Geoshape 约束求解引擎软件
单机版 V1.1.030

使用说明

文档版本 01
发布日期 2024-6-30

深圳泊松软件技术有限公司

深圳泊松软件技术有限公司

地址：广东省深圳市龙岗区坂田街道岗头社区天安云谷产业园二期 4 栋 2301

邮编：518129

网址：https://www.poissonsoft.com

目录

1 概述

欢迎使用单机版 Geoshape 约束求解引擎软件（Geoshape Constraint Solver），本软件包含“2D 几何约束求解器引擎”和“2D 几何约束求解器”，“2D 几何约束求解器引擎”用于管理“2D 几何约束求解器”。本章对文档范围以及软件进行了简要的介绍，目的是让您更加熟练地掌握本软件的 API 使用方法。

📖 说明
为了方便您阅读该手册，在该手册中约定“Geoshape 约束求解引擎软件”简称为“本软件”；“2D 几何约束求解器引擎”简称为“2D 求解器引擎”；“2D 几何约束求解器”简称为“2D 求解器”。

1.1 文档简介

该文档主要向您介绍本软件包括的相关概念、主要功能及其对应的 API 接口。

1.2 产品介绍

Geoshape 约束求解引擎软件（Geoshape Constraint Solver）是一个基于几何元素表达和几何约束条件进行约束系统求解的软件组件，被广泛应用于草图轮廓表达、零件建模参数表达、装配约束等场景，为快速确定设计意图表达、检查干涉、模拟运动提供了强有力的支持，可帮助最终用户提高生产效率。

本软件目前主要支持二维约束系统的求解，通过为几何元素添加几何约束来满足草图的设计需求。本软件支持的几何元素包括点、直线、线段、圆、椭圆、各种参数化曲线及变量，有关几何元素的基本概念，请参见 2.1 几何元素；支持的几何约束包括距离、角度、平行、垂直、等方向、等半径、固定、方程等，有关几何约束的更多类型与基本概念，请参见 2.2 几何约束。

2 基本概念

本章涉及 Geoshape 约束求解引擎软件操作中的重要概念，同时引用了部分接口对功能加以说明。

2.1 几何元素

本软件支持以下几何元素类型：

解析几何元素

点、直线、线段、圆和椭圆。

参数化几何元素

- 控制点样条曲线：
  由一组控制点定义的曲线，这些控制点决定了曲线的大致形状，但曲线本身不一定通过所有控制点，适用于需要灵活调整曲线形状的场景。如果应用程序需要精确定义一条样条曲线，可以在添加控制点时传入插值点。更多详细信息，请参见 7 样条曲线。
- 插值点样条曲线：
  由一组数据点定义的曲线，曲线必须穿过给定的数据点，适用于需要确保曲线严格按照指定数据点进行绘制的场景。更多详细信息，请参见 7 样条曲线。
- 圆锥曲线：
  一种用于表示有界圆锥形截面区域的几何元素类型，具有三种表现形式，分别是椭圆、双曲线和抛物线。更多详细信息，请参见 8 圆锥曲线。
- 偏移曲线：
  将一条给定曲线偏移一定距离后形成的一条新曲线。本软件支持偏移的几何元素类型有椭圆、样条曲线、圆锥曲线、自定义参数曲线、自定义回调曲线、复制曲线以及偏移曲线。更多详细信息，请参见 9 偏移曲线与复制曲线。
- 复制曲线：
  通过仿射变换矩阵生成的一条与指定曲线形状相同的新曲线。本软件支持复制的几何元素类型有偏移曲线、样条曲线、圆锥曲线、自定义参数曲线及自定义回调曲线。更多详细信息，请参见 9 偏移曲线与复制曲线。

- 自定义参数曲线：
  一条由变量、方程、局部坐标系（LCS）的原点及 LCS 绕原点的旋转参数定义的参数曲线。更多详细信息，请参见 10 自定义曲线。
- 自定义回调曲线：
  一条由变量、回调函数、局部坐标系（LCS）的原点及 LCS 绕原点的旋转参数定义的回调曲线。应用程序根据需求来定义回调函数，求解时本软件会根据回调函数来获取自定义回调曲线的数据。 更多详细信息，请参见 10 自定义曲线。

特殊几何元素

- 刚性集合：
  一组几何元素的集合，刚性集合中的所有几何元素的相对位置不会发生改变，但是可以进行整体的平移和旋转。更多详细信息，请参见 4.4 刚性集合。
- 可伸缩集合：
  一组几何元素的集合，能够在任意方向上等比例缩放。更多详细信息，请参见 4.5 可伸缩集合。
- 单向可伸缩集合：
  可伸缩集合的一种特殊形式，只能在一个指定方向上进行缩放，集合的长宽比会发生变化。更多详细信息，请参见 4.6 单向可伸缩集合。
- 双向可伸缩集合：
  可伸缩集合的一种特殊形式，可以在两个垂直方向上根据各自的缩放因子独立进行缩放。更多详细信息，请参见 4.7 双向可伸缩集合。
- 阵列几何元素：
  一种特殊的几何元素，用于存储和执行阵列约束的参考元素与阵列值。本软件包括线性与环形阵列几何元素，线性阵列几何元素有一个参考元素或两个参考元素，环形阵列几何元素只有一个参考元素。更多详细信息，请参见 4.8 阵列几何元素。
- 相对变换几何元素：
  一种特殊的几何元素，用于存储和执行等相对变换约束的刚性变换。更多详细信息，请参见 5.2.13 等相对变换约束。
- 变量：
  一种特殊的几何元素，能够被加入到方程或不等式中形成约束，有关变量几何元素的详细信息，请参见 2.3 变量、方程和不等式中的普通变量部分。

2.2 几何约束

几何约束是草图设计过程中用于确定几何元素之间的相对位置、大小、形状和方向等关系的限制条件。

几何约束分为尺寸约束（数值约束）与逻辑约束。

尺寸约束

尺寸约束采用确切的值来表达几何元素之间相对位置信息（如距离、夹角）或形状信息（如半径）。

尺寸约束：

- 角度约束
- 距离约束
- 方向距离约束
- 平行距离约束
- 垂直距离约束
- 半径约束
- 椭圆的长轴半径约束
- 椭圆的短轴半径约束
- 曲线长度约束
- 圆锥曲线的 rho 约束
- 弧长约束
- 轮廓偏置约束

逻辑约束

逻辑约束用以表达几何元素之间非数值的、结构性的约束信息，如水平、竖直、平行等。

逻辑约束：

- 水平约束
- 竖直约束
- 平行约束
- 垂直约束
- 重合约束
- 同心约束
- 相切约束
- 对称约束
- 中点约束
- 等距离约束
- 等半径约束
- 等参数约束
- 等方向约束
- 等相对变换约束
- 等一阶导数约束
- 等二阶导数约束
- 等曲率约束
- 等曲率导数约束
- 阵列约束
- 固定约束
- 法线约束

当尺寸约束具有特定值时，对几何元素的约束效果等同于逻辑约束。例如，角度约束的值为 0 可以等效为平行约束；距离约束的值为 0 可以等效为重合约束或相切约束。此时，建议使用等效的逻辑约束，有助于成功求解出过约束模型的解决方案。

在以上几何约束中，部分几何约束较为复杂，如方向距离约束、阵列约束和参考线距离约束（包括平行距离约束和垂直距离约束）等，2D 求解器会进行单独处理，如果应用程序想要添加这些复杂几何约束，可以调用它们各自对应的接口。然而，对于其余简单的逻辑约束，2D 求解器开放了统一的接口供应用程序调用，意味着应用程序添加

简单逻辑约束时都需要调用同一个的接口，通过传入不同的几何约束类型来进行区分。

本软件还能处理特殊的几何约束，如方程和不等式。方程和不等式可以用来限制曲线或曲面上点的位置，以符合特定的几何条件或关系。有关方程和不等式的基本概念，请参见 2.3 变量、方程和不等式。

2.2.1 约束对齐方式与约束方位

本软件中的任意几何约束都具有一个约束对齐方式（Alignment）和约束方位（Orientation）。但是只有在有方向的几何元素之间关联了几何约束时，本软件才会考虑该几何约束的约束对齐方式或约束方位对解决方案的影响。

2.2.1.1 约束对齐方式

约束对齐方式

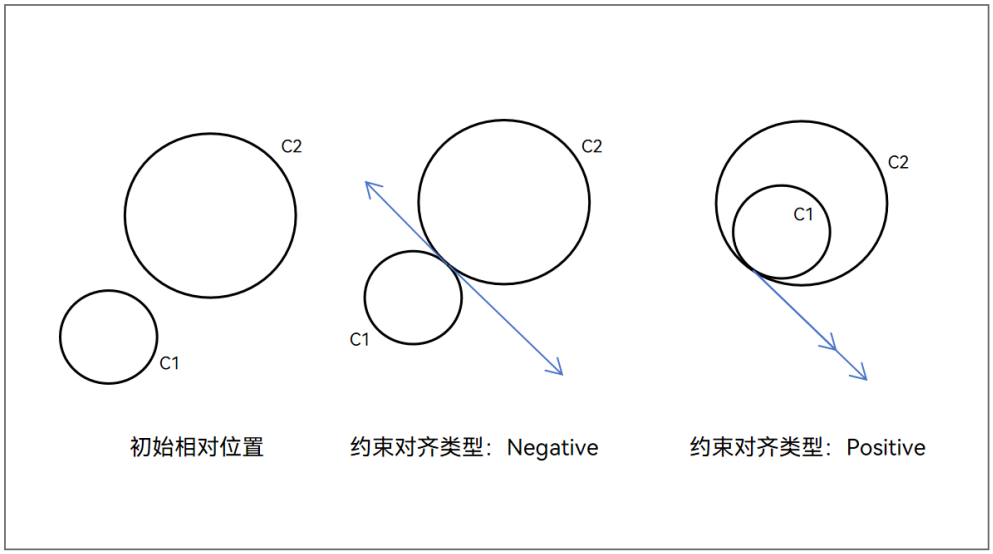
几何约束的约束对齐方式主要作用于与其相关联的几何元素的方向或切线方向。几何约束的对齐方式有 Positive（同向对齐方式）、Negative（反向对齐方式）、Current（当前对齐方式）、Any（任意方式）。

为了让您更好地理解约束对齐方式的使用，下面展示了一个典型示例——两个圆之间的相切约束。应用程序向本软件添加两个圆和一个相切约束。

- 当约束对齐方式为 Positive 时，本软件将按照同向对齐方式进行求解，求解出的两个圆的切线方向相同，它们在解决方案中的相对位置为内切。
- 当约束对齐方式为 Negative 时，本软件将按照反向对齐方式进行求解，求解出的两个圆的切线方向相反，它们在解决方案中的相对位置为外切。
- 当约束对齐方式为 Current 时，本软件将按照当前对齐方式（同向对齐方式与反向对齐方式的其中一种）进行求解。
- 当约束对齐方式为 Any 时，本软件将按照任意方式进行求解。

如图 2-1 更新约束对齐方式所示，圆 C1 与圆 C2 的初始相对位置不相交，此时在 C1 与 C2 之间添加相切约束，根据不同的约束对齐方式，将产生不同的解决方案，图中蓝色箭头分别表示 C1 与 C2 在相切位置的切线方向。

图 2-1 更新约束对齐方式



除了圆与圆之间的相切约束，约束对齐方式还能影响其他几何约束的求解结果。例如应用程序向本软件先后添加两条直线 L1 与 L2，以及一个小于 90°的角度约束。在初始阶段，两条直线被视为是同一方向。固定 L1，此时如果将约束对齐方式设置为 Positive，本软件会按照同向对齐方式进行求解，L2 的方向不变；但是如果此时将约束对齐方式设置为 Negative，本软件会根据反向对齐方式进行求解，从而改变 L2 的方向。

本软件默认保留与几何元素初始相对位置相同的解决方案。应用程序可以调用 **updateConstraintAlignment** 接口来请求不同的解决方案。

约束对齐方式 Current

当约束对齐方式为 Current 时，本软件会根据几何元素的当前位置进行计算，从而确定解决方案的求解方式——同向对齐（切线方向相同）或反向对齐（切线方向相反）。

以下给出一个详细的示例，解释了 Current 类型对解决方案的影响。现有一个圆与一条直线，在圆与直线之间添加一个相切约束。

将相切约束的约束对齐方式设置为 Current 后，2D 求解器的求解方式如下：

- 相切约束有帮助点
  1. 获取直线的方向 dir。
  2. 获取圆心指向帮助点的向量 dir2。
  3. 计算 dir2 的相切单位向量 tangentDir。
  4. 计算 dir × tangentDir。
     - dir × tangentDir < 0，直线方向与相切向量方向相反。
     - dir × tangentDir > 0，直线方向与相切向量方向相同。

- 相切约束无帮助点
  1. 获取直线的方向 dir。
  2. 获得直线原点指向圆心的向量 vecLineToCenter。
  3. 计算 dir × vecLineToCenter。
     - dir × vecLineToCenter < 0，直线方向与相切向量方向相反。
     - dir × vecLineToCenter > 0，直线方向与相切向量方向相同。

2.2.1.2 约束方位

约束方位

几何约束的约束方位主要作用于与其相关联的几何元素的相对位置，其类型有 Positive（同向方式）、Negative（反向方式）、Current（当前方式）、Any（任意方式）。

例如，应用程序先后创建了直线 L1 与直线 L2，并且在它们之间添加了距离约束，L2 此时位于 L1 正方向的右侧，固定 L1。

- 当约束方位为 Positive 时，本软件将按照同向方式（左手定则）进行求解， 求解后，L2 将在 L1 正方向的左侧。
- 当约束方位为 Negative 时，本软件将按照反向方式（右手定则）进行求解，求解后，L2 将在 L1 正方向的右侧。
- 当约束方位为 Current 时，本软件将按照当前方式（同向方式与反向方式的其中一种）进行求解。
- 当约束方位为 Any 时，本软件将按照任意方式（同向方式与反向方式的其中一种）进行求解。

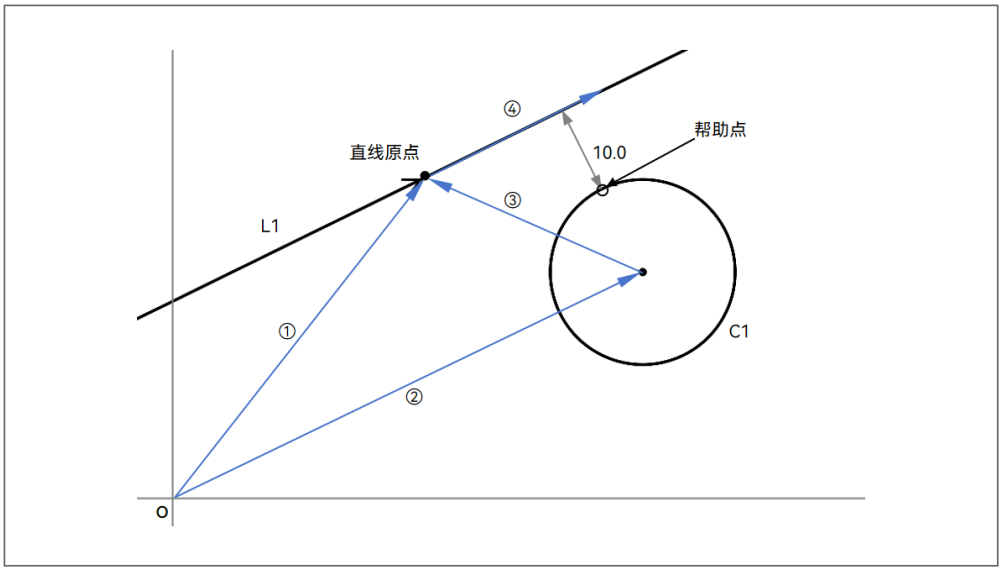
本软件默认保留与几何元素初始相对位置相同的解决方案。应用程序可以调用 **updateConstraintOrientation** 接口来请求不同的解决方案。

约束方位 Current

当约束方位为 Current 时，本软件会根据几何元素的当前位置进行计算，从而确定解决方案的求解方式——同向（左手定则）或反向（右手定则）。

以下给出一个详细的示例，解释了 Current 类型对解决方案的影响。如图 2-2 圆与直线之间的距离约束所示，现有一个圆 C1 与一条直线 L1，在 C1 与 L1 之间添加一个距离约束，图中蓝色箭头表示向量。

图 2-2 圆与直线之间的距离约束



将距离约束的约束方位设置为 Current 后，2D 求解器的求解方式如下：

1. 向量③ = 向量① - 向量②
2. ● 向量③ × 向量④ < 0 ,以反向方式求解，符合右手定则。
   ● 向量③ × 向量④ > 0 ,以同向方式求解，符合左手定则。

上图中，向量③ × 向量④ < 0 ,以反向方式求解，符合右手定则，故帮助点位于直线的右侧。

2.2.1.3 适用范围

目前本软件中，实际求解结果会受到约束对齐方式或约束方位影响的几何约束如表 2-1 受影响的几何约束所示。

表 2-1 受影响的几何约束

几何约束	名称	约束对齐方式	约束方位
Horizontal	水平约束	√	×
Vertical	竖直约束	√	×
Parallel	平行约束	√	×
Perpendicular	垂直约束	√	×
Coincident	重合约束	√	×
Tangent	相切约束	√	×
Distance	距离约束	√	√
DirectionDistance	方向距离约束	√	×
ParallelDistance	平行距离约束	√	×

几何约束	名称	约束对齐方式	约束方位
PerpendicularDistance	垂直距离约束	√	×
Angle	角度约束	√	√
Symmetric	对称约束	√	×
EqualDirection	等方向约束	√	×
Normal	法线约束	√	×
Patterned	阵列约束	√	×
EQUAL_FIRST_DERIVATIVE	等一阶导数约束	√	×
EQUAL_SECOND_DERIVATIVE	等二阶导数约束	√	×
EQUAL_CURVATURE	等曲率约束	√	×
EQUAL_CURVATURE_DERIVATIVE	等曲率导数约束	√	×
EQUAL_RELATIVE_TRANSFORM	等相对变换约束	×	×
CONTOUR_OFFSET	轮廓偏置约束	×	×

下面详细列出各几何约束与几何元素组合中，会受到约束对齐方式或约束方位影响的情况，表格中的曲线包括椭圆、控制点样条曲线、插值点样条曲线、圆锥曲线、偏移曲线、复制曲线、自定义参数曲线以及自定义回调曲线。A 表示支持约束对齐方式（Alignment）, O 表示支持约束方位（Orientation），空白表示都不支持。

表 2-2 水平约束

直线	线段	曲线
A	A	A

表 2-3 竖直约束

直线	线段	椭圆
A	A	A

表 2-4 平行约束

	直线	线段	椭圆
直线	A	A	A
线段	A	A	A

	直线	线段	椭圆
椭圆	A	A	A

表 2-5 垂直约束

	直线	线段	椭圆
直线	A	A	A
线段	A	A	A
椭圆	A	A	A

表 2-6 重合约束

	直线	线段
直线	A	A
线段	A	A

表 2-7 相切约束

	直线	线段	圆	曲线
直线	A	A	A	A
线段	A	A	A	A
圆	A	A	A	A
曲线	A	A	A	A

表 2-8 距离约束

	点	直线	线段	圆	曲线
点		O	O		O
直线	O	A,O	A,O	O	O
线段	O	A,O	A,O	O	O
圆		O	O		O
曲线	O	O	O	O	A,O

表 2-9 方向距离约束

	点	圆	曲线
点	A	A	A
圆	A	A	A
曲线	A	A	A

表 2-10 平行距离约束

	点	圆	曲线
点	A	A	A
圆	A	A	A
曲线	A	A	A

表 2-11 垂直距离约束

	点	圆	曲线
点	A	A	A
圆	A	A	A
曲线	A	A	A

表 2-12 角度约束

	直线	线段	曲线
直线	A,O	A,O	A,O
线段	A,O	A,O	A,O
曲线	A,O	A,O	A,O

表 2-13 对称约束

	点	直线	线段	圆	曲线
点	A				
直线		A			
线段			A		
圆				A	
曲线					A

表 2-14 法线约束

	直线	线段	圆	曲线
直线			A	A
线段			A	A
圆	A	A	A	A
曲线	A	A	A	A

表 2-15 等方向约束

	直线	线段	曲线
直线			A

	直线	线段	曲线
线段			A
曲线	A	A	A

表 2-16 等一阶导数约束

	曲线
曲线	A

表 2-17 等二阶导数约束

	曲线
曲线	A

表 2-18 等曲率约束

	曲线
曲线	A

表 2-19 等曲率导数约束

	曲线
曲线	A

约束方位与约束对齐方式的更改只有在调用全量求解与增量求解时才会生效，应用程序修改几何约束的约束方位后，建议调用 **reEvaluate** 接口进行增量求解，与全量求解相比起来，求解效率更高。

说明
只有在有方向的几何元素之间关联了几何约束，2D 求解器才会考虑约束对齐方式与约束方位对解决方案的影响。

2.2.2 自动逻辑约束与自动尺寸约束

自动逻辑约束与自动尺寸约束用于对约束系统中处于欠约束状态下但满足约束条件的几何元素自动添加相关的几何约束，使得几何元素在约束系统中的位置保持不变，更多详细信息，请参见 12 自动逻辑约束与自动尺寸约束。

2.3 变量、方程和不等式

本软件能够通过求解方程和不等式来找到变量的值。

变量

本软件支持以下三种类型的变量：

- 普通变量：
  - Dimension：与尺寸约束绑定的变量，称为尺寸约束变量，变量的值被用作尺寸约束的大小。
  - Variable：与尺寸约束无关的变量，称为自由变量。
- 几何元素参数变量：
  - Radius：圆的半径。
  - MajorRadius：椭圆的长轴半径。
  - MinorRadius：椭圆的短轴半径。
  - X：点、圆心等的 X 坐标。
  - Y：点、圆心等的 Y 坐标。
  - Distance：直线距坐标原点的距离参数或偏移曲线的偏移距离。
  - Angle：直线、椭圆、自定义参数曲线、自定义回调曲线、复制曲线、刚性集合、可伸缩集合、单向可伸缩集合、双向可伸缩集合的角度参数。
  - PATTERN_VALUE：阵列几何元素中，一个参考元素的距离值，或两个参考元素的第一个距离值。
  - PATTERN_VALUE2：阵列几何元素中，两个参考元素的第二个距离值。
- 帮助参数变量：
  - HELP_PARAMETER1：几何约束的第一个帮助参数。
  - HELP_PARAMETER2：几何约束的第二个帮助参数。
  - HELP_PARAMETER3：几何约束的第三个帮助参数。
  - HELP_PARAMETER4：几何约束的第四个帮助参数。

帮助参数主要用于指示几何约束在支持帮助参数的几何元素上的期望生效位置，可作用于椭圆与参数化曲线。一般情况下，帮助参数的具体类型由几何约束关联几何元素的次序来决定，例如在样条曲线与点之间添加距离约束，帮助参数作用在样条曲线上，如果样条曲线是距离约束关联的第一个几何元素，那么帮助参数类型为 HELP_PARAMETER1；如果样条曲线是第二个几何元素，那么帮助参数类型为 HELP_PARAMETER2。当几何约束涉及四个几何元素且每个几何元素都支持帮助参数时，该几何约束将支持上述四种类型的帮助参数。

特殊情况下，几何约束中多个帮助参数关联到一个几何元素上，如在一条样条曲线的两个位置之间添加等一阶导数约束，两个帮助参数作用在同一样条曲线上，几何约束在样条曲线上的生效的第一个位置的帮助参数类型为 HELP_PARAMETER1，生效的第二个位置的帮助参数类型为 HELP_PARAMETER2。

普通变量与几何元素参数变量用于定义方程与不等式，在定义方程与不等式时，本软件采用变量描述符的方式来标识这些变量，以便用户查询和使用。变量描述符由几何元素 ID、变量类型以及变量名组成。例如，当您需要将圆的半径作为变量加入到方程或不等式中时，您需要使用圆的 ID、Radius 变量类型以及变量名字符串来定义该变量。

命名变量名时需要遵循以下规则：

- 变量名由英文字母（a~z, A~Z），数字（0~9）和下划线（-）构成。
  - 合法变量名：x, x1, A12, Center_x

  - 非法变量名：r 10, x$, %5
- 变量名第一个字符必须为字母。
  - 合法变量名：p1_x, r10
  - 非法变量名：5y, _a
- 变量名长度不能为 0。

方程和不等式

在本小节关于方程和不等式的介绍中，我们将使用 v 来表示普通变量与几何元素参数变量。本软件可以求解以下类型的方程和不等式：

方程

- 线性方程
  a1*v1 + a2*v2 + a3*v3 + ... + c = 0
  其中系数 a1、a2、a3 等和常数 c 为数值，v1、v2、v3 等表示变量。
- 非线性方程
  f(v1, v2, v3...) = 0
  其中函数 f 可以是变量 v 的任何函数。2D 求解器只需要知道哪些变量在哪个方程中，不必知道变量的具体使用方法。2D 求解器可以查询函数在指定变量处的值和导数。非线性方程可用于表示本软件不直接支持的复杂几何约束。例如，应用程序可以使用非线性方程来约束一个轮廓的周长或面积，或将点约束为封闭轮廓的质心等。

不等式

- 线性不等式
  - a1*v1 + a2*v2 + a3*v3 + ... + c <= 0
  - a1*v1 + a2*v2 + a3*v3 + ... + c < 0
  - a1*v1 + a2*v2 + a3*v3 + ... + c >= 0
  - a1*v1 + a2*v2 + a3*v3 + ... + c > 0
- 非线性不等式
  - f(v1, v2, v3...) <= 0
  - f(v1, v2, v3...) < 0
  - f(v1, v2, v3...) >= 0
  - f(v1, v2, v3...) > 0

  有关变量、方程和不等式的详细信息，请参见 11 变量、方程和不等式 。

2.4 求解和拖拽

2.4.1 通用操作

应用程序将数据传递给本软件后，2D 求解器会执行一些通用的操作来快速求解满足几何约束的模型。

在求解过程中，2D 求解器将尝试执行以下操作：

- 确保模型中几何约束的一致性。
  2D 求解器会确保模型中的所有几何约束都具有一致性（一致性是指虽然过约束但是约束的结果一致，那么问题可解），并且能够改变非刚性的几何约束和方程的值，这样便于得到有效的解决方案。
- 求解方程并找到变量的值。
  在求解过程中，2D 求解器会求解模型中的所有方程，并找到所有变量的值，从而确定模型的状态。
- 确定应用于几何元素的变换。
  为了满足当前定义的所有几何约束，2D 求解器会确定应用于几何元素的变换，比如调整几何元素的位置和方向。
- 设置求解状态码。
  状态码用于表示模型在求解过程中的状态，例如成功、失败或其他状态。本软件能够通过设置状态码反映模型求解出现的问题，为进一步的操作或决策提供依据。

当存在多个模型分区待求解时，本软件会并行处理所有分区，以提高求解的效率和速度，从而支持实时的模型分析。关于模型分区的详细信息，请参见 2.4.3 全量求解和增量求解的“模型分区”部分。

本软件会根据几何元素的初始相对位置、几何约束、方程以及应用程序传递给本软件的选项等来确定唯一的解决方案。这些选项主要影响 2D 求解器对解决方案的选择，下面几个小节中将对选项进行简要描述。

📖 说明
应用程序传递给本软件的选项不能影响处于良好约束状态的模型的解决方案。

2.4.2 线性容差值和角度容差值

容差值是指 2D 求解器进行比较时使用的数值。

2D 求解器具有两个容差值：线性容差值和角度容差值，2D 求解器比较角度时使用角度容差值，比较长度时使用线性容差值。例如，当两点之间的距离小于 2D 求解器的线性容差时，2D 求解器会将两点视为重合。

应用程序能够使用 **setTolerances** 与 **getTolerances** 接口来设置与获取容差值，建议应用程序在调用 2D 求解器接口之前，首先调用 **setTolerances** 接口设置约束系统的容差。容差的取值范围为(0,1.0e-2)，超出范围的输入值会导致容差设置失败。如果容差设置失败或者应用程序没有调用 **setTolerances** 接口，2D 求解器将使用默认的容差值。本软件中，默认的线性容差值为 1.0e-8，默认的角度容差值为 1.0e-11。

2.4.3 全量求解和增量求解

应用程序可以选择全量方法或增量方法来进行求解，既能够实现对整个模型（全部模型分区）执行完整的求解，又能够针对变更部分（变更的模型分区）进行增量求解。

模型分区

定义

如果一个模型包含两组或两组以上完全不连接的几何元素组（即相互之间没有任何共享几何约束），本软件会自动识别模型中的断开区域，并将这些区域保存在称为“分区”的内部结构中。

特性

- 每个分区都可以独立于其他分区进行求解。应用程序对单个分区中的几何元素或几何约束所做的更改不会影响其他分区的数据。例如，当模型被划分成一个个分区时，添加几何约束只会变更关联该约束的几何元素所在的分区。
- 当调用 **reEvaluate** 函数时，本软件只会对已变更的分区进行增量求解。
- 在添加或删除几何元素或几何约束期间，本软件会自动维护分区结构。例如，如果在两个不同分区中的几何元素之间添加一个几何约束，本软件会将这些分区合并为单个分区。当本软件长时间执行一系列顺序求解的时候，自动分区功能可以显著提高求解的性能。

全量求解

当您调用 **evaluate** 函数时，本软件将会对模型的所有分区进行全量求解。

增量求解

当应用程序向模型中添加、删除、更新几何元素或几何约束时，与操作关联的分区发生变更，其他分区不受影响。当您调用 **reEvaluate** 函数时，本软件将检测到变更的分区，对这些分区进行增量求解。增量求解能够快速处理发生变化的模型，从而提高求解的效率。

涉及的接口：**reEvaluate**，**evaluate**。

2.4.4 拖拽几何元素

拖拽是指对几何元素执行平移、旋转以及改变几何元素形状（例如改变圆和椭圆的半径）等操作的任意组合。

拖拽几何元素需要注意以下几点：

- 对大量几何元素执行拖拽可能会降低拖拽速度。
- 在拖拽过程中，除了拖拽的几何元素对象，设计中与其相关的其他几何元素也将发生一定的改变来满足它们之间的几何约束。
- 拖拽是一种多阶段求解操作，模型在多个小步骤中基于上一次计算的位置逐步被更新。
- 分区内执行的拖拽不会影响其他分区的几何元素。

涉及的接口：**beginDrag**，**dragging**，**endDrag**。

2.4.5 解析求解和数值求解

在进行解析求解或数值求解之前，本软件会先将一个大规模的模型分解为一系列可以分别求解的求解序列。求解序列中的每一个求解步骤都对应一个方程组。

解析求解

解析求解是针对特定方程组的求解方法，尤其是通过剪枝分解得到的求解步骤对应的方程组。

这类方程组往往有明确的几何意义，利用已知公式进行直接计算得到方程组的解，因此求解的效率高，稳定性好。但是不同的解析求解公式只能求解特定的方程组，没有通用性，需要针对不同的解析求解公式生成不同的求解函数。

求解序列中的很多步骤都可以使用解析求解，例如求两个圆或两条直线的交点等。

数值求解

当求解步骤对应的方程组无法进行解析求解时，需要通过数值方法进行求解，最广泛采用的方法是牛顿-拉夫森(Newton-Raphson)迭代算法。

数值求解方法的最大优点是通用性好，可以采用统一的迭代算法求解方程组。数值求解方法的缺点是速度较慢，当初值与最终解的取值差距较远时，可能导致无法收敛，稳定性较差。

求解方法比对

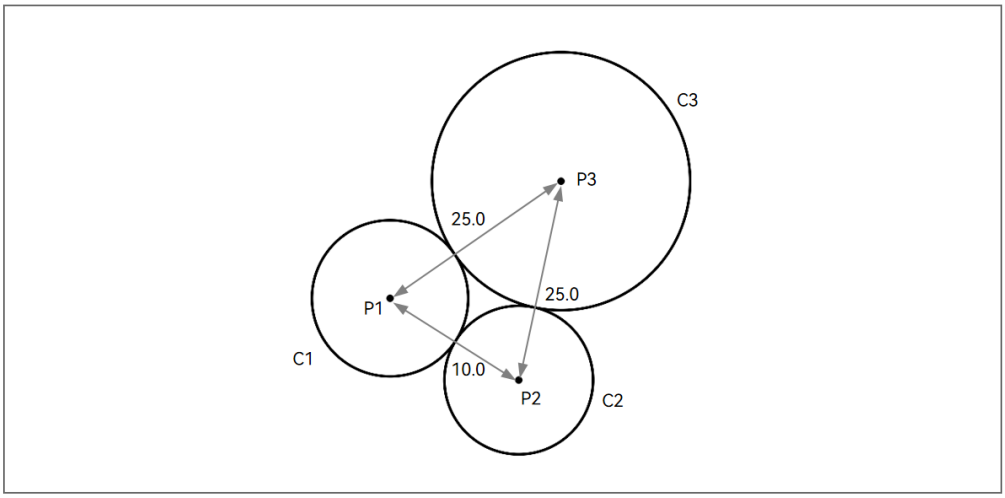
本软件将优先使用解析求解。解析求解与数值求解相比，有以下几点优势：

- 当使用解析求解时，无论几何约束的变化有多大，模型的手性都将保持不变。而数值求解可能会导致解决方案的跳跃与滞后，不过这种风险可以通过控制变化的增量最小化来克服。
- 解析求解本质上比数值求解的求解结果更加准确。
- 解析求解通常比数值求解方法的求解速度更快。

示例

以下给出一个涉及圆的示例，图 2-3 不能进行解析求解的模型中，圆 C1、圆 C2、圆 C3 之间两两相切，圆心 P1、P2、P3 之间互相关联有距离约束，该模型不能使用解析求解。

图 2-3 不能进行解析求解的模型



2.4.6 尺寸约束方案中的问题

尺寸约束方案中的问题可能导致 2D 求解器无法计算出满足所有几何约束的解决方案。在这种情况下，本软件将向应用程序传递几何元素状态信息来进行解释说明。

几何元素状态有以下三种：

- WELL_DEFINED：几何元素处于良好约束状态时，表示几何元素的所有自由度都受到了约束。例如，一个具有固定约束的点为 WELL_DEFINED 状态。
- UNDER_DEFINED：几何元素处于欠约束状态时，表示几何元素的自由度尚未完全受到约束。例如，两条直线仅添加了平行约束，此时这两条直线都是 UNDER_DEFINED 状态。
- OVER_DEFINED：几何元素处于过约束状态，表示几何元素的部分或全部自由度被重复约束。例如，一条直线添加了水平约束和垂直约束，则该直线为 OVER_DEFINED 状态。

2.4.7 求解过约束模型

在求解过约束模型时需要识别模型中的约束类型，即是否存在一致性的过约束或不一致性的过约束。一致性过约束通常可解，而不一致性过约束是指约束之间存在矛盾，模型不可解。

在许多模型中，应用程序可能会指定超过最小所需数量的尺寸约束或逻辑约束。在求解过程中，本软件要么忽略冗余的几何约束继续求解模型，要么将模型的某些部分标记为过约束状态。具体情况将在下面的小节中进行案例描述。

下面的描述中涉及一致性的逻辑约束和尺寸约束。一致性是指虽然过约束但是约束的结果一致，那么模型可解。如果尺寸约束的值是通过几何元素的位置来满足的，那么该尺寸约束是一致性的。

2.4.7.1 过约束模型

当一个尺寸约束的值不能独立于其他尺寸约束而改变，也不能找到一致性的解决方案时，本软件将会把该尺寸约束关联的模型标记为过约束状态。

2.4.7.2 过约束但一致的模型

本软件能够求解过约束但一致的模型，这里的过约束是指过度的逻辑约束。

本软件能够识别出固定几何元素之间与同一集合中的几何元素之间的几何约束。当把这些几何约束全部添加到模型中时，模型几乎都是过约束状态。对过约束但一致的情况进行分析时，应该考虑到这些识别出的几何约束。

本软件能够正确有效地管理常见案例，但对于极为复杂的案例，其管理性能可能会受到影响。因此，尽管应用程序原则上可以任意组合并使用一致性的几何约束，但是考虑到管理的性能，请合理地构造过约束但一致的模型。

以下例举了本软件能够有效解决的案例，并指出了应避免哪些情况。

相同类型的几何元素之间的多个约束

例如，两个圆之间可以有多个相切约束。

相同类型的几何元素之间的多个重合约束

例如，三个点中的任意一个都可以与其他两个点重合。

直线和点之间的多个重合约束

例如，两条有界线重合，它们的端点可以与其他线重合。

平行约束和垂直约束

任何平行约束和垂直约束的组合都将被减少到模型所需几何约束的最小集，其他多余的几何约束都将被忽略。请注意，两条直线之间的距离约束被视为平行约束，故距离约束永远不会被忽略。

等半径约束

等半径约束的存在可能会使其他约束变得冗余。

对称约束

在许多模型中，本软件能够识别出因对称约束而变得冗余的其他几何约束。例如，一条有界线由一条无界线和两个点组成，这两个点被约束为与该无界线重合。现有两条这样的有界线，如果这两条有界线被指定了对称约束，那么此时两点和无界线之间的两个重合约束是冗余的。

本软件可以有效管理几何元素关于两条对称轴对称的冗余情况。但是当几何元素关于三条或更多对称轴对称，并且这些对称约束是冗余的，本软件可能无法有效地求解该

模型。请注意，即使对称约束不是直接指定给模型的，而是通过模型中的其他几何约束推断得来的，模型中也相当于存在该对称约束。

中点约束

应用程序一般在一个点与另外两个点或两条直线之间指定中点约束。本软件可以有效管理许多常见的过约束但一致的情况，比如一个点被约束为一条有界线中点的情况。

例如，如果在一个矩形的相对边（矩形的长或宽互相平行，即两条长互为相对边，两条宽互为相对边）的中点之间构造线，本软件能够识别出这些构造出的线与矩形的边是平行的，并且忽略掉多余的平行约束。

请注意，如果一个点是多个几何元素对的中点，本软件不能有效管理此类情况。

等距离约束

等距离约束被指定作用于四个几何元素。

例如，如果一个平行四边形具有等距离约束，如果相对的边具有相同的长度，那么这些边就是平行的，此时本软件将忽略其他多余的平行约束。

一般情况下，本软件不能有效地管理多个等距离约束相互作用的复杂情况。

2.5 状态分析

2.5.1 自由度

自由度是指约束系统中能够独立变化的参数的个数，这些独立参数能够描述几何元素的绝对空间位置和形状特征 。向约束系统中添加几何约束将占用约束系统中的自由度，从而确定几何元素在约束系统中的位置与形状。

约束平衡能够反映约束系统中的自由度详情，约束平衡的计算方程如下：

约束平衡 = 几何元素的初始自由度总和 - 几何约束占用的自由度 - 刚体剩余自由度

几何元素的初始自由度总和表示在整个约束系统中几何元素没有受到任何约束时的自由度总和，下面的小节将对这些术语展开详细的说明。

如果约束平衡的值为 0，表示整个约束系统处于良好约束状态。因此，应用程序通常会以 0 值的约束平衡为目标，以确保最终用户设计的准确性。

2.5.1.1 几何元素的初始自由度

如果几何元素具有一定的自由度，则其几何元素状态为欠约束状态。如果一个几何元素的所有自由度被一系列几何约束和其他固定几何元素所限制，则其几何元素状态为良好约束状态。

二维欧几里得空间中的几何元素或刚性集合的自由度与两个平移自由度、一个旋转自由度、平移对称性、旋转对称性及内部自由度有关。两个平移自由度分别是指沿 X 轴与沿 Y 轴方向上的平移。旋转自由度不区分顺时针旋转和逆时针旋转，故只有一个。如果几何元素或刚性集合具有对称性（无论是平移对称性还是旋转对称性），它们的自由度都会因为对称性而减少。

此外，如果几何元素或刚性集合内部有额外的自由度，也会增加它们的整体自由度，通常内部自由度会影响几何元素或刚性集合的形状与大小。例如 2D 空间中的一个圆，圆的半径变化可以改变圆的大小，属于内部自由度。

根据上述信息，我们可以建立以下数学公式：

自由度 = (平移自由度 + 旋转自由度) - (平移对称性数量 + 旋转对称性数量) + 内部自由度

即：自由度 = 3 - (平移对称性数量 + 旋转对称性数量) + 内部自由度

当几何元素没有被几何约束或其他条件占用自由度时，此时几何元素的自由度为初始自由度，将用作整个约束系统初始自由度的统计。表 2-20 常用几何元素的初始自由度展示了常用几何元素在没有任何约束条件时的初始自由度。

表 2-20 常用几何元素的初始自由度

几何元素	自由度（个）	描述
点	2	点具有 0 个平移对称性，1 个旋转对称性，0 个内部自由度，根据公式得出自由度为 2。
直线	2	直线具有 1 个平移对称性，0 个旋转对称性，0 个内部自由度，根据公式得出自由度为 2。
圆	3	圆具有 0 个平移对称性，1 个旋转对称性，1 个内部自由度（圆的半径决定圆的形状大小），根据公式得出自由度为 3。
椭圆	5	椭圆具有 0 个平移对称性，0 个旋转对称性，2 个内部自由度（椭圆的长轴半径和短轴半径决定椭圆的形状大小），根据公式得出自由度为 5。
刚性集合	3	刚性集合的自由度与其中包含的几何元素数量无关。
控制点样条曲线	控制点个数 × 2	控制点样条曲线的自由度是定义该曲线的所有几何元素的整体自由度。所以，如果一条控制点样条曲线有 n 个点，那么该曲线具有 2n 个自由度（每个点的坐标分为 X 与 Y 两个自由度）。
插值点样条曲线	控制点个数 × 2	与控制点样条曲线类似。

被固定的几何元素的自由度为 0，故本软件不能移动固定的几何元素。

如果应用程序固定了一个集合，此时集合中所有几何元素的自由度都将被占用，本软件不能移动该集合以及其中的几何元素。但是，如果只固定了集合中的单个几何元素，并不能完全占用集合中所有几何元素的自由度。例如，一个刚性集合包含一条固定直线，该集合沿着直线方向仍具有单个自由度。

一个变量具有一个自由度，变量是一种特殊的几何元素。

2.5.1.2 几何约束占用的自由度

几何约束通过对几何元素的位置、形状和方向施加限制条件来占用自由度，使它们不能随意移动或变化。

大多数几何约束只能占用一个自由度，但也有一些特殊情况。比如，两条直线之间的距离约束相当于指定了平行约束和两条直线之间的距离，相当于占用了两个自由度。两个点之间的重合约束也可占用两个自由度，同心约束亦是如此。对称和重合约束分别可占用点和直线的两个自由度，圆的三个自由度，椭圆的五个自由度。

方程会占用几何元素的一个自由度。通过合理地应用几何约束条件，应用程序可以有效地控制几何元素的运动范围和形状变化，从而确保设计的可靠性和稳定性。例如，一个线性方程可以占用一个线性自由度，使得几何元素只能在其他方向上移动。

不等式不能占用任何自由度。不等式可以被分为两种类型，分别是主动与被动，主动不等式是指在某个特定解下，不等式变成了等式的情况。例如，在不等式 $x \leqslant 5$ 中，当 $x = 1$ 时，该不等式是被动的；而当 $x = 5$ 时，该不等式是主动的，因为它在此点上变成了等式。无论不等式是主动的还是被动的，它都无法占用几何元素的自由度。

2.5.1.3 刚体剩余自由度

我们将整个约束系统视为一个刚体，假设约束系统中所有几何元素之间的相对位置及尺寸都已确定，但是由于刚体整体的平移和旋转，这些几何元素仍然存在三个自由度，分别是两个平移自由度和一个旋转自由度。

刚体的剩余自由度是指，平移自由度与旋转自由度因对称性而发生变化后得到的自由度。例如，将一个较为复杂的约束系统视为一个刚体，如果其不具有平移对称性与旋转对称性，那么该刚体的剩余自由度为 3。有关对称性的详细信息，请参见 2.5.1.1 几何元素的初始自由度。

应用程序可以通过固定刚体内部的几何元素来占用刚体的剩余自由度，从而使整个约束系统达到良好约束状态。

如果约束系统中只有单个几何元素，那么刚体的剩余自由度与几何元素的自由度相等。例如，现约束系统中仅有一个点，点的初始自由度为 2，此时整个刚体的自由度也为 2；为点添加一个固定约束，此时点的自由度为 0，刚体的自由度也为 0。

🕮 说明
刚体与刚性集合不同。刚性集合是本软件为应用程序提供的一个特殊的几何元素，刚性集合中的几何元素之间相对位置与尺寸始终保持不变。但是刚体并不是一个几何元素，当我们将一个约束系统视为刚体时，目的是忽略该约束系统中各几何元素之间的相对变形或位移，并假定他们在运动中始终保持相对固定，从而简化约束平衡的计算与分析过程。

2.5.1.4 约束平衡

应用程序可以调用 **constraintBalance** 接口获取约束系统中几何元素的初始自由度总和、几何约束占用的自由度以及剩余的刚体自由度，从而计算出约束平衡的值。约束平衡的计算方程如下：

约束平衡 = 几何元素的初始自由度总和 - 几何约束占用的自由度 - 刚体剩余自由度

约束平衡的值能够体现当前约束系统的约束状态：

- 约束平衡的值 = 0，整个约束系统处于良好约束状态。
- 约束平衡的值 > 0，整个约束系统处于欠约束状态。
- 约束平衡的值 <0，整个约束系统处于过约束状态。

请注意，在以下情况中，约束平衡的结果并不准确：

- 模型中出现等效几何约束，比如两个几何约束等效，实际上只占用了一个自由度，然而在接口计算中却识别为占用了两个自由度，此时约束平衡的值会比实际值更小。建议应用程序在进行约束平衡分析之前先调用 **evaluate** 或 **reEvaluate** 接口，2D 求解器会识别并处理这些冗余的几何约束。
- 尺寸约束的数量正确，但是相互的值发生冲突，这种情况也会导致未被有效约束的自由度被计入几何约束占用的自由度中，从而造成约束平衡的值比实际值更小，建议应用程序先删除掉一些冲突的尺寸约束。
- 两个点之间的距离约束为 0，此时该距离约束占用了两个自由度，但是除 0 值以外的其他距离约束往往只占用一个自由度，在本软件的自由度计算中，无论距离约束的值为多少，都默认其占用 1 个自由度，这将导致约束平衡的值大于实际值。建议应用程序使用重合约束来替换两点之间的 0 值距离约束。

2.5.2 状态分析接口

除了上述的约束平衡分析，应用程序还能够在向约束系统添加一个几何约束之前使用 2D 求解器判断其是否会导致模型过约束，以决定是否继续添加该几何约束，帮助应用程序更好地控制约束系统的自由度。状态分析相关接口如下表所示。

接口	作用
overConstraintAnalysis	使用数值方法对当前 2D 求解器中过度约束的几何元素和几何约束进行过约束分析。如果检测到过约束的模型，2D 求解器会自动将几何元素与几何约束的状态设置为过约束状态。有关约束系统中的状态信息，请参见 13 约束系统状态信息。
check	检查尺寸约束、方向距离约束或参考线距离约束是否会导致模型过约束。参考线距离约束包括平行距离约束与垂直距离约束。
underdefinedDof	获取指定几何元素欠约束的自由度数量和相关信息。

2.6 参数配置

应用程序可以根据需求配置全局选项定制特性行为，如求解拖拽行为的方式等。

当应用程序调用 **setOption** 接口时，对选项参数的设置只会作用在调用了 **setOption** 接口的单个 2D 求解器 上，并不会影响其他 2D 求解器。**setOption** 接口涉及的选项如下表所示。

选项	描述
REPLAY_RECORD	定义 2D 求解器在全量求解、增量求解以及拖拽接口操作过程中，是否记录操作前后的数据至 replay 文件中，用于后续的操作回放。
DRAGGING_WITH_FULL_TRANSFORM	定义 2D 求解器求解拖拽的方式，决定调用 **dragging** 接口时传入的 transform 参数为全量数据还是增量数据。
DISABLE_ANALYSIS_EVALUATE	定义 2D 求解器在求解过程中是否对欠约束和过约束模型进行约束状态分析。
ENABLE_OVER_ANALYSIS_EVALUATE_ SUCCESS	定义 2D 求解器在求解成功后是否对模型进行过约束状态分析。求解失败会默认进行过约束状态分析。

以上选项默认为关闭状态。打开 REPLAY_RECORD 与 ENABLE_OVER_ANALYSIS_EVALUATE_ SUCCESS 后，应用程序在实现选项对应功能时，会降低一部分求解性能。相反，打开 DISABLE_ANALYSIS_EVALUATE 能够显著提升求解性能。有关选项取值的具体作用，请参见 **PSCSApiOption**。

2.7 文件管理

本软件为您开放了 replay 文件、操作日志与系统日志的存储接口。

- replay 文件负责记录单次求解前后的约束系统状态信息。
- 操作日志负责记录 2D 求解器中除回调函数以外的接口调用信息。
- 系统日志负责记录与运行、接口、统计相关的错误、警告、跟踪信息等。

如果您在使用本软件的过程中出现了不符合预期的结果，您可以为泊松工程师提供 replay 文件、操作日志、系统日志等材料，帮助泊松工程师复现并有效定位问题。有关上述文件的详细信息，请参见 14.10 操作日志、Replay 文件与系统日志。

2.8 事件通知处理器

2D 求解器可以通过事件通知处理器（PSCSApiNotificationHandler）向应用程序主动通知约束系统中几何元素、几何约束以及求解的状态。在使用事件通知处理器的功能之前，应用程序需要调用 **registerNotificationHandler** 接口将事件通知处理器注册到 2D 求解器中。

事件通知处理器为纯虚接口类（抽象类），应用程序需要根据自身需求重写相关方法。

接口	作用
onGeometryChanged	除应用程序主动调用接口外，如果约束系统中的几何元素发生变更，此时几何元素的变化状态为 CHANGED，2D 求解器将自动调用该接口。如果几何元素是通过应用程序调用接口直接更新的，2D 求解器并不会在求解结束时调用此接口。
onGeometryFixed	除应用程序主动调用接口外，如果约束系统中的几何元素被固定，此时几何元素的变化状态为 FIXED，2D 求解器将自动调用该接口。如果几何元素是通过应用程序调用接口直接固定的，2D 求解器并不会在求解结束时调用此接口。
onGeometryStatusChanged	在求解或状态分析结束时，除应用程序主动调用接口外，如果几何元素状态被更改，2D 求解器会自动调用该接口。有关状态分析的详细信息，请参见 2.5.2 状态分析接口。

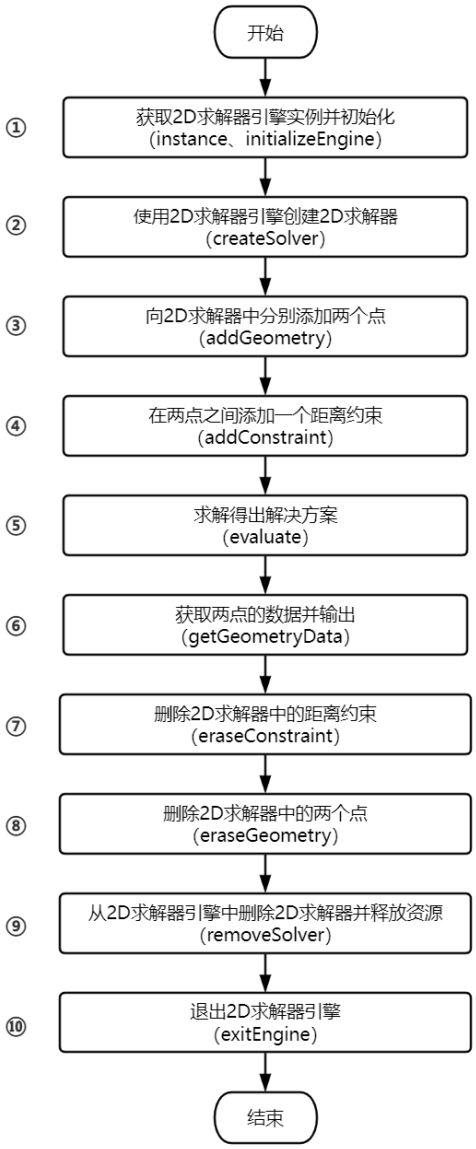
接口	作用
onGeometryRemoved	如果应用程序删除一个几何元素时，导致与其相关联的其他几何元素被删除，2D 求解器将自动调用该接口。例如，应用程序删除一条控制点样条曲线，2D 求解器会自动删除控制点，此时控制点不是应用程序调用接口直接删除的，那么 2D 求解器将在求解结束时自动调用此接口。
onConstraintSatisfiedChanged	除应用程序主动调用接口外，如果约束系统中的几何约束满足状态发生变更，2D 求解器将自动调用该接口。
onConstraintRemoved	如果应用程序删除一个几何约束时，导致 2D 求解器中与其相关联的几何元素被删除，2D 求解器将自动调用该接口。
onConstraintStatusChanged	在求解或状态分析结束时，除应用程序主动调用接口外，如果几何约束状态被更改，2D 求解器将自动调用该接口。有关状态分析的详细信息，请参见 2.5.2 状态分析接口。
onConstraintCreated	当应用程序调用自动逻辑约束或自动尺寸约束的接口时，2D 求解器自动识别约束系统中符合条件的几何约束并添加后，2D 求解器会自动调用该接口。有关自动几何约束的详细信息，请参见 12 自动逻辑约束与自动尺寸约束。
onSolvingEnd	2D 求解器在求解、拖拽或约束状态分析结束时调用该接口。有关状态分析的详细信息，请参见 2.5.2 状态分析接口。
onEvaluateProgress	在长时间求解过程中，2D 求解器会每 3 秒自动调用该接口来询问应用程序是否继续求解。由于该接口是在求解过程中被回调，如果该接口阻塞，将阻塞整个求解过程。

3 快速入门

本章以“在两个点之间添加距离约束并求解”为例，帮助您快速掌握接口调用流程，包括初始化 2D 求解器引擎、创建 2D 求解器、在 2D 求解器中添加几何元素与几何约束、求解模型、释放 2D 求解器。

“在两个点之间添加距离约束并求解”的功能实现与接口调用流程如图 3-1 功能实现与接口调用流程所示。

图 3-1 功能实现与接口调用流程



1. ① 调用 **instance** 接口获取 2D 求解器引擎实例，并调用 **initializeEngine** 接口初始化 2D 求解器引擎（仅需要初始化一次）。在本软件中，2D 求解器引擎以单例的形式存在，表示一个进程里最多只能运行一个 2D 求解器引擎。filePath 表示当前项目所在的路径。

```
PSCSApiSolver2DEngine::instance().initializeEngine(filePath);
```

2. ② 调用 **createSolver** 接口创建一个 2D 求解器，应用程序可以使用创建成功的 2D 求解器执行一系列约束求解工作。一个 2D 求解器引擎中可以创建多个 2D 求解器。

```
PSCSApiSolver2D* solver = PSCSApiSolver2DEngine::instance().createSolver();
```

3. ③ 调用 **addGeometry** 接口向 2D 求解器中分别添加两个点，inputPoint1 与 inputPoint2，两点的位置坐标分别为（0,0），（0,2）。

```
PSCSApiPoint2D inputPoint1;
inputPoint1.x = 0.0;
inputPoint1.y = 0.0;
```



```
PSCSApiId inputPoint1Id = solver->addGeometry(inputPoint1);

PSCSApiPoint2D inputPoint2;
inputPoint2.x = 0.0;
inputPoint2.y = 2.0;
PSCSApiId inputPoint2Id = solver->addGeometry(inputPoint2);
```

4. ④ 调用 **addConstraint** 接口在两点之间添加一个距离约束，距离约束的值为 1。

```
PSCSApiDimension2D inputDistance;
inputDistance.geometryIDs.append(inputPoint1Id);
inputDistance.geometryIDs.append(inputPoint2Id);
inputDistance.value = 1.0;
inputDistance.constraintData.constraintType =
PSCSApiConstraint2DType::DISTANCE;
PSCSApiId distanceId = solver->addConstraint(inputDistance);
```

5. ⑤ 调用 **evaluate** 接口执行全量求解，找到满足距离约束的几何元素相对位置，也就是解决方案。有关全量求解的详细信息，请参见 2.4.3 全量求解和增量求解。

```
solver->evaluate();
```

6. ⑥ 调用 **getGeometryData** 接口获取求解后的几何元素数据，并输出。

```
std::cout << "========== after solve ==========" << std::endl;
PSCSApiPoint2D p1;
PSCSApiPoint2D p2;
std::cout << "Point1 : {id = " << inputPoint1Id << ", ";
solver.getGeometryData(inputPoint1Id, p1);
std::cout << '[' << p1.x << ", " << p1.y << ']';
std::cout << "}" << std::endl;

std::cout << "Point2 : {id = " << inputPoint2Id << ", ";
solver.getGeometryData(inputPoint2Id, p2);
std::cout << '[' << p2.x << ", " << p2.y << ']';
std::cout << "}" << std::endl;
```

7. ⑦调用 **eraseConstraint** 接口删除 2D 求解器中的距离约束。

```
solver->eraseConstraint(distanceId);
```

8. ⑧调用 **eraseGeometry** 接口删除 2D 求解器中的两个点。

```
solver->eraseGeometry(inputPoint1Id);
solver->eraseGeometry(inputPoint2Id);
```

9. ⑨调用 **removeSolver** 接口通知 2D 求解器引擎删除使用完成的 2D 求解器，同时释放其占用的资源，例如所有的几何元素与几何约束。

```
PSCSApiSolver2DEngine::instance().removeSolver(solver);
```

10. ⑩调用 **exitEngine** 接口退出 2D 求解器引擎，一般在应用程序退出之前调用该接口（仅需要调用一次）。

```
PSCSApiSolver2DEngine::instance().exitEngine();
```

在上述示例中，可以忽略步骤⑦，此时距离约束仍然存在于约束系统中。接下来应用程序执行步骤⑧删除两个点，将会连同这两个点关联的距离约束一同删除。

说明
如果您要执行步骤⑨，此时可以跳过步骤⑦与步骤⑧，因为步骤⑨可以释放掉 2D 求解器中的所有资源，包括几何元素与几何约束。

完整示例代码

```
int main()
{
    // Must initialize the solving engine
    if (PSCSApiSolver2DEngine::instance().initializeEngine(getModuleFilePath()))
    {
        return -1;
    }

    // Create a solver
    PSCSApiSolver2D* solver = PSCSApiSolver2DEngine::instance().createSolver();
    if (nullptr == solver)
    {
        return -1;
    }

    // create two points
    PSCSApiPoint2D inputPoint1;
    inputPoint1.x = 0.0;
    inputPoint1.y = 0.0;
    PSCSApiId inputPoint1Id = solver->addGeometry(inputPoint1);

    PSCSApiPoint2D inputPoint2;
    inputPoint2.x = 0.0;
    inputPoint2.y = 2.0;
    PSCSApiId inputPoint2Id = solver->addGeometry(inputPoint2);

    // Add a distance dimension between the two points
    PSCSApiDimension2D inputDistance;
    inputDistance.geometryIDs.append(inputPoint1Id);
    inputDistance.geometryIDs.append(inputPoint2Id);
    inputDistance.value = 1.0;
    inputDistance.constraintData.constraintType =
PSCSApiConstraint2DType::DISTANCE;
    PSCSApiId distanceId = solver->addConstraint(inputDistance);

    solver->evaluate();

    std::cout << "========== after solve ==========" << std::endl;
    PSCSApiPoint2D p1;
    PSCSApiPoint2D p2;
    std::cout << "Point1 : {id = " << inputPoint1Id << ", ";
          solver.getGeometryData(inputPoint1Id, p1);
          std::cout << '[' << p1.x << ", " << p1.y << ']';
          std::cout << "}" << std::endl;
    std::cout << "Point2 : {id = " << inputPoint2Id << ", ";
          solver.getGeometryData(inputPoint2Id, p2);
          std::cout << '[' << p2.x << ", " << p2.y << ']';
          std::cout << "}" << std::endl;

    solver->eraseConstraint(distanceId);

    solver->eraseGeometry(inputPoint1Id);
    solver->eraseGeometry(inputPoint2Id);
```



```
    // Remove the solver from engine
    PSCSApiSolver2DEngine::instance().removeSolver(solver);

    // Exit engine
    PSCSApiSolver2DEngine::instance().exitEngine();
    return 0;
}
```

如果您想监控整个约束系统的几何元素状态、几何约束状态、几何元素变化状态等，您可以使用事件通知处理器来实现这一点，有关详细信息，请参见 2.8 事件通知处理器。

如果您想了解更加具体的场景及代码，请参见*《Geoshape 约束求解引擎软件 单机版 V1.1.030 接口开发文档 01》*中的典型场景示例代码。

4 几何元素

本章描述了 2D 求解器直接支持的几何元素及各自的表示形式。

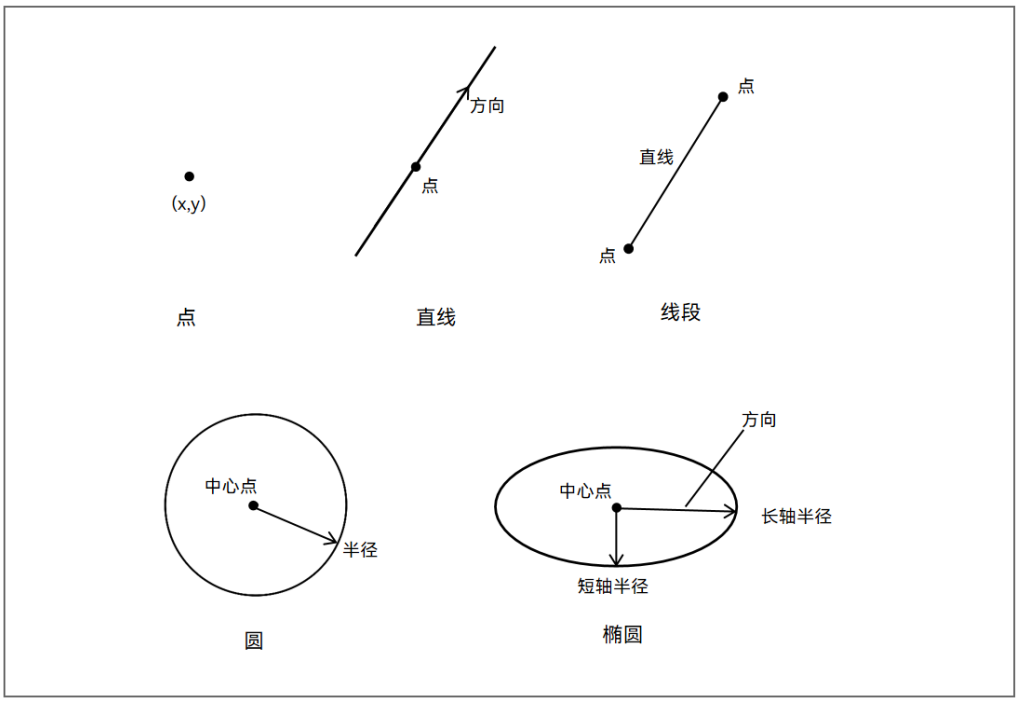
4.1 几何元素的表达

几何元素的表达通常涉及数学符号、公式和图形，用来表示几何元素的形状、大小、位置和方向等属性。本节将为您介绍一些常见的几何元素及其表达形式。

说明
下文提到的 getGeometryData 是一个同名函数，根据不同的参数实现不同的功能。

图 4-1 解析几何元素表示展示了本软件如何表示解析几何元素。

图 4-1 解析几何元素表示



- **点**
  点由其位置的 X 坐标和 Y 坐标表示。
  涉及的接口：**getGeometryData**。
- **直线**
  直线由位于直线上的任意位置点及方向向量表示。
  涉及的接口：**getGeometryData**。
- **线段**
  线段由两点表示。
  涉及的接口：**getGeometryData**。
- **圆**
  圆由中心位置点和半径表示。圆的半径必须大于线性容差。如果传入了负半径的圆，本软件会反转圆的符号，防止负半径导致的求解失败。
  涉及的接口：**getGeometryData**。
- **椭圆**
  椭圆由中心位置点、长轴半径、短轴半径以及长轴半径的方向表示。椭圆可以通过在几何约束中设置帮助点或帮助参数来指定特定位置，从长轴的逆时针方向开始计算，椭圆参数的周期为 2π。有关帮助点和帮助参数的详细信息，请参见 5.1.2.3 帮助点和帮助参数。
  涉及的接口：**getGeometryData**。
- **控制点/插值点样条曲线**
  本软件采用非均匀有理 B 样条 (NURBS) 的定义来表示控制点/插值点样条曲线。更多详细信息，请参见 7 样条曲线。
  涉及的接口：**getGeometryData**。
- **圆锥曲线**
  本软件中，圆锥曲线类型是有界圆锥截面。圆锥曲线的操作接口和行为与样条曲线相似。更多详细信息，请参见 8 圆锥曲线。
  涉及的接口：**getGeometryData**。
- **偏移曲线**
  表示椭圆、样条曲线、圆锥曲线、自定义参数曲线、自定义回调曲线、复制曲线以及偏移曲线的精确偏移量的曲线。应用程序可以使用 **getGeometryData** 接口查询偏移曲线的偏移距离、相对于原始曲线的位置（哪一侧）以及从偏移父曲线派生出的其他几何元素信息。更多详细信息，请参见 9 偏移曲线与复制曲线。
  涉及的接口：**getGeometryData**。
- **复制曲线**
  表示样条曲线、圆锥曲线、自定义参数曲线、自定义回调曲线以及偏移曲线的精确副本。本软件能够查询原始曲线映射到复制曲线的变换以及从原始曲线派生出的其他几何元素信息。更多详细信息，请参见 9 偏移曲线与复制曲线。
  涉及的接口：**getGeometryData**。
- **自定义参数曲线**
  自定义参数曲线由变量、方程、局部坐标系（LCS）的原点及方向来表示。调用 **evaluateCurve** 接口能够获取指定参数在曲线上对应的位置与导数。更多详细信息，请参见 10 自定义曲线。
  涉及的接口：**getGeometryData**。
- **自定义回调曲线**
  自定义回调曲线由变量、回调函数、局部坐标系（LCS）的原点及方向来表示。本软件能够使用应用程序定义的回调函数来查询曲线信息。更多详细信息，请参见 10 自定义曲线。
  涉及的接口：**PSCSCallbackCurveFunction** 。

4.2 几何元素的分类

对几何元素进行分类能够帮助您掌握几何元素和几何约束之间的有效组合。以下从四个不同的方面来介绍几何元素的类型。

参数化几何元素

参数化几何元素也可以叫做参数化曲线。曲线上每个位置都有对应的参数唯一确定。参数化几何元素包括控制点/插值点样条曲线、圆锥曲线、偏移曲线、复制曲线、自定义回调曲线和自定义参数曲线。

具有内部自由度的几何元素

内部自由度通常会影响几何元素的形状与大小。如果几何元素存在多余的自由度，本软件将无法唯一确定一个解决方案。为了占用多余的自由度，应用程序可以直接添加对应的尺寸约束，也可以添加其他几何约束来间接占用自由度。

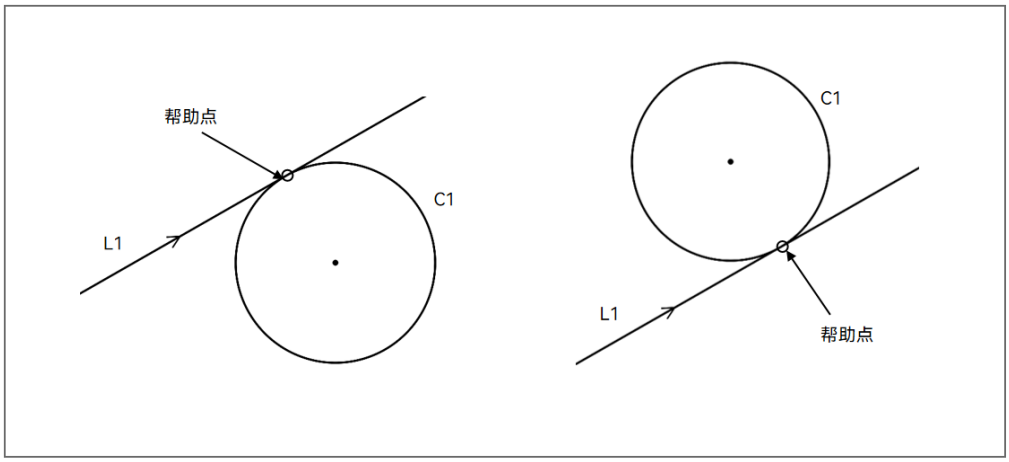
- 圆、椭圆和圆锥曲线的内部自由度：
  圆的一个内部自由度由半径决定。椭圆的两个内部自由度分别由长轴半径和短轴半径决定。请注意，椭圆的长轴半径不一定大于短轴半径。圆锥曲线的自由度是由 rho 的值来决定的。
- 参数化曲线的内部自由度：
  本软件无法直接修改参数化曲线的形状。所以您只能通过应用程序修改参数化曲线的内部自由度。
- 点和直线没有任何内部自由度。

曲线几何元素

圆、椭圆和参数化曲线都是曲线几何元素。曲线几何元素的一个重要特点是，它与其他几何元素之间关联的一个几何约束可能有多个解决方案。

现有一个圆 C1 和一条直线 L1，在 C1 和 L1 之间添加相切约束，由于为相切约束指定的帮助点位置不同，因此产生了不同的解决方案，如图 4-2 圆和直线的相切约束中所示。有关帮助点和帮助参数的详细信息，请参见 5.1.2.3 帮助点和帮助参数。

图 4-2 圆和直线的相切约束



通常，与圆和椭圆关联的几何约束一般都存在两种解决方案。与复杂的参数化曲线关联的几何约束可能有更多的解决方案。

具有方向的几何元素

- 直线：方向向量。
- 线段：线段的方向由两点的顺序决定。
- 椭圆：长轴半径的方向。
- 圆锥曲线：有界圆锥曲线的方向与其整个无界曲线的对称轴平行，方向从曲线的顶点到焦点，通常就是几何元素的开口方向。
- 线性阵列几何元素：由 2D 求解器进行推断识别。
- 自定义参数曲线：局部坐标系（LCS）绕原点的旋转角度。
- 自定义回调曲线：局部坐标系（LCS）绕原点的旋转角度。

🕮 说明
应用程序只能在具有方向的几何元素之间添加角度约束、平行约束和垂直约束。

4.3 几何元素的方向

几何元素的方向是描述几何元素在空间中所处位置及其相对位置关系的重要属性。不同的几何元素具有不同的方向属性，例如点、圆没有方向，而线段、直线、椭圆以及参数化曲线等几何元素都具有方向。

对角度约束、几何约束的约束对齐方式与约束方位这三者而言，几何元素的方向是必不可少的。因此，应用程序传递给本软件的几何元素方向必须符合规范，并且应用程序不能任意地反转它们的方向。

应用程序不需要将方向向量归一化（即不需要将向量缩放到单位长度）后再传递给 2D 求解器，2D 求解器会自动对非单位化的向量进行处理。同时应用程序需要避免传入长度小于或等于线性容差的向量。

4.4 刚性集合

刚性集合用于描述一组在刚性变换下保持相对位置不变的几何元素。换句话说，刚性集合中的所有几何元素之间的相对位置是固定的。例如，在一个机械装配设计中，一些零件可能被绑定在一起，形成一个刚性集合，在这个刚性集合中，零件与零件之间的位置是相对固定的。此时可以通过求解该刚性集合的变换（例如平移或旋转）来了解整个装配体的运动状况。

刚性集合能够使应用程序将一组具有较少几何约束的几何元素视为刚性。由于同一刚性集合中的任意几何元素之间相对固定，因此本软件不会求解同一刚性集合中两个几何元素之间的任何几何约束。创建一个包含几何元素的刚性集合有以下两种方式：

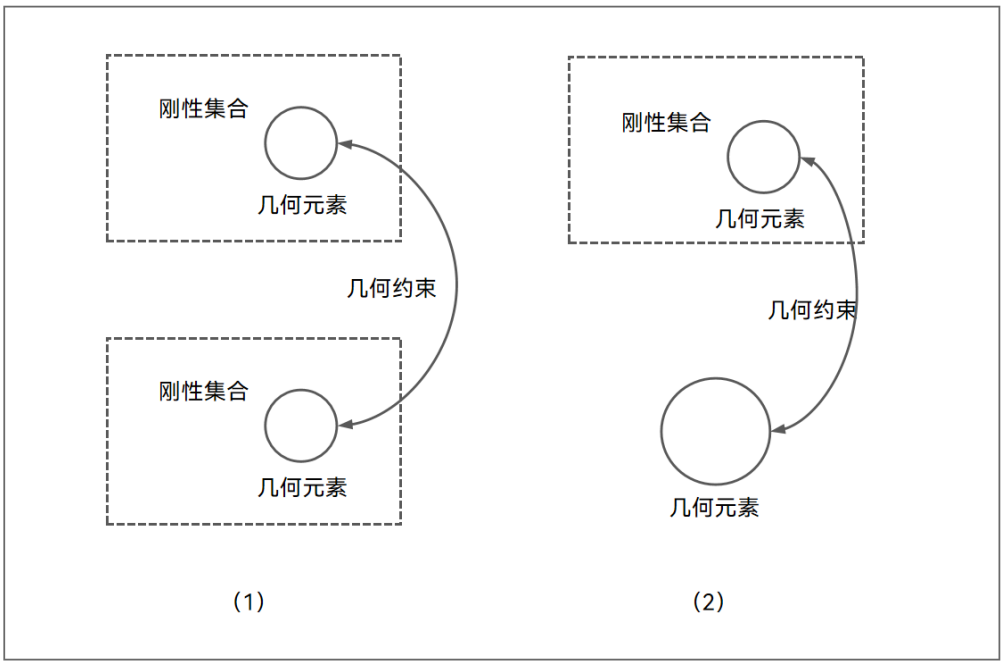
1. 调用 **addSet** 接口创建一个刚性集合时，将几何元素作为参数传入到集合中。
2. 调用 **addSet** 接口创建一个空的刚性集合，再调用 **addToSet** 接口将几何元素添加到刚性集合中。

说明

一个集合中可以容纳任意数量的几何元素，但一个几何元素只能存在于一个集合当中。同时，本软件不支持将一个集合添加到另一个集合当中。

刚性集合的位置可以通过关联几何约束来确定，图 4-3 刚性集合的位置提供了两种方式。

图 4-3 刚性集合的位置



- 图（1）：在一个刚性集合中的几何元素与其他刚性集合中的几何元素之间添加几何约束。

- 图（2）：在一个刚性集中的几何元素与不在任何集合中的几何元素之间添加几何约束。

涉及的接口：**addSet**，**addGeometry**，**addToSet**，**removeFromSet**。

4.5 可伸缩集合

可伸缩集合是一组形状比例能够发生等比变化的几何元素的集合。

可伸缩集合具有一个内部自由度与三个外部自由度。三个外部自由度分别由旋转、X 方向上的平移及 Y 方向上的平移决定，一个内部自由度由缩放因子决定，应用程序能够根据该缩放因子均匀地缩放可伸缩集合。关于自由度的相关详情，请参见 2.5.1 自由度。

应用程序可以通过以下三种方式来占用可伸缩集合的内部自由度：

- 在可伸缩集合中的几何元素之间添加一个距离约束。
- 如果可伸缩集合中包含圆，可以在圆上添加一个半径约束。
- 如果可伸缩集合中包含椭圆，可以在椭圆上添加一个长轴半径约束或短轴半径约束。

应用程序为可伸缩集合添加上述约束的任意一种后，可以通过改变几何约束的值来控制缩放因子的值。请注意，当一个可伸缩集合包含上述几何约束中的两种及以上时，此时模型处于过约束但一致的状态，改变任意几何约束的值都将导致约束冲突，无法得出解决方案。除上述几何约束外，2D 求解器进行求解时将自动忽略其他几何约束。

可伸缩集合的原点（集合的局部坐标系原点）将作为缩放中心，缩放操作均围绕缩放中心进行。应用程序能够使用 **updateSetCenter** 接口为集合更新缩放中心。

可伸缩集合的创建方式与刚性集合类似，分别是创建集合时，将几何元素作为参数传入一并创建，或先创建空集合，再向其中添加几何元素。在处理可伸缩集合时，必须根据变换矩阵和缩放因子来更新集合。

涉及的接口：**updateSetCenter**，**getSetCenter**。

4.6 单向可伸缩集合

单向可伸缩集合是一组形状比例只在某个方向上发生改变的几何元素的集合。

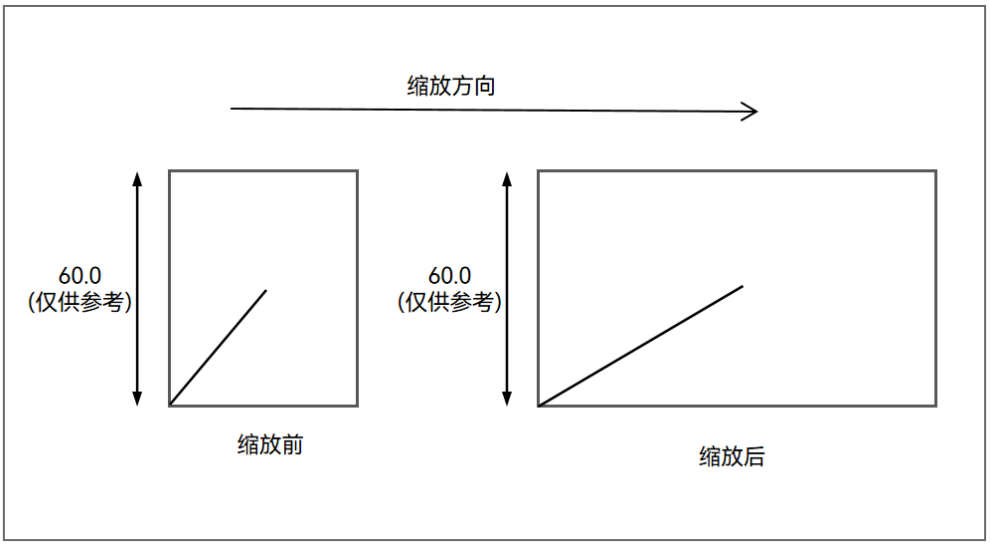
单向可伸缩集合具有一个内部自由度与三个外部自由度。三个外部自由度分别由旋转、X 方向上的平移及 Y 方向上的平移决定，一个内部自由度由缩放因子决定，该内部自由度允许集合中几何元素的长宽比发生变化。现有一个添加到单向可伸缩集合中的矩形，如果其宽的方向与集合的缩放方向平行，那么宽度就可以自由改变，同时高

度保持不变。单向可伸缩集合仅支持点、直线与线段。关于自由度的相关详情，请参见 2.5.1 自由度。

图 4-4 单向可伸缩集合给出了单向可伸缩集合的示例。现有一个可伸缩集合包含一个矩形与一条线段，设置缩放方向，使其与矩形的某一条边平行，此时对集合进行缩放，矩形的长或宽的大小会根据缩放因子改变，但是方向不变。由于线段的方向与缩放方向不平行，矩形内部线段的长度与角度均发生变化。

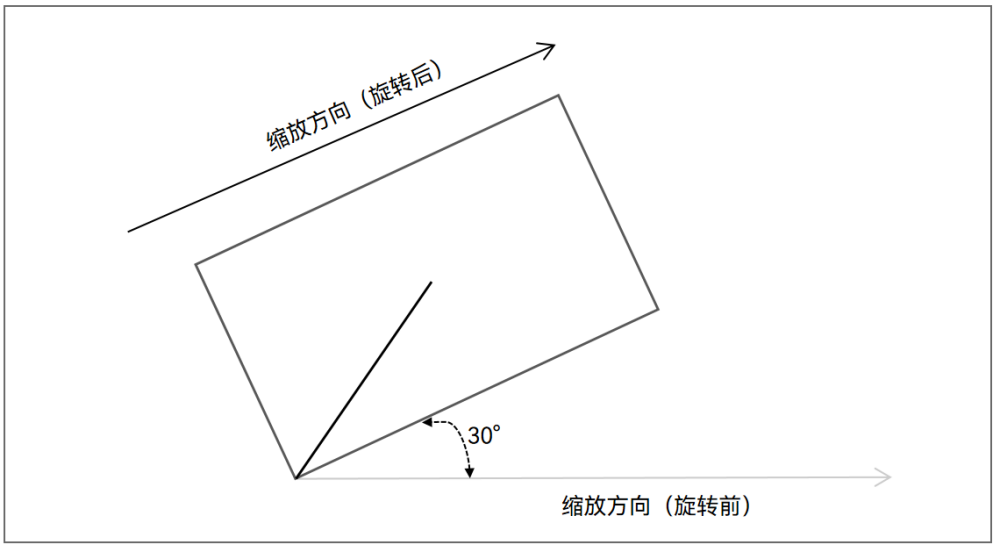
请注意，与缩放方向垂直的距离约束不会因可伸缩集合的缩放而改变，图 4-4 单向可伸缩集合中的测量值 60.0 仅供参考。如果在缩放方向上或在垂直于缩放方向上添加任何额外的几何约束，都会导致模型过约束。

图 4-4 单向可伸缩集合



如果单向可伸缩集合在求解过程中被旋转，那么缩放方向也将随之旋转。例如，如果一个集合的缩放方向为 X 轴正方向，然而该集合在求解期间被旋转了 90°，那么该集合的缩放方向变为 Y 轴正方向。图 4-5 旋转缩放方向展示了集合旋转 30°后缩放方向的变化。

图 4-5 旋转缩放方向



应用程序可以使用 **updateSetDirection** 接口设置缩放方向，通过 **getSetDirection** 接口查询缩放方向。

涉及的接口：**updateSetCenter**，**getSetCenter**，**updateSetScale**，**getSetScale**，**updateSetDirection**，**getSetDirection**。

4.7 双向可伸缩集合

双向可伸缩集合是一组形状比例能够在两个互相垂直的方向上发生改变的几何元素的集合。

执行缩放操作之前，应用程序需要指定一个缩放方向 D1，2D 求解器会从 D1 的逆时针方向推断出与 D1 垂直的缩放方向 D2。双向可伸缩集合的两个缩放方向对应两个内部自由度。关于自由度的相关详情，请参见 2.5.1 自由度。

双向可伸缩集合的创建方式与单向可伸缩集合类似，仅支持点、直线与线段。并通过 **updateSetDirection** 函数指定第一个缩放方向。

涉及的接口：**updateSetDirection**。

4.8 阵列几何元素

当一组相同的几何元素按照一定的规则线性排列或旋转排列时，我们将这组几何元素的整体视为阵列几何元素。

要想实现几何元素的规则排列，需要利用阵列约束。阵列约束可以约束一组几何元素规则地排列在线性或环形阵列几何元素中，有关详细信息，请参见 5.2.12 阵列约束。

阵列约束将单个几何元素的相对自由度替换为整个阵列几何元素的内部自由度。单个参考元素的阵列几何元素具有一个自由度，即线性距离或角度距离；两个参考元素的阵列几何元素具有两个方向上的线性距离自由度。

阵列几何元素包括参考元素与阵列值。参考元素定义了阵列几何元素的中心或方向。更多详细信息，请参见 4.8.1 参考元素和阵列值。

涉及的接口：**addGeometry**，**addConstraint**。

4.8.1 参考元素和阵列值

参考元素与阵列值的数量不同，阵列几何元素的类型也不同。阵列值可以表示线性距离或角度距离，阵列值的正负用于指定排列方向。

单个参考元素的阵列几何元素

参考元素可以为点、圆、线段或直线。

当参考元素为点或圆时，种子几何元素会按照一定的阵列值（角度距离）偏移，从而排列为环形阵列几何元素。阵列值为正，被阵列约束的几何元素按照逆时针方向排列；阵列值为负，被阵列约束的几何元素按照顺时针方向排列。

当参考元素为线段或直线时，种子几何元素会按照一定的阵列值（线性距离）偏移，从而排列为线性阵列几何元素。阵列值为正，被阵列约束的几何元素按照线段或直线的正向方向排列；阵列值为负，被阵列约束的几何元素按照线段或直线的反向方向排列。

两个参考元素的阵列几何元素

参考元素可以为线段、直线或两者的组合。

当参考元素符合规定时，种子几何元素会按照两个阵列值（线性距离）在两个方向上进行偏移，从而排列为线性阵列几何元素。此时的两个阵列值分别表示平行于两个参考元素的方向上的距离。阵列值为正，阵列几何元素按照线段或直线的正向方向排列；阵列值为负，阵列几何元素按照线段或直线的反向方向排列。

涉及的接口：**addConstraint**。

5 几何约束

本章主要向您介绍 2D 求解器支持的所有尺寸约束和逻辑约束，以及它们的使用方法。

尺寸约束（例如距离、角度、半径或曲线长度等）具有相关的值，而逻辑约束（例如平行、垂直等）没有关联的值。如果模型中具有过多的逻辑约束，但这些逻辑约束是一致性的，那么 2D 求解器仍然能够成功求解模型。注意，2D 求解器不能求解具有冗余尺寸约束的模型。

在下面的小节中，由于控制点/插值点样条曲线、自定义参数曲线与自定义回调曲线能够组合的几何约束类似，因此将他们统称为参数化曲线，只有在他们涉及的几何约束类型不同时，才分开描述。

5.1 尺寸约束

使用逻辑约束来替代等效的尺寸约束，有助于 2D 求解器找到更明确、更稳定的解决方案。

当使用数值 0 来表示一个尺寸约束时，2D 求解器会假设该尺寸约束的值之后会发生改变，从而找到一系列符合假设的解决方案。但是如果在实际应用中需要表示一个固定的、不可更改的几何关系，那么在本软件中使用 0 值这种不明确的解决方案就会出现问题。

逻辑约束能够更明确地表示这种固定关系，减少求解过程中的歧义。例如，两点之间距离约束的值为 0，此时可以使用等效的逻辑约束（重合约束）来表示这两个点距离为 0。除此之外，逻辑约束有助于避免数值误差，因为逻辑约束是基于几何元素的固有属性，而不是简单的数值近似。

涉及的接口：**addConstraint**，**addConstraint**，**getConstraintType**。

5.1.1 尺寸约束的值

每个尺寸约束的值可以通过应用程序指定的一个数值（如 0、1、2）来确定。也可以通过关联其他尺寸约束的值来确定。现需确定一个尺寸约束的值，此时将该尺寸约束

与其他尺寸约束视为变量添加到方程或不等式中，求解方程或不等式就相当于求解尺寸约束的值，更多详细信息，请参见 2.4 求解和拖拽。

涉及的接口：**getDimensionValue**，**updateDimensionValue**。

5.1.2 距离约束

所有类型的几何元素对之间都可以添加距离约束。

添加距离约束需要注意以下几点（线包括直线与线段）：

- 线与线：两条线之间的距离约束意味着它们相互平行且相隔距离等于距离约束的值。2D 求解器将视这两条线之间有一个平行约束。
- 点与点或点与线：指定点与几何元素之间的最小距离。
- 参数化几何元素与任意几何元素：请参见 6 参数化几何元素。
- 圆与任意几何元素或椭圆与任意几何元素：与圆或椭圆关联的距离约束分为默认的距离约束和带有帮助点的距离约束。下面的章节将作更加详细的介绍。

有关距离约束的符号，请参见 5.1.2.5 有符号的距离约束。

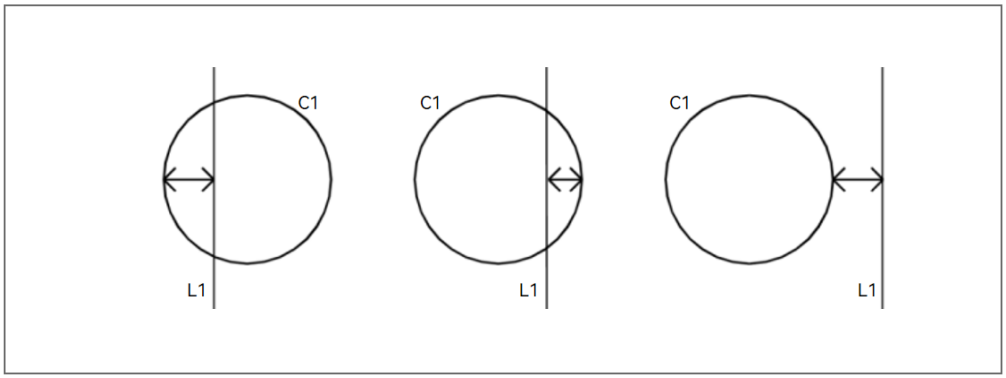
5.1.2.1 默认的距离约束行为

在应用程序没有指定距离约束的帮助点或帮助参数时，有以下两种情况：

- 如果几何元素不相交，那么距离约束的值表示它们之间的最小距离。
- 如果几何元素相交，那么距离约束的值表示可以将两个几何元素分开至不产生重叠的最小距离。

例如一个圆 C1 与一条直线 L1，如果两者相交，那么最小距离约束的值一定小于该圆的直径。本软件支持图 5-1 最小距离示例所示的三种模型中（圆和直线之间）的最小距离：

图 5-1 最小距离示例



设直线到圆心的垂直距离为 d，圆的半径为 r。

- 如果 d > r，表示直线与圆不相交，此时的最小距离约束的值是 d - r，即直线到圆的最小距离。
- 如果 d = r，表示直线与圆相切，此时的最小距离约束的值为 0，即直线与圆只有一个交点。
- 如果 d < r，表示直线与圆相交，此时的最小距离约束的值是 r - d，即圆与直线分开至不产生重叠的最小距离。

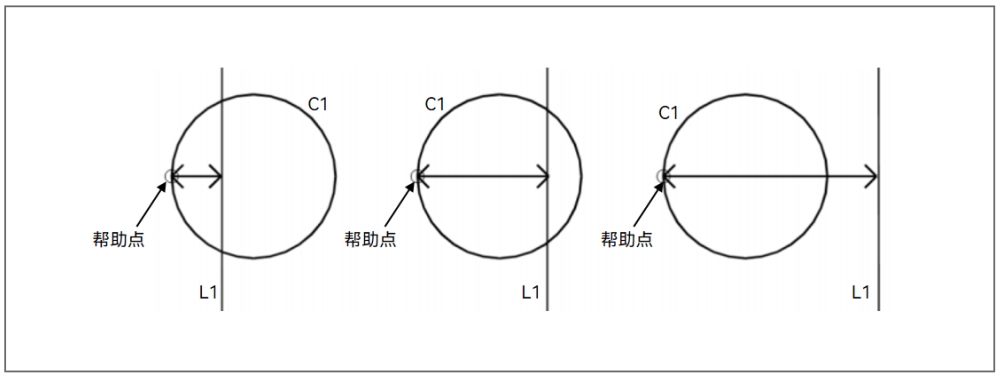
5.1.2.2 带有帮助点的距离约束

本软件提供了帮助点来指定几何约束作用在圆与椭圆的“哪一侧”。只有在圆和椭圆这两类几何元素与其他几何元素之间添加几何约束时，应用程序才能够未为其设置帮助点。所以，虽然帮助点用于指示几何约束在几何元素上的期望生效位置，但是它属于几何约束的属性。

帮助点并不是几何元素实体，它是与几何约束和几何元素相关联的一个位置矢量。圆与直线的距离约束可以有一个帮助点，圆与圆的距离约束可以有两个帮助点。在下一节中将详细介绍帮助点的使用方法。

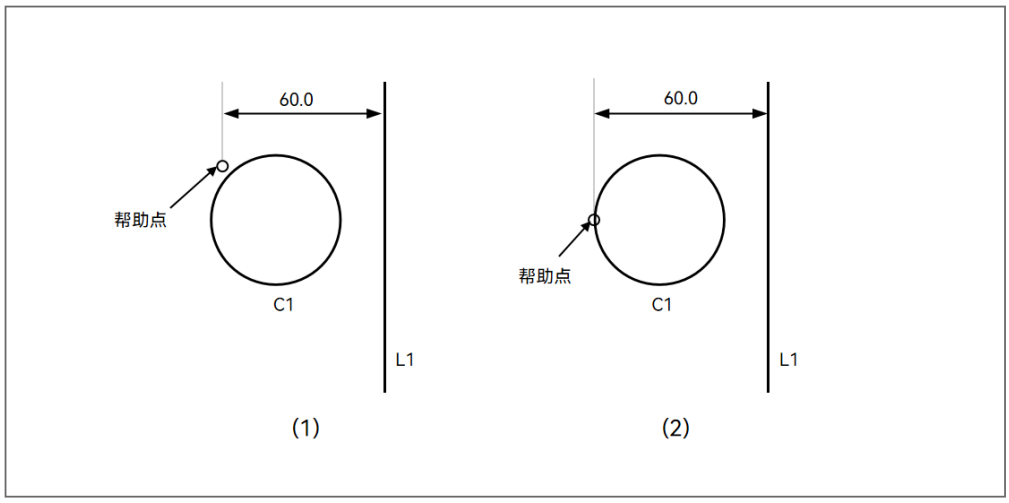
图 5-2 带有帮助点的距离约束展示了在圆和直线之间的距离约束使用帮助点的三种情况。小圆圈表示帮助点的位置。

图 5-2 带有帮助点的距离约束



在添加几何约束时，应用程序只需要提供帮助点的近似位置。在求解时，本软件会自动将帮助点移动到几何元素的准确位置上。图 5-3 圆和直线之间的距离约束与帮助点中，图（1）显示了一个圆 C1 和一条直线 L1 之间的距离约束以及帮助点的初始近似位置，图（2）是 2D 求解器在求解过程中自动移动帮助点得到的结果。

图 5-3 圆和直线之间的距离约束与帮助点



以圆或椭圆为约束对象的时候，应用程序可以在距离约束、相切约束、法线约束和重合约束中指定帮助点。如果在距离约束中没有指定帮助点，本软件将以默认的距离约束行为来进行求解，也就是求几何元素之间的最小距离。如果在逻辑约束中没有指定帮助点，本软件将使用最接近初始相对位置的解决方案。

5.1.2.3 帮助点和帮助参数

本软件中，与圆或椭圆相关的几何约束支持帮助点，与椭圆或各参数化曲线相关的几何约束支持帮助参数，关于对帮助点和帮助参数的支持详情，请参考表 5-1 支持帮助点与帮助参数的几何元素。虽然帮助点与帮助参数用于表示几何约束在几何元素上的期望生效位置，但是它们均属于几何约束的属性。

表 5-1 支持帮助点与帮助参数的几何元素

几何元素	帮助点	帮助参数
点	×	×
直线	×	×
线段	×	×
圆	√	×
椭圆	√	√
控制点样条曲线	×	√
插值点样条曲线	×	√
圆锥曲线	×	√
偏移曲线	×	√
复制曲线	×	√
自定义参数曲线	×	√

几何元素	帮助点	帮助参数
自定义回调曲线	×	√

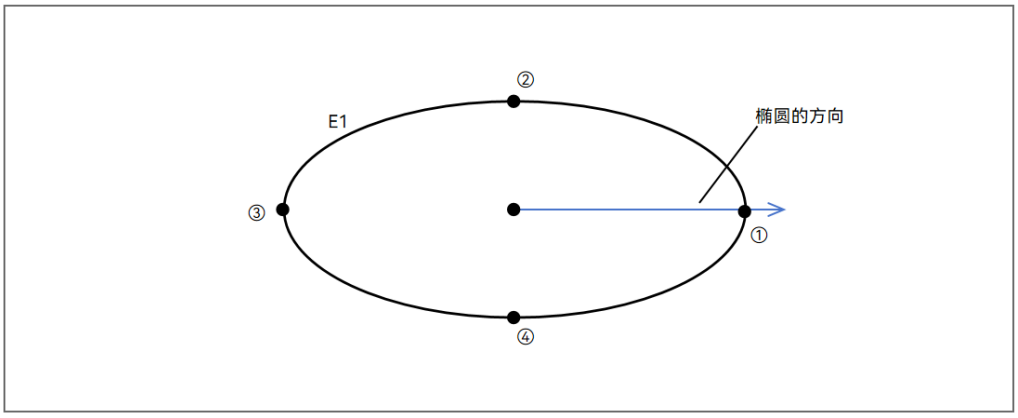
在添加与椭圆或参数化曲线关联的几何约束时，在几何约束上设置帮助参数，能够指示几何约束在椭圆或参数化曲线上的期望生效位置。如果需要精确定义与控制几何约束的行为，需要使用 **setHelpParameterFixedState** 来固定帮助参数。

根据不同的几何元素类型，帮助参数的取值范围也不相同，如作用于圆锥曲线的帮助参数取值范围是[0,1]，作用于椭圆的帮助参数取值范围是[0,2π]等等。

- 对于参数化曲线来说，如果应用程序指定的值超出了取值范围，2D 求解器将选取指定值最接近的取值边界进行求解，如指定与圆锥曲线关联的几何约束的帮助参数为 1.2 或 1.7，2D 求解器会自动调整帮助参数的值为 1。
- 对于椭圆来说，如果应用程序指定的值超出了取值范围，2D 求解器将根据周期自动调整帮助参数的值，使其位于[0,2π]的区间，如指定与椭圆关联的几何约束的帮助参数为 5π，2D 求解器会自动调整帮助参数的值为 π。

以椭圆为例，帮助参数指示的几何约束生效位置如图 5-4 椭圆的帮助参数 所示，图中蓝色箭头表示椭圆 E1 的方向，应用程序可以使用帮助参数来表示图中①、②、③、④分别对应的位置。

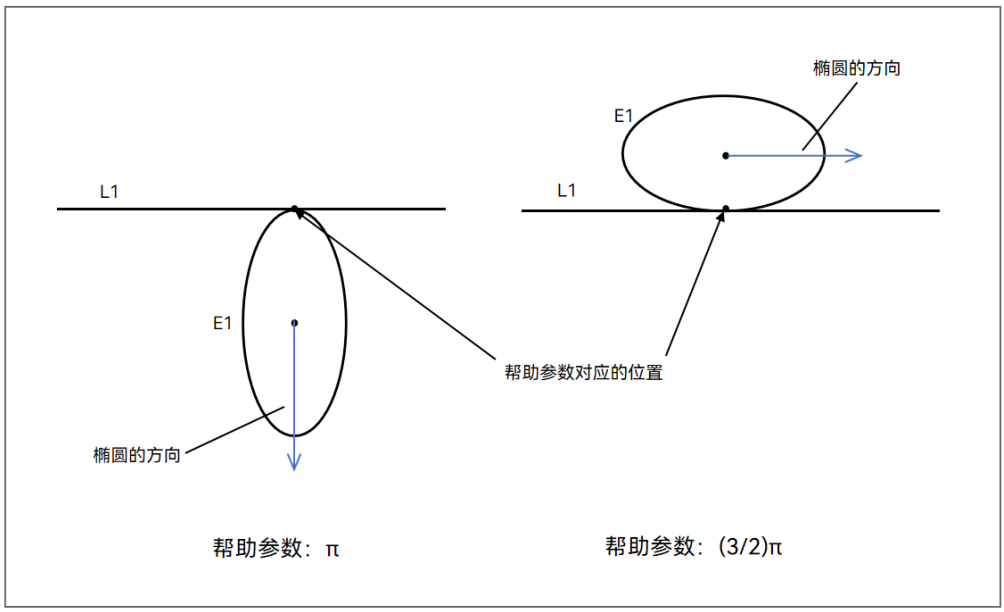
图 5-4 椭圆的帮助参数



- ①：0。
- ②：1.57，近似(1/2)π。
- ③：3.14，近似 π。
- ④：4.74，近似(3/2)π。

在椭圆与直线的距离约束中（距离约束的值为 0），指定距离约束的帮助参数相当于指定了距离约束在椭圆上的期望生效位置。如图 5-5 距离约束的帮助参数所示，图中椭圆 E1 与直线 L1 距离约束的值为 0，由于帮助参数对应的具体位置不同，故得到以下两种解决方案。图中蓝色箭头表示 E1 的方向。

图 5-5 距离约束的帮助参数



说明
如果在椭圆上添加了几何约束，该几何约束不能同时支持帮助点与帮助参数。

涉及的接口：**setHelpPoint**，**getHelpPoint**，**setHelpParameter**，**getHelpParameter**。

5.1.2.4 有方向的距离约束

您可以在除直线和线段以外的所有几何元素之间添加有方向的距离约束，用于约束指定方向上的距离。

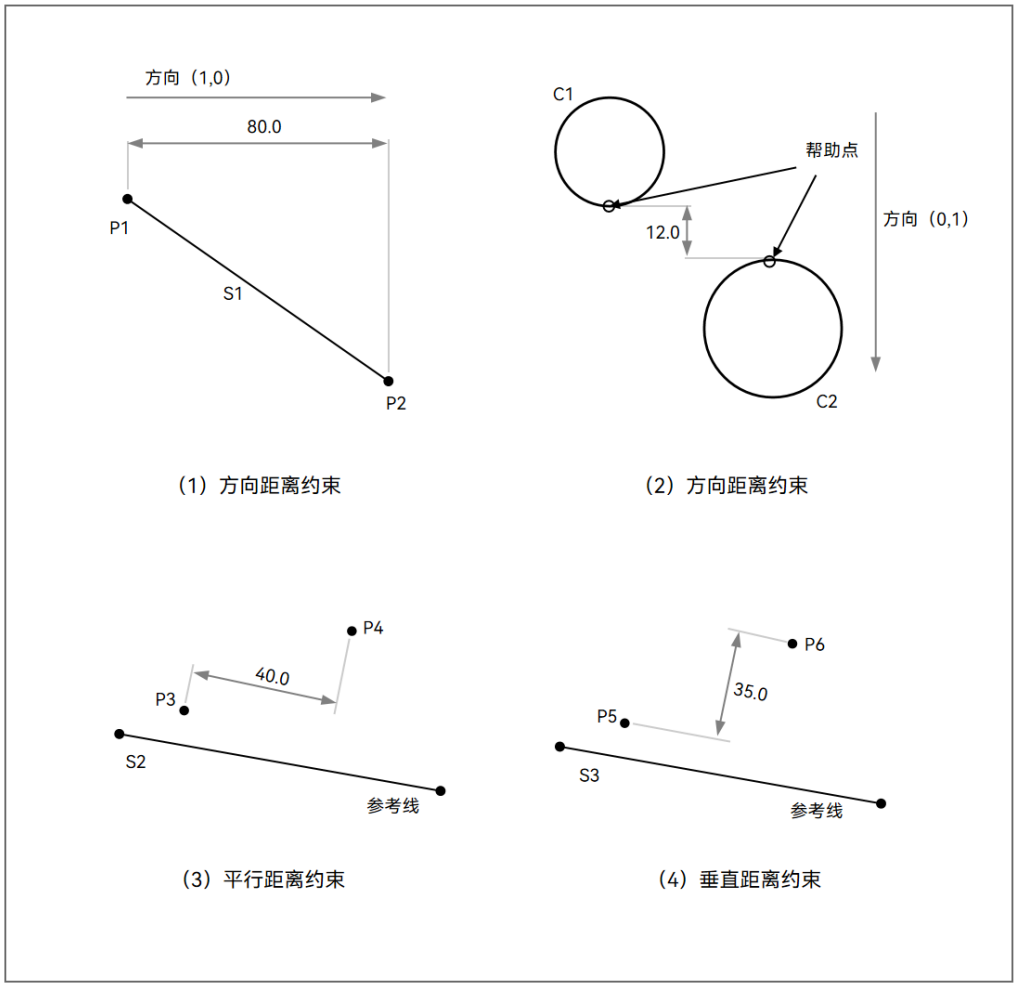
有方向的距离约束有以下两种：

- **方向距离约束**
  通过分别设置水平方向和竖直方向上的向量来确定最终方向。比如在两个点之间添加一个方向距离约束，点 P1 位置坐标为(0,0)，点 P2 位置坐标为(2,0)，固定 P1，指定方向距离约束的值为 1，方向向量为(1,0)，求解后，P1 位置坐标为(0,0)，P2 位置坐标为(1,0)。如果应用程序需要限制两个几何元素在竖直方向上的距离，可以将方向向量设置为(0,1)，如图 5-6 有方向的距离约束中（1）与（2）所示。
- **参考线距离约束**
  - 平行距离约束：指定一条直线或线段作为参考线，2D 求解器会求解两个几何元素平行于参考线方向上的距离。
  - 垂直距离约束：指定一条直线或线段作为参考线，2D 求解器会求解两个几何元素垂直于参考线方向上的距离。

  当参考线的角度发生变化，距离约束的方向也会随之改变。

图 5-6 有方向的距离约束中分别给出两点之间以及两个圆之间的有方向的距离约束示例，图中距离约束的值仅作参考。

图 5-6 有方向的距离约束



- 图（1）：点 P1 和点 P2 之间在水平方向上的方向距离约束。
- 图（2）：圆 C1 和圆 C2 在竖直方向上带有两个帮助点的方向距离约束。
- 图（3）：点 P3 和点 P4 之间平行于线段 S2 方向上的平行距离约束。
- 图（4）：点 P5 和点 P6 之间垂直于线段 S3 方向上的垂直距离约束。

说明
请注意上述水平与平行的区别，以及竖直与垂直的区别。

涉及的接口：**addConstraint**。

5.1.2.5 有符号的距离约束

对于常规的距离约束和半径约束，如果它们取值的正负对解决方案不会造成影响，2D 求解器在计算解决方案之前会取尺寸约束的绝对值，此时尺寸约束的正负性对解决方案来说没有任何区别。

如果距离约束的方向与参考方向平行，那么其值的正负性会影响解决方案，这种情况下，值的符号与两个几何元素在参考方向上的相对顺序有关。垂直于参考方向的距离约束的正负性无效，不会影响解决方案。

对于关联在有方向的几何元素上的距离约束，应用程序可以指定一个几何元素位于另一个几何元素的“哪一侧”。例如，点与直线之间的距离约束有两种解决方案，该点分别位于该直线两侧的两根平行直线上。本软件视这两种解决方案具有不同的约束方位，默认给出最接近初始相对位置的解决方案。

对于有符号的尺寸约束，本软件在计算解决方案时会考虑尺寸约束值的符号。对于有符号的距离约束，本软件可以在一个从正到负的连续值区间内进行求解。

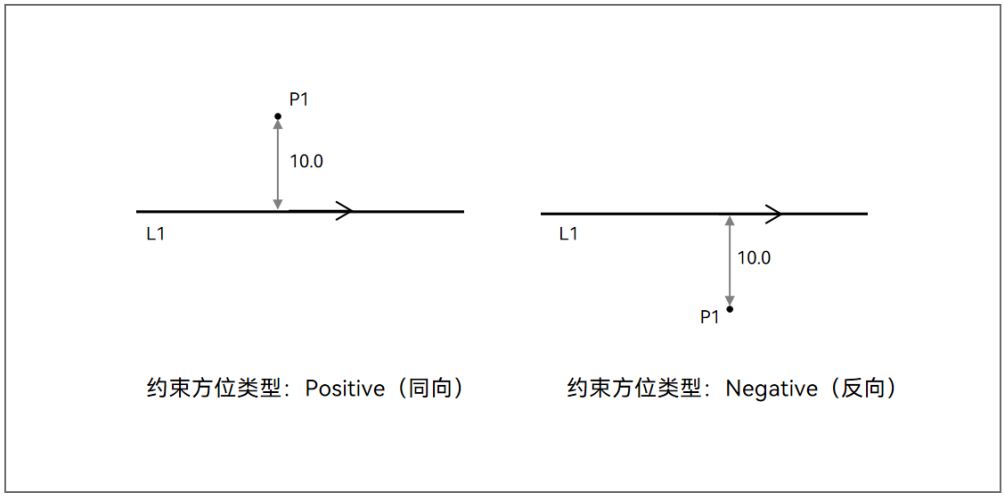
5.1.2.6 反转距离约束

反转距离约束是更新约束方位最简单的示例。

反转距离约束有如下三个示例（线包括直线与线段）：

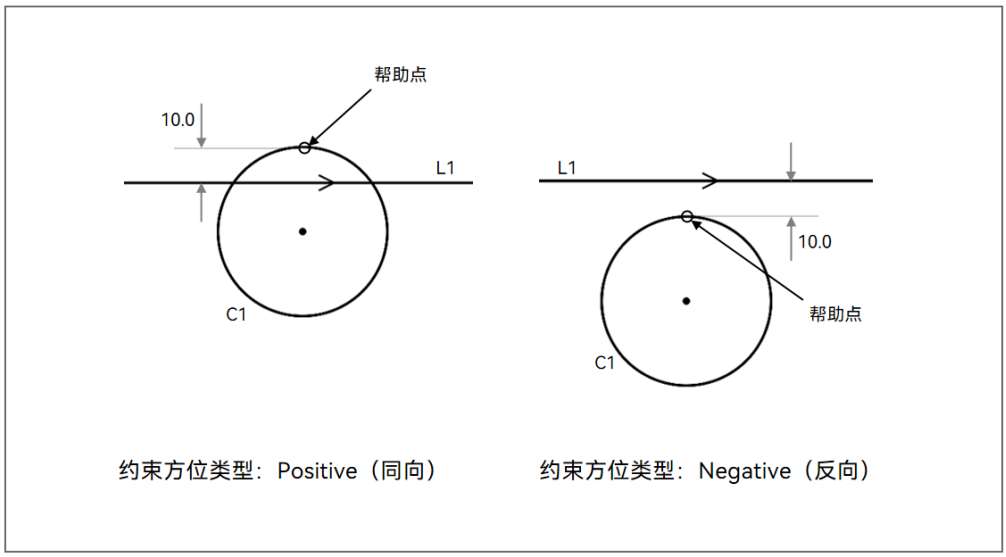
- 在点与线之间添加距离约束，反转距离约束会将点移动至线的另一侧。例如，在点 P1 与直线 L1 之间添加一个距离约束，改变距离约束的约束方位将得到图 5-7 改变点与线之间的距离约束的约束方位中所示的两种解决方案。

图 5-7 改变点与线之间的距离约束的约束方位



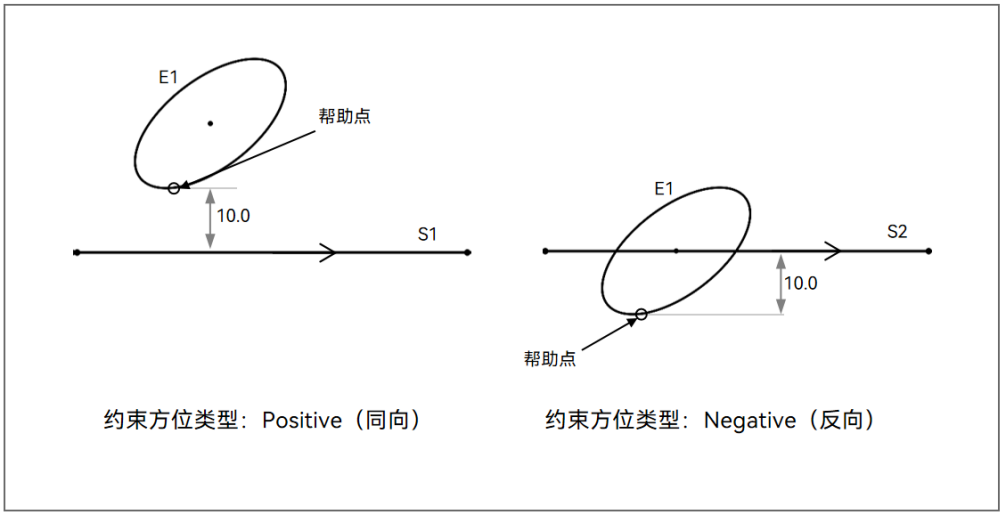
- 在线与圆之间添加距离约束，反转距离约束不会影响帮助点在圆上对应的具体位置，只会将帮助点移动到线的另一侧，例如，在直线 L1 与圆 C1 之间添加一个距离约束，改变距离约束的约束方位将得到图 5-8 改变圆与线之间的距离约束的约束方位所示的两种解决方案。

图 5-8 改变圆与线之间的距离约束的约束方位



- 在线与圆、椭圆或参数化几何元素之间添加距离约束，反转距离约束不会影响帮助点或帮助参数在几何元素上对应的具体位置，只会将帮助点或帮助参数移动到线的另一侧。例如，在椭圆 E1 与线段 S2 之间添加一个距离约束，改变距离约束的约束方位将得到图 5-9 改变椭圆与线之间的距离约束的约束方位所示的两种解决方案。

图 5-9 改变椭圆与线之间的距离约束的约束方位



如果您想更直观地了解距离约束，您可以调用接口执行以下步骤：

1. 创建两个圆和一条直线,两个圆均位于直线的同一侧。
2. 在两个圆与直线之间分别添加一个距离约束。
3. 更新两个距离约束的约束方位，分别是同向和反向。
4. 将距离约束的值设置为零，观察此时两个圆与直线的相对位置。
5. 将距离约束的值从 0 逐渐增大到 10，观察此时两个圆的移动方向。

可以观察到两个圆朝着相反的方向移动。

5.1.3 半径约束

半径约束仅针对圆和椭圆。

对于一个圆来说，半径约束指定了圆的半径。圆的半径必须大于线性容差。

对于一个椭圆来说，长轴半径约束指定了椭圆的长轴半径，短轴半径约束指定了椭圆的短轴半径。长轴半径的值不一定大于短轴半径。两条半径都必须大于当前的线性容差。

5.1.4 角度约束

应用程序只能在具有方向的几何元素之间添加角度约束。

角度约束一般作用于两个几何元素方向之间的扇区弧度值。从第一个几何元素方向逆时针旋转至第二个几何元素方向的弧度值为正，相反，从第一个几何元素方向顺时针旋转至第二个几何元素方向的弧度值为负。此外，应用程序可以为角度约束设置任意值，本软件会根据 2π 周期来进行求解，从而得到合理的解决方案。

图 5-10 两条直线之间的角度约束中，直线 L1 与直线 L2 之间的角度约束作用于 L1 方向逆时针旋转至 L2 方向的扇形弧度值（ang），此时 ang 为正值。

图 5-10 两条直线之间的角度约束

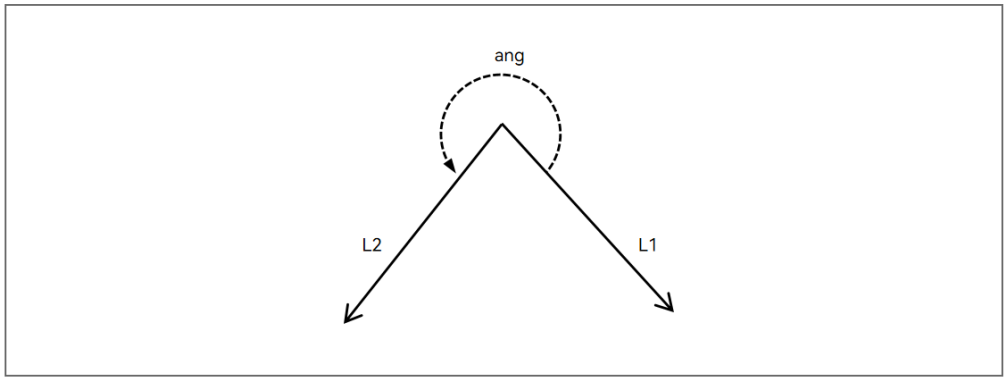
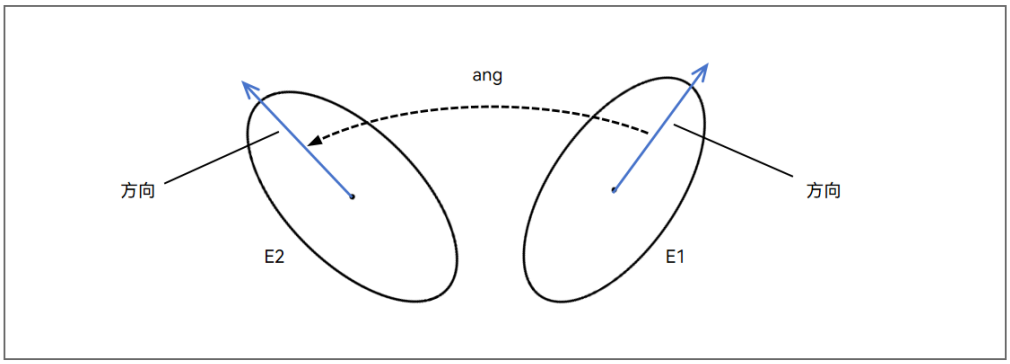


图 5-11 两个椭圆之间的角度约束中，椭圆 E1 与椭圆 E2 之间的角度约束作用于 E1 方向逆时针旋转至 E2 方向的扇形弧度值（ang），此时 ang 为正值，图中蓝色箭头表示椭圆的方向。

图 5-11 两个椭圆之间的角度约束



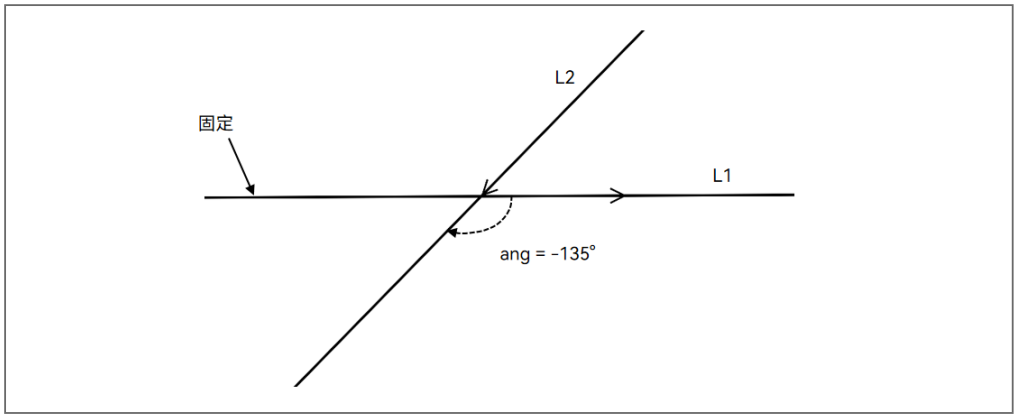
本软件会参考不同的约束对齐方式与约束方位，将您输入的值 angle 转换为用于求解的值 ang，从而影响角度约束的解决方案。

1. 当约束方位为 Negative 时，ang = -angle；为 Positive 时，ang = angle。
2. 当约束对齐方式为 Negative 时，ang = ang + π；为 Positive 时，ang 不变。

例如，现有两条相交的直线 L1 与 L2，固定 L1，在 L1 与 L2 之间添加角度约束，更新角度约束的值 ang 为-135°。根据约束对齐方式与约束方位的两两组合，可以得到以下四种解决方案。

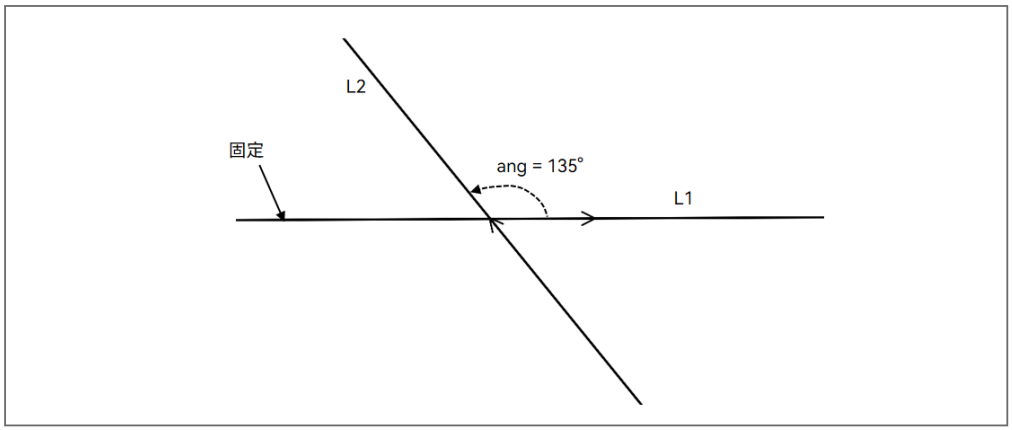
- 约束方位：Positive
  约束对齐方式：Positive
  ang = -135°

图 5-12 同向对齐与同向方位



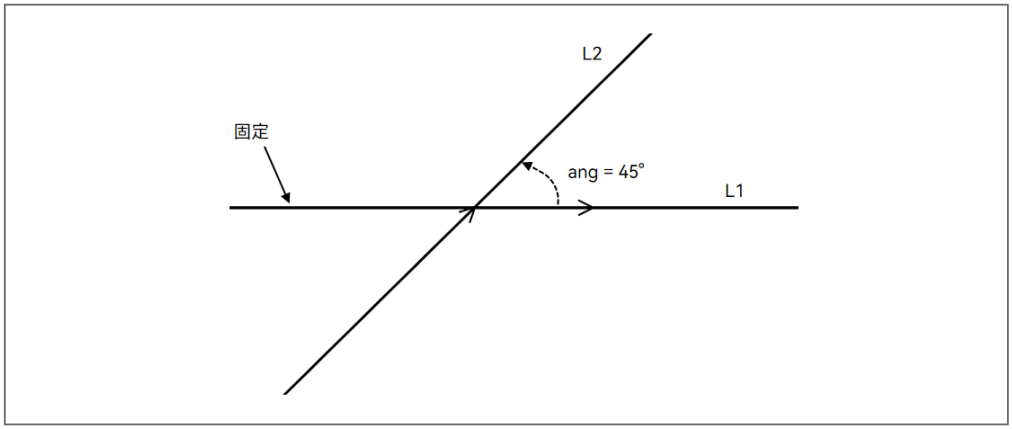
- 约束方位：Negative
  约束对齐方式：Positive
  ang = -（-135°） = 135°

图 5-13 同向对齐与反向方位



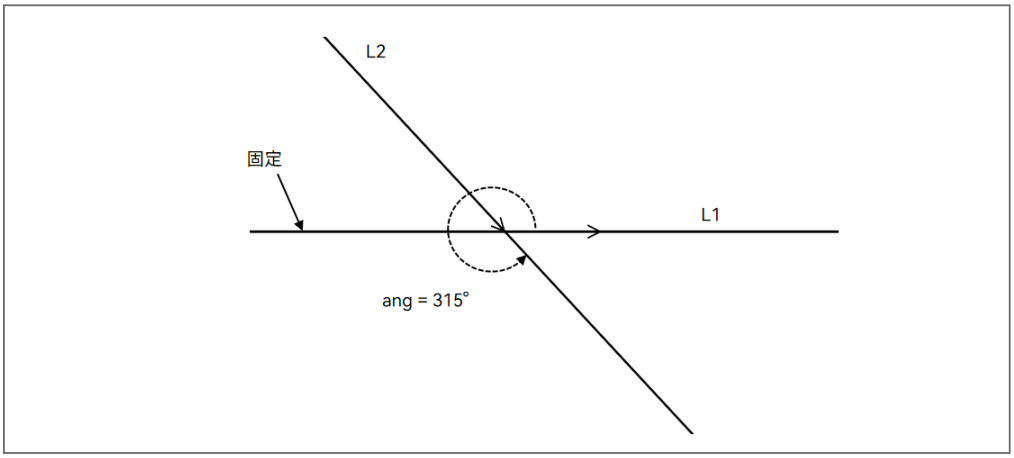
- 约束方位：Positive
  约束对齐方式：Negative
  ang = π-135° = 45°

图 5-14 反向对齐与同向方位



- 约束方位：Negative
  约束对齐方式：Negative
  ang = π+135° = 315°

图 5-15 反向对齐与反向方位



5.1.5 弧长约束

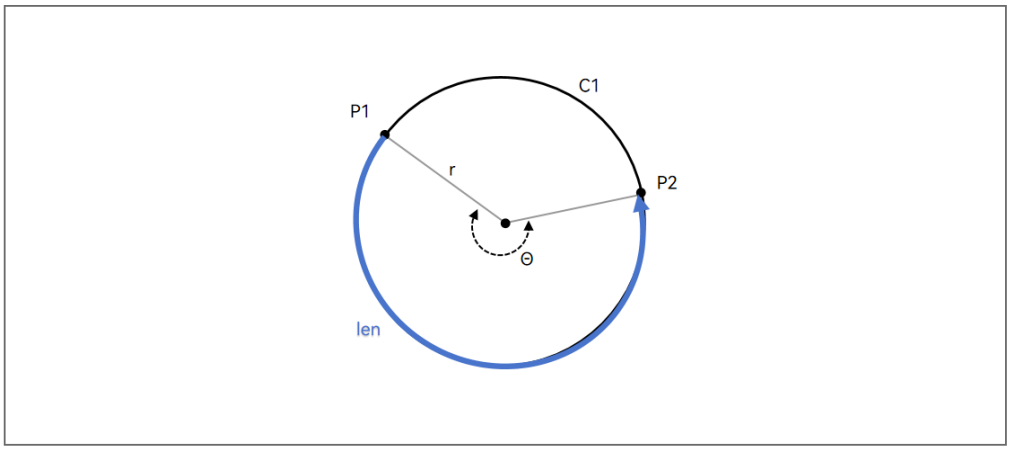
弧长约束限制了圆弧的长度。弧长约束的值是沿圆周的起点逆时针方向行进到终点的长度。

弧长约束带有一个具体的值，该值作用于三个几何元素，分别是一个圆和定义有界弧的两个点。如果圆位于可伸缩集合中，那么不支持在该圆上添加弧长约束。

如果模型中有足够的自由度，本软件可以改变两点的位置和圆的半径来求解弧长约束。请注意，这里的点不是指圆心，而是用于定义有界弧的两个点。

由于弧长约束是有符号的尺寸约束，所以在定义弧长约束时，点的选择顺序很重要，以圆周上从起点到终点的逆时针方向为正向，弧所对应的角度我们称为环绕角。图 5-16 弧长和环绕角中，假设圆 C1 的半径为 r，环绕角大小为 θ，则弧长 len 可以表示为：len = θ * r。

图 5-16 弧长和环绕角



如果两点之间的弧长包含了多个圆周，本软件可以将弧环绕圆周，使其长度大于单个圆的周长，从而求解弧长，这种弧环绕圆周的方式称为环绕（winding）。请注意，弧长约束并未限制环绕角的大小。当本软件通过环绕的方式求解模型时，环绕角将超过 2π（360 度）。

对于欠约束的模型，本软件可能会改变圆的半径或点的位置来寻找解决方案。

涉及的接口：**addConstraint**。

5.1.5.1 环绕角和环绕数

无论两点与圆之间的相对位置如何排列，只要应用程序通过本软件添加弧长约束，本软件就会自动在两点与圆之间添加重合约束。

本软件中，一条弧可以包括一个圆任意数量的完整周长。弧中包含的完整圆周数称为环绕数，弧所对应的角度称为环绕角。环绕数是求解弧长约束时环绕角的模数。比如，环绕角为 5π 的弧所对应的环绕数为 2，环绕角为-3π 的弧所对应的环绕数为 1（以相反方向测量）。

应用程序指定的环绕角可以作为弧长环绕圆周的次数指南，并对解决方案进行更精细的控制。建议应用程序使用角度值 π 来定义环绕角，而非具体的数值，这样能够确保解决方案的连续性并避免不必要的跳跃。

对于欠约束的模型，本软件可能会改变圆的半径或点的位置来找到解决方案。例如，如果圆的半径不受约束，且弧长约束的值大于当前圆的周长，那么在求解过程中，本软件可能会增大圆的半径，使环绕角小于 2π。

请注意，环绕角只是为所需的解决方案提供一个大致的依据，并不会迫使本软件求解任何特定的环绕数。根据模型的其他输入，本软件仍然可以将弧围绕圆周多次来找到解决方案。

5.1.6 曲线长度约束

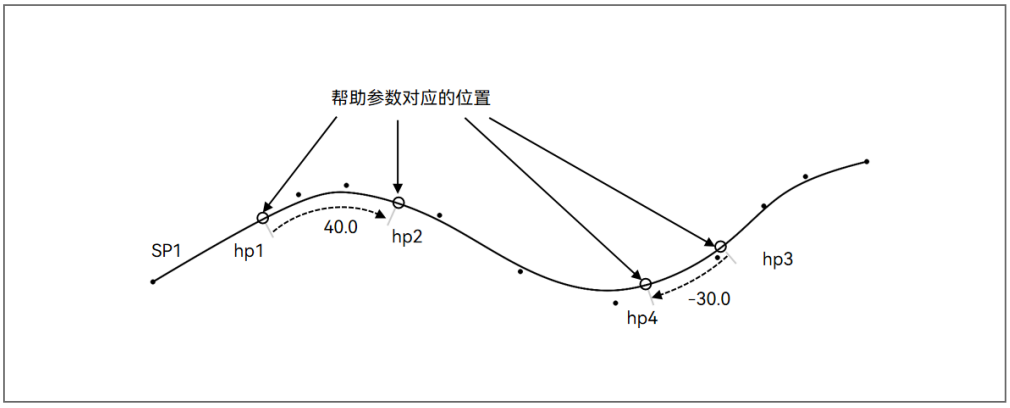
曲线长度约束可以用于约束椭圆与参数化曲线的长度。参数化曲线包括控制点/插值点样条曲线、圆锥曲线、偏移曲线、复制曲线、自定义回调曲线与自定义参数曲线。

曲线长度约束可以作用于曲线的全长，也可以作用于曲线的一部分。约束的对象为一条曲线与两个端点，端点的位置由曲线的帮助参数来决定。您可以在添加曲线长度约束时指定帮助参数的初始近似值，正式求解时，本软件会根据实际情况自动调整帮助参数的值。关于帮助参数的详细信息，请参见 5.1.2.3 帮助点和帮助参数。

曲线长度约束的值是一个带符号的值，如果将曲线长度约束的值设置为负，求解后，第一个帮助参数的值大于第二个帮助参数的值。图 5-17 控制点样条曲线上的曲线长度约束中显示了控制点样条曲线 SP1 上的两个曲线长度约束的值，图中 hp1、hp2、hp3 及 hp4 均为指定 SP1 上具体位置的帮助参数。

- 曲线长度约束的第一个帮助参数为 hp1，第二个帮助参数为 hp2，设置曲线长度约束的值为正，求解后，hp1 < hp2。
- 曲线长度约束的第一个帮助参数为 hp3，第二个帮助参数为 hp4，设置曲线长度约束的值为负，求解后，hp3 > hp4。

图 5-17 控制点样条曲线上的曲线长度约束



如果您需要测量开放曲线的全长或周期曲线的全长，请参见 6.2.5 曲线长度约束。

请注意，在一个周期曲线上，曲线长度约束的值可以环绕数次。

涉及的接口：**addConstraint**。

5.1.7 圆锥曲线的 rho 约束

圆锥曲线有双曲线、抛物线与椭圆三种形状，rho 是一个控制圆锥曲线形状的参数，圆锥曲线的 rho 约束主要用于约束 rho 的值，使得圆锥曲线在求解过程中保持形状不变。应用程序可以使用 **addConstraint** 来添加该约束。圆锥曲线的 rho 约束属于尺寸约束，与圆的半径约束类似，该尺寸约束关联了一个变量值，并通过该变量值来限制圆锥曲线的 rho。

在添加圆锥曲线的 rho 约束时需要控制 rho 的取值范围为(0,1)，rho 的取值决定了圆锥曲线的形状，如双曲线、抛物线或椭圆。

- 0 < rho < 0.5：圆锥曲线表现为一条椭圆形曲线。
- rho = 0.5：圆锥曲线表现为一条抛物线。
- 0.5 < rho < 1：圆锥曲线表现为一条双曲线的单侧曲线。

rho 的值越大，曲线的曲率越大。相反，rho 的值越小，曲线曲率越小。当 rho 的值趋近于 0 时，曲线变得平坦并近似为一条直线。当 rho 的值趋近于 1 时，曲线近似由两个线性部分组成，在控制点处存在切线不连续的情况。有关圆锥曲线的详细信息，请参见 8 圆锥曲线。

5.1.8 轮廓偏置约束

轮廓是由多个独立的几何元素、连接这些几何元素的点以及多个重合约束组成。

您可以使用 **addConstraint** 来添加轮廓偏置约束。添加轮廓偏置约束需要的数据有：

- 基础轮廓 ID（baseGeometryIDs）：用于偏置的原始轮廓。

- 偏置轮廓 ID（offsetGeometryIDs）：偏置后的轮廓。
- 偏置距离（offset）：用于偏置的距离。
- 偏置方向（side）：用于偏置的方向。

添加轮廓约束的 6 点注意事项：

- 2D 求解器不负责创建轮廓，所以在应用程序添加轮廓偏置约束时，需要提前在约束系统中创建好基础轮廓与偏置轮廓。
- 基础轮廓与偏置轮廓的几何元素类型要一 致，以下情况除外：
  - 可以在直线与线段之间添加轮廓偏置约束。
  - 如果基础轮廓的类型为椭圆或参数化曲线，那么偏置轮廓所对应的几何元素必须是他们的偏移曲线。
- 基础轮廓的数量必须等于偏置轮廓的数量，数组中基础轮廓和偏置轮廓的关系是一对一的且类型一致，比如 baseGeometryIDs[0]是圆，baseGeometryIDs[1]是直线，那么 offsetGeometryIDs[0]也必须是圆，offsetGeometryIDs[1]是直线。
- 偏置方向只支持 LEFT 与 RIGHT，不支持 INSIDE 与 OUTSIDE。对于圆（椭圆）来说，偏置方向为 LEFT 时，半径（长轴半径与短轴半径）变小，偏置方向为 RIGHT 时，半径（长轴半径与短轴半径）会变大。
- 确保偏置轮廓位于正确的位置，并且方向与基础轮廓相同，否则可能无法获得预期的解决方案。
- 单独的点不能添加轮廓偏置约束。

5.2 逻辑约束

逻辑约束与尺寸约束最大的不同在于其没有关联值。工程图纸上通常会明确地显示尺寸约束，但不会明显表示出逻辑约束。

在一种特殊情况下，具有值的尺寸约束能够暗示逻辑约束的存在，即两条线之间的距离约束，暗示了它们之间的平行约束。

本章将对本软件支持的逻辑约束进行逐一介绍。

涉及的接口：**addConstraint**，**getConstraintType**，**getFixationType**。

5.2.1 平行约束

应用程序只能在具有方向的几何元素之间添加平行约束，具有方向的几何元素包括线段、直线、椭圆以及参数化几何元素。

在两个几何元素之间添加平行约束与将两个几何元素的角度约束设置为 0 是不同的。平行约束可以指定两个几何元素的方向同向或逆向，而 0 角度规定两个几何元素的方向必须同向。例如，现有两条直线 L1 与 L2，在它们之间添加一个平行约束。求解后，两条直线可以是平行的或反平行的（平行是指两条线同向，反平行是指两条线逆向），如图 5-18 使用平行约束实现的两种解决方案所示。

图 5-18 使用平行约束实现的两种解决方案

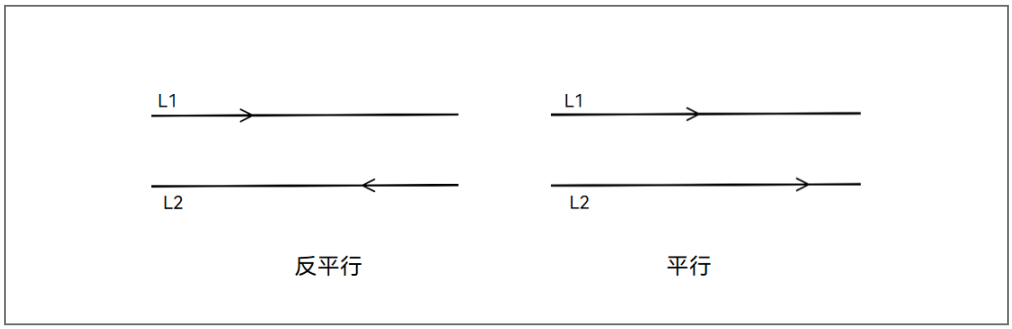
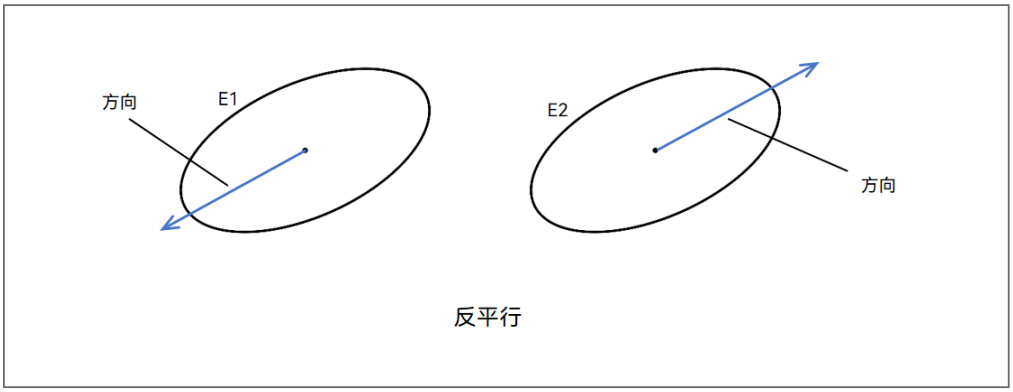


图 5-19 添加了平行约束的椭圆对展示了椭圆 E1 与椭圆 E2 反平行的解决方案，图中蓝色箭头分别表示 E1 与 E2 的方向。

图 5-19 添加了平行约束的椭圆对



平行或反平行这两种解决方案可以通过平行约束的约束对齐方式来控制，如果您想改变约束对齐方式，您可以使用 **updateConstraintAlignment** 接口来控制。

5.2.2 垂直约束

应用程序只能在具有方向的几何元素之间添加垂直约束，从而使两个几何元素的方向相互垂直。

与平行约束一样，垂直约束也有两个解决方案，您可以使用 **updateConstraintAlignment** 接口来控制约束对齐方式。

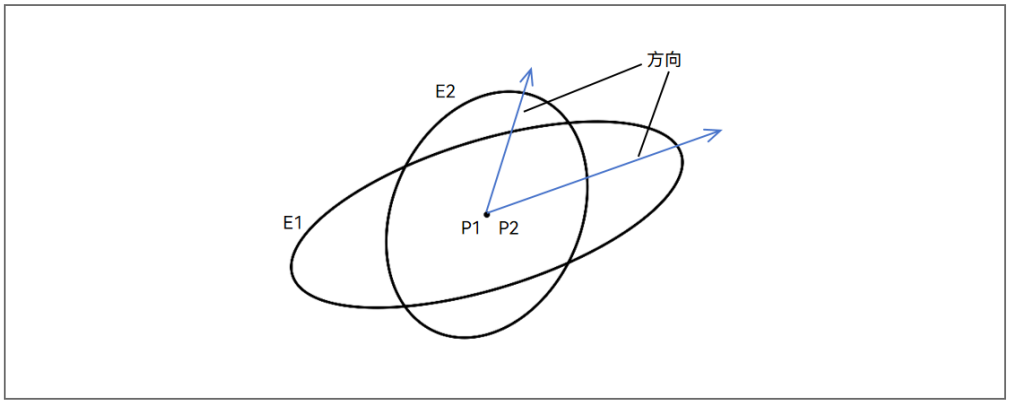
5.2.3 同心约束

应用程序可以在圆、椭圆和点的任意组合之间添加同心约束。

同心约束意味着几何元素的中心重合。例如，对于点和点而言，同心约束与重合约束的结果相同；对于圆和圆而言，同心约束会使两个圆构成同心圆；对于圆和点而言，同心约束会将该点约束在圆的圆心；对于椭圆和椭圆而言，长轴半径和短轴半径的交点会被视为中心，同心约束会使两个中心重合。

图 5-20 同心椭圆展示了在椭圆 E1 与椭圆 E2 之间添加同心约束的示例，图中蓝色箭头表示椭圆的方向，P1 表示 E1 的中心，P2 表示 E2 的中心。

图 5-20 同心椭圆



5.2.4 相切约束

应用程序只能在曲线几何元素与另一个曲线几何元素或线（直线或线段）之间添加相切约束。曲线几何元素包括圆、椭圆和参数化几何元素。

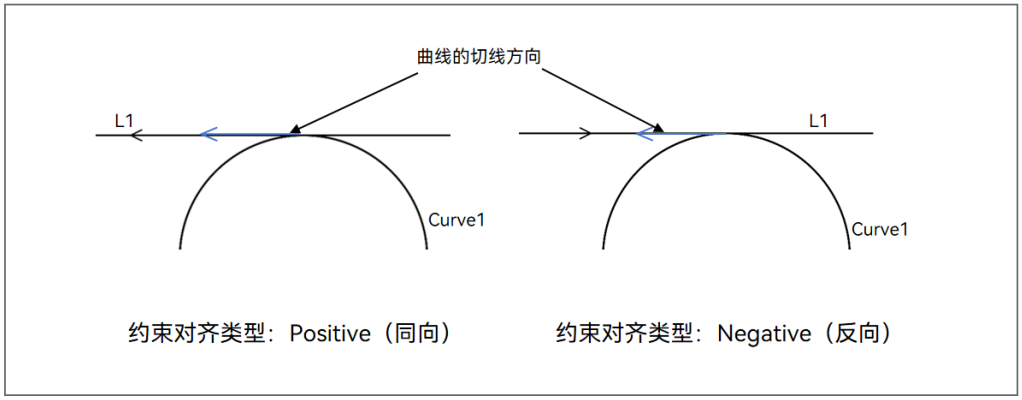
与圆或椭圆关联的相切约束可以指定一个帮助点。与参数化几何元素关联的相切约束必须指定一个帮助参数。有关帮助点与帮助参数的详细信息，请参见 5.1.2.3 帮助点和帮助参数。

5.2.4.1 相切约束的对齐方式

应用程序可以指定相切约束的对齐方式。对齐方式决定了与相切约束关联的两个几何元素的切线方向是同向还是反向。

如果方向之间的角度为 0°，则切线正方向对齐。如果方向是 180°，则切线反方向对齐。图 5-21 相切约束的对齐方式展示了切线对齐的两种情况，图中蓝色箭头表示曲线在相切时的切线方向。有关约束对齐方式的详细信息，请参见 2.2.1.1 约束对齐方式。

图 5-21 相切约束的对齐方式



添加相切约束时，如果应用程序未指定约束对齐方式，本软件可能自动调整约束对齐方式来获得解决方案。

涉及的接口：**addConstraint**，**updateConstraintAlignment**。

5.2.5 重合约束

应用程序可以在一个点与任意几何元素之间添加重合约束，此外，还能够在除了参数化几何元素之外的相同类型的几何元素之间添加重合约束。

如果在一个点与其他几何元素之间添加重合约束，那么点将位于该几何元素上。点与圆之间或点与椭圆之间关联的重合约束可以指定一个帮助点。点和参数化几何元素之间关联的重合约束必须指定一个帮助参数。有关帮助点和帮助参数的详细信息，请参见 5.1.2.3 帮助点和帮助参数。

在两个圆或两个椭圆之间添加重合约束，求解后的几何元素中心重合且半径相同；在两条线（直线或线段）或两个椭圆之间添加重合约束，可以使用 **updateConstraintAlignment** 接口指定重合约束的对齐方式，指示两者的方向为同向或反向。

涉及的接口：**addConstraint**，**updateConstraintAlignment**。

5.2.6 对称约束

应用程序只能在两个相同类型的几何元素和一条线（直线或线段）之间添加对称约束。

添加对称约束时需要指定一条线作为轴线，选取的两个相同类型的几何元素会在轴线的两侧对称排列，例如，如果与对称约束关联的几何元素是两个圆或两个椭圆，那么求解后它们的半径大小相同。

除了上述的几何元素以外，您可以在两条参数化曲线之间添加对称约束，更多详细信息，请参见 6.2 与参数化曲线关联的几何约束。

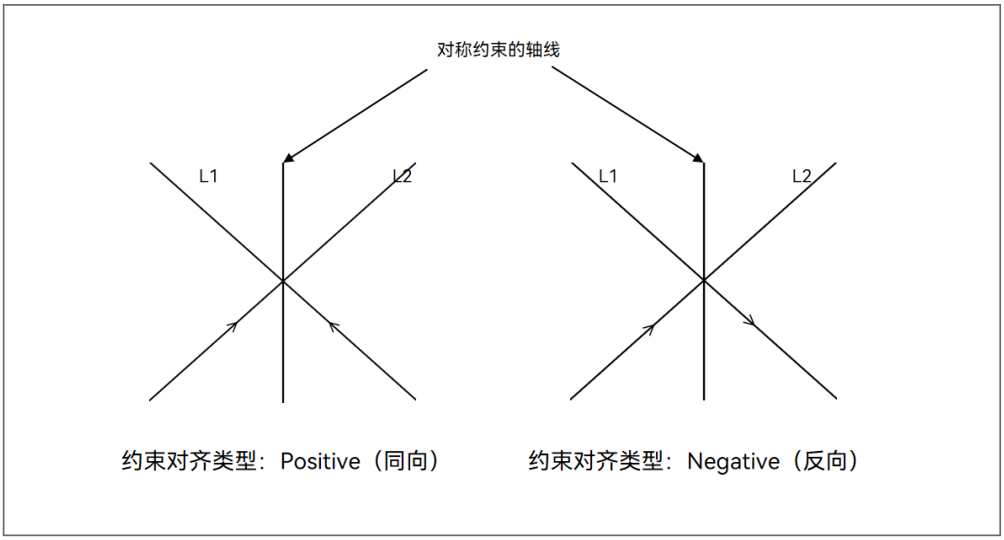
关于轴线数量的限制，请参见 2.4.7.2 过约束但一致的模型。

5.2.6.1 对称约束的对齐方式

如果关联对称约束的几何元素具有方向性，那么您可以为对称约束指定约束对齐方式。添加了对称约束的两个几何元素能够沿轴线折叠后相互重合。如果对称约束的对齐方式为同向，那么两者重合后的方向相同。如果对称约束的对齐方式为反向，那么两者重合后的方向相反。

请注意，轴线的方向与对称约束的对齐方式无关，在图 5-22 对称约束的对齐方式的示例中，直线 L1 与直线 L2 关于轴线对称，轴线的方向可以是上下两个方向，不影响解决方案。

图 5-22 对称约束的对齐方式



对称约束创建完成后，如果您想更改约束对齐方式，您可以通过 **updateConstraintAlignment** 接口来更新。

涉及的接口：**updateConstraintAlignment**。

5.2.7 法线约束

除了两条线（直线或线段）的组合以外，您可以在线、圆、椭圆以及参数化几何元素的两两组合之间添加法线约束。

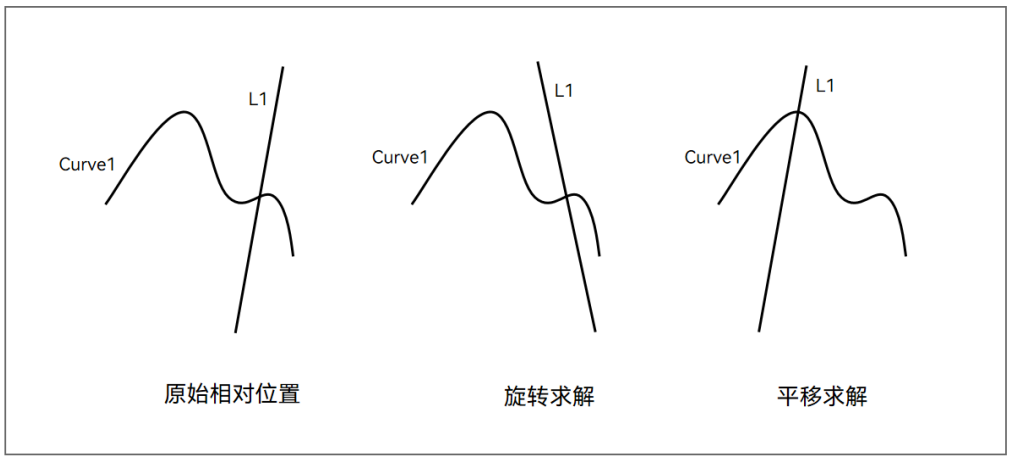
添加了法线约束的几何元素能够在特定位置相交，并且两者位于交点处的切线方向相互垂直。

说明
不能在两条线（直线或线段）之间添加法线约束，如果需要使两条线垂直，必须使用垂直约束。

在两个椭圆之间添加法线约束时，可能有多个交点，此时需要使用帮助点来识别交集。

由于旋转与平移相比能够提供更加直观的解决方案，本软件默认情况下会优先使用旋转来求解法线约束。图 5-23 通过旋转和平移来求解法线约束呈现了一条直线 L1 与一条样条曲线 Curve1 的原始相对位置，以及分别通过旋转与平移两种方式得到的解决方案。

图 5-23 通过旋转和平移来求解法线约束

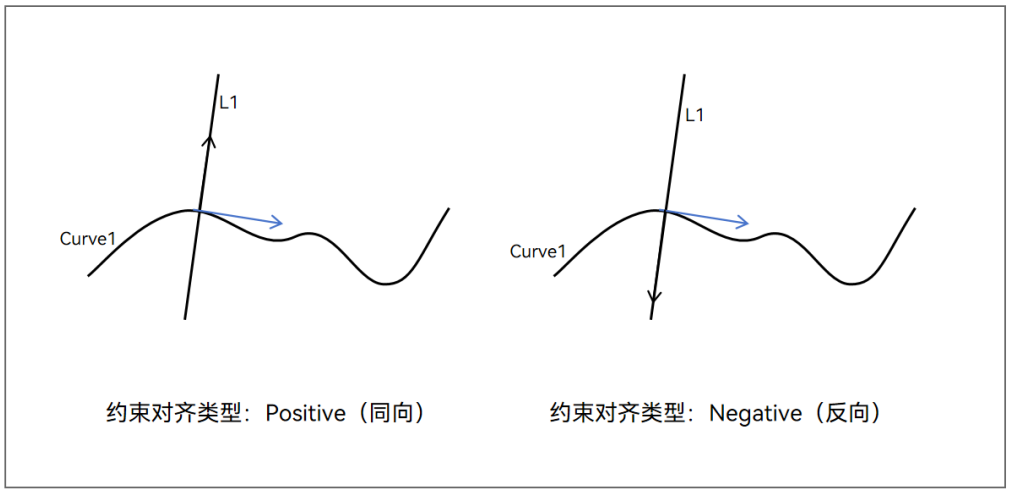


5.2.7.1 法线约束的对齐方式

法线约束的对齐方式有两种，分别是同向（逆时针）和反向（顺时针）。

约束对齐方式为同向时，第一个几何元素的方向或切线方向逆时针旋转 90°后，与第二个几何元素的方向或切线方向相同；约束对齐方式为反向时，第一个几何元素的方向或切线方向顺时针旋转 90°后，与第二个几何元素的方向或切线方向相同。如图 5-24 法线约束的对齐方式所示，在添加法线约束时，将样条曲线 Curve1 作为第一个几何元素，将直线 L1 作为第二个几何元素，图中蓝色箭头表示 Curve1 在相交位置的切线方向。设置约束对齐方式为 Positive，Curve1 的切线方向逆时针旋转 90°后与 L1 的方向相同。

图 5-24 法线约束的对齐方式



法线约束创建完成后，如果您想更改约束对齐方式，您可以通过 **updateConstraintAlignment** 接口来更新。

涉及的接口：**updateConstraintAlignment**。

5.2.8 等半径约束

应用程序可以在两个圆之间添加等半径约束。等半径约束能够控制两个圆的半径大小保持一致，但是不控制半径的实际值。

5.2.9 等参数约束

等参数约束主要作用在几何约束的帮助参数上，只有与椭圆或参数化曲线关联的几何约束才支持帮助参数。

您可以调用 **addConstraint** 接口来添加等参数约束。添加等参数约束需要的数据有：

- 第一个几何约束的 ID（firstConstraintId）
- 第二个几何约束的 ID（secondConstraintId）
- 第一个几何约束中，与公有几何元素相关帮助参数类型（firstParameterType）
- 第二个几何约束中，与公有几何元素相关的帮助参数类型（secondParameterType）

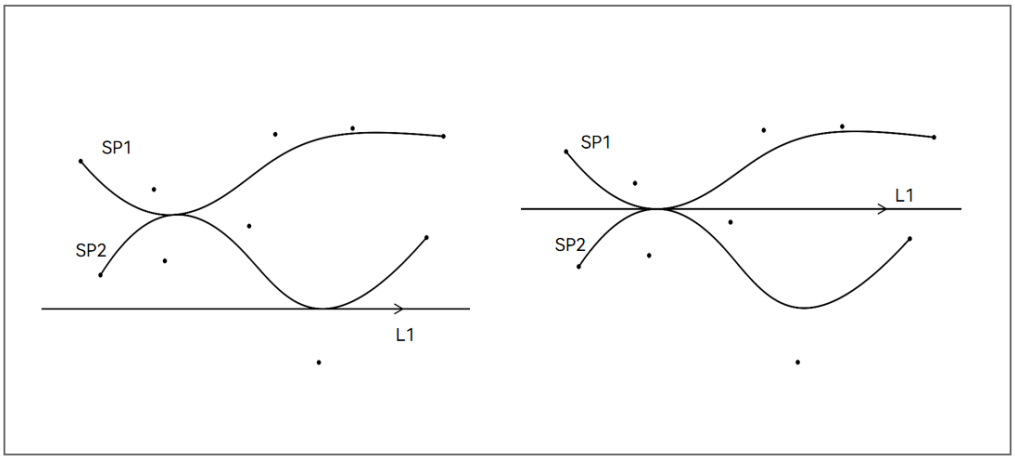
添加等参数约束的注意事项：

- 两个几何约束需作用到同一椭圆或参数化曲线上。
- 输入的帮助参数类型必须与两个几何约束中公有几何元素相关的两个帮助参数类型一致。

下面为您介绍一个简单的例子，现约束系统中有两条样条曲线 SP1 与 SP2，以及一条直线 L1。L1、SP1 分别与 SP2 之间关联了相切约束，SP2 此时为公有几何元素，如图 5-25 等参数约束所示。假设在 SP1 与 SP2 之间的相切约束中，SP1 是相切约束的第一个几何元素，SP2 是相切约束的第二个几何元素，那么相切约束在 SP1 上的生效位置对应的帮助点类型为 HELP_PARAMETER1，在 SP2 上生效位置对应的帮助点类型为 HELP_PARAMETER2； 同理，假设在 L1 与 SP2 之间的相切约束中，SP2 是相切约束的第一个几何元素，L1 是相切约束的第二个几何元素，那么相切约束在 SP2 上的生效位置对应的帮助点类型为 HELP_PARAMETER1，直线不支持帮助参数。

此时应用程序可以调用 **addConstraint** 接口添加一个等参数约束，同时将相关参数传入 2D 求解器，参数包括两个相切约束的 ID 与公有几何元素（SP2）相关的帮助参数类型，既 SP1 与 SP2 之间的相切约束的 HELP_PARAMETER2，L1 与 SP2 之间的相切约束的 HELP_PARAMETER1。求解后的 SP1 与 L1 会在 SP2 的同一个位置与其相切，如图 5-25 等参数约束所示。有关帮助参数类型的定义与注意事项，请参见 2.3 变量、方程和不等式。

图 5-25 等参数约束

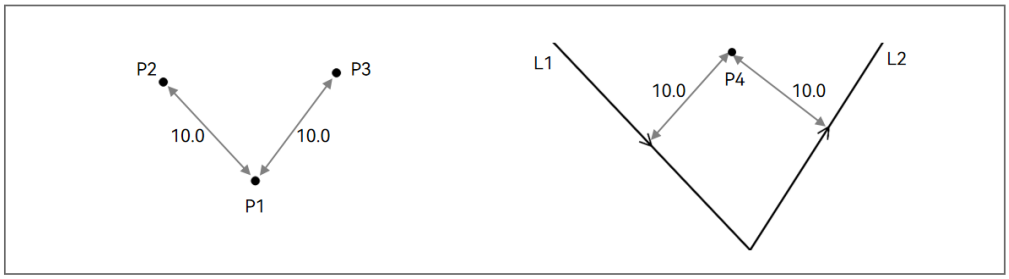


5.2.10 中点约束

应用程序可以在一个点与另外两个点之间或一个点与两条线（直线或线段）之间添加中点约束。中点约束能够控制该点到其他两个几何元素的距离相等，但是不控制距离的实际值。

如图 5-26 常用场景所示，点 P1 被约束为点 P2 和点 P3 的中点，点 P4 被约束为直线 L1 和直线 L2 的中点，此时 P4 位于 L1 和 L2 的角平分线上，直线的方向与中点约束无关，由于中点约束只控制距离相等，而不控制距离的实际值，所以图中的数值仅作参考。

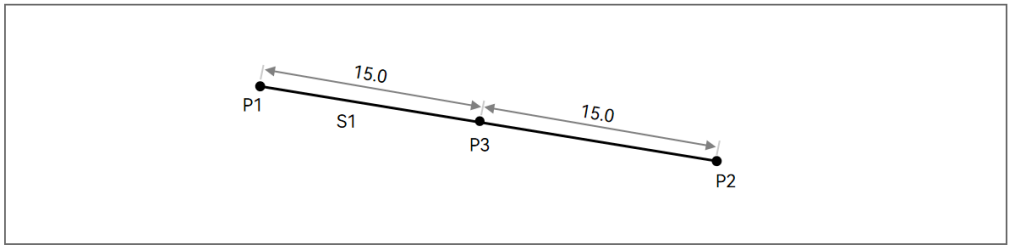
图 5-26 常用场景



中点约束的一个典型示例：通过中点约束将点约束在一条线段的正中间，如图 5-27 典型示例所示，中点约束不控制距离的实际值，图中的数值仅作参考。

1. 创建一个点 P3 与一条线段 S1。
2. 在 P3 与 S1 之间添加重合约束，使 P3 位于 S1 上。
3. 在 P3 与 S1 的两个端点 P1、P2 之间添加中点约束，使 P3 到 P1、P2 的距离相等，从而使 P3 位于 S1 的正中。

图 5-27 典型示例



有关为单个几何元素添加多个中点约束的限制，请参见 2.4.7.2 过约束但一致的模型。

5.2.11 等距离约束

应用程序可以在两对几何元素之间添加等距离约束。等距离约束能够控制第一对几何元素中的两个几何元素之间的距离与第二对几何元素中的两个几何元素之间的距离相等，但是不控制距离的实际值。

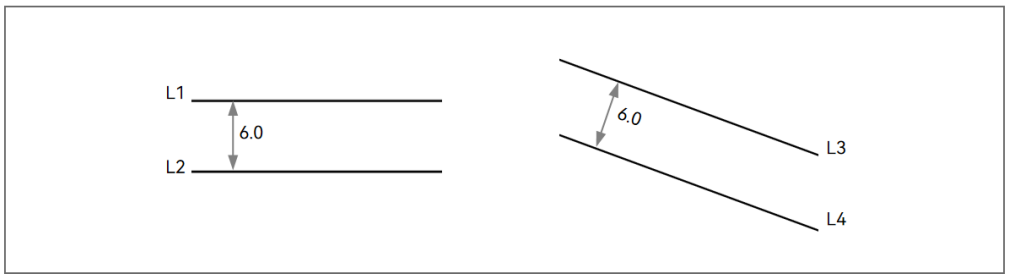
所有支持距离约束的几何元素都支持等距离约束。

- 两对圆：可以在两对圆中的任意圆上独立定义一个帮助点。
- 两对椭圆：需要指定帮助点或帮助参数。
- 两对参数化几何元素：必须指定帮助参数。但是当偏移曲线被约束到其父曲线或共享同一父曲线的另一条偏移曲线上时，等距离约束将控制偏移距离或偏移距离的差。

等距离约束的一个典型示例：通过在两条线段的端点之间放置等距离约束来使两条线段的长度相等。

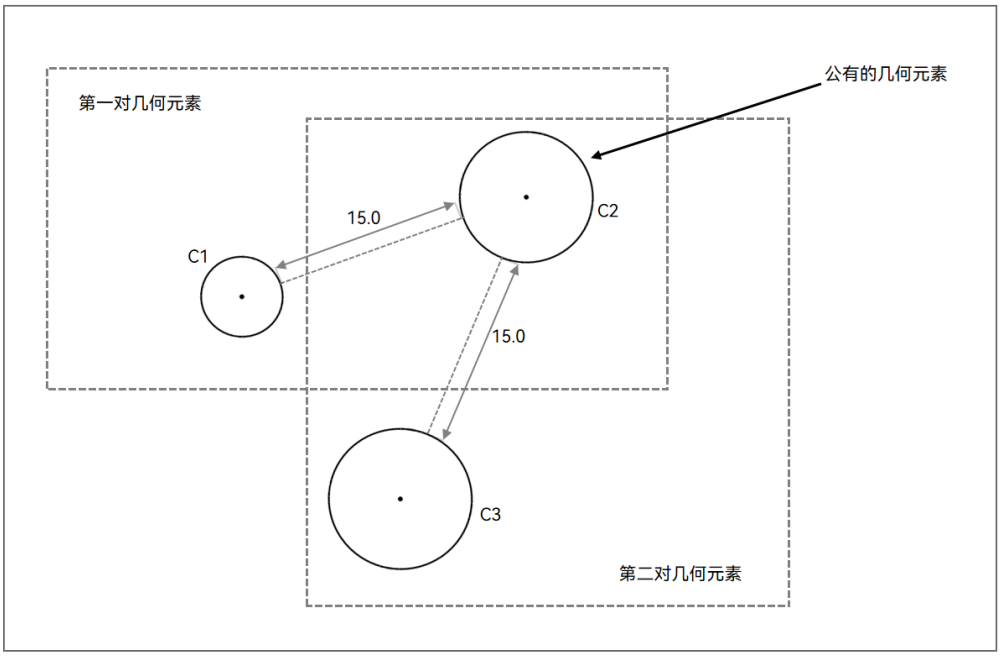
请注意，如果等距离约束关联的两个几何元素都是线（直线或线段），那么每一对中的两条线相互平行，就像在两条线之间使用距离约束的情况一样。如图 5-28 添加了等距离约束的直线对所示，现有两对直线，L1 与 L2 为第一对直线，L3 与 L4 为第二对直线，等距离约束使第一对直线的间隔距离与第二对直线的间隔距离相等，当前任意一对直线之间的距离值都为 6.0，意味着每对直线两两平行。 等距离约束并不控制距离的实际值，图中的数值仅作参考。

图 5-28 添加了等距离约束的直线对



等距离约束可以重复引用单个几何元素，但是最多引用两次。相当于等距离约束关联的两个几何元素对能够包含同一个几何元素，如图 5-29 重复引用单个几何元素的等距离约束所示，将 C1 与 C2 作为第一对几何元素，将 C2 与 C3 作为第二对几何元素，同样能够正确求解等距离约束。等距离约束并不控制距离的实际值，图中的数值仅作参考。

图 5-29 重复引用单个几何元素的等距离约束



关于等距离约束的限制，请参见 2.4.7.2 过约束但一致的模型。

5.2.12 阵列约束

在阵列几何元素中，同一类型的几何元素以规则的方式重复出现。这种规则的方式由阵列约束来控制。

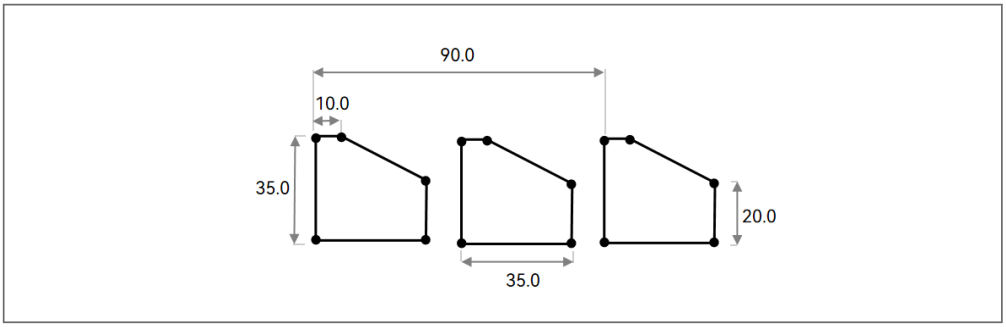
本软件支持线性与环形的阵列几何元素，线性阵列几何元素有一个参考元素或两个参考元素。环形阵列几何元素只有一个参考元素。有关阵列几何元素的详细信息，请参见 4.8 阵列几何元素。

阵列约束包括一个种子几何元素、一个被阵列约束的几何元素以及距离乘数。种子几何元素与被阵列的几何元素在阵列几何元素中都被称为“实例”。

应用程序可以在阵列几何元素中的任意实例上添加尺寸约束，此外，还能够在不同实例之间添加几何约束，本软件会自动处理过约束但一致的模型，并且保证阵列几何元素中的每个实例保持一致。

图 5-30 添加距离约束的线性阵列几何元素显示了为阵列几何元素内部的实例添加距离约束的示例，无论在哪一个实例上添加距离约束，都将被应用到其他实例上。如果您希望 2D 求解器能够更高效的求解阵列约束，建议您在单个实例上添加尺寸约束。

图 5-30 添加距离约束的线性阵列几何元素



5.2.12.1 涉及一个参考元素的阵列约束

涉及一个参考元素的阵列约束包括一个种子几何元素、一个被阵列约束的几何元素以及一个距离乘数。

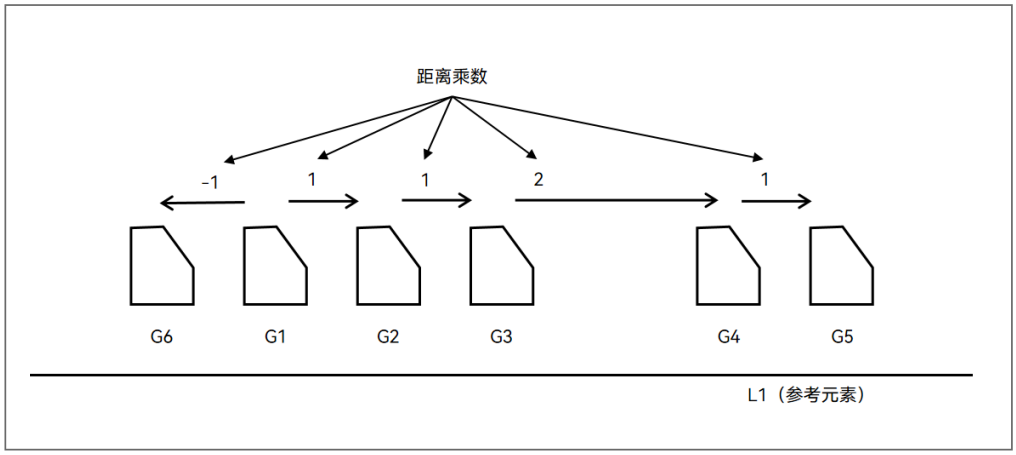
种子几何元素与被阵列约束的几何元素的类型必须相同，两者在阵列几何元素中都被称为“实例”。距离乘数是一个非零整数，与阵列值结合使用，能够定义实例之间的距离。有关阵列值的详细信息，请参见 4.8.1 参考元素和阵列值。请注意，距离乘数的符号以及阵列值的符号共同决定了几何元素被阵列的方向。

应用程序可以通过以下两种方案在几何元素之间添加此类阵列约束，从而形成一个完整的阵列几何元素：

- 主实例方法：在一个实例和其他每个实例之间分别添加阵列约束。我们称该实例为主实例。此时所有距离乘数都具有不同的绝对值，例如 1、2、3 等。这种方法能够提高阵列约束的使用性能。
- 链方法：在相邻的实例之间添加阵列约束。此时所有距离乘数都具有相同的绝对值，例如 1、1、1 等。

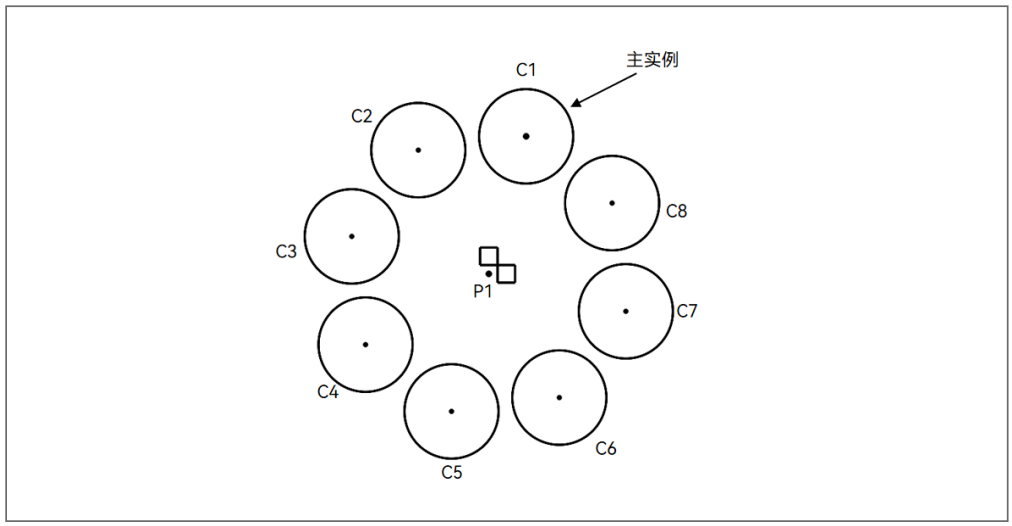
设置不同的距离乘数可以得到带有“间隙”的阵列几何元素。例如，图 5-31 线性阵列几何元素中的距离乘数中呈现的是以直线 L1 为参考元素，采用链方法实现的阵列几何元素，几何元素 G1、G2、G3、G4、G5 及 G6 均为阵列几何元素中的实例，上方的数字表示阵列约束的乘数。当突然增大某个阵列约束的乘数，会使实例与实例之间的距离成倍增长，比如 G3 与 G4 之间的阵列约束的乘数为 2。 从图中可以看出，G1 关联了两个阵列约束，乘数分别为+1 与-1。

图 5-31 线性阵列几何元素中的距离乘数



应用程序可以在两个实例之间添加多个阵列约束。图 5-32 环形阵列几何元素展示了一个典型的示例，以点 P1 为参考元素，利用多个阵列约束创建了一个封闭的环形阵列几何元素，圆 C1、C2、C3、C4、C5、C6、C7 及 C8 均为阵列几何元素中的实例，每个实例之间的角度距离相等，每个阵列约束都关联到同一个实例上，也就是 C1 上，我们称之为主实例。从 C1 开始，以逆时针方向旋转，C2 与 C1 之间的阵列约束的距离乘数为 1；C3 与 C1 之间的阵列约束的距离乘数为 2；以此类推。最后一个实例 C8 与 C1 之间关联了两个阵列约束，距离乘数分别为 7 和-1。

图 5-32 环形阵列几何元素



封闭的环形阵列几何元素不需要指定阵列值，因为阵列值会消除环形阵列几何元素的内部自由度。利用这一点，应用程序可以为一系列点和直线添加阵列约束来创建正多边形。有关如何在应用程序中使用阵列约束的详细信息，请参见 14.4 阵列约束和 14.5 正多边形。

涉及的接口：**addConstraint**，**setGeometryData**，**getGeometryData**。

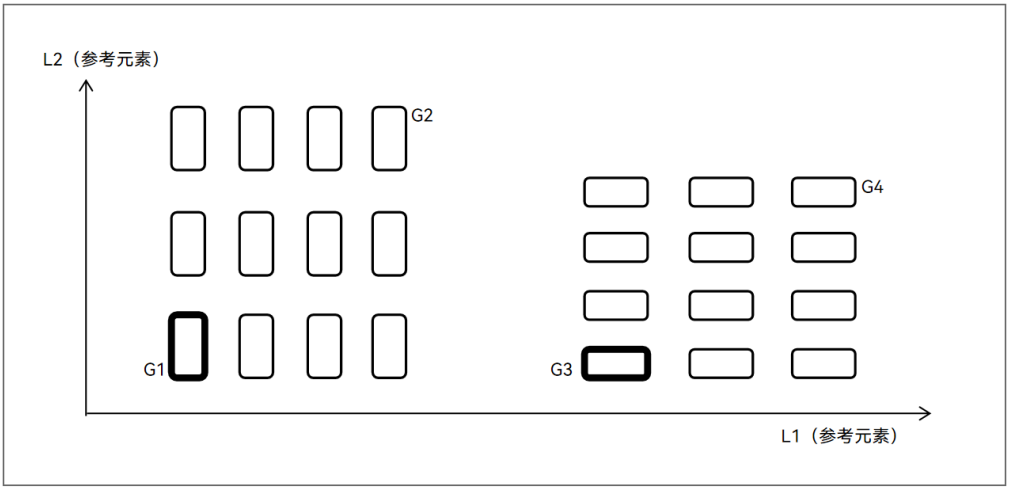
5.2.12.2 涉及两个参考元素的阵列约束

涉及两个参考元素的阵列约束包括一个种子几何元素、一个被阵列约束的几何元素以及两个距离乘数。

种子几何元素与被阵列约束的几何元素的类型必须相同，两者在阵列几何元素中都被称为“实例”。距离乘数是一个非零整数，与阵列值结合使用，能够定义实例之间的距离。有关阵列值的详细信息，请参见 4.8.1 参考元素和阵列值。请注意，距离乘数的符号以及阵列值的符号共同决定了几何元素在两个方向上被阵列的方向。

添加此类阵列约束与添加只涉及一个参考元素的阵列约束的方案有所不同，应用程序只能使用主实例方法而不能使用链方法，也就是说应用程序只能在主实例与其他实例之间添加阵列约束，不能在相邻实例之间添加阵列约束。在整个阵列几何元素中，阵列约束具有不同值对的距离乘数，例如(0, 1)，(1, 2)，(1, -1)。图 5-33 线性阵列几何元素示例展示了两个线性阵列几何元素，均以直线 L1 与 L2 作为参考元素，G1 与 G3 分别是两个阵列几何元素的主实例，其余的实例都与之关联有阵列约束。该图的两个阵列几何元素中，G1 和 G2 之间的阵列约束在 X 方向上的乘数为 3，在 Y 方向上的乘数为 2；G3 和 G4 之间的阵列约束在 X 方向上的乘数为 2，在 Y 方向上的乘数为 3。

图 5-33 线性阵列几何元素示例



涉及的接口：**addConstraint**，**setGeometryData**，**getGeometryData**。

5.2.13 等相对变换约束

等相对变换约束涉及两个等效的几何元素，以及一个相对变换几何元素。等相对变换约束是指在两个等效的几何元素中，一个几何元素通过刚性变换映射到另一个几何元素上，这种刚性变换由相对变换几何元素来决定。例如，两个几何元素都是控制点样条曲线且控制点的个数一致，我们将其称之为等效。相对变换几何元素是一个特殊类型的节点，用于存储和执行等相对变换约束的刚性变换。在不改变相对变换几何元素数据的情况下，通过将两个等效几何元素约束到相对变换几何元素上，我们可以确保这两个几何元素的相对位置与方向始终不变，即无论从任意方向观察或移动至任意位置，它们之间的相对关系都是不变的。

在添加等相对变换约束之前，应用程序需要添加一个相对变换几何元素，再使用等相对变换约束将等效的几何元素绑定到相对变换几何元素上。应用程序能够将多个几何元素绑定到一个相对变换几何元素上，从而形成多组映射，使每组中两个等效几何元素之间的相对位置关系保持一致。

等相对变换约束本身并没有定义几何元素之间应该如何映射或变换，这种定义存储在相对变换几何元素当中。应用程序可以通过更新相对变换几何元素的数据来调整刚性变换，便于在后续的几何处理中灵活地调整等效几何元素之间的关系。

涉及的接口：**addConstraint**，**setGeometryData**。

5.2.14 等曲线属性约束

根据需要约束的特定属性，应用程序可以在曲线几何元素（圆、椭圆和参数化几何元素）与直线之间或两个曲线几何元素之间添加等曲线属性约束，也可以在同一曲线几何元素上的两个位置之间添加等曲线属性约束。曲线几何元素包括圆、椭圆与参数化几何元素，本小节中统称为曲线。关于几何元素分类的详细信息，请参见 4.2 几何元素的分类。

等曲线属性约束需要通过帮助点或帮助参数来指定曲线的特定位置，并设置两个位置的属性相等。

当等曲线属性约束关联的其中一个几何元素为直线或圆时，由于两者的属性是常数，因此不需要指定特定位置。如果等曲线属性约束关联有椭圆，那么需要使用帮助点或帮助参数来定义特定位置。本软件支持的等曲线属性约束有以下五类：

- **等方向约束**
  设置两个几何元素在特定位置的切线方向相同。该几何约束与相切约束不同，相切约束关联的两个几何元素会在相切的位置重合，但是等方向约束没有在特定位置重合的定义。应用程序可以在以下几何元素之间添加等方向约束：
  - 参数化几何元素与参数化几何元素
  - 参数化几何元素与椭圆
  - 参数化几何元素与直线
  - 同一参数化几何元素上的两个位置
  - 椭圆与椭圆
  - 椭圆与直线
  - 同一椭圆上的两个位置
- **等一阶导数约束**
  设置两条曲线在特定位置的一阶导数相同。应用程序可以在以下几何元素之间添加等一阶导数约束：
  - 参数化几何元素与参数化几何元素
  - 参数化几何元素与椭圆
  - 同一参数化几何元素上的两个位置
  - 椭圆与椭圆

- 同一椭圆上的两个位置

■ **等二阶导数约束**
设置两条曲线在特定位置的二阶导数相同。应用程序可以在以下几何元素之间添加等二阶导数约束：

- 参数化几何元素与参数化几何元素
- 参数化几何元素与椭圆
- 同一参数化几何元素上的两个位置
- 椭圆与椭圆
- 同一椭圆上的两个位置

■ **等曲率约束**
设置两条曲线在特定位置的曲率相同。应用程序可以在以下几何元素之间添加等曲率约束：

- 参数化几何元素与参数化几何元素
- 参数化几何元素与椭圆
- 参数化几何元素与圆
- 同一参数化几何元素上的两个位置
- 椭圆与椭圆
- 椭圆与圆
- 同一椭圆上的两个位置

■ **等曲率导数约束**
设置两条曲线在特定位置的曲率导数相同。应用程序可以在以下几何元素之间添加等曲率导数约束：

- 参数化几何元素与参数化几何元素
- 参数化几何元素与椭圆
- 参数化几何元素与圆
- 同一参数化几何元素上的两个位置
- 椭圆与椭圆
- 椭圆与圆
- 同一椭圆上的两个位置

等曲线属性约束作用在曲线上的特定位置需要使用帮助点或帮助参数来指定。如果您还没有完全掌握帮助点或帮助参数的使用方法，请参见 5.1.2.3 帮助点和帮助参数。如果您想要使等曲线属性约束作用的具体位置不变，可以使用 **setHelpParameterFixedState** 接口来固定帮助参数。

等曲线属性约束支持约束对齐方式，约束对齐方式可以指定两条曲线的矢量属性（方向、一阶导数或二阶导数）为同向（平行）或反向（反平行），但是对于曲线的标量属性（曲率和曲率导数）来说，约束对齐方式仅仅表示值的符号，同向表示值的符号相等，反向表示值的符号相反。

涉及的接口：**addConstraint**，**setHelpParameter**，**setHelpParameter**，**getHelpParameter**，**getHelpParameter**，**setHelpParameterFixedState**，**setHelpParameterFixedState**。

5.2.15 固定约束

固定约束可以防止几何元素或集合在求解时发生移动。它既可以控制整个几何元素，也可以仅控制几何元素的中心或方向。

应用程序可以为同一个几何元素或集合添加多个固定约束，对于一些特殊的几何元素，固定约束能够固定其属性，例如圆的半径、椭圆的长轴半径以及椭圆的短轴半径。

固定约束有以下 6 种类型：

- Complete（完全固定）：固定几何元素或集合的所有自由度，防止整个几何元素或集合在求解时发生移动。适用于所有类型的几何元素与集合，并且该类型是唯一可以添加在点上的固定约束。
- Center（中心固定）：固定几何元素的中心坐标，防止几何元素的中心在求解时发生移动。该类型的固定约束仅适用于圆、椭圆以及其他具有 LCS（局部坐标系）原点的几何元素，例如自定义参数曲线、自定义回调曲线、刚性集合、可伸缩集合、单向可伸缩集合以及双向可伸缩集合。
- Direction（方向固定）：固定几何元素或集合的方向，防止几何元素或集合的方向在求解时发生改变。适用于直线、线段、椭圆、圆锥曲线、自定义参数曲线、自定义回调曲线、刚性集合、可伸缩集合、单向可伸缩集合以及双向可伸缩集合。这种固定约束适用于任何有方向的几何元素，并且可以适用于整个集合，包括刚性集合、可伸缩集合、单向可伸缩集合与双向可伸缩集合。对于具有旋转对称性的集合，可以使用该类型的固定约束来占用集合的旋转自由度。
- Radius（半径固定）：固定圆的半径，防止圆的半径长度在求解时发生改变。
- Major_radius（长轴半径固定）：固定椭圆的长轴半径，防止椭圆的长轴半径长度在求解时发生改变。
- Minor_radius（短轴半径固定）：固定椭圆的短轴半径，防止椭圆的长轴半径长度在求解时发生改变。

🕮 说明
半径固定、长轴半径固定与短轴半径固定均只固定轴的长度，与方向无关。

涉及的接口：**fixGeometry**。

5.3 扩展几何约束类型

本章的前几节详细介绍了本软件直接支持的几何约束。您可以在应用程序中组合已有的几何约束，或使用方程和不等式来构建更复杂的几何约束。

有关构建复杂约束的其他方法，请参见 14.6 约束曲线的长度和 14.7 使用非线性方程构建高级几何约束。

5.3.1 几何约束的迭代求解

应用程序可以实现本软件本身未提供的复杂几何约束，例如面积约束或体积约束等。

求解面积约束相当于求解一个一维方程：

area(dimension) - target_area = 0

尺寸约束有线性的，也有与角度相关的。对于与角度相关的尺寸约束，通常有一个限制，即它的值可以从 0 变化到 2π（360°），之后又回到起点，形成一个循环或称为“环绕”。而面积函数相对来说更加复杂，它包含最大值、最小值以及不连续点，意味着不同的值可能会使面积产生较大的变化，甚至在某些点上不连续或不可导。

通过改变面积约束的值，评估约束系统，然后计算面积，来对函数本身进行采样。由于需要确保函数在任何时候都是单值，即每个面积约束的值对应唯一值，因此在尝试采样另一个函数点之前，需要撤销之前的求解。如果没有正确地撤销之前的评估，模型可能会变得路径依赖，相当于即使起点和终点相同，途中经过不同的路径，结果也会不同。在物理和工程应用中，路径依赖对确定的和可预测的结果相悖。

一旦确定了根的区间（即已知函数在两个点之间变号），一维方程的求解就会变得相对简单，有许多有效的方法可以从一对区间中找到一个零点。问题的核心在于如何确定包含零点的区间，以及如何处理多个偶重根（即触及零点的最大值和最小值）。在处理这类问题时，我们通常会使用数值方法，如二分法、牛顿法等来逼近函数的零点。这些方法的基本思想是通过迭代来逐步缩小根的区间，直到满足所需的精度要求。

5.3.2 使用非线性方程求解几何元素

使用非线性方程来求解几何元素可以模拟各种复杂几何约束。

有关非线性方程的使用方法，请参见 14.7 使用非线性方程构建高级几何约束。

6 参数化几何元素

本软件直接支持的几何元素分为解析几何元素和参数化几何元素。参数化几何元素包括多种曲线，这些曲线上的任一位置都与一个参数相关联。

本软件支持的参数化曲线有以下四种类型：

- 样条曲线
- 圆锥曲线
- 自定义参数曲线
- 自定义回调曲线

本章主要介绍任意类型的参数化几何元素可能产生的问题。关于如何创建和使用这些类型的曲线，请参见 7 样条曲线 ，8 圆锥曲线，10 自定义曲线。

说明
如果一个模型包含同一参数化曲线的多个实例，应用程序可以使用复制曲线有效地将这些实例添加到本软件中。有关更多信息，请参见 9 偏移曲线与复制曲线。

6.1 固定的、可伸缩的、自由的参数化曲线

本软件中，参数化曲线可以通过特定操作被分为形状固定的、形状可伸缩的以及灵活的曲线。

被加入刚性集合的参数化曲线相当于形状固定的曲线，这类曲线能够被平移或旋转，但是其形状与大小无法被更改。

被加入特定可伸缩集合的曲线相当于形状可伸缩的曲线，这类曲线具有一到两个额外的自由度，支持以下三种缩放形式：

- 形状等比伸缩：曲线形状在任意方向上等比例缩放。
- 形状单向伸缩：曲线形状在单个方向上缩放，与该方向垂直的方向不发生变化。
- 形状双向伸缩：曲线形状在两个垂直方向上依据对应的比例独立进行缩放。

灵活的参数化曲线具有大量的内部自由度，自由度的数量和类型取决于曲线的定义方式，进一步的细节如下所示。

6.2 与参数化曲线关联的几何约束

本节主要介绍可以添加到参数化曲线上的几何约束类型。除了特殊说明以外，样条曲线、圆锥曲线、自定义回调曲线和自定义参数曲线均符合下述内容。

6.2.1 相切约束，法线约束，距离约束，与点的重合约束

简要介绍与参数化曲线相关的相切约束，法线约束，距离约束，与点的重合约束。

以下几何元素组合支持添加相切约束与法线约束：

- 参数化曲线与线（直线和线段）
- 参数化曲线与圆
- 参数化曲线与椭圆
- 参数化曲线与参数化曲线

应用程序能够在点与参数化曲线之间添加重合约束，求解后该点位于参数化曲线上。如果参数化曲线包含不连续处（如尖点、角点等），那么该点也有可能位于这些不连续处。

在一条参数化曲线与其他任何类型的几何元素之间添加距离约束，可以控制它们之间的距离，并找到精确位于不连续点上的解决方案。例如，与 G1 不连续点具有距离约束的点可以位于以该不连续点为中心的圆弧上的任何位置。更多详细信息，请参见 6.4 与不连续的曲线相关的几何约束。

当使用上述类型的几何约束时，应用程序必须向本软件提供帮助参数。更多详细信息，请参见 6.3 帮助参数。

6.2.2 对称约束，重合约束，阵列约束，等相对变换约束

简要介绍与参数化曲线相关的对称约束，重合约束，阵列约束与等相对变换约束。

应用程序可以在两条自定义回调曲线或自定义参数曲线之间添加上述几何约束。应用程序能够控制这些曲线的形状，当曲线关联的其他几何元素发生变化时，应用程序会控制曲线的形状来满足它们之间的几何约束。

当两条样条曲线具有相同的阶数、控制点数量、控制点权重及节点向量时，应用程序才能在这两条样条曲线之间添加上述几何约束。例如，两条样条曲线满足前提条件，此时在两条样条曲线之间添加重合约束，求解后的控制点将会重合。

圆锥曲线的类型不同，其所支持的几何约束类型也不相同。更多详细信息，请参见 8.3 圆锥曲线支持的几何约束。

与添加具有相同形状的曲线相比，使用本软件的复制曲线类型对参数化曲线进行建模会更加高效。本软件支持应用程序在参数化曲线和它的复制曲线之间使用重合约束、阵列约束与等相对变换约束。有关参数化曲线与复制曲线的详细信息，请参见 9 偏移曲线与复制曲线。

6.2.3 平行约束，垂直约束，角度约束

简要介绍与参数化曲线相关的平行约束、垂直约束与角度约束。

以下几何元素组合支持添加上述几何约束：

- 参数化曲线与线（直线和线段）
- 参数化曲线与椭圆
- 参数化曲线与参数化曲线

这些几何约束能够控制几何元素的相对方向。

- 应用程序可以选择任意方向作为自定义回调曲线与自定义参数曲线的方向。更多详细信息，请参见 10 自定义曲线。
- 本软件会根据现有数据来计算圆锥曲线的方向。

6.2.4 等曲线属性约束

简要介绍与参数化曲线相关的等曲线属性约束。

以下几何元素组合支持添加等曲线属性约束的：

- 参数化曲线/线（直线和线段）
- 参数化曲线/圆
- 参数化曲线/椭圆
- 参数化曲线/参数化曲线

请注意，上述组合并不支持所有类型的等曲线属性约束。例如，在参数化曲线与圆之间添加的等方向约束无效。更多详细信息，请参见 5.2.14 等曲线属性约束。

6.2.5 曲线长度约束

应用程序可以使用曲线长度约束来控制参数化曲线与椭圆的长度。

更多详细信息，请参见 5.1.6 曲线长度约束。

帮助参数的位置使用一个确切的值来表示。要想确定一条开放曲线的完整长度，可以将帮助参数的位置分别固定在曲线的起始位置与结束位置，要想确定一条周期曲线的

完整长度，需要使用两个固定的帮助参数位置，其中一个帮助参数的值 = 参数周期 + 另一个帮助参数的值。

6.3 帮助参数

与参数化曲线关联的任何重合约束、距离约束、相切约束、法线约束或等曲线属性约束都需要帮助参数来指定期望生效位置。

上述几何约束都可以关联在曲线的任意点上，固存在多个解决方案。应用程序可以通过 **setHelpParameter** 函数向 2D 求解器提供一个帮助参数，来指明用户的需求，减少偏离用户意图的解决方案数量。帮助参数类似于圆与椭圆上的帮助点。

只要为帮助参数指定一个与目标值近似的初始值，2D 求解器在求解时会自动计算出一个准确的值。应用程序可以调用 **getHelpParameter** 接口来获取调整后的值。

涉及的接口：**setHelpParameter**，**getHelpParameter**。

6.3.1 固定帮助参数

帮助参数的位置能够沿着参数化曲线不断移动，有利于寻找解决方案。应用程序可以要求 2D 求解器在求解期间不更改帮助参数的值，这被称为固定帮助参数。

固定帮助参数的典型示例：

- 固定点与参数化曲线之间的重合约束的帮助参数，点将不能在参数化曲线上自由移动，移动该点将会改变参数化曲线的形状。
- 固定直线与参数化曲线之间的相切约束的帮助参数，并将帮助参数的值设置为曲线的起始点，直线将始终在曲线的初始位置与曲线保持相切。

涉及的接口：**setHelpParameterFixedState**，**getHelpParameterFixedState**。

6.3.2 等参数约束

如果两个几何约束作用于同一椭圆或参数化曲线，那么应用程序可以在这两个几何约束的帮助参数之间添加等参数约束，使两个帮助参数的值在求解时相等，对应的位置相同，但并不会控制帮助参数的实际值。

例如，两条直线与一条样条曲线相切，在这两个相切约束之间添加一个等参数约束，此时两条直线会在样条曲线的同一个参数处与其相切。

6.3.3 椭圆的帮助参数

应用程序为椭圆添加相关的几何约束时，可以使用帮助点或帮助参数来指定几何约束关联的具体位置。

当一个几何约束与椭圆关联时，既可以指定帮助参数，又可以指定帮助点来指定椭圆的具体位置。这种情况下，如果指定了帮助参数，本软件会优先使用帮助参数进行求解。

如果应用程序选择使用帮助参数，那么必须遵循本软件的参数化方法。对于椭圆来说，使用帮助参数 t 来定义椭圆的方程：

$$P(t) = C + (R\cos t)X + (r\sin t)Y$$

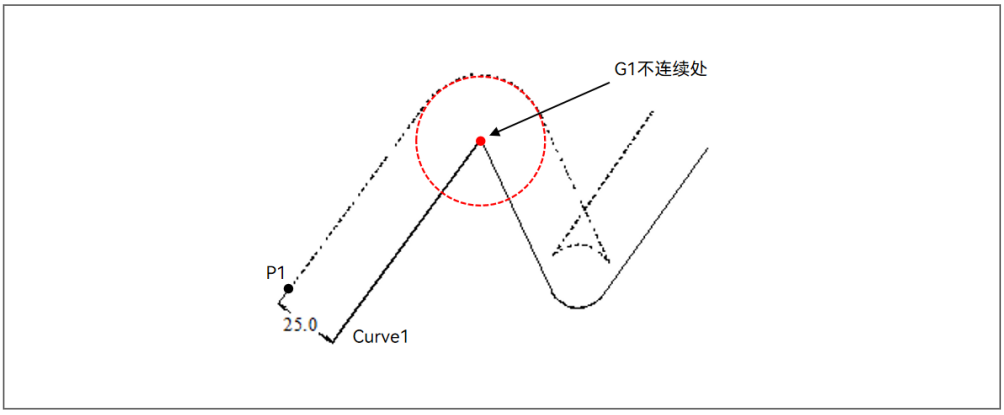
其中 $(0 <= t < 2\pi)$，C 是椭圆的圆心，X 和 Y 分别是长轴半径和短轴半径的单位向量，R 和 r 分别表示长轴半轴和短轴半轴的长度。

6.4 与不连续的曲线相关的几何约束

模型的解决方案可能位于参数化曲线的不连续处。

当一个点与 G1 不连续处关联了距离约束，该点将位于以 G1 不连续处为中心的圆弧上的任意位置。图 6-1 点与不连续的曲线之间的距离约束显示了在不连续的参数化曲线 Curve1 与点 P1 之间添加距离约束后的解决方案。黑色虚线表示 P1 沿 Curve1（包括 G1 不连续处）被拖拽时，满足距离约束的路径。红色点表示 G1 不连续处，红色虚线是以该处为中心的圆弧。

图 6-1 点与不连续的曲线之间的距离约束

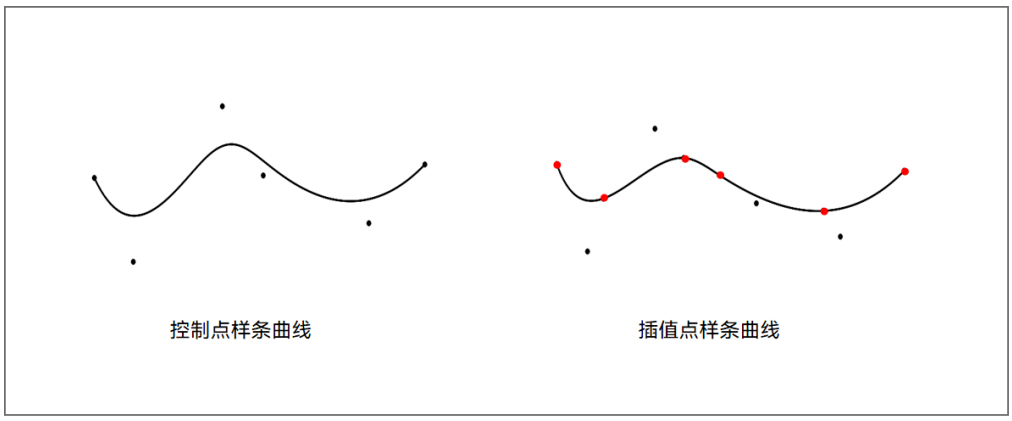


7 样条曲线

本软件使用 NURBS 曲线来表示样条曲线。一般来说，所有应用程序对 NURBS 曲线的基本定义相同。

本软件采用控制点或插值点两种方式来定义样条曲线。图 7-1 由控制点与插值点定义的样条曲线中红色点为插值点，黑色点为控制点。

图 7-1 由控制点与插值点定义的样条曲线



说明
使用插值点定义的样条曲线会自动生成控制点。如果您需要查询插值点样条曲线的控制点信息，需要调用与控制点样条曲线相关的查询接口。

此外，要想完整地定义一条样条曲线，应用程序还需要指定曲线的阶数与周期性等。关于定义控制点样条曲线与插值点样条曲线的详细信息，请参见 7.2 定义控制点样条曲线与 7.3 定义插值点样条曲线。

7.1 本软件对样条曲线的操作

对样条曲线来说，应用程序可以执行以下操作：

- 使用合理的数据组合来创建一条样条曲线，通过不同的定义方式将样条曲线分为控制点样条曲线与插值点样条曲线。

- 更新现有样条曲线的数据，比如样条曲线的阶数、控制点权重等。
- 查询现有样条曲线的数据。
  应用程序可以选择返回哪些数据。例如，对于由控制点或插值点定义的样条曲线，可以在修改样条曲线后返回点的当前位置。
- 在指定参数处求解控制点样条曲线。
- 获取样条曲线的参数范围。

涉及的接口：**addGeometry**，**addGeometry**，**evaluateCurve**，**setGeometryData**，**getGeometryData**，**getGeometryData**，**getSplineParameterLimits**。

7.2 定义控制点样条曲线

本软件支持应用程序使用控制点定义样条曲线。

定义一条完整的控制点样条曲线需要以下参数：

- 曲线的阶数。
- 曲线的周期性。
  周期曲线是闭合的，虽然参数值会在曲线上的某个位置发生“跳转”，但是仍然没有“开始”与“结束”。曲线在该位置的一阶导数连续。
- 控制点数组，控制点的数量必须大于阶数加 1。控制点可以是原本模型中已存在的点，也可以是本软件在定义样条曲线时根据选择的位置新增的点。
- 控制点权重数组，控制点权重数量等于控制点的数量。控制点权重能够影响曲线与控制点之间的紧密程度。当 NURBS 中的控制点权重全部为 1 时，NURBS 将退化成 B-spline。当 NURBS 中的控制点权重一致但不为 1 时，此时的 NUBRS 则可以视为 B-spline 的仿射。
- 节点数组，用于控制曲线的参数化，单个节点使用一个浮点数来表示。
- 插值点数组，位于曲线上的点，该参数为可选项。
- 参数值数组，用于指示曲线上的插值点位置，数组大小与插值点数组大小相同，该参数为可选项。

应用程序可以通过传递以上数据，在本软件中定义样条曲线。此外，本软件能够准确读取以任何方式定义的样条曲线数据，有助于用户将一个应用程序中的样条曲线导入另一个应用程序。

7.3 定义插值点样条曲线

穿过一系列位置的 NURBS 曲线被称为插值点样条曲线。

定义一条完整的插值点样条曲线需要以下参数：

- 曲线的阶数。
- 曲线的周期性，与控制点样条曲线类似。

- 插值点数组，位于曲线上的点的数组。插值点的数量不能小于阶数加 1。插值点可以是原本模型中已存在的点，也可以是本软件在定义样条曲线时根据选择的位置自动新增的点。
- 插值点样条曲线的参数化方法，目前支持弦长参数化与向心参数化两种方式，默认选择弦长参数化。

如果使用插值点来生成样条曲线，本软件将自动计算满足条件的控制点位置，并且在这些位置添加点。所以无论是使用控制点还是使用插值点来定义样条曲线，应用程序均能够查询到控制点的相关信息，如果您需要查询插值点样条曲线的控制点信息，需要调用与控制点样条曲线相关的 **getGeometryData** 接口。

7.4 周期性样条曲线

对于周期性样条曲线来说，本软件通过增加节点的数量来实现“连接”处的连续性，从而使它们环绕。

对于插值点样条曲线来说，应用程序不必传入节点与控制点，本软件会自动生成节点来进行环绕；对于控制点样条曲线来说，应用程序必须直接添加节点数组来实现环绕。

在控制点样条曲线中：

节点个数：knot_num

节点数组：knots

曲线阶数：degree

控制点个数：cp_num

- 非闭合曲线：knot_num = degree + cp_num + 1
- 闭合曲线：knot_num = degree*2 + cp_num + 1

前（degree+1）个节点之间的间隔应该与第（cp_num+1）个到第（degree+cp_num+1）个的节点之间的间隔相等。最后（degree+1）个节点应该与第（degree+1）个到（2*degree+1）的节点之间的间隔相等。

以下为您举例说明：

degree	2
cp_num	4
knot_num	9

knots	a1	a2	a3	a4	a5	a6	a7	a8	a9

前（degree+1）个节点之间的间隔应该与第（cp_num+1）个到第（degree+cp_num+1）个的节点之间的的间隔相等。
前（degree+1）个节点，也就是前3个节点a1，a2，a3；
第（cp_num+1）个到第（degree+cp_num+1）个的节点，也就是第5个到第7个节点a5，a6，a7；
也就是 a2 - a1 = a6 - a5，a3 - a2 = a7 - a6。

最后（degree+1）个节点应该与第（degree+1）个到（2*degree+1）的节点之间的间隔相等。
最后（degree+1）个节点，也就是最后3个节点a7，a8，a9；
第（degree+1）个到（2*degree+1）的节点，也就是第3个到第5个节点a3，a4，a5；
最后（degree+1）个节点应该与第（degree+1）个到（2*degree+1）的节点之间的间隔相等。
也就是 a9 - a8 = a5 - a4，a8 - a7 = a4 - a3。

您可以使用以下方式来计算前后 degree 个节点的值：

```
period = knots[cp_num + degree] - knots[degree];
for(int i=0; i<degree; i++)
{
    knots[i] = knots[cp_num + i] – period
    knots[cp_num + degree + i + 1] = knots[degree + i + 1] + period
}
```

7.5 样条曲线的示例

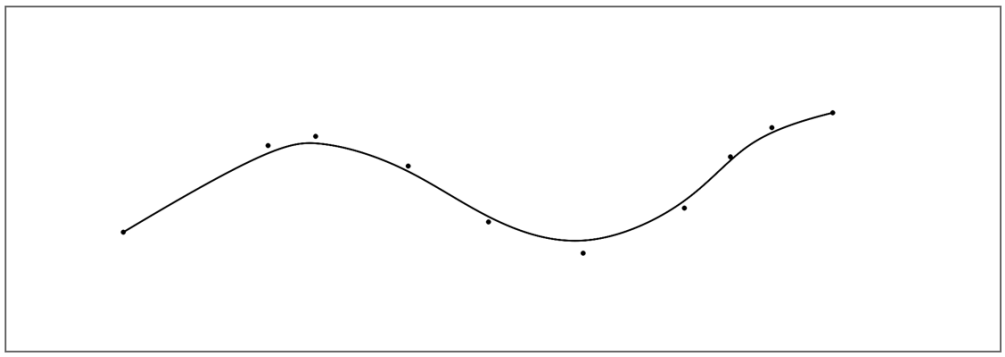
本节将为您介绍一些可以添加到本软件的简单样条曲线及其结构体。

7.5.1 控制点样条曲线

本小节主要介绍一条简单控制点样条曲线及 PSCSApiSpline2D 结构体，该示例中不包含插值点数组及其参数值，两者均为可选参数。

图 7-2 控制点样条曲线展示了一条非周期性的、非理性的、阶数为 3 及有 10 个控制点的样条曲线。

图 7-2 控制点样条曲线



输入到 PSCSApiSpline2D 结构体中的值包括：

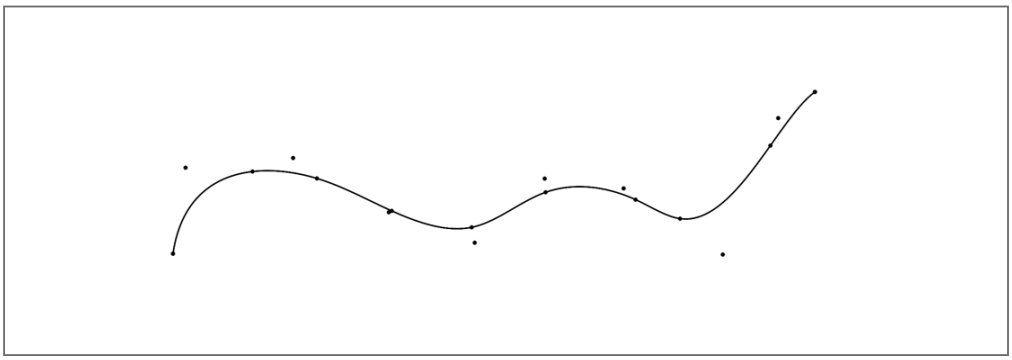
- Degree = 3
- isPeriodic = false
- Points = [p1...p10]
- Weights = [1,1,1,1,1,1,1,1,1,1]
- Knots = [0,0,0,0,1,2,3,4,5,6,7,7,7,7]

7.5.2 插值点样条曲线

本小节主要介绍一条简单插值点样条曲线及 PSCSApiInterpolatedSpline2D 结构体。

图 7-3 插值点样条曲线展示了一条非周期性的、非理性的、阶数为 3 及有 10 个插值点的样条曲线。

图 7-3 插值点样条曲线



输入到 PSCSApiInterpolatedSpline2D 结构体中的值包括：

- Degree = 3
- isPeriodic = false
- InterpolatedPoints = [p1...p10]
- Parameterisation = CHORD_LENGTH

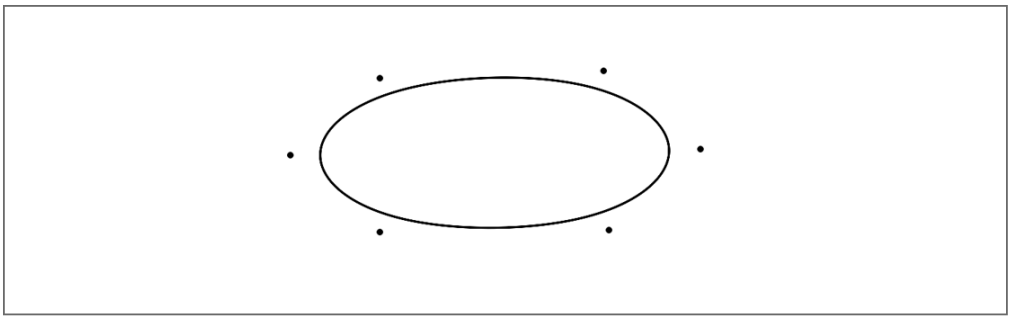
本软件会根据现有的数据进行计算，生成控制点数组、控制点权重数组以及节点数组。

7.5.3 周期性控制点样条曲线

本小节主要介绍一条简单的周期性控制点样条曲线及 PSCSApiSpline2D 结构体，该示例中不包含插值点数组及其参数值，两者均为可选参数。

图 7-4 周期性控制点样条曲线显示了一个周期性控制点样条曲线。该曲线通过 6 个控制点来定义，这 6 个点可以是是原本模型中已存在的点，也可以是本软件在定义曲线时根据选择的位置自动新增的点。

图 7-4 周期性控制点样条曲线



输入到 PSCSApiSpline2D 结构体中的值包括：

- Degree = 3
- isPeriodic = true
- Points = [p1...p6]
- Weights = [1,1,1,1,1,1]
- Knots = [1,2,3,4,5,6,7,8,9,10,11,12,13]

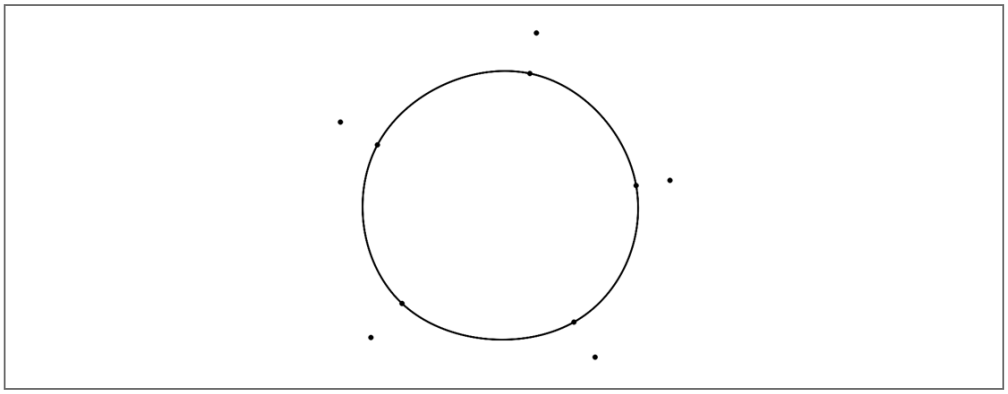
周期性控制点样条曲线的标志 isPeriodic 被设置为 true。

7.5.4 周期性插值点样条曲线

本小节主要介绍一条简单的周期性插值点样条曲线及 PSCSApiInterpolatedSpline2D 结构体。

图 7-5 周期性插值点样条曲线显示了一个周期性插值点样条曲线。该曲线穿过了 5 个插值点，这 5 个点可以是是原本模型中已存在的点，也可以是本软件在定义曲线时根据选择的位置新增的点。

图 7-5 周期性插值点样条曲线



输入到 PSCSApiInterpolatedSpline2D 结构体中的值包括：

- Degree = 3
- isPeriodic = true
- InterpolatedPoints = [p1...p5]
- Parameterisation = CHORD_LENGTH

本软件会根据现有的数据进行计算，生成控制点数组、控制点权重数组以及节点数组。在曲线上的点为插值点，未在曲线上的点为控制点，第一个和最后一个插值点不需要重合。周期性的插值点样条曲线的标志 isPeriodic 被设置为 true。

7.6 求解样条曲线

大多数应用程序期望独立于本软件对曲线执行操作。例如独立于本软件绘制一条样条曲线。这些操作取决于是否能找到给定参数值在曲线上对应的位置和导数。

应用程序能够通过本软件查询基本的样条曲线数据来使用自己的独立功能。此外，本软件还能为应用程序提供以下信息：

- 曲线的参数范围。
- 曲线上任何导数不连续点的数量和位置。
- 参数值在曲线上的位置和导数。对于不连续的曲线，应用程序可以指定返回不连续点某一侧的数据。

涉及的接口：**evaluateCurve**。

8 圆锥曲线

圆锥曲线是由一个二次锥面与一个平面相交形成的曲线。这些曲线有些是开放的，无界的，例如双曲线和抛物线；有些是闭合的，比如椭圆（圆是特殊的椭圆）。

定义

说明
本软件中，圆锥曲线仅表示其整个无界曲线的有界部分。

本软件中的圆锥曲线由两个端点、一个控制点以及一个形状参数（rho）来定义。两个端点与控制点形成的直线分别与圆锥曲线相切，即控制点位于在两个端点处与曲线相切的两条直线的交点处。

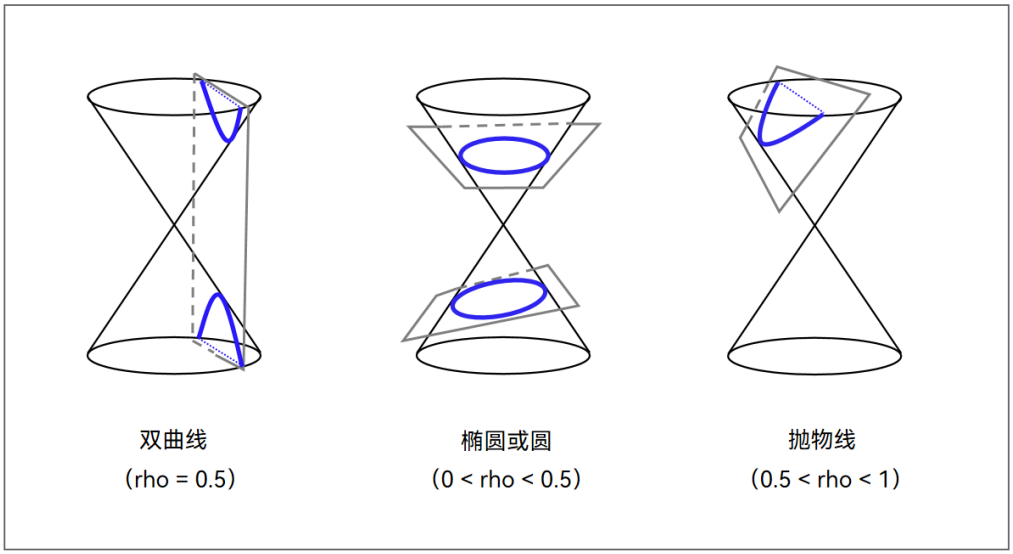
圆锥曲线的标准方程

Ax^2 + By^2 + Cxy + Dx + Ey + G = 0

圆锥曲线的形状参数 ρ（rho）

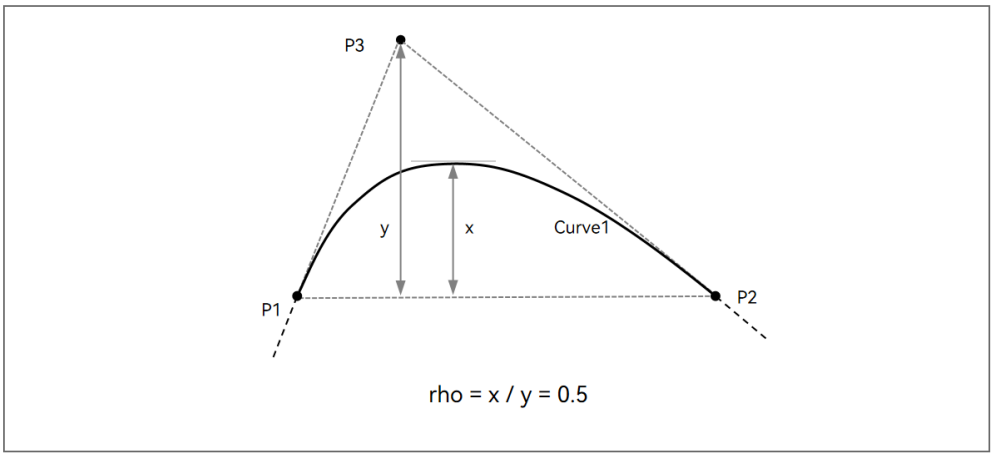
- rho 是一个定义圆锥曲线形状的参数，取值范围为（0,1）。
- rho 的取值决定了圆锥曲线的形状，如双曲线、抛物线或椭圆，如图 8-1 抛物线的 rho 值所示。
  - 0 < rho < 0.5：圆锥曲线表现为一条椭圆形曲线。
  - rho = 0.5：圆锥曲线表现为一条抛物线。
  - 0.5 < rho < 1：圆锥曲线表现为一条双曲线的单侧曲线。

图 8-1 抛物线的 rho 值



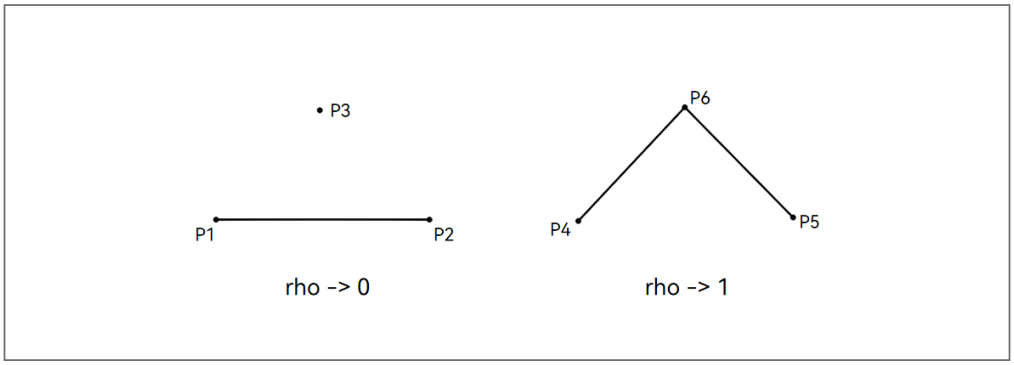
- rho 的计算公式为 rho=x/y，其中 x 是曲线上最远点到端点 P1 与 P2 所在直线的垂直距离，y 是指控制点 P3 到 P1 与 P2 所在直线的垂直距离，如下图所示。

图 8-2 抛物线的 rho 值



- rho 的值越大，曲线的曲率越大；相反，rho 的值越小，曲线曲率越小。例如添加一条椭圆弧，如果 rho 值无限接近 0，将产生极端曲率。
  当 rho 的值趋近于 0 时，曲线变得平坦并近似为一条直线。当 rho 的值趋近于 1 时，曲线近似由两个线性部分组成，在控制点处存在切线不连续的情况。如图 8-3 rho 无限趋近于 0 或 1 所示，P1、P2、P4、P5 表示端点，P3、P6 表示控制点。

图 8-3 rho 无限趋近于 0 或 1



圆锥曲线界限的影响因素

- 曲线必须位于本软件设定的线性容差、角度容差及包围盒内，否则无法准确显示。
- 曲线必须是连续的。由于连续性要求曲线在连接点处平滑过渡，而双曲线的两个分支在连接点处是不连续的。因此本软件只能表示双曲线两个分支中的其中一个。
- 过曲线端点的切线必须以特定方式相交。意味着在椭圆形的圆锥曲线上，有界的圆锥曲线长度将始终小于整个椭圆的一半。因为在椭圆形的圆锥曲线上，如果两个点到中心点构成的弧度差为 π（即半个圆周），那么在这两个点处的切线将平行，两条切线不相交，无法定义圆锥曲线的控制点。如果两个点到中心点构成的弧度差大于 π，切线将沿曲线定义的方向发散。

8.1 圆锥曲线相关操作

本小节介绍了应用程序可以使用本软件对圆锥曲线执行的一系列操作。

- 通过 **addGeometry** 使用任何合理的数据组合来创建一条圆锥曲线。
- 使用 **getGeometryData** 查询现有曲线的数据。应用程序可以选择查询部分数据，例如只查询圆锥曲线的当前 rho 值。
- 使用 **setGeometryData** 更新圆锥曲线的部分数据，同时保持其他数据不变。
- 使用 **getConicExtraData** 查询圆锥曲线的附加数据，比如圆锥曲线的方向、焦点、顶点和离心率。
- 使用 **transform** 刚性变换圆锥曲线。
- 使用 **evaluateCurve** 在指定参数处计算所需的曲线数据，如当前参数处的曲线坐标以及曲线导数。

8.2 NURBS 表示法

一条圆锥曲线可以被定义为特殊的 NURBS 曲线。

NURBS 表示的是一条非均匀有理的二次贝塞尔曲线，使用 NURBS 来表示圆锥曲线属性如下：

- degree：2
- 3 control points
- knots：[0, 0, 0, 1, 1, 1]
- weights：[1, rho / (1 - tho), 1]

虽然上述形式的刚性曲线可以直接使用本软件的样条曲线相关接口来添加，但是这种刚性曲线不适合表示灵活的圆锥曲线，因为在圆锥曲线中决定形状的 rho，在样条曲线中不被视为一个自由度。除此之外，如果使用样条曲线接口定义圆锥曲线，本软件不会识别到圆锥曲线的附加属性，如焦点、方向与离心率等。

本软件中，所有参数化曲线都是具有方向性的几何元素。上文强调过，本软件中的圆锥曲线仅表示其整个无界曲线的有界部分。圆锥曲线的方向与其整个无界曲线的对称轴平行，方向从曲线的顶点到焦点，通常就是几何元素的开口方向。顶点是整个无界曲线的对称轴与曲线本身的交点。圆锥曲线的方向与控制点的位置和 rho 值密切相关。

- 圆锥曲线对称时：
  - 控制点与抛物线（双曲线）的两个端点等距，两个端点的对称轴与圆锥曲线的方向平行。
  - 控制点与椭圆的两个端点等距，两个端点的对称轴与圆锥曲线的方向平行或垂直，平行或垂直由 rho 的值来决定。
- 圆锥曲线不对称时，其方向是关于控制点位置与 rho 的更复杂的函数。

圆锥曲线定义的椭圆方向在下列情况中不连续：

- 对于一个非圆的椭圆来说，当控制点从两个端点的对称轴一侧翻转到另一侧时，椭圆的方向符号也随之翻转。
- 当 rho 在（0,0.5）的范围内取特定值时，圆锥曲线可能是一个圆的某一部分。由于圆的方向是不确定的，因此当圆锥曲线表示为圆的某一部分时，圆锥曲线的方向是顶点到两个端点的中点的方向。当 rho 的变化使圆锥曲线的表现从圆形变为椭圆形，或从椭圆形变为圆形时，方向可能不连续。

8.3 圆锥曲线支持的几何约束

圆锥曲线支持所有参数化曲线支持的几何约束。

- 竖直约束，水平约束，垂直约束，平行约束，角度约束。
  这些约束作用于圆锥曲线的方向。圆锥曲线的方向与其整个无界曲线的对称轴平行，有关圆锥曲线方向的详细信息，请参见上一小节 8.2NURBS 表示法。
- 对称约束，重合约束。
  无论圆锥曲线的形状是椭圆、抛物线或双曲线，两两之间均能添加对称约束与重合约束。
- 相切约束，法线约束。
- 阵列约束。
- rho 约束，可以直接应用于圆锥曲线的 rho 参数。

理论上椭圆圆锥曲线包括长轴半径与短轴半径。但是本软件不支持对椭圆圆锥曲线添加长轴半径约束与短轴半径约束。

8.4 查询圆锥曲线数据

圆锥曲线是通过三个控制点和一个 rho 参数的贝塞尔曲线定义的，本软件可以查询定义圆锥曲线时的基础数据，以及本软件计算出的数据。

使用 **getGeometryData** 获取定义圆锥曲线时的基本信息，包括三个控制点的值以及形状参数 rho。

使用 **getConicExtraData** 获取本软件计算出的圆锥曲线的附加信息，包括圆锥曲线的方向、焦点、顶点和离心率。

- 离心率：圆锥曲线上任意点到焦点的距离与该点到准线的（最小）距离之比。
- 焦点：一般圆锥曲线有两个焦点，但由于贝塞尔曲线表示的部分不超过整条无界曲线的一半，因此本章提及的焦点都指近焦点。
- 顶点：顶点是整条无界曲线最靠近准线的点，也是对称轴与圆锥曲线的交点。该点位置可能超出本软件圆锥曲线的有界区域。
- 方向：从曲线准线到焦点的方向。因此，它垂直于准线，也是从顶点到焦点的方向。

8.5 求解圆锥曲线

大多数应用程序期望独立于本软件对曲线执行操作。例如独立于本软件绘制一条圆锥曲线。这些操作取决于是否能找到给定参数值在曲线上对应的位置和导数。

应用程序能够通过本软件查询圆锥曲线数据来使用自己的独立功能。此外，本软件还能为应用程序提供以下信息：

- 曲线的参数范围。
- 曲线上任何导数不连续点的数量和位置。
- 参数值在曲线上的位置和导数。对于不连续的曲线，应用程序可以指定返回不连续点某一侧的数据。

涉及的接口：**evaluateCurve**。

9 偏移曲线与复制曲线

9.1 偏移曲线

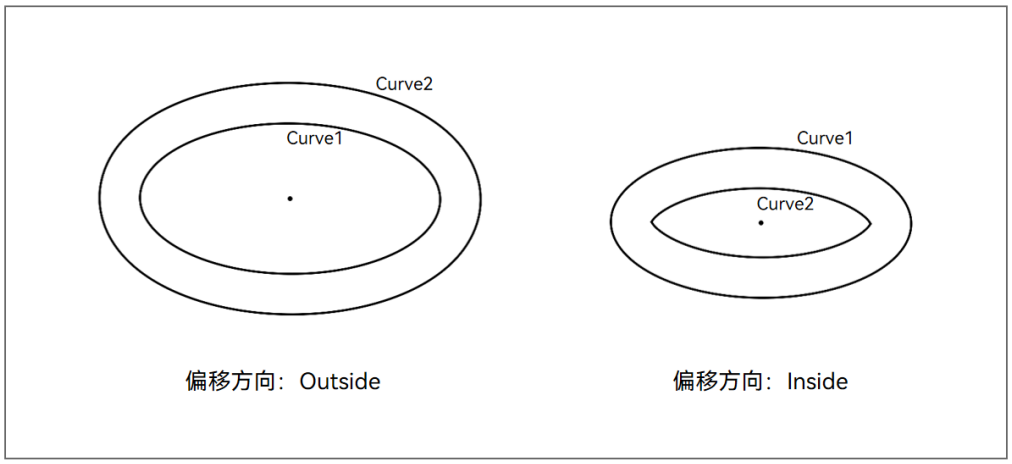
偏移曲线是基于任一椭圆或参数化曲线偏移一定距离后的曲线。

用于定义偏移曲线的曲线称为偏移父曲线，本节中称其为父曲线。本软件支持从一条偏移曲线创建另一条偏移曲线，即一条偏移曲线可以是另一条偏移曲线的父曲线。最原始的椭圆或参数化曲线被称为偏移根曲线，本节中称其为根曲线。

特性

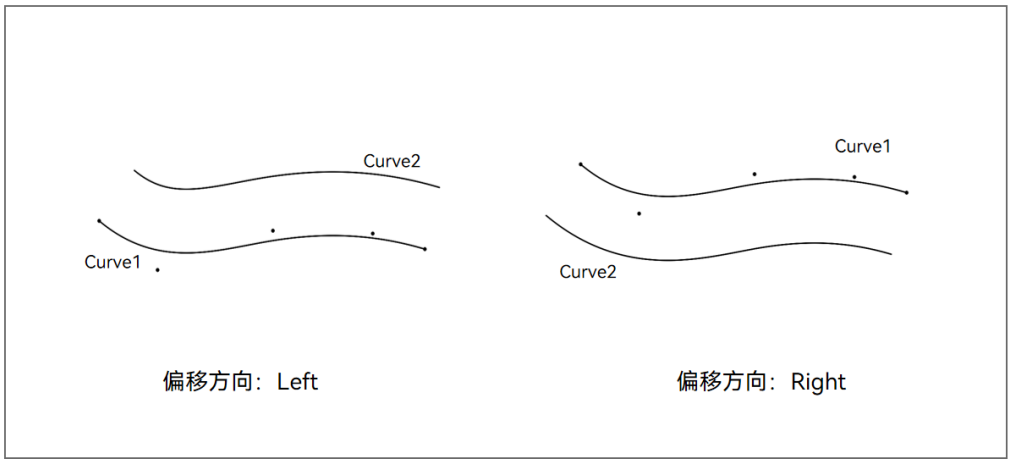
- 定义偏移曲线需要父曲线、偏移方向、偏移距离三个要素。
- 偏移曲线与父曲线之间的任意对应位置始终保持等距。
  - 取偏移曲线与父曲线上相对应的两点 P1 与 P2，P1 与 P2 构成的向量是 P2 的法线向量。
  - 取偏移曲线与父曲线上相对应的两点 P1 与 P2，P1 与 P2 之间的距离等于偏移距离。
- 偏移曲线与父曲线的相对位置通过偏移方向和父曲线的方向来进行区分。
  - 对椭圆来说，偏移方向只能选择 Inside 或 Outside，如图 9-1 椭圆的偏移方向所示，图中 Curve1 为父曲线，Curve2 为偏移曲线。

图 9-1 椭圆的偏移方向



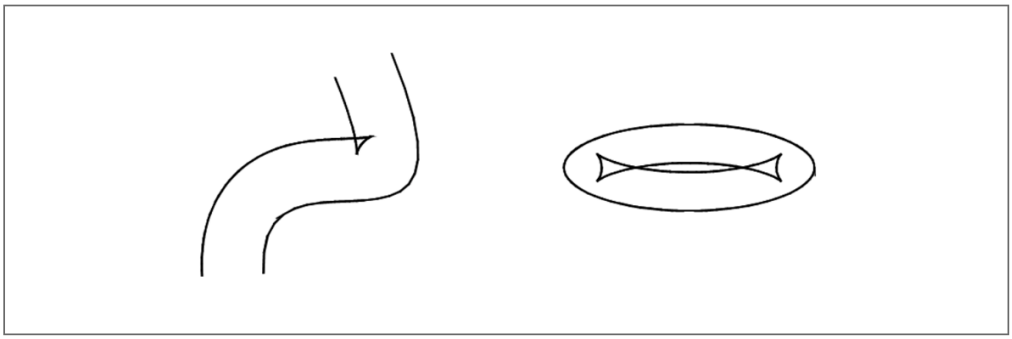
— 对参数化曲线来说，偏移方向只能选择 Left 或 Right。Left 表示偏移曲线在父曲线方向的左侧，Right 表示偏移曲线在父曲线的右侧。图 9-2 控制点样条曲线的偏移方向以控制点样条曲线 Curve1 作为父曲线，Curve2 为偏移曲线。

图 9-2 控制点样条曲线的偏移方向



- 偏移曲线的方向与父曲线的方向保持一致。
- 通常情况下，偏移曲线与父曲线的形状类似。但是当偏移距离过大时，偏移曲线会出现自交且不连续的情况，如图 9-3 偏移曲线自相交所示。

图 9-3 偏移曲线自相交



- 父曲线与偏移曲线之间的形状相互影响。当偏移曲线（父曲线）的形状发生改变时，父曲线（偏移曲线）的形状也随之改变。

- 当删除一条曲线时，只要从该曲线偏移出了其他曲线，求解器将会自动删除从该曲线偏移出的所有曲线。
- 如果没有在偏移曲线与父曲线之间添加距离约束，应用程序可以使用拖拽来动态地修改两者之间的偏移距离。

偏移曲线支持的几何约束

- 偏移曲线与其他非偏移曲线元素之间通常情况下支持所有曲线相关的几何约束。
- 两条具有相同根曲线的偏移曲线之间支持以下几何约束：
  - 距离约束
  - 重合约束
  - 等距离约束
- 两条具有不同根曲线的偏移曲线之间支持以下几何约束：
  - 平行约束
  - 垂直约束
  - 角度约束
  - 相切约束
  - 法线约束
  - 距离约束

9.2 复制曲线

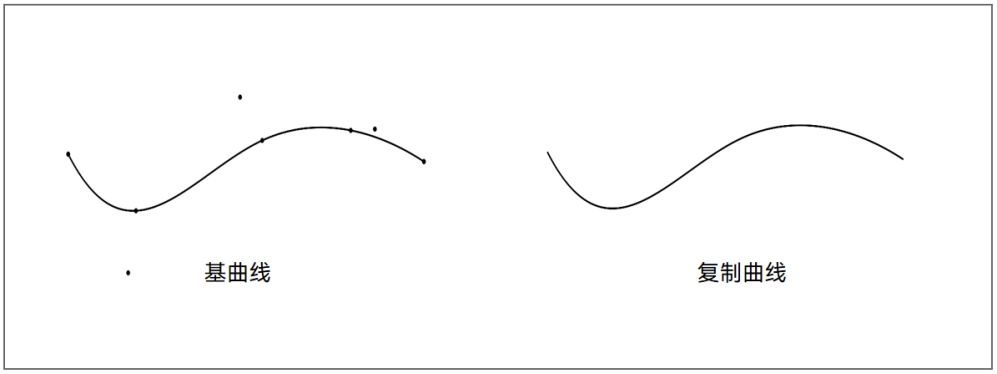
复制曲线是基曲线的精确副本，即根据仿射变换矩阵对基曲线进行变换后生成的参数化曲线。用于定义复制曲线的曲线称为基曲线。

基曲线的类型包括偏移曲线、样条曲线、圆锥曲线、自定义回调曲线和自定义参数曲线。注意，复制曲线不能作为基曲线。

特性

- 定义复制曲线需要基曲线和仿射变换矩阵两个要素。
- 基曲线与复制曲线之间的变换只支持平移和旋转，不支持缩放与对称等。
- 复制曲线跟基曲线的形状一致，但是局部坐标系位置，方向或角度的相对关系取决于仿射变换参数。
- 复制曲线和基曲线的形状相互影响，并且始终保持一致。
- 能够在基曲线上添加的几何约束同样适用于复制曲线，但是已经关联到基曲线的几何约束和子元素不会复制到复制曲线上，例如插值点样条曲线的插值点以及他们之间的重合约束不会被复制到复制曲线上，如图 9-4 插值样条曲线与其复制曲线所示，复制曲线并没有复制基曲线上的点、控制点、及五个点与曲线的重合约束。

图 9-4 插值样条曲线与其复制曲线



- 当删除基曲线时，求解器将会自动删除从该基曲线复制出的所有复制曲线。
- 拖拽基曲线或复制曲线时，求解器会根据两者的相对位置自动更新仿射变换矩阵。

复制曲线支持的几何约束

- 基曲线支持添加的几何约束都适用于复制曲线。
- 如果两条复制曲线都基于同一条基曲线，那么这两条复制曲线无法支持对称约束。

10 自定义曲线

本软件直接支持的几何元素有点、线（直线和线段）、圆、椭圆、样条曲线和圆锥曲线。除了这些基本几何元素类型之外，如果应用程序还需要其他类型的曲线，可以使用本软件提供的工具创建和使用自定义曲线。

本软件支持的自定义曲线有以下两种类型：

- 自定义参数曲线：应用程序通过变量、方程（不等式）、局部坐标系（LCS）的原点及方向来添加自定义参数曲线（PSCSApiOffsetCurve2D）。
- 自定义回调曲线：应用程序通过变量、回调函数、局部坐标系（LCS）的原点及方向来添加自定义回调曲线（PSCSApiCallbackCurve2D）。

自定义曲线的方向是局部坐标系（LCS）绕原点的旋转参数，与样条曲线和圆锥曲线的方向相比，前者由本软件定义，应用程序来分配；后者是本软件自动计算得出。此外，方向向量的模长必须大于线性容差，如果应用系统未设置容差值，将参考默认线性容差，即方向向量的模长必须大于 1.0e-8。

本章将为您介绍以上两种类型的自定义曲线以及应用程序如何使用它们。两者最大的不同在于自定义参数曲线是本软件直接提供给应用程序的自定义曲线类型；而自定义回调曲线需要应用程序来定义其回调函数。

🕮 说明
在本软件中，应用程序可以使用自定义回调曲线来表示样条曲线。但是，建议尽可能地使用本软件直接提供的样条曲线。

10.1 自定义参数曲线

在本软件中，自定义参数曲线表示为 PSCSApiEvaluatedCurve2D 类型。

如果应用程序想要创建一条自定义参数曲线，需要向本软件传递以下信息：

- x：曲线的 X 分量的函数 f(t)。
- y：曲线的 Y 分量的函数 g(t)。
- variables：用于定义函数的变量描述符列表。当曲线的形状与其他节点之间存在某种关联时，应用程序需要将节点对应的变量描述符加入该列表。变量描述符包括

节点所属几何元素的 ID，节点的类型以及名称。这里的节点是指几何元素的参数变量、尺寸约束变量及自由变量。例如，一条自定义参数曲线的 X 分量是一个圆半径的倍数，假设该圆的半径的变量名为 R1，此时应用程序传递给本软件的 x 表达式中应该包括 R1，同时将 R1 对应的变量描述符加入到 variables 中。

- origin：局部坐标系（LCS）的原点。
- direction：局部坐标系（LCS）绕原点的旋转参数。应用程序可以使用三种方式来更新自定义参数曲线的方向，分别是使用本软件的 **setGeometryData** 接口，添加平行约束或角度约束。

x、y 与 variables 主要用于控制自定义参数曲线的形状，origin 与 direction 主要控制自定义参数曲线的具体位置。x 与 y 的字符串必须是合法的表达式。

如果应用程序需要获取某个参数在曲线上对应的具体位置等信息，需要调用 **evaluateCurve** 接口，本软件会根据应用程序给出的自定义参数曲线 ID、某个具体参数与导数的阶数来进行求解，如果导数的阶数为 2，该接口将输出该具体参数对应的具体位置、一阶导数与二阶导数。

10.2 自定义回调曲线

在本软件中，自定义回调曲线表示为 PSCSApiCallbackCurve2D 类型。

如果应用程序想要创建一条自定义回调曲线，需要向本软件传递以下信息：

- variables：用于定义函数的变量描述符列表。当曲线的形状与其他节点之间存在某种关联时，应用程序需要将节点对应的变量描述符加入该列表。变量描述符包括节点所属几何元素的 ID，节点的类型以及名称。这里的节点是指几何元素的参数变量、尺寸约束变量及独立变量。
- callback：应用程序提供的回调函数。在 PSCSCallbackCurveFunction 的定义中，应用程序能够根据一个给定的参数值，返回参数值在该曲线上对应的一个位置、一阶导数和二阶导数等的近似值。本软件可以调用用户定义的 PSCSCallbackCurveFunction 函数来获取这些信息。
- origin：局部坐标系（LCS）的原点。
- direction：局部坐标系（LCS）绕原点的旋转参数。与自定义参数曲线不同的是，应用程序只能调用 setCallbackCurveLCS 接口来更新自定义回调曲线的方向。

如果应用程序需要获取某个参数在曲线上对应的具体位置等信息，应用程序需要在回调函数的定义中给出相关功能，然后调用 evaluateCurve 接口，evaluateCurve 接口会根据应用程序定义的回调函数来进行求解。

11 变量、方程和不等式

应用程序可以使用变量、方程和不等式来表示模型中存在的关系。2D 求解器通过求解方程和不等式来找到变量的值。

11.1 变量

本软件支持以下三种变量类型：

- 普通变量：
  - Dimension：与尺寸约束绑定的变量，称为尺寸约束变量，变量的值被用作尺寸约束的大小。
  - Variable：与尺寸约束无关的变量，称为自由变量。
- 几何元素参数变量：
  - Radius：圆的半径。
  - MajorRadius：椭圆的长轴半径。
  - MinorRadius：椭圆的短轴半径。
  - X：点、圆心等的 X 坐标。
  - Y：点、圆心等的 Y 坐标。
  - W：齐次 W 坐标。
  - Distance：直线距坐标原点的距离参数。
  - Angle：直线、椭圆、自定义参数曲线、自定义回调曲线、复制曲线、刚性集合、可伸缩集合、单向可伸缩集合、双向可伸缩集合的角度参数。
  - PATTERN_VALUE：阵列几何元素中，一个参考元素的距离值，或两个参考元素的第一个距离值。
  - PATTERN_VALUE2：阵列几何元素中，两个参考元素的第二个距离值。
- 几何约束参数变量：
  - HELP_PARAMETER1：几何约束的第一个帮助参数。
  - HELP_PARAMETER2：几何约束的第二个帮助参数。
  - HELP_PARAMETER3：几何约束的第三个帮助参数。
  - HELP_PARAMETER4：几何约束的第四个帮助参数。

大多数尺寸约束的值可以用一个确切的值来表示，也可以用一个变量来表示。当为一个尺寸约束绑定一个变量时，与之关联的模型会增加一个额外的自由度。为了占用这

个自由度，应用程序可以使用一个方程来约束变量的值。本软件默认新添加一个尺寸约束时，该尺寸约束为刚性，为尺寸约束绑定变量后，变为非刚性。

通常，求解几何元素的位置是为了满足指定的几何约束，除此之外，它们还可用于求解方程。

说明
变量的取值范围为[-10^10,10^10]，本软件不会处理超出取值范围的值，以免发生精度问题。

涉及的接口：**addGeometryVariable**。

11.1.1 变量的初始值

所有变量都必须有一个初始值，以便区分非线性方程的多个根和欠约束模型的解决方案。在求解非线性方程时，本软件会使用变量的初始值作为求解的起点。

对于尺寸约束变量来说，本软件将尺寸约束的测量值作为变量的初始值；对于自由变量来说，应用程序必须提供初始值。

如果在变量的初始值处找不到方程的导数，本软件可能找不到最终的解决方案。例如，如果方程为 xy - 1 = 0，并且 x 与 y 的初始值都为 0，显然这种情况无法得出解决方案。要想本软件成功求解该方程，至少有一个变量的初始值非 0。

11.1.2 角度约束变量的初始值

本软件会将一个角度的当前值作为角度约束变量的初始值。默认情况下，角度的当前值取值在[0,2π]。

虽然本软件设置了角度约束变量的默认初始值，但是应用程序可以更改初始值。如果当前角度值与方程中实际需要的值相差 2π 的倍数，建议应用程序更改初始值。

以齿轮的转动举例，如果将一个当前超过 2π 倍数的角度值作为角度约束变量的初始值，那么在每次拖拽之前调用 **undoEvaluation** 接口，可能会导致角度约束的变量值在特定位置发生“跳跃”，相当于在初始值的基础上加上或减去 2π 的倍数。在实际应用当中，通常表现为齿轮突然急速转动较大的弧度。所以建议应用程序对角度约束变量的初始值进行规范地更改。

11.1.3 负值变量

当应用程序为一个尺寸约束指定一个负值时，如果值的正负对解决方案不会造成影响，那么本软件会取该值的绝对值来进行求解。如果尺寸约束与几何元素方向或位置有关，如约束方位，此时负值会导致不同的解决方案。

对尺寸约束变量来说，即使该尺寸约束与几何元素方向或位置无关，也能得到一个负值。例如，一个模型中包含一个尺寸约束变量 v1，v1 被加入到方程 v1 + 10 = 0 中，由此得出 v1 的值为-10。这种做法会使应用程序中的操作更加复杂，因此，对于尺寸约束变量，强烈建议应用程序使用带符号的距离来明确指定变量符号的意义。

在求解过程中，尺寸约束变量可能会从正值变为负值（或反之）。为了让模型能够稳定地变化，本软件会将尺寸约束变量的符号变化视为约束方位的变化。例如，如果一个点到线的距离约束绑定了一个变量，并且这个变量的值从+10 变为-10，实际结果与更改约束方位的解决方案一致，点从线的一侧移动到另一侧。

11.2 方程和不等式

本软件支持以表达式与回调函数两种形式来添加方程和不等式。如果应用程序需要使用回调函数的形式来添加方程或不等式，应用程序必须将变量数组、变量数组的长度以及要赋值的导数数组作为回调函数的输入参数。本软件提供的两种形式均支持线性与非线性的方程（不等式）类型。

本节介绍中，我们将使用 v 来表示变量，其中包括普通变量与几何元素参数变量，有关变量的分类，请参见 2.3 变量、方程和不等式。

涉及的接口：**addEquation**。

11.2.1 线性方程和不等式

线性方程形式如下：

a1*v1 + a2*v2 + a3*v3 + ... + c = 0

线性不等式形式如下：

- a1*v1 + a2*v2 + a3*v3 + ... + c <= 0
- a1*v1 + a2*v2 + a3*v3 + ... + c < 0
- a1*v1 + a2*v2 + a3*v3 + ... + c >= 0
- a1*v1 + a2*v2 + a3*v3 + ... + c > 0

其中 a1、a2、a3 为系数，v1、v2、v3 为变量，c 是一个常数。为了完整地定义一个线性方程或线性不等式，应用程序必须将具体的变量、系数以及常数传递到本软件中。

11.2.2 非线性方程和不等式

非线性方程形式如下：

f(v1, v2, v3...) = 0

非线性不等式形式如下：

- f(v1, v2, v3...) <= 0
- f(v1, v2, v3...) < 0
- f(v1, v2, v3...) >= 0
- f(v1, v2, v3...) > 0

其中 f 是变量 v1、v2、v3 等的任何函数。应用程序只需将变量与函数传递到本软件中，无需说明它们的具体使用方法。

对于一个非线性方程和不等式，本软件会根据当前给定的变量值来计算函数的值，该函数的值被称为残差（residual）。求解非线性方程和不等式通常采用迭代方式。每次迭代中，本软件都会计算当前变量值下的函数值（即残差），不断更新变量的值。这个过程会一直重复，直到残差满足某种收敛条件或达到最大迭代次数为止，从而无限逼近非线性方程和不等式的解。

应用程序需要确保变量的初始值足够接近期望的解，以便本软件对方程与不等式的求解能够收敛。一些变量的值可能无法计算出残差，例如，b = sqrt (a)。如果方程和不等式中涉及角度约束相关的变量或直线的角度参数变量，这些变量通常以弧度为单位。本软件支持任何大小的弧度值，因此变量对的值可能小于 0 或大于 2π。

应用程序在添加非线性方程和不等式时，还需要传入包含特定变量的函数的导数值。如果导数可用，有助于本软件更快速地找到解决方案。即使导数不可用，本软件仍然能够成功求解模型，只是求解速度相对较低。

如果非线性方程和不等式的阶数是线性的，解决方案将更可靠。例如，如果 a 和 b 都是距离约束变量，最好使用 a = 1000/b 方程进行求解，而不是 a*b = 1000。

注意，当 a <= f(v1,v2,...,g1,g2,...) <= b 时，需要使用两个不等式来使函数满足条件，分别是 f(v1,v2,...,g1,g2,...) - a >= 0 与 f(v1,v2,...,g1,g2,...) - b <= 0。

12 自动逻辑约束与自动尺寸约束

自动逻辑约束与自动尺寸约束的功能就是对约束系统中处于欠约束状态下但满足约束条件的几何元素自动添加相关的几何约束，使得几何元素在约束系统中的位置保持不变。约束条件是指几何元素之间的关系是否能够满足几何约束的容差，例如重合约束，如果两个点的位置差异满足线性容差，2D 求解器会自动为两点添加重合约束。自动逻辑约束和自动尺寸约束会按照用户传入的约束类型顺序来控制约束优先级。

自动逻辑约束与自动尺寸约束的应用主要有以下方面：

- 从外部导入模型时，如果模型处于欠约束状态，2D 求解器可以为其自动添加逻辑约束与尺寸约束。
- 使用草图绘图时，某些几何元素的定义可能不够准确，使用自动逻辑约束与尺寸约束可以更加精确地定义模型。
- 快速定义整个模型，后期用户再根据自身需求手动添加或删除几何约束。

在对自动逻辑约束与自动尺寸约束进行详细说明之前，先向您简单介绍一元约束、二元约束，以及多元约束的概念。在约束系统中，只需与一个几何元素关联的几何约束被称为一元约束，如与一条直线关联的竖直约束；需要与两个几何元素关联的几何约束称为二元约束，如在两个圆之间添加的相切约束。多元约束同理。

12.1 自动逻辑约束

当处于欠约束状态的模型被导入应用程序时，本软件提供了自动逻辑约束功能来快速定义整个模型，以满足用户需求。

自动逻辑约束支持的约束类型：

- 一元约束：水平约束、竖直约束。
- 二元约束：重合约束、同心约束、平行约束、垂直约束、相切约束、法线约束、等半径约束。
- 三元约束：对称约束、中点约束。
- 四元约束：等距离约束。

🕮 说明
添加对称约束与中点约束类型时，均需要三个几何元素。对称约束的对称轴直接在对称轴列表进行查询；中点约束的中点会自动从 targetGeometries 与 groundGeometries 的并集中获取。

在应用程序调用本软件的自动逻辑约束之前，需要先向本软件提供线性容差，角度容差、需要进行判断的几何约束类型以及几何元素的列表等，以下进行简要说明。

- 自动逻辑约束的线性容差（linearTolerance）：如果应用程序没有提供自动逻辑约束的线性容差，本软件将使用 2D 求解器的线性容差。
- 自动逻辑约束的角度容差（angleTolerance）：如果应用程序没有提供自动逻辑约束的角度容差，本软件将使用 2D 求解器的角度容差。
- 逻辑约束关联的几何元素列表（groundGeometries）：存储二元约束与三元约束中除对称轴以外的目标几何元素。
  - 与 targetGeometries 中的几何元素组合生成二元逻辑约束。
  - 与 targetGeometries 和 symmetryAxises 中的几何元素组合生成对称约束。

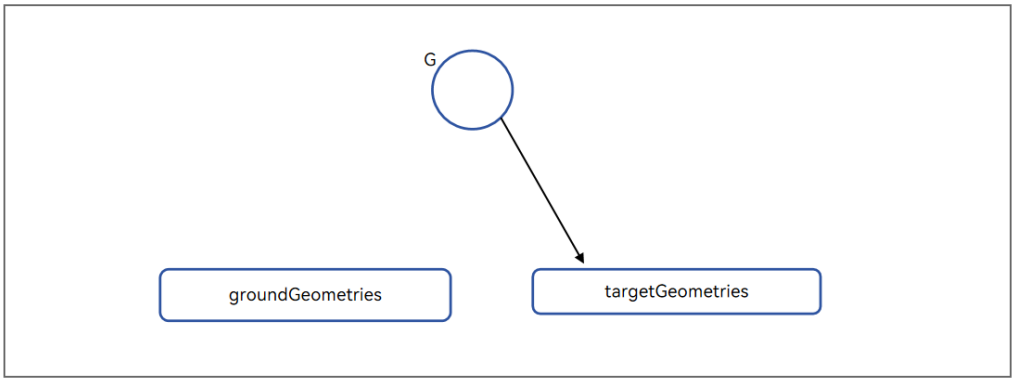
  具体组合方式，请见下方图示。
- 逻辑约束关联的几何元素列表（targetGeometries）：存储一元约束、二元约束及三元约束中除对称轴以外的目标几何元素。
  - 数组内部的几何元素自动生成一元约束或组合生成二元逻辑约束。
  - 与 symmetryAxises 中的几何元素组合生成对称约束。
  - 与 groundGeometries 中的几何元素组合生成二元逻辑约束。
  - 与 groundGeometries 和 symmetryAxises 中的几何元素组合生成对称约束。

  具体组合方式，请见下方图示。
- 对称轴列表（symmetryAxises）：当几何约束类型为三元约束中的对称约束时，用于存储对称约束的对称轴。当添加除对称约束以外的其他逻辑约束时，该列表无意义。
- 等距离约束关联的几何元素列表（equalDistanceFrom）：成对存储等距离约束关联的几何元素。
- 等距离约束关联的几何元素列表（equalDistanceTo）：成对存储等距离约束关联的几何元素。
- 自动逻辑约束的约束类型（constraintTypes）：本软件将根据列表中的几何约束类型来识别模型。

关于 groundGeometries、targetGeometries、symmetryAxises、equalDistanceFrom 与 equalDistanceTo 的具体存储方式，有以下几种组合：

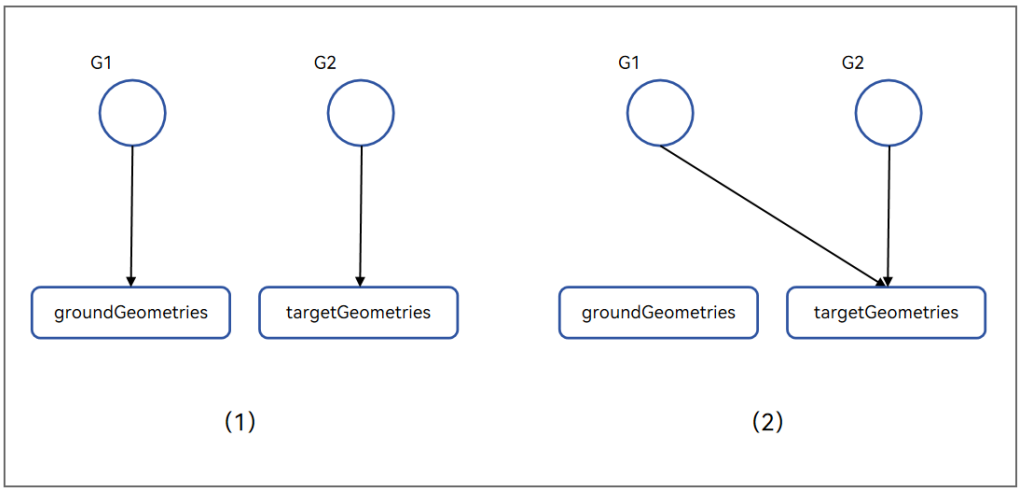
- 一元约束：几何元素 G 存储在 targetGeometries 中，如图 12-1 一元约束的传参方式所示。

图 12-1 一元约束的传参方式



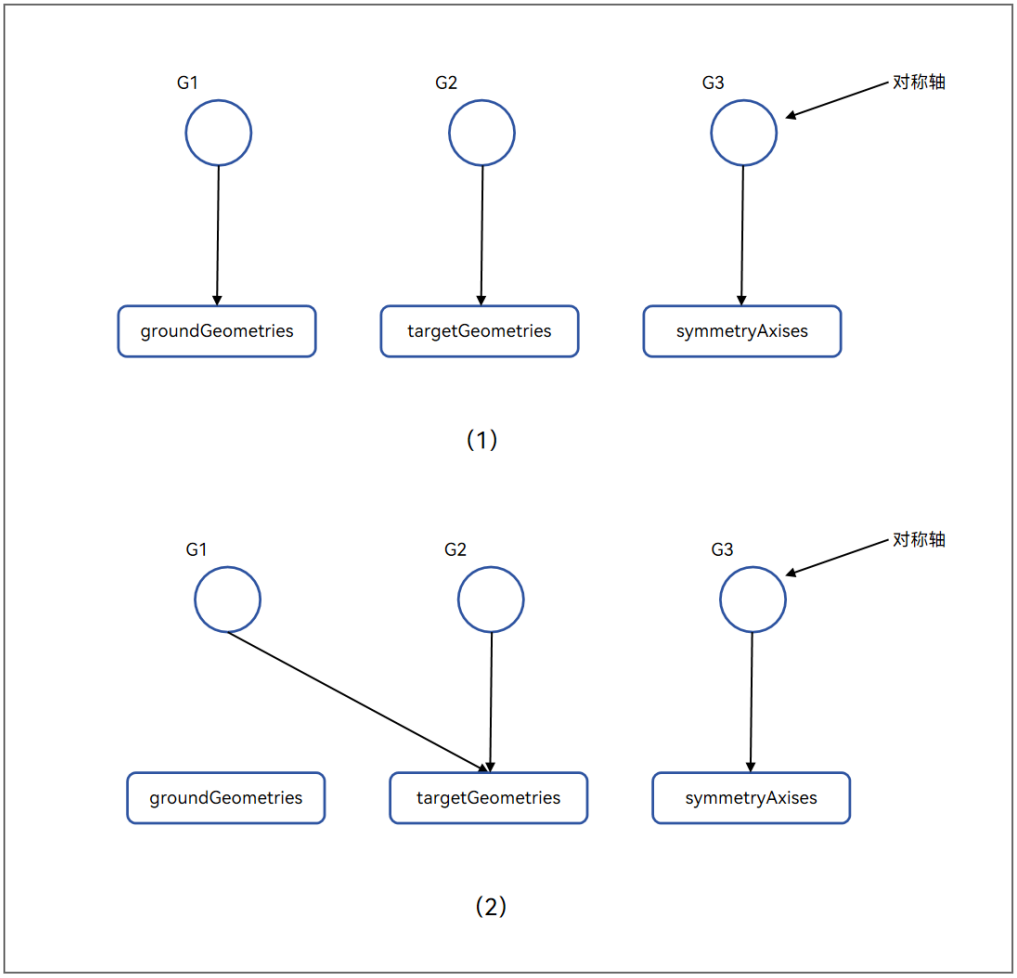
- 二元约束：如图图 12-2 二元约束的传参方式所示，推荐您使用图（1）中的传参方式。
  - 图（1）：两个几何元素 G1 与 G2 分别存储在 groundGeometries 与 targetGeometries 中。
  - 图（2）：两个几何元素 G1 与 G2 均存储在 targetGeometries 中。

图 12-2 二元约束的传参方式



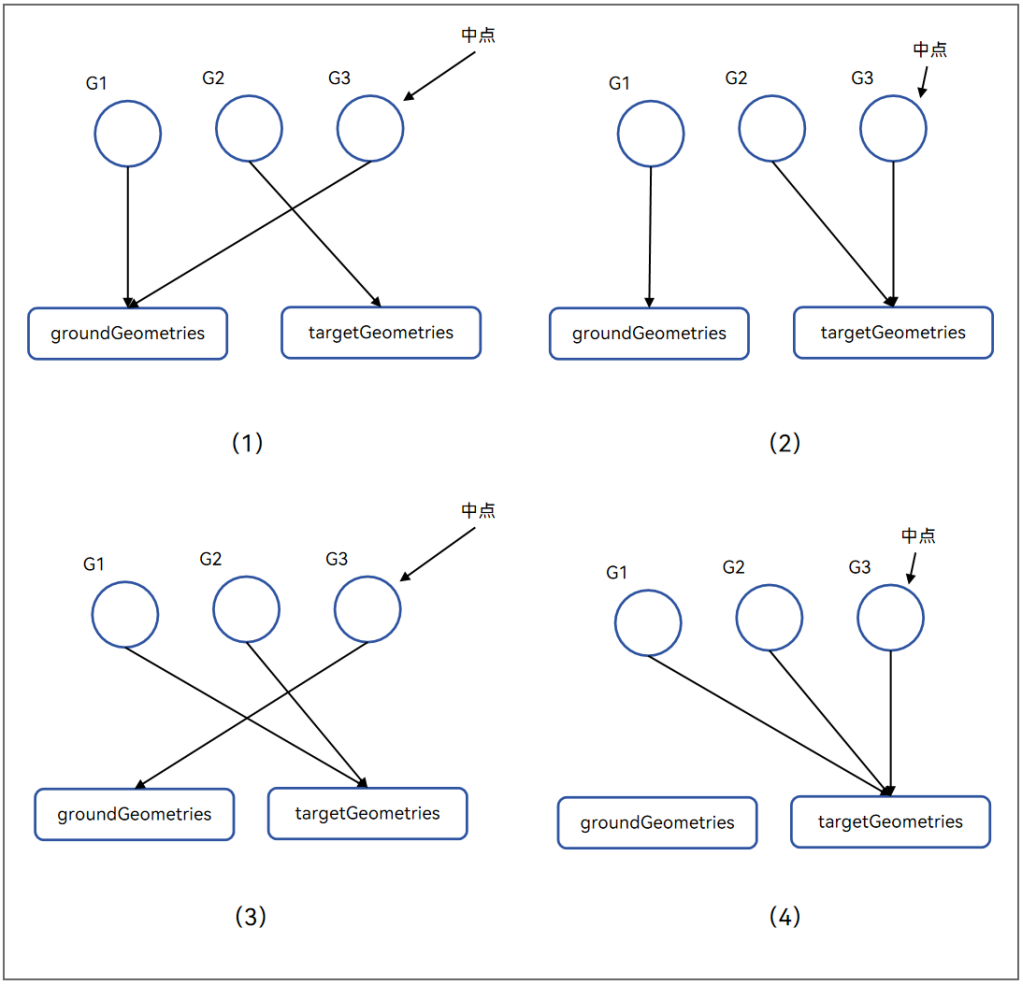
- 三元约束：自动逻辑约束中，三元约束包括对称约束与中点约束，对称约束关联的几何元素的传入规则如图 12-3 对称约束的传参方式所示，推荐您使用图（1）中的传参方式；中点约束关联的几何元素的传入规则如图 12-4 中点约束的传参方式所示，推荐您使用图（2）中的传参方式。
  - 图（1）：对称约束的对称轴 G3 存储在 symmetryAxises 中，其他两个几何元素 G1 与 G2 分别存储在 groundGeometries 与 targetGeometries 中。
  - 图（2）：对称约束的对称轴 G3 存储在 symmetryAxises 中，其他两个几何元素 G1 与 G2 均存储在 targetGeometries 中。

图 12-3 对称约束的传参方式



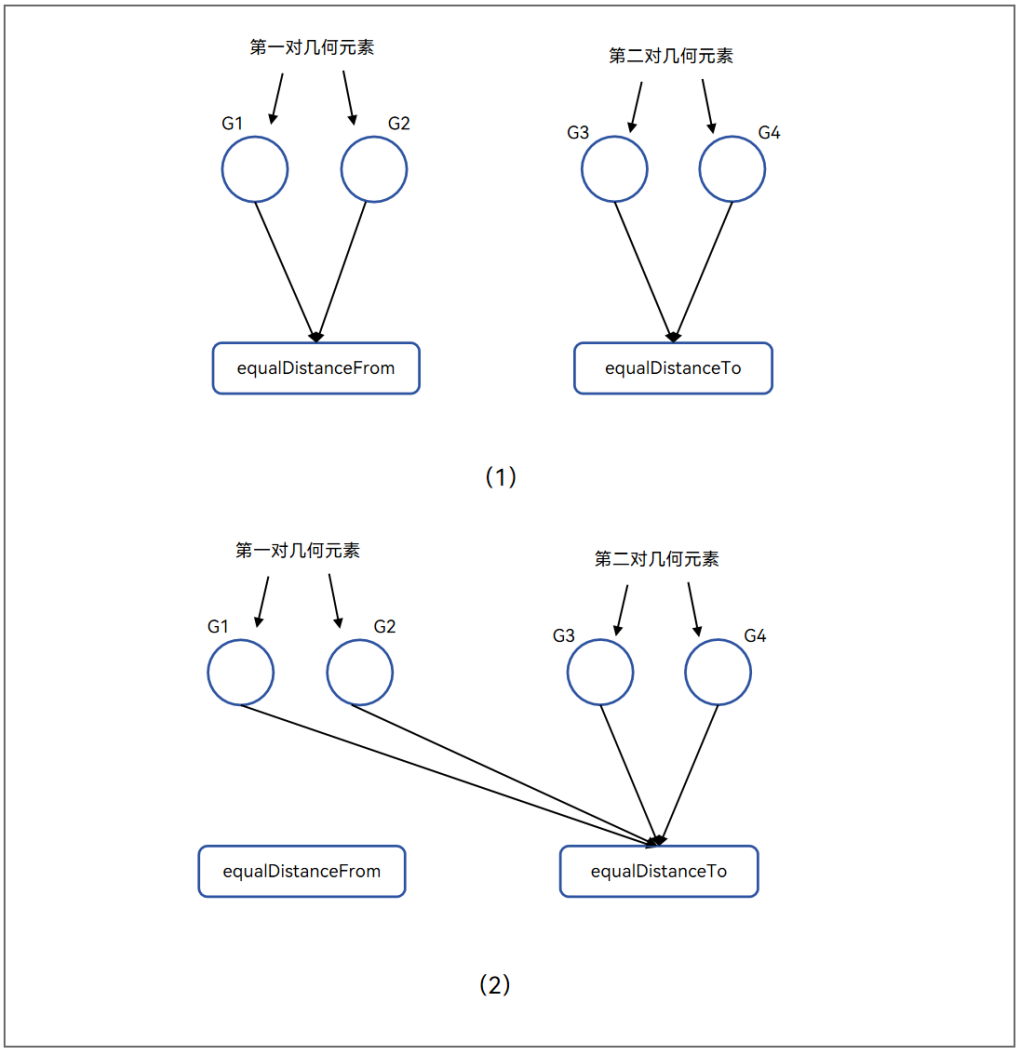
- 图（1）：中点约束的中点 G3 存储在 groundGeometries 中，其他两个几何元素 G1 与 G2 分别存储在 groundGeometries 与 targetGeometries 中。
- 图（2）：中点约束的中点 G3 存储在 targetGeometries 中，其他两个几何元素 G1 与 G2 分别存储在 groundGeometries 与 targetGeometries 中。
- 图（3）：中点约束的中点 G3 存储在 groundGeometries 中，其他两个几何元素 G1 与 G2 均存储在 targetGeometries 中。
- 图（4）：中点约束的中点 G3、以及其他两个几何元素 G1 与 G2 均存储在 targetGeometries 中。

图 12-4 中点约束的传参方式



- 四元约束：自动逻辑约束中，四元约束也就表示等距离约束，如图图 12-5 等距离约束的传参方式所示，推荐您使用图（1）中的传参方式 。
  - 图（1）：等距离约束的第一对几何元素 G1 与 G2 存储在 equalDistanceFrom 中，另外一对几何元素 G3 与 G4 存储在 equalDistanceTo 中。
  - 图（2）：等距离约束的第一对几何元素 G1 与 G2，以及另外一对几何元素 G3 与 G4 均存储在 equalDistanceTo 中。

图 12-5 等距离约束的传参方式



再次提醒，添加对称约束与中点约束类型时，均需要三个几何元素。对称约束的对称轴直接在对称轴列表进行查询；中点约束的中点会自动从 groundGeometries 或 targetGeometries 中获取。

本软件会根据容差找到符合几何元素相对位置的几何约束，然后分析这些几何约束是否与系统中原有的几何约束存在冲突或冗余关系，如果存在，则丢弃这部分几何约束，以免出现过约束的情况。

涉及的接口：**autoConstraint**。

12.1.1 自动逻辑约束接口使用

在使用自动逻辑约束接口时，应用程序需要考虑以下几点。

- 默认情况下，自动逻辑约束会为添加在椭圆与参数化曲线的几何约束提供帮助参数。

- 默认情况下，本软件设计的自动逻辑约束会检测几何元素之间支持的约束关系，并控制新增几何约束数量的最小化，尽可能使约束系统达到良好约束状态。相当于如果本软件检测到的几何约束已经与模型上存在的或隐含的几何约束相同，本软件将不会新增该几何约束，比如两条线之间的距离约束隐含了平行约束。
- 自动逻辑约束只会检测无界几何元素之间的几何约束。例如，本软件会找到圆与直线之间的相切约束的多个解决方案，但是部分应用程序可能需要在直线与有界的弧之间添加相切约束，此时应用程序可以舍去不需要的解决方案。
- 一个大型的模型中可能包含较多约束条件。本软件会对每个约束条件进行检查，并丢弃部分不必要的约束条件。
- 自动逻辑约束不会移动约束系统中的任何几何元素，因此，尽管调用 **autoConstraint** 时会根据指定的容差来检测几何约束，但实际上这些容差可能不满足本软件的容差标准。建议应用程序调用 **autoConstraint** 后使用 **evaluate** 再次求解模型。
- 等距离约束只能应用于合理的几何元素对之间。比如自动为两条线段的端点或两对偏置轮廓添加等距离约束。在本软件中，为两条线段的端点添加等距离约束相当于限制两条线段具有相同的长度。
- 直线、椭圆、样条曲线、圆锥曲线、自定义回调曲线、自定义参数曲线以及线性阵列几何元素都是具有一个方向的几何元素，本软件在自动逻辑约束期间能够推断出与这些几何元素相关的平行约束或垂直约束。

12.2 自动尺寸约束

自动尺寸约束是为欠约束模型中的几何元素添加尺寸约束的过程。自动尺寸约束与自动逻辑约束一样，向约束系统中添加几何约束的同时保证约束系统不会过约束。

建议您在添加尺寸约束之前优先添加逻辑约束。

通常，在一个具有 N 个欠约束的几何元素的模型中，需要 N 个尺寸约束使模型达到良好约束状态，并且有 N^2 种方法将它们添加到约束系统中。

自动尺寸约束支持的约束类型：

- 一元约束：半径约束。
- 二元约束：角度约束、距离约束、方向距离约束。
- 三元约束：平行距离约束、垂直距离约束。

应用程序要求本软件进行自动尺寸约束之前，需要先向本软件提供线性容差，角度容差、需要进行判断的几何约束类型以及几何元素的列表等，以下进行简要说明。

- 自动尺寸约束的线性容差（linearTolerance）：如果应用程序没有提供自动尺寸约束的线性容差，本软件将使用 2D 求解器的线性容差。
- 自动尺寸约束的角度容差（angleTolerance）：如果应用程序没有提供自动尺寸约束的角度容差，本软件将使用 2D 求解器的角度容差。
- 尺寸约束关联的几何元素列表（groundGeometries）：存储二元约束与三元约束中除参考线、参考方向以外的目标几何元素。
  - 与 targetGeometries 中的几何元素组合生成二元尺寸约束。
  - 与 targetGeometries 和 referenceLines 中的几何元素组合生成参考线距离约束。

— 与 targetGeometries 中的几何元素和 referenceDirections 中的参考方向组合生成方向距离约束。

具体组合方式，请见下方图示。

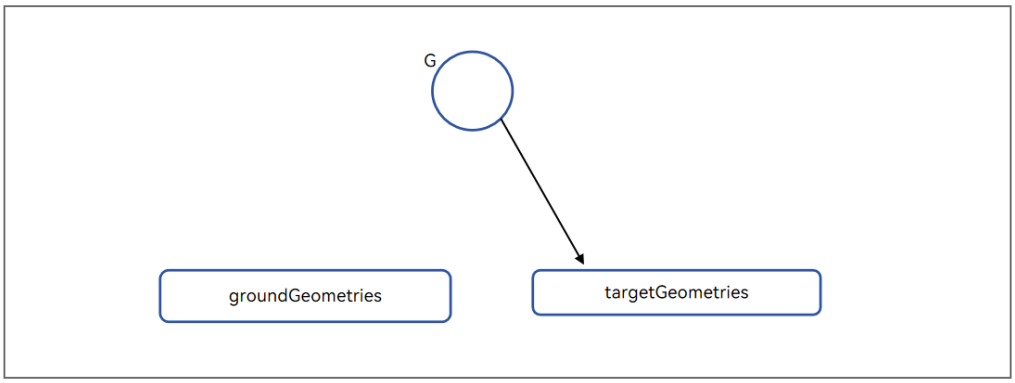
- 尺寸约束关联的几何元素列表（targetGeometries）：存储一元约束、二元约束与三元约束中除参考线、参考方向以外的目标几何元素。
  - 数组内部的几何元素自动生成一元约束或组合生成二元尺寸约束。
  - 与 referenceLines 中的几何元素组合生成参考线距离约束。
  - 与 referenceDirections 中的参考方向组合生成方向距离约束。
  - 与 groundGeometries 中的几何元素组合生成二元尺寸约束。
  - 与 groundGeometries 和 referenceLines 中的几何元素组合生成参考线距离约束。
  - 与 groundGeometries 中的几何元素和 referenceDirections 中的参考方向组合生成方向距离约束。

  具体组合方式，请见下方图示。
- 自动尺寸约束的约束类型（constraintTypes）：本软件将根据列表中的几何约束类型来识别模型。
- 参考线列表（referenceLines）：如果要自动添加平行距离约束与垂直距离约束，referenceLines 用于存储参考线。当添加除平行距离约束与垂直距离约束以外的其他尺寸约束时，该列表无意义。
- 参考方向列表（referenceDirections）：如果要添加方向距离约束，referenceDirections 用于存储方向。当添加除方向距离约束以外的其他尺寸约束时，该列表无意义。

关于 groundGeometries、targetGeometries、referenceLines 与 referenceDirections 的具体存储方式，有以下几种组合：

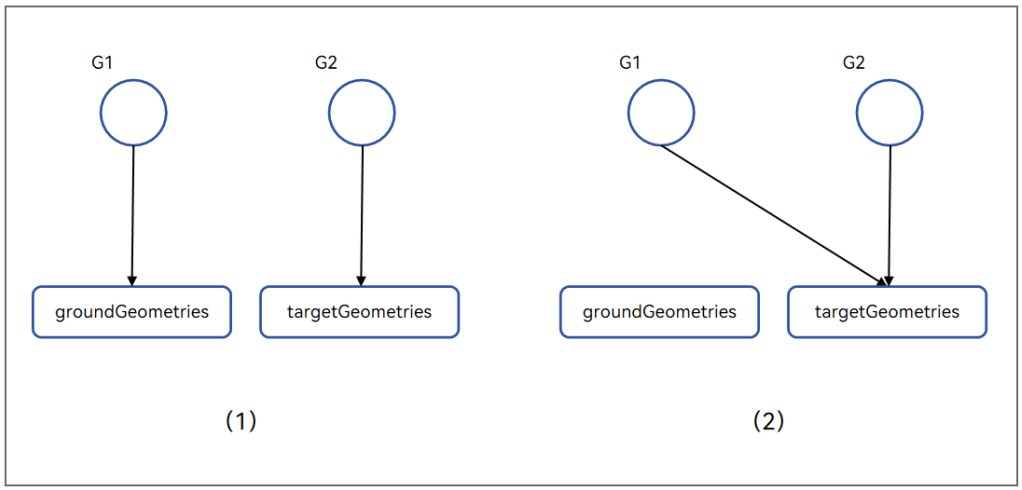
- 一元约束：几何元素 G 存储在 targetGeometries 中，如图 12-6 一元约束的传参方式所示。

图 12-6 一元约束的传参方式



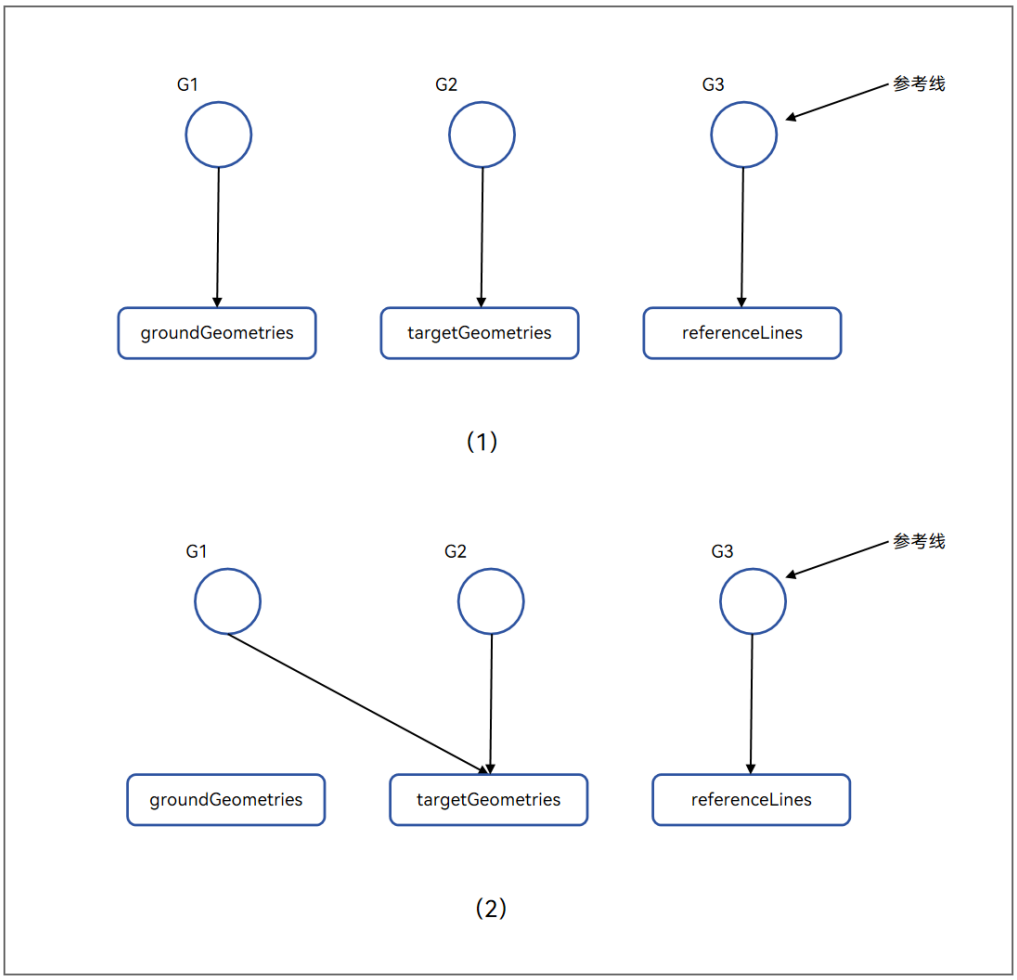
- 二元约束：如图图 12-7 二元约束的传参方式所示。注意，方向距离约束的方向存储在 referenceDirections 中，推荐您使用图（1）中的传参方式。
  - 图（1）：两个几何元素 G1 与 G2 分别存储在 groundGeometries 与 targetGeometries 中。
  - 图（2）：两个几何元素 G1 与 G2 均存储在 targetGeometries 中。

图 12-7 二元约束的传参方式



- 三元约束：自动逻辑约束中，三元约束包括平行距离约束与垂直距离约束，统称为参考线距离约束，参考线距离约束关联的几何元素的传入规则如图 12-8 参考线距离约束的传参方式所示，推荐您使用图（1）中的传参方式。
  - 图（1）：参考线距离约束的参考线 G3 存储在 referenceLines 中，其他两个几何元素 G1 与 G2 分别存储在 groundGeometries 与 targetGeometries 中。
  - 图（2）：参考线距离约束的参考线 G3 存储在 referenceLines 中，其他两个几何元素 G1 与 G2 均存储在 targetGeometries 中。

图 12-8 参考线距离约束的传参方式



再次强调，平行距离约束与垂直距离约束的参考线在参考线列表中查询；距离约束的参考方向在参考方向列表中查询。

涉及的接口：**autoDimension**。

13 约束系统状态信息

本软件除了计算几何元素的相对位置以外，还会计算模型中每个实体的状态。本章详细描述了每个几何元素、每个几何约束以及求解结果的状态信息。

调用以下接口获取状态码：：

- **getGeometryStatus**：获取几何元素状态。
- **getGeometryChangeStatus**：获取几何元素变化状态。
- **getConstraintStatus**：获取几何约束状态。
- **getConstraintSatisfiedStatus**：获取几何约束满足状态。

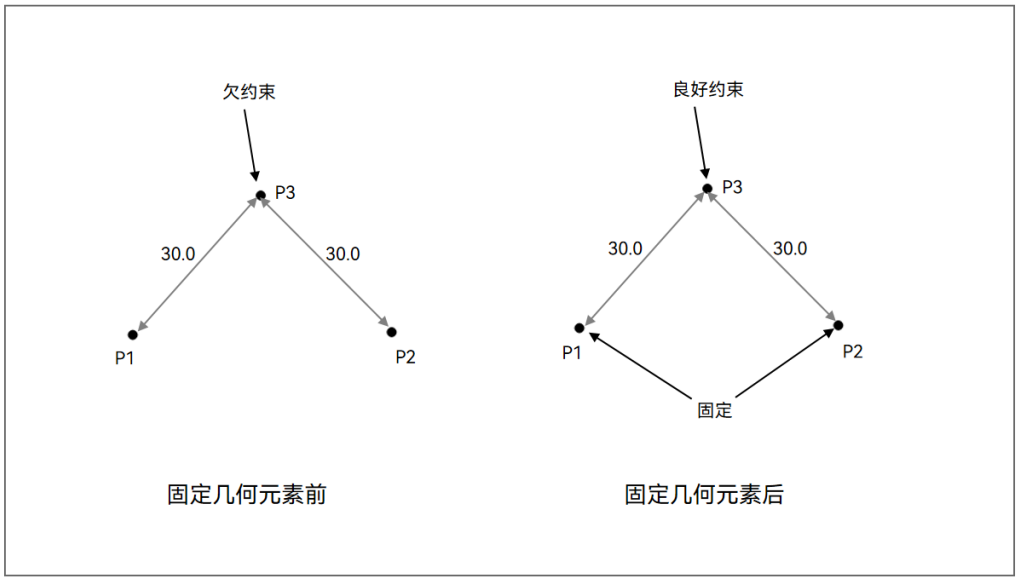
求解状态会在 **measureDimension**、**evaluate**、**reEvaluate**、**dragging**、**moveAndEvaluate**、**overConstraintAnalysis** 及 **evaluateSolution** 的接口调用中返回。

13.1 几何元素状态

几何元素状态有以下四种类型：

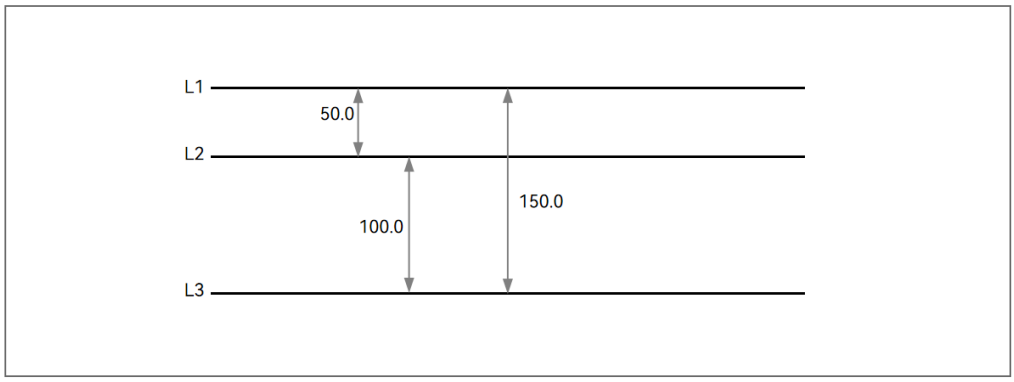
- UNKNOWN（未知状态）：一个几何元素被首次添加到 2D 求解器中，此时尚未求解，其几何元素状态为 UNKNOWN。
- WELL_DEFINED（良好约束状态）：求解后，约束系统中的所有几何约束都能够被满足，几何元素的所有自由度都受到了约束。例如，一个具有固定约束的点为 WELL_DEFINED 状态。
- UNDER_DEFINED（欠约束状态）：求解后，约束系统中的所有几何约束都能够被满足，但是几何元素的自由度尚未完全受到约束。例如，在两条直线之间仅添加了平行约束，此时两条直线之间的距离没有受到约束，因此两条直线都是 UNDER_DEFINED 状态。
  如果模型中不包含任何固定的几何元素，那么模型不能达到良好的约束状态。图 13-1 过约束模型示例给出了包含三个点与两个距离约束的模型，在没有固定任意点之前，求解后 P1、P2、P3 均处于欠约束状态；固定 P1 与 P2，求解后除了 P1 与 P2 以外，P3 也处于良好约束状态。

图 13-1 过约束模型示例



- OVER_DEFINED（过约束状态）：表示几何元素的部分或全部自由度被重复约束。例如，为一条直线添加“水平”和“垂直”约束，则该直线为 OVER_DEFINED 状态，不能被求解。
  当几何元素的自由度被几何约束重复约束时，本软件会使用 OVER_DEFINED 来告知应用程序哪些几何元素是过约束状态。当使用变量和方程时，如果方程太多或变量不足，也会出现类似的过约束情况。若要解决过约束的问题，应用程序需要删除对应的几何约束、方程、刚性集合，或删除几何元素的固定约束。
  当求解包含一个或多个尺寸约束的模型时，本软件会找到支持每个尺寸约束的值能够独立变化的解决方案。如果解决方案只对特定的值组合有效，那么本软件会视该模型为过约束状态。例如，图 13-2 过约束模型示例中有三条直线 L1、L2 与 L3，三条直线当中，两两关联有距离约束，此时任一距离约束都不能独立变化。它们只有如图的特定位置能够满足几何约束，本软件将该模型视为过约束状态。

图 13-2 过约束模型示例



添加过多几何约束可能会导致刚性集合过约束。但是只要这些几何约束是逻辑约束和刚性的距离约束，并且几何约束是一致性的，本软件就能够成功求解模型。例如，在一个集合中的矩形轮廓与另一个集合中相同大小的矩形轮廓之间添加重合约束。

本软件提供了 check 函数来检查某几何约束会不会导致模型变成过约束状态。对求解模型来说，在应用几何约束之前先使用 check 函数进行检查，运行速度会比直接应用几何约束的速度更快。

只有当一个几何元素的状态码为 WELL_DEFINED 或 UNDER_DEFINED 时，才能求解成功。

13.2 几何元素变化状态

几何元素变化状态有以下四种类型：

- UNKNOWN（未知状态）：一个几何元素被首次添加到 2D 求解器中，此时尚未求解，其几何元素变化状态为 UNKNOWN。
- FIXED（固定状态）：几何元素被固定。如果应用程序调用接口为几何元素添加了固定约束，此时几何元素的变化状态为 FIXED。2D 求解器不能移动固定的几何元素。求解时，本软件会忽略掉固定几何元素之间的所有几何约束。

  说明
  本软件将找到一个假设固定几何元素始终保持在同一位置的解决方案。2D 求解器成功求解模型以后，如果应用程序此时移动了固定几何元素，那么该模型可能无法再次被成功求解。

- CHANGED（变化状态）：在上一次求解过程中，几何元素的位置与形状被更改，此时几何元素的变化状态为 CHANGED。
- NOT_CHANGED:（未变化状态）：如果本软件发现模型的某些部分是非代数的、过度约束的或约束不一致，那么这部分几何元素将不会被求解，几何元素的位置与形状没有被更改，此时几何元素的变化状态为 NOT_CHANGED。除此之外，每一个几何元素的固定约束被删除后，本软件会将几何元素的变化状态设置为 NOT_CHANGED。

13.3 几何约束状态

几何约束状态有以下五种类型：

- UNKNOWN（未知状态）：一个几何约束被首次添加到 2D 求解器中，此时尚未求解，其几何约束状态为 UNKNOWN。
- OVERDEFINED（过约束状态）：一个几何约束与其他一个或多个几何约束发生冲突，导致无法求解，其几何约束状态为 OVERDEFINED。
- BETWEEN_FIXED（固定元素间的几何约束状态）：当一个几何约束关联的所有几何元素的变化状态均为 FIXED 时，该几何约束的状态为 BETWEEN_FIXED。例如在两个点之间添加一个距离约束，只有当两个点都被固定住以后，该距离约束的状态才能被设置为 BETWEEN_FIXED。除了二元或多元几何约束外，该状态码还能作用于固定的单个几何元素上的几何约束，如固定整个圆，为该圆添加一个半径约束，此时半径约束的状态为 BETWEEN_FIXED。
- BETWEEN_SET_MEMBERS：在同一集合中的几何元素之间的几何约束状态，例如集合中的半径约束、长轴半径约束等。

- REGULAR（正常状态）：能够求解出满足几何约束的解决方案。

只有当一个几何约束的状态码为 REGULAR 时，才能成功求解。几何约束状态为 BETWEEN_FIXED 或 BETWEEN_SET_MEMBERS 之间的几何约束通常会求解失败。

13.4 几何约束满足状态

几何约束满足状态有以下三种类型：

- UNKNOWN（未知）：一个几何约束被首次添加到 2D 求解器中，此时尚未求解，其几何约束满足状态为 UNKNOWN。
- SATISFIED（满足）：向约束系统中添加一个几何约束，如果其关联的几何元素具有可供其约束的自由度，那么该几何约束的满足状态为 SATISFIED。
- NOT_SATISFIED（不满足）：向约束系统中添加一个几何约束，如果其关联的几何元素没有可供其约束的自由度，那么该几何约束的满足状态为 NOT_SATISFIED。

13.5 求解状态

求解状态有以下十种类型：

- UNKNOWN：未初始化。
- SUCCESS：求解成功。
- VALIDATION_FAILED：结果验证失败。求解成功，但结果不能满足容差要求。
- SOLUTION_NOT_FOUND：求解失败。
- INTERNAL_FAILURE：发生内部错误。
- TIMEOUT：求解超时，默认单次求解时间为 60s。
- ABORTED：求解中断。
- PROCESSING：求解进行中，求解器没有足够的资源来实现当前任务。
- SAVE_REPLAY_FAILURE：保存 replay 文件失败。
- EVALUATE_MULTIPLE_SOLUTION_FAILURE：多解方案求解失败。

14 集成和使用 Geoshape

Geoshape 约束求解引擎软件是一个软件模块。本章将为您介绍如何在应用程序中使用本软件提供的接口，并列出了在设计应用程序时一些重要的注意事项。

ISV 工具集成并使用了本软件的主要功能，将与本软件一同交付给您，为您提供基于本软件开发应用程序的示例。

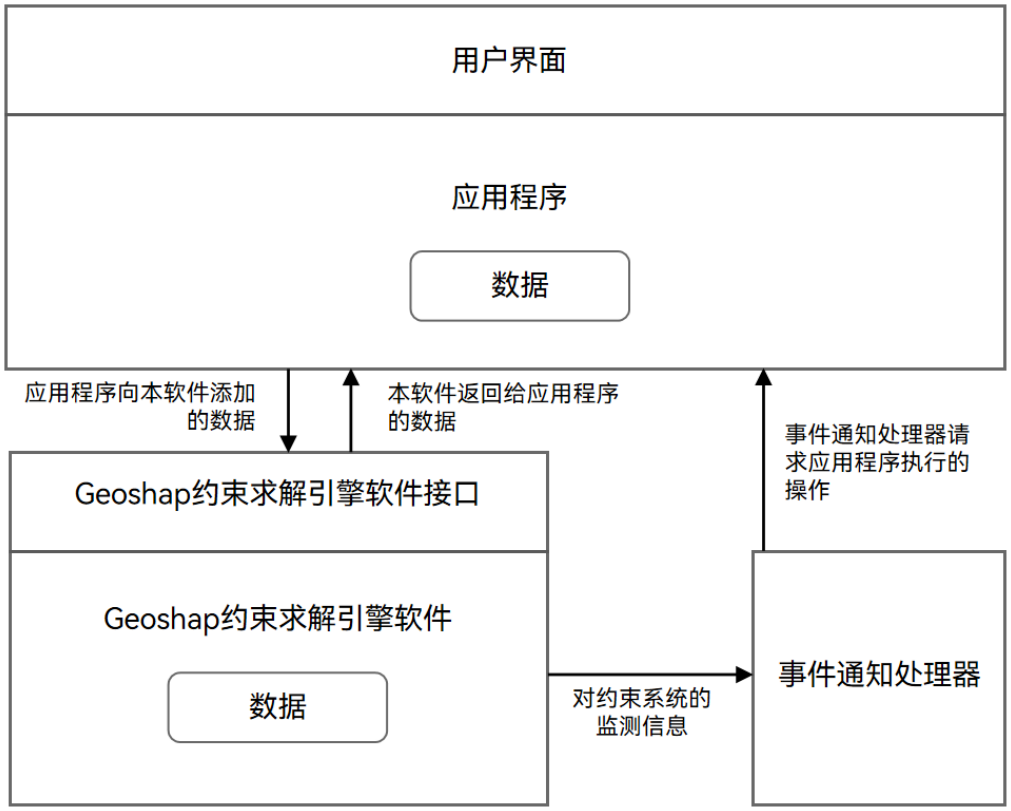
您可以在本软件提供的接口上作一层封装来满足用户的需求，这也是当前市场上，每个应用程序有所不同的原因。下面的章节中也会告诉您如何高效利用本软件的接口来做开发。

14.1 软件结构

本软件的设计意图是作为其他应用程序的软件组件来实现对模型的求解。本软件采用C++高级程序设计语言进行编码，您可以使用 C++、C 或任何支持 ANSIC 的高级语言来调用本软件的接口。

整体软件结构如下图所示：

图 14-1 软件结构



您可以通过 Geoshape 约束求解引擎软件接口来访问本软件主要功能。如果应用程序在 2D 求解器中注册了事件通知处理器，那么当约束系统中出现除应用程序主动调用接口外产生的变更时，事件通知处理器能够自动监测到这些变更部分并通知应用程序，监测到变更后的具体行为由应用程序定义。例如，如果您想在求解结束时自动执行某些操作，您需要重写事件通知处理器中的 onSolvingEnd 函数，在其中定义您想要执行的操作，并将事件通知处理器注册到您想要实现自定义操作的 2D 求解器中。如果您不需要本软件自动传递监测信息，您可以直接使用事件通知处理器中的虚函数，或不在 2D 求解器中进行注册。

本软件和应用程序之间通过标识 ID 来交换数据。本软件具有自身的存储空间，与应用程序的数据相互独立。

14.2 集成约束求解引擎

本节介绍将约束求解引擎软件集成到应用程序的操作步骤。本节假设您对 C++编程有基础了解。

解压缩约束求解引擎软件包后的目录结构如下。

```
─install
   ├─api          //API 接口头文件
   ├─bin          //DLL 及相关文件
```

```
├─example    //示例代码
└─lib        //lib 文件
```

在 bin\conf 文件夹下有“log.config.json”文件。此文件可以设置约束求解引擎的日志相关信息。

- MAX_FILE_SIZE
  ：单个日志文件大小的最大值。
- MAX_BACKUP_INDEX
  ：日志文件保存的最大数量。
- LOG_GRADE
  ：设置是否打开日志及日志级别。
  - 1：打开日志并设置日志级别为 Error。
  - 2：打开日志并设置日志级别为 Warning。
  - 3：打开日志并设置日志级别为 Info。
  - 4：打开日志并设置日志级别为 Fine。

在 bin\conf 文件夹下有“pscs.sdk.ini”文件。此文件可以设置约束求解引擎使用网络浮动 License 时网络许可服务器的信息。内核会默认校验本地 License 文件，如果没有本地 License 文件，会根据此配置进行网络 License 校验。

- FLOAT_LICENSE_SERVER_IP：网络许可证服务器的 IP 地址。
- FLOAT_LICENSE_SERVER_PORT：网络许可证服务的端口。

前提条件

- 已经获取到约束求解引擎软件包并解压缩。
- 已经创建应用程序工作目录。
- 已经获取到 License 相关信息。

操作步骤

1. 分别将“api”、“bin”、“lib”文件夹下的文件复制到引用程序工作目录下对应的文件夹。
2. 设置日志相关信息。
3. 将单机版 License 存放于内核 DLL 文件同级目录。
4. 在需要调用约束求解引擎 API 接口的文件引入头文件。举例如下。

```
#include <PSCSApiSolver2D.h>
#include <PSCSApiSolver2DEngine.h>
#include <PSCSApiType2D.h>
```

后续操作

在 example 文件夹下，提供了一个用 C++编写的示例应用程序，请阅读并编译使用该实例程序。

14.3 数据处理

本软件具有自身的存储空间，与应用程序的数据相互独立。应用程序可以自由地定义数据结构，但是必须包含本软件接口所需的信息。

14.3.1 存储几何元素与几何约束的数据

本节主要介绍本软件对应用程序中数据结构的设计要求。

14.3.1.1 直线上的位置

本节主要介绍直线上的位置的相关特性。

本软件中，直线是通过两个向量来定义的，分别是直线上某具体位置的向量与直线的方向向量。这种表示方法能够让本软件明确直线的走向及其在空间中的具体位置。大多数情况下，即使应用程序改变指定的位置或方向向量的长度，也不会对 2D 求解器的行为造成影响，因此应用程序具有很大的操作空间。例如，应用程序可以自由地选择直线上的任意位置作为参数，不必担心位置会影响 2D 求解器对直线的处理。

当几何元素处于欠约束状态时，直线上的位置对求解结果可能会产生重要影响。以旋转直线为典型示例，例如，在两条相交直线之间添加一个角度约束，交点位于尺寸框内，本软件将会围绕交点旋转其中一条直线来寻找解决方案。除了与上述情况类似的特殊模型外，本软件无法检测到类似两条直线的交点这样的位置，就会使用应用程序指定的位置来旋转直线。

14.3.1.2 直线的方向

本节主要介绍直线方向的相关特性。

本软件中，直线的方向向量是定义直线的基本信息之一。对直线来说，其方向能够确定角度约束的求解方式，以及几何约束的约束对齐方式与约束方位。因此，应用程序有必要保持直线的方向，并且只有在本软件有需求时才进行更改。

应用程序可以使用两个点、一条直线以及两点与该直线之间的两个重合约束来构造线段。当线段长度缩小为 0 时，起始点与结束点重合，继续移动两点的位置会交换它们的顺序。基于这种构造方式，应用程序不能通过线段的起始点与结束点来定义线段的方向，线段的方向应该由直线的方向决定。因此，应用程序有必要保持直线的方向。

14.3.1.3 删除数据

本软件删除一个节点时，可能会删除掉应用程序添加的其他节点。例如当本软件删除一个几何元素时，同时会自动删除与其关联的几何约束。为了帮助应用程序接收数据被删除的信息，应用程序可以将事件通知处理器注册到 2D 求解器中，2D 求解器会自动调用相关接口来通知应用程序。

14.3.1.4 帮助点和帮助参数

应用程序需要存储几何约束的帮助点和帮助参数。

如果应用程序中不设置帮助点或帮助参数，本软件也能进行求解，但是存在一定的局限性。例如，如果没有帮助点，直线到圆的距离约束只能是最小距离，而不能使用方向距离约束。所以建议您为符合条件的几何约束加上帮助点或帮助参数。有关最小距离的详细信息，请参见 5.1.2.1 默认的距离约束行为。

14.3.1.5 有界的几何元素

本软件主要支持无界的直线、圆和椭圆，“无界”意味着这些几何形状没有明确的起始点和结束点，它们可以无限地延伸下去。但是在实际应用中，应用程序通常需要表示有界的几何元素，即有明确的起始点和结束点的形状，例如，一条线段、一段圆弧或者一个封闭的图形等。

为了表示有界几何元素，应用程序可以在创建无界几何元素的同时，定义其起始点和结束点，并确保这两个点与无界几何元素重合。这样，虽然本软件支持的是无界几何元素，但是在两点（起始点和结束点）与无界几何元素之间添加重合约束后，能够得到的一个有界几何元素。

应用程序需要维护起始点、结束点与无界几何元素之间的关系记录。当修改或查询有界几何元素的数据时，应用程序才能够正确地处理这些关系，以确保数据的准确性和一致性。

除了简单的有界几何元素外，您还可以利用这种方法构建更复杂的数据结构，如闭合的环。这些环可以由多条线段、圆弧或其他有界曲线组成，形成一个完整的封闭图形。

14.3.2 独立于本软件更改数据

除了调用本软件的接口来更改模型的数据以外，应用程序还可以独立于本软件更改几何元素、几何约束、方程式中的常数和系数等数据。但是应用程序不能更改实体的类型，比如将直线“弯曲”成弧。

14.3.3 在模型上绘制轮廓

通常，应用程序允许用户创建模型后，再使用本软件在模型的某个面上绘制轮廓。此时模型在该面的轮廓可以被固定，在求解时，本软件不能移动该轮廓，从而保持模型的稳定性和一致性。在某些应用程序中，用户可能希望固定某些关键的几何特征，如基准面或基准轴，以确保这些特征在后续的编辑或操作中不会被意外移动或改变。

14.3.4 错误检查

为了避免本软件与应用程序进行大量重复检查，本软件仅为应用程序调用单个接口可能产生的错误进行了检查。

调用 **getLastErrorCode** 能够获取最后一次调用的接口产生的错误码；调用 **getLastErrorCode** 能够获取错误码的详细信息，错误码信息结构体如下：

- Error Message：错误信息
- errorCode：错误码
- errorIndex：错误索引

如果您认为应用程序的实现有误，您可以使用日志查看接口调用的详细信息。

14.3.4.1 异常处理

本软件不会直接抛出具体的内部异常，如果在求解、拖拽与过约束分析等接口调用时发生内部错误，将返回 InternalFailure 给应用程序。

14.3.4.2 中止求解任务

如果一个接口被中断，则其数据结构可能会损坏。如果应用程序没有释放 2D 求解器占用的资源，那么唯一安全的操作是中止求解任务。应用程序可以调用 **abortEvaluate** 接口请求本软件尽快中止求解。

14.3.5 撤销求解

由于求解的结果与实体添加顺序或时间序列无关，因此可以通过重新创建以前的实体结构和重置几何元素的位置来撤销求解。使用 **undoEvaluation** 接口，可以撤销本次求解，将模型恢复到上一次求解结束时的状态，恢复的数据包括模型中的帮助参数、几何约束的状态、几何元素的状态以及几何元素的位置。

如果应用程序连续两次调用了 **undoEvaluation** 接口，第二次接口调用会撤销第一次接口调用的结果，相当于回到第一次调用 **undoEvaluation** 之前的求解状态。

如果应用程序在约束系统第一次求解结束之前调用了 **undoEvaluation** 接口，由于在第一次求解结束之前模型没有可用的状态数据，所以模型无法恢复到上一次求解结束时的状态，此时接口调用无效。

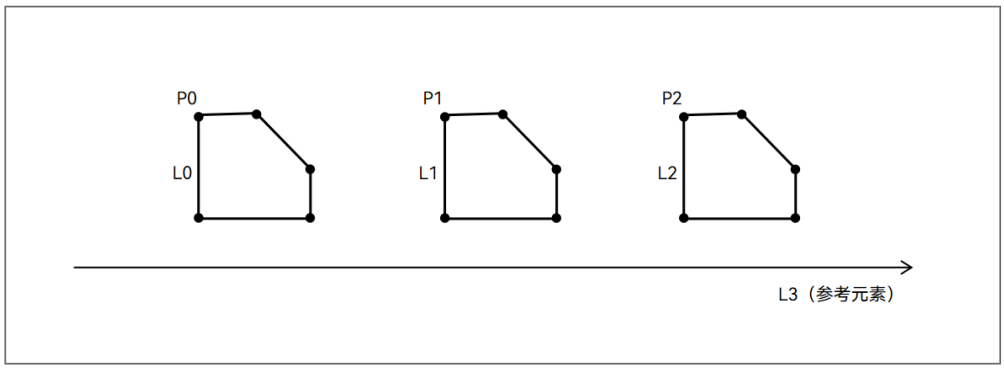
涉及的接口：**undoEvaluation**。

14.4 阵列约束

在 5.2.12 阵列约束中已经为您介绍了如何在模型上使用阵列约束来使一组几何元素以规则的方式重复出现。本节将为您介绍在一组阵列几何元素上添加几何约束的最佳方式，并且对使用阵列约束生成规则的多边形进行了说明。

图 14-2 一个参考元素的线性阵列展示了以一条直线 L3 作为参考元素的线性阵列几何元素。

图 14-2 一个参考元素的线性阵列



要想达到图中展示的效果，应用程序必须先将参考元素等几何元素添加到本软件中，再使用 **addGeometry** 接口定义阵列几何元素，明确阵列约束的参考元素与阵列值，最后使用 **addConstraint** 接口创建一个线性阵列约束。此外，应用程序还应执行以下操作：

- 复制多边形中的所有几何约束，例如 P0 与 L0 之间关联了重合约束，因此 P1 也应与 L1 重合，P2 应与 L2 重合，以此类推。这能保证每个被阵列的多边形与包含种子几何元素的多边形的几何约束保持一致。
- 在多边形中包含的几何元素之间添加阵列约束，例如 P0、P1 和 P2 之间有阵列约束，L0、L1 和 L2 也关联有阵列约束。

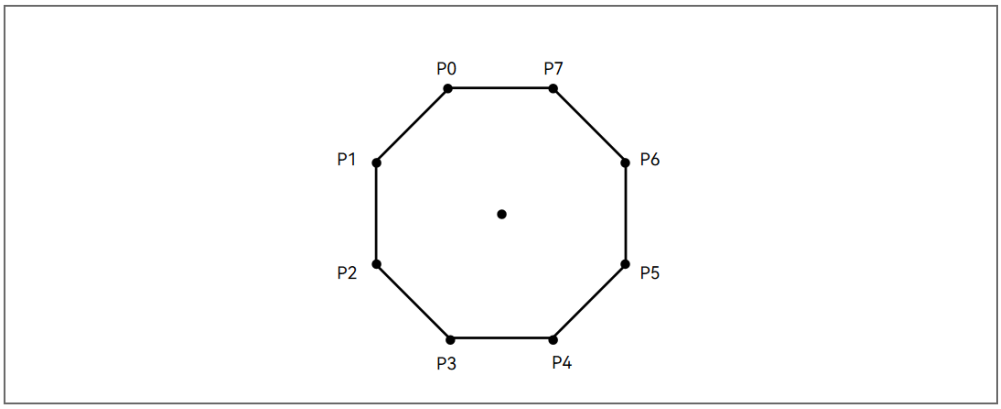
尽管这些几何约束在定义模型时不是必需的，但是添加这些几何约束后，本软件可以更准确地理解和求解阵列几何元素中各元素之间的关系，从而生成更准确、更稳定的模型，所以它们的存在有助于本软件更优地求解阵列几何元素。

14.5 正多边形

本软件推荐您使用环形阵列约束来绘制规则的多边形。

图 14-3 一个参考元素的线性阵列展示了一个通过封闭的环形阵列创建的八边形。

图 14-3 一个参考元素的线性阵列



要想达到图中展示的效果，应用程序必须先向本软件中添加构成八边形所需的直线和点，以及环形阵列约束所需的参考元素（点或圆），随后根据直线和参考元素创建一个环形阵列约束。此时八边形的每条边都是一整条开放的直线，您可以在点和直线之间添加重合约束来实现上图的效果。

在创建阵列几何元素时，有两种主要的方法：“主实例方法”和“链方法”。

主实例方法

- 概念：在主实例与其他每个实例之间分别添加一个阵列约束，在图 14-3 一个参考元素的线性阵列中，相当于 P0 和 P1 之间阵列约束的乘数为 1，P0 和 P2 之间阵列约束的乘数为 2，以此类推。在最后一个实例与主实例之间添加一个负号乘数来闭合阵列几何元素，相当于 P0 和 P7 之间有两个阵列约束，对应两个乘数，分别是 7 与-1。
- 性能与可靠性：这种方法通常能提供更好的性能和更可靠的测量。
- 闭合性：只要第一个和最后一个几何元素存在，阵列几何元素将始终保持闭合。即使中间的一条边或一个点被删除，阵列几何元素的闭合性仍保持不变。
- 删除特性：删除主实例，整个阵列几何元素中的几何约束都将被删除。
- 应用：适用于需要整体性和稳定性的应用，因为在一定程度范围内的修改不会影响阵列几何元素的整体结构。

链方法

- 概念：在每个实例之间依次添加一个阵列约束，在图 14-3 一个参考元素的线性阵列中，相当于在 P0 和 P1 之间、P1 和 P2 之间添加阵列约束，以此类推，最后在 P7 和 P0 之间添加阵列约束，所有阵列约束的乘数都为 1。
- 开放性：当链方法中的单个实例被删除时，整个阵列几何元素会变成开放的。即删除一个几何元素会破坏阵列几何元素的连续性和完整性。
- 删除特性：在链方法中，删除任一几何元素，都无法删除所有的阵列约束。每个几何元素都是维持阵列几何元素结构的成员，但是它们可能因阵列约束的具体类型和配置而不同。
- 应用：适用于需要灵活性和可调整性的应用，因为删除或添加几何元素能够更容易地改变阵列几何元素的整体形状或配置。

使用对称约束，等距离约束、角度约束等均能创建规则的多边形，但是使用阵列约束具有一定的优势：

- 使用阵列约束时，本软件可以更可靠地求解多边形中的几何关系。这是因为阵列约束提供了一种明确的、结构化的方式来定义多边形的顶点、边和它们之间的关系。相较于其他方法，阵列约束减少了歧义，从而提高了计算精度和可靠性。
- 使用阵列约束可以确保多边形的顶点按照正确的顺序和方向排列，从而避免产生镜像或反转的多边形。
- 使用阵列约束能够识别并处理过约束但一致的情况。即使添加了多余的几何约束，只要这些几何约束逻辑上是一致的，本软件也能够正确地解析和处理它们，而不会导致错误或计算失败。

总的来说，使用阵列约束来绘制规则的多边形有许多优势。选择哪种方式来构建阵列几何元素取决于具体的应用场景和需求。

14.6 约束曲线的长度

本节将为您介绍如何使用本软件来约束曲线的长度。曲线都由基础曲线（直线、圆、椭圆或参数化曲线）与两个点来定义。曲线长度可以为一个具体的数值，也可以是变量。

- **圆弧**
  应用程序可以使用弧长约束来约束圆弧的长度。弧长约束作用于三个几何元素，分别是圆和两个点，这两个点与圆重合。
  除了弧长约束以外，应用程序也可以使用非线性方程来约束弧长。步骤如下：
  1. 构造两条线，一条穿过圆心与圆弧的起始点，另一条穿过圆心与圆弧的结束点。
  2. 在两条线之间添加一个角度约束。
  3. 在圆上添加一个半径约束。
  4. 设置非线性方程，将角度约束和半径约束作为变量，它们的乘积作为弧长。圆弧的长度实际上是它对应的角度与圆的半径的乘积。请注意，角度约束必须调整参数使其位于正确的扇区。
- **椭圆、参数化曲线及它们的弧**
  应用程序可以使用曲线长度约束来约束椭圆和参数化曲线的长度。如果只需要约束曲线的部分长度，比如椭圆弧，那么应用程序需要在曲线长度约束的帮助参数与弧的端点对应的参数之间添加等参数约束。

14.7 使用非线性方程构建高级几何约束

本节描述了应用程序如何使用非线性方程中的几何元素来实现高级的几何约束，例如环的面积约束、环的质心约束等。

14.7.1 约束一个标量的值

本节将为您介绍如何使用几何元素参数变量构造一个非线性方程，以约束标量值，例如环的长度、环的面积等。

为了实现这一点，应用程序需要向本软件添加一个非线性方程，该方程用于描述标量值与几何元素参数之间的关系，以下使用 g 来表示几何元素的参数变量。

将依赖标量值的所有几何元素参数变量加到方程中。这些变量包括点的 X 坐标，点的 Y 坐标，圆的半径等等。形如：f(g1, g2, …, gn) = 0.0。

软件内部会自动计算残差，残差是方程当前值与所需值（required_value）之间的差（residual）。形如：f(g1, g2, …, gn) - required_value = residual。其中，所需值是用户指定的几何约束的值，所需值可以作为一个尺寸约束变量添加到非线性方程中。每一轮残差计算均使用上一轮计算得出的几何元素的参数值。

例如，为了约束由多条曲线组成的环的长度，应用程序需要把每条曲线及边界点的相关参数添加到非线性方程组中。如果环中包含有线段，没必要将线段的长度添加到方程中，因为线段的长度由边界点的位置参数来定义的。

14.7.2 约束一个向量的值

上一节为您介绍了如何使用几何元素参数变量构造约束标量值的非线性方程。该方法可以扩展到约束向量的值，例如环的质心。这种情况下，向量的每个分量都需要一个单独的方程。

一个应用程序可以通过以下方式约束一个点位于环的质心：

- 将组成环的所有几何元素的相关参数变量（g1, g2, …, gn），包括点的 X 坐标和 Y 坐标等，添加到本软件中。
- 向本软件中添加一个点（P）来代表环的质心。
- 向本软件中添加两个非线性方程，并将所有环的几何元素参数变量添加到这两个方程中。

软件内部会自动计算残差，每一轮残差计算均使用上一轮计算得出的几何元素的参数值。两个方程之间的区别在于，一个方程仅使用点 P 的 X 分量，而另一个方程使用点 P 的 Y 分量。

通过这种方式，应用程序可以确保质心点 P 位于由几何元素定义的环的正确位置。这种方法为复杂的几何约束提供了强大的工具，特别是当需要确保某些特性（如质心位置）满足特定条件时。

14.8 已知限制

本节列出了即使存在一个解决方案，本软件仍然无法找到这个解决方案的情况。

- 不等式只能作用于欠约束的模型。本软件会尝试求解尚未满足不等式的模型，但是如果不等式与模型发生冲突，解决方案可能并不准确。
- 如果变量的初始值位于方程的奇异点，本软件可能无法找到有效的解决方案。奇异点通常是指方程在该点附近的行为异常或不可定义的位置，这可能导致数值方法失效或产生不准确的结果。

14.9 变换

本软件使用变换矩阵来表示几何元素的平移和旋转。

本软件的变换包含两个部分：

- 一个平移向量，包括 x 方向上的微分与 y 方向上的微分：
  [1.0, 0.0, dx, 0.0, 1.0, dy]
- 角度为 θ 的旋转矩阵：
  [cosθ, -sinθ, 0.0, sin(θ), cos(θ), 0.0]

这些矩阵被组合在一个单一的变换矩阵中，本软件中表示为：

[cosθ, -sinθ, dx, sin(θ), cos(θ), dy]

变换矩阵能够对位置向量进行线性变换。这种变换通常是通过前乘的方式来实现的，即把变换矩阵放在位置向量的左侧进行矩阵乘法运算。例如，现在约束系统有一点 P 与一个变换矩阵 T，点的位置向量为(x, y, 1)，在齐次坐标中，通常使用三维向量来表示 2D 约束系统中的点，其中最后一个分量是 1。应用变换后的新位置向量 P'可以通过以下公式计算：

TP = P'

$$\begin{bmatrix} cos\theta & sin\theta & dx \\ -sin\theta & cos\theta & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} position\ x \\ position\ y \\ 1 \end{bmatrix} = \begin{bmatrix} new\ position\ x \\ new\ position\ y \\ 1 \end{bmatrix}$$

如果现在约束系统中有一个方向向量和一个旋转矩阵，要想旋转这个方向向量，应该将旋转矩阵前乘到这个方向向量上，如下：

$$\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} direction\ x \\ direction\ y \end{bmatrix} = \begin{bmatrix} new\ direction\ x \\ new\ direction\ y \end{bmatrix}$$

旋转矩阵是定义在原点（0,0）的旋转，且顺时针方向为旋转的正方向，这里不需要知道具体的旋转角度值 θ。旋转的方向与本软件中角度约束的方向是不同的，逆时针方向表示角度约束的正方向。

在 C 或 C++语言当中，矩阵的定义如下：

```
double data[6];
data[0] = cos(θ);
data[1] = -sinθ,;
data[2] = dx;
data[3] = sin(θ);
data[4] = cos(θ);
data[5] = dy;
```

本软件中，dragging 接口与 moveAndEvaluate 接口的调用都需要一个变换矩阵的数组。下面的代码给出了一种在用 C 或 C++编写的应用程序中创建此数据结构的方法。

```
solver->beginDrag(dragObjs);

// Set transformations
PSCSApiTransform2D inputTransform1;
PSCSApiTransformMatrix2D inputTransformMatrix1;
inputTransformMatrix1.geometryID = geometryIds[2];
inputTransformMatrix1.transformMatrix = PSCSApiMatrix2D{ -1, -2 };
inputTransform1.matrices.append(inputTransformMatrix1);

// Solve
solver->dragging(inputTransform1);
```

14.10 操作日志、Replay 文件与系统日志

操作日志中包含 2D 求解器中除回调函数以外的接口调用信息，Replay 文件中包含从本软件求解类接口调用中捕获到的信息，系统日志中包含与运行、接口、统计相关的错误、警告、跟踪信息等。如果您在使用本软件的过程中发生了故障，您可以查看操作日志与 Replay 文件进行错误排查，使用系统日志来定位错误。您也可以将这些文件以及相关数据提供给泊松工程师，泊松工程师会根据这些信息重现并评估问题，并会在未来的版本处理中修复这些问题。

如果您需要进一步提高 2D 求解器的性能，您可以选择关闭对操作日志与 Replay 文件的记录。如果您还需要了解更多影响性能的操作，请参见 14.11 避免性能问题。

14.10.1 操作日志

操作日志中包含的数据

1. 2D 求解器的版本。
2. 开启操作日志功能之前，2D 求解器中已有的数据。
3. 开启操作日志功能之后调用的接口名称及对应参数。

如何生成操作日志

应用程序可以调用 startJournaling 接口来开启日志，例如：

```
solver->startJournaling("/path/to/journal");
```

solver 表示应用程序已成功创建的 2D 求解器，startJournaling 为本次开启操作日志功能的接口，"/path/to/journal"表示操作日志的存储路径，如果以相对路径来存储文件，将在主目录下生成对应目录与文件。

本软件中，操作日志用于记录调用了 startJournaling 接口的单个 2D 求解器的接口调用信息，而不涉及其他 2D 求解器的接口调用，同时与 2D 求解器引擎无关。如果想要进一步提高求解性能，应用程序可以使用 stopJournaling 接口关闭操作日志功能。

14.10.2 Replay 文件

Replay 文件中包含的数据

1. 本次求解的类型，Solve 或 Drag。
   - 当调用 evaluate、reEvaluate 与 evaluateSolution 时，本次求解类型为 Solve。
   - 当调用 dragging 与 moveAndEvaluate，本次求解类型为 Drag。
2. 求解前约束系统包含的基本信息。
   - 所有几何元素在求解前的基本信息，如几何元素 ID、位置、类型、状态及变化状态。
   - 所有变量在求解前的基本信息，如变量 ID，几何元素状态、变化状态及变量值（变量是一种特殊类型的几何元素）。

- 所有几何约束在求解前的基本信息，如几何约束 ID、类型、状态、满足状态及其关联的几何元素 ID。
- 所有方程和不等式在求解前的基本信息，如方程或不等式 ID、表达式、几何约束状态、满足状态等（方程是一种特殊类型的几何约束）。
- 线性容差与角度容差。

3. 求解后约束系统包含的基本信息。
   - 所有几何元素在求解后的基本信息，如几何元素 ID、位置、类型、状态及变化状态。
   - 所有变量在求解后的基本信息，如变量 ID，几何元素状态、变化状态及变量值。
   - 所有几何约束在求解后的基本信息，如几何约束 ID、类型、状态、满足状态及其关联的几何元素 ID。
   - 所有方程和不等式在求解后的基本信息，如方程或不等式 ID、表达式、几何约束状态、满足状态等。
   - 线性容差与角度容差。
4. 本次求解操作的求解状态。
   - UNKNOWN：未初始化。
   - SUCCESS：求解成功。
   - VALIDATION_FAILED：求解成功，但结果不能满足容差要求，验证失败。
   - SOLUTION_NOT_FOUND：系统无法求解请求的方程组，未找到解决方案。
   - INTERNAL_FAILURE：发生内部错误。
   - TIMEOUT：求解超时，默认单次求解时长为 60s。
   - ABORTED：求解中止。
   - PROCESSING：求解进行中，求解器没有足够的资源来执行当前任务。
   - SAVE_REPLAY_FAILURE：保存 replay 文件失败。
   - EVALUATE_MULTIPLE_SOLUTION_FAILURE：多解方案求解失败。

📖 说明
relay 文件的路径以字符串的形式传入，replay 文件只能记录单次求解的相关信息，如果您在两次求解或拖拽接口调用中使用了同一个 replay 文件，那么第二次接口调用会覆盖掉第一次接口调用的相关信息。

如何生成 Replay 文件

当应用程序调用 evaluate、reEvaluate、dragging、moveAndEvaluate 及 evaluateSolution 等求解或拖拽类接口时，需要将 replay 文件的存储路径作为参数传入接口，例如：

```
solver->evaluate("/path/to/replay");
```

solver 表示应用程序已成功创建的 2D 求解器，evaluate 为本次求解调用的接口，"/path/to/replay"表示 replay 文件的存储路径，如果以该相对路径来存储文件，将在主目录下生成对应目录与文件。

14.10.3 系统日志

系统日志中包含的数据

系统日志中包含与运行、接口、统计相关的错误、警告、跟踪信息等。

如何生成系统日志

当应用程序调用 initializeEngine 接口对系统进行初始化时，需要将系统日志的存储路径作为参数传入，例如：

```
PSCSApiSolver2DEngine::instance().initializeEngine("/path/to/engine","/path/to/log");
```

PSCSApiSolver2DEngine::instance()表示应用程序已实例化的 2D 求解器引擎，initializeEngine 初始化 2D 求解器引擎，"/path/to/engine"表示工作路径，"/path/to/log"表示系统日志的存储路径。

工作路径必须是一个绝对路径，系统日志的路径可以是绝对路径也可以是相对路径，如果以相对路径来存储文件，将在主目录下生成对应目录与文件。如果没有设置系统日志的存储路径，将默认使用工作路径来存储。

14.11 避免性能问题

本节主要向您介绍对本软件性能产生影响的操作，帮助您在应用程序的开发中合理高效地使用本软件。

以下列表中提到的操作只针对本软件的性能，不会影响解决方案。

- 避免使用本软件进行一些频繁的简单操作，这些操作可以在应用程序中轻松完成。例如查找一个点是否位于一条直线上。
- 本软件操作的时间开销与模型中的几何元素数量成正比，所以应用程序在保证模型基本信息不变的情况下，应该尽可能地减少添加到本软件中的几何元素数量。例如，如果两条线段在一个端点相遇，那么在此处只需要一个点。如果使用了两个点，应用程序应该确保两点是重合的。
- 在线段上添加中点会增加本软件求解模型的时间。除非有相应的需求，否则不建议在线段上随意添加中点。
- 避免使用过多固定的几何元素。即使这些几何元素没有关联几何约束，但是仍然会影响本软件的性能。
- 在处理刚性集合时，应用程序只需要将集合中对集合外部有约束作用的自由几何元素添加到本软件中。集合中的其他自由几何元素不会影响解决方案，故也不需要添加至本软件当中。
- 不要一次性拖拽大量的几何元素。
- 尽量使用增量求解，并且在必要时计算良好约束的几何元素的状态。
- 当使用方向距离约束时，本软件会在内部创建一条额外的参考线来确保测量的方向。这些额外的计算量会增加模型的复杂性和求解时间。所以应用程序应该减少对方向距离约束的不必要的使用。例如现存在两点已被约束到一条有方向的线上，那么最好使用简单的距离约束来取代方向距离约束。如果一个大型模型完全使用方向距离约束来进行尺寸标注，那么性能会显著下降。

- 如果将尺寸约束都放置在阵列几何元素的其中一个实例上。能够提高本软件的求解效率。

14.12 许可证相关功能

virtual PSLicenseInfo queryLicenseInfo() const = 0;

查询许可证的详细信息。如果您需要调用该接口，在这之前您需要先调用 instance 接口与 initializeEngine 接口完成 2D 求解器引擎的实例初始化。此接口能够在多线程环境下安全运行。可供应用程序查询的信息包括：

- 许可证有效的剩余天数。
- 当前许可证最大的连接数。
- 许可证的类型。
- 许可证的序列号。
- 许可证的 ID。
- 客户名称，授权此客户使用本软件。
- 绑定用户计算机的 MAC 地址，如果更改为其他计算机，当前许可证将无效。
- 本软件的产品类型。
- 本软件的产品版本。
- 颁发当前许可证的时间。
- 许可证版本。

virtual bool isLicenseValid(int* errorCode, const char errorMessage) const = 0;**

检查应用程序当前的许可证是否有效。如果您需要调用该接口，在这之前您需要先调用 instance 接口与 initializeEngine 接口完成 2D 求解器引擎的实例初始化。此接口能够在多线程环境下安全运行。以下列出了所有情况的返回结果：

- 0：无误。
- 1：无法连接许可证管理器。
- 2：没有可用的许可证。
- 3：文件不存在。
- 4：许可证无法打开。
- 5：错误的 MAC ID。
- 6：产品类型与许可证不匹配。
- 7：产品版本与许可证不匹配。
- 8：无法连接管理器以保持激活状态。
- 9：错误签名。
- 10：无法从许可连接管理器获得许可证。
- 11：保存许可证失败。
- 12：非法 ID。
- 13：许可证过期。
- 14：配置文件无法修改。
- 15：许可证管理员吊销许可证。

- 16：许可证被吊销。
- 17：非法配置。
- 18：由于无法写入 verify.config，已吊销和过期的文件将被拒绝。
- 19：用户调整系统时间。
- 20：无法正确解析文件。

如果您的许可证检查的返回结果非 0，请联系 PoissonSoft 售后部门寻求帮助。

