Федеральное государственное автономное

образовательное учреждение

высшего профессионального образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Политехнический институт

**КУРСОВАЯ РАБОТА**

"Расчет конической передачи, автоматизированное построение и конечно-элементный анализ зубчатого зацепления"

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Колбасина Н. А.

подпись, дата

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Морозов Д. И.

подпись, дата

Студент МТ14-09Б \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Янчукович П.А.

подпись, дата

Красноярск 2016

**Задание на курсовую работу № 19**

**в разделе «конечно-элементный анализ»**

**"Расчет конической передачи, автоматизированное построение и конечно-элементный анализ зубчатого зацепления"**

### Содержание курсовой работы:

1. Формирование входного и выходного потоков данных, описание типов
2. Создание структуры класса и ее обоснование
3. Реализация интерфейса, обеспечение соответствия его стандартам Windows
4. Реализация защиты класса от ошибок ввода
5. Формализация задачи на уровне алгоритма
6. Структурирование алгоритма по подпрограммы
7. Разработка кода подпрограмм и оформление их как методов класса
8. Организация представления, хранения, печати протоколов работы приложения
9. Оформление пояснительной записки

### Содержание пояснительной записки:

1. Задание на проектирование
2. Содержание
3. Аннотация (цель, средства реализации, достигнутый результат)
4. Теоретический материал по типу передачи
5. Использование API Solidworks
6. Входной и выходной потоки данных, их описание
7. Алгоритм реализации поставленной задачи
8. Формирование из алгоритма библиотеки подпрограмм (обоснование принципов деления кода на подпрограммы)
9. Отладка приложения, обеспечение его устойчивости
10. Приложения (код подпрограмм и обработчиков событий с комментариями)
11. Список используемой литературы

**Оглавление**

[**Задание на курсовую работу № 19** 2](#_Toc452675508)

[**1 Аннотация** 4](#_Toc452675511)

[**2 Теоретический материал** 5](#_Toc452675512)

[**3 Использование API SolidWorks** 6](#_Toc452675513)

[**5 Алгоритм реализации поставленной задачи** 9](#_Toc452675519)

[**6 Структура модуля построения** 11](#_Toc452675520)

[**7 Формирование библиотеки подпрограмм** 11](#_Toc452675525)

[**8 Отладка приложения, обеспечение работоспособности** 1](#_Toc452675525)2

[**9 Моделирование передачи** 13](#_Toc452675526)

[**10Список литературы** 17](#_Toc452675527)

**1 Аннотация**

Целью данной курсовой работы является следующее: создание класса расчета конической прямозубой передачи, создание модуля автоматического построения колес передачи по данным, полученным при помощи класса расчета передачи, а также анализ контактной задачи пары колес данной передачи.

При выполнении поставленных задач использовались следующие ресурсы:

* Язык программирования – Delphi
* Среда разработки – Codegear Delphi 10 Seattle
* Среда проектирования – Solidworks
* Среда проведения анализа – Solidworks Simulation

По окончании курсовой работы планируется получить следующее:

* Программу расчета и построения конической передачи
* Смоделированную и собранную коническую передачу
* Проведенный анализ контактной задачи пары колес полученной передачи

Средства реализации:

* Функции APISolidWorks.
* СредаразработкиCodegear Delphi 10 Seattle.
* SolidSimulation.

В ходе выполнения курсовой работы были получены следующие результаты:

* Созданы модули автоматического построения колес конической передачи с помощью API SolidWorks
* Создан класс расчета конической передачи
* Создана тестирующая программа, использующая полученные модули и класс
* Создана 3D модель конической передачи в среде SolidWorks
* Проведен конечно-элементный анализ пары колес созданной передачи в среде SolidWorks Simulation

**2 Теоретический материал**

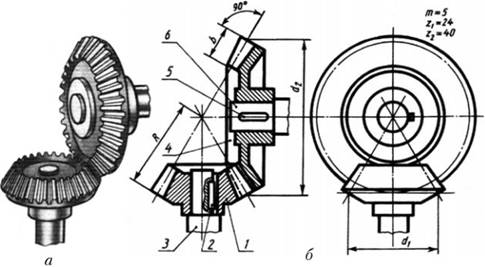


рисунок 2 – коническая передача

Конические зубчатые колеса применяют в передачах между валами, оси которых расположены под углом. Основное применение имеют передачи с пересекающимися под углом 90° осями, т. е. ортогональные передачи, которые рассматриваются ниже. Передачи с межосевым углом, не равным 90°, применяют редко из-за сложности форм и технологии изготовления корпусных деталей, несущих эти передачи, хотя для изготовления самих колес межосевой угол передачи не имеет значения. Пересечение осей валов затрудняет размещение опор. Одно из конических колес, как правило, располагают консольно. При этом увеличивается неравномерность распределения нагрузки по длине зуба. В коническом зацеплении действуют осевые силы, наличие которых усложняет конструкцию опор. Все это приводит к тому, что по опытным данным нагрузочная способность конической прямозубой передачи составляет лишь около 0,85 цилиндрической. Несмотря на отмеченные недостатки, а также то, что конические колеса сложнее, чем цилиндрические в изготовлении и монтаже, конические передачи имеют, широкое применение, поскольку по условиям компоновки механизмов довольно часто необходимо располагать валы под углом. Конические колеса выполняют с прямыми, тангенциальными, круговыми и другими криволинейными зубьями. Прямозубые конические колеса следует применять при невысоких окружных скоростях (до 2...3 м/с) как наиболее простые в монтаже (допустимо до 8 м/с). При более высоких скоростях целесообразно применять колеса с круговыми зубьями, как обеспечивающие более плавное зацепление, меньший шум, большую несущую способность и более технологичные. Зубья обрабатывают на специальных станках для нарезания конических колес. В массовом и крупносерийном производстве в связи с возможностью компенсации при нарезании зубьев последующих закалочных деформаций конические колеса не шлифуют, а ограничиваются притиркой. В конических передачах для обеспечения при сборке правильного контакта зубьев предусматривают возможность осевой регулировки зубчатых колес. Несущая способность конических зубчатых передач с повышенным перекосом осей (от консольного расположения, недостаточной жесткости валов и корпусов) может быть несколько повышена даже по сравнению с передачами, имеющими круговой зуб, выполнением зубьев двояковыпуклыми и вогнутыми. Обе стороны зуба шестерни нарезают выпуклыми, а колеса — вогнутыми..

**3 Использование API SolidWorks**

SolidWorks API (Application Programming Interface) — это интерфейс, позволяющий разрабатывать пользовательские приложения для системы SolidWorks. API – интерфейс содержит множество функций, которые можно вызывать из программ Microsoft Visual Basic, Microsoft Visual C++, Microsoft Visual Studio или из файлов-макросов SolidWorks. Эти функции предоставляют программисту прямой доступ к функциональным возможностям SolidWorks.

С помощью API – приложений можно решать множество различных задач, например такие как: интеграция SolidWorks с другими программными пакетами, разработка специализированных модулей, добавляющих к базовым возможностям SolidWorks дополнительную функциональность и различные другие задачи. API – приложения позволяют получить множество конфигураций одной детали или сборки, тем самым выиграть огромное количество времени при принятии конструкторских решений.

Разработка API — приложения может осуществляться на уровне создания макроса в SolidWorks, либо на уровне отдельного приложения, написанного на языке C# или VisualBasic. Все динамические библиотеки, необходимые для работы с API – приложениями автоматически инсталлируются вместе с SolidWorks. Как правило, если необходимо разработать полноценное приложение, для геометрических построений удобнее использовать программный код, записанный в макрос SolidWorks. Для начала работы с макросами, необходимо в программном пакете SolidWorks отобразить панель инструментов «Макрос».

### Схема

Application Programming Interface (API) SolidWorks – это интерфейс, позволяющий программировать приложения для системы SolidWorks. API интерфейс содержит сотни функций, которые можно вызывать из программ Microsoft Visual Basic, VBA (Microsoft Excel, Word, Access и т.д.), Microsoft Visual C, C++ или файлов-макросов SolidWorks. Эти функции предоставляют программисту прямой.

Программный интерфейс SolidWorks использует объектно-ориентированную модель. Все функции SolidWorks API – это методы или свойства, которые применяются к объектам SolidWorks. Диаграмма объектов SolidWorks представляет собой иерархическую структуру, которая отображает связи между объектами SolidWorks и показывает, в каким образом можно получить доступ к тому или иному объекту SolidWorks API.

Диаграммы объектов SolidWorks приведена на рис. 1.

Главным или корневым элементом дерева объектов является SLDWorks.Этот объект может быть получен при помощи метода CreateObject.

Set SldWorks = CreateObject("SldWorks.Application")

Метод CreateObject запускает программу SolidWorks, если она не была запущена, или создаёт объект SldWorks для последней активной сессии SolidWorks. Создав объект SldWorks, можно получить доступ ко всем объектам SolidWorks для данной сессии. Одним из ключевых объектов SolidWorks API является ModelDoc. Объект даёт программисту интерфейс к озданию и модификации 3D моделей и чертежей SolidWorks. Объект ModelDoc отвечает за функциональность деталей, сборок и чертежей.

Рис.1. Схема API программного комплекса Solidworks.

### Cоздание нового макроса и его выполнение

Последовательность действий для создания макроса в SolidWorks.

1. Перед тем, как начать записывать макрос, необходимо создать новый файл детали (сборки).

2. Для начала записи макроса на панеле «Макрос» кликнуть «Запись\пауза макроса», после чего каждое построение или изменение свойства будет записываться в файл макроса.

3. Далее проводим все построения, которые должны быть отображены в программном коде макроса.

4. После этого останавливаем запись, кликнув «Остановить запись макроса». Появится окно, в котором необходимо указать физический путь файла макроса на жестком диске и указать формат, в котором он будет записан.

### Форматы, в которых может быть сохранен макрос:

• VBA – упрощенная реализация языка программирования Visual Basic, встроенная в линейку продуктов Microsoft Office.

• VSTA VB – макрос будет сохранен в программном коде на языке Visual Basic. VSTA – инструментарий расширения функциональности приложений, основанный на .NET.

• VSTA C# — макрос будет сохранет в программном коде на языке C#.

Для того, чтобы выполнить записанный макрос необходимо на панеле «Макрос» кликнуть «Выполнить макрос», после чего появится окно «Открыть».

### Редактирование макроса

Для того, чтобы редактировать созданный макрос, необходимо на панеле «Макрос» кликнуть «Редактировать макрос», после чего в появившемся окне указать расположение файла макроса на жестком диске и его формат. Для редактирование макроса (VB, C#) используется набор инструментов Visual Studio for Applications (VSTA). VSTA автоматически запускается после выбора макроса для редактирования. В окне VSTA отображается сгенерированный код редактируемого макроса.

С помощью этого инструмента можно внести изменения в сгенерированный код и пересохранить макрос. Необходимо обратить внимание на то, что в окне Project Explorer уже указаны ссылки на все необходимые динамические библиотеки.

### Основные методы

Основные методы, изменение формальных параметров которых может повлиять на геометрию данной модели:

• Метод InsertSketch класса SketchManager служит для добавления эскиза в активной плоскости. В качестве формального параметра принимает значение типа bool. Прежде чем создавать новый эскиз следует выбрать плоскость, на которой он будет размещен.

boolstatus = swDoc.Extension.SelectByID2("Спереди", "PLANE", 0, 0, 0, false, 0, null, 0);

          swDoc.SketchManager.InsertSketch(true);

• Метод CreateLine класса SketchManager служит для создания новой линии эскиза в режиме редактирования эскиза. Принимает 6 формальных параметров – координаты конечной и начальной точки линии. Аналогично методу CreateLine существуют методы для создания других элементов эскиза. Пример использования метода CreateLine:

skSegment= ((SketchSegment) (swDoc.SketchManager.CreateLine (0.116812, 0.026916, 0.000000, 0.149183, 0.026916, 0.000000)));

• Метод EditDimensionProperties класса swDoc служит для редактирования свойств нанесенного размера. Принимает множество формальных параметров, каждый из которых отвечает за определенное состояния какого – либо свойства размера.

• Метод AddDimension класса swDoc служит для создания нового размера и в качестве формальных параметров принимает координаты граничных очек размера. Для корректного использования необходимо явное приведение к типу DisplayDimension.

Интерфейс созданного приложения представлен на рисунке 1. Входными параметрами являются число зубьев, модуль и коэффициент смещения.

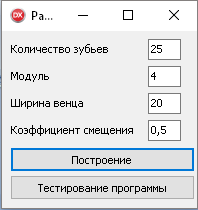


Рисунок 1 – интерфейс приложения

**4 Входной и выходной потоки данных**

При создании класса расчета конической передачи были сформированы входной и выходной поток данных Input и Output соответственно. Они представлены в таблице 1 и 2.На входе класса будут подаваться данные необходимые для расчета передачи, на выходе будут все полученные в ходе расчета результаты.

Таблица 1 – Входной поток данных

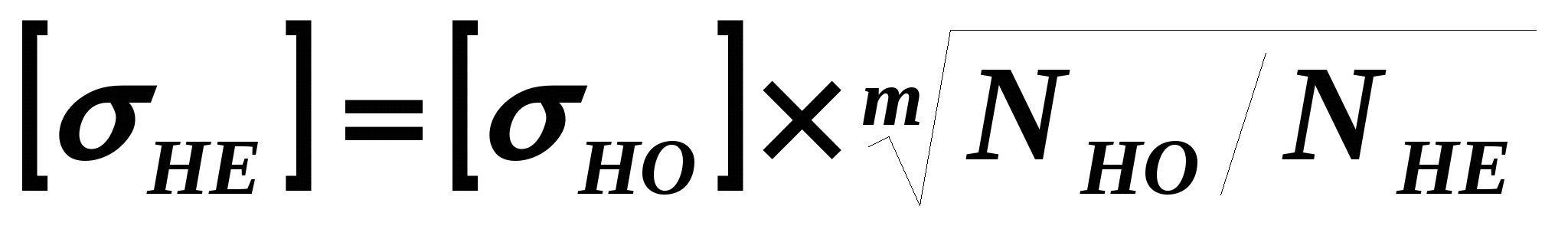
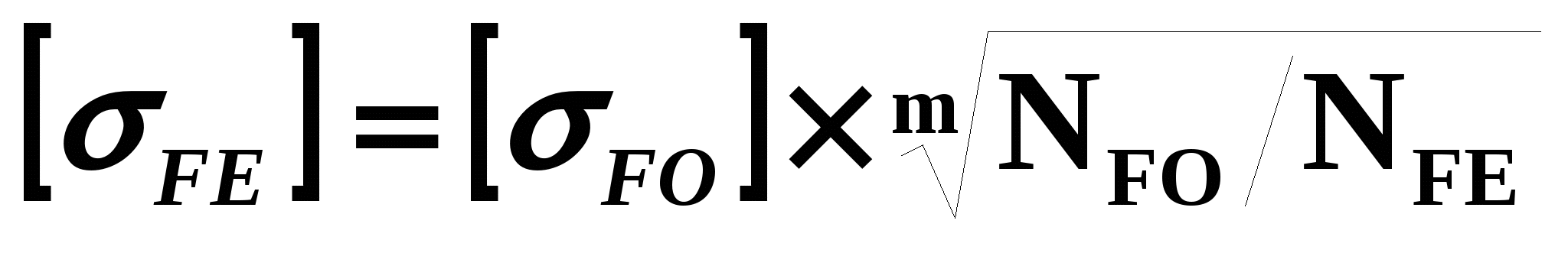
|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип |
| Передаваемая мощность | P1 | extended |
| Частота вращения | N1 | byte |
| Передаточное число | U | extended |
| Расчетный ресурс передачи | Lh | integer |
| Тип зубьев колес | Tipz | integer |
| Направление наклона зубьев | Napr\_Vr | integer |
| Марка стали для шестерни | Mc1 | integer |
| Марка стали для колеса | Mc2 | integer |
| Форма зуба | Forma | integer |
| Термообработка зуба шестерни | Termoobr1 | integer |
| Термообработка зуба колеса | Termoobr2 | string |
| Твердость сердцевины зуба шестерни | H\_Hvs1 | string |
| Твердость сердцевины зуба колеса | H\_Hvs2 | string |
| Предел текучести материала шестерни | Sigma\_t1 | integer |
| Предел текучести материала колеса | Sigma\_t2 | integer |
| Предельное напряжение для шестерни | Sigma\_Fst01 | integer |
| Предельное напряжение для колеса | Sigma\_Fst02 | integer |

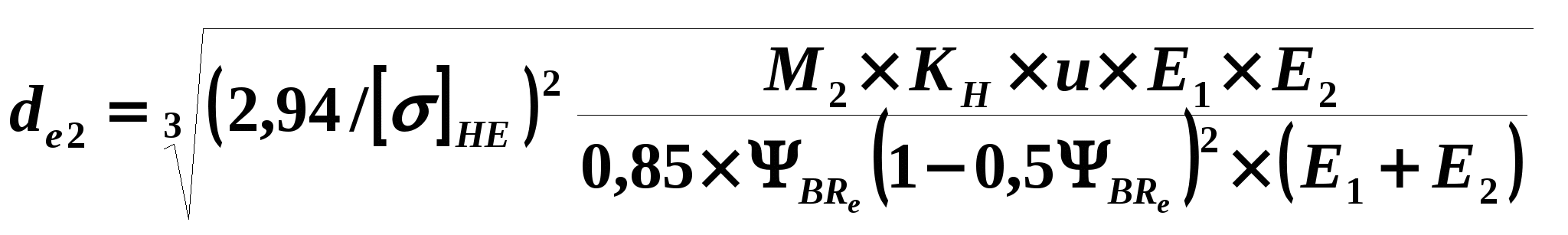
Таблица 2 – Выходной поток данных

|  |  |  |
| --- | --- | --- |
| Наименование | Обозначение | Тип |
| Суммарная масса зубчатых колес | Massa | string |
| Обьем занимаемый передачей | Massa1 | string |
| Модуль нормальный в среднем сечении | Re | string |
| Модуль торцевой на внешнем торце | Mte | string |
| Модуль торцевой на внешнем торце | Me | string |
| Нормальный угол наклона в среднем сечении в радианах | Bent | string |
| Коэффициенты тангенциального смещения | Xtau1, Xtau2 | integer |
| Коэффициент смещения исходного контура | x1,x2 | extended |
| Коэффициент смещения исходного контура  в нормальном сечении | Xn1,xn2 | extended |
| Число зубьев шестерни | z1 | extended |
| Число зубьев колеса | z2 | extended |
| Степень точности | St | extended |
| Фактическое передаточное число передачи | Uf | integer |
| Частота вращения колеса | N2 | integer |
| Скорость | V | integer |
| Диаметр делительного конуса шестерни | de1 | extended |
| Диаметр окружности вершин | Dae1 | extended |
| Диаметр окружности впадин шестерни | Dfe1 | extended |
| Средний делительный диаметр шестерни | D1 | extended |
| Угол делительного конуса шестерни | Delta1 | extended |
| Угол конуса вершин шестерни | Delta\_A1 | extended |
| Измерительная высота | Hce | extended |
| Диаметр делительного конуса колеса | De2 | extended |
| Угол делительного конуса колеса | Delta2 | extended |
| Внешняя постоянная хорда | Sce2 | extended |
| Измерительная высота | Hce2 | extended |
| Среднее конусное расстояние по делительному конусу | Rs | extended |
| Ширина венца | B | extended |
| Расстояние от вершины конуса до базовой поверхности | B1 | extended |
| Окружное усилие шестерни | Ft1 | extended |
| Радиальное усилие шестерни | Fr1 | extended |
| Осевое усилие шестерни | Fx1 | extended |
| Момент, передаваемый быстроходным валом | T1 | extended |
| Момент, передаваемый тихоходным валом | T2 | extended |

**5 Алгоритм реализации поставленной задачи**

В ходе исследования поставленной задачи был составлен следующий алгоритм расчета конической передачи:

1. Получение входных данных из модуля Input (P1,N1,U,Lh,Tipz, Napr\_Vr, Mc1,Mc2, Forma, Termoobr1, Termoobr2, H\_Hvs1, H\_Hvs2, Sigma\_t1, Sigma\_t2, Sigma\_Fst01, Sigma\_Fst02)
2. Выбор материала зубчатых колес. Рекомендуется выбирать материалы и термообработку таким образом, чтобы твердость поверхности зубьев шестерни была на ***20÷30 НВ*** выше, чем колеса.
3. Определение допускаемых напряжений. С учетом фактических условий нагружения: ; 
4. Определение чисел зубьев шестерни, колеса и передаточнго числа. Число зубьев шестерни по условию отсутствия подрезания для прямозубых колес должно быть ***Z1≥ 17 cos(90o- arctg ί12),*** где ***ί12*** - требуемое передаточное отношение передачи ( отношение угловых скоростей шестерни и колеса). Рекомендуется выбирать ***Z1= 18 ÷ 30***.
5. Из условия контактной прочности рабочих поверхностей зубьев:

,где ***M2***- вращающий момент на колесе ,***H/мм***; ***u***- передаточное число зубчатой передачи; ***[σ]HE***- наименьшее допускаемое контактное напряжение с учетом фактических условий нагружений; ***ψвRе= в/Re≤0,3*** - коэффициент ширины зубчатого венца, при выполнении

1. Расчёт геометрических параметров зубчатой передачи:

Внешний окружной модуль ***me =dе2 ⁄Z2***.

Угол делительного конуса колеса ***δ2= arctg u***.

Угол делительного конуса шестерни ***δ1= 90○ - δ2*.**

Внешний делительный диаметр шестерни ***dе1 = me1×Z1*.**

Внешнее конусное расстояние ***Re= 0,5 de2 ⁄ sin δ2***.

Ширина зубчатого венца ***b = ΨвRе × Re***.

Среднее конусное расстояние ***R = Re – 0,5 b***.

Средний окружной модуль ***m = meR ⁄ Re* .**

Средний делительный диаметр шестерни ***d1 = m ×Z1***.

Средний делительный диаметр колеса ***d2 = m × Z2***.

Внешняя высота зуба ***he = 2,2 ×me***.

Внешняя высота головки зуба ***hae= me***.

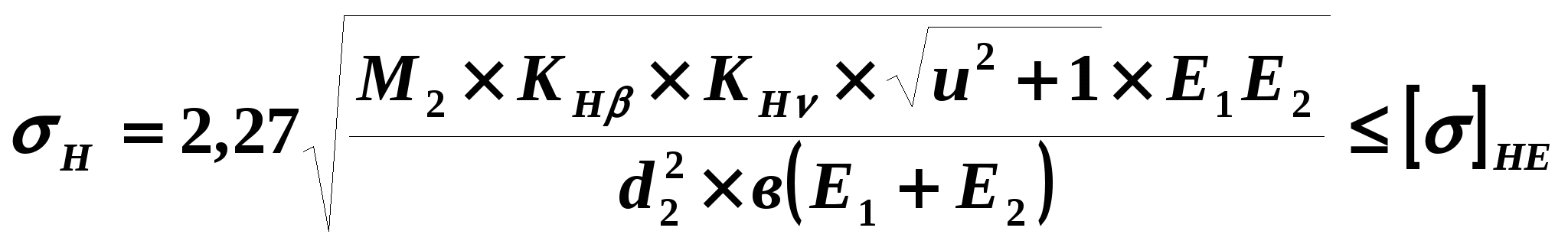
Внешняя высота ножки зуба ***hfe= 1,2 ×me***.

Угол ножки зуба ***θf = arc tg ( hfe ⁄ Re)***.

Угол головки зуба ***θd = θf***.

Внешний диаметр вершин зубьев шестерни ***dae1 = de1+ 2hae ×cosδ1***.

Внешний диаметр вершин зубьев колеса ***dae2 = dae2****+****2hae×cos δ2***.

1. Проверочный расчёт контактных напряжений на рабочих поверхностях зубьев ***Кнβ***- коэффициент концентрации нагрузки, учитывающий распределение нагрузки по ширине венца, определяется по таблице 3. ***Кнν***- коэффициент, учитывающий влияние динамической нагрузки, влияющей в зацеплении, определяется в зависимости от степени точности зубчатых колес, Если полученное в результате расчета фактическое контактное напряжение ***σн***не превышает допускаемое напряжение ***[σ]HE***более чем на ***5%***, т.е. ***ξ= σн-***[***σ***]***НЕ/***[***σ***]***HE× 100% ≤ 5%*,**то прочность зубчатой передачи по контактным напряжениям можно считать удовлетворительной. Если же расхождение ***ξ >5%***, то необходимо увеличить внешние делительные диаметры зубчатых колес ***de2*** и***de1***или подобрать для изготовления зубчатых колес материал, обеспечивающий более высокое значение ***[σ]HE***. В зависимости от принятого решения производятся вновь необходимые расчеты в соответствии с данной методикой.
2. Определение сил в коническом зацеплении.Окружные усилия на шестерне и колесе ***Ft1=Ft2=2M1/d1=2M2/d2****.* Радиальная сила на шестерне, равная осевой силе на колесе, ***Fr1=2M1×tgλ×cos b1/ d1***, где угол зацепления ***λ=200***. Осевая сила на шестерне, равная радиальной силе на колесе, ***Fa1=2M1×tgλ× sin b1/d1= Fr2***.
3. Проверочный расчет зубьев на выносливость по напряжениям изгиба, ***σF =***(***Ft×KFв×KFν×YF***)***/***(***0,85×b×m***)***≤***[***σ***]***FE,*** где***КFβ***- коэффициент концентрации нагрузки, ***KFν***– коэффициент динамичности, учитывающий динамическое действие нагрузки, ***YF***– коэффициент прочности зубьев по местным напряжениям, определяется по таблице П8 в зависимости от эквивалентного числа зубьев на колесе (шестерни)***Z ν= Z / cos b***. Расчет следует вести для зубьев того из колес, для которого отношение [***σ***]***FЕ/YF***меньше.
4. Вывод выходных данных Output(Massa, Massa1, Re, Mte, Me, Bent, Xtau1, Xtau2, x1, x2, Xn1, Xn2, z1, z2, St, Uf, N2, V, Del, Dae1, Dfe1, D1, Delta1, Delta\_A1, Hce, De2, Delta2, Sce2, Hce2, Rs, B, B1, Ft1, Fr1, Fx1, T1, T2)

**6 Структура модуля построения**

function formuls(m, x: extended; z: integer): функция вычисления эвольвенты и выкружки. Формальные параметры: m – модуль, x – коэффициент смещения, z – количество зубьев.

procedure DrawKol(m, x: extended; z: integer) процедура отрисовки конического колеса. Формальные параметры: m – модуль, x – коэффициент смещения, z – количество зубьев.

procedure TForm1.ApplicationEvents1Exception(Sender: TObject; E: Exception): процедура обработки ошибок. Формальные параметры: Е: тип ошибки.

procedure TForm1.Button2Click(Sender: TObject): процедура тестирования приложения.

**7 Формирование библиотеки подпрограмм**

На основании алгоритма, описанного выше была составлена следующая библиотека подпрограмм, встроенных в класс расчета:

1. Основной модуль расчета – kuurs. Внутри этой процедуры реализован весь расчет передачи
2. Модуль с константами материалов– materials
3. Вспомогательный модуль для задания входных параметров – Nom\_Sx
4. Модуль проверки вводимых параметров - Proverk

**8** **Отладка приложения, обеспечение его устойчивости**

Тестирование модуля построения происходило в диапазоне z(количество зубьев) 18 – 30 и modulmas(допустимы модули), была написана процедура тестирования:

procedure TForm1.Button2Click(Sender: TObject);

var

z: integer; // количество зубьев

m, x: extended; // m: модуль, x: коэффициент смещения

begin

x := 0; //коэффициент смещения не учитываем

for z := 18 to 30 do //запускаем цикл по количеству зубьев от 18 до 30

begin

for I := 1 to length(modulmas) do // запускаем следующий цикл по модулям

begin

m := modulmas[I]; // присваиваем переменной m(модуль) значение [I] стандартного модуля

b := 6 \* m; // ширину венца b делаем равной b:= 6\*m

ma := formuls(m, x, z); //заполняем массив точек по расчету эвольвенты

SetLength(qwerty, length(ma) \* 3); //задание массива для сплайна

for I := 0 to length(ma) - 1 do // перевод из двумерного массива координат эвольвенты в одномерный массив, пригодный для создания сплайна

begin

qwerty[I \* 3] := ma[I, 0];

qwerty[I \* 3 + 1] := ma[I, 1];

qwerty[I \* 3 + 2] := 0;

end;

A1 := VarArrayCreate([0, length(qwerty) - 1], varDouble); // обьявляем этот массив типа Double

for I := 0 to length(qwerty) - 1 do //заносим в массив А1 точки сплайна

begin

A1[I] := qwerty[I] / 1000;

end;

DrawKol(m, x, z); //вызываем процедуру построения колеса

end;

end;

end;

Обработка ошибок происходит в процедуре TForm1.ApplicationEvents1Exception:

procedure TForm1.ApplicationEvents1Exception(Sender: TObject; E: Exception);

var

LogFile: TextFile; // текстовый лог-файл

FileName: string; // путь и имялог-файла

begin

FileName := ChangeFileExt(Application.ExeName, '.txt'); // Имя лог-файла сделать таким же как у приложения, но с расширением txt

AssignFile(LogFile, FileName); // если файл существует перезаписать, иначе создать

if FileExists(FileName) then

Append(LogFile) // открыть существующий файл

else

Rewrite(LogFile); // создать новый файл

try

Writeln(LogFile, DateTimeToStr(Now) + ' ' + E.Message); // записать дату+время и текст ошибки в лог-файл

Application.ShowException(E); // показать ошибку

finally

CloseFile(LogFile);// закрыть файл

end;

end;

Запись log происходит с именем приложения в папку debug, в файле log мы увидим дату и время происхождения ошибки и код самой ошибки.

**9 Моделирование передачи**

Для построения модели конического колеса были созданы массивы Lines(массив линий) и Points(массив точек):

Points: array [0 .. 40] of ISketchPoint;

Lines: array [0 .. 30] of ISketchSegment;

Заполнение массива происходит с помощью линий по координатам, координаты рассчитываем с помощью переменных: dv – диаметр вала, d1 – диаметр шестерни, d2 – диаметр колеса, b – ширина венца, ha – высота зуба. После построения линий, мы присваивали каждой линии начальные и конечные точки из массива Points:

Lines[0] := MD.ICreateLine2(d1 - b / 2, dv / 2, 0, d1 + b / 2, dv / 2, 0);

Points[0] := (Lines[0] as ISketchline).GetStartPoint2 as ISketchPoint;

Points[1] := (Lines[0] as ISketchline).GetEndPoint2 as ISketchPoint;

Для выполнения операции “Повернутая бобышка” необходим замкнутый эскиз, для этого мы выделяем конечную точку первой линии и начальную второй и добавляем к ним взаимосвязь “слияние”:

Points[1].Select(true);

Points[2].Select(true);

MD.SketchAddConstraints('sgMERGEPOINTS');

MD.ClearSelection;

Добавление размеров происходит по координатам:

MD.AddDimension((b/4)/ 1000,0,0);

После построения эскиза и добавления взаимосвязей выполняем операцию “повернутая бобышка”, для этого выбираем осевую линию и эскиз, после чего выполняем саму операцию:

MD.FeatureManager.FeatureRevolve2(true, true, false, false, false, false, 0,

0, 6.2831853071796, 0, false, false, 0.01, 0.01, 0, 0, 0, true, true, true);

Для построения зуба нам необходима дополнительная плоскость, для этого выбираем необходимую плоскость и грань, по которой дополнительная плоскость будет проходить:

MD.FeatureManager.InsertRefPlane(2, 0, 32, 0, 0, 0);

Вычисление эвольвенты:

for J := length(masx2) - 1 downto 0 do

begin

masx2[J, I] := masx[J] \* Cos(DegToRad((180) \* I)) - masy[J] \*

Sin(DegToRad((180) \* I));

masy2[J, I] := masx[J] \* Sin(DegToRad((180) \* I)) + masy[J] \*

Cos(DegToRad((180) \* I));

Mas[count, 0] := masx2[J, I];

Mas[count, 1] := masy2[J, I];

count := count + 1;

end;

for J := 0 to length(masx10) - 1 do

begin

masx10[J] := masx1[J] \* Cos(DegToRad((180) \* I)) - masy1[J] \*

Sin(DegToRad((180) \* I));

masy10[J] := masx1[J] \* Sin(DegToRad((180) \* I)) + masy1[J] \*

Cos(DegToRad((180) \* I));

Mas[count, 0] := masx10[J];

Mas[count, 1] := masy10[J];

count := count + 1;

Создаем сплайн по точкам эвольвенты и замыкаем первую и последнюю точку линией:

MD.CreateSpline(A1);

if MD.SelectByID('Point1', 'SKETCHPOINT', qwerty[0] , qwerty[1] , 0) then

Points[14] := SelMgr.IGetSelectedObject(1) as ISketchPoint;

MD.ClearSelection;

if MD.SelectByID('Point'+IntToStr(Length(qwerty)-1), 'SKETCHPOINT', qwerty[Length(qwerty)-3] , qwerty[Length(qwerty)-2] , 0) then

Points[15] := SelMgr.IGetSelectedObject(1) as ISketchPoint;

MD.ClearSelection;

Lines[7] := MD.ICreateLine2(qwerty[0] , qwerty[1] , 0,

qwerty[Length(qwerty)-3] , qwerty[Length(qwerty)-2] , 0);

MD.ClearSelection;

if MD.SelectByID('Point'+IntToStr(Length(qwerty)), 'SKETCHPOINT', qwerty[0] , qwerty[1] , 0) then

Points[16] := SelMgr.IGetSelectedObject(1) as ISketchPoint;

MD.ClearSelection;

if MD.SelectByID('Point'+IntToStr(Length(qwerty)+1), 'SKETCHPOINT', qwerty[Length(qwerty)-3] , qwerty[Length(qwerty)-2] , 0) then

Points[17] := SelMgr.IGetSelectedObject(1) as ISketchPoint;

MD.ClearSelection;

Points[14].Select(true);

Points[16].Select(true);

MD.SketchAddConstraints('sgMERGEPOINTS');

Выполняем твердотельную операцию “Вытянутый вырез”:

IF1:=MD.FeatureManager.FeatureCut3(True, False, False, 0, 0, 0.02, 0.01, True, False, False, False, 5.23598775598299E-02, 1.74532925199433E-02, False, False, False, False, False, True, True, True, True, False, 0, 0, False);

MD.ClearSelection;

Создаем круговой массив вырезов:

MD.Extension.SelectByID2('Ось1', 'AXIS', 0, 0, 0, True, 1, nil, 0);

MD.FeatureManager.FeatureCircularPattern4(Z, 6.2831853071796, False, 'NULL', False, True, False);

После выполнения данных операций мы получаем модель конического зубчатого колеса. внешний вид модели представлен на рисунке 3.

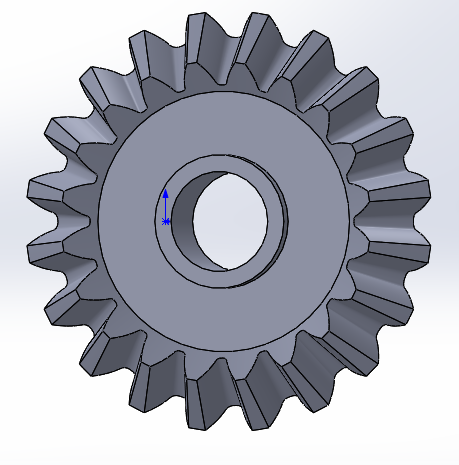


Рисунок 3 – 3D модель конического колеса

**10 Список литературы**

1. Эйдлина Г.М. Delphi: программирование в примерах и задачах : практикум/Г. М. Эйдлина, К. А. Милорадов. - 2012   
2. Фленов М.А. Библия Delphi. – СПб.: БХВ-Петербург, 2011. – 686 с.  
3. Дамир Тенишев Лингвистическое обеспечение автоматизированных систем. – М.: Профессия, 2011.-408 с.   
4. Сулейман Лалани, ActiveX. Рамеш Чандэк. М.: Попурри, 2010. - 624 с.   
5. Каплун, А. Б. ANSYS в руках инженера: практическое руководство / А. Б. Каплун, Е. М. Морозов, М. А. Олферьева.- М.: Либроком, 2015. – 272 с.   
6. Басов, К. А. ANSYS. Справочник пользователя. – М.: ДМК Пресс, 2014. – 640 с.