Computer Methods in Applied Mechanics and Engineering 89 (1991) 141–219
North-Holland

A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier–Stokes equations

Farzin Shakib
CENTRIC Engineering Systems, Inc., 3801 East Bayshore Road, Palo Alto, CA 94303, USA

Thomas J.R. Hughes and Zdeněk Johan
Division of Applied Mechanics, Durand Building, Stanford University, Stanford, CA 94305, USA

Received 7 November 1990

A space–time element method is presented for solving the compressible Euler and Navier–Stokes equations. The proposed formulation includes the variational equation, predictor multi-corrector algorithms and boundary conditions. The variational equation is based on the time-discontinuous Galerkin method, in which the physical entropy variables are employed. A least-squares operator and a discontinuity-capturing operator are added, resulting in a high-order accurate and unconditionally stable method. Implicit/explicit predictor multi-corrector algorithms, applicable to steady as well as unsteady problems, are presented; techniques are developed to enhance their efficiency. Implementation of boundary conditions is addressed; in particular, a technique is introduced to satisfy nonlinear essential boundary conditions, and a consistent method is presented to calculate boundary fluxes. Numerical results are presented to demonstrate the performance of the method.

1. Introduction

This paper presents a space–time finite element method to solve the compressible Euler and Navier–Stokes equations. Mathematical analyses, feasibility and implementation studies, and numerical experiments have been used to design this method. We have placed a great deal of emphasis on formulating this method in such a manner that its stability, convergence and accuracy could be rigorously established. This is crucial, since a method which cannot be proved stable, convergent, accurate, etc., is very likely not to have these properties. When such a method is faced with the special situations which arise frequently in industrial applications, it may fail. Since our basic method has firm foundations, peripheral techniques, such as domain decomposition and efficient linear solution algorithms, were able to be incorporated simply and effectively. Finally, we have tested this method on a variety of problems, ranging from steady hypersonic flows to transient nearly-incompressible flows.

1.1. Overview

We begin, in Section 2, by stating the compressible Euler and Navier–Stokes equations, and reviewing their symmetrization via the (physical) entropy variables as described in [1]. The Galerkin formulation of this symmetric form is dimensionally consistent. This is not only essential for theoretical analyses, such as stability and convergence proofs, but also desirable for implementation, since it eliminates the need for *subjective* nondimensionalizations and contributes to the efficiency of iterative solution algorithms. More significantly, the discrete solution of the Galerkin formulation based on these variables automatically satisfies the Clausius–Duhem inequality, or the second law of thermodynamics.

In Section 3, we define our variational formulation based on the symmetric form of the compressible Navier–Stokes equations. We employ the time-discontinuous Galerkin method as the starting point of our formulation. This results in a complete space–time finite element discretization, which eliminates the need for an additional ordinary differential equation solver. Moreover, the mathematical properties of this method, applied to unsteady problems, are completely analogous to those of the Galerkin method, applied to steady problems. This was first observed by Johnson et al. [2]. This is important because the construction of a good method based on the Galerkin method for steady problems readily extends to unsteady problems.

When applied to advection-dominated problems, the Galerkin method, and likewise the discontinuous Galerkin method, lack stability. Unresolved internal and boundary layers give rise to oscillations which pollute the entire solution. To remedy this difficulty, we add a least-squares operator and a discontinuity-capturing operator to our basic Galerkin method. These operators provide the needed stability without sacrificing accuracy. The end result is a high-order accurate and unconditionally stable method. For an overview see [3], and for a similar method applied to second-order hyperbolic equations see [4].

The predecessors to our Galerkin/least-squares method and discontinuity-capturing operator are the SUPG method and the discontinuity-capturing operator proposed by Hughes and Mallet [5, 6]; see also [7]. Earlier references to the SUPG method are [8, 9], and a review of the method is presented in [10]. For the SUPG method, and to a lesser extent for this method with the discontinuity-capturing operator adjoined, a complete set of mathematical analyses has been developed. Hughes et al. [11] have derived error estimates for linear symmetric advective-diffusive systems, and Johnson et al. [12] have established a convergence proof for nonlinear hyperbolic conservation laws, such as the Euler equations. See [13] for recent results. In the latter part of Section 3, we extend these results, or provide the basis for their extension, to our variational formulation, as applied to the compressible Navier–Stokes equations. To our knowledge, no such mathematical analyses exist for other existing finite element or difference methods in computational fluid dynamics.

Finite element discretization of the space–time variational equation leads to a system of nonlinear algebraic equations at each time step. In Section 4, we develop classes of implicit/explicit–predictor multi-corrector algorithms to reduce the nonlinear system to sequences of linear systems. In particular, we present two finite element discretizations. The first discretization consists of a constant-in-time approximation, which leads to an inexpensive and highly stable first-order time-accurate algorithm, ideal for steady problems. The performance of this algorithm is further improved by employing a local-time-stepping strategy.

The second discretization consists of a linear-in-time approximation. This leads to a stable, third-order time-accurate algorithm, involving a linear system with twice as many equations and unknowns as the constant-in-time approximation. Employing a preconditioning technique, the size of this system is reduced by half. The end result is a stable third-order time-accurate algorithm, appropriate for capturing transient behavior. A comprehensive stability and accuracy analysis of these algorithms as applied to a time-dependent scalar advective-diffusive model problem is presented in [14].

In Section 5, we address the implementational aspects of boundary conditions. In particular, we present a consistent technique for satisfying nonlinear essential boundary conditions within the context of the predictor multi-corrector algorithms. This has practical importance since, in our formulation, the equations relating the working variables (i.e., entropy variables) to the variables used in specifying boundary conditions (i.e., primitive variables) are highly nonlinear. This is also true for methods based on the conservative variables, and the technique can be extended to these cases. We go on to study flux-type boundary conditions that arise naturally from our variational formulation, and conclude the section by presenting a consistent method for calculating wall fluxes.

In addition to numerical tests in Sections 3–5 that are designed to examine particular aspects of our method, in Section 6 we present numerical results to demonstrate overall effectiveness. These results are for two-dimensional problems ranging from steady high-speed to transient low-speed, nearly-incompressible flows. A multi-element group preconditioned GMRES algorithm for solving the nonsymmetric linear systems arising from the discretization is presented in [15] along with several additional numerical studies. In Section 7, we draw conclusions.

In Appendix A, we present the flux vectors and coefficient matrices of the compressible Navier–Stokes equations written in terms of entropy variables. In Appendix B, we provide the transformation matrices employed to satisfy essential boundary conditions.

2. Problem statement

In this section we review the compressible Euler and Navier–Stokes equations. Hereafter, we consider the compressible Euler equations simply as a special case of the compressible Navier–Stokes equations. Classically, these equations have been written in terms of so-called *conservative variables*; see [16]. The Galerkin formulation of the compressible Navier–Stokes equations based on these variables lacks certain properties which are needed to establish stability proofs and convergence analyses. To obtain the needed properties, Hughes et al. [1] employed physical entropy variables. They symmetrized the compressible Navier–Stokes equations by writing them in terms of these variables. The Galerkin formulation of this symmetric form not only is dimensionally consistent, but also makes physical sense in that the Clausius–Duhem inequality, or the second law of thermodynamics (which is the relevant physical stability condition for the compressible Navier–Stokes equations), is automatically satisfied by the discrete solution. In this section we describe the physical entropy variables along with the symmetric form of the compressible Euler and Navier–Stokes equations. These equations are the starting point of the work herein.

2.1. The compressible Euler and Navier–Stokes equations

In conservation form, the compressible Navier–Stokes equations are written as

$$\boldsymbol{U}_{,t} + \boldsymbol{F}_{i,i} = \boldsymbol{F}^{\mathrm{d}}_{i,i} + \mathscr{F} \,, \tag{2.1}$$

where, in three dimensions,

$$\boldsymbol{U} = \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{Bmatrix} = \rho \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e \end{Bmatrix} \quad \text{(conservative variables)} \,, \tag{2.2}$$

$$\boldsymbol{F}_i = \rho u_i \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e \end{Bmatrix} + p \begin{Bmatrix} 0 \\ \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ u_i \end{Bmatrix} \quad \text{(Euler flux)} \,, \tag{2.3}$$

$$\boldsymbol{F}^{\mathrm{d}}_i = \begin{Bmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ \tau_{3i} \\ \tau_{ij} u_j \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -q_i \end{Bmatrix} \quad \text{(diffusive flux)} \,, \tag{2.4}$$

(i^{th} column of visc. stress tensor.)

$$\mathscr{F} = \rho \begin{Bmatrix} 0 \\ b_1 \\ b_2 \\ b_3 \\ b_i u_i + r \end{Bmatrix} \quad \text{(source vector)} \,. \tag{2.5}$$

In the above relations: ρ is the density; $\boldsymbol{u} = \{u_1, u_2, u_3\}^{\mathrm{t}}$ is the velocity vector; e is the total energy density, which is the sum of the internal energy density ι and the kinetic energy density $|\boldsymbol{u}|^2/2$; p is the thermodynamic pressure; δ_{ij} is the Kronecker delta (i.e., $\delta_{ij} = 1$ for $i = j$, and $\delta_{ij} = 0$ for $i \neq j$); $\boldsymbol{\tau} = [\tau_{ij}]$ is the viscous-stress tensor; $\boldsymbol{q} = \{q_1, q_2, q_3\}^{\mathrm{t}}$ is the heat-flux vector; $\boldsymbol{b} = \{b_1, b_2, b_3\}^{\mathrm{t}}$ is the body force vector per unit mass; r is the heat supply per unit mass; and the summation convention is used throughout.

The five equations in (2.1) represent the conservation of mass, momentum and energy. To (2.1) we append the following *constitutive* relations:

$$\iota = c_v \theta \,, \tag{2.6}$$

$$p = (\gamma - 1)\rho\iota \,, \tag{2.7}$$

$$\tau_{ij} = \lambda u_{k,k} \delta_{ij} + \mu(u_{i,j} + u_{j,i}) \,, \tag{2.8}$$

$$q_i = -\kappa \theta_{,i} \,, \tag{2.9}$$

where c_v is the specific heat at constant volume; θ is the absolute temperature; γ is the ratio of specific heats (i.e., $\gamma = c_p/c_v$, where c_p is the specific heat at constant pressure); λ and μ are the viscosity coefficients; and κ is the coefficient of thermal conductivity. Equations (2.6) and (2.7) constitute the *perfect gas law*; (2.8) defines the viscous stress components; and (2.9) is *Fourier's law* of heat conduction.

It is useful to rewrite (2.1) in quasi-linear form

$$\boldsymbol{U}_{,t} + \boldsymbol{A}_i \boldsymbol{U}_{,i} = (\boldsymbol{K}_{ij}\boldsymbol{U}_{,j})_{,i} + \mathscr{F} , \tag{2.10}$$

where $\boldsymbol{A}_i = \boldsymbol{F}_{i,\boldsymbol{U}}$ is the ith Euler–Jacobian matrix; and $\boldsymbol{K} = [\boldsymbol{K}_{ij}]$ is the diffusivity matrix, with the $\boldsymbol{K}_{ij}$'s satisfying $\boldsymbol{K}_{ij}\boldsymbol{U}_{,j} = \boldsymbol{F}_i^{\mathrm{d}}$.

We view the compressible Euler equations as the compressible Navier–Stokes equations with zero diffusion and no source terms. Hence, the compressible Euler equations are simply

$$\boldsymbol{U}_{,t} + \boldsymbol{A}_i \boldsymbol{U}_{,i} = \boldsymbol{0} . \tag{2.11}$$

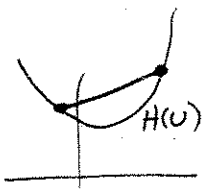
Note that the $\boldsymbol{A}_i$'s are not symmetric. However, all linear combinations of the $\boldsymbol{A}_i$'s have real eigenvalues and a complete set of eigenvectors; hence, (2.11) constitutes a hyperbolic system of conservation laws; see [16].

2.2. Symmetrization using entropy variables

Entropy variables have been investigated by Godunov [17], Mock [18], Harten [19], Tadmor [20], Hughes et al. [1], Dutt [21], and Johnson et al. [12]. Here we review and adopt the physical entropy variables:

We define a scalar *generalized entropy function* as

$$H = H(\boldsymbol{U}) = -\rho(s - s_0) , \tag{2.12}$$



where

$$s = \ln(p/\rho^{\gamma}) \tag{2.13}$$

is the nondimensional entropy, and s_0 is its reference value. H is a strictly convex function of $\boldsymbol{U}$. Consequently, a change of variables $\boldsymbol{U} \mapsto \boldsymbol{V}$ can be defined by letting

$$\boldsymbol{V}^{\mathrm{t}} = \partial H/\partial \boldsymbol{U} , \tag{2.14}$$

$\boldsymbol{V}$ is referred to as the vector of (*physical*) *entropy variables*. Equation (2.14) yields the following mapping:

$$\boldsymbol{V} = \frac{1}{\rho\iota} \begin{Bmatrix} -U_5 + \rho\iota(\gamma + 1 - s + s_0) \\ U_2 \\ U_3 \\ U_4 \\ -U_1 \end{Bmatrix} , \tag{2.15}$$

where

$$s = \ln((\gamma - 1)\rho\iota / U_1^{\gamma}) , \tag{2.16}$$

$$\rho\iota = U_5 - (U_2^2 + U_3^2 + U_4^2)/2U_1 . \tag{2.17}$$

The inverse mapping, $V \mapsto U$, is given by

$$\boldsymbol{U} = \rho\iota \begin{Bmatrix} -V_5 \\ V_2 \\ V_3 \\ V_4 \\ 1 - (V_2^2 + V_3^2 + V_4^2)/2V_5 \end{Bmatrix}, \tag{2.18}$$

where

$$\rho\iota = ((\gamma - 1)/(-V_5)^{\gamma})^{1/(\gamma-1)} \exp((-s + s_0)/(\gamma - 1)) , \tag{2.19}$$

$$s = \gamma - V_1 + (V_2^2 + V_3^2 + V_4^2)/2V_5 . \tag{2.20}$$

Using the change of variables $\boldsymbol{U} = \boldsymbol{U}(\boldsymbol{V})$, (2.10) is written as

$$\tilde{\boldsymbol{A}}_0 \boldsymbol{V}_{,t} + \tilde{\boldsymbol{A}}_i \boldsymbol{V}_{,i} = (\tilde{\boldsymbol{K}}_{ij} \boldsymbol{V}_{,j})_{,i} + \mathscr{F} , \tag{2.21}$$

where

$$\tilde{\boldsymbol{A}}_0 = \boldsymbol{U}_{,V} , \tag{2.22}$$

$$\tilde{\boldsymbol{A}}_i = \boldsymbol{A}_i \tilde{\boldsymbol{A}}_0 , \tag{2.23}$$

$$\tilde{\boldsymbol{K}}_{ij} = \boldsymbol{K}_{ij} \tilde{\boldsymbol{A}}_0 . \tag{2.24}$$

Due to the structure of H:

(i) $\tilde{\boldsymbol{A}}_0$ is symmetric positive-definite,

(ii) $\tilde{\boldsymbol{A}}_i$ is symmetric, ✳

(iii) $\tilde{\boldsymbol{K}} = [\tilde{\boldsymbol{K}}_{ij}]$ is symmetric positive-semidefinite. ✳

The explicit definition of the flux vectors and coefficient matrices is presented in terms of $\boldsymbol{V}$-variables in Appendix A. In addition to these arrays, it is useful to express the source vector as the product of a coefficient matrix and the vector $\boldsymbol{V}$. That is, we wish to define $\tilde{\boldsymbol{C}}$ such that

$$\tilde{\boldsymbol{C}}\boldsymbol{V} = -\mathscr{F} . \tag{2.25}$$

Note that $\tilde{\boldsymbol{C}}$ is not uniquely defined. In Appendix A, we have included one possible definition which is symmetric and well-defined. Similar to (2.22)–(2.24), we define a matrix $\boldsymbol{C}$ such that

$$\tilde{\boldsymbol{C}} = \boldsymbol{C}\tilde{\boldsymbol{A}}_0 . \tag{2.26}$$

Note that in general

$$CU \neq -\mathscr{F} \,. \tag{2.27}$$

The symmetry of (2.21) not only is important for mathematical analyses, but it can also be exploited in numerical algorithms, such as iterative solution algorithms (see [15]). In addition, this symmetric system of the compressible Navier–Stokes equations leads to a global statement of stability. A stability result is obtained by taking the dot product of (2.21) with V:

$$\begin{aligned} 0 &= V \cdot (\tilde{A}_0 V_{,t} + \tilde{A}_i V_{,i} - (\tilde{K}_{ij} V_{,j})_{,i} - \mathscr{F}) \\ &= \frac{1}{c_v} \left(-(\rho\eta)_{,t} - (\rho\eta u_i)_{,i} + c_v V_{,i} \cdot (\tilde{K}_{ij} V_{,j}) - (q_i/\theta)_{,i} + \rho r/\theta \right), \end{aligned} \tag{2.28}$$

where $\eta = c_v s$ denotes the thermodynamic entropy. Rearranging yields

$$(\rho\eta)_{,t} + (\rho\eta u_i)_{,i} + (q_i/\theta)_{,i} - \rho r/\theta = c_v V_{,i} \cdot (\tilde{K}_{ij} V_{,j}) \geq 0 \,. \tag{2.29}$$

This is the Clausius–Duhem inequality, which is the basic nonlinear stability condition for the compressible Navier–Stokes equations. This leads to an important result: *The Galerkin finite element solution automatically inherits the entropy production property of the compressible Navier–Stokes equations.* (See also Section 3.6.1.)

Observe from (2.28) and (2.29) that the Galerkin formulation of the compressible Euler equations results in the *conservation of entropy*. This means that the Galerkin finite element method is inadequate for solving compressible Euler problems where entropy is actually produced, such as problems with shocks. Some additional mechanism is needed to attain the proper entropy production.

REMARK 2.1. Neither the flux vectors nor the coefficient matrices are dependent on the value of s_0. Consequently, s_0 plays no role in the numerical solution, and for implementational convenience it is set to zero. In mathematical analyses, however, s_0 plays an important role: in assuming s_0 is greater than s throughout the domain, H becomes strictly positive and can be considered as an *entropy-norm* measure (see Section 3.6.1).

3. Space–time Galerkin/least-squares variational formulation

In this section we define our finite element variational equation for the solution of the compressible Euler and Navier–Stokes equations. We employ the *time-discontinuous Galerkin method* as the basis of our formulation. This method generates a complete space–time finite element discretization which eliminates the need for any additional ordinary differential equation solver to resolve the temporal behavior of the problem. By using discontinuous discretization in time, we are able to march sequentially through time and solve for only a fraction of the total solution at one time. The mathematical properties of the time-discontinuous Galerkin method for unsteady problems are completely analogous to those of the

Galerkin method for the steady case, as observed by Johnson et al. [2]. This is important because the construction of a good method based on the Galerkin method for steady problems readily extends to unsteady problems.

The Galerkin method, and similarly the discontinuous Galerkin method, lack stability. This is manifested by spurious oscillations generated by unresolved internal and boundary layers. To improve upon the stability of these methods, while maintaining their order of accuracy, we add a *least-squares operator* to our basic Galerkin formulation. This operator is similar to the SUPG operator presented by Hughes and Mallet [5]; however, it has a conceptually simpler construction and appears more amenable to mathematical analysis.

Utilizing the SUPG operator, Johnson et al. [2] obtained error estimates for linear first-order symmetric hyperbolic systems, Hughes et al. [11] established error estimates for linear symmetric advective-diffusive systems, and Johnson et al. [12] derived a convergence proof for nonlinear hyperbolic conservation laws (see [13] for recent results). Extension of these works to the Navier–Stokes equations is non-trivial. We include in this section some of the main ingredients needed for such an extension.

The Galerkin/least-squares method is a linear method. It is well known that higher-order linear methods cannot produce non-oscillatory approximations to discontinuities. In practice, we observe overshoots and undershoots in the immediate vicinity of discontinuities and unresolved sharp layers. It is possible, in principle, to construct nonlinear operators to control these oscillations and at the same time maintain higher-order accuracy in smooth regions. Hughes et al. [22] and Hughes and Mallet [6] have proposed nonlinear *discontinuity-capturing operators* which fit nicely within the framework of the Galerkin/least-squares method. Johnson et al. [23] have employed these ideas, and Galeão and Dutra do Carmo [24] have proposed an alternate form. We further improve and numerically study two such operators in this section.

3.1. Finite element space

Consider the partition $0 = t_0 < t_1 < \cdots < t_N = T$ of the time interval $I =]0, T[$. Denote by $I_n =]t_n, t_{n+1}[$ the nth time interval. Clearly $I = \bigcup_{n=0}^{N-1} I_n \cup \{t_1, t_2, \ldots, t_{N-1}\}$. A *space–time* '*slab*' is then defined as

$$Q_n = \Omega \times I_n , \tag{3.1}$$

with boundary

$$P_n = \Gamma \times I_n , \tag{3.2}$$

where Ω denotes the d-dimensional spatial domain with boundary Γ; see Fig. 3.1.1.

For the nth space–time slab, let the spatial domain be subdivided into $(n_{el})_n$ elements, Ω_n^e, $e = 1, \ldots, (n_{el})_n$. Then, for the nth slab, we define space–time element domains as

$$Q_n^e = \Omega_n^e \times I_n , \quad e = 1, \ldots, (n_{el})_n . \tag{3.3}$$

Within each space–time element the trial solution and weighting functions are approximated by kth-order interpolation polynomials, $\mathscr{P}_k$. These functions are assumed C^0 continuous within

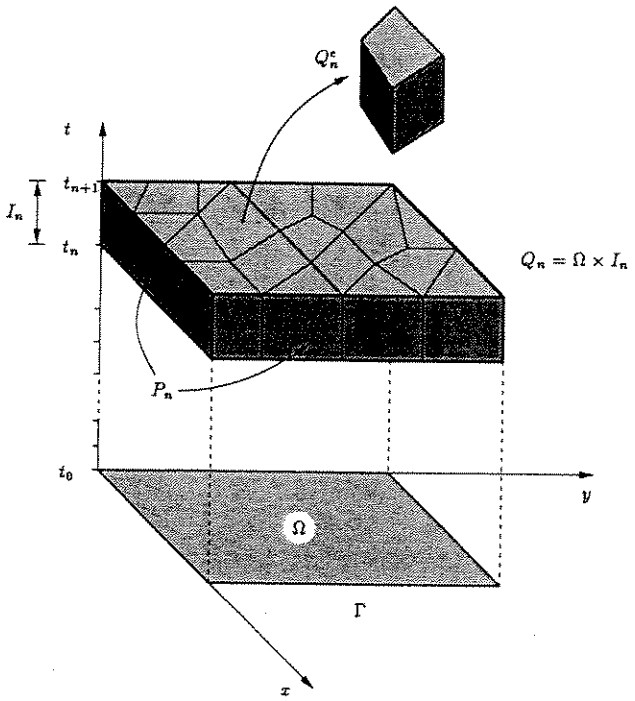


Fig. 3.1.1. A space–time slab.

each space–time slab, but are discontinuous across the interfaces of the slabs, namely at times $t_1, t_2, \ldots, t_{N-1}$. The finite element spaces are defined as

trial functions:

$$\mathcal{S}_n^h = \{V^h \mid V^h \in (C^0(Q_n))^m, V^h|_{Q_n^e} \in (\mathcal{P}_k(Q_n^e))^m, \boldsymbol{q}(V^h) = \boldsymbol{g}(t) \text{ on } P_n\} ; \tag{3.4}$$

weighting functions:

$$\mathcal{V}_n^h = \{W^h \mid W^h \in (C^0(Q_n))^m, W^h|_{Q_n^e} \in (\mathcal{P}_k(Q_n^e))^m, \boldsymbol{q}'(W^h) = \boldsymbol{0} \text{ on } P_n\} ; \tag{3.5}$$

where m is the number of degrees of freedom; $\boldsymbol{q}$, $\boldsymbol{q}' : \mathcal{R}^m \mapsto \mathcal{R}^i$, $0 \leq i \leq m$, are the nonlinear boundary condition transformation functions and $\boldsymbol{g}$ is the prescribed boundary condition (see Section 5 for detailed treatment of boundary conditions).

Considering that the finite element functions are discontinuous at the space–time slab interfaces, let

$$W(t_n^{\pm}) = \lim_{\varepsilon \to 0^{\pm}} W(t_n + \varepsilon) . \tag{3.6}$$

And define the *jump* in time of W as

$$[\![W(t_n)]\!] = W(t_n^+) - W(t_n^-) . \tag{3.7}$$

3.2. Weighted residual formulation

The statement of our finite element space–time weighted residual formulation is as follows: Within each Q_n, $n = 0, \ldots, N-1$, find $\boldsymbol{V}^h \in \mathcal{S}_n^h$ such that for all $\boldsymbol{W}^h \in \mathcal{V}_n^h$ the following variational equation is satisfied:

$$\begin{aligned}
&\int_{Q_n} (-\boldsymbol{W}_{,i}^h \cdot \boldsymbol{U}(\boldsymbol{V}^h) - \boldsymbol{W}_{,i}^h \cdot \boldsymbol{F}_i(\boldsymbol{V}^h) + \boldsymbol{W}_{,i}^h \cdot \tilde{\boldsymbol{K}}_{ij} \boldsymbol{V}_{,j}^h + \boldsymbol{W}^h \cdot \tilde{\boldsymbol{C}} \boldsymbol{V}^h)\, \mathrm{d}Q \\
&\quad + \int_{\Omega} (\boldsymbol{W}^h(t_{n+1}^-) \cdot \boldsymbol{U}(\boldsymbol{V}^h(t_{n+1}^-)) - \boldsymbol{W}^h(t_n^+) \cdot \boldsymbol{U}(\boldsymbol{V}^h(t_n^-)))\, \mathrm{d}\Omega \\
&\quad + \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{Q_n^e} (\mathscr{L}\boldsymbol{W}^h) \cdot \boldsymbol{\tau}(\mathscr{L}\boldsymbol{V}^h)\, \mathrm{d}Q + \sum_{e=1}^{(n_{\mathrm{el}})_n} \int_{Q_n^e} \nu^h \bar{\nabla}_\xi \boldsymbol{W}^h \cdot \left[\begin{array}{c} \nwarrow \\ \tilde{\boldsymbol{A}}_0 \\ \searrow \end{array} \right] \bar{\nabla}_\xi \boldsymbol{V}^h\, \mathrm{d}Q \\
&= \int_{P_n} \boldsymbol{W}^h \cdot (-\boldsymbol{F}_i(\boldsymbol{V}^h) + \boldsymbol{F}_i^{\mathrm{d}}(\boldsymbol{V}^h)) n_i\, \mathrm{d}P\,. \qquad (3.8)
\end{aligned}$$

The first, second and last integrals in (3.8) constitute the time-discontinuous Galerkin formulation. The flux terms, including the time flux term, are formulated in the integrated-by-part form, which results in conservation of fluxes under inexact quadrature rules. The non-integrated-by-parts form, in contrast, results in loss of conservation.

The time boundary integral resulting from the integration-by-parts of the time flux term is added to the *jump condition*

$$\int_{\Omega} \boldsymbol{W}^h(t_n^+) \cdot [\![\boldsymbol{U}(\boldsymbol{V}^h(t_n))]\!]\, \mathrm{d}\Omega \qquad (3.9)$$

to give the second integral in (3.8). This jump condition is the mechanism by which the information is propagated from one space–time slab to the next. In other words, the jump condition imposes a weakly enforced initial condition for the space–time slabs. The role the jump condition plays is to add a consistent and higher-order numerical dissipation, which facilitates the stability proof. Strongly enforced initial conditions lead to a less stable method, and only under certain restrictions can stability be readily shown; see [25].

The boundary integrals resulting from the integration-by-parts of the Euler and diffusive flux terms are written on the right-hand side of (3.8). These terms give rise to a set of natural boundary conditions, which will be examined in Section 5.

The third integral in (3.8) is the *least-squares operator*. $\mathscr{L}$ is defined as the compressible Navier–Stokes differential operator:

$$\mathscr{L} = \tilde{\boldsymbol{A}}_0 \partial/\partial t + \tilde{\boldsymbol{A}}_i \partial/\partial x_i - (\partial/\partial x_i)(\tilde{\boldsymbol{K}}_{ij} \partial/\partial x_j) + \tilde{\boldsymbol{C}}\,. \qquad (3.10)$$

$\boldsymbol{\tau}$ is an $m \times m$ symmetric positive-semidefinite *least-squares matrix* of intrinsic time scales (see [5]). Definition of this matrix, which will be examined in detail in the next section, greatly influences the behavior of the solution. The least-squares integral is only defined in the interior of the elements. Hence, the integrals on the boundaries of the elements are excluded. The continuity of the diffusive fluxes is weakly enforced by the Galerkin integral in the usual way.

The fourth integral in (3.8) is the *discontinuity-capturing operator*. (See [6, 22] for initial presentation of ideas of this kind.) Here, $\widehat{\nabla}_\xi$ is defined as the generalized local coordinates gradient operator. Depending on the differential terms present in $\mathscr{L}$, $\widehat{\nabla}_\xi$ may include up to second-order local derivatives. (The precise definition will be given in the next section.) ν^h is a scalar *discontinuity-capturing factor* of dimension one over time, and

$$\left[\begin{array}{c} \nwarrow \\ \tilde{A}_0 \\ \searrow \end{array} \right] \stackrel{\text{def}}{=} \begin{bmatrix} \tilde{A}_0 & & \\ & \ddots & \\ & & \tilde{A}_0 \end{bmatrix}. \tag{3.11}$$

Using two different design philosophies, we propose two definitions for ν^h, namely
(1) *Linear form*:

$$\nu^h = |\mathscr{L}V^h|_\tau \Big/ \left| \left[\begin{array}{c} \nwarrow \\ \tilde{A}_0 \\ \searrow \end{array} \right] \widehat{\nabla}_\xi V^h \right|_\tau ; \tag{3.12}$$

(2) *Quadratic form*:

$$\nu^h = 2|\mathscr{L}V^h|_\tau^2 / |\widehat{\nabla}_\xi V^h|_{\tilde{A}_0}^2 ; \tag{3.13}$$

where $|\mathscr{L}V^h|_\tau^2 = \mathscr{L}V^h \cdot \tau \mathscr{L}V^h$. Note that both definitions of ν^h are proportional to powers of the residual $\mathscr{L}V^h$. The detailed derivation of both forms will be presented in Section 3.4.

The variational equation (3.8) is a consistent formulation in the sense that it is exactly satisfied by the exact solution of the compressible Navier–Stokes equations.

3.3. Least-squares operator

The general construction of the least-squares operator is quite simple. Let $\mathscr{L}$ be the compressible Navier–Stokes differential operator (3.10). Then the least-squares operator is defined as

$$\sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} (\mathscr{L}W^h) \cdot \tau(\mathscr{L}V^h) \, \mathrm{d}Q . \tag{3.14}$$

The structure of τ is the crux of this method. Although the construction of the least-squares operator is simple and straightforward, the construction of τ is not. Only under severe restrictions, such as those for one-dimensional and/or diagonalizable systems, can one find *optimal definitions*; see [5]. These cases, however, do provide us with general design properties to emulate. Simple model problems show that τ *cannot* be proportional to the identity matrix. Linear error estimates, convergence proofs and dimensional analysis also provide us with design conditions to satisfy, but these analyses are not sufficient to produce a unique definition for τ. Hughes and Mallet [5] and Mallet [7] have determined a definition for τ for the Navier–Stokes equations by generalizing the definition of τ for the restricted cases. The form of this definition will be presented in Section 3.3.2. We endeavor to improve upon this definition herein. Consequently, we will propose a more general definition in Section 3.3.1. In Section 3.6, we will show that the new definition satisfies the properties needed to

establish linear error estimates and nonlinear convergence proofs. For an overview of the Galerkin/least-squares method we refer to [3, 26].

Before going on, let us look briefly at the predecessor of the least-squares operator, namely the SUPG operator; for details we refer to [5, 7]. The general form of this operator is

$$\sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} (\tilde{A}_i W_{,i}^h) \cdot \tau(\mathscr{L} V^h)\, \mathrm{d}Q\,. \tag{3.15}$$

The weighting function 'slot' consists only of the first-order part of the differential operator. Consequently, in general, this operator is not symmetric. In the presence of diffusion, the stability of the SUPG operator is dependent on the exact structure of the τ matrix and a so-called inverse error estimate (see [11]). These limit the generality of τ. In contrast, the least-squares operator is stable for any symmetric positive-semidefinite matrix τ.

3.3.1. A general design for τ

In this section, we present a general design for τ. Denote by ∇_ξ and $\nabla_{\xi'}$ the local gradient in element spatial and space–time coordinate systems, respectively. That is,

$$\nabla_\xi \overset{\text{def}}{=} \begin{bmatrix} (\partial/\partial\xi_1)\boldsymbol{I}_n \\ \vdots \\ (\partial/\partial\xi_d)\boldsymbol{I}_n \end{bmatrix}, \quad (d\cdot n)\times n\,, \tag{3.16}$$

$$\nabla_{\xi'} \overset{\text{def}}{=} \begin{bmatrix} (\partial/\partial\xi_0)\mathbf{I}_n \\ (\partial/\partial\xi_1)\mathbf{I}_n \\ \vdots \\ (\partial/\partial\xi_d)\mathbf{I}_n \end{bmatrix}, \quad (d'\cdot n)\times n\,, \tag{3.17}$$

where $\mathbf{I}_n$ is the identity matrix of dimension n, and $d' \overset{\text{def}}{=} d+1$. Let ∇_ξ^i be ∇_ξ applied i times. For example,

$$\nabla_\xi^2 = \nabla_\xi \nabla_\xi = \begin{bmatrix} (\partial^2/\partial\xi_1^2)\mathbf{I}_n \\ (\partial^2/\partial\xi_1\partial\xi_2)\mathbf{I}_n \\ \vdots \\ (\partial^2/\dot{\nabla}\xi_d^2)\mathbf{I}_n \end{bmatrix}, \quad (d^2\cdot n)\times n\,. \tag{3.18}$$

Let the generalized gradient operator in the local element coordinates be defined as

$$\widehat{\nabla}_\xi = \begin{bmatrix} \mathbf{I}_n \\ \nabla_{\xi'} \\ \nabla_\xi^2 \end{bmatrix}, \quad ((1+d'+d^2)\cdot n)\times n\,. \tag{3.19}$$

The number of terms included in $\widehat{\nabla}_\xi$ depends on the differential operator $\mathscr{L}$. The above definition accounts for all the gradient terms in (3.10).

Let the generalized Jacobian matrix in the local element coordinates be defined as

$$\widehat{A}_\xi = \begin{bmatrix} \tilde{C} \\ (\partial\xi_0/\partial x_0)\tilde{A}_0 \\ (\partial\xi_1/\partial x_i)\tilde{A}_i \\ \vdots \\ (\partial\xi_d/\partial x_i)\tilde{A}_i \\ (\partial\xi_1/\partial x_i)(\partial\xi_1/\partial x_j)\tilde{K}_{ij} \\ \vdots \\ (\partial\xi_d/\partial x_i)(\partial\xi_d/\partial x_j)\tilde{K}_{ij} \end{bmatrix}, \quad ((1+d'+d^2)\cdot m)\times m\,. \tag{3.20}$$

Assuming that $\widehat{A}_\xi$ is locally constant, the differential operator acting on V^h can be written as

$$\mathscr{L}V^h = \widehat{A}_\xi^{\mathrm{t}}\widehat{\nabla}_\xi V^h\,. \tag{3.21}$$

Note that the assumption on $\widehat{A}_\xi$ is made only for purposes of designing τ. Moreover, this assumption is only required in the presence of diffusion.

The least-squares operator for the element Q_n^e can now be written as

$$\int_{Q_n^e}(\mathscr{L}W^h)\cdot\tau(\mathscr{L}V^h)\,\mathrm{d}Q = \int_{Q_n^e}\widehat{\nabla}_\xi W^h\cdot\widehat{K}_\xi\widehat{\nabla}_\xi V^h\,\mathrm{d}Q\,, \tag{3.22}$$

where

$$\widehat{K}_\xi \overset{\text{def}}{=} \widehat{A}_\xi\tau\widehat{A}_\xi^{\mathrm{t}} \tag{3.23}$$

is a $((1+d'+d^2)\cdot m)\times((1+d'+d^2)\cdot m)$ symmetric positive-semidefinite matrix, with rank no greater than the rank of τ (i.e., $\leq m$). The $m\times m$ blocks of $\widehat{K}_\xi$ are dimensionally proportional to $\tilde{A}_0$ and to inverse-powers of the temporal and spatial mesh size.

For low-order elements, we proceed by designing τ such that

$$\widehat{K}_\xi = (\widehat{A}_\xi\tilde{A}_0^{-1}\widehat{A}_\xi^{\mathrm{t}})^{1/2}\,, \tag{3.24}$$

where the square root is taken on the nondegenerate part and with respect to the metric tensor $\tilde{A}_0$. In other words, the square-root inverse is defined based on the m nonzero eigenvalues of the following generalized eigenvalue problem:

$$\left((\widehat{A}_\xi\tilde{A}_0^{-1}\widehat{A}_\xi^{\mathrm{t}}) - \lambda_i^2\begin{bmatrix}\nwarrow & \tilde{A}_0 & \nearrow\end{bmatrix}\right)\boldsymbol{\Phi}_i = \mathbf{0}\,,\quad i=1,\ldots,m\,, \tag{3.25}$$

$$[\boldsymbol{\Phi}_1\cdots\boldsymbol{\Phi}_m]^{\mathrm{t}}\begin{bmatrix}\nwarrow & \tilde{A}_0 & \nearrow\end{bmatrix}[\boldsymbol{\Phi}_1\cdots\boldsymbol{\Phi}_m] = \mathbf{I}_m\,, \tag{3.26}$$

which yields

$$\widehat{K}_\xi = \begin{bmatrix}\nwarrow & \tilde{A}_0 & \nearrow\end{bmatrix}\left(\sum_{i=1}^{m}\lambda_i\boldsymbol{\Phi}_i\boldsymbol{\Phi}_i^{\mathrm{t}}\right)\begin{bmatrix}\nwarrow & \tilde{A}_0 & \nearrow\end{bmatrix}. \tag{3.27}$$

As a consequence

$$\widehat{K}_\xi \left[\overleftarrow{\tilde{A}_0^{-1}} \right] \widehat{K}_\xi = \widehat{A}_\xi \tilde{A}_0^{-1} \widehat{A}_\xi^t . \tag{3.28}$$

$\widehat{K}_\xi$ can be viewed as the 'absolute value' of the rectangular matrix A_ξ with respect to the metric tensor $\tilde{A}_0^{-1}$. In the case of one-dimensional pure advection systems, the combination of this operator and the Galerkin integral results in full-upwind differencing on each mode of the system.

The above diffusivity matrix can be attained by setting

$$\tau = \tilde{L}^{-t} \left(\tilde{L}^{-1} \widehat{A}_\xi^t \left[\overleftarrow{\tilde{A}_0^{-1}} \right] \widehat{A}_\xi \tilde{L}^{-t} \right)^{-1/2} \tilde{L}^{-1} = \tilde{A}_0^{-1} \left(\widehat{A}_\xi^t \left[\overleftarrow{\tilde{A}_0^{-1}} \right] \widehat{A}_\xi \tilde{A}_0^{-1} \right)^{-1/2} , \tag{3.29}$$

where

$$\tilde{L}\tilde{L}^t \overset{\text{def}}{=} \tilde{A}_0 \quad \text{(symmetric decomposition)} \tag{3.30}$$

and the square-root inverse is taken on a small matrix of size m. Note that (3.30) represents any symmetric decomposition, e.g., Cholesky decomposition.

To verify that the above definition of τ results in (3.24), let $\bar{K}_\xi$ and $\bar{\tau}$ be the nondimensional counterparts of $\widehat{K}_\xi$ and τ defined as

$$\bar{K}_\xi = \left[\overleftarrow{\tilde{L}^{-1}} \right] \widehat{K}_\xi \left[\overleftarrow{\tilde{L}^{-t}} \right] , \tag{3.31}$$

$$\bar{\tau} = \tilde{L}^t \tau \tilde{L} . \tag{3.32}$$

Then from (3.24)

$$\bar{K}_\xi^2 = \left[\overleftarrow{\tilde{L}^{-1}} \right] (\widehat{A}_\xi \tilde{A}_0^{-1} \widehat{A}_\xi^t) \left[\overleftarrow{\tilde{L}^{-t}} \right] , \tag{3.33}$$

where we have used the fact that the square root in $\widehat{K}_\xi$ is taken with respect to $\tilde{A}_0$. Also, from (3.23) and (3.29)–(3.32)

$$\begin{aligned} \bar{K}_\xi^2 &= \left[\overleftarrow{\tilde{L}^{-1}} \right] \widehat{A}_\xi \tau \widehat{A}_\xi^t \left[\overleftarrow{\tilde{A}_0^{-1}} \right] \widehat{A}_\xi \tau \widehat{A}_\xi^t \left[\overleftarrow{\tilde{L}^{-t}} \right] \\ &= \left(\left[\overleftarrow{\tilde{L}^{-1}} \right] \widehat{A}_\xi \tilde{L}^{-t} \right) \bar{\tau} \left(\tilde{L}^{-1} \widehat{A}_\xi^t \left[\overleftarrow{\tilde{A}_0^{-1}} \right] \widehat{A}_\xi \tilde{L}^{-t} \right) \bar{\tau} \left(\tilde{L}^{-1} \widehat{A}_\xi^t \left[\overleftarrow{\tilde{L}^{-t}} \right] \right) \\ &= \left[\overleftarrow{\tilde{L}^{-1}} \right] (\widehat{A}_\xi \tilde{A}_0^{-1} \widehat{A}_\xi^t) \left[\overleftarrow{\tilde{L}^{-t}} \right] . \end{aligned} \tag{3.34}$$

The above equation and (3.33) are the same. This completes the verification.

Substitution of (3.20) and the definition of the Jacobian matrices from (2.23), (2.24) and

(2.26) into (3.29) leads to

$$\boldsymbol{\tau} = \tilde{\boldsymbol{A}}_0^{-1}\left(\boldsymbol{C}^2 + \left(\frac{\partial \xi_0}{\partial x_0}\right)^2 \mathbf{I}_m + \left(\frac{\partial \xi_i}{\partial x_j}\,\frac{\partial \xi_i}{\partial x_k}\right)\boldsymbol{A}_j\boldsymbol{A}_k + \left(\frac{\partial \xi_i}{\partial x_k}\,\frac{\partial \xi_j}{\partial x_l}\,\frac{\partial \xi_j}{\partial x_m}\,\frac{\partial \xi_i}{\partial x_n}\right)\boldsymbol{K}_{kl}\boldsymbol{K}_{mn}\right)^{-1/2} . \tag{3.35}$$

It is obvious from the first equality in (3.29) that $\boldsymbol{\tau}$ is symmetric, and from (3.35) that $\boldsymbol{\tau}$ is positive-definite with rank m. When solving steady problems (i.e., $\partial/\partial t = 0$), the second term in the square-root inverse in (3.35) may be set to zero.

The square-root inverse in (3.35) can be computed either by employing the Cayley–Hamilton theorem (see [27]), or iteratively by using some form of Newton's algorithm, or by solving an eigenvalue problem.

3.3.2. An alternative design for $\boldsymbol{\tau}$

In this section we briefly describe the definition for $\boldsymbol{\tau}$ presented by Hughes and Mallet [5]. Their presentation was restricted to the steady case. We generalize it to the unsteady space–time case. In this definition, $\boldsymbol{\tau}$ is first formulated for pure advection, and then adjusted for the presence of diffusion. The advective $\boldsymbol{\tau}$ is defined identically to the one in the previous section, namely

$$\boldsymbol{\tau}^{\mathrm{adv}} = \tilde{\boldsymbol{A}}_0^{-1}\left(\left(\frac{\partial \xi_0}{\partial x_0}\right)^2 \mathbf{I}_m + \left(\frac{\partial \xi_i}{\partial x_j}\,\frac{\partial \xi_i}{\partial x_k}\right)\boldsymbol{A}_j\boldsymbol{A}_k\right)^{-1/2} . \tag{3.36}$$

Then $\boldsymbol{\tau}^{\mathrm{adv}}$ is decomposed in terms of its eigenvalues and eigenvectors:

$$\boldsymbol{\tau}^{\mathrm{adv}} = \sum_{i=1}^{m} \tau_i^{\mathrm{adv}} \boldsymbol{\phi}_i \boldsymbol{\phi}_i^{\mathrm{t}} . \tag{3.37}$$

Note that the eigenvalues τ_i^{adv} are proportional to mesh spacing and inversely proportional to advective speeds. Since $\boldsymbol{\tau}$ is symmetric and dimensionally proportional to $\tilde{\boldsymbol{A}}_0^{-1}$, $\boldsymbol{\Phi} = {}_{\mathrm{def}}[\boldsymbol{\Phi}_1 \ldots \boldsymbol{\Phi}_m]$ can be scaled such that

$$\boldsymbol{\Phi}^{\mathrm{t}} \tilde{\boldsymbol{A}}_0 \boldsymbol{\Phi} = \mathbf{I}_m . \tag{3.38}$$

The effect of diffusion is then incorporated into $\boldsymbol{\tau}$ based on the one-dimensional linear scalar advective-diffusive model equation by modifying the eigenvalues of $\boldsymbol{\tau}^{\mathrm{adv}}$ as follows: Let

$$\sigma_i = \frac{1}{d} \sum_{j,k,l=1}^{d} \boldsymbol{\Phi}_i^{\mathrm{t}}\left(\frac{\partial \xi_j}{\partial x_l}\,\frac{\partial \xi_k}{\partial x_l}\,\tilde{\boldsymbol{K}}_{ll}\right)\boldsymbol{\Phi}_i , \quad i = 1, \ldots, m , \tag{3.39}$$

be the projection of the diffusivity matrix in the local element coordinates onto the modes of $\boldsymbol{\tau}^{\mathrm{adv}}$. In other words, σ_i is a measure of the diffusivity of the system in the ith eigenmode of $\boldsymbol{\tau}^{\mathrm{adv}}$. The element Peclet numbers can be defined as

$$\alpha_i = 1/\tau_i^{\mathrm{adv}} \sigma_i , \quad i = 1, \ldots, m . \tag{3.40}$$

Based on this, τ_i is formulated as

$$\tau_i = \tau_i^{\text{adv}} \tilde{\xi}(\alpha_i) , \tag{3.41}$$

where

$$\tilde{\xi}(\alpha_i) = \coth(\alpha_i) - 1/\alpha_i . \tag{3.42}$$

Finally

$$\boldsymbol{\tau} = \sum_{i=1}^{m} \tau_i \boldsymbol{\Phi}_i \boldsymbol{\Phi}_i^{\mathrm{t}} . \tag{3.43}$$

This formulation is more complicated than the previous definition of $\boldsymbol{\tau}$. In particular, it requires the solution of the eigenvalue problem. Under the assumption that the elements have approximately equal-length sides, Mallet [7] has analytically solved this problem.

Our current implementation of $\boldsymbol{\tau}$ is based on a simplified version of the $\boldsymbol{\tau}$ presented above. In two dimensions, let (s, n) be an orthogonal coordinate system with s pointing in the direction of the streamline. Then we rewrite $\boldsymbol{\tau}^{\text{adv}}$, (3.36), in (s, n) coordinates as

$$\boldsymbol{\tau}_s^{\text{adv}} = \tilde{\boldsymbol{A}}_0^{-1} \left((\partial \xi_0 / \partial x_0)^2 \boldsymbol{I}_m + \alpha_{ss} \boldsymbol{A}_s \boldsymbol{A}_s + \alpha_{sn} \boldsymbol{A}_s \boldsymbol{A}_n + \alpha_{ns} \boldsymbol{A}_n \boldsymbol{A}_s + \alpha_{nn} \boldsymbol{A}_n \boldsymbol{A}_n \right)^{-1/2} , \tag{3.44}$$

where

$$\alpha_{ss} = \frac{\partial \xi_i}{\partial x_s} \frac{\partial \xi_i}{\partial x_s} ; \qquad \alpha_{sn} = \alpha_{ns} = \frac{\partial \xi_i}{\partial x_s} \frac{\partial \xi_i}{\partial x_n} ; \qquad \alpha_{nn} = \frac{\partial \xi_i}{\partial x_n} \frac{\partial \xi_i}{\partial x_n} , \tag{3.45}$$

and $\boldsymbol{A}_s$ and $\boldsymbol{A}_n$ are Euler Jacobians in the (s, n) coordinate system. To analytically decompose $\boldsymbol{\tau}_s^{\text{adv}}$ as in (3.37), we assume the case $\alpha_{sn} = \alpha_{ns} = 0$. This case is similar to the one presented in [7] where, in addition, he assumed $\alpha_{ss} = \alpha_{nn} = (2/h)^2$. Once the eigenvectors of the simplified $\boldsymbol{\tau}_s^{\text{adv}}$ are transformed from (s, n) to the global coordinate system, we proceed by computing $\boldsymbol{\tau}$ as in (3.39)–(3.43). All the results presented herein are based on this formulation.

3.3.3. *One-dimensional model problem*

To gain additional insight into the behavior of the Galerkin/least-squares method, and in particular the effect of the different definitions of $\boldsymbol{\tau}$ on the approximate solution, we analyze the one-dimensional linear scalar advection-diffusion model problem:

$$a u_{,x} = \kappa u_{,xx} \quad \text{for } x \in]0, 1[, \tag{3.46}$$

where u is the unknown; a is the advective speed; and κ is the diffusion coefficient. (The time-dependent case is analyzed in [14]). To the above equation we append the following boundary conditions:

$$u(0) = 0 , \tag{3.47}$$

$$u(1) = 1 , \tag{3.48}$$

The exact solution of (3.46)–(3.48) is

$$u = (e^{(a/\kappa)x} - 1)/(e^{a/\kappa} - 1)\,. \tag{3.49}$$

When the a/κ ratio is large the exact solution exhibits a sharp gradient near $x = 1$.

The statement of the Galerkin/least-squares method for (3.46) is: Find $u^h \in \mathcal{S}^h$ such that for all $w^h \in \mathcal{V}^h$

$$\int_\Omega (w^h a u^h_{,x} + w^h_{,x} \kappa u^h_{,x})\,dx + \sum_{e=1}^{n_{el}} \int_{\Omega^e} (a w^h_{,x} - \kappa w^h_{,xx}) \tau (a u^h_{,x} - \kappa u^h_{,xx})\,dx = 0\,. \tag{3.50}$$

It is well documented that, for piecewise linear elements, a τ can be determined such that the H^1 seminorm of the solution error, $u^h - u$, is minimized. This condition is equivalent to having a nodally exact solution, and is known as a *superconvergence* result. This *optimal* τ is

$$\tau_{opt} = (h/2|a|)\tilde{\xi}_{opt}(\alpha)\,, \tag{3.51}$$

$$\tilde{\xi}_{opt}(\alpha) = \coth(\alpha) - 1/\alpha\,, \tag{3.52}$$

$$\alpha = h|a|/2\kappa\,, \tag{3.53}$$

where h is the element mesh size and α is the local element Peclet number. The difference stencil resulting from Galerkin/least-squares with τ_{opt} has zero local truncation error at the nodes, which is equivalent to having a nodally exact solution.

The τ based only on the advection operator can be written as

$$\tau_{adv} = (h/2|a|)\tilde{\xi}_{adv}(\alpha)\,, \tag{3.54}$$

$$\tilde{\xi}_{adv}(\alpha) = 1\,. \tag{3.55}$$

The local truncation error analysis in this case reveals only first-order accuracy.

The τ defined from the general design conditions of Section 3.3.1 yields

$$\tau_{gen} = ((2a/h)^2 + (4\kappa/h^2)^2)^{-1/2} = (h/2|a|)\tilde{\xi}_{gen}(\alpha)\,, \tag{3.56}$$

$$\tilde{\xi}_{gen}(\alpha) = \sqrt{\alpha^2/(1+\alpha^2)}\,. \tag{3.57}$$

For general systems, this τ is considerably simpler than τ_{opt}. However, local truncation error analysis shows only second-order accuracy in this case. The error can be improved to fourth-order by modifying the definition of τ_{gen} slightly to

$$\tau_{mod} = ((2a/h)^2 + 9(4\kappa/h^2)^2)^{-1/2} = (h/2|a|)\tilde{\xi}_{mod}(\alpha)\,, \tag{3.58}$$

$$\tilde{\xi}_{mod}(\alpha) = \sqrt{\alpha^2/(9+\alpha^2)}\,. \tag{3.59}$$

In all four cases, τ can be written in terms of $h/2|a|$ multiplied by a diffusion correction factor $\tilde{\xi}(\alpha)$. These factors are compared in Figure 3.3.1. Note that $\tilde{\xi}_{opt}$ produces the least amount of diffusion. $\tilde{\xi}_{mod}$ matches $\tilde{\xi}_{opt}$ very well as α approaches zero and infinity; and in the midrange of α, $\tilde{\xi}_{mod}$ is a good approximation to $\tilde{\xi}_{opt}$.

The nodally exact properties of the τ_{opt} case and the fourth-order accuracy of the τ_{mod} case do not extend to multi-dimensional problems or general advective-diffusive systems. In contrast, the error estimates in L_2 and H^1 do extend to these situations. Consequently, we proceed by analyzing these error measures, using an example: $a = 100$ and $\kappa = 1$. The exact solution of this problem is an exponential function that exhibits a sharp boundary layer. This problem was solved using a series of nested meshes starting with two elements. The L_2 norm and the H^1 seminorm of the solution errors for the four cases plus the Galerkin method are compared in Fig. 3.3.2.

The Galerkin method (i.e., $\tau = 0$) is second-order accurate, as measured in the L_2 norm, in the entire range of h. This is, however, misleading since for $\alpha > 1$ the solution of the Galerkin method is oscillatory and its absolute error is an order of magnitude worse than the Galerkin/least-squares methods (i.e., $\tau \neq 0$). For $\alpha > 1$, all four nonzero definitions of τ lead to first-order accuracy. This is expected, since the measured error comes from the interpolation error, which for sharp gradients is only first-order. The τ_{adv} method remains first-order even as h approaches zero; in contrast, the other three methods are second-order accurate for $\alpha < 1$. The L_2 errors of the τ_{mod} and τ_{opt} methods are nearly identical. This justifies the interchanging of τ_{opt} with τ_{mod}, even for this simple model problem. The H^1 seminorm of the error for the Galerkin method is first-order, while for non-zero τ methods, the errors are zero-order for $\alpha > 1$ and first-order for $\alpha < 1$. With the exception of the τ_{adv} method, the H^1 errors of all the methods are nearly the same for small α.

To demonstrate that the Galerkin/least-squares method is indeed a higher-order accurate method, the above example was solved using quadratic elements. For this element, no optimal definition of τ exists. Numerical experiments indicate that τ for the quadratic element should be approximately half the τ for the linear element. Hence, the problem was solved using quadratic elements with τ reduced by one half. The L_2 norm and H^1 seminorm of the solution

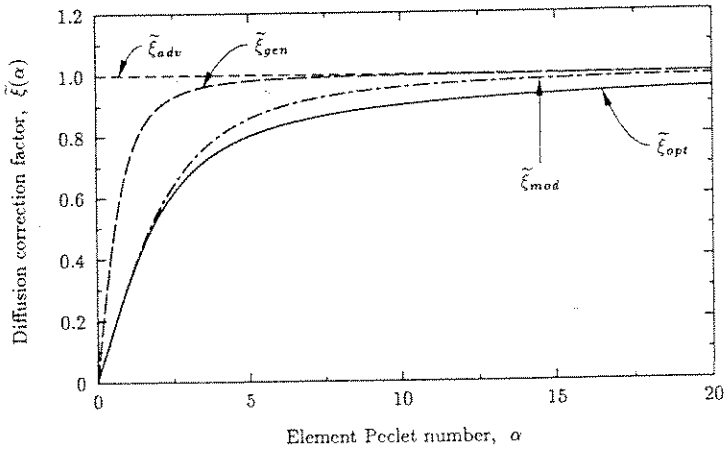


Fig. 3.3.1. Diffusion correction factors applied for the one-dimensional advective-diffusive model problem.

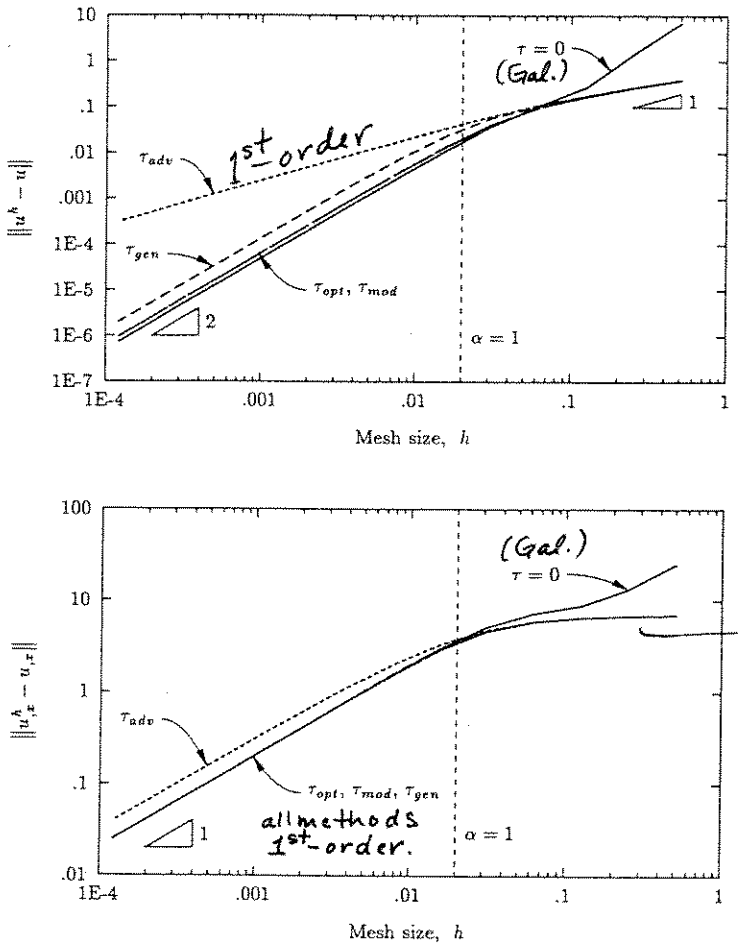


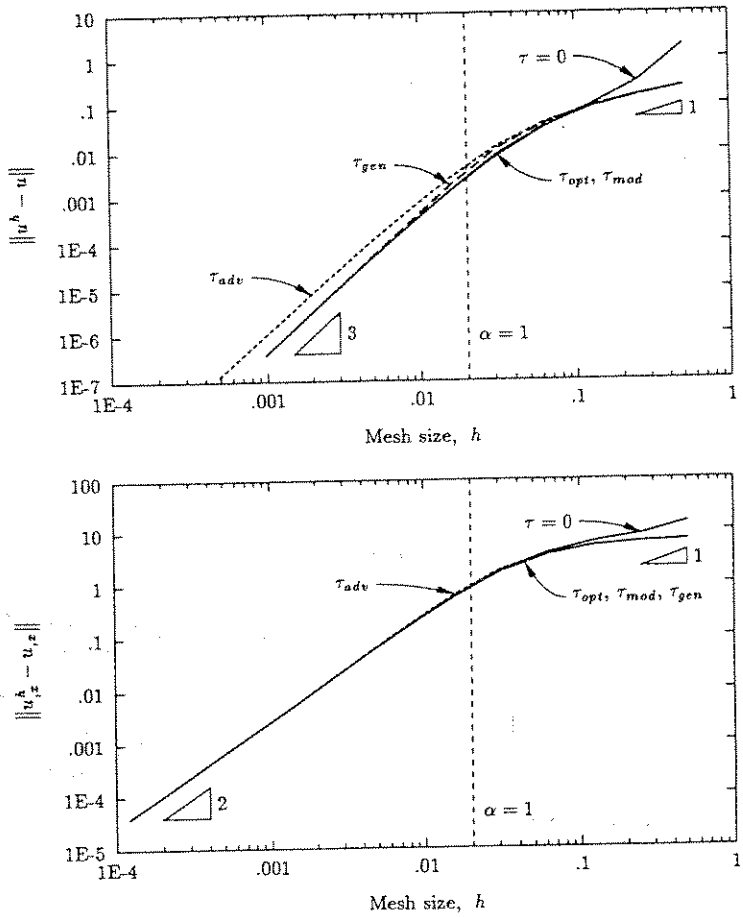
Fig. 3.3.2. Advective-diffusive problem ($a = 100$, $\kappa = 1$). L_2 norm and H^1 seminorm of solution error. Galerkin and Galerkin least-squares methods with equal-length linear elements.

errors are compared in Fig. 3.3.3. This figure shows that for small h all the methods are third-order accurate in the L_2 norm, and for large element Peclet numbers the Galerkin/least-squares methods with any of the τ definitions are better than the pure Galerkin method.

3.4. *Discontinuity-capturing operators*

Up to the definition of τ, the least-squares operator is in canonical form. This is, however, not true for the discontinuity-capturing operator. The discontinuity-capturing operator needs to satisfy only a few design conditions: In order to control the oscillations, this operator should act in the direction of the gradient; for consistency it should be proportional to the residual, $\mathcal{L}V^h$; and for accuracy it should vanish quickly in smooth regions of the solution.

In this section, we present two operators that satisfy the above conditions. The first operator, the so-called *linear form*, is an extension of Hughes and Mallet [6] to general systems in time and space. The second operator, the *quadratic form*, has features in common



• τ_{adv} has the right slope here, but the constant is worse

Fig. 3.3.3. Advective-diffusive problem ($a = 100$, $\kappa = 1$). L_2 norm and H^1 seminorm of solution error. Galerkin and Galerkin/least-squares methods with equal-length quadratic elements.

with the discontinuity-capturing operator proposed by Galeão and Dutra do Carmo [24] for the scalar advective-diffusive equation.

3.4.1. The linear discontinuity-capturing operator

Consider a $((1 + d' + d^2) \cdot m) \times m$ rank one operator, $\widehat{\boldsymbol{D}}_\xi$, to be the 'projection' of the residual $\mathscr{L}\boldsymbol{V}^h$ onto the direction of the generalized local gradient. That is, $\widehat{\boldsymbol{D}}_\xi$ satisfies the following two conditions:

$$\text{(i)} \qquad \widehat{\boldsymbol{D}}_\xi \cdot \widehat{\boldsymbol{\nabla}}_\xi \boldsymbol{V}^h = \mathscr{L}\boldsymbol{V}^h \,, \tag{3.60}$$

$$\text{(ii)} \qquad \widehat{\boldsymbol{D}}_\xi \cdot \boldsymbol{Z} = 0 \quad \text{for all } \boldsymbol{Z} \text{ such that } \boldsymbol{Z} \cdot \left[\overleftarrow{\tilde{\boldsymbol{A}}_0} \rightarrow \right] \widehat{\boldsymbol{\nabla}}_\xi \boldsymbol{V}^h = 0 \,. \tag{3.61}$$

Clearly

$$\widehat{\boldsymbol{D}}_\xi = \left(\left[\overleftarrow{\tilde{\boldsymbol{A}}_0} \rightarrow \right] \widehat{\boldsymbol{\nabla}}_\xi \boldsymbol{V}^h \right) (\mathscr{L}\boldsymbol{V}^h)^t \Big/ \left(\widehat{\boldsymbol{\nabla}}_\xi \boldsymbol{V}^h \cdot \left[\overleftarrow{\tilde{\boldsymbol{A}}_0} \rightarrow \right] \widehat{\boldsymbol{\nabla}}_\xi \boldsymbol{V}^h \right) . \tag{3.62}$$

The discontinuity-capturing operator for an element is then defined as

$$\int_{Q_n^e} (\widehat{D}_\xi^t \widehat{\nabla}_\xi W^h) \cdot \tau_D(\mathscr{L}V^h)\, \mathrm{d}Q\,, \tag{3.63}$$

or equivalently as

$$\int_{Q_n^e} (\widehat{D}_\xi^t \widehat{\nabla}_\xi W^h) \cdot \tau_D(\widehat{D}_\xi^t \widehat{\nabla}_\xi V^h)\, \mathrm{d}Q\,, \tag{3.64}$$

where τ_D is the discontinuity-capturing matrix.

τ_D is constructed in a similar manner to τ; see Section 3.3.1. To get a desirable form for τ_D, we use τ^{-1} as a Riemannian metric tensor in the place of $\tilde{A}_0$. This leads to the definition of τ_D (see (3.29)):

$$\tau_D = \tau^{1/2}\left(\tau^{1/2}\widehat{D}_\xi^t\left[\overleftarrow{\ \tau\ }\right]\widehat{D}_\xi \tau^{1/2}\right)^{-1/2}\tau^{1/2}\,. \tag{3.65}$$

Then τ_D is obtained via the following eigenvalue problem:

$$\left(\left(\tau^{1/2}\widehat{D}_\xi^t\left[\overleftarrow{\ \tau\ }\right]\widehat{D}_\xi \tau^{1/2}\right) - \lambda_D^2 \mathbf{I}_m\right)\boldsymbol{\Phi}_D = \mathbf{0}\,. \tag{3.66}$$

Since $\widehat{D}_\xi$ has rank one, this yields a single eigenvalue and eigenvector

$$\lambda_D^2 = \left(\widehat{\nabla}_\xi V^h \cdot \left[\overleftarrow{\ (\tilde{A}_0 \tau \tilde{A}_0)\ }\right]\widehat{\nabla}_\xi V^h\right)(\mathscr{L}V^h \cdot \tau \mathscr{L}V^h)\Big/\left(\widehat{\nabla}_\xi V^h \cdot \left[\overleftarrow{\ \tilde{A}_0\ }\right]\widehat{\nabla}_\xi V^h\right)^2, \tag{3.67}$$

$$\boldsymbol{\Phi}_D = \tau^{1/2}\mathscr{L}V^h/(\mathscr{L}V^h \cdot \tau \mathscr{L}V^h)^{1/2}\,. \tag{3.68}$$

These lead to

$$\tau_D = \frac{|\widehat{\nabla}_\xi V^h|^2_{\tilde{A}_0}}{|\mathscr{L}V^h|_\tau^3 \left|\left[\overleftarrow{\ \tilde{A}_0\ }\right]\widehat{\nabla}_\xi V^h\right|_\tau}(\tau\mathscr{L}V^h)(\tau\mathscr{L}V^h)^t\,. \tag{3.69}$$

Substituting the above equation and (3.62) in (3.64) yields the final form

$$\int_{Q_n^e} \frac{|\mathscr{L}V^h|_\tau}{\left|\left[\overleftarrow{\ \tilde{A}_0\ }\right]\widehat{\nabla}_\xi V^h\right|_\tau}\widehat{\nabla}_\xi W^h \cdot \left[\overleftarrow{\ \tilde{A}_0\ }\right]\widehat{\nabla}_\xi V^h\, \mathrm{d}Q\,. \tag{3.70}$$

This operator provides control over all the derivatives present in the differential equation. Moreover, it is dimensionally consistent and invariant with respect to rotations of the global coordinate system.

3.4.2. The quadratic discontinuity-capturing operator

The Galerkin/least-squares method produces excellent results in smooth regions of the solution. In non-smooth regions oscillations appear in the direction of the gradients. Therefore, it is desirable to increase the diffusion of the least-squares operator in that direction.

The component of the least-squares diffusion matrix, $\widetilde{K}_\xi$, in the direction of the generalized local gradients, with respect to the metric tensor $\tilde{A}_0$, is

$$\frac{\widetilde{\nabla}_\xi V^h \left(\left[\overleftarrow{\tilde{A}_0} \overrightarrow{\ } \right] \widetilde{\nabla}_\xi V^h \right)^t}{|\widetilde{\nabla}_\xi V^h|^2_{\tilde{A}_0}} \cdot \tilde{K}_\xi \frac{\widetilde{\nabla}_\xi V^h \left(\left[\overleftarrow{\tilde{A}_0} \overrightarrow{\ } \right] \widetilde{\nabla}_\xi V^h \right)^t}{|\widetilde{\nabla}_\xi V^h|^2_{\tilde{A}_0}} . \tag{3.71}$$

Using (3.21) and (3.23), (3.71) can be written as

$$\left(\left[\overleftarrow{\tilde{A}_0} \overrightarrow{\ } \right] \widetilde{\nabla}_\xi V^h \right) \frac{|\mathscr{L} V^h|^2_\tau}{|\widetilde{\nabla}_\xi V^h|^4_{\tilde{A}_0}} \left(\left[\overleftarrow{\tilde{A}_0} \overrightarrow{\ } \right] \widetilde{\nabla}_\xi V^h \right)^t . \tag{3.72}$$

Clearly, this matrix is rank one. We proceed to define the quadratic form of the discontinuity-capturing operator by replacing $\widetilde{K}_\xi$ in (3.23) by twice the above matrix. This leads to

$$\int_{Q_n^e} 2 \frac{|\mathscr{L} V^h|^2_\tau}{|\widetilde{\nabla}_\xi V^h|^2_{\tilde{A}_0}} \widetilde{\nabla}_\xi W^h \cdot \left[\overleftarrow{\tilde{A}_0} \overrightarrow{\ } \right] \widetilde{\nabla}_\xi V^h \, \mathrm{d}Q . \tag{3.73}$$

Note that the above discontinuity-capturing operator is proportional to the square of the residual $\mathscr{L} V^h$, whereas (3.70) is proportional to the first power of the residual.

3.4.3. Discussion

Both forms of the discontinuity-capturing operator are proportional to the residual $\mathscr{L} V^h$. Consequently, in the smooth regions of the solution, where $\mathscr{L} V^h$ is small, both operators are small and have little effect on the solution. In the non-smooth regions of the solution, such as near singularities, discontinuities and shocks, these operators become large and provide control over the gradients.

The idea of the nonlinear discontinuity-capturing operators is to provide strong control in the regions of the flow where there are strong gradients, yet have negligible effect in smooth regions. Presently, we prefer the quadratic form, although this remains a somewhat open issue (see, e.g., [13, 24, 28]).

3.5. Inviscid calculations

In this section, we numerically study the effect of the least-squares and discontinuity-capturing operators on the solution of steady inviscid problems with shocks. In each study, the problem was solved using:

(LS) – the Galerkin/least-squares method;

(DC-Linear) – (LS) plus the linear form of the discontinuity-capturing operator;

(DC-Quad.) – (LS) plus the quadratic form of the discontinuity-capturing operator;

(DC-Mallet) – (LS) plus the discontinuity-capturing operator employed by Mallet [7]. For the

Euler equations this operator is given by

$$\int_{Q_n^e} \tau_D \nabla \boldsymbol{W}^h \cdot \left[\overleftarrow{\tilde{\boldsymbol{A}}_0} \overrightarrow{\ } \right] \nabla \boldsymbol{V}^h \, \mathrm{d}Q \,, \tag{3.74}$$

where

$$\tau_D = \max(|\mathscr{L}\boldsymbol{V}^h|_{\tilde{A}_0^{-1}} / |\xi_{,x} \nabla \boldsymbol{V}^h|_{\tilde{A}_0} - |\mathscr{L}\boldsymbol{V}^h|_\tau^2 / |\nabla \boldsymbol{V}^h|_{\tilde{A}_0}^2, 0) \,. \tag{3.75}$$

All the problems were solved on a CONVEX-C1 in double precision (64 bits per floating point) using bilinear quadrilaterals, an implicit solver and the 2×2 Gaussian quadrature rule.

3.5.1. One-dimensional steady shock problem

This problem consists of two flow regions separated by a shock. If the flow properties on the two sides of the shock satisfy the normal shock conditions, the shock will be stationary. The following initial conditions satisfy these conditions:

$$\text{for } x < 0: \quad M = 2 \,, \quad \rho = 1 \,, \quad u = 1 \,, \quad p = 0.17857 \,; \tag{3.76}$$

$$\text{for } x > 0: \quad M = 0.57735 \,, \quad \rho = 2.66667 \,, \quad u = 0.37500 \,, \quad p = 0.80357 \tag{3.77}$$

($\gamma = 1.4$ and $c_v = 716.5$). The boundary conditions were set to the above data, and the vertical velocity was set to zero on the entire domain. The mesh consisted of 39×1 square elements, covering the domain $-19.5 \leq x \leq 19.5$ and $-0.5 \leq y \leq 0.5$.

The computed density using the Galerkin/least-squares method is presented in Fig. 3.5.1. This solution shows that the initial position of the shock is maintained by the method, which indicates that the method is indeed a flux-conservative method. The solution has small undershoots and overshoots near the shock. These oscillations are, however, very localized and do not corrupt the solution a small distance away from the shock.

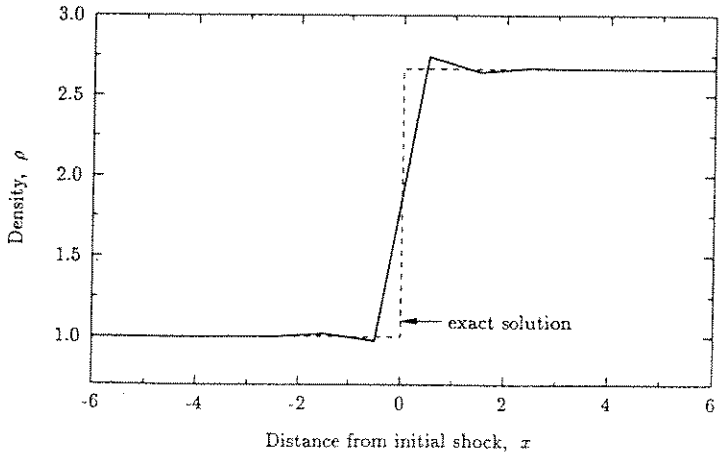


Fig. 3.5.1. One-dimensional steady shock problem ($M = 2$). Galerkin/least-squares method.

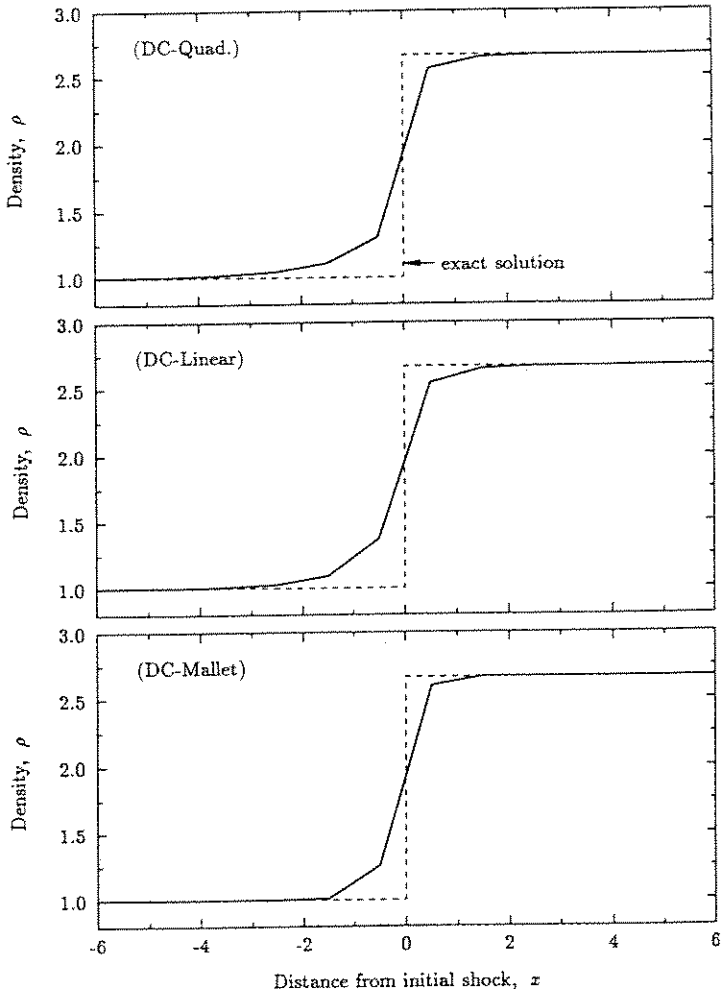


Fig. 3.5.2. One-dimensional steady shock problem ($M = 2$). Galerkin/least-squares with discontinuity-capturing operators.

Computed densities in the vicinity of the shock are compared in Fig. 3.5.2 for the three methods with discontinuity-capturing operators. This figure shows that all three discontinuity-capturing operators are capable of controlling the overshoots and undershoots. Note that for this problem, DC-Mallet results in the sharpest shock.

3.5.2. Oblique shock problem

This two-dimensional steady problem consists of a Mach two flow over a wedge at an angle of 10°, resulting in the occurrence of an oblique shock with an angle of 29.3° emanating from the leading edge of the wedge; see the schematics in Fig. 3.5.3.

Prescribing the following flow data at the inflow (i.e., on the left and top sides of the shock):

$$\text{Inflow:} \quad M = 2\,, \quad \rho = 1\,, \quad u_1 = \cos 10^\circ\,, \quad u_2 = -\sin 10^\circ\,, \quad p = 0.17857\,, \tag{3.78}$$

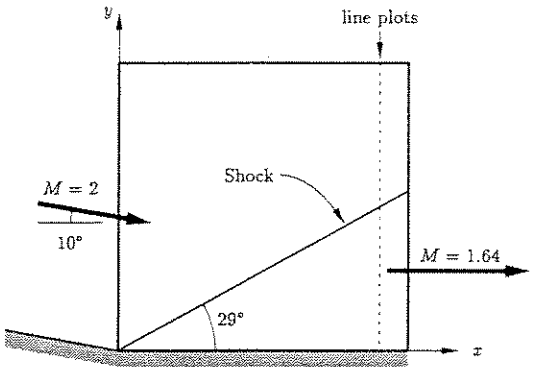


Fig. 3.5.3. Oblique shock problem.

results in the exact solution with the following flow data past the shock:

$$\text{Outflow:} \quad M = 1.64052\,, \quad \rho = 1.45843\,, \quad u_1 = 0.88731\,, \quad u_2 = 0\,, \quad p = 0.30475\,. \tag{3.79}$$

In the computations, four inflow conditions were imposed on the left and top boundaries; the slip condition (i.e., $u_2 = 0$) was imposed on the wedge; and no boundary conditions were imposed on the outflow (right) boundary. A 20×20 mesh of square elements, covering the domain $0 \leq x \leq 1$ and $0 \leq y \leq 1$, was employed. To avoid ambiguity in the boundary conditions, the node on the wedge at the inflow boundary was moved to $x = 0$ and $y = 0.025$. (This resulted in a slightly non-square element at that corner.)

For the three methods with discontinuity-capturing operators, the computed densities along $x = 0.9$ are plotted in Fig. 3.5.4. These results demonstrate the ability of these methods to properly determine the position of the shock, which is oblique to the mesh, and to satisfy the Rankine–Hugoniot jump conditions. Note that for this problem, DC-Linear spreads the shock slightly more than the other two methods.

3.5.3. Shock-reflection problem

This two-dimensioned steady problem consists of three flow regions separated by an oblique shock and its reflection from a wall; see Fig. 3.5.5. Prescribing the following Mach 2.9 flow data at the inflow (the first region on the left):

$$\text{Region 1:} \quad M = 2.9\,, \quad \rho = 1\,, \quad u_1 = 2.9\,, \quad u_2 = 0\,, \quad p = 0.71429\,, \tag{3.80}$$

and requiring that the incident shock be at an angle of 29°, leads to the following exact solution:

$$\text{Region 2:} \quad M = 2.3781\,, \quad \rho = 1.7\,, \quad u_1 = 2.61934\,, \quad u_2 = -0.50632\,, \quad p = 1.52819\,; \tag{3.81}$$

$$\text{Region 3:} \quad M = 1.94235\,, \quad \rho = 2.68728\,, \quad u_1 = 2.40140\,, \quad u_2 = 0\,, \quad p = 2.93407\,. \tag{3.82}$$

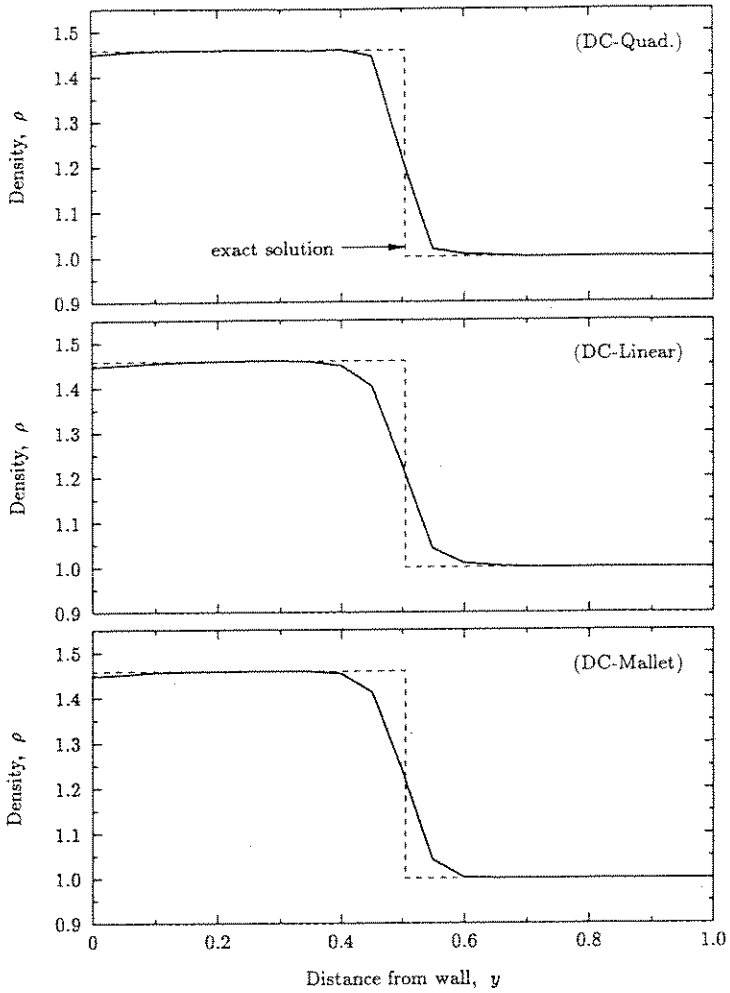


Fig. 3.5.4. Oblique shock problem. Galerkin/least-squares with discontinuity-capturing operators.

In the computations, four flow conditions were imposed on the left and top boundaries; the slip condition was imposed on the wall; and no boundary conditions were imposed on the outflow (right) boundary. The mesh consisted of 60×20 rectangular elements covering the domain $0 \leqslant x \leqslant 4.1$ and $0 \leqslant y \leqslant 1$.

For the three methods with discontinuity-capturing operators, the computed densities along $y = 0.25$ are plotted in Fig. 3.5.6. As may be seen, DC-Quad. results in the sharpest shock.

3.6. Mathematical properties

In this section we prove that our method is unconditionally stable, as measured in its natural entropy norm. Moreover, we provide some necessary results towards a nonlinear convergence proof.

In a series of studies, Johnson and Szepessy [23, 29, 30] proved that for nonlinear hyperbolic conservation laws equipped with a convex entropy function, such as for the Euler

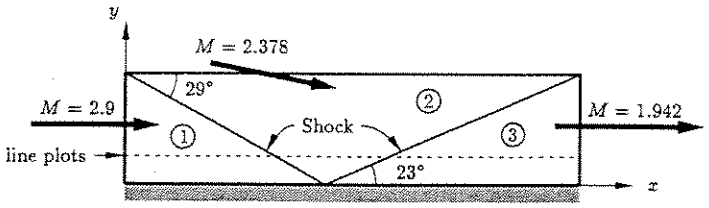


Fig. 3.5.5. Shock-reflection problem.

equations, the solution of the Galerkin/least-squares method (equivalent to SUPG in this case) converges to the entropy solution of the conservation laws *assuming* the finite element solutions are uniformly bounded. Johnson et al. [12] extended these results to Galerkin/least-squares method with an additional shock-capturing term. They also proved that for Burger's equation with triangular elements the finite element solution is uniformly bounded. Szepessy [13] extended the proofs to scalar nonlinear hyperbolic conservation laws in two space-

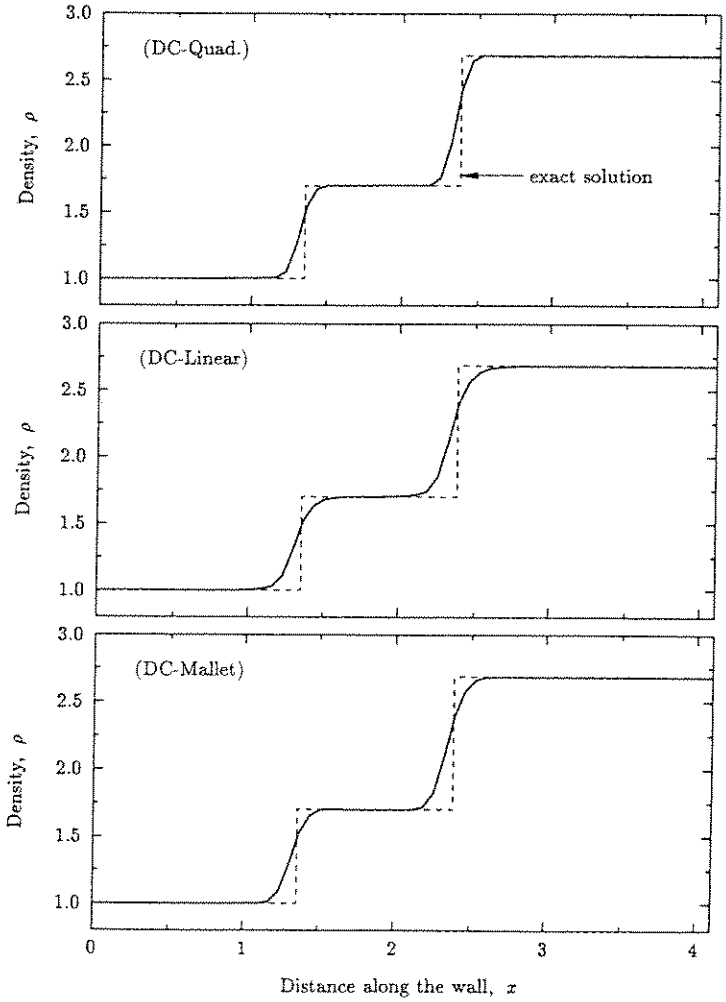


Fig. 3.5.6. Shock-reflection problem. Galerkin/least-squares with discontinuity-capturing operators.

dimensions with linear elements. Extension of these results to the Navier–Stokes equations is by no means an easy task. In this section, however, we consider some of the ingredients necessary for such an extension. (In addition, one can establish linear error estimates for the Galerkin/least-squares method by reproducing the analysis of Hughes et al. [11] with the results of this section.) For error analysis of methods with discontinuity-capturing operators for linear problems we refer to [13, 28].

3.6.1. Stability proof

We need to establish the following lemmas to prove stability:

LEMMA 3.1 (Properties of V). Defining $H = \rho(s_0 - s)$ *and* $\boldsymbol{V} = H_{,U}^{\mathrm{t}}$, *then*

$$\boldsymbol{V} \cdot \boldsymbol{U}_{,t} = H_{,t} , \tag{3.83}$$

$$\boldsymbol{V} \cdot \boldsymbol{F}_{i,i} = (u_i H)_{,i} , \tag{3.84}$$

$$\boldsymbol{V} \cdot \boldsymbol{F}_i^{\mathrm{d}} = q_i / c_v \theta , \tag{3.85}$$

$$\boldsymbol{V} \cdot \mathscr{F} = \rho r / c_v \theta , \tag{3.86}$$

$$\boldsymbol{V} \cdot \tilde{\boldsymbol{A}}_0 \boldsymbol{V} = \rho(1 + (s_0 - s + \bar{\gamma})^2 / \bar{\gamma}) , \tag{3.87}$$

where $\bar{\gamma} = \gamma - 1$.

PROOF. These results follow directly from the definition and structure of the above matrices; see Section 2 and Appendix A. □

LEMMA 3.2. H is a strictly convex function of U. Furthermore, there exists a positive, nondimensional constant $\hat{c}$ *such that*

$$H(t_n^-) - H(t_n^+) + \boldsymbol{V}(t_n^+) \cdot [\![\boldsymbol{U}(t_n)]\!] \geqslant \hat{c} \| [\![\boldsymbol{U}(t_n)]\!] \|_{\tilde{\boldsymbol{A}}_0^{-1}}^2 , \tag{3.88}$$

where $[\![\boldsymbol{U}(t_n)]\!] = \boldsymbol{U}(t_n^+) - \boldsymbol{U}(t_n^-)$.

PROOF. The first assertion has been established in [19]. By strict convexity,

$$\tilde{\boldsymbol{A}}_0^{-1} = \partial^2 H / \partial \boldsymbol{U}^2 \tag{3.89}$$

is positive-definite. By Taylor's formula (see, e.g., [31]):

$$\begin{aligned} &H(t_n^-) - H(t_n^+) + \boldsymbol{V}(t_n^+) \cdot [\![\boldsymbol{U}(t_n)]\!] \\ &\quad = \int_0^1 (1 - \varepsilon) [\![\boldsymbol{U}(t_n)]\!] \cdot \tilde{\boldsymbol{A}}_0^{-1} (\boldsymbol{U}(t_n^+) - \varepsilon [\![\boldsymbol{U}(t_n)]\!]) [\![\boldsymbol{U}(t_n)]\!] \, \mathrm{d}\varepsilon \geqslant \hat{c} \| [\![\boldsymbol{U}(t_n)]\!] \|_{\tilde{\boldsymbol{A}}_0^{-1}}^2 . \quad \square \end{aligned} \tag{3.90}$$

THEOREM 3.1 (Stability proof). The variational equation (3.8) is entropy stable. That is, for $N = 1, 2, \ldots,$

$$\int_\Omega H(t_N^-)\,\mathrm{d}\Omega + \sum_{n=0}^{N-1} \|\nabla V^h\|^2_{\tilde{K},Q_n} + \sum_{n=0}^{N-1} \int_{Q_n} \frac{\rho r}{c_v \theta}\,\mathrm{d}Q + \hat{c} \sum_{n=0}^{N-1} \|[\![U^h(t_n)]\!]\|^2_{\tilde{A}_0^{-1},\Omega}$$
$$+ \sum_{n=0}^{N-1} \sum_{e=1}^{(n_{el})_n} \|\mathscr{L} V^h\|^2_{\tau,Q_n^e} + \sum_{n=0}^{N-1} \sum_{e=1}^{(n_{el})_n} \|\tilde{\nabla}_\xi V^h\|^2_{\nu^h \tilde{A}_0, Q_n^e}$$
$$\leqslant \int_\Omega H(t_0^-)\,\mathrm{d}\Omega + \sum_{n=0}^{N-1} \int_{P_n} \left(-u_i H + \frac{q_i}{c_v \theta}\right) n_i\,\mathrm{d}P\,, \tag{3.91}$$

where the L_2 *norm with respect to metric* $\boldsymbol{M}$ *in domain* S *is defined as*

$$\|W\|^2_{M,S} = \int_S \boldsymbol{W} \cdot \boldsymbol{M} \boldsymbol{W}\,\mathrm{d}S\,. \tag{3.92}$$

PROOF. This result is obtained by substituting V^h for W^h in the variational equation (3.8), summing over $n = 0, \ldots, N-1$, and using Lemmas 3.1 and 3.2. □

The above stability result shows that $\int_\Omega H\,\mathrm{d}\Omega$ at the end of every space–time slab is bounded by the initial data, provided that the boundary integral on the right-hand-side of (3.91) is negative. This condition can be used to derive a set of well-posed boundary conditions; see [21].

PROPOSITION 3.1 (Norm equivalence). Assuming that $\rho > 0$ *and* $\bar{\gamma} \leqslant s_0 - s \leqslant c\bar{\gamma}$, *where* c *is a positive constant, there exist positive constants* c_1 *and* c_2 *such that the following pointwise entropy norms are equivalent:*

$$c_1 H \leqslant V \cdot \tilde{A}_0 V \leqslant c_2 H\,. \tag{3.93}$$

PROOF. Note that an s_0 can always be defined such that the first inequality in the second hypothesis is true. However, the second inequality in that hypothesis is contingent upon the boundedness of the solution.

From Lemma 3.1

$$V \cdot \tilde{A}_0 V = \rho(1 + (s_0 - s + \bar{\gamma})^2/\bar{\gamma})$$
$$= 2\rho(s_0 - s) + \gamma\rho + \rho(s_0 - s)^2/\bar{\gamma} \geqslant 2\rho(s_0 - s) = 2H\,. \tag{3.94}$$

By setting $c_1 \leqslant 2$, the first inequality is obtained. Also

$$V \cdot \tilde{A}_0 V = \rho(s_0 - s)\left(2 + \frac{\gamma}{(s_0 - s)} + \frac{(s_0 - s)}{\bar{\gamma}}\right) \leqslant \rho(s_0 - s)\left(2 + \frac{\gamma}{\bar{\gamma}} + c\right). \tag{3.95}$$

By setting $c_2 \geqslant (2 + \gamma/\bar{\gamma} + c)$, the second inequality is obtained. □

3.6.2. Properties of $\boldsymbol{\tau}$

Let h and Δt denote the spatial and temporal mesh parameters defined as

$$h = \max_{n=0,\ldots,N-1} \left(\max_{e=1,\ldots,(n_{\mathrm{el}})_n} (h_n^e) \right), \tag{3.96}$$

$$\Delta t = \max_{n=0,\ldots,N-1} (t_{n+1} - t_n), \tag{3.97}$$

where h_n^e is the diameter of the smallest sphere containing Ω_n^e. Consider the eigenvalue problems: For $i = 1, \ldots, m$,

$$\mathbf{0} = (\boldsymbol{C} - \tilde{\lambda}_i \mathbf{I})\boldsymbol{\Phi}_i = (\tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}_0^{-1} - \tilde{\lambda}_i \mathbf{I})\boldsymbol{\Phi}_i = (\tilde{\boldsymbol{C}} - \tilde{\lambda}_i \tilde{\boldsymbol{A}}_0)\tilde{\boldsymbol{A}}_0^{-1}\boldsymbol{\Phi}_i ; \tag{3.98}$$

$$\mathbf{0} = (\boldsymbol{A}_j\boldsymbol{A}_j - \bar{\lambda}_i^2 \mathbf{I})\boldsymbol{\varphi}_i = (\tilde{\boldsymbol{A}}_j\tilde{\boldsymbol{A}}_0^{-1}\tilde{\boldsymbol{A}}_j - \bar{\lambda}_i^2 \tilde{\boldsymbol{A}}_0)\tilde{\boldsymbol{A}}_0^{-1}\boldsymbol{\varphi}_i ; \tag{3.99}$$

$$\mathbf{0} = (\boldsymbol{K}_{jk}\boldsymbol{K}_{kj} - \hat{\lambda}_i^2 \mathbf{I})\boldsymbol{\Psi}_i = (\tilde{\boldsymbol{K}}_{jk}\tilde{\boldsymbol{A}}_0^{-1}\tilde{\boldsymbol{K}}_{kj} - \hat{\lambda}_i^2 \tilde{\boldsymbol{A}}_0)\tilde{\boldsymbol{A}}_0^{-1}\boldsymbol{\Psi}_i . \tag{3.100}$$

Using the structure of these matrices, the eigenvectors can be scaled with respect to $\tilde{\boldsymbol{A}}_0^{-1}$. For example,

$$[\boldsymbol{\Phi}_1 \ldots \boldsymbol{\Phi}_m]^{\mathrm{t}}\tilde{\boldsymbol{A}}_0^{-1}[\boldsymbol{\Phi}_1 \ldots \boldsymbol{\Phi}_m] = \mathbf{I} . \tag{3.101}$$

Define the maximum eigenvalue of $\boldsymbol{C}$, $\boldsymbol{A}_i\boldsymbol{A}_i$ and $\boldsymbol{K}_{ij}\boldsymbol{K}_{ji}$ as

$$\alpha_C = \max_{i=1,\ldots,m} (\tilde{\lambda}_i) , \tag{3.102}$$

$$\alpha_A = \max_{i=1,\ldots,m} (\bar{\lambda}_i) , \tag{3.103}$$

$$\alpha_K = \max_{i=1,\ldots,m} (\hat{\lambda}_i) . \tag{3.104}$$

For the compressible Navier–Stokes equations in three dimensions:

$$\alpha_C = r/2\iota + (b_i b_i + (r/2\iota)^2)^{1/2} , \tag{3.105}$$

$$\alpha_A = (u^2 + 2c^2 + c\sqrt{4u^2 + c^2})^{1/2} , \tag{3.106}$$

$$\alpha_K = \max(((8\mu^2 + 4\mu\lambda + 3\lambda^2)/\rho^2)^{1/2}, \sqrt{3}\kappa/c_v\rho) , \tag{3.107}$$

where in (3.106) $u = |\boldsymbol{u}|$ and $c = \sqrt{\gamma p/\rho}$ are the particle and acoustic speeds.

PROPOSITION 3.2. Given $\boldsymbol{W} \in \mathcal{R}^m$, *there exists a positive constant* c *such that*

$$\boldsymbol{W} \cdot \boldsymbol{\tau}^{-1}\boldsymbol{W} \leq c(\alpha_C + \Delta t^{-1} + \alpha_A h^{-1} + \alpha_K h^{-2})\boldsymbol{W} \cdot \tilde{\boldsymbol{A}}_0\boldsymbol{W} . \tag{3.108}$$

PROOF. This result follows directly from the definition of τ in (3.35). □

PROPOSITION 3.3. Given $\boldsymbol{W}$, *there exists a positive constant* c *such that*

$$\mathscr{L}\boldsymbol{W}\cdot\tau\mathscr{L}\boldsymbol{W} \leqslant c\Big\{\alpha_C \boldsymbol{W}\cdot\tilde{\boldsymbol{A}}_0\boldsymbol{W} + \Delta t\, \boldsymbol{W}_{,t}\cdot\tilde{\boldsymbol{A}}_0\boldsymbol{W}_{,t} + \alpha_A h\nabla\boldsymbol{W}\cdot\begin{bmatrix}\nwarrow & \tilde{\boldsymbol{A}}_0 & \nearrow\end{bmatrix}\nabla\boldsymbol{W}$$
$$+ h^2(\nabla\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W})\cdot\tilde{\boldsymbol{A}}_0^{-1}(\boldsymbol{K}_{ij}\boldsymbol{K}_{ji})^{-1/2}(\nabla\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W})\Big\}. \tag{3.109}$$

PROOF. Using the triangle inequality and (3.10), we have

$$\mathscr{L}\boldsymbol{W}\cdot\tau\mathscr{L}\boldsymbol{W} \leqslant c\{\boldsymbol{W}\cdot\tilde{\boldsymbol{C}}\boldsymbol{\tau}\tilde{\boldsymbol{C}}\boldsymbol{W} + \boldsymbol{W}_{,t}\cdot\tilde{\boldsymbol{A}}_0\boldsymbol{\tau}\tilde{\boldsymbol{A}}_0\boldsymbol{W}_{,t} + \nabla\boldsymbol{W}\cdot\tilde{\boldsymbol{A}}\boldsymbol{\tau}\tilde{\boldsymbol{A}}^{\mathrm{t}}\nabla\boldsymbol{W}$$
$$+ (\nabla\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W})\cdot\boldsymbol{\tau}(\nabla\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W})\}. \tag{3.110}$$

Substituting for $\boldsymbol{\tau}$ from (3.35) into the first term on the right-hand-side of the above inequality leads to

$$\boldsymbol{W}\cdot\tilde{\boldsymbol{C}}\boldsymbol{\tau}\tilde{\boldsymbol{C}}\boldsymbol{W} \leqslant \boldsymbol{W}\cdot\tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}_0^{-1}\left(\boldsymbol{C}^2 + \left(\frac{2}{\Delta t}\right)^2\mathbf{I}_m + \left(\frac{2}{h}\right)^2\boldsymbol{A}_i\boldsymbol{A}_i + \left(\frac{2}{h}\right)^4\boldsymbol{K}_{ij}\boldsymbol{K}_{ji}\right)^{-1/2}\tilde{\boldsymbol{C}}\boldsymbol{W}$$
$$\leqslant \boldsymbol{W}\cdot\tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}_0^{-1}(\boldsymbol{C}^2)^{-1/2}\tilde{\boldsymbol{C}}\boldsymbol{W} \leqslant \boldsymbol{W}\cdot\tilde{\boldsymbol{C}}\boldsymbol{W} \leqslant \alpha_C\boldsymbol{W}\cdot\tilde{\boldsymbol{A}}_0\boldsymbol{W}. \tag{3.111}$$

The remaining terms follow analogously. □

REMARK 3.1. For linear error analysis $\tilde{\boldsymbol{K}}$ is assumed to be constant, hence, the last term in (3.110) simplifies to

$$(\nabla\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W})\cdot\boldsymbol{\tau}(\nabla\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}) \leqslant \alpha_K h^2\nabla^2\boldsymbol{W}\cdot\begin{bmatrix}\nwarrow & \tilde{\boldsymbol{A}}_0 & \nearrow\end{bmatrix}\nabla^2\boldsymbol{W}. \tag{3.112}$$

PROPOSITION 3.4. Given $\boldsymbol{W}^h \in \mathscr{V}_n^h$, *there exist a positive constant* c *such that*

$$\int_{Q_n^e}(\nabla\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}^h)\cdot\boldsymbol{\tau}(\nabla\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}^h)\,\mathrm{d}Q \leqslant c\int_{Q_n^e}\nabla\boldsymbol{W}^h\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}^h\,\mathrm{d}Q. \tag{3.113}$$

PROOF. We need the following inverse estimate: For $i = 1, \ldots, m$,

$$\int_{Q_n^e}\boldsymbol{Y}_{\underline{i}}^{\mathrm{t}}(\tilde{\boldsymbol{K}}_{jk}\boldsymbol{W}_{,k}^h)_{,j}\boldsymbol{Y}_{\underline{i}}^{\mathrm{t}}(\tilde{\boldsymbol{K}}_{pq}\boldsymbol{W}_{,q}^h)_{,p}\,\mathrm{d}Q \leqslant \bar{c}h^{-2}\int_{Q_n^e}(\boldsymbol{Y}_{\underline{i}}^{\mathrm{t}}\tilde{\boldsymbol{K}}_{jk}\boldsymbol{W}_{,k}^h)(\boldsymbol{Y}_{\underline{i}}^{\mathrm{t}}\tilde{\boldsymbol{K}}_{jq}\boldsymbol{W}_{,q}^h)\,\mathrm{d}Q, \tag{3.114}$$

where $\boldsymbol{Y}_i = \hat{\lambda}_{\underline{i}}^{-1/2}\tilde{\boldsymbol{A}}_0^{-1}\boldsymbol{\Psi}_{\underline{i}}$ and $\hat{\lambda}_i$ and $\boldsymbol{\Psi}_i$ are defined in (3.100). In addition, define for $l = 1, \ldots, d$,

$$\upsilon_i^{(l)} = \begin{Bmatrix}\delta_{1l}\boldsymbol{Y}_i \\ \vdots \\ \delta_{dl}\boldsymbol{Y}_i\end{Bmatrix}, \quad (d\cdot m)\times 1. \tag{3.115}$$

Then, for $i = 1, \ldots, m$,

$$(\underline{Y}_i^{\mathrm{t}}\tilde{\boldsymbol{K}}_{jk}\boldsymbol{W}_{,k}^h)(\underline{Y}_i^{\mathrm{t}}\tilde{\boldsymbol{K}}_{jq}\boldsymbol{W}_{,q}^h) = \nabla\boldsymbol{W}^h\cdot\tilde{\boldsymbol{K}}\Big[\overleftarrow{\ }(\underline{Y}_i\underline{Y}_i^{\mathrm{t}})\overrightarrow{\ }\Big]\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}^h$$
$$= \sum_{l=1}^{d}(\nabla\boldsymbol{W}^h\cdot\tilde{\boldsymbol{K}}\boldsymbol{v}_i^{(l)})(\boldsymbol{v}_i^{(l)}\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}^h) \leqslant \sum_{l=1}^{d}(\boldsymbol{v}_i^{(l)}\cdot\tilde{\boldsymbol{K}}\boldsymbol{v}_i^{(l)})(\nabla\boldsymbol{W}^h\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}^h)$$
$$= (\underline{Y}_i\cdot\tilde{\boldsymbol{K}}_{jj}\underline{Y}_i)(\nabla\boldsymbol{W}^h\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}^h) \leqslant (\nabla\boldsymbol{W}^h\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}^h)\,. \tag{3.116}$$

The last step follows directly from the definition of $\hat{\lambda}_i$ in (3.100). Proceeding as in (3.111) and using (3.35), (3.100), (3.114) and (3.116), we have

$$\int_{Q_n^e}(\nabla\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}^h)\cdot\boldsymbol{\tau}(\nabla\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}^h)\,\mathrm{d}Q$$
$$\leqslant \tfrac{1}{4}h^2\int_{Q_n^e}(\tilde{\boldsymbol{K}}_{jk}\boldsymbol{W}_{,k}^h)_{,j}\tilde{\boldsymbol{A}}_0^{-1}(\boldsymbol{K}_{rs}\boldsymbol{K}_{sr})^{-1/2}(\tilde{\boldsymbol{K}}_{pq}\boldsymbol{W}_{,q}^h)_{,p}\,\mathrm{d}Q$$
$$= \tfrac{1}{4}h^2\sum_{i=1}^{m}\int_{Q_n^e}\underline{Y}_i^{\mathrm{t}}(\tilde{\boldsymbol{K}}_{jk}\boldsymbol{W}_{,k}^h)_{,j}\underline{Y}_i^{\mathrm{t}}(\tilde{\boldsymbol{K}}_{pq}\boldsymbol{W}_{,q}^h)_{,p}\,\mathrm{d}Q$$
$$\leqslant \tfrac{1}{4}\bar{c}m\int_{Q_n^e}\nabla\boldsymbol{W}^h\cdot\tilde{\boldsymbol{K}}\nabla\boldsymbol{W}^h\,\mathrm{d}Q\,. \tag{3.117}$$

Choosing $c = \bar{c}m/4$ completes the proof. □

Propositions 3.2–3.4 are essential ingredients for establishing linear error estimates of the kind presented in [11, 26] and nonlinear convergence proofs as in [12]. (See also [5] for an application of Proposition 3.4.)

4. Predictor multi-corrector algorithms

In this section, we present two finite element discretizations of the space–time Galerkin/least-squares variational equation introduced in Section 3. Each discretization leads to a system of nonlinear algebraic equations at every time step. For each case, we develop a class of predictor multi-corrector algorithms to reduce the resulting nonlinear system to a sequence of linear systems, the solution of which is discussed in [15].

In the first discretization, we use a finite element space consisting of functions that are piecewise linear in space and constant in time. This leads to a time-marching method which has a low order of accuracy in time, but has good stability properties and is computationally efficient, thus, rendering it an attractive method for solving *steady* problems.

In the second discretization, we use a finite element space consisting of functions that are piecewise linear in time as well as in space. This method exhibits a higher order of temporal accuracy than the constant-in-time method. However, it is computationally more expensive. This method is attractive for solving *transient* problems.

4.1. Constant-in-time approximation

Experience has shown that, in many situations, a good strategy for obtaining the solution of a steady problem is to solve it as a time dependent problem. Starting from an initial guess, the solution is marched in time until it converges to a steady state. For nonlinear problems, such as compressible flow problems, this strategy has been observed to be superior to those which solve the steady problem directly, using a Newton-type method. Since the objective is to obtain the solution at steady state, it is desirable to employ a transient algorithm which has good stability properties, is computationally efficient, and is accurate in space at steady state, even if it is not spatially accurate on the way to steady state. To obtain these objectives, in this section we develop a predictor multi-corrector algorithm based on the constant-in-time approximation of the finite element spaces. This algorithm is the nonlinear version of the first-order predictor multi-corrector algorithm of Shakib and Hughes [14], applied to the compressible Navier–Stokes equations.

4.1.1. Finite element discretization

The finite element spaces are assumed to be constant in time within each space–time slab and discontinuous across the space–time slab interfaces. Hence, within the nth space–time slab, the finite element trial solution and the weighting function are defined as

$$\boldsymbol{V}^h(\boldsymbol{x}, t) = \sum_{A=1}^{(n_{\mathrm{np}})_{(n)}} N_A^{(n)}(\boldsymbol{x})\boldsymbol{v}_{A;(n+1)} \quad \text{for } \boldsymbol{x}\in\Omega\,,\ t\in I_n\,, \tag{4.1}$$

$$\boldsymbol{W}^h(\boldsymbol{x}, t) = \sum_{A=1}^{(n_{\mathrm{np}})_{(n)}} N_A^{(n)}(\boldsymbol{x})\boldsymbol{w}_{A;(n+1)} \quad \text{for } \boldsymbol{x}\in\Omega\,,\ t\in I_n\,, \tag{4.2}$$

where $\boldsymbol{v}_{A;(n+1)}$ is the $m\times 1$ vector of nodal unknowns at node A for the nth space–time slab; $\boldsymbol{w}_{A;(n+1)}$ is the nodal values of the weighting function corresponding to $\boldsymbol{v}_{A;(n+1)}$; $(n_{\mathrm{np}})_n$ is the number of nodal points for the nth space–time slab; and $N_A^{(n)}(\boldsymbol{x})$ is the finite element spatial shape-function of node A at the nth space–time slab. To simplify the notation of the following presentation, the superscripts and subscripts associated with the nth space–time slab are dropped.

Substituting (4.1) and (4.2) into the space–time Galerkin/least-squares variational equation (3.8) leads to

$$\begin{aligned}
&\sum_{A=1}^{n_{\mathrm{np}}} \boldsymbol{w}_A \cdot \Bigg\{ \int_{Q_n} \Bigg(-N_{A,i}\boldsymbol{F}_i\Bigg(\sum_{B=1}^{n_{\mathrm{np}}} N_B\boldsymbol{v}_B\Bigg) + N_{A,i}\tilde{\boldsymbol{K}}_{ij}\sum_{B=1}^{n_{\mathrm{np}}} N_{B,j}\boldsymbol{v}_B + N_A\tilde{C}\sum_{B=1}^{n_{\mathrm{np}}} N_B\boldsymbol{v}_B\Bigg)\,\mathrm{d}Q \\
&\quad + \int_\Omega \Bigg(N_A U\Bigg(\sum_{B=1}^{n_{\mathrm{np}}} N_B\boldsymbol{v}_B\Bigg) - N_A U\Bigg(\sum_{B=1}^{(n_{\mathrm{np}})_{(n-1)}} N_B^{(n-1)}\boldsymbol{v}_{B;(n)}\Bigg)\Bigg)\,\mathrm{d}\Omega \\
&\quad + \sum_{e=1}^{n_{\mathrm{el}}} \int_{Q_n^e} \mathscr{L}_A^h\cdot\boldsymbol{\tau}\sum_{B=1}^{n_{\mathrm{np}}} \mathscr{L}_B^h\boldsymbol{v}_B\,\mathrm{d}Q \\
&\quad + \sum_{e=1}^{n_{\mathrm{el}}} \int_{Q_n^e} 2\,\frac{\left|\sum_{C=1}^{n_{\mathrm{np}}} \mathscr{L}_C^h\boldsymbol{v}_C\right|_\tau^2}{\left|\sum_{D=1}^{n_{\mathrm{np}}} \bar{\boldsymbol{\nabla}}_\xi N_D\boldsymbol{v}_D\right|_\tau^2}\,(\bar{\boldsymbol{\nabla}}_\xi N_A)^{\mathrm{t}}\cdot\Bigg[\ \overleftarrow{\tilde{\boldsymbol{A}}_0}\ \Bigg]\sum_{B=1}^{n_{\mathrm{np}}} \bar{\boldsymbol{\nabla}}_\xi N_B\boldsymbol{v}_B\,\mathrm{d}Q
\end{aligned}$$

$$+\int_{P_n} N_A\left(F_i\left(\sum_{B=1}^{n_{np}} N_B \boldsymbol{v}_B\right) - F_i^d\left(\sum_{B=1}^{n_{np}} N_B \boldsymbol{v}_B\right)\right) n_i \, dP\Bigg\}$$

$$= 0\,, \tag{4.3}$$

where $\mathscr{L}_A^h$ is the discrete counterpart of the differential operator, (3.10), acting on the contributions of node A; this operator is defined as

$$\mathscr{L}_A^h = \tilde{A}_i N_{A,i} - (\tilde{K}_{ij} N_{A,j})_{,i} + \tilde{C} N_A\,. \tag{4.4}$$

Define

$$\boldsymbol{v} = \{\boldsymbol{v}_1^t, \boldsymbol{v}_2^t, \ldots, \boldsymbol{v}_{n_{np}}^t\}^t\,, \tag{4.5}$$

$$\boldsymbol{w} = \{\boldsymbol{w}_1^t, \boldsymbol{w}_2^t, \ldots, \boldsymbol{w}_{n_{np}}^t\}^t\,, \tag{4.6}$$

$$\boldsymbol{v}_{(n)} = \{\boldsymbol{v}_{1;(n)}^t, \boldsymbol{v}_{2;(n)}^t, \ldots, \boldsymbol{v}_{(n_{np})_{(n-1)};(n)}^t\}^t\,. \tag{4.7}$$

Then (4.3) can be written as

$$\boldsymbol{w} \cdot \boldsymbol{G}(\boldsymbol{v}; \boldsymbol{v}_{(n)}) = 0\,, \tag{4.8}$$

where $\boldsymbol{G}(\boldsymbol{v}; \boldsymbol{v}_{(n)})$ is an $(n_{np} \cdot m) \times 1$ system of nonlinear algebraic equations with $\boldsymbol{v}$ as its $(n_{np} \cdot m) \times 1$ vector of unknowns and $\boldsymbol{v}_{(n)}$ as its $((n_{np})_{(n-1)} \cdot m) \times 1$ vector of initial conditions.

Equation (4.8) holds for all unconstrained $\boldsymbol{w}$. Assuming there are no essential boundary conditions (the case with essential boundary conditions is analyzed in Section 5), (4.8) leads to the *nonlinear algebraic system*

$$\boldsymbol{G}(\boldsymbol{v}; \boldsymbol{v}_{(n)}) = \boldsymbol{0}\,, \tag{4.9}$$

where there are $n_{np} \times m$ equations and $n_{np} \times m$ unknowns.

It is important to note that the above nonlinear system is unconditionally stable, provided it is solved exactly. This is a direct consequence of the stability result presented in Section 3.6.1. In practice, we do not solve $\boldsymbol{G}$ exactly, because the cost associated with exact solution is excessive. Thus, the method is not unconditionally stable. The stability of the algorithm is further discussed in Section 4.3.

Since the finite element functions are discontinuous at the space–time slab interfaces, the spatial discretization can be changed from one space–time slab to the next. This provides a natural mechanism for incorporating adaptive meshing in the formulation. Moreover, this change of meshing does not degrade the order of accuracy of the method, provided the second term in the jump-condition integral (i.e., the second integral in (4.3)) is computed accurately. In the current implementation of the method we have assumed that the spatial discretization is kept constant from one space–time slab to the next. That is, we assume $\Omega_{n+1}^e = \Omega_n^e = \Omega^e$.

We shall restrict the spatial discretization to linear or multilinear interpolations (i.e., in two dimensions linear triangles or bilinear quadrilaterals). To avoid the evaluation of the second

derivatives we assume

$$\mathscr{L}_A^h = \tilde{A}_i N_{A,i} + \tilde{C} N_A \tag{4.10}$$

and

$$\tilde{\nabla}_\xi N_A = \{N_A, N_{A,\xi_1}, \ldots, N_{A,\xi_d}\}^t . \tag{4.11}$$

To justify this omission of the diffusive terms, consider a mixed finite element method where the diffusive flux vector is assumed to be a dependent variable. Let $\hat{F}_i^d$ denote the diffusive flux vector and assume it is constant within each element. That is,

$$\hat{F}_i^d(x) = \sum_{e=1}^{n_{el}} \hat{N}_e \hat{F}_{i;(e)}^d \quad \text{for } x \in \Omega\ ,\ t \in I_n\ , \tag{4.12}$$

where $\hat{F}_{i;(e)}^d$ is the diffusive flux vector of the eth element; $\hat{N}_e$ is a piecewise constant shape function. $\hat{F}_i^d$ is evaluated using

$$\int_{Q_n^e} \delta\hat{F}_i^d \cdot (\hat{F}_i^d - \tilde{K}_{ij} V_{,j}^h)\, dQ = 0\ , \tag{4.13}$$

where $\delta\hat{F}_i^d$ is the corresponding weighting function. On element interiors, since $\hat{F}_i^d$ is piecewise constant,

$$\begin{aligned}\sum_{A=1}^{n_{np}} \mathscr{L}_A^h \boldsymbol{v}_A &= \sum_{A=1}^{n_{np}} \tilde{A}_i N_{A,i} \boldsymbol{v}_A - \sum_{e=1}^{n_{el}} \hat{N}_{e,i} \hat{F}_{i;(e)}^d + \sum_{A=1}^{n_{np}} \tilde{C} N_A \boldsymbol{v}_A \\ &= \sum_{A=1}^{n_{np}} \tilde{A}_i N_{A,i} \boldsymbol{v}_A + \sum_{A=1}^{n_{np}} \tilde{C} N_A \boldsymbol{v}_A \end{aligned} \tag{4.14}$$

or

$$\mathscr{L}_A^h = \tilde{A}_i N_{A,i} + \tilde{C} N_A\ . \tag{4.15}$$

Analogously, the second derivatives in $\tilde{\nabla}_\xi N_A$ are omitted, but $\tilde{K}_{ij} V_{,j}^h$ is retained in the Galerkin integral.

4.1.2. First-order predictor multi-corrector algorithm

An iterative solver is used to solve the nonlinear algebraic system, (4.9). In the predictor phase, the solution of the current time step is initialized to the final solution of the previous time step. Let $\boldsymbol{v}^{(i)}$ be the ith iterative approximation of $\boldsymbol{v}_{(n+1)}$, with $\boldsymbol{v}^{(0)} = \boldsymbol{v}_{(n)}$; then the Taylor series expansion of $\boldsymbol{G}$, retaining only the first two terms, yields

$$\boldsymbol{G}(\boldsymbol{v}^{(i+1)}; \boldsymbol{v}_{(n)}) \simeq \boldsymbol{G}(\boldsymbol{v}^{(i)}; \boldsymbol{v}_{(n)}) + \frac{\partial \boldsymbol{G}(\boldsymbol{v}^{(i)}; \boldsymbol{v}_{(n)})}{\partial \boldsymbol{v}} \Delta\boldsymbol{v}^{(i)} = \boldsymbol{0}\ , \tag{4.16}$$

where

$$\Delta\boldsymbol{v}^{(i)} \overset{\text{def}}{=} \boldsymbol{v}^{(i+1)} - \boldsymbol{v}^{(i)}\ . \tag{4.17}$$

Denote by

$$\boldsymbol{R}^{(i)} = \boldsymbol{G}(\boldsymbol{v}^{(i)}; \boldsymbol{v}_{(n)}) , \tag{4.18}$$

$$\boldsymbol{M}^{(i)} = \partial \boldsymbol{G}(\boldsymbol{v}^{(i)}; \boldsymbol{v}_{(n)}) / \partial \boldsymbol{v} , \tag{4.19}$$

the residual vector and the consistent tangent matrix, respectively. Let $\boldsymbol{V}^e(\boldsymbol{x})$ denote $\boldsymbol{V}^h(\boldsymbol{x})$ within the eth element at the ith iteration:

$$\boldsymbol{V}^e(\boldsymbol{x}) = \sum_{a=1}^{n_{\text{en}}} N_a^e(\boldsymbol{x}) \boldsymbol{v}_a^{(i)} \quad \text{for } \boldsymbol{x} \in \Omega^e , \tag{4.20}$$

where n_{en} is the number of element nodes (e.g., $n_{\text{en}} = 3$ for linear triangles and $n_{\text{en}} = 4$ for bilinear quadrilaterals); $N_a^e(\boldsymbol{x})$ is the shape function of node a for element e; and $\boldsymbol{v}_a^{(i)}$ is the localized value of $\boldsymbol{v}^{(i)}$. Similarly, let $\boldsymbol{V}_{(n)}^e$ denote $\boldsymbol{V}^h$ in the previous space–time slab. Then $\boldsymbol{R}^{(i)}$ can be written as

$$\boldsymbol{R}^{(i)} = \mathop{\mathsf{A}}_{e=1}^{n_{\text{el}}} \boldsymbol{R}^e , \tag{4.21}$$

$$\boldsymbol{R}^e = \{\boldsymbol{R}_a^e\} , \quad a = 1, \ldots, n_{\text{en}} , \tag{4.22}$$

$$\begin{aligned}
\boldsymbol{R}_a^e = & \int_{\Omega^e} (N_a^e \boldsymbol{U}(\boldsymbol{V}^e) - N_a^e \boldsymbol{U}(\boldsymbol{V}_{(n)}^e)) \, \mathrm{d}\Omega && \text{(Jump-condition)} \\
& + \Delta t \int_{\Omega^e} (-N_{a,i}^e \boldsymbol{F}_i(\boldsymbol{V}^e) + N_{a,i}^e \tilde{\boldsymbol{K}}_{ij} \boldsymbol{V}_{,j}^e + N_a^e \tilde{\boldsymbol{C}} \boldsymbol{V}^e) \, \mathrm{d}\Omega && \text{(Galerkin)} \\
& + \Delta t \int_{\Omega^e} (N_{a,i}^e \tilde{\boldsymbol{A}}_i + N_a^e \tilde{\boldsymbol{C}}) \cdot \boldsymbol{\tau} (\tilde{\boldsymbol{A}}_j \boldsymbol{V}_{,j}^e + \tilde{\boldsymbol{C}} \boldsymbol{V}^e) \, \mathrm{d}\Omega && \text{(Least-squares)} \\
& + \Delta t \int_{\Omega^e} \nu^e (N_{a,\xi_i} \tilde{\boldsymbol{A}}_0 \boldsymbol{V}_{,\xi_i}^e + N_a \tilde{\boldsymbol{A}}_0 \boldsymbol{V}^e) \, \mathrm{d}\Omega && \text{(DC)} \\
& + \Delta t \int_{\Gamma^e} N_a^e (\boldsymbol{F}_i(\boldsymbol{V}^e) - \boldsymbol{F}_i^{\text{d}}(\boldsymbol{V}^e)) n_i \, \mathrm{d}\Gamma && \text{(Boundary)} ,
\end{aligned} \tag{4.23}$$

where A is the finite element assembly operator, and

$$\nu^e = 2 \frac{(\tilde{\boldsymbol{A}}_i \boldsymbol{V}_{,i}^e + \tilde{\boldsymbol{C}} \boldsymbol{V}^e) \cdot \boldsymbol{\tau} (\tilde{\boldsymbol{A}}_j \boldsymbol{V}_{,j}^e + \tilde{\boldsymbol{C}} \boldsymbol{V}^e)}{\boldsymbol{V}_{,\xi k}^e \cdot \tilde{\boldsymbol{A}}_0 \boldsymbol{V}_{,\xi k}^e + \boldsymbol{V}^e \cdot \tilde{\boldsymbol{A}}_0 \boldsymbol{V}^e} . \tag{4.24}$$

Note that $\mathrm{d}Q$ and $\mathrm{d}P$ are replaced for by $\Delta t \, \mathrm{d}\Omega$ and $\Delta t \, \mathrm{d}\Gamma$, respectively. Up to the second derivative terms, $M^{(i)}$ can be written as

$$\boldsymbol{M}^{(i)} = \mathop{\mathsf{A}}_{e=1}^{n_{\text{el}}} \boldsymbol{M}^e , \tag{4.25}$$

$$\boldsymbol{M}^e = [\boldsymbol{M}^e_{ab}], \quad a, b = 1, \ldots, n_{\text{en}}, \tag{4.26}$$

$$\begin{aligned} M^e_{ab} = & \int_{\Omega^e} N^e_a \tilde{\boldsymbol{A}}_0 N^e_b \, \mathrm{d}\Omega && \text{(Jump-condition)} \\ & + \Delta t \int_{\Omega^e} (-N^e_{a,i} \tilde{\boldsymbol{A}}_i N^e_b + N^e_{a,i} \tilde{\boldsymbol{K}}_{ij} N^e_{b,j} + N^e_a \tilde{\boldsymbol{C}} N^e_b) \, \mathrm{d}\Omega && \text{(Galerkin)} \\ & + \Delta t \int_{\Omega^e} (N^e_{a,i} \tilde{\boldsymbol{A}}_i + N^e_a \tilde{\boldsymbol{C}}) \cdot \boldsymbol{\tau} (N^e_{b,j} \tilde{\boldsymbol{A}}_j + N^e_b \tilde{\boldsymbol{C}}) \, \mathrm{d}\Omega && \text{(Least-squares)} \\ & + \Delta t \int_{\Omega^e} \nu^e (N^e_{a,\xi_i} \tilde{\boldsymbol{A}}_0 N^e_{b,\xi_i} + N^e_a \tilde{\boldsymbol{A}}_0 N^e_b) \, \mathrm{d}\Omega && \text{(DC)} \\ & + \Delta t \int_{\Gamma^e} N^e_a (\tilde{\boldsymbol{A}}_i N^e_b - \tilde{\boldsymbol{K}}_{ij} N^e_{b,j}) n_i \, \mathrm{d}\Gamma && \text{(Boundary)}. \end{aligned} \tag{4.27}$$

The first-order predictor multi-corrector algorithm for the solution of the compressible Euler and Navier–Stokes equations is presented in Box 4.1.1.

In Box 4.1.1, n_{step} is the number of time steps; $i_{\max}$ is the number of corrector passes; i_{freq} is the frequency of formation of the left-hand-side matrix (in practice, we use a different i_{freq} for each element group; see [15] for element group partitioning); and $\boldsymbol{M}^*$ is the effective tangent matrix. This algorithm is a *residual driven* algorithm. Provided the iterative solutions converge, they converge towards the solution of the nonlinear algebraic system $\boldsymbol{G}$. Although the condition of convergence and the convergence rate is highly dependent on the definition of $\boldsymbol{M}^*$ and the values of $i_{\max}$ and i_{freq}, the order of accuracy of the converged solution is not. In general, $\boldsymbol{M}^*$ and i_{freq} are designed based on the stability condition, convergence rate and

Box 4.1.1
First-order predictor multi-corrector algorithm, applied to the compressible Euler and Navier–Stokes equations.

Given n_{step}, $i_{\max}$ and i_{freq}, proceed as follows:
(Initialize)
Set $\boldsymbol{v}_{(0)}$
(Time-steps)
For $n = 0, \ldots, n_{\text{step}} - 1$
(Predictor)
$\boldsymbol{v}^{(0)} = \boldsymbol{v}_{(n)}$
Set Δt
(Multi-corrector loop)
For $i = 0, \ldots, i_{\max} - 1$
Form $\boldsymbol{R}^{(i)}(\boldsymbol{v}^{(i)}; \boldsymbol{v}_{(n)})$
If $\text{mod}(n \times i_{\max} + i, i_{\text{freq}}) = 0$, Form $\boldsymbol{M}^*(\boldsymbol{v}^{(i)})$
Solve for $\Delta \boldsymbol{v}^{(i)}$:
$\boldsymbol{M}^* \Delta \boldsymbol{v}^{(i)} = -\boldsymbol{R}^{(i)}$
$\boldsymbol{v}^{(i+1)} = \boldsymbol{v}^{(i)} + \Delta \boldsymbol{v}^{(i)}$
(End multi-corrector loop)
$\boldsymbol{v}_{(n+1)} = \boldsymbol{v}^{(i+1)}$
(End time step loop)
Exit

computational convenience, and the corresponding $i_{\max}$ is selected based on accuracy. Here we advocate two choices for $\boldsymbol{M}^*$:

(1) *Implicit.* In this case, we set $\boldsymbol{M}^* = \boldsymbol{M}^{(i)}$. This results in a nonsymmetric matrix, having a symmetric profile. The nonsymmetry is caused by the convective part of the Galerkin integral. The nodal-block submatrices of $\boldsymbol{M}^*$ (i.e., $\boldsymbol{M}^e_{ab}$, see (4.27)) are symmetric. We have observed that for the compressible Euler and Navier–Stokes equations the nodal-block diagonal submatrices of $\boldsymbol{M}^*$ are also positive-definite. This also can be verified analytically for some simplified linear systems.

To simplify the implementation of $\boldsymbol{M}^*$, we use the non-integrated-by-parts form of the convective part of the Galerkin integral. This form eliminates the contribution of the convective term to the boundary integral. In addition, since either the magnitude of the diffusive boundary integral is negligible or some kind of boundary conditions are imposed on the boundary, this diffusive integral is omitted from $\boldsymbol{M}^*$. Thus, all boundary integral terms are eliminated in the formation of $\boldsymbol{M}^*$.

(2) *Explicit.* In this case, $\boldsymbol{M}^*$ includes only the jump-condition integral, integrated using nodal quadrature. That is

$$\boldsymbol{M}^* = \begin{bmatrix} \alpha_1 \tilde{\boldsymbol{A}}_0(\boldsymbol{v}_1) & & \\ & \ddots & \\ & & \alpha_{n_{\text{np}}} \tilde{\boldsymbol{A}}_0(\boldsymbol{v}_{n_{\text{np}}}) \end{bmatrix}, \quad (n_{\text{np}} \cdot m) \times (n_{\text{np}} \cdot m), \tag{4.28}$$

where

$$\alpha_A = \sum_{e=1}^{n_{\text{el}}} \begin{cases} (\int_{\Omega^e} \mathrm{d}\Omega)/n_{\text{en}}, & \text{if node } A \text{ is in element } e; \\ 0, & \text{otherwise.} \end{cases} \tag{4.29}$$

This results in a nodally block-diagonal matrix. That is, the nodal points are uncoupled from each other, but the degrees of freedom belonging to each node remain coupled. Since $\tilde{\boldsymbol{A}}_0$ is symmetric positive-definite, so is $\boldsymbol{M}^*$ in this case.

Motivated by the analysis in [14], we define an *algorithmic Courant number*:

$$C_r = 2(\Delta t/h^2)\hat{\lambda}_{\max} + 2(\Delta t/h^2)\bar{\lambda}_{\max}, \tag{4.30}$$

where Δt and h are the element temporal and spatial mesh-size parameters; $\hat{\lambda}_{\max}$ and $\bar{\lambda}_{\max}$ are, respectively, upper bounds on the eigenvalues of the following eigenvalue problems:

$$\left(\tilde{\boldsymbol{K}} - \hat{\lambda}_i \begin{bmatrix} \nwarrow & & \\ & \tilde{\boldsymbol{A}}_0 & \\ & & \searrow \end{bmatrix}\right)\boldsymbol{\Psi}_i = \boldsymbol{0} \quad \text{for } i = 1, \ldots, d \cdot m, \tag{4.31}$$

$$\left(\tilde{\boldsymbol{A}}\tau\tilde{\boldsymbol{A}}^{\mathrm{t}} - \bar{\lambda}_i \begin{bmatrix} \nwarrow & & \\ & \tilde{\boldsymbol{A}}_0 & \\ & & \searrow \end{bmatrix}\right)\boldsymbol{\varphi}_i = \boldsymbol{0} \quad \text{for } i = 1, \ldots, d \cdot m. \tag{4.32}$$

These yield

$$C_\tau = 2(\Delta t/h^2)\max(2\mu/\rho, \kappa/c_v\rho) + (\Delta t/h)u_\tau, \tag{4.33}$$

where μ is the viscosity coefficient; ρ is the density; κ is the coefficient of thermal conductivity; c_v is the specific heat at constant volume;

$$u_\tau = \begin{cases} \left(u^2 + \frac{3}{2}c^2 + c\sqrt{16u^2 + c^2}\right)^{1/2}, & \text{in two-dimensions}, \\ (u^2 + 2c^2 + c\sqrt{4u^2 + c^2})^{1/2}, & \text{in three-dimensions}, \end{cases} \tag{4.34}$$

and u and c are the particle and acoustics speeds.

4.1.3. Local-time stepping strategy

In the predictor multi-corrector algorithm of the previous section, at the beginning of every time step, one time increment, Δt, was computed for the entire spatial domain. This technique is called the *global-time-stepping* strategy. When the transient algorithm is used to obtain a steady solution, the time increment is usually set to the maximum value which satisfies the stability condition everywhere. This means that in problems with high variations in the spatial element size, convective speed and diffusivity properties, the flow information propagates at considerably different rates on different parts of the domain. On some parts of the domain this rate could possibly be optimal and on the rest it could be highly suboptimal. This slows down convergence and increases computational cost. Alternatively, we can locally determine time increments such that the flow information would propagate at nearly optimal rate throughout the domain, without violating the stability condition. This approach is called the *local-time-stepping* strategy.

In our approach to local time-stepping, a different time increment is computed for every nodal point of the domain at the beginning of every time step. The nodal time increment is chosen such that the algorithmic Courant number is equal to a predetermined value, set by the user. This is an appropriate technique since the algorithmic Courant number locally measures the maximum number of elements over which the information of a node propagates in one time step.

The next step is to interpolate the time increment within each element (or set it equal to a constant value in each element), and bring it inside the integrals of $\boldsymbol{R}_a^e$ and $\boldsymbol{M}_{ab}^e$, (4.23) and (4.27), respectively. This procedure, however, leads to loss of flux conservation throughout the domain. It also alters the residual vector related to the steady problem. To overcome these difficulties, we predivide $\boldsymbol{R}_a^e$ and $\boldsymbol{M}_{ab}^e$ by Δt. In this case, only the jump-condition integral in $\boldsymbol{R}_a^e$ and the mass integral in $\boldsymbol{M}_{ab}^e$ are dependent on Δt. As the solution approaches steady state, the jump-condition integral vanishes, hence, the solution becomes independent of Δt. At steady state the solution obtained by using this method is identical to the one obtained by using the global-time-stepping strategy.

See [15] for a numerical evaluation of the local and global time-stepping strategies.

4.2. Linear-in-time approximation

In this section we develop a predictor multi-corrector algorithm based on the linear-in-time approximation of the finite element spaces. This algorithm is the nonlinear version of the third-order predictor multi-corrector algorithm of Shakib and Hughes [14], applied to the compressible Navier–Stokes equations.

4.2.1. Finite element discretization

The finite element spaces are assumed to be linear in time within each space–time slab and discontinuous across them. Hence, within the nth space–time slab, the finite element trial solution can be defined as

$$\boldsymbol{V}^h(\boldsymbol{x}, t) = \sum_{A=1}^{(n_{\mathrm{np}})_{(n)}} N_A^{(n)}(\boldsymbol{x})(\pi_n(t)\boldsymbol{v}_{A;(n+1)} + \tilde{\pi}_n(t)\tilde{\boldsymbol{v}}_{A;(n)}) \quad \text{for } \boldsymbol{x} \in \Omega\,,\ t \in I_n\,, \tag{4.35}$$

where $\boldsymbol{v}_{A;(n+1)}$ and $\tilde{\boldsymbol{v}}_{A;(n)}$ are the $m \times 1$ vectors of unknowns V^h at node A and times t_{n+1}^- and t_n^+, respectively; $\pi_n(t)$ and $\tilde{\pi}_n(t)$ are the temporal shape-functions at space–time slab n:

$$\pi_n(t) = (t - t_n)/\Delta t\,, \tag{4.36}$$

$$\tilde{\pi}_n(t) = (t_{n+1} - t)/\Delta t\,. \tag{4.37}$$

To enhance the stability of the resulting predictor multi-corrector algorithm, we incorporate a preconditioner into the definition of the weighting function; see [14] for a detailed discussion. This yields

$$\boldsymbol{W}^h(\boldsymbol{x}, t) = \sum_{A=1}^{(n_{\mathrm{np}})_{(n)}} N_A^{(n)}(\boldsymbol{x})(\boldsymbol{w}_{A;(n+1)} + (\tilde{\pi}_n(t) - \pi_n(t))\tilde{\boldsymbol{w}}_{A;(n)}) \quad \text{for } \boldsymbol{x} \in \Omega\,,\ t \in I_n\,, \tag{4.38}$$

where $\boldsymbol{w}_{A;(n+1)}$ and $\tilde{\boldsymbol{w}}_{A;(n)}$ are the nodal values of the weighting function corresponding to $\boldsymbol{v}_{A;(n+1)}$ and $\tilde{\boldsymbol{v}}_{A;(n)}$, respectively.

Let $\boldsymbol{v}$, $\boldsymbol{w}$ and $\boldsymbol{v}_{(n)}$ be defined as in (4.5)–(4.7), and define $\tilde{\boldsymbol{v}}$ and $\tilde{\boldsymbol{w}}$ similarly. Then, substituting these definitions and (4.35)–(4.38) into the Galerkin/least-squares variational equation (3.8), and assuming there are no essential boundary conditions, yields

$$\boldsymbol{G}(\boldsymbol{v}; \tilde{\boldsymbol{v}}, \boldsymbol{v}_{(n)}) = \boldsymbol{0}\,, \tag{4.39}$$

$$\tilde{\boldsymbol{G}}(\tilde{\boldsymbol{v}}; \boldsymbol{v}, \boldsymbol{v}_{(n)}) = \boldsymbol{0}\,. \tag{4.40}$$

We restrict the spatial discretizations to the same interpolation functions as before; see the end of Section 4.1.1 for the consequence of this action.

4.2.2. Third-order predictor multi-corrector algorithm

Denote by

$$\boldsymbol{R}^{(i)} = \boldsymbol{G}(\boldsymbol{v}^{(i)}; \tilde{\boldsymbol{v}}^{(i)}, \boldsymbol{v}_{(n)})\,, \tag{4.41}$$

$$\tilde{\boldsymbol{R}}^{(i)} = \tilde{\boldsymbol{G}}(\tilde{\boldsymbol{v}}^{(i)}; \boldsymbol{v}^{(i+1)}, \boldsymbol{v}_{(n)})\,, \tag{4.42}$$

$$\boldsymbol{M}^{(i)} = \partial \boldsymbol{G}(\boldsymbol{v}^{(i)}; \tilde{\boldsymbol{v}}^{(i)}, \boldsymbol{v}_{(n)})/\partial \boldsymbol{v}\,, \tag{4.43}$$

$$\tilde{\boldsymbol{M}}^{(i)} = \partial \tilde{\boldsymbol{G}}(\tilde{\boldsymbol{v}}^{(i)}; \boldsymbol{v}^{(i+1)}, \boldsymbol{v}_{(n)})/\partial \tilde{\boldsymbol{v}}\,, \tag{4.44}$$

the primary and secondary residual vectors and consistent tangent matrices, respectively; where $\tilde{\boldsymbol{v}}^{(i)}$ is the ith iterative approximation of $\tilde{\boldsymbol{v}}$, with $\tilde{\boldsymbol{v}}^{(0)} = \boldsymbol{v}_{(n)}$.

Let $\boldsymbol{V}^e(\boldsymbol{x})$ and $\tilde{\boldsymbol{V}}^e(\boldsymbol{x})$, respectively, be the current iterative approximations of $\boldsymbol{V}^h(\boldsymbol{x}, t_{n+1}^-)$ and $\boldsymbol{V}^h(\boldsymbol{x}, t_n^+)$ in the eth element (see, e.g., (4.20)). In addition, let

$$\bar{\boldsymbol{V}}^e(\boldsymbol{x}) = \tfrac{1}{2}(\boldsymbol{V}^e(\boldsymbol{x}) + \tilde{\boldsymbol{V}}^e(\boldsymbol{x}))\,, \tag{4.45}$$

$$\overset{\triangle}{\boldsymbol{V}}{}^e(\boldsymbol{x}) = \tfrac{1}{2}(\boldsymbol{V}^e(\boldsymbol{x}) - \tilde{\boldsymbol{V}}^e(\boldsymbol{x}))\,, \tag{4.46}$$

denote the average- and difference-in-time values; see Fig. 4.2.1.

Assuming the Jacobian matrices are constant in time as evaluated at the mid-time of the space–time slab (i.e., using $\bar{\boldsymbol{V}}^e(\boldsymbol{x})$ values), then $\boldsymbol{R}^{(i)}$ can be written as

$$\boldsymbol{R}^{(i)} = \mathop{\mathsf{A}}_{e=1}^{n_{\mathrm{el}}} \tilde{\boldsymbol{R}}^e\,, \tag{4.47}$$

$$\boldsymbol{R}^e = \{\tilde{\boldsymbol{R}}_a^e\}\,, \quad a = 1, \ldots, n_{\mathrm{en}}\,, \tag{4.48}$$

$$\begin{aligned}
\boldsymbol{R}_a^e = {} & \int_{\Omega^e} (N_a^e \boldsymbol{U}(\boldsymbol{V}^e) - N_a^e \boldsymbol{U}(\boldsymbol{V}_{(n)}^e))\,\mathrm{d}\Omega && \text{(Jump-condition)} \\
& + \Delta t \int_{\Omega^e} (-N_{a,i}^e \bar{\boldsymbol{F}}_i + N_{a,i}^e \tilde{\boldsymbol{K}}_{ij} \bar{\boldsymbol{V}}_{,j}^e + N_a^e \tilde{\boldsymbol{C}} \bar{\boldsymbol{V}}^e)\,\mathrm{d}\Omega && \text{(Galerkin)} \\
& + \Delta t \int_{\Omega^e} (N_{a,i}^e \tilde{\boldsymbol{A}}_i + N_a^e \tilde{\boldsymbol{C}}) \cdot \boldsymbol{\tau}\left(\frac{2}{\Delta t} \tilde{\boldsymbol{A}}_0 \overset{\triangle}{\boldsymbol{V}}{}^e + \tilde{\boldsymbol{A}}_j \bar{\boldsymbol{V}}_{,j}^e + \tilde{\boldsymbol{C}} \bar{\boldsymbol{V}}^e\right) \mathrm{d}\Omega && \text{(Least-squares)} \\
& + \Delta t \int_{\Omega^e} \nu^e (N_{a,\xi_i}^e \tilde{\boldsymbol{A}}_0 \bar{\boldsymbol{V}}_{,\xi_i}^e + N_a^e \tilde{\boldsymbol{A}}_0 \bar{\boldsymbol{V}}^e)\,\mathrm{d}\Omega && \text{(DC)} \\
& + \Delta t \int_{\Gamma^e} N_a^e (\bar{\boldsymbol{F}}_i - \tilde{\boldsymbol{K}}_{ij} \bar{\boldsymbol{V}}_{,j}^e) n_i\,\mathrm{d}\Gamma\,, && \text{(Boundary)}\,,
\end{aligned} \tag{4.49}$$

where

$$\nu^e = 2\,\frac{(2\tilde{\boldsymbol{A}}_0 \overset{\triangle}{\boldsymbol{V}}{}^e/\Delta t + \tilde{\boldsymbol{A}}_i \bar{\boldsymbol{V}}_{,i}^e + \tilde{\boldsymbol{C}} \bar{\boldsymbol{V}}^e) \cdot \tau (2\tilde{\boldsymbol{A}}_0 \overset{\triangle}{\boldsymbol{V}}{}^e/\Delta t + \tilde{\boldsymbol{A}}_j \bar{\boldsymbol{V}}_{,j}^e + \tilde{\boldsymbol{C}} \bar{\boldsymbol{V}}^e)}{\overset{\triangle}{\boldsymbol{V}}{}^e \cdot \tilde{\boldsymbol{A}}_0 \overset{\triangle}{\boldsymbol{V}}{}^e + \bar{\boldsymbol{V}}_{,\xi_k}^e \cdot \tilde{\boldsymbol{A}}_0 \bar{\boldsymbol{V}}_{,\xi_k}^e + \bar{\boldsymbol{V}}^e \cdot \tilde{\boldsymbol{A}}_0 \bar{\boldsymbol{V}}^e}\,, \tag{4.50}$$

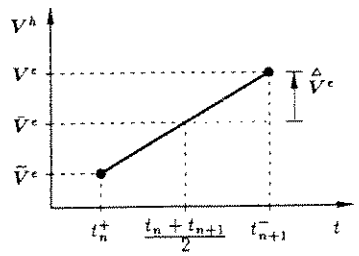


Fig. 4.2.1. Temporal variation of $\boldsymbol{V}^h$ at a given $\boldsymbol{x}$.

and $\bar{F}_i = \int_{I_n} F_i \, dt/\Delta t$ is integrated using a two-point in time Gaussian quadrature. (Using a one-point in time Gaussian quadrature (i.e., setting $\bar{F}_i = F_i(\bar{V}^e)$) degrades accuracy.) Note that when $\tilde{V}^e = V^e$ the residual in (4.49) reduces to the residual of the constant-in-time approximation, (4.23). Therefore, for implementation, the constant-in-time can be considered as a special case of the linear-in-time approximation. The secondary residual, $\tilde{R}^{(i)}$, can be written as

$$\tilde{R}^{(i)} = \mathop{\mathsf{A}}_{e=1}^{n_{\text{el}}} R^e \,, \tag{4.51}$$

$$\tilde{R}^e = \{R^e_a\} \,, \quad a = 1, \ldots, n_{\text{en}} \,, \tag{4.52}$$

$$\begin{aligned}
\tilde{R}^e_a = {} & \int_{\Omega^e} (-N^e_a U(V^e) - N^e_a U(V^e_{(n)})) \, d\Omega && \text{(Jump-condition)} \\
& + \int_{\Omega^e} (2N^e_a \bar{U}) \, d\Omega && \text{(Galerkin-1)} \\
& + \Delta t \int_{\Omega^e} (-N^e_{a,i} \hat{F}_i - \tfrac{1}{3} N^e_{a,i} \tilde{K}_{ij} \overset{\triangle}{V}{}^e_{,j} - \tfrac{1}{3} N^e_a \tilde{C} \overset{\triangle}{V}{}^e) \, d\Omega && \text{(Galerkin-2)} \\
& + \int_{\Omega^e} (-2N^e_a \tilde{A}_0) \cdot \tau \Big(\frac{2}{\Delta t} \tilde{A}_0 \overset{\triangle}{V}{}^e + \tilde{A}_i \bar{V}^e_{,i} + \tilde{C} \bar{V}^e \Big) \, d\Omega && \text{(Least-squares-1)} \\
& + \Delta t \int_{\Omega^e} (-\tfrac{1}{3} N^e_{a,i} \tilde{A}_i - \tfrac{1}{3} N^e_a \tilde{C}) \cdot \tau (\tilde{A}_j \overset{\triangle}{V}{}^e_{,j} + \tilde{C} \overset{\triangle}{V}{}^e) \, d\Omega && \text{(Least-squares-2)} \\
& + \Delta t \int_{\Omega^e} \nu^e (-\tfrac{1}{3} N^e_{a,\xi_i} \tilde{A}_0 \overset{\triangle}{V}{}^e_{,\xi_i} - \tfrac{4}{3} N^e_a \tilde{A}_0 \overset{\triangle}{V}{}^e) \, d\Omega && \text{(DC)} \\
& + \Delta t \int_{\Gamma^e} N^e_a (\hat{F}_i + \tfrac{1}{3} \tilde{K}_{ij} \overset{\triangle}{V}{}^e_{,j}) n_i \, d\Gamma && \text{(Boundary)} \,,
\end{aligned} \tag{4.53}$$

where $\bar{U} = \int_{I_n} U \, dt/\Delta t$ and $\hat{F}_i = \int_{I_n} (\tilde{\pi}_n - \pi_n) F_i \, dt/\Delta t$.

Analogous to Box 4.1.1, the third-order predictor multi-corrector algorithm for the solution of the compressible Euler and Navier–Stokes equations is defined and presented in Box 4.2.1. We advocate two choices for M^*:

(*1*) *Implicit.* In this case, we set M^* to $M^{(i)}$ of the first-order algorithm, (4.25)–(4.27), with Δt replaced by $\frac{1}{2}\Delta t$.

(*2*) *Explicit.* In this case, M^* includes only the jump-condition integral, integrated with a nodal quadrature rule; see (4.28).

4.3. Numerical results

In this section we present numerical examples to demonstrate the stability, accuracy and convergence of the first- and third-order predictor multi-corrector algorithms, applied to the compressible Euler and Navier–Stokes equations.

All computations were done on a CONVEX-C1 in double precision (64 bits per floating point), using 2×2 Gaussian quadrature for bilinear elements and 3-point quadrature for linear triangular elements.

Box 4.2.1.
Third-order predictor multi-corrector algorithm, applied to the compressible Euler and Navier–Stokes equations.

Given n_{step}, i_{max} and i_{freq}, proceed as follows:
(Initialize)
Set $\boldsymbol{v}_{(0)}$
(Time-steps)
For $n = 0, \ldots, n_{step} - 1$
(Predictor)
$\boldsymbol{v}^{(0)} = \tilde{\boldsymbol{v}}^{(0)} = \boldsymbol{v}_{(n)}$
Set Δt
(Multi-corrector loop)
For $i = 0, \ldots, i_{max} - 1$
Form $\boldsymbol{R}^{(i)}(\boldsymbol{v}^{(i)}; \tilde{\boldsymbol{v}}^{(i)}, \boldsymbol{v}_{(n)})$
If $\text{mod}(n \times i_{max} + 2i, i_{freq}) = 0$, Form $\boldsymbol{M}^*(\boldsymbol{v}^{(i)}, \tilde{\boldsymbol{v}}^{(i)})$
Solve for $\Delta \boldsymbol{v}^{(i)}$:
$\boldsymbol{M}^* \Delta \boldsymbol{v}^{(i)} = -\boldsymbol{R}^{(i)}$
$\boldsymbol{v}^{(i+1)} = \boldsymbol{v}^{(i)} + \Delta \boldsymbol{v}^{(i)}$
If $i = i_{max}$, **Exit** the loop
Form $\tilde{\boldsymbol{R}}^{(i)}(\tilde{\boldsymbol{v}}^{(i)}; \boldsymbol{v}^{(i+1)}, \boldsymbol{v}_{(n)})$
If $\text{mod}(n \times i_{max} + 2i + 1, i_{freq}) = 0$, Form $\boldsymbol{M}^*(\boldsymbol{v}^{(i+1)}, \tilde{\boldsymbol{v}}^{(i)})$
Solve for $\Delta \tilde{\boldsymbol{v}}^{(i)}$:
$\boldsymbol{M}^* \Delta \tilde{\boldsymbol{v}}^{(i)} = -\tilde{\boldsymbol{R}}^{(i)}$
$\tilde{\boldsymbol{v}}^{(i+1)} = \tilde{\boldsymbol{v}}^{(i)} + \Delta \tilde{\boldsymbol{v}}^{(i)}$
(End multi-corrector loop)
$\boldsymbol{v}_{(n+1)} = \boldsymbol{v}^{(i+1)}$
(End time-steps)
Exit

4.3.1. *Flow over a flat plate: Stability and spatial accuracy*

The problem of flow over a flat plate is used here to study the stability and spatial accuracy of the first-order predictor multi-corrector algorithm. The problem statement of this two-dimensional Navier–Stokes flow is pictorially shown in Fig. 4.3.1. This problem, known as Carter's problem, consists of a Mach three flow passing over an infinitely thin plate at zero angle of attack, causing a curved shock and a boundary layer to develop from the leading edge of the plate. The Reynolds number, based on the free stream values and distance from the leading edge of the plate, is 10^3. The Sutherland viscosity law, $\mu = 0.0906\,\theta^{1.5}/(\theta + 0.0001406)$, is employed. This corresponds to the free stream temperature of 216.7 K.

The computational domain covers the area $-0.2 \leq x \leq 1.2$, $0 \leq y \leq 0.8$, with the leading edge of the plate placed at $x = 0$. On the inflow boundary ($x = -0.2$) and top boundary ($y = 0.8$), four quantities, ρ, u_1, u_2 and θ, are prescribed; on the line of symmetry ($y = 0$ and $x < 0$), the symmetry conditions, i.e., $u_2 = \tau_{12} = q_2 = 0$, are imposed; on the plate ($y = 0$ and $x \geq 0$), the no-slip condition ($u_1 = u_2 = 0$) and stagnation temperature ($\theta_{stag} = \theta_{inflow}(1 + \frac{1}{2}(\gamma - 1)M^2_{inflow})$) are prescribed; and finally, on the outflow boundary ($x = 1.2$), no quantity is prescribed.

To solve this problem, we employed five uniform nested meshes, consisting of 14×8, 28×16, 56×32, 112×64 and 224×128 square elements, respectively. These correspond to mesh sizes of $h = 0.1$, 0.05, 0.025, 0.0125 and 0.00625, respectively. To validate the results of

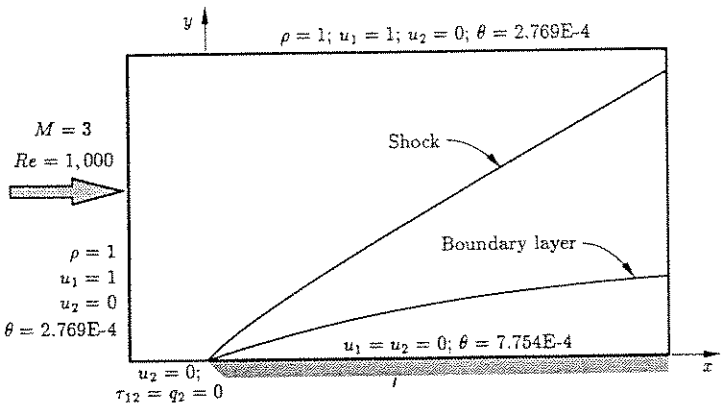


Fig. 4.3.1. Flow over a flat plate problem.

our method, the solution obtained using the fourth mesh is shown in Fig. 4.3.2. This figure presents the contour lines of the computer Mach number and pressure.

We address the stability of the first-order predictor multi-corrector algorithm by examining two quantities: (1) the *stable* algorithmic Courant number, C_τ^{stab}, defined as the value of C_τ which, if exceeded, causes instability; (2) the *optimal* algorithmic Courant number, C_τ^{opt}, at which the best convergence rate is obtained. (Recall, the algorithmic number, C_τ, was defined in (4.33).)

Values of C_τ^{stab} and C_τ^{opt} for the explicit – first-order predictor multi-corrector algorithm using the Galerkin/least-squares method (LS), and Galerkin/least-squares plus the quadratic form of the discontinuity-capturing operator (DC-Quad.) are plotted in Fig. 4.3.3, for the first four meshes. To obtain these values, we employed one corrector pass and the global-time-stepping strategy over fifty time steps. Note that for the Galerkin/least-squares method, both C_τ^{stab} and C_τ^{opt} are greater than one. This is consistent with the definition of C_τ in (4.33), which is a conservative estimate for this method. In contrast, for the method with the discontinuity-capturing operator, $C_\tau^{\text{stab}} = C_\tau^{\text{opt}} < 1$. This result is to be expected because of failure to incorporate the discontinuity-capturing operator in the definition of C_τ (4.33). Finally, note that for (LS), $C_\tau^{\text{opt}} < C_\tau^{\text{stab}}$.

The convergence characteristics of the implicit – first-order predictor multicorrector algorithm using (LS) and (DC-Quad.) computed for the fourth mesh and several values of C_τ are plotted in Fig. 4.3.4. The convergence is measured by the $\tilde{A}_0^{-1}$-norm of the residual. In this study, one corrector pass and the global-time-stepping strategy were employed. Note that the convergence improves with increasing C_τ up to C_τ^{opt} ($\simeq$70, for (LS)), past which the convergence deteriorates. For the (LS), the algorithm becomes unstable when $C_\tau \geqslant 80$. Also note that the discontinuity-capturing operator stabilizes the implicit predictor multi-corrector algorithm. Crude estimations of C_τ^{stab} for (LS) and C_τ^{opt} for (DC-Quad.) are plotted in Fig. 4.3.5. (Because $C_\tau^{\text{opt}} \simeq C_\tau^{\text{stab}}$ for (LS) and accurate evaluation of C_τ^{stab} for (DC-Quad.) was difficult, these two curves are omitted.)

Next, *assuming* that the solution of the fifth mesh is the exact solution, the spatial L_2 norm and H^1 seminorm of the solution errors for the remaining four meshes are computed and plotted as functions of mesh size h in Fig. 4.3.6. This figure also shows the norms computed on

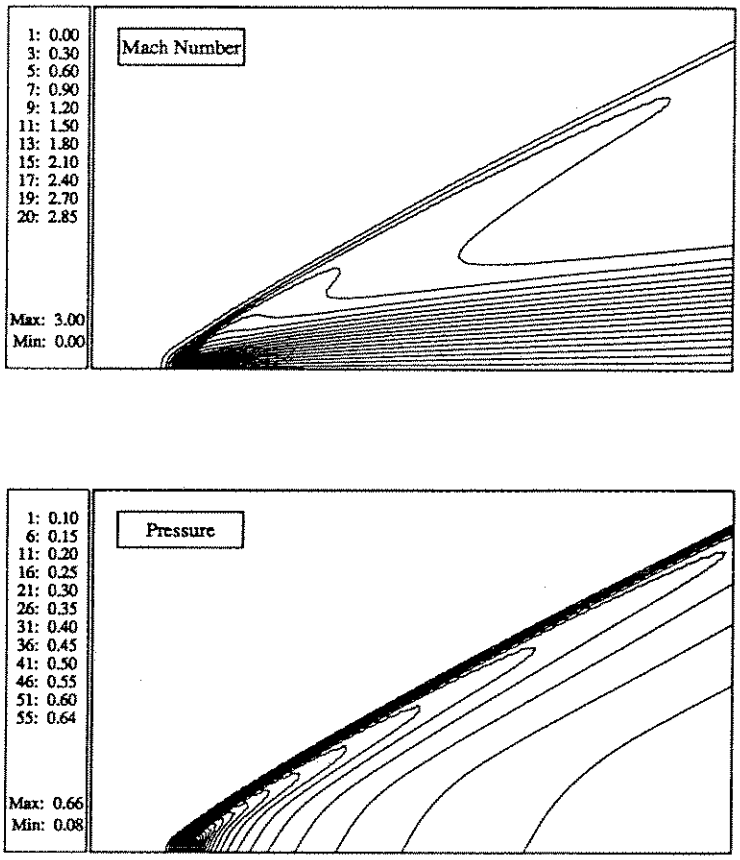


Fig. 4.3.2. Solution of the flow over a flat plate problem (7,168 elements).

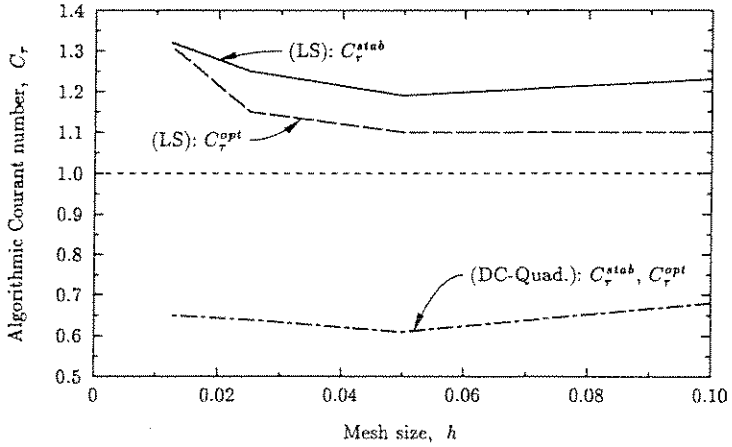


Fig. 4.3.3. Flow over a flat plate. Stable and optimal algorithmic Courant numbers for the explicit – first-order predictor multi-corrector algorithms.

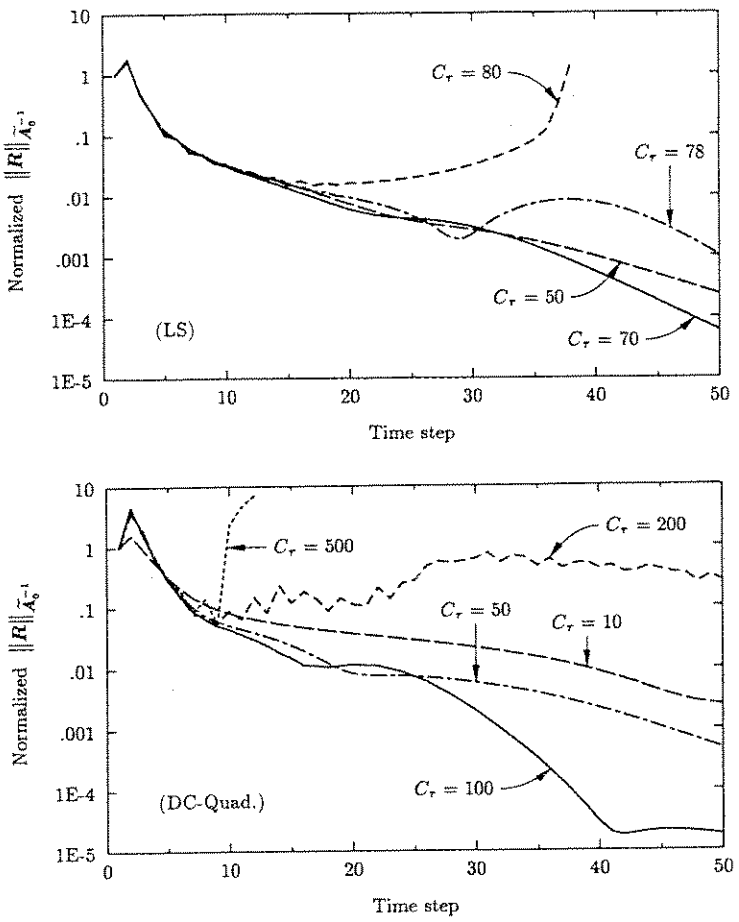


Fig. 4.3.4. Flow over a flat plate (7,168 elements). Convergence of the implicit – first-order predictor multi-corrector algorithm.

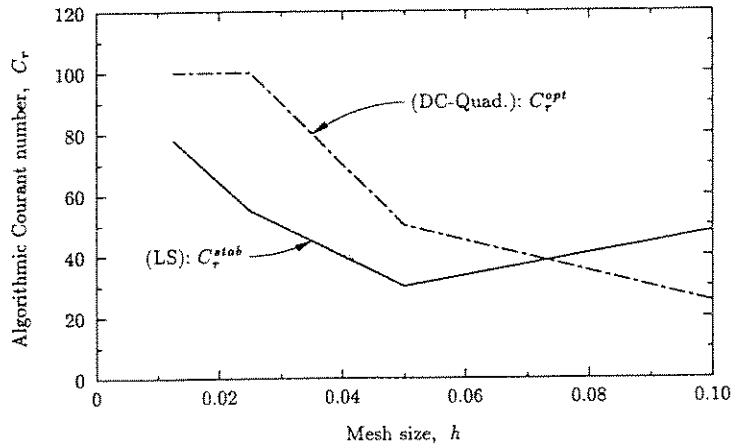


Fig. 4.3.5. Flow over a flat plate. Stable and optimal algorithmic Courant numbers for the implicit – first-order predictor multi-corrector algorithms.

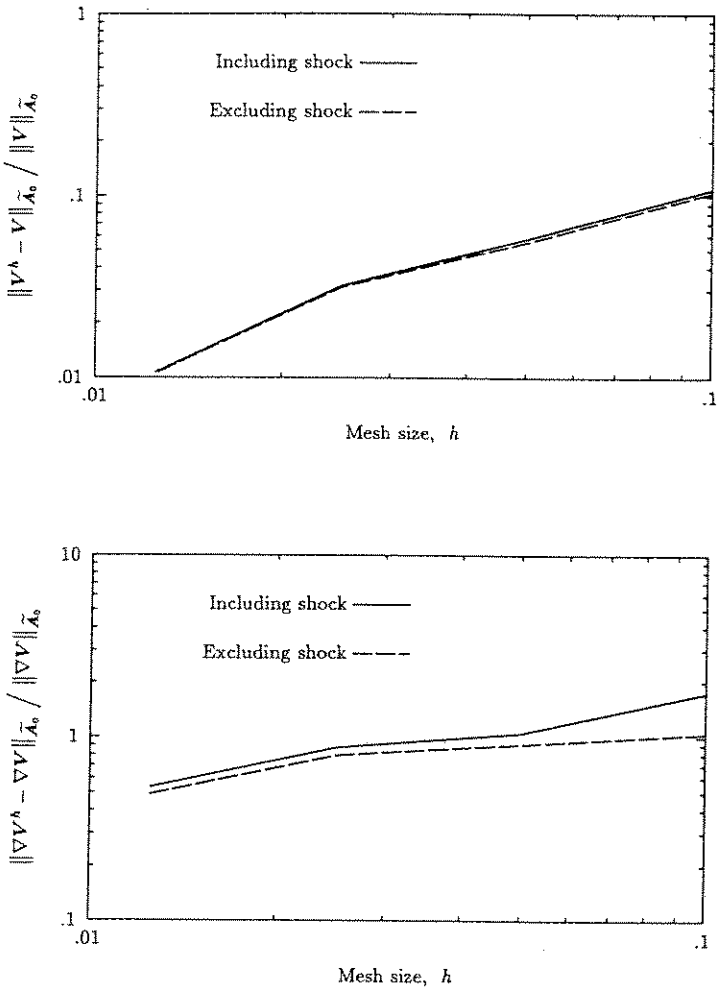


Fig. 4.3.6. Flow over a flat plate. L_2 norm and H^1 seminorm of the spatial error of the solution.

the domain excluding a layer of elements around the shock. The L_2 norm of the error exhibits first-order accuracy for large values of h and order 1.5 accuracy as h decreases.

4.3.2. Acoustic wave: Temporal accuracy

The temporal accuracy of the first- and third-order predictor multi-corrector algorithms is analyzed using a one-dimensional acoustic wave problem. For the initial conditions, one period of a low amplitude sine wave is superimposed on an inviscid fluid at rest. More precisely,

$$\text{at } t = 0: \quad u_1 = c_0 S\,, \quad u_2 = 0\,, \quad \rho = \rho_0(1+S)\,, \quad p = \rho_0 c_0^2 (1+S)\,, \qquad (4.54)$$

where $\rho_0 = 1$ and $c_0 = 1$ are the density and speed of sound for the stationary fluid; $S(x)$ is the initial perturbation:

$$S(x) = \begin{cases} (\rho_0/1000)(1 - \cos((2\pi/\lambda)x)) , & 0 \leq x \leq \lambda , \\ 0 , & \text{otherwise} , \end{cases} \tag{4.55}$$

and λ is the wave length.

With the above data, this 'linear' wave travels virtually without attenuation at the speed of sound in the direction of the increasing x. The density perturbations at the initial time and after traveling one wavelength (at $T = \lambda/c_0$, solved using linear acoustics theory; see [32]) are shown in Fig. 4.3.7.

In order to reduce the effects of the spatial discretizations and far field boundary conditions on the numerical solutions, a uniform mesh having 400×1 square elements extending over $-\lambda \leq x \leq 3\lambda$ is selected. This yields 100 elements per wave length. The values at the boundaries are set to the values of the stationary fluid, and the vertical velocity is set to zero on the entire domain.

The flow is computed over a time interval T for several combinations of the algorithmic parameters. As an example, the computed density using the implicit – third-order algorithm and four corrector passes with the Galerkin/least-squares method is shown in Fig. 4.3.8 for several values of $C_{\Delta t} = \Delta t\, c_0/h$. The convergence curves, for the same combination of algorithmic parameters plus different numbers of corrector passes, are presented in Fig. 4.3.9. Note that for small values of Δt the error is mostly caused by the spatial discretization. We compute the temporal order of accuracy from the mid-range values of Δt. The temporal orders of accuracy of the implicit algorithms are listed in Table 4.3.1. Note that spatial errors may still be adversely effecting convergence rates. The stability of the explicit algorithms (for $C_{\Delta t} < 1$) is summarized in Table 4.3.2. Note that the explicit – third-order algorithm with the Galerkin/least-squares method is stable when three or more corrector passes are employed. Since the stability condition of the explicit algorithms necessitates the use of small time steps, and for small time steps the solution errors mainly come from the spatial discretizations, we have not attempted to calculate the temporal order of accuracy of the explicit algorithms.

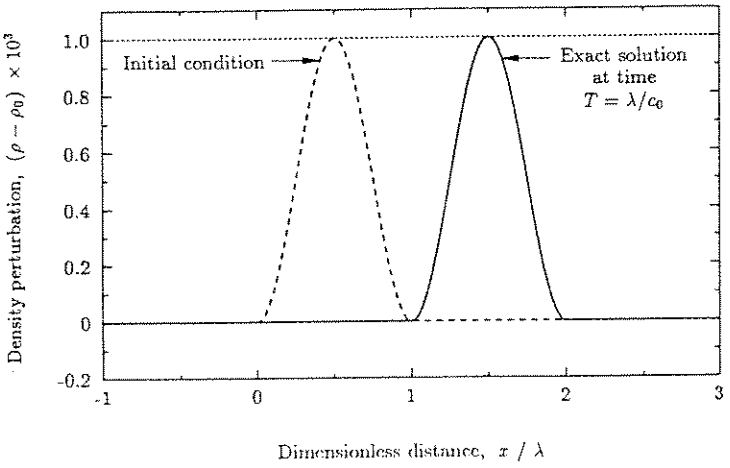


Fig. 4.3.7. Acoustic wave problem. Density perturbations at the initial time and at time $T = \lambda/c_0$.

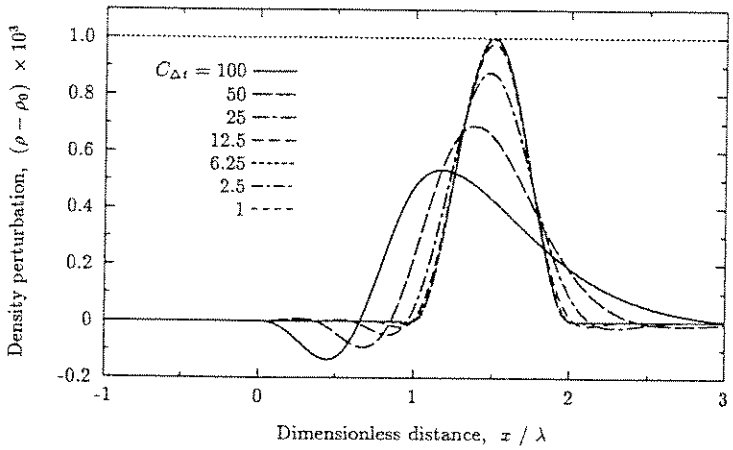


Fig. 4.3.8. Acoustic wave problem. Computed density perturbations at $T = \lambda / c_0$, using four pass implicit – third-order algorithm with the Galerkin/least-squares method.

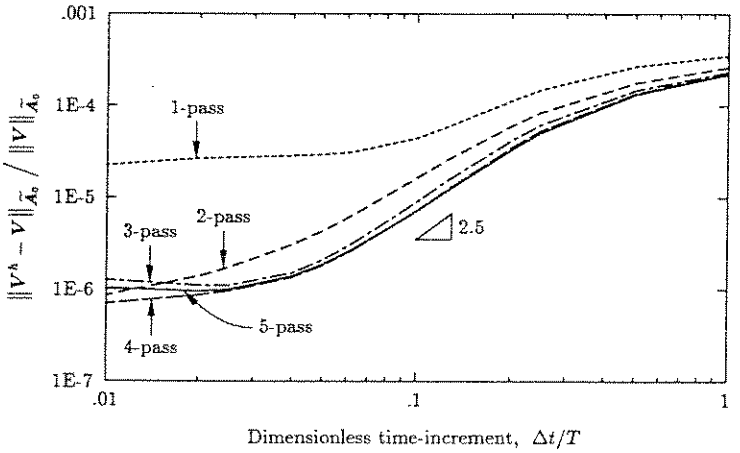


Fig. 4.3.9. Acoustic wave problem. Convergence of the implicit – third-order predictor multi-corrector algorithm with the Galerkin/least-squares method.

Table 4.3.1
Acoustic wave problem. Temporal order of accuracy of the implicit – first- and third-order predictor multi-corrector algorithms

Algorithm	No. passes	Galerkin	Galerkin/least-squares
Implicit –	1	1.96	1.41
3rd-order	2	2.28	1.99
(linear-in-time)	3	2.41	2.35
	4	2.40	2.33
	5	2.35	2.23
Implicit –	1	0.83	0.50
1st-order	2	0.83	0.50
(constant-in-time)	3	0.83	0.50

Table 4.3.2
Acoustic wave problem. Stability of the explicit – first- and third-order predictor multi-corrector algorithms (results are valid for $C_{\Delta t} \leq 0.5$)

Algorithm	No. passes	Galerkin	Galerkin/least-squares
Explicit – 3rd-order (linear-in-time)	1	U	U
	2	U	U
	3	U	S
	4	U	S
Explicit – 1st-order (constant-in-time)	1	U	S
	2	S	S
	3	S	S

S: stable; U: unstable

5. Treatment of boundary conditions and the boundary integral

The two main questions concerning the boundary conditions are: What are the well-posed boundary conditions and how are these conditions implemented within the framework of the nonlinear finite element methods? Many authors have investigated the first question in the context of the compressible Euler and Navier–Stokes equations. Among them, we can refer to [21, 33], who have analyzed the weak form of the differential equations, and determined sets of boundary conditions which lead to energy decaying systems.

In this section we address only the implementational aspects of the boundary conditions. We introduce a nonlinear technique to treat the essential boundary conditions. This technique is capable of handling the nonlinearities arising from the use of working variables other than the variables used to prescribe the boundary conditions. We then extract a set of natural boundary conditions from the boundary integral in our variational equation. Next, we present a consistent method for calculating boundary fluxes, followed by some numerical results.

5.1. *Treatment of essential boundary conditions*

The set of quantities used as our finite element trial functions, namely the $\boldsymbol{V}$-variables, are different from the quantities used to prescribe boundary conditions, such as density, velocity, etc. The equations relating these two sets of variables are highly nonlinear. Consequently, the treatment of essential boundary conditions, in our context, is non-trivial. In this section, we introduce a consistent method to properly treat these nonlinearities within the framework of the predictor multicorrector algorithms.

5.1.1. *Essential boundary conditions as constraint equations*

Let $\boldsymbol{q}$ be a list of prescribed quantities. Then the essential boundary conditions can be expressed in terms of $\boldsymbol{q}$ as

$$q_\alpha(x, t) = g_\alpha(x, t) \quad \text{for } x \in \Gamma_\alpha \,, \; t \in]0, T[\,, \tag{5.1}$$

where the index α refers to the quantity being prescribed on $\Gamma_\alpha \subset \Gamma$, and g_α is its prescribed

function. In general, the range of α is higher than m, where m is the number of differential equations in the system; but, at most m independent quantities can be prescribed at a given point on the boundary. To simplify the analysis, let $\boldsymbol{q}$ include only the primitive variables. That is, in three dimensions,

$$\boldsymbol{q} = \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{Bmatrix} = \begin{Bmatrix} \rho: & \text{density} \\ u_r: & \text{component of velocity in } r\text{-direction} \\ u_s: & \text{component of velocity in } s\text{-direction} \\ u_t: & \text{component of velocity in } t\text{-direction} \\ \theta: & \text{temperature} \\ p: & \text{pressure} \end{Bmatrix}, \tag{5.2}$$

where u_r, u_s and u_t are the prescribed components of the velocity in the orthogonal coordinate system (r, s, t) which is oriented according to the physical problem. More precisely,

$$q_2 = u_r = C_1^r u_1 + C_2^r u_2 + C_3^r u_3 , \tag{5.3}$$

$$q_3 = u_s = C_1^s u_1 + C_2^s u_2 + C_3^s u_3 , \tag{5.4}$$

$$q_4 = u_t = C_1^t u_1 + C_2^t u_2 + C_3^t u_3 , \tag{5.5}$$

where u_1, u_2 and u_3 are velocity components in the global coordinate system. C_i^r, C_i^s and C_i^t are, respectively, the direction cosines of the r, s and t axes with respect to the global coordinate system. Without loss of generality, we assume that $|C_1^r| \geq |C_{i\neq 1}^r|$, $|C_2^s| \geq |C_{i\neq 2}^s|$ and $|C_3^t| \geq |C_{i\neq 3}^t|$. Note that the list $\boldsymbol{q}$, in (5.2), can be extended to include other prescribed scalar and vector quantities, such as Mach number, mass-flux vector, etc.

The prescribed boundary conditions, (5.1), can be rewritten in terms of the $\boldsymbol{V}$-variables as

$$q_\alpha(\boldsymbol{V}) = g_\alpha , \tag{5.6}$$

where $\boldsymbol{q}$ is now assumed to be a function of $\boldsymbol{V}$. From the definition of $\boldsymbol{V}$, (2.15)–(2.17) and (5.2) we have

$$\boldsymbol{q}(\boldsymbol{V}) = \begin{Bmatrix} \rho \\ u_r \\ u_s \\ u_t \\ \theta \\ p \end{Bmatrix} = \begin{Bmatrix} \bar{\gamma}^{1/\bar{\gamma}}(-V_5)^{-1/\bar{\gamma}} \exp((-1/\bar{\gamma})(\gamma - V_1 + (V_2^2 + V_3^2 + V_4^2)/2V_5)) \\ C_1^r(-V_2/V_5) + C_2^r(-V_3/V_5) + C_3^r(-V_4/V_5) \\ C_1^s(-V_2/V_5) + C_2^s(-V_3/V_5) + C_3^s(-V_4/V_5) \\ C_1^t(-V_2/V_5) + C_2^t(-V_3/V_5) + C_3^t(-V_4/V_5) \\ (1/c_v)(-1/V_5) \\ \bar{\gamma}^{\gamma/\bar{\gamma}}(-V_5)^{-\gamma/\bar{\gamma}} \exp((-1/\bar{\gamma})(\gamma - V_1 + (V_2^2 + V_3^2 + V_4^2)/2V_5)) \end{Bmatrix}, \tag{5.7}$$

where $\bar{\gamma} = \gamma - 1$.

We proceed by rewriting (5.6) in the form

$$V_i = \hat{q}_\alpha(V_{j\neq i};\, g_\alpha) , \tag{5.8}$$

where the function $\hat{q}_\alpha$ relates the ith component of V to the remaining components of V and the prescribed value of the boundary condition (i.e., g_α). Before explicitly defining the above relations, it is necessary to determine which V_i should be constrained for a given q_α. When the temperature is prescribed (i.e., $q_5 = g_5$), it is clear from (5.7) that V_5 should be constrained. In the particular cases when the (r, s, t) coordinate system coincides with the global system, and one or more of the velocities u_r, u_s and u_t are prescribed (i.e., $q_2 = u_1 = g_2$, $q_3 = u_2 = g_3$ or $q_4 = u_3 = g_4$), respectively V_2, V_3 or V_4 should be constrained. In general configurations, when the (r, s, t) coordinate system does not coincide with the global axes, as long as C_1^r, C_2^s and C_3^t are bounded away from zero, this selection is still appropriate. Finally, when either density or pressure is prescribed (i.e., $q_1 = g_1$ or $q_6 = g_6$), V_1 should be constrained. With this selection, one can simultaneously prescribe any of the quantities in the list $\boldsymbol{q}$, with one exception: Since density and pressure constrain the same variable (i.e., V_1), they cannot be prescribed simultaneously. However, this does not pose any difficulty, since from the perfect gas law, (2.6) and (2.7), density and temperature can be prescribed equivalently.

Substituting the above selection into (5.7) leads to the constraint equation in the desired form (5.8):

$$\begin{aligned}
&\text{for } \rho: \quad V_1 = \gamma - \ln \bar{\gamma} + \bar{\gamma} \ln g_1 + \ln(-V_5) + \tfrac{1}{2}(V_2^2 + V_3^2 + V_4^2)/V_5 ,\\
&\text{for } u_r: \quad V_2 = -(C_2^r/C_1^r)V_3 - (C_3^r/C_1^r)V_4 - (g_2/C_1^r)V_5 ,\\
&\text{for } u_s: \quad V_3 = -(C_1^s/C_2^s)V_2 - (C_3^s/C_2^s)V_4 - (g_3/C_2^s)V_5 ,\\
&\text{for } u_t: \quad V_4 = -(C_1^t/C_3^t)V_2 - (C_2^t/C_3^t)V_3 - (g_4/C_3^t)V_5 ,\\
&\text{for } \theta: \quad V_5 = -1/c_v g_5 ,\\
&\text{for } p: \quad V_1 = \gamma - \gamma \ln \bar{\gamma} + \bar{\gamma} \ln g_6 + \gamma \ln(-V_5) + \tfrac{1}{2}(V_2^2 + V_3^2 + V_4^2)/V_5 .
\end{aligned} \tag{5.9}$$

These constraint equations are not only nonlinear in V but also coupled (i.e., they are functions of each other). For example, the constraint equation for density, which is a function of g_1, is also a function of V_5. If the temperature is prescribed along with the density, then the density constraint equation is indirectly a function of g_5. In this case, either the density constraint equation should be expressed in terms of g_1 and g_5 (instead of g_1 and V_5), or else the temperature constraint equation should be satisfied first. The latter choice is preferred. In general, when two or more boundary conditions are prescribed at a given point, their corresponding constraint equations should be satisfied sequentially in the order: temperature, velocities, and then density or pressure. When two or more velocities are prescribed, their couplings must be reconciled. This is always possible. For instance, assume that two velocities u_r and u_s are prescribed:

$$u_r = C_1^r u_1 + C_2^r u_2 + C_3^r u_3 , \tag{5.10}$$

$$u_s = C_1^s u_1 + C_2^s u_2 + C_3^s u_3 . \tag{5.11}$$

Then from (5.9), we have

$$V_2 = -(C_2^r/C_1^r)V_3 - (C_3^r/C_1^r)V_4 - (u_r/C_1^r)V_5 , \tag{5.12}$$

$$V_3 = -(C_1^s/C_2^s)V_2 - (C_3^s/C_2^s)V_4 - (u_s/C_2^s)V_5 \,. \tag{5.13}$$

Note that V_2 depends on V_3, and V_3 depends on V_2. To resolve this problem, we can eliminate u_2 from (5.10) and u_1 from (5.11):

$$\bar{u}_r = \bar{C}_1^r u_1 \qquad + \bar{C}_3^r u_3 \,, \tag{5.14}$$

$$\bar{u}_s = \qquad \bar{C}_2^s u_2 + \bar{C}_3^s u_3 \,. \tag{5.15}$$

Thus,

$$V_2 = -(\bar{C}_3^r/\bar{C}_1^r)V_4 - (\bar{u}_r/\bar{C}_1^r)V_5 \,, \tag{5.16}$$

$$V_3 = -(\bar{C}_3^s/\bar{C}_2^s)V_4 - (\bar{u}_s/\bar{C}_2^s)V_5 \,. \tag{5.17}$$

These equations are equivalent to (5.12) and (5.13), but do not exhibit any coupling between V_2 and V_3, and can be satisfied independently of each other. Finally, when all velocities are prescribed, the constraint velocities can be expressed with respect to the global coordinate system. This eliminates the couplings.

In addition to the constraint equations for $\boldsymbol{V}$, (5.9), we need the constraint equations for the weighting function, $\boldsymbol{W}$, and for the $\boldsymbol{V}$ increments, $\Delta\boldsymbol{V}$. In finite element methods based on the minimization of a potential function, the weighting function belongs to the tangent space of $\boldsymbol{V}$. We use the same concept here and derive the constraint equation for $\boldsymbol{W}$ by taking the variation of the constraint equations for $\boldsymbol{V}$. That is,

$$W_i = \delta V_i = \sum_{j \neq i} \frac{\partial \hat{q}_\alpha(V_{k\neq i};\, g_\alpha)}{\partial V_j} W_j \,, \tag{5.18}$$

where δ is the variation operator. Substituting (5.9) into the above equation leads to the $\boldsymbol{W}$ constraint equations

$$\begin{aligned}
&\text{for } \rho\text{:} \quad W_1 = \frac{V_2}{V_5} W_2 + \frac{V_3}{V_5} W_3 + \frac{V_4}{V_5} W_4 + \left(\frac{1}{V_5} - \frac{V_2^2 + V_3^2 + V_4^2}{2V_5^2}\right) W_5 \,, \\
&\text{for } u_r\text{:} \quad W_2 = -(C_2^r/C_1^r)W_3 - (C_3^r/C_1^r)W_4 - (g_2/C_1^r)W_5 \,, \\
&\text{for } u_s\text{:} \quad W_3 = -(C_1^s/C_2^s)W_2 - (C_3^s/C_2^s)W_4 - (g_3/C_2^s)W_5 \,, \\
&\text{for } u_t\text{:} \quad W_4 = -(C_1^t/C_3^t)W_2 - (C_2^t/C_3^t)W_3 - (g_4/C_3^t)W_5 \,, \\
&\text{for } \theta\text{:} \quad W_5 = 0 \,, \\
&\text{for } p\text{:} \quad W_1 = \frac{V_2}{V_5} W_2 + \frac{V_3}{V_5} W_3 + \frac{V_4}{V_5} W_4 + \left(\frac{\gamma}{V_5} - \frac{V_2^2 + V_3^2 + V_4^2}{2V_5^2}\right) W_5 \,.
\end{aligned} \tag{5.19}$$

The constraint equations (5.19) are linear in $\boldsymbol{W}$, with coefficients nonlinear in $\boldsymbol{V}$. For reference, let us rewrite (5.19) as

$$q'_\alpha(\boldsymbol{W}) = 0 \quad \text{on } \Gamma_\alpha \times]0, T[\,. \tag{5.20}$$

The constraint equations for ΔV are derived by linearizing the constraint equations for $\boldsymbol{V}$. This leads to constraint equations (5.19) with W_i's replaced by ΔV_i's.

REMARK 5.1. In finite element applications where the prescribed quantities are the same as the trial solution components, the constraint equations for $\boldsymbol{V}$ become $V_i = g_i$ and the corresponding constraint equations for $\boldsymbol{W}$ and $\Delta \boldsymbol{V}$ become $W_i = 0$ and $\Delta V_i = 0$.

5.1.2. Incorporation in predictor multi-corrector algorithms

Discretization and linearization of the finite element variational equation, (3.8), leads to the scalar equation

$$\boldsymbol{w} \cdot \boldsymbol{M}\, \Delta \boldsymbol{v} = -\boldsymbol{w} \cdot \boldsymbol{R}\,, \tag{5.21}$$

which holds for all unconstrained $\boldsymbol{w}$; see Section 4 for details, in particular, see (4.8) and (4.16)–(4.19). Here $\boldsymbol{w} = \{\boldsymbol{w}_1^{\mathrm{t}}, \boldsymbol{w}_2^{\mathrm{t}}, \ldots, \boldsymbol{w}_{n_{\mathrm{np}}}^{\mathrm{t}}\}^{\mathrm{t}}$; where $\boldsymbol{w}_A$ is $\boldsymbol{W}$ evaluated at node A; $\Delta \boldsymbol{v} = \{\Delta \boldsymbol{v}_1^{\mathrm{t}}, \Delta \boldsymbol{v}_2^{\mathrm{t}}, \ldots, \Delta \boldsymbol{v}_{n_{\mathrm{np}}}^{\mathrm{t}}\}^{\mathrm{t}}$; where $\Delta \boldsymbol{v}_A$ is $\Delta \boldsymbol{V}$ evaluated at node A; n_{np} is the number of nodal points; $\boldsymbol{M}$ is the effective tangent matrix; and $\boldsymbol{R}$ is the residual vector. Before extracting a linear system of equations from (5.21), it is necessary to satisfy the $\boldsymbol{w}$ constraint equations: Define the density transformation matrix as

$$\boldsymbol{S}_1 = \begin{bmatrix} 0 & S_{12} & S_{13} & S_{14} & S_{15} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{5.22}$$

with coefficients

$$S_{12} = \frac{V_2}{V_5}\,; \quad S_{13} = \frac{V_3}{V_5}\,; \quad S_{14} = \frac{V_4}{V_5}\,; \quad S_{15} = \frac{1}{V_5} - \frac{V_2^2 + V_3^2 + V_4^2}{2V_5^2}\,. \tag{5.23}$$

Then the transformation of $\boldsymbol{W}$ by $\boldsymbol{S}_1$ yields

$$\boldsymbol{S}_1 \boldsymbol{W} = \begin{Bmatrix} \dfrac{V_2}{V_5} W_2 + \dfrac{V_3}{V_5} W_3 + \dfrac{V_4}{V_5} W_4 + \left(\dfrac{1}{V_5} - \dfrac{V_2^2 + V_3^2 + V_4^2}{2V_5^2}\right) W_5 \\ W_2 \\ W_3 \\ W_4 \\ W_5 \end{Bmatrix}. \tag{5.24}$$

The first term in (5.24) is precisely the density constraint equation for $\boldsymbol{W}$; see (5.19). Consequently, to satisfy the density constraint for $\boldsymbol{w}$ and $\Delta \boldsymbol{v}$ at node A, $\boldsymbol{w}$ and $\Delta \boldsymbol{v}$ vectors are replaced by $\boldsymbol{S}_1^A \boldsymbol{w}$ and $\boldsymbol{S}_1^A\, \Delta \boldsymbol{v}$ in (5.21); where $\boldsymbol{S}_1^A = \mathrm{diag}(\mathbf{I}_m, \ldots, \boldsymbol{S}_1, \ldots, \mathbf{I}_m)$ is an $(n_{\mathrm{np}} \cdot m) \times$

$(n_{np} \cdot m)$ identity matrix with its Ath nodal block-diagonal entry replaced by S_1. This yields a new equation

$$\boldsymbol{w} \cdot \bar{\boldsymbol{M}}\, \Delta \boldsymbol{v} = -\boldsymbol{w} \cdot \bar{\boldsymbol{R}}\,, \tag{5.25}$$

where

$$\bar{\boldsymbol{M}} = (\boldsymbol{S}_1^A)^t \boldsymbol{M} \boldsymbol{S}_1^A\,, \tag{5.26}$$

$$\bar{\boldsymbol{R}} = (\boldsymbol{S}_1^A)^t \boldsymbol{R}\,. \tag{5.27}$$

Equation (5.25) automatically satisfies the density constraint equations for $\boldsymbol{w}$ and $\Delta \boldsymbol{v}$ at node A. Therefore, if there is only a density boundary condition and only at node A, then satisfying (5.21) for all unconstrained $\boldsymbol{w}$ becomes equivalent to satisfying (5.25) for all $\boldsymbol{w}$; or

$$\bar{\boldsymbol{M}}\, \Delta \boldsymbol{v} = -\bar{\boldsymbol{R}}\,. \tag{5.28}$$

Because of the transformations (5.26) and (5.27), the row and column entries of $\bar{\boldsymbol{M}}$ and the row entry of $\bar{\boldsymbol{R}}$ corresponding to the constrained degree of freedom are zero. (Recall that the first column of $\boldsymbol{S}_1$ is zero.) In order to avoid singularity in (5.28), the diagonal entry of $\bar{\boldsymbol{M}}$ corresponding to the constrained degree of freedom can be set to a non-zero value. This does not effect the solution of the linear system. In addition, this makes the existence of boundary conditions transparent to the linear solver. Since the transformations (5.26) and (5.27) are linear with respect to $\boldsymbol{w}$ and $\Delta \boldsymbol{v}$, they can be performed on the element arrays before assembly. This is an important implementational consideration.

Similar to (5.22), the transformation matrices for velocities, temperature and pressure are defined and listed in Appendix B. With these matrices, satisfaction of the $\boldsymbol{w}$ and $\Delta \boldsymbol{v}$ constraint equations amounts to the recursive transformation of the element arrays as

$$\boldsymbol{M}^e \leftarrow (\boldsymbol{S}_\alpha^A)^t \boldsymbol{M}^e (\boldsymbol{S}_\alpha^A)\,, \tag{5.29}$$

$$\boldsymbol{R}^e \leftarrow (\boldsymbol{S}_\alpha^A)^t \boldsymbol{R}^e\,, \tag{5.30}$$

for all constraints α of nodes A; where $\boldsymbol{S}_\alpha^A = \mathrm{diag}(\mathbf{I}_m, \ldots, \boldsymbol{S}_\alpha, \ldots, \mathbf{I}_m)$ is an $(n_{np} \cdot m) \times (n_{np} \cdot m)$ identity matrix with its Ath nodal block-diagonal entry replaced by $\boldsymbol{S}_\alpha$.

In summary, to account for the essential boundary conditions, the following two steps are added to the predictor multi-corrector algorithms of Section 4:

(1) When $\boldsymbol{v}$ is updated at the end of every predictor and corrector step, the constrained degrees of freedom of $\boldsymbol{v}$ are updated sequentially using (5.9).
(2) Upon their formation, the element effective tangent and residual arrays are transformed recursively using (5.29) and (5.30).

Numerical experience has shown that elimination of the second step in the above procedure can greatly degrade numerical convergence.

REMARK 5.2. If we assume that the variational equation (3.8) results from minimizing some

potential function, then the consistent linearized tangent matrix with constraints is the sum of two terms. The first and crucial term is the unconstrained left-hand-side matrix symmetrically transformed by the first derivative of the constraint equations. This term is provided by (5.29). The second term is the unconstrained residual vector transformed by the second derivative of the constraint equations. This amounts to adding the $m \times m$ matrix $\hat{\boldsymbol{M}} = [\hat{M}_{jk}] = R_i^A \partial^2 \hat{q}_\alpha / \partial v_j^A \partial v_k^A$, $1 \leq j, k \leq m$, to the Ath nodal block-diagonal entry of the left-hand side matrix, for every constraint α of node A. Here, R_i^A is the ith component of the residual vector (prior to (5.30)) at node A; i is the degree of freedom corresponding to the constraint α; $\hat{q}_\alpha(v_{j\neq i}^A; g_\alpha)$ is the constraint equation defined in (5.8); and v_j^A is the jth component of $\boldsymbol{v}_A$. For the density constraint ($\alpha = 1$ and $i = 1$), we have

$$\hat{\boldsymbol{M}} = R_1^A \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ & 1/v_5^A & 0 & 0 & -v_2^A/(v_5^A)^2 \\ & & 1/v_5^A & 0 & -v_3^A/(v_5^A)^2 \\ & \text{symm} & & 1/v_5^A & -v_4^A/(v_5^A)^2 \\ & & & & -\dfrac{1}{(v_5^A)^2} + \dfrac{(v_2^A)^2 + (v_3^A)^2 + (v_4^A)^2}{(v_5^A)^3} \end{bmatrix}. \tag{5.31}$$

For the velocity and temperature constraints $\hat{\boldsymbol{M}}$ is zero, and for the pressure constraint $\hat{\boldsymbol{M}}$ has a construction similar to (5.31). The addition of this term may further improve the stability and convergence of the predictor multi-corrector algorithms. This is a topic for future research.

REMARK 5.3. Since the boundary nodes make up only a small subset of the total number of nodes in an analysis, it is not cost effective to vectorize the operations for the many possible combinations of boundary conditions. This is perhaps the only place in our finite element computer program where vectorization is not attained.

5.2. *The boundary integral and natural boundary conditions*

The boundary integral in the variational formulation (3.8) is

$$\int_{P_n} \boldsymbol{W}^h \cdot (-\boldsymbol{F}_i(\boldsymbol{V}^h) + \boldsymbol{F}_i^{\mathrm{d}}(\boldsymbol{V}^h)) n_i \, \mathrm{d}P \,, \tag{5.32}$$

where $\boldsymbol{F}_i$ and $\boldsymbol{F}_i^{\mathrm{d}}$ are the Euler and diffusive fluxes, respectively, and n_i is the ith component of the spatial unit outward normal to the space–time boundary, P_n. Recall that this boundary integral resulted from the integration by parts of the Galerkin term. Substituting the definition of the fluxes from (2.3) and (2.4) into the above integral, yields (in three dimensions)

$$\int_{P_n} \boldsymbol{W}^h \cdot \left(-\rho u_n \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e \end{Bmatrix} - p \begin{Bmatrix} 0 \\ n_1 \\ n_2 \\ n_3 \\ u_n \end{Bmatrix} + \begin{Bmatrix} 0 \\ \tau_{1n} \\ \tau_{2n} \\ \tau_{3n} \\ \tau_{in} u_i \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -q_n \end{Bmatrix} \right) \mathrm{d}P \,, \tag{5.33}$$

where e is the total energy; $\boldsymbol{\tau}$ is the viscous stress; $\boldsymbol{q}$ is the heat flux vector; and the subscript n refers to the outward normal component. The above boundary integral provides the following natural boundary conditions:

$$(1)\ \textit{Normal mass flux}:\ \rho u_n = h^m\ , \tag{5.34}$$

$$(2)\ \textit{Pressure}:\ p = h^p\ , \tag{5.35}$$

$$(3)\ \textit{Normal viscous flux}:\ \tau_{in} = h_i^v\ , \tag{5.36}$$

$$(4)\ \textit{Normal heat flux}:\ -q_n = h^h\ , \tag{5.37}$$

where h^m, h^p, $\boldsymbol{h}^v$ and h^h are the prescribed data.

It is essential to include the boundary integral on the entire boundary. Neglecting this leads to loss of flux conservation in the method. If a natural boundary condition is imposed, then the prescribed value is substituted for the corresponding flux on that part of the boundary. Otherwise, the fluxes are evaluated from the current values of V^h. Note that pressure can be prescribed either strongly, as described in Section 5.1, or weakly, as above. Experience, however, has shown that strongly enforced pressure leads to higher numerical stability and faster convergence. A total normal stress boundary condition can be obtained by combining the pressure term with the viscous flux.

The following two combinations of boundary conditions are worth noting:

(1) *Symmetry condition.* Let (r, s, n) be an orthogonal coordinate system with n pointing in the direction of the outward normal to the boundary. Then the symmetry condition dictates: $u_n = 0$, $u_{n,r} = u_{n,s} = 0$, $u_{r,n} = u_{s,n} = 0$, and $\theta_{,n} = 0$. Substitution of these conditions into the definitions of $\boldsymbol{\tau}$ and $\boldsymbol{q}$ yields $\tau_{rn} = \tau_{sn} = q_n = 0$. Thus, the symmetry condition becomes

$$u_n = 0\ , \tag{5.38}$$

$$\tau_{rn} = \tau_{sn} = q_n = 0\ . \tag{5.39}$$

(2) *Outflow condition.* In advection-dominated (high Reynolds number) flows, no boundary condition is imposed on the outflow boundaries. Nevertheless, the boundary integral must be included for all flux terms, including the viscous and heat transfer contributions. These last two terms are often neglected. The rationale for this is that the omission of these terms (i.e., assuming traction- and heat-flux-free boundary) is consistent with normal gradient-free flow at the boundary. Setting $\tau_{rn} = \tau_{sn} = 0$ corresponds to weakly enforcing $u_{r,n} + u_{n,r} = 0$ and $u_{s,n} + u_{n,s} = 0$. These conditions are erroneous. This fact is demonstrated using the flow over a flat plate problem described in Section 4.3.1. Here, the (r, s, n) coordinate system line up with the global (y, z, x) axes. Near the wall at the outflow boundary $u_{1,2} > 0$. As explained above, the traction-free boundary condition weakly enforces $u_{2,1} = -u_{1,2} < 0$. This leads to oscillations of the velocity vectors in the corner element as shown in Fig. 5.2.1.

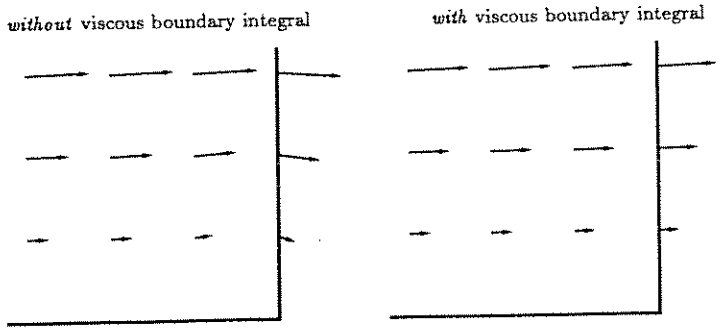


Fig. 5.2.1. Flow over a flat plate. Velocity vectors by the wall at the outflow boundary.

5.3. *Consistent calculation of boundary fluxes*

Traditionally the wall quantities, namely the wall viscous and heat fluxes, are calculated by substituting the numerical derivatives of the flow quantities $\boldsymbol{u}$ and θ (e.g., $\boldsymbol{u}_{,i} = \sum_{A=1}^{n_{\mathrm{np}}} N_{A,i}\boldsymbol{u}_A$) into the definition of the fluxes, (2.8) and (2.9). Although this *classical* method may provide adequate results, it does not utilize all of the information available in the variational equation. To correct for this, the *consistent* boundary-flux calculation technique has been developed; for reference see [34–37].

5.3.1. *Formulation*

The wall viscous flux, $\bar{\boldsymbol{h}}^v$, and heat flux, $\bar{h}^h$, are defined as

$$\bar{h}_i^v = \tau_{ij} n_j \,, \tag{5.40}$$

$$\bar{h}^h = -q_i n_i \,. \tag{5.41}$$

These are the unknown counterparts of the boundary conditions h_i^v and h^h in (5.36) and (5.37). Consider the flux vector f defined on the wall boundary $(P_f)_n = \Gamma_f \times I_n$ as

$$f = \begin{Bmatrix} 0 \\ \bar{\boldsymbol{h}}^v \\ \bar{h}^h \end{Bmatrix} = \boldsymbol{F}_i^{\mathrm{d}} n_i \quad \text{on } (P_f)_n \,, \tag{5.42}$$

where the velocity on the wall is assumed to be zero (in general $f_{\mathrm{d}} = \bar{h}^h + \boldsymbol{u}\cdot\bar{\boldsymbol{h}}^v$). Taking $\boldsymbol{f}^h$, the discrete counterpart of $\boldsymbol{f}$, to be unknown and $\boldsymbol{V}^h$ to be known, the variational formulation (3.8) can be generalized to

$$\begin{aligned}
&\int_{(P_f)_n} \boldsymbol{W}^h \cdot \boldsymbol{f}^h \,\mathrm{d}P + \int_{P_n - (P_f)_n} \boldsymbol{W}^h \cdot \boldsymbol{F}_i^{\mathrm{d}}(\boldsymbol{V}^h) n_i \,\mathrm{d}P \\
&\quad = \int_{Q_n} (-\boldsymbol{W}_{,t}^h \cdot \boldsymbol{U}(\boldsymbol{V}^h) - \boldsymbol{W}_{,i}^h \cdot \boldsymbol{F}_i(\boldsymbol{V}^h) + \boldsymbol{W}_{,i}^h \cdot \tilde{\boldsymbol{K}}_{ij} \boldsymbol{V}_{,j}^h + \boldsymbol{W}^h \cdot \tilde{\boldsymbol{C}} \boldsymbol{V}^h) \,\mathrm{d}Q \\
&\qquad + \int_{\Omega} (\boldsymbol{W}^h(t_{n+1}^-) \cdot \boldsymbol{U}(\boldsymbol{V}^h(t_{n+1}^-)) - \boldsymbol{W}^h(t_n^+) \cdot \boldsymbol{U}(\boldsymbol{V}^h(t_n^-))) \,\mathrm{d}\Omega
\end{aligned}$$

$$+ \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} (\mathscr{L}\boldsymbol{W}^h) \cdot \boldsymbol{\tau}(\mathscr{L}\boldsymbol{V}^h)\, \mathrm{d}Q + \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \nu^h \widetilde{\nabla}_\xi \boldsymbol{W}^h \cdot \left[\begin{array}{c} \nwarrow \\ \tilde{A}_0 \\ \searrow \end{array} \right] \widetilde{\nabla}_\xi \boldsymbol{V}^h \, \mathrm{d}Q$$

$$+ \int_{P_n} \boldsymbol{W}^h \cdot \boldsymbol{F}_i(\boldsymbol{V}^h) n_i \, \mathrm{d}P \,. \tag{5.43}$$

Considering the above equation as an *auxiliary* equation to the *original* variational equation (3.8), we can state the consistent boundary-flux calculation technique: Given $\boldsymbol{V}^h \in \mathscr{S}_n^h$, find $\boldsymbol{f}^h \in (\mathscr{S}_f^h)_n$ such that for all $\boldsymbol{W}^h \in \mathscr{V}_n^h$ the auxiliary equation (5.43) is satisfied. $\mathscr{S}_n^h$ is the finite dimensional space of $\boldsymbol{V}^h$'s as defined in (3.4); $\mathscr{V}_n^h$ is the finite dimensional space of $\boldsymbol{W}^h$'s as defined in (3.5) with one modification: $\boldsymbol{q}'(\boldsymbol{W}^h) \neq \boldsymbol{0}$ on $(P_f)_n$, where $\boldsymbol{q}'$ is the set of $\boldsymbol{W}$ constraint equations defined in (5.20); and $(\mathscr{S}_f^h)_n$ is the discrete space of $\boldsymbol{f}^h$ defined as

$$(\mathscr{S}_f^h)_n = \{ \boldsymbol{f}^h \mid \boldsymbol{f}^h \in (\mathrm{C}^0(P_f)_n)^m, \boldsymbol{f}^h|_{\bar{Q}_n^e \cap (P_f)_n} \in (\mathscr{P}_k(\bar{Q}_n^e \cap (P_f)_n))^m \} \,, \tag{5.44}$$

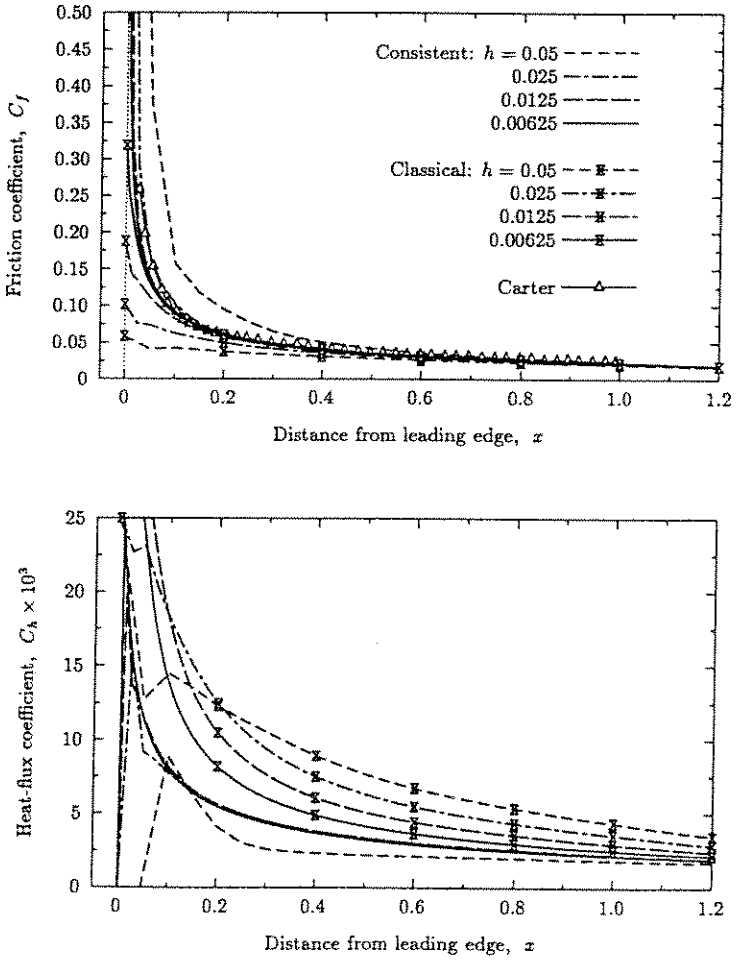


Fig. 5.3.1. Flow over a flat plate. Wall quantities computed via classical and consistent schemes.

where C^0 is the space of continuous functions; and $\mathscr{P}_k$ is the space of kth-order interpolation polynomials.

The auxiliary problem, (5.43), is solved for f^h after the original problem, (3.8), is solved for $\boldsymbol{V}^h$. This auxiliary problem is inexpensive compared to the original problem. Due to the local behavior of the discrete spaces, the integrals in (5.43) need to be performed only near the flux boundaries, and usually only after the last time step of the problem is completed. Moreover, when the nodal quadrature rule is used to evaluate the left-hand-side integral of (5.43), the unknown nodal fluxes decouple. This eliminates the need to solve a linear system of equations. In addition, the right-hand side of (5.43) is similar to the right-hand side of the original problem. Therefore, minimal coding is required to implement this technique. This is referred to as the consistent flux procedure.

5.3.2. Numerical example: Flow over a flat plate

The flow over a flat plate problem (introduced in Section 4.3.1) is used to evaluate the performance of the consistent boundary flux calculation. For the four finest meshes of Section 4.3.1, the skin friction and heat-flux coefficients computed via classical and consistent schemes are presented in Fig. 5.3.1. Here, the skin friction coefficient is defined as $C_f = \tau_{\text{wall}} / \frac{1}{2}\rho_\infty u_\infty^2$; and the heat-flux coefficient is defined as $C_h = -q_{\text{wall}} / \frac{1}{2}\rho_\infty u_\infty^3$. Note that (i) In the case of the skin friction the solution of the consistent scheme converges from above whereas the solution of the classical scheme converges from below; (ii) the consistent scheme converges faster (this allows the use of a coarser mesh if wall quantities are the primary interest); (iii) the consistent scheme captures the singularity in skin friction considerably better than the classical method; and (iv) the consistent scheme calculates the nearly-zero values of the heat-flux more accurately.

6. Numerical examples

In this section, we present numerical examples to demonstrate the performance and range of applicability of the method described in the previous sections. In particular, we present supersonic and hypersonic viscous flow over compression corners, and transient nearly-incompressible flow past a circular cylinder. Other calculations can be found in [15, 36–38].

All computations were performed on a CONVEX-C1 in double precision (64 bits per floating point), using the implicit algorithm with the GMRES iterative solver, and 2×2 Gaussian quadrature for bilinear elements.

6.1. Compression corner – Mach 3

In this Navier–Stokes problem, a Mach 3 flow passes over a compression corner at an angle of 10°; see Fig. 6.1.1. The Reynolds number, based on the free-stream values and the distance from the leading edge of the plate to the corner, is 16,800. The Sutherland viscosity law, $\mu = 0.0906\ \theta^{1.5}/(\theta + 0.0001406)$, is employed. This corresponds to a free-stream temperature of 216.7 K.

The computational domain covers the area $-0.2 \leq x \leq 1.8$, $0 \leq y \leq 0.575$ on the plate, and a height of 0.575 above the wall past the corner. The leading edge of the plate is placed at $x = 0$

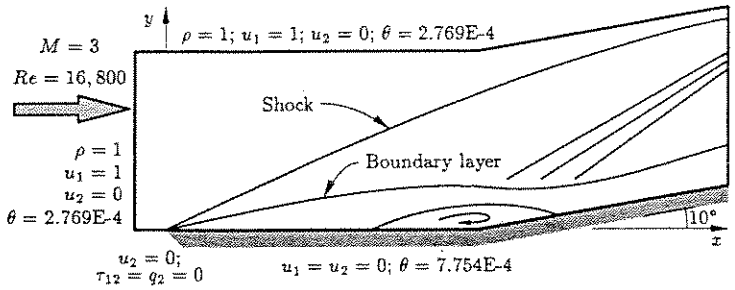


Fig. 6.1.1. Mach 3 compression corner problem.

and the corner at $x = 1$. On the inflow and top boundaries, ρ, u_1, u_2 and θ are prescribed; on the line of symmetry ($y = 0$ and $x < 0$), the symmetry condition, $u_2 = \tau_{12} = q_2 = 0$, is imposed; on the plate, the no-slip condition, $u_1 = u_2 = 0$, and the stagnation temperature, $\theta_{\text{stag}} = \theta_\infty(1 + \frac{1}{2}(\gamma - 1)M_\infty^2)$, are prescribed; while no condition is imposed on the outflow boundary.

To solve this problem, two bilinear-element meshes were employed. The first mesh, shown in Fig. 6.1.2(a), consists of 104×39 (=4,056) elements with $\Delta y_{\min} = 6.47 \times 10^{-4}$ and a maximum aspect ratio of 26. The second mesh, shown in Fig. 6.1.2(b), consists of 4,095 elements with $\Delta y_{\min} = 3.25 \times 10^{-4}$ and a maximum aspect ratio of 51.

The computed flow field using the first mesh is presented in Fig. 6.1.3. This figure demonstrates the ability of the method to capture the leading edge shock, the boundary layer which separates from the wall ahead of the corner and reattaches after the corner, resulting in a region of reverse flow by the corner and a compression fan at the reattachment point. (See [39] for a description of the physics of the problem.) Note that the leading edge shock crosses the upper boundary near the outflow. Its effect, however, does not propagate downward to disturb the flow near the wall.

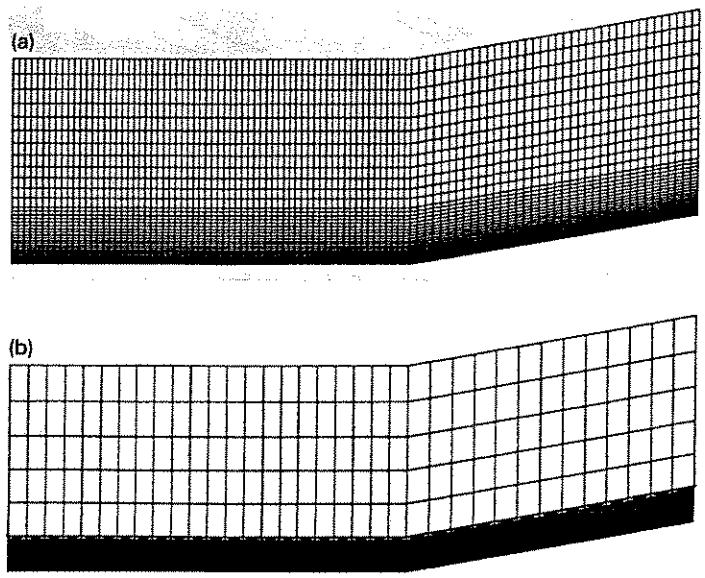


Fig. 6.1.2. Mach 3 compression corner. Finite element meshes: (a) 4,056 elements; (b) 4,095 elements.

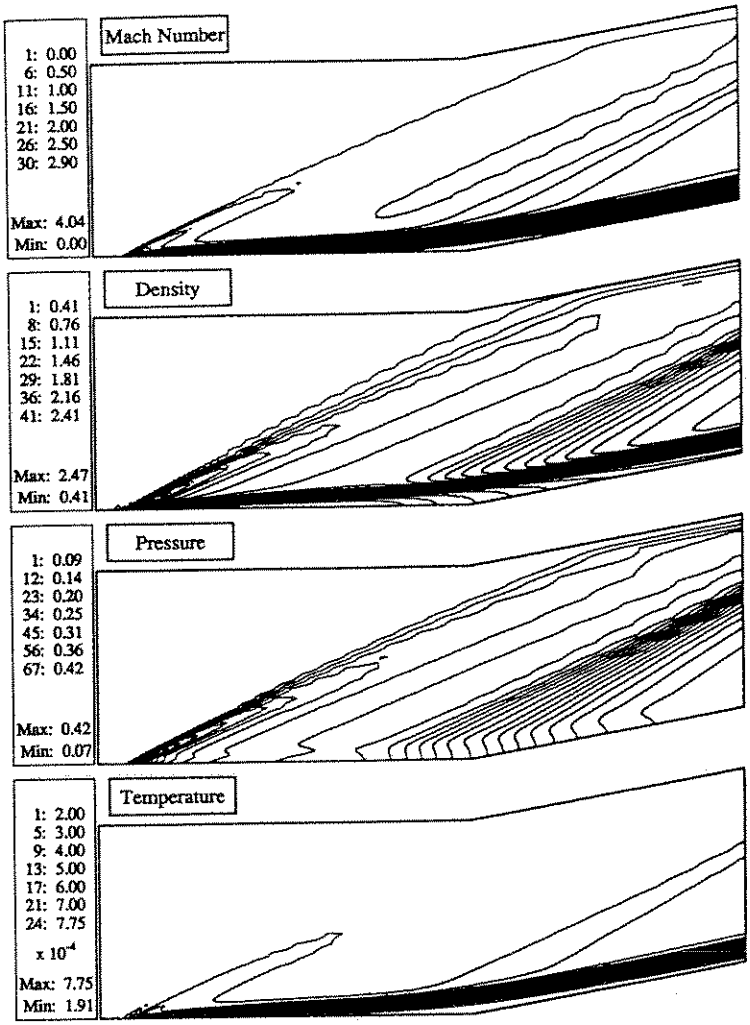
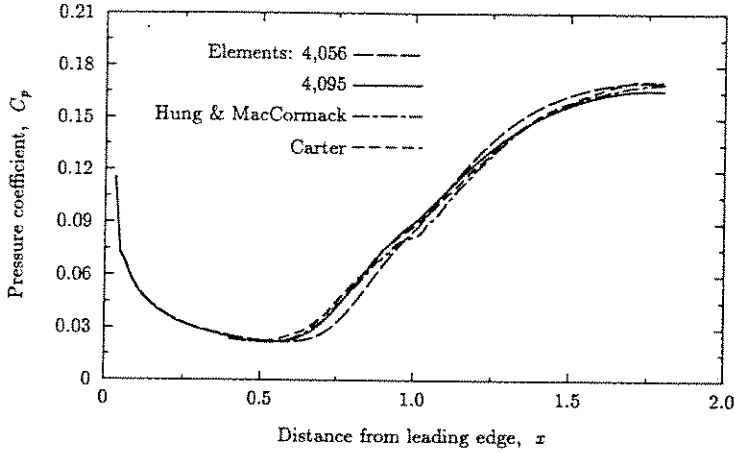


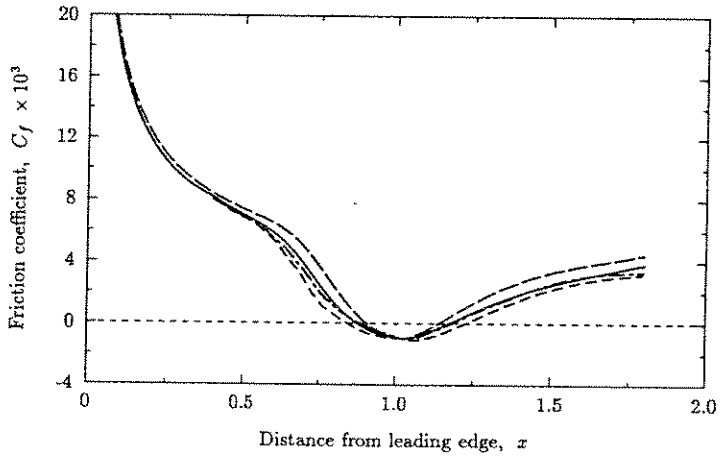
Fig. 6.1.3. Solution of the Mach 3 compression corner problem (4,056 elements).

The pressure, skin-friction and heat-flux coefficients along the wall are shown in Fig. 6.1.4. The heat-flux coefficient is defined as $C_h = q_{\text{wall}} / \frac{1}{2}\rho_\infty u_\infty^3$. The results show very good agreement with the solutions of Hung and MacCormack [39] and Carter [40]. The extent of the recirculation region is presented in Table 6.1.1.

6.2. Compression Corner – Mach 11.68

This problem, known as Holden's problem, consists of a Mach 11.68, Reynolds number 248,600 flow over a compression corner at an angle of 15°; see Fig. 6.2.1. The Reynolds number is defined based on the free-stream values and the distance from the leading edge of the plate to the corner. The Sutherland viscosity law, $\mu = 0.04428\ \theta^{1.5}/(\theta + 3.113 \times 10^{-5})$, is employed. This corresponds to a reference temperature of 505.2 K and a free-stream temperature of 64.6 K.





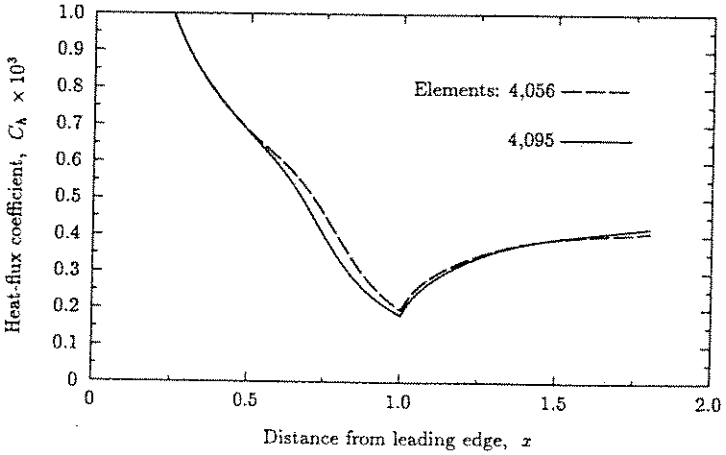


Fig. 6.1.4. Mach 3 compression corner. Wall quantities.

Table 6.1.1
Mach 3 compression corner. Extent of the recirculation region

Calculation	Separation point	Reattachment point
4,095-element	0.88	1.17
Hung and MacCormack	0.89	1.18
Carter	0.84	1.22

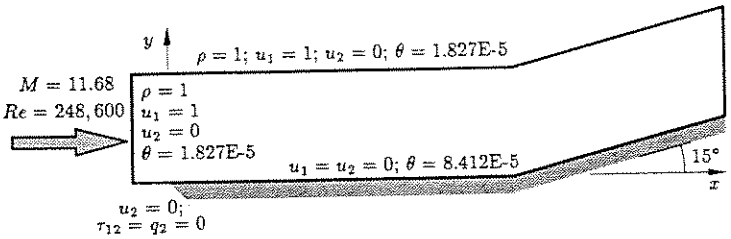


Fig. 6.2.1. Mach 11.68 compression corner problem.

The computational domain covers the area $-1.74 \leq x \leq 28$, $0 \leq y \leq 5.5$ on the plate, and is adjusted accordingly past the corner. The leading edge of the plate is placed at $x = 0$ and the corner at $x = 17.4$. The boundary conditions are similar to the Mach 3 compression corner problem presented in the previous section. Here, however, the wall temperature is maintained (cooled) at 8.412×10^{-5}.

To solve this problem, two meshes with bilinear elements were employed; see Fig. 6.2.2. These meshes consist of 4,056 and 14,144 elements distributed parabolically in y with $\Delta y_{\min} = 0.00562$ and 0.00125. The domain of the finest mesh extends only to the height of 4.5 above the wall.

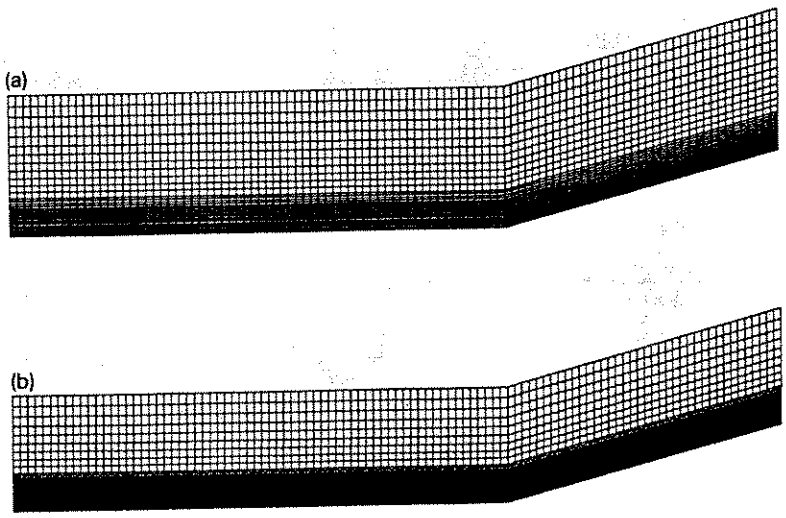


Fig. 6.2.2. Mach 11.68 compression corner. Finite element meshes: (a) 4,056 elements; (b) 14,144 elements.

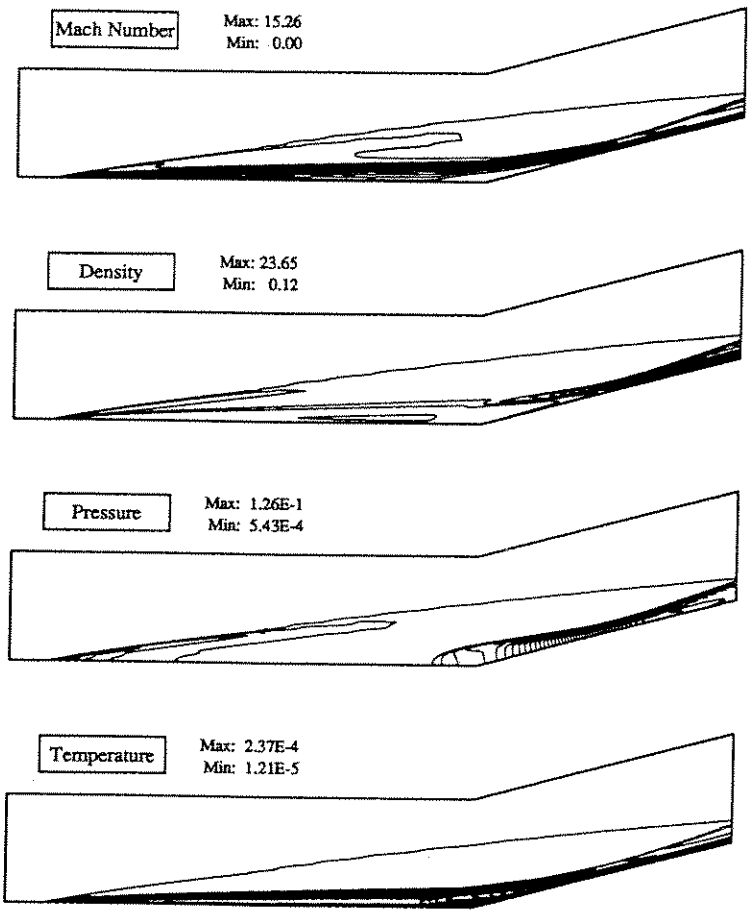


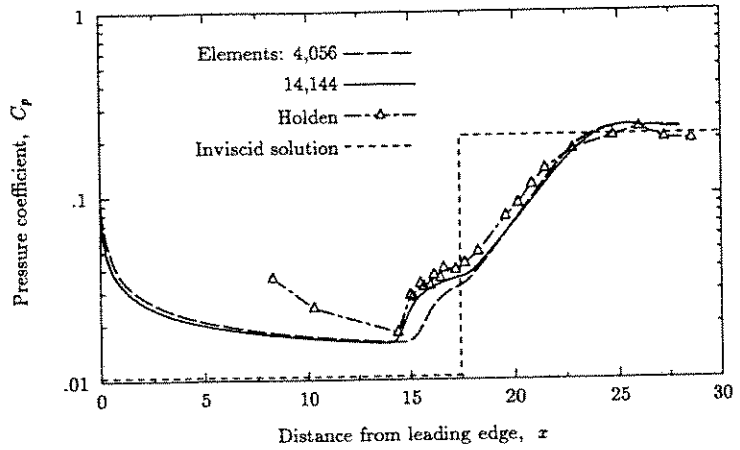
Fig. 6.2.3. Solution of the Mach 11.68 compression corner problem (14,144 elements).

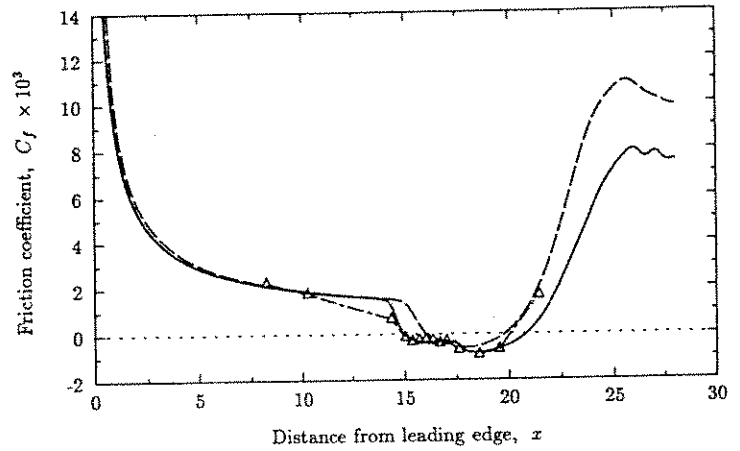
The computed flow field using the finest mesh is presented in Fig. 6.2.3. The wall quantities are compared with the experimental data of Holden [41] in Fig. 6.2.4. In this figure, the pressure coefficient is defined as $C_p = p / \frac{1}{2}\rho_\infty u_\infty^2$.

6.3. *Flow past a circular cylinder – Nearly-incompressible*

This two-dimensional problem is chosen to demonstrate the method's ability in modeling flow instabilities at low Mach numbers. At time $t = 0$, a circular cylinder is introduced in a uniform flow at Mach 0.01 and Reynolds number 100; see Fig. 6.3.1. The Reynolds number is measured using the uniform flow data and the diameter of the cylinder. The viscosity is assumed constant.

The computational domain encompasses $-4.5 \leq x \leq 15.5$, $-4.5 \leq y \leq 4.5$, with the circular cylinder having a diameter, $D = 1$, centered at $x = y = 0$. On the inflow boundary ($x = -4.5$), ρ, u_1, u_2 and θ are prescribed. (Although well-posedness requires us to prescribe only three quantities on this subsonic boundary, experience has shown that the additional quantity helps with the stability.) On the top and bottom boundaries ($y = \pm 4.5$), the symmetry condition,





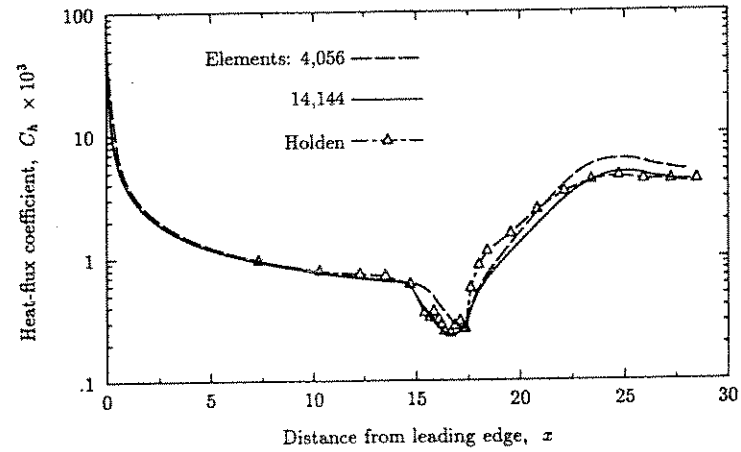


Fig. 6.2.4. Mach 11.68 compression corner. Wall quantities.

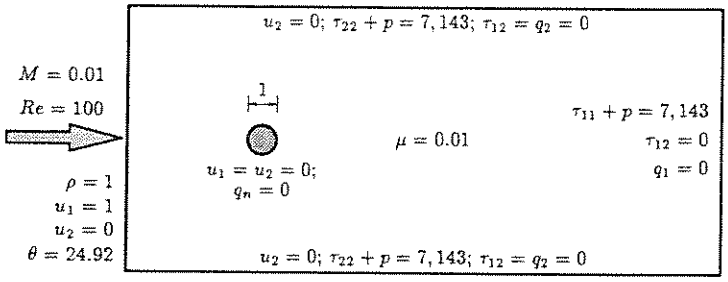


Fig. 6.3.1. Flow past a circular cylinder problem.

$u_2 = \tau_{12} = q_2 = 0$, with the prescribed total normal stress, σ_{22}, are imposed. On the outflow boundary ($x = 15.5$), zero traction and heat-flux, $\tau_{12} = q_2 = 0$, and the total normal stress, σ_{11}, are prescribed. On the cylinder, the no-slip condition, $u_1 = u_2 = 0$, and the no heat-flux condition, $q_n = 0$, are imposed.

To solve this problem, we employed a mesh consisting of 4,936 bilinear elements with 5,063 nodes; see Fig. 6.3.2. In addition, we employed the third-order predictor multi-corrector algorithm (cf. Box. 4.2.1) with four corrector passes using the Galerkin/least-squares method without any discontinuity-capturing operator. We employed a constant time increment of 0.1,

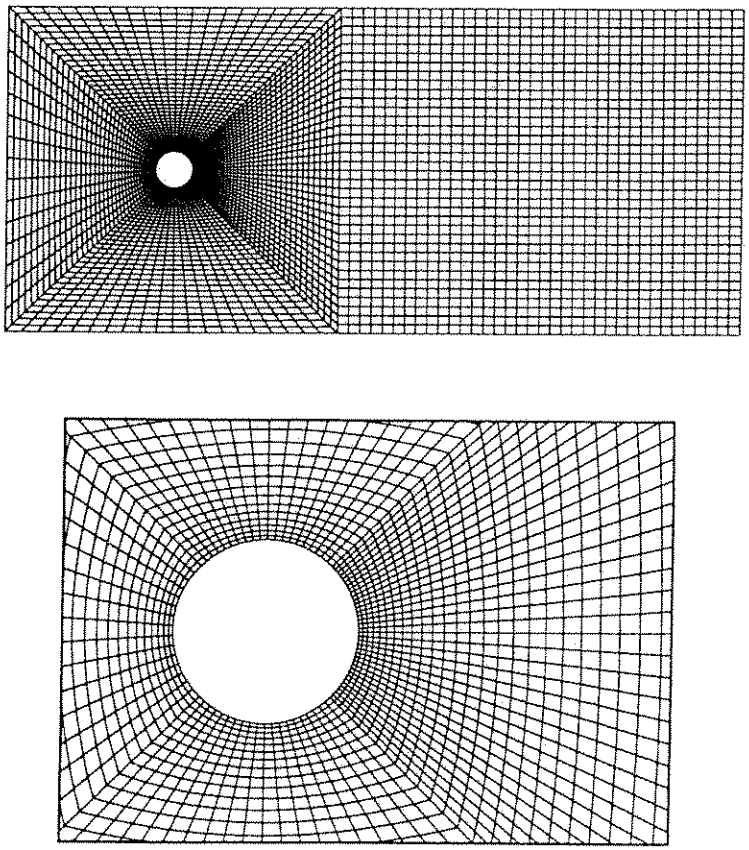


Fig. 6.3.2. Flow past a circular cylinder. Finite element mesh (4,936 elements; 5,063 nodes).

which yielded a maximum algorithmic Courant number (defined in (4.33)) of 656, at the initial time step.

Figure 6.3.3 shows the temporal development of the stationary streamlines. First, a symmetric pair of eddies appears behind the cylinder. This becomes unstable, leading to a periodic vortex shedding, known as the Von Karman vortex street. Figure 6.3.4 shows the temporal development of the drag coefficient, $C_D = \int_0^{2\pi} \sigma_{1n}\, d\phi / \pi \rho_\infty u_\infty^2$, and the lift coefficient, $C_L = \int_0^{2\pi} \sigma_{2n}\, d\phi / \pi \rho_\infty u_\infty^2$. Note the sensitivity of the lift coefficient in detecting the periodicity of the flow. A shedding period, T, of 6.4 was observed, yielding a Strouhal number ($\mathrm{St} = D/u_\infty T$) of 0.156. This value is 6.3% lower than the commonly observed value of one sixth; see, e.g., [42, 43].

We have attempted to solve the problem using a coarser mesh with 1,436 elements (employed by Brooks and Hughes [42]); but we were unsuccessful in capturing the shedding

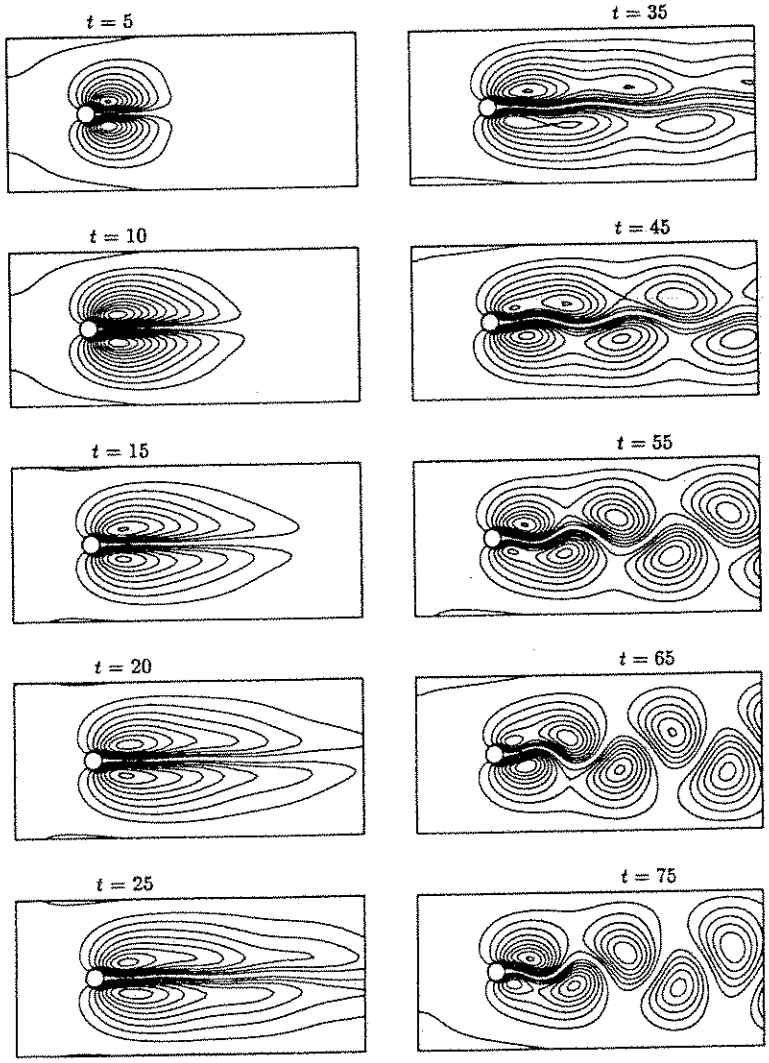


Fig. 6.3.3. Flow past a circular cylinder. Stationary streamlines.

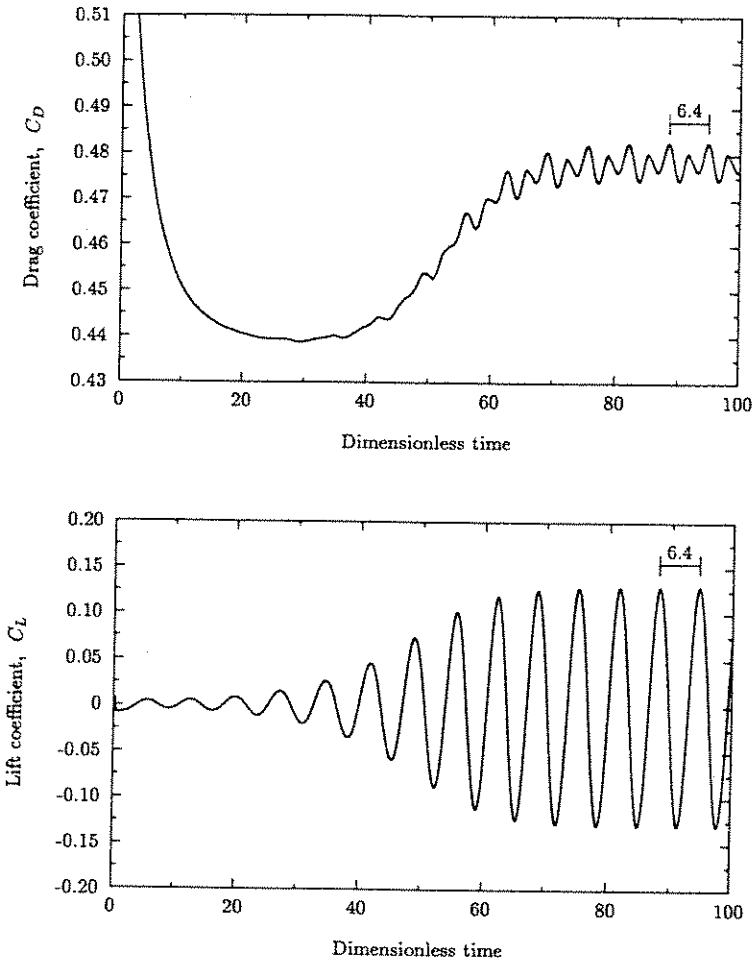


Fig. 6.3.4. Flow past a circular cylinder. Development of drag and lift coefficients.

phenomena, even for a Reynolds number of 10^3, starting from a shedding solution and using a very small time increment. The solution of this problem is sensitive to the spatial refinement of the mesh, but it is fairly insensitive to the temporal refinement. Using the mesh in Fig. 6.3.2, we also solved the problem for a Reynolds number of 10^3, which yielded a Strouhal number of 0.217.

7. Conclusions

The goal of this research was to develop an efficient finite element method capable of solving a wide spectrum of compressible flow problems. To achieve this goal, we have employed the symmetric form of the compressible Navier–Stokes equations, designed a space–time Galerkin/least-squares variational formulation including a discontinuity-capturing operator, developed predictor multi-corrector algorithms for steady as well as unsteady

problems, addressed the issues concerning the implementation of boundary conditions, and evaluated the performance of the method on a variety of two-dimensional problems. The development was accompanied by mathematical and numerical analyses. Although the underlying objective was to solve the compressible Navier–Stokes equations, most techniques and procedures presented are equally applicable to other physical problems of interest.

The symmetric form of the compressible Navier–Stokes equations was obtained by expressing the equations in terms of the physical entropy variables, as in [1]. The Galerkin formulation of the symmetric form automatically satisfies the second law of thermodynamics. Use of the entropy variables in finite element formulations ensures dimensional consistency, which was previously obtained by subjective nondimensionalizations. In addition, the symmetry of the differential equations facilitates use of many basic techniques in finite element programs, such as block-diagonal pre-preconditioning.

The space–time Galerkin/least-squares variational formulation generates complete space–time finite element discretizations, permitting discrete discontinuities in time. The least-squares operator has emerged from the SUPG operator of Hughes and Mallet [5]. Both of these operators compensate for the lack of stability of the basic Galerkin method for advection-dominated problems; while maintaining the accuracy of the Galerkin method for smooth solutions. The least-squares operator, however, has a conceptually simpler and more general construction, applicable to a wider class of problems. The space–time Galerkin/least-squares method is observed to be an excellent method for smooth solutions of the advective-diffusive systems in the entire range of advective-diffusive parameters. We believe that up to the definition of τ, this is the canonical method to employ.

We have proposed a new definition for τ which has certain advantages over the original definition given in [5]: The new τ directly accounts for the contribution of diffusion and circumvents solving an eigenvalue problem; it also accommodates source terms that are proportional to the solution; it has a simple and general structure which can extend to other problem classes; and it facilitates mathematical analysis of the Galerkin/least-squares method. Nevertheless, further research into simpler definitions is still warranted.

The discontinuity-capturing operator is added to the Galerkin/least-squares method to control unresolved internal and boundary layers. This nonlinear operator acts in the direction of gradients to control the gradients and is proportional to the residual $\mathscr{L}V^h$ to ensure variational consistency and to diminish automatically in regions where $\mathscr{L}V^h$ is small (i.e., where the solution is smooth). Numerical solutions of Euler flows with shocks have shown that while the Galerkin/least-squares method well resolves the flow in the smooth regions and confines the overshoots and undershoots caused by the shock to a small region near the shock, the discontinuity-capturing operator effectively controls oscillations near the shock. We have proposed two discontinuity-capturing operators which are very effective in practice. These operators can be considered as two members of a class of discontinuity-capturing operators, which require further investigation.

The implicit/explicit predictor multi-corrector algorithm emanating from the constant-in-time approximation of the space–time Galerkin/least-squares variational equation has low order of time accuracy, but has good stability properties and is computationally efficient. This algorithm is well-suited for solving steady problems. Numerical solutions on unstructured meshes have shown that the convergence of this algorithm can be significantly improved through the use of the local-time-stepping strategy. The linear-in-time approximation of the

space–time Galerkin/least-squares variational equation leads to an algebraic system having twice as many equations and unknowns as in the constant-in-time approximation. We have presented a technique to transform and reduce this system to two weakly-coupled subsystems, leading to a predictor multi-corrector algorithm which can employ the same left-hand side matrix for both subsystems. This fairly efficient algorithm has a higher order of time accuracy than the constant-in-time algorithm which makes it attractive for solving unsteady problems. The solution of the flow past a circular cylinder has shown that the time accuracy of this algorithm is more than sufficient to capture flow instabilities.

We have presented a consistent technique for treating nonlinear essential boundary conditions within the framework of predictor multi-corrector algorithms. We have also studied the boundary integral residing in the space–time variational equation. This boundary integral yields the following natural boundary conditions: the normal mass flux, pressure, normal viscous flux and heat flux boundary conditions. The boundary integral also provides a consistent method for calculating boundary fluxes. The convergence of this method is faster than the classical method; it also captures singularities and nearly-zero values more accurately.

The numerical examples presented illustrate the performance and range of applicability of the method developed. In particular, they have substantiated the ability of the method to solve steady high-speed (as high as Mach 11.68) flows with shocks, boundary and shear layers, and recirculations, as well as unsteady nearly-incompressible (Mach 0.01) flows with instabilities and periodic vortex sheddings. A number of other numerical simulations are presented in Shakib et al. [15].

Acknowledgment

The authors would like to express their appreciation to Greg Hulbert, Michel Mallet and Arthur Raefsky for helpful comments.

This research was supported by a fellowship from the IBM Corporation, the IBM Almaden Research Center under Grant No. 604912, the NASA Langley Research Center under Grant NASA-NAG-1-361, and Avions Marcel Dassault–Breguet Aviation, St. Cloud, France.

Appendix A. Flux vectors and coefficient matrices

In this appendix we present the flux vectors and the coefficient matrices of the compressible Navier–Stokes equations, as expressed in terms of the (physical) entropy variables.

For convenience, the mapping from $\boldsymbol{U}$ to $\boldsymbol{V}$, (2.15)–(2.17), is provided here:

$$\boldsymbol{V}=\begin{Bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{Bmatrix} = \frac{1}{\rho\iota} \begin{Bmatrix} -U_5 + \rho\iota(\gamma + 1 - s + s_0) \\ U_2 \\ U_3 \\ U_4 \\ -U_1 \end{Bmatrix}, \tag{A.1}$$

where

$$s = \ln((\gamma - 1)\rho\iota / U_1^{\gamma}) , \tag{A.2}$$

$$\rho\iota = U_5 - (U_2^2 + U_3^2 + U_4^2)/2U_1 . \tag{A.3}$$

The inverse mapping $V \mapsto U$, (2.18)–(2.20), is

$$U = \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{Bmatrix} = \rho \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e \end{Bmatrix} = \rho\iota \begin{Bmatrix} -V_5 \\ V_2 \\ V_3 \\ V_4 \\ 1 - (V_2^2 + V_3^2 + V_4^2)/2V_5 \end{Bmatrix}, \tag{A.4}$$

where

$$\rho\iota = ((\gamma - 1)/(-V_5)^{\gamma})^{1/(\gamma-1)} \exp(-s + s_0)/(\gamma - 1) , \tag{A.5}$$

$$s = \gamma - V_1 + (V_2^2 + V_3^2 + V_4^2)/2V_5 . \tag{A.6}$$

To be compatible with the notations used by Hughes et al. [6], we express the flux vectors and coefficient matrices with the help of the following variables:

$$\begin{aligned}
&\bar{\gamma} = \gamma - 1 , && k_1 = (V_2^2 + V_3^2 + V_4^2)/2V_5 , && k_2 = k_1 - \gamma , \\
&k_3 = k_1^2 - 2\gamma k_1 + \gamma , && k_4 = k_2 - \bar{\gamma} , && k_5 = k_2^2 - \bar{\gamma}(k_1 + k_2) , \\
&c_1 = \bar{\gamma}V_5 - V_2^2 , && d_1 = -V_2V_3 , && e_1 = V_2V_5 , \\
&c_2 = \bar{\gamma}V_5 - V_3^2 , && d_2 = -V_2V_4 , && e_2 = V_3V_5 , \\
&c_3 = \bar{\gamma}V_5 - V_4^2 , && d_3 = -V_3V_4 , && e_3 = V_4V_5 .
\end{aligned} \tag{A.7}$$

The Riemannian metric tensor $\tilde{A}_0 = U_{,V}$ and its inverse $\tilde{A}_0^{-1} = V_{,U}$ can be written as

$$\tilde{A}_0 = \frac{\rho\iota}{\bar{\gamma}V_5} \begin{bmatrix} -V_5^2 & e_1 & e_2 & e_3 & V_5(1 - k_1) \\ & c_1 & d_1 & d_2 & V_2k_2 \\ & & c_2 & d_3 & V_3k_2 \\ & \text{symm} & & c_3 & V_4k_2 \\ & & & & -k_3 \end{bmatrix} \tag{A.8}$$

and

$$\tilde{A}_0^{-1} = \frac{-1}{\rho\iota V_5} \begin{bmatrix} k_1^2 + \gamma & k_1V_2 & k_1V_3 & k_1V_4 & (k_1 + 1)V_5 \\ & V_2^2 - V_5 & -d_1 & -d_2 & e_1 \\ & & V_3^2 - V_5 & -d_3 & e_2 \\ & \text{symm} & & V_4^2 - V_5 & e_3 \\ & & & & V_5^2 \end{bmatrix} . \tag{A.9}$$

The Euler fluxes, (2.3), in terms of the primitive variables are

$$\boldsymbol{F}_i = \rho u_i \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e \end{Bmatrix} + p \begin{Bmatrix} 0 \\ \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ u_i \end{Bmatrix}. \tag{A.10}$$

In terms of the V-variables, the Euler fluxes can be expressed as

$$\boldsymbol{F}_1 = \frac{\rho\iota}{V_5} \begin{Bmatrix} e_1 \\ c_1 \\ d_1 \\ d_2 \\ k_2 V_2 \end{Bmatrix}, \qquad \boldsymbol{F}_2 = \frac{\rho\iota}{V_5} \begin{Bmatrix} e_2 \\ d_1 \\ c_2 \\ d_3 \\ k_2 V_3 \end{Bmatrix}, \qquad \boldsymbol{F}_3 = \frac{\rho\iota}{V_5} \begin{Bmatrix} e_3 \\ d_2 \\ d_3 \\ c_3 \\ k_2 V_4 \end{Bmatrix}. \tag{A.11}$$

The Jacobians of the Euler fluxes, $\tilde{\boldsymbol{A}}_i = \boldsymbol{F}_{i,V} = \boldsymbol{A}_i \tilde{\boldsymbol{A}}_0$, are given by

$$\tilde{\boldsymbol{A}}_1 = \frac{\rho\iota}{\bar{\gamma} V_5^2} \begin{bmatrix} e_1 V_5 & c_1 V_5 & d_1 V_5 & d_2 V_5 & k_2 e_1 \\ & -(c_1 + 2\bar{\gamma} V_5) V_2 & -c_1 V_3 & -c_1 V_4 & c_1 k_2 + \bar{\gamma} V_2^2 \\ & & -c_2 V_2 & -d_1 V_4 & k_4 d_1 \\ & \text{symm} & & -c_3 V_2 & k_4 d_2 \\ & & & & k_5 V_2 \end{bmatrix}, \tag{A.12}$$

$$\tilde{\boldsymbol{A}}_2 = \frac{\rho\iota}{\bar{\gamma} V_5^2} \begin{bmatrix} e_2 V_5 & d_1 V_5 & c_2 V_5 & d_3 V_5 & k_2 e_2 \\ & -c_1 V_3 & -c_2 V_2 & -d_1 V_4 & k_4 d_1 \\ & & -(c_2 + 2\bar{\gamma} V_5) V_3 & -c_2 V_4 & c_2 k_2 + \bar{\gamma} V_3^2 \\ & \text{symm} & & -c_3 V_3 & k_4 d_3 \\ & & & & k_5 V_3 \end{bmatrix}, \tag{A.13}$$

$$\tilde{\boldsymbol{A}}_3 = \frac{\rho\iota}{\bar{\gamma} V_5^2} \begin{bmatrix} e_3 V_5 & d_2 V_5 & d_3 V_5 & c_3 V_5 & k_2 e_3 \\ & -c_1 V_4 & -d_2 V_3 & -c_3 V_2 & k_4 d_2 \\ & & -c_2 V_4 & -c_3 V_3 & k_4 d_3 \\ & \text{symm} & & -(c_3 + 2\bar{\gamma} V_5) V_4 & c_3 k_2 + \bar{\gamma} V_4^2 \\ & & & & k_5 V_4 \end{bmatrix}. \tag{A.14}$$

The diffusive fluxes, (2.4), are given as

$$\boldsymbol{F}_i^{\mathrm{d}} = \begin{Bmatrix} 0 \\ \tau_{1i} \\ \tau_{2i} \\ \tau_{3i} \\ \tau_{ij} u_j \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -q_i \end{Bmatrix}, \tag{A.15}$$

where

$$\tau_{ij} = \lambda u_{k,k} \delta_{ij} + \mu (u_{i,j} + u_{j,i}), \tag{A.16}$$

$$q_i = -\kappa \theta_{,i}. \tag{A.17}$$

The velocity components and temperature can be written in terms of the V-variables as

$$u_i = -V_{i+1}/V_5 \quad \text{for } i = 1, 2, 3 , \tag{A.18}$$

$$\theta = -1/c_v V_5 . \tag{A.19}$$

The spatial gradients of these variables yield

$$u_{i,j} = (-V_5 V_{i+1,j} + V_{i+1} V_{5,j})/V_5^2 , \tag{A.20}$$

$$\theta_{,i} = (1/c_v V_5^2) V_{5,i} . \tag{A.21}$$

The coefficient of thermal conductivity, κ, can be expressed as

$$\kappa = \gamma c_v \mu / \mathrm{Pr} , \tag{A.22}$$

where Pr is the Prandtl number. Furthermore, let

$$\chi = \lambda + 2\mu . \tag{A.23}$$

With the above relations, the diffusivity coefficient-matrices, $\tilde{\boldsymbol{K}}_{ij} \boldsymbol{V}_{,j} = \boldsymbol{F}_i^{\mathrm{d}}$, in terms of the $\boldsymbol{V}$-variables are

$$\tilde{\boldsymbol{K}}_{11} = \frac{1}{V_5^3} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -\chi V_5^2 & 0 & 0 & \chi e_1 \\ 0 & 0 & -\mu V_5^2 & 0 & \mu e_2 \\ 0 & 0 & 0 & -\mu V_5^2 & \mu e_3 \\ 0 & \chi e_1 & \mu e_2 & \mu e_3 & -\chi V_2^2 - \mu(V_3^2 + V_4^2) + \gamma\mu V_5/\mathrm{Pr} \end{bmatrix} , \tag{A.24}$$

$$\tilde{\boldsymbol{K}}_{22} = \frac{1}{V_5^3} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -\mu V_5^2 & 0 & 0 & \mu e_1 \\ 0 & 0 & -\chi V_5^2 & 0 & \chi e_2 \\ 0 & 0 & 0 & -\mu V_5^2 & \mu e_3 \\ 0 & \mu e_1 & \chi e_2 & \mu e_3 & -\chi V_3^2 - \mu(V_2^2 + V_4^2) + \gamma\mu V_5/\mathrm{Pr} \end{bmatrix} , \tag{A.25}$$

$$\tilde{\boldsymbol{K}}_{33} = \frac{1}{V_5^3} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -\mu V_5^2 & 0 & 0 & \mu e_1 \\ 0 & 0 & -\mu V_5^2 & 0 & \mu e_2 \\ 0 & 0 & 0 & -\chi V_5^2 & \chi e_3 \\ 0 & \mu e_1 & \mu e_2 & \chi e_3 & -\chi V_4^2 - \mu(V_2^2 + V_3^2) + \gamma\mu V_5/\mathrm{Pr} \end{bmatrix} , \tag{A.26}$$

$$\tilde{K}_{12} = \tilde{K}_{21}^{t} = \frac{1}{V_5^3}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\lambda V_5^2 & 0 & \lambda e_2 \\ 0 & -\mu V_5^2 & 0 & 0 & \mu e_1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \mu e_2 & \lambda e_1 & 0 & (\lambda+\mu) d_1 \end{bmatrix}, \tag{A.27}$$

$$\tilde{K}_{13} = \tilde{K}_{31}^{t} = \frac{1}{V_5^3}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\lambda V_5^2 & \lambda e_3 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -\mu V_5^2 & 0 & 0 & \mu e_1 \\ 0 & \mu e_3 & 0 & \lambda e_1 & (\lambda+\mu) d_2 \end{bmatrix}, \tag{A.28}$$

$$\tilde{K}_{23} = \tilde{K}_{32}^{t} = \frac{1}{V_5^3}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\lambda V_5^2 & \lambda e_3 \\ 0 & 0 & -\mu V_5^2 & 0 & \mu e_2 \\ 0 & 0 & \mu e_3 & \lambda e_2 & (\lambda+\mu) d_3 \end{bmatrix}. \tag{A.29}$$

The source vector, (2.5), in terms of the primitive variables is given by

$$\mathscr{F} = \rho \begin{Bmatrix} 0 \\ b_1 \\ b_2 \\ b_3 \\ b_i u_i + r \end{Bmatrix}. \tag{A.30}$$

In terms of the V-variables, this yields

$$\mathscr{F} = \rho\iota \begin{Bmatrix} 0 \\ -b_1 V_5 \\ -b_2 V_5 \\ -b_3 V_5 \\ b_i V_{i+1} - r V_5 \end{Bmatrix}. \tag{A.31}$$

The source coefficient-matrix, $\tilde{C}$ (where $\tilde{C}V = -\mathscr{F}$), is not uniquely defined. One possible definition, which leads to a symmetric matrix, is

$$\tilde{C} = \rho\iota \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & b_1 \\ & & 0 & 0 & b_2 \\ & & & 0 & b_3 \\ & \text{symm} & & & -2(b_1 V_2 + b_2 V_3 + b_3 V_4)/V_5 + r \end{bmatrix}. \tag{A.32}$$

Appendix B. Essential boundary condition transformation matrices

The transformation matrices used to incorporate the W and ΔV constraint equations in the predictor multi-corrector algorithms are given as follows:

The *density* transformation matrix is

$$S_1 = \begin{bmatrix} 0 & S_{12} & S_{13} & S_{14} & S_{15} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{B.1}$$

where

$$S_{12} = \frac{V_2}{V_5}\,; \quad S_{13} = \frac{V_3}{V_5}\,; \quad S_{14} = \frac{V_4}{V_5}\,; \quad S_{15} = \frac{1}{V_5} - \frac{V_2^2 + V_3^2 + V_4^2}{2V_5^2}\,. \tag{B.2}$$

The u_r *velocity*-component transformation matrix is

$$S_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & S_{23} & S_{24} & S_{25} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{B.3}$$

where

$$S_{23} = -C_2^r/C_1^r\,; \quad S_{24} = -C_3^r/C_1^r\,; \quad S_{25} = -g_2/C_1^r\,. \tag{B.4}$$

The u_s *velocity*-component transformation matrix is

$$S_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & S_{32} & 0 & S_{34} & S_{35} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{B.5}$$

where

$$S_{32} = -C_1^s/C_2^s\,; \quad S_{34} = -C_3^s/C_2^s\,; \quad S_{35} = -g_3/C_2^s\,. \tag{B.6}$$

The u_t *velocity*-component transformation matrix is

$$S_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & S_{42} & S_{43} & 0 & S_{45} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{B.7}$$

where

$$S_{42} = C_1^t/C_3^t\,; \quad S_{43} = C_2^t/C_3^t\,; \quad S_{45} = -g_4/C_3^t\,. \tag{B.8}$$

The *temperature* transformation matrix is

$$S_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{B.9}$$

The *pressure* transformation matrix is

$$S_6 = \begin{bmatrix} 0 & S_{12} & S_{13} & S_{14} & S_{15} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{B.10}$$

where

$$S_{12} = \frac{V_2}{V_5}\,; \quad S_{13} = \frac{V_3}{V_5}\,; \quad S_{14} = \frac{V_4}{V_5}\,; \quad S_{15} = \frac{\gamma}{V_5} - \frac{V_2^2 + V_3^2 + V_4^2}{2V_5^2}\,. \tag{B.11}$$

References

[1] T.J.R. Hughes, L.P. Franca and M. Mallet, A new finite element formulation for computational fluid dynamics: I. Symmetric forms of the compressible Euler and Navier–Stokes equations and the second law of thermodynamics, Comput. Methods Appl. Mech. Engrg. 54 (1986) 223–234.

[2] C. Johnson, U. Nävert and J. Pitkäranta, Finite element methods for linear hyperbolic problems, Comput. Methods Appl. Mech. Engrg. 45 (1984) 285–312.

[3] T.J.R. Hughes and F. Shakib, Computational aerodynamics and the finite element method, AIAA 26th Aerospace Sciences Meeting, Paper No. 88-0031, Reno, Nevada, January 1988.

[4] T.J.R. Hughes and G.M. Hulbert, Space–time finite element methods for elastodynamics: Formulations and error estimates, Comput. Methods Appl. Mech. Engrg. 66 (1988) 339–363.

SUPG [5] T.J.R. Hughes and M. Mallet, A new finite element formulation for computational fluid dynamics: III. The generalized streamline operator for multidimensional advective-diffusive systems, Comput. Methods Appl. Mech. Engrg. 58 (1986) 305–328.

[6] T.J.R. Hughes and M. Mallet, A new finite element formulation for computational fluid dynamics: IV. A discontinuity-capturing operator for multidimensional advective-diffusive systems, Comput. Methods Appl. Mech. Engrg. 58 (1986) 329–339.

[7] M. Mallet, A finite element method for computational fluid dynamics, PhD Thesis, Stanford University, 1985.

[8] T.J.R. Hughes and A.N. Brooks, A multidimensional upwind scheme with no crosswind diffusion, in: T.J.R. Hughes, ed., Finite Element Methods for Convection Dominated Flows, AMD Vol. 34 (ASME, New York, 1979) 19–35.

[9] T.J.R. Hughes and A.N. Brooks, A theoretical framework for Petrov–Galerkin methods with discontinuous weighting functions: Application to the streamline upwind procedure, in: R.H. Gallagher et al., eds., Finite Element in Fluids, Vol. 4 (Wiley, Chichester, 1982) 47–65.

[10] T.J.R. Hughes, Recent progress in the development and understanding of SUPG methods with special reference to the compressible Euler and Navier–Stokes equations, Internat. J. Numer. Methods Fluids 7 (1987) 1261–1275.

[11] T.J.R. Hughes, L.P. Franca and M. Mallet, A new finite element formulation for computational fluid

dynamics: VI. Convergence analysis of the generalized SUPG formulation for linear time-dependent multi-dimensional advective-diffusive systems, Comput. Methods Appl. Mech. Engrg. 63 (1987) 97–112.
[12] C. Johnson, A. Szepessy and P. Hansbo, On the convergence of shock-capturing streamline diffusion finite element methods for hyperbolic conservation laws, Technical Report 1987-21, Mathematics Department, Chalmers University of Technology, Göteborg, 1987.
[13] A. Szepessy, Convergence of a shock-capturing streamline diffusion finite element method for a scalar conservation law in two space dimensions, Technical Report 1988-07, Mathematics Department, Chalmers University of Technology, Göteborg, Sweden, 1988.
[14] F. Shakib and T.J.R. Hughes, A new finite element formulation for computational fluid dynamics: IX. Fourier analysis of space–time Galerkin/least-squares algorithms, Comput. Methods Appl. Mech. Engrg. 87 (1991) 35–58.
[15] F. Shakib, T.J.R. Hughes and Z. Johan, A multi-element group preconditioned GMRES algorithm for nonsymmetric systems arising in finite element analysis, Comput. Methods Appl. Mech. Engrg. 75 (1989) 415–456.
[16] R.F. Warming, R.M. Beam and B.J. Hyett, Diagonalization and simultaneous symmetrization of the gas-dynamics matrices, Math. Comput. 29 (132) (1975) 1037–1045.
[17] S.K. Godunov, The problem of a generalized solution in the theory of quasilinear equations and in gas dynamics, Russ. Math. Surveys 17 (1962) 145–156.
[18] M.S. Mock, Systems of conservation laws of mixed type, J. Differential Equations 37 (1980) 70–88.
[19] A. Harten, On the symmetric form of systems of conservation laws with entropy, J. Comput. Phys. 49 (1983) 151–164.
[20] E. Tadmor, Skew-selfadjoint forms for systems for conservation laws, J. Math. Anal. Appl. 103 (1984) 428–442.
[21] P.K. Dutt, Stable boundary conditions and difference schemes for Navier–Stokes type equations, PhD Thesis, University of California, Los Angeles, 1985.
[22] T.J.R. Hughes, M. Mallet and A. Mizukami, A new finite element formulation for computational fluid dynamics: II. Beyond SUPG, Comput. Methods Appl. Mech. Engrg. 54 (1986) 341–355.
[23] C. Johnson and A. Szepessy, A shock-capturing streamline diffusion finite element method for a nonlinear hyperbolic conservation law, Technical Report 1986-09, Mathematics Department, Chalmers University of Technology, Göteborg, Sweden, 1986.
[24] A.C. Galeão and E.G. Dutra do Carmo, A consistent approximate upwind Petrov–Galerkin method for convection-dominated problems, Comput. Methods Appl. Mech. Engrg. 68 (1988) 83–95.
[25] C.I. Bajer, Notes on the stability of non-rectangular space–time finite elements, Internat. J. Numer. Methods Engrg. 24 (1987) 1721–1739.
[26] T.J.R. Hughes, L.P. Franca and G.M. Hulbert, A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least-squares method for advective-diffusive equations, Comput. Methods Appl. Mech. Engrg. 73 (1989) 173–189.
[27] J.E. Marsden and T.J.R. Hughes, Mathematical Foundations of Elasticity (Prentice-Hall, Englewood Cliffs, NJ, 1983).
[28] G.M. Hulbert, Space–time finite element methods for second-order hyperbolic equations, PhD Thesis, Stanford University, 1989.
[29] C. Johnson and A. Szepessy, Convergence of a finite element method for a nonlinear hyperbolic conservation law, Technical Report 1985-25, Mathematics Department, Chalmers University of Technology, Göteborg, Sweden, 1985.
[30] C. Johnson and A. Szepessy, On the convergence of streamline diffusion finite element methods for hyperbolic conservation laws, in: T.E. Tezduyar and T.J.R. Hughes, eds., Numerical Methods for Compressible Flows – Finite Difference, Element and Volume Techniques, AMD Vol. 78 (ASME, New York, 1986) 75–91.
[31] S. Lang, Differential Manifolds (Addison-Wesley, Reading, MA, 1972).
[32] P.A. Thompson, Compressible-Fluid Dynamics (McGraw-Hill, New York, 1972).
[33] B. Gustafsson and A. Sundström, Incompletely parabolic problems in fluid dynamics, SIAM J. Appl. Math. 35 (2) (1978) 343–357.
[34] A. Mizukami, A mixed finite element method for boundary flux computation, Comput. Methods Appl. Mech. Engrg. 57 (1986) 239–243.

[35] E.A. Thornton, P. Dechaumphai and G. Vemaganti, A finite element approach for prediction of aerothermal loads, AIAA/ASME 4th Fluid Mechanics, Plasma Dynamics and Lasers Conference, Paper No. 86-1050, Atlanta, Georgia, 12–14 May 1986.

[36] L.P. Franca, I. Harari, T.J.R. Hughes, M. Mallet, F. Shakib, T.E. Spelce, F. Chalot and T.E. Tezduyar, A Petrov–Galerkin finite element method for the compressible Euler and Navier–Stokes equations, in: T.E. Tezduyar and T.J.R. Hughes, eds., Numerical Methods for Compressible Flows – Finite Difference, Element and Volume Techniques, AMD Vol. 78 (ASME, New York, 1986) 19–43.

[37] T.J.R. Hughes, L.P. Franca, I. Harari, M. Mallet, F. Shakib and T.E. Spelce, Finite element method for high-speed flows: Consistent calculation of boundary flux, AIAA 25th Aerospace Sciences Meeting, Paper No. 87-0556, Reno, Nevada, January 1987.

[38] F. Chalot, L.P. Franca, I. Harari, T.J.R. Hughes, F. Shakib, M. Mallet, J. Periaux and B. Stoufflet, Calculation of two-dimensional Euler flows with a new Petrov–Galerkin finite element method, in: A. Dervieux and B. Van Leer, eds., Notes on Numerical Fluid Mechanics (Vieweg, Braunschweig, 1989) 88–104.

[39] C.M. Hung and R.W. MacCormack, Numerical solutions of supersonic and hypersonic laminar compression corner flows, AIAA J. 14 (4) (1976) 475–481.

[40] J.E. Carter, Numerical solutions of the Navier–Stokes equations for the supersonic laminar flow over a two-dimensional compression corner, NASA Technical Report, NASA TR R-385, 1972.

[41] M.S. Holden, A study of flow separation in regions of shock wave-boundary layer interaction in hypersonic flow, AIAA 11th Fluid and Plasma Dynamics Conference, Seattle, Washington, 10–12 July 1978.

[42] A.N. Brooks and T.J.R. Hughes, Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 32 (1982) 199–259.

[43] S.K. Jordan and J.E. Fromm, Oscillatory drag, lift, and torque on a circular cylinder in a uniform flow, Phys. Fluids 15 (3) (1972) 371–376.

Codina, R. "Comparison of some FE methods for solving the diff.-conv.-rxn. eqn." Comput. Methods Appl. Mech. Eng. 1998 156:185-210.

$|J| \sim \frac{\partial x}{\partial \xi}$ (in 1D, this $\frac{h}{2}$)

$|J^{-1}| \sim \frac{\partial \xi}{\partial x}$ ($\frac{2}{h}$ in 1D)

$\left(\frac{\partial \xi_i}{\partial x_j}\ \frac{\partial \xi_i}{\partial x_k}\right) A_j A_k$ ...result is mxm

(dxd)(dxm) (mxd)(dxd)(dxm) = (mxm)

how does one interpret this?

— index notation with parens, what does it mean?

$\frac{\partial \xi_i}{\partial x_j} = J^{-1}_{ij}$

$\left[\frac{\partial \xi_i}{\partial x_j}\ \frac{\partial \xi_i}{\partial x_k}\right] = \sum_i J^{-1}_{ij} \cdot J^{-1}_{ik}$ $\quad j,k = 1,..,d$

$= \sum_i$

let: $J^{-1}_{ij} := N_{ij}$ be the inverse Jacobian

d = # dim

m = # vars

N is dxd

A_i are mxm

$$\sum_i N_{ij} N_{ik} = \begin{bmatrix} \sum_i N_{i1}^2 & \sum_i N_{i1}N_{i2} & \sum_i N_{i1}N_{i3} \\ \text{sym.} & \sum_i N_{i2}^2 & \sum_i N_{i2}N_{i3} \\ \text{sym.} & \text{sym.} & \sum_i N_{i3}^2 \end{bmatrix}$$

$A_j^T \frac{\partial \xi_i}{\partial x_j} \frac{\partial \xi_i}{\partial x_k} A_k$ or $A_k^T \frac{\partial \xi_i}{\partial x_k} \frac{\partial \xi_i}{\partial x_j} A_j$?

τ should be symmetric..

x: d=2, m=1 $\quad N = 2\times 2$, $A_i = 2\times 1$

$\frac{\partial u}{\partial t} + \underset{\sim}{a} \cdot \nabla u - \kappa \Delta u = 0$