CAD 笔记

许中兴
PLCT 实验室
智能软件研究中心
中国科学院软件研究所
xuzhongxing@iscas.ac.cn

1 EXPRESS 语言

ISO 10303-11 标准定义的是 STEP 标准的描述语言 EXPRESS。STEP 标准里定义了各种数据：schema, entity, type, rule, function 等，都是用 EXPRESS 语言定义的。

STEP 定义了`SCHEMA step_merged_ap_schema`. 这个 schema 合并了几乎所有的 STEP 数据定义。

所有的 STEP 标准定义的数据可以在`https://www.steptools.com/`这个网站查到。

在 OpenCascade 里有一整套和 STEP 标准一一对应的 C++ 类定义。

与 C++ 不同的是，EXPRESS 里的 complex entity data type 有可能由多个 entity type 一起构成。也就是说，EXPRESS 允许用多个 entity 来描述一个对象的不同方面，这些 entity 不会被最终的一个 entity 继承，而是在类型树里形成一颗可能有多个根和多个叶子的子树。这些 entity 的 instance 合起来描述一个对象。在 C++ 中，描述对象的只能是最终的一个类。

了解了这一点，就很容易理解下面要说的 external mapping 了。我一开始很难理解 external mapping 的意思。

2 STEP 文件

ISO 10303-21 标准定义的是 STEP 文件的格式。STEP 文件使用 STEP 标准里定义的各种数据类型来定义一个具体的 CAD 对象，包括它所使用的所有数据实例。

STEP 文件总体上分成二个部分：HEADER 和 DATA.

STEP 文件里的数据表示很简单，list, array, set, bag 都用小括号表示。其他都很直观，熟悉编程语言的话很容易辨认。

使用 STEP 标准里的 entity 实例的定义形式是:`entity_name(data or #id)`。entity 名字后面的括号里是对应于属性的参数，可以是具体的数据，也可以是对其他实例的引用。参数列表里的 * 表示略过的参数。

对复合 entity 实例的定义有二种形式。一种称为 internal mapping, entity 名写在外面，后面的括号里按照顺序把所有的属性数据或实例引用都列出来，包括它自己定义的属性和所有父类型的属性。例如:

```
#186 = DEFINITIONAL_REPRESENTATION('',(#187),#191);
#181 = PLANE('',#182);
#180 = PCURVE('',#181,#186);
```

`#180`定义了一个 pcurve, 包括 3 个属性，其中后 2 个属性都是引用其他的数据定义实例，限于篇幅不再展开。

entity 实例的参数可以引用 entity 类型定义里指定的类型的子类型实例。比如，pcurve 的第二个属性的类型是 surface，但在实例定义时可以引用一个 plane 实例，因为 plane 是 surface 的子类型。这一点在 10303-11 标准的 12.11 Type compatibility 里进行了说明。

另一种称为 external mapping, 用一个列表来定义一个实例。在一个列表中按照字母升顺列出 entity 定义子树上所有的 entity 节点。每个 entity 节点的实例定义只包括自己定义的属性，不列出它的父类型包含的属性。

这种形式的数据定义中不出现要定义的 entity 类型的名字，只出现子树中所有 entity 的名字，因为 EXPRESS 的实例化规则很复杂，允许的实例化可以不对应于唯一的一个 entity 类型，而是对应多个 entity 类型，也就是对应于整个一个子树。可以使用父类型来引用这样一个复杂的实例。

一个 external mapping 的例子。

```
#179 = ( GEOMETRIC_REPRESENTATION_CONTEXT(2)
         PARAMETRIC_REPRESENTATION_CONTEXT()
```

```
        REPRESENTATION_CONTEXT('2D SPACE','') );
#174 = DEFINITIONAL_REPRESENTATION('',(#175),#179);
```

#179实例定义了一个 external mapping 形式的复合 entity. definitional_representation的第 3 个属性的类型是 representation_context, 但representation_context没有 3 个属性, #179是子类型 path_parameter_representation_context.

3 AM 和 AIC 的区别

AM 用文字描述数据结构的语义。AIC 用 EXPRESS 语言来给出数据结构语义的规范。

一个对比的例子参见：10303-1511 和 10303-511。

4 B-Spline 基函数

熟悉 B 样条函数最好的方法就是把它们画出来看看, 获得直观感受。我们画几个二次函数。

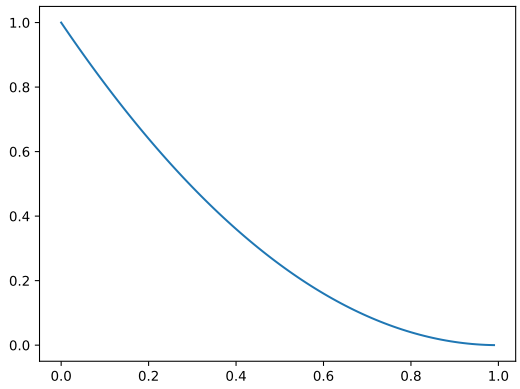

图 1: knots: [0, 0, 0, 1]

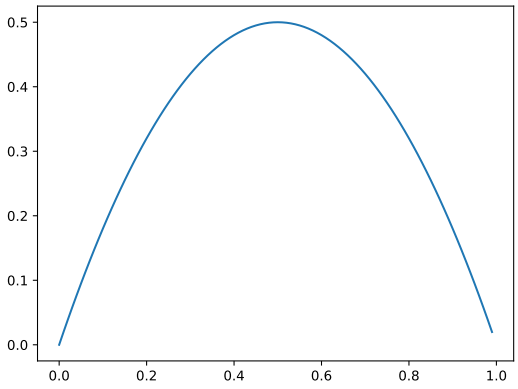

图 2: knots: [0, 0, 1, 1]

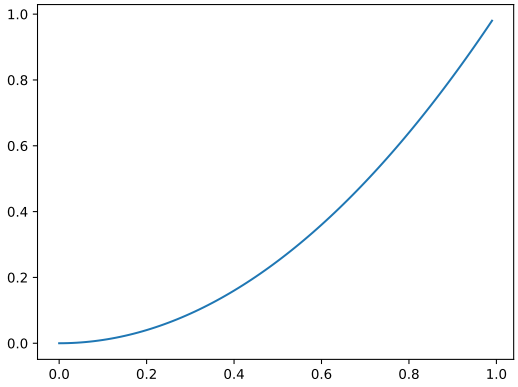

图 3: knots: [0, 1, 1, 1]

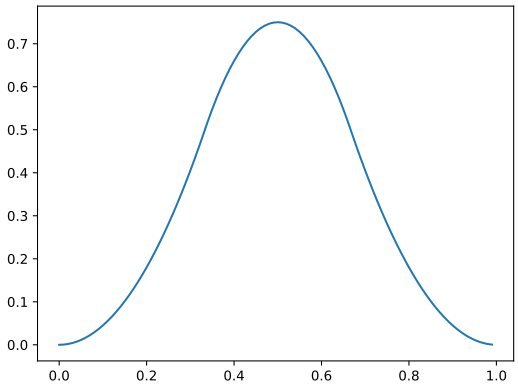

图 4: knots: [0, 1/3, 2/3, 1]

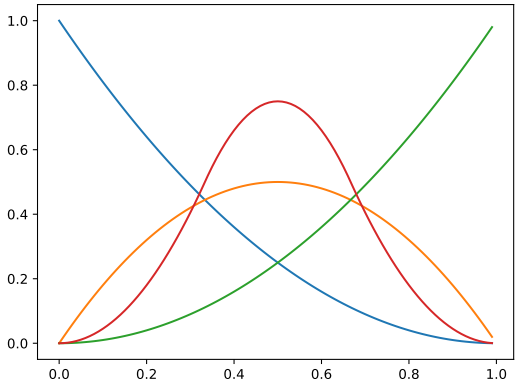

图 5: 上面四种 knots 图合在一起

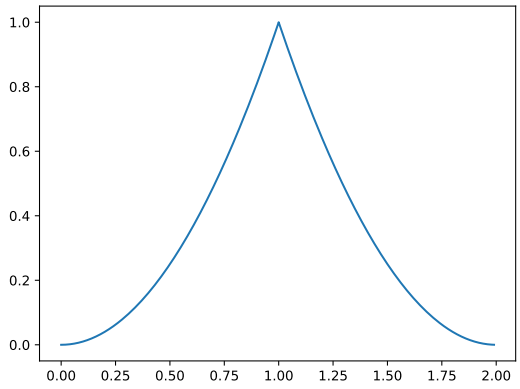

图 6: knots: [0, 1, 1, 2]

注意两种 knots: [0, 1, 1, 1] 和 [0, 1, 1, 2] 在 [0,1] 区间的图像是完全重合的。从而可以得知，对于 knots: [0, 0, 0, 1, 1, 2, 2, 2]，它在 [0, 1] 区间的图像和 [0, 0, 0, 1, 1, 1] 一致，在 [1, 2] 区间的图像和 [1, 1, 1, 2, 2, 2] 一致。它会经过第 4 个控制点。这个 knots 序列在表示 biarc 时会用到。它实际是 2 个 Bezier 曲线的拼接。

5 Fillet

3 个棱上的 fillet 相交的角上的 fillet 面生成算法：3 个棱上的 fillet 在 3 个面上分别确定 3 个交点。交点两两之间确定一条 pcurve，是 2d 的直线，作为棱上的 fillet 面上的参数曲线，这 3 条参数曲线形成 Coons patch，就是角上的 fillet 面。这个三角形的 Coons patch 的生成算法可以复用四边形的 Coons patch 算法，其中有一条边上的 2 个参数重合，作为一种退化的情况。

6 Surface/Surface intersection

occt 中的曲面交线算法：从本质上说，求解两个曲面的交线相当于将两个曲面的参数方程联立。但是 3 个方程要求解 4 个参数是无法求解的。所以 occ 先确定一个 isoparameter, 对这个参数手动增加一定的量，然后以增

量之后的 4 个参数为初始值，牛顿迭代法求解关于其余 3 个参数的一个解。由于 isoparameter 是手动增量过的，这个新的解必然“向前”走了一点距离。

以上就是 occ 的算法基本思路，涉及的源文件包括：`IntImp_int2S.gxx`, `IntWalk_PWalking.cxx`。

7 Constraint Solving

`https://en.wikipedia.org/wiki/Powell%27s_dog_leg_method`

7.1 数值解法

将几何约束表示成一个非线性方程组。

最简单的解法是牛顿迭代法。但是牛顿法要求方程数量和变量数量相等。而几何约束经常有过约束和欠约束的情况，所以将方程平方之后，变成一个求最小值的优化问题。可能的解法包括：Levenberg-Marquardt, BFGS, DogLeg 等。