Numerical Differentiation

Tutorials

November 9, 2025

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 Differentiation 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 differentiation using classical Gauss and Gauss-Lobatto rules in a standard interval $\xi \in [-1, 1]$;
- Using the *Nektar++* programming concepts of a `NekMatrix`, an `Array`, a `PointsKey` and the `PointsManager` to generate Gaussian quadrature zeros and differentiation matrices;
- Differentiating 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 a derivative in a general straight sided quadrilateral region.

Task 1.1

Prepare for the tutorial. Make sure that you have:

- Installed and tested *Nektar++* v5.9.0 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
  - LocDifferentiation2D.cpp
  - StdDifferentiation1D.cpp
  - StdDifferentiation2D.cpp

- Make a directory of your chosing, for example `tutorial`, and download the tutorial files from http://doc.nektar.info/tutorials/5.9.0/fundamentals/differentiation/fundamentals-differentiation.tar.gz into this directory.

- Unpack the tutorial files by using

  ```
  tar -xzvf fundamentals-differentiation.tar.gz
  ```

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

- Change to the

  `fundamentals-differentiation/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-differentiation/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

Differentiation

Assuming a polynomial approximation of the form:

$$u^{\delta}(x) = \sum_{p=0}^{P} \hat{u}_p \phi_p(\chi^{-1}) = \sum_{p=0}^{P} \hat{u}_p \phi_p(\xi),$$

where $\chi(\xi)$ is the mapping from the standard region $\xi \in \Omega^s$ to the region containing x in the interval $[a, b]$, we can differentiate $u(x)$ using the chain rule to obtain

$$\frac{du^{\delta}(\xi)}{dx} = \frac{du^{\delta}(\xi)}{d\xi}\frac{d\xi}{dx} = \sum_{p=0}^{P} \hat{u}_p \frac{d\phi_p(\xi)}{d\xi}\frac{d\xi}{dx}.$$

The differentiation of $u^{\delta}(x)$ is therefore dependent on evaluating $d\phi_p(\xi)/d\xi$ and $\frac{d\xi}{dx}$. In this section we shall consider the case where $\phi_p(\xi)$ is the Lagrange polynomial $h_p(\xi)$ and discuss how to evaluate $d\phi_p(\xi)/d\xi$. If $\chi(\xi)$ is an isoparametric mapping this technique can also be used to evaluate $\frac{d\chi}{d\xi} = \left[\frac{d\xi}{dx}\right]^{-1}$. Differentiation of this form is often referred to as differentiation in physical space or *collocation differentiation.*

If we assume that $u^{\delta}(\xi)$ is a polynomial of order equal to or less than P [that is, $u^{\delta}(\xi) \in P_P([-1, 1])$], then it can be exactly expressed in terms of Lagrange polynomials $h_i(\xi)$ through a set of q nodal points ξ_i $(0 \leq i \leq q-1)$ as

$$u(\xi) = \sum_{i=0}^{q-1} u(\xi_i) h_i(\xi), \qquad h_i(\xi) = \frac{\Pi_{j=0, j\neq i}^{q-1}(\xi - \xi_j)}{\Pi_{j=0, j\neq i}^{q-1}(\xi_i - \xi_j)}$$

where $q \geq P + 1$. Therefore the derivative of $u(\xi)$ can be represented as

$$\frac{du(\xi)}{d\xi} = \sum_{i=0}^{q-1} u(\xi_i) \frac{d}{d\xi} h_i(\xi).$$

Typically, we only require the derivative at the nodal points ξ_i which is given by

$$\left.\frac{du(\xi)}{d\xi}\right|_{\xi=\xi_i} = \sum_{j=0}^{q-1} d_{ij}\ u(\xi_j),$$

where

$$d_{ij} = \left.\frac{dh_j(\xi)}{d\xi}\right|_{\xi=\xi_i}.$$

An alternative representation of the Lagrange polynomial is

$$h_i(\xi) = \frac{g_q(\xi)}{g'_q(\xi_i)(\xi - \xi_i)}, \qquad g_q(\xi) = \prod_{j=0}^{q-1}(\xi - \xi_j).$$

Taking the derivative of $h_i(\xi)$ we obtain

$$\frac{dh_i(\xi)}{d\xi} = \frac{g'_q(\xi)(\xi - \xi_i) - g_q(\xi)}{g'_q(\xi_i)(\xi - \xi_i)^2}.$$

Finally, noting that because numerator and denominator of this expression are zero as $\xi \to \xi_i$, and because $P_q(\xi_i) = 0$ by definition,

$$\lim_{\xi\to\xi_i} \frac{dh_i(\xi)}{d\xi} = \lim_{\xi\to\xi_i} \frac{g''_q(\xi)}{2g'_q(\xi)} = \frac{g''_q(\xi_i)}{2g'_q(\xi_i)}$$

so we can write d_{ij} as

$$d_{ij} = \begin{cases} \dfrac{g'_q(\xi_i)}{g'_q(\xi_j)}\dfrac{1}{(\xi_i - \xi_j)} & i \neq j, \\ \\ \dfrac{g''_q(\xi_i)}{2g'_q(\xi_i)} & i = j. \end{cases} \tag{2.1}$$

Equation (2.1) is the general representation of the derivative of the Lagrange polynomials evaluated at the nodal points ξ_i $(0 \leq i \leq q-1)$. To proceed further we need to know specific information about the nodal points ξ_i which will allow us to deduce alternative forms of $g'_q(\xi_i)$ and $g''_q(\xi_i)$.

2.1 Legendre Formulae

The most common differentiation matrices d_{ij} are those corresponding to the Gauss-Legendre quadrature points. In this section we illustrate the final form of the differential matrices that correspond to the use of Gauss-Legendre, Gauss-Radau-Legendre, and Gauss-Lobatto-Legendre quadrature points. Denoting by $\xi^{\alpha,\beta}_{i,P}$ the P zeros of the Jacobi polynomial $P_P^{\alpha,\beta}(\xi)$ such that

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

the derivative matrix d_{ij} used to evaluate $\frac{du(\xi)}{d\xi}$ at ξ_i, that is,

$$\left.\frac{du(\xi)}{d\xi}\right|_{\xi=\xi_i} = \sum_{j=0}^{q-1} d_{ij}\, u(\xi_j),$$

is defined as:

(1) *Gauss-Legendre*

$$\xi_i = \xi_{i,q}^{0,0}$$

$$d_{ij} = \begin{cases} \dfrac{L_q'(\xi_i)}{L_q'(\xi_j)(\xi_i - \xi_j)} & i \neq j, 0 \leq i, j \leq q-1 \\ \\ \dfrac{\xi_i}{(1-\xi_i^2)} & i = j \end{cases}$$

(2) *Gauss-Radau-Legendre*

$$\xi_i = \begin{cases} -1 & i = 0 \\ \xi_{i-1,q-1}^{0,1} & i = 1, \ldots, q-1 \end{cases}$$

$$d_{ij} = \begin{cases} \dfrac{-(q-1)(q+1)}{4} & i = j = 0 \\ \\ \dfrac{L_{q-1}(\xi_i)}{L_{q-1}(\xi_j)} \dfrac{(1-\xi_j)}{(1-\xi_i)} \dfrac{1}{(\xi_i - \xi_j)} & i \neq j, 0 \leq i, j \leq q-1 \\ \\ \dfrac{1}{2(1-\xi_i)} & 1 \leq i = j \leq q-1 \end{cases}$$

(3) *Gauss-Lobatto-Legendre*

$$\xi_i = \begin{cases} -1 & i = 0 \\ \xi_{i-1,q-2}^{1,1} & i = 1, \ldots, q-2 \\ 1 & i = q-1 \end{cases}$$

$$d_{ij} = \begin{cases} \dfrac{-q(q-1)}{4} & i = j = 0 \\ \\ \dfrac{L_{q-1}(\xi_i)}{L_{q-1}(\xi_j)} \dfrac{1}{(\xi_i - \xi_j)} & i \neq j, 0 \leq i, j \leq q-1 \\ \\ 0 & 1 \leq i = j \leq q-2 \\ \\ \dfrac{q(q-1)}{4} & i = j = q-1 \end{cases}$$

In a similar way to the quadrature formulae the construction of the differentiation matrices require the quadrature zeros to be determined numerically. Having determined the zeros, the components of the differentiation matrix can be evaluated directly from the above formulae by generating the Legendre polynomial from the recursion relationship.

Chapter 3

Computational Exercises

3.1 One dimensional differentiation in a standard segment

In this first exercise we will demonstrate how to differentiate the function $f(\xi) = \xi^7$ in the standard segment $\xi \in [-1, 1]$ using Gaussian quadrature. The Gaussian quadrature zeros and the differentiation matrix 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 zeros and differentiation matrix 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-differentiation/tutorial` directory open the file named `StdDifferentiation1D.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, a differentiation matrix and some arrays to hold the zeros and solution. Finally a `PointsKey` is defined which is then used to obtain the zeros in an array called `quadZeros` and the differentiation matrix in a pointer to a matrix called `derivMatrix`.

Task 3.1

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

$$\left.\frac{du(\xi)}{d\xi}\right|_{\xi=\xi_i} = \sum_{j=0}^{q-1} d_{ij}\, u(\xi_j),$$

Tip

To access individual elements in `MatrixSharedPtrType`, the pointer must first be dereferenced then accessed using the parantheses operator. This should look like `(*derivMatrix)(i, j)`. On the other hand, to access elements in `Array<OneD, NekDouble>`, the bracket operator can be directly used such as `quadZerosDir1[i]`. If the array is two-dimensional (`Array<TwoD, NekDouble>`), elements would be accessed through `quadDerivsDir1[i][j]`.

To compile your code type

```
make StdDifferentiation1D
```

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

```
./StdDifferentiation1D
```

You should now get some output similar to

```
=======================================================
|      DIFFERENTIATION IN A 1D STANDARD REGION        |
=======================================================
Differentiate the function f(xi) = xi^7 in the
standard segment xi=[-1,1] using quadrature points
         Q = 7: Error = 1.49647
```

Task 3.2

Evaluate the previous derivatives for a quadrature order of $Q = Q_{\max}$ where $Q_{\max} = 8$ is the number of quadrature points required for an exact evaluation of the derivatives (calculate this value analytically). Verify that the error is zero (up to numerical precision).

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

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

with

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

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

3.2 Two-dimensional differentiation 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. Differentiation in $Q^2 = \{-1 \leq \xi_1, \xi_2 \leq 1\}$ in the ξ_1 direction is defined as

$$\frac{du(\xi_1, \xi_2)}{d\xi_1} = \sum_{i=0}^{q_1-1} \sum_{j=0}^{q_2-1} u(\xi_{1i}, \xi_{2j}) \frac{d}{d\xi_1}(h_{1i}(\xi_1) h_{2j}(\xi_2))$$

where h_1 and h_2 are the polynomials associated respectively with coordinates ξ_1 and ξ_2.

Because h_2 is not a function of ξ_1, we can rewrite the formula as

$$\frac{du(\xi_1, \xi_2)}{d\xi_1} = \sum_{i=0}^{q_1-1} \sum_{j=0}^{q_2-1} u(\xi_{1i}, \xi_{2j}) h_{2j}(\xi_2) \frac{d}{d\xi_1} h_{1i}(\xi_1). \tag{3.1}$$

Typically, we only require the derivative at the nodal points (ξ_{1i}, ξ_{2j}). At these points, h_2 has trivial values, i.e. unit value at ξ_{2j} and null value at all other points. The summation over j therefore drops and we are left with the same formula as in our one-dimensional problem

$$\left.\frac{du(\xi_1, \xi_2)}{d\xi_1}\right|_{\xi=(\xi_{1i}, \xi_{2j})} = \sum_{k=0}^{q_1-1} d_{1ik}\, u(\xi_{1k}, \xi_{2j}) \tag{3.2}$$

where d_1ik is the differentiation matrix in the ξ_1 direction such that

$$d_{1ik} = \left.\frac{dh_{1k}(\xi_1)}{d\xi_1}\right|_{\xi_1=\xi_{1i}}.$$

Likewise, the derivative with respect to ξ_2 can be expressed by

$$\left.\frac{du(\xi_1, \xi_2)}{d\xi_2}\right|_{\xi=(\xi_{1i}, \xi_{2j})} = \sum_{k=0}^{q_2-1} d_{2jk}\, u(\xi_{1i}, \xi_{2k}).$$

Task 3.3

Differentiate the function $f(\xi_1, \xi_2) = \xi_1^7 \, \xi_2^9$ in the standard quadrilateral element $Q \in [-1, 1] \times [-1, 1]$ using Gaussian quadrature points.
Using a series of one-dimensional Gaussian quadrature rules as outlined above evaluate the derivative in each direction and for each quadrature point by completing the first part of the code in the file `StdDifferentiation2D.cpp` in the directory
`$NekDist/tutorial/fundamentals-differentiation/tutorial`.
The quadrature zeros and differentiation matrices in each of the coordinate directions have already been setup and are initially set to $q_1 = 7, q_2 = 9$ 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 $q_1 = 8, q_2 = 10$.
Recall that to compile the file you type

```
make StdDifferentiation2D
```

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

When executing the tutorial with the quadrature order $q_1 = 7, q_2 = 9$ you should get an output of the form:

```
============================================================
|    DIFFERENTIATION IN 2D ELEMENT in Standard Region     |
============================================================

Differentiate the function f(x1,x2) = (x1)^7*(x2)^9
on the standard quadrilateral element:
         q1 = 7, q2 = 9: Error = 7.19196
```

3.2.2 General straight-sided quadrilateral element

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:

$$\begin{aligned} x_i \quad = \quad & \chi_i(\xi_1, \xi_2) = x_i^A \frac{(1-\xi_1)}{2}\frac{(1-\xi_2)}{2} + x_i^B \frac{(1+\xi_1)}{2}\frac{(1-\xi_2)}{2} \\ & + x_i^D \frac{(1-\xi_1)}{2}\frac{(1+\xi_2)}{2} + x_i^C \frac{(1+\xi_1)}{2}\frac{(1+\xi_2)}{2}. \quad i = 1, 2 \end{aligned} \tag{3.3}$$

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 differentiate in Ω^e we

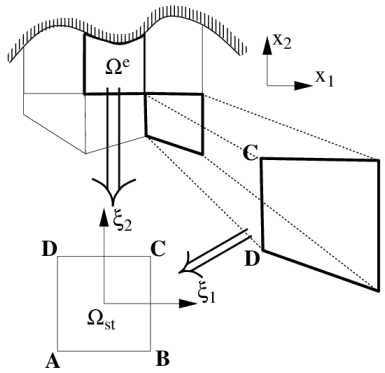

Figure 3.1 To construct a C^0 expansion from multiple elements of specified shapes (for example, triangles or rectangles), each elemental region Ω^e is mapped to a standard region Ω_{st} in which all local operations are evaluated.

transform this region into the standard region Ω_{st} defined in terms of (ξ_1, ξ_2). We begin with the basic definition of differentiation:

$$\frac{du(x_1, x_2)}{dx_1} = \sum_{i=0}^{q_1-1} \sum_{j=0}^{q_2-1} u(x_{1i}, x_{2j}) \frac{d}{dx_1}(h_{1i}(x_1, x_2) h_{2j}(x_1, x_2))$$

Unlike differentiation in the standard region, both h_1 and h_2 are functions of both local coordinates. The chain rule can be applied to obtain a system similar to the previous exercise:

$$\frac{du(x_1, x_2)}{dx_1} = \sum_{i=0}^{q_1-1} \sum_{j=0}^{q_2-1} u(x_{1i}, x_{2j}) [h_{2j}(\xi_2) \frac{d\xi_1}{dx_1} \frac{d}{d\xi_1} h_{1i}(\xi_1)) + h_{1i}(\xi_1) \frac{d\xi_2}{dx_1} \frac{d}{d\xi_2} h_{2j}(\xi_2))] \quad (3.4)$$

where h_1 and h_2 are functions of ξ_1 and ξ_2 respectively only and where $\frac{d\xi_2}{dx_1}$ and $\frac{d\xi_1}{dx_1}$ come from the inverse two-dimensional Jacobian matrix due to the transformation, defined as:

$$\mathbf{J}_{2D}^{-1} = \begin{bmatrix} \frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_2} \\ \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_2} \end{bmatrix}^{-1} = \begin{bmatrix} \frac{\partial \xi_1}{\partial x_1} & \frac{\partial \xi_1}{\partial x_2} \\ \frac{\partial \xi_2}{\partial x_1} & \frac{\partial \xi_2}{\partial x_2} \end{bmatrix}$$

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 matrix. 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 (3.3).

Equation (3.4) turns out to be similar to equation (3.1) in the standard region and can be also rewritten in the style of equation (3.2):

$$\begin{aligned}\left.\frac{du(x_1,x_2)}{dx_1}\right|_{x=(x_{1i},x_{2j})} &= \left.\frac{d\xi_1}{dx_1}\right|_{x=(x_{1i},x_{2j})} \sum_{k=0}^{q_1-1} d_{1ik}\, u(x_{1k},x_{2j}) \\ &+ \left.\frac{d\xi_2}{dx_1}\right|_{x=(x_{1i},x_{2j})} \sum_{k=0}^{q_2-1} d_{2jk}\, u(x_{1i},x_{2k}) \end{aligned} \tag{3.5}$$

Similarly to the differentiation in the standard region, derivatives with respect to the other local coordinate can also be found by substituting dx_2 for dx_1. We can also note that equation (3.2) is a particular case of equation (3.5) where $\frac{d\xi_1}{dx_1} = 1$ and $\frac{d\xi_2}{dx_1} = 0$. This effectively corresponds to the situation where the local region has the same shape and dimensions as the standard region (with the admission of a translation in the plane).

Task 3.4

We now consider how to differentiate the function $f(x_1, x_2) = x_1^7\, x_2^9$ in a *local* rectangular quadrilateral element. Consider the local quadrilateral element with vertices

$$\begin{aligned}(x_1^A, x_2^A) &= (0,-1), \qquad (x_1^B, x_2^B) = (1,-1), \\ (x_1^C, x_2^C) &= (1,1), \qquad (x_1^D, x_2^D) = (0,0).\end{aligned}$$

This is clearly similar to the previous exercise. However, as we are calculating the derivatives of a function defined in a local element rather than in 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 function $f(x_1, x_2)$ at the quadrature points.
2. Elements of the inverse Jacobian matrix of the transformation between local and reference coordinates should be taken into account when evaluating the derivatives.

In the file `LocDifferentiation2D.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 matrix analytically and find its inverse. 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 LocDifferentiation2D
```

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

```
============================================================
|     DIFFERENTIATION IN 2D ELEMENT in Local Region      |
============================================================

Differentiate the function f(x1,x2) = x1^7 * x2^9
in a local quadrilateral element:
         q1 = 8, q2 = 10: Average Error = 0.0346594
```

Advanced Task 3.5

As it turns out in the previous task, the average error is not equal to zero. Why is that?

Try different values of q_1 and q_2 and plot the average error with respect to these two parameters, either one by one or simultaneously. Why are the values of $q_{1\max}$ and $q_{2\max}$ (at which the average error reaches computer precision) different from those expected in the standard region?[a]

[a] Hint: How can this be explained by the geometry of the local element?

Chapter 4

Summary

You should be now familiar with the following topics:

- Define an `Array`, a `NekMatrix` and a `PointsKey` in *Nektar++*.
- Use the `PointsManager` with a `PointsKey` to get hold of quadrature zeros and differentiation matrices.
- Differentiate 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 derivatives in a general straight sided quadrilateral region.