МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

А. Ю. Виноградов

МЕТОДЫ РЕШЕНИЯ ЖЕСТКИХ И НЕЖЕСТКИХ КРАЕВЫХ ЗАДАЧ

Монография

Волгоград 2016

УДК 51(075.8)
ББК 22.311я73
В49

Рекомендовано к публикации ученым советом
Межотраслевого научно-исследовательского института
институционального консалтинга

Рецензенты:
Гамонов Евгений Викторович – доктор физико-математических наук,
профессор, старший научный сотрудник SITU IBC;
Воянов Артур Сергеевич – кандидат экономических наук, доцент,
старший научный сотрудник АНОО ДПФО «НИПИ»

Виноградов, А. Ю.

В49 Методы решения жестких и нежестких краевых задач [Текст] : монография / А. Ю. Виноградов. – Волгоград : Изд-во ВолГУ, 2016. – 128 с.

ISBN 978-5-9669-1574-2

На примере расчета цилиндрической оболочки ракеты на прочность предлагаются усовершенствование метода Годунова (для жестких случаев), 3 метода для нежестких случаев, 2 метода для жестких случаев, в том числе и метод расчета оболочек составных и со шпангоутами. Приводятся 3 программы на С++. Приводится перевод на английский язык части материалов.

ISBN 978-5-9669-1574-2

ИЗДАТЕЛЬСТВО ВОЛГОГРАДСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

Оглавление

Глава 1. Введение – известные формулы теории матриц для обыкновенных дифференциальных уравнений

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

Система линейных обыкновенных дифференциальных уравнений имеет вид:

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

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

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

Краевые условия имеют вид:

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

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

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

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

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

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

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

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

Тогда решение задачи Коши может быть записано в виде:

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

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

Из теории матриц [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, как это известно, 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:

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

Глава 2. Усовершенствование метода ортогональной прогонки С.К. Годунова для решения краевых задач с жесткими обыкновенными дифференциальными уравнениями

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

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 коэффициентов будет иметь размерность 8x8, искомая вектор-функция $\boldsymbol{Y}(x)$ 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 [Годунов]:

$$\boldsymbol{Y}(x) = \boldsymbol{Y_1}(x)c_1 + \boldsymbol{Y_2}(x)c_2 + \boldsymbol{Y_3}(x)c_3 + \boldsymbol{Y_4}(x)c_4 + \boldsymbol{Y}^*(x),$$

или можно записать в матричном виде:

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

где векторы $\boldsymbol{Y_1}(x), \boldsymbol{Y_2}(x), \boldsymbol{Y_3}(x), \boldsymbol{Y_4}(x)$ - это линейно независимые вектора-решения однородной системы дифференциальных уравнений, а вектор $\boldsymbol{Y}^*(x)$ - это вектор частного решения неоднородной системы дифференциальных уравнений.

Здесь $\boldsymbol{Y}_{матрица}(x) = \|\boldsymbol{Y_1}(x), \boldsymbol{Y_2}(x), \boldsymbol{Y_3}(x), \boldsymbol{Y_4}(x)\|$ это матрица размерности 8x4, а $\boldsymbol{c}$ это соответствующий вектор размерности 4x1 из 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 $\boldsymbol{Y_1}(x), \boldsymbol{Y_2}(x), \boldsymbol{Y_3}(x), \boldsymbol{Y_4}(x)$, а полностью из всех 8 линейно независимых векторов-решений однородной системы дифференциальных уравнений:

$$\boldsymbol{Y}(x) = \boldsymbol{Y_1}(x)c_1 + \boldsymbol{Y_2}(x)c_2 + \boldsymbol{Y_3}(x)c_3 + \boldsymbol{Y_4}(x)c_4 +$$
$$+ \boldsymbol{Y_5}(x)c_5 + \boldsymbol{Y_6}(x)c_6 + \boldsymbol{Y_7}(x)c_7 + \boldsymbol{Y_8}(x)c_8 + \boldsymbol{Y}^*(x).$$

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 $\boldsymbol{Y_1}(0),\boldsymbol{Y_2}(0),\boldsymbol{Y_3}(0),\boldsymbol{Y_4}(0),\boldsymbol{Y}^*(0)$ векторов $\boldsymbol{Y_1}(x),\boldsymbol{Y_2}(x),\boldsymbol{Y_3}(x),\boldsymbol{Y_4}(x),\boldsymbol{Y}^*(x)$, чтобы можно было начать прогонку с левого края x=0, то есть чтобы удовлетворялись условия $U\boldsymbol{Y}(0) = \boldsymbol{u}$ 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 $\boldsymbol{Y_1}(0),\boldsymbol{Y_2}(0),\boldsymbol{Y_3}(0),\boldsymbol{Y_4}(0),\boldsymbol{Y}^*(0)$ 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:

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

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:

$$U_{орто}\boldsymbol{Y}(0) = \boldsymbol{u_{орто}},$$

где в результате ортонормирования матрицы U вектор $\boldsymbol{u}$ преобразован в вектор $\boldsymbol{u_{орто}}$.

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

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

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

где матрица M размерности 4x8 должна достраивать матрицу $U_{орто}$ 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 $W_{орто}$ размерности 8x8 с ортонормированными строками:

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

Можем записать, что

$$\boldsymbol{Y}_{\textit{матрица}}(0) = (M_{орто})_{транспонированная} = M_{орто}^{T}.$$

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

$$\boldsymbol{Y}(0) = \boldsymbol{Y}_{\textit{матрица}}(0)\boldsymbol{c} + \boldsymbol{Y}^{*}(0)$$

или

$$\boldsymbol{Y}(0) = M_{орто}^{T}\boldsymbol{c} + \boldsymbol{Y}^{*}(0).$$

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

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

Отсюда получаем, что на левом крае константы $\boldsymbol{c}$ уже не на что не влияют, так как

$U_{орто}M^{T}_{орто}=0$ и остается только найти вектор $\boldsymbol{Y}^{*}(0)$ из выражения:

$$U_{орто}\boldsymbol{Y}^{*}(0)=\boldsymbol{u}_{орто}\,.$$

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

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

где $\boldsymbol{0}$ - любой вектор, в том числе вектор из нулей.

Отсюда получаем при помощи обратной матрицы:

$$\boldsymbol{Y}^{*}(0)=\left\|\frac{U_{орто}}{M_{орто}}\right\|^{-1}\left\|\frac{\boldsymbol{u}_{орто}}{\boldsymbol{0}}\right\|.$$

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

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

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

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

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

Начальные значения $\boldsymbol{Y_1}(0), \boldsymbol{Y_2}(0), \boldsymbol{Y_3}(0), \boldsymbol{Y_4}(0), \boldsymbol{Y}^*(0)$ находятся из решения следующих систем линейных алгебраических уравнений:

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

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

где $\boldsymbol{0}$ - вектор из нулей размерности 4x1.

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

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

$$\boldsymbol{Y}(x) = \boldsymbol{Y}_{\textit{матрица}}(x)\boldsymbol{c} + \boldsymbol{Y}^*(x).$$

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:

$$\boldsymbol{Y}_{\textit{матрица}}(x_j) = K(x_j - x_i)\boldsymbol{Y}_{\textit{матрица}}(x_i).$$

Так выполнять вычисления быстрее, особенно для дифференциальных уравнений с постоянными коэффициентами, так как в случае постоянных коэффициентов достаточно вычислить один раз матрицу Коши на малом участке и в последующем лишь умножать на эту однажды вычисленную матрицу Коши.

И аналогично через теорию матриц можно вычислять и вектор $\boldsymbol{Y}^*(x)$ 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.

Глава 3. Метод «переноса краевых условий» (прямой вариант метода) для решения краевых задач с нежесткими обыкновенными дифференциальными уравнениями

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

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

Подставим формулу для $\boldsymbol{Y}(0)$ в краевые условия левого края и получим:

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

Аналогично для правых краевых условий получаем:

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

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

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

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

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

Глава 4. Метод «дополнительных краевых условий» для решения краевых задач с нежесткими обыкновенными дифференциальными уравнениями

Запишем на левом крае ещё одно уравнение краевых условий:

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

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 U 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 $\boldsymbol{m}$ 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 $\boldsymbol{Y}(0)$ из выражения:

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

то есть вектор $\boldsymbol{Y}(0)$ 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:

$$N\boldsymbol{Y}(1) = \boldsymbol{n},$$

где матрица N записывается из тех же соображений дополнительных линейно независимых параметров на правом крае, а вектор $\boldsymbol{n}$ неизвестен.

Для правого края тоже справедлива соответствующая система уравнений:

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

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

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

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

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

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

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

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

и подставим в предыдущую формулу:

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

Таким образом, мы получили систему уравнений вида:

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

где матрица B известна, векторы $\boldsymbol{u}$ и $\boldsymbol{s}$ известны, а векторы $\boldsymbol{m}$ и $\boldsymbol{t}$ неизвестны.

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

$$\left\|\begin{array}{cc}B_{11} & B_{12}\\ B_{21} & B_{22}\end{array}\right\|\cdot\left\|\frac{\boldsymbol{u}}{\boldsymbol{m}}\right\|=\left\|\frac{\boldsymbol{s}}{\boldsymbol{t}}\right\|,$$

откуда можем записать, что

$$B_{11}\boldsymbol{u} + B_{12}\boldsymbol{m} = \boldsymbol{s},$$
$$B_{21}\boldsymbol{u} + B_{22}\boldsymbol{m} = \boldsymbol{t}.$$

Следовательно, искомый вектор $\boldsymbol{m}$ вычисляется по формуле:

$$\boldsymbol{m} = B_{12}^{-1}(\boldsymbol{s} - B_{11}\boldsymbol{u})$$

А искомый вектор $\boldsymbol{n}$ вычисляется через вектор $\boldsymbol{t}$:

$$\boldsymbol{t} = B_{21}\boldsymbol{u} + B_{22}\boldsymbol{m},$$
$$\boldsymbol{n} = \boldsymbol{t} + N \cdot \boldsymbol{Y}^{*}(1 \leftarrow 0).$$

Глава 5. Метод «половины констант» для решения краевых задач с нежесткими обыкновенными дифференциальными уравнениями

Формула для начала счета с левого края только с половиной возможных констант:

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

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

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

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

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

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

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

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

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

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

$$V[K(1 \leftarrow 0)\boldsymbol{Y}(0) + \boldsymbol{Y}^{*}(1 \leftarrow 0)] = \boldsymbol{v},$$
$$VK(1 \leftarrow 0)\boldsymbol{Y}(0) = \boldsymbol{v} - V\boldsymbol{Y}^{*}(1 \leftarrow 0)$$

и подставим в эту формулу выражение для **Y**(0):

$$VK(1 \leftarrow 0)\left\| U_{орто}^{T} \quad M_{орто}^{T} \right\| \left\| \frac{\boldsymbol{u}_{орто}}{\boldsymbol{c}} \right\| = \boldsymbol{v} - V\boldsymbol{Y}^{*}(1 \leftarrow 0)$$

или

$$VK(1 \leftarrow 0)\left\| U_{орто}^{T} \quad M_{орто}^{T} \right\| \left\| \frac{\boldsymbol{u}_{орто}}{\boldsymbol{c}} \right\| = \boldsymbol{p}.$$

Таким образом, мы получили выражение вида:

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

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

$$\left\| D_1 \quad D_2 \right\| \cdot \left\| \frac{\boldsymbol{u}_{орто}}{\boldsymbol{c}} \right\| = \boldsymbol{p}.$$

Тогда можем записать:

$$D_1 \boldsymbol{u}_{орто} + D_2 \boldsymbol{c} = \boldsymbol{p}.$$

Отсюда получаем, что:

$$\boldsymbol{c} = D_2^{-1}(\boldsymbol{p} - D_1 \boldsymbol{u}_{орто}).$$

Таким образом, искомые константы найдены.

Глава 6. Метод «переноса краевых условий» (пошаговый вариант метода) для решения краевых задач с жесткими обыкновенными дифференциальными уравнениями

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

Полное решение системы дифференциальных уравнений имеет вид

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

Или можно записать:

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

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

$$\begin{gathered} U\boldsymbol{Y}(0) = \boldsymbol{u}, \\ U[K(0 \leftarrow x_1)\boldsymbol{Y}(x_1) + \boldsymbol{Y}^*(0 \leftarrow x_1)] = \boldsymbol{u}, \\ UK(0 \leftarrow x_1)\boldsymbol{Y}(x_1) = \boldsymbol{u} - U\boldsymbol{Y}^*(0 \leftarrow x_1). \end{gathered}$$

Или получаем краевые условия, перенесенные в точку x_1:

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

где $U_1 = UK(0 \leftarrow x_1)$ и $\boldsymbol{u}_1 = \boldsymbol{u} - U\boldsymbol{Y}^*(0 \leftarrow x_1)$.

Далее запишем аналогично

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

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

$$\begin{gathered} U_1\boldsymbol{Y}(x_1) = \boldsymbol{u_1}, \\ U_1[K(x_1 \leftarrow x_2)\boldsymbol{Y}(x_2) + \boldsymbol{Y}^*(x_1 \leftarrow x_2)] = \boldsymbol{u_1}, \\ U_1K(x_1 \leftarrow x_2)\boldsymbol{Y}(x_2) = \boldsymbol{u_1} - U_1\boldsymbol{Y}^*(x_1 \leftarrow x_2). \end{gathered}$$

Или получаем краевые условия, перенесенные в точку x_2:

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

где $U_2=U_1K(x_1\leftarrow x_2)$ и $\boldsymbol{u}_2=\boldsymbol{u}_1-U_1\boldsymbol{Y}^*(x_1\leftarrow x_2)$.

И так в точку x^* переносим матричное краевое условие с левого края и таким же образом переносим матричное краевое условие с правого края.

Покажем шаги переноса краевых условий правого края.

Можем записать:

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

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

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

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

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

где $V_{n-1}=VK(1\leftarrow x_{n-1})$ и $\boldsymbol{v}_{n-1}=\boldsymbol{v}-V\boldsymbol{Y}^*(1\leftarrow x_{n-1})$.

Далее запишем аналогично

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

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

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

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

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

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

И так во внутреннюю точку x^* интервала интегрирования переносим матричное краевое условие, как показано, и с левого края и таким же образом переносим матричное краевое условие с правого края и получаем:

$$U^*\boldsymbol{Y}(x^*) = \boldsymbol{u}^*,$$
$$V^*\boldsymbol{Y}(x^*) = \boldsymbol{v}^*.$$

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

$$\left\| \frac{U^*}{V^*} \right\| \cdot \boldsymbol{Y}(x^*) = \left\| \frac{\boldsymbol{u}^*}{\boldsymbol{v}^*} \right\|.$$

6.2. Случай «жестких» дифференциальных уравнений

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

$$U_1\boldsymbol{Y}(x_1) = \boldsymbol{u}_1$$

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

$$U_{1орто}\boldsymbol{Y}(x_1) = \boldsymbol{u}_{1орто}.$$

И теперь уже в это проортонормированное построчно уравнение подставляем

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

И получаем

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

Или получаем краевые условия, перенесенные в точку x_2:

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

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

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

$$U_{2орто}\boldsymbol{Y}(x_2) = \boldsymbol{u}_{2орто}$$

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, которая решается методом Гаусса с выделением главного элемента для получения решения $\boldsymbol{Y}(x^*)$ в рассматриваемой точке x^*:

$$\left\| \frac{U^*_{орто}}{V^*_{орто}} \right\| \cdot \boldsymbol{Y}(x^*) = \left\| \frac{\boldsymbol{u}^*_{орто}}{\boldsymbol{v}^*_{орто}} \right\|.$$

6.3. Формулы для вычисления вектора частного решения неоднородной системы дифференциальных уравнений

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

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

предлагается использовать следующую формулу для каждого отдельного участка интервала интегрирования:

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

Правильность приведенной формулы подтверждается следующим:

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

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

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

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

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

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

что и требовалось подтвердить.

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

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

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

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

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

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:

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

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

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

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

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

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

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

Если рассматриваемый участок интервала интегрирования не мал, то предлагаются следующие итерационные (рекуррентные) формулы.

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

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

Также имеем формулу для отдельного подучастка:

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

Можем записать:

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

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

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

Сравним полученное выражение с формулой:

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

и получим, очевидно, что:

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

и для частного вектора получаем формулу:

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

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

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

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

Сравнив полученное выражение с формулой:

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

очевидно, получаем, что:

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

и вместе с этим получаем формулу для частного вектора:

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

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

6.4. Применяемые формулы ортонормирования

Взято из [Березин, Жидков]. 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:

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

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

Второе уравнение системы заменяется на:

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

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

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

Аналогично поступаем дальше. Уравнение с номером i примет вид:

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

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

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

$$d_i^{/} = b_i - (\overline{a_i}, \overline{c_1}) d_1 - (\overline{a_i}, \overline{c_2}) d_2 - \ldots - (\overline{a_i}, \overline{c_{i-1}}) d_{i-1}.$$

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 $C \cdot C^T = E$, где E - это единичная матрица.

Глава 7. Простейший метод решения краевых задач с жесткими обыкновенными дифференциальными уравнениями без ортонормирования – метод «сопряжения участков интервала интегрирования», которые выражены матричными экспонентами

Идея преодоления трудностей вычислений путем разделения интервала интегрирования на сопрягаемые участки принадлежит д.ф.-м.н. профессору Ю.И. Виноградову, а реализация этой идеи через формулы теории матриц принадлежит к.ф.-м.н. А.Ю. Виноградову.

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

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

Имеем краевые условия в виде:

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

Можем записать матричные уравнения сопряжения участков:

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

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

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

где E - единичная матрица.

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

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

Эта система решается методом Гаусса с выделением главного элемента.

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

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

Применять ортонормирование для краевых задач для жестких обыкновенных дифференциальных уравнений оказывается не надо, так как на каждом участке интервала интегрирования вычисление каждой матричной экспоненты выполняется независимо и от начальной единичной (ортонормированной) матрицы, что делает ненужным применение ортонормирования в отличие от метода Годунова, что значительно упрощает программирование по сравнению с методом Годунова.

Вычислять матрицы Коши можно не в виде матричных экспонент, а при помощи методов типа Рунге-Кутты от стартовой единичной матрицы, а вектор частного решения неоднородной системы дифференциальных уравнений вычислять на каждом участке методами типа Рунге-Кутты следует от стартового нулевого вектора. В случае применения методов типа Рунге-Кутты оценки погрешностей хорошо известны, что означает, что вычисления можно выполнять с заранее известной точностью.

Глава 8. Расчет оболочек составных и со шпангоутами простейшим методом «сопряжения участков интервала интегрирования»

8.1. Вариант записи метода решения жестких краевых задач без ортонормирования – метода «сопряжения участков, выраженных матричными экспонентами» – через положительные направления формул матричного интегрирования дифференциальных уравнений

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:

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

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

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

где E - единичная матрица.
В итоге получаем систему линейных алгебраических уравнений:

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

Эта система решается методом Гаусса с выделением главного элемента.

Оказывается, что применять ортонормирование не нужно, так как участки интервала интегрирования выбираются такой длинны, что счет на них является устойчивым.

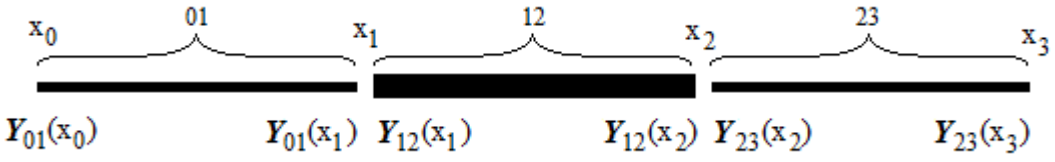
В точках вблизи узлов решение находится путем решения соответствующих задач Коши с началом в i-ом узле:

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

8.2. Составные оболочки вращения

Рассмотрим сопряжения участков составной оболочки вращения.

Пусть имеем 3 участка, где каждый участок может выражаться своими дифференциальными уравнениями и физические параметры могут выражаться по-разному – разными формулами на разных участках:



В общем случае (на примере участка 12) физические параметры участка (вектор $\boldsymbol{P}_{12}(x)$) выражаются через искомые параметры системы обыкновенных дифференциальных уравнений этого участка (через вектор $\boldsymbol{Y}_{12}(x)$) следующим образом:

$$\boldsymbol{P}_{12}(x) = M_{12}\boldsymbol{Y}_{12}(x)\,,$$

где матрица M_{12} - квадратная невырожденная.

При переходе точки сопряжения можем записать в общем виде (но на примере точки сопряжения x_1):

$$\boldsymbol{P}_{01}(x_1)+\Delta\boldsymbol{P}_{01-12}=L_{01-12}\boldsymbol{P}_{12}(x_1),$$

где ΔP_{01-12} - дискретное приращение физических параметров (сил, моментов) при переходе с участка «01» на участок «12», а матрица L_{01-12} квадратная невырожденная диагональная и состоит из единиц и минус единиц на главной диагонали для установления правильного соответствия принятых *положительных направлений* сил, моментов, перемещений и углов при переходе с участка «01» на участок «12», которые могут быть разными (в разных дифференциальных уравнениях разных сопрягаемых участков) – в уравнениях слева от точки сопряжения и в уравнениях справа от точки сопряжения.

Два последних уравнения при объединении образуют уравнение:

$$M_{01}\boldsymbol{Y}_{01}(x_1)+\Delta\boldsymbol{P}_{01-12}=L_{01-12}M_{12}\boldsymbol{Y}_{12}(x_1).$$

В точке сопряжения x_2 аналогично получим уравнение:

$$M_{12}\boldsymbol{Y}_{12}(x_2)+\Delta\boldsymbol{P}_{12-23}=L_{12-23}M_{23}\boldsymbol{Y}_{23}(x_2).$$

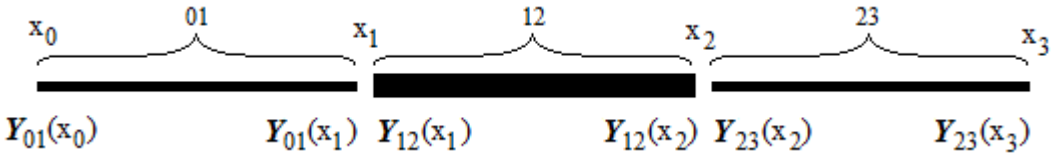
Если бы оболочка состояла бы из одинаковых участков, то мы могли бы записать в объединенном матричном виде систему линейных алгебраических уравнений в следующей форме:

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

Но в нашем случае оболочка состоит из 3 участков, где средний участок можно считать, например, шпангоутом, выражаемым через свои дифференциальные уравнения.

Тогда вместо векторов $\boldsymbol{Y}(x_0)$, $\boldsymbol{Y}(x_1)$, $\boldsymbol{Y}(x_2)$, $\boldsymbol{Y}(x_3)$ мы должны рассмотреть вектора:

$$\boldsymbol{Y}_{01}(x_0),\boldsymbol{Y}_{01}(x_1),\boldsymbol{Y}_{12}(x_1),\boldsymbol{Y}_{12}(x_2),\boldsymbol{Y}_{23}(x_2),\boldsymbol{Y}_{23}(x_3).$$



Тогда матричные уравнения

$$U\boldsymbol{Y}(x_0)=\boldsymbol{u},$$
$$V\boldsymbol{Y}(x_3)=\boldsymbol{v}.$$
$$E\boldsymbol{Y}(x_1)-K(x_1\leftarrow x_0)\boldsymbol{Y}(x_0)=\boldsymbol{Y}^*(x_1\leftarrow x_0),$$
$$E\boldsymbol{Y}(x_2)-K(x_2\leftarrow x_1)\boldsymbol{Y}(x_1)=\boldsymbol{Y}^*(x_2\leftarrow x_1),$$
$$E\boldsymbol{Y}(x_3)-K(x_3\leftarrow x_2)\boldsymbol{Y}(x_2)=\boldsymbol{Y}^*(x_3\leftarrow x_2)$$

примут вид:

$$U\boldsymbol{Y}_{01}(x_0)=\boldsymbol{u},$$
$$V\boldsymbol{Y}_{23}(x_3)=\boldsymbol{v}.$$
$$E\boldsymbol{Y}_{01}(x_1)-K_{01}(x_1\leftarrow x_0)\boldsymbol{Y}_{01}(x_0)=\boldsymbol{Y}_{01}^*(x_1\leftarrow x_0),$$
$$M_{01}\boldsymbol{Y}_{01}(x_1)+\Delta\boldsymbol{P}_{01-12}=L_{01-12}M_{12}\boldsymbol{Y}_{12}(x_1),$$
$$E\boldsymbol{Y}_{12}(x_2)-K_{12}(x_2\leftarrow x_1)\boldsymbol{Y}_{12}(x_1)=\boldsymbol{Y}_{12}^*(x_2\leftarrow x_1),$$
$$M_{12}\boldsymbol{Y}_{12}(x_2)+\Delta\boldsymbol{P}_{12-23}=L_{12-23}M_{23}\boldsymbol{Y}_{23}(x_2),$$
$$E\boldsymbol{Y}_{23}(x_3)-K_{23}(x_3\leftarrow x_2)\boldsymbol{Y}_{23}(x_2)=\boldsymbol{Y}_{23}^*(x_3\leftarrow x_2).$$

После перестановки слагаемых получаем:

$$U\boldsymbol{Y}_{01}(x_0)=\boldsymbol{u},$$
$$V\boldsymbol{Y}_{23}(x_3)=\boldsymbol{v}.$$
$$-K_{01}(x_1\leftarrow x_0)\boldsymbol{Y}_{01}(x_0)+E\boldsymbol{Y}_{01}(x_1)=\boldsymbol{Y}_{01}^*(x_1\leftarrow x_0),$$
$$M_{01}\boldsymbol{Y}_{01}(x_1)-L_{01-12}M_{12}\boldsymbol{Y}_{12}(x_1)=-\Delta\boldsymbol{P}_{01-12},$$
$$-K_{12}(x_2\leftarrow x_1)\boldsymbol{Y}_{12}(x_1)+E\boldsymbol{Y}_{12}(x_2)=\boldsymbol{Y}_{12}^*(x_2\leftarrow x_1),$$
$$M_{12}\boldsymbol{Y}_{12}(x_2)-L_{12-23}M_{23}\boldsymbol{Y}_{23}(x_2)=-\Delta\boldsymbol{P}_{12-23},$$
$$-K_{23}(x_3\leftarrow x_2)\boldsymbol{Y}_{23}(x_2)+E\boldsymbol{Y}_{23}(x_3)=\boldsymbol{Y}_{23}^*(x_3\leftarrow x_2).$$

В итоге мы можем записать итоговую систему линейных алгебраических уравнений:

$$\left\|\begin{array}{cccccc} U & 0 & 0 & 0 & 0 & 0 \\ -K_{01}(x_1 \leftarrow x_0) & E & 0 & 0 & 0 & 0 \\ 0 & M_{01} & -L_{01-12}M_{12} & 0 & 0 & 0 \\ 0 & 0 & -K_{12}(x_2 \leftarrow x_1) & E & 0 & 0 \\ 0 & 0 & 0 & M_{12} & -L_{12-23}M_{23} & 0 \\ 0 & 0 & 0 & 0 & -K_{23}(x_3 \leftarrow x_2) & E \\ 0 & 0 & 0 & 0 & 0 & V \end{array}\right\| \cdot \left\|\begin{array}{c} \boldsymbol{Y}_{01}(x_0) \\ \boldsymbol{Y}_{01}(x_1) \\ \boldsymbol{Y}_{12}(x_1) \\ \boldsymbol{Y}_{12}(x_2) \\ \boldsymbol{Y}_{23}(x_2) \\ \boldsymbol{Y}_{23}(x_3) \end{array}\right\| = \left\|\begin{array}{c} \boldsymbol{u} \\ \boldsymbol{Y}^*_{01}(x_1 \leftarrow x_0) \\ -\Delta\boldsymbol{P}_{01-12} \\ \boldsymbol{Y}^*_{12}(x_2 \leftarrow x_1) \\ -\Delta\boldsymbol{P}_{12-23} \\ \boldsymbol{Y}^*_{23}(x_3 \leftarrow x_2) \\ \boldsymbol{v} \end{array}\right\|$$

Эта система решается методом Гаусса с выделением главного элемента.

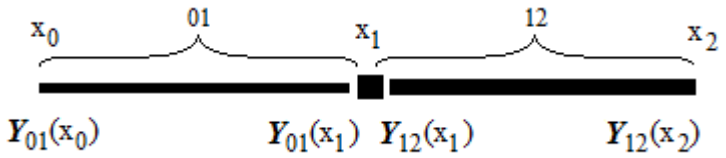
В точках, расположенных между узлами, решение находиться при помощи решения задач Коши с начальными условиями в i-ом узле:

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

Применять ортонормирование для краевых задач для жестких обыкновенных дифференциальных уравнений оказывается не надо.

8.3. Шпангоут, выражаемый не дифференциальными, а алгебраическими уравнениями

Рассмотрим случай, когда шпангоут (в точке x_1) выражается не через дифференциальные уравнения, а через алгебраические уравнения.



Выше мы записывали, что:

$$\boldsymbol{P}_{01}(x_1) + \Delta\boldsymbol{P}_{01-12} = L_{01-12}\boldsymbol{P}_{12}(x_1)$$

Можем представить вектор $\boldsymbol{P}_{01}(x_1)$ силовых факторов и перемещений в виде:

$$\boldsymbol{P}_{01}(x_1) = \left\|\begin{array}{c} \boldsymbol{R}_{01}(x_1) \\ \boldsymbol{S}_{01}(x_1) \end{array}\right\|,$$

где $\boldsymbol{R}_{01}(x_1)$ - вектор перемещений, $\boldsymbol{S}_{01}(x_1)$ - вектор сил и моментов.

Алгебраическое уравнение для шпангоута:

$$G\boldsymbol{R} = \Delta\boldsymbol{S},$$

где G – матрица жесткости шпангоута, $\boldsymbol{R}$ – вектор перемещений шпангоута, $\Delta\boldsymbol{S}$ – вектор силовых факторов, которые действуют на шпангоут.

В точке шпангоута имеем:

$$\Delta\boldsymbol{R} = 0, \Delta\boldsymbol{S} = G\boldsymbol{R},$$

то есть нет разрыва в перемещениях $\Delta\boldsymbol{R} = 0$, но есть результирующий вектор силовых факторов $\Delta\boldsymbol{S} = G\boldsymbol{R}$, который складывается из сил и моментов слева плюс сил и моментов справа от точки шпангоута.

$$\boldsymbol{P}_{01}(x_1) + \left\| \begin{matrix} \Delta\boldsymbol{R} \\ \Delta\boldsymbol{S} \end{matrix} \right\| = L_{01-12}\boldsymbol{P}_{12}(x_1),$$

$$\boldsymbol{P}_{01}(x_1) + \left\| \begin{matrix} \boldsymbol{0} \\ G\boldsymbol{R} \end{matrix} \right\| = L_{01-12}\boldsymbol{P}_{12}(x_1),$$

$$\left\| \begin{matrix} \boldsymbol{R}_{01}(x_1) \\ \boldsymbol{S}_{01}(x_1) \end{matrix} \right\| + \left\| \begin{matrix} \boldsymbol{0} \\ G\boldsymbol{R} \end{matrix} \right\| = L_{01-12} \left\| \begin{matrix} \boldsymbol{R}_{12}(x_1) \\ \boldsymbol{S}_{12}(x_1) \end{matrix} \right\|,$$

$$M_{01}\boldsymbol{Y}_{01}(x_1) + \left\| \begin{matrix} \boldsymbol{0} \\ G\boldsymbol{R} \end{matrix} \right\| = L_{01-12}M_{12}\boldsymbol{Y}_{12}(x_1),$$

$$M_{01}\boldsymbol{Y}_{01}(x_1) + \boldsymbol{g}^* = L_{01-12}M_{12}\boldsymbol{Y}_{12}(x_1), \text{ где } \boldsymbol{g}^* = \left\| \begin{matrix} \boldsymbol{0} \\ G\boldsymbol{R} \end{matrix} \right\|,$$

что справедливо, если мы не забываем, что в данном случае имеем:

$$\boldsymbol{P}_{01}(x_1) = \left\| \begin{matrix} \boldsymbol{R}_{01}(x_1) \\ \boldsymbol{S}_{01}(x_1) \end{matrix} \right\|,$$

то есть вектор перемещений и силовых факторов составляется сначала из перемещений (выше) $\boldsymbol{R}_{01}(x_1)$, а потом из силовых факторов (ниже) $\boldsymbol{S}_{01}(x_1)$.

Здесь необходимо вспомнить, что вектор перемещений $\boldsymbol{R}_{01}(x_1)$ выражается через искомый вектор состояния $\boldsymbol{Y}_{01}(x_1)$:

$$\boldsymbol{g}^* = \begin{Vmatrix} \boldsymbol{0} \\ G\boldsymbol{R} \end{Vmatrix} = \begin{Vmatrix} \boldsymbol{0} \\ G\boldsymbol{R}_{01}(x_1) \end{Vmatrix},$$

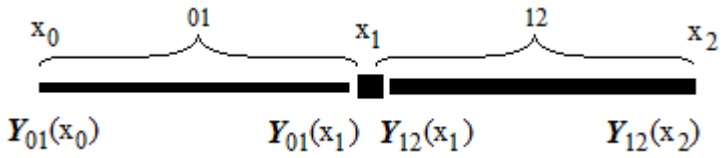
$$\boldsymbol{P}_{01}(x_1) = \begin{Vmatrix} \boldsymbol{R}_{01}(x_1) \\ \boldsymbol{S}_{01}(x_1) \end{Vmatrix} = M_{01}\boldsymbol{Y}_{01}(x_1) = M^{\,p}\boldsymbol{Y}_{01}(x_1) = \begin{Vmatrix} M_{11}^{p} & M_{12}^{p} \\ M_{21}^{p} & M_{22}^{p} \end{Vmatrix} \cdot \boldsymbol{Y}_{01}(x_1),$$

где для удобства было введено переобозначение $M_{01} = M^{p}$.
Тогда можем записать:

$$\boldsymbol{R}_{01}(x_1) = \begin{Vmatrix} M_{11}^{p} & M_{12}^{p} \end{Vmatrix} \cdot \boldsymbol{Y}_{01}(x_1),$$

$$\boldsymbol{g}^* = \begin{Vmatrix} \boldsymbol{0} \\ G\boldsymbol{R}_{01}(x_1) \end{Vmatrix} = \begin{Vmatrix} G \begin{Vmatrix} M_{11}^{p} & M_{12}^{p} \end{Vmatrix} \cdot \boldsymbol{Y}_{01}(x_1) \end{Vmatrix} = \begin{Vmatrix} G \begin{Vmatrix} \boldsymbol{0}..\boldsymbol{0} \\ M_{11}^{p} & M_{12}^{p} \end{Vmatrix} \end{Vmatrix} \cdot \boldsymbol{Y}_{01}(x_1) = \begin{Vmatrix} G \begin{Vmatrix} 0 \\ M_{11}^{p} & M_{12}^{p} \end{Vmatrix} \end{Vmatrix} \cdot \boldsymbol{Y}_{01}(x_1)$$

Запишем матричные уравнения для этого случая:



$$U\boldsymbol{Y}_{01}(x_0) = \boldsymbol{u},$$

$$V\boldsymbol{Y}_{12}(x_2) = \boldsymbol{v}.$$

$$E\boldsymbol{Y}_{01}(x_1) - K_{01}(x_1 \leftarrow x_0)\boldsymbol{Y}_{01}(x_0) = \boldsymbol{Y}_{01}^{*}(x_1 \leftarrow x_0),$$

$$M_{01}\boldsymbol{Y}_{01}(x_1) + \boldsymbol{g}^* = L_{01\text{–}12}M_{12}\boldsymbol{Y}_{12}(x_1),$$

$$E\boldsymbol{Y}_{12}(x_2) - K_{12}(x_2 \leftarrow x_1)\boldsymbol{Y}_{12}(x_1) = \boldsymbol{Y}_{12}^{*}(x_2 \leftarrow x_1).$$

Распишем здесь в уравнении вектор $\boldsymbol{g}^*$:

$$M_{01}\boldsymbol{Y}_{01}(x_1) + \begin{Vmatrix} G \begin{Vmatrix} 0 \\ M_{11}^{p} & M_{12}^{p} \end{Vmatrix} \end{Vmatrix} \cdot \boldsymbol{Y}_{01}(x_1) = L_{01\text{–}12}M_{12}\boldsymbol{Y}_{12}(x_1),$$

$$(M_{01} + \begin{Vmatrix} G \begin{Vmatrix} 0 \\ M_{11}^{p} & M_{12}^{p} \end{Vmatrix} \end{Vmatrix}) \cdot \boldsymbol{Y}_{01}(x_1) = L_{01\text{–}12}M_{12}\boldsymbol{Y}_{12}(x_1).$$

Для обеспечения негромоздкости введем обозначение:

$$(M_{01} + \begin{Vmatrix} G \begin{Vmatrix} 0 \\ M_{11}^{p} & M_{12}^{p} \end{Vmatrix} \end{Vmatrix}) = M^{*}.$$

Тогда уравнение

$$M_{01}\boldsymbol{Y}_{01}(x_1)+\boldsymbol{g}^{*}=L_{01\text{–}12}M_{12}\boldsymbol{Y}_{12}(x_1)$$

примет вид:

$$M^{*}\boldsymbol{Y}_{01}(x_1)=L_{01\text{–}12}M_{12}\boldsymbol{Y}_{12}(x_1).$$

Для удобства переставим слагаемые в матричных уравнениях, чтобы итоговая система линейных алгебраических уравнений записывалась очевидно:

$$\begin{gathered}U\boldsymbol{Y}_{01}(x_0)=\boldsymbol{u},\\ V\boldsymbol{Y}_{12}(x_2)=\boldsymbol{v}.\\ -K_{01}(x_1\leftarrow x_0)\boldsymbol{Y}_{01}(x_0)+E\boldsymbol{Y}_{01}(x_1)=\boldsymbol{Y}^{*}_{01}(x_1\leftarrow x_0),\\ M^{*}\boldsymbol{Y}_{01}(x_1)-L_{01\text{–}12}M_{12}\boldsymbol{Y}_{12}(x_1)=\boldsymbol{0},\\ -K_{12}(x_2\leftarrow x_1)\boldsymbol{Y}_{12}(x_1)+E\boldsymbol{Y}_{12}(x_2)=\boldsymbol{Y}^{*}_{12}(x_2\leftarrow x_1).\end{gathered}$$

Таким образом, получаем итоговую систему линейных алгебраических уравнений:

$$\left\|\begin{array}{c|c|c|c}U & 0 & 0 & 0\\ \hline -K_{01}(x_1\leftarrow x_0) & E & 0 & 0\\ \hline 0 & M^{*} & -L_{01\text{–}12}M_{12} & 0\\ \hline 0 & 0 & -K_{12}(x_2\leftarrow x_1) & E\\ \hline 0 & 0 & 0 & V\end{array}\right\|\cdot\left\|\begin{array}{c}\boldsymbol{Y}_{01}(x_0)\\ \hline \boldsymbol{Y}_{01}(x_1)\\ \hline \boldsymbol{Y}_{12}(x_1)\\ \hline \boldsymbol{Y}_{12}(x_2)\end{array}\right\|=\left\|\begin{array}{c}\boldsymbol{u}\\ \hline \boldsymbol{Y}^{*}_{01}(x_1\leftarrow x_0)\\ \hline \boldsymbol{0}\\ \hline \boldsymbol{Y}^{*}_{12}(x_2\leftarrow x_1)\\ \hline \boldsymbol{v}\end{array}\right\|.$$

Если к шпангоуту приложено внешнее силовое-моментное воздействие $\boldsymbol{g}^{p}$, то

$\boldsymbol{g}^{*}=\left\|\begin{array}{c}\boldsymbol{0}\\ \hline G\boldsymbol{R}\end{array}\right\|$ следует переписать в виде $\boldsymbol{g}^{*}=\left\|\begin{array}{c}\boldsymbol{0}\\ \hline G\boldsymbol{R}+\boldsymbol{g}^{p}\end{array}\right\|=\left\|\begin{array}{c}\boldsymbol{0}\\ \hline G\boldsymbol{R}_{01}(x_1)+\boldsymbol{g}^{p}\end{array}\right\|$, тогда:

$$\boldsymbol{g}^{*}=\left\|\begin{array}{c}\boldsymbol{0}\\ \hline G\left\|M^{p}_{11}\quad M^{p}_{12}\right\|\cdot\boldsymbol{Y}_{01}(x_1)+\boldsymbol{g}^{p}\end{array}\right\|=\left\|\begin{array}{c}\boldsymbol{0}..\boldsymbol{0}\\ \hline G\left\|M^{p}_{11}\quad M^{p}_{12}\right\|\end{array}\right\|\cdot\boldsymbol{Y}_{01}(x_1)+\left\|\begin{array}{c}\boldsymbol{0}\\ \hline \boldsymbol{g}^{p}\end{array}\right\|.$$

Тогда матричное уравнение

$$M_{01}\boldsymbol{Y}_{01}(x_1)+\left\|\begin{array}{c}0\\ \hline G\left\|M^{p}_{11}\quad M^{p}_{12}\right\|\end{array}\right\|\cdot\boldsymbol{Y}_{01}(x_1)=L_{01\text{–}12}M_{12}\boldsymbol{Y}_{12}(x_1)$$

примет вид:

$$M_{01}\boldsymbol{Y}_{01}(x_1)+\left\|\begin{matrix}0 & \\ G\|M_{11}^{p} & M_{12}^{p}\end{matrix}\right\|\cdot\boldsymbol{Y}_{01}(x_1)+\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|=L_{01-12}M_{12}\boldsymbol{Y}_{12}(x_1),$$

$$M^{*}\boldsymbol{Y}_{01}(x_1)-L_{01-12}M_{12}\boldsymbol{Y}_{12}(x_1)=-\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|.$$

Итоговая система линейных алгебраических уравнений примет вид:

$$\left\|\begin{array}{c|c|c|c} U & 0 & 0 & 0 \\ \hline -K_{01}(x_1\leftarrow x_0) & E & 0 & 0 \\ \hline 0 & M^{*} & -L_{01-12}M_{12} & 0 \\ \hline 0 & 0 & -K_{12}(x_2\leftarrow x_1) & E \\ \hline 0 & 0 & 0 & V \end{array}\right\|\cdot\left\|\begin{array}{c}\boldsymbol{Y}_{01}(x_0)\\ \hline \boldsymbol{Y}_{01}(x_1)\\ \hline \boldsymbol{Y}_{12}(x_1)\\ \hline \boldsymbol{Y}_{12}(x_2)\end{array}\right\|=\left\|\begin{array}{c}\boldsymbol{u}\\ \hline \boldsymbol{Y}_{01}^{*}(x_1\leftarrow x_0)\\ \hline -\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|\\ \hline \boldsymbol{Y}_{12}^{*}(x_2\leftarrow x_1)\\ \hline \boldsymbol{v}\end{array}\right\|.$$

8.4. Случай, когда уравнения (оболочки и шпангоута) выражаются не через абстрактные вектора, а через вектора, состоящие из конкретных физических параметров

Рассмотрим случай, когда части оболочечной конструкции и шпангоут выражаются через вектора состояния (типа $\boldsymbol{Y}_{12}(x)$), которые (в частном случае) совпадают с векторами физических параметров (типа $\boldsymbol{P}_{12}(x)$ - перемещения, угол, силы, момент). Тогда матрицы типа M_{12} будут единичными: $M_{12}=E$. И пусть положительные направления физических параметров одинаковы для всех частей оболочки и шпангоута ($L_{01-12}=E$).

Тогда будем иметь уравнения:

$$\boldsymbol{P}_{12}(x)=M_{12}\boldsymbol{Y}_{12}(x),$$

$$\boldsymbol{P}_{01}(x_1)+\Delta\boldsymbol{P}_{01-12}=L_{01-12}\boldsymbol{P}_{12}(x_1),$$

$$M_{01}\boldsymbol{Y}_{01}(x_1)+\Delta\boldsymbol{P}_{01-12}=L_{01-12}M_{12}\boldsymbol{Y}_{12}(x_1),$$

в виде:

$$\boldsymbol{P}_{12}(x)=E\boldsymbol{Y}_{12}(x),$$

$$\boldsymbol{P}_{01}(x_1)+\Delta\boldsymbol{P}_{01-12}=E\boldsymbol{P}_{12}(x_1),$$

$$E\boldsymbol{Y}_{01}(x_1)+\Delta\boldsymbol{P}_{01-12}=E\boldsymbol{Y}_{12}(x_1),$$

где E – единичная матрица.

Уравнения

$$M_{01}\boldsymbol{Y}_{01}(x_1)+\boldsymbol{g}^{*}=L_{01\text{–}12}M_{12}\boldsymbol{Y}_{12}(x_1),$$

$$\boldsymbol{P}_{01}(x_1)+\left\|\begin{matrix}\boldsymbol{0}\\ G\boldsymbol{R}_{01}(x_1)+\boldsymbol{g}^{p}\end{matrix}\right\|=L_{01\text{–}12}\boldsymbol{P}_{12}(x_1),$$

$$\boldsymbol{P}_{01}(x_1)=\left\|\begin{matrix}\boldsymbol{R}_{01}(x_1)\\ \boldsymbol{S}_{01}(x_1)\end{matrix}\right\|=M_{01}\boldsymbol{Y}_{01}(x_1)=M^{p}\boldsymbol{Y}_{01}(x_1)=\left\|\begin{matrix}M_{11}^{p} & M_{12}^{p}\\ M_{21}^{p} & M_{22}^{p}\end{matrix}\right\|\cdot\boldsymbol{Y}_{01}(x_1),$$

$$\boldsymbol{R}_{01}(x_1)=\left\|M_{11}^{p}\quad M_{12}^{p}\right\|\cdot\boldsymbol{Y}_{01}(x_1),$$

$$\boldsymbol{g}^{*}=\left\|\begin{matrix}\boldsymbol{0}\\ G\left\|M_{11}^{p}\quad M_{12}^{p}\right\|\cdot\boldsymbol{Y}_{01}(x_1)\end{matrix}\right\|+\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|=\left\|\begin{matrix}0\\ G\left\|M_{11}^{p}\quad M_{12}^{p}\right\|\end{matrix}\right\|\cdot\boldsymbol{Y}_{01}(x_1)+\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|,$$

примут вид:

$$E\boldsymbol{Y}_{01}(x_1)+\boldsymbol{g}^{*}=E\boldsymbol{Y}_{12}(x_1),$$

$$E\boldsymbol{Y}_{01}(x_1)+\left\|\begin{matrix}\boldsymbol{0}\\ G\boldsymbol{R}_{01}(x_1)+\boldsymbol{g}^{p}\end{matrix}\right\|=E\boldsymbol{Y}_{12}(x_1),$$

$$\boldsymbol{P}_{01}(x_1)=\left\|\begin{matrix}\boldsymbol{R}_{01}(x_1)\\ \boldsymbol{S}_{01}(x_1)\end{matrix}\right\|=E\boldsymbol{Y}_{01}(x_1)=M^{p}\boldsymbol{Y}_{01}(x_1)=\left\|\begin{matrix}E & 0\\ 0 & E\end{matrix}\right\|\cdot\boldsymbol{Y}_{01}(x_1),$$

$$\boldsymbol{R}_{01}(x_1)=\left\|E\quad 0\right\|\cdot\boldsymbol{Y}_{01}(x_1),$$

$$\boldsymbol{g}^{*}=\left\|\begin{matrix}\boldsymbol{0}\\ G\boldsymbol{R}_{01}(x_1)\end{matrix}\right\|+\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|=\left\|\begin{matrix}\boldsymbol{0}\\ G\left\|E\quad 0\right\|\cdot\boldsymbol{Y}_{01}(x_1)\end{matrix}\right\|+\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|=\left\|\begin{matrix}0\\ G\left\|E\quad 0\right\|\end{matrix}\right\|\cdot\boldsymbol{Y}_{01}(x_1)+\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|.$$

А уравнения

$$M_{01}\boldsymbol{Y}_{01}(x_1)+\left\|\begin{matrix}0\\ G\left\|M_{11}^{p}\quad M_{12}^{p}\right\|\end{matrix}\right\|\cdot\boldsymbol{Y}_{01}(x_1)+\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|=L_{01\text{–}12}M_{12}\boldsymbol{Y}_{12}(x_1),$$

$$M^{*}\boldsymbol{Y}_{01}(x_1)-L_{01\text{–}12}M_{12}\boldsymbol{Y}_{12}(x_1)=-\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|.$$

примут вид:

$$E\boldsymbol{Y}_{01}(x_1)+\left\|\begin{matrix}0\\ G\left\|E\quad 0\right\|\end{matrix}\right\|\cdot\boldsymbol{Y}_{01}(x_1)+\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|=E\boldsymbol{Y}_{12}(x_1),$$

$$M^{*}\boldsymbol{Y}_{01}(x_1)-E\boldsymbol{Y}_{12}(x_1)=-\left\|\begin{matrix}\boldsymbol{0}\\ \boldsymbol{g}^{p}\end{matrix}\right\|,\ \text{где}\ (E+\left\|\begin{matrix}0\\ G\left\|E\quad 0\right\|\end{matrix}\right\|)=M^{*}$$

Итоговая система линейных алгебраических уравнений

$$\left\|\begin{array}{c|c|c|c} U & 0 & 0 & 0 \\ \hline -K_{01}(x_1 \leftarrow x_0) & E & 0 & 0 \\ \hline 0 & M^* & -L_{01-12}M_{12} & 0 \\ \hline 0 & 0 & -K_{12}(x_2 \leftarrow x_1) & E \\ \hline 0 & 0 & 0 & V \end{array}\right\| \cdot \left\|\begin{array}{c} \boldsymbol{Y}_{01}(x_0) \\ \hline \boldsymbol{Y}_{01}(x_1) \\ \hline \boldsymbol{Y}_{12}(x_1) \\ \hline \boldsymbol{Y}_{12}(x_2) \end{array}\right\| = \left\|\begin{array}{c} \boldsymbol{u} \\ \hline \boldsymbol{Y}^*_{01}(x_1 \leftarrow x_0) \\ \hline -\left\|\begin{array}{c} \boldsymbol{0} \\ \boldsymbol{g}^p \end{array}\right\| \\ \hline \boldsymbol{Y}^*_{12}(x_2 \leftarrow x_1) \\ \hline \boldsymbol{v} \end{array}\right\|$$

примет вид:

$$\left\|\begin{array}{c|c|c|c} U & 0 & 0 & 0 \\ \hline -K_{01}(x_1 \leftarrow x_0) & E & 0 & 0 \\ \hline 0 & M^* & -E & 0 \\ \hline 0 & 0 & -K_{12}(x_2 \leftarrow x_1) & E \\ \hline 0 & 0 & 0 & V \end{array}\right\| \cdot \left\|\begin{array}{c} \boldsymbol{Y}_{01}(x_0) \\ \hline \boldsymbol{Y}_{01}(x_1) \\ \hline \boldsymbol{Y}_{12}(x_1) \\ \hline \boldsymbol{Y}_{12}(x_2) \end{array}\right\| = \left\|\begin{array}{c} \boldsymbol{u} \\ \hline \boldsymbol{Y}^*_{01}(x_1 \leftarrow x_0) \\ \hline -\left\|\begin{array}{c} \boldsymbol{0} \\ \boldsymbol{g}^p \end{array}\right\| \\ \hline \boldsymbol{Y}^*_{12}(x_2 \leftarrow x_1) \\ \hline \boldsymbol{v} \end{array}\right\|,$$

где $M^* = (\underset{8x8}{E} + \left\|\begin{array}{cc} 0 & \\ \underset{4x4}{G} & \underset{4x4}{E} \end{array}\right\| \begin{array}{c} \\ \underset{4x8}{} \\ \\ \end{array} \begin{array}{c} \\ \\ \underset{4x4}{0} \end{array}\right\|) = (\underset{8x8}{E} + \left\|\begin{array}{cc} 0 & \\ \underset{4x4}{G} & \underset{4x4}{0} \end{array}\right\| \begin{array}{c} \\ \underset{4x8}{} \\ \\ \end{array} \begin{array}{c} \\ \\ \underset{4x4}{0} \end{array}\right\|) = \left\|\begin{array}{cc} \underset{4x4}{E} & \underset{4x4}{0} \\ \underset{4x4}{G} & \underset{4x4}{E} \end{array}\right\|.$

Это означает, что уравнение

$$M^* \boldsymbol{Y}_{01}(x_1) - E\boldsymbol{Y}_{12}(x_1) = -\left\|\begin{array}{c} \boldsymbol{0} \\ \boldsymbol{g}^p \end{array}\right\|$$

принимает вид:

$$\left\|\begin{array}{cc} \underset{4x4}{E} & \underset{4x4}{0} \\ \underset{4x4}{G} & \underset{4x4}{E} \end{array}\right\| \boldsymbol{Y}_{01}(x_1) - E\boldsymbol{Y}_{12}(x_1) = -\left\|\begin{array}{c} \boldsymbol{0} \\ \boldsymbol{g}^p \end{array}\right\|$$

$$\left\|\begin{array}{cc} \underset{4x4}{E} & \underset{4x4}{0} \\ \underset{4x4}{G} & \underset{4x4}{E} \end{array}\right\| \left\|\begin{array}{c} \boldsymbol{R}_{01}(x_1) \\ \boldsymbol{S}_{01}(x_1) \end{array}\right\| - \left\|\begin{array}{cc} \underset{4x4}{E} & \underset{4x4}{0} \\ \underset{4x4}{0} & \underset{4x4}{E} \end{array}\right\| \left\|\begin{array}{c} \boldsymbol{R}_{12}(x_1) \\ \boldsymbol{S}_{12}(x_1) \end{array}\right\| = -\left\|\begin{array}{c} \boldsymbol{0} \\ \boldsymbol{g}^p \end{array}\right\|$$

$\boldsymbol{R}_{01}(x_1) - \boldsymbol{R}_{12}(x_1) = \boldsymbol{0}$, (нет скачка в перемещениях и угле) и

$G\boldsymbol{R}_{01}(x_1) + \boldsymbol{S}_{01}(x_1) - \boldsymbol{S}_{12}(x_1) = -\boldsymbol{g}^p$ - равновесие шпангоута,

то есть:

$\boldsymbol{R}_{01}(x_1) = \boldsymbol{R}_{12}(x_1)$ (перемещения и угол: нет разрыва)

$\boldsymbol{S}_{01}(x_1) + \Delta\boldsymbol{S} + \boldsymbol{g}^p = \boldsymbol{S}_{12}(x_1)$, где $\Delta\boldsymbol{S} = G\boldsymbol{R}_{01}(x_1)$ (силы и момент: равновесие).

Список литературы

1. Гантмахер Ф.Р. Теория матриц. М.: Наука, 1988. 548с.
2. Годунов С.К. О численном решении краевых задач для систем линейных обыкновенных дифференциальных уравнений // Успехи математических наук. 1961. т.16. Вып. 3 (99). С. 171-174.
3. Березин И.С., Жидков Н.П. Методы вычислений: том II. М.: Государственное издательство физико-математической литературы, 1962. 635 с.

Список опубликованных статей по теме

1. Виноградов А.Ю. Вычисление начальных векторов для численного решения краевых задач // Деп. в ВИНИТИ. 1994. N 2073-В94. 15 с.
2. Виноградов А.Ю., Виноградов Ю.И. Совершенствование метода прогонки Годунова для задач строительной механики // Изв. РАН Механика твердого тела. 1994. №4. С. 187-191.
3. Виноградов А.Ю. Вычисление начальных векторов для численного решения краевых задач //Журнал вычислительной математики и математической физики. 1995. Т.35. №1. С. 156-159.
4. Виноградов А.Ю. Численное моделирование произвольных краевых условий для задач строительной механики тонкостенных конструкций // Тез. докладов Белорусского Конгресса по теоретической и прикладной механике "Механика-95". Минск, 6-11 февраля 1995 г., Гомель: Изд- во ИММС АНБ, 1995. С. 63-64.
5. Vinogradov A.Yu. Numerical modeling of boundary conditions in deformation problems of structured material in thin wall constructions // International Symposium "Advances in Structured and Heterogeneous Continua II". Book Absfracts. August, 14-16, 1995, Moscow, Russia. P.51.
6. Виноградов А.Ю. Численное моделирование краевых условий в задачах деформирования тонкостенных конструкций из композиционных материалов // Механика Композиционных Материалов и Конструкций. 1995. Т.I. N2. С. 139-143.
7. Виноградов А.Ю. Приведение краевых задач механики элементов приборных устройств к задачам Коши для выбранной точки // Прикладная механика в приборных устройствах. Меж вуз. сб. научных трудов. Москва: МИРЭА, 1996.
8. Виноградов А.Ю. Модификация метода Годунова // Труды Международной научно-технической конференции "Современные проблемы машиноведения". Гомель: ГПИ им. П.О. Сухого, 1996. С.39-41.
9. Виноградов А.Ю., Виноградов Ю.И. Методы переноса сложных краевых условий для жестких дифференциальных уравнений строительной механики // Труды Международной конференции «Ракетно-космическая техника: фундаментальные проблемы механики и теплообмена». Москва, 1998.
10. Виноградов А.Ю. Метод решения краевых задач путем переноса условий с краев интервала интегрирования в произвольную

точку // Тез. докладов Международной конференции "Актуальные проблемы механики оболочек". Казань, 2000. С. 176.

11. Виноградов Ю.И., Виноградов А.Ю., Гусев Ю.А. Метод переноса краевых условий для дифференциальных уравнений теории оболочек // Труды Международной конференции "Актуальные проблемы механики оболочек". Казань, 2000. С. 128-132.

12. Виноградов А.Ю., Виноградов Ю.И. Метод переноса краевых условий функциями Коши-Крылова для жестких линейных обыкновенных дифференциальных уравнений. // ДАН РФ. 2000. Т. 373. №4. С. 474-476.

13. Виноградов А.Ю., Виноградов Ю.И. Функции Коши-Крылова и алгоритмы решения краевых задач теории оболочек // ДАН РФ. 2000. Т. 375. №3. С. 331-333.

14. Виноградов А.Ю. Численные методы переноса краевых условий // Математическое моделирование. 2000. Т. 12. № 7. С. 3-6.

15. Виноградов А.Ю., Гусев Ю.А. Перенос краевых условий функциями Коши-Крылова в задачах строительной механики // Тез. докладов VIII Всероссийского съезда по теоретической и прикладной механике. Пермь, 2001. С. 109-110.

16. Виноградов Ю.И., Виноградов А.Ю. Перенос краевых условий функциями Коши-Крылова // Тез. докладов Международной конференции "Dynamical System Modeling and Stability Investigation". "DSMSI-2001". Киев, 2001.

17. Виноградов А.Ю., Виноградов Ю.И., Гусев Ю.А, Клюев Ю.И. Перенос краевых условий функциями Коши-Крылова и его свойства// Изв. РАН МТТ. 2001. №2. С. 155-161.

18. Виноградов Ю.И., Виноградов А.Ю., Гусев Ю.А.Численный метод переноса краевых условий для жестких дифференциальных уравнений строительной механики // Математическое моделирование. 2002. Т. 14. №9. С. 3-8.

19. Виноградов Ю.И., Виноградов А.Ю. Простейший метод решения жестких краевых задач // Фундаментальные исследования. 2014. № 12–12. С. 2569-2574.

20. Виноградов Ю.И., Виноградов А.Ю. Решение жестких краевых задач строительной механики (расчет оболочек составных и со шпангоутами) методом Виноградовых (без ортонормирования) // Современные проблемы науки и образования. 2015. №1-1.

Приложение 1. Программа на C++ расчета цилиндрической оболочки - для метода из главы 6

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.

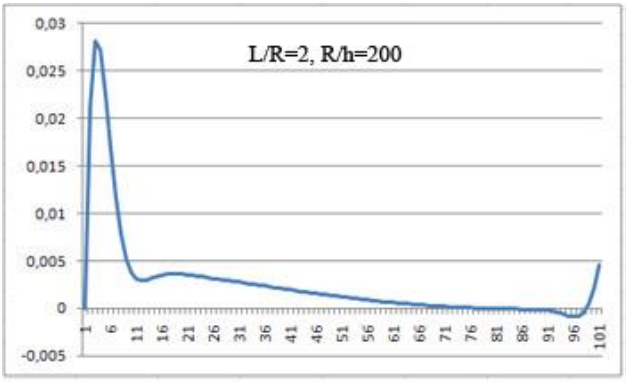
Приводятся графики изгибающего обезразмеренного момента M1:

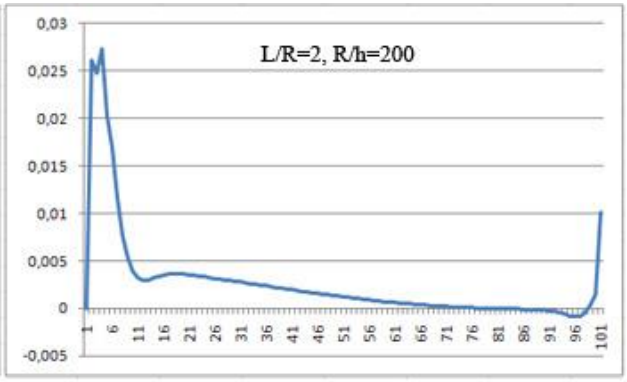
– слева приводятся графики, полученные при использовании операций построчного ортонормирования на каждом из 100 шагов, на которые разделялся участок между краями цилиндра,

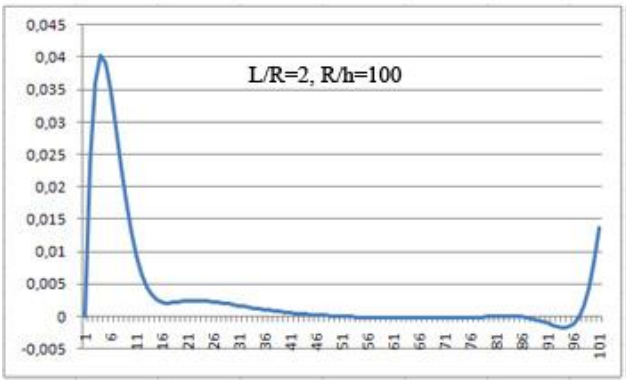
– справа приводятся графики, полученные совсем без применения операций построчного ортонормирования.

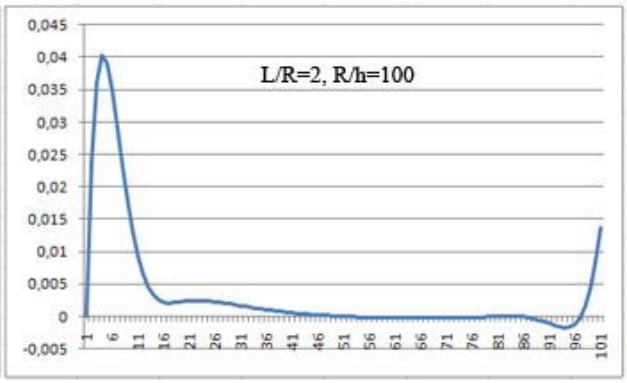
Следует сказать, что в качестве расчетной среды использовалась 32-х битная операционная система Windows XP и среда программирования Microsoft Visual Studio 2010 (Visual C++) использовалась в тех же рамках 32-х битной организации операций с числами. Параметры компьютера такие: ноутбук ASUS M51V (CPU Duo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


L/R=2, R/h=200
0,03
0,025
0,02
0,015
0,01
0,005
0
-0,005
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101


L/R=2, R/h=200
0,03
0,025
0,02
0,015
0,01
0,005
0
-0,005
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101


L/R=2, R/h=100
0,045
0,04
0,035
0,03
0,025
0,02
0,015
0,01
0,005
0
-0,005
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101


L/R=2, R/h=100
0,045
0,04
0,035
0,03
0,025
0,02
0,015
0,01
0,005
0
-0,005
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101

Приложение 2. 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;
}
```

Приложение 3. Программа на C++ расчета сферической оболочки (переменные коэффициенты) - для метода из главы 6

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);//M
омент 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;
}
```

Приложение 4. Программа на C++ расчета цилиндра – для метода из главы 7

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.

Приложение 5. Программа на C++ (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;
}
```

Приложение 6. Метод главы 7 (лучший метод из предложенных) и программа на C++ на английском языке

The simplest method of solution of the stiff boundary value problems – the method of solution of the stiff boundary value problems without orthonormalization
offered by Y. I. Vinogradov and A. Y. Vinogradov.

Alexey Yurievich Vinogradov (Candidate of science /Physics and Mathematics/)
(14 November 2011, Moscow, Russia)

The authors: Yuri Ivanovich Vinogradov (1938 year of birth), doctor of science (Physics and Mathematics) and Alexey Yurievich Vinogradov (1970 year of birth), candidate of science (Physics and Mathematics).

The idea of overcoming the problems of computing by dividing an integration interval into matching sectors belongs to Y. I. Vinogradov (doctor of science /Physics and Mathematics/). His doctorate thesis was based on that material (including that idea).

Proposed realization of this idea (division and matching) through matrix theory formulas i.e. by means of the matrix exponents (or by other words by means of a matrizant – the Cauchy matrix, Cauchy-Krylov functions) belongs to A. Y. Vinogradov (candidate of science /Physics and Mathematics/).

AT PRESENT MOMENT THIS METHOD IS THE BEST ONE. The method was finalized in the mid October 2011 and a C++ software program (that is given in this paper as well) was finalized and checked by comparative computations on 14 November 2011.

1. The simplest method of solution of the stiff boundary value problems.

Let's divide a boundary value problem integration interval e.g. for three segments. We will have points (nodes) including the boundaries:

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

We have the boundary conditions in the following form:

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

We can write sector matching matrix equations as follows:

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

It is possible to rewrite these formulas in the more convenient form:

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

where E - unit matrix.

Then in the combined matrix form we obtain a system of the linear algebraic equations in the following form:

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

This system is solved by the Gauss method by discrimination of the basic element.

Solution in the points located between the nodes is obtained by solving the Cauchy problem with initial conditions in the i^{th} node:

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

It appears to be that there is no need to apply orthonormalization for the boundary value problems for stiff ordinary differential equations.

2. Detailed description of the designations used above.

Let's use as an example a system of differential equations of the cylindrical shell of the rocket – a system of the ordinary differential equations of 8^{th} order (after partial derivatives partitioning by the Fourier method).

A system of the linear ordinary differential equations has the following form:

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

where $\boldsymbol{Y}(x)$ - an unknown vector-function of the 8x1 dimensionality problem, $\boldsymbol{Y}'(x)$ - a derivative of the unknown vector-function of 8x1 dimensionality, A - a square matrix of the coefficients of the differential equation of 8x8 dimensionality, $\boldsymbol{F}(x)$ - a vector-function of the external influence on the system of 8x1 dimensionality.

The boundary conditions have the following form:

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

where $\boldsymbol{Y}(0)$ - is the value of the unknown vector-function at the left boundary x=0 of the 8x1 dimensionality, U - a square horizontal matrix of the coefficients of the boundary conditions of the left boundary of the 4x8 dimensionality, and $\boldsymbol{u}$ - an external influence on the left boundary of the 4x1 dimensionality.

$\boldsymbol{Y}(1)$ is the value of the unknown vector-function at the right boundary x=1 of the 8x1 dimensionality, V - a square horizontal matrix of the coefficients of the boundary conditions of the right boundary of the 4x8 dimensionality, and $\boldsymbol{v}$ - an external influence on the left boundary of the 4x1 dimensionality.

In case when a system of differential equations has a matrix with constant coefficients A= const the solution of the Cauchy problem has the following form:

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

where $e^{A(x-x0)} = E + A(x-x_0) + A^2(x-x_0)^2/2! + A^3(x-x_0)^3/3! + \ldots,$

where E is a unit matrix.

Matrix exponent can be named as the Cauchy matrix or matrizant and be denoted in the following form:

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

Then Cauchy problem solution can be written in the following form:

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

where $\boldsymbol{Y}^*(x \leftarrow x_0) = e^{Ax} \int_{x0}^{x} e^{-At}\boldsymbol{F}(t)dt$ is a vector of the partial solution of the heterogeneous system of the differential equations.

A matrix exponent (the Cauchy matrix) multiplication feature is known from the matrix theory [1]:

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

In case when a differential equation system has a matrix with variable coefficients $A = A(x)$ it is offered to find a solution of the Cauchy problem by means of the Cauchy matrix multiplication feature. That is the integration integral is split into small segments and the Cauchy matrixes are computed approximately in these small segments by means of the formula for a constant matrix in the exponent. Then the Cauchy matrixes computed in these small sections are multiplied:

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

where the Cauchy matrixes are computed approximately by the following formula:

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

Instead of the formula for computation of the vector of the partial solution of the heterogeneous system of the differential equations in the form [1]:

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

it is proposed to use the following formula for each individual segment of the integration interval:

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

Correctness of the above mentioned formula is justified by the following:

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

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

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

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

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

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

as was to be justified.

Computation of the vector of the partial solution of the heterogeneous system of differential equations is carried out by representing the Cauchy matrix under the integral sign in the form of the series and integrating this series element by element:

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

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

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

This formula is correct for a system of differential equations with the constant coefficient matrix A=const.

$\boldsymbol{F}(t)$ vector can be considered in the segment $(x_j - x_i)$ approximately in the form of the constant value $\boldsymbol{F}(x_i) = \text{constant}$ that allows its taking off from the integral sign that results in very simple series for computation in the considered segment.

In case of the differential equations with variable coefficients the averaged matrix $A_i = A(x_i)$ of the coefficients of the differential equation system can be used in the above mentioned formula for each segment.

Let's consider an option when the integration interval steps are selected as sufficiently small that results in considering $\boldsymbol{F}(t)$ vector in the segment $(x_j - x_i)$ approximately in the form of the constant value $\boldsymbol{F}(x_i) = \text{constant}$resulting in taking off this vector from the integral signs:

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

It is known that at T=(at+b) we have:

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

In our case we have:

$$\int (\mathrm{b}-\mathrm{t})^{\mathrm{n}} dt = \frac{1}{(\text{-}1)(\mathrm{n}+1)}(\mathrm{b}-\mathrm{t})^{\mathrm{n}+1} + \mathrm{const} \quad (\mathrm{at\ n} \neq \text{-}1).$$

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

Then we obtain the series for computation of the vector of the partial solution of the heterogeneous system of differential equations in the small segment $(x_j - x_i)$:

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

If the segment $(x_j - x_i)$ is not a small one it is possible to divide it by sub-segments and then it would be possible to propose the following recurrent (iteration) formulas for particular vector computing:

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

We have the formula for an individual sub-segment as well:

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

We can write:

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

Let's substitute $\boldsymbol{Y}(x_1)$ in $\boldsymbol{Y}(x_2)$ and obtain the following:

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

Let's compare the obtained expression with the formula:

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

and we will obtain evidently that:

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

so, we obtain the formula for a particular vector:

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

Hence, the vectors of the sub-segments $\boldsymbol{Y}^*(x_1 \leftarrow x_0), \boldsymbol{Y}^*(x_2 \leftarrow x_1)$ are not added to each other simply but they are added with involvement of the Cauchy matrix of the sub-segment.

Similarly we can write $\boldsymbol{Y}(x_3) = K(x_3 \leftarrow x_2)\boldsymbol{Y}(x_2) + \boldsymbol{Y}^*(x_3 \leftarrow x_2)$ and having substituted the formula for $\boldsymbol{Y}(x_2)$ in this expression we will obtain:

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

having compared this obtained expression with the formula:

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

we have obtained evidently that:

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

and at the same time we have obtained a formula for a particular vector:

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

Hence, a particular vector – a vector of the partial solution of the heterogeneous system of differential equations is computed by this way exactly, for example, the particular vector $\boldsymbol{Y}^*(x_3 \leftarrow x_0)$ is computed similarly in the considered segment $(x_3 \leftarrow x_0)$ through the computed particular vectors $\boldsymbol{Y}^*(x_1 \leftarrow x_0)$, $\boldsymbol{Y}^*(x_2 \leftarrow x_1)$, and $\boldsymbol{Y}^*(x_3 \leftarrow x_2)$ of the corresponding subs-segments $(x_1 \leftarrow x_0)$, $(x_2 \leftarrow x_1)$ and $(x_3 \leftarrow x_2)$.

Computation accuracy control can be carried out by the following way.
For a homogeneous system of differential equations we have:

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

We may write the following:

$$\boldsymbol{Y}(x_j) = K(x_j \leftarrow x_i)\boldsymbol{Y}(x_i) \text{ and}$$
$$\boldsymbol{Y}(x_i) = K(x_i \leftarrow x_j)\boldsymbol{Y}(x_j)$$

Having substituted one formula in another one we obtain:

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

i.e. we obtain:

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

however, the latter is possible only in case when

$$K(x_j \leftarrow x_i)K(x_i \leftarrow x_j) = E \text{ - unit matrix,}$$

i.e. the matrixes $K(x_j \leftarrow x_i)$ and $K(x_i \leftarrow x_j)$ are inverse ones.
In other words it has been proved that

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

i.e.

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

Hence, computing accuracy can be controlled by means of determination of the accuracy of matrix exponent computation (in other words the Cauchy matrixes or matrizants), that can be checked by having ascertained that the reciprocal inversion of the corresponding matrix exponents is observed in the integration segments:

$$K(x_j \leftarrow x_i)K(x_i \leftarrow x_j) = E.$$

3. Computational experiments.

Computational experiments have been carried out in comparison with the method of the boundary conditions transfer of Alexey Vinogradov. Line-by-line orthonormalization is used in this method.

Without use of orthonormalization it is possible by means of the boundary conditions transfer method to solve a problem of cylindrical shell loading that is fixed in cantilever fashion at the right boundary and loaded at the left boundary by the force distributed uniformly by the circular arch with the ratio of the length to the radius L/R=2 and with ratio of the radius to the thickness R/h=100. In case of ratio R/h=200 the problem by means of the method of the boundary conditions transfer without orthonormalozation cannot be solved by this time because there are mistakes resulted in due to counting instability. However, in case of use of orthonormalization in the method of the boundary conditions transfer the problems related to the parameters R/h=300, R/h=500, R/h=1000 can be solved.

A new method proposed in this paper allows solving of all above mentioned test problems without use of orthonormalization operation that results in significant simplification of its programming.

In case of the test computations of the problems characterized by the above mentioned parameters by means of this new proposed method the integration interval is divided into 10 segments while between the nodes as aforesaid the solution was found as a solution of the Cauchy problem. 50 harmonics of the Fourier series were used for solving the problem since the result in case of usage of 50 harmonics didn't differ from the case when 100 harmonics were used.

Test problem computing speed by means of the proposed method is not less compared to the boundary conditions transfer method because both methods when used for test problems while using 50 harmonics of the Fourier series produced a final solution instantaneously after launching a program (notebook ASUS M51V CPU Duo T5800). At the same time programming of this newly proposed method is significantly simpler because there is no need in orthonormalization procedure programming.

List of references

1. Gantmaher F.R. Matrix theory. M.: Nauka, 1988. 548 p.

Приложение 7. C++ program

```
// sopryazhenie.cpp: main file of the project.
//Solution of the boundary value problem – a cylindrical shell problem.
//Integration interval is divided into 10 matching segments: left boundary – point 0 and right boundary – point 10.
//WITHOUT ORTHONORMALIZATION
#include "stdafx.h"
#include <iostream>
#include <conio.h>

using namespace std;

//Multiplication of A matrix by b vector and obtain 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];
		}
	}
}

//Computation of the matrix exponent EXP=exp(A*delta_x)
void exponent(double A[8][8], double delta_x, double EXP[8][8]) {

	//n – number of the terms of the series in the exponent, m – a counter of the number of the terms of the series (m<=n)
	int n=100, m;
	double E[8][8]={0}, TMP1[8][8], TMP2[8][8];
	int i,j,k;

//E – unit matrix – the first term of the series of the exponent
	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;
//initial filling-in of the auxiliary array TMP1 – the previous term of the series for follow-up multiplication
	//and initial filling-in of the exponent by the first term of the series
	for(i=0;i<8;i++) {
```

```
                for(j=0;j<8;j++) {
                        TMP1[i][j]=E[i][j];
                        EXP[i][j]=E[i][j];
                }
        }

        //series of EXP exponent computation starting from the 2nd term of the series (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;//taken out beyond the cycle of multiplication of the row by the column
                                TMP2[i][j]/=(m-1);// taken out beyond the cycle of multiplication of the row by the column
                                EXP[i][j]+=TMP2[i][j];
                        }
                }
                //filling-in of the auxiliary array TMP1 for computing the next term of the series TMP2 in the next step of the cycle by m
        if (m<n) {
                        for(i=0;i<8;i++) {
                                for(j=0;j<8;j++) {
                                        TMP1[i][j]=TMP2[i][j];
                                }
                        }
                }
        }


        }
        //computation of the matrix MAT_ROW in the form of the matrix series for follow-up use
```

```
//when computing a vector - partial vector – a vector of the partial solution of the heterogeneous system of the ordinary differential equations at the step delta x
void mat_row_for_partial_vector(double A[8][8], double delta_x, double MAT_ROW[8][8]) {

//n – number of the terms of the series in MAT_ROW, m – a counter of the number of the terms of the series (m<=n)
int n=100, m;
double E[8][8]={0}, TMP1[8][8], TMP2[8][8];
int i,j,k;

//E – unit matrix – the first term of the series 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;

//initial filling-in of the auxiliary array TMP1 – the previous term of the series for following multiplication
//and initial filling-in of MAT_ROW by the first term of the series
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];
}
}

//a series of computation of MAT_ROW starting from the second term of the series (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];
}
```

```
}

//filling-in of the auxiliary array TMP1 for computing the next term of the series – TMP2 in the next step of the cycle by m
                if (m<n) {
                        for(i=0;i<8;i++) {
                                for(j=0;j<8;j++) {
                                        TMP1[i][j]=TMP2[i][j];
                                }
                        }
                }
        }
        }
//specifying the external influence vector in the system of ordinary differential equations – POWER vector: 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;
        }
//computation of the vector – ZERO (particular case) vector of the partial solution
//heterogeneous system of differential equations in the segment of interest:
        void partial_vector(double vector[8]){
                for(int i=0;i<8;i++){
                        vector[i]=0.0;
                }
        }

//computation of the vector – the vector of the partial solution of the heterogeneous system of differential equations in the segment of interest 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};//External influence vector on the shell
    double REZ[8]={0};
    double REZ_2[8]={0};
    power_vector_for_partial_vector(x_, POWER_);//Computing POWER_ at coordinate x_
    mat_on_vect(mat_row, POWER_, REZ);//Multiplication of the matrix mat_row by POWER vector and obtain REZ vector
    mat_on_vect(expo_, REZ, REZ_2);// Multiplication of matrix expo_ by vector REZ and obtain vector REZ_2
    for(int i=0;i<8;i++){
        vector[i]=REZ_2[i]*delta_x;
    }
}

//Solution of SLAE of 88 dimensionality by the Gauss method with discrimination of the basic element
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];//we will operate with the vector of the и right parts to provide that initial vector bb would not change when exiting the subprogram
        for(int j=0;j<(8*11);j++){
            A[i][j]=AA[i][j];//we will operate with A matrix to provide that initial AA matrix would not change when exiting the subprogram
        }
    }

int e;//number of the row where main (maximal) coefficient in the column jj is found
    double s, t, main;//Ancillary quantity

    for(int jj=0;jj<((8*11)-1);jj++){//Cycle by columns jj of transformation of A matrix into upper-triangle one
```

```
e=-1; s=0.0; main=A[jj][jj];
            for(int i=jj;i<(8*11);i++){//there is a number of y row where main (maximal) element is placed in the column jj and row interchanging is made
                  if ((A[i][jj]*A[i][jj])>s) {//Instead of multiplication (potential minus sign is deleted) it could be possible to use a function by abs() module
                        e=i; s=A[i][jj]*A[i][jj];
      }
            }

            if (e<0) {
            cout<<"Mistake "<<jj<<"\n"; return 0;
            }
            if (e>jj) {//If the main element isn't placed in the row with jj number but is placed in the row with y number
                  main=A[e][jj];
                  for(int j=0;j<(8*11);j++){//interchanging of two rows with e and jj numbers
                        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++){//reduction to the upper-triangle matrix
                  for(int j=(jj+1);j<(8*11);j++){
                        A[i][j]=A[i][j]-(1/main)*A[jj][j]*A[i][jj];//re-calculation of the coefficients of the row i>(jj+1)
                  }
                  b[i]=b[i]-(1/main)*b[jj]*A[i][jj];
                  A[i][jj]=0.0;//nullified elements of the row under diagonal element of A matrix
            }

      }//Cycle by jj columns of transformation of A matrix into upper-triangle one
      x[(8*11)-1]=b[(8*11)-1]/A[(8*11)-1][(8*11)-1];//initial determination of the last element of the desired solution x (87th)
      for(int i=((8*11)-2);i>=0;i--){//Computation of the elements of the solution x[i] from 86th to 0th
```

```
                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;//Number of the harmonic starting from the 1st (without zero one)
        int nn_last=50;//Number of the last harmonic
        double Moment[100+1]={0};//An array of the physical parameter (momentum) that is computed in each point between the boundaries

double step=0.05; //step=(L/R)/100 – step size of shell computation – a step of integration interval (it should be over zero, i.e. be positive)
double h_div_R;//Value of h/R
        h_div_R=1.0/100;
        double c2;
        c2=h_div_R*h_div_R/12;//Value of h*h/R/R/12
        double nju;
        nju=0.3;
        double gamma;
        gamma=3.14159265359/4;//The force distribution angle by the left boundary

//printing to files:
        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++){ //A CYCLE BY HARMONICS STARTING FROM THE 1st HARMONIC (EXCEPT ZERO ONE)

double x=0.0;//A coordinate from the left boundary – it is needed in case of heterogeneous system of the ODE for computing the particular vector FF

double expo_from_minus_step[8][8]={0};//The matrix for placement of the exponent in it at the step of (0-x1) type

double expo_from_plus_step[8][8]={0};// The matrix for placement of the exponent in it in the step of (x1-0) type

double mat_row_for_minus_expo[8][8]={0};//the auxiliary matrix for particular vector computing when moving at step of (0-x1) type

double mat_row_for_plus_expo[8][8]={0};// the auxiliary matrix for particular vector computing when moving at step of (x1-0) type

double U[4][8]={0};//The matrix of the boundary conditions of the left boundary of the dimensionality 4x8

double u_[4]={0};//Dimensionality 4 vector of the external influence for the boundary conditions of the left boundary

double V[4][8]={0};//The boundary conditions matrix of the right boundary of the dimensionality 4x8

double v_[4]={0};// The dimensionality 4 vector of the external influence for the boundary conditions for the right boundary

double Y[100+1][8]={0};//The array of the vector-solutions of the corresponding linear algebraic equations system (in each point of the interval between the boundaries): MATRIXS*Y=VECTORS

double A[8][8]={0};//Matrix of the coefficients of the system of ODE

double FF[8]={0};//Vector of the particular solution of the heterogeneous ODE at the integration interval sector

double Y_many[8*11]={0};// a composite vector consisting of the vectors Y(xi) in 11 points from point 0 (left boundary Y(0) to the point 10 (right boundary Y(x10))

double MATRIX_many[8*11][8*11]={0};//The matrix of the system of the linear algebraic equations

double B_many[8*11]={0};// a vector of the right parts of the SLAE: MATRIX_many*Y_many=B_many

double Y_vspom[8]={0};//an auxiliary vector

double Y_rezult[8]={0};//an auxiliary vector

```
double nn2,nn3,nn4,nn5,nn6,nn7,nn8;//Number of the nn harmonic raised to corresponding powers
        nn2=nn*nn; nn3=nn2*nn; nn4=nn2*nn2; nn5=nn4*nn; nn6=nn4*nn2; nn7=nn6*nn; nn8=nn4*nn4;

    //Filling-in of non-zero elements of the A matrix of the coefficients of ODE system
        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;
    //Here firstly it is necessary to make filling-in by non-zero values of the matrix and boundary conditions vector U*Y[0]=u_ (on the left) and V*Y[100]=v_ (on the right):
        U[0][1]=1.0; U[0][2]=nn*nju; U[0][4]=nju; u_[0]=0.0;//Force T1 at the left boundary is equal to zero
    U[1][0]=-(1-nju)/2*nn; U[1][3]=(1-nju)/2; U[1][5]=(1-nju)*nn*c2; u_[1]=0.0;//Force S* at the left boundary is equal to zero
        U[2][4]=-nju*nn2; U[2][6]=1.0; u_[2]=0;//Momentum M1 at the left boundary is equal to zero
        U[3][5]=(2-nju)*nn2; U[3][7]=-1.0;
        u_[3]=-sin(nn*gamma)/(nn*gamma);//Force Q1* at the left boundary is distributed at the angle -gamma +gamma

    V[0][0]=1.0; v_[0]=0.0;// The right boundary displacement u is equal to zero
        V[1][2]=1.0; v_[1]=0.0;//The right boundary displacement v is equal to zero
        V[2][4]=1.0; v_[2]=0.0;//The right boundary displacement w is equal to zero
        V[3][5]=1.0; v_[3]=0.0;//The right boundary rotation angle is equal to zero
        //Here initial filling-in of U*Y[0]=u_ и V*Y[100]=v_ terminates
```

```
exponent(A,(-step*10),expo_from_minus_step);//A negative step (step value is less than zero due to direction of matrix exponent computation)
        //x=0.0;//the initial value of coordinate – for partial vector computation
        //mat_row_for_partial_vector(A, step, mat_row_for_minus_expo);

        //Filling-in of the SLAE coefficients matrix 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) unit matrix and EXPO matrix should be written into MATRIX_many
            for(int i=0;i<8;i++){
                MATRIX_many[i+4+kk*8][i+kk*8]=1.0;//filling-in by unit matrixes
                for(int j=0;j<8;j++){
                    MATRIX_many[i+4+kk*8][j+8+kk*8]=-expo_from_minus_step[i][j];//filling-in by matrix exponents
                }
            }
    }

        //Solution of the system of linear algebraic equations
        GAUSS(MATRIX_many,B_many,Y_many);

        //Computation of the state vectors in 101 points – left point 0 and right point 100
        exponent(A,step,expo_from_plus_step);

        for(int i=0;i<11;i++){//passing points filling-in in all 10 intervals (we will obtain points from 0 to 100) between 11 nodes
            for(int j=0;j<8;j++){
```

```
                    Y[0+i*10][j]=Y_many[j+i*8];//in 11 nodes the vectors are taken from SLAE solutions – from Y many
                }
            }
        for(int i=0;i<10;i++){//passing points filling-in in 10 intervals
                for(int j=0;j<8;j++){
                    Y_vspom[j]=Y[0+i*10][j];//the initial vector for i^th segment, zero point, starting point of the i^th segment
                }

                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];//the first point of the interval filling-in
                    Y_vspom[j]=Y_rezult[j];//for the next step
                }

        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];//filling-in of the 2nd point of the interval
                    Y_vspom[j]=Y_rezult[j];//for the next step
                }

                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];//filling-in of the 3rd point of the interval
                    Y_vspom[j]=Y_rezult[j];//for the next step
                }

                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];//filing-in of the 4th point of the interval
                    Y_vspom[j]=Y_rezult[j];//for the next step
                }

                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];// filing-in of the 5th point
of the interval
                        Y_vspom[j]=Y_rezult[j];// for the next step
                }

                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];// filing-in of the 6th point
of the interval
                        Y_vspom[j]=Y_rezult[j];// for the next step
                }

                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];// filing-in of the 7th point
of the interval
                        Y_vspom[j]=Y_rezult[j];// for the next step
                }

        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];// filing-in of the 8th point
of the interval
                        Y_vspom[j]=Y_rezult[j];// for the next step
                }

                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];// filing-in of the 9th point
of the interval
                        Y_vspom[j]=Y_rezult[j];// for the next step
                }

            }



        //Computation of the momentum in all points between the boundaries
```

```
		for(int ii=0;ii<=100;ii++){
			Moment[ii]+=Y[ii][4]*(-
nju*nn2)+Y[ii][6]*1.0;//Momentum M1 in the point [ii]
			//U[2][4]=-nju*nn2;   U[2][6]=1.0;   u_[2]=0;//Momentum
M1
		}



		}//CYCLE BY HARMONICS TERMINATES HERE




		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;

	}
```

My web-site related to stiff boundary value problems solution methods:
www.vinogradov-design.narod.ru/math.html

Научное издание

Виноградов Алексей Юрьевич

Методы решения жестких и нежестких краевых задач

Монография

Публикуется в авторской редакции с готового оригинал-макета

Рассылка обязательных экземпляров осуществлена
в соответствии с действующими нормативными актами

Подписано в печать 22.07 2016 г. Формат 60x84/16.
Бумага офсетная. Гарнитура Georgia. Усл. печ. л. 7,4.
Уч.-изд. л. 8,0. Тираж 1000. Заказ 129. «С» 77.

Отпечатано в издательстве Волгоградского государственного университета.
400062 г. Волгоград, просп. Университетский, 100.
E-mail: izvolgu@volsu.ru