Discrete Geometries of Mathematics and Physics

# Mimetic spectral element method[1]

# Assignment #3

Numerical integration

Yi Zhang (张仪)
: www.mathischeap.com
@: zhangyi_aero@hotmail.com
git: https://github.com/mathischeap

You probably have used the `scipy.quad` module for integration. However, its efficiency is low and it is like a black box; usually no one carefully checks the source code of `scipy`. Thus, you'd better know how to do integration by yourself such that you can freely use it in your programs. You will see that it is very important in the coming assignments. And, in this assignment, lets find out how it works togerther.

## 1 Gauss quadrature in $[-1, 1]$

On a segment $[-1, 1]$, the most widely used numerical integration is the Gauss(-Legendre) quadrature. Given a positive integer $N_q$ (called the quadrature degree), one can compute the Gauss sample nodes $G_\mathrm{n}$ and Gauss weights $G_\mathrm{w}$,

$$G_\mathrm{n} = \left\{\lambda_1, \lambda_2, \cdots, \lambda_{N_q}\right\},$$

$$G_\mathrm{w} = \left\{w_1, w_2, \cdots, w_{N_q}\right\},$$

where $-1 < \lambda_1 < \lambda_2 < \cdots < \lambda_{N_q} < 1$ and $w_i \in \mathbb{R}$, $i \in \{1, 2, \cdots, N_q\}$. Note that these $\lambda_i$ are not a partition of $[-1, 1]$. You can use, for example,

`numpy.polynomial.legendre.leggauss`

function to compute these nodes and weights.

Given a function $f(\lambda)$ over $[-1, 1]$, we can compute its integration over $[-1, 1]$ by

$$\int_{-1}^{1} f(\lambda)\mathrm{d}\lambda \approx \sum_{i=1}^{N_q} w_i f(\lambda_i). \tag{1}$$

[1] https://mathischeap.com/contents/teaching/advanced_numerical_methods/mimetic_spectral_element_method/main

This is saying we just need to 1) compute the values of $f(\lambda)$ at the Gauss sample nodes, 2) multiply the values with the Gauss weights, and 3) compute the summation. If $f_p(\lambda)$ is a polynomial of degree $2N-1$ or less, this numerical integration will be exact, i.e.,

$$\int_{-1}^{1} f_p(\lambda)\mathrm{d}\lambda = \sum_{i=1}^{N_q} w_i f_p(\lambda_i).$$

We can see that the Gauss quadrature (1) is very handy; it computes integral using a computation of muliplication and summation which are very easy for computers. What is better, it can be exact for polynomials and our mimetic spectral element spaces are spaces of polynomials in this series of assignments.

## 2 Gauss quadrature in $\Omega_\mathrm{r} = [-1,1]^2$

In one dimension, the Gauss quadrature (1) is straight-forward. And it is simple to extend it to the reference element $\Omega_\mathrm{r} = [-1,1]^2$ in two dimensions.

In $\Omega_\mathrm{r}$, given a function $f(\xi,\eta)$, let $\{\xi_1,\xi_2,\cdots,\xi_N\}$ and $\{\eta_1,\eta_2,\cdots,\eta_N\}$ be the Gauss sample nodes along two axes. The Gauss quadrature is

$$\int_{\Omega_\mathrm{r}} f(\xi,\eta)\mathrm{d}\Omega = \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta)\mathrm{d}\eta\mathrm{d}\xi \approx \sum_{i=1}^{N_q}\sum_{j=1}^{N_q} w_i w_j f(\xi_i,\eta_j).$$

We see that it is simply like applying the one-dimensional Gauss quadrature to the two axes.

## 3 Gauss quadrature in an arbitrary element $\Omega_n \in \mathbb{R}^2$

Through the last assignment, we already know that the real problem usually is not defined in $\Omega_\mathrm{r}$. Thus we need to know how to do the numerical integration in an arbitrary element $\Omega_n$.

Suppose a $C^1$ diffeomorphism $\Phi_n$ maps $\Omega_\mathrm{r}$ into $\Omega_n$, i.e.,

$$\Phi_n : \Omega_\mathrm{r} \to \Omega_n.$$

This mapping maps the Gauss sample nodes in $\Omega_\mathrm{r}$ to the Gauss sample nodes in $\Omega_n$ as

$$\begin{bmatrix} x_i \\ y_j \end{bmatrix} = \Phi_n(\xi_i,\eta_j), \quad i,j \in \{1,2,\cdots,N_q\}.$$

Now, given a function $f(x,y)$ in $\Omega_n$, we can compute it integral by

$$\int_{\Omega_n} f(x,y)\mathrm{d}\Omega \approx \sum_{i=1}^{N_q}\sum_{j=1}^{N_q} w_i w_j f(x_i,y_j) J_n(x_i,y_i), \tag{2}$$

where $J_n$ is the Jacobian of mapping $\Phi_n$.

## 4 Gauss quadrature in an orthogonal rectangle

Samilar to Assignment #2, we only consider orthogonal rectangles here. So, we assume $\Omega_n$ to be an orthognal rectangle

$$\Omega_n := [x_0,x_1] \times [y_0,y_1].$$

And we use $d_x = x_1 - x_0$ and $d_y = y_1 - y_0$. From Assignment #2, we know that for such an orthogonal rectangle, its Jacobian is a constant, i.e.,

$$J_n = \frac{d_x d_y}{4}.$$

So, in (2), the contribution of $J_n(x_i, y_i)$ becomes no longer coordinates-related; (2) can be written as

$$\int_{\Omega_n} f(x,y)\mathrm{d}\Omega \approx \frac{d_x d_y}{4} \sum_{i=1}^{N_q} \sum_{j=1}^{N_q} w_i w_j f(x_i, y_j). \tag{3}$$

This makes the numerical integral more or less as simple as that in the reference element.

**Assignment 3.1.0: Numerical integral in an orthogonal rectangle**

You need to program a function to compute the numerical integral in an orthogonal rectangle. You can apply your function to, for example,

$$f(x,y) = \sin(2\pi x)\cos(2\pi y) + 1,$$

and compute its integral over a rectangle $[1,2] \times [2,3]$. Compare your results for different quadrature degree $N_q$ to the analytical result.

## 5 Numerical integration for mimetic spectral element spaces

Consider the mimetic spectral element space $C(\Omega_n)$. If $\alpha_h, \beta_h \in C(\Omega_n)$, we know they can be expressed as

$$\begin{aligned} \alpha_h &= \sum_{i=0}^{j=N} \sum_{j=0}^{N} \mathsf{a}_{ij} \mathtt{ll}_n^{ij}(x,y), \\ \beta_h &= \sum_{i=0}^{j=N} \sum_{j=0}^{N} \mathsf{b}_{ij} \mathtt{ll}_n^{ij}(x,y). \end{aligned} \tag{4}$$

Now, we try to use the numerical integral to compute the inner product between $\alpha_h$ and $\beta_h$, $\langle \alpha_h, \beta_h \rangle_{\Omega_n}$. Using (4), the inner product can be written explicitly as

$$\langle \alpha_h, \beta_h \rangle_{\Omega_n} = \left\langle \sum_{i=0}^{j=N} \sum_{j=0}^{N} \mathsf{a}_{ij} \mathtt{ll}_n^{ij}(x,y), \sum_{i=0}^{j=N} \sum_{j=0}^{N} \mathsf{b}_{ij} \mathtt{ll}_n^{ij}(x,y) \right\rangle_{\Omega_n}. \tag{5}$$

If we use a one-dimensional indexing (also called a local labeling) to label the expansion coefficients and basis functions as

$$\begin{aligned} &\mathsf{a}_k = \mathsf{a}_{j\times(N+1)+i+1} = \mathsf{a}_{ij}, \\ &\mathsf{b}_k = \mathsf{b}_{j\times(N+1)+i+1} = \mathsf{b}_{ij}, \\ &\mathtt{ll}^k(x,y) = \mathtt{ll}^{j\times(N+1)+i+1}(x,y) = \mathtt{ll}^{ij}(x,y). \end{aligned}$$

The inner product (5) can be further expressed as

$$
\begin{aligned}
\langle \alpha_h, \beta_h \rangle_{\Omega_n} &= \left\langle \sum_{i=0}^{j=N}\sum_{j=0}^{N} \mathsf{a}_{ij}\mathtt{ll}_n^{ij}(x,y), \sum_{i=0}^{j=N}\sum_{j=0}^{N} \mathsf{b}_{ij}\mathtt{ll}_n^{ij}(x,y) \right\rangle_{\Omega_n} . \\
&= \begin{bmatrix} \mathsf{a}_0 & \mathsf{a}_1 & \cdots & \mathsf{a}_K \end{bmatrix} \begin{bmatrix} \langle \mathtt{ll}^1, \mathtt{ll}^1 \rangle_{\Omega_n} & \langle \mathtt{ll}^1, \mathtt{ll}^2 \rangle_{\Omega_n} & \cdots & \langle \mathtt{ll}^1, \mathtt{ll}^K \rangle_{\Omega_n} \\ \langle \mathtt{ll}^2, \mathtt{ll}^1 \rangle_{\Omega_n} & \langle \mathtt{ll}^2, \mathtt{ll}^2 \rangle_{\Omega_n} & \cdots & \langle \mathtt{ll}^2, \mathtt{ll}^K \rangle_{\Omega_n} \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathtt{ll}^K, \mathtt{ll}^1 \rangle_{\Omega_n} & \langle \mathtt{ll}^K, \mathtt{ll}^1 \rangle_{\Omega_n} & \cdots & \langle \mathtt{ll}^K \mathtt{ll}^K \rangle_{\Omega_n} \end{bmatrix} \begin{bmatrix} \mathsf{b}_0 \\ \mathsf{b}_1 \\ \cdots \\ \mathsf{b}_K \end{bmatrix},
\end{aligned} \tag{6}
$$

where $K = (N+1)^2$ and we have omitted $(x, y)$ (for example, $\mathtt{ll}^1$ means $\mathtt{ll}^1(x, y)$). It is clear that the inner product becomes a matrix multiplication of three inputs, a $1 \times (N+1)^2$ matrix (row vector), an $(N+1)^2 \times (N+1)^2$ square matrix, an $(N+1)^2 \times 1$ matrix (colume vector). The output is a $1 \times 1$ matrix, i.e., a real number. With (6), we can see that, if we have pre-computed

$$
\mathbb{M}_C = \begin{bmatrix} \langle \mathtt{ll}^1, \mathtt{ll}^1 \rangle_{\Omega_n} & \langle \mathtt{ll}^1, \mathtt{ll}^2 \rangle_{\Omega_n} & \cdots & \langle \mathtt{ll}^1, \mathtt{ll}^K \rangle_{\Omega_n} \\ \langle \mathtt{ll}^2, \mathtt{ll}^1 \rangle_{\Omega_n} & \langle \mathtt{ll}^2, \mathtt{ll}^2 \rangle_{\Omega_n} & \cdots & \langle \mathtt{ll}^2, \mathtt{ll}^K \rangle_{\Omega_n} \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathtt{ll}^K, \mathtt{ll}^1 \rangle_{\Omega_n} & \langle \mathtt{ll}^K, \mathtt{ll}^2 \rangle_{\Omega_n} & \cdots & \langle \mathtt{ll}^K, \mathtt{ll}^K \rangle_{\Omega_n} \end{bmatrix},
$$

we can quickly compute the $L^2$-inner product as

$$
\langle \alpha_h, \beta_h \rangle_{\Omega_n} = \vec{\alpha}^{\mathsf{T}} \mathbb{M}_C \vec{\beta},
$$

where $\vec{\alpha}$ and $\vec{\beta}$ are colume vectors of expansion coefficients of $\alpha_h$ and $\beta_h$, respectively. And we call $\mathbb{M}_C$ the mass matrix of $C(\Omega_n)$. Obviously, $\mathbb{M}_G$, the mass matrix of $G(\Omega_n)$ is same to $\mathbb{M}_C$. Clearly $\mathbb{M}_C$ is symmetric, i.e., $\mathbb{M}_C^{\mathsf{T}} = \mathbb{M}_C$.

Similarly, we can easily derive the mass matrix of $S(\Omega_n)$,

$$
\mathbb{M}_S = \begin{bmatrix} \langle \mathsf{ee}^1, \mathsf{ee}^1 \rangle_{\Omega_n} & \langle \mathsf{ee}^1, \mathsf{ee}^2 \rangle_{\Omega_n} & \cdots & \langle \mathsf{ee}^1, \mathsf{ee}^L \rangle_{\Omega_n} \\ \langle \mathsf{ee}^2, \mathsf{ee}^1 \rangle_{\Omega_n} & \langle \mathsf{ee}^2, \mathsf{ee}^2 \rangle_{\Omega_n} & \cdots & \langle \mathsf{ee}^2, \mathsf{ee}^L \rangle_{\Omega_n} \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathsf{ee}^L, \mathsf{ee}^1 \rangle_{\Omega_n} & \langle \mathsf{ee}^L, \mathsf{ee}^2 \rangle_{\Omega_n} & \cdots & \langle \mathsf{ee}^L, \mathsf{ee}^L \rangle_{\Omega_n} \end{bmatrix},
$$

where we have used a local labeling $l = (j-1) \times N + i$, and $L = N^2$.

Now, we consider the mimetic spectral element space $\boldsymbol{D}(\Omega_n)$. If $\boldsymbol{u}_h, \boldsymbol{\rho}_h \in \boldsymbol{D}(\Omega_n)$, they can be expressed as

$$
\begin{aligned}
\boldsymbol{u}_h &= \begin{bmatrix} \sum_{i=0}^{j=N}\sum_{j=1}^{N} \mathsf{u}_{ij}\mathtt{le}_n^{ij}(x,y) \\ \sum_{i=1}^{j=N}\sum_{j=0}^{N} \mathsf{v}_{ij}\mathtt{el}_n^{ij}(x,y) \end{bmatrix}, \\
\boldsymbol{\rho}_h &= \begin{bmatrix} \sum_{i=0}^{j=N}\sum_{j=1}^{N} \mathsf{p}_{ij}\mathtt{le}_n^{ij}(x,y) \\ \sum_{i=1}^{j=N}\sum_{j=0}^{N} \mathsf{q}_{ij}\mathtt{el}_n^{ij}(x,y) \end{bmatrix}.
\end{aligned} \tag{7}
$$

The inner product between them can be written explicitly as

$$
\begin{aligned}
\langle \boldsymbol{u}_h, \boldsymbol{\rho}_h \rangle_{\Omega_n} &= \left\langle \begin{bmatrix} \sum_{i=0}^{j=N}\sum_{j=1}^{N} \mathsf{u}_{ij}\mathtt{le}_n^{ij}(x,y) \\ \sum_{i=1}^{j=N}\sum_{j=0}^{N} \mathsf{v}_{ij}\mathtt{el}_n^{ij}(x,y) \end{bmatrix}, \begin{bmatrix} \sum_{i=0}^{j=N}\sum_{j=1}^{N} \mathsf{p}_{ij}\mathtt{le}_n^{ij}(x,y) \\ \sum_{i=1}^{j=N}\sum_{j=0}^{N} \mathsf{q}_{ij}\mathtt{el}_n^{ij}(x,y) \end{bmatrix} \right\rangle_{\Omega_n} \\
&= \vec{\boldsymbol{u}}^{\mathsf{T}} \mathbb{M}_{\boldsymbol{D}} \vec{\boldsymbol{\rho}}.
\end{aligned} \tag{8}
$$

where

$$\vec{\boldsymbol{u}} = \begin{bmatrix} \mathsf{u}_1 \\ \mathsf{u}_2 \\ \vdots \\ \mathsf{u}_M \\ \mathsf{v}_1 \\ \mathsf{v}_2 \\ \vdots \\ \mathsf{v}_M \end{bmatrix}, \quad \vec{\boldsymbol{\rho}} = \begin{bmatrix} \mathsf{p}_1 \\ \mathsf{p}_2 \\ \vdots \\ \mathsf{p}_M \\ \mathsf{q}_1 \\ \mathsf{q}_2 \\ \vdots \\ \mathsf{q}_M \end{bmatrix},$$

and

$$\mathbb{M}_{\boldsymbol{D}} = \begin{bmatrix} \mathbb{M}^x_{\boldsymbol{D}} & \mathbf{0} \\ \mathbf{0} & \mathbb{M}^y_{\boldsymbol{D}} \end{bmatrix}$$

with

$$\mathbb{M}^x_{\boldsymbol{D}} = \begin{bmatrix} \langle \mathtt{le}^1, \mathtt{le}^1 \rangle_{\Omega_n} & \langle \mathtt{le}^1, \mathtt{le}^2 \rangle_{\Omega_n} & \cdots & \langle \mathtt{le}^1, \mathtt{le}^M \rangle_{\Omega_n} \\ \langle \mathtt{le}^2, \mathtt{le}^1 \rangle_{\Omega_n} & \langle \mathtt{le}^2, \mathtt{le}^2 \rangle_{\Omega_n} & \cdots & \langle \mathtt{le}^2, \mathtt{le}^M \rangle_{\Omega_n} \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathtt{le}^M, \mathtt{le}^1 \rangle_{\Omega_n} & \langle \mathtt{le}^M, \mathtt{ee}^2 \rangle_{\Omega_n} & \cdots & \langle \mathtt{le}^M, \mathtt{le}^M \rangle_{\Omega_n} \end{bmatrix},$$

$$\mathbb{M}^x_{\boldsymbol{D}} = \begin{bmatrix} \langle \mathtt{el}^1, \mathtt{el}^1 \rangle_{\Omega_n} & \langle \mathtt{el}^1, \mathtt{el}^2 \rangle_{\Omega_n} & \cdots & \langle \mathtt{el}^1, \mathtt{el}^M \rangle_{\Omega_n} \\ \langle \mathtt{el}^2, \mathtt{el}^1 \rangle_{\Omega_n} & \langle \mathtt{el}^2, \mathtt{el}^2 \rangle_{\Omega_n} & \cdots & \langle \mathtt{el}^2, \mathtt{el}^M \rangle_{\Omega_n} \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathtt{el}^M, \mathtt{el}^1 \rangle_{\Omega_n} & \langle \mathtt{el}^M, \mathtt{el}^2 \rangle_{\Omega_n} & \cdots & \langle \mathtt{el}^M, \mathtt{el}^M \rangle_{\Omega_n} \end{bmatrix}.$$

Note that we have locally labeled the first component (corresponding to basis functions $\mathtt{le}^{ij}$) as

$$m = (j-1) \times (N+1) + i + 1$$

and locally labeled the second component (corresponding to basis functions $\mathtt{el}^{ij}$) as

$$m = j \times N + i.$$

Thus, for both components, the total number of basis functions (and expansion coefficients) is $M = N \times (N+1)$.

Similarly, you should be able to derive the mass matrix of $\boldsymbol{R}(\Omega_n)$.

It is easy to see, just as $\mathbb{M}_C$, all mass matrices are symmetric.

## 6 Compute the mass matrices

As we have seen, each entry of a mass matrix is just a inner product between two basis functions. In other words, it is an integral. So, we can apply the numerical integration to compute it. For example,

$$\langle \mathtt{ll}^1, \mathtt{ll}^2 \rangle_{\Omega_n} = \int_{\Omega_n} \mathtt{ll}^1(x,y)\mathtt{ll}^2(x,y)\mathrm{d}\Omega \approx \frac{d_x d_y}{4} \sum_{i=1}^{N_q} \sum_{j=1}^{N_q} w_i w_j \mathtt{ll}^1(x_i, y_j)\mathtt{ll}^2(x_i, y_j).$$

Similarly, you can compute entris of all mass matrices. And remember that, since all mass matrices are symmetric, we only need to compute the upper (or lower) triangular parts of the mass matrices.

**Assignment 3.2.0: Compute mass matrices**

You need to program four functions to compute the four different mass matrices.