Оглавление

Введение

В течении многих лет изучается распространение сверхвысокочастотных (СВЧ) спиновых волн в магнитных пространственно-периодических волноведущих структурах [1]. Такие структуры называются магнонными кристаллами [2][3][4]. Для магнонных кристаллов характерны изменение толщины или ширины волновода, металлизация его поверхности, ионная имплантация для создания периодичности по одному из параметров волноведущей структуры. На основе магнонных кристаллов возможно создание СВЧ генераторов [5], шумоподавителей [6], ограничителей мощности [6], фильтров [7]. Актуальным направлением также является исследование феррит-сегнетоэлектрических структур [8]. Преимуществом данного подхода является отсутствие электромагнитов для перестройки системы в частотном диапазоне: система перестраивается путём изменения внешнего электрического поля, что позволяет миниатюризировать устройства относительно их сегнетоэлектрических аналогов[9]. Другой важной особенностью данного типа устройств является их способность накапливать значительный сдвиг фазы на миллиметровом пробеге волны из-за низкой групповой и фазовой скорости волн в феррите. [10]

Целью данной работы является исследование и численное моделирование методом конечных элементов периодических структур феррит-сегнетоэлектрик при прохождении через них электромагнитной СВЧ волны при определённых граничных условиях, построение дисперсионных диаграм и их анализ.

1 Теория метода конечных элементов

Суть метода следует из его названия. Область, в которой ищется решение дифференциальных уравнений, разбивается на конечное количество подобластей одинм из генераторов сетки. В каждой подобласти (в данной работе были выбраны треугольные элементы) вводится аппроксимирующая гладкая функция, равная в данном узле единице, во всех соседних - нулю и равномерно убывающую от данного узла к соседним, и произвольно выбирается ее вид: в простейшем случае это полином первой степени. Значения функций в узлах решётки являются решением задачи и заранее неизвестны. Коэффициенты аппроксимирующих функций выражаются через значения функций в узлах элементов. Составляется система линейных алгебраических уравнений. Количество уравнений равно количеству неизвестных значений в узлах, на которых ищется решение исходной системы, прямо пропорционально количеству элементов и ограничивается только возможностями ЭВМ. Так как каждый из элементов связан с ограниченным количеством соседних, система линейных алгебраических уравнений имеет разрежённый вид, что существенно упрощает её решение. Для работы с подобным типом данных созданы специальные библиотеки (mtl4, SparseLib++, SPARSPAK и другие)

Если говорить в матричных терминах, то собираются так называемые матрицы жёсткости (или матрица Дирихле) и масс. Далее на эти матрицы накладываются граничные условия (например, при условиях Неймана в матрицах не меняется ничего, а при условиях Дирихле из матриц вычёркиваются строки и столбцы, соответствующие граничным узлам, так как в силу краевых условий значение соответствующих компонент решения известно). Затем собирается система линейных уравнений и решается одним из известных методов.

В Приложении В приведён код на bash/c++, который, исходя из предгенерированной трехмерной сетки, расчитывает матрицы жесткости и масс; а по ним - волновые вектора волн, распространяющихся в данной структуре. Там же изложена структура сгенерированной Comsol'ом сетки. Стоит упомянуть, что код, изложенный в Приложении В, не проверялся на корректность работы; то есть нет никаких оснований доверять числам на выходе. Однако, подобный код, прошедший проверку и написанный на Fortran, есть в книге [11].

Ниже будет подробно описан метод расчета матрицы жесткости, исходя из предгенерированной сетки. Матрица масс собирается почти аналогично.

Элементы матрицы жёсткости A^k в общем случае равны

$A_{ij} = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j \, dx.$

Пусть дано уравнение Пуассона $-\nabla^2 u = f$ в пространстве Ω и при граничных условниях $u_{bound} = 0$.

Представим функцию как ряд:

$u \approx u^h = u_1\varphi_1 + \cdots + u_n\varphi_n.$

Если u_i — это известные значения функции в узлах, а φ — некие базисные функции, то

$A_{ij}^{[k]} = \int_{triangle} \nabla\varphi_i \cdot \nabla\varphi_j \, dx.$

Теперь соберем матрицу для одного треугольника. Пусть дан один конечный элемент, для простоты — треугольный. Матрица жёсткости, по сути, задаёт связи между узлами. Так как у элемента три узла (в локальной нумерации — 0, 1 и 2), то матрица будет иметь вид

$$\begin{bmatrix} S_{00} & S_{01} & S_{02} \\ S_{10} & S_{11} & S_{12} \\ S_{20} & S_{21} & S_{22} \end{bmatrix}$$

В дальнейшем матрицу для одного треугольника будем называть локальной, для всей сетки сразу - глобальной.

В общем случае, элементы S_{ij} определеяются через линейные функции

$$\alpha_1 = \frac{1}{4A}\big((x_1y_2 + x_2y_1) \ + \ (y_1 - y_2)x \ + \ (x_2 - x_1)y\big).$$

где A — площадь треугольного элемента.

α_2 и α_3 получаются из α_1 цикличной перестановкой индексов. Удобно искать A как определитель матрицы

$$A = \det \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}$$

Сами $S_{ij} = \int (\nabla\alpha_i)(\nabla\alpha_j)dS \quad i, j = 0, 1, 2$

В описываемом случае для каждого треугольника составляется такая матрица:

$$\begin{bmatrix} S_{00} = 0 & S_{01} & S_{02} \\ S_{10} & S_{11} = 0 & S_{12} \\ S_{20} & S_{21} & S_{22} = 0 \end{bmatrix}$$

$$S_{01} = \frac{(y_1 - y_2)(y_2 - y_0) + (x_2 - x_1)(x_0 - x_2)}{4A}$$

$$S_{02} = \frac{(y_2 - y_1)(y_1 - y_0) + (x_1 - x_2)(x_0 - x_1)}{4A}$$

$$S_{10} = \frac{(y_0 - y_2)(y_2 - y_1) + (x_2 - x_0)(x_1 - x_2)}{4A}$$

$$S_{12} = \frac{(y_2 - y_0)(y_0 - y_1) + (x_0 - x_2)(x_1 - x_0)}{4A}$$

$$S_{20} = \frac{(y_0 - y_1)(y_1 - y_2) + (x_1 - x_0)(x_2 - x_1)}{4A}$$

$$S_{21} = \frac{(y_1 - y_0)(y_0 - y_2) + (x_0 - x_1)(x_0 - x_2)}{4A}$$

После того, как была собрана матрица для одного треугольника, ее надо включить в глобальную матрицу. Это можно сделать двумя путями.

1.1 Способ объединения матриц

Для того, чтобы сделать из многих локальных матриц, полученых выше, одну большую матрицу, описывающую отношения между узлами всей области расчёта, необходимо произвести процедуру объединения матриц. Пусть символ d обозначает разделённые элементы (а), а символ c — объединённые элементы (б).

Обозначим

$$u_d^T = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{bmatrix}$$

— вектор-строку значений функции в вершинах двух треугольников. Символ u^T обозначает транспонирование матрицы u. То есть это вектор значений функции в шести узлах треугольников. Очевидно, что при объединении оных получится вектор u_d, содержащий только четыре компоненты.

Преобразование происходит по схеме

$$u_d = Cu_c \ \Leftrightarrow \ \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & & 1 & & \\ & 1 & & & & \\ & & & & & 1 \\ 1 & & & & & \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

Нумерация, конечно же, произвольная: необходимо равенство функции в соответствующих вершинах. Матрицу C называют матрицей преобразования, а само уравнение называют связанной системой.

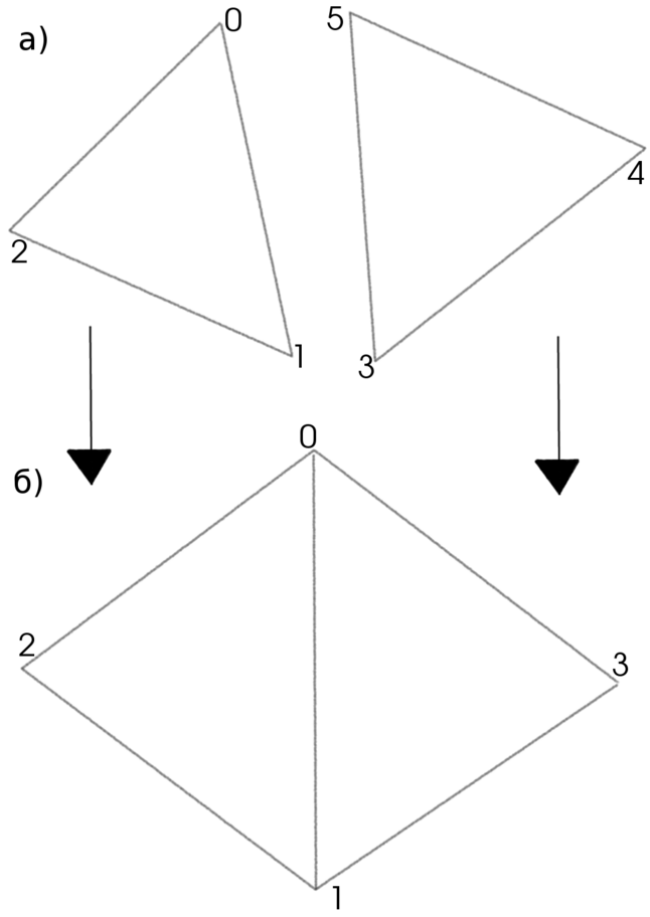

Рисунок 1 — Сшивание треугольных элементов.

Запишем теперь матрицу жёсткости для двух треугольников:

$$S_d = \begin{bmatrix} S^{(1)} & 0 \\ 0 & S^{(2)} \end{bmatrix} \Leftrightarrow \begin{bmatrix} S_{00} & S_{01} & S_{02} & & & \\ S_{10} & S_{11} & S_{12} & & & \\ S_{20} & S_{21} & S_{22} & & & \\ & & & S_{33} & S_{34} & S_{35} \\ & & & S_{43} & S_{44} & S_{45} \\ & & & S_{53} & S_{54} & S_{55} \end{bmatrix}$$

Результирующая матрица

$$S_{global} = C^T S_d C = \begin{bmatrix} S_{00}^{(1)} + S_{55}^{(2)} & S_{01}^{(1)} + S_{53}^{(2)} & S_{02}^{(1)} & S_{54}^{(2)} \\ S_{10}^{(1)} + S_{35}^{(2)} & S_{11}^{(1)} + S_{33}^{(2)} & S_{12}^{(1)} & S_{34}^{(2)} \\ S_{20}^{(1)} & S_{20}^{(1)} & S_{22}^{(1)} & 0 \\ S_{45}^{(2)} & S_{43}^{(2)} & 0 & S_{44}^{(2)} \end{bmatrix}$$

То есть на каждом следующем шаге необходимо добавлять новые элементы к уже существующим.

1.2 Способ дозаписи матриц

Пусть есть область, представленная и разбитая на треугольники так, как преставлено на рисунке. Пусть данная сетка содержит N узлов. Создадим глобальную матрицу $\mathfrak{S}$ (размера, очевидно, $N \times N$) и заполним её нулями. Начнём строиль локальные матрицы S для треугольников, например, для $\Delta 036$.

Введём локальную нумерацию для данного треугольника: пусть его верхняя вершина имеет локальный номер 0, далее по часовой стрелке 1 и 2. Иначе говоря, пусть глобальным номерам $0, 3, 6$ соответствуют локальные номера $0, 1, 2$ соответственно.

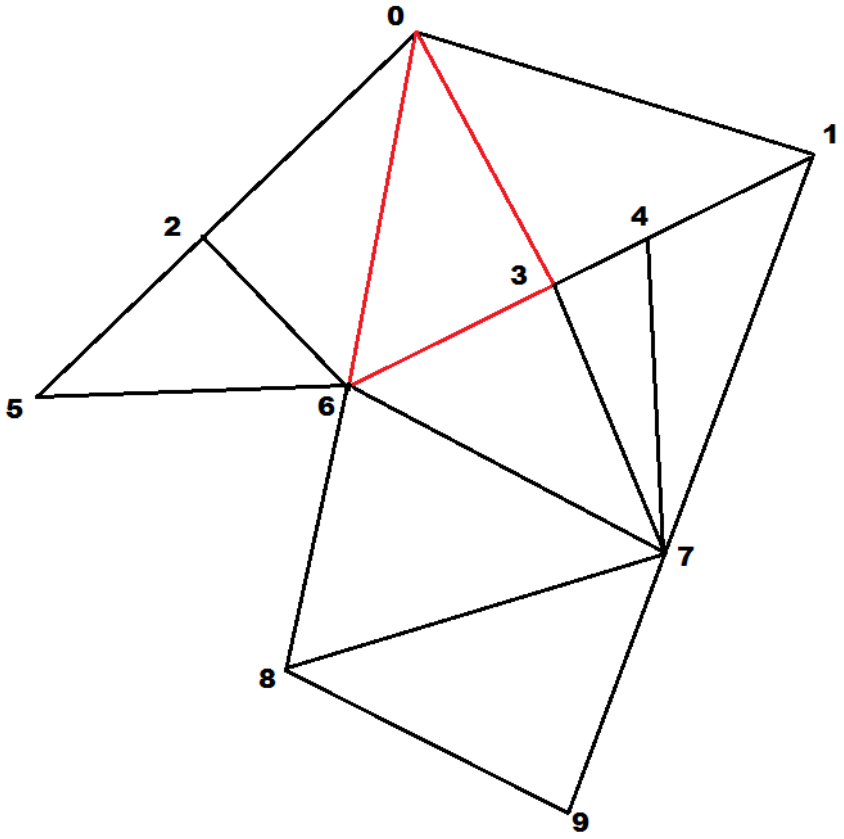

Рисунок 2 — Расчетная область, покрытая сеткой.

Составим матрицу для этого треугольника так, как описано выше, получив что-то типа

$$S = \begin{bmatrix} S_{00} & S_{01} & S_{02} \\ S_{10} & S_{11} & S_{12} \\ S_{20} & S_{21} & S_{22} \end{bmatrix}$$

Теперь заменим локальную нумерацию на глобальную. То есть запишем локальное число S_{00} как глобальное число $\mathfrak{S}_{00}$, S_{01} - как $\mathfrak{S}_{03}$, S_{02} - как $\mathfrak{S}_{06}$ и так далее.

Получим

$$\mathfrak{S} = \begin{bmatrix} S_{00} & 0 & 0 & S_{03} & 0 & 0 & S_{06} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ S_{10} & 0 & 0 & S_{13} & 0 & 0 & S_{16} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ S_{20} & 0 & 0 & S_{21} & 0 & 0 & S_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

С остальными треугольниками поступаем аналогично. Необходимо помнить, что надо именно дописать число в глобальную ячейку, то есть прибавить к уже существующему.

1.3 Учет граничных условий

В случае граничных условий первого рода необходимо изменить матрицу $\mathfrak{S}$. Граничное условие гласит, что функция в узлах на границе равна нулю. для узла $u_{i,j}$ необходимо вычеркнуть i-тый столбей и j-ую строку в матрице $\mathfrak{S}$, а так же вычеркнуть сам узел из массива узлов решётки.

В случае граничных условий второго рода глобальная матрица не меняется.

2 Исследуемая модель

Данная работа состоит из двух частей. В первой части рассматриваются двумерные фотонно-кристаллические волноведущие структуры. На их примере демонстрируется тип структуры, способной избирательно пропускать входящее излучение. Без потери общности можно говорить о двумерной задаче, так как переход к трехмерному случаю фотонного кристалла не представляет труда. Во второй части исследуются трёхмерные феррит-сегнетоэлектрические структуры, в которых распространяется магнитостатическая волна. Эти структуры наоборот: избирательно не пропускают входящее излучение, и для этой структуры важна трехмерность.

2.1 Двумерные фотонные кристаллы

Фотонный кристалл в рассматриваемой работе представляет собой периодическую решетку из стержней цилиндрической формы с заданной диэлектрической проницаемостью $\varepsilon = 9.85$. Радиус стержней r и частота бегущей волны выбраны так, чтобы частота бегущей волны попадала в зону частотного непропускания фотонного кристалла (запрещенную зону). На самом деле, в реальности варьируется не частота, а расстояние между стержнями (период фотонного кристалла). Очевидно, что изменение частоты волны физически эквивалентно изменению периода решётки, однако последнее много проще программируется, в счёт чего и был сделан выбор. Стоит напомнить, что частотное непропускание появляется из-за дифракции Брэгга на неоднородностях кристалла. Волны на данной частоте экспоненциально затухают по мере удаления от источника. Это позволяет, введя в кристалл неоднородность путем удаления нескольких стержней, получить направленную волну.

В итоге, если пустить в такой волноведущей структуре волну заданной частоты, то её энергия будет сфокусирована в пространстве дефекта и практически не будет излучаться в окружающее пространство.

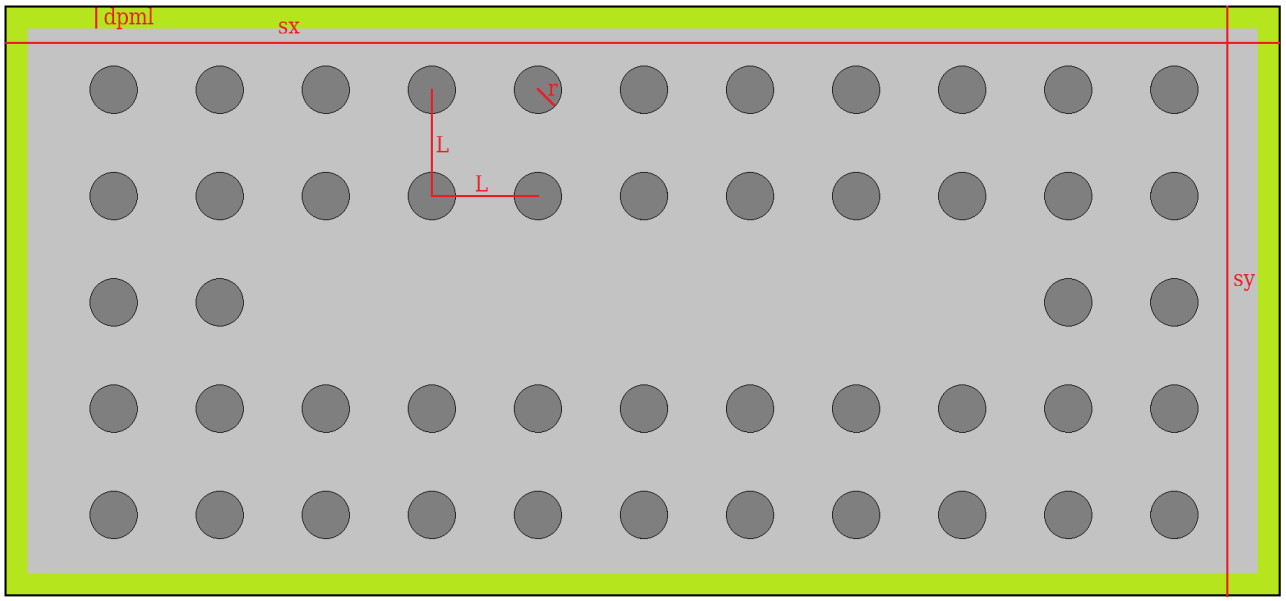

Рисунок 3 — Условная структура фотонного кристалла.

Моделирование этой системы проводилось в пакете meep на специальном внутреннем языке программы, наследуемом от lisp. Код и инструкция по исполнению находится в Приложении В.

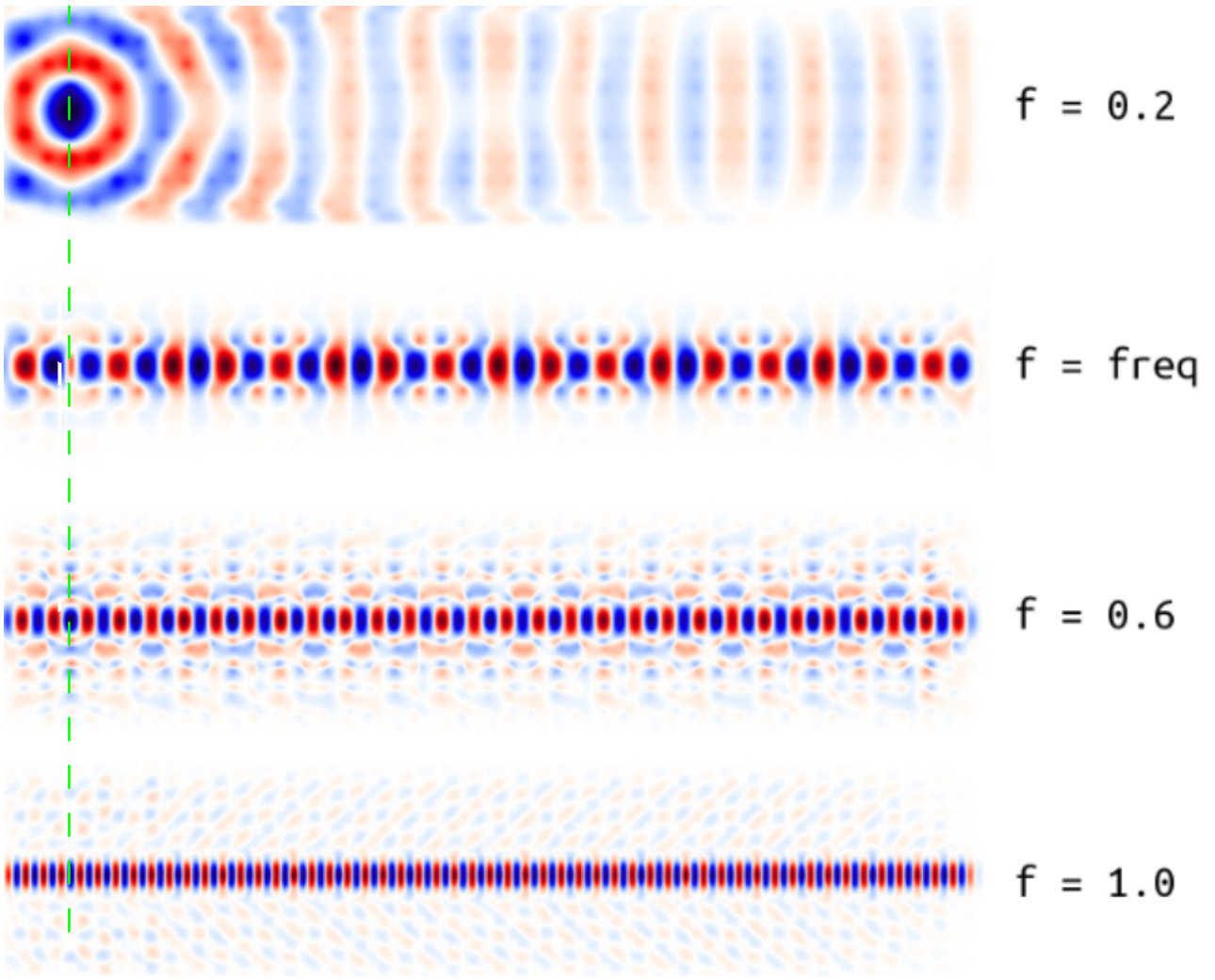

Рисунок 4 — Распределение амплитуды поля в волноводе. Точечный источник в каждой структуре находится слева в центре.

2.2 Трёхмерные феррит-сегнетоэлектрические структуры

Для исследования во втором разделе были выбраны три трёхмерных модели, представленные на рисунках ниже. Отличающиеся деталями, в целом они одинаковы: тонкая ферритовая плёнка (параметры в приложении), на которой расположен диэлектрик (параметры в приложении). Магнитное поле $\vec{H} = 108$ Э направлено перпендикулярно плоскости рисунка. Поля предполагаются однородными. На структуру наложены граничные условия: по границам, нормальным к электрическому полю - условия постоянства разности фаз (периодические условия Флоке); по остальным - металлические стенки. В рассматриваемой структуре сонаправлено полю $\vec{E}$ бежит магнитостатическая волна (МСВ). Далее подразумевается рабочий диапазон частот от 2.5 ГГц до 3.0 ГГц, если не указано иное. При таких частотах волна будет распределяться в феррите, но не в сегнетоэлектрике; иными словами, сегнетоэлектрик играет роль нагрузки, не имея собственных запрещённых зон в рассматриваемом диапазоне.

Суть данной работы сводится к исследованию запрещённых зон в феррит-сегнетоэлектрической структуре, а конкретно - управлением параметрами (шириной и положением) оных. Запрещённой зоной называется область частот, на которых бегущая волна в периодической структуре затухает из-за Брэгговского рассеивания. Эта волна имеет волновой вектор $k = \frac{\pi}{L}$, где L - период структуры. На дисперсионной диаграмме эта зона будет находиться в месте пересечения прямой и встречной волны одной моды. На ширину запрещённой зоны в данной структуре влияет проницаемость диэлектрика (чем

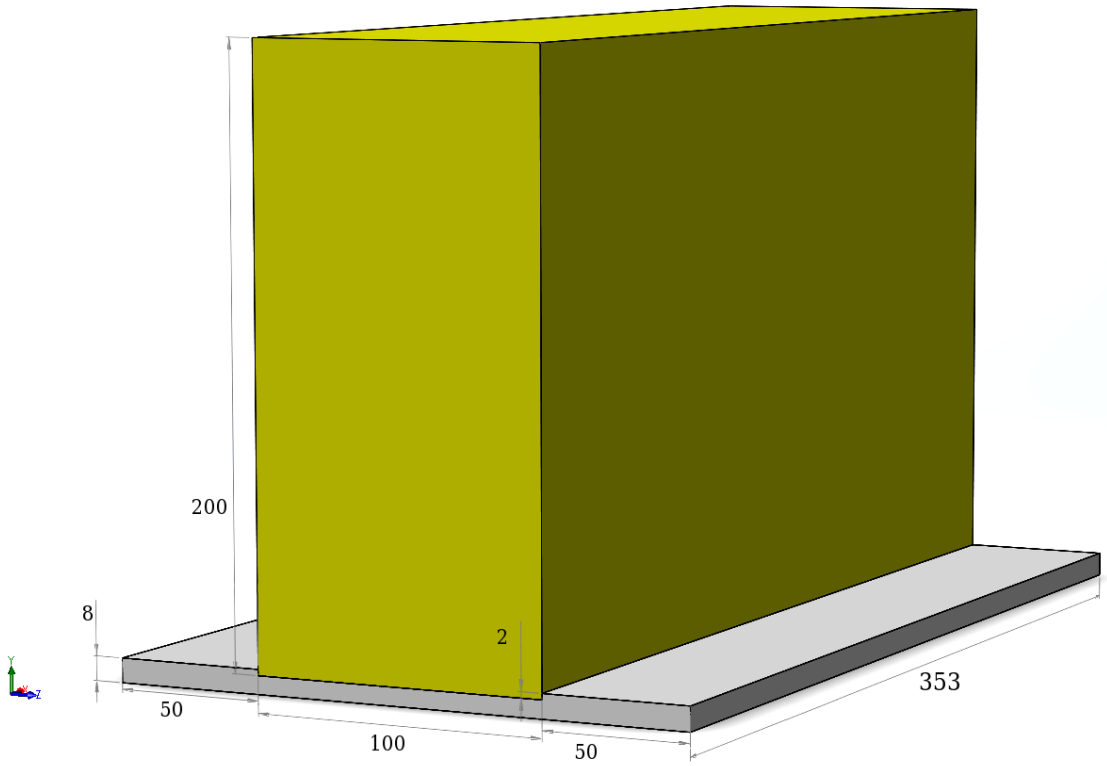

Рисунок 5 — Общий вид исследуемой структуры. Размеры указаны в микрометрах.

больше - тем уже) и напряжённость электрического поля (чем больше - тем уже).

Очевидно, что запрещённая зона появляется только в структуре с периодичностью. Это показывает первая исследуемая структура: она не имеет периодичности в феррите, из-за чего в ней и нет запрещённой зоны.

Если ввести в феррит периодичность путём (например) вырезания канавки определённой толщины, то проявится взаимодействие прямой и обратной волны, что приведёт к появлению запрещённой зоны, что демонстрирует четвёртая структура, на которой будет сосредоточено внимание всю оставшуюся часть текста. Вторая и третья структуры являют собой переходную форму к основной структуре.

На следующих четырёх страницах приводятся размеры и структура исследуемых моделей. Везде приводится сечение оной, подразумевая, что в направлении, перпендикулярном плоскости рисунка, структура имеет то же сечение, и длина её в этом направлении такова, как указано в данных в Приложении А, а именно - 353 мкм

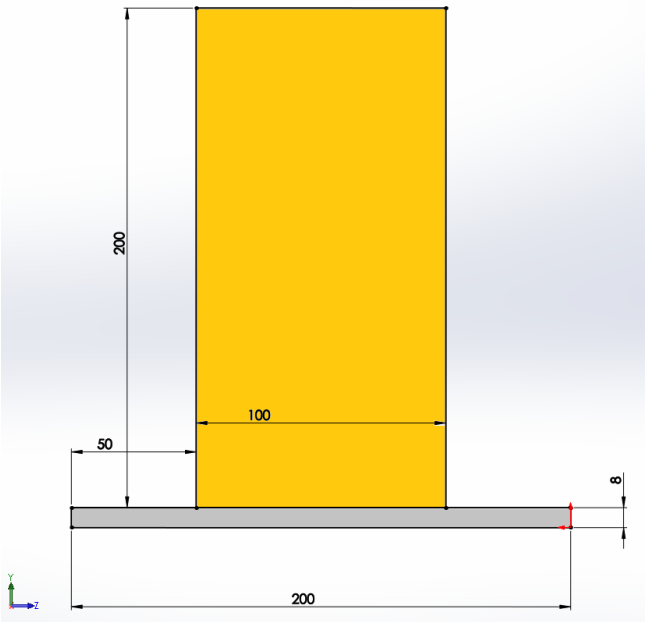

Рисунок 6 — Сечение первой (базовой) структуры (с сегнетоэлектрической периодичностью). Размеры указаны в микрометрах.

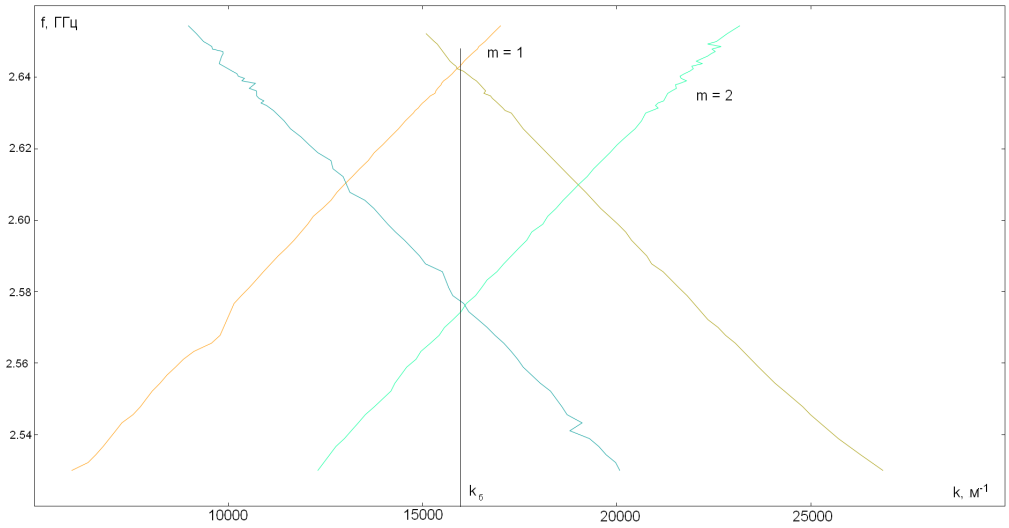

Рисунок 7 — Дисперсионная характеристика волны, распространяющейся в данной периодической структуре.

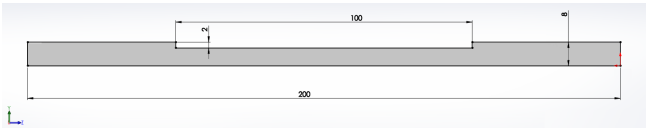

Рисунок 8 — Сечение второй структуры (без сегнетоэлектрика). Размеры указаны в микрометрах.

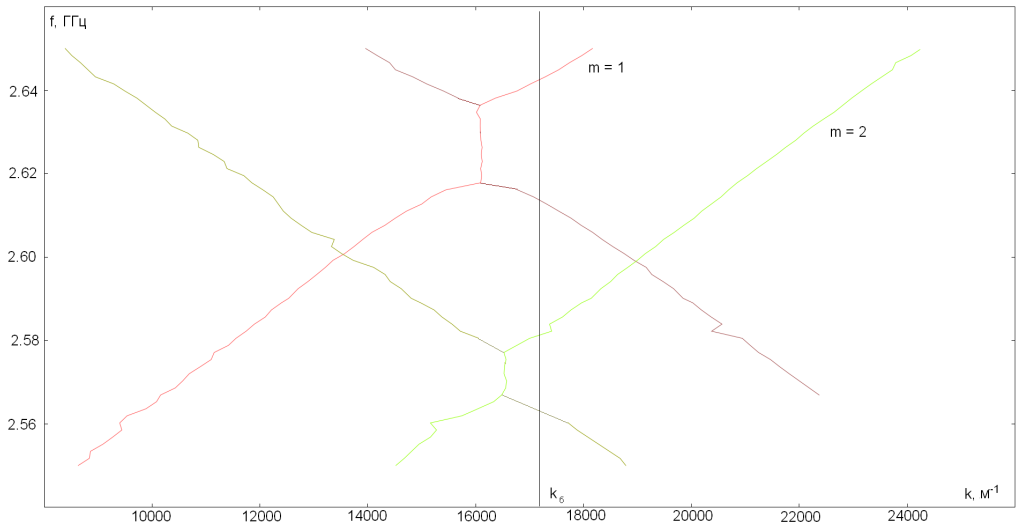

Рисунок 9 — Дисперсионная характеристика волны, распространяющейся в данной периодической структуре.

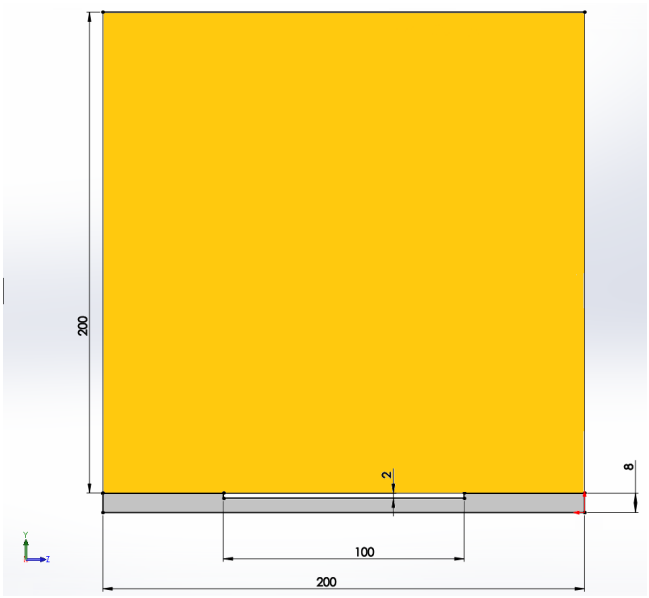

Рисунок 10 — Сечение третьей структуры (с ферритовой периодичностью). Размеры указаны в микрометрах.

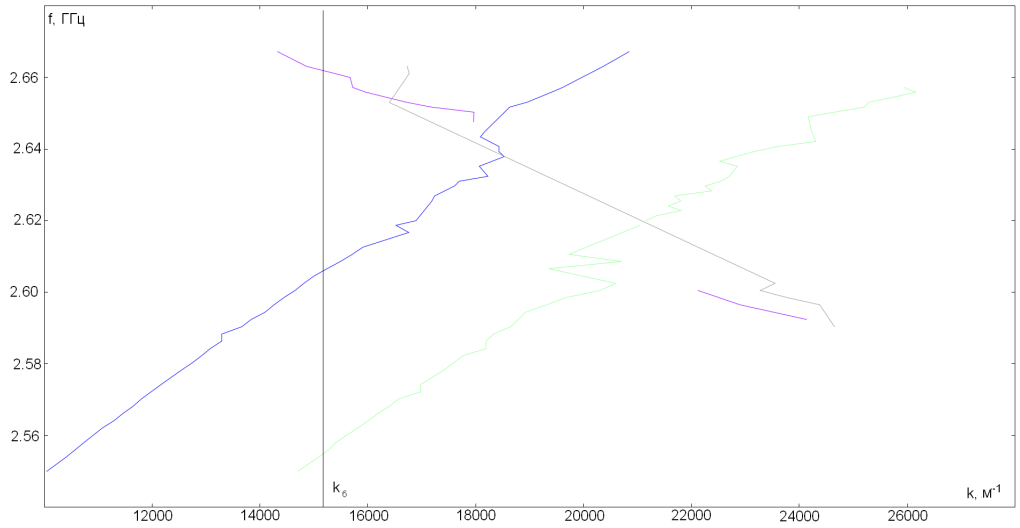

Рисунок 11 — Дисперсионная характеристика волны, распространяющейся в данной периодической структуре.

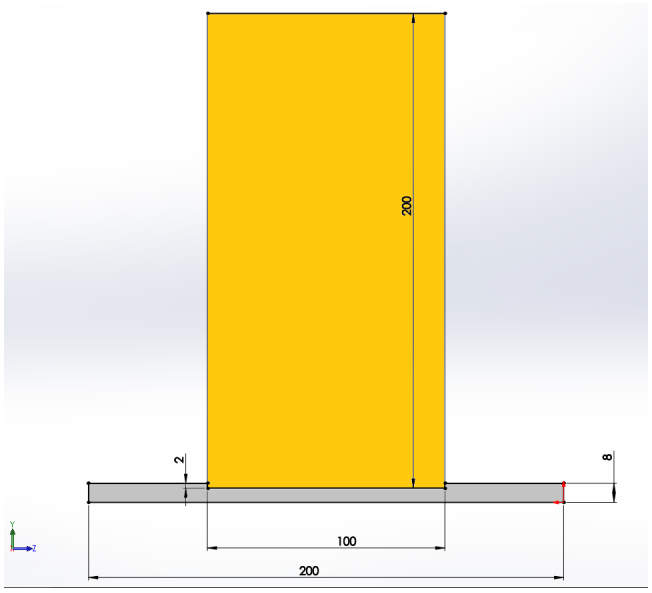

Рисунок 12 — Сечение четвёртой структуры (с феррит-сегнетоэлектрической периодичностью). Размеры указаны в микрометрах.

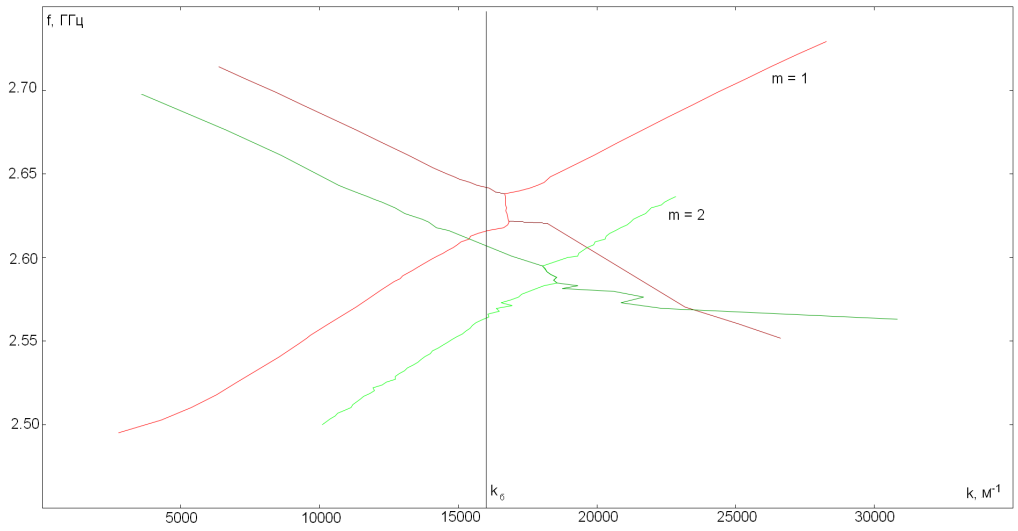

Рисунок 13 — Дисперсионная характеристика волны, распространяющейся в данной периодической структуре.

2.3 Невзаимность волн в рассматриваемой структуре

Стоит отметить, что запрещённая зона при добавлении сегнетоэлектрика смещается не только вниз на дисперсионной диаграмме, но и влево. Это связано с невзаимностью структуры. Как было сказано ранее, запрещённая зона - это место пересечения прямой и обратной волны. Невзаимность означает, что обратная волна пойдёт иначе, чем прямая; в данном случае ниже. Легко понять, что это приведёт к смещению зоны влево.

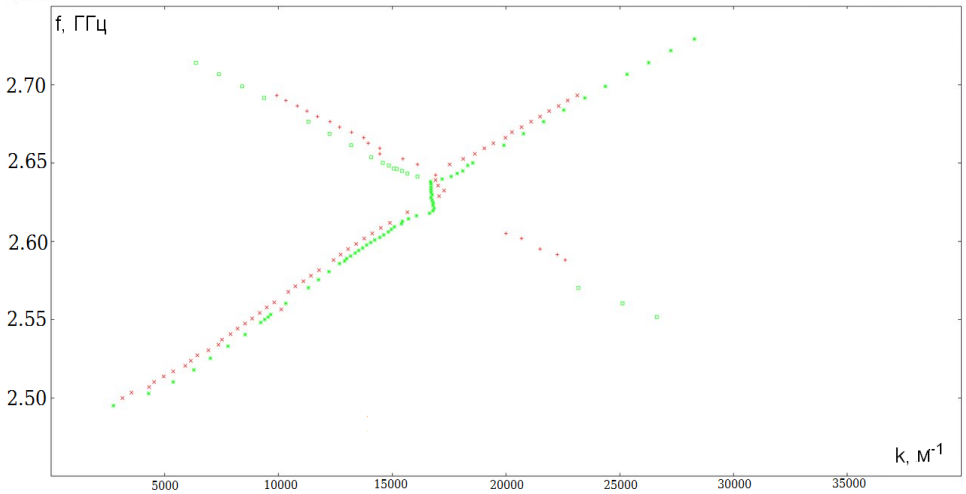

Рисунок 14 — Дисперсионные характеристики для четвёртой структуры для первой моды. Красным обозначена первая плюсовая гармоника, зелёным - первая минусовая.

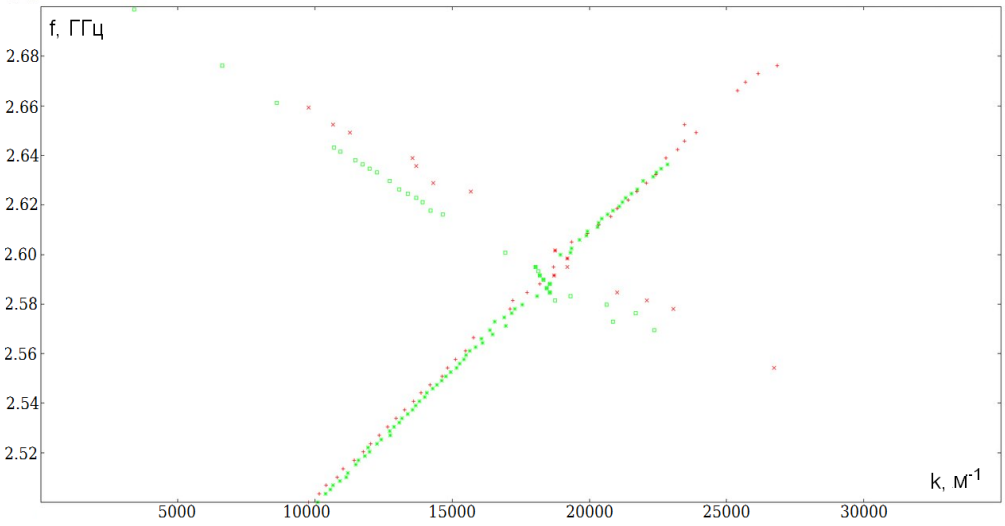

Рисунок 15 — Дисперсионные характеристики для четвёртой структуры для второй моды (б). Красным обозначена первая плюсовая гармоника, зелёным - первая минусовая.

2.4 Случай отсутствия запрещённой зоны для волны в структуре

Ранее было упомянуто, что на ширина запрещённой зоны обратно пропорционально зависит от проницаемости сегнетоэлектрика. Возможно ли подобрать этот параметры так, чтобы запрещённая зона исчезла?

В принципе, можно. На графике показаны несколько дисперсионных кривых для первой прямой моды при разных ε. Видно, что с увеличением проницаемости уменьшается ширина запрещённой зоны вплоть до её исчезновения при $\varepsilon = 50000$.

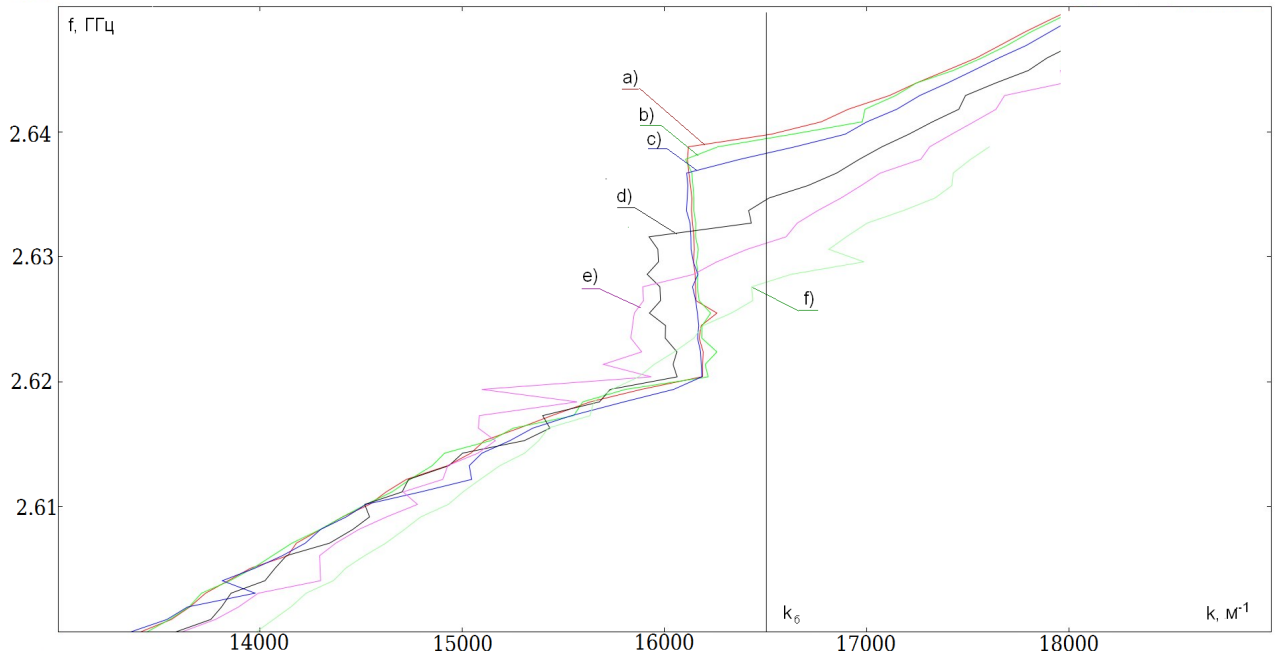

Рисунок 16 — Дисперсионные характеристики при различных проницаемостях сегнетоэлектрика. Проницаемости сегнетоэлектрика равны: a) 2000; b)4000; c)8000; d)20000; e)30000; f)50000;

К сожалению, такие значения проницаемостей трудно достижимы в реальном эксперименте, поэтому необходимо будет использовать и другие методы уменьшения ширины зоны для достижения этого эффекта.

Заключение

В настоящей работе проведено исследование процессов распространения электромагнитных волн в двумерных магнонных кристаллах и процессов распространения электромагнитных волн в трёхмерных периодических структурах феррит-сегнетоэлектрик в зависимости от геометрических размеров и материальных параметров слоя сегнетоэлектрика при прохождении через них электромагнитной СВЧ волны при определённых граничных условиях, построение дисперсионных диаграм и их анализ. В работе рассмотрены различные конфигурации расположения слоя сегнетоэлектрика относительно канавок на поверхности плёнки железо-иттриевого граната. С помощью метода конечных элементов, реализованного в программе COMSOL 5.0, проведён расчет дисперсионных характеристик рассматриваемой структуры. Показано, что путем нагрузки магнонного кристалла слоем сегнетоэлектрика удается управлять частотными границами запрещенной зоны.

Список использованных источников

[1] Hillebrands B Serga A.A., Chumak A. V. *J. Phys. D: Appl.Phys.*, 43(26):264, 2010.

[2] Tsai C.S Nikitov S.A., Tailhades Ph. *J. Magnet. Magn. Mater.*, 236:320–330, 2001.

[3] Устинов А.Б. Ковшиков Н.Г. Калиникос Б.А. Дроздовский А.В., Черкасский М.А. *Письма в ЖЭТФ*, 91:17–22, 2010.

[4] Дулин Ю.В. Никитов С.А. Гришин С. В., Бегинин Е.Н. *Письма в ЖТФ*, 38(13):87–95, 2012.

[5] Nikitov S.A. Beginin E.N. Grishin S.V., Sharaevskii Y.P. *IEEE Transactions on Magnetics.*, 47(10), 2011.

[6] Kalinikos B.A. Ustinov A.B., Drozdovskii A.V. *APL*, 96:142, 2010.

[7] Высоцкий С.Л. Никитов С.А., Филимонов Ю.А. *Гетеромагнитная микроэлектроника*, 5:78–86, 2008.

[8] Srinivasan G. Sun N.X. *Spin.*, 2(3):1–46, 2012.

[9] Sergey F. Karmanenko Alexandr A. Semenov Vladislav E. Demidov, Boris A. Kalinikos and Peter Edenhofer. *IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES*, 51(10), 2003.

[10] W. S. Ishak. Magnetostatic wave technology: A review. *Proc. IEEE*, 76:171–187, 1988.

[11] Феррари Сильвестер. *МКЭ для радиофизиков*. Мир, 1986.

Приложение А

.1 Численные параметры структуры

Длина в направлении oZ - $200\mu m$

Длина в направлении oY - $8\mu m$

Длина в направлении oX - $353\mu m$

Отношение глубины канавки к толщине ферритовой плёнки (модуляция) - 25%

Период L - $200\mu m$

.2 Параметры сегнетоэлектрика

Длина в направлении oZ - $100\mu m$

Длина в направлении oY - $200\mu m$

Длина в направлении oX - $353\mu m$

Диэлектрическая проницаемость $\varepsilon = 8000$

.3 Параметры ферритовой плёнки

Магнитный тензор $\hat{\mu}$:

Обозначим

$g = 2.8[\frac{MHz}{Oe}]$

$h_0 = 438[Oe]$

$M_0 = 108[Oe]$

$\omega_h = gh_0$

$\omega_m = 4\pi gM_0$

Пусть

$$m_a = m_a(\omega) = \frac{\omega_m \cdot \omega}{\omega_h^2 - \omega^2}$$

$$m_u = m_u(\omega) = \frac{\omega_h \cdot (\omega_h + \omega_m) - \omega^2}{\omega_h^2 - \omega^2}$$

Тогда

$$\hat{\mu} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{m_u}{m_u^2 - m_a^2} & -j \cdot \frac{m_a}{m_u^2 - m_a^2} \\ 0 & j \cdot \frac{m_a}{m_u^2 - m_a^2} & \frac{m_u}{m_u^2 - m_a^2} \end{bmatrix}$$

Приложение B

.1 Листинг кода на meep для расчета магнонного кристалла

```
(define-param freq 0.386658203776574)
(define-param sx 44)
(define-param sy 11)
(define-param dpml 2)
(define-param sourcewidth 20)
(define-param amp 1)
(define-param cyleps 9.85)    ; dielectric constant of the cylinder
(define-param blockeps 1.75) ; dielectric constant of the space
(define-param r 0.2)
(define-param Tmax 200)

(set! geometry-lattice      ; Making word [sx x sy]
    (make lattice
        (center 0)
        (size sx sy no-size)
    )
)



(set! geometry              ; Making a ball inside
    (list
        (make cylinder (center 0 0) (radius r) (height infinity)
            (material (make dielectric (epsilon cyleps)))
        )
    )
)



(set! geometry (geometric-objects-lattice-duplicates geometry))
                            ; And multiply it by the word



(set! geometry              ; Making waveguide, deleting cylinders.
    (append geometry
        (list
            (make block (center 0 0) (size sx 1 infinity)
               (material (make dielectric (epsilon blockeps)))
            )
        )
    )
```

```
)

(set! symmetries                ; Task has symetric.
    (list
        (make mirror-sym(direction Y))
    )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(set! pml-layers (list (make pml (thickness dpml))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(set! sources
    (list
        (make source
            (src (make continuous-src (frequency freq) (width sourcewidth)))
            (component Ez)
            (center -18 0)
            (size 0 1)
            (amplitude amp)
        )
    )
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(set! resolution 10)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(run-until Tmax
    (at-beginning output-epsilon)
    (to-appended "ez" (at-every 0.5 output-efield-z))
    (at-end output-efield-z)
)
```

Данный код можно выполнить нижеследующим скриптом в рабочем каталоге:

```
# "meep" or "meep-mpi"
meep-mpi foo.ctl >& foo.out
h5topng -S2 -Zc dkbluered foo-eps-000000.00.h5
h5ls ez.h5
h5topng -S2 -t 0:329 -R -Zc dkbluered foo-ez.h5
convert foo-ez.t*.png ez.gif
rm foo-ez.t*.png
```

.2 Листинг кода для расчта матрицы жсткости и масс.

```
#include <iostream>                          // cin/cout.
#include <string>                            // strings
#include <fstream>                           // fin/fout
#include <stdlib.h>                          // System calls.
#include <sstream>                           // Smth for inttostr.
#include <vector>                            // For vector.
#include <algorithm>                         // For vector operations.
#include <boost/numeric/mtl/mtl.hpp>         // For mtl4 (dense*)

using namespace std;
/*
 *      This program take inside .mphtxt file and calculate wave vectors.
 * What important:
 *      # Deps:
 *              * g++
 *              * boost
 *              * mtl4
 *              * matlab
 *              * coreutils
 *          How to install mtl4:
 *              * install boost
 *              * download mtl4 from simunova.com/node/145
 *              * unarchive mtl4, where it's structure tells you.
 *                In that site there's an Installation guide, btw.
 *      # main function at the bottom.
 *      # In this version there's no
 *              * There's no input check:
 *                                  program think, that you gave correct file.
 *              * Small brains, analyzing .mphtxt file.
 *              * There can be strange solution due to inexperience author.
 *                If you can fix something from this list - write to
 *                          marcor@yandex.ru or do it with you own: code is
 *                          under GNU GPL or CC BY-SA (whatever you want)
 *      # Calling syntax - in main()
 *      # This code is under GNU/GPL or CC BY-SA
 *      # Author - $|hbar omk$
 ********************************************************************************/

        /*
         * Stuff.
         */

string inttostr(int number)
{
```

```
        stringstream out;
        out << number;
        return out.str();
}

string floattostr(float number)
{
        stringstream out;
        out << number;
        return out.str();
}

string doubletostr(double number)
{
        stringstream out;
        out << number;
        return out.str();
}

int show_double_matrix(double *A, int n, int m)
{

       /*
        * Show matrix *A[n][m].
        * n - number of lines, m - number of column.
        *
        * return 0.
        */

        int i = 0;
        int j = 0;

                for (i = 0; i < n; i++)
                {
                        cout << "\n";
                                for (j = 0; j < m; j++)
                                {
                                        printf("%8.2F", 5, *(A + i*m + j));
                                }
                        cout << endl;
                }
        return 0;
}

int multmatr(double *A, double *B, double *R, int n, int m, int k)
{
```

```
        /*
         * Multiply matrix *A[n][m] and matrix *B[m][k].
         * *R[n][k] - result matrix.
         *
         * return 0
         */

        system("echo -ne '\''calling multmatr...'\''");

        int i       = 0;
        int j       = 0;
        int r       = 0;
        double tmp  = 0;

            for(i = 0; i < n; i++)
            {
                    for(j = 0; j < k; j++)
                    {
                            tmp = 0;
                                        for(r = 0; r < m; r++)
                                        {
                                        tmp += *(A + i*n + r) * *(B + r*k + j);
                                        }
                                    *(R + i*k + j) = tmp;
                        }
                }

        system("echo -e '\''done.'\''");
        return 0;
}

int transmatr(double *A, double *B, int n, int m)
{

        /*
         * Transpose matrix *A[n][m] to matrix *B[m][n].
         *
         * return 0
         */

        system("echo -ne '\''calling transmatr...'\''");

        int i = 0;
        int j = 0;

                for (i = 0; i < n; i++)
                {
```

```
                for (j = 0; j < m; j++)
                {
                        *(B + j*n + i) = *(A + i*m + j);
                }
                }

        system("echo -e \' done.\'");
        return 0;
}

int findnumbofnodes(string file)
{

       /*
        * This function find, how many nodes is on the bound. .
        *
        * Return amount of bound nodes.
        */

        system("echo -ne \'calling findnumbofnodes...\'");

        int N               = 0;

        string Str          = "";

        string startanchor  = "# Mesh point coordinates";
        string endanchor    = "3 # number of element types";

        ifstream fin(file.c_str());

                while (getline(fin, Str) && (Str != startanchor))
                {
                        /* nothing*/
                }
                while (getline(fin, Str) && (Str != endanchor))
                {
                        N++;
                }

        fin.close();
        system("echo -ne \' done.\'");
        // N--;
        cout << " Result: N = " << N << endl;
        return N;
}

int getcoords(string file, double *x, double *y)
```

```
{

        /*
         * This function find coordinates of all nodes
         *
         * return 0
         */

        system("echo -ne \'calling getcoords...\'");

        string Str          = "";
        string Stroutx      = "";
        string Strouty      = "";

        string startanchor  = "# Mesh point coordinates";
        string endanchor    = "3 # number of element types";

        bool k              = false;
        int i               = 0;
        int nend            = 0;
        int strlength       = 0;

        ifstream fin(file.c_str());

                while (getline(fin, Str) && (Str != startanchor))
                {
                        /* nothing */
                }

                while (getline(fin, Str) && (Str != endanchor))
                {
                        k = false;
                        strlength = Str.length();
                        // Due to string looks like x0 y0 x1 y1 x2 y2 ...
                        // I'm alternating here.
                                for (i = 0; i < strlength; i++)
                                {
                                        if (Str[i] == ' ')
                                        {
                                                k = true;
                                        }

                                        if (k == false)
                                        {
                                                Stroutx += Str[i];
                                        }
                                        else
```

```
                                        {
                                                Strouty += Str[i];
                                        }
                                }

                        x[nend] = atof(Stroutx.c_str());
                        y[nend] = atof(Strouty.c_str());

                        Stroutx = "";
                        Strouty = "";
                        nend++;
                }

        fin.close();
        system("echo -e \'done.\'");
        return 0;
}

string findsizeGlS(string file)
{
        /*
         * This function find size of mass matrix.
         *
         * return amount of bound nodes
         */

        system("echo -ne \'calling findsizeGlS...\'");

        int i               = 0;
        int j               = 0;
        int k               = 0;
        int tmp             = 0;
        int strlength       = 0;

        vector <int> a;

        string endanchor    = "2 # number of parameter values per element";

        string Str          = "";
        string Strout       = "";
        string Out          = "";


        ifstream fin(file.c_str());

        // There's three string, ending like this.
        string startanchor  = " # number of elements";
```

```
    while (getline(fin,Str))
    {
        if (Str.find(startanchor) != -1)
        {
            i++;
                if (i == 2)
                {
                    break;
                }
        }
    }
// Taking second
startanchor = Str;

fin.close();
fin.open(file.c_str());


// Finding, how many bound nodes with repetition are if file
        while (getline(fin, Str) && (Str != startanchor))
        {
                /* nothing */
        }
getline(fin, Str);
        while (getline(fin, Str) && (Str != endanchor))
        {
                Strout = Strout + "␣" + Str;
                k++;
        }
k = k * 2;



        for (i = 0; i < k; i++)
        {
                a.push_back(0);
        }
j = 0;

strlength = Strout.length();
        for (i = 0; i < strlength; i++)
        {
                if (Strout[i] != '␣')
                {
                        Out += Strout[i];
                }
```

```
                        else
                        {
                                a[j] = atoi(Out.c_str());
                                j++;
                                Out = "";
                        }
                }

        sort(a.begin(), a.end());

        // Deleting repetition
        unique(a.begin(), a.end());
        Strout = "";
        for (i = 0; i < k-1; i++)
        {
                if (a[i] < a[i+1])
                {
                        Strout += inttostr(a[i]);
                        Strout += ",";
                }
                else
                {
                        break;
                }
        }

        fin.close();
        system("echo -e \'done.\'");
        cout << Strout << endl;
        return Strout;
}

int findnumbtr(string file)
{

       /*
        * This function fing, how many elements in mesh we have.
        * return amount elements in mesh.
        */

        system("echo -ne \'calling findnumbtr...\'");

        int N               = 0;
        int i               = 0;

        string Str          = "";
        string Strout       = "";
```

```
        ifstream fin(file.c_str());

        // There's three strings, ending like this
        string startanchor  = "␣#␣number␣of␣elements";

            while (getline(fin,Str))
            {
                if (Str.find(startanchor) != -1)
                {
                    i++;
                        if (i == 3)
                        {
                            break;
                        }
                }
            }
        // Taking third
        startanchor == Str;

                for (i = 0; i < Str.length(); i++)
                {
                        if (Str[i] != '␣')
                        {
                                Strout += Str[i];
                        }
                        else
                        {
                                break;
                        }
                }

        N = atoi(Strout.c_str());

        fin.close();
        system("echo␣-e␣\'␣done.\'");
        return N;
}

string findtrangle(string file, int N)
{

       /*
        * This function return global numbers of triangle's N nodes
        *       node1,node2,node3,
        *
        * With a comma in the end, yep.
```

```
*/

// system("echo -ne \'calling findtrangle...\'");

string endanchor    = "3 # number of parameter values per element";

string Str          = "";
string Strout       = "";

int i               = 0;
int strlength       = 0;

ifstream fin(file.c_str());

// There's three strings, ending like that..
string startanchor  = " # number of elements";

    while (getline(fin,Str))
    {
        if (Str.find(startanchor) != -1)
        {
            i++;
                if (i == 3)
                {
                    break;
                }
        }
    }
startanchor = Str;

fin.close();
fin.open(file.c_str());

        while (getline(fin, Str) && (Str != startanchor))
        {
                /* nothing */
        }
getline(fin, Str);
        for (i = 0; i <= N; i++)
        {
                getline(fin, Str);
        }

strlength = Str.length();
        for (i = 0; i < strlength; i++)
        {
                if ((Str[i] >= '0') && (Str[i] <= '9'))
```

```
                        {
                                Strout += Str[i];
                        }
                        else
                        {
                                Strout += ",";
                        }
                }

        // To ending a string with a comma, it has to be like tis.
        Strout += ",";
        fin.close();
        return Strout;
}

int find_locale_matrix (string Str, double *GlX, double *GlY,
                        double *GlS, int GlSn, double *GlT, int GlTn)
{
        /*
         * This function compute matrix S and T
         * (mass & stiffness) for one triangle.
         * And write it to global matrix *GlS and *GlT
         *
         * return 0.
         */

         // system("echo -ne \'calling find_locale_matrix...\'");

        int a[3];                       // Array of local numbering
        int k                   = 0;    // and it's index.

        int i                   = 0;
        int j                   = 0;

        vector <double> x(3);
        vector <double> y(3);

        string Strtmp           = "";

        int first_nod           = 0;
        int second_nod          = 1;
        int third_nod           = 2;

        int trianglenod_first   = 0;
        int trianglenod_second  = 1;
        int trianglenod_third   = 2;
```

```
int buf                 = 0;

double A                = 0;        // Square of a triangle.
double CTNG             = 0;        // Dunno
double T[3][3];                     // Mass matrix
double S[3][3];                     // stiffness matrix

        // Separate Str to globar coords.
        // It will be first-, second- and third_nod
        for (i = 0; i < Str.length(); i++)
        {
                if (Str[i] != ',')
                {
                        Strtmp += Str[i];
                }
                else
                {
                        a[k] = atoi(Strtmp.c_str());
                        Strtmp = "";
                        k++;
                }
        }

x[first_nod]  = GlX[a[0]];  //
x[second_nod] = GlX[a[1]];  // Nodes coords.
x[third_nod]  = GlX[a[2]];  //

y[first_nod]  = GlY[a[0]];  //
y[second_nod] = GlY[a[1]];  //
y[third_nod]  = GlY[a[2]];  //

A =
abs(((x[second_nod] - x[first_nod]) * (y[third_nod] - y[first_nod]) -
(x[third_nod] - x[first_nod]) * (y[second_nod] - y[first_nod])) / 2);

        // Making matrix T
        for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
        {
                S[i][j] = 0;
                T[i][j] = 0;
        }

        for (i = 0; i < 3; i++)
        {
                        for (j = 0; j < 3; j++)
                        {
```

```
                                T[i][j] = A / 12;
                        }
                T[i][i] = 2 * T[i][i];
        }

GlT[a[0] * GlTn + a[0]] += T[0][0]; ////////
GlT[a[0] * GlTn + a[1]] += T[0][1]; ////
GlT[a[0] * GlTn + a[2]] += T[0][2]; //
GlT[a[1] * GlTn + a[0]] += T[1][0]; // Writing it to global
GlT[a[1] * GlTn + a[1]] += T[1][1]; // matrix GlT.
GlT[a[1] * GlTn + a[2]] += T[1][2]; //
GlT[a[2] * GlTn + a[0]] += T[2][0]; //
GlT[a[2] * GlTn + a[1]] += T[2][1]; ////
GlT[a[2] * GlTn + a[2]] += T[2][2]; ////////

        // Making matrix S

        for (i = 1; i < 3; i++)
        {
                CTNG = (( (x[second_nod] - x[first_nod])
                        * (x[third_nod] - x[first_nod]))
                        + ((y[second_nod] - y[first_nod])
                        * (y[third_nod] - y[first_nod])))
                        / (4 * A);

                S [trianglenod_second] [trianglenod_second] += CTNG;
                S [trianglenod_second] [trianglenod_third]  -= CTNG;
                S [trianglenod_third]  [trianglenod_second] -= CTNG;
                S [trianglenod_third]  [trianglenod_third]  += CTNG;

                                        buf = trianglenod_first;
                 trianglenod_first = trianglenod_second;
                trianglenod_second = trianglenod_third;
                 trianglenod_third = buf;

                        buf = first_nod;
                 first_nod = second_nod;
                second_nod = third_nod;
                 third_nod = buf;
        }

GlS[a[0] * GlSn + a[0]] += S[0][0]; ////////
GlS[a[0] * GlSn + a[1]] += S[0][1]; ////
GlS[a[0] * GlSn + a[2]] += S[0][2]; //
GlS[a[1] * GlSn + a[0]] += S[1][0]; // Writing it to global
GlS[a[1] * GlSn + a[1]] += S[1][1]; // matrix GlS.
GlS[a[1] * GlSn + a[2]] += S[1][2]; //
```

```
        GlS[a[2] * GlSn + a[0]] += S[2][0]; //
        GlS[a[2] * GlSn + a[1]] += S[2][1]; ////
        GlS[a[2] * GlSn + a[2]] += S[2][2]; ////////

        // system("echo -e \' done.\'");
        return 0;
}

int make_identity_matrix(double *C, int Cn, int GlSn, int *nodes, int k)
{

       /*
        * Making identity matrix C[GlSn x Cn].
        */

system("echo -ne \'calling make_identity_matrix...\'");

int buf_height  = 0;
int buf_width   = 0;
int i           = 0;
int j           = 0;
vector <int> local_nodes;

        for (i = 0; i < k; i++)
        {
                local_nodes.push_back(nodes[i]); // I wanna work with vector.
        }

        for (i = 0; i < GlSn; i++)      // For every bound node
        {
                if (i <= k)
                {
                        for (j = 0; j <= k; j++)
                        {
                                if (local_nodes.size() > 0)
                                {
                                        // IF node on the bound
                                        if (i == local_nodes[j])
                                        {
                                        buf_height++;
                                        local_nodes.erase(local_nodes.begin());
                                        break;
                                        }
                                        else
                                        {
                                        C[buf_height * Cn + buf_width] = 1;
                                        buf_width++;
```

```
                                        buf_height++;
                                        break;
                                        }
                                }
                        }
                }
                else
                {
                        C[buf_height * Cn + buf_width] = 1;
                        buf_height++;
                        buf_width++;
                }
        }

        system("echo -e \'done.\'");
        return 0;
}

int call_matlab(double *A, int An, double *B, int Bn)
{
        /*
        * Str is a calling string
        */
        system("echo -ne \'calling matlab...\'");

        int i                   = 0;
        int j                   = 0;
        string matlabdir        = "/usr/local/MATLAB/R2014b/bin/matlab";
        string workfolderdir    = "/home/homk/prg/dip";
        string Str              = "";

        Str = "rm " + workfolderdir + "/outS.dat " + workfolderdir
                + "/outT.dat 2>/dev/null";
        system(Str.c_str());

        Str = workfolderdir + "/outS.dat";
        ofstream fout(Str.c_str());

                for (i = 0; i < An; i++)
                {
                                for (j = 0; j < An; j++)
                                {
                                        fout << doubletostr(A[i * An + j]);
                                        fout << " ";
                                }
                        fout << "\n ";
                }
```

```
        fout.close();
        Str = workfolderdir + "/outT.dat";
        fout.open(Str.c_str());

                for (i = 0; i < Bn; i++)
                {
                        for (j = 0; j < Bn; j++)
                        {
                                fout << doubletostr(B[i * Bn + j]);
                                fout << " ";
                        }
                fout << "\n ";
                }
        fout.close();

        Str = matlabdir + " -nodisplay -nosplash -nodesktop -nojvm -r
 \"S = load(\'" + workfolderdir + "/outS.dat\') ; T = load(\'"
        + workfolderdir + "/outT.dat\') ; eigS = eig(S) ; eigT = eig(T)
 ; save " + workfolderdir + "/eigS.dat eigS -double -ascii
 ; save " + workfolderdir + "/eigT.dat eigT -double -ascii
 ; exit\"";;

        system(Str.c_str());
        Str = "date && echo -e 'eig(S):' && echo -e '\n' && cat "
        + workfolderdir + "/eigS.dat && echo -e '\neig(T):'
 && cat " + workfolderdir + "/eigT.dat";
        system(Str.c_str());

        system("echo -e \' done.\'");
        return 0;
}

////////////////////////////////
////////////////    _    ///////
//  _ __ ___   __ _(_)_ __    //
// | '_ ` _ \ / _` | | '_ \   //
// | | | | | | (_| | | | | |  //
// |_| |_| |_|\__,_|_|_| |_|  //
////////////////////////////////

        /*
         * Call: programm_name path_to_file_.mphtxt bound_cond
         *
         * path_to_file_.mphtxt better to write in full form.
         *
         * bound_count can be:
```

```
         *      * d - Dirihle,
         *      * n - Neymann,
         *      * f - Floke.
         */

int main(int argc, char *argv[])
{
        int GlNoftr             = 0;
        int GlSn                = 0;
        int GlTn                = 0;
        int GlXn                = 0;
        int GlYn                = 0;
        // mtl::dense_vector <double> GlT(GlTn * GlTn);
        // mtl::dense_vector <double> GlS(GlSn * GlSn); I'll use it later
        vector <double> GlX;
        vector <double> GlY;
        vector <int> nodes;
        double A                = 0;
        int i                   = 0;
        int j                   = 0;
        int r                   = 0;
        int k                   = 0;
        int Cn                  = 0;
        double tmp              = 0;
        string file             = argv[1];
        char* bound_cond        = argv[2];
        string Str              = "";
        string nod_str          = "";

        Str = "sed -ri \'s/\\s+$//g\' " + file;
        system(Str.c_str());

                if (argc < 2)
                {
                        cout << "Not enough options." << endl
                        << "Program terminated." << endl;
                        exit(1);
                }

        GlXn = findnumbofnodes(file);
        GlYn = GlXn;
        GlSn = GlXn;
        GlTn = GlSn;

        mtl::dense_vector <double> GlT(GlTn * GlTn);
        mtl::dense_vector <double> GlS(GlSn * GlSn);
        GlT = 0;
```

```
GlS = 0;

        for (i = 0; i < GlXn; i++)
        {
                GlX.push_back(0);
        }

        for (i = 0; i < GlYn; i++)
        {
                GlY.push_back(0);
        }

getcoords(file, &GlX[0], &GlY[0]);

GlNoftr = findnumbtr(file);

        for (i = 0; i < GlNoftr; i++)
        {
                Str = findtrangle(file, i);
                find_locale_matrix(Str, &GlX[0], &GlY[0], &GlS[0],
                        GlSn, &GlT[0], GlTn);
        }

        switch (*bound_cond)
        {
        case 'd':
                cout << endl << "Matrix S:" << endl;
                show_double_matrix(&GlS[0], GlSn, GlSn);
                cout << endl;

                cout << endl << "Matrix T:" << endl;
                show_double_matrix(&GlT[0], GlTn, GlTn);
                cout << endl;

                call_matlab(&GlS[0], GlSn, &GlT[0], GlTn);
                break;
        case 'f':
                cout << endl << "Be soon.";
                cout << endl;
                break;
        case 'n':
                Str = findsizeGlS(file);    // Finding bound nodes.
                // Transform Str "number,number,number"
                // to int array[number, number, number]
                        for (i = 0; i < Str.length(); i++)
                        {
                                if (Str[i] == ',')
```

```
                                        {
                                        nodes.push_back(atof(nod_str.c_str()));
                                        k++;
                                        nod_str = "";
                                        }
                                        else
                                        {
                                        nod_str += Str[i];
                                        }
                                }

                        Cn = GlSn - k;        // Size of matrixes outS2 and outT2

                        // Identity matrix
                        mtl::dense_vector <double> C(Cn * GlSn + GlSn);
                        // Transpose identity matrix.
                        mtl::dense_vector <double> C_trans(Cn * GlSn + GlSn);

                        int outST1n = GlSn * Cn;
                        int outST2n = Cn * Cn;
                        double outS1[outST1n];
                        double outS2[outST2n];
                        double outT1[outST1n];
                        double outT2[outST2n];

                        C       = 0;
                        C_trans = 0;

                make_identity_matrix(&C[0], Cn, GlSn, &nodes[0], k);
                transmatr(&C[0], &C_trans[0], GlSn, Cn);

                // GlS here transform to outS2
                multmatr(&GlS[0], &C[0], &outS1[0], GlSn, GlSn, Cn);
                multmatr(&C_trans[0], &outS1[0], &outS2[0], GlSn, GlSn, Cn);

                // And GlT - to outT2.
                multmatr(&GlT[0], &C[0], &outT1[0], GlSn, GlSn, Cn);
                multmatr(&C_trans[0], &outT1[0], &outT2[0], GlSn, GlSn, Cn);

                call_matlab(&outS2[0], Cn, &outT2[0], Cn);
                }
    return 0;
}
```