INL/EXT-13-28278

Newton-Krylov Based P2 Projection Solver for Fluid Flows

Robert Nourgaliev
Mark Christon
Jozsef Bakosi
Anh Bui

June 2013

The INL is a U.S. Department of Energy National Laboratory
operated by Battelle Energy Alliance

INL/EXT-13-28278

Newton-Krylov Based P2 Projection Solver for Fluid Flows

Robert Nourgaliev
Mark Christon
Jozsef Bakosi
Anh Bui

June 2013

Idaho National Laboratory
Nuclear Science & Technology
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

This Page is Intentionally Left Blank

Newton-Krylov based P2 Projection Solver for Fluid Flows

ROBERT NOURGALIEV*, MARK CHRISTON**, JOZSEF BAKOSI**, ANH BUI*

*Nuclear Science & Technology
Idaho National Laboratory
P.O. Box 1625, Idaho Falls, ID 83415-3840, USA

**Computational Physics Group
Computer, Computational and Statistical Sciences Division
Los Alamos National Laboratory
Los Alamos, NM 87545, USA

INL/EXT-13-28278 (100 p.)
June 17, 2013

This Page is Intentionally Left Blank

Abstract

THE purpose of the present document is to formulate Jacobian-free Newton-Krylov algorithm for approximate projection method used in Hydra-TH code. Hydra-TH is developed by Los Alamos National Laboratory (LANL) under the auspices of the Consortium for Advanced Simulation of Light-Water Reactors (CASL) for thermal-hydraulics applications ranging from grid-to-rod fretting (GTRF) to multiphase flow subcooled boiling. Currently, Hydra-TH is based on the semi-implicit projection method, which provides an excellent platform for simulation of transient single-phase thermalhydraulics problems. This algorithm however is not efficient when applied for very slow or steady-state problems, as well as for highly non-linear multiphase problems relevant to nuclear reactor thermalhydraulics with boiling and condensation. These applications require fully-implicit tightly-coupling algorithms. The major technical contribution of the present report is the formulation of *fully-implicit projection algorithm* which will fulfill this purpose. This includes the definition of non-linear residuals used for GMRES-based linear iterations, as well as physics-based preconditioning techniques.

This Page is Intentionally Left Blank

Acknowledgements

THIS work has been authored by Battelle Energy Alliance, LLC under contract No. DE-AC07-05ID14517 (INL/EXT-13-28278) with the U.S. Department of Energy. The United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manual, or allow others to do so, for United States Government purposes.

This Page is Intentionally Left Blank

Contents

This Page is Intentionally Left Blank

List of Tables

List of Figures

Chapter 1

Introduction

THE solution of the time-dependent incompressible single- and multiphase flows poses several algorithmic problems due to the div-free constraint, and the concomitant spatial and temporal resolution required to perform time-accurate solutions particularly when complex geometry is involved. The initial deployment of Hydra-TH has focused on projection methods because of their computational efficiency and accuracy for transient flows. However, when applied to slow transients and steady-state problems, the currently existing projection methods are not cost-effective, due to stability restrictions imposed by material Courant limit. For these applications, fully-implicit algorithms are required. Here, we reformulate semi-implicit projection method to fit into the fully-implicit Jacobian-free Newton Krylov solution strategy.

We start with a short description of governing equations, defined in Chapter 3. Even though we limit our discussion here to single-phase flows, the basic ideas introduced are extendable to multi-fluid formulation [NC12].

A detailed review of projection methods is beyond the scope of this document, but a partial list of relevant work is provided for the interested reader. Projection methods, also commonly referred to as fractional-step, pressure correction methods, or Chorin's method [Cho68] have grown in popularity over the past 20 years due to the relative ease of implementation and computational performance. This is reflected by the volume of work published on the development of second-order accurate projection methods, see for example van Kan [Kan86], Bell, et al. [BCG89], Gresho, et al. [Gre90, GC90, GCCH95, GC96], Almgren, et al. [ABCH93, ABS96, ABC00], Rider [Rid94b, Rid94a, RKM+95, Rid95],

Minion [Min96], Guermond and Quartapelle [GQ97], Puckett, et al. [PAB+97], Sussman, et al. [SAB+99], and Knio, et al. [KNW99]. The numerical performance of projection methods has been considered by Brown and Minion [BM95, MB97], Wetton [Wet98], Guermond [Gue96, Gue97], Guermond and Quartapelle [GQ98b, GQ98a], and Almgren et al. [ABC00]. A short introduction to semi-implicit projection method is given in Chapter 4.

The main technical contribution of this report is described in chapters 5 and 6, introducing fully-implicit projection and its physics-based preconditioning.

Concluding remarks are given in the final chapter 8.

Chapter 2

Hydra-TH

HYDRA-TH [Chr11] refers to the specific physics module that provides the hybrid finite-volume/finite-element incompressible/low-Mach flow solver. This is built as one of the many physics modules using the Hydra multiphysics toolkit. The toolkit provides a rich suite of components that permits rapid application development, I/O interfaces to permit reading/writing multiple file formats for meshes, plot data, time-history and surface-based output. The toolkit also provides run-time parallel domain decomposition with data-migration for both static and dynamic load-balancing. Linear algebra is handled through an abstract interface that permits use of popular libraries such as PetSC and Trilinos. Hydra's toolkit model for development provides lightweight, high-performance and reusable code components for agile development. Currently the toolkit supports finite-element based solvers for time-dependent heat conduction, time-dependent advection-diffusion, time-dependent incompressible flow, multiple Lagrangian hydrodynamics solvers, rigid-body dynamics, etc. In addition, unstructured-grid finite-volume solvers are available for solving time-dependent advection-diffusion, Burgers' equation, the compressible Euler equations, and incompressible/low-Mach Navier-Stokes equations. There are also interfaces to the FronTier front-tracking software and to level-set methods.

This Page is Intentionally Left Blank

Chapter 3

Governing Equations

In the current chapter, we formulate governing fluid dynamics equations considered in the following chapters. We discuss both *incompressible* and (*weakly*[1]) *compressible* formulations.

3.1 Mass conservation

The mass conservation principle in divergence form is

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0. \tag{3.1}$$

Incompressible. In the incompressible limit, the velocity field is solenoidal,

$$\nabla \cdot \mathbf{v} = 0 \tag{3.2}$$

while density can be either constant, or variable – a function of some material index function χ[2] (in multiphase flow configurations), $\rho(\chi)$. This implies a mass density transport equation,

$$\frac{\partial \rho}{\partial t} + v_j \frac{\partial \rho}{\partial x_j} = 0. \tag{3.3}$$

Compressible. We are considering two compressible flow configurations:

[1] Projection algorithms are generally considered inadequate for strong (shock-wave dynamics) compressible flows.

[2] The index function χ could be either the level set function (for the LS method [Set99]), or the volume fraction (for the VOF method [HN81]).

1. Acoustically-filtered, $\rho(T)$. In this case, the velocity field can still be divergence-free, eq.(3.2), which allows to eliminate sound waves from consideration and simplify numerical treatment.

2. Fully-compressible, $\rho(P,T)$, in which case the energy equation is tightly coupled to both mass and momentum conservation.

3.2 Momentum conservation

The conservation of linear momentum is

$$\rho\frac{\partial v_i}{\partial t} + \rho v_j \frac{\partial v_i}{\partial x_j} = \frac{\partial \sigma_{ij}}{\partial x_j} + \rho f_i \tag{3.4}$$

where v_i are components of the velocity vector $\mathbf{v}$, σ_{ij} is the stress tensor, ρ is the mass density, and f_i is the body force. The body force contribution ρf_i typically accounts for buoyancy forces with f_i representing the acceleration due to gravity.

The stress may be written in terms of the fluid pressure and the deviatoric stress tensor as

$$\sigma_{ij} = -p\delta_{ij} + \tau_{ij} \tag{3.5}$$

where p is the pressure, δ_{ij} is the Kronecker delta, and τ_{ij} is the deviatoric stress tensor. A constitutive equation relates the deviatoric stress and the strain rate, e.g.,

$$\tau_{ij} = 2\mu S_{ij}. \tag{3.6}$$

The strain-rate tensor is written in terms of the velocity gradients as

$$S_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right). \tag{3.7}$$

3.3 Energy conservation

The energy equation may be expressed in terms of temperature, T, as

$$\frac{\partial \rho C_p T}{\partial t} + \frac{\partial}{\partial x_j}\Big(\rho v_j C_p T\Big) = \frac{\partial q_j}{\partial x_j} + q''' \tag{3.8}$$

where C_p is the specific heat at constant pressure, q_i is the diffusional heat flux rate, and q''' represents volumetric heat sources and sinks, e.g., due to exothermic/endothermic chemical reactions. Fourier's law relates the heat flux rate to the temperature gradient and thermal conductivity

$$q_i = \kappa \frac{\partial T}{\partial x_i} \tag{3.9}$$

where κ is the thermal conductivity.

Alternatively, one can solve in terms of specific internal energy:

$$\frac{\partial \rho u}{\partial t} + \frac{\partial}{\partial x_j}\Big(\rho v_j u\Big) = \frac{\partial q_j}{\partial x_j} + q''' + \Phi - p\frac{\partial v_j}{\partial x_j} \tag{3.10}$$

with a given function

$$u = \mathcal{F}\left(T\right)$$

For example,

$$u\left(T\right) = u_{_0} + C_{_v}\left(T - T_{_0}\right)$$

where $u_{_0}$ and $T_{_0}$ are the values of specific internal energy and temperature at some reference point, while C_v is specific heat. Φ represents viscous heating, which we will ignore, as well as the last term in eq.(3.10), as the flow of interest is incompressible.

3.4 Scalar transport and turbulence

In addition, we consider a coupled solution for transport of scalars:

$$\frac{\partial \rho \phi_{_n}}{\partial t} + \frac{\partial}{\partial x_j}\Big(\rho v_j \phi_{_n}\Big) = \frac{\partial \mathcal{J}_{_{n_j}}}{\partial x_j} + \mathcal{J}_{_n}''' \tag{3.11}$$

where by $\mathcal{J}_{_{n_j}}$ and $\mathcal{J}_{_n}'''$ we denote diffusive flux and volumetric sources for a scalar $\phi_{_n}$. Note that $\phi_{_n}$ could represent turbulent transport quantities (e.g., turbulent viscosity $\mu_{_t}$, or turbulent kinetic energy k and energy dissipation rate ε). In this case, momentum and heat diffusion coefficients are considered to be a function of $\phi_{_n}$. In the most general case,

$$\mu\Big(T, \phi_n\Big) \quad \text{and} \quad \kappa\Big(T, \phi_n\Big), \quad n = 0, ..., N-1$$

As an example, we will consider Spalart-Allmaras turbulence model [SA92]. In this case, $N = 1$, $\phi_0 = \mu_t$ and:

$$\mathcal{J}_{0_j} = \frac{\mu_m + \mu_t}{\sigma} \frac{\partial \mu_t}{\partial x_j} \tag{3.12}$$

and

$$\mathcal{J}_0''' = C_{b1} \mathcal{S}_a \mu_t - \frac{C_{w1} f_w}{\rho} \left(\frac{\mu_t}{d} \right)^2 + \frac{C_{b2}}{\rho \sigma} \frac{\partial \mu_t}{\partial x_j} \frac{\partial \mu_t}{\partial x_j} \tag{3.13}$$

where the damping functions and the rest of the coefficients are defined as:

$$f_w = g \left[\frac{1 + C_{w3}^3}{g^6 + C_{w3}^3} \right]^{1/6}, \quad g = r + C_{w2} \left(r^6 - r \right), \quad r = \frac{\mu_t}{\rho \mathcal{S}_a \psi^2 d^2} \tag{3.14}$$

$$\mathcal{S}_a = \mathcal{S}_r + \frac{\mu_t}{\rho \psi^2 d^2} f_{v2}, \quad \mathcal{S}_r = \sqrt{2 \mathcal{R}_{ij} \mathcal{R}_{ij}}, \quad \mathcal{R}_{ij} = \frac{1}{2} \left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i} \right) \tag{3.15}$$

$$f_{v1} = \frac{\chi^3}{\chi^3 + C_{v1}^3}, \quad f_{v2} = 1 - \frac{\chi}{1 - \chi f_{v1}}, \quad \chi = \frac{\mu_t}{\mu_m} \tag{3.16}$$

and $C_{b1} = 0.1355$, $C_{b2} = 0.622$, $\sigma = 2/3$, $\psi = 0.41$, $C_{w1} =$, $C_{v1} = 7.1$, $C_{v2} = 5$, $C_{w1} = \left(\frac{C_{b1}}{\psi^2} + \frac{1 + C_{b2}}{\sigma} \right)$, $C_{w2} = 0.3$ and $C_{w3} = 2$.

Here, d is the normal diatance from the wall, while $\mu_m(T)$ is molecular dynamic viscosity, which is in general is a function of temperature. The effective eddy viscosity and thermal conductivity are defined as

$$\mu(\mu_t, T) = \mu_m(T) + \mu_t f_{v1} \quad \text{and} \quad \kappa(\mu_t, T) = \kappa_m(T) + \frac{\mu_t}{Pr_t} \tag{3.17}$$

Chapter 4

Semi-Implicit Projection

FOR simplicity, we consider here only incompressible, constant-density, isothermal, laminar flow formulation.

Following the well-established finite-volume procedure, we discretize momentum equation in space, integrate by parts, and apply the divergence theorem. Using a piecewise-constant weight functions yields

$$\rho\frac{d}{dt}\int_{\Omega^e}\mathbf{v}\,d\Omega^e+\oint_{\Gamma^e}\rho\mathbf{v}(\mathbf{v}\cdot\mathbf{n})\,d\Gamma^e-\oint_{\Gamma^e}\boldsymbol{\tau}\cdot\mathbf{n}\,d\Gamma^e+\int_{\Omega^e}\nabla p\,d\Omega^e-\int_{\Omega^e}\mathbf{f}\,d\Omega^e \tag{4.1}$$

Using definition for the cell-average,

$$\overline{u}=\frac{1}{\Omega^e}\int_{\Omega^e}u^h \tag{4.2}$$

the spatially-discrete momentum equations become

$$\rho\Omega^e\,\frac{d\overline{\mathbf{v}}}{dt}+\oint_{\Gamma^e}\rho\mathbf{v}(\mathbf{v}\cdot\mathbf{n})\,d\Gamma^e-\oint_{\Gamma^e}\boldsymbol{\tau}\cdot\mathbf{n}\,d\Gamma^e+\int_{\Omega^e}\nabla p\,d\Omega^e-\int_{\Omega^e}\mathbf{f}\,d\Omega^e \tag{4.3}$$

The projection algorithm can be derived a number of ways. Here, we choose to first develop the time-integrator, and identify the terms associated with the projection via a Helmholtz decomposition of the velocity. Before proceeding we define

the following mass, advective, viscous, gradient and body-force operators.

$$M = \rho\Omega^e \tag{4.4}$$

$$A(\rho,\mathbf{v})\overline{\mathbf{v}} = \oint_{\Gamma^e} \rho\mathbf{v}(\mathbf{v}\cdot\mathbf{n})\,d\Gamma^e \tag{4.5}$$

$$K\overline{\mathbf{v}} = \oint_{\Gamma^e} \boldsymbol{\tau}\cdot\mathbf{n}\,d\Gamma^e \tag{4.6}$$

$$\mathbf{B}\bar{p} = \int_{\Omega^e} \nabla p\,d\Omega^e \tag{4.7}$$

$$\mathbf{F} = \int_{\Omega^e} \mathbf{f}\,d\Omega^e \tag{4.8}$$

We form the global operators, apply forward-Euler first, then backward-Euler with explicit advection in both cases, and take the sum of the fully-discrete systems results in the following

$$\begin{aligned} &M\frac{\overline{\mathbf{v}}^{n+1}-\overline{\mathbf{v}}^{n}}{\Delta t} = (1-\theta)K\overline{\mathbf{v}}^{n} + \theta K\overline{\mathbf{v}}^{n+1} + (1-\theta)\mathbf{F}^{n} + \theta\mathbf{F}^{n+1} - \\ &-(1-\theta)A(\rho,\mathbf{v})\overline{\mathbf{v}}^{n} - \theta A(\rho,\mathbf{v})\overline{\mathbf{v}}^{n+1} - \mathbf{B}\bar{p}^{n} - \theta_{p}\mathbf{B}(\bar{p}^{n+1}-\bar{p}^{n}) \end{aligned} \tag{4.9}$$

where $0 \leq \theta \leq 1$, $\theta = 0$ corresponds to a forward-Euler, $\theta = 1/2$ a trapezoidal rule, and $\theta = 1$ backward-Euler treatment of viscous and body-force terms.

Using the *Helmholtz decomposition* as

$$\rho\overline{\mathbf{v}}^{*} = \rho\overline{\mathbf{v}}^{n+1} + \nabla\lambda \tag{4.10}$$

we introduce the following definition

$$\lambda = \theta_{p}\Delta t(\bar{p}^{n+1} - \bar{p}^{n}) \tag{4.11}$$

Plugging these into Eq. (4.9), the momentum equation can be formulated for the approximate ("predictor") velocity as

$$\begin{aligned} [M - \theta\Delta t\,(K - A\,(\rho,\mathbf{v}))]\,\overline{\mathbf{v}}^{*} &= [M + (1-\theta)\Delta t\,(K - A\,(\rho,\mathbf{v}))]\,\overline{\mathbf{v}}^{n} + \\ &+\Delta t\left((1-\theta)\mathbf{F}^{n} + \theta\mathbf{F}^{n+1} - B\bar{p}^{n}\right) + \\ &+\theta\Delta t A\,(\rho,\mathbf{v})\,\cancelto{0}{\frac{\nabla\lambda}{\rho}} - \theta\Delta t K\,\cancelto{0}{\frac{\nabla\lambda}{\rho}} + \cancelto{0}{\left[M\frac{\nabla\lambda}{\rho} - B\lambda\right]} \end{aligned} \tag{4.12}$$

Using the Helmholtz decomposition, and requiring $\nabla\overline{\mathbf{v}}^{n+1} = 0$, yields a pressure-Poisson equation (PPE) that can be solved for the *Lagrange multiplier* λ:

$$\nabla \cdot \frac{1}{\rho}\nabla\lambda = \nabla \cdot \overline{\mathbf{v}}^{*} \tag{4.13}$$

Given a velocity and pressure at time-level n, the P2 algorithm proceeds as follows.

Algorithm 1 *Basic P2 Algorithm*

1. *Solve for* $\overline{\mathbf{v}}^{*}$

$$\begin{aligned}\left[M - \theta\Delta t\left(K - A\left(\rho, \mathbf{v}\right)\right)\right]\overline{\mathbf{v}}^{*} = \left[M + (1-\theta)\Delta t\left(K - A\left(\rho, \mathbf{v}\right)\right)\right]\overline{\mathbf{v}}^{n} + \\ + \Delta t\left((1-\theta)\mathbf{F}^{n} + \theta\mathbf{F}^{n+1} - B\bar{p}^{n}\right)\end{aligned} \tag{4.14}$$

2. *Form the right-hand-side of the PPE, solve for* λ*,*

$$K_{p}\lambda = D \tag{4.15}$$

3. *Update the pressure*

$$\bar{p}^{n+1} = \bar{p}^{n} + \frac{1}{\theta_{p}\Delta t}\lambda \tag{4.16}$$

   Note that testing over the last 20 *years or so has indicated that using* $\theta_{p} = 1/2$ *to update the pressure can lead to temporal oscillations in the pressure. For this reason, we use* $\theta_{p} = 1$ *in the implementation.*

4. *Project the cell-centered velocities*

$$\overline{\mathbf{v}}^{n+1} = \overline{\mathbf{v}}^{*} - \frac{1}{\rho}\mathbf{B}\lambda \tag{4.17}$$

5. *Compute face gradients and project the face-centered velocities*

$$v_{f} = v_{f}^{*} - \frac{1}{\rho_{f}}((B)\lambda)_{f} \cdot \mathbf{n} \tag{4.18}$$

6. *Repeat steps 1 - 5 until the termination time is reached*

This Page is Intentionally Left Blank

Chapter 5

Fully-Implicit Projection

In the present chapter, we consider non-linear solution strategies for fully-implicit projection-based algorithms.

5.1 Incompressible, isothermal, laminar flow

For the sake of simplicity, we first consider isothermal, laminar, constant-density flows. The governing equations are eqs.(3.2) and (3.4). Let's define the following vector field, based on *Helmholtz decomposition*:

$$\overline{\mathbf{v}}^{\star} \equiv \overline{\mathbf{v}} + \frac{1}{\rho}\nabla\lambda \tag{5.1}$$

splitting a vector field into *solenoidal* and *irrotational* parts. In the following, we shall call this vector field *"HD velocity"*.

We will consider three options for defining the vector of unknowns. First, the vector of primitive variables,

$$\mathbf{U} = \begin{bmatrix} \bar{p} \\ \overline{\mathbf{v}} \end{bmatrix}$$

Second, in terms of Lagrange multiplier:

$$\mathbf{V} = \begin{bmatrix} \lambda \\ \overline{\mathbf{v}} \end{bmatrix}$$

and third, in terms of HD-velocity and Lagrange multiplier:

$$\mathbf{W} = \begin{bmatrix} \lambda \\ \overline{\mathbf{v}}^{\star} \end{bmatrix}$$

5.1.1 On interpretation of Lagrange multiplier

Let us write momentum conservation equation as

$$\frac{\partial \overline{\mathbf{v}}}{\partial t} = -\frac{1}{\rho}\nabla P - \nabla \cdot (\overline{\mathbf{v}} \otimes \overline{\mathbf{v}}) + \frac{1}{\rho}\nabla \boldsymbol{\tau} + \mathbf{f} \tag{5.2}$$

and then in the following semi-discrete form:

$$\begin{aligned} \overline{\mathbf{v}}^{\star^{n+1}} = \overline{\mathbf{v}}^n - \Delta t \tfrac{1}{\rho}\nabla \underbrace{\left(-\frac{\lambda^n}{\Delta t} + P^{n+\frac{1}{2}} - \left(\frac{\partial \lambda}{\partial t}\right)^{n+\frac{1}{2}} \right)}_{P^\star} + \\ + \Delta t \left(-\nabla \cdot (\overline{\mathbf{v}} \otimes \overline{\mathbf{v}}) + \tfrac{1}{\rho}\nabla \boldsymbol{\tau} + \mathbf{f} \right)^{n+\frac{1}{2}} \end{aligned} \tag{5.3}$$

where we used eq.(5.1) and $P^\star$ is a discrete pressure representation in the HD-velocity "predictor" step. Thus,

$$P^\star = P^{n+\frac{1}{2}} - \frac{\lambda^n}{\Delta t} - \frac{\lambda^{n+1} - \lambda^n}{\Delta t} \tag{5.4}$$

Leading to the following discrete definition of the Lagrange multiplier:

$$\lambda^{n+1} = \Delta t \left(P^{n+\frac{1}{2}} - P^\star \right) \tag{5.5}$$

From this equation, one can see that *the Lagrange multiplier represents a measure of numerical error in time discretization of pressure gradient of momentum equation*. In order to make a sensible projection algorithm, it is necessary to have this error, so that the PPE and projection are well defined.

Pressure form: $\mathbf{P^{(0)}}$

The first obvious option is to ignore pressure in the "predictor" step ($\bar{p}^\star$), as in the original Chorin's method [Cho68, Cho69], leading to

$$\lambda^{n+1} \rightsquigarrow \Delta t P^{n+\frac{1}{2}} \tag{5.6}$$

As we show in our numerical experiments (Section 7.2), this form leads to the first-order accurate solutions, for both pressure and velocity.

Pressure gradient form: $\mathbf{P^{(1)}}$

In the semi-implicit method described in Chapter 4,

$$\begin{array}{lcll} P^{n+\frac{1}{2}} & = & (1-\theta_p)\,P^n + \theta_p P^{n+1} + O\left(\Delta t^{1+4\theta_p(1-\theta_p)}\right) & \text{(Trapezoidal rule)} \\ P^{\star} & = & P^n + O\left(\Delta t\right) & \text{(Forward Euler)} \end{array} \tag{5.7}$$

leading to

$$\lambda^{n+1} = \theta_p \Delta t \left(P^{n+1} - P^n\right) \tag{5.8}$$

Thus,

$$P^{n+1} = P^n + \underbrace{\frac{\lambda^{n+1}}{\theta_p \Delta t}}_{\left(\frac{\partial P}{\partial t}\right)^n \Delta t + \ldots} \tag{5.9}$$

and

$$\lambda^{n+1} \rightsquigarrow \theta_p \Delta t^2 \left(\frac{\partial P}{\partial t}\right)^n \tag{5.10}$$

Therefore, λ is a representation of *the second-order truncation errors*, and we call this scheme as ***"pressure gradient"*** form.

Pressure curvature form: $\mathbf{P^{(2)}}$

Another usefull form can be created if we add one more time level to the pressure discretization, $\left\{P^{n-1}, P^n, P^{n+1}\right\}$. With this, we can use the second-order extrapolation for $P^{\star}$ and trapezoidal rule for $P^{n+\frac{1}{2}}$:

$$\begin{array}{lcll} P^{n+\frac{1}{2}} & = & (1-\theta_p)\,P^n + \theta_p P^{n+1} + O\left(\Delta t^{1+4\theta_p(1-\theta_p)}\right) & \text{(Trapezoidal rule)} \\ P^{\star} & = & P^n + \frac{P^{n-1}-P^n}{2\Delta t_n}\Delta t + O\left(\Delta t^2\right) & \text{(Adams-Bashforth)} \end{array} \tag{5.11}$$

leading to

$$\lambda^{n+1} = \theta_p \Delta t \left(P^{n+1} - P^n\right) - \frac{P^n - P^{n-1}}{2\Delta t_n}\Delta t^2 \tag{5.12}$$

or

$$P^{n+1} = P^n + \frac{1}{\theta_p \Delta t}\left(\lambda^{n+1} + \frac{P^n - P^{n-1}}{2\Delta t_n}\Delta t^2\right) \tag{5.13}$$

After some simple algebraic manipulations with eq.(5.12) under $\theta_p = \frac{1}{2}$ and $\Delta t_n = \Delta t$,

$$\lambda^{n+1} \rightsquigarrow \frac{\Delta t^3}{2}\left(\frac{\partial^2 P}{\partial t^2}\right)^n \tag{5.14}$$

Thus, λ is a representation of *the third-order truncation errors*[1], and we call this scheme as ***"pressure curvature"*** form.

In general, both pressure-gradient and pressure-curvature forms belong to the class of "incremental projection" algorithms, as introduced in [BCG89].

5.1.2 Incremental forms

Let us search a new-time solution iteratively, defining new-iteration values $\mathbf{U}^{\diamond\diamond}$, $\mathbf{V}^{\diamond\diamond}$ or $\mathbf{W}^{\diamond\diamond}$ in the following incremental form:

$$\bar{p}^{\diamond\diamond} = \bar{p}^{\diamond} + p' \tag{5.15}$$

$$\lambda^{\diamond\diamond} = \lambda^{\diamond} + \lambda' \tag{5.16}$$

$$\overline{\mathbf{v}}^{\diamond\diamond} = \overline{\mathbf{v}}^{\diamond} + \mathbf{v}' \tag{5.17}$$

$$\overline{\mathbf{v}}^{\star\diamond\diamond} = \overline{\mathbf{v}}^{\star\diamond} + \mathbf{v}^{\star\prime} = \overline{\mathbf{v}}^{\diamond\diamond} + \frac{1}{\rho}\nabla\lambda^{\diamond\diamond} \tag{5.18}$$

and assume the following linearization of body force:

$$\mathbf{F}^{\diamond\diamond} = \mathbf{F}^{\diamond} + \mathbb{F}_{\mathrm{v}}\left(\mathbf{U}^{\diamond}\right)\mathbf{v}' + \mathbf{f}_{p}\left(\mathbf{U}^{\diamond}\right)p' \tag{5.19}$$

[1] Trully speaking, this holds only for $\theta_p = \frac{1}{2}$.

where specific forms of the linearization matrix $\mathbb{F}_{\mathbf{v}}$ and vector $\mathbf{f}_p$ are problem-dependent.

We note that from eq.(5.18),

$$\overline{\mathbf{v}}^{\diamond\diamond} = \overline{\mathbf{v}}^{\star\diamond\diamond} - \frac{1}{\rho}\nabla\lambda^{\diamond\diamond} = \overline{\mathbf{v}}^{\star\diamond} + \mathbf{v}^{\star\prime} - \frac{1}{\rho}\nabla\left(\lambda^{\diamond} + \lambda^{\prime}\right) \tag{5.20}$$

and

$$\overline{\mathbf{v}}^{\diamond} + \mathbf{v}^{\prime} = \underbrace{\overline{\mathbf{v}}^{\star\diamond} - \frac{1}{\rho}\nabla\lambda^{\diamond}}_{\overline{\mathbf{v}}^{\diamond}} + \mathbf{v}^{\star\prime} - \frac{1}{\rho}\nabla\lambda^{\prime} \tag{5.21}$$

Thus,

$$\mathbf{v}^{\prime} = \mathbf{v}^{\star\prime} - \frac{1}{\rho}\nabla\lambda^{\prime} \tag{5.22}$$

Also,

$$\lambda^{\diamond} = \theta_p \Delta t\left(\bar{p}^{\diamond} - \bar{p}^{n}\right) \tag{5.23}$$

and

$$\lambda^{\prime} = \theta_p \Delta t p^{\prime} \tag{5.24}$$

Plug eqs.(5.17) and (5.16) into eq.(4.9):

$$\begin{aligned} M\left(\overline{\mathbf{v}}^{\diamond} + \mathbf{v}^{\star\prime} - \tfrac{1}{\rho}\nabla\lambda^{\prime} - \overline{\mathbf{v}}^{n}\right) &= \Delta t(1-\theta)\left[K - A(\rho, \overline{\mathbf{v}}^{n})\right]\overline{\mathbf{v}}^{n} + \\ &+ \Delta t\theta K\left(\overline{\mathbf{v}}^{\diamond} + \mathbf{v}^{\star\prime} - \tfrac{1}{\rho}\nabla\lambda^{\prime}\right) - \\ &- \Delta t\theta \underbrace{A\left(\rho, \overline{\mathbf{v}}^{\diamond} + \mathbf{v}^{\star\prime} - \frac{1}{\rho}\nabla\lambda^{\prime}\right)\left(\overline{\mathbf{v}}^{\diamond} + \mathbf{v}^{\star\prime} - \frac{1}{\rho}\nabla\lambda^{\prime}\right)}_{\approx A\left(\rho, \overline{\mathbf{v}}^{\diamond}\right)\left(\overline{\mathbf{v}}^{\diamond} + \mathbf{v}^{\star\prime} - \frac{1}{\rho}\nabla\lambda^{\prime}\right)} + \\ &+ \Delta t(1-\theta)\mathbf{F}^{n} + \Delta t\theta\left(\mathbf{F}^{\diamond} + \mathbb{F}_{\mathbf{v}}\left(\mathbf{v}^{\star\prime} - \tfrac{1}{\rho}\nabla\lambda^{\prime}\right) + \tfrac{1}{\theta_p \Delta t}\mathbf{f}_p\lambda^{\prime}\right) - \\ &- \Delta t\mathbf{B}\bar{p}^{n} - \Delta t\theta_p\mathbf{B}(\bar{p}^{\diamond} + \tfrac{1}{\theta_p \Delta t}\lambda^{\prime} - \bar{p}^{n}) \end{aligned} \tag{5.25}$$

After re-grouping and collecting terms, the *momentum correction equation* becomes:

$$\begin{aligned} &\left[\left(\Delta t\theta\left(K - A\left(\rho, \overline{\mathbf{v}}^{\diamond}\right) + \mathbb{F}_{\mathbf{v}}\right) - M\right)\tfrac{1}{\rho}\nabla - \tfrac{\theta}{\theta_p}\mathbf{f}_p + \mathbf{B}\right]\lambda^{\prime} + \\ &+ \left[M - \Delta t\theta\left(K - A\left(\rho, \overline{\mathbf{v}}^{\diamond}\right) + \mathbb{F}_{\mathbf{v}}\right)\right]\mathbf{v}^{\star\prime} = -\mathbf{res}_{\mathbf{v}} \end{aligned} \tag{5.26}$$

where

$$\begin{aligned}\mathbf{res}_{\mathbf{v}} = M\left(\overline{\mathbf{v}}^{\diamond} - \overline{\mathbf{v}}^{n}\right) - \Delta t\left((1-\theta)K\overline{\mathbf{v}}^{n} + \theta K\overline{\mathbf{v}}^{\diamond}\right) + \\ +\Delta t\left((1-\theta)A\left(\rho,\overline{\mathbf{v}}^{n}\right)\overline{\mathbf{v}}^{n} + \theta A\left(\rho,\overline{\mathbf{v}}^{\diamond}\right)\overline{\mathbf{v}}^{\diamond}\right) - \\ -\Delta t\left((1-\theta)\mathbf{F}^{n} + \theta\mathbf{F}^{\diamond}\right) + \\ +\Delta t\left(\mathbf{B}\bar{p}^{n} + \theta_{p}\mathbf{B}(\bar{p}^{\diamond} - \bar{p}^{n})\right)\end{aligned} \tag{5.27}$$

To derive pressure correction (Lagrange multiplier) equation, we take divergence of eq.(5.20):

$$\cancelto{0}{\nabla\cdot\overline{\mathbf{v}}^{\diamond}} = \nabla\cdot\overline{\mathbf{v}}^{\star^{\diamond}} + \nabla\cdot\mathbf{v}^{\star'} - \nabla\cdot\frac{1}{\rho}\nabla\lambda^{\diamond} - \nabla\cdot\frac{1}{\rho}\nabla\lambda' \tag{5.28}$$

After collecting the terms, we derive the following *pressure correction equation*:

$$\nabla\cdot\frac{1}{\rho}\nabla\lambda' - \nabla\cdot\mathbf{v}^{\star'} = -res_{\lambda} \tag{5.29}$$

where

$$res_{\lambda} = \nabla\cdot\frac{1}{\rho}\nabla\lambda^{\diamond} - \nabla\cdot\overline{\mathbf{v}}^{\star^{\diamond}} \tag{5.30}$$

With this, linear iterations of a Newton-based algorithm are defined by the following equation:

$$\underbrace{\begin{bmatrix} \nabla\cdot\frac{1}{\rho}\nabla & -\nabla\cdot \\ \left(\Delta t\theta\left(K - A\left(\rho,\overline{\mathbf{v}}^{\diamond}\right) + \mathbb{F}_{\mathbf{v}}\right) - M\right)\frac{1}{\rho}\nabla - \frac{\theta}{\theta_p}\mathbf{f}_p + \mathbf{B} & M - \Delta t\theta\left(K - A\left(\rho,\overline{\mathbf{v}}^{\diamond}\right) + \mathbb{F}_{\mathbf{v}}\right) \end{bmatrix}}_{\text{Jacobian, } \mathbb{J}_{\mathbf{W}}} \underbrace{\begin{bmatrix} \lambda' \\ \mathbf{v}^{\star'} \end{bmatrix}}_{\mathbf{W}'} = -\underbrace{\begin{bmatrix} res_{\lambda} \\ \mathbf{res}_{\mathbf{v}} \end{bmatrix}}_{\vec{res}_{\mathbf{V}}} \tag{5.31}$$

Non-linear residuals res_{λ} and $\mathbf{res}_{\mathbf{v}}$ are supplied to `PETSC-SNES` [BBE+04] for JFNK implementation.

5.1.3 Implicit treatment of advection operator

In the present section, we discuss Hydra's treatment of advection operator, and introduce modifications removing operator-splitting on spatial variation of an advected scalar.

For simplicity of presentation, we consider only time derivative and convective terms of a generic scalar advection equation,

$$\partial_t \omega\varphi = -\nabla\cdot(\omega\varphi\hat{\mathbf{v}})$$

for an arbitrary scalar function φ, where

$$\omega = \begin{cases} 1: & \text{volume coordinates} \\ \rho: & \text{mass coordinates} \\ \text{etc.} & \end{cases}$$

In the context of ongoing discussion of momentum equation, φ represents components of velocity vector v_x, v_y and v_z. Vector $\hat{\mathbf{v}}$ is generally a divergence-free velocity field. Since we operate in terms of an *approximate projection*, there are small numerical divergence errors, which will be accounted for by substracting the correction term from the discrete representation of advection operator, as

$$\partial_t \omega\varphi = -\nabla\cdot(\omega\varphi\hat{\mathbf{v}}) + \underbrace{\omega\varphi\nabla\cdot\hat{\mathbf{v}}}_{Correction} \tag{5.32}$$

Next, we write the n-th and the $(n+1)$-th time level dicrete contributions for discretization at cell a as:

$$\begin{aligned}(1-\theta)\,\omega\Omega\tfrac{\varphi_a^{n+1}-\varphi_a^n}{\Delta t} = \\ -(1-\theta)\sum_f \tfrac{\omega\Gamma_f}{2}\left[v_f^n\left(\varphi_a^{-n}+\varphi_b^{+n}\right) - \left|v_f^n\right|\left(\varphi_b^{+n}-\varphi_a^{-n}\right)\right] + \\ +(1-\theta)\,\omega\Omega\varphi_a^n\mathcal{D}\hat{\mathbf{v}}_a^n\end{aligned} \tag{5.33}$$

and

$$\begin{aligned}\theta\omega\Omega\tfrac{\varphi_a^{n+1}-\varphi_a^n}{\Delta t} = \\ -\theta\sum_f \tfrac{\omega\Gamma_f}{2}\left[v_f^{n+1}\left(\varphi_a^{-n+1}+\varphi_b^{+n+1}\right) - \left|v_f^{n+1}\right|\left(\varphi_b^{+n+1}-\varphi_a^{-n+1}\right)\right] + \\ +\theta\omega\Omega\varphi_a^{n+1}\mathcal{D}\hat{\mathbf{v}}_a^{n+1}\end{aligned} \tag{5.34}$$

where we used the LLF-based approximate Riemann solver[2]. Summation is performed over all faces of the cell a. Γ_f is the area of the face between cells a and b, while v_f is the face normal velocity and $\mathcal{D}$ is the discrete divergence operator, corresponding to the used approximate projection.

Next, we add eqs.(5.33) and (5.34), using a simple linearization of non-linear terms and re-grouping:

$$\begin{aligned}\omega\Omega\varphi_a^{\diamond\diamond} - \omega\Omega\varphi_a^n = \\ -\Delta t\,(1-\theta)\sum_f \tfrac{\omega\Gamma_f}{2}\left[v_f^n\left(\varphi_a^{-n}+\varphi_b^{+n}\right) - \left|v_f^n\right|\left(\varphi_b^{+n}-\varphi_a^{-n}\right)\right] - \\ -\Delta t\theta\sum_f \tfrac{\omega\Gamma_f}{2}\left[v_f^{\diamond}\left(\varphi_a^{-\diamond\diamond}+\varphi_b^{+\diamond\diamond}\right) - \left|v_f^{\diamond}\right|\left(\varphi_b^{+\diamond\diamond}-\varphi_a^{-\diamond\diamond}\right)\right] + \\ +\Delta t\,(1-\theta)\,\omega\Omega\varphi_a^n\mathcal{D}\hat{\mathbf{v}}_a^n + \Delta t\theta\omega\Omega\varphi_a^{\diamond\diamond}\mathcal{D}\hat{\mathbf{v}}_a^{\diamond}\end{aligned} \tag{5.35}$$

The sided face values are computed as

$$\begin{aligned}\varphi_a^- &= \varphi_a + \tilde{\nabla}\varphi_a\delta\mathbf{r}_a \\ \varphi_b^+ &= \varphi_b - \tilde{\nabla}\varphi_b\delta\mathbf{r}_b\end{aligned} \tag{5.36}$$

where $\tilde{\nabla}$ is a limited gradient evaluated at cell center, and $\delta\mathbf{r}$ is radius-vector poniting from the cell center to edge. Plugging eq.(5.36) into eq.(5.35) and re-

[2]This is the "incompressible" version, where there exist only one eigenvalue – material velocity.

arranging,

$$\underbrace{\omega\Omega\left(1-\Delta t\theta\mathcal{D}\hat{\mathbf{v}}_{a}^{\diamond}\right)\varphi_{a}^{\diamond\diamond}+\Delta t\theta\sum_{f}\frac{\omega\Gamma_{f}}{2}\left[\begin{array}{c}v_{f}^{\diamond}\left(\varphi_{a}^{\diamond\diamond}+\varphi_{b}^{\diamond\diamond}\right)-\\-\left|v_{f}^{\diamond}\right|\left(\varphi_{b}^{\diamond\diamond}-\varphi_{a}^{\diamond\diamond}\right)\end{array}\right]}_{\tilde{A}^{\diamond}\varphi^{\diamond\diamond}}=$$

$$\underbrace{\begin{array}{l}=\omega\Omega\varphi_{a}^{n}-\Delta t\left(1-\theta\right)\sum_{f}\frac{\omega\Gamma_{f}}{2}\left[\begin{array}{c}v_{f}^{n}\left(\varphi_{a}^{n}+\tilde{\nabla}\varphi_{a}^{n}\delta\mathbf{r}_{a}+\varphi_{b}^{n}-\tilde{\nabla}\varphi_{b}^{n}\delta\mathbf{r}_{b}\right)-\\-\left|v_{f}^{n}\right|\left(\varphi_{b}^{n}-\tilde{\nabla}\varphi_{b}^{n}\delta\mathbf{r}_{b}-\varphi_{a}^{n}-\tilde{\nabla}\varphi_{a}^{n}\delta\mathbf{r}_{a}\right)\end{array}\right]+\\+\Delta t\left(1-\theta\right)\omega\Omega\varphi_{a}^{n}\mathcal{D}\hat{\mathbf{v}}_{a}^{n}-\end{array}}_{A^{n}\varphi^{n}}$$

$$\underbrace{-\Delta t\theta\sum_{f}\frac{\omega\Gamma_{f}}{2}\left[\begin{array}{c}v_{f}^{\diamond}\left(\tilde{\nabla}\varphi_{a}^{\clubsuit}\delta\mathbf{r}_{a}-\tilde{\nabla}\varphi_{b}^{\clubsuit}\delta\mathbf{r}_{b}\right)+\\+\left|v_{f}^{\diamond}\right|\left(\tilde{\nabla}\varphi_{b}^{\clubsuit}\delta\mathbf{r}_{b}+\tilde{\nabla}\varphi_{a}^{\clubsuit}\delta\mathbf{r}_{a}\right)\end{array}\right]}_{\delta\tilde{A}^{\diamond}\varphi^{\clubsuit}} \tag{5.37}$$

In the semi-implicit projection, the limited spatial gradients are evaluated at the n-th time level, $\clubsuit = n$, introducing some stability restrictions and operator-splitting errors. In the fully-implicit projection, one can use current-iterate values, $\clubsuit = \diamondsuit$, removing these deficiencies upon non-linear iteration convergence.

5.1.4 Relevance to semi-implicit projection

To establish relation to the semi-implicit projection, we note that

$$\overline{\mathbf{v}}^{*}=\overline{\mathbf{v}}^{\diamond\diamond(d)}+\frac{1}{\rho}\nabla\lambda^{\diamond\diamond} \tag{5.38}$$

or

$$\overline{\mathbf{v}}^{*}=\overline{\mathbf{v}}^{n+1}+\frac{1}{\rho}\nabla\lambda^{n+1} \tag{5.39}$$

upon convergence of non-linear procedure. Also,

$$\lambda^{\diamond\diamond}=\lambda^{\diamond}+\lambda' \tag{5.40}$$

We plug these into eq.(4.9), and re-group as

$$M\frac{\overline{\mathbf{v}}^{*}-\overline{\mathbf{v}}^{n}}{\Delta t}=\underbrace{(1-\theta)\left(K\overline{\mathbf{v}}^{n}-A(\rho,\overline{\mathbf{v}}^{n})\overline{\mathbf{v}}^{n}+\mathbf{F}^{n}\right)-\mathbf{B}\bar{p}^{n}}_{\bar{\mathcal{S}}^{n}}+$$

$$+\theta\left(\left[K-A(\rho,\overline{\mathbf{v}}^{\diamond})\right]\overline{\mathbf{v}}^{*}+\mathbf{F}^{\diamond}-\underbrace{\left[K-A(\rho,\overline{\mathbf{v}}^{\diamond})\right]\left(\frac{1}{\rho}\nabla\lambda^{\diamond}\right)}_{\Xi_t}\right)+ \tag{5.41}$$

$$\underbrace{-\theta\left(\left[K-A(\rho,\overline{\mathbf{v}}^{n+1})\right]\left(\frac{1}{\rho}\nabla\lambda'\right)+\frac{1}{\theta_{p}\Delta t}\mathbf{f}_{p}\lambda'+\mathbb{F}_{\mathbf{v}}\mathbf{v}'\right)}_{\delta\mathcal{J}_{\mathbf{v}}}$$

Now, comparing with eq.(4.14), the difference is in the following term on the r.h.s.:

$$\mathbf{m}'=-\theta\left(\Xi_t+\delta\mathcal{J}_{\mathbf{v}}\right) \tag{5.42}$$

Further re-grouping, we can write:

$$\begin{aligned}\left[M-\theta\Delta t\left(K-A\left(\rho,\overline{\mathbf{v}}^{\diamond}\right)\right)\right]\overline{\mathbf{v}}^{*}=\left[M+(1-\theta)\Delta t\left(K-A\left(\rho,\overline{\mathbf{v}}^{n}\right)\right)\right]\overline{\mathbf{v}}^{n}+\\ +\Delta t\left((1-\theta)\mathbf{F}^{n}+\theta\mathbf{F}^{\diamond}-\mathbf{B}\bar{p}^{n}-\theta\,\delta\tilde{A}\overline{\mathbf{v}}^{\diamond}+\mathbf{m}'\right)\end{aligned} \tag{5.43}$$

where we used the results of Section 5.1.3 for discretization of advection operator.

We can further notice that Helmholtz decomposition becomes

$$\overline{\mathbf{v}}^{*}=\underbrace{\overline{\mathbf{v}}^{\diamond}+\overline{\mathbf{v}}'}_{\overline{\mathbf{v}}^{\diamond\diamond}}+\frac{1}{\rho}\nabla\left(\lambda^{\diamond}+\lambda'\right) \tag{5.44}$$

which leads to the following PPE:

$$\nabla\cdot\frac{1}{\rho}\nabla\left(\lambda^{\diamond}+\lambda'\right)=\nabla\cdot\overline{\mathbf{v}}^{*} \tag{5.45}$$

which is exactly the same as eq.(4.13).

Concerning eq.(5.42), we would like to add a couple of comments.

1. In semi-implicit P2 projection and in the following Picard iterations, the term $\delta\mathcal{J}_{\mathrm{v}}$ is naturally dropped.

   - For P2 projection, this introduces certain operator splitting errors, which not necessary means that the method becomes 1st-order accurate in time. In fact, it is well established that the method converges with the 2nd-order for velocity. It is however only 1-st order accurate for pressure.
   - For Picard iterations, dropping these terms means that convergence rate is not quadratic, as non-linear iterations do not follow slope (defined by Jacobian matrix), as some elements of the Jacobian matrix are effectively zero-ed out.

2. In P2 projection, the term Ξ_t is also ignored.

3. The term Ξ_t is effectively the generalized[3] version of the pressure-update formula eq.(13) in [BCM01]. Instead of adding $\left(-\frac{\nu\Delta t}{2}\nabla^2\phi^{n+1}\right)$ (in the notation of [BCM01]) explicitely to the left of eq.(5.48), we add the current-iteration-based correction to the r.h.s. of the "predictor" velocity equation (5.43).

4. If the term Ξ_t is dropped, the "predictor" velocity equation (5.43) is effectively decoupled from PPE, along the lines of the classical fractional-step projection algorithm. There are small feedbacks which could be still introduced by possible non-linear momentum source terms, and because of fully-implicit treatment of advection operator, but these might be ignored w/o significant impact.

5. Dropping the term Ξ_t is however undesirable when this algorithm is used as a preconditioning for Newton-based non-linear solver, as upon non-linear convergence, this predictor would produce different solution from what is attempted to be solved by JFNK.

5.1.5 Picard Iterations

Based on eqs.(5.43)-(5.45) we can build point-iteration algorithm as follows.

[3] In addition to viscous effects, we account for splitting errors due to advection.

Algorithm 2 *Picard Iterations*

1. *Set initial guess for* $m = 0$*:*

$$\begin{aligned} \overline{\mathbf{v}}^{\diamond} &= \overline{\mathbf{v}}^{n} \\ \mathbf{v}' &= 0 \\ \bar{p}^{\diamond} &= \bar{p}^{n} \\ p' &= 0 \\ \lambda^{\diamond} &= 0 \\ \lambda' &= 0 \end{aligned}$$

2. *Start the* m^{th} *iteration.*

3. *Solve for* $\overline{\mathbf{v}}^{*}$*:*

$$\begin{aligned} \left[M - \theta\Delta t\left(K^{\diamond} - \tilde{A}^{\diamond}\right)\right]\overline{\mathbf{v}}^{*} = \left[M + (1-\theta)\Delta t\left(K^{n} - A^{n}\right)\right]\overline{\mathbf{v}}^{n} + \\ +\Delta t\left((1-\theta)\mathbf{F}^{n} + \theta\left(\mathbf{F}^{\diamond} - \delta\tilde{A}^{\diamond}\overline{\mathbf{v}}^{\diamond}\right) - \mathbf{B}\bar{p}^{n} - \theta\boldsymbol{\Xi}_{t}\right) \end{aligned} \tag{5.46}$$

4. *Compute face-centered velocities* $v_f^{\star}$*.*

5. *Form the right-hand-side of the PPE eq.(5.45), solve for* $\lambda^{\diamond\diamond}$*,*

$$K_p\lambda^{\diamond\diamond} = D \tag{5.47}$$

6. *Update the pressure*

$$\bar{p}^{\diamond\diamond} = \bar{p}^{n} + \frac{1}{\theta_p\Delta t}\lambda^{\diamond\diamond} \tag{5.48}$$

7. *Project the cell-centered velocities*

$$\overline{\mathbf{v}}^{\diamond\diamond} = \overline{\mathbf{v}}^{*} - \frac{1}{\rho}\mathbf{B}\lambda^{\diamond\diamond} \tag{5.49}$$

8. *Compute face gradients and project the face-centered velocities*

$$v_f = v_f^{*} - \frac{1}{\rho_f}((B)\lambda^{\diamond\diamond})_f \cdot \mathbf{n} \tag{5.50}$$

9. *Picard iteration velocity and pressure corrections are now:*

$$\begin{aligned} \mathbf{v}' &= \overline{\mathbf{v}}^{\diamond\diamond} - \overline{\mathbf{v}}^{\diamond} \\ \lambda' &= \lambda^{\diamond\diamond} - \lambda^{\diamond} \end{aligned} \tag{5.51}$$

which allows to compute errors as

$$\begin{aligned} \mathcal{E}_{\mathbf{V}}^{(m)} &= \mathcal{L}_2(\mathbf{v}') \\ \mathcal{E}_{\lambda}^{(m)} &= \mathcal{L}_2(\lambda') \end{aligned} \tag{5.52}$$

10. *Check for convergence:*

$$\begin{aligned} \mathcal{E}_{\mathbf{V}}^{(m)} &< \mathrm{tol}_a \\ \mathcal{E}_{\lambda}^{(m)} &< \mathrm{tol}_a \\ \mathcal{E}_{\mathbf{V}}^{(m)} &< \mathrm{tol}_r\, \mathcal{E}_{\mathbf{V}}^{(0)} \\ \mathcal{E}_{\lambda}^{(m)} &< \mathrm{tol}_r\, \mathcal{E}_{\lambda}^{(0)} \end{aligned} \tag{5.53}$$

If not satisfied, set new Picard iteration (m ++):

$$\begin{aligned} \overline{\mathbf{v}}^{\diamond} &= \overline{\mathbf{v}}^{\diamond\diamond} \\ \bar{p}^{\diamond} &= \bar{p}^{\diamond\diamond} \\ \lambda^{\diamond} &= \lambda^{\diamond\diamond} \end{aligned}$$

and repeat starting from (2).

11. *Otherwice, finish time step:*

$$\begin{aligned} \overline{\mathbf{v}}^{n+1} &= \overline{\mathbf{v}}^{\diamond\diamond} \\ \bar{p}^{n+1} &= \bar{p}^{\diamond\diamond} \end{aligned}$$

5.1.6 Interpretation with the generalized block LU decomposition

It is very informative to cast the above-presented algorithm along the lines of the *generalized block LU factorization*, as introduced by Perot in [Per93]. We do this by introducing the following operators:

$$\mathcal{C} \equiv \Delta t\theta\,(K - A + \mathbf{F}) \tag{5.54}$$

$$\mathcal{Q} \equiv M - \mathcal{C} \tag{5.55}$$

where the operators M, A, K and $\mathbf{F}$ are defined in eq.(4.8). Also, by $\mathcal{D}$ we will denote the discrete divergence operator. With these, one can write eq.(5.2) in the following semi-discrete form:

$$\begin{cases} \mathcal{Q}\overline{\mathbf{v}}^{n+1} + \Delta t \mathbf{B} P^{n+\frac{1}{2}} = \underbrace{\left[M + \frac{1-\theta}{\theta}\mathcal{C} \right] \overline{\mathbf{v}}^{n}}_{\mathbf{a}} \\ \mathcal{D}\overline{\mathbf{v}}^{n+1} = 0 \end{cases} \tag{5.56}$$

or, in matrix form:

$$\begin{bmatrix} \mathcal{Q} & \Delta t \mathbf{B} \\ \mathcal{D} & 0 \end{bmatrix} \begin{pmatrix} \overline{\mathbf{v}}^{n+1} \\ P^{n+\frac{1}{2}} \end{pmatrix} = \begin{pmatrix} \mathbf{a} \\ 0 \end{pmatrix} \tag{5.57}$$

Following Perot [Per93], we introduce the operator $\mathcal{E}$, and approximate eq.(5.57) with

$$\begin{bmatrix} \mathcal{Q} & (\mathcal{Q}\mathcal{E})\,\Delta t \mathbf{B} \\ \mathcal{D} & 0 \end{bmatrix} \begin{pmatrix} \overline{\mathbf{v}}^{n+1} \\ P^{n+\frac{1}{2}} \end{pmatrix} = \begin{pmatrix} \mathbf{a} \\ 0 \end{pmatrix} \tag{5.58}$$

which enables the following LU factorization:

$$\begin{bmatrix} \mathcal{Q} & (\mathcal{Q}\mathcal{E})\,\Delta t \mathbf{B} \\ \mathcal{D} & 0 \end{bmatrix} = \begin{bmatrix} \mathcal{Q} & 0 \\ \mathcal{D} & -\mathcal{D}\mathcal{E}\Delta t \mathbf{B} \end{bmatrix} \cdot \begin{bmatrix} \mathcal{I} & \mathcal{E}\Delta t \mathbf{B} \\ 0 & \mathcal{I} \end{bmatrix} \tag{5.59}$$

where $\mathcal{I}$ denotes the identity matrix. For $\mathcal{E}$, Perot suggested to use the following operator[4]

$$\mathcal{E} = \Delta t^2 \left(M^{-1} + \Delta t \theta K \right) \tag{5.60}$$

This operator allows to compensate for the 1st-order errors, leading to the 2nd order (in time) method for velocity[5].

As noted by Perot, the choice of $\mathcal{E} = \mathcal{Q}^{-1}$ corresponds to the Uzawa's method. The closely-related variations of SIMPLE algorithm [PS72] correspond to $\mathcal{E} \approx$

[4] Perot utilized explicit Adams-Bashforth method for advection operator, and also ignored body forces. Also, we did change notations relative to [Per93].

[5] As noted by Perot, no matter what scheme is used, the discrete pressure will always be first-order accurate in time, which is consistent with our findings.

$\mathcal{Q}^{-1}$. Both methods require nested iterative loops.

Let's re-cast our semi-implicit and fully-implicit algorithms into this LU factorization framework. First, we replace the pressure with the Lagrange multiplier defined by eq.(5.5), leading to:

$$\begin{bmatrix} \mathcal{Q} & (\mathcal{Q}\mathcal{E})\,\mathbf{B} \\ \mathcal{D} & 0 \end{bmatrix} \begin{pmatrix} \overline{\mathbf{v}}^{n+1} \\ \lambda^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{a} - \Delta t\,(\mathcal{Q}\mathcal{E})\,\mathbf{B}P^{\star} \\ 0 \end{pmatrix} \tag{5.61}$$

Next, we can re-write eq.(5.61) as

$$\begin{bmatrix} \mathcal{Q} & \mathcal{Q}M^{-1}\mathbf{B} \\ \mathcal{D} & 0 \end{bmatrix} \begin{pmatrix} \overline{\mathbf{v}}^{n+1} \\ \lambda^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{a} - \Delta t\,(\mathcal{Q}\mathcal{E})\,\mathbf{B}P^{\star} + \mathcal{Q}\,(M^{-1} - \mathcal{E})\,\mathbf{B}\lambda^{\diamond} \\ 0 \end{pmatrix} \tag{5.62}$$

Now, we define $\mathcal{E}$ as

$$\mathcal{E} = \mathcal{Q}^{-1}$$

i.e., the same as in Uzawa/SIMPLE-based methods. This will lead to

$$\begin{bmatrix} \mathcal{Q} & \mathcal{Q}M^{-1}\mathbf{B} \\ \mathcal{D} & 0 \end{bmatrix} \begin{pmatrix} \overline{\mathbf{v}}^{n+1} \\ \lambda^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{a} - \Delta t\,(\mathcal{Q}\mathcal{E})\,\mathbf{B}P^{\star} - \boxed{\mathcal{C}M^{-1}\mathbf{B}\lambda^{\diamond}} \\ 0 \end{pmatrix} \tag{5.63}$$

In the semi-implicit projection, the term in box is dropped, without affecting the order of accuracy and robustness. In the fully-implicit projection, this term is exactly the advection/diffusion of irrotational part of velocity field (term $\boldsymbol{\Xi}_t$ in eq.(5.41)), which can also be safely ignored for large-CFL simulations. However, we found that ignoring this term is responsible for formation of "projection boundary layers" for high-Fo_ν-number transient simulations (see Section 7.2, also briefly mentioned by Perot in [Per93]), so we keep it for low-CFL transients[6]. Now, the l.h.s. matrix of eq.(5.63) can be LU-factorized as

$$\begin{bmatrix} \mathcal{Q} & \mathcal{Q}M^{-1}\mathbf{B} \\ \mathcal{D} & 0 \end{bmatrix} = \begin{bmatrix} \mathcal{Q} & 0 \\ \mathcal{D} & -\underbrace{\mathcal{D}M^{-1}\mathbf{B}}_{\approx K_p} \end{bmatrix} \cdot \begin{bmatrix} \mathcal{I} & M^{-1}\mathbf{B} \\ 0 & \mathcal{I} \end{bmatrix} \tag{5.64}$$

which is exactly what we do in our 3-step Picard iteration loop, provided that operator $\mathcal{D}M^{-1}\mathbf{B}$ is replaced by approximate Laplacian K_p:

[6] It can be seen that this term introduces a coupling to PPE, which might be too stiff when solving large-CFL flows with the Picard-iteration based algorithm.

1,2 Solve for HD-velocity and Lagrange multiplier (PPE):

$$\begin{bmatrix} \mathcal{Q} & 0 \\ \mathcal{D} & -K_p \end{bmatrix} \begin{pmatrix} \overline{\mathbf{v}}^{\star} \\ \lambda^{\diamond\diamond} \end{pmatrix} = \begin{pmatrix} \mathbf{a} - \Delta t \left(\mathcal{Q}\mathcal{E}\right) \mathbf{B} P^{\star} - \boxed{\mathcal{C} M^{-1} \mathbf{B} \lambda^{\diamond}} \\ 0 \end{pmatrix} \tag{5.65}$$

3 Project (LU-backsubstitution):

$$\begin{bmatrix} \mathcal{I} & M^{-1}\mathbf{B} \\ 0 & \mathcal{I} \end{bmatrix} \begin{pmatrix} \overline{\mathbf{v}}^{\diamond\diamond} \\ \lambda^{\diamond\diamond} \end{pmatrix} = \begin{pmatrix} \overline{\mathbf{v}}^{\star} \\ \lambda^{\diamond\diamond} \end{pmatrix} \tag{5.66}$$

5.2 Adding thermal and turbulent effects

The governing equations are eqs.(3.2), (3.4), (3.8) and (3.11). For the sake of discussion, we can think of Spalart-Allmaras turbulence model, when $N = 1$ and $\phi_n = \mu_t$ is turbulent viscosity. The vectors of unknowns are

$$\mathbf{U} = \begin{bmatrix} \bar{p} \\ \overline{\mathbf{v}} \\ \bar{T} \\ \bar{\nu}_t \end{bmatrix}; \quad \mathbf{V} = \begin{bmatrix} \lambda \\ \overline{\mathbf{v}} \\ \bar{T} \\ \bar{\nu}_t \end{bmatrix}; \text{ or } \mathbf{W} = \begin{bmatrix} \lambda \\ \overline{\mathbf{v}}^{\star} \\ \bar{T} \\ \bar{\nu}_t \end{bmatrix}; \tag{5.67}$$

and the governing equations can be written as

$$\nabla \cdot \mathbf{v} = 0 \tag{5.68}$$

$$\rho \frac{\partial \mathbf{v}}{\partial t} + \rho \nabla \cdot (\mathbf{v} \otimes \mathbf{v}) = -\nabla P + \nabla \cdot \boldsymbol{\tau}\left(T, \nu_t, \mathbf{v}\right) + \rho \mathbf{f}\left(T, \mathbf{v}\right) \tag{5.69}$$

$$\frac{\partial \rho u\left(T\right)}{\partial t} + \nabla \cdot \Big(\rho u(T) \mathbf{v}\Big) = \nabla \cdot \Big(\kappa\left(T, \nu_t\right) \nabla T\Big) + q'''\left(T, \mathbf{v}\right) \tag{5.70}$$

$$\frac{\partial \left(\rho \nu_t\right)}{\partial t} + \nabla \cdot \Big(\rho \nu_t \mathbf{v}\Big) = \nabla \cdot \Big(\rho \zeta\left(T, \nu_t\right) \nabla \nu_t\Big) + \mathcal{T}\left(\nu_t, T, \mathbf{v}\right) \tag{5.71}$$

where u, $\nu_t = \frac{\mu_t}{\rho}$, ζ and $\mathcal{T}$ are specific internal energy, turbulent kinematic viscosity, diffusivity for turbulent viscosity, and source/damping term in Spalart-Allmaras model, correspondingly.

Now, we define the following operators:

$$M = \rho\Omega^e \tag{5.72}$$

$$C\left(T\right)T = \int_{\Omega^e} \rho C_p\left(T\right)T\, d\Omega^e \tag{5.73}$$

$$A(\rho,\mathbf{v})\overline{\mathbf{v}} = \oint_{\Gamma^e} \rho\mathbf{v}(\mathbf{v}\cdot\mathbf{n})\, d\Gamma^e \tag{5.74}$$

$$A_\phi(\rho,\mathbf{v})\phi = \oint_{\Gamma^e} \rho\phi(\mathbf{v}\cdot\mathbf{n})\, d\Gamma^e \tag{5.75}$$

$$K\left(T,\nu_t\right)\overline{\mathbf{v}} = \oint_{\Gamma^e} \boldsymbol{\tau}\left(T,\nu_t,\mathbf{v}\right)\cdot\mathbf{n}\, d\Gamma^e \tag{5.76}$$

$$D\left(T,\nu_t\right)T = \oint_{\Gamma^e} \kappa\left(T,\nu_t\right)\left(\nabla T\cdot\mathbf{n}\right) d\Gamma^e \tag{5.77}$$

$$L\left(T,\nu_t\right)\nu_t = \oint_{\Gamma^e} \rho\zeta\left(T,\nu_t\right)\left(\nabla\nu_t\cdot\mathbf{n}\right) d\Gamma^e \tag{5.78}$$

$$\mathbf{B}\bar{p} = \int_{\Omega^e} \nabla p\, d\Omega^e \tag{5.79}$$

$$\mathbf{F}\left(T,\mathbf{v}\right) = \int_{\Omega^e} \mathbf{f}\left(T,\mathbf{v}\right)\, d\Omega^e \tag{5.80}$$

$$Q\left(T,\mathbf{v}\right) = \int_{\Omega^e} q'''\left(T,\mathbf{v}\right)\, d\Omega^e \tag{5.81}$$

$$W\left(\nu_t,T,\mathbf{v}\right) = \int_{\Omega^e} \mathcal{T}\left(\nu_t,T,\mathbf{v}\right)\, d\Omega^e \tag{5.82}$$

Next, we define new-iteration values $\mathbf{U}^{\diamond\diamond}$ (or $\mathbf{V}^{\diamond\diamond}$) in the following incremental form:

$$\bar{p}^{\diamond\diamond} = \bar{p}^{\diamond} + p' \tag{5.83}$$

$$\lambda^{\diamond\diamond} = \lambda^{\diamond} + \lambda' \tag{5.84}$$

$$\overline{\mathbf{v}}^{\diamond\diamond} = \overline{\mathbf{v}}^{\diamond} + \mathbf{v}' \tag{5.85}$$

$$\overline{\mathbf{v}}^{\star\diamond\diamond} = \overline{\mathbf{v}}^{\star\diamond} + \mathbf{v}^{\star\prime} = \overline{\mathbf{v}}^{\diamond\diamond} + \frac{1}{\rho}\nabla\lambda^{\diamond\diamond} \tag{5.86}$$

$$\bar{T}^{\diamond\diamond} = \bar{T}^{\diamond} + T' \tag{5.87}$$

$$\bar{\nu_t}^{\diamond\diamond} = \bar{\nu_t}^{\diamond} + \nu_t{}' \tag{5.88}$$

and introduce the following linearization of source terms:

$$\mathbf{F}^{\diamond\diamond} = \mathbf{F}^{\diamond} + \mathbb{F}_{\mathbf{v}}\left(\mathbf{U}^{\diamond}\right)\mathbf{v}' + \mathbf{f}_{T}\left(\mathbf{U}^{\diamond}\right)T' \tag{5.89}$$

$$Q^{\diamond\diamond} = Q^{\diamond} + \mathbf{Q}_{\mathbf{v}}\left(\mathbf{U}^{\diamond}\right)\mathbf{v}' + Q_{T}\left(\mathbf{U}^{\diamond}\right)T' \tag{5.90}$$

$$W^{\diamond\diamond} = W^{\diamond} + \mathbf{W}_{\mathbf{v}}\left(\mathbf{U}^{\diamond}\right)\mathbf{v}' + W_{T}\left(\mathbf{U}^{\diamond}\right)T' + W_{\nu}\left(\mathbf{U}^{\diamond}\right)\nu_t' \tag{5.91}$$

Also, similar to Section 5.1,

$$\mathbf{v}' = \mathbf{v}^{\star\prime} - \frac{1}{\rho}\nabla\lambda' \tag{5.92}$$

$$\lambda^{\diamond} = \theta_p \Delta t\left(\bar{p}^{\diamond} - \bar{p}^{n}\right) \tag{5.93}$$

and

$$\lambda' = \theta_p \Delta t p' \tag{5.94}$$

Momentum. Plugging these into discrete form of eq.(5.69):

$$\begin{aligned} M\left(\overline{\mathbf{v}}^{\diamond} + \mathbf{v}^{\star\prime} - \tfrac{1}{\rho}\nabla\lambda' - \overline{\mathbf{v}}^{n}\right) = {} & \Delta t(1-\theta)\Big(K\left(T^{n},\nu_t^{n}\right) - A(\rho,\overline{\mathbf{v}}^{n})\Big)\overline{\mathbf{v}}^{n} + \\ & + \Delta t\theta \underbrace{K\left(T^{\diamond}+T',\nu_t^{\diamond}+\nu_t'\right)\left(\overline{\mathbf{v}}^{\diamond} + \mathbf{v}^{\star\prime} - \frac{1}{\rho}\nabla\lambda'\right)}_{\approx K\left(T^{\diamond},\nu_t^{\diamond}\right)\overline{\mathbf{v}}^{\diamond} + \tilde{K}\mathbf{v}' + \hat{\mathbf{K}}T' + \bar{\mathbf{K}}\nu_t' = K^{\diamond}\overline{\mathbf{v}}^{\diamond} + \tilde{\tilde{K}}\mathbf{v}^{\star\prime} + \hat{\hat{K}}\nabla\lambda' + \hat{\mathbf{K}}T' + \bar{\mathbf{K}}\nu_t'} \\ & - \Delta t\theta \underbrace{A\left(\rho,\overline{\mathbf{v}}^{\diamond} + \mathbf{v}^{\star\prime} - \frac{1}{\rho}\nabla\lambda'\right)\left(\overline{\mathbf{v}}^{\diamond} + \mathbf{v}^{\star\prime} - \frac{1}{\rho}\nabla\lambda'\right)}_{\approx A\left(\rho,\overline{\mathbf{v}}^{\diamond}\right)\overline{\mathbf{v}}^{\diamond} + \tilde{\tilde{A}}\mathbf{v}^{\star\prime} - \hat{\tilde{A}}\frac{1}{\rho}\nabla\lambda'} + \\ & + \Delta t(1-\theta)\mathbf{F}^{n} + \Delta t\theta\left(\mathbf{F}^{\diamond} + \mathbb{F}_{\mathbf{v}}^{\diamond}\left(\mathbf{v}^{\star\prime} - \tfrac{1}{\rho}\nabla\lambda'\right) + \mathbf{f}_{T}^{\diamond}T'\right) - \\ & - \Delta t\mathbf{B}\bar{p}^{n} - \Delta t\theta_p\mathbf{B}(\bar{p}^{\diamond} + \tfrac{1}{\theta_p\Delta t}\lambda' - \bar{p}^{n}) \end{aligned} \tag{5.95}$$

Note, for the purpose of our discussion, exact forms of the linearization coefficients $\tilde{\tilde{A}}$, $\tilde{\tilde{K}}$, $\hat{\hat{K}}$, $\hat{\mathbf{K}}$ and $\bar{\mathbf{K}}$ are not important, as we would never need to explicitely

compute them.

After re-grouping and term collection, we got the following ***momentum correction equation***:

$$\boxed{\left[\left(\Delta t\theta\left(\hat{\hat{K}}-\hat{\hat{A}}+\mathbb{F}_{\mathrm{v}}\right)-M\right)\frac{1}{\rho}\nabla+\mathbf{B}\right]}\lambda'-\boxed{\Delta t\theta\left(\hat{\mathbf{K}}+\mathbf{f}_{T}^{\diamond}\right)}T'+ \\ +\boxed{\left[M-\Delta t\theta\left(\tilde{\tilde{K}}-\tilde{\tilde{A}}+\mathbb{F}_{\mathrm{v}}\right)\right]}\mathbf{v}^{\star'}-\boxed{\Delta t\theta\bar{\mathbf{K}}}\,\nu_{t}'=-\mathbf{res}_{\mathrm{v}} \tag{5.96}$$

where

$$\begin{aligned}\mathbf{res}_{\mathrm{v}}=M\left(\overline{\mathbf{v}}^{\diamond}-\overline{\mathbf{v}}^{n}\right)-\Delta t\left((1-\theta)K^{n}\overline{\mathbf{v}}^{n}+\theta K^{\diamond}\overline{\mathbf{v}}^{\diamond}\right)+\\ +\Delta t\left((1-\theta)A^{n}\overline{\mathbf{v}}^{n}+\theta A^{\diamond}\overline{\mathbf{v}}^{\diamond}\right)-\\ -\Delta t\left((1-\theta)\mathbf{F}^{n}+\theta\mathbf{F}^{\diamond}\right)+\\ +\Delta t\left(\mathbf{B}\bar{p}^{n}+\theta_{p}\mathbf{B}(\bar{p}^{\diamond}-\bar{p}^{n})\right)\end{aligned} \tag{5.97}$$

The terms shown in boxes are the elements of the Jacobian matrix, which will never be computed explicitely.

Pressure. To derive pressure correction (Lagrange multiplier) equation, we take divergence of eq.(5.20). After collecting the terms, we derive ***pressure correction equation***, which is identical to eq.(5.29)"

$$\boxed{\nabla\cdot\frac{1}{\rho}\nabla}\lambda'-\boxed{\nabla\cdot}\,\mathbf{v}^{\star'}=-res_{\lambda} \tag{5.98}$$

where

$$res_{\lambda}=\nabla\cdot\frac{1}{\rho}\nabla\lambda^{\diamond}-\nabla\cdot\overline{\mathbf{v}}^{\star\diamond} \tag{5.99}$$

Energy. Next, consider the following discrete form of energy conservation:

$$\underbrace{C\Big(T^{\diamond}+T'\Big)\Big(T^{\diamond}+T'\Big)}_{\approx C^{\diamond}T^{\diamond}+\hat{C}T'} - C^nT^n = \Delta t\,(1-\theta)\Big(D^n - A_{T}^{n} + Q^n\Big)T^n +$$

$$+\Delta t\theta\Big(\underbrace{D\Big(T^{\diamond}+T',\nu_t^{\diamond}+\nu_t'\Big)\Big(T^{\diamond}+T'\Big)}_{\approx D^{\diamond}T^{\diamond}+\hat{D}T'+\bar{D}\nu_t'} - \underbrace{A_{T}\Big(\rho,\overline{\mathbf{v}}^{\diamond}+\mathbf{v}'\Big)\Big(T^{\diamond}+T'\Big)}_{\approx A_{T}^{\diamond}T^{\diamond}+\hat{A}_{T}T'+\tilde{\hat{\mathbf{A}}}_{\mathbf{T}}\mathbf{V}^{\star'}-\hat{\hat{\mathbf{A}}}_{\mathbf{T}}\frac{1}{\rho}\nabla\lambda'}\Big) + \tag{5.100}$$

$$+\Delta t\theta\Big(Q^{\diamond}+\mathbf{Q}_{\mathbf{v}}^{\diamond}\mathbf{v}^{\star'} - \mathbf{Q}_{\mathbf{v}}^{\diamond}\frac{1}{\rho}\nabla\lambda' + Q_{T}^{\diamond}T'\Big)$$

As stated before, the exact forms of linearization coefficients $\hat{D}$, $\bar{D}$, $\hat{A}_{T}$, $\tilde{\tilde{\mathbf{A}}}_{\mathbf{T}}$ and $\hat{\hat{\mathbf{A}}}_{\mathbf{T}}$ are not important.

Collecting the terms, we got the following ***temperature correction equation***:

$$\boxed{\Big(\hat{C}-\Delta t\theta\Big(\hat{D}-\hat{A}_{T}+Q_{T}^{\diamond}\Big)\Big)}T' + \boxed{\Delta t\theta\Big(\tilde{\tilde{\mathbf{A}}}_{\mathbf{T}}-\mathbf{Q}_{\mathbf{v}}^{\diamond}\Big)}\mathbf{v}^{\star'} +$$

$$+\boxed{\Delta t\theta\Big(\mathbf{Q}_{\mathbf{v}}^{\diamond}-\hat{\hat{\mathbf{A}}}_{\mathrm{T}}\Big)\frac{1}{\rho}\nabla}\lambda' - \boxed{\Delta t\theta\bar{D}}\,\nu_t' = -res_{T} \tag{5.101}$$

where

$$\begin{aligned} res_{T} = C^{\diamond}T^{\diamond} - C^nT^n - \Delta t\Big(&(1-\theta)\,D^nT^n + \theta D^{\diamond}T^{\diamond}\Big) + \\ &+\Delta t\Big((1-\theta)\,A_{T}^{n}T^n + \theta A_{T}^{\diamond}T^{\diamond}\Big) - \\ &-\Delta t\Big((1-\theta)\,Q^n + \theta Q^{\diamond}\Big) \end{aligned} \tag{5.102}$$

Turbulent viscosity. Finally, lets consider discrete turbulent viscosity equation:

$$M\left(\nu_t^{\diamond}+\nu_t'-\nu_t^n\right) = \Delta t(1-\theta)\Big(L\left(T^n,\nu_t^n\right) - A_{\nu}^{n}\Big)\nu_t^n +$$

$$+\Delta t\theta\Big(\underbrace{L\Big(T^{\diamond}+T',\nu_t^{\diamond}+\nu_t'\Big)\Big(\nu_t^{\diamond}+\nu_t'\Big)}_{\approx L^{\diamond}\nu_t^{\diamond}+\hat{L}T'+\bar{L}\nu_t'} - \underbrace{A_{\nu}\Big(\rho,\overline{\mathbf{v}}^{\diamond}+\mathbf{v}'\Big)\Big(\nu_t^{\diamond}+\nu_t'\Big)}_{\approx A_{\nu}^{\diamond}\nu_t^{\diamond}+\bar{A}_{\nu}\nu_t'+\tilde{\hat{\mathbf{A}}}_{\nu}\mathbf{V}^{\star'}-\hat{\hat{\mathbf{A}}}_{\nu}\frac{1}{\rho}\nabla\lambda'}\Big) + \tag{5.103}$$

$$+\Delta t(1-\theta)W^n + \Delta t\theta\Big(W^{\diamond}+\mathbf{W}_{\mathbf{v}}^{\diamond}\mathbf{v}^{\star'} - \mathbf{W}_{\mathbf{v}}^{\diamond}\frac{1}{\rho}\nabla\lambda' + W_{T}^{\diamond}T' + W_{\nu}^{\diamond}\nu_t'\Big)$$

which lead to the following ***turbulent viscosity correction equation***:

$$
\begin{aligned}
&-\boxed{\Delta t\theta\left(\hat{L}+W_{T}^{\diamond}\right)}T'+\boxed{\Delta t\theta\left(\tilde{\tilde{\mathbf{A}}}_{\nu}-\mathbf{W}_{\mathbf{v}}^{\diamond}\right)}\mathbf{v}^{\star'}+\\
&+\boxed{\Delta t\theta\left(\mathbf{W}_{\mathbf{v}}^{\diamond}-\tilde{\tilde{\mathbf{A}}}_{\nu}\right)\frac{1}{\rho}\nabla}\lambda'+\boxed{\left(M-\Delta t\theta\left(\bar{L}-\bar{A}_{\nu}+W_{\nu}^{\diamond}\right)\right)}\nu_{t}^{'}=-res_{\nu}
\end{aligned}
\tag{5.104}
$$

where

$$
\begin{aligned}
res_{\nu}=M\left(\nu_{t}^{\diamond}-\nu_{t}^{n}\right)-\Delta t\Big(&\left(1-\theta\right)L^{n}\nu_{t}^{n}+\theta L^{\diamond}\nu_{t}^{\diamond}\Big)+\\
+\Delta t\Big(&\left(1-\theta\right)A_{\nu}^{n}\nu_{t}^{n}+\theta A_{\nu}^{\diamond}\nu_{t}^{\diamond}\Big)-\\
-\Delta t\Big(&\left(1-\theta\right)W^{n}+\theta W^{\diamond}\Big)
\end{aligned}
\tag{5.105}
$$

With this, linear iterations of a Newton-based algorithm are defined by the following equation:

$$
\underbrace{\begin{bmatrix}
\nabla\cdot\frac{1}{\rho}\nabla & -\nabla\cdot & 0 & 0\\
\left[\left(\Delta t\theta\left(\hat{K}-\hat{A}+\mathbb{F}_{\mathbf{v}}\right)-M\right)\frac{1}{\rho}\nabla+\mathbf{B}\right] & \left[M-\Delta t\theta\left(\tilde{\tilde{K}}-\tilde{\tilde{A}}+\mathbb{F}_{\mathbf{v}}\right)\right] & -\Delta t\theta\left(\hat{\mathbf{K}}+\mathbf{f}_{T}^{\diamond}\right) & -\Delta t\theta\bar{\mathbf{K}}\\
\Delta t\theta\left(\mathbf{Q}_{\mathbf{v}}^{\diamond}-\hat{\mathbf{A}}_{\mathrm{T}}\right)\frac{1}{\rho}\nabla & \Delta t\theta\left(\tilde{\tilde{\mathbf{A}}}_{\mathbf{T}}-\mathbf{Q}_{\mathbf{v}}^{\diamond}\right) & \hat{C}-\Delta t\theta\left(\hat{D}-\hat{A}_{T}+Q_{T}^{\diamond}\right) & -\Delta t\theta\bar{D}\\
\Delta t\theta\left(\mathbf{W}_{\mathbf{v}}^{\diamond}-\tilde{\tilde{\mathbf{A}}}_{\nu}\right)\frac{1}{\rho}\nabla & \Delta t\theta\left(\tilde{\tilde{\mathbf{A}}}_{\nu}-\mathbf{W}_{\mathbf{v}}^{\diamond}\right) & -\Delta t\theta\left(\hat{L}+W_{T}^{\diamond}\right) & M-\Delta t\theta\left(\bar{L}-\bar{A}_{\nu}+W_{\nu}^{\diamond}\right)
\end{bmatrix}}_{\text{Jacobian, }\mathbb{J}_{\mathbf{V}}}\cdot
\underbrace{\begin{bmatrix}\lambda'\\ \mathbf{v}^{\star'}\\ T'\\ \nu_{t}^{'}\end{bmatrix}}_{\mathbf{W}'}=-\underbrace{\begin{bmatrix}res_{\lambda}\\ \mathbf{res}_{\mathbf{v}}\\ res_{T}\\ res_{\nu}\end{bmatrix}}_{\vec{res}_{\mathbf{V}}}
\tag{5.106}
$$

Non-linear residuals res_{λ}, $\mathbf{res}_{\mathbf{v}}$, res_{T} and res_{ν} are supplied to `PETSC-SNES` [BBE+04] for JFNK implementation.

5.2.1 Relevance to semi-implicit projection

In the case of energy and turbulence equations, Hydra-TH *semi-implicit* P2 projection algorithm is implemented as follows:

1. Solve energy equation.
2. Solve (predictor) velocity equations (no divergence-free constraint enforced).
3. Solve turbulent transport.
4. Solve PPE for Lagrange multiplier.
5. Enforce mass conservation by projecting velocity field into the divergence-free subspace.

Based on this sequence, the counterpart of eq.(5.106) is

$$\underbrace{\begin{bmatrix} \nabla\cdot\frac{1}{\rho}\nabla & -\nabla\cdot & 0 & 0 & [4] \\ 0 & \left[M-\Delta t\theta\left(\tilde{\tilde{K}}-\tilde{\tilde{A}}+\mathbb{F}_{\mathbf{v}}\right)\right] & -\Delta t\theta\left(\hat{\mathbf{K}}+\mathbf{f}_T^{\diamond}\right) & 0 & [2] \\ 0 & 0 & \hat{C}-\Delta t\theta\left(\hat{D}-\hat{A}_T+Q_T^{\diamond}\right) & 0 & [1] \\ 0 & \Delta t\theta\left(\tilde{\tilde{\mathbf{A}}}_{\nu}-\mathbf{W}_{\mathbf{v}}^{\diamond}\right) & -\Delta t\theta\left(\hat{L}+W_T^{\diamond}\right) & M-\Delta t\theta\left(\bar{L}-\bar{A}_{\nu}+W_{\nu}^{\diamond}\right) & [3] \end{bmatrix}}_{\text{OS solution matrix}} \cdot \underbrace{\begin{bmatrix} \lambda' \\ \mathbf{v}^{\star\prime} \\ T' \\ \nu_t' \end{bmatrix}}_{\mathbf{W}'} = -\underbrace{\begin{bmatrix} res_{\lambda} \\ \mathbf{res}_{\mathbf{v}} \\ res_T \\ res_{\nu} \end{bmatrix}}_{\vec{res}_{\mathbf{V}}} \tag{5.107}$$

where we show the sequence of the operator-splitting (fractional) steps, as the last column of the OS solution matrix. In the residual computations, one should use

$\overline{\mathbf{v}}^{\diamond} = \overline{\mathbf{v}}^{n}$, $T^{\diamond} = T^{n}$, $\nu_{t}^{\diamond} = \nu_{t}^{n}$, $\bar{p}^{\diamond} = \bar{p}^{n}$ and $\lambda^{\diamond} = 0$. The last (projection) step of the algorithm is symbolically:

$$\overline{\mathbf{v}}^{n+1} = \overline{\mathbf{v}}^{\diamond} + \underbrace{\mathbf{v}^{\star\prime} - \frac{1}{\rho}\nabla\lambda'}_{(\mathbf{v}^{d})'}$$

5.2.2 Picard Iterations

We can now formulate point-iteration algorithm as following.

Algorithm 3 *Picard Iterations*

1. *Set initial guess for* $m = 0$*:*

$$\begin{aligned}
\overline{\mathbf{v}}^{\diamond} &= \overline{\mathbf{v}}^{n} \\
\mathbf{v}' &= 0 \\
\bar{p}^{\diamond} &= \bar{p}^{n} \\
p' &= 0 \\
\lambda^{\diamond} &= 0 \\
\lambda' &= 0 \\
T^{\diamond} &= T^{n} \\
T' &= 0 \\
\nu_{t}^{\diamond} &= \nu_{t}^{n} \\
\nu_{t}' &= 0
\end{aligned}$$

2. *Start* m^{th} *iteration.*

3. *Solve for* $T^{\diamond\diamond}$*:*

$$\begin{aligned}
\left[C^{\diamond} - \theta\Delta t\Big(D^{\diamond} - A_{T}^{\diamond}\Big)\right] T^{\diamond\diamond} = \left[C^{n} + (1-\theta)\,\Delta t\Big(D^{n} - A_{T}^{n}\Big)\right] T^{n} + \\
+\Delta t\Big(\,(1-\theta)\,Q^{n} + \theta Q^{\diamond} - \theta\,\delta\tilde{A}_{T}T^{\diamond}\Big)
\end{aligned} \tag{5.108}$$

4. *Compute temperature correction as*

$$T' = T^{\diamond\diamond} - T^{\diamond}$$

5. *Solve for* $\overline{\mathbf{v}}^*$*:*

$$\left[M-\theta\Delta t\left(K^{\diamond}-A^{\diamond}\right)\right]\overline{\mathbf{v}}^*=\left[M+(1-\theta)\Delta t\left(K^n-A^n\right)\right]\overline{\mathbf{v}}^n+ \\ +\Delta t\left((1-\theta)\mathbf{F}^n+\theta\left(\mathbf{F}^{\diamond}+\mathbf{f}_{T}^{\diamond}T'-\delta\tilde{A}^{\diamond}\overline{\mathbf{v}}^{\diamond}\right)-\mathbf{B}\bar{p}^n-\theta\boldsymbol{\Xi}_t\right) \tag{5.109}$$

6. *Compute velocity corrections as*

$$(\mathbf{v}')^*=\overline{\mathbf{v}}^*-\mathbf{v}^{\diamond}$$

7. *Compute face-centered velocities* $v_f^\star$.

8. *(Need some work!!!) Solve for* $\nu_t^{\diamond\diamond}$*:*

$$\left[M-\theta\Delta t\left(L^{\diamond}-A_{\nu}^{\star}\right)\right]\nu_t^{\diamond\diamond}=\left[M+(1-\theta)\Delta t\left(L^n-A_{\nu}^n\right)\right]\nu_t^n+ \\ +\Delta t\left((1-\theta)W^n+\theta\left(W^{\diamond}+\mathbf{W}_{\mathbf{v}}^{\diamond}(\mathbf{v}')^*+W_{T}^{\diamond}T'\right)\right) \tag{5.110}$$

9. *Form the right-hand-side of the PPE eq.(5.45), solve for* $\lambda^{\diamond\diamond}$*,*

$$K_p\lambda^{\diamond\diamond}=D \tag{5.111}$$

10. *Update the pressure*

$$\bar{p}^{\diamond\diamond}=\bar{p}^n+\frac{1}{\theta_p\Delta t}\lambda^{\diamond\diamond} \tag{5.112}$$

11. *Project the cell-centered velocities*

$$\overline{\mathbf{v}}^{\diamond\diamond}=\overline{\mathbf{v}}^*-\frac{1}{\rho}\mathbf{B}\lambda^{\diamond\diamond} \tag{5.113}$$

12. *Compute face gradients and project the face-centered velocities*

$$v_f^{\diamond\diamond}=v_f^*-\frac{1}{\rho_f}((B)\lambda^{\diamond\diamond})_f\cdot\mathbf{n} \tag{5.114}$$

13. *Picard iteration velocity, turbulent viscosity and pressure corrections are now:*

$$\begin{aligned} \mathbf{v}' &= \overline{\mathbf{v}}^{\diamond\diamond} - \overline{\mathbf{v}}^{\diamond} \\ \nu_t' &= \nu_t^{\diamond\diamond} - \nu_t^{\diamond} \\ \lambda' &= \lambda^{\diamond\diamond} - \lambda^{\diamond} \end{aligned} \tag{5.115}$$

which allows to compute errors as

$$\begin{aligned} \mathcal{E}_{\mathbf{V}}^{(m)} &= \mathcal{L}_2\left(\mathbf{v}'\right) \\ \mathcal{E}_{\lambda}^{(m)} &= \mathcal{L}_2\left(\lambda'\right) \\ \mathcal{E}_{T}^{(m)} &= \mathcal{L}_2\left(T'\right) \\ \mathcal{E}_{\nu_t}^{(m)} &= \mathcal{L}_2\left(\nu_t'\right) \end{aligned} \tag{5.116}$$

14. *Check for convergence:*

$$\begin{aligned} \mathcal{E}_{\mathbf{V}}^{(m)} &< \mathrm{tol}_a \\ \mathcal{E}_{\lambda}^{(m)} &< \mathrm{tol}_a \\ \mathcal{E}_{T}^{(m)} &< \mathrm{tol}_a \\ \mathcal{E}_{\nu_t}^{(m)} &< \mathrm{tol}_a \\ \mathcal{E}_{\mathbf{V}}^{(m)} &< \mathrm{tol}_r\, \mathcal{E}_{\mathbf{V}}^{(0)} \\ \mathcal{E}_{\lambda}^{(m)} &< \mathrm{tol}_r\, \mathcal{E}_{\lambda}^{(0)} \\ \mathcal{E}_{T}^{(m)} &< \mathrm{tol}_r\, \mathcal{E}_{T}^{(0)} \\ \mathcal{E}_{\nu_t}^{(m)} &< \mathrm{tol}_r\, \mathcal{E}_{\nu_t}^{(0)} \end{aligned} \tag{5.117}$$

If not satisfied, set new Picard iteration (m ++):

$$\begin{aligned} \overline{\mathbf{v}}^{\diamond} &= \overline{\mathbf{v}}^{\diamond\diamond} \\ \bar{p}^{\diamond} &= \bar{p}^{\diamond\diamond} \\ \lambda^{\diamond} &= \lambda^{\diamond\diamond} \\ T^{\diamond} &= T^{\diamond\diamond} \\ \nu_t^{\diamond} &= \nu_t^{\diamond\diamond} \end{aligned}$$

and repeat starting from (2).

15. *Otherwice, finish time step:*

$$\begin{aligned} \overline{\mathbf{v}}^{n+1} &= \overline{\mathbf{v}}^{\diamond\diamond} \\ \bar{p}^{n+1} &= \bar{p}^{\diamond\diamond} \\ T^{n+1} &= T^{\diamond\diamond} \\ \nu_t^{n+1} &= \nu_t^{\diamond\diamond} \end{aligned}$$

This Page is Intentionally Left Blank

Chapter 6

Preconditioning

6.1 General strategy

Consider the following modification of eq.(5.31):

$$\underbrace{\mathbb{J}_{\mathbf{V}}\mathbb{P}^{-1}}_{\mathbb{J}_{\mathbf{P}}}\underbrace{\mathbb{P}\mathbf{V}'}_{\mathbf{V}''}=\underbrace{-r\vec{e}s_{\mathbf{V}}\left(\mathbf{V}'\right)}_{\vec{b}} \tag{6.1}$$

where $\mathbb{P}$ symbolically represents the preconditioning matrix (or process), and $\mathbb{P}^{-1}$ is its inverse. Thus, the solution procedure is splitted into two processes:

1. Solving for

$$\mathbb{J}_{\mathbf{P}}\mathbf{V}''=\vec{b} \tag{6.2}$$

   (this is what actually crunched by GMRES), and

2. Preconditioning:

$$\mathbf{V}'=\mathbb{P}^{-1}\mathbf{V}'' \tag{6.3}$$

While one refers to the matrix/process $\mathbb{P}$, operationally the algorithm only requires the action of $\mathbb{P}^{-1}$ on a vector. The main requirement is that $\mathbb{P}$ designed properly, to enable clustering eigenvalues of the $\mathbb{J}_{\mathbf{P}}$, making the solution of eq.(6.2) to converge faster.

For effective preconditioning of the fully-implicit projection algorithm, we can use semi-implicit algorithm described in Chapter 4. The strategy with involving a legacy (e.g., operator-splitting) algorithm for preconditioning is commonly referred to as *Physics-(Process)-based preconditioning (PBP)* [KK04, KR00, KCMM03, KMK96, KMCR05], to be contrasted to the *Matrix-(Math)-based preconditioning (MBP)* algorithms. The later include different flavors of SOR, SSOR, ILU, MILU, ILUT, ILUTP, ILUS, ILUC, etc. preconditioners, see [SS86] for review. In these cases, the preconditioning matrix $\mathbb{P}$ is required, as a suitable approximation for $\mathbb{J}_{\mathbf{v}}$.

In the following section, we will describe details of our implementation of the *semi-implicit projection* as PBP, emphasizing all differences relative to the using this algorithm as a solver (in an operator-splitting OS mode).

6.2 Semi-Implicit Projection as Physics-Based Preconditioning

At the input of the preconditioning step, we have current Newton iteration values of $\overline{\mathbf{v}}^{\diamond}$, $\bar{p}^{\diamond}$ and $\lambda^{\diamond}$, and current update values $\overline{\mathbf{v}}''$, $\bar{p}''$ and λ''. In the OS splitting mode, these are:

$$\overline{\mathbf{v}}^{\diamond} = \overline{\mathbf{v}}^{n},\ \ \bar{p}^{\diamond} = \bar{p}^{n},\ \ \lambda^{\diamond} = 0,\ \ \overline{\mathbf{v}}'' = 0,\ \ \bar{p}'' = 0 \ \text{ and } \ \lambda'' = 0$$

The task of the preconditioning is to convert these into $\overline{\mathbf{v}}^{\heartsuit}$, $\bar{p}^{\heartsuit}$, $\lambda^{\heartsuit}$, $\overline{\mathbf{v}}'$, $\bar{p}'$ and λ', where

$$\begin{array}{rcl} \overline{\mathbf{v}}^{\heartsuit} & = & \overline{\mathbf{v}}^{\diamond} + \overline{\mathbf{v}}' \\ \bar{p}^{\heartsuit} & = & \bar{p}^{\diamond} + \bar{p}' \\ \lambda^{\heartsuit} & = & \lambda^{\diamond} + \lambda' \end{array} \tag{6.4}$$

In the OS mode, $\Phi^{\heartsuit} = \Phi^{n+1}$, where $\Phi = \overline{\mathbf{v}},\ \bar{p}$ and λ.

We define Helmholtz decomposition as

$$\overline{\mathbf{v}}^{\heartsuit,\star} = \underbrace{\overline{\mathbf{v}}^{\diamond} + \mathbf{v}'}_{\text{Divergence-free part, } \overline{\mathbf{v}}^{\heartsuit}} + \frac{1}{\rho}\nabla \underbrace{\left(\lambda^{\diamond} + \lambda'\right)}_{\lambda^{\heartsuit}} \tag{6.5}$$

Non-incremental Form

1. The first step would be to solve for non-solenoidal ("predictor") velocity field, $\overline{\mathbf{v}}^{\heartsuit,\star}$, using one of the following options.

Option-A:

$$
\begin{aligned}
&[M-\Delta t\theta\left(K-A(\rho,\mathbf{v})+\mathbb{F}_{\mathbf{v}}\right)]\,\overline{\mathbf{v}}^{\heartsuit,\star}= \\
&=\left[M+\Delta t(1-\theta)\left(K-A(\rho,\mathbf{v})\right)\right]\overline{\mathbf{v}}^{n}-\boxed{\Delta t\theta K\left(\frac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond}+p''-\bar{p}^{n}\right)\right)}+ \\
&\qquad+\boxed{\Delta t\theta A(\rho,\mathbf{v})\left(\frac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond}+p''-\bar{p}^{n}\right)\right)}+ \\
&\qquad+\Delta t(1-\theta)\mathbf{F}^{n}+\Delta t\theta\left(\mathbf{F}^{\diamond}-\mathbb{F}_{\mathbf{v}}\left(\tfrac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond}-\bar{p}^{n}\right)+\overline{\mathbf{v}}^{\diamond}\right)\right)- \\
&\qquad-\Delta t\theta\left[\mathbb{F}_{\mathbf{v}}\tfrac{\Delta t\theta_{p}}{\rho}\nabla-\mathbf{f}_{p}\right]p''- \\
&-\Delta t\mathbf{B}\bar{p}^{n}-\boxed{\left(\Delta t\theta_{p}\mathbf{B}(\bar{p}^{\diamond}+p''-\bar{p}^{n})-\frac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond}+p''-\bar{p}^{n}\right)\right)}
\end{aligned}
\tag{6.6}
$$

In the OS mode, $\overline{\mathbf{v}}^{\heartsuit,\star}=\overline{\mathbf{v}}^{\star}$ and eq.(6.6) reduces to eq.(4.14).

Option-B:

Equation (6.6) is of advection-diffusion type, which is not well ameanable to multigrid algorithm, and solved in `Hydra` by ILU-based solver. Another viable option would be to convert it into the parabolic equation, by taking out advection operator on the left-hand-side (leaving it to GMRES to deal

with). Thus, the parabolic equation would be:

$$\begin{aligned}
&[M-\Delta t\theta\left(K+\mathbb{F}_{\mathbf{v}}\right)]\overline{\mathbf{v}}^{\heartsuit,\star}=[M+\Delta t(1-\theta)\left(K-A(\rho,\mathbf{v})\right)]\overline{\mathbf{v}}^{n}-\\
&-\Delta t\theta A(\rho,\mathbf{v})\left(\overline{\mathbf{v}}^{\diamond}+\overline{\mathbf{v}}''\right)-\boxed{\Delta t\theta K\left(\frac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond}+p''-\bar{p}^{n}\right)\right)}+\\
&+\boxed{\Delta t\theta A(\rho,\mathbf{v})\left(\frac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond}+p''-\bar{p}^{n}\right)\right)}+\\
&+\Delta t(1-\theta)\mathbf{F}^{n}+\Delta t\theta\left(\mathbf{F}^{\diamond}-\mathbb{F}_{\mathbf{v}}\left(\tfrac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond}-\bar{p}^{n}\right)+\overline{\mathbf{v}}^{\diamond}\right)\right)-\\
&-\Delta t\theta\left[\mathbb{F}_{\mathbf{v}}\tfrac{\Delta t\theta_{p}}{\rho}\nabla-\mathbf{f}_{p}\right]p''-\\
&-\Delta t\mathbf{B}\bar{p}^{n}-\boxed{\left(\Delta t\theta_{p}\mathbf{B}(\bar{p}^{\diamond}+p''-\bar{p}^{n})-\frac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond}+p''-\bar{p}^{n}\right)\right)}
\end{aligned}\tag{6.7}$$

2. The second step would be to form and solve PPE. Taking divergence of eq.(6.5) leads to the following PPE:

$$\underbrace{\nabla\cdot\frac{1}{\rho}\nabla\lambda^{\heartsuit}}_{K_{p}\lambda^{\heartsuit}}=\underbrace{\nabla\cdot\overline{\mathbf{v}}^{\heartsuit,\star}}_{D}\tag{6.8}$$

which reduces to eq.(4.15) in the OS mode.

3. Pressure is computed from the new Lagrange multiplier as:

$$\bar{p}^{\heartsuit}=\bar{p}^{n}+\frac{1}{\theta_{p}\Delta t}\lambda^{\heartsuit}\tag{6.9}$$

4. Next, we project the cell-centered velocities as

$$\overline{\mathbf{v}}^{\heartsuit}=\overline{\mathbf{v}}^{\heartsuit,\star}-\frac{1}{\rho}\mathbf{B}\lambda^{\heartsuit}\tag{6.10}$$

5. Finally, we can compute

$$\begin{aligned}
\overline{\mathbf{v}}' &= \overline{\mathbf{v}}^{\heartsuit}-\overline{\mathbf{v}}^{\diamond}\\
\bar{p}' &= \bar{p}^{\heartsuit}-\bar{p}^{\diamond}\\
\lambda' &= \lambda^{\heartsuit}-\lambda^{\diamond}
\end{aligned}\tag{6.11}$$

and these are the values which are returned to PETSC-SNES. As mentioned above, in the OS mode, this step is absent, as $\Phi^{\heartsuit} = \Phi^{n+1}$.

Incremental Form One can re-write eq.(6.5) as:

$$\overline{\mathbf{v}}^{\heartsuit,\star} = \overline{\mathbf{v}}^{\diamond} + \mathbf{v}^{\star'} \tag{6.12}$$

where

$$\mathbf{v}^{\star'} = \mathbf{v}' + \frac{1}{\rho}\nabla\left(\lambda^{\diamond} + \lambda'\right) \tag{6.13}$$

1. Solve for non-solenoidal velocity increment $\mathbf{v}^{\star'}$.

   Option-A:

$$\begin{aligned}
\left[M - \Delta t\theta\left(K - A(\rho,\mathbf{v}) + \mathbb{F}_{\mathbf{v}}\right)\right]\mathbf{v}^{\star'} = &-\left[M - \Delta t\theta\left(K - A(\rho,\mathbf{v}) + \mathbb{F}_{\mathbf{v}}\right)\right]\overline{\mathbf{v}}^{\diamond} + \\
&+\left[M + \Delta t(1-\theta)\left(K - A(\rho,\mathbf{v})\right)\right]\overline{\mathbf{v}}^{n} - \boxed{\Delta t\theta K\left(\frac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond} + p'' - \bar{p}^{n}\right)\right)} + \\
&+\boxed{\Delta t\theta A(\rho,\mathbf{v})\left(\frac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond} + p'' - \bar{p}^{n}\right)\right)} + \\
&+\Delta t(1-\theta)\mathbf{F}^{n} + \Delta t\theta\left(\mathbf{F}^{\diamond} - \mathbb{F}_{\mathbf{v}}\left(\frac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond} - \bar{p}^{n}\right) + \overline{\mathbf{v}}^{\diamond}\right)\right) - \\
&-\Delta t\theta\left[\mathbb{F}_{\mathbf{v}}\frac{\Delta t\theta_{p}}{\rho}\nabla - \mathbf{f}_{p}\right]p'' - \\
&-\Delta t\mathbf{B}\bar{p}^{n} - \boxed{\left(\Delta t\theta_{p}\mathbf{B}(\bar{p}^{\diamond} + p'' - \bar{p}^{n}) - \frac{\Delta t\theta_{p}}{\rho}\nabla\left(\bar{p}^{\diamond} + p'' - \bar{p}^{n}\right)\right)}
\end{aligned} \tag{6.14}$$

Option-B:

$$
\begin{aligned}
\left[M-\Delta t\theta\left(K+\mathbb{F}_{\mathbf{v}}\right)\right]\mathbf{v}^{\star'} &= -\left[M-\Delta t\theta\left(K+\mathbb{F}_{\mathbf{v}}\right)\right]\overline{\mathbf{v}}^{\diamond}+ \\
&\quad +\left[M+\Delta t(1-\theta)\left(K-A(\rho,\mathbf{v})\right)\right]\overline{\mathbf{v}}^{n}- \\
-\Delta t\theta A(\rho,\mathbf{v})\left(\overline{\mathbf{v}}^{\diamond}+\overline{\mathbf{v}}''\right) &- \boxed{\Delta t\theta K\left(\frac{\Delta t\theta_{_p}}{\rho}\nabla\left(\bar{p}^{\diamond}+p''-\bar{p}^{n}\right)\right)}+ \\
&+\boxed{\Delta t\theta A(\rho,\mathbf{v})\left(\frac{\Delta t\theta_{_p}}{\rho}\nabla\left(\bar{p}^{\diamond}+p''-\bar{p}^{n}\right)\right)}+ \\
+\Delta t(1-\theta)\mathbf{F}^{n}+\Delta t\theta\left(\mathbf{F}^{\diamond}-\mathbb{F}_{\mathbf{v}}\left(\tfrac{\Delta t\theta_{_p}}{\rho}\nabla\left(\bar{p}^{\diamond}-\bar{p}^{n}\right)+\overline{\mathbf{v}}^{\diamond}\right)\right)- \\
-\Delta t\theta\left[\mathbb{F}_{\mathbf{v}}\tfrac{\Delta t\theta_{_p}}{\rho}\nabla-\mathbf{f}_{_p}\right]p''- \\
-\Delta t\mathbf{B}\bar{p}^{n}-\boxed{\left(\Delta t\theta_{_p}\mathbf{B}(\bar{p}^{\diamond}+p''-\bar{p}^{n})-\frac{\Delta t\theta_{_p}}{\rho}\nabla\left(\bar{p}^{\diamond}+p''-\bar{p}^{n}\right)\right)}
\end{aligned}
\tag{6.15}
$$

2. Solve incremental PPE:

$$\nabla\cdot\frac{1}{\rho}\nabla\lambda'=\nabla\cdot\mathbf{v}^{\star'}-\nabla\cdot\frac{1}{\rho}\nabla\lambda^{\diamond} \tag{6.16}$$

3. Convert to pressure correction as

$$\bar{p}'=\frac{\lambda'}{\Delta t\theta_{_p}} \tag{6.17}$$

4. Return to `PETSC-SNES` a preconditioned solution as

$$\begin{bmatrix} \bar{p}'\ \ (\text{or }\lambda') \\ \mathbf{v}'=\mathbf{v}^{\star'}-\frac{1}{\rho}\nabla\left(\lambda^{\diamond}+\lambda'\right) \end{bmatrix} \tag{6.18}$$

Chapter 7

Numerical Examples

7.1 Vortex shedding behind a cylinder

As the first numerical test, we use vortex shedding flow behind a cylindrical obsticle. The Reynolds number was set to $Re = 100$. Simulations were performed on the fixed mesh, with 10,080 linear HEX elements (20,612 nodes), Figure 7.1. We started with an initial solution adjusted to form well-posed initial guess by our start-up procedure, and run the solution until the well-established Karman vortex street established at dimensionless time $t^{\star} \equiv \frac{t\nu}{d^2} = 500$, where d is the diameter of the cylinder, while ν is kinematic viscosity of the fluid. By changing time step, we observe time convergence, comparing the results of semi-implicit and fully-implicit projection algorithms, with different pressure forms and parameter θ_p. The parameter θ for advection and viscous operator was set to $\frac{1}{2}$, in all simulations. For Picard-based fully-implicit projection, we always start with initial CFL=1, increasing time step with factor $\Delta t_{n+1} = 1.1\Delta t_n$, until the desired value is reached. For solutions with CFL> 50, the advection and viscous diffusion of the irrotational part of velocity field (see eq.(5.39)) are ignored $\boldsymbol{\Xi}_t = 0$.

7.1.1 Time convergence

The results of the time convergence study are shown in Figures 7.2-7.12.

We start with the semi-implicit algorithm, shown in Figure 7.2, showing velocity magnitude fields for CFL numbers ranging from 0.44 to 60, when used with

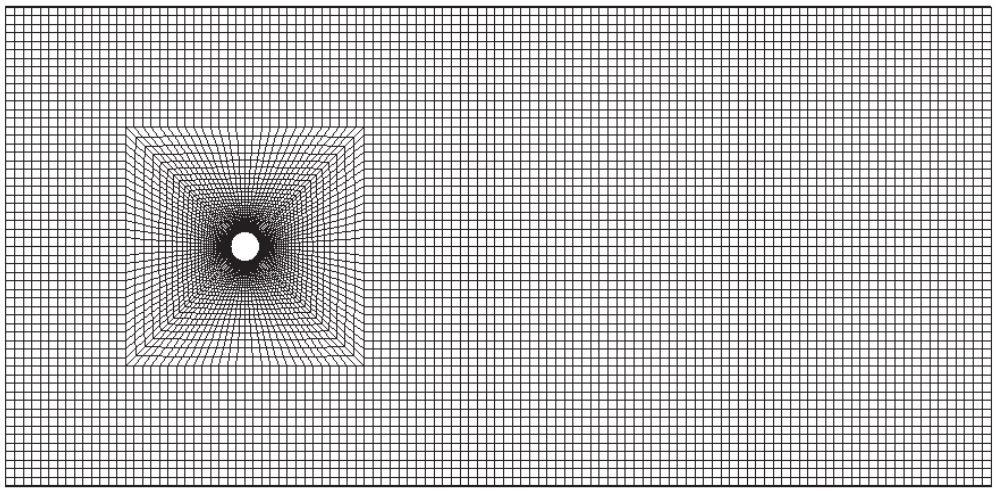

Fig. 7.1 : Computational mesh for vortex shedding test: 20,612 nodes and 10,080 HEX8 elements.

pressure-gradient form and $\theta_p = 1$. It is apparent that the accuracy of the algorithm starts to deteriorate at CFL≈ 10. When the maximum CFL exceeds 40-50, the method becomes unstable. Switching from the pressure-gradient form to the pressure-curvature form (Figure 7.3) shows only marginal improvement in accuracy, and no improvement in robustness.

Time convergence for fully-implicit algorithm is shown in Figures 7.4-7.7. First, one can note a significant improvement in the accuracy of the solution, as the eddies in the wake are decently resolved for CFL≈ 40. Moreover – the method is unconditionally stable for any CFL numbers (provided that non-linear iterations do converge[1]). It can be seen that with fully-implicit projection, we can run simulations with maximum CFL number in the excess of 500. Obviously, for these very large time steps, the vortex shedding becomes under-resolved, which is why the Karman vortex street associated with the Hopf bifurcation is not visible in Fig.

[1]For high CFL numbers (> 50), we found it necessary to turn off the advection and diffusion of the irrotational part of the velocity field, $\boldsymbol{\Xi}_t = 0$, as this correction appears on the r.h.s. of the momentum equation and introduces a stiffness in the Picard-based non-linear iteration loop. Without this correction term, the Picard-based algorithm converges within 10-15 non-linear iterations (to get down to the relative error below the given tolerance level of 10^{-8}), regardless of the maximum CFL number in the flow.

7.4 under $CFL > 400$.

It is instructive to notice significant effect of the method accuracy on vortex shedding. For the "pressure form" (original Chorin's formulation), the Karman street is not resolved for CFL numbers as low as 7, Figure 7.7.

In Figure 7.9, we show direct comparison of the results with the fully-implicit (Picard-based) and the semi-implicit P2 projection algorithms. The improvement in the accuracy and robustness is evident.

It is important to note that the accuracy of the semi-implicit method can be improved by using the "predictor-corrector" (PC) strategy – within our Picard algorithm – we just applied two iterations. We show the comparison in Figure 7.10. The improvement in the vortex resolution is evident. However, this simple strategy does not improve the robustness, as the method becomes unstable at CFL numbers exceeding 40.

We shall note also, that even though the maximum CFL number is high – the actual CFL number in the transient wake region is lower, as the mesh sizes are larger in the wake zone, Figure 7.1. We demostrate this in Figures 7.11 and 7.12, showing the CFL number distribution in the domain and in the wake zone. The reason why we can get very decent eddy resolution with these high CFL numbers is that, in general, the dynamic time scale of the resolved eddies is higher than what is given by CFL=1. We can argue that one can get very decent vortex shedding resolution with CFL≈ 10, when high-order time discretization is used. This gives a significant boost in performance, especially accounting for the fact that at this range of CFL numbers, the Picard non-linear iteration loop converges within only 3-4 iterations.

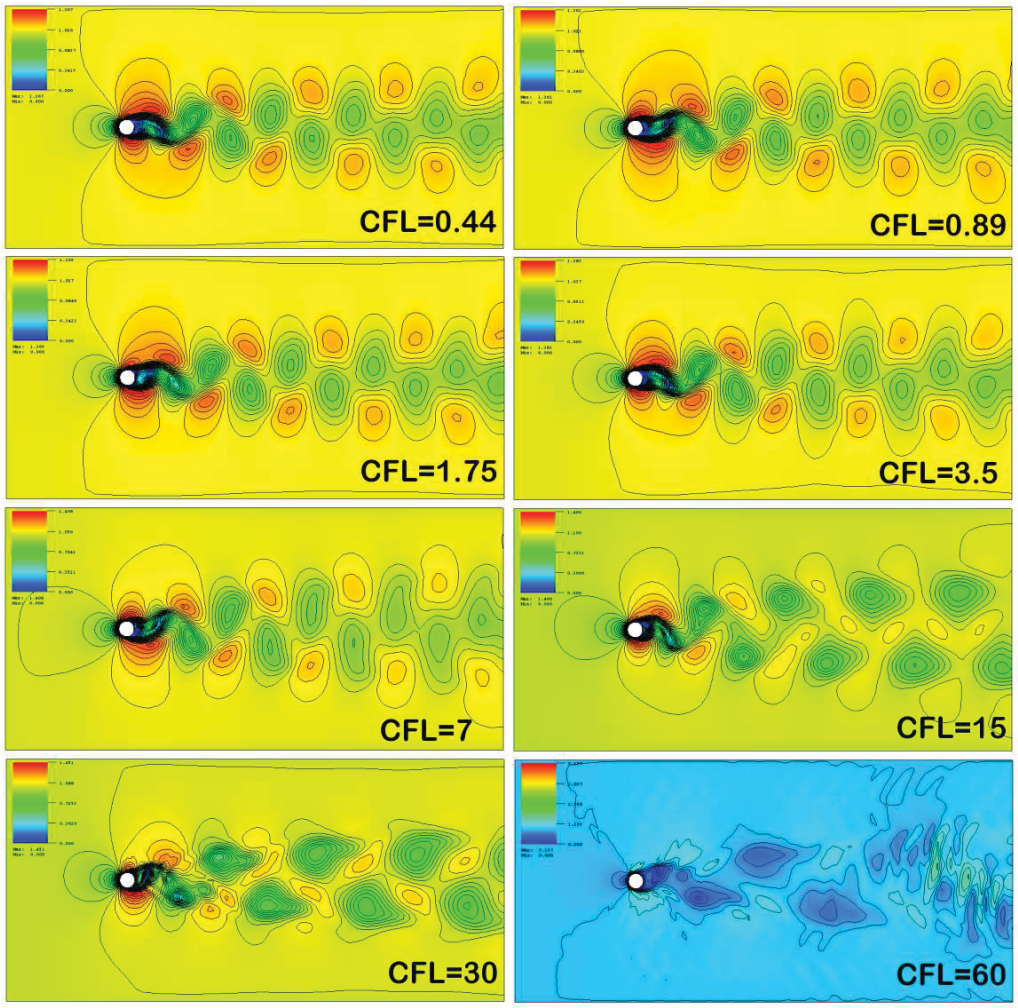

Fig. 7.2 : Convergence of velocity field for semi-implicit projection algorithm with $P^{(1)}$ pressure form and $\theta_p = 1$.

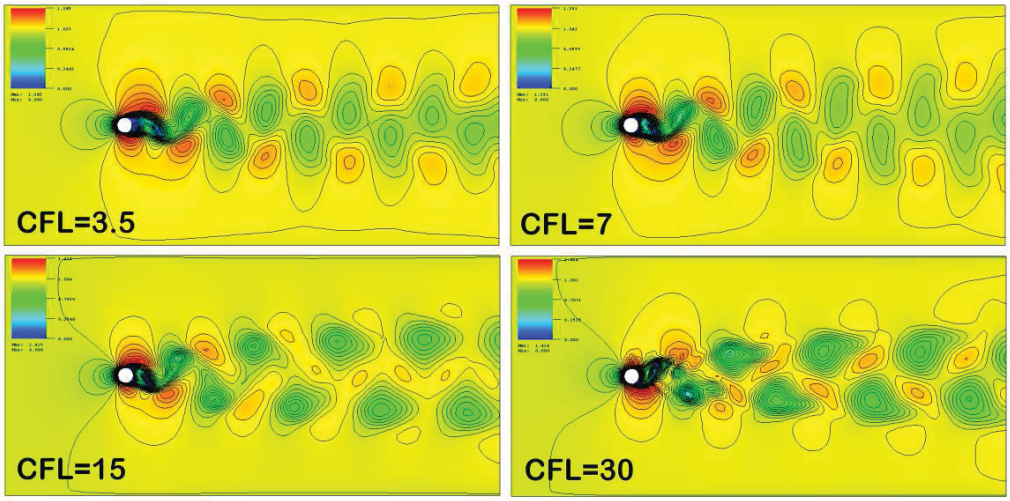

Fig. 7.3 : Convergence of velocity field for semi-implicit projection algorithm with $P^{(2)}$ pressure form and $\theta_p = 1$.

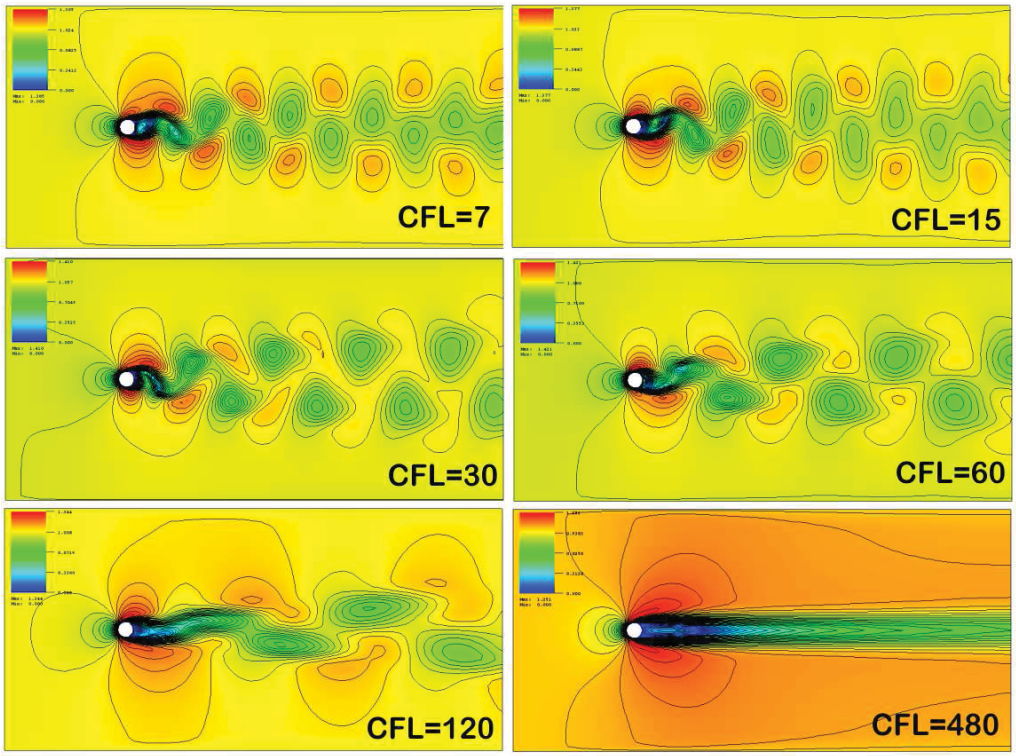

Fig. 7.4 : Convergence of velocity field for fully-implicit projection algorithm with $P^{(1)}$ pressure form and $\theta_p = \frac{1}{2}$.

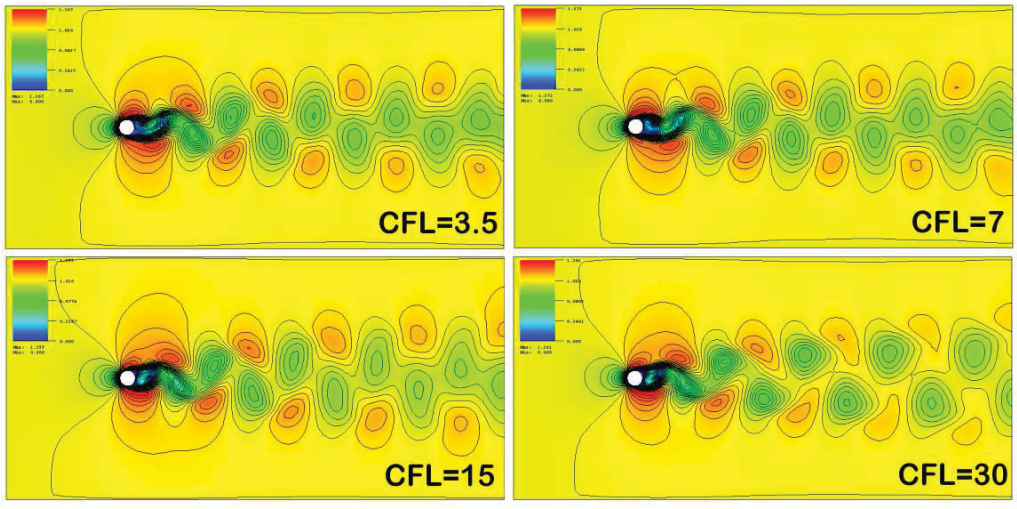

Fig. 7.5 : Convergence of velocity field for fully-implicit projection algorithm with $P^{(1)}$ pressure form and $\theta_p = 1$.

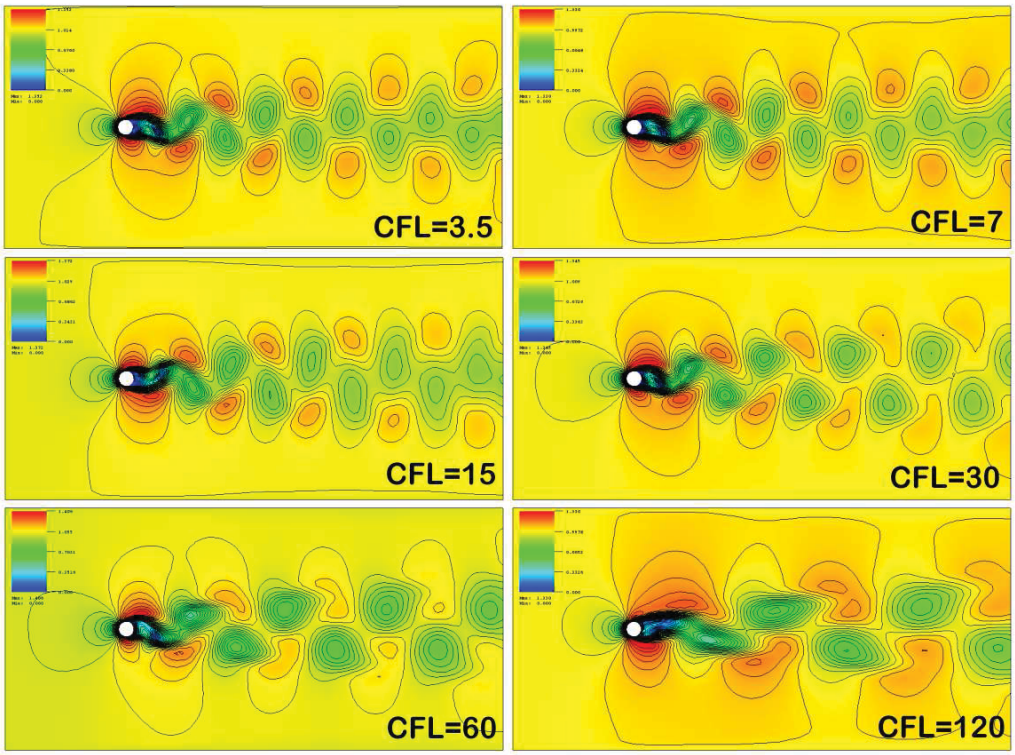

Fig. 7.6 : Convergence of velocity field for fully-implicit projection algorithm with $P^{(2)}$ pressure form and $\theta_p = 1$.

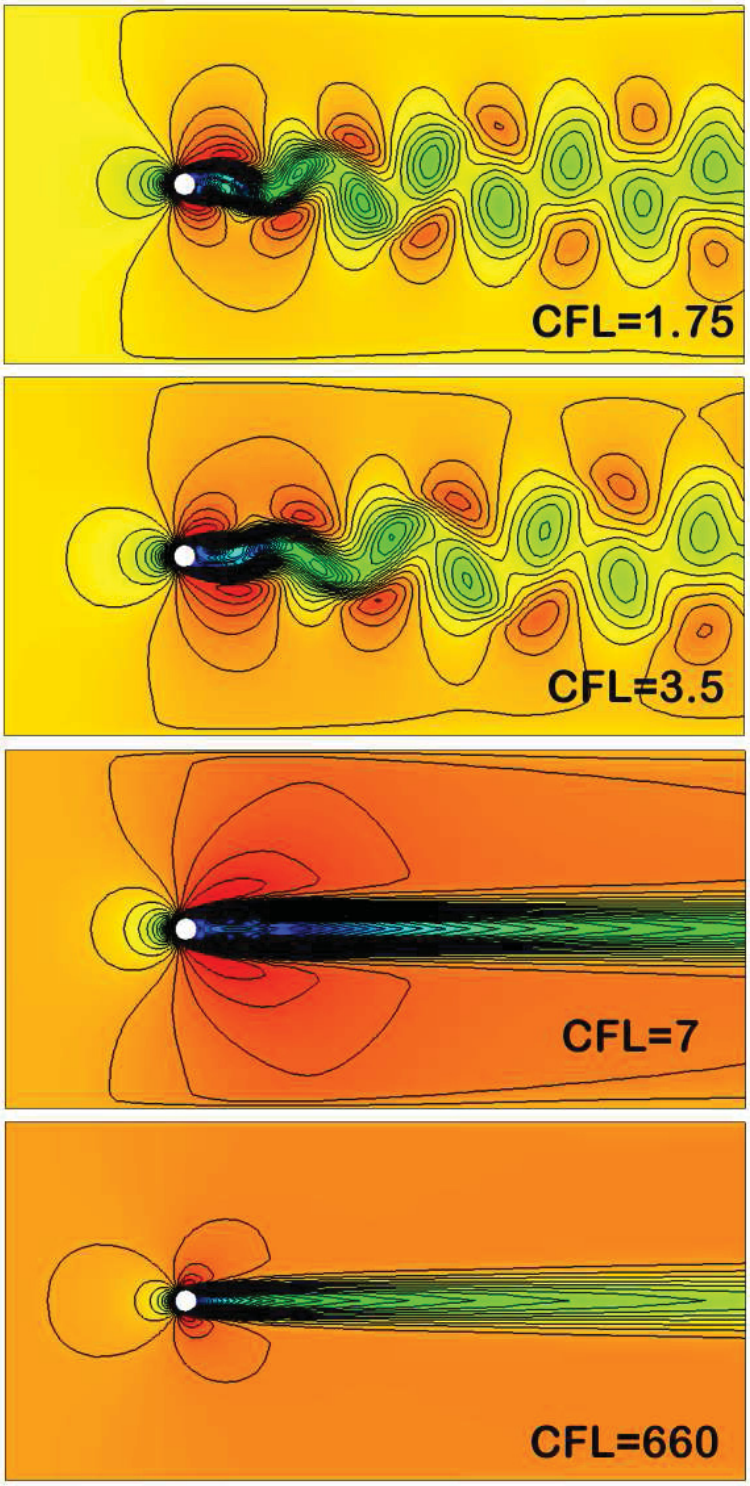

Fig. 7.7 : Convergence of the velocity field for fully-implicit projection algorithm with $P^{(0)}$ pressure form (Chorin method) and $\theta_p = 1/2$.

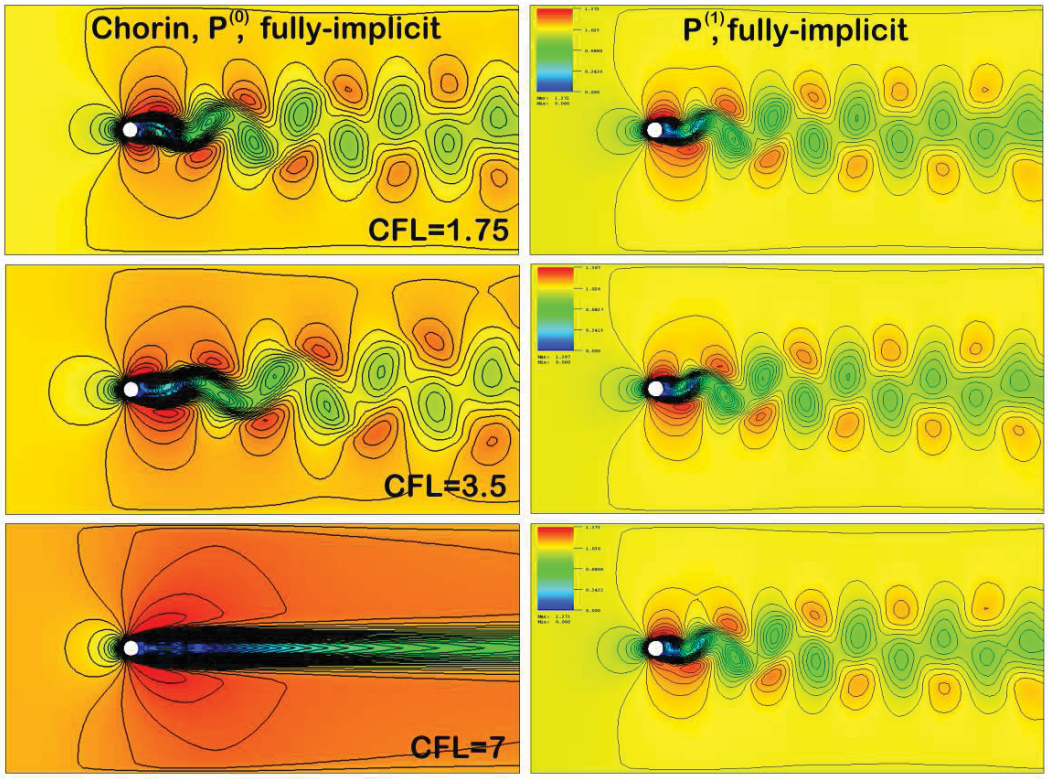

Fig. 7.8 : Comparison of velocity fields for fully-implicit projections $P^{(0)}$ (Chorin) and $P^{(1)}$, for different CFL numbers. For $P^{(0)}$, with used $\theta = 1/2$, while for $P^{(1)}$ – $\theta = 1$.

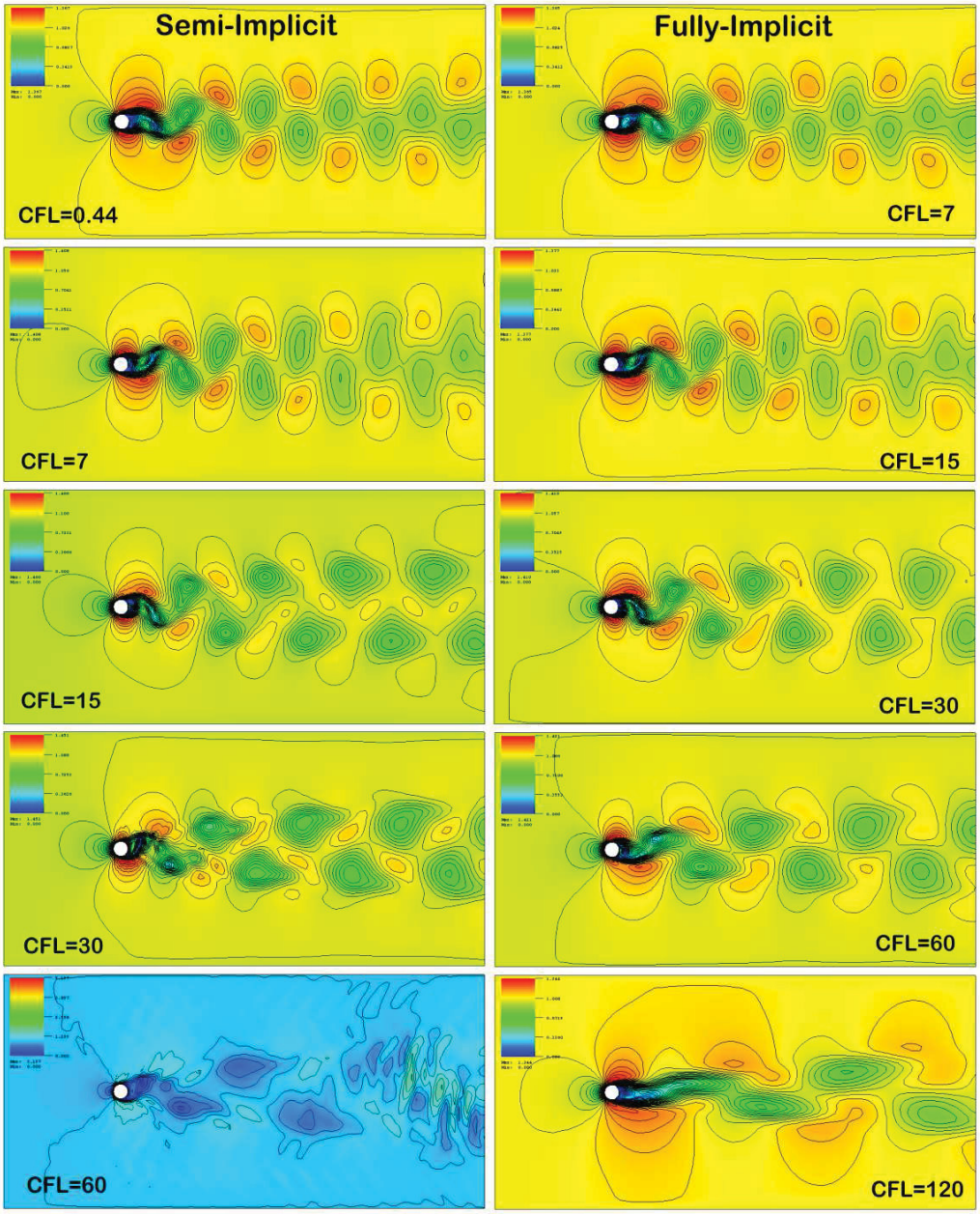

Fig. 7.9 : Comparison of semi-implicit (SI, $\theta_p = 1$, $P^{(1)}$) and fully-implicit projection (FI, $\theta_p = \frac{1}{2}$, $P^{(1)}$), on the example of vortex shedding behind a cylinder. Velocity field for $Re = 100$.

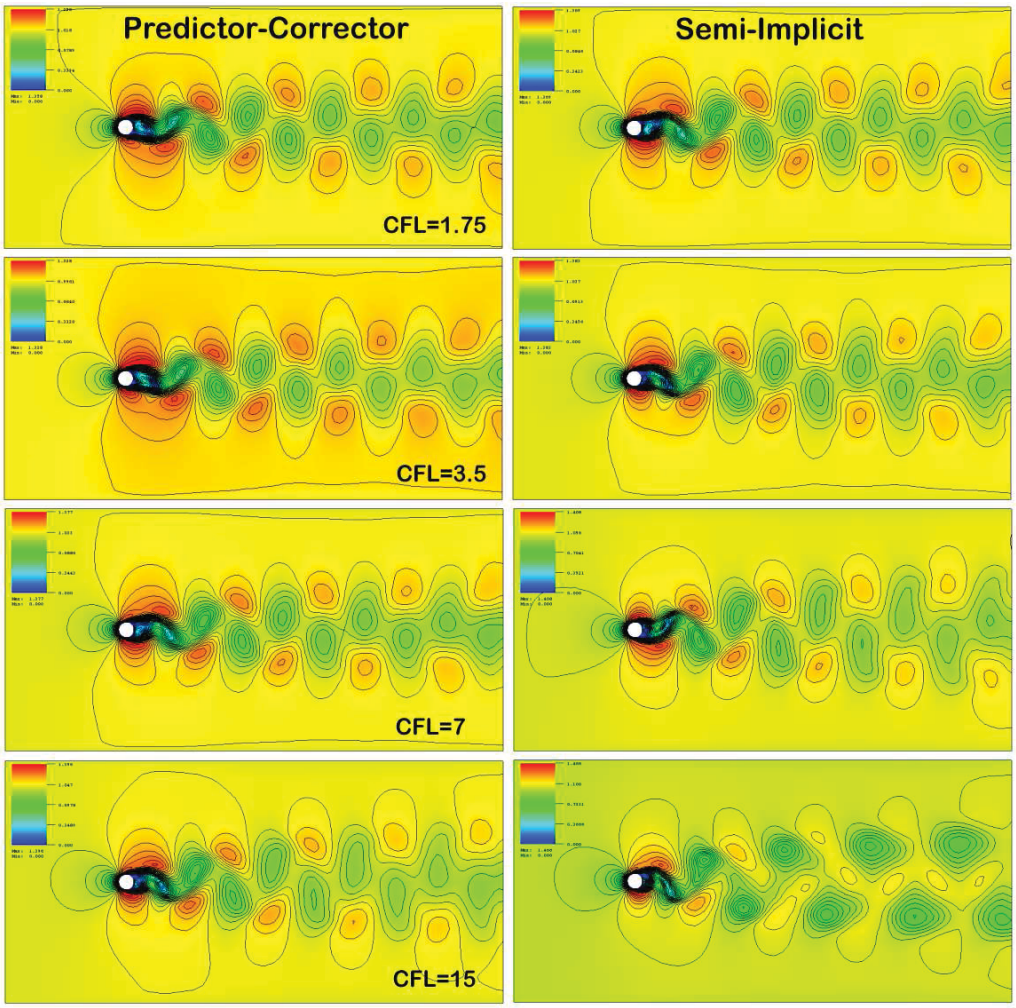

Fig. 7.10 : Comparison of the predictor-corrector (PC) and semi-implicit (SI) projection algorithms, both with $\theta_p = 1$, $P^{(1)}$). Velocity field for $Re = 100$.

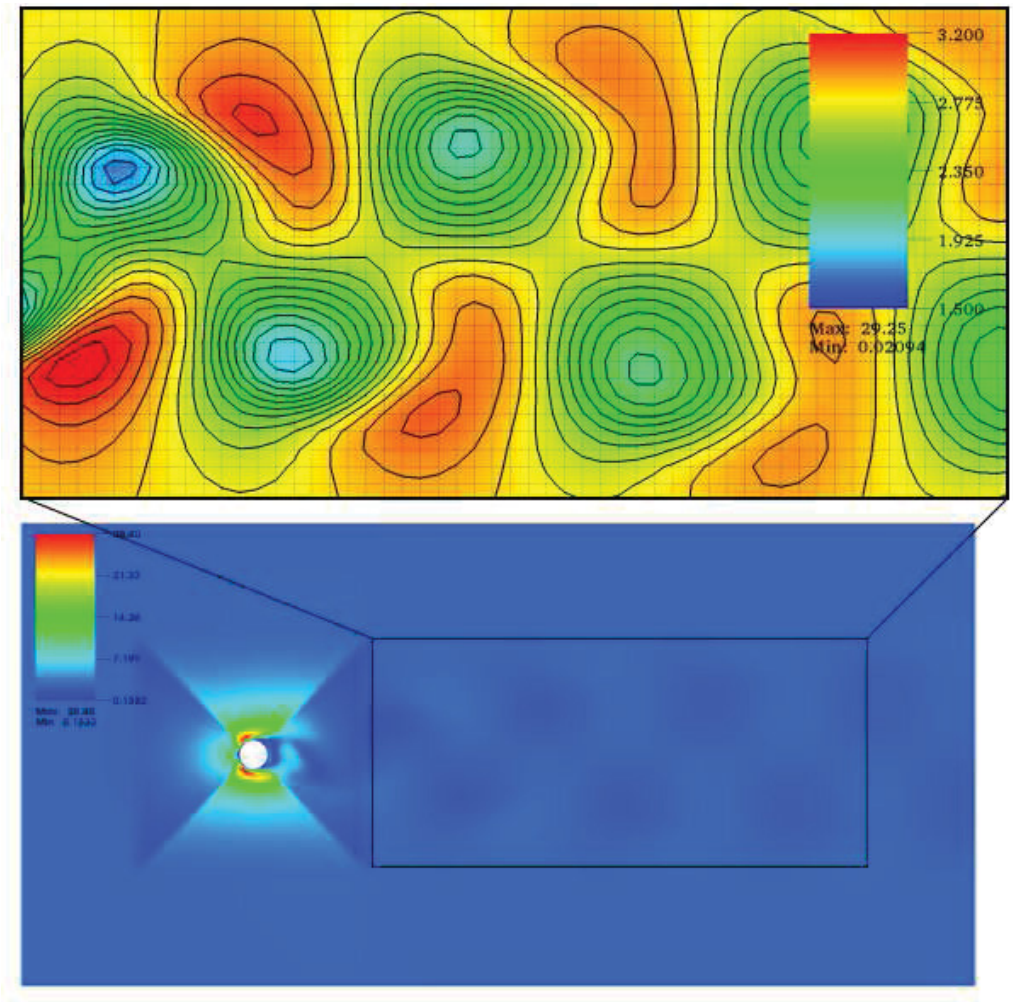

Fig. 7.11 : A snapshot of CFL distribution for for simulation with fully-implicit projection ($\theta_p = 1$, $P^{(1)}$), for the case with maximum CFL≈ 30.

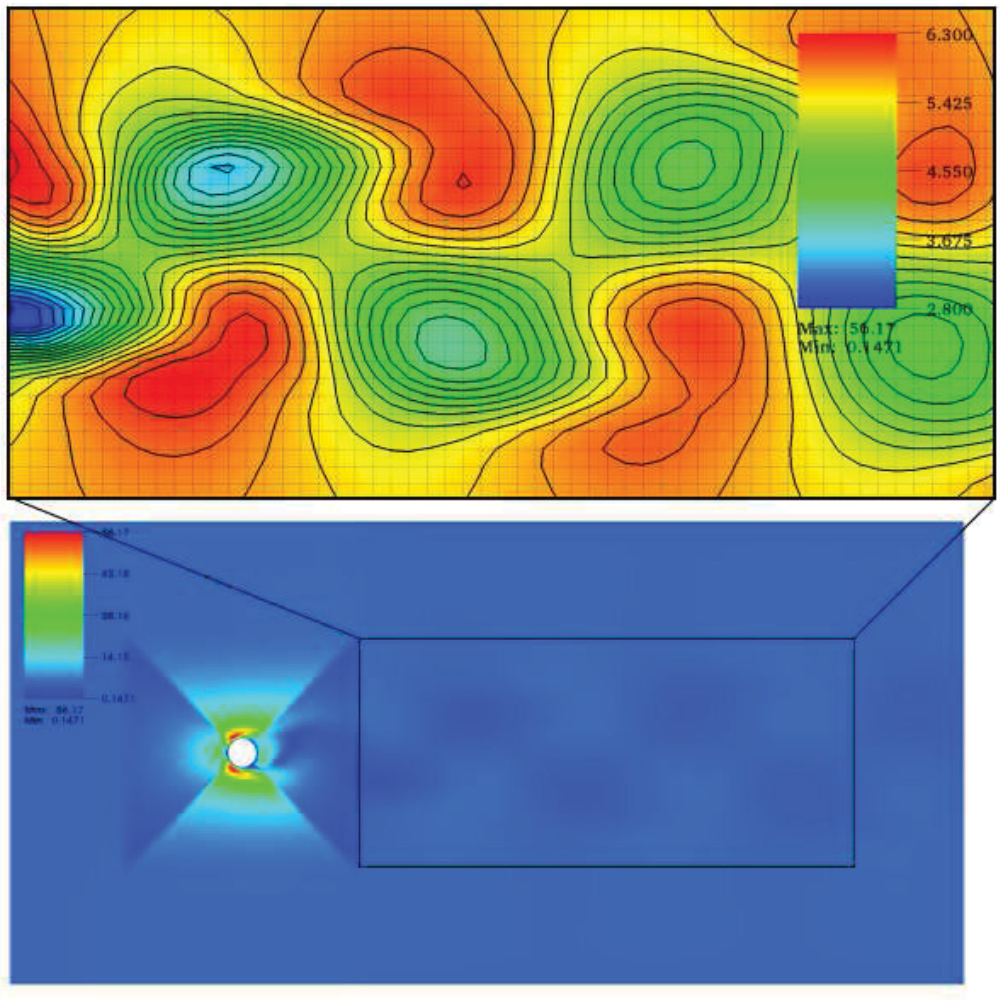

Fig. 7.12 : A snapshot of CFL distribution for for simulation with fully-implicit projection ($\theta_p = 1$, $P^{(1)}$), for the case with maximum CFL≈ 60.

7.1.2 Non-linear iteration convergence

Convergence of non-linear iterative algorithms is defined by

$$\lim_{n\to\infty} \frac{||\varepsilon_{n+1}||}{||\varepsilon_n||^r} = C^{(r)} \tag{7.1}$$

The method is said to be of order r if there is a constant $C^{(r)} > 0$, which is called *convergence rate*. In the computational results presented below, we compute the rates and order as:

Linear rate:

$$C_n^{(r)} = \frac{||\varepsilon_{n+1}||}{||\varepsilon_n||^r} \tag{7.2}$$

Order:

$$r = \frac{\ln||\varepsilon_{n+1}|| - \ln||\varepsilon_n||}{\ln||\varepsilon_n|| - \ln||\varepsilon_{n-1}||} \tag{7.3}$$

where ε_n is an increment of the solution vector $\mathbf{V}'$ at iteration n.

In Figures 7.13-7.21, we demonstrate the performance of our Picard-based non-linear solver. In Figure 7.13, we show convergence of relative errors for the global solution vector, during the last 40 non-linear iterations (the last three to six time steps of the transient). We re-scaled the global non-linear iteration counts to get these curves overlapped. One can see clear rapid convergence of non-linear iterations, essentially independently of the problem stiffness (CFL number) – all high-CFL (> 50) solutions converge within 10-15 iterations to the chosen tolerance of 10^{-8}. The small-CFL case (CFL$= 7$) converges even more rapidly, with only 6 iterations.

Superb convergence properties can also be seen from Figure 7.14, which compares non-linear iteration counts – local (per time step) and global (over the whole transient). In general, with larger CFL, the solution time of 500 was reached faster (with less total number of non-linear iterations). It must be noted though that all simulations were started with CFL=1, and the time steps were gradually increased to the specified values. For CFL=480, only a few last steps were done with $\Delta t = 12.8$, and a significant non-linear iteration count is due to the time-stepping start-up, which should explain why the total non-linear iteration count is only 2.5-3 times smaller than for the case of CFL=60 (it is roughly 7 times smaller

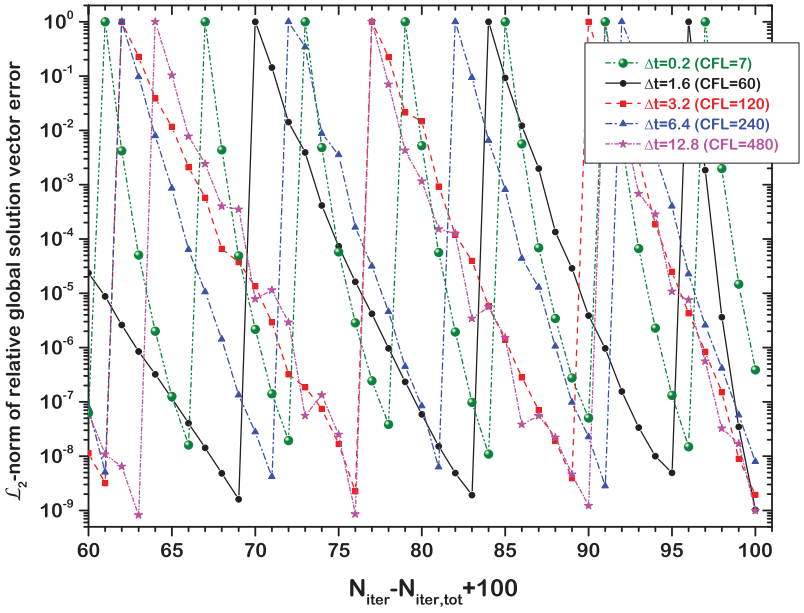

Fig. 7.13 : Convergence of non-linear iterations for different CFL numbers (last 40 non-linear iterations of the transient).

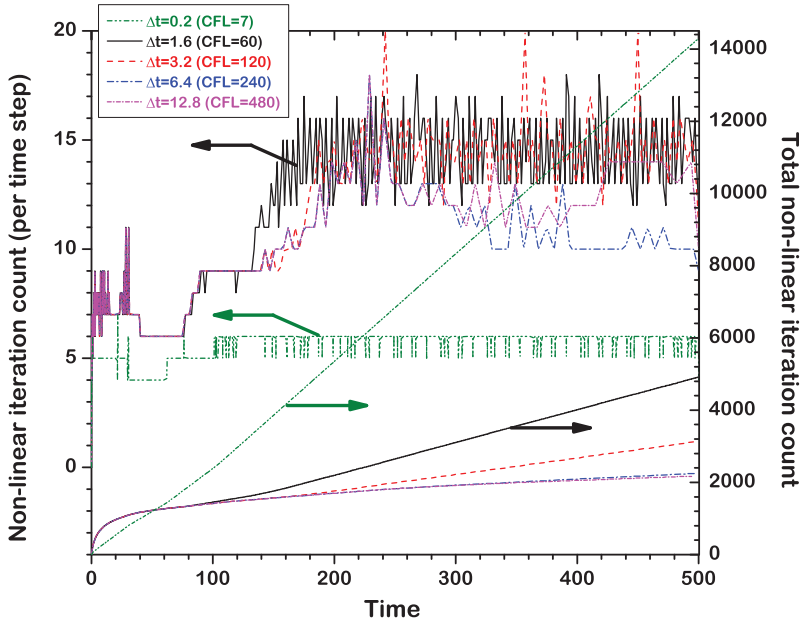

Fig. 7.14 : Convergence of non-linear iterations for different CFL numbers. Comparison of local (per time step) and global iteration counts.

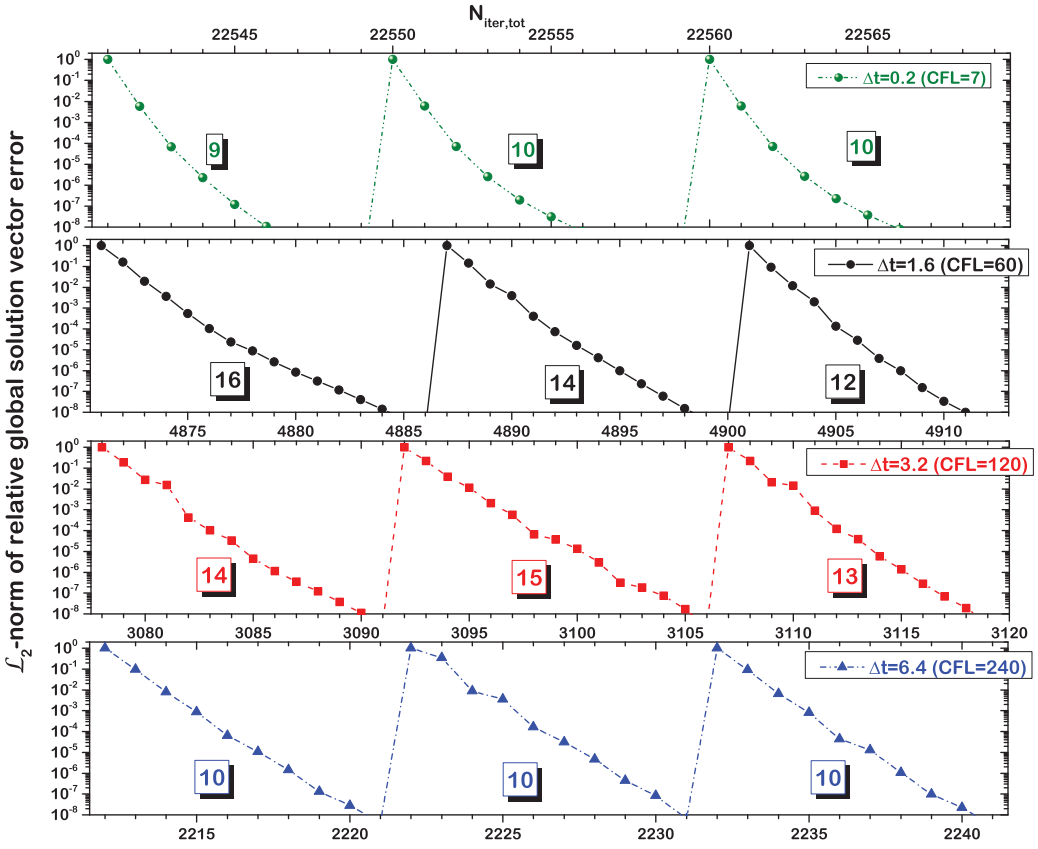

Fig. 7.15 : Convergence of non-linear iterations for different CFL numbers (last 3 time steps of the transient $t = 0, .., 500$). The non-linear iteration count per time step is shown in shadowed boxes. Tolerance is set to 10^{-8}.

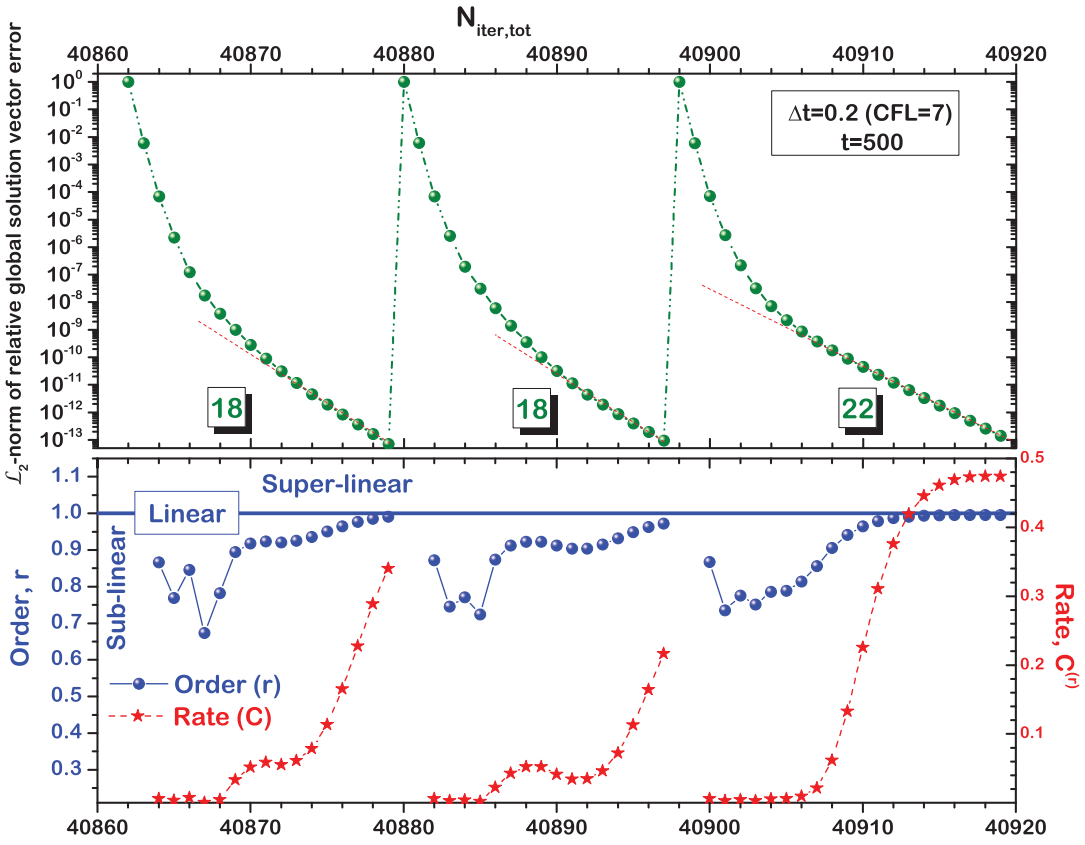

Fig. 7.16 : Convergence of non-linear iterations for CFL=7 (last 3 time steps of the transient $t = 0, .., 500$). The non-linear iteration count per time step is shown in shadowed boxes. Tolerance is set to 10^{-12}.

compared to the case of CFL=7). Also, the tolerance level was intentionally chosen to be tight. For production runs, it is reasonable to set tolerance around 10^{-5}, which, as one can see from Figure 7.13, can be reached with only three non-linear iterations, for CFL< 10. Thus, running with ten times larger time step would mean roughly 3 times faster solution.

In Figure 7.15 we show non-linear convergence history with total non-linear iteration count, during the last three time steps of the transients, for a sequence of CFL numbers 7, 60, 120 and 240. From the global non-linear iterations count, one can see that large time step solutions are faster[2], due to the fact that our non-linear convergence is scalable – i.e., independent of CFL numbers.

The measurement of the convergence order and rate is shown in Figure 7.16. We set non-linear tolerance to 10^{-12}, to get a good asymptotic convergence properties. It can be seen that the method results in monotonic (asymptotically) linear convergence with very high convergence rate[3], < 0.5. All iteration starts with very high-rate sublinear convergence, which slows down to the well-defined linear slope.

7.1.3 Comparison to SIMPLE and PISO (OpenFoam)

Next, we compare the perfomance of our algorithm to the well-established SIMPLE (Semi-Implicit Method for Pressure Linked Equations) [Pat80] and PISO (Pressure Implicit with Splitting of Operators) algorithms [Iss85, OI01], as implemented in OpenFoam open-source CFD toolbox [ope13]. Both SIMPLE and PISO are considered as work-horse algorithms in a majority of currently available commercial CFD codes.

To have a fair "apple-to-apple" comparison, we run simulations on exactly the same mesh (see Figure 7.1), with exactly the same time steps. In OpenFoam, we set time discretization based on the second-order Crank-Nicholson scheme, which is an exact counterpart of our algorithm with $\theta = \frac{1}{2}$. For space discretization, we used the second-order van Leer scheme, which should be comparable to our space discretization. In OpenFoam, all variables are collocated, with Rhie-Chow inter-

[2] The cost of each non-linear iteration is comparable, for any CFL number.

[3] At $C^{(r)} = 1$, the method is non-convergent, at $C^{(r)} > 1$ – it is disconvergent, and at $C^{(r)} < 1$ – it is convergent. The smaller $C^{(r)}$, the faster is the convergence rate.

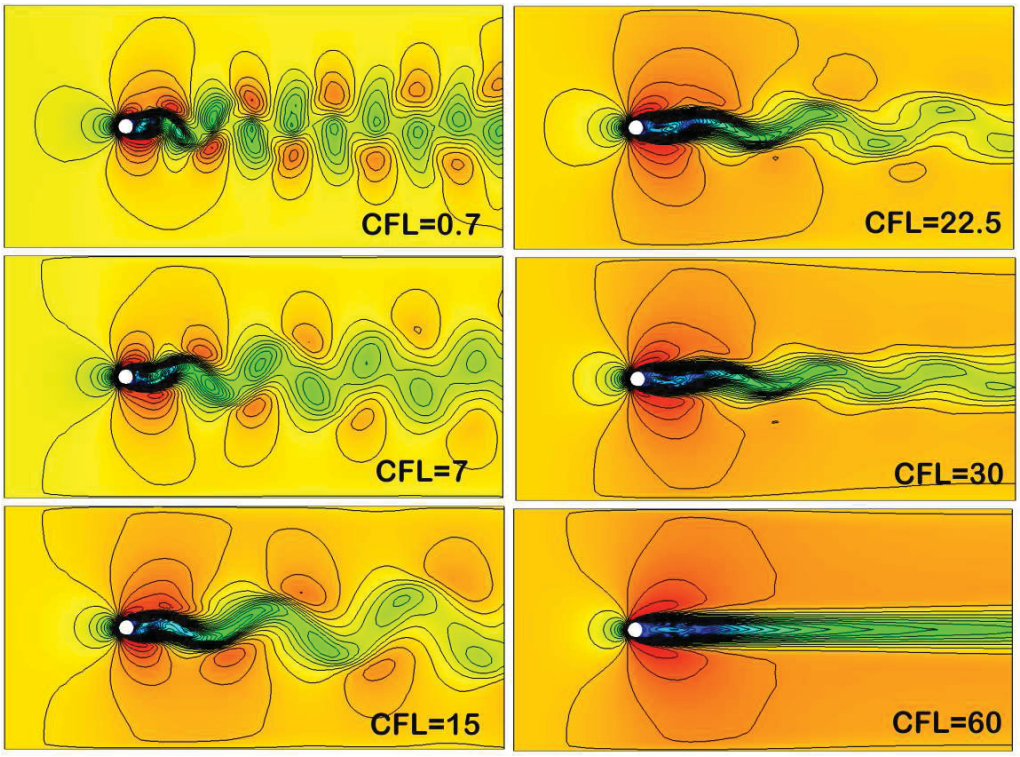

Fig. 7.17 : Convergence of velocity field for PISO algorithm.

polation [RC82] used to eliminate null-space degeneracy. This is different from Hydra-TH strategy, with cell-centered flow variables and nodal representation of the pressure field.

Furthermore, in OpenFoam, we coded computation of non-linear iteration errors and convergence criteria to be exactly the same as in our Picard-iteration algorithm, Section 5.1.5. For linear solves of momentum equations at the first PISO step, we used PBiCG – Preconditioned Bi-Conjugate Gradient solver for asymmetric matrices (vs. PETSC-based ILU-FGMRES in Hydra-TH), while for solving pressure equation, we used GAMG – generalised Geometric-Algebraic Multi-Grid solver (in Hydra-TH – PETSC/ML-based). All linear tolerances are set to 10^{-12}.

First, we show convergence of velocity field, for SIMPLE and PISO algo-

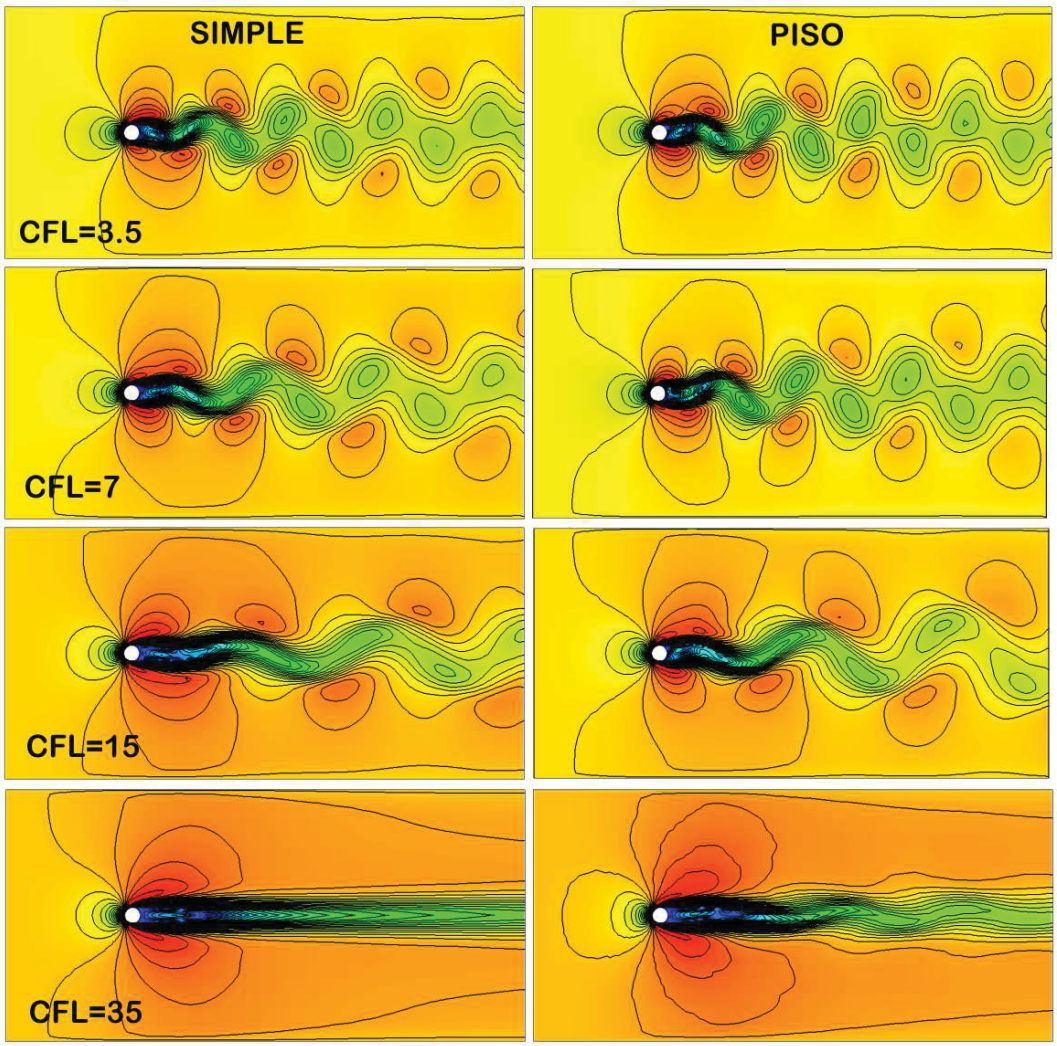

Fig. 7.18 : Comparison of velocity fields for SIMPLE vs. PISO solution, for different CFL numbers.

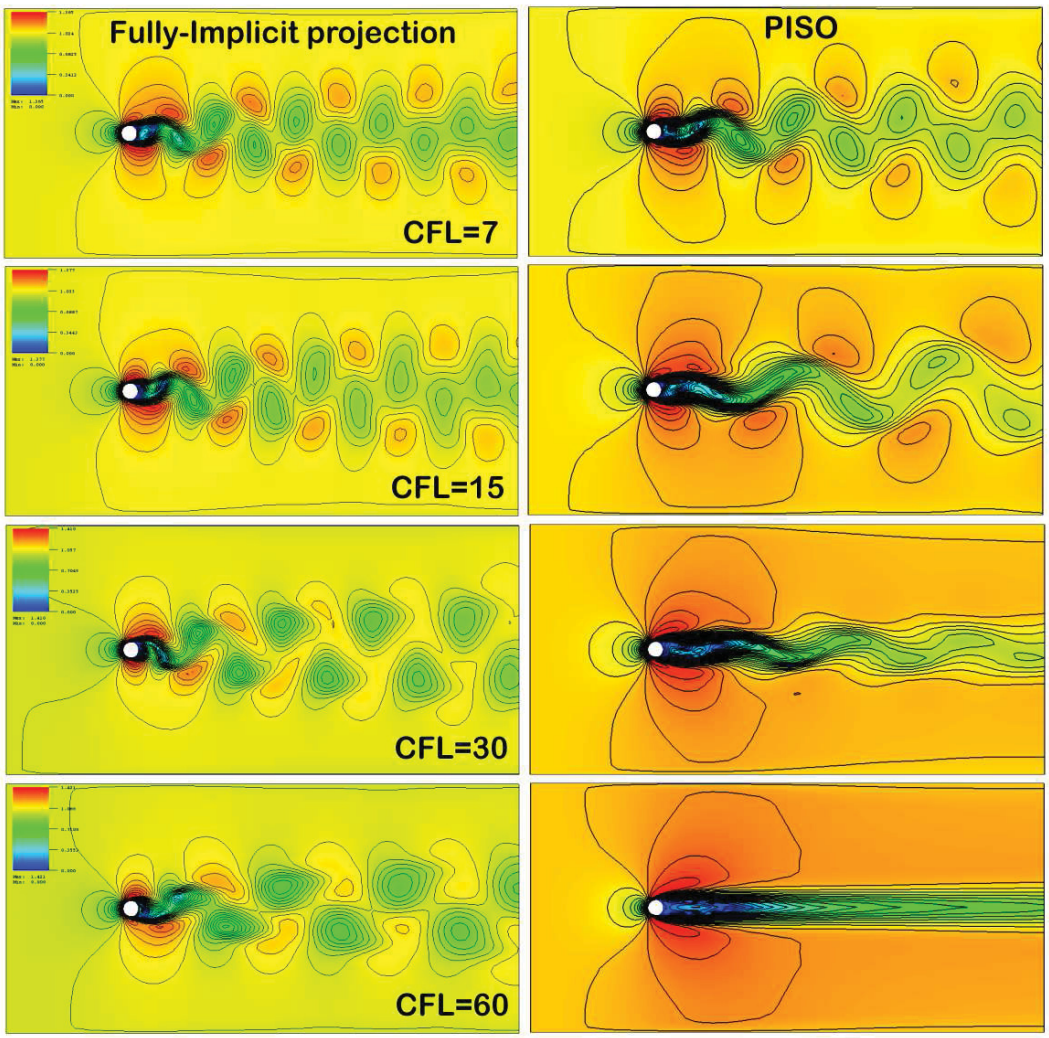

Fig. 7.19 : Comparison of velocity fields for fully-implicit projection vs. PISO solution, for different CFL numbers.

rithms, in Figures 7.17 and 7.18. At CFL=7, there is very noticeable diffusion of vortex street behind the cylinder. At CFL>30, the diffusion is so significant, that the Karman street cannot be resolved at all (note that PISO algorithm is slightly less diffusive than SIMPLE). With projection algorithm, vortex shedding is reasonably resolved even at CFL>120. Direct comparison of PISO to projection-based algorithm, for different time steps (CFL number), is shown in Figure 7.19.

Next, we present the performance of the PISO vs. fully-implicit projection, as preconditioners of Picard iterations, in Figures 7.20 and 7.21. First, one can notice oscillatory (non-monotonic) convergence of PISO at the begining of iterations. Eventually, the method converge linearly (as expected), but with very slow convergence rate, $C^{(1)} \rightarrow 1$. It takes approximately five times more iterations than in the case of fully-implicit projection, for CFL=7; and ten times more for the case of CFL=60. Each iteration of PISO though is cheaper, as after the first iteration, it replaces the full implicit momentum solves with cheap lagged ("splitted") diagonal solves, which is obviously very impactful (detrimental) on the non-linear convergence of the algorithm. Moreover, the convergence is non-scalable, i.e. with higher CFL, it requires much more non-linear iterations to reach the same tolerance level.

It must be noted that in most implimentations of PISO, the algorithm is used in an operator-splitting/predictor-corrector manner, running only a few (2-3) iterations, without any attempt to achieve tight nonlinear convergence. This would bring non-linear tolerance to the level of 10^{-2}, for the considered here problem (vs. $\approx 10^{-2}$, for Hydra-TH). If the non-linearity of the problem is very significant (as exhibited in many multiphase flow problems), the performance of PISO (and SIMPLE-based algorithms) is extremely poor, as exhibited by the evidence of practical application for all IPSA-derived [Spa83] multi-fluid solvers.

Table 7.1 : Performance of the algorithms, for vortex shedding problem

METHOD	CFL	CPU-time, sec
PISO	7	11,750
FI-projection	7	6,584
PISO	60	3,208
FI-projection	60	1,249

Finally, we show CPU-time comparison in Table 7.1. While not exactly a fair

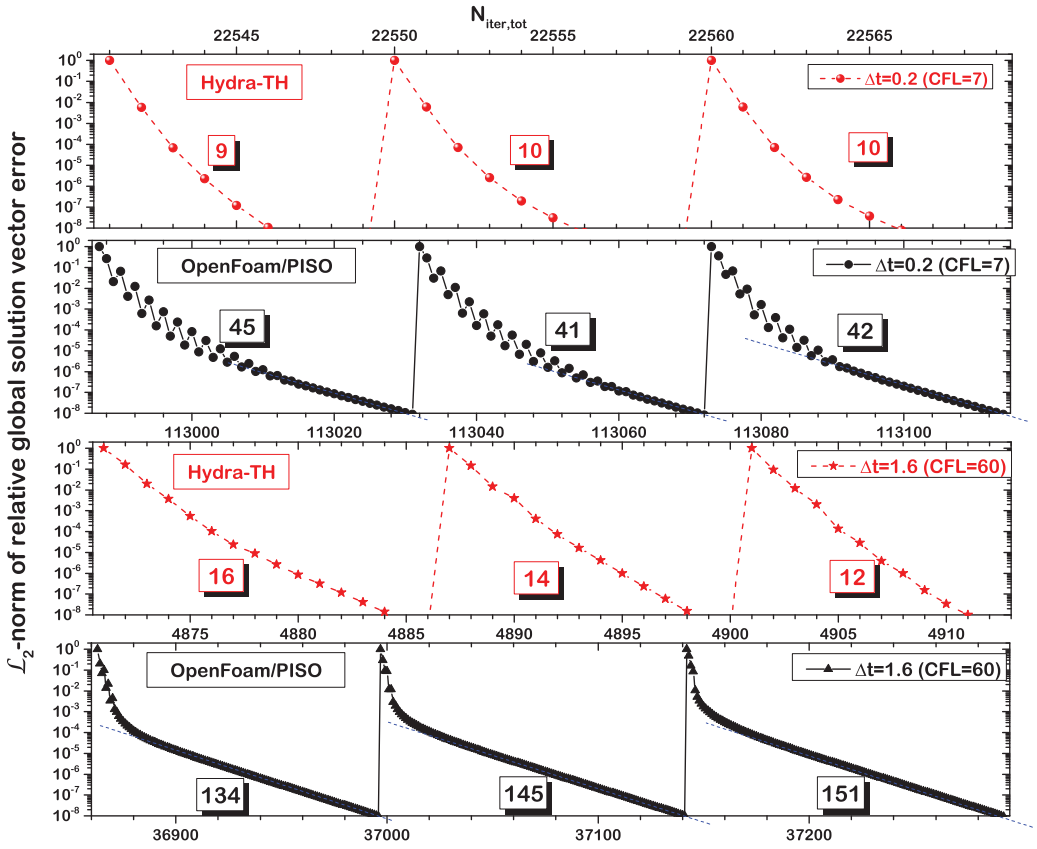

Fig. 7.20 : Comparison of the convergence of non-linear iterations – Hydra-TH's Picard algorithm vs. OpenFoam's PISO algorithm. Last 3 time steps of the transient $t = 0, .., 500$. The non-linear iteration count per time step is shown in shadowed boxes.

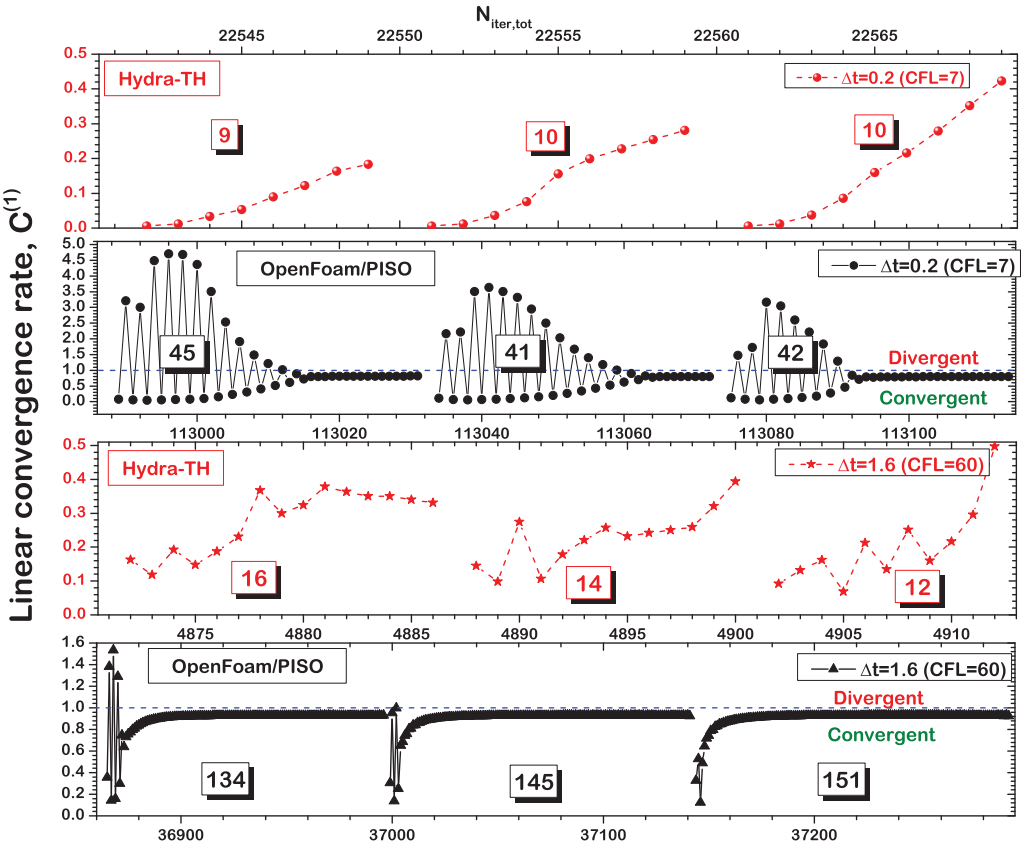

Fig. 7.21 : Comparison of the linear convergence rates for non-linear iterations – Hydra-TH's Picard algorithm vs. OpenFoam's PISO algorithm. Last 3 time steps of the transient $t = 0, .., 500$. The non-linear iteration count per time step is shown in shadowed boxes.

apple-to-apple comparison (as different linear solvers and implementations are used), this nevertheless should give a decent estimate on expected performance of the solution algorithms. The tests were run on Mac laptop (Darwin, 11.4.2 Lion OS), with two-core $i7$ 2.66 GHz processor. As one can see, for CFL=7, the projection-based algorithm in Hydra-TH is approximately twice as fast, compared to the PISO-based OpenFoam solution. For larger CFL=60, Hydra-TH is 2.5-3 times faster.

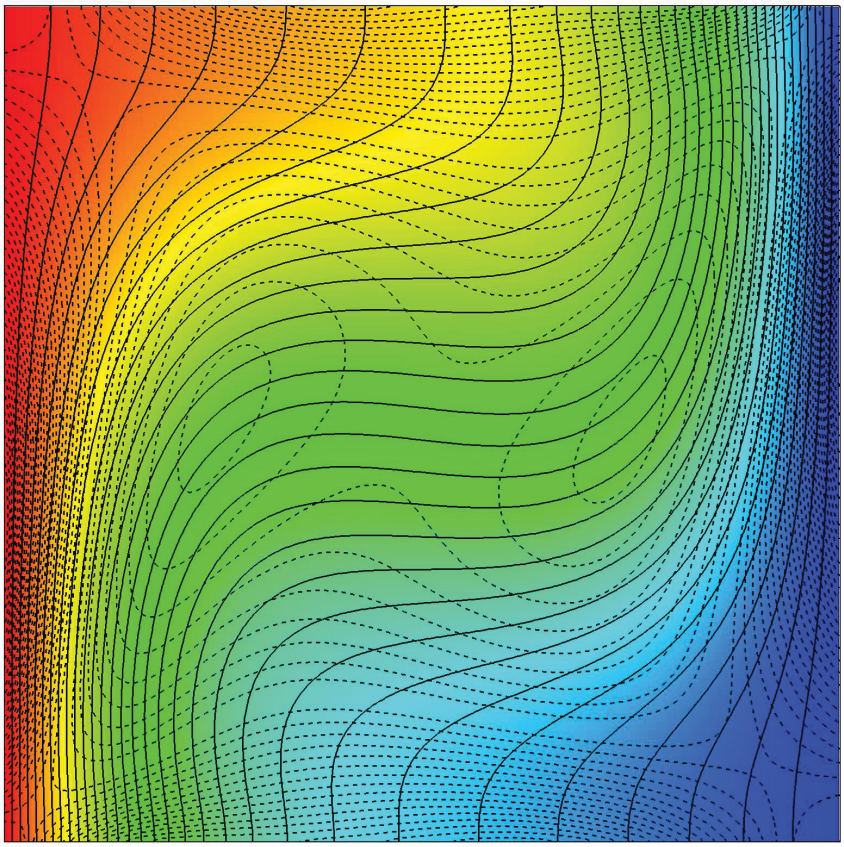

Fig. 7.22 : Steady-state solution. $Ra = 10^4$, $Pr = 0.71$. Temperature field (color map and solid isolines) and contours of vorticity (dashed isolines).

7.2 Natural Convection in a Square Cavity with Oscillatory Temperature Boundary Conditions

In the next test, we consider natural convection in a square cavity, with Dirichlet (no-slip, temperature-specified) boundary conditions at the left and right wall, and adiabatic no-slip horizontal walls. Steady-state solutions for this problem are well-documented by de Vahl Davis in [dJ83, de 83]. Steady-state temperature and vorticity field solutions are shown in Figure 7.22.

We modified this test to make it transient as follows. Temperature at the right

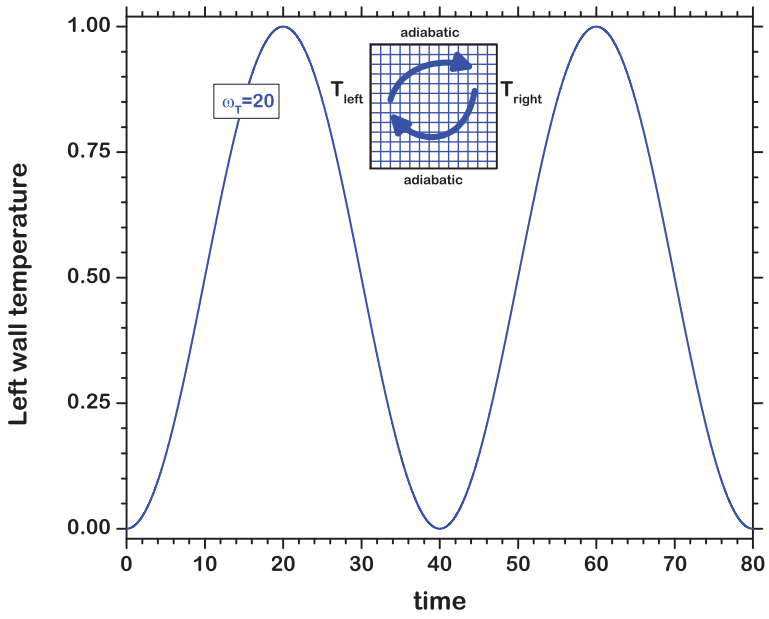

Fig. 7.23 : On the formulation of the oscillating-temperature natural convection test.

wall is fixed, $T_{\text{rgt}} = 0$. The temperature at the left boundary is varied in time as

$$T(t) = \frac{1}{2}\left(\sin\left(\frac{t}{\omega_T}\pi - \frac{\pi}{2}\right) + 1\right) \tag{7.4}$$

where ω_T is the period of oscillations, Fig.7.23. Simulations are started with motionless fluid at $T_{\text{init}} = 0$. At the peak of left-wall temperature, $Ra = 10^4$. Prandtl number is set to $Pr = 0.71$. Grid resolution is set to $\Delta h = \frac{1}{320}$, providing fine-enough resolution, allowing to focus on time discretization errors. For all simulations presented, we set $\theta = \frac{1}{2}$ for advection, diffusion and body-force operators. The parameter θ_p was varied from $\frac{1}{2}$ to 1, as well as pressure forms ($P^{(0)}$, $P^{(1)}$ and $P^{(2)}$). Computational results are summarized in Figures 7.24-7.30.

Figures 7.24 and 7.25 depict time history of temperature and vorticity fields, for two solution algorithms – semi-implicit, and fully-implicit, with fixed time step of $\Delta t = 1$. This corresponds to maximum CFL=87, Fo_ν=863 and Fo_α=1215, were Courant and Fourier (viscous and thermal) numbers are defined as

$$\text{CFL} = \frac{|\mathbf{v}|_{\max}\,\Delta t}{\Delta h}$$

and

$$\begin{aligned} \text{Fo}_\nu &= \frac{\nu \Delta t}{\Delta h^2} \\ \text{Fo}_\alpha &= \frac{\alpha \Delta t}{\Delta h^2} \end{aligned}$$

respectively.

Figure 7.26 is a comparison of semi-implicit and fully-implicit projection for vorticity field at $t = 10$. It can be clearly seen that the fully-implicit algorithm is superior to the semi-implicit, showing highly-accurate solutions for very large time steps/(CFL and Fourier numbers). Moreover, when CFL and Fo numbers are high, the semi-implicit algorithm results in so-called ***"projection boundary layers"***, clearly seen by plotting isolines of the vorticity field. These artifacts are the results of ignoring advection/diffusion of irrotational part of velocity field, $\boldsymbol{\Xi}_t$ (see eq.(5.39)), in the semi-implicit algorithm. While not affecting directly the convergence rate of the algorithm (see Figures 7.27-7.30), they are significant and visible near domain boundaries. In the fully-implicit algorithm, these terms are accounted for during non-linear iteration loop. We performed fully-implicit simulations were these terms are ignored, and as one can see from Figure 7.26 (second row) – the "projection boundary layers" are present.

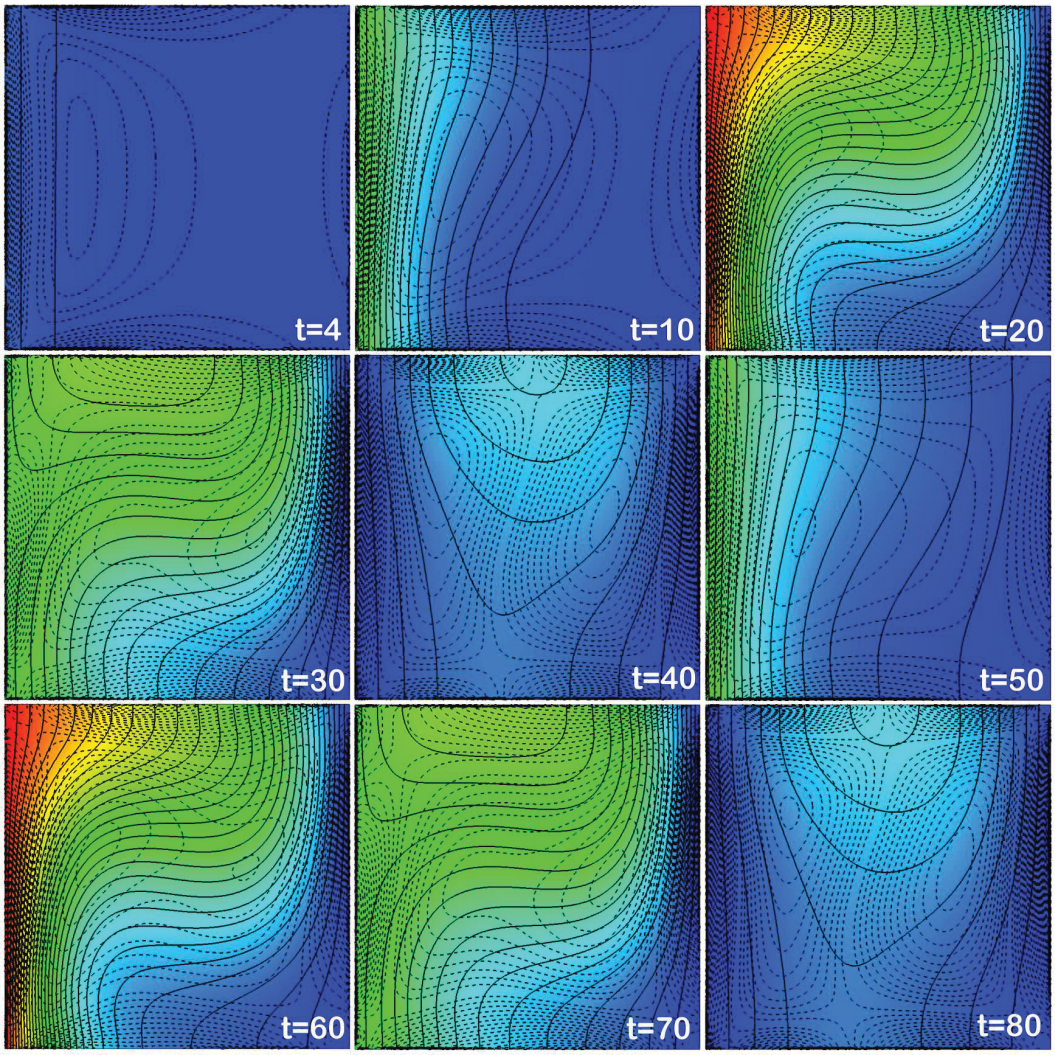

Fig. 7.24 : Dynamics of temperature (color-map and thick solid isolines) and vorticity magnitude (thin dashed isolines), using the semi-implicit projection (pressure gradient form, $P^{(1)}$, $\theta_p = \frac{1}{2}$), with time step $\Delta t = 1$.

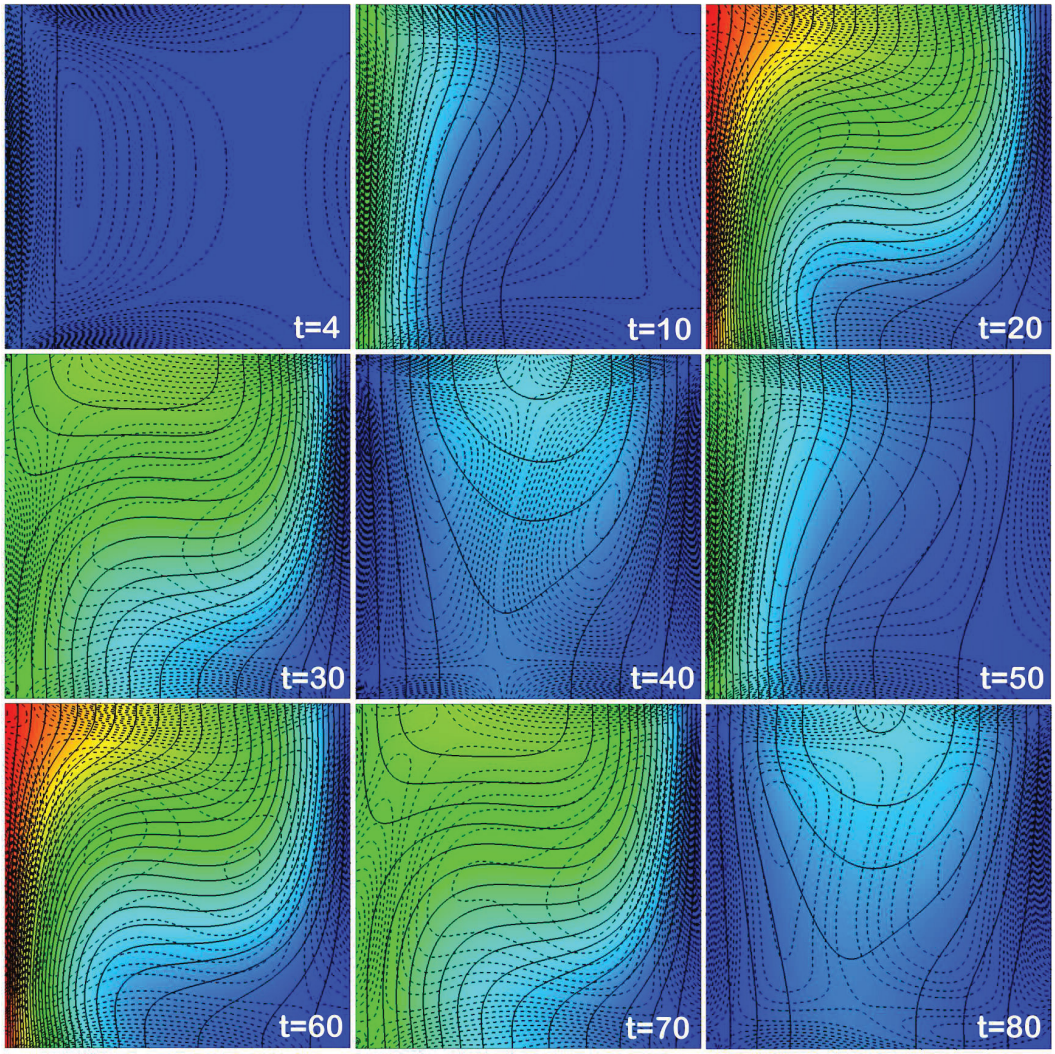

Fig. 7.25 : Dynamics of temperature (color-map and thick solid isolines) and vorticity magnitude (thin dashed isolines), using the fully-implicit (Picard-based, pressure curvature form, $P^{(2)}$ $\theta_p = \frac{1}{2}$) projection, with time step $\Delta t = 1$.

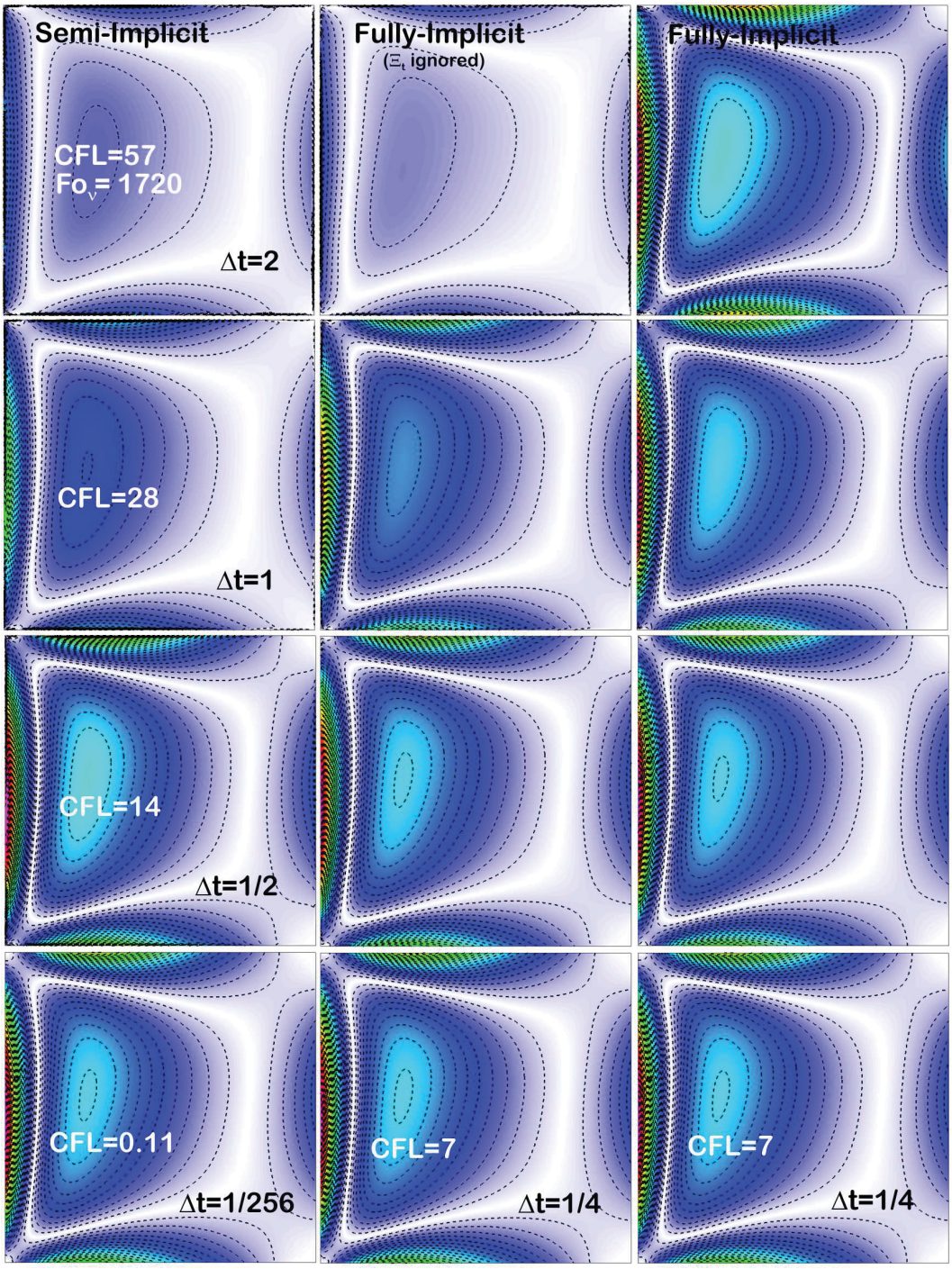

Fig. 7.26 : Time convergence for vorticity field, comparing semi-implicit (pressure gradient form, $P^{(1)}$, $\theta_p = \frac{1}{2}$) and fully-implicit algorithms (pressure curvature forms, $P^{(2)}$, $\theta_p = \frac{3}{5}$), for time $t = 10$.

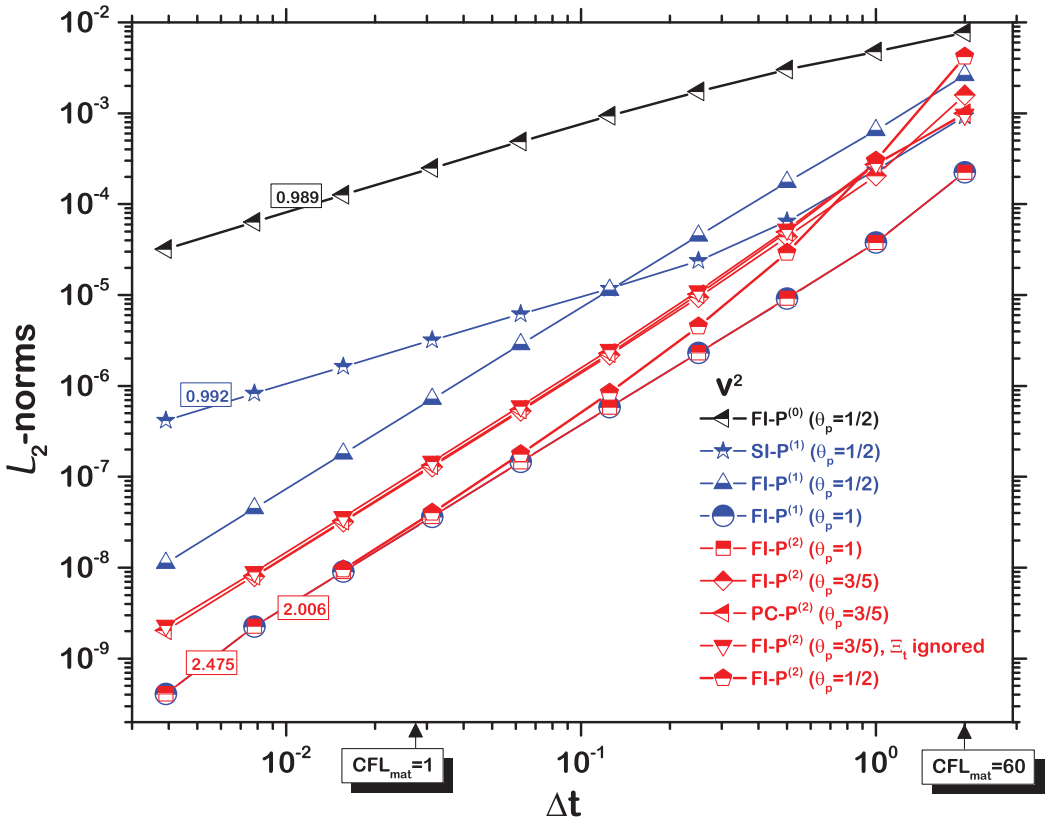

Fig. 7.27 : Time convergence of $\mathcal{L}_2$-norm of errors for kinetic energy.

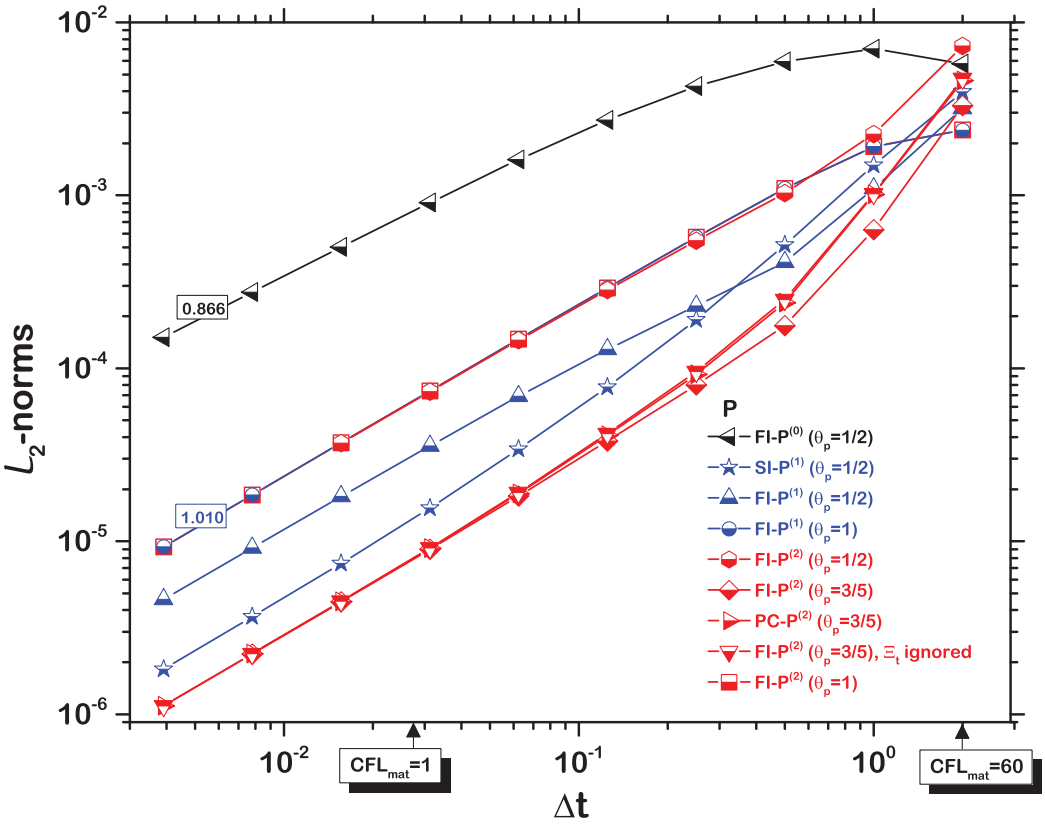

Fig. 7.28 : Time convergence of $\mathcal{L}_2$-norm of errors for pressure.

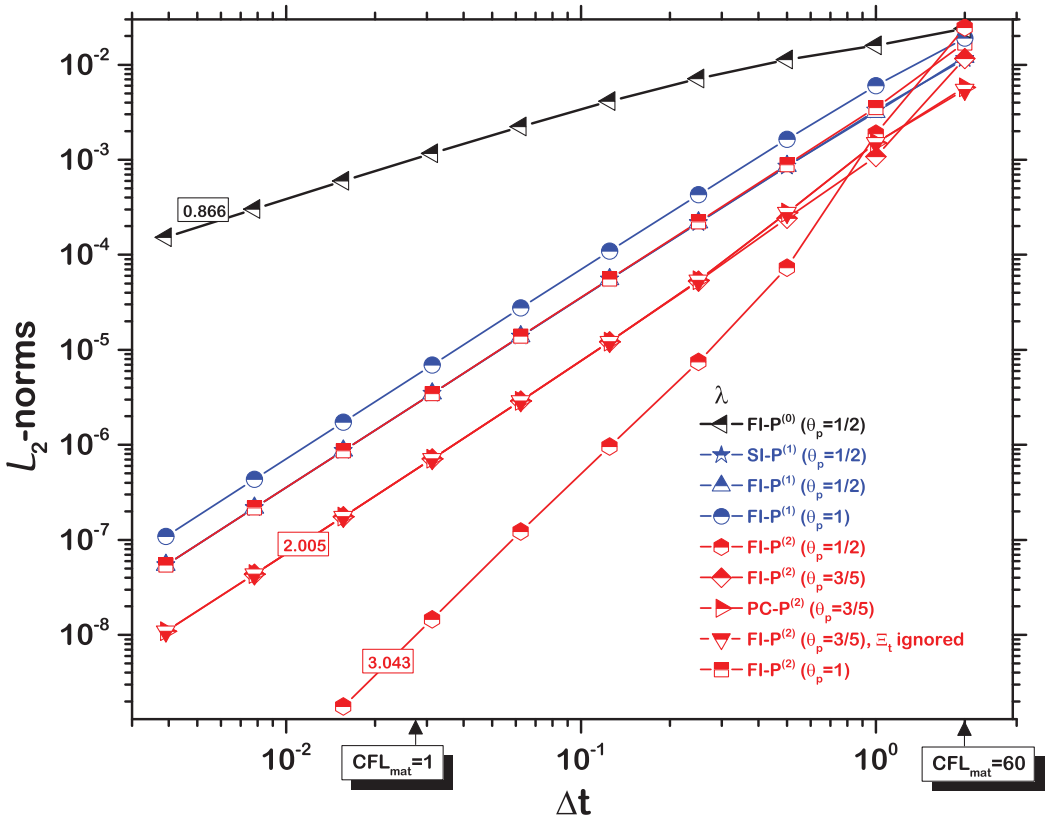

Fig. 7.29 : Time convergence of $\mathcal{L}_2$-norm of errors for Lagrange multiplier.

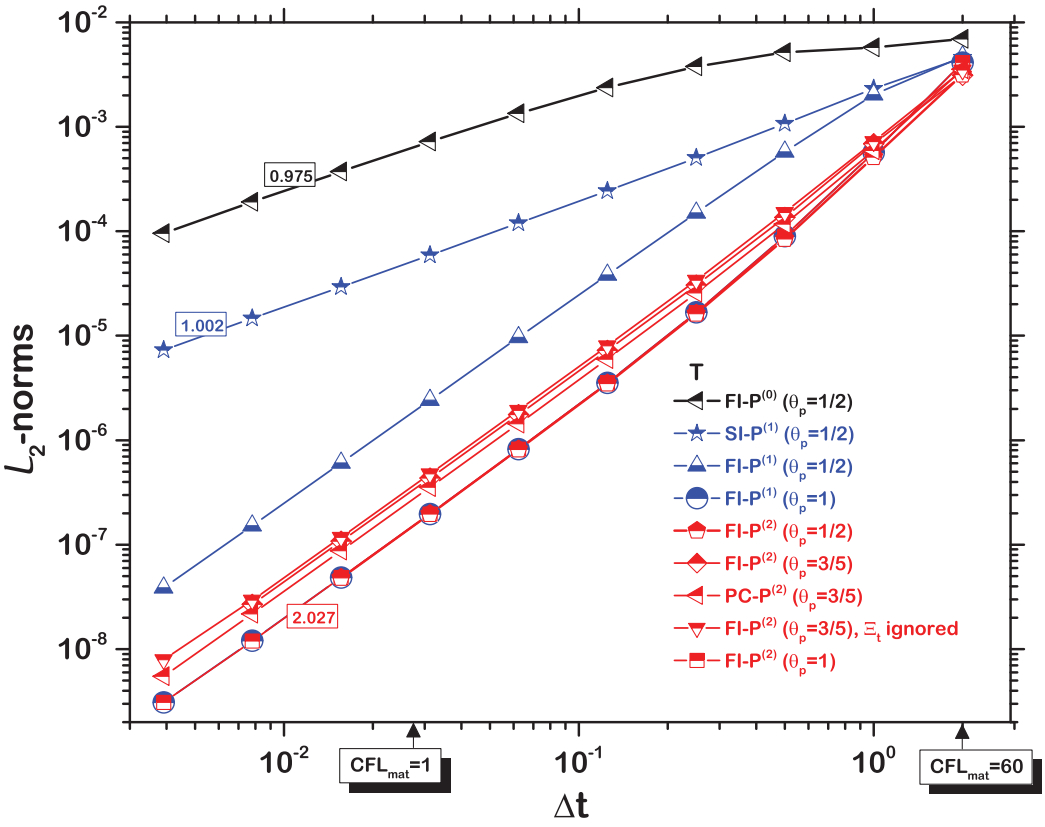

Fig. 7.30 : Time convergence of $\mathcal{L}_2$-norm of errors for temperature.

Finally, we show the measured convergence rates for velocity, pressure, Lagrange multiplier and temperature, in Figures 7.27-7.30. All semi-implicit method solutions are showing asymptotically the 1^{st}-order convergence, due to operator splitting errors on the solution slopes when computing advection operator (being "frozen" at t_n). In the fully-implicit algorithm, we removed these errors, resulting in clear the 2^{nd}-order convergence for velocity and temperature fields, Figures 7.27 and 7.30. Pressure however is always only 1^{st}-order accurate, Figure 7.28, regardless of the used pressure form. Under the same θ_p, the pressure curvature form $P^{(2)}$ is generally more accurate than the pressure gradient form $P^{(1)}$. Importantly to note here, that the Lagrange multiplier evolves with high-order accuracy in time, clearly exhibiting the 2^{nd} (for pressure gradient form, $P^{(1)}$) and the 3^{rd} (for pressure curvature form, $P^{(2)}$) order convergence rates, Figure 7.29. This is the key to have the 2^{nd} convergence for velocity field. As one can see – in the case of "pressure form" (the original Chorin's projection algorithm), the Lagrange multiplier is only 1^{st} order accurate, resulting in the 1^{st} order accurate velocity and temperature.

It is interesting to note that just applying two Picard iterations in a "predictor-corrector" (PC) manner – is sufficient to get the 2^{nd} order convergence for velocity and temperature, Figures 7.27 and 7.30. This does remove/(reduce) operator splitting errors in treatment of advection operator. It will not however ensure unconditionally stable solution. Also, "projection boundary layers" are not removed, see Figure 7.31.

An example of the non-linear iteration convergence is shown in Figures 7.32 and 7.33, for $P^{(1)}$, $\theta_p = \frac{1}{2}$. The errors are rapidly dropped to the chosen tolerance level of 10^{-8}, within 10 iterations. Convergence rate is asymptotically linear, with very high convergence rates $\approx 0.12 \ll 1$, Figure 7.33.

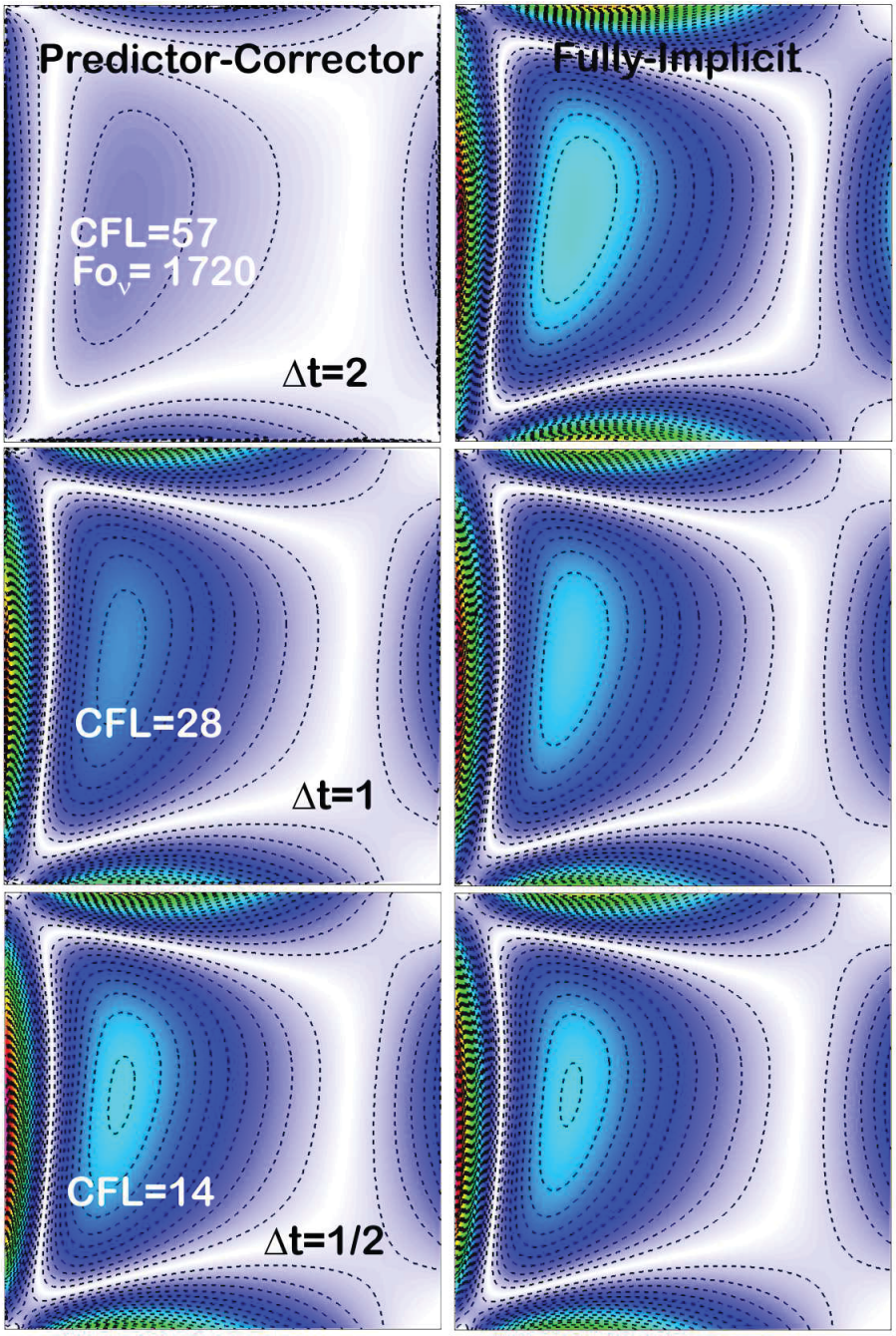

Fig. 7.31 : Time convergence for vorticity field, comparing predictor-corrector (PC) and fully-implicit algorithms (both with pressure curvature forms, $P^{(2)}$, $\theta_p = \frac{3}{5}$), for time $t = 10$.

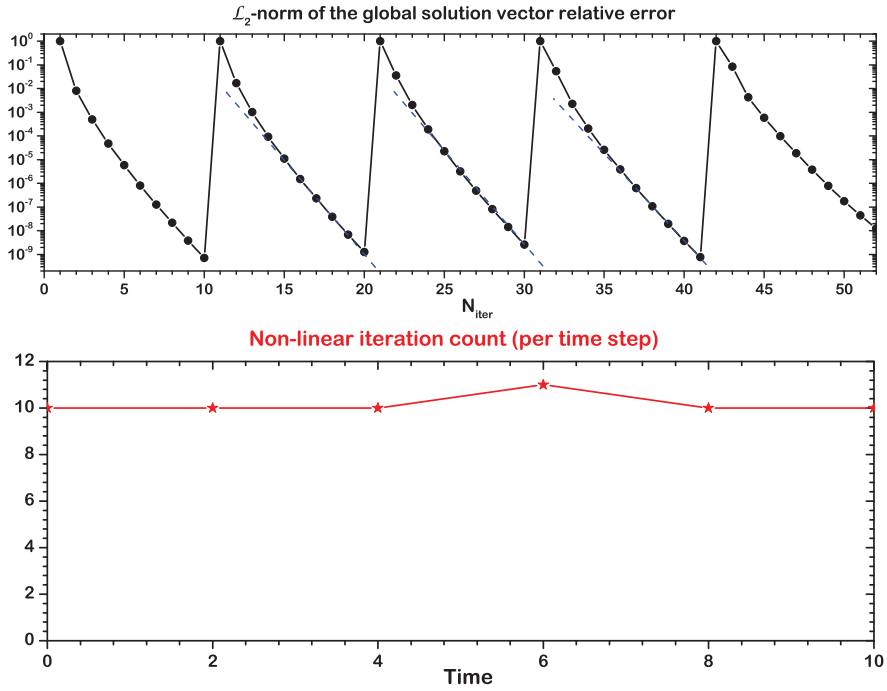

Fig. 7.32 : Convergence of non-linear iterations for the case of fully-implicit algorithm with $P^{(1)}$, $\theta_p = \frac{1}{2}$, with time steps $\Delta t = 2 \times 5$.

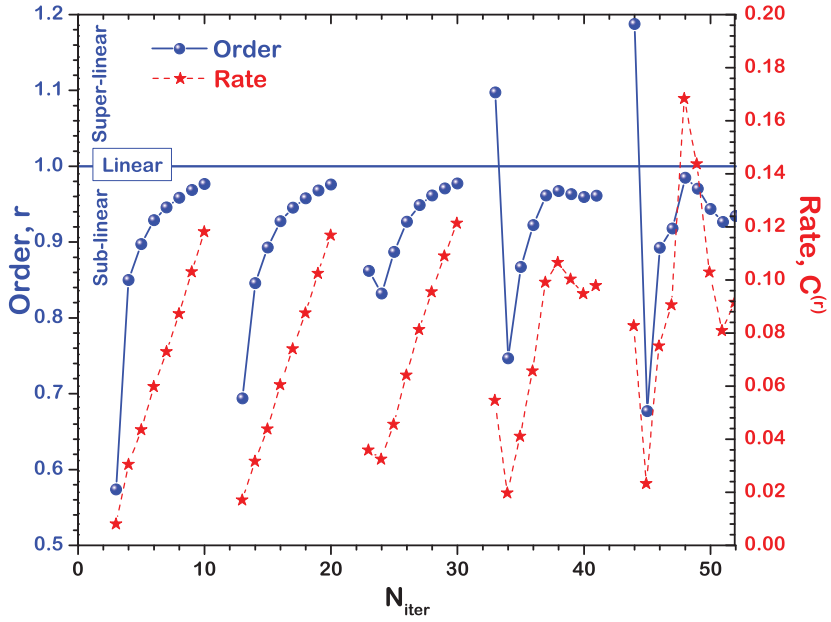

Fig. 7.33 : Convergence order and rate of non-linear iterations.

Chapter 8

Concluding Remarks

THE main technical contribution of the present report is the formulation of the fully-implicit projection algorithm for implementation in Hydra-TH code. We discussed definition of non-linear residual vector, as well as the strategy for efficient preconditioning of linear (GMRES) solver, utilizing the variation of the currently-available in Hydra-TH semi-implicit projection algorithm. While focusing here on single-phase flow formulation, the basic ideas of the fully-implicit projection should be straightforwardly extandable to multi-fluid flows. These extensions will be presented in future.

This Page is Intentionally Left Blank

APPENDICES

This Page is Intentionally Left Blank

Appendix A

Extension to Compressible Flows

In this appendix, we discuss how to extend the fully-implicit projection algorithm to fully-compressible flows, along the lines of Harlow and Amsden [HA68, HA71, HA75b, HA75a] all-speed *"Implicit Continuous-fluid Eulerian (ICE)"* algorithm.

The governing equations are *mass*,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \tag{A.1}$$

momentum,

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v}) = -\nabla P + \mathbf{S}_{\mathbf{v}} \tag{A.2}$$

and *energy conservation* (written in terms of specific internal energy, u):

$$\frac{\partial \rho u}{\partial t} + \nabla \cdot (\rho u \mathbf{v}) = -P \nabla \cdot \mathbf{v} + S_u \tag{A.3}$$

complemented with a given equation of state:

$$\rho\,(P, u) \tag{A.4}$$

Let's consider decomposition of an arbitrary momentum field along the lines of *Helmholtz decomposition*:

$$\rho \overline{\mathbf{v}}^{\star} \equiv \rho \overline{\mathbf{v}} + \nabla \lambda \tag{A.5}$$

We splitted a momentum vector into the *"divergence-constrained"*[1] and the *"curl-free"* parts. Now, because of the compressibility, these are not generally orthogonal, and the level of "skeweness" is defined by the Mach number – the lower it is – the more effective the considered "projection-like" procedure is. The applications of interest here involve relatively low Mach number flows. For high-Mach flows $M > 1$, the conservative-variable (density-based) algorithms are more cost-effective.

Next, we can apply similar decomposition of energy flux vector field,

$$\rho u \overline{\mathbf{v}}^{\star} = \rho u \overline{\mathbf{v}} + u \nabla \lambda \tag{A.6}$$

To get equation for Lagrange multiplier, we first take divergence of eq.(A.5):

$$\nabla^2 \lambda = \nabla \cdot (\rho \overline{\mathbf{v}}^{\star}) - \nabla \cdot (\rho \overline{\mathbf{v}}) \tag{A.7}$$

and use mass conservation for "divergence constraint" (dilation):

$$\nabla^2 \lambda = \nabla \cdot (\rho \overline{\mathbf{v}}^{\star}) + \partial_t \rho \tag{A.8}$$

Next, we can use equation of state for variable change:

$$\partial_t \rho = \partial_t P \left.\frac{\partial \rho}{\partial P}\right|_u + \partial_t u \left.\frac{\partial \rho}{\partial u}\right|_P \tag{A.9}$$

Thus,

$$\nabla^2 \lambda = \nabla \cdot (\rho \overline{\mathbf{v}}^{\star}) + \left.\frac{\partial \rho}{\partial P}\right|_u \partial_t P + \left.\frac{\partial \rho}{\partial u}\right|_P \partial_t u \tag{A.10}$$

This equation ensures mass conservation. As our next step, apply divergence to eq.(A.6), leading to

$$\nabla \cdot (u \nabla \lambda) = \nabla \cdot (\rho u \overline{\mathbf{v}}^{\star}) - \underbrace{\nabla \cdot (\rho u \overline{\mathbf{v}})}_{-\partial_t \rho u - P \nabla \cdot \overline{\mathbf{V}} + S_u} \tag{A.11}$$

and

$$\nabla \cdot (u \nabla \lambda) = \nabla \cdot (\rho u \overline{\mathbf{v}}^{\star}) + \left(\rho + u \left.\tfrac{\partial \rho}{\partial u}\right|_P\right) \partial_t u + u \left.\tfrac{\partial \rho}{\partial P}\right|_u \partial_t P + \\ + P \nabla \cdot \overline{\mathbf{v}} - S_u \tag{A.12}$$

[1] To make this distinct from "divergence-free" or solenoidal.

From eq.(A.5)

$$\overline{\mathbf{v}} = \overline{\mathbf{v}}^{\star} - \frac{1}{\rho}\nabla\lambda \tag{A.13}$$

which leads to

$$\nabla\cdot(u\nabla\lambda) + P\nabla\cdot\left(\tfrac{1}{\rho}\nabla\lambda\right) = \nabla\cdot(\rho u\overline{\mathbf{v}}^{\star}) + P\nabla\cdot\overline{\mathbf{v}}^{\star} + \left(\rho + u\left.\frac{\partial\rho}{\partial u}\right|_{P}\right)\partial_t u + \\ + u\left.\frac{\partial\rho}{\partial P}\right|_{u}\partial_t P - S_u \tag{A.14}$$

Next, use eq.(A.10) to eliminate $\partial_t u$:

$$\partial_t u = \frac{1}{\left.\frac{\partial\rho}{\partial u}\right|_{P}}\left[\nabla^2\lambda - \nabla\cdot(\rho\overline{\mathbf{v}}^{\star}) - \left.\frac{\partial\rho}{\partial P}\right|_{u}\partial_t P\right] \tag{A.15}$$

and

$$\nabla\cdot(u\nabla\lambda) + P\nabla\cdot\left(\tfrac{1}{\rho}\nabla\lambda\right) = \nabla\cdot(\rho u\overline{\mathbf{v}}^{\star}) + P\nabla\cdot\overline{\mathbf{v}}^{\star} + \\ + \underbrace{\frac{\rho + u\left.\frac{\partial\rho}{\partial u}\right|_{P}}{\left.\frac{\partial\rho}{\partial u}\right|_{P}}}_{\hat{u}}\left[\nabla^2\lambda - \nabla\cdot(\rho\overline{\mathbf{v}}^{\star}) - \left.\frac{\partial\rho}{\partial P}\right|_{u}\partial_t P\right] + u\left.\frac{\partial\rho}{\partial P}\right|_{u}\partial_t P - S_u \tag{A.16}$$

or

$$\boxed{\left.\frac{\partial\rho}{\partial P}\right|_{u}\left(\frac{u}{\hat{u}} - 1\right)\partial_t P - \frac{1}{\hat{u}}\left(\nabla\cdot(u\nabla\lambda) + P\nabla\cdot\left(\frac{1}{\rho}\nabla\lambda\right)\right)} + \nabla^2\lambda = \\ = \nabla\cdot(\rho\overline{\mathbf{v}}^{\star}) - \boxed{\frac{1}{\hat{u}}\left(\nabla\cdot(\rho u\overline{\mathbf{v}}^{\star}) + P\nabla\cdot\overline{\mathbf{v}}^{\star} - S_u\right)} \tag{A.17}$$

where the terms in boxes represent compressibility effects. Using the difinition of sound speed

$$c = \sqrt{\frac{1}{\left.\frac{\partial\rho}{\partial P}\right|_{u}}\left(1 - \frac{P}{\rho^2}\left.\frac{\partial\rho}{\partial u}\right|_{P}\right)} \tag{A.18}$$

and

$$\hat{u} = u + \frac{\rho}{\left.\frac{\partial\rho}{\partial u}\right|_{P}} \tag{A.19}$$

we can write the following *Pressure-Helmholtz* equation (PHE) for pressure/Lagrange multiplier:

$$\boxed{\frac{1}{\chi c^2}\partial_t P - \frac{1}{\hat{u}}\left(\nabla\cdot(u\nabla\lambda) + P\nabla\cdot\left(\frac{1}{\rho}\nabla\lambda\right)\right)} + \nabla^2\lambda = \\ = \nabla\cdot(\rho\overline{\mathbf{v}}^\star) - \boxed{\frac{1}{\hat{u}}\left(\nabla\cdot(\rho u\overline{\mathbf{v}}^\star) + P\nabla\cdot\overline{\mathbf{v}}^\star - S_u\right)} \tag{A.20}$$

where

$$\chi = \frac{1 + \frac{u}{\rho}\left.\frac{\partial\rho}{\partial u}\right|_P}{\frac{P}{\rho^2}\left.\frac{\partial\rho}{\partial u}\right|_P - 1} \tag{A.21}$$

Importantly, this equation incorporates both ***mass and energy conservation*** during the *"skewed projection"* procedure applied to the momentum vector field. The major challenging issues are related to the details of discretization of the "compressibility" terms (in boxes) of the PHE equation (A.20).

Appendix B

Cartesian Vector Calculus

GIVEN two Cartesian vectors $\mathbf{a} = \left\{a_x, a_y, a_z\right\}^{\mathrm{T}}$ and $\mathbf{b} = \left\{b_x, b_y, b_z\right\}^{\mathrm{T}}$, the *dot product* is defined as

$$\mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y + a_z b_z = a_k b_k \tag{B.1}$$

The *dyadic product* is denoted as [Ari]

$$\mathbf{ab} = \mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_x b_x & a_x b_y & a_x b_z \\ a_y b_x & a_y b_y & a_y b_z \\ a_z b_x & a_z b_y & a_z b_z \end{bmatrix} = a_k b_l \tag{B.2}$$

Spatial derivatives are denoted as

$$\nabla = \left\{\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right\}^{\mathrm{T}} = \left\{\partial_x, \partial_y, \partial_z\right\}^{\mathrm{T}} = \partial_j \tag{B.3}$$

Thus, the *gradient of* an arbitrary *scalar* φ is defined as

$$\nabla\varphi = \left\{\frac{\partial\varphi}{\partial x}, \frac{\partial\varphi}{\partial y}, \frac{\partial\varphi}{\partial z}\right\}^{\mathrm{T}} = \left\{\partial_x\varphi, \partial_y\varphi, \partial_z\varphi\right\}^{\mathrm{T}} = \partial_j\varphi \tag{B.4}$$

and, accordingly, dot product of a vector and a gradient of a scalar is

$$\mathbf{a} \cdot \nabla\varphi = a_x \frac{\partial\varphi}{\partial x} + a_y \frac{\partial\varphi}{\partial y} + a_z \frac{\partial\varphi}{\partial z} \tag{B.5}$$

Laplacian of an arbitrary *scalar* is defined as

$$\nabla \cdot \nabla\varphi = \nabla^2\varphi = \Delta\varphi = \frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} + \frac{\partial^2\varphi}{\partial z^2} = \partial_k^2\varphi \tag{B.6}$$

Divergence of a vector is defined as

$$\nabla \cdot \mathbf{a} = \frac{\partial a_x}{\partial x} + \frac{\partial a_y}{\partial y} + \frac{\partial a_z}{\partial z} \tag{B.7}$$

Gradient of an arbitrary *vector* is a tensor, defined as

$$\nabla \mathbf{a} = \begin{bmatrix} \frac{\partial a_x}{\partial x} & \frac{\partial a_x}{\partial y} & \frac{\partial a_x}{\partial z} \\ \frac{\partial a_y}{\partial x} & \frac{\partial a_y}{\partial y} & \frac{\partial a_y}{\partial z} \\ \frac{\partial a_z}{\partial x} & \frac{\partial a_z}{\partial y} & \frac{\partial a_z}{\partial z} \end{bmatrix} \tag{B.8}$$

and its *transpose*:

$$\nabla \mathbf{a}^{\mathrm{T}} = \begin{bmatrix} \frac{\partial a_x}{\partial x} & \frac{\partial a_y}{\partial x} & \frac{\partial a_y}{\partial x} \\ \frac{\partial a_x}{\partial y} & \frac{\partial a_y}{\partial y} & \frac{\partial a_y}{\partial y} \\ \frac{\partial a_x}{\partial z} & \frac{\partial a_y}{\partial z} & \frac{\partial a_y}{\partial z} \end{bmatrix} \tag{B.9}$$

Scalar product of a vector and divergence of a vector is a vector:

$$\mathbf{a} \cdot \nabla \mathbf{b} = \begin{Bmatrix} a_x \cdot \nabla b_x \\ a_y \cdot \nabla b_y \\ a_z \cdot \nabla b_z \end{Bmatrix} = \begin{Bmatrix} a_x \frac{\partial b_x}{\partial x} + a_y \frac{\partial b_x}{\partial y} + a_z \frac{\partial b_x}{\partial z} \\ a_x \frac{\partial b_y}{\partial x} + a_y \frac{\partial b_y}{\partial y} + a_z \frac{\partial b_y}{\partial z} \\ a_x \frac{\partial b_z}{\partial x} + a_y \frac{\partial b_z}{\partial y} + a_z \frac{\partial b_z}{\partial z} \end{Bmatrix} \tag{B.10}$$

Divergence of a dyadic product of two vectors is defined as

$$\nabla \cdot (\mathbf{ab}) = \nabla \cdot (\mathbf{a} \otimes \mathbf{b}) = \begin{Bmatrix} \frac{\partial}{\partial x}(a_x b_x) + \frac{\partial}{\partial y}(a_x b_y) + \frac{\partial}{\partial z}(a_x b_z) \\ \frac{\partial}{\partial x}(a_y b_x) + \frac{\partial}{\partial y}(a_y b_y) + \frac{\partial}{\partial z}(a_y b_z) \\ \frac{\partial}{\partial x}(a_z b_x) + \frac{\partial}{\partial y}(a_z b_y) + \frac{\partial}{\partial z}(a_z b_z) \end{Bmatrix} \tag{B.11}$$

Bibliography

[ABC00] Ann S. Almgren, John B. Bell, and William Y. Crutchfield. Approximate projection methods: Part i. inviscid analysis. *SIAM Journal on Scientific Computing*, 22(4):1139–1159, 2000.

[ABCH93] Ann S. Almgren, John B. Bell, Phillip Colella, and Louis H Howell. An adaptive projection method for the incompressible euler equations. In *Eleventh AIAA Computational Fluid Dynamics Conference*, pages 530–539. AIAA, 1993.

[ABS96] Ann S. Almgren, John B. Bell, and William G. Szymcyzk. A numerical method for the incompressible navier-stokes equations based on an approximate projection. *SIAM Journal for Scientific Computing*, 17(2):358–369, March 1996.

[Ari] R. Aris. *Vectors, Tensors, and the Basic Equations of Fluid Mechanics*. Dover Publications, Inc., New York.

[BBE+04] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[BCG89] John B. Bell, Philip Colella, and Harland M. Glaz. A second-order projection method for the incompressible navier-stokes equations. *Journal of Computational Physics*, 85:257–283, 1989.

[BCM01] David L. Brown, R. Cortez, and Michael L. Minion. Accurate projection methods for the incompressible navier-stokes equations. *Journal of Computational Physics*, 168:464–499, 2001.

[BM95] David L. Brown and Michael L. Minion. Performance of under-resolved two-dimensional incompressible flow simulations. *Journal of Computational Physics*, 122:165–183, 1995.

[Cho68] Alexandre Joel Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computations*, 22:745–762, 1968.

[Cho69] Alexandre Joel Chorin. On the convergence of discrete approximations to the Navier-Stokes equations. *Mathematics of Computations*, 57:341–353, 1969.

[Chr11] M. A. Christon. Hydra-th theory manual. Technical Report LA-UR-11-05387, Los Alamos National Laboratory, Los Alamos, New Mexico, September 2011.

[de 83] G. de Vahl Davis. Natural convection of air in a square cavity: a bench mark numerical solution. *International Journal for Numerical Methods in Fluids*, 3:249–264, 1983.

[dJ83] G. de Vahl Davis and I. P. Jones. Natural convection in a square cavity: a comparison exercise. *International Journal for Numerical Methods in Fluids*, 3:227–248, 1983.

[GC90] Philip M. Gresho and Stevens T. Chan. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 2: Implementation. *International Journal for Numerical Methods in Fluids*, 11:621–659, 1990.

[GC96] Philip M. Gresho and Stevens T. Chan. Projection 2 goes turbulent – and fully implicit. *preprint International Journal for Computational Fluid Dynamics*, March 1996. (LLNL UCRL-JC-123727).

[GCCH95] Philip M. Gresho, Stevens T. Chan, Mark A. Christon, and Allen C. Hindmarsh. A little more on stabilized $q1q1$ for transient viscous incompressible flow. *International Journal for Numerical Methods in Fluids*, 21:837–856, 1995.

[GQ97] J.-L. Guermond and L. Quartapelle. Calculation of incompressible viscous flow by an unconditionally stable projection fem. *Journal of Computational Physics*, 132:12–23, 1997.

[GQ98a] Jean-Luc Guermond and L. Quartapelle. On stability and convergence of projection methods based on pressure poisson equation. *International Journal for Numerical Methods in Fluids*, 26:1039–1053, 1998.

[GQ98b] Jean-Luc Guermond and L. Quartapelle. On the approximation of the unsteady navier-stokes equations by finite element projection methods. *Numerische Mathematik*, 80:207–238, 1998.

[Gre90] Philip M. Gresho. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 1: Theory. *International Journal for Numerical Methods in Fluids*, 11:587–620, 1990.

[Gue96] Jean-Luc Guermond. Some implementations of projection methods for navier-stokes equations. *Mathematical Modelling and Numerical Analysis*, 30(5):637–667, 1996.

[Gue97] Jean-Luc Guermond. A convergence result for the approximation of the navier-stokes equations by an incremental projection method. *C. R. Acad. Sci. Paris*, 325:1329–1332, 1997.

[HA68] F. H. Harlow and A. A. Amsden. Numerical calculation of almost incompressible flow. *Journal of Computational Physics*, 3:80–93, 1968.

[HA71] F. H. Harlow and A. A. Amsden. A numerical fluid dynamics calculation method for all flow speeds. *Journal of Computational Physics*, 8:197–213, 1971.

[HA75a] F. H. Harlow and A. A. Amsden. Flow of interpenetrating material phases. *Journal of Computational Physics*, 18:440–464, 1975.

[HA75b] F. H. Harlow and A. A. Amsden. Numerical calculation of multiphase fluid flow. *Journal of Computational Physics*, 17:19–52, 1975.

[HN81] C. Hirt and B. Nichols. Volume of fluid (VOF) method for the dynamics of free surfaces. *Journal of Computational Physics*, 39:201, 1981.

[Iss85] R. I. Issa. Solution of the Implicitely Discretized Fluid Flow Equations by Operator-Splitting. *Journal of Computational Physics*, 62:40–65, 1985.

[Kan86] J. Van Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM Journal for Scientific and Statistical Computing*, 7:870–891, 1986.

[KCMM03] D.A. Knoll, L. Chacon, L.G. Margolin, and V.A. Mousseau. On balanced approximations for time integration of multiple time scales systems. *Journal of Computational Physics*, 185:583–611, 2003.

[KK04] D. A. Knoll and D. Keyes. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. *Journal of Computational Physics*, 193:357–397, 2004.

[KMCR05] D.A. Knoll, V.A. Mousseau, L. Chacon, and J. M. Reisner. Jacobian-free Newton-Krylov methods for the accurate time integration of stiff wave systems. *SIAM Journal of Scientific Computing*, 25:213–230, 2005.

[KMK96] D.A. Knoll, P. R. McHugh, and D. E. Keyes. Newton-Krylov methods for low-Mach-number compressible combussion. *AIAA Journal*, 34(5):961, 1996.

[KNW99] Omar M. Knio, Habib N. Najm, and Peter S. Wyckoff. A semi-implicit numerical scheme for reacting flow. ii. stiff operator-split formulation. *pre-print submitted to Journal of Computational Physics*, 1999.

[KR00] D. A. Knoll and W. J. Rider. A multigrid preconditioned Newton-Krylov method. *SIAM Journal of Scientific Computing*, 21:691–710, 2000.

[MB97] Michael L. Minion and David L. Brown. Performance of under-resoloved two-dimensional incompressible flow simulations, ii. *Journal of Computational Physics*, 138:734–765, 1997.

[Min96] Michael L. Minion. A projection method for locally refined grids. *Journal of Computational Physics*, 127:158–178, 1996.

[NC12] R.R. Nourgaliev and M. A. Christon. Solution algorithms for multi-fluid-flow averaged equations. Technical Report INL/EXT-12-27187, Idaho National Laboratory, Idaho Falls, Idaho, September 2012.

[OI01] P. J. Oliveira and R. I. Issa. An improved PISO algorithm for the computation of buoyancy-driven flows. *Numerical Heat Transfer, Part B*, 40:473–493, 2001.

[ope13] OpenFoam Web page: The open source CFD toolbox, 2013. http://www.openfoam.com.

[PAB+97] Elbridge G. Puckett, Ann S. Almgren, John B. Bell, Daniel L. Marcus, and William J. Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *Journal of Computational Physics*, 130:269–282, 1997.

[Pat80] S. Patankar. *Numerical Heat Transfer and Fluid Flow*. Taylor & Francis, 1980.

[Per93] J. Blair Perot. An analyis of the fractional step method. *Journal of Computational Physics*, 108:51–58, 1993.

[PS72] S.V. Patankar and D.B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat Mass Transfer*, 15:1787–1806, 1972.

[RC82] C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an isolated airfoil with trailing edge separation. *AIAA Journal*, 21(11):1525–32, 1982.

[Rid94a] William J. Rider. Filtering nonsolenoidal modes in numerical solutions of incompressible flows. Technical Report LA-UR-3014, Los Alamos National Laboratory, Los Alamos, New Mexico, September 1994.

[Rid94b] William J. Rider. The robust formulation of approximate projection methods for incompressible flows. Technical Report LA-UR-3015, Los Alamos National Laboratory, 1994.

[Rid95] William J. Rider. Approximate projection methods for incompressible flow: implementation, variants and robustness. Technical Report LA-UR-2000, Los Alamos National Laboratory, Los Alamos, New Mexico, July 1995.

[RKM+95] W. J. Rider, D. B. Kothe, S. J. Mosso, J. H. Cerutti, and J. I. Hochstein. Accurate solution algorithms for incompressible multiphase flows. Technical Report AIAA-95-0699, AIAA, Reno, Nevada, January 1995.

[SA92] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. In *AIAA-92-0439*, Reno, Nevada, January 1992. AIAA 30th Aerospace Science Meeting and Exhibit.

[SAB+99] M. Sussman, A.S. Almgren, J.B. Bell, P. Colella, and L.H. Howell. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics*, 148:81–124, 1999.

[Set99] J.A. Sethian. *Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and material science*. Cambridge University Press, 1999.

[Spa83] D.B.. Spalding. Development in the IPSA procedure for numerical computation of multiphase-flow phenomena with interfacial slip, unequal temperatures. In T.M. Shih, editor, *Numerical Methodologies in Heat Transfer, Proc. Second National Symposium*, pages 421–436. Hemisphere, 1983.

[SS86] Y. Saad and M.H. Schultz. GMRES: a Generalized Minimal Residual algorithm for solving linear systems. *SIAM Journal of Science and Statistical Computing*, 7:856, 1986.

[Wet98] Brian B. Wetton. Error analysis of pressure increment schemes. *submitted to SINUM*, April 1998.

This Page is Intentionally Left Blank

This Page is Intentionally Left Blank