МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра математического и компьютерного моделирования

РАБОТА ПРОВЕРЕНА
Рецензент, доцент кафедры УМФ
канд. физ.-мат. наук, доцент
_____/ Д.Е. Шафранов
«____»_____2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой МиКМ,
д-р физ.-мат. наук, доцент
_____/ С.А.Загребина
«____»_____2019 г.

3D МОДЕЛИРОВАНИЕ ВИРТУАЛЬНЫХ ЗАЦЕПЛЕНИЙ РОДА 1 МАЛОЙ СЛОЖНОСТИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ–02.03.01.2019.41.ВКР

Руководитель работы,
доцент кафедры МиКМ,
канд. физ.-мат. наук, доцент
_____/ А.А. Акимова
«____»_____2019 г.

Автор работы,
студент группы ЕТ-411
_____/ К.А. Брынза
«____»_____2019 г.

Нормоконтролер,
доцент кафедры МиКМ,
канд. физ.-мат. наук
_____/ А.А. Акимова
«____»_____2019 г.

Челябинск 2019

АННОТАЦИЯ

Брынза К.А. 3D моделирование виртуальных зацеплений рода 1 малой сложности. – Челябинск: ЮУрГУ, ЕТ-411, 22 с., 12 ил., библиогр. список – 13 наим., 4 прил.

Данная работа выполнена с целью получения навыков работы с графическим моделированием, разработки метода и алгоритма визуализации виртуальных зацеплений рода 1 малой сложности и анализа.

В квалификационной работе разработан метод визуализации виртуальных узлов в утолщенном торе, сложность которых не превышает 3 в среде программирования MatLab.

Также был изучен вектор развития теории узлов, а именно сделан обзор на существующее программное обеспечение, реализующее визуализацию узлов и зацеплений, что поможет и в развитии другой науки – биологии. Был изучен проект, которые предполагает изучать молекулы белка при помощи теории узлов.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Классический узел представляет собой произвольную простую замкнутую кривую в трехмерном пространстве R^3. При изображении классических узлов принято использовать их проекции на плоскость. При этом перекрестками называются точки плоскости, в которые проектируются две различные точки узла. Диаграмма узла получается из проекции указанием (путем разрыва нити в окрестности каждого перекрестка), какой из участков узла проходит выше другого. На рисунке 0.1 показан пример классического узла, лежащего в трехмерной сфере S^3, его проекция и диаграмма на плоскости.

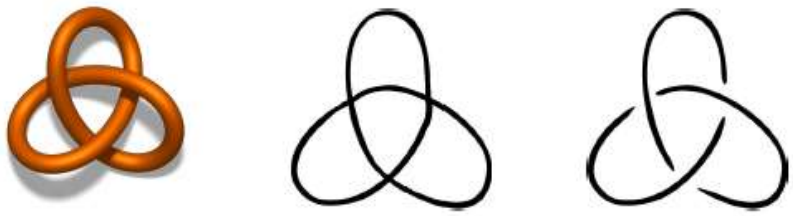


Рисунок 0.1. Пример трилистника (классического узла), его проекция и диаграмма на плоскости

В последнее время усилился интерес к зацеплениям в трехмерных многообразиях типа $F \times I$, которые называются виртуальными. Они представляют собой зацепления в утолщенных поверхностях $F \times I$, где F – ориентированная замкнутая поверхность, I — ориентированный отрезок, с точностью до стабилизации/дестабилизации. Замкнутая ориентируемая поверхность – это двусторонняя поверхность с выбранным направлением нормали не имеющая границ. Под дестабилизацией мы понимаем следующее. Пусть S – некоторая нестягиваемая окружность на замкнутой двумерной ориентированной поверхности F, для которой существует цилиндр C, лежащий в $F \times \{0,1\}$, с краями па разных краях многообразия $F \times \{0,1\}$, гомотопный цилиндру $S \times I$, причем цилиндр C с краями не пересекает зацепления. Тогда дестабилизация – это разрезание двумерного многообразия

$F \times I$ вдоль цилиндра с заклеиванием появившихся компонент края шайбами $D^2 \times I$ (рисунок 0.2). Под стабилизацией мы понимаем операцию, обратную к дестабилизации.

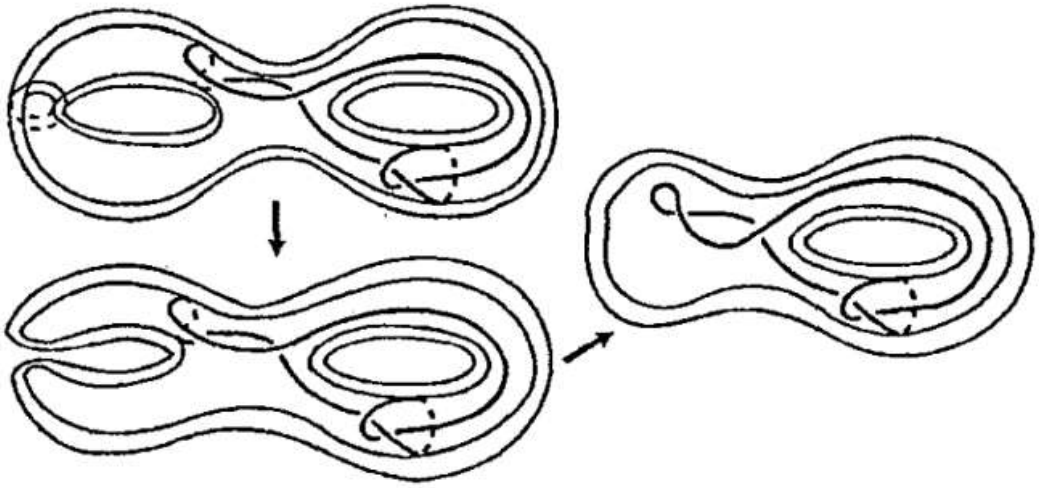


Рисунок 0.2. Дестабилизация пары ($S_2 \times I$, K), где S_2 – сфера с двумя ручками

Усиливающийся интерес к виртуальным зацеплениям можно объяснить тем, что на них обобщаются многие инварианты узлов в S^3, а теории узлов в S^3 и в утолщенной сфере $S^2 \times I$ совпадают. Под инвариантом здесь понимается функция, значения которой сохраняются неизменными при попытке из одного узла получить другой, не разрывая при этом нить. Поскольку тор $T = S^1 \times S^1$ – самая простая замкнутая ориентируемая поверхность после сферы, то теория узлов в $T \times I$ является естественным продолжением теории узлов в S^3.

Зацепления в утолщенных поверхностях могут иметь несколько нитей. Виртуальный узел же является частным случаем зацепления, при котором узел образуется из одной нити.

В данной работе мы рассматриваем виртуальные узлы рода 1. Род виртуального узла K – это наименьший род поверхности S такой, что K расположен в $S \times I$. Род поверхности – это число, характеризующее порядок связности поверхности. Каждую замкнутую ориентируемую поверхность можно взаимно однозначно и непрерывно отобразить на сферу с p ручками. Число p называется родом такой поверхности. Род поверхности тора равен 1, поэтому в данной работе мы рассматриваем узлы в утолщенном торе $T \times I$.

Как и в классическом случае, узлы в утолщенном торе $T \times I$ можно задавать проекциями и диаграммами (пример на рисунке 0.3). Проекция некоторого узла на утолщенный тор $K \subset T \times I$ представляет собой регулярный граф $G \subset T$ степени 4. Регулярный граф – граф, степени вершин которого равны, т.е. в таком графе каждая вершина имеет одинаковое количество соседей. В нашем случае, для регулярного графа $G \subset T$ прохождение вершин по правилу «прямо вперед» определяет полный обход, отвечающий этому узлу. Диаграмма узла K получается из этого графа указанием (путем разрывов обхода), какой из проходящих через каждую вершину участков узла расположен выше, какой – ниже другого в смысле величины координаты $t \in I$.

Рисунок 0.3. Пример диаграммы виртуального узла в утолщенном торе $T \times I$

Прохождение нити через вершины узла задается Гауссовым кодом. Он содержит список перекрестков диаграммы, выписанных в том порядке, в котором они встречаются при прохождении узла вдоль его ориентации.

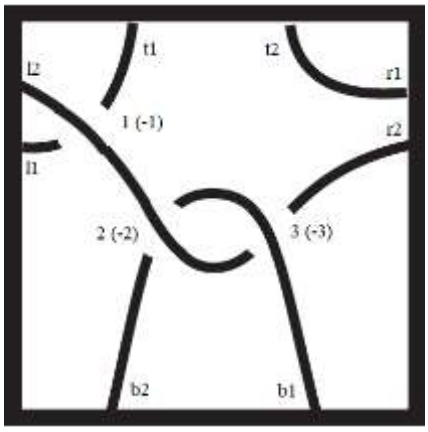


Рисунок 0.4. Пример диаграммы узла с обозначениями кода Гаусса

Пример кода Гаусса для виртуального узла, представленного на рисунке 0.4: 1 2 -3 r2 l1 -1 t1 b2 -2 3 b1 t2 r1.

Главной целью данной работы является визуализация узлов в утолщенном торе $T \times I$, сложность которых не превышает 3 (т.е. узлов, диаграммы которых содержат не более трех перекрестков). Данные узлы взяты из статьи [1] и продублированы в Приложении А. Для достижения поставленной цели необходимо решить следующие задачи.

1. С помощью параметрического уравнения задать внутреннюю и внешнюю поверхность утолщенного тора.
2. Задать координаты точек, которые будут соответствовать перекресткам диаграммы для всех узлов, диаграммы которых имеют не более трех перекрестков.
3. Используя интерполяцию с помощью кубических сплайнов нарисовать нить в утолщенном торе, проходящую через каждый перекресток в соответствии с Гауссовым кодом.

К тому же, еще одной целью является поиск вектора развития теории узлов, как науки, для чего нужно сделать обзор существующих баз данных и программ, визуализирующих классические и виртуальные зацепления. А также найти точки соприкосновения теории узлов с другими науками.

1. ВИЗУАЛИЗАЦИЯ ВИРТУАЛЬНЫХ УЗЛОВ РОДА 1 В СРЕДЕ ПРОГРАММИРОВАНИЯ MATLAB

1.1. Постановка задачи

Необходимо визуализировать виртуальные узлы рода 1, сложность которых не превышает 3.

Обозначим общий алгоритм действий.

1. С помощью параметрических уравнений нарисовать внутреннюю и внешнюю поверхность утолщенного тора.
2. Задать координаты перекрестков.
3. Определить опорные точки нити на всей поверхности утолщенного тора.
4. Составить матрицу всех точек узла из координат перекрестков и опорных точек нити.
5. Интерполировать матрицу методом кубических сплайнов.
6. Нарисовать нить по полученной матрице.

1.2. Математическая модель программы

На первом шаге нам необходимо изобразить два тора, один из которых будет непрозрачной внутренней поверхностью, а второй – прозрачной внешней поверхностью утолщенного тора.

Рассмотрим тор (рисунок 1.1), полученный вращением образующей окружности вокруг оси *Oz*, заданный параметрическим уравнением:

$$\begin{cases} x(\varphi,\psi) = (R + r\,cos\psi)\;cos\varphi \\ y(\varphi,\psi) = (R + r\,cos\psi)\;sin\varphi\,, \\ z(\varphi,\psi) = r\,sin\varphi \end{cases} \qquad \varphi \in [0, 2\pi), \psi \in [-\pi, \pi)$$

где *R* – радиус вращения вокруг оси *Oz*, а *r* – радиус образующей окружности.

Чтобы визуализировать внутреннюю поверхность утолщенного тора была использована функция *surfl(x, y, z)*. Данная функция выводит на экран

затененную поверхность с подсветкой. Так же была применена функция *alpha(const)*, где *const* – число от 0 до 1, задающее степень прозрачности.

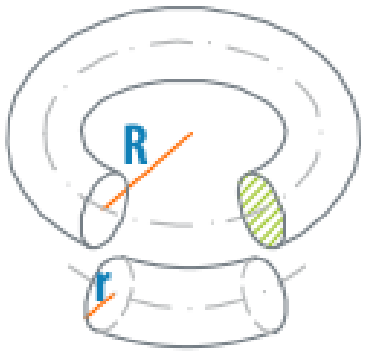


Рисунок 1.1. Изображение тора

Различие в визуализации внешней и внутренней поверхности утолщенного тора состоит только в увеличении радиуса образующей окружности и уровне прозрачности фигуры, задаваемой функцией *alpha(const)*. Прозрачность нужна для того, чтобы была видна нить, лежащая внутри утолщенного тора. Визуализированная поверхность утолщенного тора представлена на рисунке 1.2.

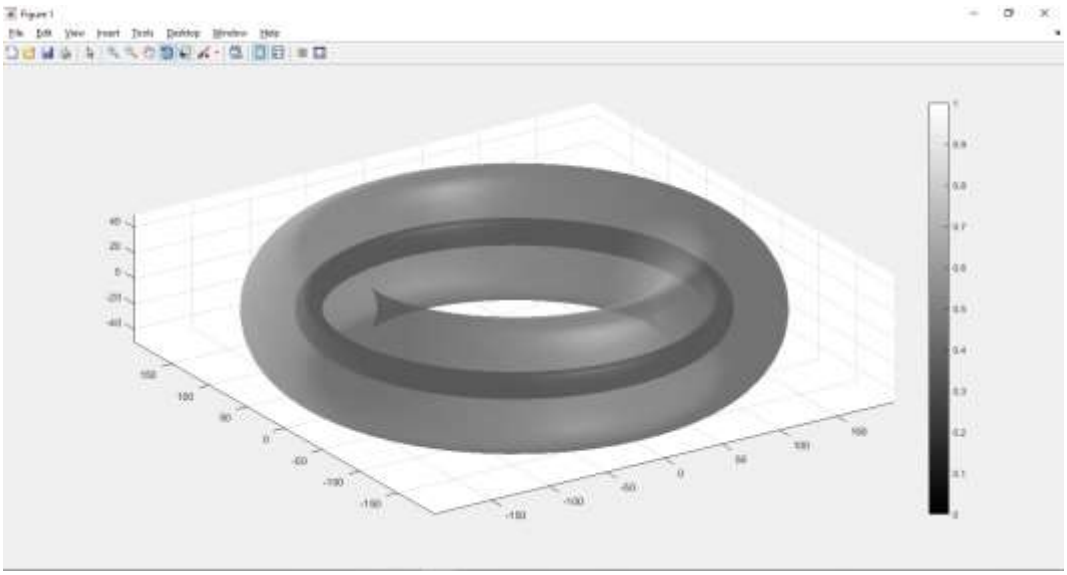


Рисунок 1.2. Поверхность утолщенного тора

Следующий шаг визуализации виртуального узла рода 1 – расстановка точек, которые соответствуют перекресткам диаграммы узла внутри реализованной поверхности утолщенного тора. Для осуществления этой

задачи мы воспользуемся диаграммами виртуальных узлов, указанных в работе [1] и Приложении А.

Алгоритм размещения таких точек.

1. Определим примерную область внутри утолщенного тора, где будет размещаться виртуальный узел.
2. Внутри этой области обозначим по две точки, которые соответствуют одному перекрестку диаграммы узла: одна точка для прохождения нити сверху, вторая точка для прохождения нити снизу.
3. Размещение точек следует соотнести с расположением их на диаграмме узла

Итак, на данном этапе мы имеем визуализированный утолщенный тор и координаты точек, соответствующих перекресткам диаграммы виртуального узла (рисунок 1.3).

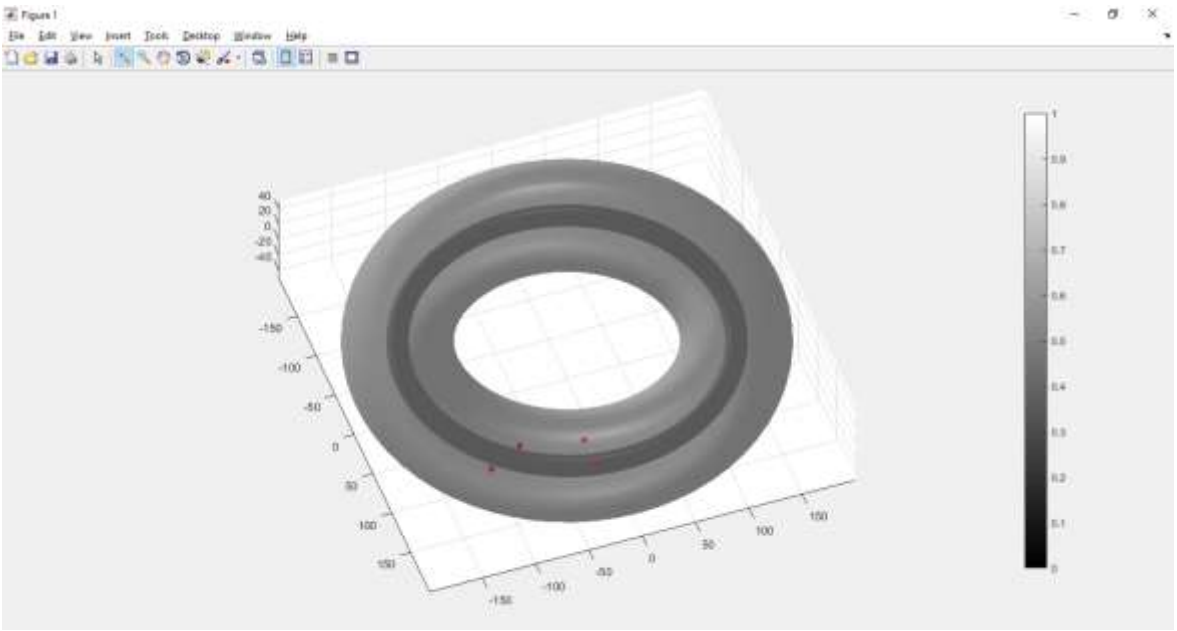


Рисунок 1.3. Поверхность утолщенного тора с расставленными точками, которые соответствуют перекресткам диаграммы узла

Теперь нужно расставить оставшиеся опорные точки, которые соответствуют точкам прохождения нити через нижнюю и верхнюю грань диаграммы. Таких точек нам понадобится всего 3, причем первая и третья из них будут иметь одинаковые координаты по x и y, а координата по z будет в

этих двух точках будет равноудалена от 0. Третья точка будет смещена ближе к внутреннему радиусу внешнего тора и ее координата по оси *z* будет равна 0. Расставляем эти координаты в той же области, в которой мы расставляли координаты точек, соответствующих перекресткам диаграммы, соотнося точки с диаграммой виртуального узла.

Следующим действием будет определение опорных точек, которые соответствуют точкам прохождения нити через левую и правую грань диаграммы. Для этого сформируем массив элементов с помощью функции *linspace(x1, x2, n)*, которая формирует линейный массив размера 1 × *n* начальным и конечным элементами которого являются точки *x1* и *x2*. Возьмем начальный элемент *x1* равный $\frac{\pi}{3}$, а конечный элемент *x2* равный $\frac{5\pi}{3}$. Всего нам понадобится 10 элементов. После этого сформируем еще один массив элементов, который нам понадобится в случае, когда мы будем замыкать нить. Для этого массива так же воспользуемся функцией *linspace(x1, x2, n)*, где *x1* = $\frac{\pi}{12}$, а *x2* = $\frac{\pi}{3}$. Таких элементов нам понадобится всего 4. Выбор начальных элементов обусловлен сглаживанием нити при ее замыкании.

Далее, сформируем массив элементов, в котором будут содержаться координаты опорных точек. Для этого воспользуемся параметрическим уравнением окружности:

$$\begin{cases} x(\varphi) = r\cos\varphi_i \\ y(\varphi) = r\sin\varphi_i, \\ z(\varphi) = 0 \end{cases} \qquad i = 1,2,\dots,n.$$

где r – радиус утолщенного тора, увеличенный на некоторую константу для того, чтобы нить не пересекала поверхность утолщенного тора, а φ – элементы массива, сформированного в предыдущем абзаце.

В случае, если визуализируемый узел имеет несколько точек на левой грани (и соответственно правой), необходимо сформировать два массива φ_i и два массива с координатами опорных точек, но $z(\varphi)$ должна быть равноудалена от 0.

Следующий этап – формирование матрицы всех опорных точек узла, которые включают в себя:

1) точки, соответствующие перекресткам диграммы;
2) опорные точки, которые соответствуют точкам прохождения нити через нижнюю и верхнюю грань диаграммы;
3) опорные точки, которые соответствуют точкам прохождения нити через левую и правую грань диаграммы.

Формировать такую матрицу нужно согласно Гауссова кода, который описывает движение нити виртуального узла.

Обозначения кода Гаусса:

1) «C» – прохождение нити сверху перекрестка;
2) «-C» – прохождение нити снизу перекрестка;
3) «b1» – прохождение узла через нижнюю грань диаграммы, где буква означает грань диаграммы («b» – нижняя, «t» – верхняя, «l» – левая, «r» – правая), а цифра означает номер пересечения на одной грани.

Сформированный массив опорных точек нити показан на рисунке 1.4.

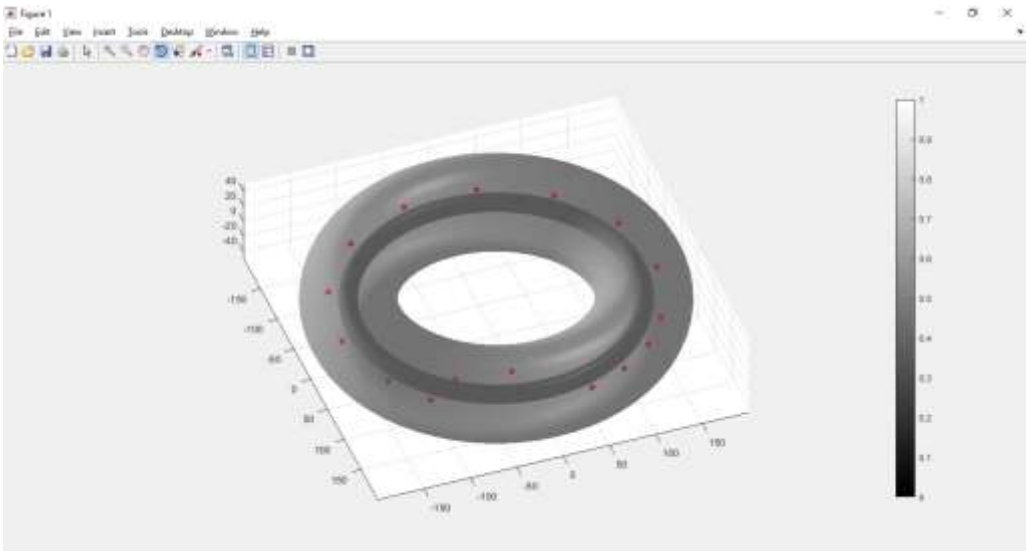


Рисунок 1.4. Все опорные точки нити в утолщенном торе

После формирования матрицы всех опорных точек узла нужно интерполировать координаты, содержащиеся в этой матрице. Для этого используем функцию *spline(x, y, xi)*, которая интерполирует функцию *y* в

точках *xi* внутри области определения функции, используя кубические сплайны. Входящие данные для этой функции возьмем следующие:

1) *y* – массив опорных точек, содержащийся в сформированной матрице;
2) *x* – индексы массивов этой матрицы;
3) *xi* – индексы массивов, разделенные на 0,01 (точность интерполирования).

После этого нам осталось применить функцию *plot3(x, y, z)* для прорисовки виртуального узла в утолщенном торе, где *x*, *y* и *z* – первая, вторая и третья строки матрицы опорных точек. Результат работы программы представлен на рисунке 1.5.

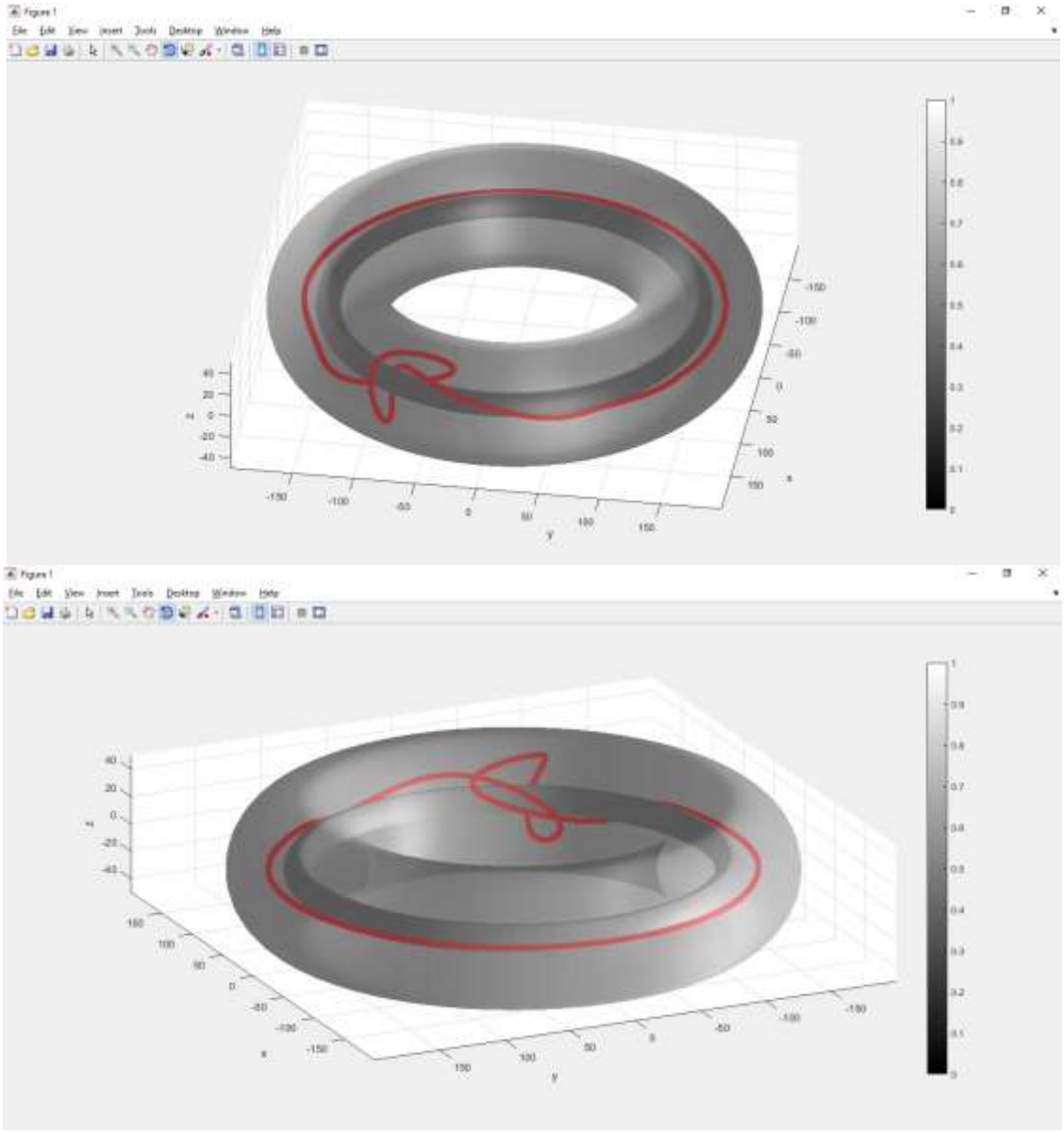


Рисунок 1.5. Виртуальный узел рода 1 сложности 2

1.3. Выводы по главе 1

В результате разработан алгоритм 3Д визуализации виртуальных рода 1, сложность которых не превышает 3.

Также реализована программа для всех виртуальных узлов в утолщенном торе сложности 2 и 3 в среде программирования MatLab. Блок-схема программы представлена в Приложении Б, результат работы – в Приложении В, а текст программы – в приложении Г.

В процессе разработке было использовано: параметрическое уравнение тора (см. [3]), параметрическое уравнение окружности (см. [3]), интерполяция кубическими сплайнами, справочник MatLab (см. [2]).

2. ОПРЕДЕЛЕНИЕ ВЕКТОРА РАЗВИТИЯ ТЕОРИИ УЗЛОВ

2.1. Базы данных классических и виртуальных узлов

2.1.1. База данных Лукаса Леварка

База данных Лукаса Леварка (см. [9]) – одна из самых полных баз данных, в которой собраны полезные программы для исследования теории узлов.

Автор, Лукас Леварк – математик, работающий в области низкоразмерной топологии и теории узлов. В частности, его интересует согласованность узлов и гомология Хованова-Розанского.

2.1.2. Knot Atlas

Википедия по теории узлов. Свободно редактируется пользователями. Сервер находится под управлением департамента математики университета Торонто в Канаде.

Knot Atlas (см. [5]) хранит проекции и плоские изображения классических виртуальных узлов и зацеплений, которые содержат не более 11 перекрестков и пересечений. Виртуальные узлы отсутствуют.

2.1.3. KnotInfo и LinkInfo

KnotInfo (см. [4]) и LinkInfo (см. [7]) был создан и поддерживается Чаком Ливингстоном при содействии Чхе Чун Ча. Так же как и Knot Atlas находится под управлением департамента математики университета Торонто в Канаде.

База данных отдельно по классическим узлам и отдельно по зацеплениям с удобным поиском с множеством фильтров. Для каждого узла и зацепления представлены всевозможные параметры, включая диаграммы.

В LinkInfo хранятся зацепления, которые имеют не более 11 пересечений, а в KnotInfo – узлы, имеющие не более 12 перекрестков.

2.2. Существующие программы для визуализации узлов

2.1.1. KnotPlot

Сложная программа KnotPlot (см. [13]), которую разработал Роб Шарейн (см. [11]), для визуализации и управления математическими узлами в трех или четырех измерениях. Пример показан на рисунке 2.1.

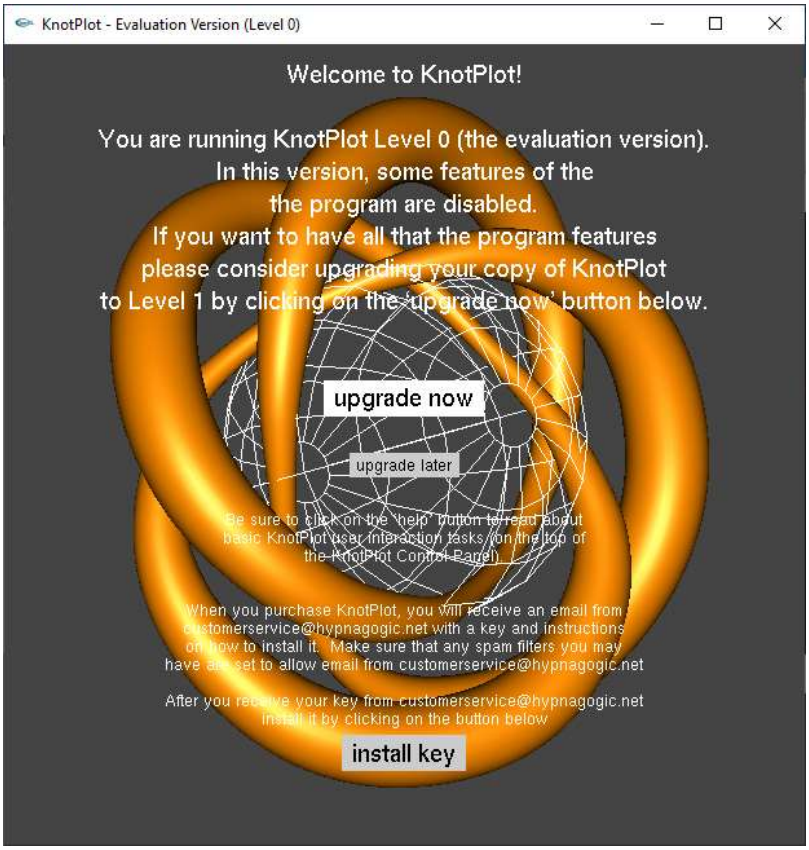


Рисунок 2.1. Пример визуализации узла в программе KnotPlot

Бесплатная пробная версия с ограниченными возможностями. Полная версия платная. Виртуальные узлы отсутствуют.

Узлы могут быть загружены из базы данных, содержащей более 3000 узлов и зацеплений, или могут быть нарисованы вручную в трех измерениях. Кроме того, узлы могут быть построены через код Конвея или с помощью калькулятора тэнглов. Ряд специальных типов узлов (торические узлы, цепочки узлов, узлы Лиссажу – torus knots, knot chains, Lissajous knots) могут создаваться на лету. Наконец, новые узлы могут быть созданы из старых узлов

с помощью различных инструментов преобразования. Использование всех этих функций позволяет создавать огромное число различных узлов.

Создание и сохранение картинок и анимаций. Также программа используется создателем, например, для выявления узлов с определенными свойствами (см. [6]), т.е. для формирования пользовательских баз данных.

2.1.2. LinKnot

Программа разработана S. Jablan, R. Sazdanovic, работает с помощью пакета Mathematica.

LinKnot (см. [8]) – программа, которая работает с узлами и зацеплениями, описанными с помощью нотации Конвея. В своей базе данных она содержит классические узлы и зацепления, которые имеют не более 12 перекрестков и виртуальных узлов, а также зацепления по нотации Конвея, образованные из узлов, имеющих не более 8 пересечений. Пример работы программы представлен на рисунке 2.2

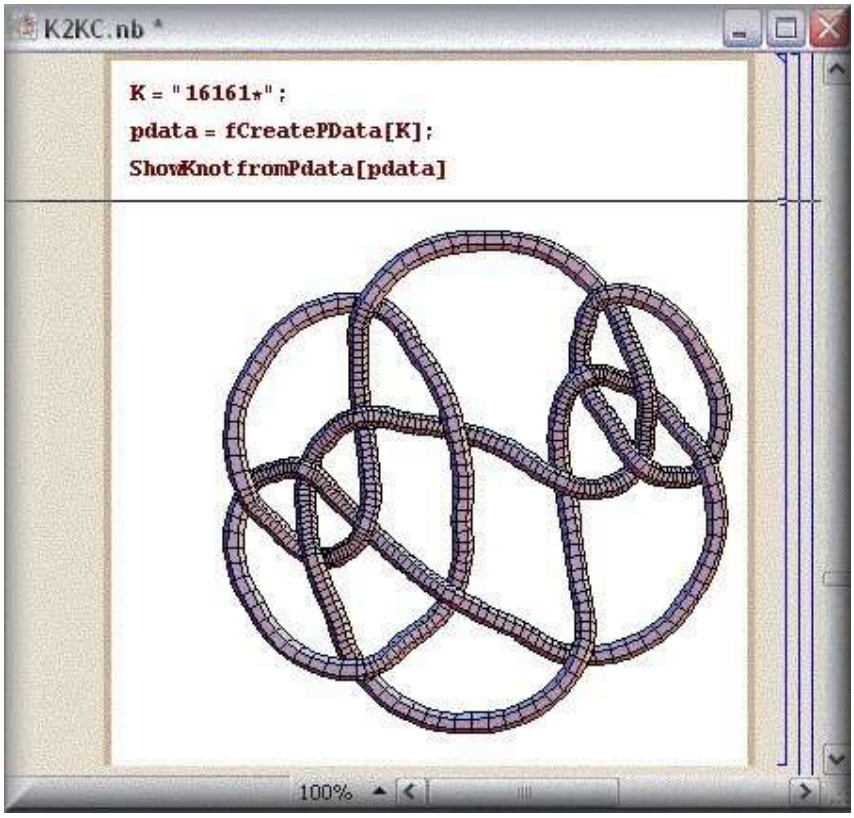


Рисунок 2.2. Пример работы программы LinKnot

2.1.2. Ming

Ming (см. [10]) – это программа, которая служит для нахождения локальных минимумов функции энергии полигональных узлов, уменьшения количества ребер и визуализации их трехмерных изображений. Написан с помощью таких языков программирования, как C ++, OpenInventor и Motif.

«Ming» – графическая версия более ранней программы «min». Она использует разработку компании Silicon Graphics – Open Inventor для рисования изображений узлов. Open Inventor стал одним из направлений развития OpenGL. В программе можно указать разные цвета и размеры для узла.

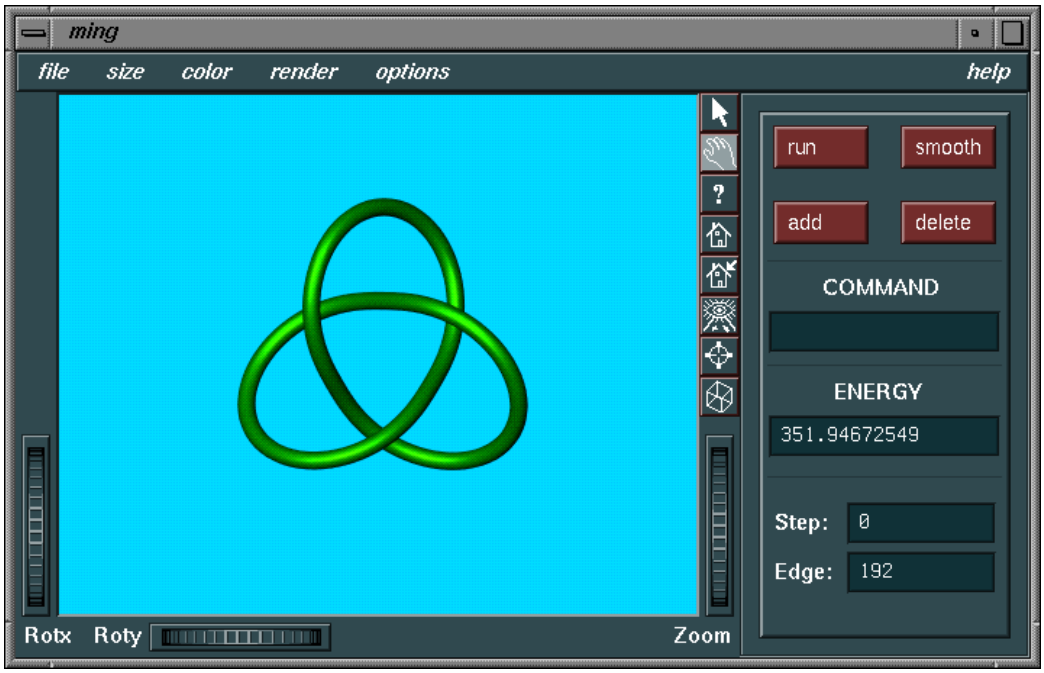


Рисунок 2.3. Пример работы программы Ming

2.3. Исследование молекулы белка с помощью теории узлов

Ученые, участвующие в проекте SPOCK (Scientific Properties of Complex Knots – Научные свойства сложных узлов) (см. [12]), из Университета Бристоля и Даремского университета, показали, что узлы, в которые завязываются нити аминокислот в молекулах белка, можно интерпретировать как «виртуальные узлы» – ими занимается один из подразделов теории узлов,

до некоторой поры считавшийся сугубо абстрактным и не имеющим практического применения. Успешное решение практических проблем в этом направлении позволит визуализировать молекулу белка в 3D.

Известно, что во множестве критически важных для жизни процессов участвуют белки. Это длинные молекулы, которые складываются в трехмерные фигуры и затем исполняют свои биологические функции. Поскольку они состоят из цепочек аминокислот, сложенная молекула белка напоминает моток проволоки, которая, как всем хорошо известно из практики, имеет свойства завязываться в узлы.

Итак, в молекуле белка есть узлы, теория узлов изучает узлы — трудность в том, что теория изучает замкнутые узлы, а в цепочках аминокислот концы свободны. Предложенный учеными из Бристольского университета метод предполагает рассмотрение белка с разных сторон, что можно математически обрабатывать как виртуальные узлы, без того, чтобы достраивать несуществующие линии. Это позволяет оценить имманентную амбивалентность кривых в молекуле белка.

Преимущество теории виртуальных узлов над «родительской» теорией узлов в том, что последняя требует, что все узлы были «закрытыми», что принуждает теоретически достраивать цепочки аминокислот и иметь дело с ограниченным набором узлов. Между тем виртуальные узлы могут быть и незамкнутыми, и это дает возможность исследовать форму молекулы в больших деталях.

Рассмотрение кривой белка с разных сторон дает различные проекции, или «тени». Применение теории виртуальных узлов к проекциям позволяет математическими методами представить строение узлов реальных. Результаты наносятся на сферу, где постепенно образуются своего рода моря и острова из узлов разных типов.

Цель проекта — разработать новые вычислительные инструменты и математические методы для анализа, синтеза и использования узловых структур в широком диапазоне физических феноменов.

2.4. Выводы по главе 2

Теория узлов набирает популярность с каждым годом. Сейчас существует немало различных баз данных по классическим и виртуальным узлам и зацеплениям. Также в свободном доступе имеются программы визуализирующие узлы и зацепления, но только классические.

До 2017 года теория узлов была сугубо абстрактной наукой не имеющей практического применения, но ученые нашли метод, который предполагает рассмотрение белка с разных сторон, что можно математически обрабатывать как виртуальные узлы.

ЗАКЛЮЧЕНИЕ

Данная работа посвящена разработке алгоритма визуализации частного случая виртуальных зацеплений – виртуальных узлов рода 1 малой сложности, а также – определению вектора развития теории узлов и применение ее в других науках.

В первой главе был разработан алгоритм, визуализирующий виртуальные узлы в утолщенном торе, сложность (число перекрестков на диаграмме) которых не превышает 3. В алгоритме было использовано:

1) параметрическое уравнение тора и окружности;
2) интерполяция кубическими сплайнами;

Во второй главе сделан обзор на существующие базы данных по теории узлов и программы, визуализирующие в двумерном или трехмерном пространстве классические и виртуальные зацепления и узлы. А также рассмотрен проект, который предполагает изучение молекул белка с помощью методов теории узлов.

В результате работы, была реализована программа для визуализации виртуальных узлов рода 1, сложность которых не превышает 3, с возможностью рассматривать полученные узлы с разных ракурсов для более простого изучения их структуры.

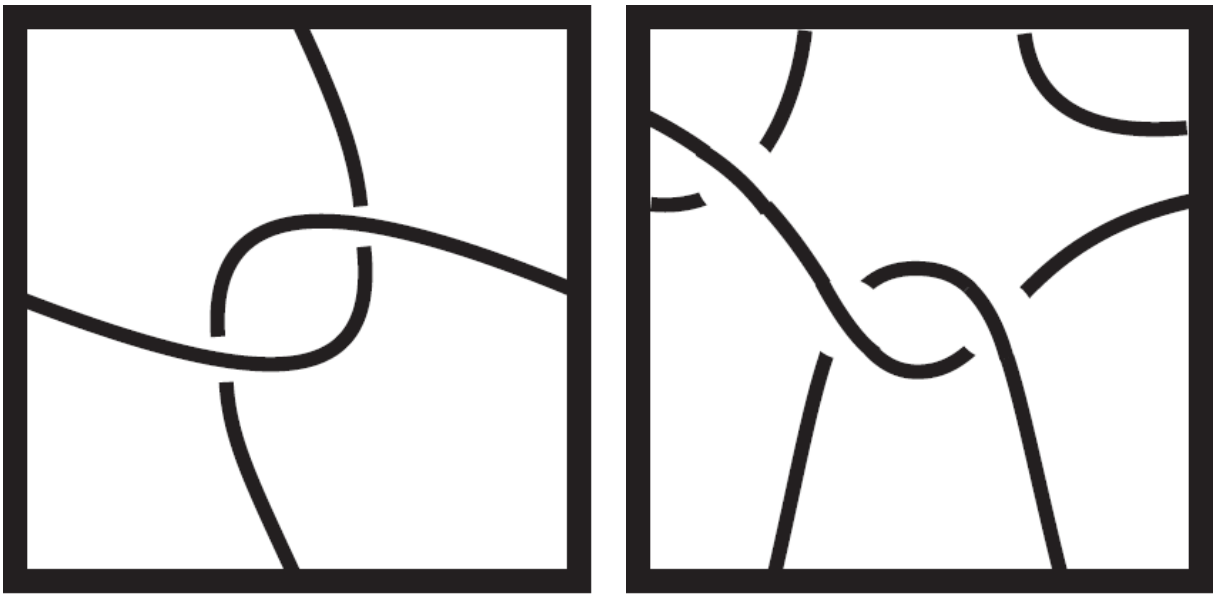
К тому же был сделан вывод, что популярность теории узлов растет с каждым годов, что подтверждает появление нового и совершенствование старого программного обеспечения, позволяющего более подробно изучать классические и виртуальные узлы. А также изучен проект, который способен при помощи методов из теории узлов рассматривать молекулы белка с разных сторон, которые в таком случае можно обрабатывать математически как виртуальные узлы.

Таким образом, все поставленные задачи полностью решены и цель работы достигнута.

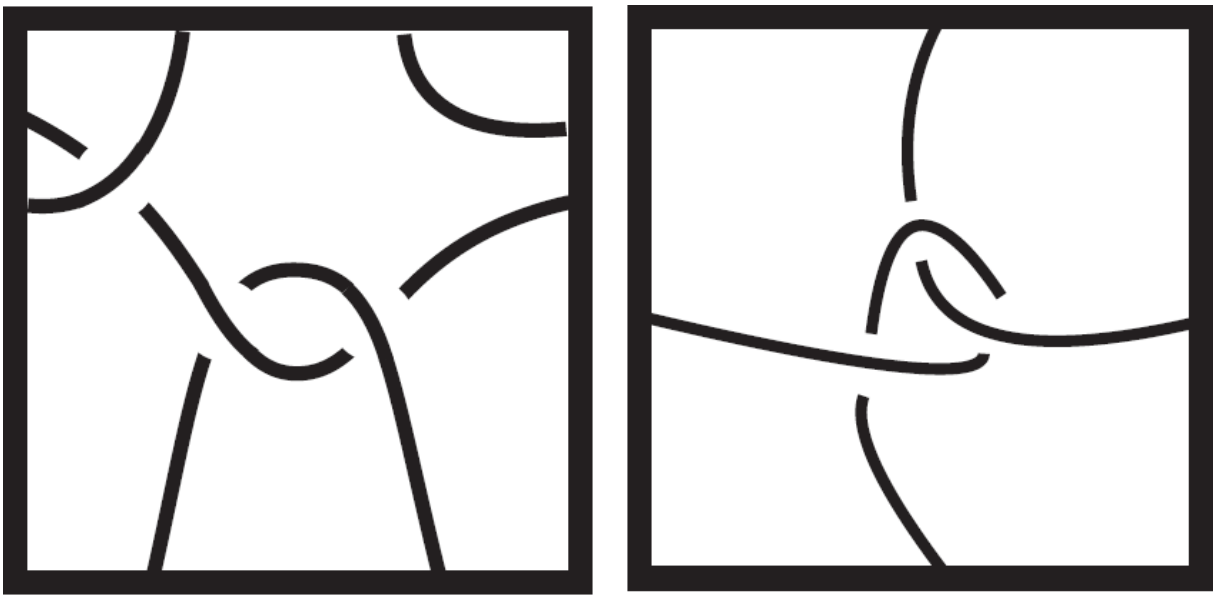
СПИСОК ЛИТЕРАТУРЫ

1. Акимова, А.А. Классификация узлов малой сложности в утолщенном торе / А.А. Акимова, С.В. Матвеев // Вестник НГУ. Серия «Математика. Механика. Информатика». – 2012. – Т.12. №.3. – С. 10-21.
2. Дьяконов В.П. /MATLAB. Полный самоучитель. – М.: ДМК Пресс, 2012.
3. Никулин Е.А. Компьютерная геометрия и алгоритмы машинной графики. – СПб.: БХВ-Петербург, 2003.
4. KnotInfo, доступ: http://www.indiana.edu/~knotinfo/ (Дата обращения – 10 июня 2019 года).
5. Knot Atlas, доступ: http://katlas.org/wiki/Main_Page (Дата обращения – 8 июня 2019 года).
6. Knot Theory with KnotPlot, доступ: http://www.colab.sfu.ca/KnotPlot/ktheory.html (Дата обращения – 18 мая 2019 года).
7. LinkInfo, доступ: http://www.indiana.edu/~linkinfo/ (Дата обращения – 10 июня 2019 года).
8. LinKnot, доступ: https://www.mi.sanu.ac.rs/vismath/linknot/index.html (Дата обращения – 18 мая 2019 года).
9. Lukas Lewark – Knot Software, доступ: http://lewark.de/lukas/software.html (Дата обращения – 14 июня 2019 года).
10. Ming – Knot energy minimizer, доступ: http://homepage.divms.uiowa.edu/~wu/ming/ming.html (Дата обращения – 7 июня 2019 года)
11. Rob Scharein’s Home Page, доступ: https://hypnagogic.net/rob/ (Дата обращения – 18 мая 2019 года).
12. SPOCK (Scientific Properties of Complex Knots), доступ: http://www.maths.dur.ac.uk/spock/index.html/ (Дата обращения – 7 июня 2019 года).
13. The KnotPlot Site, доступ: https://knotplot.com (Дата обращения – 18 мая 2019 года)

ПРИЛОЖЕНИЕ А



2_1 3_1



3_2 3_3

Рисунок А.1. Диаграммы виртуальных узлов в утолщенном торе $T \times I$, имеющие $n \leq 3$ перекрестков. Тор T представлен в виде квадрата с отождествленными сторонами

Таблица А.1. Коды Гаусса для виртуальных узлов в утолщенном торе $T \times I$, имеющие $n \leq 3$ перекрестков.

Узел 2_1: 2 -1 t1 b1 -2 1 r1 l1
Узел 3_1: -1 t1 b2 -2 3 b1 t2 r1 1 2 -3 r2 l1
Узел 3_2: 1 t1 b2 -2 3 b1 t2 r1 -1 2 -3 r2 l1
Узел 3_3: 1 -2 3 -1 b1 t1 -3 2 r1 l1

ПРИЛОЖЕНИЕ Б

Блок-схема программы

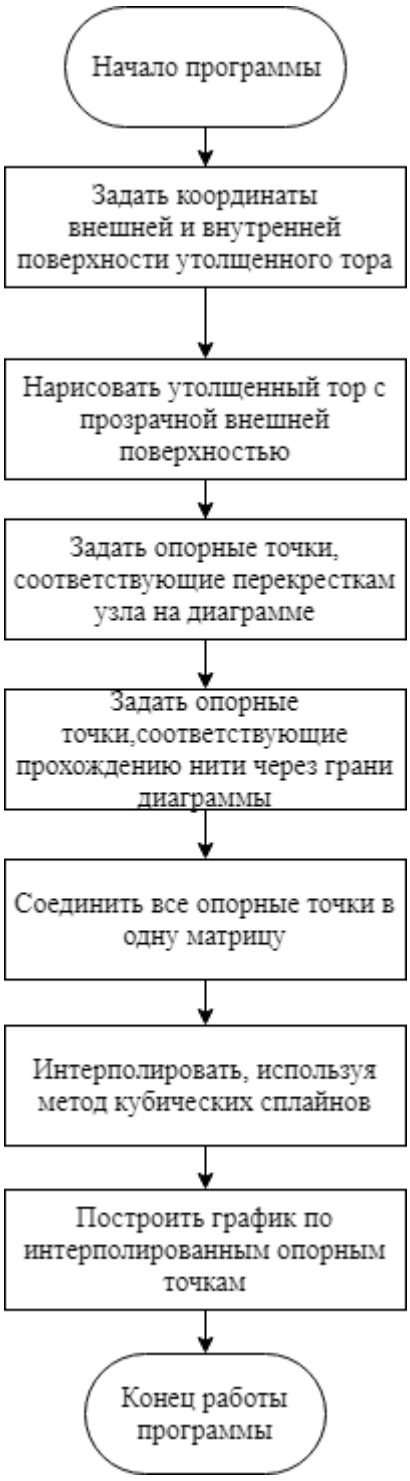


Рисунок Б.1. Алгоритм работы программы

ПРИЛОЖЕНИЕ В

Результат работы программы

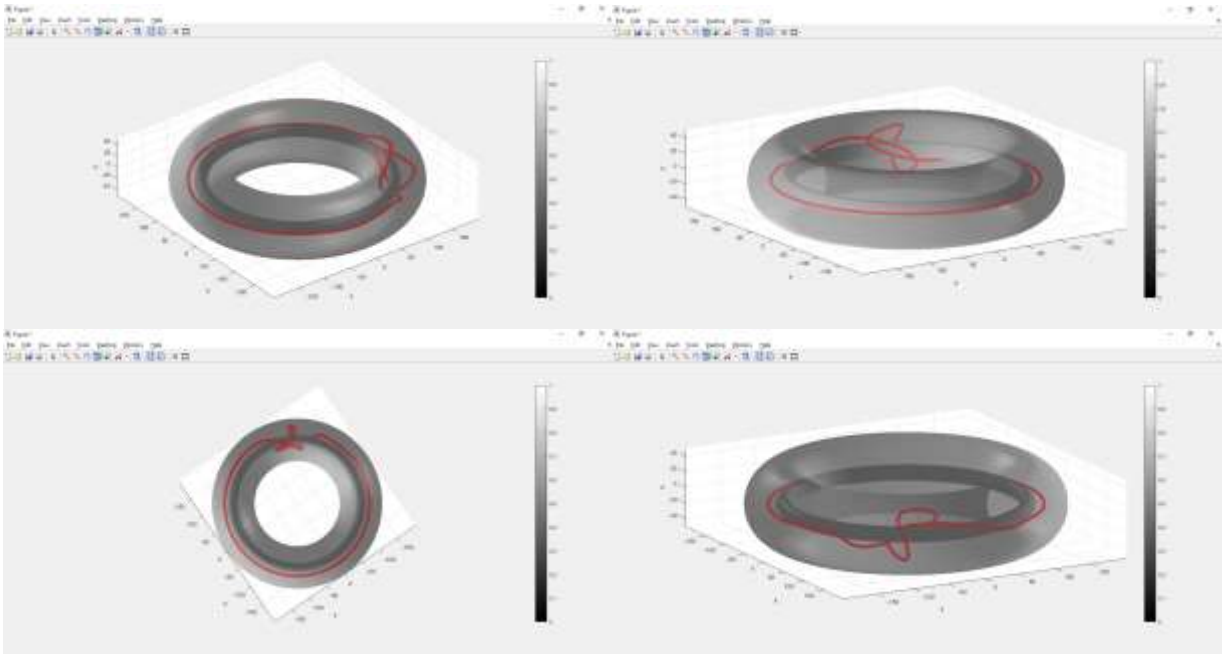


Рисунок В.1. Визуализированный узел 2_1

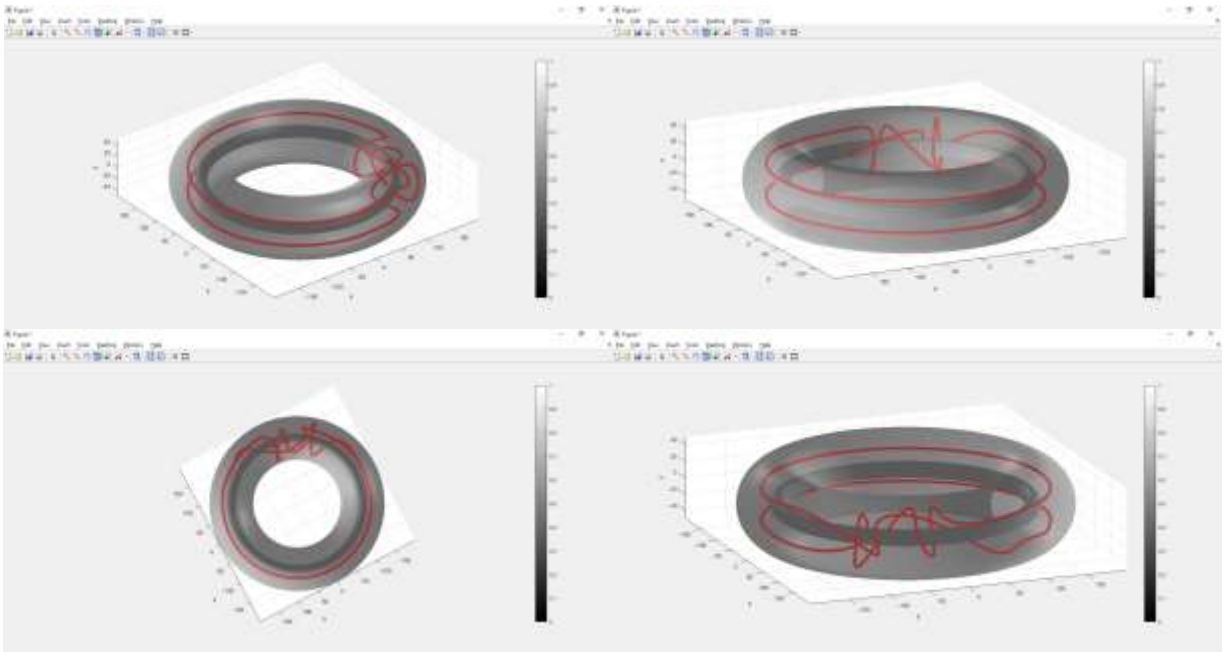


Рисунок В.2. Визуализированный узел 3_1

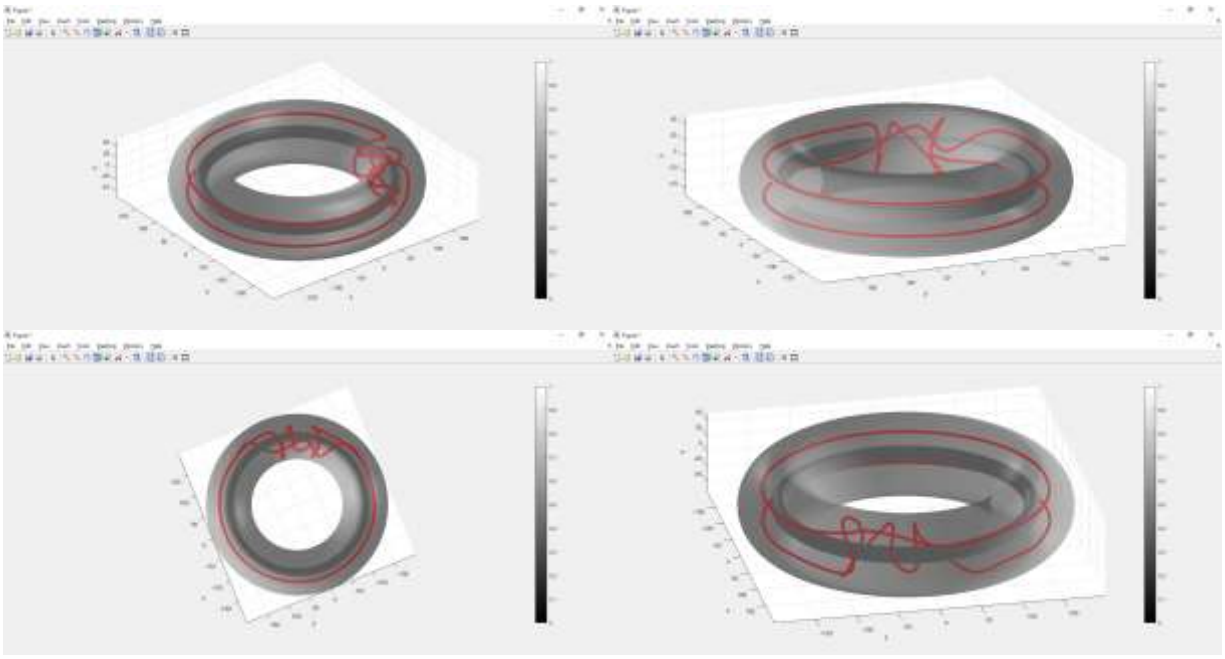


Рисунок В.1. Визуализированный узел 2_1

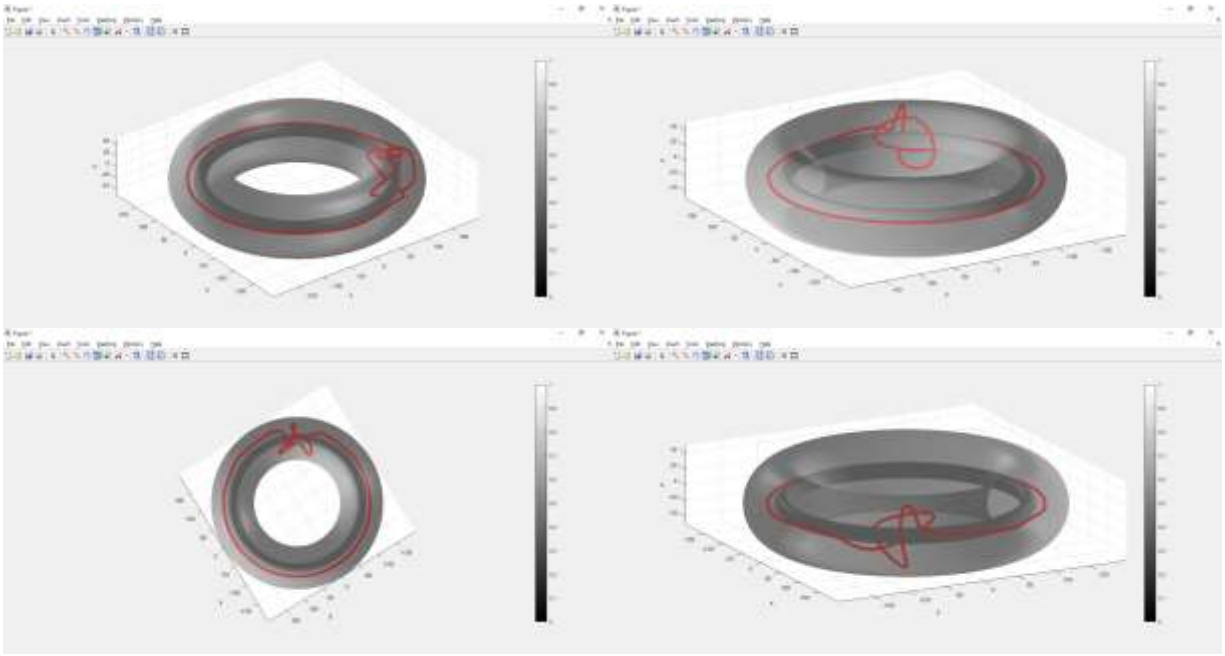


Рисунок В.2. Визуализированный узел 3_1

ПРИЛОЖЕНИЕ Г

Код программы

2.1.m

```
% Прорисовка поверхности утолщенного тора
aminor = 50.; % радиус образующей окружности внешней поверхности тора
aminorInt = 10.; % внутренней
Rmajor = 150.; % радиус вращения образующей окружности внешней поверхности тора
RmajorInt = 150.; % внутренней
theta  = linspace(pi, -pi, 64);
phi    = linspace(0, 2.*pi, 64);
thetaInt  = linspace(pi, -pi, 64);
phiInt    = linspace(0, 2.*pi, 64);
[t, p] = meshgrid(phi, theta);
[tInt, pInt] = meshgrid(phiInt, thetaInt);

% массивы с координатами внешней поверхности тора
x = (Rmajor + aminor.*cos(p)) .* cos(t);
y = (Rmajor + aminor.*cos(p)) .* sin(t);
z = aminor.*sin(p);

% массивы с координатами внутренней поверхности тора
xInt = (RmajorInt + aminorInt.*cos(pInt)) .* cos(tInt);
yInt = (RmajorInt + aminorInt.*cos(pInt)) .* sin(tInt);
zInt = aminorInt.*sin(pInt);

% прорисовка внешней поверхности тора
Tor = surfl(x,y,z);
colormap(gray)
shading interp
colorbar
alpha(Tor, 0.6);
hold on

% прорисовка внутренней поверхности тора
TorInt = surfl(xInt, yInt, zInt);
shading interp
alpha(TorInt, 1);

% координаты перекрестков
FT = [110; -110; -15]; % первая буква – номер перекрестка
FB = [100; -80; -5]; % вторая буква – снизу или сверху проходит нить
ST = [160; -30; 15];
SB = [130; -30; 15];

% Координаты опорных точек при проходе нити через нижнюю и верхнюю грань диаграммы
SCtopTB = [135 160 135; -80 -90 -80; 30 0 -30];
% Координаты опорных точек при проходе нити через левую и правую грань диаграммы
thetaC = linspace(2*pi/6, 2* pi*5/6, 10);
thetaCEnd = linspace(2*pi/24, 2*pi/6, 4);
rC = 170;
zC = [0 0 0 0 0 0 0 0 0 0];
zCEnd = [0 0 0 0];

% Формирование матрицы опорных точек нити
pointsKnotEnd = rC*[cos(thetaCEnd); sin(thetaCEnd); zCEnd];
pointsKnot = rC*[cos(thetaC); sin(thetaC); zC];
pointsKnot(:,end+1) = FT;
pointsKnot(:,end+1) = SB;
pointsKnot = [pointsKnot, SCtopTB];
pointsKnot(:,end+1) = FB;
```

```
pointsKnot(:,end+1) = ST;
pointsKnot = [pointsKnot, pointsKnotEnd];

% Интерполяция
tK = 1:length(pointsKnot(1,:)); % Исходная сетка (индексы массивов)
tiK = 1:0.01:length(pointsKnot(1,:)); % Новая мелкая сетка
xiK = spline(tK,pointsKnot(1,:),tiK);
yiK = spline(tK,pointsKnot(2,:),tiK);
ziK = spline(tK,pointsKnot(3,:),tiK);

% подпись координатных осей
xlabel('x')
ylabel('y')
zlabel('z')

% Прорисовка нити по массивам интерполированных опорных точек
Knot = plot3(xiK, yiK, ziK, 'r', 'LineWidth', 5);
hold off
axis equal
```

3.1.m

```
% Прорисовка поверхности утолщенного тора
aminor = 50.; % радиус образующей окружности внешней поверхности тора
aminorInt = 10.; % внутренней
Rmajor = 150.; % радиус вращения образующей окружности внешней поверхности тора
RmajorInt = 150.; % внутренней
theta  = linspace(pi, -pi, 64);
phi    = linspace(0, 2.*pi, 64);
thetaInt  = linspace(pi, -pi, 64);
phiInt    = linspace(0, 2.*pi, 64);
[t, p] = meshgrid(phi, theta);
[tInt, pInt] = meshgrid(phiInt, thetaInt);

% массивы с координатами внешней поверхности тора
x = (Rmajor + aminor.*cos(p)) .* cos(t);
y = (Rmajor + aminor.*cos(p)) .* sin(t);
z = aminor.*sin(p);

% массивы с координатами внутренней поверхности тора
xInt = (RmajorInt + aminorInt.*cos(pInt)) .* cos(tInt);
yInt = (RmajorInt + aminorInt.*cos(pInt)) .* sin(tInt);
zInt = aminorInt.*sin(pInt);

% прорисовка внешней поверхности тора
Tor = surfl(x,y,z);
colormap(gray)
shading interp
colorbar
alpha(Tor, 0.6);
hold on

% прорисовка внутренней поверхности тора
TorInt = surfl(xInt, yInt, zInt);
shading interp
alpha(TorInt, 1);

% координаты перекрестков
FT = [145; -95; -15]; % первая буква – номер перекрестка
FB = [80; -85; -15]; % вторая буква – снизу или сверху проходит нить
ST = [130; -90; 15];
SB = [110; -45; 25];
```

```
TT = [165; -50; 10];
TB = [120; -25; 15];

% Координаты опорных точек при проходе нити через нижнюю и верхнюю грань диаграммы
SCtopTB1 = [125 145 125; -105 -120 -105; 30 0 -30];
SCbotBT2 = [125 105 125; -15 -10 -15; -30 0 30];
% Координаты опорных точек при проходе нити через левую и правую грань диаграммы
thetaC1 = linspace(0, 2* pi*5/6, 10);
rC1 = 170;
zC1 = [20/170 20/170 20/170 20/170 20/170 20/170 20/170 20/170 20/170 20/170];
thetaC2 = linspace(2*pi/6, 2* pi*5/6, 10);
thetaC2End = linspace(2*pi/24, 2*pi/6, 4);
rC2 = 170;
zC2 = [-20/170 -20/170 -20/170 -20/170 -20/170 -20/170 -20/170 -20/170 -20/170 -20/170];
zC2End = [-20/170 -20/170 -20/170 -20/170];

% Формирование матрицы опорных точек нити
pointsKnotBC1 = rC1*[cos(thetaC1); sin(thetaC1); zC1];
pointsKnotBC2End = rC2*[cos(thetaC2End); sin(thetaC2End); zC2End];
pointsKnot = rC2*[cos(thetaC2); sin(thetaC2); zC2];
pointsKnot(:,end+1) = FB;
pointsKnot = [pointsKnot, SCtopTB1];
pointsKnot(:,end+1) = SB;
pointsKnot(:,end+1) = TT;
pointsKnot = [pointsKnot, SCbotBT2];
pointsKnot = [pointsKnot, pointsKnotBC1];
pointsKnot(:,end+1) = FT;
pointsKnot(:,end+1) = ST;
pointsKnot(:,end+1) = TB;
pointsKnot = [pointsKnot, pointsKnotBC2End];

% Интерполяция
tK = 1:length(pointsKnot(1,:)); % Исходная сетка (индексы массивов)
tiK = 1:0.01:length(pointsKnot(1,:)); % Новая мелкая сетка
xiK = spline(tK,pointsKnot(1,:),tiK);
yiK = spline(tK,pointsKnot(2,:),tiK);
ziK = spline(tK,pointsKnot(3,:),tiK);

% подпись координатных осей
xlabel('x')
ylabel('y')
zlabel('z')

% Прорисовка нити по массивам интерполированных опорных точек
Knot = plot3(xiK, yiK, ziK, 'r', 'LineWidth', 5);
hold off
axis equal
```

3.2.m

```
% Прорисовка поверхности утолщенного тора
aminor = 50.; % радиус образующей окружности внешней поверхности тора
aminorInt = 10.; % внутренней
Rmajor = 150.; % радиус вращения образующей окружности внешней поверхности тора
RmajorInt = 150.; % внутренней
theta  = linspace(pi, -pi, 64);
phi    = linspace(0, 2.*pi, 64);
thetaInt  = linspace(pi, -pi, 64);
phiInt    = linspace(0, 2.*pi, 64);
[t, p] = meshgrid(phi, theta);
[tInt, pInt] = meshgrid(phiInt, thetaInt);
% массивы с координатами внешней поверхности тора
```

```
x = (Rmajor + aminor.*cos(p)) .* cos(t);
y = (Rmajor + aminor.*cos(p)) .* sin(t);
z = aminor.*sin(p);
% массивы с координатами внутренней поверхности тора
xInt = (RmajorInt + aminorInt.*cos(pInt)) .* cos(tInt);
yInt = (RmajorInt + aminorInt.*cos(pInt)) .* sin(tInt);
zInt = aminorInt.*sin(pInt);

% прорисовка внешней поверхности тора
Tor = surfl(x,y,z);
colormap(gray)
shading interp
colorbar
alpha(Tor, 0.6);
hold on

% прорисовка внутренней поверхности тора
TorInt = surfl(xInt, yInt, zInt);
shading interp
alpha(TorInt, 1);

% координаты перекрестков
FT = [145; -95; -15]; % первая буква – номер перекрестка
FB = [80; -85; 5]; % вторая буква – снизу или сверху проходит нить
ST = [125; -70; 25];
SB = [110; -45; 25];
TT = [165; -50; 10];
TB = [125; 0; 0];

% Координаты опорных точек при проходе нити через нижнюю и верхнюю грань диаграммы
SCbotTB1 = [115 85 115; -100 -70 -100; 35 0 -35];
SCbotBT2 = [125 105 125; -15 -10 -15; -30 0 30];
% Координаты опорных точек при проходе нити через левую и правую грань диаграммы
thetaC1 = linspace(0, 2* pi*5/6, 10);
rC1 = 170;
zC1 = [20/170 20/170 20/170 20/170 20/170 20/170 20/170 20/170 20/170 20/170];
thetaC2 = linspace(2*pi/6, 2* pi*5/6, 10);
thetaC2End = linspace(2*pi/24, 2*pi/6, 4);
rC2 = 170;
zC2 = [-20/170 -20/170 -20/170 -20/170 -20/170 -20/170 -20/170 -20/170 -20/170 -20/170];
zC2End = [-20/170 -20/170 -20/170 -20/170];

% Формирование матрицы опорных точек нити
pointsKnotBC1 = rC1*[cos(thetaC1); sin(thetaC1); zC1];
pointsKnotBC2End = rC2*[cos(thetaC2End); sin(thetaC2End); zC2End];
pointsKnot = rC2*[cos(thetaC2); sin(thetaC2); zC2];
pointsKnot(:,end+1) = FT;
pointsKnot = [pointsKnot, SCbotTB1];
pointsKnot(:,end+1) = SB;
pointsKnot(:,end+1) = TT;
pointsKnot = [pointsKnot, SCbotBT2];
pointsKnot = [pointsKnot, pointsKnotBC1];
pointsKnot(:,end+1) = FB;
pointsKnot(:,end+1) = ST;
pointsKnot(:,end+1) = TB;
pointsKnot = [pointsKnot, pointsKnotBC2End];
% Интерполяция
tK = 1:length(pointsKnot(1,:)); % Исходная сетка (индексы массивов)
tiK = 1:0.01:length(pointsKnot(1,:)); % Новая мелкая сетка
xiK = spline(tK,pointsKnot(1,:),tiK);
yiK = spline(tK,pointsKnot(2,:),tiK);
ziK = spline(tK,pointsKnot(3,:),tiK);
```

```
% подпись координатных осей
xlabel('x')
ylabel('y')
zlabel('z')

% Прорисовка нити по массивам интерполированных опорных точек
Knot = plot3(xiK, yiK, ziK, 'r', 'LineWidth', 5);
hold off
axis equal
```

3.3.m

```
% Прорисовка поверхности утолщенного тора
aminor = 50.; % радиус образующей окружности внешней поверхности тора
aminorInt = 10.; % внутренней
Rmajor = 150.; % радиус вращения образующей окружности внешней поверхности тора
RmajorInt = 150.; % внутренней
theta  = linspace(pi, -pi, 64);
phi    = linspace(0, 2.*pi, 64);
thetaInt  = linspace(pi, -pi, 64);
phiInt    = linspace(0, 2.*pi, 64);
[t, p] = meshgrid(phi, theta);
[tInt, pInt] = meshgrid(phiInt, thetaInt);

% массивы с координатами внешней поверхности тора
x = (Rmajor + aminor.*cos(p)) .* cos(t);
y = (Rmajor + aminor.*cos(p)) .* sin(t);
z = aminor.*sin(p);

% массивы с координатами внутренней поверхности тора
xInt = (RmajorInt + aminorInt.*cos(pInt)) .* cos(tInt);
yInt = (RmajorInt + aminorInt.*cos(pInt)) .* sin(tInt);
zInt = aminorInt.*sin(pInt);

% прорисовка внешней поверхности тора
Tor = surfl(x,y,z);
colormap(gray)
shading interp
colorbar
alpha(Tor, 0.6);
hold on
% прорисовка внутренней поверхности тора
TorInt = surfl(xInt, yInt, zInt);
shading interp
alpha(TorInt, 1);

% координаты перекрестков
FT = [120; -100; -15]; % первая буква – номер перекрестка
FB = [80; -85; -15]; % вторая буква – снизу или сверху проходит нить
ST = [165; -50; 10];
SB = [120; -25; 15];
TT = [130; -90; 25];
TB = [120; -50; 20];

% Координаты опорных точек при проходе нити через нижнюю и верхнюю грань диаграммы
SCtopBT = [145 160 145; -80 -90 -80; -35 0 40];
% Координаты опорных точек при проходе нити через левую и правую грань диаграммы
thetaC = linspace(2*pi/6, 2* pi*5/6, 10);
thetaCEnd = linspace(2*pi/24, 2*pi/6, 4);
rC = 170;
zC = [0 0 0 0 0 0 0 0 0 0];
zCEnd = [0 0 0 0];
```

```
% Формирование матрицы опорных точек нити
pointsKnotEnd = rC*[cos(thetaCEnd); sin(thetaCEnd); zCEnd];
pointsKnot = rC*[cos(thetaC); sin(thetaC); zC];
pointsKnot(:,end+1) = FT;
pointsKnot(:,end+1) = SB;
pointsKnot(:,end+1) = TT;
pointsKnot(:,end+1) = FB;
pointsKnot = [pointsKnot, SCtopBT];
pointsKnot(:,end+1) = TB;
pointsKnot(:,end+1) = ST;
pointsKnot = [pointsKnot, pointsKnotEnd];

% Интерполяция
tK = 1:length(pointsKnot(1,:)); % Исходная сетка (индексы массивов)
tiK = 1:0.01:length(pointsKnot(1,:)); % Новая мелкая сетка
xiK = spline(tK,pointsKnot(1,:),tiK);
yiK = spline(tK,pointsKnot(2,:),tiK);
ziK = spline(tK,pointsKnot(3,:),tiK);

% подпись координатных осей
xlabel('x')
ylabel('y')
zlabel('z')

% Прорисовка нити по массивам интерполированных опорных точек
Knot = plot3(xiK, yiK, ziK, 'r', 'LineWidth', 5);
hold off
axis equal
```