FEALPy 偏微分方程数值解程序设计与实现: 任意次任意维的拉格朗日有限元空间

魏华祎

weihuayi@xtu.edu.cn

湘潭大学 • 数学与计算科学学院

July 17, 2020

Outline

Outline

单纯形上的重心坐标函数

记 $\{\boldsymbol{x}_i := [x_{i,0}, x_{i,1}, \ldots, x_{i,d-1}]\}_{i=0}^{d}$ 为 $\mathbb{R}^d$ 空间中的一组点，假设它们不在同一个超平面上，也即是说 d 个向量 $\boldsymbol{x}_0\boldsymbol{x}_1$，$\boldsymbol{x}_0\boldsymbol{x}_2$，$\cdots$，和 $\boldsymbol{x}_0\boldsymbol{x}_d$ 是线性无关的，等价于矩阵

$$\boldsymbol{A} = \begin{bmatrix} x_{0,0} & x_{1,0} & \cdots & x_{d,0} \\ x_{0,1} & x_{1,1} & \cdots & x_{d,1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{0,d-1} & x_{1,d-1} & \cdots & x_{d,d-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

是非奇异的.

单纯形上的重心坐标函数

给定任一点 $\boldsymbol{x} = [x_0, x_1, \cdots, x_{d-1}]^T \in \mathbb{R}^d$, 求解如下线性代数系统, 可得一组实数值 $\boldsymbol{\lambda} := [\lambda_0(\boldsymbol{x}), \lambda_1(\boldsymbol{x}), \cdots, \lambda_d(\boldsymbol{x})]^T$:

$$\boldsymbol{A}\boldsymbol{\lambda} = \boldsymbol{x},$$

满足如下性质

$$\boldsymbol{x} = \sum_{i=0}^{d} \lambda_i(\boldsymbol{x})\boldsymbol{x}_i, \quad \sum_{i=0}^{d} \lambda_i(\boldsymbol{x}) = 1.$$

点集 $\{\boldsymbol{x}_i\}_{i=0}^d$ 形成的凸壳

$$\tau = \{\boldsymbol{x} = \sum_{i=0}^{d} \lambda_i \boldsymbol{x}_i | 0 \le \lambda_i \le 1, \sum_{i=0}^{d} \lambda_i = 1\}$$

称为一个**几何 d-单纯形**. $\boldsymbol{\lambda}$ 称为 $\boldsymbol{x}$ 对应的**重心坐标**向量.

单纯形上的重心坐标函数

易知，$\lambda_0(\boldsymbol{x})$，$\lambda_1(\boldsymbol{x})$，$\cdots$，和 $\lambda_d(\boldsymbol{x})$ 是关于 $\boldsymbol{x}$ 线性函数，且

$$\lambda_i(\boldsymbol{x}_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}, i, j = 0, \cdots, d$$

区间单元 $[x_0, x_1]$ 上的重心坐标函数

给定区间单元上的一个重心坐标 (λ_0, λ_1)，存在 $x \in [x_0, x_1]$，使得：

$$\lambda_0 := \frac{x_1 - x}{x_1 - x_0}, \quad \lambda_1 := \frac{x - x_0}{x_1 - x_0}$$

显然

$$\lambda_0 + \lambda_1 = 1$$

重心坐标关于 x 的导数为：

$$\frac{\mathrm{d}\lambda_0}{\mathrm{d}x} = -\frac{1}{x_1 - x_0}, \quad \frac{\mathrm{d}\lambda_1}{\mathrm{d}x} = \frac{1}{x_1 - x_0}$$

三角形单元 $[x_0, x_1, x_2]$ 上的重心坐标函数

因为 $\lambda_0, \lambda_1, \lambda_2$ 是关于 $\boldsymbol{x}$ 线性函数, 它梯度分别为:

$$\nabla\lambda_0 = \frac{1}{2|\tau|}(\boldsymbol{x}_2 - \boldsymbol{x}_1)\boldsymbol{W}$$
$$\nabla\lambda_1 = \frac{1}{2|\tau|}(\boldsymbol{x}_0 - \boldsymbol{x}_2)\boldsymbol{W}$$
$$\nabla\lambda_2 = \frac{1}{2|\tau|}(\boldsymbol{x}_1 - \boldsymbol{x}_0)\boldsymbol{W}$$

其中

$$\boldsymbol{W} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

注意这里的 $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, 和 $\boldsymbol{x}_2$ 是行向量。

三角形单元上的重心坐标函数的梯度计算

Python code 1 重心坐标梯度的计算代码。

```python
import numpy as np
from fealpy.mesh import MeshFactory
mf = MeshFactory()
box = [0, 1, 0, 1]
mesh = mf.boxmesh2d(box, nx=1, ny=1, meshtype='tri')
NC = mesh.number_of_cells()

node = mesh.entity('node')
cell = mesh.entity('cell')
v0 = node[cell[:, 2], :] - node[cell[:, 1], :] # x_2 - x_1
v1 = node[cell[:, 0], :] - node[cell[:, 2], :] # x_0 - x_2
v2 = node[cell[:, 1], :] - node[cell[:, 0], :] # x_1 - x_0
nv = np.cross(v2, -v1)

Dlambda = np.zeros((NC, 3, 2), dtype=np.float64)
length = nv #
W = np.array([[0, 1], [-1, 0]], dtype=np.int_)
Dlambda[:,0,:] = v0@W/length.reshape(-1, 1)
Dlambda[:,1,:] = v1@W/length.reshape(-1, 1)
Dlambda[:,2,:] = v2@W/length.reshape(-1, 1)
```

作业

(1) 给定一个一维区间网格, 计算每个单元上的重心坐标函数的导数。

```python
import numpy as np
from fealpy.mesh import IntervalMesh
node = np.array([[0.0], [0.5], [1.0]], dtype=np.float64)
cell = np.array([[0, 1], [1, 2]], dtype=np.int_)
mesh = IntervalMesh(node, cell)
```

(2) 给定一个三维四面体网格, 计算每个单元上的重心坐标函数的梯度。

```python
import numpy as np
from fealpy.mesh import MeshFactory

mf = MeshFactory()
mesh = mf.one_tetrahedron_mesh(meshtype='iso')
```

Outline

$d+1$ 维多重指标

记 $\boldsymbol{m}$ 为 $d+1$ 维多重指标向量 $[m_0, m_1, \cdots, m_d]$, 满足

$$m_i \geq 0, i = 0, 1, \cdots, d, \text{ and } \sum_{i=0}^{d} m_i = p.$$

固定次数 p, $\boldsymbol{m}$ 的所有可能取值个数为

$$n_p := \binom{d}{p+d} = \begin{cases} p+1, & 1D \\ \frac{(p+1)(p+2)}{2}, & 2D \\ \frac{(p+1)(p+2)(p+3)}{6}, & 3D \end{cases}$$

多重指标向量 m 编号规则

记 α 为多重向量指标 $\boldsymbol{m}$ 的一个从 0 到 $n_p - 1$ 一维编号, 编号规则如下:

α	$\boldsymbol{m}_\alpha$				
0	p	0	0	$\cdots$	0
1	p-1	1	0	$\cdots$	0
2	p-1	0	1	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
d	p-1	0	0	$\cdots$	1
d+1	p-2	2	0	$\cdots$	0
d+2	p-2	1	1	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
2d-1	p-2	1	0	$\cdots$	1
2d	p-2	0	2	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
n_p	0	0	0	$\cdots$	p

Table: 多重指标 $\boldsymbol{m}_\alpha$ 的编号规则.

高次拉格朗日形函数的一般公式

给定第 α 个多重指标向量 $\boldsymbol{m}_\alpha$, 在 d-单纯形 τ 上可以构造如下的 p 次多项式函数:

$$\phi_\alpha = \frac{1}{\boldsymbol{m}_\alpha!} \prod_{i=0}^{d} \prod_{j_i=0}^{m_i-1} (p\lambda_i - j_i). \tag{1}$$

其中

$$\boldsymbol{m}_\alpha! = m_0!m_1!\cdots m_d!, \qquad \prod_{j_i=0}^{-1} (p\lambda_i - j_i) = 1, \quad i = 0, 1, \cdots, d$$

Sylvester's Formula

$$R_i(p,\lambda) = \begin{cases} \frac{1}{i!}\prod_{j_i=0}^{i-1}(p\lambda - j_i), & 1 \le i \le p \\ 1, & i = 0 \end{cases}$$

d-单纯形上的插值点

每个多重指标 $\boldsymbol{m}_\alpha$, 都对应 d-单纯形 τ 上的一个点 $\boldsymbol{x}_\alpha$,

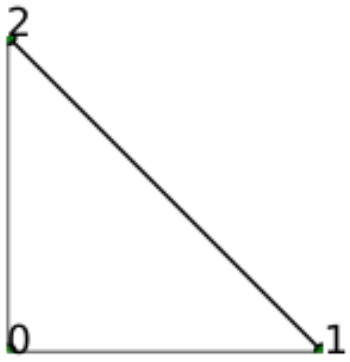
$$\boldsymbol{x}_\alpha = \sum_{i=0}^{d} \frac{m_i}{p} \boldsymbol{x}_i.$$

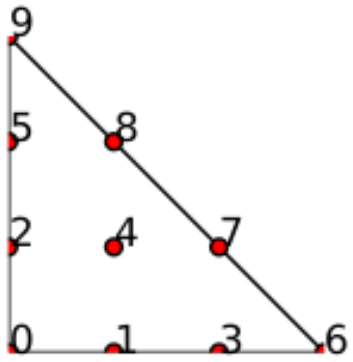
其中 m_i 是多重指标向量 $\boldsymbol{m}_\alpha$ 的第 i 个分量. 易知 $\boldsymbol{x}_\alpha$ 是 ϕ_α 对应的插值点, 满足

$$\phi_\alpha(\boldsymbol{x}_\beta) = \begin{cases} 1, & \alpha = \beta \\ 0, & \alpha \neq \beta \end{cases} \text{和} \alpha, \beta = 0, 1, \cdots, n_p - 1 \qquad (2)$$

插值点的局部编号规则

$$\phi_{m,n,k} = \frac{1}{m!n!k!} \prod_{j_0=0}^{m-1}(p\lambda_0 - j_0) \prod_{j_1=0}^{n-1}(p\lambda_1 - j_1) \prod_{j_2=0}^{k-1}(p\lambda_2 - j_2).$$



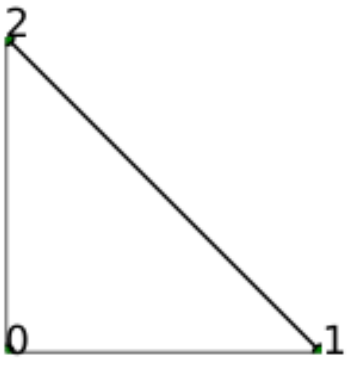


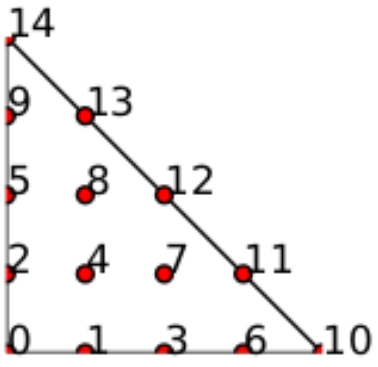
	m	n	k
0:	3	0	0
1:	2	1	0
2:	2	0	1
3:	1	2	0
4:	1	1	1
5:	1	0	2
6:	0	3	0
7:	0	2	1
8:	0	1	2
9:	0	0	3

Figure: 三角形上的 $p = 3$ 次形函数对应的编号规则.

插值点的局部编号规则

$$\phi_{m,n,k} = \frac{1}{m!n!k!} \prod_{j_0=0}^{m-1}(p\lambda_0 - j_0) \prod_{j_1=0}^{n-1}(p\lambda_1 - j_1) \prod_{j_2=0}^{k-1}(p\lambda_2 - j_2).$$





	m	n	k
0:	4	0	0
1:	3	1	0
2:	3	0	1
3:	2	2	0
4:	2	1	1
5:	2	0	2
6:	1	3	0
7:	1	2	1
8:	1	1	2
9:	1	0	3
10:	0	4	0
11:	0	3	1
12:	0	2	2
13:	0	1	3
14:	0	0	4

Figure: 三角形上的 $p = 4$ 次形函数对应的编号规则.

插值点的局部编号规则

Figure: 区间上的 $p = 4$ 次形函数对应的编号规则

插值点的局部编号规则

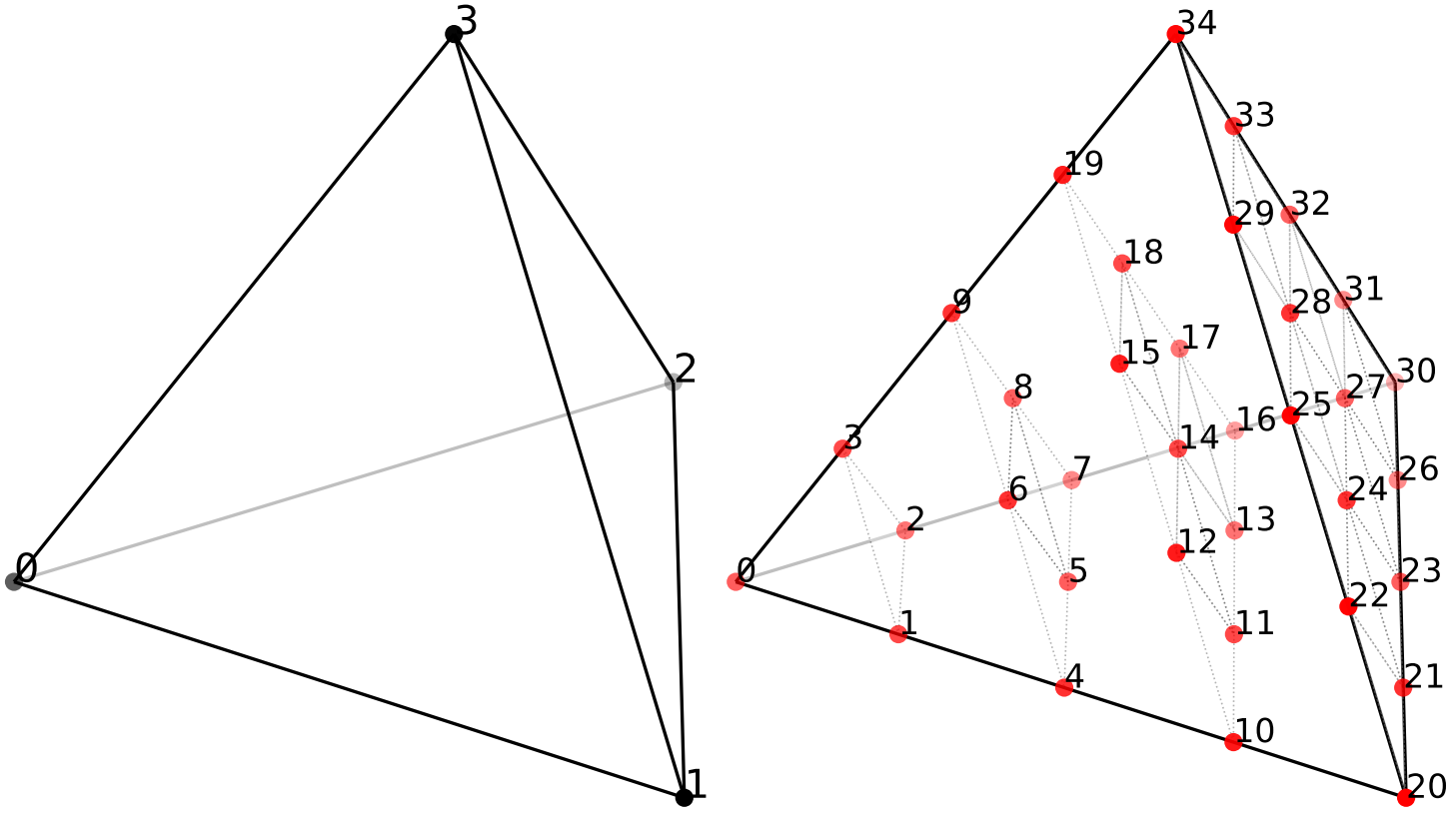


Figure: 四面体上的 $p = 4$ 次形函数对应的编号规则.

ϕ_α 的数组化计算过程

首先构造向量和矩阵

$$P = \left[\frac{1}{0!}, \frac{1}{1!}, \frac{1}{2!}, \cdots, \frac{1}{p!}\right],$$

$$\boldsymbol{A} := \begin{bmatrix} 1 & 1 & \cdot & 1 \\ p\lambda_0 & p\lambda_1 & \cdots & p\lambda_d \\ p\lambda_0 - 1 & p\lambda_1 - 1 & \cdots & p\lambda_d - 1 \\ \vdots & \vdots & \ddots & \vdots \\ & & \vdots & \\ p\lambda_0 - (p-1) & p\lambda_1 - (p-1) & & p\lambda_d - (p-1) \end{bmatrix},$$

Remark

$$\phi_\alpha = \frac{1}{\boldsymbol{m}_\alpha!} \prod_{j_0=0}^{m_0-1} (p\lambda_0 - j_0) \prod_{j_1=0}^{m_1-1} (p\lambda_1 - j_1) \cdots \prod_{j_d=0}^{m_d-1} (p\lambda_d - j_d)$$

ϕ_α 的数组化计算过程

$$\boldsymbol{B} = \operatorname{diag}(\boldsymbol{P}) \begin{bmatrix} 1 & 1 & \cdots & 1 \\ p\lambda_0 & p\lambda_1 & \cdots & p\lambda_d \\ \prod_{j_0=0}^{1}(p\lambda_0 - j_0) & \prod_{j_1=0}^{1}(p\lambda_1 - j_1) & \cdots & \prod_{j_d=0}^{1}(p\lambda_d - j_d) \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{j_0=0}^{p-1}(p\lambda_0 - j_0) & \prod_{j_1=0}^{p-1}(p\lambda_1 - j_1) & \cdots & \prod_{j_d=0}^{p-1}(p\lambda_d - j_d) \end{bmatrix}$$

Remark

$$\phi_\alpha = \frac{1}{\boldsymbol{m}_\alpha!} \prod_{j_0=0}^{m_0-1}(p\lambda_0 - j_0) \prod_{j_1=0}^{m_1-1}(p\lambda_1 - j_1) \cdots \prod_{j_d=0}^{m_d-1}(p\lambda_d - j_d)$$

ϕ_α 的数组化计算过程

$$\boldsymbol{B} = \operatorname{diag}(\boldsymbol{P}) \begin{bmatrix} 1 & 1 & \cdots & 1 \\ p\lambda_0 & p\lambda_1 & \cdots & p\lambda_d \\ \prod_{j_0=0}^{1}(p\lambda_0 - j_0) & \prod_{j_1=0}^{1}(p\lambda_1 - j_1) & \cdots & \prod_{j_d=0}^{1}(p\lambda_d - j_d) \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{j_0=0}^{p-1}(p\lambda_0 - j_0) & \prod_{j_1=0}^{p-1}(p\lambda_1 - j_1) & \cdots & \prod_{j_d=0}^{p-1}(p\lambda_d - j_d) \end{bmatrix}$$

注意 $\boldsymbol{B}$ 包含 ϕ_α 的所有计算模块，可重写 ϕ_α 为如下形式：

$$\phi_\alpha = \prod_{i=0}^{d} \boldsymbol{B}[m_i, i]$$

其中 m_i 为 $\boldsymbol{m}_\alpha$ 的第 i 个分量.

任意次任意维拉格朗日有限元基函数代码实现

Python code 2 拉格朗日有限元基函数计算展示代码。

```python
def basis(self, bc): # bc.shape = (NQ, TD+1)
    p = self.p   # The degree of polynomial basis function
    TD = self.TD # The toplogy dimension of the mesh
    multiIndex = self.dof.multiIndex # The multiindex matrix of m_α

    # construct vector P = (1/1!, 1/2!, ···, 1/p!).
    c = np.arange(1, p+1, dtype=np.int)
    P = 1.0/np.multiply.accumulate(c)

    # construct the matrix A.
    t = np.arange(0, p)
    shape = bc.shape[:-1]+(p+1, TD+1)
    A = np.ones(shape, dtype=self.ftype)
    A[..., 1:, :] = p*bc[..., np.newaxis, :] - t.reshape(-1, 1)

    # construct matrix B and here we still use the memory of A
    np.cumprod(A, axis=-2, out=A)
    A[..., 1:, :] *= P.reshape(-1, 1)

    # compute φ_α
    idx = np.arange(TD+1)
    phi = np.prod(A[..., multiIndex, idx], axis=-1)
    return phi[..., np.newaxis, :] # (NQ, 1, ldof)
```

注意这里为清晰的展示代码, 去掉了一些非必要的代码, 全部代码见 fealpy/functionspace/LagrangeFiniteELementSpace.py。

$\nabla\phi_\alpha$ 的数组化计算过程

为计算 $\nabla\phi_\alpha$，首先需要用到函数乘积求导法则来计算 $\prod_{j_i=0}^{m_i-1}(p\lambda_i - j_i)$ 的导数，即

$$\nabla \prod_{j_i=0}^{m_i-1}(p\lambda_i - j_i) = p\sum_{j_i=0}^{m_i-1}\prod_{0\le k\le m_i-1, k\neq j_i}(p\lambda_i - k)\nabla\lambda_i.$$

Remark

这是一种标量的表达方式!

Remark

$$\phi_\alpha = \frac{1}{\boldsymbol{m}_\alpha!}\prod_{j_0=0}^{m_0-1}(p\lambda_0 - j_0)\prod_{j_1=0}^{m_1-1}(p\lambda_1 - j_1)\cdots\prod_{j_d=0}^{m_d-1}(p\lambda_d - j_d)$$

$\nabla\phi_\alpha$ 的数组化计算过程

用数组化的方式, 需要首先构造 $d+1$ 阶矩阵

$$\boldsymbol{D}^i = \begin{pmatrix} p & p\lambda_i - 0 & \cdots & p\lambda_i - 0 \\ p\lambda_i - 1 & p & \cdots & p\lambda_i - 1 \\ \vdots & \vdots & \ddots & \vdots \\ p\lambda_i - (p-1) & p\lambda_i - (p-1) & \cdots & p \end{pmatrix}, \quad 0 \le i \le d,$$

把 $\boldsymbol{D}^i$ 的每一列做累乘运算, 然后取其下三角矩阵, 再每一行求和, 即可得到矩阵 $\boldsymbol{B}$ 的每一列各个元素的求导后系数. 可得到矩阵 $\boldsymbol{D}$, 其元素定义为

$$D_{i,j} = \sum_{m=0}^{j} \prod_{k=0}^{j} D_{k,m}^{i}, \quad 0 \le i \le d, \text{ and } 0 \le j \le p-1.$$

$\nabla\phi_\alpha$ 的数组化计算过程

进而可以计算 $\nabla \boldsymbol{B}$

$$
\begin{aligned}
\nabla \boldsymbol{B} =& \operatorname{diag}(\boldsymbol{P}) \begin{pmatrix} 0 & 0 & \cdots & 0 \\ D_{0,0}\nabla\lambda_0 & D_{1,0}\nabla\lambda_1 & \cdots & D_{d,0}\nabla\lambda_d \\ \vdots & \vdots & \ddots & \vdots \\ D_{0,p-1}\nabla\lambda_0 & D_{1,p-1}\nabla\lambda_1 & \cdots & D_{d,p-1}\nabla\lambda_d \end{pmatrix} \\
=& \operatorname{diag}(\boldsymbol{P}) \begin{pmatrix} \mathbf{0} \\ \boldsymbol{D} \end{pmatrix} \begin{pmatrix} \nabla\lambda_0 & & & \\ & \nabla\lambda_1 & & \\ & & \ddots & \\ & & & \nabla\lambda_d \end{pmatrix}
\end{aligned}
$$

数组化编程的一些总结

- 算法的数组化表达是数组化编程的基础和关键。数组化的表达要明确：
  - 算法的输入数组是什么？
  - 算法最终期望得到的数组是什么？
  - 算法中间需要的数组运算是什么？
- 代数学是算法数组化表达的有力工具。

Outline

有限维空间的实现关键

Galerkin 类型的离散方法，其程序实现的关键是**有限维空间的实现**，而有限维空间的实现的关键是

- 基函数及其导数计算
  - 主要是计算空间基函数及其导数在数值积分点处的值。
- 全局和局部自由度的管理
  - 全局自由度的编号规则，这与最终的离散代数系统中矩阵的行号和列号相对应。
  - 局部自由度的编号规则，即基函数在每个单元上的形函数的编号。
  - 局部自由度和全局自由度的对应规则，这是单元矩阵组装与总矩阵需要知道的关键信息。

LagrangeFiniteELementSpace 类的命名和接口约定

mesh	网格对象
p	空间的次数
GD	几何空间的维数
TD	几何空间的拓扑维数
dof	函数空间的自由度管理对象, 用于管理函数空间的自由度
ftype	函数空间所用的浮点数类型
itype	函数空间所用的整数类型
integralalg	数值积分算法类
spacetype	函数空间的类型, 'C' 表示分片连续空间, 'D' 表示断空间

Table: LagrangeFiniteElementSapce 的常用数据成员 (属性)。

LagrangeFiniteELementSpace 类的命名和接口约定

ldof = space.number_of_local_dofs()	获得每个单元上的自由度个数
gdof = space.number_of_global_dofs()	获得所有单元上的总自由度个数
cell2dof = sapce.cell_to_dof()	获得单元与自由度的对应关系数组
bdof = space.boundary_dof()	获得边界的自由度
phi = space.basis(bc, ...)	计算单元上积分点处的基函数值
gphi = space.grad_basis(bc, ...)	计算单元上积分点处的基函数梯度值
val = space.value(uh, bc, ...)	计算有限元函数值
val = space.grad_value(uh, bc, ...)	计算有限元函数梯度值
ul = space.interpolation(u)	计算函数 u 在有限元空间中的插值函数
uh = space.function(...)	创建拉格朗日有限元空间的函数对象
A = space.stiff_matrix(...)	构造刚度矩阵
M = space.mass_matrix(...)	构造质量矩阵
F = space.source_vector(.. .)	构造源项向量

Table: LagrangeFiniteElementSapce 常用方法成员(属性), 其中省略号表示有默认参数。

LagrangeFiniteELementSpace 类的命名和接口约定

Remark

FEALPy 中的其它类型的有限维空间对象和拉格朗日有限元空间遵守几乎一样的命名和接口约定。

LagrangeFiniteELementSpace 调用演示

(1) 基函数及其导数的计算
(2) 自由度管理
(3) 有限元函数
(4) 插值
(5) 刚度矩阵的组装
(6) 载荷向量的组装
(7) 边界条件的处理

Outline