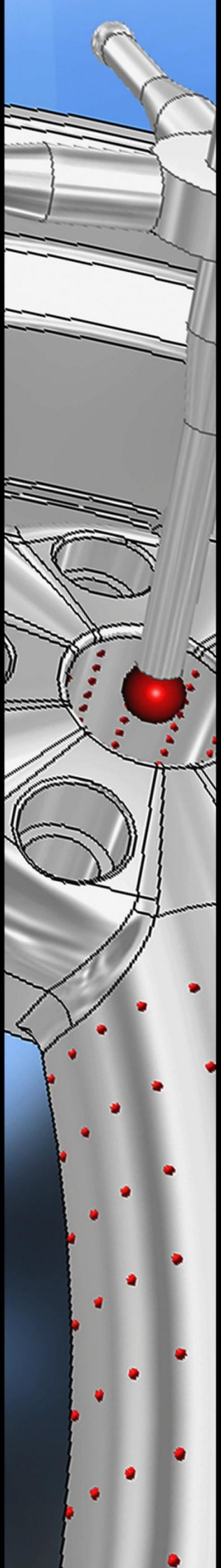


ТЕХНО™
КООРД
ТЕХНОЛОГИЯ КООРДИНАТНЫХ ИЗМЕРЕНИЙ

Оглавление

Введение

На сегодняшний день автоматизация процесса измерения является важной производственной задачей для повышения качества производства. Закрытое акционерное общество Челяб*НИИ*контроль занимается не только разработкой, конструированием и выпуском измерительных приборов, но и программного обеспечения к ним.

Программное обеспечение **ТЕХНО**коорд™ разработано для облегчения труда метролога. Мы обеспечиваем заказчика не только надежным программным продуктом для настройки и программирования контрольно-измерительных машин, но и разрабатываем эргономичный, интуитивно-понятный для пользователя интерфейс, а также предоставляем средства для максимальной автоматизации статистической обработки результатов измерения и формирования наглядных отчетов. Благодаря высокой интеллектуальности нашего программного обеспечения, от пользователя, помимо знания основ координатных измерений, требуются только базовые навыки работы с операционной системой Microsoft Windows™.

ТЕХНОкоорд™ – это не просто программное обеспечение для точных координатных измерений, соответствующих стандартам, на контактных и оптических измерительных машинах различного типа, оно позволяет превратить обычную компьютерную систему в мощный инструмент для автоматизации измерений, оснащенный виртуальной измерительной машиной, позволяющей проводить измерения без использования реальной машины, что экономит рабочее время и ресурсы, повышает безопасность измерения.

В данном руководстве подробно и максимально просто описаны принципы и средства работы в различных модулях программного обеспечения **ТЕХНО**коорд™, включающих три контактные схемы измерения и две оптические:

- **Стандартная контактная схема измерения** позволяет:
  - экспортировать CAD-модель из любого существующего CAD-редактора;
  - автоматически распознавать тип поверхности и составлять стратегии измерения автоматически и вручную;
  - автоматически составлять маршрут таким образом, чтобы время измерения было минимальным, щуп не столкнулся с деталью и крепежами, причем система автоматически подберет необходимый щуп в случае использования щупа с несколькими наконечниками;
  - производить привязку системы координат машины к системе координат CAD-модели детали;
  - визуализировать движения щупа относительно детали;
  - осуществлять измерения различными типами щуповых головок.

- **Расширенная контактная схема измерения** позволяет:
  - получать координаты точек, измеренных вручную;
  - программировать перемещения и измерения на КИМ;
  - создавать собственные системы координат и использовать их для управления машиной;
  - производить произвольные расчеты, используя встроенный язык;
  - прикреплять мультимедийную информацию для оператора;
  - выводить измеренные параметры в отчет;

- **Реверсивная контактная схема измерения** позволяет:
  - измерять отдельные элементы детали без предварительно подготовленной модели;

  - проводить измерение в автоматическом цикле после первого ручного измерения;
  - производить привязку системы координат машины к системе координат модели;
  - визуализировать движения щупа относительно детали;
  - осуществлять измерения различными типами щуповых головок.

- **Стандартная оптическая схема измерения** позволяет:
  - выполнять измерения в автоматическом цикле на оптической координатной машине, оснащенной двигателем;
  - производить привязку системы координат машины к системе координат CAD-модели;
  - проводить измерения на основе построенной CAD-модели детали;
  - проводить измерения как с применением реальной оптической измерительной системы, так и в режиме симуляции;
  - визуализировать движения камеры относительно детали.

- **Реверсивная оптическая схема измерения** позволяет:
  - проводить измерения без предварительного построения CAD-модели;
  - измерять точки вручную: путем указания точек для измерения, и автоматически: путем указания контура для измерения.
  - проводить измерения как с применением реальной оптической измерительной системы, так и в режиме симуляции;
  - передавать полученную схему для измерения в стандартную оптическую схему измерения.

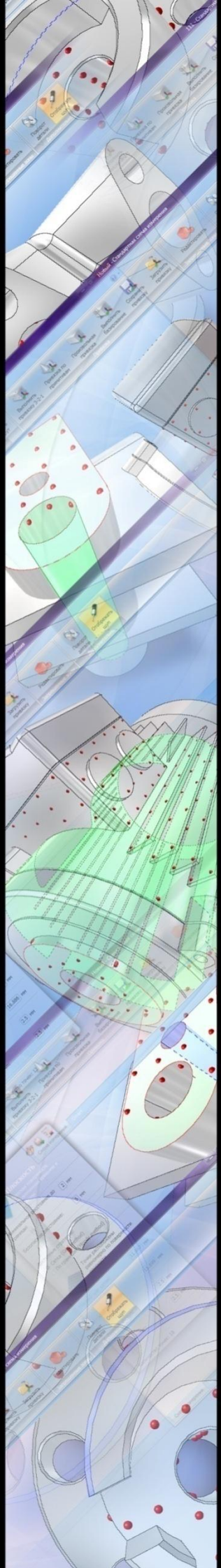
Каждая схема измерения позволяет передавать полученные данные в мощный инструмент координатного анализа геометрии детали, позволяющий производить сложные расчеты одним кликом мыши и создавать отчеты, наглядно отображающие результаты. Универсальный блок компенсаций позволяет убирать любые систематические погрешности любой координатно-измерительной машины.

Настройка контактной и оптической измерительных машин подробно описана в дополнительном руководстве «Настройка измерительной машины».

Символы и обозначения в документе

Этот символ указывает на дополнительную полезную информацию.

Этим символом обозначаются ситуации, которые могут привести к ложным или неточным измерениям, а также другим затруднениям в работе с программным продуктом.



ТЕХНО КООРД™

ТЕХНОЛОГИЯ КООРДИНАТНЫХ ИЗМЕРЕНИЙ

Координатные схемы измерения

Оглавление

Редактор щуповой системы координатно-измерительной машины

Обзор щуповой системы

Мастер настройки щуповой системы представлен на рисунке ниже (см. **Рисунок 1**).

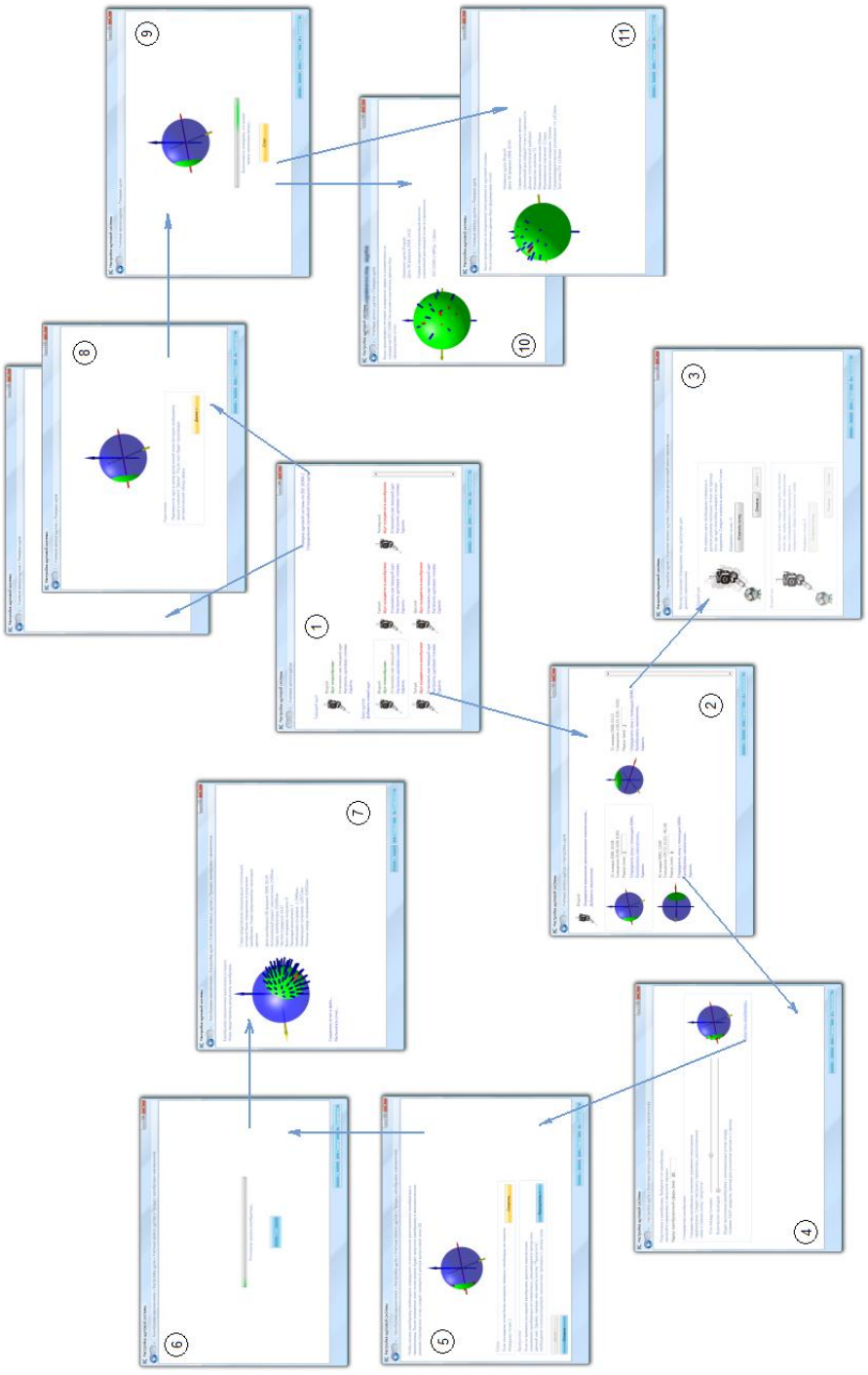


Рисунок 1. Мастер настройки щуповой системы: 1 – список щупов, 2 – список наконечников щупа, 3- определение доступной зоны с использованием КИМ, 4 – установка параметров калибровки наконечника, 5 – получение первых точек в ручном режиме, 6 – автоматический процесс измерения сферы, 7 – результаты калибровки (отчет о калибровке), 8 – запуск определения повторяемости щупа или тест по ISO 10360-2, 9 – автоматический процесс обхода сферы, 10 –отчет по ISO 10360-2, 11 – отчет о повторяемости щупа

Возможности щуповой системы

Данная система поддерживает следующие типы щуповых головок:

- стандартный вертикальный щуп;
- головки с несколькими наконечниками, в т.ч. щупы типа «Звезда»;
- щупы повернутые на определенный угол;
- поворотные головки (articulated probe).

Система не поддерживает дисковые наконечники, наконечники цилиндрической или другой отличной от сферы формы.

Основные возможности системы:

- База данных щупов позволяет создавать и хранить различные конфигурации щуповых головок.
- Гибкая настройка допустимой зоны наконечника позволяет использовать даже нестандартные щуповые системы.
- Калибровка позволяет убрать систематическую составляющую погрешности щуповой системы.
- Специальный мастер поверки поможет определить:
  - случайную составляющую погрешности щуповой головки;
  - погрешность щупа по стандарту ISO 10360-2;
  - погрешность щуповой системы с несколькими наконечниками по стандарту ISO 10360-5;
- Визуализация отклонений в отчетах о калибровке и поверке.

Рекомендуемый порядок работ

Каждая щуповая головка используемая на КИМ должна быть внесена в базу данных щуповых систем (см. «Создание щупа», на стр. 10), также необходимо произвести настройку и калибровку наконечников (см. «Калибровка щупа», на стр. 15).

При физической замене щупа на КИМ следует выбрать новую текущую щуповую систему из имеющихся. Если калибровка щуповой системы производилась давно или она была смещена, то рекомендуется повторить калибровку.

Периодически следует запускать поверку щуповой системы (см. «Определение погрешностей», на стр. 20) чтобы определить, соответствует ли точность паспортной. Периодичность поверки и точность щуповой системы должна быть указана в документации на КИМ.

Модель щуповой системы

Ниже описывается модель данной системы. Модель определяет терминологию и некоторые упрощения.

Основным объектом щуповой системы является *наконечник*. В данной модели *наконечник* определяется как сферическая поверхность с заданными ограничениями (допустимой зоной).

Допустимая зона наконечника задается путем указания части калибровочной сферы, с которой наконечник имеет возможность свободно соприкоснуться. Зона задается плоскостью, которая разделяет калибровочную сферу на недоступную и доступную зоны (см. **Рисунок 2**).

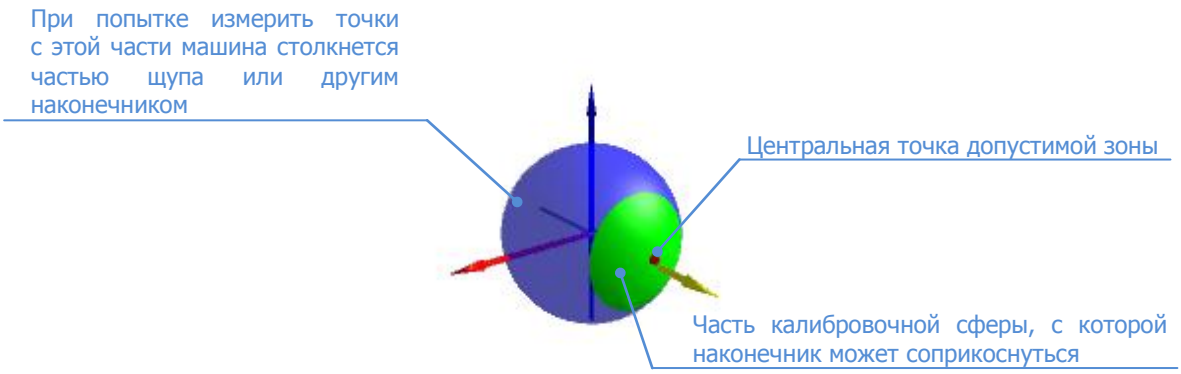
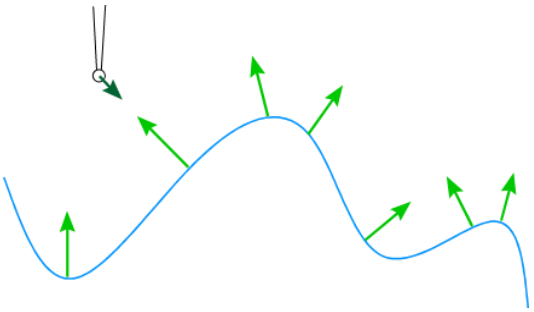


Рисунок 2. Допустимая зона наконечника



Форма калибровочной сферы считается идеальной, а ее радиус известным. Радиус наконечника считается известным, а погрешность формы задается функцией, аргументом которой является значение нормального вектора к поверхности в точке касания (см. **Рисунок 3**).

Рисунок 3.
Движение по нормали к поверхности

Щуповая система – это набор щупов, расположенных в пространстве (для корректной работы программы требуется визуально сконструировать геометрию щуповой системы: это можно сделать в отдельном редакторе или в данном мастере).

Расположение щупов определяется относительно: каждому щупу соответствует вектор, определяющий его смещение от первого наконечника (первому наконечнику соответствует нулевой вектор). Точные значения векторов определяются при калибровке наконечников и учитываются при измерении (см. **Рисунок 4**).

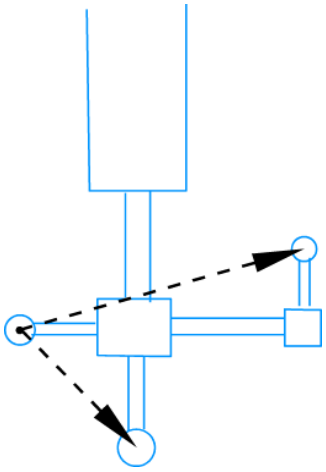


Рисунок 4. Вектора смещения от первого наконечника к остальным

Создание щуповой системы

Создание щуповой системы

На главной странице мастера настройки щуповой системы расположен список всех доступных щуповых систем, которые уже были созданы. Щуповая системы, учетная запись которой находится в верхней части окна, является текущей используемым системой (см. **Рисунок 5**).

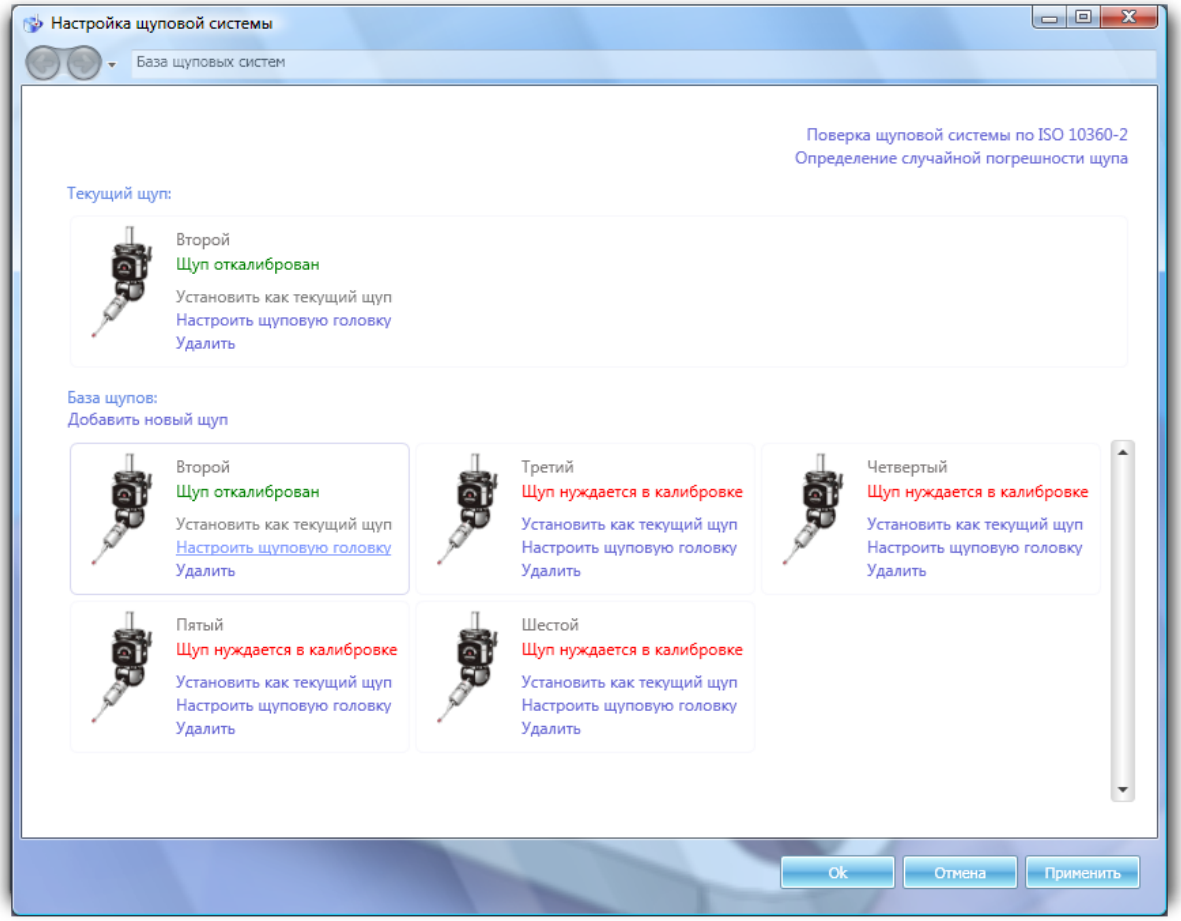


Рисунок 5. Окно со списком щупов

Чтобы создать новую щуповую конфигурацию следует нажать на кнопку «Добавить новую щуповую систему». В список будет добавлена учетная запись.

Учетная запись щуповой системы

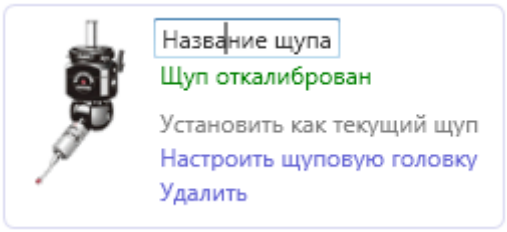


Рисунок 6. Редактирование названия щупа

После создания учетной записи щуповой системы рекомендуется сразу же переименовать её. Для этого следует один раз щелкнуть мышью по названию, после чего появится возможность редактирования (см. **Рисунок 6**).

Чтобы удалить щуп следует нажать на кнопку «Удалить», учетная запись исчезнет из списка.

Все изменения, которые происходят, не сохраняются автоматически. Для применения изменений необходимо нажать на кнопку «Применить», которая находится в нижней части окна. Если же изменения были ошибочны, следует нажать «Отмена», в этом случае произведенные изменения будут утеряны.

Программа выделяет одну щуповую систему, которая используется для измерений. Чтобы установить щуповую систему в качестве текущей существует кнопка «Установить как текущую». Если кнопка не активна, значит, система уже является текущей.

Рисунок 7. Учетная запись щупа

Надпись «Система откалибрована» означает, что наконечники щупа настроены и откалиброваны, а надпись «Система не откалибрована» означает, что не все наконечники щупа откалиброваны или вовсе отсутствуют. В последнем случае требуется зайти на страницу настройки щуповой головки: для этого следует нажать кнопку «Настроить щуповую головку», и выполнить настройку.

Геометрия щуповой системы

Геометрия щуповой системы – это примитивная CAD-модель щуповой системы, приблизительно описывающая расположение тех или иных частей щупа. Геометрия щуповой системы используется для:

- автоматического поиска пути;
- корректного отображения щупа в схеме измерения;
- симуляции измерения на виртуальном КИМ.

Геометрию щуповой системы можно сохранять и загружать из файла. Для этого следует использовать кнопки в верхней части окна: «Импортировать геометрию» и «Экспортировать геометрию». Геометрия щуповой системы сохраняется в файл с расширением *.probegeometry.

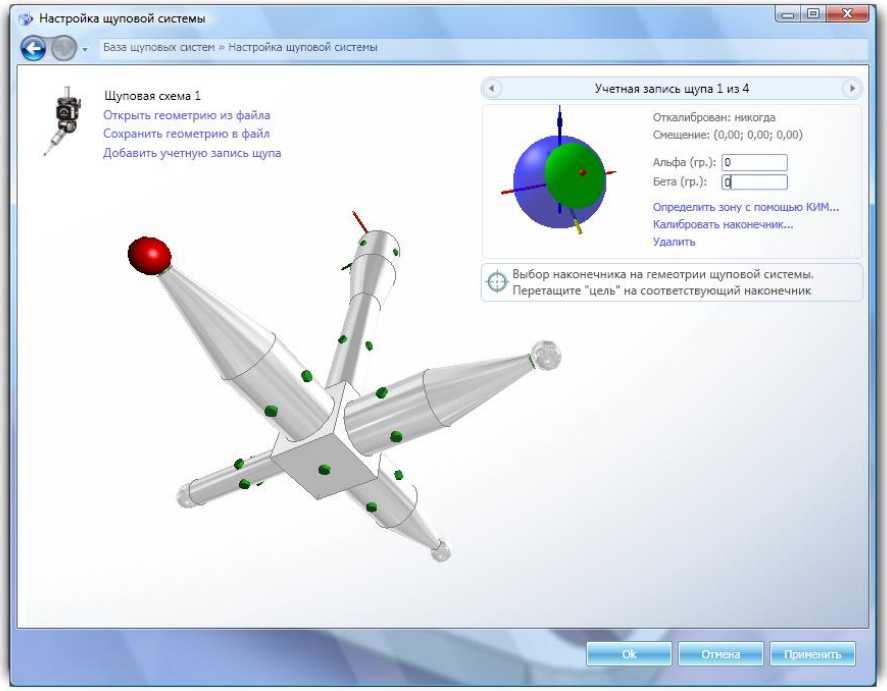


Рисунок 8. Страница настройки щуповой системы

Редактировать геометрию можно непосредственно в мастере настройки щуповой системы, а можно воспользоваться отдельным редактором геометрии щуповой системы, который может открывать файлы с расширением *.probegeometry.

После создания новой учетной записи, геометрия будет отсутствовать, поэтому экран окажется пустой. Первым элементом, который следует создать, является элемент, примыкающий к щуповой головке (как правило, это удлинитель цилиндрической формы). Все действия, которые можно производить, находятся в контекстном меню, которое вызывается однократным нажатием правой клавиши мыши (см. **Рисунок 9**).

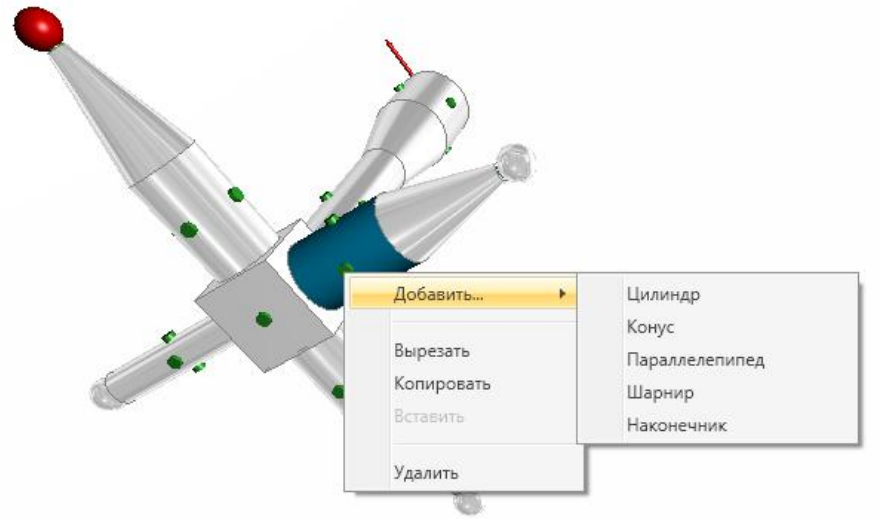


Рисунок 9. Контекстное меню редактора геометрии щуповой системы

Например, чтобы добавить первый элемент - цилиндр, следует вызвать контекстное меню, щёлкнув в любом месте и выбрать «Добавить… → Цилиндр». После того, как один элемент есть, последующие можно «крепить» только к специальным местам, которые обозначены маркерами зеленого цвета.

Параметры созданных элементов можно изменить. Для этого нужно выделить соответствующий элемент левой клавишей мыши, после чего в нижней части появятся параметры элемента, которые можно редактировать.

Для того чтобы ориентироваться в направлениях осей машины относительно редактируемой геометрии в месте крепления первого элемента расположена система координат (оси следует различать по цветам: красная(X), зеленая(Y), синяя(Z)).

Чтобы перетащить выделенный элемент, а также элементы, которые к нему прикреплены, на другое место крепления, нужно нажать левую клавишу мыши и, не отпуская, дотянуть до нового места крепления.

Редактор позволяет копировать / вырезать элемент геометрии (вместе с элементами, которые прикреплены к нему) и вставлять в новые места крепления. Это может быть полезно, если щуповая система имеет одинаковые части. Можно также открыть с помощью отдельного редактора готовый щуп и скопировать из него часть. Копирование и вставка осуществляется через контекстное меню.

Управление просмотром с помощью мыши осуществляется следующим образом:
Вращение вокруг центральной точки – правая клавиша мыши.

Смещение центральной точки – средняя клавиша мыши.

Учетные записи наконечников

По нажатию на кнопку «Добавить учетную запись наконечника» появляется новая учетная запись наконечника с параметрами по умолчанию. Чтобы удалить учетную запись существует кнопка «Удалить» у каждого наконечника (см. **Рисунок 10**).

Рисунок 10. Учетные записи наконечников

Один из параметров наконечника – это допустимая зона, т.е. область к которой наконечник способен подойти/измерить, не задев при этом измеряемый объект. Зона задается визуально.

Другой параметр – это диаметр наконечника. Диаметр наконечника задается непосредственно в геометрии щуповой системы у соответствующего элемента. Каждая учетная запись обязательно должна быть привязана к одному из наконечников геометрии. Для задания соответствия следует «перетащить» специальный элемент «цель» на соответствующий шарик, после чего шарик подсветится красным цветом.

Кроме того, каждая учетная запись наконечника имеет два угла - это углы поворотной головки в тот момент, когда наконечник калибруется. Если головка не поворотная, то значения углов нужно установить равными нулю.

Настройка зоны вручную

На рисунке ниже представлен редактор допустимой для наконечника зоны (см. **Рисунок 11**). Редактор содержит виртуальную сферу, на которой зеленым цветом отмечена зона, к которой наконечник может подойти без помех.

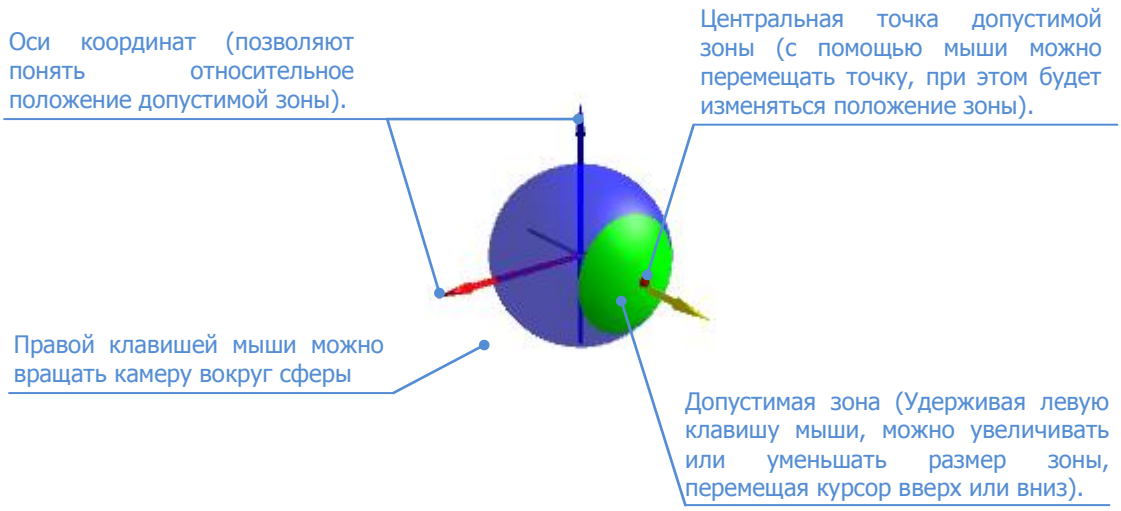


Рисунок 11. Элемент, редактор допустимой зоны наконечника

Можно сказать, что доступная зона задается точкой на сфере и частью сферы, которая доступна наконечнику. Поэтому редактирование зоны заключается в том, чтобы переместить «точку» в позицию, где наконечник располагался бы по нормали к поверхности сферы, затем отрегулировать размер зоны.

Устанавливать точно допустимую зону не требуется, однако, если зона установлена некорректно, при калибровке может произойти ошибка (непредвиденное касание).

Если калибровка была выполнена без ошибок, то в процессе измерения ошибки из-за некорректной допустимой зоны наконечника произойти не может.

Получение зоны с помощью КИМ

Более надежным, но более продолжительным способом является определение допустимой зоны наконечника, используя сам наконечник и КИМ.

В мастере предлагается измерить точки на границе зоны: там, где наконечник еще способен измерять, не задевая поверхность. Затем, измерить пару точек так чтобы направление наконечника было сонаправлено с нормалью поверхности сферы (см. **Рисунок 12**).

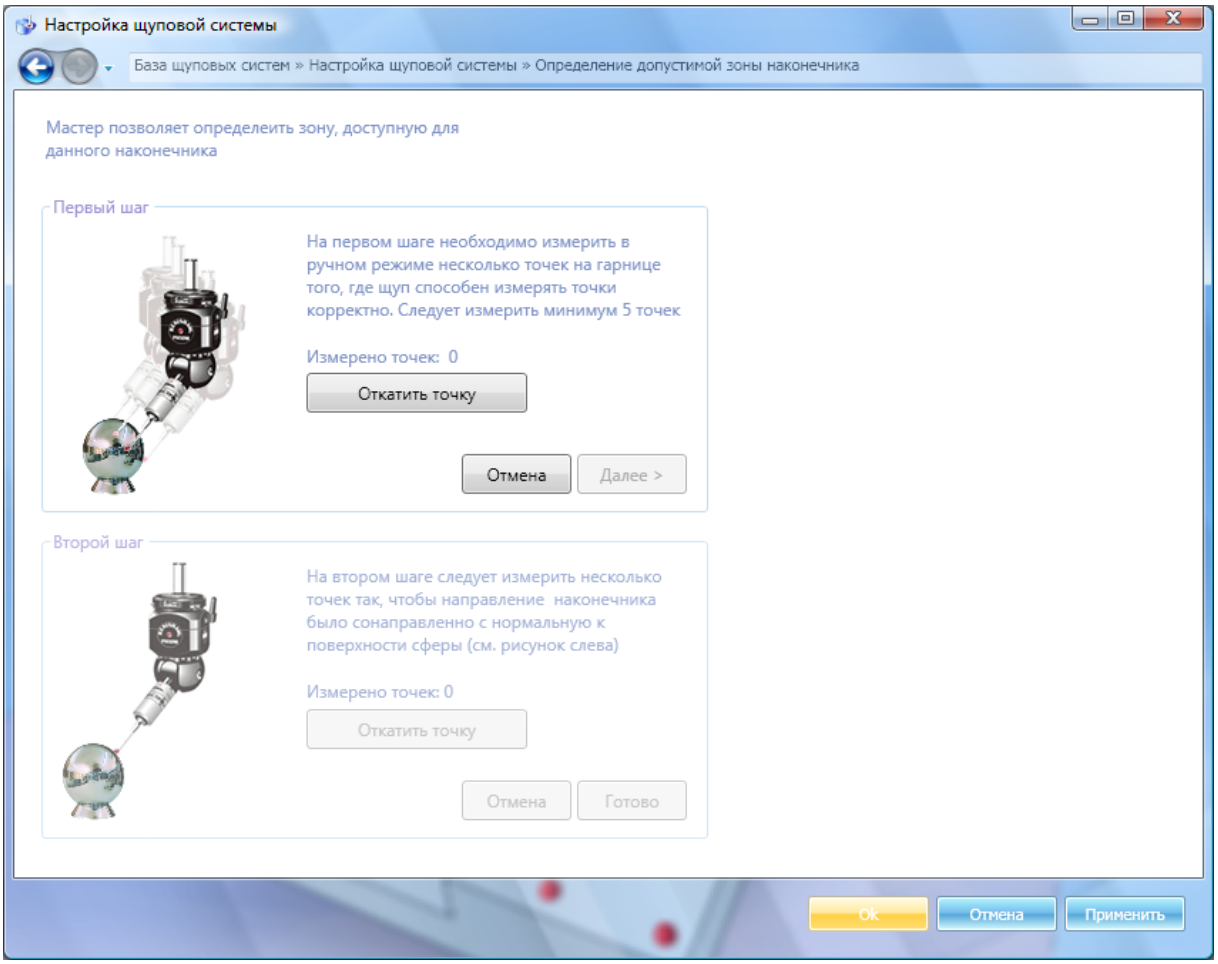


Рисунок 12. Получение допустимой зоны наконечника с помощью КИМ

Калибровка щупа

Обзор

Назначение калибровки

Калибровка щупа выполняется с целью определить отклонения датчика, которые возникают при измерении под разными углами к поверхности. В качестве калибратора выбирается сфера с аттестованным радиусом и отклонением формы не более 0.5 мкм (в общем случае зависит от точностных возможностей машины). В процессе измерения деталей полученные отклонения используются в качестве компенсации.

Другой целью калибровки наконечника ставится определить его относительное расположение – относительно других наконечников щупа. Когда щупов несколько, каждый щуп хранит в себе вектор, который указывает на центр первого наконечника. Это позволяет приводить измерения разными наконечниками к одной системе координат.

Этапы калибровки

Первым шагом калибровки является получение пяти точек в ручном режиме. По пяти точкам производится аппроксимация сферы. Теперь известно положение калибратора относительно наконечника.

Следующей задачей калибровки является обход сферы, касаясь под различными углами, но каждый раз по нормали. Следует обратить внимание, что качество калибровки определяется частотой расположения точек на сфере. Однако с увеличением их количества увеличивается продолжительность калибровки.

Естественным является так же то, что полученный в результате ручного измерения центр является неточным, а значит, и движение по нормали будет производиться с погрешностью. С целью уточнить центр измеряется минимально возможное число точек и центр калибратора перерасчитывается.

После уточнения центра происходит основной проход. Для получения дополнительной точности можно выполнить несколько проходов, каждый раз все больше приближаясь к действительному центру калибратора.

Запуск калибровки

Перед запуском калибровки следует ввести диаметр калибровочной сферы, настроить угол, через который будут расставлены точки, и количество проходов (см. **Рисунок 13**).

Выбранные параметры, сохраняются, поэтому повторно вводить параметры не придется.

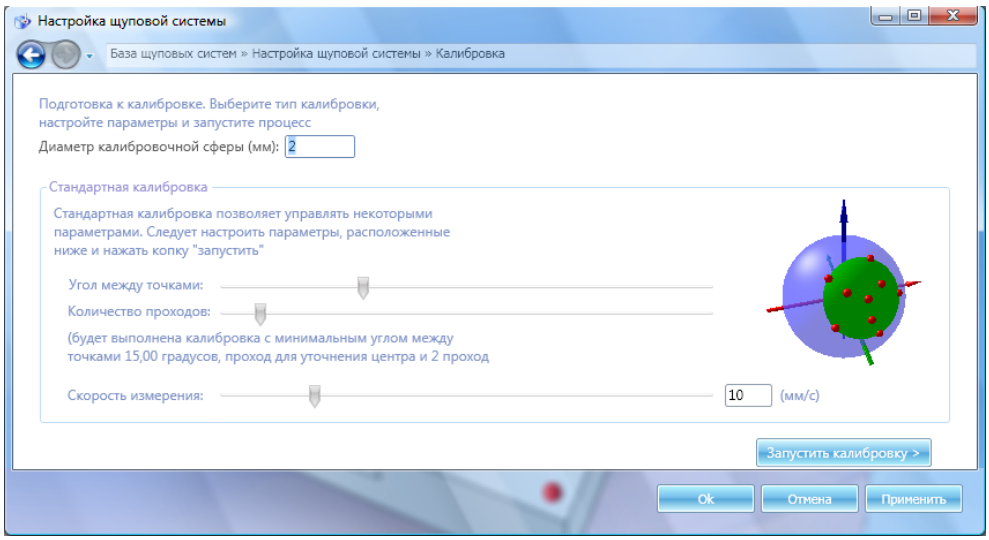


Рисунок 13. Параметры калибровки наконечника

Чтобы запустить процесс калибровки с выбранными параметрами следует нажать кнопку «Запустить калибровку».

Получение центра в ручном режиме

Для того чтобы обойти в автоматическом режиме точки на сфере предварительно необходимо узнать положение сферы относительно щупа. Для этого пользователю предлагается измерить в ручном режиме пять равномерно распределенных по поверхности сферы точек. На картинке в центре экрана изображено рекомендуемое расположение точек (см. **Рисунок 14**).

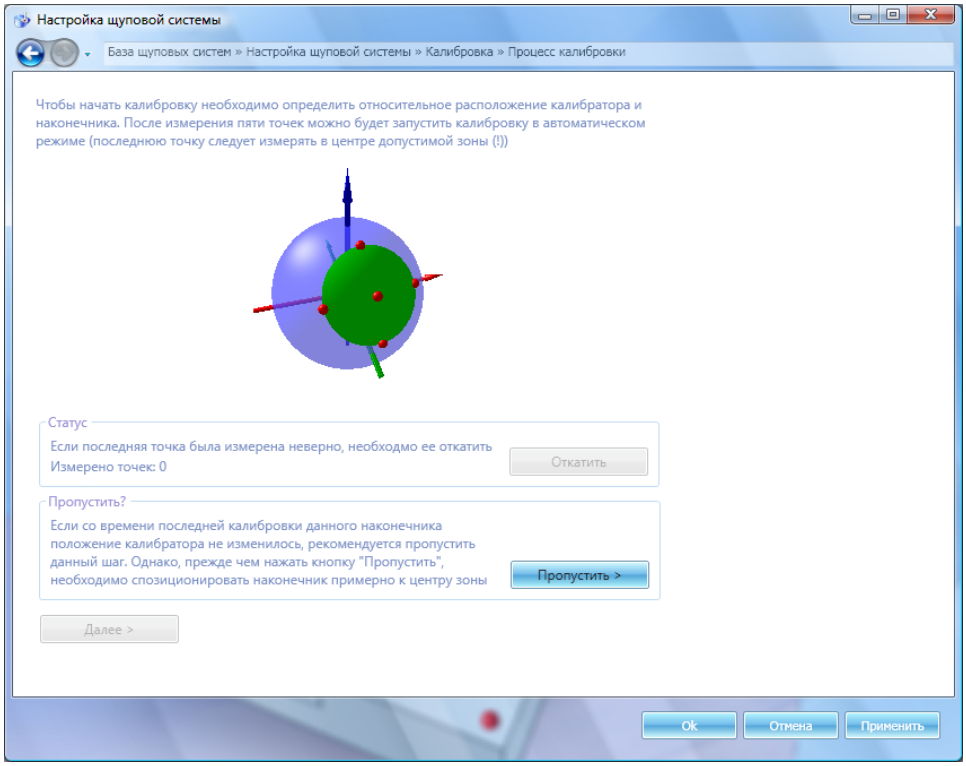


Рисунок 14. Получение первых пяти точек

Необходимо заметить, что получение центра сферы в ручном режиме следует делать в следующих случаях:

- калибровочная сфера была перемещена;
- калибровочная сфера была заменена;
- выполняется первая калибровка для данного наконечника.

Чтобы пропустить данный шаг можно воспользоваться кнопкой «Пропустить». Мастер в этом случае использует вычисленный ранее центр калибровочной сферы и начнет автоматическое измерение. Однако во избежание столкновения со сферой наконечник необходимо спозиционировать примерно к центру допустимой зоны.

Калибровочная сфера должна оставаться неподвижной. Запрещается перемещать сферу. Если сфера была перемещена, следует произвести перекалибровку всех наконечников всех щупов без пропуска шага получения первых пяти точек.

Автоматический объезд сферы

На данном шаге мастер выполняет объезд сферы с выбранными параметрами. Это может занять несколько минут в зависимости от производительности координатно-измерительной машины.

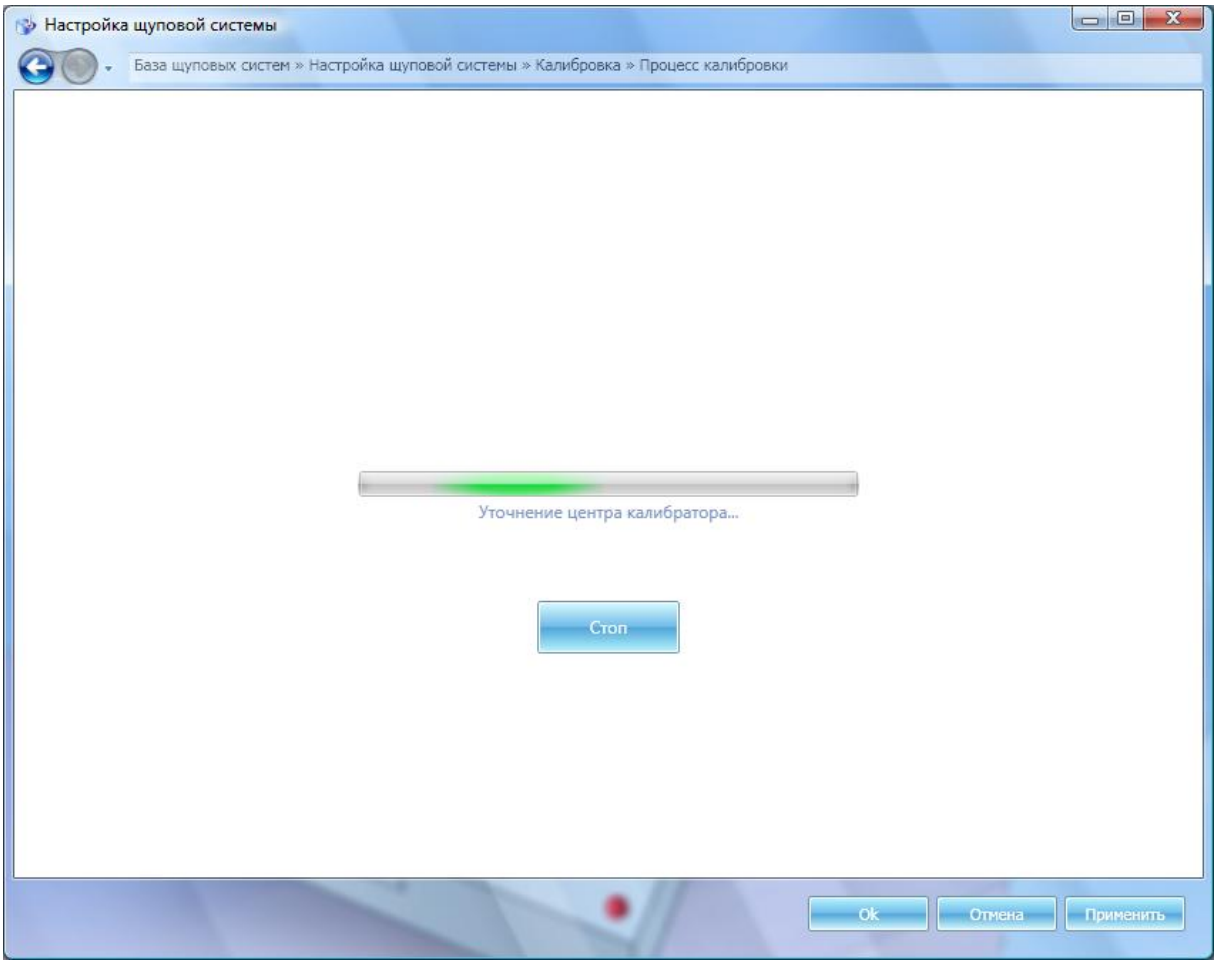


Рисунок 15. Окно автоматического объезда сферы

Процесс автоматического измерения можно остановить, для этого следует нажать кнопку «Стоп» (см. **Рисунок 15**).

Возможные проблемы и их решения:

- Выдается ошибка о непредвиденном касании в процессе измерения сферы.
  - Возможно, в ручном режиме центр был получен недостаточно точно – попробуйте повторить шаг с измерением точек в ручном режиме более аккуратно, расставляя их более равномерно; попробуйте измерять большее количество точек.
  - Возможно, неверно заполнен радиус калибратора или наконечника.
- При переходе к автоматическому обходу машина перемещается в

неверную сторону.
  - Возможно, в ручном режиме получены некорректные точки, повторите калибровку с начала.
  - Возможно, начальные точки были измерены не тем наконечником.
- При переходе к автоматическому обходу машина сталкивается со сферой.
  - Возможно, последняя точка при ручном измерении не находилась в центре допустимой зоны.
  - В случае пропуска шага с ручным измерением, убедитесь, что наконечник был спозиционирован в центр допустимой зоны.
  - Возможно, неверно заполнен радиус калибратора или наконечника.
  - В случае, не описанном выше, обратитесь в поддержку.

Отчет о калибровке

Если процесс калибровки завершился успешно, то появляется отчет о калибровке наконечника, в котором кратко приведены данные о калибровке, а также произведена визуализация зафиксированных отклонений (см. **Рисунок 16).**

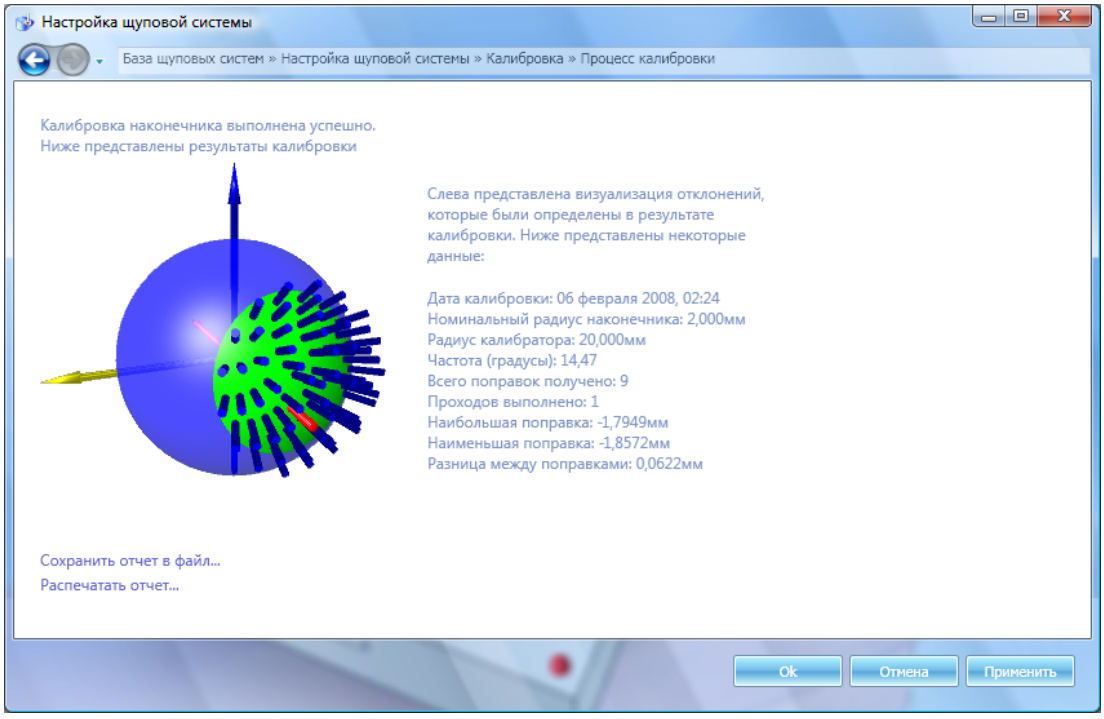


Рисунок 16. Отчет о калибровке наконечника

* На рисунке выше данные были получены с использованием симулятора, поэтому значения могут выглядеть странными по сравнению со значениями на реальной координатно-измерительной машине.

Определение погрешностей

Погрешность щуповой системы – это погрешность текущего установленного щупа. Доступно определение повторяемости щупа (случайной составляющей) и погрешности в соответствии со стандартом ISO 10360.

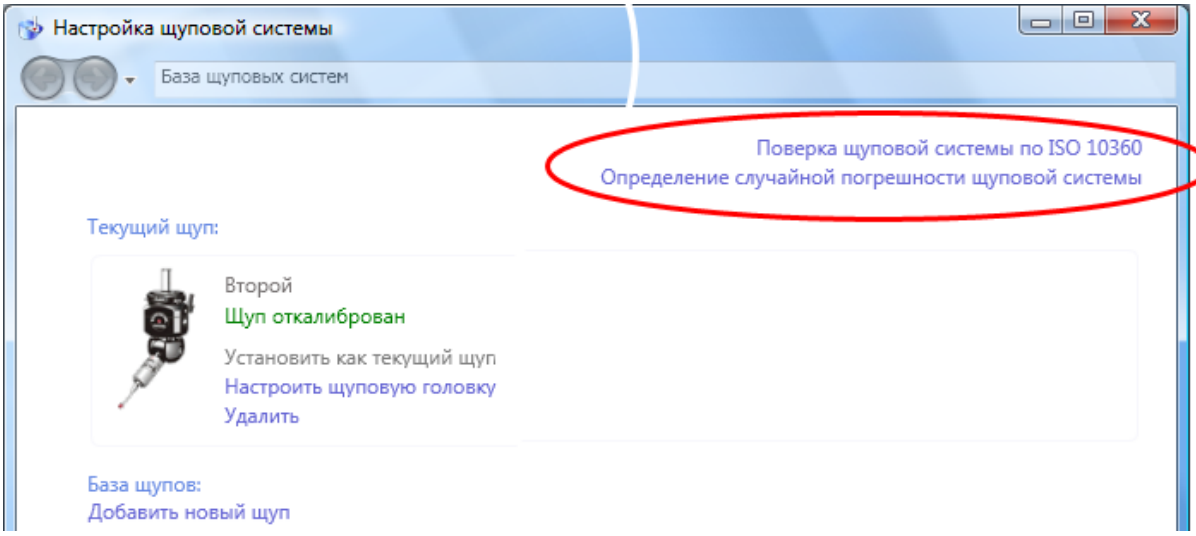


Рисунок 17. Кнопки запуска определения погрешности щуповой системы

До запуска процесса поверки установленный щуп должен быть настроен и откалиброван. Чтобы запустить поверку воспользуйтесь кнопками на главной странице со списком щупов (см. **Рисунок 17**).

Сбор данных для поверки

В зависимости от выбранного типа поверки мастер должен будет произвести измерение определенных точек на сфере. Для того чтобы это выполнить предлагается спозиционировать наконечник к центру зоны, которая показана зеленым цветом на рисунке мастера (см. **Рисунок 18**).

Калибровочная сфера должна оставаться неподвижной. Запрещается перемещать сферу. Если сфера была перемещена, следует произвести перекалибровку всех наконечников всех щупов без пропуска шага получения первых пяти точек.

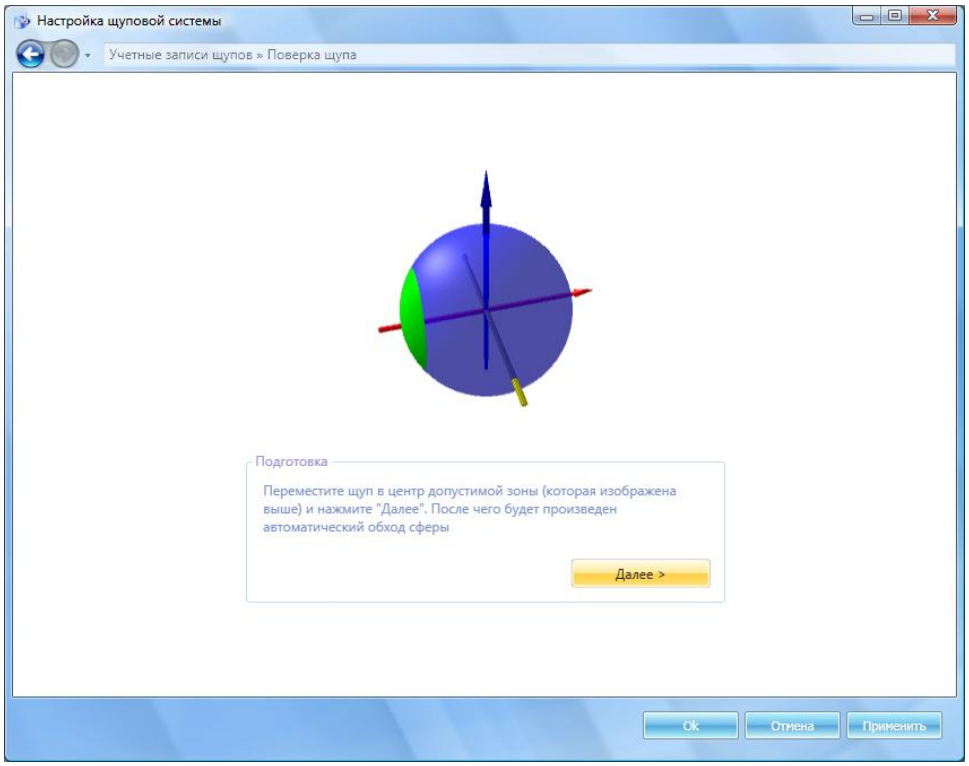


Рисунок 18. Подготовка к сбору данных наконечника

По завершении вывода наконечника в центр предложенной зоны следует нажать кнопку «Далее» - будет произведен автоматический сбор тестовых данных наконечника. Если щуп содержит еще наконечники, то потребуется спозиционировать следующий наконечник в его доступную зону и т.д.

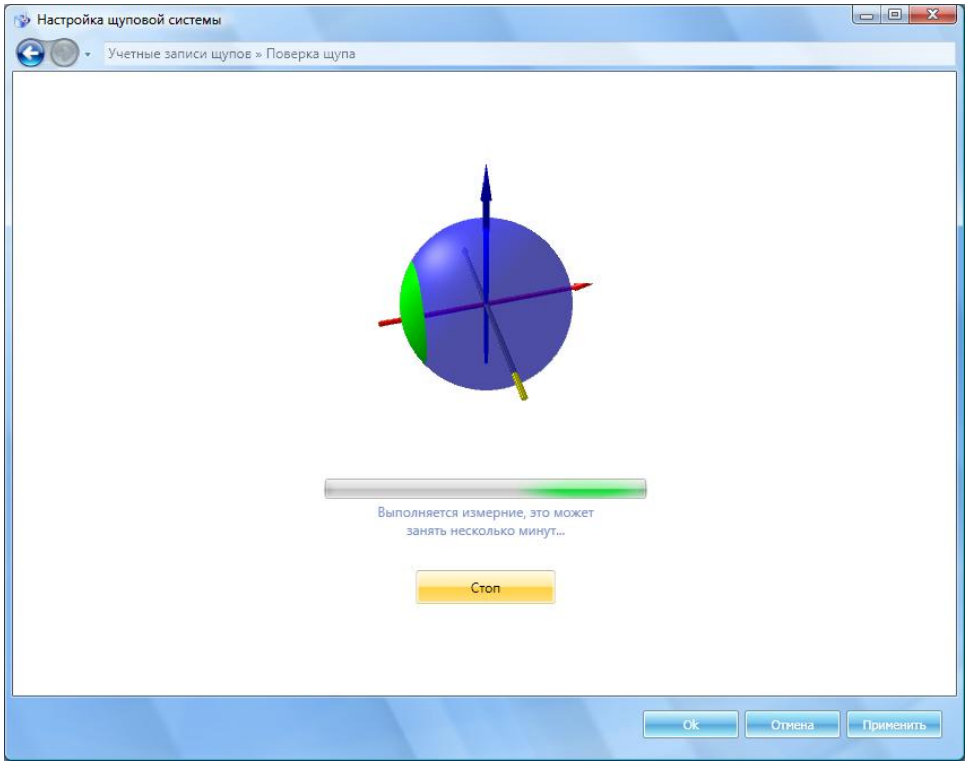


Рисунок 19. Процесс сбора данных для поверки

Повторяемость щуповой системы

Чтобы определить повторяемость щуповой головки, мастер производит измерения точек, равномерно распредленных по поверхности сферы доступной зоны. Затем мастер производит несколько проходов, измеряя эти же самые точки. Отклонение повторно измеренных точек от начальных считается случайным.

Над всеми собранными отклонениями проводятся расчеты, которые помещаются в отчет о повторяемости щуповой головки (см. **Рисунок 20**).

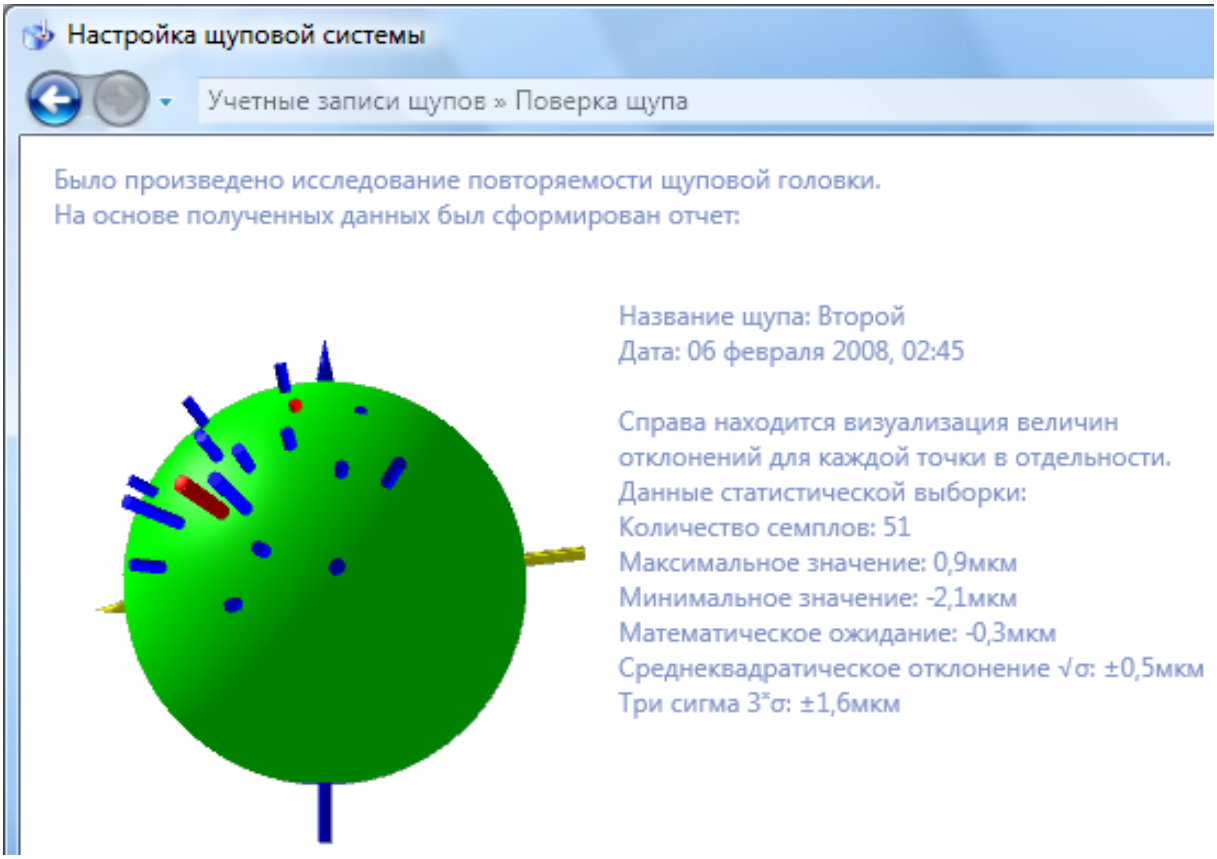


Рисунок 20. Отчет о повторяемости щупа

* На рисунке выше данные были получены с использованием симулятора, поэтому значения могут выглядеть странными по сравнению со значениями на реальной координатно-измерительной машине.

Погрешность по ISO 10360-2

В соответствии с ISO 10360-2 погрешность щуповой головки определяют как сумму максимальных отклонений измеренного профиля в положительную и отрицательную область от средней сферы, рассчитанной по методу наименьших квадратов:

$$MPE_p = \max(d_{i+}) + \max(d_{i-})$$

где d_{i+} отклонение i-й точки от средней сферы в положительную область,

d_{i-} отклонение i-й точки от средней сферы в отрицательную область

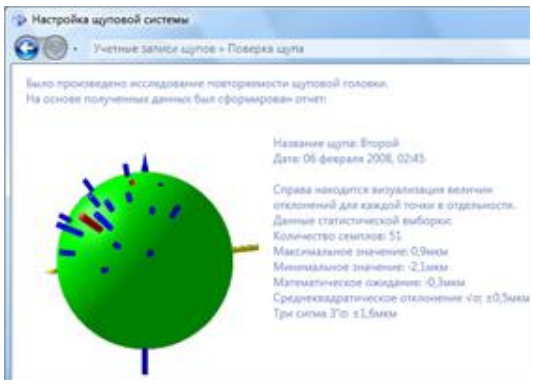
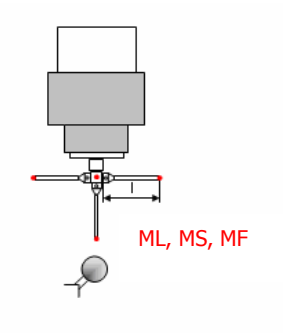


Рисунок 21.
Отчет о поверке по ISO 10360-2

Погрешность по ISO 10360-5



Координатно-измерительная машина может иметь как фиксированный щуп с одним наконечником, так и щуповую систему с несколькими наконечниками (см. **Рисунок 22**) или даже с поворотной головкой. Стандарт ISO 10360-5 определяет погрешности, которые возникают при измерении одного объекта различными наконечниками одного щупа.

Рисунок 22.
Щуп с несколькими наконечниками

Калибровочная сфера измеряется каждым наконечником. Чтобы определить ошибку размера, связанную с использованием многощуповой системы (MS/AS) рассчитывается максимальный разброс точек относительно аттестованного диаметра сферы (см. **Рисунок 23**).

Чтобы определить ошибку формы, связанную с использованием многощуповой системы (MF/AF) рассчитывается максимальное отклонение формы по всем измеренным точкам (см. **Рисунок 23**). Выделяется также ошибка расположения щупов (ML/AL), для ее расчета вычисляется максимальный размах между полученными центрами.

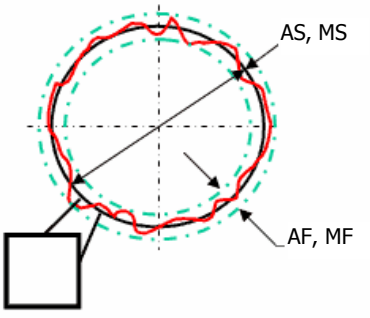


Рисунок 23.
Ошибки размера и формы

Стандартная контактная схема измерения

Назначение и преимущества

Стандартная схема измерения является высокоинтерактивным инструментом для измерения различных деталей с применением различных типов координатно-измерительных машин. Благодаря высокой интеллектуальности данной схемы измерения от пользователя требуются только следующие знания и умения:

- базовые навыки работы с операционной системой Microsoft Windows™.
- знание основных принципов координатных измерений;
- общие знания в области аналитической геометрии;

Данный тип схемы измерения позволяет:

- составлять стратегии измерения автоматически и вручную;
- автоматически составлять маршрут с обходом препятствий;
- производить привязку системы координат машины к системе координат CAD-модели;
- передавать данные для последующих расчетов в мощный инструмент «координатный анализ» (см. «Координатный анализ геометрических параметров детали»);
- визуализировать движения щупа относительно детали;
- производить настройку отчетной формы и выводить ее на печать.

Стандартная схема измерения ориентирована на:

- измерение мейнстрим продукции предприятия;
- интеграцию в CALS-процессы предприятия.

Разработанная схема измерения может быть запущена на любой поддерживаемой координатно-измерительной машине, даже другого типа.

Редактор схемы измерения

Структура редактора

Редактор схемы измерения состоит из окна, включающего главное меню, 3D-просмотр и интегрированный координатный анализ. Круглая кнопка в верхнем левом углу окна

позволяет работать с файлом схемы измерения: сохранить, открыть, создать, отправить на печать и т.д. (см. **Рисунок 24**).

Редактор сохраняет и открывает файлы с расширением *.standardscheme

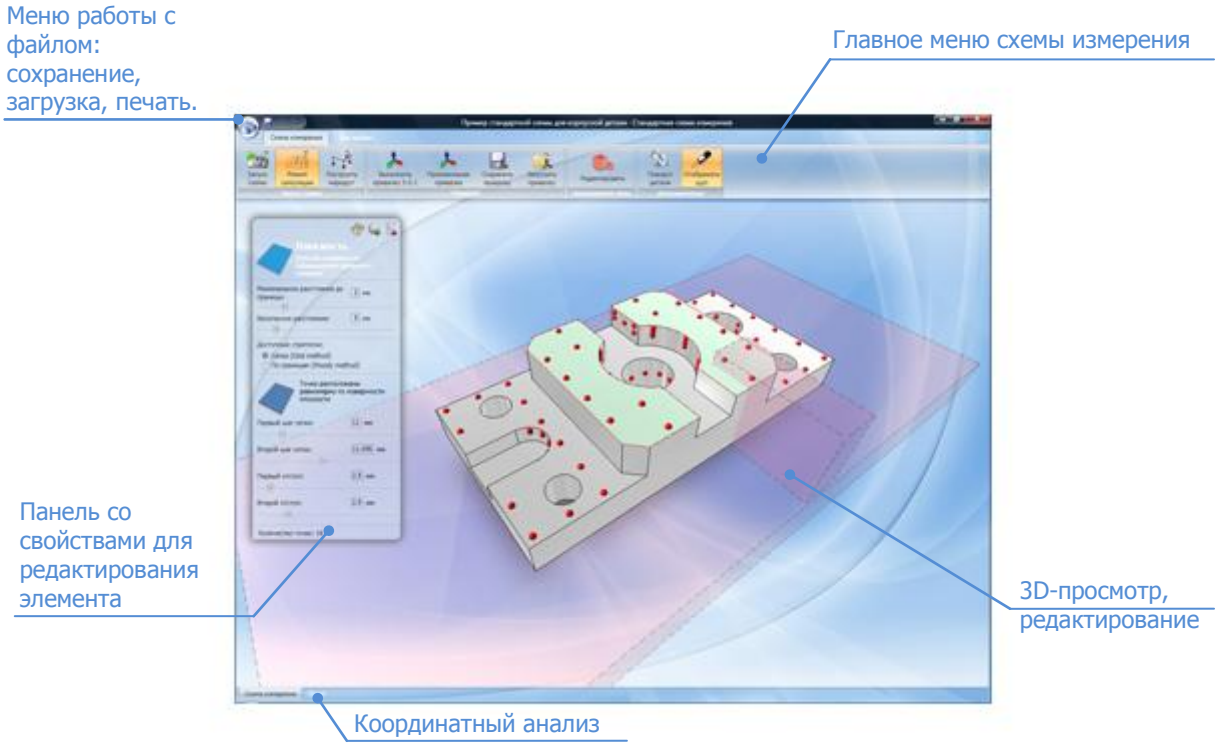


Рисунок 24. Структура редактора

Рекомендуемый порядок работ

Данная схема измерения предоставляет высокоуровневые возможности по управлению координатно-измерительной машиной. Управление машиной программируется интерактивно на основе CAD-модели детали.

Задача схемы измерения – собрать облако точек с реальной детали и передать их для обработки в специальный редактор «Координатный анализ», с помощью которого можно рассчитать различные параметры

Работа с редактором начинается с создания схемы измерения, и первый шаг заключается в выборе CAD-модели детали. CAD-модель можно загрузить из формата *.step, *.stp или из внутреннего формата CAD-модели ЧелябНИИконтроль (*.cad).

В процессе работы со схемой измерения изменить CAD-модель нельзя

После загрузки модели детали следует создать стратегии измерения для элементов. Стратегия измерения элемента – это принцип размещения измеряемых точек на поверхности элемента (см. **Рисунок 25**). Выбор стандартной стратегии ложится на разработчика схемы и обуславливается дальнейшими расчетами, в которых будут участвовать измеренные точки. Стратегия измерения элемента может быть составлена вручную.

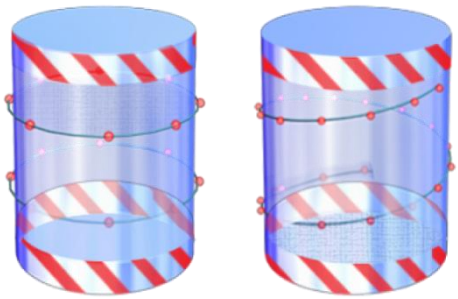


Рисунок 25.Пример двух стратегий измерения цилиндра

Если ориентация CAD-модели не совпадает с ориентацией детали на столе можно повернуть модель на некоторый угол. Углы поворота детали задаются приблизительно. Правильная ориентация необходима для более удобной работы в редакторе и адекватной визуализации процесса измерения.

Если деталь закреплена специальными приспособлениями, следует отметить эти места, расставив запрещающие зоны. При измерении машина будет объезжать запрещенные зоны и не столкнется с креплениями.

Полученную схему измерения можно проверить на симуляторе, а затем запустить на координатно-измерительной машине. Такая особенность позволяет проектировать и выполнять схемы измерения без использования реальной машины, что сокращает время на разработку и повышает безопасность измерения. Это дает возможность экономить время реальной машины, затраты на использование которой обычно достаточно велики. Полученные данные автоматически передаются в координатный анализ, где формируется отчет об измерении.

Подробно о редактировании отчетной формы и выводе результатов следует читать в главе «Координатный анализ геометрических параметров детали».

Поворот CAD-модели

CAD-модель находится в собственной системе координат, где она может быть произвольно ориентирована и не совпадать с расположением детали на столе КИМ. Чтобы работа в редакторе была удобнее, рекомендуется повернуть CAD-модель так,

чтобы она совпадала с расположением детали на столе КИМ. Для этого в главном меню следует выбрать «Схема измерения → Поворот детали».

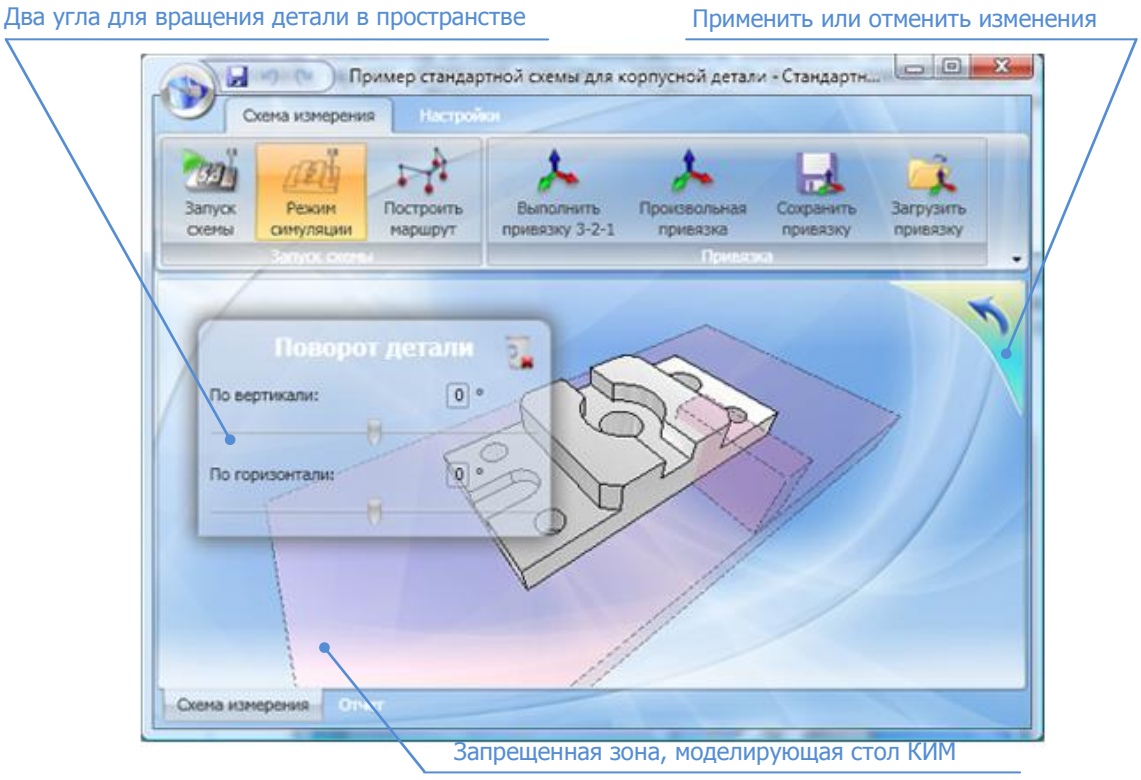


Рисунок 26. Вращение CAD-модели в редакторе

Привязка CAD-модели

Для того чтобы производить перемещения машины вокруг детали, измерять точки на ее поверхности необходимо узнать как расположена деталь на столе (см. **Рисунок 27**). Такой процесс называется привязка текущего расположения детали к CAD-модели.

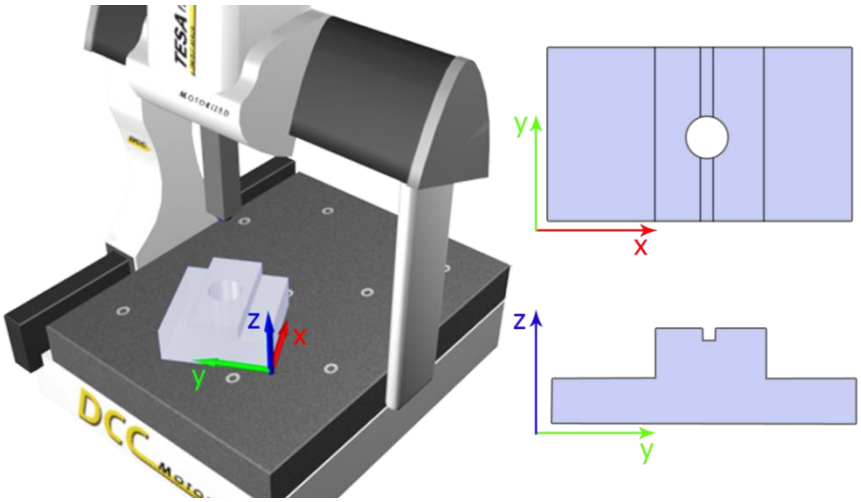


Рисунок 27. Привязка CAD-модели

Привязка производится в ручном режиме. Пользователю необходимо измерить несколько точек, расположенных на определенных поверхностях детали, после чего будет выполнен расчет перевода координат из системы координат модели в систему координат детали.

Привязка 3-2-1

Привязка 3-2-1 выполняется очень быстро. Для данной привязки требуется обязательное наличие трех непараллельных плоскостей.

Чтобы начать привязку следует нажать кнопку в главном меню «Выполнить привязку 3-2-1». Первым шагом мастера будет выбор трех непараллельных плоскостей. При выборе плоскостей рекомендуется руководствоваться следующими правилами:

- Выбирать плоскости расположенные перпендикулярно осям машины. Это связано с тем, что при измерении в ручном режиме важно производить измерение, двигаясь по нормали к поверхности детали, а проще всего это сделать, если поверхность перпендикулярна одной из осей.
- Рекомендуется выбирать плоскости с большой площадью.

После выбора плоскостей мастер по очереди будет предлагать измерить три, две и одну точку на соответствующих плоскостях, поочередно подсвечивая их. Рекомендуется выбирать наиболее удалённые друг от друга точки. При измерении точек необходимо менять щуп на тот, которым предполагается произвести касание (для этого можно воспользоваться элементом управления в нижнем левом углу, см. **Рисунок 28**).

Запрещается:

- первые три точки выбирать лежащие на одной прямой;
- две точки на второй плоскости измерять на прямой, перпендикулярной линии пересечения первой и второй плоскости.

Если привязка выполняется некорректно:

- проверьте, что не было измерено одинаковых точек;
- постарайтесь измерять точки наиболее удаленные друг от друга;
- попробуйте выбрать плоскости с большей площадью;
- проверьте, что точки на отмеченных поверхностях были измерены соответствующими щупами.

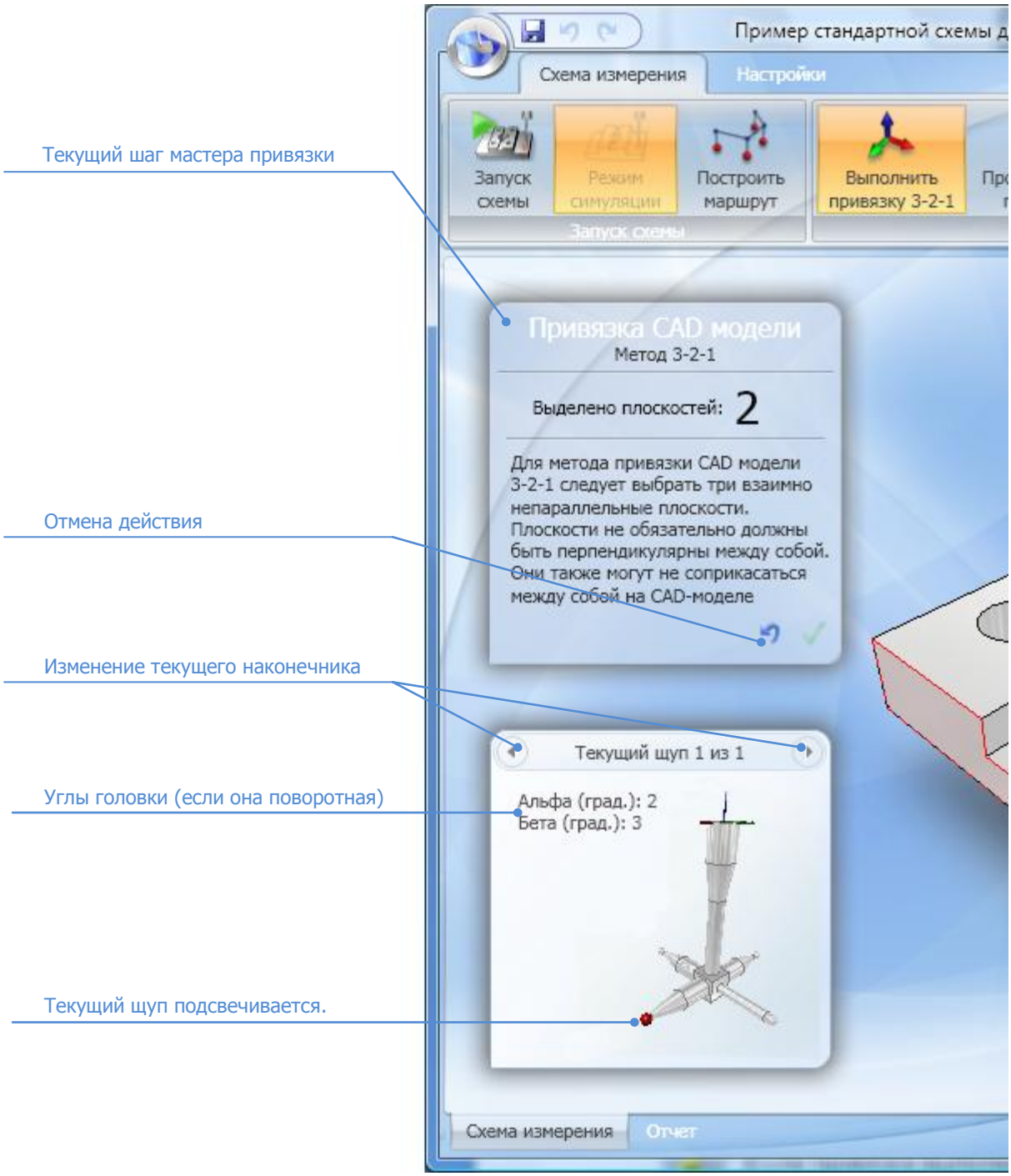


Рисунок 28. Привязка 3-2-1

Привязка по примитивам

Для данной привязки требуется выбрать комплект баз, т.е. набор элементов, которые ограничивают все степени свободы детали. Элементы ограничивают деталь в порядке убывания числа лишаемых степеней свободы (см. **Рисунок 29**). Например, три непараллельных плоскости, или две непараллельных плоскости и цилиндр, ось которого непараллельна прямой, получаемой в результате пересечения плоскостей и т.д.

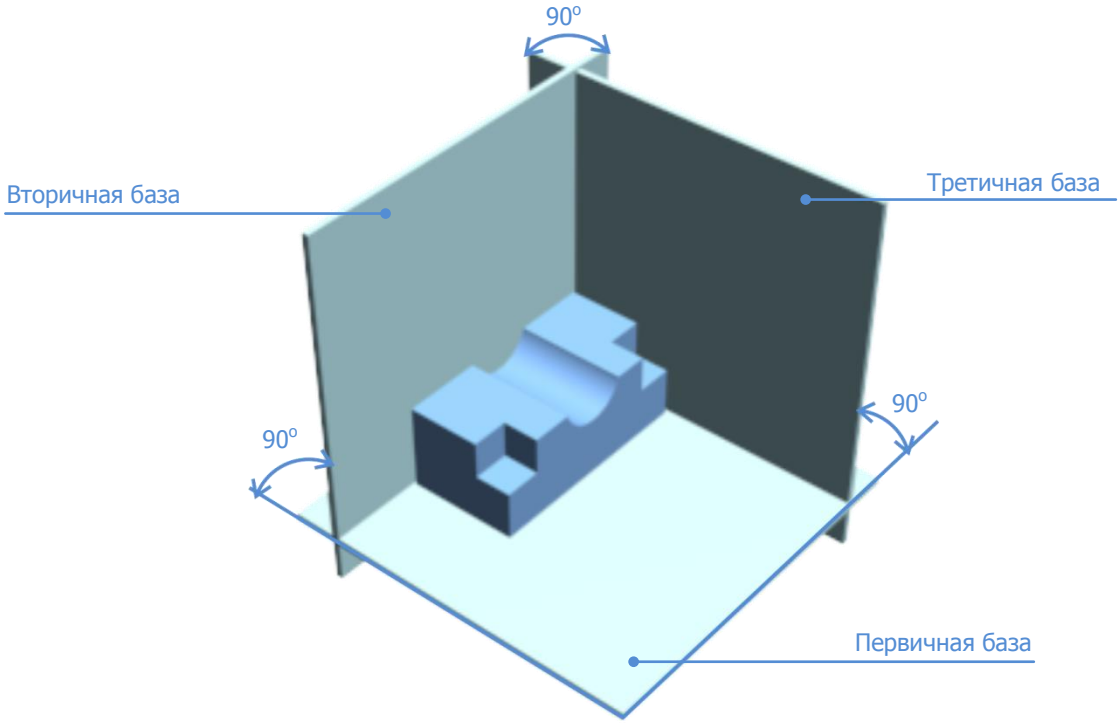


Рисунок 29. Комплект баз

В качестве элементов для привязки можно выбирать:

- Плоскость;
- Цилиндр;
- Конус.

Чтобы начать привязку следует нажать кнопку в главном меню «Привязка по примитивам». Первым шагом мастера будет выбор нескольких элементов, ограничивающих степени свободы (см. **Рисунок 30**).

При выборе элементов рекомендуется выбирать поверхности с большей площадью.

После выбора плоскостей мастер по очереди будет предлагать измерить три, две и одну точку на соответствующих плоскостях, поочередно подсвечивая их. Рекомендуется выбирать наиболее удалённые друг от друга точки. При измерении точек необходимо менять щуп на тот, которым предполагается произвести касание (для этого можно воспользоваться элементом управления в нижнем левом углу).

Рисунок 30. Панель для выбора элементов для привязки

Универсальная привязка

Данная функция позволяет производить привязку системы координат детали к системе координат CAD-модели. Универсальную привязку можно применять даже к деталям со сложной геометрией, например, содержащим такие поверхности как NURBS.

Мастер произвольной привязки предлагает пользователю выбрать щуп (для этого можно воспользоваться элементом управления в нижнем левом углу (см. **Рисунок 32**)), выбрать поверхность левой кнопкой мыши и измерить на ней ряд точек, затем выбрать другую поверхность и щуп и снова измерить ряд точек. Измерение следует продолжать до тех пор, пока величина «Максимальное отклонение точек» не станет удовлетворительной, а измеренных поверхностей не будет достаточно чтобы ограничить степени свободы детали.

Важно понимать, что измеряемые точки далеки от идеальных, и будут получены с некоторой погрешностью, и это окажет влияние на качество привязки. Влияние на качество привязки окажется решающим, если точки были сконцентрированы в одном месте детали (см. **Рисунок 31**).

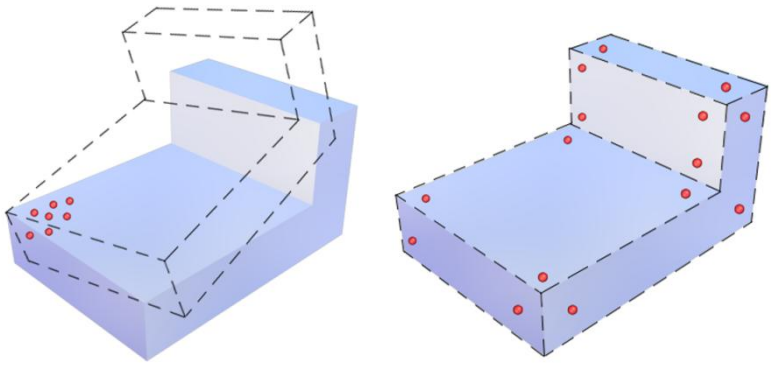


Рисунок 31. Правильное и неправильное распределение точек

В случае если привязка выполняется некорректно:

- постарайтесь измерять точки наиболее удаленные друг от друга;
- проверьте, что точки на отмеченных поверхностях были измерены соответствующими щупами;
- возможно, расположение точек оставляет детали слишком большую свободу см. **Рисунок 31**.

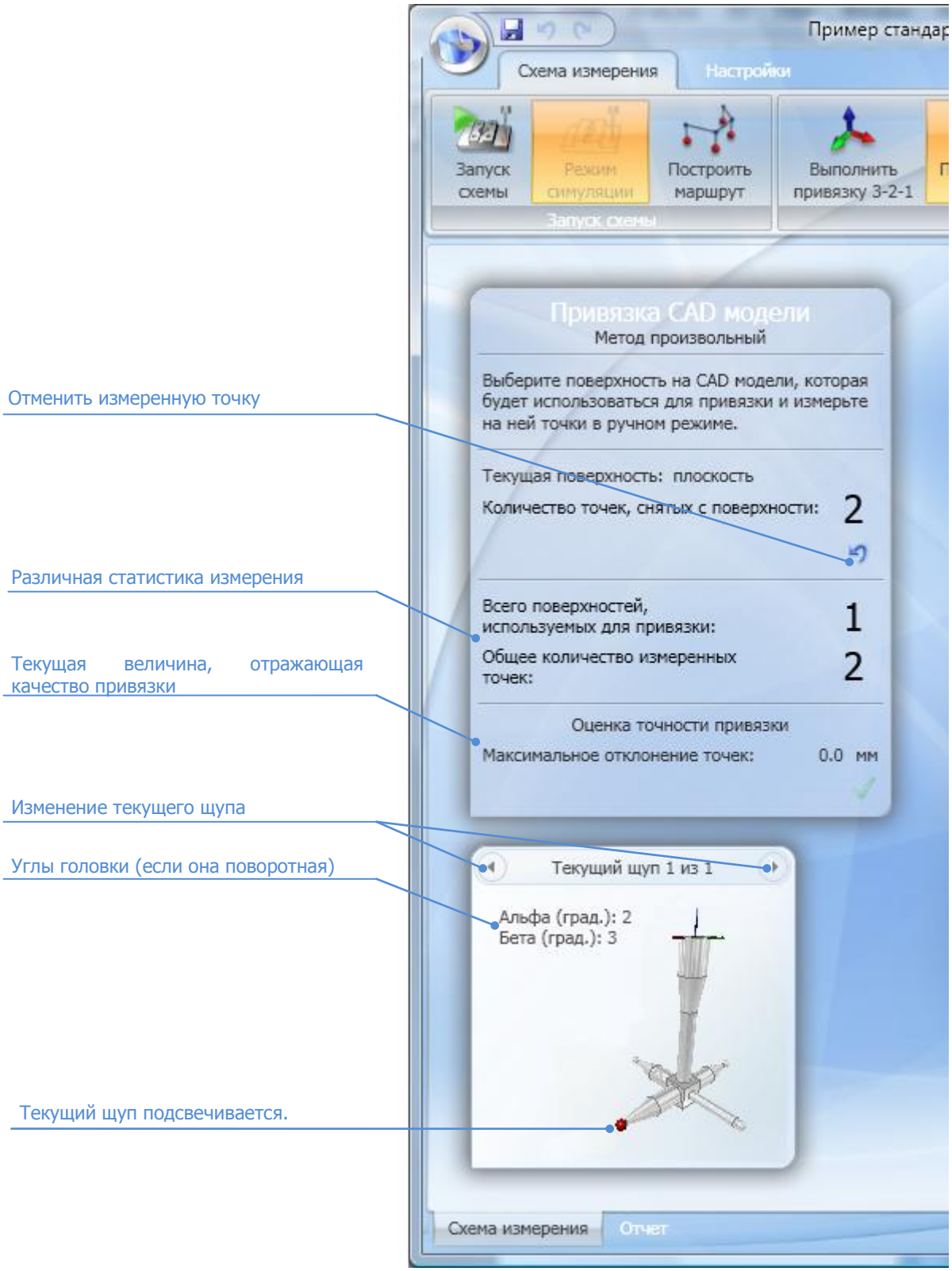


Рисунок 32. Универсальная привязка

Настройка стратегий измерения

Чтобы создать стратегию измерения для элемента детали следует вызвать контекстное меню, выбрать «Создать» - появится панель для настройки стратегии измерения (см. **Рисунок 34**).

Иногда при экспорте из CAD-редактора элементы «разбиваются» на несколько кусков (см. **Рисунок 33**). Редактор позволяет логически объединять элементы в один. Для этого необходимо воспользоваться кнопкой в верхней части панели для настройки стратегии измерения (см. **Рисунок 34**).

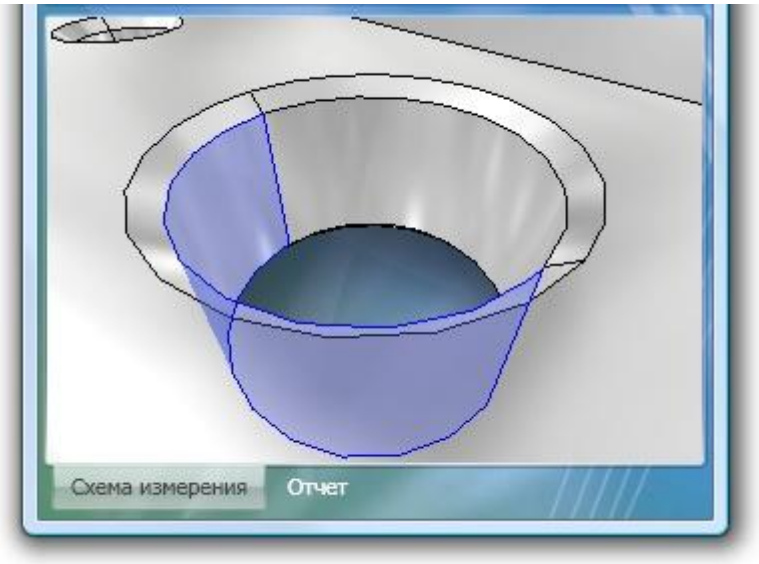


Рисунок 33. Пример элемента, состоящего из нескольких частей

Можно удалить стратегию измерения воспользовавшись контекстным меню или кнопкой в верхнем правом углу панели (см. **Рисунок 34**).

Настройка стратегии измерения начинается с автоматической расстановки точек одной из стандартных стратегий, а затем редактируется в ручном режиме.

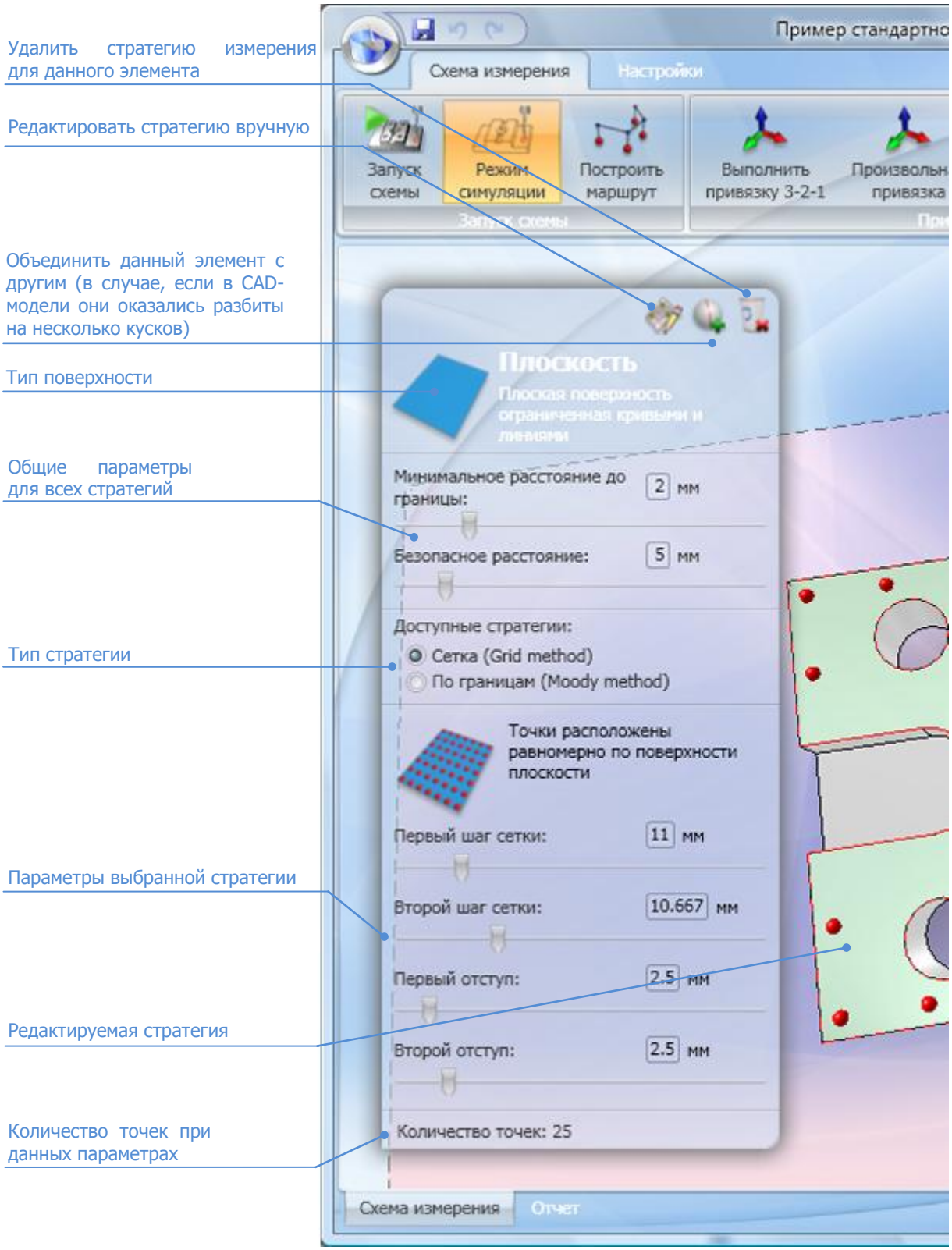


Рисунок 34. Редактирование стратегии измерения

Стандартные стратегии

Независимо от типа элемента для всех стратегий доступны два параметра: минимальное расстояние до границы и безопасное расстояние (см. **Рисунок 34**). Параметр «Безопасное расстояние» определяет, с какого расстояния от детали щуп начнет движение к поверхности. Параметр «Минимальное расстояние до границы» позволяет отсечь точки, которые оказались близко к границам элемента.

В зависимости от типа элемента доступны различные стратегии измерения.

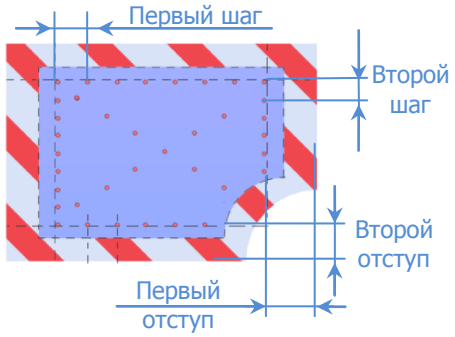


Рисунок 35. Сетка на плоскости

Сетка на плоскости. Позволяет разместить точки равномерно по поверхности плоскости. Задаются два отступа с разных направлений, а также шаг в одном направлении (т.е. расстояние между соседними точками) и шаг в другом направлении (см. **Рисунок 35**).

Граничный метод на плоскости (англ. Moody method). Позволяет расположить измеряемые точки в шести направлениях (на шести отрезках) (см. **Рисунок 36**). Позволяет регулировать шаг, с которым расставляются точки на отрезках, а также учитывает отступы.

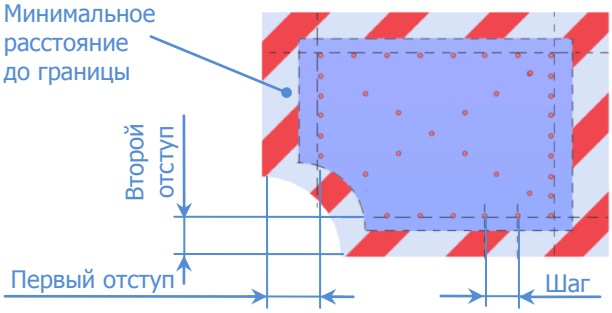


Рисунок 36.
Граничный метод на плоскости

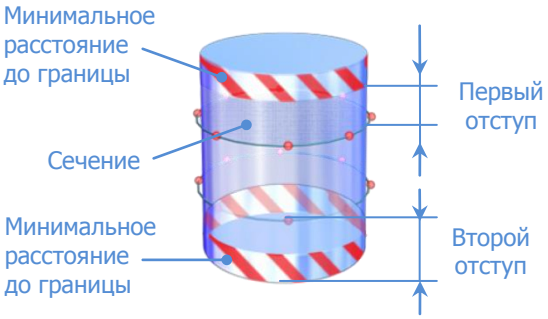


Рисунок 37.
Измерение цилиндра по сечениям

Стратегия для цилиндра по сечениям. Точки располагаются в нескольких сечениях. Есть возможность регулировать количество сечений, количество точек в сечении и отступы сверху и снизу (см.**Рисунок 37**).

Стратегия для цилиндра по спирали. Точки располагаются на спирали, проходящей по поверхности цилиндра с указанным шагом. Имеется возможность регулировать количество точек на шаг спирали, а также отступы сверху и снизу (см. **Рисунок 38**).

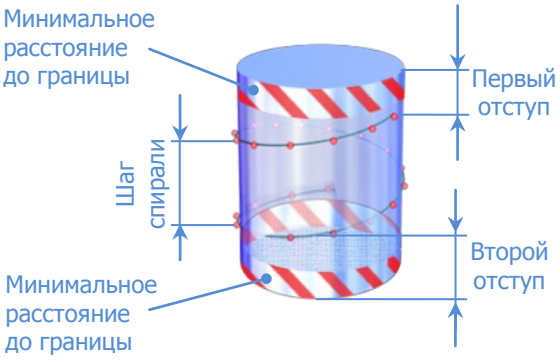


Рисунок 38.Измерение цилиндра по спирали

Стратегии для конуса подобны стратегиям цилиндра, также доступны по сечениям и по спирали (см. **Рисунок 39**).

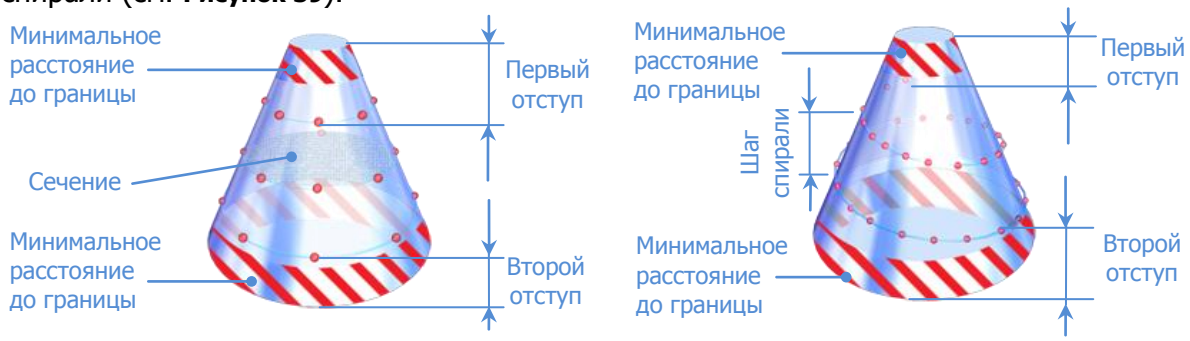
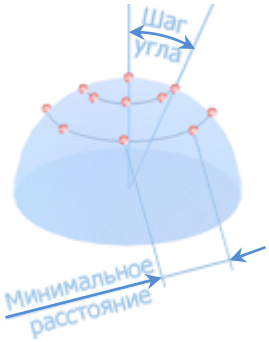


Рисунок 39. Стратегии для конуса

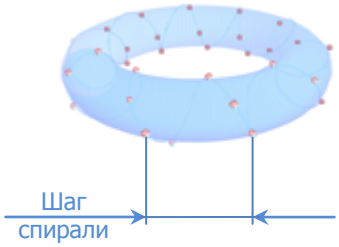


Для сферы доступна стратегия сеткой, позволяющая расставлять точки равномерно по поверхности сферы. Расположение точек регулируется за счет угла между сечениями и расстояния между точками в сечении (см. **Рисунок 40**).

Рисунок 40.
Стратегия для сферы по сетке

Стратегия измерения тора по сечениям. Можно управлять количеством сечений и точек в сечении (см. **Рисунок 41**).

Рисунок 41.
Сечение тора



Стратегия измерения тора по спирали. Имеется возможность управлять шагом спирали и количеством точек на один виток (см. **Рисунок 42**).

Рисунок 42.
Измерение тора по спирали

Для сплайновой поверхности предусмотрена стратегия, распределяющая равномерно точки по поверхности (см. **Рисунок 43**).

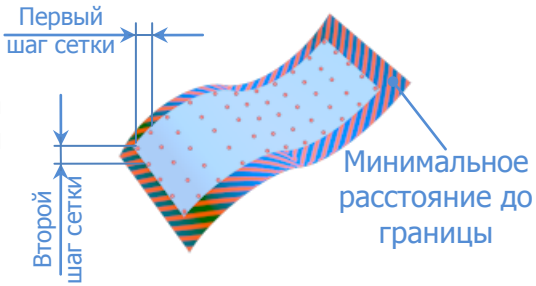


Рисунок 43.
Стратегия для сплайновой поверхности

Ручное составление стратегии

В случае если ни одна из стандартных стратегий не подходит, то в программе предусмотрена возможность размещать точки вручную. Для этого следует нажать кнопку «Редактировать точки» в верхнем правом углу панели для настройки стратегии измерения (см. **Рисунок 34**) после чего редактор перейдет в режим ручного редактирования точек (см. **Рисунок 44**).

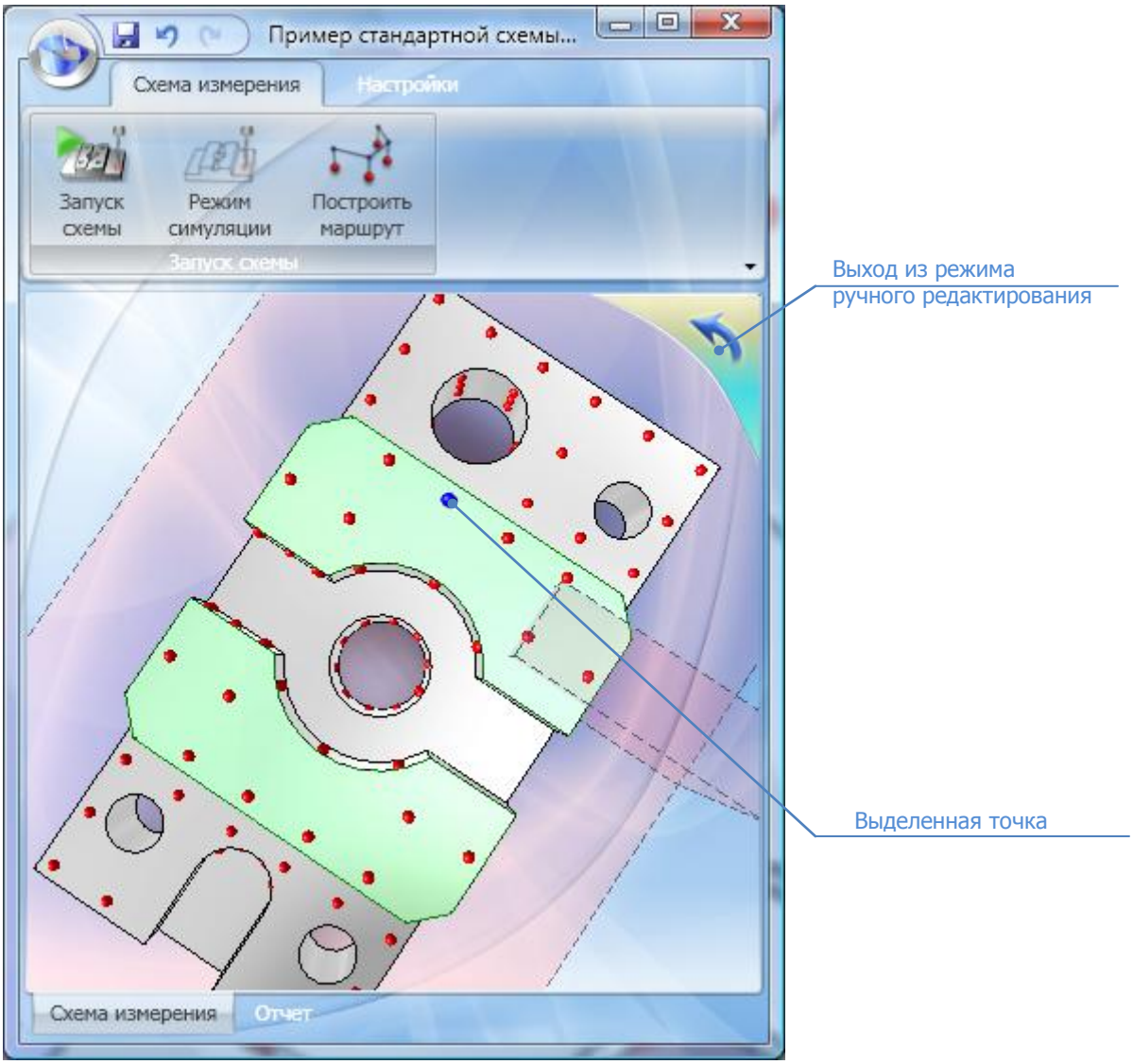


Рисунок 44. Ручное редактирование стратегии

Для редактирования точек в данном режиме следует выделить точку и «перетащить» ее, удерживая левую клавишу мыши. Добавление новой точки можно сделать двойным щелчком, удаление – клавишей Delete.

При изменении параметров автоматической стратегии, расставленные вручную точки, будут утеряны.

Расстановка запрещенных зон

Как правило, измеряемое изделие закрепляется на столе с помощью специальных креплений. Если не учитывать их геометрическое расположение, то в процессе измерения машина столкнется с одним из них. Чтобы исправить ситуацию, можно использовать CAD-модель с расположенными на ней зажимами, а можно воспользоваться инструментом, который позволит отметить часть пространства как недоступную для машины.

Для того чтобы войти в режим расстановки запрещенных зон нужно выбрать в главном меню «Запрещенные зоны → Редактировать» (см. **Рисунок 45**).

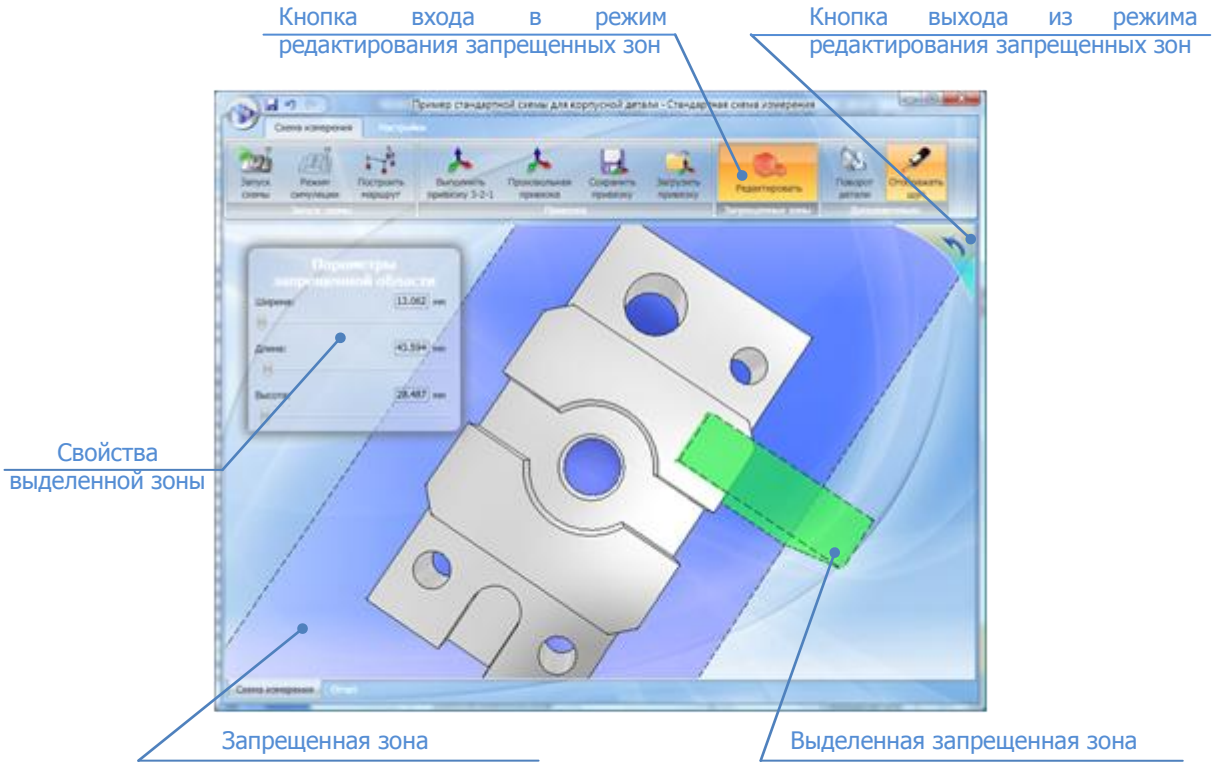


Рисунок 45. Редактирование запрещенных зон

Выделить запрещенную зону можно левой клавишей мыши. Удаление зоны осуществляется клавишей Delete. Чтобы создать новую зону следует сделать двойной клик, потянуть мышь в сторону для образования основания параллелограмма, затем выполнить клик и отрегулировать зону по высоте.

Изменять параметры (длину, ширину, высоту) позволяет панель со свойствами, которая отображается для выделенной зоны. Перемещение зоны производится следующим образом: зажав левую клавишу мыши на одной из сторон зоны, смещать зону в направлении перпендикулярном данной стороне.

Запуск измерения

Для запуска схемы на реальной машине следует убедиться, что кнопка «Режим симуляции» неактивна, и нажать кнопку «Запуск схемы» (см. **Рисунок 46**). Машина произведет подключение, выход в ноль, затем начнут выполняться запрограммированные действия.

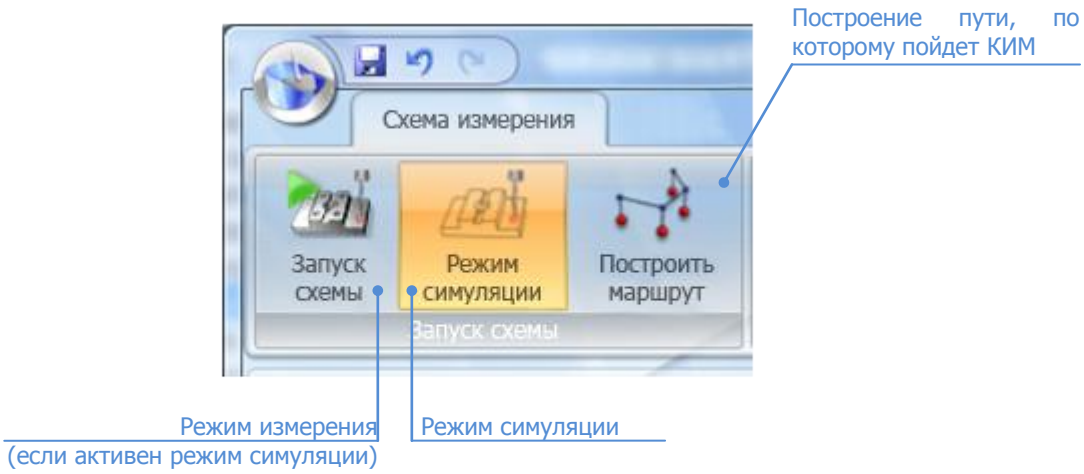


Рисунок 46. Главное меню схемы измерения

Существует возможность запустить измерение только одного элемента, для этого следует правой клавишей мыши вызвать контекстное меню данного элемента и выбрать «Измерить» (см. **Рисунок 47**).

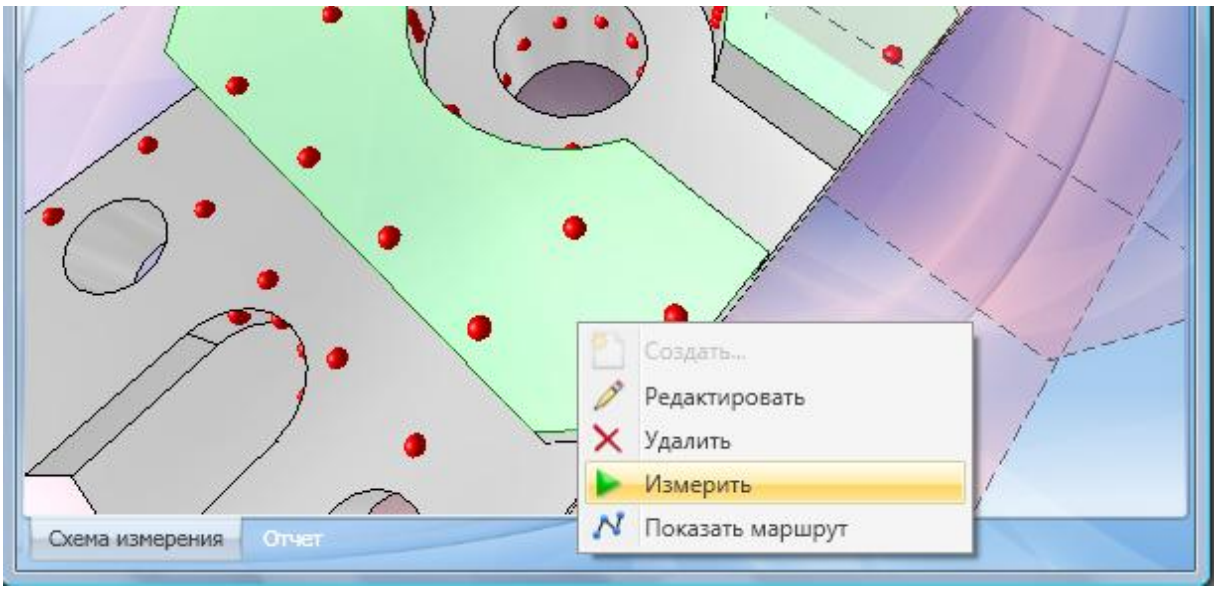


Рисунок 47. Запуск измерения одного элемента

Чтобы измерить несколько элементов, можно выделить элементы с зажатой клавишей Ctrl, вызвать контекстное меню и выбрать «Измерить» (см. **Рисунок 48**).

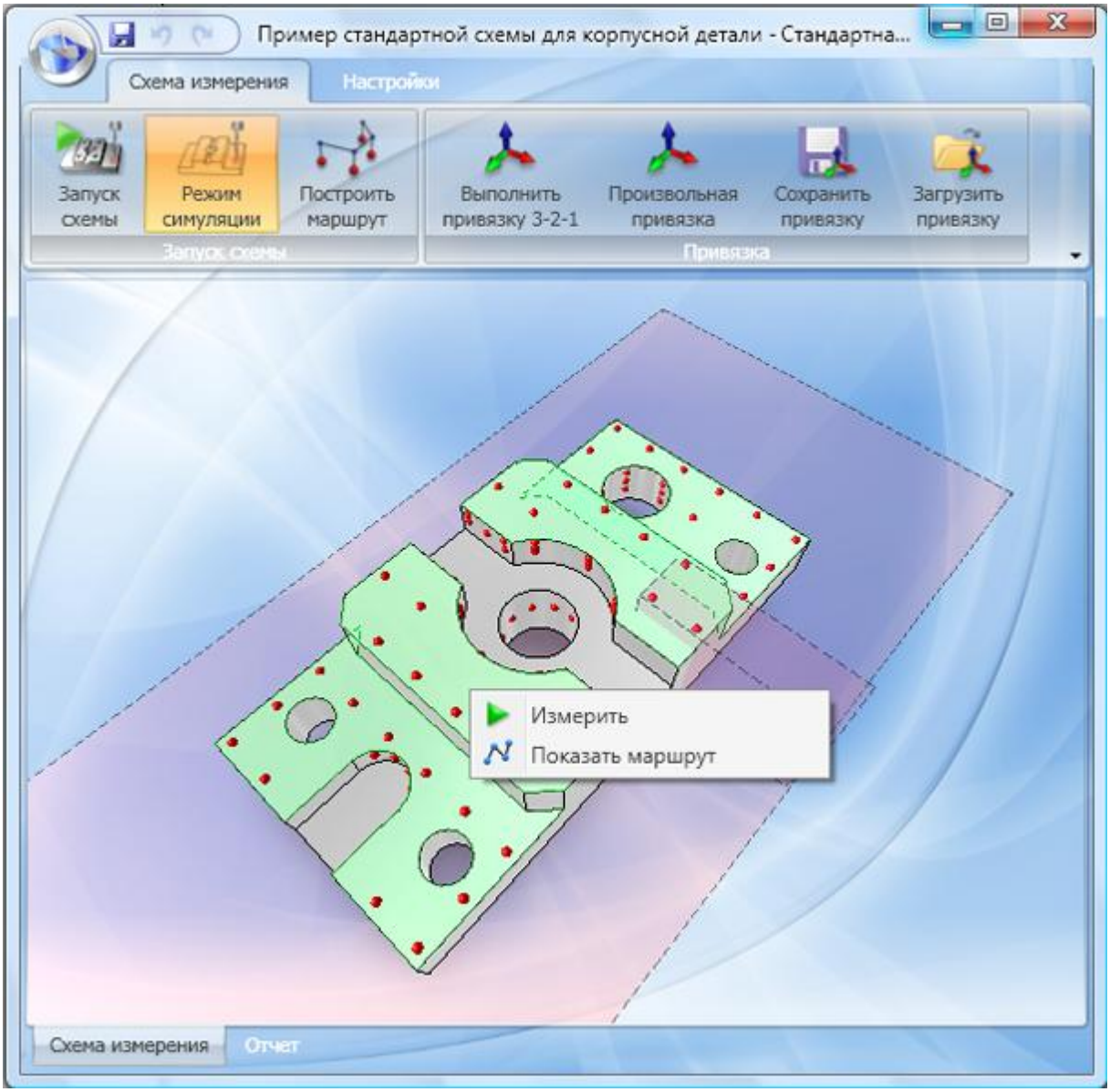


Рисунок 48. Запуск измерения нескольких элементов

Симуляция измерения

Для отладки программы измерения детали предназначена специальная функция – симуляция. Симуляция позволяет разрабатывать и проверять схемы измерения без использования настоящей координатной машины, такой способ программирования КИМ называется offline-программированием.

Для запуска симуляции необходимо открыть для редактирования схему измерения и нажать в главном меню «Запуск симуляции». Симулятор практически полностью повторяет поведение реальной координатно-измерительной машины, поэтому измерение на виртуальной машине сильно приближено к реальному измерению.

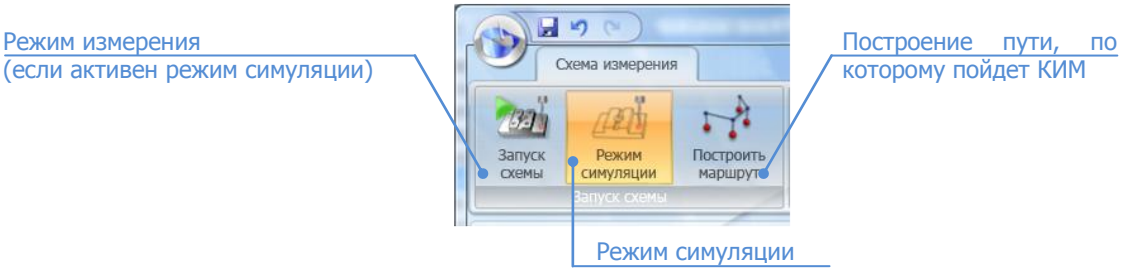


Рисунок 49. Запуск симуляции измерения

В начале работы с симулятором потребуется добавить модель детали. Для этого нажмите кнопку «Добавить деталь», выберите подходящую (деталь может быть загружена из файла в формате *.obj, *.stp, *.step, *.stl, *.cad).

Рисунок 50. Внешний вид симулятора

После этого нажмите кнопку в нижнем левом углу окна, чтобы продолжить процесс запуска схемы измерения. Дальнейшая работа с симулятором не отличается от работы с реальной машиной, необходимо выполнять запрограммированные действия, например, измерять точки в ручном режиме, используя джойстик (в случае, если джойстика нет, то на экране появится эмулятор, см. **Рисунок 51**).

Рисунок 51. Эмулятор джойстика

Расширенная контактная схема измерения

Назначение и преимущества

Расширенная схема измерения предназначена для пользователей, обладающих следующими знаниями и умениями:

- знание принципов координатных измерений;
- знания в области аналитической геометрии;
- умение программировать на алгоритмическом языке высокого уровня.

Данный тип схемы измерения позволяет:

- получать координаты точек, измеренных вручную;
- программировать перемещения и измерения на портальной КИМ;
- создавать собственные системы координат и использовать их для управления машиной;
- производить произвольные расчеты, используя встроенный язык;
- прикреплять мультимедийную информацию для оператора;
- выводить измеренные параметры в отчет;
- производить настройку отчетной формы и выводить ее на печать.

Расширенная схема измерения ориентирована на:

- студентов, желающих освоить принципы координатных измерений;
- специалистов, которым необходимо разработать нестандартное измерение.

Редактор схемы измерения

Структура редактора

Окно редактора схемы измерения содержит меню работы с файлом, где можно открыть, сохранить, экспортировать файл, главное меню программы для запуска схемы измерения, меню «Настройки» для изменения параметров щуповой системы, меню «Анализ», содержащее набор инструментов для оформления отчета, а также вкладки «Программа» для написания текста программы, «Отчет» для оформления отчета и «Слайды» для создания наглядных заметок и примечаний для пользователя или оператора (см. **Рисунок 52**).

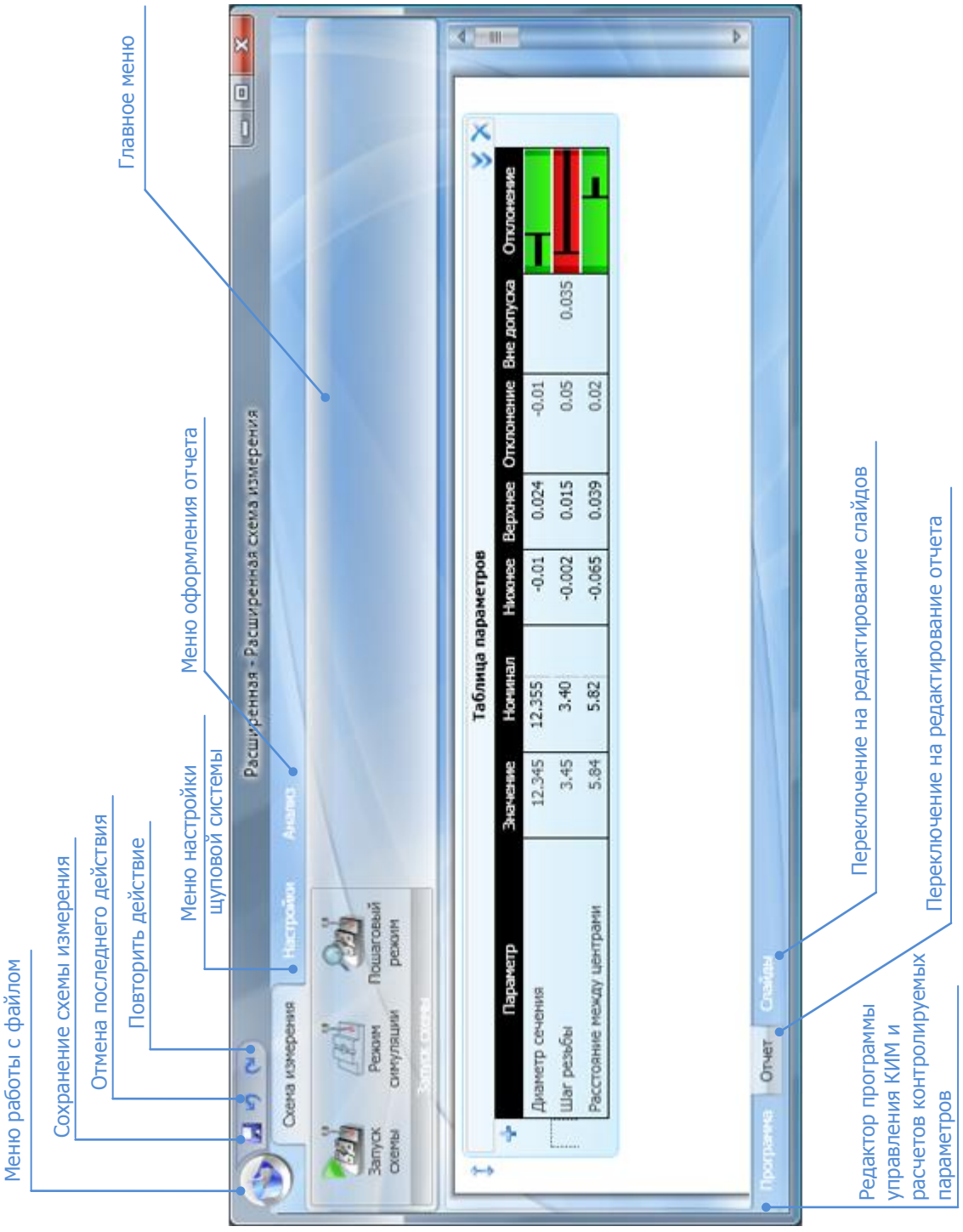


Рисунок 52. Окно редактора расширенной контактной схемы измерения

Рекомендуемый порядок работы

Данная схема измерения предоставляет низкоуровневые возможности по управлению координатной измерительной машиной. Управление машиной программируется на специальном языке координатных измерений (Measurements Scripting, см. «Язык для математической обработки данных координатного анализа»). Имеется возможность управлять машиной в ее системе координат или создать собственную систему координат и задавать управление уже в новой системе координат.

Изначально нужно определить какие элементы детали будут измеряться, какие элементы можно взять за базу. Если деталь располагается в зафиксированном положении на измерительном столе, то управление можно производить в системе координат машины, в противном случае необходимо в ручном режиме получить ряд точек и составить систему координат детали (подробнее см. «Математическое базирование», на стр. 47).

После того, как решена проблема с положением детали относительно машинной системы координат, нужно сформировать стратегию измерения, т.е. запрограммировать перемещение и измерение машины. Задача данного этапа собрать облако точек (подробнее см. «Составление стратегии измерения», на стр. 49). Стратегию измерения машины необходимо отладить, т.е. проверить правильность ее выполнения. Отладку рекомендуется производить не на реальной машине, а на симуляторе (см. «Проверка схемы измерения», на стр. 49).

После того, как получено облако точек следует производить аппроксимацию примитивов и расчет контролируемых параметров (см. «Расчет контролируемых параметров», на стр. 50).

Рассчитанные параметры будут отображены в отчете, где можно настроить их представление, допустимые отклонения, квалитеты и т.д. Если данная схема измерения будет использоваться многократно оператором, который не знает об особенностях реализации схемы измерения, следует добавить слайды с указаниями для оператора.

Подробно о редактировании отчетной формы и выводе результатов следует читать в главе «Координатный анализ геометрических параметров детали».

После удачного прохождения симуляции можно запустить программу на реальной машине (см. «Запуск измерения»,на стр. 39).

Управление машиной

Для управления машиной в программе используется объект Cmm. Он позволяет производить линейную интерполяцию, измерять координаты точек в ручном и автоматическом режиме.

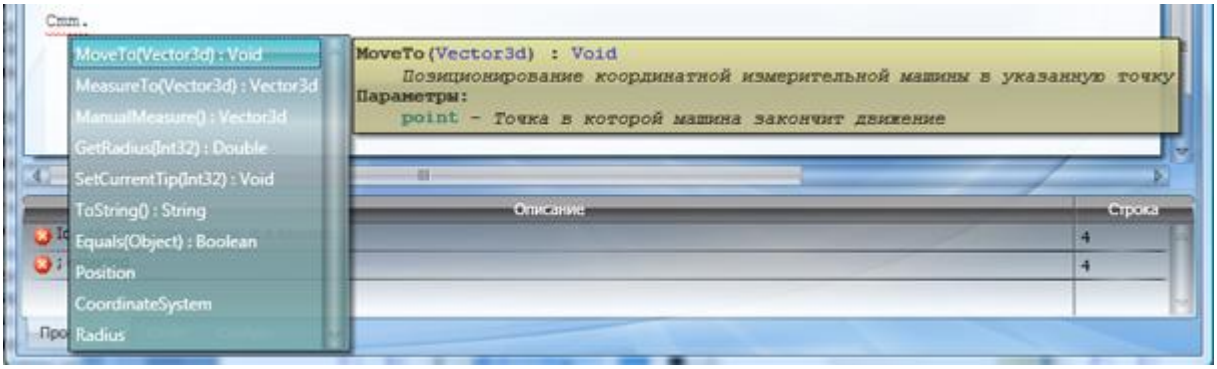


Рисунок 53. Программирование управления машиной

Позиционирование машины

Для позиционирования машины используется линейная трехосная интерполяция, которая позволяет переместить щуп по кратчайшему пути от текущей точки до точки назначения.

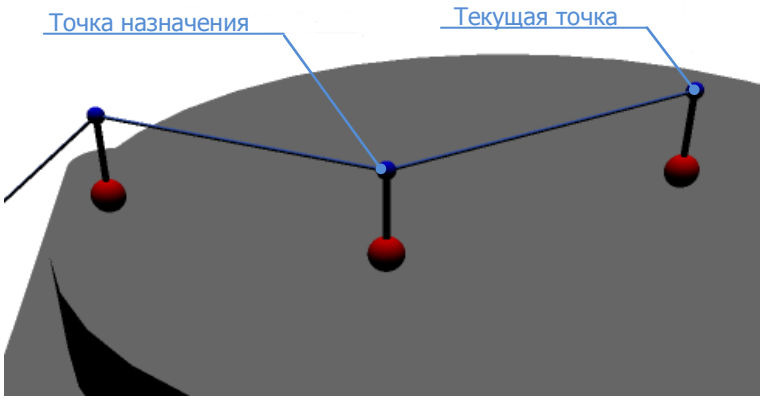


Рисунок 54. Позиционирование машины

Для позиционирования машины предназначена функция Cmm.MoveTo(Vector3d). Например, так можно переместить машину в позицию (10, 10, 10):

```
Vector3d position = new Vector3d(10, 10, 10);
Cmm.MoveTo(position);
```

Измерение точки в автоматическом режиме

В автоматическом режиме машина пытается измерить указанную точку, при этом машина движется из текущей позиции в точку назначения по кратчайшему пути (т.е. по прямой линии).

Очень важно задавать измерение точки таким образом, чтобы движение щупа происходило по нормали. Отклонение от нормали приведет к дополнительной погрешности.

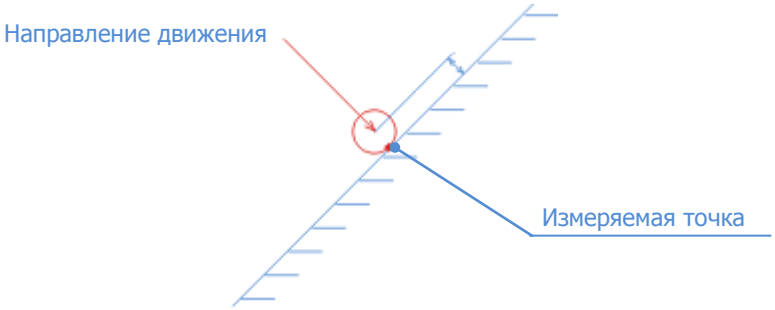


Рисунок 55. Иллюстрация цикла измерения точки

Цикл измерения состоит из следующих шагов: щуп движется по прямой линии до измеряемой точки, в процессе движения поступает сигнал касания, по которому фиксируются координаты, затем щуп возвращается в исходную позицию. Программа автоматически компенсирует размер наконечника и компенсирует погрешности датчика, если таковые присутствуют.

Для того, чтобы производить измерение координат точек, предназначена функция Cmm.MeasureTo(Vector3d), которая принимает на вход точку, к которой следует

двигаться, чтобы произвести измерение, и возвращает Vector3d – измеренную точку. Пример:

```
Vector3d point = new Vector3d(10, 10, 10);
Vector3d result = Cmm.MeasureTo(point);
```

Измерение точки в ручном режиме

Для измерения точки в ручном режиме предназначена функция Cmm.ManualMeasure(), которая возвращает измеренную точку:

Vector3d result = Cmm.ManualMeasure();

Функция работает следующим образом: машина управляется пользователем в ручном режиме: джойстиком или другим способом; при этом функция завершается, как только появляется сигнал о касании на датчике. Функция возвращает координаты касания.

Рисунок 56. Эмулятор джойстика

Измерение точек в ручном режиме не следует использовать для точных измерений, т.к. в этом режиме не происходит компенсации погрешностей щуповой системы машины.

Текущая позиция машины

Для того, чтобы узнать текущую позицию машины, следует использовать свойство `Cmm.Position`, которое возвращает `Vector3d` - текущую позицию машины в машинной системе координат.

Текущий размер наконечника

Для того чтобы, узнать радиус текущего установленного щупа, следует использовать свойство `Cmm.Radius`, которое возвращает скаляр - радиус. Для получения радиуса произвольного щупа входящего в состав текущей щуповой системы можно воспользоваться функцией `Cmm.GetRadius(index)`, в которую передать номер щупа, радиус которого необходимо получить.

Изменение текущего щупа

В случае если щуповая система имеет несколько щупов переключение между ними следует осуществлять с помощью функции `Cmm.SetCurrentTip(index)`, в которую необходимо передать индекс щупа, устанавливаемого текущим.

Функции перемещения и измерения точек осуществляются относительно текущего щупа. Это следует учитывать при построении маршрута измерения детали.

Система координат машины

Машина выполняет команды позиционирования и измерения в системе координат, которую можно получить или изменить используя свойство `Cmm.CoordinateSystem`:

```
CoordinateSystem system = new CoordinateSystem(origin, x, y, z);
Cmm.CoordinateSystem = system;
```

Подробнее см. «Математическое базирование»,ниже.

Математическое базирование

Для удобства построения маршрута измерения рекомендуется пользоваться созданием новых систем координат, привязанных к детали. Это позволит создавать маршрут измерения, который не будет зависеть от расположения детали на столе.

Обычно процесс заключается в получении нескольких точек в ручном режиме и расчете по ним системы координат. Если полученная точность математического базирования не удовлетворяет требованиям пользователя (это может произойти, так как измерение в ручном режиме является неточным), то можно измерить эти же точки в автоматическом режиме, чтобы уточнить математическое базирование.

В общем случае полученная погрешность при базировании влияет на точность измерения, т.к. для измерения в автоматическом режиме важным является движение по нормали к контролируемой поверхности

Для составления и хранения системы координат существует класс CoordinateSystem.

Чтобы составить систему координат необходимо иметь начало системы координат *origin* и три вектора, образующих базис *i, j* и *k*. Сконструировать систему координат имея все исходные данные можно, например, так:

```
CoordinateSystem system = new CoordinateSystem(origin, i, j, k);
```

Можно создать несколько систем координат. Чтобы применить одну из них к машине, необходимо присвоить к свойству Cmm.CoordinateSystem определенный объект, например, так:

```
CoordinateSystem system = new CoordinateSystem(origin, i, j, k);
Cmm.CoordinateSystem = system;
```

Существует другой способ, позволяющий менять отдельные компоненты, определяющие систему координат:

```
CoordinateSystem system = new CoordinateSystem();
system.Origin = Vector3d.Zero;
system.I = Vector3d.AxisX;
system.J = Vector3d.AxisY;
system.K = Vector3d.AxisZ;
```

Последнее дает возможность поменять уже установленную систему координат, например, изменить ее начало:

```
Cmm.CoordinateSystem.Origin = Vector3d.Zero;
```

При составлении системы координат необходимо учитывать направление осей машины. Все машины обязаны иметь направление осей как показано на рисунке ниже (см. **Рисунок 57**).

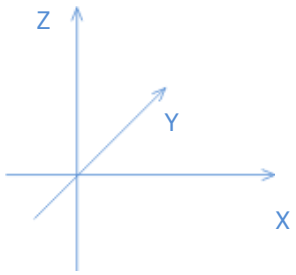


Рисунок 57. Правильное направление осей машины

Математическое базирование можно сохранять, однако надо понимать, что базирование останется актуальным, только если деталь не меняла своего местоположения. Для сохранения мат. базирования нужно использовать следующую конструкцию:

```
Cmm.CoordinateSystem.Save();
```

Для загрузки математического базирования необходимо использовать следующую операцию:

```
Cmm.CoordinateSystem.TryLoad();
```

Эта функция работает следующим образом: сначала происходит попытка загрузки сохраненного ранее базирования, если базирование ранее было сохранено, то пользователь увидит на экране диалог (см. **Рисунок 58**). В случае выбора «Загрузить базирование» операция вернет *true,* в другом случае будет возвращено *false*.

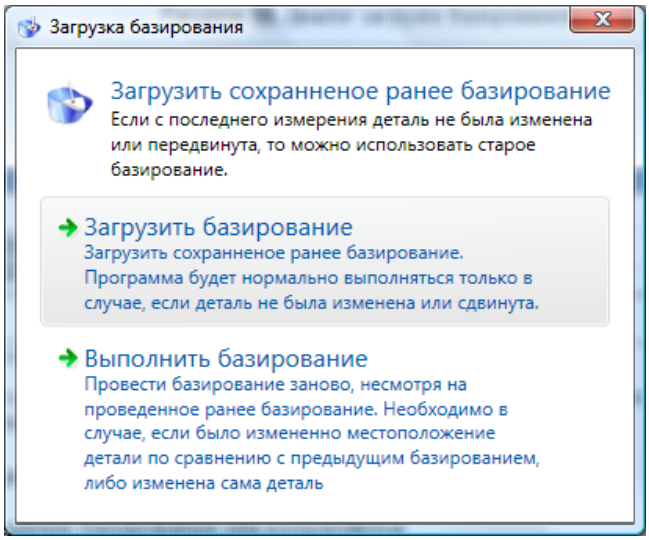


Рисунок 58. Диалог загрузки базирования

Рекомендуемая конструкция программы для загрузки базирования:

```
// Если функция загрузки возвращает – false, то производится
базирование
if (!Cmm.CoordinateSystem.TryLoad())
{
      // Далее делается базирование
      ...
   // После проведения базирования оно сохраняется
Cmm.CoordinateSystem.Save();
}
```

Необходимо заметить, что машина будет перемещаться отнюдь не в положительной области. Каждая машина имеет т.н. «куб безопасности», который задается специалистом по пуско-наладке.

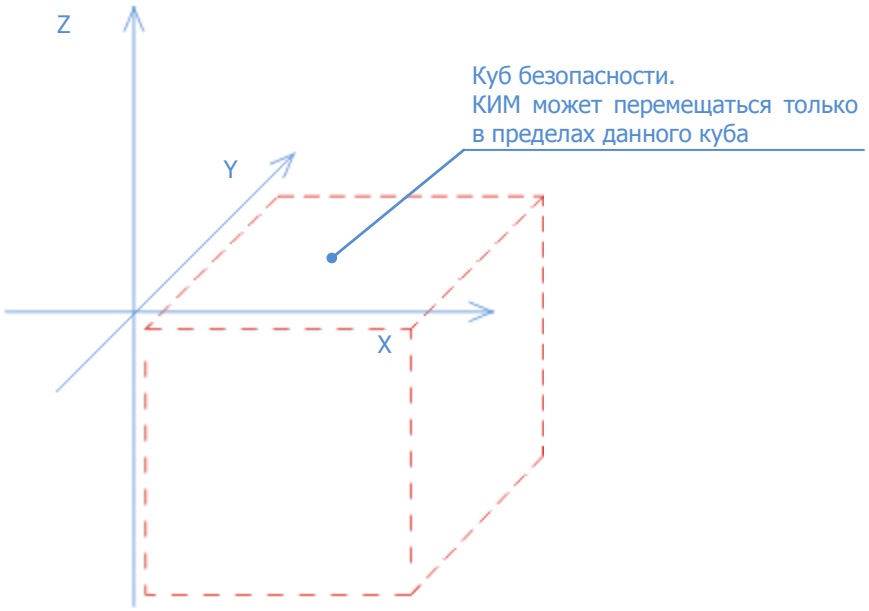


Рисунок 59. Куб безопасности КИМ

В машинах с фиксированным порталом (т.е. машины, где подвижным является стол, а не портал) считаем, что подвижным является щуп, а не стол.

Составление стратегии измерения

После того, как была получена система координат детали, можно программировать маршрут перемещения машины. Подробнее см. «Управление машиной»,на стр. 44.

Проверка схемы измерения

В любой момент времени разработки схемы измерения можно запустить симуляцию, нажав кнопку «Запуск симуляции» в главном меню. Это позволит проверить составляемую схему измерения, даже работая без реальной машины.

Подробнее о симуляции см. «Симуляция измерения», на стр. 40.

Для отладки написанного скрипта, в программе существует возможность пошагового запуска как на симуляторе, так и на реальной машине. Пошаговый режим заключается в том, что все операции перемещения и измерения (Cmm.MoveTo(…),Cmm.MeasureTo(…)) производятся только после подтверждения пользователем. После того, как программа дойдет до операции автоматического движения, на экране появится диалог (см. **Рисунок 60**).

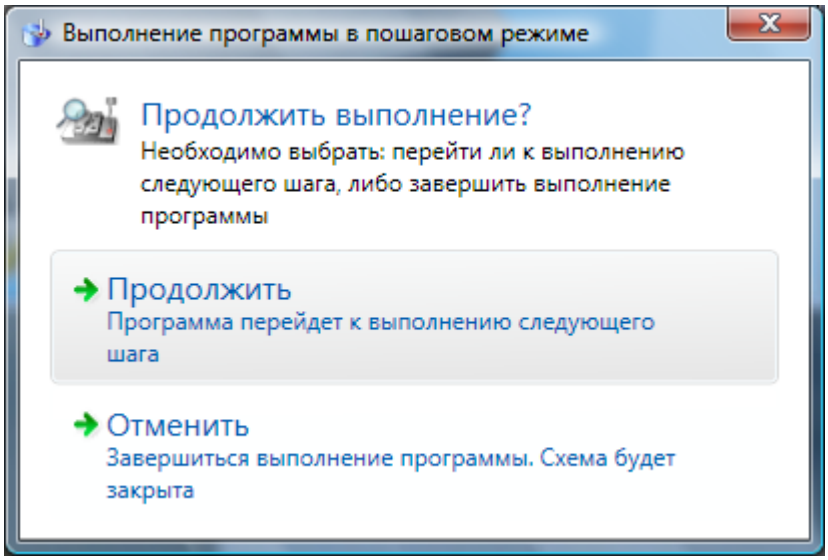


Рисунок 60. Диалог продолжения выполнения программы

Для включения пошагового режима необходимо нажать кнопку «Пошаговый режим». Повторное нажатие кнопки отменит пошаговый режим.

Расчет контролируемых параметров

Расчет контролируемых параметров осуществляется с использованием встроенных конструкций языка координатных измерений (см. главу «Язык для математической обработки данных координатного анализа»).

Чтобы рассчитать значение и передать его в отчет существует объект `Report`, который имеет метод `Add`. Например, так можно измерить сферу и передать ее радиус в отчет:

```
Vector3d[] points = new Vector3d[5];
for(int i = 0; i < points.Length; i++)
{
    // Измеряем точки в ручном режиме
    points[i] =  Cmm.ManualMeasure();
}
Sphere sphere = Average.Sphere(points);
Report.Add("Радиус сферы", sphere.Radius);
```

Подробно о редактировании отчетной формы и выводе результатов следует читать в главе «Координатный анализ геометрических параметров детали».

Симуляция измерения

Для отладки программы измерения детали предназначена специальная функция – симуляция. Симуляция позволяет разрабатывать и проверять схемы измерения без использования настоящей координатной машины, такой способ программирования КИМ называется offline-программированием.

Для запуска симуляции необходимо открыть для редактирования схему измерения и нажать в главном меню «Схема измерения → Запуск симуляции». Симулятор практически полностью повторяет поведение реальной координатно-измерительной машины, поэтому измерение на виртуальной машине сильно приближено к реальному измерению.

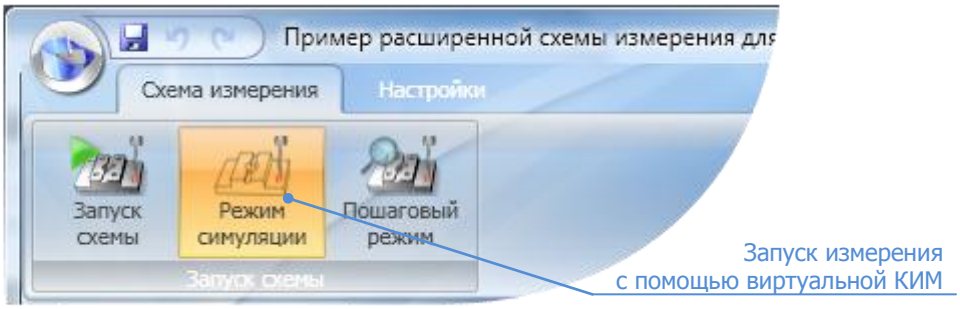


Рисунок 61. Запуск симуляции измерения

В начале работы с симулятором потребуется добавить модель детали. Для этого нажмите кнопку «Добавить деталь», выберите подходящую (деталь может быть загружена из файла в формате *.obj, *.stp, *.step, *.stl, *.cad).

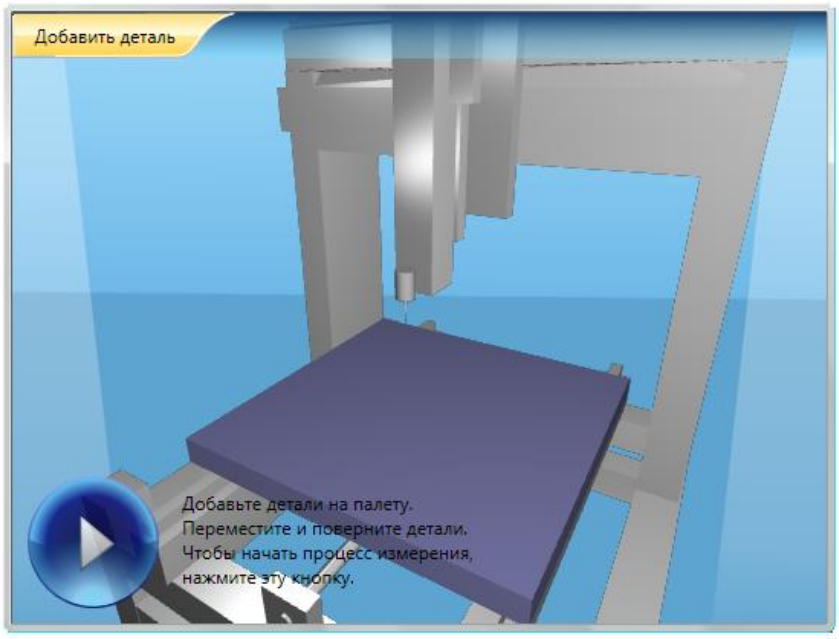


Рисунок 62. Подготовка к работе с симулятором

После этого нажмите кнопку в нижнем левом углу окна, чтобы продолжить процесс запуска схемы измерения.

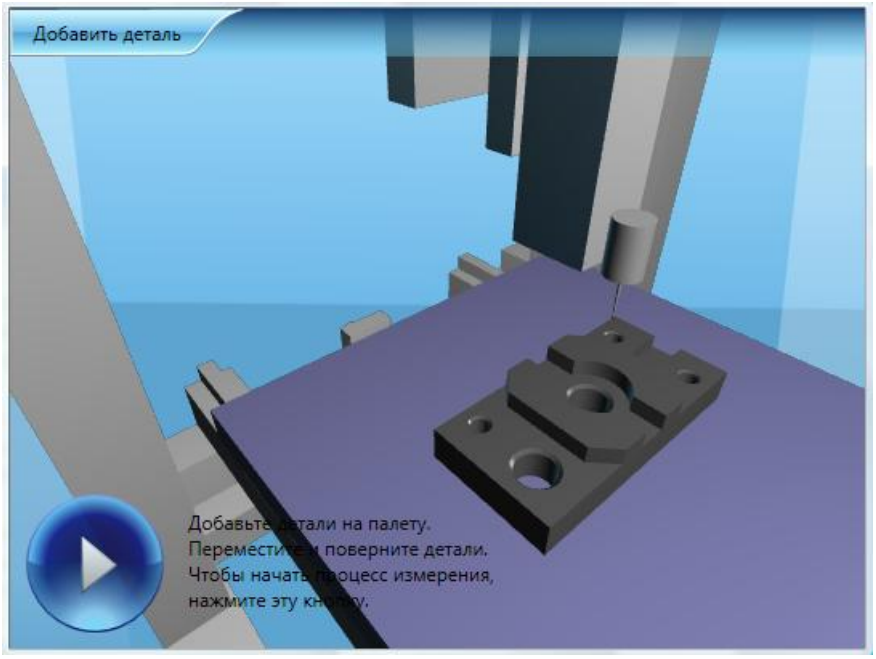


Рисунок 63. Деталь добавлена на стол

Дальнейшая работа с симулятором не отличается от работы с реальной машиной: необходимо выполнять запрограммированные действия, например, измерять точки в ручном режиме, используя джойстик (если джойстика нет, то на экране появится эмулятор (см. **Рисунок 64**)).

Рисунок 64. Эмулятор джойстика

Запуск схемы измерения

Для запуска схемы измерения на реальной машине следует нажать кнопку «Запуск схемы» (см. **Рисунок 65**). Машина произведет подключение, выход в ноль, затем начнут выполняться запрограммированные действия.

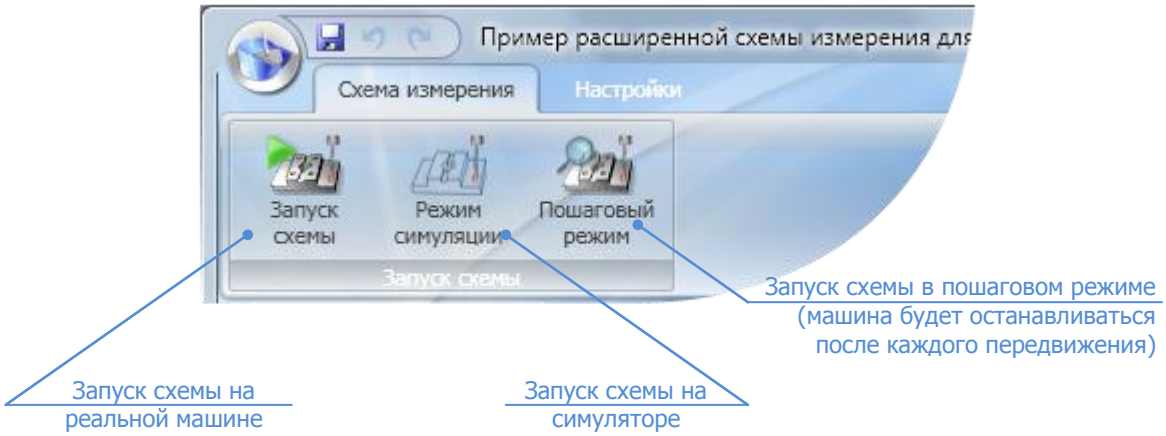


Рисунок 65. Главное меню схемы измерения

При необходимости можно просмотреть траекторию движения КИМ, нажав кнопку «Пошаговый режим», после чего при запуске программа будет останавливаться после каждого действия машины: перемещения, измерения точки.

Примеры программ

Измерение концевой меры длины[1]

Базирование в данном случае выполняется по трем плоскостям 3-2-1. В данном случае выбор направления осей сделан исходя из того как установлена деталь на столе, это накладывает ограничения на то, как должна устанавливаться деталь на стол. На мере измеряются 6 точек в ручном режиме для того, чтобы составить систему координат, затем измеряются 3 точки на одной и 1 точку на противоположной плоскости в автоматическом режиме, чтобы в качестве примера рассчитать размер меры длины.

Ниже приведен листинг программы:

```
// Идея метода: базирование детали осуществляется
// по 3 – м взаимно перпендикулярным плоскостям.
// Всего для осуществления данного типа базирования
// необходимо снять 6 точек: 3 точки на одной
// плоскости, 2 на второй и 1 точку на третьей плоскости.
// Создаем массив из 6-ти точек
Vector3d[] points = new Vector3d[6];
```

[1] Данный пример был написан и проверен для концевой меры длины размером 40 мм на машине Kosy + ЛИР940Р.

```
for(int i = 0; i < points.Length; i++)
{
    // Измеряем точки в ручном режиме, точки с
    // индексами 0..2 необходимо измерить на
    // первой плоскости, с индексами 3,4 на второй,
    // и точка с индексом 5 снимается с 3-ей плоскости
    if( i == 3 || i == 5)
    {Slides.Next();}
    points[i] =  Cmm.ManualMeasure();}
// Строим первую плоскость по 3-м точкам, нормаль
// для плоскости получаем как векторное поизведение
// 2- х векторов постоенных через 3 точки плоскости
Plane plane1 = new Plane(points[0], (points[1] -
      points[0]).CrossProduct(points[2] - points[0]));
// С учетом того что плоскости перпендикулярны, получаем
// недостоющую 3-ю точку для второй плоскости как проекцию
// точки второй плоскости на первую плоскость
Vector3d vectorProj21 = Projection.PointPlane(points[3], plane1);
// Нормализуем вектор, задающий нормаль плоскости
plane1.Normal.Normalize();
// По 3-м точкам получаем вторую плоскость
Plane plane2 = new Plane(points[3], (points[4] -
      points[3]).CrossProduct(vectorProj21 - points[3]));
// Нормализуем вектор, задающий нормаль плоскости
plane2.Normal.Normalize();
// Вторую точку третьей плоскости получаем
// как проекцию точки на первую плоскость
Vector3d vectorProj31 = Projection.PointPlane(points[5], plane1);
// Третью точку третьей плоскости получаем
// как проекцию точки на вторую плоскость
Vector3d vectorProj32 = Projection.PointPlane(points[5], plane2);
// По 3-м точкам получаем третью плоскость
Plane plane3 = new Plane(points[5], (vectorProj31 -
      points[5]).CrossProduct(vectorProj32 - points[5]));
// Нормализуем вектор, задающий нормаль плоскости
plane3.Normal.Normalize();
//  Получаем 3 линии путем пересечения попарно 3-х плоскостей
Line3d line1 = Intersection.PlanePlane(plane1,plane2);
Line3d line2 = Intersection.PlanePlane(plane1,plane3);
Line3d line3 = Intersection.PlanePlane(plane2,plane3);
// Для нахождения точки начала координат,
// ищем точку пересечения какой либо плоскости
// и прямой не лежащей в этой плоскости (!)
// (другие 2 полученные прямые будут принадлежать плоскости),
// берем, например, первую прямую и третью плоскость –
// получаем точку начала координат – общую точку 3-х плоскостей
Vector3d origin = Intersection.LinePlane(line1, plane3);
// Базисы получаем как направления, полученных раньше линий
// Замечание! В следствии пересечения плоскостей направления
// линий получают неоднозначное значения (возможны 2 варианта,
// отличающиеся друг от друга знаками), поэтому необходимо
// контролировать направление базисов, с учетом того какую
// систему координат мы хотим получить
Vector3d xBasis = line1.Direction;
// Данном случае мы задаем, что новые базисы по x и y
// имеют направления сходные с осями x и y машины,
// а базис по z направлен против оси z машины
if(xBasis.X < 0)
   xBasis = -xBasis;
Vector3d yBasis = line2.Direction;
if(yBasis.Y < 0)
   yBasis = -yBasis;
```

```
Vector3d zBasis = line3.Direction;
if(zBasis.Z < 0)
   zBasis = -zBasis;
// Устанавливаем начало координат
Cmm.CoordinateSystem.Origin = origin;
// Устанавливаем базисные вектора
Cmm.CoordinateSystem.I = xBasis;
Cmm.CoordinateSystem.J = yBasis;
Cmm.CoordinateSystem.K = zBasis;
// Производим движения в полученной системе координат
// Измеряем одну плоскость (3 точки) и точку на другой.
// Рассчитываем расстояние от точки до плоскости
Cmm.MoveTo(new Vector3d(-5,-5,3));
Cmm.MoveTo(new Vector3d(5,-3,-5));
Vector3d[] pointsPl = new Vector3d[3];
pointsPl[0] = Cmm.MeasureTo(new Vector3d(5,0,-5));
Cmm.MoveTo(new Vector3d(15,-3,-3));
pointsPl[1] = Cmm.MeasureTo(new Vector3d(15,0,-3));
Cmm.MoveTo(new Vector3d(25,-3,-6));
pointsPl[2] = Cmm.MeasureTo(new Vector3d(25,0,-6));
Cmm.MoveTo(new Vector3d(25,-3,8));
Cmm.MoveTo(new Vector3d(25,50,8));
Cmm.MoveTo(new Vector3d(25,50,-3));
Vector3d point = Cmm.MeasureTo(new Vector3d(25,40,-3));
Plane plane = Average.Plane(pointsPl);
Number distance = Distance.PointPlane(point, plane);
Report.Add("Размер концевой меры",distance);
```

Измерение корпусной детали

В качестве более комплексного измерения приведен пример измерения корпусной детали показанной на рисунке ниже (см **Рисунок 66**). Требуется произвести контроль следующих параметров:

- радиус левого цилиндра ∅17;
- расстояние от левого цилиндра ∅17 до ближней плоскости;
- отклонение от перпендикулярности оси левого цилиндра ∅17 относительно плоскости над ним;
- радиус центрального цилиндра ∅17;
- радиус центрального цилиндра R16,50;
- отклонение от концентричности двух центральных цилиндров в верхней плоскости.

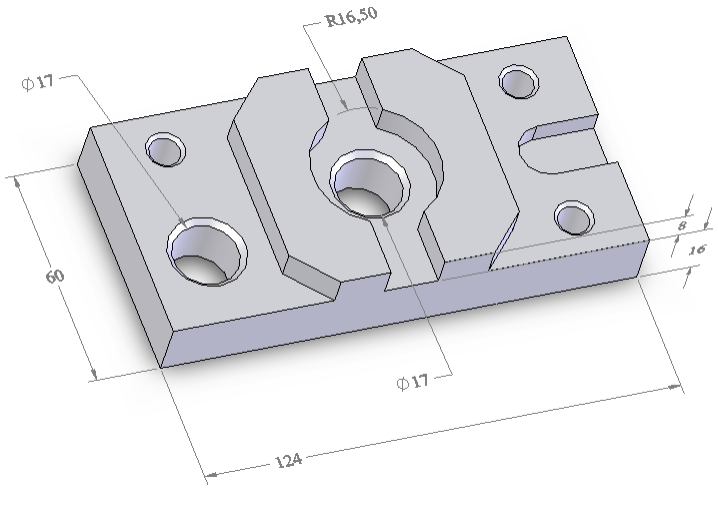


Рисунок 66. Измеряемая корпусная деталь

Математическое базирование

Математическое базирование в данном случае совпадает с базированием концевой меры длины (см. «Измерение концевой меры длины», на стр. 53).

```
Vector3d[] points = new Vector3d[6];
for(int i = 0; i < points.Length; i++)
{
    // Измеряем точки в ручном режиме, точки с индексами
    // 0..2 необходимо измерить на первой плоскости,
    // с индексами 3,4 на второй, и точка с индексом
    // 5 снимается с 3-ей плоскости
    points[i] =  Cmm.ManualMeasure();
    // Устанавливаем следующий слайд
    Slides.Next();}
// Строим первую плоскость по 3-м точкам, нормаль для
// плоскости получаем как векторное поизведение
// 2- х векторов постоенных через 3 точки плоскости
Plane plane1 = new Plane(points[0], (points[1] -
      points[0]).CrossProduct(points[2] - points[0]));
// С учетом того что плоскости перпендикулярны,
// получаем недостоющую 3-ю точку для второй плоскости
// как проекцию точки второй плоскости на первую плоскость
Vector3d vectorProj21 = Projection.PointPlane(points[3], plane1);
// Нормализуем вектор, задающий нормаль плоскости
plane1.Normal.Normalize();
// По 3-м точкам получаем вторую плоскость
Plane plane2 = new Plane(points[3], (points[4] -
      points[3]).CrossProduct(vectorProj21 - points[3]));
// Нормализуем вектор, задающий нормаль плоскости
plane2.Normal.Normalize();
// Вторую точку третьей плоскости получаем
// как проекцию точки на первую плоскость
Vector3d vectorProj31 = Projection.PointPlane(points[5], plane1);
// Третью точку третьей плоскости получаем
// как проекцию точки на вторую плоскость
Vector3d vectorProj32 = Projection.PointPlane(points[5], plane2);
// По 3-м точкам получаем третью плоскость
Plane plane3 = new Plane(points[5], (vectorProj31 -
      points[5]).CrossProduct(vectorProj32 - points[5]));
// Нормализуем вектор, задающий нормаль плоскости
plane3.Normal.Normalize();
//  Получаем 3 линии путем пересечения попарно 3-х плоскостей
Line3d line1 = Intersection.PlanePlane(plane1,plane2);
Line3d line2 = Intersection.PlanePlane(plane1,plane3);
Line3d line3 = Intersection.PlanePlane(plane2,plane3);
// Для нахождения точки начала координат,
// ищем точку пересечения какой либо плоскости
// и прямой не лежащей в этой плоскости!!
// (другие 2 полученные прямые будут принадлежать плоскости),
// берем, например, первую прямую и третью плоскость –
// получаем точку начала координат – общую точку 3-х плоскостей
Vector3d origin = Intersection.LinePlane(line1, plane3);
// Базисы получаем как направления, полученных раньше линий
// Замечание!!! В следствии пересечения плоскостей направления
// линий получают неоднозначное значения (возможны 2 варианта,
// отличающиеся друг от друга знаками), поэтому необходимо
// контролировать направление базисов, с учетом того какую
// систему координат мы хотим получить
Vector3d xBasis = line1.Direction;
```

```
// Данном случае мы задаем, что новые базисы по x и y
// имеют направления сходные с осями x и y машины,
// а базис по z направлен против оси z машины
if(xBasis.X < 0)
   xBasis = -xBasis;
Vector3d yBasis = line2.Direction;
if(yBasis.Y < 0)
   yBasis = -yBasis;
Vector3d zBasis = line3.Direction;
if(zBasis.Z < 0)
   zBasis = -zBasis;
// Устанавливаем начало координат
Cmm.CoordinateSystem.Origin = origin;
// Устанавливаем базисные вектора
Cmm.CoordinateSystem.I = xBasis;
Cmm.CoordinateSystem.J = yBasis;
Cmm.CoordinateSystem.K = zBasis;
```

Программирование маршрута

```
// Выходим в первую точку маршрута
Cmm.MoveTo(new Vector3d(-5,-5,3));
// Измеряем первую плоскость: для этого измеряем на ней 3 точки
Vector3d[] pointsPlane1 = new Vector3d[3];
// Перемещаемся для измерения первой точки
Cmm.MoveTo(new Vector3d(5,-4,-3));
// Измеряем первую точку
pointsPlane1[0] = Cmm.MeasureTo(new Vector3d(5,0,-3));
// Перемещаемся для измерения второй точки
Cmm.MoveTo(new Vector3d(15,-4,-5));
// Измеряем вторую точку
pointsPlane1[1] = Cmm.MeasureTo(new Vector3d(15,0,-5));
// Перемещаемся для измерения третьей точки
Cmm.MoveTo(new Vector3d(25,-4,-8));
// Измеряем третью точку
pointsPlane1[2] = Cmm.MeasureTo(new Vector3d(25,0,-8));
// Объезжаем деталь, переходим к измерению второй плоскости
Cmm.MoveTo(new Vector3d(16,-4,6));
Slides.Next();
// Измеряем вторую плоскость:  для этого измеряем на ней 3 точки
Vector3d[] pointsPlane2 = new Vector3d[3];
// Перемещаемся для измерения первой точки
Cmm.MoveTo(new Vector3d(5,5,4));
// Измеряем первую точку
pointsPlane2[0] = Cmm.MeasureTo(new Vector3d(5,5,0));
// Перемещаемся для измерения второй точки
Cmm.MoveTo(new Vector3d(20,7,4));
// Измеряем вторую точку
pointsPlane2[1] = Cmm.MeasureTo(new Vector3d(20,7,0));
// Перемещаемся для измерения третьей точки
Cmm.MoveTo(new Vector3d(16,40,4));
// Измеряем третью точку
pointsPlane2[2] = Cmm.MeasureTo(new Vector3d(17,40,0));
// Устанавливаем следующий слайд
Slides.Next();
// Перемещаемся для измерения первого цилиндра
Cmm.MoveTo(new Vector3d(16,20,6));
Cmm.MoveTo(new Vector3d(16,20,-3));
// Измеряем первый цилиндр: для этого измеряем 6 точек
// по 3 точки в 2-х плоскостях (измерения точек
// производятся из центра цилиндра).
Vector3d[] pointsCylinder1 = new Vector3d[6];
```

```
Vector3d centerPointer = new Vector3d(16,20,0);
// Угол на который отличаются друг от друга снимаемые точки
Number angle = 60;
// Радиус второго цилиндра
Number radius = 17/2;
// Координата по z, на которой снимаются точки цилиндра
Number zCoord = -3;
for(int i = 0; i < 6; i++)
{
    // Рассчитываем угол для съема точки
    Number currentAngle = angle*i;
    if (i==3)
    {
        // После измерения 3 точек переходим в другую плоскость по z
        zCoord = -7;
        Cmm.MoveTo(new
Vector3d(centerPointer.X,centerPointer.Y,zCoord));
    }
    // Измеряем точку цилиндра при измерении перемещаемся
    // только в плоскости xy, z координата не меняется
    //   centerPointer.X   +   radius*Basic.Cos((currentAngle/180)   *
Basic.Pi)
    // - таким образом получаем координату x точки на поверхности
цилиндра
    //   centerPointer.Y   +   radius*Basic.Sin((currentAngle/180)   *
Basic.Pi)
    // - таким образом получаем координату x точки на поверхности
цилиндра
    Vector3d measurePoint = new Vector3d(centerPointer.X +
             radius*Basic.Cos((currentAngle/180) *
            Basic.Pi),centerPointer.Y +
             radius*Basic.Sin((currentAngle/180) *
             Basic.Pi),zCoord);
    pointsCylinder1[i] =  Cmm.MeasureTo(measurePoint);
}
// Выезжаем из цилиндра
Cmm.MoveTo(new Vector3d(16,20,15));
// Измеряем третью плоскость, для этого измеряем на ней 3 точки
// Устанавливаем следующий слайд
Slides.Next();
Vector3d[] pointsPlane3 = new Vector3d[3];
// Перемещаемся для измерения первой точки
Cmm.MoveTo(new Vector3d(40,40,12));
// Измеряем первую точку
pointsPlane3[0] = Cmm.MeasureTo(new Vector3d(40,40,8));
// Перемещаемся для измерения второй точки
Cmm.MoveTo(new Vector3d(44,20,12));
// Измеряем вторую точку
pointsPlane3[1] = Cmm.MeasureTo(new Vector3d(44,20,8));
// Перемещаемся для измерения третьей точки
Cmm.MoveTo(new Vector3d(85,30,12));
// Измеряем третью точку
pointsPlane3[2] = Cmm.MeasureTo(new Vector3d(85,30,8));
// Устанавливаем следующий слайд
Slides.Next();
// Измеряем второй цилиндр, для этого измеряем 6 точек по
// 3 точки в 2-х плоскостях, измерения точек производится
// из точек удаленных от поверхности цилиндра на radius/2
Vector3d[] pointsCylinder2 = new Vector3d[6];
Vector3d centerPointer2 = new Vector3d(62,30,15);
// Угол на который отличаются друг от друга снимаемые точки
angle = 60;
```

```
// Радиус второго цилиндра
radius = 17/2;
// Координата по z на которой снимаются точки цилиндра
zCoord = -3;
// Перемещаемся для измерения цилиндра
Cmm.MoveTo(centerPointer2);
Cmm.MoveTo(new Vector3d(centerPointer2.X, centerPointer2.Y, zCoord));
for(int i = 0; i < 6; i++)
{
    // Рассчитываем угол измерения точки
    Number currentAngle = angle*i;
    if (i==3)
    {
        // После измерения 3 точек переходим в другую плоскость по z
        zCoord = -7;
    }
    // Перемещаемся в точку удаленную от поверхности на расстояние
radius/2
    Cmm.MoveTo(new Vector3d(
        centerPointer2.X  +  radius/2*Basic.Cos((currentAngle/180)  *
Basic.Pi),
        centerPointer2.Y  +  radius/2*Basic.Sin((currentAngle/180)  *
Basic.Pi),
        zCoord));
    // Измеряем точку на поверхности цилиндра
    Vector3d measurePoint = new Vector3d(
        centerPointer2.X  +  radius*  Basic.Cos((currentAngle/180)  *
Basic.Pi),
        centerPointer2.Y  +  radius*  Basic.Sin((currentAngle/180)  *
Basic.Pi),
        zCoord);
    pointsCylinder2[i] =  Cmm.MeasureTo(measurePoint);
}
// Измеряем третий цилиндр, для этого измеряем 6 точек
// по 3 точки в 2-х плоскостях, измерения точек производится
// из точек удаленных от поверхности цилиндра на radius*0.25
Vector3d[] pointsCylinder3 = new Vector3d[6];
Vector3d centerPointer3 = new Vector3d(62,30,15);
angle = -30;
// Угол на который отличаются друг от друга снимаемые точки
Number stepAngle = 30;
// Радиус второго цилиндра
radius = 16.5;
// Координата по z, на которой снимаются точки цилиндра
zCoord = 6;
// Устанавливаем следующий слайд
Slides.Next();
Cmm.MoveTo(new Vector3d(centerPointer3.X, centerPointer3.Y, zCoord));
for(int i = 0; i < 6; i++)
{
    if (i==3)
    {
        zCoord = 3;
        // Так как 3-ий цилиндр имеет разрывы,
        // изменяем угол измерения точек чтобы не попасть в разрыв
цилиндра
        angle = 30;
    }
    // Расчитываем угол измерения точек
    Number currentAngle = angle + stepAngle*i;
    // Перемещаемся в точку отстоящую от поверхности цилиндра на
radius*0.25
```

```
        Cmm.MoveTo(new Vector3d(
            centerPointer3.X + (0.75*radius)*
            Basic.Cos((currentAngle/180) * Basic.Pi),
            centerPointer3.Y+ (0.75*radius)*
            Basic.Sin((currentAngle/180) * Basic.Pi), zCoord));
        // Измеряем точку на поверхности цилиндра
        Vector3d measurePoint = new Vector3d(
            centerPointer3.X + radius*
            Basic.Cos((currentAngle/180) * Basic.Pi),
            centerPointer3.Y + radius*
            Basic.Sin((currentAngle/180) * Basic.Pi), zCoord);
        pointsCylinder3[i] =  Cmm.MeasureTo(measurePoint);}
    // В конце выезжаем из цилиндра
    Cmm.MoveTo(centerPointer3);
```

Добавление информации для оператора

Чтобы оператору было понятно, какие действия от него ожидает разработчик схемы измерения, следует добавить ряд сопроводительных фотографий, которые помогут ему расположить правильно деталь на паллете, измерить правильные точки в ручном режиме и т.д.

Рисунок 67. Добавление фото с пояснениями для оператора

Расчет контролируемых параметров

```
// По измеренным точкам строим 3 плоскости
Plane planeMeasure1 = new Plane(pointsPlane1[0], (pointsPlane1[1] -
      pointsPlane1[0]).CrossProduct(pointsPlane1[2] -
pointsPlane1[0]));
Plane planeMeasure2 = new Plane(pointsPlane2[0], (pointsPlane2[1] -
      pointsPlane2[0]).CrossProduct(pointsPlane2[2] -
pointsPlane2[0]));
Plane planeMeasure3 = new Plane(pointsPlane3[0], (pointsPlane3[1] -
      pointsPlane3[0]).CrossProduct(pointsPlane3[2] -
pointsPlane3[0]));

// Аппроксимируем 3 цилиндра по измеренным точкам
Cylinder cylinder1 = Average.Cylinder(pointsCylinder1,
Vector3d.ZAxis);
Cylinder cylinder2 = Average.Cylinder(pointsCylinder2,
Vector3d.ZAxis);
Cylinder cylinder3 = Average.Cylinder(pointsCylinder3,
Vector3d.ZAxis);

// Находим точки пересечения осей 2-го и
// 3-го цилиндра с третьей плоскостью
Vector3d point1 = Intersection.LinePlane(cylinder2.Axis,
planeMeasure3);
Vector3d point2 = Intersection.LinePlane(cylinder3.Axis,
planeMeasure3);
// Вычисляем отклонение от концентричности 2-х
// цилиндров как расстояние между полученными точками
Report.Add("Отклонение от
концентричности",Distance.PointPoint(point1,point2));
// Вычисляем перпендикулярность между осью первого
// цилиндра и второй плоскостью на 100 миллиметрах
Report.Add("Перпендикулярность",
       Perpendicularity.LinePlane(cylinder1.Axis,planeMeasure2,100));
// Заносим в отчет радиусы цилиндров
Report.Add("Радиус цилиндра 1",cylinder1.Radius);
Report.Add("Радиус цилиндра 2",cylinder2.Radius);
Report.Add("Радиус цилиндра 3",cylinder3.Radius);
```

Настройка отчета

Так может быть настроен отчет по составленной в данном примере программе (см. **Рисунок 68**). Как настроить отчет см. подробнее в главе «Координатный анализ геометрических параметров детали»

Рассчитанные параметры становятся доступными в отчете только после реального запуска или симуляции

Эскиз

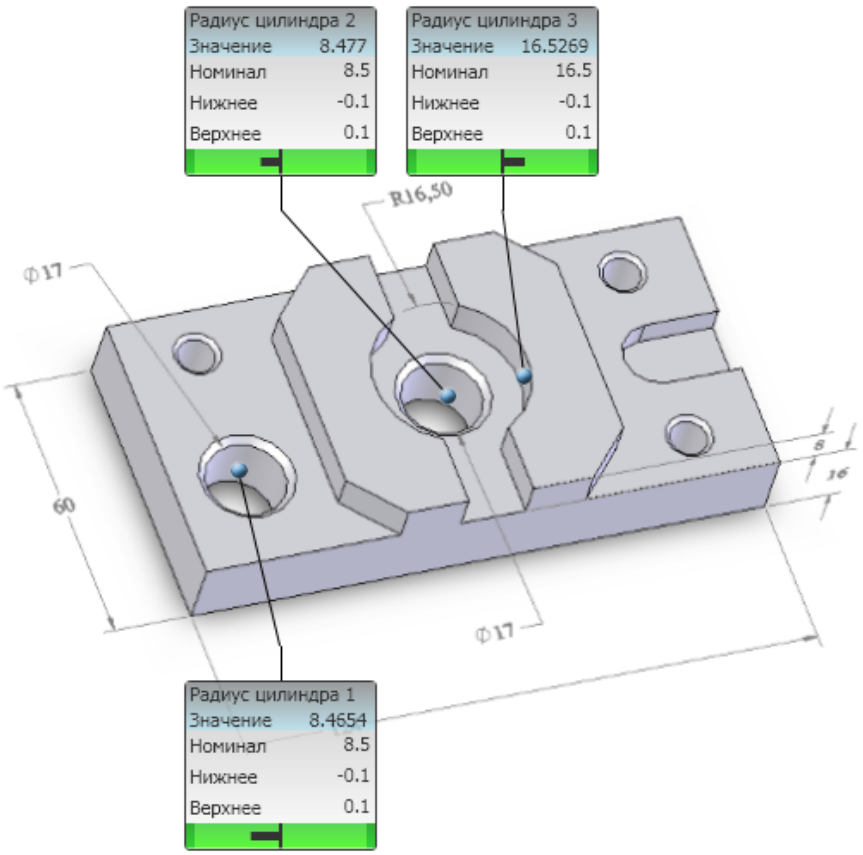


Таблица параметров

Параметр	Значение	Номинал	Нижнее	Верхнее	Отклонение	Вне допуска	Отклонение
Отклонение от концентричности	0.7551	0	-1	1	0.7551		
Перпендикулярность	0.9005	0	-1	1	0.9005		
Радиус цилиндра 1	8.4654	8.5	-0.1	0.1	-0.0346		
Радиус цилиндра 2	8.477	8.5	-0.1	0.1	-0.023		
Радиус цилиндра 3	16.5269	16.50	-0.1	0.1	0.0269		

Рисунок 68. Отчет об измерении корпусной детали

Примеры функций

Функция измерения цилиндра

Функция принимает следующие параметры:

`startPoint` – стартовая точка цилиндра, точка, расположенная на оси цилиндра;
`radiusCylinder` – номинал радиуса измеряемого цилиндра;
`lengthCylinder` – длина измеряемого цилиндра;
`numberOfPoint` – количество точек измеряемых на цилиндре, должно быть не меньше 6;

Функция производит измерение точек цилиндра по спирали. В результате выполнения функции возвращается аппроксимированный по точкам цилиндр.

```
function Cylinder MeasureCylinder(Vector3d startPoint, double
radiusCylinder, double lengthCylinder, int numberOfPoint)
{
    Vector3d[] pointsCylinder = new Vector3d[numberOfPoint];
    // Угол на который отличаются друг от друга снимаемые точки
    Number angleTurn = 50;
    if (numberOfPoint < 7)
    {
        angleTurn = 360/ numberOfPoint;
    }
    Cmm.MoveTo(startPoint);

    // Координата по z на которой снимаются точки цилиндра
    Number startDistance = -lengthCylinder/10;
    Number stepLength = (lengthCylinder + startDistance)/numberOfPoint;
    for(int i = 0; i < numberOfPoint; i++)
    {

        // Рассчитываем угол для съема точки
        Number currentAngle = angleTurn*i;
Cmm.MoveTo(startPoint + new
Vector3d((radiusCylinder/2)*Basic.Cos((currentAngle/180) *
Basic.Pi),(radiusCylinder/2)*Basic.Sin((currentAngle/180) *
Basic.Pi),startDistance - i*stepLength));
Vector3d measurePoint = startPoint + new
Vector3d(radiusCylinder*Basic.Cos((currentAngle/180) *
Basic.Pi),radiusCylinder*Basic.Sin((currentAngle/180) *
Basic.Pi),startDistance - i*stepLength);
        pointsCylinder[i] =  Cmm.MeasureTo(measurePoint);
    }

    Cylinder cylinder = Average.Cylinder(pointsCylinder,Vector3d.ZAxis);
    Cmm.MoveTo(startPoint);
    return cylinder;
      }
```

Пример вызова функции:

```
Cylinder cylinder1 = MeasureCylinder(new Vector3d(16, 20, 0), 8.5,
10.0, 15);
```

Реверсивная контактная схема измерения

Назначение и преимущества

Реверсивная схема измерения является высокоинтерактивным инструментом для измерения различных деталей с применением различных типов координатно-измерительных машин. Благодаря высокой интеллектуальности данной схемы измерения от пользователя требуются следующие знания и умения:

- базовые навыки работы с операционной системой Microsoft Windows™.
- знание основных принципов координатных измерений;
- общие знания в области аналитической геометрии;

Данный тип схемы измерения позволяет:

- измерять отдельные элементы детали без предварительно подготовленной модели;
- проводить измерение в автоматическом цикле после первого ручного измерения;
- производить привязку системы координат машины к системе координат модели;
- передавать данные для последующих расчетов в мощный инструмент «координатный анализ» (см. «Координатный анализ геометрических параметров детали»);
- визуализировать движения щупа относительно детали;
- производить настройку отчетной формы и выводить ее на печать.

Разработанная схема измерения может быть запущена на любой поддерживаемой координатно-измерительной машине.

Структура редактора

Редактор схемы измерения состоит из окна, включающего главное меню, 3D-просмотр и интегрированный координатный анализ. Круглая кнопка в верхнем левом углу окна позволяет работать с файлом схемы измерения: сохранить, открыть, создать, отправить на печать и т.д. (см. **Рисунок 69**).

Редактор сохраняет и открывает файлы с расширением *.reversescheme

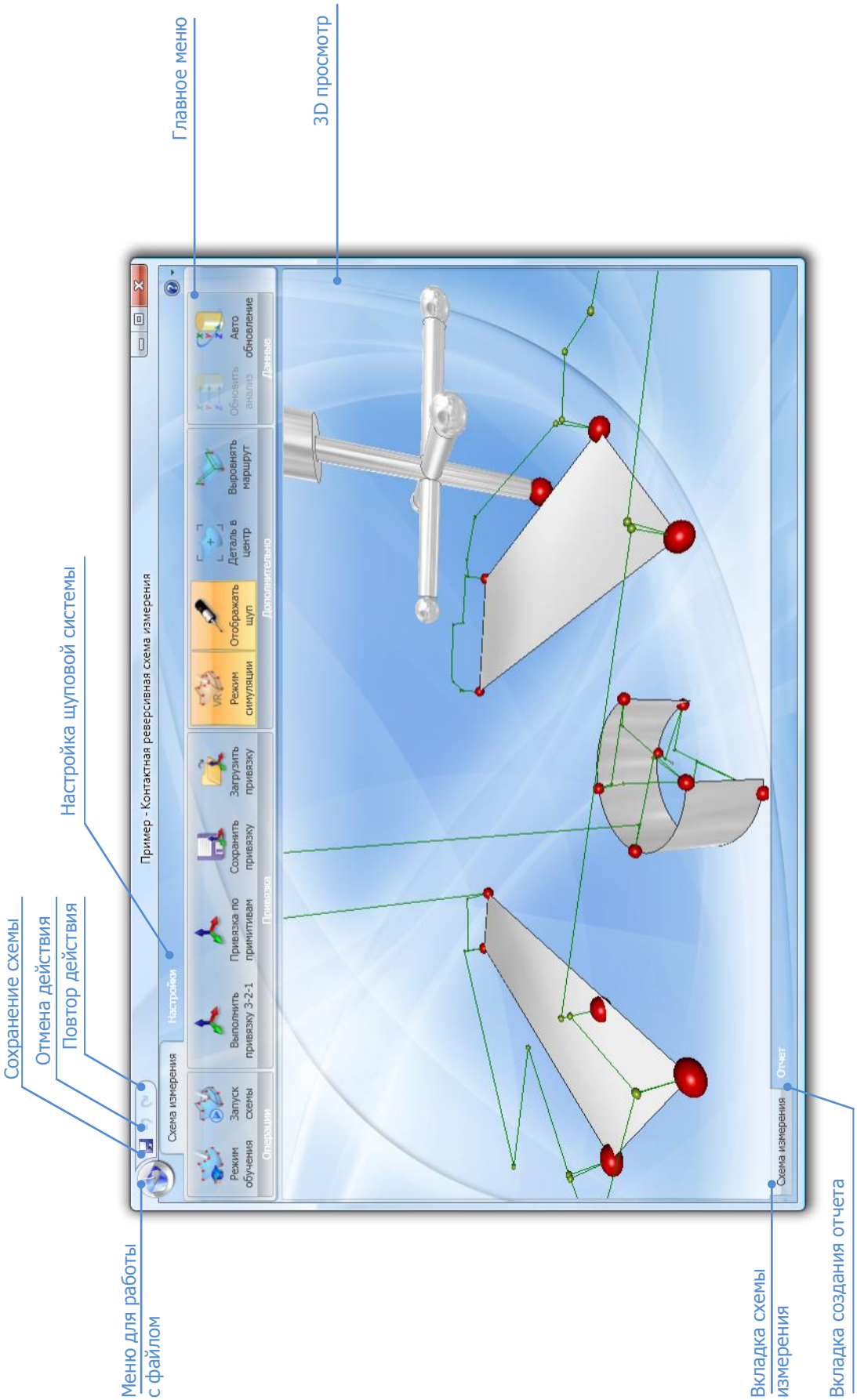


Рисунок 69. Структура редактора

Рекомендуемый порядок работ

Данная схема измерения предоставляет возможности по управлению координатно-измерительной машиной. Управление машиной программируется путем ее «обучения» в ручном режиме.

Задача схемы измерения – собрать облако точек с реальной детали и передать их для обработки в специальный редактор «Координатный анализ», с помощью которого можно рассчитать различные параметры.

Работа с редактором начинается с измерения нескольких элементов в ручном режиме. Пользователь в режиме обучения (см. «Режим обучения») измеряет несколько точек на каждом элементе, при этом программа восстанавливает по ним геометрический элемент.

После ручного измерения элементов доступна восстановленная по измеренным точкам модель детали, которая вместе с измеренными точками доступна в координатном анализе для различных расчетов. Уже на данном этапе можно перейти на вкладку «Отчет» чтобы рассчитать необходимые параметры.

Обратите внимание, что при добавлении нового измеренного элемента, удалении, измерении изменения не отображаются в анализе. Для того чтобы изменения предать в координатный анализ необходимо нажать кнопку «Обновить анализ». Существует возможность автоматического обновления координатного анализа при любых изменениях, для этого необходимо «вдавить» кнопку «Авто обновление» в главном меню, однако в случае большого количества элементов или низкопроизводительного компьютера это может вызывать неудобства из-за снижения скорости работы программы.

Как правило, точность измерения в ручном режиме оставляет желать лучшего. Это связано в первую очередь с тем, что

- скорость движения щупа при измерении точки отличается от откалиброванной номинальной скорости;
- траектория движения к поверхности не направлена по нормали;

Чтобы получить более точное измерение необходимо выполнить измерение в автоматическом режиме (см. «Запуск измерения»). Предварительно также рекомендуется выполнить выравнивание маршрута, чтобы измерение точки проходило по нормали к измеряемой поверхности, для этого необходимо нажать кнопку «Выровнять маршрут» в главном меню.

Полученную схему измерения можно сохранить, а затем использовать для измерения такой же детали. Поскольку деталь будет расположена в другом месте на столе машины, необходимо выполнить привязку (см. «Привязка CAD-модели»).

Режим обучения

Режим обучения предназначен для добавления измеренных поверхностей в ручном режиме. Для того чтобы войти в режим обучения следует нажать кнопку «Режим обучения» (см. **Рисунок 70**) и измерять точки на поверхности элемента.

С момента включения режима обучения начинается запись перемещений щупа. Если пользователь не вывел щуповую систему в желаемое исходное положение заранее, то в режиме обучения после помещения щупа в желаемое начальное положение следует нажать на кнопку сброса (в виде корзины).

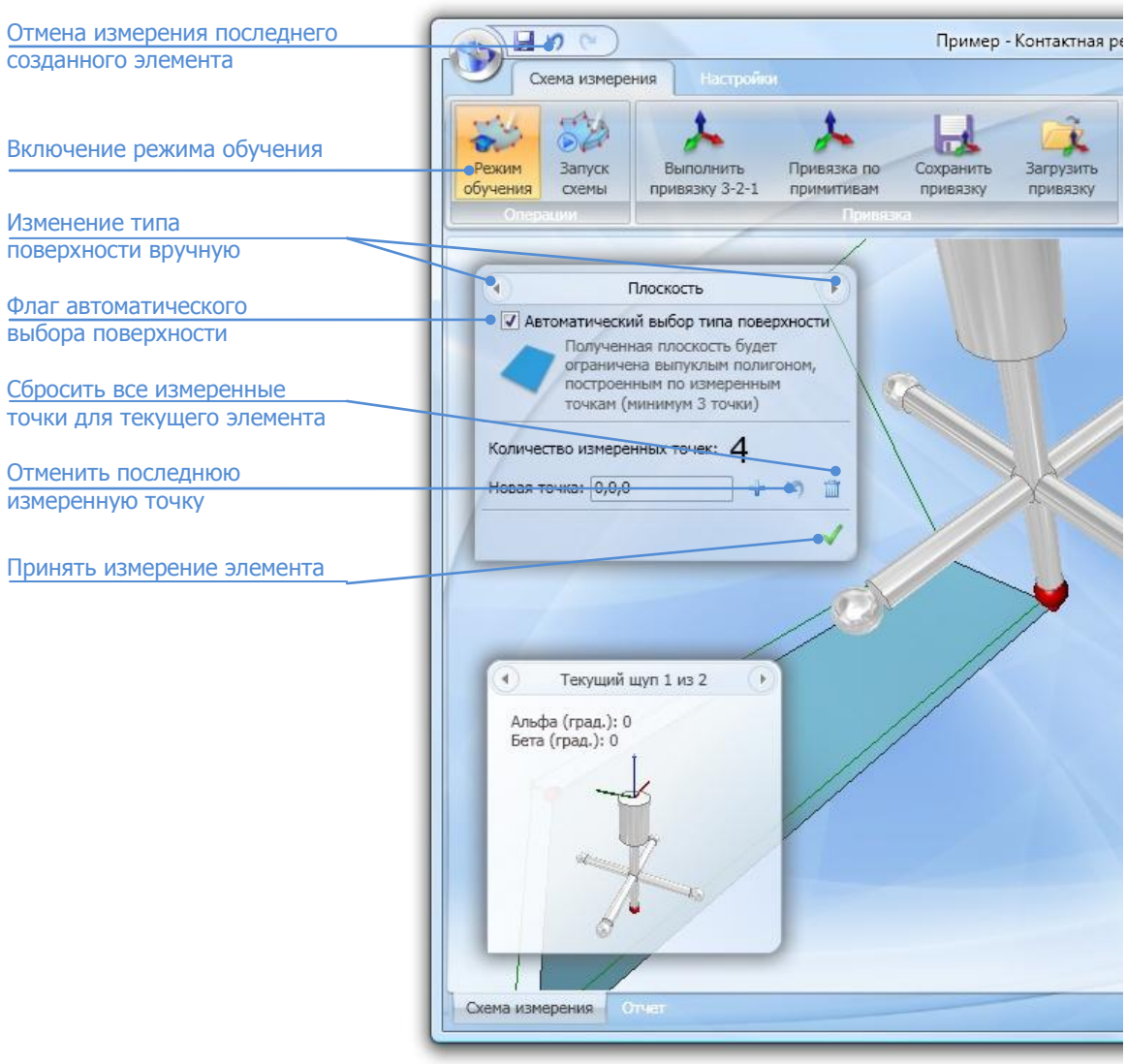


Рисунок 70. Режим обучения

В процессе построения элемента в окне отображается маршрут и восстановленная по точкам поверхность. Тип поверхности можно указать вручную или установить флаг «Автоматический выбор поверхности» чтобы программа подбирала наиболее подходящий к данным точкам тип поверхности.

Неверно измеренную точку можно отменить, используя кнопку отмены последней измеренной точки (см. **Рисунок 70**). Если точек достаточно можно нажать на кнопку

принятия элемента, чтобы создать его – программа создает новый элемент и автоматически переходит к записи маршрута следующего элемента.

Созданный элемент можно удалить. Для этого следует выйти из режима обучения, выделить элемент и нажать «Delete» или выбрать в контекстном меню «Удалить». Чтобы удалить последний созданный элемент, можно нажать кнопку отмены измерения последнего созданного элемента (см. **Рисунок 70**), не выходя из режима обучения.

Замечания относительно измеряемых элементов:

- **Плоскость**. Для измерения необходимо от **трех** различных точек расположенных не на одной прямой. Восстановленный элемент – плоскость, ограниченная выпуклым полигоном, построенным по измеренным точкам (см. **Рисунок 71**).

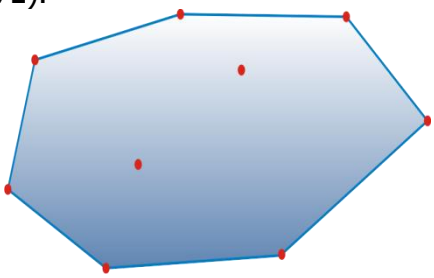


Рисунок 71. Плоскость

- **Сфера**. Для измерения необходимо от **пяти** различных точек расположенных не в одной плоскости. Восстановленный элемент – сектор сферы, включающий в себя измеренные точки (см. **Рисунок 72**).

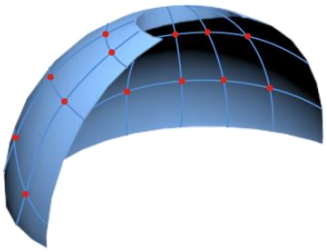


Рисунок 72. Сфера

- **Цилиндр**. Для измерения необходимо от **шести** различных точек расположенных не в одной плоскости. Рекомендуется выбирать *минимум* четыре точки в одном сечении и две наиболее удаленно от выбранного сечения вдоль направляющей цилиндра.

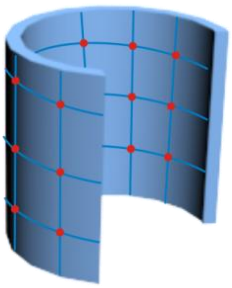


Рисунок 73. Цилиндр

Восстановленный элемент – часть цилиндра включающая в себя измеренные точки (см. **Рисунок 73**).

- **Конус**. Для измерения необходимо от **семи** различных точек расположенных не в одной плоскости. Рекомендуется выбирать *минимум* четыре точки в одном три в другом наиболее удаленно от первого сечения. Восстановленный элемент – часть конуса включающая в себя измеренные точки (см. **Рисунок 74**).

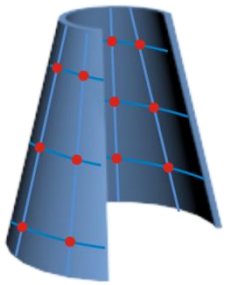


Рисунок 74. Конус

- **Тор**. Для измерения необходимо от **восьми** различных точек расположенных не в одной плоскости. Рекомендуется выбирать *минимум* четыре точки в продольном сечении, четыре в поперечном и две произвольно вне этих сечений. Восстановленный элемент – часть конуса включающая в себя измеренные точки (см. **Рисунок 75**).

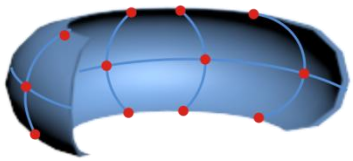


Рисунок 75. Тор

- **Сплайновая поверхность**. Для измерения необходимо от трех различных точек. Точки рекомендуется выбирать равномерно сеткой. Метод работает только для поверхностей геометрически близких к плоскости (более точное ограничение формулируется так: максимальный угол между любыми двумя нормалями должен быть меньше чем 180°) (см. **Рисунок 76**).

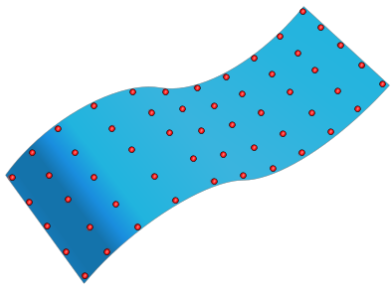


Рисунок 76. Сплайновая поверхность

Запуск измерения

Автоматическое измерение дает более высокую точность, т.к. в случае ручного измерения

- скорость движения щупа при измерении точки отличается от откалиброванной номинальной скорости;
- траектория движения к поверхности не направлена по нормали.

Предварительно рекомендуется выполнить выравнивание маршрута, чтобы измерение точки проходило по нормали к измеряемой поверхности, для этого необходимо нажать кнопку «Выровнять маршрут» в главном меню.

Для запуска схемы на реальной машине следует убедиться, что кнопка «Режим симуляции» неактивна, и нажать кнопку «Запуск схемы» (см. **Рисунок 77**). Машина произведет подключение, выход в ноль, затем начнут выполняться запрограммированные действия.

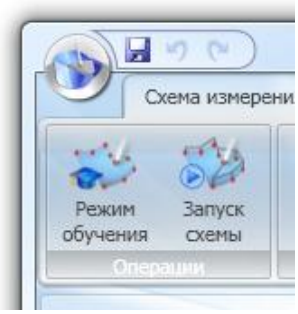


Рисунок 77. Главное меню схемы измерения

Очень важно, что измерения необходимо выполнять той же щуповой системой, которая использовалась в режиме обучения.

Существует возможность запустить измерение только одного элемента, для этого следует правой клавишей мыши вызвать контекстное меню данного элемента и выбрать «Измерить» (см. **Рисунок 78**).

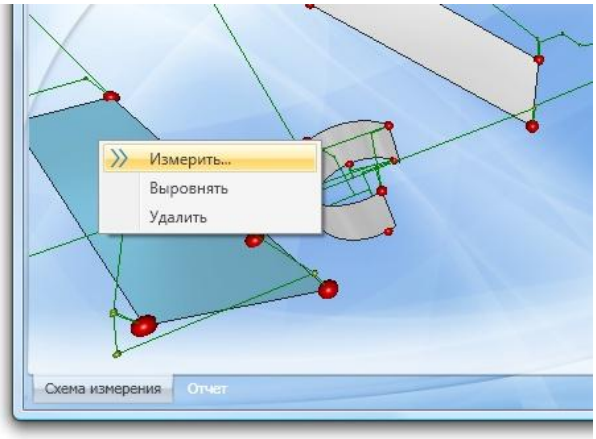


Рисунок 78. Запуск измерения одного элемента

Чтобы измерить несколько элементов, можно выделить элементы с зажатой клавишей Ctrl, вызвать контекстное меню и выбрать «Измерить» (см. **Рисунок 48**).

В процессе измерения необходимо вывести машину в положение, из которого она сможет беспрепятственно подъехать и измерить очередной элемент. Для этого следует в ручном режиме вывести щуп в нужную позицию (на экране в этот момент будет отображаться предполагаемый маршрут) и нажать на кнопку «продолжить» см. **Рисунок 79**, после чего машина произведет измерение данного элемента в автоматическом режиме.

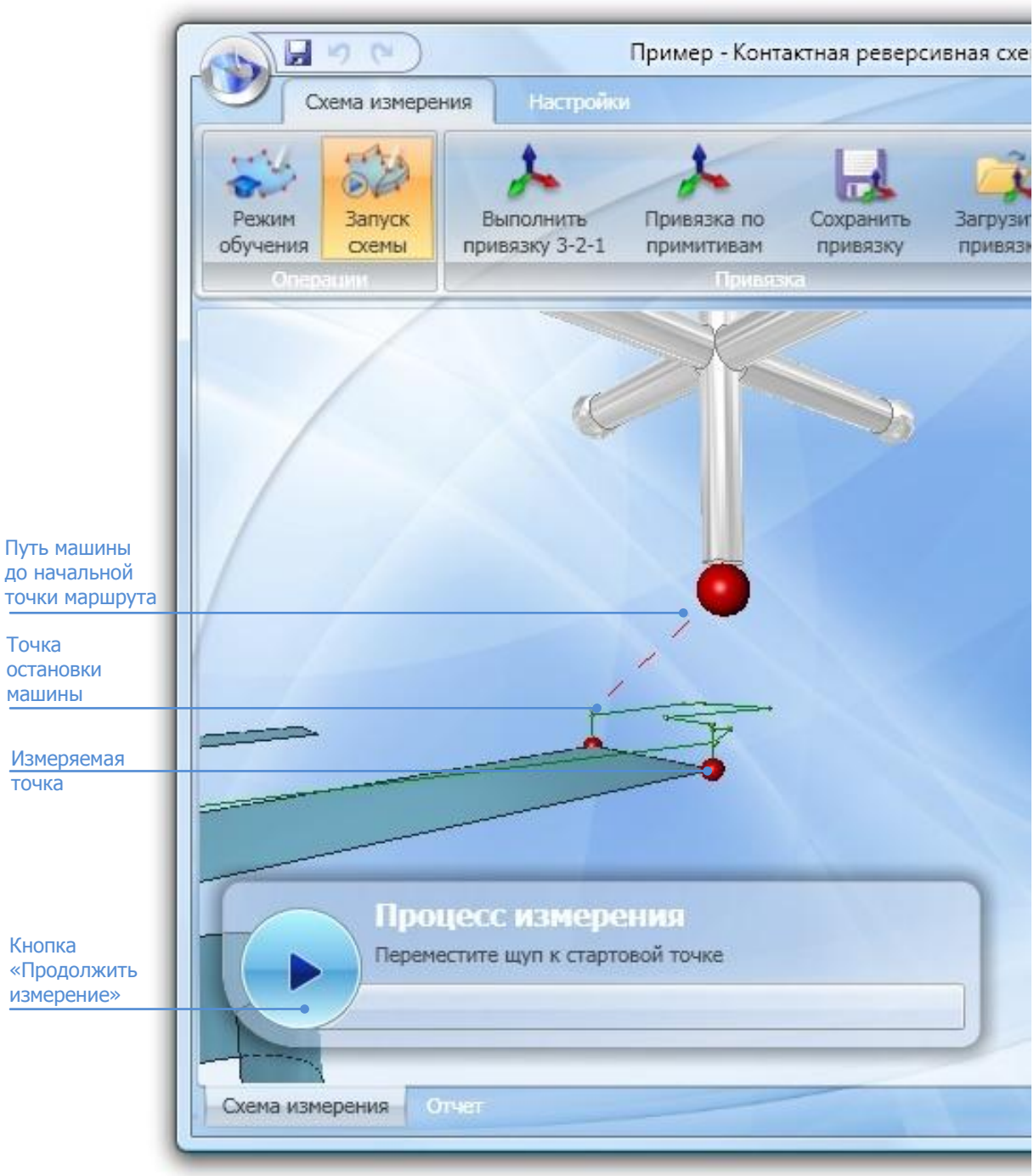


Рисунок 79. Ожидание перемещения к стартовой точке

Привязка CAD-модели

Для того чтобы производить перемещения машины вокруг детали, измерять точки на ее поверхности необходимо узнать как расположена деталь на машине (см. **Рисунок 80**). Такой процесс называется привязка текущего расположения детали к CAD-модели.

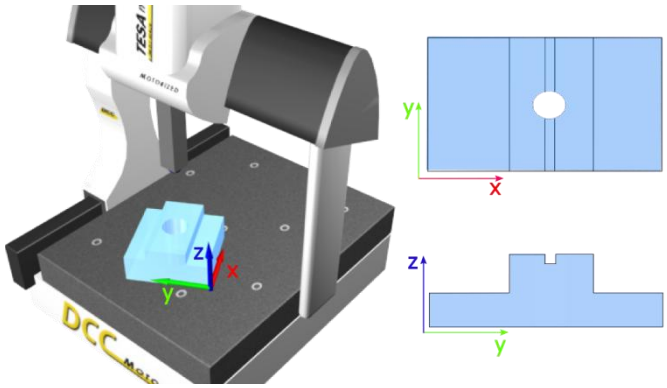


Рисунок 80. Привязка CAD-модели

Привязка производится в ручном режиме. Пользователю необходимо измерить несколько точек, расположенных на определенных поверхностях детали, после чего будет выполнен расчет перевода координат из системы координат модели в систему координат детали.

Привязка 3-2-1

Привязка 3-2-1 выполняется очень быстро. Для данной привязки требуется обязательное наличие трех непараллельных плоскостей.

Чтобы начать привязку следует нажать кнопку в главном меню «Выполнить привязку 3-2-1». Первым шагом мастера будет выбор трех непараллельных плоскостей. При выборе плоскостей рекомендуется руководствоваться следующими правилами:

- Выбирать плоскости расположенные перпендикулярно осям машины. Это связано с тем, что при измерении в ручном режиме важно производить измерение, двигаясь по нормали к поверхности детали, а проще всего это сделать, если поверхность перпендикулярна одной из осей.
- Рекомендуется выбирать плоскости с большой площадью.

После выбора плоскостей мастер по очереди будет предлагать измерить три, две и одну точку на соответствующих плоскостях, поочередно подсвечивая их. Рекомендуется выбирать наиболее удалённые друг от друга точки. При измерении точек необходимо менять щуп на тот, которым предполагается произвести касание (для этого можно воспользоваться элементом управления в нижнем левом углу, см. **Рисунок 81**).

Запрещается:

- первые три точки выбирать лежащие на одной прямой;
- две точки на второй плоскости измерять на прямой, перпендикулярной линии пересечения первой и второй плоскости.

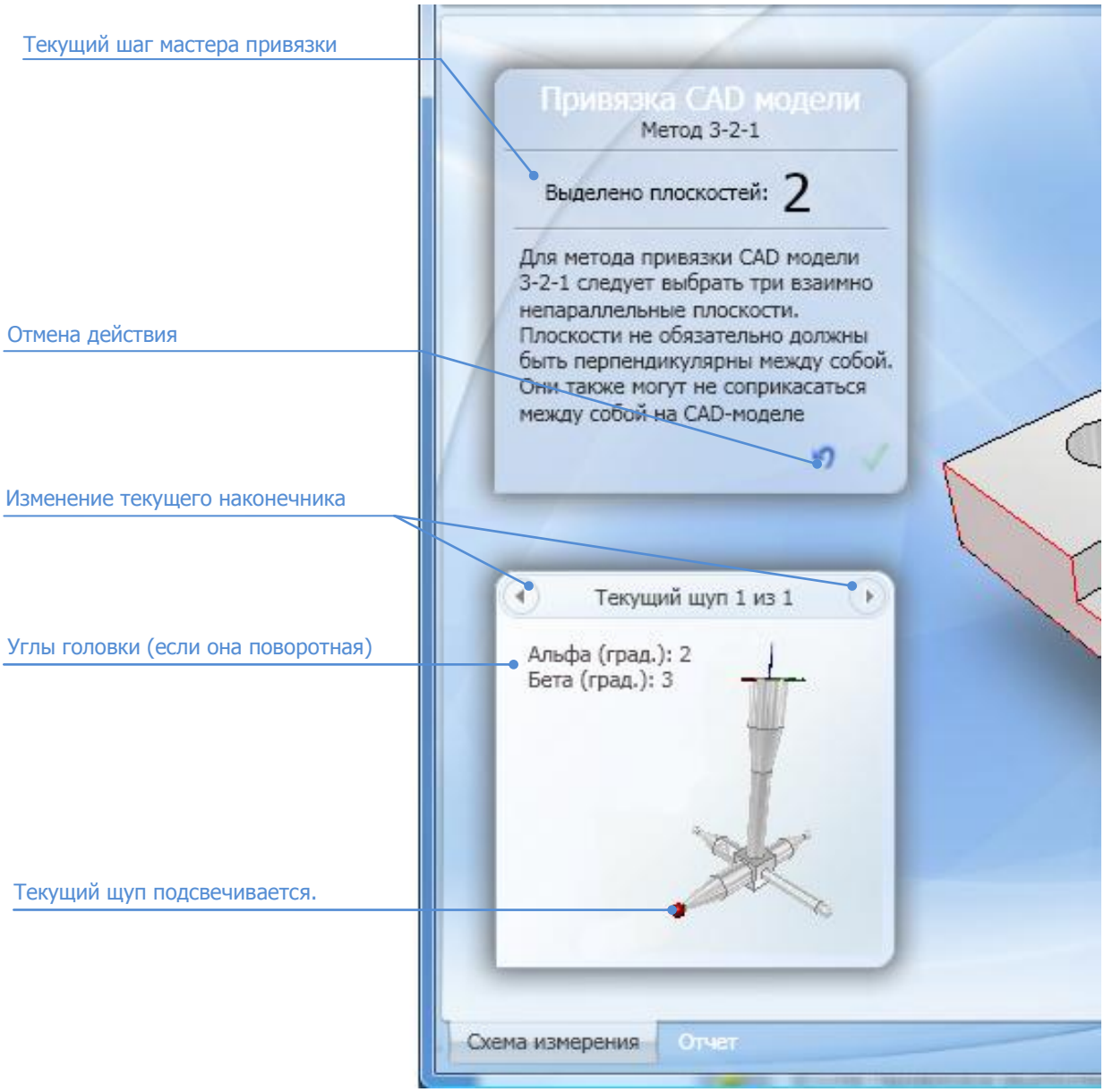


Рисунок 81. Привязка 3-2-1

Если привязка выполняется некорректно:

- проверьте, что не было измерено одинаковых точек;
- постарайтесь измерять точки наиболее удаленные друг от друга;
- попробуйте выбрать плоскости с большей площадью;
- проверьте, что точки на отмеченных поверхностях были измерены соответствующими щупами;
- попробуйте выбрать плоскости так, чтобы углы между ними были более близки к прямым.

Привязка по примитивам

Для данной привязки требуется выбрать комплект баз, т.е. набор элементов, которые ограничивают все степени свободы детали. Элементы ограничивают деталь в порядке убывания числа лишаемых степеней свободы (см. **Рисунок 82**). Например, три непараллельных плоскости, или две непараллельных плоскости и цилиндр, ось которого непараллельна прямой, получаемой в результате пересечения плоскостей и т.д.

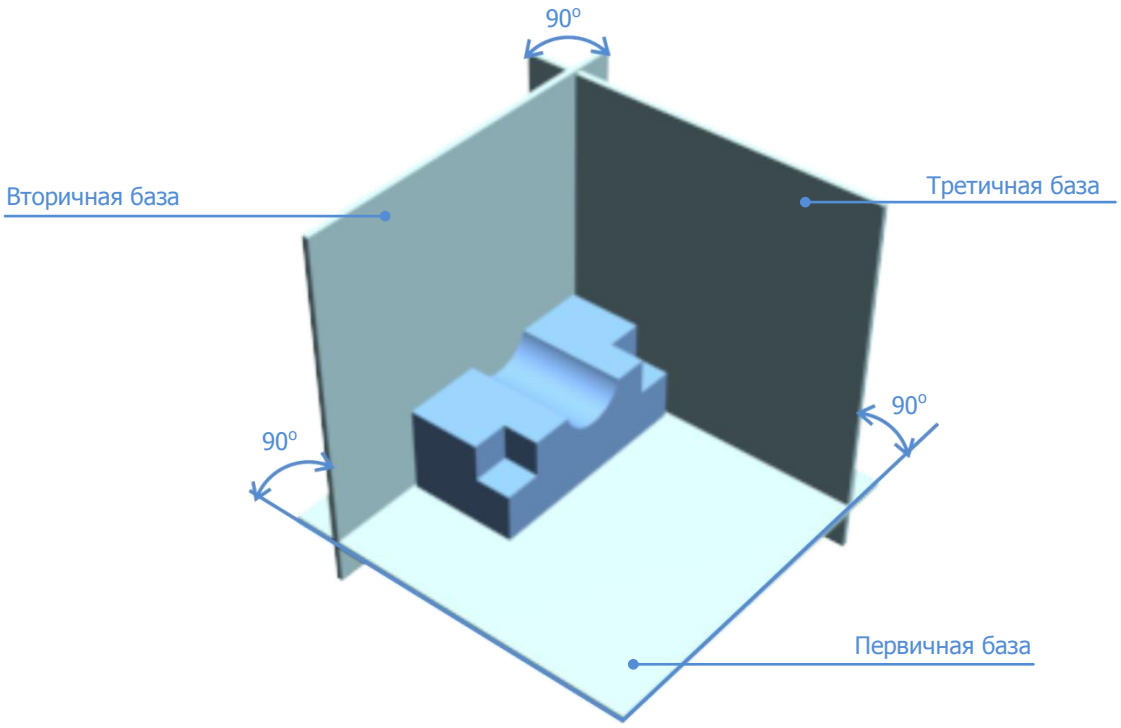


Рисунок 82. Комплект баз

В качестве элементов для привязки можно выбирать:

- Плоскость;
- Цилиндр;
- Конус.

Чтобы начать привязку следует нажать кнопку в главном меню «Привязка по примитивам». Первым шагом мастера будет выбор нескольких элементов, ограничивающих степени свободы (см. **Рисунок 83**).

При выборе элементов рекомендуется выбирать поверхности с большей площадью.

После выбора плоскостей мастер по очереди будет предлагать измерить три, две и одну точку на соответствующих плоскостях, поочередно подсвечивая их. Рекомендуется выбирать наиболее удалённые друг от друга точки. При измерении точек необходимо менять щуп на тот, которым предполагается произвести касание (для этого можно воспользоваться элементом управления в нижнем левом углу).

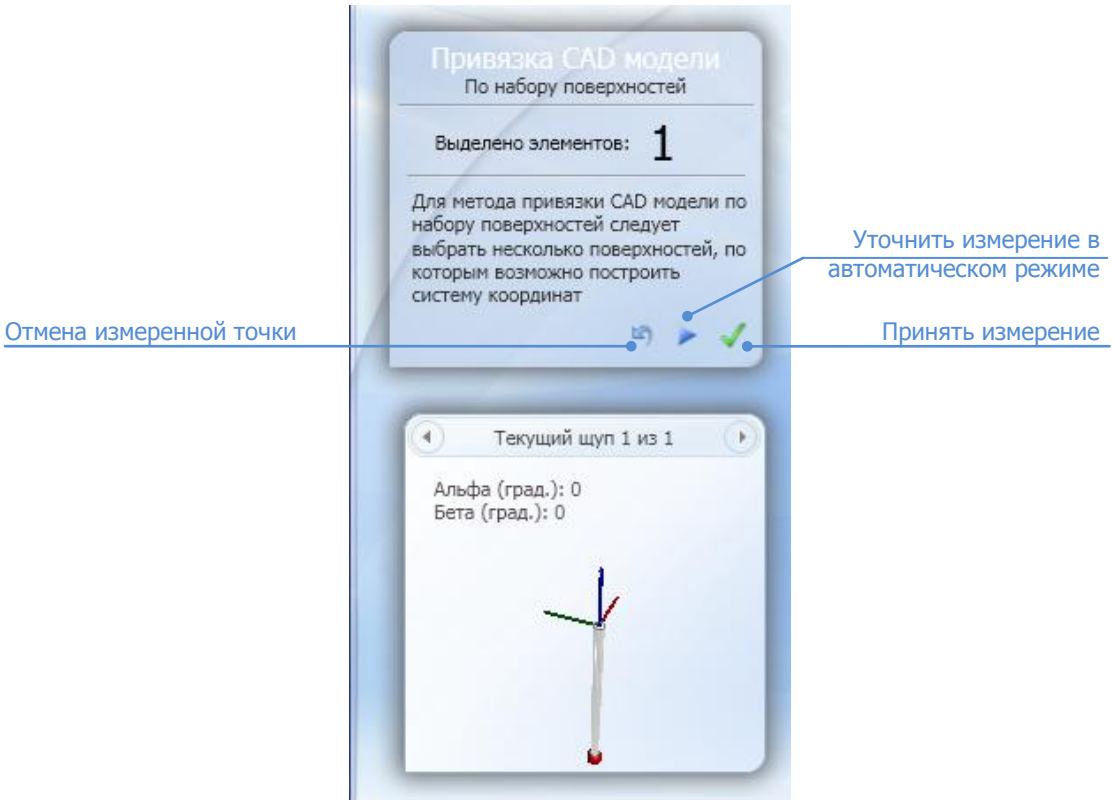


Рисунок 83. Панель для выбора элементов для привязки

Симуляция измерения

Для отладки программы измерения детали предназначена специальная функция – симуляция. Симуляция позволяет разрабатывать и проверять схемы измерения без использования настоящей координатной машины, такой способ программирования КИМ называется offline-программированием.

Для запуска симуляции необходимо открыть для редактирования схему измерения и нажать в главном меню «Схема измерения → Запуск симуляции». Симулятор практически полностью повторяет поведение реальной координатно-измерительной машины, поэтому измерение на виртуальной машине сильно приближено к реальному измерению.

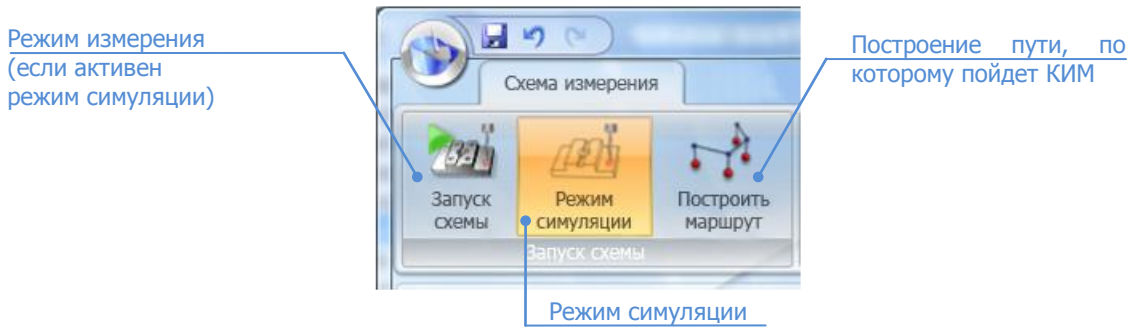


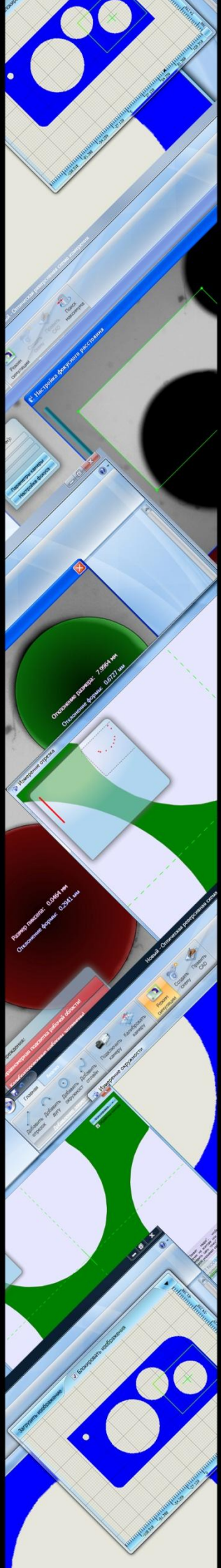
Рисунок 84. Запуск симуляции измерения

В начале работы с симулятором потребуется добавить модель детали. Для этого нажмите кнопку «Добавить деталь», выберите подходящую (деталь может быть загружена из файла в формате *.obj, *.stp, *.step, *.stl, *.cad).

Рисунок 85. Внешний вид симулятора

После этого нажмите кнопку в нижнем левом углу окна, чтобы продолжить процесс запуска схемы измерения. Дальнейшая работа с симулятором не отличается от работы с реальной машиной, необходимо выполнять запрограммированные действия, например, измерять точки в ручном режиме, используя джойстик (в случае, если джойстика нет, то на экране появится эмулятор, см. **Рисунок 86**).

Рисунок 86. Эмулятор джойстика



ТЕХНО КООРД™

ТЕХНОЛОГИЯ КООРДИНАТНЫХ ИЗМЕРЕНИЙ

Оптические схемы измерения

Оглавление

Теоретические основы оптических измерений

Проекционное измерение

В случае проекционного измерения (или измерения на просвет) оптическая система включает в себя следующие компоненты:

- Цифровая камера;
- Объектив;
- Подсветка;

Источник освещения (подсветка) направлена на объектив, который прикреплен к матрице цифровой камеры. Лучи доходящие до матрицы заполняют пиксели матрицы камеры, образуя так называемый фон, остальные пиксели остаются черными, т.к. измеряемый объект не пропустил лучи – это проекция детали. В результате анализа полученного изображения программное обеспечение выделяет контур. *Контур*, найденный на одном кадре – это набор точек в системе координат матрицы камеры. Точки, полученные со всех кадров, служат входными параметрами для последующей математической и метрологической обработки.

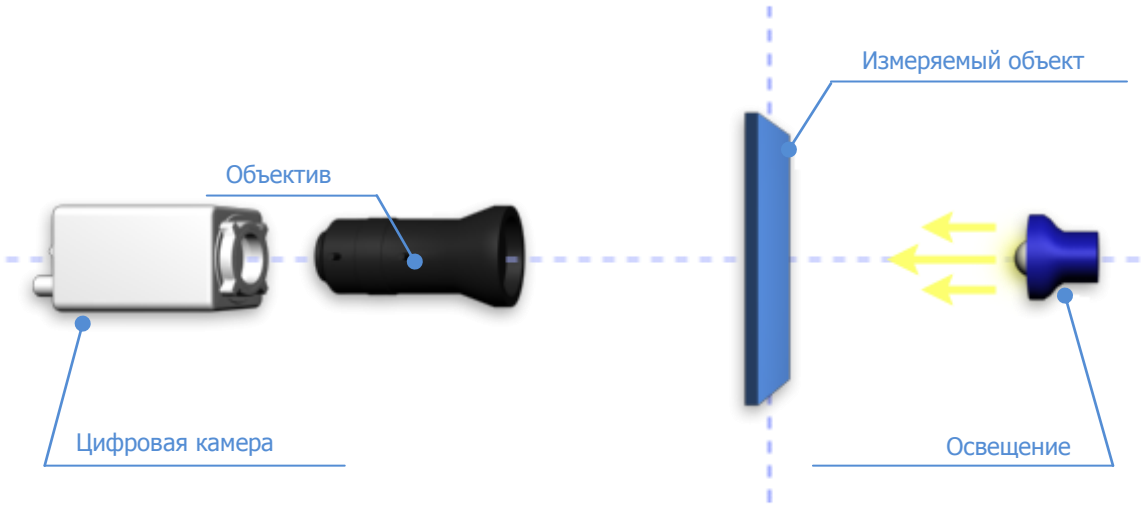


Рисунок 87. Измерение на проходящий свет

Различают телецентрические и асферические объективы. Данное программное обеспечение работает преимущественно с телецентрическими объективами, которые не дают перспективного искажения. Применение асферических объективов возможно после дополнительной специальной пространственной калибровки или для измерения в ручном режиме, где измеряемая точка всегда находится в единственном месте без искажений – в центре.

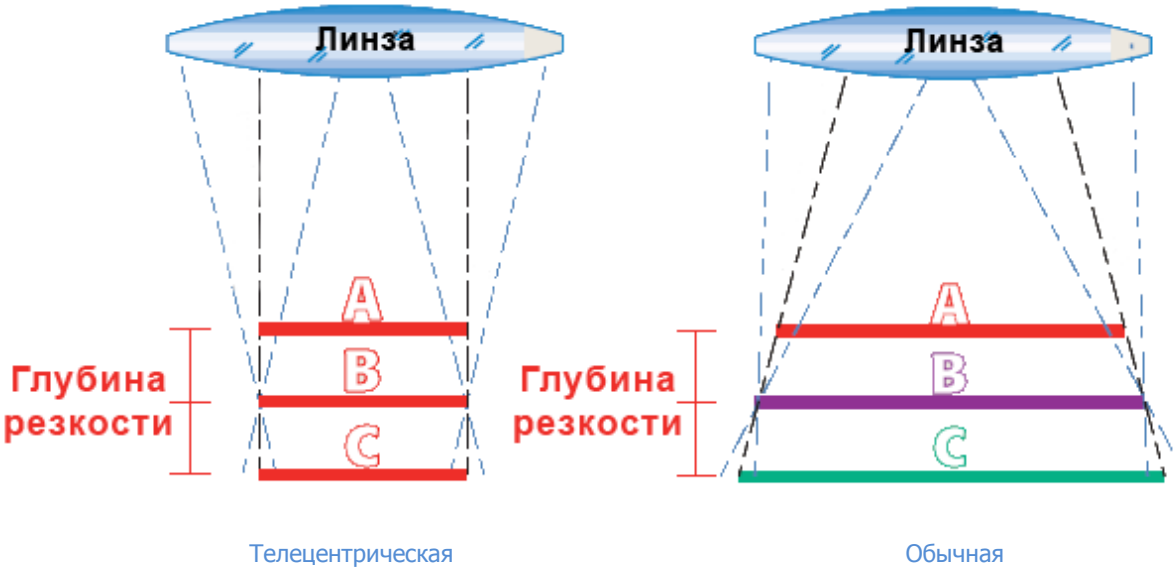


Рисунок 88. Телецентрическая и обычная оптика

Одной из значимых для программного обеспечения особенностей цифровой камеры является разрешение ее матрицы.

Определение контура

Стандартный процесс измерения предполагает объезд камерой кромки детали для того чтобы получить ее контур. Первоначально выделенный контур называется «грубым контуром», т.к. точность его составляет 1px (один пиксель цифровой камеры). После получения грубого контура, производится его уточнение. Для этого запускается специальный алгоритм перехода на субпиксельный уровень. Алгоритм субпиксельного определения контура позволяет получить точность до 1/20px.

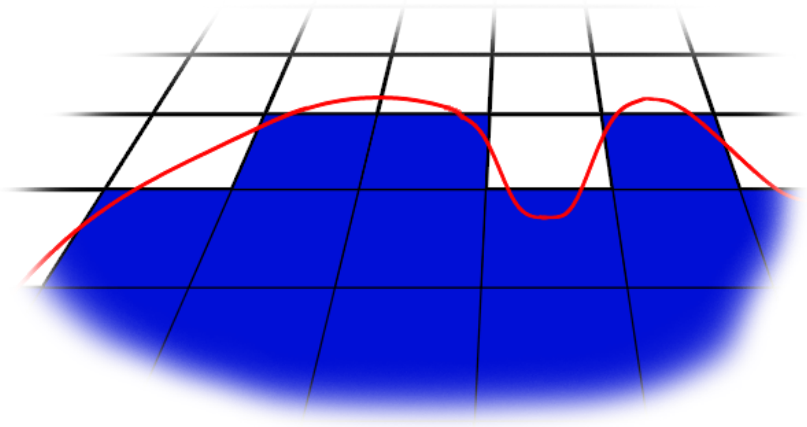


Рисунок 89. Субпиксельный алгоритм

После получения контура состоящего из субпикселей встает вопрос о получении контура в метрической системе измерения, поэтому следующий шаг – это перевод в миллиметры, зная размер пикселя, который вычисляется специальным мастером.

Точность оптической головки

Теоретическая точность оптической головки равна 1/20 части от разрешающей способности матрицы. Реальная точность оптической части обусловлена многими факторами, перечисленными ниже.

Компоненты погрешностей камеры

Кривизна поля изображения

При выборе устанавливаемого объектива следует учитывать искажения, которые допускаются определенной моделью.

Кривизна поля изображения — аберрация, в результате которой изображение плоского объекта, перпендикулярного к оптической оси объектива, лежит на поверхности, вогнутой к объективу. Аберрация вызывает неравномерную резкость по полю изображения. Поэтому когда центральная часть изображения фокусирована резко, то его края будут лежать не в фокусе, и отобразятся нечетко. Если установку на резкость производить по краям изображения, то его центральная часть будет нечеткой.

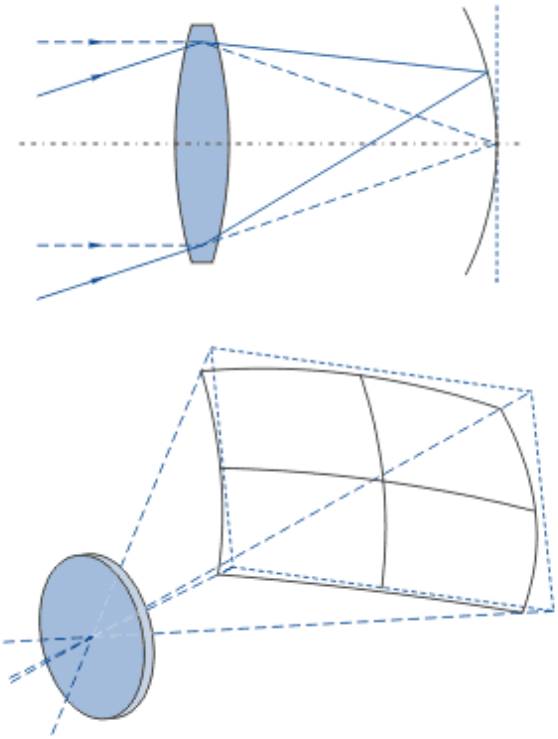


Рисунок 90. Кривизна поля изображения

Кривизна поля изображения является следствием астигматизма. В телецентрических объективах данное искажение, как правило, сведено к минимуму.

Хроматическая аберрация

Возникновение хроматической аберрации вызывается тем, что в построении оптического изображения участвуют лучи света с различной длиной волны, а показатель преломления линз объектива (и, соответственно, и их фокусное расстояние) зависит от длины волны.

Применение подсветки близкой к монохроматической (осветители - зеленые или красные светодиоды с регулируемым током и с диффузным рассеянием, молочными стеклами или синтетической калькой) позволит свести на нет хроматические искажения оптики.

Разрешающая сила объектива

Разрешающая сила объектива — характеристики объектива, отображающие его свойства по передаче чёткого изображения. Разрешающая способность объектива оценивается по количеству воспроизводимых штрихов на 1 мм изображения, который тот способен спроецировать на матрицу цифровой камеры. Измерения разрешающей способности проводят с помощью специальных мир.

Разрешающая сила объективов неоднородна по полю изображения, обычно уменьшаясь к краям изображения. Это обусловлено наличием у объектива аберраций, значение которых на краях всегда больше, чем в центре. Разрешающая сила у объективов одинаковой конструкции уменьшается с увеличением главного фокусного расстояния: у короткофокусных (широкоугольных) она выше, чем у длиннофокусных.

Объективы служат для получения изображения на цифровой матрице, которые также обладают определённой разрешающей способностью. Поэтому для полного использования разрешающей силы объектива следует использовать его с соответствующими фотоматериалами или матрицами, разрешающая способность которых равна или выше разрешающей способности объектива, так как разрешающая способность системы объектив + светочувствительный элемент заведомо не выше разрешения каждого компонента.

Для определения разрешающей силы объектива используют различного вида *ми́ры* — испытательные таблицы с нанесёнными на них штрихами различной ширины и длины.

Разрешающая сила объектива по ГОСТ измеряется в линиях на 1 мм, она всегда больше в центральной части изображения и меньше на его краях. Разрешающая способность системы объектив + светочувствительный элемент приближенно определяется по формуле:

$$\frac{1}{R_S} = \frac{1}{R_O} + \frac{1}{R_E},$$

где – R_O разрешающая сила объектива в линиях на 1 мм; R_E — разрешающая сила матрицы в линиях на 1 мм.

Цифровой шум матрицы

Цифровой шум проявляется в виде случайным образом расположенных элементов растра (точек), имеющих размеры близкие к размеру пикселя. Цифровой шум отличается от изображения более светлым или тёмным оттенком серого и цвета (яркостный шум) и/или по цвету (хроматический шум). Цифровой шум вносит случайную погрешность.

Подавление цифрового стохастического шума проводится усреднением. Данная функция подавления шума реализована в программном обеспечении.

Другие погрешности матрицы

Существует большое разнообразие цифровых камер, которые имеют специфические погрешности и особенности. Большое значение имеет настройка камеры, т.е. регулировка различных ее параметров. Специальный мастер настройки камеры может помочь произвести оптимальную настройку камеры.

Компоненты погрешностей монтажа

Наклон камеры

Ось обзора камеры должна быть перпендикулярна измерительной плоскости. Отклонение от перпендикулярности вызывает искажения, потерю фокуса в некоторой области и др.

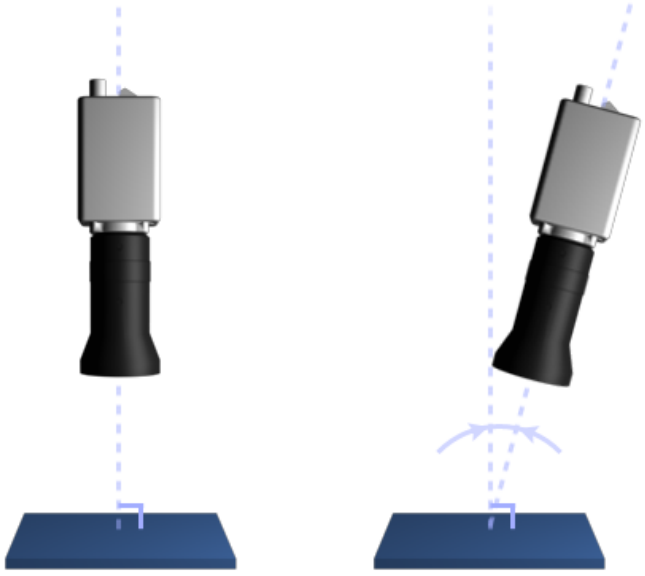


Рисунок 91. Наклон камеры

Разворот камеры

Матрица камеры должна быть выровнена по осям перемещения. Данный компонент погрешности является значимым. При измерении, контуры, полученные с разных кадров, собираются в один, разворот камеры приводит к искажению формы контура.

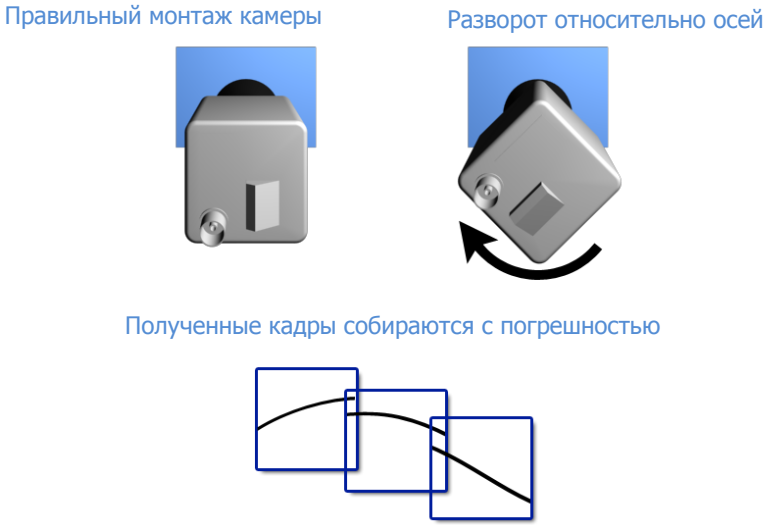


Рисунок 92. Разворот камеры

Разворот камеры приведет к тому, что при сборке общего контура конец одного контура на одном кадре не будет совпадать с началом на следующем кадре.

Дефокусировка

Фокусное расстояние должно быть одинаковым при вычислении размера пикселя и при измерении. Высота калибратора не совпадает с высотой измеряемого объекта, поэтому необходимо изменять положение камеры. Для точной настройки фокусного расстояния предназначен специальный инструмент (см. «Подключение и настройка камеры в программном обеспечении **ТЕХНО**координ™»).

Неперпендикулярность подсветки

Источник освещения должен быть направлен строго перпендикулярно объективу камеры и измерительной плоскости. В случае несоблюдения данного требования будет происходить искажение контура.

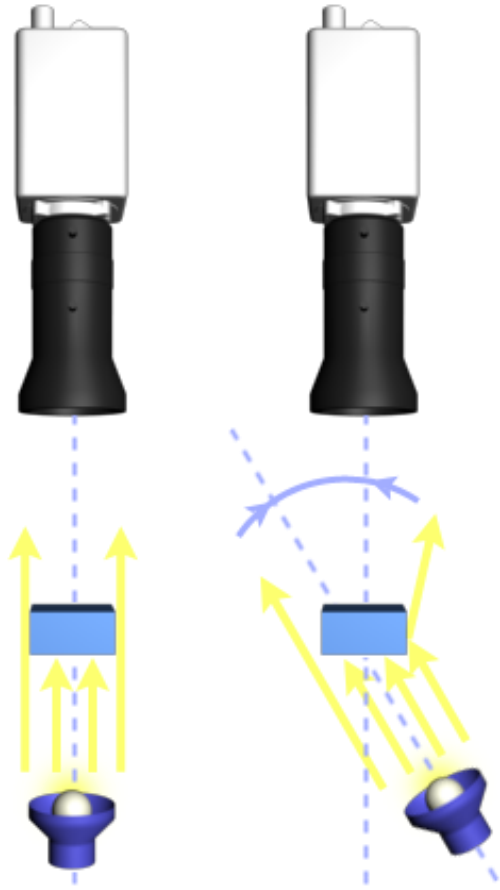


Рисунок 93. Неперпендикулярность подсветки

Непараллельность измерительной плоскости и плоскости перемещения камеры

При перемещении камеры в пределах рабочей области расстояние от объектива до измеряемого объекта должно быть постоянным, иначе произойдет потеря фокусного расстояния.

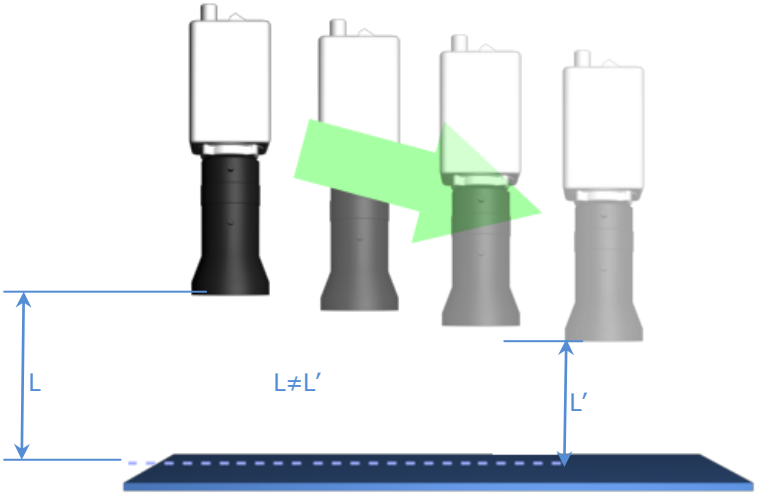


Рисунок 94. Непараллельность плоскостей: измерительной и перемещения камеры

Другие компоненты погрешностей

Неравномерная подсветка

Проходящий свет должен иметь одинаковую интенсивность на всей рабочей области. Неравномерная интенсивность освещения приводит к искажению выделяемого контура. Для диагностики неравномерности подсветки может помочь специальное окно диагностики (см. на стр. 92).

Неверная яркость подсветки

Может оказаться, что подсветка слишком яркая или наоборот слишком тусклая. Это может негативно сказаться на алгоритме выделения контура, т.к. цвета фона или объекта будут находиться на границе цветового диапазона, или даже за ним, т.е. произойдет потеря информации. Окно диагностики способно определить данную проблему (см. на стр. 92). Из простых решений данной проблемы может быть регулировка интенсивности ламп подсветки или изменение параметра выдержки камеры.

Влияние внешнего освещения

Внешнее освещение может оказывать влияние. Наиболее существенным примером может являться появление бликов.

Форма измеряемого объекта

К сожалению, форма объекта так же оказывает влияние на точность измерения, особенно когда речь идет об особо прецизионных измерениях. На объектах сферической формы, объектах с фасками можно наблюдать дифракционную аберрацию. Но это сильно зависит от качества объектива и подсветки.

Подключение и настройка камеры в программном обеспечении **ТЕХНО**координ™

Прежде, чем начать измерение необходимо однократно подключить и откалибровать реальную камеру или симулятор. Начать следует с выбора объектива и разрешения камеры.

Следует учитывать, что чем больше увеличение, тем меньше область видимости камеры, а это значит, что время проведения измерения увеличивается, но точность возрастает. Чем больше разрешение матрицы камеры, тем выше точность, но тем медленнее обработка полученных кадров. При выборе следует учитывать теоретическую точность.

Чтобы подключить камеру, следует нажать в главном меню кнопку «Подключить камеру», после чего появится окно (см. **Рисунок 95**). В первом списке предлагается выбрать цифровую камеру, подключенную к компьютеру. Когда камера подключится, то появится возможность выбрать разрешение камеры.

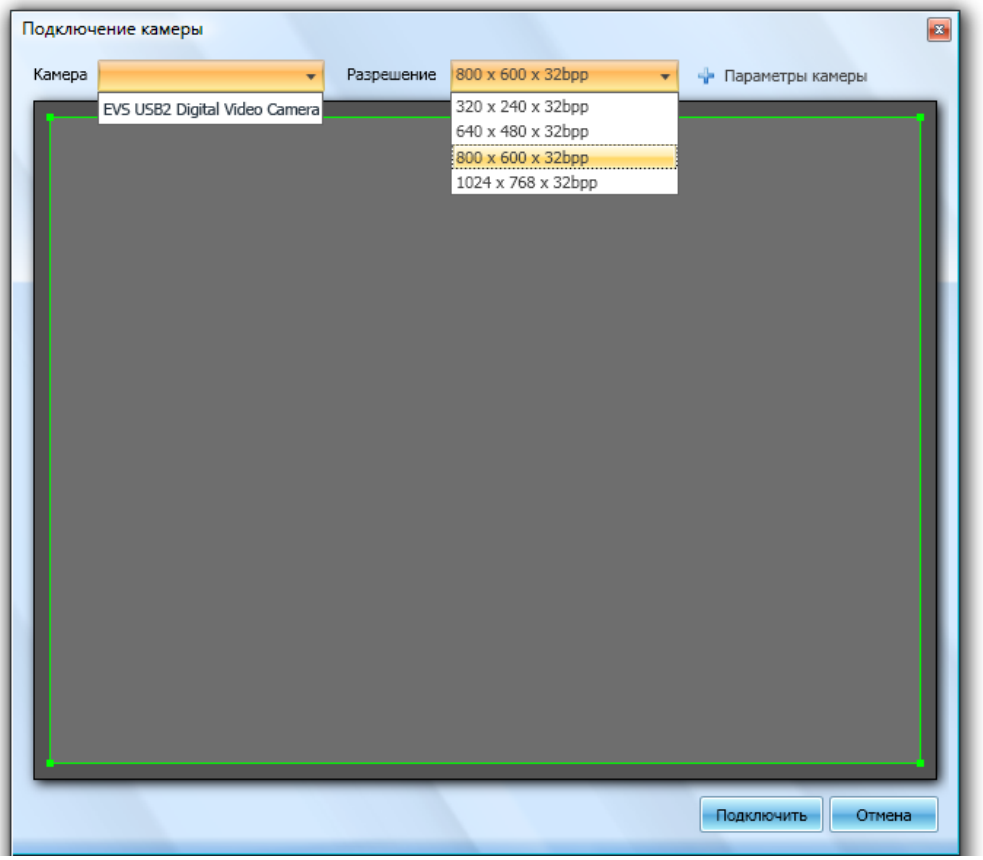


Рисунок 95. Окно подключения камеры

Разрешение камеры рекомендуется выбирать, руководствуясь формулой приведенной на стр. 84

При необходимости можно настроить отдельные параметры камеры, если нажать на кнопку «Параметры камеры».

При этом появится окно с параметрами, относящимися к конкретной цифровой камере, т.е. для другой камеры они будут другими.

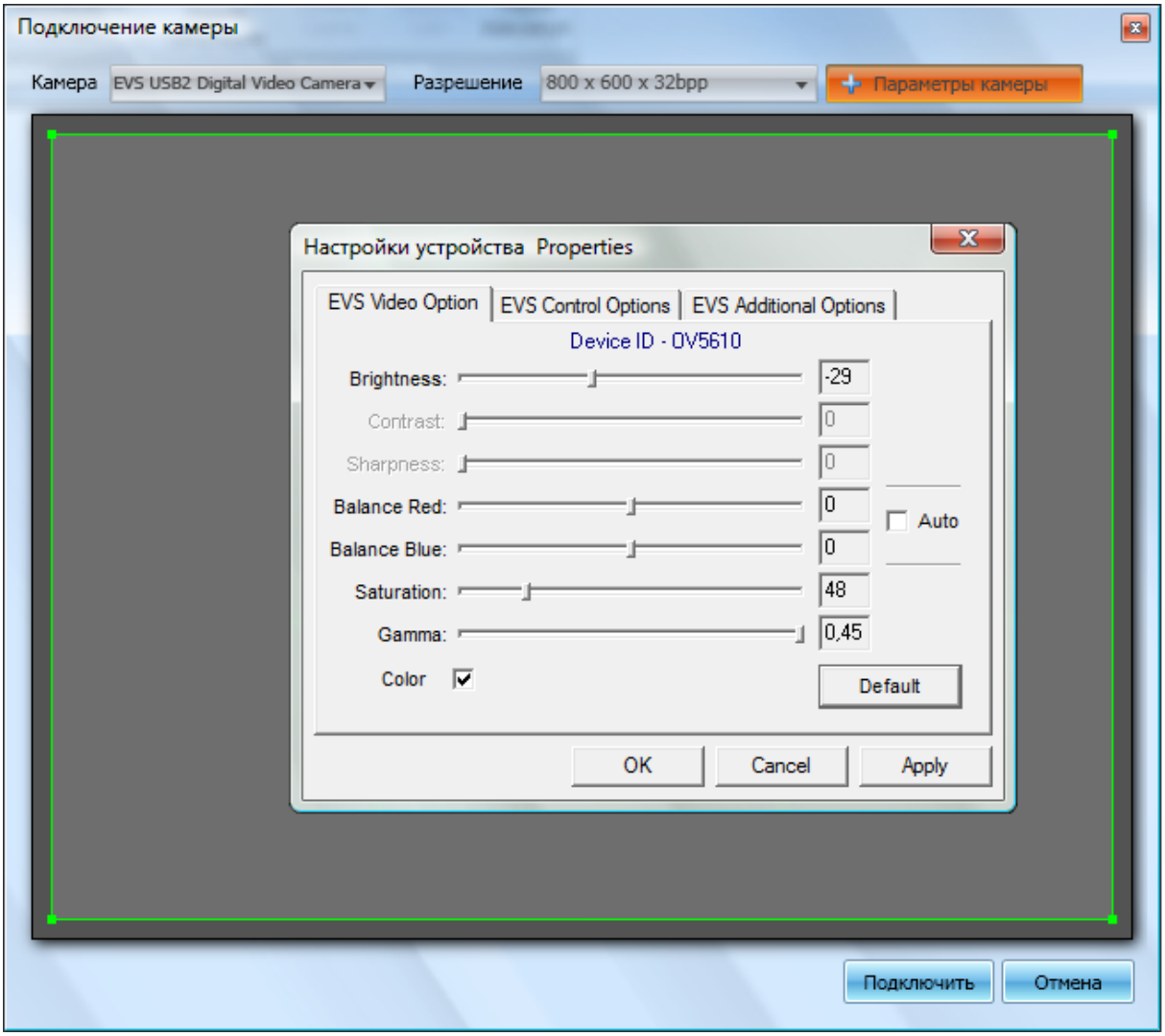


Рисунок 96. Настройка параметров камеры

Параметры камеры можно настроить позже в окне калибровки камеры

Последняя важная процедура – это настройка «полезной области». В ряде случаев возникает потребность в том, чтобы не использовать часть изображения, которое находится на краях. Потребность в этом может возникнуть из-за геометрических аберраций оптики или особенностей матрицы. «Полезная область» обозначена зеленым прямоугольником и настраивается маркерами (зеленые квадратики) в углах прямоугольника.

Чтобы принять новые настройки, нажмите «Ok».

Подключение камеры необходимо производить только один раз.
При следующем запуске программы камера будет подключена автоматически с сохраненными настройками

Калибровка камеры

Чтобы запустить калибровку камеры, необходимо выбрать «Калибровать камеру» в главном меню. Появится окно калибровки камеры.

В окне калибровки камеры можно настроить параметры камеры (см. **Рисунок 96**) и фокусное расстояние.

Изменение параметров освещения, настроек камеры и т.д. изменяет размер пикселя и другие калибруемые параметры, поэтому все настройки необходимо сделать до калибровки камеры, или выполнить перекалибровку.

Поддержание постоянного фокусного расстояния до измеряемого объекта – важная задача. Погрешность в выборе фокусного расстояния может привести к ложным результатам. Для того чтобы установить оптимальное фокусное расстояние, существует специальный мастер настройки фокуса. Чтобы запустить мастер необходимо нажать кнопку «Настройка фокуса» в окне калибровки камеры. После чего появится окно, показанное на рисунке ниже.

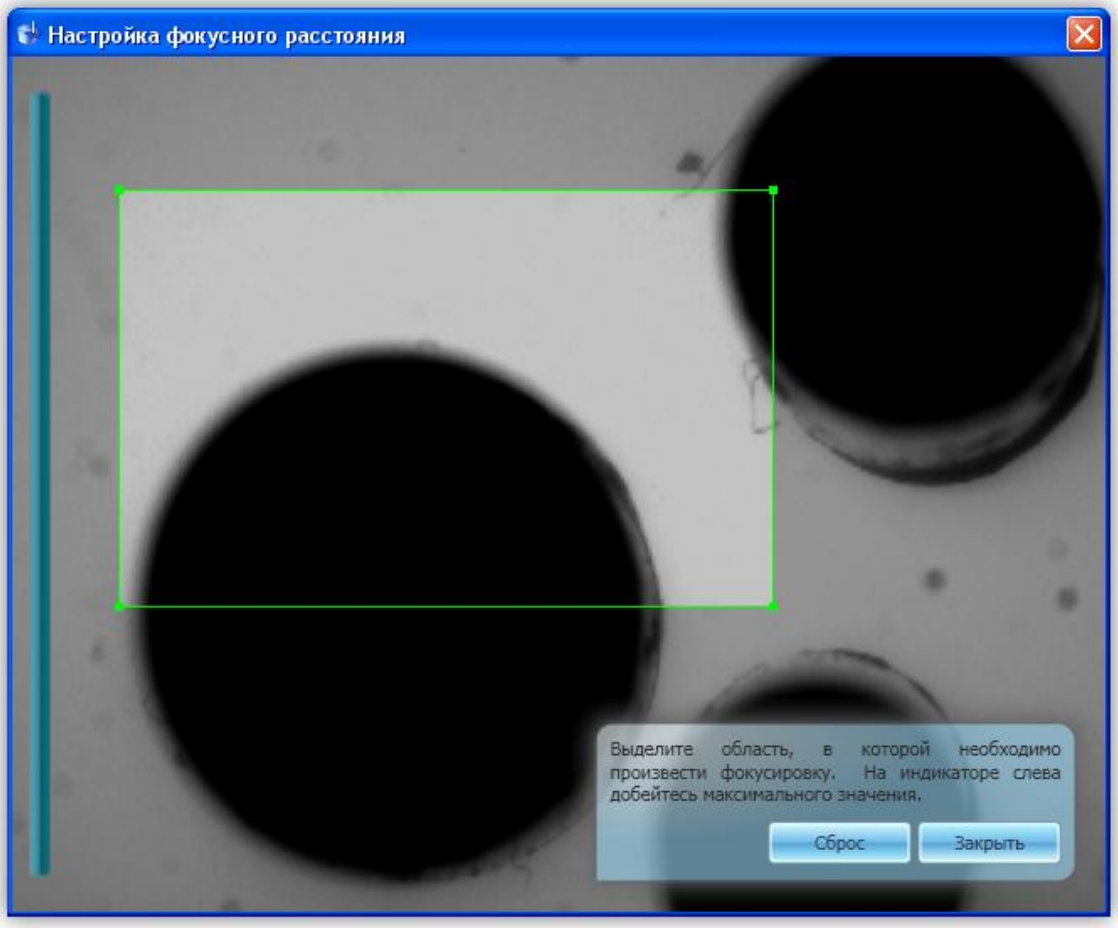


Рисунок 97. Настройка фокусного расстояния

В данном окне нужно навести область видимости на кромку детали и вручную отрегулировать фокусное расстояние: необходимо добиться максимального значения на индикаторе слева. Затем следует нажать кнопку «Далее».

Точное позиционирование на кромке вовсе не обязательно

Будьте внимательны, при каждой смене измеряемого объекта необходимо настраивать фокусное расстояние.

Для оптической КИМ, оборудованной двигателями по оси Z, существует возможность автоматической фокусировки камеры.

Расчет размера пикселя

Основной функцией калибровки камеры является расчет размера пикселя. Для расчета размера пикселя и диагностики требуется специальный калибратор. Калибратор должен удовлетворять следующим требованиям:

- В проекции калибратор должен давать три круга;
- Калибратор должен умещаться в поле видимости камеры;
- Точность изготовления калибратора должна соответствовать заявленной паспортной точности прибора. Отклонение формы калибратора должно быть минимум в три раза меньше заявленной точности прибора. При этом размер калибраторов может быть просто аттестован;

Некоторые рекомендации по изготовлению калибратора:

- Рекомендуется, чтобы один из кругов был много больше остальных. Это позволит получить большую точность при расчете размера пикселя, потому что точность расчета размера пикселя зависит прямо пропорционально размеру калибратора;
- Рекомендуется в качестве калибратора использовать цилиндры с минимальной фаской. Это позволит свести к минимуму дифракционную аберрацию.

Чтобы рассчитать размер пикселя необходимо ввести аттестованные значения окружностей, проецируемых калибратором, и нажать кнопку «Запустить калибровку» в окне калибровки камеры (см. **Рисунок 98**).

Введенные значения будут сохранены и при следующем открытии данного окна будут восстановлены прежние значения

На полученном кадре выделяются контуры и распознаются три окружности. По наибольшей из них производится расчет размера пикселя. После этого производится аппроксимация остальных двух окружностей с целью определить отклонение от размера и формы. Поскольку погрешность формы калибратора незначительна, а размер аттестован с высокой точностью, полученные отклонения являются

погрешностью измерения. Таким образом, можно получить представление о том, какую погрешность вносит в измерительную систему оптическая часть.

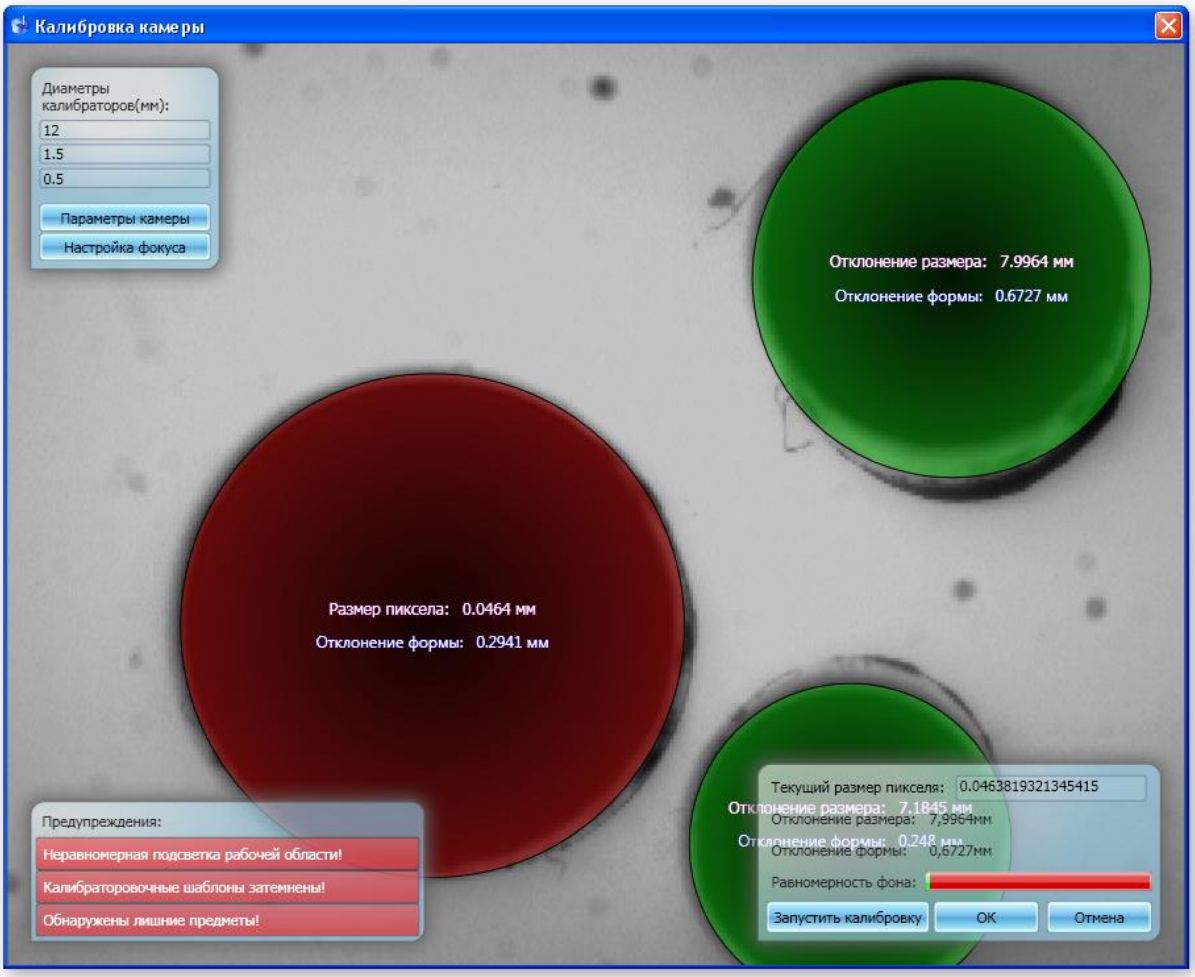


Рисунок 98. Вид окна диагностики

Если результаты удовлетворительные, можно нажать кнопку «ОК» чтобы применить рассчитанное значение пикселя.

Если полученные результаты являются неудовлетворительными, то необходимо проводить наладочные работы, среди которых:

- Настройка параметров камеры (чтобы вызвать окно с параметрами камеры нажмите кнопку «Параметры камеры»);
- Проверка монтажа элементов измерительной головки;
- Настройка подсветки;

Если при калибровке камеры возникли трудности в расчете размера пикселя, то левом нижнем углу окна калибровки появятся предупреждения. Предупреждения говорят о серьезной проблеме, которая может искажать результаты или снижать точность измерений. Расшифровка предупреждений приведена в таблице ниже:

Калибраторы не обнаружены на изображении	Следует поместить калибровочные окружности в область захвата и настроить фокус, подробнее см. **Рисунок 98**.

Неравномерная подсветка рабочей области	Это говорит о серьезной проблеме в системе. Алгоритм выделения контуров определяет фон и деталь. В данном случае имеют место третьи цвета, которые могут повлиять на точность выделения контуров. В частности может быть вызвано неравномерностью размещения ламп подсветки или отражениями от внешнего окружения, а также хроматической аберрацией объектива.
Фон засвечен	Засвеченный фон негативно сказывается на точности выделения контуров. Можно уменьшить интенсивность подсветки или уменьшить выдержку камеры. Данная проблема описана на стр. 88
Калибровочные шаблоны затемнены	Излишне затемнённые калибраторы негативно сказывается на точности выделения контуров. Можно уменьшить интенсивность подсветки или увеличить выдержку камеры. Данная проблема описана на стр. 88
Недостаточно калибровочных шаблонов	В области захвата было выделено недостаточное для калибровки количество окружностей. Возможно, калибратор не полностью попадает в область захвата. Подробнее о калибраторе см. выше.
Обнаружены лишние предметы	На изображении были обнаружены контуры лишних предметов. Это может очень негативно отразиться на калибровке. Следует проверить чистоту кромок калибратора, отсутствие пятен и царапин на фоне и пр.

ВНИМАНИЕ! После окончания калибровки параметры камеры, объектива (за исключением фокусного расстояния), параметры подсветки изменять категорически нельзя.

Режим симуляции

Симуляция позволяет разрабатывать и проверять схемы измерения без использования настоящей оптической системы, такой способ программирования КИМ называется offline-программированием.

Чтобы включить режим симуляции необходимо щелкнуть по соответствующей кнопке в главном меню программы. Симулятор практически полностью повторяет поведение реальной камеры, поэтому измерение на виртуальной машине сильно приближено к реальному измерению.

При подключении виртуальной камеры можно выбрать для нее разрешение.

Щелчок по кнопке «Подключить» сворачивает окно симулятора. Чтобы вернуть окно симулятора необходимо дважды щелкнуть по иконке в области уведомлений (см. **Рисунок 99**).

Симулятор оснащен координатными линейками в левой и нижней части окна, а также координатной сеткой по всей рабочей поверхности симулятора.

Область видимости камеры обозначена зеленым прямоугольником с перекрестием в центре (см. **Рисунок 99**).

Перемещение области видимости камеры осуществляется левой клавишей мыши или стрелочками на цифровой клавиатуре. Масштабирование изображения осуществляется колесиком мыши. Перемещение окна симулятора по рабочей поверхности осуществляется средней клавишей мыши.

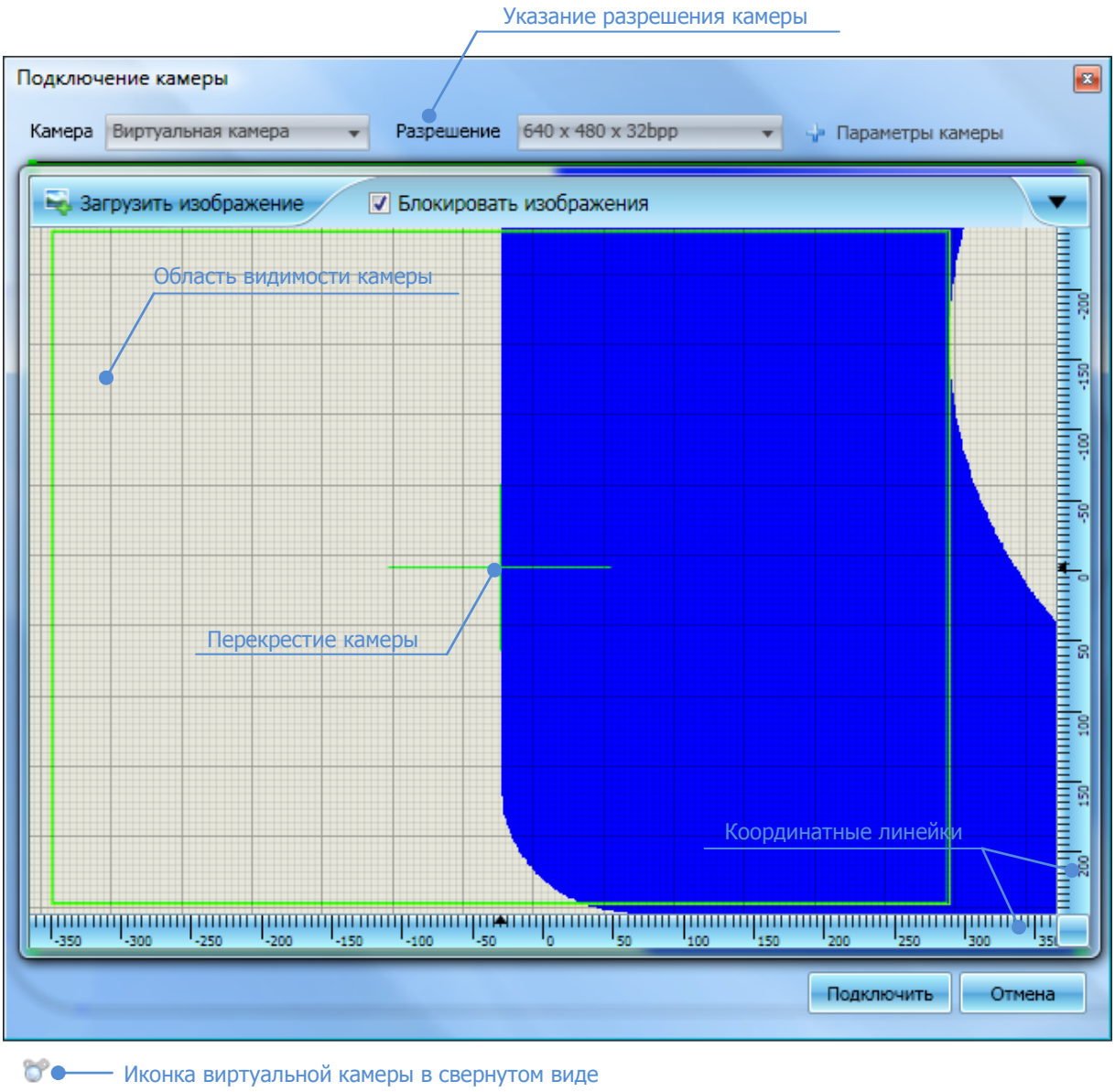


Рисунок 99. Окно виртуальной камеры

В симуляторе предусмотрена работа с несколькими изображениями одновременно. Изображения можно загружать через кнопку «Загрузить изображение». Поддерживаемые форматы:

- *.PNG (Portable Network Graphics);
- *.JPEG (Joint Photographic Experts Group);
- *.GIF (Graphics Interchange Format);
- *.BMP (Bitmapped Graphics Format);
- *.TIFF (Tagged Image File Format).

Для загруженных изображений необходимо создать изображения калибраторов.

Также можно загрузить CAD-модель детали из формата *.DXF. Использование модели предпочтительнее. Размер модели в симуляторе определятся в зависимости от введенного размера пикселя в окне калибровки, что обеспечивает повышенную точность измерения.

Изображения можно перемещать мышью по полю симулятора или клавишами управления курсором (при этом флажок «Блокировать изображение должен быть снят», а изображение выделено (выделенное изображение обозначено красной рамкой)). Изменить порядок наложения изображений друг на друга или удалить изображение можно, вызвав правой кнопкой мыши контекстное меню выделенного изображения.

Чтобы избежать случайного смещения изображения необходимо поставить флажок «Блокировать изображения»

Диагностика повторяемости

Диагностику повторяемости предлагается проводить исходя из идеи о том, что при изменении положения калибратора результат калибровки в идеале меняться не должен, в реальности же значения будут меняться. Разницу между полученными значениями в данном контексте будем называть повторяемостью.

Чтобы оценить повторяемость следует сделать следующее. Передвинуть калибратор в новое положение. Нажать кнопку «Запустить калибровку». Сравнить с предыдущими результатами. Так необходимо сделать несколько раз.

Стандартная оптическая схема измерения

Назначение и преимущества

Стандартная оптическая схема измерения является высокоинтерактивным инструментом для измерения различных деталей на основе построенной CAD-модели с применением различных типов оптических координатно-измерительных машин. Благодаря высокой интеллектуальности данной схемы измерения от пользователя требуются только следующие знания и умения:

- базовые навыки работы с операционной системой Microsoft Windows™.
- знание основных принципов координатных измерений;
- знание основ оптических измерений;
- общие знания в области аналитической геометрии.

Данный тип схемы измерения позволяет:

- выполнять измерения в автоматическом цикле на оптической координатной машине, оснащенной двигателем;
- производить привязку системы координат машины к системе координат CAD-модели;
- проводить измерения на основе построенной CAD-модели детали;
- проводить измерения как с применением реальной оптической измерительной системы, так и в режиме симуляции;
- визуализировать движения камеры относительно детали;
- передавать данные для последующих расчетов в мощный инструмент «координатный анализ» (см. главу «Координатный анализ геометрических параметров детали»);
- производить настройку отчетной формы и выводить ее на печать.

Реверсивная оптическая схема измерения ориентирована на:

- проведение сложных автоматизированных измерений на основе CAD-модели детали;
- быстрое получение результата;
- применение с оптическими измерительными системами, оснащенными двигателем.

Редактор схемы измерения

Структура редактора

Редактор схемы измерения состоит из окна, включающего главное меню, расчетную программу и интегрированный координатный анализ. Круглая кнопка в верхнем левом углу окна позволяет работать с файлом схемы измерения: сохранить, открыть, создать, отправить на печать и т.д (см. **Рисунок 100**).

Редактор сохраняет и открывает файлы с расширением *.standerdoptics

Рекомендуемый порядок работы

При создании новой стандартной оптической схемы измерения программа требует указать CAD-модель детали для измерения в формате *.dxf.

Стратегия измерения создается автоматически. Пользователь необходимо только выбрать двойным щелчком мыши элементы для измерения. Выбранные элементы отображаются в списке элементов для измерения в окне редактора схемы измерения слева (см. **Рисунок 100**).

Схему измерения можно запустить как на реальной машине, так и в режиме симуляции.

Прежде, чем начать измерения необходимо однократно подключить и откалибровать реальную камеру или симулятор (см. «Подключение и настройка камеры в программном обеспечении **ТЕХНО**координ™»). Мастер калибровки камеры поможет провести диагностику оптической головки, рассчитать размер пикселя, настроить подсветку, определить цвет фона и цвет детали.

Для того чтобы производить перемещения машины вокруг детали, измерять точки на ее поверхности необходимо узнать как расположена деталь на столе. Такой процесс называется привязка текущего расположения детали к CAD-модели. Привязка производится в ручном режиме. Пользователю достаточно измерить три точки, расположенных, после чего будет выполнен расчет перевода координат из системы координат модели в систему координат детали.

Если пользователь не произвел предварительную калибровку камеры и привязку, программа потребует выполнить данные настройки перед измерением.

Полученные после измерения данные автоматически отображаются в отчете. Также дополнительные данные можно получить в расчетной программе. Для всех отображенных в отчете параметров можно настроить их представление, допустимые отклонения, квалитеты и т.д.

Подробно о редактировании отчетной формы и выводе результатов следует читать в главе «Координатный анализ геометрических параметров детали».

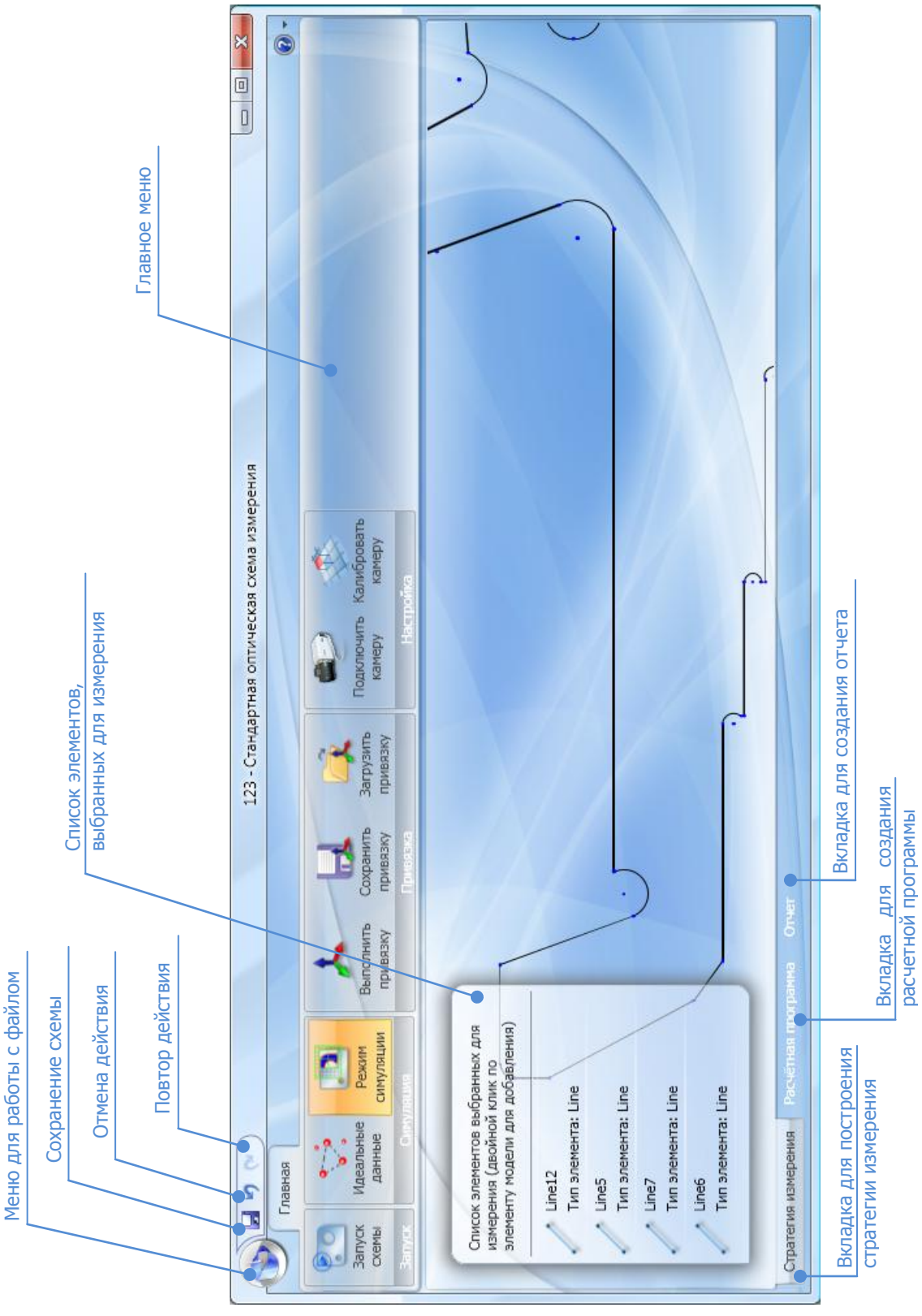


Рисунок 100. Окно редактора стандартной оптической схемы измерения

Привязка CAD-модели

Мастер привязки можно вызвать щелчком по кнопке «Выполнить привязку» в главном меню (см. **Рисунок 100**).

Необходимо выбрать не менее трех опорных точек для привязки чертежа к детали. Для привязки можно также использовать прямые, окружности и дуги (в этом случае для привязки будут использоваться точки, полученные в результате пересечения этих элементов).

Затем необходимо измерить выбранные точки. Красным цветом обозначается точка, предназначенная для измерения. Необходимо навести на нее перекрестие камеры и нажать кнопку «Измерить точку». Измеренная точка окрашивается в зеленый цвет.

Кнопка «Отменить» отменяет последнюю измеренную точку на текущем (выделенном) элементе.

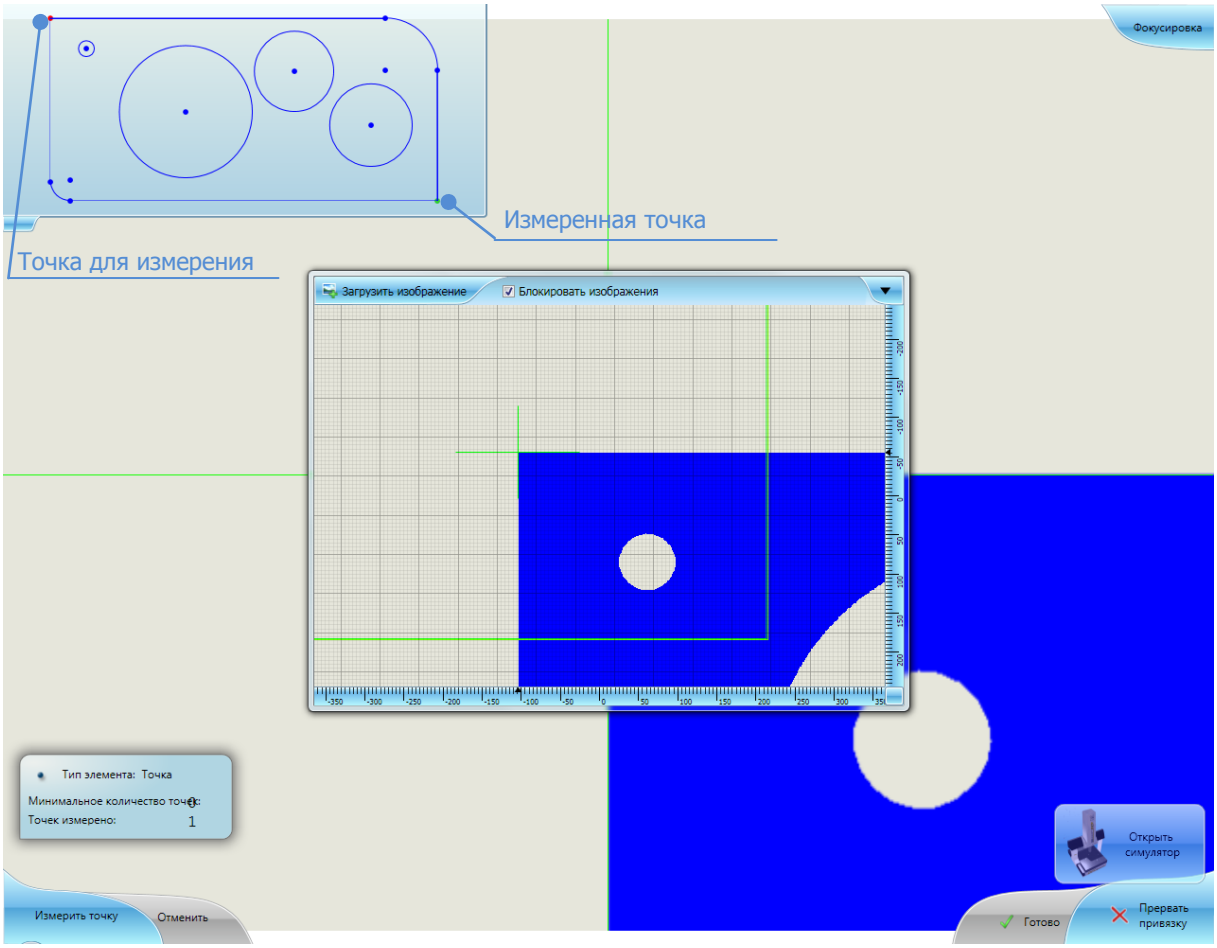


Рисунок 101. Привязка CAD-модели к детали

После измерения опорных точек необходимо подтвердить правильность привязки, нажав на кнопку «Готово»

В процессе выполнения привязки CAD-модели к детали можно выполнить фокусировку камеры. Щелчок по кнопке «Фокусировка» вызывает мастер фокусировки.

Расчетная программа

Часто возникает необходимость произвести координатный расчет нестандартизованных параметров. Для данной цели существует расчетная программа.

Расчетная программа позволяет проводить расчет необходимых параметров на специально разработанном языке (Measurements Scripting). Пользователю не нужно помнить все возможные методы и свойства объектов языка, т.к. во время ввода текста программы появляются списки доступных в данный момент методов и свойств объекта со справочным материалом для каждого метода и свойства, а пользователю необходимо лишь выбрать нужный. В программе предусмотрено выявление синтаксических ошибок: предупреждения и ошибки отображаются в нижней части окна расчетной программы. (см. **Рисунок 102**). Подробное описание языка координатных измерений см. главу «Язык для математической обработки данных координатных измерений».

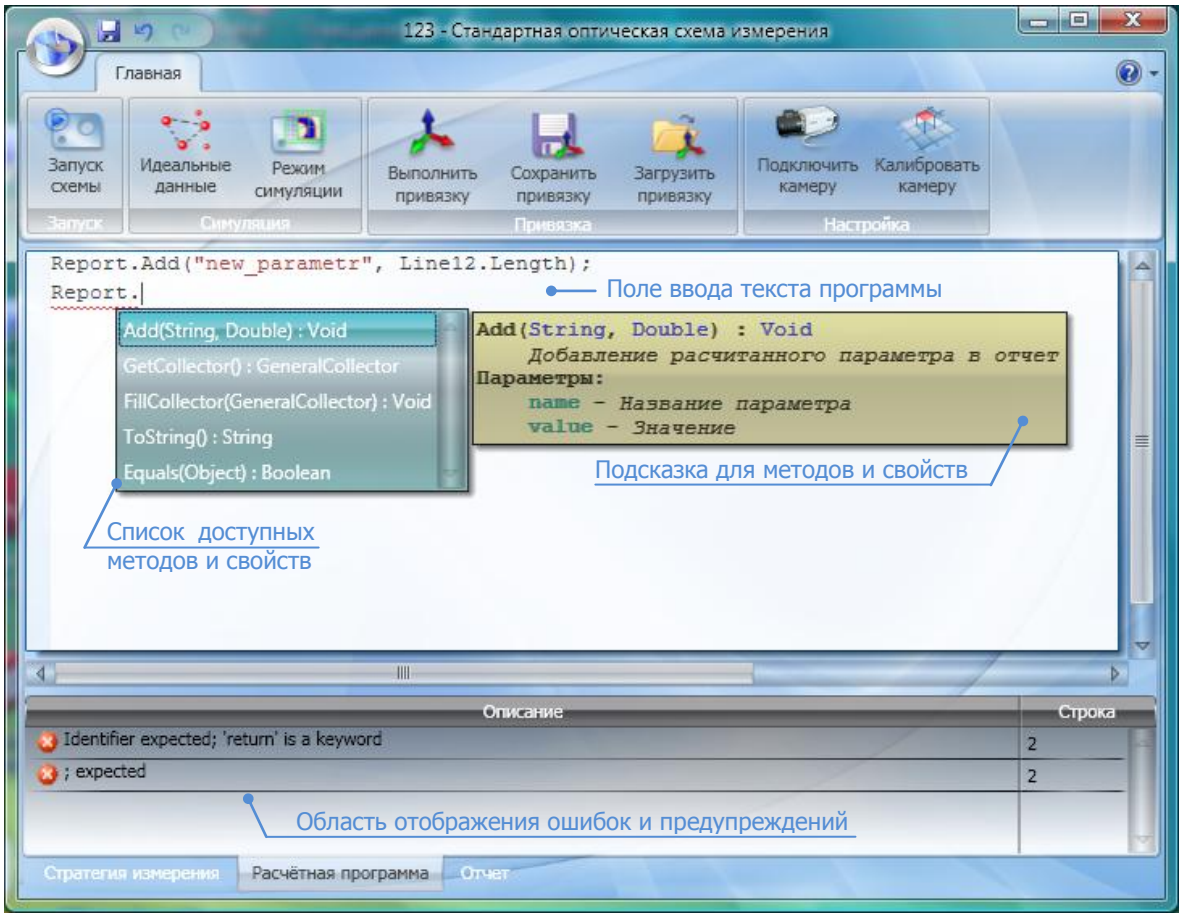


Рисунок 102. Вкладка расчетная программа

Параметры, рассчитанные в расчетной программе можно передать в координатный анализ. Для передачи параметра используется конструкция вида `Report.Add("Название параметра", x)`, где *x* – это числовое значение параметра (или переменная хранящее это значение). Затем в отчете можно будет добавить данный параметр.

Реверсивная оптическая схема измерения

Назначение и преимущества

Реверсивная оптическая схема измерения является высокоинтерактивным инструментом для измерения различных деталей без предварительного построения CAD-модели с применением различных типов оптических координатно-измерительных машин. Благодаря высокой интеллектуальности данной схемы измерения от пользователя требуются только следующие знания и умения:

- базовые навыки работы с операционной системой Microsoft Windows™.
- знание основных принципов координатных измерений;
- знание основ оптических измерений;
- общие знания в области аналитической геометрии.

Данный тип схемы измерения позволяет:

- проводить измерения без предварительного построения CAD-модели;
- измерять точки вручную: путем указания точек для измерения, и автоматически: путем указания контура для измерения.
- проводить измерения как с применением реальной оптической измерительной системы, так и в режиме симуляции;
- передавать данные для последующих расчетов в мощный инструмент «координатный анализ» (см. главу «Координатный анализ геометрических параметров детали»);
- передавать полученную схему для измерения в стандартную оптическую схему измерения;
- производить настройку отчетной формы и выводить ее на печать.

Реверсивная оптическая схема измерения ориентирована на:

- проведение достаточно сложных измерений без предварительного построения CAD-модели;
- быстрое получение результата;
- использование для модернизации старых оптических систем.

Редактор схемы измерения

Структура редактора

Редактор схемы измерения состоит из окна, включающего главное меню и интегрированный координатный анализ. Круглая кнопка в верхнем левом углу окна позволяет работать с файлом схемы измерения: сохранить, открыть, создать, отправить на печать и т.д (см. **Рисунок 103**).

Редактор сохраняет и открывает файлы с расширением *.reverseoptics

Рекомендуемый порядок работы

Прежде, чем начать измерения необходимо однократно подключить и откалибровать реальную камеру или симулятор (см. «Подключение и настройка камеры в программном обеспечении **ТЕХНО**коорд™»).

Мастер калибровки камеры поможет провести диагностику оптической головки, рассчитать размер пикселя, настроить подсветку, определить цвет фона и цвет детали.

Для проведения измерений используется мастер измерений элементов - очень удобный инструмент, позволяющий быстро и просто получать необходимые измерения.

Чтобы начать измерение, нужно навести камеру на измеряемый элемент, произвести захват кадра. При захвате кадра производится стабилизация изображения и выделение контура. Стабилизация позволяет снизить уровень шума. В результате получится облако точек, находящихся на контуре детали.

Выделение контура происходит с субпиксельной точностью. Это позволяет на основе цветовых градаций на границе перехода от фона к детали выделять истинное положение контура с точностью до долей пикселя.

Затем необходимо убрать точки, не принадлежащие элементу детали. Результат измерения автоматически отображается в отчете.

После того, как получено облако точек следует производить аппроксимацию примитивов и расчет контролируемых параметров.

Рассчитанные параметры будут отображены в отчете, где можно настроить их представление, допустимые отклонения, квалитеты и т.д.

Подробно о редактировании отчетной формы и выводе результатов следует читать в главе «Координатный анализ геометрических параметров детали».

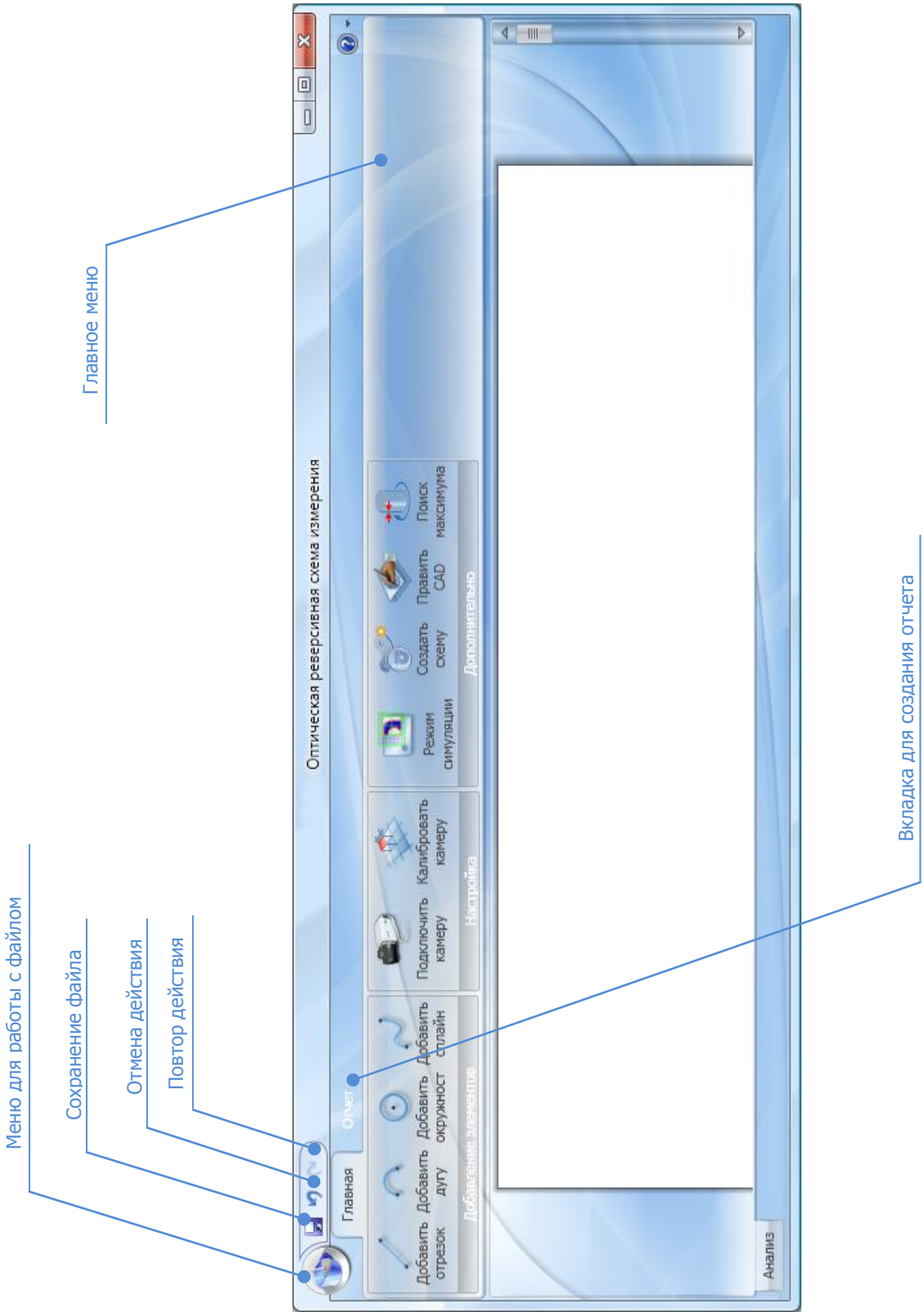


Рисунок 103. Окно редактора оптической схемы измерения

Измерение элемента

Чтобы начать измерение, нужно указать программе элемент для измерения. Для измерения доступны следующие элементы:

- отрезок;
- дуга;
- окружность;
- сплайн.

Указание элементов для измерения производится в главном меню в категории «Добавление элементов» (см. **Рисунок 104**)

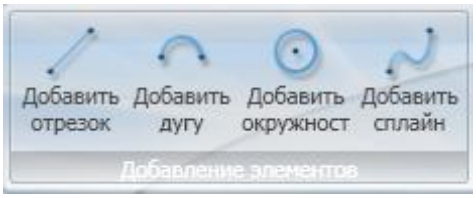


Рисунок 104. Меню "Добавление элементов"

Появляется стандартное для всех элементов окно измерения элемента (см. **Рисунок 105**). В верхней левой части окна измерения элементов отображается миникарта измеренных элементов, на которой красными точками обозначены измеренные элементы, а пунктиром – область видимости камеры. В правом верхнем углу окна имеется кнопка «Фокусировка» для перехода в режимы ручной или автоматической фокусировки и флажок «Обработка контура».

При установке флажка «Обработка контура» выводится векторный контур детали, который позволяет с более высокой повторяемостью спозиционировать машину на кромку, особенно при использовании функции приближения. Будьте внимательны, данную функцию рекомендуется включать на высокопроизводительных компьютерных системах, т.к. она требует больших вычислительных ресурсов

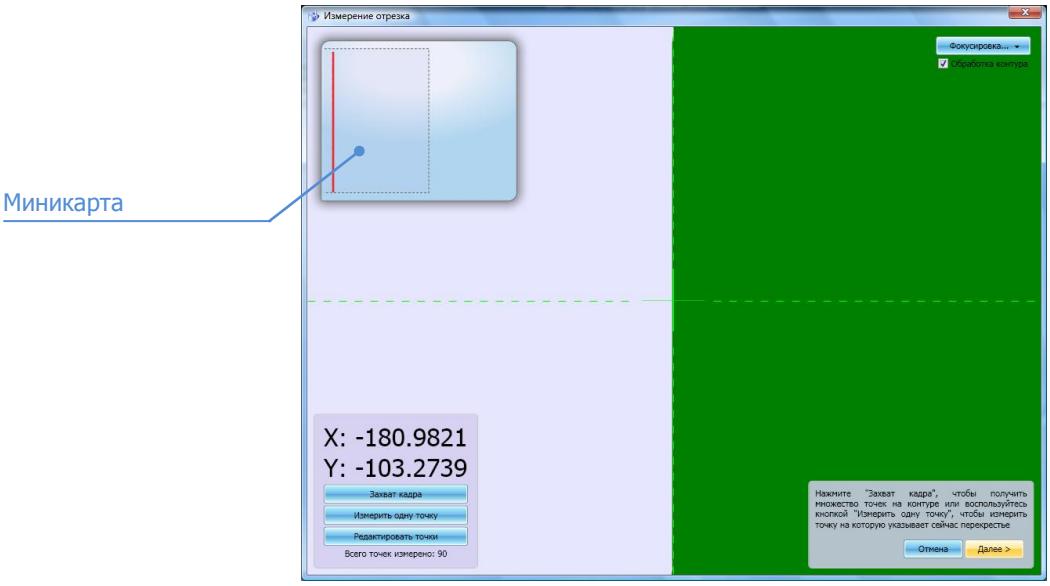


Рисунок 105. Окно измерения элемента

Затем необходимо навести камеру на измеряемый элемент, произвести захват кадра, щелкнув по кнопке «Захват кадра» в окне измерения элемента или нажать сочетание клавиш «Ctrl+Enter».

При захвате кадра производится стабилизация изображения и выделение точек для измерения на контуре детали. Стабилизация позволяет снизить уровень шума. В результате получится равномерное облако точек, находящихся на контуре детали. С помощью ластика необходимо удалить точки, не относящиеся к измеряемому элементу (см. **Рисунок 106**) .

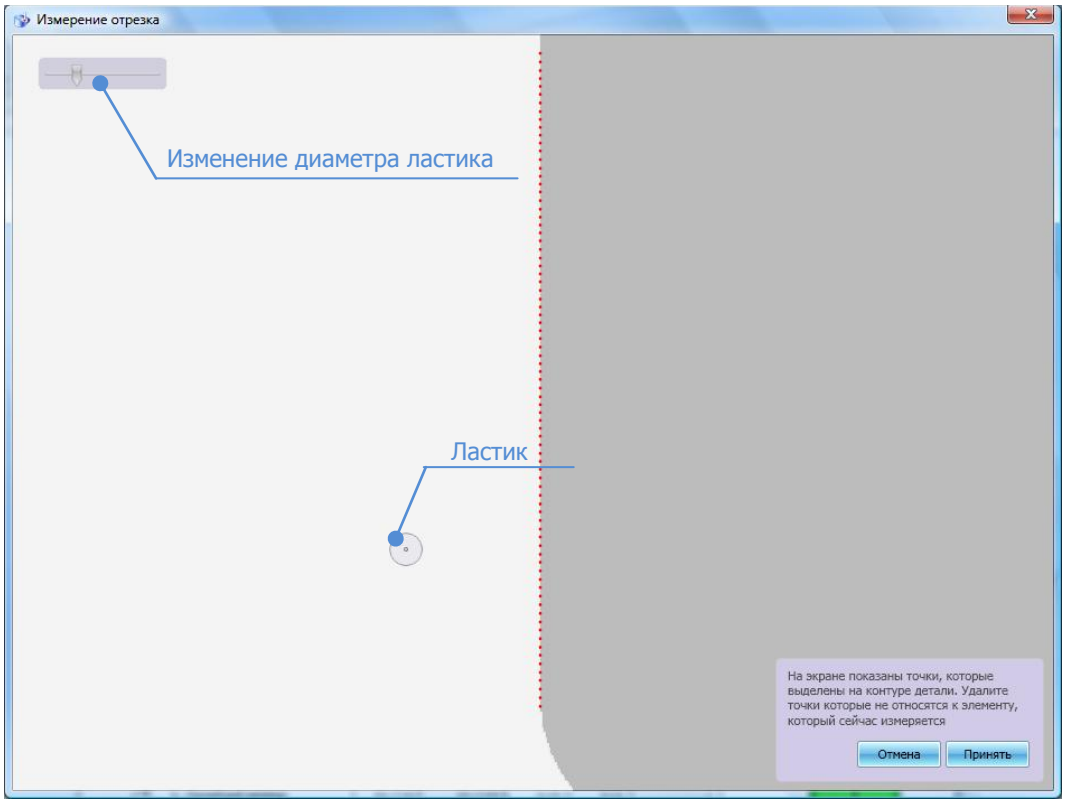


Рисунок 106. Редактирование контура

Существует возможность измерения произвольного контура вручную: путем указания точек для измерения. Для этого в окне измерения детали необходимо дважды щелкнуть мышью по измеряемой точке или навести перекрестие камеры на измеряемую точку и щелкнуть по кнопке «Измерить точку»(данное действие можно заменить сочетанием клавиш «Shift+Enter»).

Указание точки для измерения перекрестьем камеры является более точным, т.к. на измеряемую точку не оказывают влияния оптические искажения камеры.

Для более точного наведения перекрестия камеры можно масштабировать отображение детали в окне измерения элемента: клавиша «+» включает режим десятикратного увеличения изображения, клавиша «-» отключает данный режим. Следует помнить, что в данном случае происходит цифровое, а не оптическое, масштабирование изображения.

Измеренные элементы автоматически добавляются в отчет (о возможностях отчета см. главу «Координатный анализ геометрических параметров детали»). Полученные элементы не соединены между собой. В общем случае для расчетов соединение элементов детали необязательно. Но для получения корректного вида модели необходимо устранить разрывы.

Для правки CAD-модели детали существует специальный редактор, который можно вызвать по кнопке «Править CAD» в главном меню (см.**Рисунок 103**). Чтобы соединить два элемента, необходимо выбрать крайнюю точку одного элемента и подтянуть её к крайней точке другого элемента (см. **Рисунок 107**).

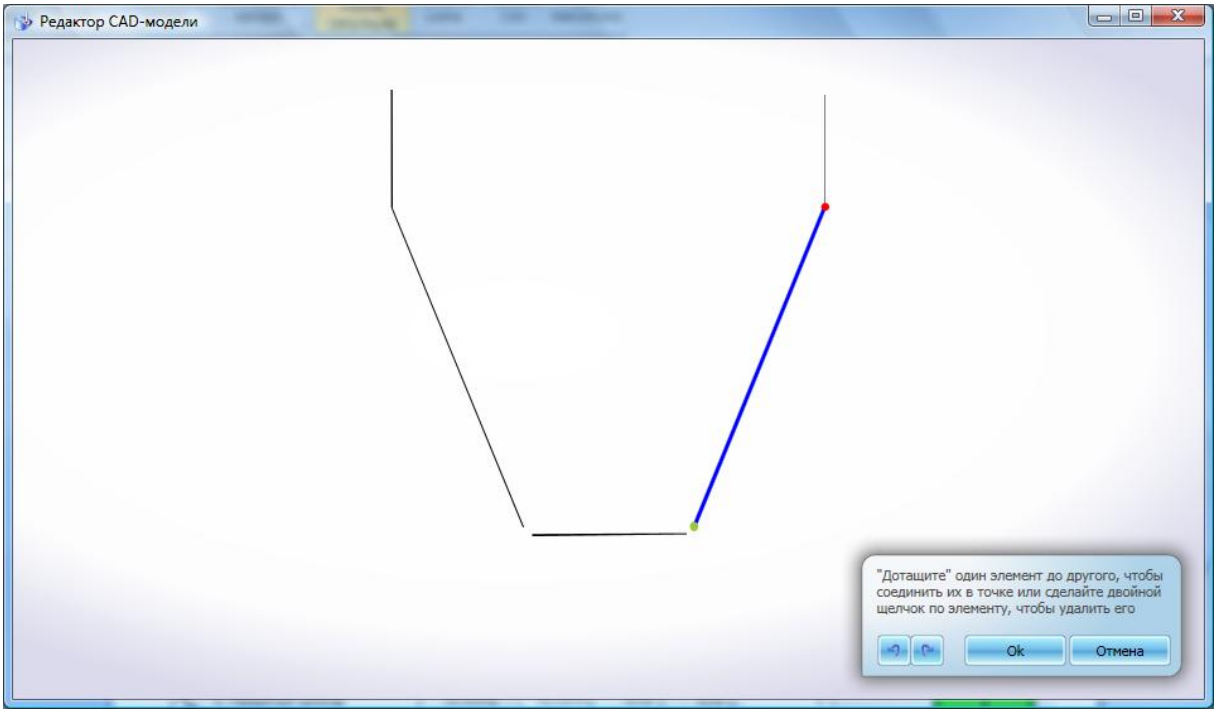


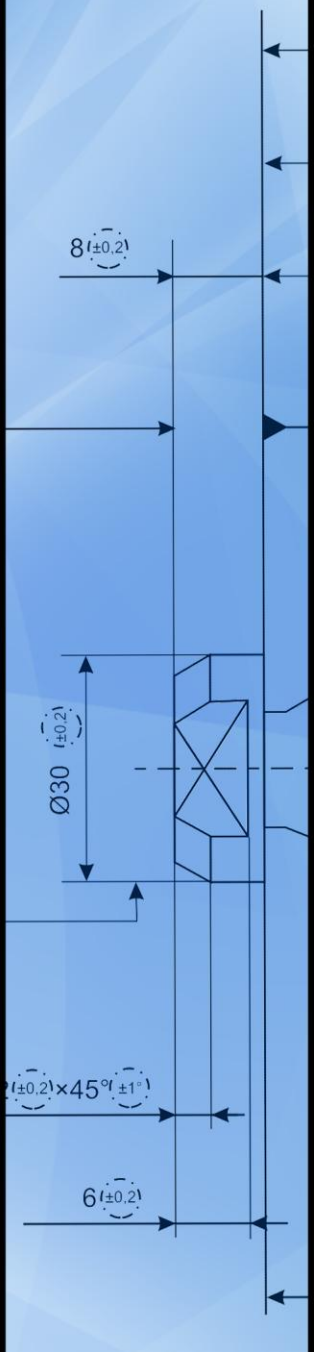
Рисунок 107. Окно редактора CAD-модели

В результате такого соединения параметры CAD-элемента могут быть искажены: например, может измениться радиус дуги. Тем не менее, на результаты измерения это не повлияет, т.к. расчеты производятся на основе координат измеренных точек.

Полученную CAD-модель можно использовать для стандартной схемы, которая позволяет производить измерения в автоматическом цикле. Чтобы передать модель в стандартную схему измерения необходимо щелкнуть по кнопке «Создать схему» в главном меню (см. **Рисунок 103**).

При передаче модели в стандартную схему измерения, необходимо привести модель к корректному виду, устранив разрывы.

На практике часто возникают ситуации, когда при измерении на просвет тел вращения необходимо найти радиальный максимум, т.е. расположить деталь определенным образом. Для решения этой задачи существует соответствующий инструмент на главной панели инструментов.



ТЕХНО КООРД™

ТЕХНОЛОГИЯ КООРДИНАТНЫХ ИЗМЕРЕНИЙ

Координатный анализ геометрических параметров детали

Оглавление

Общие сведения

Введение

Подсистема анализа предназначена для унифицированной обработки данных координатных измерений. Данная подсистема интегрирована в большинство программ **ТЕХНО**коорд™, что позволяет единообразно подходить к обработке данных координатных измерений полученных различными способами и на различных приборах (см. **Рисунок 108**).

Рисунок 108. Унификация анализа координатных измерений

Важно подчеркнуть, что данный инструмент позволяет унифицировать обработку данных измерения, только там, где используется координатный метод измерения, т.е. на выходе схемы измерения находится массив координат (а также метаданные, такие как CAD-модель). Саму же подсистему можно рассматривать как мощный инструмент для проведения сложных расчетов в удобном, интерактивном и простом виде.

В программном обеспечении **ТЕХНО**коорд™ можно как сначала провести измерения, а затем создать анализ, так и сначала создать анализ, а затем выполнить необходимые для него измерения.

Аппроксимация элементов – это нахождение заменяющих элементов по измеренным точкам на их поверхности. Существует несколько различных методов аппроксимации геометрического элемента. Ниже описаны три основных метода (нахождение среднего, прилегающего или минимальной зоны).

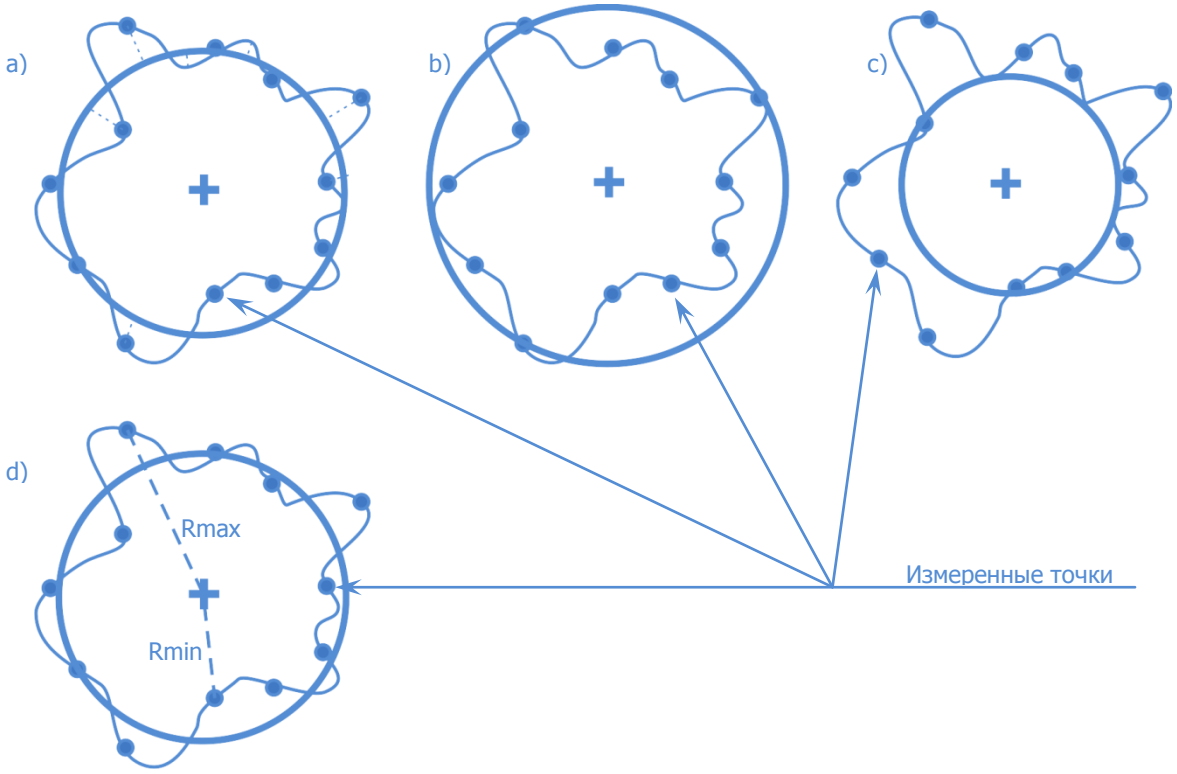


Рисунок 109. (методы аппроксимации в соответствии с ISO 6318: a) среднеквадратичный элемент b)c) минимально описанный и максимально вписанный элемент d)минимальная зона)

Средний элемент (аппроксимация среднеквадратичными кривыми и поверхностями (англ. least-square fitting)) – поверхность, имеющая номинальную форму и такие размеры и/или расположение, чтобы сумма квадратов расстояний между реальным и средним элементами в пределах нормируемого участка имела минимальное значение. Среднеквадратичная поверхность (кривая) является решением задачи минимизации суммы квадратов расстояний от измеренной точки до поверхности (кривой):

$$\sum_{i=1}^{n} {d_i}^2 \rightarrow min$$

где n – количество измеренных точек,
d_i – ортогональное расстояние от i-ой измеренной точки до искомой поверхности.

Прилегающей поверхностью (кривой) называется поверхность, имеющая форму номинальной поверхности, соприкасающаяся с реальной поверхностью (кривой) и расположенная вне материала детали так, что отклонение от нее наиболее удаленной точки реальной поверхности в пределах нормируемого участка имеет минимальное значение. Это понятие относится к прилегающей плоскости, прямой, конуса, однако указанное условие минимального значения отклонения не распространяется на отклонения формы цилиндра, сферы и окружности.

Данная задача есть задача минимизации нормы Чебышева:

$$sup_{i=1}^{n}|d_i| \rightarrow min$$

где d_i *- расстояние от измеренной точки до поверхности (кривой),*
n *– количество измеренных точек.*

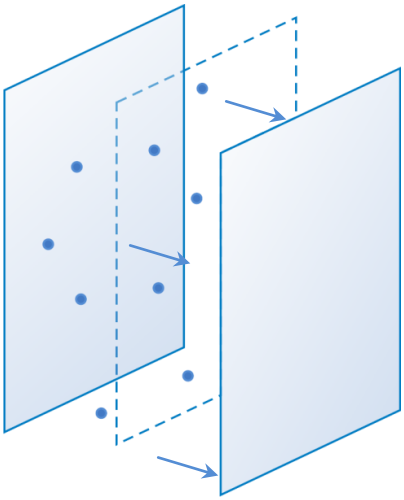


Рисунок 110. Прилегающие плоскости

Прилегающим цилиндром (сферой, окружностью) называется цилиндр (сфера, окружность) минимального диаметра, описанного вокруг реальной наружной поверхности, или максимального диаметра, вписанного в реальную внутреннюю поверхность (см. **Рисунок 111**).

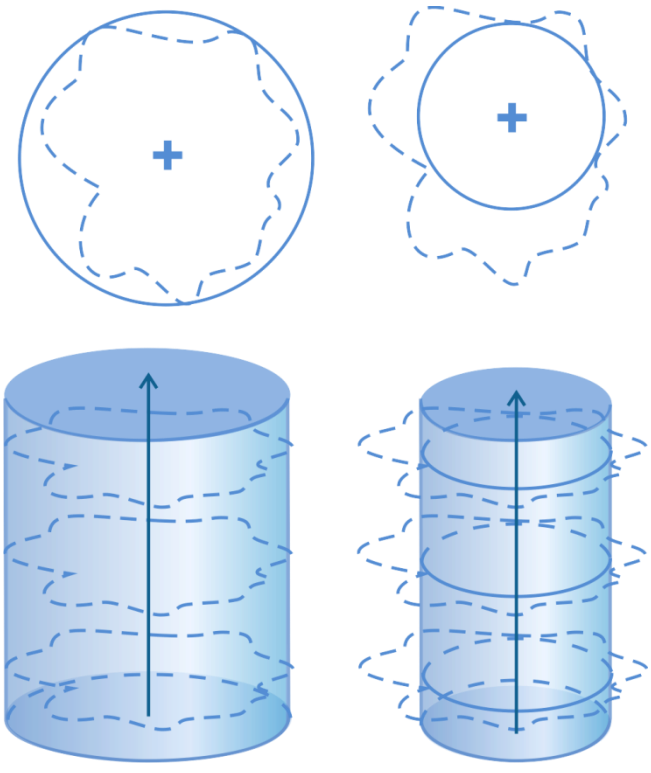


Рисунок 111. Прилегающие окружности и цилиндры

Замечание о различии в стандартах. Необходимо заметить, что российские стандарты вводят понятия прилегающих, как основных заменяющих элементов для расчетов, а также минимальные зоны и средние элементы (например, ГОСТ 24642-81), причем в случае цилиндра, сферы и окружности функции минимизации меняются между понятиями прилегающих и минимальных зон. Международные стандарты (ISO 6318, см. **Рисунок 112**) определяют более четко в математическом смысле заменяющие элементы: минимальная зона (отклонение наиболее удаленной точки имеет минимальное значение), максимально вписанного, минимально описанного элемента и среднеквадратичного элемента. **ТЕХНО**координ™ придерживается терминологии описанной в ГОСТ. Таким образом, прилегающим плоскостям, прямым, конусам соответствует понятие минимальной зоны, а прилегающим окружностям, цилиндрам, сферам соответствуют минимально описанные и максимально вписанные элементы. Окружности, сферы и цилиндры минимальной зоны, а также средние элементы не имеют различий в стандартах.

Вместо прилегающего цилиндра (окруности, сферы) в качестве базы для определения отклонений допускается также использовать цилиндр минимальной зоны. Цилиндр (окружность, сфера) минимальной зоны – цилиндр (конус, сфера, тор), соприкасающийся с реальной поверхностью и расположенный вне материала так, чтобы наибольшее расстояние между реальной поверхностью и заменяющим элементом имело минимальное значение (см. **Рисунок 112**).

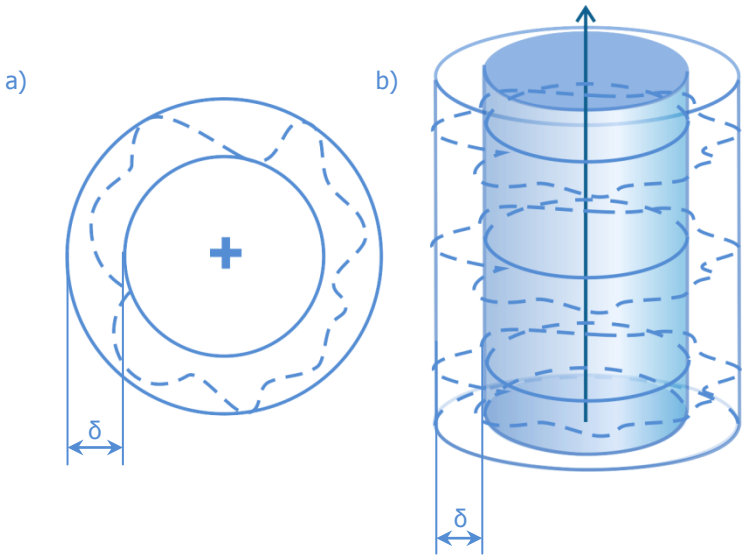


Рисунок 112. a - окружности минимальной зоны, b - цилиндры минимальной зоны

Редактор анализа

Редактор анализа интегрирован во все схемы измерения программного обеспечения **ТЕХНО**коорд™. Независимо от конкретного приложения интерфейс остается подобным.

Данные в координатном анализе можно сохранить в формате *.measure и *.analysis. В файл с расширением *.measure сохраняются только рассчитанные данные, которые можно применять для других анализов (кнопка «Сохранить результат»). Данные в файле *.analysis представлены наглядно с сохранением всей расчетной информации (кнопка «Сохранить анализ», см. **Рисунок 114**.)

В верхней части окна расположено главное меню (см. **Рисунок 114**). Через главное меню можно добавлять отдельные *элементы анализа*. Под элементом анализа здесь понимается небольшой редактор встраиваемый в данный. Чтобы лучше понять структуру редактора необходимо изучить рисунок ниже (см. **Рисунок 113**).

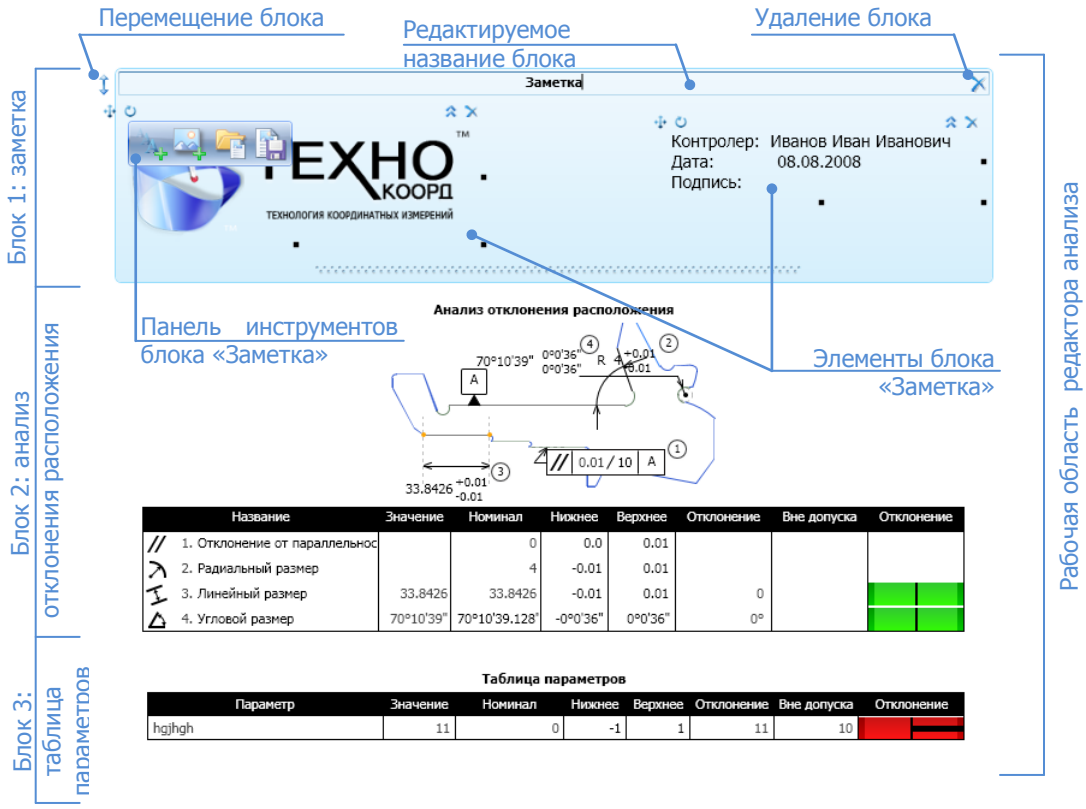


Рисунок 113. Структура редактора анализа

Таким образом, редактор анализа состоит из отдельных блоков (мини-редакторов). Отдельные блоки можно удалять, менять местами, редактировать название, используя кнопки в верхней части каждого блока в момент, когда он выделен.

Рисунок 114. Меню координатного анализа

Отчетные возможности

Расчетная программа

Подсистема анализа позволяет производить расчет достаточно сложных параметров, но данные параметры являются стандартными (ГОСТ 24642-81 и др.). Часто необходимо провести координатный расчет нестандартизованных параметров. Для данной цели существует расчетная программа.

Расчетная программа интегрирована в расширенную контактную схему измерения и в стандартную оптическую схему измерения, где рассчитанные параметры можно передать в анализ для их последующей репрезентации.

Для того чтобы рассчитанный параметр передать в анализ необходимо использовать конструкцию вида `Report.Add("Название параметра", x)`, где x – это числовое значение параметра (или переменная хранящее это значение). Затем в таблице параметров в отчете можно будет выбрать нужный параметр.

Табличная форма

Данный элемент анализа предназначен для репрезентации параметров в виде таблицы. Кроме вычисленного значения имеется возможность установить допуски, показать отклонение численно и визуально. Чтобы добавить данный элемент необходимо щелкнуть в главном меню «Таблица параметров» (см. **Рисунок 115**). После чего в редакторе анализа появится шапка таблицы.

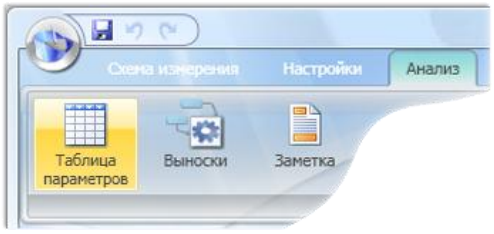


Рисунок 115. Главное меню редактора анализа

В таблице параметров отклонение может задано в предельной или номинальной форме, что можно выбрать из выпадающего меню в верхнем правом углу блока (см. **Рисунок 116**). Если выбрано предельное задание отклонения, то в таблице будут отображены столбцы «Нижнее» и «Верхнее» (допустимое отклонение). При номинальном задании отклонения в таблицу добавляется столбец «Номинал». Здесь же можно поставить галочку для отображения в таблице визуального вида отклонения (см. **Рисунок 119**).

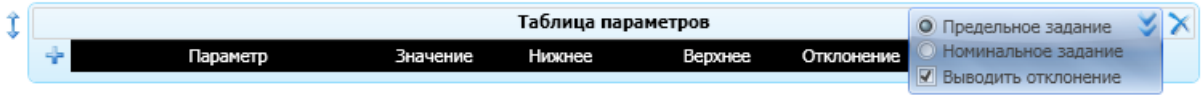


Рисунок 116. Выбор вида таблицы

Чтобы добавить параметр, который был выведен из расчетной программы нужно нажать на значок «плюс» в углу таблицы и выбрать нужный параметр (см.**Рисунок 117**).

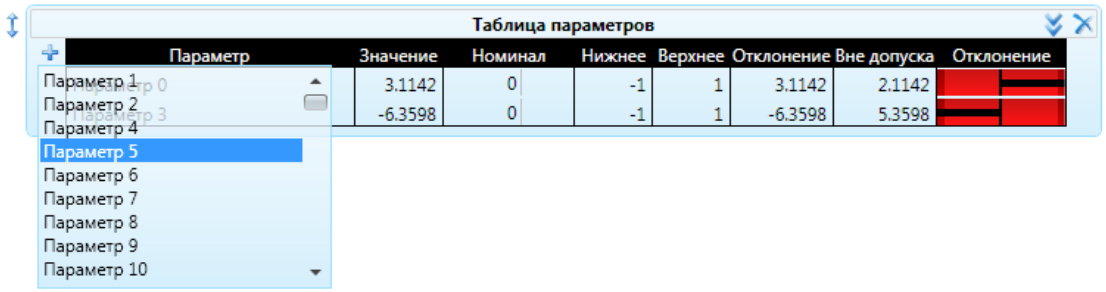


Рисунок 117. Добавление нового параметра в таблицу

После добавления параметра в таблицу рассчитанное значение отобразится автоматически. Пользователю остается заполнить значения в графах: «Номинал», «Нижнее» и «Верхнее» (допустимое отклонение). Чтобы удалить параметр из таблицы, необходимо нажать кнопку «Удалить» справа от параметра (см. **Рисунок 118**).

Необходимо заметить, что графа «Номинал» разделена на два поля: первое - предназначено для ввода значения номинала, а второе - для ввода квалитета (т.е. h15, s7 и т.д.). Таким образом, если заполнено значение квалитета, то значения полей «Нижнее» и «Верхнее» устанавливаются автоматически.

На рисунке ниже показаны две таблицы: верхняя находится в режиме редактирования (т.е. пользователь изменяет значение квалитета), нижняя находится в обычном состоянии (в таком виде она и будет выведена на печать).

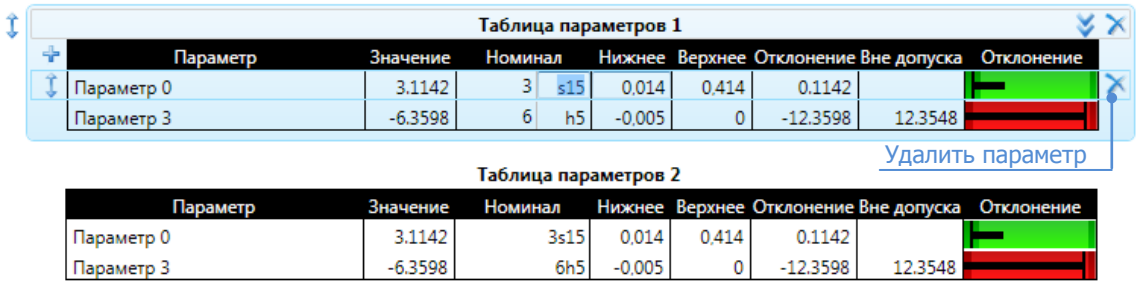


Рисунок 118. Использование квалитетов в табличной форме

Полезной особенностью данной таблицы является визуальное отображение отклонения. Оно позволяет быстро интерпретировать полученные результаты (см. **Рисунок 119**). Если отклонение не выходит за пределы допуска, то элемент принимает зеленый вид, иначе - красный.

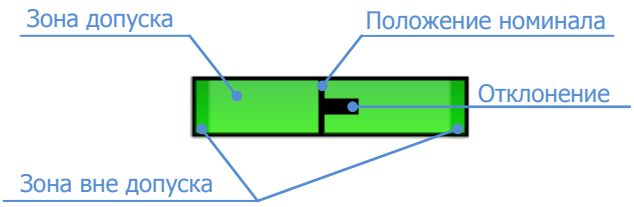


Рисунок 119. Визуализация отклонения

В некоторых случаях измеренное значение находится на границе допускаемого диапазона, а если принять во внимание тот факт, что прибор имеет определенную погрешность, то значение, которое показано допустимым цветом (зеленым), вызывает

сомнение. Для уточнения подобных ситуаций вводится величина «погрешность прибора». Чтобы ее заполнить, нужно щелкнуть на визуализацию отклонения и ввести числовое значение погрешности прибора в миллиметрах. После этого, значения, получающиеся на границе, будут отмечены желтым цветом (см. **Рисунок 120**).

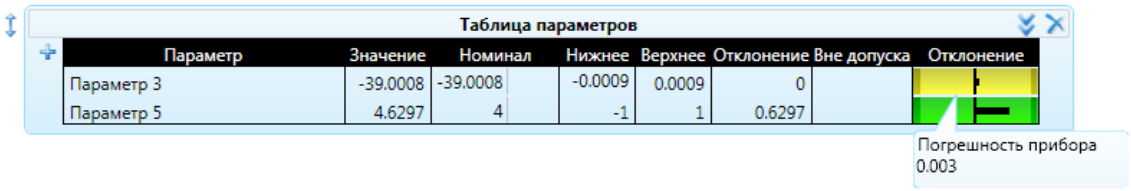


Рисунок 120. Ввод погрешности прибора

Если после добавления параметра в таблицу в расчетной программе поменять название параметра, то данный параметр пропадет из таблицы.

Эскиз с выносками

Данный элемент анализа дает возможность отобразить значение, вычисленное в расчетной программе, и указать стрелками на элемент (или место), которому соответствует данный параметр. Данный тип репрезентации особенно полезен в случае необходимости сопоставить измеренные параметры с определенными элементами детали. Кроме вычисленного значения имеется возможность установить допуски, показать отклонение численно и визуально.

Чтобы добавить данный элемент необходимо щелкнуть в главном меню «Выноски» (см. **Рисунок 121**). После чего в редакторе анализа появится новый раздел с эскизом по умолчанию.

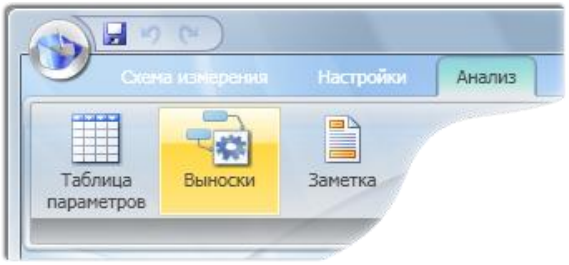


Рисунок 121. Главное меню редактора анализа

Эскиз – это изображение, схематично отображающее измеряемую деталь. Создать эскиз можно, используя графический редактор, или, например, взяв «скриншот» с окна CAD редактора. Полученное изображение необходимо сохранить в файл. Поддерживаемые форматы файлов включают:

- *.PNG (Portable Network Graphics);
- *.JPEG (Joint Photographic Experts Group);
- *.GIF (Graphics Interchange Format);
- *.BMP (Bitmapped Graphics Format);
- *.TIFF (Tagged Image File Format).

Обратите внимание, что поддерживается формат PNG с альфа каналом, т.е. допустимы изображения с прозрачностью.

Чтобы загрузить изображение, необходимо нажать кнопку «Загрузить изображение» на панели инструментов данного элемента анализа. Добавленный эскиз можно перемещать левой кнопкой мыши, а также масштабировать при помощи колеса прокрутки мыши или регулятора масштаба отображения (см. **Рисунок 122**).

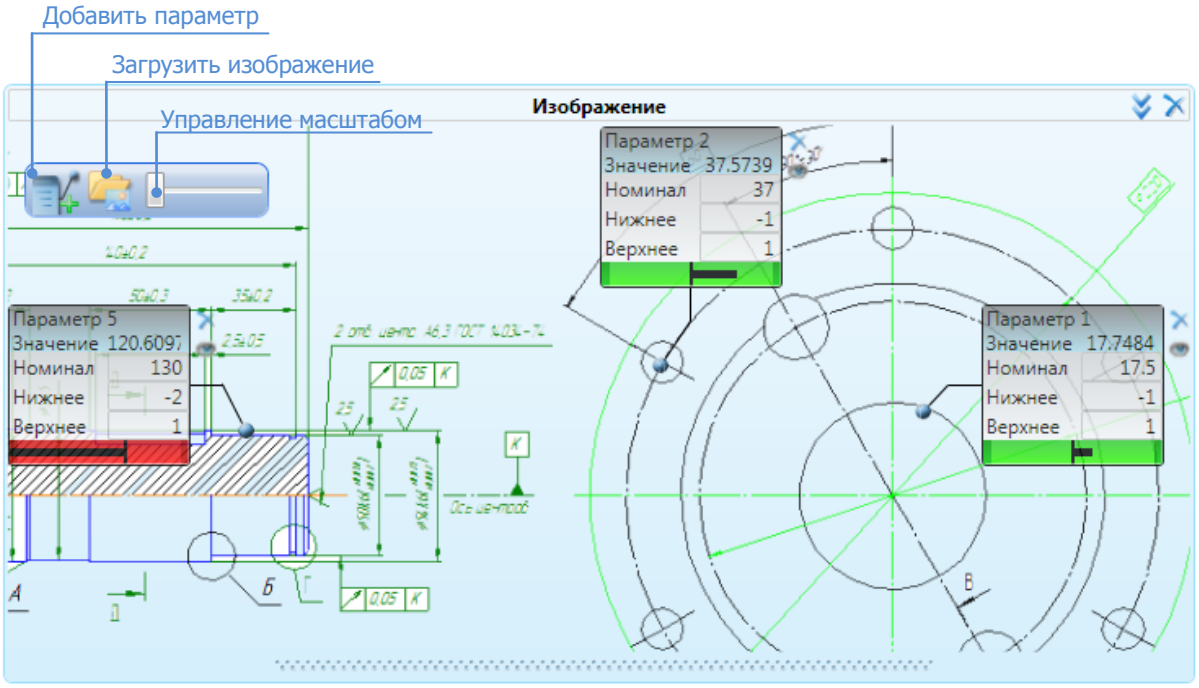


Рисунок 122. Внешний вид элемента с выносками (редактирование)

Чтобы добавить параметр необходимо щелкнуть по кнопке «Добавить параметр». (см. **Рисунок 122**). Рассчитанный параметр в данном редакторе представлен в виде выноски, т.е. небольшой таблицы и указателя (ломаной, которая указывает на точку на изображении). Указатель можно перемещать по изображению левой клавишей мыши. Чтобы удалить выноску, надо щелкнуть по кнопке «Удалить выноску» рядом с выноской. Также можно скрыть или отобразить указатель выноски, щелкнув по кнопке «Отобразить/скрыть» рядом с выноской (см. **Рисунок 123**).

В выносках отклонение может задано в предельной или номинальной форме, что можно выбрать из выпадающего меню в верхнем правом углу блока. Если выбрано предельное задание отклонения, то в выноске будут отображены строки «Нижнее» и «Верхнее» (допустимое отклонение). При номинальном задании отклонения в таблицу добавляется столбец «Номинал». Здесь же можно поставить галочку для отображения в выноске визуального вида отклонения (см. **Рисунок 123**).

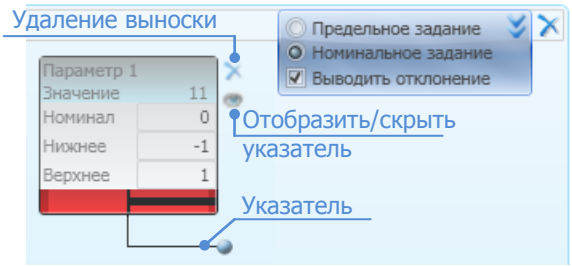


Рисунок 123. Оформление выноски

В выноске можно изменить нижнее и верхнее допустимое отклонение, но в таблице параметров эти изменения не отобразятся.

Анализ расположения в плоскости

Введение

Практически во всем мире используются семь параметров для нормирования точности расположения элементов детали:

- отклонение от параллельности;
- отклонение от перпендикулярности;
- отклонение наклона;
- позиционное отклонение;
- отклонение от соосности;
- отклонение от симметричности;
- отклонение от пересечения осей;

Данный анализ работает с координатными данными, измерение которых было произведено в двухмерном пространстве. В большей степени по этой причине данный анализ не позволяет производить расчет отклонений соосности, симметричности и пересечения осей.

В программном обеспечении **ТЕХНО**коорд™ анализ расположения в плоскости включен в стандартную и реверсивную оптические схемы измерения.

Поверхности, из которых состоит деталь, не могут быть идеальными и часто в значительной мере искажены. Поэтому возникают различные варианты, как нормировать и измерять расположение поверхностей, которые имеют искаженную форму. В программном обеспечении **ТЕХНО**коорд™ расчеты ведутся с использованием заменяющих элементов: средних элементов, элементов минимальной зоны, прилегающих элементов.

После создания анализа расположения появится геометрия детали и полупрозрачная панель инструментов в левом верхнем углу отчета (см. **Рисунок 124**).

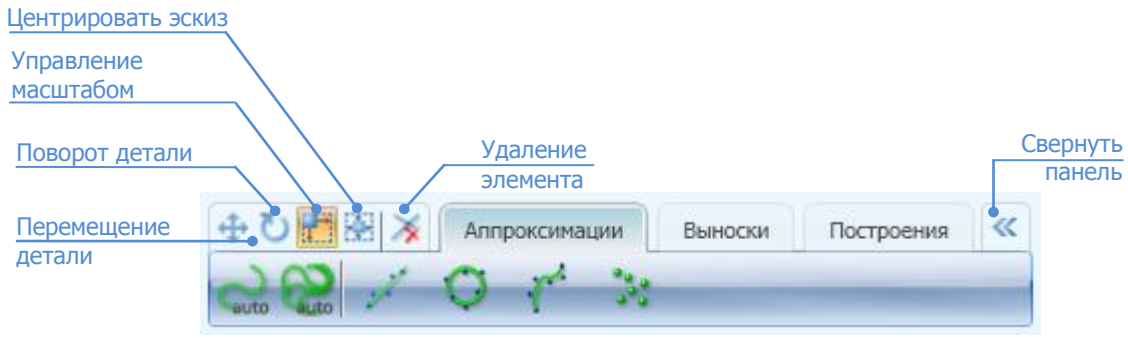


Рисунок 124. Панель инструментов, управление видом

Чтобы активизировать панель инструментов, необходимо навести на нее указатель мыши. Для каждой кнопки панели предусмотрены подсказки. Чтобы вызвать подсказку необходимо навести и удерживать курсор мыши на кнопке.

Чтобы скрыть панель инструментов можно воспользоваться соответствующей кнопкой на панели инструментов или нажать клавишу «Shift»

После этого можно расположить деталь правильным образом и изменить размеры элемента анализа. Для ориентации детали существуют специальные кнопки на панели инструментов (см. **Рисунок 124**).

Масштабирование детали можно осуществлять с помощью колесика мыши, а перемещение – средней клавишей мыши.

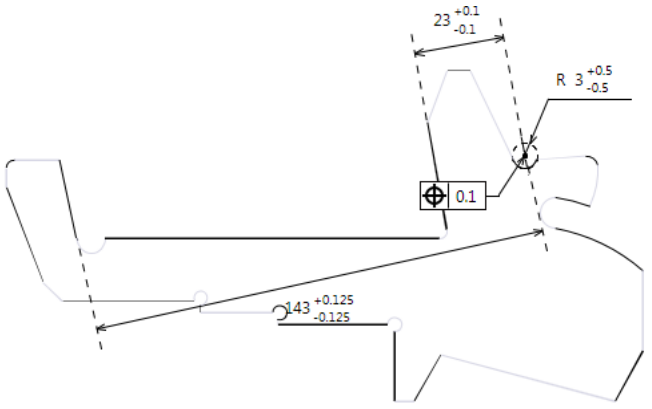
Принцип работы

Данный редактор дает возможность выносить размеры, указывать допуски, а также производить дополнительные построения, т.е. достраивать новые элементы по определенным законам (например, точка пересечения двух прямых или центр окружности).

При указании размеров и допусков в таблицу автоматически добавляется строчка, где есть рассчитанное значение и можно заполнить поля: «Номинал», «Нижнее» и «Верхнее» (допустимое отклонение). В случае если данный геометрический элемент не был измерен, то он отображается серым цветом, а в таблице отображается пустая строчка.

В программном обеспечении **ТЕХНО**координ™ при составлении схемы измерения значения рассчитываются с использованием заменяющих элементов: средних элементов, элементов минимальной зоны, прилегающих элементов. Поэтому прежде, чем указывать допуски и производить дополнительные построения, необходимо аппроксимировать элементы, участвующие в расчетах.

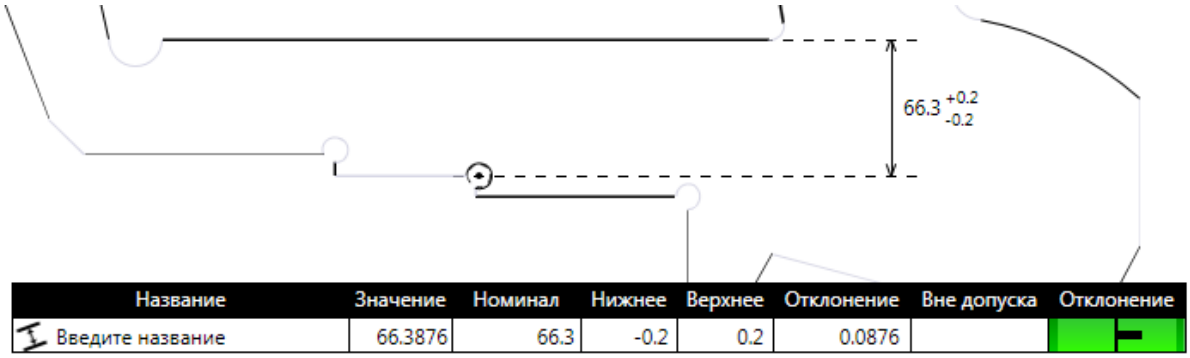
В общем случае требуется расставить размерные параметры (радиальные, линейные и т.д.), в случае необходимости произвести дополнительные построения, затем указать допуски (см. **Рисунок 125**).



Название	Значение	Номинал	Нижнее	Верхнее	Отклонение	Вне допуска	Отклонение
Введите название	3.2096	3	-0.5	0.5	0.2096		
Введите название	23.0876	23	-0.1	0.1	0.0876		
Введите название	0.0935	0	0.0	0.1	0.0935		
Введите название	143.0935	143js11	-0,125	0,125	0.0935		

Рисунок 125. Пример указания позиционного допуска

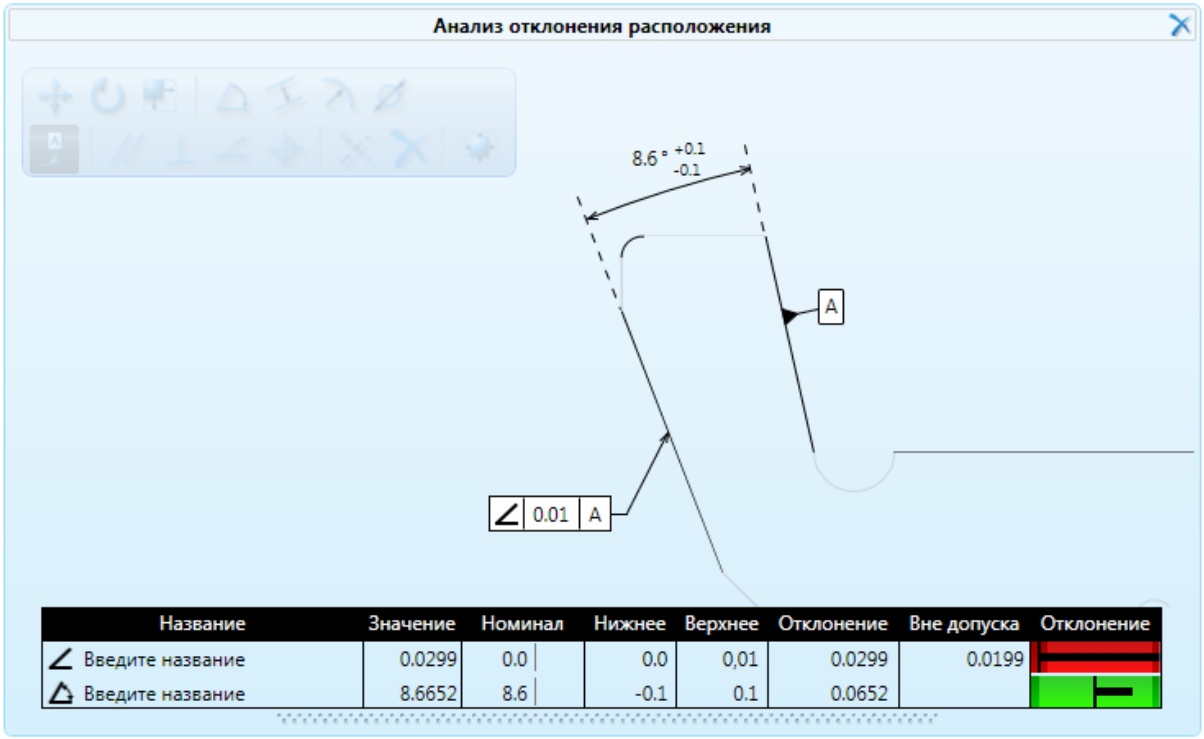
Добавленные размеры можно рассматривать отдельно от допусков и указывать для них номиналы, нижние и верхние допустимые отклонения (см. **Рисунок 126**).



Название	Значение	Номинал	Нижнее	Верхнее	Отклонение	Вне допуска	Отклонение
Введите название	66.3876	66.3	-0.2	0.2	0.0876		

Рисунок 126. Пример расчета расстояния от линии до центра окружности

Некоторые допуски работают только при наличии определенных указанных размеров, например, допуск наклона, позиционный допуск (см. **Рисунок 127**). Если соответствующих размеров недостаточно, то строчка в таблице останется пустой.



Название	Значение	Номинал	Нижнее	Верхнее	Отклонение	Вне допуска	Отклонение
Введите название	0.0299	0.0	0.0	0,01	0.0299	0.0199	
Введите название	8.6652	8.6	-0.1	0.1	0.0652		

Рисунок 127. Пример указания допуска на наклон

Аппроксимация элементов детали

В программном обеспечении **ТЕХНО**координ™ при составлении схемы измерения значения рассчитываются с использованием заменяющих элементов: средних элементов, элементов минимальной зоны, прилегающих элементов. Поэтому прежде, чем указывать допуски и производить дополнительные построения, необходимо аппроксимировать элементы, участвующие в расчетах.

Инструменты для аппроксимации расположены на соответствующей вкладке на панели инструментов (см. **Рисунок 128**).

На панели инструментов представлены инструменты двух типов: инструменты автоматической аппроксимации, которые автоматически находят подходящий заменяющий элемент, и инструменты для аппроксимации элементов детали вручную, включающие построение средних, прилегающих (вписанных и описанных) заменяющих элементов или элементов (вписанных и описанных) минимальной зоны для прямой, окружности, сплайна.

Инструмент «Точки» позволяет создать облако точек вместо аппроксимируемого элемента. Полученные точки можно использовать для нестандартизированных расчетов и построений в расчетной программе (подробнее см. главу «Язык для математической обработки данных координатного анализа»).

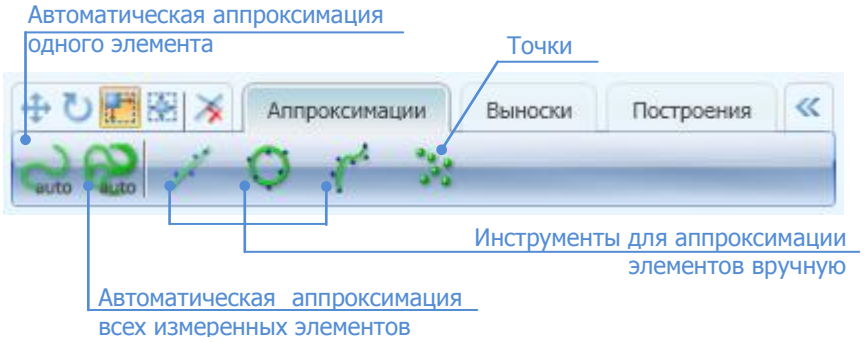


Рисунок 128. Панель аппроксимации элементов детали

Указание допусков

Чтобы указать допуск в редакторе, необходимо выбрать соответствующий инструмент на вкладке «Выноски» на панели инструментов (см. **Рисунок 129**) и щелчком мыши указать аппроксимированный элемент геометрии в эскизе.

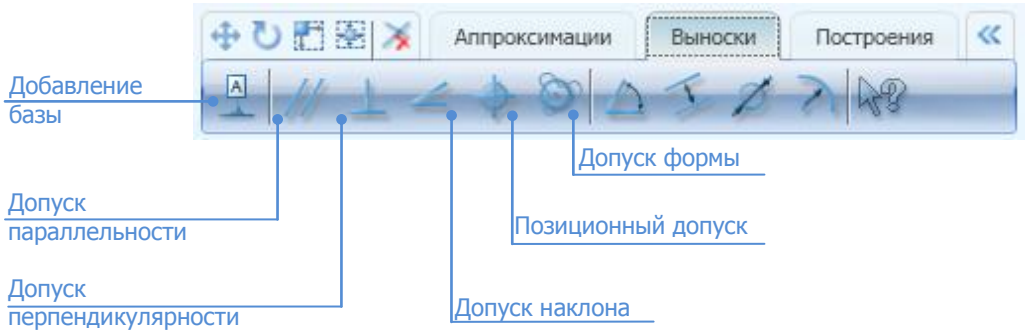


Рисунок 129. Панель инструментов, указание допусков

Указание базы

Необходимо определить элементы детали, по отношению к которым задаются допуски, т.е. базовые элементы или база.

Чтобы указать базу для определенного элемента необходимо:

- выбрать соответствующий инструмент на панели инструментов (см. **Рисунок 129**);
- навести мышь на геометрический элемент, в этот момент он подсветится и можно будет увидеть, где он начинается и заканчивается;
- нажать левую клавишу мыши и оттащить в сторону, затем отпустить. В месте, где будет отпущена мышь, появится значок базы.

После установки базу можно перемещать, удерживая левую клавишу мыши.

Чтобы удалить базу или отредактировать название нужно дважды щелкнуть мышью по выноске.

Базу можно перемещать с одного геометрического элемента на другой, для этого необходимо нажать мышью на элементе, к которому относится база, и «дотащить» до другого.

Допуск параллельности

Данный анализ позволяет нормировать отклонение от параллельности между двухмерными прямыми (отрезками).

- Навести мышь на прямую, нажать левую клавишу мыши;
- Оттащить мышь в сторону, где предполагается разместить выноску;
- Справа из списка выбрать базу, от которой планируется нормировать;
- Заполнить параметры в таблице ниже.

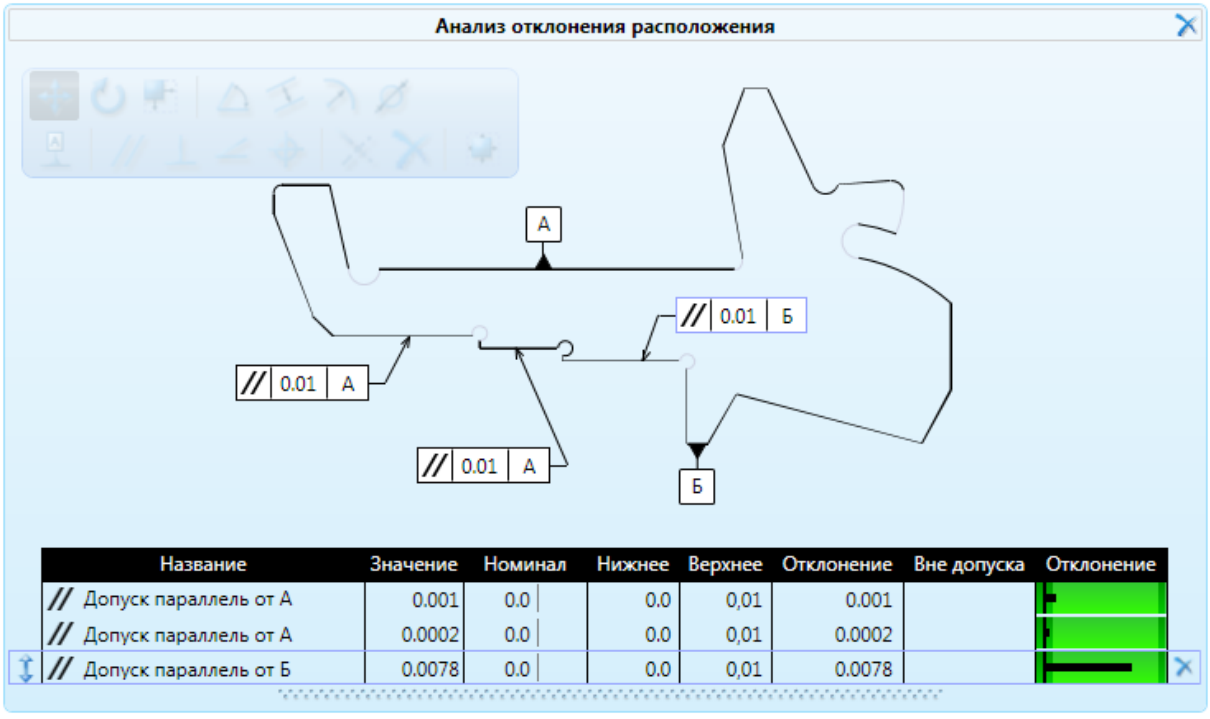


Рисунок 130. Пример указания допуска параллельности

Допуск перпендикулярности

Аналогично допуску параллельности можно установить допуск перпендикулярности. Также как в случае с допуском параллельности в качестве базового элемента и нормируемого элемента допустимо брать прямые (отрезки) см. **Рисунок 131**.

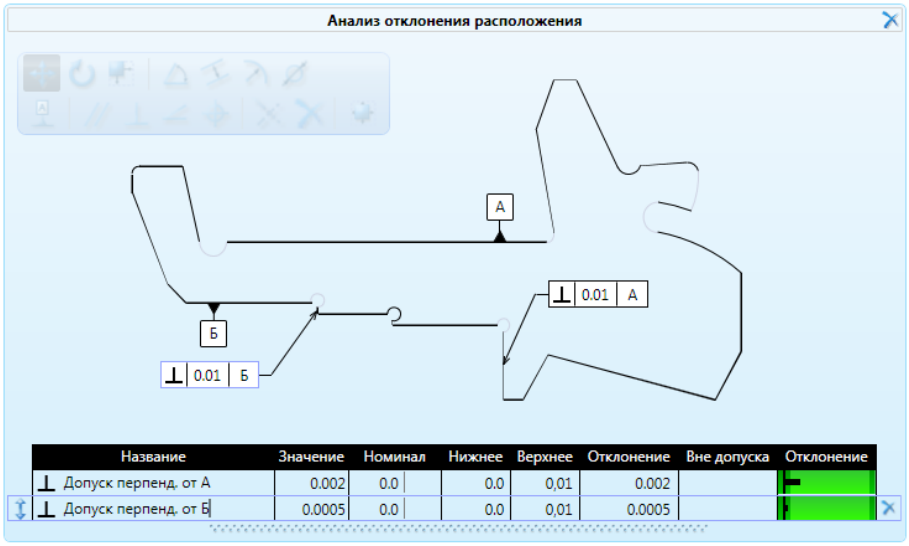


Рисунок 131. Пример указания допуска перпендикулярности

Допуск наклона

Аналогично допуску параллельности можно установить допуск наклона. Также как в случае с допуском параллельности в качестве базового элемента и нормируемого элемента допустимо брать прямые (отрезки). Но в отличие от допуска параллельности необходимо нанести угловой размер с указанием номинала. В случае если такой размер не указан, расчет отклонения наклона произведен не будет и в таблице будет частично пустая строка (см. **Рисунок 132**).

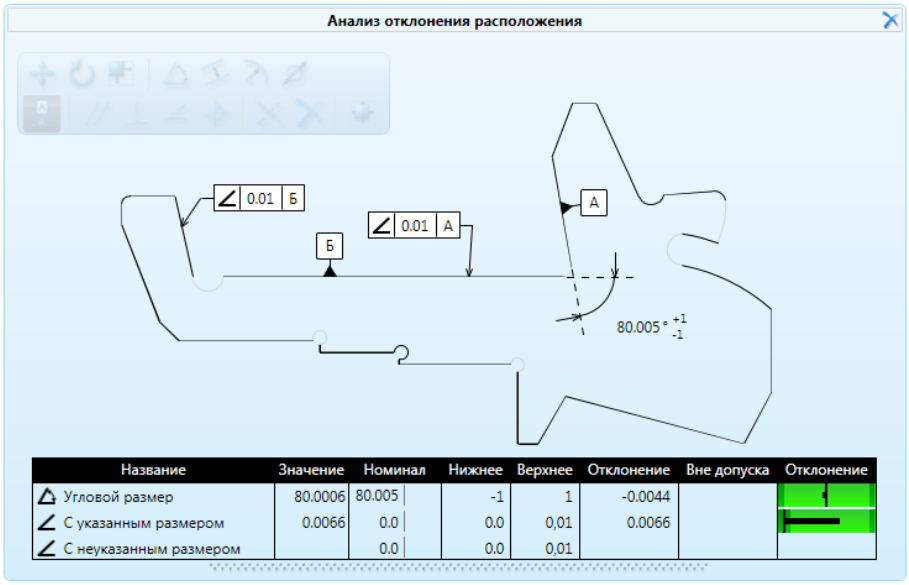
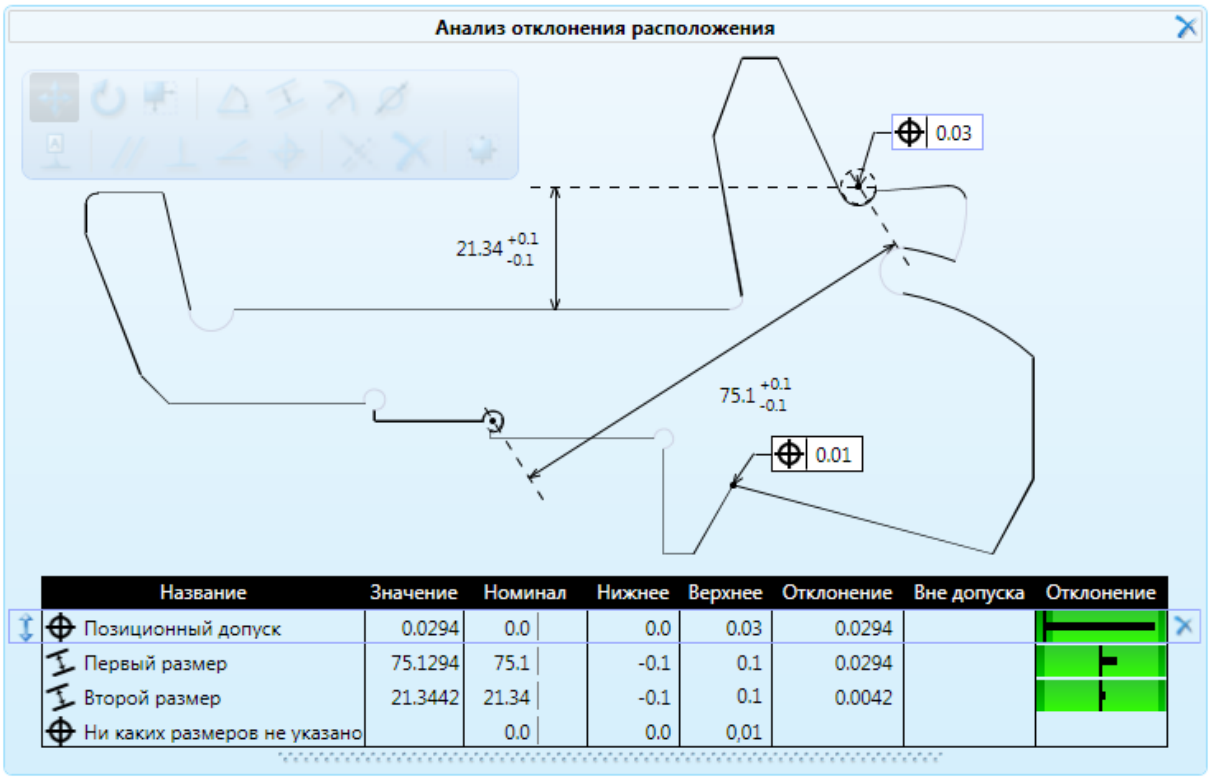


Рисунок 132. Пример указания допуска наклона

Позиционный допуск

Позиционный допуск в общем случае – это наибольшее расстояние между реальным расположением элемента делали и его номинальным расположением. Данный анализ позволяет задавать допуск расположения, привязывая его к определенному геометрическому элементу. В качестве таких элементов могут выступать точки (полученные, например, путем пересечения двух прямых) или центры окружностей (дуг).

Каким же образом рассчитывается позиционное отклонение? Если просто указать позиционный допуск для определенной точки (окружности, дуги), то в таблицу добавится частично пустая строка. Это говорит о том, что с данным геометрическим элементом не связано ни каких размерных параметров. Если добавить хотя бы один размер от данной точки (окружности, дуги), то строка заполнится. Иными словами, позиционное отклонение будет рассчитано исходя из указанных номинальных размеров.



Название	Значение	Номинал	Нижнее	Верхнее	Отклонение	Вне допуска	Отклонение
Позиционный допуск	0.0294	0.0	0.0	0.03	0.0294		
Первый размер	75.1294	75.1	-0.1	0.1	0.0294		
Второй размер	21.3442	21.34	-0.1	0.1	0.0042		
Ни каких размеров не указано		0.0	0.0	0,01			

Рисунок 133. Пример указания позиционного допуска

На рисунке выше показано, что в случае второго допуска строка осталась пустой, т.к. связанные с точкой размеры отсутствуют.

Указание размеров

Наиболее часто возникает необходимость указать размер элемента. В программном обеспечении **ТЕХНО**координ™ предусмотрена возможность вынесения углового, линейного, радиального и углового размеров.

Чтобы добавить размер в редакторе, необходимо выбрать соответствующий инструмент (см. **Рисунок 134**) на вкладке «Выноски» и щелкнуть на элемент геометрии.

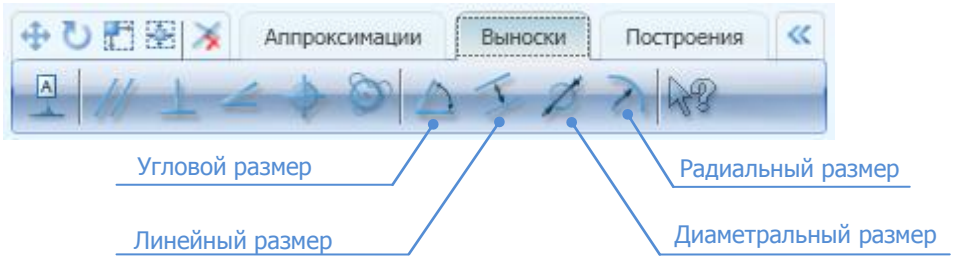


Рисунок 134. Отклонения расположения, панель инструментов, размерные параметры

Построенные выноски можно перемещать мышью.

Ниже подробно описаны размеры, которые можно указать, и способы их вынесения.

Линейные размеры

Линейный размер можно рассчитать как:

- Расстояние от точки до линии;
- Расстояние от точки до ближайшей точки окружности;
- Расстояние от точки до ближайшей точки дуги;
- Расстояние от образующей дуги до образующей окружности;
- Расстояние от образующей окружности до образующей окружности;
- Расстояние от образующей дуги до образующей дуги;
- Расстояние от точки до точки;

Если подходящих элементов нет, можно воспользоваться дополнительным построением.

Чтобы установить линейный размер, необходимо выбрать соответствующий инструмент на панели инструментов (см. **Рисунок 134**), после чего щелкнуть на геометрический элемент и, удерживая кнопку мыши, дотащить указатель до другого элемента. Если данная комбинация допустима, то оба элемента подсветятся, и появится размерная линия с номинальным размером и допусками. В таблице, расположенной ниже, следует ввести номинальное значение и допуск в соответствии с чертежом. Чтобы узнать к какой строчке относится тот или иной размер следует навести на него указатель мыши, при этом в таблице будет выделена соответствующая строчка.

На рисунке ниже (см. **Рисунок 135**) показаны примеры линейных размеров между различными элементами. В частности для двух размеров использовалось дополнительное построение: пересечение двух прямых.

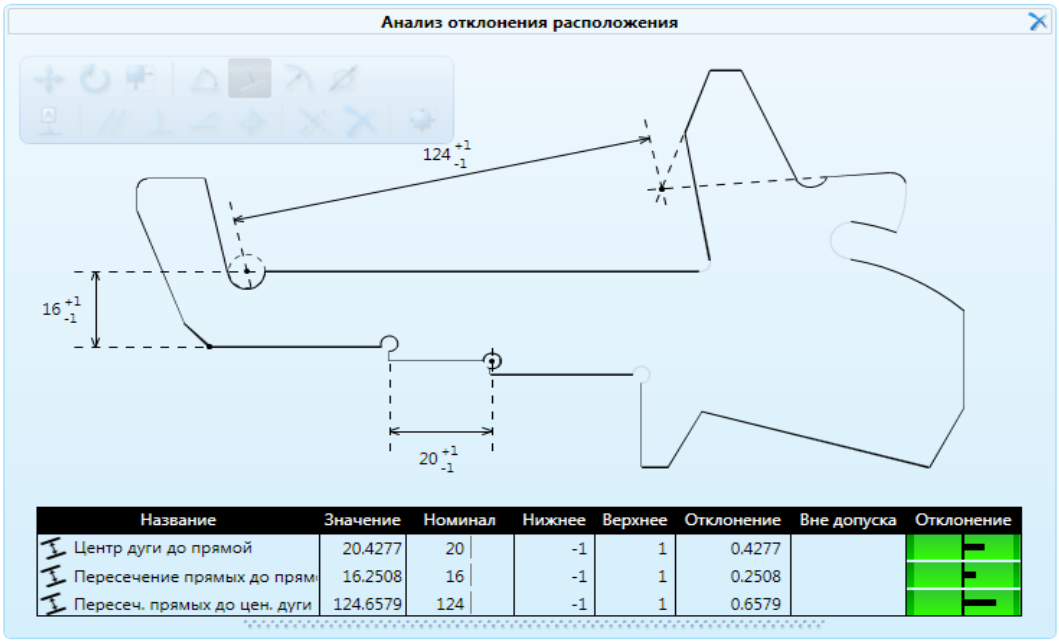


Рисунок 135. Пример установки линейных размеров

Радиальные размеры

Устанавливать радиальные размеры возможно для окружностей и дуг (см. **Рисунок 136**). Чтобы это сделать, необходимо выбрать соответствующий инструмент на панели инструментов (см. **Рисунок 134**), после чего щелкнуть на геометрический элемент и, удерживая кнопку мыши, оттащить указатель в другую точку. В точке, где будет отпущена мышь, появится выносная линия с номинальным радиальным размером и допусками. В таблице, расположенной ниже, следует ввести номинальное значение и допуск в соответствии с чертежом. Чтобы узнать к какой строчке относится тот или иной размер следует навести на него мышь, а в таблице будет выделена соответствующая строчка (см. **Рисунок 136**).

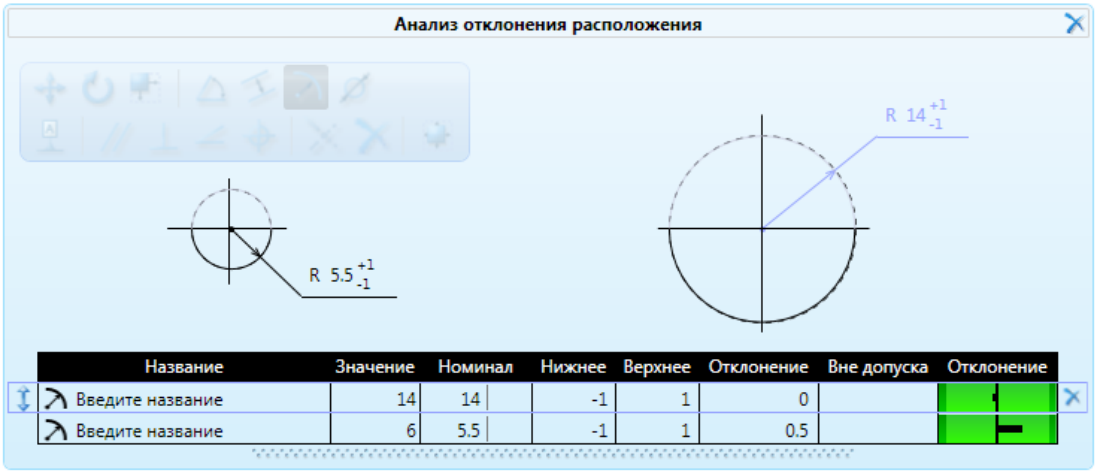
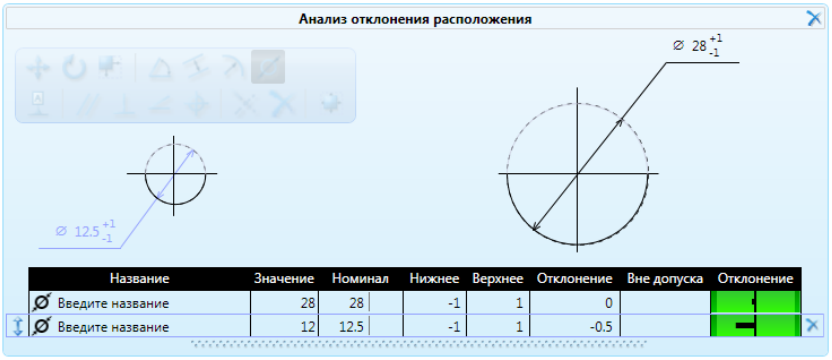


Рисунок 136. Пример установки радиальных параметров

Диаметральные размеры

Диаметральные размеры, аналогично радиальным размерам, можно устанавливать для окружностей и дуг (см. **Рисунок 137**). Чтобы это сделать, необходимо выбрать соответствующий инструмент на панели инструментов (см. **Рисунок 134**), после чего щелкнуть на геометрический элемент и, удерживая кнопку мыши, оттащить указатель в другую точку. В точке, где будет отпущена мышь, появится выносная линия с номинальным размером и допусками. В таблице, расположенной ниже, следует ввести номинальное значение и допуск в соответствии с чертежом. Чтобы узнать к какой строчке относится тот или иной размер следует навести на него мышь, а в таблице будет выделена соответствующая строчка (см. **Рисунок 137**).

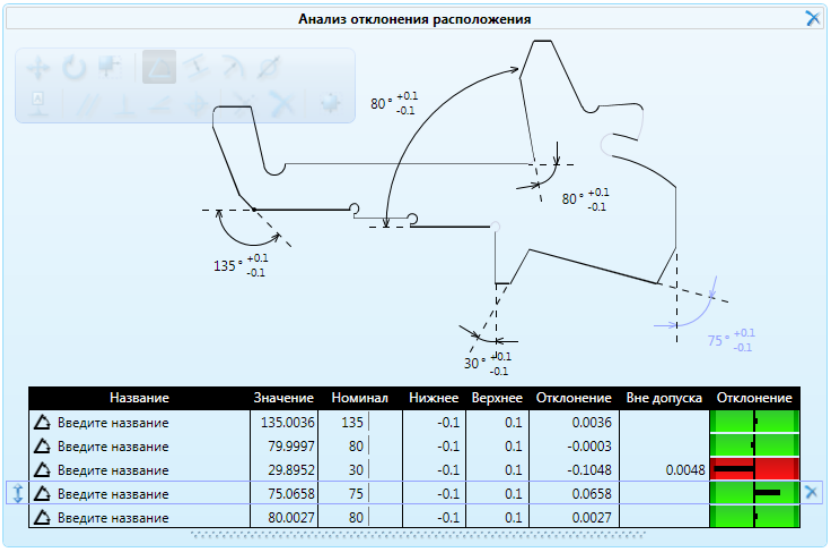


Название	Значение	Номинал	Нижнее	Верхнее	Отклонение	Вне допуска	Отклонение
Введите название	28	28	-1	1	0		
Введите название	12	12.5	-1	1	-0.5		

Рисунок 137. Пример установки диаметральных параметров

Угловые размеры

Угловые размеры допустимо указывать между двумя непараллельными прямыми. Чтобы это сделать, необходимо выбрать соответствующий инструмент на панели инструментов (см. **Рисунок 134**), после чего нажать левую клавишу на первой прямой и, удерживая кнопку мыши, дотащить указатель до другой. В таблице, расположенной ниже, следует ввести номинальное значение и допуск в соответствии с чертежом. Чтобы узнать к какой строчке относится тот или иной размер следует навести на него мышь, а в таблице будет выделена соответствующая строчка (см. **Рисунок 138**).



Название	Значение	Номинал	Нижнее	Верхнее	Отклонение	Вне допуска	Отклонение
Введите название	135.0036	135	-0.1	0.1	0.0036		
Введите название	79.9997	80	-0.1	0.1	-0.0003		
Введите название	29.8952	30	-0.1	0.1	-0.1048	0.0048	
Введите название	75.0658	75	-0.1	0.1	0.0658		
Введите название	80.0027	80	-0.1	0.1	0.0027		

Рисунок 138. Пример расстановки угловых размеров

Дополнительные построения

Часто для измерения определенного параметра не достаточно измеренных геометрических элементов CAD модели. Возникает потребность в построении виртуальных геометрических элементов по определенным правилам. Координатный анализ позволяет производить построение новых геометрических элементов: получать точки пересечения, строить параллельные, перпендикулярные элементы, т.е. создавать дополнительные построения.

Дополнительные построения можно найти на вкладке «Построения» на панели инструментов (см. **Рисунок 139**). Готовые дополнительные построения отображаются оранжевым цветом.

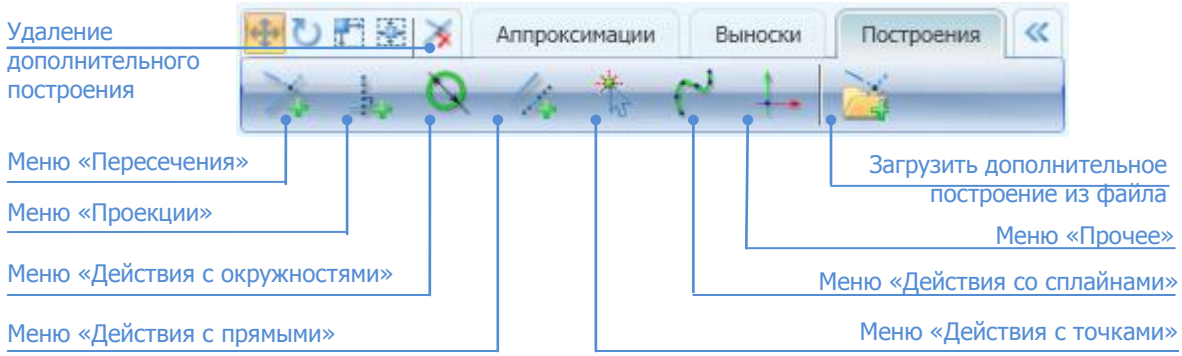


Рисунок 139. Панель инструментов для дополнительных построений

Дополнительные построения позволяют произвести большое количество расчетов без написания оператором (или метрологом) дополнительной расчетной программы, которая, как правило, имеется в схеме измерения.

Поиск нужного инструмента очень прост, т.к. инструменты дополнительных построений логически сгруппированы в меню:

- пересечения;
- проекции;
- действия с окружностями;
- действия с прямыми;
- действия с точками;
- действия со сплайнами;
- прочее.

Для каждого инструмента имеется подсказка. Чтобы вызвать подсказку, необходимо навести и удерживать указатель мыши на кнопке инструмента.

Выполнение простейших построений интуитивно понятно. Например, чтобы пересечь две прямые, необходимо выбрать инструмент «Две прямые» в меню «Пересечения». Выбрать первую прямую и дотянуть её до второй, в момент, когда вторая прямая подсветится – отпустить клавишу мыши; чтобы построить центр окружности, необходимо выбрать инструмент «Центр окружности» в меню «Действия с окружностями» и указать окружность или дугу.

При выполнении более сложных построений появляется мастер дополнительных построений, который разбивает построение на шаги. На каждом шаге кратко описано,

какой элемент необходимо выбрать. Мастер позволяет возвращаться к предыдущему шагу и изменять входные данные для построения. Также на любом шаге можно отменить выполнение дополнительного построения (см. **Рисунок 140**).

Например, при построении касательных к окружности через точку на первом шаге мастер требует указать точку, через которую будут проведены касательные, а на втором шаге – окружность, к которой их необходимо провести.

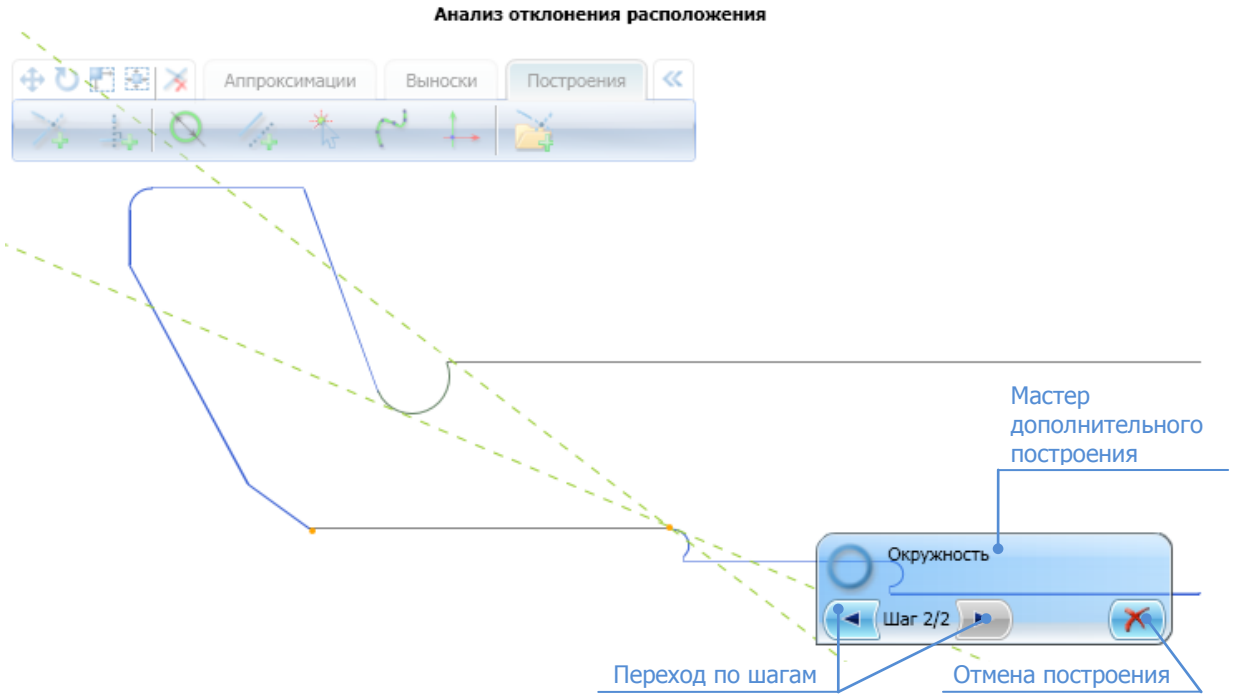


Рисунок 140. Мастер дополнительного построения

В случае невозможности произвести определенное дополнительное построение из-за нехватки базовых инструментов можно воспользоваться загрузкой дополнительного построения из файла. В этом случае разработка такого дополнительного построения производится специалистами Челяб*НИИ*контроль по индивидуальному заказу (оплачивается отдельно).

Чтобы загрузить такое дополнительное построение следует нажать соответствующую кнопку на панели инструментов, после чего появится стандартный мастер дополнительного построения.

На вкладке «Выноски» имеется инструмент «Информация о дополнительном построении», позволяющий узнать, какое дополнительное построение было проведено. Для этого необходимо взять данный инструмент и навести указатель мыши на дополнительное построение – появится всплывающая подсказка с информацией о дополнительном построении.

Анализ расположения в пространстве

Введение

Практически во всем мире используются семь параметров для нормирования точности расположения элементов детали:

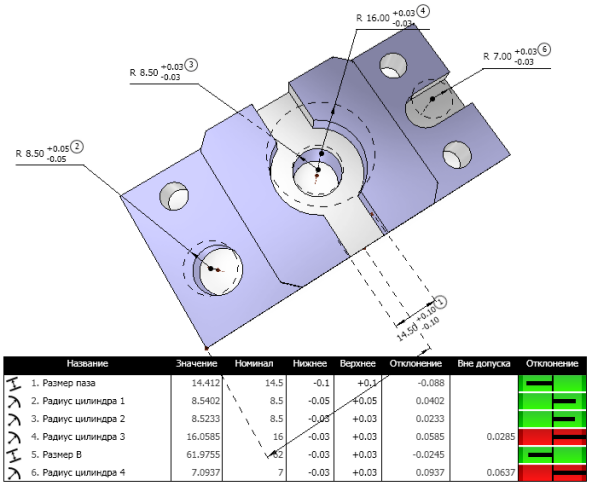
- отклонение от параллельности;
- отклонение от перпендикулярности;
- отклонение наклона;
- позиционное отклонение;
- отклонение от соосности;
- отклонение от симметричности;
- отклонение от пересечения осей;

Данный анализ работает с координатными данными, измерение которых было произведено в трехмерном пространстве и позволяет производить расчет отклонений соосности, симметричности и пересечения осей.

В программном обеспечении **ТЕХНО**коорд™ анализ расположения в пространстве включен в стандартную и реверсивную контактные схемы измерения.

Принцип работы

Анализ расположения в пространстве позволяет производить вычисления параметров, таких как расстояния, угловые размеры, отклонения от перпендикулярности, параллельности и т.д., а также производить дополнительные построения, т.е. достраивать новые элементы по определенным законам (например, точка пересечения трех плоскостей или центр сферы). Все операции производятся визуально (см. **Рисунок 141**).



	Название	Значение	Номинал	Нижнее	Верхнее	Отклонение	Вне допуска	Отклонение
1	1. Размер паза	14.412	14.5	-0.1	+0.1	-0.088		
2	2. Радиус цилиндра 1	8.5402	8.5	-0.05	+0.05	0.0402		
3	3. Радиус цилиндра 2	8.5233	8.5	-0.03	+0.03	0.0233		
4	4. Радиус цилиндра 3	16.0585	16	-0.03	+0.03	0.0585	0.0285	
5	5. Размер B	61.9755	62	-0.03	+0.03	-0.0245		
6	6. Радиус цилиндра 4	7.0937	7	-0.03	+0.03	0.0937	0.0637	

Рисунок 141. Пример отклонения расположения в пространстве

После создания анализа расположения появится геометрия детали и полупрозрачная панель инструментов в левом верхнем углу отчета (см. **Рисунок 142**).

Чтобы активизировать панель инструментов, необходимо навести на нее указатель мыши. Для каждой кнопки панели предусмотрены подсказки. Чтобы вызвать подсказку необходимо навести и удерживать курсор мыши на кнопке.

Чтобы скрыть панель инструментов можно воспользоваться соответствующей кнопкой на панели инструментов или нажать клавишу «Shift»

Рисунок 142. Панель инструментов

После этого можно расположить деталь правильным образом и изменить размеры элемента анализа. Для ориентации детали существуют специальные кнопки на панели инструментов (см. **Рисунок 142**).

Масштабирование детали можно осуществлять с помощью колесика мыши, а перемещение – средней клавишей мыши.

Анализ расположения в пространстве использует трехмерную CAD-модель. Измеренные точки соотносятся с определенными элементами CAD-модели. Прежде всего, необходимо получить из точек заменяющие элементы: средние элементы, прилегающие или минимальные зоны.

При указании размеров и допусков в таблицу автоматически добавляется строчка, где есть рассчитанное значение и можно заполнить: «Номинал», «Нижнее» и «Верхнее» (допустимое отклонение). В случае если данный геометрический элемент не был измерен, то он отображается серым цветом, а в таблице отображается пустая строчка.

Для добавленных размеров также можно указать номиналы, нижние и верхние отклонения, поэтому их можно рассматривать и отдельно тоже (см. **Рисунок 143**).

Название	Значение	Номинал	Нижнее	Верхнее	Отклонение	Вне допуска	Отклонение
Введите название	66.3876	66.3	-0.2	0.2	0.0876		

Рисунок 143. Пример строчки из таблицы

Некоторые допуски работают только при наличии определенных указанных размеров, например, допуск наклона, позиционный допуск. Если соответствующих размеров недостаточно, то строчка в таблице останется пустой.

Аппроксимация элементов детали

Все расчеты в редакторе происходят с элементами, которые называются «Заменяющие элементы» - это геометрические элементы, полученные в результате аппроксимации какого-либо примитива по координатам точек.

Анализ расположения в пространстве использует следующие типы геометрических элементов:

- Плоскость;
- Сфера;
- Цилиндр;
- Конус;
- Тор;
- Сплайновая поверхность;
- Сплайновая кривая;
- Окружность;
- Прямая;
- Точка;
- Группа точек;

Средний элемент – поверхность, имеющая номинальную форму и такие размеры и/или расположение, чтобы сумма квадратов расстояний между реальным и средним элементами в пределах нормируемого участка имела минимальное значение.

Прилегающей поверхностью называется поверхность, имеющая форму номинальной поверхности, соприкасающаяся с реальной поверхностью и расположенная вне материала детали так, что отклонение от нее наиболее удаленной точки реальной поверхности в пределах нормируемого участка имеет минимальное значение. Это понятие относится к прилегающей плоскости, прилегающей прямой, однако указанное условие минимального значения отклонения не распространяется на отклонения формы цилиндра и окружности.

Прилегающим цилиндром (конусом, сферой, тором) называется цилиндр (конус, сфера, тор) минимального диаметра, описанного вокруг реальной наружной поверхности, или максимального диаметра, вписанного в реальную внутреннюю поверхность.

Вместо прилегающего цилиндра (конуса, сферы, тора) в качестве базы для определения отклонений допускается также использовать цилиндр минимальной зоны. Цилиндр (конус, сфера, тор) минимальной зоны – цилиндр (конус, сфера, тор), соприкасающийся с реальной поверхностью и расположенный вне материала так, чтобы наибольшее расстояние между реальной поверхностью и заменяющим элементом имело минимальное значение.

Чтобы создать элемент можно выбрать в меню его тип и метод аппроксимации (см. **Рисунок 144**), затем выбрать поверхности, которые относятся к данному элементу, и завершить создание кнопкой в верхнем правом углу.

Выбор нескольких поверхностей позволяет, например, аппроксимировать одну плоскость из двух площадок на детали, если это требуется с точки зрения контроля, или выбрать две половинки одной поверхности, «разбитой» при экспорте из CAD-редактора.

Инструменты для аппроксимации расположены на соответствующей вкладке на панели инструментов (см. **Рисунок 144**).

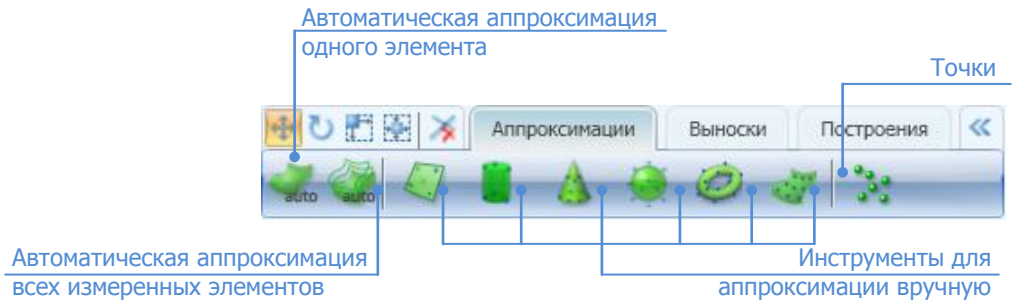


Рисунок 144. Инструменты аппроксимации для анализа расположения в пространстве

Если поверхность представлена одним элементом на CAD-модели, требуется средний заменяющий элемент предлагается быстрый способ создания заменяющих элементов – автоматическое создание элементов одним кликом (см. **Рисунок 144**). После выбора данного инструмента один клик мыши по CAD-поверхности автоматически создает средний элемент соответствующего типа.

Также имеется инструмент автоматической аппроксимации, который строит заменяющие элементы для всех измеренных поверхностей.

Ниже, расположена таблица указывающая на допустимость того или иного типа геометрического примитива для того или иного действия.

	Средний элемент	Прилегающий элемент	Минимальная зона	Результат построения[1]
Плоскость	✔	✔	✔	✔
Цилиндр	✔	✔	✔	✔
Конус	✔	✔	✔	✔
Сфера	✔	✔	✔	✔
Тор	✔		✔	✔
Сплайновая поверхность	✔			
Сплайновая кривая				✔
Окружность				✔
Прямая				✔
Точка				✔
Группа точек[2]	✔			

[1] Наличие элемента в данном столбике означает, что он может быть получен в результате дополнительного построения. Можно заметить, что элементы Окружность, прямая и точка могут быть только получены в результате каких-либо операций, например пересечений, нахождения центра сферы и т.д.

[2] Инструмент «Точки» позволяет создать облако точек вместо аппроксимируемого элемента. Полученные точки можно использовать для нестандартизированных расчетов и построений в расчетной программе (подробнее см. главу «Язык для математической обработки данных координатного анализа»).

Указание допусков

Чтобы указать допуск в редакторе, необходимо выбрать соответствующий инструмент на вкладке «Выноски» на панели инструментов (см. **Рисунок 145**) и щелчком мыши указать аппроксимированный элемент геометрии в эскизе.

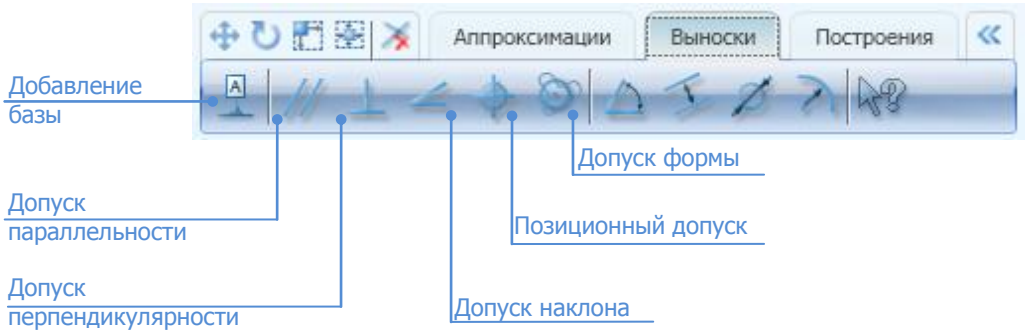


Рисунок 145. Панель инструментов, указание допусков

Для установки допуска следует на базовой плоскости установить базу. Для этого следует выбрать соответствующую кнопку в панели инструментов («База») и «кликнуть» мышью на соответствующем элементе. Затем выбрать один из инструментов, позволяющих задавать допуск и «кликнуть» на соответствующем элементе (см. **Рисунок 146**).

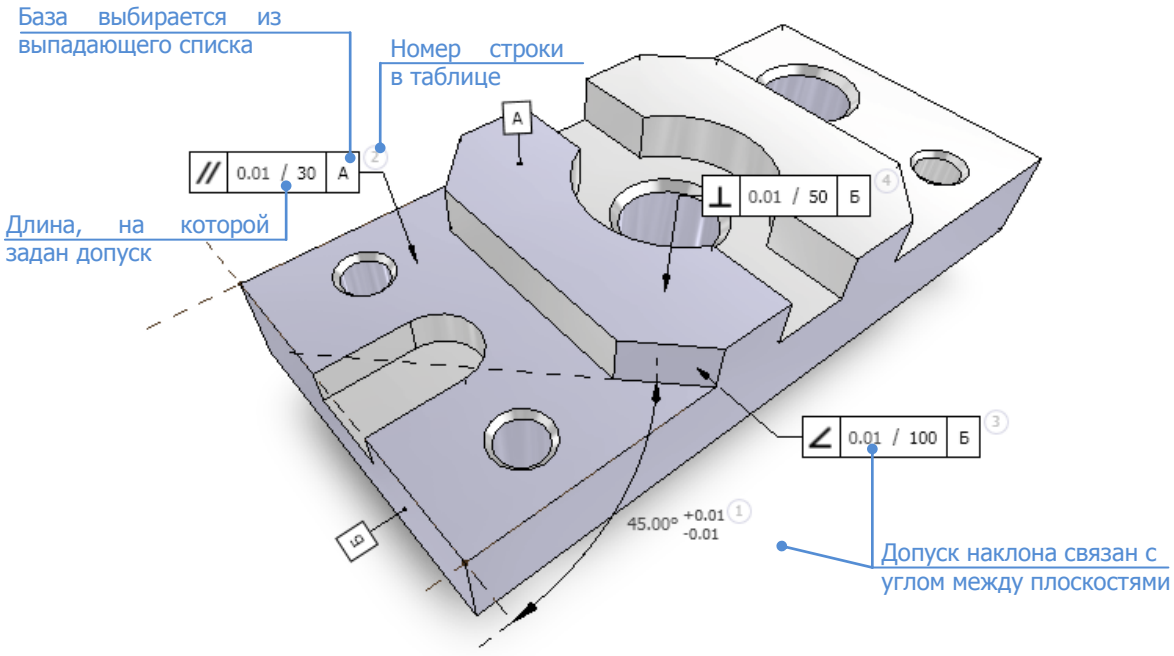


Рисунок 146. Пример расстановки допусков

Для допусков параллельности, перпендикулярности требуется указать только базу. Для допуска наклона требуется также наличие указанного углового размера (см. **Рисунок 146**).

Позиционный допуск можно привязать только к точке. Расчет позиционного допуска производится следующим образом: из всех связанных с точкой размеров выбирается наиболее отклоняющийся от номинала, данное отклонение и считается действительным (см. **Рисунок 147**).

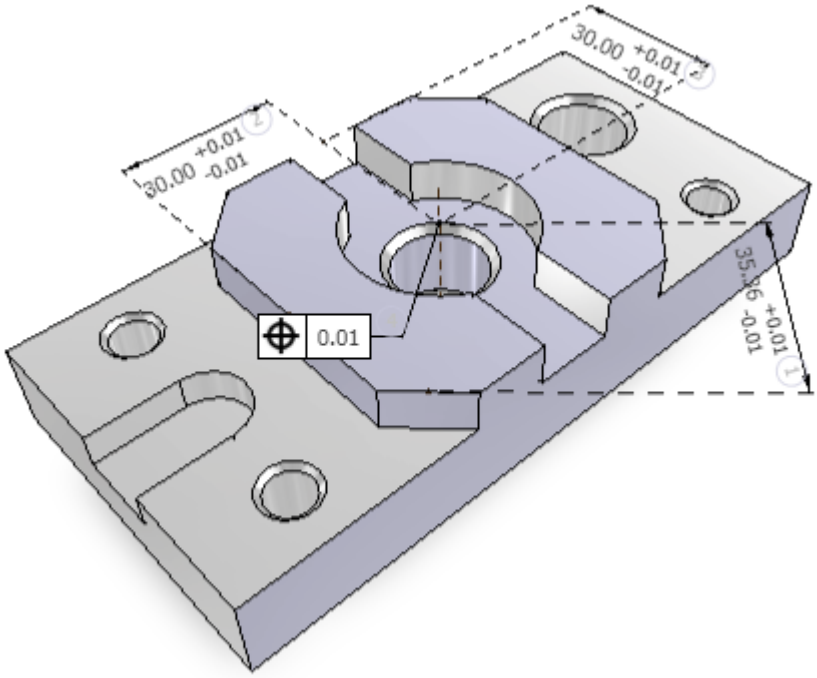


Рисунок 147. Пример установки позиционного допуска

Указание размеров

Наиболее часто возникает необходимость указать размер элемента. В программном обеспечении **ТЕХНО**координ™ предусмотрена возможность вынесения углового, линейного, радиального и углового размеров.

Чтобы добавить размер в редакторе, необходимо выбрать соответствующий инструмент (см. **Рисунок 148**) на вкладке «Выноски» и щелкнуть на элемент геометрии.

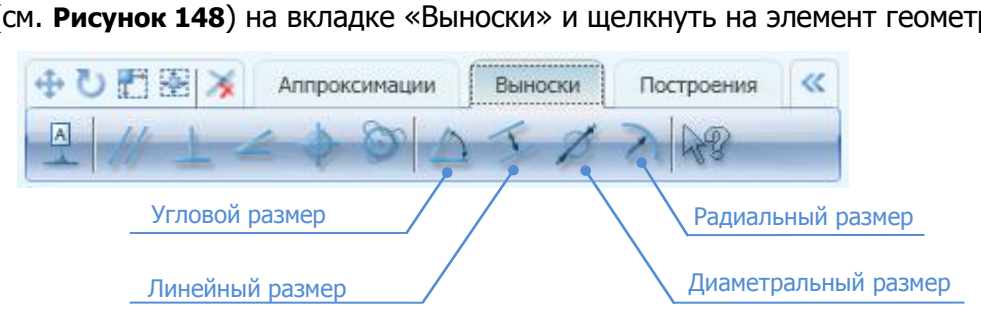


Рисунок 148. Отклонения расположения, панель инструментов, размерные параметры

Построенные выноски можно перемещать мышью.

Линейный размер позволяет вывести в отчет расстояние между двумя точками. Чтобы вывести в отчет линейный размер, следует выбрать инструмент «Линейный размер», нажать левую кнопку мыши на первой точке и отпустить на второй, в результате чего появится элемент, визуально отображающий расстояние (см. **Рисунок 149**), и соответствующая строка в таблице. Чтобы удалить созданный размер достаточно удалить соответствующую строку в таблице («крестик» с правой стороны).

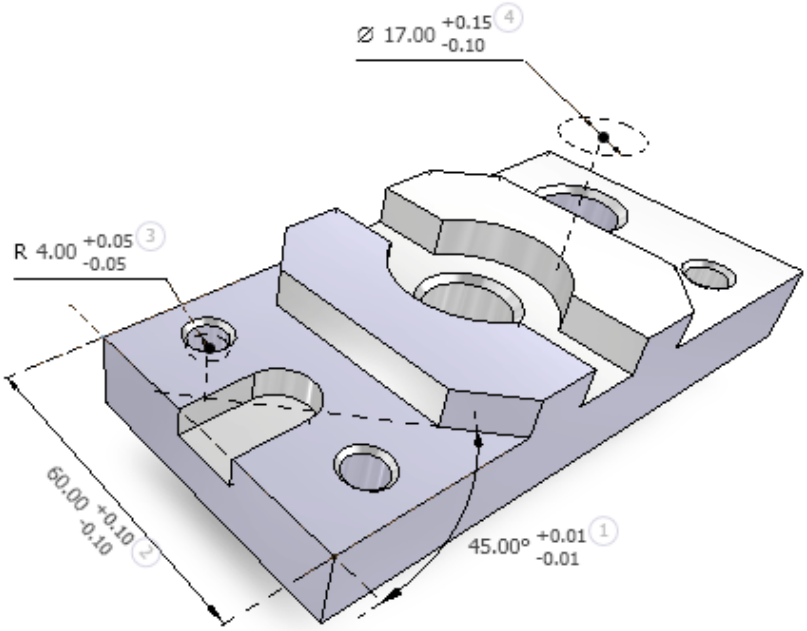


Рисунок 149. Пример указания размеров

Чтобы получить расстояние между плоскостью и точкой можно получить проекцию данной точки на плоскость и построить расстояние между двумя точками. Аналогично можно получить расстояние от точки до прямой.

Угловой размер позволяет вывести в отчет угол между двумя плоскостями или между линией и плоскостью. Радиальный и диаметральные размеры позволяют вывести в отчет радиус (или диаметр) цилиндра, сферы или окружности (см. **Рисунок 149**).

Дополнительные построения

Часто для измерения определенного параметра недостаточно измеренных геометрических элементов CAD модели. Возникает потребность в построении виртуальных геометрических элементов по определенным правилам. Координатный анализ позволяет производить построение новых геометрических элементов: получать точки пересечения, строить параллельные, перпендикулярные элементы и т.д., т.е. создавать дополнительные построения.

Дополнительные построения позволяют произвести большое количество расчетов без написания оператором (или метрологом) дополнительной расчетной программы, которая, как правило, имеется в схеме измерения.

Поиск нужного инструмента очень прост, т.к. инструменты дополнительных построений логически сгруппированы в меню:

- пересечения;
- проекции;
- действия с окружностями;
- действия с прямыми;
- действия с точками;
- действия со сплайновой кривой;
- действия с плоскостью;
- действия со сферой;
- действия с цилиндром;
- действия с конусом;
- действия с тором;
- действия со сплайновой поверхностью;
- загрузка внешних дополнительных построений.

Для каждого инструмента имеется подсказка. Чтобы вызвать подсказку, необходимо навести и удерживать указатель мыши на кнопке инструмента (см. **Рисунок 150**).

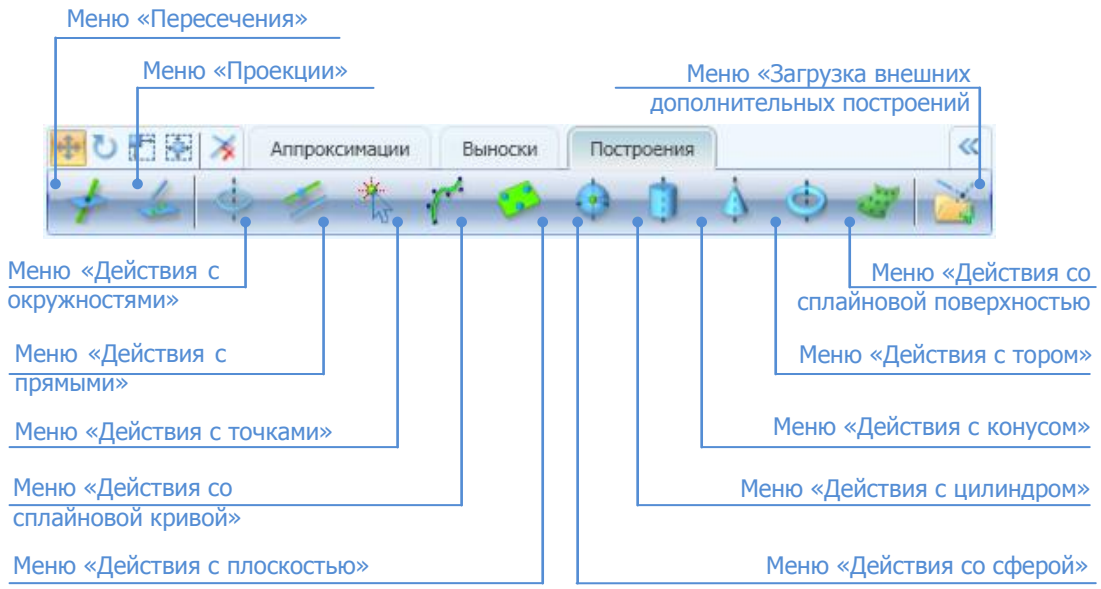


Рисунок 150. Панель инструментов для дополнительных построений

Редактор предоставляет большое количество элементарных дополнительных построений, комбинируя которые, можно произвести расчет многих контролируемых параметров на детали.

При запуске дополнительного построения появляется мастер, который предлагает выбрать набор геометрических элементов (см. **Рисунок 151**). Выбор производится щелчком мыши. В результате получаются один или несколько новых геометрических элементов.

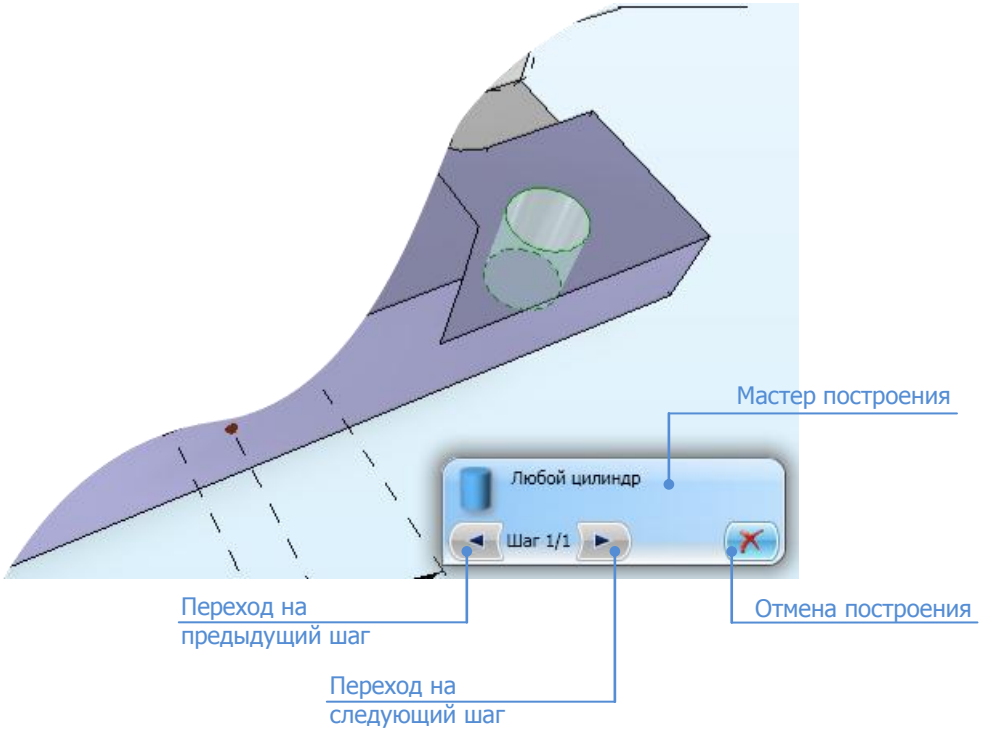


Рисунок 151. Мастер создания дополнительного построения

Дополнительные построения могут быть загружены из файлов. Данные файлы имеют расширение *.construction. Они могут быть загружены с web-узла, переданы по электронной почте и т.д. Для однократного запуска можно воспользоваться кнопкой на панели инструментов «Запустить из файла» в меню «Загрузка внешних дополнительных построений». Запуск и выполнение загружаемого дополнительного построения полностью аналогичен стандартным дополнительным построениям.

Если дополнительное построение используется часто целесообразно добавить его непосредственно в меню «Загрузка внешних дополнительных построений» на панели инструментов. Для этого достаточно сделать двойной клик на соответствующем файле и данное дополнительное построение будет добавлено на панель инструментов (см. **Рисунок 150**).

В случае невозможности произвести определенное дополнительное построение из-за нехватки базовых инструментов можно воспользоваться загрузкой дополнительного построения из файла. В этом случае разработка такого дополнительного построения производится специалистами ЗАО «ЧелябНИИконтроль» по индивидуальному заказу (оплачивается отдельно).

На вкладке «Выноски» имеется инструмент «Информация о дополнительном построении», позволяющий узнать, какое дополнительное построение было проведено. Для этого необходимо взять данный инструмент и навести указатель мыши на дополнительное построение – появится всплывающая подсказка с информацией о дополнительном построении.

Оформление отчета

Нередко после измерения, в отчет (протокол измерения), требуется добавить дополнительную текстовую или графическую информацию. Так, например,

- Шапка отчета (с названием и логотипом организации);
- Имя контролера, который произвел измерение (или калибровку);
- Фотография измеренной детали;
- Или просто информация, которая может быть полезна при просмотре печатного варианта протокола измерения.

Анализ имеет возможность добавления информации такого рода. Для этого предназначен специальный элемент «Заметка». Данный элемент позволяет отображать текстовую и графическую информацию. На рисунке ниже (см. **Рисунок 152**) показан пример оформления шапки документа.

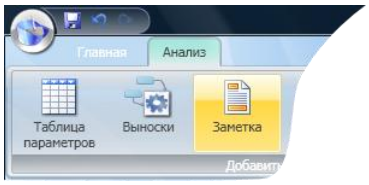


Рисунок 152. Главное меню: "Заметка"

Чтобы добавить элемент оформления отчета в главном меню необходимо нажать на кнопку «Заметка», после чего в редакторе появится еще один элемент.

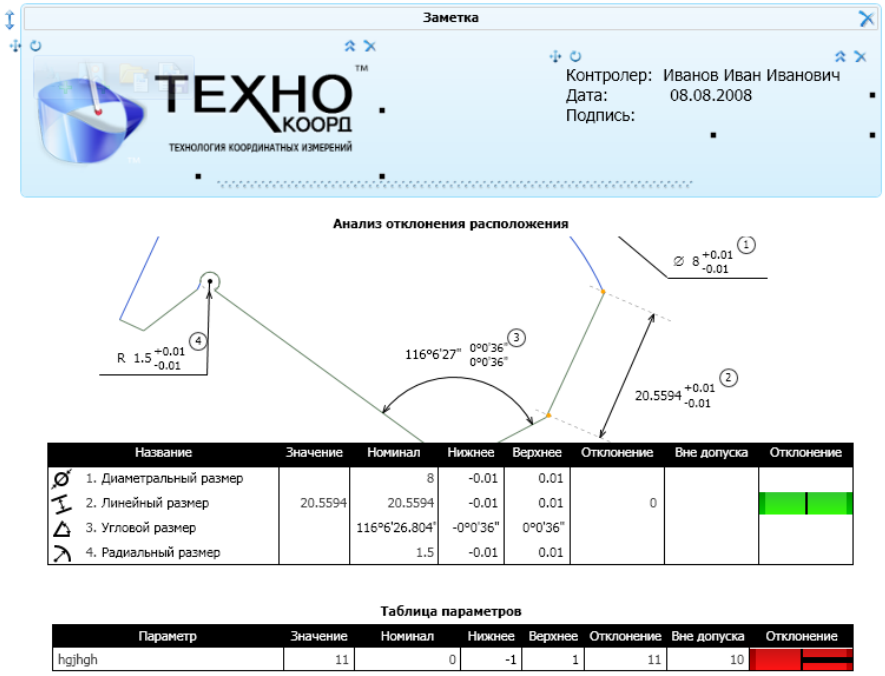


Рисунок 153. Пример оформления шапки отчета с помощью элемента "Заметка"

Текстовые блоки

Текстовый блок можно использовать, чтобы вписать в отчет Ф.И.О. оператора, дату выполнения измерения или др. Чтобы добавить текстовый блок следует нажать «Добавление текстового элемента» на панели инструментов

блока «Заметка» (см. **Рисунок 154**). Текстовые блоки позволяют размещать текст определенного размера, шрифта, цвета и т.д. (см. **Рисунок 155**)

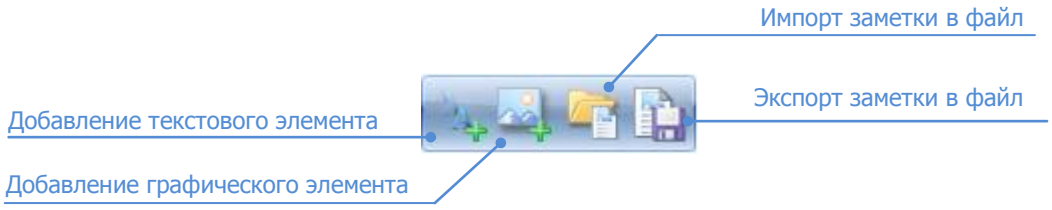


Рисунок 154. Панель инструментов элемента оформления

Текстовым блоком можно манипулировать, используя маркеры, позволяющие менять его размер, поворачивать, перемещать и т.д. (см. **Рисунок 155**)

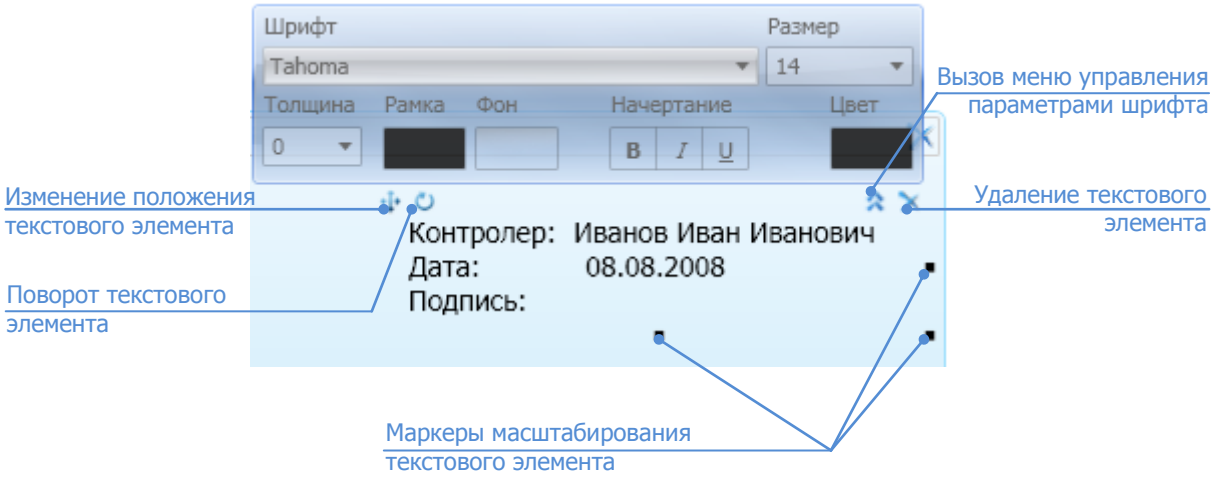


Рисунок 155. Редактирование текстового блока

Изображения

Для того чтобы разместить логотип, фотографию изделия и т.п., можно добавить изображение. Изображение также как текстовый блок можно поворачивать, перемещать, менять размеры, используя маркеры (см. **Рисунок 156**).

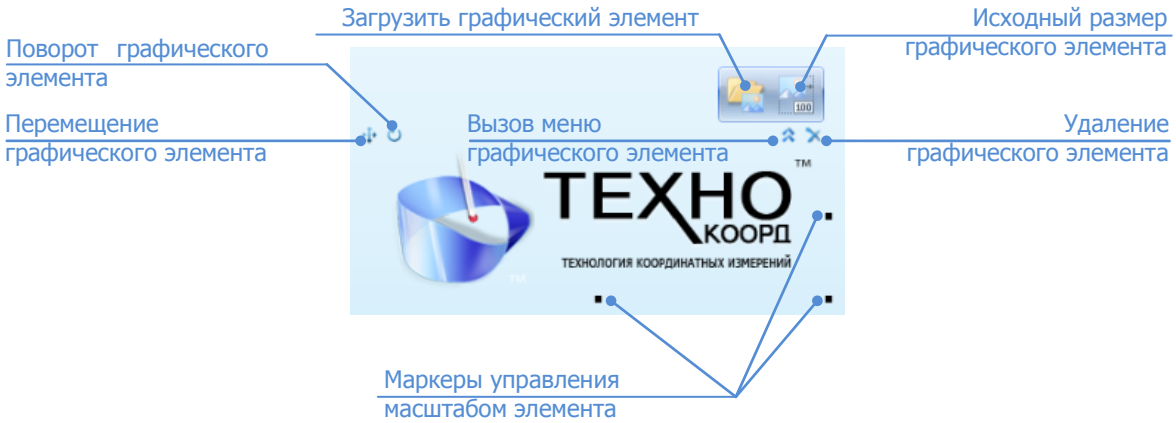


Рисунок 156. Управление параметрами графического изображения в блоке "Заметка"

Экспорт данных анализа

Данные в координатном анализе можно сохранить в формате *.measure и *.analysis. В файл с расширением *.measure сохраняются только рассчитанные данные, которые можно применять для других анализов. Данные в файле *.analysis представлены наглядно с сохранением всей расчетной информации.

Чтобы сохранить данные в формате *measure, необходимо щелкнуть по кнопке «Сохранить результат» в главном меню, указать название файла и выбрать директорию для сохранения (см. **Рисунок 114**).

Чтобы сохранить данные в формате *.analysis, необходимо щелкнуть по кнопке «Сохранить анализ» в главном меню, указать название файла и выбрать директорию для сохранения (см. **Рисунок 114**).

Сохранение и печать отчета

Чтобы сохранить анализ в файл отчет с текущим содержимым (т.е. текстовый файл для печати) следует нажать круглую кнопку, выбрать пункт «Публикация» в главном меню редактора анализа, «Публикация как XPS» (см. **Рисунок 157**). После чего ввести название файла, выбрать директорию для сохранения, сохранить в файл.

Сохранение отчета осуществляется в открытый формат Microsoft XPS™. Данный формат платформенно независим и может быть перенесен на другую машину.

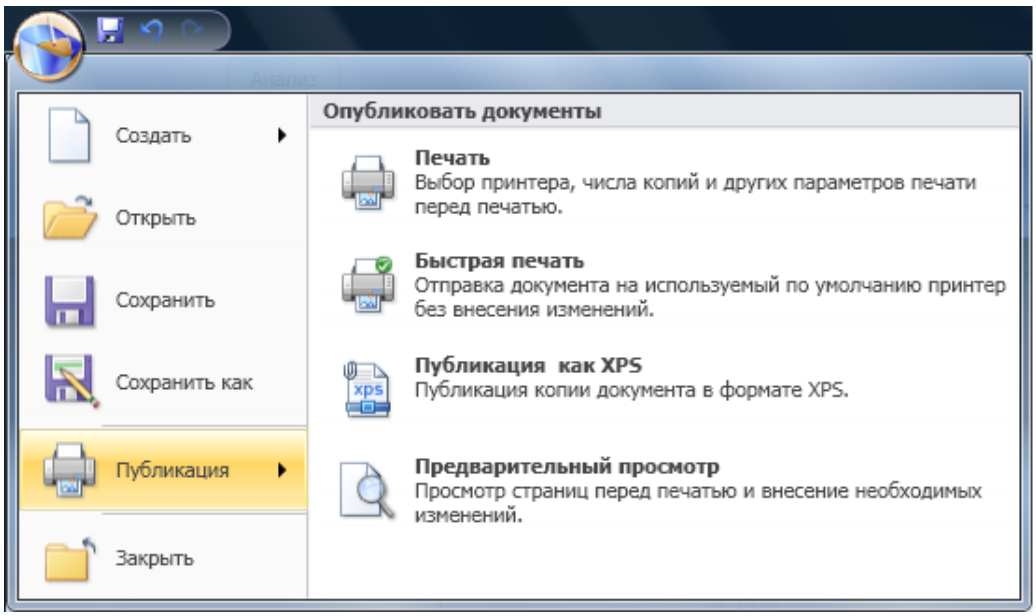


Рисунок 157. Сохранение и печать отчета

После экспорта в формат XPS его редактирование будет затруднено. Экспорт в формат XPS предназначен, прежде всего, для последующей распечатки, дистрибуции или помещении в архив протоколов.

Существует две функции печати текущего содержимого координатного анализа: печать с предварительным выбором параметров печати (пункт «Печать») и печать документа без установки параметров печати (пункт «Быстрая печать», см. **Рисунок 157**). Чтобы распечатать, нажмите кнопку в главном меню «Печать», в появившемся окне необходимо выбрать принтер и нажать «Ok».

Обратите внимание, что разбиение документа на листы в редакторе соответствует разбиению документа на листы в печатном виде (см. **Рисунок 158**).

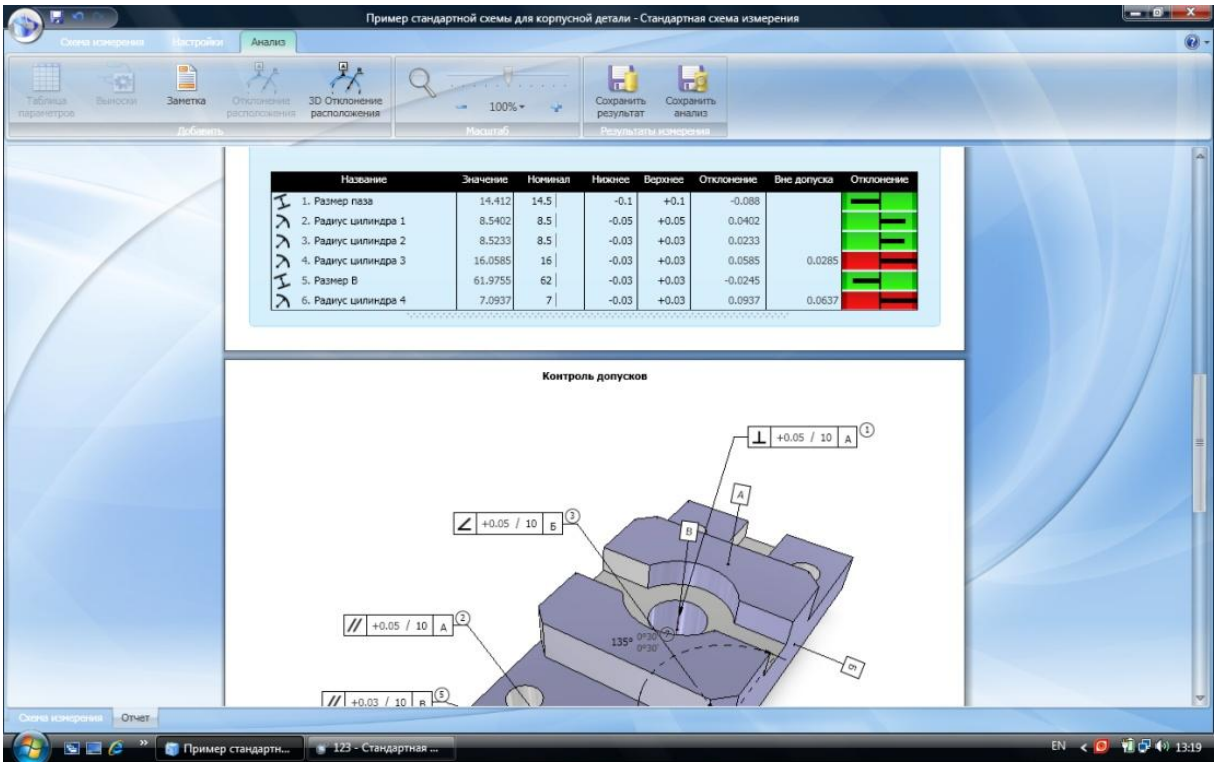


Рисунок 158. Редактор автоматически разбивает отчет на листы

ТЕХНО™
КООРД
ТЕХНОЛОГИЯ КООРДИНАТНЫХ ИЗМЕРЕНИЙ

Язык для математической обработки данных координатных измерений

Оглавление

Назначение языка

Данный язык предназначен для расчетов геометрических параметров деталей, используя только координаты точек, измеренных каким либо прибором.

Терминологически большинство функций соответствует стандарту ГОСТ 24642-81 «Допуски формы и расположения поверхностей». Стандарт соответствует в части терминологии международным стандартам ISO 1101-83 и ISO 5459-81. Данное соответствие делает язык более понятным и удобным в использовании специалисту – метрологу.

Редактор данного языка интегрируется в различные приложения, предоставляя возможность производить обработку полученных данных измерения. Кроме того нередко в язык добавляются специальные объекты – расширения, которые предоставляют дополнительную функциональность, неописанную в данном руководстве. Такой функциональностью может являться, например, функции управления координатной измерительной машиной. Такие расширения языка следует изучать в руководствах на соответствующие продукты.

Синтаксис языка

Синтаксис данного языка является подмножеством языка C#, который соответствует стандарту ISO/IEC 23270:2006. Настоящая документация не описывает всех особенностей, описанных в стандарте ISO/IEC 23270:2006, а только конструкции, которые необходимы для проведения расчетов.

Структура программы

Обозначать начало и конец программы не требуется. Текст расчетной программы представляет собой просто последовательность инструкций.

Создание переменной

Данный язык является строго типизированным, это означает в частности, что при объявлении переменной, необходимо указывать какого она типа. Например, создание числовой переменной выглядит так:

```
Number number = 12.8;
```

То есть, сначала указывается тип, затем название переменной, через равно записывается инициальное значение.

Существует возможность создавать переменную и сразу же присваивать ей значение, возвращаемое из метода, например, присвоить квадратный корень из двух:

```
Number sqrt = Basic.Sqrt(2);
```

Присвоение переменной

Присвоение переменной производится с помощью знака равно =. Например, так:

```
number = 1;
number = 2 + 2;
```

Можно присваивать значения другой переменной, например, так:

```
Number a = 1;
Number b = 2;
a = b;
```

Существуют два типа переменных: *ссылочные* и *значимые*. Большинство типов языка *значимые* и для них конструкция a=b будет означать именно помещение значения b в a. Для *ссылочных* же типов это будет означать копирование ссылки b в a, но не перенос реального значения, т.е. получится две переменных указывающих на одно и тоже значение. Для *ссылочных* типов необходимо использовать метод Clone(), например, так: a = b.Clone();

Вызов методов

Каждая переменная и даже класс (тип) имеет методы. Методы необходимы, чтобы производить различные операции с данными. Любой метод характеризуется тем, что он принимает на вход, какие операции производит с данными и что возвращает. Входные значения методу передаются через запятую, возвращаемые значения можно присвоить в переменную, например, передадим в метод Method, переменной a, значения 1 и 2:

```
Number variable = a.Method(1, 2);
```

Возвращаемое значение в данном случае записывается в переменную типа Number.

Иногда метод не возвращает ничего, а только изменяет внутренние данные переменной, например, операция нормализации вектора записывается так:

```
Vector3d a = new Vector3d(5.67, 84.1, 0.5);
a.Normalize();
```

Как уже отмечалось ранее, типы (классы) также имеют методы. Данные методы называются статическими. Для вызова статического метода необходимо написать название типа и поставить точку. Так, например, чтобы аппроксимировать сферу (т.е. получить средний заменяющий элемент) по набору точек points нужно обратиться к соответствующему статическому методу класса Average:

```
Sphere sphere = Average.Sphere(points);
```

Условные выражения

Язык позволяет использовать условные выражения и ветвления. Для конструирования условных выражений используются следующие операторы:

Оператор	Описание
&&	Логическое И
||	Логическое ИЛИ

Для сравнения числовых переменных / констант служат следующие операторы:

Оператор	Описание
==	Равно
>=	Больше или равно
<=	Меньше или равно
!=	Не равно

Например, чтобы вычислить косинус в случае значения больше пяти или синус в противном случае, можно написать следующее:

```
Number a = 7;
Number result = 0;
if (a > 5)
{result = Basic.Cos(a);}
else
{result = Basic.Sin(a);}
```

В расчетах используется форма с плавающей точкой для хранения чисел, что приводит к тому, что в результате расчетов значение числа накапливает погрешность. Данная погрешность приводит к тому, что простое сравнение может дать ложный результат, для решения этой проблемы следует использовать метод `Basic.ApproxEqual`

Сравнение с учетом накопленной погрешности:

```
// Сравнение с толерантностью по умолчанию
Boolean result1 = Basic.ApproxEqual(v, u);
// Сравнение с указанной толерантностью
Boolean result2 = Basic.ApproxEqual(v, u, 0.0001);
```

Циклические конструкции

Циклические конструкции предоставляют удобный способ организации ветвлений в программе. Как правило, циклы требуются для однотипной обработки каких-то однородных данных.

Цикл for

Этот цикл чаще всего используется, когда число повторений известно заранее.

```
for (int i = 0; i < 5; i++)
{result += Basic.Cos(input[i]);}
```

Синтаксис. Сначала указывается начальное значение переменной, затем условие продолжения цикла и выражение, которое выполняется каждую итерацию. В фигурных скобках, которые находятся следом, располагается тело цикла.

Цикл while

Пока истинно условие выполняется оператор - тело цикла.

```
while (i < 5)
{
    result += Basic.Cos(input[i]);
    i++;
}
```

Цикл do while

Здесь сначала один раз выполняется тело цикла, затем проверяется условие. Если условие истинно, то выполнение тела цикла повторяется.

```
do
{
    result += Basic.Cos(input[i]);
    i++;
}
while (i < 5);
```

Цикл foreach

Этот цикл полезен для перебора всех элементов в массиве.

```
int[] array = new int[]{1,2,3};
foreach (int item in array)
{
   result += item;
}
```

При этом переменной item присваивается по очереди каждый элемент массива. Если переменная item является значимым типом (а не ссылочным), то изменять элементы массива через нее не получится. Если же переменная - объект, то через нее можно менять текущий элемент массива.

Операторы управления циклом

`break`

Этот оператор может стоять внутри циклов for, do-while и while. Выполнение оператора break приводит к выходу из цикла.

`continue`

Этот оператор может стоять внутри циклов for, do-while и while. Выполнение оператора continue приводит к переходу на проверку условия цикла.

Создание собственных функций (версия 3.0)

Возможность создания пользовательских функций доступна в версии 3.0 и выше

Свои собственные функции можно определять в любом месте программы, как показано в примере ниже:

```
function Vector3d Function(Number a, Circle3d b)
{
    return a * b.Radius;
}
```

После ключевого слова `function` следует возвращаемый тип, затем название функции. В скобках указываются входные значения через запятую. В фигурных скобках расположено тело функции, где должен производиться какой-то расчет. Чтобы закончить выполнение функции и возвратить результат используется ключевое слово `return`.

Если необходимо, чтобы функция производила какие-то операции, но не возвращала никаких значений, то вместо возвращаемого типа следует написать ключевое слово `void`

Элементы геометрии

Двухмерный вектор

Значимый тип. Содержит абсциссу и ординату.

Инициализация

Для создания вектора достаточно передать значения его компонент, например:

```
Vector2d vector = new Vector2d(45.9, 32);
```

Для копирования вектора достаточно его присвоить другому вектору, т.к. это значимый тип, то будет скопировано его содержимое, например:

```
Vector2d a = new Vector2d(1, 2);
Vector2d b = a; // теперь вектор b содержит (1, 2)
```

Доступны следующие константы:

- `Vector2d.Zero` – нулевой вектор;
- `Vector2d.XAxis` – единичный вектор, направленный вдоль оси абсцисс;
- `Vector2d.YAxis` – единичный вектор, направленный вдоль оси ординат;

Пример:

```
Vector2d vector = Vector2d.Zero;
```

Скалярное произведение

Скалярным произведением двух векторов называется число, равное произведению длин этих векторов на косинус угла между ними:

$$\vec{a}\vec{b} = |\vec{a}||\vec{b}|\cos\phi$$

Пример:

```
Vector2d a = new Vector2d(5.67, 84.1);
Vector3d b = new Vector2d(43.2, 5.8);
// Первый вариант, через метод класса
 Number result1 = a.DotProduct(b);
 // Второй вариант, через статическую функцию
Number result2 = Vector3d.DotProduct(a, b);
```

Скалярное произведение двух векторов $\vec{a}(a_1;a_2)$ и $\vec{b}(b_1;b_2)$ заданных своими координатами, может быть вычислено по формуле:

$$(\vec{a},\vec{b}) = a_1b_1 + a_2b_2$$

Сравнение векторов

Для того чтобы узнать, равны ли два вектора друг другу, можно использовать оператор равно, например:

```
Vector2d a = new Vector2d(5.67, 84.1);
Vector2d b = new Vector2d(43.2, 5.8);
Boolean result = (a == b);
```

В расчетах используется форма с плавающей точкой для хранения чисел, что приводит к тому, что в результате расчетов значение числа накапливает погрешность. Данная погрешность приводит к тому, что простое сравнение может дать ложный результат, для решения этой проблемы следует использовать метод `ApproxEqual`

Сравнение с учетом накопленной погрешности:

```
// Сравнение с толерантностью по умолчанию
Boolean result1 = Vector2d.ApproxEqual(v, u);
// Сравнение с указанной толерантностью
Boolean result2 = Vector2d.ApproxEqual(v, u, 0.0001);
```

Доступно также покомпонентное сравнение, когда все компоненты одного вектора меньше другого (или больше), доступны > , >=, <, <=, например:

```
Boolean result1 = v < u;
Boolean result2 = v >= u;
```

Нормализация

Нормализация - это преобразование, строящее вектор коллинеарный (параллельный) данному вектору, но с единичной длинной, например:

```
Vector2d a = new Vector2d(5.67, 84.1);
// Нормализуем текущий вектор
a.Normalize();
// Через статическую функцию, а результат
// помещаем в другой вектор
Vector2d b = Vector2d.Normalize(a);
```

Длину вектора можно определить с помощью функции `GetLength()`

Сложение векторов

Для проведения операций сложения векторов определены операторы «+» и «-».

```
Vector2d a = new Vector2d(5.67, 84.1);
Vector2d b = new Vector2d(43.2, 5.8);
Vector2d c = a + b;
```

Операции со скалярами

Операция сложения со скаляром представляет собой прибавление к каждому компоненту вектора этот скаляр, т.е.:

```
Vector2d a = new Vector2d(1, 1);
a = a + 1; // теперь a = (2, 2)
```

Определены также аналогичные операции «-», «*» и «/».

Двухмерная прямая

Значимый тип. Двухмерная прямая задается с помощью точки и направления.

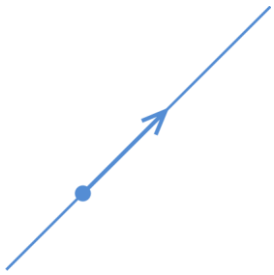


Рисунок 159. Двухмерная прямая

Описание построения	Для того, чтобы создать прямую, нужно передать два двухмерных вектора, которые будут задавать соответственно точку на конструируемой прямой и направление.

Пример	`Vector2d point = Vector2d.Zero;`
	`Vector2d direction = new Vector2d(1.0, 1.0);`
	`Line2d line = new Line2d(point, direction);`

Существует другой способ инициализации: передать по порядку компоненты векторов, например, такая же прямая получиться при такой инициализации:

```
Line2d line = new Line2d(0, 0, 1, 1);
```

Двухмерная окружность

Значимый тип. Двухмерная окружность задается с помощью указания ее центра и радиуса.

Описание построения	Для того чтобы создать окружность, нужно передать двухмерный вектор, который будет задавать соответственно центр, а также радиус.

Пример	`Number radius = 5.0;`
	`Vector2d center = new Vector2d(1.0, 1.0);`
	`Circle2d circle = new Circle2d(center, radius);`

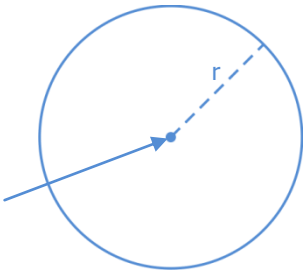


Рисунок 160. Двухмерная окружность

Существует другой способ инициализации, передать по порядку компоненты вектора, например, такая же окружность получиться при такой инициализации:

```
Circle2d circle = new Circle2d(1, 1, 5);
```

Трехмерный вектор

Значимый тип. Содержит абсциссу, ординату и аппликату.

Инициализация

Для создания вектора достаточно передать значения его компонент, например:

```
Vector3d vector = new Vector3d(45.9, 32, 56.7);
```

Для копирования вектора достаточно его присвоить другому вектору, т.к. это значимый тип, то будет скопировано его содержимое, например:

```
Vector3d a = new Vector2d(1, 2, 3);
Vector3d b = a; // теперь вектор b содержит (1, 2, 3)
```

Доступны следующие константы:

- `Vector3d.Zero` – нулевой вектор;
- `Vector3d.XAxis` – единичный вектор, направленный вдоль оси абсцисс;
- `Vector3d.YAxis` – единичный вектор, направленный вдоль оси ординат;
- `Vector3d.ZAxis` – единичный вектор, направленный вдоль оси аппликат;

Пример:

```
Vector3d vector = Vector3d.Zero;
```

Скалярное произведение

Скалярным произведением двух векторов называется число, равное произведению длин этих векторов на косинус угла между ними:

$$\vec{a}\vec{b} = |\vec{a}||\vec{b}|\cos\phi$$

Пример:

```
Vector3d a = new Vector3d(5.67, 84.1, 5.34);
Vector3d b = new Vector3d(43.2, 5.8, 84.12);

// Первый вариант, через метод класса
Number result1 = a.DotProduct(b);

// Второй вариант, через статическую функцию
Number result2 = Vector3d.DotProduct(a, b);
```

Скалярное произведение двух векторов $\vec{a}(a_1;a_2;a_3)$ и $\vec{b}(b_1;b_2;b_3)$ заданных своими координатами, может быть вычислено по формуле:

$$(\vec{a}, \vec{b}) = a_1b_1 + a_2b_2 + a_3b_3$$

Векторное произведение

Векторным произведением вектора a на вектор b называется вектор c, удовлетворяющий следующим требованиям:

- длина вектора c равна произведению длин векторов a и b на синус угла φ между ними

$$|\vec{c}| = |\vec{a}|\,|\vec{b}|\sin\phi$$

- вектор c ортогонален каждому из векторов a и b
- вектор c направлен так, что тройка векторов abc является правой.

Пример:

```
Vector3d a = new Vector3d(5.67, 84.1, 5.34);
Vector3d b = new Vector3d(43.2, 5.8, 84.12);

// Первый вариант, через метод класса
Vector3d result1 = a.CrossProduct(b);

// Второй вариант, через статическую функцию
Vector3d result2 = Vector3d.CrossProduct(a, b);
```

Сравнение векторов

Для того чтобы узнать, равны ли два вектора друг другу, можно использовать оператор равно, например:

```
Vector3d a = new Vector3d(5.67, 84.1, 57);
Vector3d b = new Vector3d(43.2, 5.8, 0.67);
Boolean result = (a == b);
```

В расчетах используется форма с плавающей точкой для хранения чисел, что приводит к тому, что в результате расчетов значение числа накапливает погрешность. Данная погрешность приводит к тому, что простое сравнение может дать ложный результат, для решения этой проблемы следует использовать метод `ApproxEqual`

Сравнение с учетом накопленной погрешности:

```
// Сравнение с толерантностью по умолчанию
Boolean result1 = Vector3d.ApproxEqual(v, u);
// Сравнение с указанной толерантностью
Boolean result2 = Vector3d.ApproxEqual(v, u, 0.0001);
```

Доступно также покомпонентное сравнение, когда все компоненты одного вектора меньше другого (или больше), доступны > , >=, <, <=, например:

```
Boolean result1 = v < u;
Boolean result2 = v >= u;
```

Нормализация

Нормализация - это преобразование, строящее вектор коллинеарный (параллельный) данному вектору, но с единичной длинной, например:

```
Vector3d a = new Vector3d(5.67, 84.1, 0.5);

// Нормализуем текущий вектор
a.Normalize();
// Через статическую функцию, а результат
// помещаем в другой вектор
Vector3d b = Vector3d.Normalize(a);
```

Длину вектора можно определить с помощью функции `GetLength()`

Сложение векторов

Для проведения операций сложения векторов определены операторы «+» и «-».

```
Vector3d a = new Vector3d(5.67, 84.1);
Vector3d b = new Vector3d(43.2, 5.8);
Vector3d c = a + b;
```

Операции со скалярами

Операция сложения со скаляром представляет собой прибавление к каждому компоненту вектора этот скаляр, т.е.:

```
Vector3d a = new Vector3d(1, 1, 1);
a = a + 1; // теперь a = (2, 2, 2)
```

Определены также аналогичные операции «-»,«*» и «/».

Трехмерная прямая

Значимый тип. Трехмерная прямая задается с помощью точки и направления.

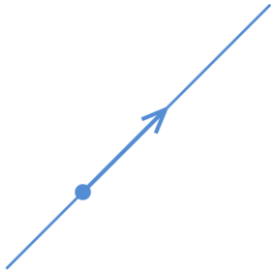


Рисунок 161. Трехмерная прямая

Описание построения	Для того чтобы создать прямую, нужно передать два трехмерных вектора, которые будут задавать соответственно точку на конструируемой прямой и направление.

Пример	`Vector3d point = Vector3d.Zero;`
	`Vector3d direction = new Vector3d(1.0, 1.0, 1.0);`
	`Line3d line = new Line3d(point, direction);`

Существует другой способ инициализации, передать по порядку компоненты векторов, например, такая же прямая получится при такой инициализации:

```
Line3d line = new Line3d(0, 0, 0, 1, 1, 1);
```

Трехмерная окружность

Значимый тип. Трехмерная окружность задается с помощью указания ее центра и радиуса.

Описание построения	Для того чтобы создать окружность, нужно передать трехмерный вектор, который будет задавать соответственно центр, а также радиус.

Пример	`Number radius = 5.0;`
	`Vector3d center = new Vector3d(1.0, 1.0, 1.0);`
	`Circle3d circle = new Circle3d(center, radius);`

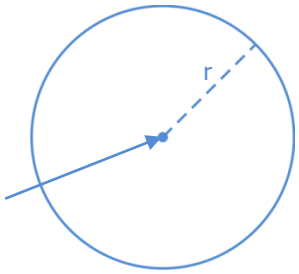


Рисунок 162. Трехмерная окружность

Существует другой способ инициализации, передать по порядку компоненты вектора, например, такая же окружность получиться при такой инициализации:

```
Circle3d circle = new Circle3d(1, 1, 1, 5);
```

Плоскость

Значимый тип. Плоскость задается с помощью указания расположенной на ней точки и нормали.

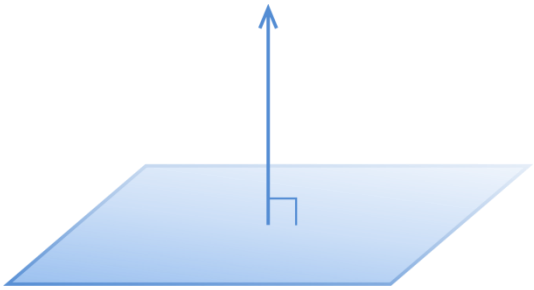


Рисунок 163. Плоскость

Описание построения	Для того чтобы создать плоскость, нужно передать два трехмерных вектора, которые будут задавать соответственно точку на ней и нормаль.

Пример	`Vector3d point = new Vector3d(0.0, 5.0, 8.0);`
	`Vector3d normal = new Vector3d(1.0, 0.0, 0.0);`
	`Plane plane = new Plane(point, normal);`

Описание построения	Существует другой способ инициализации, передать по порядку компоненты вектора.
Пример	`Plane plane = new Plane(0.0, 0.5, 8.0, 1.0, 0.0, 0.0);`
Описание построения	Существует возможность получать плоскости образованные координатными осями: XY, XZ, YZ.
Пример	`Plane plane = Plane.XY;`
Описание построения	Дополнительно существует способ задать плоскость через прямую и точку вне этой прямой (версия 2.0).

Пример	`Vector3d point = new Vector3d(0.0, 5.0, 8.0);`
	`Line3d line = new Line3d(Vector3d.Zero, Vector3d.ZAxis);`
	`Plane plane = new Plane(point, line);`

Сфера

Значимый тип. Сфера задается с помощью указания ее центра и радиуса.

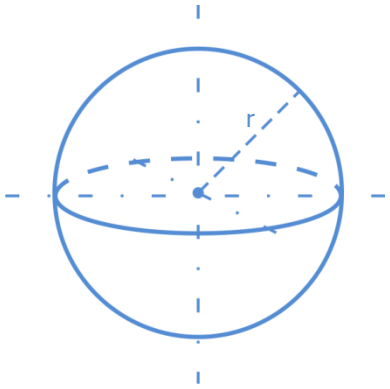


Рисунок 164. Сфера

Описание построения	Для того чтобы создать сферу, нужно передать трехмерный вектор, который будет задавать соответственно центр, а также радиус.

Пример	`Number radius = 5.0;`
	`Vector3d center = new Vector3d(1.0, 1.0, 1.0);`
	`Sphere sphere = new Sphere(center, radius);`

Существует другой способ инициализации, передать по порядку компоненты вектора, например, такая же сфера получиться при такой инициализации:

```
Sphere sphere = new Sphere (1, 1, 1, 5);
```

Цилиндр

Значимый тип. Цилиндр задается с помощью указания его оси (трехмерная прямая) и радиус.

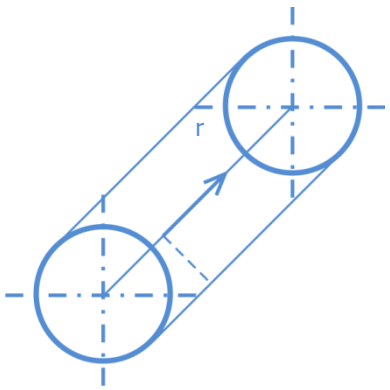


Рисунок 165. Цилиндр

Описание	Для того чтобы создать цилиндр, нужно передать трехмерную прямую, которая будет задавать соответственно ось, а также

построения	радиус.

Пример	`Number radius = 5.0;`
	`Line3d axis = new Line3d(Vector3d.Zero, Vector3d.ZAxis);`
	`Cylinder cylinder = new Cylinder(axis, radius);`

Описание построения	Второй способ инициализации. Необходимо передать по порядку точку на оси, направление оси и радиус.
Пример	`Cylinder cylinder = new Cylinder(Vector3d.Zero, Vector3d.ZAxis, 5.0);`
Описание построения	Третий способ предполагает передачу тех же элементов что и при втором способе, только по отдельным компонентам.
Пример	`Cylinder cylinder = new Cylinder(0, 0, 0, 0, 0, 1, 5);`

Конус

Значимый тип. Конус задается с помощью указания точки симметрии, ориентации и угла.

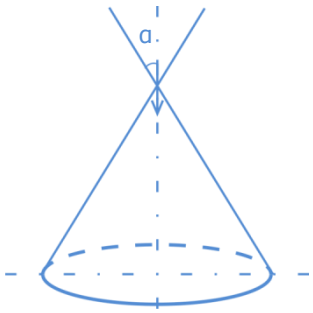


Рисунок 166. Конус

Описание построения	Для того чтобы создать конус, нужно передать трехмерный вектор, который будет задавать точку симметрии, вектор, который будет задавать ориентацию и угол в радианах.

Пример	`Number angle = Basic.Pi / 8.0;`
	`Vector3d center = new Vector3d(1.0, 1.0, 1.0);`
	`Vector3d direction = new Vector3d(0.0, 0.0, 1.0);`
	`Cone cone = new Cone(center, direction, angle);`

Описание построения	Второй способ предполагает передачу тех же элементов что и при первом способе, только по отдельным компонентам.
Пример	`Cone cone = new Cone(1.0, 1.0, 1.0, 0.0, 0.0, 1.0, Basic.Pi / 8.0);`
Описание построения	Зная два радиуса двух сечений и расстояние между ними можно создать конус следующим способом. Следует передать данные: центр первого сечения, направление конуса, радиус первого сечения, радиус второго сечения и расстояние между ними.

Пример	`Vector3d center = new Vector3d(1.0, 1.0, 1.0);`
	`Vector3d direction = new Vector3d(0.0, 0.0, 1.0);`
	`Number radius1 = 5.0;`
	`Number radius2 = 7.0;`
	`Number distance = 1.0;`

	`Cone cone = new Cone(center, direction, radius1, radius2, distance);`

Тор (версия 4.0)

Значимый тип. Тор задается через центр, ориентацию оси, больший и меньший радиусы.

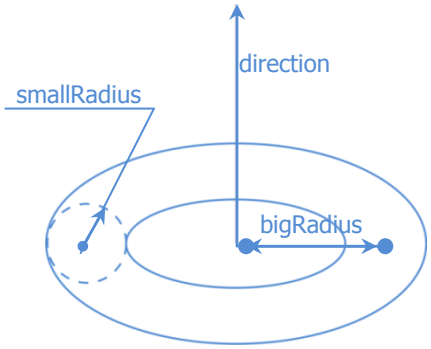


Рисунок 167. Тор

Описание построения	Для того чтобы создать тор, нужно передать трехмерный вектор, который будет задавать соответственно центр, вектор, который будет задавать направление оси, а также радиусы.

Пример	`Number bigRadius = 5.0;`
	`Number smallRadius = 1.0;`
	`Vector3d center = new Vector3d(1.0, 1.0, 1.0);`
	`Vector3d direction = new Vector3d(1.0, 0.0, 0.0);`
	`Torus torus = new Torus(center, direction, bigRadius, smallRadius);`

Двухмерный сплайн (версия 5.0)

Значимый тип. Задается с помощью двух функций аппроксимации и интерполяции.

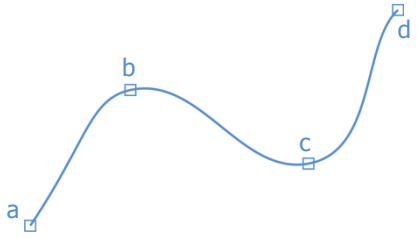


Рисунок 168. Двухмерный сплайн

Описание построения	Для интерполяции существует функция `Spline2d.Interpolate(…)`. Точки можно передавать просто, перечисляя их через запятую. В результате данного метода будет построен интерполяционный сплайн, проходящий через данные точки.

Пример	`Vector2d a = new Vector2d(1.0, 1.0);`
	`Vector2d b = new Vector2d(1.0, 2.0);`
	`Vector2d c = new Vector2d(1.0, 5.0);`
	`Vector2d d = new Vector2d(2.0, 3.0);`
	`Spline2d spline = Spline2d.Interpolate(a, b, c, d);`

Описание построения	Можно передать массив точек

Пример	`Vector2d[] array = new Vector2d[]`
	`{new Vector2d(1.0, 1.0),`
	`new Vector2d(1.0, 2.0),`
	`new Vector2d(1.0, 5.0),`
	`new Vector2d(2.0, 3.0)`
	`};Spline2d spline = Spline2d.Interpolate(array);`

Методом `Spline2d.Approximate(...)` можно построить среднеквадратичный сплайн. Передаваемые параметры аналогичны методу интерполяции.

Трехмерный сплайн (версия 5.0)

Значимый тип. Задается с помощью двух функций аппроксимации и интерполяции.

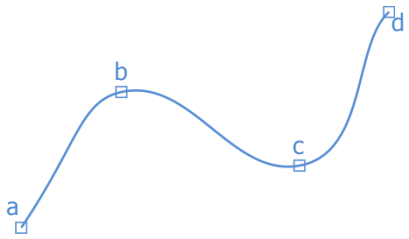


Рисунок 169. Интерполяционный сплайн

Описание построения	Для интерполяции существует функция `Spline3d.Interpolate(...)`. Точки можно передавать просто, перечисляя их через запятую. В результате данного метода будет построен интерполяционный сплайн, проходящий через данные точки.

Пример	`Vector3d a = new Vector2d(1.0, 1.0, -5.0);`
	`Vector3d b = new Vector2d(1.0, 2.0, 1.0);`
	`Vector3d c = new Vector2d(1.0, 5.0, 4.0);`
	`Vector3d d = new Vector2d(2.0, 3.0, 5.0);`
	`Spline3d spline = Spline3d.Interpolate(a, b, c, d);`

Описание построения	Можно передать массив точек

Пример	`Vector3d[] array = new Vector3d[]`
	`{`
	`new Vector3d(1.0, 1.0, -2.0),`
	`new Vector3d(1.0, 2.0, 1.0),`
	`new Vector3d(1.0, 5.0, 1.0),`

```
 new Vector3d(2.0, 3.0, 4.0)
};
   Spline3d spline = Spline3d.Interpolate(array);
```

Методом `Spline3d.Approximate(…)` можно построить среднеквадратичный сплайн. Передаваемые параметры аналогичны методу интерполяции.

Сплайновая поверхность (версия 5.0)

Значимый тип. Не имеет конструктора, но может поступать на вход программы, использоваться в расчетах.

```
SplineSurface surface = null;
```

Аппроксимация элементов

Аппроксимация элементов – это нахождение заменяющих элементов по измеренным точкам на их поверхности. Существует несколько различных методов аппроксимации геометрического элемента. Ниже описаны три основных метода (нахождение среднего, прилегающего или минимальной зоны).

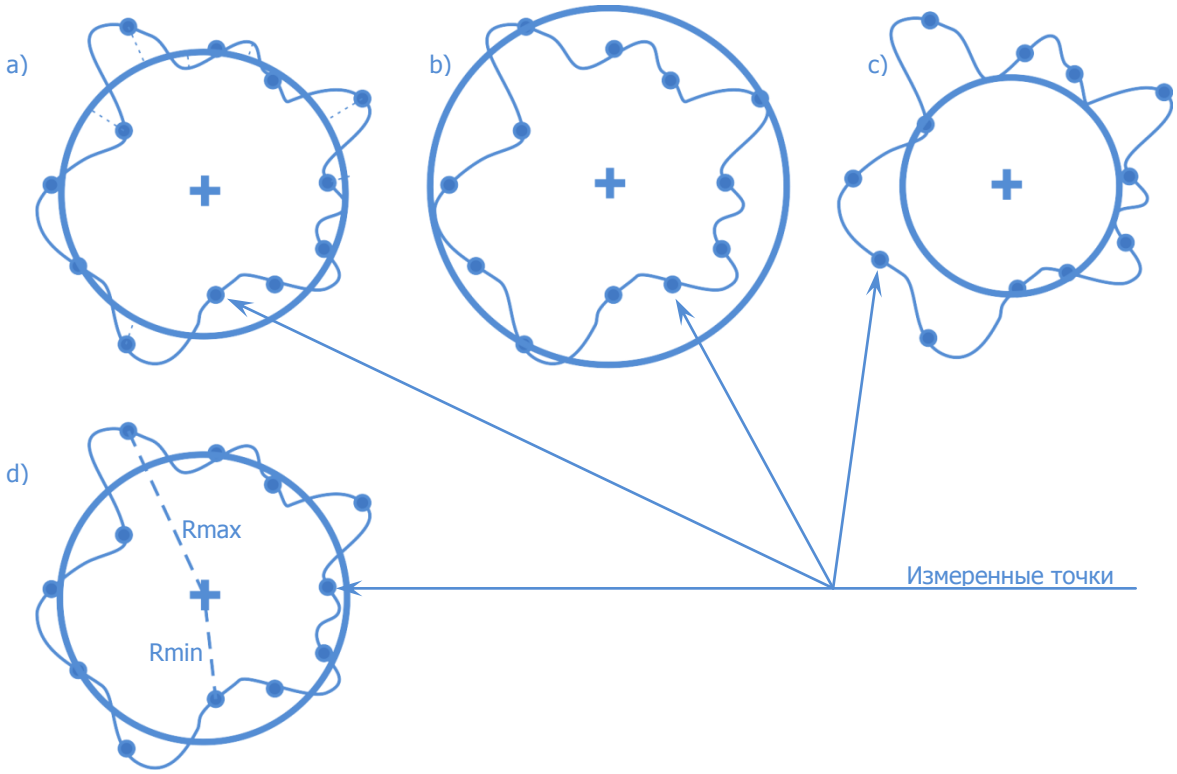


Рисунок 170. (методы аппроксимации в соответствии с ISO 6318: a) среднеквадратичный элемент b)c) минимально описанный и максимально вписанный элемент d)минимальная зона)

Средний элемент (аппроксимация среднеквадратичными кривыми и поверхностями (англ. least-square fitting)) – поверхность, имеющая номинальную форму и такие размеры и/или расположение, чтобы сумма квадратов расстояний между реальным и средним элементами в пределах нормируемого участка имела минимальное значение. Среднеквадратичная поверхность (кривая) является решением задачи минимизации суммы квадратов расстояний от измеренной точки до поверхности (кривой):

$$\sum_{i=1}^{n} {d_i}^2 \rightarrow min$$

где n – количество измеренных точек,
d_i *– ортогональное расстояние от i-ой измеренной точки до искомой поверхности.*

Прилегающей поверхностью (кривой) называется поверхность, имеющая форму номинальной поверхности, соприкасающаяся с реальной поверхностью (кривой) и расположенная вне материала детали так, что отклонение от нее наиболее удаленной точки реальной поверхности в пределах нормируемого участка имеет минимальное значение. Это понятие относится к прилегающей плоскости, прямой, конуса, однако указанное условие минимального значения отклонения не распространяется на отклонения формы цилиндра, сферы и окружности.

Данная задача есть задача минимизации нормы Чебышева:

$$sup_{i=1}^{n}|d_i| \rightarrow min$$

где d_i - расстояние от измеренной точки до поверхности (кривой),
n – количество измеренных точек.

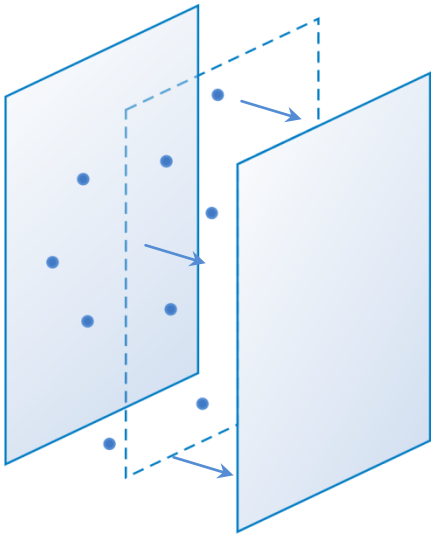


Рисунок 171. Прилегающие плоскости

Прилегающим цилиндром (сферой, окружностью) называется цилиндр (сфера, окружность) минимального диаметра, описанного вокруг реальной наружной поверхности, или максимального диаметра, вписанного в реальную внутреннюю поверхность (см. **Рисунок 172**).

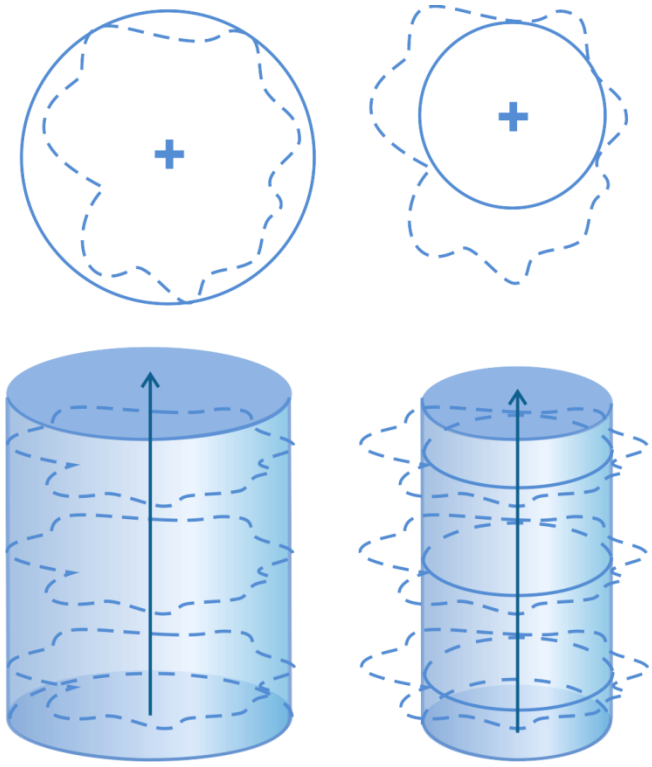


Рисунок 172. Прилегающие окружности и цилиндры

Замечание о различии в стандартах. Необходимо заметить, что российские стандарты вводят понятия прилегающих, как основных заменяющих элементов для расчетов, а также минимальные зоны и средние элементы (например, ГОСТ 24642-81), причем в случае цилиндра, сферы и окружности функции минимизации меняются между понятиями прилегающих и минимальных зон. Международные стандарты (ISO 6318, см. **Рисунок 173**) определяют более четко в математическом смысле заменяющие элементы: минимальная зона (отклонение наиболее удаленной точки имеет минимальное значение), максимально вписанного, минимально описанного элемента и среднеквадратичного элемента. **ТЕХНО**коорд™ придерживается терминологии описанной в ГОСТ. Таким образом, прилегающим плоскостям, прямым, конусам соответствует понятие минимальной зоны, а прилегающим окружностям, цилиндрам, сферам соответствуют минимально описанные и максимально вписанные элементы. Окружности, сферы и цилиндры минимальной зоны, а также средние элементы не имеют различий в стандартах.

Вместо прилегающего цилиндра (окруности, сферы) в качестве базы для определения отклонений допускается также использовать цилиндр минимальной зоны. Цилиндр (окружность, сфера) минимальной зоны – цилиндр (конус, сфера, тор), соприкасающийся с реальной поверхностью и расположенный вне материала так, чтобы наибольшее расстояние между реальной поверхностью и заменяющим элементом имело минимальное значение (см. **Рисунок 173**).

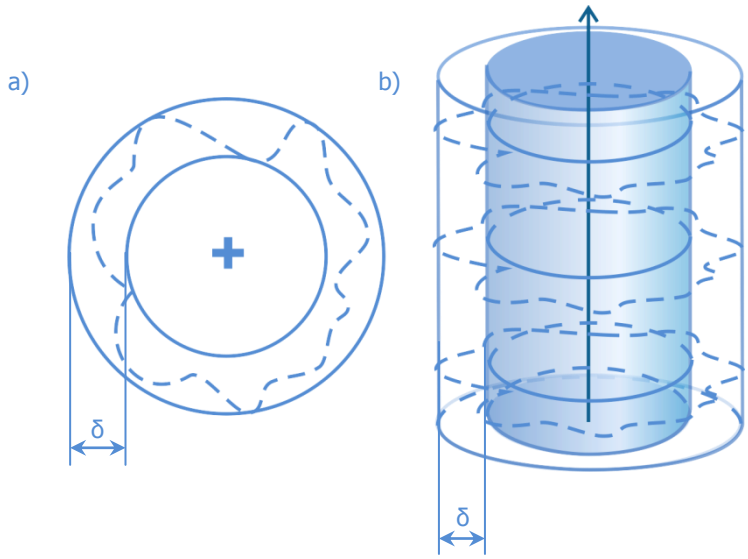


Рисунок 173. a - окружности минимальной зоны, b - цилиндры минимальной зоны

Средние элементы

Средний элемент – поверхность (кривая), имеющая номинальную форму и такие размеры и/или расположение, чтобы сумма квадратов расстояний между реальным и средним элементами в пределах нормируемого участка имела минимальное значение.

Для нахождения средних элементов предназначен класс Average. Для аппроксимации элемента необходимо передать набор точек с поверхности элемента. Иногда требуется уточняющая информация (например, для нахождения среднего цилиндра необходимо указать направляющий вектор его оси).

Двухмерная прямая

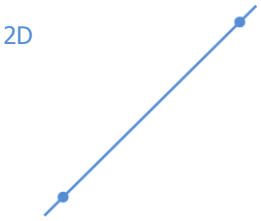


Рисунок 174. Средняя двухмерная прямая

Минимальное количество точек	2
Рекомендации	Дополнительных ограничений на измеренные точки нет. Для более точного измерения желательно выбирать точки, расположенные наиболее далеко друг от друга.
Пример	`Vector2d[] points = new Vector2d[] {...};` `Line2d line = Average.Line(points);`

Трехмерная прямая

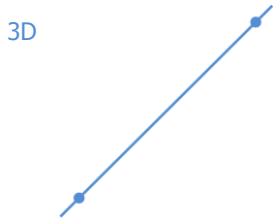


Рисунок 175. Средняя трехмерная прямая

Минимальное количество точек	2
Рекомендации	Дополнительных ограничений на измеренные точки нет. Для более точного измерения желательно выбирать точки, расположенные наиболее далеко друг от друга.
Пример	`Vector3d[] points = new Vector3d[] {...};` `Line3d line = Average.Line(points);`

Двухмерная окружность

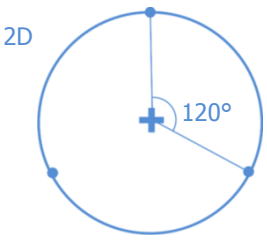


Рисунок 176. Средняя двухмерная окружность

Минимальное количество точек	3
Рекомендации	Точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой. Для более точного измерения желательно выбирать точки, расположенные на более далеком расстоянии друг от друга.
Пример	`Vector2d[] points = new Vector2d[] {...};` `Circle2d circle = Average.Circle(points);`

Трехмерная окружность

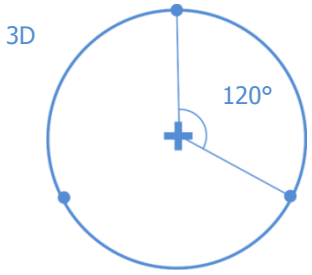


Рисунок 177. Средняя трехмерная окружность

Минимальное количество точек	3
Рекомендации	Точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой. Для более точного измерения желательно выбирать точки, расположенные на более далеком расстоянии друг от друга.
Пример	`Vector3d[] points = new Vector3d[] {...};` `Circle3d line = Average.Circle(points);`

Плоскость

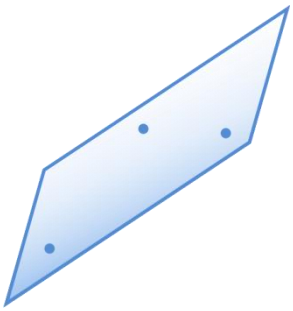


Рисунок 178. Средняя плоскость

Минимальное количество точек	3
Рекомендации	Точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой. Дополнительных ограничений на измеренные точки нет. Для более точного измерения желательно выбирать

	точки, расположенные наиболее далеко друг от друга.
Пример	`Vector3d[] points = new Vector3d[] {...};` `Plane plane = Average.Plane(points);`

Сфера

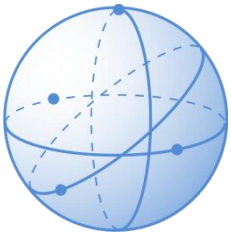


Рисунок 179. Средняя сфера

Минимальное количество точек	4
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать в одной плоскости. Для более точного измерения желательно набирать точки, расположенные на более далеком расстоянии друг от друга.
Пример	`Vector3d[] points = new Vector3d[] {...};` `Sphere sphere = Average.Sphere(points);`

Цилиндр

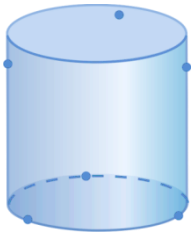


Рисунок 180. Средний цилиндр

Минимальное количество точек	6
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать в одной плоскости. Вырожденные случае возникают, если точки выбраны в одном

	сечении поверхности. Для более точного измерения желательно следовать следующим рекомендациям: выбирать точки в нескольких сечений, ортогональным оси вращения поверхности; второе сечение выбирать так, чтобы расстояние вдоль образующей линии между ними было наибольшим.

Пример	`Vector3d[] points = new Vector3d[] {...};`
	`Cylinder cylinder = Average.Cylinder(points);`
	
	В функцию передаются точки, измеренные на поверхности и направляющий вектор предполагаемой оси цилиндра.

Конус

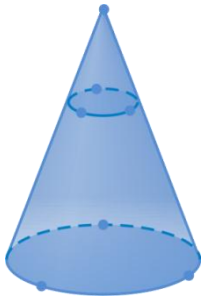


Рисунок 181. Средний конус

Минимальное количество точек	7
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать в одной плоскости. Вырожденные случаи возникают, если точки выбраны в одном сечении поверхности. Для более точного измерения желательно следовать следующим рекомендациям: выбирать точки в нескольких сечениях, ортогональных оси вращения поверхности; второе сечение выбирать так, чтобы расстояние вдоль образующей линии между ними было наибольшим.

Пример	`Vector3d[] points = new Vector3d[] {...};`
	`Cone cone = Average.Cone(points);`
	
	В функцию передаются точки, измеренные на поверхности и направляющий вектор предполагаемой оси конуса.

Тор (версия 4.0)

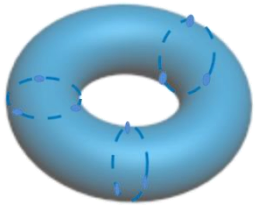


Рисунок 182. Средний тор

Минимальное количество точек	8
Рекомендации	Необходимо взять равноудаленные друг от друга точки, как минимум на трех сечениях.
Пример	`Vector3d[] points = new Vector3d[] {...};` `Torus torus = Average.Torus(points);`

Двухмерный сплайн (версия 5.0)

Рисунок 183. Средний двухмерный сплайн

Минимальное количество точек	2
Рекомендации	Следует взять равноудаленные точки

Пример	`Vector2d[] points = new Vector2d[] {...};`
	`Spline2d spline = Average.Spline(points);`
	Функция строит среднеквадратический B-сплайн.

Трехмерный сплайн (версия 5.0)

Рисунок 184. Средний трехмерный сплайн

Минимальное количество точек	2

Рекомендации	Следует взять равноудаленные точки

Пример	`Vector3d[] points = new Vector3d[] {...};`
	`Spline3d spline = Average.Spline(points);`
	Функция строит среднеквадратический B-сплайн.

Прилегающие элементы (версия 3.0)

Прилегающей поверхностью (кривой) называется поверхность, имеющая форму номинальной поверхности, соприкасающаяся с реальной поверхностью (кривой) и расположенная вне материала детали так, что отклонение от нее наиболее удаленной точки реальной поверхности в пределах нормируемого участка имеет минимальное значение. Это понятие относится к прилегающей плоскости, прямой, конуса, однако указанное условие минимального значения отклонения не распространяется на отклонения формы цилиндра, сферы и окружности.

Прилегающим цилиндром (сферой, окружностью) называется цилиндр (сфера, окружность) минимального диаметра, описанного вокруг реальной наружной поверхности, или максимального диаметра, вписанного в реальную внутреннюю поверхность.

Для нахождения прилегающих элементов предназначен класс `Superimposed`. Для аппроксимации элемента необходимо передать набор точек с поверхности элемента. Иногда требуется уточняющая информация.

Двухмерная описанная прилегающая окружность (версия 3.0)

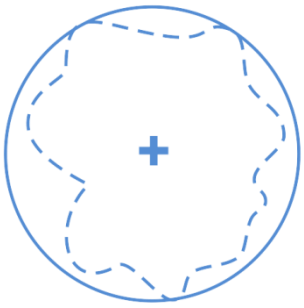


Рисунок 185. Описанная прилегающая окружность

Минимальное количество точек	3
Рекомендации	Точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой. Точки необходимо выбирать так, чтобы центр окружности попал в выпуклую оболочку измеренных точек (например, угол между точками должен быть не более 120°). Для более точного измерения желательно выбирать точки,

	расположенные на более далеком расстоянии друг от друга (см. **Рисунок 186**).
Пример	`Vector2d[] points = new Vector2d[] {...};` `Circle2d circle = Superimposed.EscribedCircle(points);`

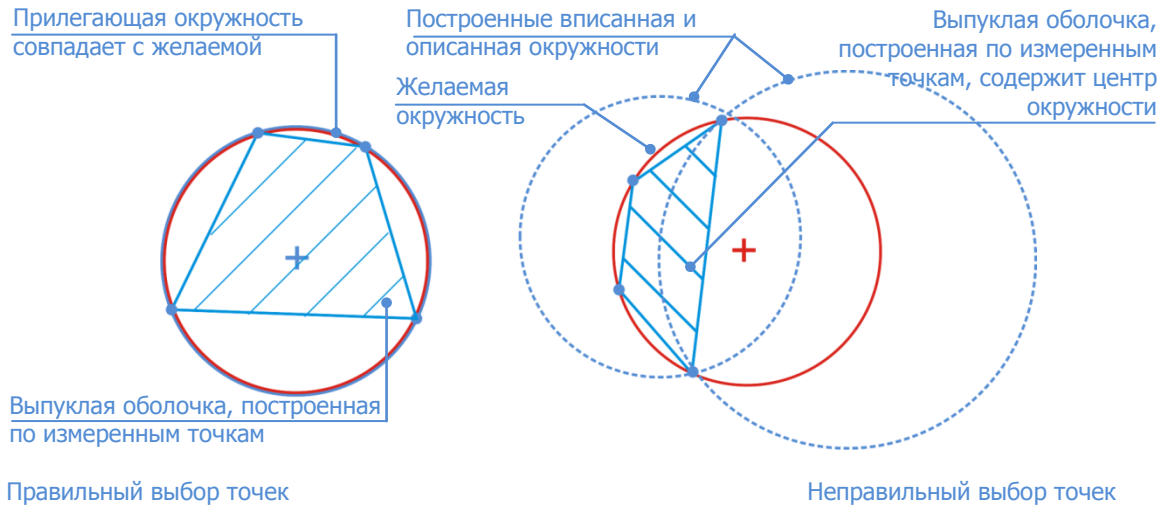


Рисунок 186. Пример правильного и неправильного выбора точек для прилегающей окружности

Двухмерная вписанная прилегающая окружность (версия 3.0)

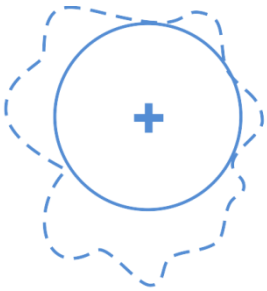


Рисунок 187. Вписанная прилегающая окружность

Минимальное количество точек	3
Рекомендации	Точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой. Точки необходимо выбирать так, чтобы центр окружности попал в выпуклую оболочку измеренных точек (например, угол между точками должен быть не более 120°). Для более точного измерения желательно выбирать точки, расположенные на более далеком расстоянии друг от друга (см. **Рисунок 186**).
Пример	`Vector2d[] points = new Vector2d[] {...};` `Circle2d circle = Superimposed.InscribedCircle(points);`

Описанный прилегающий цилиндр (версия 3.0)

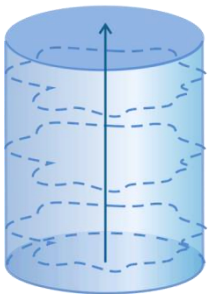


Рисунок 188. Описанный прилегающий цилиндры

Минимальное количество точек	6
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать в одной плоскости. Вырожденные случае возникают, если точки выбраны в одном сечении поверхности. Для более точного измерения желательно следовать следующим рекомендациям: выбирать точки в нескольких сечениях, ортогональных оси вращения поверхности. Точки в первом сечении необходимо выбирать так, чтобы центр сечения попал в выпуклую оболочку измеренных точек (например, угол между точками должен быть не более 120°), второе сечение выбирать так, чтобы расстояние вдоль образующей линии между ними было наибольшим (см. **Рисунок 186**).
Пример	`Vector3d[] points = new Vector3d[] {...};` `Cylinder cylinder=Superimposed.EscribedCylinder(points);`

Вписанный прилегающий цилиндр (версия 3.0)

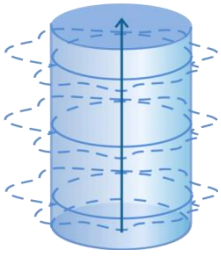


Рисунок 189. Вписанный прилегающий цилиндры

Минимальное количество точек	6
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать в одной плоскости. Вырожденные случаи возникают, если точки выбраны в одном сечении поверхности. Для более точного измерения желательно следовать следующим рекомендациям: выбирать точки в нескольких сечений, ортогональным оси вращения поверхности; второе сечение выбирать так, чтобы расстояние вдоль образующей линии между ними было наибольшим (см. **Рисунок 186**). Точки в первом сечении необходимо выбирать так, чтобы центр сечения попал в выпуклую оболочку измеренных точек (например, угол между точками должен быть не более 120°).

Пример	`Vector3d[] points = new Vector3d[] {...};`
	`Cylinder cylinder =`
	`Superimposed.InscribedCylinder(points);`

Описанная прилегающая сфера (версия 3.0)

Рисунок 190. Описанная прилегающая сфера

Минимальное количество точек	4
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать на одной окружности. Для более точного измерения желательно набирать точки, расположенные на более далеком расстоянии друг от друга. Точки необходимо выбирать так, чтобы центр сферы попал в выпуклую оболочку измеренных точек (**Рисунок 192**).
Пример	`Vector3d[] points = new Vector3d[] {...};` `Sphere sphere = Superimposed.EscribedSphere(points);`

Вписанная прилегающая сфера (версия 3.0)

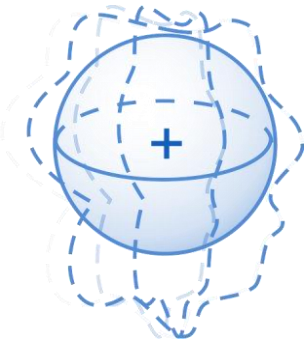


Рисунок 191. Вписанная прилегающая схема

Минимальное количество точек	4
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать на одной окружности. Для более точного измерения желательно набирать точки, расположенные на более далеком расстоянии друг от друга. Точки необходимо выбирать так, чтобы центр сферы попал в выпуклую оболочку измеренных точек (**Рисунок 192**).
Пример	`Vector3d[] points = new Vector3d[] {...};` `Sphere sphere = Superimposed.InscribedSphere(points);`

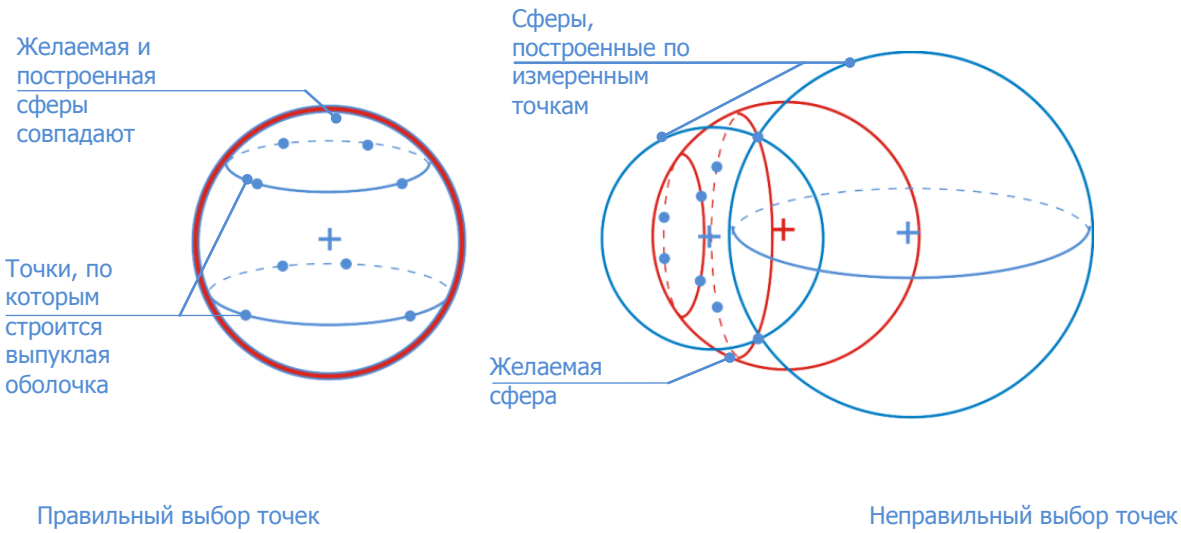


Рисунок 192. Пример правильного и неправильного выбора точек для прилегающей сферы

Вписанный прилегающий конус (версия 5.0)

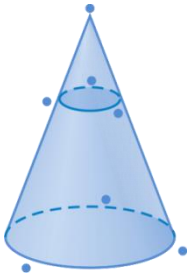


Рисунок 193. Вписанный прилегающий конус

Минимальное количество точек	7
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать в одной плоскости. Вырожденные случаи возникают, если точки выбраны в одном сечении поверхности. Для более точного измерения желательно следовать следующим рекомендациям: выбирать точки в нескольких сечений, ортогональным оси вращения поверхности; второе сечение выбирать так, чтобы расстояние вдоль образующей линии между ними было наибольшим.
Пример	`Vector3d[] points = new Vector3d[] {...};` `Cone cone = Superimposed.InscribedCone(points);`

Описанный прилегающий конус (версия 5.0)

Рисунок 194. Описанный прилегающий конус

Минимальное количество точек	7
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой;

	• не должны быть копланарными, т.е. лежать в одной плоскости. Вырожденные случаи возникают, если точки выбраны в одном сечении поверхности. Для более точного измерения желательно следовать следующим рекомендациям: выбирать точки в нескольких сечений, ортогональным оси вращения поверхности; второе сечение выбирать так, чтобы расстояние вдоль образующей линии между ними было наибольшим.
Пример	`Vector3d[] points = new Vector3d[] {...};` `Cone cone = Superimposed.EscribedCone(points);`

Двухмерная прилегающая прямая (версия 5.0)

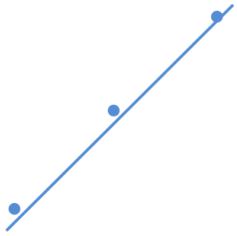


Рисунок 195. Двухмерная прилегающая прямая

Минимальное количество точек	2
Рекомендации	Дополнительных ограничений на измеренные точки нет. Для более точного измерения желательно выбирать точки, расположенные наиболее далеко друг от друга.

Пример	`Vector2d[] points = new Vector2d[] {...};`
	`Vector2d pointInMaterial = new Vector2d(30, 50);`
	`Line2d line = Superimposed.Line(points,pointInMaterial);`

Прилегающая плоскость (версия 5.0)

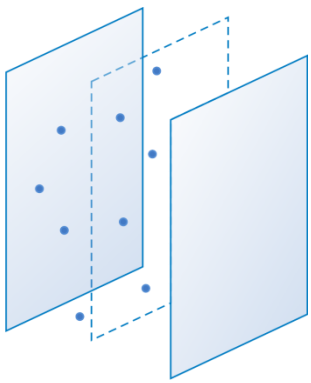


Рисунок 196. Прилегающая плоскость

Минимальное количество точек	3
Рекомендации	Дополнительных ограничений на измеренные точки нет. Для более точного измерения желательно выбирать точки, расположенные наиболее далеко друг от друга.

Пример	`Vector3d[] points = new Vector3d[] {...};`
	`Vector3d pointInMaterial = new Vector3d(30, 50, 1);`
	`Plane plane = Superimposed.Plane(points, pointInMaterial);`

Минимальные зоны (версия 4.0)

Вместо прилегающего цилиндра (конуса, сферы, тора) в качестве базы для определения отклонений допускается также использовать цилиндр минимальной зоны. Цилиндр (конус, сфера, тор) минимальной зоны – цилиндр (конус, сфера, тор), соприкасающийся с реальной поверхностью и расположенный вне материала так, чтобы наибольшее расстояние между реальной поверхностью и заменяющим элементом имело минимальное значение.

Для нахождения минимальных зон предназначен класс `MinimalZone`. Для аппроксимации элемента необходимо передать набор точек с поверхности элемента. Иногда требуется уточняющая информация.

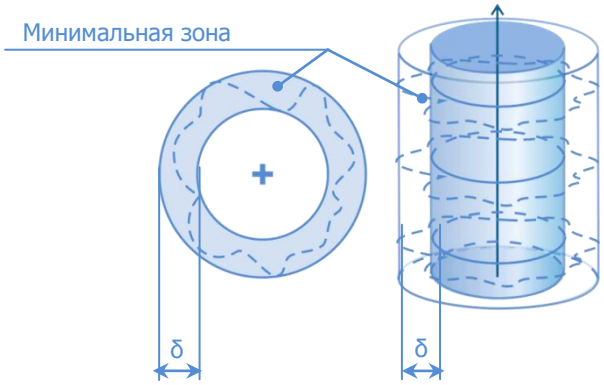


Рисунок 197. a - окружности минимальной зоны, b - цилиндры минимальной зоны

Двухмерная описанная окружность (версия 4.0)

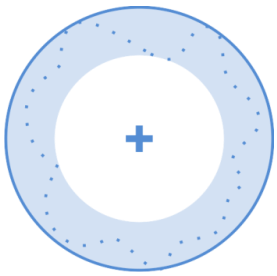


Рисунок 198. Двумерная описанная окружность минимальной зоны

Минимальное количество точек	3
Рекомендации	Точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой. Для более точного измерения желательно выбирать точки, расположенные на более далеком расстоянии друг от друга.
Пример	`Vector2d[] points = new Vector2d[] {...};` `Circle2d circle = MinimalZone.EscribedCircle(points);`

Двухмерная вписанная окружность (версия 4.0)

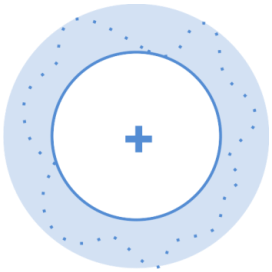


Рисунок 199. Двумерная вписанная окружность минимальной зоны

Минимальное количество точек	3
Рекомендации	Точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой. Для более точного измерения желательно выбирать точки, расположенные на более далеком расстоянии друг от друга.
Пример	`Vector2d[] points = new Vector2d[] {...};` `Circle2d circle = MinimalZone.InscribedCircle(points);`

Описанный цилиндр (версия 5.0)

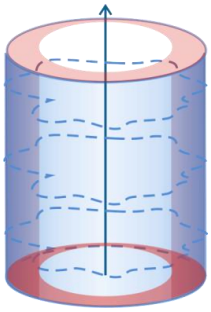


Рисунок 200. Описанный прилегающий цилиндр минимальной зоны

Минимальное количество точек	6

Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать в одной плоскости. Вырожденные случае возникают, если точки выбраны в одном сечении поверхности. Для более точного измерения желательно следовать следующим рекомендациям: выбирать точки в нескольких сечений, ортогональным оси вращения поверхности; второе сечение выбирать так, чтобы расстояние вдоль образующей линии между ними было наибольшим.
Пример	`Vector3d[] points = new Vector3d[] {...};` `Cylinder cylinder =MinimalZone.EscribedCylinder(points);`

Вписанный цилиндр (версия 5.0)

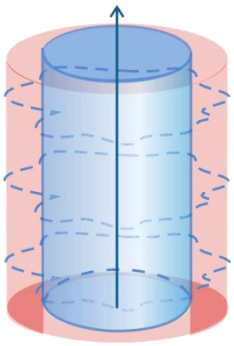


Рисунок 201. Вписанный прилегающий цилиндр минимальной зоны

Минимальное количество точек	6
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать в одной плоскости. Вырожденные случае возникают, если точки выбраны в одном сечении поверхности. Для более точного измерения желательно следовать следующим рекомендациям: выбирать точки в нескольких сечений, ортогональным оси вращения поверхности; второе сечение выбирать так, чтобы расстояние вдоль образующей линии между ними было наибольшим.

Пример	`Vector3d[] points = new Vector3d[] {...};`
	`Cylinder cylinder =`
	`MinimalZone.InscribedCylinder(points);`

Описанная сфера (версия 4.0)

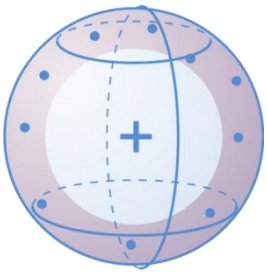


Рисунок 202. Описанная прилегающая сфера минимальной зоны

Минимальное количество точек	5
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать на одной окружности. Для более точного измерения желательно набирать точки, расположенные на более далеком расстоянии друг от друга.
Пример	`Vector3d[] points = new Vector3d[] {...};` `Sphere sphere = MinimalZone.EscribedSphere(points);`

Вписанная сфера (версия 4.0)

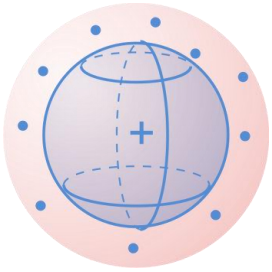


Рисунок 203. Вписанная прилегающая сфера минимальной зоны

Минимальное количество точек	5
Рекомендации	Ограничения на набор минимального количества измеренных точек: • точки не должны быть коллинеарными, т.е. не должны лежать на одной прямой; • не должны быть копланарными, т.е. лежать на одной окружности. Для более точного измерения желательно набирать точки, расположенные на более далеком расстоянии друг от друга.
Пример	`Vector3d[] points = new Vector3d[] {...};` `Sphere sphere = MinimalZone.InscribedSphere(points);`

Операции с элементами геометрии

Нахождение пересечений

Операция предполагает нахождение примитива – результата операции пересечения. Ниже приведена таблица допустимых операций:

	Двухмерная прямая	Двухмерный сплайн	Двухмерная окружность
Двухмерная прямая	Двухмерная точка	Массив точек	Массив точек
Двухмерный сплайн	Массив точек	Массив точек	Массив точек
Двухмерная окружность	Массив точек	Массив точек	Массив точек

	Трехмерная прямая	Трехмерный сплайн	Плоскость
Трехмерная прямая			Массив точек
Трехмерный сплайн			Массив точек
Плоскость	Массив точек	Массив точек	Трех-ная прямая
Сфера	Массив точек	Массив точек	Трех-ная окружность
Цилиндр	Массив точек	Массив точек	Трех-ный сплайн
Конус	Массив точек	Массив точек	Трех-ный сплайн
Тор	Массив точек	Массив точек	Трех-ные сплайны
Сплайновая поверхность	Массив точек	Массив точек	Трех-ный сплайн

Для нахождения пересечений следует использовать специальный класс `Intersection`.

Две двухмерных прямые

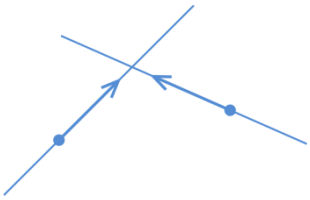


Рисунок 204. Пересечение двухмерных прямых

Формат	Intersection.LineLine(a, b);
Вход	Двухмерная линия, Двухмерная линия
Выход	Двухмерная точка

Пример	`Line2d a = new Line2d(Vector2d.Zero, Vector2d.XAxis);`
	`Line2d b = new Line2d(Vector2d.Zero, Vector2d.YAxis);`
	`Vector2d result = Intersection.LineLine(a, b);`

Операция приведет к ошибке выполнения программы, если имеет место параллельность элементов

Плоскость и трехмерная прямая

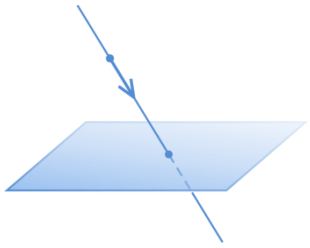


Рисунок 205. Пересечение плоскости и прямой

Формат	Intersection.LinePlane(a, b);
Вход	Трехмерная линия, Плоскость
Выход	Трехмерная точка

Пример	`Line3d a = new Line3d(Vector3d.Zero, Vector3d.YAxis);`
	`Plane b = new Plane(Vector3d.Zero, Vector3d.YAxis);`
	`Vector3d result = Intersection.LinePlane(a, b);`

Операция приведет к ошибке выполнения программы, если прямая лежит параллельной плоскости

Плоскость и плоскость

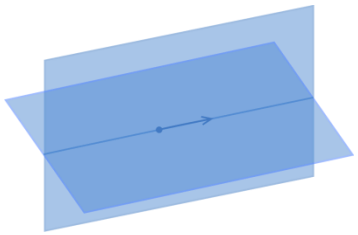


Рисунок 206. Пересечение двух плоскостей

Формат	Intersection.PlanePlane(a, b);
Вход	Плоскость, Плоскость
Выход	Трехмерная прямая

Пример	`Plane a = new Plane(Vector3d.Zero, Vector3d.XAxis);`
	`Plane b = new Plane(Vector3d.Zero, Vector3d.YAxis);`
	`Line result = Intersection.PlanePlane(a, b);`

Операция пресечения плоскости плоскостью к ошибке выполнения программы, если имеет место параллельность элементов

Плоскость и сфера

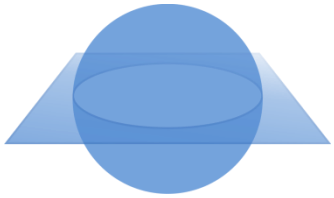


Рисунок 207. Пересечение сферы плоскостью

Формат	Intersection.PlaneSphere(a, b);
Вход	Плоскость, Сфера
Выход	Трехмерная окружность

Пример	`Plane a = new Plane(Vector3d.Zero, Vector3d.XAxis);`
	`Sphere b = new Sphere(Vector3d.Zero, 5);`
	`Circle result = Intersection.PlaneSphere(a, b);`

Операция пресечения сферы плоскостью приведет к ошибке выполнения программы, если центр сферы удален более чем на ее радиус от плоскости.

Плоскость и цилиндр (версия 5.0)

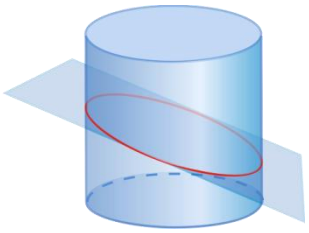


Рисунок 208. Пересечение цилиндра и плоскости

Формат	Intersection.PlaneCylinder(a, b);
Вход	Плоскость, Цилиндр
Выход	Сплайновая кривая

Пример	`Plane a = new Plane(Vector3d.Zero, Vector3d.XAxis);`
	`Cylinder b = new Cylinder(`
	`  Vector3d.Zero,`
	`  Vector3d.XAxis, 5.0);`
	`Spline3d result = Intersection.PlaneCylinder(a, b);`

Операция пресечения плоскости и цилиндра приведет к ошибке выполнения программы, если плоскость параллельна оси цилиндра.

Плоскость и конус (версия 5.0)

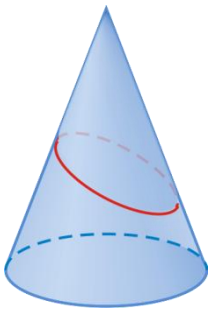


Рисунок 209. Пересечение плоскости и конуса

Формат	Intersection.PlaneCone(a, b);
Вход	Плоскость, Конус
Выход	Сплайновая кривая
Пример	`Plane a = new Plane(Vector3d.Zero, Vector3d.XAxis);` `Number angle = Basic.Pi / 8.0;`

```
Vector3d center = new Vector3d(1.0, 1.0, 1.0);
Vector3d direction = new Vector3d(1.0, 0.0, 0.0);
Cone b = new Cone(center, direction, angle);
Spline3d result = Intersection.PlaneCone(a, b);
```

Операция пресечения плоскости и конуса приведет к ошибке выполнения программы, если плоскость параллельна оси конуса.

Плоскость и тор (версия 5.0)

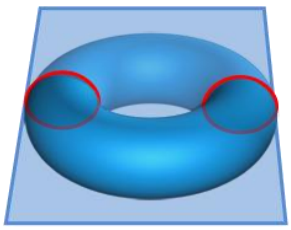


Рисунок 210. Пересечение плоскости и тора

Формат	Intersection.PlaneTorus(a, b);
Вход	Плоскость, Тор
Выход	Массив сплайновых кривых

Пример	`Plane a = new Plane(Vector3d.Zero, Vector3d.XAxis);`
	`Number bigRadius = 5.0;`
	`Number smallRadius = 1.0;`
	`Vector3d center = new Vector3d(1.0, 1.0, 1.0);`
	`Vector3d dir = new Vector3d(1.0, 0.0, 0.0);`
	`Torus a = new Torus(center, dir, bigRadius, smallRadius);`
	`Spline3d[] result = Intersection.PlaneTorus(a, b);`

Плоскость и сплайновая поверхность (версия 5.0)

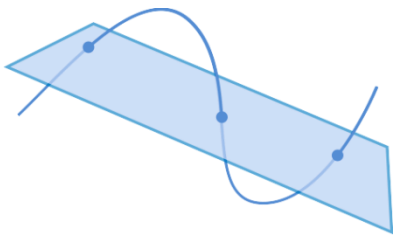


Рисунок 211. Пересечение плоскости и сплайна

Формат	Intersection.PlaneSurface(a, b);

Вход	Плоскость, Сплайновая поверхность
Выход	Массив сплайновых кривых

Пример	`Plane a = new Plane(Vector3d.Zero, Vector3d.XAxis);`
	`SplineSurface b = input;`
	`Spline3d[] result = Intersection.PlaneSurface(a, b);`

Сплайн и прямая в плоскости (версия 5.0)

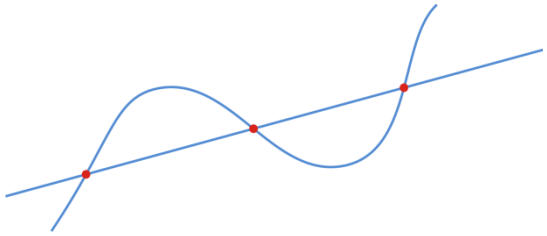


Рисунок 212. Пересечение сплайна и прямой

Формат	Intersection.LineSpline(a, b);
Вход	Двухмерная прямая, Двухмерный сплайн
Выход	Массив точек

Пример	`Line2d line = new Line2d(Vector2d.Zero, Vector2d.XAxis);`
	`Spline2d spline = new Spline2d(a, b, c, d, e);`
	`Vector2d[] result = Intersection.LineSpline(line, spline);`

Два двухмерных сплайна (версия 5.0)

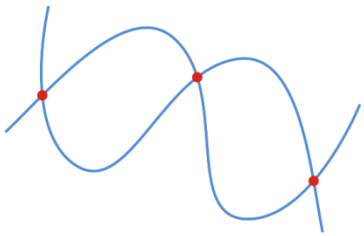


Рисунок 213. Пересечение двух двухмерных сплайнов

Формат	Intersection.SplineSpline(a, b);
Вход	Двухмерный сплайн, Двухмерный сплайн
Выход	Массив точек

Пример	`Spline2d a = new Spline2d(c, d, e);`
	`Spline2d b = new Spline2d(f, g, h);`
	`Vector2d[] result = Intersection.SplineSpline(a, b);`

Окружность и прямая в плоскости (версия 2.0)

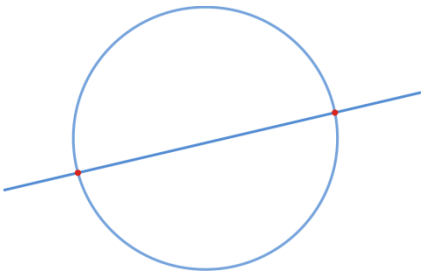


Рисунок 214. Пересечение окружности прямой

Формат	Intersection.LineCircle(a, b);
Вход	Двухмерная прямая, Двухмерная окружность
Выход	Массив точек (0..2)

Пример	`Line2d line = new Line2d(Vector2d.Zero, Vector2d.XAxis);`
	`Circle2d circle = new Circle2d(Vector2d.Zero, 5);`
	`Vector2d[] result = Intersection.LineCircle(line, circle);`

Окружность и сплайн в плоскости (версия 5.0)

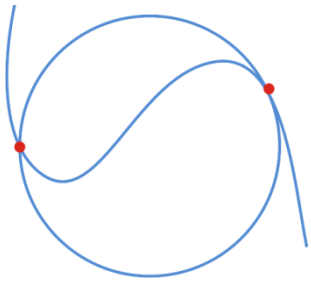


Рисунок 215. Пересечение окружности и сплайна

Формат	Intersection.SplineCircle(a, b);
Вход	Двухмерный сплайн, Двухмерная окружность
Выход	Массив точек

Пример	`Spline2d spline = new Spline2d(a, b, c, d, e);`
	`Circle2d circle = new Circle2d(Vector2d.Zero, 5);`
	`Vector2d[] result=Intersection.SplineCircle(spline,circle);`

Две окружности в плоскости (версия 2.0)

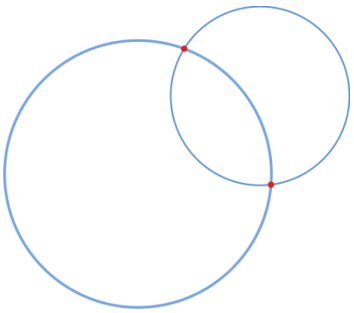


Рисунок 216. Пересечение двух окружностей

Формат	Intersection.CircleCircle(a, b);
Вход	Двухмерная окружность, Двухмерная окружность
Выход	Массив точек (0..2)

Пример	`Circle2d a = new Circle2d(Vector2d.XAxis, 5);`
	`Circle2d b = new Circle2d(Vector2d.Zero, 5);`
	`Vector2d[] result = Intersection.CircleCircle(a, b);`

Цилиндр и прямая (версия 4.0)

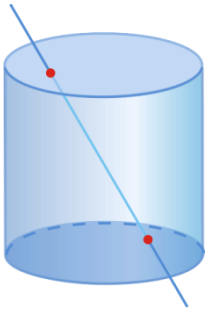


Рисунок 217. Пересечение цилиндра и прямой

Формат	Intersection.LineCylinder(a, b);
Вход	Трехмерная прямая, Цилиндр
Выход	Массив точек (2 элемента)

Пример	`Line3d a = new Line3d(Vector3d.Zero, Vector3d.XAxis);`
	`Cylinder b=new Cylinder(Vector3d.Zero,Vector3d.YAxis,5);`
	`Vector3d[] result = Intersection.LineCylinder(a, b);`

Конус и прямая (версия 4.0)

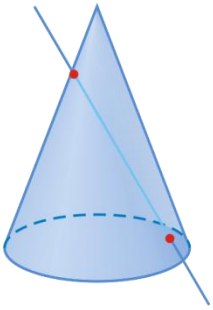


Рисунок 218. Пересечение конуса и прямой

Формат	Intersection.LineCone(a, b);
Вход	Трехмерная прямая, Конус
Выход	Массив точек (2 элемента)

Пример	`Line3d a = new Line3d(Vector3d.Zero, Vector3d.XAxis);`
	`Cone b = new Cone(1.0, 1.0, 1.0,`
	`0.0, 0.0, 1.0,`
	`Basic.Pi / 8.0);`
	`Vector3d[] result = Intersection.LineCone(a, b);`

Тор и прямая (версия 4.0)

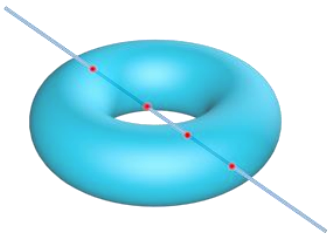


Рисунок 219. Пересечение тора и прямой

Формат	Intersection.LineTorus(a, b);
Вход	Трехмерная прямая, Тор

Выход	Массив точек (2 или 4 элемента)

Пример	`Line3d a = new Line3d(Vector3d.Zero, Vector3d.XAxis);`
	`Torus b = new Torus(new Vector3d(1.0, 1.0, 1.0),`
	`new Vector3d(0.0, 0.0, 1.0),`
	`8.0, 1.0);`
	`Vector3d[] result = Intersection.LineTorus(a, b);`

Сфера и прямая (версия 4.0)

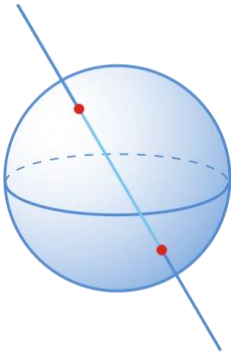


Рисунок 220. Пересечение сферы и прямой

Формат	Intersection.LineSphere(a, b);
Вход	Трехмерная прямая, Сфера
Выход	Массив точек (0..2 элементов)

Пример	`Line3d a = new Line3d(Vector3d.Zero, Vector3d.XAxis);`
	`Sphere b = new Sphere(new Vector3d(1.0, 1.0, 1.0), 8.0);`
	`Vector3d[] result = Intersection.LineSphere(a, b);`

Сплайновая поверхность и прямая (версия 4.0)

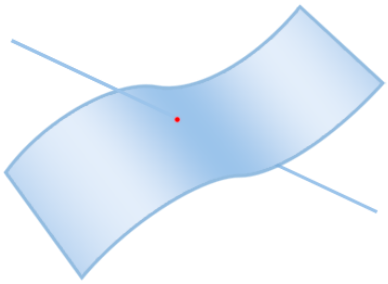


Рисунок 221. Пересечение сплайновой поверхности и прямой

Формат	Intersection.LineSurface(a, b);
Вход	Трехмерная прямая,

	Поверхность
Выход	Массив точек (0..N элементов)

Пример	`Line3d a = new Line3d(Vector3d.Zero, Vector3d.XAxis);`
	`SplineSurface b = <input>;`
	`Vector3d[] result = Intersection.LineSurface(a, b);`

Сплайновая кривая и плоскость (версия 5.0)

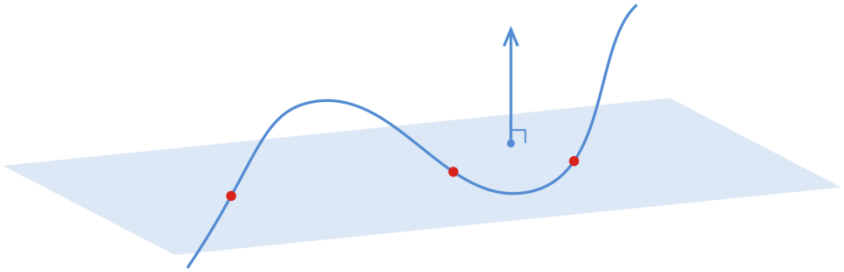


Рисунок 222. Пересечение сплайна с плоскостью

Формат	Intersection.SplinePlane(a, b);
Вход	Трехмерный сплайн, Плоскость
Выход	Массив точек

Пример	`Plane plane = new Plane(Vector3d.Zero, Vector3d.XAxis);`
	`Spline3d spline = <input>;`
	`Vector3d[] result = Intersection.SplinePlane(spline, plane);`

Сплайновая кривая и цилиндр (версия 5.0)

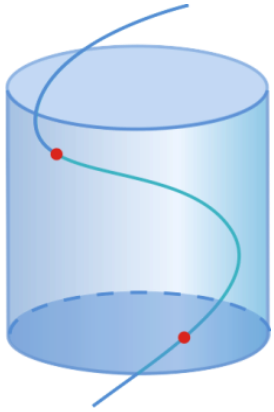


Рисунок 223. Пересечение цилиндра и сплайновой кривой

Формат	Intersection.SplineCylinder(a, b);
Вход	Трехмерный сплайн, Цилиндр
Выход	Массив точек

Пример	`Cylinder cylinder = new Cylinder(`
	`  Vector3d.Zero,`
	`  Vector3d.XAxis, 5.0);`
	`Spline3d spline = <input>;`
	`Vector3d[] result = Intersection.SplineCylinder(`
	`  spline, cylinder);`

Сплайновая кривая и конус (версия 5.0)

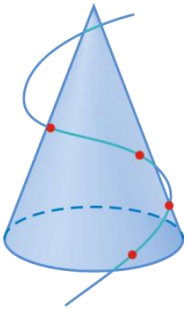


Рисунок 224. Пересечение конуса и сплайновой кривой

Формат	Intersection.SplineCone(a, b);
Вход	Трехмерный сплайн, Конус
Выход	Массив точек

Пример	`Cone cone = new Cone(`
	`  1.0, 1.0, 1.0,`
	`  0.0, 0.0, 1.0, Basic.Pi / 8.0);`
	`Spline3d spline = <input>;`
	`Vector3d[] result = Intersection.SplineCone(`
	`  spline,`
	`  cone);`

Сплайновая кривая и тор (версия 5.0)

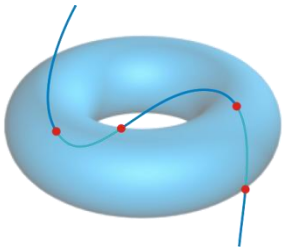


Рисунок 225. Пересечение сплайновой кривой и тора

Формат	Intersection.SplineTorus(a, b);
Вход	Трехмерный сплайн, Тор
Выход	Массив точек

Пример	`Torus torus = new Torus(Vector3d.Zero,Vector3d.XAxis, 5, 1);`
	`Spline3d spline = <input>;`
	`Vector3d[] result = Intersection.SplineTorus(spline, torus);`

Сплайновая кривая и сфера (версия 5.0)

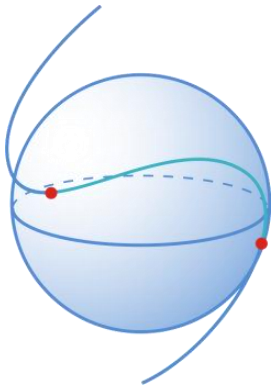


Рисунок 226. Пересечение сферы и сплайновой кривой

Формат	Intersection.SplineSphere(a, b);
Вход	Трехмерный сплайн, Сфера
Выход	Массив точек

Пример	`Sphere sphere = new Sphere(Vector3d.Zero, 5.0);`
	`Spline3d spline = <input>;`
	`Vector3d[] result=Intersection.SphereSpline(spline, sphere);`

Сплайновые кривая и поверхность (версия 5.0)

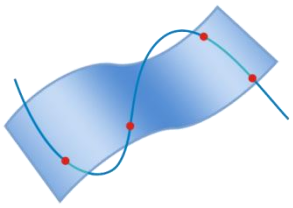


Рисунок 227. Пересечение сплайновой поверхности и сплайновой кривой

Формат	Intersection.SplineSurface(a, b);
Вход	Трехмерный сплайн, Сплайновая поверхность
Выход	Массив точек

Пример	`SplineSurface surface = input;`
	`Spline3d spline = <input>;`
	`Vector3d[] result = Intersection.SurfaceSpline(`
	`spline, surface);`

Расчет расстояний

Операция предполагает нахождение расстояния между двумя элементами геометрии. Ниже приведена таблица допустимых операций:

	Двухмерный вектор	Двухмерная прямая	Двухмерная окружность	Двухмерный сплайн	Трехмерный вектор	Трехмерная прямая	Трехмерный сплайн	Трехмерная окружность	Плоскость	Цилиндр	Сфера	Конус	Сплайновая пов-сть
Двухмерный вектор	✔	✔	✔	✔									
Трехмерный вектор					✔	✔	✔	✔	✔	✔	✔	✔	✔

Два двухмерных вектора

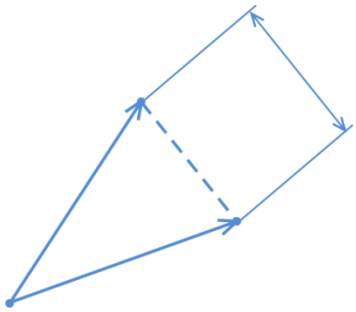


Рисунок 228. Расстояние от точки до точки

Формат	Distance.PointPoint(a, b);
Вход	Двухмерный вектор, Двухмерный вектор
Выход	Расстояние от одной точки до другой

Пример	`Vector2d a = new Vector2d(1.0, 1.0);`
	`Vector2d b = new Vector2d(2.0, 2.0);`
	`Number distance = Distance.PointPoint(a, b);`

Двухмерные точка и прямая

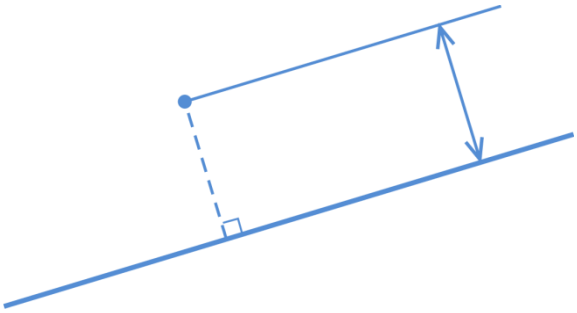


Рисунок 229. Расстояние от точки до прямой, двухмерный случай

Формат	Distance.PointLine(a, b);
Вход	Двухмерный вектор, Двухмерная прямая
Выход	Расстояние от точки до прямой

Пример	`Vector2d a = new Vector2d(1.0, 1.0);`
	`Line2d b = new Line2d(Vector2d.Zero, Vector2d.XAxis);`
	`Number distance = Distance.PointLine(a, b);`

Двухмерные точка и окружность

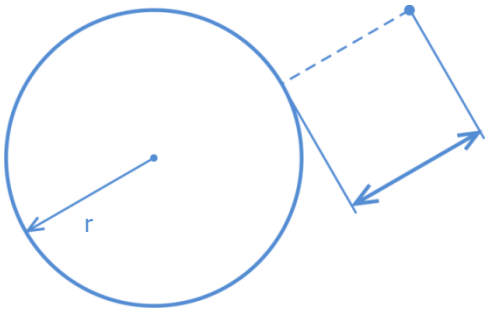


Рисунок 230. Расстояние от точки до окружности, двухмерный случай

Формат	Distance.PointCircle(a, b);
Вход	Двухмерный вектор, Двухмерная окружность
Выход	Расстояние от точки до окружности

Пример	`Vector2d a = new Vector2d(5.0, 5.0);`
	`Circle2d b = new Circle2d(Vector2d.Zero, 1.0);`
	`Number distance = Distance.PointPoint(a, b);`

Два трехмерных вектора

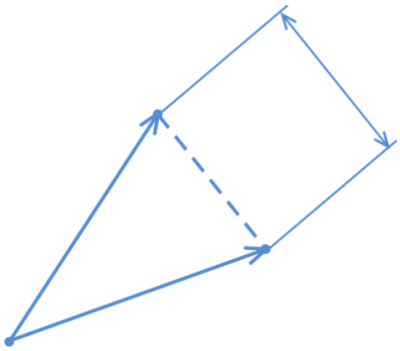


Рисунок 231. Расстояние между двумя трехмерными точками

Формат	Distance.PointPoint(a, b);
Вход	Трехмерный вектор, Трехмерный вектор
Выход	Расстояние от одной точки до другой

Пример	`Vector3d a = new Vector3d(1.0, 1.0, 1.0);`
	`Vector3d b = new Vector3d(2.0, 2.0, 2.0);`
	`Number distance = Distance.PointPoint(a, b);`

Трехмерный вектор и прямая

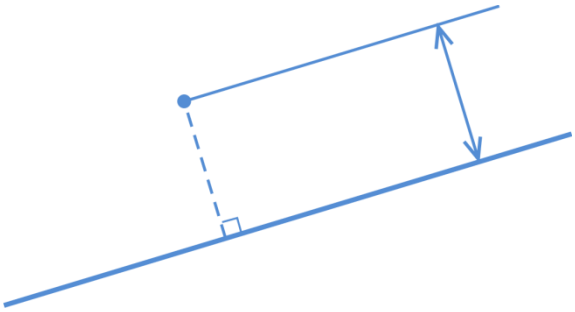


Рисунок 232. Расстояние от точки до прямой, трехмерный случай

Формат	Distance.PointLine(a, b);
Вход	Трехмерный вектор, Трехмерная прямая
Выход	Расстояние от точки до прямой

Пример	`Vector3d a = new Vector3d(1.0, 1.0, 1.0);`
	`Line3d b = new Line3d(Vector3d.Zero, Vector3d.XAxis);`
	`Number distance = Distance.PointLine(a, b);`

Трехмерный вектор и плоскость

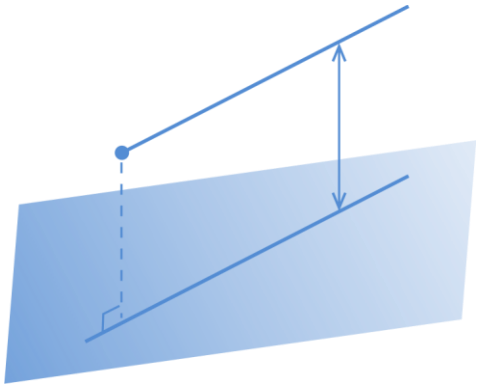


Рисунок 233. Расстояние от точки до плоскости

Формат	Distance.PointPlane(a, b);
Вход	Трехмерный вектор, Плоскость
Выход	Расстояние от точки до плоскости

Пример	`Vector3d a = new Vector3d(1.0, 1.0, 1.0);`
	`Plane b = new Plane(Vector3d.Zero, Vector3d.XAxis);`
	`Number distance = Distance.PointPlane(a, b);`

Трехмерный вектор и окружность

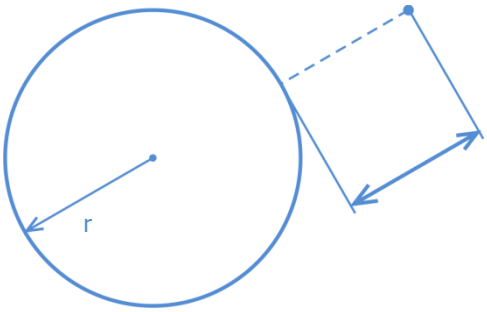


Рисунок 234. Расстояние от точки до окружности, трехмерный случай

Формат	Distance.PointCircle(a, b);
Вход	Трехмерный вектор, Окружность
Выход	Расстояние от точки до окружности

Пример	`Vector3d a = new Vector3d(5.0, 5.0, 5.0);`
	`Circle3d b = new Circle3d(Vector3d.Zero, 1.0);`
	`Number distance = Distance.PointCircle(a, b);`

Трехмерный вектор и сфера

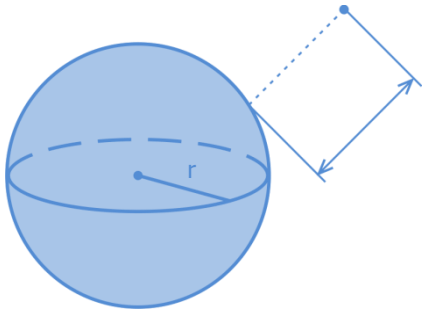


Рисунок 235. Расстояние от точки до сферы

Формат	Distance.PointShpere(a, b);
Вход	Трехмерный вектор, Сфера
Выход	Расстояние от точки до сферы

Пример	`Vector3d a = new Vector3d(5.0, 5.0, 5.0);`
	`Sphere b = new Sphere(Vector3d.Zero, 1.0);`
	`Number distance = Distance.PointSphere(a, b);`

Трехмерный вектор и цилиндр

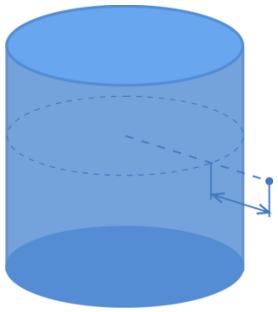


Рисунок 236. Расстояние от точки до цилиндра

Формат	Distance.PointCylinder(a, b);
Вход	Трехмерный вектор, Цилиндр
Выход	Расстояние от точки до цилиндра
Пример	`Vector3d a = new Vector3d(5.0, 5.0, 5.0);` `Cylinder b = new Cylinder(Vector3d.Zero, Vector3d.XAxis, 2.0); Number distance = Distance.PointCylinder(a, b);`

Трехмерный вектор и тор (версия 4.0)

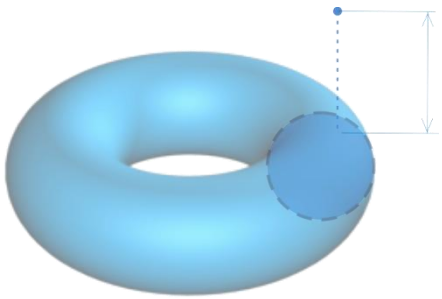


Рисунок 237. Расстояние от точки до тора

Формат	Distance.PointTorus(a, b);
Вход	Трехмерный вектор, Тор
Выход	Расстояние от точки до тора

Пример	`Vector3d a = new Vector3d(5.0, 5.0, 5.0);`
	`Torus b = new Torus(Vector3d.Zero, Vector3d.XAxis,5,1);`
	`Number distance = Distance.PointTorus(a, b);`

Трехмерный вектор и конус

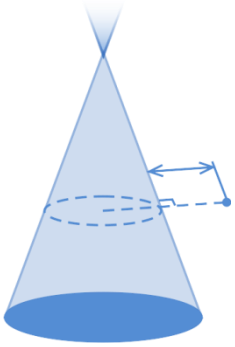


Рисунок 238. Расстояние от точки до конуса

Формат	Distance.PointCone(a, b);
Вход	Трехмерный вектор, Конус
Выход	Расстояние от точки до конуса

Пример	`Vector3d a = new Vector3d(5.0, 5.0, 5.0);`
	`Cone b = new Cone(Vector3d.Zero, Vector3d.XAxis, 3.0);`
	`Number distance = Distance.PointCone(a, b);`

Двухмерные точка и сплайновая кривая (версия 2.0)

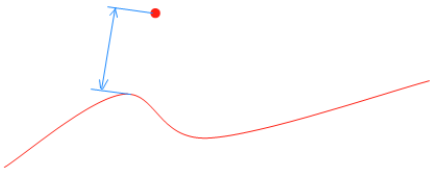


Рисунок 239. Расстояние от точки до сплайна в плоскости

Формат	Distance.PointSpline(a, b);
Вход	Двухмерный вектор, Двухмерный сплайн
Выход	Расстояние от точки до сплайна

Пример	`Vector2d point = new Vector2d(5.0, 5.0);`
	`Spline2d spline = new Spline2d(a, b, c, d, e);`
	`Number distance = Distance.PointSpline(point, spline);`

Трехмерные точка и сплайновая кривая (версия 2.0)

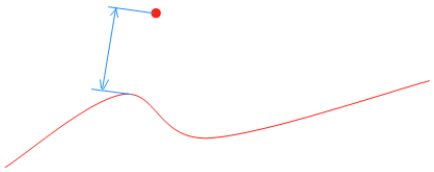


Рисунок 240. Расстояние от точки до сплайна в пространстве

Формат	Distance.PointSpline(a, b);
Вход	Трехмерный вектор, Трехмерный сплайн
Выход	Расстояние от точки до сплайна

Пример	`Vector3d point = new Vector3d(5.0, 5.0, 5.0);`
	`Spline3d spline = new Spline3d(a, b, c, d, e);`
	`Number distance = Distance.PointSpline(point, spline);`

Трехмерные точка и сплайновая поверхность (версия 5.0)

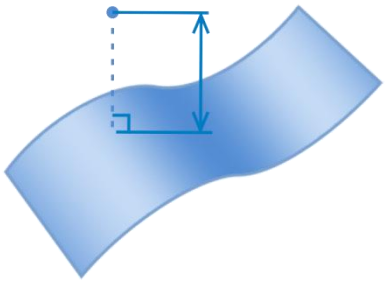


Рисунок 241. Расстояние от точки до сплайновой поверхности

Формат	Distance.PointSurface(a, b);
Вход	Трехмерный вектор, Сплайновая поверхность
Выход	Расстояние от точки до сплайновой поверхности

Пример	`Vector3d point = new Vector3d(5.0, 5.0, 5.0);`
	`SplineSurface spline = <input>;`
	`Number distance = Distance.PointSurface(point, spline);`

Расчет углов

Операция предполагает нахождение углов между двумя элементами геометрии. Ниже приведена таблица допустимых операций:

	Двухмерный вектор	Двухмерная прямая	Двухмерная окружность	Трехмерный вектор	Трехмерная прямая	Трехмерная окружность	Плоскость	Цилиндр	Сфера	Конус
Двухмерный вектор										
Двухмерная прямая		✔								
Двухмерная окружность										
Трехмерный вектор										
Трехмерная прямая					✔		✔			
Трехмерная окружность										
Плоскость					✔		✔			
Цилиндр										
Сфера										
Конус										

Две двухмерные прямые

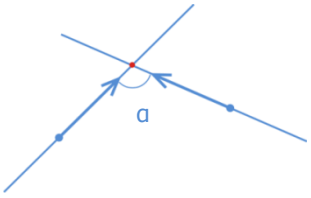


Рисунок 242. Угол между двумя двухмерными прямыми

Формат	Angle.LineLine(a, b);

Вход	Двухмерная прямая, Двухмерная прямая
Выход	Угол в радианах

Пример	`Line2d a = new Line2d(Vector2d.Zero, Vector2d.XAxis);`
	`Line2d b = new Line2d(Vector2d.Zero, Vector2d.YAxis);`
	`Number angle = Angle.LineLine(a, b);`

Две трехмерные прямые

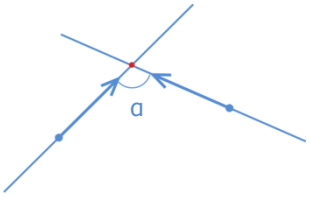


Рисунок 243. Угол между двумя трехмерными прямыми

Формат	Angle.LineLine(a, b);
Вход	Трехмерная прямая, Трехмерная прямая
Выход	Угол в радианах

Пример	`Line3d a = new Line3d(Vector3d.Zero, Vector3d.XAxis);`
	`Line3d b = new Line3d(Vector3d.Zero, Vector3d.YAxis);`
	`Number angle = Angle.LineLine(a, b);`

Трехмерная прямая и плоскость

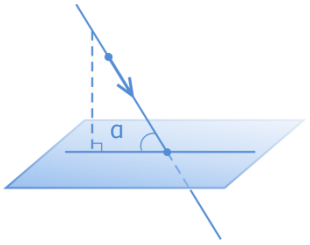


Рисунок 244. Угол между плоскостью и прямой

Формат	Angle.LinePlane(a, b);
Вход	Трехмерная прямая, Плоскость
Выход	Угол в радианах

Пример	Line3d a = new Line3d(Vector3d.Zero, Vector3d.XAxis); Plane b = new Plane(Vector3d.Zero, Vector3d.XAxis); Number angle = Angle.LineLine(a, b);

Две плоскости

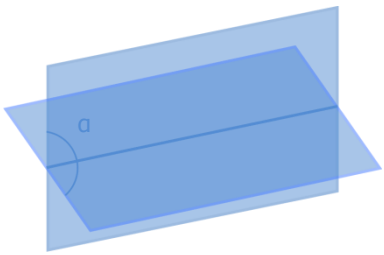


Рисунок 245. Угол между двумя плоскостями

Формат	Angle.PlanePlane(a, b);
Вход	Плоскость, Плоскость
Выход	Угол в радианах
Пример	Plane a = new Plane(Vector3d.Zero, Vector3d.YAxis); Plane b = new Plane(Vector3d.Zero, Vector3d.XAxis); Number angle = Angle.PlanePlane(a, b);

Построение проекций элементов

Операция предполагает нахождение проекции элемента на другой элемент. Ниже приведена таблица допустимых операций:

	Двухмерная прямая	Двухмерная окружность	Двухмерный сплайн	Трехмерный вектор	Трехмерная прямая	Трехмерный слайн	Трехмерная окружность	Плоскость	Цилиндр	Сфера	Тор	Конус	Сплайновая пов-сть
Двухмерный вектор	✔	✔	✔										
Трехмерный вектор					✔	✔	✔	✔	✔	✔	✔	✔	✔
Трехмерная прямая								✔					

Двухмерная точка на прямую

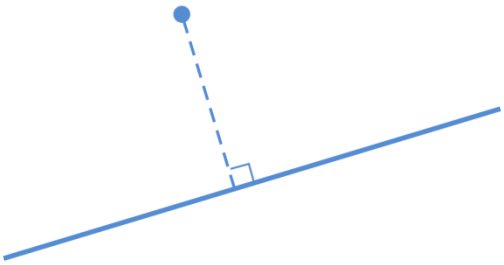


Рисунок 246. Проекция точки на двухмерную прямую

Формат	Projection.PointLine(a, b);
Вход	Двухмерный вектор, Двухмерная прямая
Выход	Двухмерный вектор – проекция на прямую

Пример	`Vector2d point = new Vector2d(1.0, 1.0);`
	`Line2d line = new Line2d(Vector2d.Zero, Vector2d.XAxis);`
	`Vector2d result = Projection.PointLine(point, line);`

Двухмерная точка на окружность (версия 5.0)

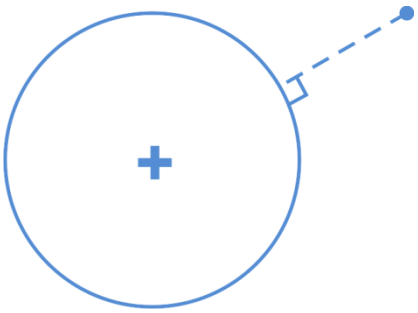


Рисунок 247. Проекция точки на окружность

Формат	Projection.PointCircle(a, b);
Вход	Двухмерный вектор, Двухмерная окружность
Выход	Двухмерный вектор – проекция на окружность

Пример	`Vector2d point = new Vector2d(1.0, 1.0);`
	`Circle2d circle = new Circle2d(Vector2d.Zero, 0.5);`
	`Vector2d result = Projection.PointCircle(point, circle);`

Двухмерная точка на сплайновую кривую (версия 5.0)

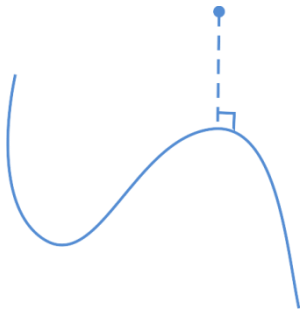


Рисунок 248. Проекция точки на сплайновую кривую

Формат	Projection.PointSpline(a, b);
Вход	Двухмерный вектор, Двухмерная сплайновая кривая
Выход	Двухмерный вектор – проекция на сплайн

Пример	`Vector2d point = new Vector2d(1.0, 1.0);`
	`Spline2d spline = new Spline2d(a, b, c, d);`
	`Vector2d result = Projection.PointSpline(point, spline);`

Трехмерная точка на прямую

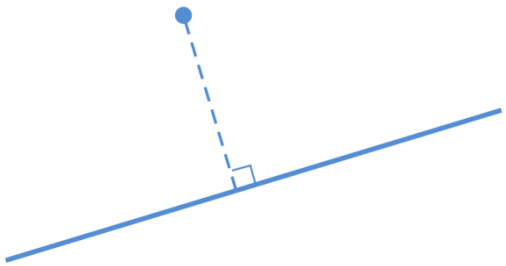


Рисунок 249. Проекция точки на трехмерную прямую

Формат	Projection.PointLine(a, b);
Вход	Трехмерный вектор, Трехмерная прямая
Выход	Трехмерный вектор – проекция на прямую

Пример	`Vector3d point = new Vector3d(1.0, 1.0, 1.0);`
	`Line3d line = new Line3d(Vector3d.Zero, Vector3d.XAxis);`
	`Vector3d result = Projection.PointLine(point, line);`

Трехмерная точка на плоскость

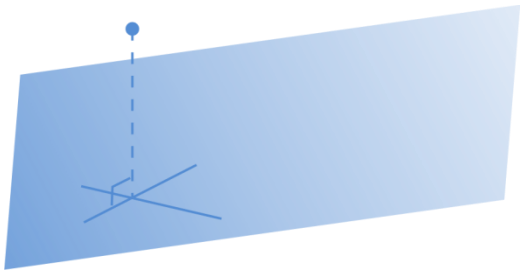


Рисунок 250. Проекция точки на плоскость

Формат	Projection.PointPlane(a, b);
Вход	Трехмерный вектор, Плоскость
Выход	Трехмерный вектор – проекция на плоскость

Пример	`Vector3d point = new Vector3d(1.0, 1.0, 1.0);`
	`Plane plane = new Plane(Vector3d.Zero, Vector3d.XAxis);`
	`Vector3d result = Projection.PointPlane(point, plane);`

Трехмерная точка на сферу (версия 4.0)

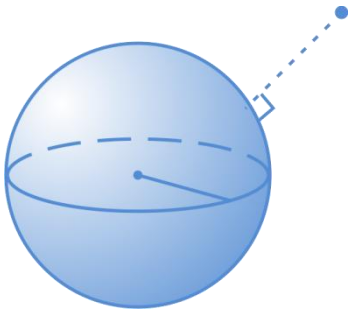


Рисунок 251. Проекция точки на сферу

Формат	Projection.PointSphere(a, b);
Вход	Трехмерный вектор, Сфера
Выход	Трехмерный вектор

Пример	`Vector3d point = new Vector3d(1.0, 1.0, 1.0);`
	`Sphere sphere = new Sphere(Vector3d.Zero, 5.0);`
	`Vector3d result = Projection.PointSphere(point, sphere);`

Трехмерная точка на цилиндр (версия 4.0)

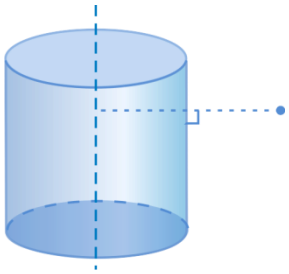


Рисунок 252. Проекция точки на цилиндр

Формат	Projection.PointCylinder(a, b);
Вход	Трехмерный вектор, Цилиндр
Выход	Трехмерный вектор

Пример	`Vector3d point = new Vector3d(1.0, 1.0, 1.0);`
	`Cylinder cylinder = new Cylinder(`
	`  Vector3d.Zero,`
	`  Vector3d.ZAxis, 5.0);`
	`Vector3d result = Projection.PointCylinder (`
	`  point, cylinder);`

Трехмерная точка на конус (версия 4.0)

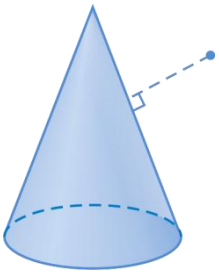


Рисунок 253. Проекция точки на конус

Формат	Projection.PointCone(a, b);
Вход	Трехмерный вектор, Конус
Выход	Трехмерный вектор

Пример	`Vector3d point = new Vector3d(10.0, 10.0, 10.0);`
	`Cone cone = new Cone(`
	`  1.0, 1.0, 1.0,`
	`  0.0, 0.0, 1.0,`
	`  Basic.Pi / 8.0);`
	`Vector3d result = Projection.PointCone (`
	`  point, cone);`

Трехмерная точка на тор (версия 4.0)

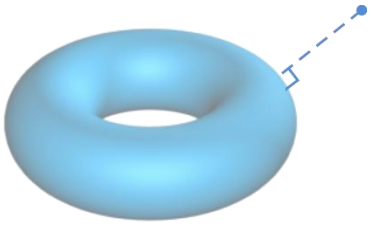


Рисунок 254. Проекция трехмерной точки на тор

Формат	Projection.PointTorus(a, b);
Вход	Трехмерный вектор, Тор
Выход	Трехмерный вектор

Пример	`Vector3d point = new Vector3d(100.0, 100.0, 100.0);`
	`Number bigRadius = 5.0;`
	`Number smallRadius = 1.0;`
	`Vector3d center = new Vector3d(1.0, 1.0, 1.0);`
	`Vector3d direction = new Vector3d(1.0, 0.0, 0.0);`
	`Torus torus = new Torus(center, direction,`
	`bigRadius, smallRadius);`
	`Vector3d result = Projection.PointTorus(point, torus);`

Трехмерная точка на окружность (версия 5.0)

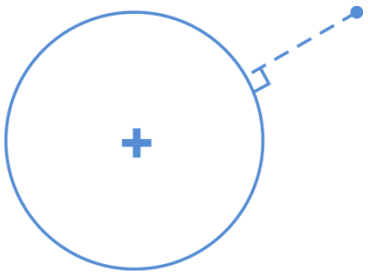


Рисунок 255. Проекция трехмерной точки на окружность

Формат	Projection.PointCircle(a, b);
Вход	Трехмерный вектор, Трехмерная окружность
Выход	Трехмерный вектор

Пример	`Vector3d point = new Vector3d(1.0, 1.0, 1.0);`
	`Circle3d circle = new Circle3d(`
	`Vector3d.Zero,`
	`Vector3d.XAxis, 5.0);`
	`Vector3d result = Projection.PointCircle(point, circle);`

Трехмерная точка на сплайновую кривую (версия 5.0)

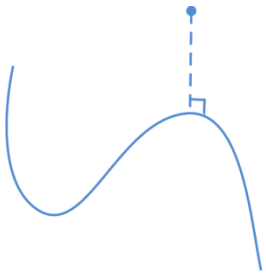


Рисунок 256. Проекция трехмерной точки на сплайновую кривую

Формат	Projection.PointSpline(a, b);
Вход	Трехмерный вектор, Трехмерная сплайновая кривая
Выход	Трехмерный вектор

Пример	`Vector3d point = new Vector3d(1.0, 1.0, 1.0);`
	`Spline3d spline = new Spline3d(a, b, c, d);`
	`Vector3d result = Projection.PointSpline(point, spline);`

Трехмерная точка на сплайновую поверхность (версия 5.0)

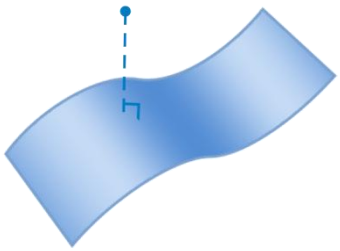


Рисунок 257. Проекция трехмерной точки на сплайновую поверхность

Формат	Projection.PointSurface(a, b);
Вход	Трехмерный вектор, Сплайновая поверхность
Выход	Трехмерный вектор

Пример	`Vector3d point = new Vector3d(1.0, 1.0, 1.0);`
	`SplineSurface surface = <input>;`
	`Vector3d result=Projection.PointSurface(point,surface);`

Построение симметричных элементов (версия 2.0)

Данный класс построений позволяет создавать элемент, симметричный другому элементу относительно элемента симметрии.

Построение симметричной двухмерной прямой (версия 2.0)

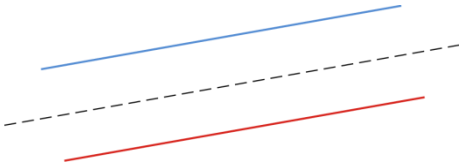


Рисунок 258. Симметричная двухмерная прямая

Назначение	Производит построение симметричной данной прямую относительно оси симметрии.

Пример	`Line2d line = Symmetry.Line(a, b);`
	`a – двухмерная прямая, симметричная которой требуется`
	`b – двухмерная прямая, ось симметрии`

Построение симметричной двухмерной окружности (версия 2.0)

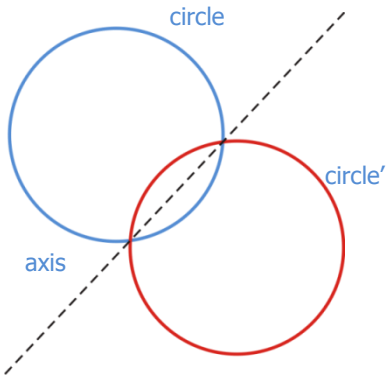


Рисунок 259. Симметричная двухмерная окружность

Назначение	Производит построение симметричной данной окружность относительно оси симметрии.

Пример	`Circle2d circle = Symmetry.Circle(a, b);`
	`a – двухмерная окружность, симметричная которой требуется`
	`b – двухмерная прямая, ось симметрии`

Построение симметричной трехмерной прямой (версия 2.0)

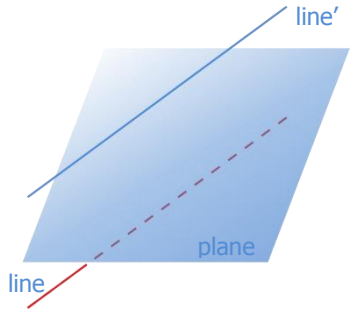


Рисунок 260. Симметричная трехмерная прямая

Назначение	Производит построение симметричной данной прямую относительно плоскости симметрии.

Пример	`Line3d line = Symmetry.Line(a, b);`
	`a – трехмерная прямая, симметричная которой требуется`
	`b – плоскость симметрии`

Построение симметричной плоскости (версия 2.0)

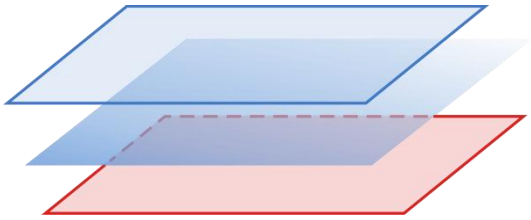


Рисунок 261. Симметричная плоскость

Назначение	Производит построение симметричной данной плоскость относительно плоскости симметрии.

Пример	`Plane plane = Symmetry.Plane(a, b);`
	`a – плоскость, симметричная которой требуется`
	`b – плоскость симметрии`

Построение симметричной трехмерной окружности (версия 2.0)

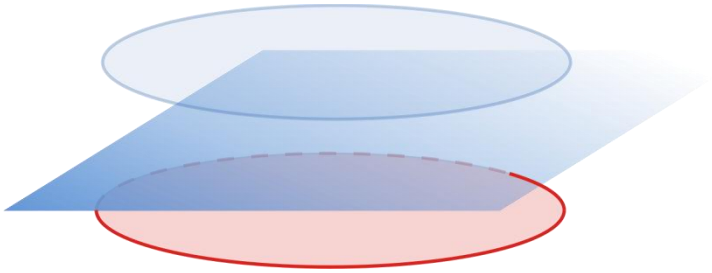


Рисунок 262. Симметричная трехмерная окружность

Назначение	Производит построение симметричной данной окружность относительно плоскости симметрии.

Пример	`Circle3d plane = Symmetry.Circle(a, b);`
	`a – окружность, симметричная которой требуется`
	`b – плоскость симметрии`

Построение симметричной сферы (версия 2.0)

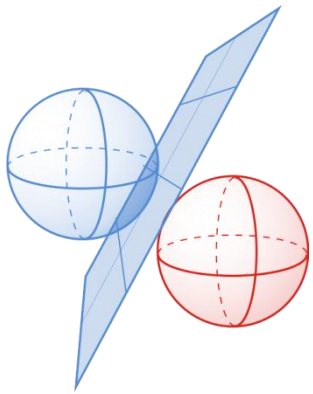


Рисунок 263. Симметричная сфера

Назначение	Производит построение симметричной данной сферу относительно плоскости симметрии.

Пример	`Sphere sphere = Symmetry.Sphere(a, b);`
	`a – сфера, симметричная которой требуется`
	`b – плоскость симметрии`

Построение симметричного конуса (версия 2.0)

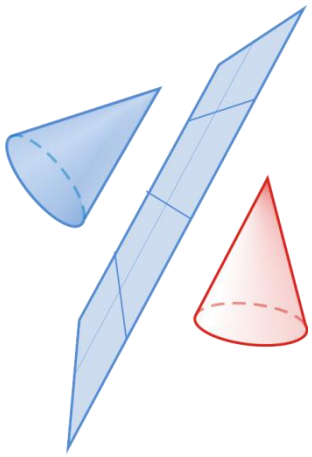


Рисунок 264. Симметричный конус

Назначение	Производит построение симметричного данному конуса относительно плоскости симметрии.

Пример	`Cone cone = Symmetry.Cone(a, b);`
	`a – конус, симметричный которому требуется`
	`b – плоскость симметрии`

Построение симметричного цилиндра (версия 2.0)

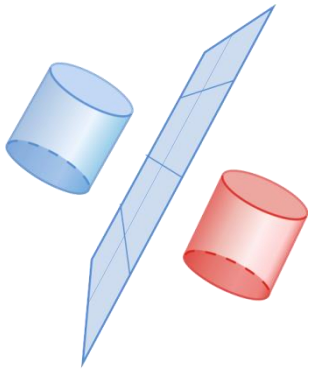


Рисунок 265. Симметричный цилиндр

Назначение	Производит построение симметричного данному цилиндра относительно плоскости симметрии.

Пример	`Cylinder cylinder = Symmetry.Cylinder(a, b);`
	`a – цилиндр, симметричный которому требуется`
	`b – плоскость симметрии`

Построение параллельных элементов (версия 2.0)

Построение параллельной двухмерной прямой (версия 2.0)

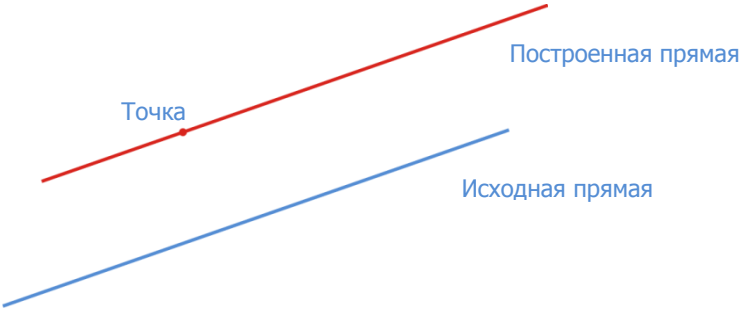


Рисунок 266. Параллельная прямая через точку

Назначение	Производит построение прямой, параллельной данной, проходящей через указанную точку, в плоскости.

Пример	`Line2d line = Parallel.Line(a, b);`
	`a – двухмерная прямая, параллельная которой требуется`
	`b – точка`

Построение параллельной трехмерной прямой (версия 2.0)

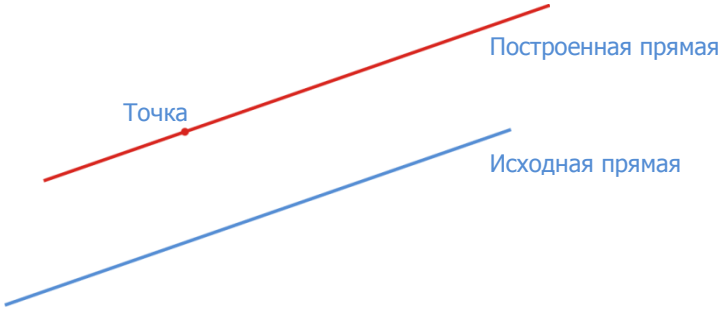


Рисунок 267. Трехмерная параллельная прямая через точку

Назначение	Производит построение прямой, параллельной данной, проходящей через указанную точку, в пространстве.

Пример	`Line3d line = Parallel.Line(a, b);`
	`a – трехмерная прямая, параллельная которой требуется`
	`b – точка`

Построение параллельной плоскости (версия 2.0)

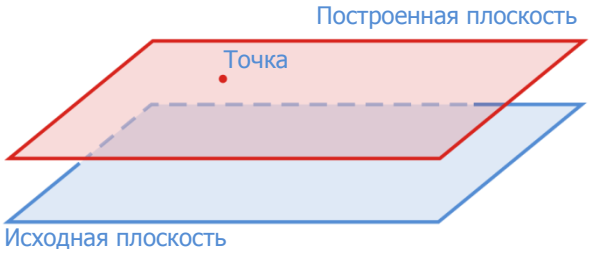


Рисунок 268. Параллельная плоскость через точку

Назначение	Производит построение плоскости, параллельной данной, проходящей через указанную точку.

Пример	`Plane plane = Parallel.Plane(a, b);`
	`a – плоскость, параллельная которой требуется`
	`b – точка`

Построение перпендикулярных элементов (версия 2.0)

Построение перпендикулярной двухмерной прямой (версия 2.0)

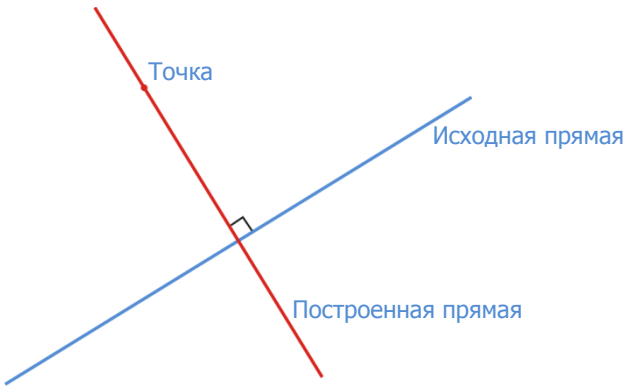


Рисунок 269. Параллельная прямая через точку

Назначение	Производит построение прямой, перпендикулярной данной, проходящей через указанную точку, в плоскости.

Пример	`Line2d line = Perpendicular.Line(a, b);`
	`a – двухмерная прямая, перпендикулярная которой требуется`
	`b – точка`

Построение перпендикулярной трехмерной прямой (версия 2.0)

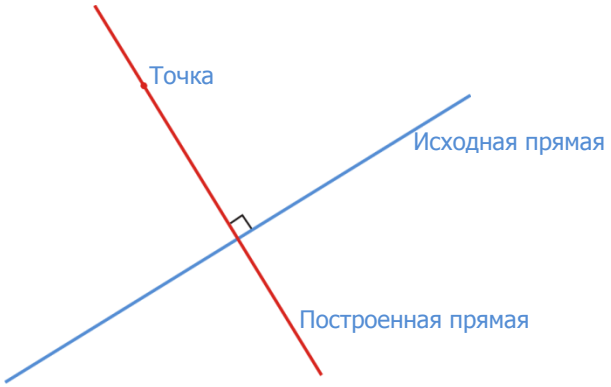


Рисунок 270. Перпендикулярная прямая через точку

Назначение	Производит построение прямой, перпендикулярной данной, проходящей через указанную точку, в пространстве.

Пример	`Line3d line = Perpendicular.Line(a, b);`
	`a – трехмерная прямая, перпендикулярная которой требуется`
	`b – точка`

Построение перпендикулярной плоскости (версия 2.0)

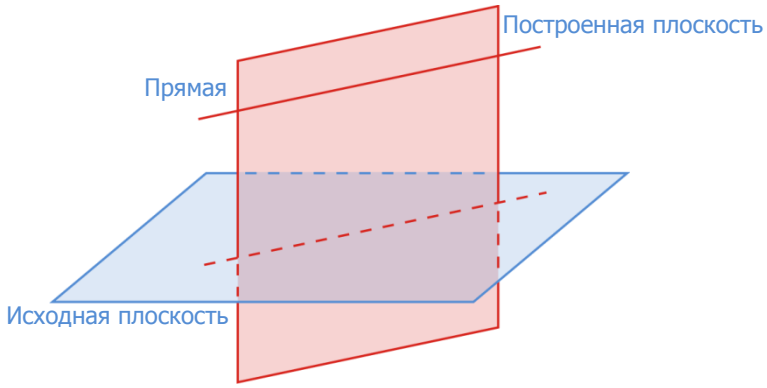


Рисунок 271. Перпендикулярная прямая через прямую

Назначение	Производит построение плоскости, перпендикулярной данной, проходящей через указанную прямую, в пространстве.

Пример	`Plane plane = Parallel.Plane(a, b);`
	`a – плоскость, параллельная которой требуется`
	`b – трехмерная прямая`

Построение наклонных элементов (версия 2.0)

Построение двухмерной прямой под углом (версия 2.0)

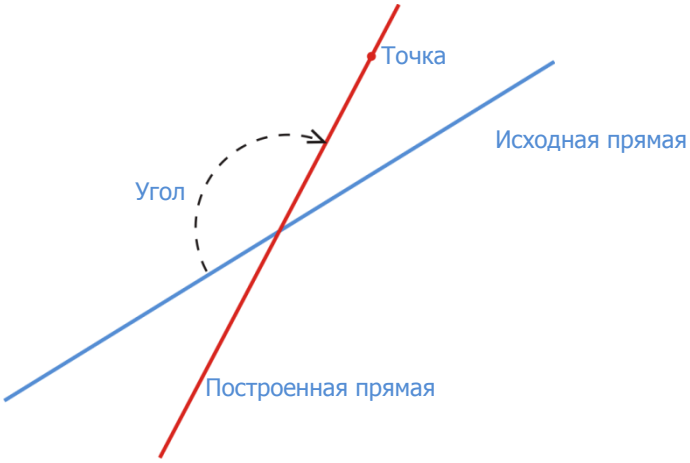


Рисунок 272. Параллельная прямая через точку

Назначение	Производит построение прямой под заданным углом к данной, проходящей через указанную точку, в плоскости. При этом угол откладывается по часовой стрелке.

Пример	`Line2d line = Slope.Line(a, b, alpha);`
	`a – двухмерная прямая, угол от которой отсчитывается`
	`alpha – угол по часовой стрелке, между прямыми`
	`b – точка`

Расчет отклонений

Отклонения формы

Наибольшее отклонение формы

Для расчета наибольшего отклонения формы следует применять класс FormDeviation. Первым аргументом передается заменяющий элемент, вторым передается набор измеренных точек с поверхности. Доступны следующие функции расчета наибольшего отклонения, в зависимости от геометрического элемента:

Пример:

```
Vector2d[] points = new Vector2d[] {
   new Vector2d(0.02, 0.04),
   new Vector2d(0.5, 0.067),
   new Vector2d(0.9, 0.8) };
Line2d line = new Line(Vector2d.Zero, new Vector2d(1, 1));
Number deviation = FormDeviation.Line(line, points);
```

Двухмерная прямая

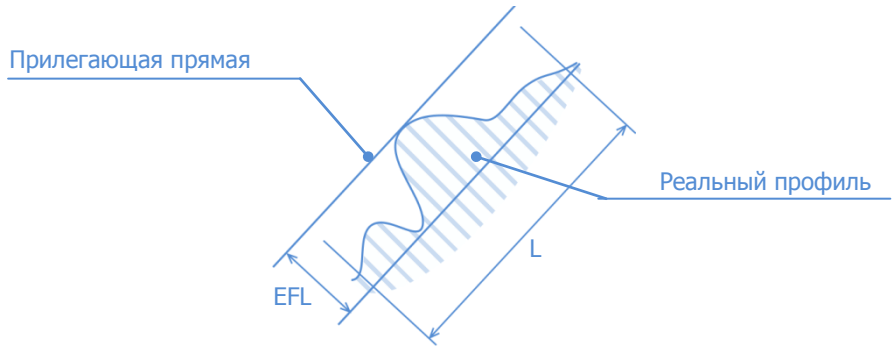


Рисунок 273. Отклонение от прямолинейности, двухмерный случай

Назначение	Находит наибольшее отклонение (EFL) прямой от прямолинейности на заданном отрезке (L) в плоскости.

Пример	`FormDeviation.Line(line, points);`
	`line – базовая двухмерная прямая, относительно которой рассчитывается отклонение;`
	`points – масив измеренных точек.`

Трехмерная прямая

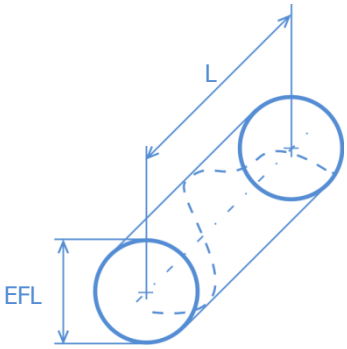


Рисунок 274. Отклонение от прямолинейности, трехмерный случай

Назначение	Находит наибольшее отклонение (EFL) прямой от прямолинейности на заданном отрезке (L) в пространстве.

Пример	`FormDeviation.Line(line, points);`
	`line – базовая трехмерная прямая, относительно которой рассчитывается отклонение;`
	`points – масив измеренных точек.`

Двухмерная окружность

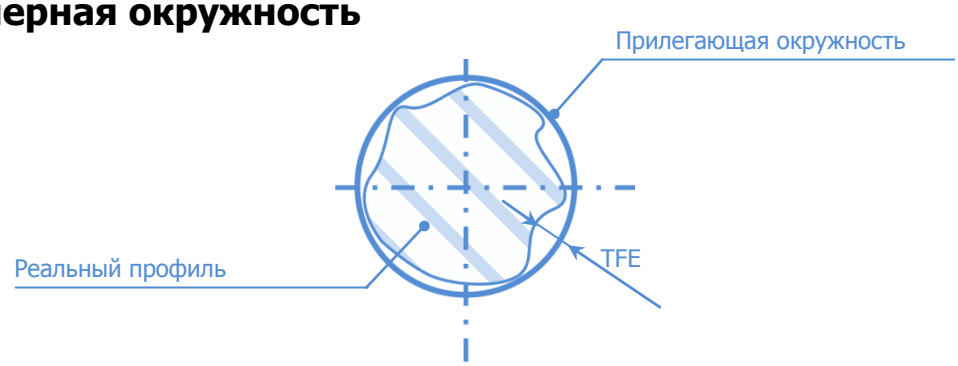


Рисунок 275. Отклонение от круглости

Назначение	Находит наибольшее отклонение (TFE) реального профиля от круглости в плоскости.

Пример	`FormDeviation.Circle(circle, points);`
	`circle – базовая двухмерная окружность, относительно которой рассчитывается отклонение;;`
	`points – масив измеренных точек.`

Трехмерная окружность

Назначение	Находит наибольшее отклонение (TFE) реального профиля от круглости в плоскости (см. **Рисунок 275**).

Пример	`FormDeviation.Circle(circle, points);`
	`circle – базовая двухмерная окружность, относительно которой рассчитывается отклонение;`
	`points – масив измеренных точек.`

Плоскость

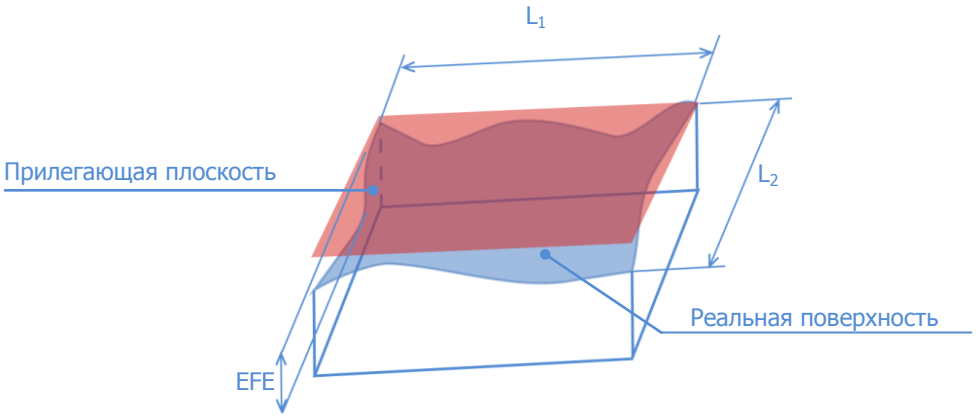


Рисунок 276. Отклонение от плоскостности

Назначение	Находит наибольшее отклонение (EFE) реального профиля от плоскостности в пространстве.

Пример	`FormDeviation.Plane(plane, points);`
	`plane – базовая плоскость, относительно которой рассчитыватся отклонение;`
	`points – масив измеренных точек.`

Сфера

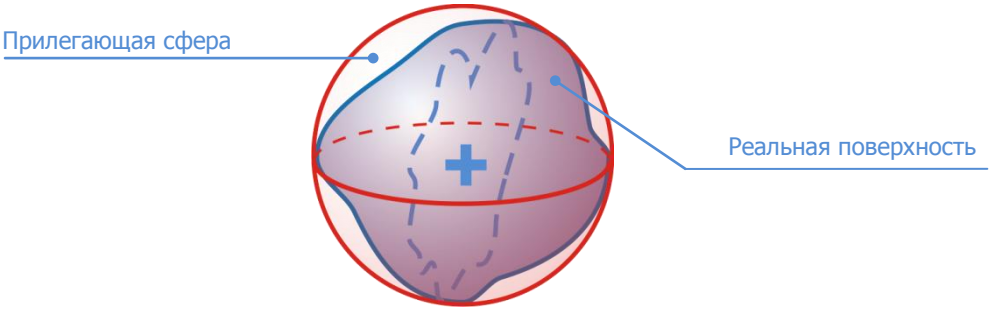


Рисунок 277. Отклонение от сферы

Назначение	Находит наибольшее отклонение реального профиля от сферы в пространстве.

Пример	`FormDeviation.Sphere(sphere, points);`
	`sphere – базовая сфера, относительно которой рассчитыватся отклонение;`
	`points – масив измеренных точек.`

Цилиндр

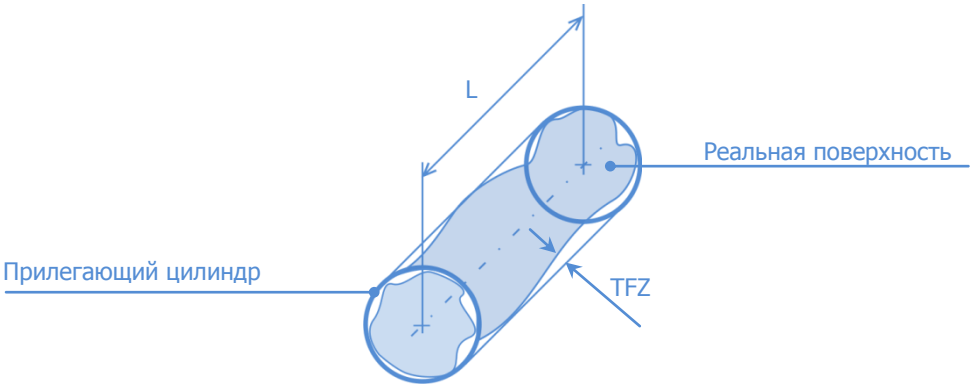


Рисунок 278. Отклонение от цилиндричности

Назначение	Находит наибольшее отклонение (TFZ) реального профиля от цилиндричности в пространстве на расстоянии L.

Пример	`FormDeviation. Cylinder(cylinder, points);`
	`cylinder – базовый цилиндр, относительно которого рассчитыватся отклонение;`
	`points – масив измеренных точек.`

Конус

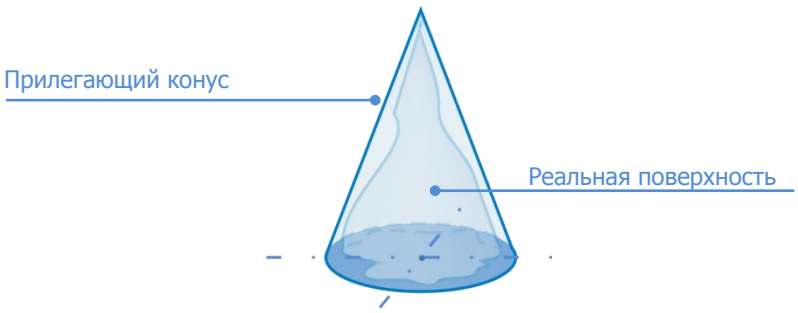


Рисунок 279. Отклонение от конусности

Назначение	Находит наибольшее отклонение реального профиля от конусности в пространстве.

Пример	`FormDeviation.Cone(cone, points);`
	`cone – базовый конус, относительно которого рассчитыватся отклонение;`
	`points – масив измеренных точек.`

Тор (версия 4.0)

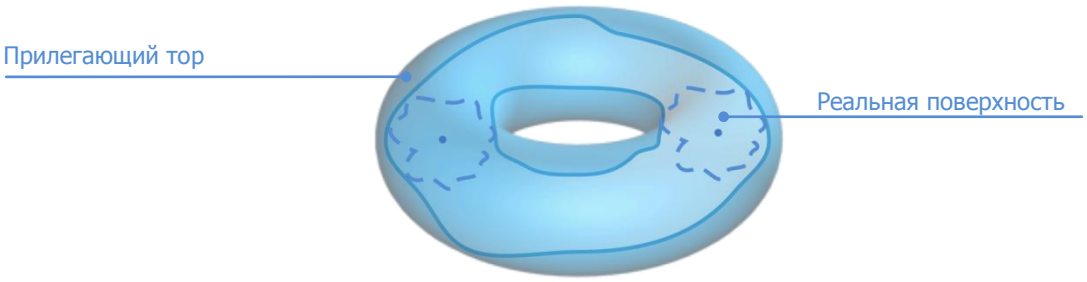


Рисунок 280. Отклонение от тора

Назначение	Находит наибольшее отклонение реального профиля от тора в пространстве.

Пример	`FormDeviation. Torus(torus, points);`
	`torus – базовый тор, относительно которого рассчитывается отклонение;`
	`points – масив измеренных точек.`

Двухмерный сплайн (версия 2.0)

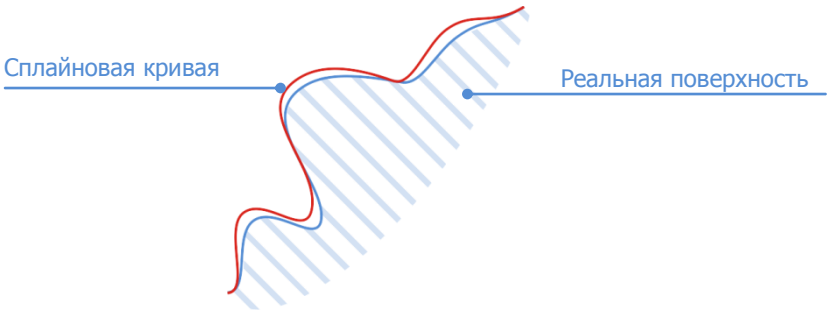


Рисунок 281. Отклонение от сплайновой кривой

Назначение	Находит наибольшее отклонение реального профиля от сплайновой кривой на плоскости.

Пример	`FormDeviation. Spline(spline, points);`
	`spline – базовая двухмерная сплайновая кривая, относительно которй рассчитывается отклонение;`
	`points – масив измеренных точек.`

Трехмерный сплайн (версия 2.0)

Назначение	Находит наибольшее отклонение реального профиля от сплайновой кривой в пространстве (см. **Рисунок 281**).

Пример	`FormDeviation.Spline(spline, points);`
	`spline – базовая трехмерная сплайновая кривая, относительно которй рассчитывается отклонение;`
	`points – масив измеренных точек.`

Сплайновая поверхность (версия 5.0)

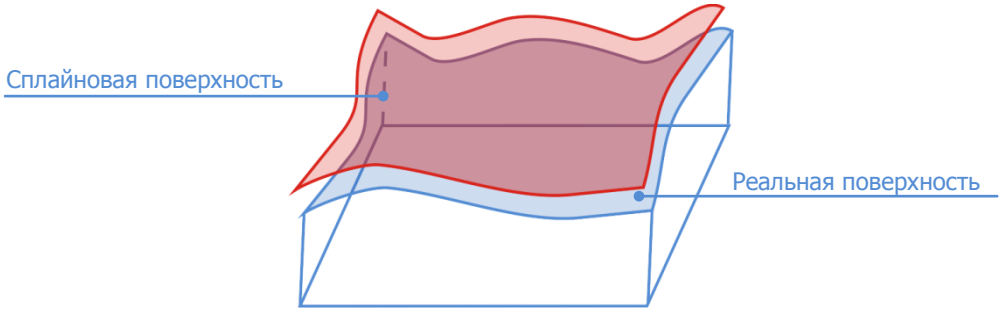


Рисунок 282. Отклонение от сплайновой поверхности

Назначение	Находит наибольшее отклонение реального профиля от сплайновой поверхности в пространстве.
Пример	`FormDeviation.Surface(spline, points);` `spline` – базовая сплайновая поверхность, относительно которй рассчитыватся отклонение; `points` – масив измеренных точек.

Среднеквадратичное отклонение формы (версия 5.0)

Среднее квадратическое расстояние h(s) между реальным профилем и средним профилем по нормали к среднему профилю в пределах нормируемого участка:

$$EFq = \sqrt{\frac{1}{L_0}\int_0^L h^2(s)ds}$$

Среднее квадратическое отклонение расстояний $h(s_1, s_2)$ между реальной поверхностью и средней поверхностью по нормали к средней поверхности в пределах нормируемого участка:

$$EFq = \sqrt{\frac{1}{S_L}\int_0^{L_1}\int_0^{L_2} h^2(s_1, s_2)ds_1 ds_2}$$

Для расчета среднеквадратичного отклонения следует применять класс FormDeviation. Функции, относящиеся к расчету среднеквадратических отклонений, оканчиваются суффиксом MeanSquared, например, FormDeviation.LineMeanSquared(…); Первым аргументом передается заменяющий элемент, вторым передается набор измеренных точек с поверхности.

Доступны следующие функции расчета среднеквадратичного отклонения, в зависимости от геометрического элемента:

- Двухмерная прямая FormDeviation.`LineMeanSquared(line, points);`
- Трехмерная прямая FormDeviation.`LineMeanSquared(line, points);`
- Двухмерная окружность FormDeviation.`CircleMeanSquared(circle, points);`

- Трехмерная окружность FormDeviation.`CircleMeanSquared(circle, points);`
- Плоскость FormDeviation.`PlaneMeanSquared(plane, points);`
- Сфера FormDeviation.`SphereMeanSquared(sphere, points);`
- Цилиндр FormDeviation.`CylinderMeanSquared(cylinder, points);`
- Конус FormDeviation.`ConeMeanSquared(cone, points);`
- Тор FormDeviation.`TorusMeanSquared(torus, points);`
- Трехмерный сплайн FormDeviation.`SplineMeanSquared(spline, points);`
- Двухмерный сплайн FormDeviation.`SplineMeanSquared(spline, points);`
- Сплайновая поверхность FormDeviation.`SurfaceMeanSquared(spline, points);`

Пример:

```
Vector2d[] points = new Vector2d[] {
   new Vector2d(0.02, 0.04),
   new Vector2d(0.5, 0.067),
   new Vector2d(0.9, 0.8) };
Line2d line = new Line(Vector2d.Zero, new Vector2d(1, 1));
Number deviation = FormDeviation.LineMeanSquared(line, points);
```

Среднее арифметическое отклонение формы (версия 5.0)

Среднее арифметическое из абсолютных значений расстояний h(s) между реальны профилем и средним профилем по нормали к среднему профилю в пределах нормируемого участка:

$$EFa = \frac{1}{L_0}\int_0^L [h(s)]ds$$

Среднее арифметическое из абсолютных значений расстояний $h(s_1, s_2)$ между реальной поверхностью и средней поверхностью перпендикулярно или радиально к средней поверхности в пределах нормируемого участка:

$$EFa = \frac{1}{S_L}\int_0^{L_1}\int_0^{L_2} [h\ (s_1, s_2)]ds_1 ds_2$$

Для расчета среднеарифметического отклонения следует применять класс FormDeviation. Функции, относящиеся к расчету среднеквадратических отклонений, оканчиваются суффиксом Mean, например, FormDeviation.LineMean(…);Первым аргументом передается заменяющий элемент, вторым передается набор измеренных точек с поверхности.

Доступны следующие функции расчета среднеарифметического отклонения, в зависимости от геометрического элемента:

- Двухмерная прямая FormDeviation.`LineMean(line, points);`
- Трехмерная прямая FormDeviation.`LineMean(line, points);`
- Двухмерная окружность FormDeviation.`CircleMean(circle, points);`
- Трехмерная окружность FormDeviation.`CircleMean(circle, points);`
- Плоскость FormDeviation.`PlaneMean(plane, points);`
- Сфера FormDeviation.`SphereMean(sphere, points);`
- Цилиндр FormDeviation.`CylinderMean(cylinder, points);`
- Конус FormDeviation.`ConeMean(cone, points);`
- Тор FormDeviation.`TorusMean(torus, points);`
- Трехмерный сплайн FormDeviation.`SplineMean(spline, points);`
- Двухмерный сплайн FormDeviation.`SplineMean(spline, points);`
- Сплайновая поверхность FormDeviation.`SurfaceMean(spline, points);`

Пример:

```
Vector2d[] points = new Vector2d[] {
   new Vector2d(0.02, 0.04),
   new Vector2d(0.5, 0.067),
   new Vector2d(0.9, 0.8) };

Line2d line = new Line(Vector2d.Zero, new Vector2d(1, 1));
Number deviation = FormDeviation.LineMean(line, points);
```

Отклонения расположения

Отклонением расположения называется отклонение реального расположения рассматриваемого элемента от номинального расположения.

Отклонения от параллельности (версия 2.0)

Для расчета отклонения от параллельности предназначен класс `Parallelism.`

В качестве базы здесь берется заменяющий элемент, определяемый пользователем.

Между двухмерными прямыми (версия 2.0)

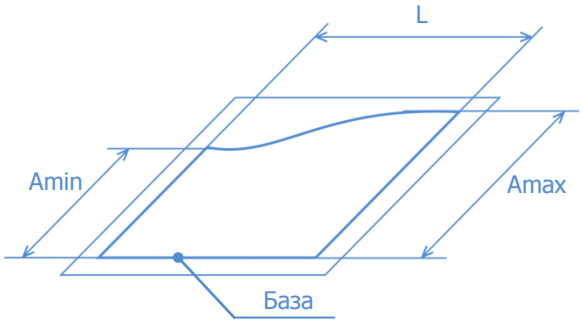


Рисунок 283. Отклонение от параллельности между двухмерными прямыми

Назначение	Определяет отклонение от параллельности между двумя прямыми в плоскости на отрезке L.

Пример	`Parallelism.LineLine(base, line, length);`
	`base – база (двухмерная прямая);`
	`line – двухмерная прямая;`
	`length – длина нормируемого участка.`

Между трехмерными прямыми (версия 2.0)

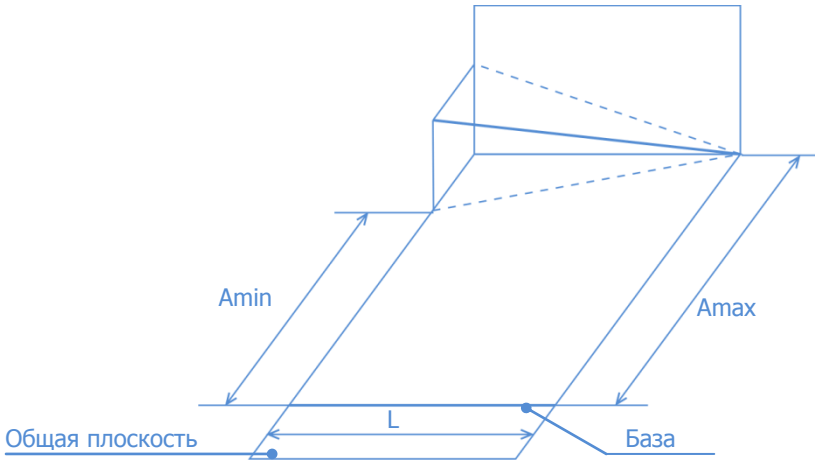


Рисунок 284. Отклонение между трехмерными прямыми

Назначение	Определяет отклонение от параллельности между двумя прямыми в пространстве на отрезке L.

Пример	`Parallelism.LineLine(base, line, length);`
	`base – база (трехмерная прямая);`
	`line – трехмерная прямая;`
	`length – длина нормируемого участка.`

Между плоскостью и прямой (версия 2.0)

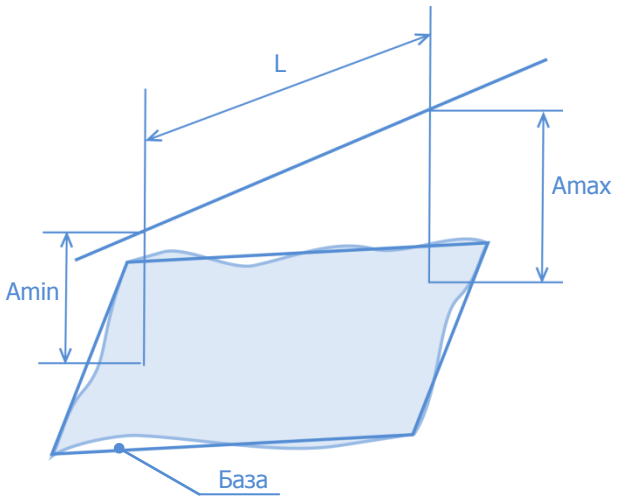


Рисунок 285. Отклонение от параллельности между плоскостью и прямой

Назначение	Определяет отклонение от параллельности между плоскостью и прямой в пространстве на отрезке L.

Пример	`Parallelism.LinePlane(base, line, length);`
	`base – база (плоскость);`
	`line – трехмерная прямая`
	`length – длина нормируемого участка.`

Между двумя плоскостями (версия 2.0)

Нормируемый участок определяется автоматически исходя из переданных точек на втором элементе. Это в свою очередь налагает некоторые требования на выбор положения точек, т.е. они должны быть расположены как минимум в крайних положениях нормируемого участка

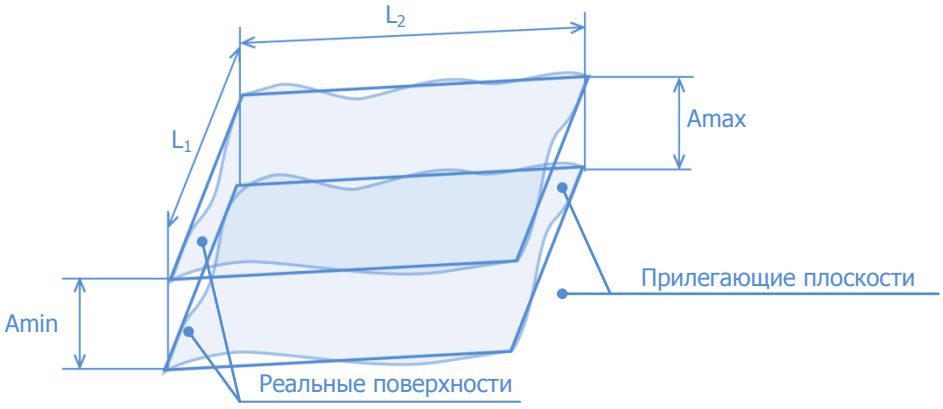


Рисунок 286. Отклонение параллельности между двумя поскостями

Назначение	Определяет отклонение от параллельности между двумя плоскостями.

Пример	`Parallelism.PlanePlane(base, points);`
	`base – база (плоскость);`
	`points – массив точек на другой плоскости.`

Отклонения от перпендикулярности (версия 2.0)

Для удобства расчета отклонения от перпендикулярности существует специальный класс `Perpendicularity.` В качестве базы здесь берется заменяющий элемент, определяемый пользователем.

Между двухмерными прямыми (версия 2.0)

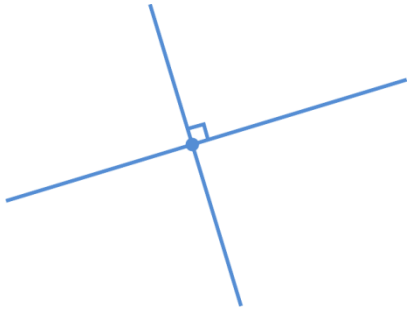


Рисунок 287. Отклонение от перпендикулярности между двумя прямыми

Назначение	Определяет отклонение от перпендикулярности между двумя прямыми.

Пример	`Perpendicularity.`LineLine(base, line, length);
	`base – база (двухмерная прямая);`
	`line – двухмерная прямая`
	`length – длина нормируемого участка.`

Между плоскостью и прямой (версия 2.0)

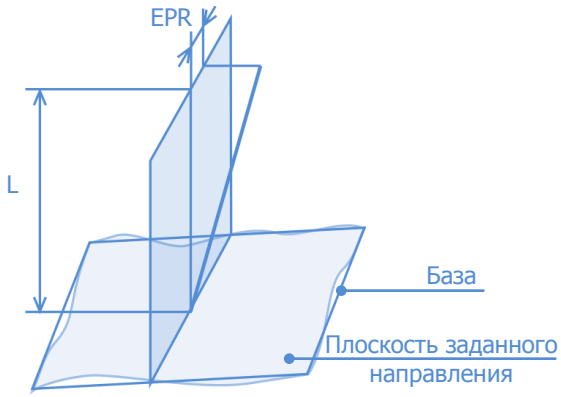


Рисунок 288. Отклонение от перпендикулярности между прямой и плоскостью

Назначение	Определяет отклонение от перпендикулярности между двумя прямыми.

Пример	`Perpendicularity.`LinePlane(base, line, length);
	`base – база (плоскость);`
	`line – трехмерная прямая`
	`length – длина нормируемого участка.`

	Возможен вариант с расчетом используя плоскость заданного направления:
	`Perpendicularity.`LinePlane(base, line, plane, length);
	`base – база (плоскость);`
	`line – трехмерная прямая`
	`plane – плоскость заданного направления`
	`length – длина нормируемого участка.`

Между двумя плоскостями (версия 2.0)

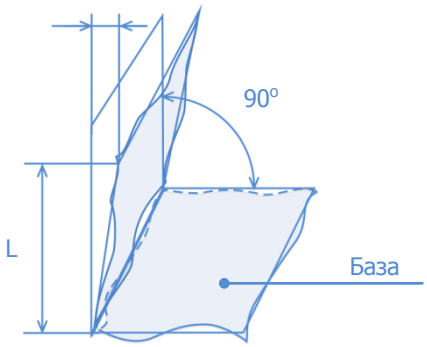


Рисунок 289. Отклонение от перпендикулярности между двумя плоскостями

Назначение	Определяет отклонение от перпендикулярности между двумя плоскостями.

Пример	`Perpendicularity.`PlanePlane(base, plane, length);
	`base – база (плоскость);`
	`plane – плоскость`
	`length – длина нормируемого участка.`

Отклонения наклона

Для расчета отклонения наклона следует использовать функции расчета углов (см. «Расчет углов» на стр. 207).

Отклонения от соосности (версия 2.0)

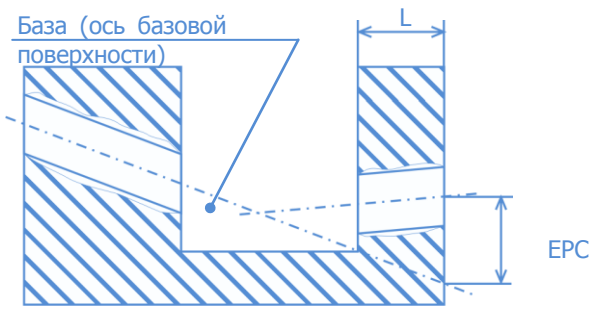


Рисунок 290. Отклонение от соосности

Назначение	Определяет отклонение от соосности.

Пример	`NumberDeviation = Coaxiality.Evaluate(axis1, axis2, plane1, plane2);`
	`axis1 – первая ось (трехмерная прямая);`
	`axis2 – вторая ось (трехмерная прямая);`
	`plane1, plane2 – две плоскости, которые определяют нормируемый участок;`

Отклонения позиционные

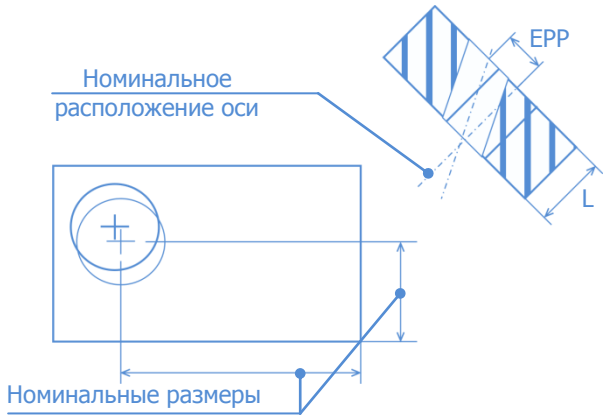


Рисунок 291. Позиционные отклонения

Позиционное отклонение – наибольшее расстояние между реальным расположением элемента детали (его центра, оси или плоскости симметрии) и его номинальным расположением в пределах нормируемого участка. Позиционное отклонение можно нормировать для элементов, находящихся в плоскости, или в пространстве, или в заданном направлении. Данный вид отклонения не сложно рассчитать, используя основную функциональность.

Отклонения суммарные

Расчет суммарных отклонений следует использовать соответствующие комбинации других функций языка.

Общематематические операции

Степенные функции

Квадратный корень

Формат	Basic.Sqrt(x)
Вход	Число
Выход	Квадратный корень числа

Возведение в степень

Формат	Basic.Pow(x, y)
Вход	Возводимое число, Степень
Выход	x^y

Натуральный логарифм

Формат	Basic.Ln(x)
Вход	Число
Выход	Натуральный логарифм переданного числа

Логарифм по основанию

Формат	Basic.Log(x, base)
Вход	Число Основание логарифма
Выход	Логарифм числа x по основанию base

Экспонента в заданной степени

Формат	Basic.Exp(power)
Вход	Степень
Выход	e^{power}

Тригонометрические функции

Косинус

Формат	Basic.Cos(angle)
Вход	Угол (в радианах)
Выход	Косинус переданного угла

Синус

Формат	Basic.Sin(angle)
Вход	Угол (в радианах)
Выход	Синус переданного угла

Тангенс

Формат	Basic.Tan(angle)
Вход	Угол (в радианах)
Выход	Тангенс переданного угла

Котангенс

Формат	Basic.Cotan(angle)
Вход	Угол (в радианах)
Выход	Котангенс переданного угла

Гиперболический косинус

Формат	Basic.Cosh(angle)
Вход	Угол (в радианах)
Выход	Гиперболический косинус переданного угла

Гиперболический синус

Формат	Basic.Sinh(angle)
Вход	Угол (в радианах)
Выход	Гиперболический синус переданного угла

Гиперболический тангенс

Формат	Basic.Tanh(angle)
Вход	Угол (в радианах)
Выход	Гиперболический тангенс переданного угла

Гиперболический котангенс

Формат	Basic.Cotanh(angle)
Вход	Угол (в радианах)
Выход	Гиперболический котангенс переданного угла

Арккосинус

Формат	Basic.Arccos(x)
Вход	Число $-1 \leq x \leq 1$
Выход	Угол (в радианах)

Арксинус

Формат	Basic.Arcsin(x)
Вход	Число $-1 \leq x \leq 1$
Выход	Угол (в радианах)

Арктангенс

Формат	Basic.Arctan(x)
Вход	Число
Выход	Угол (в радианах)

Функции округления

Наименьшее целое, превосходящее данное

Формат	Basic.Ceiling(x)
Вход	Число
Выход	Наименьшее целое, превосходящее данное

Наибольшее целое, меньше данного

Формат	Basic.Floor(x)
Вход	Число
Выход	Наибольшее целое, меньше данного

Простое округление

Формат	Basic.Round(x)
Вход	Число
Выход	Наибольшее целое, меньше данного

Константы

Число π

Формат	Basic.Pi
Вход	
Выход	Значение числа «Пи»

Число *e*

Формат	Basic.Exponent
Вход	
Выход	Значение экспоненты

Другое

Получение абсолютного значения

Формат	Basic.Abs(x)
Вход	Число
Выход	Модуль переданного числа |x|

Сравнение с учетом погрешности вычислений

В расчетах используется форма с плавающей точкой для хранения чисел, что приводит к тому, что в результате расчетов значение числа накапливает погрешность. Данная погрешность приводит к тому, что простое сравнение может дать ложный результат, для решения этой проблемы следует использовать метод `Basic.ApproxEqual`

Формат	Basic.ApproxEqual(a, b)
Вход	Первое число, Второе число
Выход	Истина, если первое число приблизительно равно второму с учетом допустимой погрешности по умолчанию, иначе ложь

Сравнение с учетом указанной допустимой погрешности

Формат	Basic.ApproxEqual(a, b, tolerance)
Вход	Первое число, Второе число, Допустимая погрешность
Выход	Истина, если первое число приблизительно равно второму с учетом указанной допустимой погрешности, иначе ложь

Поддержка

Для разрешения возникающих проблем с *программным обеспечением* можно обращаться любым из перечисленных ниже способов:

Web: www.technocoord.ru

E-mail: soft@toolmaker.ru

Phone: +7 (351) 210-49-29