Российская Академия Наук
Сибирское Отделение
Институт Систем Информатики

На правах рукописи

ЕРШОВ Алексей Геннадьевич

# Алгоритмы и программные системы для геометрических задач параметрического проектирования

Специальность 05.13.11 –
Математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

Диссертационная работа на соискание ученой степени
кандидата физико-математических наук

Научный руководитель
к. ф.-м. н. Ушаков Д. М.

Новосибирск 2007

ВВЕДЕНИЕ ..... 4
1. Задачи параметрического проектирования ..... 12
1.1 Вводные определения ..... 12
1.2 Формальное определение задачи параметрического проектирования ..... 16
1.3 Степени свободы в задачах параметрического проектирования ..... 19
1.4 Обзор существующих методов решения ..... 28
1.4.1 Классификация направлений исследования ..... 28
1.4.2 Методы алгебраического моделирования ..... 30
1.4.3 Методы геометрической декомпозиции ..... 32
1.4.4 Методы искусственного интеллекта ..... 34
1.4.5 Методы решения систем нелинейных уравнений ..... 36
1.5 Выводы ..... 37
2 Алгебраическое моделирование задач параметрического проектирования ..... 40
2.1 Постановка задачи ..... 40
2.2 Метод остовного моделирования ..... 43
2.2.1 Геометрические основы метода ..... 43
2.2.2 Алгоритмическое описание метода остовного моделирования ..... 56
2.3 Схема использования метода базисов Гребнера ..... 62
2.4 Выводы ..... 67
3. Методы геометрической декомпозиции задач параметрического проектирования ..... 69
3.1 Начальные определения ..... 69
3.2 Критика известных методов декомпозиции ..... 73
3.3 Описание метода отделяющей декомпозиции ..... 79
3.4 Алгоритм отделяющей декомпозиции и его временная сложность ..... 85
3.5 Расширения метода отделяющей декомпозиции ..... 89
3.6 Свойства метода отделяющей декомпозиции ..... 92
3.7 Выводы ..... 94
4. Интервальные методы в задачах параметрического проектирования ..... 96
4.1 Постановка задачи ..... 96
4.2 Интервальная математическая библиотека ..... 99
4.2.1 Выбор подхода к вычислению элементарных функций ..... 99
4.2.2 Минимизация погрешности вычислений на каноническом интервале ..... 102
4.2.3 Минимизация погрешности вычислений при приведении аргумента ..... 107
4.2.4 Другие вычислительные аспекты ..... 110
4.3 Интервальные методы решения задач в ограничениях ..... 113
4.3.1 Описание метода CP распространения ограничений. ..... 113
4.3.2 Описание интервального метода бисекции. ..... 117
4.3.3 Применение для задач параметрического проектирования ..... 121
4.4 Выводы ..... 124
5. Аспекты программной реализации и практического использования ..... 125
5.1 Реализованные программные компоненты ..... 125
5.2 Архитектура решателей LGS 2D и 3D ..... 126
5.2.1 Объекты интерфейса и их трансляция во внутреннее представление ..... 126
5.2.2 Схема использования эвристических вычислительных ветвей. ..... 128
5.2.3 Механизм кластерных типов ..... 130
5.2.4 Функциональность решателей LGS ..... 132
5.3 Решатель систем нелинейных уравнений LGSEP ..... 133
5.4 Аспекты программной реализации в LEMO ..... 135
5.5 Конечно-пользовательские приложения решателей LGS ..... 136
5.5.1 Клиент-серверная технология FlashLGS и веб-приложение на ее основе ..... 136

5.5.2 Использование LGS 3D как вычислительного ядра САПР ADEM-VX ..... 138
5.5.3 Интеграция с САПР bCAD ..... 138
5.5.4 Другие примеры интеграции ..... 139
5.6 Экспериментальные результаты реализации предложенных методов ..... 141
5.6.1 Эффект от использования остовного моделирования ..... 141
5.6.2 Результаты реализации метода отделяющей декомпозиции ..... 144
5.6.3 Характеристики библиотеки вычислений элементарных функций ..... 146
5.6.4 Оценка эффективности решателя LGS в сравнении с другим решателем ..... 148
Заключение ..... 150
Литература ..... 153
Приложения ..... 160
1. Количество степеней свободы, снимаемых разными ограничениями ..... 160
2. Список образцов отделяющей декомпозиции ..... 165
3. Пример использования интерфейса LGS 2D ..... 167

# ВВЕДЕНИЕ

Системы автоматизированного проектирования (далее – САПР) по праву считаются одной из наиболее значимых для промышленности областей информатики. На протяжении многих лет они являются ключевыми компонентами процесса разработки изделий на современном предприятии [6]. Именно САПР позволяют выпускать продукты лучшего качества при более низкой себестоимости и за меньшее время [17]. В последние десять-пятнадцать лет в функциональном развитии и масштабировании систем автоматизированного проектирования достигнут серьезный прогресс. Были спроектированы полностью автоматизированным способом такие сложные устройства, как самолет Boeing 787 Dreamliner, множество различных моделей автомобилей, которые представляют собой механизмы с десятками и сотнями тысяч элементов. Кроме того, системы автоматизации проектирования развиваются и вширь, объединяясь с системами инженерного анализа, автоматизации производства и управления данными в единый комплекс управления жизненным циклом изделия (Product Lifecycle Management, или PLM) [15]. Долгое и интенсивное развитие САПР позволяет считать исследования в области их математических основ постоянно актуальными и практически важными.

Особый интерес представляют системы, обладающие возможностями параметризации модели, которая является мощным инструментом создания новых и переиспользования готовых моделей. Параметризация позволяет конструировать модели с заранее заданными характеристиками или модифицировать существующую модель путем изменения ее параметров. Наличие параметризации специалисты считают отличительной характеристикой наиболее совершенных САПР [16].

Далее будет рассматриваться широко востребованный класс параметрических задач, состоящий из задач определения положений деталей

составной геометрической модели на плоскости или в пространстве. Для краткости такие задачи будут называться “задачи параметрического проектирования”. Существует два классических типа параметрического проектирования: более традиционный иерархический, основанный на истории построения модели, и более совершенный по предоставляемым возможностям вариационный [33]. В настоящее время значительно более распространены САПР на основе истории построения модели [13].

При иерархическом проектировании определяющее значение имеет порядок создания элементов, точнее, порядок их подчинения друг другу – иерархия, которая отражается в дереве построения модели. Элемент, для создания которого использовались любые части или характеристики другого элемента, считается подчиненным этому элементу. Любая такая параметрическая модель хранит историю своего создания и позволяет редактировать произвольный элемент посредством изменения параметров подчинения вышестоящему объекту. Пересчет положения и параметров измененного элемента при этом происходит по явным формулам [34].

Этот способ проще в реализации, но лишает пользователя многих возможностей, таких как размещение друг относительно друга двух объектов, уже вошедших в историю построения модели [14]. Это оказывается невозможным в силу образования циклов зависимостей между элементами и разрушения древовидной иерархической структуры. Если в модели имеются такие нетривиальные зависимости между элементами, то для ее описания нужен существенно другой подход [71]. Серьезным недостатком традиционного метода иерархического проектирования является и необходимость постоянно работать с полностью определенной геометрической моделью без внутренних степеней свободы, что ограничивает возможности ее редактирования [33].

На смену традиционному параметрическому подходу, основанному на истории построения модели, пришла концепция *вариационного*

*проектирования*, которая заключается в выражении отношений между элементами и определяющими их параметрами посредством вариационных связей [4], или *ограничений*. Ограничение – это формальное декларативное описание произвольной связи между объектами модели, не предполагающее подчинения одного объекта другому [92]. Обычно ограничения классифицируются на *логические геометрические* (параллельность прямых, касание плоскости и цилиндра), *параметрические геометрические* (длина отрезка, радиус сферы, угол между гранями) и *инженерные* (площадь замкнутого контура, равенство параметров двух ограничений). С помощью задания ограничений пользователь может полностью управлять формой проектируемого изделия, при этом ему не надо просчитывать взаимное расположение частей модели – САПР автоматически определяет положения всех объектов, которые удовлетворяют всем заданным пользователем ограничениям [31].

Главным преимуществом вариационного подхода по сравнению с подходом, основанным на истории построения, являются значительно более высокие выразительные возможности системы. Конструктору модели не нужно полностью продумывать пошаговое создание изделия, а лишь задать желаемые свойства [25]. В любой момент можно добавлять и удалять ограничения или менять значения их параметров, после чего получать новую модель с заданными свойствами. Более совершенный механизм описания геометрической модели позволяет иметь дело с циклическими зависимостями [14] и не полностью определенными моделями [33]. Однако для поддержки всех этих возможностей необходимо уметь решать системы наложенных ограничений, или так называемую *задачу вариационного проектирования*. Для этого разрабатываются специальные решатели задач вариационного проектирования, или *геометрические решатели*.

Среди существующих сейчас на рынке сотен САПР лишь немногие системы предоставляют возможности вариационного проектирования, а еще меньшее количество САПР имеют собственный решатель, поскольку

реализация этой программной компоненты весьма трудоемка и наукоемка. Тем не менее, все современные крупномасштабные САПР, такие как CATIA, NX, Pro/E, SolidWorks, Solid Edge, Inventor, основаны именно на концепции вариационного проектирования, поскольку с ее помощью можно разрабатывать существенно более сложные изделия [53, 60]. Проблемы, которые порождает одновременное решение системы ограничений, можно решить с помощью разработки мощных методов решения задач вариационного проектирования [25, 49], что и является целью диссертационного исследования.

В силу серьезной коммерческой востребованности геометрических решателей о них существует достаточно мало научных публикаций, а существующие публикации, обозначая некоторые алгоритмические идеи в целом, скрывают технологические особенности их воплощения. Создается ситуация “войны гигантов”: 3-5 крупнейших компаний в мире вкладывают значительные деньги в разработку собственных геометрических решателей и скупают независимые разработки в надежде переиграть в этом конкурентов, закрывая путь к свободному научному обсуждению данной проблематики. Тем самым, любые открытые публикации по тематике геометрических решателей обретают повышенную актуальность. Актуальность исследования подтверждается также ключевым значением геометрических вычислительных ядер для функциональной полноты, эффективности и масштабируемости САПР, полезность которых в современной промышленности, в частности, в автомобиле-, авиа-, приборостроении, не вызывает никаких сомнений.

К сожалению, при всем обилии публикаций о практических применениях САПР и разработках САПР для конкретных прикладных областей [22, 29], в русскоязычной литературе очень скудно освещается вопрос о фундаментальных математических методах обработки параметрических задач. Если рассмотреть даже небольшое количество существующих русскоязычных публикаций о математических методах

параметрического проектирования [20, 21, 26], среди них лишь считанное количество работ будет посвящены решению задач вариационного проектирования. Как правило, исследования посвящены иерархическому проектированию, задачам представления знаний [35] и различного рода гибридным методам для решения специальных классов задач. Обзор литературы позволяет сделать вывод, что в области общих задач вариационного проектирования нет сложившейся российской научной школы, есть лишь разрозненные исследования, недостаточно апробированные в промышленной практике. Тем не менее, все эти исследования выполнены недавно [8, 14], что позволяет надеяться на постепенное формирование этого направления как одного из значимых для отечественной науки в области САПР.

Объектами этого диссертационного исследования являются геометрические задачи параметрического проектирования и методы их решения. Термин “параметрическое проектирование” не вполне точен, так как охватывает и более широкую область иерархического проектирования, однако более корректный термин “вариационное проектирование” является значительно менее известным в русскоязычной литературе и имеет нежелательные ассоциации с вариационным исчислением. В иностранной литературе, как правило, вместо термина “геометрическая задача параметрического проектирования“ применяется оборот “geometric constraint problem” [58], дословно переводимый как “задача в геометрических ограничениях”. Тем не менее, этот термин не является достаточно устоявшимся, поэтому в данной работе автор будет придерживаться русскоязычной традиции [13].

**Цель диссертационной работы** – разработка математических методов и алгоритмов, а также создание эффективных программных средств, позволяющих решать геометрические задачи параметрического проектирования.

Задачи, направленные на достижение указанной цели, включают:

- Формальное описание задачи параметрического проектирования, изучение существующих подходов к их решению, выделение наиболее адекватных направлений исследования;
- Разработку методов алгебраического моделирования таких задач;
- Исследование вопроса о корректной декомпозиции задач параметрического проектирования и создание корректных методов декомпозиции;
- Изучение применимости интервальных методов для решения задач параметрического проектирования;
- Эффективную программную реализацию предложенных алгоритмов в рамках решателей задач параметрического проектирования.

**В первой главе** дана математическая формализация понятия “задача параметрического проектирования”, более строгая, чем в существующих исследованиях, и учитывающая характер задач, встречающихся на практике. Далее через понятие числа степеней свободы задачи параметрического проектирования дано определение недо-, пере-, и хорошо определенных задач, свободное от недостатков, встречающихся в других работах. Выполнен тематический обзор подходов к решению задач параметрического проектирования, включающий методы алгебраического моделирования, геометрической декомпозиции и методы искусственного интеллекта.

**Во второй главе** изложен новый метод алгебраического моделирования задач параметрического проектирования. Этот метод, называемый методом остовного моделирования, основан на параметрическом представлении положения одних геометрических объектов по отношению к положению других объектов. Предложенное внутреннее представление задачи позволяет автоматически учитывать некоторые ограничения, тем самым, сокращая размер генерируемых систем уравнений и значительно

увеличивая быстродействие решателя задач параметрического проектирования. Глава содержит описание математических основ метода, алгоритмических аспектов его реализации, оценку его вычислительной сложности. Кроме того, в главе рассмотрен вопрос о возможности использования метода базисов Гребнера в решателе задач параметрического проектирования и предложен способ применения этого метода для решения небольших аналитически неразрешимых подзадач.

**В третьей главе** проводится критика известных методов геометрической декомпозиции задач параметрического проектирования на основе разработанного математического аппарата, особое внимание уделено вопросу о корректности предлагаемых декомпозиций. После этого предлагается новый корректный метод отделяющей декомпозиции задач параметрического проектирования. Глава содержит описание идеи метода, алгоритмических аспектов его реализации, приводится псевдокод работы метода и оценка его вычислительной сложности. Также обсуждаются подходы к расширению области применимости метода, целесообразность комбинирования его с методом двусвязной декомпозиции.

**В четвертой главе** приведено описание методов вычисления интервальных элементарных математических функций, необходимых для применения интервальных методов при решении задачи параметрического проектирования. Предложенный подход основан на вычислениях с направленными округлениями и разложении функций в степенной ряд Чебышева, который имеет высокие характеристики аппроксимации функции на интервале, в том числе превосходящие традиционно используемый ряд Тейлора. Обсуждаются вопросы контроля и минимизации погрешности вычислений. Также исследован вопрос о целесообразности применения интервальных методов в рамках задач параметрического проектирования, в том числе для выявления несовместности таких задач.

**В пятой главе** описывается комплекс созданных при участии автора программных систем, их архитектурные особенности и другие аспекты программной реализации. Излагается используемая в решателях LGS 2D и 3D схема работы вычислительных ветвей и механизм классификации задач параметрического проектирования по кластерным типам. Приводятся данные об интеграции разработанных геометрических решателей в различные программные продукты в области САПР. Далее излагаются экспериментальные результаты, показывающие эффективность разработанных автором и описанных в предыдущих главах методов решения задач параметрического проектирования, а также приводятся результаты сравнения решателя LGS с известным на рынке коммерческим геометрическим решателем.

# 1. Задачи параметрического проектирования

## 1.1 Вводные определения

Основными сущностями задачи параметрического проектирования в $d$-мерном пространстве являются множество объектов $V$ и множество ограничений $E$ между ними [58].

Любой объект $v$ из $V$ является либо вещественной переменной, либо элементарным геометрическим $d$-мерным объектом. Множество переменных обозначается $V^*$, множество $d$-мерных геометрических объектов обозначается $V^d$. Любой геометрический объект характеризуется типом (точка, сфера, кривая и т.п.) и связанным с ним набором обобщенных координат. Так, точка характеризуется своими $d$ координатами в пространстве, цилиндр – $d$ координатами центра, $d$ компонентами направляющего вектора и радиусом, лоскут двухмерной поверхности – компонентами матрицы перехода $A$ и вектора сдвига $\bar{b}$ из глобальной системы координат в локальную, в которой он задан параметрическим представлением $\begin{pmatrix} x_1 = x_1(u,v) \\ .. \\ x_d = x_d(u,v) \end{pmatrix}$, $u,v \in [0,1]$. Задание координат центра и направляющего вектора можно рассматривать как часть описания перехода в локальную систему координат объекта. Следовательно, можно считать, что обобщенные координаты включают два вида параметров: координаты локальной системы координат (параметры матрицы перехода $A$ и вектора сдвига $\bar{b}$) и дополнительные вещественные параметры (например, радиус).

Матрица перехода $A$ и вектор сдвига $\bar{b}$ в локальную систему координат геометрического объекта в $d$-мерном пространстве задаются $C_2^d$ и $d$ параметрами соответственно [1]. При этом для некоторых объектов какие-то из $C_2^{d+1} = d(d+1)/2 = d(d-1)/2 + d = C_2^d + d$ параметров могут быть избыточными, например, для задания точки избыточны все $C_2^d$ параметров

матрицы перехода. Число дополнительных вещественных параметров объекта задается функцией $P: V^d \to \mathrm{N}$, которую можно обобщить: $P: V \to \mathrm{N}$, $v \in V^* \Rightarrow P(v) = 0$. Число обобщенных координат объекта тем самым можно выразить функцией $C: V \to \mathrm{N}$, причем

1) для $v \in V^*$, $C(v) = 1$; (1.1a)

2) для $v \in V^d$, $C(v) = P(v) + C_2^{d+1}$. (1.1b)

**Определение 1.1** *Означиванием* объекта $v \in V$, или *меткой*, называется пара $< v, s >$, где $s \in R^{C(v)}$ – некоторый конкретный набор значений обобщенных координат объекта. Означиванием множества объектов $\{v_1,..,v_n\} = V$ называется множество меток $S = \{< v_1, s_1 >,..,< v_n, s_n >\}$, где $v_i \neq v_j \Leftrightarrow i \neq j$, причем такое означивание будем отождествлять с функцией $S(v_i) = s_i$. Множество означиваний множества объектов $V$ будем обозначать $S(V)$.

Далее под движениями в $d$-мерном пространстве будем иметь в виду собственные движения в пространстве, как известно, они представляются в виде $t = (A, \bar{b})$, где $A$ – ортогональная $d \times d$ матрица с детерминантом, равным 1, а $\bar{b}$ – $d$-мерный вектор сдвига. Множество движений в $d$-мерном пространстве будем обозначать $t^d$.

**Определение 1.2** Действием движения $t = (A', \bar{b}') \in t^d$ на метку $< v^d, s >$, где $v^d \in V^d$, $s = (s_1,..,s_q, s_{q+1},..,s_{q+p})$, $q = C_2^{d+1}$, $p = P(v^d)$, причем $(s_1,..,s_q)$ – координаты локальной системы координат, будем называть метку $< v^d, t(s) >$, $t(s) = (t_1,..,t_q, s_{q+1},..,s_{q+p})$, где $(t_1,..,t_q) = t(s_1,..,s_q)$ – координаты локальной системы координат геометрического объекта после выполнения движения $t$. Если $u = (A(s_1,..,s_{q-d}), \bar{b}(s_{q-d+1},..,s_q))$ – матрица и вектор перехода в локальную систему координат $v^d$, то после выполнения $t$ они

будут равны $u' = (A'^{*}A(s_1,..,s_{q-d}), A'^{*}\bar{b}(s_{q-d+1},..,s_q) + \bar{b}')$. Также будем применять обозначение $t(< v^d, s >) =< v^d, t(s) >$.

Таким образом, действие движения на означивание геометрического объекта состоит только в изменении координат локальной системы координат объекта, но не в изменении его дополнительных параметров. Например, при движении сферы меняются координаты ее центра, но не радиус. Дополнительные параметры объектов, как и переменные, изменяются с помощью *приращений*.

**Определение 1.3** Действием приращения $\Delta \in R$ на метку $< v^*, s >$, $v^* \in V^*$ будем называть метку $< v^*, s + \Delta >$, и обозначать выражением $\Delta(< v^*, s >)$. Действием на метку $< v, s >$, где $v \in V^d$, $s = (s_1,..,s_q,s_{q+1},..,s_{q+p})$, $q = C_2^{d+1}$, $p = P(v)$ приращения $\Delta = (\Delta_1,..,\Delta_p) \in R^p$ будем называть метку $< v, \Delta(s) >$, где $\Delta(s) = (s_1,..,s_q,s_{q+1} + \Delta_1,..,s_{q+p} + \Delta_p)$ и обозначать выражением $\Delta(< v, s >)$.

Каждое ограничение $e \in E$ имеет определенный тип: угол между плоскостями, расстояние между точками, инцидентность сфер, симметрия прямых относительно плоскости, уравнение над вещественными переменными и т.п. Для каждого типа ограничения известна его *арность*, или число аргументов (объектов, которое оно связывает), тем самым задана функция $a : E \to \mathrm{N}$. Каждое конкретное ограничение задачи параметрического проектирования обладает конкретным набором аргументов, которыми являются объекты из $V$, их количество равно арности ограничения. Набор аргументов ограничения задается с помощью функции $A : E \times N \to V \cup \{\varnothing\}$, причем $A(e,i) = \varnothing \Leftrightarrow a(e) < i$. Ограничение как математический объект является отношением [32], то есть подмножеством декартова произведения наборов координат его аргументов: $e \subseteq R^{C(A(e,1))} \times .. \times R^{C(A(e,a(e)))}$.

**Определение 1.4** Решающим означиванием (решением) задачи параметрического проектирования $Z = (V,E)$ называется такое означивание

$S = \{< v_1, s_1 >,..,< v_n, s_n >\}$ всех объектов $V = \{v_1,..,v_n\}$, что все ограничения на этом означивании выполняются: $\forall e \in E, (S(A(e,1)),..,S(A(e,a(e)))) \in e$.

Такое определение является достаточно общим и встречается в большинстве литературных источников, однако, как правило, в неформальном, менее детальном виде, без определения означиваний объектов и действия на них движений и приращений. Тем не менее, и приведенное выше определение не учитывает нескольких свойств задач параметрического проектирования, встречающихся на практике.

При реальном проектировании для представления одной детали требуется множество простейших геометрических объектов. Вследствие этого разумно использовать понятие "*твердое тело"* (далее – *тело*), которое является множеством простейших геометрических объектов, которые при любых движения перемещаются одинаково. Таким образом, еще одним элементом задачи параметрического проектирования является набор тел – взаимно непересекающихся множеств объектов.

При обработке задачи параметрического проектирования на ЭВМ в силу приближенного представления вещественных чисел используется некоторая желаемая точность вычислений $\varepsilon$. Ограничения задачи задаются не как отношения, а как вещественнозначные функции $e : (R^{C(A(e,1))} \times .. \times R^{C(A(e,a(e)))}) \to R$, определяющие невязку "идеальных" ограничений. Ограничения считаются удовлетворенными, если невязка не превышает $\varepsilon$.

Задача параметрического проектирования может иметь несколько решений, которые обладают разной ценностью с точки зрения пользователя. Как правило, пользователь нуждается в том решении, которое наименее всего отличается от *эксиза* – заданного им начального расположения объектов. Это *требование оптимальности* можно учесть с помощью задания *начального означивания* объектов $S_0$ и положительно определенной вещественнозначной

*функции близости* означиваний $F(S_1,S_2)$. Функция $F(S_1,S_2)$, используемая для измерения близости между означиваниями, хоть и похожа по смыслу на метрику, тем не менее, не является метрикой в некоторых практически важных случаях. Например, если она задана как количество координат, для которых разность в означиваниях $S_1,S_2$ не превышает точности вычислений $\varepsilon$, то из $F(S_1,S_2)=0$ и $F(S_2,S_3)=0$ не следует, что $F(S_1,S_3)=0$. В то же время, если $\varepsilon=0$, то так заданная функция $F(S_1,S_2)$ окажется метрикой. Для многих других практически значимых способов задания функции $F$ имеет место такая же ситуация: при $\varepsilon=0$ $F$ является метрикой, при $\varepsilon>0$ нет.

## 1.2 Формальное определение задачи параметрического проектирования

**Определение 1.5** Общая постановка (на ЭВМ) задачи параметрического проектирования в $d$-мерном пространстве есть упорядоченная шестерка $Z=(\tilde{V},\tilde{E},G,S_0,F,\varepsilon)$, где

1) $\tilde{V}=(V,P)$ – *структура объектов*, состоящая из множества объектов $V=V^d\cup V*$, где $V^d$ – множество $n$ $d$-мерных геометрических объектов, $V*$ – множество $q$ вещественных переменных, и *функции числа дополнительных параметров* $P:V^d\to N$. Через $P:V^d\to N$ определяются *общее число дополнительных параметров* $p=\sum_v P(v)$ и *функция числа обобщенных координат* $C:V\to \mathrm{N}$ по формулам (1.1a) и (1.1b);

2) $\tilde{E}=(E,a,A)$ – *структура ограничений*, состоящая из $E\subset\bigcup_{n\in N}R^n\times R$ – множества $m$ ограничений над объектами из $V$, *функции арности* $a:E\to \mathrm{N}$ и *функции аргументов* $A:E\times N\to V\cup\{\varnothing\}$. Каждое ограничение $e\in E$ является веществennозначной функцией, вычисляющей невязку: $e:(R^{C(A(e,1))}\times..\times R^{C(A(e,a(e)))})\to[0,\infty)$;

3) $G = \{g_1,..,g_k\}$ – множество $k$ тел, $\forall i, g_i \subseteq V^d$ и $\bigcup_{i\in\{1,..,k\}} g_i = V^d$, причем $i \neq j \Leftrightarrow g_i \bigcap g_j = \varnothing$. Из этих свойств следует, что $\forall v \in V^d \exists! g \in G : v \in g$, и можно ввести функцию $\overline{G} : V \to G \cup \{\varnothing\}$, сопоставляющую $\forall v \in V^d$ единственный содержащий его $g \in G$, причем $\overline{G}(v) = \varnothing \Leftrightarrow v \in V*$. Также используется естественное расширение этой функции для подмножеств $\overline{G} : 2^V \to 2^G$, $\overline{G}(p) = \{g \mid \exists v \in p : v \in g\}$;

4) $S_0 = \{< v_1, s_1 >,..,< v_n, s_n >\} \in S(V)$, где $\{v_1,..,v_n\} = V$, $v_i \neq v_j \Leftrightarrow i \neq j$, – начальное означивание множества объектов $V$, задающее *начальную функцию означивания* $S_o(v_i) = s_i$;

5) $F : S(V) \times S(V) \to [0,\infty)$ – функция близости означиваний;

6) $\varepsilon \in [0,\infty)$ – точность вычислений.

**Определение 1.6** Набором трансформаций $T = ((t_1,..,t_k),(\Delta_1,..,\Delta_{p+q}))$ для задачи $Z = ((V^d \cup V*, P), \widetilde{E}, G, S_0, F, \varepsilon)$ будем называть набор из движений $t_i$ всех $k$ тел $\{g_1,..,g_k\} = G$ этой задачи и приращений $\Delta_j \in R$ всех $p$ дополнительных параметров геометрических объектов из $V^d$ и $q$ переменных $V*$. Действие $T : S(V) \to S(V)$ этого набора трансформаций на означивании $S = \{< v_1, s_1 >,..,< v_n, s_n >\} \in S(V)$, где $\{v_1,..,v_{n-q}\} \in V^d$, $\{v_{n-q+1},..,v_n\} \in V*$ задачи $Z$ есть означивание $T(S) = \{< v_1, \tilde{s}_1 >,..,< v_n, \tilde{s}_n >\} \in S(V)$, причем:

1) если $i \in \{1,..,n-q\}$, то $\tilde{s}_i = \Delta(t_u(s_i))$, где $u : g_u = \overline{G}(v_i)$, $\Delta = (\Delta_j,..,\Delta_l)$, $l = \sum_{r\in\{1,..,i\}} P(v_r)$, $j = l - P(v_i)$;

2) если $i \in \{n-q+1,..,n\}$, то $\tilde{s}_i = s_i + \Delta_{p+q-n+i}$.

**Определение 1.7** Означивание $S$ задачи $Z=(\tilde{V},\tilde{E},G,S_0,F,\varepsilon)$ называется допустимым, если $\exists T$ – набор трансформаций задачи $Z$ такой, что $S=T(S_0)$. Такой набор трансформаций $T$ означивания $S$ будем обозначать $T_S$.

**Определение 1.8** Означивание $S=\{<v_1,s_1>,..,<v_n,s_n>\}$ задачи $Z=(\tilde{V},(E,a,A),G,S_0,F,\varepsilon)$ называется решающим, если оно допустимо и $\forall e\in E$, $e(S(A(e,1)),..,S(A(e,a(e))))\le\varepsilon$. Набор трансформаций $T_S$ для такого означивания $S$ называется решением задачи $Z$. Множество всех решений задачи $Z$ обозначается $T(Z)$.

**Определение 1.9** Оптимальным решением задачи $Z=(\tilde{V},\tilde{E},G,S_0,F,\varepsilon)$ называется $T\in T(Z)$, такое что $\forall T'\in T(Z), F(T(S_0),S_0)\le F(T'(S_0),S_0)$.

Как легко видеть, если положить $\varepsilon=0$, то задача сведется к “идеальной” математической задаче параметрического проектирования $Z=(\tilde{V},\tilde{E},G,S_0,F)$, в которой ограничения понимаются как отношения, а не функции невязки. Если положить $F(S_1,S_2)$ тождественно равной нулю, а $S_0$ выбрать произвольным, то получится задача $Z=(\tilde{V},\tilde{E},G)$ без требования оптимальности решения. Наконец, если положить $G=\{\{v_1\},..,\{v_n\}\},\{v_1,..,v_n\}=V^d$, то задача сведется к известной классической постановке $Z=(\tilde{V},\tilde{E})$.

**Определение 1.10** Ограничение $e\in E$ над объектами $v_1,..,v_{a(e)}$ называется геометрическим, если для любого движения $t$ и любого означивания $S$ объектов $v_1,..,v_{a(e)}$ выполняется равенство $e(t(S(v_1)),..,t(S(v_{a(e)})))=e(S(v_1),..,S(v_{a(e)}))$. Ограничения, не являющиеся геометрическими, называются инженерными.

**Определение 1.11** Гиперграфом задачи $Z=(\tilde{V},(E,a,A),G,S_0,F,\varepsilon)$ называется двойка $(G,E')$, где $E'\subseteq 2^G$ и $e'\in E'\Leftrightarrow \exists e\in E:\bigcup\limits_{i\in\{1,..,a(e)\}}\overline{G}(A(e,i))=e'$.

## 1.3 Степени свободы в задачах параметрического проектирования

В случае достаточно простых тел возможно, что после выполнения нетождественного движения тело переходит в себя, то есть движение тела является автоморфизмом. Например, если тело состоит только из точки, лежащей в начале координат, то любое вращение будет автоморфизмом [72]. Для таких тел будем рассматривать наименьший набор параметров движений (компонент матрицы перехода $A$ и вектора сдвига $\bar{b}$), представляющий все движения, различные с точностью до автоморфизма. Такой набор будем называть набором независимых параметров. Например, для двухмерной прямой им будет набор из 2 параметров: угла с осью абсцисс и расстояния до начала координат.

**Определение 1.12** Числом степеней свободы тела $g$ в $d$-мерном пространстве называют количество $d(g)$ независимых параметров, описывающих все его движения с точностью до автоморфизмов. Тела, не имеющие автоморфизмов, называются полными телами, для них число степеней свободы равно $C_2^{d+1}$.

Пространством независимых параметров задачи $Z = ((V = V^d \cup V^*, P), \tilde{E}, G, S_0, F, \varepsilon)$ будем называть набор параметров, состоящий из объединения наборов независимых параметров всех тел $g_i \in G$ и множества из $p = \sum_{v \in V} P(v)$ дополнительных параметров объектов и $q$ вещественных переменных $V^*$. В пространстве независимых параметров множество решений задачи с $\varepsilon = 0$ представляет собой некоторое многообразие.

**Определение 1.13** Размерность $i(Z)$ многообразия решений задачи $Z = (\tilde{V}, \tilde{E}, G, S_0, F, 0)$ называется истинным числом степеней свободы задачи $Z = (\tilde{V}, \tilde{E}, G, S_0, F, \varepsilon)$. Эта величина отражает тот факт, что количество

независимых параметров, необходимых для выделения из множества "идеальных" решений задачи некоторого конкретного, равно $i(Z)$.

Заметим, что при добавлении в модель нового геометрического объекта число степеней свободы модели увеличивается на количество независимых параметров этого объекта. Например, при добавлении двухмерной прямой с обобщенными координатами $\begin{pmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{pmatrix}, (x, y)$ число степеней свободы увеличивается на два. Действительно, любое движение прямой можно параметризировать одним смещением вдоль ее нормали и одним вращением вокруг начала координат, а любое решение новой задачи можно представить как решение старой и произвольное движение добавленной прямой, никак не связанной ограничениями с остальными объектами.

Каждому типу геометрических объектов можно сопоставить число добавляемых степеней свободы. При добавлении ограничения число степеней свободы, наоборот, уменьшается. Каждому типу геометрических ограничений можно сопоставить число снимаемых степеней свободы (см. приложение 1). Например, расстояние от точки до прямой в двухмерном пространстве снимает одну степень свободы.

**Определение 1.14** Пусть для задачи $Z = (\tilde{V}, (E, a, A), G, S_0, F, \varepsilon)$, где $p = \sum_{v \in V} P(v)$, $q$ – мощность множества переменных, задана *функция степеней свободы* $h : E \to N$, сопоставляющая каждому ограничению $e \in E$ число снимаемых им степеней свободы. Формальным числом степеней свободы задачи $Z$ назовем

$$d(Z) = \sum_{g \in G} d(g) + p + q - \sum_{e \in E} h(e) \qquad (1.2)$$

**Определение 1.15** Задача параметрического проектирования $Z = (\tilde{V}, \tilde{E}, G, S_0, F, \varepsilon)$, где $\tilde{V} = (V^d \cup V^*, P)$ называется переопределенной, если для

нее формальное число степеней свободы $d(Z)$ меньше числа степеней свободы $d(g)$ тела $g = V^d$, составленного из всех объектов этой задачи.

**Определение 1.16** Задача параметрического проектирования $Z' = (\tilde{V}', \tilde{E}', G', S_0', F', \varepsilon)$ называется подзадачей задачи $Z = ((V = V^d \cup V^*, P), (E, a, A), G, S_0, F, \varepsilon)$ по множеству тел $G'$, что обозначается $Z|_{G'} Z'$, если:

1) $G' \subseteq G$;

2) $\tilde{V}' = (V' = V^d' \cup V^{*'}, P), V^d' = \{v \in V^d \mid \overline{G}(v) \in G'\}$;

3) $\tilde{E}' = (E', a, A), E' = \{e \in E \mid \forall i \le a(e), A(e,i) \in V^d' \cup V^*\}$;

4) $V^{*'} = \{v \in V^* \mid \exists e \in E' \exists i \le a(e) : A(e,i) = v\}$;

5) $S_0' = \{< v, s > \in S_0 \mid v \in V'\}$;

6) $F' : S(V') \times S(V') \to [0, \infty)$, такая что $F'(S_1', S_2') = F(S_1' \cup S_\Delta, S_2' \cup S_\Delta)$, где $S_\Delta = \bigcup_{v \in V \setminus V'} < v, S_0(v) >$.

**Определение 1.17** Задача параметрического проектирования $Z = (\tilde{V}, \tilde{E}, G, S_0, F, \varepsilon)$ называется структурно переопределенной, если существует ее подзадача, являющаяся переопределенной.

**Определение 1.18** Задача параметрического проектирования $Z = (\tilde{V}, \tilde{E}, G, S_0, F, \varepsilon)$ называется недоопределенной, если она не является структурно переопределенной и для нее формальное число степеней свободы $d(Z)$ больше числа степеней свободы $d(g)$ тела $g = V^d$, составленного из всех объектов этой задачи.

**Определение 1.19** Задача параметрического проектирования $Z$ называется хорошо определенной, если она не является ни структурно переопределенной, ни недоопределенной.

Иногда в литературе вместо указанных выше определений используется формулировка понятия хорошо определенной задачи как задачи

с $d(Z)$ меньшим 3 в двухмерном и 6 в трехмерном пространстве [51]. Однако такая формулировка не является корректной для многих задач, которые, будучи рассмотрены как твердое тело, имеют нетривиальные автоморфизмы. Как правило, в определениях не учитываются также дополнительные параметры объектов и вещественные переменные задачи.

Классификация задач на недо-, пере- и хорошо определенные позволяет априорно оценить эффективность для них различных методов, в частности, многих методов декомпозиции [58].

Далее в этом параграфе ограничения будут считаться бинарными, таким образом, гиперграф задачи превращается в мультиграф. Во многих контекстах ребра мультиграфа между одними и теми же объектами можно склеить и получить обыкновенный граф.

**Определение 1.20** Пусть для задачи $Z = (\bar{V}, (E, a, A), G, S_0, F, \varepsilon)$ задана $h : E \to N$, сопоставляющая каждому ограничению $e \in E$ число снимаемых им степеней свободы. Граф $(G, E^*)$ задачи $Z$ назовем помеченным, если заданы функции пометки $\bar{f} : G \to N$ и $\bar{h} : E^* \to N$, такие что:

1) $\forall g \in G, \bar{f}(g) = d(g)$;

2) $\bar{h}(e^*) = \sum_{e \in M(e^*)} h(e)$, где $M(e^*) = \{e \mid \{\overline{G}(A(e,1)), \overline{G}(A(e,2))\} = e^*\}$.

Таким образом, в помеченном графе $(G, E^*, \bar{f}, \bar{h})$ функции $\bar{f}, \bar{h}$ задают, сколько степеней свободы есть у каждого тела, и сколько степеней свободы снимается ограничениями между любой парой тел. Для разметки графа необходимо знать функцию степеней свободы $h : E \to N$.

Если ограничить класс рассматриваемых сущностей перечислением их конкретных типов, например:

1) двухмерные элементарные геометрические объекты: точки, прямые, окружности, эллипсы и произвольные кривые, не меняющие свою форму;

2) трехмерные элементарные геометрические объекты: точки, плоскости, прямые, сферы, цилиндры, окружности, произвольные кривые и поверхности, не меняющие свою форму;

3) геометрические ограничения в двух- и трехмерном пространстве: инцидентность (один объект является частью другого), расстояние, угол, параллельность, перпендикулярность, касание, концентричность; в трехмерном пространстве дополнительно рассматривается ограничение соостности;

то из простых геометрических соображений функцию степеней свободы $h : E \to N$ можно считать заданной (см. приложение 1), так что формальное число степеней свободы задачи становится легко вычислимой характеристикой задачи параметрического проектирования.

**Определение 1.21** Два помеченных графа $(G_1, E_1^*, \bar{f}_1, \bar{h}_1)$ и $(G_2, E_2^*, \bar{f}_2, \bar{h}_2)$ будем называть изоморфными, если существуют биекция вершин и ребер $b : (G_1, E_1^*) \to (G_2, E_2^*)$ такая, что:

1) $\forall g \in G_1, \bar{f}_2(b(g)) = \bar{f}_1(g)$

2) $\forall e \in E_1^*, \bar{h}_2(b(e)) = \bar{h}_1(e)$

**Определение 1.22** Две задачи $Z_1$ и $Z_2$, включающие только бинарные ограничения, называются эквивалентными по степеням, если их помеченные графы изоморфны. Классы эквивалентности на множестве графов задач по отношению изоморфизма называются формальными типами.

Заметим, что совершенно разные по смыслу задачи параметрического проектирования могут иметь изоморфные помеченные графы, например, двухмерные задача позиционирования трех точек на заданных расстояниях друг от друга и задача позиционирования точки на заданных расстояниях от прямых, угол между которыми известен, имеют изоморфные графы. Действительно, все объекты этих задач имеют две степени свободы, а все ограничения – одну (см. рис. 1.1).

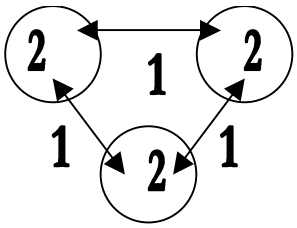


Рисунок 1.1. Разные задачи обозначаются одним помеченным графом

**Определение 1.23** Формальные типы для хорошо определенных задач называются формальными кластерными типами. Формальный кластерный тип, в котором нет подграфа, тоже являющегося формальным кластерным типом, называется несократимым.

**Предложение 1.1** В любом формальном кластерном типе $(G, E^*, \bar{f}, \bar{h})$ задачи $Z$ верно

$$\forall g' \in G, [d(\bigcup_{g \in G} g) = d(\bigcup_{g \in G \setminus g'} g) \Rightarrow \bar{f}(g') \le \sum_{e \in N(g')} \bar{h}(e)], \qquad (1.3)$$

где $N(g') = \{e \in E^* \mid g' \in e\}$. Если же тип является несократимым, то неравенство в формуле (1.3) строгое.

**Доказательство** Рассмотрим подграф $(G \setminus g', E^* \setminus N(g'), \bar{f}, \bar{h})$ и соответствующую ему подзадачу $Z' = Z|_{G \setminus g'}$. Легко видеть, что $d(Z') = d(Z) - \bar{f}(g') + \sum_{e \in N(g')} \bar{h}(e)$. Так как $Z$ хорошо определена, то $d(\bigcup_{g \in G} g) = d(Z)$, а раз никакая ее подзадача не может быть переопределенной, то $d(\bigcup_{g \in G \setminus g'} g) \le d(Z')$, причем если $(G, E^*, \bar{f}, \bar{h})$ несократим, то неравенство строгое. Отсюда получаем, что

$$\bar{f}(g') - \sum_{e \in N(g')} \bar{h}(e) = d(Z) - d(Z') \le d(\bigcup_{g \in G} g) - d(\bigcup_{g \in G \setminus g'} g) = 0$$

Можно сказать, что формальные кластерные типы описывают структуру степеней свободы в хорошо определенных задачах. Эти задачи представляют особый интерес в методах геометрической декомпозиции,

особенно если соответствующий им формальный кластерный тип является несократимым.

Попробуем определить несократимые формальные кластерные типы для простейшего класса двухмерных задач параметрического проектирования, объектами которых являются имеющие две степени свободы точки и прямые, а рассматриваемые ограничения между ними снимают одну степень свободы. Этот класс задач рассматривается во многих источниках. Все тела таких задач могут иметь лишь 2 или 3 степени свободы. Вследствие этого у всех рассматриваемых формальных кластерных типов число степеней свобод будет равно трем. В помеченном графе таких задач все вершины имеют пометку 2 или 3, поэтому обозначим этот класс $Z^{\{2,3\}}$.

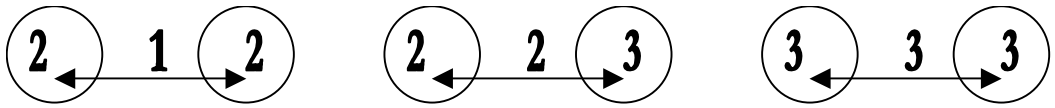


Рисунок 1.2. $I$ -, $V$ - и $E$ -типы

Как легко видеть, есть три несократимых формальных кластерных типа $Z^{\{2,3\}}$, состоящих из двух вершин [44] (см. рис. 1.2). Далее они будут именоваться соответственно $I$ -типом, $V$ -типом и $E$ -типом.

**Предложение 1.2** В любом несократимом формальном кластерном типе $Z^{\{2,3\}}$

1) любая вершина с пометкой 2 не может иметь ребро с вершиной, пометка которой равна 2;
2) любая вершина с пометкой 2 не может иметь ребро, пометка которого больше 1, с вершиной, пометка которой равна 3;
3) любая вершина с пометкой 3 не может иметь ребро, пометка которого больше 2, с вершиной, пометка которой равна 3;

**Доказательство** От противного, если бы это было не так, то есть некоторая пара вершин противоречила одному из упоминающихся выше

утверждений, то подграф из этих двух вершин был бы $I$-, $V$- или $E$-формальным кластерным типом, что противоречит свойству несократимости.

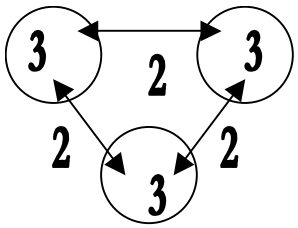

Рисунок 1.3

**Предложение 1.3** Единственным несократимым формальным кластерным типом $Z^{\{2,3\}}$, содержащим три вершины, является тип, описанный на рис. 1.3.

**Доказательство** Пусть существует несократимый формальный кластерный тип $Z^{\{2,3\}}$, содержащий три вершины, хотя бы одна из которых имеет пометку 2. Тогда, по предложению 1.1, левая часть формулы (1.3) в котором всегда истинна для $Z^{\{2,3\}}$, ее степень не меньше 3. Но, так как по предложению 1.2, она не может иметь ребра с пометкой более 1 с каждым из оставшихся двух объектов, откуда следует противоречие. Значит, несократимый формальный кластерный тип состоит из трех вершины с пометкой 3. Он должен также содержать ребра с пометками, сумма которых равна 6, чтобы его число степеней свободы равнялось 3. Так как ребро между каждой из трех пар вершин не может иметь пометки выше 2 по предложению 1.2, то единственно возможным вариантом несократимого формального кластерного типа $Z^{\{2,3\}}$ является описанный в формулировке предложения 1.3 тип.

**Предложение 1.4** Множество несократимых формальных кластерных типов для $Z^{\{2,3\}}$ бесконечно.

**Доказательство** Докажем по индукции, что для любого $k \geq 3$ существует несократимый формальный кластерный тип с $k$ вершинами, в котором некоторое ребро с пометкой 2 соединяет вершины с пометкой 3 (это условие будем называть условием $A$). Базой индукции будет удовлетворяющий условию $A$ несократимый формальный кластерный тип из предложения 1.3. Рассмотрим несократимый формальный кластерный тип $(G, E^*, \bar{f}, \bar{h})$ некоторой задачи $Z$, удовлетворяющий условию $A$, и построим новый несократимый формальный кластерный тип $(G', E'^*, \bar{f}', \bar{h}')$ с большим

числом вершин, также удовлетворяющий условию $A$. Из условия $A$ следует, что $\exists e^* = \{a,b\} \in E^*$ с пометкой 2, введем новую вершину $g' \notin G$ и зададим $(G', E'^*, \bar{f}', \bar{h}')$ так:

1) $G' = G \cup g'$;

2) $E'^* = E^* \cup \{g', a\} \cup \{g', b\}$;

3) $\forall g \in G, \bar{f}'(g) = \bar{f}(g)$ и $\bar{f}'(g') = 3$;

4) $\forall e \in E^* \backslash e^*, \bar{h}'(e) = \bar{h}(e)$, $\bar{h}'(e^*) = \bar{h}(e^*) - 1$, $\bar{h}'(\{g', a\}) = 2$, $\bar{h}'(\{g', b\}) = 2$.

Ребро $\{g', a\}$ имеет пометку 2 и соединяет вершины с пометкой 3, так что $(G', E'^*, \bar{f}', \bar{h}')$ удовлетворяет условию $A$. Осталось доказать, что $(G', E'^*, \bar{f}', \bar{h}')$ есть несократимый формальный кластерный тип задачи $Z'$, полученной из $Z$ построением 1)-4). Действительно, $d(\bigcup_{g \in G'} g) = 3 = d(\bigcup_{g \in G} g)$, и

$$d(Z') = d(Z) + \bar{f}'(g') - \bar{h}'(\{g', a\}) - \bar{h}'(\{g', b\}) + 1 = d(Z) + 3 - 2 - 2 + 1 = d(Z), \qquad (1.4)$$

то есть новый помеченный граф является формальным кластерным типом. Предположим, что у него найдется собственный подграф $(G^*, E^{**}), G^* \subseteq G', E^{**} = \{e \in E'^* \mid e \subseteq G^*\}$, являющийся формальным кластерным типом. Если $g' \notin G^*$, то $G^* \subseteq G$, что противоречит несократимости $(G, E^*, \bar{f}, \bar{h})$. Значит, $g' \in G^*$ и, по предложению 1.1, $a \in G^*$, $b \in G^*$, из чего следует, что $e^* = \{a,b\} \in E^{**}$. Рассмотрим подграф $(\hat{G}, \hat{E})$, получаемый из $(G^*, E^{**})$ удалением вершины $g'$ и ребер $\{g', a\}, \{g', b\}$ и увеличением пометки ребра $e^*$ на единицу. По выкладкам из формулы (1.4) в обратном порядке, $(\hat{G}, \hat{E})$ является формальным кластерным типом. Но по построению $\hat{G} \subseteq G$ и $\hat{E} \subseteq E$, что противоречит несократимости $(G, E^*, \bar{f}, \bar{h})$. Предложение доказано.

Итак, задача выделения несократимых формальных кластерных типов в графе не сводится к перебору заранее заданных образцов, так как их число бесконечно даже для $Z^{\{2,3\}}$, что показано конструктивным построением.

## 1.4 Обзор существующих методов решения

### 1.4.1 Классификация направлений исследования

Далее будет приведен краткий обзор существующих подходов к решению задач параметрического проектирования. Задача параметрического проектирования является NP-сложной даже в двухмерном случае при наличии ограничений только одного типа – расстояний между точками [50]. К свойству NP-сложности необходимо добавить и свойство наличия экспоненциально большого числа решений.

При создании и совершенствовании моделей в САПР необходимо решать большое количество задач параметрического проектирования, поэтому использование экспоненциальных алгоритмов практически невозможно, вместо них применяются различные полиномиальные алгоритмы поиска одного из решений. Такие алгоритмы, в общем случае, не гарантируют нахождения решения.

Тем не менее, небольшое число публикаций посвящено разработке методов автоматического поиска в пространстве решений [90]. Другим подходом к решению этой проблемы является итеративное взаимодействие с пользователем, который может подсказывать системе, в каких подобластях искать решение задачи [13], но такой подход можно считать лишь сугубо экспериментальным, не подходящим для широкого использования в промышленности.

Одним из способов классификации подходов к решению задач параметрического проектирования является деление на символьные, численные и декомпозиционные методы [73]. Однако такой способ основывается только на формальной родовой принадлежности используемых методов, каждый из которых может быть лишь частью сложной схемы решения задачи. Поэтому обычно в литературе выделяют три классических вида подходов к решению задач параметрического проектирования [38]: алгебраическое моделирование, геометрическая декомпозиция и подход,

основанный на выводе в системе правил. С точки зрения автора, рационально обобщить третий из перечисленных подходов, так как методы вывода в системе правил можно отнести к области искусственного интеллекта, которая, кроме того, включает в себя и другие, появившиеся в последнее время, методы решения задач параметрического проектирования. В следующих трех параграфах будет рассмотрено каждое из этих направлений.

В решателях, применяющих численное алгебраическое моделирование, геометрические ограничения переводятся в алгебраические (обычно это полиномиальные уравнения), и полученная система решается некоторым набором численных методов, как правило, итеративных. Таким образом, при использовании алгебраического моделирования кроме выбора собственно метода моделирования необходимо также выбрать подходящие методы решения системы нелинейных уравнений. Поэтому последний параграф главы 1 будут посвящены таким методам, как численным, так и символьным.

Кроме исследований в хорошо изученных направлениях, есть и менее разработанные идеи, некоторые из которых рассматриваются ниже.

В работе [94] предлагается интерактивный метод решения задач с геометрическими ограничениями с помощью специального представления множеств возможных геометрических мест объектов, являющихся аргументами геометрических ограничений, и пересечения таких множеств. К сожалению, метод недостаточно алгоритмически корректно описан. Кроме того, он в значительной степени применяет взаимодействие с пользователем, что невозможно при индустриальном использовании геометрического решателя, который может многократно запускаться экспертной системой, встроенной в САПР, в течение небольшого промежутка времени.

Одним из предлагаемых неалгебраических подходов является метод геометрической релаксации [36]. Вектор значений параметров ограничений, при которых все ограничения являются удовлетворенными в начальной геометрической конфигурации, рассматривается как исходное

геометрическое описание состояния системы ограничений, из которого надо получить желаемое пользователем состояние. Для этого конструируется матрица коэффициентов, описывающих влияние изменений параметров двух ограничений друг на друга. Эта матрица позволяет оценивать возможные направления изменения состояния, с ее помощью выполняются итеративные сдвиги в направлении желаемого состояния системы. К сожалению, предлагаемый подход не описан общим алгоритмом, а лишь иллюстрируется на частных двухмерных примерах для систем точек и расстояний между ними.

### 1.4.2 Методы алгебраического моделирования

Наиболее известным способом алгебраического моделирования является декартово моделирование, при котором искомые координаты объектов являются переменными, а ограничения представляются в виде уравнений. Есть несколько вариантов такого моделирования, отличающихся количеством генерируемых для каждого тела переменных, более подробно они будут рассмотрены в главе 2.

Группа профессора Клемана применяет для разрешения геометрических моделей анализ на группах возможных перемещений геометрических объектов [40]. Задача параметрического проектирования представляется в виде набора точек и связывающих их векторов, а зависимости, которые необходимо разрешить – в виде замкнутых циклов векторов и ограничений угла между векторами [85]. К сожалению, описание построения по исходной задаче системы векторов и ограничений на них не является полным, поскольку этот подход применяется в коммерческих целях.

Одним из путей снижения временных затрат при решении систем нелинейных уравнений для небольших задач является использования вместо общего способа моделирования некоторого другого, специализированного для конкретной задачи. Например, для задач, включающих только два тела и несколько ограничений в трехмерном пространстве, предлагается прямое, без

итераций метода Ньютона, вычисление взаимных трансформаций на основе решения специальной системы линейных уравнений. Этот метод основан на анализе кинематики и применяется для разрешения ограничений в хорошо определенных задачах [68, 69]. Другим подходом является попытка сведения трехмерных задач к более простым двухмерным задачам [95].

Перспективным методом считается также использование в качестве переменных вместо декартовых координат объекта в глобальной системе координат того или иного способа позиционирования объекта по отношению к другому объекту, связанному с ним ограничениями. В работе [70] приводится описание метода разрешения одинарного цикла ограничений итеративной техникой релаксации ограничений. При этом внутри цикла ограничений объекты параметризируются друг относительно друга с помощью кинематических пар, использующих меньшее количество переменных.

Некоторые исследователи используют моделирование геометрических ограничений как операторов над тензорами в алгебре Грассмана-Кэли [76]. Описание такого метода моделирования является достаточно подробным, кроме того, авторы заявляют об успешном его прототипировании. Существенным недостатком изложенного подхода является ограничение класса решаемых задач из-за моделирования лишь проективных, то есть логических ограничений, что не позволяет решать задачи с параметрическими ограничениями.

Другая возможность моделировать задачу параметрического проектирования состоит в использовании детерминантов Кэли-Менгера как составных частей генерируемых полиномиальных уравнений [81]. Предлагаемый подход, к сожалению, ограничен классом задач, включающих только двухмерные точки и расстояния между ними.

### 1.4.3 Методы геометрической декомпозиции

Второй, и наиболее разработанный подход заключается в декомпозиции задачи на дерево подзадач, решать которые можно последовательно [60]. Решатели, основанные на таком подходе, позволяют локализовать вносимые в модель изменения, и не выполнять при них полный пересчет. Более того, так как символьные методы имеют, как правило, меньшую вычислительную сложность, то существенно сокращается общая сложность вычислений.

Наиболее известны две группы исследователей, работающих в этом направлении – это группа Оуэна и группа Хоффманна [66].

Группа Оуэна использует для решения задач параметрического проектирования мощный алгоритм декомпозиции модели сверху вниз с помощью разбиения графа объектов и ограничений на компоненты двойной и тройной связности [83, 84]. Эта группа имеет мало публикаций, зато хорошо известен разработанный ей решатель DCM, являющийся на данный момент лидером мирового рынка и внедренный в большинство САПР среднего и высшего уровня. Метод выделения компонент двойной связности корректно декомпозирует задачу и имеет линейную вычислительную сложность [7], что позволяет его считать одним из самых эффективных методов декомпозиции. Метод же выделения компонент тройной связности не всегда корректно декомпозирует задачу, и условия корректности такой декомпозиции в статьях Оуэна освещены не полным образом, что не позволяет эффективно использовать его для решения задач параметрического проектирования.

Наибольшее количество публикаций и наибольшую известность имеет группа профессора Хоффманна, которая развивает так называемый формальный анализ степеней свобод [59], позволяющий выделять в модели хорошо определенные подзадачи при обработке снизу вверх или сверху вниз.

Первыми появились более простые методы декомпозиции, основанные на выделении в графе объектов и ограничений хорошо определенных подзадач с помощью сравнения с предзаданными образцами. Позднее были предложены общие методы выделения хорошо определенных задач минимального размера на основе алгоритма поиска минимального потока в сети [61]. Этот алгоритм также пытались применять для так называемой структурной ригидификации задачи параметрического проектирования [64]. В более поздних работах Хоффманн дает формальное определение процесса декомпозиции задачи с помощью так называемого “плана декомпозиции-рекомбинации”.

Доказательства корректности описанных выше методов проводятся на достаточно узких классах задач [60]. Предложенные Хоффманном методы общи, но имеют большое количество недостатков, на которые он сам указывал, и некоторые из которых пытался устранить в более поздних статьях. В частности, он пытался улучшить плохую работу методов с переопределенными задачами [62], которые встречаются достаточно часто [75], однако в качестве решения предлагал удаление ограничений из переопределенных частей задач, что не является приемлемым на практике. Аналогичным образом предлагается избавляться от другого недостатка – неэффективной работы метода с недоопределенными задачами: введение дополнительных ограничений способно трансформировать исходную задачу в хорошо определенную задачу. Такой способ приводит к итеративному введению лишних уравнений в систему до получения квадратного вида, поэтому он способен серьезно увеличивать общее время решения.

Кроме того, отмечается малая эффективность предложенного метода для решения промышленных трехмерных задач. Другим недостатком является плохая работа методов Хоффманна с окружностями нефиксированного радиуса, которые требуют специальной нетривиальной обработки [39, 55, 56]. К ошибкам в работе метода приводит и зависимость

ограничений задачи между собой, которую предлагается определять с помощью специальных техник [78].

Некоторые работы в области геометрической декомпозиции посвящены более широким гибридным методам, например, в работе [89] есть описание сочетания метода декомпозиции с иерархическим проектированием, которое не вполне укладывается в рамки обычного решателя задач параметрического проектирования.

### 1.4.4 Методы искусственного интеллекта

Так как многие задачи, возникающие в САПР, в том числе задача параметрического проектирования, являются NP-сложными, то для их решения некоторые исследователи пытались применить методы искусственного интеллекта [5, 41].

Наибольшую историю применения имеют методы из области искусственного интеллекта, основанные на выводе в системе правил. При реализации этого подхода ограничения рассматриваются как известные факты, из которых можно выводить факты-следствия по заранее определенным правилам, среди которых могут быть как тождественно истинные, так и эвристические правила [88]. База фактов, таким образом, постепенно пополняется, в результате конструируются факты, которые задают координаты объектов, определяя тем самым решение задачи. Есть и другие известные алгоритмы, основанные на этом подходе, например, метод резолюций [52], однако разработанный на основе этого метода продукт не получил достаточного развития. Современные решатели, как правило, не используют подход на основе правил, однако разработки в этой области все еще ведутся [87].

Некоторые исследователи предлагают применять для решения генетические алгоритмы [67]. Однако в указанной работе эти алгоритмы рассматриваются не как общий метод решения, а как метод перебора в экспоненциальном пространстве решений задачи, учитывающий некоторую

функцию оптимальности решения. Таким образом, предложенные в статье алгоритмы не способны стать ядром подхода к решению задач параметрического проектирования.

В работе [79] рассматривается возможность применения мультиагентных систем. Предложенная схема позволяет достаточно эффективно решать задачи параметрического проектирования, которые возможно разделить на независимые или почти независимые части с помощью методов геометрической декомпозиции, после чего каждую из частей можно решать с помощью специального агента. Эта работа повышает общую значимость исследований в области геометрической декомпозиции, поскольку позволяет снизить время решения декомпозируемых задач.

Особое место занимают методы программирования в ограничениях, позволяющие работать с недоопределенными данными, а также недифференцируемыми, разрывными и дискретными ограничениями.

Работа с недоопределенными данными в виде множеств в трехмерной САПР описывается в работах [82, 91], однако предлагаемые методы являются лишь надстройкой над вычислительным ядром. Рассматриваются также интервальные методы и интервальные постановки задач, например, задача определения интервалов значений одного параметра по другим параметрам, означенным константами [93]. В алгоритме используется существенным образом априорно существующая декомпозиция на простейшие аналитически разрешимые подзадачи, что не позволяет его считать практически значимым.

Использование интервальных методов для борьбы с проблемами ошибок округления предлагается в [54], однако, в этой работе не рассматривается конкретный подход, а лишь описываются интервальные вычисления и их свойства. Перспективность разработки интервальных решателей показана и в статье-обзоре профессора Миккелуччи о наиболее важных современных направлениях исследования в задачах

параметрического проектирования [80]. Достаточно полный подход описывается в [65], однако предлагаемый в нем метод поиска решения с откатами имеет экспоненциальную вычислительную сложность.

### 1.4.5 Методы решения систем нелинейных уравнений

Наиболее часто используемым численным методом является метод Ньютона-Рафсона (далее – просто метод Ньютона), который привлекает своей относительной простотой и высокой скоростью сходимости [4]. Тем не менее, этот метод требует хороших начальных приближений, не гарантирует сходимость к решению и не так быстр в многомерном случае. Все большую и большую популярность завоевывает метод гомотопии [74], который, по сравнению с методом Ньютона, ведет себя более регулярно, с высокой вероятностью сходясь к ближайшему относительно начального приближения решению, однако является более медленным. Менее популярны, но изредка применяются для решения задач параметрического проектирования методы релаксации, метод проекций и некоторые другие общие и специализированные методы.

В целом, численные методы хороши тем, что позволяют решать переопределенные (но совместные) и недоопределенные задачи, но плохи тем, что имеют высокую вычислительную сложность, а изменение параметров модели ведет к полному ее пересчету.

При использовании таких популярных методов моделирования, как простое декартово моделирование, генерируемые нелинейные системы являются полиномиальными. Как известно, такие системы можно приводить к более удобному для решения виду с помощью базисов Гребнера [37]. Использование символьных методов, к числу которых принадлежит метод базисов Гребнера, позволяет улучшать решаемость задач благодаря избежанию ошибок округления [54]. Однако вычислительная сложность самого алгоритма нахождения базиса Гребнера не менее чем

экспоненциальна, поэтому его неразумно использовать в процессе решения получаемых нелинейных систем.

Значительно более эффективной является декомпозиция Далмейджа-Мендельсона, которая выделяет в системе уравнений подсистемы, содержащие одинаковое число уравнений и переменных [42]. Если считать, что каждое уравнение снимает одну степень свободы с набора переменных, то такая подсистема должна иметь конечное число корней, следовательно, ее можно решить и подставить полученные значения переменных в оставшиеся уравнения. Несмотря на то, что такие действия не всегда корректны, этот метод очень эффективен, так как способен многократно уменьшить размеры решаемых систем нелинейных уравнений.

Кроме методов преобразования всей системы нелинейных уравнений используются также более простые техники: метод исключения и подстановки Гаусса для линейных уравнений, решение квадратных уравнений от одной переменной, метод приведения квадратичной формы к главным осям и использование универсальной тригонометрической подстановки [43]. Эти методы являются эффективными, полиномиальными по сложности вычислительными алгоритмами, позволяющими снизить размерность решаемой системы уравнений, поэтому их применение, как правило, является оправданным. Некоторым недостатком этих методов является накопление погрешностей при их выполнении, которое может привести к численным проблемам при дальнейшем решении с помощью метода Ньютона.

## 1.5 Выводы

Существующие работы в области задач параметрического проектирования имеют значительные недостатки. К ним следует отнести: нечеткую формализацию объектов исследований, рассмотрение частных случаев вместо общей проблемы, недостаточное обоснование корректности

предлагаемых методов, неполное изложение алгоритмов, необходимых для реализации предлагаемых подходов.

В данной главе предложены формальные определения понятия “задача параметрического проектирования” и ее составных частей, а также указано, что является решением и оптимальным решением такой задачи. Кроме того, усовершенствованы некоторые известные определения таких характеристик задачи параметрического проектирования, как число степеней свободы, принадлежность к классам пере-, недо- и хорошо определенных задач. Введены определения, позволяющие работать с задачами параметрического проектирования с помощью методов анализа графов и гиперграфов.

Проведенный обзор подходов к решению задач параметрического проектирования позволяет сформулировать дальнейшие задачи диссертационного исследования.

Первой задачей целесообразно считать разработку общего метода алгебраического моделирования задач параметрического проектирования, способного генерировать системы нелинейных уравнений, размер которых меньше, чем у существующих методов моделирования. Именно размер системы уравнений определяет общее время работы геометрического решателя, так как решение сгенерированной системы уравнений методом Ньютона занимает львиную долю этого времени.

Второй задачей предлагается считать исследование эффективности существующих и поиск новых высокопроизводительных методов геометрической декомпозиции. Именно эти методы позволяют существенно уменьшить размер системы уравнений для метода Ньютона.

В качестве третьей и последней теоретической задачи рассматривается изучение возможности применения методов искусственного интеллекта, в частности, интервальных методов, для решения задач параметрического проектирования и разработка математических основ интервального решателя для таких задач. Это направление исследований является одним из самых

современных, но все еще не имеет достаточного количества конкретных результатов.

Полученные в ходе выполнения вышеупомянутых задач теоретические результаты должны получить экспериментальное подтверждение эффективности на репрезентативном наборе индустриальных тестов.

Для этого необходимо практическое воплощение предложенных методов в виде качественных программных систем для решения задач параметрического проектирования (в общей постановке на ЭВМ) как для двухмерного, так и для трехмерного случая. Это и является последней, практической задачей, элементом которой должна стать и апробация реализованных программных компонент в конечно-пользовательских программах, в том числе в промышленных САПР.

# 2 Алгебраическое моделирование задач параметрического проектирования

## 2.1 Постановка задачи

Как было сказано в главе 1, при решении задача параметрического проектирования рано или поздно превращается в систему нелинейных уравнений, решаемую с помощью специальных вычислительных методов. Основными переменными в такой системе уравнений являются либо новые координаты ее объектов, либо параметры набора трансформаций, описывающие процесс перевода объектов в новое положение. Далее без ограничения общности рассмотрим случай трехмерных задач параметрического проектирования.

Реальные детали в САПР достаточно сложны и при моделировании они представляются твердыми телами – наборами элементарных объектов, которые не меняют положения по отношению друг к другу. Поэтому на практике вместо нахождения координат всех элементарных объектов ищутся так называемые *трансформации* твердых тел вида

$$T(\alpha,\beta,\gamma,x,y,z)=\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}, \text{ где } R=R_\alpha \circ R_\beta \circ R_\gamma,\ t=(x,y,z)^T, \qquad (2.1)$$

$R_\alpha, R_\beta, R_\gamma$ – композиция матриц 3x3 поворота вокруг осей $Ox, Oy, Oz$ на углы Эйлера $\alpha,\beta,\gamma$ и $t=(x,y,z)^T$ – вектор смещения вдоль этих осей.

Действие трансформации на объект тела происходит согласно формуле

$$\begin{pmatrix} A_{new} & b_{new} \\ 0 & 1 \end{pmatrix} = X_{new} = T \circ X_{old} = \begin{pmatrix} R \circ A_{old} & R \circ b_{old} + t \\ 0 & 1 \end{pmatrix}, \qquad (2.2)$$

где $X_{old} = \begin{pmatrix} A_{old} & b_{old} \\ 0 & 1 \end{pmatrix}$, $X_{old}, X_{new}$ – новые и старые координаты объекта соответственно (для любого объекта его координаты можно представить в

виде матрицы перехода $A$ и вектора сдвига $b$ из глобальной системы координат в локальную систему координат объекта).

Ограничения, заданные пользователем, можно записать как уравнения на искомые положения тел $X_{new}$, таким образом, они выражаются через ротационные переменные $\alpha, \beta, \gamma$ и трансляционные $x, y, z$. Семейство методов моделирования, в которых неизвестными считаются переменные, определяющие трансформации твёрдых тел вида (2.1), называется декартовым моделированием. Разные методы декартова моделирования отличаются, в частности, способом представления трансформаций.

Так называемое 12-параметрическое моделирование заключается в использовании в качестве переменных для каждого твердого тела 12 переменных – 9 на матрицу поворота и 3 на вектор смещения:

$$A = \begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{pmatrix}, \; b = \begin{pmatrix} x_{10} \\ x_{11} \\ x_{12} \end{pmatrix}$$

Так как матрица A должна быть матрицей поворота в трехмерном пространстве, то есть ортогональной матрицей с единичным определителем, в систему автоматически добавляется шесть квадратных уравнений, обеспечивающих это свойство. Новые положения объектов являются линейными комбинациями переменных, а невязка большинства встречающихся на практике ограничений есть линейная или квадратичная форма от положения объектов-аргументов, вследствие чего генерируемые системы уравнений часто содержат только квадратные уравнения.

Другой, более эффективный способ декартова моделирования, называется 6-параметрическим моделированием. В нем переменными являются углы Эйлера $\alpha, \beta, \gamma$ и компоненты вектора сдвига $x, y, z$. Тогда матрица поворота $R = R_\alpha \circ R_\beta \circ R_\gamma$ примет следующий вид [18]:

$$\begin{pmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \sin\alpha\sin\beta\cos\gamma+\cos\alpha\sin\gamma & -\sin\alpha\sin\beta\sin\gamma+\cos\alpha\cos\gamma & -\sin\alpha\cos\beta \\ -\cos\alpha\sin\beta\cos\gamma+\sin\alpha\sin\gamma & \cos\alpha\sin\beta\sin\gamma+\sin\alpha\cos\gamma & \cos\alpha\cos\beta \end{pmatrix} \quad (2.3)$$

Плюс этого подхода состоит в том, что в генерируемой системе уравнений вдвое уменьшается количество переменных и значительно уменьшается количество уравнений. Минусом является то, что при этом возрастает степень получаемых уравнений. Кроме того, система уравнений включает тригонометрические функции. Тем не менее, в целом этот подход является более эффективным, поскольку главное требование, предъявляемое к любому методу моделирования, заключается в генерации как можно меньшей системы уравнений. Именно решение систем уравнений, полученных после завершения работы метода моделирования, является наиболее вычислительно сложной частью процесса решения задачи параметрического проектирования.

Таким образом, практически важной задачей является создание метода алгебраического моделирования, который сводит задачу параметрического проектирования к решению системы уравнений, размерность которой меньше, чем у 6-параметрического декартова моделирования. Ниже будет предложен новый метод остовного моделирования, который позволяет значительно уменьшить размер решаемых систем уравнений.

Другим важным вопросом, требующим исследования, является вопрос о методах решения систем полиномиальных уравнений, получающихся после декартова моделирования задач параметрического проектирования. Предлагается рассмотреть вопрос о применимости символьных методов решения таких уравнений, которые позволяют избежать появления погрешностей округления, в частности, метода базисов Гребнера, который, кроме того, позволяет свести решение системы к решению меньших систем и даже отдельных уравнений от одной переменной.

## 2.2 Метод остовного моделирования

### 2.2.1 Геометрические основы метода

**Определение 2.1** Каноническим положением элементарного объекта в трехмерной системе координат будем называть:

- Для точки: положение в центре $(0,0,0)$ системы координат;
- Для прямой: совпадение с осью $Ox$ системы координат;
- Для плоскости: совпадение с плоскостью $Oyz$ системы координат;
- Для цилиндра: положение, при котором ось цилиндра есть ось $Ox$;
- Для окружности: положение, при котором ось окружности, проходящая через ее центр перпендикулярно ее плоскости, есть ось $Ox$;
- Для сферы: положение, при котором центр сферы есть центр $(0,0,0)$ системы координат;
- Для кривой и плоскости: положение, при котором система координат, в которой они заданы, совпадает с указанной системы координат;

Если система координат не указана, то каноническое положение рассматривается в глобальной системе координат. Легко видеть, что положения объектов можно представить как переход из глобальной системы координат в систему координат объекта, тогда канонические положения объектов описываются тождественной трансформацией $I = \begin{pmatrix} E & 0 \\ 0 & 1 \end{pmatrix}$, где $E$ – единичная 3x3 матрица. Заметим, что такое представление объекта единственно с точностью до его автоморфизмов, которые будут описаны позднее.

**Определение 2.2** Говорят, что трансформация $T = \begin{pmatrix} A & b \\ 0 & 1 \end{pmatrix}$ удовлетворяет бинарное ограничение по первому (второму) аргументу, если $e(T,I) \le \varepsilon$ $(e(I,T) \le \varepsilon)$, то есть положение объекта, описываемого такой

трансформацией, совместно по ограничению $e$ с каноническим положением другого объекта-аргумента. Для конкретного объекта-аргумента $v$ такая трансформация также называется производным положением объекта $v$ по бинарному ограничению $e \in E$. Множество таких трансформаций обозначается $T_e^1$ ($T_e^2$).

Далее без ограничения общности будем считать, что рассматриваются производные положения первого аргумента ограничения.

**Определение 2.3** Частичной параметрической трансформацией, или частью, будем называть трансформацию $T(p_1,..,p_k)$, получающуюся из $T(\alpha,\beta,\gamma,x,y,z)$, заданной формулой (2.1), положением ненулевого подмножества $Q=\{q_1,..,q_l\}$, $k+l=6$ из параметров $U=\{\alpha,\beta,\gamma,x,y,z\}$ тождественно равными нулю. Параметры из $\{p_1,..,p_k\}=P=U\setminus Q$ называются действительными, а параметры из $Q$ – исключенными. Количество $k$ параметров из $P$ называется мощностью части.

**Предложение 2.1** Частичными параметрическими трансформациями $A^v(p_1,..,p_k)$, описывающими все автоморфизмы объекта $v$ в каноническом положении, являются:

- Для точки и сферы: $A^T(\alpha,\beta,\gamma)=\begin{pmatrix} R_\alpha \circ R_\beta \circ R_\gamma & 0 \\ 0 & 1 \end{pmatrix}$;
- Для прямой и цилиндра: $A^L(\alpha,x)=\begin{pmatrix} R_\alpha & t_x \\ 0 & 1 \end{pmatrix}, t_x=(x,0,0)$;
- Для плоскости: $A^P(\alpha,y,z)=\begin{pmatrix} R_\alpha & t_{y,z} \\ 0 & 1 \end{pmatrix}, t_{y,z}=(0,y,z)$;
- Для окружности: $A^C(\alpha)=\begin{pmatrix} R_\alpha & 0 \\ 0 & 1 \end{pmatrix}$;
- Для кривой и поверхности: $A^0=I$.

Доказательство легко следует из геометрических свойств этих объектов. Заметим, что если $T$ удовлетворяет бинарное ограничение $e \in E$, то и $T \circ A$ тоже удовлетворяет $e$.

**Определение 2.4** Производным представлением объекта $v$ по бинарному ограничению $e \in E$, аргументами которого являются $v, w$, называют параметрическое представление всех производных положений объекта $v$ по $e$ следующего вида:

$$T(p_1,.., p_k) \circ C, \quad \text{причем} \qquad (2.4)$$

1) $C$ – некоторое производное положение $v$ по $e$;
2) операция "$\circ$" действует согласно формуле (2.2);
3) $T(p_1,.., p_k)$ – часть $T(\alpha, \beta, \gamma, x, y, z)$, описанной формулой (2.1);
4) положения, задаваемые формулой (2.4) есть все производные положения $v$ по $e$, и только они.

Производное представление объекта позволяет, за счет означивания нескольких параметров $\alpha, \beta, \gamma, x, y, z$ нулем, использовать для задания всех трансформаций объекта, удовлетворяющих ограничение $e$, меньшее число параметров, чем при его представлении 6-параметрическим моделированием. Это можно сделать в том случае, когда наложение ограничения на объект можно интерпретировать как изъятие одной или нескольких ротационных или трансляционных степеней свободы объекта.

Например, пусть между двумя плоскостями $A_1$, $B_1$ задано ограничение параллельности. Тогда производным представлением $B_1$ по ограничению параллельности является

$$T(\alpha, x, y, z) \circ I = \begin{pmatrix} R_\alpha & t \\ 0 & 1 \end{pmatrix} \circ I, t = (x, y, z)^T,$$

Действительно, $T(\alpha, x, y, z)$ есть часть $T(\alpha, \beta, \gamma, x, y, z)$ с $\beta = 0, \gamma = 0$. $I$ есть одно из производных положений плоскости по ограничению параллельности, и после применения вращения вокруг $Ox$ на любой угол $\alpha$ и произвольного смещения новое положение все равно будет удовлетворять ограничению параллельности с плоскостью в каноническом положении, то есть быть производным. Наконец, любое производное положение плоскости по ограничению параллельности представимо в указанном виде, так как в нем нормаль к плоскости имеет вид $(1,0,0)$, а любой сдвиг плоскости от начала координат легко получить выбором конкретного $t = (x, y, z)^T$ производного представления.

Заметим, что не для всех объектов и ограничений существует производное представление объекта по ограничению. Например, для точки и ограничения инцидентности кривой это не так. Пусть все производные положения, и только они, задаются видом $T(p_1,.., p_k) \circ C$, где $C = \begin{pmatrix} E & c^T \\ 0 & 1 \end{pmatrix}$, $c = (x_0, y_0, z_0)$ – некоторое производное положение точки по этому ограничению. Это означает, что если $\alpha \in \{p_1,.., p_k\}$, то $R_\alpha \circ (x_0, y_0, z_0)$ всегда тоже лежит на кривой. Очевидно, что для большинства кривых это не так, значит $\alpha \notin \{p_1,.., p_k\}$, аналогично рассуждая, получим $\alpha, \beta, \gamma \notin \{p_1,.., p_k\}$. Если $x \in \{p_1,.., p_k\}$, то $(x_0 + x, y_0, z_0)$ всегда тоже лежит на кривой. Тоже очевидно, что для большинства кривых это не так, значит $x \notin \{p_1,.., p_k\}$, и, в конце концов, $\{p_1,.., p_k\} = \varnothing$. Таким образом, есть единственное производное положение $c = (x_0, y_0, z_0)$, однако, очевидно, кроме $c = (x_0, y_0, z_0)$, есть и другие точки, лежащие на кривой – противоречие.

Другим примером может быть ограничение инцидентности точки и прямой. Допустим, что есть производное представление $T(p_1,.., p_k) \circ C$

прямой по этому ограничению. Оно должно задавать все возможные положения прямой, при которых ограничение инцидентности выполнено, значит, оно должно представлять все положения прямой, проходящей через $(0,0,0)$ с произвольным направляющим вектором. Выберем одно из таких положений $T$ с направляющим вектором $t = (x, y, z)^T$, у которого являются ненулевыми все координаты. Тогда положения $\begin{pmatrix} E & r*t \\ 0 & 1 \end{pmatrix} \circ T, r \in R$ тоже являются производным положением. Очевидно, что такое множество трансформаций, получаемых друг из друга сдвигами, может быть представлено только если среди $p_1,.., p_k$ есть трансляционные переменные. Но любая нефиксированная трансляционная переменная $T(\alpha, \beta, \gamma, x, y, z)$ согласно формуле (2.2) осуществляет сдвиг в направлении координатной оси, а так как в $t = (x, y, z)^T$ все компоненты не равны нулю, такой сдвиг приведет к тому, что прямая перестанет содержать точку $(0,0,0)$, получено противоречие.

Очевидно, что производное представление не единственно, например, для параллельности прямых вместо $\begin{pmatrix} R_\alpha & t \\ 0 & 1 \end{pmatrix} \circ I, t = (x, y, z)^T$, где $C = I$, можно выбрать $\begin{pmatrix} R_\alpha & t \\ 0 & 1 \end{pmatrix} \circ \begin{pmatrix} R_{\alpha 0} & t_0 \\ 0 & 1 \end{pmatrix}, t = (x, y, z)^T$, где $C = \begin{pmatrix} R_{\alpha 0} & t_0 \\ 0 & 1 \end{pmatrix} \neq I$.

**Предложение 2.2** Производными представлениями объекта $v$ по ограничению $e \in E$ являются:

- Для ограничения инцидентности двух одинаковых объектов $A^v(p_1,.., p_k) \circ I$, где $A^v(p_1,.., p_k)$ – частичная параметрическая трансформация, описывающая все автоморфизмы;
- Для ограничения расстояния $d_0$ между плоскостями $A^P(\alpha, y, z) \circ t_x(d_0), t_x(d_0) = (d_0,0,0)$;

- Для ограничения расстояния параллельности плоскостей или прямых (параллельности плоскости и прямой понимается как сонаправленность нормали и направляющего вектора) $T(\alpha, x, y, z) \circ I$;

**Предложение 2.3** В производном представлении $T(p_1,.., p_k) \circ C$ объекта $v$ по бинарному ограничению $e \in E$, снимающему $l$ степеней свободы, число действительных параметров $k$ равно $(6-l)$.

Это утверждение, подтверждаемое рассмотренными выше примерами, следует из того, что число степеней свободы, снимаемых ограничением, равно изменению размерности многообразия решений задачи после наложения ограничения. Так как без требования удовлетворения ограничения $e \in E$ один объект представлялся относительно другого полным набором $\alpha, \beta, \gamma, x, y, z$, а теперь представляется набором $p_1,.., p_k$, то $k = 6 - l$.

**Определение 2.5** Компонентным представлением объекта $v$ по бинарному ограничению $e \in E$ назовем параметрическое представление всех производных положений объекта $v$ по $e$ следующего вида:

$$T(p_1',.., p_{k'}') \circ C \circ A^v(p_1*,.., p_{k*}*), \qquad (2.5)$$

в котором $\{p_1',.., p_{k'}'\} \cap \{p_1*,.., p_{k*}*\} = \varnothing$.

В рассмотренном ранее примере компонентным представлением $B_1$ по ограничению параллельности, очевидно, является $\begin{pmatrix} E & t_x \\ 0 & 1 \end{pmatrix} \circ I \circ \begin{pmatrix} R_\alpha & t_{y,z} \\ 0 & 1 \end{pmatrix}, t_x = (x,0,0)^T, t_{y,z} = (0, y, z)^T$. Таким образом, это представление позволяет выделить среди действительных параметров $\alpha, x, y, z$ производного представления параметры $\alpha, y, z$, описывающие автоморфизмы объекта, и параметр $x$, описывающий нетривиальные сдвиги.

Заметим, что, даже если производное представление объекта $v$ по бинарному ограничению $e \in E$ не существует, компонентное представление может и существовать. Ранее рассматривался вопрос существования

производного представления прямой по ограничению инцидентности с точкой, и было показано, что такого представления нет. Однако, как легко видеть, $\begin{pmatrix} R_\beta \circ R_\gamma & 0 \\ 0 & 1 \end{pmatrix} \circ I \circ \begin{pmatrix} R_\alpha & t_x \\ 0 & 1 \end{pmatrix}, t_x = (x,0,0)^T$ есть компонентное представление. Более того, компонентными представлениями являются

$\begin{pmatrix} R_a \circ R_\gamma & 0 \\ 0 & 1 \end{pmatrix} \circ R_\alpha^\beta \circ \begin{pmatrix} R_\alpha & t_x \\ 0 & 1 \end{pmatrix}$ и $\begin{pmatrix} R_a \circ R_\beta & 0 \\ 0 & 1 \end{pmatrix} \circ R_\alpha^\gamma \circ \begin{pmatrix} R_\alpha & t_x \\ 0 & 1 \end{pmatrix}$, где

$R_\alpha^\beta, R_\alpha^\gamma$ – трансформации, переводящие соответственно ось $Ox$ в ось $Oy$ и ось $Ox$ в ось $Oz$ без изменения положения третьей оси. Таким образом, в компонентном представлении не является единственным не только положение $C$, но и набор параметров.

Другим примером может служить представление плоскости по ограничению инцидентности или расстояния, равного $d_0$, с точкой. Аналогично можно показать, что производного представления не существует, а компонентным представлением является $\begin{pmatrix} R_\beta \circ R_\gamma & 0 \\ 0 & 1 \end{pmatrix} \circ \begin{pmatrix} E & t_x(d_0) \\ 0 & 1 \end{pmatrix} \circ \begin{pmatrix} R_\alpha & t_{y,z} \\ 0 & 1 \end{pmatrix}, t_{y,z} = (0,y,z)^T$, где $d_0 = 0$ для ограничения инцидентности.

Тем не менее, для ограничения инцидентности точки и кривой компонентного представления не существует.

**Определение 2.6** Обобщенным представлением объекта $v$ по бинарному ограничению $e \in E$, снимающему $l$ степеней свободы, аргументами которого являются $v, w$, называют параметрическое представление $Q(p_1^1,..,p_{k1}^1,..,p_1^N,..,p_{kN}^N)$ всех производных положений объекта $v$ по $e$ вида

$T(p_1^1,..,p_{k1}^1) \circ C_1 \circ .. \circ T(p_1^N,..,p_{kN}^N) \circ C_N$, где (2.6)

$\forall i, \{p_1^i,.., p_{ki}^i\} \subset \{\alpha, \beta, \gamma, x, y, z\}$, $i \neq j \leftrightarrow \{p_1^i,.., p_{ki}^i\} \cap \{p_1^j,.., p_{kj}^j\} = \varnothing$,

$\sum_{i \in \{1,..,N\}} k_i = 6 - l$, а $\forall i, C_i$ – константная трансформация.

Очевидно, что производное и компонентное представление являются частными случаями обобщенного представления.

**Предложение 2.4** Пусть существует обобщенное представление $Q_v(p_1,.., p_k)$ объекта $v$ по бинарному геометрическому ограничению $e$. Тогда в любой задаче, включающей такое ограничение $e$ с аргументами $v, w$, с начальным положением объекта $w$ равным $X_w^0$, все положения $X_v$ объекта $v$, удовлетворяющие $e$ с положением $X_w^0$, представляются следующей формулой

$$X_v(p_1,.., p_k) = X_w^0 \circ Q_v(p_1,.., p_k) \qquad (2.8)$$

**Доказательство** Так как $Q_v(p_1,.., p_k)$ – производное представление, то выполнено $e(Q_v(p_1,.., p_k), I)$, осуществим движение $X_w^0$ и в силу геометричности $e$ получим выполненность $e(X_w^0 \circ Q_v(p_1,.., p_k), X_w^0)$, то есть все положения, задаваемые формулой (2.8), удовлетворяют $e$. Обратно, если положение $X_v$ удовлетворяет $e$ с положением $w$ равным $X_w^0$, то выполним движение $(X_w^0)^{-1}$, которое не вырождено в силу унитарности матрицы перехода, и получим удовлетворенность $e((X_w^0)^{-1} \circ X_v, I)$. Значит, $(X_w^0)^{-1} \circ X_v$ есть производное положение, то есть оно задано производным представлением и $(X_w^0)^{-1} \circ X_v = Q_v(p_1',.., p_k')$ для каких-то конкретных значений $(p_1',.., p_k')$, таким образом $X_v = X_w^0 \circ Q_v(p_1',.., p_k')$.

Большинство ограничений, встречающихся на практике, являются геометрическими, например, геометрическими являются все ограничения, перечисленные в приложении 1. Примером ограничения, не являющегося

геометрическим, может служить $e(v_1, v_2)$, которое удовлетворено, если равна единице площадь треугольника, образованного точками $v_1, v_2$ и центром глобальной системы координат. Очевидно, что $e((1,0,0),(0,2,0))$ выполнено, но $e((1+x,0,0),(x,2,0))$ для $x \neq 0$ не выполнено.

**Предложение 2.5** Если существует представление $X_v(p_1,..,p_k) = X_w^0 \circ Q_v(p_1,..,p_k)$ из предыдущего предложения, удовлетворяющее $e$ с объектом $w$ в положении $X_w^0$, и все положения $w$ описываются каким-то представлением $Q_w(r_1,..,r_s)$, то все положения $v$, удовлетворяющие $e$, описываются представлением

$$X_v(r_1,..,r_s, p_1,..,p_k) = Q_w(r_1,..,r_s) \circ X_w^0 \circ Q_v(p_1,..,p_k) \qquad (2.9)$$

**Доказательство** Полностью аналогично предыдущему с рассмотрением движения $T_w(r_1,..,r_s)$.

**Предложение 2.6** Пусть существует производное представление $Q_v(p_1,..,p_k)$ объекта $v$ по бинарному геометрическому ограничению $e$, и дана задача $Z = (\widetilde{V},(E,a,A),G,S_0,F,\varepsilon)$, в которой $e \in E$, $A(e,1) = v$ и $v \in g \in G$. Тогда для такой задачи при генерации системы уравнений методом декартова моделирования можно уменьшить число уравнений и переменных так, чтобы в полученной системе не было уравнений для ограничения $e$ и число переменных для тела $g$ уменьшилось на число степеней свободы, снимаемых $e \in E$.

**Доказательство** Все положения $v$, удовлетворяющие $e$ с положением объекта $w$, равным $Q_w(r_1,..,r_s) \circ X_w^0$, описываются тогда формулой (2.9) $Q_w(r_1,..,r_s) \circ X_w^0 \circ Q_v(p_1,..,p_k)$ по предыдущему предложению. Тогда можно описать трансформацию тела $g$ как

$$T_g(r_1,..,r_s, p_1,..,p_k) = Q_w(r_1,..,r_s) \circ X_w^0 \circ Q_v(p_1,..,p_k) \circ (X_v^0)^{-1}, \qquad (2.10)$$

при этом будет выполнено $e(T_g(r_1,..,r_s,p_1,..,p_k)\circ X_v^0, Q_w(r_1,..,r_s)\circ X_w^0)$, то есть такие трансформации описывают только положения объектов, удовлетворяющие $e$. Более того, согласно предложению 2.5, все положения тел такие, что $e$ удовлетворено, имеют указанные выше представления. Значит, можно исключить из набора переменных декартова моделирования все шесть переменных тела $g$, и добавить переменные $p_1,..,p_k$, $k<6$, подставив во все уравнения новые выражения для координат объектов $g$, выраженные через $T_g$. Число переменных при этом уменьшится на $6-k=l$ – число степеней свободы, снимаемых $e$.

Итак, тем самым представлен способ улучшения моделирования задачи параметрического проектирования. В частных случаях его можно еще улучшить.

**Предложение 2.7** Пусть существует компонентное представление $T(p_1,..,p_m)\circ C\circ A(p_m,..,p_k)$ объекта $v$ по бинарному геометрическому ограничению $e$, и дана задача $Z=(\tilde{V},(E,a,A),G,S_0,F,\varepsilon)$, в которой $e\in E$, $A(e,1)=v$ и $v\in\{v\}=g\in G$. Тогда для такой задачи при генерации системы уравнений методом декартова моделирования можно уменьшить число уравнений и переменных так, чтобы в полученной системе не было уравнений для ограничения $e$ и число переменных для тела $g$ уменьшилось до $m$.

**Доказательство** Выпишем частный вид формулы (2.10), полученный подстановкой конкретного вида компонентного представления: $T_g(r_1,..,r_s,p_1,..,p_k)=Q_w(r_1,..,r_s)\circ X_w^0\circ T_v(p_1,..,p_m)\circ C\circ A^V(p_m,..,p_k)\circ(X_v^0)^{-1}$. Положим $p_m,..,p_k$ равными нулю, тем самым $T_v(p_1,..,p_m)\circ C$ представит все положения $v$ с точностью до автоморфизмов. Так как других объектов, кроме $v$, в теле $g$ нет, то представление

$$T_g(r_1,..,r_s,p_1,..,p_m) = Q_w(r_1,..,r_s) \circ X_w^0 \circ T_v(p_1,..,p_m) \circ C \circ (X_v^0)^{-1}$$

описывает все положения $g$ (с точностью до автоморфизмов), в которых $e$ выполнено, и его переменные можно заменить на $m$ переменных $p_1,..,p_m$.

Рассмотрим пример, в котором два тела $A$ и $B$ связаны ограничением параллельности плоскостей $A_1$ и $B_1$ (см. рис. 2.1). Тогда положение тела $B$ можно описать с помощью всего четырех переменных, так как при любом положении тела $A$ тело $B$ должна быть расположено так, чтобы его грань $B_1$ была параллельна грани $A_1$. Значит, $B$ можно лишь вращать вокруг общего перпендикуляра к плоскостям $A_1$, $B_1$ и смещать в произвольном направлении. Таким образом, тело $B$ относительно тела $A$ имеет одну ротационную и три трансляционных степени свободы.

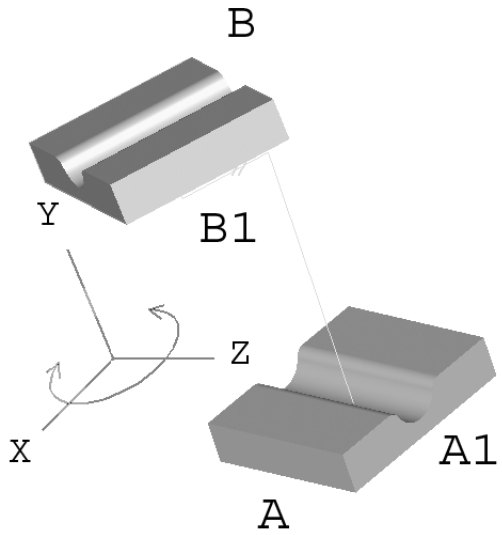


Рисунок 2.1. Взаимные степени свободы тел с учетом параллельности

Действительно, пусть начальные положения тел $A$ и $B$ есть $X_A^0, X_B^0$ соответственно. Пусть $Q_A(r_1,..,r_s) \circ X_A^0$ описывает все положения $A$. Тогда $Q_A(r_1,..,r_s) \circ X_A^0 \circ \begin{pmatrix} R_\alpha & t(x,y,z) \\ 0 & 1 \end{pmatrix} \circ (X_B^0)^{-1}$ опишет все положения $B$ с помощью четырех дополнительных параметров $(\alpha, x, y, z)$. Если же известно, что в задаче нет ограничений на отличные от $B_1$ объекты $B$, то можно

сократить число дополнительных параметров до одного представлением $Q_A(r_1,..,r_s) \circ X_A^0 \circ \begin{pmatrix} E & t(x) \\ 0 & 1 \end{pmatrix} \circ (X_B^0)^{-1}$.

Генерируемую систему уравнений можно еще уменьшить, если учитывать несколько ограничений, заданных между одними и теми же телами. В таком случае, трансформацию $T_g$ одного тела относительно другого тоже, как правило, можно представить в виде, заданным формулой (2.10). Задача определения взаимного положения пары тел, связанных несколькими ограничениями, не является тривиальной [86], однако ее результат может быть найден заранее и не влияет на время вычисления решения системы уравнений.

**Определение 2.8** Обобщенным представлением тела $g^* = \{v_1^*,..,v_{n^*}^*\}$ относительно тела $g = \{v_1,..,v_n\}$, объекты которого имеют координаты $X_1,..,X_n$ соответственно, по такому набору ограничений $\{e_1,..,e_m\}$, что $\forall i \in \{1,..,m\}, \forall j \in \{1,..,a(e_i)\}, A(e_i,j\} \in g \cup g^*$, называется параметрическое представление трансформаций $Q_{g^*}(p_1,..,p_k)$ тела $g^*$ относительно $g$ такое, только для них и только для них выполнено

$$\forall i \in \{1,..,m\}, e_i(Q_g(r_1,..,r_s) \circ Q_{g^*}(p_1,..,p_k), Q_g(r_1,..,r_s)),$$

где $Q_g(r_1,..,r_s)$ – параметрическое представление трансформаций $g$. При этом количество $k$ параметров называют мощностью представления, а $(6-k)$ – числом снимаемых степеней свободы.

Это определение является обобщением определения 2.6 об обобщенном представлении на случай нескольких ограничений, заданных между двумя телами. Рассмотрим без доказательства его частный случай для $n=2, n^*=2, m=2$ и $\forall i \in \{1,2\}, a(e_i)=2 \wedge A(e_i,1\} \in g^* \wedge A(e_i,2\} \in g$

**Предложение 2.8** Обобщенными представлениями тела $g^* = \{v_1^*, v_2^*\}$ по набору ограничений $\{e_1, e_2\}$ относительно $g = \{v_1, v_2\}$ являются:

- Если оба объекта $v_1{*}, v_2{*}$ есть прямые, а $e_1, e_2$ есть ограничения инцидентности прямых, то в случаях, когда углы или расстояния между прямыми внутри $g$ и $g*$ не равны, задача не имеет решений. Если же они равны и прямые $v_1{*}, v_2{*}$ не параллельны, то такое представление не имеет действительных параметров; если они параллельны, то им является $R(\alpha, v_1) \circ \begin{pmatrix} E & t_x \\ 0 & 1 \end{pmatrix} \circ R(\alpha, v_1{*})^{-1}$, где $R(\alpha, v)$ – константная трансформация, переводящая ось $Ox$ в $v$;
- Если оба объекта $v_1{*}, v_2{*}$ есть плоскости, а $e_1, e_2$ есть ограничения инцидентности между плоскостями, то в случаях, когда углы или расстояния между плоскостями внутри $g$ и $g*$ не равны, задача не имеет решений. Если же они равны и плоскости $v_1{*}, v_2{*}$ не параллельны, то таким представлением является $R_1 \circ \begin{pmatrix} E & t_x \\ 0 & 1 \end{pmatrix} \circ R_2$, где $R_1, R_2$ – некоторые константные трансформации. В случае параллельных плоскостей представлением является $\overline{R}(\alpha, v_1) \circ \begin{pmatrix} R_a & t_{y,z} \\ 0 & 1 \end{pmatrix} \circ \overline{R}(\alpha, v_1{*})^{-1}$, где $\overline{R}(\alpha, v)$ – константная трансформация перевода плоскости $Oyz$ в $v$.

Итак, идея метода остовного моделирования заключается в том, чтобы вместо полного набора из шести переменных $\alpha, \beta, \gamma, x, y, z$ для каждого тела использовать меньший набор за счет учета наложенных ограничений, и представлять уравнения специальным образом через такие наборы переменных. Идея представления тел меньшим набором параметров за счет исключения из рассмотрения некоторых степеней свободы возникала ранее в задачах анализа кинематики [68]. Тем не менее, в этой работе не описаны условия, при которых исключение ограничения и сокращенное представление объекта можно осуществить в рамках моделирования задачи

параметрического проектирования, и вид возникающих при этом параметрических трансформаций одного объекта относительно другого.

**Определение 2.9** Пара тел $g*$ и $g$ в задаче $Z$ называется подходящей для остовного моделирования, если

1) Все ограничения, включающие хоть один объект из $g*$ и $g$, являются геометрическими;
2) Существует обобщенное представление тела $g*$ относительно тела $g$ по набору ограничений $\{e_1,..,e_k\}$, все аргументы которых содержатся в $g*\cup g$.

Эти условия являются достаточными для того, чтобы расширить предложение 2.6 для случая обобщенных представлений не только объектов по одному ограничению, но произвольных тел по произвольным наборам ограничений.

### 2.2.2 Алгоритмическое описание метода остовного моделирования

Указанные в предложениях 2.2 и 2.8 случаи существования обобщенного представления одних тел по отношению к другим не исчерпывают всех возможных ситуаций. Каждый такой случай можно выразить с помощью *шаблона остовного моделирования*. Это специальная сущность, которая представляет собой пару тел, связанную набором конкретных геометрических ограничений, для которого можно осуществить обобщенное представление одного тела по отношению к другому. Шаблоны остовного моделирования можно использовать для выявления пар тел, удовлетворяющих определению 2.9. С увеличением числа описанных случаев обобщенного представления одних тел по отношению к другим можно расширять список таких шаблонов.

Рассмотрим граф, вершинами которого являются тела задачи, а ребро между парой вершин существует, если соответствующая пара тел является

подходящей для остовного моделирования. Остовное дерево (в общем случае – остовный лес) в этом графе, с каждым ребром которого ассоциирован некоторый шаблон остовного моделирования, может служить основой для моделирования с сокращением числа переменных и уравнений [11]. Действительно, выберем корень дерева $r$, и ориентируем ребра в остове от листьев к корню, для каждой вершины $v \neq r$ тем самым задана функция определения отца $p(v)$. Можно ее обобщить с помощью введения функции $P(v,k)$ такой что $P(v,0) = v, P(v,k) = p(P(v,k-1))$. Значение $k : P(v,k) = r$ обозначим $l(v)$.

Дуга “отец-сын” показывает, что возможные трансформации “сына” определяются относительно трансформаций “отца” согласно формуле вида (2.10). Так как “отец” тоже может быть параметризирован относительно другого тела, то общее выражение для трансформации тела имеет вид

$$T_v(..) = Q_{P(v,l(v))}(..) \circ C_{l(v)} .. \circ Q_{P(v,0)}(..) \circ C_0 , \qquad (2.11)$$

где $C_i$ – какие-то константные трансформации.

Количество используемых для такого представления переменных уменьшается на число степеней свободы, которые снимают все шаблоны остовного дерева. Количество уравнений тоже существенно снижается, поскольку все ограничения, которые включены в обобщенные представления пар тел, попавших в остовное дерево, будут автоматически выполнены и не требуют генерации уравнений.

Остовное моделирование задачи параметрического проектирования требует выполнения дополнительных операций: построения оптимального остовного дерева (с генерацией сокращенной системы уравнений), а также вычисления невязок и градиентов сгенерированных уравнений.

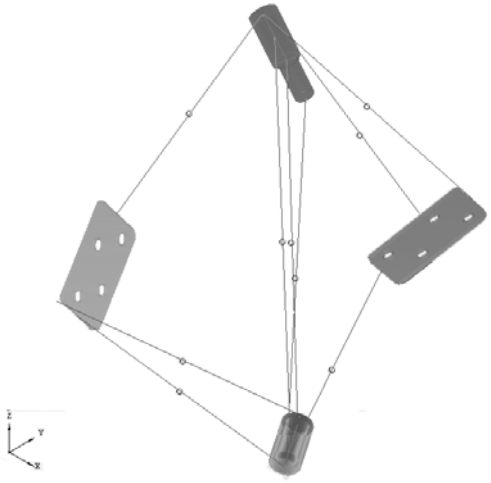

Рисунок 2.2. Исходная модель из 4 тел и 9 ограничений

На первом шаге построения оптимального остовного дерева создается мультиграф, вершинами которого являются геометрические тела, а ребрами – ограничения, которыми связаны объекты в этих телах. Каждое ребро мультиграфа помечается числом, равным количеству степеней свобод, снимаемых соответствующим ограничением (см. рис. 2.2 и 2.3). Так, например, совпадение плоскостей снимает три степени свободы, а расстояние между точкой и плоскостью – только одну.

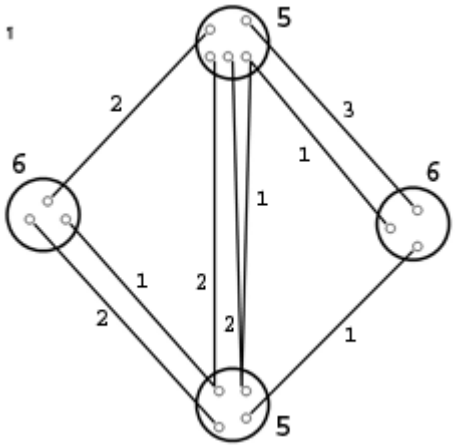


Рисунок 2.3. Мультиграф с помеченными ребрами

Затем происходит анализ попарных взаимосвязей геометрических тел с учетом всех заданных между ними ограничений. Например, на рис. 2.3 для пары из верхнего и нижнего тел требуется рассмотреть три ограничения, а для пары из нижнего и правого тел – только одно. В процессе анализа происходит поиск в базе предопределенных шаблонов остовного

моделирования, каждый из которых определяет для пары тел, какие из переменных $\alpha, \beta, \gamma, x, y, z$ трансформации одного из тел можно зафиксировать при позиционировании его по отношению к другому телу. Количество таких переменных называется весом шаблона. Он не превышает суммы степеней свобод, снимаемых всеми входящими в шаблон ограничениями, равенство достигается тогда, когда все ограничения снимают степени свободы независимо. После проведения анализа мультиграф превращается в *граф взаимосвязей тел:* для каждой пары тел набор ребер-ограничений между ними заменяется одним ребром, которому приписывается вес, равный весу найденного шаблона (см. рис. 2.4).

Набор всех возможных шаблонов, описывающих взаимосвязи двух тел, хоть и объемен, но конечен; на практике же достаточно рассмотреть несколько десятков наиболее эффективных шаблонов схожей структуры. Если набор из $N$ ограничений между двумя телами не совпадает ни с каким шаблоном, то в качестве результата выбирается шаблон, набор ограничений $M$ которого является подмножеством $N$. Это означает, что при генерации системы уравнений ограничения из $M$ не будут генерироваться, а ограничения из $M \setminus N$ будут обрабатываться обычным образом.

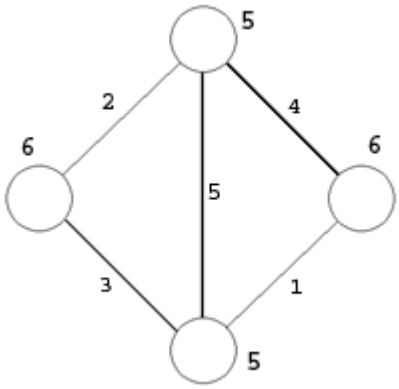


Рисунок 2.4. Максимальное остовное дерево в помеченном графе

С целью минимизации количества переменных и уравнений в генерируемой системе уравнений необходимо найти в построенном графе взаимосвязей тел остов максимального суммарного веса (см. рис. 2.4). Это

можно эффективно сделать, например, с помощью алгоритма Борувки, имеющего сложность $O(n\log(n))$, где $n$ – число ограничений в модели [7]. Однако, учитывая тот факт, что веса ребер графа могут принимать лишь значения $\{1,..,6\}$, так как они указывают снимаемое число степеней свободы в трехмерной задаче, максимальное остовное дерево можно построить за $O(n)$.

Для ограничений, которые вошли в шаблоны, включенные в построенное остовное дерево, уравнения генерировать не надо – такие ограничения автоматически выполнятся за счет механизма параметрических трансформаций. В результате система уравнений будет построена только для ограничений исходной модели, которые замыкают циклы в графе тел и ограничений, причем эта система уравнений будет иметь значительно меньшее число переменных.

Корректная генерация уравнений для незадействованных в остове ограничений требует создания специальных объектов, представляющих уравнения. Для решения задачи необходимо такое представление уравнения, которое позволяет вычислять для любого текущего значения переменных невязку уравнения и градиент этой невязки, так как именно эти величины необходимы для решения системы уравнений методом Ньютона. Согласно формуле (2.11), такие вычисления требуют перемножения набора параметрических и константных трансформаций. При этом текущие означивания переменных подставляются в параметрические трансформации, после чего происходит вычисление невязки и градиента перемножением конкретных константных трансформаций.

Представление трансформаций “сына” через трансформации “отца” требует вычисления константных трансформаций $C_i$ из формулы (2.11). Для производного или компонентного представления эти трансформации определяются через каноническое положение объектов (см. предложения 2.2 и 2.8, рис. 2.5). Они вычисляются в процессе генерации уравнений и не требуют дополнительных затрат при вычислении невязок и градиентов.

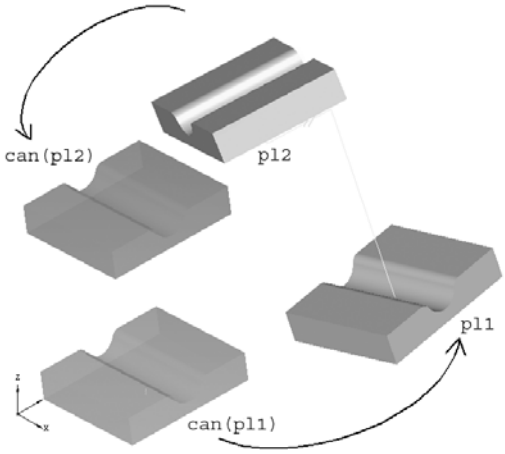


Рисунок 2.5. Определение константных трансформаций тел

Вычисление невязки и градиента невязки одного уравнения при остовном моделировании сложнее, чем при декартовом моделировании. Например, для вычисления невязки уравнения, построенного по ограничению $e$, аргументы которого лежат в телах $\{g_1,..,g_{a(e)}\}$, требуется выполнить порядка $\sum_{i\in\{1,..,a(e)\}} l(g_i)$ операций умножения матриц константного размера, пройдя по всем путям от тел $\{g_1,..,g_{a(e)}\}$ до корня дерева. Эта длина может быть оценена сверху лишь как $O(n)$, где $n$ – число тел в модели, в то время как для обычного декартова моделирования требуется лишь константное время. Если же одновременно с этим вычислять градиенты уравнения, то сложность вырастет до $O(k*n)$, где $k$ – число переменных. Итак, если вычислять невязки и градиенты всех ограничений независимо, то сложность определения якобиана для метода Ньютона составит $O(n*k*m)$, где $m$ – число уравнений.

Тем не менее, общая сложность вычислений будет не более чем квадратичной при определенной организации этих вычислений. Действительно, вычисление невязки и ее градиента сводится, согласно формулам дифференцирования композиции функций, к вычислению параметрической трансформаций тела (на текущем означивании переменных) и ее производных. Но параметрическая трансформация тела $g$

вычисляется через параметрическую трансформацию “отца” тела $g$, причем для вычисления самой трансформации нужно константное время, а для вычисления производных – время $O(k)$.

Значит, если вычисления параметрических трансформаций и их производных производить от корня дерева к его вершинам, используя при каждом новом вычислении результаты, полученные на предыдущем шаге, то на переход от одной вершины к ее сыну всего потратится не более $O(k)$ операций. Тогда, обойдя весь остов, потратится не более $O(n*k)$, то есть не более квадратичного от входа задачи параметрического проектирования времени.

Так как для представления якобиана требуется квадратичный объем памяти, то и его вычисление не может иметь менее чем квадратичную сложность, поэтому при переходе от метода декартова моделирования к методу остовного моделирования теоретическая оценка сложности не ухудшается. На практике, тем не менее, якобиан для метода остовного моделирования будет менее разреженным, так как любое уравнение будет иметь ненулевые производные по всем переменным, входящим в представление его трансформации по формуле (2.11). Тем не менее, так как при решении системы нелинейных уравнений методом Ньютона на каждой итерации самой трудоемкой операцией является кубическое по времени решение линейной системы уравнений, эффект ускорения будет достигнут за счет уменьшения размера системы уравнений.

## 2.3 Схема использования метода базисов Гребнера

Наиболее известным символьным методом решения систем полиномов является построение базиса Гребнера – специальной системы полиномов, алгебраически эквивалентной исходной. Надо заметить, что уравнения, генерируемые 12-параметрическим декартовым моделированием для большинства практически важных ограничений (например, для углов, инцидентностей и расстояний между точками, прямыми и плоскостями)

являются квадратными. Другое, 6-параметрическое декартово моделирование, генерирует согласно формуле (2.3) полиномиальные формы от косинусов и синусов углов Эйлера, которые могут быть представлены в дробно-рациональном виде с помощью универсальной тригонометрической подстановки $\sin(x) = 2t/(1+t^2), \cos(x) = (1-t^2)/(1+t^2)$, где $t = tg(x/2)$. После подстановки этих форм в уравнение часто можно умножением избавиться от знаменателей и тоже получить полином. Наконец, остовное моделирование использует для представления одних тел относительно других умножение на константные и параметрические матрицы вида (2.1) с некоторыми переменными, положенными тождественно равными нулю, то есть тоже генерирует полиномиальные уравнения.

Базисом Гребнера системы полиномов $P = \begin{cases} p_1(x_1,..,x_n) \\ .. \\ p_m(x_1,..,x_n) \end{cases}$ по некоторому фиксированному упорядочиванию мономов называется алгебраически эквивалентная ей система полиномов $G = \begin{cases} g_1(x_1,..,x_n) \\ .. \\ g_k(x_1,..,x_n) \end{cases}$, обладающая некоторыми специальными свойствами [3]. К главным преимуществам метода базисов Гребнера следует отнести его способность сводить решение системы полиномиальных уравнений от многих переменных, имеющую конечное число корней, к решению существенно более простой системы уравнений. Эта система, являющаяся базисом Гребнера исходной системы, часто имеет треугольную по вхождению переменных структуру. Для ее решения достаточно сначала решить единственное уравнение от одной переменной, подставить его решения в другие уравнения и, получив систему такого же вида меньшего размера, продолжить вычисления по аналогии. При этом можно подставлять на каждом шаге все решения уравнения от одной переменной в оставшиеся полиномы, получая, таким образом, все решения

исходной задачи, что невозможно при применении традиционных численных методов.

К сожалению, метод базисов Гребнера имеет и существенные недостатки. Так как при поиске базиса Гребнера не должны происходить ошибки округления при операциях с коэффициентами, то вместо вещественных коэффициентов применяются целые коэффициенты, которые быстро растут в процессе вычислений. Известные алгоритмы вычисления базисов Гребнера являются не менее чем экспоненциальными, что резко снижает их применимость на практике. Скорость и результат вычисления базиса Гребнера может меняться в зависимости от различных факторов [10]: упорядочивания переменных, упорядочивания мономов, использования различных техник (в частности, использования критериев Бухбергера тривиальности критических пар полиномов), и т. п.

С учетом отмеченных выше свойств эффективное использование базисов Гребнера возможно при обработке задач по следующей схеме. После выполнения геометрической декомпозиции полученные подзадачи классифицируются на разные типы. Среди этих типов есть как специальные, для решения каждого из которых будет применяться некоторый заранее заданный алгоритм, так и общий, включающий все достаточно сложные задачи, для решения которых применяется метод Ньютона. Такая классификация позволяет для решения подзадач использовать наиболее подходящие для этого методы, в частности, аналитические. Специально обрабатывать разумно, как минимум, все двухмерные задачи $I$-типа, $V$-типа и $E$-типа [44], то есть:

1) подзадачи с двумя одинокими объектами и одним ограничением;
2) подзадачи с одним телом, одним одиноким объектом и двумя ограничениями;
3) подзадачи с двумя телами и тремя ограничениями.

Некоторые из задач, являющихся аналитически неразрешимыми, тем не менее, достаточно малы, и их можно эффективно решать с помощью методов базиса Гребнера. В качестве примера двухмерной аналитически неразрешимой задачи небольшого размера можно привести задачу размещения двух тел с тремя ограничениями расстояния между точкой и прямой между ними [75].

Опишем более подробно предлагаемый процесс решения подзадач с помощью метода базисов Гребнера. Будем считать, что набор объектов и ограничений в подзадаче известен, но могут варьироваться значения параметров ограничений и расположения объектов внутри тел. Такую задачу можно представить одной параметрической системой полиномиальных уравнений. Эта система включает как обычные переменные (трансформации размещаемых тел), так и переменные-параметры, к которым относятся координаты объектов внутри твердых тел и параметры ограничений.

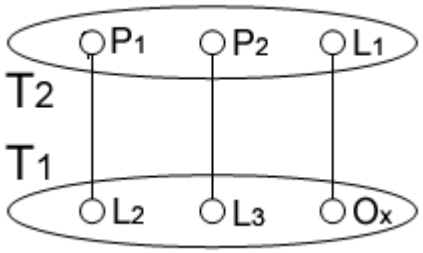


Рисунок 2.6. Задача, решаемая с помощью метода базисов Гребнера

Например, рассмотрим задачу о расположении двух твердых тел $T_1$, $T_2$, связанных одним ограничением угла между прямыми и двумя ограничениями расстояния между точкой и прямой (см. рис. 2.6). За счет выбора системы координат первого тела можно расположить его так, что прямая, связанная ограничением угла, совпадала с осью $Ox$. Тогда координаты второго тела, которые составляют величины `x` и `y` сдвига его локальной системы координат и величины `s` и `c`, равные синусу и косинусу наклона этой локальной системы координат, будут составлять набор обычных переменных системы уравнений. В число переменных-параметров

войдут координаты p11, p12, p21, p22 точек $P_1, P_2$ внутри второго тела, синусы и косинусы s1, c1, s2, c2, s3, c3 прямых $L_1, L_2, L_3$, параметры ограничений угла a1 и расстояний d2 и d3 (из них вычтены известные расстояния от прямых $L_2, L_3$ до начала их локальной системы координат). Параметрическая система полиномиальных уравнений будет иметь вид:

(x+c*p11+s*p12)*c2+(y-s*p11+c*p12)*s2-d2=0
(x+c*p21+s*p22)*c3+(y-s*p21+c*p22)*s3-d3=0
s*s1+c*c1-a1=0
s^2+c^2-1=0

Для этой системы целесообразно заранее найти ее базис Гребнера при лексикографическом упорядочивании переменных, в котором все переменные-параметры обладают меньшим весом, чем обычные переменные, которые упорядочены, например, таким образом: s > c > x > y. В таком случае, базис Гребнера будет иметь треугольный вид относительно обычных переменных:

F1(p11, p12, p21, p22, s1, c1, s2, c2, s3, c3, d1, d2, a1; y)=0
F2(p11, p12, p21, p22, s1, c1, s2, c2, s3, c3, d1, d2, a1; y, x)=0
F3(p11, p12, p21, p22, s1, c1, s2, c2, s3, c3, d1, d2, a1; y, x, c)=0
F4(p11, p12, p21, p22, s1, c1, s2, c2, s3, c3, d1, d2, a1; y, x, c, s)=0

При решении конкретной подзадачи все переменные-параметры обретут свои численные значения, и задача сведется к решению конкретной треугольной системы полиномиальных уравнений, что можно эффективно осуществить с помощью методов решения полиномиальных уравнений от одной переменной: метода отделения корней и одномерного метода Ньютона.

Например, последний из полиномов F1 – F4 будет равен

F4= s * (c1*c3*c2*p12-p11*s1*c2*c3+c1*s3*p21*c2+c3*p21*s1*c2 +s3*p22*s1*c2-c1*c3*p22*c2-s1*p12*c3*s2-p11*c1*c3*s2) + (s2*c3*p12*a1-c3*c1*d2+s2*c3*y*c1-c3*c2*p21*a1-p22*c2*s3*a1+p11*c3*c2*a1-c1*c2*y*s3 +c1*c2*d3),

то есть будет линеен по вхождению `s`, так что его решение можно будет легко найти аналитически. Все остальные полиномы F1 – F4 тоже являются линейными или квадратичными по вхождению своей старшей переменной, но они довольно объемны и потому здесь не приводятся.

Заметим, что для обработки аналитически разрешимых задач параметрического проектирования можно применять готовые явные формулы вместо численных методов, которые могут внести значительно большую ошибку округления. При решении же более сложных задач, для которых используются методы базисов Гребнера, не всегда можно построить явные формулы. Тем не менее, решение уравнений от одной переменной значительно более эффективно и позволяет получить меньшую погрешность в определении решения.

Предложенный выше способ не использует никакие экспоненциальные алгоритмы во время решения задачи, наоборот, он уменьшает это время за счет сведения системы уравнений к последовательности уравнений от одной переменной. Недостатком предложенной схемы является увеличение объема кода решателя за счет реализации специальных методов решения подзадач и процесса классификации. Кроме того, метод не может быть применим для решения больших подзадач, так как количество разных типов подзадач растет экспоненциально с ростом их размера, и так же быстро увеличиваются временные затраты метода поиска базиса Гребнера.

## 2.4 Выводы

В рамках разработки алгебраических методов решения задач параметрического проектирования получены следующие результаты: разработан новый метод остовного моделирования и предложена схема использования базисов Гребнера в решателе задач параметрического проектирования.

Метод остовного моделирования является представителем класса методов алгебраического моделирования. Он основан на декартовом

моделировании, но позволяет генерировать систему уравнений меньшего размера за счет специального представления положения тел, связанных одним или несколькими ограничениями. Это представление определяет все положения одного тела относительно другого тела при условии, что заданные между ними ограничения выполняются. Таким образом, из системы уравнений можно исключить эти ограничения, а положение одного тела по отношению к другому представить меньшим набором параметров. Это представление положения тел возможно при определенных условиях на заданные между ними ограничения, которые выполняются для многих практически важных ограничений.

Схема использования методов базиса Гребнера для решения задач параметрического проектирования, предложенная автором, нацелена на то, чтобы не допустить экспоненциальных вычислений во время решения этой задачи, и в то же время использовать специальный вид базиса Гребнера системы полиномиальных уравнений. Для этого предлагается использовать классификацию подзадач и для аналитически неразрешимых задач небольшого размера заданного вида использовать заранее вычисленный базис Гребнера для специального упорядочивания переменных и параметров системы полиномиальных уравнений. После подстановки в базис Гребнера фактических значений параметров системы он примет треугольный вид, и решение системы сведется к решению набора уравнений от одной переменной. В целом такая схема является применимой для узкого класса задач параметрического проектирования, однако при их решении она может повысить точность и надежность вычислений.

# 3. Методы геометрической декомпозиции задач параметрического проектирования

## 3.1 Начальные определения

Решение задачи параметрического проектирования представляет собой сложную проблему, и основной применяемый для этого метод, который заключается в моделировании геометрической задачи с помощью алгебраической системы уравнений и решении ее методом Ньютона, имеет кубическую вычислительную сложность. В связи с этим возникает проблема ускорения решения задачи параметрического проектирования с помощью методов декомпозиции, то есть разбиения задачи на набор подзадач меньшего размера, последовательное решение которых позволяет решить исходную задачу. Если метод декомпозиции имеет временную сложность меньшую, чем кубическая, то его применение, как правило, позволяет сократить общее время решения задачи параметрического проектирования. Определим формально процесс декомпозиции.

**Определение 3.1** Задача $Z' = (\tilde{V}', \tilde{E}', G', S_0', F', \varepsilon)$ называется следствием задачи $Z = ((V,P),(E,a,A),G,S_0,F,\varepsilon)$ по множеству тел $G^* \subset G, G^* \neq \varnothing$, что обозначается $Z \xrightarrow[G^*]{} Z'$, если:

1) $G' = (G \setminus G^*) \cup g'$, $g' = \bigcup_{g \in G^*} g$;

2) $\tilde{E}' = (E', a, A), E' = \{e \in E \mid \exists i \le a(e), \overline{G}(A(e,i)) \in G \setminus G^*\}$;

3) $\tilde{V}' = (V', P)$, $V' = \{v \in V \mid \exists e \in E' \exists i \le a(e) : A(e,i) = v\}$;

4) $S_0' = \{< v, s > \in S_0 \mid v \in V'\}$;

5) $F' : S(V') \times S(V') \to [0, \infty)$, такая что $F'(S_1', S_2') = F(S_1' \cup S_\Delta, S_2' \cup S_\Delta)$, где $S_\Delta = \bigcup_{v \in V \setminus V'} < v, S_0(v) >$.

**Определение 3.2** Элементарной декомпозицией задачи $Z=(\tilde{V},\tilde{E},G,S_0,F,\varepsilon)$ по множеству тел $G^*\subset G$ называют пару задач $(Z_1,Z_2)$ такую, что $Z|_{G^*}$ $Z_1$ и $Z\xrightarrow[G^*]{}Z_2$, этот факт обозначают $Z\prec_{G^*}(Z_1,Z_2)$.

Итак, элементарная декомпозиция задачи параметрического проектирования $Z$ состоит в выделении некоторого подмножества $G^*$ тел этой задачи, после чего исходная задача разбивается на $Z_1$ – подзадачу $Z$ по $G^*$ – и $Z_2$ – следствие $Z$ по $G^*$.

**Определение 3.3** Последовательность $(Z_1,..,Z_c)$, $c\geq 2$ задач параметрического проектирования называют декомпозицией $Z$, что обозначается $Z\prec(Z_1,..,Z_c)$:

1) При $c=2$, если она является элементарной декомпозицией $(Z_1,Z_2)$ по некоторому подмножеству тел $G^*$;
2) При $c>2$, если $\exists i\in\{1,..,c-1\},\exists Z^*\prec(Z_i,Z_{i+1})$ – элементарная декомпозиция, такая что $(Z_1,..,Z_{i-1},Z^*,Z_{i+2},..,Z_c)$ – декомпозиция задачи $Z$.

**Определение 3.4** Методом декомпозиции на некотором классе задач параметрического проектирования $U$ будем называть отображение $D$, которое любой задаче $Z=(\tilde{V},\tilde{E},G,S_0,F,\varepsilon)\in U$ ставит в соответствие некоторое множество $P\subset 2^G$ подмножеств ее тел.

**Определение 3.5** Говорят, что метод декомпозиции $D$ допускает элементарную декомпозицию $Z\prec_{G^*}(Z_1,Z_2)$, если $G^*\in D(Z)$. Метод декомпозиции $D$ допускает декомпозицию $Z\prec(Z_1,..,Z_c)$, если можно построить эту декомпозицию из элементарных по правилам из определения 3.3, причем $D$ допускает каждую элементарную декомпозицию.

**Определение 3.6** Решение $T'\in T(Z')$ подзадачи $Z'$ задачи $Z$ называется частью решения $T\in T(Z)$ задачи $Z$, что обозначается $T'\subseteq T$, если $T'(S_0')\subseteq T(S_0)$, где $S_0$ – начальное означивание $Z$, а $S_0'\subseteq S_0$ – начальное означивание $Z'$.

**Предложение 3.1** Если $Z = ((V,P),(E,a,A),G,S_0,F,\varepsilon)$, $Z' = ((V',P),(E',a,A),G',S_0',F',\varepsilon)$, $Z \mid Z'$, тогда $T \in T(Z) \Rightarrow \exists T' \subseteq T, T' \in T(Z')$.

**Доказательство** $T \in T(Z) \Rightarrow \exists S \in S(V), S = T(S_0)$, причем $\forall e \in E$, $e(S(A(e,1)),..,S(A(e,a(e)))) \le \varepsilon$. Пусть $S' = \{< v,s > \in S \mid v \in V'\}$ тогда $\forall e' \in E'$, $e'(S'(A(e,1)),..,S'(A(e,a(e)))) \le \varepsilon$. Положим $T' = T_{S'}$ – свойство доказано, так как $S' \subseteq S$ по построению.

Итак, для любого решения задачи найдется решение подзадачи, являющееся его частью. Обратное, вообще говоря, не верно.

**Определение 3.7** Элементарная декомпозиция $Z \prec_{G^*} (Z', Z^*)$ корректна, если $T(Z) \ne \varnothing \to \forall T' \in T(Z') \exists T \in T(Z) : T' \subseteq T$. Декомпозиция задачи $Z$ корректна, если существует ее построение из элементарных декомпозиций, где все использованные элементарные декомпозиции были корректны. Метод декомпозиции $D$ называется безусловно корректным (для краткости – просто корректным), если он допускает только корректные декомпозиции, иначе он называется условно корректным.

Как видно из определения, декомпозиция корректна, если любое решение подзадачи может быть дополнено до решения всей задачи. Формально говоря, методом декомпозиции может быть любой метод, сопоставляющий каждой задаче параметрического проектирования некоторое подмножество ее тел. Однако наибольший интерес представляют те методы, которые допускают корректные декомпозиции, поскольку некорректные декомпозиции могут привести к невозможности решить исходную задачу.

**Определение 3.8** Задачу, решение которой единственно с точностью до общего движения объектов, называют *однозначно жесткой*. Если задача имеет конечное число решений с точностью до общего движения объектов, то она называется *жесткой.* Такая подзадача связывает относительное расположение всех своих объектов так, что невозможно осуществить

непрерывное движение одних объектов относительно других без нарушения ограничений.

**Предложение 3.2** Если метод декомпозиции допускает только однозначно жесткие задачи, то он является безусловно корректным.

**Доказательство** Пусть метод допускает декомпозицию $Z \prec_{G^*} (Z', Z^*)$, причем $Z'$ является однозначно жесткой. Возьмем некоторое $T' \in T(Z')$, по условию однозначной жесткости оно единственно. Если $T(Z) = \varnothing$, то декомпозиция корректна по определению, если же нет, то $\exists T \in T(Z)$. Однако, по предложению 3.1, так как $Z \mid Z'$, то $T \in T(Z) \Rightarrow \exists T'' \subseteq T, T'' \in T(Z')$. Итак, $\{T', T''\} \in T(Z')$, но по условию решение $Z'$ единственно, откуда получаем, что $T'' = T'$ и $T' \subseteq T$. То есть, $\forall T' \in T(Z') \exists T \in T(Z), T' \subseteq T$.

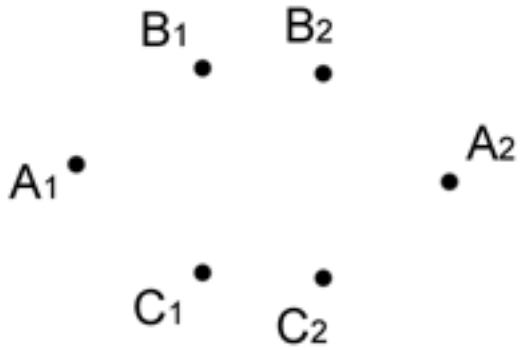


Рисунок 3.1. Два решения задачи, не совмещаемые общим движением

Примером однозначно жесткой задачи может быть трехмерная задача размещения одинокой точки относительно тела, включающего набор координатных плоскостей, по трем расстояниям до каждой из плоскостей (расстояния считаются по нормали и могут быть отрицательными). Так как эти расстояния задают координаты точки, которые единственны, решение задачи единственно. Примером жесткой, но не однозначно жесткой, задачи может быть двухмерная задача размещения трех одиноких точек по трем заданным расстояниям (см. рис. 3.1): двумя ее решающими означиваниями являются два треугольника с разным порядком вершин при положительном

обходе. Эти два решения не переводятся друг в друга собственным движением, а лишь с помощью зеркального отражения.

## 3.2 Критика известных методов декомпозиции

Как видно из приведенных примеров, и в однозначно жестких, и в жестких задачах ограничения снимают все взаимные степени свободы между телами, “сильно связывают” задачу. Методы декомпозиции, нацеленные на нахождение однозначно жестких и жестких задач, называются “сильносвязанными”. Существуют также “слабосвязанные” методы, использующие, наоборот, малую связанность частей задачи между собой. К ним относится двухсвязная декомпозиция, которая будет описана позднее.

При выделении нескольких вложенных не однозначно жестких подзадач возникает экспоненциальный перебор всех их решений. В полиномиальном решателе это не возможно, поэтому выбирается одно из решений каждой подзадачи. Поэтому методы сильносвязанной декомпозиции, не способные отличить жесткую подзадачу от однозначно жесткой, в полиномиальном решателе носят эвристический характер.

Наиболее известным методом сильносвязанной декомпозиции является метод формального анализа степеней свобод, или метод кластеринга, предложенный Хоффманном [59]. Он позволяет с помощью анализа помеченного гиперграфа задачи выделять несократимые формальные кластерные типы, представляющие хорошо определенные подзадачи. При этом для определения таких подзадач используется следующая формула:

$$\sum_{g\in G} d(g) - \sum_{e\in E} h(e) = \begin{cases} 3, D = 2 \\ 6, D = 3 \end{cases}, \text{ где } D \text{ – размерность пространства.}$$

Как видно, это соотношение отличается от соотношения, ранее указанного в главе 1 для хорошо определенной подзадачи:

$$\sum_{g\in G} d(g) + p + q - \sum_{e\in E} h(e) = d(\bigcup_{g\in G} g)$$

Вследствие этого метод Хоффманна может ошибаться при работе на следующих классах задач:

1) Задачи, в которых $q \neq 0$, то есть множество переменных не пусто;
2) Задачи, в которых $p \neq 0$, то есть объекты обладают внутренними степенями свободы (правда, Хоффманн предлагает видоизменение своего метода для некоторых классов таких задач [55, 56]);
3) Задачи, в которых $d(\bigcup_{g \in G} g) \neq C_2^{D+1}$, то есть тело, получающееся объединением всех тел подзадачи, не является полным телом. Таким свойством обладает достаточно большое количество задач, особенно в трехмерном пространстве.

Хоффманн излагает соображения, по которым хорошо определенные подзадачи являются жесткими или однозначно жесткими. Им установлено, что разумной стратегией (в смысле оптимизации общего времени решения) предложенной им декомпозиции является выделение минимального по размеру несократимого формального кластерного типа [61].

В отношении недоопределенных задач метод Хоффманна не столь эффективен, однако и в них он частично способен выделять сильно связанные части. Тем не менее, теоретический успех метода Хоффманна, к сожалению, не всегда ведет к практическому успеху.

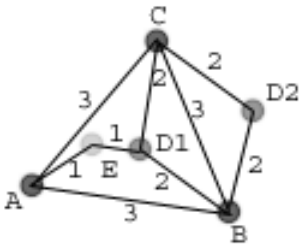


Рисунок 3.2. Не все решения подзадач ведут к решению всей задачи

Существуют примеры, в которых формальный анализ степеней свободы не может корректно декомпозировать задачу. Одним из вариантов такого некорректного поведения может быть выделение формально хорошо

определенной подзадачи, не являющейся однозначно жесткой, некоторые решения которой не могут быть дополнены до решения исходной задачи. Таким образом, получив решение подзадачи, иногда невозможно получить решение всей задачи. В примере на рис. 3.2 приведена задача размещения пяти точек $A,B,C,D,E$ в двухмерном пространстве по ограничениям расстояния между ними. Легко видеть, что из двух возможных означиваний $D_1,D_2$ точки $D$ в выделенной подзадаче, включающей точки $A,B,C,D$, лишь означивание $D_1$ может быть дополнено до решающего означивания всей задачи, включающей точки $A,B,C,D,E$. Действительно, означивание $D_2$ противоречит правилу треугольника, так как расстояние от $A$ до $D_2$ больше суммы расстояний от $A$ до $E$ и от $E$ до $D_2$.

Для того, чтобы избежать случайного выбора решения подзадачи, ведущего к невозможности решения всей задачи, необходимо провести перебор всех решений подзадач. Такой перебор является экспоненциальным по сложности от длины последовательности декомпозиции. Естественно, он неприемлем на практике, что ограничивает применимость методов, не являющихся безусловно корректными, в том числе и метода формального анализа степеней свободы.

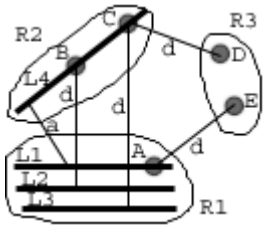


Рисунок 3.3. Зависимость ограничений ведет к некорректности

Еще большие проблемы возникают при декомпозиции двухмерной задачи, показанной на рис. 3.3. На нем пометкой ‘d’ обозначены ограничения расстояния, а пометкой ‘a’ – ограничения угла. Внутри тела $R_1$ прямые $L_1,L_2,L_3$ являются параллельными, внутри тела $R_2$ точки $B,C$ инцидентны прямой $L_4$. Такое специальное взаимное расположение объектов внутри тел приводит к тому, что состоящая из тел $R_1$ и $R_2$ формально хорошо

определенная задача имеет решения, неэквивалентные с точностью до общего движения. Действительно, $R_2$ можно двигать по горизонтали, не нарушая ограничений расстояния и угла с неподвижным $R_1$, и получать конфигурации, несовместные с ограничениями расстояния с объектами тела $R_3$. Поиску решения такой декомпозированной задачи не поможет даже метод перебора решений подзадач, поскольку таких решений бесконечно много. Причиной этого эффекта является зависимость ограничений между $R_1$ и $R_2$: зная угол и расстояние между прямой $L_2$ и точкой $B$, можно посчитать расстояние между прямой $L_3$ и точкой $C$.

Рисунок 3.4. Вращение частей формально хорошо определенной задачи

Таким образом, значительным недостатком метода Хоффманна является то, что он основан на формальном анализе степеней свободы и не учитывает семантические зависимости между ограничениями. В трехмерном пространстве этот недостаток метода проявляется еще чаще – на рис. 3.4 изображен известный пример “два банана” [73], состоящий только из точек и ограничений расстояния между ними. Эта формально хорошо определенная задача разделяется на две части, имеющих две общие точки (верхняя и нижняя точки на рисунке), которые могут свободно вращаться вокруг оси, проходящей через эти точки, таким образом, у этой подзадачи тоже бесконечное множество решений.

Существование подобных конфигураций побуждает разработчиков геометрических решателей не использовать общий формальный анализ степеней свободы в графе. Вместо него используют декомпозиции на основе выделения заранее заданных образцов в графе. При этом, как правило, применяют те образцы, которые позволяют получить однозначно жесткую

задачу. Как было показано в главе 1, множество несократимых формальных кластерных типов бесконечно даже для простейших классов задач параметрического проектирования, поэтому невозможно представить все хорошо определенные задачи конечным числом заранее заданных образцов. Это сильно сужает полезность декомпозиции на основе выделения заранее заданных образцов в графе, так как такой метод допускает меньшее число декомпозиций.

Результат декомпозиции Хоффманна, кроме того, сильно зависит от порядка применения элементарных декомпозиций. Это характерно как для общего формального анализа степеней свободы в графе, так и для выделения в нем заранее заданных образцов.

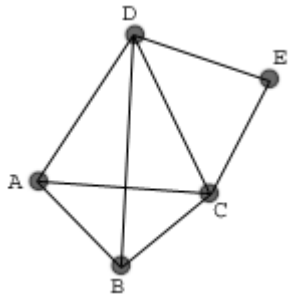


Рисунок 3.5. Задача с существенно различными декомпозициями

В трехмерной задаче на рис. 3.5, состоящей только из точек и ограничений расстояния между ними, можно выделить разные подзадачи, в частности, подзадачу $C,D,E$ и подзадачу $A,B,C$. Если продолжать эти декомпозиции, то получатся задачи-следствия из двух тел в каждом из случаев – $\{\{A,B\},\{C,D,E\}\}$ и $\{\{A,B,C,D\},\{E\}\}$ соответственно. При этом в первом случае в задаче будет четыре, а во втором – два ограничения, что существенно влияет на сложность решения этих задач.

Итак, метод декомпозиции Хоффманна не всегда является корректным. В связи с изложенными выше недостатками метод Хоффманна в индустриальных решателях используется в следующем режиме: при неуспехе в решении одной из найденных им подзадач происходит останов и запуск решения недекомпозированной задачи. Методы такого рода называются

эвристическими. Как правило, все условно корректные методы используются в геометрических решателях как эвристические.

Простейшим видом слабосвязанных декомпозиций является декомпозиция на двусвязные компоненты, или декомпозиция по вершинам сочленения [66]. Ее идея состоит в том, что, выявив все тела, являющиеся вершинами сочленения в графе задачи, можно определить его компоненты двусвязности, на которые вершины сочленения разбивают граф, и разделить исходную задачу на подзадачи, которые можно решать полностью независимо. Формально говоря, этот метод декомпозиции допускает такие множества подмножеств тел, которые являются компонентами двусвязности.

Если исключать из допускаемых методом декомпозиции по вершинам сочленения подмножеств тел такие, которые содержат вершину сочленения, включающую объект с дополнительными параметрами, то двусвязные декомпозиция на двусвязные компоненты является корректной. Для того чтобы это проверить, достаточно совместить решения подзадач: если $t_1$ и $t_2$ – движения являющегося вершиной сочленения задачи тела $g$ в решениях обеих подзадач соответственно, то легко построить общее решение задачи, умножив движения всех тел во второй подзадаче на $t_1(t_2)^{-1}$. Это можно сделать при условии, что все ограничения являются геометрическими. В то же время, декомпозиция на двусвязные компоненты весьма ограничена в применении, так как не способна разбивать циклы в графе объектов и ограничений, в то время как нетривиальными задачами параметрического проектирования являются именно задачи с циклами ограничений [33].

Таким образом, практически важным стал вопрос о разработке такого метода геометрической декомпозиции, который был бы безусловно корректным и способным разбивать циклы в задаче параметрического проектирования. Описание такого метода декомпозиции, предложенного автором [46] и получившего название метода отделяющей декомпозиции, будет приведено ниже.

## 3.3 Описание метода отделяющей декомпозиции

Пусть дана задача параметрического проектирования, граф которой таков, что декомпозиция по вершинам сочленения не может быть применена. Тем не менее, во многих случаях можно декомпозировать задачу полностью корректным образом, разорвав один или несколько циклов ограничений. Например, можно исключить из задачи прямую (при условии, что в содержащем ее теле нет других объектов), если на нее наложено только два ограничения инцидентности точкам. Для выделения такой подзадачи достаточно локального анализа графа задачи в окрестности прямой, поскольку не важно, какие еще ограничения существуют в задаче. Действительно, каковы бы ни были взаимные положения точек, всегда существует проходящая через них прямая. Следовательно, можно изъять прямую и два наложенных на нее ограничения из задачи, решить редуцированную задачу, а затем расположить исключенную прямую по двум инцидентным ей точкам. При этом если точки соединены некоторым другим путем из остальных ограничений задачи, то один из циклов ограничений в задаче будет разбит и сложность ее решения уменьшится.

Случай прямой, инцидентной двум точкам, не является единственным: существует множество вариантов сочетаний объекта и наложенных на него ограничений, которые позволяют изъять этот объект из задачи и расположить его после нахождения решения редуцированной задачи, каково бы ни было это решение. Способ проверки того, можно удалить объект $v$ или нет, заключается в составлении специальной задачи $Z_v$, состоящей из наложенных на объект ограничений $E_v \subseteq E$ и двух твердых тел, одно их которых $g$ совпадает с телом, содержащим $v$, а другое $g^*$ содержит все другие аргументы ограничений из $E_v$. Если эта задача разрешима при любых значениях координат объектов, то такое удаление возможно. Проверить это свойство можно путем анализа семантики наложенных ограничений. Заметим, что в общем случае эта проверка является весьма нетривиальной.

Если считать $g^*$ зафиксированным, то число степеней свободы тел задачи равно числу степеней свободы $d(g) = d(v)$ объекта $v$. Так как каждое ограничение уменьшает размерность многообразия решения задачи на число своих степеней свободы (при условии, что ограничения независимы), то для разрешимости задачи $Z_v$ надо, как минимум, чтобы суммарное число степеней свободы ограничений не превышало $d(v)$.

Другими примерами корректного отделения объекта являются:

- прямая в двухмерном пространстве, связанная двумя ограничениями, первое из которых – угол или параллельность другой прямой, а второе – расстояние до точки или окружности, инцидентность точке или касание окружности;
- окружность в двухмерном пространстве, связанная двумя ограничениями, первое из которых – инцидентность с точкой либо касание окружности, а второе – касание другой окружности или прямой.

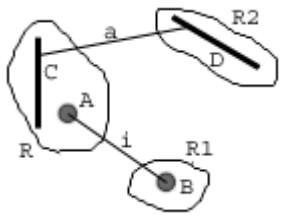


Рисунок 3.6. Корректное отделение тела с несколькими объектами

Этот список далеко не исчерпывающий. Как в двухмерном, так и в трехмерном случае существует множество вариантов сочетаний объекта и наложенных на него ограничений, при котором объект можно отделить, а после удовлетворения оставшихся ограничений, расположить в пространстве при любых заданных положениях других объектов. Кроме того, можно отделять не только одиночный геометрический объект, но и тело, если оно состоит, например, из точки и прямой, причем точка связана только ограничением инцидентности с другой точкой, а прямая связана только ограничением угла с другой прямой (см. рис. 3.6).

Тело удаляется из задачи вместе с наложенными на него ограничениями, следовательно, у других аргументов этих ограничений уменьшаются количества заданных для них ограничений. В результате, возможно, некоторые из этих тел тоже можно будет удалить из задачи, как показано на рис. 3.7.

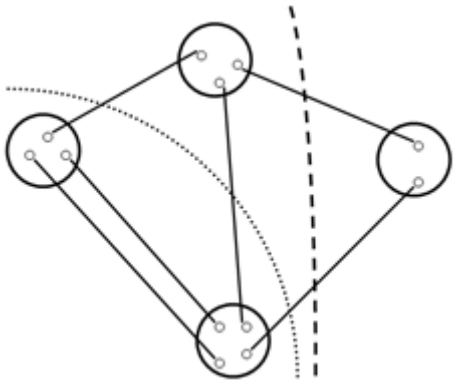

Рисунок 3.7. Последовательное отделение нескольких тел

Для выделения конфигураций, включающих тело и наложенные на него ограничения, которые можно корректно удалить из задачи, предлагается осуществлять поиск в графе специальных заранее заданных образцов. Определим далее понятие такого образца.

Типом геометрических объектов будем называть класс всех объектов, имеющих одинаковую геометрическую природу, например, все двухмерные точки или все кривые в трехмерном пространстве. Будем говорить, что конкретный объект $v$ является представителем типа $x$, и записывать этот факт записью $x = X(v)$. Множество всех типов объектов будем обозначать $\hat{X}$. Типом геометрических ограничений тоже будем называть класс всех ограничений, имеющих одинаковую геометрическую природу, например, все расстояния между трехмерными точками или все концентричности двухмерных окружностей. Будем говорить, что конкретное ограничение $e$ является представителем типа $y$, и записывать этот факт записью $y = Y(e)$. Множество всех типов ограничений будем обозначать $\hat{Y}$. Арности ограничений одного типа одинаковы, поэтому введем функцию арности $a : \hat{Y} \to N$, сопоставляющую типу количество его аргументов. Введем также

функцию соответствия типов $A:\hat{Y}\times N\to\hat{X}$, которая любому типу $y$ и номеру $i\le a(y)$ сопоставляет тип $x$ объектов, являющихся $i$-ми аргументами ограничений типа $y$.

**Определение 3.9** $V$-образцом для объекта типа $x$ назовем двойку $P=(Y,M),Y=(y_1,..,y_r),\forall i,y_i\in\hat{Y}$, $M=(n_1,..,n_r),\forall i,n_i\in N$, в которой

1) $\forall i\in\{1,..,r\},n_i\le a(y_i)$.

2) $\forall i\in\{1,..,r\},A(y_i,n_i)=x$.

Будем называть $r$ – мощностью образца, а $x$ – собственным типом объекта образца, все другие $A(y_i,j)$, $i\in\{1,..,r\},j\in\{1,..,a(i)\},j\ne n_i$ называются несобственными. Определение $V$-образцов отделяющей декомпозиции для тел, состоящих из более чем одного объекта, производится по аналогии.

**Определение 3.10** Пусть даны $V$-образец $P=((y_1,..,y_r),(n_1,..,n_r))$, задача $Z=(\hat{V},(E,a,A),G,S_0,F,\varepsilon)$, тело $g\in G$. Обозначим $E_g=\{e\in E\mid\exists i\le a(e):A(e,i)\in g\}$. Образец $P$ применим для тела $g$ в задаче $Z$, если существует $W:E_g\to\{1,..,r\}$ такая, что:

1) $\forall e_1,e_2\in E_g,e_1\ne e_2\Rightarrow W(e_1)\ne W(e_2)$

2) $\forall e\in E_g,Y(e)=y_{W(e)}$

3) $\forall e\in E_g\forall i\le a(e),A(e,i)\in g\Rightarrow i=n_{W(e)}$

**Определение 3.11** Пусть дан $V$-образец $P=((y_1,..,y_r),(n_1,..,n_r))$ для объекта типа $x$ и $\forall s(i,j)\in R^{N(i,j)}\left(i\in\{1,..,r\},j\in\{1,..,a(y_i)\}\setminus n_i,N(i,j)=C(A(y_i,j))\right)$ $\exists s\in R^{C(x)}$ такой, что для любого $\forall i\in\{1,..,r\}$ ограничение типа $y_i$ на означивании $(s(i,1),..,s(i,n_i-1),s,s(i,n_i+1),..,s(i,a(y_i))$ выполнено. Такой образец называется образцом отделяющей декомпозиции для объекта типа $x$.

Легко проверить, что указанные ранее конкретные случаи корректного отделения объекта от задачи, такие, как отделение прямой, связанной двумя ограничениями инцидентности точкам, есть образцы отделяющей

декомпозиции для объекта в смысле определения 3.11. Список образцов для метода отделяющей декомпозиции приведен в приложении 2.

Абстрагировавшись от конкретных образцов, позволяющих корректно исключать объекты из задачи, опишем формально процесс отделяющей декомпозиции.

**Определение 3.12** Пусть дана последовательность из $c$ неповторяющихся тел $(g_1,..,g_c) \subset G$ задачи параметрического проектирования (без требования оптимальности) $Z = (\tilde{V}, \tilde{E}, G, \varepsilon)$. Будем говорить, что этой последовательностью тел индуцирована последовательность подзадач $Z_0,..,Z_c$, $\forall i \in \{0,..,c\}\ Z_i = (\tilde{V}_i, \tilde{E}_i, G_i, \varepsilon)$, если:

1) $G_o = G \setminus \{g_1,..,g_c\}$, $Z|_{G_0} Z_0$;

2) $\forall i \in \{1,..,c\}, \tilde{E}_i = (E_i, a, A),\ E_i = \{e \in E \mid g_i \in \bigcup_{i \in \{1,..,a(e)\}} \overline{G}(A(e,i)) \subset G_o \cup \{g_1,..,g_i\}\}$;

3) $\forall i \in \{1,..,c\} : V_i = \{v \in V \mid \exists e \in E_i, \exists j \le a(e) : A(e,j) = v\}$

4) $\forall i \in \{1,..,c\} : G_i = \{g_i, g_i^*\}$, где $g_i^* = (V_i \cap G) \setminus g_i$

Заметим, что тогда выполняются следующие свойства:

1) $E = E_0 \cup .. \cup E_c$, $i \ne j \Leftrightarrow E_i \cap E_j = \varnothing$;

2) $V = V_0 \cup .. \cup V_c$;

3) $G_0 \cup .. \cup G_c = G \cup \{g_1^*,..,g_c^*\}$, $\forall i \in \{1,..,c\}, g_i^* \notin G$.

**Предложение 3.4** Если последовательность подзадач $Z_0,..,Z_c$ задачи $Z$ индуцирована последовательностью тел $(g_1,..,g_c) \subset G$, то $Z \prec (Z_0,..,Z_c)$.

**Доказательство** Докажем по индукции. Условие 1) в определении 3.12 указывает, что если $Z_0, Z_1$ индуцирована последовательностью $(g_1) \subset G$, то $Z \prec (Z_0, Z_1)$. Для обоснования индукционного шага для последовательности $Z_0,..,Z_c$ заметим, что если взять $G' = G \setminus \{g_c\}$, то элементарной декомпозицией по этому множеству будет пара $(Z_c', Z_c)$, где $Z|_{G'} Z_c'$. Таким образом, осталось

доказать, что $Z_c' \prec Z_0,..,Z_{c-1}$, но, как легко видеть, $Z_0,..,Z_{c-1}$ – индуцированная $(g_1,..,g_{c-1}) \subset G$ последовательность подзадач $Z_c'$, имеющая меньшую длину, и, по предположению индукции, это уже выполнено.

**Определение 3.13** Назовем последовательность $Z_0,..,Z_n$ подзадач $Z$, индуцированную последовательностью тел, корректной, если

$$\forall S_0 \in S(Z_0) \exists S \in S(Z) : S_0 \subset S$$

**Определение 3.14** Функция $D$, которая задаче $Z = (\tilde{V}, \tilde{E}, G, \varepsilon)$ сопоставляет множество $\hat{G} \subset 2^G$ подмножеств тел, называется методом декомпозиции отделением тел, если $\forall G' \in \hat{G}$, $G \setminus G' = \{g'\}$ содержит только одно тело, причем найдется корректная последовательности подзадач, индуцированная последовательностью тел, в которой $g'$ является последним.

Заметим, что тогда задача $Z^*, Z \xrightarrow[G']{} Z^*$ является разрешимой при любых означиваниях своих объектов. Следовательно, такой метод декомпозиции является безусловно корректным.

**Определение 3.15** Пусть задан набор $\{P_1,..,P_n\}$ образцов отделяющей декомпозиции. Функция $D$, которая задаче $Z = (\tilde{V}, \tilde{E}, G, \varepsilon)$ сопоставляет множество $\hat{G} \subset 2^G$ подмножеств ее тел, причем $\forall G' \in \hat{G}$, множество $G \setminus G' = \{g'\}$ содержит единственное тело, для которого некоторый $P_i$ является образцом отделяющей декомпозиции, называется методом отделяющей декомпозиции.

Из определений 3.11, 3.14 и 3.15 легко видеть, что метод отделяющей декомпозиции $D$ для любого набора образцов является методом декомпозиции отделением тел. Тогда генерируемая им корректная последовательность подзадач называется последовательностью отделяющей декомпозиции

**Определение 3.16** Пусть $D$ – метод отделяющей декомпозиции. Будем говорить, что генерируемая им последовательность $Z_0,..,Z_c$ отделяющей декомпозиции является полной, если $D(Z_0) = \varnothing$.

## 3.4 Алгоритм отделяющей декомпозиции и его временная сложность

Опишем алгоритм, позволяющий практически осуществлять отделяющую декомпозицию задачи $Z = ((V = V^d \cup V^*), (E, a, A), G, \varepsilon)$. При этом будем считать, что значения функций $a, A, \overline{G}$ вычислимы за константное время. Это является практически оправданным, так как обычно задачи параметрического проектирования задаются через перечисление аргументов каждого ограничения и указания для каждого объекта тела, которому он принадлежит. Вычисления будем проводить на гиперграфе задачи $(G, E)$, который легко построить по задаче $Z$ за линейное время, воспользовавшись функцией $\overline{G}$.

Легко видеть, что для построения последовательности отделяющей декомпозиции $Z_0, .., Z_c$ достаточно найти последовательность неповторяющихся тел $G^* = (g_1, .., g_c) \subset G$, которая индуцирует эту декомпозицию, после чего можно воспользоваться формулами из определения 3.12. Ниже приведен псевдокод, вычисляющий $G^*$, используемая в нем константа C равна 3 для двухмерного и 6 для трехмерного случая:

```
1 Function CuttingDecomposition(Z)
2 For each e from E Rem(e) := FALSE
3 Form(Degree, Mapping)
4 G′ := {g from G | Degree(g)<=C}, G* := {}
5 While (G′ is non empty) do loop A
6   g := G′.pop()
7   If (CheckCutting(g,Mapping(g))) then do
8       For each e from Mapping(g) do loop B
9           If (Rem(e)=TRUE) continue
10          Rem(e) := TRUE
11          For each g′ from A(e) do loop C
12              Degree(g′) := Degree(g′)-1
13              If (Degree(g′)<=C) then do
14                  Reform(Mapping(g′),Rem)
15                  G'.push(g′)
16              Endif
17          End loop C
```

```
18        End loop B
19        G*.push(g)
20        G.pop(g)
21    Endif
22End loop A
```

В приведенном коде используются следующие структуры данных

- `Rem(E)` – массив булевых пометок, указывающий для каждого гиперребра, было ли оно использовано для отделения вершин;
- `Degree(G)` – массив степеней вершин в `G`, в начале инициализируется обходом ограничений функцией `Form`;
- `Mapping(G)` – обращение функции $A$, то есть сопоставление вершине $g \in G$ массива из `Degree(g)` инцидентных ей гиперребер; инициализируется функцией `Form`, которая обходит все гиперребра графа;
- `G'` – стек вершин графа, которые рассматриваются как потенциально подходящие для действия отделяющей декомпозиции;
- `G*` – готовая последовательность неповторяющихся вершин-тел $(g_1,..,g_c) \subset G$, которая индуцирует отделяющую декомпозицию.

Кроме того, используются вызовы следующих функций:

- `Form` – формирование `Degree(G)`, `Mapping(G)` однократным обходом всех гиперребер;
- `CheckCutting(g,Mapping(g))` – функция, проверяющая, можно ли отделить вершину-тело $g \in G$, связанную с другими вершинами лишь ограничениями `Mapping(g)`, которых не больше заранее заданной константы `C`. В своей работе она использует сравнение с несколькими заранее заданными образцами отделяющей декомпозиции, сравнение с каждым из образцов требует константного времени;

- `Reform` – исключение из `Mapping(G)` всех гиперребер `e`, для которых `Rem(e)=TRUE`;

Пусть количество тел в задаче параметрического проектирования $Z$ равно $n$, количество ограничений равно $m$, а $w = \sum_{e \in E} a(e)$, без ограничений общности $w \geq m$, $w \geq n$. Заметим, что, так как в записи задачи необходимо задать соответствие $A$, которое каждому ограничению сопоставляет набор его аргументов, то длина описания задачи не меньше $w$.

Покажем, что если функция `CheckCutting` выполнима за константное время, то предложенный алгоритм будет линейным от $w$, а значит, линейным от длины входа алгоритма `CuttingDecomposition`. Действительно:

1) Шаги 2-4 выполняются при однократном обходе всех ограничений, суммарное время, потраченное на их выполнение, не превышает $O(w)$;
2) Операторы непосредственно внутри цикла A выполняются для каждой вершины `g` не более чем `C` раз, (то есть всего $O(n)$ раз), так как исполнение цикла влечет за собой удаление `g` из `G'`, а вновь попасть в `G'` `g` может лишь при выполнении шага 15. Но, как следует из шагов 8-10, выполнение шага 15 возможно лишь при использовании одного из `Degree(g)` гиперребер, инцидентных `g` и имеющих пометку `Rem(e)=FALSE`. Таких случаев будет не больше, чем `C`, ведь каждому новому выполнению шага 15 предшествует пометка `Rem(e)=TRUE`, которая предотвращает повторное использование гиперребра;
3) Операторы непосредственно внутри цикла B исполняются не более $O(w)$ раз, так как каждый раз выполнение цикла происходит для некоторого гиперребра `e`, инцидентного ранее не использовавшейся вершине `g`;

4) Операторы непосредственно внутри цикла C исполняются не более $O(w)$ раз, так как каждый раз выполнение цикла происходит для некоторой вершины `g'`, инцидентной ранее не использовавшемуся гиперребру `e`;

5) Как было замечено в 2), шаги 14-15 выполняются для любой вершины `g` не более чем `C` раз. При этом функция `Reform` лишь при первом вызове для каждого g требует времени, линейно зависящего от степени `g` до начала работы алгоритма, после чего она требует константного времени. Следовательно, суммарное время на выполнение этих шагов тоже не превышает $O(w)$.

Легко видеть, что, опираясь на найденную последовательность тел $(g_1,..,g_c) \subset G$, пользуясь определением 3.12 и описанными выше структурами данных можно сконструировать все компоненты задач $Z_0,..,Z_n$ за время $O(w)$. Таким образом, все алгоритмические процедуры, необходимые для построения последовательности отделяющей декомпозиции, требуют времени, линейно зависящего от описания исходной задачи.

Как видно из сказанного выше, смысл использования константы `C` состоит в ограничении временной сложности алгоритма. Однако, она имеет и простой геометрический смысл: каким бы ни было тело, оно не может иметь больше степеней свободы, чем `C`, а каждое геометрическое ограничение снимает не менее одной степени свободы, поэтому для обеспечения решения задачи после отделения объекта нужно использовать образцы с не более чем `C` ограничениями.

**Предложение 3.5** Для любой задачи параметрического проектирования $Z$, ее подзадачи $Z'$ и метода отделяющей декомпозиции $D$, если образец отделяющей декомпозиции $P$ применим для тела $g \in Z'$ в $Z$, то он применим и для $g$ в $Z'$.

**Доказательство** Множество $E_g$ из определения 3.10 может лишь уменьшиться от применения других образцов отделяющей декомпозиции, откуда легко следует доказательство.

**Предложение 3.6** Для любой задачи параметрического проектирования $Z$ и метода отделяющей декомпозиции $D$, если $Z_0',..,Z_n'$ и $Z_0*,..,Z_n*$ – две полные последовательности отделяющей декомпозиции, сгенерированные $D$, то $Z_0' = Z_0*$.

**Доказательство** Пусть $Z_0',..,Z_n'$ индуцирована $\{g_1',..,g_{c'}'\}$. $g_{c'}'$ был отделен в $Z$, поэтому, $g_{c'}' \notin Z_0*$, так как иначе получится противоречие с предложением 3.5. Значит, подзадача $Z_1$ задачи $Z$ по телу $g_{c'}'$ содержит в качестве подзадачи $Z_0*$. Далее, $g_{c'-1}'$ был отделен в $Z_1$, поэтому $g_{c'}' \notin Z_0*$, которая является подзадачей $Z_1$. Рассуждая аналогично, будем последовательно получать, что $\forall i, Z_i$ – подзадача $Z$ по телу $g_{c'-i+1}'$ содержит в качестве подзадачи $Z_0*$ и $Z_i = Z_0' | Z_0*$. В силу симметрии, $Z_0* | Z_0'$, $Z_0' = Z_0*$.

Так как любой образец отделяющей декомпозиции является достаточно простым и, в силу корректности, соответствующая ему подзадача всегда разрешима, то при решении последовательности задач отделяющей декомпозиции $Z_0',..,Z_n'$ трудности может представлять только решение $Z_0'$. Однако, в силу предложения 3.6, эта подзадача не зависит от последовательности применения образцов отделяющей декомпозиции, а только от их множества. В силу этого, метод отделяющей декомпозиции можно считать детерминированным.

## 3.5 Расширения метода отделяющей декомпозиции

Существуют V-образцы, корректность которых зависит от взаимного расположения объектов в отделяемом теле. На рис. 3.8 тело из плоскости и прямой, каждая из которых связана ограничением расстояния до точки, не может быть отделено в случае, когда объекты параллельны (слева), а может –

в случае, когда объекты не параллельны (справа). Действительно, в первом случае, если точки случайно совпадут и сумма расстояния до них будет меньше расстояния между прямой и плоскостью, тело нельзя будет разместить по расстояниям до точек.

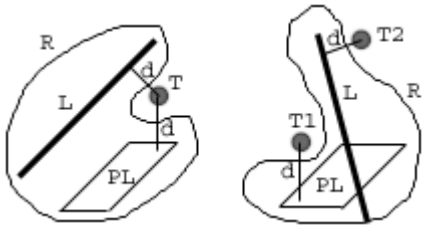


Рисунок 3.8. Некорректное и корректное отделение тела

Такие V-образцы называются условно корректными. Так как взаимное положение объектов внутри тела известно априори, то легко изменить понятие V-образца так, чтобы использование таких образцов было корректно. Необходимо лишь добавить проверку ограничений на расположение объектов внутри проверяемого тела к условиям применимости в определении 3.10. Некоторые условно корректные образцы могут быть весьма большими, существуют условно корректные образцы в трехмерном пространстве даже с шестью ограничениями, что является предельным случаем из геометрических соображений. Например, тело с тремя плоскостями, связанными 3, 2 и 1 ограничением инцидентности точкам соответственно, может быть отделено, если все плоскости перпендикулярны (т.е. образуют систему координат), как показано на рис. 3.9.

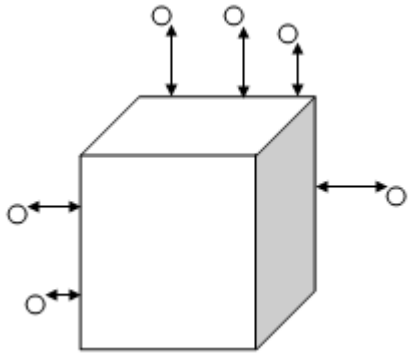

Рисунок 3.9. Корректный V-образец с шестью ограничениями

Кроме описанных выше условно корректных образцов, есть условно корректные образцы несколько иного рода. Например, двухмерную точку нельзя отделить по двум ограничениям расстояния до прямых, однако, если известно, что прямые пересекаются, то такое отделение корректно. Иногда тот факт, что прямые пересекаются, легко вывести априори: между прямыми может быть задано ограничение угла, они могут быть инцидентны общей точке, лежать в одном теле и т. п. Проводя такой дополнительный анализ, можно выполнить еще большее количество корректных декомпозиций.

Некоторые V-образцы не всегда корректны, причем их корректность нельзя проверить априори. Образцы, корректность которых нельзя проверить априори, называются эвристическими. Наиболее интересны из них те V-образцы, которые корректны при почти всех возможных расположениях объектов – то есть в случаях, когда множество означиваний $S'$ несобственных объектов, при которых не существует означивания собственного, удовлетворяющего все ограничения, имеет меру нуль. Примером может быть V-образец, включающий три ограничения касания между собственным объектом – окружностью в двухмерном пространстве – и несобственными прямыми. Если прямые попарно не параллельны, то всегда можно найти окружность, которая касается их всех. Множество расположений трех прямых на плоскости, при котором какие-то две из них параллельны, имеет меру нуль.

Такие V-образцы называют нуль-эвристическими, что соответствует тому, что их использование может привести к невозможности решения исходной задачи во множестве случаев меры нуль. Также к нуль-эвристическим образцам следует отнести: двухмерную точку с двумя заданными расстояниями до двух прямых, трехмерную точку с тремя заданными расстояниями до плоскостей или прямых, и многие другие V-образцы.

Нуль-эвристические V-образцы достаточно эффективны на практике, в отличие от образцов, где множество “плохих” расположений объектов имеет

меру, большую нуля. Например, V-образец, включающий двухмерную точку и два расстояния $d_1, d_2$ до двух несобственных точек, приведет к невозможности решить исходную задачу во всех случаях, когда несобственные объекты будут расположены так, что нарушится правило треугольника: расстояние между ними будет меньше $|d_1 - d_2|$ или больше $(d_1 + d_2)$. Таким образом, использование этих V-образцов для отделяющей декомпозиции приведет к тем же проблемам, что и в "плохих" случаях для метода Хоффманна.

## 3.6 Свойства метода отделяющей декомпозиции

Если сравнивать предложенную декомпозицию с декомпозицией, основанной на формальном анализе степеней свободы, то надо заметить, что отделяющая декомпозиция применима для несколько другого класса задач. Она неэффективна на переопределенных задачах, ограниченно эффективна на хорошо определенных задачах, так как требует, чтобы отделяемый ею объект был слабо связан с остальной задачей. В то же время, опыт ее использования на недоопределенных задачах, как будет показано позднее, доказывает ее высокую эффективность. Надо заметить, что в реальной практике проектирования инженер большую часть времени имеет дело именно с недоопределенными задачами, так как конструирование деталей и механизмов, как правило, сводится к последовательному наложению ограничений, которые все более и более специфицируют модель.

Отделяющая декомпозиция, как было показано, является детерминированным методом – в результате последовательного отделения объектов, независимо от порядка выбора объектов для отделения, будет отделено одно и то же множество объектов и будет получена одна и та же редуцированная задача.

Отделяющую декомпозицию можно усилить добавлением эвристических образцов, то есть таких образцов, у которых решения редуцированной задачи не всегда может быть дополнено до решения исходной задачи. При этом надо понимать, что выигрыш в среднем времени

счета будет достигнут только в ситуации, когда уменьшение времени решения задач, в которых отделяющая декомпозиция удачно декомпозирует задачу, превышает время, потраченное на неуспешное решение задач, в которых отделяющая декомпозиция ошибочно декомпозирует задачу. Итак, при программной реализации метода отделяющей декомпозиции разумно:

1) использовать условно корректные V-образцы с дополнительными условиями, проверяемыми априори;
2) использовать нуль-эвристические V-образцы (хотя такое использование требует дополнительного тестирования на репрезентативной для конкретной области базе тестов);
3) не использовать эвристические V-образцы, не являющиеся нуль-эвристическими.

Отделяющая декомпозиция является полностью корректным методом, поскольку выбор образцов для нее происходит именно по принципу гарантии дополнения решения редуцированной задачи до решения всей исходной задачи. В отличие от метода Хоффманна, отделяющая декомпозиция учитывает семантику обрабатываемых ею ограничений. Ее целесообразно комбинировать с другими корректными методами, например, с декомпозицией по вершинам сочленения. Более того, совместное использование этих двух декомпозиций существенно расширяет круг декомпозируемых задач: из задачи, не декомпозируемой одной из декомпозиций, можно получать посредством другой декомпозиции задачу, которая уже может быть декомпозирована с помощью первой декомпозиции.

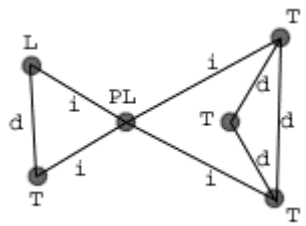


Рисунок 3.10. Отделяющая декомпозиция после двусвязной декомпозиции

На рис. 3.10 после применения декомпозиции по вершинам сочленения плоскость PL может быть отделена в обеих возникающих подзадачах, как инцидентная лишь двум объектам – точке и прямой в левой подзадаче и двум точкам в правой подзадаче, что было невозможно до выполнения декомпозиции по вершинам сочленения.

На рис. 3.11 отделение прямых L1, L2, L3 и L4, каждая из которых инцидентна двум точкам, позволяет разбить все циклы и получить древовидный граф, который полностью декомпозируется по вершинам сочленения.

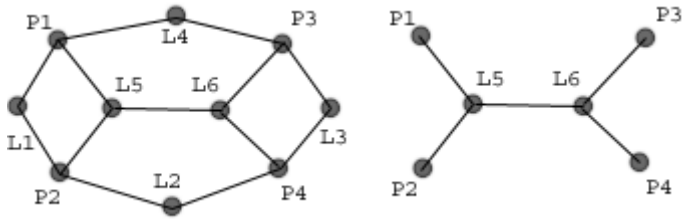


Рисунок 3.11. Двухсвязная декомпозиция после отделяющей декомпозиции

## 3.7 Выводы

В этой главе диссертации введены дополнительные определения, связанные с задачей параметрического проектирования, на их основе осуществлена критика имеющихся методов декомпозиции и предложен новый метод декомпозиции задач параметрического проектирования.

Этот метод – отделяющая декомпозиция – является корректным методом декомпозиции геометрических задач; он применим для задач с циклами, для которых обычные неэвристические методы, такие, как декомпозиция по вершинам сочленения, не могут быть применены. Более того, использование этого метода позволяет получать из задач, к которым неприменима декомпозиция по вершинам сочленения, задачи, которые могут быть ею успешно декомпозированы. Метод является достаточно общим, он может использоваться как в двухмерном, так и в трехмерном пространстве для недоопределенных и хорошо определенных задач. Его главное идейное отличие от известных методов формального анализа степеней свободы и

метода двухсвязной декомпозиции состоит в существенном использовании семантики ограничений.

Автором была предложена идея метода отделяющей декомпозиции, он же предложил его теоретическое и алгоритмическое описание, получил оценку вычислительной сложности и доказал свойство детерминированности метода. Автор также выполнил реализацию этого метода в двухмерном и трехмерном решателях задач параметрического проектирования. В силу эффективности метода дальнейшие работы в области отделяющей декомпозиции, включающие в себя разработку новых корректных V-образцов, велись под руководством автора несколькими исследователями, наибольший вклад в эти исследования внес И. Рыков.

# 4. Интервальные методы в задачах параметрического проектирования

## 4.1 Постановка задачи

Интервальные методы считаются некоторыми исследователями одним из самых перспективных направлений развития в теории решения задач параметрического проектирования [80]. Их преимущества заключаются в способности бороться с привнесением ошибок округления в процессе решения [54], а также в способности работать с неточно заданными входными данными. Как правило, интервальные методы по сравнению с классическими имеют меньшую производительность, поэтому их применение в промышленных решателях является вопросов компромисса между скоростью и качеством решения. Кроме того, есть и другие сложности в применении интервальных методов, например, введение интервальных данных может существенно усложнить класс решаемых задач [9].

**Определение 4.1** Интервал над полем действительных чисел $R$ есть множество

$$[\underline{x},\ \overline{x}] = \left\{x \,\middle|\, x \in R, \underline{x},\ \overline{x} \in R \cup \{-\infty, +\infty\},\ \underline{x} \le x \le \overline{x}\right\}.$$

$\underline{x}, \overline{x}$, называются соответственно нижней и верхней границами интервала. Если $\underline{x} > \overline{x}$, то $[\underline{x}, \overline{x}]$ не содержит ни одного элемента, то есть является пустым множеством. Если $\underline{x} = \overline{x}$, то интервал содержит только одно число, такой интервал называется вырожденным. Множество всех интервалов обозначается $IR$.

Далее в формулах будет использоваться расширенная интервальная арифметика [2]. Сложение и умножение в этой арифметике обладают ассоциативностью и коммутативностью, но не обладают дистрибутивностью. Для интервалов выполняется свойство, называемое субдистрибутивностью:

$$a*(b+c) \subseteq a*b + a*c$$

Для операций над интервалами выполняется основное свойство интервальных вычислений – монотонность по включению, то есть, если $a^1, a^2, b^1, b^2 \in IR$ и $a^1 \subseteq b^1, a^2 \subseteq b^2$, тогда для любой операции ‘$*$’ справедливо

$$a^1 * a^2 \subseteq b^1 * b^2.$$

Все интервальные методы требуют в процессе своей работы вычислений интервальных значений функций на тех или иных промежутках. Задача вычисления интервального значения композиции функций сводится за счет введения промежуточных переменных к вычислению интервальных значений простейших функций и операций [28]. К ним относятся все арифметические действия и элементарные математические функции: $\sin, \cos, tg, \exp, \ln$ и т. п.

Интервальное значение для монотонно возрастающей (убывающей) на интервале-аргументе функции равно промежутку, левый конец которого равен значению функции в левом (правом) конце интервала-аргумента, а правый конец – значению в правом (левом) конце. Пользуясь этим свойством и свойствами кусочно-монотонности по каждому аргументу элементарных функций и арифметических операций, для вычислений интервальных результатов любой арифметической операции достаточно выполнить ее над несколькими точечными значениями.

При вычислениях на ЭВМ арифметических операций, в силу конечной разрядкой сетки для представления действительных чисел необходимо производить округления в заранее заданных направлениях: вниз при вычислении нижней границы результата, вверх при вычислении верхней границы. Для вычисления же верхних и нижних границ интервала значений элементарной функции этого недостаточно – необходимо использовать специальную математическую библиотеку, которая корректно производит такие вычисления. Простое изменение направления округления перед вызовом встроенной библиотеки элементарных функций приводит к ошибочному результату [45].

В настоящее время существует несколько разработок интервальных математических библиотек, однако почти каждая из них, находящаяся в открытом доступе, обладает теми или иными недостатками. Так, например, для реализации библиотеки математических функций Pascal-XSC используется целочисленная арифметика, которая позволяет получать точные результаты при операциях над числами с плавающей точкой, однако использование целочисленной арифметики значительно увеличивает время вычислений [12]. В некоторых работах используется в качестве интервальной оценки результата вычисления математической функции интервал $[a-e, a+e]$, где $a$ – результат, выдаваемый стандартной библиотечной функцией, а $e$ – некоторая малая константа. В таком случае интервальный результат получается неоправданно широким, к тому же нет гарантий, что вычисления функций стандартной библиотекой вносят погрешность, всегда меньшую $e$.

Таким образом, для реализации в геометрическом решателе интервальных методов необходимо разработать высокопроизводительную библиотеку вычисления гарантированных интервальных оценок элементарных функций, что и будет первой задачей, решение которой описано в этой главе. Такую библиотеку целесообразно использовать и вне контекста интервальных методов, для вычисления неинтервальных математических функций с высокой точностью, что важно для избежания накопления ошибок округления [54].

Вторая задача, которой будет посвящена эта глава, состоит в разработке общего подхода к использованию интервальных методов решения систем нелинейных уравнений для задач параметрического проектирования. Будет рассмотрен общий способ, заключающийся в применении комбинации интервальных методов распространения ограничений и интервальных методов поиска решения. Далее будут, в соответствии с отмеченными достоинствами и недостатками, указаны сферы его практического применения.

## 4.2 Интервальная математическая библиотека

### 4.2.1 Выбор подхода к вычислению элементарных функций

**Определение 4.2** Множество вещественных чисел, представимых на ЭВМ (далее такие числа будут называться машиннопредставимыми), будем обозначать $R_{FP}$. Согласно стандарту IEEE-754 представления на ЭВМ вещественных чисел двойной точности, мантисса таких чисел представляется 52 битами, порядок 11 битами, еще 1 бит задает знак числа. В дальнейших рассуждениях предполагается, что $R_{FP}$ удовлетворяет IEEE-754.

**Определение 4.3** Числа $x, y \in R_{FP} : x \neq y, [x, y] = \{x, y\}$ называют подряд идущими или последовательными.

**Определение 4.4** Шириной интервала $[x, y], x, y \in R_{FP}$ называют число различных $z \in R_{FP} : z \in [x, y], z \neq y$. Ширина интервала, границы которого есть два последовательных числа из $R_{FP}$, равна 1.

При разработке алгоритмов вычисления математических функций стоят две главные задачи – точность и скорость. Время вычисления значений интервальных функций существенно влияет на эффективность интервальных вычислений в целом. Использование результатов для интервальных методов накладывает дополнительное требование *гарантированности вычислений*, то есть, получения корректных нижних и верхних оценок вычисляемой функции. Поэтому для интервальных вычислений свойство точности вычислений должно быть сформулировано в особом виде – необходимым является получение как можно более узкого интервала, содержащего истинный результат, в идеале, состоящего их двух подряд идущих чисел из $R_{FP}$. Таким образом, измеряется не абсолютная, а относительная точность.

Классическим методом вычисления элементарных математических функций является разложение их в ряд Тейлора. Однако, из-за высоких требований к производительности автор предлагает использовать приближения многочленами Чебышева всюду, кроме окрестности нуля

элементарных функций. Разложение Чебышева обеспечивает, при фиксированном числе членов ряда, значительно меньшую абсолютную погрешность на интервале разложения, а при фиксированной погрешности – меньшее число членов ряда. Существует так же аппроксимация так называемым оптимальным полиномом, наилучшая в смысле минимизации максимума отклонения для полиномов конкретной степени, однако для вычисления коэффициентов этих полиномов нужно найти корни многочлена высокой степени итерационным методом Ньютона, что вносит неустранимую погрешность во все дальнейшие вычисления. Так как погрешность разложения в ряд Чебышева лишь немного больше погрешности аппроксимации оптимальными полиномами [27], в качестве основного метода вычислений было выбрано разложение в ряды Чебышева.

Разложения Чебышева строятся в виде

$$f(x)=\sum_{n=0}^{\infty} a_n T_n(x),\ x\in[-1,1]$$

В этой формуле $T_n(x)$ – чебышевские многочлены, заданные на интервале $[-1,1]$. Для них существует рекуррентные формулы [27]:

$$T_o(x)=1,\ T_1(x)=x,\ T_{n+1}(x)=2xT_n(x)-T_{n-1}(x)$$

При этом $x$ может быть некоторой линейной функцией: если необходимо рассматривать разложение функции на интервале $[a,b]$, то используется линейное преобразование $t=(a+b)/2+x*(b-a)/2$.

Как известно, общий член разложения в ряд Чебышева $T_n(x)/(2^{n-1})$ будет иметь наименьшее отклонение от нуля среди всех многочленов данной степени с единичным коэффициентом при старшем мономе. В этом случае оценка погрешности разложения $L_n(x)$ функции $f(x)$ в ряд Чебышева примет вид:

$$|f(x)-L_n(x)|\le\frac{M_{n+1}}{2^n(n+1)!},\ \text{где}\ M_{n+1}=\sup|f^{n+1}(x)|,\qquad(4.1)$$

легко вычисляемый для любой элементарной функции.

Вычисление элементарной функции на всей области определения сводится к вычислению ее на некотором, называемым каноническим, интервале, который составляет:

$$\begin{cases} [0,\pi/2],\sin(x) \\ [0,\pi/4],tg(x) \\ [0,\sqrt{2}/2],\arcsin(x) \\ [0,1],arctg(x) \\ [0,\ln(2)],\exp(x) \\ [0.75,1.5],\ln(x) \end{cases}$$

Значения этих функций на других областях, а также значения остальных функций $\cos(x), ctg(x), \arccos(x), arcctg(x)$ вычисляются с помощью известных формул приведения.

Для уменьшения вычислительных затрат канонический интервал делится на подинтервалы длиной $2^{-k}$, на каждом из которых действует свое разложение, требующее вычисления значительно меньшего числа членов ряда за счет масштабирования коэффициентов полиномов Чебышева и ускорения сходимости ряда. Выбор ширины подинтервала кратной 2 позволяет исключить появление ошибок при приведении аргумента, так как мантисса при таком делении не изменяется.

Вид частичной суммы ряда Чебышева можно преобразовать

$$\sum_{i=0}^{n} a_i T_i(x) = \sum_{i=0}^{n} a_i \sum_{j=0}^{i} T_{i,j} x^j = \sum_{k=0}^{n} c_k x^k,\ c_k = \sum_{i=k}^{n} a_i T_{i,k}\ , \qquad (4.2)$$

получив тем самым частичную сумму обычного степенного ряда. Коэффициенты разложений в степенные ряды на каждом подинтервале вычисляются заранее с необходимой точностью и хранятся в отдельных таблицах. Такой подход увеличивает объем памяти, зато существенно увеличивает и скорость вычисления функции.

### 4.2.2 Минимизация погрешности вычислений на каноническом интервале

При вычислении элементарных функций с помощью разложения в ряд возникает три рода погрешностей:

1) погрешность, вызываемая использованием вместо бесконечного ряда его конечной частичной суммы. Эту погрешность можно оценить сверху с помощью ранее приведенной формулы (4.1) оценки остаточного члена;
2) погрешность, вызываемая использованием вместо истинных значений коэффициентов полиномов их машинных приближений;
3) погрешность округления результатов арифметических операций.

Важным фактором в организации машинных вычислений частичной суммы ряда является порядок суммирования, поскольку он определяет величину погрешности третьего рода. Так как заранее известен вид элементарной функции и ее поведение на интервале, то можно выбрать максимально эффективный порядок суммирования, при котором сначала складываются две меньшие величины, а потом к ним последовательно добавляются все большие слагаемые.

Разные члены вносят разный вклад в сумму ряда, обычно тем меньший, чем больше номер члена ряда. Оптимальным, таким образом, является суммирование частичной суммы, начиная с меньших по модулю членов высокого порядка к большим по модулю членам более низкого порядка. В таком случае ошибки округления, внесенные на более ранних этапах вычисления, становятся незначительными на более поздних этапах вычисления. В результате, общая погрешность вычислений рода 3) формируется практически исключительно при добавлении члена первого порядка и свободного члена. Заметим, что такому порядку вычисления соответствует известная и вычислительно эффективная (минимум операций с плавающей точкой) схема Горнера.

Для корректного осуществления вычисления верхней (нижней) границы элементарной функции с помощью разложения в ряд к частичной сумме ряда необходимо добавить (отнять) две дополнительных поправки: первая оценивает сверху погрешность первого рода (остаточный член ряда), вторая – погрешность второго рода, возникающую из-за использования вместо истинных коэффициентов Чебышева и Тейлора их машинных приближений. Учет погрешности третьего рода при вычислениях с направленными округлениями произойдет автоматически.

Первую поправку можно сделать сколь угодно малой за счет выбора большого количества членов ряда в разложении. Величина второй поправки определяется разностью между истинным и точным значением наибольшего по величине из членов разложения. На большинстве подинтервалов разложения таким членом разложения является свободный член. Так как вся сумма имеет порядок наибольшего по величине членов разложения, то отношение второй поправки к вычисляемой сумме всегда имеет порядок машинной точности. Значит, учет второго слагаемого способен негативно повлиять на ширину интервала-результата, так как расстояния между машиннопредставимыми числами тоже имеют порядок машинной точности. Для того чтобы избежать этого эффекта, предлагается использовать следующие формулы вычисления верхней и нижней оценки (при условии $x \geq 0$, включены соответственно флаг округления вниз и вверх):

```
Fr(x) = c + (((…(r[n])*x …)*x+r[1])*x+dr),        (4.3а)

Fs(x) = c + (((…(s[n])*x …)*x+s[1])*x+ds).        (4.3б)
```

Здесь **r** и **s** – массивы, содержащие приближения коэффициентов ряда разложения с округлением вверх и вниз соответственно, *dr* и *ds* – два последовательно идущих вещественных числа, отношение *dr* к *c* имеет порядок машинной точности и

$$c + dr \geq a_o \geq c + ds, \qquad (4.4)$$

где $a_o$ – истинное значение свободного члена разложения.

При организованных таким образом вычислениях свободный член учитывается со значительно более высокой точностью, так как он представляется суммой двух чисел существенно разных порядков. То есть, для него погрешность второго рода снижается до квадрата порядка машинной точности. В таком случае порядок погрешности второго рода равен разности между истинным и точным значением члена $a_1 x$, но $2^{-k} \geq x \Rightarrow \delta(a_1 x) \leq 2^{-k} * \delta(a_1)$, где $\delta(x)$ – погрешность представления в ЭВМ числа $x$. Таким образом, порядок погрешности второго рода есть $2^{-k-52}$. Ценой за достигнутую точность вычислений является одна дополнительная операция сложения, расходы на выполнение которой не слишком значительны по сравнению с общим временем вычисления элементарной функции.

Рассмотренный выше прием имеет смысл, если наибольший по величине вклад в частичную сумму вносит именно свободный член – тогда представление его в виде большей и меньшей части оправдано. В противном случае больший вклад вносит член при первой степени аргумента, и погрешность, вносимая им, превышает погрешность, вносимую неточным представлением свободного члена. На таких интервалах разбиения предлагается использовать следующие формулы:

```
Fr(x) = x + (((…(r[n])*x …)*x+r[1])*x+r[0]),
```
(4.5а)

```
Fs(x) = x + (((…(s[n])*x …)*x+s[1])*x+s[0]).
```
(4.5б)

Здесь на два слагаемых разбивается уже член первой степени: в качестве большего слагаемого используется **x**, а в качестве меньшего – **r[0]*x** или **s[0]*x**. Таким образом, коэффициент $a_1$ при $x$ представляется суммой

$$1 + s[0] \leq a_1 \leq 1 + r[0], \qquad (4.6)$$

старший член, которой есть в точности единица. Определенное отличие от формулы (4.4) имеет под собой три основания:

- рассматриваемые функции в областях, где свободный член меньше первого члена, достаточно малы и хорошо приближаются именно функцией $x$. Действительно, все они, кроме $\ln(x)$, имеют вид $x + O(x^3)$ в окрестности своих корней;
- представление члена первой степени в виде суммы большей и меньшей части $c*x$ и $d*x$ ведет к тому, что возникает погрешность третьего рода при вычислении произведения $c*x$, чего можно избежать при $c = 1$;
- если отказаться от лишнего умножения на коэффициент, можно сэкономить время вычислений.

Такой способ вычисления функции $f(x)$ подходит для подинтервалов, на которых колебание функции превышает колебание функции $x - f(x)$.

Существенным недостатком разложения в ряд Чебышева является тот факт, что частичная сумма ряда Чебышева обеспечивает наилучшее приближение на интервале в смысле абсолютного отклонения, но не в смысле относительного отклонения, которое наиболее важно, если в качестве критерия точности используется ширина интервала. Поэтому на интервалах, где относительная погрешность существенно превышает абсолютную погрешность, то есть в малой окрестности нуля функции, используется разложение в ряд Тейлора. Для этого достаточно использовать разложение в ряд Тейлора на единственном интервале $[0,2^{-k}]$ для функций $\sin(x), tg(x), \arcsin(x), arctg(x)$, у которых $f(x) = 0 \Rightarrow x = 0$, и на двух интервалах $[1-2^{-k},1],[1,1+2^{-k}]$ для $\ln(x)$, $\ln(x) = 0 \Rightarrow x = 1$.

Заметим, что вычисление этих функций в нуле с помощью разложения в ряд Чебышева привело бы к плачевным результатам – максимальное отклонение приближения Чебышева от истинного значения достигается как

раз на концах интервала приближения, и в малой окрестности нуля относительная погрешность была бы слишком большой. Кроме того, есть еще один аргумент в пользу применения на этом интервале разложения в ряд Тейлора – все функции кроме $\ln(x)$ разлагаются в ряд Тейлора в окрестности нуля только по нечетным степеням, что существенно экономит время вычислений, которые осуществляются по формулам:

```
Fr(x) = (((…(r[n])*x*x …)*x*x+r[0])*x*x)*x+x,
```
(4.7а)

```
Fs(x) = (((…(s[n])*x*x …)*x*x+s[0])*x*x)*x+x.
```
(4.7б)

В этих формулах учет погрешности первого и второго типа сделан не в свободном члене, а в коэффициентах **r[0] и s[0]**, что возможно, так как остаточный член ряда Тейлора оценивается как $|f(x)-L_n(x)|\le C_n * x^n$, где $C_n$ – константа.

В результате, предлагается вычислять значения элементарных функций следующим образом:

1) В интервалах, содержащих нуль функции, по формулам (4.7а) и (4.7б);
2) В остальных интервалах, на которых колебание функции $f(x)$ превышает колебание функции $x - f(x)$, по формулам (4.5а) и (4.5б);
3) В остальных интервалах по формулам (4.3а) и (4.3б).

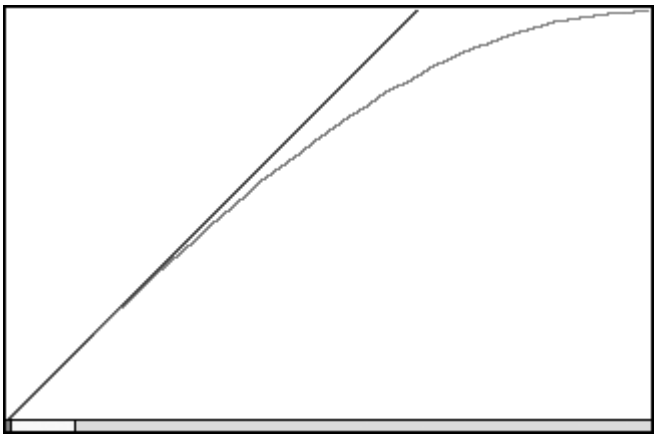

Рисунок 4.1. Три интервала с разными видами разложения $\sin(x)$

На рис. 4.1 на примере функции $\sin(x)$ (для ориентировки приведен также график функции $f(x)=x$) изображены слева направо величины промежутков, на которых применяются эти три разных способа разложения.

### 4.2.3 Минимизация погрешности вычислений при приведении аргумента

Вычисления значений всех функций для значений аргумента, выходящих за пределы канонического интервала, требуют применения тех или иных формул приведения. Основной трудностью в них является корректное вычитание полных периодов тригонометрических функций для значений аргумента, достаточно удаленных от нуля. Так как число $\pi$ является трансцендентным, при вычитании кратных ему чисел всегда будет вноситься ошибка округления, можно лишь пытаться сделать ее меньшей. Выберем некоторый достаточно большой интервал, на котором значение тригонометрических функций желательно вычислять с высокой точностью (например, в осуществленной автором реализации – $[-10^9,10^9]$), а за пределами этого интервала высокая точность вычислений не требуется.

**Определение 4.5** Пусть дано $a \in R$, $2^q \le a < 2^{q+1}$ и набор $\{p_0,..,p_m\} \in R_{FP}$ чисел с мантиссой длины не более $l$, $2^{q-l*i} \le p_i < 2^{q+1-l*i}$, причем $\sum_{i=0}^{m} p_i \le a < \sum_{i=o}^{m} p_i + 2^{q-l*(m+1)}$. Такой набор $\{p_0,..,p_m\}$ назовем $l$-представлением числа $a \in R$ длины $m+1$.

**Предложение 4.1** $\forall a \in R$, его $l$-представление длины $m+1$ существует и единственно.

**Доказательство** Если $m=0$, то надо доказать, что есть ровно одно $p_0 \in [2^q,2^{q+1})$ с мантиссой длины не более $l$, такое что $p_0 \le a < p_0 + 2^{q-l}$. Но разница между числами $p_0, p_0' \in [2^q,2^{q+1})$ с мантиссой длины не более $l$, между которыми нет другого такого числа, составляет $2^{q-l}$. Значит, весь

интервал $[2^q,2^{q+1})$ можно разбить на $2^l$ непересекающихся интервалов длины $2^{q-l}$, $a\in[2^q,2^{q+1})$ попадет ровно в один из таких интервалов, нижнюю границу которого и обозначим $p_0$. Такое $p_0$ удовлетворяет всем условиям и единственно по построению. Если же $m>0$, то воспользуемся индукцией, и найдем $l$-представление $\{p_1,..,p_m\}$ длины $m$ для числа $a-p_0\in[2^{q-l},2^{q-l+1})$, которое единственно.

**Предложение 4.2** Пусть $\{p_0,..,p_m\}$ есть $(52-k)$–представление числа $a\in[2^q,2^{q+1})$ длины $m+1$ и дан любой аргумент $x\in(-a*2^k,a*2^k)$. Тогда величину приведенного на ЭВМ аргумента

$$y=(..((x-p_on)-p_1n)-..p_mn), \tag{4.8}$$

где $n: x\in[an,a(n+1))$, можно вычислить без ошибок округления, и его отличие от истинного приведенного аргумента оценивается формулами

$$y\geq(x-an)>y-n*2^{q-(52-k)*(m+1)}. \tag{4.9}$$

**Доказательство** Для отсутствия ошибок округления надо, чтобы все вычисляемые величины полностью представлялись в ЭВМ. Для этого необходимо, чтобы все $p_in$ были машиннопредставимыми, а для этого достаточно, чтобы сумма длин мантисс $p_i$ и $n$ не превосходила 52, что верно по условию. В силу оценки из определения 4.5 $\sum_{i=0}^{m}p_i\leq a<\sum_{i=o}^{m}p_i+2^{q-(52-k)*(m+1)}$, откуда умножением на $(-n)$ и добавлением $x$ следует необходимая формула.

Таким образом, ошибку нахождения приведенного аргумента можно оценить величиной $n*2^{q-(52-k)*(m+1)}\leq 2^{q+k-(52-k)*(m+1)}$, где $q$ зависит только от вычитаемого периода $a$, а $k$ еще и от интервала $(-a*2^k,a*2^k)$, в котором выполняется приведение. Приведение аргумента функций производится согласно формулам

$$f(x+2\pi*n)=f(x), f\in\{\sin,\cos,tg,ctg\}$$

$$\exp(x+n*\ln 2)=\exp(x)*2^n$$

$$\ln(x*2^n)=\ln(x)+n*\ln(2)$$

В случае $\ln(x)$ приведение аргумента вычитанием периодов не требуется, достаточно просто рассмотреть мантиссу аргумента. Для других функций подстановкой конкретных данных вытекают следующие утверждения.

**Предложение 4.3** Для задачи приведения тригонометрического аргумента из интервала $[-10^9,10^9]$ достаточно взять $k=28$, $l=24$ и найти 24-представление $\{p_0,p_1,p_2,p_3\}$ числа $2\pi$ длины 4. Тогда вычисление приведенного аргумента по формуле (4.8) выполнимо без ошибок округления и с абсолютной точностью $2^{2+28-24*4}=2^{-66}$, которая будет превышать относительную точность $2^{-52}$ для чисел вне интервала $[-2^{-14},2^{-14}]$. Таким образом, если результат приведения лежит вне этого интервала, то вычисленный приведенный аргумент есть минимальное машиннопредставимое число, большее истинного приведенного аргумента.

**Предложение 4.4** Для задачи приведения аргумента экспоненты из интервала $x\in[-750,750]$, достаточно взять $k=11$, $l=41$ и найти 41-представление $\{p_0,p_1\}$ числа $\ln 2$ длины 2. Тогда вычисление приведенного аргумента по формуле (4.8) выполнимо без ошибок округления и с абсолютной точностью $2^{-1+11-41*2}=2^{-72}$, которая будет превышать относительную точность $2^{-52}$ для чисел вне интервала $[-2^{-20},2^{-20}]$. Таким образом, если результат приведения лежит вне этого интервала, то вычисленный приведенный аргумент есть минимальное машиннопредставимое число, большее истинного приведенного аргумента.

### 4.2.4 Другие вычислительные аспекты

В качестве примеров приведем алгоритмы вычисления некоторых конкретных элементарных функций.

**Вычисление функции синус**

Разложение синуса в ряд проводится на интервале $[0,\pi/2]$. Этот интервал разбивается на подинтервалы длиной $2^{-k}$, на каждом подинтервале $x\in[i/2^k,(i+1)/2^k]$ с помощью преобразования $t=2^k*(x-i/2^k)$ задача о разложении синуса сводится к задаче о разложении функции $\sin(2^{-k}*t+i/2^k)$ на каноническом интервале $t\in[0,1]$. Пусть, например, **k = 7**, тогда число подинтервалов длины $1/128$ равно 202. Для этой длины интервала достаточно использовать разложение в ряд Чебышева до пятой степени включительно. Пусть теперь, например, **k = 7** и **i = 30**, то есть $\sin(x)$ разлагается на промежутке $x\in[30/128,31/128]$. После вычисления верхних и нижних оценок коэффициентов Чебышева и составления для вычислений верхней и нижней границы синуса коэффициентов разложения в степенной ряд по формуле (4.2), получаем формулы для приведенного аргумента $t=128*(x-30/128)$**:**

```
sin+(x)  =  (((((8.09787061255167 2648e-03*t
+9.67653050324012 5813e-03)*t   -1.62109946646884 0788e-01)*t   -
1.16117559305255 8991e-01)*t      +9.72659678244912 5067e-01)*t
+1.95231113836677 7654e-17) + 0.23223511861151 1443,

sin-(x)  =  -((((((-8.09787061255167 0913e-03*t   -
9.67653050324012 4079e-03)*t      +1.62109946646884 1066e-01)*t
+1.16117559305255 9130e-01)*t   -9.72659678244912 3957e-01)*t   -
1.94245858163322 2228e-17) -0.23223511861151 1443),

sin(x) = [sin-(x), sin+(x)]
```

Для вычисления синуса от аргументов вне интервала $[0,\pi/2]$ надо использовать приведение аргумента согласно предложению 4.3. Вычисление функции синус для интервального значения аргумента состоит в том, чтобы найти глобальные минимум и максимум на заданном интервале. Понятно,

что если ширина интервального значения аргумента $x$ больше, чем $2\pi$, то $\sin(x) = [-1.0, 1.0]$. Если ширина интервала-аргумента меньше, чем $2\pi$, то определяются значения аргумента $x_{\min}$ и $x_{\max}$, в которых достигаются соответственно минимальное и максимальное значения функции синус на данном интервале. Это можно сделать, используя свойство кусочно-монотонности функции. Результатом вычисления синуса для невырожденного интервального аргумента будет интервал, нижняя граница которого равна нижней границе значения синуса в точке $x_{\min}$, а верхняя – верхней границе значения синуса в точке $x_{\max}$.

**Вычисление логарифма**

Разложение логарифма в ряд проводится на интервале $[0.75,1.5]$. Так как $\forall x \exists! y \in [0.75,1.5), m \in Z : x = 2^m y$, то вне этого интервала применяется формула $x=2^m y$ для выбора подходящего значения $y$ из интервала разложения, что легко сделать без ошибок округления, используя мантиссу $x$, и вычисляя $\ln(x)=\ln(y)+m\ln(2)$. В качестве интервала разложения можно выбирать и другие интервалы вида $[s,2s]$, но выбор интервала $[0.75,1.5]$ обладает двумя преимуществами:

- он содержит значение $1.0$, на котором логарифм обращается в нуль, что позволяет вычислять логарифм в окрестности $1.0$ с высокой относительной точностью;
- в силу простого представления чисел $\{0.75,1.5\}$, мантисса которых имеет единичную длину, легко найти $y : x=2^m y$.

В силу того, что логарифм является монотонно возрастающей функцией, его вычисление для невырожденного интервального аргумента сводится к вычислению нижней границы на левом конце и верхней границы на правом конце интервала-аргумента.

**Вычисление экспоненты**

Разложение экспоненты в ряд проводится на интервале $[0,\ln 2]$. Вне этого интервала используется приведение аргумента $x$ к $y \in [0,\ln 2]$ согласно предложению 4.4, а затем применяется соотношение $\exp(x)=2^m \exp(y)$. Вычисления по последней формуле проводятся без привнесения ошибки округления. Так как экспонента растет очень быстро, представимы лишь ее значения для аргументов $[\ln(2^{-1075}),\ln(2^{1075})] \subseteq [-750,750]$, для которых приведение можно проводить по формулам (4.8).

**Вычисление квадратного корня**

Поскольку на современных компьютерных платформах затраты на вычисление квадратного корня малы (они примерно равны затратам на вычисление операции деления), автором было принято решение использовать для вычисления верхних и нижних границ этой функции специальную технику. Для вычисления нижней границы $sqrt(x)$ используется значение $t$, возвращаемое стандартной математической библиотекой C, проверяется, является ли оно нижней границей с помощью вычисления $t*t$ с округлением вверх. Если результат не превосходит аргумента, то $t$ является нижней границей, иначе выполняется сдвиг: берется минимальное машиннопредставимое число, меньшее $t$ и снова выполняется проверка возведением в квадрат. Так выполняется несколько итераций, пока условие выполнено. Вычисление верхней границы проводится аналогичным способом с помощью сдвига вверх. Как показали эксперименты, для вычисления границ ни разу не пришлось выполнять более одного сдвига.

**Использование одного флага округления**

Реализация флагов округления на многих компьютерных платформах такова, что время, затрачиваемое на изменение флагов округления, сравнимо со временем, затрачиваемым на вычисление элементарной функции как частичной суммы ряда Чебышева или Тейлора. В связи с этим разумно выставлять флаг округления один раз, до проведения вычислений, а

вычисления верхней и нижней границ функций организовать так, чтобы они производились при одном и том же флаге округления. Например, если выставлен флаг округления “округлять вверх”, и $x \geq 0$, то вычисление верхней границы элементарной функции происходит, как обычно, по формуле (4.3а), а вычисление нижней границы происходит по следующей модифицированной формуле:

**Fs(x) = c-(((…(-s[n])*x ……)*x-s[1])*x-ds)**, (4.10)

где **s[i]** – коэффициенты полинома Чебышева, посчитанные для определения нижней границы. Легко, видеть, что эта формула эквивалентна формуле (4.3б) при предположении бесконечной точности вычислений.

Использование функций вычисления верхних и нижних границ с одним и тем же флагом округления позволяет эффективно осуществлять большие объемы вычислений, что характерно для интервальных методов. Кроме того, во избежание влияния вызова функций математической библиотеки на дальнейшие вычисления, все функции математической библиотеки должны иметь дубликаты, которые дополнительно меняют флаг округления на значение “округлять вверх” перед началом вычисления элементарной функции и меняют флаг округления на значение “округлять к ближайшему значению” после завершения вычисления.

## 4.3 Интервальные методы решения задач в ограничениях

### 4.3.1 Описание метода CP распространения ограничений.

Использование интервальных методов с гарантированными вычислениями для решения произвольной задачи возможно в рамках программирования в ограничениях. Основным объектом этой концепции является задача удовлетворения ограничений, которые, вообще говоря, могут быть произвольными подмножествами декартова произведения доменов переменных. Ограничение может быть задано явным перечислением допустимых означиваний переменных, уравнением, неравенством, таблицей,

логической формулой, внешней пользовательской функцией [63]. В силу общности такого определения ограничения, многие задачи в области PLM являются частным случаем задачи удовлетворения ограничений [47], к их числу относится и задача параметрического проектирования.

Процесс решения задач удовлетворения ограничений состоит из двух фаз – удаления областей значений переменных, заведомо не содержащих решения, и разбиения оставшихся областей на меньшие подобласти. Описание задачи является чисто декларативным, то есть оно задается просто набором ограничений, без необходимости определять алгоритм для нахождения решений [77]. Это позволяет легко использовать решатель задач удовлетворения ограничений для решения задач параметрического проектирования. Необходимым требованием является лишь возможность выполнить интервальную оценку всех используемых функций, что легко гарантировать, если отношения задачи выражены в элементарных математических функциях [23].

Основными методами программирования в ограничениях являются методы сужения недоопределенных оценок, или методы распространения ограничений [24]. Далее будет описан представитель этого класса, метод **CP,** подходящий для непрерывных областей переменных, характерных для задач параметрического проектирования**.** Другим представителем класса методов сужения интервальных недоопределенных оценок является интервальный метод Ньютона, но его описание выходит за рамки диссертационной работы.

Пусть математическая модель задана системой из $m$ алгебраических уравнений от $n$ переменных:

$$\begin{cases} f_1(x_1,..,x_n) = 0 \\ .. \\ f_m(x_1,..,x_n) = 0 \end{cases} \qquad (4.11)$$

Множество всех решений такой системы обозначается $P=\{\mathbf{x}\in R^n \mid \mathbf{F}(\mathbf{x})=\mathbf{0}\}$. Будем считать, что $P$ есть подмножество некоторого ограниченного

параллелепипеда (или *бруса*) $P=[l_1,r_1]\times..\times[l_n,r_n]$, то есть рассматривать только решения $\{\mathbf{x}\in R^n \mid \forall i, x_i \in [l_i,r_i]\}$. Если решение задачи имеет какой-то физический или геометрический смысл, то на практике нужны лишь решения такого рода.

Так как в общем случае *i*-е уравнение системы содержит не все переменные, то его можно записать в следующем виде:

$$f_i(x_{i,1},..,x_{i,n(i)})=0, \tag{4.12}$$

где $n(i)\le n$. Предположим, что из этого уравнения можно выразить каждую переменную через другие, тогда можно получить $n(i)$ следующих соотношений:

$$x_{i,j}=f_i^j(x_{i,1},..,x_{i,j-1},x_{i,j+1},..,x_{i,n(i)}),\ j\in\{1,..,n(i)\} \tag{4.13}$$

**Определение 4.6** Соотношения (4.13) называются *соотношениями интерпретации*, а их правые части – *функциями интерпретации*.

Записав в таком виде все уравнения исходной системы, получим набор из $\sum_{i=1}^{m} n(i)$ соотношений интерпретации, определяющих в общем случае для каждой переменной $j$ набор из $m(j), 1\le m(j)\le m$ ее представлений в виде функций интерпретации.

Итак, пусть $\{\mathbf{x}\in R^n \mid \forall i, x_i \in [l_i,r_i]\}$. Опишем работу метода **CP,** сужающего первоначальные оценочные интервалы $X_j=[l_j,r_j]$ для каждой переменной $x_j$. Итерационный процесс уточнения значений переменных состоит в получении все более узких оценок $X_j^k=[l_j^k,r_j^k]$ на значения переменной $x_j$. Не ограничивая общности, равенства с функциями интерпретации можно переписать для интервальных типов данных с операциями интервальной математики, и получить новые интервальные оценки:

$$X_{i,j}^{k+1}=f_i^j(X_1^k,...,X_{j-1}^k,X_{j+1}^k,...,X_{n(i)}^k) \tag{4.14}$$

Используя разложение исходных функций (4.11) на элементарные функции и операции, такого рода оценки можно получить почти для всех практически важных функций.

**Определение 4.7** Методом СР называется следующий итерационный процесс:

1) для $k=0$ полагается $\forall j, X_j^k=[l_j,r_j]$, создается *множество активных функций*, которое содержит все интервальные функции интерпретации;
2) вычисляются все $\sum_{i=1}^{m} n(i)$ активные функции на аргументах $X_j^k$ и получаются новые оценки $X_{i,j}^{k+1}$ по формулам (4.14);
3) выполняются пересечения интервальных оценок
$$X_j^{k+1}=X_j^k\cap(X_{1,j}^{k+1}\cap..\cap X_{m,j}^{k+1}); \qquad (4.15)$$
4) $k:=k+1$. Из множества активных функций удаляются функции интерпретации, аргументы которых не изменились на шаге 3. Аргумент считается изменившимся, если ширина оценочного интервала сузилась более чем на заранее заданный $\varepsilon$.
5) Если множество активных функций пусто, то процесс вычислений заканчивается, иначе – переход на шаг 2.

Таким образом, значения переменных в процессе итерационных вычислений представляются последовательностью нерасширяющихся интервалов $X_j\supseteq X_j^1\supseteq..\supseteq X_j^k$. Такая последовательность сходится, причем порядок выполнения функций интерпретации не влияет на окончательный результат вычислений [32] (при работе метода с идеальной точностью $\varepsilon=0$).

В результате применения итерационного процесса будет получен многомерный параллелепипед $D\subseteq P$. Этот параллелепипед с гарантией содержит все решения системы (4.11). Если $D=\varnothing$, то система *не совместна* – ее множество решений пусто.

### 4.3.2 Описание интервального метода бисекции.

Алгоритмы распространения ограничений не позволяют в общем случае сузить области значения переменных до точных решений задачи. Для дальнейшего отсечения областей, в которых решения отсутствуют, используется комбинация метода бисекции, осуществляющего дробление области, с методом сужения для подобластей.

Принцип работы метода бисекции прост – на каждой своей итерации он производит деление области возможных значений одной из переменных $x_j$ модели на две части, порождая тем самым две подмодели, множество переменных и ограничений которых совпадает с множеством переменных и ограничений исходной задачи. Затем метод бисекции инициирует запуск двух процессов распространения ограничений на порожденных подзадачах, а после сужения областей производит рекурсивный запуск себя на каждой из подзадач.

Поведение метода бисекции во многом определяет используемая им стратегия выбора переменных для деления. Для вещественных переменных характерны различные кусочно-монотонные зависимости (тригонометрические функции, степенная функция, арифметические операции). Если после деления область возможных значений переменных лежит в интервале монотонности функции интерпретации, то ее применение дает точную оценку области значения переменной. Поэтому из нескольких вещественных переменных для деления метод бисекции обычно выбирает переменную, у которой самая широкая область возможных значений. При этом под шириной подразумевается:

1) для интервалов, включающих нуль – абсолютная ширина интервала, то есть разница между верхней и нижней границей;

2) для всех других интервалов – относительная ширина интервала, то есть отношение ширины интервала к модулю ближней к нулю границы интервала.

Метод бисекции прекращает свое выполнение, когда все интервальные оценки во всех рассматриваемых подмоделях становятся “достаточно определенными”, то есть в тот момент, когда оценочные интервалы всех вещественных переменных имеют ширину меньшую, чем некоторая заранее заданная константа. При применении метода бисекции для решения задачи параметрического проектирования $Z = (\tilde{V}, \tilde{E}, G, S_0, F, \varepsilon)$ целесообразно использовать точность $\varepsilon$.

Результатом работы метода бисекции является набор брусьев – интервальных оценок решений, при этом гарантируется, что любое из решений исходной системы содержатся в одном из таких брусьев. Этот набор называется *интервальным решением* комбинации методов СР и бисекции.

Достаточно часто в задачах с вещественными переменными встречается эффект “ползущих корней” – наличие большого количества находящихся близко друг от друга квазирешений с малой невязкой [28]. Для метода бисекции множеством решений является набор интервальных оценок, каждая из которых является “достаточно определенной”. Если представить каждое решение малым многомерным брусом, тогда эффект “ползущих корней” заключается в наличии нескольких близкорасположенных, часто соприкасающихся брусьев-квазирешений, соответствующих одному-единственному истинному корню исходной системы. Понятно, что такое представление одного решения несколькими брусьями неудобно, и пользователь реальной системы желал бы получить вместо набора брусьев один брус, возможно, большего размера.

Поэтому автором была разработана техника склейки близкорасположенных корней. Одним из очевидных требований к этой технике была необходимость различения эффекта “ползущих корней” от

ситуации, когда задача недоопределена и существует целое многообразие истинных решений, которое не надо объединять в одно решение. Примером может служить множество решений модели, состоящей из одного ограничения $x = y$, а начальные оценки переменных равны $[0,1]$. Интервальное решение такой системы комбинацией методов СР и бисекции содержит интервалы вида $[a_i, b_i] \times [a_i, b_i], (b_i - a_i) \in [0, \varepsilon]$, причем $\bigcap_i [a_i, b_i] = [0,1]$, и, возможно, еще какие-то интервалы. Наивная техника, которая склеивает два корня, если расстояние между ними мало (или даже равно нулю, как в данном случае), склеила бы все корни в один брус $[0,1]$, то есть вернулась бы к исходной интервальной оценке. Предлагаемая автором техника, свободная от этого недостатка, основывается на понятии доли решений в брусе.

**Определение 4.8** Пусть рассматривается содержащее $q$ брусьев интервальное решение комбинации методов **СР** и бисекции $R = \{I_1, .., I_q\}, \forall k \in \{1, ..q\}, I_k = [l_1^k, r_1^k] \times .. \times [l_n^k, r_n^k]$. Объемом любого бруса $I = [l_1, r_1] \times .. \times [l_n, r_n]$ назовем $V(I) = (r_1 - l_1) * .. * (r_n - l_n)$. Долей решения $R = \{I_1, .., I_q\}$ в произвольном брусе $I*$ назовем число $D(R, I*) = \sum_{k \in \{1,..,q\}} V(I_k) / V(I*)$.

**Определение 4.9** Будем называть обертывающим брусом для $R = \{I_1, .., I_q\}$ минимальный по включению брус $I*: \forall k, I_k \subseteq I*$. Объединенным $c$-решением для интервального решения $R$ называется:

1) Обертывающий брус $I'$ для подмножества $R' = \{I_{i1}, .., I_{in}\}$, для которого $D(R', I') \geq c$. Заметим, что всегда $D(R, I') \geq D(R', I')$;
2) Обертывающий брус $I*$ для некоторого множества $\{I_1, .., I_r\}$, каждый элемент которого $I_i$ есть объединенное $d_i$-решение, причем

$$\sum_{i \in \{1,..,r\}} d_i * V(I_i) \Big/ V(I*) \geq c. \qquad (4.16)$$

Величина $c$ – это заранее заданная константа, контролирующая, насколько легко будут склеиваться близкорасположенные брусья-решения.

При $c=1$ брусья будут склеиваться, только если их теоретико-множественное объединение тоже является брусом.

Предложенная техника склеивания описывается следующим алгоритмом:

1) Пусть дано множество брусьев $R=\{I_1,..,I_q\}$, являющееся интервальным решением, каждому элементу $I_i$ припишем число $d(I_i)=1$, породим множество $P$ всех разных пар брусьев $\{I_i,I_j\}$;
2) Берем любую пару $\{I_i,I_j\}$ из множества $P$ и проверяем, является ли обертывающий брус $I*$ для $(I_i,I_j)$ объединенным $c$-решением;
3) Если это так, то удалим $\{I_i,I_j\}$ из $R$ и добавим туда $I*$, вычислив $d(I*)$ как левую часть формулы (4.16). Также $\forall k$ удалим из $P$ пары $\{I_i,I_k\}$, $\{I_k,I_j\}$ и добавим пару $\{I_k,I*\}$, перейдем на шаг 2).
4) Иначе удалим пару $(I_i,I_j)$ из $P$. Если $P$ не пусто, то переход на шаг 2), иначе конец.

Проверка того, что брус является $c$-решением, требует лишь вычисления объемов брусьев, так как величины $d(I_i)$ известны. Эта операция, как и порождение обертывающего бруса, имеет сложность $O(n)$, где $n$ – число переменных. Шаг 2) выполнится не более $O(q^2)$ раз, так как любое выполнение следующих за ним шага 3) и 4) уменьшает количество пар в $P$. Итого, сложность алгоритма составляет $O(n*q^2)$.

Как легко видеть, предложенная техника не будет склеивать, например, множество решений $x=y$, где $x,y\in[0,1]$, в один брус, так как в процессе склеивания доля корней в объединенных решениях будет снижаться, и, рано или поздно, станет меньшей, чем некоторая заранее заданная константа $c$. В то же время, для большинства случаев такая техника позволяет избавиться от эффекта ползущих корней.

### 4.3.3 Применение для задач параметрического проектирования

Так как интервальные методы решения систем нелинейных уравнений основаны на методах поиска, таких, как бисекция, то они имеют экспоненциальную вычислительную сложность. В связи с этим они значительно менее применимы на практике, чем численные методы типа метода Ньютона, и не могут быть основным способом решения задач параметрического проектирования в промышленном решателе.

Тем не менее, интервальные методы способны решать такие задачи, которые не может решить подход, основанный на методе Ньютона:

1) Доказательство несовместности задачи;
2) Решение задач с интервальными данными.

Например, рассмотрим задачу вписывания равностороннего треугольника $T = (A_1, A_2, A_3)$ со стороной $a = 11$ в квадрат $D = (D_1, D_2, D_3, D_4)$ со стороной $b = 10$ (см. рис. 4.2). Эта задача, очевидно, является двухмерной задачей параметрического проектирования, включающей два указанных тела, шесть переменных $p_i, q_i, i \in \{1,2,3\}$, шесть знаковых ограничений расстояния вида $d(A_i, L_1) = p_i$, $d(A_i, L_2) = q_i$, и шесть неравенств $0 \le p_i \le 10$, $0 \le q_i \le 10$.

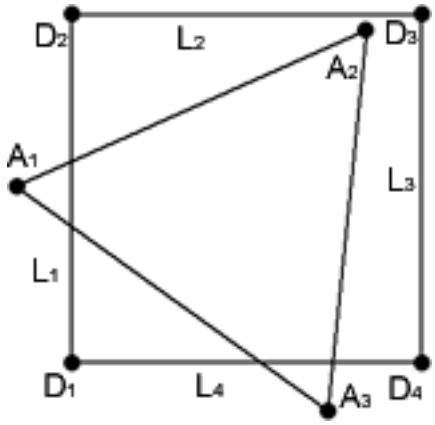


Рисунок 4.2. Задача вписывания треугольника в квадрат

Задача является несовместной, так как максимальное значение длины стороны треугольника, при котором он может быть вписан в квадрат со

стороной 10, есть $(10/\cos(\pi/12)) \sim 10.35$. Применение для нее обычных методов решения задач параметрического проектирования не дает никакой информации. Действительно, решение не может быть найдено, а диагностировать несовместность задачи обычными методами не получается, поэтому время будет просто потрачено впустую.

Эта задача, кроме того, является по своей сути интервальной, поскольку заданы интервальные оценки величин $p_i, q_i \subseteq [0,10]$. При работе связки методов СР и бисекции такие означивания естественным образом учитываются как начальные интервальные оценки при первом запуске метода СР. В решателе задач параметрического проектирования, основанного на классических методах типа метода Ньютона, такие ограничения моделируются сложнее, например, уравнениями вида $(p_i - 5)^2 + t_i^2 = 5^2$, где $t_i$ – дополнительные вещественные переменные, что увеличивает размер решаемой системы уравнений. Другой возможный способ учета этих ограничений состоит в урезании вектора приращения, считаемого на каждой итерации метода Ньютона, если его применение ведет к выходу за пределы указанных интервалов, что способствует ухудшению сходимости метода Ньютона.

Проверка несовместности задачи интервальными методами может быть также не самоцелью, а средством уменьшения времени решения, потраченного впустую обычными методами на несовместной задаче. Дело в том, что с целью решения как можно большего числа задач в геометрических решателях применяется механизм вычислительных ветвей, позволяющий при неуспехе одной ветви решить задачу одной из следующих ветвей. К сожалению, это ведет к увеличению общего времени решения, причем больше всего – на несовместных задачах, на которых последовательно запускаются все вычислительные ветви.

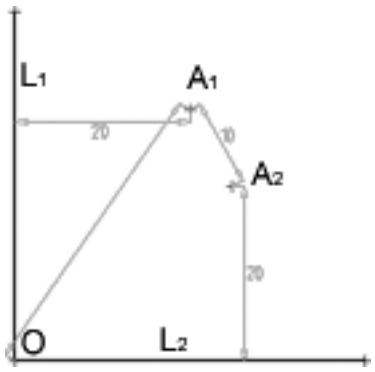


Рисунок 4.3. Параметры расстояний могут быть несовместными

Например, на рис. 4.3 описана двухмерная задача позиционирования двух точек $A_1, A_2$ относительно системы координат с центром $O$ и осями абсцисс и ординат $L_2, L_1$. Это хорошо определенная задача, в ней задан набор расстояний $d_1(A_1, L_1) = 20$, $d_2(A_2, L_2) = 20$, $d_3(A_1, A_2) = 10$, $d_4(A_1, O) = c$, где $c$ - некоторая константа. При $c < 10\sqrt{5}$ эта задача несовместна, поскольку из ограничений $d_2, d_3$ по метрическим свойствам следует $d(A_1, L_2) \geq 10$, а $d_4(A_1, O) = \sqrt{d(A_1, L_1) + d(A_1, L_2)} \geq 10\sqrt{5}$. С помощью интервального решателя несовместность такой задачи быстро обнаруживается простым сужением интервалов методом CP, без применения бисекции.

Этот пример иллюстрирует более общий подход, заключающийся в применении интервальных решателей для обнаружения таких видов несовместности, как нарушения правила треугольника и прочих соотношений на расстояниях между объектами. Использование интервальных методов для обнаружения несовместности позволило бы сократить время, потраченное на решение заведомо несовместных задач.

Поскольку метод CP имеет полиномиальную временную сложность [32], его можно применять и в промышленных решателях задач параметрического проектирования. В более сложных случаях, тем не менее, доказать несовместность не получается без использования метода бисекции. В таком случае для контроля времени работы предлагается применять полиномиальный вариант метода бисекции, который ограничен по числу одновременно обрабатываемых подзадач.

## 4.4 Выводы

Использование интервальных методов вызывает необходимость разработки соответствующей математической библиотеки. По этой причине были разработаны и реализованы методы вычисления гарантированных верхних и нижних границ элементарных математических функций с помощью направленных округлений на основе разложений в ряды Чебышева и Тейлора. Эти методы позволяют контролировать различные погрешности, появляющиеся при вычислении функций на ЭВМ. На основе проведенных исследований была разработана интервальная математическая библиотека на C с простым и универсальным интерфейсом, которая может быть использована отдельно от геометрического решателя. Теоретические работы по методам вычисления элементарных функций автор проводил совместно с Т.П. Кашеваровой, практическая реализация выполнена им единолично.

Кроме того, были проведены исследования в области интервальных методов программирования в ограничениях, предложена схема работы метода бисекции, позволяющая избежать эффекта “ползущих корней”. Рассмотрены вопросы применения интервальных методов к решению задач параметрического проектирования и сделан вывод, что они не должны применяться в качестве основного метода решения. Их рационально использовать для:

1) решения задач с интервальными данными;
2) решения проблемы определения совместности задачи;
3) быстрого обнаружения несовместности и экономии времени вычислений для обычных задач параметрического проектирования.

Практическая разработка интервальных методов программирования в ограничениях производилась автором в большом коллективе, он участвовал в разработке метода CP (в части вычисления функций интерпретации и оптимизации очереди активных функций), реализация усовершенствованного метода бисекции проведена им единолично.

# 5. Аспекты программной реализации и практического использования

## 5.1 Реализованные программные компоненты

В ходе работы над диссертацией автор участвовал в разработке и реализации следующих программных компонент, так или иначе используемых при решении задачи параметрического проектирования:

1) LGSEP – решатель систем нелинейных уравнений, характерных для задач параметрического проектирования (объем исходного кода – 733К);
2) LEMO – универсальный решатель систем уравнений с недоопределенными данными (1795К);
3) LGS 2D – решатель двухмерных задач параметрического проектирования (2417К);
4) LGS 3D – решатель трехмерных задач параметрического проектирования (2849К);
5) Imath – библиотека вычисления элементарных математических функций с направленными округлениями (760К).

Взаимодействие между ними организовано по следующей схеме:

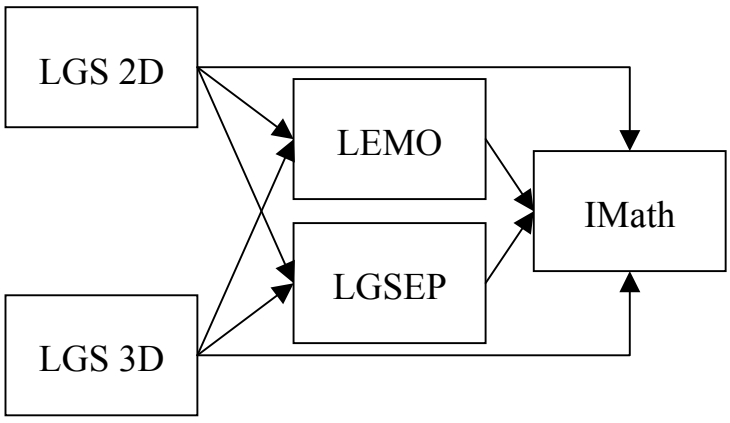


Рисунок 5.1. Общая схема взаимодействия компонент

При этом поток исходных данных идет по стрелкам от начала к концу, а поток результирующих данных от конца стрелок к началу. Так, исходная геометрическая модель попадает либо в двухмерный решатель LGS 2D, либо в трехмерный LGS 3D. После обработки ее внутренними модулями этих программных компонент – модулем геометрической декомпозиции, упрощения геометрической модели, а также модулем алгебраического моделирования – задача преобразуется в систему нелинейных уравнений. Эту систему можно подать на вход либо решателю LGSEP, оптимизированному под класс задач параметрического проектирования, либо универсальному решателю LEMO, основанному на методах распространения ограничений и позволяющему работать с недоопределенными данными. Для решения индустриальных задач использование LGSEP значительно более эффективно, но в некоторых случаях, в частности, при работе с интервальными данными, необходимо использовать LEMO. Для вычислений элементарных математических функций LEMO во избежание потери решений применяет библиотеку IMath, кроме того, любая другая программная компонента может использовать функции этой библиотеки вместо стандартных.

Комплекс этих программных компонент позволяет эффективно решать как двухмерные, так и трехмерные задачи параметрического проектирования. Каждая программная компонента состоит из нескольких модулей. Рассмотрим более подробно архитектуру этих компонент.

## 5.2 Архитектура решателей LGS 2D и 3D

### 5.2.1 Объекты интерфейса и их трансляция во внутреннее представление

Взаимодействие прикладных программ с LGS осуществляется через программный интерфейс LGS API, который является набором экспортируемых C-функций [48]. Геометрические тела не являются отдельными сущностями, а задаются как атрибуты содержащихся в них объектов. Объекты и ограничения в LGS разделяются на внешние и

внутренние. Внешние объекты и ограничения создаются и используются через LGS API, внутренние объекты и ограничения, а также кластеры и контейнеры объектов образуют GSIR – внутреннее представление LGS. Как внутренние, так и внешние объекты и ограничения образуют непересекающиеся иерархии C++-классов. Внутреннее представление является универсальным, в то время как внешнее нужно для наиболее удобного для конкретного пользователя способа описания геометрической задачи. Такое разделение позволяет при необходимости изменить внешний набор объектов и ограничений, не затрагивая вычислительные механизмы и внутреннее представление геометрической модели. Внутренние объекты и ограничения образуют гиперграф, вершинами которого являются объекты, а гиперребрами – ограничения.

Связывающие различные единицы локализации данных части кода делятся на трансляцию внешних объектов во внутренние объекты, моделирование задачи параметрического проектирования системой нелинейных уравнений, получение решения системы с помощью внешнего решателя, последовательное восстановление координат внутренних, а затем внешних объектов.

В настоящее время в решателях LGS есть один стандартный набор внешних объектов и транслятор, позволяющий переводить информацию из внешнего представления во внутреннее, и наоборот. Тем не менее, в процессе разработки применялись и существенно другие наборы внешних объектов, так что указанную выше схему трансляции можно считать апробированной для разных программных интерфейсов. При трансляции некоторые внешние объекты могут переводиться в совокупность внутренних объектов. Например, набор внешних объектов может не содержать прямых, но содержать ломаные, распадающиеся при трансляции на внутренние объекты – точки и прямые. Далее о добавлении (или удалении) внутренних объектов и ограничений уведомляется менеджер вычислительных ветвей.

В модель может вноситься последовательно произвольное число изменений, но они вызывают лишь запуск процедуры трансляции, без применения сложных вычислительных методов. Вообще, решатели LGS 2D и 3D устроены так, чтобы в наибольшей степени поддерживать *инкрементальность* вычислений. Под этим свойством имеется в виду способность последовательно выполнять сложные сценарии создания и редактирования модели с наименьшими общими затратами. Таким образом, минимизируется не время счета некоторой конкретной задачи параметрического проектирования, а время работы над набором задач, последовательно получаемых друг из друга. Для этого применяются следующие подходы:

1) Эвристическая склейка тел, между которыми нет неудовлетворенных ограничений (ригидификация);
2) Переиспользование деревьев подзадач, построенных геометрической декомпозицией, и других структур данных;
3) Программный интерфейс представлен набором элементарных функций, большинство из которых имеет минимальное время обработки.

### 5.2.2 Схема использования эвристических вычислительных ветвей.

При запуске модели на счет начинает работу менеджер вычислительных ветвей, который последовательно вызывает вычислительные ветви, начиная с наиболее быстрых, и останавливается при первом получении решения. Рассмотрим более подробно схему использования вычислительных ветвей, которую в дальнейшем будем называть схемой эвристик.

Каждая вычислительная ветвь состоит из одного или нескольких применяемых методов, составляющих вместе некоторый способ решения задачи параметрического проектирования. Примером может служить

эвристическая ветвь ригидификации, эффективная для задач с малым числом неудовлетворенных ограничений. При ее работе объекты, между которыми нет неудовлетворенных ограничений, склеиваются в одно тело, после чего происходит решение с помощью обычного алгебраического моделирования. Получаемое решение сохраняет многие взаимные положения тел, что важно с точки зрения пользователя.

Далее используется декартово моделирование, в котором тела могут лишь сдвигаться, но не вращаться. Этот способ моделирования далеко не всегда способен решить поставленную задачу и тоже является эвристическим. Его применение разумно, так как он позволяет за счет генерации меньшей (и более простой) системы уравнений для некоторых задач получить решение гораздо быстрее, чем другие методы моделирования. Кроме того, полученные им решения не содержат вращений, и поэтому тоже весьма ценны с точки зрения пользователя.

В качестве эвристической ветви используется и метод декомпозиции Хоффманна. Говоря формально, так как задача параметрического проектирования является NP-сложной, а любой промышленный решатель не может позволить себе экспоненциальный рост времени вычислений при масштабировании задачи, то любой способ решения задачи в LGS 2D является эвристическим. Действительно, применяемый после алгебраического моделирования метод Ньютона не гарантирует нахождение решения даже при неограниченном числе итераций.

Эвристические вычислительные ветви упорядочиваются в соответствии с двумя характеристиками – ожидаемым временем решения и *процентом успеха*. Под процентом успеха имеется в виду процентное отношение количества тестов, которые успешно решаются данной вычислительной ветвью, к общему количеству тестов. Ожидаемое время решения есть отношение времени, потраченного на обработку всех тестов данной вычислительной ветвью, к общему количеству тестов. Запуск

вычислительных ветвей в менеджере устроен так, чтобы ветви были упорядочены по убыванию отношения процента успеха к ожидаемому времени решения. Такой подход к упорядочиванию ветвей тоже является эвристическим: для двух разных вычислительных ветвей события успешного решения некоторого теста не являются независимыми, поэтому наиболее эффективное упорядочивание все равно приходится искать перебором, оценивая общее время решения тестов для различных порядков вычислительных ветвей.

### 5.2.3 Механизм кластерных типов

Все вычислительно сложные эвристические ветви используют геометрическую декомпозицию, разбивающую исходную задачу на дерево подзадач (кластеров). При запуске геометрической модели на счет и вызове ветви с геометрической декомпозицией сначала происходит анализ того, изменила ли задача свою структуру со времени последнего запуска метода декомпозиции. Под изменением структуры понимается изменение множества объектов и ограничений задачи или связей между ними, но не изменение параметров объектов (координат, радиусов) и ограничений (значений углов, расстояний). Если изменения структуры не произошло, то дерево кластеров никак не меняется. Если же изменения структуры имели место, то запускается модуль геометрической декомпозиции, который пытается разбить задачу на подзадачи и перестроить соответствующим образом дерево кластеров, которое задает порядок решения подзадач.

Каждый выделенный кластер ассоциируется с наиболее точно соответствующим узлом в иерархии кластерных типов [49]. Кластерный тип – это C++ класс, представляющий собой множество геометрических моделей с общей спецификой, которые можно решать одним и тем же вычислительным методом. Запуск этого метода осуществляется виртуальной функцией Calculate(). Каждый кластерный тип имеет функцию isMatched(), позволяющую определить, является ли конкретный кластер представителем

данного кластерного типа, или нет. Кластерные типы как множества геометрических моделей организованы в древовидную структуру отношением включения, такую же структуру имеет иерархия кластерных типов как C++ классов. В ходе определения наиболее точно соответствующего некоторому кластеру кластерного типа происходит безоткатный поиск в глубину в этом дереве, в результате которого определяется кластерный тип наиболее специального вида, для которого данный кластер является представителем. В процессе совершенствования вычислительного аппарата LGS 2D происходило и будет происходить пополнение дерева кластерных типов новыми узлами, соответствующими новым эффективным специализированным вычислительным методам.

Функции Calculate() для простейших кластерных типов вызывают аналитические вычисления. При расположении простейшего объекта по двум ограничениям, каждое их которых снимает одну степень свободы, всегда используются аналитические методы как более точные. Для некоторых более сложных кластеров, например, содержащих два тела и три ограничения между ними, разумно использовать заранее построенный базис Гребнера и одноперменные аналитические и численные методы. Для еще более сложных кластеров используется алгебраическое моделирование и численные методы решения систем нелинейных уравнений, основным из которых является метод Ньютона. Также применяется эвристический метод декартова моделирования, ищущий решения без поворотов тел, градиентный метод безусловной оптимизации для энергетической целевой функции, которая равна сумме квадратов невязок всех уравнений, и другие методы.

Каждый вычислительный метод Calculate(), используемый для решения некоторого кластерного типа, организует свою работу как последовательность нескольких (предположительно, большого числа) шагов. После выполнения каждого из шагов он может быть прерван с помощью опроса специальной булевой функции, а потом возобновлен с помощью нового запуска Calculate(). Для быстрейших из функций Calculate(),

например, аналитических вычислений, работа метода состоит из одного шага, в то время как наиболее сложные методы способны останавливаться после каждой итерации (например, метод Ньютона).

### 5.2.4 Функциональность решателей LGS

Решатели LGS позволяют оперировать со следующими геометрическими объектами:

- Точки, прямые, окружности; в двухмерных задачах – эллипсы;
- Плоскости, цилиндры, сферы в трехмерных задачах;
- Пользовательские кривые и, в трехмерных задачах, поверхности;

Набор поддерживаемых ограничений включает:

- Совпадение, расстояние, углы, касание, радиусы;
- Вертикальность, горизонтальность, фиксация;
- Равенство расстояний/радиусов, симметрия, средняя точка (для двухмерных задач);

Параметры ограничений могут быть связаны произвольными уравнениями, заданными аналитическим способом или через вызов функции пользователя.

Помимо основной функциональности – решения задачи параметрического проектирования – LGS 2D способен выполнять и другие функции, алгоритмическая основа которых не включена в теоретическую часть диссертационного исследования. В их число входят:

- Движение с сохранением наложенных ограничений;
- Диагностика конфликтующих ограничений;
- Доопределение задачи параметрического проектирования до хорошо определенной задачи;
- Оптимизация параметров задачи;

- Поддержка дискретных ограничений (таблиц и перечислений).

В поддержке всех этих функций автор принимал непосредственное участие как разработчик или руководитель.

## 5.3 Решатель систем нелинейных уравнений LGSEP

Вычислительное ядро LGSEP разрабатывалось для решения произвольных дифференцируемых систем нелинейных уравнений. LGSEP состоит их следующих модулей:

1) LGSEPProcessor – модуль-менеджер, который управляет работой других модулей, осуществляет поддержку совместности данных;
2) LGSEPDecomposer – модуль декомпозиции Далмейджа-Мендельсона;
3) LGSEPEquationReduction – модуль сокращения уравнений;
4) LGSEPSolver – модуль решения систем нелинейных уравнений и задач условной оптимизации итерационными методами;
5) LGSEPGenerator – модуль чтения/записи данных для тестирования LGSEP как отдельной программной единицы;

Входными данными LGSEP служат: система уравнений, стартовое приближение, желаемая точность вычислений, опционально – список методов решения и их настроек.

Основным объектом LGSEP является абстрактный класс уравнения LGSEPEquation. В этом классе определяются основные функции, необходимые различным методам для работы с уравнением: вычисление в конкретной точке невязки и градиента невязки, фиксирование переменной, определение списка фактических переменных. Существует большое число производных по отношению к LGSEPEquation классов, представляющих специальные типы уравнений: квадратное уравнение, тригонометрическое уравнение, произвольное аналитически задаваемое уравнение, уравнение, существующее как объект на стороне пользовательского приложения, и т. д. Наиболее часто используемым является класс LGSEPQuadraticEquation,

представляющий квадратное уравнение от многих переменных. Генерируемые LGS уравнения являются, как правило, разреженными, поэтому квадратное уравнение представлено списком ненулевых термов.

LGSEPProcessor – это основной модуль, с которым взаимодействует пользовательская программа. Он позволяет пользователю решать систему уравнений с нужной точностью, задавая при этом начальное приближение. Его работа организована так, что он сначала декомпозирует систему уравнений на подсистемы с помощью модуля LGSEPDecomposer, а затем пытается решить подсистемы, используя сначала модуль LGSEPEquationReduction, а потом модуль LGSEPSolver. Если обнаруживается, что при решении некоторой подсистемы нужная точность не достигнута, то делается попытка решить систему уравнений целиком. При решении подсистем используется большая точность, чем при сборке решения всей системы из частных решений, что позволяет контролировать ошибки округления.

Модуль LGSEPEquationReduction использует простейшие методы сокращения системы уравнений: решение линейных и квадратных уравнений от одной переменной, выражение одной из переменных линейного уравнения через другие, распознавание сумм полных квадратов и т.п.

Модуль LGSEPSolver содержит методы решения систем алгебраических уравнений, в том числе:

- SVD и REF методы решения систем линейных алгебраических уравнений;
- метод релаксации и метод Ньютона решения произвольных систем алгебраических уравнений.

Метод Ньютона используется в модифицированном варианте – при достаточно большой невязке системы обращение якобиана применяется один раз в пять итераций с целью сокращения времени решения.

## 5.4 Аспекты программной реализации в LEMO

Наиболее затратным по объемам используемой памяти методом этой программной компоненты является метод бисекции. Кроме того, именно качество работы бисекции во многом определяет время решения задачи, поэтому остановимся подробнее на том, как реализован этот метод.

В процессе его работы необходимо хранить бинарное дерево рассматриваемых подзадач, отличающихся только интервальными оценками переменных. Хранение набора таких оценок в виде единой таблицы позволяет эффективно порождать таблицы для подзадач быстрым копированием всей таблицы как одного объекта и изменением оценки всего лишь одной переменной в подзадаче. Переход между подзадачами легко производить изменением указателя на текущую таблицу, никак не затрагивая переменные и ограничения, которые являются едиными для всех подзадач объектами. Учитывая этот факт, в LEMO для стратегий обхода дерева подзадач в глубину было реализовано представление данных подмоделей в виде стека таблиц. В таком случае при откате легко построить таблицу недоопределенных значений “брата” некоторого узла, зная лишь таблицы интервальных оценок этого узла и его непосредственного родителя.

Несмотря на все вышесказанное, стеки таблиц обычно занимают достаточно большое количество памяти. На практике таблицы интервальных значений узла и его непосредственного родителя отличаются мало – имеет место эффект “локального распространения ограничений”. Следовательно, вместо хранения всей таблицы можно хранить лишь список изменившихся оценок. Если считать измененными только те переменные, у которых ширина оценочного интервала по определению 4.4 сократилась не менее, чем в некоторое константное число $c$ раз (в текущей реализации, $c = 1.001$), то одна и та же переменная может попасть в список измененных не более чем $\log_c 2^{64}$ раз ($2^{64}$ – это количество машиннопредставимых чисел).

Тем самым можно гарантировать, что количество памяти, используемое для хранения оценок переменных всех подзадач, линейно зависит от числа переменных. Правда, эта оценка верна лишь тогда, когда сохраняется единственная таблица первоначальных оценок переменных модели, относительно которой регистрируются все изменения оценок. В таком случае при откате на достаточно высокий уровень в дереве будет необходимо пробежать достаточно большую часть этого списка. Чтобы избежать этого, в LEMO была избрана комбинированная стратегия: таблицы интервальных оценок переменных сохраняются в узлах, глубина которых $n*k$ в дереве кратна некоторой целочисленной константе $k$ (в текущей реализации $k=16$), список измененных значений для узла формируется по отношению к ближайшему прародителю глубины кратной $k$.

## 5.5 Конечно-пользовательские приложения решателей LGS

### 5.5.1 Клиент-серверная технология FlashLGS и веб-приложение на ее основе

С целью популяризации систем параметрического проектирования на базе LGS была разработана клиент-серверная технология FlashLGS. Эта технология позволяет клиентским приложениям на Adobe Flash на разных платформах и устройствах удаленно использовать LGS для решения двухмерных задач параметрического проектирования. Идея состоит в том, чтобы дать возможность использовать LGS с помощью “тонких“ клиентских приложений, перекладывая сложные вычисления на сервер, что является нетрадиционным подходом для САПР. Более того, благодаря популярности Adobe Flash как компоненты веб-браузера, такое веб-приложение можно использовать без предварительной установки на большинстве компьютеров (и других устройств) с выходом в Интернет.

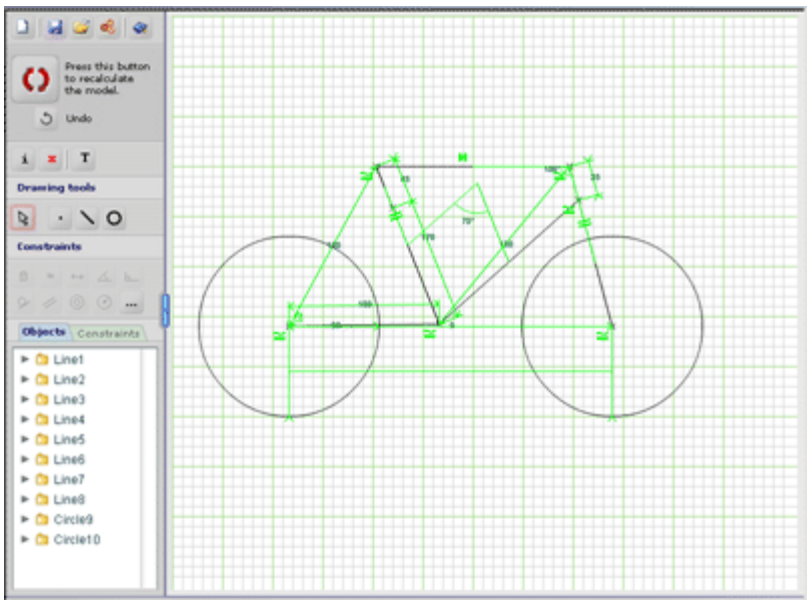


Рисунок 5.2. Параметрическая модель велосипеда в FlashLGS

В среде FlashLGS Симоновым В. А. под руководством автора было реализовано интерактивное веб-приложение для вариационного черчения [30]. Редактор позволяет работать с точками, окружностями, отрезками и ограничениями 16 разных видов. Пользователю предоставляется набор стандартных параметрических моделей, возможность создавать, редактировать и сохранять свои модели в персонализированных каталогах в XML-формате. Для минимизации передачи данных через Интернет был разработан специальный компактный формат обмена данными.

Работающее приложение доступно в сети Интернет по адресу

http://www.ledas.com/products/lgs2d/flash_lgs

Это веб-приложение предназначено, в первую очередь, для демонстрации возможностей систем параметрического проектирования. В силу такого предназначения оно может быть полезным для обучения инженерным специальностям. Начиная с 2007 года, FlashLGS используется при обучении студентов НГТУ.

### 5.5.2 Использование LGS 3D как вычислительного ядра САПР ADEM-VX

Примером удачного практического применения решателя LGS 3D является его использование в российской индустриальной интегрированной системе ADEM-VX автоматизации проектирования и производства. Среди нескольких продуктов этой системы LGS 3D более всего использовался в продукте для сборочного проектирования ADEM Assembly.

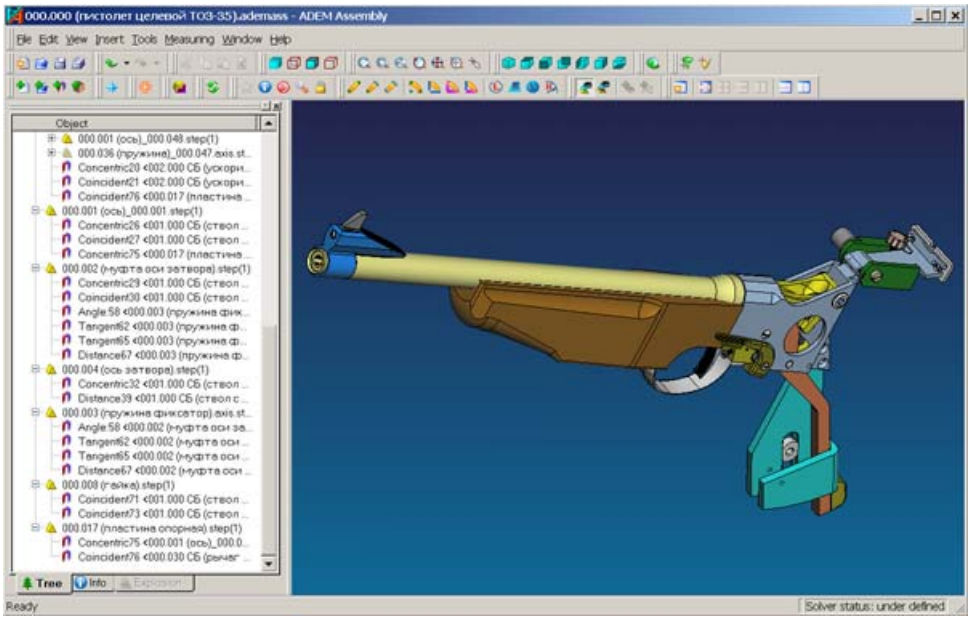


Рисунок 5.3. Параметрическая модель пистолета в ADEM Assembly

Интеграция LGS 3D была успешно произведена в 2004 году без изменений в его программном интерфейсе, после чего на протяжении трех лет производилась техническая поддержка. Необходимость моделирования в ADEM задач с цилиндрическими объектами потребовала поддержки в LGS 3D нового объекта – цилиндра. Такая поддержка была обеспечена в краткие сроки за счет расширяемости архитектуры LGS 3D и используемых в нем методов обработки задач параметрического проектирования.

### 5.5.3 Интеграция с САПР bCAD

Примером удачного практического применения решателя LGS 2D является внедрение его в САПР bCAD. Приведенный ниже рис. 5.4 взят из

работы [19], посвященной системе bCAD, в которой освещается и вопрос интеграции системы с LGS 2D. Этот рисунок показывает, как одна из деталей, в которой допустимы различные стандартные варианты наборов параметров, параметризируется с помощью LGS 2D, позволяя тем самым конструктору вместо хранения набора деталей с разными значениями параметров легко получать деталь с заданными характеристиками.

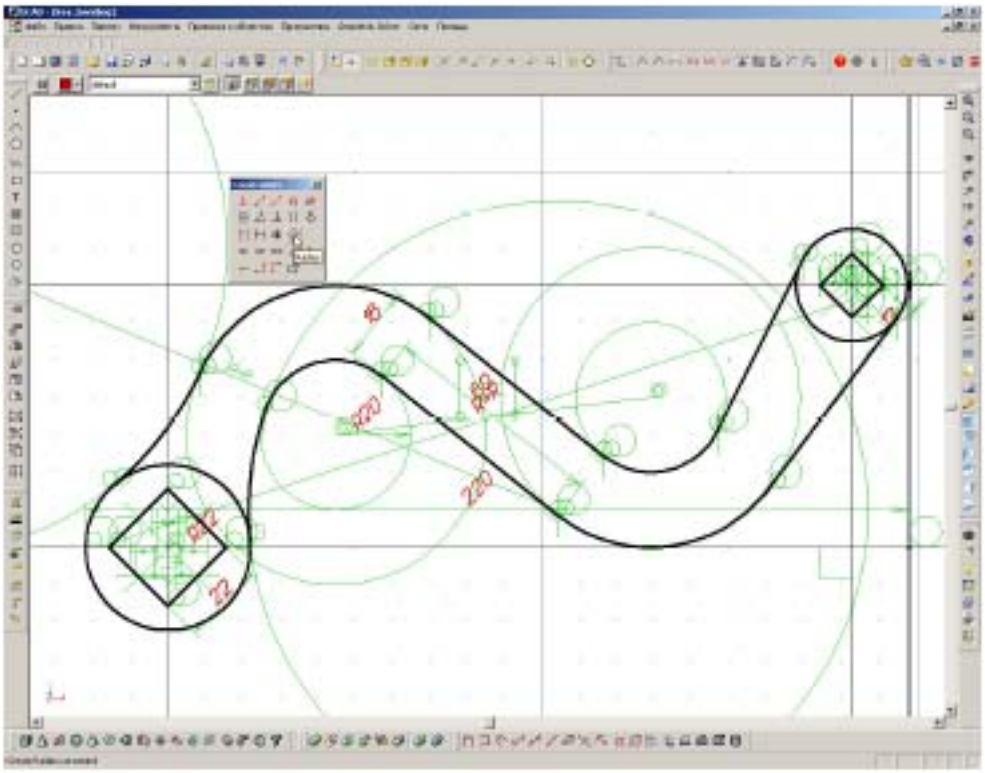

Рисунок 5.4 Использование LGS 2D для параметризации модели

### 5.5.4 Другие примеры интеграции

В области двухмерных задач параметрического проектирования одним из практических результатов является интеграция LGS 2D в современное приложение для параметрического черчения, являющееся частью системы автоматизированного производства Masterwork. Эта система разработана итальянской компанией Tecnos G.A. Srl., производителем программного обеспечения для станков ЧПУ, и позволяет порождать код для таких станков, используя двух- и трехмерное геометрическое моделирование. Приложение

построено на основе графического ядра OpenCASCADE и предназначено, в первую очередь, для нужд мебельной промышленности.

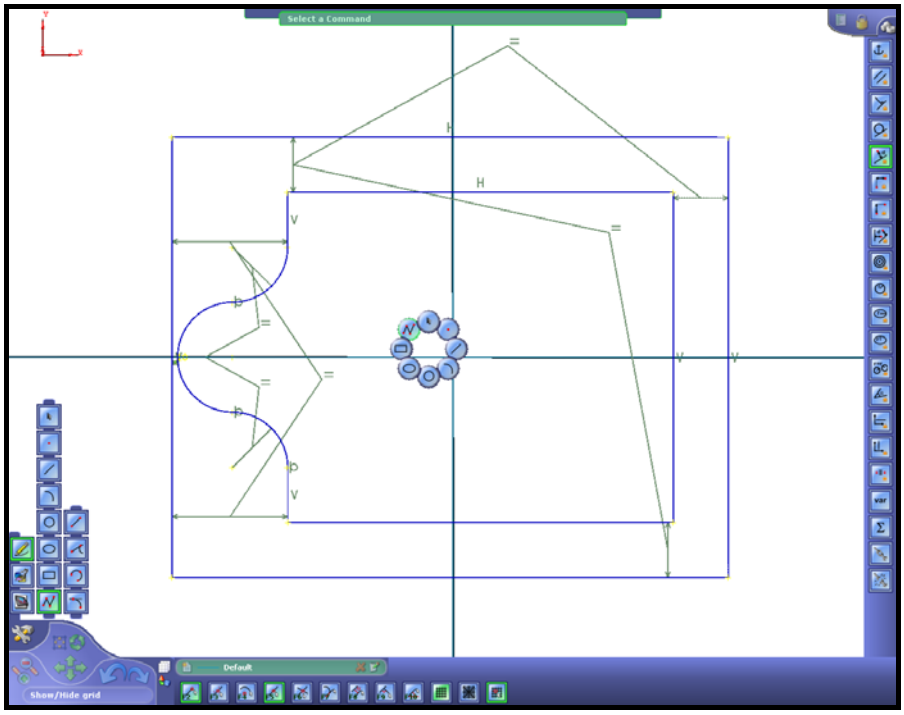


Рисунок 5.5 Использование LGS 2D в системе Masterwork

Интеграция решателя LGS в данный продукт была выполнена в трехмесячный срок. Большая часть этого времени была потрачена на вопросы интеграции кода обоих программных продуктов посредством графического ядра OpenCASCADE. Трудности, связанные с внедрением LGS 2D, касались не его производительности или способности решить задачу параметрического проектирования, а с проблемой выбора одного из нескольких возможных решений этой задачи. В связи с этим стоит считать вероятным направлением дальнейшего развития решателей LGS обеспечение на всех уровнях вычислительного ядра поддержки близости решения к эскизу в смысле задаваемых пользователем функций близости.

Еще одним практическим результатом является интеграция обоих решателей LGS 2D и LGS 3D в продукт ClassCAD швейцарской компании AWV, сотрудничающей с цюрихским Университетом Прикладных Технологий. ClassCAD является интегрированной средой для порождения конечно-пользовательских приложений в области САПР.

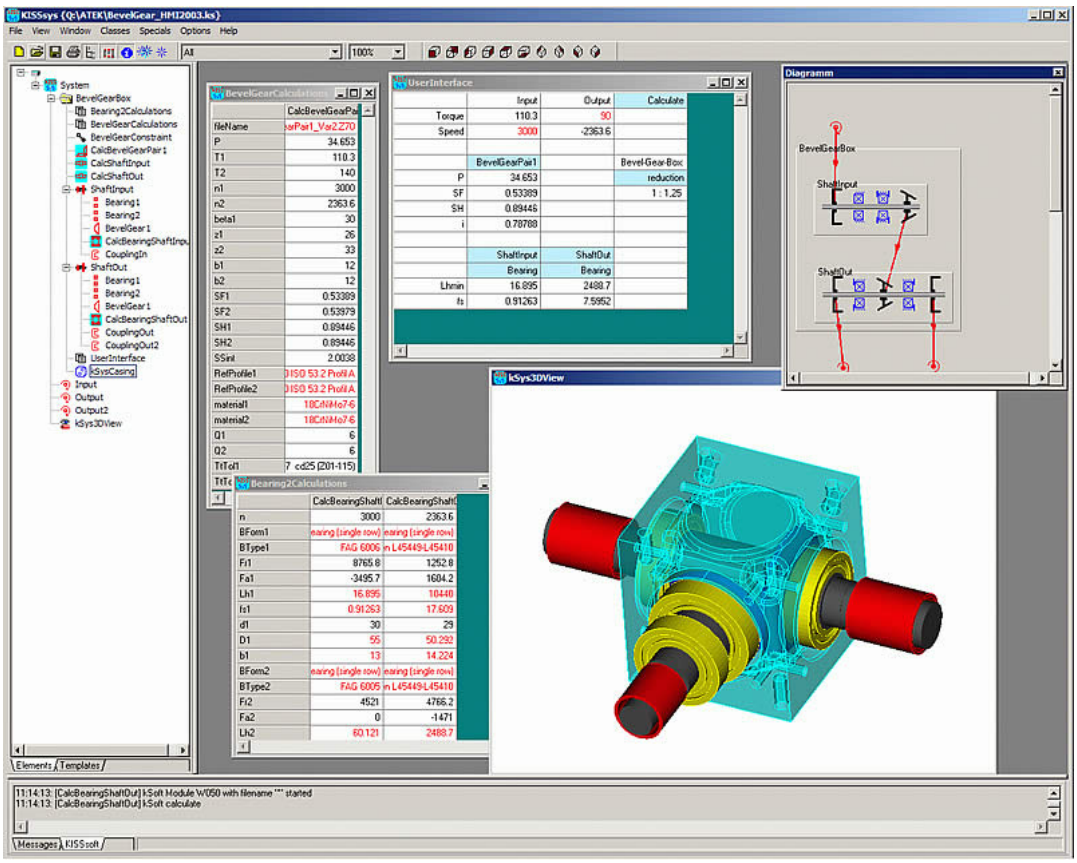

Рисунок 5.6 Интеграция LGS 3D в ClassCAD

## 5.6 Экспериментальные результаты реализации предложенных методов

### 5.6.1 Эффект от использования остовного моделирования

Тестовая база, использовавшаяся для оценки эффективности реализованного метода, включает в себя две группы трехмерных тестов: основную, состоящую из 3106 тестов, большинство из которых индустриальные, и базу больших тестов, включающую 14 наиболее сложных задач. Для сравнения использовались два метода моделирования: стандартное моделирование, в котором для каждого тела используется набор из 3 параметров вращения и 3 параметров сдвига, и остовное моделирование, в котором некоторые из этих параметров фиксируются за счет относительной параметризации тел. Результаты сравнения, полученные при работе разработанного под руководством автора геометрического решателя LGS 3D, показаны в таблице 5.1.

Таблица 5.1. Сравнение остовного моделирования с декартовым

| Тестовая база | Метод моделирования | Среднее количество переменных | Среднее число уравнений | Общее время решения | Процент решенных задач |
|---|---|---|---|---|---|
| Основная | Моделирование шестью параметрами | 8.48 | 7.12 | 695.05 | **95.10%** |
| Бол. тесты | | 29.57 | 37.25 | 242.02 | **71.42%** |
| **Всего** | | | | **937.07** | |
| Основная | Остовное моделирование | 5.46 | 3.04 | 187.13 | **97.52%** |
| Бол. тесты | | 16.46 | 18.25 | 115.12 | **78.57%** |
| **Всего** | | | | **302.25** | |

Как из нее видно, метод остовного моделирования по всем показателям превосходит метод стандартного моделирования:

1) Среднее количество переменных и уравнений в генерируемой нелинейной системе уменьшилось на обеих базах в 1.5-2.5 раза (см. также рис 5.7). Надо заметить, что размеры системы уравнений не так велики также и из-за работы методов геометрической декомпозиции;
2) Общее время решения всех тестов уменьшилось более чем в три раза, хотя эффект ускорения более заметен на основной базе тестов;
3) На обеих базах тестов число успешно решенных тестов у метода остовного моделирования выше.

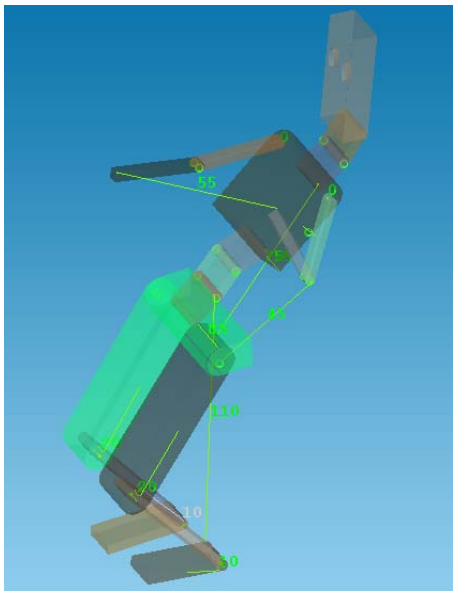

Рисунок 5.7. Модель с 22 ограничениями требует всего 9 уравнений

Приведенное выше сравнение двух методов однозначно указывает на полезность применения остовного моделирования. Тем не менее, в реальных геометрических решателях, в том числе в использованном для сравнения решателе LGS 3D, как правило, применяется последовательно несколько подходов на основе разных методов моделирования. Поэтому для корректной оценки эффекта от применения нового метода на практике необходимо сравнить характеристики поведения геометрического решателя в его обычной конфигурации с запуском нескольких вычислительных ветвей, до и после внедрения метода остовного моделирования. Это сравнение проводилось на той же тестовой базе, результаты приведены в таблице 5.2.

Таблица 5.2. Эффект от внедрения метода остовного моделирования

| Тестовая база | Конфигурация решателя | Общее время решения | Процент решенных задач |
|---|---|---|---|
| Основная | До подключения остовного моделирования | 77.89 | **98.55%** |
| Большие тесты | | 478.18 | **85.71%** |
| **Всего** | | **556.07** | |
| Основная | После подключения остовного моделирования | 52.78 | **99.16%** |
| Большие тесты | | 235.98 | **92.85%** |
| **Всего** | | **288.75** | |

Таким образом, практический эффект от реализации метода остовного моделирования положителен по обеим характеристикам:

1) Общее время решения всех тестов уменьшилось примерно в два раза, причем коэффициент ускорения на больших тестах выше;
2) Примерно в два раза уменьшилось количество тестов, которые LGS 3D не в состоянии решить.

Так как второй из показателей является, вообще говоря, более важным, чем первый, то именно в нем проявляется главный эффект от подключения остовного моделирования.

### 5.6.2 Результаты реализации метода отделяющей декомпозиции

Таблица 5.3. Эффект от использования отделяющей декомпозиции.

| Классы решаемых задач | Количество задач в классе | Процент решенных задач | | Время решения | | |
|---|---|---|---|---|---|---|
| | | Без отделяющей декомпозиции | С отделяющей декомпозицией | Без отделяющей декомпозиции | С отделяющей декомпозицией | **Коэффициент ускорения** |
| Двухмерные искусственные | 930 | 87.1% | 87.2% | 30.50 | 23.28 | **1.31** |
| Двухмерные промышленные | 4902 | 90.9% | 91.8% | 1058.74 | 150.72 | **7.02** |
| Двухмерные большие промышленные | 42 | 54.8% | 88.1% | 5015.06 | 72.12 | **69.54** |
| **Все двухмерные** | **5874** | **90.0%** | **91.0%** | **6104.30** | **246.12** | **24.80** |
| Трехмерные искусственные | 880 | 90.8% | 90.8% | 8.47 | 7.21 | **1.17** |
| Трехмерные промышленные | 2843 | 99.3% | 99.5% | 162.17 | 37.53 | **4.32** |
| Трехмерные большие промышленные | 5 | 100.0% | 100.0% | 81.80 | 22.03 | **3.71** |
| **Все трехмерные** | **3728** | **97.4%** | **97.6%** | **252.44** | **66.77** | **3.78** |
| **Все задачи** | **9602** | **93.0%** | **93.7%** | **6356.74** | **312.89** | **20.32** |

Эффект от реализации метода отделяющей декомпозиции в геометрическом решателе оценивался по двум показателям: количеству решенных тестов и времени вычислений. По обеим этим характеристикам метод отделяющей декомпозиции показал положительные результаты на всех рассмотренных классах задач, увеличив количество успешно решаемых тестов с 93.0% до 93.7% и уменьшив общее время решения с 6356.74 до 312.89 секунд. При оценке эффекта была использована тестовая база из 9602 тестов, разделенная на 3 класса двухмерных и 3 класса трехмерных задач:

1) искусственные, сконструированные для отладки решателя;
2) индустриальные – полученные из реальных моделей САПР, используемых в промышленности;
3) большие индустриальные, в которые вошли промышленные задачи, решаемые за время порядка 1 секунды и более.

Таким образом, наиболее существенный эффект от применения этого метода заключается в двадцатикратном общем ускорении решателя. Этот прогресс достигнут в основном за счет решения двухмерных задач: коэффициент ускорения для них составил 24.80 против 3.78 для трехмерных задач. Заметен опережающий рост производительности метода при увеличении размера модели – коэффициент ускорения для больших промышленных задач выше, чем для других классов задач. Также легко видеть, что метод особенно эффективен для решения промышленных задач, в то время как на искусственных задачах он достигает менее чем 50%-ного ускорения. Существенное увеличение процента успешно решаемых задач зафиксировано на классе двухмерных больших промышленных задач – с 54.8% до 88.1%.

Больший эффект разработанного метода декомпозиции на двухмерных задачах объясняется его спецификой. Двухмерные задачи, в отличие от трехмерных, часто получаются при работе в системах черчения, а не сборочного проектирования, поэтому для них характерна низкая встречаемость тел с более чем одним элементарным объектом. Метод отделяющей декомпозиции основан на поиске образцов, которые имеют наиболее простой вид для тел с одним объектом, именно поэтому такие образцы были поддержаны в первую очередь. Дальнейшее развитие метода отделяющей декомпозиции за счет подключения новых, более сложных образцов для тел с более чем одним объектом, должно увеличить производительность геометрического решателя трехмерных задач.

Пример действия метода отделяющей декомпозиции приведен на рис. 5.8 для задачи, полученной при проектирования в САПР бытовой электронной техники. Эта задача содержит эллипсы, окружности, прямые, точки и разные наложенные на них ограничения, в том числе тернарные ограничения симметрии. Исходную задачу из 51 тела и 81 ограничения нельзя было корректно разложить известными методами декомпозиции, но с помощью применения отделяющей декомпозиции это стало возможным. При работе отделяющей декомпозиции было успешно отделено 31 тело, после чего оставшаяся задача была разбита методом двусвязной декомпозиции на подзадачи, наибольшая из которых содержала 11 тел и 14 ограничений.

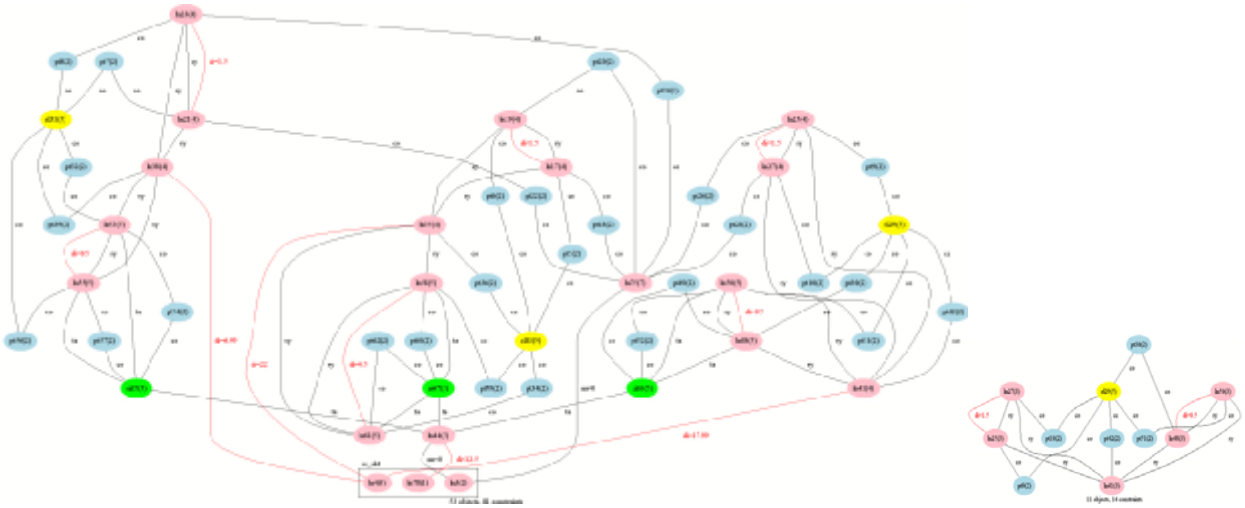

Рисунок 5.8. Пример действия метода отделяющей декомпозиции

### 5.6.3 Характеристики библиотеки вычислений элементарных функций

Для оценки качества реализованной библиотеки элементарных математических функций использовались три показателя: точность вычислений, время и количество необходимой памяти. Сравнение времени вычислений проводилось со стандартными библиотечными математическими функциями C. Каждая функция была вычислена 10 млн. раз на стандартном промежутке своего разложения с равномерным распределением аргументов. Суммарное время вычислений некоторых функций библиотеки в секундах приведено в таблице 5.4. Результаты позволяют говорить о практически

одинаковой производительности реализованной библиотеки и стандартной математической библиотеки C++.

Таблица 5.4. Сравнение производительности математических библиотек

| Функции | sin | cos | exp | arccos | arctg | **Всего** |
|---|---|---|---|---|---|---|
| Стандартные | 0.546 | 0.656 | 0.375 | 1.313 | 0.875 | **3.765** |
| Реализованные | 0.750 | 0.797 | 0.359 | 1.015 | 0.812 | **3.733** |

Так как разработанная библиотека является интервальной, то естественным способом оценки ее точности оказывается измерение ширины интервалов, являющихся результатами вычисления функций на точечных значениях. Ширину разумно измерять не в абсолютных величинах, а в величинах машинной точности, то есть в количестве машиннопредставимых чисел, содержащихся в интервале-результате. Тестирование показало, что в 80-90% вычислений интервальных функций математической библиотеки результат представляет собой два идущих подряд машиннопредставимых числа, то есть является неулучшаемым. Максимальная ширина интервала-результата составляет 2-5 машиннопредставимых числа.

Таблица 5.5. Параметры реализации вычисления разных функций

| Функции | Интервал разбиения | Число подинтер-валов | Количество членов в разложении | Число коэффици-ентов | Объем памяти, байт |
|---|---|---|---|---|---|
| arcsin и arccos | [0, sqrt(2)/2] | **91** | **7** | 1365 | 10920 |
| arctg и arcctg | [0, 1] | **128** | **7** | 1920 | 15360 |
| exp | [0, ln(2)] | **90** | **6** | 1170 | 9360 |
| sin и cos | [0, PI/2] | **202** | **6** | 2626 | 21008 |
| tg и ctg | [0, PI/4] | **101** | **7** | 1515 | 12120 |
| **Итого** | | | | **8596** | **68768** |

Параметры разложения элементарных функций в степенные ряды приведены в таблице 5.5. Как видно, для разложения всех функций используется 6-7 членов степенного ряда, то есть любая функция на каждом

подинтервале разложения представляется полиномом 5-6 степени. При генерации коэффициентов для функций на всех интервалах разбиения было проведено сравнение с разложением в ряд Тейлора. Оказалось, что если оставить количество членов в разложении тем же самым (6-7 членов), количество подинтервалов, необходимых для разложения функции в ряд с такой же точностью, увеличивается в 4-8 раз. Таким образом, применение рядов Чебышева позволяет существенно экономить память для хранения коэффициентов.

### 5.6.4 Оценка эффективности решателя LGS в сравнении с другим решателем

Решатели LGS 2D и 3D являются коммерческими программными продуктами и были лицензированы несколькими компаниями-разработчиками САПР. При встраивании LGS 3D в один из таких продуктов было проведено сравнение его с другим геометрическим решателем, ранее использовавшемся в этом продукте. Сравнение производилось на базе из 3120 трехмерных индустриальных тестов, разделенных на две группы: основную, включающую 3106 промышленных тестов, и группу из 14 наиболее сложных теста, каждый из которых решался несколько секунд. Результаты тестирования приведены в таблице 5.6.

Таблица 5.6. Сравнение LGS 3D с используемым в САПР решателем

| Группа тестов | Число тестов | Встроенный в САПР решатель | | LGS 3D | | |
|---|---|---|---|---|---|---|
| | | Процент решенных задач | Общее время решения | Процент решенных задач | Общее время решения | Отношение времен решения |
| Основная | 3106 | 83.32% | 307.53 | 99.32% | 63.40 | **4.85** |
| Большие тесты | 14 | 50.00% | 121.92 | 92.86% | 283.48 | **0.43** |
| **Все тесты** | **3120** | **83.17%** | **429.45** | **99.29%** | **346.68** | **1.24** |

Легко заметить, что LGS 3D существенно превосходит соперника по проценту успешно решаемых задач на обеих группах тестов. Время решения

больших тестов у LGS 3D хуже, однако этот эффект во многом вызван значительно большим процентом успешно решаемых больших тестов – 92.86% против 50.00%. Более того, суммарное время решения LGS 3D всех тестов на 24% лучше, чем у другого решателя.

## Заключение

В ходе исследований были получены новые результаты, полноценно изложенные в тексте диссертации. К теоретическим результатам относятся следующие:

1) Автором разработан новый метод отделяющей декомпозиции, относящийся к классу геометрических методов решения задач параметрического проектирования. Метод корректно декомпозирует задачи, имеет линейную вычислительную сложность и способен разбивать циклы зависимости ограничений;
2) Разработан метод остовного моделирования, относящийся к классу алгебраических методов решения задач параметрического проектирования, позволяющий для решения этой задачи использовать систему уравнений с меньшим числом переменных и уравнений. Метод применим для широкого класса задач, является расширяемым и вычислительно эффективным. Он был разработан совместно с Руколеевым Е.В., наибольший вклад автором внесен в разработку алгоритмов построения остовного представления;
3) Проведены исследования в области интервальных методов решения задач параметрического проектирования, в том числе разработан подход к вычислению интервальных элементарных математических функций, основанный на разложении в ряды Чебышева и Тейлора с направленными округлениями. Работа выполнялась в сотрудничестве с Кашеваровой Т.П., при этом методы приведения аргумента в канонический интервал и схема контроля погрешности вычислений разработаны автором единолично;
4) Автором выполнен обзор исследований в области подходов к решению задач параметрического проектирования, произведена критика рассмотренных подходов, в том числе критика корректности известных

методов геометрической декомпозиции. Выполнена математическая формализация понятия “задача параметрического проектирования”, более строгая и общая, чем в опубликованных работах, учитывающая характер задач, встречающихся на практике. Также дано определение других понятий, ассоциированных с задачей параметрического проектирования.

Кроме того, был достигнут ряд практических результатов в виде создания новых программных систем, способных решать геометрические задачи параметрического проектирования. Разработка этих систем велась коллективно, и степень личного вклада автора в них определяется следующим образом:

1) В решателе систем нелинейных уравнений LGSEP автор являлся руководителем, определявшим набор и конфигурацию воплощаемых методов, разработчиком архитектурной схемы и, в меньшей степени, исполнителем реализаций конкретных вычислительных методов;
2) В универсальном решателе систем уравнений с недоопределенными данными LEMO автор являлся, под руководством Семенова А.Л. и Важева И.В., разработчиком большого числа вычислительных методов, в том числе интервального метода бисекции и интервального метода Ньютона. Совместно с Петуниным Д.В. был реализован метод CP распространения ограничений;
3) В решателе двухмерных задач параметрического проектирования LGS 2D автор сначала выступал ответственным исполнителем под руководством Ушакова Д.М., а позднее – руководителем и архитектором. Автором была предложена и реализована идея метода отделяющей декомпозиции, после чего набор образцов для метода успешно увеличивался силами Рыкова И.А. Кроме этого, автор участвовал в разработке других методов LGS 2D, в том числе

совместно с Еремченко А.А. реализовал и совершенствовал метод декомпозиции Хоффманна;

4) В геометрическом решателе трехмерных задач параметрического проектирования LGS 3D автор был руководителем и, совместно с Кусковым Р.Е., архитектором. Совместно с Руколеевым Е.В. был разработан и реализован метод остового моделирования, который в дальнейшем был усовершенствован Киселевым А.В.;
5) При разработке библиотеки вычисления элементарных математических функций с направленными округлениями Imath автор являлся единственным программистом-исполнителем.

К числу экспериментально проверяемых результатов, достигнутых автором, следует отнести:

1) разработку метода отделяющей декомпозиции, который многократно ускорил работу геометрических решателей на общей базе из 10 тысяч двухмерных и трехмерных тестов;
2) разработку метода остовного моделирования, который в 2 раза ускорил работу решателя на общей базе из 3 тысяч трехмерных тестов, и в два раза уменьшил число тестов, для которых LGS 3D не может найти решения;
3) разработку интервальной библиотеки вычисления элементарных функций с направленными округлениями на основе разложения в ряды Чебышева и Тейлора, производительность которой практически равна производительности стандартной библиотеки C++, результат вычислений является гарантированно корректным и в 80%-90% случаев неулучшаемым, а количество требуемой оперативной памяти составляет примерно 70 Кбайт.

## Литература


1. Адамар Ж. Элементарная геометрия. – М., ОГИЗ, 1948. – 608 с.
2. Алефельд Г., Херцбергер Ю. Введение в интервальные вычисления. – М., Мир, 1987. – 360 с.
3. Бухбергер Б., Калме Ж., Калтофен Э., Коллинз Дж., Лауэр М., Лафон Ж., Лоос Р., Минньотт М., Нойбюзер И., Норман А., Уинклер Ф., ван Хюльзен Я. Компьютерная алгебра. Символьные и алгебраические вычисления. – М., Мир, 1986 г. – 392 с.
4. Голованов Н.Н. Геометрическое моделирование. – М., издательство физико-математической литературы, 2002. – 472 с.
5. Горбатов В.А., Огиренко А.Г., Смирнов М.И. Искусственный интеллект в САПР: Учебное пособие. – М., МГГУ, 1994. – 183 с.
6. Грувер М., Зиммерс Э. САПР и автоматизация производства. – М., Мир, 1987. – 528 с.
7. Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. Лекции по теории графов. – М., Наука, 1990. – 384 с.
8. Еремченко А., Ершов А. Два новых метода декомпозиции задачи с геометрическими ограничениями // Препринт №11 – Новосибирск, ЗАО ЛЕДАС, 2004. – 36 с.
9. Ершов А. Гарантированно субоптимальные решения задач линейной оптимизации // Препринт №1 – Новосибирск, ЗАО ЛЕДАС, 2002. – 20 с.
10. Ершов А. Использование алгоритма построения базисов Гребнера в рамках концепции программирования в ограничениях // Тез. докл. Четвертого сибирского конгресса по прикладной и индустриальной математике (ИНПРИМ-2000); Институт математики СО РАН. – Новосибирск, 2000. – Часть IV. – С.105-106.
11. Ершов А. Новый метод моделирования задач параметрического проектирования // САПР и Графика. – 2007. – №9. – С.32-35.
12. Клатте Р., Кулиш У., Неага М., Рац Д., Ульрих У. PASCAL-XSC. Язык численного программирования. – М., ДМК Пресс, 2000. – 352 с.
13. Копорушкин П.А. Разработка структур данных и алгоритмов расчета параметрических моделей геометрических объектов : Автореф. дис. к-та тех. наук. – Екатеринбург, 2005. – 22 с.
14. Копорушкин П.А., Партин А.С. Алгоритм расчета параметризованных геометрических объектов // Электронный журнал “ИССЛЕДОВАНО В РОССИИ”. – 2004. – С.184-197.
15. Кошелев В., Молочник В. Что такое PLM? // САПР и Графика. – 2003. – №10. – С.34-37.

16. Кураксин С. А. Повышение эффективности проектирования изделий машиностроения на основе разработки автоматизированных методов и средств формирования параметрических сборочных моделей: Автореф. дис. к-та тех. наук. – М., 1997. – 28 с.
17. Ли К. Основы САПР (CAD/CAM/CAE). – СПб., Питер, 2004. – 559 с.
18. Лурье А.И. Аналитическая механика. – М., Физматгиз, 1961. – 824 с.
19. Малюх В. Архитектура и базовые алгоритмы инструментального ядра графических САПР общего назначения: Дис. к-та физ.-мат. наук. – Новосибирск, 2006. – 93 с.
20. Математическое и программное обеспечение САПР: Сб. науч. тр. / Под ред. В.К.Погребного; ТПУ – Томск, 1997. – 262 с.
21. Математическое методы и модели в САПР: Сб. науч. тр. / Под ред. А.А.Калентьева; Самар. авиац. ин-т. – Самара, 1991. – 145 с.
22. Моделирование интеллектуальных процессов проектирования и производства: Сб. науч. тр. / Под ред. А.Г.Ракович; Ин-т техн. кибернетики АН Беларуси. – Минск, 1996. – 177 с.
23. Нариньяни А.С. Недоопределенные модели и операции с недоопределенными значениями // Препринт №400; АН СССР, Сиб. Отделение, ВЦ – Новосибирск, 1982. – 33 с.
24. Нариньяни А.С. Недоопределенность в системах представления и обработки знаний // Известия АН СССР, серия "Теоретическая кибернетика". – 1986. – №5. – С. 3-28.
25. Прейс С. LGS – эффективный и доступный решатель геометрических задач // САПР и Графика. – 2003. – №9 – С. 48-50.
26. Программное обеспечение САПР: Сб. науч. тр. / Под редакцией И.Е.Педанова; ВЦ РАН. – М., 1997 – 161 с.
27. Ремез У.Я. Основы численных методов чебышевского приближения. – Киев, Наукова думка, 1969. – 623 с.
28. Семенов А. Л., Важев И. В., Кашеварова Т.П., Бревнов Е.В., Ершов А.Г., Клейменов А.Е., Лещенко А.С., Лоенко М.Ю., Петунин Д.В Интервальные методы распространения ограничений и их приложения: Сб. науч. тр. Системная Информатика №9; СО РАН, Институт систем информатики. – Новосибирск, 2004 – С.245-358.
29. Системы автоматизированного проектирования: ретроспективный библиографический указатель (1989-1993) / сост. Т.И. Кукуева. – М., ГПНТБ РОССИИ, 1994. – 96 с.
30. Симонов В.В. Интерактивное приложение для вариационного черчения с архитектурой "клиент-сервер" // Тез. докл. Конференции-конкурса студентов, аспирантов и молодых ученых "Технологии Microsoft в теории и практике образования". – Новосибирск, 2007. – С.193-194.

31. Ушаков Д. М. Введение в математические основы САПР. – Новосибирск, ЗАО ЛЕДАС, 2006. – 180 с.
32. Ушаков Д. М. Объектно-ориентированная среда для недоопределенных вычислений : Дис. к-та физ.-мат. наук. – Новосибирск, 1998. – 160 с.
33. Ушаков Д. М. Технологии вариационного проектирования для разработки типичных приложений САПР – http://plmnews.ru/11571.
34. Шалумов А.С., Багаев Д.В., Осипов А.С. Система автоматизированного проектирования КОМПАСС-График: Учебное пособие. – Ковров, КГТА, 2003. – 42 с.
35. Шестопал Ю.Т., Моисеев В.Б., Дорофеев В.Д. Основы интеллектуальных САПР-технологий. – Пенза, Изд. ПГТУ, 1995. – 244 с.
36. Albajes L.S., Crosa P.B. Geometric Relaxation for Solving Constraint-Based Models // Geometric Constraint Solving and Applications / Editors B. Bruderlin and D. Roller. – Berlin, 1998. – P.259-270.
37. Becker T., Kredel H., Weispfenning V. Grobner bases: a computational approach to commutative algebra. – London, Springer-Verlag, 1993. – 574 p.
38. Bouma W., Fudos I., Hoffmann C. M. A Geometric Constraint Solver // Computer-Aided Design. – 1995. – №27. – P.487-501.
39. Chiang C.-S., Joan-Arinyo R. Revisiting variable radius circles in constructive geometric constraint solving // Computer-Aided Geometric Design. – 2004. – Vol. 21, Issue 4. – P.371-399.
40. Clement A., Riviere A., Serre P. New Technology for Solving Large-Scale Geometrical Networks // Int. Forum isiCAD : Proc / Conf. held at Novosibirsk, Russia, June 2004. – Novosibirsk, 2004. – P.11-20.
41. Dohmen M. A survey of constraint satisfaction techniques for geometric modeling // Computers and Graphics. – 1995. – Vol. 19, №6. – P.831-845.
42. Dulmage A., Mendelsohn N. Coverings of Bipartite Graphs // Canad. J. Math. – 1958. – Vol. 10. – P. 517-534.
43. Durand C., Hoffmann C. M. Variational Constraints in 3D // Int. Conf. on Shape Modeling and Applications: Proc / Conf. held at Aizu-Wakamatsu, Japan, March 1999. – P.90-97.
44. Ershov A. Enchancing Techniques for Geometric Constraint Solving // Preprint №3 – Novosibirsk, LEDAS Ltd., 2003. – 40 p.
45. Ershov A. G., Kashevarova T. P. Interval Mathematical Library Based on Chebyshev and Taylor Series Expansion // Reliable Computing. – 2005. – Vol. 11, №5. – P.359-367.
46. Ershov A., Eremchenko A. Two New Decomposition Techniques in Geometric Constraint Solving // Int. Forum isiCAD: Proc / Conf. held at Novosibirsk, Russia, June 2004. – Novosibirsk, 2004. – P.91-102.

47. Ershov A., Ivanov I., Kornienko V., Preis S., Rasskazov A., Rykov I. A new scheduling engine for PLM // Int. J. of Product Lifecycle Management. – 2006. – Vol. 1, № 2 – P.164-180.
48. Ershov A., Ivanov I., Kazakov A., Lipski S., Markin V., Preis S., Rukoleev E., Sidorov V., Ushakov D. LEDAS Geometric Solver: product overview // Preprint №5 – Novosibirsk, LEDAS Ltd., 2003. – 56 p.
49. Ershov A., Ivanov I., Preis S., Rukoleev E., Ushakov D. LGS: Geometric Constraint Solver // Int. Conf. Perspectives of Systems Informatics, Novosibirsk, 2003. – Berlin, 2003. – P.423-430. – (Lecture Notes in Computer Science, 2890).
50. Essert-Villard C., Schreck P., Dufourd J.-F. Sketch-based pruning of a solution space within a formal geometric constraint solver // Artificial Intelligence. – 2000. – №124. – P.139-159.
51. Gao X.-S., Lin Q., Zhang G.-F. A C-tree decomposition algorithm for 2D and 3D geometric constraint solving // Computer-Aided Design. – 2006. – №38. – P.1-13.
52. Heydon A., Nelson G. The Juno-2 Constraint-Based Drawing Editor : Research report 131a; Systems Research Center, Digital Equipment Corporation. – Palo Alto, 1995. – 19 p.
53. Hoffmann C. M. Constraint-Based Computer-Aided Design // J. of Computing and Information Science in Engineering. – 2005. – Vol. 5, №3. – P.182-187.
54. Hoffmann C. M. Robustness in Geometric Computations // J. of Computing and Information Science in Engineering. – 2001. – Vol. 1, №2. – P.143-155.
55. Hoffmann C. M., Chiang C.-S. Variable-radius circles of cluster merging in geometric constraints: Part I. Translational clusters // Computer-Aided Design. – 2002. – Vol. 34, №11. – P.787-797.
56. Hoffmann C. M., Chiang C.-S. Variable-radius circles of cluster merging in geometric constraints: Part II. Rotational clusters // Computer-Aided Design. – 2002. – Vol. 34, №11. – P.799-805.
57. Hoffmann C. M., Fudos I. Correctness Proof of a Geometric Constraint Solver // Int. J. of Computational Geometry and Application. – 1996. – Vol. 6, №4. – P.405-420.
58. Hoffmann C. M., Joan-Arinyo R. A Brief on Constraint Solving // Computer-Aided Design and Applications. – 2005. – Vol. 2, №. 5 – P.655-663.
59. Hoffmann C. M., Lomonosov A., Sitharam M. Geometric Constraint Decomposition // Geometric Constraint Solving and Applications / Editors B. Bruderlin and D. Roller. – Berlin, 1998. – P.170-195.
60. Hoffmann C. M., Lomonosov A., Sitharam M. Planning geometric constraint decompositions via optimal graph transformations // Int. Workshop on Applications of Graph Transformations with Industrial Relevance: Proc. – London, 1999. – P.309-324 – (Lecture Notes in Computer Science, 1779).

61. Hoffmann C. M., Lomonosov A., Sitharam M. Finding Solvable Subsets of Constraint Graphs // Third Int. Conf. on Principles and Practice of Constraint Programming: Proc / Conf. held at Linz, Austria, 1997. – P.463-477. – (Lecture Notes in Computer Science, 1330).

62. Hoffmann C. M., Sitharam M., Yuana B. Making constraint solvers more usable: overconstraint problem // Computer-Aided Design. – 2004. – Vol. 36, №4. – P.377-399.

63. Jafar J., Maher M. J. Constraint Logic Programming: A survey // J. of Logic Programming. – 1994. – Vol. 19/20 – P.503-581.

64. Jermann C., Neveu B., Trombettoni G. A New Structural Rigidity for Geometric Constraints System // Int. Workshop on Automated Deduction in Geometry ADG'2002 : Proc / Conf. held at Hagenberg Castle, Austria, September 2002. – Berlin, 2004. – P.87-105. – (Lecture Notes in Computer Science, 2930).

65. Jermann C., Trombettoni G., Neveu B., Rueher M. A Constraint Programming Approach for Solving Rigid Geometric Systems. // Sixth Int. Conference on Principles and Practice of Constraint Programming: Proc / Conf. held at Singapore, September 2000. – Berlin, 2000. – P.233-248. – (Lecture Notes in Computer Science, 1894).

66. Jermann C., Trombettoni G., Neveu B., Mathis P. Decomposition of Geometric Constraint Systems: a Survey // Int. J. of Computational Geometry and Application. – 2006. – Vol. 16, №5-6. – P.379-414.

67. Joan-Arinyo R., Luzon M.V., Soto A. Genetic algorithms for root multiselection in constructive geometric constraint solving // Computers and Graphics. – 2003. – Vol. 27, №1. – P.51-60.

68. Kim J., Kim K., Choi K., Lee J. Solving 3D Geometric Constraints for Assembly Modelling // Int. J. of Advanced Manufacturing Technology. – 2000. – Vol. 16. – P.843-849.

69. Kim J., Kim K., Lee J., Jeong J. Generation of assembly models from kinematic constraints // Int. J. of Advanced Manufacturing Technology. – 2005. – Vol. 26. – P.131-137.

70. Kim J., Kim K., Lee J., Jung H. Solving 3D geometric constraints for closed-loop assemblies // Int. J. of Advanced Manufacturing Technology. – 2004. – Vol. 23. – P.755-761.

71. Klein R. The Role of Constraints in Geometric Modelling // Geometric Constraint Solving and Applications / Editors B. Bruderlin and D. Roller. – Berlin, 1998. – P.3-23.

72. Kramer G. A. A geometric constraint engine // Artificial Intelligence. – 1992. – Vol. 58 – P.327-360.

73. Lamure H., Michelucci D. Qualitative Study of Geometric Constraints // Geometric Constraint Solving and Applications / Editors B. Bruderlin and D. Roller. – Berlin, 1998. – P.234-258.

74. Lamure H., Michelucci D. Solving geometric constraint by homotopy // Third ACM Symposium on Solid modeling and applications: Proc. / Conf. held at Salt Lake City, USA. – New York, USA, 1995. – p. 263-269.

75. Lee K.-Y., Kwon O-H., Lee J.-Y., Kim T.-W. A hybrid approach to geometric constraint solving with graph analysis and reduction // Advances in Engineering Software. – 2003 – Vol. 34. – P.103-113.

76. Lhomme O., Kuzo P., Mace P. Desargues: A Constraint-based system for 3D Projective Geometry // Geometric Constraint Solving and Applications / Editors B. Bruderlin and D. Roller. – Berlin, 1998. – P.196-210.

77. Marriott K, Stuckey J. Programming with constraints: an introduction. – Cambridge, MIT, USA. – 467 p.

78. Martinez M. L., Felez J. A constraint solver to define correctly dimensioned and overdimensioned parts // Computer-Aided Design. – 2005. – Vol. 37. – P.1353-1369.

79. Mathis P., Schreck P., Dufourd J.-F. YAMS: A Multi-Agent System for 2D Constraint Solving // Geometric Constraint Solving and Applications / Editors B. Bruderlin and D. Roller. – Berlin, 1998. – P.211-233.

80. Michelucci D. The Next 100 Papers About Geometric Constraint Solving // Int. Forum isiCAD: Proc / Conf. held at Novosibirsk, Russia, June 2004. – Novosibirsk, 2004. – P.1-11.

81. Michelucci D., Foufou S. Using Cayley-Menger determinants for geometric constraint solving // Ninth ACM Symposium on Solid and Physical Modeling : Proc. / Conf. held at Genoa, Italy. – Aire-la-Ville, Switzerland, 2004. – P.285-290.

82. Nahm Y., Ishikawa H. A new 3D-CAD system for set-based parametric design // Int. J. of Advanced Manufacturing Technology. – 2006. – Vol. 29. – P.137-150.

83. Owen J. C. Algebraic Solution for Geometry from Geometrical Constraints // First ACM Symposium on Solid modeling foundations and CAD/CAM applications: Proc. / Conf. held at Austin, USA. – New York, USA, 1991. – P.397-407.

84. Owen J. C. Constraint on simple geometry in two and three dimensions // Int. J. of Computational Geometry and Applications. – 1996. – Vol.6, №4. – P.421-434.

85. Serre P., Ortuzar A., Riviere A. Non-cartesian modelling for analysis of the consistency of a geometric specification for conceptual design // Int. J. of Computational Geometry and Applications. – 2006. – Vol.16, №5-6. – P.549-565.

86. Shi Z., Chen L. An angular constraints solving approach for assembly modeling based on spherical geometry // Int. J. of Advanced Manufacturing Technology. – 2007. – Vol. 32. – P.366-377.
87. Shimizu S., Numao M. Constraint-based design for 3D shapes // Artificial Intelligence. – 1997. – Vol. 91 – P.51-69.
88. Shpitalni M., LipsonH. Automatic Reasoning for Design Under Geometric Constraints // Annals of the CIRP. – 1997. – Vol. 46, №1 – P.85-89.
89. Sitharam M., Oung J.J., Zhou Y., Arbree A. Geometric constraints with feature hierarhies // Computer-Aided Design. – 2006. – Vol. 38 – P.22-38.
90. Sitharam M., Arbree A., Zhou Y., Kohareswaran N. Solution space navigation for geometric constraint systems // ACM Transactions on Graphics – 2006. – Vol. 25, №2. – P.194-213.
91. Telerman V., Preis S., Snytnikov N., Ushakov D. Interval/set based collaborative engineering design // Int. J. of Product Lifecycle Management. – 2006. – Vol. 1, №2. – P.150-163.
92. Ushakov D. Adding intelligence to software solutions for PLM: constraint-based approach // Int. J. of Product Lifecycle Management. – 2006. – Vol. 1, №2 – P.181-193.
93. Van der Meiden H. A., Bronsvoort W. F. A constructive approach to calculate parameter ranges for systems of geometric constraints // Computer-Aided Design. – 2006. – Vol. 38. – P.275-283.
94. Veltkamp R. C., Arbabz F. Interactive Geometric Constraint Satisfaction : Technical report; Institute of Information and Computing Sciences, Utrecht University. – Utrecht, 1996. – 24 p. .
95. Wang Y., Chen L., Huang Z., Wu J., Zhong Y. A History-Independent Modelling-Oriented Approach to Solve Geometric Constraints Between Features in 3D Space // Int. J. of Advanced Manufacturing Technology. – 2005. – Vol. 25. – P.334-342.

## Приложения

# 1. Количество степеней свободы, снимаемых разными ограничениями

В идущих далее списках под ограничениями угла (а также перпендикулярности и параллельности) между объектами в трехмерном пространстве подразумеваются ограничения между их главными векторами, которыми являются: направляющий вектор для прямой, ось для цилиндра и окружности, нормаль для плоскости. Поэтому, например, параллельность прямой и плоскости соответствует перпендикулярности прямой и плоскости в обычном геометрическом смысле.

Рассматриваются только унарные и бинарные ограничения в двух- и трехмерном пространстве. Списки не претендуют на полноту перечисления ограничений указанного класса. Ограничения обозначаются записью вида <ТипОбъекта1>[<ТипОбъекта2>]<ТипОтношения>, в которой последовательно заданы типы всех сущностей.

Одну степень свободы в трехмерном пространстве снимают следующие ограничения:

- `ТочкаТочкаРасстояние;`
- `ТочкаПрямаяРасстояние;`
- `ПрямаяПрямаяРасстояние;`
- `ПрямаяПрямаяУгол;`
- `ТочкаПлоскостьРасстояние;`
- `ПрямаяПлоскостьУгол;`
- `ПрямаяПлоскостьПерпендикулярность;`
- `ПлоскостьПлоскостьУгол;`
- `ТочкаКриваяРасстояние;`
- `ПрямаяКриваяРасстояние;`
- `ТочкаПоверхностьИнцидентность;`
- `ТочкаПоверхностьРасстояние;`
- `ПрямаяПоверхностьРасстояние;`

- ПрямаяПоверхностьКасание;
- ПлоскостьПоверхностьРасстояние;
- ПлоскостьПоверхностьКасание;
- ПоверхностьПоверхностьРасстояние;
- ПоверхностьПоверхностьКасание;
- ТочкаОкружностьРасстояние;
- ПрямаяОкружностьРасстояние;
- ПрямаяОкружностьУгол;
- ПлоскостьОкружностьРасстояние;
- ПлоскостьОкружностьУгол;
- ОкружностьОкружностьРасстояние;
- ОкружностьОкружностьКасание;
- ОкружностьОкружностьУгол;
- ОкружностьРадиус;
- ТочкаЦилиндрРасстояние;
- ТочкаЦилиндрИнцидентность;
- ПрямаяЦилиндрРасстояние;
- ПрямаяЦилиндрКасание;
- ПрямаяЦилиндрУгол;
- ПлоскостьЦилиндрУгол;
- ПлоскостьЦилиндрПерпендикулярность;
- ОкружностьЦилиндрРасстояние;
- ОкружностьЦилиндрКасание;
- ОкружностьЦилиндрУгол;
- ОкружностьЦилиндрПерпендикулярность;
- ЦилиндрЦилиндрРасстояние;
- ЦилиндрЦилиндрУгол;
- ЦилиндрРадиус;
- ТочкаСфераРасстояние;
- ТочкаСфераИнцидентность;
- ПрямаяСфераКасание;
- ПлоскостьСфераРасстояние;
- СфераСфераРасстояние;
- СфераРадиус.

Одну степень свободы в двухмерном пространстве снимают следующие ограничения:

- ТочкаТочкаРасстояние;
- ТочкаПрямаяРасстояние;
- ТочкаПрямаяИнцидентность;
- ПрямаяПрямаяУгол;
- ПрямаяПрямаяПараллельность;
- ПрямаяПрямаяПерпендикулярность;
- ТочкаОкружностьРасстояние;
- ТочкаОкружностьИнцидентность;
- ПрямаяОкружностьРасстояние;
- ПрямаяОкружностьКасание;
- ОкружностьОкружностьРасстояние;
- ОкружностьОкружностьКасание;
- ОкружностьРадиус;
- ТочкаЭллипсРасстояние;
- ТочкаЭллипсИнцидентность;
- ПрямаяЭллипсРасстояние;
- ПрямаяЭллипсКасание;
- ОкружностьЭллипсРасстояние;
- ОкружностьЭллипсКасание;
- ЭллипсЭллипсРасстояние;
- ЭллипсЭллипсКасание;
- ЭллипсБольшойРадиус;
- ЭллипсМалыйРадиус;
- ТочкаКриваяРасстояние;
- ТочкаКриваяИнцидентность;
- ПрямаяКриваяРасстояние;
- ПрямаяКриваяКасание;
- ОкружностьКриваяРасстояние;
- ОкружностьКриваяКасание;
- ЭллипсКриваяРасстояние;
- ЭллипсКриваяКасание;
- КриваяКриваяРасстояние;

- КриваяКриваяКасание.

Две степени свободы в трехмерном пространстве снимают следующие ограничения:

- ТочкаПрямаяИнцидентность;
- ПрямаяПрямаяПараллельность;
- ПрямаяПлоскостьРасстояние;
- ПрямаяПлоскостьПараллельность;
- ПлоскостьПлоскостьПараллельность;
- ТочкаКриваяИнцидентность;
- КриваяКриваяРасстояние;
- КриваяПоверхностьРасстояние;
- КриваяПоверхностьКасание;
- ОкружностьПоверхностьКасание;
- ЭллипсПоверхностьКасание;
- ТочкаОкружностьИнцидентность;
- ПрямаяОкружностьПараллельность;
- ПлоскостьОкружностьПараллельность;
- ОкружностьОкружностьПараллельность;
- ПрямаяЦилиндрПараллельность;
- ПлоскостьЦилиндрРасстояние;
- ПлоскостьЦилиндрПараллельность;
- ОкружностьЦилиндрПараллельность;
- ЦилиндрЦилиндрПараллельность.

Две степени свободы в двухмерном пространстве снимают следующие ограничения:

- ТочкаТочкаИнцидентность;
- ПрямаяПрямаяРасстояние;
- ПрямаяПрямаяИнцидентность;
- ТочкаОкружностьКонцентричность;
- ОкружностьОкружностьКонцентричность;
- ТочкаЭллипсКонцентричность;
- ОкружностьЭллипсКонцентричность;
- ЭллипсЭллипсКонцентричность.

Три степени свободы в трехмерном пространстве снимают следующие ограничения:

- ТочкаТочкаИнцидентность;
- ПлоскостьПлоскостьИнцидентность;
- ПлоскостьПлоскостьРасстояние;
- ПрямаяКриваяКасание;
- КриваяКриваяКасание;
- ТочкаОкружностьКонцентричность;
- ОкружностьОкружностьКонцентричность;
- ПлоскостьОкружностьИнцидентность;
- ПрямаяЦилиндрИнцидентность;
- ТочкаСфераКонцентричность;
- СфераСфераКонцентричность.

Три степени свободы в двухмерном пространстве снимает ограничение ОкружностьОкружностьИнцидентность.

Четыре степени свободы в трехмерном пространстве снимают следующие ограничения:

- ПрямаяПрямаяИнцидентность;
- ПрямаяОкружностьСоосность;
- ОкружностьОкружностьСоосность;
- ПрямаяЦилиндрСоосность;
- ОкружностьЦилиндрСоосность;
- ЦилиндрЦилиндрСоосность;
- СфераСфераИнцидентность.

Пять степеней свободы в трехмерном пространстве снимают следующие ограничения:

- ОкружностьЦилиндрИнцидентность;
- ЦилиндрЦилиндрИнцидентность.

Пять степеней свободы в двухмерном пространстве снимает ограничение ЭллипсЭллипсИнцидентность.

Шесть степеней свободы в трехмерном пространстве снимает ограничение ОкружностьОкружностьИнцидентность.

## 2. Список образцов отделяющей декомпозиции

Описания V-образцов, идущие далее, имеют следующий формат.

**<ТипОтношения> := Инцидентность | Расстояние | Угол | Параллельность | Перпендикулярность | Касание | Концентричность | Радиус**

**<ТипОбъекта> := Точка | Прямая | Плоскость | Окружность**

**<ТипОграничения> := [<ТипОбъекта>]<ТипОбъекта><ТипОтношения>**

**<Список> := {<ТипОграничения>[, <ТипОграничения>]..}**

**<НаборОграничений> := (<Список>[, <Список>]..)**

**<СписокОбъектов> := <ТипОбъекта>[, <ТипОбъекта>]**

**<V-Образец> := <СписокОбъектов> <НаборОграничений>**

В записи каждого **<ТипОграничения>** один из **<ТипОбъекта>** выделен жирным, это означает, что именно этот объект является отделяемым. Смысл записи **<V-Образец>** состоит в том, что указывается **<СписокОбъектов>** в отделяемом теле, а потом с помощью **<НаборОграничений>** задается набор возможных ограничений на эти объекты, причем на каждом месте в **<НаборОграничений>** может стоять любой из **<ТипОграничения>,** записанного в **<Список>.**

Далее идущие списки V-образцов не являются исчерпывающими, они ограничены только простейшими типами объектов (точка, прямая, плоскость, окружность в двухмерном пространстве). Более того, здесь не приводятся условно корректные и нуль-эвристические образцы.

Следующие V-образцы являются корректными образцами отделяющей декомпозиции в двухмерных задачах:

- Прямая ({Точка**Прямая**Инцидентность}, {Точка**Прямая**Инцидентность});
- Прямая ({**Прямая**ПрямаяУгол, **Прямая**ПрямаяПараллельность}, {Точка**Прямая**Инцидентность, Точка**Прямая**Расстояние, **Прямая**ОкружностьКасание, **Прямая**ОкружностьРасстояние});
- Окружность ({Окружность**Окружность**Концентричность, Точка**Окружность**Концентричность, Точка**Окружность**Инцидентность, Прямая**Окружность**Касание, Окружность**Окружность**Касание},

{Точка**Окружность**Инцидентность, Прямая**Окружность**Касание, Окружность**Окружность**Касание});

- Окружность ({Окружность**Окружность**Концентричность, Точка**Окружность**Концентричность, Точка**Окружность**Инцидентность, Прямая**Окружность**Расстояние, Окружность**Окружность**Расстояние}, {Прямая**Окружность**Касание, Прямая**Окружность**Расстояние});
- Окружность ({**Окружность**Радиус}, {Окружность**Окружность**Концентричность, Точка**Окружность**Концентричность, Точка**Окружность**Инцидентность, Точка**Окружность**Расстояние, Прямая**Окружность**Расстояние, Прямая**Окружность**Касание, Окружность**Окружность**Расстояние, Окружность**Окружность**Касание});
- Прямая, Окружность ({**Прямая**ПрямаяУгол, **Прямая**ПрямаяПараллельность}, {Окружность**Окружность**Концентричность, Окружность**Окружность**Расстояние, Окружность**Окружность**Касание, Окружность**Окружность**Инцидентность, Точка**Окружность**Концентричность, Точка**Окружность**Инцидентность, Точка**Окружность**Расстояние, Прямая**Окружность**Расстояние, Прямая**Окружность**Касание});
- Прямая, Точка ({**Прямая**ПрямаяУгол, **Прямая**ПрямаяПараллельность}, {**Точка**ОкружностьКонцентричность, **Точка**ОкружностьИнцидентность, **Точка**ОкружностьРасстояние, **Точка**ПрямаяИнцидентность, **Точка**ПрямаяРасстояние, **Точка**ТочкаИнцидентность, **Точка**ТочкаРасстояние});

Следующие V-образцы являются корректными образцами отделяющей декомпозиции в трехмерных задачах:

- Прямая ({Точка**Прямая**Инцидентность}, {Точка**Прямая**Инцидентность});
- Прямая ({**Прямая**ПрямаяУгол, **Прямая**ПрямаяПараллельность, **Прямая**ПлоскостьУгол, **Прямая**ПлоскостьПараллельность, **Прямая**ПлоскостьПерпендикулярность}, {Точка**Прямая**Инцидентность, Точка**Прямая**Расстояние, **ПрямаяП**рямаяРасстояние});
- Плоскость ({Точка**Плоскость**Инцидентность}, {Точка**Плоскость**Инцидентность}, {Точка**Плоскость**Инцидентность});
- Плоскость ({Прямая**Плоскость**Инцидентность}, {Точка**Плоскость**Инцидентность});
- Плоскость ({Прямая**Плоскость**Угол, Прямая**Плоскость**Параллельность, Прямая**Плоскость**Перпендикулярность, **Плоскость**ПлоскостьПараллельность, **Плоскость**ПлоскостьПерпендикулярность}, {Точка**Плоскость**Инцидентность, Точка**Плоскость**Расстояние });
- Прямая, Точка ({**Прямая**ПрямаяУгол, **Прямая**ПрямаяПараллельность, **Прямая**ПлоскостьУгол, **Прямая**ПлоскостьПараллельность, **Прямая**ПлоскостьПерпендикулярность}, {**Точка**ПрямаяИнцидентность, **Точка**ПрямаяРасстояние, **Точка**ПлоскостьИнцидентность, **Точка**ПлоскостьРасстояние, **Точка**ТочкаИнцидентность, **Точка**ТочкаРасстояние});
- Плоскость, Точка ({Прямая**Плоскость**Угол, Прямая**Плоскость**Параллельность, Прямая**Плоскость**Перпендикулярность, **Плоскость**ПлоскостьПараллельность, **Плоскость**ПлоскостьПерпендикулярность}, {**Точка**ПрямаяИнцидентность,

**Точка**ПрямаяРасстояние, **Точка**ПлоскостьИнцидентность,
**Точка**ПлоскостьРасстояние, **Точка**ТочкаИнцидентность,
**Точка**ТочкаРасстояние});

## 3. Пример использования интерфейса LGS 2D

```
#include "LGSPublicAPI.h"

void main()
{ // This is a simple example - a snowman model.
 LGSContext ctx = LGSCreateContext(LGS_DEFAULT_LINEAR_TOLERANCE,
                  LGS_DEFAULT_ANGULAR_TOLERANCE);

 LGSReal C1_center_coord[ 2 ] = {0.5, 0};
 LGSCircle C1 = LGSCreateCircle (ctx, C1_center_coord, 10);
 LGSPoint C1_center = LGSCreatePoint (ctx, C1_center_coord);
 LGSCreateConcentric (ctx, C1, C1_center);
 LGSReal C2_center_coord[ 2 ] = {1.5, 11};
 LGSCircle C2 = LGSCreateCircle (ctx, C2_center_coord, 5);
 LGSPoint C2_center = LGSCreatePoint (ctx, C2_center_coord);
 LGSCreateConcentric (ctx, C2, C2_center);
 LGSReal C3_center_coord[ 2 ] = {0, 17};
 LGSCircle C3 = LGSCreateCircle (ctx, C3_center_coord, 2);
 LGSPoint C3_center = LGSCreatePoint (ctx, C3_center_coord);
 LGSCreateConcentric (ctx, C3, C3_center);

// We want centers of all circles to be on the vertical line.
 LGSReal L1_point[ 2 ] = {0, 0};
 LGSReal L1_direction[ 2 ] = {    0.5, 1 };
 LGSLine L1 = LGSCreateLine( ctx, L1_point, L1_direction);
 LGSCreateIncidence( ctx, L1, C1_center);
 LGSCreateIncidence( ctx, L1, C2_center);
 LGSCreateIncidence( ctx, L1, C3_center);
 LGSCreateVerticality(ctx, L1);

 LGSCreateTangency(ctx, C1, C2);
 LGSCreateTangency(ctx, C2, C3);
 LGSCreateFixation(ctx, C2_center);

// Now we can start calculations.
 LGSResult res = LGSApplyChanges(ctx);
```

```
 LGSObject res = LGSGetContextFirstObject (ctx, LGS_TYPE_ANY,
                 LGS_STATE_ANY);
 while (res != LGS_NULL) {
  cout << "Object data" << endl;
  LGSUInt n = LGSGetObjectParamsCount(ctx, res);
  for (LGSUInt i = 0; i < n; i++) {
   LGSReal value = 0.0;
   LGSGetObjectParamValue(ctx, res, i, &value);
   cout << "I-th parameter of object is equal to " << value;
  }
  res = LGSGetContextNextObject (ctx, LGS_TYPE_ANY,
        LGS_STATE_ANY, res);
 }

// Finally we must free all resources.
 LGSRemoveContext( ctx);
 cout << "Context removed" << endl;
}
```