МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

МИРЭА – РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

**А.В. ПИМЕНОВ, А.М. ВОЛОДИНА**

**ПЛАНЕТАРНЫЕ И ВОЛНОВЫЕ РЕДУКТОРЫ**

**УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ**

Москва – 2020

УДК 621.01 ББК 34.445

**Пименов А.В. Планетарные и волновые редукторы** [Электронный ресурс]: Учебно-методическое пособие / Пименов А.В., Володина А.М. – М.: МИРЭА – Российский технологический университет, 2020. – 10 электрон. опт. диск (CD-ROM)

Разработано в помощь студентам, изучающим курс по дисциплине «Технологии проектирования промышленных изделий»

Предназначено для освоения учебной программы и отработки практического применения полученных знаний

Учебно-методические указания издаются в авторской редакции.

Авторский коллектив: Пименов Алексей Владимирович, Володина Анна Михайловна

Авторы выражают благодарность студентам кафедры промышленной информатики ИИТ Чалкову А.А., Пищикову Н.А., Карпу Г.В. за активное участие в разработке программного обеспечения автоматизированных расчетов планетарных механизмов, апробации созданной информационной системы и помощь в оформлении материалов пособия.

Рецензенты:

Парфенов А.В., к.т.н., исполнительный директор АО «НИИВК им. М.А. Карцева» Болбаков Р.Г. к.т.н., доцент заведующий кафедрой ИППО

Минимальные системные требования:

Наличие операционной системы Windows, поддерживаемой производителем. Наличие свободного места в оперативной памяти не менее 128 Мб.

Наличие свободного места в памяти хранения (на жестком диске) не менее 30 Мб. Наличие интерфейса ввода информации.

Дополнительные программные средства: программа для чтения pdf-файлов (Adobe Reader). Подписано к использованию по решению Редакционно-издательского совета

МИРЭА – Российского технологического университета от \_\_\_ \_\_\_\_\_\_\_\_\_\_ 2020 г. Объем \_\_\_ Мб

Тираж 10

© Пименов А.В., Володина А.М. 2020 © МИРЭА – Российский технологический университет, 2020

СОДЕРЖАНИЕ

1 ОБЩИЕ СВЕДЕНИЯ О ПЛАНЕТАРНЫХ И ВОЛНОВЫХ ЗУБЧАТЫХ МЕХАНИЗМАХ...................................................................................................... 3

2 СООТНОШЕНИЯ ДЛЯ КИНЕМАТИЧЕСКОГО И ГАБАРИТНОГО РАСЧЕТОВ ПЛАНЕТАРНЫХ ЗУБЧАТЫХ МЕХАНИЗМОВ.......................... 8

3 АЛГОРИТМЫ ПРОЕКТИРОВОЧНОГО РАСЧЕТА НА ЭВМ................... 14

4 ПРОГРАММЫ ДЛЯ ЭВМ ПРОЕКТИРОВОЧНОГО РАСЧЕТА ПЛАНЕТАРНЫХ МЕХАНИЗМОВ .................................................................... 18

5 РАСТЧЕТЫ ПЛАНЕТАРНЫХ МЕХАНИЗМОВ НА ЭВМ .......................... 28

6 СООТНОШЕНИЯ И АЛГОРИТМЫ КИНЕМАТИКО-ГАБАРИТНОГО РАСЧЕТА ВОЛНОВЫХ ЗУБЧАТЫХ МЕХАНИЗМОВ.................................. 31

7 ПРОГРАММА И РАСЧЕТ НА ЭВМ ВОЛНОВЫХ ЗУБЧАТЫХ МЕХАНИЗМОВ .................................................................................................... 38

8 ОПИСАНИЕ ИНТЕРФЕЙСА И СТРУКТУРА ИНФОРМАЦИОННОЙ СИСТЕМЫ............................................................................................................. 49

9 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.............................................................. 70 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ ........................................... 74

**КИНЕМАТИЧЕСКИЕ И ГАБАРИТНЫЕ РАСЧЕТЫ ПЛАНЕТАРНЫХ И ВОЛНОВЫХ ЗУБЧАТЫХ МЕХАНИЗМОВ**

**1 ОБЩИЕ СВЕДЕНИЯ О ПЛАНЕТАРНЫХ И ВОЛНОВЫХ ЗУБЧАТЫХ МЕХАНИЗМАХ**

В настоящее время с развитием электротехники и металлургии редкоземельных металлов и сплавов для нужд приборостроения в широком доступе появились высокооборотные электродвигатели, рабочие частоты вращения которых достигают значений десятков тысяч оборотов в минуту. Для согласования высокооборотного двигателя и рабочего механизма между валом двигателя и входным валом механизма включается такой передаточный механизм, который должен обеспечивать большое передаточное отношение при приемлемом значении коэффициента полезного действия и надлежащей плавности хода.

В электромеханическом приводе приборных устройств, автоматики и робототехники широко применяют разнообразные планетарные и производные от них волновые зубчатые механизмы, отвечающие названным выше требованиям.

Планетарные механизмы, используемые в качестве редукторов или мультипликаторов, реализуют широкий диапазон передаточных отношений. Одно из основных достоинств этих механизмов — компактность конструкции. При небольших передаточных отношениях редукторов (до 10) достигается также весьма высокий КПД. Значительные передаточные отношения в устройствах точной механики получают обычно за счет последовательного соединения простейших планетарных передач, что обеспечивает хорошую технологичность конструкции и достаточный КПД.

На рис. 1 и 2 приведены некоторые наиболее распространенные в приборостроении схемы планетарных механизмов. Их звенья: центральные колеса *а* и 𝑏, оси которых совпадают с основной осью OO механизма; сателлиты 𝑔, 𝑓 имеющие подвижные оси; водило ℎ, служащее для установки опор сателлитов. Так как основными звеньями здесь являются два центральных колеса и водило, то эти планетарные механизмы называют механизмами типа

2𝑘 — ℎ. Механизм, выполненный по схеме рис. 1, обозначим 𝐴𝑎ℎ , а механизм, выполненный по схеме рис. 2, - *В*𝑎ℎ. В приборных конструкциях количество сателлитов 𝑛𝑤 чаще всего 3 или 2, используют также планетарные механизмы с одним сателлитом. Применение нескольких сателлитов позволяет получить

3

многопоточную передачу мощности, когда каждое зацепление передает только часть общей нагрузки. Применение внутреннего зацепления колес обеспечивает многопарность контакта зубьев и высокую нагрузочную способность.

Рисунок 1 — Кинематическая схема планетарного редуктора 𝐴𝑎ℎ

Рисунок 2 — Кинематическая схема планетарного редуктора со сдвоенными сателлитами 𝐵𝑎ℎ

4

Наибольшее распространение в приводах получил механизм типа 𝐴𝑎ℎ (рис. 1) благодаря сравнительной простоте изготовления, малым габаритам и массе,

высокому КПД (до 0,99); его максимальное передаточное отношение как

редуктора относительно небольшое (𝑖*пл* ≤ 9). Механизм типа 𝐵𝑎ℎ (рис. 2) дает большее передаточное отношение (𝑖*пл* ≤ 16) при несколько меньшем КПД, но конструктивно он сложнее. В приборостроении нашли применение и другие

схемы планетарных механизмов типа 2𝑘 — ℎ: с двумя внутренними и двумя внешними зацеплениями. В первом случае центральные колеса 𝑎, 𝑏 механизма

𝐵𝑎ℎ выполняются с внутренними зубьями, во втором — оба с внешними. Такие схемы позволяют реализовать еще большее передаточное отношение редуктора, но КПД этих механизмов небольшой. Расчет их аналогичен

механизмам типа 𝐴𝑎ℎ и 𝐵𝑎ℎ.

Волновые зубчатые механизмы — структурная разновидность

планетарных типа 2𝑘 — ℎ и используются в качестве редукторов. Они в наибольшей мере удовлетворяют требованиям снижения габаритов и массы передаточного механизма и обеспечения высокого КПД. Рациональный диапазон передаточных отношений, реализуемых одноступенчатым волновым редуктором, составляет 𝑖*в* = 70 .. . 300 при КПД до 0,9. На рис. 3 и 4 показаны кинематические схемы волнового механизма, состоящего из генератора волн деформации ℎ, закрепленного на входном валу, жесткого зубчатого колеса 𝐶 с внутренними зубьями и гибкого зубчатого колеса 𝐹, представляющего собой тонкостенный стакан с нарезанными внешними зубьями. Одно из колес — неподвижно (на рис. 3 неподвижно колесо 𝐹), второе связано с выходным валом. В соответствии с принятой классификацией волновой механизм обозначается 𝐶𝐹 − ℎ.

5

Рисунок 3 — Кинематическая схема волнового редуктора с неподвижным гибким колесом

Рисунок 4 — Кинематическая схема волнового редуктора с неподвижным жестким колесом

6

Генератор ℎ при вращении деформирует гибкое колесо 𝐹 и последовательно вводит его зубья в зацепление с колесом *С*. В зонах зацепления образуются волны деформации, перемещающиеся по гибкому колесу. Многопарность зацепления волнового механизма обеспечивает его высокую нагрузочную способность и хорошую кинематическую точность.

Возможность размещения электродвигателя (ЭД), датчиков (например, датчика скорости ДС), подшипниковых узлов (ПУ) и других устройств во внутренней полости гибкого колеса 𝐹 (см. рис. 3, 4) приводит к весьма компактной конструкции. В то же время схема с неподвижным гибким колесом (рис. 3) может обеспечить герметичность, что бывает необходимо для нормальной работы привода в условиях космоса, агрессивных сред и т. п.

Конструкция генератора волн во многом определяет работоспособность механизма. В точной механике применяют генераторы как принудительной, так и свободной деформации. Генератор принудительной деформации кулачкового типа со стандартным гибким шарикоподшипником (ГП, рис. 5) обеспечивает наиболее надежную конструкцию, высокий КПД, но создает большую инерционность привода. Так как гибкие подшипники генераторов могут надежно работать лишь при ограниченных частотах вращения, то между валом высокоскоростного электродвигателя и генератором может устанавливаться дополнительный редуктор, понижающий частоту вращения. Этим же уменьшается инерционность привода.

Рисунок 5 — Гибкий шарикоподшипник и его геометрические параметры 7

Конструкция гибкого зубчатого колеса определяет долговечность и надежность волнового механизма. В приборостроении применяют разнообразные конструкции гибких колес, описанные в литературе [1, 2, 3]. На рис. 6 и 7 показаны типовые конструкции зубчатых венцов жесткого и гибкого колес.

Рисунок 6 — Вариант конструкции гибкого и жесткого колес

Рисунок 7 — Геометрические параметры гибкого колеса

**2 СООТНОШЕНИЯ ДЛЯ КИНЕМАТИЧЕСКОГО И ГАБАРИТНОГО РАСЧЕТОВ ПЛАНЕТАРНЫХ**

**ЗУБЧАТЫХ МЕХАНИЗМОВ**

Проектировочный расчет кинематических и габаритных параметров планетарного зубчатого механизма состоит в определении чисел зубьев и размеров центральных колес и сателлитов, габаритных размеров. Обычно при этом расчете ставятся задачи вписывания механизма в требуемые габариты и реализации передаточного отношения в пределах допускаемой погрешности.

8

Основные кинематические и геометрические соотношения для расчета планетарных механизмов приводятся в литературе [1, 2, 3]. Здесь эти соотношения даны применительно к рассматриваемым схемам механизмов

типа 𝐴𝑎ℎ и*В*𝑎ℎ (см. рис. 1 и 2).

Входными и выходными звеньями здесь могут быть водило ℎ и

центральные колеса 𝑎, 𝑏 (если колесо 𝑏 сделано подвижным, то *а* жестко связано со стойкой). Но конструкция с входным звеном 𝑏 нерациональна в отношении габаритов. Поэтому далее рассматриваются только механизмы с неподвижным центральным колесом 𝑏 при входном звене *а* — это редуктор, а при входном водиле ℎ — мультипликатор.

Передаточное отношение планетарных механизмов 2𝑘 — ℎ∗

𝑖*пл* = 𝑖𝑎ℎ = 1 − 𝑖𝑎𝑏, (2.1)

где 𝑖𝑎ℎ — передаточное отношение ступенчатого зубчатого ряда, получаемого из планетарного механизма при остановленномводиле. Знак 𝑖𝑎𝑏 определяется знаками передаточных отношений, входящих в этот ряд (для внешнего

зацепления «—», для внутреннего —«+»). Для схем 𝐴𝑎ℎ и 𝐵𝑎ℎ соответственно: 𝑖𝑎𝑏 = (−𝑖𝑎𝑔) ∗ 𝑖𝑔𝑏 *и* 𝑖𝑎𝑏 = (−𝑖𝑎𝑔) ∗ 𝑖𝑓𝑏. Замена в формуле (2.1) 𝑖𝑎𝑔, 𝑖𝑔𝑏 и 𝑖𝑓𝑏 передаточными числами 𝑈𝑎𝑔, 𝑔𝑏 и 𝑓𝑏 этих зубчатых пар дает согласно оптимальному диапазону изменения передаточных чисел для цилиндрической

прямозубой передачи (1.1):

1 = 𝑢 𝑚𝑖𝑛 ≤ 𝑢 = 𝑧2𝑗 ≤ 𝑢 𝑚𝑎𝑥 = 6 1𝑗

для механизма 𝐴𝑎ℎ

𝐼*плА* = 1 + 𝑧𝑏𝑎 , (2.2)

а для механизма 𝐵𝑎ℎ

𝑧

𝐼*пл*𝐵 = 1 + 𝑧𝑏 ∗ 𝑧𝑎 ∗ 𝑧𝑓 ,

(2.3)

где 𝑧𝑎 и 𝑧𝑏 — числа зубьев центральных колес *а* и 𝑏; 𝑧𝑔 и 𝑧𝑓 — числа зубьев сателлитов 𝑔 и 𝑓.

9

Рациональный в отношении габаритов и КПД диапазон изменения передаточных отношений планетарных редукторов 𝐴𝑎ℎ и 𝐵𝑎ℎ соответственно:

4 ≤ 𝑖*плА* ≤ 9,7 ≤ 𝑖*пл В* ≤ 16. (2.4)

При этом диапазоны изменения передаточных чисел пар внешнего и внутреннего зацеплений, входящих в планетарный механизм:

1 ≤ 𝑈𝑎𝑔 ≤ 6 *,* 1,4 ≤ (𝑈 𝑏, 𝑈 𝑏) ≤ 8. (2.5)

Область вариации чисел зубьев примерно соответствует условию для шестерен и колес ступенчатых рядов:

20(14) ≤ 𝑧𝑎 ≤ 30 *;* 20 ≤ 𝑧𝑔 ≤ 120 *;* 20 ≤ 𝑧𝑓 ≤ 45. (2.6)

Поскольку ГОСТ 13733—77 не распространяется на числа зубьев планетарных передач, значения чисел зубьев могут приниматься любыми из диапазона (2.6). Для центрального колеса с внутренними зубьями величина 𝑧𝑏 не может быть задана произвольно и определяется при выбранных значениях 𝑧𝑎, 𝑧𝑔, 𝑧𝑓 из условия соосности. При назначении 𝑧𝑏 необходимо предусматривать исключение интерференции зубьев в зацеплениях 𝑔 — 𝑏 и 𝑓 — 𝑏. Так как в приборных устройствах используются в основном зубчатые передачи внутреннего зацепления без смещения, при значениях чисел зубьев 𝑧𝑔,𝑓 ≤ 27 отсутствие интерференции во внутреннем зацеплении с параметрами исходного контура ℎ𝑎 = 1 *и* 𝛼 = 20*°* возможно при следующем условии:

𝑧𝑏 ≥

(𝑧𝑔,𝑓 − 34)

(2 ∗ 𝑧𝑔,𝑓 − 34)

(2.7)

При 𝑧𝑔,𝑓 > 27 отсутствие интерференции гарантируется принятыми выше неравенствами (2.5), ограничивающими минимально возможные передаточные числа 𝑢𝑔𝑏 и 𝑢𝑓𝑏, при которых разность чисел зубьев 𝑧𝑏 − 𝑧𝑔,𝑓 ≥ 7.. .9.

Проектируя планетарные механизмы, нужно удовлетворить трем условиям: соосности, соседства и сборки.

Условие соосности требует равенства межосевых расстояний передач, входящих в механизм, т. е.

10

𝑎𝑤𝑎𝑔 = 𝑎𝑤𝑔𝑏 *,* 𝑎𝑤𝑎𝑔 = 𝑎𝑤𝑓𝑏. (2.8)

Эти условия для передач, где все колеса изготовлены без смещения, имеют вид: для механизма 𝐴𝑎ℎ

𝑎𝑎𝑔 = 𝑎𝑔ℎ *,* 𝑧𝑏 = 𝑧𝑓 + (𝑚21) ∗ (𝑧𝑎 + 𝑧𝑔); (2.9)

для механизма 𝐵𝑎ℎ

𝑎𝑎𝑔 = 𝑎𝑓𝑏 *,* 𝑧𝑏 = 𝑧𝑓 + (𝑚21) ∗ (𝑧𝑎 + 𝑧𝑔), (2.10)

где 𝑎𝑎𝑔, 𝑎𝑔𝑏, 𝑎𝑓𝑔 —делительные межосевые расстояния соответствующих пар колес; 𝑚1 и 𝑚2 — модули передач *а* — 𝑔 и 𝑓— 𝑏.

Выполнение условия (2.10) существенно ограничивает для механизма 𝐵𝑎ℎ выбор возможных сочетаний чисел зубьев, обеспечивающих требуемое передаточное отношение (2.3). Для увеличения количества возможных

вариантов сочетаний чисел зубьев поступают точно так же, как и при проектировании соосного ступенчатого редуктора: передачу 𝑎 *—*𝑔 механизма 𝐵𝑎ℎ выполняют с суммарным смещением 𝑥𝛴 = 𝑥𝑎 + 𝑥𝑔 ≠ 0 (передача 𝑓—𝑏*—* без смещения). Тогда аналогично (1.17):

𝑎𝑤𝑗 = 𝑎 + 𝑦 𝑚 (1.17)

где 𝑎𝑤𝑗 – межосевое расстояние, 𝑗 – коэффициент воспринимаемого смещения, условие соосности примет вид:

𝑎𝑤𝑎𝑔 = 𝑎𝑎𝑔 + 𝑚1𝑦 = 𝑎𝑤𝑓𝑏 = 𝑎𝑓𝑏 = 0,5𝑚2(𝑧𝑏 − 𝑧𝑓), (2.11)

т. е. необходимый коэффициент воспринимаемого смещения 𝑦 для передачи 𝑎 — 𝑔

𝑦 = (𝑎𝑓𝑏 − 𝑎𝑎𝑔)/𝑚1 ; 𝑦 <= |𝑦|, (2.12)

11

где |𝑦| *—* предельное значение коэффициента воспринимаемого смещения (обычно |𝑦| <= 1,0).

Диаметр окружности вершин колеса определяется по формуле (1.18):

𝑑𝑎𝑖𝑗 = 𝑑𝑖𝑗 + 2ℎ𝑎𝑚 , (1.18)

где 𝑑𝑖𝑗 – делительный диаметр, ℎ𝑎 – коэффициент ножки зуба.

При числе сателлитов 𝑛𝑤 => 3, равномерно расположенных по окружности диаметра 𝐷ℎ = 2𝑎𝑤𝑎𝑔 (см. рис. 2.1, в), необходимо предусмотреть зазор 𝛿 между окружностями вершин сателлитов. Это требование, называемое условием соседства, для передач без смещения имеет вид:

(𝑧𝑎 + 𝑧𝑔) 𝑠𝑖𝑛(𝜋/𝑛𝑤) − (𝑧𝑔 + 2) <= 𝛿 ∗ (2.13) (𝑧𝑎 + 𝑧𝑔) 𝑠𝑖𝑛(𝜋/𝑛𝑤) − (𝑚1/𝑚2)(𝑧𝑓 + 2) => 𝛿 ∗ (2.14)

𝛿 ∗ = 𝛿/𝑚1,2

где 𝛿 ∗ — относительный зазор. Для механизма 𝐴𝑎ℎ требуется выполнение только условия (2.13). Величина 𝛿 ∗ устанавливается в пределах 0,5...5 в

зависимости от модуля передач, технологических и эксплуатационных условий. В данном расчете с учетом мелкомодульных зацеплений принято 𝛿 ∗= 5, что позволяет использовать условия (2.13), (2.14) также и для передач со смещением.

При числе сателлитов 𝑛𝑤 = 2 одновременное зацепление зубьев сателлитов и центральных колес возможно при выполнении условия сборки:

для механизма𝐴𝑎ℎ

(𝑧𝑎 + 𝑧𝑏)/𝑛𝑤 = 𝐼, (2.15)

для механизма 𝐵𝑎ℎ

(𝑧𝑏𝑧𝑔 + 𝑧𝑎𝑧𝑓)/(𝑧𝑔𝑛𝑤) = 𝐼, (2.16)

где 𝐼— целое число.

Учитывать условия (2.16) необходимо при нерегулируемом угловом положении зубчатых венцов сателлитов 𝑔 и 𝑓, когда двухвенцовый сателлит изготавливается цельным или, венцы жестко фиксируются в одном блоке при

12

сборке. В приборных конструкциях, где часто используется штифтовое раздельное крепление сателлитов 𝑔 и 𝑓 соотношение (2.16) может и не выполняться, но конструкция при этом получается менее компактной.

Для высокоскоростных планетарных механизмов (в основном при окружных скоростях зубчатых колес свыше 10 м/с) с целью повышения выносливости и износостойкости зубьев необходимо подбирать числа зубьев сопряженных колес таким образом, чтобы они не имели общих множителей [см. условие (1.6)], а число зубьев центральных колес не было бы кратно числу сателлитов, т. е.

𝑧1𝑗/𝐾 = 𝐼1 = 𝐼2 = 𝑧2𝑗/𝐾 (1.6)

где 𝐾,𝐼1, 𝐼2 – целые числа, причем 𝐾 = 1,2, … 𝑧1𝑗,

𝑧𝑎𝑏/𝑛𝑤 ≠ 𝐾, (2.17)

где *К —* целое число.

Вписывание конструкции в заданный габарит *А*𝑟 требует выполнения следующих геометрических условий (см. рис. 2.1, а, б):

для механизма 𝐴𝑎ℎ

*А*𝑟 => 𝐷𝑝; 𝐷𝑝 = 𝑑𝑓𝑏 = 𝑚1[𝑧𝑏 + 2(ℎ𝑎 + 𝑐∗)] (2.18)

для механизма 𝐵𝑎ℎ

*А*𝑟 => 𝐷𝑝1; 𝐷𝑝1 = 2𝑎𝑎𝑔 + 𝑑𝑎𝑔 (2.19) *А*𝑟 => 𝐷𝑝2; 𝐷𝑝2 = 𝑑𝑓𝑏 = 𝑚2[𝑧𝑏 + 2(ℎ𝑎 + 𝑐∗)] (2.20)

В приведенных формулах: 𝐷𝑝 и 𝐷𝑝1,2 *—* реализуемые диаметральные габаритные размеры механизма по зубчатым венцам колес; 𝑑𝑓𝑏 — диаметр окружности впадин колеса 𝑏*;* 𝑑𝑎𝑔*—*диаметр окружности вершин сателлита 𝑔*;*

ℎ∗ и 𝑐∗— параметры исходного контура зубчатых колес (ГОСТ 13755— 81).

Габаритные размеры зубчатых колес 𝑎, 𝑔,𝑓 определяются по формулам (1.15), (1.18), колеса 𝑏 *—* по формулам (2.18) или (2.20), межосевые расстояния — по формулам (1.19) и (2.11).

13

𝐴𝜔𝑗 = 𝐴 = 0.5(𝑧1𝑗 + 𝑧2𝑗)𝑚 (1.19)

Совместный учет вышеприведенных соотношений обычно не позволяет точно реализовать передаточные отношения. Поэтому при проектировании допускается некоторая погрешность ∆𝑖*пл* (*в* %) реализации требуемого передаточного отношения 𝑖*пл* механизма:

∆𝑖*пл* = 100 |𝑖*пл* − 𝑖*пл*| 𝑖*пл* ; ∆𝑖*пл* ≤ |∆𝑖|, (2.21)

где |∆𝑖| − допускаемое значение погрешности.

**3 АЛГОРИТМЫ ПРОЕКТИРОВОЧНОГО РАСЧЕТА НА ЭВМ**

Для расчета кинематических и габаритных параметров планетарных механизмов выбран такой же принцип построения алгоритма, как и для ступенчатых рядных передач. Варианты решения задачи получаются в ходе последовательного целенаправленного перебора сочетаний чисел зубьев центральных колес и сателлитов с проверкой принятых кинематических, геометрических и иных условий. Диапазон изменений чисел зубьев ограничивается, например, соотношением (2.5). Задача оптимизации конст-рукции механизма здесь непосредственно не рассматривается, однако получаемые в ходе расчета решения позволяют поставить такую задачу в качестве завершающей операции.

Рассмотрим алгоритм проектировочного расчета планетарного редуктора

𝐵𝑎ℎ (см. рис. 2.1, *а)* при заданных габаритных *ограничениях*∗. На рис. 2.3 приведена структурная схема алгоритма. Исходными данными, ввод которых

осуществляется блоком 1, являются: 𝑖*пл —* требуемое передаточное отношение планетарного механизма; [∆𝑖] −допускаемая погрешность реализации

передаточного отношения; 𝑚1,𝑚2 *—* модули зубчатых передач *а* — 𝑔, 𝑓—𝑏 соответственно*;* 𝐴𝑟— предельный габаритный размер механизма по зубчатым венцам колес1; 𝐿𝑇𝑅 — условно кодируемое требование обеспечения повышенной выносливости и износостойкости зубьев за счет выполнения условий (2.15) и (1.6); 𝑧𝑎𝑚𝑖𝑛 , 𝑧𝑎𝑚𝑎𝑥 — минимальное и максимальное числа

1 Если габарит не ограничивается, а нужно только подобрать числа зубьев, то величина Ar задается достаточно большой [например, можно использовать зависимость 𝐴𝑟 ≥ 𝑚(𝑧𝑎𝑚𝑎𝑥 + 2𝑧𝑎𝑚𝑎𝑥) + 5. .10,которая прямо следует из формул (2.18) и (2.19)]

14

зубьев солнечного колеса *а,* определяющие границы области его возможных значений; 𝑧𝑔𝑚𝑖𝑛 , 𝑧𝑔𝑚𝑎𝑥 — тo же, для сателлита *g;* 𝑧𝑓𝑚𝑖𝑛, 𝑧𝑓𝑚𝑎𝑥 – то же для сателлита f; 𝑛𝑤𝑜, 𝑛𝑤𝑘 *—* минимальное и максимальное значения числа сател-литов, определяющие границы области изменения 𝑛𝑤 при расчете.

Процедура анализа возможных сочетаний чисел зубьев и отыскания вариантов решения организована в данном алгоритме в виде четырех вложенных циклов. Открывается внешний цикл, в котором число сателлитов 𝑛𝑤 изменяется от 𝑛𝑤𝑜 до 𝑛𝑤𝑘*.* По окончании цикла — прекращение расчета. При принятом 𝑛𝑤 варьируются числа зубьев центрального колеса *а*, сателлита *g,* и сателлита 𝑓*.* Внутри внешнего цикла открывается внутренний, в котором число зубьев 𝑧𝑎 изменяется от 𝑧𝑎𝑚𝑖𝑛 до 𝑧𝑎𝑚𝑎𝑥 , внутри цикла по числам зубьев солнечного колеса —третий цикл, где меняется 𝑧𝑔 от 𝑧𝑔𝑚𝑖𝑛 до 𝑧𝑔𝑚𝑎𝑥, внутри цикла по числам зубьев первого сателлита – четвертый цикл, где меняется 𝑧𝑓 от 𝑧𝑓𝑚𝑖𝑛 до 𝑧𝑓𝑚𝑎𝑥.

Сразу после входа в цикл по числам зубьев первого сателлита g происходит проверка выполнения условий (2.15) и (1.6) износостойкости для колес ступени a-g. В случае невыполнения условия (1.6), когда есть общие множители у чисел зубьев, происходит переход к следующей итерации цикла. В случае невыполнения условия (2.15), когда присутствуют общие множители у числа зубьев солнечного колеса и числа сателлитов 𝑛𝑤, происходит выход из обоих циклов для расчета при следующем по циклу значении числа сателлитов 𝑛𝑤.

После вышеописанных действий проверяются условия вхождения передаточного отношения ступени a-g в требуемый диапазон (2.5). В случае, если передаточное отношение не вошло в диапазон слева по числовой оси, то происходит переход к следующей итерации цикла по числу зубьев сателлита 𝑧𝑔. Если же передаточное отношение больше необходимого, то происходит переход к следующей итерации цикла по числу зубьев солнечного колеса 𝑧𝑎.

При успешном прохождении комбинацией чисел зубьев колес ступени a-g проверок на общие множители происходит расчет геометрических параметров колес данной ступени и производится проверка вхождения размера ступени в заданные габаритные ограничения. Если ступень не вошла в заданный габарит, то происходит переход к следующей итерации цикла по числу зубьев сателлита 𝑧𝑔, что даст следующую комбинацию чисел зубьев с большим передаточным отношением.

15

В дальнейшем внутри цикла после проверок условий открывается еще один внутренний цикл по числу зубьев второго сателлита 𝑧𝑓. Первой операцией в цикле является определение числа зубьев 𝑧𝑏 коронного колеса 𝑏 исходя из условия соосности механизма. Внутри этого цикла получаются комбинации чисел зубьев всех колес редуктора, что позволяет проверять механизм на выполнение условий отсутствия интерференции зубьев, сборки, соседства. Сперва проверяется вхождение передаточного отношения ступени b-f в требуемый диапазон (2.5). Если данное отношение лежит ниже диапазона, то происходит переход к следующей итерации цикла по 𝑧𝑓, если выше – то к следующей итерации цикла по числу сателлитов 𝑛𝑤. После этого выдвигается общее требование возможности синтеза механизма при данной комбинации чисел зубьев колес и числа сателлитов: одновременное выполнение условий отсутствия интерференции, соседства и сборки. Если общее условие не выполнено, то происходит переход к следующей итерации цикла по числу зубьев второго сателлита 𝑧𝑓.

Далее необходимо рассчитать геометрические параметры ступени второй сателлит-коронное колесо и проверить вхождение назначенного коэффициента суммарного смещения на вхождение в предельное значение, заданное пользователем. Если полученный коэффициент больше заданного предельного значения, то – переход к новой итерации цикла по числу зубьев второго сателлита 𝑧𝑓.

Полученная в ходе расчетов комбинация геометро-кинематических параметров проходит проверку на вхождение в допустимую погрешность по передаточному отношению (производится расчет передаточного отношения всего механизма) и на вхождение размеров второй ступени b-f в заданные габаритные ограничения. Если погрешность полученного механизма больше заданной пользователем, то происходит переход к новой итерации вышестоящего цикла по числу зубьев первого сателлита 𝑧𝑔. Если же габариты полученного механизма превышают заданные, происходит переход к следующей итерации данного цикла по числу зубьев второго сателлита 𝑧𝑓.

Когда полученная конфигурация параметров соответствует условиям допустимой погрешности и габаритов, то происходит вывод получившихся значений и выход из основного цикла по 𝑛𝑤, расчет прекращается.

16

Рисунок 8 — Блок-схема основного алгоритма расчета

На рисунке 8 представлена блок-схема основного алгоритма расчета планетарного редуктора со сдвоенными сателлитами.

17

**4 ПРОГРАММЫ ДЛЯ ЭВМ ПРОЕКТИРОВОЧНОГО РАСЧЕТА ПЛАНЕТАРНЫХ МЕХАНИЗМОВ**

Для решения задачи построения информационной системы был использован объектно-ориентированный подход программирования на языке C#. Создан класс «Planet» планетарных механизмов. Поля класса обозначены в табл. 1, программное описание полей класса «Planet» показано на листинге хх. Описание класса Planet размещено в отдельном файле проекта Planetarka.cs.

Из программного кода (листинг 1) можно увидеть, что в описании класса объявляются рассчитываемые, сервисные, входные переменные. Для входных переменных устанавливается связь с полями класса фреймворка, для которых установлены методы set и get установки и получения значений, а так же свойство OnPropertyChanged(), регулирующее направление изменения переменной (в данном случае одностороннее, так как изменение может произойти только из объекта класса фреймворка).

*Таблица 1 - Поля класса "Planet" планетарных редукторов*

18

19

*Листинг 1 — Описание* **using** System**;**

**using** System**.**Collections**.**Generic**; using** System**.**ComponentModel**;**

**using** System**.**Runtime**.**CompilerServices**;**

**public** class Planet**:**INotifyPropertyChanged **{**

//-----------------------------РАССЧИТЫВАЕТСЯ МЕТОДАМИ--------------------------

**public** int Za**,** Zb**,** Zg**,** Zf**,** N**;**

// Z1 - число зубьев солнечного, // Z2 - число зубьев сателлита, // Z3 - число зубьев короны,

// N - число сателлитов, // m - модуль ступени.

**private** double U**,** Aw1**,** Aw2**,** A**;** // U - передаточное отношение редуктора, //M1 - модуль первой ступени,

//M2 - модуль второй ступени

//-------------------------ВВОДИТСЯ ИЗ ИНТЕРФЕЙСА---------------------------

**private** int zamin**;**

**public** int ZaMin **{ get { return** zamin**; } set {** zamin **= (**int**)value;** OnPropertyChanged**(); } }**

**public** static List**<**string**>** Result **= new** List**<**string**>(**13**); private** int zamax**;**

**public** int ZaMax **{ get { return** zamax**; } set {** zamax **= value;** OnPropertyChanged**(); } }**

**private** int zgmin**;**

**public** int ZgMin **{ get { return** zgmin**; } set {** zgmin **= value;** OnPropertyChanged**(); } }**

**private** int zgmax**;**

**public** int ZgMax **{ get { return** zgmax**; } set {** zgmax **= value;** OnPropertyChanged**(); } }**

**private** int zfmin**;**

**public** int ZfMin **{ get { return** zfmin**; } set {** zfmin **= value;** OnPropertyChanged**(); } }**

**private** int zfmax**;**

**public** int ZfMax **{ get { return** zfmax**; } set {** zfmax **= value;** OnPropertyChanged**(); } }**

**private** int nMin**;**

**public** int NMin **{ get { return** nMin**; } set {** nMin **= value;** OnPropertyChanged**(); } }**

**private** int nMax**;**

**public** int NMax **{ get { return** nMax**; } set {** nMax **= value;** OnPropertyChanged**(); } }**

20

**private** double m1**;**

**public** double M1 **{ get { return** m1**; } set {** m1 **= value;** OnPropertyChanged**(); } }**

**private** double m2**;**

**public** double M2 **{ get { return** m2**; } set {** m2 **= value;** OnPropertyChanged**(); } }**

**private** double \_du**;**

**public** double du **{ get { return** \_du**; } set {** \_du **= value;** OnPropertyChanged**(); } }**

**private** double ut**;**

**public** double UT **{ get { return** ut**; } set {** ut **= value;** OnPropertyChanged**(); } }**

**private** double \_ag**;**

**public** double ag **{ get { return** \_ag**; } set {** \_ag **= value;** OnPropertyChanged**(); } }**

**private** double haz**;**

**public** double HAZ **{ get { return** haz**; } set {** haz **= value;** OnPropertyChanged**(); } }**

**private** double cz**;**

**public** double CZ **{ get { return** cz**; } set {** cz **= value;** OnPropertyChanged**(); } }**

**private** double yd**;**

**public** double YD **{ get { return** yd**; } set {** yd **= value;** OnPropertyChanged**(); } }**

**public** bool LTR**;**

**private** double DU**;**

//UT - требуемое передаточное отношение; // du - требуемая погрешность;

// ag - требуемый габарит;

//-----------------------------------------------------------------------------

//---------------СЕРВИСНЫЕ ПЕРЕМЕННЫЕ (ИСПОЛЬЗУЮТСЯ ТОЛЬКО В РАСЧЕТЕ)-----------

**private** double Da**,** Db**,** Dg**,** Df**,** DR1**,** DR2**,** DR**,** AW**,** Uvr**,** Y**,** D1**,** D2**,** ALF**;**

**private** int KPZ**;** // Управляющий признак, который определяется по картинке в интерфейсе

//-----------------------------------------------------------------------------

//----------------------ВЫВОДЯТСЯ НА ЭКРАН ПОСЛЕ РАСЧЕТА-----------------------

//-----------------------------------------------------------------------------

// Za - число зубьев солнца; // Zb - число зубьев короны;

// Zg - число зубьев сателлита 1; // Zf - число зубьев сателлита 2;

21

// N - число сателлитов;

// Da - Db - Dg - Df - диаметры делительных окружностей коле; // А - габарит механизма

// Y - суммарный коэффициент смещения; // U - передаточное отношение;

// DU - погрешность реализации передаточного отношения.

//-------------------------------------------------------

Для проведения расчетов в информационной системе расчета применены вспомогательные функции, которые являются методами класса.

Рассмотрим принципы работы вспомогательных функций по приведенным ниже листингам их программных кодов, написанных на языке С#.

Рассмотрим алгоритм функции CheckLTR (листинг 1), которая выполняет проверку наличия общих множителей у чисел зубьев колес и числа зубьев колеса с числом сателлитов. Тип возвращаемого значения – булева переменная. Аргументы: два целых числа. Если общие множители есть, возвращается 1, если нет или числа равны – 0.

*Листинг 1 — Функция проверки наличия общих множителей* private bool CheckLTR (int K, int P)

//Проверка наличия общих множителей у целых чисел при требовании износостойкости. //Общие есть – 1, нет - 0.

{

if (LTR) {

if (K % P == 0 || P % K == 0) return true; else return false;

}

else return false; }

Булева функция Sosedstvo (листинг 2) принимает в качестве аргументов числа зубьев солнечного колеса и всех сателлитов, число сателлитов и модули ступеней. Возможность соседства проверяется по соотношению (2.14). Если соседство невозможно, то функция возвращает 1, в обратном случае – 0.

*Листинг 2 — Функция проверки условия соседства*

private bool Sosedstvo (int Zsol, int Zs1, int Zs2, int Zk, double m1, double m2, int Nw)

//Проверка соседства. Соседство невозможно - 1, возможно - 0.

{

if ((Zsol + Zs1) \* Math.Sin(3.14 / Nw) - (Zs2 + 2) \* m2 / m1 < 5 && Nw >= 3) return true;

else return false;

22

}

Функция Sborka (листинг 3) проверяет условие сборки. Функция так же возвращает логическую переменную. В качестве аргументов принимаются числа зубьев всех колес редуктора и число сателлитов. Если условие сборки (2.16) не выполняется, то возврат 1, в обратном случае 0.

*Листинг 3 — Функция проверки условия сборки*

private bool Sborka (int Zsol, int Zs1, int Zs2, int Zk, int Nw)

//Проверка условия сборки. Собрать нельзя - 1, можно - 0.

{

if ((Zk \* Zs1 + Zsol \* Zs2) / (Zs1 \* Nw) - Math.Round((double)(Zk \* Zs1 + Zsol \* Zs2) / (Zs1 \* Nw), 0) != 0 && Nw >= 2) return true;

else return false; }

Функция Interfer (листинг 4) определяет наличие интерференции колес в планетарном редукторе по соотношению (2.7) и соответствующим значениям чисел зубьев. Возвращаемое значение – булева переменная. Аргументы – числа зубьев второго сателлита и короны, вещественная переменная коэффициента ножки зуба. Если случилась интерференция зубьев колес, то функция возвращает 1, в противном случае – 0.

*Листинг 4 — Функция проверки интерференции зубьев сателлитов и короны* private bool Interfer(int Zs2, int Zk, double haz)

//Проверка возможности наличия интерференции. Интерференция есть - 1, нет -0.

{

if (((Zs2 \* Zs2 - 34) / (2 \* Zs2 - 34) > Zk && Zs2 <= 27 && haz == 1) || (Zk - Zs2 < 7 && Zs2 > 27)) return true;

else return false;

}

Для расчетов геометрических параметров обеих ступеней введены методы функции GeomParamSolnSAT и GeomParamSolnKOR соответственно. Тексты программ функций приведены на листингах 5, 6. Данные функции так же являются методами класса Planetarka и применяются для заполнения соответствующих полей.

*Листинг 5 — Функция расчета параметров ступени солнце-сателлит a-g* private void GeomParamSolnSAT()

23

//Вычисление геометрических параметров ступени солнечное колесо - сателлит1

{

Da = Math.Round(M1 \* Za,2); Dg = Math.Round(M1 \* Zg,2);

Aw1 = Math.Round((Da + Dg) / 2,2); DR1 = Math.Round(Aw1 + Dg/2,2);

}

*Листинг 6 — Функция расчета параметров ступени сателлит-корона f-b* private void GeomParamSolnKOR()

//Вычисление геометрических параметров ступени сателлит 2-коронное колесо

{

Df = Math.Round(M2 \* Zf,2); Db = Math.Round(M2 \* Zb,2);

Aw2 = Math.Round((Db - Df) / 2,2);

DR2 = Math.Round(M2 \* (Zb + 2 \* (HAZ + CZ)),2); }

Программный код функции основного алгоритма ZTMM46 приведен на листинге 7.

Главная функция расчета представляет собой конструкцию из вложенных циклов, где происходит подбор параметров для наиболее подходящей конфигурации планетарного механизма. Наиболее подходящей конфигурацией является такая, в которой выполняются условия синтеза планетарного механизма и габаритный размер редуктора входит в диапазон заданных ограничений.

Внешний цикл устанавливает число сателлитов рассчитываемого механизма, внутри трех вложенных в него циклов происходит подбор комбинации перебором чисел зубьев солнечного колеса и сателлитов.

Внутри цикла по числу зубьев Zg на основе числа сателлитов, числа зубьев солнечного колеса и самого значения Zg проводится расчет параметров, которые могут быть получены на данном этапе без учета свойств сателлита f. Такими параметрами являются геометрические размеры колес ступени солнечное колесо – сателлит g. Параметры проходят необходимые проверки, в зависимости от результатов которых (вхождение ступени в габаритные ограничения, наличие или отсутствие общих множителей при расчете при повышенной износостойкости) происходит переход либо на следующую итерацию настоящего цикла, либо с помощью оператора безусловного перехода goto - на следующую итерацию одного из циклов, находящихся выше по иерархии. Таким образом перебор и расчет параметров значительно ускоряется,

24

так как заданные переходы по иерархии отметают заведомо бесполезные комбинации значений. Если все проверки были успешно пройдены, то запускается последний цикл по числу зубьев сателлита f.

Внутри этого цикла по числу зубьев сателлита f уже имеется возможность получить все параметры, необходимые для расчета полной конфигурации механизма. Вычисляются параметры ступени сателлит f – коронное колесо b, геометрические размеры проверяются на вхождение в заданный предел, определяется общий габаритный размер редуктора (как наибольший из диаметра коронного колеса и суммы межосевого расстояния ступени солнце-сателлит g с радиусом сателлита g). Проверяется возможность постройки механизма по полученной конфигурации (отсутствие интерференции зубьев короны и сателлита f, выполнение условий сборки, соседства), вхождение коэффициента суммарного смещения в заданный предел.

При успешном прохождении проверок конфигурация считается удовлетворяющей всем требованиям и условиям, происходит выход из высшего по иерархии цикла по числу сателлитов. Тем самым расчет прекращается, полученные данные идут на заполнение соответствующих полей класса и на последующую выдачу пользователю посредством интерфейса.

*Листинг 7 — Основная функция расчета планетарного редуктора* public void ZTMM46()

{

for (NW = NMin; NW <= NMax; NW++) {

Za = ZaMin;

while (Za != ZaMax) {

Zg = ZgMin;

while (Zg != ZgMax) {

if (CheckLTR(Za, Zg)) goto M80; //проверка наличия общих множителей колес ступени при требовании износостойкости

if (CheckLTR(Za, NW)) goto M100;//проверка общих множителей числа зубьев солнца и числа сателлитов при требовании износостойкости

if (Zg / Za < 1) goto M80;// проверка допустимости передаточного отношения ступени a-g

if (Zg / Za > 6) goto M100;// проверка допустимости передаточного отношения ступени a-g

GeomParamSolnSAT();//вычисление геометрических параметров ступени солнце-сателлит1

25

if (DR1 > ag) goto M80;// проверка вхождения механизма в габарит

Zf = ZfMin;

while (Zf != ZfMax) {

Zb = (int)(Zf + ((Za + Zg) \* M1 / M2));

if (Zb / Zf < 1.4) goto M50;// проверка допустимости передаточного отношения ступени f-b

if (Zb / Zf > 8) goto M200vix;// проверка допустимости передаточного отношения ступени f-b

if (Interfer(Zf, Zb, HAZ) || Sosedstvo(Za, Zg, Zf, Zb, M1, M2, NW) || Sborka(Za, Zg, Zf, Zb, NW)) goto M50;

//проверка невыполнения какого-либо из условий: отсутствия интерференции, соседства, сборки

GeomParamSolnKOR();//вычисление геометрических параметров ступени сателлит2-корона

Y = Math.Round((Aw2 - Aw1) / M1,2);

if (YD<Y&&Y!=0) goto M50;//проверка вычисленного коэффициента суммарного смещения на вхождение в предел

U = Math.Round((double)(1 + Zg \* Zb / (Za \* Zf)),2); DU = (UT - U) \* 100 / UT;

if (Math.Abs(DU) > du) goto M80;

if (DR2 > ag) goto M50; //проверка вхождения механизма в габарит

if (DR1 > DR2) A = DR1; else A = DR2; //принятие габарита механизма как наибольшего из габаритов ступеней

goto M200;//окончание расчета

M50: Zf++; }

M80: Zg++; }

M100: Za++; }

M200vix:; }

M200: N = NW; SetResult();

}

26

Использование оператора безусловного перехода goto с соответствующими метками в телах методов в данной задаче является оправданной мерой для упрощения расчета и уменьшения времени его проведения. В программной реализации выполнено главное условие безопасной работы оператора безусловного перехода – не происходит переходов к ранее выполненным инструкциям (все переходы только вниз по программе).

Помимо основных и вспомогательных в классе имеются служебные методы. На листинге 9 представлен метод SetResult() добавления в соответствующий список «Result», объявленный выше, для последующей передачи его в объект базы данных и вывода на экран. Метод Format используется для обеспечения требуемого количества знаков после запятой.

*Листинг 9 — Метод SetResult* **private** void SetResult**()**

**{**

Result**.**Add**(** Za**.**ToString**());** Result**.**Add**(**Zb**.**ToString**());** Result**.**Add**(**Zg**.**ToString**());** Result**.**Add**(**Zf**.**ToString**());** Result**.**Add**(**N**.**ToString**());** Result**.**Add**(**string**.**Format**(**"{0:0.000}"**,** Da**));** Result**.**Add**(**string**.**Format**(**"{0:0.000}"**,** Db**));** Result**.**Add**(**string**.**Format**(**"{0:0.000}"**,** Dg**));** Result**.**Add**(**string**.**Format**(**"{0:0.000}"**,** Df**));** Result**.**Add**(**string**.**Format**(**"{0:0.000}"**,** A**));** Result**.**Add**(**string**.**Format**(**"{0:0.000}"**,** Y**));** Result**.**Add**(**string**.**Format**(**"{0:0.000}"**,**U**));** Result**.**Add**(**string**.**Format**(**"{0:0.000}"**,** DU**));**

**}**

На листинге 10 приведен программный код метода, устанавливающего для полей класса значения, соответствующие исходным для показательного расчета (при нажатии кнопки «Пример расчета»).

*Листинг 10 — Метод, устанавливающий для полей класса значения, соответствующие исходным для показательного расчета*

**public** void SetExample**() {**

ZaMin **=** 18**;** ZaMax **=** 30**;** ZgMin **=** 25**;** ZgMax **=** 65**;** ZfMin **=** 22**;** ZfMax **=** 40**;** M1 **=** 0.4**;**

27

M2 **=** 0.5**;** NMin **=** 2**;** NMax **=** 4**;**

UT **=** 15**;** du **=** 7**;** ag **=** 60**;** YD **=** 0.5**;** HAZ **=** 1**;** CZ **=** 0.25**; }**

Для корректной работы кнопки «Очистить» в меню ввода данных для расчета в класс «Planet» введен служебный метод обнуления значений полей класса (листинг 11).

*Листинг 11 — Служебный метод обнуления значений* **public** void ClearInput**()**

**{**

ZaMin **=** 0**;** ZaMax **=** 0**;** ZgMin **=** 0**;** ZgMax **=** 0**;** ZfMin **=** 0**;** ZfMax **=** 0**;** M1 **=** 0**;** M2 **=** 0**;** NMin **=** 0**;** NMax **=** 0**;**

UT **=** 0**;** du **=** 0**;** ag **=** 0**;** YD **=** 0**;** HAZ **=** 0**;** CZ **=** 0**; }**

Также для своевременного обновления полей, был создан стандартный обработчик событий, срабатывающий при изменении свойства поля (вызывается в методе свойства get), который дает сигнал WPF либо обновить значение в поле ввода, либо изменить значение свойства в коде (так как включена двусторонняя зависимость).

*Листинг 18 — Обработчик событий*

**public event** PropertyChangedEventHandler PropertyChanged**;**

**public** void OnPropertyChanged**([**CallerMemberName**]**string prop **=** ""**) {**

**if (**PropertyChanged **!= null)**

PropertyChanged**(this, new** PropertyChangedEventArgs**(**prop**)); }**

**5 РАСТЧЕТЫ ПЛАНЕТАРНЫХ МЕХАНИЗМОВ НА ЭВМ** Выбор рационального диапазона изменения чисел зубьев колеса *а* и

сателлитов 𝑔 и 𝑓 т. е. назначение 𝑧𝑎,𝑔,𝑓𝑚𝑖𝑛 и 𝑧𝑎,𝑔,𝑓𝑚𝑎𝑥, обусловлен соображениями экономии машинного времени*.* Однако не следует и чрезмерно сужать этот диапазон, так как будут потеряны возможные варианты решения задачи. Назначение верхней границы числа зубьев сателлита *g* может быть 28

выполнено, исходя из габаритного условия (2.19). Подстановка в формулу (2.19) выражений (1.18), (1.19) дает

𝑧𝑔𝑚𝑎𝑥 ≥ *А*𝑟⁄2𝑚 − 0,5(𝑧𝑎𝑚𝑖𝑛 + 2), (2.22)

где 𝑚—модуль передачи 𝑎— 𝑔—𝑏*;* 𝑧𝑎𝑚𝑖𝑛— принятое минимальное число зубьев колеса *а.*

Учет требований 𝐿𝑇𝑅 ставит довольно жесткие ограничения, при которых не всегда существует решение задачи, даже при достаточно большом значении (5 ... 10%) допускаемой погрешности [∆𝑖]. Для того, чтобы получить решение, целесообразно расширить диапазон изменения чисел зубьев или снять требование.

Рассмотрим примеры выполнения проектировочных расчетов планетарных механизмов.

*Пример 1.* Подобрать числа зубьев и вписать планетарный механизм 𝐵𝑎ℎ привода вращения радиолокационной антенны с входным колесом *а* (см. рис. 2*)* в заданный габарит *А*𝑟 = 60 *мм*. Исходные данные: 𝑖*пл* = 9; модули зацеплений *а* — 𝑔*,* 𝑓 — 𝑏 —𝑚1 = 𝑚2 = 0,5 *мм*; [∆𝑖] = 0; передачи без смещения, требуется повышенная износостойкость. Расчет выполнить при количестве

сателлитов 𝑛𝑤 = 2 и 3.

Проведем расчет. Согласно заданию принимаем 𝐿𝑇𝑅 = 1. В соответствии с неравенствами (2.6) назначаем: 𝑧𝑎𝑚𝑖𝑛 = 20, 𝑧𝑎𝑚𝑎𝑥 = 30, 𝑧𝑔𝑚𝑖𝑛 = 20. Согласно (2.22): 𝑧𝑔𝑚𝑎𝑥 = 60⁄(2 ∗ 0,5) − 0,5(20 + 2) = 49, с небольшим запасом —

𝑧𝑔𝑚𝑎𝑥 = 50. Числа сателлитов принимаем от 𝑛𝑤𝑜 = 2 до 𝑛𝑤𝑘 = 5; для сателлита 𝑓 принимаем согласно (2.6): 𝑧𝑓𝑚𝑖𝑛 = 22, 𝑧𝑓𝑚𝑎𝑥 = 40.

На рисунке 9 представлены исходные данные в меню ввода, на рисунке 9 – результаты расчета.

29

Рисунок 9.1 — Исходные данные и результаты расчета

Рисунок 9.2 — Исходные данные и результаты расчета

*Пример 2.* Найти варианты реализации планетарного редуктора 𝐵𝑎ℎ (см. рис. 2*)* для 𝑖*пл* = 15,0 при 𝑛𝑤 = (2; 4). Остальные исходные данные:[∆𝑖] =

5%, 𝑚1 = 0,4 *мм*; 𝑚2 = 0,5 *мм*; 𝑛𝑤 = 60 *мм*; зацепление *а*— 𝑔 может быть выполнено со смещением, [у] = 1,0; при этом оба зацепления должны быть повышенной выносливости.

При подготовке исходных данных назначаем изменение чисел зубьев аналогично примеру 2.3.

Исходные данные в меню ввода приведены на рисунке 10, результаты расчета – на рисунке 11.

30

Рисунок 10 — Исходные данные

Рисунок 11 — Результаты расчета

**6 СООТНОШЕНИЯ И АЛГОРИТМЫ КИНЕМАТИКО-ГАБАРИТНОГО РАСЧЕТА ВОЛНОВЫХ ЗУБЧАТЫХ**

**МЕХАНИЗМОВ**

Расчет волнового механизма имеет существенную особенность по сравнению с другими типами зубчатых передач: если в рассмотренных ранее ступенчатых и планетарных механизмах ставилась задача вписаться в

31

требуемые габариты, то для волнового механизма часто необходимо получить конструкцию с размерами *не менее требуемых*, так как внутренняя полость гибкого колеса занята другими узлами (см. п.1 и рис. 2.2, а, *б*)*.* Поэтому передаточное отношение механизма не задается, а является одним из искомых параметров; в ходе расчета определяются также число зубьев и размеры гибкого и жесткого зубчатых колес, а при использовании кулачкового двухволнового генератора подбирается стандартный гибкий подшипник.

Передаточное отношение волнового механизма (рис. 3 и 4), представляющего собой аналог планетарного типа 2𝑘 — ℎ*,* может быть получено с использованием соотношения (2.1). Входное звено волнового механизма — генератор ℎ (аналог водила планетарного механизма), вместо центральных колес *а* и *b —* либо С и 𝐹 (для схемы рис. 3), либо 𝐹 и С (для схемы рис. 4)*.* Тогда для этих схем выражение 2.1) примет вид:

𝑖В = 𝑖ℎ𝐶 = 1⁄ 𝐹

𝐶ℎ

= 1⁄(1 − 𝑖𝐶𝐹); 𝑖В = 𝑖ℎ𝐹 = 1⁄ 𝐹ℎ = 1⁄(1 − 𝑖𝐹𝐶). (2.23)

Замена в (2.23) 𝑖𝐹𝐶 и 𝑖ℎ𝐹 соответствующими передаточными числами

𝑢𝐶𝐹 = 𝑧𝐶⁄𝑧𝐹 и 𝑢𝐹𝐶 = 𝑧𝐹⁄𝑧𝐶 *(*𝑧𝐶 и 𝑧𝐹 *—*числа зубьев жесткого и гибкого колес) дает:

𝑖В = 𝑖ℎ𝐶 = 𝑧𝐶⁄(𝑧𝐶 − 𝑧𝐹); 𝑖В = 𝑖ℎ𝐹 = −𝑧𝐹⁄(𝑧𝐶 − 𝑧𝐹). (2.24)

Знак минус для 𝑖ℎ𝐹 показывает, что направления вращения генератора и колеса 𝐹 противоположны.

Разность числа зубьев 𝑧𝐶 − 𝑧𝐹 должна быть кратной числу волн деформации; при 𝑧𝐶 − 𝑧𝐹 = 2 передаточное отношение волнового механизма максимально. С учетом этого

𝑖В = 𝑖ℎ𝐶 = 𝑧𝐶⁄2; 𝑖В = 𝑖ℎ𝐹 = −𝑧𝐹⁄2. (2.25)

Диапазон изменения передаточного отношения для схем рис. 2.2

70 ≤ 𝑖В ≤ 300. (2.26)

32

Минимальный диаметр 𝑑𝐹𝑚𝑖𝑛 начальной окружности гибкого зубчатого колеса (мм) определяется из двух условий:

прочности гибкого колеса

𝑑𝐹𝑚𝑖𝑛 ≅ 15,5√𝑇 ых, (2.27)

и габаритных ограничений

𝑑𝐹𝑚𝑖𝑛 = 𝐷Г + 2∆зап≅ 1,1𝐷Г, (2.28)

где 𝑇 ых − вращающий момент нагрузки (Н.м) на выходном валу механизма;

𝐷Г − габаритный размер, определяемый компоновкой внутренней полости гибкого колеса различными устройствами (рис. 3); ∆зап − необходимый запас, учитывающий зазоры между гибким колесом и внутренними устройствами, толщину стенки гибкого колеса. Величину 2∆зап можно принять равной 0,1𝐷Г. Проектный диаметр гибкого колеса 𝑑𝐹 принимается как наибольший из 𝑑𝐹𝑚𝑖𝑛 и 𝑑𝐹𝑚𝑖𝑛.

Проектный диаметр гибкого подшипника

𝐷ГП = 𝑑∗ + 0.99𝑚[6 − 2(ℎ∗ + 𝑐∗)], (2.29)

где 𝑚 — модуль передачи 𝐹 — С; ℎ∗ и 𝑐∗ − коэффициент высоты головки зуба и коэффициент радиального зазора, принимаемые в соответствии с ГОСТ 13755-81. Выбор гибкого шарикоподшипника по ГОСТ 23179—78 проводится по условию

𝐷ГП ≥ 𝐷ГП, (2.30)

где 𝐷ГП *—* наружный диаметр гибкого шарикоподшипника.

В табл. 2 приведены, основные размеры ряда гибких подшипников, применяемых в точной механике (см. также рис. 5*).*

*Таблица 2 - Параметры гибких радиальных шарикоподшипников по ГОСТ 23179-78*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Условное обозначение подшипника NP | Наружный диаметр 𝐷ГП, мм | Внутренний диаметр 𝑑ГП, мм | Ширина ВГП, мм | Предельная частота вращения  [𝑛ГП],об⁄мин |

33

Правильность выбора гибкого подшипника по частоте его вращения nгп проверяют по условию:

𝑛ГП = 𝑛вых 𝑖В; 𝑛ГП ≤ [ 𝑛ГП], (2.31)

где [ 𝑛ГП] − предельная частота вращения гибкого подшипника (см. табл. 2.2). Число зубьев гибкого колеса 𝑧𝐹 находится путем округления до меньшего

целого четного значения расчетного числа зубьев, вычисляемого по формуле

𝑧𝐹 = (𝐷ГП⁄𝑚) − 5,88 + 1,96(ℎ∗ + 𝑐∗). (2.32)

Число зубьев жесткого колеса при двухволновом генераторе

𝑧𝐶 = 𝑧𝐹 + 2. (2.33)

Для того чтобы в волновых механизмах избежать интерференции головок зубьев при разности 𝑧𝐶 − 𝑧𝐹 = 2, гибкое и жесткое колеса изготовляются со значительным смещением исходного контура. Коэффициенты смещения находятся по формулам:

для гибкого колеса

𝑥𝐹 = 3 + 0,01𝑧𝐹, (2.34)

для жесткого колеса

𝑥𝐶 = 𝑥𝐹 − 1 + 𝐾𝑤 (1 + 5 ∗ 10−5𝐾𝐹𝑧𝐹), (2.35)

34

где 𝐾𝑤 − коэффициент радиальной деформации гибкого колеса, принимаемый в среднем равным 1,1; 𝐾𝑤 ≅ 0,4 − коэффициент, характеризующий глубину захода зубьев колёс.

Диаметры гибкого и жесткого колес: делительных окружностей

𝑑𝐹,𝐶 = 𝑚𝑧𝐹,𝐶, (2.36)

окружностей вершин

𝑑𝑎𝐹 = 𝑑𝐹 + 2(𝑥𝐹 + 𝐾𝐹)𝑚, (2.37) 𝑑𝑎С = 𝑑𝐶 + 2(𝑥𝐶 + ℎ𝑎)𝑚; (2.38)

окружностей впадин

𝑑𝑓𝐹 = 𝑑𝐹 + 2(𝑥𝐹 − ℎ𝑎 − 𝑐∗)𝑚, (2.39) 𝑑𝑓𝐶 = 𝑑𝑎𝐶 + 2ℎ𝐶, (2.40)

где ℎ𝐶 − высота зуба жесткого колеса, зависящая от параметров инструмента (долбяка), используемого для изготовления колеса. При проектировочном расчете

ℎ𝐶 ≅ (ℎ𝑎 + 𝑐∗ + 𝐾𝐹)𝑚. (2.41)

Наиболее важные для проектировочного расчета конструктивные эле-менты гибкого и жесткого колес (см. рис. 2.2, г, д)*:* толщина стенки гибкого колеса под зубьями

𝛿𝐶3 = 𝑚[0,5𝑧𝐹 + 𝑥𝐹 − (ℎ𝑎 + 𝑐∗)] − 0,5𝐷ГП, (2.42)

толщина стенки стакана гибкого колеса

𝛿𝐶 ≅ 0,2𝑑𝐹; (2.43)

ширина зубчатого венца гибкого колеса

35

𝑏𝑤𝐹 ≅ 0,2 𝑑𝐹; (2.44)

ширина венца жесткого колеса

𝑏𝑤𝐶 = 𝑏𝑤𝐹 + 3 мм; (2.45)

длина стакана гибкого колеса

𝑙ст ≅ 𝑑𝐹; (2.46)

ширина пояска, вводимого для снижения перекоса зубьев и концентрации напряжений на наружном торце зубчатого венца гибкого колеса,

𝑏𝑘 > 0,06𝑑𝐹; (2.47)

толщина обода жесткого колеса под зубьями, обеспечивающая его радиальную деформацию в допускаемых пределах,

ℎоб ≅ 0,18𝑑𝐶. (2.48)

Алгоритм отыскания возможных решений при кинематико-габаритном расчете волнового механизма построен по принципу вариации сочетаниями модуля передачи *F — С* и размеров гибкого подшипника. В приводимом ниже алгоритме выбран следующий ряд наиболее употребительных модулей (мм): 0,3; 0,4; 0,5; 0,6; 0,8; 1,0, сведенных в числовой массив AM (6). На рис. 2.11 приведена структурная схема алгоритма. Первым блоком вводятся исходные данные: шифр типа волнового механизма — PR (если проектируется механизм по схеме рис. 3, то PR=1*,* a по схеме рис. 4 *—* PR=2): максимальный момент нагрузки на выходном валу механизма —Твых; частота вращения выходного вала—𝑛вых минимальный габаритный размер внутренней полости гибкого колеса — 𝐷*Г* начальное и конечное значения варьируемых при расчете модулей, соответствующих массиву AM (6) — NO, NK.

Сперва определяется проектный диаметр 𝑑𝐹 гибкого колеса. Затем производится определение индексов элементов массива модулей АМ, соответствующих введенным значениям модуля. Открывается цикл вариации значениями модулей от номера NO до номера NK массива модулей AM (6).

36

Вычисляется проектный диаметр 𝐷ГП гибкого подшипника. Так как в данном алгоритме принято ограничение типоразмеров гибких подшипников наружным диаметром 𝐷ГП𝑚𝑎𝑥 = 240 мм (см. табл. 2), то проводится соответствующая проверка, а при невозможности выбора подшипника формируется признак невозможности синтеза механизма. Выбор гибкого подшипника и определение его параметров (см. табл. 2) проводятся с помощью функции SVOL2.

С помощью функции SVOL3 определяется число зубьев гибкого колеса, после чего устанавливается число зубьев жесткого колеса*.* Передаточное отношение 𝑖𝑏 механизма определяется с помощью функции SVOL4 в зависимости от требуемого типа PR механизма. Этой же функцией проверяются неравенства (2.26), вычисляется признак 𝐼𝑃 возможности реализации механизма. Если 𝐼𝑃 = 1 (при 𝑖В < 300), необходимо выбрать подшипник с большим наружным диаметром Drп что осуществляется функцией SVOL2. Если 𝐼𝑃 = 3 (𝑖В > 300), то выдается сообщение о невозможности реализации механизма. Если 𝑖В находится в требуемых пределах (𝐼𝑃 = 2), то в блоках проводится проверка выбранного гибкого подшипника на допускаемую частоту вращения (2.31); при нарушении этого условия опять следует вывод о невозможности реализации механизма с последующим переходом к новому значению модуля*.* По выполнении всех принятых условий функцией SVOL5 рассчитываются геометрические параметры гибкого и жесткого колес в виде массива AZ (8). В функции SVOL6 определяются по формуле (2.42) и принятым выше соотношениям основные конструктивные элементы гибкого и жесткого колес. После выполнения данного шага расчетов следует переход к новому значению модуля и выполнение очередного шага расчетов. По окончании цикла вариации модулями *—* прекращение расчета.

37

Рисунок 2.2 — Алгоритм расчета волнового редуктора

**7 ПРОГРАММА И РАСЧЕТ НА ЭВМ ВОЛНОВЫХ ЗУБЧАТЫХ МЕХАНИЗМОВ**

Для проектирования волновых редукторов был создан класс волновых механизмов — «Wave». Все функции алгоритма, описанного выше, являются методами данного класса. На листингах 8 - 13 приведены тексты главного метода алгоритма «Construction» и вспомогательных функций-методов для кинематико-габаритного расчета волновых зубчатых механизмов в соответствии с рассмотренным алгоритмом. В табл. 3 приведены поля класса «Wave», используемые для исходных данных, промежуточных вычислений (проектные диаметры из разных соображений и текущие значения) и результатов расчета. Описание класса Wave расположено в отдельном файле проекта под названием Wave.cs.

38

*Таблица 3 - Поля класса волновых редукторов «Wave»*

39

На листинге 13 представлено программное объявление полей класса «Wave» волновых редукторов.

*Листинг 13 — Объявление полей класса волновых редукторов* **using** System**;**

**using** System**.**Collections**.**Generic**; using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**; using** System**.**ComponentModel**;**

40

**using** System**.**Runtime**.**CompilerServices**;**

**public** class Wave **:** INotifyPropertyChanged **{**

//--------------------------РАССЧИТЫВАЮТСЯ В МЕТОДАХ---------------------------

public List<string> NameParams { get; set; } = new List<string> { "Число зубьев колеса f", "Число зубьев колеса с", "Номер гибкого подшипника", "Модуль ступени", "Передаточное отношение","Коэффициент смещения колеса F", "Коэффициент смещения колеса C","Диаметр делительной окружности колеса F","Диаметр окружности вершин колеса F","Диаметр окружности впадин колеса F", "Диаметр делительной окружности колеса С", "Диаметр окружности вершин колеса С", "Диаметр окружности впадин колеса С", "Толщина стенки гибкого стакана под зубьями", "Tолщина стенки стакана колеса", "Ширина венца гибкого колеса", "Ширина венца жесткого колеса", "Ширина пояска для снижения перекоса зубьев", "Длина стакана гибкого колеса", "Толщина обода жесткого колеса под зубьями"};

**private** double U**,** Dgp**;**

//U - передаточное отношение волнового редуктора; //Dgp - диаметр гибкого подшипника;

**private** double**[]** az **= new** double**[**8**];**//Массив геометрических параметров зацепления колес F и С

//-----------------Разметка массива геометрических параметров-------------------

//az[0] - Xf - коэфф-т смещения колеса F; //az[1] - Xc - коэф-т смещения колеса С;

//az[2] - df - диаметр делительной окружности колеса F; //az[3] - daf - диаметр окружности вершин колеса F; //az[4] - dff - диаметр окружности впадин колеса F;

//az[5], az[6], az[7] - dc,dac,dfc - аналогичные параметры для колеса С.

**private** double**[]** ak **= new** double**[**7**];**// массив конструкционных параметров зацепления

//---------------Разметка массива конструкционных параметров-------------------

//ak[0] - sigmacz - толщина стенки гибкого стакана под зубьями; //ak[1] - sigmac - толщина стенки стакана колеса;

//ak[2] - bwf - ширина венца гибкого колеса; //ak[3] - bwc - ширина венца жесткого колеса;

//ak[4] - bk - ширина пояска для снижения перекоса зубьев; //ak[5] - l - длина стакана гибкого колеса;

//ak[6] - hob - толщина обода жесткого колеса под зубьями;

**public** int Zf**,** Zc**,** Ngp**;** //числа зубьев колес F, C

//Ngp - номер (обозначение) гибкого подшипника

41

**public** double modulfc**;**//modulfc - модуль ступени FC, определяемый на этапе конструирования объекта

**public** string**[,]** Result**;**

//U - передаточное отношение волнового редуктора //Dgp - диаметр гибкого подшипника

//-----------------------------------ВВОДЯТСЯ В ИНТЕРФЕЙСЕ----------------------------------

**public** double INo **{ get { return** mo**; } set {** mo **= (**double**)value;** OnPropertyChanged**(**"INo"**); } }**

**public** double INk **{ get { return** mk**; } set {** mk **= (**double**)value;** OnPropertyChanged**(**"INk"**); } }**

**public** double Itout **{ get { return** Tout**; } set {** Tout **= (**double**)value;** OnPropertyChanged**(**"Itout"**); } }**

**public** double Inout **{ get { return** Nout**; } set {** Nout **= (**double**)value;** OnPropertyChanged**(**"Inout"**); } }**

**public** double Idr **{ get { return** Dr**; } set {** Dr **= (**double**)value;** OnPropertyChanged**(**"Idr"**); } }**

**public** double Ihaz **{ get { return** Haz**; } set {** Haz **= (**double**)value;** OnPropertyChanged**(**"Ihaz"**); } }**

**public** double Icz **{ get { return** Cz**; } set {** Cz **= (**double**)value;** OnPropertyChanged**(**"Icz"**); } }**

//Tout - вращательный момент нагрузки //nout - частота вращения выходного вала

//Dr - минимальный габаритный размер внутренней полости гибкого колеса

//No - начальный номер поиска в массиве модулей; Nk - конечный номер диапазона поиска модуля в массиве модулей

//-----------------------------ВВОДЯТСЯ В ИНТЕРФЕЙСЕ---------------------------

**public** int PR **{ get; set; } =** 1**;** //public int No { get; set; } //public int Nk { get; set; }

**public** double mo **public** double mk **public** double Tout **public** double Nout **public** double Dr **public** double Haz

**public** double Cz

**{ get; set; } { get; set; } { get; set; } { get; set; } { get; set; } { get; set; }**

**{ get; set; }**

//--------------------------------------------СЛУЖЕБНЫЕ------------------------------------

private double haz, cz;// исходный производящий контур

private int private double

конструирования

IP;//индикатор для расчетов

dfpp;//проектный диаметр гибкого подшипника. определяется в блоке

42

private double блоке конструирования

private double

Dp;//проектный наружный диаметр гибкого подшипника. определяется в

ngp;//переменная частоты вращения подшипника. Используется для

проверки допустимости подшипника

private double ngpp;//переменная предельной частоты вращения выбранного подшипника. Используется для проверки допустимости подшипника.

private double sm;//временный модуль. перепишется в modulfc

В программном описании полей класса присутствуют подключенные пространства имен. Опишем те, которые не встречаются в описаниях классов интерфейса.

System.ComponentModel – пространство имен, которое предоставляет классы, используемые для реализации поведения компонентов и элементов управления во время разработки и выполнения. Это пространство имен содержит базовые классы и интерфейсы для реализации атрибутов и преобразователей типов, привязки к источникам данных и лицензирования компонентов. Методами классов этого пространства в программе осуществляются приведения типов.

System.Runtime.ComplierServices - пространство имен, предоставляющее функции средствам записи компилятора, которые используют управляемый код для задания в метаданных атрибутов, влияющих на поведение среды CLR (Common Language Runtime – общеязыковая исполняющая среда) во время выполнения. CLR компилирует код приложения на языке CIL, реализация компиляции которого компанией Microsoft называется MSIL, во время его исполнения, а также предоставляет MSIL-программам (а следовательно, и программам, написанным на языках высокого уровня, поддерживающих .NET Framework) доступ к библиотеке классов .NET Framework, или так называемой .NET FCL. Служебное пространство имен для корректной компиляции и отладки программы.

На листинге 14 представлен код программы функции подбора гибкого подшипника. Внутри функции объявлен и инициализирован динамический двумерный массив, содержащий в себе данные из таблицы 2. В соответствии с условиями максимальной частоты вращения и габаритных требований производится инициализация полей класса, относящихся к параметрам гибкого подшипника: обозначение, наружный диаметр, внутренний диаметр, предельная частота вращения. В соответствии со значениями этих полей будут заполняться остальные поля при конструировании объекта класса «Волновой редуктор».

43

*Листинг 14 — Функция подбора гибкого подшипника* private void Svol2()

{

int[,] a = new int[11, 5];

a[0, 0] = 806; a[1, 0] = 808; a[2, 0] = 809; a[3, 0] = 811; a[4, 0] = 812; a[5, 0] = 815; a[6, 0] = 818; a[7, 0] = 822; a[8, 0] = 824; a[9, 0] = 830; a[10, 0] = 836;

a[0, 1] = 42; a[1, 1] = 52; a[2, 1] = 62; a[3, 1] = 72; a[4, 1] = 80; a[5, 1] = 100; a[6, 1] = 120; a[7, 1] = 150; a[8, 1] = 160; a[9, 1] = 200; a[10, 1] = 240;

a[0, 2] = 30; a[1, 2] = 40; a[2, 2] = 45; a[3, 2] = 55; a[4, 2] = 60; a[5, 2] = 75; a[6, 2] = 90; a[7, 2] = 110; a[8, 2] = 120; a[9, 2] = 150; a[10, 2] = 180;

a[0, 3] = 7; a[1, 3] = 8; a[2, 3] = 9; a[3, 3] = 11; a[4, 3] = 13; a[5, 3] = 15; a[6, 3] = 18; a[7, 3] = 24; a[8, 3] = 24; a[9, 3] = 30; a[10, 3] = 35;

44

a[0, 4] = 4000; a[1, 2] = 4000; a[2, 4] = 3500; a[3, 4] = 3500; a[4, 4] = 3500; a[5, 4] = 3000; a[6, 4] = 3000; a[7, 4] = 2500; a[8, 4] = 2000; a[9, 4] = 1600; a[10, 4] = 1600;

for (int i = 0; i <= 10; i++) {

if ((Dp <= a[i, 1]) && (Nout <= a[i, 4]))

{

Dgp = a[i, 1]; Ngp = a[i, 0]; ngpp = a[i, 4]; break;

}

} return;

}

Программный код метода расчета чисел зубьев жесткого и гибкого колес представлен на листинге 15.

*Листинг 15 - Функция расчета числа зубьев колес* private void Svol3()

{

Zf = (int)Math.Round((Dgp / sm - 5.88 + (1.96 \* (haz + cz))), 0); Zc = Zf + 2;

return; }

В функции Svol4, программный код которой приведен на листинге 16, формируется признак возможности синтеза волнового механизма с полученными в ходе предшествовавших вычислений параметрами.

*Листинг 16 — Функция расчета передаточного числа и формирования управляющего признака*

private void Svol4()

45

{

if (PR <= 1) {

U = 0.5 \* Zf;

}

else U = 0.5 \* Zf; if (U <= 70)

{

IP = 1; }

else {

if ((300 - U) < 0) {

IP = 3; }

else IP = 2; }

return; }

Если синтез механизма возможен, то методами Svol5 и Svol6 (листинги 17 и 18) рассчитываются остальные необходимые геометро-кинематические параметры волнового механизма.

*Листинг 17 — Функция расчета геометро-кинематических параметров редуктора* private void Svol5()

{

double qw = 1.1; double qf = 0.4;

az[0] = Math.Round((3 + 0.01 \* Zf),2); // Xf az[1] = Math.Round(az[1] - 1 + qw \* (1 + 0.00005 \* qf \* Zf),2); // Xc az[2] = Math.Round(sm \* Zf,2); // df az[3] = Math.Round(az[2] + 2 \* (az[1] + qf) \* sm,2); //daf az[4] = Math.Round(az[2] + 2 \* (az[1] - haz - cz) \* sm,2); //dff az[5] = Math.Round(sm \* Zc,2); //dc az[6] = Math.Round(az[5] + 2 \* (az[1] - haz) \* sm,2); //dac az[7] = Math.Round(az[6] + 2 \* (haz + cz + qf) \* sm,2); //dfc return;

}

*Листинг 18 — Функция расчета остальных параметров редуктора* private void Svol6()

{

ak[0] = Math.Round(sm \* (0.5 \* Zf + az[0] - (haz + cz)) - 0.5 \* Dgp,2);//sigmacz

ak[1] = Math.Round(0.7 \* ak[0],2); ak[2] = Math.Round(0.2 \* az[2],2);

ak[3] = Math.Round(ak[2] + 3,2);

//sigmac //

//

ak[4] = Math.Round(0.06 \* az[3],2); // ak[5] = Math.Round(1.0 \* az[2],2); //

46

ak[6] = Math.Round(0.18 \* az[5],2); // return;

}

На листинге 19 представлен программный код метода основного расчетного алгоритма для геометро-кинематических параметров волнового редуктора при габаритных ограничениях.

*Листинг 19 — Основная функция расчета волнового редуктора* public void Construction()

{

int No, Nk;

No = 0; Nk = 5; haz = Haz;

cz = Cz;

double[] am = new double[6];//объявление массива модулей

am[0] = 0.3; am[1] = 0.4; am[2] = 0.5; am[3] = 0.6; am[4] = 0.8;

am[5] = 1.0;//инициализация массива модулей

double dfp = 15.5 \* (Math.Pow(Tout, 1 / 3));//локальная переменная диаметра подшипника из прочностных соображений

double dfr = 1.1 \* Dr;//локальная переменная диаметра подшипника из соображений габаритов

if (dfp >= dfr) dfpp = dfp; else dfpp = dfr;//определение проектного значения диаметра гибкого подшипника

//поиск индексов массива, в соответствии с введенным модулем, для последующего поиска по массиву модулей

for (int i = 0; i <= 5; i++) {

if (am[i] == mo) { No = i; } if (am[i] == mk) { Nk = i; }

}

if (mo == 0.7) { No = 3; } if (mk == 0.7) { Nk = 4; }

//------------------------------------------------------------------------------//----Поиск по массиву модулей оптимального варианта построения редуктора-------//------------------------------------------------------------------------------

for (int n = No; n <= Nk; n++) {

sm = am[n];

Dp = dfpp + 0.99 \* sm \* (6 - 2 \* (haz + cz));

47

if (Dp <= 240) {

Svol2(); Svol3(); Svol4();

if (IP == 1) {

Dp = Dp + 5; }

if (IP == 2) {

ngp = Nout \* U;

if ((ngpp - ngp) >= 0) {

Svol5(); Svol6(); modulfc = sm; break;

}

}

if (IP == 3) {

Svol5(); Svol6();

modulfc = 0;//обнуление модуля - индикатор того, что необходимо выводить сообщение о невозможности синтеза механизма

break;

} }

}

*Пример 1.* Выполнить расчет кинематических и габаритных параметров волнового зубчатого механизма привода манипулятора по схеме рис. 2.2.

Исходные данные: 𝑇 ых = 30 Н ∗ м; 𝑛вых = 35об⁄мин; 𝐷Г = 55 мм. Расчет провести для модулей 0,3 .. . 0,5 мм.

На рисунке 12 представлены исходные данные в меню ввода, на рисунке 13 – результаты расчета.

48

Рисунок 12 — Исходные данные

Рисунок 13 — Результаты расчета

**8 ОПИСАНИЕ ИНТЕРФЕЙСА И СТРУКТУРА ИНФОРМАЦИОННОЙ СИСТЕМЫ**

Учитывая, что для написания программы был выбран язык C#, а ключевой платформой запуска – операционная система Windows, для создания интерфейса рассматривались два фреймворка: WPF (Windows Presentation

49

Foundation) и WinForms. Так как WPF имеет возможность последующей поддержки, больший функционал, то выбор был остановлен именно на нем.

Размещение элементов пользовательского интерфейса было осуществлено с помощью языка гипертекстовой разметки XAML в специально назначенных для этого файлах. WPF позволяет писать приложения и без использования XAML, в этом случае все элементы программного интерфейса создаются непосредственно в коде на основном языке программы - C#, но XAML имеет ряд преимуществ:

 Возможность отделить графический интерфейс от логики приложения, благодаря чему над разными частями приложения могут относительно автономно работать разные специалисты: над интерфейсом -дизайнеры, над кодом логики - программисты.

 Компактность, понятность, удобство поддержки XAML-кода.

Все окна приложения помещаются в контейнеры компоновки (чаще всего это Grid). Благодаря компоновке мы можем удобным нам образом настроить элементы интерфейса, позиционировать их определенным образом. Например, элементы компоновки в WPF позволяют при изменении размера окна - сжатии или растяжении масштабировать элементы, что очень удобно, а визуально не создает дефектов изображения окна в виде шероховатостей и незаполненных участков на форме.

При формировании проекта для каждого класса интерфейса (любая страница или панель являются классами, значениями полей которых управляет пользователь с помощью методов классов, вызываемых по событию) по умолчанию создаются два файла: один с расширением .xaml.cs, в котором представлены все поля и методы класса (иными словами, логика работы приложения при нахождении пользователя в данном окне), и второй – с расширением .xaml, в котором находится XAML-код графического отображения объектов интерфейса (стили объектов, верстка окна и его дизайн).

Главное окно программы представляет собой элемент, использующий для отображения вкладки, т.е. каждая основная составляющая окна (описание алгоритмов, описание математической модели и т.д.) скрыта в своей вкладке. В WPF такой способ отображения достигается с помощью элемента TabControl, где каждая составляющая этого окна является TabItem этого элемента. Пример TabItem:

*Листинг 20 — Пример Tabltem*

<TabItem x:Name="glavnaya">

50

<TabItem.Header>

<StackPanel Orientation="Horizontal"> <TextBlock Margin="3">Главная</TextBlock>

</StackPanel> </TabItem.Header>

<TabItem.Content> <Grid>

<local:WavePage x:Name="WavePage" Visibility="Collapsed"/>

<local:StartPageControl x:Name="StartPage" <local:PlanetaryPage

Visibility="Collapsed" Grid.ColumnSpan="3" Grid.RowSpan="3" />

</Grid>

/>

x:Name="PlanetaryPage"

</TabItem.Content>

</TabItem>

В данном примере разметки, можно увидеть определение заголовка вкладки определенного в вложенном элементе TabItem.Header и самого содержимого окна, определенного в вложенном элементе TabItem.Content. Как можно увидеть из листинга, содержимое представляет из себя сразу три пользовательских элемента: WavePage, StartPageControl, PlanetaryPage. На самом деле в содержимом вкладки отображается только один элемент, а остальные два находятся в скрытом состоянии. Данный подход был применен с целью предоставления работы в рамках одного окна, без создания дополнительных, вспомогательных окон. Какой именно из трех элементов будет отображаться указывается в коде логики программы, на основании порядка пользовательского ввода. Например, изначально пользователь находится на начальном окне, но нажав на кнопку «Планетарный редуктор» окно заменится на форму PlanetaryPage. Также при нажатии кнопки «Назад», пользователь вернется на начальное окно, а PlanetaryPage перейдет в состояние collapsed, т.е. скроется. Рассмотрим каждый пользовательский элемент отдельно.

Элемент **StartPageControl** представляет собой стартовый интерфейс приложения (только у StartPageControl не выбрано состояние Visibility как collapsed). Верхняя часть окна состоит из заголовка и изображений. Заголовок представляет из себя TextBlock – элемент, предназначенный для отображения многострочного форматированного текста. Изображения представляют из себя элемент Image – предназначенный для отображения картинок различных

51

форматов. Нижняя часть представлена элементом Image в левой части и надписью с двумя кнопкам в правой. Надпись сделана на базе элемента label, предназначенного для отображения однострочного текста. Кнопки являются элементами Button, который может отлавливать различные события, такие как нажатие или наведения курсора мыши на кнопку. В зависимости от того, на какую кнопку наведен курсор, элемент Image в левой части меняет свое содержимое, т.е. меняется изображение кинематической схемы редуктора. Это достигается путем использования триггеров.

*Листинг 21 — Стиль, направленный на элементы Image без содержимого* <Style TargetType="Image">

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=PlanetaryBut, Path=IsMouseOver}" Value="True">

<Setter Property="Source" Value="Images/Planet.jpg"/> </DataTrigger>

<DataTrigger Binding="{Binding ElementName=PlanetaryBut, Path=IsMouseOver}" Value="False">

<Setter Property="Source" Value="Images/Planet.jpg"/> </DataTrigger>

<DataTrigger Binding="{Binding ElementName=WavesBut, Path=IsMouseOver}" Value="True">

<Setter Property="Source" Value="Images/Wave.jpg"/> </DataTrigger>

</Style.Triggers> </Style>

В листинге выше создается стиль, направленный на элементы Image без содержимого (на начало старта программы, для нашей картинки значение Source не установлено), содержащий триггеры, изменяющие данные (DataTrigger). Каждый триггер привязан к кнопкам, и отлавливает событие наведения курсора мыши на эти кнопки (устанавливается в переменной Path). При срабатывании триггера устанавливается содержимое Value из надлежащего блока Setter (содержимое представлено в виде относительного адреса расположения изображения в файловой системе). Нажатие на кнопку сопровождается событием, обрабатываемом в коде C#, например:

*Листинг 22 — Нажатие на кнопку*

private void WaveCallClick (Object sender,RoutedEventArgs e) {

Visibility = Visibility.Collapsed; mainWindow.WaveCall();

}

52

Здесь обрабатывается событие нажатия на кнопку перехода к окну ввода данных для расчета волнового редуктора. В первой строчке внутри тела функции текущее окно скрывается, а во второй вызывается функция развертки окна WavePage.

На листинге 23 представлено обобщение всего того, что было сказано выше, в виде полного программного кода, описывающего работу главной страницы приложения.

*Листинг 23 — Полный программный код* **using** System**;**

**using** System**.**Collections**.**Generic**; using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**; using** System**.**Windows**;**

**using** System**.**Windows**.**Controls**; using** System**.**Windows**.**Data**; using** System**.**Windows**.**Documents**; using** System**.**Windows**.**Input**; using** System**.**Windows**.**Media**;**

**using** System**.**Windows**.**Media**.**Imaging**; using** System**.**Windows**.**Navigation**; using** System**.**Windows**.**Shapes**;**

**namespace** Planetary\_REDUCT **{**

/// <summary>

/// Логика взаимодействия для StartPageControl.xaml /// </summary>

**public partial** class StartPageControl **:** UserControl **{**

**public delegate** void MainFunctions**();**

**public** MainWindow mainWindow **=** Application**.**Current**.**MainWindow **as** MainWindow**;**

**public** StartPageControl**() {**

InitializeComponent**();**

**}**

**private** void PlanetaryCallClick **(**Object sender**,** RoutedEventArgs e**) {**

Visibility **=** Visibility**.**Collapsed**;** mainWindow**.**PlanetaryCall**();**

53

**}**

**private** void WaveCallClick **(**Object sender**,**RoutedEventArgs e**) {**

Visibility **=** Visibility**.**Collapsed**;** mainWindow**.**WaveCall**();**

**}**

**} }**

Работа триггеров описывается в файле верстки главной страницы, так как отвечает за смену графического содержимого интерфейса. На листинге 24 приведен полный код верстки стартового интерфейса.

*Листинг 24 — Полный код верстки стартового интерфейса* <UserControl x:Class=**"Planetary\_REDUCT.StartPageControl"**

xmlns=**"http://schemas.microsoft.com/winfx/2006/xaml/presentation"** xmlns:x=**"http://schemas.microsoft.com/winfx/2006/xaml"** xmlns:mc=**"http://schemas.openxmlformats.org/markup-**

**compatibility/2006"** xmlns:d=**"http://schemas.microsoft.com/expression/blend/2008"** xmlns:local=**"clr-namespace:Planetary\_REDUCT"**

mc:Ignorable=**"d"**

d:DesignHeight=**"560"** d:DesignWidth=**"905"**>

<UserControl.Resources>

<Style TargetType=**"Image"**> <Style.Triggers>

<DataTrigger Binding=**"{Binding ElementName=PlanetaryBut, Path=IsMouseOver}"**

Value=**"True"**>

<Setter Property=**"Source"** Value=**"Images/Planet.jpg"**/> </DataTrigger>

<DataTrigger Binding=**"{Binding ElementName=PlanetaryBut, Path=IsMouseOver}"**

Value=**"False"**>

<Setter Property=**"Source"** Value=**"Images/Planet.jpg"**/> </DataTrigger>

<DataTrigger Binding=**"{Binding ElementName=WavesBut, Path=IsMouseOver}"**

Value=**"True"**>

<Setter Property=**"Source"** Value=**"Images/Wave.jpg"**/> </DataTrigger>

</Style.Triggers> </Style>

<Style TargetType=**"Button"**>

<Setter Property=**"FontFamily"** Value=**"Microsoft Sans Serif"**/> </Style>

</UserControl.Resources>

54

<Grid >

<Grid.RowDefinitions>

<RowDefinition Height=**"\*"**/>

<RowDefinition Height=**"1.6\*"**/>

</Grid.RowDefinitions> <Grid.ColumnDefinitions>

<ColumnDefinition Width=**"0.25\*"**/> <ColumnDefinition Width=**"0.25\*"**/> <ColumnDefinition Width=**"0.25\*"**/> <ColumnDefinition Width=**"0.25\*"**/>

</Grid.ColumnDefinitions>

<TextBlock FontSize=**"25"** Grid.Row=**"0"** Grid.Column=**"1"** VerticalAlignment=**"Center"** HorizontalAlignment=**"Center"** Grid.ColumnSpan=**"2"** TextAlignment=**"Center"** FontFamily=**"Times New Roman"** FontWeight=**"Bold"**>**Программа расчета**

<LineBreak></LineBreak>**планетарных и волновых редукторов** <LineBreak></LineBreak>**при габаритных ограничениях**</TextBlock> <Image Name=**"ImReductor"** Grid.Row=**"1"** Grid.Column=**"0"**

VerticalAlignment=**"Bottom"** Stretch=**"Fill"** Grid.ColumnSpan=**"2"**> </Image>

<Image Name=**"robot"** Grid.Row=**"0"** Grid.Column=**"0"** VerticalAlignment=**"Bottom"** Stretch=**"Fill"** HorizontalAlignment=**"Center"** Source=**"images/robot.png"**>

</Image>

<Image Name=**"shesterni"** Grid.Row=**"0"** Grid.Column=**"3"** VerticalAlignment=**"Bottom"** Stretch=**"Fill"** HorizontalAlignment=**"Center"** Source=**"images/shesterni.png"**>

</Image>

<Grid Grid.Row=**"1"** Grid.Column=**"2"** Grid.ColumnSpan=**"2"**> <Grid.RowDefinitions>

<RowDefinition Height=**"\*"** /> <RowDefinition Height=**"20"** /> <RowDefinition Height=**"30"**/> <RowDefinition Height=**"20"**/> <RowDefinition Height=**"30"**/> <RowDefinition Height=**"1.5\*"**/>

</Grid.RowDefinitions> <Grid.ColumnDefinitions>

<ColumnDefinition Width=**"\*"**/>

<ColumnDefinition

<ColumnDefinition

Width=**"auto"**/>

Width=**"\*"**/>

</Grid.ColumnDefinitions>

<TextBlock Text=**"Выберите тип редуктора"** FontSize=**"14"** Grid.Row=**"0"** Grid.Column=**"0"** Grid.ColumnSpan=**"3"** VerticalAlignment=**"Bottom"**

55

HorizontalAlignment=**"Center"** FontWeight=**"Bold"** FontFamily=**"Microsoft Sans Serif"**></TextBlock>

<Button x:Name=**"PlanetaryBut"** Grid.Row=**"2"** Width=**"auto"** Height=**"30"** Grid.Column=**"1"** Content=**"Планетарный редуктор"** HorizontalAlignment=**"Center"** Click=**"PlanetaryCallClick"** ></Button>

<Button x:Name=**"WavesBut"** Grid.Row=**"4"** Width=**"auto"** Height=**"30"** Grid.Column=**"1"** Content=**"Волновой редуктор"** Click=**"WaveCallClick"**></Button>

</Grid>

</Grid> </UserControl>

Окно **WavePage** является формой ввода исходных данных для расчета волнового редуктора. Все данные в данном окне вводятся с помощью элементов TextBox (кроме диапазона модулей), который принимает пользовательский ввод с клавиатуры. Для получения данных пользовательского ввода используются двусторонние связи свойств с интерфейсом. Например:

*Листинг 25 — Двусторонние связи свойств с интерфейсом*

<TextBox Grid.Column="4" Grid.Row="1" Text="{Binding Path=INk,Mode=TwoWay}"></TextBox>

Параметр Path определяет свойство, к которому привязывается данный интерфейс, а параметр Mode, тип их взаимодействия. В данном случае конструкция Mode=TwoWay означает, что изменения в привязанном свойстве будут влиять на интерфейс, а изменения в интерфейсе менять свойства объекта класса. Это полезно как при изменении всех полей ввода разом (очистка, пример исходных данных), так и при последовательном вводе. Представление свойства в коде:

*Листинг 26 — Представление свойства в коде*

public double INk { get { return mk; } set { mk = (double)value; OnPropertyChanged("INk"); } }

Функция OnPropertyChanged() сообщает интерфейсу об изменении свойства.

Даже после ввода данных пользователь с помощью элемента Button, может вставить готовый пример данных для расчета, очистить введенные ранее данные, перейти к расчету редуктора или вернуться на главную страницу.

Окно меню ввода данных для волнового редуктора имеет особенность, отличающую ее от меню для планетарного редуктора. Так как выбор диапазона модулей колес находится, согласно алгоритму, в заданных числовых пределах,

56

то было принято решение сделать задание диапазона в виде двух чисел, которые выбираются с помощью стороннего пользовательского элемента интерфейса. Сторонний элемент представляет собой слайдер - графическое изображение отрезка с двумя точками-ползунками. С помощью мыши по отрезку можно перемещать ползунки по отрезку. Левый ползунок устанавливает начальное значение диапазона, а правый - конечное. Так же, как и обычное текстовое поле слайдер диапазона передает значения на ползунках в класс расчета. Программно слайдер был реализован на базе двух стандартных одноползунковых слайдеров фреймворка, которые накладываются друг на друга внутри таблицы (стиль контроллера реализован в файле RangeSlider.xaml). В файле RangeSlider.xaml.cs описывается логика, согласно которой, ползунок левого слайдера не может пересечь ползунок правого и наоборот. На листингах 27 представлены программные коды, описывающие класс слайдеров с логикой работы и разметку наложения двух одноползунковых слайдеров друг на друга с защитой перемещения ползунков для достижения корректности задания диапазона.

*Листинг 27 — Описание класса слайдеров с логикой работы и разметки наложения* **using** System**;**

**using** System**.**Collections**.**Generic**; using** System**.**Linq**;**

**using** System**.**Text**; using** System**.**Windows**;**

**using** System**.**Windows**.**Controls**; using** System**.**Windows**.**Data**; using** System**.**Windows**.**Documents**; using** System**.**Windows**.**Input**; using** System**.**Windows**.**Media**;**

**using** System**.**Windows**.**Media**.**Imaging**; using** System**.**Windows**.**Navigation**; using** System**.**Windows**.**Shapes**;**

**namespace** UserControlFun **{**

/// <summary>

/// Interaction logic for RangeSlider.xaml /// </summary>

**public partial** class RangeSlider **:** UserControl **{**

**public** RangeSlider**() {**

InitializeComponent**();**

**this.**Loaded **+=** Slider\_Loaded**; }**

57

void Slider\_Loaded**(object** sender**,** RoutedEventArgs e**) {**

LowerSlider**.**ValueChanged **+=** LowerSlider\_ValueChanged**;** UpperSlider**.**ValueChanged **+=** UpperSlider\_ValueChanged**;**

**}**

**private** void LowerSlider\_ValueChanged**(object** sender**,** RoutedPropertyChangedEventArgs**<**double**>** e**)**

**{**

UpperSlider**.**Value **=** Math**.**Max**(**UpperSlider**.**Value**,** LowerSlider**.**Value**); }**

**private** void UpperSlider\_ValueChanged**(object** sender**,** RoutedPropertyChangedEventArgs**<**double**>** e**)**

**{**

LowerSlider**.**Value **=** Math**.**Min**(**UpperSlider**.**Value**,** LowerSlider**.**Value**); }**

**public** double Minimum **{**

**get { return (**double**)**GetValue**(**MinimumProperty**); } set {** SetValue**(**MinimumProperty**, value); }**

**}**

**public** static **readonly** DependencyProperty MinimumProperty **=** DependencyProperty**.**Register**(**"Minimum"**, typeof(**double**),**

**typeof(**RangeSlider**), new** UIPropertyMetadata**(**0d**));**

**public** double LowerValue **{**

**get { return (**double**)**GetValue**(**LowerValueProperty**); } set {** SetValue**(**LowerValueProperty**, value); }**

**}**

**public** static **readonly** DependencyProperty LowerValueProperty **=** DependencyProperty**.**Register**(**"LowerValue"**, typeof(**double**),**

**typeof(**RangeSlider**), new** UIPropertyMetadata**(**0d**));**

**public** double UpperValue **{**

**get { return (**double**)**GetValue**(**UpperValueProperty**); } set {** SetValue**(**UpperValueProperty**, value); }**

**}**

**public** static **readonly** DependencyProperty UpperValueProperty **=** DependencyProperty**.**Register**(**"UpperValue"**, typeof(**double**),**

**typeof(**RangeSlider**), new** UIPropertyMetadata**(**0d**));**

**public** double Maximum **{**

**get { return (**double**)**GetValue**(**MaximumProperty**); } set {** SetValue**(**MaximumProperty**, value); }**

**}**

58

|  |
| --- |
| **public** static **readonly** DependencyProperty MaximumProperty **=** DependencyProperty**.**Register**(**"Maximum"**, typeof(**double**),**  **typeof(**RangeSlider**), new** UIPropertyMetadata**(**1d**)); }**  **}** |
| <UserControl x:Class=**"UserControlFun.RangeSlider"** xmlns=**"http://schemas.microsoft.com/winfx/2006/xaml/presentation"** xmlns:x=**"http://schemas.microsoft.com/winfx/2006/xaml"** x:Name=**"root"**>  <UserControl.Resources>  <ControlTemplate x:Key=**"simpleSlider"** TargetType=**"{x:Type Slider}"**> <Border SnapsToDevicePixels=**"true"** BorderBrush=**"{TemplateBinding BorderBrush}"** BorderThickness=**"{TemplateBinding BorderThickness}"**>  <Grid> <Grid.RowDefinitions>  <RowDefinition Height=**"Auto"**/>  <RowDefinition Height=**"Auto"** MinHeight=**"{TemplateBinding MinHeight}"**/>  <RowDefinition Height=**"Auto"**/> </Grid.RowDefinitions>  <Rectangle x:Name=**"PART\_SelectionRange"**/>  <Track x:Name=**"PART\_Track"** Grid.Row=**"1"**> <Track.Thumb>  <Thumb x:Name=**"Thumb"**> <Thumb.Template>  <ControlTemplate TargetType=**"Thumb"**> <Rectangle Fill=**"White"**  Stroke=**"Gray"** StrokeThickness=**"1"** Width=**"10"** Height=**"18"**  SnapsToDevicePixels=**"{TemplateBinding SnapsToDevicePixels}"**/>  </ControlTemplate> </Thumb.Template>  </Thumb> </Track.Thumb>  </Track> </Grid>  </Border> </ControlTemplate>  </UserControl.Resources> <Grid VerticalAlignment=**"Top"**>  <Border BorderThickness=**"0,1,0,0"** BorderBrush=**"Black"** VerticalAlignment=**"Center"** Height=**"1"** |

59

Margin=**"5,0,5,0"**/> <Slider x:Name=**"LowerSlider"**

Minimum=**"{Binding ElementName=root, Path=Minimum}"** Maximum=**"{Binding ElementName=root, Path=Maximum}"** Value=**"{Binding ElementName=root, Path=LowerValue,**

**Mode=TwoWay}"**

Template=**"{StaticResource simpleSlider}"** Margin=**"0,0,10,0"** IsSnapToTickEnabled=**"True"** TickFrequency=**"0.1"**

/>

<Slider x:Name=**"UpperSlider"**

Minimum=**"{Binding ElementName=root, Path=Minimum}"** Maximum=**"{Binding ElementName=root, Path=Maximum}"** Value=**"{Binding ElementName=root, Path=UpperValue,**

**Mode=TwoWay}"**

Template=**"{StaticResource simpleSlider}"** Margin=**"10,0,0,0"** IsSnapToTickEnabled=**"True"** TickFrequency=**"0.1"**

/> </Grid>

</UserControl>

В программном коде класс панели меню ввода исходных параметров для расчета волнового редуктора WavePage описывается так, как на листинге 28.

*Листинг 28 — Описание класса панели меню ввода исходных параметров* **using** System**;**

**using** System**.**IO**;**

**using** System**.**Collections**.**Generic**; using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**; using** System**.**Windows**;**

**using** System**.**Windows**.**Controls**; using** System**.**Windows**.**Data**; using** System**.**Windows**.**Documents**; using** System**.**Windows**.**Input**; using** System**.**Windows**.**Media**;**

**using** System**.**Windows**.**Media**.**Imaging**; using** System**.**Windows**.**Navigation**; using** System**.**Windows**.**Shapes**;**

**using** UserControlFun**;**

**namespace** Planetary\_REDUCT **{**

/// <summary>

/// Логика взаимодействия для WavePage.xaml /// </summary>

**public partial** class WavePage **:** UserControl **{**

60

**public** Wave wave**;**

**public** MainWindow mainWindow **=** Application**.**Current**.**MainWindow **as** MainWindow**;**

**public** WavePage**() {**

InitializeComponent**();** wave **= new** Wave **{ }; this.**DataContext **=** wave**;**

**}**

**private** void CalculatingClick**(**Object sender**,** RoutedEventArgs e**) {**

**if (**wave**.**Cz **>** 1 **||** wave**.**Cz **<** 0.1 **||** wave**.**Tout **>** 500 **||** wave**.**Tout **<** 30 **||** wave**.**Nout **>** 400

**||** wave**.**mo **<** 0.1 **||** wave**.**mo **>** 1 **||** wave**.**mk **>** 1 **||** wave**.**mk **<** 0.1 **||** wave**.**Dr **>** 1000 **||** wave**.**Dr **<** 20**)**

**{**

MessageBox**.**Show**(**"Данные введены неверно."**); return;**

**}** wave**.**Construction**();**

InputGrid**.**Visibility **=** Visibility**.**Collapsed**;**

OutScreenPage outScreenPage **= (**OutScreenPage**)**OutputGrid**.**Children**[**0**];** outScreenPage**.**LoadWaveData**(**wave**);**

OutputGrid**.**Visibility **=** Visibility**.**Visible**;** wave**.**ResetData**();**

//wave = new Wave { }; //this.DataContext = wave;

// MessageBox.Show("Zf =" +wave.Zf.ToString() + "Zc = " + wave.Zc.ToString() + "Ngp = " +wave.Ngp.ToString() + "Modul = " + wave.modulfc);

**}**

void StartPageClick**(**Object sender**,** EventArgs e**) {**

Visibility **=** Visibility**.**Collapsed**;** mainWindow**.**StartPageCall**();**

**}**

**private** void ExampleClick**(object** sender**,** RoutedEventArgs e**) {**

wave**.**SetExample**(); }**

**private** void ClearClick**(object** sender**,** RoutedEventArgs e**) {**

ClearFields**(); }**

**private** void textBox\_PreviewTextInput**(object** sender**,** TextCompositionEventArgs e**)**

**{**

**if (!(**Char**.**IsDigit**(**e**.**Text**,** 0**) || (**e**.**Text **==** "."**) && (!(((**TextBox**)**sender**).**Text**.**Contains**(**"."**)) &&** e**.**Text**.**Length **!=** 0**)))**

**{**

e**.**Handled **= true;**

61

**} }**

**private** void textBox\_PreviewKeyDown**(object** sender**,** KeyEventArgs e**) {**

**if (**e**.**Key **==** Key**.**Space**) {**

e**.**Handled **= true; }**

**}**

**public** void ClearFields**() {**

wave**.**ClearInput**(); ((**RangeSlider**)**InputGrid**.**Children**[**6**]).**LowerValue **=** 0**; ((**RangeSlider**)**InputGrid**.**Children**[**6**]).**UpperValue **=** 0**;**

**for (**int i **=** 9**;** i **<=** 13**;** i**++) ((**TextBox**)**InputGrid**.**Children**[**i**]).**Text **=** ""**;**

**} }**

**}**

Окно **PlanetaryPage** аналогично предыдущему, за исключением того, что для параметра износостойкости был добавлен элемент CheckBox, который по своей сути является отображением булевой переменной. На листинге 29 приведен программный код методов, которые обрабатывают ситуации, когда галочка в меню ввода данных установлена или снята. Результатом обработки является инициализация булевой переменной LTR класса Planet планетарных редукторов.

*Листинг 29 — Методы, которые обрабатывают ситуации* **private** void ltr\_Checked**(object** sender**,** RoutedEventArgs e**)**

**{**

planet**.**LTR **= true; }**

**private** void ltr\_Unchecked**(object** sender**,** RoutedEventArgs e**) {**

planet**.**LTR **= false; }**

При нажатии кнопки «Расчет» мы переходим к окну **OutScreenPage**. Это окно предоставляет результат обработки введённых пользователем данных. Данные отображаются в элементе DataGrid, который по виду своего отображения на экране является таблицей. Большинство ячеек таблицы привязаны к локальной базе данных для заполнения статических данных, таких

62

как заголовки колонок и строк, отображающие текстовую информацию: название параметра, единицы измерения и т.д. Остальные ячейки, являются непосредственно результатами расчета, и их заполнение прописано в коде логики приложения (заполнение ячеек происходит в ходе выполнения программы). Для этого в коде используется класс DataTable в совокупности с DataRow и DataColumn.

Рассмотрим логику работы класса «OutScreenPage», описание которого находится в файле OutScreenPage.xaml.cs. На листинге 30 представлен программный код описания класса OutScreenPage.

*Листинг 30 — Описание класса OutScreenPage* **using** System**;**

**using** System**.**Collections**.**Generic**; using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**; using** System**.**Windows**;**

**using** System**.**Windows**.**Controls**; using** System**.**Windows**.**Data**; using** System**.**Windows**.**Documents**; using** System**.**Windows**.**Input**; using** System**.**Windows**.**Media**;**

**using** System**.**Windows**.**Media**.**Imaging**; using** System**.**Windows**.**Navigation**; using** System**.**Windows**.**Shapes**;**

**using** System**.**Globalization**; using** System**.**Data**;**

**public partial** class OutScreenPage **:** UserControl **{**

**public** DataTable data**;**

**public** OutScreenPage**() {**

InitializeComponent**();**

data **=** GetStartEmptyTable**();**

ResultGrid**.**ItemsSource **=** data**.**DefaultView**; }**

**private** DataTable GetStartEmptyTable**() {**

DataTable data **= new** DataTable**();**

DataColumn param **=** **new** DataColumn**(**"Вычисляемый параметр"**,typeof(**string**));**

DataColumn designation **=** **new** DataColumn**(**"Обозначение"**,typeof(**string**));**

DataColumn ed **= new** DataColumn**(**"Единицы измерения"**,typeof(**string**));**

63

DataColumn **value=new** DataColumn**(**"Рассчитанное значение"**,typeof(**string**));**

data**.**Columns**.**Add**(**param**);** data**.**Columns**.**Add**(**designation**);** data**.**Columns**.**Add**(**ed**);** data**.**Columns**.**Add**(value); return** data**;**

**}**

**public** void SetTable **(**DataTable table**) {**

data **=** table**;**

ResultGrid**.**ItemsSource **=** data**.**DefaultView**; }**

**public** void LoadWaveData**(**Wave wave**) {**

**for (**int i **=** 0**;** i **<** wave**.**NameParams**.**Count**;** i**++) {**

DataRow dataRow **=** data**.**NewRow**();** dataRow**[**0**] =** wave**.**Result**[**i**,** 0**];** dataRow**[**3**] =** wave**.**Result**[**i**,** 3**];** dataRow**[**1**] =** wave**.**Result**[**i**,** 1**];** dataRow**[**2**] =** wave**.**Result**[**i**,** 2**];**

// dataRow[1] = wave.Test[i]; data**.**Rows**.**Add**(**dataRow**);**

**} }**

**private** void SaveResult**(object** sender**,** RoutedEventArgs e**) {**

//SaveClick

**}**

**private** void ReturnClick**(object** sender**,** RoutedEventArgs e**) {**

Grid w **=(**Grid**) this.**Parent**;** w**.**Visibility **=** Visibility**.**Collapsed**;** Grid s **= (**Grid**)**w**.**Parent**;**

s**.**Children**[**0**].**Visibility **=** Visibility**.**Visible**;** data**.**Clear**();**

**} }**

В приведенном программном коде описания класса можно выделить конструктор объекта класса OutScreenPage(). У класса только одно поле – объект класса DataTable со свободным доступом и представлением на экране в

64

виде таблицы. Также имеются методы загрузки данных на вывод, обработки нажатий кнопок «Вернуться к расчету» и «Сохранить».

Следует отметить, что в каждом файле, содержащем описание классов, присутствуют подключенные пространства имен. На листинге 31 приведен список этих пространств.

*Листинг 31 — Список пространств* **using** System**;**

**using** System**.**Collections**.**Generic**; using** System**.**Linq**;**

**using** System**.**Text**; using** System**.**Windows**;**

**using** System**.**Windows**.**Controls**; using** System**.**Windows**.**Data**; using** System**.**Windows**.**Documents**; using** System**.**Windows**.**Input**; using** System**.**Windows**.**Media**;**

**using** System**.**Windows**.**Media**.**Imaging**; using** System**.**Windows**.**Navigation**; using** System**.**Windows**.**Shapes**;**

Рассмотрим каждое пространство имен и работу его классов в рамках приложения.

System – пространство имен, содержащее базовые классы с их методами. В разряд базовых включены так же и классы платформы .NET, на базе которой реализован фреймворк WPF.

System.IO - содержит типы, позволяющие осуществлять чтение и запись в файлы и потоки данных, а также типы для базовой поддержки файлов и папок.

System.Collections.Generic – пространство имен универсальных строго типизированных коллекций, которые обеспечивают большую безопасность и производительность по сравнению с неуниверсальными коллекциями. В данной программе применен класс List из этого пространства имен, который реализует список однотипных элементов.

System.Linq – пространство имен, содержащее классы, которые реализуют стандартные операторы запросов, последовательности и методы их сортировки.

System.Text – пространство имен классов, которые представляют кодировки ASCII и Юникода, абстрактных базовых классов для преобразования блоков знаков в блоки байтов и обратно; вспомогательного класса, который обрабатывает и форматирует объекты String, не создавая промежуточные экземпляры String.

System.Windows – пространство имен важных базовых классов элементов Windows Presentation Foundation (WPF) и других типов, широко применяемых в

65

ядре и инфраструктуре WPF. Одним из таких классов является непосредственно класс приложения Application. Так же пространство имен содержит классы, управляющие данными в приложении и реакцией приложения на события.

System.Windows.Controls - предоставляет классы для создания элементов, называемых элементами управления, которые позволяют пользователю взаимодействовать с приложением. Классы элементов управления являются основой пользовательского взаимодействия с приложением, так как они позволяют пользователю просматривать, выбирать или вводить данные или другую информацию. Примером классов этого пространства, применяемых в данной программе, являются TextBox, CheckBox, Button, ComboBox, Control, DataGrid и другие.

System.Windows.Data - обеспечивает доступ к классам, представляющим архитектуру ADO.NET. Архитектура ADO.NET позволяет создавать компоненты, эффективно управляющие данными из нескольких источников данных. Из данного пространства в программе используются классы DataColumn (схема столбца данных в DataTable), DataRow (схема строки данных в DataTable), DataTable (таблица данных в памяти).

System.Windows.Documents - cодержит типы, поддерживающие FixedDocument, FlowDocument и создание документов XPS. На базе классов данного пространства в программе реализованы вкладки на главной странице.

System.Windows.Input - предоставляет типы для поддержки системы ввода Windows Presentation Foundation (WPF). Сюда входят классы абстрагирования устройств для устройств мыши, клавиатуры и пера, часто используемые классы диспетчера ввода, поддержка для команд и пользовательских команд, а также различные служебные классы. На базе классов пространства реализована работа устройств ввода данных в приложение – мыши и клавиатуры, а также управления работой программы с помощью этих устройств.

System.Windows.Media - предоставляет типы, обеспечивающие интеграцию мультимедийных данных (изображения, текст, аудио и видео) в приложения.

System.Windows. Media.Imaging - предоставляет типы, используемые для кодирования и декодирования растровых изображений.

System.Windows.Navigation - предоставляет типы для поддержки навигации, т.е. перемещения между окнами и по журналам.

66

System.Windows.Shapes - предоставляет доступ к библиотеке фигур, которые могут использоваться в XAML или в коде.

System.Windows.Globalization - содержит классы, определяющие сведения, относящиеся к культуре, такие как язык, название страны, используемые календари, шаблоны форматирования дат, денежных сумм и чисел, а также порядок сортировки строк. Эти классы применяются при создании многоязыковых приложений. Такие классы, как StringInfo и TextInfo, обладают расширенными возможностями глобализации, включая поддержку заменителей и обработку текстовых элементов.

Как было сказано выше, большинство ячеек таблицы вывода привязаны к локальной базе данных для заполнения статических данных, таких как заголовки колонок и строк, отображающие текстовую информацию: название параметра, единицы измерения и т.д.

Как было сказано выше, ячейки таблицы вывода результатов, содержащие статические данные (текстовая информация заголовков строк и столбцов) привязаны к локальной базе данных. Это говорит о том, что статические данные содержатся в этой базе.

Решение использования базы данных для хранения статических текстовых данных создает перспективы дальнейшего расширения функционала программного продукта, увеличивает гибкость и облегчает поддержку программного кода.

Для хранения статических данных была применена локальная база данных SQL-Lite, которая поддерживает полный набор команд языка SQL и находится в бесплатном общем доступе. База данных в проекте имеет название Planetarydb.db.

Создание базы данных и описание ее свойств осуществляется через интерфейс среды разработки MS Visual Studio 2019. При создании базы данных в файле App.config автоматически появится следующее ее описание (листинг 32).

*Листинг 32 — Создание базы данных* <system.data>

<DbProviderFactories>

<remove invariant=**"System.Data.SQLite.EF6"** />

<add name=**"SQLite Data Provider (Entity Framework 6)"** invariant=**"System.Data.SQLite.EF6"** description=**".NET Framework Data Provider for SQLite (Entity Framework 6)"** type=**"System.Data.SQLite.EF6.SQLiteProviderFactory, System.Data.SQLite.EF6"** />

<remove invariant=**"System.Data.SQLite"** /><add name=**"SQLite Data Provider"** invariant=**"System.Data.SQLite"** description=**".NET Framework Data Provider for SQLite"** type=**"System.Data.SQLite.SQLiteFactory, System.Data.SQLite"**/>

67

</DbProviderFactories> </system.data>

Так же при создании базы следует файл, содержащий класс DBHelper, предназаченный для управления БД. Класс содержит методы: конструктор, метод обновления базы данных, метод создания таблицы в базе данных, метод дополнения данных.

На листинге 33 представлен программный код описания класса управления базой данных DBHelper. Программное описание данного класса находится в файле DBHelper.cs.

*Листинг 33 — Описание класса управления базой данных DBHelper* **using** System**;**

**using** System**.**Collections**.**Generic**; using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**; using** System**.**Data**.**SqlClient**; using** System**.**Data**;**

**using** System**.**Data**.**SQLite**; namespace** Planetary\_REDUCT **{**

class DBHelper **{**

**private** SQLiteConnection sqlConnection**; private** SQLiteCommand command**;**

**public** DataTable Table**;**

**private** SQLiteDataAdapter adapter**;**

string squary **=** "Select \* from Planetary\_tab"**; public** DBHelper**()**

**{**

sqlConnection **= new** SQLiteConnection**(**"Data Source = planetarydb.db; Version=3;"**);**

sqlConnection**.**Open**();**

**}**

**public** void UpdateDB**() {**

SQLiteCommandBuilder comandbuilder **= new** SQLiteCommandBuilder**(**adapter**);**

adapter**.**Fill**(**Table**);**

// adapter.Update(Table); //some problems **}**

**public** DataTable CreateTable**() {**

Table **= new** DataTable**(**"Planetary\_tab"**);**

command **= new** SQLiteCommand**(**squary**,** sqlConnection**);** 68

command**.**ExecuteNonQuery**();**

adapter **= new** SQLiteDataAdapter**(**command**);** //MySqlCommandBuilder commandBuilder= new

MySqlCommandBuilder(adapter);

// adapter.Fill(Table); **return** Table**;**

**}**

**public** void InsertParams **(**List**<**string**> value) {**

**for (**int i **=** 0**;** i **< value.**Count**;** i**++) {**

//non insert

string quary **=** "Update Planetary\_tab Set ParamValue = ('" **+ value[**i**].**ToString**() +** "') where (id= " **+ (**i**+**1**).**ToString**() +** ")"**;**

command **= new** SQLiteCommand**(**quary**);** command**.**Connection **=** sqlConnection**;** //command.Prepare(); command**.**ExecuteNonQuery**();**

**} }**

**} }**

Рассмотрим пространства имен, подключаемые к описанию данного класса и не описанные выше.

System.Data.SQLClient – поставляет данные платформы .NET (фреймворк WPF) для SQL Server.

System.Data – обеспечивает доступ к классам, представляющим архитектуру ADO.NET. ADO.NET позволяет создавать компоненты, эффективно управляющие данными из нескольких источников данных. К этому же пространству относятся классы строк и столбцов таблицы вывода.

System.Data.SQLite – непосредственно пространство имен, содержащее класс базы данных и систему управления ею.

System.Threading.Tasks – предоставляет типы, которые упрощают работу по написанию параллельного и асинхронного кода. Основными такими типами являются: Task, представляющий асинхронную операцию, которую можно ожидать и отменить, и Task<TResult>, представляющий собой задачу, которая может вернуть значение. Класс TaskFactory предоставляет статические методы для создания задач, а класс TaskScheduler предоставляет инфраструктуру планирования потоков по умолчанию.

69

**9 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

На рисунке 14 представлен вид главного меню программы, которое открывается при запуске. В верхней части окна расположены вкладки, перейдя к которым пользователь может ознакомиться с документацией к программе. Документация содержит описание математических моделей проектировочных расчетов планетарных и волновых редукторов, базовые расчетные алгоритмы и их блок-схемы, программный код основных модулей информационной системы, а также сведения о разработчиках.

Рисунок 14 — Вид главного меню программы

На главной вкладке под названием программы расположено главное меню, в котором пользователь выбирает тип редуктора, геометро-кинематические параметры которого необходимо рассчитать. При наведении курсора на соответствующую кнопку слева от главного меню отображается кинематическая схема планетарного или волнового редуктора соответственно.

Рисунок 15 — Меню ввода данных 70

После выбора схемы редуктора происходит переход к меню ввода данных (Рисунок 15). В информационной системе предусмотрена защита от ввода некорректных значений. При введении некорректных значений исходных параметров для расчета или при незаполненности нескольких или всех полей ввода система сообщит об ошибке при попытке провести расчет. После ввода всех исходных данных расчет проводится после нажатия кнопки «Рассчитать». Также в меню ввода данных есть возможность автоматического заполнения полей значениями теоретического примера. Для автозаполнения необходимо нажать кнопку «Пример данных». Есть возможность очистки всех полей для проведения следующего расчета или при ошибке во введенных данных, для этого присутствует кнопка «Очистить». Для расчета при повышенных требованиях к износостойкости и выносливости механизма необходимо поставить маркер в виде галочки возле соответствующей надписи в меню ввода данных. Имеется возможность выхода из меню ввода данных на основную вкладку главного меню нажатием кнопки «Вернуться к выбору редуктора».

Рисунок 16 — Результаты расчета

Далее открывается окно вывода результатов. Результаты расчета представляются в табличной форме. При недостаточной величине окна программы для отображения таблицы целиком можно развернуть окно программы на весь экран или использовать прокрутку таблицы колесом мыши, полосой прокрутки на правом краю таблицы, либо с клавиатуры.

71

Рисунок 17 — Результаты расчета

После ознакомления с результатом расчета можно либо вернуться к меню ввода данных нажатием кнопки «Вернуться к расчетам», либо сохранить полученную таблицу в Exel. Для сохранения необходимо нажать кнопку «Сохранить», в открывшемся диалоговом окне выбрать место сохранения файла и дать Exel-файлу название.

Меню ввода исходных данных для расчета волнового редуктора имеет отличительную особенность: диапазон модулей вводится установкой ползунков на шкале, которая находится возле соответствующей надписи в меню.

72

Рисунок 18 — Меню ввода исходных данных

Рисунок 19 — Результаты расчета

73

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Дружинин Ю.А., Зубов В.А., В.Ю. Лавров. Проектирование механизмов приборов и вычислительных систем с применением ЭВМ: Учеб. пособие для студентов приборостр. и электротех. спец. вузов. – М.: Высш. шк., 1988. – 160 с.: ил. – (ЭВМ в техническом вузе).

2. Кудрявцев В.Н., Державец Ю.А., Глухарев Е.Г. Конструкции и расчет зубчатых редукторов. − М.: Машиностроение, 1977.

3. Истомин С.Н. Проектирование мелкомодульных передач приборов с применением ЭВМ. – М.: Машиностроение, 1985. – 175 с.: ил.

4. Попов С. А., Черная Л. А. Математическое и программное обеспечение расчетов кинематических характеристик: Учебное пособие. - М.: Изд. МГТУ, 1991.

5. Зиновьев В.А. Курс теории механизмов и машин. – М.: ФИЗМАТГИЗ, 1960. – 431 с.

6. Троелсен Э., Джепикс Ф. Язык программирования C# 7 и платформы .NET и .NET Core. 8-е изд.: Пер. с англ. – СПб.: ООО Диалектика, 2018. – 1328 с. ил.

7. Албахари Б., Албахари Д. C# 7.0 Справочник. Полое описание языка.: Пер. с англ. – СПб.: ООО «Альфа-книга», 2018 – 1024 с.

8. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C#. – Пер. с англ. – СПб.: Символ Плюс, 2011 – 768 с., ил.

9. WPF и C# полное руководство [электронный ресурс] //Metanit.com: Сайт о программировании, 2012-2020. URL: https://metanit.com/sharp/wpf/ (дата обращения 20.04.2020).

10. Microsoft Docs [электронный ресурс] // docs.microsoft.com: хранилище документации Майкрософт для пользователей, разработчиков и ИТ-специалистов, 2020. URL: https://docs.microsoft.com/ru-ru/ (дата обращения 20.05.2020).

74

СВЕДЕНИЯ ОБ АВТОРАХ

Пименов Алексей Владимирович, к.т.н., доцент кафедры промышленной информатики Института информационных технологий РТУ МИРЭА

Володина Анна Михайловна, старший преподаватель кафедры промышленной информатики Института информационных технологий РТУ МИРЭА

Чалков Андрей Александрович, студент 3 курса Института информационных технологий РТУ МИРЭА

Пищиков Николай Андреевич, студент 3 курса информационных технологий РТУ МИРЭА

Карп Георгий Вячеславович, студент 3 курса информационных технологий РТУ МИРЭА

Института

Института