e^{A(x_i - t)} \cdot \mathbf{F}(t)\, dt ,$$

$$\mathbf{Y^*}(x_j - x_i) = \int_{x_i}^{x_j} e^{A(x_j - x_i)} \cdot e^{A(x_i - t)} \cdot \mathbf{F}(t)\, dt ,$$

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

$$\mathbf{Y^*}(x_j - x_i) = \int_{x_i}^{x_j} e^{A(x_j - t)} \cdot \mathbf{F}(t)\, dt ,$$

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

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

что и требовалось подтвердить.

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

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

$$= K(x_j - x_i) \cdot \int_{x_i}^{x_j} (E + A(x_i - t) + A^2 (x_i - t)^2/2! + \dots) \cdot \mathbf{F}(t)\, dt =$$

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

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

Вектор **F**(t) может рассматриваться на участке $(x_j - x_i)$ приближенно в виде постоянной величины $\mathbf{F}(x_i)$=constant, что позволят 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: $A_i = A(x_i)$ коэффициентов системы дифференциальных уравнений.

Приведем (итерационные или рекуррентные) формулы вычисления вектора частного решения, например, $\mathbf{Y}^*(x_3 \leftarrow x_0)$ на рассматриваемом участке $(x_3 \leftarrow x_0)$ через вектора частного решения $\mathbf{Y}^*(x_1 \leftarrow x_0)$, $\mathbf{Y}^*(x_2 \leftarrow x_1)$, $\mathbf{Y}^*(x_3 \leftarrow x_2)$ соответствующих подучастков $(x_1 \leftarrow x_0)$, $(x_2 \leftarrow x_1)$, $(x_3 \leftarrow x_2)$.

Имеем:

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

Также имеем формулу для отдельного подучасточка:

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

Можем записать:

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

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

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

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

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

Сравним полученное выражение с формулой:

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

и получим, очевидно, что $K(x_2 \leftarrow x_0) = K(x_2 \leftarrow x_1) \cdot K(x_1 \leftarrow x_0)$ и, самое главное здесь - для частного вектора получаем формулу:

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

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

Аналогично запишем:

$$\mathbf{Y}(x_3) = K(x_3 \leftarrow x_2) \cdot \mathbf{Y}(x_2) + \mathbf{Y^*}(x_3 \leftarrow x_2)$$

И подставим сюда формулу для $\mathbf{Y}(x_2)$ и получим:

$$\mathbf{Y}(x_3) = K(x_3 \leftarrow x_2) \cdot [K(x_2 \leftarrow x_1) \cdot K(x_1 \leftarrow x_0) \cdot \mathbf{Y}(x_0) + K(x_2 \leftarrow x_1) \cdot \mathbf{Y^*}(x_1 \leftarrow x_0) +$$

$$\mathbf{Y^*}(x_2 \leftarrow x_1)] + \mathbf{Y^*}(x_3 \leftarrow x_2) =$$

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

$$K(x_3 \leftarrow x_2) \cdot \mathbf{Y^*}(x_2 \leftarrow x_1) + \mathbf{Y^*}(x_3 \leftarrow x_2).$$

Сравнив полученное выражение с формулой:

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

очевидно, получаем, что $K(x_3 \leftarrow x_0) = K(x_3 \leftarrow x_2) \cdot K(x_2 \leftarrow x_1) \cdot K(x_1 \leftarrow x_0)$ и вместе с этим получаем формулу для частного вектора:

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

$$\mathbf{Y^*}(x_3 \leftarrow x_2).$$

То есть именно так (по своеобразным рекуррентным формулам) и вычисляется частный вектор – вектор частного решения неоднородной системы дифференциальных уравнений, то есть так вычисляется, например, частный вектор $\mathbf{Y^*}(x_3 \leftarrow x_0)$ всего участка $(x_3 \leftarrow x_0)$ на основе вычисленных векторов $\mathbf{Y^*}(x_1 \leftarrow x_0)$, $\mathbf{Y^*}(x_2 \leftarrow x_1)$, $\mathbf{Y^*}(x_3 \leftarrow x_2)$ подучастков $(x_1 \leftarrow x_0)$, $(x_2 \leftarrow x_1)$, $(x_3 \leftarrow x_2)$.

4.1. Метод «переноса краевых условий» в произвольную точку интервала интегрирования.

Метод обсчитан на компьютерах. По нему уже сделано 3 кандидатских физ-мат диссертации.

Метод подходит для любых краевых задач. А для «жестких» краевых задач показано, что метод считает быстрее, чем метод С.К.Годунова до 2-х порядков (в 100 раз), а для некоторых «жестких» краевых задач не требует ортонормирования вовсе. Смотри:

Численный метод переноса краевых условий для жестких дифференциальных уравнений строительной механики

Журнал "ММ", Том: 14 (2002), Номер: 9, 3 стр. 1409-003r.pdf

Полное решение системы дифференциальных уравнений имеет вид

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

Или можно записать:

$$\mathbf{Y}(0) = K(0 \leftarrow x_1) \cdot \mathbf{Y}(x_1) + \mathbf{Y}^*(0 \leftarrow x_1) \ .$$

Подставляем это выражение для $\mathbf{Y}(0)$ в краевые условия левого края и получаем:

$$U \cdot \mathbf{Y}(0) = \mathbf{u},$$

$$U \cdot [\ K(0 \leftarrow x_1) \cdot \mathbf{Y}(x_1) + \mathbf{Y}^*(0 \leftarrow x_1)\ ] = \mathbf{u},$$

$$[\ U \cdot K(0 \leftarrow x_1)\ ] \cdot \mathbf{Y}(x_1) = \mathbf{u} - U \cdot \mathbf{Y}^*(0 \leftarrow x_1) \ .$$

Или получаем краевые условия, перенесенные в точку x_1:

$$U_1 \cdot \mathbf{Y}(x_1) = \mathbf{u}_1 \ ,$$

где $U_1 = [\ U \cdot K(0 \leftarrow x_1)\ ]$ и $\mathbf{u}_1 = \mathbf{u} - U \cdot \mathbf{Y}^*(0 \leftarrow x_1)$.

Далее запишем аналогично

$$\mathbf{Y}(x_1) = K(x_1 \leftarrow x_2) \cdot \mathbf{Y}(x_2) + \mathbf{Y}^*(x_1 \leftarrow x_2)$$

И подставим это выражение для $\mathbf{Y}(x_1)$ в перенесенные краевые условия точки x_1

$$U_1 \cdot \mathbf{Y}(x_1) = \mathbf{u}_1,$$

$$U_1 \cdot [\, K(x_1 \leftarrow x_2) \cdot \mathbf{Y}(x_2) + \mathbf{Y}^*(x_1 \leftarrow x_2)\,] = \mathbf{u}_1 \ ,$$

$$[\, U_1 \cdot K(x_1 \leftarrow x_2)\,] \cdot \mathbf{Y}(x_2) = \mathbf{u}_1 - U_1 \cdot \mathbf{Y}^*(x_1 \leftarrow x_2) \ ,$$

Или получаем краевые условия, перенесенные в точку x_2:

$$U_2 \cdot \mathbf{Y}(x_2) = \mathbf{u}_2 \ ,$$

где $U_2 = [\, U_1 \cdot K(x_1 \leftarrow x_2)\,]$ и $\mathbf{u}_2 = \mathbf{u}_1 - U_1 \cdot \mathbf{Y}^*(x_1 \leftarrow x_2)$.

Покажем перенос краевых условий с правого края.

Можно записать:

$$\mathbf{Y}(1) = K(1 \leftarrow x_{n-1}) \cdot \mathbf{Y}(x_{n-1}) + \mathbf{Y}^*(1 \leftarrow x_{n-1}) \ .$$

Подставляем это выражение для $\mathbf{Y}(1)$ в краевые условия правого края и получаем:

$$V \cdot \mathbf{Y}(1) = \mathbf{v},$$

$$V \cdot [\, K(1 \leftarrow x_{n-1}) \cdot \mathbf{Y}(x_{n-1}) + \mathbf{Y}^*(1 \leftarrow x_{n-1})\,] = \mathbf{v},$$

$$[\, V \cdot K(1 \leftarrow x_{n-1})] \cdot \mathbf{Y}(x_{n-1}) = \mathbf{v} - V \cdot \mathbf{Y}^*(1 \leftarrow x_{n-1}).$$

Или получаем краевые условия, перенесенные в точку x_{n-1}:

$$V_{n-1} \cdot \mathbf{Y}(x_{n-1}) = \mathbf{v}_{n-1} \ ,$$

где $V_{n-1} \cdot = [V \cdot K(1 \leftarrow x_{n-1})]$ и $\mathbf{v}_{n-1} = \mathbf{v} - V \cdot \mathbf{Y}^*(1 \leftarrow x_{n-1})$.

Далее запишем аналогично

$$\mathbf{Y}(x_{n-1}) = K(x_{n-1} \leftarrow x_{n-2}) \cdot \mathbf{Y}(x_{n-2}) + \mathbf{Y}^*(x_{n-1} \leftarrow x_{n-2})$$

И подставим это выражение для $\mathbf{Y}(x_{n-1})$ в перенесенные краевые условия точки x_{n-1}

$$V_{n-1} \cdot \mathbf{Y}(x_{n-1}) = \mathbf{v}_{n-1} \ ,$$

$$V_{n-1} \cdot [K(x_{n-1} \leftarrow x_{n-2}) \cdot \mathbf{Y}(x_{n-2}) + \mathbf{Y}^*(x_{n-1} \leftarrow x_{n-2})] = \mathbf{v}_{n-1} \ ,$$

$$[V_{n-1} \cdot K(x_{n-1} \leftarrow x_{n-2})] \cdot \mathbf{Y}(x_{n-2}) = \mathbf{v}_{n-1} - V_{n-1} \cdot \mathbf{Y}^*(x_{n-1} \leftarrow x_{n-2}).$$

Или получаем краевые условия, перенесенные в точку x_{n-2}:

$$V_{n-2} \cdot \mathbf{Y}(x_{n-2}) = \mathbf{v}_{n-2} \ ,$$

где $V_{n-2} \cdot = [V_{n-1} \cdot K(x_{n-1} \leftarrow x_{n-2})]$ и $\mathbf{v}_{n-2} = \mathbf{v}_{n-1} - V_{n-1} \cdot \mathbf{Y}^*(x_{n-1} \leftarrow x_{n-2})$.

И так в точку x_* переносим матричное краевое условие с левого края и таким же образом переносим матричное краевое условие с правого края и получаем:

$$U_* \cdot \mathbf{Y}(x_*) = \mathbf{u}_* \ ,$$
$$V_* \cdot \mathbf{Y}(x_*) = \mathbf{v}_* \ .$$

Из этих двух матричных уравнений с прямоугольными горизонтальными матрицами коэффициентов очевидно получаем одну систему линейных алгебраических уравнений с квадратной матрицей коэффициентов:

$$\left\| \frac{U_*}{V_*} \right\| \cdot \mathbf{Y}(x_*) = \left| \frac{\mathbf{u}_*}{\mathbf{v}_*} \right| .$$

А в случае «жестких» дифференциальных уравнений предлагается применять построчное ортонормирование матричных краевых условий в процессе их переноса в рассматриваемую точку. Для этого формулы ортонормирования систем линейных алгебраических уравнений можно взять в [Березин, Жидков].

То есть, получив

$$U_1 \cdot \mathbf{Y}(x_1) = \mathbf{u}_1,$$

применяем к этой группе линейных алгебраических уравнений построчное ортонормирование и получаем эквивалентное матричное краевое условие:

$$U_{1\,орто} \cdot \mathbf{Y}(x_1) = \mathbf{u}_{1\,орто}.$$

И теперь уже в это проортонормированное построчно уравнение подставляем

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

И получаем

$$U_{1\,орто} \cdot [\, K(x_1 \leftarrow x_2) \cdot \mathbf{Y}(x_2) + \mathbf{Y^*}(x_1 \leftarrow x_2)\,] = \mathbf{u}_{1\,орто} ,$$

$$[\, U_{1\,орто} \cdot K(x_1 \leftarrow x_2)\,] \cdot \mathbf{Y}(x_2) = \mathbf{u}_{1\,орто} - U_{1\,орто} \cdot \mathbf{Y^*}(x_1 \leftarrow x_2) ,$$

Или получаем краевые условия, перенесенные в точку x_2:

$$U_2 \cdot \mathbf{Y}(x_2) = \mathbf{u}_2 ,$$

где $U_2 = [\, U_{1\,орто} \cdot K(x_1 \leftarrow x_2)\,]$ и $\mathbf{u}_2 = \mathbf{u}_{1\,орто} - U_{1\,орто} \cdot \mathbf{Y^*}(x_1 \leftarrow x_2)$.

Теперь уже к этой группе линейных алгебраических уравнений применяем построчное ортонормирование и получаем эквивалентное матричное краевое условие:

$$U_{2\,орто} \cdot \mathbf{Y}(x_2) = \mathbf{u}_{2\,орто}.$$

И так далее.

И аналогично поступаем с промежуточными матричными краевыми условиями, переносимыми с правого края в рассматриваемую точку.

В итоге получаем систему линейных алгебраических уравнений с квадратной матрицей коэффициентов, состоящую из двух независимо друг от друга поэтапно проортонормированных матричных краевых условий. Эта система решается методом Гаусса с выделением главного элемента для получения решения $\mathbf{Y}(x_*)$ в рассматриваемой точке x_*:

$$\left\| \frac{U_{*\,\text{орто}}}{V_{*\,\text{орто}}} \right\| \cdot \mathbf{Y}(x_*) = \left| \frac{\mathbf{u}_{*\,\text{орто}}}{\mathbf{v}_{*\,\text{орто}}} \right|.$$

4.2. Программа на C++ расчета цилиндрической оболочки.

В качестве проверочных задач использовалась схема консольно закрепленных цилиндрической и сферической оболочек с параметрами R/h=50, 100, 200. Длина цилиндрической оболочки рассматривалась L/R=2, а угловые координаты сферической оболочки рассматривались от $\pi/4$ до $3\pi/4$. На свободном крае рассматривалось нормальное к поверхности оболочек погонное усилие, равномерно распределенное в интервале $[-\pi/4, \pi/4]$. В качестве среды программирования использовалась система Microsoft Visual Studio 2010 (Visual C++).

Первоначально метод был предложен и обсчитывался в кандидатской диссертации А.Ю.Виноградова в 1993-1995 годах. Тогда оказалось, что без использования ортонормирования в рамках метода успешно решаются задачи осесимметрично нагруженных оболочек вращения. Расчеты тогда выполнялись на компьютере поколения 286. Задачи же неосесимметричного нагружения оболочек вращения можно было решать на компьютерах поколения 286 только с применением процедур построчного ортонормирования - как это и предлагалось в рамках метода. Без процедур ортонормирования в неосесимметричных случаях выдавались только ошибочные

графики, представлявшие собой хаотично скачущие большие отрицательные и большие положительные значения, например, изгибающего обезразмеренного момента M1.

Современные компьютеры имеют значительно более совершенное внутреннее устройство и более точные внутренние операции с числами, чем это было в 1993-1995 годах. Поэтому было интересно рассмотреть возможность расчета неосесимметрично нагруженных оболочек, например, цилиндров, на современном аппаратном и программном обеспечении в рамках предложенного метода «переноса краевых условий» совсем без использования процедур построчного ортонормирования.

Оказалось, что неосесимметрично нагруженные цилиндры при некоторых параметрах на современных компьютерах уже можно решать в рамках предложенного метода «переноса краевых условий» совсем без применения операций построчного ортонормирования. Это, например, при параметрах цилиндра L/R=2 и R/h=100.

При параметрах цилиндра L/R=2 и R/h=200 все же оказываются необходимыми процедуры ортонормирования. Но на современных персональных компьютерах уже не наблюдаются сплошные скачки значений от больших отрицательных до больших положительных по всему интервалу между краями цилиндра - как это было на компьютерах поколения 286. В частном случае L/R=2 и R/h=200 наблюдаются лишь незначительные скачки в районе максимума изгибающего обезразмеренного момента M1 на левом крае и небольшой скачек обезразмеренного момента M1 на правом крае.

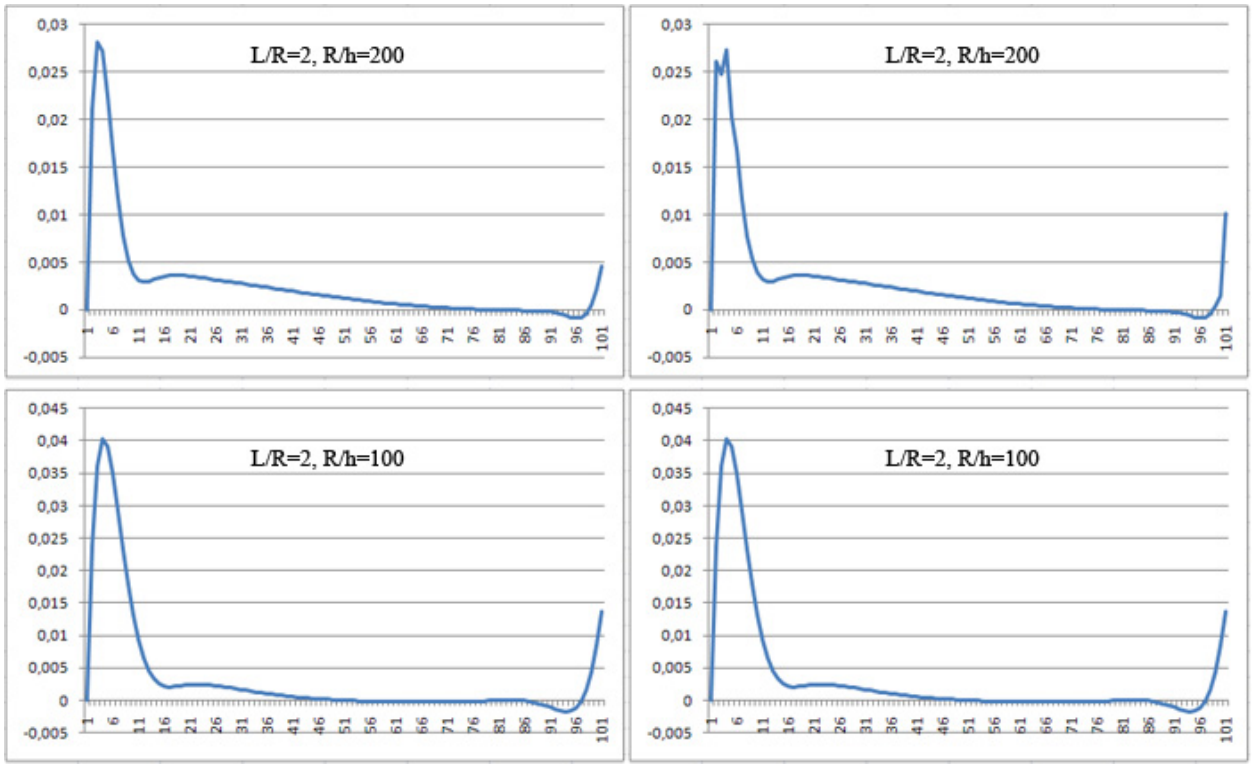
Приводятся графики изгибающего обезразмеренного момента M1:

- слева приводятся графики, полученные при использовании операций построчного ортонормирования на каждом из 100 шагов, на которые разделялся участок между краями цилиндра,

- справа приводятся графики, полученные совсем без применения операций построчного ортонормирования.

Следует сказать, что в качестве расчетной среды использовалась 32-х битная операционная система Windows XP и среда программирования Microsoft Visual Studio 2010 (Visual C++) использовалась в тех же рамках 32-х битной организации операций с числами. Параметры компьютера такие: ноутбук ASUS M51V (CPU Duo T5800).

Компьютеры будут и дальше развиваться такими же темпами как сейчас и это означает, что в самое ближайшее время для подобных расчетов типа расчета неосесимметрично нагруженных оболочек вращения совсем не потребуется применять ортонормирование в рамках предложенного метода «переноса краевых условий», что существенно упрощает программирование метода и увеличивает скорость расчетов не только по сравнению с другими известными методами, но и по сравнению с собственными характеристиками метода «переноса краевых условий» предыдущих лет.



ПРОГРАММА НА C++ (РАСЧЕТ ЦИЛИНДРА):

```
//from_A_Yu_Vinogradov.cpp: главный файл проекта.
//Решение краевой задачи - цилиндрической оболочки.
//Интервал интегрирования разбит на 100 участков: левый край - точка 0 и правый край -
точка 100

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

```
using namespace std;

//Скалярное произведение векторов - i-й строки матрицы A и j-й строки матрицы C.
double mult(double A[8][8], int i, double C[8][8], int j){
	double result=0.0;
	for(int k=0;k<8;k++){
		result+=A[i][k]*C[j][k];
	}
	return result;
}

//Вычисление нормы вектора, где вектор это i-я строка матрицы A.
double norma(double A[8][8], int i){
	double norma_=0.0;
	for(int k=0;k<8;k++){
		norma_+=A[i][k]*A[i][k];
	}
	norma_=sqrt(norma_);
	return norma_;
}

//Выполнение ортонормирования. Исходная система A*x=b размерности 8x8 приводиться к
системе C*x=d, где строки матрицы C ортонормированы.
void orto_norm_8x8(double A[8][8], double b[8], double C[8][8], double d[8]){
	double NORM;
	double mult0,mult1,mult2,mult3,mult4,mult5,mult6,mult7;

	//Получаем 1-ю строку уравнения C*x=d:
	NORM=norma(A,0);
	for(int k=0;k<8;k++){
		C[0][k]=A[0][k]/NORM;
	}
	d[0]=b[0]/NORM;

	//Получаем 2-ю строку уравнения C*x=d:
	mult0=mult(A,1,C,0);
	for(int k=0;k<8;k++){
		C[1][k]=A[1][k]-mult0*C[0][k];
	}
	NORM=norma(C,1);
	for(int k=0;k<8;k++){
		C[1][k]/=NORM;
	}
	d[1]=(b[1]-mult0*d[0])/NORM;

	//Получаем 3-ю строку уравнения C*x=d:
	mult0=mult(A,2,C,0); mult1=mult(A,2,C,1);
	for(int k=0;k<8;k++){
		C[2][k]=A[2][k]-mult0*C[0][k]-mult1*C[1][k];
	}
	NORM=norma(C,2);
	for(int k=0;k<8;k++){
		C[2][k]/=NORM;
	}
	d[2]=(b[2]-mult0*d[0]-mult1*d[1])/NORM;

	//Получаем 4-ю строку уравнения C*x=d:
	mult0=mult(A,3,C,0); mult1=mult(A,3,C,1); mult2=mult(A,3,C,2);
	for(int k=0;k<8;k++){
		C[3][k]=A[3][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k];
	}
	NORM=norma(C,3);
	for(int k=0;k<8;k++){
		C[3][k]/=NORM;
```

```
	}
	d[3]=(b[3]-mult0*d[0]-mult1*d[1]-mult2*d[2])/NORM;

	//Получаем 5-ю строку уравнения C*x=d:
	mult0=mult(A,4,C,0); mult1=mult(A,4,C,1); mult2=mult(A,4,C,2);
mult3=mult(A,4,C,3);
	for(int k=0;k<8;k++){
		C[4][k]=A[4][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k]-
				mult3*C[3][k];
	}
	NORM=norma(C,4);
	for(int k=0;k<8;k++){
		C[4][k]/=NORM;
	}
	d[4]=(b[4]-mult0*d[0]-mult1*d[1]-mult2*d[2]-mult3*d[3])/NORM;

	//Получаем 6-ю строку уравнения C*x=d:
	mult0=mult(A,5,C,0); mult1=mult(A,5,C,1); mult2=mult(A,5,C,2);
mult3=mult(A,5,C,3); mult4=mult(A,5,C,4);
	for(int k=0;k<8;k++){
		C[5][k]=A[5][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k]-
				mult3*C[3][k]-mult4*C[4][k];
	}
	NORM=norma(C,5);
	for(int k=0;k<8;k++){
		C[5][k]/=NORM;
	}
	d[5]=(b[5]-mult0*d[0]-mult1*d[1]-mult2*d[2]-mult3*d[3]-mult4*d[4])/NORM;

	//Получаем 7-ю строку уравнения C*x=d:
	mult0=mult(A,6,C,0); mult1=mult(A,6,C,1); mult2=mult(A,6,C,2);
mult3=mult(A,6,C,3); mult4=mult(A,6,C,4); mult5=mult(A,6,C,5);
	for(int k=0;k<8;k++){
		C[6][k]=A[6][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k]-
				mult3*C[3][k]-mult4*C[4][k]-mult5*C[5][k];
	}
	NORM=norma(C,6);
	for(int k=0;k<8;k++){
		C[6][k]/=NORM;
	}
	d[6]=(b[6]-mult0*d[0]-mult1*d[1]-mult2*d[2]-mult3*d[3]-mult4*d[4]-
		mult5*d[5])/NORM;

	//Получаем 8-ю строку уравнения C*x=d:
	mult0=mult(A,7,C,0); mult1=mult(A,7,C,1); mult2=mult(A,7,C,2);
mult3=mult(A,7,C,3); mult4=mult(A,7,C,4); mult5=mult(A,7,C,5);
	mult6=mult(A,7,C,6);
	for(int k=0;k<8;k++){
		C[7][k]=A[7][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k]-
				mult3*C[3][k]-mult4*C[4][k]-mult5*C[5][k]-mult6*C[6][k];
	}
	NORM=norma(C,7);
	for(int k=0;k<8;k++){
		C[7][k]/=NORM;
	}
	d[7]=(b[7]-mult0*d[0]-mult1*d[1]-mult2*d[2]-mult3*d[3]-mult4*d[4]-
		mult5*d[5]-mult6*d[6])/NORM;

}

//Выполнение ортонормирования системы A*x=b с прямоугольной матрицей A коэффициентов
размерности 4x8.
void orto_norm_4x8(double A[4][8], double b[4], double C[4][8], double d[4]){
	double NORM;
	double mult0,mult1,mult2,mult3,mult4,mult5,mult6,mult7;
```

```
	//Получаем 1-ю строку уравнения C*x=d:
	NORM=norma(A,0);
	for(int k=0;k<8;k++){
		C[0][k]=A[0][k]/NORM;
	}
	d[0]=b[0]/NORM;

	//Получаем 2-ю строку уравнения C*x=d:
	mult0=mult(A,1,C,0);
	for(int k=0;k<8;k++){
		C[1][k]=A[1][k]-mult0*C[0][k];
	}
	NORM=norma(C,1);
	for(int k=0;k<8;k++){
		C[1][k]/=NORM;
	}
	d[1]=(b[1]-mult0*d[0])/NORM;

	//Получаем 3-ю строку уравнения C*x=d:
	mult0=mult(A,2,C,0); mult1=mult(A,2,C,1);
	for(int k=0;k<8;k++){
		C[2][k]=A[2][k]-mult0*C[0][k]-mult1*C[1][k];
	}
	NORM=norma(C,2);
	for(int k=0;k<8;k++){
		C[2][k]/=NORM;
	}
	d[2]=(b[2]-mult0*d[0]-mult1*d[1])/NORM;

	//Получаем 4-ю строку уравнения C*x=d:
	mult0=mult(A,3,C,0); mult1=mult(A,3,C,1); mult2=mult(A,3,C,2);
	for(int k=0;k<8;k++){
		C[3][k]=A[3][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k];
	}
	NORM=norma(C,3);
	for(int k=0;k<8;k++){
		C[3][k]/=NORM;
	}
	d[3]=(b[3]-mult0*d[0]-mult1*d[1]-mult2*d[2])/NORM;
}

//Произведение матрицы A1 размерности 4x8 на матрицу A2 размерности 8x8. Получаем матрицу
rezult размерности 4x8:
void mat_4x8_on_mat_8x8(double A1[4][8], double A2[8][8], double rezult[4][8]){
	for(int i=0;i<4;i++){
		for(int j=0;j<8;j++){
			rezult[i][j]=0.0;
			for(int k=0;k<8;k++){
				rezult[i][j]+=A1[i][k]*A2[k][j];
			}
		}
	}
}

//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];
		}
	}
}
```

```
//Умножение матрицы A размерности 4x8 на вектор b размерности 8 и получаем rezult
размерности 4.
void mat_4x8_on_vect_8(double A[4][8], double b[8], double rezult[4]){
	for(int i=0;i<4;i++){
		rezult[i]=0.0;
		for(int k=0;k<8;k++){
		rezult[i]+=A[i][k]*b[k];
		}
	}
}

//Вычитание из вектора u1 вектора u2 и получение вектора rez=u1-u2. Все вектора
размерности 4.
void minus(double u1[4], double u2[4], double rez[4]){
	for(int i=0;i<4;i++){
		rez[i]=u1[i]-u2[i];
	}
}

//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 8 методом Гаусса с выделением главного элемента
int GAUSS(double AA[8][8], double bb[8], double x[8]){
	double A[8][8];
	double b[8];
	for(int i=0;i<8;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;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<(8-1);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;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;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;i++){//Приведение к верхнетреугольной матрице
                for(int j=(jj+1);j<8;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[8-1]=b[8-1]/A[8-1][8-1];//Первоначальное определение последнего элемента
искомого решения x (7-го)
    for(int i=(8-2);i>=0;i--){//Вычисление елементов решения x[i] от 6-го до 0-го
        t=0;
        for(int j=1;j<(8-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=0.02; //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 MATRIXS[100+1][8][8]={0};//Массив из матриц размерности 8x8 для решения СЛАУ в каждой точке интервала интегрирования
	double VECTORS[100+1][8]={0};//Массив векторов правых частей размерности 8 соответствующих СЛАУ
	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 Ui[4][8]={0};//Вспомогательная матрица коэффициентов переносимых краевых условий с левого края
	double ui_[4]={0};//Правые части переносимых краевых условий с левого края
	double ui_2[4]={0};//вспомогательный вектор (промежуточный)
	double UiORTO[4][8]={0};//Ортонормированная переносимая матрица с левого края
	double ui_ORTO[4]={0};//Соответственно правые части ортонормированного переносимого уравнения с левого края

	double Vj[4][8]={0};//Вспомогательная матрица коэффициентов переносимых краевых условий с правого края
	double vj_[4]={0};//Правые части переносимых краевых условий с правого края
	double vj_2[4]={0};//Вспомогательный вектор (промежуточный)
	double VjORTO[4][8]={0};//Ортонормированная переносимая матрица с правого края
	double vj_ORTO[4]={0};//Соответственно правые части ортонормированного переносимого уравнения с правого края

	double MATRIX_2[8][8]={0};//Вспомогательная матрица
	double VECTOR_2[8]={0};//Вспомогательный вектор
	double Y_2[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_


	orto_norm_4x8(U, u_, UiORTO, ui_ORTO);//Первоначальное ортонормирование краевых условий
	orto_norm_4x8(V, v_, VjORTO, vj_ORTO);

	//Первоначальное заполнение MATRIXS и VECTORS матричными уравнениями краевых условий соответственно
	//UiORTO*Y[0]=ui_ORTO    и    VjORTO*Y[100]=vj_ORTO:
	for(int i=0;i<4;i++){
		for(int j=0;j<8;j++){
			MATRIXS[0][i][j]=UiORTO[i][j];//Левый край; верхнее матричное уравнение
			MATRIXS[100][i+4][j]=VjORTO[i][j];//Правый край (точка номер 101 с индексом 100 - отсчет идет с нуля); нижнее матричное уравнение
		}
		VECTORS[0][i]=ui_ORTO[i];//Левый край; верхнее матричное уравнение
		VECTORS[100][i+4]=vj_ORTO[i];//Правый край (точка номер 101 с индексом 100 - отсчет идет с нуля); нижнее матричное уравнение
	}

	//Цикл по точкам ii интервала интегрирования заполнения ВЕРХНИХ частей матричных уравнений MATRIXS[ii]*Y[ii]=VECTORS[ii],
	//начиная со второй точки - точки с индексом ii=1
	exponent(A,(-step),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);
	for(int ii=1;ii<=100;ii++){
		x+=step;//Координата для расчета частного вектора на шаге
		mat_4x8_on_mat_8x8(UiORTO,expo_from_minus_step,Ui);//Вычисление матрицы Ui=UiORTO*expo_from_minus_step
		//partial_vector(FF);//Вычисление НУЛЕВОГО вектора частного решения системы ОДУ на шаге
		partial_vector_real(expo_from_minus_step, mat_row_for_minus_expo, x, (-step),FF);// - для движения слева на право
		mat_4x8_on_vect_8(UiORTO,FF,ui_2);//Вычисление вектора ui_2=UiORTO*FF
		minus(ui_ORTO, ui_2, ui_);//Вычисление вектора ui_=ui_ORTO-ui_2
		orto_norm_4x8(Ui, ui_, UiORTO, ui_ORTO);//Ортонормирование для текущего шага по ii
		for(int i=0;i<4;i++){
			for(int j=0;j<8;j++){
				MATRIXS[ii][i][j]=UiORTO[i][j];
			}
			VECTORS[ii][i]=ui_ORTO[i];
		}
	}//Цикл по шагам ii (ВЕРХНЕЕ заполнение)

	//Цикл по точкам ii интервала интегрирования заполнения НИЖНИХ частей матричных уравнений MATRIXS[ii]*Y[ii]=VECTORS[ii],
```

```
	//начиная с предпоследней точки - точки с индексом ii=(100-1) используем ii--
(уменьшение индекса точки)
	exponent(A,step,expo_from_plus_step);//Шаг положительный (значение шага больше
нуля из-за направления вычисления матричной экспоненты)
	x=step*100;//Координата правого края
	mat_row_for_partial_vector(A, (-step), mat_row_for_plus_expo);
	for(int ii=(100-1);ii>=0;ii--){
		x-=step;//Движение справа на лево - для расчета частного вектора
		mat_4x8_on_mat_8x8(VjORTO,expo_from_plus_step,Vj);//Вычисление матрицы
Vj=VjORTO*expo_from_plus_step
		//partial_vector(FF);//Вычисление НУЛЕВОГО вектора частного решения системы
ОДУ на шаге
		partial_vector_real(expo_from_plus_step,  mat_row_for_plus_expo, x,
step,FF);// - для движения справа на лево
		mat_4x8_on_vect_8(VjORTO,FF,vj_2);//Вычисление вектора vj_2=VjORTO*FF
		minus(vj_ORTO, vj_2, vj_);//Вычисление вектора vj_=vj_ORTO-vj_2
		orto_norm_4x8(Vj, vj_, VjORTO, vj_ORTO);//Ортонормирование для текущего
шага по ii
		for(int i=0;i<4;i++){
			for(int j=0;j<8;j++){
				MATRIXS[ii][i+4][j]=VjORTO[i][j];
			}
			VECTORS[ii][i+4]=vj_ORTO[i];
		}
	}//Цикл по шагам ii (НИЖНЕЕ заполнение)

	//Решение систем линейных алгебраических уравнений
	for(int ii=0;ii<=100;ii++){
		for(int i=0;i<8;i++){
			for(int j=0;j<8;j++){
				MATRIX_2[i][j]=MATRIXS[ii][i][j];//Вспомогательное присвоение
для соответствия типов в вызывающей функции GAUSS
			}
			VECTOR_2[i]=VECTORS[ii][i];//Вспомогательное присвоение для
соответствия типов в вызывающей функции GAUSS
		}

		GAUSS(MATRIX_2,VECTOR_2,Y_2);

		for(int i=0;i<8;i++){
			Y[ii][i]=Y_2[i];
		}
	}

	//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;
}
```

4.3. Программа на C++ расчета сферической оболочки (переменные коэффициенты).

ПРОГРАММА НА C++ (РАСЧЕТ СФЕРЫ):

```
//sfera_from_A_Yu_Vinogradov.cpp: главный файл проекта.
//Решение краевой задачи с переменными коэффициентами - сфера.
//Интервал интегрирования разбит на 100 участков: левый край - точка 0 и правый край -
точка 100

#include "stdafx.h"
#include <iostream>
#include <conio.h>
#include <math.h> //for tan()


using namespace std;

//Вычисление для гармоники с номером nn для значения переменной (угла) angle_fi - матрицы
A_perem 8x8 коэффициентов системы ОДУ
void A_perem_coef(double nju, double c2, int nn, double angle_fi, double A_perem[8][8]){
	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;

	for(int i=0;i<8;i++){
		for(int j=0;j<8;j++){
			A_perem[i][j]=0.0;//Первоначальное обнуление матрицы
		}
	}

	//Заполнение ненулевых элементов матрицы A коэффициентов системы ОДУ
	A_perem[0][1]=1.0;
	A_perem[1][0]=(1-
nju)*nn2/2/sin(angle_fi)/sin(angle_fi)+nju+1.0/tan(angle_fi)/tan(angle_fi);
	A_perem[1][1]=-1.0/tan(angle_fi);
	A_perem[1][2]=(3-nju)/2/sin(angle_fi)/tan(angle_fi);
	A_perem[1][3]=-(1+nju)*nn/2/sin(angle_fi);
	A_perem[1][5]=-(1+nju);
	A_perem[2][3]=1.0;
	A_perem[3][0]=(3-nju)*nn/(1-nju)/sin(angle_fi)/tan(angle_fi);
	A_perem[3][1]=(1+nju)*nn/(1-nju)/sin(angle_fi);
	A_perem[3][2]=2*nn2/(1-nju)/sin(angle_fi)/sin(angle_fi)-
1.0+1.0/tan(angle_fi)/tan(angle_fi);
	A_perem[3][3]=-1.0/tan(angle_fi);
	A_perem[3][4]=(1+nju)*2*nn/(1-nju)/sin(angle_fi);
	A_perem[4][5]=1.0;
	A_perem[5][6]=1.0;
	A_perem[6][7]=1.0;
	A_perem[7][0]=-(1+nju)/tan(angle_fi)/c2;
	A_perem[7][1]=-(1+nju)/c2;
	A_perem[7][2]=-(1+nju)*nn/c2/sin(angle_fi);
	A_perem[7][4]=nn2/sin(angle_fi)/sin(angle_fi)*(2+(2-
nn2)/sin(angle_fi)/sin(angle_fi)+2.0/tan(angle_fi)/tan(angle_fi))-2*(1+nju)/c2;
	A_perem[7][5]=(-2.0-(2*nn2+1)/sin(angle_fi)/sin(angle_fi))/tan(angle_fi);
	A_perem[7][6]=-1.0+(2*nn2+1)/sin(angle_fi)/sin(angle_fi);
```

```
	A_perem[7][7]=-2.0/tan(angle_fi);

}

//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;
}

//Скалярное произведение векторов - i-й строки матрицы A и j-й строки матрицы C.
double mult(double A[8][8], int i, double C[8][8], int j){
	double result=0.0;
	for(int k=0;k<8;k++){
		result+=A[i][k]*C[j][k];
	}
	return result;
}

//Вычисление нормы вектора, где вектор это i-я строка матрицы A.
double norma(double A[8][8], int i){
	double norma_=0.0;
	for(int k=0;k<8;k++){
		norma_+=A[i][k]*A[i][k];
	}
	norma_=sqrt(norma_);
	return norma_;
}

//Выполнение ортонормирования. Исходная система A*x=b размерности 8x8 приводиться к
системе C*x=d, где строки матрицы C ортонормированы.
void orto_norm_8x8(double A[8][8], double b[8], double C[8][8], double d[8]){
	double NORM;
	double mult0,mult1,mult2,mult3,mult4,mult5,mult6,mult7;

	//Получаем 1-ю строку уравнения C*x=d:
	NORM=norma(A,0);
	for(int k=0;k<8;k++){
		C[0][k]=A[0][k]/NORM;
	}
	d[0]=b[0]/NORM;

	//Получаем 2-ю строку уравнения C*x=d:
	mult0=mult(A,1,C,0);
	for(int k=0;k<8;k++){
		C[1][k]=A[1][k]-mult0*C[0][k];
	}
	NORM=norma(C,1);
	for(int k=0;k<8;k++){
		C[1][k]/=NORM;
	}
	d[1]=(b[1]-mult0*d[0])/NORM;

	//Получаем 3-ю строку уравнения C*x=d:
	mult0=mult(A,2,C,0); mult1=mult(A,2,C,1);
	for(int k=0;k<8;k++){
		C[2][k]=A[2][k]-mult0*C[0][k]-mult1*C[1][k];
	}
```

```
	NORM=norma(C,2);
	for(int k=0;k<8;k++){
		C[2][k]/=NORM;
	}
	d[2]=(b[2]-mult0*d[0]-mult1*d[1])/NORM;

	//Получаем 4-ю строку уравнения C*x=d:
	mult0=mult(A,3,C,0); mult1=mult(A,3,C,1); mult2=mult(A,3,C,2);
	for(int k=0;k<8;k++){
		C[3][k]=A[3][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k];
	}
	NORM=norma(C,3);
	for(int k=0;k<8;k++){
		C[3][k]/=NORM;
	}
	d[3]=(b[3]-mult0*d[0]-mult1*d[1]-mult2*d[2])/NORM;

	//Получаем 5-ю строку уравнения C*x=d:
	mult0=mult(A,4,C,0); mult1=mult(A,4,C,1); mult2=mult(A,4,C,2);
mult3=mult(A,4,C,3);
	for(int k=0;k<8;k++){
		C[4][k]=A[4][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k]-
				mult3*C[3][k];
	}
	NORM=norma(C,4);
	for(int k=0;k<8;k++){
		C[4][k]/=NORM;
	}
	d[4]=(b[4]-mult0*d[0]-mult1*d[1]-mult2*d[2]-mult3*d[3])/NORM;

	//Получаем 6-ю строку уравнения C*x=d:
	mult0=mult(A,5,C,0); mult1=mult(A,5,C,1); mult2=mult(A,5,C,2);
mult3=mult(A,5,C,3); mult4=mult(A,5,C,4);
	for(int k=0;k<8;k++){
		C[5][k]=A[5][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k]-
				mult3*C[3][k]-mult4*C[4][k];
	}
	NORM=norma(C,5);
	for(int k=0;k<8;k++){
		C[5][k]/=NORM;
	}
	d[5]=(b[5]-mult0*d[0]-mult1*d[1]-mult2*d[2]-mult3*d[3]-mult4*d[4])/NORM;

	//Получаем 7-ю строку уравнения C*x=d:
	mult0=mult(A,6,C,0); mult1=mult(A,6,C,1); mult2=mult(A,6,C,2);
mult3=mult(A,6,C,3); mult4=mult(A,6,C,4); mult5=mult(A,6,C,5);
	for(int k=0;k<8;k++){
		C[6][k]=A[6][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k]-
				mult3*C[3][k]-mult4*C[4][k]-mult5*C[5][k];
	}
	NORM=norma(C,6);
	for(int k=0;k<8;k++){
		C[6][k]/=NORM;
	}
	d[6]=(b[6]-mult0*d[0]-mult1*d[1]-mult2*d[2]-mult3*d[3]-mult4*d[4]-
		mult5*d[5])/NORM;

	//Получаем 8-ю строку уравнения C*x=d:
	mult0=mult(A,7,C,0); mult1=mult(A,7,C,1); mult2=mult(A,7,C,2);
mult3=mult(A,7,C,3); mult4=mult(A,7,C,4); mult5=mult(A,7,C,5);
	mult6=mult(A,7,C,6);
	for(int k=0;k<8;k++){
		C[7][k]=A[7][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k]-
				mult3*C[3][k]-mult4*C[4][k]-mult5*C[5][k]-mult6*C[6][k];
	}
```

```
	NORM=norma(C,7);
	for(int k=0;k<8;k++){
		C[7][k]/=NORM;
	}
	d[7]=(b[7]-mult0*d[0]-mult1*d[1]-mult2*d[2]-mult3*d[3]-mult4*d[4]-
		mult5*d[5]-mult6*d[6])/NORM;

}

//Выполнение ортонормирования системы A*x=b с прямоугольной матрицей A коэффициентов
размерности 4x8.
void orto_norm_4x8(double A[4][8], double b[4], double C[4][8], double d[4]){
	double NORM;
	double mult0,mult1,mult2,mult3,mult4,mult5,mult6,mult7;

	//Получаем 1-ю строку уравнения C*x=d:
	NORM=norma(A,0);
	for(int k=0;k<8;k++){
		C[0][k]=A[0][k]/NORM;
	}
	d[0]=b[0]/NORM;

	//Получаем 2-ю строку уравнения C*x=d:
	mult0=mult(A,1,C,0);
	for(int k=0;k<8;k++){
		C[1][k]=A[1][k]-mult0*C[0][k];
	}
	NORM=norma(C,1);
	for(int k=0;k<8;k++){
		C[1][k]/=NORM;
	}
	d[1]=(b[1]-mult0*d[0])/NORM;

	//Получаем 3-ю строку уравнения C*x=d:
	mult0=mult(A,2,C,0); mult1=mult(A,2,C,1);
	for(int k=0;k<8;k++){
		C[2][k]=A[2][k]-mult0*C[0][k]-mult1*C[1][k];
	}
	NORM=norma(C,2);
	for(int k=0;k<8;k++){
		C[2][k]/=NORM;
	}
	d[2]=(b[2]-mult0*d[0]-mult1*d[1])/NORM;

	//Получаем 4-ю строку уравнения C*x=d:
	mult0=mult(A,3,C,0); mult1=mult(A,3,C,1); mult2=mult(A,3,C,2);
	for(int k=0;k<8;k++){
		C[3][k]=A[3][k]-mult0*C[0][k]-mult1*C[1][k]-mult2*C[2][k];
	}
	NORM=norma(C,3);
	for(int k=0;k<8;k++){
		C[3][k]/=NORM;
	}
	d[3]=(b[3]-mult0*d[0]-mult1*d[1]-mult2*d[2])/NORM;
}

//Произведение матрицы A1 размерности 4x8 на матрицу A2 размерности 8x8. Получаем матрицу
rezult размерности 4x8:
void mat_4x8_on_mat_8x8(double A1[4][8], double A2[8][8], double rezult[4][8]){
	for(int i=0;i<4;i++){
		for(int j=0;j<8;j++){
			rezult[i][j]=0.0;
			for(int k=0;k<8;k++){
				rezult[i][j]+=A1[i][k]*A2[k][j];
			}
```

```
		}
	}
}

//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];
		}
	}
}

//Умножение матрицы A размерности 4x8 на вектор b размерности 8 и получаем rezult
размерности 4.
void mat_4x8_on_vect_8(double A[4][8], double b[8], double rezult[4]){
	for(int i=0;i<4;i++){
		rezult[i]=0.0;
		for(int k=0;k<8;k++){
		rezult[i]+=A[i][k]*b[k];
		}
	}
}

//Вычитание из вектора u1 вектора u2 и получение вектора rez=u1-u2. Все вектора
размерности 4.
void minus(double u1[4], double u2[4], double rez[4]){
	for(int i=0;i<4;i++){
		rez[i]=u1[i]-u2[i];
	}
}

//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];
                                }
                        }
                }
        }
        }

//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 8 методом Гаусса с выделением главного элемента
int GAUSS(double AA[8][8], double bb[8], double x[8]){
        double A[8][8];
        double b[8];
        for(int i=0;i<8;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;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<(8-1);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;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;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;i++){//Приведение к верхнетреугольной матрице
				for(int j=(jj+1);j<8;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[8-1]=b[8-1]/A[8-1][8-1];//Первоначальное определение последнего элемента
искомого решения x (7-го)
		for(int i=(8-2);i>=0;i--){//Вычисление елементов решения x[i] от 6-го до 0-го
			t=0;
			for(int j=1;j<(8-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 angle;
		double start_angle, finish_angle;
		start_angle=3.14159265359/4;
		finish_angle=start_angle+(3.14159265359/2);
		double step=(3.14159265359/2)/100; //step=(3.14159265359/2)/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=1.0/200;
		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 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 MATRIXS[100+1][8][8]={0};//Массив из матриц размерности 8x8 для решения
СЛАУ в каждой точке интервала интегрирования
        double VECTORS[100+1][8]={0};//Массив векторов правых частей размерности 8
соответствующих СЛАУ
        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 Ui[4][8]={0};//Вспомогательная матрица коэффициентов переносимых краевых
условий с левого края
        double ui_[4]={0};//Правые части переносимых краевых условий с левого края
        double ui_2[4]={0};//вспомогательный вектор (промежуточный)
        double UiORTO[4][8]={0};//Ортонормированная переносимая матрица с левого края
        double ui_ORTO[4]={0};//Соответственно правые части ортонормированного
переносимого уравнения с левого края

        double Vj[4][8]={0};//Вспомогательная матрица коэффициентов переносимых краевых
условий с правого края
        double vj_[4]={0};//Правые части переносимых краевых условий с правого края
        double vj_2[4]={0};//Вспомогательный вектор (промежуточный)
        double VjORTO[4][8]={0};//Ортонормированная переносимая матрица с правого края
        double vj_ORTO[4]={0};//Соответственно правые части ортонормированного
переносимого уравнения с правого края

        double MATRIX_2[8][8]={0};//Вспомогательная матрица
        double VECTOR_2[8]={0};//Вспомогательный вектор
        double Y_2[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;


        //Здесь надо первоначально заполнить ненулевыми значениями матрицы и вектора
краевых условий U*Y[0]=u_ (слева) и V*Y[100]=v_ (справа) :
        U[0][0]=nju/tan(start_angle);
        U[0][1]=1.0;
        U[0][2]=nju*nn/sin(start_angle);
        U[0][4]=(1+nju);
        u_[0]=0.0;//Сила T1 на левом крае равна нулю
```

```
	U[1][0]=-(1-nju)/2/sin(start_angle);
	U[1][2]=-(1-nju)/2/tan(start_angle);
	U[1][3]=(1-nju)/2;
	U[1][4]=-c2*nn*(1-nju)/sin(start_angle)/tan(start_angle);
	U[1][5]=c2*nn*(1-nju)/sin(start_angle);
	u_[1]=0.0;//Сила S* на левом краю равна нулю

	U[2][4]=-nju*nn2/sin(start_angle)/sin(start_angle);
	U[2][5]=nju/tan(start_angle);
	U[2][6]=1.0;
	u_[2]=0;//Момент M1 на левом краю равен нулю

	U[3][4]=-(3-nju)*nn2/sin(start_angle)/sin(start_angle)/tan(start_angle);
	U[3][5]=nju+1.0/tan(start_angle)/tan(start_angle)-(nju-
2)*nn2/sin(start_angle)/sin(start_angle);
	U[3][6]=-1.0/tan(start_angle);
	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_


	orto_norm_4x8(U, u_, UiORTO, ui_ORTO);//Первоначальное ортонормирование краевых
условий
	orto_norm_4x8(V, v_, VjORTO, vj_ORTO);

	//Первоначальное заполнение MATRIXS и VECTORS матричными уравнениями краевых
условий соответственно
	//UiORTO*Y[0]=ui_ORTO    и    VjORTO*Y[100]=vj_ORTO:
	for(int i=0;i<4;i++){
		for(int j=0;j<8;j++){
			MATRIXS[0][i][j]=UiORTO[i][j];//Левый край; верхнее матричное
уравнение
			MATRIXS[100][i+4][j]=VjORTO[i][j];//Правый край (точка номер 101 с
индексом 100 - отсчет идет с нуля); нижнее матричное уравнение
		}
		VECTORS[0][i]=ui_ORTO[i];//Левый край; верхнее матричное уравнение
		VECTORS[100][i+4]=vj_ORTO[i];//Правый край (точка номер 101 с индексом 100
- отсчет идет с нуля); нижнее матричное уравнение
	}

	//Цикл по точкам ii интервала интегрирования заполнения ВЕРХНИХ частей матричных
уравнений MATRIXS[ii]*Y[ii]=VECTORS[ii],
	//начиная со второй точки - точки с индексом ii=1
	angle=start_angle;//начальное значение угловой координаты
	for(int ii=1;ii<=100;ii++){

		angle+=step;//Угловая координата
		A_perem_coef(nju, c2, nn, angle, A);//Вычисление матрицы A коэффициентов
системы ОДУ при данной угловой координате angle
		exponent(A,(-step),expo_from_minus_step);//Шаг отрицательный (значение шага
меньше нуля из-за направления вычисления матричной экспоненты)
		mat_row_for_partial_vector(A, step, mat_row_for_minus_expo);

		mat_4x8_on_mat_8x8(UiORTO,expo_from_minus_step,Ui);//Вычисление матрицы
Ui=UiORTO*expo_from_minus_step
		//partial_vector(FF);//Вычисление НУЛЕВОГО вектора частного решения системы
ОДУ на шаге
```

```
		partial_vector_real(expo_from_minus_step, mat_row_for_minus_expo, angle, (-
step),FF);// - для движения слева на право
		mat_4x8_on_vect_8(UiORTO,FF,ui_2);//Вычисление вектора ui_2=UiORTO*FF
		minus(ui_ORTO, ui_2, ui_);//Вычисление вектора ui_=ui_ORTO-ui_2
		orto_norm_4x8(Ui, ui_, UiORTO, ui_ORTO);//Ортонормирование для текущего
шага по ii
		for(int i=0;i<4;i++){
			for(int j=0;j<8;j++){
				MATRIXS[ii][i][j]=UiORTO[i][j];
			}
			VECTORS[ii][i]=ui_ORTO[i];
		}
	}//Цикл по шагам ii (ВЕРХНЕЕ заполнение)

	//Цикл по точкам ii интервала интегрирования заполнения НИЖНИХ частей матричных
уравнений MATRIXS[ii]*Y[ii]=VECTORS[ii],
	//начиная с предпоследней точки - точки с индексом ii=(100-1) используем ii--
(уменьшение индекса точки)
	angle=finish_angle;//Угловая координата правого края
	for(int ii=(100-1);ii>=0;ii--){

		angle-=step;//Движение справа на лево
		A_perem_coef(nju, c2, nn, angle, A);//Вычисление матрицы A коэффициентов
системы ОДУ при данной угловой координате angle
		exponent(A,step,expo_from_plus_step);//Шаг положительный (значение шага
больше нуля из-за направления вычисления матричной экспоненты)
		mat_row_for_partial_vector(A, (-step), mat_row_for_plus_expo);

		mat_4x8_on_mat_8x8(VjORTO,expo_from_plus_step,Vj);//Вычисление матрицы
Vj=VjORTO*expo_from_plus_step
		//partial_vector(FF);//Вычисление НУЛЕВОГО вектора частного решения системы
ОДУ на шаге
		partial_vector_real(expo_from_plus_step,  mat_row_for_plus_expo, angle,
step,FF);// - для движения справа на лево
		mat_4x8_on_vect_8(VjORTO,FF,vj_2);//Вычисление вектора vj_2=VjORTO*FF
		minus(vj_ORTO, vj_2, vj_);//Вычисление вектора vj_=vj_ORTO-vj_2
		orto_norm_4x8(Vj, vj_, VjORTO, vj_ORTO);//Ортонормирование для текущего
шага по ii
		for(int i=0;i<4;i++){
			for(int j=0;j<8;j++){
				MATRIXS[ii][i+4][j]=VjORTO[i][j];
			}
			VECTORS[ii][i+4]=vj_ORTO[i];
		}
	}//Цикл по шагам ii (НИЖНЕЕ заполнение)

	//Решение систем линейных алгебраических уравнений
	for(int ii=0;ii<=100;ii++){
		for(int i=0;i<8;i++){
			for(int j=0;j<8;j++){
				MATRIX_2[i][j]=MATRIXS[ii][i][j];//Вспомогательное присвоение
для соответствия типов в вызывающей функции GAUSS
			}
			VECTOR_2[i]=VECTORS[ii][i];//Вспомогательное присвоение для
соответствия типов в вызывающей функции GAUSS
		}

		GAUSS(MATRIX_2,VECTOR_2,Y_2);

		for(int i=0;i<8;i++){
			Y[ii][i]=Y_2[i];
		}
	}

	//Вычисление момента во всех точках между краями
```

```
	angle=start_angle;//начальное значение угловой координаты
	for(int ii=0;ii<=100;ii++){
		Moment[ii]+=Y[ii][4]*(-
nju*nn2/sin(angle)/sin(angle))+Y[ii][5]*(nju/tan(angle))+Y[ii][6]*(1.0);//Момент M1 в
точке [ii]
		angle+=step;
		//U[2][4]=-nju*nn2/sin(start_angle)/sin(start_angle);
		//U[2][5]=nju/tan(start_angle);
		//U[2][6]=1.0; Момент

	}



	}//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;
}
```

5. Второй вариант метода «переноса краевых условий» в произвольную точку интервала интегрирования.

Этот вариант метода еще не обсчитан на компьютерах.

Предложено выполнять интегрирование по формулам теории матриц [Гантмахер] сразу от некоторой внутренней точки интервала интегрирования к краям:

$$\mathbf{Y}(0) = K(0 \leftarrow x) \cdot \mathbf{Y}(x) + \mathbf{Y}^*(0 \leftarrow x) \ ,$$
$$\mathbf{Y}(1) = K(1 \leftarrow x) \cdot \mathbf{Y}(x) + \mathbf{Y}^*(1 \leftarrow x) \ .$$

Подставим эти формулы в краевые условия и получим:

$$U \cdot \mathbf{Y}(0) = \mathbf{u},$$

$$U \cdot [\, K(0 \leftarrow x) \cdot \mathbf{Y}(x) \ + \ \mathbf{Y}^*(0 \leftarrow x) \,] = \mathbf{u},$$

$$[\, U\cdot K(0\leftarrow x)\,]\cdot \mathbf{Y}(x) = \mathbf{u} - U\cdot \mathbf{Y}^{*}(0\leftarrow x)\ .$$

и

$$V\cdot \mathbf{Y}(1) = \mathbf{v},$$

$$V\cdot[\, K(1\leftarrow x)\cdot \mathbf{Y}(x) + \mathbf{Y}^{*}(1\leftarrow x)\,] = \mathbf{v},$$

$$[\, V\cdot K(1\leftarrow x)\,]\cdot \mathbf{Y}(x) = \mathbf{v} - V\cdot \mathbf{Y}^{*}(1\leftarrow x)\ .$$

То есть получаем два матричных уравнения краевых условий, перенесенные в рассматриваемую точку x:

$$[\, U\cdot K(0\leftarrow x)\,]\cdot \mathbf{Y}(x) = \mathbf{u} - U\cdot \mathbf{Y}^{*}(0\leftarrow x)\ ,$$
$$[\, V\cdot K(1\leftarrow x)\,]\cdot \mathbf{Y}(x) = \mathbf{v} - V\cdot \mathbf{Y}^{*}(1\leftarrow x)\ .$$

Эти уравнения аналогично объединяются в одну систему линейных алгебраических уравнений с квадратной матрицей коэффициентов для нахождения решения $\mathbf{Y}(x)$ в любой рассматриваемой точке x:

$$\left\| \frac{U\cdot K(0\leftarrow x)}{V\cdot K(1\leftarrow x)} \right\| \cdot \mathbf{Y}(x) = \left| \frac{\mathbf{u} - U\cdot \mathbf{Y}^{*}(0\leftarrow x)}{\mathbf{v} - V\cdot \mathbf{Y}^{*}(1\leftarrow x)} \right|.$$

В случае «жестких» дифференциальных уравнений предлагается следующий алгоритм.

Используем свойство перемножаемости матриц Коши:

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

и запишем выражения для матриц Коши, например, в виде:

$$K(0\leftarrow x) = K(0\leftarrow x_{2L})\cdot K(x_{2L}\leftarrow x_{1L})\cdot K(x_{1L}\leftarrow x),$$
$$K(1\leftarrow x) = K(1\leftarrow x_{3R})\cdot K(x_{3R}\leftarrow x_{2R})\cdot K(x_{2R}\leftarrow x_{1R})\cdot K(x_{1R}\leftarrow x),$$

Тогда перенесенные краевые условия можно записать в виде:

$$[\, U\cdot K(0\leftarrow x_{2L}) \cdot K(x_{2L}\leftarrow x_{1L}) \cdot K(x_{1L}\leftarrow x)\,] \cdot \mathbf{Y}(x) = \mathbf{u} - U\cdot\mathbf{Y}^*(0\leftarrow x)\ ,$$
$$[\, V\cdot K(1\leftarrow x_{3R}) \cdot K(x_{3R}\leftarrow x_{2R}) \cdot K(x_{2R}\leftarrow x_{1R}) \cdot K(x_{1R}\leftarrow x)\,] \cdot \mathbf{Y}(x) = \mathbf{v} - V\cdot\mathbf{Y}^*(1\leftarrow x)$$

или в виде:

$$[\, U\cdot K(0\leftarrow x_{2L}) \cdot K(x_{2L}\leftarrow x_{1L}) \cdot K(x_{1L}\leftarrow x)\,] \cdot \mathbf{Y}(x) = \mathbf{u}^*\ ,$$
$$[\, V\cdot K(1\leftarrow x_{3R}) \cdot K(x_{3R}\leftarrow x_{2R}) \cdot K(x_{2R}\leftarrow x_{1R}) \cdot K(x_{1R}\leftarrow x)\,] \cdot \mathbf{Y}(x) = \mathbf{v}^*\ .$$

Тогда рассмотрим левое перенесенное краевое условие:

$$[\, U\cdot K(0\leftarrow x_{2L}) \cdot K(x_{2L}\leftarrow x_{1L}) \cdot K(x_{1L}\leftarrow x)\,] \cdot \mathbf{Y}(x) = \mathbf{u}^*\ ,$$

$$[\, U\cdot K(0\leftarrow x_{2L})\,] \cdot \{\, K(x_{2L}\leftarrow x_{1L}) \cdot K(x_{1L}\leftarrow x) \cdot \mathbf{Y}(x)\,\} = \mathbf{u}^*\ ,$$
$$[\quad \text{матрица} \quad] \cdot \{\quad\quad \text{вектор} \quad\quad\} = \mathbf{вектор}\ .$$

Эту группу линейных алгебраических уравнений можно подвергнуть построчному ортонормированию, которое сделает строчки [матрицы] ортонормированными, {вектор} затронут не будет, а **вектор** получит преобразование. То есть получим:

$$[\, U\cdot K(0\leftarrow x_{2L})\,]_{\text{орто}} \cdot \{\, K(x_{2L}\leftarrow x_{1L}) \cdot K(x_{1L}\leftarrow x) \cdot \mathbf{Y}(x)\,\} = \mathbf{u}^*{}_{\text{орто}}\ .$$

Далее последовательно можно записать:

$$[[\, U\cdot K(0\leftarrow x_{2L})\,]_{\text{орто}} \cdot K(x_{2L}\leftarrow x_{1L})\,] \cdot \{\, K(x_{1L}\leftarrow x) \cdot \mathbf{Y}(x)\,\} = \mathbf{u}^*{}_{\text{орто}}\ ,$$
$$[\quad\quad \text{матрица} \quad\quad] \cdot \{\quad \text{вектор} \quad\} = \mathbf{вектор}\ .$$

Аналогично и эту группу линейных алгебраических уравнений можно подвергнуть построчному ортонормированию, которое сделает строчки [матрицы] ортонормированными, {вектор} затронут не будет, а **вектор** получит преобразование. То есть получим:

$$[[\,U\cdot K(0\leftarrow x_{2L})\,]_{орто}\cdot K(x_{2L}\leftarrow x_{1L})\,]_{орто}\cdot\{K(x_{1L}\leftarrow x)\cdot \mathbf{Y}(x)\}=\mathbf{u}^*_{2\,орто}\ ,$$

Далее аналогично можно записать:

$$[[[\,U\cdot K(0\leftarrow x_{2L})\,]_{орто}\cdot K(x_{2L}\leftarrow x_{1L})\,]_{орто}\cdot K(x_{1L}\leftarrow x)\,]\cdot\{\ \mathbf{Y}(x)\ \}=\mathbf{u}^*_{2\,орто}\ ,$$

$$[\qquad\qquad\text{матрица}\qquad\qquad\qquad]\cdot\{\text{вектор}\}=\mathbf{вектор}\ .$$

Аналогично и эту группу линейных алгебраических уравнений можно подвергнуть построчному ортонормированию, которое сделает строчки [матрицы] ортонормированными, {вектор} затронут не будет, а **вектор** получит преобразование. То есть получим:

$$[[[\,U\cdot K(0\leftarrow x_{2L})\,]_{орто}\cdot K(x_{2L}\leftarrow x_{1L})\,]_{орто}\cdot K(x_{1L}\leftarrow x)\,]_{орто}\cdot \mathbf{Y}(x)=\mathbf{u}^*_{3\,орто}\ .$$

Аналогично можно проортонормировать матричное уравнение краевых условий и для правого края независимо от левого края.

Далее проортонормированные уравнения краевых условий:

$$[\,U\cdot K(0\leftarrow x)\,]_{3\,орто}\cdot \mathbf{Y}(x)=\mathbf{u}^*_{3\,орто}\ ,$$

$$[\,V\cdot K(1\leftarrow x)\,]_{4\,орто}\cdot \mathbf{Y}(x)=\mathbf{v}^*_{4\,орто}$$

как и ранее объединяются в одну обычную систему линейных алгебраических уравнений с квадратной матрицей коэффициентов для нахождения искомого вектора $\mathbf{Y}(x)$:

$$\left\|\frac{[U\cdot K(0\leftarrow x)]_{3\,орто}}{[V\cdot K(1\leftarrow x)]_{4\,орто}}\right\|\cdot\mathbf{Y}(x)=\left|\frac{\mathbf{u}^*_{3\,орто}}{\mathbf{v}^*_{4\,орто}}\right|.$$

6. Метод дополнительных краевых условий.

Этот метод еще не обсчитан на компьютерах.

Запишем на левом крае ещё одно уравнение краевых условий:

$$M \cdot \mathbf{Y}(0) = \mathbf{m} .$$

В качестве строк матрицы M можно взять те краевые условия, то есть выражения тех физических параметров, которые не входят в параметры краевых условий левого края L или линейно независимы с ними. Это вполне возможно, так как у краевых задач столько независимых физических параметров какова размерность задачи, а в параметры краевых условий входит только половина физических параметров задачи. То есть, например, если рассматривается задача об оболочке ракеты, то на левом крае могут быть заданы 4 перемещения. Тогда для матрицы M можно взять параметры сил и моментов, которых тоже 4, так как полная размерность такой задачи – 8. Вектор **m** правой части неизвестен и его надо найти и тогда можно считать, что краевая задача решена, то есть сведена к задаче Коши, то есть найден вектор $\mathbf{Y}(0)$ из выражения:

$$\left\| \frac{U}{M} \right\| \cdot \mathbf{Y}(0) = \left| \frac{\mathbf{u}}{\mathbf{m}} \right|,$$

то есть вектор $\mathbf{Y}(0)$ находится из решения системы линейных алгебраических уравнений с квадратной невырожденной матрицей коэффициентов, состоящей из блоков U и M.

Аналогично запишем на правом крае ещё одно уравнение краевых условий:

$$N \cdot \mathbf{Y}(0) = \mathbf{n} ,$$

где матрица N записывается из тех же соображений дополнительных линейно независимых параметров на правом крае, а вектор **n** неизвестен.

Для правого края тоже справедлива соответствующая система уравнений:

$$\left\| \frac{V}{N} \right\| \cdot \mathbf{Y}(1) = \left| \frac{\mathbf{v}}{\mathbf{n}} \right| .$$

Запишем $\mathbf{Y}(1) = K(1 \leftarrow 0) \cdot \mathbf{Y}(0) + \mathbf{Y}^*(1 \leftarrow 0)$ и подставим в последнюю систему линейных алгебраических уравнений:

$$\left\|\frac{V}{N}\right\| \cdot [\, K(1\leftarrow 0)\cdot \mathbf{Y}(0) + \mathbf{Y^*}(1\leftarrow 0)\,] = \left|\frac{\mathbf{v}}{\mathbf{n}}\right|,$$

$$\left\|\frac{V}{N}\right\| \cdot K(1\leftarrow 0)\cdot \mathbf{Y}(0) = \left|\frac{\mathbf{v}}{\mathbf{n}}\right| - \left\|\frac{V}{N}\right\| \cdot \mathbf{Y^*}(1\leftarrow 0),$$

$$\left\|\frac{V}{N}\right\| \cdot K(1\leftarrow 0)\cdot \mathbf{Y}(0) = \left|\frac{\mathbf{v} - V\cdot \mathbf{Y}^*(1\leftarrow 0)}{\mathbf{n} - N\cdot \mathbf{Y}^*(1\leftarrow 0)}\right|,$$

$$\left\|\frac{V}{N}\right\| \cdot K(1\leftarrow 0)\cdot \mathbf{Y}(0) = \left|\frac{\mathbf{s}}{\mathbf{t}}\right| .$$

Запишем вектор $\mathbf{Y}(0)$ через обратную матрицу:

$$\mathbf{Y}(0) = \left\|\frac{U}{M}\right\|^{-1} \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right|$$

и подставим в предыдущую формулу:

$$\left\|\frac{V}{N}\right\| \cdot K(1\leftarrow 0)\cdot \left\|\frac{U}{M}\right\|^{-1} \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right| = \left|\frac{\mathbf{s}}{\mathbf{t}}\right|.$$

Таким образом, мы получили систему уравнений вида:

$$B \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right| = \left|\frac{\mathbf{s}}{\mathbf{t}}\right|,$$

где матрица B известна, векторы **u** и **s** известны, а векторы **m** и **t** неизвестны.

Разобьем матрицу B на естественные для нашего случая 4 блока и получим:

$$\left\|\begin{matrix} B11 & B12 \\ B21 & B22 \end{matrix}\right\| \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right| = \left|\frac{\mathbf{s}}{\mathbf{t}}\right|,$$

откуда можем записать, что

$$B11 \cdot \mathbf{u} + B12 \cdot \mathbf{m} = \mathbf{s},$$
$$B21 \cdot \mathbf{u} + B22 \cdot \mathbf{m} = \mathbf{t}.$$

Следовательно, искомый вектор **m** вычисляется по формуле:

$$\mathbf{m} = B12^{-1} \cdot (\mathbf{s} - B11 \cdot \mathbf{u}).$$

А искомый вектор **n** вычисляется через вектор **t**:

$$\mathbf{t} = B21 \cdot \mathbf{u} + B22 \cdot \mathbf{m},$$

$$\mathbf{n} = \mathbf{t} + N \cdot \mathbf{Y^*}(1 \leftarrow 0).$$

В случае «жестких» дифференциальных уравнений предлагается выполнять поочередное построчное ортонормирование.

Запишем приведенную выше формулу

$$\left\|\frac{V}{N}\right\| \cdot K(1 \leftarrow 0) \cdot \left\|\frac{U}{M}\right\|^{-1} \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right| = \left|\frac{\mathbf{s}}{\mathbf{t}}\right|$$

в виде:

$$\left\|\frac{V}{N}\right\| \cdot K(1 \leftarrow x2) \cdot K(x2 \leftarrow x1) \cdot K(x1 \leftarrow 0) \cdot \left\|\frac{U}{M}\right\|^{-1} \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right| = \left|\frac{\mathbf{s}}{\mathbf{t}}\right|.$$

Эту формулу можно записать в виде разделения левой части на произведение матрицы на вектор:

$$\left[\left\|\frac{V}{N}\right\| \cdot K(1 \leftarrow x2)\right] \cdot \left\{K(x2 \leftarrow x1) \cdot K(x1 \leftarrow 0) \cdot \left\|\frac{U}{M}\right\|^{-1} \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right|\right\} = \left|\frac{\mathbf{s}}{\mathbf{t}}\right|$$

[матрица] · { вектор } = **вектор**

Эту группу линейных алгебраических уравнений можно подвергнуть построчному ортонормированию, которое сделает строчки [матрицы] ортонормированными, {вектор} затронут не будет, а **вектор** получит преобразование. То есть получим:

$$\left[\left\|\frac{V}{N}\right\| \cdot K(1\leftarrow x2)\right]_{орто} \cdot \left\{K(x2\leftarrow x1)\cdot K(x1\leftarrow 0)\cdot \left\|\frac{U}{M}\right\|^{-1} \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right|\right\} = \left|\frac{\mathbf{s}}{\mathbf{?}}\right|_{орто}$$

Здесь следует сказать, что подвектор **t** подвергать преобразованию не нужно, так как невозможно, так как его первоначальное значение не известно. Но подвектор **t** нам оказывается и не нужен для решения задачи.

Далее запишем:

$$\left[\left[\left\|\frac{V}{N}\right\| \cdot K(1\leftarrow x2)\right]_{орто} \cdot K(x2\leftarrow x1)\right] \cdot \left\{K(x1\leftarrow 0)\cdot \left\|\frac{U}{M}\right\|^{-1} \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right|\right\} = \left|\frac{\mathbf{s}}{\mathbf{?}}\right|_{орто}$$

$$[\quad \text{матрица} \quad] \cdot \{\quad \text{вектор} \quad\} = \mathbf{вектор}$$

Аналогично и эту группу линейных алгебраических уравнений можно подвергнуть построчному ортонормированию, которое сделает строчки [матрицы] ортонормированными, {вектор} затронут не будет, а **вектор** получит преобразование. То есть получим:

$$\left[\left[\left\|\frac{V}{N}\right\| \cdot K(1\leftarrow x2)\right]_{орто} \cdot K(x2\leftarrow x1)\right]_{орто} \cdot \left\{K(x1\leftarrow 0)\cdot \left\|\frac{U}{M}\right\|^{-1} \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right|\right\} = \left|\frac{\mathbf{s}}{\mathbf{?}}\right|_{2\,орто}.$$

И так далее.

В результате поочередного ортонормирования получим:

$$B_{орт} \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right| = \left|\frac{\mathbf{s}}{\mathbf{t}}\right|_{орт},$$

$$\left\|\begin{matrix} B11 & B12 \\ B21 & B22 \end{matrix}\right\|_{орт} \cdot \left|\frac{\mathbf{u}}{\mathbf{m}}\right| = \left|\frac{\mathbf{s}}{\mathbf{t}}\right|_{орт}.$$

Следовательно, искомый вектор **m** вычисляется по формуле:

$$\mathbf{m} = B12_{орт}^{-1} \cdot (\mathbf{s}_{орт} - B11_{орт} \cdot \mathbf{u}).$$

7. Формула для начала счета методом прогонки С.К.Годунова.

Эта формула обсчитана на компьютерах в кандидатской диссертации.

Рассмотрим проблему метода прогонки С.К.Годунова.

Предположим, что рассматривается оболочка ракеты. Это тонкостенная труба. Тогда система линейных обыкновенных дифференциальных уравнений будет 8-го порядка, матрица A(x) коэффициентов будет иметь размерность 8x8, искомая вектор-функция **Y**(x) будет иметь размерность 8x1, а матрицы краевых условий будут прямоугольными горизонтальными размерности 4x8.

Тогда в методе прогонки С.К.Годунова для такой задачи решение ищется в следующем виде:

$$\mathbf{Y}(x) = \mathbf{Y}_1(x)\, c_1 + \mathbf{Y}_2(x)\, c_2 + \mathbf{Y}_3(x)\, c_3 + \mathbf{Y}_4(x)\, c_4 + \mathbf{Y}^*(x),$$

или можно записать в матричном виде:

$$\mathbf{Y}(x) = \mathbf{Y}_{матрица}(x) \cdot \mathbf{c} + \mathbf{Y}^*(x),$$

где векторы $\mathbf{Y}_1(x)$, $\mathbf{Y}_2(x)$, $\mathbf{Y}_3(x)$, $\mathbf{Y}_4(x)$ – это линейно независимые вектора-решения однородной системы дифференциальных уравнений, а вектор $\mathbf{Y}^*(x)$ – это вектор частного решения неоднородной системы дифференциальных уравнений.

Здесь $\mathbf{Y}_{матрица}(x)$=|| $\mathbf{Y}_1(x)$, $\mathbf{Y}_2(x)$, $\mathbf{Y}_3(x)$, $\mathbf{Y}_4(x)$ || это матрица размерности 8x4, а **c** это соответствующий вектор размерности 4x1из искомых констант c_1, c_2, c_3, c_4.

Но вообще то решение для такой краевой задачи с размерностью 8 (вне рамок метода прогонки С.К.Годунова) может состоять не из 4 линейно независимых векторов $\mathbf{Y}_i(x)$, а полностью из всех 8 линейно независимых векторов-решений однородной системы дифференциальных уравнений:

$$\mathbf{Y}(x)=\mathbf{Y}_1(x)c_1+\mathbf{Y}_2(x)c_2+\mathbf{Y}_3(x)c_3+\mathbf{Y}_4(x)c_4+$$
$$+\mathbf{Y}_5(x)c_5+\mathbf{Y}_6(x)c_6+\mathbf{Y}_7(x)c_7+\mathbf{Y}_8(x)c_8+\mathbf{Y}^*(x),$$

И как раз трудность и проблема метода прогонки С.К.Годунова и состоит в том, что решение ищется только с половиной возможных векторов и констант и проблема в том, что такое решение с половиной констант должно удовлетворять условиям на левом крае (стартовом для прогонки) при всех возможных значениях констант, чтобы потом найти эти константы из условий на правом крае.

То есть в методе прогонки С.К.Годунова есть проблема нахождения таких начальных значений $\mathbf{Y}_1(0)$, $\mathbf{Y}_2(0)$, $\mathbf{Y}_3(0)$, $\mathbf{Y}_4(0)$, $\mathbf{Y}^*(0)$ векторов $\mathbf{Y}_1(x)$, $\mathbf{Y}_2(x)$, $\mathbf{Y}_3(x)$, $\mathbf{Y}_4(x)$, $\mathbf{Y}^*(x)$, чтобы можно было начать прогонку с левого края x=0, то есть чтобы удовлетворялись условия $U\cdot\mathbf{Y}(0) = \mathbf{u}$ на левом крае при любых значениях констант c_1, c_2, c_3, c_4.

Обычно эта трудность «преодолевается» тем, что дифференциальные уравнения записываются не через функционалы, а через физические параметры и рассматриваются самые простейшие условия на простейшие физические параметры, чтобы начальные значения $\mathbf{Y}_1(0)$, $\mathbf{Y}_2(0)$, $\mathbf{Y}_3(0)$, $\mathbf{Y}_4(0)$, $\mathbf{Y}^*(0)$ можно было угадать. То есть задачи со сложными краевыми условиями так решать нельзя: например, задачи с упругими условиями на краях.

Ниже предлагается формула для начала вычислений методом прогонки С.К.Годунова.

Выполним построчное ортонормирование матричного уравнения краевых условий на левом крае:

$$U\cdot\mathbf{Y}(0) = \mathbf{u},$$

где матрица U прямоугольная и горизонтальная размерности 4x8.

В результате получим эквивалентное уравнение краевых условий на левом крае, но уже с прямоугольной горизонтальной матрицей $U_{орто}$ размерности 4x8, у которой будут 4 ортонормированные строки:

$$U_{орто}\cdot\mathbf{Y}(0) = \mathbf{u}_{орто},$$

где в результате ортонормирования вектор **u** преобразован в вектор $\mathbf{u}_{\text{орто}}$.

Как выполнять построчное ортонормирование систем линейных алгебраических уравнений можно посмотреть в [Березин, Жидков].

Дополним прямоугольную горизонтальную матрицу $U_{\text{орто}}$ до квадратной невырожденной матрицы W:

$$W = \left\| \frac{U_{\text{орто}}}{M} \right\|,$$

где матрица M размерности 4x8 должна достраивать матрицу $U_{\text{орто}}$ до невырожденной квадратной матрицы W размерности 8x8.

В качестве строк матрицы M можно взять те краевые условия, то есть выражения тех физических параметров, которые не входят в параметры левого края или линейно независимы с ними. Это вполне возможно, так как у краевых задач столько независимых физических параметров какова размерность задачи, то есть в данном случае их 8 штук и если 4 заданы на левом крае, то ещё 4 можно взять с правого края.

Завершим ортонормирование построенной матрицы W, то есть выполним построчное ортонормирование и получим матрицу $W_{\text{орто}}$ размерности 8x8 с ортонормированными строками:

$$W_{\text{орто}} = \left\| \frac{U_{\text{орто}}}{M_{\text{орто}}} \right\|.$$

Можем записать, что

$$\mathbf{Y}_{\text{матрица}}(0) = (M_{\text{орто}})\text{транспонированная} = M^{T}_{\text{орто}}.$$

Тогда, подставив в формулу метода прогонки С.К.Годунова, получим:

$$\mathbf{Y}(0) = \mathbf{Y}_{\text{матрица}}(0) \cdot \mathbf{c} + \mathbf{Y}^{*}(0)$$

или

$$\mathbf{Y}(0) = M^{T}_{\text{орто}} \cdot \mathbf{c} + \mathbf{Y}^{*}(0).$$

Подставим эту последнюю формулу в краевые условия $U_{\text{орто}} \cdot \mathbf{Y}(0) = \mathbf{u}_{\text{орто}}$ и получим:

$$U_{\text{орто}} \cdot [\, M^{T}_{\text{орто}} \cdot \mathbf{c} + \mathbf{Y}^{*}(0)\,] = \mathbf{u}_{\text{орто}}.$$

Отсюда получаем, что на левом крае константы **c** уже не на что не влияют, так как $U_{\text{орто}} \cdot M^{T}_{\text{орто}} = 0$ и остается только найти $\mathbf{Y}^{*}(0)$ из выражения:

$$U_{\text{орто}} \cdot \mathbf{Y}^{*}(0) = \mathbf{u}_{\text{орто}}.$$

Но матрица $U_{\text{орто}}$ имеет размерность 4x8 и её надо дополнить до квадратной невырожденной, чтобы найти вектор $\mathbf{Y}^{*}(0)$ из решения соответствующей системы линейных алгебраических уравнений:

$$\left\| \frac{U_{\text{орто}}}{M_{\text{орто}}} \right\| \cdot \mathbf{Y}^{*}(0) = \left| \frac{\mathbf{u}_{\text{орто}}}{\mathbf{0}} \right|,$$

где **0** – любой вектор, в том числе вектор из нулей.

Отсюда получаем при помощи обратной матрицы:

$$\mathbf{Y}^{*}(0) = \left\| \frac{U_{\text{орто}}}{M_{\text{орто}}} \right\|^{-1} \cdot \left| \frac{\mathbf{u}_{\text{орто}}}{\mathbf{0}} \right|,$$

Тогда итоговая формула для начала вычислений методом прогонки С.К.Годунова имеет вид:

$$\mathbf{Y}(0) = M^{T}_{\text{орто}} \cdot \mathbf{c} + \left\| \frac{U_{\text{орто}}}{M_{\text{орто}}} \right\|^{-1} \cdot \left| \frac{\mathbf{u}_{\text{орто}}}{\mathbf{0}} \right|.$$

8. Второй алгоритм для начала счета методом прогонки С.К.Годунова.

Этот алгоритм обсчитан на компьютерах в кандидатской диссертации.

Этот алгоритм требует дополнения матрицы краевых условий U до квадратной невырожденной:

$$\left\| \frac{U}{M} \right\|$$

Начальные значения $\mathbf{Y}_1(0)$, $\mathbf{Y}_2(0)$, $\mathbf{Y}_3(0)$, $\mathbf{Y}_4(0)$, $\mathbf{Y}^*(0)$ находятся из решения следующих систем линейных алгебраических уравнений:

$$\left\| \frac{U}{M} \right\| \cdot \mathbf{Y}^*(0) = \left| \frac{\mathbf{u}}{\mathbf{0}} \right|,$$

$$\left\| \frac{U}{M} \right\| \cdot \mathbf{Y}_i(0) = \left| \frac{\mathbf{0}}{\mathbf{i}} \right|, \text{ где } \mathbf{i} = \left| \begin{matrix} 1 \\ 0 \\ 0 \\ 0 \end{matrix} \right|, \left| \begin{matrix} 0 \\ 1 \\ 0 \\ 0 \end{matrix} \right|, \left| \begin{matrix} 0 \\ 0 \\ 1 \\ 0 \end{matrix} \right|, \left| \begin{matrix} 0 \\ 0 \\ 0 \\ 1 \end{matrix} \right|,$$

где **0** – вектор из нулей размерности 4x1.

9. Замена метода численного интегрирования Рунге-Кутта в методе прогонки С.К.Годунова.

Эта замена формул Рунге-Кутта на формулу теории матриц обсчитана на компьютерах в кандидатской диссертации.

В методе С.К.Годунова как показано выше решение ищется в виде:

$$\mathbf{Y}(x) = \mathbf{Y}_{\text{матрица}}(x) \cdot \mathbf{c} + \mathbf{Y}^*(x).$$

На каждом конкретном участке метода прогонки С.К.Годунова между точками ортогонализации можно вместо метода Рунге-Кутта пользоваться теорией матриц и выполнять расчет через матрицу Коши:

$$\mathbf{Y}_{\text{матрица}}(x_j) = K(x_j - x_i) \cdot \mathbf{Y}_{\text{матрица}}(x_i).$$

Так выполнять вычисления быстрее, особенно для дифференциальных уравнений с постоянными коэффициентами.

И аналогично через теорию матриц можно вычислять и вектор **Y***(x) частного решения неоднородной системы дифференциальных уравнений. Или для этого вектора отдельно можно использовать метод Рунге-Кутта, то есть можно комбинировать теорию матриц и метод Рунге-Кутта.

10. Метод половины констант.

Этот метод пока не обсчитан на компьютерах.

Выше было показано, что решение системы линейных обыкновенных дифференциальных уравнений можно искать в виде только с половиной возможных векторов и констант. Была приведена формула для начала вычислений:

$$\mathbf{Y}(0) = M_{орто}^{T} \cdot \mathbf{c} + \left\| \frac{U_{орто}}{M_{орто}} \right\|^{-1} \cdot \left| \frac{\mathbf{u}_{орто}}{\mathbf{0}} \right| .$$

Из теории матриц известно, что если матрица ортонормирована, то её обратная матрица есть её транспонированная матрица. Тогда последняя формула приобретает вид:

$$\mathbf{Y}(0) = M_{орто}^{T} \cdot \mathbf{c} + U_{орто}^{T} \cdot \mathbf{u}_{орто}$$

или

$$\mathbf{Y}(0) = U_{орто}^{T} \cdot \mathbf{u}_{орто} + M_{орто}^{T} \cdot \mathbf{c}$$

или

$$\mathbf{Y}(0) = \left\| U_{орто}^{T} \quad M_{орто}^{T} \right\| \cdot \left| \frac{\mathbf{u}_{орто}}{\mathbf{c}} \right| ,$$

Таким образом записана в матричном виде формула для начала счета с левого края, когда на левом крае удовлетворены краевые условия.

Далее запишем $V \cdot \mathbf{Y}(1) = \mathbf{v}$ и $\mathbf{Y}(1) = K(1 \leftarrow 0) \cdot \mathbf{Y}(0) + \mathbf{Y}^*(1 \leftarrow 0)$ совместно:

$$V\cdot [\ K(1\leftarrow 0)\cdot \mathbf{Y}(0) + \mathbf{Y}^*(1\leftarrow 0)\ ] = \mathbf{v}$$

$$V\cdot K(1\leftarrow 0)\cdot \mathbf{Y}(0) = \mathbf{v} - V\cdot \mathbf{Y}^*(1\leftarrow 0)$$

и подставим в эту формулу выражение для $\mathbf{Y}(0)$:

$$V\cdot K(1\leftarrow 0)\cdot \left\| U^T_{\text{орто}} \quad M^T_{\text{орто}} \right\| \cdot \left| \frac{\mathbf{u}_{\text{орто}}}{\mathbf{c}} \right| = \mathbf{v} - V\cdot \mathbf{Y}^*(1\leftarrow 0).$$

$$V\cdot K(1\leftarrow 0)\cdot \left\| U^T_{\text{орто}} \quad M^T_{\text{орто}} \right\| \cdot \left| \frac{\mathbf{u}_{\text{орто}}}{\mathbf{c}} \right| = \mathbf{p}.$$

Таким образом мы получили выражение вида:

$$D\cdot \left| \frac{\mathbf{u}_{\text{орто}}}{\mathbf{c}} \right| = \mathbf{p},$$

где матрица D имеет размерность 4x8 и может быть естественно представлена в виде двух квадратных блоков размерности 4x4:

$$\left\| D1 \quad D2 \right\| \cdot \left| \frac{\mathbf{u}_{\text{орто}}}{\mathbf{c}} \right| = \mathbf{p}.$$

Тогда можем записать:

$$D1\cdot \mathbf{u}_{\text{орто}} + D2\cdot \mathbf{c} = \mathbf{p}.$$

Отсюда получаем, что:

$$\mathbf{c} = D2^{-1}\cdot (\ \mathbf{p} - D1\cdot \mathbf{u}_{\text{орто}}\ )$$

Таким образом, искомые константы найдены.

Далее показано как применять этот метод для решения «жестких» краевых задач.

Запишем

$$V\cdot K(1\leftarrow 0)\cdot \left\| U^{T}_{орто} \quad M^{T}_{орто} \right\| \cdot \left| \frac{\mathbf{u}_{орто}}{\mathbf{c}} \right| = \mathbf{p}.$$

совместно с $K(1\leftarrow 0) = K(1\leftarrow x2)\cdot K(x2\leftarrow x1)\cdot K(x1\leftarrow 0)$ и получим:

$$V\cdot K(1\leftarrow x2)\cdot K(x2\leftarrow x1)\cdot K(x1\leftarrow 0)\cdot \left\| U^{T}_{орто} \quad M^{T}_{орто} \right\| \cdot \left| \frac{\mathbf{u}_{орто}}{\mathbf{c}} \right| = \mathbf{p}.$$

Эту систему линейных алгебраических уравнений можно представить в виде:

$$[\ V\cdot K(1\leftarrow x2)\ ]\cdot \{\ K(x2\leftarrow x1)\cdot K(x1\leftarrow 0)\cdot \left\| U^{T}_{орто} \quad M^{T}_{орто} \right\| \cdot \left| \frac{\mathbf{u}_{орто}}{\mathbf{c}} \right| \} = \mathbf{p}.$$

[матрица] · { вектор } = **вектор**

Эту группу линейных алгебраических уравнений можно подвергнуть построчному ортонормированию, которое сделает строчки [матрицы] ортонормированными, {вектор} затронут не будет, а **вектор** получит преобразование. То есть получим:

$$[\ V\cdot K(1\leftarrow x2)\ ]_{орто}\cdot \{\ K(x2\leftarrow x1)\cdot K(x1\leftarrow 0)\cdot \left\| U^{T}_{орто} \quad M^{T}_{орто} \right\| \cdot \left| \frac{\mathbf{u}_{орто}}{\mathbf{c}} \right| \} = \mathbf{p}_{орто}.$$

И так далее.

В итоге поочередного вычленений матриц слева из вектора и ортонормирования получим систему:

$$D_{ортонорм}\cdot \left| \frac{\mathbf{u}_{орто}}{\mathbf{c}} \right| = \mathbf{p}_{ортонорм},$$

Отсюда получаем, что:

$$\mathbf{c} = D2^{-1}_{ортонорм}\cdot (\mathbf{p}_{ортонорм} - D1_{ортонорм}\cdot \mathbf{u}_{орто})$$

Таким образом, искомые константы найдены.

11. Применяемые формулы ортонормирования.

Эти формулы обсчитаны в кандидатской диссертации.

Взято из: Березин И.С., Жидков Н.П. Методы вычислений, том II, Государственное издательство физико-математической литературы, Москва, 1962 г. 635 стр.

Пусть дана система линейных алгебраических уравнений порядка n:

$$A\bar{x}=\bar{b}.$$

Здесь над векторами поставим черточки вместо их обозначения жирным шрифтом.

Будем рассматривать строки матрицы A системы как векторы:

$$\overline{a_i}=(a_{i1},a_{i2},\dots,a_{in}).$$

Ортонормируем эту систему векторов.

Первое уравнение системы $A\bar{x}=\bar{b}$ делим на $\sqrt{\sum_{k=1}^{n}a_{1k}^2}$.

При этом получим:

$$c_{11}x_1+c_{12}x_2+\dots+c_{1n}x_n=d_1,\quad \overline{c_1}=(c_{11},c_{12},\dots,c_{1n}),$$

$$\text{где } c_{1k}=\frac{a_{1k}}{\sqrt{\sum_{k=1}^{n}a_{1k}^2}},\ d_1=\frac{b_1}{\sqrt{\sum_{k=1}^{n}a_{1k}^2}},\ \sum_{k=1}^{n}c_{1k}^2=1.$$

Второе уравнение системы заменяется на:

$$c_{21}x_1+c_{22}x_2+\dots+c_{2n}x_n=d_2,\quad \overline{c_2}=(c_{21},c_{22},\dots,c_{2n}),$$

$$\text{где } c_{2k}=\frac{c'_{2k}}{\sqrt{\sum_{k=1}^{n}c'^{2}_{2k}}},\ d_2=\frac{d'_2}{\sqrt{\sum_{k=1}^{n}c'^{2}_{2k}}},$$

$$c'_{2k}=a_{2k}-(\overline{a_2},\overline{c_1})c_{1k},\quad d'_2=b_2-(\overline{a_2},\overline{c_1})d_1.$$

Аналогично поступаем дальше. Уравнение с номером i примет вид:

$$c_{i1}x_1+c_{i2}x_2+\dots+c_{in}x_n=d_i,\quad \overline{c_i}=(c_{i1},c_{i2},\dots,c_{in}),$$

где $c_{ik} = \frac{c'_{ik}}{\sqrt{\sum_{k=1}^{n} c'^{2}_{ik}}}$, $d_i = \frac{d'_i}{\sqrt{\sum_{k=1}^{n} c'^{2}_{ik}}}$,

$$c'_{ik} = a_{ik} - (\overline{a_i}, \overline{c_1}) c_{1k} - (\overline{a_i}, \overline{c_2}) c_{2k} - \ldots - (\overline{a_i}, \overline{c_{i-1}}) c_{i-1,k},$$

$$d'_i = b_i - (\overline{a_i}, \overline{c_1}) d_1 - (\overline{a_i}, \overline{c_2}) d_2 - \ldots - (\overline{a_i}, \overline{c_{i-1}}) d_{i-1}.$$

Процесс будет осуществим, если система линейных алгебраических уравнений линейно независима.

В результате мы придем к новой системе $C\bar{x} = \bar{d}$, где матрица C будет с ортонормированными строками, то есть обладает свойством $C*C^T = E$, где E – это единичная матрица.

(Таким образом, решение системы можно записать в виде $\bar{x} = C^T \bar{d}$.)

12. Вывод формул, позаимствованный из «Теории матриц» Гантмахера.

Система линейных обыкновенных дифференциальных уравнений с постоянными коэффициентами имеет вид:

$$\mathbf{Y}'(x) = A\,\mathbf{Y}(x) + \mathbf{F}(x). \qquad (1)$$

Разложим **Y**(x) в ряд Маклорена по степеням x:

$$\mathbf{Y}(x) = \mathbf{Y}_0 + \mathbf{Y}'_0 x + \mathbf{Y}''_0 x^2/2! + \ldots, \text{ где } \mathbf{Y}_0 = \mathbf{Y}(0), \mathbf{Y}'_0 = \mathbf{Y}'(0), \ldots \qquad (2).$$

Из (1) почленным дифференцированием при A=const и **F**(x)=0 получим:

$$\mathbf{Y}'' = A\mathbf{Y}' = A^2\mathbf{Y}, \quad \mathbf{Y}''' = A\,\mathbf{Y}'' = A^3\mathbf{Y}, \qquad (3)$$

Положив в (3) x=0 и подставив в (2) получим:

$$\mathbf{Y}(x) = \mathbf{Y}_0 + Ax\,\mathbf{Y}_0 + A^2 x^2/2!\ \mathbf{Y}_0 + \ldots = e^{Ax}\ \mathbf{Y}_0, \quad (4)$$

где $e^{Ax} = E + Ax + A^2 x^2/2! + \ldots$, где E – единичная матрица. (5)

Если принять $x = x_0$, то (4) заменится на

$$\mathbf{Y}(x) = e^{A(x-x_0)}\ \mathbf{Y}(x_0), \qquad (6)$$

Рассмотрим случай A=const и **F**≠0.

Введем в рассмотрение вектор-функцию **Ya**(x) в виде: $\mathbf{Y}(x) = e^{Ax}\,\mathbf{Ya}(x)$. (7)

Продиффренцируем (7) и подставим в (1). Получим:

$$e^{Ax}\,\mathbf{Ya}'(x) = \mathbf{F}(x). \quad (8)$$

При получении (8) учитывалось, что:

$$\frac{d(e^{Ax})}{dx} = \frac{d(E + Ax + A^2x^2/2!+...)}{dx} = A + A^2\,x + A^3\,x^2/2! + \ldots = A\,e^{Ax}.$$

Из (8) следует, что:

$$\mathbf{Ya}(x) = c + \int_{x_0}^{x} e^{(-At)}F(t)dt. \quad (9)$$

Подставим в (7) и получаем:

$$\mathbf{Y}(x) = e^{Ax}\,c + e^{Ax}\int_{x_0}^{x} e^{(-At)}F(t)dt. \quad (10)$$

Положив $x=x_0$ в (10) получим:

$$c = e^{-Ax_0}\,\mathbf{Y}(x_0). \quad (11)$$

Окончательно получаем:

$$\mathbf{Y}(x) = e^{A(x-x_0)}\,\mathbf{Y}(x_0) + e^{Ax}\int_{x_0}^{x} e^{(-At)}F(t)dt. \quad (12)$$

ЛИТЕРАТУРА

1. Гантмахер Ф.Р. Теория матриц. – М.: Наука, 1988. – 548 с.
2. Березин И.С., Жидков Н.П. Методы вычислений, том II, Государственное издательство физико-математической литературы, Москва, 1962 г., 635 с.

Смотрите мои сайты:

www.vinogradov-design.narod.ru/math.html

www.vinogradov-best.narod.ru

www.AlexeiVinogradov.narod.ru

www.VinogradovAlexei.narod.ru

www.Vinogradov-Alexei.narod.ru

www.Vinogradov-math.narod.ru

Пишите мне:

AlexeiVinogradov@yandex.ru

13. Метод Вольтерра. (P.S. 28 февраля 2010):

Мой отец (доктор физико-математических наук профессор МГТУ им. Баумана Юрий Иванович Виноградов) предложил использовать и другую (гораздо более эффективную по времени счета) матричную формулу вместо матричной экспоненты – что-то на основе Вольтерра. Это есть в статье в журнале «Математическое моделирование»:

Численный метод переноса краевых условий для жестких дифференциальных уравнений строительной механики

Журнал "ММ", Том: 14 (2002), Номер: 9, 3 стр. 1409-003r.pdf

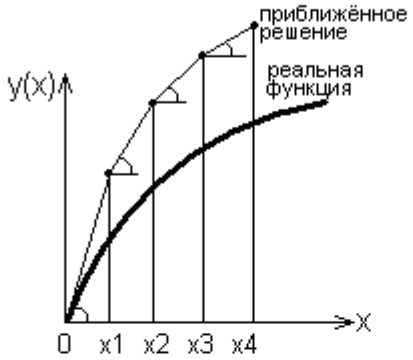
14. P.P.S. Метод для численного интегрирования дифференциальных уравнений.

Читали нам как-то в бауманке численные методы решения дифференциальных уравнений. И, кажется, приводили аналитический вывод формул одного из авторов. Или это просто мелькнуло в учебнике (я имею в виду вывод формул). Уже не очень помню. Запомнилась только собственная мысль, что людям вообще-то проще всего даются геометрические аналогии и выводы, сделанные на основе понятных геометрических картинок. Ну, вот тогда я и нарисовал один из вариантов численного решения дифференциальных уравнений и помню даже перевёл геометрические картинки в буквенные формулы приближённых вычислений. Сейчас повторно выводить буквенные формулы для численного интегрирования дифференциальных уравнений мне не кажется интересным. А вот привести картинки тех студенческих мыслей вполне можно для обсуждения.

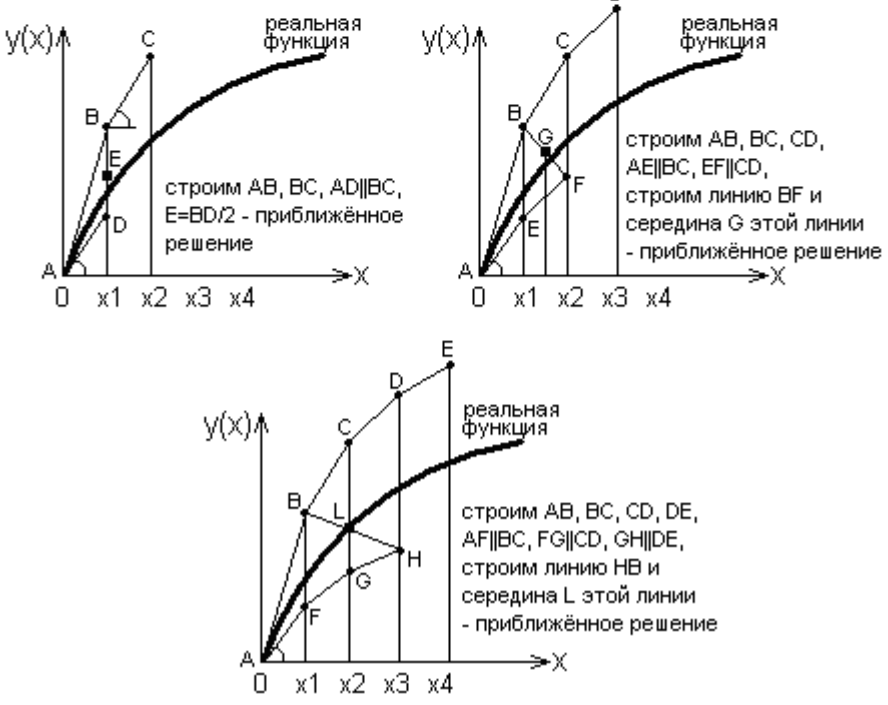
Далее идёт картинка с текстом и с рисунками численного интегрирования:

Предположим, что дано дифференциальное уравнение
y'(x)=a(x)y(x)+f(x) с начальными условиями y(0)=y0
или дана система диф.ур. Y'(x)=A(x)Y(x)+F(x) с н.у. Y(0)=Y0.
Допустим, что надо найти решение не аналитически, а численно.

Есть множество методов разных авторов как выполнять приближённое численное решение (интегрирование) дифференциального уравнения (или системы диф.ур.). Простейший и самый ошибочный метод таков: по начальным условиям из диф.ур. высчитывается y'(0) и делается некий шаг на x1 в предположении, что функция на участке от 0 до x1 моделируется прямой линией: y(x1)=y'(0)x1 . И так далее шаг за шагом. Геометрически это выглядит вот так:



Это был приведён самый неэффективный из известных методов. Разные авторы численных методов использовали разные приёмы получения формул численного интегрирования с идеей уменьшения ошибочности вычислений. Можно предложить ещё один вариант - геометрический метод, который, конечно может быть выражен и самыми обычными формулами для приближённых вычислений.



15. 17 сентября 2010: Забыл ранее кое-что добавить – добавляю сегодня насчет обратной матрицы.

Для однородной системы дифференциальных уравнений имеем:

$$\mathbf{Y}(x) = K(x \leftarrow x_0) \cdot \mathbf{Y}(x_0).$$

Можем записать:

$$\mathbf{Y}(x_j) = K(x_j \leftarrow x_i) \cdot \mathbf{Y}(x_i) \text{ и}$$

$$\mathbf{Y}(x_i) = K(x_i \leftarrow x_j) \cdot \mathbf{Y}(x_j).$$

Подставляем одну формулу в другую и получаем:

$$\mathbf{Y}(x_j) = K(x_j \leftarrow x_i) \cdot \mathbf{Y}(x_i) = K(x_j \leftarrow x_i) \cdot K(x_i \leftarrow x_j) \cdot \mathbf{Y}(x_j),$$

то есть получаем:

$$\mathbf{Y}(x_j) = K(x_j \leftarrow x_i) \cdot K(x_i \leftarrow x_j) \cdot \mathbf{Y}(x_j),$$

но последнее возможно только когда

$$K(x_j \leftarrow x_i) \cdot K(x_i \leftarrow x_j) = E \text{ – единичная матрица,}$$

то есть матрицы $K(x_j \leftarrow x_i)$ и $K(x_i \leftarrow x_j)$ взаимообратны.

То есть доказано, что

$$K^{-1}(x_j \leftarrow x_i) = K(x_i \leftarrow x_j),$$

то есть

$$K^{-1}(x_j - x_i) = K(x_i - x_j).$$

16. 18 сентября 2010: Еще кой о чем вспомнил – вычисление матрицы Коши методами типа Рунге-Кутта.

Я сейчас ниже выскажу мысли, которые мне кажутся очевидными, но они приводятся без доказательства и они не проверялись вычислительными экспериментами – так что все в Ваших руках (можно проверить численно на компьютере и можно написать и опубликовать соответствующие статьи).

Итак. Матрица Коши $K(x_j - x_i)$ вычисляется как матричная экспонента или по формуле Вольтерра (смотри выше). Я же считаю очевидным, что матрицу Коши можно вычислять и методами типа Рунге-Кутта и другими аналогичными численными методами, включая

мой «геометрический» (неопробованный) численный метод, который приводится на картинке выше.

Матрица Коши $K(x_j-x_i)$ обладает тем свойством, что ее значение при нулевом аргументе $(x_j-x_i)=\Delta x=0$ это есть единичная матрица $K(0)=E$.

Тогда мне кажется очевидным, что вектора, составляющие матрицу Коши $K(x_j-x_i)$ при НЕнулевом аргументе $\Delta x=(x_j-x_i)\neq 0$ можно получить следующим путем: берем из единичной матрицы E вертикальный вектор (столбец) с номером i и методом Рунге-Кутта от этого взятого начального значения (i-го столбца из единичной матрицы) вычисляем некий вектор-столбец на выбранном шаге интегрирования $\Delta x=(x_j-x_i)\neq 0$ и записываем этот вектор-столбец в результирующую матрицу Коши на свое же i-ое место, то есть полученный вектор-столбец записываем в качестве i-го столбца результирующей матрицы Коши. И так формируем все столбцы вычисляемой (результирующей) матрицы Коши.

Например, для дифференциальных уравнений цилиндрической оболочки ракеты матрица Коши имеет размерность 8x8, то есть состоит из 8 нужных нам столбцов размерности 8x1. И методами типа Рунге-Кутта мы вычисляем соответственно 8 столбцов матрицы Коши $K(x_j-x_i)$ на выбранном шаге интегрирования $\Delta x=(x_j-x_i)\neq 0$, беря в качестве начальных значений векторов-столбцов для метода Рунге-Кутта столбцы размерности 8x1 из единичной матрицы размерности 8x8.

Кстати, для этого случая круговой цилиндрической оболочки мы имеем систему дифференциальных уравнений с постоянными коэффициентами и поэтому можем вычислить матрицу Коши однажды на одном маленьком интервале $\Delta x=(x_j-x_i)\neq 0$, а на всем интервале задачи (1-0) получаем полную матрицу Коши перемножением самой на себя единожды вычисленной матрицы Коши малого участка: $K(1\text{-}0)= K(\Delta x)\cdot K(\Delta x)\cdot\ldots\cdot K(\Delta x)$, что очень сильно ускоряет вычисления по сравнению со случаями переменных коэффициентов (конус, сфера), так как при случае переменных коэффициентов приходится вычислять матрицы Коши $K(x_j-x_i)$ независимо для каждого маленького отдельного шага (x_j-x_i) полного интервала интегрирования всей задачи с их аналогичным последующим перемножением.

То есть для случая постоянных коэффициентов системы дифференциальных уравнений не особенно важна скорость вычисления матрицы Коши K(Δx) одного отдельного малого участка Δx, так как на малом участке матрица Коши K(Δx) вычисляется единожды, так как потом полученная матрица Коши малого участка перемножается сама на себя.

Тогда получается, что для вычисления матрицы Коши можно применять все известные численные методы, начиная с метода Рунге-Кутта и заканчивая моим предложенным выше и пока неопробованным «геометрическим» численным методом.

Скажу дополнительно, что касательно вектора частного решения неоднородной системы дифференциальных уравнений у меня нет таких очевидных догадок, какие высказаны выше касательно матрицы Коши.

То есть я не догадываюсь как вычислять методами типа Рунге-Кутта вектор $\mathbf{Y^*}(x \leftarrow x_0)$:

$$\mathbf{Y^*}(x \leftarrow x_0) = e^{Ax} \cdot \int_{x_0}^{x} e^{-At} \cdot \mathbf{F}(t)\, dt \quad -$$

вектор частного решения неоднородной системы дифференциальных уравнений.

Хотя вполне может оказаться, что все на самом деле просто и надо применять метод Рунге-Кутта к начальному полностью нулевому вектору, то есть вектору, состоящему полностью из нулей. Но это не факт. То есть мы так очевидно получим вариант частного решения неоднородной системы дифференциальных уравнений, но не факт, что этот частный вектор будет удовлетворять формуле:

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

То есть, интегрируя Рунге-Куттом от полностью нулевого вектора неоднородную систему дифференциальных уравнений, мы получим таки некий вектор частного решения, но не факт, что этот вектор уложиться без поправок в формулу:

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

То есть касательно частного вектора $\mathbf{Y^*}(x \leftarrow x_0)$ тут тоже все в Ваших руках – можете проверить численно разные численные методы типа Рунге-Кутта и может быть можете придумать коррекцию (если она потребуется) и можно публиковать статьи.

17. 19 сентября 2010: еще об ускорении вычислений – применение «параллельных» вычислений.

В современной математике для современных компьютеров разрабатываются и применяются так называемые методы «параллельных вычислений». Особенно для промышленных суперкомпьютеров на основе многопроцессорной технологии. Эти методы сильно ускоряют вычисления для практических задач в НИИ и КБ.

Приведенные здесь методы как будто специально предназначены для «параллельных вычислений» даже без дополнительной модификации.

В приведенных методах используется матрица Коши и ее свойство перемножаемости:

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

Так вот, очевидно, что множители $K(x_i \leftarrow x_{i-1})$, $K(x_{i-1} \leftarrow x_{i-2})$, … , $K(x_2 \leftarrow x_1)$, $K(x_1 \leftarrow x_0)$, входящие в формулу для вычисления матрицы Коши $K(x_i \leftarrow x_0)$, вычисляются полностью независимо друг от друга и поэтому могут вычисляться «параллельно» без привлечения дополнительных математических приемов.

То есть отдельные матрицы Коши на отдельных участках (для случая системы дифференциальных уравнений с переменными коэффициентами) могут вычисляться «параллельно» и например, просто на разных процессорах многопроцессорного компьютера какого-нибудь КБ (конструкторского бюро).

Аналогично «параллельно» может вычисляться и вектор частного решения неоднородной системы дифференциальных уравнений, так как подвектора $\mathbf{Y^*}(x_j \leftarrow x_i)$, из которых складывается частный вектор, вычисляются полностью независимо, то есть «параллельно»:

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

$$= K(x_j - x_i) \cdot \int_{x_i}^{x_j} (E + A_i\,(x_i - t) + A_i{}^2\,(x_i - t)^2/2! + \dots) \cdot \mathbf{F}(t)\,dt =$$

$$= K(x_j - x_i) \cdot (E \int_{x_i}^{x_j} \mathbf{F}(t)\,dt + A_i \cdot \int_{x_i}^{x_j} (x_i - t) \cdot \mathbf{F}(t)\,dt + A_i{}^2/2! \cdot \int_{x_i}^{x_j} (x_i - t)^2 \cdot \mathbf{F}(t)\,dt + \dots),$$

где $A_i = A(x_i)$ – матрица коэффициентов системы дифференциальных уравнений, вычисленная в приближении ее постоянства на малом подучастке $(x_j - x_i)$ и вычисленная на малом подучастке $(x_j - x_i)$, например, в начальной точке x_i рассматриваемого подучастка $(x_j - x_i)$. Хотя матрица A подучастка может быть вычислена и, например, в точке x_j подучастка $(x_j - x_i)$: $A_i = A(x_j)$.

Кстати, в приведенной выше формуле осреднению может подвергаться не только матрица A: $A_i = A(x_i)$ коэффициентов системы дифференциальных уравнений, но и вектор **F**(t) может рассматриваться на подучастке $(x_j - x_i)$ приближенно в виде постоянной величины $\mathbf{F}(x_i)$=constant, что позволят вынести его из под знака интеграла, что приводит к совсем простому ряду для вычислений на рассматриваемом подучастке.

То есть все приведенные выше методы Алексея Юрьевича Виноградова очень удачно встраиваются в современную тенденцию высокоскоростных «параллельных вычислений».

☺

18. 23 августа 2011: Вычисление вектора частного решения неоднородной системы дифференциальных уравнений.

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.

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 $\int \mathrm{T}^{\mathrm{n}} dt = \frac{1}{\mathrm{a}(\mathrm{n}+1)}\mathrm{T}^{\mathrm{n}+1} + \text{const}$ (при $\mathrm{n} \neq -1$).

В нашем случае имеем $\int (\mathrm{b} - \mathrm{t})^{\mathrm{n}} dt = \frac{1}{(-1)(\mathrm{n}+1)}(\mathrm{b} - \mathrm{t})^{\mathrm{n}+1} + \text{const}$ (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 $(x_j - x_i)$:

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

19. 02 октября 2011: Авторство.

Мой метод - метод Алексея Юрьевича Виноградова «переноса краевых условий» первоначально был опубликован в Межвузовском сборнике МИРЭА (кажется в 1995 году). МИРЭА это Московский институт радиотехники, электроники и автоматики. Точное название и год выхода статьи можно посмотреть в Ленинской библиотеке в списке литературы моей диссертации. Там у меня только одна статья в МИРЭА. К сожалению, на руках у меня нет экземпляра моей кандидатской диссертации, поэтому не могу привести точное название статьи, но называется она, кажется, что-то вроде «Метод приведения краевых задач к задаче Коши».

13 мая 2011 нашел я в интернете случайно свою старую статью по начальным векторам, в которой я первоначально предложил ортонормировать краевые условия: Вычисление начальных векторов для численного решения краевых задач, А. Ю. Виноградов «Ж. вычисл. матем. и матем. физ.», 1995, 35:1, 156–159. Теперь я эту статью положил на свой сайт www.VinogradovAlexei.narod.ru.

После защиты своей кандидатской диссертации в 1996 году я совсем бросил заниматься наукой и с 1996 года по 2005 год совсем не занимался математикой. И после 1996 года мой отец (доктор физико-математических наук профессор МГТУ имени Баумана Виноградов Юрий Иванович) уже без моего ведома публиковал эти материалы теперь уже как наш совместный с ним метод. Включая, например, публикацию в Докладах Академии наук: А.Ю.Виноградов, Ю.И.Виноградов, Метод переноса краевых условий функциями Коши-Крылова для жестких линейных обыкновенных дифференциальных уравнений. // ДАН. – М.: 2000, т. 373, №4, с. 474-476.

Алексей Юрьевич Виноградов

Кандидат физико-математических наук (1996 года защиты)

Дата рождения: 12 апреля 1970 (а то в интернете много моих полных тезок)

Мои сайты по методам решения краевых задач в интернете:

www.vinogradov-design.narod.ru/math.html

www.vinogradov-best.narod.ru

www.AlexeiVinogradov.narod.ru www.VinogradovAlexei.narod.ru

www.Vinogradov-Alexei.narod.ru www.Vinogradov-Math.narod.ru

20.1. 27 ноября 2011: Метод решения жестких краевых задач без ортонормирования – метод сопряжения участков, выраженных матричными экспонентами – метод д.ф.-м.н. Юрия Ивановича Виноградова и к.ф.-м.н. Алексея Юрьевича Виноградова.

Этот метод проверен компьютерными расчетами.

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:

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

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 оказывается не надо.

20.2. Программа на C++ расчета цилиндра.

Вычислительные эксперименты проводились в сравнении с методом переноса краевых условий Алексея Юрьевича Виноградова. В этом методе используется построчное ортонормирование.

Без ортонормирования в методе переноса краевых условий А.Ю.Виноградова 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.

ПРОГРАММА НА С++ (ЦИЛИНДР):

```
// 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;
}
```

21. 19 декабря 2012: Случай переменных коэффициентов (ошибка).

Где-то 2 года назад мой отец доктор физико-математических наук профессор МГТУ им. Баумана показал мне его статью в «Докладах Российской Академии наук» о том, как вычислять матрицу Коши в случае, когда система дифференциальных уравнений имеет матрицу с переменными коэффициентами $A = A(x)$.

До статьи моего отца это можно было делать так:

Из теории матриц [Гантмахер] 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 предлагается 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:

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

Новый материал (ошибка):

Когда я читал статью своего отца, то я не понял сложный вывод формулы. А сегодня утром мне вдруг (непонятно откуда) отчётливо пришла в голову мысль о том, как можно вывести формулу моего отца почти в одно действие.

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

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

$$= \exp(A(x_i) \cdot \Delta x_i) \cdot \exp(A(x_{i-1}) \cdot \Delta x_{i-1}) \cdot \ldots \cdot \exp(A(x_2) \cdot \Delta x_2) \cdot \exp(A(x_1) \cdot \Delta x_1) =$$

$$= \exp(A(x_i) \cdot \Delta x_i + A(x_{i-1}) \cdot \Delta x_{i-1} + \ldots + A(x_2) \cdot \Delta x_2 + A(x_1) \cdot \Delta x_1) =$$

$$\text{пусть } \Delta x_i = x_i - x_{i-1} = \Delta x_{i-1} = x_{i-1} - x_{i-2} = \ldots = \Delta x_1 = \text{const} = \Delta x$$

$$= \exp[(A(x_i) + A(x_{i-1}) + \ldots + A(x_2) + A(x_1)) \cdot \Delta x] =$$

$$= \exp[(1/i) \cdot (A(x_i) + A(x_{i-1}) + \ldots + A(x_2) + A(x_1)) \cdot (x_i - x_0)].$$

17 июля 2013: Исправление ошибок.

В журнале «Математическое моделирование» РАН мне сообщили, что оказывается широко известно, что для матриц A и B: exp(A+B) НЕ РАВНО exp(A)*exp(B).

1) Это означает, что ФОРМУЛА

$$K(x_i \leftarrow x_0) = \exp[(1/i)\cdot(A(x_i) + A(x_{i-1}) + ... + A(x_2) + A(x_1))\cdot(x_i - x_0)]$$

НЕ ВЕРНА.

2) Также это означает, что следующая ФОРМУЛА ТОЖЕ НЕ ВЕРНА:

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

Но я даже сейчас не очень понимаю, зачем я переусложнял жизнь, когда, по-видимому, можно для участка $(x_j \leftarrow x_i)$ пользоваться формулой из «Теории матриц» Гантмахера:

$\boldsymbol{Y}^*(x \leftarrow x_i) = \exp(Ax)\int_{x_i}^{x} \exp(-At)\boldsymbol{F}(t)dt$, где вместо x наверное надо просто подставить x_j .

19 июля 2013: Исправление исправленного.

Формула из пункта 2), приведенная только что, оказывается всё же ВЕРНА. Это я с перепугу подумал, что она тоже НЕ верна. ВЕРНА формула:

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

потому, что у Гантмахера в «Теории матриц» записано, что для матрицы A справедливо:

$$\exp(Ax)\exp(-Ax_0) = \exp(A(x - x_0)) .$$

Эта формула из Гантмахера верна по-видимому потому, что тут используется одна и та же матрица A, а не матрицы A и B.