**有限元空间描述**

这个类用于描述有限元空间，有一个抽象基类FiniteElement, 由此派生出具体的有限元类。

实际上，这些类必须实现的函数提供了在参考单元上的给定点处的形函数值(或导数值)。为了能够计算矩阵中的积分及右端项积分等，还必须能够把这些形函数或导数值映射到真实网格上去。而这是通过Mapping基类的派生类并结合FEValues类来实现的。

**向量有限元**

deal.II提供两种不同的向量有限元。首先，有一组真正意义上的“向量”有限元，也就是说，向量的每个分量由一组不同的各项异性多项式组成。这些有限元通常具有各异的形式。目前，有如下这些：

* FE\_ABF
* FE\_BDM, FE\_DGBDM
* FE\_Nedelec, FE\_DGNedelec
* FE\_RaviartThomas, FE\_DGRaviartThomas

此外，deal.II还提供了从已有的标量或向量有限元创建新的向量有限元的机制。FESystem类就是为此服务的：它本身并不对形函数进行描述，而是由其他有限元对象组装出一个向量有限元。

注：向量有限元的实现是由FE\_PolyTensor类提供的。通常，新的向量有限元需要又这个类派生出。

**间断伽辽金**

对每种从属于弱可微函数空间的有限元(如H1或Hcurl函数)，我们可以简单地通过把所有位于角点或边或面上的自由度分配到网格内部去，从而定义出类似的DG空间。这需要在拓扑意义下进行理解。对于这种自由度的插值算子仍将位于边界上。虽然在dealii中所完成得不一定是这样，但我们提供了相当数量的这种有限元，以及，如FE\_DGP。下面是一个当前DG有限元的列表：

* 标量：FE\_DGP, FE\_DGQ
* 标量，不同的形函数：FE\_DGPMonomial, FE\_DGPNonparametric, FE\_DGQArbitraryNodes
* 向量：FE\_DGBDM, FE\_DGNedelec, FE\_DGRaviartThomas

注：向量DG有限元的实现是由FE\_DGVector类提供的，只需提供向量多项式空间即可。继承于这个类的实际类只需要实现一个构造函数和FiniteElement::get\_name()即可。

**template<int dim, int spacedim=dim>**

**class FE\_Q<dim, spacedim>**

这个类实现了标量拉格朗日有限元Qp，从而产生了在每个坐标方向上是p次的连续、分片多项式。这个类的实现依赖于基于1D拉格朗日多项式的**张量积多项式**，支持点为等距的(至多2次)，或Gauss-Lobatto(从3次开始)，或人为给定。

这个类的标准构造函数需要多项式次数p作参数。或者，也可以输入一个积分公式定义了一个方向上的拉格朗日插值函数支持点。

FE\_Q<dim, spacedim>::FE\_Q(const unsigned int p)

p次多项式构成的张量积，基于Gauss-Lobatto节点。对于一次或二次多项式，则是普通等距节点。

FE\_Q<dim, spacedim>::FE\_Q(const Quadrature<1>& points)

张量积多项式，有支持点points是基于一维积分公式的。有限元的次数是points.size()-1。注意，第一个点必须是0，最后一个必须是1。构造FE\_Q<dim> (QgaussLobatto<1>(fe\_degree+1))与只指定多项式次数的构造函数是等价的。对于fe\_degree>2的有限元，如果要使用等距节点，则构造FE\_Q<dim>(Qiterated<1>(Qtrapez<1>(), fe\_degree))

//--------------------------------------------插叙：张量积多项式------------------------------------------------------

**template<int dim, typename PolynomialType = Polynomials::Polynomial<double>>**

**class TensorProductPolynomials< dim, PolynomialType >**

这个类生成给定多项式的张量积

给定由n个一维多项式P1到Pn构成的向量，这个类生成ndim个形式为的多项式。如果基多项式在区间[-1,-1]或[0,1]上是互相正交的，则张量积多项式在[-1,1]dim或[0,1]dim上是正交的。

索引顺序如下：x坐标上的多项式先循环，然后y坐标的多项式，然后z的。 则前几项多项式为：P1(x)P1(y), P2(x)P1(y), P3(x)P1(y), ..., P1(x)P2(y), P2(x)P2(y), P3(x)P2(y), ...

output\_indices()函数可打印出dim维多项式的顺序，即，对于每个多项式，它会提供x，y，z方向上的一维多项式的指标i，j，k。dim维多项式的顺序可以通过函数set\_numbering()来改变。

//-----------------------------------------------------------------------------------------------------------------------------

**Implementation**

构造函数创建一个TensorProductPolynomials对象，它包含了p次等距朗格朗日多项式的张量积。这个TensorProductPolynomials对象提供了所有形函数的值(及导数值)。

**Unit support point distribution and conditioning of interpolation**

当构造多项式次数为一次或者二次的FE\_Q对象时，用到的是等距支持点，线性为0,1点，二次为0,0.5,1点。单位支持点(或nodal points) 是那些在该点上的第j个拉格朗日多项式满足性质的点，即：在该点处的多项式值为1，而在其他所有点处多项式值为0。对于更高次数的多项式，默认的支持点就不是等距的了，而是选择了degree+1阶的Gauss-Lobatto积分公式的积分点作支持点。这种支持点的分布可以在任意多项式次数下产生良态的拉格朗日插值性质。相反地，如果采用等距点分布，则随着次数增加，多项式插值性质会变得病态。在插值中，这种现象称为龙格现象。对于伽辽金方法，龙格现象不是体现在解的质量上，而是体现在系统矩阵的条件数上。例如，对于10次多项式，若采用等距点，则生成的单元质量矩阵会有条件数2.6e6，而采用Gauss-Lobatto点的条件数为400.

一维情况下的Gauss-Lobatto点包括单位区间的两个端点0和+1。内部点则逐渐往端点偏移，使得支持点分布会趋于汇聚在单元边界上。

如果与Gauss-Lobatto积分公式配合使用，那么基于默认支持点的FE\_Q有限元会产生对角型的质量矩阵。这在step-48中举例说明了。然而，这种有限元也可以配合任意的积分公式进行使用，在最一般的情形下，生成的质量矩阵不是对角型的。

**Numbering of the degree of freedom(DoFs)**

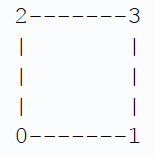
自由度顺序按如下：

**Q1单元：**

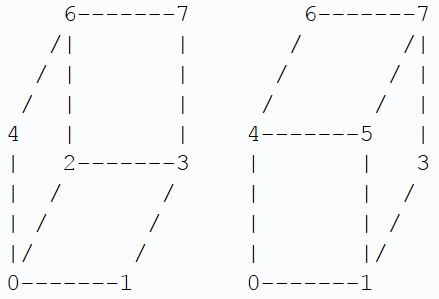
* 一维情况:

C:\Users\zeng\AppData\Roaming\Tencent\Users\154462188\QQ\WinTemp\RichOle\U8M_RVJU{GAZLL(E5SRP{6L.png

* 二维情况:



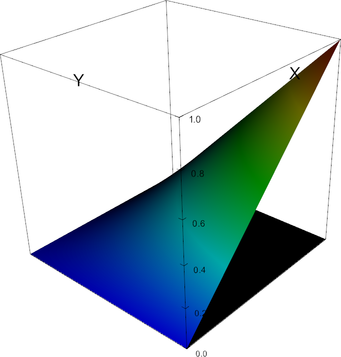
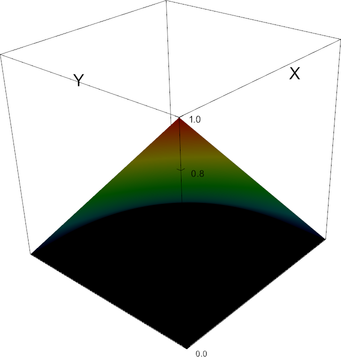
* 三维情况:

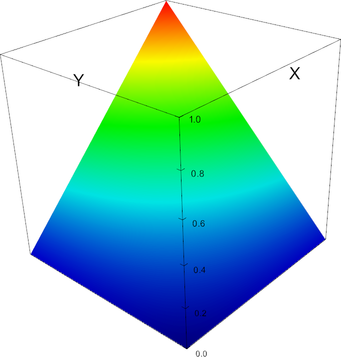
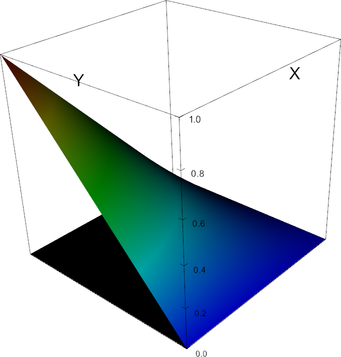


形函数相应的支持点坐标值是:

* + Shape function 0: [0, 0, 0];
  + Shape function 1: [1, 0, 0];
  + Shape function 2: [0, 1, 0];
  + Shape function 3: [1, 1, 0];
  + Shape function 4: [0, 0, 1];
  + Shape function 5: [1, 0, 1];
  + Shape function 6: [0, 1, 1];
  + Shape function 7: [1, 1, 1];

在二维情况下，这些形函数如下图所示：





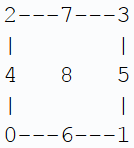
Q1单元，依次为形函数0-1-2-3

**Q2单元：**

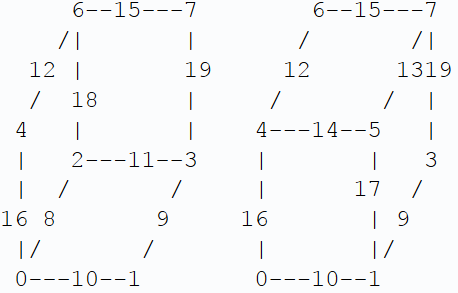
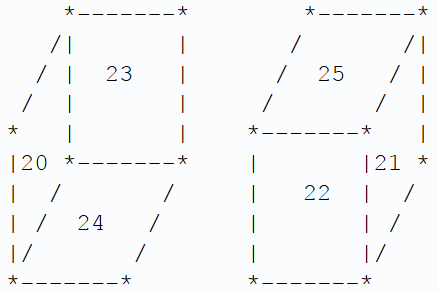
* 一维情况:

C:\Users\zeng\AppData\Roaming\Tencent\Users\154462188\QQ\WinTemp\RichOle\F[0AK57A1(6IE8F5XG}SWYH.png

* 二维情况:



* 三维情况:

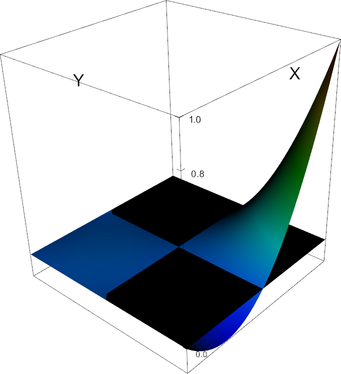
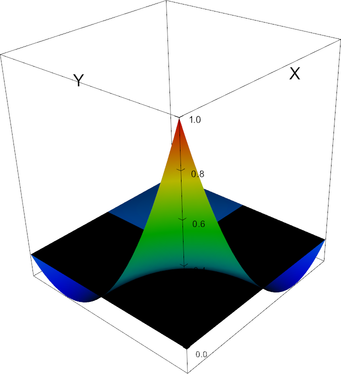
 

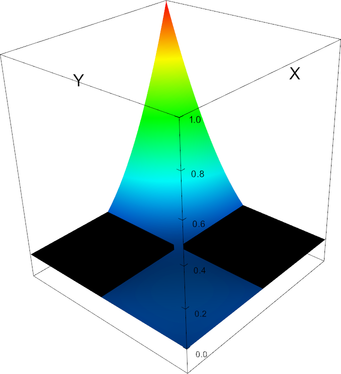
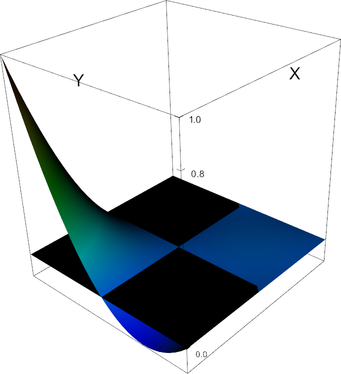
中心角点为26。

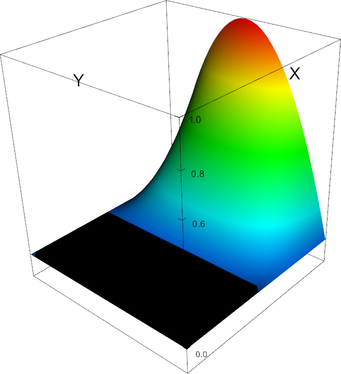
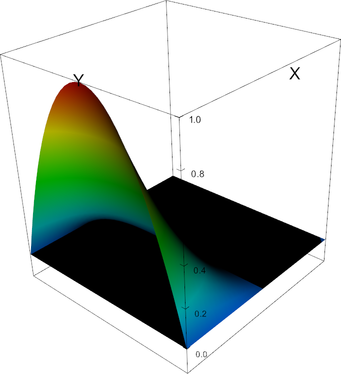
形函数相应的支持点坐标值是:

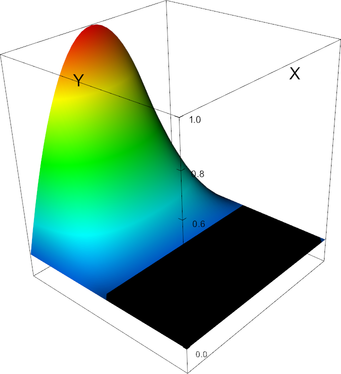
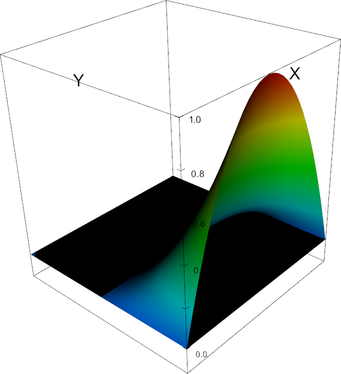
* + Shape function 0: [0, 0, 0];
  + Shape function 1: [1, 0, 0];
  + Shape function 2: [0, 1, 0];
  + Shape function 3: [1, 1, 0];
  + Shape function 4: [0, 0, 1];
  + Shape function 5: [1, 0, 1];
  + Shape function 6: [0, 1, 1];
  + Shape function 7: [1, 1, 1];
  + Shape function 8: [0, 1/2, 0];
  + Shape function 9: [1, 1/2, 0];
  + Shape function 10: [1/2, 0, 0];
  + Shape function 11: [1/2, 1, 0];
  + Shape function 12: [0, 1/2, 1];
  + Shape function 13: [1, 1/2, 1];
  + Shape function 14: [1/2, 0, 1];
  + Shape function 15: [1/2, 1, 1];
  + Shape function 16: [0, 0, 1/2];
  + Shape function 17: [1, 0, 1/2];
  + Shape function 18: [0, 1, 1/2];
  + Shape function 19: [1, 1, 1/2];
  + Shape function 20: [0, 1/2, 1/2];
  + Shape function 21: [1, 1/2, 1/2];
  + Shape function 22: [1/2, 0, 1/2];
  + Shape function 23: [1/2, 1, 1/2];
  + Shape function 24: [1/2, 1/2, 0];
  + Shape function 25: [1/2, 1/2, 1];
  + Shape function 26: [1/2, 1/2, 1/2];

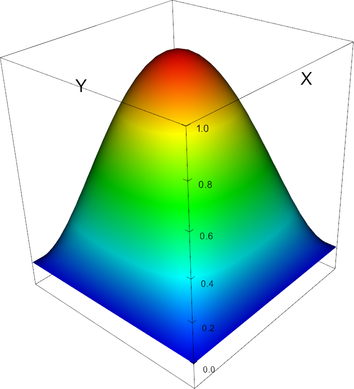
在二维情况下，这些形函数如下图所示（空白平面对应0值；负值形函数可能无法显示出来）：











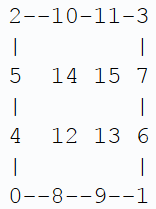
Q2单元，形函数依次为：0-1-2-3-4-5-6-7-8

**Q3单元：**

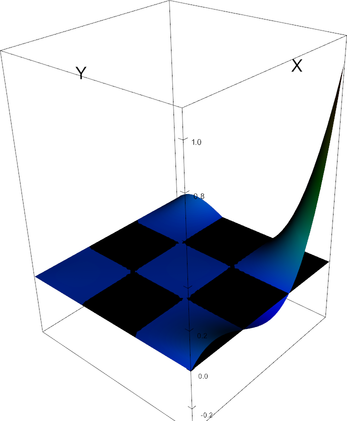
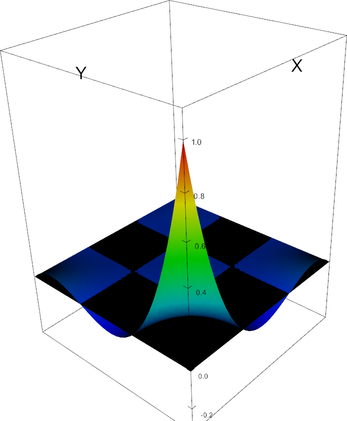
* 一维情况:

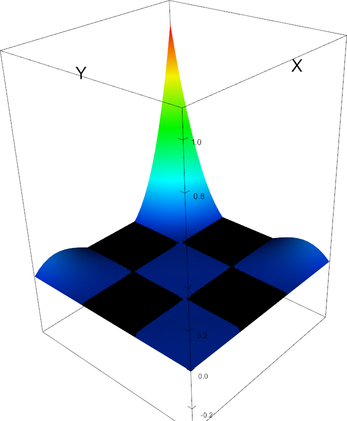
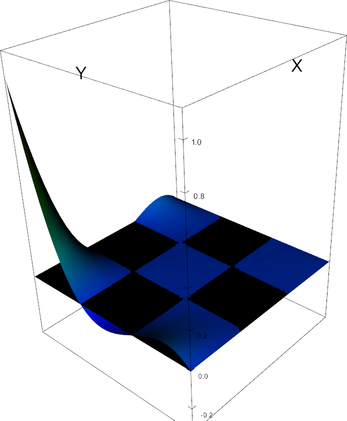
C:\Users\zeng\AppData\Roaming\Tencent\Users\154462188\QQ\WinTemp\RichOle\7N)S8QZK%@({CR(RHAZWUHN.png

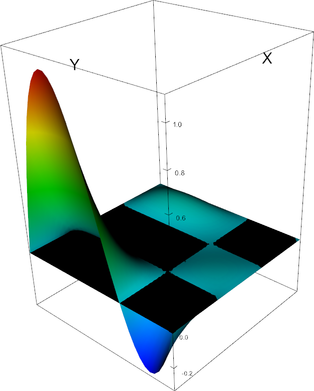
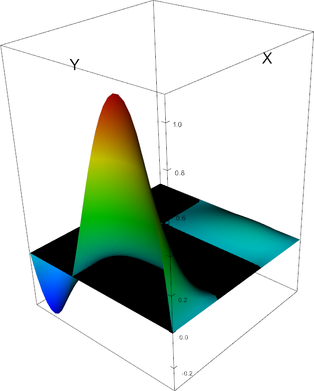
* 二维情况:

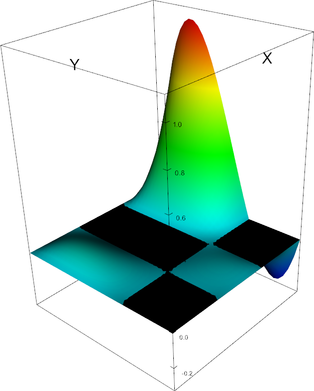
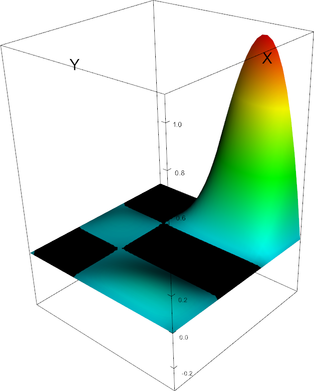


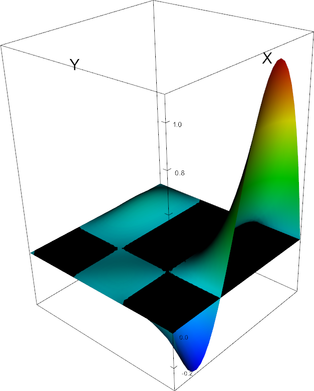
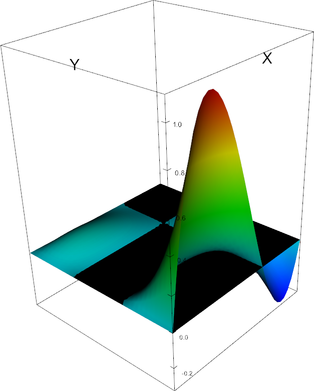
在二维情况下，这些形函数如下图所示：

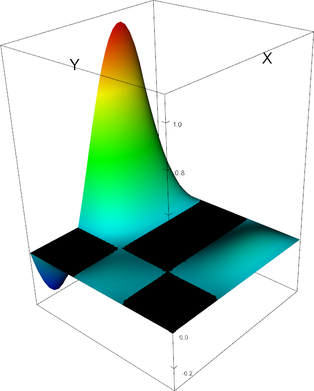
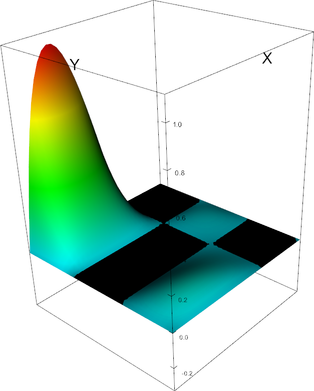


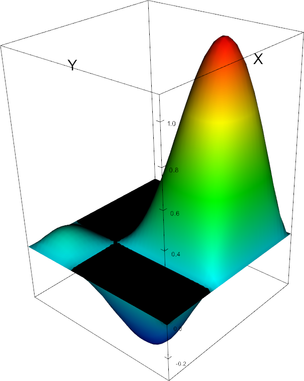
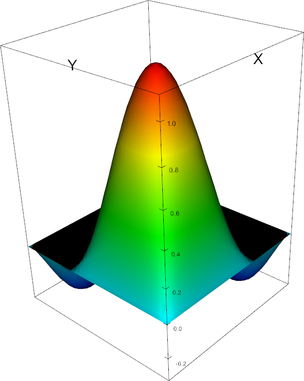


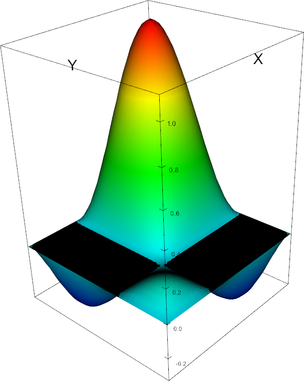
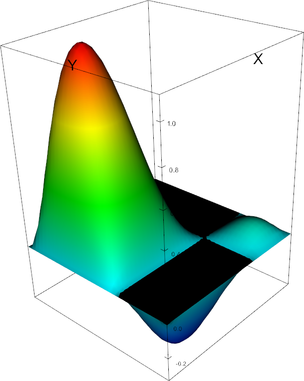












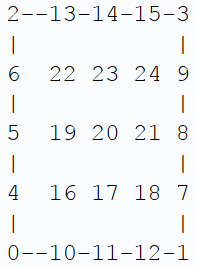
Q3单元，形函数依次为：0——15

**Q4单元：**

* 一维情况:

C:\Users\zeng\AppData\Roaming\Tencent\Users\154462188\QQ\WinTemp\RichOle\UQGRBX4$()M~LB33XF16YJS.png

* 二维情况:



图像略。