Math Primitives and Algorithms

数学图元及其算法

eryar@163.com

一、概述 ***Overview***

Open CASCADE 中的数学图元及其算法包括以下内容：

- 向量和矩阵（***Vectors and matrices***）；
- 几何图元（***Geometric primitives***）；
- 数学算法（***Math algorithms***）；

二、向量和矩阵 ***Vectors and Matrices***

向量和矩阵部分提供了基本类型的矩阵和向量的 ***C***++实现，也可用来定义更复杂的数据结构。实数类型的矩阵和向量支持标准操作，如：加法、乘法、转置、求逆操作等。

向量和矩阵的维数是任意的，所以必须在定义的时候声明，以便分配存储空间。在声明后就不能对其维数进行修改。

Example

```
math_Vector v(1, 3);
// a vector of dimension 3 with range (1..3)

math_Matrix m(0, 2, 0, 2);
// a matrix of dimension 3x3 with range (0..2, 0..2)

math_Vector v(N1, N2);
// a vector of dimension N2-N1+1 with range (N1..N2)
```

向量和矩阵对象的本体和实体是相同的，即它们不能被共享且通过赋值操作来复制。

Example

```
math_Vector v1(1, 3), v2(0, 2);
v2 = v1;
// v1 is copied into v2. a modification of v1 does not
//affect v2
```

向量和矩阵的值可以通过索引来初始化和赋值，且其索引必须在维数即范围之内。

Example

```
math_Vector v(1, 3);
math_Matrix m(1, 3, 1, 3);
Standard_Real value;

v(2) = 1.0;
value = v(1);
m(1, 3) = 1.0;
value = m(2, 2);
```

有些向量和矩阵的操作是非法的，在这种情况下将会产生异常。两个标准的异常将会被使：

- ***Standard_DimensionError***：当两个矩阵或向量的操作涉及到的维数不匹配时产生；
- ***Standard_RangeError***：当要访问在矩阵或向量范围之外的数据时产生；

Example

```
math_Vector v1(1, 3), v2(1, 2), v3(0, 2);
v1 = v2;
// error: Standard_DimensionError is raised

v1 = v3;
// OK: ranges are not equal but dimensions are
// compatible

v1(0) = 2.0;
// error: Standard_RangeError is raised
```

三、几何图元类型 ***Primitive Geometric Types***

1．概述 ***Overview***

当要创建几何对象之前，你必须决定将要创建的对象是用于二维还是用于三维空间。包 ***gp*** 提供了通过值而不是引用控制的二维和三维类。当这类对象复制时，对象本体和实体是一致的。改变对象的一个实例不会影响其它的。

2．包 ***gp***

包 ***gp*** 定义了基本的非持久性几何实体，可用来进行代数计算和在二维和三维空间中基本的几何分析。也提供了基本的变换操作，如单位化、旋转、平移、镜像、缩放变换，及这些变换的组合。这些几何实体都是由值控制的。可用的几何实体如下所示；

- 二维和三维笛卡尔坐标（2***D*** & 3***D Cartesian cooridinates***）
- 矩阵（***Matrices***）
- 笛卡尔点（***Cartesian Points***）
- 向量（***Vector***）
- 方向（***Direction***）
- 轴（***Axis***）
- 直线（***Line***）
- 圆（***Circle***）
- 椭圆（***Ellipse***）
- 双曲线（***Hyperbola***）
- 抛物线（***Parabola***）
- 平面（***Plane***）
- 无限的柱面（***Infinite Cylindrical Surface***）
- 球面（***Spherical Surface***）
- 超环面（***Toroidal Surface***）
- 锥面（***Conical Surface***）

四、几何图元类型集合 ***Collections of Primitive Geometric Types***

若你不想使这些几何元素的单个实例而是他们的集合，包 ***TColgp*** 就是用来处理这类对象集合的。

- ***TColgp***：包 ***TColgp*** 提供了包 ***gp*** 中的类的 ***TCollection*** 的实例化，如：***XY***、***XYZ***、***Pnt***、***Pnt2d***、***Vec***、***Vec2d***、***Lin***、***Lin2d***、***Circ***、***Circ2d*** 等。它们都是非持久性类。

五、基本几何库 ***Basic Geometric Libraries***

有几个库可用来对曲线和曲面作基本的计算。若待处理的对象是由包 ***gp*** 中创建的，则初等曲线和曲面库—包 ***ElCLib*** 和包 ***ElSLib***—中很多算法是有用的。

包 ***Precision*** 描述了两个数比较时的精度标准。

- ***ElCLib***：（***Elementary Curves Libraries***）提供用于解析曲线的方法，可对 ***gp*** 包中曲线进行简单计算的库。根据给定的参数计算点或根据点计算参数。
- ***ElSLib***：（***Elementary Surfaces Libraries***）提供用于解析曲面的方法，可对 ***gp*** 包中的曲面进行简单计算。通过给定的两个参数计算点或计算一个点的参数。还提供了计算曲线和曲面法向的库。
- ***Bnd***：（***Bounding Boxes***）二维或三维空间中的包围盒。

六、通用数学算法 ***Common Math Algorithms***

通用数学算法库提供了 ***C***++实现的最常用的数学算法，包括以下内容：

- 求解线性方程组的算法
- 求函数极值的算法
- 求非线性方程（组）的根的算法
- 求方阵特征值和特征向量的算法

算法的实现 ***Implementation of Algorithms***

所有数学算法的实现原则是相同的，他们包含：

- 构造函数：执行几乎所有的计算，或给出合适参数。所有相关信息保存在结果对象中，所以使后期计算更高效。
- 函数 ***IsDone***：这个函数返回计算的成功与否。
- 一些特定函数：对每个算法特定的函数，用来确保得到各种结果。这些函数只有在 ***IsDone*** 返回 ***true*** 时才可能调用，否则将会产生 ***StdFail_NotDone*** 异常。

下面的例子演示的是使用高斯方法求解线性方程组的代码。代码截取自类 ***math_Gauss*** 的头文件。

Example

```
class Gauss {
public:
  Gauss (const math_Matrix& A);
  Standard_Boolean IsDone() const;
  void Solve (const math_Vector& B,
  math_Vector& X) const;
};
```

使用高斯方法求解方程组的主程序如下：

Example

```
#include <math_Vector.hxx> #include <math_Matrix.hxx>
main ()
{
  math_Vector a(1, 3, 1, 3);
  math_Vector b1(1, 3), b2(1, 3);
  math_Vector x1(1, 3), x2(1, 3);
  // a, b1 and b2 are set here to the appropriate values
  math_Gauss sol(a);          // computation of the
  // LU decomposition of A
  if(sol.IsDone()) {          // is it OK ?
    sol.Solve(b1, x1);        // yes, so compute x1
    sol.Solve(b2, x2);        // then x2
    ...
  }
  else {                      // it is not OK:
    // fix up
    sol.Solve(b1, x1);        // error:
    // StdFail_NotDone is raised
  }
}
```