Tutorial exercises

“A hands-on approach to implementing and using spectral/hp elements”

Spencer Sherwin, Mike Kirby, Peter Vos

Tutorial 1

1 Integration on a one-dimensional standard region

Assignment (a): Integrate the function $f(\xi) = \xi^{12}$ on the standard segment $\xi = [-1, 1]$ using Gaussian quadrature.
As explained in section 1.4.1 of the Course Notes , the Gaussian quadrature technique requires the quadrature weights w_i and zeros ξ_i in the standard interval $\xi = [-1, 1]$. Both the weights and zeros have already been calculated using the `PointsManager` of the Nektar++ library. Implement the Gaussian quadrature rule by completing the code in the following way:

1. Write a loop to numerically evaluate the integral $\int_{-1}^{1} f(\xi)d(\xi)$ using 4^{th} order Gauss-Lobatto-Legendre quadrature.
2. Now evaluate the integral for a quadrature order of $Q = Q_{max}$ where Q_{max} 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).

Assignment (b): Calculate the integral $\int_{-1}^{1} cos(\xi)d(\xi)$ using Gaussian quadrature for $2 \leq Q \leq 8$.
Based on the implementation of the previous exercise, write your own piece of code which evaluates the integral $\int_{-1}^{1} cos(\xi)d(\xi)$ and displays the error ε for the range $2 \leq Q \leq 8$. As we are essentially approximating the smooth function $cos(\xi)$ with polynomials of order $Q-1$ when using Gaussian quadrature, the error ε should be proportional to $\varepsilon \propto C^Q$. Therefore, the error should drop one order of magnitude with increasing Q. Verify that this is the case.

2 Differentiation on a one-dimensional standard region

Assignment (a): Numerically evaluate the derivative of the function $f(\xi) = \xi^7$ on the standard segment $\xi = [-1, 1]$.
As outlined in Section 1.4.2 of the Course Notes , numerical differentiation implies the use of a differentiation matrix. Similar as for the quadrature points, the differentiation matrix can be obtained through the `PointsManager` of the Nektar++ library. This has already been done for you. Write your own piece of code which implements the numerical differentiation rule to obtain the derivative of the function $f(\xi) = \xi^7$ at the quadrature points using $Q = 7, 8, 9$. Verify how many quadrature points are required to get an exact answer to numerical precision.

3 Interpolation on a one-dimensional standard region

Assignment (a): Interpolate the function $f(\xi) = \xi^{12}$ evaluated at the $Q = 13$ Gauss-Lobatto-Legendre zeros to $Q = 10$ equidistant points in the standard interval $\xi = [-1, 1]$.

In Section 1.3.4.1, it is explained how Lagrange polynomials can be employed to interpolate between different set of quadrature points. The interpolation is typically evaluated using the interpolation matrix I:

$$I[i][p] = h_p(\xi_i)$$

where $h_p(\xi_i)$ is the p^{th} Lagrange polynomial associated to the initial set of quadrature points evaluated at the i^{th} interpolation point. Similar as for the differentiation matrix, the interpolation matrix I can be obtained through the `PointsManager` of the Nektar++ library. This has already been done for you. Write your own piece of code which interpolates the function $f(\xi) = \xi^{12}$ from the Gauss-Lobatto-Legendre set of points onto the set of equidistant points. Verify that the interpolation is exact up to machine accuracy for the given parameters.

Now that you have obtained the values of the function $f(\xi) = \xi^{12}$ at $Q = 10$ equidistant points, calculate the quadrature weights associated with these points (using the `PointsManager`) and numerically evaluate the integral $\int_{-1}^{1} f(\xi) d(\xi)$. Verify that for $Q = 10$ equidistant points, the integral is not approximated exactly. Look back to Exercise 1(a) to check how many Gauss-Lobatto-Legendre points were required to exactly integrate the function $f(\xi)$.

Tutorial 2

1 Integration on two-dimensional elements

Assignment (a): Integrate the function $f(\xi_1, \xi_2) = \xi_1^{12}\xi_2^{14}$ on the standard quadrilateral element using Gaussian quadrature.
One dimensional Gaussian quadrature can be trivially extended to the two-dimensional standard quadrilateral region as outlined in section 3.1.1 of the Course Notes . The quadrature weights and zeros in each of the coordinate directions have already been calculated. Complete the code by writing a structure of loops which implement the two-dimensional Gaussian quadrature rule. Verify that the error ε is zero when $Q_1 = 8, Q_2 = 9$

Assignment (b): Integrate the function $f(x_1, x_2) = x_1^{12}x_2^{14}$ on a local quadrilateral element using Gaussian quadrature.
Consider the local quadrilateral element with vertices $(x_1^A, x_2^A) = (0, -1)$, $(x_1^B, x_2^B) = (1, -1)$, $(x_1^C, x_2^C) = (1, 1)$ and $(x_1^D, x_2^D) = (0, 0)$.
Your code can be based on the previous exercise. However, as we are calculating the integral of a function defined on a local element rather than on a reference element (see section 3.1.3.3 of the Notes), we have to take into account the geometry of the element. Therefore, the implementation should be 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. (Evaluate the expression for the Jacobian analytically rather than using numerical differentiation)

Verify that the error ε is not equal to zero when $Q_1 = 8, Q_2 = 9$. Try to reason why this is.

2 Generation of the mass matrix

Assignment (a): Generate the mass matrix for the 7^{th} order orthogonal spectral/hp basis on a local standard element using 9^{th} order Gaussian quadrature
Consider the local quadrilateral element with vertices $(x_1^A, x_2^A) = (0, -1)$, $(x_1^B, x_2^B) = (1, -1)$, $(x_1^C, x_2^C) = (1, 1)$ and $(x_1^D, x_2^D) = (0, 1)$. (Note that the last vertex has different coordinates as in the previous exercise)
As the elements of the mass matrix,

$$\boldsymbol{M}[n][m] = \int_{\Omega} \phi_n(x_1, x_2)\phi_m(x_1, x_2) dx_1 dx_2$$

consist of an integral over the local quadrilateral region, your code can be based on the previous exercise. In addition, we now also require information about the spectral/hp basis on the element. The orthogonal basis, defined in Section 2.1.1, can be formulated as:

$$\phi_n(x_1, x_2) = \phi_{pq}(x_1, x_2) = \tilde{\psi}_p^a(x_1)\tilde{\psi}_q^b(x_2)$$

As we only require the basis at the discrete quadrature points, the first step is to set up two arrays $\text{base1}[m(p, i)], \text{base2}[n(q, j)]$ for $0 \leq p, q \leq P = 7, 0 \leq i, j < Q = 9$ such that

$$\text{base1}[k(p, i)] = \tilde{\psi}_p^a(\xi_{1i})$$
$$\text{base2}[l(q, j)] = \tilde{\psi}_q^b(\xi_{2j})$$

These arrays can be obtained through the `BasisManager` of the Nektar++ library. This has already been implemented. These arrays correspond to the matrix $\boldsymbol{B}$ (see Section 3.1.5.1 of the Course Notes) of the one-dimensional bases stored in column major format. This implies that the single indices k and l can be related to the pair of one-dimensional indices (p,i) and (q,j) through the equations:

$$k(p,i) = p \times Q + i,$$
$$l(q,j) = q \times Q + j.$$

The elements of the mass matrix can than be calculated by numerically evaluating:

$$\boldsymbol{M}[n(p,q)][m(r,s)] = \int_{-1}^{1}\int_{-1}^{1} \tilde{\psi}_p^a(\xi_1)\tilde{\psi}_q^b(\xi_2)\tilde{\psi}_r^a(\xi_1)\tilde{\psi}_s^b(\xi_2)|J_{2D}|d\xi_1 d\xi_2$$

In the above expression $n(p,q)$ and $m(r,s)$ denote a mapping from the pair of one-dimensional indices (p,q) and (r,s) in a single unique numbering which represents the location of each two-dimensional mode in the matrix $\boldsymbol{M}$. There are many different choices of numbering systems and one of the most straightforward numbering scheme can be constructed as follows:

$$n(p,q) = p \times (P+1) + q,$$
$$m(r,s) = r \times (P+1) + s.$$

Complete the code by implementing a structure of loops which calculates every entry of the mass matrix. Plot the structure of the matrix and verify that the mass matrix for the orthogonal basis is diagonal in this case. (Note that this is only true for local elements when all angles are 90°, in which case the Jacobian J_{2D} is constant.)

3 Elemental projection problem

Assignment (a): Project the function $f(x_1,x_2) = x_1^6 x_2^6$ defined on a local quadrilateral element onto the spectral/hp element expansion defined on this element
Consider the local quadrilateral element with vertices $(x_1^A, x_2^A) = (0,-1)$, $(x_1^B, x_2^B) = (1,-1)$, $(x_1^C, x_2^C) = (1,1)$ and $(x_1^D, x_2^D) = (0,0)$ and use:

- a 6th order C^0 continuous modified expansion $\phi_{pq}(x_1,x_2) = \psi_p^a(x_1)\psi_q^b(x_2)$
- 8th order Gauss-Lobatto-Legendre quadrature in both directions.

Having developed routines for integration in a general elemental region we continue our spectral/*hp* element construction by considering an elemental projection problem in two-dimensions. Consider the projection problem $u^\delta(x_1,x_2) = f(x_1,x_2)$ where $f(x_1,x_2) = x_1^6 x_2^6$. Projection problems are helpful since they do not require any boundary conditions to be imposed. Extending the formulation in section 1.2 of the Course Notes our Galerkin problem in the elemental region Ω^e can be stated as:
Find $u^\delta \in \mathcal{X}^\delta$, such that

$$\int_{\Omega^e} v^\delta(x_1,x_2)u^\delta(x_1,x_2)dx_1x_2 = \int_{\Omega^e} v^\delta(x_1,x_2)f(x_1,x_2)dx_1x_2 \qquad \forall\ v^\delta \in \mathcal{V}^\delta,$$

and for a Galerkin expansion we define the expansion space $mathcalX^\delta$ to be the same as the test space $\mathcal{V}^\delta$.
Using the discrete expansion $\phi_{pq}(x_1,x_2) = \psi_p^a(x_1)\psi_q^b(x_2)$ and accordingly a discrete solution

$u^\delta(x_1, x_2) = \sum_p \sum_q \hat{u}_{pq}\phi_{pq}(x_1, x_2)$ leads to the matrix problem (see also Section 3.1.5.3 of the Course Notes)

$$\boldsymbol{M}\hat{\boldsymbol{u}} = \hat{\boldsymbol{f}}$$

where the above matrices and vectors were defined in Section 3.1.5.1 of the Course Notes .
Your implementation should consist of the following four parts:

1. Construct the mass matrix $\boldsymbol{M}$. Base your implementation on the previous exercise.

2. Construct the right hand side vector $\hat{\boldsymbol{f}}$:

$$\hat{\boldsymbol{f}}[m(p,q)] = \int_{-1}^{1}\int_{-1}^{1} \psi_p^a(\xi_1)\psi_q^b(\xi_2) f(\xi_1, \xi_2)|J_{2D}| d\xi_1 d\xi_2$$

3. Solve the resulting linear system for the expansion coefficients $\hat{\boldsymbol{u}}$. This has been implemented for you using the corresponding Nektar++ routines.

4. Perform the backward transformation

$$u^\delta(x_1, x_2) = \sum_p \sum_q \hat{u}_{pq}\psi_p^a(\xi_1)\psi_q^b(\xi_2)$$

Verify that the solution indeed approximates the function $f(x_1, x_2) = x_1^6 x_2^6$.

Tutorial 3

1 A two-dimensional Helmholtz problem

Assignment (a): Solve the Helmholtz equation with forcing function $f(x_1, x_2) = -(\lambda + 2\pi^2)\cos(\pi x_1)\cos(\pi x_2)$ *on the mesh defined in the input file QuadMesh.xml using Nektar++ .*
Information about the mesh and the boundary conditions are contained in the input file QuadMesh.xml. (Make sure to copy this file to the directory where your executable is located.) The following configuration of the problem is stored in this file:

- Domain of the problem: the bi-unit square $x_1 = [0, 1]$, $x_2 = [0, 1]$
- The mesh: 4 identical quadrilateral (square) elements
- The expansion: 4^{th} order C^0 continuous modified spectral/hp expansion
- Forcing function: $f(x_1, x_2) = -(\lambda + 2\pi^2)\cos(\pi x_1)\cos(\pi x_2)$
- Boundary Conditions: Dirichlet boundary conditions $g(x_1, x_2) = \cos(\pi x_1)\cos(\pi x_2)$ on the entire domain boundary
- Exact solution: $u(x_1, x_2) = \cos(\pi x_1)\cos(\pi x_2)$

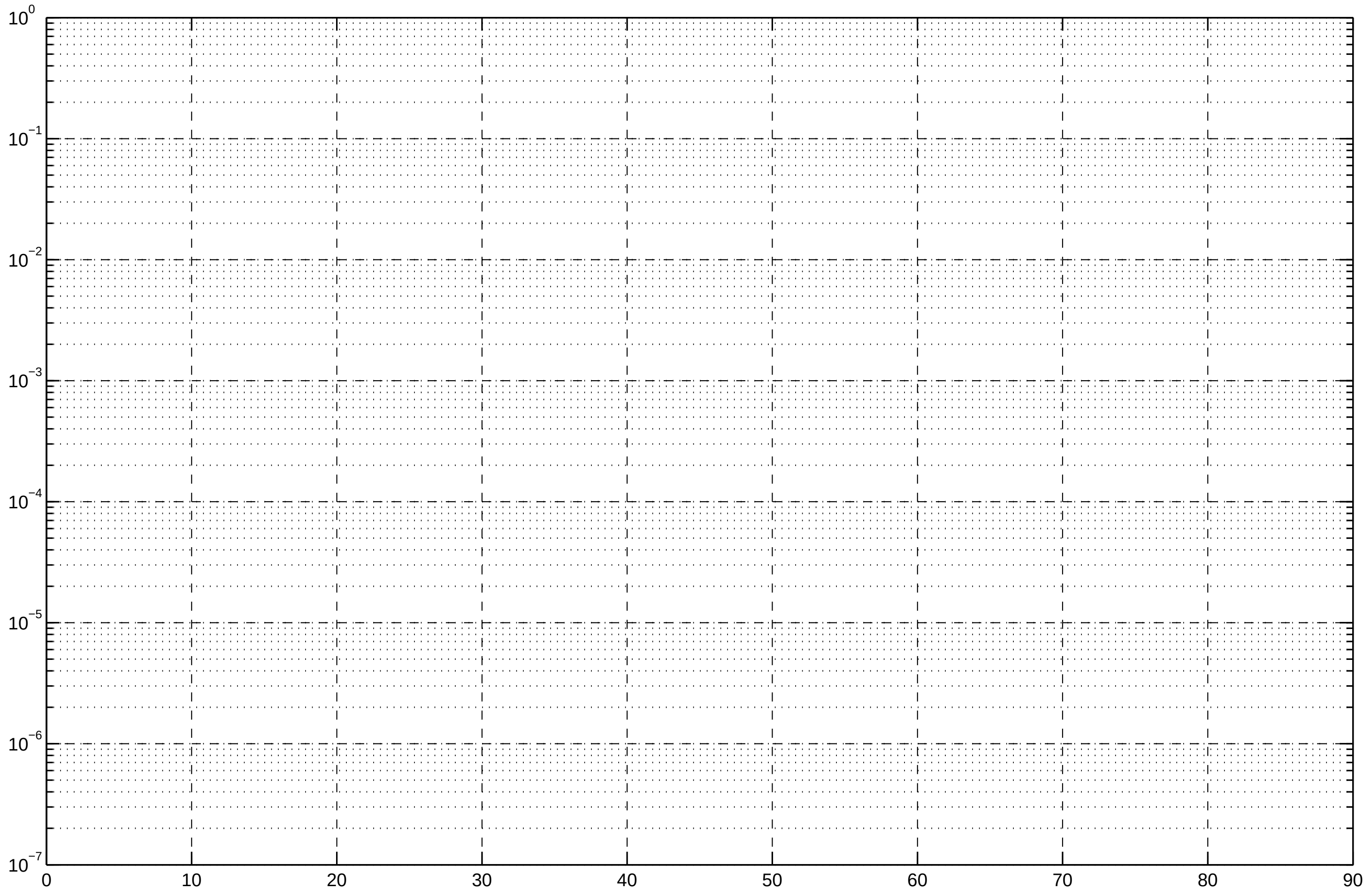
This exercise contains of two parts:

1. Complete the Helmholtz solver.
   Solve the exercise using the proper Nektar++ classes and functions. The major part is already implemented. However, some essential Nektar++ calls have been left out. The task is to, using the *doxygen* documentation of the Nektar++ code, find out how the code should be properly completed. The doxygen documentation can be found on: `http://www.nektar.info`. Once this is finished, the solution should have been written to the file Helmholtz2DSolution.pos. Open this file using the program *Gmsh* to see a plot of the solution.

2. Convergence study.
   Now you have a working Helmholtz solver, run it for the following configurations:

1 element , $P = 2$ (`NUMMODES`$= P + 1 = 3$)	
$h-$refinement	$P-$refinement
4 elements, $P = 2$	1 element , $P = 4$
9 elements, $P = 2$	1 element , $P = 6$
16 elements, $P = 2$	1 element , $P = 8$

   To do so, you will have to modify (or create a new) input file for every case. Plot the approximation error (calculated in the L_2 norm) for every case in function of the number of degrees of freedom in a semi-log plot. This can be done in the figure on the next page. If your implementation is correct, you should observe two different convergence rates, depending on the strategy (i.e. $h-$refinement versus $P-$refinement)


10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
0
10
20
30
40
50
60
70
80
90

2 Differentiation on a two-dimensional local region

Assignment (a): Calculate the gradient ∇f of the function $f(x_1, x_2) = x_1^7 x_2^7$ on a local quadrilateral element

As outlined in Section 3.1.3.4 of the Course Notes , the derivative of a function f within a local region can be evaluated as:

$$\nabla = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \\ \dfrac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \xi_1}{\partial x_1}\dfrac{\partial f}{\partial \xi_1} & + & \dfrac{\partial \xi_2}{\partial x_1}\dfrac{\partial f}{\partial \xi_2} \\ \\ \dfrac{\partial \xi_1}{\partial x_2}\dfrac{\partial f}{\partial \xi_1} & + & \dfrac{\partial \xi_2}{\partial x_2}\dfrac{\partial f}{\partial \xi_2} \end{bmatrix}.$$

Numerically evaluate ∇f at all quadrature points using $Q = 8$ quadrature points in both directions. The implementation consists of three different steps. Complete the code and:

1. Calculate the derivatives of the function f with respect to the reference coordinates ξ_1 and ξ_2. (see Section 3.1.2.1 of the Course Notes),
2. Calculate the metric terms $\frac{\partial \xi_i}{\partial x_j}$ of the transformation between reference and local coordinates. (Analytically evaluate the expression for these metric terms, see Section 3.1.3.4 of the Course Notes),
3. Apply the chain rule to calculate the gradient of f with respect to the local coordinates x_1 and x_2.

Display the error ε for every quadrature points and verify that the numerical solution does approximate the exact solution. However, although $Q = 8$ quadrature points are used the solution is not exact. Try to reason why this is.