4
Multi-dimensional Formulation

In this chapter we consider the implementation and formulation details used in constructing multi-dimension spectral/*hp* element approximations. This chapter directly builds upon the expansion bases within *standard* regions defined in chapter 3. Although more involved, the construction of a multiple dimensions is very similar to the one-dimensional approach discussed in chapter 2. By analogy to the one-dimensional formulation, the operations of integration and differentiation of a known function can be performed at an elemental level and may therefore be considered *local* or elemental operations. We therefore start our multi-dimensional discussion in section 4.1 by considering local operations. To extend these techniques to a C^0 multi-dimensional basis, as typically adopted in a Galerkin construction, we then require *global* operations such as matrix numbering, connectivity and assembly. These topics are introduced in 4.2. In section 4.3 we conclude our formulation by discussing more specialised topics relating to the pre- and post-processing aspects of a general multi-dimensional solver such as boundary condition representation, curvilinear mesh generation and consistent particle tracking in high-order elements. Finally, in section 4.4, we suggest a series of programming exercises to help construct a multi-dimensional Galerkin approximation.

Although integration and differentiation are important local operations the elemental mapping which allows us to generalise the local operations in a standard region to elements of general shape is equally important. The structure of section 4.1 therefore initially introduces integration and differentiation in the standard regions in sections 4.1.1 and 4.1.2. Subsequently in section 4.1.1, we introduce the concept of an elemental mapping which then allows us to discuss how to perform integration and differentiation in general elemental regions. Having defined these concepts, we are then able to discuss elemental transformation in section 4.1.5 for representing general functions over an elemental region using either collocation or Galerkin type projections. Within this section we also introduce a matrix notation which is helpful in illustrating the necessary operations to numerically perform many of the local operations. The matrix construction is convenient to clarify many of the numerical operations, however, when using tensorial based operations, it is computationally more efficient to use the sum-factorisation technique as detailed in section 4.1.6. Since all the techniques discussed in section 4.1 only apply on a single elemental region, they may equally well be used on the modified or orthogonal bases discussed in chapter 3. Many of the local operations are also relevant to the non-tensorial expansion bases for simplexes as defined in chapter 3. When using the non-tensorial basis we can-

not use the sum-factorisation technique and so matrix operators can be applied. Finally, we note that most of the local operations are equally valid for the standard Galerkin formulation introduced in chapter 2 or the discontinuous Galerkin formulation which is introduced in chapters 6 and 7.

For the standard Galerkin formulation we have seen in chapter 2 that normally a global C^0 continuous expansion is required. This was also the motivation behind the derivation of modified expansion bases defined in chapter 3. To construct a multiple domain C^0 expansion from the elemental contributions in a computationally efficient manner requires the introduction of suitable global operation. We, therefore, start our discussion on global operations in section 4.2.1 by introducing the concept of global assembly. Using the previously defined vector notation we illustrate, from a matrix operation point of view, how the elemental degrees of freedom are assembled (or combined) to make a global C^0 system. Implicit in this discussion is how local degrees of freedom have to be numbered and orientated. The global assembly operation can be used for an explicit and implicit implementation of the problem of interest. However, implicit implementation typically leads to global matrix systems the construction of which is discussed in section 4.2.2 using the elemental matrix notation introduced in section 4.1.5. Finally, having constructed a global matrix system in section 4.2.3 we discuss the *static condensation* technique which makes use of natural boundary/interior decomposition of the spectral/*hp* element expansion to decouple the interior modes from the boundary degrees of freedom. We then complete our discussion of global operations in section 4.2.4 by presenting how to set up a global numbering scheme and then using it to impose Dirichlet boundary conditions with examples in two-dimensions.

4.1 Local Elemental Operations

As a motivation for the topics to be introduced in this section we recall that to solve the Galerkin formulation of the Laplace equation we need to evaluate within every elemental region of our mesh the inner products of the form

$$(\nabla\phi_i \cdot \nabla\phi_j) = \int_{\Omega^e} \nabla\phi_i \cdot \nabla\phi_j \; d\boldsymbol{x} = \int_{\Omega_{st}} \nabla\phi_i \cdot \nabla\phi_j J \; d\boldsymbol{\xi},$$

where Ω^e denotes the element region, Ω_{st} denotes the standard elemental region such that $\xi \in \Omega_{st}$, and J is the Jacobian of the mapping between these two regions. From the structure of this inner product we note that there are three important concepts we need to understand how to implement. Firstly, we need to know how to integrate within Ω_{st}, then we need to know how to differentiate, initially in the standard region Ω_{st} and then in the elemental region Ω^e. To perform the differentiation (and integration) within the elemental region we need to define a mapping between these regions which is the third concept we will discuss in this section.

In section 2.4.1 we discussed an accurate form of one-dimensional numerical integration known as Gaussian integration. This is generally preferred in spectral/*hp* element methods as polynomial integrands of order $2P$ can be ex-

actly integrated using a summation over $O(P)$ points. In section 4.1.1 we will review how the one-dimensional Gaussian integration can be applied to the standard regions, Ω_{st}, for multi-dimensional expansions.

Similarly, in section 2.4.2 we illustrated how the differentiation of polynomial functions in physical space using the derivatives of a Lagrange polynomial through the Gaussian quadrature zeros. In section 4.1.2, we extend this concept to the standard multi-dimensional regions, Ω_{st}.

Having defined integration and differentiation in the standard regions in section 4.1.3, we extend these ideas to an arbitrary region by introducing appropriate elemental mappings. In section 4.1.5, we illustrate how to represent a mu1ti-dimensional function within a general elemental region, thereby defining the idea of a forward and backward transformation. To this end, we introduce a matrix notation. Finally, in section 4.1.6 we discuss the use of a numerically efficient technique, known as sum factorisation, to evaluate the salient numerical operations required in a Galerkin *hp*/spectral element formulation based on tensor product expansion.

4.1.1 Integration within the Standard Region Ω_{st}

The one-dimensional integral of a smooth function may be approximated using Gaussian quadrature as a summation of the form (see also section 2.4.1.1 and appendix B)

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

where ξ_i is the Q discrete quadrature points or *zeros* at which the function $u(\xi)$ is evaluated and w_i is the set of coefficients of *weights*. The term $R(u)$ denotes the approximation error which, providing a sufficient number of quadrature points are used, will be zero if the integrand is a polynomial. For example, if $u(\xi)$ represents the local expansion basis $u(\xi) = \phi_p(\xi)$ then providing there are sufficient quadrature points there will be no approximation error. We recall that the classical Gauss-Legendre quadrature does not include any zeros at the ends of the interval. If both end-points are included the integration is referred to as *Gauss-Lobatto* and if only one end-point is included the integration is referred to as *Gauss-Radau*. A more general form of quadrature involving a weight function in the integrand is referred to as Gauss-Jacobi (see appendix B).

4.1.1.1 *Quadrilateral and Hexahedral Regions*

Implementation note: *Numerical integration in Ω_{st}: quadrilateral and hexahedral regions.*

A trivial 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 $\mathcal{Q}^2 = \{-1 \leq \xi_1, \xi_2 \leq 1\}$ 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, respectively. 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,

$$\int_{Q^2} u(\xi_1, \xi_2)\ d\xi_1\ d\xi_2 \simeq \sum_{i=0}^{Q_1-1} w_i\ f(\xi_{1i}),$$
$$f(\xi_{1i}) = \sum_{j=0}^{Q_2-1} w_j\ u(\xi_{1i}, \xi_{2j}).$$

The corresponding three-dimensional numerical integral for $Q^3 = \{-1 \leq \xi_1, \xi_2, \xi_3 \leq 1\}$ is

$$\int_{Q^3} u(\xi_1, \xi_2, \xi_3)\ d\xi_1\ d\xi_2\ d\xi_3 \simeq \sum_{i=0}^{Q_1-1} w_i \left\{ \sum_{j=0}^{Q_2-1} w_j \left\{ \sum_{i=0}^{Q_3-1} w_k u(\xi_{1i}, \xi_{2j}, \xi_{3k}) \right\} \right\},$$

which is evaluated in a similar fashion to the two-dimensional case except that the innermost summation must be evaluated $Q_1 \cdot Q_2$ times. Although any type of Gauss-Legendre quadrature may be used, the preferred distributions include the end-points as boundary conditions may then be more easily imposed.

4.1.1.2 *Simplex and Hybrid Regions*

Implementation note: *Numerical integration in Ω_{st}: simplex and hybrid regions.*

Triangular Region

Unlike the structured regions, the standard triangular regions $\mathcal{T}^2 = \{-1 \leq \xi_1 \leq \xi_1, \xi_2; \xi_1 + \xi_2 \leq 0\}$ (see figure 3.6) expressed in Cartesian coordinates ξ_1, ξ_2 are not very conveniently represented in terms of one-dimensional Gaussian integration as the upper boundary of the region is described in terms of both coordinates. Since the one-dimensional rule is expressed in terms of an interval with constant bounds (that is, [-1,1]) we need to perform a coordinate transformation before we can apply this technique.

The transformation of a triangular region into a region bounded by constants is equivalent to mapping the triangular region into a quadrilateral as described in section 3.2.1. The collapsed Cartesian system is therefore suitable for Gauss integration in the unstructured regions. For modal expansions, integration in terms of the collapsed Cartesian system remains accurate as the generalised tensorial

bases are polynomials in both the Cartesian and the collapsed Cartesian systems. Although alternative integration schemes using the barycentric coordinate system (see section 3.2.1) have been developed by various researchers including Dunavant [145] and Jinyun [506], this type of integration does not take advantage of the tensorial construction of the unstructured basis as shown in section 4.1.6. The order of these schemes also tends to be restricted by the numerical process of evaluating the quadrature weights.

The two-dimensional collapsed Cartesian system (see section 3.2.1) is defined by the coordinate transformation:

$$\eta_1 = 2\frac{(1+\xi_1)}{(1-\xi_2)} - 1, \qquad \eta_2 = \xi_2.$$

If we express our integral over the region $\mathcal{T}^2$, the collapsed Cartesian system (η_1, η_2), we obtain

$$\begin{aligned}\int_{\mathcal{T}^2} u(\xi_1,\xi_2)\; d\xi_1 d\xi_2 &= \int_{-1}^{1}\int_{-1}^{-\xi_2} u(\xi_1,\xi_2)\; d\xi_1 d\xi_2 \\ &= \int_{-1}^{1}\int_{-1}^{1} u(\eta_1,\eta_2)\left|\frac{\partial(\xi_1,\xi_2)}{\partial(\eta_1,\eta_2)}\right|\; d\eta_1 d\eta_2, \end{aligned} \tag{4.1}$$

where $\partial(\xi_1,\xi_2)/\partial(\eta_1,\eta_2)$ is the Jacobian of the Cartesian to local coordinate transformation and can be expressed in terms of η_1, η_2 by

$$\frac{\partial(\xi_1,\xi_2)}{\partial(\eta_1,\eta_2)} = \left(\frac{1-\eta_2}{2}\right).$$

The last term in equation (4.1) can be approximated using one-dimensional Gaussian quadrature rules to arrive at

$$\int_{-1}^{1}\int_{-1}^{1} u(\eta_1,\eta_2)\left(\tfrac{1-\eta_2}{2}\right) d\eta_1 d\eta_2 = \sum_{i=0}^{Q_1-1} w_i \left\{ \sum_{j=0}^{Q_2-1} w_j\; u(\eta_{1i},\eta_{2j})\left(\tfrac{1-\eta_{2j}}{2}\right)\right\} \tag{4.2}$$

where η_{1i}, η_{2j} are the quadrature points in the η_1 and η_2 directions. The weights w_i and w_j used in equation (4.2) correspond to the standard Gauss-Legendre rule which may or may not include the end-points. However, a more general quadrature rule, which we shall refer to as Gauss-Jacobi quadrature, includes the factor $(1-\xi)^\alpha(1+\xi)^\beta$ in the integrand, that is,

$$\int_{-1}^{1}(1-\xi)^\alpha(1+\xi)^\beta u(\xi)d\xi = \sum_{i=0}^{Q-1} w^{\alpha,\beta} u(\xi_i^{\alpha,\beta}),$$

where $w^{\alpha,\beta}$ and $\xi_i^{\alpha,\beta}$ are the weights and zeros which correspond to the choice of the exponents α and β (see Ghizzetti and Ossicini [180]). If $(\alpha = \beta = 0)$

we recover the standard Gauss-Legendre quadrature rules. The Gauss-Jacobi quadrature rules can be derived for a Lobatto and Radau distribution of zeros as well as the classical Gaussian distribution (see appendix B).

The Gauss-Jacobi rules are convenient in evaluating the integral (4.2) since we are able to include the Jacobian term $\partial(\xi_1,\xi_2)/\partial(\eta_1,\eta_2) = \left(\frac{1-\eta_2}{2}\right)$ directly in the quadrature weights by setting $\alpha = 1, \beta = 0$. Accordingly, the integration scheme over $\mathcal{T}^2$ becomes:

$$\int_{-1}^{1}\int_{-1}^{1} u(\eta_1,\eta_2)\left(\tfrac{1-\eta_2}{2}\right) d\eta_1 d\eta_2 = \sum_{i=0}^{Q_1-1} w_i^{0,0}\left\{\sum_{j=0}^{Q_2-1} \hat{w}_j^{1,0} u(\eta_{1i},\eta_{2j})\right\}$$

where

$$\hat{w}_j^{1,0} = \frac{w_j^{1,0}}{2}.$$

The Gauss-Jacobi rule therefore uses fewer quadrature points than the standard Gauss-Legendre quadrature rule to achieve an equivalent accuracy.

When choosing a distribution of points on which to integrate, the Lobatto-type quadrature is preferred since it includes the end-points of the interval $[-1,1]$, which is helpful when setting boundary conditions. However, when integrating over a triangular region we note that the use of the Radau distribution in the η_2 direction [which includes the point at $(\eta_2 = -1)$] is advantageous as it avoids the need for explicit calculation of any information at the degenerate vertex $(\eta_1 = -1, \eta_2 = 1)$. Although this vertex does not cause any problems when integrating over $\mathcal{T}^2$ it does present added complications when differentiating in $\mathcal{T}^2$ (see section 4.1.2). The distribution of quadrature points in $\mathcal{T}^2$ for $Q_1 = Q_2 = 7$ using a Gauss-Lobatto-Legendre scheme in the η_1 direction and a Gauss-Radau-Jacobi scheme in the η_2 direction is shown in figure 4.1.

Tetrahedral Region

To integrate over $\mathcal{T}^3 = \{-1 \le \xi_1,\xi_2,\xi_3;\ \xi_1+\xi_2+\xi_3 \le -1\}$ we use the collapsed Cartesian coordinate system for the tetrahedron defined as

$$\eta_1 = \frac{2(1+\xi_1)}{(-\xi_2-\xi_3)} - 1, \qquad \eta_2 = \frac{2(1+\xi_2)}{(1-\xi_3)} - 1, \qquad \eta_3 = \xi_3.$$

Using this system the integration becomes

$$\int_{\mathcal{T}^3} u(\xi_1,\xi_2,\xi_3)\ d\xi_1 d\xi_2 d\xi_3 = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} u(\eta_1,\eta_2,\eta_3) J\ d\eta_1 d\eta_2 d\eta_3.$$

where the Jacobian, J, is

$$J = \frac{\partial(\xi_1,\xi_2,\xi_3)}{\partial(\eta_1,\eta_2,\eta_3)} = \left(\frac{1-\eta_2}{2}\right)\left(\frac{1-\eta_3}{2}\right)^2.$$

As in the triangular integration we can include the Jacobian in the quadrature weights by using the Gauss-Jacobi integration rules with $(\alpha = 0, \beta = 0)$ in the

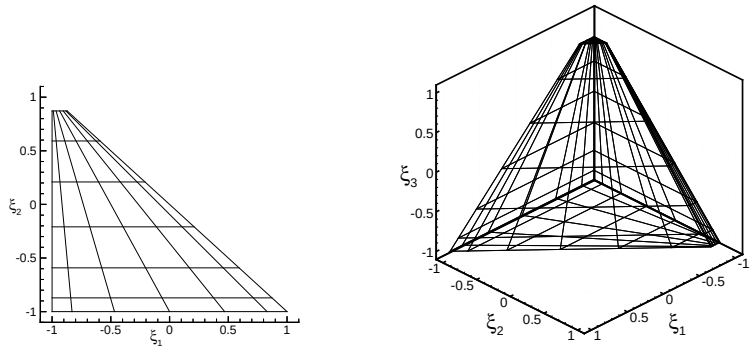


Figure 4.1 Quadrature points in the standard triangular $\mathcal{T}^2$ and tetrahedral $\mathcal{T}^3$ space with $Q_1 = Q_2 = Q_3 = 7$. In the "η_1" direction a Gauss-Lobatto-Legendre distribution has been used and in the "η_2" and "η_3" directions a Gauss-Radau-Jacobi distribution was used.

"η_1" direction (i.e., Gauss-Legendre quadrature), $(\alpha = 1, \beta = 0)$ in the "η_2" direction and $(\alpha = 2, \beta = 0)$ in the "η_3" direction. The integration rule over $\mathcal{T}^3$ then becomes:

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} u(\eta_1,\eta_2,\eta_3)\left(\tfrac{1-\eta_2}{2}\right)\left(\tfrac{1-\eta_3}{2}\right)^2 = \sum_{i=0}^{Q_1-1}\sum_{j=0}^{Q_2-1}\sum_{k=0}^{Q_3-1} u(\eta_{1i}^{0,0},\eta_{2j}^{1,0},\eta_{3k}^{2,0}) w_i^{0,0}\hat{w}_j^{1,0}\hat{w}_k^{2,0},$$

where

$$\hat{w}_j^{1,0} = \frac{w_j^{1,0}}{2}, \qquad \hat{w}_k^{2,0} = \frac{w_k^{2,0}}{4},$$

and Q_1, Q_2, Q_3 are the number of quadrature points in the η_1, η_2, and η_3 directions, respectively.

Once again we can choose any type of point distribution in the quadrature rule (that is, Gauss, Gauss-Radau or Gauss-Lobatto). There are no explicit restrictions as to what type we use. The Gauss-Lobatto can be convenient as it has zeros at the ends of the integration domain, thus allowing greater ease in imposing the boundary conditions. Its use does mean, however, that we have multiple quadrature points at the vertices $(\xi_1 = -1, \xi_2 = -1, \xi_3 = 1)$ and $(\xi_1 = -1, \xi_2 = 1, \xi_3 = -1)$ as well as along the edge between these vertices. As in the two-dimensional case, this is undesirable because of the redundancy of quadrature points and the difficulty of evaluating the derivatives at these points. The use of Gauss-Radau quadrature, which includes a zero at -1 in the "η_2" and "η_3" directions circumvents this problem. The use of Gauss-Lobatto-Legendre quadrature in the "η_1" directions means that, as shown in figure 4.1,

there are quadrature points along all of the boundaries of $\mathcal{T}^3$ except at the vertices $(\xi_1 = -1, \xi_2 = -1, \xi_3 = 1)$ and $(\xi_1 = -1, \xi_2 = 1, \xi_3 = -1)$ and the edge between them.

Prismatic and Pyramidic Regions

To complete the integration section we summarise the quadrature rules in the prismatic and pyramidic regions as shown in figure 3.9. For both these regions it is preferable to use Lobatto-type quadrature in the first two ordinates and Radau type in the third ordinate. This generates as many quadrature points as possible on the boundary of the region while avoiding any complications with differentiation in the region.

For the *prismatic region* we use the local coordinate system

$$\overline{\eta}_1 = \frac{2(1+\xi_1)}{(1-\xi_3)} - 1, \qquad \xi_2, \qquad \xi_3.$$

Integration within this region can be approximated as

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} u(\overline{\eta}_1, \xi_2, \xi_3)\left(\tfrac{1-\xi_3}{2}\right)\, d\overline{\eta}_1 d\xi_2 d\xi_3 \\ &= \sum_{i=0}^{Q_1-1}\sum_{j=0}^{Q_2-1}\sum_{k=0}^{Q_3-1} w_i^{0,0} w_j^{0,0} \hat{w}_k^{1,0} u(\overline{\eta}_{1i}^{0,0}, \xi_{2j}^{0,0}, \xi_{3k}^{1,0})\end{aligned}$$

where

$$\hat{w}_k^{1,0} = \frac{w^{1,0}}{2}.$$

For the *pyramidic region* the local coordinate system is

$$\overline{\eta}_1 = \frac{2(1+\xi_1)}{(1-\xi_3)} - 1, \qquad \eta_2 = \frac{2(1+\xi_2)}{(1-\xi_3)} - 1, \qquad \eta_3 = \xi_3,$$

and integration within this region can be approximated as

$$\begin{aligned}&\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} u(\overline{\eta}_1, \eta_2, \eta_3)\left(\tfrac{1-\eta_3}{2}\right)^2 d\overline{\eta}_1 d\eta_2 d\eta_3 \\ &\qquad = \sum_{i=0}^{Q_1-1}\sum_{j=0}^{Q_2-1}\sum_{k=0}^{Q_3-1} w_i^{0,0} w_j^{0,0} \hat{w}_k^{2,0} u(\overline{\eta}_{1i}^{0,0}, \eta_{2j}^{0,0}, \eta_{3k}^{2,0})\end{aligned}$$

where

$$\hat{w}_k^{2,0} = \frac{w^{2,0}}{4}.$$

4.1.2 Differentiation in the Standard Region Ω_{st}

In formulating a differential problem we typically require the derivative in terms of the general Cartesian coordinates such as $\partial u/\partial x$. We note, however, that the use of a mapping from an elemental region $\boldsymbol{x} \in \Omega_e$ to the standard region $\boldsymbol{\xi} \in \Omega_{st}$ allows us to evaluate the derivative using the chain rule of the form

$$\frac{\partial u}{\partial x_1} = \frac{\partial u}{\partial \xi_1}\frac{\partial \xi_1}{\partial x_1} + \frac{\partial u}{\partial \xi_2}\frac{\partial \xi_2}{\partial x_1} + \frac{\partial u}{\partial \xi_3}\frac{\partial \xi_3}{\partial x_1}.$$

In section 4.1.5 we will introduce the mapping which will allow us to determine the geometric factors $(\frac{\partial \xi_1}{\partial x_1}, \frac{\partial \xi_2}{\partial x_1}, ...)$ to evaluate the full derivatives in terms of Cartesian coordinates.

In this section we first discuss how to differentiate a polynomial function in the local region Ω_{st} to evaluate derivatives of the form $\frac{\partial u}{\partial \xi_1}, \frac{\partial u}{\partial \xi_2}, ...$ As for the one-dimensional case (see section 2.4.2) we shall restrict ourselves to differentiation in physical space, which means that the polynomial function is represented by Lagrange polynomials through a set of discrete points which are typically the quadrature points.

There is no need to distinguish between the modal or nodal expansions discussed in chapter 3 since a modal expansion can always be represented as a Lagrange polynomial of equivalent order, that is,

$$u^\delta(\xi) = \sum_{p=0}^{P} \hat{u}\phi_p(\xi) = \sum_{p=0}^{P} u_p h_p(\xi),$$

where $h_p(\xi)$ is the Lagrange polynomial which passes through a set of $(P+1)$ points. Due to the collocation property of the Lagrangian representation [that is, $h_i(\xi_j) = \delta_{ij}$] the coefficients u_p are the values of the approximation at the nodal point, for example in one-dimension

$$u_p = u^\delta(\xi_p).$$

This collocation property also implies that the derivative of $u^\delta(\xi)$ at the nodal points, ξ_p, can also be described in terms of an expansion in Lagrange polynomials, that is,

$$\frac{\partial u^\delta}{\partial \xi}(\xi) = \sum_{p=0}^{P} u_p \frac{\partial h_p}{\partial \xi}(\xi) = \sum_{p=0}^{P} u'_p h_p(\xi)$$

where

$$u'_p = \sum_{q=0}^{P} u_q \left.\frac{\partial h_q}{\partial \xi}(\xi)\right|_{\xi_p}.$$

This is very significant when calculating non-linear terms such as the advection operator in the Navier-Stokes equation. For example, to determine the value of the non-linear product

$$u^\delta(\xi)\frac{\partial u^\delta}{\partial \xi}(\xi)$$

at a point ξ_i we have:

$$\begin{aligned} u^\delta(\xi_i)\frac{\partial u^\delta}{\partial \xi}(\xi_i) &= \left(\sum_{p=0}^{P} u_p h_p(\xi_i)\right)\left(\sum_{q=0}^{P} u_q \frac{\partial h_q}{\partial \xi}(\xi_i)\right) \\ &= \left(\sum_{p=0}^{P} u_p h_p(\xi_i)\right)\left(\sum_{q=0}^{P} u'_q h_q(\xi_i)\right). \end{aligned}$$

Since $h_p(\xi_i) = \delta_{pi}$ and $h_q(\xi_i) = \delta_{qi}$ then

$$u^\delta(\xi_i)\frac{\partial u^\delta}{\partial \xi}(\xi_i) = u_i u'_i.$$

Finally, we can represent our nonlinear product in terms of an expansion of Lagrange polynomials as

$$u^\delta(\xi)\frac{\partial u^\delta}{\partial \xi}(\xi) \simeq \sum_{p=0}^{P} u_p u'_p h_p(\xi).$$

We note however that if $u^\delta(\xi)$ is a polynomial of order P then the non-linear product $u^\delta(\xi)\frac{\partial u^\delta}{\partial \xi}(\xi)$ is a polynomial of order $(2P-1)$ and so it cannot be exactly represented by the Lagrange polynomial expansion of order P. At the nodal points the coefficient $u_p u'_p$ will be identical to the value of $u^\delta(\xi_p)\frac{\partial u^\delta}{\partial \xi}(\xi_p)$. Nevertheless, projecting the non-linear terms to a lower polynomial order in this fashion can lead to aliasing errors as discussed in section 2.4.1.2.

Although this example is in one-dimension, the same properties apply in multiple dimensions provided the expansion can be represented by a tensor product of Lagrange polynomials. Using the collapsed Cartesian coordinates systems described in section 3.2.1 it is possible to represent any polynomial expansion as a tensor product of one-dimensional Lagrange polynomials.

4.1.2.1 *Two Dimensions Differentiation in the Standard Regions, Ω_{st}*

Implementation note: *Numerical differentiation in Ω_{st}: Quadrilateral and triangular regions.*

Quadrilateral Region

To differentiate an expansion within the standard quadrilateral region Q^2 of the form:

$$u^\delta(\xi_1, \xi_2) = \sum_{p=0}^{P_1}\sum_{q=0}^{P_2} \hat{u}_{pq}\phi_{pq}(\xi_1, \xi_2),$$

we first represent the function in terms of Lagrange polynomials so

$$u^\delta(\xi_1, \xi_2) = \sum_{p=0}^{Q_1-1}\sum_{q=0}^{Q_2-1} u_{pq}\ h_p(\xi_1)h_q(\xi_2),$$

where

$$u_{pq} = u^{\delta}(\xi_{1p}, \xi_{2q}), \qquad Q_1 > P_1, \qquad Q_2 > P_2$$

and ξ_p, ξ_q are typically the zeros of an appropriate Gaussian quadrature. The operation of evaluating u_{pq} from $\hat{u}_{pq}$ is a backwards transformation which we discuss further in section 4.1.5. The partial derivative with respect to ξ_1 is therefore:

$$\frac{\partial u^{\delta}}{\partial \xi_1}(\xi_1, \xi_2) = \sum_{p=0}^{P_1} \sum_{q=0}^{P_2} u_{pq} \frac{dh_p(\xi_1)}{d\xi_1} h_q(\xi_2). \tag{4.3}$$

A procedure for evaluating $dh_p(\xi)/d\xi$ at the Gaussian quadrature points is illustrated in section 2.4.2 and appendix C. From equation (4.3) we can see that to evaluate the partial derivative at an arbitrary point in (ξ_1, ξ_2) we need to perform an $\mathcal{O}(P^2)$ summation over p, q. If we evaluate the derivative at a nodal point (ξ_{1i}, ξ_{2j}) of the Lagrange polynomial the operation count is only $\mathcal{O}(P)$ since $h_q(\xi_{2j}) = \delta_{qj}$, that is,

$$\frac{\partial u^{\delta}}{\partial \xi_1}(\xi_{1i}, \xi_{2j}) = \sum_{p=0}^{P_1} \sum_{q=0}^{P_2} \left\{ u_{pq} \left. \frac{dh_p(\xi_1)}{d\xi_1} \right|_{\xi_{1i}} \delta_{qj} \right\} = \sum_{p=0}^{P_1} u_{pj} \left. \frac{dh_p(\xi_1)}{d\xi_1} \right|_{\xi_{1i}}.$$

For a Galerkin formulation we normally only require the derivatives at the nodal points of the Gaussian quadrature since we typically have to evaluate inner products of the form $(\nabla\phi, \nabla\phi)$. The total cost of evaluating the derivative at $\mathcal{O}(P^2)$ quadrature points will therefore be $\mathcal{O}(P^3)$. The partial derivative with respect to ξ_2 can be evaluated in a similar fashion to arrive at:

$$\frac{\partial u^{\delta}}{\partial \xi_2}(\xi_{1i}, \xi_{2j}) = \sum_{q=0}^{P_2} u_{iq} \left. \frac{dh_q(\xi_2)}{d\xi_2} \right|_{\xi_{2j}}.$$

Triangular Region

For the triangular region, $\mathcal{T}^2$, we can also represent any polynomial expansion in terms of the Lagrange polynomial using the collapsed coordinates η_1, η_2:

$$u^{\delta}(\xi_1, \xi_2) = \sum_{p,q}^{P_1, P_2} \hat{u}_{pq}\ \phi_{pq}(\eta_1, \eta_2) = \sum_{p=0}^{P_1} \sum_{q=0}^{P_2} u_{pq} h_p(\eta_1) h_q(\eta_2),$$

where

$$u_{pq} = u^{\delta}(\eta_{1p}, \eta_{2q}), \qquad \eta_1 = \frac{2(1+\xi_1)}{(1-\xi_2)} - 1, \qquad \eta_2 = \xi_2,$$

and η_{1p}, η_{2q} refers to the nodal points of the Lagrange polynomial. The summation over the indices p, q for the modified triangular expansion is dependent upon P_1, P_2 but does not have a *close packed* form and so it cannot be summed

consecutively. However, if $Q_1 > P_1$ and $Q_2 > P_2$ then the polynomial space of the basis $\phi_{pq}(\eta_1, \eta_2)$ is a subset of the space spanned by the Lagrange polynomials $h_p(\eta_1)h_q(\eta_2)$. The partial derivative with respect to Cartesian system ξ_1 and ξ_2 may be determined by applying the chain rule:

$$\begin{pmatrix} \dfrac{\partial}{\partial \xi_1} \\ \\ \dfrac{\partial}{\partial \xi_2} \end{pmatrix} = \begin{pmatrix} \dfrac{2}{(1-\eta_2)} \dfrac{\partial}{\partial \eta_1} \\ \\ 2\dfrac{(1+\eta_1)}{(1-\eta_2)} \dfrac{\partial}{\partial \eta_1} + \dfrac{\partial}{\partial \eta_2} \end{pmatrix}. \tag{4.4}$$

Similarly, to differentiate in the quadrilateral region, the value of the partial derivative with respect to η_1 and η_2 at the nodal points is given by

$$\frac{\partial u^\delta}{\partial \eta_1}(\eta_{1i}, \eta_{2j}) = \sum_{p=0}^{P_1} u_{pj} \left. \frac{dh_p(\eta_1)}{d\eta_1} \right|_{\eta_{1i}} \tag{4.5a}$$

$$\frac{\partial u^\delta}{\partial \eta_2}(\eta_{1i}, \eta_{2j}) = \sum_{q=0}^{P_2} u_{iq} \left. \frac{dh_q(\eta_2)}{d\eta_2} \right|_{\eta_{2j}}. \tag{4.5b}$$

Finally, substituting equations (4.5a) and (4.5b) into (4.4) we obtain the partial derivatives of the function u with respect to the Cartesian coordinates (ξ_1, ξ_2) (see section 2.4.2 and appendix C for the formula to determine $dh_q(\eta)/d\eta$).

Before considering the three-dimensional cases we note that determining the derivative at $(\eta_2 = 1)$ presents a problem as the coefficients of the $\partial/\partial\eta_1$ term in equation (4.4) have a numerator which tends to zero as $\eta_2 \to 0$. Although the derivative is well defined at this point, from a computational viewpoint special treatment is necessary. This issue may, however, be circumvented by using a Radau-type Gaussian quadrature in the η_2 direction. An analytic form of the derivative at $(\xi_1 = -1, \xi_2 = 1)$ may be constructed from the modal representation of the expansion, for example, the partial derivative with respect to ξ_1 may be written:

$$\frac{\partial u}{\partial \xi_1}(\eta_1, \eta_2) = \sum_{p,q}^{P_1,P_2} \hat{u}_{pq} \frac{2}{(1-\eta_2)} \frac{\partial \phi_{pq}}{\partial \eta_1}(\eta_1, \eta_2).$$

This summation still appears to be singular at $(-1, 1)$ but we note that all of the modes ϕ_{pq}, except one, contain the factor $(1 - \eta_2)$ which cancels the $1/(1 - \eta_2)$ term. The only mode which does not contain this factor corresponds to the top vertex mode which is independent of η_1 and therefore is necessarily zero when we take the partial derivative with respect to η_1. Furthermore, this summation need not be evaluated over all $\mathcal{O}(P^2/2)$ modes as many modes contain a factor of $(1 - \eta_2)^k$ where $k > 1$ and therefore, even when differentiated, will be zero at $\eta_2 = 1$.

4.1.2.2 *Three-Dimensional Differentiation in the Standard Regions, Ω_{st}*

Implementation note: *Numerical differentiation in Ω_{st}: Three dimensional regions.*

The three-dimensional derivatives are analogous to the two-dimensional case save that the function is now described in terms of a tensor product of three one-dimensional Lagrange polynomials, that is,

$$u^\delta(\xi_1,\xi_2,\xi_3) = \sum_p^{P_1}\sum_q^{P_2}\sum_r^{P_3} u_{pqr} h_p(\xi_1) h_q(\xi_2) h_r(\xi_3)$$

and so the partial derivative with respect to ξ_1 is:

$$\frac{\partial u^\delta}{\partial \xi_1}(\xi_1,\xi_2,\xi_3) = \sum_p^{P_1}\sum_q^{P_2}\sum_r^{P_3} u_{pqr} \frac{\partial h_p(\xi_1)}{\partial \xi_1} h_q(\xi_2) h_r(\xi_3),$$

which when evaluated at the nodal points of the Lagrange polynomials becomes:

$$\frac{\partial u^\delta}{\partial \xi_1}(\xi_{1i},\xi_{2j},\xi_{3k}) = \sum_p^{P_1} u_{pjk} \left.\frac{\partial h_p(\xi_1)}{\partial \xi_1}\right|_{\xi_{1i}}. \tag{4.34a}$$

Similarly, the other two partial derivatives at nodal points are given by

$$\frac{\partial u^\delta}{\partial \xi_2}(\xi_{1i},\xi_{2j},\xi_{3k}) = \sum_q^{P_2} u_{iqk} \left.\frac{\partial h_q(\xi_2)}{\partial \xi_2}\right|_{\xi_{2j}} \tag{4.34b}$$

$$\frac{\partial u^\delta}{\partial \xi_3}(\xi_{1i},\xi_{2j},\xi_{3k}) = \sum_r^{P_3} u_{ijr} \left.\frac{\partial h_r(\xi_3)}{\partial \xi_3}\right|_{\xi_{3k}}. \tag{4.34c}$$

Each of these expressions takes an $\mathcal{O}(P)$ operation to evaluate a single derivative, which implies that the cost to evaluate a derivative at $\mathcal{O}(P^3)$ quadrature points within the region is $\mathcal{O}(P^4)$. The formula for evaluating $dh_p(\xi)/d\xi$ can be found in section 2.4.2 and appendix C.

Equations (4.34a), (4.34b), and (4.34c) clearly define the partial derivatives for the *hexahedral region*. If, however, we replace the Cartesian coordinates by any of the local collapsed Cartesian coordinates we can interpret these equations as representing the partial derivatives with respect to the local coordinates. Just as in the two-dimensional case for the triangular region, we can use the chain rule to obtain the local partial derivatives with respect to the Cartesian coordinates. For hybrid three-dimensional domains the partial derivatives are evaluated using the expressions:

Tetrahedral Region:

$$\begin{pmatrix} \dfrac{\partial}{\partial \xi_1} \\ \dfrac{\partial}{\partial \xi_2} \\ \dfrac{\partial}{\partial \xi_3} \end{pmatrix} = \begin{pmatrix} \dfrac{4}{(1-\eta_2)(1-\eta_3)} \dfrac{\partial}{\partial \eta_1} \\ \dfrac{2(1+\eta_1)}{(1-\eta_2)(1-\eta_3)} \dfrac{\partial}{\partial \eta_1} + \dfrac{2}{(1-\eta_3)} \dfrac{\partial}{\partial \eta_2} \\ \dfrac{2(1+\eta_1)}{(1-\eta_2)(1-\eta_3)} \dfrac{\partial}{\partial \eta_1} + \dfrac{(1+\eta_2)}{(1-\eta_3)} \dfrac{\partial}{\partial \eta_2} + \dfrac{\partial}{\partial \eta_3} \end{pmatrix},$$

where

$$\eta_1 = 2\frac{(1+\xi_1)}{(-\xi_2-\xi_3)} - 1, \qquad \eta_2 = 2\frac{(1+\xi_2)}{(1-\xi_3)} - 1, \qquad \eta_3 = \xi_3.$$

Pyramidic Region:

$$\begin{pmatrix} \dfrac{\partial}{\partial \xi_1} \\ \dfrac{\partial}{\partial \xi_2} \\ \dfrac{\partial}{\partial \xi_3} \end{pmatrix} = \begin{pmatrix} \dfrac{2}{(1-\eta_3)} \dfrac{\partial}{\partial \overline{\eta_1}} \\ \dfrac{2}{(1-\eta_3)} \dfrac{\partial}{\partial \eta_2} \\ \dfrac{(1+\overline{\eta_1})}{(1-\eta_3)} \dfrac{\partial}{\partial \overline{\eta_1}} + \dfrac{(1+\eta_2)}{(1-\eta_3)} \dfrac{\partial}{\partial \eta_2} + \dfrac{\partial}{\partial \eta_3} \end{pmatrix},$$

where

$$\overline{\eta_1} = 2\frac{(1+\xi_1)}{(1-\xi_3)} - 1, \qquad \eta_2 = 2\frac{(1+\xi_2)}{(1-\xi_3)} - 1, \qquad \eta_3 = \xi_3.$$

Prismatic Region:

$$\begin{pmatrix} \dfrac{\partial}{\partial \xi_1} \\ \dfrac{\partial}{\partial \xi_2} \\ \dfrac{\partial}{\partial \xi_3} \end{pmatrix} = \begin{pmatrix} \dfrac{2}{(1-\eta_3)} \dfrac{\partial}{\partial \overline{\eta}_1} \\ \dfrac{\partial}{\partial \xi_2} \\ \dfrac{(1+\overline{\eta}_1)}{(1-\eta_3)} \dfrac{\partial}{\partial \overline{\eta}_1} + \dfrac{\partial}{\partial \eta_3} \end{pmatrix},$$

where

$$\overline{\eta}_1 = 2\frac{(1+\xi_1)}{(1-\xi_3)} - 1, \qquad \eta_3 = \xi_3.$$

For the tetrahedral, prismatic, and pyramidic regions there is a potential problem when $\eta_2 = 1$ or $\eta_3 = 1$ due to the factors $\frac{1}{(1-\eta_2)(1-\eta_3)}$ and $\frac{1}{(1-\eta_3)}$. As

before, evaluation of the derivative at this point can be avoided by the use of Gauss-Radau quadrature in both the "η_2" and "η_3" directions for the tetrahedral region, and in the "η_3" direction for the prismatic and pyramidic regions. The derivatives are well defined in these regions although evaluation at $\eta_2 = 1$ or $\eta_3 = 1$ does require analytic differentiation of the expansion modes or interpolation from the derivatives evaluated at the Gauss-Radau distribution of points.

4.1.3 Operations within General Shaped Elements

We have seen how to integrate and differentiate within the standard region Ω_{st}, however, in practice we need to perform these operations in the elemental regions, Ω^e, which may be of a generalised shape and orientation as illustrated in figure 4.2. To consider these cases we need to define a one-to-one mapping between the Cartesian coordinates (x_1, x_2) and the local Cartesian coordinates (ξ_1, ξ_2) which are denoted by

$$x_1 = \chi_1^e(\xi_1, \xi_2), \qquad x_2 = \chi_2^e(\xi_1, \xi_2)$$

in two-dimensions, and similarly

$$x_1 = \chi_1^e(\xi_1, \xi_2, \xi_3), \qquad x_2 = \chi_2^e(\xi_1, \xi_2, \xi_3), \qquad x_3 = \chi_3^e(\xi_1, \xi_2, \xi_3)$$

in three-dimensions.

In section 4.1.3.1 we start by discussing how to define a mappings χ_i^e, from the elemental region to the standard region for straight sided elements which simply requires information about the vertices of an element. Elements can also be curvilinear although in this case some information about how the edges (or faces in three–dimensions) are curved is also required. When this is known, we can define a more complex elemental mapping as discussed in section 4.1.3.2. Regardless of the complexity of the mapping, we still need to know how to integrate and differentiate in a general domain. This is discussed in sections 4.1.3.3 and 4.1.3.4. Finally, in section 4.1.4 we discuss how to evaluate the Jacobian of the mapping between a curved boundary and a standard region as typically required in a surface integral term.

4.1.3.1 *Elemental Mappings for General Straight-Sided Elements*

For elemental shapes with straight sides a simple mapping may be constructed using the linear vertex modes of a modified hierarchical/modal expansion. For example, to map a triangular region [as in figure 4.2(a)] assuming that the global coordinates of the triangle $\{(x_1^A, x_2^A), (x_1^B, x_2^B), (x_1^C, x_2^C)\}$ are known (with C the collapsed vertex), we can use

$$x_i = \chi_1^e(\eta_1, \eta_2) = x_i^A \frac{(1-\eta_1)}{2}\frac{(1-\eta_2)}{2} + x_i^B \frac{(1+\eta_1)}{2}\frac{(1-\eta_2)}{2} + x_i^C \frac{(1+\eta_2)}{2}, \qquad i = 1, 2. \quad (4.35)$$

Equation (4.35) is expressed in terms of collapsed Cartesian coordinates but can easily be expressed in terms of the Cartesian coordinates by recalling that $(\eta_1 = 2\frac{(1+\xi_1)}{(1-\xi_2)} - 1, \eta_2 = \xi_2)$ which on substitution into (4.35) gives:

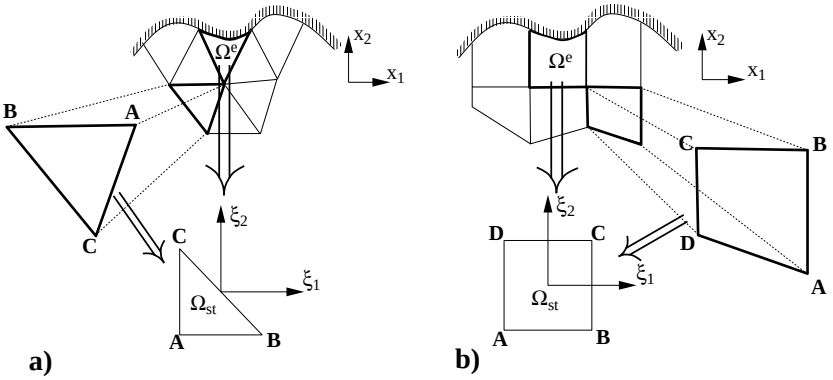


Figure 4.2 To construct a C^0 expansion from multiple elements of specified shapes (for example, triangles or rectangles), each elemental region Ω^e is mapped to a standard region Ω_{st} in which all local operations are evaluated.

$$x_i = \chi(\xi_1, \xi_2) = x_i^A \frac{(-\xi_2 - \xi_1)}{2} + x_i^B \frac{(1+\xi_1)}{2} + x_i^C \frac{(1+\xi_2)}{2}. \quad i = 1, 2 \qquad (4.36)$$

A similar approach leads to the bilinear mapping for an arbitrary shaped straight-sided quadrilateral where only the vertices need to be prescribed. For the straight-sided quadrilateral with vertices labelled as shown in figure 4.2(b) the mapping is:

$$\begin{aligned} x_i = \chi_1(\xi_1, \xi_2) = x_i^A \frac{(1-\xi_1)}{2}\frac{(1-\xi_2)}{2} + x_i^B \frac{(1+\xi_1)}{2}\frac{(1-\xi_2)}{2} \\ + x_i^D \frac{(1-\xi_1)}{2}\frac{(1+\xi_2)}{2} + x_i^C \frac{(1+\xi_1)}{2}\frac{(1+\xi_2)}{2}. \quad i = 1, 2 \end{aligned} \qquad (4.37)$$

When developing a mapping it is important to ensure that the Jacobian of the mapping to the standard region is non-zero and of the same sign. To ensure this condition is satisfied when using the mappings given above, we require all elemental regions to have internal corners with angles that are less than 180° and so are convex. It is not, in fact, possible to generate a straight-sided triangular or tetrahedral region which does not satisfy this condition, but, as shown in figure 4.3, it is certainly possible within a quadrilateral or other three-dimensional regions.

4.1.3.2 *Elemental Mappings for General Curvilinear Elements*

For a general straight-sided elemental domain we have seen in section 4.1.3.1 that a one-to-one linear mapping can be constructed onto the standard region using the vertex modes of the hierarchical modal expansion. For example, a quadrilateral domain of the form shown in figure 4.2(b) the mapping can be defined by equation (4.37).

We note that this simply involves the vertex modes of the modified hierarchical expansion basis within a quadrilateral domain (see section 3.1.1). We could, therefore, have written the expansion as

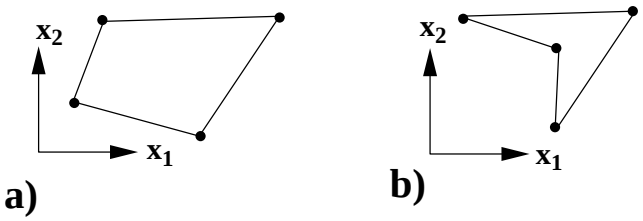


Figure 4.3 (a) Illustration of permissible quadrilateral region with all interior angles less than 180°. (b) Illustration of non-permissible quadrilateral region, which has a re-entrant corner and so has interior angles which are greater than 180°.

$$x_i = \chi_i(\xi_1, \xi_2) = \sum_{p=0}^{p=P_1} \sum_{q=0}^{q=P_2} \hat{x}^i_{pq} \phi_{pq}(\xi_1, \xi_2) \tag{4.38}$$

where $\phi_{pq} = \psi^a_p(\xi_1)\psi^a_q(\xi_2)$ and $\hat{x}^i_{pq} = 0$ except for the vertex modes which have a value of

$$\hat{x}^i_{00} = x^A_i \qquad \hat{x}^i_{P_1 0} = x^B_i \qquad \hat{x}^i_{P_1 P_2} = x^C_i \qquad \hat{x}^i_{0 P_2} = x^D_i.$$

The construction of a mapping based upon the expansion modes in this form can be extended to include curved sided regions using an *isoparametric* representation. In this technique the geometry is represented with an expansion of the same form and polynomial order as the unknown variables.

To describe a straight-sided region we needed only to know the values of the vertex locations. To describe a curved region, however, requires more information. As illustrated in figure 4.4, we typically expect to have a definition of a mapping of the shape of each edge in terms of the local coordinates which we denote as $f^A_i(\xi_1), f^B_i(\xi_2), f^C_i(\xi_1)$ and $f^D_i(\xi_2)$. The process of defining the mapping functions can be considered as part of the mesh generation process, the discussion of which is in section 4.3.3.

Knowing the definition of the edges (or faces in three-dimensions) a mapping for a curvilinear domains can be determined using the isoparametric form of equation (4.38) to include more non-zero expansion coefficients than simply the vertex contributions. In two-dimensions we wish to use the coefficient along each edge of the element, and in three-dimensions we can use the face coefficients as well. Along each edge we therefore need to approximate the shape mapping $f_i(\xi)$ if it is not already represented by a polynomial of appropriate order. We might therefore consider the following approximations for $f^A_i(\xi_1)$

$$f^A(\xi_1) \simeq \sum_{p=0} f^A(\xi_i, i) h_p(\xi_1) \tag{4.39a}$$

$$\simeq \sum_{p=0} \hat{x}^i_{p0} \psi_p(\xi_1). \tag{4.39b}$$

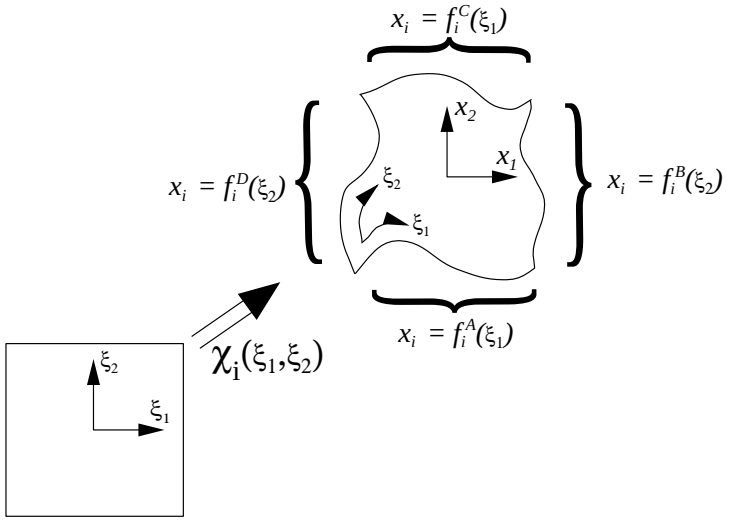


Figure 4.4 A general curved element can be described in terms of a series of parametric functions $f^A(\xi_1), f^B(\xi_2), f^C(\xi_1)$, and $f^D(\xi_2)$. Representing these functions as a discrete expansion we can construct an iso-parametric mapping $\chi_i(\xi_1, \xi_2)$ relating the standard region (ξ_1, ξ_2) to the deformed region (x_1, x_2).

One important feature of the approximation and consequently the mapping χ_i is that the vertices of each element coincide so that elements remain continuous. One way to ensure this is to use a collocation projection where the collocation points include the end-points $\xi_1 \pm 1$. The Lagrange representation of equation (4.39a) is therefore a consistent way of approximating $f_i^A(\xi)$. Using the Gauss-Lobatto-Legendre quadrature points for our collocation projection is attractive due to their small Lebesgue constant associated with this interpolation as discussed in section 3.3.1. Similarly, within a triangular face of a three-dimensional element the Fekete or electrostatic points are a favourable choice for collocation points. By making collocation projections at a series of nodal points we have then represented the function f^A as a polynomial which can be equivalently expressed in terms of a hierarchical expansion, $\psi_p(\xi)$ to obtain the coefficients $\hat{x}^i_{p0}$ in equation (4.39b). This final transformation can be performed either by a collocation or Galerkin projection if the polynomials span the same space. If more collocation points are used then a modified Galerkin projection as discussed in section 4.3.1 can be applied. Having determined the coordinate expansion coefficients, $\hat{x}^i_{p0}$, we can then evaluate equation (4.38) to determine the isoparametric mapping from the standard region to the curvi-linear region.

We note that the form of the boundary-interior decomposition of the modal quadrilateral and hexahedral expansion is discretely equivalent to using a linear blending function as originally proposed by Gordon and Hall [194]. For the quadrilateral region shown in figure 4.4 the linear blending function is given by

$$\begin{aligned}\chi_i(\xi_1,\xi_2) = {} & f^A(\xi_1)\tfrac{(1-\xi_2)}{2} + f^C(\xi_1)\tfrac{(1+\xi_2)}{2} \\ & + f^D(\xi_2)\tfrac{(1-\xi_1)}{2} + f^B(\xi_2)\tfrac{(1+\xi_1)}{2} \\ & - \tfrac{(1-\xi_1)}{2}\tfrac{(1-\xi_2)}{2} f^A(-1) - \tfrac{(1+\xi_1)}{2}\tfrac{(1-\xi_2)}{2} f^A(1) \\ & - \tfrac{(1-\xi_1)}{2}\tfrac{(1+\xi_2)}{2} f^C(-1) - \tfrac{(1+\xi_1)}{2}\tfrac{(1+\xi_2)}{2} f^C(1),\end{aligned} \tag{4.40}$$

where the vertex points are continuous [for example, $f^A(-1) = f^D(-1)$]. If we replace the analytic curves $f^A(\xi_1), f^B(\xi_2), f^C(\xi_1)$ and $f^D(\xi_2)$ in expression (4.40) with an approximate expansion of the type given in equation (4.39a) and rearrange, we can obtain an expansion of the form given by equation (4.38). The blending function (4.40) with approximations of the form (4.39a) to the mapped edges has traditionally been applied in spectral element methods. However, we note the aforementioned similarity with the modal expansion approach.

As a final point we note that for curved triangular or tetrahedral elements, the linear blending function expressed in terms of the local collapsed coordinates η_1, η_2, η_3 should not be used as this can generate a non-smooth Jacobian (at the singular vertices) and cause a loss in exponential convergence of smooth functions in curved domains. The use of the isoparametric representation of the coordinates (equation 4.38) generates a mapping with a sufficiently smooth Jacobian.

4.1.3.3 *Integration within a General Shaped Elemental Region*

Formulation note: *Integration within a general region: definition of the 2D and 3D Jacobians.*

We denote an arbitrary triangular or 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:

$$J_{2D} = \begin{vmatrix} \dfrac{\partial x_1}{\partial \xi_1} & \dfrac{\partial x_1}{\partial \xi_2} \\ \dfrac{\partial x_2}{\partial \xi_1} & \dfrac{\partial x_2}{\partial \xi_2} \end{vmatrix} = \frac{\partial x_1}{\partial \xi_1}\frac{\partial x_2}{\partial \xi_2} - \frac{\partial x_1}{\partial \xi_2}\frac{\partial x_2}{\partial \xi_1}. \tag{4.41}$$

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 as discussed in section 4.1.2. If the elemental region is straight-sided then we have seen that a mapping from $(x_1, x_2) \rightarrow (\xi_1, \xi_2)$ is given by equations (4.36) and (4.37). The simple form of these mappings means that the partial derivatives, and therefore the Jacobian, are constant for quadrilateral regions with similar shape and orientation to the standard region, as well as for all *triangular* regions. For deformed regions the Jacobian may be evaluated

and stored at the quadrature points. This essentially represents the Jacobian as a polynomial function and can therefore increase the polynomial order of the integrand.

We note that integration over the triangular region now involves two transformations, that is, $(x_1, x_2) \rightarrow (\xi_1, \xi_2)$ and $(\xi_1, \xi_2) \rightarrow (\eta_1, \eta_2)$. However, the second transformation $(\xi_1, \xi_2) \rightarrow (\eta_1, \eta_2)$ may be absorbed entirely into the quadrature weights as discussed in section 4.1.1. This is preferable since the polynomial order, which may be exactly integrated, is higher. Having evaluated the Jacobian at the quadrature points, we multiply the integrand by the Jacobian and evaluate the integral in the same manner as discussed in section 4.1.1.

Integration over a three-dimensional region Ω^e is performed in an analogous fashion where the three-dimensional Jacobian now has the form:

$$J_{3D} = \begin{vmatrix} \frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_2} & \frac{\partial x_1}{\partial \xi_3} \\ \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_2} & \frac{\partial x_2}{\partial \xi_3} \\ \frac{\partial x_3}{\partial \xi_1} & \frac{\partial x_3}{\partial \xi_2} & \frac{\partial x_3}{\partial \xi_3} \end{vmatrix} = \begin{array}{l} \frac{\partial x_1}{\partial \xi_1}\left(\frac{\partial x_2}{\partial \xi_2}\frac{\partial x_3}{\partial \xi_3} - \frac{\partial x_2}{\partial \xi_3}\frac{\partial x_3}{\partial \xi_2}\right) \\ -\frac{\partial x_1}{\partial \xi_2}\left(\frac{\partial x_2}{\partial \xi_1}\frac{\partial x_3}{\partial \xi_3} - \frac{\partial x_2}{\partial \xi_3}\frac{\partial x_3}{\partial \xi_1}\right) \\ +\frac{\partial x_1}{\partial \xi_3}\left(\frac{\partial x_2}{\partial \xi_1}\frac{\partial x_3}{\partial \xi_2} - \frac{\partial x_2}{\partial \xi_2}\frac{\partial x_3}{\partial \xi_1}\right). \end{array} \tag{4.42}$$

4.1.3.4 *Differentiation within a General Shaped Elemental Region*

Formulation note: *Differentiation within a general region in terms of local Cartesian coordinates: Geometric factors.*

To differentiate a function within the arbitrary region Ω^e as illustrated in figure 4.2 we once again apply the chain rule which, for the 2D case, gives:

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi_1}{\partial x_1}\frac{\partial}{\partial \xi_1} + \frac{\partial \xi_2}{\partial x_1}\frac{\partial}{\partial \xi_2} \\ \frac{\partial \xi_1}{\partial x_2}\frac{\partial}{\partial \xi_1} + \frac{\partial \xi_2}{\partial x_2}\frac{\partial}{\partial \xi_2} \end{bmatrix}. \tag{4.43}$$

In section 4.1.1 we illustrated differentiation with respect to ξ_1 and ξ_2 but we now also need to evaluate partial derivatives of the form $\partial \xi_1 / \partial x_1$. For the linear mapping case given by equations (4.36) and (4.37) it is possible to obtain an analytic formula, but in general we need a technique to handle a curvilinear elemental region. To do this, we express the partial derivatives such as $\partial \xi_1 / \partial x_1$ in terms of partial derivatives with respect to ξ_1, ξ_2, which we already know how to evaluate. For a general function dependent on two variables, $u(\xi_1, \xi_2)$ we know from the chain rule that the total change in $u(\xi_1, \xi_2)$ is

$$du(\xi_1, \xi_2) = \frac{\partial u}{\partial \xi_1} d\xi_1 + \frac{\partial u}{\partial \xi_2} d\xi_2. \tag{4.44}$$

If we replace $u(\xi_1, \xi_2)$ by $x_1 = \chi_1(\xi_1, \xi_2)$ and $x_2 = \chi_2(\xi_1, \xi_2)$ we obtain the matrix system

$$\begin{bmatrix} dx_1 \\ \\ dx_2 \end{bmatrix} = \begin{bmatrix} \dfrac{\partial x_1}{\partial \xi_1} & \dfrac{\partial x_1}{\partial \xi_2} \\ \\ \dfrac{\partial x_2}{\partial \xi_1} & \dfrac{\partial x_2}{\partial \xi_2} \end{bmatrix} \begin{bmatrix} d\xi_1 \\ \\ d\xi_2 \end{bmatrix},$$

which can be inverted to obtain

$$\begin{bmatrix} d\xi_1 \\ \\ d\xi_2 \end{bmatrix} = \frac{1}{J} \begin{bmatrix} \dfrac{\partial x_2}{\partial \xi_2} & -\dfrac{\partial x_1}{\partial \xi_2} \\ \\ -\dfrac{\partial x_2}{\partial \xi_1} & \dfrac{\partial x_1}{\partial \xi_1} \end{bmatrix} \begin{bmatrix} dx_1 \\ \\ dx_2 \end{bmatrix}, \tag{4.45}$$

where J is the Jacobian defined by equation (4.41). However, as the mapping is assumed to be one-to-one and have an inverse we assume that $\xi_1 = (\chi_1)^{-1}(x_1, x_2)$ and $\xi_2 = (\chi_2)^{-1}(x_1, x_2)$ and so we can apply the chain rule directly to ξ_1, ξ_2 to obtain:

$$\begin{bmatrix} d\xi_1 \\ \\ d\xi_2 \end{bmatrix} = \frac{1}{J} \begin{bmatrix} \dfrac{\partial \xi_1}{\partial x_1} & -\dfrac{\partial \xi_1}{\partial x_2} \\ \\ -\dfrac{\partial \xi_2}{\partial x_1} & \dfrac{\partial \xi_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} dx_1 \\ \\ dx_2 \end{bmatrix}. \tag{4.46}$$

Finally, equating (4.45) and (4.46) we see that:

$$\frac{\partial \xi_1}{\partial x_1} = \frac{1}{J}\frac{\partial x_2}{\partial \xi_2}, \qquad \frac{\partial \xi_1}{\partial x_2} = -\frac{1}{J}\frac{\partial x_1}{\partial \xi_2}, \qquad \frac{\partial \xi_2}{\partial x_1} = -\frac{1}{J}\frac{\partial x_2}{\partial \xi_1}, \qquad \frac{\partial \xi_2}{\partial x_2} = \frac{1}{J}\frac{\partial x_1}{\partial \xi_1}.$$

We can now evaluate the two-dimensional gradient operator in equation (4.43) as all the partial derivatives can be expressed in terms of differentials with respect to ξ_1, ξ_2 which may be evaluated using the Lagrange polynomial representation explained in section 4.1.2.

For the three-dimensional gradient operator we assume that the coordinates x_1, x_2, x_3 are also known in terms of mappings dependent upon ξ_1, ξ_2, ξ_3. So applying the chain rule we obtain

$$\nabla = \begin{bmatrix} \dfrac{\partial}{\partial x_1} \\ \\ \dfrac{\partial}{\partial x_2} \\ \\ \dfrac{\partial}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \xi_1}{\partial x_1}\dfrac{\partial}{\partial \xi_1} + \dfrac{\partial \xi_2}{\partial x_1}\dfrac{\partial}{\partial \xi_2} + \dfrac{\partial \xi_3}{\partial x_1}\dfrac{\partial}{\partial \xi_3} \\ \\ \dfrac{\partial \xi_1}{\partial x_2}\dfrac{\partial}{\partial \xi_1} + \dfrac{\partial \xi_2}{\partial x_2}\dfrac{\partial}{\partial \xi_2} + \dfrac{\partial \xi_3}{\partial x_2}\dfrac{\partial}{\partial \xi_3} \\ \\ \dfrac{\partial \xi_1}{\partial x_3}\dfrac{\partial}{\partial \xi_1} + \dfrac{\partial \xi_2}{\partial x_3}\dfrac{\partial}{\partial \xi_2} + \dfrac{\partial \xi_3}{\partial x_3}\dfrac{\partial}{\partial \xi_3} \end{bmatrix}. \tag{4.47}$$

Following a similar analysis to that used for the two-dimensional case we express the partial derivatives with respect to x_1, x_2, and x_3 in terms of derivatives with respect to ξ_1, ξ_2 and ξ_3 using the relations:

$$\frac{\partial \xi_1}{\partial x_1} = \frac{1}{J_{3D}} \left(\frac{\partial x_2}{\partial \xi_2} \frac{\partial x_3}{\partial \xi_3} - \frac{\partial x_2}{\partial \xi_3} \frac{\partial x_3}{\partial \xi_2} \right), \quad \frac{\partial \xi_1}{\partial x_2} = -\frac{1}{J_{3D}} \left(\frac{\partial x_1}{\partial \xi_2} \frac{\partial x_3}{\partial \xi_3} - \frac{\partial x_1}{\partial \xi_3} \frac{\partial x_3}{\partial \xi_2} \right),$$

$$\frac{\partial \xi_1}{\partial x_3} = \frac{1}{J_{3D}} \left(\frac{\partial x_1}{\partial \xi_2} \frac{\partial x_2}{\partial \xi_3} - \frac{\partial x_1}{\partial \xi_3} \frac{\partial x_2}{\partial \xi_2} \right),$$

$$\frac{\partial \xi_2}{\partial x_1} = -\frac{1}{J_{3D}} \left(\frac{\partial x_2}{\partial \xi_1} \frac{\partial x_3}{\partial \xi_3} - \frac{\partial x_2}{\partial \xi_3} \frac{\partial x_3}{\partial \xi_1} \right), \quad \frac{\partial \xi_2}{\partial x_2} = \frac{1}{J_{3D}} \left(\frac{\partial x_1}{\partial \xi_1} \frac{\partial x_3}{\partial \xi_3} - \frac{\partial x_1}{\partial \xi_3} \frac{\partial x_3}{\partial \xi_1} \right),$$

$$\frac{\partial \xi_2}{\partial x_3} = -\frac{1}{J_{3D}} \left(\frac{\partial x_1}{\partial \xi_1} \frac{\partial x_2}{\partial \xi_3} - \frac{\partial x_1}{\partial \xi_3} \frac{\partial x_2}{\partial \xi_1} \right),$$

$$\frac{\partial \xi_3}{\partial x_1} = \frac{1}{J_{3D}} \left(\frac{\partial x_2}{\partial \xi_1} \frac{\partial x_3}{\partial \xi_2} - \frac{\partial x_2}{\partial \xi_3} \frac{\partial x_3}{\partial \xi_1} \right), \quad \frac{\partial \xi_3}{\partial x_2} = -\frac{1}{J_{3D}} \left(\frac{\partial x_1}{\partial \xi_1} \frac{\partial x_3}{\partial \xi_2} - \frac{\partial x_1}{\partial \xi_2} \frac{\partial x_3}{\partial \xi_1} \right),$$

$$\frac{\partial \xi_3}{\partial x_3} = \frac{1}{J_{3D}} \left(\frac{\partial x_1}{\partial \xi_1} \frac{\partial x_2}{\partial \xi_2} - \frac{\partial x_1}{\partial \xi_2} \frac{\partial x_2}{\partial \xi_1} \right),$$

where J_{3D} is as defined in equation (4.42).

4.1.4 Discrete Evaluation of the Surface Jacobian

As a final local operation which combines the concepts of elemental mapping, integration and differentiation we consider the evaluation of the surface Jacobian. Surface integrals of the form

$$< v, g_{\mathcal{N}} >= \int_{\partial\Omega_{\mathcal{N}}} v g_{\mathcal{N}} dS = \sum_{e=1}^{N_{el}} \int_{\partial\Omega_{\mathcal{N}} \bigcap \partial\Omega^e} v^e g_{\mathcal{N}} dS^e \tag{4.48}$$

typically arise in the Galerkin discretisation of the Laplacian operators due to the application of the divergence theorem. Since these types of integrals normally appear as right-hand-side contributions in our matrix problems they can be evaluated as a series of integrals over different elemental boundaries. Therefore, in a similar way to integration in the elemental region our strategy for evaluating (4.48) is to transform each elemental contribution to a standard interval and perform the integration using Gaussian quadrature. This transformation to a standard region necessarily introduces a Jacobian which we shall refer to as the *surface Jacobian*.

4.1.4.1 *Surface Jacobian of a Two-Dimensional Region*

In two-dimensions the surface integral is simply a line integral of the form

$$\int_a^b f(x_1, x_2) ds, \tag{4.49}$$

where $ds = \sqrt{(dx_1)^2 + (dx_2)^2}$ is the differential length. The region is naturally broken into elemental regions $\partial\Omega^e \bigcap s$ within which we want to evaluate each

segment of the integral (4.49). We know that the global coordinates x_1, x_2 within each element are related to the local coordinates ξ_1, ξ_2 in terms of the mapping, that is,

$$x_1 = \chi_1(\xi_1, \xi_2), \qquad x_2 = \chi_2(\xi_1, \xi_2).$$

We can therefore relate the differential change in x_1 and x_2 in terms of a differential change in ξ_1 and ξ_2 using the chain rule:

$$dx_1 = \frac{\partial x_1}{\partial \xi_1} d\xi_1 + \frac{\partial x_1}{\partial \xi_2} d\xi_2,$$
$$dx_2 = \frac{\partial x_2}{\partial \xi_1} d\xi_1 + \frac{\partial x_2}{\partial \xi_2} d\xi_2,$$

where the partial derivatives may be evaluated as shown in section 4.1.2. Along the boundary of any element the edge is completely parameterised by either ξ_1 or ξ_2 as the other local coordinate is a constant having a value of 1 or -1. For example, in figure 4.22 the edge touching the domain boundary in element 'e' is parameterised by ξ_1 since $\xi_2 = -1$. For this case we can relate the differential length ds in terms of a differential change $d\xi_1$ as

$$ds = \sqrt{(dx_1)^2 + (dx_2)^2} = \sqrt{\left(\left. \frac{\partial x_1}{\partial \xi_1} \right|_{\xi_2 = -1} \right)^2 (d\xi_1)^2 + \left(\left. \frac{\partial x_2}{\partial \xi_1} \right|_{\xi_2 = -1} \right)^2 (d\xi_1)^2},$$

where the partial derivatives $\frac{\partial x_1}{\partial \xi_1}, \frac{\partial x_2}{\partial \xi_1}$ are evaluated at $\xi_2 = -1$. The contribution of element "e" to the integral in equation (4.49) can now be written as

$$\int_{\partial \Omega^e \bigcap s} f(x_1, x_2) ds = \int_{-1}^{1} f(\xi_1, \xi_2) \sqrt{\left(\tfrac{\partial x_1}{\partial \xi_1} \right)^2 + \left(\tfrac{\partial x_2}{\partial \xi_1} \right)^2} \, d\xi_1,$$

which can be evaluated using standard Gaussian quadrature. The whole surface is then generated as a summation of elemental contributions although in general the integrand contains the test function which only has non-zero support in, at most, two-elements.

4.1.4.2 *Surface Jacobian of a Three-Dimensional Region*

In three-dimensions we want to evaluate a two-dimensional surface integral of the form

$$\int_{\partial \Omega} f(x_1, x_2, x_3) dS. \tag{4.50}$$

Just as a one-dimensional surface can be completely parameterised in terms of a single parameter, a two-dimensional surface can be described by a doubly infinite set of parametric curves.

As indicated in figure 4.5 we have already parameterised the surface in terms of two of the local Cartesian coordinates (ξ_1, ξ_2, ξ_3) depending on which face we are considering. If we consider the face $\xi_3 = -1$, as shown in figure 4.5, then

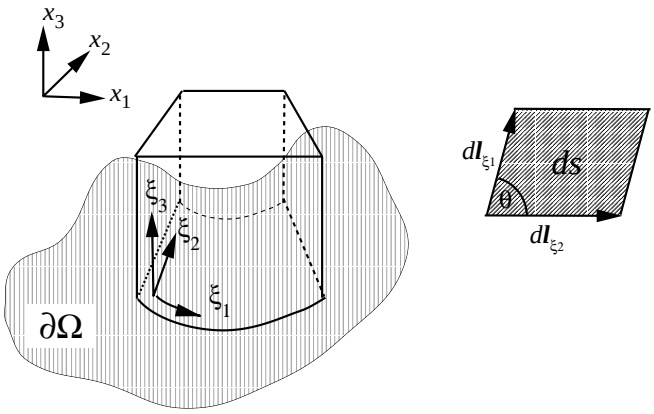


Figure 4.5 The surface of the solution domain $\partial\Omega$ is represented as a tessellation of rectangular (or triangular) partitions within which we have a double parameterisation (that is, ξ_1, ξ_2). We can use this parameterisation to relate the differential surface area dS to the differential change in ξ_1 and ξ_2 via the surface tangent vectors $d\boldsymbol{l}|_{\xi_1}$ and $d\boldsymbol{l}|_{\xi_1}$.

we know that our surface can be completely described by the two parameters ξ_1 and ξ_2, that is,

$$x_1 = \chi_1(\xi_1, \xi_2, -1), \qquad x_2 = \chi_2(\xi_1, \xi_2, -1), \qquad x_3 = \chi_3(\xi_1, \xi_2, -1).$$

To relate the differential surface area dS in terms of the parametric coordinates ξ_1, ξ_2, we note that the change in surface position $d\boldsymbol{x} = [dx_1, dx_2 dx_3]^T$ due to a change in the parametric coordinates is given by

$$d\boldsymbol{x} = \frac{\partial \boldsymbol{x}}{\partial \xi_1} d\xi_1 + \frac{\partial \boldsymbol{x}}{\partial \xi_2} d\xi_2,$$

where $\frac{\partial \boldsymbol{x}}{\partial \xi_1}, \frac{\partial \boldsymbol{x}}{\partial \xi_2}$ are the surface tangent vectors along lines of constant ξ_2, ξ_1, respectively. Therefore, along a line of constant ξ_2 the change in differential length $d\boldsymbol{l}|_{\xi_2}$ due to a differential change in ξ_1 is

$$d\boldsymbol{l}_{\xi_2} = \left.\frac{\partial \boldsymbol{x}}{\partial \xi_1}\right|_{\xi_2} d\xi_1 = \begin{bmatrix} \dfrac{\partial x_1}{\partial \xi_1} \\ \dfrac{\partial x_2}{\partial \xi_1} \\ \dfrac{\partial x_3}{\partial \xi_1} \end{bmatrix} d\xi_1.$$

Similarly, along a line of constant ξ_1 the change in differential length $d\boldsymbol{l}|_{\xi_1}$ for a differential change in ξ_2 is

$$d\boldsymbol{l}_{\xi_1} = \left.\frac{\partial \boldsymbol{x}}{\partial \xi_2}\right|_{\xi_1} d\xi_2 = \begin{bmatrix} \dfrac{\partial x_1}{\partial \xi_2} \\ \dfrac{\partial x_2}{\partial \xi_2} \\ \dfrac{\partial x_3}{\partial \xi_2} \end{bmatrix} d\xi_2.$$

We note that $d\boldsymbol{l}_{\xi_2}$ and $d\boldsymbol{l}|_{\xi_1}$ only involve derivatives of the coordinates with respect to the local coordinates (ξ_1, ξ_2) and so they can be evaluated as explained in section 4.1.2. In general $d\boldsymbol{l}_{\xi_2}$ and $d\boldsymbol{l}|_{\xi_1}$ are not orthogonal and as shown in figure 4.5, the differential surface area dS is given by a parallelogram

$$ds = |d\boldsymbol{l}_{\xi_2}||d\boldsymbol{l}_{\xi_1}| \sin\theta,$$

where θ is the angle between the vectors $d\boldsymbol{l}_{\xi_2}$ $d\boldsymbol{l}_{\xi_1}$. The magnitudes $|d\boldsymbol{l}_{\xi_2}|$ and $|d\boldsymbol{l}_{\xi_1}|$ are also known as the *Lame parameters*.

This expression may also be written as a cross-product in the form:

$$dS = |d\boldsymbol{l}_{\xi_2} \times d\boldsymbol{l}_{\xi_1}| = \left|\frac{\partial \boldsymbol{x}}{\partial \xi_1} \times \frac{\partial \boldsymbol{x}}{\partial \xi_2}\right| d\xi_1 d\xi_2.$$

Finally, we are in a position to evaluate the surface integral given in equation (4.50) over the elemental face as

$$\int_{\partial\Omega^e \bigcap \partial\Omega} f(x_1, x_2, x_3) dS = \int_{-1}^{1}\int_{-1}^{1} f(\xi_1, \xi_2, -1) \left|\tfrac{\partial \boldsymbol{x}}{\partial \xi_1} \times \tfrac{\partial \boldsymbol{x}}{\partial \xi_2}\right| d\xi_1 d\xi_2,$$

which can be evaluated using Gaussian quadrature.

Clearly, for different elemental faces we need to change the parametric coordinates. The shape of the surface region is not dependent on the derivation and therefore the same argument can be applied to a triangular surface although the integral is usually evaluated using the local collapsed Cartesian coordinates η_1, η_2. This type of construction more commonly arises in structural mechanics for thin shell theory and further discussion may be found in Akin [7].

4.1.5 Elemental Projections and Transformations

As discussed in chapter 3 there are a variety of choices of expansion bases, $\phi_m(\boldsymbol{\xi})$, which correspond to different shaped regions as well as different constructions such as modal and nodal or tensorial and non-tensorial expansions. In general, we assume the notation

$$u^\delta(\boldsymbol{x}) = \sum_{m=0}^{N_m-1} \hat{u}_m \ \phi_m(\boldsymbol{\xi}) = \sum_{m(pqr)=0}^{N_m-1} \hat{u}_{pqr}\phi_{pqr}(\boldsymbol{\xi}), \qquad (\boldsymbol{x}) \in \Omega^e, \qquad (4.51)$$

where $\boldsymbol{x} = [x_1, x_2, x_3], \boldsymbol{\xi} = [\xi_1, \xi_2, \xi_3]$ and $m(pqr)$ represents the connection between the tensorial indices p, q, r and an global index m which will be discussed

in more detail below. We also recall that the local, $\boldsymbol{\xi}$, and global, $\boldsymbol{x}$, coordinates are related through the mappings

$$\xi_1 = (\chi_1^e)^{-1}(x_1, x_2, x_3); \quad \xi_2 = (\chi_2^e)^{-1}(x_1, x_2, x_3); \quad \xi_3 = (\chi_3^e)^{-1}(x_1, x_2, x_3).$$

By analogy to the Fourier transform, we will refer to $u^\delta(\boldsymbol{x})$ as representing a *physical* space variable and $\hat{u}_m, \hat{u}_{pqr}$ as being in *transformed* space.

In this section we discuss how to obtain the transformed coefficients $\hat{u}_{pqr}$ from the physical representation, $u^\delta(\boldsymbol{x})$, using collocation and Galerkin projections and thereby define the *forward transformation*. Although we shall later discuss it in more detail we note that equation (4.51) represents the *backward transformation* from the coefficient to the physical values.

4.1.5.1 *Elemental Matrix and Vector Notation*

The formulation in multiple dimensions and using different types of expansion bases necessitates the introduction of a large number of super- and subscript notations. To help clarify the operations to be discussed in this section as well as those in section 4.2 we, therefore, introduce a matrix and vector notation to represent operations such as integration and differentiation. This notation is helpful in illustrating the construction of discrete operations such as the transformation from physical to transformed space. It is also useful in providing an insight into the fundamental operations when implementing a numerical scheme. We note, however, that explicit construction of these matrices is not always necessary or sometimes even desirable in practice. Many of the matrices are diagonal or very sparse and therefore explicit construction is unnecessarily costly from the computational point of view. Where the expansion bases is based upon a tensor or generalised tensor product construction we can apply the sum-factorisation technique discussed in section 4.1.6. When using higher order polynomial expansions (i.e., $P > 6$) this technique requires far less memory storage and a far lower operation count than the equivalent operations using the full matrices. Nevertheless, as we shall highlight in our discussion, when using a non-tensorial basis it may be necessary to explicitly construct and store some matrices.

Finally, we note that all definitions in this section refer to a single element. In section 4.2 we will extend the notation to include multiple elements. To distinguish between the two uses we will introduce a superscript 'e'.

Vectors: $\boldsymbol{u}, \hat{\boldsymbol{u}}$

Typically, we require the value of a function at a nodal set of points $\boldsymbol{\xi}_m$, $0 \leq m < N$ and so we define the vector $\boldsymbol{u}$ to denote the evaluation of $u(LCccord)$ at these points, i.e.

$$\boldsymbol{u}[m] = u(\boldsymbol{\xi}_m),$$

where we have dropped the δ superscript for notational convenience.

When using a tensor based expansion we typically require the function to be evaluated at the quadrature points. In this case we can take our set of nodes

as the set of one-dimensional quadrature points, for example in the hexahedral region we can say $\boldsymbol{\xi}_{m(i,j,k)} = [\xi_{1i}, \xi_{2j}, \xi_{3k}]$ where i, j and k are the indices over the one-dimensional quadrature nodes. If we have Q_1, Q_2, Q_3 quadrature points in each direction then the vector $\boldsymbol{u}$ becomes

$$\boldsymbol{u}[m(ijk)] = u(\xi_{1i}, \xi_{2j}, \xi_{3k}),$$

where

$$m(ijk) = i + j \cdot Q_1 + k \cdot Q_1 \cdot Q_2.$$

In the above, array $m(ijk)$ is an integer value which runs consecutively from 0 to $N_Q = (Q_1 \cdot Q_2 \cdot Q_3)$ as the indices i, j, k run through the ranges $(0 \leq i \leq Q_1 - 1, 0 \leq j \leq Q_2 - 1, 0 \leq k \leq Q_3 - 1)$. By convention we let the ξ_1 coordinate run fastest followed by the ξ_2 and finally the ξ_3 coordinate. Written explicitly the vector $\boldsymbol{u}$ is

$$\boldsymbol{u} = [u(\xi_{10}, \xi_{20}, \xi_{30}), \cdots u(\xi_{1Q_1}, \xi_{20}, \xi_{30}), u(\xi_{10}, \xi_{21}, \xi_{30}) \cdots u(\xi_{1Q_1}, \xi_{2Q_2}, \xi_{3Q_3})]^T.$$

The two-dimensional case is analogously defined with $k = 0$. In the case of a non-tensorial basis the index m simply lists the set of discrete nodal points $\boldsymbol{\xi}_m$.

To represent the expansion coefficient $\hat{u}_m$ in vector form we adopt a similar notation but with a circumflex, (that is, $\hat{\boldsymbol{u}}$) such that

$$\hat{\boldsymbol{u}}[m] = \hat{u}_m.$$

The number of the expansion coefficients is clearly related to the number of expansion modes which has previously been denoted as N_m. For a non-tensorial nodal expansion m would also correspond to the expansion coefficient of the Lagrange polynomial with unit value at $\boldsymbol{\xi}_m$.

For tensor based expansions the storage convention for the vector $\hat{\boldsymbol{u}}$ depends on the local indices of the one-dimensional functions usually denoted as p, q, r. Therefore for the orthogonal tensorial expansions we can define the "packing process" which relates $m(p, q, r)$ to p, q, r. For example, an orthogonal expansion in the hexahedral region of polynomial order P_1, P_2, P_3, $\hat{\boldsymbol{u}}$ is defined as

$$\hat{\boldsymbol{u}}[m(pqr)] = \hat{u}_{pqr},$$

where

$$m(pqr) = r + q(P_2 + 1) + p(P_2 + 1)(P_3 + 1)$$

or equivalently

$$\hat{\boldsymbol{u}} = [\hat{u}_{000}, \hat{u}_{001} \cdots \hat{u}_{00P_3}, \hat{u}_{010} \cdots \hat{u}_{01P_3}, \cdots \hat{u}_{020}, \cdots \hat{u}_{P_1P_2P_3}]^T.$$

We note that the skipping factors are $(P + 1)$ as it requires $(P + 1)$ modes to represent a polynomial of order P. By convention we let the index r run fastest followed by q and finally p. Although this convection would appear to be the

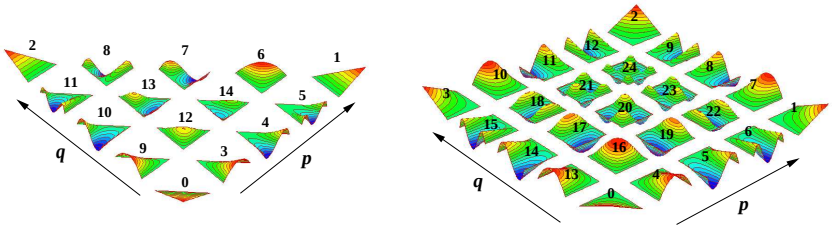


Figure 4.6 Ordering of the expansion coefficients for the modified expansion in a triangular and quadrilateral region with $P_1 = P_2 = 4$. In this ordering we label the vertices first, followed by the edges, and finally the interior. For the interior modes the q index is allowed to run fastest.

reverse of the nodal point definition, the ordering is necessary to take advantage of the partial orthogonality of the generalised tensor product expansions as illustrated in figure 3.21.

The orthogonal expansion within a triangular region, $\phi_{pq} = \phi^a_p(\eta_1)\phi^b_{pq}(\eta_2)$, of order P_1, P_2 defined in section 3.2.2 would have an index system defined by

$$m(pq) = q + \frac{p(2P_2 + 1 - p)}{2}$$

where the index range of p, q is

$$0 \leq p, q;\ p \leq P_1;\ p + q \leq P_2;\ P_1 \leq P_2.$$

For the orthogonal, generalised tensor product expansions, it is possible to define $m(pqr)$ analytically since the indices are *close packed*. However, for the modified generalised tensor product expansion bases defined in section 3.2.3 there is not a *close packed* form for $m(pqr)$ due to the way we have defined the expansion modes. This is also true for the serendipity and variable expansion in quadrilateral and hexahedral regions. In general, we interpret the form of $m(pqr)$ as reordering the N_m modes of an elemental expansion into a consecutive order. A typical storage scheme might place the indices of the vertex modes first, followed by the edge, face and finally the interior modes where we shall still allow r to run fastest, followed by q then p within each group of modes. An example of this ordering for quadrilateral and triangular expansions of order $P_1, P_2 = 4$ is shown in figure 4.6.

Weight and Basis Matrices: $\boldsymbol{W}, \boldsymbol{B}$

A matrix of dimension $m + 1, n + 1$ is understood to be a rectangular array of the form

$$\boldsymbol{A} = \begin{bmatrix} a_{00} & a_{01} & \cdots & a_{0n} \\ a_{10} & a_{11} & \cdots & a_{0n} \\ \cdot & \cdot & \cdots & \cdot \\ a_{m0} & a_{m1} & \cdots & a_{mn} \end{bmatrix}.$$

To complement the vectors $\boldsymbol{u}$ and $\hat{\boldsymbol{u}}$ we introduce two matrices $\boldsymbol{W}$ and $\boldsymbol{B}$. $\boldsymbol{W}$, the weight matrix, is a diagonal matrix containing the Gaussian quadrature weights multiplied by the Jacobian at the quadrature points and is defined to be consistent with $\boldsymbol{u}$ evaluated at the set of quadrature points. This matrix is not defined for an arbitrary set of nodal points. If J_{ijk} represents the discrete value of the Jacobian $J(\xi_{1i}, \xi_{2j}, \xi_{3k})$ then $\boldsymbol{W}$ in three dimensions is defined as

$$\boldsymbol{W}[m(ijk)][n(rst)] = J_{ijk}\ w_i\ w_j\ w_k\ \delta_{mn}$$

where δ_{mn} is the Dirac delta and $m(ijk), n(rst)$ are defined in a consistent fashion to $\boldsymbol{u}$ so

$$m(ijk) = n(ijk) = i + j \cdot Q_1 + k \cdot Q_1 \cdot Q_2.$$

The quadrature weights are defined according to the elemental region so for a hexahedral region $w_i = w_i^{0,0}, w_j = w_j^{0,0}, w_k = w_k^{0,0}$ whereas for a tetrahedral region $w_i = w_i^{0,0}, w_j = \hat{w}_j^{1,0}, w_k = \hat{w}_k^{2,0}$. The explicit form of the weight matrix in a tetrahedral region is therefore

$$\boldsymbol{W} = \begin{bmatrix} w_0^{0,0}\hat{w}_0^{1,0}\hat{w}_0^{2,0}J_{ijk} & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & w_{Q_1}^{0,0}\hat{w}_0^{1,0}\hat{w}_0^{2,0}J_{ijk} & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & w_{Q_1-1}^{0,0}\hat{w}_{Q_2-1}^{1,0}\hat{w}_{Q_3-1}^{2,0}J_{ijk} \end{bmatrix}.$$

The basis matrix, $\boldsymbol{B}$, is a type of Vandemonde matrix as introduced in section 3.3.2. The matrix $\boldsymbol{B}$ is therefore defined as having columns which are a fixed expansion modes $\phi_n(\boldsymbol{\xi})$ evaluated at all the nodal points $\boldsymbol{\xi}_m$, that is,

$$\boldsymbol{B}[m][n] = \phi_n(\boldsymbol{\xi}_m).$$

Using the previously defined indexing for the tensorial expansions we can define the basis matrix for a tensor expansion, $\phi_{m(pqr)} = \phi_{pqr}$ evaluated at the quadrature points $\boldsymbol{\xi}_{n(ijk)} = [\xi_{1i}, \xi_{2j}, \xi_{3k}]$ as

$$\boldsymbol{B}[m(ijk)][n(pqr)] = \phi_{pqr}(\xi_{1i}, \xi_{2j}, \xi_{3k}).$$

The matrix is formulated so that its columns [looping over $m(ijk)$] are ordered in a consistent fashion to $\boldsymbol{u}$ and its rows [looping over $n(pqr)$] are ordered in a consistent fashion to $\hat{\boldsymbol{u}}$, that is,

$$\boldsymbol{B} = \begin{bmatrix} \phi_{000}(\xi_{10}, \xi_{20}, \xi_{30}) & \cdot & \phi_{00P_3}(\xi_{10}, \xi_{20}, \xi_{30}) & \cdot & \phi_{pqr}(\xi_{10}, \xi_{20}, \xi_{30}) \\ \vdots & . & \vdots & . & \vdots \\ \phi_{000}(\xi_{1Q_1}, \xi_{20}, \xi_{30}) & \cdot & \phi_{00P_3}(\xi_{1Q_1}, \xi_{20}, \xi_{30}) & \cdot & \phi_{pqr}(\xi_{1Q_1}, \xi_{20}, \xi_{30}) \\ \vdots & . & \vdots & . & \vdots \\ \phi_{000}(\xi_{1Q_1}, \xi_{2Q_2}, \xi_{3Q_3}) & \cdot & \phi_{00P_3}(\xi_{1Q_1}, \xi_{2Q_2}, \xi_{3Q_3}) & \cdot & \phi_{pqr}(\xi_{1Q_1}, \xi_{2Q_2}, \xi_{3Q_3}) \end{bmatrix}.$$

For the *non-tensorial nodal basis* in simplex regions discussed in section 3.3 there is not a close form expression to evaluate the multi-dimensional Lagrange polynomial $L_n(\boldsymbol{\xi})$ at an arbitrary points, $\boldsymbol{\xi}_m$. It is therefore not immediately obvious how to construct the basis matrix $\boldsymbol{B}$ in this case. Following the formulation of Warburton *et al.* [488] we recall from section 3.3.2 that the triangular (or tetrahedral) Lagrange polynomial can be expressed in terms of another basis $\phi_{pq}(\boldsymbol{\xi})$ (such as the orthogonal tensorial expansion in a simplex) using the transformation

$$\begin{bmatrix} L_0(\boldsymbol{\xi}) \\ \vdots \\ L_{N_m-1}(\boldsymbol{\xi}) \end{bmatrix} p = \boldsymbol{B}_{\mathcal{N}}^{-T} \begin{bmatrix} \phi_0(\boldsymbol{\xi}) \\ \vdots \\ \phi_{N_m-1}(\boldsymbol{\xi}) \end{bmatrix} \tag{4.52}$$

where

$$\boldsymbol{B}_{\mathcal{N}}[n'][n(pq)] = \phi_{pq}(\boldsymbol{\xi}_{n'})$$

and $\xi_{n'}$ $(0 \leq n' < N_m)$ are the set of nodes which define the Lagrange polynomial $L_n(\xi)$. Note that we have used $\boldsymbol{B}_{\mathcal{N}}$ whereas we previously used $\boldsymbol{V}$ in equation (3.22) of section 3.3.2. We can now evaluate the Lagrange polynomial basis at an arbitrary set of points by evaluating $\phi_{pq}(\xi)$ at the desired points. Therefore, the matrix $\boldsymbol{B}[m][n] = L_n(\boldsymbol{\xi}_m)$ of the non-tensorial triangular Lagrange basis evaluated at the quadrature points in the triangle, $\boldsymbol{\xi}_{m(ij)} = [\eta_{1i}, \eta_{2j}]$, can be determined as

$$\boldsymbol{B}^T = \boldsymbol{B}_{\mathcal{N}}^{-T} \boldsymbol{B}_{\mathcal{T}}^T$$

or equivalently

$$\boldsymbol{B} = \boldsymbol{B}_{\mathcal{T}} \boldsymbol{B}_{\mathcal{N}}^{-1} \tag{4.53}$$

where

$$\boldsymbol{B}_{\mathcal{T}}[m(ij)][n(pq)] = \phi_{pq}(\boldsymbol{\xi}_{m(ij)}).$$

Using this construction in the following section we will be able exactly to integrate and differentiate the non-tensorial expansions in general shaped elemental domains.

Differentiation matrix: $\boldsymbol{D}_{\xi_i}$

The final matrix we require to complete our set of discrete operators is a matrix representing the action of differentiation. Recall that partial differentiation with respect to the local coordinates ξ_1 defined in physical space can be written as

$$\frac{\partial u}{\partial \xi_1}(\xi_{1i}, \xi_{2j}, \xi_{3k}) = \sum_{r=0}^{Q_1} \sum_{s=0}^{Q_2} \sum_{t=0}^{Q_3} \left. \frac{dh_r(\xi_1)}{d\xi_1} \right|_{\xi_{1i}} h_s(\xi_{2j}) h_t(\xi_{3k})\ u_{rst},$$

where $h_r(\xi)$ is the one-deimesional Lagrange polynomial through the Q_1 quadrature points and $u_{rst} = u(\xi_{1r}, \xi_{2s}, \xi_{3t})$. A similar definition follows for the other

partial derivatives, (see section 4.1.2). We can, therefore, define a derivative matrix $\boldsymbol{D}_{\xi_1}$ which acts on $\boldsymbol{u}$ evaluated at the quadrature nodes such that $\boldsymbol{D}_{\xi_1}\boldsymbol{u}$ is the derivative at the quadrature points $\boldsymbol{u}_{\xi_1} = \frac{\partial \boldsymbol{u}}{\partial \xi_1}$, that is,

$$\frac{\partial \boldsymbol{u}}{\partial \xi_1} = \boldsymbol{D}_{\xi_1}\boldsymbol{u},$$

where

$$\boldsymbol{D}_{\xi_1}[m(ijk)][m'(rst)] = \left.\frac{dh_r(\xi_1)}{d\xi_1}\right|_{\xi_{1i}} h_s(\xi_{2j})h_t(\xi_{3k}) \tag{4.54}$$

and

$$m(ijk) = m'(ijk) = i + j \cdot Q_1 + k \cdot Q_1 \cdot Q_2.$$

The differentiation matrix can clearly be used to evaluate the derivative of the expansion bases, ϕ_n, at the quadrature points, $\xi_{m(i,j,k)}$, if applied to the basis matrix $\boldsymbol{B}$. Therefore to evaluate the derivative with respect to ξ_1 of the basis $\phi_{n(pqr)}$ at the quadrature nodes $\boldsymbol{\xi}_{m(ijk)} = [\xi_{1i}, \xi_{2j}, \xi_{3k}]$ we can write

$$\frac{\partial \boldsymbol{\phi}_n}{\partial \xi_1}(\boldsymbol{\xi}_m) = \left(\boldsymbol{D}_{\xi_1}\boldsymbol{B}\right)[m][n]$$

For the non-tensorial nodal basis expansion $\phi_n = L_n$ the derivative is evaluated in an identical fashion with the basis matrix $\boldsymbol{B}$ evaluated using equation (4.53).

The matrix $\boldsymbol{D}_{\xi_1}$ is very sparse as can be appreciated from the fact that $h_s(\xi_{2j}) = \delta_{sj}$ and $h_t(\xi_{3k}) = \delta_{tk}$. This is particularly evident in the case of $\boldsymbol{D}_{\xi_1}$ because of the way we have chosen to order the quadrature points. Since the ξ_1 coordinate index runs fastest we find that $\boldsymbol{D}_{\xi_1}$ is a block diagonal matrix made from the one-dimensional derivative matrix, that is,

$$\boldsymbol{D}_{\xi_1} = \begin{bmatrix} \boldsymbol{D}^{1d} & 0 & 0 & 0 \\ 0 & \boldsymbol{D}^{1d} & 0 & 0 \\ & & \ddots & \\ 0 & 0 & 0 & \boldsymbol{D}^{1d} \end{bmatrix},$$

where

$$\boldsymbol{D}^{1d}[i][r] = \left.\frac{dh_r(\xi_1)}{d\xi_1}\right|_{\xi_{1i}}, \qquad (0 \le i, r \le Q_1).$$

This illustrates the potential inefficiency of forming the matrix $\boldsymbol{D}_{\xi_1}$ since the matrix vector product $\boldsymbol{D}_{\xi_1 u}$ is an $\mathcal{O}\left((Q_1 \cdot Q_2 \cdot Q_3)^2\right)$ operation. However, if we perform the equivalent operation using the one-dimensional matrix $\boldsymbol{D}^{1d}$ along all lines of constant ξ_2, ξ_3 it is an $\mathcal{O}\left((Q_1)^2 \cdot Q_2 \cdot Q_3\right)$ operation. This reduction in cost is a direct consequence of the tensor construction of the basis and can also be considered as a sum factorisation operation as discussed in section 4.1.6.

Diagonal Coefficient Matrices: $\boldsymbol{\Lambda}(c)$

We have seen previously that the derivative with respect to the global coordinates x_1, x_2, and x_3 can be obtained via the chain rule from the derivative with respect to the local coordinates [see equation (4.47)]. The equivalent matrix operation requires us to premultiply the derivative matrices by a diagonal matrix containing factors such as $\frac{\partial \xi_1}{\partial x_1}, \frac{\partial \xi_1}{\partial x_2}, \frac{\partial \xi_1}{\partial x_2}, \ldots$ evaluated at the quadrature points. To denote these diagonal coefficient matrices we introduce the notation $\mathbf{\Lambda}\left(f(\xi_1, \xi_2, \xi_3)\right)$ to be a diagonal matrix whose diagonal components are the evaluation of the function $f(\xi_1, \xi_2, \xi_3)$ at the quadrature points, that is,

$$\mathbf{\Lambda}(f(\xi_1, \xi_2, \xi_3))[m(ijk)][n(rst)] = f(\xi_{1i}, \xi_{2j}, \xi_{3k})\delta_{mn}, \tag{4.55}$$

where

$$m(ijk) = n(ijk) = i + j \cdot Q_1 + k \cdot Q_1 \cdot Q_2.$$

Therefore, since we know that

$$\frac{\partial u}{\partial x_1} = \frac{\partial \xi_1}{\partial x_1}\frac{\partial u}{\partial \xi_1} + \frac{\partial \xi_2}{\partial x_1}\frac{\partial u}{\partial \xi_2} + \frac{\partial \xi_3}{\partial x_1}\frac{\partial u}{\partial \xi_3}$$

we can evaluate $\boldsymbol{u}_{x_1} = \frac{\partial \boldsymbol{u}}{\partial x_1}$ at the quadrature nodes using the notation

$$\boldsymbol{u}_{x_1} = \left[\mathbf{\Lambda}(\tfrac{\partial \xi_1}{\partial x_1})\boldsymbol{D}_{\xi_1} + \mathbf{\Lambda}(\tfrac{\partial \xi_2}{\partial x_1})\boldsymbol{D}_{\xi_2} + \mathbf{\Lambda}(\tfrac{\partial \xi_3}{\partial x_1})\boldsymbol{D}_{\xi_3}\right]\boldsymbol{u}.$$

The introduction of the diagonal matrix notation also allows us to represent the derivatives with respect to the local collapsed coordinate systems. For example, within a triangular region we can express the derivative with respect to the local Cartesian coordinate, ξ_1, ξ_2 in terms of the local collapsed coordinates η_1, η_2 as

$$\begin{aligned}\boldsymbol{D}_{\xi_1} &= \mathbf{\Lambda}(\tfrac{2}{1-\eta_1})\boldsymbol{D}_{\eta_2} \\ \boldsymbol{D}_{\xi_2} &= \mathbf{\Lambda}(\tfrac{1+\eta_1}{1-\eta_2})\boldsymbol{D}_{\eta_1} + \boldsymbol{D}_{\eta_2}.\end{aligned}$$

4.1.5.2 *Backward Transformation*

The backward transformation from coefficient space $\hat{u}_{pqr}$ to physical space $u(\xi_1, \xi_2, \xi_3)$ is defined in equation (4.51) and simply involves the summation of the coefficients multiplied by the modes. We shall primarily be concerned with the discrete backward transform where the function $u(\xi_1, \xi_2, \xi_3)$ is evaluated at the quadrature points

$$u^{\delta}(\xi_{1i}, \xi_{2j}, \xi_{3k}) = \sum_{m(pqr)} \hat{u}_{pqr}\ \phi_{pqr}(\xi_{1i}, \xi_{2j}, \xi_{3k}).$$

This operation can be represented in terms of the vector $\boldsymbol{u}$, $\hat{\boldsymbol{u}}$ and the matrix $\boldsymbol{B}$ as

$$\boldsymbol{u} = \boldsymbol{B}\hat{\boldsymbol{u}}. \tag{4.56}$$

Formulation note: *Matrix formulation of the elemental backward transformation.*

The two-dimensional case is simply given by $k = 0$ and the summation is performed over p, q only. Although we have represented the solution in terms of the

local coordinates (ξ_1, ξ_2, ξ_3) we note that this is equivalent to evaluating the functions at the global coordinates (x_1, x_2, x_3) under the mapping $x_i = \chi_i^e(\xi_1, \xi_2, \xi_3)$. In general, the discrete backward transformation may be evaluated at any set of discrete points depending on the definition of the matrix $\mathbf{B}$. For example, a more evenly distributed set of points may be required graphically to display the solution. For most computational needs however we will normally evaluate the basis at the quadrature points.

Nodal Expansions

We note that when using a nodal expansions basis there are special cases of the backward transform matrix where the matrix becomes the identity matrix. If the matrix $\boldsymbol{B}$ is generated using a Lagrange polynomial through a set of nodal points and the basis is evaluated at the same nodal points then $\boldsymbol{B} = \boldsymbol{I}$. This arises when using either the non-tensorial simplex basis or the quadrilateral and hexahedral nodal expansions. When $\boldsymbol{B} = \boldsymbol{I}$ we observe that

$$\boldsymbol{u} = \boldsymbol{B}\hat{\boldsymbol{u}} = \boldsymbol{I}\hat{\boldsymbol{u}} = \hat{\boldsymbol{u}}.$$

As discussed previously, this demonstrates that, for the classical spectral element method and Lagrange expansion in simplexes, the expansion coefficients are simply the values of the solution at the nodal points. We note, however, that when the basis is evaluated at points other than the nodal points the matrix $\boldsymbol{B}$ is full. Such a situation would arise even in the tensorial quadrilateral/hexahedral nodal basis if the quadrature order is not exactly equivalent to the polynomial order plus one (i.e., $Q = P + 1$).

4.1.5.3 *Elemental Forward Transformation*

In this section we discuss the formulation, using the previously introduced matrix and vector notation, of the forward transformation. The action of the forward transformation is that given either a continuous, $u(\boldsymbol{\xi})$, or discrete $u^\delta(\boldsymbol{\xi})$ function we determine the expansion coefficients $\hat{\boldsymbol{u}}$. There are two commonly applied approaches using either a *collocation* or *Galerkin* projection both of which can be formulated from the method of weighted residual statement which we will first review. A similar construction was outlined in one-dimension in section 2.3.2.1.

If we consider a two-dimensional function $u(\xi_1, \xi_2)$ which does *not* lie within the polynomial space of the expansion basis there will be an approximation error between our approximation $u^\delta = \sum_{p,q} \hat{u}_{pq}\phi_{pq}$ and the function $u(\boldsymbol{\xi})$ which we denote by $R(u)$, that is,

$$u^\delta(\xi_1, \xi_2) - u(\xi_1, \xi_2) = R(u) \tag{4.57}$$

or equivalently

$$\left(\sum_{pq} \hat{u}_{pq}\phi_{pq}(\xi_1, \xi_2)\right) - u(\xi_1, \xi_2) = R(u).$$

Following the method of weighted residuals, we represent the inner product of both sides of this equation by a presently undefined function $v(\xi_1, \xi_2)$

$$(v, \sum_{p,q} \hat{u}_{pq}\phi_{pq}) - (v, u) = (v, R(u)),$$

and set the right-hand-side term to zero, $(v, R(u)) = 0$, to obtain

$$(v, \sum_{p,q} \hat{u}_{pq}\phi_{pq}) = (v, u). \tag{4.58}$$

The choice of $v(\boldsymbol{\xi})$ will define the type of projection this is illustrated in the next two sections for the case of a collocation and Galerkin projection.

Collocation Projection - Interpolation

We note that the matrix $\boldsymbol{B}$ introduced in the previous section is directly analogous to the generalised Vandemonde matrix $\boldsymbol{V}$ introduced in section 3.3.2. In particular, if we consider the case where the set of distinct points $\boldsymbol{\xi}_i$ is of the same dimension as the expansion basis then the matrix $\boldsymbol{B}$ is identical to our previous definition of $\boldsymbol{V}$ in section 3.3.2. In this case $\boldsymbol{B}$ is square and invertible and the inversion of this matrix is equivalent to a collocation projection. To see how this fits into the statement of the method of weighted residuals we write our approximation as

$$u^{\delta}(\boldsymbol{\xi}) = \sum_{n=1}^{N_m-1} \hat{u}_n\phi_n(\boldsymbol{\xi}),$$

then the methods of weighted residual equation (4.58) implies that

$$\int_{\Omega} v_m u^{\delta}(\boldsymbol{\xi})\ d\boldsymbol{\xi} = \int_{\Omega} v_m \sum_{n=1}^{N_m-1} \hat{u}_n\phi_n(\boldsymbol{\xi})\ d\boldsymbol{\xi} \qquad m = 0, \ldots, N_m - 1. \tag{4.59}$$

In the collocation method we set $v_m = \delta(\boldsymbol{\xi}_m)$ where $\delta(\boldsymbol{\xi}_m)$ is the Dirac delta function at the N_m discrete nodal points $\boldsymbol{\xi}_m$ and implies that $R(u(\boldsymbol{\xi}_m)) = 0$. The action of the Dirac delta functions on the integrals means that equation (4.59) can be evaluated as

$$u^{\delta}(\boldsymbol{\xi}_m) = \sum_{n=1}^{N_m-1} \hat{u}_n\phi_n(\boldsymbol{\xi}_m) \qquad n = 0, \ldots, N_m - 1$$

which can be written in matrix form to obtain

$$\boldsymbol{u} = \boldsymbol{B}_{\mathcal{N}}\hat{\boldsymbol{u}} \qquad \text{or} \qquad \hat{\boldsymbol{u}} = \boldsymbol{B}_{\mathcal{N}}^{-1}\boldsymbol{u} \tag{4.60}$$

where $\boldsymbol{B}_{\mathcal{N}}[m][n] = \phi_n(\boldsymbol{\xi}_m)$.

In general, the above formulation can be applied to any function $u(\boldsymbol{\xi})$ not just the approximation $u^{\delta}(\boldsymbol{\xi})$ and thus we can use it as a technique to interpolate a function within the region $\boldsymbol{\xi} \in \Omega_{st}$.

Discrete Galerkin Projection

The discrete forward transform which determines the modal coefficients $\hat{u}_{pq}$ from a prescribed function $u(\boldsymbol{\xi})$ evaluated at a set of nodal or quadrature points, can be expressed in terms of the matrix formulation previously introduced as

$$\hat{\boldsymbol{u}} = \left(\boldsymbol{B}^T \boldsymbol{W} \boldsymbol{B}\right)^{-1} \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{u}, \tag{4.61}$$

where we note that the inverted matrix $\boldsymbol{BWB}$ is symmetric in contrast to the collocation matrix $\boldsymbol{B}_{\mathcal{N}}$. We refer to this as a *discrete* forward transform since if $u(\boldsymbol{\xi})$ is not a polynomial function of similar space as the projecting polynomial then a collocation projection is being performed when evaluating the function at the quadrature points to obtain $\boldsymbol{u}$. We note, however, that the collocation projection may be onto a richer polynomial space than the Galerkin projection and so the error associated with the Galerkin projection typically dominates.

In the Galerkin projection we choose the weight function in the method of weighted residuals (4.58) to be the same as the expansion basis so that $v(\xi_1, \xi_2) = \phi_{rs}(\xi_1 \xi_2)$. Equation (4.58) can therefore be written as

$$(\phi_{rs}, \sum_{p,q} \hat{u}_{pq} \phi_{pq}) = (\phi_{rs}, u).$$

Noting that the coefficients $\hat{u}_{pq}$ are independent of ξ_1, ξ_2 we can rewrite this equation as

$$\sum_{p,q} (\phi_{rs}, \phi_{pq}) \hat{u}_{pq} = (\phi_{rs}, u), \tag{4.62}$$

which is a scalar equation. If we test this equation versus all N_m modes ϕ_{rs} we then have N_m scalar equations to solve for the N_m unknowns degrees of freedom $\hat{u}_{pq}$.

Equation (4.62) is the functional representation of a linear system which can be solved to determine $\hat{u}_{pq}$ where the term (ϕ_{rs}, ϕ_{pq}) represents the components of the two-dimensional elemental mass matrix $\boldsymbol{M}$, which has a rank equal to N_m. An identical procedure to the one outlined above using ϕ_{pqr} would have led to the three-dimensional system.

Although equation (4.62) represents the forward transformation it is not immediately clear how to construct the matrix system. First, we need to make an approximation to represent the inner product discretely by using Gaussian quadrature. The integral in the inner product on the left-hand-side may be evaluated exactly using Gaussian quadrature providing that a sufficient number of quadrature points are used. The right-hand-side inner product in equation (4.62) involves the arbitrary function $u(\xi_1, \xi_2, \xi_3)$ which may not be a polynomial. Nevertheless, providing the function u is sufficiently smooth the error in evaluating

the integral will be consistent with the approximation error. Similarly, by evaluating the continuous function at the collocation points we are performing a collocation projection onto the quadrature points. The discrete form of equation (4.62) can be written

$$\sum_{pq}(\phi_{rs},\phi_{pq})_\delta\ \hat{u}_{pq} = (\phi_{rs},u)_\delta, \qquad \forall r,s. \tag{4.63}$$

This equation represents a system of N_m scalar equations for every ϕ_{rs}.

We can now illustrate the use of the matrix and vector notation by initially considering the inner product of a function $v(x_1,x_2,x_3)$ with a function $u(x_1,x_2,x_3)$ and is defined as

$$(v,u)_\delta = \int v(\xi_1,\xi_2,\xi_3)u(\xi_1,\xi_2,\xi_3)|J|\ d\xi_1\ d\xi_2\ d\xi_3.$$

Representing the integral using Gaussian quadrature we have a discrete approximation such that

$$(v,u)_\delta = \sum_{i=0}^{Q_1-1}\sum_{j=0}^{Q_2-1}\sum_{k=0}^{Q_3} w_iw_jw_k\ v(\xi_{1i},\xi_{2j},\xi_{3k})\ u(\xi_{1i},\xi_{2j},\xi_{3k})\ |J_{ijk}|, \tag{4.64}$$

where

$$(v,u) = (v,u)_\delta + \varepsilon$$

and ε is the error due to the numerical integration or collocation projection, as defined in appendix B. If the functions u and v are sufficiently smooth in the sense that the first Q derivatives are bounded, then ε will be of the same order as the approximation error, which is important if this error is not to dominate [87].

The operation in equation (4.64) can be evaluated using the vectors $\boldsymbol{v}$ and $\boldsymbol{u}$ and the matrix $\boldsymbol{W}$ as

$$(v,u)_\delta = \boldsymbol{v}^T\boldsymbol{W}\boldsymbol{u}. \tag{4.65}$$

Now to assemble equation (4.63) into a matrix system we note that when $v(\boldsymbol{\xi}) = \phi_{rs}(\boldsymbol{\xi})$ in equation (4.65) we have the right-hand-side of equation (4.63) for a single expansion mode. To evaluate the complete right-hand-side of the matrix system we need to evaluate this inner product over all N_m expansion modes to produce a vector of length N_m. The columns of the matrix $\boldsymbol{B}$ represent the expansion modes ϕ_{rs} at the quadrature points and so to evaluate the inner product with respect to all modes we replace $\boldsymbol{v}$ in equation (4.65) with $\boldsymbol{B}$, that is,

$$\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{u}[m(rs)] = (\phi_{rs},u)_\delta,$$

where $m(rs)$ represents the consecutive ordering of the expansion modes whose indices run over r,s. The next step is to express the left-hand-side of equation

(4.63) in a similar fashion. The left-hand-side is the inner product of every expansion mode with respect to every other expansion mode which leads to the elemental mass matrix $\boldsymbol{M}$:

$$\boldsymbol{M}[m(rs)][n(pq)] = \boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B}[m(rs)][n(pq)] = (\phi_{rs}, \phi_{pq})_\delta,$$

where $n(pq)$ is analogous to $m(rs)$ and represents the consecutive ordering of expansion modes. Finally, the summation in equation (4.63) is identical to multiplying the mass matrix $\boldsymbol{M}$ by the coefficient vector $\hat{\boldsymbol{u}}$ to obtain the matrix representation of the system (4.63):

$$\left(\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B}\right)\hat{\boldsymbol{u}} = \boldsymbol{B}^T\boldsymbol{W}\boldsymbol{u}.$$

The solution to this system determines the vector of expansion coefficients $\hat{\boldsymbol{u}}$ from the values of a function at the quadrature points denoted by $\boldsymbol{u}$, that is,

$$\hat{\boldsymbol{u}} = \left(\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B}\right)^{-1}\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{u} = (\boldsymbol{M})^{-1}\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{u}$$

Formulation note: *Matrix formulation of the elemental Galerkin forward transformation.*

which is the *discrete forward transform*. We note that if $\boldsymbol{u}$ spans the same space as the polynomial basis used to evaluate $\boldsymbol{B}$ and integration is exact then the above projection would give an identical answer to equation (4.60) up to numerical precision.

Galerkin Projection with Nodal Expansions

Just as the nodal expansion basis was a special case for the backward transformation, it is also true for the forward transformation. We recall that for the spectral element method the the nodal expansion is defined by the Lagrange polynomials through the quadrature points, implying that

$$\boldsymbol{B} = \boldsymbol{B}^T = \boldsymbol{I}.$$

Therefore, the discrete forward transform becomes

$$\begin{aligned}\hat{\boldsymbol{u}} &= \left(\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B}\right)^{-1}\boldsymbol{B}^T\boldsymbol{W}\boldsymbol{u}\\ &= (\boldsymbol{I}\boldsymbol{W}\boldsymbol{I})^{-1}\boldsymbol{W}\boldsymbol{I}\boldsymbol{u}\\ &= \boldsymbol{W}^{-1}\boldsymbol{W}\boldsymbol{u} = \boldsymbol{u}.\end{aligned}$$

Analogous with the interpretation of the forward transformation we observe that the modal coefficients, $\hat{\boldsymbol{u}}$, are simply the values of the solution at the nodal points, $\boldsymbol{u}$.

Positive-Definiteness of the Elemental Mass Matrix

Working back from the matrix formulation to the functional form of the integral operator it is possible to show that the elemental mass matrix

$$\boldsymbol{M} = \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{B}$$

is positive-definite. A sufficient condition for the matrix $\boldsymbol{M}$ to be positive-definite is that

$$\hat{\boldsymbol{u}}^T \boldsymbol{M} \hat{\boldsymbol{u}} > 0 \qquad \forall \text{ non-zero vectors } \hat{\boldsymbol{u}}.$$

If we replace $\boldsymbol{M}$ by its full matrix components we find

$$\hat{\boldsymbol{u}}^T \boldsymbol{M} \hat{\boldsymbol{u}} = \hat{\boldsymbol{u}}^T \left(\boldsymbol{B}^T \boldsymbol{W} \boldsymbol{B} \right) \hat{\boldsymbol{u}}.$$

Now, from the definition of the backward transformation (4.56), we see that

$$\hat{\boldsymbol{u}}^T \left(\boldsymbol{B}^T \boldsymbol{W} \boldsymbol{B} \right) \hat{\boldsymbol{u}} = (\boldsymbol{B}\hat{\boldsymbol{u}})^T \boldsymbol{W} (\boldsymbol{B}\hat{\boldsymbol{u}}) = \boldsymbol{u}^T \boldsymbol{W} \boldsymbol{u}.$$

A comparison with equation (4.65) shows that the last expression is simply the inner product of $u^\delta(\xi_1, \xi_2)$ with itself, that is,

$$\boldsymbol{u}^T \boldsymbol{W} \boldsymbol{u} = (u^\delta, u^\delta)_\delta.$$

Providing the quadrature order is sufficiently high the integration will be exact since the $u^\delta(\xi_1, \xi_2)$ is a polynomial and, therefore,

$$(u^\delta, u^\delta)_\delta = \int (u^\delta)^2 d\xi_1 d\xi_2 \ \geq \ 0,$$

which is positive for any non-zero value of $u^\delta(\xi_1, \xi_2)$ thereby proving that $\boldsymbol{M}$ is positive-definite.

Discrete Galerkin Projection to Physical Space

We have previously considered the projection of the continuous function $u(\boldsymbol{\xi})$ evaluated at the quadrature points $\boldsymbol{u}$ onto the polynomial space to obtain a set of expansion coefficients $\hat{\boldsymbol{u}}$, for example

$$\hat{\boldsymbol{u}} = (\boldsymbol{M})^{-1} \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{u}.$$

However, if we now want to re-evaluate the projected function at the same quadrature points we can perform a backwards transform or equivalently multiply by the basis matrix $\boldsymbol{B}$ such that

$$\boldsymbol{u}^{\boldsymbol{\delta}} = \boldsymbol{P}^{\boldsymbol{\delta}} \boldsymbol{u} = \boldsymbol{B} (\boldsymbol{M})^{-1} \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{u}.$$

This entire process, denoted by $\boldsymbol{P}^\delta$, can be considered as a discrete Galerkin projection to physical space and has the property that $\boldsymbol{P}^{\boldsymbol{\delta}} \boldsymbol{P}^{\boldsymbol{\delta}} \boldsymbol{u} = \boldsymbol{P}^{\boldsymbol{\delta}} \boldsymbol{u}$, which is easily demonstrated since

$$\begin{aligned} \boldsymbol{P}^{\boldsymbol{\delta}} \boldsymbol{P}^{\boldsymbol{\delta}} \boldsymbol{u} &= \boldsymbol{B}^T (\boldsymbol{M})^{-1} \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{B} (\boldsymbol{M})^{-1} \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{u} \\ &= \boldsymbol{B}^T (\boldsymbol{M})^{-1} \boldsymbol{M} (\boldsymbol{M})^{-1} \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{u} \\ &= \boldsymbol{B}^T (\boldsymbol{M})^{-1} \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{u} \\ &= \boldsymbol{P}^{\boldsymbol{\delta}} \boldsymbol{u}. \end{aligned}$$

4.1.5.4 *Differential Operators: Weak Laplacian*

To complete this section we illustrate how to construct a matrix system from a differential problem and thereby construct the weak elemental Laplacian matrix.

We wish to consider the Galerkin approximation of the two-dimensional Poisson equation within an elemental region:

$$\nabla^2 u(\boldsymbol{x}) = f(\boldsymbol{x}), \qquad (\boldsymbol{x}) \in \Omega^e.$$

The one-dimensional formulation of this equation has been dealt with in section 2.2.1 and a complete multi-dimensional formulation can be found in chapter 5. To recap the Galerkin approximation of this equation, we take the inner product with respect to a continuous function $v(\boldsymbol{x})$ to obtain

$$(v, \nabla^2 u) = (v, f).$$

Applying the divergence theorem to the left-hand-side we obtain

$$(\nabla v, \nabla u) = \int_{\partial\Omega} v \nabla u \cdot \mathbf{n} - (v, f)$$

where $\partial\Omega$ is the boundary of the problem and $\mathbf{n}$ is the unit normal along the boundary. The term $(\nabla v, \nabla u)$ is the weak Laplacian and written in full in two dimensions has the form

$$(\nabla v, \nabla u) = \left(\frac{\partial v}{\partial x_1}, \frac{\partial u}{\partial x_1}\right) + \left(\frac{\partial v}{\partial x_2}, \frac{\partial u}{\partial x_2}\right).$$

In a Galerkin formulation the same functions are used to approximate $v(\boldsymbol{x})$ and $u(\boldsymbol{x})$. Approximating all integrals with Gaussian quadrature the matrix form of the elemental weak Laplacian operator $\boldsymbol{L}^e$ becomes

$$\begin{aligned}\boldsymbol{L}^e = &\left[\left(\boldsymbol{\Lambda}(\tfrac{\partial \xi_1}{\partial x_1})\boldsymbol{D}_{\xi_1} + \boldsymbol{\Lambda}(\tfrac{\partial \xi_2}{\partial x_1})\boldsymbol{D}_{\xi_2}\right)\boldsymbol{B}\right]^T \boldsymbol{W}\left(\boldsymbol{\Lambda}(\tfrac{\partial \xi_1}{\partial x_1})\boldsymbol{D}_{\xi_1} + \boldsymbol{\Lambda}(\tfrac{\partial \xi_2}{\partial x_1})\boldsymbol{D}_{\xi_2}\right)\boldsymbol{B} \\ &+ \left[\left(\boldsymbol{\Lambda}(\tfrac{\partial \xi_1}{\partial x_2})\boldsymbol{D}_{\xi_1} + \boldsymbol{\Lambda}(\tfrac{\partial \xi_2}{\partial x_2})\boldsymbol{D}_{\xi_2}\right)\boldsymbol{B}\right]^T \boldsymbol{W}\left(\boldsymbol{\Lambda}(\tfrac{\partial \xi_1}{\partial x_2})\boldsymbol{D}_{\xi_1} + \boldsymbol{\Lambda}(\tfrac{\partial \xi_2}{\partial x_2})\boldsymbol{D}_{\xi_2}\right)\boldsymbol{B},\end{aligned}$$

which can be rearranged into the form

$$\begin{aligned}\boldsymbol{L}^e = &\boldsymbol{B}^T\left(\boldsymbol{D}_{\xi_1}^T\boldsymbol{\Lambda}(\tfrac{\partial \xi_1}{\partial x_1}) + \boldsymbol{D}_{\xi_2}^T\boldsymbol{\Lambda}(\tfrac{\partial \xi_2}{\partial x_1})\right)\boldsymbol{W}\left(\boldsymbol{\Lambda}(\tfrac{\partial \xi_1}{\partial x_1})\boldsymbol{D}_{\xi_1} + \boldsymbol{\Lambda}(\tfrac{\partial \xi_2}{\partial x_1})\boldsymbol{D}_{\xi_2}\right)\boldsymbol{B} \\ &+\boldsymbol{B}^T\left(\boldsymbol{D}_{\xi_1}^T\boldsymbol{\Lambda}(\tfrac{\partial \xi_1}{\partial x_2}) + \boldsymbol{D}_{\xi_2}^T\boldsymbol{\Lambda}(\tfrac{\partial \xi_2}{\partial x_2})\right)\boldsymbol{W}\left(\boldsymbol{\Lambda}(\tfrac{\partial \xi_1}{\partial x_2})\boldsymbol{D}_{\xi_1} + \boldsymbol{\Lambda}(\tfrac{\partial \xi_2}{\partial x_2})\boldsymbol{D}_{\xi_2}\right)\boldsymbol{B},\end{aligned}$$

which also demonstrates the symmetry of this matrix system.

4.1.6 Sum Factorisation/Tensor Product Operations

The sum-, or tensor-, product factorisation technique was first recognised by Orszag [352] and is considered the key to the efficiency of spectral methods. It is based on the fact that the expansion is a tensor product of one-dimensional

functions, which means that many important numerical operations may be numerically evaluated with a notable reduction in operation count as compared to a non-tensorial expansion.

To demonstrate this technique we consider the evaluation of a summation over r, s of an array f_{rs} with the functions h_{ir} and h_{js} for all indices i, j, that is,

$$U_{ij} = \sum_{r}^{P} \sum_{s}^{P} f_{rs} h_{ir} h_{js}, \qquad \forall i, j. \tag{4.66}$$

If $f_{rs} = f(\xi_{1r}, \xi_{2s})$ and $h_{ir} = h_r(\xi_{1i})$ and $h_{js} = h_s(\xi_{2j})$ then this summation would represent the interpolation, using Lagrange polynomials, from a set of points (ξ_{1r}, ξ_{2s}) to a set of points (ξ_{1i}, ξ_{2j}). If all indices i, j, k, l are assumed to be of $\mathcal{O}(P)$ then the evaluation of this whole operation reduces to an $\mathcal{O}(P^2)$ summation over k, l for each one of the $\mathcal{O}(P^2)$ indices i, j and so the total operation count would be $\mathcal{O}(P^4)$. However, noting that we can factor the h_{ir} term out of the second summation

$$U_{ij} = \sum_{r}^{P} h_{ir} \left(\sum_{s}^{P} f_{rs} h_{js} \right), \qquad \forall\ i, j,$$

we can then evaluate the summation over "s" and replace the terms in brackets by:

$$\bar{f}_{jr} = \sum_{s}^{P} f_{rs} h_{js}.$$

To construct $\bar{f}_{jr}$ is an $\mathcal{O}(P^3)$ operation since we are evaluating an $\mathcal{O}(P)$ summation over "s" for all the $\mathcal{O}(P^2)$ indices jr. The original summation (4.66) can now be written:

$$U_{ij} = \sum_{r}^{P} h_{ir} \bar{f}_{jr}, \qquad \forall\ i, j,$$

which is also an $\mathcal{O}(P^3)$ operation as we are evaluating an $\mathcal{O}(P)$ summation over the index r for all $\mathcal{O}(P^2)$ points i, j. We therefore see that this factorisation has reduced the cost from an $\mathcal{O}(P^4)$ operation to an $\mathcal{O}(P^3)$ operation which is the typical reduction for a two-dimensional summation. In three-dimensions it is possible to reduce an $\mathcal{O}(P^6)$ operation to an $\mathcal{O}(P^4)$ operation.

To illustrate this technique we consider the sum-factorisation applied to the backward transformation and inner product when using both tensor product and generalised tensor product expansions. Another important operation is differentiation. However, we note that in the example above if we let $h_{ir} = \frac{dh_r}{d\xi_1}(\xi_{1i})$ then the summation would have represented the numerical differentiation of the function $f(\xi_1, \xi_2)$ with respect to ξ_1.

4.1.6.1 *Backward Transformation Example*

Recall that the two-dimensional backward transformation evaluated at the quadrature points for a general basis $\phi_{pq}(\xi_1, \xi_2)$ is:

$$u(\xi_{1i}, \xi_{2j}) = \sum_p \sum_q \hat{u}_{pq} \phi_{pq}(\xi_{1i}, \xi_{2j}) \qquad \forall\ (i, j), \tag{4.67}$$

which can be described by the matrix operation

$$\boldsymbol{u} = \boldsymbol{B}\hat{\boldsymbol{u}}, \tag{4.68}$$

where $\boldsymbol{u}$ is a vector of length N_Q and denotes the evaluation of $u(\xi_1, \xi_2)$ at the quadrature points, $\hat{\boldsymbol{u}}$ is a vector of length N_m which contains all the elemental expansion coefficients, and $\boldsymbol{B}$ is a matrix of dimension $N_Q \cdot N_m$ whose columns are constructed from the expansion modes evaluated at the quadrature points. We recall that to evaluate the summation (4.67) at all the quadrature points (ξ_{1i}, ξ_{2j}) or alternatively perform the matrix-vector multiplication would be an $\mathcal{O}(P^4)$ operation. This is because each quadrature point involves a summation over $\mathcal{O}(P^2)$ modes and there are typically $\mathcal{O}(P^2)$ quadrature points. In three dimensions the equivalent operation would be $\mathcal{O}(P^6)$.

Standard Tensorial Expansion

For the quadrilateral region the tensorial expansion basis can be written as $\phi_{pq}(\xi_1, \xi_2) = \psi_p^a(\xi_1)\psi_q^a(\xi_2)$. Putting this definition into equation (4.67) and factoring out the term $\psi_p^a(\xi_{1i})$ we obtain

$$u(\xi_{1i}, \xi_{2j}) = \sum_{p=0}^{P_1} \psi_p^a(\xi_{1i}) \left\{ \sum_{q=0}^{P_2} \hat{u}_{pq} \psi_q^a(\xi_{2j}) \right\}. \tag{4.69}$$

We note that to evaluate $u(\xi_1, \xi_2)$ at an arbitrary point is still an $\mathcal{O}(P^2)$ operation. However, if we wish to evaluate the summation at all the $\mathcal{O}(P^2)$ quadrature points ξ_{1i}, ξ_{2j} we can use two steps:

$$f_p(\xi_{2j}) = \sum_{q=0}^{P_2} \hat{u}_{pq} \psi_q^a(\xi_{2j}) \tag{4.70a}$$

$$u(\xi_{1i}, \xi_{2j}) = \sum_{p=0}^{P_1} \psi_p^a(\xi_{1i}) f_p(\xi_{2j}). \tag{4.70b}$$

Inserting (4.70a) into (4.70b) recovers equation (4.69). In step (4.70a) the array $f_p(\xi_{2j})$ is evaluated by summing the modal coefficients, multiplied by the second part of the tensor expansion $\psi_q^a(\xi_{2j})$ over q at every ξ_{2j} point and for every p index. This operation is equivalent to performing a one-dimensional backward

transform and requires extra storage for an array of size $\mathcal{O}(P^2)$. The summation for every entry in this array is an $\mathcal{O}(P)$ operation and therefore the total cost to generate $f_p(\xi_{2j})$ is an $\mathcal{O}(P^3)$ operation. However, the second step (4.70b) is now independent of the summation in q and so it is also an $\mathcal{O}(P^3)$ operation involving an $\mathcal{O}(P)$ summation at $\mathcal{O}(P^2)$ points ξ_{1i}, ξ_{2j}. In summary, we have replaced the $\mathcal{O}(P^4)$ operation (4.69) with two $\mathcal{O}(P^3)$ operations (4.70a),(4.70b) and an extra array of size $\mathcal{O}(P^2)$.

In three-dimensions the backward transformation for a hexahedral expansion is

$$u(\xi_{1i}, \xi_{2j}, \xi_{3k}) = \sum_{p=0}^{P_1} \psi_p^a(\xi_{1i}) \left\{ \sum_{q=0}^{P_2} \psi_q^a(\xi_{2j}) \left\{ \sum_{r=0}^{P_3} \hat{u}_{pqr} \psi_r^a(\xi_{3k}) \right\} \right\}. \tag{4.71}$$

This summation can be evaluated at all the $\mathcal{O}(P^3)$ quadrature points in an $\mathcal{O}(P^4)$ operation using a three step process of the form:

$$f_{pq}(\xi_{3k}) = \sum_{r=0}^{P_3} \hat{u}_{pqr} \psi_r^a(\xi_{3k}) \tag{4.72a}$$

$$\bar{f}_p(\xi_{2j}, \xi_{3k}) = \sum_{q=0}^{P_2} \psi_q^a(\xi_{2j}) f_{pq}(\xi_{3k}) \tag{4.72b}$$

$$u(\xi_{1i}, \xi_{2j}, \xi_{3k}) = \sum_{p=0}^{P_1} \psi_p^a(\xi_{1i}) \bar{f}_p(\xi_{2j}, \xi_{3k}). \tag{4.72c}$$

In this three-dimensional case, we have replaced the $\mathcal{O}(P^6)$ operation (4.71) with three $\mathcal{O}(P^4)$ operation to evaluate the steps (4.72a), (4.72b) and (4.72c). This also requires memory for two $\mathcal{O}(P^3)$ arrays to store $f_{pq}(\xi_{3k})$ and $\bar{f}_p(\xi_{2j})$.

Generalised Tensorial Expansion

For the hybrid regions where the expansion bases are of the form $\phi_{pqr} = \psi_p^a \psi_{pq}^b \psi_{pqr}^c$ the sum factorisation technique may still be applied in a very similar fashion to the standard tensor product regions. However, whereas for the quadrilateral and hexahedral expansions it did not matter which part of the tensor product we factored out, for the generalised hybrid expansions there is only one choice of factorisation which maintains the efficiency. To illustrate this point, consider the backward transformation of the triangular expansion $\phi_{pq}(\xi_1, \xi_2) = \psi_p^a(\eta_1)\psi_{pq}^b(\eta_2)$ where $\eta_1 = 2(1+\xi_1)/(1-\xi_2), \eta_2 = \xi_2$

$$u(\eta_{1i}, \eta_{2j}) = \sum_p \sum_q \hat{u}_{pq}\ \psi_p^a(\eta_{1i}) \psi_{pq}^b(\eta_{2j}).$$

We are unable to factor out the term $\psi_{pq}^b(\eta_{2j})$ because it is dependent upon both indices p and q and so we can only factor the $\psi_p^a(\eta_{1i})$ term to arrive at

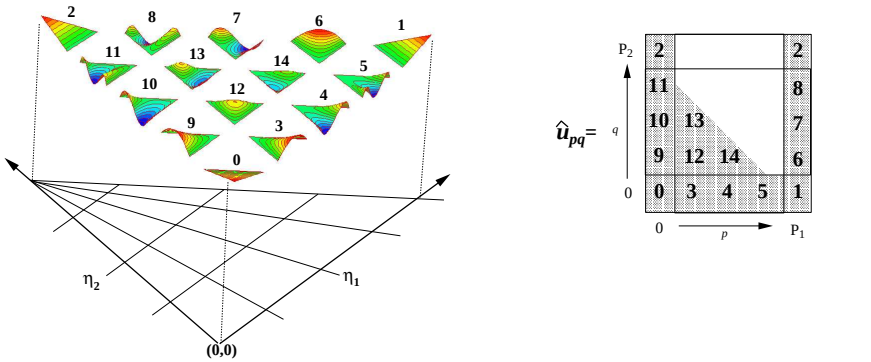


Figure 4.7 Illustration of the mapping procedure for the physical expansion coefficients corresponding to the modes shown in the left plot to the two-dimensional array $\hat{u}_{pq}$ shown in the right plot.

$$u(\eta_{1i}, \eta_{2j}) = \sum_p \psi_p^a(\eta_{1i}) \left\{ \sum_q \hat{u}_{pq}\ \psi_{pq}^b(\eta_{2j}) \right\}, \tag{4.73}$$

which can be evaluated in two steps as:

$$f_p(\xi_{2j}) = \sum_q \hat{u}_{pq}\ \psi_{pq}^b(\xi_{2j}) \tag{4.74a}$$

$$u(\xi_{1i}, \xi_{2j}) = \sum_p \psi_p^a(\xi_{1i}) f_p(\xi_{2j}). \tag{4.74b}$$

Since the expansion coefficients $\hat{u}_{pq}$ depend on both indices pq there is no extra expense in evaluating a tensor product of this form as compared with the structured case. However, we have deliberately omitted any limits on the summation indices over p and q. For the orthogonal triangular expansion the indices are close packed and the range for p, q is $(0 \le p, q; p \le P_1, p + q \le P_2)$, however, for the modified triangular expansion the indices, and therefore $\hat{u}_{pq}$, are not close packed. Nevertheless, it is possible to produce a sparse array of coefficients $\hat{u}_{pq}$ which will allow us to perform the summation (4.74a) over the range $(0 \le p \le P_1; 0 \le q \le P_2)$ or, more efficiently, over every non-zero entry of $\hat{u}_{pq}$.

To illustrate how to generate the non-sparse form of $\hat{u}_{pq}$ we consider the example shown in figure 4.7. This figure shows all $(P + 1)(P + 2)/2$ expansion modes for the modified triangular expansion when $P = P_1 = P_2 = 4$. We recall that each mode is constructed from the product of two one-dimensional functions $\psi_p^a(\eta_1)\psi_{pq}^b(\eta_2)$ (see section 3.2.3).

Implementation note: *Ordering of modified basis expansion coefficient for application of sum factorisation technique.*

In figure 4.7 we also see a numbering system for every physical mode which can be arbitrarily defined. In this case we have adopted the convention that the vertex modes are labelled first followed by edge modes and then the interior modes. The location of the modal coefficient within the array corresponds to the

p, q indices used to generate the principal function mode $\psi^b_{pq}(\xi_2)$ as shown in the right hand illustration in figure 4.7. Therefore the construction of this mapping has a physical interpretation since if we consider the bottom-left corner of the array as being at $(\eta_1 = 0, \eta_2 = 0)$ then this is also the corner where the vertex modes has a unit value. Similarly, the edges along which the modes have a non-zero value relates to their location within the two-dimensional array. Finally, the interior modes are related to the interior of the array where q runs fastest.

The final point to note is that the degenerate vertex mode, labelled vertex "2" in figure 4.7, must be entered twice within the array $\hat{u}_{pq}$ as it is generated by combining the shape of two expansion modes $\phi_{0,P_2}(\eta_1, \eta_2) + \phi_{P_1,P_2}(\eta_1, \eta_2)$. This condition arises from the fact that

$$\phi_{0,P_2}(\xi_1, \xi_2) = \frac{1 + \xi_2}{2}$$

is being represented in terms of the principal functions $\psi^a_p(\eta_1)$ and $\psi^b_{pq}(\eta_2)$ as

$$\phi_{0,P_2}(\xi_1, \xi_2) = \left(\frac{1 - \eta_1}{2} + \frac{1 + \eta_1}{2}\right) \frac{1 + \eta_2}{2} = \left[\psi^a_0(\eta_1) + \psi^a_{P_1}(\eta_1)\right] \psi^b_{0P_2}(\eta_2).$$

The *prismatic* expansion is analogous to the triangular expansion plus a tensor product of the function $\psi^a_r(\xi_3)$ and so has a similar mapping to this example. However, for the *pyramidic* expansion the top vertex is constructed from four components of ϕ_{pqr}. Accordingly, the coefficient of this vertex will need to be mapped to the four locations. Similarly, for the tetrahedral expansion the top vertex relates to four locations where as the base degenerate vertex and the degenerate edge both have double entries in the unpacked array.

4.1.6.2 *Inner Product*

The inner product with respect to all two-dimensional expansion modes requires the evaluation of the summation

$$(\phi_{pq}, u)_\delta = \sum_i \sum_j \phi_{pq}(\xi_{1i}, \xi_{2j})\ w_i w_j\ J(\xi_{1i}, \xi_{2j})\ \hat{u}_{pq}(\xi_{1i}, \xi_{2j}), \qquad \forall\ (p, q), \tag{4.75}$$

where w_i, w_j are the weights in the ξ_1, ξ_2 directions and $J(\xi_{1i}, \xi_{2j})$ is the Jacobian. The backward transform and the inner product (4.75) are closely related and may be considered as the transpose of each other. To appreciate this we can consider operation (4.75) in matrix notation:

$$\boldsymbol{B}^T \left[\boldsymbol{W}\boldsymbol{u}\right], \tag{4.76}$$

where $\boldsymbol{u}$ is the vector of function values at the quadrature points, $\boldsymbol{B}$ is the basis matrix, and $\boldsymbol{W}$ is a diagonal matrix containing the quadrature weights multiplied by the appropriate Jacobian. Since $\boldsymbol{W}$ is a diagonal matrix evaluation of the product $\boldsymbol{W}\boldsymbol{u}$ involves a multiplication at every quadrature point. We can, therefore, consider the bracketed term as a new vector $\boldsymbol{f}$ $[\boldsymbol{f} = (\boldsymbol{W}\boldsymbol{u})]$ and so

the principal operation is the matrix multiplication $\boldsymbol{B}^T\boldsymbol{f}$ which is the transpose operation to the backward transformation (4.68). As with the backward transform, to evaluate (4.75) or the matrix multiplication $\boldsymbol{B}^T\boldsymbol{f}$ would require an $\mathcal{O}(P^4)$ operation in two dimensions and an $\mathcal{O}(P^6)$ operation in three dimensions.

For standard tensorial expansions the application of the sum factorisation technique is analogous to the previous description of the backwards transformation. The generalised tensor product is also similar. The ordering of the factorisation, however, needs to be reversed and so we will consider this case in more detail.

Generalised Tensorial Expansion

When considering a generalised tensorial expansion of the form $\phi_{pqr} = \psi^a_p \psi^b_{pq} \psi^c_{pqr}$ the sum-factorisation process may still be applied although we are again restricted as to which product may be factored. Considering the case of a triangular expansion where $\phi_{pq}(\xi_1, \xi_2) = \psi^a_p(\eta_1)\psi^b_{pq}(\eta_2)$ and $(\eta_1 = 2(1+\xi_1)/(1-\xi_2), \eta_2 = \xi_2)$ the inner product becomes

$$(\phi_{pq}, u)_\delta = \sum_{i=0}^{Q_1-1} \sum_{j=0}^{Q_2-1} \psi^a_p(\eta_{1i})\psi^b_{pq}(\eta_{2j}) w_i w_j J(\eta_{1i}, \eta_{2j}) u(\eta_{1i}, \eta_{2j})$$

where $w_j = w^{1,0}_j/2$ which accounts for the transformation of the coordinates from (ξ_1, ξ_2) to (η_1, η_2). If we factor out the term $\psi^a_p(\eta_{1i})$ then the innermost summation will still involve a sum over i for every η_{2j} points as well as all the modes over p, q. This would involve an $\mathcal{O}(P^4)$ operation and can be as expensive as the unfactored case. However, if we factor out the $\psi^b_{pq}(\eta_{2j})$ term then the summation becomes

$$(\phi_{pq}, u)_\delta = \sum_{j=0}^{Q_2-1} \psi^b_{pq}(\eta_{2j}) \left\{ \sum_{i=0}^{Q_1-1} \psi^a_p(\eta_{1i}) w_i w_j J(\eta_{1i}, \eta_{2j}) u(\eta_{1i}, \eta_{2j}) \right\},$$

which can be evaluated in two steps:

$$f_p(\xi_{2j}) = \sum_{i=0}^{Q_1-1} \psi^a_p(\xi_{1i}) u(\xi_{1i}, \xi_{2j}) w_i w_j J(\xi_{1i}, \xi_{2j}) \tag{4.77a}$$

$$(\phi_{pq}, u)_\delta = \sum_{j=0}^{Q_2-1} \psi^b_{pq}(\xi_{2j}) f_p(\xi_{2j}), \tag{4.77b}$$

where both steps (4.77a),(4.77b) are $\mathcal{O}(P)$ operations requiring $\mathcal{O}(P^2)$ extra memory for the array $f_p(\xi_{2j})$. If we were using an orthogonal expansion the range for p, q would be $(0 \le p, q; p \le P_1, p+q \le P_2)$. For the modified C^0 continuous expansion, however, we need to sum over p, q according to the local sparsity. The

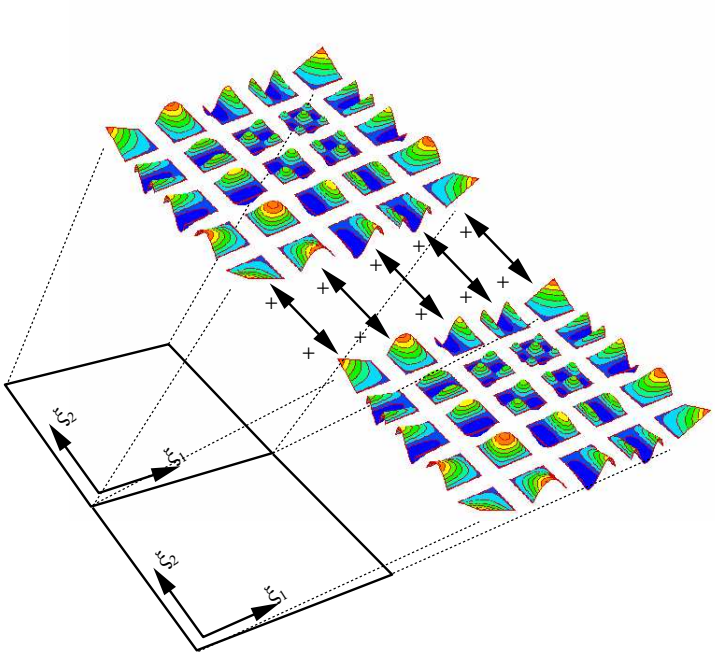


Figure 4.8 Illustration of the construction of a C^0 global expansion from two local modal expansions of order $P_1 = P_2 = 4$. The C^0 continuity condition can be simply ensured by matching the vertex and boundary modes of similar shape.

sparseness of the $\hat{u}_{pq}$ when $P = P_1 = P_2 = 4$ is shown in figure 4.7 where we interpret $\hat{u}_{pq}$ as the inner product $\hat{u}_{pq} = (\phi_{pq}, u)_\delta$. Similarly to the backward transformation, when using an unstructured expansion special attention must be paid to the degenerate vertices.

4.2 Global Operations

The operations described in section 4.1 were all local in the sense that they only involved a single element and no information was coupled from any other element. In general, however, we are interested in solving second-order partial differential equations which require that some form of continuity is maintained between elemental regions. Our primary focus in this section will be the classical Galerkin method where continuity is typically imposed by making the approximation gobally C^0 continuous. Alternative continuity requirements are imposed in techniques such as the mortar method or discontinuous Galerkin methods but we leave discussion of these technique to sections 7.4 and 7.5. We note, however, that all the local operations described in the previous section are equally applicable to all spectral/*hp* element formulations.

To construct a globally C^0 continuous expansion from elemental or local contributions we need to introduce a local to global assembly process, often referred to as *direct stiffness summation* or *global assembly*. This process was introduced

in one-dimension in section 2.3.1.4. This type of assembly is particularly important in setting up global matrices such as the Mass and Laplacian systems. In section 2.3.1 we also saw how, in one-dimension, the global linear finite element can be decomposed into elemental contributions of two similarly shaped linear varying modes.

In what follows, we will adopt a similar formulation for the multi-dimensional expansions to that defined in chapter 3. We recall that an important part of the construction of the elemental bases for Galerkin formulations was the decomposition of the basis into modes which contribute to the expansion on the boundary of an element (*boundary modes*) and the remaining modes which were zero on all boundaries (*interior modes*) (see section 3.1.1.1). As shown in figure 4.8 this boundary/interior decomposition allows us to construct a C^0 expansion by matching boundary modes of a similar shape. This figure illustrates all the modes used in a quadrilateral modal expansion of order $P_1 = P_2 = 4$, and we can appreciate that to construct the global expansion we can simply match the vertex and edge modes of similar shape.

In a practical implementation, it is advantageous to perform most operations in a local environment within each element and then assemble the local contributions to form the global system. However, to enable us to do this we need a mapping which relates to the global system from the local system. The definition of this mapping is central to the global assembly process and is discussed in section 4.2.1. Having constructed an assembly procedure we then illustrate the construction of a global matrix system in section 4.2.2 using an analogous matrix and vector notation which is introduced in section 4.1.5.1. Finally, in section 4.2.3, we introduce a matrix manipulation technique known as static condensation which takes advantage of the global matrix structure that arises for many spectral/*hp* element expansions and allows more efficient inversion of the global matrix.

4.2.1 Global Assembly and Connectivity

Before describing the global assembly procedure we need to define the local expansion modes $\phi_{pq}(\xi_1, \xi_2)$ within our global solution domain Ω. If Ω is divided into N_{el} contiguous elemental regions denoted by Ω^e the expansion modes $\phi^e_{pq}(\xi_1, \xi_2)$ are defined as:

$$\phi^e_{pq}(\xi_1, \xi_2) \begin{cases} \phi_{pq}(\xi_1, \xi_2) & (x_1, x_2) \in \Omega^e \\ 0 & \text{otherwise} \end{cases}$$

where

$$\xi_1 = [\chi^e_1]^{-1}(x_1, x_2), \qquad \xi_2 = [\chi^e_2]^{-1}(x_1, x_2)$$

and χ^e_i represents a bijective mapping from (ξ_1, ξ_2) onto $(x_1, x_2) \in \Omega^e$ (as introduced in section 4.1.3). We see that the local expansions within Ω^e are extended to the global domain Ω by having zero support everywhere except in the region Ω^e. Clearly, the elemental boundary modes cannot be C^0 continuous in the global region Ω. The interior modes, however, which are, by definition, zero on

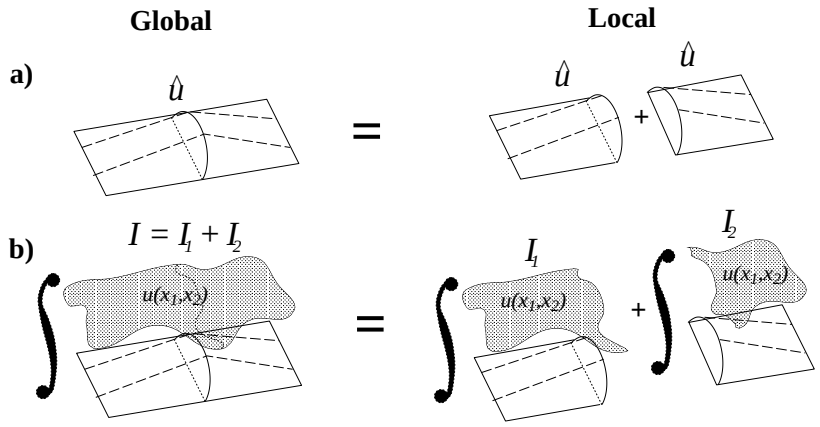


Figure 4.9 Illustration of local to global assembly. If we have a global expansion as represented in figure (a) it can be decomposed into two elemental contributions multiplied by the same global coefficient $\hat{u}$. To integrate a function $u(x_1, x_2)$ with respect to the global mode, as illustrated in figure (b), the integration in the global region is the sum of the integration in the local regions.

the boundary of the elements $\partial\Omega^e$, are C^0 continuous in the global region. This implies that the interior elemental degrees of freedom are also global degrees of freedoms.

From a practical point of view it is preferable to treat all operations locally within the standard elemental region where it is easier to define all the salient operations like integration and differentiation. This is possible if we also construct a mapping procedure which assembles our global system from the local systems defined on each element. The process is as follows:

1. Formulate a Galerkin elemental problem with respect to a set of global modes which constitute our trial space $\mathcal{X}$.
2. Split each global mode into local contributions over every element where all operations are performed.
3. Re-assemble the global system.

When we split the global expansion modes, as shown in figure 4.9(a), into their local elemental contributions the expansion coefficient $\hat{u}$ is transmitted to both of the elemental regions. However, when we need to integrate this global expansion mode with respect to some function $u(x_1, x_2)$ as shown in figure 4.9(b), this may be performed locally with respect to the elemental modes and then summed together to obtain the integral of $u(x_1, x_2)$ with respect to the global mode.

We recall that the Galerkin method is constructed from the weak problem which is an integral form. We do not need explicitly to assemble the global expansion modes as we can treat the integration locally and sum the elemental contributions. Nevertheless, in order to describe the solution within the elemental

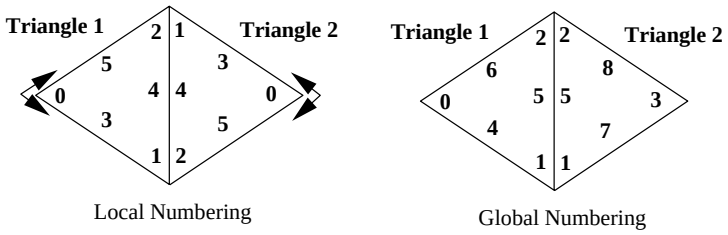


Figure 4.10 Illustration of local and global numbering schemes for a region containing two triangular elements. The numbering corresponds to a triangular modal expansion where $P_1 = P_2 = 2$, which only includes boundary modes. The orientation of the local coordinate system for each triangle is indicated by the arrow system.

region we will need first to perform the one-to-many mapping which takes the global system to the local elemental system. The assembly process is referred to as *direct stiffness summation* or *global assembly*. The word summation is somewhat misleading as only adjacent boundary modes of similar shape need to be added together and so we shall refer to the process as *global assembly*.

We define the *local degrees of freedom* as all the elemental expansion coefficients over all elements. We have previously introduced the vector $\hat{\boldsymbol{u}}$ to represent a consecutive list of all expansion modes within an elemental region. If we now use a superscript e to denote the elemental vector of expansion coefficients $\hat{\boldsymbol{u}}^e$ then the vector of all the local degrees of freedom, denoted by $\hat{\boldsymbol{u}}_l$, is,

$$\hat{\boldsymbol{u}}_l = \underline{\hat{\boldsymbol{u}}}^e = \begin{bmatrix} \hat{\boldsymbol{u}}^1 \\ \hat{\boldsymbol{u}}^2 \\ \vdots \\ \hat{\boldsymbol{u}}^{N_{el}} \end{bmatrix}, \tag{4.78}$$

which is of dimension N_{eof}. We also introduce the notation that an underlined vector implies the extension over all elemental regions. In section 4.2.2 we will see that an underlined matrix denotes a block diagonal extension of the matrix. To complement $\hat{\boldsymbol{u}}_l$ we define $\hat{\boldsymbol{u}}_g$ to denote the global degrees of freedom which is a vector of dimension N_{dof}. The many-to-one mapping from global to local degrees of freedom can be represented by the matrix operation $\boldsymbol{\mathcal{A}}$, that is,

$$\hat{\boldsymbol{u}}_l = \boldsymbol{\mathcal{A}}\hat{\boldsymbol{u}}_g. \tag{4.79}$$

The matrix $\boldsymbol{\mathcal{A}}$ is a very sparse rectangular matrix of dimension $N_{eof} \times N_{dof}$ whose values may typically be either 1 or -1 depending on the shape of connecting modes. For a nodal expansion all entries are positive. Typically only one entry will appear on any given row of the matrix. However, for different types of continuity conditions, such as the constrained approximation where two geometrically non-conforming elements meet (see section 7.3), multiple entries may appear on rows and columns of the assembly matrix. To illustrate the form of the assembly matrix $\boldsymbol{\mathcal{A}}$ we consider the case shown in figure 4.10 where we have

$$\hat{\boldsymbol{u}}_l = \begin{bmatrix} \hat{\boldsymbol{u}}^1[0] \\ \hat{\boldsymbol{u}}^1[1] \\ \hat{\boldsymbol{u}}^1[2] \\ \hat{\boldsymbol{u}}^1[3] \\ \hat{\boldsymbol{u}}^1[4] \\ \hat{\boldsymbol{u}}^1[5] \\ \cdots \\ \hat{\boldsymbol{u}}^2[0] \\ \hat{\boldsymbol{u}}^2[1] \\ \hat{\boldsymbol{u}}^2[2] \\ \hat{\boldsymbol{u}}^2[3] \\ \hat{\boldsymbol{u}}^2[4] \\ \hat{\boldsymbol{u}}^2[5] \end{bmatrix} = \begin{bmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & 1 & & & & & & \\ & & & & 1 & & & & \\ & & & & & 1 & & & \\ & & & & & & 1 & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & 1 & & & & & \\ & & 1 & & & & & & \\ & 1 & & & & & & & \\ & & & & & & & 1 & \\ & & & & 1 & & & & \\ & & & & & & 1 & & \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{u}}_g[0] \\ \hat{\boldsymbol{u}}_g[1] \\ \hat{\boldsymbol{u}}_g[2] \\ \hat{\boldsymbol{u}}_g[3] \\ \hat{\boldsymbol{u}}_g[4] \\ \hat{\boldsymbol{u}}_g[5] \\ \hat{\boldsymbol{u}}_g[6] \\ \hat{\boldsymbol{u}}_g[7] \\ \hat{\boldsymbol{u}}_g[8] \end{bmatrix}.$$

Figure 4.11 Relation between the local $\hat{\boldsymbol{u}}_l$ and global $\hat{\boldsymbol{u}}_g$ degrees of freedom using the assembly matrix $\boldsymbol{\mathcal{A}}$.

a domain containing two triangular elements. In this example we are considering an expansion order of $P_1 = P_2 = 2$ which only contains boundary modes. Therefore, the number of modes in each element is $N_m = (P_1+1)(P_2+2)/2 = 6$ and the total number of local degrees of freedom is $N_{eof} = 2N_m = 12$. In the left-hand plot of figure 4.10 we see the local numbering of the $N_m = 6$ elemental modes. This is dependent upon the orientation of the local coordinate system within the triangle as indicated by the arrow system. We have numbered the local degrees of freedom according to the convention where vertices are labelled first followed by edges, then faces (in three-dimensions), and finally the interior modes.

To enforce C^0 continuity between the two triangles we must match the boundary modes $(1, 4, 2)$ in triangle 1 with the boundary modes $(1, 4, 2)$ in triangle 2. This is achieved by assigning a global numbering scheme of $N_{dof} = 9$ global degrees of freedom as shown in the right-hand plot. Similarly to the local numbering scheme, the global numbering convention applied is numbering all global vertices first followed by all global edges, faces (in three-dimensions) and finally the interior modes where interior elemental blocks are numbered consecutively. This type of global numbering scheme, particularly when interior modes are number consecutively, is also convenient for the static condensation technique described in section 4.2.3.

The assembly matrix $\boldsymbol{\mathcal{A}}$ which relates the local degrees of freedom $\hat{\boldsymbol{u}}_l$ to the global degrees of freedom $\hat{\boldsymbol{u}}_g$ is shown in figure 4.11. In this figure we see that every row of the matrix $\boldsymbol{\mathcal{A}}$ contains only one entry signifying the fact that each local degree of freedom is related to one global degree of freedom. Every column of the matrix $\boldsymbol{\mathcal{A}}$ contains at least one entry although for geometrically non-conforming elements or mortar constructions there may be more than one

entry. In the case where every row contains only one entry the summation of the absolute value of the columns tells us how many local modes contribute to construct a global degree of freedom, which is known as the *multiplicity* of the mode.

We are now in a position to define the assembly process from local to global degrees of freedom. The action of the assembly process can be mathematically expressed as the transpose of $\boldsymbol{\mathcal{A}}$ but is heuristically captured by the integral operation similar to the case shown in figure 4.9. If we consider the inner product function $u(x_1, x_2)$ with respect to the global basis $\Phi_n(x_1, x_2)$,

$$\hat{\boldsymbol{I}}_g[n] = \int_\Omega u(x_1, x_2)\Phi_n(x_1, x_2), \qquad 0 \leq n < N_{dof}$$

this series of integrals can be expressed as elemental contributions, such that

$$\hat{\boldsymbol{I}}_g[n] = \int_\Omega u(x_1, x_2)\Phi_n(x_1, x_2) = \int_{\Omega^e} u(x_1, x_2)\phi_m(x_1, x_2)dx_1dx_2 \tag{4.80}$$

where $n(m, e)$ represents a unique global indexing of each elemental modal contribution m over each element e. This will be defined in term of a mapping array map$[e][i]$ shortly. The evaluation of the integrals (4.80) over all global modes $0 \leq n \leq N_{dof}$ can be represented in matrix form as

$$\hat{\boldsymbol{I}}_g = \boldsymbol{\mathcal{A}}^T \hat{\boldsymbol{I}}_l = \boldsymbol{\mathcal{A}}^T \underline{\hat{\boldsymbol{I}}}^e.$$

In the above equation $\hat{\boldsymbol{I}}_l$ is analogous to the definition (4.78) where

$$\hat{\boldsymbol{I}}^e[m] = \int_{\Omega^e} u(x_1, x_2)\phi_m(x_1, x_2)dx_1dx_2,$$

and m denotes the summation over all elemental modes which may involve a tensor product basis $\phi_m(p, q) = \phi_{pq}$ (see section 4.1.5.1).

We note that the matrix operations $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{A}}^T$ are not the inverse of each other, and therefore

$$\hat{\boldsymbol{u}}_g \neq \boldsymbol{\mathcal{A}}^T \boldsymbol{\mathcal{A}} \hat{\boldsymbol{u}}_g.$$

The operation of $\boldsymbol{\mathcal{A}}$ is, a scatter from a global to local system whereas the operation of $\boldsymbol{\mathcal{A}}^T$ is a global assembly or summation procedure. The inverse of the $\boldsymbol{\mathcal{A}}$ matrix would normally be considered as a standard "gather" type procedure. The operations of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{A}}^T$ are the key constructs to form a global system when using the Galerkin technique.

As an aside, we note that all the boundary modes touching the solution domain boundary have been treated as global degrees of freedom. As we shall see in section 4.3.1, boundaries with Neumann conditions are typically treated in this fashion. However, boundaries associated with Dirichlet conditions are not part of the Galerkin test space and therefore some reordering is required to remove them from the global degrees of freedom.

In a numerical implementation it is not practical, or even desirable, to construct $\boldsymbol{\mathcal{A}}$ explicitly due to the size and sparsity of the matrix. The operation may be numerically implemented by setting up a mapping array which we will denote as $n(e,i) = \text{map}[e][i]$. This array is of dimension $[N_{el} \times \max_e({N_m}^e)]$ where ${N_m}^e$ is the number of expansion modes in an elemental expansion. Typically, ${N_m}^e$ will be fixed over all elements although, in general, the value may change between elements. The array map$[e][i]$ contains the global value of the i-th expansion coefficient within the e-th element. The example shown in figure 4.10 would therefore have an array map$[e][i]$ of the form:

$$\text{map}[1][i] = \begin{cases} 0 \\ 1 \\ 2 \\ 4 \\ 5 \\ 6 \end{cases} \qquad \text{map}[2][i] = \begin{cases} 3 \\ 2 \\ 1 \\ 8 \\ 5 \\ 7 \end{cases}.$$

The total number of entries of the map$[e][i]$ is the same as the number of non-zero entries in $\boldsymbol{\mathcal{A}}$. The scatter operation $\boldsymbol{\mathcal{A}}$ from $\hat{\boldsymbol{u}}_g$ to $\hat{\boldsymbol{u}}_l$ can now be evaluated by:

$$\left.\begin{array}{l} \text{Do } e = 1, N_{el} \\ \quad \text{Do } i = 0, {N_m}^e - 1 \\ \quad\quad \hat{\boldsymbol{u}}^e[i] = \text{sign}[e][i] \cdot \hat{\boldsymbol{u}}_g[\text{map}[e][i]] \\ \quad \text{continue} \\ \text{continue} \end{array}\right\} \Leftrightarrow \hat{\boldsymbol{u}}_l = \boldsymbol{\mathcal{A}}\hat{\boldsymbol{u}}_g \qquad (4.81)$$

Implementation note: *Numerical implementation of the global to local gather operation denoted by* $\boldsymbol{\mathcal{A}}$.

where sign$[e][i]$ is an array of similar dimensions to map$[e][i]$ containing 1 or -1 entries depending on the modal connectivity between two elements as discussed in 4.2.1.1. For a nodal expansion sign$[e][i]$ would only contain positive entries and so they may be removed from the loop. The global assembly operation can be evaluated as:

$$\left.\begin{array}{l} \text{Do } e = 1, N_{el} \\ \quad \text{Do } i = 0, {N_m}^e - 1 \\ \quad\quad \hat{\boldsymbol{I}}_g[\text{map}[e][i]] = \hat{\boldsymbol{I}}_g[\text{map}[e][i]] \\ \quad\quad\quad\quad\quad\quad + \text{sign}[e][i] \cdot \hat{\boldsymbol{I}}^e[i] \\ \quad \text{continue} \\ \text{continue} \end{array}\right\} \Leftrightarrow \hat{\boldsymbol{I}}_g = \boldsymbol{\mathcal{A}}^T \hat{\boldsymbol{I}}_l. \qquad (4.82)$$

Implementation note: *Numerical implementation of the local to global assembly operation denoted by* $\boldsymbol{\mathcal{A}}^T$.

If the inner summation did not contain the $\hat{\boldsymbol{v}}_g[\text{map}[e][i]]$ term on the right-hand side it would be the standard "gather" operation.

4.2.1.1 *Local to Global Boundary Mapping: Global Boundary Assembly*

We have seen that the global assembly procedure primarily involves boundary mode connectivity as the interior modes may be independently numbered as global degrees of freedom. We shall also see in section 4.2.3 that the assembly procedure need only involve the boundary modes as the interior modes may be

removed from the full matrix problem using static condensation. In this case we only require a boundary mapping bmap$[e][i]$ rather than the full numbering system map$[e][i]$.

We assume that the local degrees of freedom are ordered so that the boundary modes are listed first. If there are $n_b[e]$ boundary modes in the e-th element the local to global assembly process is numerically evaluated as

$$\left.\begin{array}{l}\text{Do } e = 1, N_{el} \\ \quad \text{Do } i = 0, n_b[e] - 1 \\ \quad\quad \hat{\boldsymbol{I}}_g[\text{bmap}[e][i]] = \hat{\boldsymbol{I}}_g[\text{bmap}[e][i]] \\ \quad\quad\quad\quad +\text{sign}[e][i] \cdot \hat{\boldsymbol{I}}^e[i] \\ \quad \text{continue} \\ \text{continue}\end{array}\right\} \Leftrightarrow \hat{\boldsymbol{I}}_g = \boldsymbol{\mathcal{A}}_b^T \underline{\hat{\boldsymbol{I}}_b^e} \qquad (4.83)$$

Implementation note: *Numerical implementation of the local to global boundary assembly operation denoted by* $\boldsymbol{\mathcal{A}}_b^T$.

which is identical to map$[e][i]$ in (4.82) save that we are now only using the first $n_b[e]$ elements of the mapping. $\hat{\boldsymbol{I}}^e$ and $\hat{\boldsymbol{I}}_g$ have their usual meaning referring to the local and global degrees of freedom even though only the entries corresponding to the boundary modes are being used. Similar to 'bmap$[e][i]$', the array sign$[e][i]$ need only be of dimension N_{el} by the maximum size of $n_b[e]$. We see from (4.83) that the equivalent matrix operation is denoted by $\boldsymbol{\mathcal{A}}_b^T$ which is a submatrix of $\boldsymbol{\mathcal{A}}^T$ and operates on the vector of all local boundary degrees of freedom denoted by $\underline{\hat{\boldsymbol{v}}_b^e}$. If we know how to construct bmap$[e][i]$, it is a straightforward extension to generate map$[e][i]$ by adding a unique block of global degrees of freedom equal in length to the number of interior modes within the element.

Modal Edge Connectivity

Implementation note: *Modal basis sign determination for inter-element edge global assembly*

In figure 4.12 we see all the modes for an order $P_1 = P_2 = 4$ expansion in two quadrilateral elements. Note that each mode is to be interpreted as spanning the entire element. When considering an expansion with more than one edge mode we need to consider the local orientation of the element. As shown in figure 4.12, depending on the orientations of the local coordinate systems within the element the sign of odd-ordered modes may need to be reversed. This is in contrast to figure 4.8 where the cubic edge mode had a similar shape on either side of the the intersecting edge. The reason for the sign negation is that the elemental modal shapes are defined with respect to the local coordinate system (ξ_1, ξ_2). If the local systems are orientated so that the two neighbouring coordinates are in opposite directions then the sign of one odd-shaped mode will need to be reversed.

It also appears from figure 4.12 that the *order* of the edge modes needs to be reversed. This is, however, not the case. The hierarchical boundary modes only have a physical interpretation insofar as they are associated with a physical vertex or edge within the region. Therefore, we number the edge modes according to their polynomial order (that is, lowest polynomial order mode has the lowest edge number). For the example shown in figure 4.12, the numbering of the local modes is shown in figure 4.13 where we have placed all numbering for a given

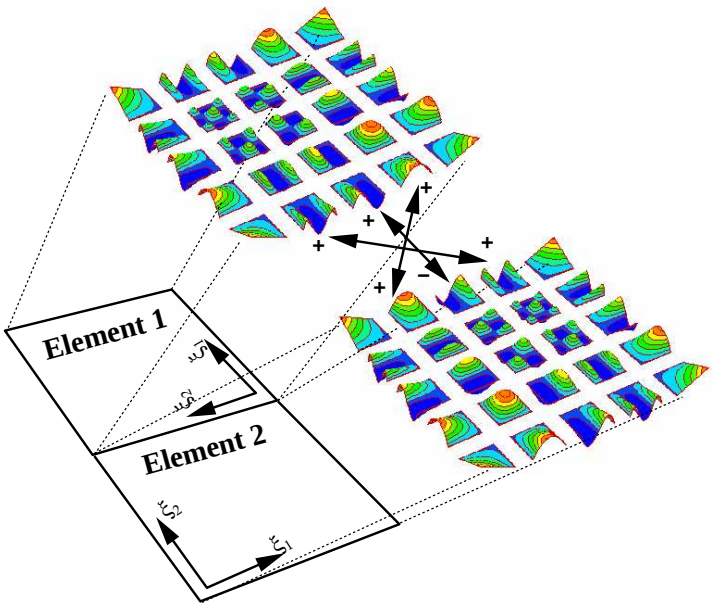


Figure 4.12 Illustration of the construction of a C^0 global expansion from two local *modal* expansions of order $P_1 = P_2 = 4$. To ensure C^0 continuity the boundary modes of similar shape need to be matched. Depending on the orientation of the local coordinate systems the modes of odd order may also need to be negated.

edge at the centre point of the edge. If we follow a similar convention when numbering the global modes, as shown in the right-hand side of figure 4.13, the modes of similar polynomial order (which we need to match) will have the same global number and so we are just left with the issue of sign reversal. In this example, the mode of cubic order needs to have its sign reversed in one element as the local coordinate system has an opposite direction along the intersecting edge. By convention, we assume that the element with the lowest number has precedence and therefore mode 14 in triangle 1 will be negated. Therefore, when assembling the array sign$[e][i]$ we require that sign$[1][11] = -1$.

In general, we need an automatic procedure to identify which edges need to have odd modes negated. Such a procedure may be constructed by considering the sign of the inner product between two vectors representing the global co-ordinate direction of an edge. Since we always know the vertices which define an element, we let $\boldsymbol{\Delta x}^e_{edg}$ denote a vector parallel to an edge in an element "e" oriented according to the local coordinate direction of ξ_1 or ξ_2. For example, along the bottom edge where $(\xi_2 = -1)$ we have

$$\boldsymbol{\Delta x}^e_{edg} = \begin{bmatrix} \chi_1(1,-1) - \chi_1(-1,-1) \\ \chi_2(1,-1) - \chi_2(-1,-1) \end{bmatrix}, \qquad x_1 = \chi_1(\xi_1,\xi_2), x_2 = \chi_2(\xi_1,\xi_2)$$

or along the edge where $\xi_1 = -1$

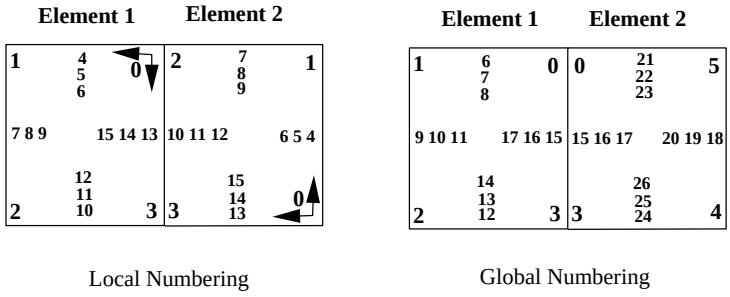


Figure 4.13 Numbering system for a hierarchical quadrilateral expansion of order $P_1 = P_2 = 4$ where the arrows indicate the local coordinate system. The individual edge modes have no physical location associated with the expansion and so are listed at the centre-point of the edge. The number nearest the edge corresponds to the edge mode of the lowest polynomial order. Following a similar ordering for the global numbering means that modes of similar order are automatically matched.

$$\boldsymbol{\Delta x}^e_{edg} = \begin{bmatrix} \chi_1(-1,1) - \chi_1(-1,-1) \\ \chi_2(-1,1) - \chi_2(-1,-1) \end{bmatrix}, \qquad x_1 = \chi_1(\xi_1,\xi_2), x_2 = \chi_2(\xi_1,\xi_2).$$

To determine whether the odd edge modes need to have their sign reversed on an edge between element e and element k we apply the test if

$$\boldsymbol{\Delta x}^e_{edg} \cdot \boldsymbol{\Delta x}^k_{edg} < 0 \tag{4.84}$$

and

$$k > e$$

then negate odd modes. The extra criterion $k > e$ simply ensures that only one of the two edge modes is negated and implies that we have some information about the edge connectivity. This test only identifies whether the local coordinate systems along a specific edge are in the same or opposite directions. An identical procedure may be applied to edges of a triangular region by use of η_1, η_2 instead of ξ_1, ξ_2. For edges in a three-dimensional mesh an analogous test may be set up where $\boldsymbol{\Delta x}^e_{edg}$ is now a three-dimensional vector. In this case, the number of elemental domains along an edge will typically be greater than two and therefore the test needs to be performed relative to the edge from the element with the lowest number.

Nodal Edge Connectivity

Implementation note: *Nodal basis numbering for inter-element edge global assembly.*

When using a nodal expansion the modes may be identified with a physical location of the nodal points where the modes have a unit value. In the nodal expansion, we are not concerned with matching edge modes of similar order (as in the hierarchical expansion case) but with matching modes with the same nodal location as illustrated in figure 4.14. The physical interpretation of an expansion

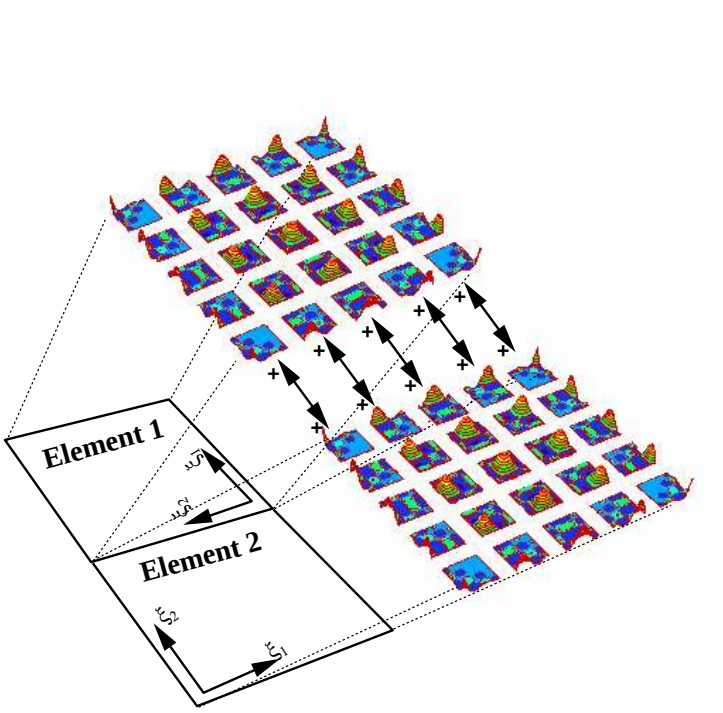


Figure 4.14 Illustration of the construction of a C^0 global expansion from two local *nodal* expansions of order $P_1 = P_2 = 4$. To ensure C^0 continuity the boundary modes with similar nodal locations need to be matched.

mode being associated with a nodal position makes it easier to generate a numbering system, by numbering the location of the nodal points along an edge as shown in figure 4.15.

In figure 4.13 we have chosen to locally number the elemental degrees of freedom using an anti-clockwise convention where the vertex modes are listed first. Using an anti-clockwise convention ensures that one side of the elemental matching is always reversed with respect to the global numbering. For example, the modes in element 1 have locally increasing numbers $(13, 14, 15)$ corresponding to globally increasing numbers $(15, 16, 17)$ whereas the modes in element 2 have locally increasing numbers $(10, 11, 12)$ corresponding to globally decreasing numbers $(17, 16, 15)$. If we had ordered the local edge modes according to the direction of the local coordinate system, we would have a similar situation to the hierarchical expansions where we would have to determine if we needed to reverse the ordering depending on the direction of the local edge coordinate. For the nodal expansion in two-dimensions, the use of an anti-clockwise local numbering scheme implies that the ordering is always reversed between two elements and therefore no extra test is required.

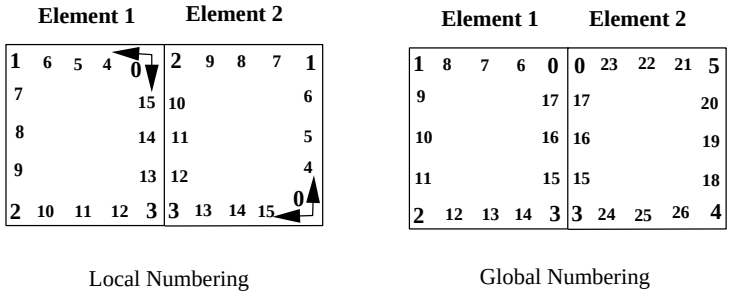


Figure 4.15 Numbering system for a nodal quadrilateral expansion of polynomial order $P_1 = P_2 = 4$. The arrows indicate the orientation of the local coordinate system (ξ_1, ξ_2). In a nodal expansion the edge mode can be physically identified with the nodal points of the Lagrange polynomial. Numbering each nodal location therefore provides a global numbering scheme which will ensure C^0 continuity.

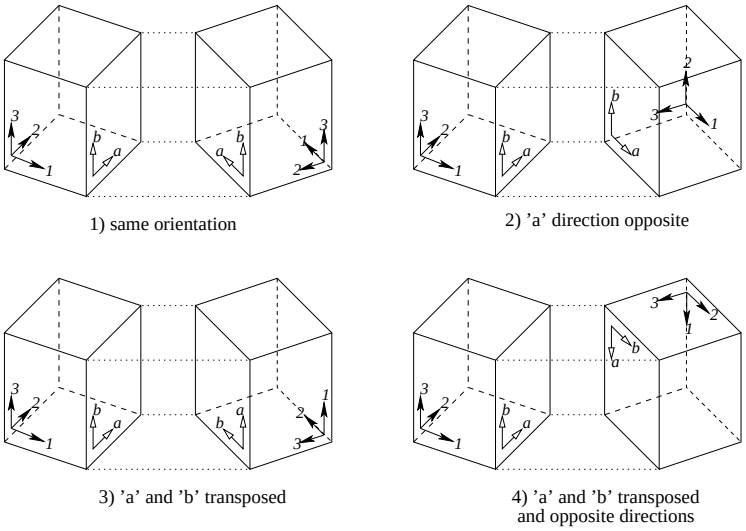


Figure 4.16 Example of some of the different local coordinate alignments when two hexahedral elements are matched. The local Cartesian coordinate are denoted by the $(1, 2, 3)$ axis and the local face coordinates are denoted by the (a, b) axis.

Modal and Nodal Quadrilateral Face Orientation and Connectivity

Implementation note: *Orientation requirement and connectivity issues for global assembly between adjacent quadrilateral faces.*

One of the complexities of matching three-dimensional shapes with similar shaped faces as compared to the two-dimensional edge matching is the number of different orientations that can be imposed. This issue is highlighted in figure 4.16 where we show some of the different local face alignments between the quadri-

laterl faces of two hexahedral elements. In this figure the $(1, 2, 3)$ axes systems denotes the local Cartesian system of each hexahedral element. Similarly the (a, b) axis system denotes the local face coordinates where the a direction is aligned to the lowest local Cartesian coordinate within the local face. From this figure we see that, as in the two-dimensional case, local coordinates can be aligned in opposite directions. However, unlike the two-dimensional case we also have a possible situation where local coordinates can be transposed (i.e., the 'a' direction in one element is aligned to the 'b' direction in another).

From the previous two subsections we have observed that when the local coordinate direction is reversed then, in a modal expansion, we expect to use the same number scheme in bmap$[e][i]$ although there will be sign changes in the array sign$[e][i]$. However, for a nodal expansion the reversing of the local coordinate direction necessitates a change in the local array numbering, i.e. bmap$[e][i]$ but does not influence sign$[e][i]$. Therefore, considering figure 4.16 in case (1) we expect the same bmap$[e][i]$ and sign$[e][i]$ in both faces. In case (2) of figure 4.16 the modal numbering is not altered between the two faces although there is a sign change due to the local $'a'$ coordinate reversing direction between the two faces. For this case there would have been a numbering alteration if we were using a nodal case. Finally, for both cases (3) and (4) in figure 4.16 we require a transposition of the numbering scheme for both modal and modal expansions due to the transposing of the local coordinate system.

In assembling bmap$[e][i]$ and sign$[e][i]$ between two modal quadrilateral faces we therefore need to know how a face is orientated and then whether each local axis is aligned in the same or opposite directions. Determining how the local face coordinates are orientated with respect to each other permits us to locally number the face degrees of freedom and therefore construct the component of bmap$[e][i]$. Determining whether the local axes of the two faces are aligned in the same or opposite directions informs us whether or not the odd order components of the expansion basis in each coordinate direction need to be negated in one of the adjacent faces and so permitting the construction of sign$[e][i]$.

There are many different techniques which could be applied to determine the orientation and sign of the local face coordinates. As a potential example we can consider the following construction. If we are considering a face defined by $\xi_3 = -1$ in element e then linear two edge vectors can be defined as:

$$\boldsymbol{\Delta x}_a^e = \begin{bmatrix} \chi_1(1,-1,-1) - \chi_1(-1,-1,-1) \\ \chi_2(1,-1,-1) - \chi_2(-1,-1,-1) \\ \chi_3(1,-1,-1) - \chi_3(-1,-1,-1) \end{bmatrix},$$

$$\boldsymbol{\Delta x}_b^e = \begin{bmatrix} \chi_1(-1,1,-1) - \chi_1(-1,-1,-1) \\ \chi_2(-1,1,-1) - \chi_2(-1,-1,-1) \\ \chi_3(-1,1,-1) - \chi_3(-1,-1,-1) \end{bmatrix},$$

where

$$x_1 = \chi_1(\xi_1, \xi_2, \xi_3);\ x_2 = \chi_2(\xi_1, \xi_2, \xi_3);\ x_3 = \chi_3(\xi_1, \xi_2, \xi_3).$$

We then analogously define two edge vectors,$\mathbf{\Delta x}_a^k$ and $\mathbf{\Delta x}_b^k$ in the adjacent face of element k under the constraint that they have the same common vertex, $[\chi_1^e(-1,-1,-1), \chi_2^e(-1,-1,-1), \chi_3^e(-1,-1,-1)]$. Whether the $(a,b)^e$ coordinate system in element e has a transpose orientation to the $(a,b)^k$ coordinate system of element k can be determined by an inner product test of the form:
If

$$\mathbf{\Delta x}_a^e \cdot \mathbf{\Delta x}_a^k = |\mathbf{\Delta x}_a^e||\mathbf{\Delta x}_a^k| \tag{4.85}$$

then

$$\mathbf{\Delta x}_a^e \parallel \mathbf{\Delta x}_a^k \quad \Rightarrow \quad a^e \parallel a^k,\ b^e \parallel b^k$$

else

$$\mathbf{\Delta x}_a^e \parallel \mathbf{\Delta x}_b^k \quad \Rightarrow \quad a^e \parallel b^k,\ b^e \parallel a^k.$$

In a modal expansion if $\mathbf{\Delta x}_a^e$ is parallel to $\mathbf{\Delta x}_a^k$ then the modes in each face can be numbered in an identical manner. However, if $\mathbf{\Delta x}_a^e$ is parallel to $\mathbf{\Delta x}_b^k$ we have to set up a transposed numbering system in one of the two adjacent faces. For example, if $\phi_{pq}^e(\xi_1, \xi_2, -1)$ represents the $(P-1)(P-1)$ modes within a face of element e then the matching modes in a face of element k must be

$$\phi_{pq0}^k(\xi_1, \xi_2, -1) = \pm\phi_{qp0}^e(\xi_1, \xi_2, -1), \qquad 0 < p, q < P, \tag{4.86}$$

where we have assumed that the connecting face in element k is defined by $\xi_3 = -1$. Therefore, if the mapping bmap$[e][i]$ for ϕ_{pq0}^e is initially chosen then the mapping in bmap$[k][i]$ must be ordered so that condition (4.86) is satisfied.

Finally, we need to determine the possible sign change between modes to set up the array sign$[e][i]$ for the global assembly process. Providing the local edge vectors $\mathbf{\Delta x}_a^e, \mathbf{\Delta x}_a^k, ...$ are defined so a positive vector is aligned in the positive direction of the local axes, then the sign of the matching modes can be determined by applying test (4.84). Clearly the local edge vectors used in this test depend on whether the coordinates are transposed or not. Testing first the a and then the b direction of a reference face indicates whether the odd numbered p and q index modes require negating within the reference face for the two face modes to match identically.

For a *nodal face* we could adopt a similar procedure where we use the orientation test (4.85) to determine how the nodal points in a face are orientated relative to the adjacent face. However, a simpler, although more expensive, approach is to number the nodal locations in one face and then determine the global numbering of the adjacent face by matching every nodal (x_1, x_2, x_3) position in one face with the other. As the numbering procedure may be considered as an overhead cost at preprocessing stage there is generally no concern over the cost of this approach.

Modal Triangular Face Orientation and Connectivity

Implementation note: *Orientation requirement and connectivity issues for global assembly between adjacent triangular faces.*

Similar to the quadrilateral face, in matching two triangular faces we also have to consider the orientation of a face. To generate a C^0 global expansion we

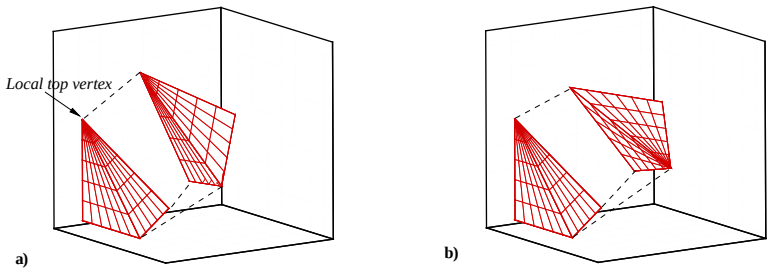


Figure 4.17 To ensure that the modal expansion can be assembled to form a C^0 expansion, the collapsed coordinate system in the triangular faces must be aligned as shown in plot (a). Connection of two faces where the local coordinate system is oriented, as shown in plot (b), is not permitted.

want to match face modes of similar shape. For modal expansions the use of the collapsed Cartesian system means that triangular faces have a local coordinate system which is not rotationally symmetric. This point is illustrated in figure 4.17 where we see the two tetrahedral regions marked with their surface coordinate lines. To be able to match modes within a triangular face we require that these coordinate lines are orientated as shown in figure 4.17(a) but not as shown in figure 4.17(b). This would appear to be rather constraining. We are, however, free to specify how the local coordinate systems within an element are orientated. For an arbitrary conforming tetrahedral mesh, it is possible to orient all the local coordinate system so that the coordinate lines are consistent. Referring to the two vertices where the coordinates system degenerates as the *local base vertex* ($\xi_1 = -1, \xi_2 = 1, \xi_3 = -1$) and the *local top vertex* ($\xi_1 = -1, \xi_2 = -1, \xi_3 = 1$) an orientation algorithm suggested by Warburton [485] is:
assuming that every global vertex has a unique number then for every element we have four vertices with unique global numbers:

(i) Place the *local top vertex* at the global vertex with the lowest global number

(ii) Place the *local base vertex* at the global vertex with the second lowest global number

(iii) Orient the last two vertices to be consistent with the local rotation of the element (typically anti-clockwise).

This algorithm is local to each element and can be implemented at a pre-processing stage. Although it is possible to guarantee this connectivity for tetrahedral meshes, it is not possible for a general mesh using tetrahedrons, prisms, and pyramids. Nevertheless, enough permutations of connectivity still exist to provide a wide range of flexibility even when using all the three-dimensional hybrid elements [485].

The orientation criteria simplify the numbering and sign evaluation process in a triangular face as all faces have a similar orientation. Therefore, for a hi-

erarchical/modal expansion the global numbering on one face may be directly applied to the adjacent face without any index swapping. There is still, however, the possibility for modes which vary in the base direction (i.e., η_1 direction on face 1 or 2, and η_2 direction on face 3 or 4) to require their sign to be reversed. This may be determined by using the edge test (4.84).

The non-tensorial nodal expansions are rotationally symmetric and therefore do not have to comply with the above orientation criteria. However, application of this criteria in the case of a nodal tetrahedral expansion can still reduce the potential orientations of adjacent faces and thus simplify the construction of a numbering scheme. In this case the edge test (4.84) would indicate whether the number should be reversed in one direction between two adjacent faces. Nevertheless, similar to the quadrilateral faces, with a nodal expansion comparison of the coordinates of nodal points can again be used to determine an appropriate face ordering for bmap$[e][i]$.

4.2.2 Global Matrix System

Formulation note: *Construction of global matrix systems from elemental contributions.*

Now that we have a way of assembling the global system from a local system, we can apply this technique to generate global matrices from the elemental matrices. To illustrate this process we can consider a global forward transformation which is similar to the elemental transformation described in section 4.1.5 except that we now want to project a function into a C^0 continuous global expansion space. To recall the notation introduced in section 4.1.5, where the superscript e now refers to the element number, we have:

$\hat{\boldsymbol{u}}^e$	Vector of length N_m containing the expansion coefficients corresponding to the order of the basis matrix $\boldsymbol{B}$.
$\boldsymbol{u}^e$	Vector of length N_Q containing the function $u(\xi_1, \xi_2)$ evaluated at the quadrature points.
$\boldsymbol{B}^e$	$N_Q \times N_m$ basis matrix whose columns contain the basis $\phi_{pq}(\xi_1, \xi_2)$ evaluated at the quadrature points.
$\boldsymbol{W}^e$	$N_Q \times N_Q$ diagonal weight matrix containing the quadrature weights multiplied by the Jacobian at the quadrature points.

For a nodal non-tensorial basis the vector $\hat{\boldsymbol{u}}^e$ can also be interpreted as the solution at the nodal points of the Lagrange expansion. In section 4.2.1 we also introduced the notation for global systems:

$\hat{\boldsymbol{u}}_g$	Vector of length N_{dof} containing the global expansion coefficients.
$\hat{\boldsymbol{u}}_l, \underline{\hat{\boldsymbol{u}}}^e$	Vector of length N_{eof} which is the concatenation of the local expansion coefficients $\hat{\boldsymbol{u}}^e$ over all N_{el} elements.
$\boldsymbol{u}_l, \underline{\boldsymbol{u}}^e$	$N_{el} \cdot N_Q$ vector which is the concatenation of the local vectors $\boldsymbol{u}^e$ over all N_{el} elements
$\boldsymbol{\mathcal{A}}$	$N_{eof} \times N_{dof}$ permutation matrix which constructs the local vector $\hat{\boldsymbol{u}}_l$ from the global vector $\hat{\boldsymbol{u}}_g$. ($\boldsymbol{\mathcal{A}}^T$ represents the global assembly process.)

In the above $\boldsymbol{u}_l$, has been introduced as an analogous vector to $\hat{\boldsymbol{u}}_l$, and contains the function evaluation at the quadrature points over all elements. We also note that

$$N_{eof} = \sum_{e=1}^{N_{el}} {N_m}^e$$

where ${N_m}^e$ is the number of expansion modes in the element e. Typically this is constant in which case $N_{eof} = N_{el} \cdot N_m$.

We recall that to determine $\hat{\boldsymbol{u}}^e$ given a vector $\boldsymbol{u}^e$ we performed a discrete elemental forward transformation which involved the solution of the matrix system

$$\boldsymbol{M}^e \hat{\boldsymbol{u}}^e = (\boldsymbol{B}^e)^T \boldsymbol{W}^e \boldsymbol{B}^e \hat{\boldsymbol{u}}^e = (\boldsymbol{B}^e)^T \boldsymbol{W}^e \boldsymbol{u}^e, \tag{4.87}$$

where $\boldsymbol{M}^e$ is the elemental mass matrix. We now want to set up an analogous system to determine the solution for the global vector $\hat{\boldsymbol{u}}_g$.

We can represent the elemental forward transformation over all elements in terms of a global matrix process by assembling diagonal matrices of the form:

$$\underline{\boldsymbol{M}^e} = \begin{bmatrix} \boldsymbol{M}^1 & 0 & 0 & 0 \\ 0 & \boldsymbol{M}^2 & 0 & 0 \\ & & \ddots & \\ 0 & 0 & 0 & \boldsymbol{M}^{N_{el}} \end{bmatrix}$$

$$\underline{(\boldsymbol{B}^e)^T \boldsymbol{W}^e} = \begin{bmatrix} (\boldsymbol{B}^1)^T \boldsymbol{W}^1 & 0 & 0 & 0 \\ 0 & (\boldsymbol{B}^2)^T \boldsymbol{W}^2 & 0 & 0 \\ & & \ddots & \\ 0 & 0 & 0 & (\boldsymbol{B}^{N_{el}})^T \boldsymbol{W}^{N_{el}} \end{bmatrix}.$$

In constructing the above matrix systems we have adopted the notation that an underlined matrix $\underline{\boldsymbol{M}^e}$ denotes the local matrices $\boldsymbol{M^1}, \boldsymbol{M^2}, ...$ as block diagonal entries to a larger matrix system. This is equivalent to the concatenation of $\boldsymbol{u}^e$ and $\hat{\boldsymbol{u}}^e$ into $\boldsymbol{u}_l = \underline{\boldsymbol{u}}^e, \hat{\boldsymbol{u}}_l = \underline{\hat{\boldsymbol{u}}}^e$, respectively. The matrix system

$$\underline{\boldsymbol{M}^e}\ \hat{\boldsymbol{u}}_l = \underline{(\boldsymbol{B}^e)^T W^e}\ \boldsymbol{u}_l \tag{4.88}$$

represents the local forward transformation over all N_{el} elements and is an equivalent statement to equation (4.87). This system is invertible since the local systems are decoupled and invertible but there is no guarantee of continuity between elements. Nevertheless, we know from section 4.2.1 that

$$\hat{\boldsymbol{u}}_l = \boldsymbol{\mathcal{A}} \hat{\boldsymbol{u}}_g, \tag{4.89}$$

which determines the local degrees of freedom from the global degrees of freedom. Substituting equation (4.89) into (4.88) we have

$$\underline{\boldsymbol{M}^e}\ \boldsymbol{\mathcal{A}} \hat{\boldsymbol{u}}_g = \underline{(\boldsymbol{B}^e)^T \boldsymbol{W}^e}\ \boldsymbol{u}_l. \tag{4.90}$$

The effect of post-multiplying the matrix $\underline{\boldsymbol{M}^e}$ by $\boldsymbol{\mathcal{A}}$ is to globally assemble the rows of this matrix from their local contributions. The matrix $\underline{\boldsymbol{M}^e} \boldsymbol{\mathcal{A}}$ is not square

as the columns of this system have not been globally assembled. This may be achieved by pre-multiplying the entire equation (4.90) by $\boldsymbol{\mathcal{A}}^T$ to obtain the global square system:

$$\left[\boldsymbol{\mathcal{A}}^T \underline{\boldsymbol{M}^e}\ \boldsymbol{\mathcal{A}}\right] \hat{\boldsymbol{u}}_g = \boldsymbol{\mathcal{A}}^T \underline{(\boldsymbol{B}^e)^T \boldsymbol{W}^e}\ \boldsymbol{u}_l, \tag{4.91}$$

which may be solved to determine the global forward transformation. The matrix in square brackets is the global mass matrix $\boldsymbol{M}$, that is,

$$\boldsymbol{M} = \boldsymbol{\mathcal{A}}^T \underline{\boldsymbol{M}^e}\ \boldsymbol{\mathcal{A}}.$$

Although we have set up a global system for the forward transformation using the local matrix $\boldsymbol{M}^e$, an identical procedure follows for any local matrix system such as the weak Laplacian system $\boldsymbol{L}^e$ defined in section 4.1.3. Therefore, the global weak Laplacian matrix $\boldsymbol{L}$ is

$$\boldsymbol{L} = \boldsymbol{\mathcal{A}}^T \underline{\boldsymbol{L}^e}\ \boldsymbol{\mathcal{A}}.$$

Finally, we note that global matrix systems such as (4.91) can only be directly assembled for relatively small problems (i.e., a low number of elements of low polynomial order). The matrix formulation, however, provides insight into understanding the important steps in constructing the global system. In practice, for a large problem we recall that the explicit inner product operation $\underline{(\boldsymbol{B}^e)^T \boldsymbol{W}^e}\ \boldsymbol{u}_l$ can be evaluated on an elemental level. If we use a tensorial basis then the sum-factorisation technique discussed in section 4.1.6 can also be applied. This produces a vector, equal in length to the local degrees of freedom, which may be assembled into the global form using a mapping array as shown by the operation (4.82). Therefore, we see that the left-hand-side of equation (4.91) can be efficiently evaluated using local elemental operations.

The idea of locally assembling the elemental matrices and then using the global assembly operator can be equally well applied to generate the global matrix system $\boldsymbol{M}$ from the local matrices $\boldsymbol{M}^e$. This system is of dimension $N_{dof} \times N_{dof}$ but is typically very sparse. Nevertheless, it is usually too large to invert, or even factor directly. In the next section we see how we can reduce this global matrix system into smaller components based on the natural decomposition of the spectral/*hp* element method.

4.2.3 Static Condensation/Substructuring

Although the following technique may be applied to a general non-symmetric matrix system we shall restrict our attention to symmetric matrix systems which typically arise in the Galerkin discretisation of symmetric operators.

We therefore assume that we have a system of the form:

$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{\mathcal{A}}^T \underline{\boldsymbol{M}^e} \boldsymbol{\mathcal{A}}\boldsymbol{x} = \boldsymbol{f}, \tag{4.92}$$

where $\boldsymbol{x}$ is a vector of global unknowns, typically the global vector of expansion coefficients $\hat{\boldsymbol{u}}_g$. $\underline{\boldsymbol{M}^e}$ is a block diagonal matrix which may have been formed from

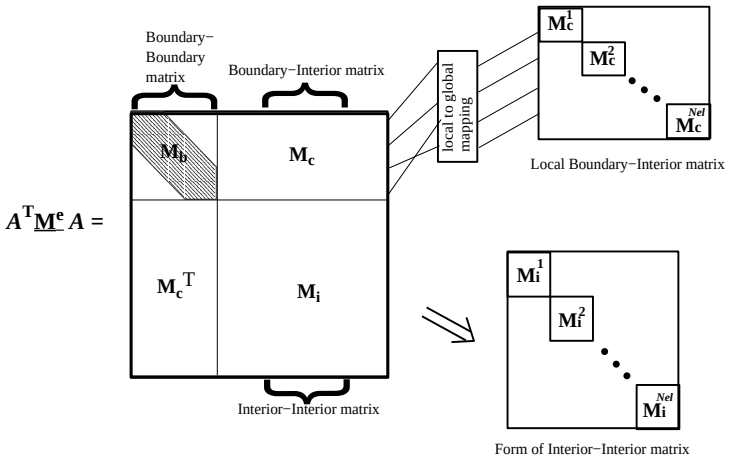


Figure 4.18 Depending on the global numbering scheme the global matrix system has a very distinctive structure as shown here. The boundary-boundary matrix is usually banded; the boundary-interior matrix can be constructed from the sparse local boundary-interior matrices; and the interior-interior matrix consists of smaller uncoupled matrices.

either the local mass or Laplacian matrices or indeed a combination of the two. The matrix $\boldsymbol{M}$ is typically very sparse although it may have a full bandwidth. It is therefore very inefficient, and potentially impossible, to store the full matrix so that it may be directly inverted. A far more efficient approach is to use the structure of the spectral/*hp* element discretisation which is the motivation behind *static condensation* or *substructuring*.

Each of the elemental matrices $\boldsymbol{M}^e$ can be split into components containing boundary and interior contributions, that is,

$$\boldsymbol{M}^e = \begin{bmatrix} \boldsymbol{M}_b^e & \boldsymbol{M}_c^e \\ (\boldsymbol{M}_c^e)^T & \boldsymbol{M}_i^e \end{bmatrix},$$

where $\boldsymbol{M}_b^e$ represents the components of $\boldsymbol{M}^e$ resulting from boundary-boundary mode interactions, $\boldsymbol{M}_c^e$ represents the components of $\boldsymbol{M}^e$ resulting from coupling between the boundary-interior modes and $\boldsymbol{M}_i^e$ represents the components of $\boldsymbol{M}^e$ resulting from interior-interior mode interactions.

As mentioned previously, if we know the value of $\boldsymbol{x}$ we can perform the matrix-vector operation $\boldsymbol{\mathcal{A}}^T \underline{\boldsymbol{M}}^e \boldsymbol{\mathcal{A}} \boldsymbol{x}$ far more efficiently by considering separately the local operations represented by the $\boldsymbol{\mathcal{A}}$ and $\underline{\boldsymbol{M}}^e$ matrices. This is one way of solving the system iteratively. However, if we need to directly invert the matrix $\boldsymbol{\mathcal{A}}^T \underline{\boldsymbol{M}}^e \boldsymbol{\mathcal{A}}$ we cannot perform each operation independently.

We recall that the assembly process, denoted by $\boldsymbol{\mathcal{A}}^T$, may be be viewed as a mapping and partial summation process where we are free to specify the order in which the global system is chosen. Previously, we have stated that the global

system is ordered so that the global boundary degrees of freedom (that is, those constructed from the local boundary modes) are listed first, followed by the global interior degrees of freedom (that is, those constructed from the interior modes of the elemental construction). In addition, the global interior degrees of freedom were numbered consecutively. This ordering is important to make maximum use of the structure of the discretisation in the static condensation process. If we adopt this ordering, the global system has the form shown in figure 4.18.

In this figure the matrix $\boldsymbol{M}_b$ corresponds to the global assembly of the elemental boundary-boundary mode interaction from $\boldsymbol{M}_b^e$ and similarly $\boldsymbol{M}_c, \boldsymbol{M}_i$ correspond to the global assembly of the elemental boundary-interior coupling and interior-interior systems $\boldsymbol{M}_c^e, \boldsymbol{M}_i^e$. A notable feature of the global system is that the global boundary-boundary, $\boldsymbol{M}_b$, matrix is sparse and may be reordered to reduce the bandwidth using, for example, a Reverse Cuthill McKee algorithm [118], or re-factored in a multi-level Schur Complement solver as discussed in section 4.2.3.1. The global boundary-interior coupling matrix, $\boldsymbol{M}_c$, is very sparse but as it only operates on known vectors we only need to store the local matrices $\boldsymbol{M}_c^e$. Finally, the natural form of $\boldsymbol{M}_i$ is a block diagonal matrix which is very inexpensive to evaluate since each block may be inverted individually. It is the structure of $\boldsymbol{M}_i$ which makes the static condensation technique so effective. This arises from the fact that the interior modes are non-overlapping and are therefore orthogonal at an elemental level.

Now, if we distinguish between the boundary and interior components of $\boldsymbol{x}$ and $\boldsymbol{f}$ using $\boldsymbol{x}_b, \boldsymbol{x}_i$ and $\boldsymbol{f}_b, \boldsymbol{f}_i$, respectively, that is,

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_b \\ \boldsymbol{x}_i \end{bmatrix} \qquad \boldsymbol{f} = \begin{bmatrix} \boldsymbol{f}_b \\ \boldsymbol{f}_i \end{bmatrix},$$

then equation (4.92) can be written in its constituent parts as:

$$\begin{bmatrix} \boldsymbol{M}_b & \boldsymbol{M}_c \\ \boldsymbol{M}_c^T & \boldsymbol{M}_i \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_b \\ \boldsymbol{x}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_b \\ \boldsymbol{f}_i \end{bmatrix}. \tag{4.93}$$

To solve this system we perform a block elimination by pre-multiplying this system by the matrix

$$\begin{bmatrix} \boldsymbol{I} & -\boldsymbol{M}_c\boldsymbol{M}_i^{-1} \\ 0 & \boldsymbol{I} \end{bmatrix},$$

to arrive at:

$$\begin{bmatrix} \boldsymbol{M}_b - \boldsymbol{M}_c\boldsymbol{M}_i^{-1}\boldsymbol{M}_c^T & 0 \\ \boldsymbol{M}_c^T & \boldsymbol{M}_i \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_b \\ \boldsymbol{x}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_b - \boldsymbol{M}_c\boldsymbol{M}_i^{-1}\boldsymbol{f}_i \\ \boldsymbol{f}_i \end{bmatrix}. \tag{4.94}$$

The equation for the boundary unknowns is therefore:

$$(\boldsymbol{M}_b - \boldsymbol{M}_c\boldsymbol{M}_i^{-1}\boldsymbol{M}_c^T)\boldsymbol{x}_b = \boldsymbol{f}_b - \boldsymbol{M}_c\boldsymbol{M}_i^{-1}\boldsymbol{f}_i.$$

Once $\boldsymbol{x}_b$ is known we can determine $\boldsymbol{x}_i$ from the second row of equation (4.94) since

$$\boldsymbol{x}_i = \boldsymbol{M}_i^{-1}\boldsymbol{f}_i - \boldsymbol{M}_i^{-1}\boldsymbol{M}_c^T\boldsymbol{x}_b. \tag{4.95}$$

The solution of equation (4.92) has been split into three operations: The first is to evaluate and invert $\left[\boldsymbol{M}_b - \boldsymbol{M}_c\boldsymbol{M}_i^{-1}\boldsymbol{M}_c^T\right]$ which is also known as the *Schur complement*. The second is to evaluate $\boldsymbol{M}_i^{-1}$ and the final operation is the evaluation of $\boldsymbol{M}_c\boldsymbol{M}_i^{-1} = \left[\boldsymbol{M}_i^{-1}\boldsymbol{M}_c^T\right]^T$.

The second and third operations can both be performed at a local elemental level. Since $\boldsymbol{M}_i$ is made up of block diagonals of the local matrices $\boldsymbol{M}_i^e$ (i.e., $\boldsymbol{M}_i = \underline{\boldsymbol{M}_i^e}$) the inverse of $\boldsymbol{M}_i$ is

$$\boldsymbol{M}_i^{-1} = \left[\underline{\boldsymbol{M}_i^e}\right]^{-1} = \underline{[\boldsymbol{M}_i^e]^{-1}}$$

which is still a block diagonal matrix that can be evaluated locally within every element. The products $\boldsymbol{M}_c\boldsymbol{M}_i^{-1}\boldsymbol{f}_i$ and $\boldsymbol{M}_i^{-1}\boldsymbol{M}_c^T\boldsymbol{x}_b$ can also treated as local operations because they only involve the matrix-vector products of a known vector (that is, $\boldsymbol{f}_i$ and $\boldsymbol{x}_b$). To illustrate this operation in its matrix form we define the matrix $\boldsymbol{\mathcal{A}}_b$ which is the boundary version of $\boldsymbol{\mathcal{A}}$ and is equivalent to the bmap$[e][i]$ operation discussed in section 4.2.1. The operation $\boldsymbol{\mathcal{A}}_b$ therefore scatters the global boundary degrees of freedom to the local boundary degrees of freedom that is,

$$\begin{bmatrix} \boldsymbol{x}_b^1 \\ \boldsymbol{x}_b^2 \\ \vdots \\ \boldsymbol{x}_b^{N_{el}} \end{bmatrix} = \boldsymbol{\mathcal{A}}_b\boldsymbol{x}_b$$

where $\boldsymbol{x}_b^e$ contains the components of $\boldsymbol{x}_b$ in element e. Similarly, the operation $\boldsymbol{\mathcal{A}}_b^T$ assembles the global boundary degrees of freedom from the local boundary degrees of freedom. The boundary-interior matrix $\boldsymbol{M}_c$ can now be written as

$$\boldsymbol{M}_c = \boldsymbol{\mathcal{A}}_b^T\underline{\boldsymbol{M}_c^e}$$

and so the products $\boldsymbol{M}_c\boldsymbol{M}_i^{-1}\boldsymbol{f}_i$ and $\boldsymbol{M}_i^{-1}\boldsymbol{M}_c^T\boldsymbol{x}_b$ become

$$\boldsymbol{M}_c\boldsymbol{M}_i^{-1}\boldsymbol{f}_i = \boldsymbol{\mathcal{A}}_b^T\underline{\boldsymbol{M}_c^e}\,\underline{[\boldsymbol{M}_i^e]^{-1}}\boldsymbol{f}_i$$
$$\boldsymbol{M}_i^{-1}\boldsymbol{M}_c^T\boldsymbol{x}_b = \underline{[\boldsymbol{M}_i^e]^{-1}}\;\underline{\boldsymbol{M}_c^{eT}}\,\boldsymbol{\mathcal{A}}_b\boldsymbol{x}_b.$$

As both $\underline{[\boldsymbol{M}_i^e]^{-1}}$ and $\underline{\boldsymbol{M}_c^e}$ are essentially local matrices, these products can be evaluated on an elemental level with respect to a vector which is either globally scattered, as represented by $\boldsymbol{\mathcal{A}}_b$, or globally assembled, as represented by $\boldsymbol{\mathcal{A}}_b^T$. The boundary scatter and assembly operation also illustrates how to construct the boundary-boundary system since

$$\boldsymbol{M}_b = \boldsymbol{\mathcal{A}}_b^T\underline{\boldsymbol{M}_b}\boldsymbol{\mathcal{A}}_b.$$

Therefore, the Schur complement system may be written as:

$$\boldsymbol{M}_b - \boldsymbol{M}_c\boldsymbol{M}_i^{-1}\boldsymbol{M}_c^T = \boldsymbol{\mathcal{A}}^T\underline{\boldsymbol{M}_b^e}\boldsymbol{\mathcal{A}}_b - \boldsymbol{\mathcal{A}}_b^T\underline{\boldsymbol{M}_c^e}\,\underline{[\boldsymbol{M}_i^e]^{-1}}\,\underline{(\boldsymbol{M}_c^e)^T}\boldsymbol{\mathcal{A}}_b$$

$$= \boldsymbol{\mathcal{A}}_b^T \left[\underline{\boldsymbol{M}_b^e - \boldsymbol{M}_c^e[\boldsymbol{M}_i^e]^{-1}(\boldsymbol{M}_c^e)^T} \right] \boldsymbol{\mathcal{A}}_b$$

which shows that the global Schur complement system may be generated from the local elemental Schur complement system $\boldsymbol{M}_i^e - \boldsymbol{M}_c^e[\boldsymbol{M}_i^e]^{-1}(\boldsymbol{M}_c^e)^T$. We now appreciate that global assembly is only necessary for the boundary system when using static condensation. Once the boundary solution is known, the solution for the interior elemental modes given by equation (4.95) can be performed at an elemental level. In this formulation, we have assumed that all the global boundary conditions are unknown values since the Schur complement matrix is of size N_b. However, in general we have some known Dirichlet boundary conditions which can be dealt with by numbering the global system so that they are ordered after the unknown degrees of freedom. We then use the appropriate sub-matrix of $\boldsymbol{M}_i - \boldsymbol{M}_c[\boldsymbol{M}_i]^{-1}(\boldsymbol{M}_c)^T$ as shown in section 4.2.4.

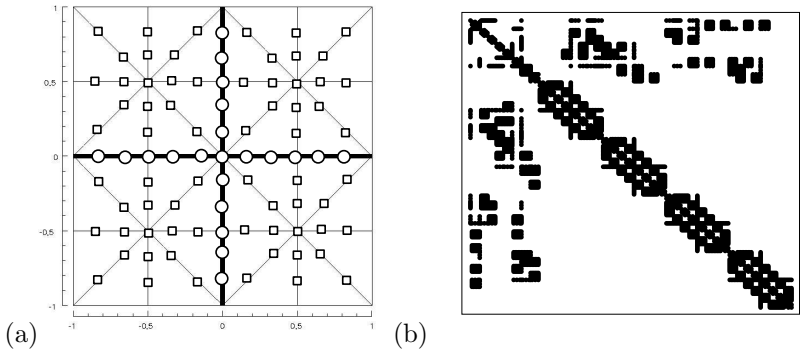
Since the majority of the storage requirement in this technique is used in the global Schur complement system, an alternative approach is to solve this system iteratively where only storage for the local Schur complements $\boldsymbol{M}_b^e - \boldsymbol{M}_c^e[\boldsymbol{M}_i^e]^{-1}(\boldsymbol{M}_c^e)^T$ is required. The Schur complement system is also better conditioned than the complete system which also makes this approach more attractive (see chapter 5).

As a final point we note that storage savings can also be made when evaluating the Laplacian and Mass matrix systems by noting that the matrices $(\boldsymbol{M}_i^e)^{-1}$ and $\boldsymbol{M}_c^e(\boldsymbol{M}_i^e)^{-1}$ are similar for straight-sided elements of the same size and orientation. However, to exploit this feature, some degree of structure in the mesh is required.

4.2.3.1 *Multi-Level Static Condensation*

The motivation behind using static condensation was the natural decoupling of the interior degrees of freedom within each element leading to a global system which contained a block diagonal sub-matrix. This decoupling can be mathematically attributed to the fact that the interior degrees of freedom in one element are orthogonal to the interior degrees of freedom of another simply because these modes are non-overlapping. To take advantage of this block diagonal sub-matrix we have to construct the Schur complement system $\boldsymbol{M}_b^e - \boldsymbol{M}_c^e[\boldsymbol{M}_i^e]^{-1}(\boldsymbol{M}_c^e)^T$ for the boundary degrees of freedom which may be evaluated locally.

The effect of constructing each of the local Schur complement matrices $\boldsymbol{M}_b^e - \boldsymbol{M}_c^e[\boldsymbol{M}_i^e]^{-1}(\boldsymbol{M}_c^e)^T$ is to othogonalise the boundary modes from the interior modes. However, the inverse matrix $[\boldsymbol{M}_i^e]^{-1}$ is typically full, which means that the boundary modes become tightly coupled. It is this coupling which dictates the bandwidth of the globally assembled Schur complement system. To compute the bandwidth we simply need to find the maximum difference between the global numbering of the boundary modes within every local element. Even though the boundary modes are coupled to all other boundary modes within the element and the boundary modes of neighbouring elements, they are not coupled with the boundary modes within non-neighbouring elements. Therefore, an appropri-



(a) (b)

Figure 4.19 The boundary degrees of freedom, on the mesh shown in plot (a), are ordered so that the boundary modes indicated by □ are first, followed by the boundary modes indicated by ∘ within each quadrant. Using this ordering the resulting Schur complement matrix has a block diagonal sub-matrix as shown in figure (b).

ate numbering of the boundary system will lead to a Schur complement matrix which also contains a sub-matrix that is block diagonal and so the static condensation technique can be reapplied. This technique has been more commonly used in the structural mechanics field and is also known as substructuring [440].

To illustrate this ordering we consider the triangular mesh shown in figure 4.19(a) using $N_{el} = 32$ elements. The construction of the global Schur complement $\boldsymbol{M}_b^e - \boldsymbol{M}_c^e[\boldsymbol{M}_i^e]^{-1}(\boldsymbol{M}_c^e)^T$ requires us to globally number all of the boundary degrees of freedom as indicated by the open circles (∘) and squares (□). We have not included the boundary of the domain as we assume that we are applying Dirichlet boundaries and so these values are not part of our Galerkin matrix system. If we order the numbering of the elemental boundary degrees of freedom so that the vertex and edge modes indicated by the open circles (∘) are first followed by the vertex and edge modes within each quadrant, indicated by the open squares (□), then the resulting Schur complement system of the mass matrix for a polynomial expansion of $p = 6$ is shown in figure 4.19(b). The block diagonal structure of the matrix is due to the fact that even after constructing the elemental Schur complement systems the boundary degrees of freedom in each quadrant do not overlap and so are orthogonal.

We can now construct another Schur complement system to solve for the (∘) degrees of freedom and decoupling each quadrant of (□) degrees of freedom. This technique can be repeated providing that there is more than one region of non-overlapping data. The approach is clearly independent of the elemental shape and may equally well be applied to quadrilateral regions or any hybrid shape in three dimensions.

4.2.4 Global Boundary System Numbering and Ordering to Enforce Dirichlet Boundary Conditions

Implementation note: *Generation of a global numbering scheme and ordering of the boundary numbering to strongly enforce Dirichlet conditions in a matrix system.*

We have previously discussed in section 4.2.1 the issues relating a local elemental numbering scheme to a global numbering taking account of inter-element connectivity. The aim of section 4.2.1 was to assemble the local expansions into a global C^0 continuous expansion where a global numbering scheme map$[e][i]$ or bmap$[e][i]$ was assumed to be known. In this section we outline the generation of this numbering scheme and then demonstrate how the numbering scheme can be re-ordered to enforce Dirichlet boundary conditions.

Heuristically a global numbering scheme for the boundary can be generated in the following manner. We assume as a starting point the output of a finite element or volume mesh generator where the global Cartesian coordinates of each vertex are known. We start our numbering scheme by assigning a unique number to every unique vertex defined through its global coordinates. In the spectral/*hp* element method we also require a global numbering of all boundary degrees of freedom including all edge modes and the face modes in three–dimensions. Using the global vertex coordinates every unique edge can be identified using the global coordinates of the vertices at the ends of each edge. The global numbering of the degrees of freedom along each global edge can then be defined. An example of this type of global numbering is shown in figure 4.20(a) where we note that the global vertices are numbered first followed by the global edges according to the element numbering (given in figure 4.20(c)). A similar strategy can also be followed to number global degrees of freedom associated with every face. If a full numbering scheme is required the interior degrees of freedom can finally be independently numbered by looping over all elements. This type of global numbering is unlikely to lead to an ordering which will give a minimal bandwidth for global matrix problems. However, once one global numbering has been obtained it is then possible to use standard algorithms and packages to reorder the scheme to minimise the difference in numbers between coupled degrees of freedom in order to reduce bandwidth or maximise matrix infill [265, 118].

It may also be necessary to enforce Dirichlet boundary conditions. In this case it is advantageous to have a global numbering scheme which orders the unknown boundary degrees of freedom first followed by the known degrees of freedom which lie on Dirichlet boundaries.

A procedure to perform the boundary reordering adopted by Henderson (Ph.D. thesis) is illustrated in figure 4.20. In this example we consider a mesh of four triangular elements and start by assuming that we have obtained a global numbering scheme as shown in figure 4.20(a) for a $P = 3$ polynomial expansion within each element. From the numbering in figure 4.20(a) we can define elemental arrays bmap$[e][i]$ which relates all the element boundary degrees of freedom to the global numbering scheme. This type of array was used in section 4.2.1 to define the global assembly operation. In the example of figure 4.20 if we assume that the local vertex boundary degrees of freedom are ordered first followed by

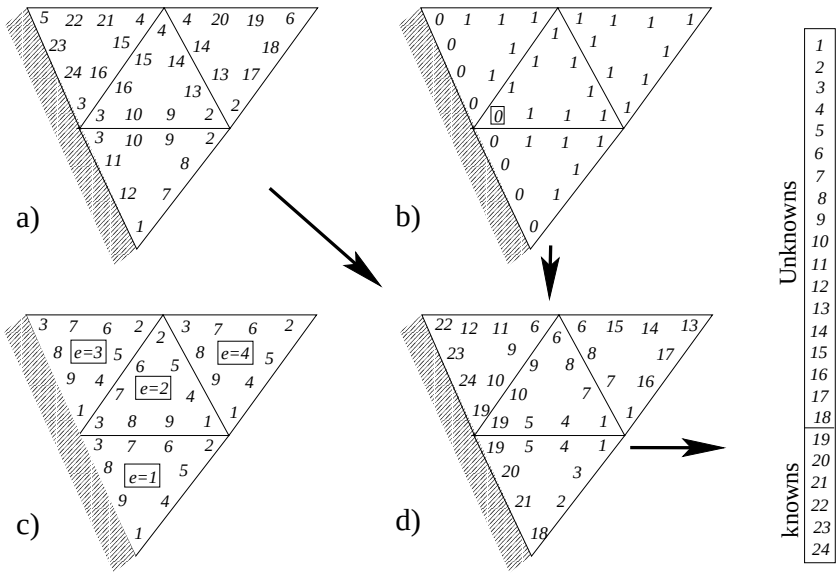


Figure 4.20 Given a global numbering scheme as shown in (a), we can re-order the system so that known degrees of freedom on a Dirichlet boundary are listed after the unknown degrees of freedom as shown in (d). A reordering strategy using the elemental information, as given in (c), is to generate a global mask array as shown in (b).

the edge degrees of freedom in each element, as shown in figure 4.20(c) then the array bmap$[e][i]$ can be defined as $(1 \leq i \leq 9)$

$$\begin{aligned}
&\text{bmap}[1][i] \Rightarrow [1, 2, 3, 7, 8, 9, 10, 11, 12],\\
&\text{bmap}[2][i] \Rightarrow [2, 4, 3, 13, 14, 15, 16, 10, 9],\\
&\text{bmap}[3][i] \Rightarrow [3, 4, 5, 16, 15, 21, 22, 23, 24],\\
&\text{bmap}[4][i] \Rightarrow [2, 6, 4, 17, 18, 19, 20, 14, 13].
\end{aligned}$$

If we assume that the boundary is of a Dirichlet type then we would like to place all the global numbers along this boundary at the end of a new global numbering system. Although this would appear to be straightforward in the example, devising an automatic implementation using elemental information is a bit more involved and a suggested implementation is provided below.

Implementation note: *Manipulation of the global numbering scheme using elemental mapping arrays.*

From the point of view of implementation, a convenient way of specifying boundary conditions, in two-dimensions, is to identify which local *edges* of the elements touch the given domain Dirichlet boundary. It is then possible to identify the degrees of freedom along these edges which have Dirichlet values and associate with them an elemental mask array 'mask$[e][i]$', as shown in figure 4.20(b). This mask array is set so that all Dirichlet degrees of freedom have entries set to "0" otherwise the entry is set to "1." If we are imposing Dirichlet boundary conditions using only edges information in two–dimensions (or faces information in three–dimensions) care must be taken to ensure that all local contributions to the mask array in elements that only touch the Dirichlet boundary

through a vertex are set. For example, setting the values of mask$[e][i]$ in elements with edges that touch the Dirichlet boundary (i.e., $e = 1, 3$) in figure 4.20 will not enforce the single vertex touching the boundary in element 2 (highlighted by a box) to be zero. To ensure the mask array of this vertex point is set correctly we can use a global assembly and scatter operation using the bmap$[e][i]$ array similar to that discussed in section 4.2.1.1. In this case, we first initialise a global array, Gmask$[i]$ to have a value of 1 on all entries and then perform the assembly:

```
Do e = 1, N_el
    Do i = 0, n_b[e] − 1
        Gmask[bmap[e][i]] = Gmask[bmap[e][i]] × mask[e][i]
    continue
continue.
```

The global array Gmask$[i]$ will now contain entries of 1 only when all vertex points have a local mask which is 1. The local mask array can then be recovered through scatter operation

```
Do e = 1, N_el
    Do i = 0, n_b[e] − 1
        mask[e][i] = Gmask[bmap[e][i]]
    continue
continue
```

and mask$[e][i]$ will then correspond to figure 4.20(b).

Finally, using the local mask array we can reorder the global numbering system using elemental information and another global array. We now initialise a global array Gbmap$[i]$ to zero and fill this array using the mask$[e][i]$ array and applying the following logic:

```
Let n_1 = n_2 = 1
Do e = 1, N_el
    Do i = 0, n_b[e] − 1
        if (Gbmap[bmap[e][i]] = 0)
            if (mask[e][i] = 1)
                Gbmap[bmap[e][i]] = n_1
                n_1 = n_1 + 1
            else
                Gbmap[bmap[e][i]] = n_2 + N_bslv
                n_2 = n_2 + 1
    continue
continue
```

where N_{bslv} is the number of unknown boundary degrees of freedom which typically has to be obtained as part of the loop and then added afterwards. The final form of the reordered global numbering scheme can be recovered into bmap$[e][i]$ using a scatter operation of the form

```
Do e = 1, N_el
    Do i = 0, n_b[e] − 1
        bmap[e][i] = Gbmap[bmap[e][i]]
    continue
continue.
```

The array bmap$[e][i]$ now corresponds to the ordering in figure 4.20(d). If we make a global assembly with the new bmap$[e][i]$ we obtain a global ordering array of the form indicated in the right-hand-side of figure 4.20 where the unknown degrees of freedom are listed first followed by the known Dirichlet values.

4.2.4.1 *Discrete Lifting of the Known Solution*

We recall that in a standard Galerkin implementation Dirichlet boundary conditions can be enforced by "lifting" a known solution satisfying these boundary conditions, see section 2.2.1.3. Performing this operation leaves us with a homogeneous Dirichlet boundary problem where the same test and trial space can be applied to the problem. The separation of the global solution arrary $\hat{\boldsymbol{u}}$ into known (i.e., Dirichlet) and unknown boundary degrees of freedom provides a way of obtaining a discrete lifted boundary solution. If we denote the homogeneous unknown solution by $u^{\mathcal{H}}(\boldsymbol{x})$ and the known Dirichlet boundary conditions by $u^{\mathcal{D}}(\boldsymbol{x})$ we can decompose the solution $u^{\delta}(\boldsymbol{x})$ into the form

$$u^{\delta}(\boldsymbol{x}) = u^{\mathcal{H}}(\boldsymbol{x}) + u^{\mathcal{D}}(\boldsymbol{x}) = \sum_{j}^{N^{\mathcal{H}}} \hat{u}_j^{\mathcal{H}} \Phi_j(\boldsymbol{x}) + \sum_{j=N^{\mathcal{H}}+1}^{N_{dof}^{b}} \hat{u}_j^{\mathcal{D}} \Phi_j(\boldsymbol{x}),$$

where we recall that $\Phi_j(\boldsymbol{x})$ is the global expansion basis and $N^{\mathcal{H}}$ is defined as the number of global homogeneous boundary degrees of freedom ($N^{\mathcal{H}} = 17$ in the example of figure 4.20). We note that the important component of the definition is that the homogeneous solution has zero contribution from modes which are non-zero on Dirichlet boundaries. The homogeneous solution also contains the interior degrees of freedom which are defined to be zero on the boundaries. For the lifted solution the values of $\hat{u}_j^{\mathcal{D}}$ can be determined using a boundary transformation as discussed in section 4.3.2. In general, the lifted solution $u^{\mathcal{D}}(\boldsymbol{x})$ may also contain any predefined contribution associated with any homogeneous mode. This is typically possible for unsteady problems with steady boundary conditions where the previous solution that satisfies boundary conditions can be used as the lifted solution of the next time step.

4.2.4.2 *Boundary Matrix Manipulation*

As a final example of the application of both the boundary numbering system and the lifted solution expressed in terms of expansion coefficients we consider the matrix solution shown in figure 4.21. If we wish to directly invert a matrix problem $\boldsymbol{M}$ arising from our spectral/hp element formulation we can apply the static condensation technique discussed in section 4.2.3. By construction this

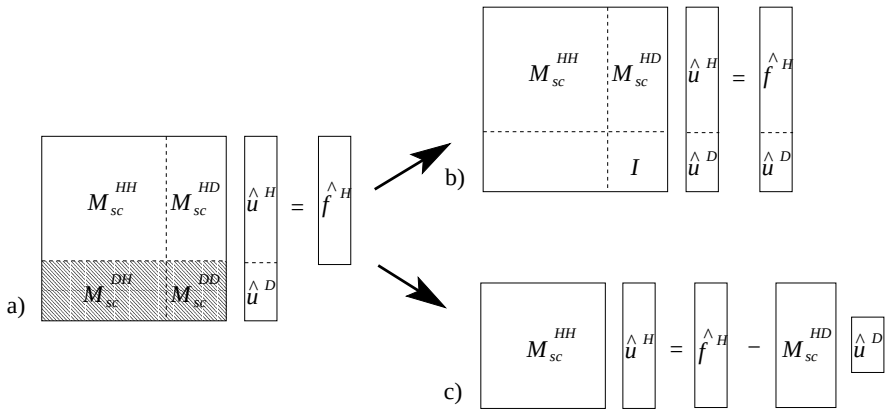


Figure 4.21 Sorting the boundary vector $\hat{\boldsymbol{u}}$ into an unknown component $\hat{\boldsymbol{u}}^{\mathcal{H}}$ and a known (lifted) component $\hat{\boldsymbol{u}}^{\mathcal{D}}$ the full boundary matrix system $\boldsymbol{M}_{SC}$ can be viewed as sub-matrices $\boldsymbol{M}_{SC}^{\mathcal{H}\mathcal{H}}, \boldsymbol{M}_{SC}^{\mathcal{H}\mathcal{D}}, \boldsymbol{M}_{SC}^{\mathcal{D}\mathcal{H}}$, and $M_{SC}^{\mathcal{D}\mathcal{D}}$ as shown in plot (a). This can be solved by removing matrices $\boldsymbol{M}_{SC}^{\mathcal{D}\mathcal{H}}$, and $\boldsymbol{M}_{SC}^{\mathcal{D}\mathcal{D}}$ with the identity matrix as shown in (b) or equivalently by taking the product $\boldsymbol{M}_{SC}^{\mathcal{H}\mathcal{D}}\hat{\boldsymbol{u}}^{\mathcal{D}}$ to the right-hand side as shown in plot (c).

technique decouples the interior degrees of freedom, and for a symmetric matrix requires the inversion of a global boundary matrix of the form $\boldsymbol{M}_{SC} = \boldsymbol{M}_b - \boldsymbol{M}_c\boldsymbol{M}_i^{-1}\boldsymbol{M}_c^T$ where $\boldsymbol{M}_b, \boldsymbol{M}_i$ and $\boldsymbol{M}_c$ are submatrices of $\boldsymbol{M}$. For the standard Galerkin formulation $\boldsymbol{M}_{SC}$ can be constructed from its elemental contributions $\boldsymbol{M}_{SC}^e$ and by applying the global assembly technique over the boundary degrees of freedom, such that $\boldsymbol{M}_{SC} = \boldsymbol{\mathcal{A}}_b^T\underline{\boldsymbol{M}}_{SC}^e\boldsymbol{\mathcal{A}}_b$ as discussed in section 4.2.1.1. We note, however, that the matrix $\boldsymbol{M}_{SC}$ constructed in this manner contains both the Dirichlet and unknown degrees of freedom. Strictly speaking, the matrix does not correspond to the Galerkin problem since it contains test (or weight) functions which are now zero on Dirichlet boundary conditions. The application of a boundary numbering scheme where known Dirichlet values are listed after the unknown boundary degrees of freedom allows us to manipulate the resulting matrix into the appropriate form.

This process is highlighted in figure 4.21. The full boundary matrix $\boldsymbol{M}_{SC} = \boldsymbol{\mathcal{A}}_b^T\underline{\boldsymbol{M}}_{SC}^e\boldsymbol{\mathcal{A}}_b$ is indicated in figure 4.21(a) where the numbering system allows us to view the matrix as a series of sub-matrices $\boldsymbol{M}_{SC}^{\mathcal{H}\mathcal{H}}, \boldsymbol{M}_{SC}^{\mathcal{H}\mathcal{D}}, \boldsymbol{M}_{SC}^{\mathcal{D}\mathcal{H}}, \boldsymbol{M}_{SC}^{\mathcal{D}\mathcal{D}}$ corresponding to the homogeneous and Dirichlet solutions $\hat{\boldsymbol{u}}^{\mathcal{H}}$ and $\hat{\boldsymbol{u}}^{\mathcal{D}}$, respectively. The highlighted matrices $\boldsymbol{M}_{SC}^{\mathcal{D}\mathcal{H}}$ and$\boldsymbol{M}_{SC}^{\mathcal{D}\mathcal{D}}$ are not part of the Galerkin problem since they involve trial (or weight) functions which are not zero at Dirichlet boundary conditions. To enforce the Dirichlet boundary conditions we therefore have two choices. As shown in figure 4.21(b) we can replace the matrices $\boldsymbol{M}_{SC}^{\mathcal{D}\mathcal{H}}$ and $\boldsymbol{M}_{SC}^{\mathcal{D}\mathcal{D}}$ with a zero block and the identity matrix $\boldsymbol{I}$ on the diagonal entries as well as copy the vector $\hat{\boldsymbol{u}}^{\mathcal{D}}$ to the right-hand-side. An equivalent technique is often applied in finite element methods where this approach is favoured since it

is not necessary to reorder the degrees of freedom as a row containing a Dirichlet degree of freedom can be deleted and a unit value placed on the diagonal. There are, however, two potential drawbacks. The first is that the matrix is now not symmetric even if the original matrix $\boldsymbol{M}_{SC}$ was symmetric. The second is that the conditioning of the system can be significantly influenced by the introduction of diagonal terms if all other entries of the system are very small.

Alternatively, we can manipulate the matrix system as shown in figure 4.21(c). This system is the same as shown in figure 4.21(a) and (b) but the known Dirichlet boundary conditions $\hat{\boldsymbol{u}}^{\mathcal{D}}$ have now been taken to the right-hand-side of the system. In this manipulation we have maintained the symmetry of the system $\boldsymbol{M}_{SC}^{\mathcal{H}\mathcal{H}}$ by essentially lifting the known solution out of the problem. To apply this technique, however, a boundary numbering system is clearly necessary to construct a compact form of the matrix $\boldsymbol{M}_{SC}^{\mathcal{H}\mathcal{H}}$.

We close by noting that a more general lifted solution, $u^{\mathcal{D}}$, can be used involving all the boundary degrees of freedom. In this case a matrix contribution involving $\boldsymbol{M}_{SC}^{\mathcal{H}\mathcal{H}}$ will appear on the right-hand-side of the problem. We remark, however, that right-hand-side matrix–vector products do not require the assembly of global matrix systems since they can (and should) be evaluated at an elemental level and then assembled.

4.3 Pre- and Post-Processing Issues

In section 4.1 we discussed the key elemental operations used in a spectral/*hp* element solver independent of the formulation, namely integration, differentiation and basic elemental mappings. We then applied these operations in section 4.2 within the context of global C^0 expansion for a classical Galerkin formulation by introducing the concept of global assembly. Although these two sections represent the operations behind the algorithms, practical implementation also requires various pre- and post-processing techniques, the discussion of which will conclude this chapter.

In this section we will discuss issues relevant to mesh generation and boundary representation as well as particle tracking within a spectral/*hp* element computational setting. The first two topics, boundary condition discretisation and mesh generation, are important issues of any spectral/*hp* element formulation. The last topic, particle tracking, is a useful diagnostic tool in fluid mechanics as well as a key component in the strong semi-Lagrangian formulation, which is discussed in chapters 6 and 8.

4.3.1 Boundary Condition Discretisation

Until now we have assumed that all boundary conditions have been specified in terms of the expansion coefficients (whether modal or nodal), however, for a general implementation this is not typically the case. In section 4.1.3 we discussed how curved elements, which are typically due to a curved boundary of the solution domain, can be represented in terms of an isoparametric mapping if the mapping of the edges, or faces in three-dimensions, are provided. In both the case of the Dirichlet boundary condition and the curved boundary mapping

we need to project the known function onto a discrete expansion basis which is also globablly C^0 continuous.

In this section we will discuss how to generate C^0 approximations over the whole surface using local elemental information. Although we will focus on the Dirichlet boundary condition, an analagous approach can be used in the case of a curved surface mapping to define the elemental transformation.

4.3.2 Elemental Boundary Transformation

Given a Dirichlet boundary condition $g_{\mathcal{D}}(\boldsymbol{x})$ where $\boldsymbol{x} \in \partial\Omega$ we need a consistent method of approximating the boundary condition in terms of our discrete expansion. In two-dimensions the boundary of the domain is simply a one-dimensional segment but in three-dimensions, depending on our spatial discretisation, the surface becomes a tessellation of triangles or rectangles or even a combination of both shapes.

In general, we require that our discrete boundary approximation remains at least C^0 continuous. This condition could be ensured by formulating an approximation over the whole Dirichlet boundary. In two-dimensions this would only involve a one-dimensional problem with multiple elemental segments. In a three-dimensional problem, however, this implies constructing a full two-dimensional system. It can be appreciated that for a general mesh this could become excessively complicated. A more desirable method is to perform a local projection within each element. However, if we performed an elemental Galerkin (L^2) projection of $g_{\mathcal{D}}$ onto the boundary modes within each individual element we could not, in general, guarantee that our approximation would be C^0 continuous over the whole boundary.

We recall that elemental projections or forward transformations were previously discussed in section 4.1.5.3. We note that the use of a collocation projection onto a set of nodal points which include the boundary of the segment or face satisfied our requirement of a C^0 continuous approximation. An appropriate choice of collocation points is therefore the Gauss-Lobatto-Legendre quadrature points in a segment or rectangular region and any choice of the nodal non-tensorial distribution discussed in section 3.3 on a triangular face. As we have seen previously, these points also have favourable interpolation properties from a Lebesgue constant point of view. Nevertheless, if a local Galerkin projection is desired we need to modify the Galerkin projection to ensure C^0 continuity. In two–dimensions this modified projection can be viewed as a collocation projection at the vertices followed by an L^2 projection on all edges with a final interior L^2 projection of the reamining function.

To illustrate the modified Galerkin projection we consider the case shown in figure 4.22. We wish to project a known boundary condition $g_{\mathcal{D}}$ onto a boundary of element 'e' lying on along $\partial\Omega$. The discrete solution $u^\delta(x_1, x_2)$ along the edge can be written as

$$u^\delta(x_1, x_2) = \sum_{pq} \hat{u}^e_{pq} \phi^e_{pq}(\xi_1, -1) = \sum_{p=0}^{P_1} \hat{u}^e_{p0} \phi^e_{p0}(\xi_1, -1),$$

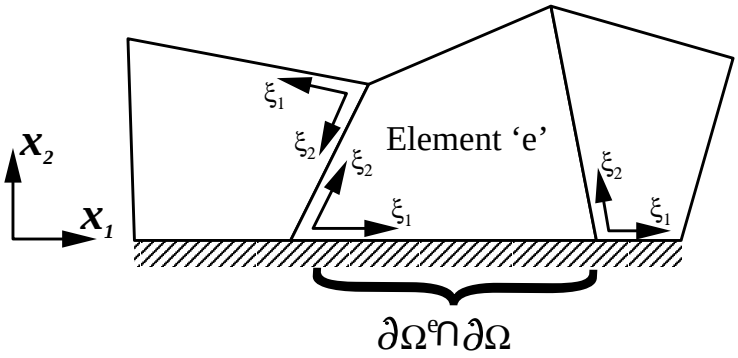


Figure 4.22 Intersection of the boundary of the e-th element with the domain boundary $\partial\Omega$. The orientation of the local coordinates is indicated by the (ξ_1, ξ_2) system.

where the reduction of the summation is due to the boundary interior decomposition of the basis and $x_1 = \chi_1^e(\xi_1, -1), x_2 = \chi_2^e(\xi_1, -1)$. If our expansion is of a modified modal type then $\phi_{p0}(\xi_1, -1) = \psi_p^a(\xi_1)\psi_0^a(-1) = \psi_p^a(\xi_1)$. Alternatively, if the expansion is of a nodal type then $\phi_{p0}(\xi_1, -1) = h_p(\xi_1)$.

For the modal expansion case we would like to determine $\hat{u}_{p0}^e$ such that

$$\sum_{p=0}^{P_1} \hat{u}_{p0}^e \psi^a(\xi_1) \simeq g_{\mathcal{D}}(\chi_1^e(\xi_1, -1)).$$

We can use the vertex-edge decomposition of the boundary modes to ensure that our approximation remains C^0 continuous over all elements. The vertex functions have, by definition, a unit value at the ends of an edge and all other boundary modes are zero at this point. C^0 continuity is, therefore, ensured if we set the vertex coefficient to

$$\begin{aligned} \hat{u}_{00}^e &= g_{\mathcal{D}}(\chi_1^e(-1, -1)) \\ \hat{u}_{P_1 0}^e &= g_{\mathcal{D}}(\chi_1^e(1, 1)). \end{aligned}$$

[Recall that $p = 0$ and $p = P_1$ refer to the vertex modes of the edge expansions $\psi^a(\xi)$ and $h_p(\xi)$]. We also note that $\chi_1^e(-1, -1), \chi_1^e(1, -1)$ are simply the values of the vertex location (x_1, x_2). The remaining unknown coefficients at the boundary may now be written as

$$\sum_{p=1}^{P_1 - 1} \hat{u}_{p0}^e \psi^a(\xi_1) \simeq g_{\mathcal{D}}(\chi_1^e(\xi_1, -1)) - \hat{u}_{00}^e \psi_0^a(\xi_1) - \hat{u}_{P_1 0}^e \psi_{P_1}^a(\xi_1).$$

Since the remaining modes do not contribute to the end-points we can solve for the remaining unknowns $\hat{u}_{p0}^e$ $(1 \leq p \leq P_1)$ without destroying the C^0 continuity. These coefficients can be found by setting up the local Galerkin projection:

Find $\hat{u}^e_{p0}$ *such that*

$$(\phi_i, \sum_{p=1}^{P_1-1} \hat{u}^e_{p0}\phi_p) = (\phi_i, g_{\mathcal{D}} - \hat{u}^e_{00}\psi^a_0 - \hat{u}^e_{P_1 0}\psi^a_{P_1}),$$

for all i *(*$0 \leq i \leq P_1 - 1$*).*

Clearly, this involves constructing and inverting the one-dimensional mass matrix $\boldsymbol{M}^{1D}[i][j] = (\phi_i(\xi_1), \phi_j(\xi_1))$ for $(1 \leq i, j \leq P_1)$. This Galerkin approximation minimises the error in the L^2 norm between the exact boundary condition and the edge approximation in the interior of the region (for a more detailed analysis, see 7.4 and also Babuška et al. [25]). In a discrete implementation of the projection we will be performing the collocation projection onto the quadrature points. However, if this collocation projection is of a sufficiently high order (i.e., the number of quadrature points is large enough) then the error in the collocation projection will be smaller than that of the L^2 projection. A directly analogous procedure can be followed for triangular regions as the boundary modes of the modified expansion are identical (see section 3.2.3.3). Although we can also use the same technique when using the spectral element nodal expansion $h_p(\xi_1)$, we recall that if we evaluate the one-dimensional mass matrix $\boldsymbol{M}^{1D}$ using a discrete inner product $(u, v)_\delta$ of the same order as $h_p(\xi_1)$ then the mass matrix is diagonal and we are essentially performing a collocation projection, see sections 2.3.4.2 and 4.1.5.3. In this case the formulation is, therefore, identical to evaluating the boundary conditions at the nodal points.

For the three-dimensional case we follow an analogous procedure using three steps:

1. Set the expansion coefficients of the vertex modes to the value of $g_{\mathcal{D}}$ evaluated at the vertex points.
2. Subtract the vertex mode approximation from $g_{\mathcal{D}}$ and evaluate the coefficients of the edge modes using a local one-dimensional Galerkin projection within the interior of the edge.
3. Subtract the vertex and edge mode approximation from $g_{\mathcal{D}}$ and determine the face modes using a local two-dimensional Galerkin projection within the interior of the face.

The three-dimensional modified transformation for a triangular face is represented diagrammatically in figure 4.23. Using the collocation property of the vertex modes, we set the coefficient of the vertex mode to the physical value of the function evaluated at the vertex as indicated in figure 4.23(a). Assuming that the boundary function is continuous, this ensures continuity of the vertex functions over the entire boundary. To calculate the remaining expansion coefficients we subtract the vertex mode approximation from the boundary function as indicated in figure 4.23(b). We then locally project each of the edge functions onto the edge modes using a one-dimensional Galerkin approximation. As we have subtracted the vertex contribution, the function is zero at the end-points

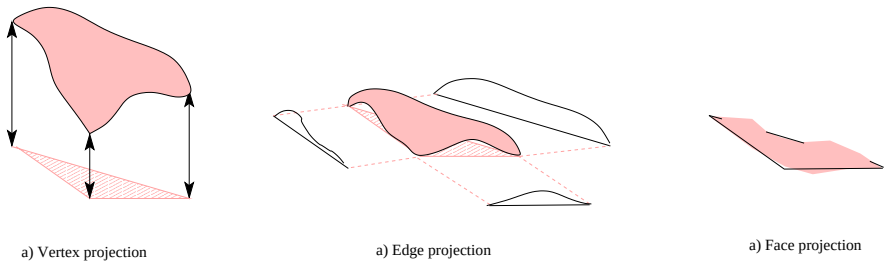


Figure 4.23 Diagrammatic representation of three-dimensional modified boundary transformation within a triangular face. Given an initial function as shown in figure (a) the vertex coefficients are set to the value of the function at the vertices. This approximation is then subtracted as shown in (b) permitting the edge contributions to be locally projected. Finally, the edge approximation is also subtracted and the remaining function is projected onto the face modes as indicated in (c).

of an edge which is consistent with the shape of the edge modes. A possible source of error in evaluating the edge modes between two elements is in the numerical integration. However, if we use a symmetric quadrature rule, such as Gauss-Lobatto-Legendre quadrature, this error will be identical for any smooth function providing the same quadrature order is used and so we maintain C^0 continuity. Finally, we subtract the edge and vertex modes approximation from the boundary function and project this onto the face modes as shown in figure 4.23(c). This function will not, in general, be exactly zero along all edges since there is an error associated with the approximation of the edge functions. However, for a smooth boundary function the error will be consistent with the overall approximation.

4.3.3 Mesh Generation for Spectral/*hp* Element Discretisation

In this section we discuss issues concerning the generation of high-order or curvilinear meshes for spectral/*hp* element discretisations of complex geometries. The extension of standard mesh generation technology for spectral/*hp* element algorithms is a non-trivial exercise. Complications arise due to the conflicting requirements of generating coarse meshes for high-order polynomial approximations whilst maintaining good elemental properties in regions of high curvature. A potential problem is illustrated in figure 4.24 that shows the occurrence of invalid curvilinear spectral/*hp* elements in reconstructing an arterial bypass graft [435]. The straight-sided element discretisation does not include any invalid elements.

A necessary starting point to discussing the issues involved in mesh generation is to define what makes the elements invalid. In the case shown in figure 4.24 the elements are invalid because the mapping between the physical region $\boldsymbol{x} \in \Omega^e$ and the standard region $\boldsymbol{\xi} \in \Omega_{st}$ is not bijective. This is highlighted by the fact that the Jacobian of the mapping $x_i = \chi_i(\xi_1, \xi_2, \xi_3)$ for $i = 1, 2, 3$ is singular. A valid element can therefore be defined as an element where the Jacobian of the mapping is strictly positive, i.e.

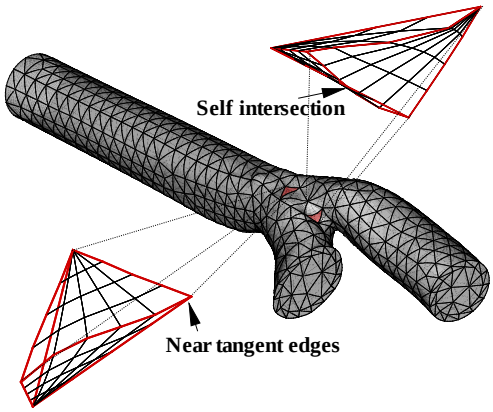


Figure 4.24 Generation of high-order curvilinear elements from coarse mesh. The deformation of straight sided element can lead to invalid elements being generated with singular mappings.

$$J^e(\boldsymbol{\xi}) = \left| \frac{\partial \chi_i^e}{\partial \xi_j} \right| > 0 \quad \forall \quad \chi_i^e(\boldsymbol{\xi}) \in \Omega^e, \boldsymbol{\xi} \in \Omega_{st}$$

From a purely implementation point of view, differentiation in an arbitrary element, as discussed in section 4.1.3.4, involves derivatives with respect to the local Cartesian coordinates divided by the Jacobian. A zero Jacobian therefore implies an infinite derivative, making the algorithm and element description invalid.

In general, we would like our mesh generation algorithm to take into account the possibility of generating invalid elements during the mesh construction process. However, such an approach is typically too complicated since it requires an algorithm to take into account all possible invalid generation scenarios. An alternative more practical strategy is, therefore, to initially design a coarse mesh of straight-sided elements using as much information of the surface topology as possible. Subsequently the coarse straight-sided mesh is deformed to conform to the curved boundary representation of the solution domain. Whilst a mesh of straight-sided elements may consist of purely valid elements, the deformation of these elements into curvilinear approximations can generate invalid elements as shown in figure 4.24. Therefore, we need to employ strategies to minimise element the generation of invalid elements such as curvature driven mesh refinement, interior edge and face deformation, and use of hybrid shape expansions.

In section 4.3.4 we outline the basic concepts behind the geometry representation and mesh generation techniques for constructing a coarse mesh of straight-sided elements with vertices that conform to the geometry boundary. In section 4.3.5 we then discuss how to deform the meshes by constructing local mappings to ensure boundary conforming curvilinear elements in two- and three–dimensions.

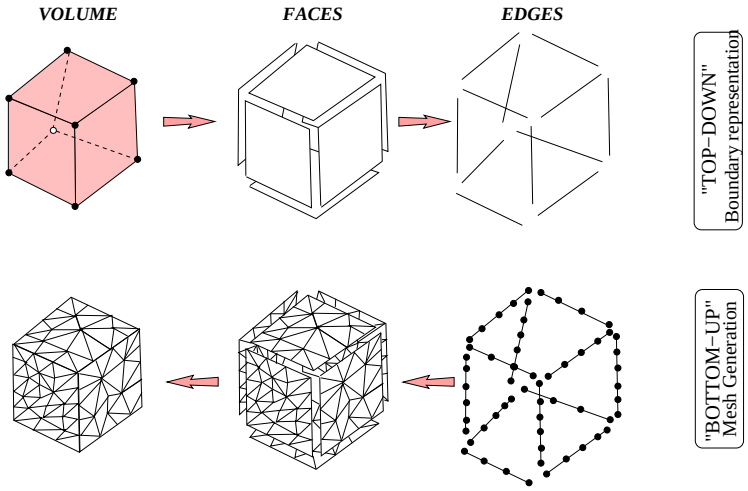


Figure 4.25 Illustration of the "top-down" boundary representation (B-Rep) of a domain for use in a "bottom-up" mesh generation strategy (Courtesy of J. Peiró)

4.3.4 Global Coarse Meshing

To understand the problem of curvilinear mesh generation we must first outline standard mesh generation techniques. We start by assuming that the geometry of the computational domain is defined through a *top-down* boundary representation. We will implicitly assume that the "real" surface can adequately be represented by this boundary representation which we subsequently treat as our exact definition of the solution domain boundary. As shown in figure 4.25 the concept of a top-down representation arises from the view point that a volume is enclosed by a series of faces, which are themselves enclosed by a set of curves, which in turn are enclosed between two points. To ensure a complete boundary description each of these surfaces may only intersect one another along curves and curves may only intersect one another at boundary points. Typically, these curves and surfaces are described using standard techniques of computer aided design (CAD).

Many standard mesh generation techniques, such as advancing front, structured meshing and Delaunay [464], follow a "bottom-up" construction procedure. We will not discuss the specific details of the different types of mesh generation but will simply outline the broad common strategy that all these techniques follow. For details on each of the different types of mesh generation techniques we point the reader to [464] and the references therein. As illustrated in figure 4.25, the "bottom-up" approach initially discretises the edges of the boundary representation into discrete segments which conform to the points of the boundary representation. Every surface of the boundary representation is then bounded by a set of discretized edges and so the next step of the generation is to develop a surface discretisation in terms of triangular or quadrilateral elements. Finally, the generation process is completed by constructing elements in the interior of

the domain which comply to the face and edge definitions constructed in the previous steps. This way it is possible to construct a discretisation which conforms to the boundary requirements. To ensure a similar condition for curvilinear elements using spectral/*hp* element expansion we will follow a similar bottom-up strategy.

We note that different types of interior/volume discretisation are also possible. An example is shown in figure 4.24 where a *boundary layer mesh* has been employed that produces a layer of high aspect ratio elements in a structured fashion adjacent to the geometry boundary. This type of discretisation can be very useful in viscous flows where boundary layers naturally occur as part of the physical solution. In this context, we are using some *a priori* information about the solution to dictate our meshing strategy. Although local refinement such as the boundary layer meshing is useful from an approximation point of view, it can also produce more invalid curvilinear elements.

4.3.5 High-Order Mesh Generation

We are now faced with the task of generating a curvilinear and boundary conforming mesh from a coarse mesh of straight elements. This coarse mesh contains vertices which conform to the points, curves and faces of the boundary representation. The process of high-order mesh generation is to determine an elemental boundary transformation, in a bottom-up fashion. Having defined the boundary transformation, an elemental mapping from the standard region Ω_{st} can be defined as discussed in section 4.1.3.2.

From our coarse mesh we know the location of the vertex points. The next stage is, therefore, to ensure that the edges of our element in two- and three-dimensions conform to the boundary representation as discussed in section 4.3.5.1. Subsequently, we can then define edges on the interior of the domain in two–dimensions as outlined in section 4.3.5.2. In three-dimensions, the interior edge evaluation is similar to the problem of determining edges lying within a boundary face. Finally, in three-dimensions we also need to determine the interior face location. These three-dimensional construction issues are discussed in section 4.3.5.3.

4.3.5.1 *Boundary Constrained Edges in Two–Dimensions*

The mappings involved in the definition of an edge constrained to a boundary segment are illustrated in figure 4.26. In general, we are seeking an elemental mapping $\chi_i(\xi_1, \xi_2)$ for $i = 1, 2$ which defines the transformation from the boundary conforming element to the standard region Ω_{st}. To define $\chi_i(\xi_1, \xi_2)$, however, we require the one-dimensional mapping $\chi_i^{1D}(\xi_1)$ which transforms the boundary conforming edge of the physical element to one side of the standard region $-1 \leq \xi_1 \leq 1$. This mapping combined with similar mapping for the other edges would allow us to define $\chi_i(\xi_1, \xi_2)$, as discussed in section 4.1.3.2 (note that $\chi_i^{1D}(\xi)$ was denoted as $f_i(\xi)$ in that section). To determine $\chi_i^{1D}(\xi)$ we need some information about the surface which is typically defined in terms of another transformation $\mathbf{g}(u) = [g_1(u), g_2(u)]$. For example, if the boundary edge is

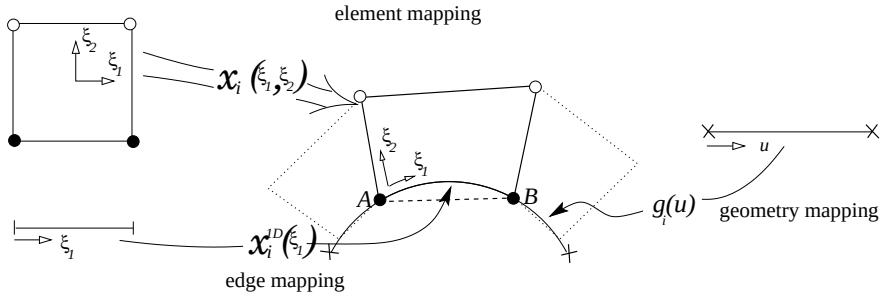


Figure 4.26 Mappings involved in the generation of a curvilinear element. The mapping $\chi_i(\xi_1, \xi_2)$ defines the element to standard region transformation. This transformation requires the definition of the edge mapping $\chi_i^{1D}(\xi_1)$. Normally, a parametric mapping $g_i(u)$ will also exist which defines the geometry boundary.

a circle, the mapping can be defined as $g_1(u) = R\cos(u), g_2(u) = R\sin(u)$ where u is the azimuthal angle and R is the circle radius. Alternatively $\boldsymbol{g}(u)$ might be expressed as a spline representation in terms of the local coordinate u. Typically, these geometrical definitions have been provided by the coarse mesh generator and are often C^1 continuous.

We next consider that our spectral/*hp* element approximation within an element is a polynomial of order P. For a consistent approximation of the geometry boundary we could also approximate the surface by a P-order polynomial expansion. Dey *et al.*[134, 132] argue that to maintain optimal convergence of a P order spectral/*hp* approximation to each integral only a $(P-1)$ polynomial order approximation of the boundary is required.

Two constraints which we must impose on our polynomial approximation of $\chi_i^{1D}(\xi)$ is that the vertex points common to adjacent edges ensure continuity of elements. To determine the mapping $\chi_i^{1D}(\xi)$ it would, therefore, seem reasonable to apply a collocation approximation where the first two constraints are that the vertex points, i.e.

$$\boldsymbol{\chi}^{1D}(-1) = \boldsymbol{x}_A,\ \boldsymbol{\chi}^{1D}(1) = \boldsymbol{x}_B,$$

where $\boldsymbol{x}_A, \boldsymbol{x}_B$ are the coordinates of the end-points. For a P-order polynomial approximation we are now free to choose $(P-1)$ conditions to define our mapping. These points can be determined by defining $(P-1)$ nodal points in $-1 \leq \xi_1 \leq 1$ and $(P-1)$ corresponding points along the curvilinear edge. This point is illustrated in figure 4.27 where we show the discretisation of the same edge illustrated in figure 4.26. To define a $P = 4$ order Lagrange polynomial approximation for $\chi_i^{1D}(\xi_1)$ we require 5 nodal points. The first two points, denoted by the solid circles in figure 4.27, are provided by vertex locations. The three interior points, denoted by open circles are to be determined. A reasonable choice for these collocation points in the standard region $-1 \leq \xi_1 \leq 1$ are the the Gauss-Lobatto-Legendre integration points due to their favourable Lebesgue properties (see section 3.3.1). We are, therefore, left with the task of determining the nodal

points, $\boldsymbol{x}_i(\xi_1)$ along the curved boundary. There are a few possible methods we could apply.

An intuitive method of obtaining the surface collocation points is to initially determine the local Gaussian quadrature points along the straight line segment between the vertices. The desired collation points are then obtained by pushing these mapped quadrature points in the surface normal direction onto the geometry boundary. This technique is illustrated in figure 4.27 (b). For a relatively well behaved curved segment with curvature that does not vary significantly between the vertices, this produces reasonably distributed points as shown in the same figure. Here, the cross points on the straight line segment represent the Gauss-Lobatto-Legendre points linearly mapped to the line $A - B$. These points are then pushed onto the surface in the normal direction. Problems arise with this technique when the curvature varies rapidly along the line segment. In this case the "normal pushing" technique can produce a nodal distribution of points which is highly irregular in spacing along the arc length of the curve as illustrated in figure 4.27(c). A spacing along the arc length, which is significantly different from the spacing in the standard region (for example the Gauss-Lobatto-Legendre integration points), will result in a highly non-linear mapping $\chi_i^{1D}(\xi_1)$. This can cause the Jacobian of the mapping to be singular and make the element invalid, thereby destroying the good convergence properties of the spectral/*hp* element approximation. We are, therefore, motivated to consider an "optimal" transformation as one which minimises the distortion of the Jacobian of the mapping $\chi_i^{1D}(\xi_1)$, $i = 1, 2$ as defined in section 4.1.4. This criterion implies that the best mapping has a constant Jacobian, which is the case between a straight-sided element and the standard region under a linear mapping. A constant Jacobian mapping is also preferable in an approximation sense because a minimal quadrature order is required to ensure optimal integration of a polynomial integrand.

An alternative method of choosing the nodal points is to select points in the parametric space u which under the mapping $\boldsymbol{g}(u)$ fix the location of the nodal points on the geometry surface. The selection of the parametric nodal points u_k depends on the form of the mapping $\boldsymbol{x}_k = \boldsymbol{g}(u_k)$. An obvious initial distribution is to use the same parametric spacing in the u-space as applied in the ξ_1-region. For many mappings this selection may be perfectly adequate and even optimal. For example, when the surface is defined as a circle, i.e. $g_1(u) = R\cos(u)$, $g_2(u) = R\sin(u)$ setting the collocation point in the u-space can be shown to lead to a mapping $\chi_i^{1D}(\xi_1)$ which has a constant Jacobian. However for a more complex mapping $\boldsymbol{g}(u)$ this type of point selection can lead to a highly distorted distribution of collocation points. In such a situation the mapping $\chi_i^{1D}(\xi_1)$ can again become singular. A modification to this approach, proposed in [435], is to define a minimisation procedure to determine the parametric discrete points u_k that minimise the Jacobian of the mapping $\chi_i^{1D}(\xi_1)$. In this work the definition of an optimal mapping was approximated as the linear spacing between two collocation points $\boldsymbol{x}_k = \boldsymbol{g}(u_k)$ on the curved geometry to be the same as the spacing between the nodal points in the standard region. It is then possible to define a functional of the form

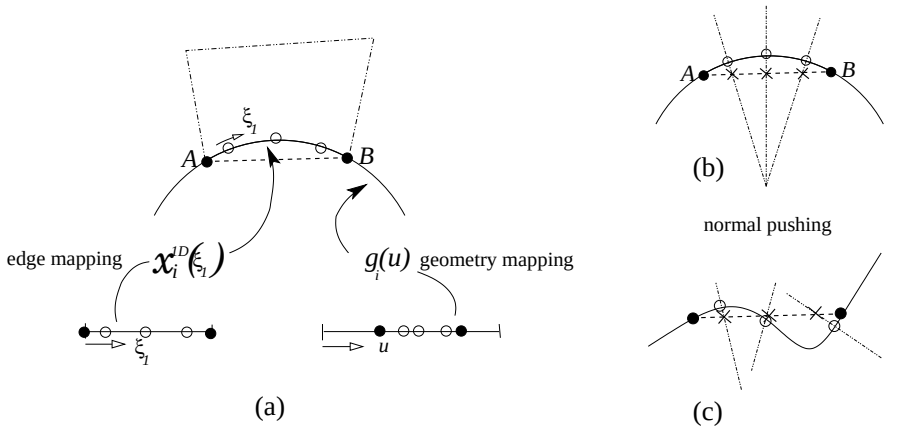


Figure 4.27 (a) Defining the mapping, $\chi_i^{1D}(\xi_1)$, using a collocation projection between the standard region, $-1 \leq \xi_1 \leq 1$, and the geometry boundary segment $A - B$. (b) Ensuring the vertices at the end points provides two collocation points (solid circles) however we are free to define other collocation/nodal points (open circles) using normal pushing (as indicated in (b) and (c)) or throught the surface parameter u-space (as shown in (a)).

$$\mathcal{J}\,(u_1, \ldots, u_{P-1}) = \sum_{i=1}^{P-1} \frac{\|\mathbf{g}(u_{i+1}) - \mathbf{g}(u_i)\|^2}{\xi_{i+1} - \xi_i}. \tag{4.96}$$

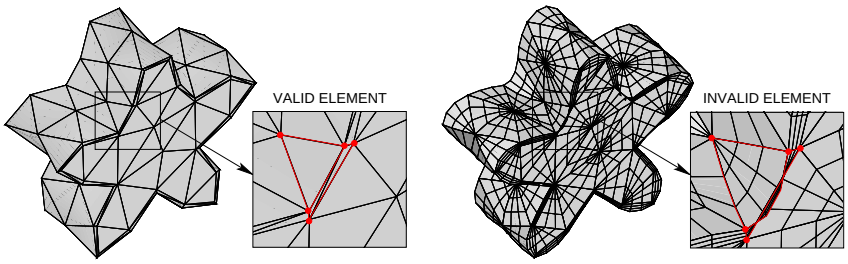
where ξ_i, $i = 0, \ldots, P$ represent the Gauss-Lobatto-Legendre integration points. The minimisation of the function $\mathcal{J}$ provides the spacing of points u_k $(1 \leq k \leq P-1)$, which gives a near optimal spacing $\boldsymbol{x}_k = \boldsymbol{g}(u_k)$ in the sense of minimising the curve Jacobian. We note, however, that the nonlinearity of the mapping $\boldsymbol{g}(u)$ makes this a non-linear optimisation problem.

Finally, we note that accounting for the surface curvature is an important prerequisite when constructing curvilinear elements. Mesh generation techniques exist (see Frey and George [165]) which take account of surface curvature as part of the mesh refinement strategy, and such an approach clearly has benefits when designing spectral/*hp* element meshes.

4.3.5.2 *Internal Edges in Two–Dimensions*

Having constructed a mapping for boundary conforming edges we require definition of the edge mappings for all other edges interior to the solution domain. Since we know the vertex location of these internal edges from the coarse mesh description, and in the absence of any other information, using a linear mapping between the vertices for interior edges is the most obvious approach. Such an approach leads to the element shapes shown in figure 4.26 where only the bottom edge is deformed.

Applying linear mappings for interior edges is attractive since it simplifies the form of the elemental mapping of the interior elements which do not touch



(a) Straight sided mesh (b) Invalid mesh

Figure 4.28 Edge deformation of a straight sided coarse mesh leading to invalid curvilinear meshes in regions of concave curvature relative to the element volume.

the solution domain boundary. This can be advantageous since the Jacobian of integral interior elements will not be influenced by any surface curvature. We note, however, that in cases where the structure of the solution is known then interior elements with curved boundaries may be advantageous from an approximation point of view. Nevertheless, a more important factor to consider is if internal edge should be deformed to avoid the generation of invalid elements after surface edge deformation. This point is illustrated in figure 4.28 where we see the generation of a curvilinear mesh from a coarse mesh generation in a flower type shape. Although this example is three-dimensional, the construction of elements within the planar surface of the surface mesh is identical to the generation of two-dimensional interior elements. Figure 4.28(a) shows a coarse straight-sided mesh, where a boundary layer region has also been generated adjacent to the surface boundary. All elements in this mesh are valid. If we now deform the edges which touch the solution domain boundary we observe in figure 4.28(b) that invalid elements are generated in regions where the surface is concave with respect to the local element (i.e., the curved edge reduces the local area of the straight sided element). In general, these problems occur when the surface deformation is larger relative to the size of the element.

Invalid Element Detection

We are, therefore, faced with the problem of identifying invalid elements and by suitable modification to make them valid. One approach for detection is to numerically evaluate the Jacobian at a set of discrete points in the element and to identify if there is a sign change or ensure that the Jacobian is larger than zero by a suitable tolerance. Although such an approach cannot absolutely guarantee the validity of the element since only a discrete set of points are evaluated, in practice an inspection at quadrature points is normally sufficient. The approach can however be numerically quite expensive. Luo *et al.* [313] have proposed to use Bezier polynomials in their curvilinear meshing strategy. Bezier curves have a number of attractive features including the property that the convex hull of the control points contain the Bezier curve. In [313] they used this property and

the fact that the derivatives and products of Bezier curves can also be expressed as Bezier curves to determine the Jacobian as a Bezier form. It is then possible to determine if the Jacobian is greater than zero by ensuring that the expansion coefficients (control points) are greater than zero.

Eliminating Invalid Elements

Having identified the invalid elements, we are faced with the problem of deciding how to correct the invalid element. This point was addressed in the work of Dey *et al.* [132, 133] where they considered two approaches: edge swapping and interior edge deformation. Edge swapping has previously been used in optimising meshes [307] but in the curvilinear context can be applied if the edge swapping does not create any more invalid elements, otherwise an infinite swapping loop might be generated [133]. An example of effective edge swapping is shown in figure 4.29(a) where the triangle ABC is invalid due to the intersection of curvilinear line $A-B$ with the straight line $A-C$. In this case swapping edges $A-C$ with $B-D$ will make two valid triangles ABD and BDC. If edge swapping is not possible then the interior edge can be deformed. A strategy for quadratic order edge deformation proposed by Dey *et al.* [133] is based on ensuring that the normals of planes containing the straight edges has the same sign as the normals of the plane containing the curved edges. Therefore, in figure 4.29(b) we consider a case where edge AC is to be deformed by a quadratic fit and moving the midpoint in the normal direction of AC. We define $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ to be the tangent vector to the straight lines AB and AC and $\boldsymbol{t}_3$ and $\boldsymbol{t}_4$ to be the tangent vector to the curved lines AB and AC. The criterion suggested by Dey *et al.* [133] is to move the point until the sign of the vector product $\boldsymbol{t}_4 \times \boldsymbol{t}_3$ is the same as that of $\boldsymbol{t}_2 \times \boldsymbol{t}_1$, or equivalently

$$(\boldsymbol{t}_2 \times \boldsymbol{t}_1) \cdot (\boldsymbol{t}_4 \times \boldsymbol{t}_3) > \epsilon,$$

where ϵ is a small constant.

Curvature Based Refinement

We have observed that problems with invalid elements tend to arise due to excessive surface deformation relative to the size of the element. An alternative approach to eliminating invalid elements is to try to avoid their generation by controlling the coarse mesh element spacing relative to the surface deformation. Curvature-based refinement in which the mesh size is obtained as a function of the curvature has been proposed by several authors [165, 290] as a way to obtaining an accurate piecewise linear approximation of a curved surface. This type of approach has also been applied to curvilinear elements in [370] and uses a variation of the curvature driven refinement of the coarse space mesh.

As shown in figure 4.30 (a) and (b), in this technique a curve is locally approximated by a circle of radius R, the radius of curvature. We assume that the mesh spacing can be represented by a chord of length c in the circle and a

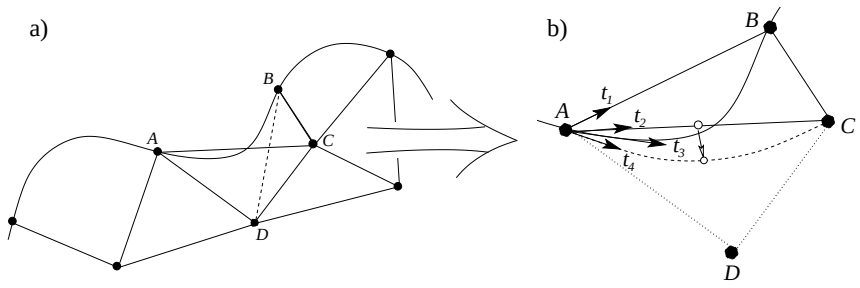


Figure 4.29 Edge swapping and edge deformation to avoid the construction of invalid triangles.

spacing δ in the normal direction. In the modelling of viscous flows, the value of δ is usually prescribed to achieve a certain boundary layer resolution. The value of c is, therefore, chosen to guarantee that the osculating circle representing the curve does not intersect the interior sides of the elements, i.e., $\theta \geq 90^o$ for the triangular element. The value of c, which should be considered as a maximum mesh spacing, can now be obtained as a function of R and δ. Its value c_t for triangular elements is

$$c_t \leq R\sqrt{\frac{2\delta}{R+\delta}}. \tag{4.97}$$

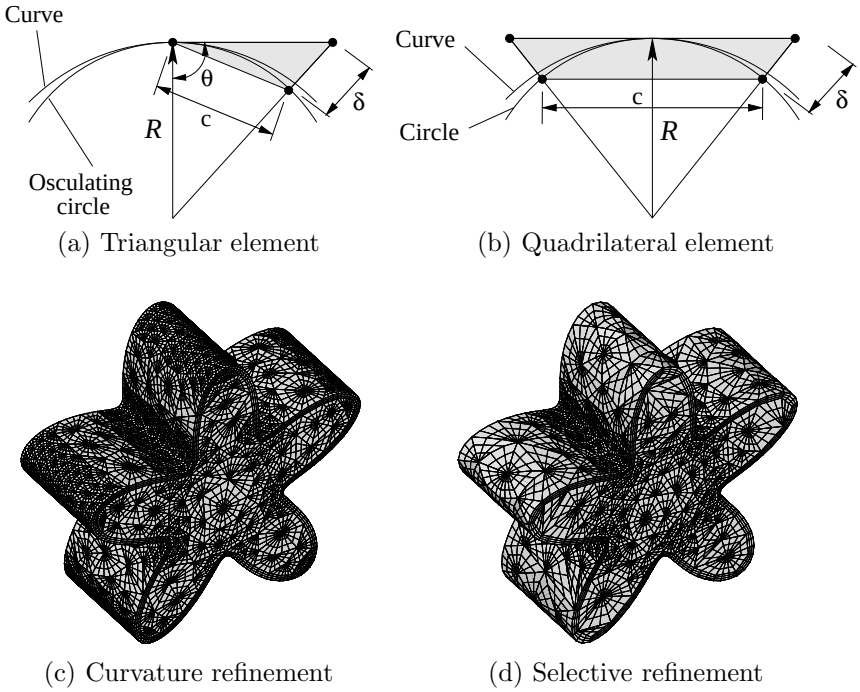
The corresponding value c_q for quadrilateral elements is

$$c_q \leq \frac{2R\delta}{R+\delta}\sqrt{1+\frac{2R}{\delta}}, \tag{4.98}$$

where the displacement δ is assumed to be the same on either side of the rectangle. It is interesting to notice that, for a given δ, the quadratic element allows for a mesh spacing c_q which is about twice the value of spacing c_t for the triangular element. If we apply the curvature-based refinement to the original problem shown in figure 4.28 we obtain a spacing which prevents the generation of invalid elements as shown in figures 4.30(c) and (d). However, we note that indiscriminate application of the curvature criteria can generate too much refinement since invalid elements are only be generated when the local curvature is concave with respect to the element volume. Selective refinement where refinement is only applied in regions of the domain which are locally concave provides a better mesh spacing as shown in figure 4.30(d).

4.3.5.3 *Extension to Three–Dimensions*

For many two-dimensional problems the techniques discussed in sections 4.3.5.1 and 4.3.5.2 are often not necessary since in two-dimensions meshes can often



(a) Triangular element (b) Quadrilateral element

(c) Curvature refinement (d) Selective refinement

Figure 4.30 Determination of minimum spacing c based on local radius of curvature R and interior spacing δ for (a) triangular elements and (b) quadrilateral elements. Application of curvature refinement to the problem of figure 4.28 using (c) indiscriminate curvature refinement and (d) selective curvature refinement.

be "hand crafted" to avoid invalid elements. However, in three-dimensions this is not the case partly due to the problems of even visualising computational meshes. It is in three-dimensions that many of the more complicated meshing strategies are most beneficial. In this section we will outline how the concepts described in sections 4.3.5.1 and 4.3.5.2 can be extended into three-dimensions.

Boundary Edges Constrained to a Surface

As we discussed in section 4.3.5.1 when a CAD edge or surface is represented by an isometric mapping (i.e., those that preserve lengths) the procedure of defining the edge mapping $\chi_i^{1D}(\xi)$ based on the same distribution of collocation points in the u-space as the ξ-space (see figure 4.26) produces reasonably good quality elements. However, when the mapping is anisometric[1] this simple approach can produce highly nonlinear mappings leading to badly shaped elements. A solution to avoid this problem is to apply an extension of the minimisation procedure similar to equation (4.96). In a three-dimensional "bottom-up" generation strat-

[1]Not isometric

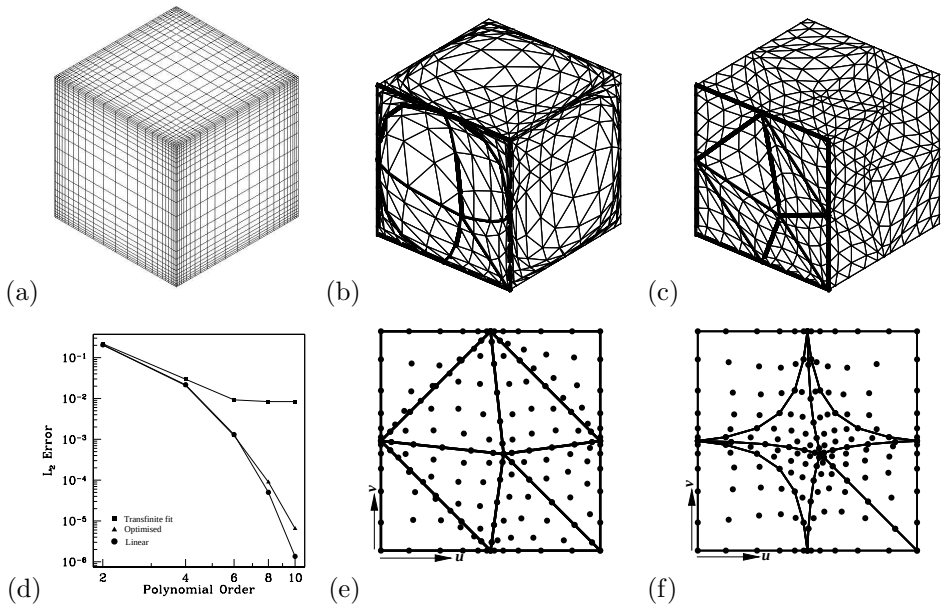


Figure 4.31 Influence of surface and edge mappings on the spectral/*hp*-type mesh generation procedure. (a) Anisometric CAD description. (b) High-order mesh using a transfinite interpolation in the parametric space as shown in figure (e). (d) High-order mesh using an optimised point placement in the parametric space as shown in figure (f).(d) Comparison of errors in an elliptic problem using the different meshes.

egy the edges which lie on a boundary curve can be dealt with by optimising equation (4.96). The next stage of the generation is to determine edges which lie within a boundary surface which is now represented by a two-dimensional parametric space $\boldsymbol{g}(\boldsymbol{u}) = [g_1(\boldsymbol{u}), g_2(\boldsymbol{u})]$ where $\boldsymbol{u} = [u, v]$. Equation (4.96), therefore, becomes a two-dimensional functional in terms of $[u, v]$ of the form

$$\mathcal{J}_e\left(\mathbf{u}_1, \ldots, \mathbf{u}_{P-1}\right) = \sum_{i=1}^{P-1} \frac{\|\mathbf{g}(u_{i+1}, v_{i+1}) - \mathbf{g}(u_i, v_i)\|^2}{\xi_{i+1} - \xi_i}.$$

To determine the location of the points (u_i, v_i) for $i = 1, .., P-1$ we need to minimise $\mathcal{J}_e$ where once again ξ_i, $i = 0, P-1$ represents the nodal spacing in the standard region for example using Gauss-Lobatto-Legendre integration points. The minimisation of $\mathcal{J}_e$ provides a set of points, $\boldsymbol{u}_k$, which give a near optimal spacing $\boldsymbol{x}_k = \boldsymbol{g}(u_k, v_k)$ in the sense of minimising the elememt surface Jacobian. The points $\boldsymbol{\xi}_k$ and $\boldsymbol{x}_k$ are then sufficient to generate a P-order polynomial approximation using a collocation projection. However, unlike the one-dimensional case the edges are now geodesics of the surface. In [435] the minimisation procedure was also extended to determine the triangular face mapping for use in hybrid elements.

As a final illustration of the role of the mapping we present the example shown in figure 4.31. We consider the generation of a tetrahedral spectral/hp mesh of fifth-order polynomials within a simple cubic computational domain $0 \leq x, y, z \leq 10$. The faces of the cube are located on tensor-product surfaces defined by an anisometric mapping shown in 4.31(a). The spacing varies linearly with a value of 0.1 at the boundary and 2.1 at the centre of the faces. The anisometry of the mapping is due to the fact that the unevenly spaced network of lines depicted in 4.31(a) is obtained as the image of a network of coordinate lines u = const. and v = const. in the parametric plane which are uniformly spaced with $\Delta u = \Delta v = 1$. The curve definition of the edges representing the intersection between each of the faces was taken to be isometric.

A standard h-type unstructured mesh generation process was used to construct a coarse mesh with 66 elements. Although the p-type elements can easily be constructed on a planar surface by a linear interpolation between the vertices, we have chosen to reconstruct the surface elements using the parametric definition of the surface as required for a non-planar surface. Therefore, in this geometry an optimal solution which would give elemental mappings with constant Jacobian is a linear distribution of points between the vertices.

Using a straightforward transfinite interpolation of points in the parametric space (u, v) results in the highly distorted surface mesh shown in figure 4.31(b). In this figure we have connected all the interior edge and face nodal points using a triangular mesh and highlighted the element boundaries on one face with dark lines. The distribution of points in the parametric plane for the highlighted surface are shown in figure 4.31(e). If we now apply the optimisation procedure [435], we obtain the surface mesh shown in figure 4.31(c) which produces a fit very close to a linear mapping between the vertices. A direct consequence of obtaining a good physical surface distribution is that the parametric distribution becomes very distorted as shown in figure 4.31(f).

Finally, we compare the L^2 error to the solution $u(x, y, z) = \sin(0.2\pi x)$ $\sin(0.2\pi y)\sin(0.2\pi z)$ of a Poisson equation to assess the influence of mesh distortion on the solution accuracy. Figure 4.31(d) compares the error of solutions obtained using computational meshes from a transfinite parametric fit (figure 4.31(b)) with those from an optimised parametric fit (figure 4.31(c)) and a standard linear fit between the physical vertices. The transfinite parametric fit leads to elements with singular Jacobian and so it is not surprising that the error quickly saturates at 1×10^{-2}. The optimised surface mapping, with a convergence tolerance in the parametric space of $\epsilon \propto 0.3/P^2$, follows the error of the linear fit up to a polynomial order of $P = 8$ where the error is of order 1×10^{-4}. For higher order expansions the rate of convergence decreases when compared to the standard linear discretisation. However, increasing the convergence tolerance of the optimal iterative procedure reduces the saturation level.

Curvature Based Refinement on a Surface - Hybrid Meshing

The extension of the curvature based refinement discussed in section 4.3.5.2 to

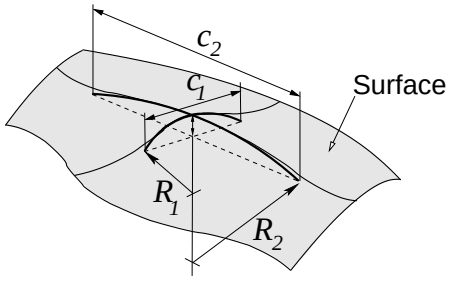


Figure 4.32 Notation for curvature refinement in a surface.

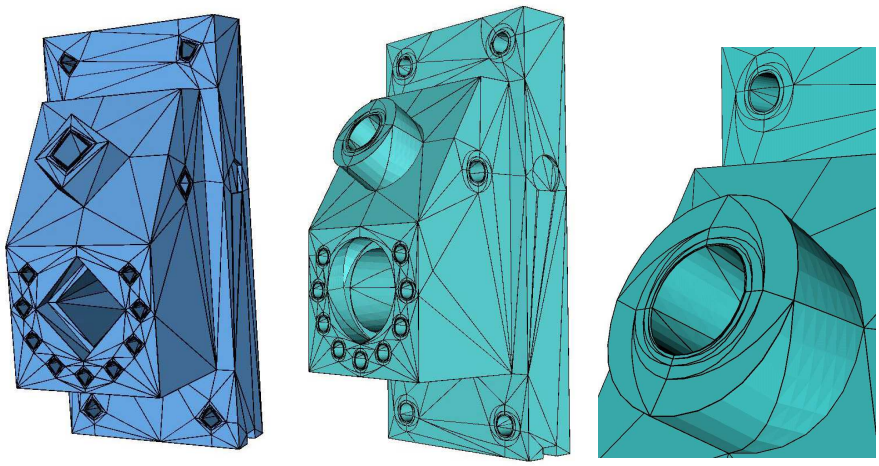
surfaces is relatively straightforward. The refinement criterion given by formulae (4.97) and (4.98) can be applied in the two principal directions or the surface as illustrated in figure 4.32. The corresponding mesh spacings, c_1 and c_2, can then be calculated from the values of the principal curvatures $k_{1,2} = 1/R_{1,2}$ and the normal mesh spacing δ using formulae (4.97) and (4.98).

As noted previously, the minimum spacing c_1, c_2 for a quadrilateral surface normal face is twice that of a triangular surface normal face. This leads us to the conclusion that *hexahedral* or *prismatic* elements adjacent to a curvilinear surface are less likely to generate invalid elements than tetrahedral elements. However, tetrahedral elements provide greater geometric flexibility in the interior of the domain. Therefore, a *hybrid* mesh containing a mix of these elements provides a good compromise between these two factors. Indeed, to generate boundary layer meshes typically involves the extension in the surface normal direction of the surface discretisation. If the surface is discretised using triangles the surface normal extension directly generates prismatic shaped elements which can be subdivided into tetrahedrons if desired. However direct use of the prismatic discretisation is favourable from an element validity point of view and local approximation. This type of strategy has been adopted in figures (4.28) and (4.30) whereas a tetrahedral boundary layer was adopted in figure (4.24).

Interior Element Modifications

As discussed in the two–dimensional case, it is often attractive from an approximation point of view, to maintain straight-sided elements when constructing three-dimensional elements . This simply arises as a consequence of the Jacobian of straight-sided elements requiring low order polynomial approximations and so not affecting the quadrature order used in integral evaluations. Although surface optimisation and curvature-based refinement can reduce the potential for generation of invalid elements it may still be necessary to deform interior edges and faces.

This point is illustrated in figure 4.33 due to Luo, O'Bara and Shephard [313, 133]. Figure 4.33 (a) shows an initial coarse straight sided mesh of this



(a) (b) (c)

Figure 4.33 High order curvilinear mesh of an engineering component: (a) initial straight sided coarse mesh, (b) high-order curved mesh conforming to curved geometry and (c) close up of high-order mesh demonstrating interior element deformation required to prevent the generation of invalid elements. (Courtsey of M. Shephard).

complex domain representing an engineering component. Figure 4.33 (b) shows the final high-order curvilinear mesh which reproduces the curved nature of the component. However, in order not to generate invalid elements the interior edges have to be curved as shown in the close up of the curvilinear mesh of the component in figure 4.33(c). Different strategies to interior modifications have been proposed in the work of Dey *et al.* [134, 133] and Luo *et al.* [313]. These strategies involved edge and face swapping and deletion as well as interior edge and face deformation. The more recent work of this group has also involved the use of Bezier representation to produce a more automatic approach to invalid element detection and modification [313].

4.3.6 Particle Tracking in Spectral/*hp* Element Discretisations

In the proceeding section we have discussed the formulation of spectral/*hp* element approximations for Eulerian descriptions of partial differential equations. These techniques will be applied to the advection equation in chapter 6 and the Navier-Stokes equations in chapters 8 and 10. Solving either the advection equation or the Navier-Stokes equations with a spectral/*hp* element formulation implies that we have a high-order polynomial approximation of the velocity field. A popular post-processing technique in fluid mechanics is to consider the streamlines, for steady flow, or the path lines, for unsteady flow. However, this requires being able to track particles over the high-order velocity field. One approach to implement particle tracking is to divide the macro element of the spectral/element discretisation into many small elements within which a linear

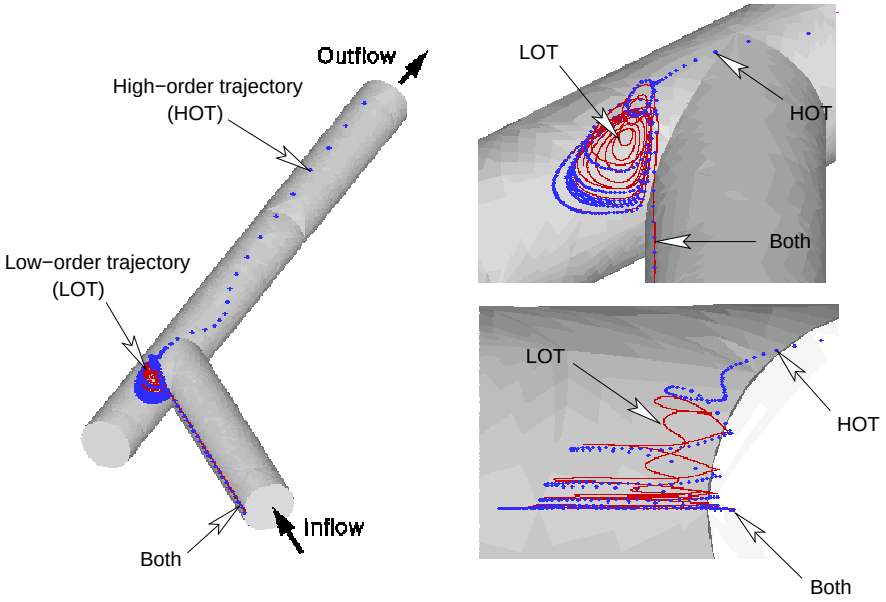


Figure 4.34 Streamlines in a junction between two straight pipes. The high-order finite element mapping is based on a seventh-order polynomial expansion and the corresponding streamline is represented by the dotted line. The solid line is the streamline starting at the same point as calculated by a commercial package using linear interpolation on a subdivision of the high-order mesh. (Remark: one high-order element was divided into 42 linear elements)

approximation is applied. On this finer mesh then commercial particle tracker can be applied. For many post-processing requirements this may be satisfactory when high accuracy is not required. However, when developing algorithms such as the strong form of the semi-Lagrangian method discussed in sections 6.4, the error associated with such a linear approximation is inadequate.

The approximation of a spectral/*hp* element velocity field by piecewise linear polynomials on smaller elements can result in the inaccurate calculation of the trajectories as shown in figure 4.34. Streamline integration is very sensitive to small changes in kinematics and therefore inconsistencies between the high-order and a linear velocity field approximation can result in kinematic changes that have a significant effect on the pathlines in complex flow regions. Figure 4.34 compares the particle traces for a spectral/*hp* element velocity field calculated by the commercial package Tecplot [10] and the algorithm proposed in [109].

The focus of this section is, therefore, to review approaches to particle tracking using a consistent approximation of the velocity field. The problem of calculating flow lines for high-order polynomial element representations has received little attention to date. Two early examples of particle tracking within quadratic

$$\text{RK1:}\ \begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad \text{RK2:}\ \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad \text{RK3:}\ \begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array} \qquad \text{RK4:}\ \begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Table 4.1 Butcher arrays for a one–stage Euler method (RK1), two–stage improved Euler method (RK2),three–stage Kutta's formula (RK3) and four–stage classical Runge-Kutta (RK4) schemes.

order elements can be found in the literature of non-Newtonian flows [199, 445] where the strain history calculated along streamlines was used in the constitutive equations for the stress tensor. In Sun & Tanner [445] the time integration was applied in physical space on a triangular mesh, whereas in Goublomme *et al.* [199] time integration was performed in the parametric space of the standard region using unstructured quadrilateral meshes. Following [109] we will discuss both of these integration strategies in what follows.

4.3.6.1 *Runge–Kutta Based Particle Tracking*

The problem of finding the trajectory $\boldsymbol{x}(t)$ of a particle is formulated as a set of ordinary differential equations

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{u}\left(\boldsymbol{x}, t\right), \tag{4.99}$$

where $\boldsymbol{x}$ is the position in space and $\boldsymbol{u}$ is the velocity field. This simply states that the tangent to the trajectory curve at a point of coordinates $\boldsymbol{x}$ is parallel to the velocity $\boldsymbol{u}$ at that point. The initial condition for this problem amounts to specifying the position $\boldsymbol{x}_0$ of the particle at a specific time, $t = t_0$, such that

$$\boldsymbol{x}\left(t_0\right) = \boldsymbol{x}_0.$$

The two main types of integration schemes for ordinary differential equations are multi-stage methods (mainly Runge-Kutta type) and multi-step methods (for example Adams-Bashforth). Whilst multi-step methods are typically more efficient in problems where the velocity field is smooth they require a start-up procedure and their time step cannot be changed easily dynamically. In particle tracking, therefore, a multi-stage algorithm is attractive since it does not require a start-up procedure and the time step can be altered dynamically.

The application of an s-stage explicit Runge-Kutta method to the general system of ordinary differential equations

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}\left(\boldsymbol{y}, t\right),$$

with the initial conditions $\boldsymbol{y}(t_0) = \boldsymbol{y}_0$ results in the iteration

$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{f}_i, \tag{4.100}$$

where $\boldsymbol{y}^n$ denotes the value $\boldsymbol{y}(t^n)$,

$$\boldsymbol{f}_i = \boldsymbol{f}\left(\boldsymbol{y}^n + \Delta t \sum_{j=1}^{i-1} a_{ij} \boldsymbol{f}_j \, , \, t^n + c_i \Delta t\right), \tag{4.101}$$

s is the number of stages and Δt is the timestep. The values b_i, c_i and a_{ij} are the entries of the corresponding *Butcher array* [287]

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

The coefficients of this array define the particular scheme employed and some examples are given in table 4.1.

The second term on the right-hand side of equation (4.100) is a weighted average of the values $\boldsymbol{f}_i$ taken at each stage. Hence, if we set

$$\overline{\boldsymbol{f}} = \sum_{i=1}^{s} b_i \boldsymbol{f}_i$$

then a generic Runge-Kutta scheme, as represented by equation (4.100), can be considered as an Euler scheme that marches in time using an averaged value of $\boldsymbol{f}$. The same consideration can be applied to each stage, hence the second term on the right-hand side of equation (4.101) can be considered as an Euler step taken with average velocity

$$\hat{\boldsymbol{f}}_i = \sum_{j=1}^{i-1} a_{ij} \boldsymbol{f}_j.$$

This interpretation of the Runge-Kutta scheme is depicted in figure 4.35 for the three-stage scheme which shows how each stage of the Runge-Kutta integration can be envisaged as an Euler step in the direction of a suitably averaged velocity.

4.3.6.2 *Particle Tracking for Spectral/hp Elements*

Two possible strategies for tracking particles depend on the space in which the time integration is performed [109]. The first strategy is to perform the time integration in the global physical space of the solution $\boldsymbol{x} \in \Omega$. Given the elemental definition of the flow field, this process involves searching for the element containing the point where the velocity is to be evaluated followed by the interpolation of the velocity using the expansion basis within the element. The two computationally intensive operations to be performed are:

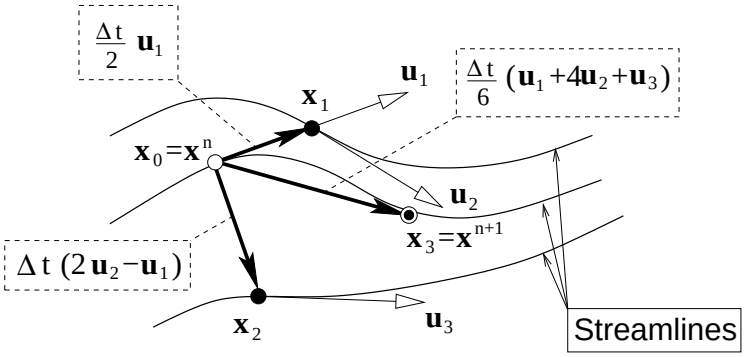


Figure 4.35 Interpretation of the Runge-Kutta integration as a series of Euler steps using a suitably averaged velocity. The figure shows the steps for a three-stage scheme.

(i) a nonlinear iterative procedure to find the local coordinates $\boldsymbol{\xi}_i$ in the parametric space from the Cartesian coordinates $\boldsymbol{x}_i$ in physical space, and

(ii) the interpolation of the velocity $\boldsymbol{u}$ at a point of parametric coordinates $\boldsymbol{\xi}_i$.

The second strategy performs the time integration in the parametric space of the standard element $\boldsymbol{\xi} \in \Omega_{st}$. This approach involves advancing the particle within an element using a transformed velocity field in the parametric space until the particle reaches the element boundary. The approach is then continued in the neighbour element sharing the boundary where the particle exits. The two main operations to be performed are:

(i) the interpolation of the velocity $\boldsymbol{u}_\xi$ at a point of coordinates $\boldsymbol{\xi}_i$ in the parametric space, and

(ii) a nonlinear iterative procedure to find the intersection of a pathline with an elemental boundary.

In the following sections we will provide an overview of the two approaches and also discuss ways of combining the methods.

Particle Tracking in the Physical Space

We recall that the explicit Runge-Kutta scheme (4.100) applied to the particle trajectory equation (4.99) results in

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{u}_i, \qquad \boldsymbol{u}_i = \boldsymbol{u}\left(\boldsymbol{x}^n + \Delta t \sum_{j=1}^{i-1} a_{ij} \boldsymbol{u}_j \,,\, t^n + c_i \Delta t \right).$$

As mentioned previously there are two computationally intensive steps. The first is to determine the inverse mapping $\boldsymbol{\xi}_i = \boldsymbol{\chi}^{-1}(\boldsymbol{x})$ and the second it to interpolate the velocity field to otain u_i. In the above algorithm this is necessary for every substep which depends on the order of the scheme. However, before

performing either of these operations it is necessary to know within which element the point x_i lies.

In the case of elements with linear mappings the inverse mapping $\boldsymbol{\xi} = \boldsymbol{\chi}^{-1}(\boldsymbol{x})$ is analytic. However, for curvilinear elements, or even bi- and tri- linear mapping of straight sided quadrilaterals and hexahedral elements the computational expense is significantly higher. To appreciate why we need to consider a way of determining the inverse mapping. One approach to determine the coordinate $\boldsymbol{\xi}_i$ such that $\boldsymbol{\chi}^e(\boldsymbol{\xi}_i) = \boldsymbol{x}_i$, is to formulate this as the problem of finding a zero of a function $\boldsymbol{F}(\boldsymbol{\xi}_i)$, where

$$\boldsymbol{F}(\boldsymbol{\xi}_i) = \boldsymbol{\chi}^e(\boldsymbol{\xi}_i) - \boldsymbol{x}_i. \tag{4.102}$$

The Newton-Raphson iteration [268] applied to equation (4.102) can be written as

$$\boldsymbol{J}_e \cdot \left[\boldsymbol{\xi}_i^{k+1} - \boldsymbol{\xi}_i^k\right] = -\boldsymbol{F}\left(\boldsymbol{\xi}_i^k\right) \qquad \boldsymbol{J}_e = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\xi}} = \frac{\partial \boldsymbol{\chi}^e}{\partial \boldsymbol{\xi}},$$

where k represents an iteration counter, i denotes the discrete point and $\boldsymbol{J}_e$ denotes the Jacobian of the mapping. In three–dimensions the above procedure, per iteration, requires three interpolation (or backward transform) operations to determine $\boldsymbol{\chi}^e(\xi_i)$ and additionally nine interpolation to evaluate $\boldsymbol{J}_e^{-1}$ at $\boldsymbol{\xi}_i$. We note that $\boldsymbol{J}_e$ can be inverted analytically as discussed in section 4.1.3.4.

The potential cost of this evaluating the inverse mapping puts a high cost on physical space particle tracking using high-order elements. The efficiency of the searching procedure to determine which element a point, $\boldsymbol{x}_i$, lies, and therefore reduce the potential algorithm cost, can be improved by using appropriate data structures. If the number of elements in the mesh is large, tree structures [65] could be used to find the element containing the starting point of a trajectory. In the data structuring of many spectral/*hp* element methods connectivity information between element is stored and so can be used to identify adjacent elements. In [500] a more intesive check was adopted where the desired physical location, $\boldsymbol{x}_i$, was checked against the normal of every discrete coordinate point known along the element boundary at the quadrature points. This approach can guarantee whether a point lies in a straight sided element eliminating any erroneous searches and inverse iterations. However, it is not sufficient test for a curvilinear element, and further checks may be necessary.

Particle Tracking in the Parametric Space

An alternative method for particle tracking that avoids the the iterative solution of equation (4.102) is to use the parametric space description of the velocity. Rather than considering the rate of change of the position of a particle in physical space $\boldsymbol{x}$, we can transform the velocity field onto the standard region, $\boldsymbol{\xi} \in \Omega_{st}$. We then obtain an equation representing the corresponding rate of change in time of the particle position in the parametric space within element e as

$$\frac{d\boldsymbol{\xi}^e}{dt} = \boldsymbol{u}_\xi^e(\boldsymbol{\xi}, t). \tag{4.103}$$

The transformed velocity $\boldsymbol{u}_\xi^e$ in the standard element can be evaluated in terms of the physical velocity $\boldsymbol{u}$ by applying the chain rule to each of its scalar components so that

$$u_{\xi_i} = \frac{d\xi_i}{dt} = \frac{\partial \xi_i}{\partial x_1}\frac{dx_1}{dt} + \frac{\partial \xi_i}{\partial x_2}\frac{dx_2}{dt} + \frac{\partial \xi_i}{\partial x_3}\frac{dx_3}{dt} \qquad i = 1, 2, 3$$

which can be written in matrix form as

$$\boldsymbol{u}_\xi^e = \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\chi}^e}\boldsymbol{u}, \tag{4.104}$$

where we note that $\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\chi}^e}$ are the standard geometrical derivatives used to differentiate in an arbitrary region and so are usually known at the quadrature points (see section 4.1.3.4).

Omitting the index e for simplicity, the use of an s-stage Runge-Kutta for the integration in time of equation (4.103) leads to

$$\boldsymbol{\xi}^{n+1} = \boldsymbol{\xi}^n + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{u}_{\xi_i}, \quad \boldsymbol{u}_{\xi_i} = \boldsymbol{u}_\xi \left(\boldsymbol{\xi}^n + \Delta t \sum_{j=1}^{i-1} a_{ij} \boldsymbol{u}_{\xi_j},\, t^n + c_i \Delta t \right). \tag{4.105}$$

We note, however, that equation (4.104) can only be applied within the corresponding elemental region since the representation of parametric velocity, $\boldsymbol{u}_\xi^e$, and the geometric derivatives $\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\chi}^e}$ are only piecewise continuous. If we can determine the intersection of the particle trajectory with the boundary of the standard element, Ω_{st}, then the continuity of the local coordinates along a boundary can be used to advance the particle to the next adjacent element. Such an approach requires replacing Δt in equations (4.105) with the time step for the particle to reach the boundary $\Delta\tau$.

As noted in [109], the parametric approach has eliminated the nonlinear inverse mapping evaluation using a physical space approach. We are, however, now faced with the problem of determining the intersection of the trajectory with the boundary of the standard element. If we use an Euler scheme ($s = 1$), the intersection with the boundary is easily determined since it is linearly dependent on Δt. For a multi-stage scheme ($s > 1$), we see by inspection of equations (4.105), that the point $\boldsymbol{\xi}^{n+1}$ is a nonlinear function of the time step. This means that the calculation of the time step $\Delta\tau$ required to move a particle exactly to the boundary is also a nonlinear problem. Although an iterative scheme can be devised to solve this problem, for elemental mappings which contain singular factors observed to be ill conditioned making this approach in curvilinear spectral/hp element methods unattractive [109].

Guided Search Approach to Particle Tracking

The previous sections have highlighted several problems in the implementation of the particle tracking algorithm in the physical and parametric spaces.

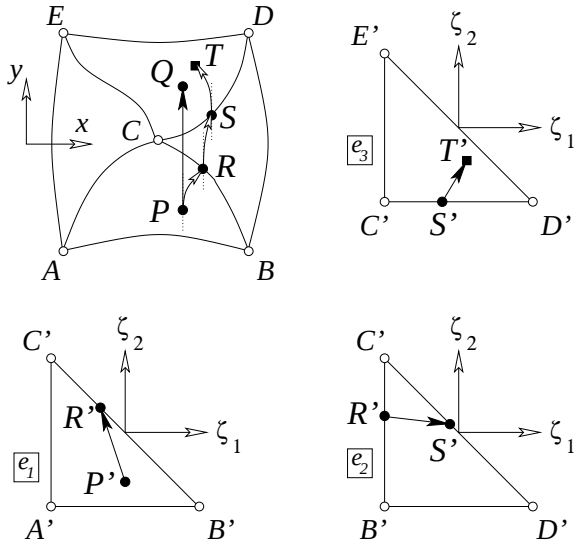


Figure 4.36 Illustration of the guided search algorithm. The vector PQ represents the step in the physical space parallel to the global averaged velocity $\mathbf{v}_P$ evaluated at P. The straight segments $P'R'$, $R'S'$ and $S'T'$ are steps in the parametric space parallel to the transformed velocity $\boldsymbol{v}_\xi^e$ at point P. The local velocity $\boldsymbol{v}_\xi$ is evaluated at the points P', R' and S' for elements e_1, e_2 and e_3 respectively. The path $PRST$ is the image in physical space of the piecewise linear steps taken in the parametric space.

The main weakness of the physical space approach is the need to solve the nonlinear inverse mapping problem. This deficiency can be overcome by using a time integration scheme in the parametric space but at the expense of requiring the solution of the nonlinear problem of finding the intersection of the trajectory with the elemental boundary. To overcome these problems a combined approach was proposed in [109] where the velocity is predominantly evaluated in physical space but utilize the parametric space. Such an approach eliminates the inverse mapping iteration at each substep although it introduces an error associated with the variation of the Jacobian of the mapping.

The philosophy behind the *guided search* is to have a particle leaving an element in the parametric space eliminating the need to resort to an iterative procedure to obtain $\boldsymbol{\xi}_i = \boldsymbol{\chi}^{-1}(\boldsymbol{x}_i)$. Since the Runge Kutta scheme can be considered as a series of linear substeps, in this approach we take a series of linear substeps in the parametric space instead of the physical space. This approach is illustrated in figure 4.36 where we consider a step starting at point P in the physical space. A linear step in physical space $\Delta\boldsymbol{x} = \boldsymbol{v}\Delta t$ would take the particle to point Q. We then require the local parametric coordinate of point Q in order to proceed. In the guided search, the parametric point P' is advanced by a linear substep $\Delta\boldsymbol{\xi} = \boldsymbol{v}_\xi^{e_1}\Delta\tau_{e_1}$ based on the local parametric velocity $\boldsymbol{v}_\xi^{e_1}$. In general, the

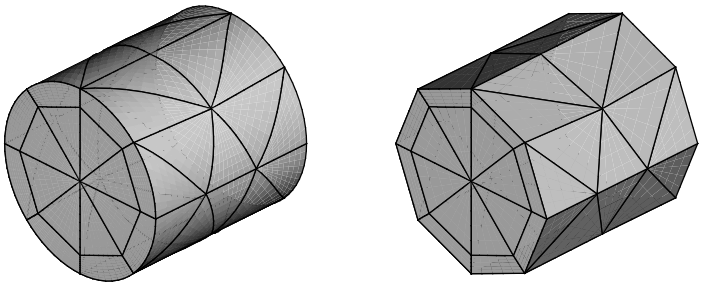
point will not remain within an element. The time taken for the point to meet a boundary of the parent element (point R' in figure 4.36) is $\Delta\tau_{e_1} \leq \Delta t$. Since the step is linear and the boundary is planar, the intersection can be evaluated analytically. To complete the guided search, a new parametric velocity $\boldsymbol{v}_\xi^{e_2}$ is then evaluated at point R' in element e_2. The point is then linearly advanced through element e_2 over a time $\Delta\tau_{e_2}$ which is evaluated as the time for the particle to reach S'. To complete the step an analogous proceedure is followed in element e_3 usign a new parametric velocity $\boldsymbol{v}_\xi^{e_3}$. The particle is then linearly advanced a time $\Delta\tau_{e_3}$ such that $\Delta t = \Delta\tau_{e_1} + \Delta\tau_{e_2} + \Delta\tau_{e_3}$.

This procedure significantly reduces the computation time required to trace a particle and overcomes the problems posed by the iterative solution of the non-linear problems associated with the two previous particle tracking strategies. The computational saving is a consequence of the fact that each small substep of the above example only requires three interpolation operations to evaluate $\boldsymbol{v}_\xi$ at each boundary intersection. This should be compared to twelve interpolation operations required in the Newton-Rhapson iteration. If many substeps are necessary due to multiple element boundary crossing, the guided search may still be expensive but in general only one or two substeps are typically required. The guided search is exact when applied to elements with a constant Jacobian but an error arises when the trajectory crosses elements with varying Jacobian. Often, high-order schemes use linear element mappings when dealing with straight-sided elements and so varying Jacobian are usually associated with curvilinear elements.

4.3.6.3 *Examples of Particle Tracking Schemes*

We now have a variety of possible strategies to handle particle tracking within high-order spatial representations and so compare the following four approaches:

1. Particle tracking in the physical space evaluating the inverse mapping using a Newton-Raphson iteration as discussed in section 4.3.6.2. We will denote this scheme as the *physical space* algorithm.
2. Particle tracking in the physical space using the guided search algorithm discussed in sections 4.3.6.2. We will denote this scheme as the *guided search* algorithm.
3. Particle tracking in the physical space using the guided search algorithm (see sections 4.3.6.2) and checking the error between the physical space advancement and the guided search. This allows the error introduced by curved elements to be monitored and requires an error tolerance ϵ above which the iterative technique to evaluate the inverse mapping is applied. We will refer to this scheme as the *guided search* (ϵ) algorithm.
4. Finally, a hybrid scheme where the particles are advanced in the parametric space, as discussed in section 4.3.6.2, provided they remain within the element during all substeps of the Runge-Kutta algorithm. If during a substep the particle leaves the elemental region then physical space scheme using



(a) (b)

Figure 4.37 Mixed prismatic and tetrahedral meshes using 83 elements within a cylindrical pipe: (a) curved elements, (b) straight-sided elements.

the error-checked guided search is applied. We will refer to this scheme as the *hybrid* algorithm.

In all of the above schemes care should be taken to consider the role of numerical precision on the criterion to determine whether a point lies within an element or when to say a point lies within an elemental face. For the interface between two curvilinear elements it is possible to get into an infinite loop if the velocity is almost tangent to the face which is only defined up to numerical roundoff. Further discussion on intersection criteria can be found in [109].

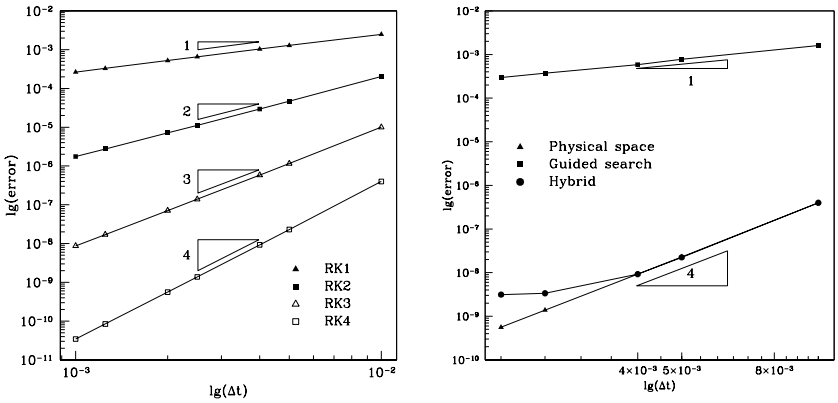
In the following tests a range of schemes are considered including Euler/RK1, RK2, RK3 and RK4 using the meshes shown in figure 4.37 which contain 37 prismatic elements adjacent to the boundary and 46 tetrahedral elements in the rest of the domain. The curvature of the surface is represented by positioning one of the triangular faces of the prismatic elements on the cylindrical surface as shown in figure 4.37(a). The elemental boundary curvature can be removed to obtain a linear surface representation as shown in figure 4.37(b). Within this domain an analytic unsteady solution, previously used in [121], was adopted of the form

$$u = -x, \qquad v = -0.1y, \qquad w = -20ze^{-0.1t},$$

which corresponds to a particle location at time t of

$$x(t) = x_0 e^{-t}, \qquad y(t) = y_0 e^{-0.1t}, \qquad z(t) = z_0 e^{200\left(e^{-0.1t}-1\right)},$$

where x_0, y_0, z_0 are initial coordinates of the particle. The starting point was taken to be $x_0 = 0.5, y_0 = 0.25, z_0 = 0.35$, and since the solution of this system is relatively stiff a relatively short final time $T = 0.2$ was considered. Figure 4.38(a) shows a comparison of the convergence rate of the guided search algorithm with error checking, using a tolerance $\epsilon = 10^{-12}$, and the physical space scheme for all the Runge-Kutta schemes. The error in these tests is measured as the distance



(a) (b)

Figure 4.38 (a) Temporal convergence for different Runge-Kutta schemes using an analytic solution with the physical space and guided search ($\epsilon = 10^{-12}$) algorithms. (b) Temporal convergence for the RK4 scheme using the physical space, guided search and hybrid algorithms.

between the final location of the particle and the analytic solution relative to the exact value.

Figure 4.38(b) shows the converge rate for three schemes using the RK4 time integration where we observe that the guided search algorithm with no error checking only produces a linear convergence rate. Since the trajectory determined by the guided search is influenced by the nonlinear elemental mapping the deterioration of convergence is to be expected. We note, however, that the hybrid algorithm maintains a fourth-order convergence rate until a level of 10^{-9} where the error of the elemental mapping saturates the results.

To compare the relative merit of each scheme, figures 4.39(a) and (b) show timings for two numerical experiments. In both cases, a circular ring of particles was released within the computational domain where the velocity was set to be the numerical solution to the Poiseuille flow. All the tests were performed using the RK4 scheme over 100 time steps with a time step of $\Delta t = 0.0125$. In the first test, shown in figure 4.39(a), we consider a ring of diameter $0.45D$ chosen to guarantee that all particles remain within the tetrahedral mesh. Since all these elements have linear mappings the results indicate that there is practically no difference between the computational cost of the different algorithms for a fixed polynomial order. The scaling within this region is approximately $O(P^{1.6})$ and is well below the asymptotic scaling value of $O(P^3)$ expected when the interpolation of the velocity field dominates.

Releasing a ring of particles of a larger diameter, $0.9D$, produces a significant difference in the timings included in figure 4.39(b). These particles now travel within the curved prismatic region of the computational domain and are there-

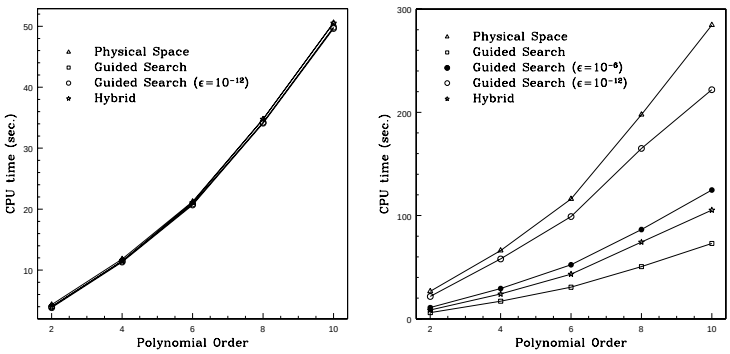


Figure 4.39 (a) Time to march 100 particles on a circle of radius $0.45D$ through a mesh of tetrahedral elements.(b) Time to march 100 particles at a radius of $0.9D$ through a region discretised by prismatic elements.

fore more sensitive to the non-linear mapping introduced by the deformation of the elements. In this example the physical space particle tracking is the most costly. It is approximately four times more expensive than the guided search algorithm without error checking. If we introduce error checking in the guided search algorithm, the cost depends on the error tolerance ϵ.

4.4 Exercises: Implementation of a 2D Spectral/*hp* Element solver for a Global Projection Problem Using a C^0 Galerkin Formulation

Implementation note: *Exercises to help implement a 2D spectral/hp element solver/projection operator.*

To complement the discussion of local and global spectral/*hp* element operations we propose the following series of exercises to help develop a two-dimensional spectral/*hp* element solver. In section 3.5 we have already suggested some exercises to develop a local elemental mass matrices for triangular and quadrilateral tensorial expansions. In this section we will build upon these exercises and develop a global C^0 continuous projection operator as well as discuss how to enforce Dirichlet boundary conditions. In section 5.6 we will then finally extend the multi-dimensional projection operator to a two-dimensional elliptic solver. The exercises are structured with a view towards developing a two-dimensional elliptic solver based on a standard Galerkin formulation. Although all of the concepts applied in this section have been discussed in the previous sections of this chapter, the exercises will demonstrate how each of the concepts can be used in practice.

Some useful codes are also available on the web page

http://www.ae.ic.ac.uk/staff/sherwin/HpSpectralBook/

1. Similar to section 2.6 a good starting point for the development of the solver is to set up routines to perform numerical integration in both the standard regions and a general element. We recall from section 4.1.1 that for integration in the standard region we require the quadrature weights w_i, w_j and zeros z_{1i}, z_{2j} which can either be generated as discussed in

appendix B.2 or using the code in the *Polylib* library from the web page. Having obtained the zeros and weights and following sections 4.1.1.1 and 4.1.1.2, try the following exercises:

(a) Integrate $u(\xi_1, \xi_2) = [\xi_1]^6 \times [\xi_2]^6$

$$\int_{-1}^{1}\int_{-1}^{1} u(\xi_1,\xi_2)\ d\xi_1 d\xi_2 = \sum_{i=0}^{Q-1}\sum_{j=0}^{Q-1} w_i w_j\ u(\xi_{1i},\xi_{2j}) = \frac{4}{49}$$

using Gauss-Lobatto-Legendre quadrature in both the ξ_1 and ξ_2 directions with $Q = 4, 5$ and 6.

(b) Integrate $u(\xi_1, \xi_2) = [\xi_1]^6 \times [\xi_2]^6$

$$\begin{aligned}\int_{-1}^{1}\int_{-1}^{-\xi_2} u(\xi_1,\xi_2)\ d\xi_1 d\xi_2 &= \int_{-1}^{1}\int_{-1}^{1} u(\eta_1,\eta_2)\left(\frac{1-\eta_2}{2}\right)\ d\eta_1 d\eta_2 \\ &= \sum_{i=0}^{Q-1}\sum_{j=0}^{Q-1} w_i w_j\ u(\eta_{1i},\eta_{2j}) = \frac{2}{49}\end{aligned}$$

using Gauss-Lobatto-Legendre quadrature in the η_1 direction and Gauss-Radau-Legendre quadrature (including the point $\eta_2 = -1$) in the η_2 directions with $Q = 4, 5$ and 6. Note that the factor $(1-\eta_2)/2$ factor could also be incorporated into the integration by using Gauss-Radau-Jacobi-(1,0) quadrature, see section 4.1.1.2.

(c) Consider a general straight sided quadrilateral element, Ω^e, defined to have vertices $(x_1^A, x_2^A) = (0,0)$, $(x_1^B, x_2^B) = (1,0)$, $(x_1^C, x_2^C) = (2,1)$, $(x_1^D, x_2^D) = (0,1)$. Numerically integrate $u(\xi_1, \xi_2) = [\xi_1]^6 \times [\xi_2]^6$, i.e.

$$\begin{aligned}\int_{\Omega^e} u(x_1,x_2)\ dx_1 dx_2 &= \int_{\Omega_{st}} u(\xi_1,\xi_2)|J|\ d\xi_1 d\xi_2 \\ &= \sum_{i=0}^{Q-1}\sum_{j=0}^{Q-1} w_i w_j u(\xi_{1i},\xi_{2j})|J(\xi_{1i},\xi_{2j})| = \frac{218}{47}\end{aligned}$$

using Gauss-Lobatto-Legendre quadrature in both the ξ_1 and ξ_2 directions with $Q = 4, 5$ and 6. You will need to determine the local mapping $\chi_i(\xi_1, \xi_2)$ as discussed in section 4.1.3 and calculate the Jacobian as discussed in section 4.1.3.3. For this case the Jacobian can either be evaluated analytically or numerically using the differentiation techniques discussed in 4.1.2.1.

(d) Consider a general straight sided triangular region, Ω^e, defined to have vertices $(x_1^A, x_2^A) = (1,0)$, $(x_1^B, x_2^B) = (2,1)$, $(x_1^C, x_2^C) = (1,1)$. Numerically integrate $u(\xi_1, \xi_2) = [\xi_1]^6 \times [\xi_2]^6$, i.e.

$$\int_{\Omega^e} u(x_1,x_2)\ dx_1 dx_2 = \int_{\Omega_{st}} u(\eta_1,\eta_2)\left(\tfrac{1-\eta_2}{2}\right)|J|\ d\eta_1 d\eta_2$$

$$= \sum_{i=0}^{Q-1}\sum_{j=0}^{Q-1} w_i w_j u(\xi_{1i},\xi_{2j}) \left(\tfrac{1-\eta_{2j}}{2}\right) |J(\xi_{1i},\xi_{2j})| = \frac{31}{7}$$

using Gauss-Lobatto-Legendre quadrature in the ξ_1 direction and Gauss-Radau-Legendre quadrature in the ξ_2 direction with $Q = 4, 5$ and 6. As with the last exercise you will need to determine the local mapping $\chi_i(\xi_1, \xi_2)$ as discussed in section 4.1.3 and calculate the Jacobian as discussed in section 4.1.3.3.

(e) As a final exercise using the elemental domains defined in 1(c) and 1(d), calculate $\mathcal{I} = \int_{\Omega^e} \sin(x_1)\sin(x_2)\, d\boldsymbol{x}$ for $2 \leq Q \leq 8$ and plot the error, ε, between the exact and numerical integrals versus Q on semi-log axes. In using Gaussian quadrature we are essentially approximating the smooth function $\sin(x)\sin(y)$ with polynomials of order $Q-1$. The error, ε, will therefore be proportional to $\varepsilon \propto C^Q$. Therefore, plotting $\log(\varepsilon)$ versus Q should asymptotically give a straight line since taking the log of the error relationship $\log(\varepsilon) \propto Q \log(C)$.

2. 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 $f(x_1, x_2)$ is a known function for example $f(x_1, x_2) = [x_1]^6 \times [x_2]^6$ or $f(x_1, x_2) = \sin(x_1)\sin(x_2)$. We recall from section 2.6 that projection problems are helpful since they do not require any boundary conditions to be imposed. Extending the formulation in section 2.2 our Galerkin problem in the elemental region Ω^e can be states as:
Find $u^\delta \in \mathcal{X}^\delta$, such that

$$\int_{\Omega^e} v^\delta(\boldsymbol{\xi}) u^\delta(\boldsymbol{\xi}) d\boldsymbol{\xi} = \int_{\Omega^e} v^\delta f(\boldsymbol{\xi}) d\boldsymbol{\xi}, \qquad \forall\ v^\delta \in \mathcal{V}^\delta,$$

and for a Galerkin expansion we define the expansion space $\mathcal{X}^\delta$ to be the same as the test space $\mathcal{V}^\delta$.
Defining a discrete expansion $\phi_{pq}(\boldsymbol{\xi})$ and accordingly a discrete solution $u^\delta(\boldsymbol{\xi}) = \sum_p \sum_q \hat{u}_{pq}\phi_{pq}(\boldsymbol{\xi})$ leads to the matrix problem (see also section 4.1.5.3)

$$\boldsymbol{M}^e \hat{\boldsymbol{u}} = \left(\boldsymbol{B}^T \boldsymbol{W} \boldsymbol{B}\right) \hat{\boldsymbol{u}} = \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{f},$$

where the above matrices and vectors were defined in section 4.1.5.1.
In section 3.5 we previously discussed how to construct the elemental mass matrix in the standard elemental region Ω_{st}. In this exercise we need to construct the elemental matrix for a general shaped subdomain. At the elemental level we could use any of the two-dimensional bases defined in chapter 3. However, for use in C^0 expansion we will focus on either the tensorial quadrilateral expansions $\phi_{pq}(\xi_1, \xi_2) = h_p(\xi_1)h_q(\xi_2)$, $\phi_{pq}(\xi_1, \xi_2) = \psi^a_p(\xi_1)\psi^a_q(\xi_2)$ or the tensorial triangular expansion $\phi_{pq}(\xi_1, \xi_2) = \psi^a_p(\eta_1)\psi^b_{pq}(\eta_2)$.

Conside the following task using one (or more) of these bases within either a straight sided quadrilateral region with vertices $(x_1^A, x_2^A) = (0,0)$, $(x_1^B, x_2^B) = (1, 0.5)$, $(x_1^C, x_2^C) = (2,1)$, $(x_1^D, x_2^D) = (0.5, 1)$ or a straight sided triangular region with vertices $(x_1^A, x_2^A) = (0,0)$, $(x_1^B, x_2^B) = (1, 0.5)$, $(x_1^C, x_2^C) = (2,1)$:

(a) Construct the mass matrix $\boldsymbol{M}^e$ in the region Ω^e for $P_1 = P_2 = 8$ and $Q_1 = Q_2 = 10$ where

$$\boldsymbol{M}^e[m(pq)][n(rs)] = \int_{\Omega_{st}} \phi_{pq}(\boldsymbol{\xi})\phi_{rs}(\boldsymbol{\xi})|J(\boldsymbol{\xi})|\, d\boldsymbol{\xi} \qquad 0 \leq m, n \leq N_m.$$

In the above equation we recall that $J(\boldsymbol{\xi})$ is the Jacobian of the mapping between the element Ω^e and the standard region Ω_{st} and $m(pq), n(rs)$ represent mappings between the index pairs (p,q) and (r,s) with the unique indices m and n as discussed in section 4.1.5.1.

(b) Construct the right-hand-side vector $\hat{\boldsymbol{f}} (= \boldsymbol{B}^T \boldsymbol{W} \boldsymbol{f})$

$$\hat{\boldsymbol{f}}[m(pq)] = \int_{\Omega^e} \phi_{pq}(\boldsymbol{\xi}) f(\boldsymbol{\xi}) |J(\boldsymbol{\xi})|\, d\boldsymbol{\xi} \qquad 0 \leq m \leq N_m$$

where $f(\boldsymbol{\xi}) = [\xi_1]^7 \times [\xi_2]^6$ and using a uniform polynomial order $P = 8$ and uniform quadrature order $Q = 10$. Note that it is economical to evaluate the inner product operation $\boldsymbol{B}^T \boldsymbol{W} \boldsymbol{f}$ by taking advantage of the tensorial nature of the expansion basis using the sum factorisation technique discussed in section 4.1.6.

(c) Using a matrix inversion technique (for example the Cholesky factorisation and routines *dpptrf* and *dpptrs* in the *LAPACK* library [13]) invert the symmetric mass matrix $\boldsymbol{M}^e$ and solve for $\hat{\boldsymbol{u}} = [\boldsymbol{M}^e]^{-1} \hat{\boldsymbol{f}}$. Verify your solution by performing the backwards transformation $\boldsymbol{B}\hat{\boldsymbol{u}}$ (using sum factorisation) to evaluate the solution at the quadrature points and verify the solution is equal to $u^\delta(\xi_{1i}, \xi_{2j}) = [\xi_{1i}]^7 \times [\xi_{2j}]^6$.

3. To solve the elemental projection problem of the question 2 using a non-tensorial nodal basis we can adopt the matrix construction directly since the sum factorisation techniques are no-longer applicable. As discussed in section 4.1.5.1 the basis matrix for a Lagrange non-tensorial basis $\boldsymbol{B}[m][n] = L_n(\boldsymbol{\xi}_m)$ can be evaluated at the quadrature points $\boldsymbol{\xi}_m(ij) = [\eta_{1i}, \eta_{2j}]$ (see equation (4.53)) as

$$\boldsymbol{B} = \boldsymbol{B}_{\mathcal{T}} \boldsymbol{B}_{\mathcal{N}}^{-1}$$

where $\boldsymbol{B}_{\mathcal{N}}$ is a square matrix of the basis ϕ_{pq} evaluated at the nodal points $\boldsymbol{\zeta}_{n'}$

$$\boldsymbol{B}_{\mathcal{N}}[n'][n(pq)] = \phi_{pq}(\boldsymbol{\zeta}_{n'})$$

and $\boldsymbol{B}_{\mathcal{T}}$ is a non-square matrix of the basis ϕ_{pq} evaluated at the quadrature points $\boldsymbol{\xi}_{m(ij)}$, i.e.

$$\boldsymbol{B}_{\mathcal{T}}[m(ij)][n(pq)] = \phi_{pq}(\boldsymbol{\xi}_{m(ij)}).$$

The elemental mass matrix can then be assembled in matrix form as $\boldsymbol{M}^e = \boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B}$ where $\boldsymbol{W}$ is defined in section 4.1.5.1.

(a) Solve the elemental projection problem in the triangular region defined in question 3 using a Lagrange polynomial basis defined through the electrostatic points as given in appendix (D). Take $P = 8$ ($N_m = 45$) and $f(\boldsymbol{\xi}) = [\xi_1]^7 \times [\xi_2]^6$ and verify that the expansion coefficients $\hat{\boldsymbol{u}} = [\boldsymbol{M}^e]^{-1}\hat{\boldsymbol{f}}$ where $\hat{\boldsymbol{f}} = \boldsymbol{B}^T\boldsymbol{W}\boldsymbol{f}$ are the exaction solution at the nodal/electrostatic points. Although any conveniently defined tensor basis can be used the properties of the orthogonal basis the bases $\phi_{pq} = \widetilde{\psi}^a_p\widetilde{\psi}^b_{pq}$ are numerically advantageous.

4. The next step is to solve the projection problem for multiple elements. Consider the four element quadrilateral or triangular mesh shown in figure 4.40. We want to construct the global projection for a $P = 6$ polynomial order expansion within this region. For either a quadrilateral or triangular expansion perform the following steps:

(a) Using the discussion in sections 4.2.1 and 4.2.1.1 construct a boundary mapping array bmap$[e][i]$ which maps the local elemental boundary degrees of freedom to the global boundary degrees of freedom. For a modal expansion you will also need to define the sign array sign$[e][i]$.

(b) Construct the elemental mass matrices $\boldsymbol{M}^e$ $1 \le e \le 4$ for the elements in the mesh. In constructing the mass matrix use a local index ordering so that the boundary degrees are listed first and follow a similar ordering convention as the mapping array you have defined in part (a). Note that since the elements are straight-sided (and in the quadrilateral case they have similar angles to the standard region) the elemental mass matrices will be identical as each element has the same constant mapping Jacobian.

(c) Construct the global mass matrix $\boldsymbol{M} = \boldsymbol{\mathcal{A}}^T\underline{\boldsymbol{M}^e}\boldsymbol{\mathcal{A}}$ as discussed in section 4.2.2. Since we have only defined a boundary mapping array the interior modes can be ordered in continuous blocks looping over each element (see also figure 2.11). The action $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{A}}^T$ can be evaluated as shown in equations (4.81), (4.82) and (4.83).

(d) Construct the right-hand-side vector $\hat{\boldsymbol{f}}_g = \boldsymbol{\mathcal{A}}^T\underline{\boldsymbol{B}^{eT}\boldsymbol{W}^e\boldsymbol{f}^e}$, where $\boldsymbol{f}^e$ contains the function $f(x_1, x_2) = \cos(x_1)\cos(x_2)$ for $(x_1, x_2) \in \Omega^e$ evaluated at the elemental quadrature points. A useful debugging case is to also consider the case where $f(x_1, x_2) = 1$.

(e) By inverting the global mass matrix solve the problem $\hat{\boldsymbol{u}}_g = \boldsymbol{M}^{-1}\hat{\boldsymbol{f}}_g$. This inversion can also be performed using the static condensation technique discussed in section 4.2.3. To recover the local solution within each element we can use the assembly operator $\hat{\boldsymbol{u}}_l = \underline{\hat{\boldsymbol{u}}^e} = \boldsymbol{\mathcal{A}}\hat{\boldsymbol{u}}_g$. The local solution at the quadrature points can then be determined in each element from $\boldsymbol{u}^e = \boldsymbol{B}\hat{\boldsymbol{u}}^e$ as discussed in section 4.1.5.2.

(f) Determine the L^2 error in the projection, $\varepsilon\left[\int_\Omega (u^\delta - u^{\text{exact}})^2 d\boldsymbol{x}\right]^{1/2}$, for different polynomial orders $4 \le P \le 10$ and for different uniform

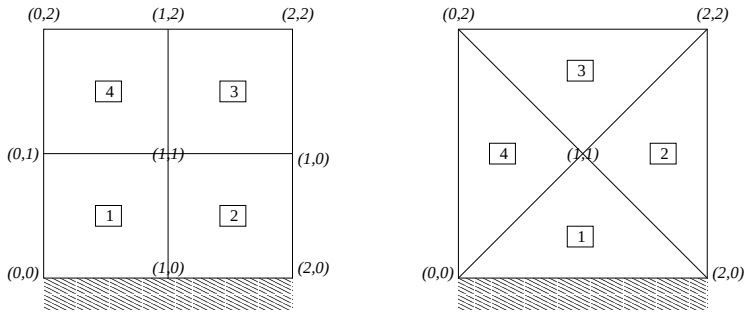


Figure 4.40 Four element domains for global projection problems

discretisation in the x_1 and x_2 directions. Plot the error as a function of polynomial order and the square root of the number of elements (or element size h) to demonstrate the difference between h and p convergence for this smooth problem.

5. The last exercise is to impose Dirichlet boundary conditions on the solution. This will also be necessary in solving the elliptic problems suggested in section 5.6. Dirichlet boundary conditions can be imposed using the lifting technique discussed in sections (2.2.1.3) and (4.2.4.1). This requires linearly decomposing the solution into a known function, which satisfies the Dirichlet boundary conditions, and the remaining solution with homogeneous boundary conditions. At an implementation level this process involves renumbering the boundary mapping array bmap$[e][i]$ as described in section 4.2.4. We also require an elemental boundary transformation to determine the expansion coefficient of the degrees of freedom on the Dirichlet boundary as discussed in section 4.3.2. Consider the global projection problem of question 4 with a Dirichlet boundary condition on the domain boundary $x_2 = 0$ of the domains shown in figure 4.40. The Dirichlet boundary condition therefore is $u(\partial\Omega_{\mathcal{D}}) = \cos(x_1)$. Solve the global projection problem of question 4, constrained to satisfy the Dirichlet boundary condition in the following step:
   (a) Determine the expansion coefficient of all boundary degrees of freedom using the elemental boundary transformation technique of section 4.3.2
   (b) Order the boundary mapping array bmap$[e][i]$ so that the known (or lifted) boundary degrees of freedom are ordered after the unknown boundary degrees of freedom. This can either be performed "by-hand" or implementing an automated procedure as discussed in section 4.2.4.
   (c) Invert the global mass matrix using a static condensation technique where only the submatrix of the boundary matrix system corresponding to the unknown degrees of freedom is inverted as shown in section (4.2.4.2) and figure (4.21). Verify that your answer converges for different polynomial orders and number of elemental domains.