Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно–Уральский государственный университет
(национальный исследовательский университет)»
Факультет механико–технологический
Кафедра технология автоматизированного машиностроения

ПРОЕКТ ПРОВЕРЕН
Рецензент, _____
_____
«___» _____2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
_____ В.И. Гузеев
«___» _____2018 г.

Совершенствование методики расчета управляющих программ для фрезерных станков с ЧПУ путем учета размерного износа инструментов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–15.04.05.2018.582.00 ПЗ ВКР

Руководитель, д.т.н, профессор

_____/ И.А. Щуров/
«_____» _____2018 г.

Автор

студент группы П – 261

_____/ И.И. Гайфуллин/

«_____» _____ _____2018 г.

Нормоконтролер, к.т.н, доцент

_____ / И.В. Шмидт/

«_____» _____2018 г.

Челябинск 2018

АННОТАЦИЯ

Гайфуллин И.И. Выпускная квалификационная работа по теме «Совершенствование методики расчета управляющих программ для фрезерных станков с ЧПУ путем учета размерного износа инструментов». Челябинск: ЮУрГУ, 2018, 172 с., 54 ил., библ. список – наим. 52.

Выпускная квалификационная работа выполнена с целью повышения точности обработки фасонных поверхностей на фрезерных станках с ЧПУ путем совершенствования методики расчета управляющих программ на основе учета размерного износа инструментов

При выполнение ВКР были решены следующие задачи :

1 Проанализирована литература по существующим методам повышения точности фасонных поверхностей деталей на чистовых операциях фрезерования.

2 Доработан типовой порядок расчета управляющих программ путем разработки и внесения математической модели траектории движения инструмента с учетом его износа для обработки на 3х координатных станках с ЧПУ.

3 Создан алгоритм обеспечивающий коррекцию траектории дви-жения инструмента на основании ранее разработанной математической модели.

4 Экспериментально проверена компьютерная программа, а также апробирована разработанная математическая модель.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....9

1 АНАЛИЗ СУЩЕСТВУЮЩИХ МЕТОДОВ ПОВЫШЕНИЯ ТОЧНОСТИ СЛОЖНО-ПРОФИЛЬНЫХ ПОВЕРХНОСТЕЙ ДЕТАЛЕЙ НА ЧИСТОВЫХ ОПЕРАЦИЯХ НА ФРЕЗЕРНЫХ СТАНКАХ С ЧПУ.....11

1.1 Проблемы действующего производства получения высокоточных сложно-профильных поверхностей деталей на станках с ЧПУ…….....….....11

1.2 Существующие методы повышения точности обработки сложно-профильных поверхностей на станках с ЧПУ……...……12

1.3 Возможности существующих CAM систем по повышению точности обработки…….....……23

1.4 Цели и задачи ВКР, направленность работы, объект и предмет исследований .....….....…….27

Выводы 27

2 РАЗРАБОТКА МАТЕМАТИЧЕСКОЙ МОДЕЛИ ТРАЕКТОРИИ ДВИЖЕНИЯ ИНСТРУМЕНТА С УЧЕТОМ ЕГО ИЗНОСА ДЛЯ ОБРАБОТКИ НА 3Х КООРДИНАТНЫХ СТАНКАХ С ЧПУ.....29

2.1 Порядок расчета управляющих программ в PowerMill.....…..30

2.2 Доработанный порядок расчета управляющих программ станков с ЧПУ.....36

2.3 Разработка математической модели траектории движения инструмента учитывающий размерный износ инструмента.....38

Выводы.....43

3 РАЗРАБОТКА АЛГОРИТМА И КОМПЬЮТЕРНОЙ ПРОГРАММЫ ДЛЯ СТАНКОВ С ЧПУ, ОБЕСПЕЧИВАЮЩИЕ КОРРЕКЦИЮ ТРАЕКТОРИИ ДВИЖЕНИЯ ИНСТРУМЕНТА ...……..…….....44

3.1 Алгоритм обеспечивающий коррекцию траектории движения инструмента…….....…44

3.2 Компьютерная программа для станков с ЧПУ, обеспечивающий коррекцию траектории движения инструмента…….....….52

Выводы…….....…….62

4 ЭКСПЕРИМЕНТАЛЬНАЯ ВЕРИФИКАЦИЯ ПОЛУЧЕННОЙ КОМПЬЮТЕРНОЙ МОДЕЛИ.....64

4.1 Цельи задачи проведения эксперимента 64

4.2 Методика проведения эксперимента.....64

4.2.1 Применяемый инструмент и оборудование используемое в эксперименте 64

4.2.2 Значения размерного износа в зависимости от пройденной длины 68

4.3 Сравнение результатов работы созданной компьютерной программ с результатами полученных из CAM систем ….....…….....76

Выводы 83

ЗАКЛЮЧЕНИЕ.....84

БИБЛОГРАФИЧЕСКИЙ СПИСОК..... 86

ПРИЛОЖЕНИЕ А.....92

ВВЕДЕНИЕ

В настоящее время детали со сложно-профильными поверхностями такие как: моноколеса, лопатки турбин и.т.д все чаще применяются в машиностроении. Обрабатываются сложно-профильные поверхности чаще всего на станках с числовым программным управлением (ЧПУ). Разработка управляющих программ (траектория движения инструмента), без применения специализированных программ практически невозможно. В следствие это, для разработки управляющих программ для станков с ЧПУ используют специализированные компьютерные программы, которые относятся к системам проектирования класса Computer Aided Manufacturing (CAM). Данная система как и системы данного класса проектирования Computer Aided Design (CAD) основаны на принципе классического твердотельного моделирования , где инструмент и деталь описываются набором поверхностей и условиями их пересечения. При разработке траектории движения инструмента решается задача формообразования, в которой определяются условия касания инструмента с поверхностью детали. Системы проектирования не учитывают влияние технологических факторов на качество получаемых поверхностей, такие факторов как : упругие и тепловые деформации элементов технологической системы, погрешности их размеров и формы, износ инструмента и др факторы.

Как известно, детали такие как турбинные лопатки и моноколеса изготавливают из высокопрочных материалов. При этом на чистовых операциях фреза обрабатывает всей поверхностью деталь. В процессе обработки режущие поверхности фрезы изнашиваются, при этом ее исходная инструментальная поверхность изменяется, оставляя все больше материала на заготовке. Данный факт негативно сказывается на эксплуатационных качествах детали типа ротор, т.к роторы энергетических машин развивают скорости от 60 до 70 тысяч оборотов в минуту и малейший дисбаланс деталей ротора, малейшая неточность изготовления приводит к повышению допустимых вибраций в энергетической машине, в следствие к занижению выходных параметров вы-

даваемой мощности и снижению эксплуатационного срока. С целью компенсации влияния износа инструмента на точность сложно-профильных поверхностей деталей была усовершенствована методика расчета управляющих программ для фрезерных станков с ЧПУ путем учета размерного износа инструментов.

1 АНАЛИЗ СУЩЕСТВУЮЩИХ МЕТОДОВ ПОВЫШЕНИЯ ТОЧНОСТИ СЛОЖНО-ПРОФИЛЬНЫХ ПОВЕРХНОСТЕЙ ДЕТАЛЕЙ НА ЧИСТОВЫХ ОПЕРАЦИЯХ НА ФРЕЗЕРНЫХ СТАНКАХ С ЧПУ

1.1 Проблемы действующего производства получения высокоточных сложно-профильных поверхностей деталей на станках с ЧПУ

В настоящее время с появлением станков с числовым программным управлением (ЧПУ) на машиностроительных предприятиях возросла производительность выпускаемых изделий, а также их сложность. Под сложностью выпускаемых изделий понимается сложно-профильные поверхности по типу винтовые поверхности лопаток турбин, моноколес и т.д. Примеры сложно-профильных поверхностей приведены на рисунке 1.1 [1].

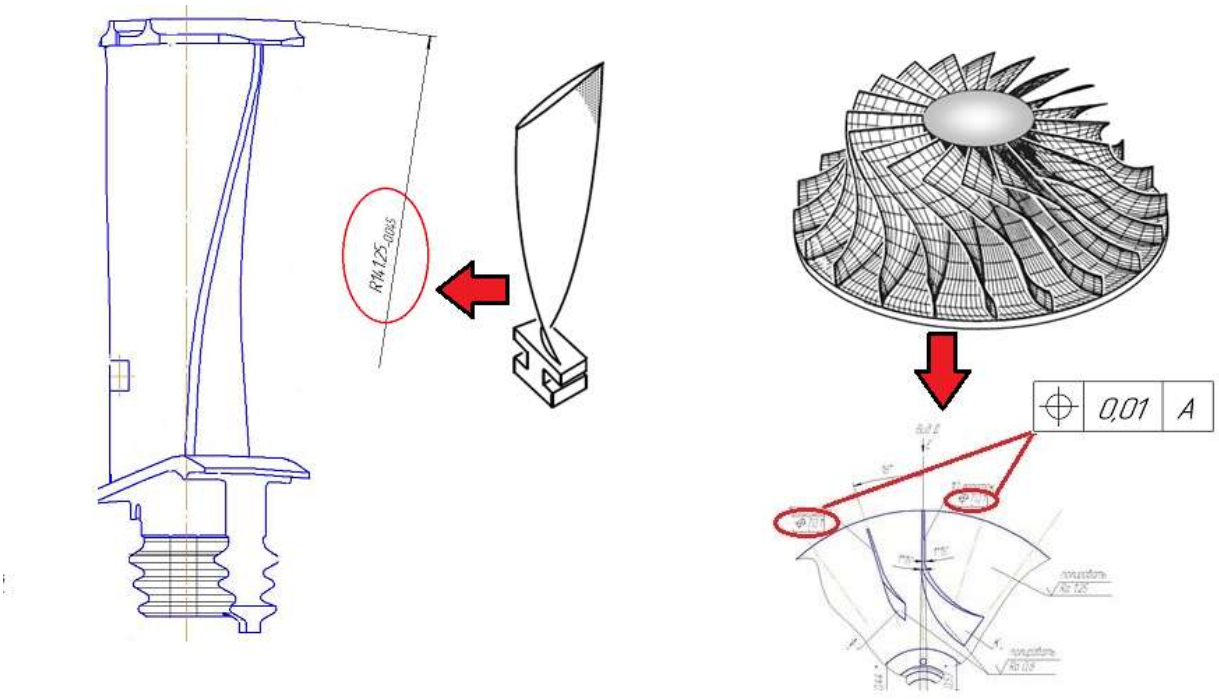


Рисунок 1.1 – Детали с сложно-профильными поверхностями [1]

Обработка некоторых сложно-профильных поверхностей таких как : лопатки турбин, возможна только фасонной фрезой с сферической режущей частью, связано это с постоянным изменением геометрии зоны резания. Если обрабатываемые поверхности из высокопрочных материалов, то время работы

станка продолжительна свыше 24 часов. При обработке высокопрочных материалов режущий инструмент изнашивается т.е исходная инструментальная поверхность инструмента изменяется, оставляя больше материала на заготовке. Чаще всего контроль подобных сложно-профильных поверхностей происходит на контрольно-измерительных машинах (КИМ), если профиль поверхности не совпадает с конструкторской документацией (3D моделью детали). Тем самым доведение детали на станке с ЧПУ возможна путем пробных проходов увеличивая трудоемкость процесса либо доведением ручными способами [2].

Основными проблемами действующего производства являются :

- Уменьшение производительности связанное с доведение профиля детали многократными рабочими проходами ;
- введение ручной доводки требуемого профиля детали ;
- трудоемкость корректирования управляющей программы связанное с размерным износом режущего инструмента.

1.2 Существующие методы повышения точности обработки сложно-профильных поверхностей на станках с ЧПУ

Рассмотрим, по мнению автора, основные технологические факторы влияющие на точность обработки и методы повышение точности учитывающие влияние данных факторов (рисунок 1.2)

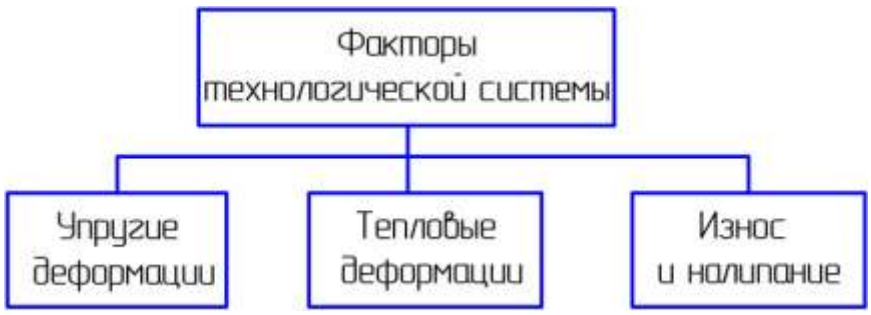


Рисунок 1.2 – Основные технологические факторы влияющие на точность обработки

Основными технологическими факторами влияющие на точность обработки являются : тепловые и упругие деформации технологической системы (ТС) , геометрические погрешности изготовления станка и износ его элементов, погрешности установки , геометрические погрешности и износ режущего инструмента.

В процессе обработки деталей на металлорежущем оборудовании, происходит выделение тепла. Выделение тепла можно разделить на : возникающее непосредственно в зоне обработки заготовки, в узлах станка. Теплообразование в зоне резания является причиной появления погрешностей обработки изменяя линейные размеры инструмента и заготовки, теплообразование в зоне резания представлено на рисунке 1.3. Но данным фактом можно пренебречь т.к в обрабатываемую деталь и инструмент, переходит малое количество тепла путем введения смазочно-охлаждающей жидкости (СОЖ) в зону обработку [3]

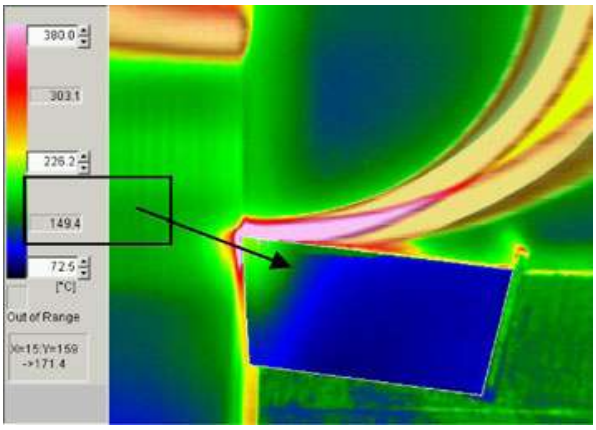


Рисунок 1.3 – Теплообразование в зоне резания [4]

Подача СОЖ не обеспечивает полное отведения тепла от обрабатываемой детали и инструмента, но учитывая тот факт что основное количество тепла сосредотачивается в стружке [5] . Нагревание узлов станка происходит в результате потерь на трение в механизмах и тепловыделения в гидроприводах и других узлах. В результате нагрева узлов станка возникают температурные деформации, чаще всего влиянию температурных деформаций подвержена станина станка по причине того что большое количество тепла передается че-

рез СОЖ , отводящей тепло из зоны резания [6]. Станина нагревается неравномерно, разность температур может достигать 10 °С ,обусловлено нерациональным расположением электродвигателей, резервуаров масла и других источников тепла. При данных условиях станина подвержена деформации тем самым изменяя взаимное расположение основным узлов. Пример деформации станины представлен на рисунке 1.4.

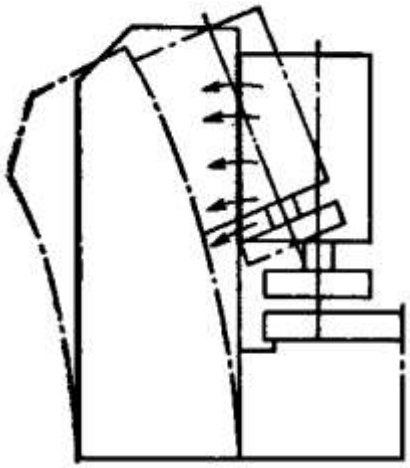


Рисунок 1.4 – Влияние температурных деформаций станины [7]

В настоящее время крупные станкостроительные заводы при разработке новых конструкций станка вводят методы компенсации влияния температурных деформаций возникающие в процессе обработки. Прецизионные металлорежущие станки компании «Mazak» оснащены интеллектуальной системой компенсацией тепловых деформаций [8]. Данная система позволяет поддерживать стабильную температуру рабочих органов станка применяя специальные датчики слежения которые компенсируют температурные колебания в основных узлах станка. Работа интеллектуальной системы компенсации температурных колебаний компании «Mazak» представлена на рисунке 1.5 на примере температурных колебаний шпинделя токарного шпинделя. Где «No compensation» без использования интеллектуальной системы (ИС), «Compensation method» с использованием других методов без ИС, «TAS» с использованием ИС.

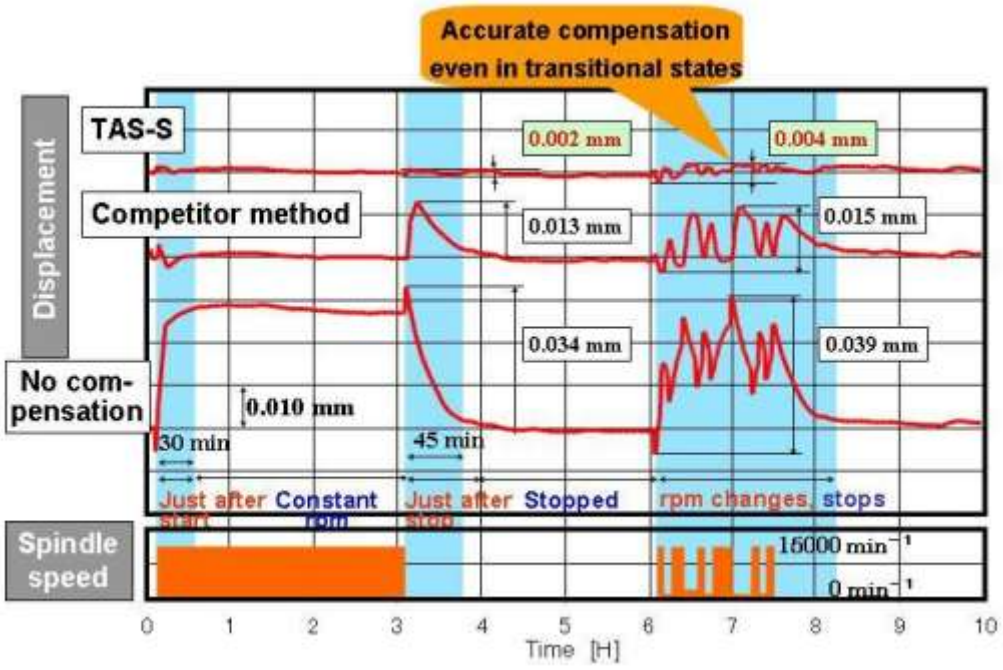


Рисунок 1.5 – Температурные смещения токарного шпинделя [8]

Следующая станкостроительная компания «Okuma» разработала концепцию прецизионных станков «Therno-Friendly-Concept» данная концепция позволяет сохранять высокую точность позиционирования при отсутствие терморегулируемых помещений. Похожий метод компенсации температурных деформаций разработали в фирме «Kitamura», метод основан на поддержание стабильной температуры основных рабочих органов станка путем использования специальной адаптивной системы контроля и компенсации температурных деформаций состоящая из следящих датчиков на основных узлах станка.

Следующий метод повышения точности в зависимости от температурных деформаций представлен в работе [9]. Метод заключается в прогнозировании температурных перемещений исполнительных органов станка на основе построения циклограмм работы станка и показаний датчиков. Основанием для разработки метода стало что при установлении функциональной взаимосвязи «перемещение–температура» характеристика показывает что для различных режимах работы станка может фиксировать одно значение температуры с различным температурным смещением. График зависимости представлен на ри-

сунке 1.6. Результатом работы стало повышения точности на 0,01 мм за учета тепловых смещений вдоль оси шпинделя.

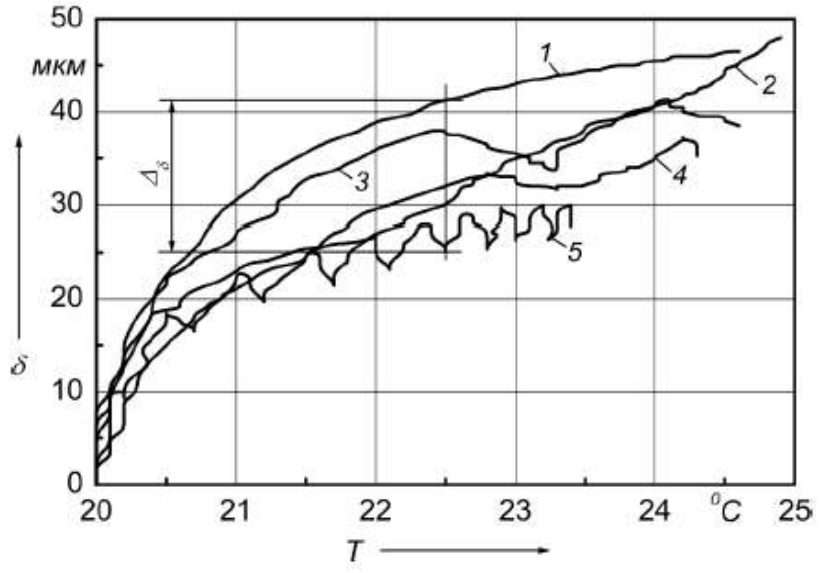


Рисунок 1.6 – Взаимосвязь «перемещение–температура» [9]

Аналогичные методы прогнозирования тепловых характеристик станков в условиях переменных тепловых режимах в работах [9-12]. Компенсация основана на полученных данных с следящих датчиков. В направлении компенсации тепловых смещений разрабатывались патенты схожие с способами у компаний Mazak, Okuma. данный способ позволяет установить величины смещения шпинделя станка в процессе обработки, введением коррекции в перемещении рабочих органов. Ключевым недостатком является невозможность полной одновременной компенсации отклонений рабочих органов в трех осях [13].

Следующий фактор влияющий на точность обработки это упругие деформации технологической системы. Упругие деформации возникающие при обработке резанием под действием сил резания изменяют положение инструмента либо заготовки от первоначальных положений (рисунок 1.7), в следствие ведет к потери точности обработки [14].

Рисунок 1.7 – Изменение положение режущего инструмента в процессе фрезерования [14]

На величину упругих деформаций кроме сил резания оказывает влияние жесткость технологической системы. Жесткость технологической системы называют отношение составляющей силы резания, направленной по нормали к обрабатываемой поверхности, к смещению режущей кромки относительно это же поверхности заготовки и в этом же направлении [15]. Влияние упругих деформаций, представлено на рисунке 1.8.

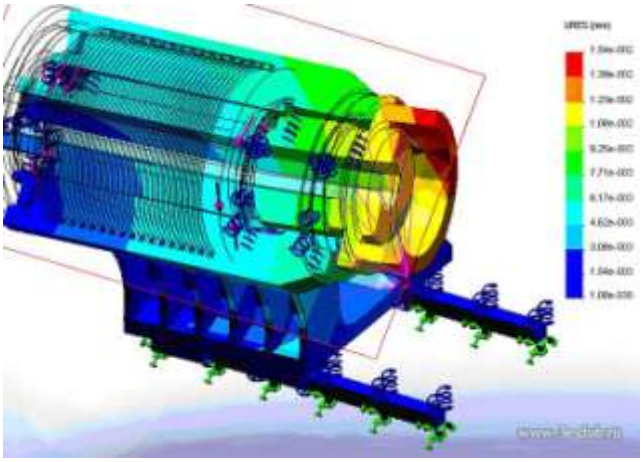


Рисунок 1.8 – Эпюра упругих напряжений [14]

На рисунке 1.8 показано сечение эпюры упругих напряжений таких узлов как : шпинделя, корпуса шпиндельного узла, встроенного электродвигателя, на фланце шпинделя установлена деталь и к детали приложена некоторая сила резания. На шпиндель действует момент равный моменту силы резания. На корпус действует равная по величине сила что и на шпиндель только с отрицательным знаком.

В работах В.И Гузеева [16-18], В.А . Батуева [19], А.В Выбойщика [20], В.В Батуева [21] рассматривался вопрос повышения точности за счет стабилизации сил резания, тем самым уменьшая влияния упругих деформаций. В.В. Батуев предлагал управлять точностью через регулирование подачи, данный метод рассматривался на пространственно-сложных поверхностях после черновой обработки имеющих равный припуск [21]. В.А. Выбойщик рассматривал силовую нестабильность частного случая процесса объемного фрезерования при работе по схеме «поперек» , т.е под углом 90º относительно ступенек. А управление точностью предлагал осуществлять количеством уточняющих проходов.

Рассмотрим следующие факторы технологической системы такие как износ и налипание режущего инструмента и методы уменьшения влияния данных факторов на точность обработки. Следует сказать, что износ режущего инструмента – это изменение размеров, формы и массы режущего инструмента в следствие разрушения поверхностного слоя, налипание – это молекулярное схватывание материалов инструмента и обрабатываемой детали [22]. Процессы резания сопровождаются износом режущего лезвия, классифицируют следующие виды износа: износ по передней поверхности, износ по задней поверхности, абразивный износ, адгезионный износ, диффузионный износ, окислительный износ.

При чистовых операциях режущий инструмент снимает минимальный припуск после черновой либо после получистовой операции. Износ по передней поверхности преобладает при съеме большого припуска [22]. Также следует отметить, что износ по передней поверхности в первичном приближении

влияет только на качество поверхности , конкретнее на параметры шероховатости поверхности.

Адгезионный износ и др. подобные виды износа возникают только при высоких температурах между режущим лезвием инструмента и поверхностью заготовки , эффект после действия адгезионого износа аналогичный сварки трением. Изменяет первоначальные геометрические параметры режущего лезвия тем самым изменяя качество и получаемые размеры на обрабатываемой детали. В настоящее время производители смазочно-охлаждаемой технологических средств (СОТС) разработали новые виды средств при котором влияние адгезионных и др. видов износов уменьшаются до состояния при котором можно пренебречь в процессе обработки [3, 23].

Последний вид износа является износ по задней поверхности режущего инструмента (РИ). При обработке высокопрочных материалов таких как : ВЖЛ, ХН50ВТЮБЛ и др. сплавов с твердостью свыше 50 HRC (твердость по роквеллу), где наблюдается существенный износ инструмента. Последствия износа по задней поверхности РИ является перенос искаженной геометрии РИ на поверхность детали, учитывая то, что при обработке на станках с ЧПУ сложно-профильных поверхностей, формообразование поверхностей производиться построчно [1].

Рассмотрим влияние некоторых факторов на интенсивность износа инструмента :

1) влияние скорости резания на интенсивность износа. Общеизвестным фактором является что главным фактором влияющим на интенсивность износа является скорость резания. Данный фактор объясняется, тем что чем выше скорость резания оказывается действия на силу трения как температурный и скоростной фактор. Как температурный фактор понижает прочность контактирующего слоя стружки тем самым уменьшая силу трения. Скоростной фактор, повышение скорости резания приводит к повышению скорости деформации тем самым приводит к уменьшению коэффициента контактного трения,

влияние скорости резания на контактные явления по задней поверхности инструмента представлены на рисунке 1.9.

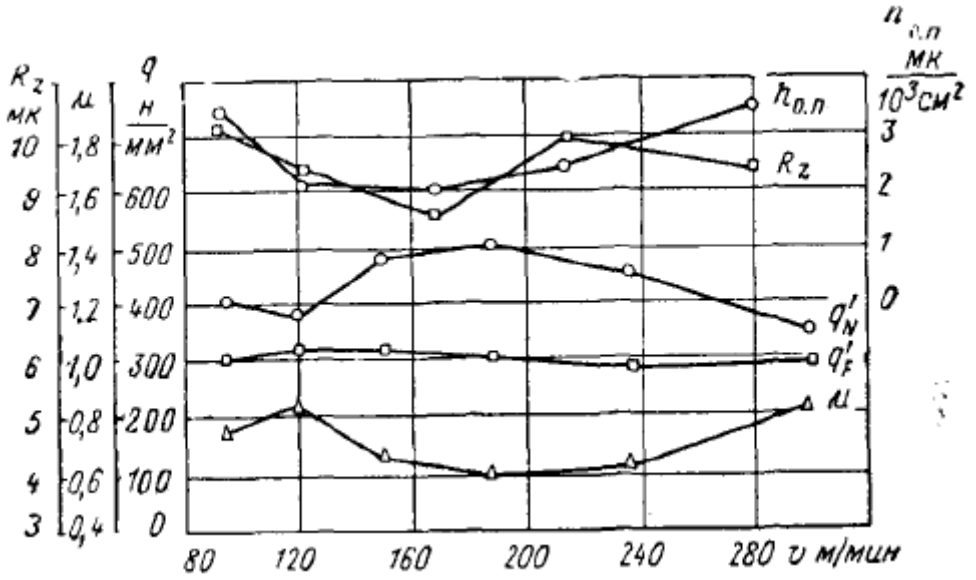


Рисунок 1.9 – Влияние скорости резания на интенсивность износа по задней поверхности инструмента [37]

На рисунке 1.8 показана обработка стали 40ХНМА, резцом Т14К8 при значениях глубины резания 1мм, подачи 0,21 мм/об, при данным значения повышение скорости показывает, снижение коэффициента трения μ^1 при повышении скорости резания. По данным исследований, приведенном в источнике [37] можно сделать выводы : , что интенсивность износа зависит от скорости трения и от нормального удельного давления, сравнительная износостойкость применяемого материала инструмента соответствует относительной интенсивности износа при обработке жаропрочных сплавов резанием ;

2) влияние подачи на интенсивность износа. Многочисленные исследования показали (рисунок 1.10), что при обработке на пониженных скоростях резания, повышение подачи приводит к снижению показателя трения μ^1 тем самым приводит к снижению размерного износа инструмента. На рисунке 1.9 показано влияние подачи при обработке стали 12Х18Н9Т резцом Т30К4, при глубине резания 0,5 мм и скорости резания 56 м/мин. Также следует отметить, что при высоких скоростях резания повышение подачи вызывает повышения коэффициента трения тем самым увеличивает изностойкость инструмента [37];

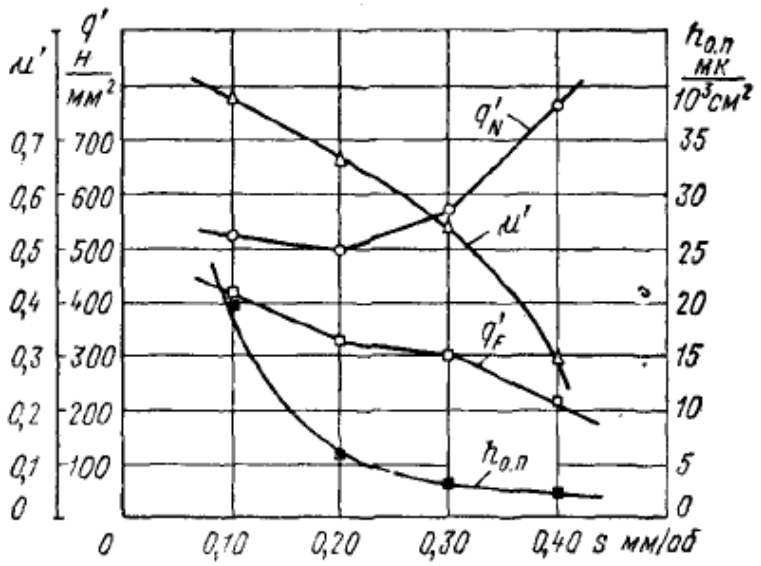
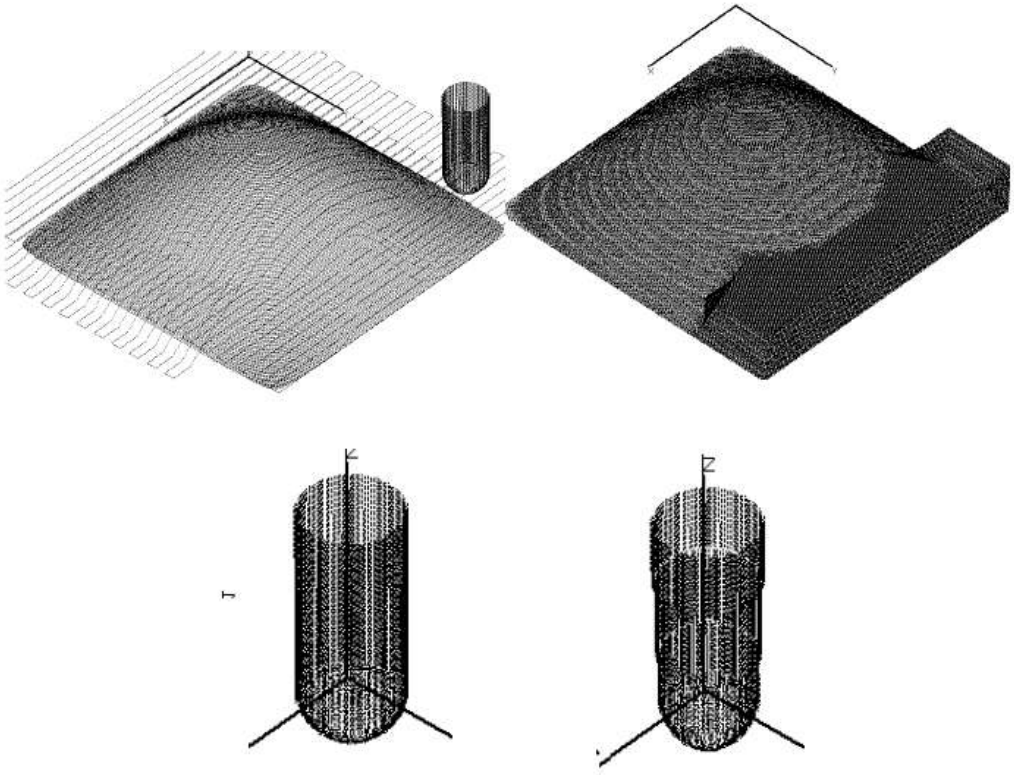


Рисунок 1.10 – Влияние подачи на интенсивность износа по задней поверхности инструмента [37]

3) влияние глубины резания. На чистовым операциях припуск минимален, в следствии влияние глубины резания не может учитываться.

Проведен анализ работ Н.А Симанина [24], А.К Гороховского [25], Евдокимов Д.В [26], Д.Ю Пименова [27], J. Menezes [28], M. Rizal [29] G. Skordaris [30] Wang, G [31] Stavropoulos P. [32] Bleys P [33] Ming L [34] Колесников В.И[35] Щурова И.А [36], где рассматривалось обработка сложнопрофильных поверхностей на фрезерных станках с ЧПУ. В работе Н.А Симанина [24] приведен метод планетарного фрезерования, позволяющий вводить коррекцию погрешности диаметра фрезы, данный метод по предположению автора позволит повысить стойкость фрез в 10 раз. Предложенный метод не эффективен на чистовых операциях фрезерования при обработке сложно-профильных поверхностей, кинематика планетарного движения будет вносить дополнительные погрешности в виде вибрации тем самым понизит качество поверхности. В работах Д.В Евдокимова [26], А.К Гороховского [25], Д.Ю Пименова [27], рассмотрены вопросы обеспечения износа на основе допустимого износа, величины допустимого износа определялись только для цилиндрических фрез. При обработке сложно-профильных поверхностей, как было сказано ранее обработка ведется построчно, касание исходной инст-

рументальной поверхности (ИИП) фрезы и поверхности детали как правило неравномерно по всему ИИП фрезы, вследствие определить допустимый износ затруднительно. В работе И.А Щурова [36] рассмотрен принципиально новый подход, применяя воксельный подход для решения задачи формообразования (рисунок 1.11), но данный метод не учитывает тот факт, что участки ИИП фрезы находятся разное время в касание с поверхностью детали, тем самым не учтена неравномерность износа ИИП фрезы . Остальные работы основаны на методе повышения точности уменьшая износ режущего инструмента путем коррекции тепловых и упругих деформаций.



. Слева узловые точки вокселей ИИП новой фрезы и «изношенной», справа отклонения поверхности детали от поверхности, заданной CAD моделью

Рисунок 1.11 – Воксельный подход [36]

Анализ имеющихся работ показал:

1) слабо изучен учет размерного износа режущего инструмента при обработке сложно-профильных поверхностей из высокопрочных материалов на чистовых операциях фрезерования ;

2) отсутствие прогнозирующих моделей износа инструмента по задней поверхности.

1.3 Возможности существующих CAM систем по повышению точности обработки

В настоящее время современное машиностроение характеризуется широким фронтом проектных задач связанных с сложными изделиями. Для выполнения технологических задач в сжатые сроки используют интегрированные системы автоматизированного проектирования (САПР). Система автоматизированного проектирования – система, реализующая информационную технологию выполнения функций проектирования, представляет собой организационно-техническую систему, предназначенную для автоматизации процесса проектирования [38]. В системы проектирования входят : CAD, CAM, CAE системы. В рамках выполняемой работы рассмотрению подлежат только CAM системы.

Computer Aided Manufacturing (CAM) – система автоматизированного производства, используется для проектирования и производства изделий , относиться к программному обеспечению. Основные функции CAM систем: построение траектории движения инструмента и заготовки в процессе обработки, моделирование процесса обработки, генерация постпроцессоров для конкретных типов оборудования [38]. Рассмотрим широко известные CAM программы применяемые в настоящее время : Creo parametric, Powermill, NX CAM , ESPIRIT, CATIA, NCG CAM, EdjeCAM, SprutCAM.

Creo parametric – программный продукт [39] компании PTC. CAM система Creo parametric, в качестве возможностей повышения точности на 3х осевой фрезерной обработке по нашим экспертным оценкам представляет :

- минимизация внезапных изменений направлений при высокоскоростной обработки, тем самым обеспечивая качество обрабатываемых поверхностей при высоких скоростях подачи уменьшая величину упругих перемещений звеньев ТС.
- непрерывный контакт инструмента с деталью, уменьшение температурный перепадов между инструментом и деталью [39].

Высокоскоростная обработка (ВСО) – направление в фрезеровании позволяющая увеличить скорость резания, путем уменьшения объема снимаемой стружки снимаемой с высокой минутной подачей тем самым увеличивая отвод тепла от детали и инструмента.

При рассмотрении данного программного продукта автором не обнаружены функции по повышению точности в зависимости от размерного износа режущего инструмента.

Powermill – программный продукт компании Autodesk. PowerMill является известной в области фрезерной обработки программным продуктом. Основным направлением развития является 5 – осевая обработка. В качестве методов по повышению точности для 3х осевой обработки, PowerMill представляет стабилизацию температуры в зоне резания путем сглаживания траектории движения инструмента при высокоскоростной обработки.

При сглаживании траектории PowerMill использует математику сплайнов для описания кривых и плоскостей любой формы. Интерполяция по кривым позволяет менять направление постепенно [40]. При рассмотрении данного программного продукта автором не обнаружены функции по повышению точности в зависимости от размерного износа режущего инструмента.

NX CAM – программный продукт компании Siemens. NX CAM предлагает широкий спектр функций для программирования фрезерных и токарных станков с ЧПУ. Основным направлением развития является повышение эффективности при 5 координатной обработке, а также скорости разработки управляющих программ. Для 3х координатной обработки NX разработаны плавные схемы траектории движения инструмента при ВСО, в результате

обеспечивается качество поверхности при высоких скоростях резания [41]. При рассмотрении данного программного продукта автором не обнаружены функции по повышению точности в зависимости от размерного износа режущего инструмента.

ESPIRIT – программный продукт компании DP Technology. ESPIRIT предлагает своим пользователем полноценную CAM систему для всех типов обработки. Ключевым направлением в развитии является оптимизация операций обработки на многоцелевых станках, а также 5 координатная обработка. При 3х координатном фрезеровании в качестве методов по повышению точности ESPIRIT предлагает сглаживание траектории движения инструмента при ВСО, тем самым обеспечивая качество получаемой поверхности [42]. При рассмотрении данного программного продукта автором не обнаружены функции по повышению точности в зависимости от размерного износа режущего инструмента.

CATIA – программный продукт компании Dassault Systemes. CATIA является одной из САПР высокого уровня. Основным направлением развития является, создание единой системы с различными возможностями, совместить возможности CAD, CAM, CAE систем в единую систему, без встраивания дополнительных модулей в области автомобилестроения [43]. При рассмотрении данного программного продукта автором не обнаружены функции по повышению точности в зависимости от размерного износа режущего инструмента. При рассмотрении данного программного продукта автором не обнаружены функции по повышению точности в зависимости от размерного износа режущего инструмента.

NCG CAM – программный продукт разработан с направленностью при использовании в инструментальном производстве, является независимым CAM системой. Основной направленностью является формообразование штампов и пресс-форм. NCG CAM представляет в качестве методов повышения точности компенсацию вибраций за счет оптимизации скоростей подач и траектории движения инструмента [44]. При рассмотрении данного про-

граммного продукта автором не обнаружены функции по повышению точности в зависимости от размерного износа режущего инструмента.

EdjeCAM – программный комплекс созданный для разработки управляющих программ для станков с ЧПУ. Основным направлением развития являются : 5 координатная обработка с множеством чистовых стратегий обработки, а также обработка на многоцелевых станках с ЧПУ [45]. При рассмотрении данного программного продукта автором не обнаружены функции по повышению точности в зависимости от размерного износа режущего инструмента.

SprutCAM – программный продукт отечественных разработчиков компании СПРУТ-Технология. Ключевым направлением развития является снижение трудоемкости разработки управляющих программ для станков с ЧПУ [46]. При рассмотрении данного программного продукта автором не обнаружены функции по повышению точности в зависимости от размерного износа режущего инструмента.

При анализе CAM систем различных производителей было установлено, что учет влияния технологических факторов на точность обработки реализован не до конца, а именно не обнаружены наличие учета влияния размерного износа инструмента на точность обработки .

На основании анализа литературы сформулированы следующие факторы актуальности :

- Отсутствие учета размерного износа режущего инструмента;
- Отсутствие прогнозирующих моделей износа инструмента.

При рассмотрении факторов актуальности, выявлено следующее противоречие : необходимость повышения точности при обработке сложнопрофильных поверхностей и учета размерного износа режущего инструмента.

На основании выявленного противоречия сформулирована проблема, и гипотеза. Проблема – получение качественных сложно-профильных поверхностей с учетом фактического состояния режущего инструмента. Решением проблемы сформулирована гипотеза – повышение точности обработки фа-

сонных поверхностей может быть достигнуто на основе динамической корректировки траектории движения режущего инструмента с учетом его износа.

На основании анализа литературы и выявленной проблемы сформулированы цель и задачи исследования.

1.4 Цель и задачи ВКР, направленность работы, объект и предмет исследования

Цель: Повышение точности обработки фасонных поверхностей на фрезерных станках с ЧПУ путем совершенствования методики расчета управляющих программ на основе учета размерного износа инструментов.

Задачи:

1) Анализ существующих методов повышения точности фасонных деталей на чистовых операциях фрезерования ;

2) Разработка математической модели траектории движения инструмента с учетом его размерного износа на операциях 3х осевого фрезерования ;

3) Разработка алгоритма и компьютерной программы для станков с ЧПУ, обеспечивающие коррекцию траектории движения инструмента за счет учета размерного износа инструмента ;

4) Экспериментальная проверка полученной компьютерной программы.

Направлением работы является совершенствование методики расчета управляющих программ.

Объект исследования : точность обработки на фрезерных станках с ЧПУ.

Предмет исследования : взаимосвязь размерного износа инструментов с точностью получаемых поверхностей.

В данной главе, был проведен обзор литературных данных по теме исследования, в процессе анализа было установлено :

1) слабо изучен учет размерного износа режущего инструмента при обработке сложно-профильных поверхностей из высокопрочных материалов на чистовых операциях фрезерования ;

2) отсутствие прогнозирующих моделей износа инструмента по задней поверхности ;

3) при анализе CAM систем различных производителей было установлено, что учет влияния технологических факторов на точность обработки реализован не до конца, а именно не обнаружены наличие учета влияния размерного износа инструмента на точность обработки .

2 РАЗРАБОТКА МАТЕМАТИЧЕСКОЙ МОДЕЛИ ТРАЕКТОРИИ ДВИЖЕНИЯ ИНСТРУМЕНТА С УЧЕТОМ ЕГО ИЗНОСА ДЛЯ ОБРАБОТКИ НА 3Х КООРДИНАТНОМ СТАНКАХ С ЧПУ

В процессе резания инструмент и заготовка движутся относительно. При относительном движении инструмента и заготовки срезается припуск в результате, которого образуется обработанная поверхность. Для формирования заданной поверхности , профилирующие участки режущих кромок должны располагаться на некоторой поверхности, которая в процессе обработки касается детали и является касательной к ней. Данная поверхность называется исходной инструментальной поверхностью (ИИП) либо производящая поверхность [49].

При задании траектории в известных CAM системах не учитываются влияние технологических факторов, т.е технологическая система абсолютно жесткая, отклонения от разработанной траектории движения инструмента и заготовки в процессе обработки отсутствуют, узлы в технологической системе не изнашиваются, тепловые деформации отсутствуют. В следствие этого реальный процесс формообразования переводят в идеальный. Идеальному процессу формообразования соответствует номинальная поверхность детали, в которой отсутствуют микронеровности и другие отклонения.

Основной задачей обработки на фрезерных станках с ЧПУ сводиться к определению траектории движения инструмента. Задача формирования траектории движения при обработке заданным инструментом по схеме формообразования называется обратной задачей формообразования. Решением данной задачи является нахождение точек контакта ИИП и номинальной поверхности детали (НПД).

2.1 Порядок расчета управляющих программ в PowerMill

Типовая порядок расчета управляющих программ для станков с ЧПУ одинаковая для всех CAM систем. Порядок представлена на рисунке 2.1.

Рисунок 2.1 – Типовой порядок расчета управляющей программы станков с ЧПУ в PowerMill

Твердотельная модель – трехмерная электронная геометрическая модель, представляющая форму изделия как результат композиции заданного множества геометрических элементов с применением операций булевой алгебры к этим геометрическим элементам [50] .

Рассмотрим классическую методику создания управляющей программы обработки сложно-профильной поверхности в виде полусферы (рисунок 2.2) для 3х координатного фрезерного станка с ЧПУ в CAM системе, на примере CAM системы PowerMill.

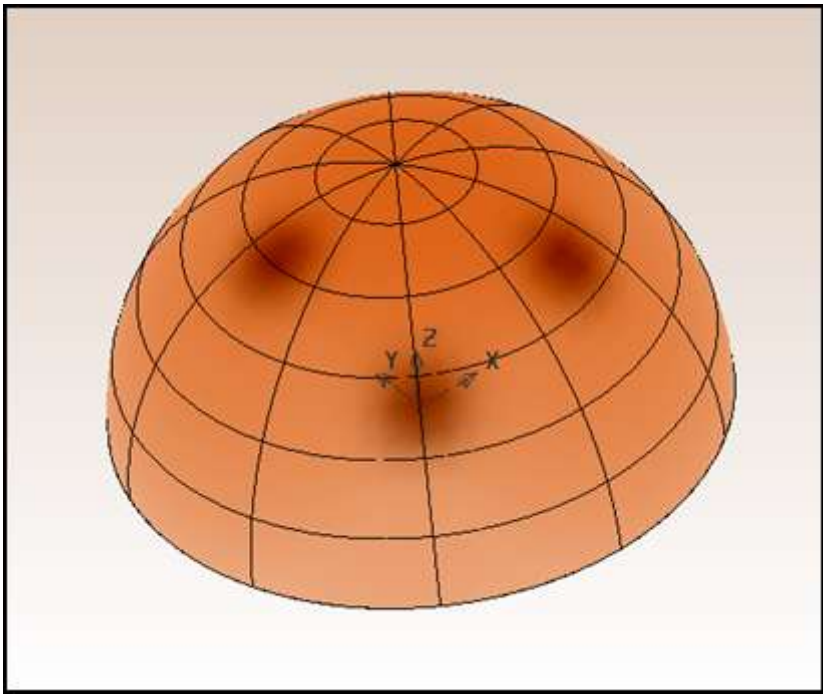


Рисунок 2.2 – Твердотельная модель сложно-профильной поверхности в виде полусферы

После создания и импортирования твердотельной модели в CAM систему, задаются геометрические параметры режущего инструмента (рисунок 2.3). Под геометрическими параметрами подразумевается габариты инструмента, радиус режущей части и количество зубьев. Следующий шаг это задание схемы формообразования обрабатываемых поверхностей (стратегия обработки).

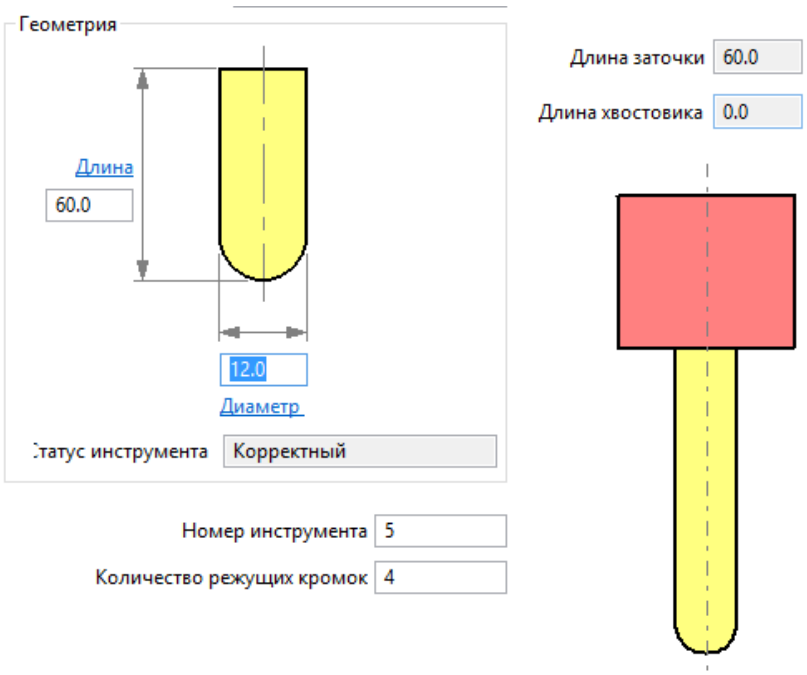


Рисунок 2.3 – Геометрические параметры инструмента

При обработке исходная инструментальная поверхность инструмента контактирует с поверхностью детали, осуществляя формообразование обрабатываемых поверхностей. Контакт осуществляется тремя способами формообразования номинальной поверхности детали [49].

- Контакт ИИП осуществляется по всей номинальной поверхности, при такой схеме формообразования требуется только подача сближения ;
- ИИП контактирует с НПД по образующей линии, для осуществления такой схемы формообразования кроме подачи сближения необходима подача по направляющей ;
- ИИП контактирует с НПД в точке, для осуществления такой схемы формообразования кроме подачи сближения необходимы две подачи : по образующей и направляющей (рисунок 2.4)

В зависимости от способа формообразования поверхности выбирается схема формообразования (стратегия обработки). Стратегии обработки CAM системы PowerMill для чистовых операций фрезерования представлены на рисунке 2.5.

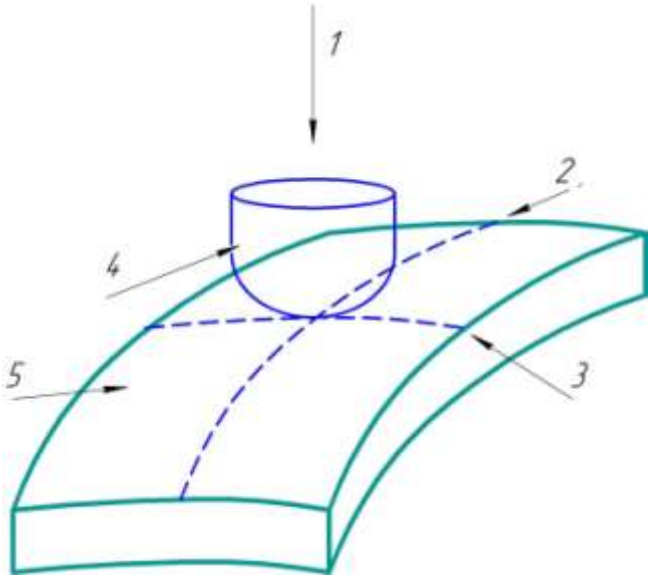


Рисунок 2.4 – Способ формообразования номинальной поверхности детали

1 – подача сближения, 2; 3 – формообразующие подачи 4 – производящая (ИИП) поверхность инструмента , 5– НПД

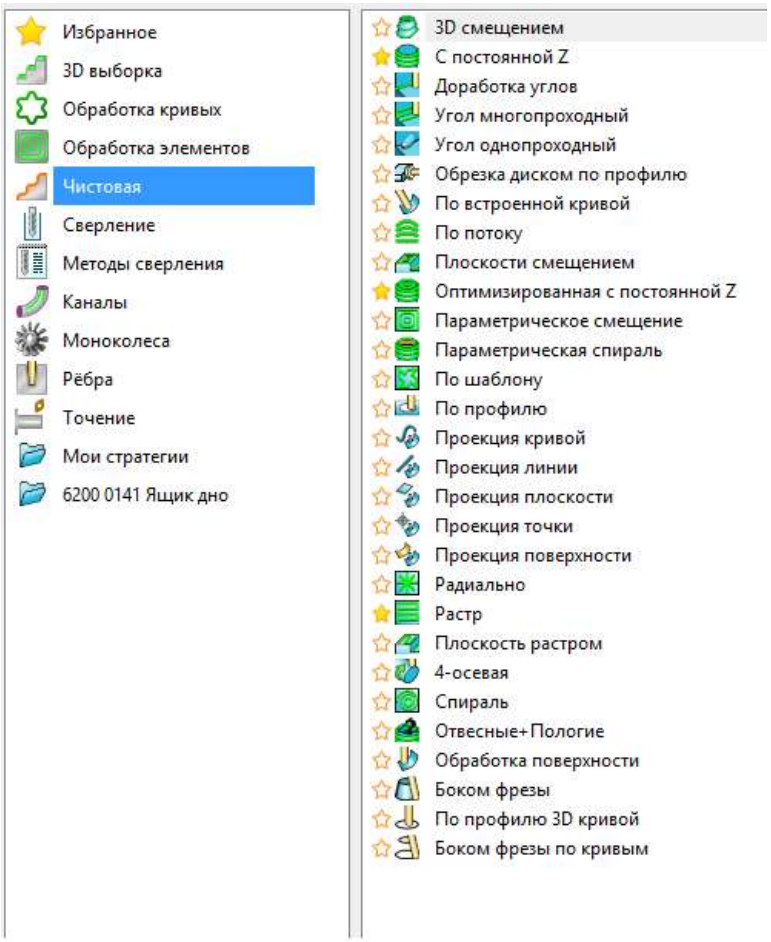


Рисунок 2.5 – Чистовые стратегии обработки CAM системы PowerMill

Для рассматриваемого примера обработки полусферы, соответствует третий способ формообразования поверхности, когда контакт ИИП с НПД в точке. При таком способе формообразование сложно-профильных поверхностей возможно только при движении РИ построчно. При этом движение инструмента по строке формообразования является непрерывным следящим движением в виде геометрических объектов«отрезок прямой». Огибание поверхности детали происходит при точечном контакте. Положение инструмента на траектории движения, определяется опорными точками траектории, выделенные точки на рисунке 2.6 происходит контакт ИИП с НПД. При движении построчно наиболее подходящая стратегия обработки в CAM системе «растр».

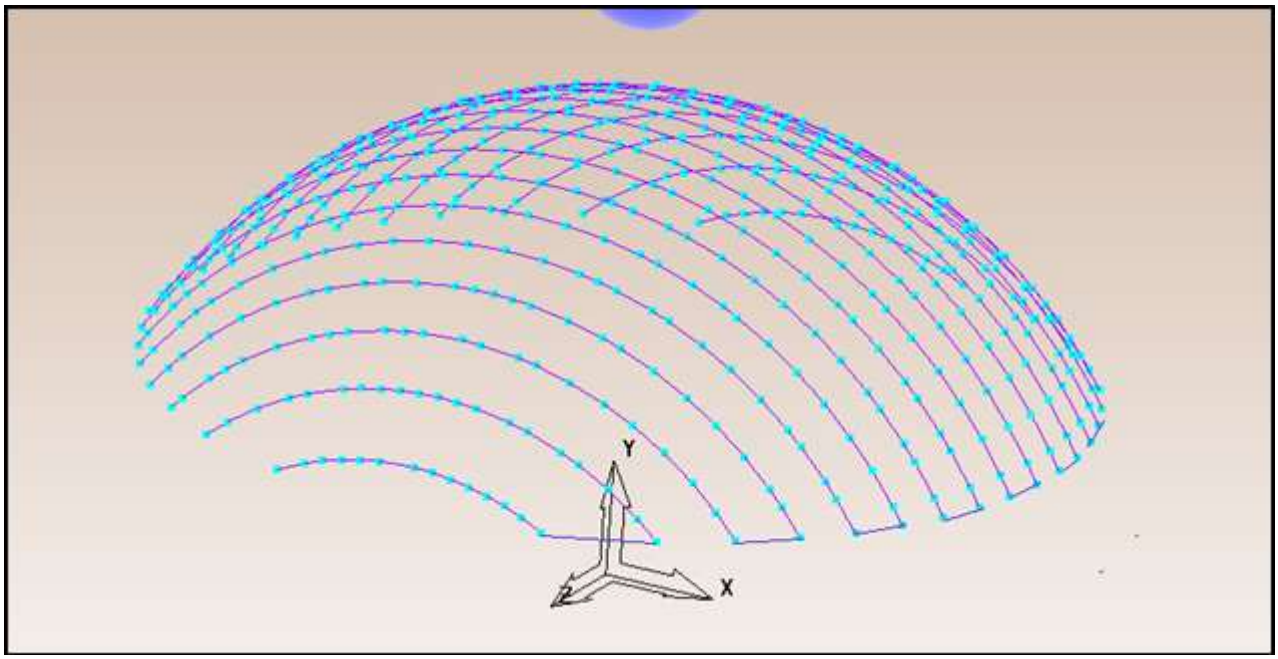


Рисунок 2.6 – Опорные точки центра концевой фрезы с сферической режущей частью

Следующий этап это ввод режимов резания. Режимы резания определяются либо расчетным способом, либо при использовании импортного инструмента. Значения режимов резания выбираем из каталогов выбранного инструмента, значения бокового шага выбирается исходя из значения шероховатости обрабатываемой поверхности, а именно от высоты гребешков между про-

ходами. Для рассматриваемого примера выбираем импортный инструмент, и режимы резания представлены на рисунке 2.7.

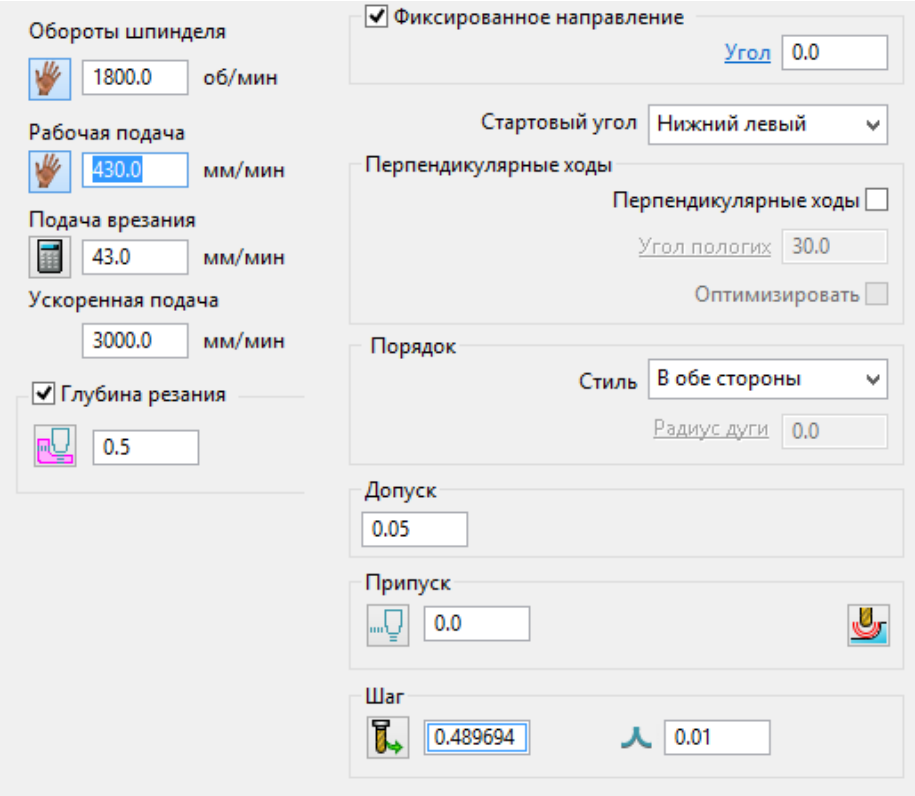


Рисунок 2.7 – Режимы резания

Построение траектории движения CAM система совершает в автоматизированном режиме. Траектория движения инструмента показана на рисунке 2.8.

После построения траектории движения инструмента, полученный файл переводят в файл управляющей программы станка с ЧПУ посредством постпроцессора. Постпроцессор – это модуль, преобразующий файл траектории движения инструмента и технологических команд, рассчитанный процессором CAM системы, в файл управляющей программы в строгом соответствие с требованием методики ручного программирования конкретного комплекса «станок – система с ЧПУ» [51].

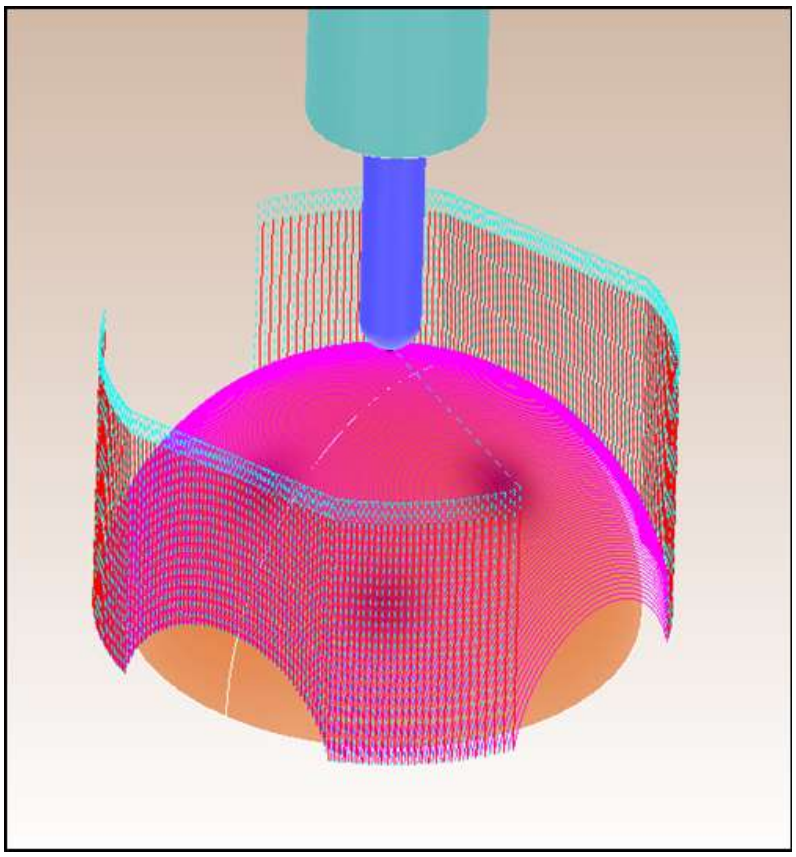


Рисунок 2.8 – Траектория движения инструмента

Классическая методика расчета управляющих программ не учитывает влияние технологических факторов в частности размерный износ режущего инструмента.

2.2 Доработка порядка расчета управляющих программ станков с ЧПУ

Доработанный порядок расчета управляющих программ представлена на рисунке 2.9. Доработка заключается в корректировки траектории движения инструмента в зависимости от размерного износа режущего инструмента. Но данная методика это идеальный случай, когда имеются права на доработку и изменение CAM системы. В рамках данной работы таких прав не предоставлено, в следствие этого корректировка траектории движения инструмента будет по готовым управляющим программам.

Рисунок 2.9 –Доработанный порядок расчета управляющих программ станков с ЧПУ

Для корректировки траектории движения инструмента требуется решить три задачи для каждого положения инструмента :

- расчет точек контакта исходной инструментальной поверхности фрезы и номинальной поверхности детали ;
- определение вектора смещения инструмента для корректировки износа ;
- вычисление величины смещения корректировки износ.

2.3 Разработка математической модели траектории движения инструмента учитывающий размерный износ инструмента

Расчет точек контакта исходной инструментальной поверхности фрезы и номинальной поверхности детали.

Расчет следует начинать с определения исходных данных. Исходные данные :

1) траектория инструмента из CAM системы, представленная множеством $\{F\}_{fk}$ точками с координатами $(x_{fk}; y_{fk}; z_{fk})$ и количеством их i_k;

2) поверхность детали в виде множества $\{D\}_{dj}$ точек с координатами $(x_{dj}; y_{dj}; z_{dj})$ и количеством их i_j;

3) данные по износу на разных участках фрезы в зависимости от пройденного расстояния, представленные множеством значений величин износа $\Delta R_i(L)$.

Построение траектории движения инструмента осуществляется эквидистантно к обрабатываемой детали, то условием контакта является минимальное расстояние от центра концевой фрезы с сферической режущей частью до поверхности инструмента. Для этого вычисляем расстояние для каждой точки множества $\{F\}_{fk}$ до всех точек на поверхности детали множества $\{D\}_{dj}$ (рисунок 2.10) по формуле (2.1).

$$d_{kj} = \sqrt{(X_{fk} - X_{dj})^2 + (Y_{fk} - Y_{dj})^2 + (Z_{fk} - Z_{dj})^2}, \qquad (2.1)$$

где d_{kj} — расстояние между точками множеств $\{F\}_{fk}$ и $\{D\}_{dj}$

$k = 1..fk$ количество точек множества $\{F\}_{fk}$

$j = 1..dj$ количество точек множества $\{D\}_{dj}$

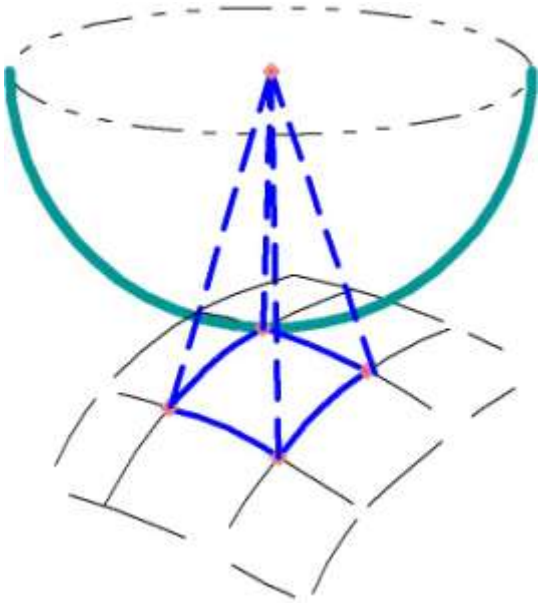


Рисунок 2.10 – Расстояния от опорной точки множества $\{F\}_{fk}$ до всех точек на поверхности детали множества $\{D\}_{dj}$

Для точки $(x_{fk}; y_{fk}; z_{fk})$ множества $\{F\}_{fk}$ определяем минимальное расстояние из всех точек множества $\{D\}_{dj}$:

$$d_{min} = \{d_{(k)(j-1)};\ d_{(k)(j-2)}; d_{(k)(j-3)}\ ; d_{(k)(j-4)}\ ; d_{(k)(j-n)}\}. \qquad (2.2)$$

Точке $(x_f; y_f; z_f)$ множества $\{F\}_{fk}$ соответствует точка $(x_{dminj}; y_{dminj}; z_{dminj})$ множества $\{D\}_{dj}$. Аналогичным образом определяем для каждой точки множества $\{F\}_{ij}$.

Определение вектора смещения инструмента для корректировки износа. После определения точки контакта для каждого положения инструмента, находим направление коррекции износа. Направлением коррекции износа будет являться, вектор нормали от центра сферической фрезы (т. O_1

$(X_{fk}\ ;\ Y_{fk}\ ;\ Z_{fk})$ до номинальной поверхности детали (т. M $(x_{dminj}; y_{dminj}; z_{dminj})$ представленный на рисунке 2.11.

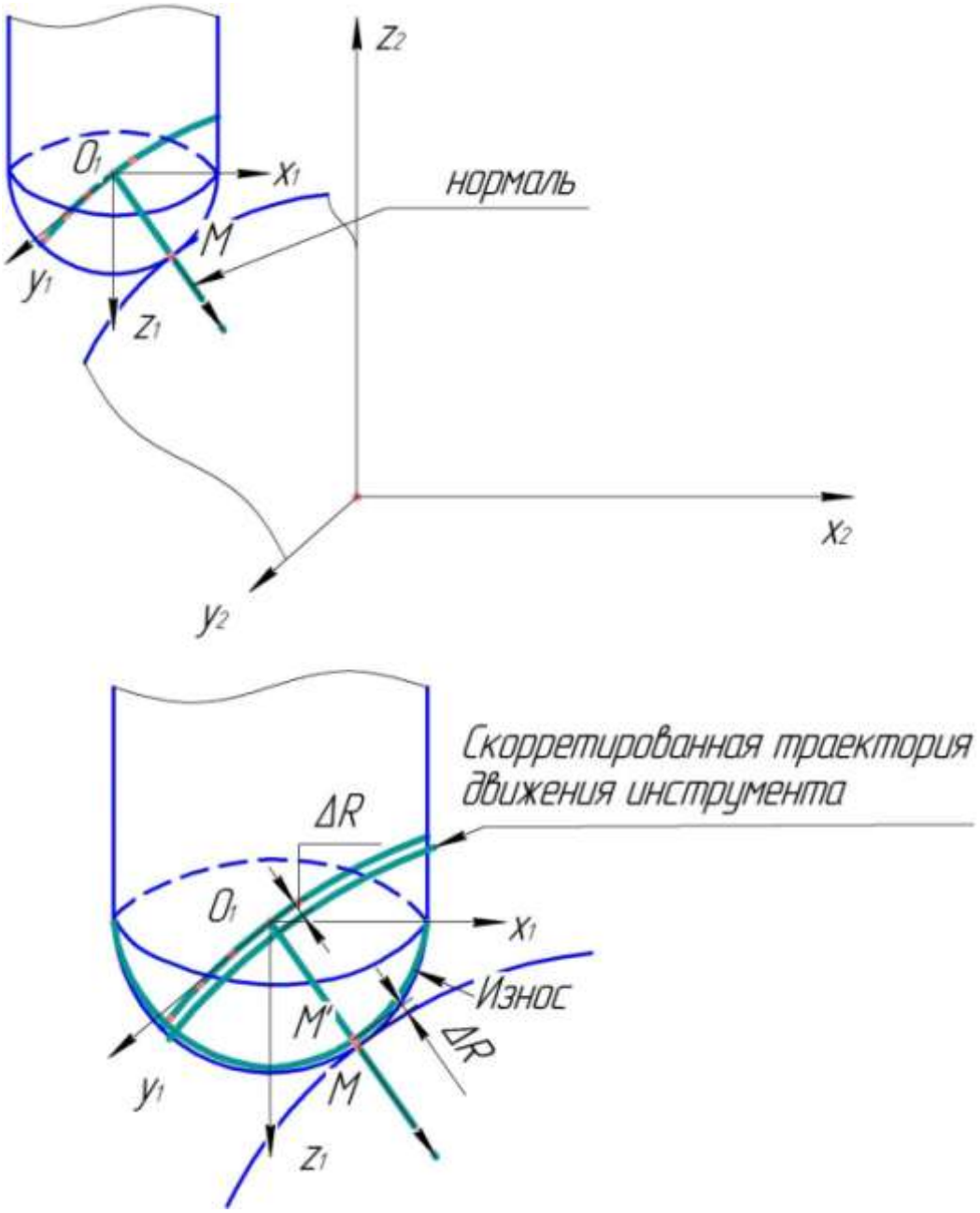


Рисунок 2.11 – Вектор нормали от центра сферической фрезы к номинальной поверхности детали

Длина вектора нормали от центра сферической фрезы на поверхность детали, равен радиусу сферической части фрезы, т.к траектория движения инструмента строилась эквидистантно поверхности детали :

$$[\,O_1M\,] = \sqrt{(X_{fk} - X_{dj})^2 + (Y_{fk} - Y_{dj})^2 + (Z_{fkk} - ; Z_{dj})^2} = R, \qquad (2.3)$$

где R – радиус фрезы, мм.

Величина коррекции на размерный износ определяется эмпирически, путем обработки предварительной заготовки с идентичными свойствами что и обрабатываемая деталь, по прохождении режущим инструментом некоторого пути в контакте с поверхностью детали, производится замер режущего лезвия и вычисляется значение размерного износа на пройденном пути, обозначение коррекции размерного износа от пройденного пути $\Delta R(L)$. Размерный износ по ИИП будет неравномерный, т.к при построчной обработке концевой фрезой с сферической части чем ближе к центру сферы контакт с деталью тем силы резания возрастают что увеличивает значение размерного износа по сравнению с периферией фрезы.

Для учета данного фактора, ИИП поверхность была разбита на три равных участка по оси Z, количество участков выбраны в соответствии с тем, что наиболее подвергаемая размерному износу при обработке, от центра и на 1/3 часть сферической фрезы, оставшиеся участки подвержены износу равномерно. Обозначим участки : 1 участок $[Z_1; Z_2]$, 2 участок $[Z_2; Z_3]$, 3 участок $[Z_3; Z_4]$ (см. рисунок 2.12). В управляющую программу значение коррекции вводиться только по декартовым осям (осям станка), в следствие этого определим величину коррекции по осям станка. Для этого определим проекцию вектора нормали на оси станка (рисунок 2.13).

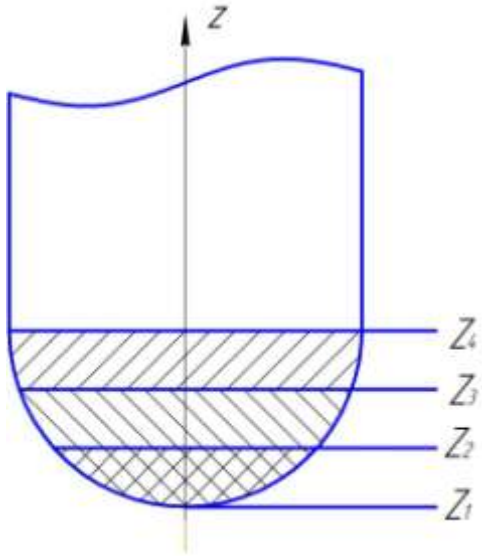


Рисунок 2.12 – Рассматриваемые участки фрезы

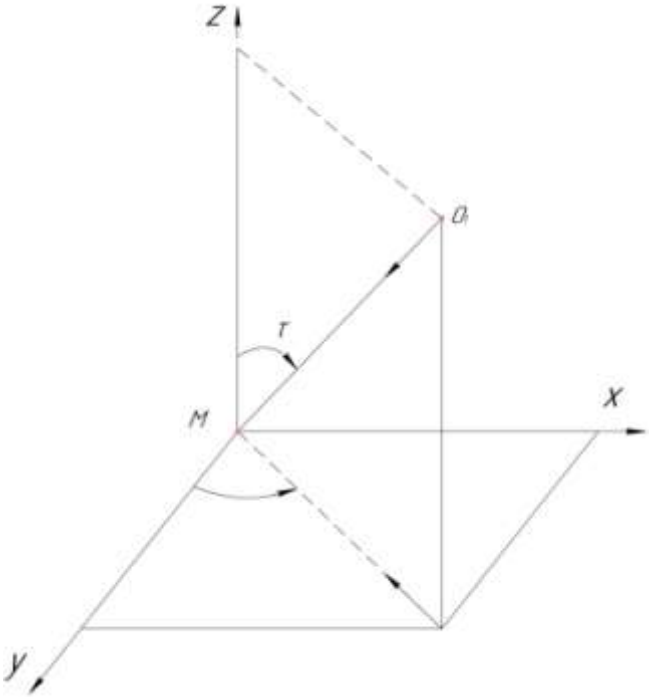


Рисунок 2.13 – Проекция вектора нормали на оси станка

Проекция вектора нормали на оси станка (см.рисунок 2.13) имеет вид :

$$X = \overline{O_1M} \cdot \sin\theta\cos\tau = R \cdot \sin\theta\cos\tau; \quad (2.4)$$

$$Y = \overline{O_1M} \cdot \sin\theta\sin\tau = R \cdot \sin\theta\sin\tau; \quad (2.5)$$

$$Z = \overline{O_1M} \cdot \cos\tau = R \cdot \cos\tau. \quad (2.6)$$

Определение координат коррекции на оси станка

$$\Delta X = (R - \Delta R(L)) \cdot \sin\theta\cos\tau; \quad (2.7)$$

$$\Delta Y = (R - \Delta R(L)) \cdot \sin\theta\sin\tau; \quad (2.8)$$

$$\Delta Z = (R - \Delta R(L)) \cdot \cos\tau. \quad (2.9)$$

После преобразований получаем значение коррекции для каждого из участков в проекции на оси станка :

$$\Delta X_{[z_i;z_{i+1}]} = \Delta R(L)_{[z_i;z_{i+1}]} \cdot \sin\theta\cos\tau; \quad (2.10)$$

$$\Delta Y_{[z_i;z_{i+1}]} = \Delta R(L)_{[z_i;z_{i+1}]} \cdot \sin\theta\sin\tau ; \quad (2.11)$$

$$\Delta Z_{[z_i;z_{i+1}]} = \Delta R(L)_{[z_i;z_{i+1}]} \cdot \cos\tau. \quad (2.12)$$

В данной главе был изложен порядок расчета управляющих программ станков с ЧПУ в CAM системе PowerMill, с описанием каждого шага при расчете. Разработан доработанный порядок расчета управляющих программ станков с ЧПУ, учитывающий влияние такого технологического фактора, как размерный износ инструменте. На основании расчета и корректировки траектории движения инструмента, была разработана математическая модель траектории движения инструмента учитывающий размерный износ инструмента. Следующим этапом станет разработка алгоритма на основе созданной математической модели с последующей разработкой компьютерной программы по описанию алгоритма.

3 РАЗРАБОТКА АЛГОРИТМА И КОМПЬЮТЕРНОЙ ПРОГРАММЫ ДЛЯ СТАНКОВ С ЧПУ, ОБЕСПЕЧИВАЮЩИЕ КОРРЕКЦИЮ ТРАЕКТОРИИ ДВИЖЕНИЯ ИНСТРУМЕНТА

3.1 Алгоритм обеспечивающий коррекцию траектории движения инструмента

Исходные данные алгоритма (вводимые данные) включают в себя :

- Траектория движения инструмента в виде набора команд языка программирования систем с числовым программным управлением регламентированным ГОСТ 20999-83 (G код). Построенная траектория движения инструмента должна быть разработана эквидистантно поверхности детали в CAM системе и состоять из геометрических элементов таких как отрезок прямой;
- Координаты узлов упорядоченной сетки на поверхности детали (рисунок 3.1), представленных в виде "Excel" таблицы. Диапазон частоты сетки конечных элементов следует назначать от 5 мкм до 10 мкм. Значения выбираются в соответствии с точностью позиционирования фрезерных станков с ЧПУ;

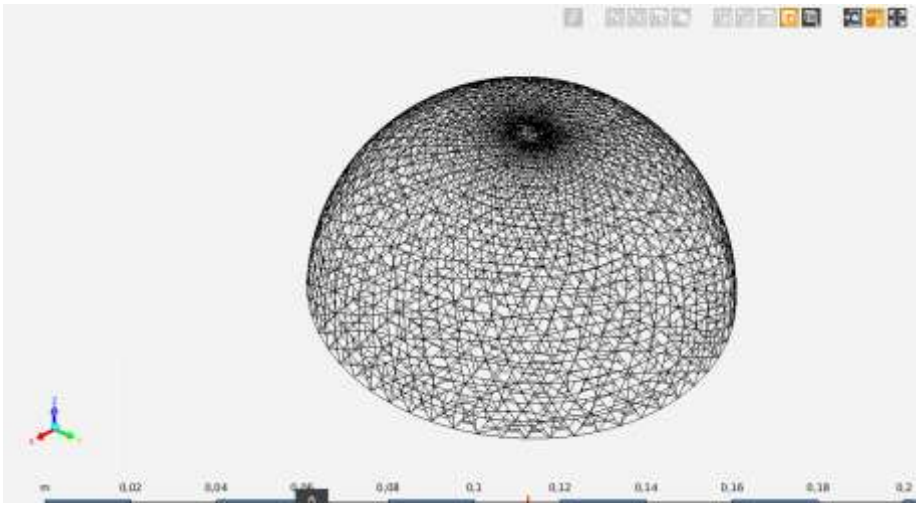


Рисунок 3.1 – Упорядоченная сетка узлов на поверхности детали

- Данные размерного износа концевой сферической фрезы в зависимости от пройденного расстояния на каждый рассматриваемый участок.

Таблица 3.1 – Множеством значений величин износа $\Delta R_i(L)$

L(мм)	1 участок (мм)	2 участок(мм)	3 участок(мм)
L_1	ΔR_1	ΔR_2	ΔR_3
L_2	$\Delta R_1'$	$\Delta R_2'$	$\Delta R_3'$
L_3	$\Delta R_1''$	$\Delta R_2'$	$\Delta R_3''$

Алгоритм разбит на подпрограммы. Общий вид алгоритма представлен на рисунке 3.2.

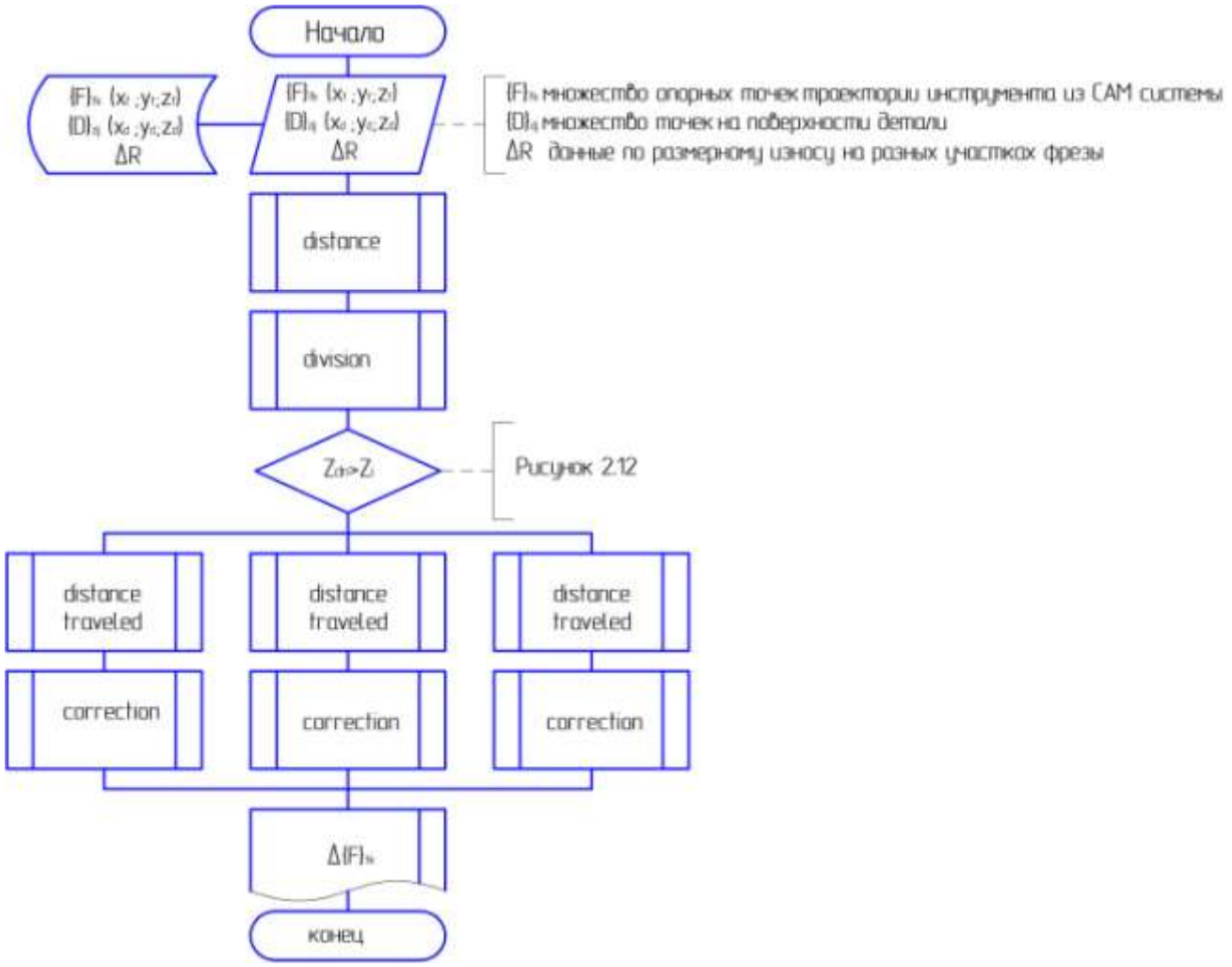


Рисунок 3.2 – Общий вид алгоритма обеспечивающий коррекцию траектории движения инструмента

Подпрограмма – это часть другой программы и удовлетворяющая требованиям языка программирования к структуре программы [52].

Подпрограмма «distance» вычисляющая точки контакта ИИП фрезы с НПД, представлена на рисунках 3.3 – 3.4.

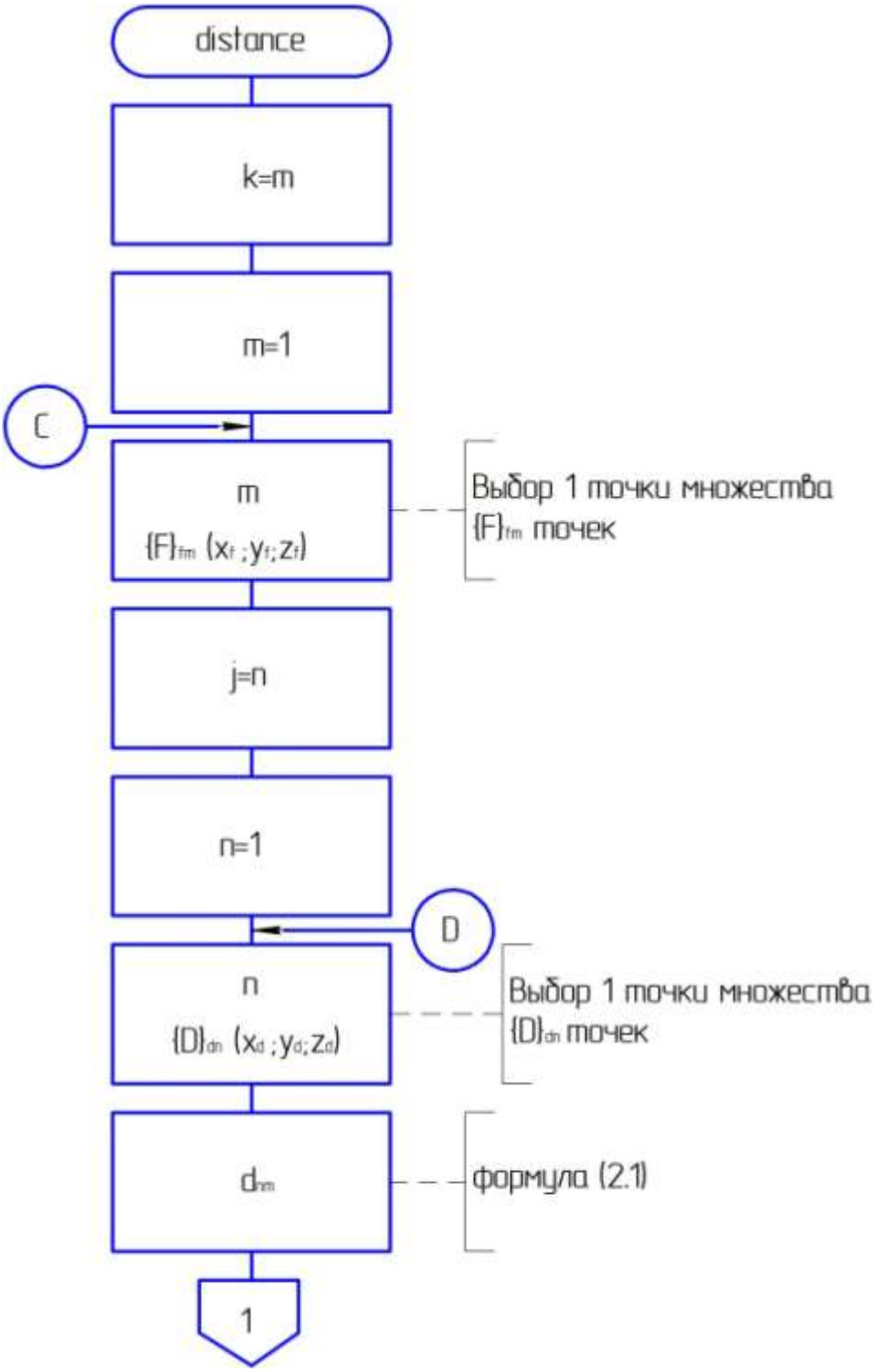


Рисунок 3.3 – Начало подпрограммы «distance»

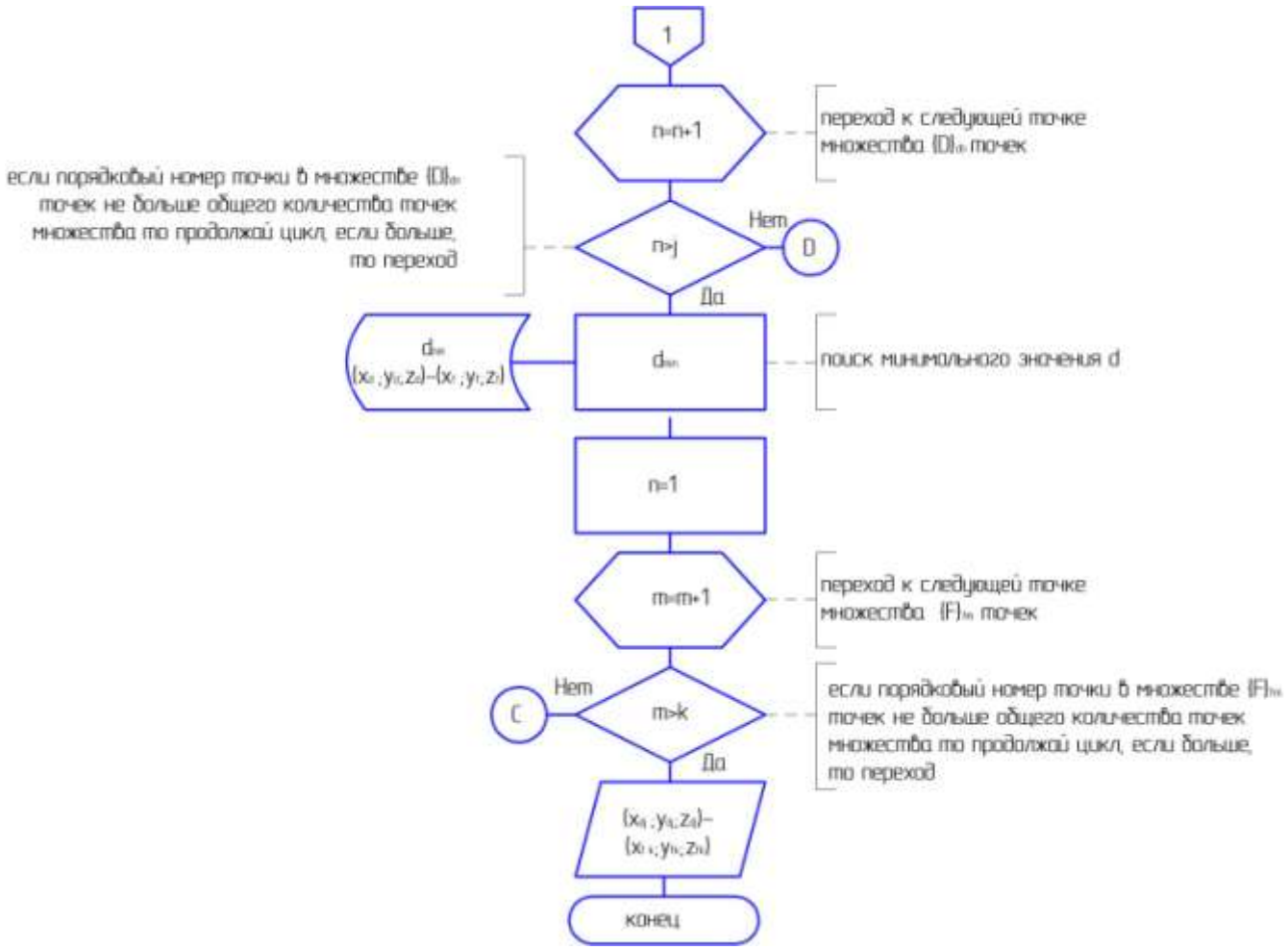


Рисунок 3.4 – Конец подпрограммы «distance»

Подпрограмма «division» выполняет деление траектории движения инструмента на три рассматриваемых участка. Подпрограмма «division» показана на рисунках 3.5 – 3.6.

Подпрограмма «distance traveled» вычисляет пройденный путь внутри каждого участка. Подпрограмма “distance traveled” представлена на рисунках 3.7 – 3.8.

Подпрограмма «correction» определяет значение коррекции в зависимости от пройденного пути внутри каждого участка. Подпрограмма «correction» представлена на рисунках 3.9 – 3.10.

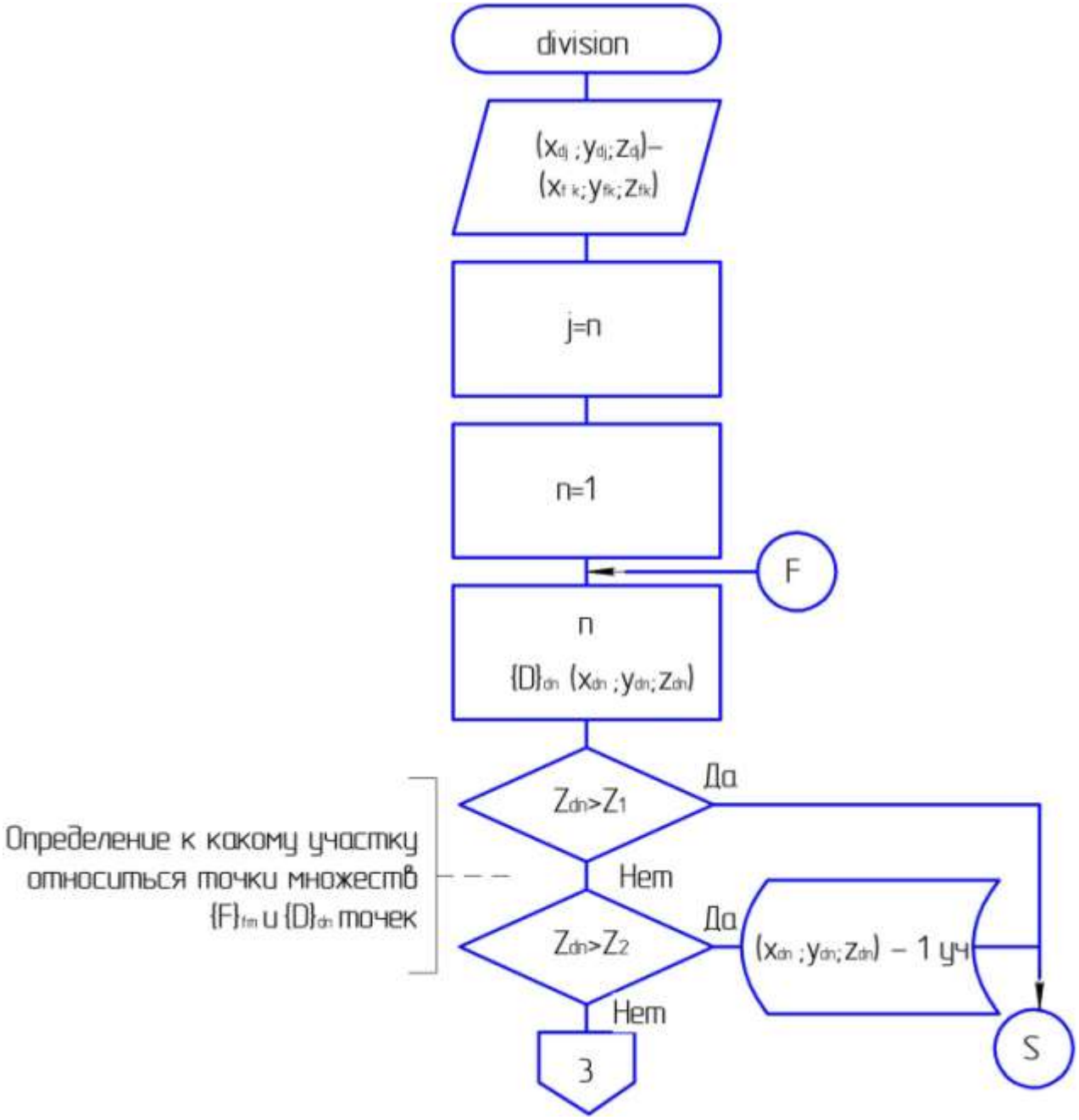


Рисунок 3.5 – Начало подпрограммы «division»

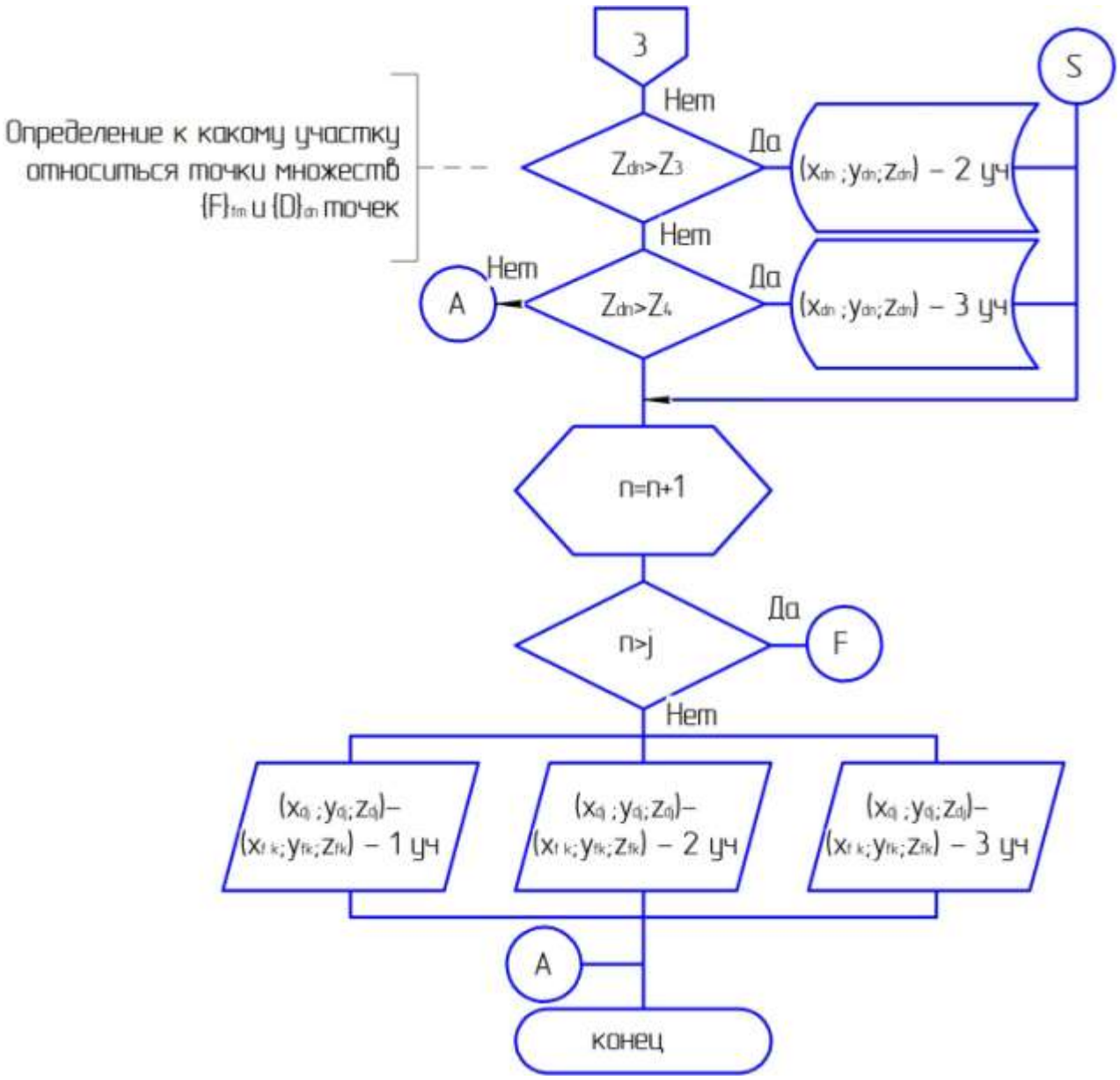


Рисунок 3.6 – Конец подпрограммы «division»

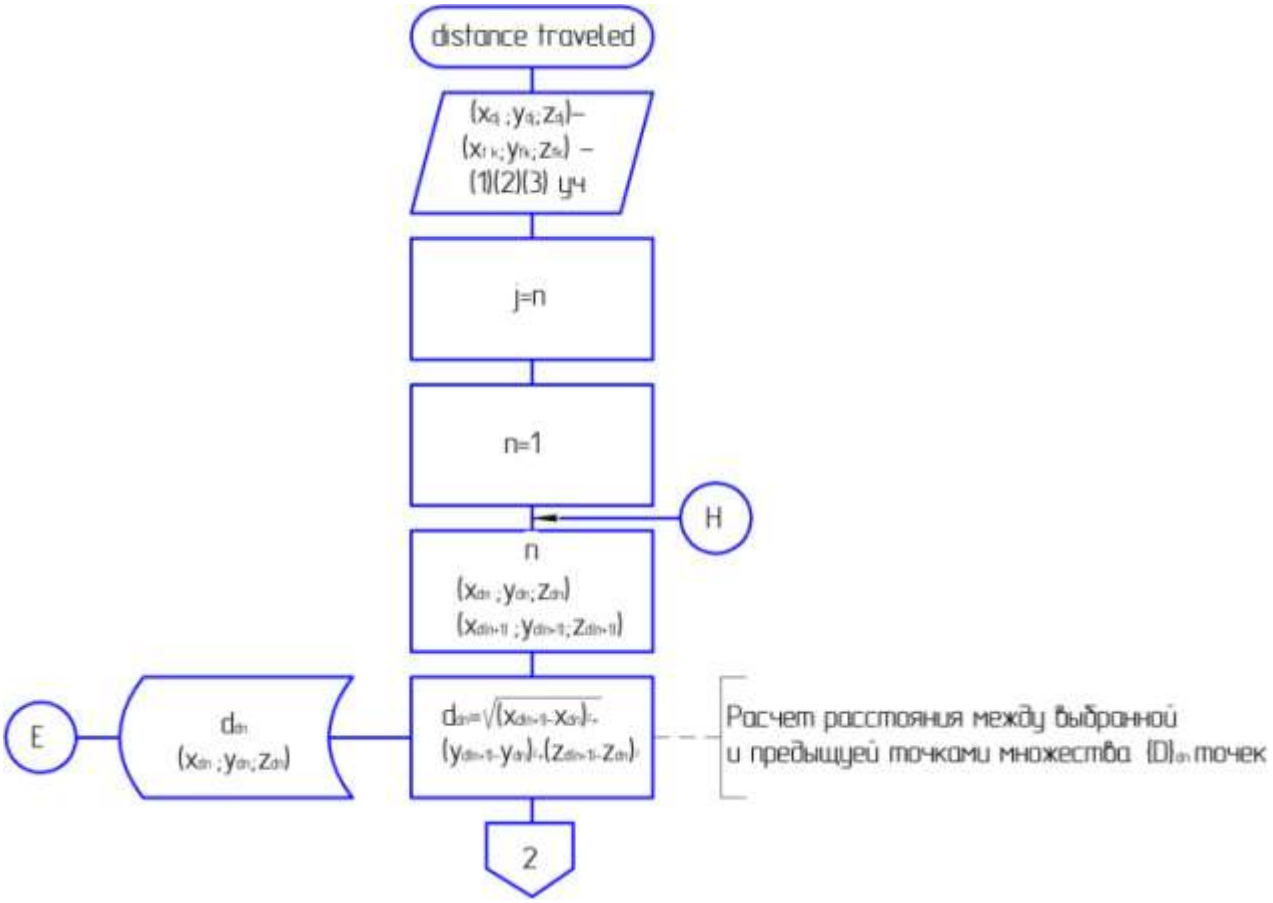


Рисунок 3.7 – Начало подпрограммы «distance traveled»

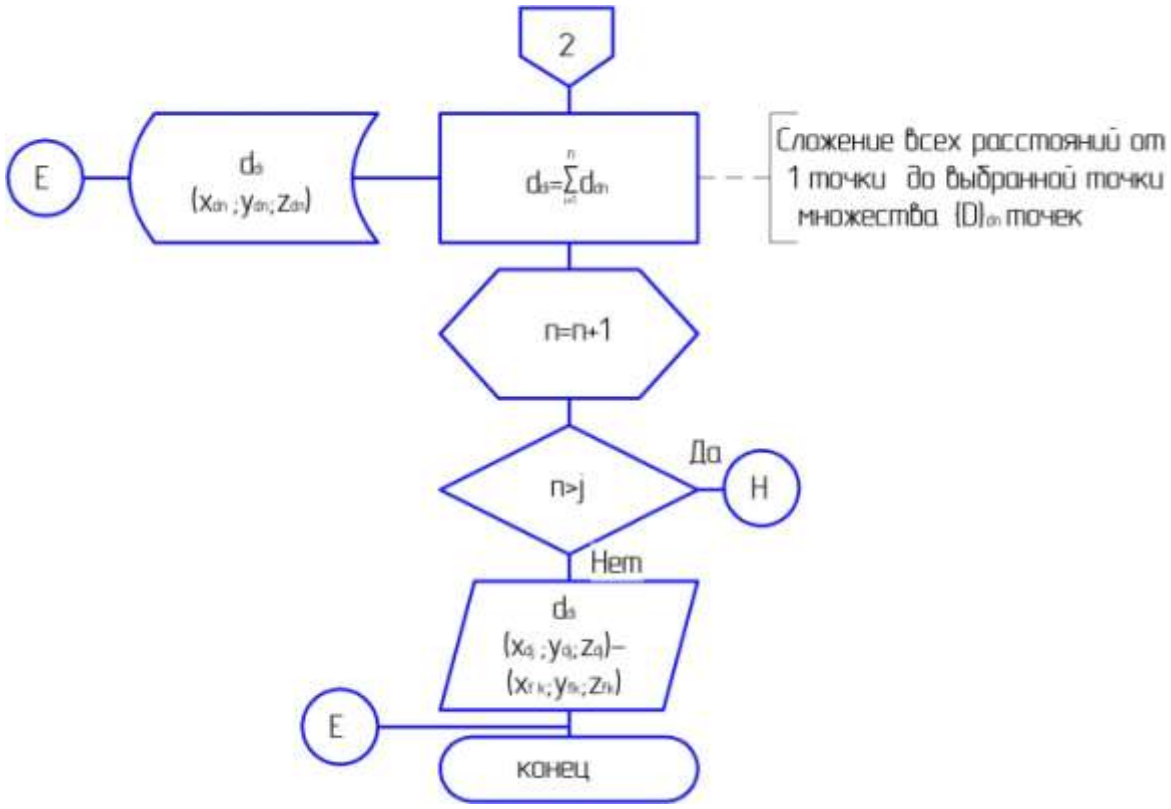


Рисунок 3.8 – Конец подпрограммы «distance traveled»

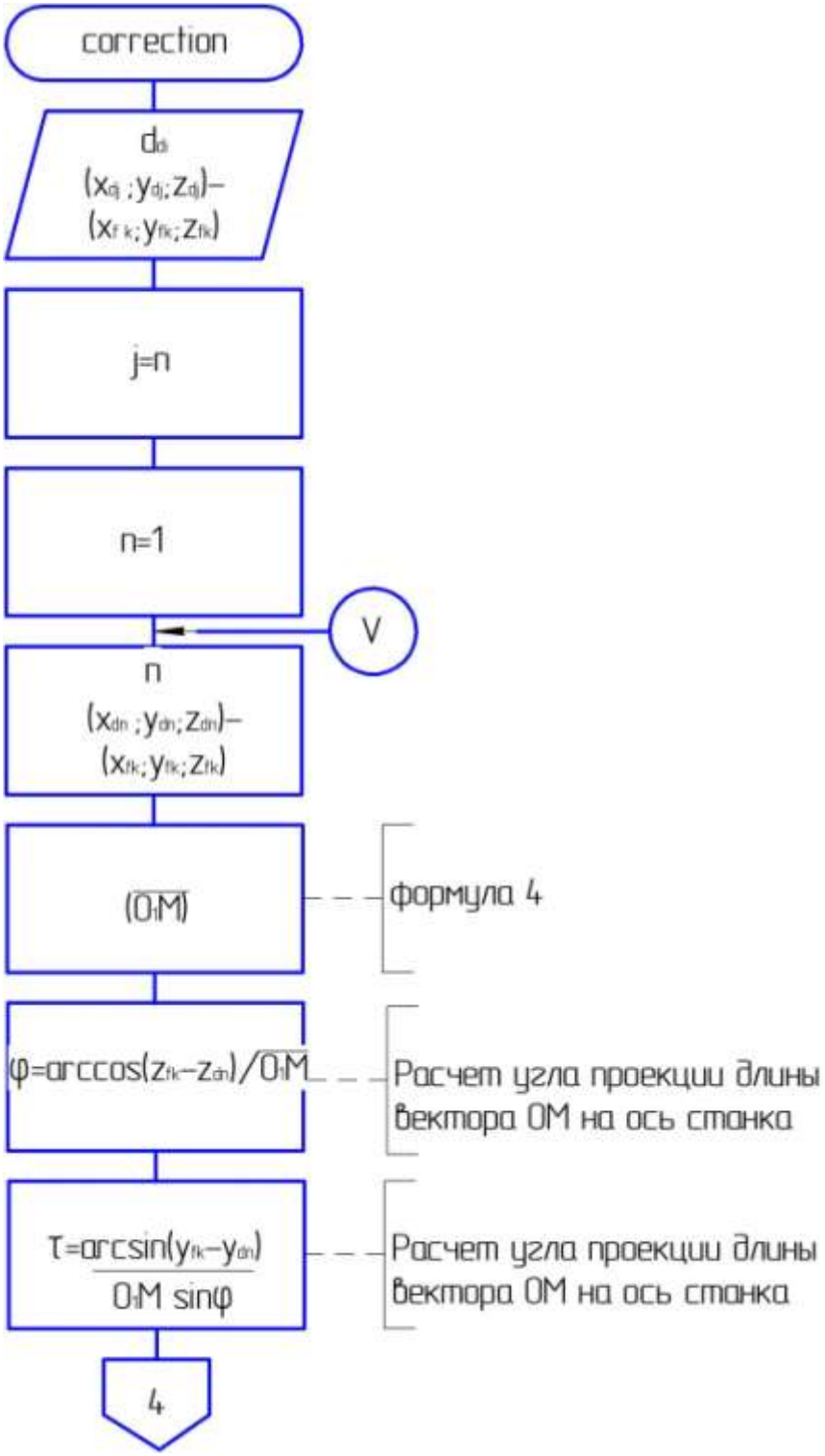


Рисунок 3.9 – Начало подпрограммы «correction»

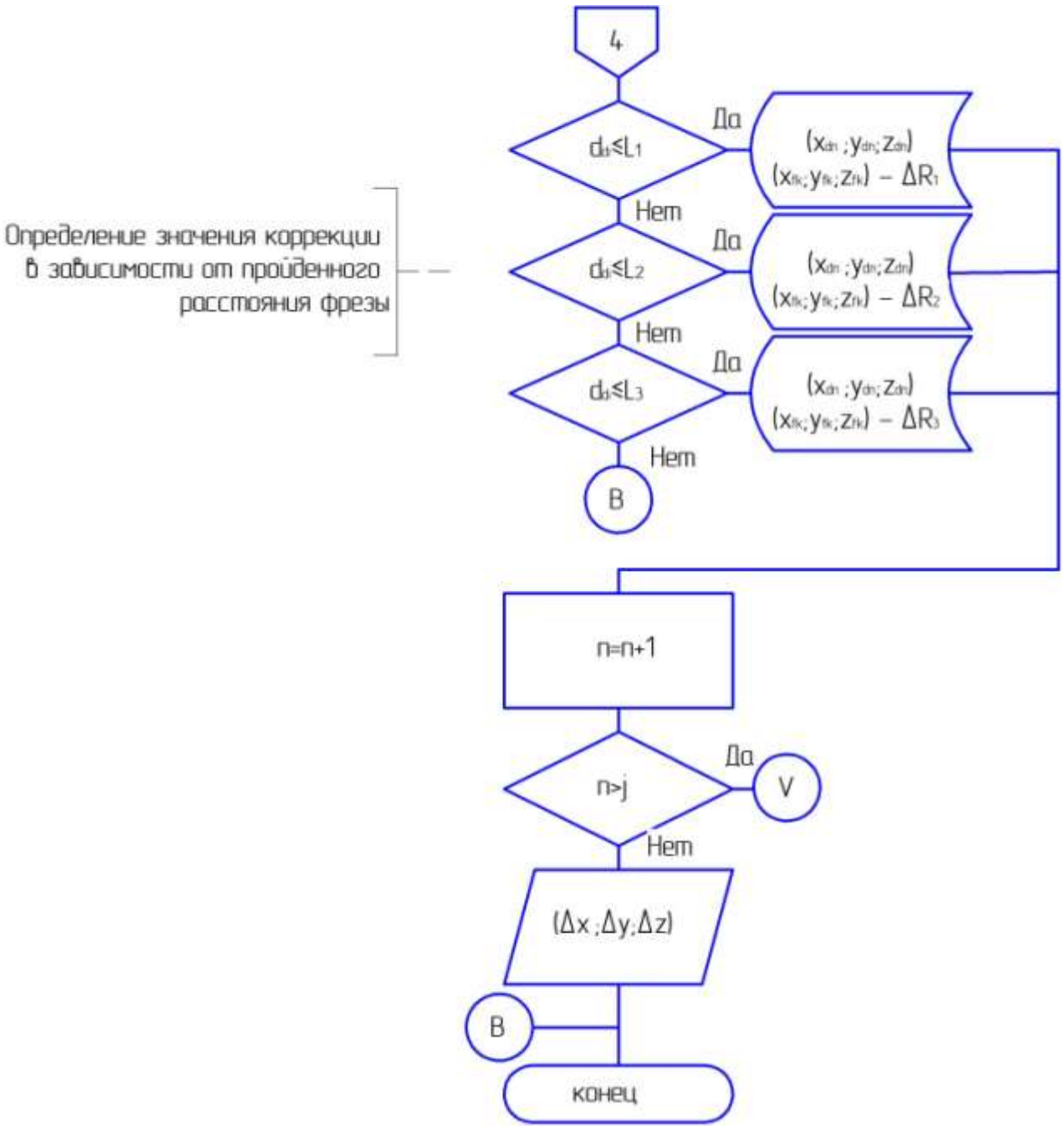


Рисунок 3.10 – Конец подпрограммы «correction»

3.2 Компьютерная программа для станков с ЧПУ, обеспечивающий коррекцию траектории движения инструмента

Компьютерная программа – это комбинация компьютерных инструкций и данных, позволяющая аппаратному обеспечению вычислительной системе выполнять вычисления или функции управления [ИСО 24765:2010].

Комбинация компьютерных инструкций описана в виде последовательности действий (алгоритма). Компьютерная программа разработана на основе применения языка программирования высокого уровня Java в среде разработки Eclipse.

Программный код относящийся к подпрограмме «distance»

```
public Point3D(long x, long y, long z) //начало подпрограммы «distance»
{
        this.x = x; //присвоение значениям точек по оси x , параметра x
        this.y = y; //присвоение значениям точек по оси y , параметра y
        this.z = z; //присвоение значениям точек по оси z , параметра z
}
@Override
public boolean equals(Object arg0) {
        if(arg0 instanceof Point3D)
        {
                Point3D p = (Point3D)arg0;
                if(this.x == p.getX() && this.y == p.getY() && this.z ==
p.getZ())  //условие на рисунке 3.4 "m=m+1"
                        return true;
        }
        return false;
@Override // запись результатов расчета
public int hashCode() {
        long result = this.x;
        result = result*7 + this.y;
```

```
        result = result*7 + this.z;
        return (int)result;
```

Программный код относящийся к подпрограмме «division»

```
        public boolean addPoint(Point3D point) //вызов  результатов
                                                подпрограммы  «distance»
        {
        return this.pointsCloud.add(point);
}
/**
 * @return
 */
public Set<Point3D> getPointsCloud() {
        return pointsCloud;
}
public void setPointsCloud(Set<Point3D> pointsCloud) {
        this.pointsCloud = pointsCloud;
        refreshTopAndBottomCoordinates();
        devideWearingZones(3);
}
public void refreshTopAndBottomCoordinates()
{
        if(pointsCloud!=null)
        {
```

```
                for(Point3D point: pointsCloud)
                {
                        if(point.getZ()>this.topZ) //Z_dn > Z_2
                                this.topZ=point.getZ();
                        if(point.getZ()<this.bottomZ)// Z_dn > Z_3
                                this.bottomZ=point.getZ();
                }
        }
}
public TreeMap<Integer, Long[]> getWearingZones() {
        return wearingZones; // разделение множества на участки
}
        public Map<Integer, Double> getWearingKoeffs() {
        return wearingKoeffs;
}
public void setWearingZones(TreeMap<Integer, Long[]> wearingZones) {
        this.wearingZones = wearingZones;
}
        public void setWearingZone(Integer zone, Long[] row) {
        this.wearingZones.replace(zone, row);
        calculateWearingKoeffs();
}
 * @param numberOfZones
public void devideWearingZones(int numberOfZones)
```

```
{
        long height = Math.round((((double)Math.abs(this.topZ-
this.bottomZ))/numberOfZones)); // расчет пройденного расстояния внутри каж-
дой зоны
        long layerTopCoordinate=this.bottomZ;
        long layerBottomCoordinate=this.bottomZ;
        TreeMap<Integer, Long[]> wearingZones = this.getWearingZones();
        if(wearingZones==null)
                wearingZones = new TreeMap<Integer, Long[]>();
        wearingZones.clear();
        int i;
        for(i=1; i<numberOfZones; i++)
        {
                layerTopCoordinate+=height;
                wearingZones.put(i, new Long[] {layerBottomCoordinate,
layerTopCoordinate, 0L, 0L});
                layerBottomCoordinate = layerTopCoordinate;
        }
        wearingZones.put(i, new Long[] {layerBottomCoordinate, this.topZ,
0L, 0L});
        this.setWearingZones(wearingZones);
        calculateWearingKoeffs();
}
```

Программный код относящийся к подпрограмме «distance traveled»

```
    public void calculateWearingKoeffs()
    {
        this.wearingKoeffs = new HashMap<Integer, Double>();
        for(Map.Entry<Integer, Long[]> entry:
this.getWearingZones().entrySet()) // формула 2.1 расчет расстояний
        {
            wearingKoeffs.put(entry.getKey(), ((dou-
ble)entry.getValue()[3])/entry.getValue()[2]);
        }
    }
        /**
     * @param z
     * @return
     */
    public int getNumberOfZoneByZ(long Z)
    {
        Long[] tempArray; // сложение расстояний
        for (Map.Entry<Integer, Long[]> entry : wearingZones.entrySet())
        {
            tempArray = entry.getValue();
        if (Z >= tempArray[0] && Z < tempArray[1]) // условие "n=n+1"
            {
                return entry.getKey();
            }
```

```
                if (Z == tempArray[1])
                {
                        return entry.getKey();
                }
        }
        return -1;
}
```

Программный код относящийся к подпрограмме «correction»

```
public void defineZoneForEachPoint() throws Exception
{
        if (this.contactPoints == null)
                rogressLog pl = new ProgressLog("Определяю зоны точек
контакта с поверхностью: ", contactPoints.size());
        this.pointZone = new ArrayList<Integer>(this.toolPath.size());
        for (Point3D point : contactPoints)
        {
        pointZone.add(this.surface.getNumberOfZoneByZ(point.getZ()));
                pl.tick();
        }
}
public void calculateVectorOfWearingForEachPoint() throws Exception
{
        if (this.contactPoints == null)
```

```
ProgressLog pl = new ProgressLog("Расчет векторов и величин износа для точек траектории: ", this.toolPath.size());
Point2D previous2DVectorOfWearing = new Point2D(0,0);
Point2D temp2DVector;
Point3D toolPathPoint;
Point3D contactPoint;
Point3D vectorOfContact; //вектор контакта
long wearedAmount=0L; //величина износа
vectorsOfWearing = new ArrayList<Point2D>(contactPoints.size());
vectorsOfContact = new ArrayList<Point3D>(contactPoints.size());
anglesOfContact = new ArrayList<Double>(contactPoints.size());
wearedAmounts = new ArrayList<Long>(contactPoints.size());
for (int i = 0; i < this.toolPath.size(); i++)
{
    pl.tick();
    if(idleToolPathPoints.contains(i) || idleToolPathPoints.contains(i-1)) //если эта или предыдущая точка - точка холостого хода, то не меняем вектор износа, а величину износа ставим 0 и выходим из цикла
    {
        vectorsOfWearing.add(new Point2D(0,0));
        vectorsOfContact.add(new Point3D(0,0,0));
        anglesOfContact.add(new Double("0"));
        wearedAmounts.add(0L);
        continue;
```

```
                }
                toolPathPoint = toolPath.get(i);
                contactPoint = contactPoints.get(i);
                wearedAmount =
Math.round(this.wearingKoeffs.get(this.pointZone.get(i))*distancesTraveled.get(i)*
1000); //находим величину износа в нанометрах
                wearedAmounts.add(wearedAmount);
                vectorOfContact = contactPoint.subtractVector(toolPathPoint);
//получаем вектор контакта, вычитая координаты инструмента из координат
точки контакта
                vectorsOfContact.add(vectorOfContact);

        temp2DVector=convert3Dto2D(vectorOfContact).resizeVector(wearedAmou
nt); //превращаем 3d вектор контакта в 2d, делаем его длину равную величине
износа
                anglesOfContact.add(temp2DVector.getAngleToAxelZ());
                previous2DVectorOfWearing = previ-
ous2DVectorOfWearing.addVector(temp2DVector); //прибавляем к вектору из-
носа прошлой точки
                vectorsOfWearing.add(previous2DVectorOfWearing);
            }
        * @throws Exception
        */
        public void calculateNewToolPath() throws Exception
        {
            if (vectorsOfContact == null)
```

```
            throw new Exception("Невозможно расчитать новую траекторию инструмента, т.к. еще не вычислены вектора контакта");
        if (vectorsOfWearing == null)
            throw new Exception("Невозможно расчитать новую траекторию инструмента, т.к. еще не вычислены вектора износа");
        ProgressLog pl = new ProgressLog("Расчет новой траектории: ", this.toolPath.size());
        modifiedToolPath = new ArrayList<Point3D>(this.toolPath.size());
        Point3D toolPathPoint; //точка траектории
        Point3D vectorOfContact; //вектор контакта
        Point2D vectorOfWearing; //вектор износа
        long x,y,projectionLength;
        for (int i = 0; i < this.toolPath.size(); i++)
        {
            toolPathPoint = toolPath.get(i);
            vectorOfContact = vectorsOfContact.get(i);
            vectorOfWearing = vectorsOfWearing.get(i);
                projectionLength = new Point2D(vectorOfContact.getX(),vectorOfContact.getY()).length(); //ищем длину проекции вектора контакта на плоскость XY

    x=Math.round((double)vectorOfWearing.getXY()*vectorOfContact.getX()/projectionLength/1000); //ищем координату x к проекции вектора

    y=Math.round((double)vectorOfWearing.getXY()*vectorOfContact.getY()/projectionLength/1000); //ищем координату y к проекции вектора
```

```
                modifiedToolPath.add(toolPathPoint.addVector(new Point3D(x,
y, vectorOfWearing.getZ()/1000))); //прибавляем к координате траектории по-
вернутый вектор, тем самым получив смещенную точку траектории
                pl.tick(); //выход
            }
        }
```

Полный объем программного кода представлен в приложении А.

Графическое окно загрузки входных параметров компьютерной программы представлено на рисунке 3.11. Сообщения информационного окна показаны на рисунке 3.12. Графическое окно выгрузки полученных результатов представлено на рисунке 3.13.

Таким образом был разработан алгоритм и компьютерная программа для фрезерных станков с ЧПУ. Созданная компьютерная программа на основе алгоритма позволяет динамически корректировать траекторию движения инструмента в зависимости от пройденной длины, и значением износа на разных участках фрезы. Следующий этап работы, экспериментальная проверка созданной компьютерной программы.

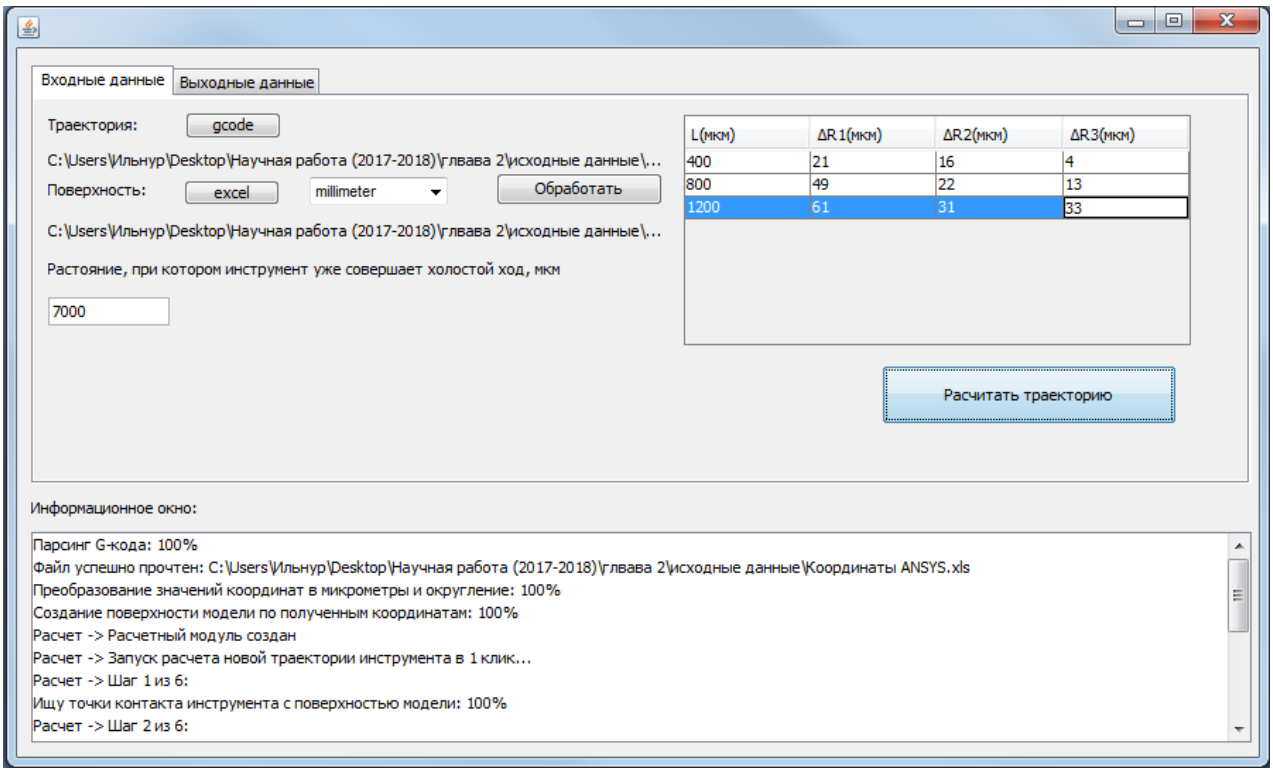


Рисунок 3.11 – Графическое окно загрузки входных параметров компьютерной программы

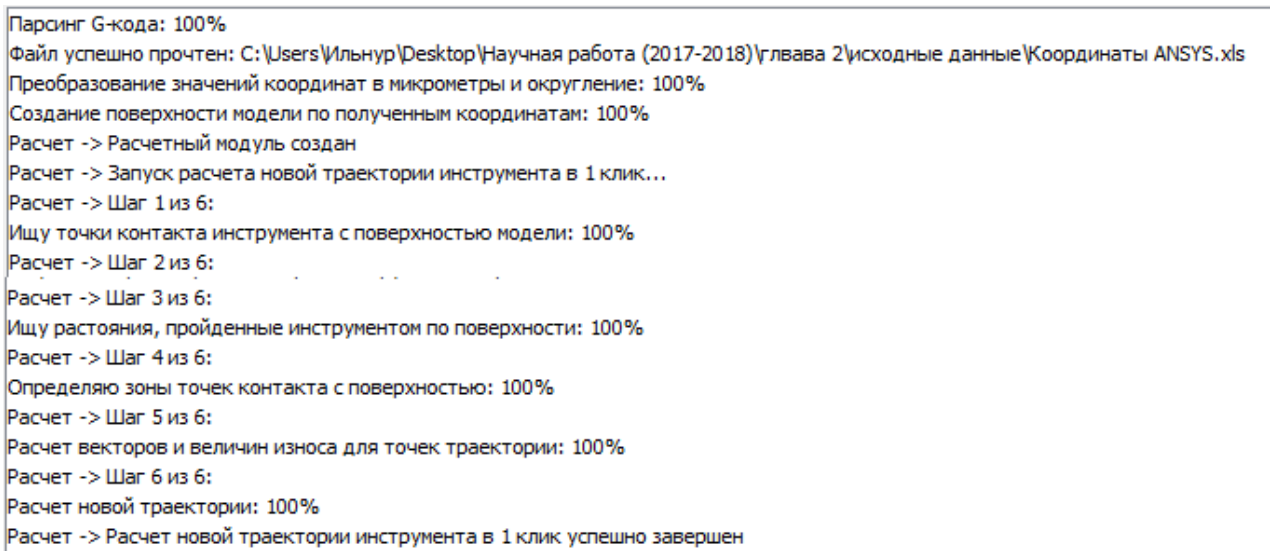


Рисунок 3.12 – Сообщения информационного окна

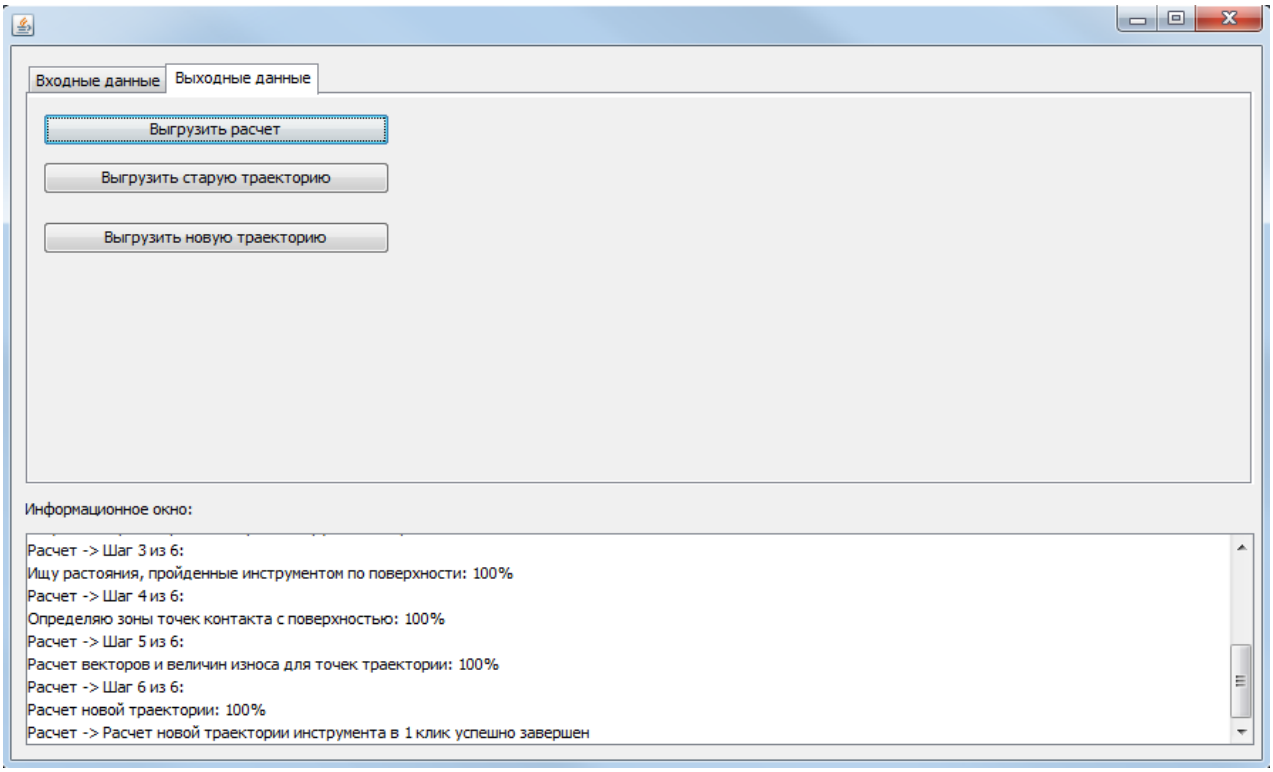


Рисунок 3.13 – Графическое окно выгрузки полученных результатов

4 ЭКСПЕРИМЕНТАЛЬНАЯ ВЕРИФИКАЦИЯ ПОЛУЧЕННОЙ КОМПЬЮТЕРНОЙ ПРОГРАММЫ

4.1 Цель и задачи проведения эксперимента

Цель исследования – экспериментально проверить работу созданной компьютерной программы.

Задачи исследования :

- установить значения размерного износа в зависимости от пройденной длины;
- сравнение результатов работы созданной компьютерной программы с предыдущими результатами полученными из CAM системы.

4.2 Методика проведения эксперимента

4.2.1 Применяемый инструмент и оборудование используемое в эксперименте :

- Режущий инструмент сферическая концевая 4х зубая фреза SBE 4160T компании Taegu Tec (рисунок 4.1) ;

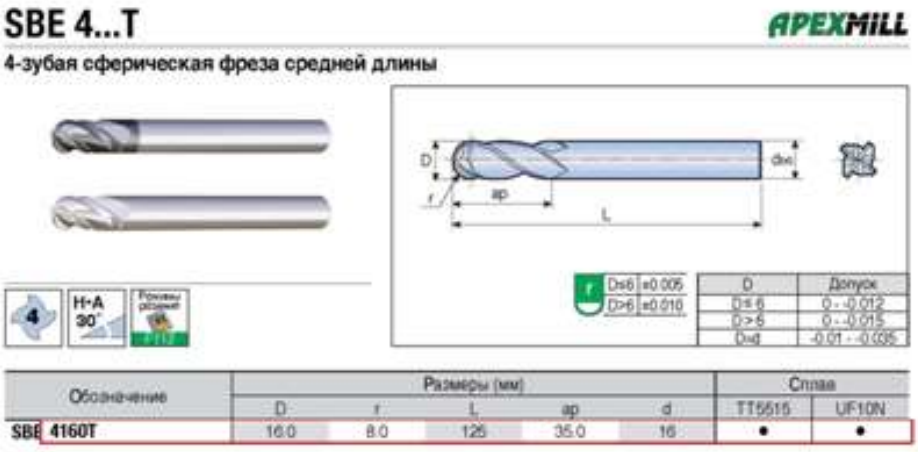


Рисунок 4.1 – Геометрические параметры режущего инструмента обрабатываемый заготовки

- Высокоточный патрон для зажима инструмента Force 63-32 D 'Andrea, данная модель практически исключает биение инструмента в патроне

на скоростях выше 3000об/мин, представлен на рисунке 4.2. Инструмент в патроне показан на рисунке 4.3 ;

Рисунок 4.2 – Высокоточный цанговый патрон для зажима инструмента

Рисунок 4.3 – Инструмент в цанговом патроне

- Вертикально-фрезерный 3х осевой станок Kitamura mycenter 3X, представлен на рисунке 4.4 основными необходимыми характеристиками данного станка это длительность хода по осям X - 760, Y - 510, Z - 510 мм , точ-

ность позиционирования по осям X и Y составляет 0.002 мм, по оси Z 0.003 мм ;

- механические прецизионные тиски серии «ТС» фирмы OML, погрешность при перемещении до 125 мм, составляет 1,2 мкм, представлены на рисунке 4.5 ;
- прибор автоматической настройки инструмента вне станка фирмы Elbo Controlli представлен на рисунке 4.7 ;
- индикатор рычажно-зубчатый ИРБ с ценой деления 1мкм представлен на рисунке 4.6 ;
- оптический датчик OMP40-2 компании renishaw представлен на рисунке 4.8. Точностные характеристики применяемого датчика ,погрешности по осям 1 мкм, повторяемость 1 мкм ;

Рисунок 4.4 – Фрезерный 3х осевой станок

Рисунок 4.5 – Механические прецизионные тиски серии «ТС» фирмы OML

Рисунок 4.6 – Индикатор ИРБ

Рисунок 4.7 – Прибор настройки инструмента вне станка фирмы Elbo Controlli

Рисунок 4.8 – Датчик OMP40-2 Renishaw

4.2.2 Значения размерного износа в зависимости от пройденной длины

Значения размерного износа определяются только эмпирически. В качестве обрабатываемой детали будет «Мастер» деталь (рисунок 4.9). «Мастер» деталь представляет собой заготовку идентичную по химическим и физическим свойствам что и предполагаемая партия деталей. В данном исследовании марка материла ХН50ВМТЮ ГОСТ 2590-88 закаленную до твердости 54 HRC.

Установление значений размерного износа необходимо начать с определения максимального значения затупления инструмента и его пройденного пути в процессе обработки. Определение максимального значения требуется для определения величины пути при котором произойдет смена инструмента на инструмент – дублер.

Методика замера значений размерного износа от пройденной длины и максимальной длины до переточки одинакова с некоторыми дополнениями.

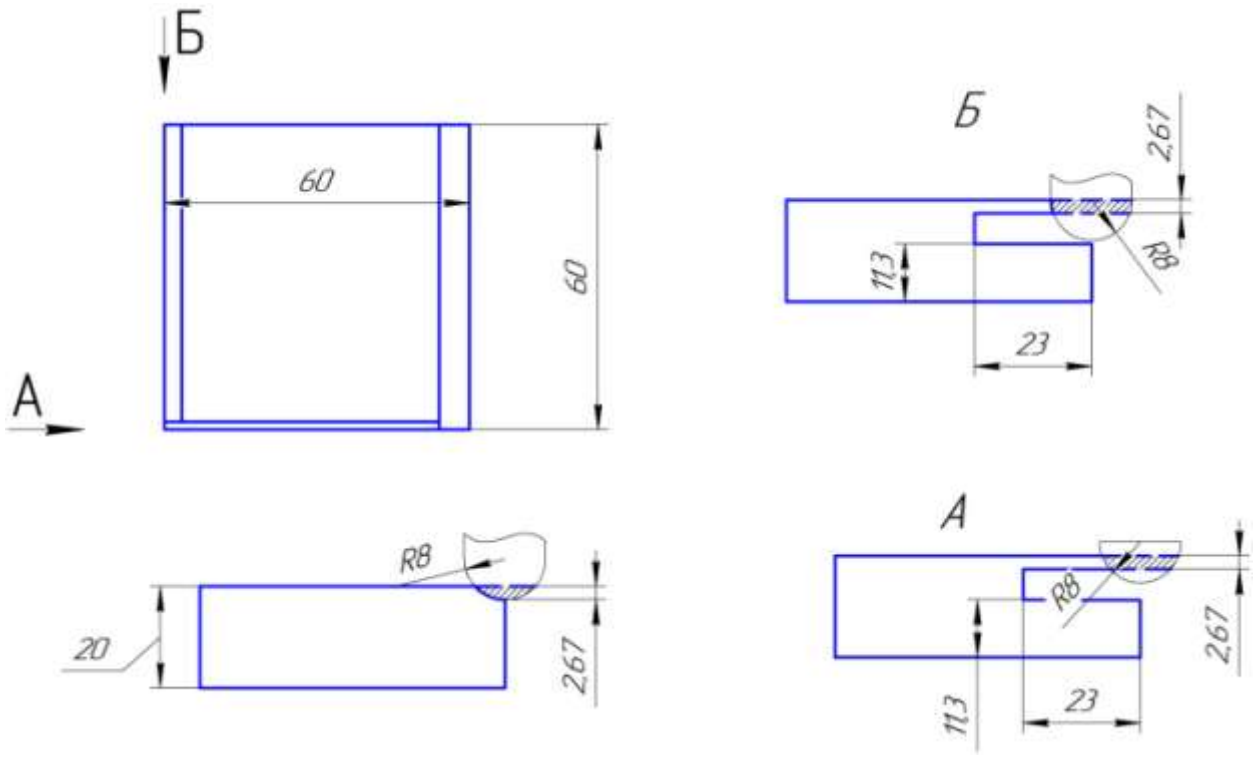


Рисунок 4.9 – Чертеж «мастер» детали

Порядок определения максимальной длины до переточки для каждого участка фрезы :

- установить «мастер» деталь в приспособлении ;
- установить режущий инструмент в цанговом патроне;
- определить рабочую систему координат ;
- установить режущий инструмент с цанговым патроном в приборе настройки инструмента вне станка , рисунок 4.3 ;
- поворачивая инструмент вокруг оси найти положение при котором значение радиуса будет совпадать до точности 0.03 мм, рисунок 4.10;
- нанести на маркером отметку положения на инструмент и прибор;
- выбрать произвольную точку в пределах границ участков фрезы, нанести курсор до пересечении с режущей кромкой и записать значения на экране фиксируя положении курсора, рисунок 4.11 ;
- установить в цанговый патрон в шпиндель станка ;
- обработать «мастер» деталь заданным участком фрезы по программе представленной ниже ;

- снять цанговый патрон и установить в приборе настройки инструмента вне станка и выставить положение инструмента по отметке сделанной ранее ;
- перемещая курсор в радиальном направлении до пересечения с режущей кромкой, записать полученные значения.

Рисунок 4.10 – Значения радиуса инструмента на приборе настройки инструмента вне станка

Рисунок 4.11 – Координаты произвольной точки первого участка относительно державки до обработки

Управляющая программа обработки «мастер» детали для системы fanuc oi.

Таблица 4.1 – Управляющая программа обработки «мастер» детали

G10X30.Y30.Z20	----------[смещение рабочей системы координат (СК)]
T01M6	----------[вызов инструмента]
G90G54	----------[рабочая СК, абсолютная СК]
G0X0.Z0	----------[ускоренное перемещение в координаты X0, Y0]
G43Z50.H01D01	----------[корректор на вылет и радиус инструмента]
#1=37.	---------[значение переменной 1, определяющая значение подхода инструмента]
#2=30.	-----[значение переменной 2, определяющая значение по оси выполняемого размера]
#3=2	-----[значение переменной 3, определяющая значение по оси выполняемого размера]
#4=1	----------[значение переменной 4, обозначающая номер участка]
#5=#3*#4	--------[значение переменной 5, обозначающий величину по оси Z в зависимости от выбранного участка]
N10	----------[начало цикла]
#6=0.2	--------[значение переменной 6, обозначающий величину бокового шага фрезерования]
#7=6	----------[значение переменной 7, обозначающий радиус режущего инструмента]
#8=#2+7.	----------[значение переменной 8, рассчитывающая значение отхода инструмента]
G0X#8Y-#1.	[ускоренное перемещение в координаты определяющая значением переменной 8 по оси X, переменной 1 по оси Y]
Z-#5	[ускоренное перемещение в координаты определяющая значением переменной 5 по оси Z]
G1G41X#2F350	---[линейное перемещение в координаты определяющая значением переменной 2 по оси X с включением коррекции на радиус слева от инструмента]
Y#1F150	---[линейное перемещение в координаты определяющей значением переменной 1 по оси Y]
G40X#8	---[линейное перемещение в координаты определяющей значением переменной 8 по оси X с отключением коррекции на радиус инструмента]
#2=#2-#3	-[пересчет значения переменной 2 при каждой итерации]
IF[#2LE5]GOTO10	----------[условие “пока значение переменной 2 меньше значения 5 выполняй цикл”]
G0Z10.	----------[линейное перемещение по оси Z]
M5M9	------[остановка вращения шпинделя, отключение СОЖ]
G91G28Z0.Y0.	----------[перемещение в референтную позицию станка]

Для разных участков необходимо установить значение переменной 4 в зависимости от выбранного участка, участки фрезы представлены на рисунке 2.12. А также развернуть систему координат путем добавления кода G68 R90 в начало управляющей программы.

Определение предельного состояния режущего лезвия производится при разных пройденных величинах пути, по качеству обрабатываемой поверхности. При условии, что значение шероховатости получаемой поверхности выше, чем значение шероховатости данное в чертеже, то инструмент заменяется на дублер.

Расчет пройденного пути фрезы рассчитывается из значения переменной 2 по формуле 4.1

$$\frac{(\#2_{\text{нач}} - \#2)}{\#6} \cdot (\#2_{\text{нач}} \cdot 2) = L_{\text{общ}}. \quad (4.1)$$

Режимы резания подбирались по каталогу, к инструменту для обрабатываемого материла, для обработки ХН50ВМТЮ ГОСТ 2590-88 закаленную до твердости 54 HRC рекомендуют скорость резания равную 40 мм/мин, подачу 0,05 мм/зуб.

Измерения произвольной точки относительно шпинделя станка для каждого участка на поверхности режущего лезвия фрезы для первой фрезы (таблица 4.2).

Таблица 4.2 – Измерение размерного износа первой фрезы

Участок фрезы		Z, мм	X, мм	Значение износа, мм	Пройденный путь, мм
1 участок	До обработки	139	7,75	0,140	840
	После		7,61		
2 участок	До обработки	137,5	9,952	0,092	
	После		9,86		
3 участок	До обработки	136	10,996	0,015	
	После		10,98		

Измерения произвольной точки относительно шпинделя станка для каждого участка на поверхности режущего лезвия фрезы для второй фрезы (таблица 4.3).

Таблица 4.3 – Измерение размерного износа второй фрезы

Участок фрезы		Z, мм	X, мм	Значение износа, мм	Пройденный путь, мм
1 участок	До обработки	140	5,588	0,54	1400
	После		5,048		
2 участок	До обработки	136,5	7,998	0,084	
	После		7,914		
3 участок	До обработки	135,5	10,194	0,052	
	После		10,142		

Измерения произвольной точки относительно шпинделя станка для каждого участка на поверхности режущего лезвия фрезы для третьей фрезы (таблица 4.4).

Таблица 4.4 – Измерение размерного износа третьей фрезы

Участок фрезы		Z, мм	X, мм	Значение износа, мм	Пройденный путь, мм
1 участок	До обработки	139,5	6,868	0,256	1400
	После		6,61		
2 участок	До обработки	137,3	9,376	0,15	
	После		9,226		
3 участок	До обработки	135,7	10,726	0,009	
	После		10,717		

При анализе получаемых поверхностей после обработки тремя фрезами, было установлено, что оптимальное значение пройденного пути равно

1200 мм. При данном значении размерного износа инструмент выполняет требования по шероховатости обрабатываемой поверхности.

Порядок замера значений размерного износа аналогичен порядку определения максимальной длины до переточки. На рисунках 4.12–4.17 представлены замеры значений размерного износа на разных участках фрезы.

Рисунок 4.12 – Координаты произвольной точки первого участка относительно державки до обработки

Рисунок 4.13 – Координаты произвольной точки первого участка относительно державки после прохождения 400 мм

Рисунок 4.14 – Координаты произвольной точки второго участка относительно державки до обработки

Рисунок 4.15 – Координаты произвольной точки второго участка относительно державки после прохождения 400 мм

Рисунок 4.16 – Координаты произвольной точки третьего участка относительно державки до обработки

Рисунок 4.17 – Координаты произвольной точки второго участка

Таблица № 4.5 – Данные по размерному износу $\Delta R_i(L)$ по оси X в разных сечениях

Расстояние, мм	1 участок, мкм	2 участок, мкм	3 участок, мкм
400	21	16	4
800	49	22	13
1200	61	31	33

4.3 Сравнение результатов работы созданной компьютерной программ с результатами полученных из CAM систем

Проверка компьютерной программы осуществляется обработкой двух идентичных заготовок по размерам и механическим свойствам. В эксперименте, применяются две траектории движения инструмента, одна без использования разработанной программы, вторая с использованием компьютерной программы. Эксперимента проводился в одинаковых условиях.

Обрабатываемая деталь представляет собой пресс-форму из материала ХН50ВМТЮ ГОСТ 2590-88 закаленную до твердости 54 HRC. Обработки на фрезерном станке подлежит поверхность поверхности полусфера R32 с допуском (-0.01 мм), выдерживая погрешность формы в пределах 0,01мм. Обрабатываемая деталь представлена на рисунке 4.19. Заготовка с припуском по сферической поверхности. Заготовка представлена на рисунке 4.20.

Входными параметрами экспериментальной проверки являются :

- Обрабатываемая деталь (рисунок 4.18);

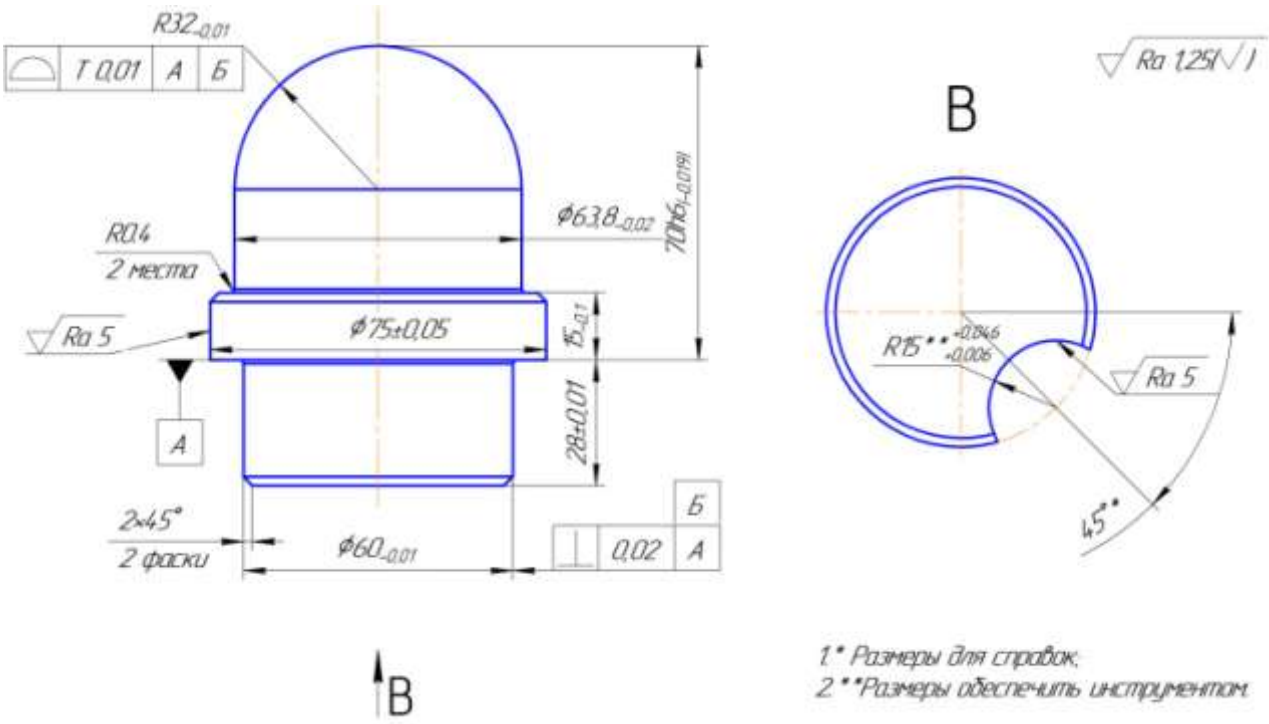


Рисунок 4.18 – Эскиз обрабатываемой детали

- Заготовка (рисунок 4.19);

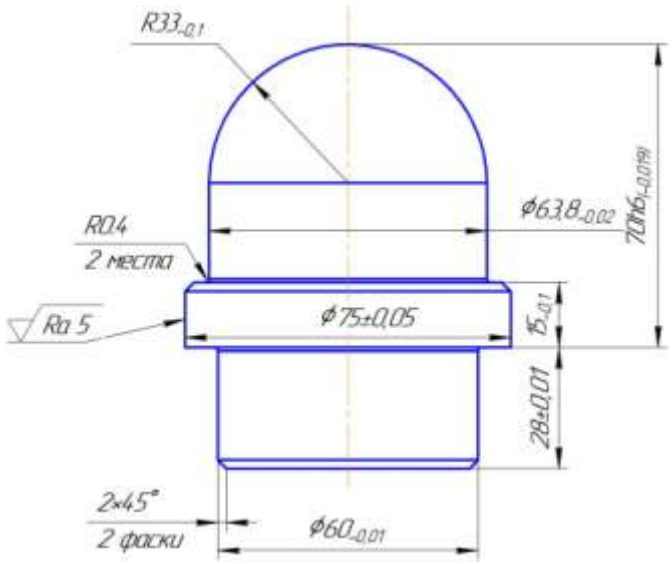


Рисунок 4.19 – Эскиз заготовки до обработки

Методика проведения эксперимента :

• устанавливаем приспособление на стол станка, выставив по параллельности по одной их губок как показано на рисунках 4.20–4.23 относительно X, Z осей станка не более 0,01 мм индикатором ИРБ;

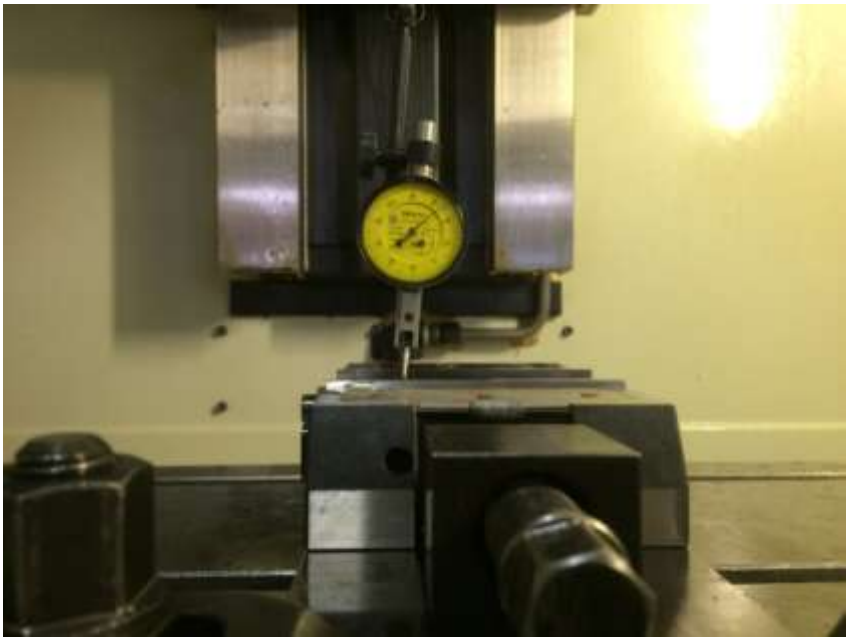


Рисунок 4.20 – Начальное положение индикатора при движении по оси Z

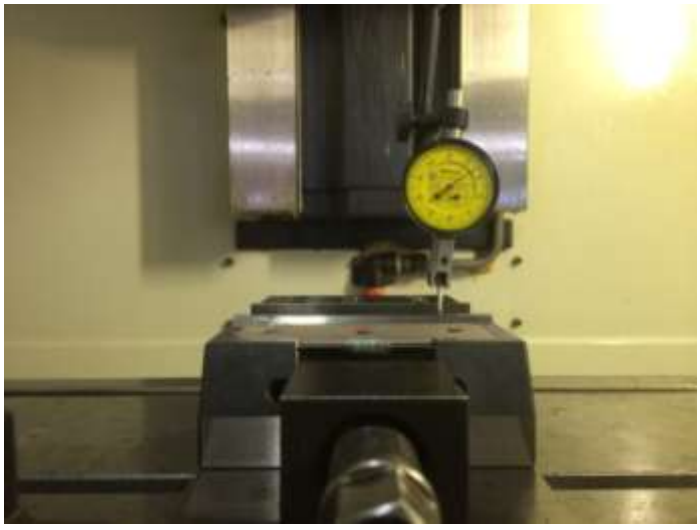


Рисунок 4.21 – Конечное положение индикатора при движении по оси Z

Рисунок 4.22 – Начальное положение индикатора при движении по оси X

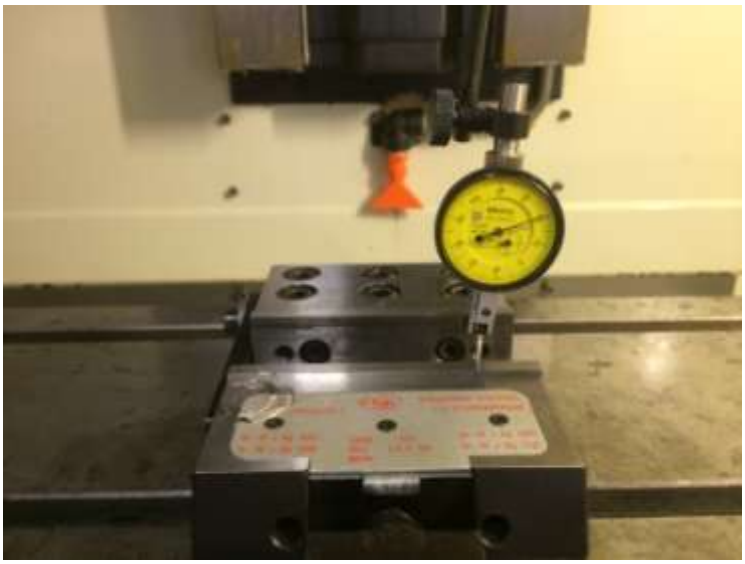


Рисунок 4.23 – Конечное положение индикатора при движении по оси X

- установить первую заготовку в приспособление, как показано на рисунке 4.24;
- определить рабочую систему координат в системе координат станка;
- обработать первую деталь по первоначальной управляющей программе ;
- замерить сферическую поверхность детали на координатно-измерительной машине ;

- установить вторую заготовку на стол станка ;
- обработать вторую деталь по скорректированной управляющей программе ;
- замерить сферическую поверхность детали на координатно-измерительной машине ;
- сравнить результаты обмера двух деталей.

Рисунок 4.24 — Заготовка в приспособлении до обработки

Протоколы измерения представлены на рисунках 4.25–4.26. На рисунке 4.25 показан протокол измерения детали без внесения коррекций на износ, проверочный размер отклонения формы с допуском 0,01 мм. На рисунке показан результат превышающий допуск на 0,038 мм в положительном направлении, данный факт означает , что сферическая фреза оставила больше припуска на обрабатываемых поверхностях за счет размерного износа режущих лезвий. На рисунке 4.26 представлен протокол измерения детали с внесением коррекций на износ, видно что деталь в допуске т.к отклонение формы назначалось в

минус, размеры детали находящиеся в пределах от 31,99 мм до 32 мм считаются годными по данному размеру.


ZEISS Calypso

Время Дата

План контроля

№ КИМ

Номер чертежа

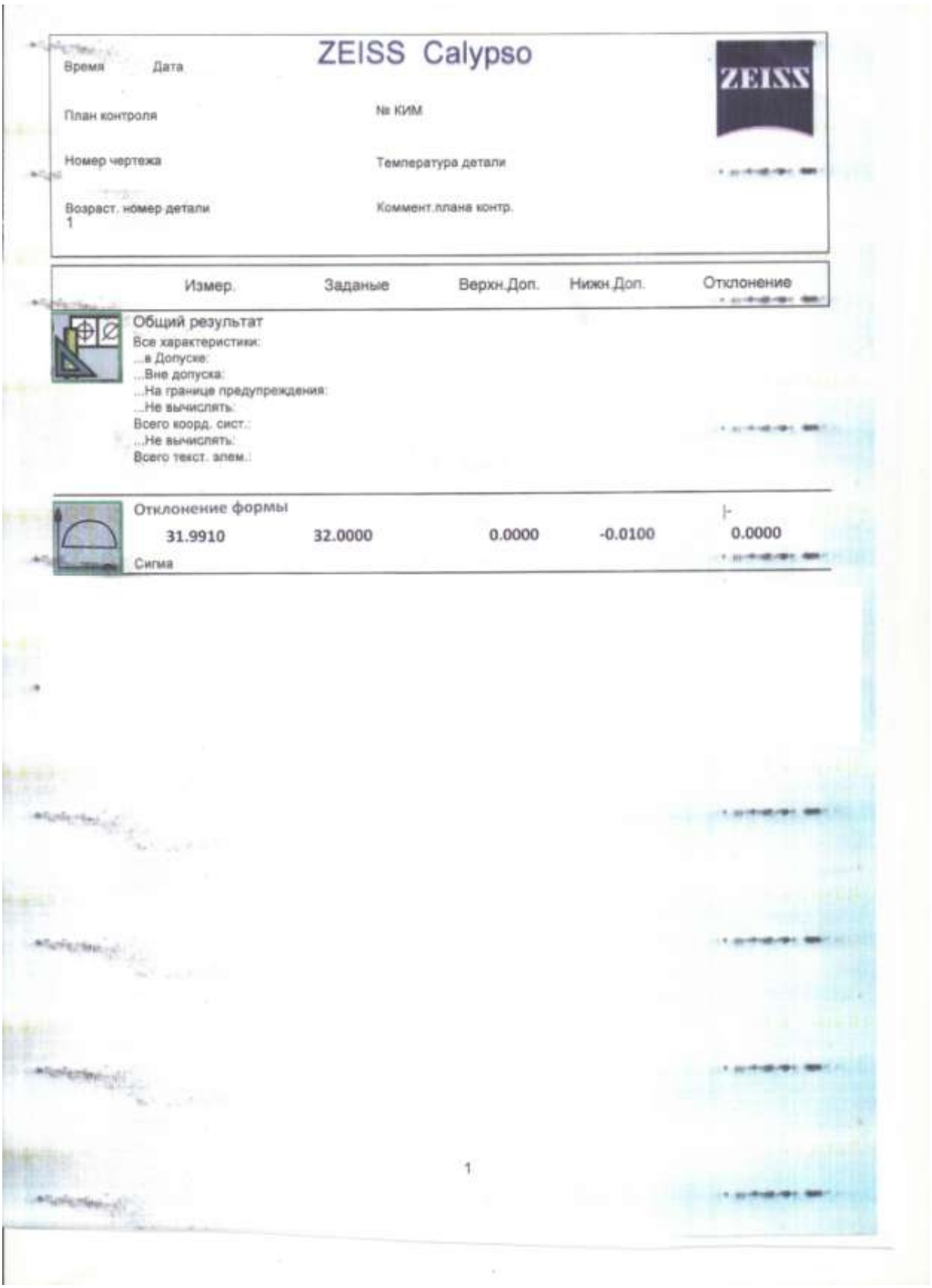
Температура детали

Возраст. номер детали
2

Коммент.плана контр.

	Измер.	Заданые	Верхн.Доп.	Нижн.Доп.	
Общий результат					
Все характеристики:					
...в Допуске:					
...Вне допуска:					
...На границе предупреждения:					
...Не вычислять:					
Всего коорд. сист.:					
...Не вычислять:					
Всего текст. элем.:					
Отклонение формы	32.0380	32.0000	0.0000	-0.0100	
Сигма					

Рисунок 4.25 – Протокол измерения первой детали


ZEISS Calpso

Время Дата

План контроля

№ КИМ

Номер чертежа

Температура детали

Возраст. номер детали
1

Коммент.плана контр.

	Измер.	Заданые	Верхн.Доп.	Нижн.Доп.	Отклонение
Общий результат					
Все характеристики:					
...в Допуске:					
...Вне допуска:					
...На границе предупреждения:					
...Не вычислять:					
Всего коорд. сист.:					
...Не вычислять:					
Всего текст. элем.:					
Отклонение формы					
Сигма	31.9910	32.0000	0.0000	-0.0100	0.0000

1

Рисунок 4.26 – Протокол измерения второй детали

Таким образом, на примере обработки двух идентичных заготовок проверили работу созданной компьютерной программы. Проверка на координатной измерительной машине показала, что обработанные поверхности детали полученная из CAM системы не соответствуют чертежу, т.к в процессе обработки режущие поверхности фрезы износились, тем самым оставили не снятым припуск предназначенный для удаления. Напротив обработанные поверхности детали которые подверглись внесению коррекции в управляющую программу показали, что измеренные поверхности детали соответствует чертежу. Из всего выше перечисленного следует, что созданная программа динамически корректирует размерный износ инструмента в зависимости от его пройденного пути. Также был определен максимальный пройденный путь фрезы в процессе обработки сохранением качества получаемой поверхности и установлена величина размерного износа третьей части пройденного пути фрезы в процессе обработки от максимального.

ЗАКЛЮЧЕНИЕ

При выполнение ВКР были решены следующие задачи :

1 Проанализирована литература по существующим методам повышения точности фасонных поверхностей деталей на чистовых операциях фрезерования. На основании проанализированных данных было установлено, что слабо изучен учет размерного износа режущего инструмента при обработке сложно-профильных поверхностей из высокопрочных материалов на чистовых операциях фрезерования и отсутствие прогнозирующих моделей износа инструмента по задней поверхности. На основании анализа литературных данных была сформулирована цель и задачи исследования

2 Изучен типовой порядок расчета управляющих программ для фрезерных станков с ЧПУ на примере CAM системы PowerMill, следующим этапом стал доработка типового порядка расчета управляющих программ лоя фрезерных станков с ЧПУ путем разработки и внесения математической модели траектории движения инструмента с учетом его износа для обработки на 3х координатных станках с ЧПУ.

3 Создан алгоритм обеспечивающий коррекцию траектории движения инструмента на основании ранее разработанной математической модели. И разработана компьютерная программа по ранее созданному алгоритму.

4 Экспериментально проверена компьютерная программа, а также апробирована разработанная математическая модель. Результатом стало решение одной производственной задачи.

В процессе выполнения работы были достигнуты следующие результаты.

Научным результатом является разработанная зависимость между величинами смещения фрезы и ее размерным износом.

Исследования позволили формулировать вывод.

1 Созданная математическая модель траектории движения инструмента с учетом его износа позволяет повысить точность фасонных поверхностей деталей при обработке их на 3х координатных станках с ЧПУ

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Радзевич, С.П. Формообразование поверхностей деталей. Основы теории: Монография. – Киев: Растан, 2001. – 592 с.

2 Маталин, А.А. Технология машиностроения : учебник для машиностроительных вузов по специальности «Технология машиностроения , металлорежущие станки и инструменты» / А.А. Маталин. – Л.: Машиностроение, 1985. – 496 с.

3 Кисель, А.Г. Влияние смазочно-охлаждающей жидкости на стойкость металлорежущего инструмента при токарной обработки / А.Г. Кисель, Д.С Реченко, А.Ю. Попов, А.А. Ражковский // Омский научный вестник. Сер. Технология машиностроения. – 2013. – Вып. 1. №4 (20). – С.138-142.

4 Поляков, А.Н. Прогнозирование температурных смещения исполнительных органов станков / А.Н. Поляков, К.В. Марусич, И.П. Никитина // Вестник УГАТУ. Серия «Динамика, прочность машин, приборов и аппаратуры». – 2012. – №2 (49). – С. 105-112.

5 Корсаков, В.С. Точность механической обработки / В.С. Корсаков. – М .: Изд-во МашГиз, 1961. – 496 с.

6 Проблемы развития технологии машиностроения : учебник / под ред. Э. А. Сателя. – М .: Машиностроение, 1967. – 592 с.

7 Гондин, Ю.Н. Металлорежущие станки: комплекс учебно-методических материалов / Ю.Н. Гондин, В.А. Колюнов, Б.В. Устинов. – Нижний Новгород: Изд-во Нижегород. гос. техн. ун-т им. Р.Е. Алексеева, 2009. – 154 с.

8 URL: http://xn--80aaung.xn--p1ai/technology – Официальный сайт компании «Mazak» (26.10.2016).

9 Марусич, К.В. Обеспечение требуемой изготовления прецизионных изделий авиационной и ракетной техники путем управления температурными

деформациями станков с числовым программным управлением / К.В. Марусич // Труды МАИ. Серия «Машиноведение. Машиностроение». – 2013. – №5 (4). – С. 23–25.

10 Поляков, А.Н. Исследование термодеформационного состояния металлорежущего станка в условиях переменных тепловых режимов работы / А.Н. Поляков, К.В. Марусич, С.В. Каменев // Вестник УГАТУ. Серия «Динамика, прочность машин, приборов и аппаратуры». – 2011. – №4(49). – С. 105-112.

11 Марусич, К.В. Исследование термодеформационного состояния металлорежущего станка / К.В. Марусич, А.Н. Поляков // Технология машиностроения. –2011.– №2. –С. 7-8.

12 Марусич, К.В. Прогноз температурных перемещений станков в условиях переменных тепловых режимов / К.В. Марусич // Вестник УГАТУ. Серия «Динамика, прочность машин, приборов и аппаратуры». – 2013. – №5(60). – С. 41-46.

13 Алферов, В.И. Исследование и расчет температурных полей и температурных деформаций прецизионных станков от колебаний температуры воздуха и внутренних источников тепла // Омский научный вестник. Сер. Технология машиностроения. – 2009. – Вып. 1. №1 (14). – С. 150-161.

14 Клебанов, Я.М. Влияние упругих деформаций прецизионного поворотного стола на погрешность позиционирования заготовок / Я.М. Клебанов, А.И. Симаков, Е.А. Солдусова // Технические науки. – 2017. – Вып №5 (59). – С. 74-77.

15 Гузеев, В.И. Прогнозирование точности и качества при проектировании технологических процессов механической обработки : электронное учебное пособие / В.И. Гузеев, Г.И. Буторин, В.Ю. Шамин. –Челябинск : Изд-во ЮУрГУ, 2013. – 77 с.

16 Гузеев, В.И. Методика расчетного определения количества стадий обработки поверхностей деталей / В.И. Гузеев, В.В. Батуев // Современные технологии и бизнес: сб. науч. тр. – Челябинск: Издание ЧНЦ РАЕН, ЧРО МААНОИ, ЧООО РС НИО, ЧелЦНТИ, 2006. – С. 33-36.

17 Гузеев, В.И. Учет влияния следов предшествующей обработки на точность фрезерования пространственно-сложных поверхностей / В.И. Гузеев, В.В. Батуев // Прогрессивные технологии в машиностроении: сб. науч. тр. – Челябинск: Изд-во ЮУрГУ, 2002. – С. 28-31.

18 Батуев, В.В. Обеспечение производительности при чистовом фрезеровании пространственно-сложных поверхностей в условии заданной точности / В.В. Батуев // Прогрессивные технологии в машиностроении: сб. науч. тр. – Челябинск: Изд-во ЮУрГУ, 2005. – С. 91-94.

19 Батуев, В.А. Управление подачей с целью обеспечения точности фрезерования пространственно-сложных поверхностей на станках с ЧПУ / В.А. Батуев, В.В. Батуев // Сборник тезисов докладов международной научно-технической конференции. – Барнаул: Изд-во АГТУ им. И.И. Ползунова, 2003. – С. 11-12.

20 Выбойщик, В.А. Учет динамических характеристик процесса объемного фрезерования с целью повышения точности обработки / В.А. Выбойщик // Прогрессивные технологии в машиностроении: сб. науч. тр. – Челябинск: Изд-во ЮУрГУ, 1988. – С. 98-102.

21 Батуев, В.В. Расчет толщины срезаемого слоя при фрезеровании пространственно-сложных поверхностей, имеющих ступенчатый припуск / В.В. Батуев // Известия Челябинского научного центра. – Челябинск : Изд-во ЮУрГУ, 2006.

22 Аршинов, В.А. Резание металлов и режущий инструмент : учебник для машиностроительных вузов / В.А. Аршинов, Г.А. Алексеев – М.: Машиностроение, 1975. – 440 с.

23 Yalçin, B. The effects of various cooling strategies on surface roughness and tool wear during soft materials milling / B. Yalçin, A. Özgür // Materials and Design. – 2009. – V. 30. № 3. – P. 896-899.

24 Симанин, Н.А. Метод компенсации износа фрезы на станках с ЧПУ / Н.А. Симанин, Ю.М. Передрей // 21 век, итоги прошлого и проблемы настоящего. – Пенза: Изд-во ПГТУ, 2015. – С. 99-102.

25 Гороховский, А.К. Методика определения влияния износа лезвий фрез на точность процесса обработки / А.К. Гороховский, В.И. Сулинов // Современные наукоемкие технологии. – Екатеринбург: Изд-во УГЛТУ, 2005. – С. 9-11.

26 Скуратов, Д.Л. Исследование влияния износа по задней поверхности зубьев концевой фрезы на контактную температуру в зоне резания / Д.Л. Скуратов, Д.В. Евдокимов // Вестник СГАУ. Сер. Машиностроение. – 2015. – №3. – С. 201.

27 Пименов, Д.Ю. Определение допустимого износа торцевых фрез для обеспечения требуемой точности / Д.Ю. Пименов, В.И. Гузеев, В.А. Пашнев // Известия Челябинского научного центра. – Челябинск : Изд-во ЮУрГУ, 2006. – С. 43-45.

28 Menezes, J. Productivity progression with tool wear in titanium milling / J.Menezes, M. Rubeo, K. Kiran // Procedia manufacturing. – 2016. – V.5. – P. 427-441.

29 Muhammad, R. The Aplication of I-Kaz based Method for tool wear monitoring using cuttin force signal / R. Muhammad, A. Jaharah // Procedia Engineering. MITC 2013. – 2013. – № 68. – P. 461-468.

30 Skordaris, G. A critical review of measures for an effective application of nano-structured coating in milling / G. Skordaris, K. Bouzakis, T.Kotsanis // Tribology in industry. – 2017. – V. 39. – P. 211-218.

31 Wang, G. On line tool wear monitoring based on auto associative NN / G. Wang, Y.Cui // Journal of Intelligent Manufacturing .Procedia CIRP. – 2013. – №58. – P. 24.

32 Stavropoulos, P. Tool wear predictability estimation in milling based on multi-sensorial data / P. Stavropoulos, A. Papacharalampopoulos, E. Vasiliadis // The International Journal of Advanced Manufacturing Technology. – 2016. – Т. 82. № 4. – P. 509-521.

33 Bleys, P. Sensing and compensation of tool wear in milling EDM / P. Bleys, J. Kruth, B. Lauwers // Journal of Materials Processing Technology. – 2004. – Т. 149. №3. – P. 139-146.

34 Yalçix, B. The effects of various cooling strategies on surface rounghness and tool wear during soft materials milling / B. Yalçin, A. Özgür, M. Koru // Materials and Design. – 2009. – Т. 30. № 3. – P. 896-899.

35 Ming, L. Tool wear length estimation with a self-learning fuzzy inference algorithm in finish milling /L. Ming, Y. Xiaohong, Y. Shuzi // The International Journal of Advanced Manufacturing Technology. – 1999. – Т. 15. № 8. – P. 537-545.

36 Колесников, В.И. Метод обеспечения точности контурного фрезерования в процессе подготовки программы для станка с ЧПУ / В.И. Колесников // Сборник тезисов докладов международной научно-технической конференции. – Свердловск: Изд-во УГТУ-УПИ, 1986. – С. 11-12.

37 Щуров, И.А. Решение задачи формообразования сложных поверхностей деталей с учетом износа инструментов при обработке на станках с ЧПУ / И.А. Щуров // Известия ТулГУ. Сер. Технические науки. – 2016. – Вып.8. – С. 120-124.

38 Основы автоматизированного проектирования : учебник для вузов / И.П. Норенков – 4-е изд., перераб. и доп. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2009. – 430 с.

39 URL: https: www. ptc.com / ru /products /cad /creo /parametric.– Официальный сайт компании PTC (01.11.2017).

40 URL: https:// www. delcam.ru – Официальный сайт компании Delcam (01.11.2017).

41 ГОСТ 7.83 – 2001. Издания. Электронные издания. Основные виды и выходные данные. – М.: Стандартинформ, 2002. – 35 с.

42 URL: https://plmclub.ru/sites/default/files/broshures/nx-cam – Официальный сайт компании NX (18.11.2017).

43 URL: http://www.espritcam.ru – Официальный сайт компании Espirit (18.11.2017).

44 ГОСТ 23501.08-80. Издания. Системы автоматизированного проектирования классификация обозначения. – М.: Изд-во стандартов, 2002. – 45 с.

45 URL: https://www.3ds.com – Официальный сайт компании 3ds (18.11.2017).

46 URL: http://www.ncgcam. com – Официальный сайт компании ncgcam (18.11.2017).

47 URL: http://ascon.ru/products/1186/review – Официальный сайт компании Askon (27.11.2017).

48 URL : https:// www. sprut.ru – Официальный сайт компании Sprut технологии (27.11.2017).

49 Шаламов В.Г. Теория и проектирование режущего инструмента : Текст лекции / В.Г. Шаламов. – Челябинск: Изд-во ЮУрГУ, 2003. – 156 с.

50 ГОСТ 2.052-2006. Издания. Единая система конструкторской документации (ЕСКД). Электронная модель изделия . Общие положения. – М.: Изд-во стандартов, 2006. – 11с.

51 URL: http://cyclowiki.org/wiki/ %D0%9F%D0%BE%D1%81%D1%82%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE

%D1%80_%D0%B4%D0%BB%D1%8F_%D1%81%D1%82%D0%B0%D0%BD%D0%BA%D0%B0_%D1%81_%D0%A7%D0%9F%D0%A3 – Постпроцессор для станка с ЧПУ (29.12.2017)

52 ГОСТ 19781-90. Издания. Обеспечение систем обработки, термины и определения. – М.: Изд-во стандартов, 2001. – 11 с.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД КОМПЬЮТЕРНОЙ ПРОГРАММЫ

```
package ru.pii.toolWearing.excel.formatters;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.List;

import javax.swing.event.EventListenerList;

import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.CellStyle;
import org.apache.poi.ss.usermodel.CreationHelper;
import org.apache.poi.ss.usermodel.DataFormatter;
import org.apache.poi.ss.usermodel.Workbook;

/**

 *
 *
 */
public abstract class AbstractFormatter
{
	protected CellStyle style = null;
	protected DataFormatter formatter = null;
	protected Object value = null;
	protected Cell cell = null;
	private EventListenerList onSetCellDataListeners = null;
	private EventListenerList onGetCellDataListeners = null;

	/**
	 */
	public AbstractFormatter()
	{
		this.formatter = new DataFormatter();
	}

	/**
	  * @return
	 */
	public CellStyle getStyle()
	{
```

```
		return style;
	}

	/**
	 * @param style
	 */
	public void setStyle(CellStyle style)
	{
		this.style = style;


	//System.out.println(Arrays.toString(Thread.currentThread().getStackTrace())
.replace(",", "\n    "));
	}

	/**
	 * @param cell
	 * @param value
	 * @throws Exception
	 */
	public abstract void setCellData(Cell cell, Object value) throws Exception;

	/**
		 * @param cell
	 * @return
	 * @throws Exception
	 */
	public abstract Object getCellData(Cell cell) throws Exception;

	/**
	 * @param _classes
	 * @return
	 * @throws Exception
	 */
	public static List<AbstractFormatter>
createFormattersByClasses(List<Class<?>> _classes) throws Exception
	{
		if (_classes == null || _classes.isEmpty())
			throw new Exception("Создание форматтеров по классам: не
переданны классы");
		List<AbstractFormatter> result = new ArrayList<AbstractFormatter>();
		for (Class<?> _class : _classes)
		{
			result.add(createFormatterByClass(_class));
```

```
            }
            return result;
        }

        /**
        *
         * @param _class
         * @return
         * @throws Exception
         */
        public static AbstractFormatter createFormatterByClass(Class<?> _class)
throws Exception
        {
            if (_class == null )
                throw new Exception("Создание форматтера по классу: вход-
ной аргумент = NULL");
            if (_class == String.class)
                return new StringFormatter();
            if (_class == Integer.class || _class == Short.class || _class == ja-
va.math.BigDecimal.class)
                return new IntegerFormatter();
            if (_class == Long.class)
                return new LongFormatter();
            if (_class == Float.class)
                return new FloatFormatter();
            if (_class == Double.class)
                return new DoubleFormatter();
            if (java.util.Date.class.isAssignableFrom(_class)) //any class, inherited
from class java.util.Date
                return new DateFormatter();
            return new StringFormatter(); //if not found - string is better then noth-
ing
        }

        /**
         */
        public static void
applyDateStylesForDateFormatters(List<AbstractFormatter> formattersList, String
datePattern, Workbook wb)
        {
            CreationHelper ch = wb.getCreationHelper();
            CellStyle style = wb.createCellStyle();
            style.setDataFormat(ch.createDataFormat().getFormat(datePattern));
            for (AbstractFormatter formatter : formattersList)
```

```
        {
            if (formatter instanceof DateFormatter)
                formatter.setStyle(style);
        }
    }

    public Object getValue()
    {
        return value;
    }

    public void setValue(Object value)
    {
        this.value = value;
    }

    public Cell getCell()
    {
        return cell;
    }

    public void setCell(Cell cell)
    {
        this.cell = cell;
    }

    /**
     * @param listener
     */
    public void addListenerOnSetCellData(ActionListener listener)
    {
        if (this.onSetCellDataListeners == null)
            this.onSetCellDataListeners = new EventListenerList();
        this.onSetCellDataListeners.add(ActionListener.class, listener);
    }

    public void removeListenerOnSetCellData(ActionListener listener)
    {
        if (this.onSetCellDataListeners == null)
            return;
        this.onSetCellDataListeners.remove(ActionListener.class, listener);
    }

    /**
```

```
	 * @param listener
	 */
	public void addListenerOnGetCellData(ActionListener listener)
	{
		if (this.onGetCellDataListeners == null)
			this.onGetCellDataListeners = new EventListenerList();
		this.onGetCellDataListeners.add(ActionListener.class, listener);
	}

	public void removeListenerOnGetCellData(ActionListener listener)
	{
		if (this.onGetCellDataListeners == null)
			return;
		this.onGetCellDataListeners.remove(ActionListener.class, listener);
	}

	/**
		 */
	public void fireGetCellDataEvent()
	{
		if (this.onGetCellDataListeners == null)
			return;
		for (ActionListener listener :
this.onGetCellDataListeners.getListeners(ActionListener.class))
		{
			listener.actionPerformed(new ActionEvent(this,
ActionEvent.ACTION_PERFORMED, ""));
		}
	}
	/**
	 */
	public void fireSetCellDataEvent()
	{
		if (this.onSetCellDataListeners == null)
			return;
		for (ActionListener listener :
this.onSetCellDataListeners.getListeners(ActionListener.class))
		{
			listener.actionPerformed(new ActionEvent(this,
ActionEvent.ACTION_PERFORMED, ""));
		}
	}

}
```

```
package ru.pii.toolWearing.excel.formatters;

import java.util.Date;

import org.apache.poi.hssf.usermodel.HSSFDateUtil;
import org.apache.poi.ss.usermodel.Cell;

public class DateFormatter extends AbstractFormatter
{
	public DateFormatter()
	{
		super();
	}

	@Override
	public void setCellData(Cell cell, Object value) throws Exception
	{
		if(!(value instanceof Date))
			throw new Exception("Значение не является датой " + value);
		this.cell = cell;
		this.value = value;
		cell.setCellValue((Date) value);
		if (this.style != null)
			cell.setCellStyle(this.style);
		fireSetCellDataEvent();
	}

	@Override
	public Object getCellData(Cell cell) throws Exception
	{
		if(!HSSFDateUtil.isCellDateFormatted(cell))
			throw new Exception("Ячейка не формата даты: "+
this.formatter.formatCellValue(cell));
		this.cell = cell;
		this.value = cell.getDateCellValue();
		fireGetCellDataEvent();
		if(this.value==null)
			return new Date(Long.parseLong("0"));
		return this.value;
	}
	/**
	 */
	@Override
```

```
	public String toString()
	{
		return "DateFormatter";

	}

}

package ru.pii.toolWearing.excel.formatters;

import java.util.Date;

import org.apache.poi.hssf.usermodel.HSSFDateUtil;
import org.apache.poi.ss.usermodel.Cell;

public class DateFormatter extends AbstractFormatter
{
	public DateFormatter()
	{
		super();
	}

	@Override
	public void setCellData(Cell cell, Object value) throws Exception
	{
		if(!(value instanceof Date))
			throw new Exception("Значение не является датой " + value);
		this.cell = cell;
		this.value = value;
		cell.setCellValue((Date) value);
		if (this.style != null)
			cell.setCellStyle(this.style);
		fireSetCellDataEvent();
	}

	@Override
	public Object getCellData(Cell cell) throws Exception
	{
		if(!HSSFDateUtil.isCellDateFormatted(cell))
			throw new Exception("Ячейка не формата даты: "+
this.formatter.formatCellValue(cell));
		this.cell = cell;
		this.value = cell.getDateCellValue();
		fireGetCellDataEvent();
```

```
            if(this.value==null)
                return new Date(Long.parseLong("0"));
            return this.value;
        }
        /**
         */
        @Override
        public String toString()
        {
            return "DateFormatter";

        }

}

package ru.pii.toolWearing.excel.formatters;

import org.apache.poi.ss.usermodel.Cell;

public class DoubleFormatter extends AbstractFormatter
{
        public DoubleFormatter()
        {
            super();
        }

        @Override
        public void setCellData(Cell cell, Object value) throws Exception
        {
            if(!(value instanceof Number))
                throw new Exception("Значение не является числом: " + val-
ue);
            this.cell = cell;
            this.value = ((Number) value).doubleValue();
            cell.setCellValue((double)this.value);
            if (this.style != null)
                cell.setCellStyle(this.style);
            fireSetCellDataEvent();
        }

        @Override
        public Object getCellData(Cell cell) throws Exception
        {
            this.value=this.formatter.formatCellValue(cell);
```

```
                this.cell = cell;
                fireGetCellDataEvent();
                if(this.value.toString().isEmpty())
                        return Double.parseDouble("0");
                try
                {
                        return Double.parseDouble(this.value.toString().replace(",", "."));
                } catch (Exception e)
                {
                        throw new Exception("Не могу произвести конвертацию: "+
this.value.toString() + " -> Double");
                }
        }
        /**
         */
        @Override
        public String toString()
        {
                return "DoubleFormatter";

        }

}

package ru.pii.toolWearing.excel.formatters;

import org.apache.poi.ss.usermodel.Cell;

public class FloatFormatter extends AbstractFormatter
{
        public FloatFormatter()
        {
                super();
        }

        @Override
        public void setCellData(Cell cell, Object value) throws Exception
        {
                if(!(value instanceof Number))
                        throw new Exception("Значение не является числом: " + val-
ue);
                this.cell = cell;
                this.value = ((Number) value).floatValue();
                cell.setCellValue((float)this.value);
```

```
            if (this.style != null)
                cell.setCellStyle(this.style);
            fireSetCellDataEvent();
        }

        @Override
        public Object getCellData(Cell cell) throws Exception
        {
            this.value=this.formatter.formatCellValue(cell);
            this.cell = cell;
            fireGetCellDataEvent();
            if(this.value.toString().isEmpty())
                return Float.parseFloat("0");
            try
            {
                return Float.parseFloat(this.value.toString().replace(",", "."));
            } catch (Exception e)
            {
                throw new Exception("Не могу произвести конвертацию: "+
this.value.toString() + " -> Float");
            }
        }
        /**
         */
        @Override
        public String toString()
        {
            return "FloatFormatter";

        }

}

package ru.pii.toolWearing.excel.formatters;

import org.apache.poi.ss.usermodel.Cell;

public class IntegerFormatter extends AbstractFormatter
{
        public IntegerFormatter()
        {
            super();
        }
```

```
	@Override
	public void setCellData(Cell cell, Object value) throws Exception
	{
		if(!(value instanceof Number))
			throw new Exception("Значение не является числом: " + val-
ue);
		this.cell = cell;
		this.value = ((Number) value).intValue();
		cell.setCellValue((int)this.value);
		if (this.style != null)
			cell.setCellStyle(this.style);
		this.fireSetCellDataEvent();
	}

	@Override
	public Object getCellData(Cell cell) throws Exception
	{
		this.value =this.formatter.formatCellValue(cell);
		this.cell = cell;
		this.fireGetCellDataEvent();
		if(this.value.toString().isEmpty())
			return Integer.parseInt("0");
		try
		{
			return Integer.parseInt(this.value.toString());
		} catch (Exception e)
		{
			throw new Exception("Не могу произвести конвертацию: "+
this.value.toString() + " -> Integer");
		}
	}
	/**
	 */
	@Override
	public String toString()
	{
		return "IntegerFormatter";

	}

}

package ru.pii.toolWearing.excel.formatters;
```

```
import org.apache.poi.ss.usermodel.Cell;

public class LongFormatter extends AbstractFormatter
{
	public LongFormatter()
	{
		super();
	}

	@Override
	public void setCellData(Cell cell, Object value) throws Exception
	{
		if(!(value instanceof Number))
			throw new Exception("Значение не является числом: " + val-
ue);
		this.cell = cell;
		this.value = ((Number) value).intValue();
		cell.setCellValue((int)this.value);
		if (this.style != null)
			cell.setCellStyle(this.style);
		this.fireSetCellDataEvent();
	}

	@Override
	public Object getCellData(Cell cell) throws Exception
	{
		this.value =this.formatter.formatCellValue(cell);
		this.cell = cell;
		this.fireGetCellDataEvent();
		if(this.value.toString().isEmpty())
			return Long.parseLong("0");
		try
		{
			return Long.parseLong(this.value.toString());
		} catch (Exception e)
		{
			throw new Exception("Не могу произвести конвертацию: "+
this.value.toString() + " -> Long");
		}
	}
	/**
	 */
	@Override
	public String toString()
```

```
	{
		return "LongFormatter";

	}

}

package ru.pii.toolWearing.excel.formatters;

import org.apache.poi.ss.usermodel.Cell;

public class StringFormatter extends AbstractFormatter
{
	public StringFormatter()
	{
		super();
	}

	@Override
	public void setCellData(Cell cell, Object value) throws Exception
	{
		this.cell = cell;
		this.value = value.toString();
		cell.setCellValue((String)this.value);
		if (this.style != null)
			cell.setCellStyle(this.style);
		this.fireSetCellDataEvent();
	}

	@Override
	public Object getCellData(Cell cell) throws Exception
	{
		this.cell = cell;
		this.value = this.formatter.formatCellValue(cell);
		this.fireGetCellDataEvent();
		return this.value.toString();
	}
	/**
	 */
	@Override
	public String toString()
	{
		return "StringFormatter";
```

```
	}

}

package ru.pii.toolWearing.excel;

import java.util.ArrayList;
import java.util.List;

import ru.pii.toolWearing.excel.formatters.AbstractFormatter;


/**
 *
 */
public class DataTable
{
	private List<List<Object>> table = null;
	private List<AbstractFormatter> columnFormatters = null;
	private List<String> columnHeaders = null;

	public DataTable(List<String> columnHeaders, List<Class<?>>
columnTypes) throws Exception{
		if(columnTypes==null || columnTypes.isEmpty() ||
columnHeaders==null || columnHeaders.isEmpty())
			throw new Exception("В таблицу не переданы заголовки или
типы колонок");
		if(columnTypes.size() != columnHeaders.size())
			throw new Exception("Колличество заголовков не совпадает с
количеством типов");

		this.setTable(new ArrayList<List<Object>>());

	this.setColumnFormatters(AbstractFormatter.createFormattersByClasses(colu
mnTypes));
		this.setColumnHeaders(columnHeaders);
	}

	/**
	 * @return
	 */
	public List<String> getColumnHeaders()
	{
		return this.columnHeaders;
```

```
	}

	/**
	 * @return
	 */
	public void setColumnHeaders(List<String> columnHeaders)
	{
		this.columnHeaders = columnHeaders;
	}

	/**
	 * @return
	 */
	public List<AbstractFormatter> getColumnFormatters()
	{
		return this.columnFormatters;
	}

	/**
	 * @return
	 */
	public void setColumnFormatters(List<AbstractFormatter>
columnFormatters)
	{
		this.columnFormatters = columnFormatters;
	}

	/**
	 * @return
	 */
	public List<List<Object>> getTable()
	{
		return this.table;
	}

	/**
	 * @param table
	 */
	public void setTable(List<List<Object>> table)
	{
		this.table = table;
	}
```

```
/**
 */
@Override
public String toString()
{
	StringBuilder sb = new StringBuilder(this.getStructureString());
	for(List<Object> row : this.table)
	{
		sb.append("[");
		for(Object value : row)
		{
			sb.append(value.toString());
			sb.append(";");
		}
		sb.append("]\n");
	}
	return sb.toString();

}
/**
 */

public String toString(int n)
{
	StringBuilder sb = new StringBuilder(this.getStructureString());
	for(int i =0; i<n; i++)
	{
		List<Object> row = this.table.get(i);
		sb.append("[");
		for(Object value : row)
		{
			sb.append(value.toString());
			sb.append(";");
		}
		sb.append("]\n");
	}
	return sb.toString();

}
/**
 * @return
 */
public String getStructureString()
```

```
	{
		StringBuilder sb = new StringBuilder();
		sb.append("[");
		for(int i=0; i<this.columnHeaders.size(); i++)
		{
			sb.append(this.getColumnHeaders().get(i));
			sb.append("(");
			sb.append(this.getColumnFormatters().get(i));
			sb.append(");");
		}
		sb.append("]\n");
		return sb.toString();
	}

}

package ru.pii.toolWearing.excel;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.poi.openxml4j.exceptions.InvalidFormatException;
import org.apache.poi.ss.usermodel.DataFormatter;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;

import ru.pii.toolWearing.excel.formatters.AbstractFormatter;

/**
 *
 *
 */
public class ExcelReader
{
	private File excelFile = null;
	private DataFormatter formatter = null;
	private Workbook wb = null;
```

```
	/**
	 * @param path
	 * @throws Exception
	 */
	public ExcelReader(String path) throws Exception
	{
		this(new File(path));
	}

	public ExcelReader(File file) throws Exception
	{
		this.excelFile = file;
		if (!this.excelFile.exists() || this.excelFile.isDirectory())
			throw new FileNotFoundException("Такого файла не сущест-
вует либо это папка");
		this.formatter = new DataFormatter();
	}

	public Workbook getWb() throws FileNotFoundException, IOException,
InvalidFormatException
	{
		if (this.wb != null)
			return this.wb;
		try (FileInputStream fis = new FileInputStream(this.excelFile))
		{
			this.wb = WorkbookFactory.create(fis);
		}
		return this.wb;
	}

	/**
	 */
	public void clearWorkbook()
	{
		this.wb = null;
	}

	/**
	 *
	 * @param sheetNumber
	 * @param columns
	 * @param columnTypes
	 * @return
```

```
     * @throws Exception
     */
    public DataTable readSheet(int sheetNumber, int[] columns, List<Class<?>>
columnTypes) throws Exception
    {
        if (columnTypes == null)
        {
            columnTypes = new ArrayList<Class<?>>();
            for (int i = 0; i < columns.length; i++)
            {
                columnTypes.add(String.class);
            }
        }
        Workbook wb = getWb();
        Sheet sheet = wb.getSheetAt(sheetNumber);
        DataTable dt = new DataTable(getHeaders(columns, sheet),
columnTypes);
        List<List<Object>> table = dt.getTable(); //we need to read data from
excel to this table
        List<AbstractFormatter> columnFormatters =
dt.getColumnFormatters();
        Iterator<Row> rowIterator = sheet.iterator();
        if (rowIterator.hasNext())
            rowIterator.next(); //single time to skip row with headers
        while (rowIterator.hasNext())
        {
            Row row = rowIterator.next();
            List<Object> rowData = new ArrayList<Object>();
            for (int i = 0; i < columns.length; i++)
            {
                Object value =
columnFormatters.get(i).getCellData(row.getCell(columns[i]));
                rowData.add(value);
            }
            table.add(rowData);
        }
        return dt;
    }

    /**
     * @param sheetNumber
     * @param columns
     * @return
     * @throws Exception
```

```
     */
    public List<List<String>> readSheet(int sheetNumber, int[] columns) throws
Exception
    {
        Workbook wb = getWb();
        Sheet sheet = wb.getSheetAt(sheetNumber);
        List<List<String>> table = new ArrayList<List<String>>(); //we need
to read data from excel to this table
        Iterator<Row> rowIterator = sheet.iterator();
        if (rowIterator.hasNext())
            rowIterator.next(); //single time to skip row with headers
        while (rowIterator.hasNext())
        {
            Row row = rowIterator.next();
            List<String> rowData = new ArrayList<String>();
            for (int i = 0; i < columns.length; i++)
            {
                String value =
this.formatter.formatCellValue(row.getCell(columns[i]));
                rowData.add(value);
            }
            table.add(rowData);
        }
        return table;
    }

    /** @param sheetNumber
     * @param column
     * @return
     * @throws Exception
     */
    public List<String> readColumn(int sheetNumber, int column) throws Excep-
tion
    {
        Workbook wb = getWb();
        Sheet sheet = wb.getSheetAt(sheetNumber);
        List<String> list = new ArrayList<String>(); //we need to read data
from excel to this list
        Iterator<Row> rowIterator = sheet.iterator();
        if (rowIterator.hasNext())
            rowIterator.next(); //single time to skip row with headers
        while (rowIterator.hasNext())
        {
            Row row = rowIterator.next();
```

```
                String value =
this.formatter.formatCellValue(row.getCell(column));
                list.add(value);
            }
            return list;
        }

        /**
         * @param columns
         * @param sheet
         * @return
         */
        public List<String> getHeaders(int[] columns, Sheet sheet)
        {
            List<String> headers = new ArrayList<String>();
            Row row = sheet.getRow(0);
            for (int i = 0; i < columns.length; i++)
            {
                head-
ers.add(this.formatter.formatCellValue(row.getCell(columns[i])));
            }
            return headers;
        }

}

package ru.pii.toolWearing.excel;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
```

```
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import ru.pii.toolWearing.excel.formatters.AbstractFormatter;
import ru.pii.toolWearing.loger.ProgressLog;

public class ExcelWriter
{
	private File excelFile = null;
	private Workbook wb = null;

	 *
	 * @throws FileNotFoundException
	 */
	public ExcelWriter(String path) throws FileNotFoundException
	{
		this(new File(path));
	}

	/**
	 * @throws FileNotFoundException
	 */
	public ExcelWriter(File file) throws FileNotFoundException
	{
		if (file == null)
			throw new FileNotFoundException("file=null");
		this.excelFile = file;
		if (this.excelFile.isDirectory())
			throw new FileNotFoundException("Путь указывает на
папку");
		File dir = new File(file.getParent());
		if (!dir.exists())
			dir.mkdirs(); //create dirictories if not exists
		if (this.excelFile.getAbsolutePath().endsWith("xls"))
		{
			setWb(new HSSFWorkbook());
		}
		if (this.excelFile.getAbsolutePath().endsWith("xlsx"))
		{
			setWb(new XSSFWorkbook());
		}
	}

	/**
	 * @param sheetName
```

```
     * @param dataTable
     * @return
     * @throws Exception
     */
    public Sheet createSheet(String sheetName, DataTable dataTable) throws Exception
    {
        if (dataTable == null || sheetName == null)
            throw new Exception("Один из переданных в метод аргументов - null");
        List<List<Object>> table = dataTable.getTable();
        if (table == null)
            throw new Exception("Объект таблицы - null");
        List<AbstractFormatter> columnFormatters = dataTable.getColumnFormatters();

    AbstractFormatter.applyDateStylesForDateFormatters(columnFormatters, "yyyy.MM.dd HH:mm", this.wb);
        Sheet sheet = this.wb.createSheet(sheetName);
        writeRow(dataTable.getColumnHeaders(), sheet.createRow(0)); //write headers
        ProgressLog pl = new ProgressLog("Выгрузка в эксель файл: ", table.size());
        for (int j = 0; j < table.size(); j++)
        {
            List<Object> tableRow = table.get(j);
            Row exRow = sheet.createRow(j + 1);
            for (int i = 0; i < tableRow.size(); i++)
            {
                Object value = tableRow.get(i);
                if (value == null)
                    continue;
                Cell cell = exRow.createCell(i);
                columnFormatters.get(i).setCellData(cell, value);
            }
            pl.tick();
        }
        return sheet;
    }

    /**
     * @param sheetName
     * @param resultSet
     * @throws Exception
```

```
	 */
	public Sheet createSheet(String sheetName, ResultSet resultSet) throws Exception
	{
		if (resultSet == null || sheetName == null)
			throw new Exception("Один из переданных в метод аргументов - null");
		ResultSetMetaData rsmd = resultSet.getMetaData();
		int columnCount = rsmd.getColumnCount();
		List<AbstractFormatter> columnFormatters = AbstractFormatter.createFormattersByClasses(this.getClassesList(rsmd)); //create formatters by classes

	AbstractFormatter.applyDateStylesForDateFormatters(columnFormatters, "yyyy.MM.dd HH:mm", this.wb);
		Sheet sheet = this.wb.createSheet(sheetName);
		writeRow(this.getColumnHeaders(rsmd), sheet.createRow(0)); //write headers

		Object value = null;
		Row row = null;
		int rowCounter = 1;
		while (resultSet.next())
		{
			row = sheet.createRow(rowCounter);
			for (int i = 0; i < columnCount; i++)
			{
				value = resultSet.getObject(i + 1);
				if (value == null)
					continue;
				columnFormatters.get(i).setCellData(row.createCell(i), value);
			}
			rowCounter++;
		}
		return sheet;
	}

	/**
	 * @param sheetName
	 * @param columnHeader
	 * @param list
	 * @return
	 * @throws Exception
```

```
	 */
	public Sheet createSheet(String sheetName, String columnHeader,
List<String> list) throws Exception
	{
		if (list == null || sheetName == null)
			throw new Exception("Один из переданных в метод аргумен-
тов - null");
		if (columnHeader == null)
			columnHeader = "";
		Sheet sheet = this.wb.createSheet(sheetName);
		Row headRow = sheet.createRow(0);
		Cell headCell = headRow.createCell(0);
		headCell.setCellValue(columnHeader);
		for (int j = 0; j < list.size(); j++)
		{
			String value = list.get(j);
			if (value == null)
				continue;
			Row exRow = sheet.createRow(j + 1);
			Cell cell = exRow.createCell(0);
			cell.setCellValue(value);
		}
		return sheet;
	}

	/**
	 * @param sheetName
	 * @param columnHeaders
	 * @param table
	 * @return
	 * @throws Exception
	 */
	public Sheet createSheet(String sheetName, List<String> columnHeaders,
List<List<String>> table) throws Exception
	{
		if (table == null || sheetName == null)
			throw new Exception("Один из переданных в метод аргумен-
тов - null");
		if (columnHeaders == null)
			columnHeaders = new ArrayList<String>();
		Sheet sheet = this.wb.createSheet(sheetName);
		Row headRow = sheet.createRow(0);
		writeRow(columnHeaders, headRow);
		for (int j = 0; j < table.size(); j++)
```

```
            {
                List<String> row = table.get(j);
                Row exRow = sheet.createRow(j + 1);
                writeRow(row, exRow);
            }
            return sheet;
        }

        /**
         * @param sheetName
         * @param columnHeader1
         * @param columnHeader2
         * @param table
         * @return
         * @throws Exception
         */
        public Sheet createSheet(String sheetName, String columnHeader1, String
columnHeader2, Map<Object, Object> table) throws Exception
        {
            if (table == null || sheetName == null)
                throw new Exception("Один из переданных в метод аргумен-
тов - null");
            if (columnHeader1 == null)
                columnHeader1 = "";
            if (columnHeader2 == null)
                columnHeader2 = "";
            Sheet sheet = this.wb.createSheet(sheetName);
            Row headRow = sheet.createRow(0);
            Cell headCell = headRow.createCell(0);
            headCell.setCellValue(columnHeader1);
            headCell = headRow.createCell(1);
            headCell.setCellValue(columnHeader2);
            int j = 0;
            for (Map.Entry<Object, Object> entry : table.entrySet())
            {
                Row exRow = sheet.createRow(j + 1);
                String value = entry.getKey().toString();
                Cell cell = exRow.createCell(0);
                cell.setCellValue(value);
                value = entry.getValue().toString();
                cell = exRow.createCell(1);
                cell.setCellValue(value);
                j++;
            }
```

```
        return sheet;
    }

    /**
     * @param rowValues
     * @param row
     */
    public void writeRow(List<String> rowValues, Row row)
    {
        for (int i = 0; i < rowValues.size(); i++)
        {
            String value = rowValues.get(i);
            if (value == null)
                continue;
            row.createCell(i).setCellValue(value);
        }
    }

    public Workbook getWb()
    {
        return wb;
    }

    public void setWb(Workbook wb)
    {
        this.wb = wb;
    }

    /**
     * @throws IOException
     */
    public void writeToFile() throws IOException
    {
        try (FileOutputStream fos = new FileOutputStream(this.excelFile))
        {
            this.wb.write(fos);
        }
    }

    /**
     * @param rsmd
     * @return
     * @throws SQLException
     * @throws ClassNotFoundException
```

```
     */
    private List<Class<?>> getClassesList(ResultSetMetaData rsmd) throws
SQLException, ClassNotFoundException
    {
        int columnCount = rsmd.getColumnCount();
        List<Class<?>> res = new ArrayList<Class<?>>();

        for (int i = 1; i <= columnCount; i++)
        {
            res.add(Class.forName(rsmd.getColumnClassName(i)));
        }
        return res;
    }

    /**
     * @param rsmd
     * @return
     * @throws SQLException
     * @throws ClassNotFoundException
     */
    private List<String> getColumnHeaders(ResultSetMetaData rsmd) throws
SQLException, ClassNotFoundException
    {
        int columnCount = rsmd.getColumnCount();
        List<String> labels = new ArrayList<String>();

        for (int i = 1; i <= columnCount; i++)
        {
            labels.add(rsmd.getColumnLabel(i));
        }
        return labels;
    }

    /**
     * @param sheet
     * @return
     */
    public Sheet autoSizeColumns(Sheet sheet)
    {
        int columsTotal = sheet.getRow(0).getLastCellNum();
        for(int i=0; i<columsTotal; i++)
        {
            sheet.autoSizeColumn(i);
        }
```

```
            return sheet;
        }
}

package ru.pii.toolWearing.excel;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;

import java.sql.Statement;
import java.sql.Timestamp;
import java.text.SimpleDateFormat;

import org.apache.poi.hssf.usermodel.HSSFCellStyle;
import org.apache.poi.ss.usermodel.CellStyle;
import org.apache.poi.ss.usermodel.CreationHelper;
import org.apache.poi.ss.usermodel.IndexedColors;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;

import ru.pii.toolWearing.excel.formatters.AbstractFormatter;

import java.util.*;

public class test
{
        public static void main(String[] args) throws Exception
        {
                Long st = System.currentTimeMillis();

                //readFromDbToExcel();
                //readFromDbToExcel1();
                //readFromExcelWriteToExcel();
                writeListToExcel();
                //writeListOfListsToExcel();
                //readFromExcel1();
                //exampleOfCreatingDataTableAndWritingToExcel();
                //System.out.println(System.currentTimeMillis());

                System.out.println("Общее время выполнения программы (ms): " +
(System.currentTimeMillis() - st));
```

```
	}

	public static void readFromExcel1() throws Exception
	{
		ExcelReader er = new
ExcelReader("C:\\Users\\ipupkevich\\Desktop\\1.xlsx");
		List<List<String>> table1 = er.readSheet(0, new int[] { 0, 1, 2 });
		for (List<String> row : table1)
		{
			for (String val : row)
			{
				Double d = Double.parseDouble(val);
				System.out.print(d + " | ");
			}
			System.out.println();
		}

		ExcelWriter ew = new
ExcelWriter("C:\\Users\\ipupkevich\\Desktop\\2.xlsx");
		ew.createSheet("dsadsad", null, table1);
		ew.writeToFile();

	}

	/**
	 * @throws Exception
	 */
	public static void readFromDbToExcel() throws Exception
	{
		Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
		try (Connection conn =
DriverManager.getConnection("jdbc:sqlserver://wind-
hyperv;dataBaseName=wind;user=viewer;password=Wind1234");
				Statement statement = conn.createStatement();
				ResultSet resultSet = statement.executeQuery("SELECT
TOP 1000 * FROM WTPartMaster");)
		{
			ExcelWriter ew = new
ExcelWriter("C:\\Users\\ipupkevich\\Desktop\\ResultSet.xls");
			ew.autoSizeColumns(ew.createSheet("WTPart", resultSet));
			ew.writeToFile();
		}
	}
```

```
	/**
	 * @throws Exception
	 */
	public static void readFromDbToExcel1() throws Exception
	{
		//String selStr = "SELECT DISTINCT jbr.wc, wc.description,
jbr.RUSOperCode, jbr.RUSOperDesc FROM jobroute jbr INNER JOIN wc ON
wc.wc = jbr.wc WHERE jbr.RUSOperCode IS NOT NULL ORDER by jbr.wc,
jbr.RUSOperCode ";
		String selStr = "SELECT * FROM item WHERE
item='1100106084000006 70' ";

		Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
		try (Connection conn =
DriverManager.getConnection("jdbc:sqlserver://SLDB\\TRB_SQL;dataBaseName=
Trb_erp;user=Otchet;password=Jnx1234g");
				Statement statement = conn.createStatement();
				ResultSet resultSet = statement.executeQuery(selStr);)
		{
			ExcelWriter ew = new
ExcelWriter("C:\\Users\\ipupkevich\\Desktop\\IIIIII\\ResultSet.xls");
			ew.createSheet("Операции ERP", resultSet);
			ew.writeToFile();
		}
	}

	@SuppressWarnings("deprecation")
	public static void readFromExcelWriteToExcel() throws Exception
	{
		List<Class<?>> columnTypes = new ArrayList<Class<?>>();
		int[] columns = new int[16];
		for (int i = 0; i < columns.length; i++)
		{
			columns[i] = i;
			switch (i)
			{
				case 1:
					columnTypes.add(Float.class);
					break;
				case 14:
					columnTypes.add(Timestamp.class);
					break;
				default:
					columnTypes.add(String.class);
```

```
                        break;
                }
            }

            ExcelReader er = new
ExcelReader("C:\\Users\\ipupkevich\\Desktop\\1.xls");
            DataTable dt = er.readSheet(0, columns, columnTypes);

            ExcelWriter ew = new
ExcelWriter("C:\\Users\\ipupkevich\\Desktop\\IIIIII\\dataTable.xls");
            final CellStyle styleRed = ew.getWb().createCellStyle();

        styleRed.setFillForegroundColor(IndexedColors.YELLOW.getIndex());
            styleRed.setFillPattern(CellStyle.SOLID_FOREGROUND);
            dt.getColumnFormatters().get(3).addListenerOnSetCellData(new
ActionListener()
            {
                @Override
                public void actionPerformed(ActionEvent e)
                {
                    AbstractFormatter fmt = (AbstractFormatter)
e.getSource();
                    if (fmt.getValue().toString().endsWith("977-88"))
                    {
                        fmt.getCell().setCellStyle(styleRed);
                    }
                }
            });
            ew.createSheet("Таблица", dt);
            ew.writeToFile();
        }

        public static void writeListToExcel() throws Exception
        {
            List<String> list = new ArrayList<String>();

            for (int i = 0; i < 65000; i++)
            {
                list.add("запись 2");
            }
            System.out.println("массив данных сгенерирован");
            ExcelWriter ew = new
ExcelWriter("C:\\Users\\ipupkevich\\Desktop\\IIIIII\\list.xls");
            ew.createSheet("SheetWithList", "header", list);
```

```
            //((XSSFSheet) sheet).protectSheet("1234");
            ew.writeToFile();
        }

        public static void writeListOfListsToExcel() throws Exception
        {
            List<String> headers = new ArrayList<String>();
            headers.add("заголовок 1");
            headers.add("заголовок 2");
            List<String> list = new ArrayList<String>();
            list.add("запись 1");
            list.add("запись 2");
            List<List<String>> table = new ArrayList<List<String>>();
            table.add(list);
            table.add(list);
            ExcelWriter ew = new
ExcelWriter("C:\\Users\\ipupkevich\\Desktop\\IIIIII\\listOfList.xls");
            ew.createSheet("SheetWithList", headers, table);
            ew.writeToFile();
        }

        /**
         *
         * @throws Exception
         */
        @SuppressWarnings("deprecation")
        public static void exampleOfCreatingDataTableAndWritingToExcel() throws
Exception
        {
            /*
            ExcelWriter ew = new
ExcelWriter("C:\\Users\\ipupkevich\\Desktop\\dataTable.xls");
            Workbook wb = ew.getWb();


            List<String> columnHeaders = new ArrayList<String>();
            columnHeaders.add("Имя");
            columnHeaders.add("Возвраст");
            columnHeaders.add("Дата");

            List<Class<?>> columnClasses = new ArrayList<Class<?>>();
            columnClasses.add(String.class);
            columnClasses.add(Integer.class);
            columnClasses.add(Date.class);
```

```
		/*Создаем DataTable*/
		DataTable dt = new DataTable(columnHeaders, columnClasses);

		/*Создаем необходимые стили ячеек (см. Apache POI для подробностей)*/
		final CellStyle styleOrange = wb.createCellStyle();/*Оранжевая ячейка*/

	styleOrange.setFillForegroundColor(IndexedColors.ORANGE.getIndex());
		styleOrange.setFillPattern(CellStyle.SOLID_FOREGROUND);
		styleOrange.setBorderBottom(HSSFCellStyle.BORDER_THIN);
		styleOrange.setBorderLeft(HSSFCellStyle.BORDER_THIN);
		styleOrange.setBorderRight(HSSFCellStyle.BORDER_THIN);
		styleOrange.setBorderTop(HSSFCellStyle.BORDER_THIN);
		final CellStyle styleGreen = wb.createCellStyle();/*Зеленая ячейка*/
		styleGreen.setFillForegroundColor(IndexedColors.GREEN.getIndex());
		styleGreen.setFillPattern(CellStyle.SOLID_FOREGROUND);
		final CellStyle styleDateGray = wb.createCellStyle();/*Серая ячейка с форматированием даты*/

	styleDateGray.setFillForegroundColor(IndexedColors.GREY_25_PERCENT.getIndex());
		styleDateGray.setFillPattern(CellStyle.SOLID_FOREGROUND);
		CreationHelper ch = wb.getCreationHelper();

	styleDateGray.setDataFormat(ch.createDataFormat().getFormat("yyyy.MM.dd HH:mm")); /*это нужно потому что по умолчанию ячейки даты делаются именно в этом формате, но как только мы меняем стиль на свой, то форматирование пропадает*/


		dt.getColumnFormatters().get(0).addListenerOnSetCellData(new ActionListener()
		{
			@Override
			public void actionPerformed(ActionEvent e)
			{
				AbstractFormatter fmt = (AbstractFormatter) e.getSource();
				/*Из форматтера вытягиваем значение, приводим к верхнему регистру и смотрим начинается ли оно на И*/
				if (fmt.getValue().toString().toUpperCase().startsWith("И"))
```

```
                {

        fmt.getCell().setCellStyle(styleOrange);/*Устанавливаем стиль ячейки*/
                }
            }
        });

        dt.getColumnFormatters().get(1).addListenerOnSetCellData(new
ActionListener()
        {
            @Override
            public void actionPerformed(ActionEvent e)
            {
                AbstractFormatter fmt = (AbstractFormatter)
e.getSource();
                /*Из форматтера вытягиваем значение, кастуем как int
и сравниваем*/
                if (((int) fmt.getValue()) >= 18)
                {

        fmt.getCell().setCellStyle(styleGreen);/*Устанавливаем стиль ячейки*/
                }
            }
        });

        final SimpleDateFormat sdf = new SimpleDateFormat("dd.MM.yyyy
HH:mm");


        dt.getColumnFormatters().get(2).addListenerOnSetCellData(new
ActionListener()
        {
            Date startDate = sdf.parse("16.06.2017 00:00");
            Date endDate = sdf.parse("20.06.2017 23:59");

            @Override
            public void actionPerformed(ActionEvent e)
            {
                AbstractFormatter fmt = (AbstractFormatter)
e.getSource();
                /*Из форматтера вытягиваем дату*/
                Date cellVal = (Date) fmt.getValue();
                if (cellVal.compareTo(startDate) >= 0 &&
cellVal.compareTo(endDate) <= 0)
```

```
                        {

        fmt.getCell().setCellStyle(styleDateGray);/*Устанавливаем стиль*/
                        }
                    }
                });

                List<List<Object>> table = new ArrayList<List<Object>>();


                String[] names = new String[] { "Илья", "илья", "Дима", "Олег" };
                Integer[] ages = new Integer[] { 25, 17, 18, 22 };
                Date[] dates = new Date[] { sdf.parse("16.06.2017 00:00"),
sdf.parse("15.06.2017 23:59"), sdf.parse("16.05.2017 00:00"),
sdf.parse("20.06.2017 23:59") };
                for (int i = 0; i < 4; i++)
                {
                    List<Object> row = new ArrayList<Object>();
                    row.add(names[i]);
                    row.add(ages[i]);
                    row.add(dates[i]);
                    table.add(row);
                }

                dt.setTable(table);

                /*Создаем в экселька лист с именем Мой красивый лист и подго-
няем ширину столбцов по содержимому*/
                Sheet sheet1 = ew.createSheet("Мой красивый лист", dt);
                sheet1.createFreezePane(0, 1);
                sheet1.setColumnWidth(0, 18); //set column width(number of symbols,
width of symbol gets from the first font that used in this workbook)
                sheet1.setColumnWidth(1, 60);
                sheet1.setColumnWidth(2, 60);
                ew.writeToFile();
        }

}



package ru.pii.toolWearing.gui;

import java.awt.Choice;
```

```
import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.SwingWorker;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import javax.swing.filechooser.FileNameExtensionFilter;

import ru.pii.toolWearing.Main;
import ru.pii.toolWearing.loger.Console;
import ru.pii.toolWearing.models.Parameters;
import ru.pii.toolWearing.models.ToolPathCalculation;
import ru.pii.toolWearing.utils.ExportImport;

public class InputTab extends JPanel
{
	public InputTab() {
		setLayout(null);

		JLabel label = new JLabel("Траектория:");
		label.setBounds(10, 11, 108, 14);
		add(label);

		final JLabel toolPathFileLabel = new JLabel("");
		toolPathFileLabel.setBackground(Color.GRAY);
		toolPathFileLabel.setBounds(10, 36, 435, 14);
		add(toolPathFileLabel);

		JLabel label_2 = new JLabel("Поверхность:");
		label_2.setBounds(10, 57, 108, 14);
		add(label_2);

		final JLabel surfaceFileLabel = new JLabel("");
		surfaceFileLabel.setBackground(Color.GRAY);
		surfaceFileLabel.setBounds(10, 86, 435, 14);
		add(surfaceFileLabel);
```

```
		final JLabel wearingFileLabel = new JLabel("");
		wearingFileLabel.setBackground(Color.GRAY);
		wearingFileLabel.setBounds(10, 134, 432, 14);
		add(wearingFileLabel);

		final Choice choice_1 = new Choice();
		choice_1.setBounds(195, 55, 98, 20);
		add(choice_1);

		choice_1.add(ExportImport.MILLIMETER);
		choice_1.add(ExportImport.MICROMETER);

		JButton btnExcel_1 = new JButton("excel");
		btnExcel_1.addActionListener(new ActionListener()
		{
			public void actionPerformed(ActionEvent arg0)
			{
				JFileChooser chooser = new JFileChooser();
				FileNameExtensionFilter filter = new
FileNameExtensionFilter("Excel files (.xls, .xlsx)", "xls", "xlsx");
				chooser.setFileFilter(filter);
				chooser.setCurrentDirectory(Main.toolPathDir);
				chooser.setDialogTitle("Выберите Excel файл");
				choos-
er.setFileSelectionMode(JFileChooser.FILES_ONLY);
				chooser.setAcceptAllFileFilterUsed(false);
				if (chooser.showOpenDialog(null) ==
JFileChooser.APPROVE_OPTION)
				{

	surfaceFileLabel.setText(chooser.getSelectedFile().getAbsolutePath());
				}
			}
		});
		btnExcel_1.setBounds(106, 57, 68, 18);
		add(btnExcel_1);

		JLabel label_3 = new JLabel("Растояние, при котором инструмент
уже совершает холостой ход, мкм");
		label_3.setBounds(10, 111, 390, 19);
		add(label_3);

		textField = new JTextField();
```

```
		textField.setBounds(10, 140, 86, 20);
		add(textField);
		textField.setColumns(10);
		textField.getDocument().addDocumentListener(new
DocumentListener()
		{

			@Override
			public void removeUpdate(DocumentEvent arg0)
			{
				updateDist();
			}

			@Override
			public void insertUpdate(DocumentEvent arg0)
			{
				updateDist();
			}

			@Override
			public void changedUpdate(DocumentEvent arg0)
			{
				updateDist();
			}

			public void updateDist()
			{
				try
				{
					Parameters.distanceOfIdleTransaction =
Long.parseLong(textField.getText());
				} catch (Exception e)
				{
					Console.printLine("Нужно целое число");
				}
			}
		});

		JButton button_3 = new JButton("Обработать");
		button_3.addActionListener(new ActionListener()
		{
			public void actionPerformed(ActionEvent arg0)
			{
```

```
                        try
                        {
                                @SuppressWarnings("rawtypes")
                                SwingWorker myWorker = new
SwingWorker<String, Void>()
                                {
                                        protected String doInBackground() throws
Exception
                                        {
                                                Main.surface =
ExportImport.importSurfaceFromExcel(surfaceFileLabel.getText(),
choice_1.getSelectedItem());
                                                Main.surface.devideWearingZones(3);
                                                table.repaint();
                                                return null;
                                        }
                                };
                                myWorker.execute();
                        } catch (Exception e)
                        {
                                Console.printLine(e.getMessage());
                        }
                }
        });
        button_3.setBounds(327, 52, 118, 23);
        add(button_3);

        JButton btnGcode = new JButton("gcode");
        btnGcode.addActionListener(new ActionListener()
        {
                public void actionPerformed(ActionEvent arg0)
                {
                        JFileChooser chooser = new JFileChooser();
                        FileNameExtensionFilter filter = new
FileNameExtensionFilter("G-code (.tap)", "tap");
                        chooser.setFileFilter(filter);
                        chooser.setCurrentDirectory(Main.toolPathDir);
                        chooser.setDialogTitle("Выберите Tap файл");
                        choos-
er.setFileSelectionMode(JFileChooser.FILES_ONLY);
                        chooser.setAcceptAllFileFilterUsed(false);
                        if (chooser.showOpenDialog(null) ==
JFileChooser.APPROVE_OPTION)
                        {
```

```
        toolPathFileLabel.setText(chooser.getSelectedFile().getAbsolutePath());

                                try
                                {
                                        @SuppressWarnings("rawtypes")
                                        SwingWorker myWorker = new
SwingWorker<String, Void>()
                                        {
                                                protected String doInBackground()
throws Exception
                                                {
                                                        Main.toolPath =
ExportImport.importToolPathFromFile(chooser.getSelectedFile());
                                                        Main.toolPathDir = choos-
er.getSelectedFile().getParentFile();
                                                        return null;
                                                }
                                        };
                                        myWorker.execute();
                                } catch (Exception e)
                                {
                                        Console.printLine(e.getMessage());
                                }
                        }
                }
        });
        btnGcode.setBounds(107, 9, 68, 19);
        add(btnGcode);

        JButton btnNewButton = new JButton("Расчитать траекторию");
        btnNewButton.addActionListener(new ActionListener()
        {
                public void actionPerformed(ActionEvent arg0)
                {
                        try
                        {
                                @SuppressWarnings("rawtypes")
                                SwingWorker myWorker = new
SwingWorker<String, Void>()
                                {
                                        protected String doInBackground() throws
Exception
                                        {
```

```
                                        Main.calc = new
ToolPathCalculation(Main.toolPath, Main.surface);
                                        Main.calc.calculateInOneStep();
                                        return null;
                                    }
                                };
                                myWorker.execute();
                            } catch (Exception e)
                            {
                                Console.printLine(e.getMessage());
                            }
                        }
                });
                btnNewButton.setBounds(600, 188, 209, 42);
                add(btnNewButton);

                JScrollPane scrollPane = new JScrollPane();
                scrollPane.setBounds(460, 11, 359, 163);
                add(scrollPane);
                WearingTableModel tableModel = new WearingTableModel();
                table = new JTable(tableModel);
                scrollPane.setViewportView(table);
                table.setVisible(true);
                tableModel.fireTableDataChanged();

        }

        private static final long serialVersionUID = 7467231050932452267L;
        private JTextField textField;
        private JTable table;
}


package ru.pii.toolWearing.gui;

import java.awt.Dimension;

import javax.swing.DefaultListModel;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTabbedPane;
```

```
import javax.swing.SwingUtilities;

import ru.pii.toolWearing.loger.Console;

public class MainFrame extends JFrame
{
	public MainFrame() {
		setSize(new Dimension(900, 540));
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setLocationRelativeTo(null);
		getContentPane().setLayout(null);

		JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.TOP);
		tabbedPane.setBounds(10, 11, 864, 297);
		getContentPane().add(tabbedPane);

		JPanel inputPanel = new InputTab();
		tabbedPane.addTab("Входные данные", null, inputPanel, null);

		JPanel outputPanel = new OutputTab();
		tabbedPane.addTab("Выходные данные", null, outputPanel, null);

		JLabel label = new JLabel("Информационное окно:");
		label.setBounds(10, 318, 201, 14);
		getContentPane().add(label);

		DefaultListModel<String> listModel = new
DefaultListModel<String>();
		JList<String> listOfTables = new JList<String>(listModel);
		JScrollPane scrollPane = new JScrollPane(listOfTables);
		scrollPane.setBounds(10, 342, 864, 149);
		Console.addListenerOnPrintLine(e ->
		{
			SwingUtilities.invokeLater(new Runnable()
			{
				@Override
				public void run()
				{
					listModel.addElement(e.getActionCommand());

	scrollPane.getVerticalScrollBar().setValue(scrollPane.getVerticalScrollBar().
getMaximum());
				}
			});
```

```
            });

            Console.addListenerReplaceLastLine(e ->
            {
                SwingUtilities.invokeLater(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        listModel.setElementAt(e.getActionCommand(),
listModel.size() - 1);
                    }
                });
            });

            listOfTables.setLayoutOrientation(JList.VERTICAL);
            getContentPane().add(scrollPane);

            setVisible(true);

        }

        private static final long serialVersionUID = -65977682240931997 85L;
}


package ru.pii.toolWearing.gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JPanel;
import javax.swing.SwingWorker;

import ru.pii.toolWearing.Main;
import ru.pii.toolWearing.loger.Console;
import ru.pii.toolWearing.utils.ExportImport;

public class OutputTab extends JPanel
{
        public OutputTab() {
            setLayout(null);
```

```
		JButton button = new JButton("Выгрузить расчет");
		button.addActionListener(new ActionListener()
		{
			public void actionPerformed(ActionEvent arg0)
			{
				try
				{
					@SuppressWarnings("rawtypes")
					SwingWorker myWorker = new
SwingWorker<String, Void>()
					{
						protected String doInBackground() throws
Exception
						{

	ExportImport.exportCalculatedData(Main.toolPathDir.getAbsolutePath() +
"\\calc.xls", Main.calc);
							return null;
						}
					};
					myWorker.execute();
				} catch (Exception e)
				{
					Console.printLine("" + e.getMessage());
				}
			}
		});
		button.setBounds(10, 11, 244, 23);
		add(button);

		JButton button_1 = new JButton("Выгрузить старую траекторию");
		button_1.addActionListener(new ActionListener()
		{
			public void actionPerformed(ActionEvent arg0)
			{
				try
				{
					@SuppressWarnings("rawtypes")
					SwingWorker myWorker = new
SwingWorker<String, Void>()
					{
						protected String doInBackground() throws
Exception
						{
```

```
        ExportImport.exportToolPath(Main.toolPathDir.getAbsolutePath() +
"\\oldToolPath.tap", Main.toolPath);
                                        return null;
                                    }
                                };
                                myWorker.execute();
                            } catch (Exception e)
                            {
                                Console.printLine("" + e.getMessage());
                            }
                        }
                    });
                    button_1.setBounds(10, 45, 244, 23);
                    add(button_1);

                    JButton button_2 = new JButton("Выгрузить новую траекторию");
                    button_2.addActionListener(new ActionListener()
                    {
                        public void actionPerformed(ActionEvent arg0)
                        {
                            try
                            {
                                @SuppressWarnings("rawtypes")
                                SwingWorker myWorker = new
SwingWorker<String, Void>()
                                {
                                    protected String doInBackground() throws
Exception
                                    {
                                        if (Main.calc == null)
                                            throw new
Exception("Траектории нет");

        ExportImport.exportToolPath(Main.toolPathDir.getAbsolutePath() +
"\\newToolPath.tap", Main.calc.getModifiedToolPath());
                                        return null;
                                    }
                                };
                                myWorker.execute();
                            } catch (Exception e)
                            {
                                Console.printLine("" + e.getMessage());
                            }
```

```
                }
            });
            button_2.setBounds(10, 87, 244, 23);
            add(button_2);

        }

        private static final long serialVersionUID = 7467231050932452244L;

}

package ru.pii.toolWearing.gui;

import java.util.ArrayList;
import java.util.List;

import javax.swing.table.AbstractTableModel;

import ru.pii.toolWearing.Main;
import ru.pii.toolWearing.loger.Console;

public class WearingTableModel extends AbstractTableModel
{
        private static final long serialVersionUID = 10223686221520559l5L;

        private List<Long[]> zatychka = new ArrayList<Long[]>();

        public WearingTableModel() {
            zatychka.add(new Long[] { 0L, 0L, 0L, 0L });
            zatychka.add(new Long[] { 0L, 0L, 0L, 0L });
        }

        @Override
        public int getColumnCount()
        {
            // TODO Auto-generated method stub
            return 4;
        }

        @Override
        public int getRowCount()
        {
            return 3;
        }
```

```
@Override
public Object getValueAt(int row, int col)
{

	if (Main.surface != null)
	{
		if (row != 0)
		{
			return zatychka.get(row - 1)[col];
		} else
		{
			if (col == 0)
				return Main.surface.getWearingZones().get(1)[2];
			if (col == 1)
				return Main.surface.getWearingZones().get(col)[3];
			if (col == 2)
				return Main.surface.getWearingZones().get(col)[3];
			if (col == 3)
				return Main.surface.getWearingZones().get(col)[3];
		}
	}
	return null;
}

@Override
public boolean isCellEditable(int row, int col)
{
	return true;
}

@Override
public void setValueAt(Object val, int row, int col)
{
	if (Main.surface != null)
	{
		try
		{
			if (row != 0) //если не первая строчка, то берем данные
из затычки
			{
				Long[] temp = zatychka.get(row-1);
				temp[col] = Long.parseLong((String) val);
				zatychka.set(row-1, temp);
```

```
                        } else
                        {
                                if (col == 0) //если колонка с дистанцией
                                {
                                        // меняем во всех трех зонах пройденную
дистанцию:
                                        Long[] modelRow =
Main.surface.getWearingZones().get(1);
                                        modelRow[2] = Long.parseLong((String) val);
                                        Main.surface.setWearingZone(1, modelRow);

                                        modelRow =
Main.surface.getWearingZones().get(2);
                                        modelRow[2] = Long.parseLong((String) val);
                                        Main.surface.setWearingZone(2, modelRow);

                                        modelRow =
Main.surface.getWearingZones().get(3);
                                        modelRow[2] = Long.parseLong((String) val);
                                        Main.surface.setWearingZone(3, modelRow);
                                        fireTableCellUpdated(row, col);
                                } else if (col == 1 || col == 2 || col == 3)
                                {
                                        Long[] modelRow =
Main.surface.getWearingZones().get(col);
                                        modelRow[3] = Long.parseLong((String) val);
                                        Main.surface.setWearingZone(col,
modelRow);
                                        fireTableCellUpdated(row, col);
                                }
                        }
                } catch (Exception e)
                {
                        Console.printLine("Допускается только целочисленное
число");
                }

        }
    }

    @Override
    public String getColumnName(int col)
    {
        if (col == 0)
```

```
                return "L(мкм)";
            if (col == 1)
                return "\u0394R1(мкм)";
            if (col == 2)
                return "\u0394R2(мкм)";
            if (col == 3)
                return "\u0394R3(мкм)";
            return "";
        }
}

package ru.pii.toolWearing.loger;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.event.EventListenerList;

public class Console
{
        private static EventListenerList onPrintLineListeners = null;
        private static EventListenerList onReplaceLastLineListeners = null;

        /**
         * @param obj
         */
        public static void printLine(Object obj)
        {
            if (onPrintLineListeners == null || obj==null)
                return;
            for (ActionListener listener :
onPrintLineListeners.getListeners(ActionListener.class))
            {
                listener.actionPerformed(new ActionEvent(listener,
ActionEvent.ACTION_PERFORMED, obj.toString()));
            }
        }

        /**
         * @param obj
         */
        public static void replaceLastLine(Object obj)
        {
            if (onReplaceLastLineListeners == null || obj==null)
```

```
                return;
            for (ActionListener listener :
onReplaceLastLineListeners.getListeners(ActionListener.class))
            {
                listener.actionPerformed(new ActionEvent(listener,
ActionEvent.ACTION_PERFORMED, obj.toString()));
            }
        }

        /**
         * @param listener
         */
        public static void addListenerOnPrintLine(ActionListener listener)
        {
            if (onPrintLineListeners == null)
                onPrintLineListeners = new EventListenerList();
            onPrintLineListeners.add(ActionListener.class, listener);
        }

        public static void removeListenerOnPrintLine(ActionListener listener)
        {
            if (onPrintLineListeners == null)
                return;
            onPrintLineListeners.remove(ActionListener.class, listener);
        }

        /**
         * @param listener
         */
        public static void addListenerReplaceLastLine(ActionListener listener)
        {
            if (onReplaceLastLineListeners == null)
                onReplaceLastLineListeners = new EventListenerList();
            onReplaceLastLineListeners.add(ActionListener.class, listener);
        }

        public static void removeListenerReplaceLastLine(ActionListener listener)
        {
            if (onReplaceLastLineListeners == null)
                return;
            onReplaceLastLineListeners.remove(ActionListener.class, listener);
        }
}
```

```
package ru.pii.toolWearing.loger;

public class NamedLog
{
	private String name;

	public NamedLog(String name)
	{
		this.name = name;
	}

	public void printLine(Object obj)
	{
		Console.printLine(this.name + obj.toString());
	}

	public void replaceLastLine(Object obj)
	{
		Console.replaceLastLine(this.name + obj.toString());
	}
}

package ru.pii.toolWearing.loger;

public class ProgressLog
{
	private String name;
	private int takts;
	private int currentTact=0;
	private double currentPercent =0;
	private double incrementPercent =1;
	private long oldPercent=0;

	/**
	 */
	public ProgressLog(String name, int takts)
	{
		this.name = name;
		this.takts = takts;
		this.incrementPercent = 100/(double)takts;
		Console.printLine(this.name + oldPercent + "%");
	}

	/**
```

```
	 */
	public void tick()
	{
		this.currentTact++;
		if(this.currentTact<=this.takts)
		{
			this.currentPercent+=incrementPercent;
		}
		if(oldPercent<Math.round(currentPercent))
		{
			oldPercent = Math.round(currentPercent);
			Console.replaceLastLine(this.name + oldPercent + "%");
		}
	}
}

package ru.pii.toolWearing.models;

public class Parameters
{
	public static long distanceOfIdleTransaction;

}

package ru.pii.toolWearing.models;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.HashMap;
import java.util.Map;

 *
 *
 */
public class Point2D {

	private long xy;
	private long z;

	private HashMap<String,Point2D> associatedPointsOrVectors;

	/**
	 * @param x
	 * @param y
```

```
     * @param z
     */
    public Point2D(long xy,long z)
    {
        this.xy = xy;
        this.z = z;
    }

    /**
     * @param point
     * @return
     */
    public long distance(Point2D point)
    {
        return Math.round(Math.sqrt((xy-point.getXY())*(xy-point.getXY()) +
(z-point.getZ())*(z-point.getZ())));
    }

    /**
     * @param point
     * @return
     */
    public long length()
    {
        return Math.round(Math.sqrt(xy*xy + z*z));
    }

    /**
     * @param point
     * @return
     */
    public long distanceSquared(Point2D point)
    {
        return ((xy-point.getXY())*(xy-point.getXY()) + (z-point.getZ())*(z-
point.getZ()));
    }

    /**
     * @param vector
     * @return
     */
    public Point2D subtractVector(Point2D vector)
    {
        return new Point2D(this.xy-vector.getXY(), this.z-vector.getZ());
```

```
	}

	/**
	 * @param vector
	 * @return
	 */
	public Point2D addVector(Point2D vector)
	{
		return new Point2D(this.xy+vector.getXY(),  this.z+vector.getZ());
	}

	/**
	 * @param vector
	 * @return
	 */
	public Point2D multiplyVector(double scalar)
	{
		return new Point2D(Math.round(this.xy*scalar),
Math.round(this.z*scalar));
	}

	/**
	 * @param newLength
	 * @return
	 */
	public Point2D resizeVector(long newLength)
	{
		double rel = (double)newLength/this.length();
		return new Point2D(Math.round(this.xy*rel), Math.round(this.z*rel));
	}

	/**
	 * @return
	 */
	public double getAngleToAxelZ()
	{
		return new BigDecimal(Math.atan2(this.xy,
this.z)*180/Math.PI).setScale(3,RoundingMode.HALF_UP).doubleValue();
	}

	/**
	 */
	public long getXY() {
		return xy;
```

```
}

/**
 */
public void setXY(long xy) {
	this.xy = xy;
}

/**
 */
public long getZ() {
	return z;
}

/**
 */
public void setZ(long z) {
	this.z = z;
}

/**
 */
@Override
public boolean equals(Object arg0) {
	if(arg0 instanceof Point2D)
	{
		Point2D p = (Point2D)arg0;
		if(this.xy == p.getXY()  && this.z == p.getZ())
			return true;
	}
	return false;
}

/**
 *  hash = (x*7+y)*7+z
 */
@Override
public int hashCode() {
	long result = this.xy;
	result = result*7 + this.z;
	return (int)result;
}

@Override
```

```
	public String toString() {
		StringBuilder sb = new StringBuilder("[");
		sb.append(this.xy);
		sb.append(",");
		sb.append(this.z);
		sb.append("]");
		return sb.toString();
	}

	/**
	 * @return
	 */
	public Map<String,Point2D> getAssociatedPointsOrVectors() {
		return associatedPointsOrVectors;
	}

	/**
	 * @param name
	 * @param point
	 */
	public void putAssociatedPointOrVector(String name, Point2D point) {
		if(this.associatedPointsOrVectors==null)
			this.associatedPointsOrVectors = new HashMap<String,
Point2D>();
		this.associatedPointsOrVectors.put(name, point);
	}

	/**
	 * @param name
	  */
	public Point2D getAssociatedPointOrVector(String name) {
		return this.associatedPointsOrVectors!=null?
this.associatedPointsOrVectors.get(name):null;
	}
}

package ru.pii.toolWearing.models;

import java.util.HashMap;
import java.util.Map;

/**
 */
public class Point3D {
```

```
	private long x;
	private long y;
	private long z;

	private HashMap<String,Point3D> associatedPointsOrVectors;

	/**
	 * @param x
	 * @param y
	 * @param z
	 */
	public Point3D(long x, long y, long z)
	{
		this.x = x;
		this.y = y;
		this.z = z;
	}

	/** * @param point
	 * @return
	 */
	public long distance(Point3D point)
	{
		return Math.round(Math.sqrt((x-point.getX())*(x-point.getX()) + (y-
point.getY())*(y-point.getY()) + (z-point.getZ())*(z-point.getZ())));
	}

	/**
	 * @param point
	 * @return
	 */
	public long length()
	{
		return Math.round(Math.sqrt(x*x + y*y + z*z));
	}

	/**
	 * @param point
	 * @return
	 */
	public long distanceSquared(Point3D point)
	{
```

```
            return ((x-point.getX())*(x-point.getX()) + (y-point.getY())*(y-
point.getY()) + (z-point.getZ())*(z-point.getZ()));
        }

        /**
         * @param vector
         * @return
         */
        public Point3D subtractVector(Point3D vector)
        {
            return new Point3D(this.x-vector.getX(), this.y-vector.getY(), this.z-
vector.getZ());
        }

        /**
         * @param vector
         * @return
         */
        public Point3D addVector(Point3D vector)
        {
            return new Point3D(this.x+vector.getX(), this.y+vector.getY(),
this.z+vector.getZ());
        }

        /**
         * @param vector
         * @return
         */
        public Point3D multiplyVector(double scalar)
        {
            return new Point3D(Math.round(this.x*scalar),
Math.round(this.y*scalar), Math.round(this.z*scalar));
        }

        /**
         * @param newLength
         * @return
         */
        public Point3D resizeVector(long newLength)
        {
            double rel = (double)newLength/(double)this.length();
            return new Point3D(Math.round((double)this.x*rel),
Math.round((double)this.y*rel), Math.round((double)this.z*rel));
        }
```

```
/**
 */
public long getX() {
	return x;
}

/**
 */
public void setX(long x) {
	this.x = x;
}
/**
 * @return
 */
public long getY() {
	return y;
}

/**
 * @param y
 */
public void setY(long y) {
	this.y = y;
}
/**
 * @return
 */
public long getZ() {
	return z;
}

/**
 * @param z
 */
public void setZ(long z) {
	this.z = z;
}

/**
 */
@Override
public boolean equals(Object arg0) {
	if(arg0 instanceof Point3D)
```

```
            {
                Point3D p = (Point3D)arg0;
                if(this.x == p.getX() && this.y == p.getY() && this.z ==
p.getZ())
                    return true;
            }
            return false;
        }

        /**
         *  hash = (x*7+y)*7+z
         */
        @Override
        public int hashCode() {
            long result = this.x;
            result = result*7 + this.y;
            result = result*7 + this.z;
            return (int)result;
        }

        @Override
        public String toString() {
            StringBuilder sb = new StringBuilder();
            sb.append("X");
            sb.append(this.x);
            sb.append(" Y");
            sb.append(this.y);
            sb.append(" Z");
            sb.append(this.z);
            return sb.toString();
        }

        public String toStringMillimeters() {
            StringBuilder sb = new StringBuilder();
            sb.append("X");
            sb.append((double)this.x/1000);
            sb.append(" Y");
            sb.append((double)this.y/1000);
            sb.append(" Z");
            sb.append((double)this.z/1000);
            return sb.toString();
        }

        /**
```

```
	 * @return
	 */
	public Map<String,Point3D> getAssociatedPointsOrVectors() {
		return associatedPointsOrVectors;
	}

	/**
	 * @param name
	 * @param point
	 */
	public void putAssociatedPointOrVector(String name, Point3D point) {
		if(this.associatedPointsOrVectors==null)
			this.associatedPointsOrVectors = new HashMap<String,
Point3D>();
		this.associatedPointsOrVectors.put(name, point);
	}

	/**
	 * @param name
	 * @return
	 */
	public Point3D getAssociatedPointOrVector(String name) {
		return this.associatedPointsOrVectors!=null?
this.associatedPointsOrVectors.get(name):null;
	}
}

package ru.pii.toolWearing.models;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;

/**
 *
 */
public class Surface {

	private Set<Point3D> pointsCloud;
	private TreeMap<Integer, Long[]> wearingZones;
	private Map<Integer, Double> wearingKoeffs;
```

```
long topZ, bottomZ;

/**
 * @param pointsCloud - облако уникальных точек.
 */
public Surface(Set<Point3D> pointsCloud)
{
	if(pointsCloud!=null)
	{
		this.pointsCloud = pointsCloud;
		refreshTopAndBottomCoordinates();
		devideWearingZones(3);
	}
	else
		this.pointsCloud = new HashSet<Point3D>();
}

/**
 *
 * @param point
 * @return true , false
 */
public boolean addPoint(Point3D point)
{
	return this.pointsCloud.add(point);
}

/**
 * @return
 */
public Set<Point3D> getPointsCloud() {
	return pointsCloud;
}

/**
 * @param pointsCloud
 */
public void setPointsCloud(Set<Point3D> pointsCloud) {
	this.pointsCloud = pointsCloud;
	refreshTopAndBottomCoordinates();
	devideWearingZones(3);
}

/**
```

```
	 */
	public void refreshTopAndBottomCoordinates()
	{
		if(pointsCloud!=null)
		{
			for(Point3D point: pointsCloud)
			{
				if(point.getZ()>this.topZ)
					this.topZ=point.getZ();
				if(point.getZ()<this.bottomZ)
					this.bottomZ=point.getZ();
			}
		}
	}

	public TreeMap<Integer, Long[]> getWearingZones() {
		return wearingZones;
	}

	public Map<Integer, Double> getWearingKoeffs() {
		return wearingKoeffs;
	}

	public void setWearingZones(TreeMap<Integer, Long[]> wearingZones) {
		this.wearingZones = wearingZones;
	}

	public void setWearingZone(Integer zone, Long[] row) {
		this.wearingZones.replace(zone, row);
		calculateWearingKoeffs();
	}

	/**
	 * @param numberOfZones
	 */
	public void devideWearingZones(int numberOfZones)
	{
		long height = Math.round((((double)Math.abs(this.topZ-
this.bottomZ))/numberOfZones));
		long layerTopCoordinate=this.bottomZ;
		long layerBottomCoordinate=this.bottomZ;
		TreeMap<Integer, Long[]> wearingZones = this.getWearingZones();
		if(wearingZones==null)
			wearingZones = new TreeMap<Integer, Long[]>();
```

```
            wearingZones.clear();
            int i;
            for(i=1; i<numberOfZones; i++)
            {
                  layerTopCoordinate+=height;
                  wearingZones.put(i, new Long[] {layerBottomCoordinate,
layerTopCoordinate, 0L, 0L});
                  layerBottomCoordinate = layerTopCoordinate;
            }
            wearingZones.put(i, new Long[] {layerBottomCoordinate, this.topZ,
0L, 0L});
            this.setWearingZones(wearingZones);
            calculateWearingKoeffs();
      }

      /**
       */
      public void calculateWearingKoeffs()
      {
            this.wearingKoeffs = new HashMap<Integer, Double>();
            for(Map.Entry<Integer, Long[]> entry:
this.getWearingZones().entrySet())
            {
                  wearingKoeffs.put(entry.getKey(), ((dou-
ble)entry.getValue()[3])/entry.getValue()[2]);
            }
      }

      /**
       * @param z
       * @return
       */
      public int getNumberOfZoneByZ(long Z)
      {
            Long[] tempArray;
            for (Map.Entry<Integer, Long[]> entry : wearingZones.entrySet())
            {
                  tempArray = entry.getValue();
                  if (Z >= tempArray[0] && Z < tempArray[1])
                  {
                        return entry.getKey();
                  }
                  if (Z == tempArray[1])
                  {
```

```
                        return entry.getKey();
                    }
                }
                return -1;
        }



}

package ru.pii.toolWearing.models;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import ru.pii.toolWearing.excel.DataTable;
import ru.pii.toolWearing.loger.ProgressLog;

/**
 *
 */
public class ToolPath {

        private List<Point3D> toolPath;

        public ToolPath()
        {
                this.toolPath = new ArrayList<Point3D>();
        }

        public ToolPath(List<Point3D> toolPath)
        {
                if(toolPath!=null)
                        this.toolPath = toolPath;
                else
                        this.toolPath = new ArrayList<Point3D>();
        }

        public void addPoint(Point3D point)
        {
                this.toolPath.add(point);
        }
        public void clear(Point3D point)
```

```
	{
		this.toolPath.clear();
	}
	public List<Point3D> getToolPath()
	{
		return this.toolPath;
	}

	@Override
	public String toString() {

		StringBuilder sb = new StringBuilder();
		for(Point3D point : this.toolPath)
		{
			sb.append(point.toString());
			sb.append("\n");
		}
		return sb.toString();
	}

	/**
	 * @return
	 * @throws Exception
	 */
	public DataTable getDataTableForExcel() throws Exception
	{
		if(this.toolPath==null)
			throw new Exception("Траектория еще не расчитана");

		String[] headers = new String[]{
			"x"
			,"y"
			,"z"
		};
		Class<?>[] columns = new Class<?>[]{
			Long.class
			,Long.class
			,Long.class
		};
		DataTable dt = new DataTable(Arrays.asList(headers), Ar-
rays.asList(columns));
		List<List<Object>> table = new ArrayList<List<Object>>();
		ProgressLog pl = new ProgressLog("Подготовка данных для выгруз-
ки: ", this.toolPath.size());
```

```
                List<Object> tempRow;
                for (int i = 0; i < this.toolPath.size(); i++)
                {
                        tempRow = new ArrayList<Object>();
                        tempRow.add(this.toolPath.get(i).getX());
                        tempRow.add(this.toolPath.get(i).getY());
                        tempRow.add(this.toolPath.get(i).getZ());

                        table.add(tempRow);
                        pl.tick();
                }
                dt.setTable(table);
                return dt;
        }
}

package ru.pii.toolWearing.models;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

import ru.pii.toolWearing.excel.DataTable;
import ru.pii.toolWearing.loger.NamedLog;
import ru.pii.toolWearing.loger.ProgressLog;

public class ToolPathCalculation
{
        private List<Point3D> toolPath;
        private Surface surface;
        private List<Point3D> modifiedToolPath;
        private List<Point3D> contactPoints;
        private List<Long> distanceToSurfaceOfModel;
        private List<Long> distancesTraveled;
        private List<Point2D> vectorsOfWearing;
        private List<Point3D> vectorsOfContact;
        private List<Long> wearedAmounts;

        private Map<Integer, Double> wearingKoeffs;
        private List<Integer> pointZone;
        private List<Integer> idleToolPathPoints;
```

```
    private List<Double> anglesOfContact;
    private NamedLog console = new NamedLog
    /**

     *
     * @param toolPath
     * @param surface
     * @throws Exception
     */
    public ToolPathCalculation(ToolPath toolPath, Surface surface) throws Ex-
ception
    {
        console.printLine("Создание расчетного модуля...");
        if (toolPath == null)
            throw new Exception("Невозможно начать расчет пока не за-
гружена траектория инструмента");
        if (surface==null||surface.getPointsCloud() == null)
            throw new Exception("Невозможно начать расчет пока не за-
гружена поверхность модели");
        if (surface.getWearingKoeffs() == null)
            throw new Exception("Невозможно начать расчет пока не оп-
ределены зоны и коэффициенты износа инструмента");
        this.toolPath = toolPath.getToolPath();
        this.surface = surface;
        this.wearingKoeffs = surface.getWearingKoeffs();
        console.replaceLastLine("Расчетный модуль создан");
    }

    /**
     *
     * @throws Exception
     */
    public void calculateInOneStep() throws Exception
    {
        console.printLine("Запуск расчета новой траектории инструмента в
1 клик...");
        console.printLine("Шаг 1 из 6:");
        findContactPointsAndDistances();
        console.printLine("Шаг 2 из 6:");
        findIdleToolPathPoints(Parameters.distanceOfIdleTransaction);
        console.printLine("Шаг 3 из 6:");
        findTravelDistances();
        console.printLine("Шаг 4 из 6:");
        defineZoneForEachPoint();
```

```
            console.printLine("Шаг 5 из 6:");
            calculateVectorOfWearingForEachPoint();
            console.printLine("Шаг 6 из 6:");
            calculateNewToolPath();
            console.printLine("Расчет новой траектории инструмента в 1 клик
успешно завершен");
        }

        /**
         */
        public void findContactPointsAndDistances()
        {
            ProgressLog pl = new ProgressLog("Ищу точки контакта инструмен-
та с поверхностью модели: ", toolPath.size());
            TreeMap<Long, Point3D> tempPointsList;
            distanceToSurfaceOfModel = new
ArrayList<Long>(this.toolPath.size());
            contactPoints = new ArrayList<Point3D>(this.toolPath.size());
            for (Point3D toolPathPoint : toolPath)
            {
                tempPointsList = new TreeMap<Long, Point3D>(); // обнуляем
                for (Point3D surfacePoint : surface.getPointsCloud())
                {
                    tempPointsList.put(toolPathPoint.distance(surfacePoint),
surfacePoint);
                }
                contactPoints.add(tempPointsList.firstEntry().getValue());
                distanceToSurfaceOfModel.add(tempPointsList.firstKey());
                pl.tick();
            }
        }

        /**
         *
         * @param max_distance
         * @throws Exception
         */
        public void findIdleToolPathPoints(long max_distance) throws Exception
        {

            if (max_distance == 0L)
                throw new Exception("Невозможно найти точки холостого
хода, пока не задано растояние холостого хода");
            if (this.distanceToSurfaceOfModel == null)
```

```
                throw new Exception("Невозможно найти точки холостого
хода, пока не известны растояния от траектории инструмента до модели");

                ProgressLog pl = new ProgressLog("Ищу точки траектории в кото-
рых инструмент совершает холостой ход: ",
this.distanceToSurfaceOfModel.size());
                this.idleToolPathPoints = new ArrayList<Integer>();
                for(int i = 0; i < this.distanceToSurfaceOfModel.size(); i++)
                {
                        if(distanceToSurfaceOfModel.get(i)>=max_distance)
                                idleToolPathPoints.add(i);
                        pl.tick();
                }
        }

        /**
         *
         * @throws Exception
         */
        public void findTravelDistances() throws Exception
        {
                if (this.contactPoints == null)
                        throw new Exception("Невозможно найти пройденный по по-
верхности модели путь, пока не известны точки контакта инструмента и моде-
ли");

                ProgressLog pl = new ProgressLog("Ищу растояния, пройденные
инструментом по поверхности: ", contactPoints.size());
                distancesTraveled = new ArrayList<Long>();
                Point3D previousPoint = contactPoints.get(0);
                for (Point3D point : contactPoints)
                {
                        distancesTraveled.add(previousPoint.distance(point));
                        previousPoint=point;
                        pl.tick();
                }
        }

        /**
         * @throws Exception
         */
        public void defineZoneForEachPoint() throws Exception
        {
                if (this.contactPoints == null)
```

```
                throw new Exception("Невозможно определить к какой зоне относится точка контакта, пока не известны точки контакта инструмента и модели");

                ProgressLog pl = new ProgressLog("Определяю зоны точек контакта с поверхностью: ", contactPoints.size());
                this.pointZone = new ArrayList<Integer>(this.toolPath.size());
                for (Point3D point : contactPoints)
                {

        pointZone.add(this.surface.getNumberOfZoneByZ(point.getZ()));
                        pl.tick();
                }
        }

        /**


         *
         * @throws Exception
         */
        public void calculateVectorOfWearingForEachPoint() throws Exception
        {
                if (this.contactPoints == null)
                        throw new Exception("Невозможно расчитать вектора износа инструмента для каждой точки траектории, пока не известны точки контакта инструмента и модели");
                if (this.distancesTraveled == null)
                        throw new Exception("Невозможно расчитать величины износа инструмента для каждой точки траектории, пока не известен путь, пройденный инструментом по поверхности модели");
                if (this.idleToolPathPoints == null)
                        throw new Exception("Невозможно расчитать вектора и величины износа инструмента для каждой точки траектории, пока не известны точки холостого хода инструмента");

                ProgressLog pl = new ProgressLog("Расчет векторов и величин износа для точек траектории: ", this.toolPath.size());
                Point2D previous2DVectorOfWearing = new Point2D(0,0);
                Point2D temp2DVector;
                Point3D toolPathPoint;
                Point3D contactPoint;
                Point3D vectorOfContact; //вектор контакта
                long wearedAmount=0L; //величина износа
```

```
            vectorsOfWearing = new ArrayList<Point2D>(contactPoints.size());
            vectorsOfContact = new ArrayList<Point3D>(contactPoints.size());
            anglesOfContact = new ArrayList<Double>(contactPoints.size());
            wearedAmounts = new ArrayList<Long>(contactPoints.size());
            for (int i = 0; i < this.toolPath.size(); i++)
            {
                pl.tick();
                if(idleToolPathPoints.contains(i) || idleToolPathPoints.contains(i-
1)) //если эта или предыдущая точка - точка холостого хода, то не меняем век-
тор износа, а величину износа ставим 0 и выходим из цикла
                {
                    vectorsOfWearing.add(new Point2D(0,0));
                    vectorsOfContact.add(new Point3D(0,0,0));
                    anglesOfContact.add(new Double("0"));
                    wearedAmounts.add(0L);
                    continue;
                }
                toolPathPoint = toolPath.get(i);
                contactPoint = contactPoints.get(i);
                wearedAmount =
Math.round(this.wearingKoeffs.get(this.pointZone.get(i))*distancesTraveled.get(i)*
1000); //находим величину износа в нанометрах
                wearedAmounts.add(wearedAmount);
                vectorOfContact = contactPoint.subtractVector(toolPathPoint);
//получаем вектор контакта, вычитая координаты инструмента из координат
точки контакта
                vectorsOfContact.add(vectorOfContact);

        temp2DVector=convert3Dto2D(vectorOfContact).resizeVector(wearedAmou
nt); //превращаем 3d вектор контакта в 2d, делаем его длину равную величине
износа
                anglesOfContact.add(temp2DVector.getAngleToAxelZ());
                previous2DVectorOfWearing = previ-
ous2DVectorOfWearing.addVector(temp2DVector); //прибавляем к вектору из-
носа прошлой точки
                vectorsOfWearing.add(previous2DVectorOfWearing);
            }
        }

        /**
         *
         * @throws Exception
         */
        public void calculateNewToolPath() throws Exception
```

```
	{
		if (vectorsOfContact == null)
			throw new Exception("Невозможно расчитать новую траекторию инструмента, т.к. еще не вычислены вектора контакта");
		if (vectorsOfWearing == null)
			throw new Exception("Невозможно расчитать новую траекторию инструмента, т.к. еще не вычислены вектора износа");

		ProgressLog pl = new ProgressLog("Расчет новой траектории: ", this.toolPath.size());
		modifiedToolPath = new ArrayList<Point3D>(this.toolPath.size());
		Point3D toolPathPoint; //точка траектории
		Point3D vectorOfContact; //вектор контакта
		Point2D vectorOfWearing; //вектор износа
		long x,y,projectionLength;
		for (int i = 0; i < this.toolPath.size(); i++)
		{
			toolPathPoint = toolPath.get(i);
			vectorOfContact = vectorsOfContact.get(i);
			vectorOfWearing = vectorsOfWearing.get(i);

			projectionLength = new Point2D(vectorOfContact.getX(),vectorOfContact.getY()).length(); //ищем длину проекции вектора контакта на плоскость XY

	x=Math.round((double)vectorOfWearing.getXY()*vectorOfContact.getX()/projectionLength/1000); //ищем координату x поверутого к проекции вектора

	y=Math.round((double)vectorOfWearing.getXY()*vectorOfContact.getY()/projectionLength/1000); //ищем координату y поверутого к проекции вектора
			modifiedToolPath.add(toolPathPoint.addVector(new Point3D(x, y, vectorOfWearing.getZ()/1000))); //прибавляем к координате траектории повернутый вектор, тем самым получив смещенную точку траектории
			pl.tick();
		}
	}

	/**
	 * @param vector
	 * @return
	 */
	private Point2D convert3Dto2D(Point3D vector3D)
	{
```

```
            return new Point2D(new Point2D(vector3D.getX(), vec-
tor3D.getY()).length(), vector3D.getZ());
        }

        /**
         * @throws Exception
         */
        public ToolPath getModifiedToolPath() throws Exception
        {
            if(this.modifiedToolPath==null)
                throw new Exception("Траектория еще не расчитана");
            return new ToolPath(this.modifiedToolPath);
        }

        /**
         * @return
         * @throws Exception
         */
        public DataTable getDataTableForExcel() throws Exception
        {
            if(this.modifiedToolPath==null)
                throw new Exception("Траектория еще не расчитана");

            String[] headers = new String[]{
                "old_x"
                ,"old_y"
                ,"old_z"
                ,"old_dist"
                ,"new_x"
                ,"new_y"
                ,"new_z"
                ,"new_dist"
                ,"surf_x"
                ,"surf_y"
                ,"surf_z"
                ,"Зона"
                ,"Пройденый участок, мкм"
                ,"Износ на участке, нм"
                ,"Угол контакта фрезы"
                ,"Холостой ход"
            };
            Class<?>[] columns = new Class<?>[]{
                Long.class
                ,Long.class
```

```
                ,Long.class
                ,Long.class
                ,Long.class
                ,Long.class
                ,Long.class
                ,Long.class
                ,Long.class
                ,Long.class
                ,Long.class
                ,Integer.class
                ,Long.class
                ,Long.class
                ,Double.class
                ,String.class
        };
        DataTable dt = new DataTable(Arrays.asList(headers), Ar-
rays.asList(columns));
        List<List<Object>> table = new ArrayList<List<Object>>();
        ProgressLog pl = new ProgressLog("Подготовка данных для выгруз-
ки: ", this.toolPath.size());

        List<Object> tempRow;
        for (int i = 0; i < this.toolPath.size(); i++)
        {
                tempRow = new ArrayList<Object>();

                tempRow.add(this.toolPath.get(i).getX());
                tempRow.add(this.toolPath.get(i).getY());
                tempRow.add(this.toolPath.get(i).getZ());
                tempRow.add(distanceToSurfaceOfModel.get(i));
                tempRow.add(this.modifiedToolPath.get(i).getX());
                tempRow.add(this.modifiedToolPath.get(i).getY());
                tempRow.add(this.modifiedToolPath.get(i).getZ());

    tempRow.add(this.modifiedToolPath.get(i).distance(contactPoints.get(i)));
                tempRow.add(this.contactPoints.get(i).getX());
                tempRow.add(this.contactPoints.get(i).getY());
                tempRow.add(this.contactPoints.get(i).getZ());
                tempRow.add(this.pointZone.get(i));
                tempRow.add(this.distancesTraveled.get(i));
                tempRow.add(this.wearedAmounts.get(i));
                tempRow.add(this.anglesOfContact.get(i));
                tempRow.add(this.idleToolPathPoints.contains(i)?"Да":"Нет");
```

```
                table.add(tempRow);
                pl.tick();
            }
            dt.setTable(table);
            return dt;
        }
}

package ru.pii.toolWearing.utils;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import ru.pii.toolWearing.excel.DataTable;
import ru.pii.toolWearing.excel.ExcelReader;
import ru.pii.toolWearing.excel.ExcelWriter;
import ru.pii.toolWearing.loger.Console;
import ru.pii.toolWearing.loger.ProgressLog;
import ru.pii.toolWearing.models.Point3D;
import ru.pii.toolWearing.models.Surface;
import ru.pii.toolWearing.models.ToolPath;
import ru.pii.toolWearing.models.ToolPathCalculation;

public class ExportImport {

        public static final String MILLIMETER = "millimeter";
        public static final String MICROMETER = "micrometer";

        /**
         * @param File.
         * @return
         * @throws Exception
         */
        public static ToolPath importToolPathFromFile(File file) throws Exception
        {
                PathParser parser = new PathParser(file);
```

```
            return new ToolPath(parser.runParsing());
        }

        /**
         * @param pathToExcelFile
         * @param unitOfMeasure
         * @return
         * @throws Exception
         */
        public static Surface importSurfaceFromExcel(String pathToExcelFile, String
unitOfMeasure) throws Exception
        {
            List<List<Long>> table= getThreeExcelColumns(pathToExcelFile,
unitOfMeasure);
            Set<Point3D> surfaceSet = new HashSet<Point3D>();
            ProgressLog pl = new ProgressLog("Создание поверхности модели
по полученным координатам: ", table.size());
            for(List<Long> row : table)
            {
                surfaceSet.add(new Point3D(row.get(0),row.get(1),row.get(2)));
                pl.tick();
            }
            return new Surface(surfaceSet);
        }

        /**

         * @param pathToExcelFile
         * @param unitOfMeasure
         * @return.
         * @throws Exception
         */
        public static void exportCalculatedData(String pathToExcelFile,
ToolPathCalculation calc) throws Exception
        {
            if(calc==null)
                throw new Exception("Траектория еще не расчитана");
            DataTable dataTable = calc.getDataTableForExcel();
            ExcelWriter ew = new ExcelWriter(pathToExcelFile);
            ew.createSheet("Данные расчета", dataTable);
            ew.writeToFile();
            Console.printLine("Расчет выгружен: "+ pathToExcelFile);
        }
```

```
	public static void exportToolPath(String pathToTapFile, ToolPath toolPath)
throws Exception
	{
		if(toolPath==null)
			throw new Exception("Траектории нет");
		if(pathToTapFile==null)
			throw new Exception("Путь не передан");

		File file = new File(pathToTapFile);
		if (file.isDirectory())
			throw new FileNotFoundException("Путь указывает на папку,
а должен на файл");
		File dir = new File(file.getParent());
		if (!dir.exists())
			dir.mkdirs(); //create dirictories if not exists
		FileOutputStream fos = new FileOutputStream(file);
		BufferedWriter bw = new BufferedWriter(new
OutputStreamWriter(fos));
		ProgressLog pl = new ProgressLog("Выгрузка траектории: ",
toolPath.getToolPath().size());
		for(Point3D point: toolPath.getToolPath())
		{
			bw.write(point.toStringMillimeters());
			bw.newLine();
			pl.tick();
		}
		bw.close();
		Console.printLine("Траектория выгружена: "+ pathToTapFile);
	}

	/**
	 * @param pathToExcelFile
	 * @param unitOfMeasureInExcelFile
	 * @return
	 * @throws Exception
	 */
	private static List<List<Long>> getThreeExcelColumns(String
pathToExcelFile, String unitOfMeasureInExcelFile) throws Exception
	{
		boolean micrometer=unitOfMeasureInExcelFile.equals(ExportImport.MICROMETER);
		ExcelReader er = new ExcelReader(pathToExcelFile);
		Console.printLine("Пытаюсь прочесть файл: " + pathToExcelFile);
		List<List<String>> stringTable = er.readSheet(0, new int[] {0,1,2});
```

```
            Console.replaceLastLine("Файл успешно прочтен: " +
pathToExcelFile);
            er.clearWorkbook();
            List<List<Long>> integerTable = new ArrayList<List<Long>>();
            List<Long> integerRow;
            ProgressLog pl = new ProgressLog("Преобразование значений коор-
динат в микрометры и округление: ", stringTable.size());
            for(List<String> stringRow : stringTable)
            {
                  integerRow = new ArrayList<Long>();
                  for(String stringCell: stringRow)
                  {
                        if(micrometer)
                        {

      integerRow.add(Math.round(Double.parseDouble(stringCell.replace(",",
".")))));
                        }else
                        {

      integerRow.add(Math.round(Double.parseDouble(stringCell.replace(",",
"."))*1000));
                        }
                  }
                  integerTable.add(integerRow);
                  pl.tick();
            }
            return integerTable;
      }


}

package ru.pii.toolWearing.utils;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.file.Files;
import java.util.ArrayList;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

```
import ru.pii.toolWearing.loger.ProgressLog;
import ru.pii.toolWearing.models.Point3D;

public class PathParser
{

	private List<String> gcodeText;

	/**
	 *
	 * @param gcodeText
	 * @throws FileNotFoundException
	 */
	public PathParser(List<String> gcodeText) throws FileNotFoundException
	{
		this.gcodeText = gcodeText;
	}

	/**
	 * @param filePath
	 * @throws IOException
	 */
	public PathParser(File file) throws IOException
	{
		if (!file.exists() || file.isDirectory())
			throw new FileNotFoundException("Такого файла не сущест-
вует либо это папка");
		gcodeText = Files.readAllLines(file.toPath());
	}

	public List<Point3D> runParsing()
	{
		List<Point3D> toolPath = new ArrayList<Point3D>();
		long X=0, Y=0, Z=0;
		Pattern pat = Pattern.compile("[xyzXYZ][-+]?[0-9]*\\.?[0-9]*");
		ProgressLog pl = new ProgressLog("Парсинг G-кода: ",
this.gcodeText.size());
		for (String line : this.gcodeText)
		{
			Matcher m = pat.matcher(line);
			String matchedText = "";
			while (m.find())
			{
				matchedText = m.group();
```

```
        if(matchedText.startsWith("X")||matchedText.startsWith("x"))
                                {
                                        X =
parseDoubleToMicrometers(matchedText.substring(1));
                                }
                                else
if(matchedText.startsWith("Y")||matchedText.startsWith("y"))
                                {
                                        Y =
parseDoubleToMicrometers(matchedText.substring(1));
                                }
                                else
if(matchedText.startsWith("Z")||matchedText.startsWith("z"))
                                {
                                        Z =
parseDoubleToMicrometers(matchedText.substring(1));
                                }
                        }
                        toolPath.add(new Point3D(X,Y,Z));
                        pl.tick();
                }
                return toolPath;
        }

        private long parseDoubleToMicrometers(String doubleVal)
        {
                return Math.round(Double.parseDouble(doubleVal)*1000);
        }

}

package ru.pii.toolWearing;

import java.io.File;
import java.nio.file.FileSystems;

import javax.swing.UIManager;

import ru.pii.toolWearing.gui.MainFrame;
import ru.pii.toolWearing.models.Surface;
import ru.pii.toolWearing.models.ToolPath;
import ru.pii.toolWearing.models.ToolPathCalculation;
```

```
public class Main
{

	public static ToolPath toolPath = null;
	public static Surface surface = null;
	public static ToolPathCalculation calc = null;
	public static File toolPathDir = new File(System.getProperty("user.home") +
FileSystems.getDefault().getSeparator() + "Desktop");

	public static void main(String[] args) throws Exception
	{
		try
		{

	UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName(
));
		} catch (Exception e)
		{
		}
		new MainFrame();
	}
}
```