Chapter 2

Numerical Methods

In this chapter,

We will discuss the fundamentals of numerical methods relevant to solving the Navier-Stokes equations. We begin the discussion of the weighted of residuals (§2.1) and the spatial discretisation using spectral/*hp* element methods in one dimension (§2.3). This is followed by techniques for solving the Navier-Stokes equations (§2.4), introducing the velocity-correction scheme, enforcing a constant flow rate and the quasi-3D approach for semi-homogeneous domains. This chapter concludes with numerical techniques for the stability analysis of the Navier-Stokes equations (§2.5), including eigenvalue computation and edge tracking.

2.1 Method of weighted residuals

Spatial discretisation errors, or residuals, arises as one seeks an approximate solution to some partial differential equation (PDE). The method of weighted residual provides a generic mathematical framework in which constraints on the residual could be applied flexibly, defining the spatial discretisation scheme and its convergence properties. In summary, we approximate the solution of PDE by considering a finite expansion of a suitable basis, to which its coefficients are sought after by minimising the inner product between the PDE and a test (or weight) function. To demonstrate this, we consider a linear partial differential equation as,

$$\mathbf{L}[u(x)] = 0, \quad x \in \Omega, \tag{2.1}$$

where $\mathbf{L}$ refers to a linear spatial differential operator subjected to some boundary conditions within the domain, Ω, while $u(x)$ refers to the exact solution of $\mathbf{L}$. Examples of PDEs with linear spatial differential operators include the Laplace equation, $\nabla^2 u = 0$, Poisson equation, $\nabla^2 u = f$, and the Helmholtz equation, $\nabla^2 u + \lambda u = f$. We suppose that the exact solution $u(x)$ can be approximated (discretised) by N finite number of basis (or expansion) functions, $\Phi(x)$.

$$u(x) \approx u^{\delta}(x) = \sum_{i=0}^{N-1} \hat{u}_i \Phi_i(x), \tag{2.2}$$

where $u^\delta(x)$ refers to the approximate solution of $u(x)$, consisting of a linear combination of the product between the i^{th} basis coefficient, $\hat{u}_i$, and the i^{th} global basis expansion, $\Phi_i(x)$, defined within Ω. Since $u^\delta(x)$ is an approximate solution of equation (2.5), we expect a residual (or 'error') between the exact solution, $u(x)$, and $u^\delta(x)$,

$$\mathbf{L}[u^\delta(x)] = R[u^\delta(x)], \tag{2.3}$$

where $R[u^\delta(x)]$ refers to the residual which depends on the approximate solution $u^\delta(x)$ and ~~varying~~ varies within Ω. In other words, equation (2.5) (2.3) might not be satisfied everywhere in Ω. We need to place next restrictions on the residual, such that ~~it~~ the residual approaches zero, $R \to 0$, and the approximate solution approaches the exact solution, $u^\delta(x) \to u(x)$. The method of residuals weighted places a restriction on the residual by applying an inner product between the governing equation, and N test (or weight) functions, $v_j(x)$, and setting it to zero,

$$(v_j(x), R[u^\delta(x)]) = 0, \quad j = 0, ..., N-1. \tag{2.4}$$

Definition 2.1.1 (Inner product)**.** The inner product between two functions $f(x)$ and $g(x)$ is,

$$(f, g) = \int_\Omega f(x)g(x)dx.$$

By ~~setting~~ enforcing equation (2.4) ~~to zero, it~~ the discrete problem becomes a system of N ordinary differential equations, ~~where~~ to solve for the N basis coefficients, $\hat{u}_i$. The choice of test function defines the projection methods, and examples of projection methods are shown in table 2.1. We emphasise that the method of weighted residuals merely describes the projection method, but does not specify the type of basis expansions, as we will discuss later in §2.3. The choice of projection method coupled with suitable basis expansions will have different solution convergence properties. ~~A~~ Of particular interest is on how quickly the residual vanishes as the number of basis expansions increases. For instance, by considering the Galerkin method coupled with Fourier expansions, one can expect exponential convergence for a sufficiently smooth problem which is desirable for an efficient representation of turbulent dynamics.

Weight functions	Projection method
$v_j(x) = \delta(x - x_j)$	Collocation
$v_j(x) = \begin{cases} 1 & \text{if } x \in \Omega_j \\ 0 & \text{if } x \notin \Omega_j \end{cases}$	Finite-Volume
$v_j(x) = \phi_j$	Galerkin
$v_j(x) = \frac{\partial R}{\partial \hat{u}_j}$	Least-squares

Table 2.1: Examples of weight functions and projection methods

2.2 Galerkin Projection

The Galerkin projection remains a standard projection method in the context of the finite element method, where the test functions, $v(x)$, are chosen to be lie in the same functional space as the global basis functions, $\Phi(x)$. To demostrate the Galerkin projection method, we consider that the differential operator earlier in equation (2.1) as a 1D Helmholtz equation,

$$\mathbf{L}[u(x)] \equiv \frac{\partial^2 u(x)}{\partial x^2} - \lambda u(x) - f(x) = 0, \quad x \in \Omega := [0, l] \tag{2.5a}$$

$$u(0) = g_D, \quad \left.\frac{\partial u}{\partial x}\right|_{x=l} = g_N. \tag{2.5b}$$

where λ is a real positive constant, $f(x)$ is a forcing function, and Ω refers to the spatial domain bounded between 0 and l. To ensure that problem is well posed, Dirichlet and Neumann boundary conditions, g_D and g_N, are imposed at $x = 0$ and $x = l$ respectively. Equation 2.5 is commonly referred to as the strong or classical form.

The subsequent step in Galerkin projection methods is take the inner product of the equation (2.5) with a test function, $v(x)$, that satisfies the homogeneous Dirichlet boundary conditions by definition, i.e. $v(0) = 0$, and setting the inner product to zero,

$$(v(x),\, \mathbf{L}[u(x)]) = \int_0^l w \left[\frac{\partial^2 u(x)}{\partial x^2} - \lambda u(x) + f(x)\right] \mathrm{d}x = 0. \tag{2.6}$$

This step is equivalent to applying the method of weighted residuals (§2.1), where $u(x)$ could refer to the approximate solution, $u^\delta(x)$. Next, we perform integration by parts,

$$\underbrace{\int_0^l \frac{\partial v}{\partial x}\frac{\partial u}{\partial x}\,\mathrm{d}x + \int_0^l \lambda v u\,\mathrm{d}x}_{a(v,u)} = \underbrace{\int_0^l v f\,\mathrm{d}x + \left[v\frac{\partial u}{\partial x}\right]_0^l}_{f(v)}. \tag{2.7}$$

This equation is typically referred to as the weak [1] form of equation (2.5). In compact notation, we define the bilinear and linear forms as,

$$a(v, u) = f(v), \tag{2.8a}$$

where $a(v, u)$ and $f(v)$ are typically refered to as the strain energy and forcing function in structural mechanics, required to remain finite. To ensure this, we restrict the choice of solutions $u(x)$ to lie in the solution space, $\mathcal{U}$, defined as

$$\mathcal{U} := \{u \,|\, u \in H^1(\Omega), u(0) = g_D\}, \tag{2.9}$$

[1] The notions of the *weak* and *strong* are refers to the smoothness (regularity) required of admissible solutions. In the weak formulation, the highest derivative involved is up to first-order, so the solution space is H^1. This space is generally larger than that of the strong formulation, which required $u \in \mathcal{H}^2(\Omega)$. Since $H^2(\Omega) \subset H^1(\Omega)$ the weak formulation imposesd a 'less stringent' constraint of the solution space of admissible functions.

where $u \in H^1$ refers to functions of u belonging to Sobolev space of order 1, and satistfying the Dirichlet condition, $u(0) = g_D$, at $x = 0$.

Definition 2.2.1 (Sobolev space)**.** We define Sobolev space of order $n \geq 1$ on Ω,

$$H^n(\Omega) = \{u \,|\, u \in L_2(\Omega), D^\alpha u \in L_2(\Omega), \forall \alpha : \alpha \leq n\},$$

where $D^\alpha u$ refers to derivatives up to order α and $L_2(\Omega)$ refers to functions that are square integrable.

Definition 2.2.2 (L_2 space)**.** The space $L_2(\Omega)$ refers to functions that are square integrable,

$$(u, u)_{L_2} = \int_\Omega |u(x)|^2 \, \mathrm{d}\Omega < \infty. \tag{2.10}$$

We consider admissible functions up to the first derivatives, the highest order derivative in the weak formulation of equation (2.6). Similarly, the space of test functions, $\mathcal{V}$, is defined as,

$$\mathcal{V} := \{v \,|\, v \in H^1, v(0) = 0\}, \tag{2.11}$$

where $v \in H^1$ ~~are~~ refers refer to test functions belonging to the Sobolev the space of order 1, and is defined to be zero, $v(0) = 0$ on Dirichlet boundary condition, $x = 0$. The generalised weak form is therefore finding $u(x) \in \mathcal{U}$, such that

$$a(v, u) = f(v), \quad \forall v \in \mathcal{V}. \tag{2.12}$$

At this point, equation (2.12) is infinite dimension as the function spaces, $\mathcal{U}$ and $\mathcal{V}$, contain infinitely many functions. To obtain an approximate solution, $u^\delta(x)$, we restrict ourselves to finite dimensional subspaces, $\mathcal{U}^\delta \subset \mathcal{U}$, and $\mathcal{V}^\delta \subset \mathcal{V}$. The problem is then to find $u^\delta \in \mathcal{U}^\delta$, such that

$$a(v^\delta, u^\delta) = f(v^\delta), \quad v^\delta \in \mathcal{V}^\delta. \tag{2.13}$$

Here, the subspaces $u^\delta \in \mathcal{U}^\delta$ and $v^\delta \in \mathcal{V}^\delta$ are not the exactly same, compare equations (2.9) and (2.11), necessary for the standard Galerkin projection procedure where they should lie in the same subspace. To ensure that they belong to the same space, we lift the solution u^δ into two parts,

$$u^\delta = u^\mathcal{H} + u^\mathcal{D}. \tag{2.14}$$

where $u^\mathcal{H} \in \mathcal{V}^\delta$ satisfies the homogeneous Dirichlet condition (e.g. is zero on Dirichlet boundaries), belonging to the same subsapce as $v^\delta \in \mathcal{V}^\delta$, while $u^\mathcal{D} \in \mathcal{U}^\delta$ satisfies the Dirichlet boundary conditions $u^\mathcal{D}(0) = g_D$. Hence, the standard Galerkin projection method is to search for the homogeneous solution, $u^\mathcal{H} \in \mathcal{V}^\delta$, such that,

$$a(v^\delta, u^\mathcal{H}) = f(v^\delta) - a(v^\delta, u^\mathcal{D}). \tag{2.15}$$

This concludes the classical Galerkin formulation. Under certain assumptions of a, a solution is guaranteed under the Lax-Milgram theorem [Lax and Milgram, 1955].

2.3 Spectral/*hp* element method

We have described the procedure for approximating a solution of a PDE using the classical Galerkin projection technique. However, the spatial discretisation scheme, related to the choice of basis (and test) functions, remains undiscussed. In this section, we discuss the spectral/*hp* element method [Patera, 1984], where the solution is partitioned into a set of non-overlapping finite elements of size h, consisting of a linear combinbation of continuous orthogonal polynomial functions up to order P. It leverages the geometric flexibility of classical finite-element methods, allowing for the representation of complex engineering geometries, and the exponential (spectral) convergence properties of classical spectral methods, where the solution error decreases exponentially. Suppose we consider $P+1$ linearly independent polynomials spanning the polynomial space of $\mathcal{P}_P$, the error of a smooth solution with element size of h and polynomial order P has the property of [Karniadakis and Sherwin, 2005],

$$||u(x) - u^{\delta}(x)|| \leq Ch^P ||u(x)|| \approx O(h^P). \tag{2.16}$$

where C is some constant. Equation 2.16 implies that the error decreases linearly with h, and exponentially with P. This section is organised into domain partition, standard elements, assembly process, modal and nodal expansion functions, numerical integration and differentiation, concluding with an example in 1D.

2.3.1 Domain partition

The first step concerns the partitioning the domain into a set of (finite) elemental regions. We consider an example in one dimension within Ω, and partition it into a set of N_{el} elements, where Ω^e, refers to the elemental partitions with $1 \geq e \geq N_{el}$, such that they meet at their boundaries and do not overlap,

$$\Omega = \bigcup_{e=1}^{N_{el}} \Omega^e, \quad \text{where} \bigcap_{e=1}^{N_{el}} \Omega^e = \emptyset \tag{2.17}$$

where the e^{th} element is defined as,

$$\Omega^e = \{x \,|\, x_{e-1} \geq x \geq x_e\}. \tag{2.18}$$

Each element can be represented by a linear combination of orthogonal basis expansions. The basis expansions can be either modal or nodal expansions, as we shall see later.

2.3.2 Standard Elements

In general, we expect to work with non-uniform elements that may have arbitrarily shapes, making the definition of basis expansions potentially unwieldy. To simplify the formulation, it is convenient to define a *standard* element,

$$\Omega_{st} = \{\xi \,|\, -1 \geq \xi \geq 1\}, \tag{2.19}$$

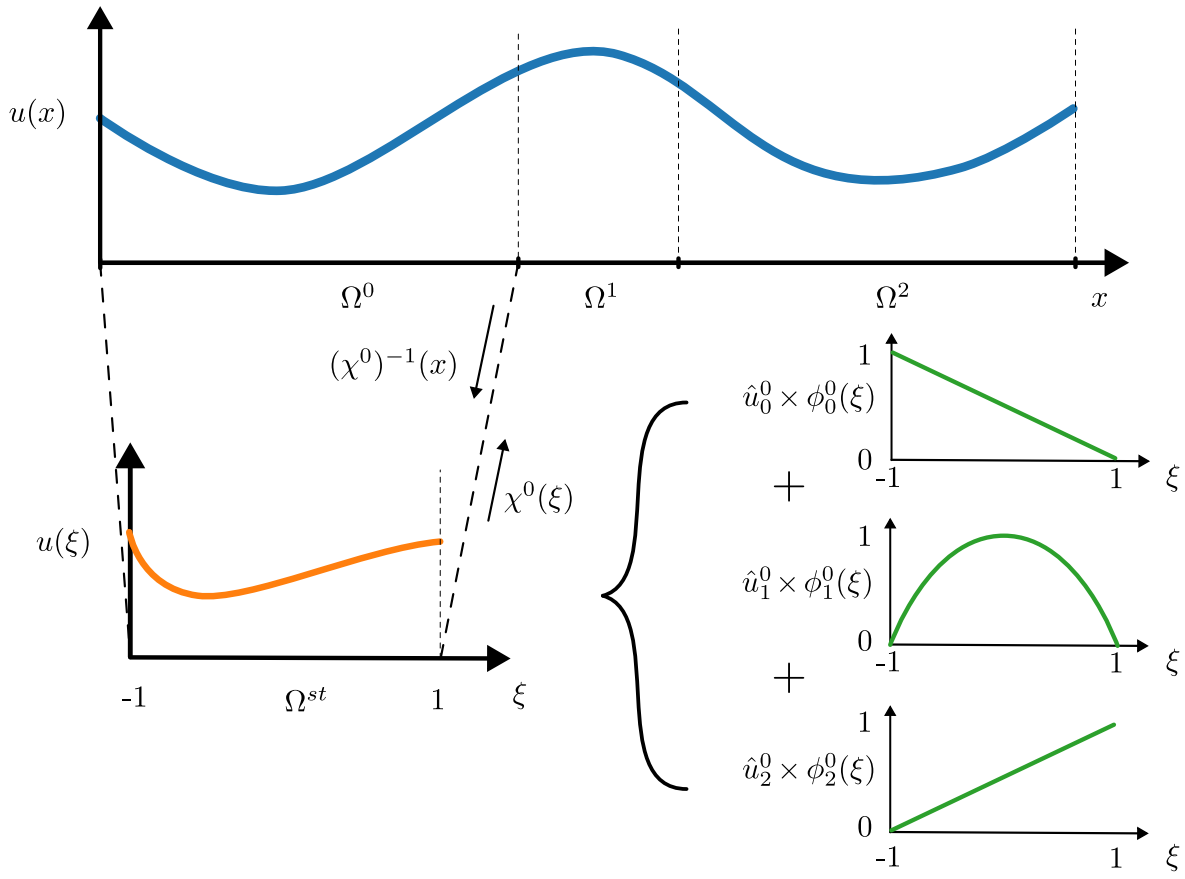

Figure 2.1: A spectral/*hp* element representation of $u(x)$, consisting of three non-overlapping finite elements, each containing a linear combination of local expansion bases of up to $P = 2$.

where Ω_{st} refers to the standard element defined in local coordinates, $\xi \in [-1, 1]$. Within this standard element, the formulation of basis expansions, as well as differential and integration operations, can be carried out in the local coordinate system ξ, before mapping the solution back to the global domain, x. We can map the standard element into any arbitrary global coordinates based on a linear mapping $\chi^e : \Omega_{st} \to \Omega$,

$$x = \chi^e(\xi) = \frac{1-\xi}{2} x_e + \frac{1+\xi}{2} x_{e+1}, \quad \xi \in \Omega_{st} \tag{2.20}$$

for this case
which has an analytical inverse, $(\chi^e)^{-1}(x)$,

$$\xi = (\chi^e)^{-1}(x) = 2\frac{x - x_{e-1}}{x_e - x_{e-1}} - 1, \quad x \in \Omega^e. \tag{2.21}$$

For illustration purposes, we consider that the standard element can be represented by three local basis expansions of polynomial order of up to $P = 2$,

$$\phi_0^e(\xi) = \frac{1-\xi}{2}, \quad \phi_1^e(\xi) = (1+\xi)(1-\xi), \quad \phi_2^e(\xi) = \frac{1+\xi}{2}, \tag{2.22}$$

where ϕ_0^e, ϕ_2^e and ϕ_1^e refers to the linear and quadratic local basis expansions of the e^{th} element. These local basis expansions is illustrated in figure 2.1. We note that the formulations of local basis

expansion here is merely an example. In practice, the local basis expansions are usually chosen to have orthogonality properties under a certain inner product. The approximate solution is now represented as,

$$u^{\delta}(x) = \sum_{e=0}^{N_{el}-1} \sum_{i=0}^{P} \hat{u}_i^e \phi_i^e(\chi^e(\xi)). \tag{2.23}$$

where $\hat{u}_i^e$ refers to the local expansion basis coefficients. The approximate solution, $u^{\delta}(x)$, now lie within the solution space $\mathcal{U}^{\delta}$ defined as,

$$\mathcal{U}^{\delta} := \left\{ u^{\delta} \,\middle|\, u^{\delta} \in H^1,\ u^{\delta}(\chi^e(\xi)) \in \phi_i^e(\xi), \forall i : 0 \leq i \leq P, \forall e : 0 \leq e \leq N_{el} \right\} \tag{2.24}$$

2.3.3 Global assembly

In this section, we introduce the concept of global assembly (or direct stiffness summation) which relates the global basis expansions (equation (2.2)), $\Phi_i(x)$, to the local basis expansions (equation (2.23)), $\phi_i^e(x)$, where the solution can be approximated using either formulation,

$$u^{\delta}(x) = \sum_{i=0}^{N-1} \hat{u}_i \Phi_i(x) = \sum_{e=0}^{N_{el}-1} \sum_{i=0}^{P} \hat{u}_i^e \phi_i^e(\chi^e(\xi)). \tag{2.25}$$

In general, we can represent the global and local basis coefficients each as a column vector,

$$\hat{\mathbf{u}}_g = \begin{pmatrix} \hat{u}_0 \\ \vdots \\ \hat{u}_N \end{pmatrix}, \quad \hat{\mathbf{u}}_l = \begin{pmatrix} \hat{\mathbf{u}}^0 \\ \vdots \\ \hat{\mathbf{u}}^{N_{el}-1} \end{pmatrix}, \tag{2.26}$$

where $\hat{\mathbf{u}}^e = (\hat{u}_0^e, ..., \hat{u}_P^e)^T$, $\hat{\mathbf{u}}_g \in \mathbb{R}^N$, $\hat{\mathbf{u}}_l \in \mathbb{R}^{N_{loc}}$ and $N_{loc} = N_{el}(P+1)$. As there can be more ~~global~~ local degrees of freedom than ~~local~~ global degrees of freedom, N ~~>~~ ≤ N_{loc}, we need to impose some conditions on the local expansion coefficients. One of the common approach is to enforce C^0 continuity across elemental boundaries, referred to as the continous Galerkin projection. Following the definition of local basis expansions in equation (2.22), this condition can be supplemented using,

$$\hat{u}_P^{e-1} = \hat{u}_0^e. \tag{2.27}$$

The graphical representation of this condition enforcing C^0 continuity between the element boundaries for three finite elements with $P = 2$ local basis expansions, and the relationship between global and local basis coefficients are shown in figure 2.2. We can relate the global and local basis coefficients with an assembly matrix, $\mathbf{A} \in \mathbb{R}^{N_{loc} \times N}$,

$$\hat{\mathbf{u}}_l = \mathbf{A}\hat{\mathbf{u}}_g. \tag{2.28}$$

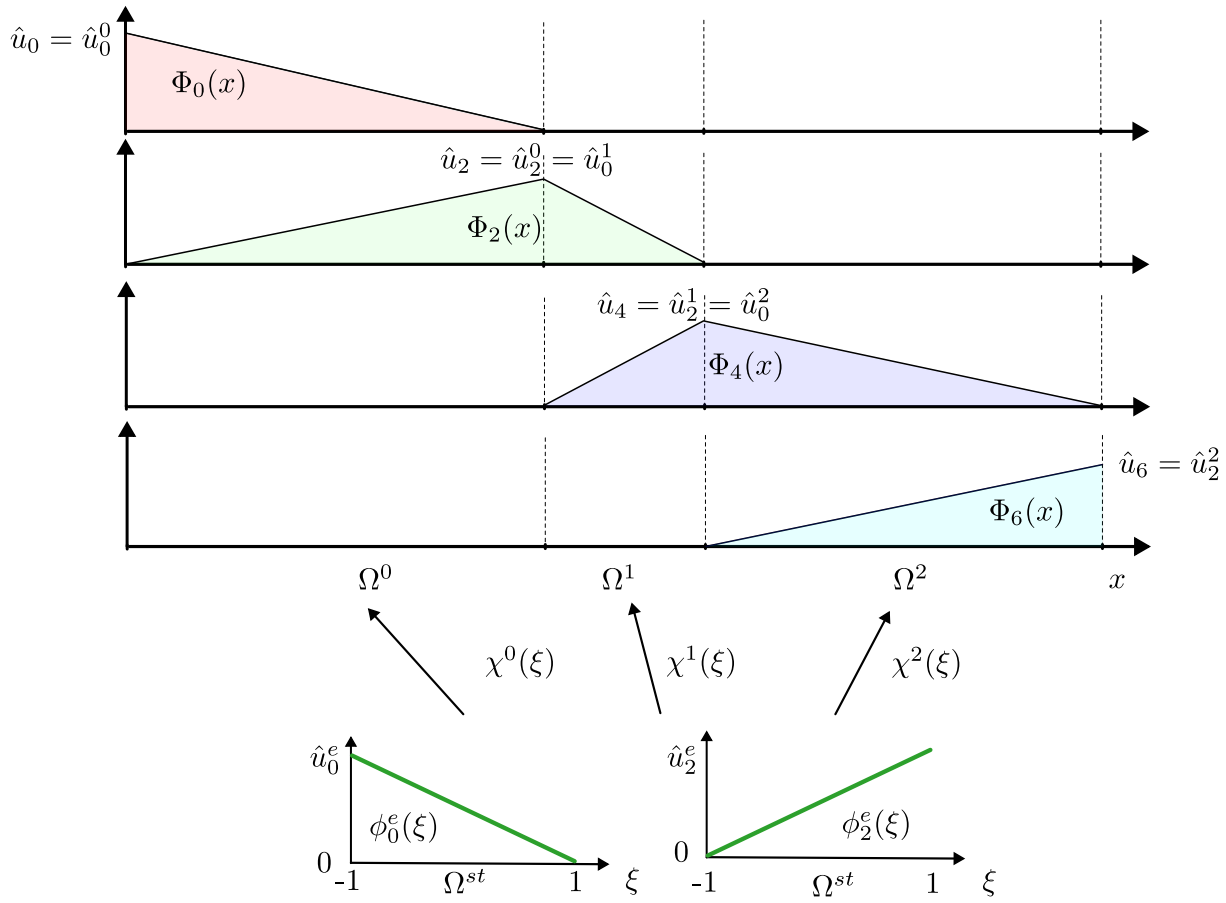

Figure 2.2: A graphical representation of C^0 across elemental boundaries and the relationship between local basis coefficients, u_0^e, u_P^e, and global basis expansions, u_i.

In the case for $P = 2$ and three finite elements as in the case of figures 2.1and 2.2, the assembly matrix and the vectors of global and local basis coefficients are given as,

$$\hat{\mathbf{u}}_l = \begin{pmatrix} \hat{u}_0^0 \\ \hat{u}_1^0 \\ \hat{u}_2^0 \\ \hat{u}_0^1 \\ \hat{u}_1^1 \\ \hat{u}_2^1 \\ \hat{u}_0^2 \\ \hat{u}_1^2 \\ \hat{u}_2^2 \end{pmatrix}, \quad \mathbf{A} = \begin{pmatrix} 1\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0\,1 \end{pmatrix}, \quad \hat{\mathbf{u}}_g \begin{pmatrix} \hat{u}_0 \\ \hat{u}_1 \\ \hat{u}_3 \\ \hat{u}_4 \\ \hat{u}_5 \\ \hat{u}_6 \end{pmatrix}, \tag{2.29}$$

The assembly matrix $\mathbf{A}$ 'scatters' the global degrees of freedom to local degrees of freedom, while the transpose of it, $\mathbf{A}^T$, performs the reverse, referred to as global assembly. For example, we wish to perform integration in the domain Ω,

$$\mathbf{I}_g[j] = (\Phi_j(x), u^\delta(x)), \tag{2.30}$$

where $\mathbf{I}_g \in \mathbb{R}^N$ refers to a vector containing the integral between $\Phi_i(x)$ and $u^\delta(x)$. This is related to first performing integration using local expansion basis within standard elements, and then assembling using $\mathbf{A}^T$,

$$\mathbf{I}_g = \mathbf{A}^T \mathbf{I}_l, \tag{2.31a}$$

where,

$$\mathbf{I}_g = \begin{bmatrix} \mathbf{I}_0 \\ \vdots \\ \mathbf{I}_{N_g-1} \end{bmatrix}, \quad \mathbf{I}_l = \begin{bmatrix} \mathbf{I}^0 \\ \vdots \\ \mathbf{I}^{N_{el}-1} \end{bmatrix}, \quad \text{with} \quad \mathbf{I}^e = \begin{bmatrix} \int_{-1}^{1} \phi_0^e(\xi) u(\chi^e) \frac{\mathrm{d}\chi^e}{\mathrm{d}\xi} \, \mathrm{d}\xi \\ \vdots \\ \int_{-1}^{1} \phi_{P-1}^e(\xi) u(\chi^e) \frac{\mathrm{d}\chi^e}{\mathrm{d}\xi} \, \mathrm{d}\xi \end{bmatrix}, \tag{2.31b}$$

and $\mathbf{I}_l \in \mathbb{R}^{N_{loc}}$ refer to the vector of integration operations performed within a standard element. In the spectral/*hp* element approach, we perform integration and differentiation using local basis expansions within a standard element. After doing so, we assemble the local operations from the standard element to the global domain by using $\mathbf{A}^T$, as we shall show later using a 1D example. We note that the structure of assembly matrix is generally sparse, where the entries either contain 0, 1 or -1 in multidimensional formulation. Therefore, the assembly matrix is not constructed in practice, and a mapping array is used instead.

2.3.4 Local basis expansions

The choice of local basis expansions, $\phi_i^e(\xi)$, concerns the representation of the solution, and the convergence properties of the numerical solver, in particular, the condition number of the mass and laplacian matrices. In general, the the local basis expansions can be classified into two groups, either *modal* or *nodal* expansions.

Modal expansions

Modal expansions, or hierarchical expansions, describes a set of expansion basis where an expansion set ($\mathcal{X}_{P-1}^\delta$) of order $P-1$, is contained within a set ($\mathcal{X}_P^\delta$) of order P, e.g. $\mathcal{X}_{P-1}^\delta \subset \mathcal{X}_P^\delta$. An example of modal expansions are the Jacobi polynomials, $P_p^{\alpha,\beta}(x)$, representing a family of solutions to the Sturm-Liouville problem within, $x \in [-1, 1]$. The Jacobi polynomials become symmetric for $\alpha = \beta$, referred to ultraspheric polynomials. Special cases of ultraspheric polynomials are the Legendre polynomials, $\alpha = \beta = 1$, and the Chebyshev polynomials, $\alpha = \beta = 1/2$. Within the Nektar++ framework, we utilise the *modified* basis, constructed using on the Jacobi polynomials and modified (hence its name) by linear expansions given as,

$$\phi_p(\xi) \to \psi_p(\xi) = \begin{cases} \frac{1-\xi}{2} & \text{for } p = 0 \\ \left(\frac{1-\xi}{2}\right)\left(\frac{1+\xi}{2}\right) P_{p-1}^{1,1}(\xi) & \text{for } 0 < p < P \\ \frac{1+\xi}{2} & \text{for } p = P, \end{cases} \tag{2.32}$$

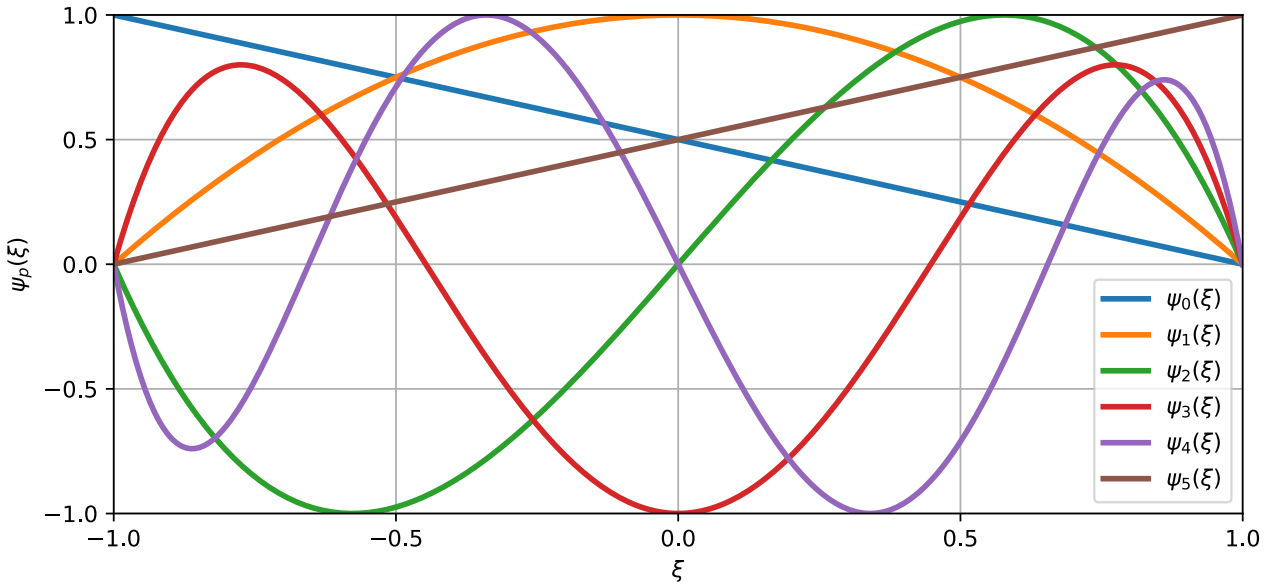

Figure 2.3: The modified basis for up to $P = 5$ normalised to $-1 \leq \psi_p \leq 1$.

We note that $\phi_p(\xi)$ refers to a general local expansion basis while $\psi_p(\xi)$ to definition of the modified basis. The one-dimensional expansion modes of the modified basis of up to $P = 5$ is shown in figure 2.3. The linear modes, corresponding to $p = 0$ and $p = P$, are the only expansions which has a magnitude of at the boundaries, referred to as boundary modes. The modified basis for $0 < p < P$, are clearly hierarchical, and have non-zero values except at the boundaries, referred to as interior/bubble modes.

Nodal expansions

Nodal expansions are basis expansions that are non-hierchical, $\mathcal{X}^{\delta}_{P-1} \not\subset \mathcal{X}^{\delta}_{P}$. An example of nodal expansions are the Lagrange polynomials,

$$\phi_p(\xi) \to h_p(\xi) = \frac{\prod_{q=0,q\neq p}^{P}(\xi - \xi_q)}{\prod_{q=0,q\neq p}^{P}(\xi_p - \xi_q)} \tag{2.33}$$

The Lagrange polynomials, $h_p(\xi)$, are particular attractive as it has a unit value at discrete nodal values, ξ_q, and zero everywhere else, $h_p(\xi_q) = \delta_{pq}$, which implies that

$$u^{\delta}(\xi_q) = \sum_{p=0}^{P} \hat{u}_p h_p(\xi_q) = \sum_{p=0}^{P} \hat{u}_p \delta_{pq} = \hat{u}_q, \tag{2.34}$$

where the Lagrange coefficient $\hat{u}_q$ is the same as the value evaluated at the node ξ_q. The nodal values, ξ_q, are based on the Gauss-Lobatto-Legendre (GLL) points which will be defined later in §2.3.5. Figure 2.4 presents Lagrange expansions evaluated along the GLL points.

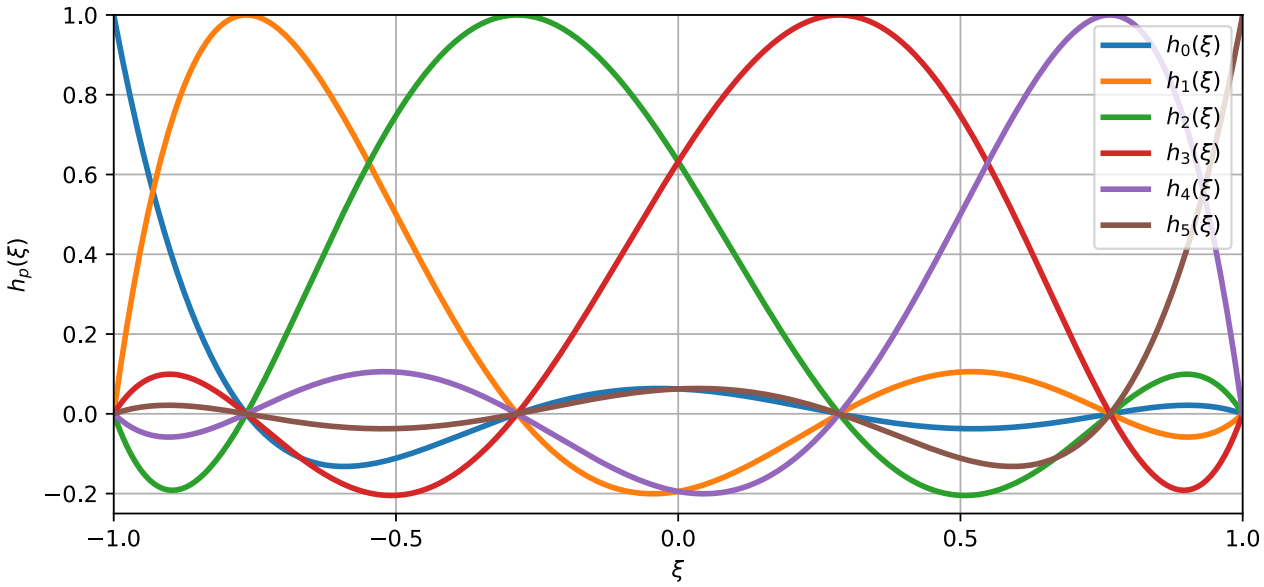

Figure 2.4: Lagrange polynomials for $P = 5$ with nodal values along GLL points.

Multi-dimensional expansions

We have introduced modal and nodal expansions in one dimension, and its extension to multi-dimensions bases can be generalised using a tensorial expansion of the local expansion bases. The standard element in a two dimensional quadrilateral, $\mathcal{Q}^2$, and a three dimensional hexahedral $\mathcal{H}^3$, are given as,

$$\mathcal{Q}^2 = \{-1 \leq \xi_1, \xi_2 \leq 1\}, \quad \mathcal{H}^2 = \{-1 \leq \xi_1, \xi_2, \xi_3 \leq 1\} \tag{2.35}$$

where ξ_1, ξ_2, ξ_3 refers to the local coordinates in multi-dimensions. Thus, the multi-dimensional expansion bases for quadrilaterals and hexadrals using modified bases are simply a tensor product of the one dimensional modified bases,

$$\phi_{pq}(\xi_1, \xi_2) = \psi_q(\xi_1)\psi_q(\xi_2), \quad \text{and} \quad \phi_{pqr}(\xi_1, \xi_2, \xi_3) = \psi_q(\xi_1)\psi_q(\xi_2)\psi_r(\xi_3). \tag{2.36}$$

An example of the modal tensorial bases, for $p = q = 4$ in a standard quadrilateral element in shown in figure 2.5. While we have discussed the tensorial the expansions for regular domains such as the standard quadrilateral and hexahedral elements, the extensions for simplex domains such as triangles, tehtrahedrals, prisms and pyramids commonly used to represent complex geometries, are less straightforward. The challenge for simplexes is that the local coordinates, ξ_1, ξ_2, ξ_3, become dependent where a direct tensorial expansion becomes unwieldy. Instead, a collapsed coordinate system is introduced, providing a transformation from a standard simplex element to a standard regular element. In this thesis, we ultilise quadrilateral elements. The reader is referred to Karniadakis and Sherwin [2005] for more details about the multi-dimensional formulation of regular and simplex elements.

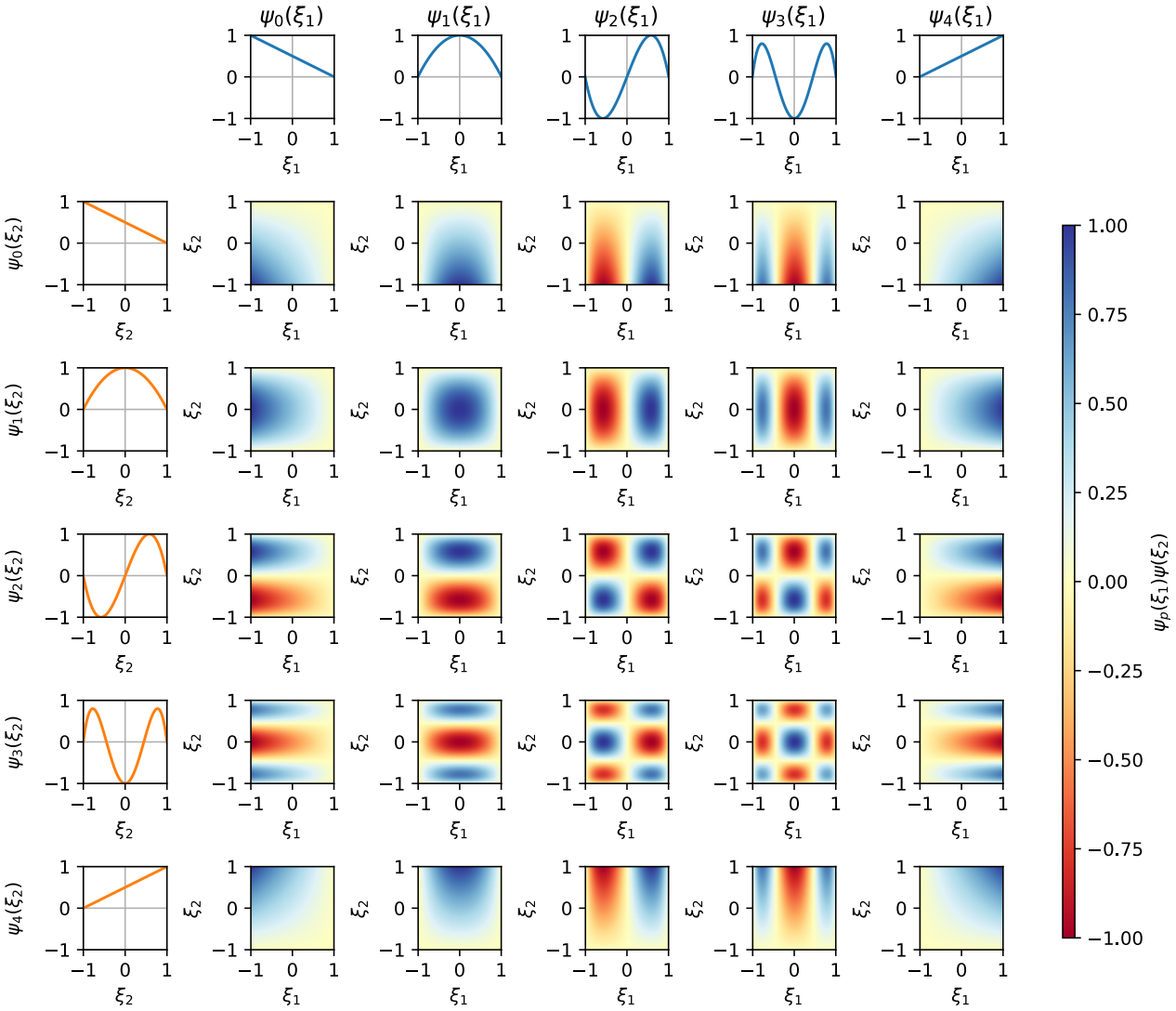

Figure 2.5: Two dimensional modified basis with $p = q = 4$ in a standard quadrilateral, $-1 \leq \xi_1, \xi_2 \leq 1$. The modified bases are normalised to $-1 \leq \phi_{pq} \leq 1$.

2.3.5 Gaussian quadrature

In the Galerkin formulation, we perform integration between basis functions rountinely, and an efficient numerical technique is sought ~~after.~~ thereofre Suppose we want to approximate the integral of a function, $u(\xi)$, in a standard element numerically given as,

$$\int_{-1}^{1} u(\xi)\, \mathrm{d}\xi = \sum_{i=0}^{Q-1} w_i u(\xi_i) + R(u). \tag{2.37}$$

The premise is to determine the optimal number of quadrature points, Q, integration weights, w_i, and zeros, ξ_i, in which the integral error, $R(u)$, can be minimised. If $u(\xi)$ is of polynomial order of P, we may expect that we require at least $P + 1$ equipspaced points to represent $u(\xi)$ sufficiently. Using Gaussian quadrature rules, we can approximate an integral of a function of order P, with far fewer that $P + 1$ points with specific integration weights and zeros. In general, Gaussian quadrature rules

can be grouped into three catergories: Gauss, Gauss-Radau and Gauss-Lobatto quadrature. The main difference between the three categories are on the inclusion of the end points. Gauss quadrature rule evaluates the integral without the end points $\xi = \pm 1$. Gauss-Radau quadrature rule either select one of the end points, ~~typically~~ often at $\xi = -1$. Gauss-Lobatto quadrature rule consider both end points. We will only focus on describing the Gauss-Lobatto quadrature rules and the zeros of Jacobi polynomials known as the Gauss-Lobatto-Jacobi quadrature rules given as,

$$\xi_i^{\alpha,\beta} = \begin{cases} -1 & i = 0, \\ \xi_{i-1,Q-2}^{\alpha+1,\beta+1} & i = 1, ..., Q-2, \\ 1, & i = Q-1, \end{cases} \tag{2.38a}$$

$$w_i^{\alpha,\beta} = \begin{cases} (\beta+1)C_{0,Q-2}^{\alpha,\beta}, & i = 0, \\ C_{i,Q-2}^{\alpha,\beta}, & i = 1, ..., Q-2, \\ (\alpha+1)C_{Q-1,Q-2}^{\alpha,\beta}, & i = Q-1, \end{cases} \tag{2.38b}$$

$$C_{i,Q-2}^{\alpha,\beta} = \frac{2^{\alpha+\beta+1}\Gamma(\alpha+Q)\Gamma(\beta+Q)}{(Q-1)(Q-1)!\Gamma(\alpha+\beta+Q+1)[P_{Q-1}^{\alpha,\beta}(\xi_i)]^2} \tag{2.38c}$$

where $w_i^{\alpha,\beta}, \xi_i^{\alpha,\beta}$ are the zeros (or sometimes referred to as quadrature points) and weights of the Gauss-Lobatto-Jacobi quadrature rules, and Γ refers to the Gamma function. For $\alpha = \beta = 0$, the quadrature points is known as the Gauss-Lobatto-Legendre (GLL) points, typically employed for Lagrange polynomials. By evaluating the integral using the zeros and integrations weights defined above, we can obtain an exact integral of the function $u(\xi)$, of polynomial order P, with at least $Q \geq (P+3)/2$ quadrature points.

2.3.6 Numerical differentiation

In the same fashion as Gaussian quadrature rules, we want to evaluate the derivative of a function, $u^\delta(\xi)$ numerically. Suppose that we want to differentiate in x using local coordinates given as,

$$\frac{\mathrm{d}u^\delta(\xi)}{\mathrm{d}x} = \frac{\mathrm{d}u^\delta(\xi)}{\mathrm{d}\xi}\frac{\mathrm{d}\xi}{\mathrm{d}x} = \sum_{p=0}^{P} \hat{u}_p \frac{\mathrm{d}\phi_p(\xi)}{\mathrm{d}\xi}\frac{\mathrm{d}\xi}{\mathrm{d}x}, \tag{2.39}$$

where $\mathrm{d}\xi/\mathrm{d}x$ is the jacobian. The main step ~~in~~ involves evaluating the derivative of the local expansion bases, $\mathrm{d}\phi_p(\xi)/\mathrm{d}\xi$, referred to as collocation differentiation ~~as~~ since differentiation is performed in physical space. Suppose that we express the solution of polynomial order P with modified polynomials, $\phi_p(\xi) \rightarrow \psi_p(\xi)$, through a set of $Q \geq P+1$ quadrature points, the derivative in discrete local coordinates is expressed as,

(nodal)

$$\frac{\mathrm{d}u^\delta(\xi)}{\mathrm{d}\xi}\Big|_{\xi=\xi_i} = \sum_{j=0}^{Q-1} \hat{u}_j \frac{\mathrm{d}\psi_j(\xi)}{\mathrm{d}\xi}\Big|_{\xi=\xi_i} = \sum_{j=0}^{Q-1} D_{ij}\hat{u}_j, \tag{2.40}$$

where D_{ij} refers to the differential matrix containing values of the derivative of the basis at discrete quadrature points given as,

$$D_{ij} = \frac{\mathrm{d}h_j(\xi)}{\mathrm{d}\xi}\Big|_{\xi=\xi_i}, \tag{2.41}$$

which is often pre-computed. To differentiate a function, $u(\xi)$, we typically need to construct the differential matrices, and the general representation of the differential matrix is,

$$D_{ij} = \begin{cases} \frac{p'_Q(\xi_i)}{p'_Q(\xi_j)}\frac{1}{\xi_i-\xi_j}, & i \neq j, \\ \frac{p''_Q(\xi_i)}{2p'_Q(\xi_i)}, & i = j. \end{cases} \tag{2.42}$$

where $p'_Q(\xi_i), p''_Q(\xi_i)$ refers to the first and second differentive of Jacobi polynomials evaluated at the quadrature points ξ_i.

2.3.7 Example in 1D

We have outlined the basic formulation of spectral/*hp* element methods in a single dimension. To conclude the section on spectral/*hp* element methods, we will describe its solution procedure, converting the weak form of the Helmholtz equation into a system of linear equations, and introduce the mass and laplacian matrices. Starting from the weak form,

$$\lambda \underbrace{\int_{-1}^{1} v^\delta u^\mathcal{H}\,\mathrm{d}\xi}_{\mathbf{M}^e\hat{\mathbf{u}}^e} + \underbrace{\int_{-1}^{1} \frac{\partial v^\delta}{\partial \xi}\frac{\partial u^\mathcal{H}}{\partial \xi}\mathrm{d}\xi}_{\mathbf{L}^e\hat{\mathbf{u}}^e} = \underbrace{\int_{-1}^{1} v^\delta f\,\mathrm{d}\xi}_{\hat{\mathbf{f}}^e} - \underbrace{\int_{-1}^{1} \frac{\partial v^\delta}{\partial \xi}\frac{\partial u^\mathcal{D}}{\partial \xi}\,\mathrm{d}\xi}_{\mathbf{L}^0} + v(l)g_N, \tag{2.43}$$

we wish to seek the solution $u^\mathcal{H}(\xi)$. Recall that the solution space of $u^\mathcal{H}(\xi)$ and $v^\delta(\xi)$ are the same, following the standard Galerkin projection procedure. Suppose they can be discretised by spectral/*hp* elements with e elements and local basis expansions of up to polynomial order P,

$$u^\mathcal{H}(\xi) = \sum_{i=0}^{P} \hat{u}_i^e \phi_i^e(\xi), \quad v^\delta(\xi) = \sum_{i=0}^{P} \hat{v}_i^e \phi_i^e(\xi). \tag{2.44}$$

Substituting into equation (2.43) and evaluating the first term on the left hand side through a set of Q quadrature points. The e^{th} elemental contribution can be evaluated as:

$$\begin{aligned} \int_{-1}^{1} \sum_{i=0}^{P} \hat{v}_i^e \phi_i^e(\xi) \sum_{i=0}^{P} \hat{u}_i^e \phi_i^e(\xi)\,\mathrm{d}\xi &= \sum_{q=0}^{Q-1} \left[\sum_{i=0}^{P} \hat{v}_i^e \phi_i^e(\xi_q) \sum_{i=0}^{P} \hat{u}_i^e \phi_i^e(\xi_q)\right] w_q^e \\ &= (\hat{\mathbf{v}}^e)^T (\mathbf{B}^e)^T \mathbf{W}^e \mathbf{B}^e \hat{\mathbf{u}}^e \\ &= \hat{\mathbf{v}}^T \mathbf{M}^e \hat{\mathbf{u}}^e \end{aligned} \tag{2.45}$$

where $\mathbf{M}^e = (\mathbf{B}^e)^T \mathbf{W} \mathbf{B}^e \in \mathbb{R}^{(P+1)\times(P+1)}$ refers to the elemental mass matrix, while $\mathbf{B}^e \in \mathbb{R}^{Q\times(P+1)}$ refers to the elemental basis matrix, and $\mathbf{W}^e \in \mathbb{R}^{Q\times Q}$, the elemental weight matrix, a diagonal matrix

consisting of discrete integration weights along Q quadrature points.

$$\mathbf{B}^e = \begin{bmatrix} \phi_0(\xi_0) & \cdots & \phi_P(\xi_0) \\ \vdots & \ddots & \vdots \\ \phi_0(\xi_Q) & \cdots & \phi_P(\xi_Q) \end{bmatrix}, \quad \mathbf{W}^e = \begin{bmatrix} w_0^e & & 0 \\ & \ddots & \\ 0 & & w_{Q-1}^e \end{bmatrix} \tag{2.46}$$

Next, we move onto the second term on the left hand side,

$$\begin{aligned} \int_{-1}^{1} \sum_{i=0}^{P} \hat{v}_i^e \frac{\mathrm{d}\phi_i^e}{\mathrm{d}\xi} \sum_{i=0}^{P} \hat{u}_i^e \frac{\mathrm{d}\phi_i^e}{\mathrm{d}\xi} \,\mathrm{d}\xi &= \sum_{q=0}^{Q} \left[\sum_{i=0}^{P} \hat{v}_i^e D_{qi}^e \phi_i^e(\xi_q) \sum_{i=0}^{P} \hat{u}_i^e D_{qi}^e \phi_i^e(\xi_q) \right] w_q^e \\ &= \hat{\mathbf{v}}^T (\mathbf{B}^e)^T (\mathbf{D}^e)^T \mathbf{W}^e \mathbf{D}^e \mathbf{B}^e \hat{\mathbf{u}}^e \\ &= \hat{\mathbf{v}}^T \mathbf{L}^e \hat{\mathbf{u}}^e \end{aligned} \tag{2.47}$$

where $\mathbf{L}^e = (\mathbf{B}^e)^T (\mathbf{D}^e)^T \mathbf{W} \mathbf{D}^e \mathbf{B}^e \in \mathbb{R}^{(P+1)\times(P+1)}$ refers to the elemental Laplacian matrix while $\mathbf{D}^e \in \mathbb{R}^{Q\times(P+1)}$ refers to the differential matrix defined in equation (2.42). Moving onto the first term on the right hand side,

$$\begin{aligned} \int_{-1}^{1} \sum_{i=0}^{P} \hat{v}_i^e \phi_i^e(\xi) f^e(\xi) \,\mathrm{d}\xi &= \sum_{q=0}^{P} \sum_{i=0}^{P} \hat{v}_i^e \phi_i^e(\xi_q) f^e(\xi_q) w_q^e, \\ &= \hat{\mathbf{v}}^T (\mathbf{B}^e)^T \mathbf{W}^e \mathbf{f}^e \\ &= \hat{\mathbf{v}}^T \hat{\mathbf{f}}^e, \end{aligned} \tag{2.48}$$

where $\hat{\mathbf{f}}^e$, is referred to the elemental forcing vector. As we bolt the elemental laplacian, mass matrices, and forcing vector, the system of linear equations within a standard element is given as,

$$[\lambda \mathbf{M}^e + \mathbf{L}^e] \, \hat{\mathbf{u}}^e = \hat{\mathbf{f}}^e. \tag{2.49}$$

We note that the boundary conditions have been omitted. To include the boundary conditions, we consider the full system of linear of equations consisting ofe number of elements,

$$\lambda \underbrace{\begin{bmatrix} \mathbf{M}^0 + \mathbf{L}^0 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \mathbf{M}^{N_{el}-1} + \mathbf{L}^{N_{el}-1} \end{bmatrix}}_{\mathbf{M}_l + \mathbf{L}_l} \underbrace{\begin{bmatrix} \hat{\mathbf{u}}^0 \\ \vdots \\ \hat{\mathbf{u}}^{N_{el}-1} \end{bmatrix}}_{\hat{\mathbf{u}}_l} = \underbrace{\begin{bmatrix} \hat{\mathbf{f}}^0 \\ \vdots \\ \hat{\mathbf{f}}^{N_{el}-1} \end{bmatrix}}_{\hat{\mathbf{f}}_l} + \underbrace{\begin{bmatrix} \mathbf{L}^0 g_D \\ \vdots \\ \mathbf{0} \end{bmatrix}}_{\mathbf{g}_D} + \underbrace{\begin{bmatrix} \mathbf{0} \\ \vdots \\ g_N \end{bmatrix}}_{\mathbf{g}_N}, \tag{2.50}$$

where $\mathbf{M}_l \in \mathbb{R}^{N_{el}(P+1)\times N_{el}(P+1)}, \mathbf{L}_l \in \mathbb{R}^{N_{el}(P+1)\times N_{el}(P+1)}, \hat{\mathbf{u}}_l \in \mathbb{R}^{N_{el}(P+1)}$ refers to the local mass matrix, local laplacian matrix and the vector of local expansion coefficients. On the right hand side, $\hat{\mathbf{f}}_l \in \mathbb{R}^{N_{el}(P+1)}, \mathbf{g}_D^{N_{el}(P+1)}, \mathbf{g}_N^{N_{el}(P+1)}$ refers local forcing vector, Dirichlet and Neumann boundary conditionsin vector form. Lastly, to ensure that the solution remains C^0 continuous across the elemental

boundaries, we perform the assembly process by using the assemble matrices (see §2.3.3),

$$\lambda \mathbf{A}^T(\mathbf{M}_l + \mathbf{L}_l)\mathbf{A}\hat{\mathbf{u}}_g = \mathbf{A}^T(\hat{\mathbf{f}}^l + \mathbf{g}_D + \mathbf{g}_N), \tag{2.51}$$

and obtain the solution for $\hat{\mathbf{u}}_g$.

2.4 Numerical techniques for solving the Navier-Stokes equations

2.4.1 Velocity Correction Scheme

The spatial discretisation of the Helmholtz operator and its numerical solution procedure has been discussed using the spectral/*hp* element methods. Here, we describe the numerical methods that is used to solve the Navier-Stokes equations given as,

$$\frac{\partial u}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f} \tag{2.52a}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2.52b}$$

with boundary conditions,

$$\mathbf{u} = 0, \quad \text{on } \partial\Omega. \tag{2.52c}$$

In the above

~~Here,~~ the primitive variables are velocity and pressure $(\mathbf{u}, p)$ and we assumed unit density, $\rho = 1$, with the kinematic viscosity appearing as the control parameter. The time evolution of velocity is ~~explicit~~ expressed in equation (2.52a), but does not appear for the pressure, which is coupled to the velocity field, enforcing the incompressibility condition. Several strategies exist for addressing the coupled velocity-pressure fields by

1. Solving the coupled system such as the Uzawa algorithm,
2. Splitting methods,
3. Change of coordinates (e.g. vorticity-streamfunction approach).

We adopt splitting methods, which solves the of the Navier-Stokes equation by splitting them into 'subequations', and solving them sequentially. These methods, belonging to the broader family of projection methods introduced by Chorin [1967] and Témam [1969], and can be further classified into pressure-correction or velocity-correction schemes. This thesis employs the use of the high-order velocity-correction scheme introduced by Karniadakis et al. [1991]. We rewrite the incompressible Navier-Stokes equations in semi-discrete form with using linear and nonlinear operators as,

and further explained by Guermond & Shen 2001

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{N}(\mathbf{u}) - \nabla p + \nu\mathbf{L}(\mathbf{u}), \tag{2.53a}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2.53b}$$

with boundary conditions,

$$\mathbf{u}|_{\Omega} = 0, \quad \mathbf{u}(t=0) = \mathbf{u}_0. \tag{2.53c}$$

The nonlinear, $\mathbf{N}$, linear, $\mathbf{L}$, operators are obtained from a suitable spatial-discretisation method such as the spectral/*hp* element method. The nonlinear and linear operators are defined as,

$$\mathbf{N}(\mathbf{u}) \equiv -(\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{2}\left[(\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla\cdot(\mathbf{u}\mathbf{u})\right], \qquad \mathbf{L}(\mathbf{u}) \equiv \nabla^2\mathbf{u}, \tag{2.54}$$

We note that the nonlinear terms are written in the skew-symmetric to minimise aliasing errors [Karniadakis et al., 1991]. To advance the velocity at time step, $\mathbf{u}^n$, to the next time step, $\mathbf{u}^{n+1}$, we integrate equation (2.53) over a time step Δt,

$$\mathbf{u}^{n+1} - \mathbf{u}^n = \underbrace{\int_{t_n}^{t_{n+1}} \mathbf{N}(\mathbf{u})\,\mathrm{d}t}_{\Delta t\sum_{q=0}^{J_e-1}\beta_q\mathbf{N}(\mathbf{u}^{n-q})} - \underbrace{\int_{t_n}^{t_{n+1}} \nabla p\,\mathrm{d}t}_{\Delta t\nabla\bar{p}^{n+1}} + \nu \underbrace{\int_{t_n}^{t_{n+1}} \mathbf{L}(\mathbf{u})\,\mathrm{d}t}_{\Delta t\sum_{q=0}^{J_i-1}\gamma_q\mathbf{L}(\mathbf{u}^{n+1-q})}. \tag{2.55}$$

The velocity correction scheme evaluates the underbraced terms in three successive independently from left to right independently, effectively 'splitting' equation (2.53) from this point onwards. The first step we perform is to extrapolate the advection velocities, by approximating the nonlinear terms using an explicit scheme such as the Adams-Bashforth family of J_e order,

$$\frac{\hat{\mathbf{u}} - \sum_{q=0}^{J_e-1}\alpha_q\mathbf{u}^{n-q}}{\Delta t} = \sum_{q=0}^{J_e-1}\beta_q\mathbf{N}(\mathbf{u}^{n-q}), \tag{2.56}$$

where $\hat{\mathbf{u}}$ is denotes the primary intermediate velocity field desired and α_e, β_e refers to the time integration coefficients for a prescribe J_e-th order, described later. After evaluting $\hat{\mathbf{u}}$, we move onto the second term in equation (2.55), which defines the pressure at time step $n+1$ as,

$$\frac{\hat{\hat{\mathbf{u}}} - \hat{\mathbf{u}}}{\Delta t} = -\nabla p^{n+1}. \tag{2.57}$$

$\hat{\hat{\mathbf{u}}}$ denotes as the secondary intermediate velocity. In this single equation, we seek to obtain two unknown solutions, $\hat{\hat{\mathbf{u}}}$ and p^{n+1}, which is ill-posed, and seek to impose certain restrictions. The splitting method assumes that the secondary intermediate velocity is divergence free, $\nabla\cdot\hat{\hat{\mathbf{u}}} = 0$, and satisfies the Dirichlet boundary conditions normal to the boundary, $\hat{\hat{\mathbf{u}}}\cdot\mathbf{n} = \mathbf{u}|_{\Omega}\cdot\mathbf{n}$. By considering the assumptions above and the divergence of equation (2.57), we obtain the pressure Poisson equation with the primary intermediate velocity acting as the forcing term,

$$\nabla^2 p^{n+1} = \nabla\cdot\left(\frac{\hat{\mathbf{u}}}{\Delta t}\right) \tag{2.58a}$$

and boundary conditions,

$$\frac{\partial p^{n+1}}{\partial n} = \mathbf{n}\cdot\left(\frac{\hat{\hat{\mathbf{u}}} - \hat{\mathbf{u}}}{\Delta t}\right). \tag{2.58b}$$

While the pressure boundary condition (2.58b) is straightforward to evalute, it is sensitive to large splitting errors [Karniadakis et al., 1991]. To overcome this, we consider a high-order boundary condition of pressure, obtained by taking the ~~normal dot product~~ dot product with respect to the normal of equation (2.53),

$$\frac{\partial p^{n+1}}{\partial t} = -\sum_{q=0}^{J_e-1} \beta_q \left[\frac{1}{\Delta t}\mathbf{u}^{n-q} + \nu[\nabla \times (\nabla \times \mathbf{u}^{n-q})] + (\mathbf{u}^{n-q} \cdot \nabla)\mathbf{u}^{n-q} \right] \cdot \mathbf{n}. \tag{2.59}$$

Notably, the linear operator is expressed as $\mathbf{L}(\mathbf{u}) = \nabla(\nabla \cdot \mathbf{u}) - \nabla \times (\nabla \times \mathbf{u})$, favouring numerical stability [Orszag et al., 1986, Karniadakis et al., 1991]. J_e is the order the explicit scheme as in equation (2.56). After solving for the pressure Poisson equation, the secondary intermediate velocity could be subsequently obtained using equation (2.57). After which, we can move onto the final substep in equation (2.55), by solving a Helmholtz equation for $\mathbf{u}^{n+1}$,

$$\frac{\gamma_0 \mathbf{u}^{n+1} - \hat{\hat{\mathbf{u}}}}{\Delta t} = \nu \sum_{q=0}^{J_i-1} \gamma_q \mathbf{L}(\mathbf{u}^{n+1-q}), \tag{2.60}$$

where the linear terms are treated based similar to the family of Adams-Moulton implicit scheme and J_i, γ_q denotes the order of the scheme and time integration coefficients, completing the velocity correction scheme. The time integration coefficients are determined from stiffly stable schemes shown in table 2.2, an improvement from the Adams-family schemes [Karniadakis et al., 1991]. The high

Coefficients	1^{st} order	2^{nd} order	3^{rd} order
γ_0	1	3/2	11/6
α_0	1	2	3
α_1	0	-1/2	-3/2
α_2	0	0	1/3
β_0	1	2	3
β_1	0	-1	-3
β_2	0	0	1

Table 2.2: Integration coefficient of stiffly stable schemes from Karniadakis et al. [1991].

order velocity correction scheme and be summarised in a three step process of the following,

$$\mathbf{u}^n \xrightarrow{\mathbf{N}(\mathbf{u}^n)} \hat{\mathbf{u}} \xrightarrow{\nabla^2 p} \hat{\hat{\mathbf{u}}} \xrightarrow{\mathbf{L}(\hat{\hat{\mathbf{u}}})} \mathbf{u}^{n+1},$$

evolving the velocity fields at time step n to $n+1$.

2.4.2 Fourier spectral/*hp* modes

Fourier-Chebyshev-Fourier type discretisation have been recognised as preferred method for performing direct numerical simulations (DNS) of transitional or turbulent incompressible channel flows [Kim et al., 1987] owing to its efficient representation of the inhomogeneous wall-normal directions and the homogeneous streamwise and spanwise directions, using Chebyshev and Fourier expansions respectively.

The Fourier spectral/*hp* element method draws on this approach, where the homogeneous and

the inhomogeneous directions are represented by the Fourier expansions and spectral/*hp* elements respectively. This approach has been commonly referred to as the Quasi-3D or (2.5D) approach, allowing for the representation of two inhomogeneous directions. For example, in the turbulent channel flows with riblets, the Fourier expansions are used to represent the periodic streamwise, while the spectral/*hp* elements are used to discretise the wall-normal direction. In the analysis of three-dimensional wakes of cylinders where the Fourier expansions are treated in the spanwise directions. In this thesis, we routinely use the the Quasi-3D approach, consisting of the 2D spectral/*hp* elements with 1D Fourier expansions are used to discretise the cross stream plane and streamwise flow respectively. The velocity and pressure in the spectral/*hp* plane is described by two dimensional modified bases with Fourier expansions,

$$\begin{bmatrix} \mathbf{u}^{\delta}(x,y,z,t) \\ p^{\delta}(x,y,z,t) \end{bmatrix} = \sum_{k=0}^{N_z-1} \sum_{p=0}^{P} \sum_{q=0}^{P} \psi_p(x)\psi_q(y) e^{ik\beta z} \begin{bmatrix} \hat{\mathbf{u}}_{p,q,k}(t) \\ \hat{p}_{p,q,k}(t) \end{bmatrix} = \sum_{k=0}^{N-1} e^{ik\beta z} \begin{bmatrix} \tilde{\mathbf{u}}_k(x,y,t) \\ \tilde{p}_k(x,y,t) \end{bmatrix} \tag{2.61}$$

where $\beta = \frac{2\pi}{L_z}$ is the spanwise wavenumber, L_z the spanwise length, N_z the number of Fourier expansions. Substituting equation 2.61 into the Navier-Stokes equations and taking the Fourier transform (equivalently to the Galerkin projection with respect to Fourier expansion as a test function) yields a system of N_z decoupled equations, amenable for parallel processing,

$$\frac{\partial \tilde{\mathbf{u}}_k}{\partial t} = -\tilde{\nabla}_k \tilde{p}_k + \nu(\nabla^2_{x,y} - k^2\beta^2)\tilde{\mathbf{u}}_k - [\widehat{(\mathbf{u}\cdot\nabla)\mathbf{u}}]_k \tag{2.62a}$$

$$-k\beta\tilde{\nabla}\cdot\tilde{\mathbf{u}}_k = 0, \quad k = 0, ..., N_z - 1 \tag{2.62b}$$

where, $\tilde{\nabla}_k = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, ik\beta)$, $\nabla^2_{x,y} = (\frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial y^2})$ and $[\widehat{(\mathbf{u}\cdot\nabla)\mathbf{u}}]_k$ refers to the Fourier-transformed of the k^{th} nonlinear term.

2.4.3 Maintaining fluid flow through a channel

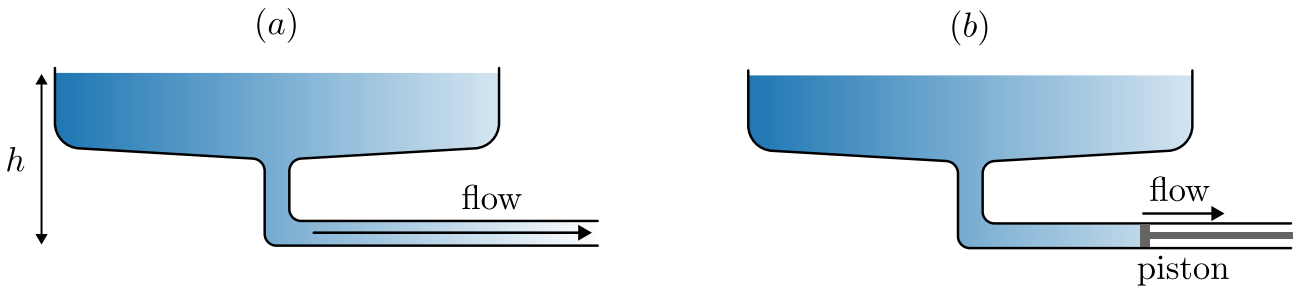

Figure 2.6: (a) Flow rate driven by a pressure gradient from an reservoir elevated by h. (b) Flow driven by a piston at a constant flow rate.

In general, there are two approaches to drive a fluid flow through a channel, either by maintaining a constant pressure drop, or a constant volumetric flux (flow rate). This difference is illustrated in figure 2.6, whereby the flow through the channel is driven by a constant pressure drop from an elevated

reservoir of constant height h in figure 2.6(a), while a piston moves at a constant speed rightwards, drawing fluid through the channel at a constant volumetric flux in figure 2.6(b).

Constant pressure via body-forcing

s a prescribe the homogeneous direction along the streamwise directions, a pressure drop cannot be prescribe directly. Instead, we substitute the constant pressure drop with a constant body force $\mathbf{f} = f_x \hat{\mathbf{e}}_x$ in the streamwise direction,

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\nabla p + \nu \nabla^2 u + f_x, \tag{2.63}$$

to maintain a

The central question now becomes what is the magnitude of body force required ~~a~~ laminar or turbulent flow. To begin this discussion, we assume that we can decompose our flow variables into a mean and a fluctuating component,

$$u(x, y, t) = U(y) + u'(x, y, t), \tag{2.64}$$

where $U(y) = \langle u \rangle$ refers to the averaged velocity and $\langle \cdot \rangle = \frac{1}{TL_xL_z} \int \cdot \, dzdxdt$ refers to the temporal and span-averaged operator. The fluctuating component is defined with an average of 0, i.e. $\langle u' \rangle = 0$. Next, we substitute this decomposition into equation (2.63), and perform the averaging operation,

$$\begin{aligned} \Bigg\langle & \frac{\partial (U + u')}{\partial t} + (U + u') \frac{\partial (U + u')}{\partial x} + (V + v') \frac{\partial (V + v')}{\partial y} \\ & = -\frac{\partial (P + p')}{\partial x} + \nu \nabla^2 (U + u') + F_x + f'_x \Bigg\rangle. \end{aligned} \tag{2.65}$$

For a statistically stationary turbulent (or laminar) channel flow with periodic streamwise boundary conditions, we can make the following assumptions:

1. stationary flow $\frac{\partial U}{\partial t} = 0$,
2. fully-developed in x, $\frac{\partial}{\partial x} \to 0$,
3. $\frac{\partial V}{\partial y} = 0$, as a consequence of continuity and the no-slip boundary condition.
4. $\langle u', v', w', p' \rangle = 0$, based on the definition of fluctuations,
5. $\frac{\partial p}{\partial x} = 0$ due to the enforced periodicity in x.

Applying the assumptions above, the mean momentum equations simplify into,

$$\langle F_x \rangle = \left\langle \frac{\partial (u'v')}{\partial y} \right\rangle - \nu \frac{\partial U^2}{\partial y^2}, \tag{2.66}$$

where the body force on the left-hand side balances the sum of Reynolds stresses and viscous diffusion on the right-hand side. Next, we integrate the expression from $y \in [-1, 1]$,

$$2F_x = [\langle u'v' \rangle]_{y=-1}^{y=1} + \nu \left[\left. \frac{\partial U}{\partial y} \right|_{y=1} - \left. \frac{\partial U}{\partial y} \right|_{y=-1} \right]. \tag{2.67}$$

The wall shear stress is defined by $\tau_w = \nu \frac{\partial U}{\partial y}|_{y=1}$ (ρ is assumed to be 1), and it is antisymmetric about the channel centreline, $\nu \frac{\partial U}{\partial y}|_{y=1} = -\nu \frac{\partial U}{\partial y}|_{y=-1}$. Due to the no-slip condition, the Reynolds shear stresses is zero, i.e. $[u'v']|_{y=-1,1} = 0$. Hence, the expression above simplifies to,

$$\tau_w = F_x. \tag{2.68}$$

In other words, the body force F_x is balanced by the wall shear stress (drag), τ_w, along the channel walls. In the case of laminar flow, τ_w can be determined analytically, and the body force required for sustaining a laminar flow for a velocity profile of $u(y) = 1 - y^2$, is $F_x = -2\nu$. However, to determine the wall shear stress (and hence the magnitude of body force) is not as straightforward task for transitional or turbulent channel flow as there isn't an analytical expression for τ_w and its dependence on Reynolds number. Instead, we can only rely on empirical relations of turbulent channel flow between the skin friction coefficient, $c_f = \tau_w / \frac{1}{2}\rho U_c^2$ and Reynolds number Re_c from Dean [1978].

$$c_f = 0.00302 Re_c^{-1/4}, \tag{2.69}$$

where Re_c is the Reynolds number based on the laminar centerline velocity. Similarly, the skin

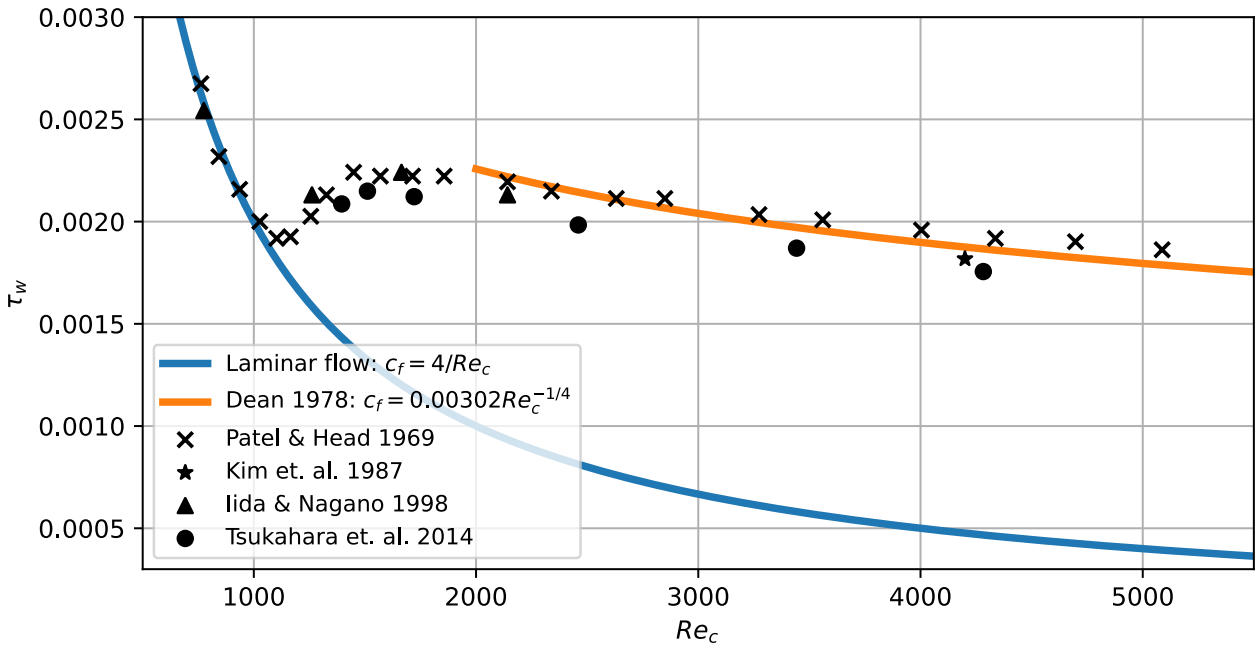

Figure 2.7: τ_w against Re_c using skin friction coefficients from Dean [1978] with $\rho = U_c = 1$. Experimental scatter points from [Patel and Head, 1969, Kim et al., 1987, Iida and Nagano, 1998, Tsukahara et al., 2014a].

friction coefficient for the case of laminar flow is $c_f = 4/Re_c$ [Dean, 1978]. Figure 2.7 illustrates the relationship between τ_w and Re_c of channel flow using empirical relationship from Dean [1978] (here $\rho = U_c = 1$) and experimental data from Patel and Head [1969], Kim et al. [1987], Iida and Nagano [1998], Tsukahara et al. [2014a]. While the empirial relation for laminar flow, $Re_c \lesssim 1000$ and turbulent flow $Re_c \gtrsim 2000$ appears reasonably robust, the wall shear stress in the transitional region is lacking therefore, the body forcing approach is not preferred.

Constant volumetric flux

An alternative approach is to enforce a constant volumetric flux, illustrated using the piston method in figure 2.6(b). We employ the efficient Green's function approach introduced by Chu and Karniadakis [1993], and outline its solution procedure. The volumetric flux is defined as,

$$Q(\mathbf{u}) = \frac{1}{2\mu(R)} \int_R \mathbf{u} \cdot \mathrm{d}\mathbf{s}, \tag{2.70}$$

where $Q(\cdot)$ refer to the flow rate operator through the surface R with surface area of $\mu(R)$. The idea is to append a correction velocity, $\mathbf{u}_{corr}$, to the velocity field at time step n, $\mathbf{u}^n$, such that the corrected solution, $\bar{\mathbf{u}}^n = \mathbf{u}^n + \mathbf{u}_{corr}$, has the desired volumetric flux $\bar{Q} = Q(\bar{\mathbf{u}}^n)$. While adding two solutions together is straightforward, the resultant velocity field may not directly satisfy the Navier-Stokes equations. Fortunately, we can leverage the velocity correction scheme which (in general) evaluates the nonlinear advection terms followed by a linear terms (pressure and dissipation). This process is summarised as,

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} = \mathbf{N}(\mathbf{u}) \\ \mathbf{u}(\mathbf{x}, 0) = \mathbf{u}^n \end{cases} \xrightarrow{\hat{\mathbf{u}}(\mathbf{x}, \Delta t)} \begin{cases} \frac{\partial \mathbf{u}}{\partial t} = -\nabla p + \nu \mathbf{L}(\mathbf{u}) \\ \mathbf{u}(\mathbf{x}, 0) = \hat{\mathbf{u}}(\mathbf{x}, \Delta t), \end{cases} \tag{2.71}$$

where $\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}^n$ and $\hat{\mathbf{u}}(\mathbf{x}, \Delta t)$ refer to the initial condition for the nonlinear advection terms, and the intermediate velocity, the initial condition for the linear terms, respectively. Since the second step correspond to solving the linear Stokes equation, any solution of the linear Stokes (such as $\mathbf{u}_{corr}$) added to the final solution will still satisfy the linear Stokes equations - a property of linear differential equations. We consider the linear Stokes equation governing the evolution of the correction velocity,

$$\frac{\partial \mathbf{u}_{corr}}{\partial t} = -\nabla p_{corr} + \nu \mathbf{L}(\mathbf{u}_{corr}) + \alpha^n \hat{\mathbf{e}}_{\mathbf{x}}, \tag{2.72}$$

where α^n is the undetermined magnitude of body force at time step n in the streamwise direction, $\hat{\mathbf{e}}_x$, required to maintain the desired flow rate $\bar{Q} = Q(\mathbf{u}^n) + Q(\mathbf{u}_{corr})$. Since $\mathbf{u}_{corr}$ is appended to $\mathbf{u}^n$, the initial condition for $\mathbf{u}_{corr}$ must be $\mathbf{u}_{corr}(\mathbf{x}, 0) = 0$, so that $\mathbf{u}^n$ remains compatible with the initial conditions in equation (2.71). Since α^n is undetermined, we normalise the equation with respect to α^n, yielding the linear Stokes equations with unit forcing,

$$\frac{\partial \mathbf{v}}{\partial t} = -\nabla \hat{p} + \nu \mathbf{L}(\mathbf{v}) + \hat{\mathbf{e}}_{\mathbf{x}}, \quad \mathbf{v}(\mathbf{x}, 0) = \mathbf{0}, \tag{2.73}$$

where $\mathbf{v} = \mathbf{u}_{corr}/\alpha^n$ and $\hat{p} = p_{corr}/\alpha^n$. The corrected velocity field becomes

$$\bar{\mathbf{u}} = \mathbf{u} + \alpha^n \mathbf{v}^1, \tag{2.74}$$

where $\mathbf{v}^1$ is solution field obtained by solving equation (2.73) in the first time step. To match the target volumetric flux, $\bar{Q}$, we need to scale α^n such that,

$$\bar{Q} = Q(\bar{\mathbf{u}}^n) = Q(\mathbf{u}^n) + Q(\alpha^n \mathbf{v}^1). \tag{2.75}$$

which gives,

$$\alpha^n = \frac{\bar{Q} - Q(\mathbf{u}^n)}{Q(\mathbf{v}^1)}, \tag{2.76}$$

evaluated at every time step n. The Green's function approach is computationally efficient as we only need to compute $\mathbf{v}^1$ and $Q(\mathbf{v}^1)$ once during the first time step and reuse it for subsequent time steps. The process of adding the correction velocity at the end of velocity correction scheme can be summarised in the procedure as follows,

$$\mathbf{u}^n \xrightarrow{\mathbf{N}(\mathbf{u}^n)} \hat{\mathbf{u}} \xrightarrow{\nabla^2 p} \hat{\hat{\mathbf{u}}} \xrightarrow{\mathbf{L}(\hat{\hat{\mathbf{u}}})} \mathbf{u}^{n+1} \xrightarrow{\alpha^{n+1}\mathbf{v}^1} \bar{\mathbf{u}}^{n+1}.$$

2.5 Stability analysis of the Navier-Stokes equations

2.5.1 Algorithms for linear stability analysis

In this section, we present a general overview of the numerical procedure for linear stability analysis. Linear stability analysis examines the stability of a base flow by considering the evolution of infinitesimal perturbations. These perturbations in general, may either grow or decay exponentially, indicating whether the base flow is linearly unstable or stable respectively. In §1.2, we introduced linear stability analysis in the context of wall-bounded shear flows leading to the Orr-Sommerfeld equations, where the base flows depend on a single inhomogeneous and two homogeneous directions, commonly referred to as local[2] stability analysis. For example, the laminar Poiseuille flow, $U(y) = 1 - y^2$ and the laminar Couette flow $U(y) = y$, $y \in [-1, 1]$. For some flows such as boundary layers, wakes and jets, their base flows are not strictly parallel. By considering a weak dependance on the stream and spanwise directions, their stability are described by the parabolised stability equations [Herbert, 1997]. When the base flow depends on two spatially inhomogeneous directions, $U(x, y)$, or three spatially inhomogeneous directions, $U(x, y, z)$, the analysis of such states are commonly referred to as biglobal or triglobal stability analysis, respectively [Theofilis, 2003]. If the base flow is time-dependent, such as in the secondary instability of cylinder flows, we use Floquet stability analysis [Henderson and Barkley, 1996].

In this section, we consider a time-independent base flow and consider a generic decomposition

[2]Referring to being spatially local in the context of 'real' flows which are typically inhomogeneous in all directions

of the velocity field in three spatial dimensions,

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{U}(\mathbf{x}) + \mathbf{u}'(\mathbf{x}, t), \tag{2.77}$$

where $\mathbf{U}(\mathbf{x}), \mathbf{u}'(\mathbf{x}, t)$ refers to the base flow and perturbations. Substituting this into the Navier-Stokes equations and linearising,

$$\frac{\partial \mathbf{u}'}{\partial t} = -(\mathbf{U} \cdot \nabla)\mathbf{u}' - (\mathbf{u}' \cdot \nabla)\mathbf{U} - \nabla p' + \frac{1}{Re}\nabla^2 \mathbf{u}', \tag{2.78a}$$

$$\nabla \cdot \mathbf{u}' = 0. \tag{2.78b}$$

This can be rewritten in as,

$$\frac{\partial}{\partial t}\mathbf{q}' = \mathcal{L}\mathbf{q}', \quad \mathcal{L} = \begin{bmatrix} -(\mathbf{U} \cdot \nabla) - (\nabla \mathbf{U}) + \frac{1}{Re}\nabla^2 & -\nabla \\ \nabla \cdot & \mathbf{0} \end{bmatrix}, \tag{2.79}$$

where $\mathcal{L}$ refer to the linearised operator and, $\mathbf{q}' = (\mathbf{u}', p')^T$. Assuming an initial perturbation, $\mathbf{q}'(\mathbf{x}, t = 0) = \mathbf{q}_0$, its evolution to time T is given by,

$$\mathbf{q}(\mathbf{x}', T) = \mathcal{A}(T, Re)\mathbf{q}_0, \quad \text{where} \quad \mathcal{A}(T, Re) = \exp(\mathcal{L}T). \tag{2.80}$$

We assume that the perturbations can be represented as a normal mode,

$$\mathbf{q}'(\mathbf{x}, t) = \tilde{\mathbf{q}}(\mathbf{x}) \exp(\lambda t) + \text{c.c} \tag{2.81}$$

where $\lambda_j, \tilde{\mathbf{q}}_j(x)$ refer to the j^{th} eigenvalue and eigenmode, and c.c refers to the complex conjugate. Substituting the normal mode into equation (2.80), we obtain an eigenvalue problem,

$$\mathcal{A}(T, Re)\tilde{\mathbf{q}}_j = \mu_j \tilde{\mathbf{q}}_j, \quad \mu_j = \exp(\lambda_j T). \tag{2.82}$$

where μ_j refers to the eigenvalue of $\mathcal{A} = \exp(\mathcal{L}T)$, and we typically set $T = 1$ [Barkley et al., 2008]. The real component of the eigenvalues determine the stability of the base flow, which can be either,

1. Unstable: $\Re(\lambda) > 0$,
2. Stable: $\Re(\lambda) < 0$,
3. Neutral: $\Re(\lambda) = 0$.

This concludes the mathematical overview of linear stabiltiy analysis, and the challenge lies in the computing the eigenpairs of $\mathcal{A}$ efficiently. For large matrices, $\mathcal{A} \in \mathbb{R}^{M \times M}$ (assuming it is real here for simplicity), direct eigenvalue solvers such as the QR algorithm costing $O(M^3)$ might be computationally infeasible. Another concern is that we are typically only interested in the most dangerous (leading) eigenvalues of largest real parts, and not the full spectrum. Lastly, we do not have access to $\mathcal{A}$ in a time stepping based code.

Power Iteration Method

A simple method in computing the dominant eigenpair is the power iteration method,

Definition 2.5.1 (Power iteration)**.** Given a diagonalisable matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and a non-zero vector $\mathbf{x}_0$, the sequence of matrix vector products between them (we neglect normalisation here),

$$\mathbf{A}\mathbf{x}_0, \mathbf{A}^2\mathbf{x}_0, \mathbf{A}^3\mathbf{x}_0, ..\mathbf{A}^k\mathbf{x}_0. \tag{2.83}$$

approaches the eigenvector of $\mathbf{A}$ with the largest magnitude. i.e. $\tilde{\mathbf{x}}_1 = \lim_{k\to\infty} \mathbf{A}^k\mathbf{x}_0$. The dominant eigenvalue, λ_1, can be computed using the Rayleigh quotient, $\lambda_1 = \frac{\tilde{\mathbf{x}}_1^T \mathbf{A}\tilde{\mathbf{x}}_1}{\tilde{\mathbf{x}}_1^T\tilde{\mathbf{x}}_1}$.

Arnoldi Method

We typically require two to four eigenpairs with the largest real parts. To compute more than one eigenpair, we ultilise the Arnoldi method [Arnoldi, 1951], belonging to a class of Krylov subspace iterative methods, for performing a Hessenberg reduction.

Definition 2.5.2 (Krylov Subspaces)**.** Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and a non-zero vector $\mathbf{x}_0 \in \mathbb{R}^n$, the k^{th}-Krylov subspace, $\mathcal{K}_n(\mathbf{A}, \mathbf{x}_0, k)$ is defined by,

$$\mathcal{K}_n(\mathbf{A}, \mathbf{x}_0, k) = \text{span}\{\mathbf{x}_0, \mathbf{A}\mathbf{x}_0, \mathbf{A}^2\mathbf{x}_0, \mathbf{A}^3\mathbf{x}_0, ..., \mathbf{A}^{k-1}\mathbf{x}_0\}. \tag{2.84}$$

Definition 2.5.3 (Hessenberg reduction)**.** The Hessenberg reduction is a matrix decomposition technique commonly used for the computing eigenpairs of matrices. Given a unsymmetric matrix $A \in \mathbb{R}^{N\times N}$ (we assume that $\mathbf{A}$ is real for simplicity), we seek a decomposition of the form,

$$\mathbf{A} = \mathbf{Q}\mathbf{H}\mathbf{Q}^T, \tag{2.85}$$

where,

- $\mathbf{H} \in \mathbb{R}^{N\times N}$ is an upper Hessenberg matrix (i.e. $a_{i,j} = 0$ for $i > j + 1$)
- $\mathbf{Q} \in \mathbb{R}^{N\times N}$ is an orthonormal matrix (i.e. $\mathbf{Q}^{-1} = \mathbf{Q}^T$), whose columns $\mathbf{q}_1, ..., \mathbf{q}_N$, form an orthonormal basis.

The Hessenberg reduction shows that $\mathbf{A}$ and $\mathbf{H}$ are similar matrices, which have the same eigenvalues. If $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, using $\mathbf{Q}^T = \mathbf{Q}^{-1}$ and multiplying (2.85) by $\mathbf{x}$,

$$\mathbf{A}\mathbf{x} = \mathbf{Q}\mathbf{H}\mathbf{Q}^{-1}\mathbf{x} \quad \Rightarrow \quad \lambda\mathbf{x} = \mathbf{Q}\mathbf{H}\mathbf{Q}^{-1}\mathbf{x} \quad \Rightarrow \quad \lambda\mathbf{Q}^{-1}\mathbf{x} = \mathbf{H}\mathbf{Q}^{-1}\mathbf{x} \quad \Rightarrow \quad \lambda\mathbf{y} = \mathbf{H}\mathbf{y}. \tag{2.86}$$

Hence, $\lambda(\mathbf{A}) = \lambda(\mathbf{H})$, and their eigenvectors are related by $\mathbf{x} = \mathbf{Q}\mathbf{y}$.

The Arnoldi method generates a sequences of vectors $[\mathbf{u}_0, \mathcal{A}\mathbf{u}_0, .., \mathcal{A}^{k-1}\mathbf{u}_0]$ that spans the k-dimensional Krylov subspace. These vectors, are known as Arnoldi vectors [Golub and Van Loan, 2013], and are used to construct an orthogonal matrix via the Gram-Schmidt process, $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_k] \in$

$\mathbb{R}^{M\times K}$. This is equivalent to performing a partial Hessenberg reduction of $\mathcal{A} = \mathbf{Q}\mathbf{H}\mathbf{Q}^T$, where the eigenvalues of $\mathcal{A} \in \mathbb{R}^{N\times N}$ can be approximated by a smaller Hessenberg matrix $\mathbf{H} \in \mathbb{R}^{k\times k}$, suitable for a direct eigenvalue computation using the QR algorithm, The $k-$step Arnoldi factorisation of $\mathcal{A}$ gives,

$$\mathcal{A}\mathbf{Q}_k = \mathbf{Q}_k\mathbf{H}_k + \mathbf{r}_k\mathbf{e}_k^T, \tag{2.87}$$

where $\mathbf{H} \in \mathbb{R}^{k\times k}$ refers to the upper Hessenberg matrix, $\mathbf{e}_k = [0, ..., 0, 1] \in \mathbb{R}^k$, and $\mathbf{r}_k \in \mathbb{R}^N$ is a residual vector. If $\mathbf{x} = \mathbf{Q}_k\mathbf{y}$, and $\mathbf{H}\mathbf{y} = \lambda\mathbf{y}$ then,

$$(\mathcal{A} - \mathbf{I}\lambda)\mathbf{x} = (\mathbf{e}_k^T\mathbf{y})\mathbf{r}_k. \tag{2.88}$$

In other words, the residual vector difference between the approximation of $\lambda(\mathcal{A})$, using $\lambda(\mathbf{H})$. If $||\mathbf{r}_k|| = 0$, then $\lambda(\mathbf{H}) \subseteq \lambda(\mathcal{A})$.

We now present the Arnoldi method by generating k Arnoldi vectors,

$$\mathbf{T}_k = [\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_{k-1}] = \left[\mathbf{u}_0, \frac{\mathcal{A}(T, Re)\mathbf{u}_0}{\alpha_1}, \frac{\mathcal{A}(T, Re)\mathbf{u}_1}{\alpha_2}, ..., \frac{\mathcal{A}(T, Re)\mathbf{u}_{k-1}}{\alpha_k}\right], \tag{2.89}$$

where α_j is scaled such that $||\mathbf{u}_j|| = 1$. Following [Barkley et al., 2008], the projection of $\mathcal{A}$ onto the Krylov subspace is given as,

$$\mathcal{A}\mathbf{T}_k = \mathbf{T}_{k+1}D_k^{(k+1)}, \tag{2.90}$$

where $D_k^{(k+1)} \in \mathbb{R}^{(k+1)\times k}$ is a shifted diagonal matrix with entries $D_{ij} = \alpha_i\delta_{i,j+1}$. We assume that $\mathbf{T}_k$ and $\mathbf{T}_{k+1}$ admit QR decompositions,

$$\mathcal{A}\mathbf{Q}_k\mathbf{R}_k = \mathbf{Q}_{k+1}\mathbf{R}_{k+1}\mathbf{D}_k^{(k+1)}, \tag{2.91}$$

where $\mathbf{Q}_k \in \mathbb{R}^{N\times k}$, $\mathbf{R}_k \in \mathbb{R}^{k\times k}$ and $\mathbf{Q}_{k+1}, \mathbf{R}_{k+1}$ are similarly defined. The upper Hessenberg matrix $\mathbf{H}_k^{(k+1)} \in \mathbb{R}^{(k+1)\times k}$ is defined as,

$$\mathbf{H}_k^{(k+1)} = \mathbf{R}_{k+1}\mathbf{D}_k^{(k+1)}\mathbf{R}_k^{-1}, \tag{2.92}$$

in which the last row of $\mathbf{H}_k^{(k+1)}$ only contains a single non-zero entry, $h^* = h_{k,k-1}$. By substituting the definition of the upper Hessenberg matrix and separating the last row of $\mathbf{H}_k^{(k+1)}$ we obtain,

$$\mathcal{A}\mathbf{Q}_k = \mathbf{Q}_k\mathbf{H}_k + h^*\mathbf{q}_k\mathbf{e}_k^T. \tag{2.93}$$

Equation (2.93) describes the projection of $\mathcal{A}$ onto the Krylov subspace spanned by orthornomal bases $\mathbf{Q}_k$, yielding a smaller $\mathbf{H}_k$ matrix. The accuracy of this approximation is dictated by the magnitude of the residual term, $h^*\mathbf{q}_k\mathbf{e}_k^T$. Assuming that $\mathbf{H}_k$ is diagonalisable as $\mathbf{H}_k = \mathbf{\Psi}_k\mathbf{\Lambda}_k\mathbf{\Psi}_k^{-1}$, we multiply equation (2.93) by $\mathbf{\Psi}_k$,

$$\mathcal{A}\mathbf{Q}_k\mathbf{\Psi}_k = \mathbf{Q}_k\mathbf{\Psi}_k\mathbf{\Psi}_k^{-1}\mathbf{H}_k\mathbf{\Psi}_k + h^*\mathbf{q}_k\mathbf{e}_k^T\mathbf{\Psi}_k. \tag{2.94}$$

Simplifying the expression above we get,

$$\mathcal{A}\mathbf{V}_k = \mathbf{V}_k\mathbf{\Lambda}_k + h^*\mathbf{q}_k\mathbf{e}_k^T\mathbf{\Psi}_k, \tag{2.95}$$

where $\mathbf{\Lambda}_k$ contains the k eigenvalues and $\mathbf{V}_k = \mathbf{Q}_k\mathbf{\Psi}_k$ the eigenvectors of $\mathcal{A}$. The error in approximating the j^{th} eigenpair is given by,

$$\varepsilon_j = ||\mathcal{A}\mathbf{v}_j - \lambda_j\mathbf{v}_j|| = ||h^*\mathbf{q}_k\mathbf{e}_k^T\psi_j|| = |h^*||\mathbf{v}_j[k-1]|, \tag{2.96}$$

where $\mathbf{v}_j[k-1]$ is the last component of the eigenvector $\mathbf{v}_j$.

Lastly, we are generally interested in obtaining the eigenpairs with the the largest real part. We introduce exponential power method [Tuckerman and Barkley, 2000], which is naturally considered by time stepping an initial perturbation $\mathbf{q}'_0$ from $t = 0$ to T,

$$\mathbf{q}'(T) = \exp(\mathcal{L}T)\mathbf{q}'_0 = \mathcal{A}(T)\mathbf{q}'_0. \tag{2.97}$$

The dominant eigenvalues, μ, of $\mathcal{A}$ obtained from the Arnoldi method described above, which correspond to the eigenvalue of the largest real part λ, of $\mathcal{L}$ by $\mu = \exp(\lambda T)$, where T is typically set to 1. For further details on this algorithm, the reader is referred to Barkley et al. [2008] for more details.

In summary, the algorithm described above have been implemented in Nektar++, referred to as the 'modified' Arnoldi algorithm, which modifies the existing time-stepper code with a wrapper function that generates Arnoldi vectors and solves the Hessenberg matrix using the subroutine `dgeev` from LAPACK (Linear Algebra PACKage, [Anderson et al., 1999]). The modified Arnoldi algorithm has been verified against using a separate implementation based on the third-party package ARPACK (ARnoldi PACKage [Lehoucq et al., 1998]) [Rocco, 2014].

2.5.2 Edge tracking

In the section, we consider the dynamical system interpretation of transition, where the laminar state is separated by the turbulent state by an edge, referred to the edge of chaos. Along this edge, there could be attractors, sometimes in the form of travelling-waves, tori, and high-order invariant sets, known as the edge states. For the edge tracking, we use the bisection method [Skufca et al., 2006, Schneider et al., 2007, Khapko et al., 2016], with an initial condition given by

$$\mathbf{x}_0 = \chi\mathbf{x}_L + (1-\chi)\mathbf{x}_T \tag{2.98}$$

where $\mathbf{x}_0$ refers to an initial condition consisting of a weighted sum, $\chi \in [0, 1]$, between a laminar state, $\mathbf{x}_l$, and a turbulent state, $\mathbf{x}_T$. Since the laminar and turbulent state forms a bistable system, there could be (at least) one critical value of $\chi \in [0, 1]$, where the trajectory walks along the 'edge' between the turbulent and laminar state without decaying to either states. To find this χ_c, we perform n successive bisections between χ_L^n, χ_T^n, the upper and lower bounds such that the trajectory relaminarises or become turbulent respectively, where χ^n is updated by $\chi^n = \frac{1}{2}(\chi_L^n + \chi_T^n)$ At every n^{th} bisection,

it involves a stopping criteria, a tolerance based on the deviation of an observable (e.g. wall shear stresses) away from the initial condition. Then, a direct numerical simulation is reinitialised with an initial condition given by equation (2.98) For every successive bisection, the difference between two trajectories, $\Delta\chi^n = \chi_L^n - \chi_T^n$, decays like $\Delta\chi^n \sim 0.5^n$, and is related to the Lyapunov exponent of the edge

$$\Delta\chi \approx C\exp(\mu_e t) \tag{2.99}$$

where μ_e, C refers to Lyapunov exponent of the edge and a constant. In practice, we consider $n = 10, 20$ and for $n = 10$, the solution along the edge is converged. After we determine the critical χ_c, we repeat the bisections step by replacing the laminar state, $\mathbf{x}_L$, and the turbulent state $\mathbf{x}_T$, ~~which~~ the solution trajectory with χ_L and χ_T, that has been terminated after exceeded the threshold. We refer this repetition as the number of 'outer' bisections, while the bisection for χ^n is referred to 'inner bisections' After a certain number of 'outer' bisections, the trajectory may converge towards an attractor, which may exist in a from of travelling-waves, periodic orbits or a chaotic attractor. This attractor sits along the edge is referred to as the edge state, a saddle acting as a separatrix between the turbulent and laminar attractor. We describe the algorithm of edge tracking in algorithm 2.5.2

Algorithm 1 Algorithm for edge tracking between a turbulent and laminar state

Initialise `maxInBisects, maxOutBisects` ▷ Maximum inner and outer bisections
Initialise `tol` ▷ Tolerance for stopping criteria (e.g., wall-shear stress)
`outBisects` $\leftarrow 0$
while `outBisects` < `maxOutBisects` **do**
  if `outBisects` == 0 **then**
    $\mathbf{x}_L, \mathbf{x}_T \leftarrow$ `input()` ▷ Initial laminar and turbulent states
  end if
  $\chi_L \leftarrow 0,\ \chi_T \leftarrow 1,\ \chi \leftarrow \frac{1}{2}(\chi_L + \chi_T)$ ▷ Initialise bisection coefficients
  $\mathbf{x}_0 \leftarrow \chi \mathbf{x}_T + (1-\chi)\mathbf{x}_L$ ▷ Initialise initial condition
  `inBisects` $\leftarrow 0$
  while `inBisects` < `maxInBisects` **do**
    $\mathbf{k} \leftarrow 0, \Delta \leftarrow 10^6$
    while $\Delta >$ `tol` **do** Why is this state guaranteed to happen at some time?
      $\mathbf{x}_{k+1} \leftarrow$ `TimeIntegrate`$(\mathbf{x}_k)$
      $\Delta \leftarrow |\mathbf{x}_{k+1} - \mathbf{x}_0|$ ▷ Deviation from initial condition
      $\mathbf{k} \leftarrow \mathbf{k} + 1$
    end while
    if `isTurbulent`$(\mathbf{x}_k)$ **then** ▷ Check if terminal state is turbulent
      $\chi_L \leftarrow \chi$ ▷ $\mathbf{x}_L$ gets larger weight
      if `inBisects` == `maxInBisects` - 1 **then**
        $\mathbf{x}_T \leftarrow \mathbf{x}_k$ ▷ Save turbulent-leaning initial condition
        `break`
      end if
    else
      $\chi_T \leftarrow \chi$
      if `inBisects` == `maxInBisects` - 1 **then**
        $\mathbf{x}_L \leftarrow \mathbf{x}_k$ ▷ Save laminar-leaning initial condition
        `break`
      end if
    end if
    $\chi \leftarrow \frac{1}{2}(\chi_L + \chi_T)$
    $\mathbf{x}_0 \leftarrow \chi \mathbf{x}_L + (1-\chi)\mathbf{x}_T$ ▷ Update initial conditions
    `inBisects++`
  end while
  `outBisects++`
end while