Integration using Gaussian Quadrature

Tutorials

December 15, 2019

Department of Aeronautics, Imperial College London, UK
Scientific Computing and Imaging Institute, University of Utah, USA

2

Chapter 1

Introduction

Welcome to the tutorial on the fundamentals of the *Nektar++* framework where we will look at how to perform 1D and 2D Gaussian Quadrature using the *Nektar++* LibUtilities library. If you have not already downloaded and installed *Nektar++*, please do so by visiting http://www.nektar.info, where you can also find the User-Guide with the instructions on how to install the library.

This tutorial requires:

- *Nektar++* compiled libraries and include files compiled from source so additional code can be compiled with the framework libraries

Goals

After completing this tutorial, you should be familiar with:

- The concept of Gaussian integration using classical Gauss and Gauss-Lobatto rules in a standard interval $\xi \in [-1, 1]$;
- Using the *Nektar++* programming concepts of an `Array`, `PointsKey` and the `PointsManager` to generate Gaussian quadrature weights;
- Integrating in the standard segment ($\xi \in [-1, 1]$) and quadrilateral region ($\xi \in [-1, 1] \times [-1, 1]$);
- The mathematical concept of mapping a general quadrilateral region to the standard region, evaluating the jacobian of this mapping and using this to evaluate an integral in a general straight sided quadrilateral region.

Task 1.1

Prepare for the tutorial. Make sure that you have:

- Installed and tested *Nektar++* v5.0.0 compiled from source. We will refer to the directory where you installed *Nektar++* as `$NEKDIST` for the remainder of the tutorial.

  The tutorial folder also contains:

  - CMakeList.txt
  - LocIntegration2D.cpp
  - StdIntegration1D.cpp
  - StdIntegration2D.cpp

- Make a directory of your chosing, for example `tutorial`, and download the tutorial files from http://doc.nektar.info/tutorials/5.0.0/fundamentals/integration/fundamentals-integration.tar.gz into this directory.

- Unpack the tutorial files by using

  ```
  tar -xzvf fundamentals-integration.tar.gz
  ```

  to produce a directory `fundamentals-integration` with subdirectories called `tutorial` and `complete`.

- Change to the

  `fundamentals-integration/tutorial`

  directory and configure the tutorial examples for compilation by typing the command

  ```
  cmake -DCMAKE_PREFIX_PATH=$NEKDIST/build .
  ```

  You should now see a file called `Makefile` in this directory.

- Change to the

  `$NEKDIST/tutorial/fundamentals-integration/complete`

  directory and configure the completed version of the tutorial examples for compilation by again typing the command

  ```
  cmake -DCMAKE_PREFIX_PATH=$NEKDIST/build
  ```

  You should now see a file called `Makefile` in this directory.

Chapter 2

Integration on a one-dimensional standard region

In our finite element formulation we typically require a technique to evaluate, within each elemental domain, integrals of the form

$$\int_{-1}^{1} u(\xi) d\xi, \tag{2.1}$$

where $u(\xi)$ may well be made up of products of polynomial bases. Since the form of $u(\xi)$ is problem specific, we need an automated way to evaluate such integrals. This suggests the use of numerical integration or *quadrature*. The fundamental building block is the approximation of the integral by a finite summation of the form

$$\int_{-1}^{1} u(\xi) d\xi \approx \sum_{i=0}^{q-1} w_i u(\xi_i),$$

where w_i are specified constants or *weights* and ξ_i represents an abscissa of q distinct points in the interval $-1 \leq \xi_i \leq 1$. Although there are many different types of numerical integration we shall restrict our attention to *Gaussian quadrature*.

2.1 Gaussian Quadrature

Gaussian quadrature is a particularly accurate method for treating integrals where the integrand, $u(\xi)$, is smooth. In this technique the integrand is represented as a Lagrange polynomial using the q points ξ_i, which are to be specified, that is,

$$u(\xi) = \sum_{i=0}^{q-1} u(\xi_i) h_i(\xi) + \varepsilon(u), \tag{2.2}$$

where $\varepsilon(u)$ is the approximation error. If we substitute equation (2.2) into (2.1) we obtain a representation of the integral as a summation:

$$\int_{-1}^{1} u(\xi)d\xi = \sum_{i=0}^{q-1} w_i u(\xi_i) + R(u), \tag{2.3}$$

where

$$w_i = \int_{-1}^{1} h_i(\xi)d\xi, \tag{2.4}$$

$$R(u) = \int_{-1}^{1} \varepsilon(u)d\xi. \tag{2.5}$$

Equation (2.4) defines the weights w_i in terms of the integral of the Lagrange polynomial but to perform this integration we need to know the location of the abscissae or zeros ξ_i. Since $u(\xi)$ is represented by a polynomial of order $(q-1)$ we would expect the relation above to be exact if $u(\xi)$ is a polynomial of order $(q-1)$ or less [that is, when $u(\xi) \in P_{q-1}([-1,1])$ then $R(u) = 0$]. This would be true if, for example, we choose the points so that they are equispaced in the interval. There is, however, a better choice of zeros which permits exact integration of polynomials of higher order than $(q-1)$. This remarkable fact was first recognised by Gauss and is at the heart of Gaussian quadrature.

We here consider only the result of the Gauss quadrature for integrals of the type shown in equation (2.3) known as Legendre integration. There are three different types of Gauss quadrature known as Gauss, Gauss-Radau, and Gauss-Lobatto, respectively. The difference between the three types of quadrature lies in the choice of the zeros. Gauss quadrature uses zeros which have points that are interior to the interval, $-1 < \xi_i < 1$ for $i = 0, \ldots, q-1$. In Gauss-Radau the zeros include one of the end-points of the interval, usually $\xi = -1$, and in Gauss-Lobatto the zeros include both end points of the interval, that is, $\xi = \pm 1$.

Introducing $\xi_{i,P}^{\alpha,\beta}$ to denote the P zeros of the P^{th} order Jacobi polynomial $P_P^{\alpha,\beta}$ such that

$$P_P^{\alpha,\beta}(\xi_{i,P}^{\alpha,\beta}) = 0, \qquad i = 0, 1, \ldots, P-1,$$

where

$$\xi_{0,P}^{\alpha,\beta} < \xi_{1,P}^{\alpha,\beta} < \cdots < \xi_{P-1,P}^{\alpha,\beta},$$

we can define zeros and weights which approximate the integral

$$\int_{-1}^{1} u(\xi)d\xi = \sum_{i=0}^{q-1} w_i u(\xi_i) + R(u),$$

as:

(1) *Gauss-Legendre*

$$\begin{aligned} \xi_i &= \xi_{i,q}^{0,0} \qquad i = 0, \ldots, q-1 \\ w_i^{0,0} &= \frac{2}{[1-(\xi_i)^2]} \left[\frac{d}{d\xi} \left(L_q(\xi) \right)|_{\xi=\xi_i} \right]^{-2} \qquad i = 0, \ldots, q-1 \\ R(u) &= 0 \qquad \text{if } u(\xi) \in P_{2q-1}([-1,1]) \end{aligned}$$

(2) *Gauss-Radau-Legendre*

$$\begin{aligned} \xi_i &= \begin{cases} -1 & i = 0 \\ \xi_{i-1,q-1}^{0,1} & i = 1, \ldots, q-1 \end{cases} \\ w_i^{0,0} &= \frac{(1-\xi_i)}{q^2[L_{q-1}(\xi_i)]^2} \qquad i = 0, \ldots, q-1 \\ R(u) &= 0 \qquad \text{if } u(\xi) \in P_{2q-2}([-1,1]) \end{aligned}$$

(3) *Gauss-Lobatto-Legendre*

$$\begin{aligned} \xi_i &= \begin{cases} -1 & i = 0 \\ \xi_{i-1,q-2}^{1,1} & i = 1, \ldots, q-2 \\ 1 & i = q-1 \end{cases} \\ w_i^{0,0} &= \frac{2}{q(q-1)[L_{q-1}(\xi_i)]^2} \qquad i = 0, \ldots, q-1 \\ R(u) &= 0 \qquad \text{if } u(\xi) \in P_{2q-3}([-1,1]) \end{aligned}$$

In all of the above quadrature formulae $L_q(\xi)$ is the Legendre polynomial ($L_q(\xi) = P_q^{0,0}(\xi)$). The zeros of the Jacobi polynomial $\xi_{i,m}^{\alpha,\beta}$ do not have an analytic form and commonly the zeros and weights are tabulated. Tabulation of data can lead to copying errors and therefore a better way to evaluate the zeros is by the use of a numerical algorithm (see the appendix in "Spectral/hp element methods for CFD").

Chapter 3

Computational Exercises

3.1 One dimensional integration in a standard segment

In this first exercise we will demonstrate how to integrate the function $f(\xi) = \xi^{12}$ on the standard segment $\xi \in [-1, 1]$ using Gaussian quadrature. The Gaussian quadrature weights and zeros are coded in the `LIbUtilities` library and for future reference this can be found under the directory `$NEKDIST/library/LibUtilities/Foundations/`. For the following exercises we will access the zero and points from the `PointsManager`. The `PointsManager` is a type of map (or manager) which requires a key defining known Gaussian quadrature types called `PointsKey`.

In the `$NEKDIST/tutorial/fundamentals-integration/tutorial` directory open the file named `StdIntegration1D.cpp`. Look over the comments supplied in the file which outline how to define the number of quadrature points to apply, the type of Gaussian quadrature and some arrays to hold the zeros, weights and solution. Finally a `PointsKey` is defined which is then used to obtain the zeros and weights in two arrays called `quadZeros` and `quadWeights`.

Task 3.1

Implement a short block of code where you see the comments `"Write your code here"` which evaluates the loop

$$\int_{-1}^{1} f(\xi) d\xi \simeq \sum_{i=0}^{i<Q_{max}} w_i f(z_i).$$

To compile your code type

```
make StdIntegation1D
```

in the tutorial directory. When your code compiles successfully[1] then type

```
./StdIntegration1D
```

You should now get some output similar to

```
=========================================================
|        INTEGRATION ON A 1D STANDARD REGION            |
=========================================================
Integrate the function f(xi) = xi^12 on the standard
segment xi=[-1,1] with Gaussian quadrature
         Q = 4: Error = 0.179594
```

Task 3.2

Evaluate the previous integral for a quadrature order of $Q = Q_{\max}$ where $Q_{\max} = 7$ is the number of quadrature points required for an exact evaluation of the integral (calculate this value analytically). Verify that the error is zero (up to numerical precision).

We can also use Gauss-Lobatto-Legendre type integration rather than Gauss-Legendre type in the previous exercises. To do this we replace

```
LibUtilities::PointsType quadPointsType =
                         LibUtilities::eGaussGaussLegendre;
```

with

```
LibUtilities::PointsType quadPointsType =
                         LibUtilities::eGaussLobattoLegendre;
```

Task 3.3

Evaluate the previous integral for a quadrature order of $Q = Q_{\max}$ where $Q_{\max} = 7$ and 8 to verify that to exactly integrate with Gauss-Lobatto type integration you require an additional quadrature point and weights.

[1] If you are unable to get your code to compile you can see a completed exercise in the `$NEKDIST/tutorial/fundamentals-integration/completed` directory. The tutorial code is contained within a `#if WITHSOLUTION` block

3.2 Two-dimensional integration in a standard and local region

3.2.1 Quadrilateral element in a standard region

A straightforward extension of the one-dimensional Gaussian rule is to the two-dimensional standard quadrilateral region and similarly to the three-dimensional hexahedral region. Integration over $Q^2 = \{-1 \leq \xi_1, \xi_2 \leq 1\}$ is mathematically defined as two one-dimensional integrals of the form

$$\int_{Q^2} u(\xi_1, \xi_2)\ d\xi_1\ d\xi_2 = \int_{-1}^{1} \left\{ \int_{-1}^{1} u(\xi_1, \xi_2)\Big|_{\xi_2}\ d\xi_1 \right\} d\xi_2.$$

So if we replace the right-hand-side integrals with our one-dimensional Gaussian integration rules we obtain

$$\int_{Q^2} u(\xi_1, \xi_2)\ d\xi_1\ d\xi_2 \simeq \sum_{i=0}^{q_1-1} w_i \left\{ \sum_{j=0}^{q_2-1} w_j\ u(\xi_{1i}, \xi_{2j}) \right\},$$

where q_1 and q_2 are the number of quadrature points in the ξ_1 and ξ_2 directions. This expression will be exact if $u(\xi_1, \xi_2)$ is a polynomial and q_1, q_2 are chosen appropriately. To numerically evaluate this expression the summation over 'i' must be performed q_1 times at every ξ_{2i} point, that is,

$$\begin{aligned} \int_{Q^2} u(\xi_1, \xi_2)\ d\xi_1\ d\xi_2 &\simeq \sum_{i=0}^{q_1-1} w_i\ f(\xi_{1i}), \\ f(\xi_{1i}) &= \sum_{j=0}^{q_2-1} w_j\ u(\xi_{1i}, \xi_{2j}). \end{aligned}$$

Task 3.4

Integrate the function $f(\xi_1, \xi_2) = \xi_1^{12}\, \xi_2^{14}$ on the standard quadrilateral element $Q \in [-1, 1] \times [-1, 1]$ using Gaussian quadrature.
Using a series of one-dimensional Gaussian quadrature rules as outlined above evaluate the integral by completing the first part of the code in the file `StdIntegration2D.cpp` in the directory `$NekDist/tutorial/fundamentals-integration/tutorial`.
The quadrature weights and zeros in each of the coordinate directions have already been setup and are initially set to $q_1 = 6, q_2 = 7$ using a Gauss-Lobatto-Legendre quadrature rule. Complete the code by writing a structure of loops which implement the two-dimensional Gaussian quadrature rule[a]. The expected output is given below). Also verify that the error is zero when $q_1 = 8, q_2 = 9$.
Recall that to compile the file you type

```
make StdIntegration2D
```

[a] If you need help there is a completed version in the completed directory

When executing the tutorial with the quadrature order $q_1 = 6, q_2 = 7$ you should get an output of the form:

```
============================================================
|        INTEGRATION ON 2 DIMENSIONAL ELEMENTS             |
============================================================

Integrate the function f(x1,x2) = (x1)^12*(x2)^14
on the standard quadrilateral element:
         q1 = 6, q2 = 7: Error = 0.00178972
```

3.2.2 General straight-sided quadrilateral element

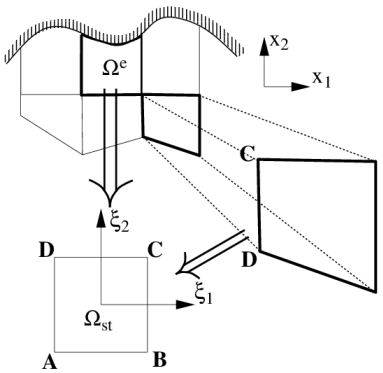

Figure 3.1 To construct a C^0 expansion from multiple elements of specified shapes (for example, triangles or rectangles), each elemental region Ω^e is mapped to a standard region Ω_{st} in which all local operations are evaluated.

For elemental shapes with straight sides a simple mapping may be constructed using a linear mapping similar to the vertex modes of a hierarchical/modal expansion. For the straight-sided quadrilateral with vertices labeled as shown in figure 3.1(b) the mapping can be defined as:

$$\begin{aligned} x_i &= \chi_i(\xi_1,\xi_2) = x_i^A \frac{(1-\xi_1)}{2}\frac{(1-\xi_2)}{2} + x_i^B \frac{(1+\xi_1)}{2}\frac{(1-\xi_2)}{2} \\ &\quad + x_i^D \frac{(1-\xi_1)}{2}\frac{(1+\xi_2)}{2} + x_i^C \frac{(1+\xi_1)}{2}\frac{(1+\xi_2)}{2}. \quad i = 1,2 \end{aligned} \tag{3.1}$$

If we denote an arbitrary quadrilateral region by Ω^e which is a function of the global Cartesian coordinate system (x_1, x_2) in two-dimensions. To integrate over Ω^e we transform this region into the standard region Ω_{st} defined in terms of (ξ_1, ξ_2) and we have

$$\int_{\Omega^e} u(x_1,x_2)\; dx_1\; dx_2 = \int_{\Omega_{st}} u(\xi_1,\xi_2)|J_{2D}|\; d\xi_1\; d\xi_2,$$

where J_{2D} is the two-dimensional Jacobian due to the transformation, defined as:

$$J_{2D} = \begin{vmatrix} \dfrac{\partial x_1}{\partial \xi_1} & \dfrac{\partial x_1}{\partial \xi_2} \\ \dfrac{\partial x_2}{\partial \xi_1} & \dfrac{\partial x_2}{\partial \xi_2} \end{vmatrix} = \frac{\partial x_1}{\partial \xi_1}\frac{\partial x_2}{\partial \xi_2} - \frac{\partial x_1}{\partial \xi_2}\frac{\partial x_2}{\partial \xi_1}. \tag{3.2}$$

As we have assumed that we know the form of the mapping [i.e., $x_1 = \chi_1(\xi_1, \xi_2)$, $x_2 = \chi_2(\xi_1, \xi_2)$] we can evaluate all the partial derivatives required to determine the Jacobian. If the elemental region is straight-sided then we have seen that a mapping from $(x_1, x_2) \to (\xi_1, \xi_2)$ is given by equations (3.1).

Task 3.5

We now consider how to integrate the function $f(x_1, x_2) = x_1^{12}\, x_2^{14}$ on a *local* rectangular quadrilateral element using Gaussian quadrature. Consider the local quadrilateral element with vertices

$$(x_1^A, x_2^A) = (0, -1), \qquad (x_1^B, x_2^B) = (1, -1),$$
$$(x_1^C, x_2^C) = (1, 1), \qquad (x_1^D, x_2^D) = (0, 0).$$

This is clearly similar to the previous exercise. However, as we are calculating the integral of a function defined on a local element rather than on a reference element, we have to take into account the geometry of the element. Therefore, the implementation is altered in two ways:

1. The quadrature zeros should be transformed to local coordinates to evaluate the integrand $f(x_1, x_2)$ at the quadrature points.
2. The Jacobian of the transformation between local and reference coordinates should be taken into account when evaluating the integral.

In the file `LocIntegration2D.cpp` you are provided with the same set up as the previous task but now with a definition of the coordinate mapping included. Evaluate the expression for the Jacobian analytically. Then write a line of code in the loop for the Jacobian as indicated by the comments `"Write your code here"`. When you have written your expression you can compile the code with the command

```
make LocIntegration2D
```

Verify that the error is not equal to zero when $q_1 = 8, q_2 = 9$. Why might this be the case?[a].

[a] Hint: What is the function in terms of ξ_1, ξ_2 and what is the polynomial degree of the Jacobian?

Using the quadrature order specified in the file your output should look like:

```
============================================================
|      INTEGRATION ON 2D ELEMENT in Local Region           |
============================================================

Integrate the function f(x1,x2) = x1^12 * x2^14
on a local quadrilateral element:
         Error = 0.000424657
```

Chapter 4

Summary

You should be now familiar with the following topics:

- Defining an `Array` and a `PointsKey` in *Nektar++*.
- Use the `PointsManager` with a `PointsKey` to get hold of quadrature weights and zeros.
- Integrate a polynomial function in the standard region $\xi \in [-1, 1]$ using Gauss-Gauss-Legendre and Gauss-Lobatto-Legendre quadrature.
- Extend the standard region to a standard quadrilateral region.
- Introduce a linear mapping from a general quadrilateral region to the standard quadrilateral region. Evaluate the Jacobian of this mapping and evaluate an integral in a general straight sided quadrilateral region.