INL/CON-08-13676
PREPRINT

Direct Numerical Simulation of Interfacial Flows: Implicit Sharp-Interface Method (I-SIM)

American Institute of Aeronautics and Astronautics

Robert Nourgaliev
Theo Theofanous
HyeongKae Park
Vincent Mousseau
Dana Knoll

January 2008

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

Direct Numerical Simulation of Interfacial Flows:
Implicit Sharp-Interface Method (I-SIM)

Robert Nourgaliev[*]
Idaho National Laboratory, Idaho Falls, ID, 83415-3840

Theo Theofanous[†]
Center for Risk Studies and Safety, University of California Santa Barbara, Goleta, CA, 93117

HyeongKae Park[*], Vincent Mousseau[*] and Dana Knoll[‡]
Idaho National Laboratory, Idaho Falls, ID, 83415-3840

In recent work (Nourgaliev, Liou, Theofanous, JCP in press) we demonstrated that numerical simulations of interfacial flows in the presence of strong shear must be cast in dynamically sharp terms (sharp interface treatment or SIM), and that moreover they must meet stringent resolution requirements (i.e., resolving the critical layer). The present work is an outgrowth of that work aiming to overcome consequent limitations on the temporal treatment, which become still more severe in the presence of phase change. The key is to avoid operator splitting between interface motion, fluid convection, viscous/heat diffusion and reactions; instead treating all these non-linear operators fully-coupled within a Newton iteration scheme. To this end, the SIM's cut-cell meshing is combined with the high-order-accurate implicit Runge-Kutta and the "recovery" Discontinuous Galerkin methods along with a Jacobian-free, Krylov subspace iteration algorithm and its physics-based preconditioning. In particular, the interfacial geometry (i.e., marker's positions and volumes of cut cells) is a part of the Newton-Krylov solution vector, so that the interface dynamics and fluid motions are fully-(non-linearly)-coupled. We show that our method is: (a) robust (*L*-stable) and efficient, allowing to step over stability time steps at will while maintaining high-(up to the 5th)-order temporal accuracy; (b) fully conservative, even near multimaterial contacts, without any adverse consequences (pressure/velocity oscillations); and (c) high-order-accurate in spatial discretization (demonstrated here up to the 12th-order for smooth-in-the-bulk-fluid flows), capturing interfacial jumps sharply, within one cell. Performance is illustrated with a variety of test problems, including low-Mach-number "manufactured" solutions, shock dynamics/tracking with slow dynamic time scales, and multi-fluid, high-speed shock-tube problems. We briefly discuss preconditioning, and we introduce two physics-based preconditioners – "Block-Diagonal" and "Internal energy-Pressure-Velocity Partially Decoupled", demonstrating the ability to efficiently solve all-speed flows with strong effects from viscous dissipation and heat conduction.

[*] Advanced Nuclear Energy Systems Department, Multiphysics Methods Group, ID 83415-3840.
[†] Professor, Chemical Engineering, 6740 Cortona Drive, Goleta, CA 93117.
[‡] Advanced Nuclear Energy Systems Department, Multiphysics Methods Group, ID 83415-3855.

I. Introduction

Free (fluid-fluid) interfaces in the presence of body forces and/or differential velocities are subject to deformation and breakup – processes that are principally responsible for flow regimes development, and thus for the macroscopic features in all multi-fluid systems. Depending on the direction of the acceleration vector, body forces normal to an interface can be stabilizing or destabilizing, differential velocities parallel to the interface are always destabilizing, and for the development of instabilities such driving forces must be sufficient to overcome the force due to interfacial tension (always stabilizing). Under unstable conditions, early growth of an interfacial disturbance is exponential in time, and the theory for understanding this regime, based on the linearized Navier-Stokes equations, rests on firm grounds. At amplitudes that are a significant fraction of the wave-length, this theory breaks down, non-linear analysis becomes scarcely feasible, and numerical simulation is the key to further progress. The profound challenge of this endeavor is that in nature, interfacial instabilities develop essentially from "nothing", while in a numerical approach (intrinsically discrete) they must be "seeded" with "something" that in most practical cases is not known *a priori* (i.e., not available from linear theory).

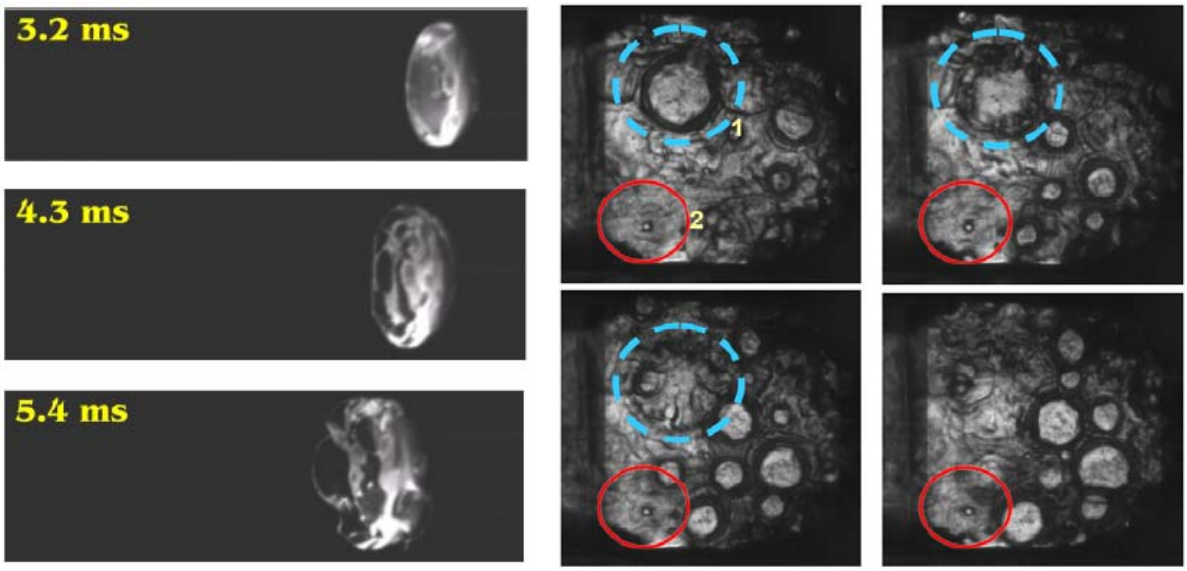

Figure 1. (Left): Sequences of experimental LIF (Laser-Induced Fluorescence) images during breakup of 3 mm TBP (Tributyl Phosphate) drops in a Mach 3 rarefied gas flow[38] (We~30). The flow is from right to left. Shown are sample deformation and penetration histories leading to breakup. (Right): Optical image of an evaporating liquid film on a heater at heat flux near burnout (1 MW/m^2). The dark rings show the dynamics of bubble-edges as they grow and retract following bursting – one such full cycle is marked by the dotted circle. Burnout occurs when the center of such "craters" dries out forming a dry spot, an instability driven by vapor recoil at the triple contact line and capillarity[13].

In a recent paper[20] we addressed ourselves to a preliminary step in this quest – that of simulating interfacial instabilities at their inception and early growth. For this linear regime, as noted above, there is a body of analytical work that provides a solid foundation to measure success, and we made use of this foundation to demonstrate an intrinsic necessity *in shear flows* that the numerical scheme observe rigorously the sharp dynamics (the stress-jump conditions) at the interface [see also references[39,40,41]]. In the present paper we present the further developments needed for high-fidelity, robust simulations well into the non-linear regime, a numerical endeavor that, even if well-founded (in the linear regime), is fraught with pitfalls. Not unexpectedly, the principal thrust in these further developments is *implicitness,* and the central role to this end is the *Jacobian-free Newton-Krylov (JFNK)* approach[19]. Our method provides yet another practical realization within these two rapidly-developing topics, along with a suite of further developments (as detailed in Section II) made necessary by the particulars of the two classes of practical problems addressed.

The one class involves the combined action of Raylegh-Taylor, viscous Kelvin-Helmholtz, and Richtmyer-Meshkov instability mechanisms, as illustrated in Figure 1(left) – these are a (perhaps even *the* key) fundamental component in high-speed atmospheric dissemination problems[37]. The other class involves the stability of contact-line motion under steep temperature gradients and intense phase changes[24], along with capillary instability in shear flow, as illustrated in Figure 1(right) – these are the key physics controlling the coolability limits in high-heat-flux boiling (burnout)[12,13,36,42]. Clearly seen in both problems are the wide ranges of operative length and time scales, a challenge that could not be effectively met within the original SIM (explicit solver, large stencil of the high-order finite-volume numerics interacting inefficiently with the structured adaptive mesh refinement infrastructure employed). In this work we take necessary steps towards practical simulations by recasting SIM in a parallel-and-time-stepping-efficient environment, while enhancing the high-fidelity capturing (tracking) of interface motions as needed for the very large CFL-operability sought. The rational for the numerical strategies employed to these ends is provided in the next Section II. Details for the *Implicit Sharp Interface Method* (I-SIM) are given in Section III. Illustrative results for test problems designed with an emphasis on wide-ranging length and time scales (and flow speeds) can be found in Section IV. The final Section V is a summary of main conclusions, along with the outlook for further developments on physics-based preconditioning and extension to 2D/3D.

II. Basic Considerations

On top of the driving considerations for the original SIM[20,21,22,23]:

a. no smearing of properties across the interface is permitted;
b. jump conditions are implemented sharply at the interface (within one cell); and
c. discretizations are at least second-order-accurate in both time and space;

the additional requirements are:

1. *implicit interface tracking without operator-splitting of interface motion and fluid dynamics;*
2. *fully-coupled, all-speed fluid dynamics without operator-splitting of hyperbolic, viscous/heat diffusion and chemical reaction terms; and*
3. *efficiency of large-scale parallel computing in a structured adaptive mesh environment.*

The key feature in meeting requirements (*a-c*) was the combination of a structured (Carterian, or C_1-) grid with an unstructured (cut-cell, or C_2-) grid defining the interface (piecewise-linearly) and anchored on a level-set-, marker-tracking-based procedure. The *C_2-grid* is continuously adapting to the interface evolution, inside an interfacial corridor, while remaining consistently embedded into a (structured-) adaptive Cartesian *(C_1-)* grid in the bulk fluids.

The central element in meeting requirements (*1-3*) is the *Jacobian-Free Newton-Krylov (JFNK) methodology*[19]. The JFNK provides an integrating framework for a *fully-coupled, implicit high-order spatio-temporal discretization* of the non-linear interface dynamics, without operator-splitting of interface motion, convection, viscous diffusion and heat conduction. It allows us to efficiently combine *Discontinuous Galerkin* (DG) type spatial discretization[8] for both the C_1- and C_2-grids with *Implicit Runge-Kutta* (ESDIRK) temporal discretization[5].

The Discontinuous Galerkin discretization provides a better (more efficient/easier/elegant) framework for dealing with high-order treatment of the C_2-(unstructured) grid, than the finite-volume discretization used in the original SIM[20]. Moreover, there are compelling reasons to believe that the DG discretization is well-suited (more efficient) for Structured AMR (mainly, due to the compactness of its stencil). Here, we introduce a "recovery family" of DG methods (rDG) to provide, within a very compact stencil, very accurate (demonstrated up to 12^{th}-order here) spatial discretization of both hyperbolic and diffusion operators on unstructured (C_2-) grids, with a limited number degrees of freedom per cell. This is based on early work of van Leer for high-order finite-volume discretization of conservation laws[43] and his recent "recovery method" for the diffusion operator[44,45]. Since the rDG is combined here with the implicit *L*-stable high-order Runge-Kutta time discretization[5], the stability and robustness of the method is not of concern, in difference to all previous (explicit RK) DG implementations[8,9,10,11,44,45].

Importantly, during a time step, the shape/volume of the *cut-cells* are allowed to vary – this is achieved by including the positions of interfacial markers to be a part of the Newton-Krylov (NK) solution vector. This is a key to removing operator-splitting, temporal errors and to providing robustness of the algorithm when stepping over fast normal-mode time scales, while maintaining full conservation of the algorithm. Even though the algorithm is fully-implicit and *L*-stable for any time step, we use an interface-CFL condition ($\frac{\mathrm{U_m}\Delta t}{\Delta x} < \frac{1}{2}$, where $\mathrm{U_m}$ is the interface speed) to prevent singularities in the C_2-grid, and to maintain the solution within the ball-of-convergence for Newton iterations.

For a linear solver in JNFK, we deploy the Jacobian-free version of the "*Generalized Minimal RESidual Method*" (*GMRES*)[32,33] – a particular implementation of the projection method based on iterations in Krylov subspace[33] – adapted to our cut-cell/rDG environment. The needed efficiency of JFNK is gained by *physics-based pre-conditioning* (PBP)[19]. To this end, we introduced two PBPs specially adapted to the rDG spatial discretization: the first one may be named *"Block-Diagonal" (BD)*; and the second one we refer to as *"Internal energy-Pressure-Velocity Partially Decoupled" (IPV-PD)*.

Test results in 1D show that our method effectively addresses the combined challenge of accuracy and speed for computations over a wide range of flow speeds, material properties and in the presence of discontinuities (shocks, contacts). In particular we demonstrate:

1. Stable time-stepping at stability CFL numbers of up to 500 (a speedup by four orders relative to SIM);
2. Asymptotic convergence of up to 12th-order in space and up to 5th-order in time, at non-linear tolerances down to 10^{-16} (implemented in quad precision);
3. Fully conservative interface tracking under shock dynamics without pressure-velocity oscillations at multi-material contacts;
4. A use of a new method (eigenscopy) to analyze efficiency of preconditioning.

III. Implicit Sharp-Interface Method (I-SIM)

Each stage of RK loop is a non-linear solve, and it is driven by a Newton iteration scheme in the sequence shown below. Each Newton iteration requires a linear solve which is done with a Jacobian-free version of *GMRES*.

- Beginning of time step, $t^{(\mathrm{n})}$:
  - Create/initialize C_2-grid and solution vector $\mathbf{X}^{(\mathrm{n})}$.
  - Start Runge-Kutta loop, $\mathrm{rk} = 1, ..., s$:
    - Start Newton's iterations, $\mathrm{m} = 1, ...$:
      - Start linear solve (GMRES):
        - Form Krylov vectors, a basis of the Krylov subspace
          $$\mathbb{K} = \mathrm{span}\left(\mathbf{r}_0, \hat{\mathbb{J}}\mathbf{r}_0, \hat{\mathbb{J}}^2\mathbf{r}_0, \hat{\mathbb{J}}^3\mathbf{r}_0, ...\right)$$
          until converged (a sufficient number of basis vectors is accumulated). $\hat{\mathbb{J}}$ and $\mathbf{r}_0$ are the right-preconditioned Jacobian matrix and initial residual vector, respectively.
      - End of linear solve.
      - Update solution vector $\mathbf{X}^{(\mathrm{m})} = \mathbf{X}^{(\mathrm{m}-1)} + \delta\mathbf{X}^{(\mathrm{m}-1)}$
      - Check for convergence of Newton's iterations.
    - End of Newton's iterations.
  - End of Runge-Kutta loop.
- End of time step.
- Choose new time step Δt.
- Move to new time step, $t^{(\mathrm{n}+1)} = t^{(\mathrm{n})} + \Delta t$.
- ...

Details of the solution procedure are given below. We start by defining the governing equations in the bulk of each fluid, Section III.A. Spatial discretization with Discontinuous Galerkin method and its higher-order "recovery" extension are presented in Section III.B. Time discretization is explained in Section III.C. Finally, we outline the details of JFNK in Section III.D and its physics-based preconditioning in Section III.E.

A. Governing Equations

The governing equations for fluid dynamics in the bulk of each fluid, including molecular diffusion terms and local sources, are:

$$\mathbb{U}_t + \mathbb{F}_x = \mathbb{S} \tag{1}$$

$$\mathbb{U} = \begin{bmatrix} \rho \\ m \\ E \end{bmatrix}; \quad \mathbb{F} = \begin{bmatrix} m \\ m^2/\rho + P - \tau \\ u\left(E + P - \tau\right) + q \end{bmatrix} \quad \text{and} \quad \mathbb{S} = \begin{bmatrix} \mathbb{S}_\rho \\ \mathbb{S}_m \\ \mathbb{S}_E \end{bmatrix} \tag{2}$$

where ρ, P, u, $m = \rho u$, $E \equiv \rho\left(i + \frac{u^2}{2}\right)$, i, $\tau \equiv \eta \partial_x u$, η, $q \equiv -\beta \partial_x i$, $\beta \equiv \frac{\kappa}{C_v}$ are density, pressure, velocity, momentum, total energy, internal energy, viscous stress, viscosity, heat flux, and ratio of thermal conductivity κ to specific heat C_v, respectively. The diffusion and source terms introduce additional time scales to that of the fluid motion. A key aim of the method is to achieve optimal time integration over a wide range of flow and fluid-property parameters. These result in a broad spectrum of normal modes and dynamic time scales, which causes the numerical system to be stiff. We use a fully-compressible formulation, with a stiffened-gas equation of state:

$$P = \rho i \left(\gamma - 1\right) - \gamma \Pi \tag{3}$$

where γ and Π are parameters specific to material.

B. "Recovery" Discontinuous Galerkin (rDG) Family

The Discontinuous Galerkin (DG) method can be viewed as a high-order extension of the Finite-Volume (FV) discretization approach. Originally introduced by Reed & Hills[31] in 1973 for neutron transport equation (steady-state linear hyperbolic problems), DG has been recently extended to transient non-linear hyperbolic (conservation-law) problems[8,9,10,11] and parabolic (diffusion) problems[4,28,44,45]. As discussed by Cockburn[8], there are several important advantages to be realized by DG schemes:

a) they are more convenient than FV in complicated geometries;
b) they can easily accommodate different mesh adaptivity strategies – refining/de-refining without considering the continuity restrictions typical of conforming finite element methods, and by simply varying the degree of in-element approximating polynomials;
c) the method uses compact stencils, which result in more efficient mesh adaptation, parallelization, and the implementation of boundary conditions.

A DG method of p^{th}-order involves (p+1) degrees of freedom (DoFs) per cell, $\mathbb{U}_j^{(n=0,..,p)}$. The 0^{th}-order DoF corresponds to a cell-average variable (finite-volume representation). All higher-order DoFs can be interpreted as "perturbations" or higher-order corrections. Cell-level solutions are represented in terms of the cell's DoFs by:

$$\mathbb{U}_{h_j}(x) = \sum_{n=0}^{p} \mathbb{U}_j^{(n)} \mathfrak{L}_{(n)}(\xi) \tag{4}$$

where $\mathfrak{L}_{(n)}(\xi)$ is the scaled $\left(\xi = \frac{2(x-x_j)}{\Delta x_j}\right)$ Legendre polynomial of order *n*. In general, these solutions are discontinuous from cell to cell.

"Recovery" DG (rDG) is a higher-order extension of the finite-volume piecewise-parabolic method[43] (PPM), as the in-cell solution is "recovered" from the available degrees of freedom of the cell and its immediate (von Neumann) neighbors:

$$\tilde{\mathbb{U}}_{h_j}(x) = \sum_{n=0}^{R} \tilde{\mathbb{U}}_j^{(n)} \mathfrak{L}_{(n)}(\xi) \tag{5}$$

where $\tilde{\mathbb{U}}_j^{(n)}$ is the n^{th} "recovered" degree of freedom. It can be shown that the recovered in-cell distribution $\tilde{\mathbb{U}}_{h_j}(x)$ is $(R+1)^{\text{th}}$-order-accurate, where $R = 3p+2$. The recovery DoFs are computed using the following "weak statement":

$$\int_{\mathcal{I}_m} \mathfrak{L}_{(n)}(\xi)\, \tilde{\mathbb{U}}_{h_m}(x)\, dx = \int_{\mathcal{I}_m} \mathfrak{L}_{(n)}(\xi)\, \mathbb{U}_{h_m}(x)\, dx \qquad \begin{matrix} m=j,j\pm1 \\ \text{and} \\ n=0,\ldots,p \end{matrix} \tag{6}$$

and

$$\tilde{\mathbb{U}}_j^{(n)} = \frac{1+2n}{\Delta x_j} \int_{\mathcal{I}_j} \mathfrak{L}_{(n)}(\xi)\, \tilde{\mathbb{U}}_{h_m}(x)\, dx \tag{7}$$

The first *N* DoFs of the unlimited rDG coincide with DG's DoFs,

$$\tilde{\mathbb{U}}_j^{(n)} = \mathbb{U}_j^{(n)}, \quad n = 0, \ldots, p \tag{8}$$

The rest of the DoFs are given in Appendix A for $\text{rDG}_{0,1,2,3}$.

Notably: *a)* rDG_0 is exactly the 3rd-order-accurate finite-volume PPM; *b)* the stencil of the rDG_n is compact, i.e. involves only immediate von Neumann neighbors for in-cell approximating polynomials of any degree, and *c)* the "recovery" operation offers a way to estimate spatial discretization errors, which are usable in AMR-tagging for refinement/de-refinement of meshes.

The evolution equations for each degree of freedom can be written as

$$\frac{d}{dt}\tilde{\mathbb{U}}_j^{(n)} = \mathfrak{F}_j^{(n)} \tag{9}$$

where

$$\mathfrak{F}_j^{(n)} = -\frac{1}{C_n}\left[\mathbb{H}_{j+\frac{1}{2}} \mathfrak{L}_{(n)}(1) - \mathbb{H}_{j-\frac{1}{2}} \mathfrak{L}_{(n)}(-1) - \int_{\mathcal{I}_j} \Psi^{(n)}(x)\, dx\right] \tag{10}$$

is the spatial discretization operator accounting for hyperbolic and source terms*. $\mathbb{H}_{j\pm\frac{1}{2}}$ are numerical fluxes at cell edges, computed with either the LLF[35] or the AUSM$^+$-up[25] schemes. The variable $\Psi^{(n)}(x)$ is defined as

* Diffusion operator is discretized along the lines of van Leer's "recovery" method[44,45] as described in Appendix B.

$$\Psi^{(n)}(x) \equiv \mathbb{F}(x)\,\partial_x \mathfrak{L}_{(n)}(x) + \mathbb{S}(x)\,\mathfrak{L}_{(n)}(x) \tag{11}$$

Integration over cell $\mathcal{I}_j$ is done using a 12-point Gaussian quadrature formula. The parameter C_n is a normalization constant for the Legendre polynomials,

$$C_n = \frac{\Delta x_j}{2n+1} \tag{12}$$

A derivation of the weak form, eq.(9), representation of the governing equation, (1), is given by Cockburn[8].

Hereafter, spatial discretization schemes are denoted as $\mathrm{rDG}^{\mathrm{C}}_{\mathrm{A(B)}}$, where A is the order of the DG, B is either the order of the "recovered" in-cell polynomial (if unlimited) or the name of the used limiter (e.g., van-Albada, vAl, in shock dynamics tests), and C is the flux scheme (LLF or AUSM).

C. Implicit Runge-Kutta (ESDIRK) method

The vector $\mathfrak{F}_j^{(n)}$ corresponds to rDG spatial discretization of convection, diffusion and reaction (source) terms. The Jacobian of $\mathfrak{F}_j^{(n)}$, $\frac{\partial \mathfrak{F}_j^{(n)}}{\partial \tilde{\mathbb{U}}_j^{(n)}}$, might have a large spread of eigenvalues, which give rise to stiffness, defined as[5] a configuration when the largest scaled eigenvalue of the Jacobian $\frac{\partial \mathfrak{F}_j^{(n)}}{\partial \tilde{\mathbb{U}}_j^{(n)}}$ located in the complex left-half-plane (LHP) is $||z = \lambda(\Delta t)|| \gg 1$. Stiffness might be associated with mesh (here – small cut cells) and wide spread of physical times (acoustic waves in low-Mach-number applications, high viscosity/thermal conductivity coefficients, phase change, chemical reactions, etc.). In an ideal time discretization scheme, the time step is selected solely based on error considerations, without concerns about stability and robustness. We are interested in schemes which not only do not amplify any LHP-scaled eigenvalues (*A-stability*), but also provide a *complete damping of all eigenvalues* including those at the limit $||z \to \infty||$ (*L-stability*).

A family of implicit Runge-Kutta schemes with *L*-stability has been recently developed by Carpenter and co-authors[5,6]. These "*explicit, singly diagonal implicit Runge-Kutta"* (ESDIRK) schemes can be written as

$$\begin{aligned}
\mathbb{U}^{[k]} &= \mathbb{U}^{[n]} + \Delta t \sum_{r=1}^{k} \mathsf{a}_{kr} \mathfrak{F}\left(\mathbb{U}^{[r]}\right), \quad k = 1, ..., s-1 \\
\mathbb{U}^{[n+1]} &= \mathbb{U}^{[n]} + \Delta t \sum_{r=1}^{s} \mathsf{b}_r \mathfrak{F}\left(\mathbb{U}^{[r]}\right) \\
\hat{\mathbb{U}}^{[n+1]} &= \mathbb{U}^{[n]} + \Delta t \sum_{r=1}^{s} \hat{\mathsf{b}}_r \mathfrak{F}\left(\mathbb{U}^{[r]}\right)
\end{aligned} \tag{13}$$

where *s* is the number of stages; and a_{kr}, b_r, $\hat{\mathsf{b}}_r$ are the stage, the main, and the embedded scheme weights, respectively. In eq.(13) we omitted all sub/superscripts associated with spatial discretization. The vectors $\mathbb{U}^{[n+1]}$ and $\hat{\mathbb{U}}^{[n+1]}$ are p^{th} and $(p\text{-}1)^{\text{th}}$ order solutions at time level n+1. The vector $\hat{\mathbb{U}}^{[n+1]}$ is a by-product (free) and can be used for temporal error estimation. The Butcher tableaus for the third, fourth and fifth-order ESDIRK are developed in[5] and take the following form:

$$
\begin{array}{c|cccccc}
0 & 0 & 0 & 0 & 0 & \ldots & 0 \\
\mathsf{c}_2 & \mathsf{a}_{21} & \gamma & 0 & 0 & \ldots & 0 \\
\mathsf{c}_3 & \mathsf{a}_{31} & \mathsf{a}_{32} & \gamma & 0 & \ldots & 0 \\
\ldots & & & \ldots & & & \\
\mathsf{c}_{s-1} & \mathsf{a}_{(s-1)1} & \mathsf{a}_{(s-1)2} & \ldots & \ldots & \gamma & 0 \\
1 & \mathsf{b}_1 & \mathsf{b}_2 & \mathsf{b}_3 & \ldots & \mathsf{b}_{(s-1)} & \gamma \\
\hline
 & \mathsf{b}_1 & \mathsf{b}_2 & \mathsf{b}_3 & \ldots & \mathsf{b}_{(s-1)} & \gamma \\
\hline
 & \hat{\mathsf{b}}_1 & \hat{\mathsf{b}}_2 & \hat{\mathsf{b}}_3 & \ldots & \hat{\mathsf{b}}_{(s-1)} & \hat{\mathsf{b}}_{(s)}
\end{array}
\tag{14}
$$

where c_r denote the point in time of the r^{th}-stage, $t^{[n]} + \mathsf{c}_r \Delta t$. Note that the first stage is explicit, and the diagonal elements for all stages $r > 1$ are the same, $\mathsf{a}_{rr} = \gamma$. The coefficients for ESDIRK$_{3,4,5}$ are given in Appendix C.

D. Jacobian-Free Newton-Krylov (JFNK) Method

Each RK stage is a non-linear solve for a system of the form

$$\mathbf{res}\left(\mathbf{X}\right) = 0 \tag{15}$$

where $\mathbf{X} = \left(x_1, x_2, ..., x_M, \mathbb{U}_1^{(k=0,...,p)^\mathsf{T}}, \mathbb{U}_2^{(k=0,...,p)^\mathsf{T}}, ..., \mathbb{U}_{\mathrm{N_{cells}}}^{(k=0,...,p)^\mathsf{T}}\right)^\mathsf{T}$ is a solution vector which includes positions of all (M) interfacial markers and all (p+1) degrees of freedom for all conservative variables (ρ, m, E) in all (N_{cells}) computational cells. The size of the solution vector therefore is $N = (M + 3(p+1)N_{\mathrm{cells}})$. The residual vector $\mathbf{res}$ for rDG's degrees of freedom takes the form:

$$\mathrm{res}_{\tilde{\mathbb{U}}_j^{(k)}} = \tilde{\mathbb{U}}_j^{(k)[rk]} - \tilde{\mathbb{U}}_j^{(k)[n]} - \Delta t \sum_{r=1}^{rk} \mathsf{a}_{rk,r} \mathfrak{F}_j^{(k)} \left(\mathbf{X}^{[r]}\right) \tag{16}$$

and for each marker the form:

$$\mathrm{res}_{x_m} = x_m^{[rk]} - x_m^{[n]} - \Delta t \sum_{r=1}^{rk} \mathsf{a}_{rk,r} \underbrace{u_m\left(\mathbf{X}^{[r]}\right)}_{\substack{\text{Marker's velocity,} \\ \text{from sharp cou-} \\ \text{pling at interface}}} \tag{17}$$

Newton's method solves a non-linear system, eq.(15), iteratively as a sequence of linear problems defined by

$$\mathbb{J}^i \delta\mathbf{X}^i = -\mathbf{res}\left(\mathbf{X}^i\right) \tag{18}$$

The matrix $\mathbb{J}$ is the Jacobian of the i^{th} Newton's iteration step and $\delta\mathbf{X}^i$ is the update vector. Each $_{(i,j)^{\text{th}}}$ element of the Jacobian matrix is a partial derivative of the i^{th} equation with respect to the j^{th} variable:

$$\mathbb{J}_{i,j} \equiv \frac{\partial \mathbf{res}_i}{\partial \mathbf{X}_j} \tag{19}$$

The linear system is solved for $\delta\mathbf{X}^i$ and the new Newton's iteration value for $\mathbf{X}$ is then computed as

$$\mathbf{X}^{i+1} = \mathbf{X}^i + \mathfrak{d}\ \delta\mathbf{X}^i \tag{20}$$

where $\mathfrak{d}$ is the damping parameter, $\mathfrak{d} \in [0,1]$. This parameter is used to keep the solution vector in physically realizable manifold and/or to help bringing the initial guess inside the Newton method's "ball of convergence". The damping parameter can be interpreted geometrically as a scaling factor, preserving the direction of the update vector, but shortening its length. In the present study, for some difficult problems/large time steps, we start by choosing $\mathfrak{d} \sim 0.5$ for the few first iterations (enough to bring the solution to the convergence range), and then switching to $\mathfrak{d} = 1$, so as to achieve quadratic convergence rate at the asymptotic iteration range.

Newton's iterations on **X** are continued until the convergence criterion

$$\left|\left|\mathbf{res}\left(\mathbf{X}^i\right)\right|\right|_2 < \mathrm{tol}_{\mathrm{N}} \left|\left|\mathbf{res}\left(\mathbf{X}^0\right)\right|\right|_2 \tag{21}$$

is satisfied. Nonlinear tolerance is set to $\mathrm{tol}_{\mathrm{N}} = 10^{-16}$. *We use quadruple precision for all arithmetic*, so as to accurately measure asymptotic convergence of our high-order spatial and temporal discretization schemes. This is also useful for enabling eigenscopy of the Jacobian and preconditioning matrices, as preventing "pollution" by spurious eigenvalues.

The linear solver used in the present study is the Arnoldi-based *Generalized Minimal RESidual method*[32] (GMRES). It belongs to the general class of Krylov subspace iteration methods. These projection (Galerkin) or generalized projection (Petrov-Galerkin) methods[33] are suitable for solving generally non-symmetric linear systems of the form in eq.(18) using the Krylov subspace, $\mathbb{K}_j$,

$$\mathbb{K}_j = \mathrm{span}\left(\mathbf{r}_0, \mathbb{J}\mathbf{r}_0, \mathbb{J}^2\mathbf{r}_0, ..., \mathbb{J}^{j-1}\mathbf{r}_0\right) \tag{22}$$

where $\mathbf{r}_0 = \mathbb{J}^i \delta\mathbf{X}_0^i + \mathbf{res}\left(\mathbf{X}^i\right)$. In GMRES, the Arnoldi basis vectors form the trial subspace out of which the m^{th}-iteration solution is constructed:

$$\delta\mathbf{X}_m^i = \delta\mathbf{X}_0^i + \mathfrak{k}_0\mathbf{r}_0 + \mathfrak{k}_1\mathbb{J}\mathbf{r}_0 + \mathfrak{k}_2\mathbb{J}^2\mathbf{r}_0 + ... + \mathfrak{k}_m\mathbb{J}^m\mathbf{r}_0 \tag{23}$$

where $(\mathfrak{k}_0, \mathfrak{k}_1, ..., \mathfrak{k}_m)$ are "coordinates" of the m^{th} trial solution in the Krylov subspace. As one can see, one matrix-vector product is required per iteration to create each new trial vector, and the iterations are terminated based on a by-product (free) estimate of the residual that does not require explicit construction of intermediate residual vectors. This is a major advantage of GMRES over other Krylov methods. GMRES has a residual minimization property in the Euclidean norm. The major drawback – it requires the storage of all previous Arnoldi/(Krylov) basis vectors. We use a "flexible" version of GMRES with Arnoldi Modified Gram-Schmidt (double-) orthonormalization[33], without restarts.

One of the particularly useful features of Krylov methods is that they do not require individual elements of the Jacobian matrix $\mathbb{J}$, but instead only matrix-vector multiplications $\mathbb{J}\mathbf{v}$ ($\mathbf{v} \in \left(\mathbf{r}_0, \mathbb{J}\mathbf{r}_0, \mathbb{J}^2\mathbf{r}_0, ...\right)$ are Krylov vectors), which allows for *Jacobian-free implementations*. The action of the Jacobian matrix is approximated by Fréchet derivatives

$$\mathbb{J}\mathbf{v} \approx \frac{\mathbf{res}\left(\mathbf{X} + \varepsilon\mathbf{v}\right) - \mathbf{res}\left(\mathbf{X}\right)}{\varepsilon} \tag{24}$$

where ε is chosen with a fine balance between approximation and floating-point rounding error as

$$\varepsilon = \frac{\sum_{i=1}^{N} bX_i}{N\left|\left|\mathbf{v}\right|\right|_2} \tag{25}$$

N is the total number of unknowns and b is a constant whose value is within a few orders of magnitude of the square root of machine roundoff (here, $b = 10^{-27}$). With the Jacobian-free formulation, the work associated with forming

the Jacobian matrix and its storage can be eliminated, which is a significant saving of both CPU time and storage for each non-linear iteration, provided that the number of Krylov vectors is kept small (see Section III.E). Moreover, in many non-linear applications (like being developed here for all-speed I-SIM with viscosity, conduction and phase change) the Jacobian matrix is not available in analytical form, this makes the JFNK method attractive.

One practically important modification used here is an *inexact Newton's method*. The term "inexact" refers to the accuracy of the iterative linear solver. The basic idea behind it is that the linear system must be solved to a tight tolerance only when the added accuracy matters – i.e. when it affects the convergence of the Newton's iterations. This is accomplished by making the convergence of the linear residual proportional to the non-linear residual:

$$\left|\left|\mathbb{J}^i \delta \mathbf{X}_m^i + \mathbf{res}\left(\mathbf{X}^i\right)\right|\right|_2 < \max\left(\mathrm{tol}_{\mathrm{GMRES}}, \nu \left|\left|\mathbf{res}\left(\mathbf{X}^i\right)\right|\right|_2\right) \tag{26}$$

where $\mathrm{tol}_{\mathrm{GMRES}} = 10^{-22}$ and the subscript *m* refers to the m^{th} trial solution of GMRES and $\nu = 10^{-3}$. This allows saving of some CPU time and storage for Krylov vectors at the early Newton's iterations, while tightening the linear solver's convergence in the asymptotic non-linear convergence range.

E. Preconditioning of GMRES

Because GMRES stores all of the previous Krylov vectors, it is necessary to keep the number of iterations relatively small, to prevent the storage and CPU time from becoming prohibitive. This can be accomplished by preconditioning the linear system. *Preconditioning is a transformation of the original linear system into one with the same solution, but is easier to solve with an iterative solver*. We will be using the right-preconditioned form of the linear system,

$$\underbrace{\mathbb{J}^i \mathbb{P}^{-1}}_{\hat{\mathbb{J}}} \underbrace{\mathbb{P}\delta \mathbf{X}^i}_{\delta \hat{\mathbf{X}}^i} = -\mathbf{res}\left(\mathbf{X}^i\right) \tag{27}$$

where $\mathbb{P}^{-1}$ approximates $\mathbb{J}^{-1}$. The right-preconditioned version of eq.(24) is

$$\mathbb{J}\mathbb{P}^{-1}\mathbf{v} \approx \frac{\mathbf{res}\left(\mathbf{X} + \varepsilon \mathbb{P}^{-1}\mathbf{v}\right) - \mathbf{res}\left(\mathbf{X}\right)}{\varepsilon} \tag{28}$$

This operation is applied once per GMRES iteration, in two steps:

I. Preconditioning: approximately solve $\delta \hat{\mathbf{X}}^i = \mathbb{P}^{-1}\mathbf{v}$
II. Compute matrix-free product: $\mathbb{J}\delta \hat{\mathbf{X}}^i \approx \left(\mathbf{res}\left(\mathbf{X}^i + \varepsilon \delta \hat{\mathbf{X}}^i\right) - \mathbf{res}\left(\mathbf{X}^i\right)\right)/\varepsilon$

Finding a good preconditioner is often a combination of art, science, and intuition. A mathematically good preconditioner should efficiently cluster the eigenvalues of the iteration matrix[19,32]. A preconditioner can also be defined as any subsidiary *approximate solver* that is combined with an outer iteration technique (e.g., multigrid or one of the Krylov iteration solvers). One of the simplest and most popular ways of defining a preconditioner is to perform an *incomplete lower-upper (ILU) factorization* of the original matrix $\mathbb{J}$. A number of variations - ILU(k), ILUT, ILUS, ILUC, etc. - are discussed in[33].

An important class of preconditioners for the JFNK method is referred to as *Physics-Based-Preconditioning (PBP)* or PDE-based[19]. The motivation behind this approach is that there exist numerous legacy operator-split algorithms to solve nonlinear systems. These algorithms were developed with some insight into physical time scales of the problem. A direct benefit of this insight – a reduced implicit system, or a sequence of segregated semi-implicit solvers can be applied, instead of attempting to solve the fully-coupled system. Relevant fluid dynamics examples include the semi-implicit all-speed-flow *Implicit Continuous-fluid Eulerian* (*ICE*) algorithm[15], the semi-implicit

incompressible-flow ***SIMPLE***[29] and the ***Projection***[7] algorithms. The successes of PB preconditioning are exemplified in numerous recent publications[18,19,26,27].

In the present section, we introduce three preconditioners for our rDG-based I-SIM method: *(i)* "Full-Coupling" (FC); *(ii)* Physics-Based, "Block-Diagonal" (BD); and *(iii)* Physics-Based, "Internal energy-Pressure-Velocity Partially Decoupled" (IPV/PD). Efficiencies of these preconditioners will be studied in Section IV.B.

Full-Coupling (FC) preconditioning. By removing interfacial markers and re-arranging the solution vector in the following order

$$\bar{\mathbf{X}} = \left(\rho_1^{(0)}, \rho_1^{(1)}, \dots, \rho_1^{(p)}, m_1^{(0)}, m_1^{(1)}, \dots, m_1^{(p)}, E_1^{(0)}, E_1^{(1)}, \dots, E_1^{(p)}, \rho_2^{(0)}, \rho_2^{(1)}, \dots, \rho_2^{(p)}, \dots, E_{N_{\text{cells}}}^{(p)}\right)^{\mathsf{T}} \tag{29}$$

the linear system is reduced to $\bar{\mathbb{J}}\delta\bar{\mathbf{X}} = -\mathbf{r\bar{e}s}$, where the Jacobian matrix $\bar{\mathbb{J}}$ is band-diagonal in 1D. The bandwidth of the rDG_{n} discretization is $(6p+5)$. Non-zero elements of $\bar{\mathbb{J}}$ in the band can be approximated by automated differentiation of the type

$$\bar{\mathbb{J}}_{i,j} = \frac{r\bar{e}s_i\left(\bar{\mathbf{X}} + e_j\varepsilon\right) - r\bar{e}s_i\left(\bar{\mathbf{X}}\right)}{\varepsilon} \tag{30}$$

where $r\bar{e}s_i$ is the residual of the i^{th} nonlinear equation, $\bar{\mathbf{X}}$ is an unperturbed solution from the current Newton's iteration, and e_j is the unit normal vector in the *j*-direction. The perturbation parameter was chosen as $\varepsilon = 10^{-27}$.

$\mathbb{P}_{\text{FC}} = \bar{\mathbb{J}}$ is a very efficient preconditioner in 1D, which in the absence of an interface collapses all eigenvalues to a single point, converging GMRES in 2-3 Krylov iterations. Since interfacial markers are left unpreconditioned, the actual number of Krylov steps in the I-SIM implementation is ~10. In 1D, FC preconditioner is probably the most efficient option, since $\mathbb{P}_{\text{FC}}$ can be inverted directly using band-diagonal LU decomposition[30] (a generalization of the Thomson's TDMA algorithm). In multi-D however, $\bar{\mathbb{J}}$ cannot be reduced to the band-diagonal form (it will have several bands, for different spatial directions), which would require involvement of ILU-type preconditioning strategies[33].

Block-Diagonal (BD) physics-based preconditioning. One useful type of preconditioning can be derived from the FC by linearizing (moving elements at new time in the matrix to old time on the right hand side) elements of $\bar{\mathbb{J}}$ coming from neighbor cells. This preconditioning operation would look like

$$\underbrace{\begin{bmatrix} \tilde{\hat{\mathbb{J}}}_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \tilde{\hat{\mathbb{J}}}_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & \cdots\cdots & & & \\ & & & & \cdots\cdots & & & \\ & & & & \cdots\cdots & & & \\ & & & & \cdots\cdots & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & \tilde{\hat{\mathbb{J}}}_{N_{\text{cells}}-1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \tilde{\hat{\mathbb{J}}}_{N_{\text{cells}}} \end{bmatrix}}_{\mathbb{P}_{\text{BD}}} \underbrace{\begin{pmatrix} \delta\mathbb{U}_1 \\ \delta\mathbb{U}_2 \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \delta\mathbb{U}_{N_{\text{cells}}-1} \\ \delta\mathbb{U}_{N_{\text{cells}}} \end{pmatrix}}_{\delta\hat{\mathbf{X}}} = \underbrace{\begin{pmatrix} \tilde{\mathbf{r}\hat{\mathbf{e}}\mathbf{s}}_1 \\ \tilde{\mathbf{r}\hat{\mathbf{e}}\mathbf{s}}_2 \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \tilde{\mathbf{r}\hat{\mathbf{e}}\mathbf{s}}_{N_{\text{cells}}-1} \\ \tilde{\mathbf{r}\hat{\mathbf{e}}\mathbf{s}}_{N_{\text{cells}}} \end{pmatrix}}_{\mathbf{v}} \tag{31}$$

where $\mathbb{U}_j = \left(\rho_j^{(0)}, \rho_j^{(1)}, \dots, \rho_j^{(p)}, m_j^{(0)}, m_j^{(1)}, \dots, m_j^{(p)}, E_j^{(0)}, E_j^{(1)}, \dots, E_j^{(p)}\right)^{\mathsf{T}}$ is the rDG's local solution vector, at cell *j*, of size $N_j = 3(p+1)$. Accordingly, $\tilde{\mathbf{r}\hat{\mathbf{e}}\mathbf{s}}_{\mathbf{j}}$ is the local residual/(Krylov) vector, at cell *j*. The preconditioning matrix is block-diagonal. Each block $\tilde{\hat{\mathbb{J}}}_j$ is an $N_j \times N_j$ matrix, which is LU-decomposed once per Newton iteration,

and the local solution vector $\delta\mathbb{U}_j$ is obtained by back-substitution at each preconditioning stage. This simple preconditioning is designed to target local effects (reaction, gravity, other sources); though, some non-local effects are also accounted for due to DG formulation, since the derivatives of the flow variables are parts of the local solution vector. This physics-based preconditioning will be particularly efficient when the stiffness of the non-linear system is caused by fast local reaction terms.

FC preconditioning in primitive variables. Before introducing our next PBP, we shall describe how to transform the preconditioned linear system written in conservative variables $\mathbb{U} = (\rho, m, E)^{\mathsf{T}}$ into the preconditioned linear system expressed in primitive variables $\mathbb{V} = (P, u, i)^{\mathsf{T}}$. While solving a nonlinear system in conservative variables is very attractive (each Newton's step is conservative to machine accuracy; if locked in the limit cycle, one can stop Newton's iterations without full convergence – the solution will be still fully conservative and accurate[*]), the physical insight needed for PBP of linear solves is better gained when operating with primitive variables, such as pressure, velocity, temperature or internal energy. A particular difficulty for transformation $\mathbb{U} \leftrightarrow \mathbb{V}$ arises due to DG discretization, since an adequate/consistent transformation from/to higher-order DG degrees of freedom must be supplied. The approach introduced below offers a robust solution to this difficulty.

For transparency of the following presentation, linear system eq.(18) can be expressed in the block-vector form

$$\underbrace{\begin{bmatrix} \mathbb{J}_{\rho\rho} & \mathbb{J}_{\rho m} & \mathbb{J}_{\rho E} \\ \mathbb{J}_{m\rho} & \mathbb{J}_{mm} & \mathbb{J}_{mE} \\ \mathbb{J}_{E\rho} & \mathbb{J}_{Em} & \mathbb{J}_{EE} \end{bmatrix}}_{\mathbb{J}^i} \underbrace{\begin{pmatrix} \delta\mathcal{R} \\ \delta\mathcal{M} \\ \delta\mathcal{E} \end{pmatrix}}_{\delta\mathbf{X}^i} = - \underbrace{\begin{pmatrix} \mathbf{res}_{\mathcal{R}} \\ \mathbf{res}_{\mathcal{M}} \\ \mathbf{res}_{\mathcal{E}} \end{pmatrix}}_{\mathbf{res}_{\mathbb{J}}(\mathbf{X}^i)} \tag{32}$$

where $\mathcal{R}$, $\mathcal{M}$ and $\mathcal{E}$ are density, momentum and total energy vectors, arranged as

$$\mathcal{R} = \left(\rho_1^{(0)}, \rho_1^{(1)}, \ldots, \rho_1^{(p)}, \rho_2^{(0)}, \rho_2^{(1)}, \ldots, \rho_2^{(p)}, \ldots, \rho_{N_{\text{cells}}}^{(0)}, \rho_{N_{\text{cells}}}^{(1)}, \ldots, \rho_{N_{\text{cells}}}^{(p)}\right)^{\mathsf{T}}$$
$$\mathcal{M} = \left(m_1^{(0)}, m_1^{(1)}, \ldots, m_1^{(p)}, m_2^{(0)}, m_2^{(1)}, \ldots, m_2^{(p)}, \ldots, m_{N_{\text{cells}}}^{(0)}, m_{N_{\text{cells}}}^{(1)}, \ldots, m_{N_{\text{cells}}}^{(p)}\right)^{\mathsf{T}}$$
$$\mathcal{E} = \left(E_1^{(0)}, E_1^{(1)}, \ldots, E_1^{(p)}, E_2^{(0)}, E_2^{(1)}, \ldots, E_2^{(p)}, \ldots, E_{N_{\text{cells}}}^{(0)}, E_{N_{\text{cells}}}^{(1)}, \ldots, E_{N_{\text{cells}}}^{(p)}\right)^{\mathsf{T}}$$

The $\mathbf{res}_{\mathcal{R}}$, $\mathbf{res}_{\mathcal{M}}$, $\mathbf{res}_{\mathcal{E}}$ are corresponding residual vectors, arranged accordingly. Each block-element $\mathbb{J}_{n,m}$ of eq.(32) is of size $(N_{\text{cells}}(p+1) \times N_{\text{cells}}(p+1))$ and represents a generally nonlinear coupling between the n^{th} and m^{th} conservative variables. Next, we define the transformation $\mathbb{U} \leftrightarrow \mathbb{V}$ by the following matrices:

$$\mathbb{A} \equiv \frac{\partial \mathbb{U}}{\partial \mathbb{V}} = \begin{pmatrix} \left.\frac{\partial \rho}{\partial P}\right|_{u,i=const} & \left.\frac{\partial \rho}{\partial u}\right|_{P,i=const} & \left.\frac{\partial \rho}{\partial i}\right|_{P,u=const} \\ \left.\frac{\partial m}{\partial P}\right|_{u,i=const} & \left.\frac{\partial m}{\partial u}\right|_{P,i=const} & \left.\frac{\partial m}{\partial i}\right|_{P,u=const} \\ \left.\frac{\partial E}{\partial P}\right|_{u,i=const} & \left.\frac{\partial E}{\partial u}\right|_{P,i=const} & \left.\frac{\partial E}{\partial i}\right|_{P,u=const} \end{pmatrix} \tag{33}$$

and

$$\mathbb{A}^{-1} \equiv \frac{\partial \mathbb{V}}{\partial \mathbb{U}} = \begin{pmatrix} \left.\frac{\partial P}{\partial \rho}\right|_{m,E=const} & \left.\frac{\partial P}{\partial m}\right|_{\rho,E=const} & \left.\frac{\partial P}{\partial E}\right|_{\rho,m=const} \\ \left.\frac{\partial u}{\partial \rho}\right|_{m,E=const} & \left.\frac{\partial u}{\partial m}\right|_{\rho,E=const} & \left.\frac{\partial u}{\partial E}\right|_{\rho,m=const} \\ \left.\frac{\partial i}{\partial \rho}\right|_{m,E=const} & \left.\frac{\partial i}{\partial m}\right|_{\rho,E=const} & \left.\frac{\partial i}{\partial E}\right|_{\rho,m=const} \end{pmatrix} \tag{34}$$

so that

[*] But not necessary robust.

$$\delta\mathbb{U} = \mathbb{A}\delta\mathbb{V} \quad \text{and} \quad \delta\mathbb{V} = \mathbb{A}^{-1}\delta\mathbb{U} \tag{35}$$

Elements of the transformation matrices eqs.(33) and (34) can be computed analytically from the equation of state.

Thus, eq.(32) can be transformed to

$$\underbrace{\mathbb{A}^{-1}\mathbb{J}^{i}\mathbb{A}}_{\mathbb{J}_{\mathbb{V}}}\delta\mathbb{V}^{i} = -\underbrace{\mathbb{A}^{-1}\mathbf{res}_{\mathbb{U}}\left(\mathbf{X}^{i}\right)}_{\mathbf{res}_{\mathbb{V}}\left(\mathbf{X}^{i}\right)} \tag{36}$$

In the DG formulation, the solution vectors $\mathbb{U}$ and $\mathbb{V}$ are composed of the cell-average values and their higher-order perturbations, which means that we actually need a transformation matrix in the form:

$$\mathbb{A} = \begin{pmatrix}
\frac{\partial\rho^{(0)}}{\partial P^{(0)}} & \frac{\partial\rho^{(0)}}{\partial P^{(1)}} & \cdots & \frac{\partial\rho^{(0)}}{\partial P^{(p)}} & \frac{\partial\rho^{(0)}}{\partial u^{(0)}} & \frac{\partial\rho^{(0)}}{\partial u^{(1)}} & \cdots & \frac{\partial\rho^{(0)}}{\partial u^{(p)}} & \frac{\partial\rho^{(0)}}{\partial i^{(0)}} & \frac{\partial\rho^{(0)}}{\partial i^{(1)}} & \cdots & \frac{\partial\rho^{(0)}}{\partial i^{(p)}} \\
\frac{\partial\rho^{(1)}}{\partial P^{(0)}} & \frac{\partial\rho^{(1)}}{\partial P^{(1)}} & \cdots & \frac{\partial\rho^{(1)}}{\partial P^{(p)}} & \frac{\partial\rho^{(1)}}{\partial u^{(0)}} & \frac{\partial\rho^{(1)}}{\partial u^{(1)}} & \cdots & \frac{\partial\rho^{(1)}}{\partial u^{(p)}} & \frac{\partial\rho^{(1)}}{\partial i^{(0)}} & \frac{\partial\rho^{(1)}}{\partial i^{(1)}} & \cdots & \frac{\partial\rho^{(1)}}{\partial i^{(p)}} \\
 & & & & & \vdots & & & & & & \\
\frac{\partial\rho^{(p)}}{\partial P^{(0)}} & \frac{\partial\rho^{(p)}}{\partial P^{(1)}} & \cdots & \frac{\partial\rho^{(p)}}{\partial P^{(p)}} & \frac{\partial\rho^{(p)}}{\partial u^{(0)}} & \frac{\partial\rho^{(p)}}{\partial u^{(1)}} & \cdots & \frac{\partial\rho^{(p)}}{\partial u^{(p)}} & \frac{\partial\rho^{(p)}}{\partial i^{(0)}} & \frac{\partial\rho^{(p)}}{\partial i^{(1)}} & \cdots & \frac{\partial\rho^{(p)}}{\partial i^{(p)}} \\
\frac{\partial m^{(0)}}{\partial P^{(0)}} & \frac{\partial m^{(0)}}{\partial P^{(1)}} & \cdots & \frac{\partial m^{(0)}}{\partial P^{(p)}} & \frac{\partial m^{(0)}}{\partial u^{(0)}} & \frac{\partial m^{(0)}}{\partial u^{(1)}} & \cdots & \frac{\partial m^{(0)}}{\partial u^{(p)}} & \frac{\partial m^{(0)}}{\partial i^{(0)}} & \frac{\partial m^{(0)}}{\partial i^{(1)}} & \cdots & \frac{\partial m^{(0)}}{\partial i^{(p)}} \\
 & & & & & \vdots & & & & & & \\
\frac{\partial E^{(0)}}{\partial P^{(0)}} & \frac{\partial E^{(0)}}{\partial P^{(1)}} & \cdots & \frac{\partial E^{(0)}}{\partial P^{(p)}} & \frac{\partial E^{(0)}}{\partial u^{(0)}} & \frac{\partial E^{(0)}}{\partial u^{(1)}} & \cdots & \frac{\partial E^{(0)}}{\partial u^{(p)}} & \frac{\partial E^{(0)}}{\partial i^{(0)}} & \frac{\partial E^{(0)}}{\partial i^{(1)}} & \cdots & \frac{\partial E^{(0)}}{\partial i^{(p)}} \\
 & & & & & \vdots & & & & & &
\end{pmatrix} \tag{37}$$

To compute elements of the transformation matrices, $\left(\frac{\partial\mathbb{U}^{(l)}}{\partial\mathbb{V}^{(m)}}\right)_{l,m=0,\ldots,p}$, we exploit weak statements as explained below.

First, the in-cell DG solutions for conservative and primitive variables are defined as

$$\mathbb{U}_{h}(x) = \sum_{l=0}^{p}\mathbb{U}^{(l)}\mathfrak{L}_{l}(x) \tag{38}$$

$$\mathbb{V}_{h}(x) = \sum_{l=0}^{p}\mathbb{V}^{(l)}\mathfrak{L}_{l}(x) \tag{39}$$

On the other hand, conservative variables are connected to primitive variables through the equation of state:

$$\mathbb{U}_{h}(x) = \mathbf{EoS}(\mathbb{V}) \tag{40}$$

Applying the chain rule, we obtain:

$$\frac{\partial \mathbb{U}_h(x)}{\partial \mathbb{V}^{(m)}} = \frac{\partial \mathbf{EoS}(\mathbb{V})}{\partial \mathbb{V}} \frac{\partial \mathbb{V}_h(x)}{\partial \mathbb{V}^{(m)}} = \frac{\partial \mathbf{EoS}(\mathbb{V})}{\partial \mathbb{V}} \mathfrak{L}_{(m)}(x) \tag{41}$$

At the same time, differentiating eq.(38):

$$\frac{\partial \mathbb{U}_h(x)}{\partial \mathbb{V}^{(m)}} = \sum_{l=0}^{p} \frac{\partial \mathbb{U}^{(l)}}{\partial \mathbb{V}^{(m)}} \mathfrak{L}_{(l)}(x) \tag{42}$$

Next, multiplying the left- and right- hand-sides of eq.(42) on Legendre polynomials $\mathfrak{L}_{(l)}$, integrating over $\mathcal{I}_j$ and using orthogonality property, we get:

$$\left(\frac{\partial \mathbb{U}^{(l)}}{\partial \mathbb{V}^{(m)}} \right)_j = \frac{1}{C_l} \int_{\mathcal{I}_j} \frac{\partial \mathbb{U}_h(x)}{\partial \mathbb{V}^{(m)}} \mathfrak{L}_{(l)}(x)\, dx \tag{43}$$

Finally, using eq.(41),

$$\left(\frac{\partial \mathbb{U}^{(l)}}{\partial \mathbb{V}^{(m)}} \right)_j = \frac{1}{C_l} \int_{\mathcal{I}_j} \frac{\partial \mathbf{EoS}(\mathbb{V})}{\partial \mathbb{V}} \mathfrak{L}_{(m)}(x)\, \mathfrak{L}_{(l)}(x)\, dx \tag{44}$$

which are the elements of the transformation matrix eq.(37). Integrals on the r.h.s. are computed with a 12-point Gaussian quadrature formula.

We numerically verified that the FC preconditioning in primitive variables $\mathbb{P}_{\text{FC}} = \mathbb{J}_{\text{V}}$ is almost identical to the one in conservative variables, collapsing eigenvalues of the no-interface problem to a single point upon convergence of Newton's iteration and requiring 2-3 Krylov vectors to converge GMRES.

Internal energy-Pressure-Velocity/Partially Decoupled (IPV/PD) physics-based preconditioning. Eq.(36) can be expressed in the following block-vector form:

$$\begin{bmatrix} \mathbb{J}_{PP} & \mathbb{J}_{Pu} & \mathbb{J}_{Pi} \\ \mathbb{J}_{uP} & \mathbb{J}_{uu} & \mathbb{J}_{ui} \\ \mathbb{J}_{iP} & \mathbb{J}_{iu} & \mathbb{J}_{ii} \end{bmatrix} \begin{pmatrix} \delta\mathcal{P} \\ \delta\mathcal{U} \\ \delta\mathcal{I} \end{pmatrix} = - \begin{pmatrix} \mathbf{res}_{\mathcal{P}} \\ \mathbf{res}_{\mathcal{U}} \\ \mathbf{res}_{\mathcal{I}} \end{pmatrix} \tag{45}$$

where the terms are interpreted the same way as in eq.(32).

First, we decouple the internal energy equation from the pressure and velocity equations, linearizing it as

$$\begin{bmatrix} 0 & 0 & \mathbb{J}_{ii} \end{bmatrix} \begin{pmatrix} \delta\mathcal{P} \\ \delta\mathcal{U} \\ \delta\mathcal{I} \end{pmatrix} = -\mathbf{res}_{\mathcal{I}} \tag{46}$$

On the one hand, zeroing-out blocks $\mathbb{J}_{iP}$ and $\mathbb{J}_{iu}$ numerically means that the dependence of internal energy on pressure and velocity is treated explicitly ("frozen" at the previous Newton's iteration values). On the other hand, heat conduction is accounted for implicitly. Eq.(46) is then solved directly for $\delta\mathcal{I}$ using the band-diagonal LU decomposition[*]:

[*] In multi-D, this step would require multigrid algorithms.

$$\underline{\textbf{\textit{Step I:}}} \quad \delta\mathcal{I} = -\mathbb{J}_{ii}^{-1}\,\mathbf{res}_{\mathcal{I}}$$

Next, algebraic manipulations of the coupled pressure-velocity equations

$$\underbrace{\begin{bmatrix} \mathbb{J}_{PP} & \mathbb{J}_{Pu} \\ \mathbb{J}_{uP} & \mathbb{J}_{uu} \end{bmatrix}}_{Pressure\text{-}velocity\ matrix} \begin{pmatrix} \delta\mathcal{P} \\ \delta\mathcal{U} \end{pmatrix} = -\begin{pmatrix} \mathbf{res}_{\mathcal{P}} + \mathbb{J}_{Pi}\delta\mathcal{I} \\ \mathbf{res}_{\mathcal{U}} + \mathbb{J}_{ui}\delta\mathcal{I} \end{pmatrix} \tag{47}$$

allow to derive the following *pressure-Poisson equation*:

$$\underbrace{\left(\mathbb{J}_{PP} - \mathbb{J}_{Pu}\mathbb{J}_{uu}^{-1}\mathbb{J}_{uP}\right)}_{Laplacian,\ \mathbb{L}}\delta\mathcal{P} = \underbrace{-\left(\mathbf{res}_{\mathcal{P}} + \mathbb{J}_{Pi}\delta\mathcal{I} - \mathbb{J}_{Pu}\mathbb{J}_{uu}^{-1}\left(\mathbf{res}_{\mathcal{U}} + \mathbb{J}_{ui}\delta\mathcal{I}\right)\right)}_{\mathbf{b}} \tag{48}$$

The Laplacian $\mathbb{L}$ is related to the Schur complement of the pressure-velocity matrix in the system eq.(47). To further simplify, we diagonalize $\mathbb{J}_{uu}^{-1} \approx [Diag\,(\mathbb{J}_{uu})]^{-1} = \tilde{\mathbb{J}}_{uu}^{-1}$ making the evaluation of the triple matrix product $\left(\mathbb{J}_{Pu}\mathbb{J}_{uu}^{-1}\mathbb{J}_{uP}\right)$ trivial, and using the *approximate Laplacian* $\tilde{\mathbb{L}} \equiv \left(\mathbb{J}_{PP} - \mathbb{J}_{Pu}\tilde{\mathbb{J}}_{uu}^{-1}\mathbb{J}_{uP}\right)$ in the second step of the preconditioning, where we solve the Poisson equation directly for $\delta\mathcal{P}$ using the band-diagonal LU decomposition[*]:

$$\underline{\textbf{\textit{Step II:}}} \quad \delta\mathcal{P} = \tilde{\mathbb{L}}^{-1}\mathbf{b}$$

The final stage of this preconditioning is to find velocity as

$$\underline{\textbf{\textit{Step III:}}} \quad \delta\mathcal{U} = -\mathbb{J}_{uu}^{-1}\left(\mathbf{res}_{\mathcal{U}} + \mathbb{J}_{uP}\delta\mathcal{P} + \mathbb{J}_{ui}\delta\mathcal{I}\right)$$

Notably, viscous stress terms are directly accounted for in steps II and III. Multi-D extension would require directional splitting, solving for steps II-III sequentially, for each spatial direction, with optional alternation of the directions.

The driving considerations behind our IPV/PD preconditioner is similar to the ones in the well-known operator-split algorithms for low-speed-compressible and incompressible flows (ICE[15], Simple[29], Projection[7]), i.e.:

a) identify the fastest physics, and
b) reduce/split to the sequence of segregated implicit scalar problems, easily solved directly or by multigrid methods.

IV. Numerical Examples

A. Accuracy/Convergence

To demonstrate accuracy of our numerical algorithm, we introduce the following manufactured solution:

[*] In multi-D, this step would require multigrid algorithms.

$$
\begin{aligned}
\rho(x,t) &= \rho_{\mathrm{f,min}} + \left(\rho_{\mathrm{f,max}} - \rho_{\mathrm{f,min}}\right) \operatorname{sech}\left(\frac{x - U_{\mathrm{int}} t}{\delta}\right) \\
P(x,t) &= P_{\mathrm{min}} + \tfrac{1}{2}\left(\Delta P_0 \left[1 + \tanh\left(\frac{x - U_{\mathrm{int}} t}{\delta}\right)\right] + \Delta P_1 \left[1 + \tanh\left(\frac{x - C_1 t}{\delta_1}\right)\right]\right) \\
u(x,t) &= U_{\mathrm{min}} + \left(U_{\mathrm{int}} - U_{\mathrm{min}}\right)\left(1 + \tanh\left(\frac{x - U_{\mathrm{int}} t}{\delta}\right)\right) \\
&\mathrm{f} = \mathrm{L} \text{ if } x < \left(U_{\mathrm{int}} t\right) \text{ and } \mathrm{f} = \mathrm{G} \text{ otherwise} \\
\eta_{\mathrm{L}} = \eta_{\mathrm{G}} = \eta; &\qquad \frac{\beta_{\mathrm{L}}}{\beta_{\mathrm{G}}} = \frac{\rho_{\mathrm{L,max}}\left(\gamma_{\mathrm{L}} - 1\right)}{\rho_{\mathrm{G,max}}\left(\gamma_{\mathrm{G}} - 1\right)}
\end{aligned}
\tag{49}
$$

where $\rho_{\mathrm{f,min}}$, $\rho_{\mathrm{f,max}}$, P_{min}, ΔP_0, ΔP_1, U_{min}, U_{int}, C_1, δ, δ_1, γ_{f}, β_{f} and η_{f} are input parameters. Dynamics of the manufactured solution used in the present study are shown in Figure 2 and Figure 3, together with corresponding input parameters, history of heat fluxes and viscous stresses and balances of forcing/source terms $\mathbb{S}_E$ and $\mathbb{S}_m$ for t=100. The solution is manufactured to satisfy the following requirements:

a) Mach number is relatively low ($M < 0.1$);
b) There are two dynamic velocity scales (U_{int} and C_1), both are significantly slower than the sound speed and comparable to material velocity (in fact, U_{int} is equal to fluid velocity at the interface);
c) Heat and momentum fluxes are continuous across the interface; and
d) Parabolic terms (viscous diffusion and heat conduction) are significant and comparable to hyperbolic terms (see Figure 3c,d).

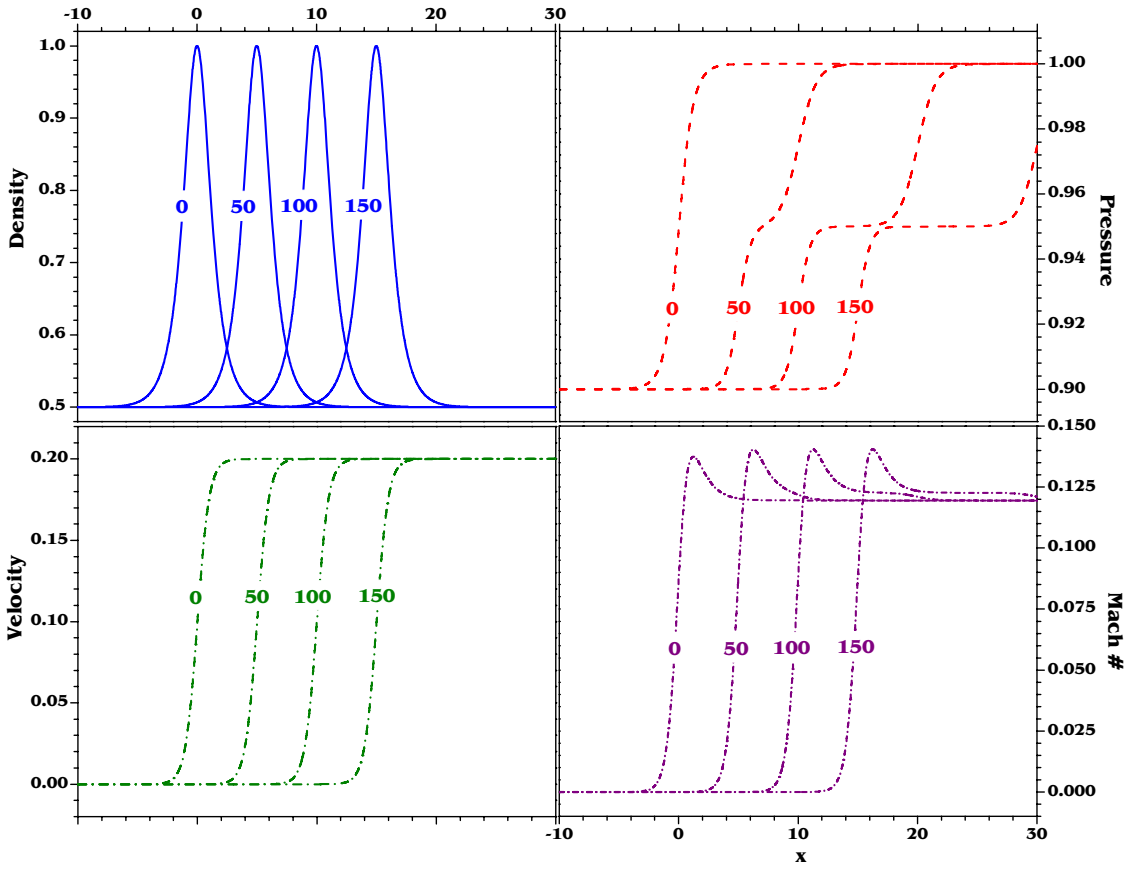

Figure 2. Dynamics of density, pressure, velocity and Mach number for manufactured solution: $\rho_{\mathrm{f,min}} = 0.5$, $\rho_{\mathrm{f,max}} = 1$, $P_{\mathrm{min}} = 0.9$, $\Delta P_0 = \Delta P_1 = 0.05$, $U_{\mathrm{min}} = 0$, $U_{\mathrm{int}} = 0.1$, $C_1 = 0.2$, $\delta = 1$, $\delta_1 = 1.5$, $\gamma_{\mathrm{f}} = 1.4$, $\beta_{\mathrm{f}} = 1$ **and** $\eta_{\mathrm{f}} = 1$, **f=L,G.**

To enforce the manufactured solution eqs.(49), the following source terms are included:

$$\mathbb{S}_\rho = \frac{\Delta U \varsigma_0 \left(\rho_{\mathrm{f,min}} \varsigma_0 + \Delta\rho_\mathrm{f}\left(2\varsigma_0^2 - 1\right)\right)}{\delta}$$

$$\mathbb{S}_m = \frac{1}{2}\left(\frac{\Delta P_1 \varsigma_1^2}{\delta_1} + \varsigma_0 \frac{2\delta\Delta\rho_\mathrm{f}\Delta U\left(\varsigma_0^2\alpha_1 - \tau_0^2\alpha_4\right) + \varsigma_0\left(\delta\left(\Delta P_0 + 2\rho_{\mathrm{f,min}}\Delta U\alpha_1\right) + 4\Delta U\tau_0\eta_\mathrm{f}\right)}{\delta^2}\right)$$

$$\mathbb{S}_E = \frac{1}{2}\left(\begin{array}{c} -\dfrac{\beta_\mathrm{f}\left(\frac{\Delta\rho_\mathrm{f}\varsigma_0\left(2\alpha_2\left(\Delta P_0\varsigma_0^2 + \alpha_7\delta\right)\tau_0 + \alpha_3\left(2\Delta\rho_\mathrm{f}\varsigma_0\tau_0^2 + \alpha_2\left(\varsigma_0^2 - \tau_0^2\right)\right)\right)}{\delta^2} - 2\alpha_2^2\alpha_8\right)}{\alpha_2^3\left(\gamma_\mathrm{f} - 1\right)} + \\ + \dfrac{\alpha_5\Delta\rho_\mathrm{f}\varsigma_0\tau_0 U_{\mathrm{int}}}{\delta} - 2\alpha_2\left(\alpha_4\alpha_6 U_{\mathrm{int}} + \dfrac{C_1\alpha_7 + \frac{\varsigma_0\left(\Delta P_0\varsigma_0 + \frac{\alpha_3\Delta\rho_\mathrm{f}\tau_0}{\alpha_2}\right)U_{\mathrm{int}}}{\delta}}{2\alpha_2\left(\gamma_\mathrm{f} - 1\right)}\right) + \\ + \alpha_4\left(\dfrac{\alpha_7\gamma_\mathrm{f} + \frac{\varsigma_0\left(\Delta P_0\gamma_\mathrm{f}\varsigma_0 + \alpha_4\left(\gamma_\mathrm{f} - 1\right)\alpha_{10}\right)}{\delta}}{\left(\gamma_\mathrm{f} - 1\right)} + \dfrac{4\alpha_6\tau_0\eta_\mathrm{f}}{\delta}\right) + \\ + \alpha_6\left(\alpha_3 + \alpha_2\alpha_5 - 2\left(\gamma_\mathrm{f}\Pi_\mathrm{f} + \alpha_6\eta_\mathrm{f}\right)\right) \end{array}\right) \tag{50}$$

where $\Delta U = U_{\mathrm{int}} - U_{\mathrm{min}}$, $\varsigma_0 = \mathrm{sech}\frac{x - U_{\mathrm{int}}t}{\delta}$, $\varsigma_1 = \mathrm{sech}\frac{x - C_1 t}{\delta_1}$, $\tau_0 = \tanh\frac{x - U_{\mathrm{int}}t}{\delta}$, $\tau_1 = \tanh\frac{x - C_1 t}{\delta_1}$, $\Delta\rho_\mathrm{f} = \rho_{\mathrm{f,max}} - \rho_{\mathrm{f,min}}$, $\alpha_1 = 2\Delta U\tau_0 + U_{\mathrm{int}}$, $\alpha_2 = \rho_{\mathrm{f,min}} + \Delta\rho_\mathrm{f}\varsigma_0$,

$$\alpha_3 = \left(\Delta P_0 + \Delta P_1 + 2\left(\gamma_\mathrm{f}\Pi_\mathrm{f} + P_{\mathrm{min}}\right) + \Delta P_0\tau_0 + \Delta P_1\tau_1\right),$$

$$\alpha_4 = U_{\mathrm{int}} + \Delta U\tau_0, \quad \alpha_5 = \frac{\alpha_3}{\left(\gamma_\mathrm{f} - 1\right)\alpha_2} + \alpha_4^2, \quad \alpha_6 = \frac{\varsigma_0^2\Delta U}{\delta}, \quad \alpha_7 = \frac{\Delta P_1\varsigma_1^2}{\delta_1},$$

$$\alpha_8 = \frac{\Delta P_0\tau_0\varsigma_0^2}{\delta^2} + \frac{\alpha_7\tau_1}{\delta_1}, \quad \alpha_9 = \tau_0\left(\tau_0 + 1\right),$$

$$\alpha_{10} = \left(\Delta\rho_\mathrm{f}\tau_0^2 - 2\alpha_2\varsigma_0\right)U_{\mathrm{min}} - \left(\rho_{\mathrm{f,min}}\left(2\left(\varsigma_0 - 1\right)\varsigma_0 - \alpha_9\right) + \rho_{\mathrm{f,max}}\left(\alpha_9 - 2\varsigma_0^2\right)\right)U_{\mathrm{int}}$$

Interfacial marker is moved with material velocity at x_m, recovered from rDG solutions in cut-cells.

Convergence in time is demonstrated in Figure 4a. To ensure no-interference from spatial discretization errors, we used a sufficiently fine grid (N=400) and 6th-order-accurate quintic DG elements. $\mathcal{L}_2$-norms of errors for density at cell centers are computed as

$$\mathcal{L}_2\left(\rho_{\mathrm{at\ c.c.}}\right) = \frac{\sqrt{\sum_{i=1}^{N_{\mathrm{cells}}}\left(\rho_i - \rho_i^{\mathrm{exact}}\right)^2}}{N_{\mathrm{cells}}} \tag{51}$$

where N_{cells} is the total number of cells.

Five implicit time discretizations are tested: *the 1st-order backward-Euler* (BE_1); *the 2nd-order Crank-Nicholson* (CN_2) and three "*explicit, singly diagonally implicit Runge-Kutta* ($\mathrm{ESDIRK}_{3\text{-}5}$)[5]" schemes. As seen from Figure 4a, all schemes converge with nearly-theoretical rate. We are able to step over CFL stability limits due to stiff waves or viscosity/conductivity, at will; running with stability CFL numbers* as high as 240, converging with 2-3 Newton

* Through the rest of the manuscript, we will define *stability CFL number* as

steps per each Runge-Kutta level and involving only a few (~10) Krylov vectors per each linear solve (preconditioning of GMRES is discussed in Section IV.B). ESDIRK schemes are found to be extremely accurate – with many orders of magnitude lower errors than BE_1 and CN_2, even under significantly larger time steps.

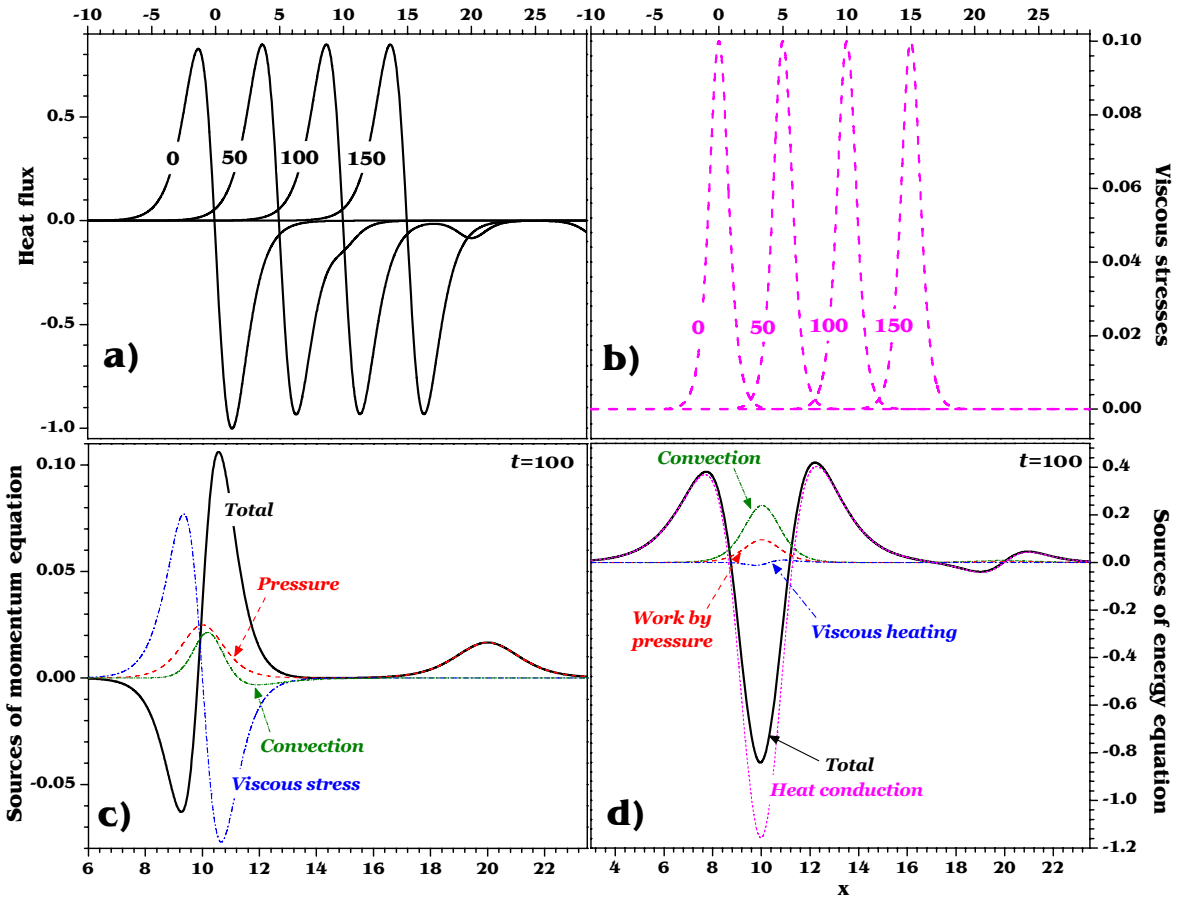

Figure 3. Dynamics of heat flux (a), viscous stress (b), and balance of source terms in momentum (c) and energy (d) equations (manufactured solution).

Convergence in space is demonstrated in Figure 4b. We tested both classical and our "recovery" DG schemes (DG_{0-5}, rDG_{0-3}). To eliminate temporal discretization errors, we ran with very small time steps ($\Delta t = \frac{1}{100}$) and high-order time discretization (CN_2, $ESDIRK_{3,4}$). In convergence plots of Figure 4b, we account for the total number of unknowns (a product of the total number of cells, N_{cells}, and the number of degrees of freedom per cell). All schemes do converge with nearly-theoretical rates*. "Recovery" DG schemes exhibit spectral accuracy – a cubic rDG converges with 11th-order at the asymptotic grid range. Notably, the rDG_3 on the grid with 12 cells is more accurate than the 1st-order finite-volume (DG_0) scheme on grid N=3200.

$$\mathrm{CFL}_{\mathrm{stb}} = \max_{i=1,N_{\mathrm{cells}}} \left(\frac{\Delta x_i}{\Delta t \max\left(|u_i|, |u_i \pm c_i|, \frac{\eta_i}{\rho_i \Delta x_i}, \frac{\beta_i}{\rho_i \Delta x_i} \right)} \right)$$

Similarly, *dynamic CFL number* is defined as $\mathrm{CFL}_{\mathrm{dyn}} = \frac{\Delta x_{\min}}{\Delta t\ \mathrm{U}}$, where $\Delta x_{\min}$ is the size of the smallest cut-cell and U is the fastest dynamic velocity (e.g., $\max(U_{\mathrm{int}}, C_1)$ in the manufactured solution, or shock speed in the Shock Tracking test of Section IV.C).

* Quadratic and quartic DG are shown to be only second and fourth-order convergent, respectively (one order lower of what supposed to be). This is because the error is measured at cell centers, eq.(51), where all odd-order scaled Legendre polynomials are zero.

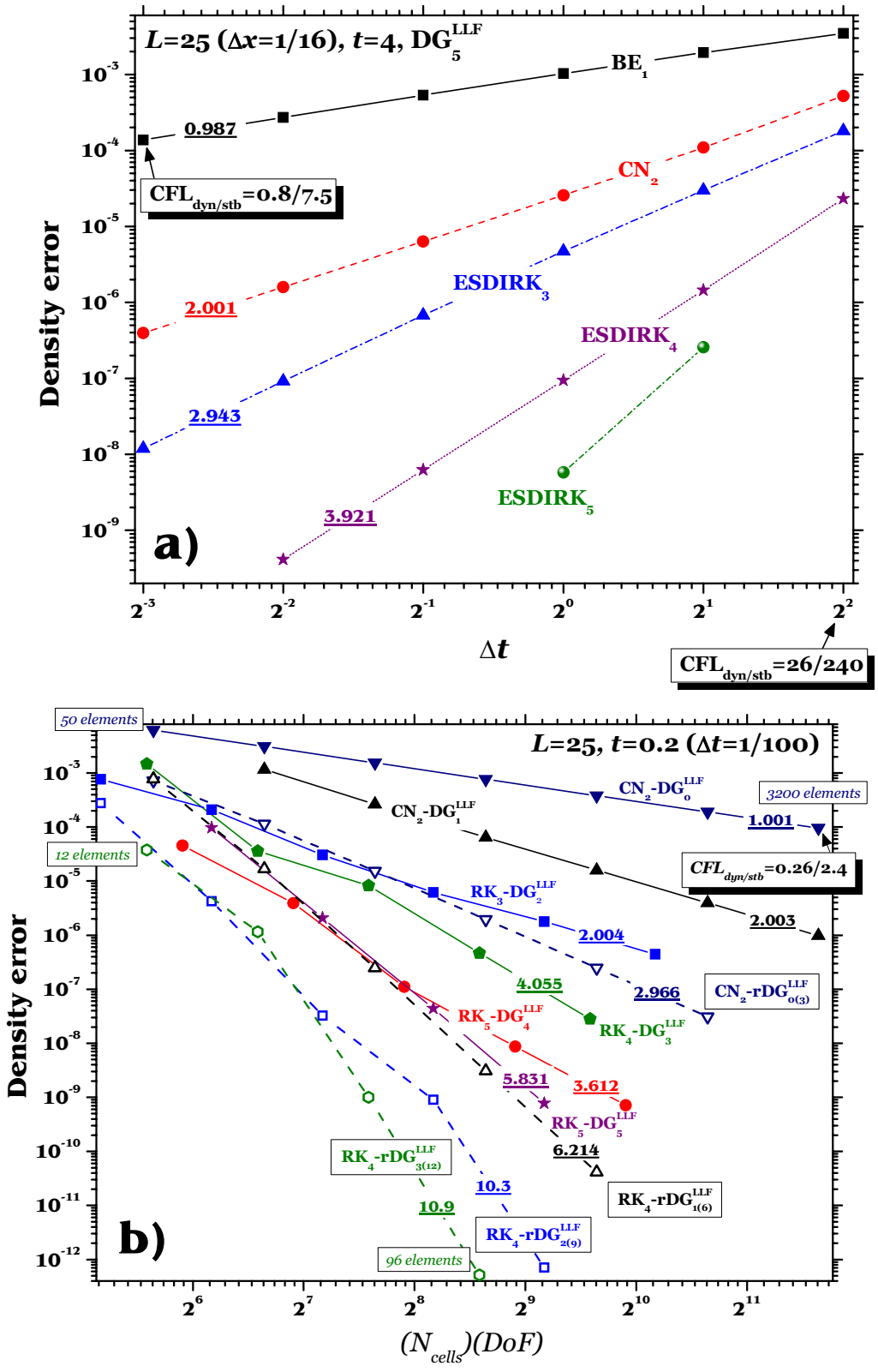

Figure 4. Convergence in time (a) and space (b).

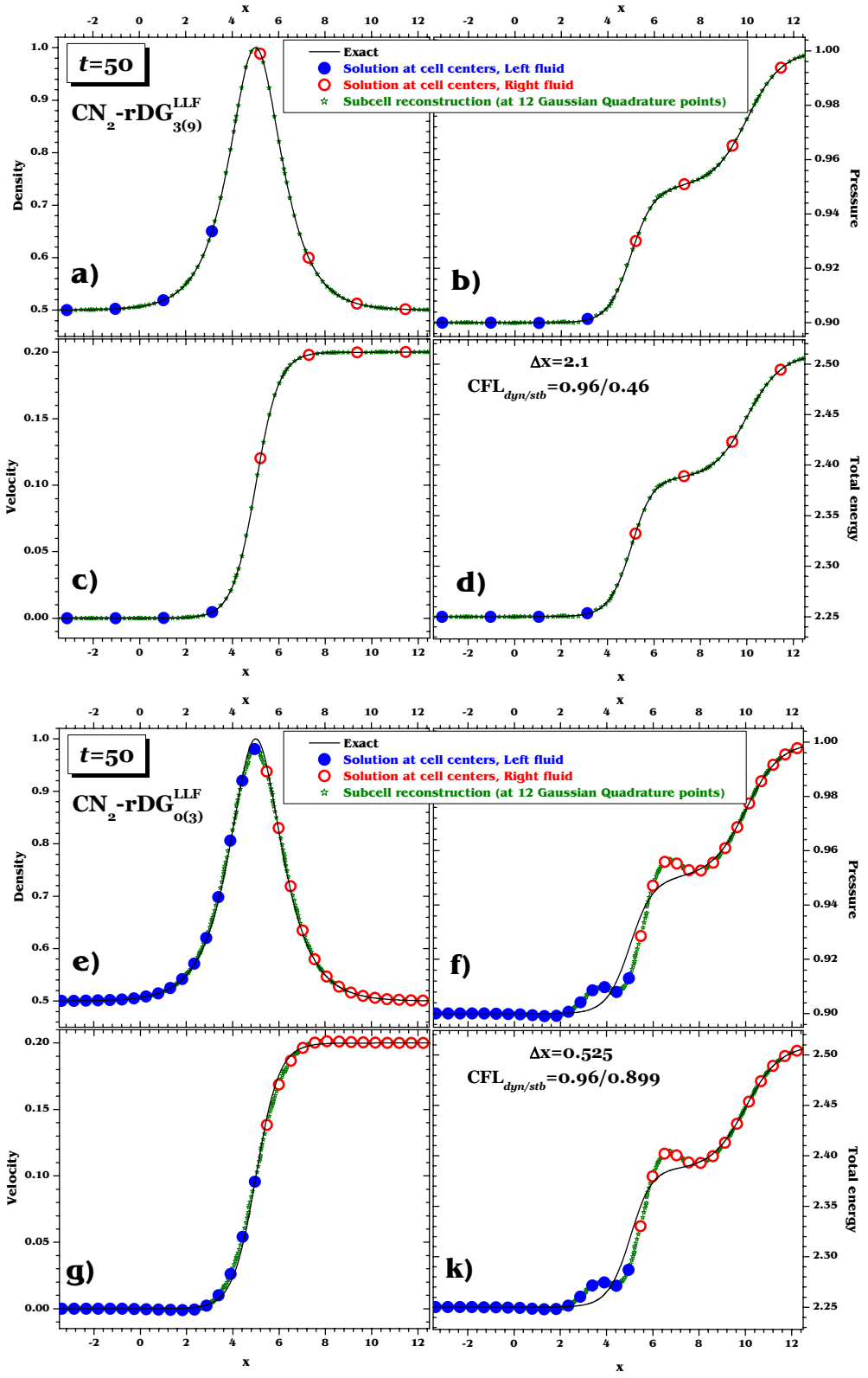

Figure 5. Profiles of density, pressure, velocity and total energy for MS, t=50. Solutions by rDG_3 (a-d) vs. rDG_0 (e-k) with the same number of unknowns (N=48) and dynamic CFL numbers (0.96).

Superb performance of the high-order rDG schemes is demonstrated in Figure 5, where we compared solutions at time t=50 for the cubic rDG_3 (Figure 5a-d) and the piecewise-constant (finite-volume, PPM) rDG_0 (Figure 5e-k). Both solutions are obtained with the same total number of unknowns ($N_{\mathrm{cells}} \cdot \mathrm{DoF} = 48$) and under the same $\mathrm{CFL}_{\mathrm{dyn/stb}}$. We show both the solution at the cell centers (circles) and the reconstructed subcell distributions at 12 Gaussian quadrature points (stars). The rDG_3 solution with 12 cells (Figure 5a-d) is significantly more accurate than the rDG_0 with 48 cells (Figure 5e-k). Inaccuracy of the 3rd-order scheme is clearly pronounced in pressure/total energy profiles, as an overshoot around fluid-fluid interface. With further grid refinement, this overshoot disappears.

B. Physics-Based Preconditioning: Eigenscopy

In this section, we demonstrate efficiency of our physics-based preconditioners, using manufactured solution introduced above. The base input parameters are given in the caption to Figure 2. The results are presented in terms of eigenvalue patterns ("*eigenscopy*", Figure 6 and Figure 7) and the maximum number of Krylov vectors per linear (GMRES) solve, Figure 8; scanning a wide range of Mach numbers and fluid's dynamic viscosities η. Mach numbers are varied by changing U_{int} and C_1 ($U_{int}/C_1 = 2$) and keeping the rest base input parameters unaltered.

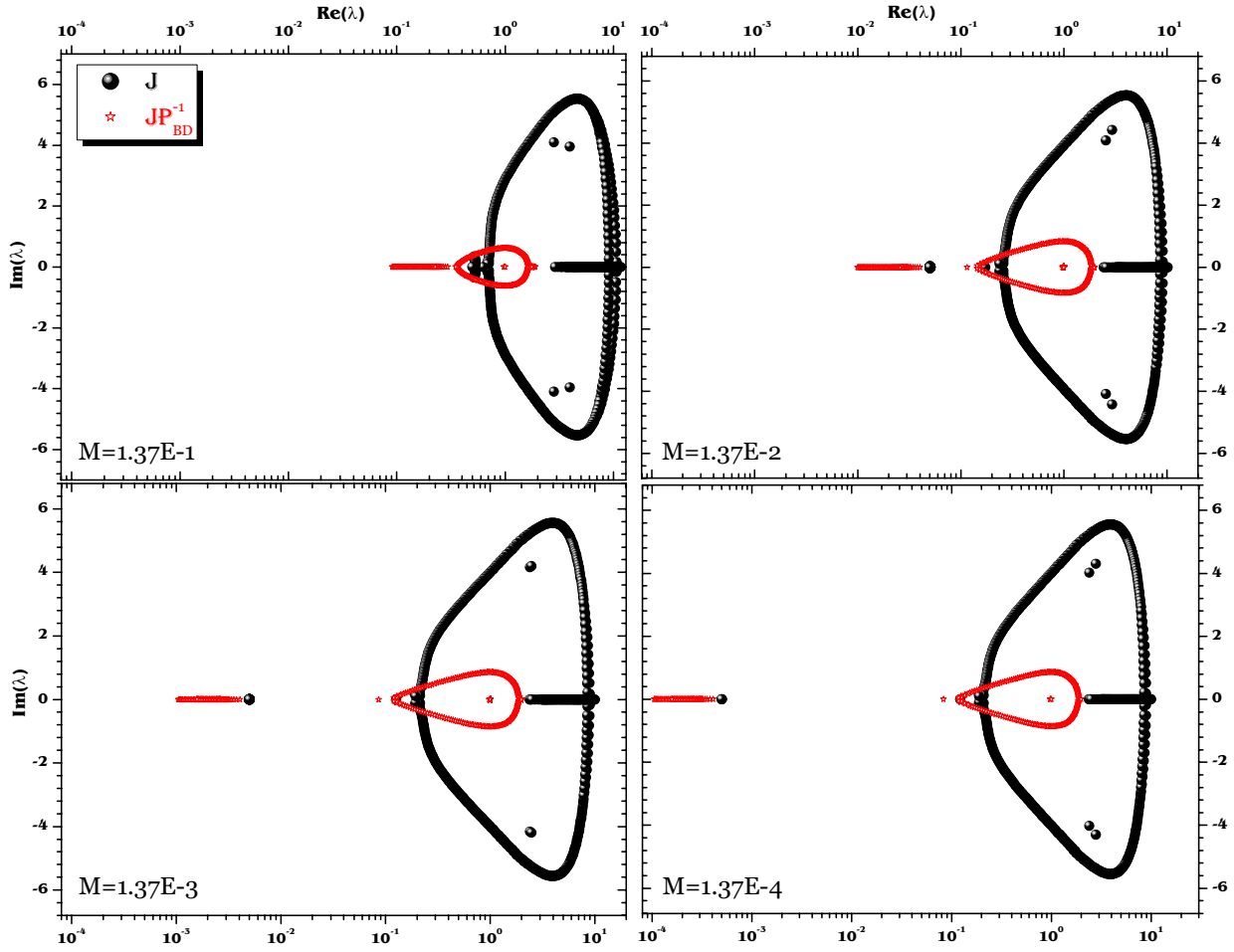

Figure 6. Eigenscopy of the Block-Diagonal (BD) preconditioner. CN$_2$-DG$_1^{\mathrm{LLF}}$, CFL$_{\mathrm{dyn}}$=0.4, $N_{\mathrm{cells}} = 100$.

Eigenvalues of the Jacobian $\mathbb{J}$ and preconditioned Jacobian $\mathbb{J}\mathbb{P}^{-1}$ matrices are computed by first balancing matrices (reducing their Euclidean norms); then transforming to Hessenberg form, and, finally, using QR algorithm to compute eigenvalues[30].

Since the Jacobian matrices are non-symmetric and non-positive-definite, their eigenvalues are complex in general. This precludes using such a popular linear algebra solver as Conjugate-Gradient (CG). More general Krylov subspace iteration methods like GMRES or FOM[33] on the other hand are capable of dealing with these matrices; however, for efficiency purposes, they need preconditioning. The action/purpose of preconditioning is to "cluster" eigenvalues of the "effective" Jacobian matrix $\mathbb{J}\mathbb{P}^{-1}$, so the Krylov (GMRES) method is able to converge in a few (~10) iterations/approximation vectors.

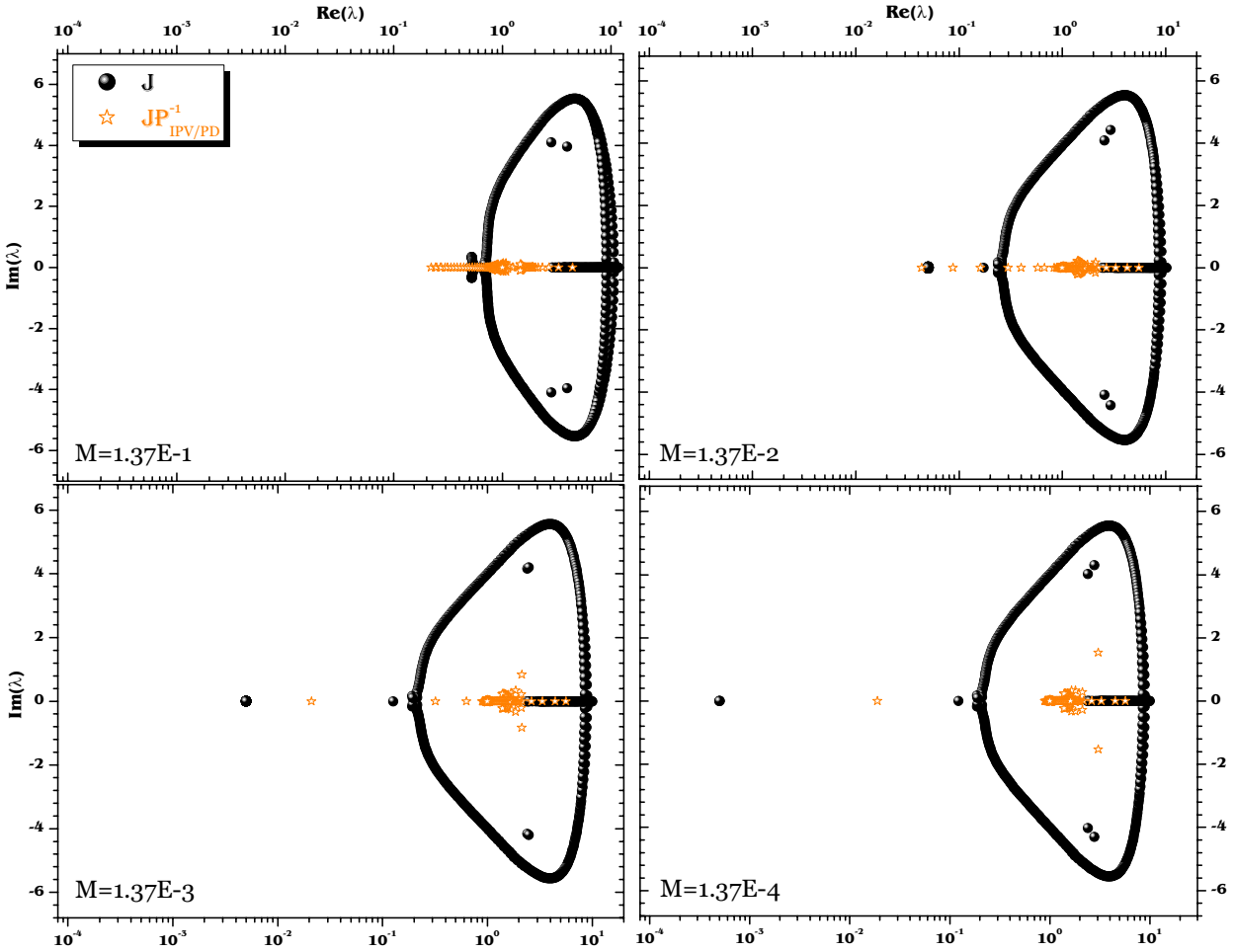

Figure 7. Eigenscopy of the IPV/PD preconditioner.
CN_2-DG_1^{LLF}, CFL_{dyn}=0.4, $N_{cells} = 100$.

In Figure 6, we show the action of the *Block-Diagonal (BD)* preconditioner. This preconditioner implicitly accounts for local coupling effects. Within the DG discretization, some non-local coupling effects are also captured, as the gradients (perturbations/high-order momenta) of the DG solution are part of the local solution vector and represented in the local block inversion. It can be seen from Figure 6, that the eigenvalues $\mathbb{J}\mathbb{P}_{BD}^{-1}$ are less spread, resulting in generally better performance than the unpreconditioned GMRES's $\mathbb{J}$. However, at the limit $M \to 0$, the elliptic components (associated with stiff acoustic waves) become very important. These are captured very well by using *IPV/PD* PBP, Figure 7 and Figure 8. "*Poisson-solve*" step of this PBP seems to "remove" complex

components of eigenvalues (associated with elliptic/pressure waves), clustering them near the Real axis, Figure 7. *Steps I and III* of the IPV/PD are designed to target parabolic/diffusion components. As a result, this PBP seems to be very effective in a wide range of Mach numbers and fluid's viscosities, Figure 8. At the high-speed ($M > 1$) conditions, acoustic and material-velocity time scales are comparable, rendering pressure-Poisson part of the preconditioner ineffective (Figure 8a,b). For these, high-speed conditions, the BD preconditioner seems to become more effective.

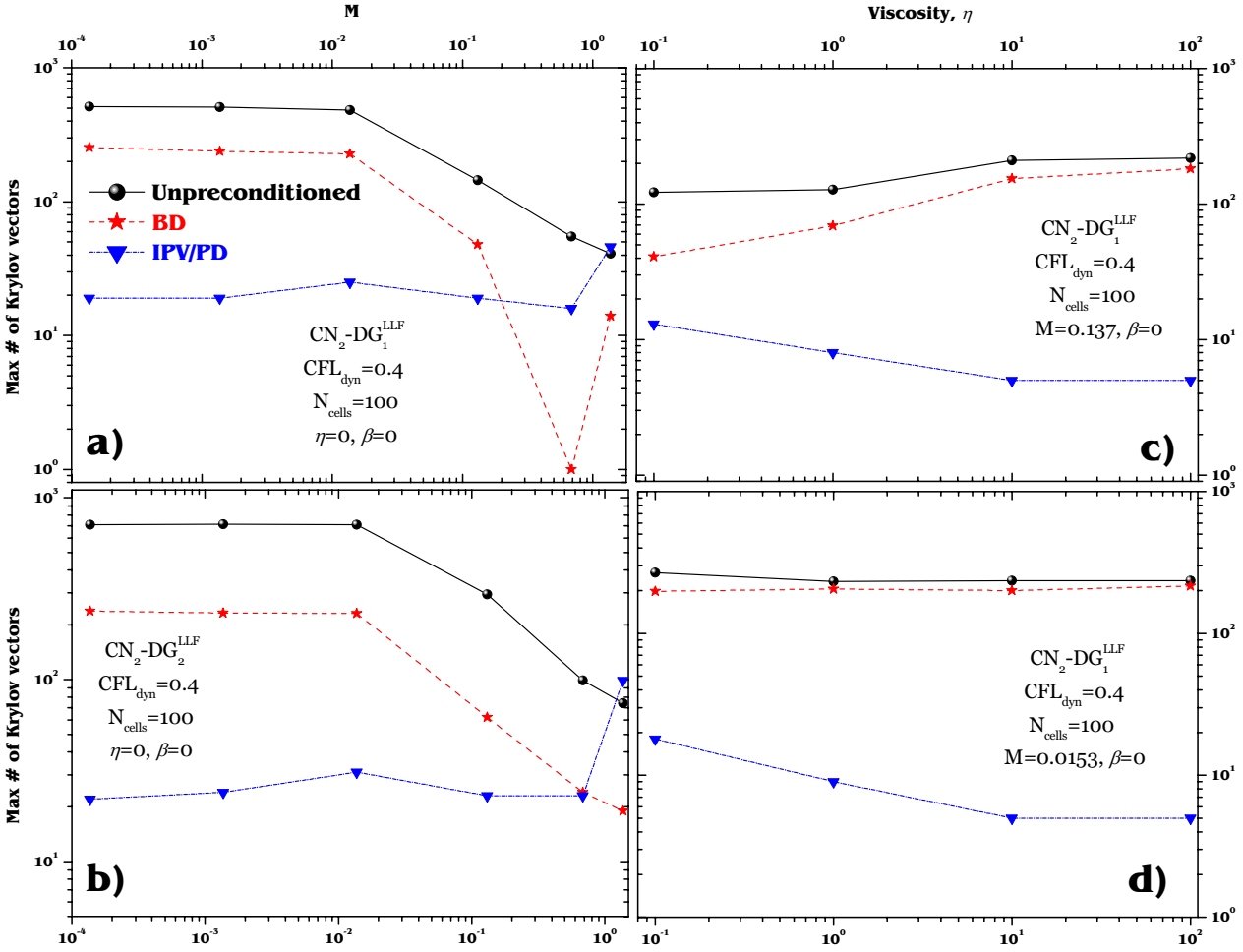

Figure 8. Efficiency (maximum # of Krylov vectors per linear solve) of the BD and IPV/PD physics-based preconditioners in a wide range of Mach numbers (a,b, M = from 10^{-4} to 1) and viscosities (c,d, η = from 10^{-4} to 10^{2}).

C. Shock Tracking

Problem formulation. A shock of strength M_{sh} is initially placed at x=0.25 of the 1-unit-long computational domain. The right boundary of the domain is a solid wall (blunt body) moving with constant speed $U_{b.b}$. Pre- and post-shock conditions are initialized as

$$\begin{array}{|l|lcl|}
\hline
\textit{Pre-Shock:} & P_{\rm pre} & = & 1 \\
 & \rho_{\rm pre} & = & 1 \\
 & u_{\rm pre} & = & \mathrm{U_{b.b}} \\
\hline
\textit{Post-Shock:} & P_{\rm post} & = & P_{\rm pre}\left[1+\frac{2\gamma}{\gamma+1}\left(\mathrm{M_{sh}^2}-1\right)\right] \\
 & \rho_{\rm post} & = & \rho_{\rm pre}\dfrac{1+\frac{\gamma+1}{\gamma-1}\frac{P_{\rm post}}{P_{\rm pre}}}{\frac{\gamma+1}{\gamma-1}+\frac{P_{\rm post}}{P_{\rm pre}}} \\
 & u_{\rm post} & = & W_{\rm sh}\left(1-\frac{\rho_{\rm pre}}{\rho_{\rm post}}\right) \\
\hline
\end{array} \tag{52}$$

where $\gamma = 1.4$, shock Mach number M_{sh} is specified and the shock speed W_{sh} is computed as

$$W_{\rm sh} = c_{\rm pre}\sqrt{1+\frac{\gamma+1}{2\gamma}\left(\frac{P_{\rm post}}{P_{\rm pre}}-1\right)} \tag{53}$$

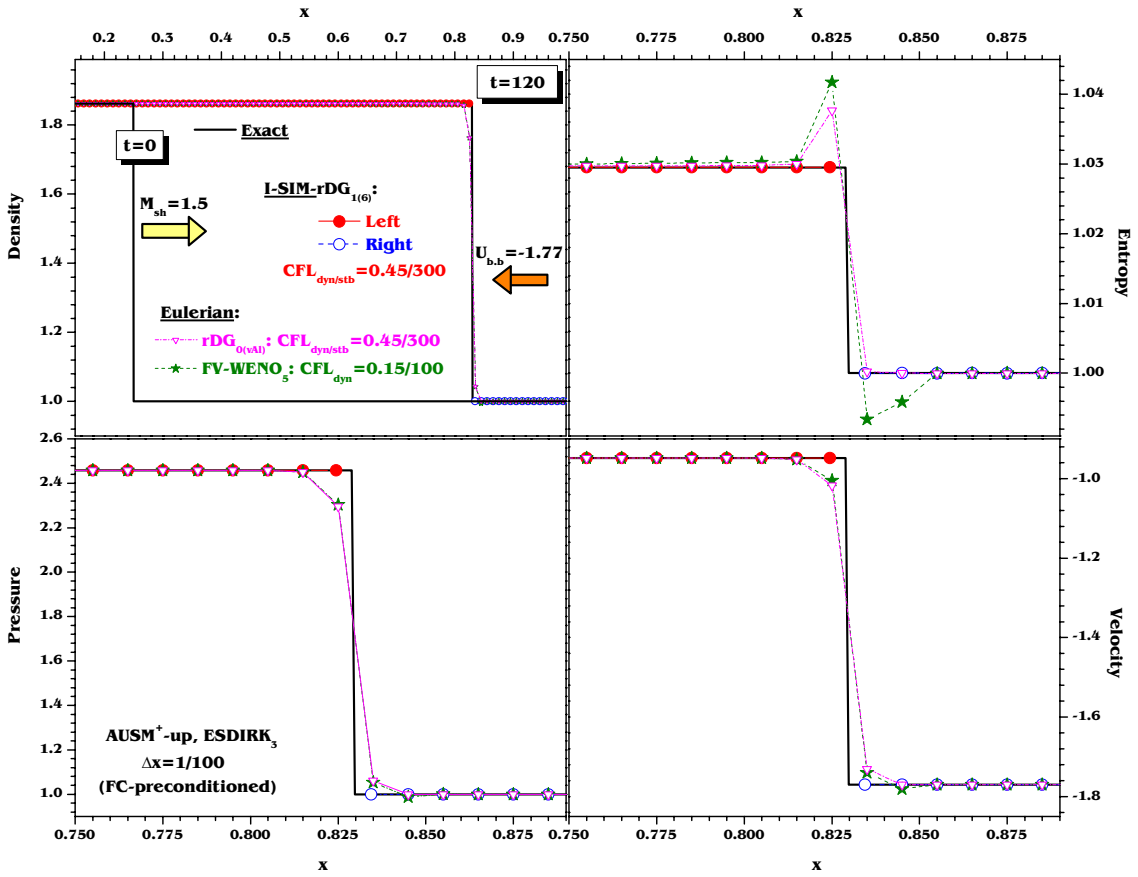

Figure 9. Comparison of the I-SIM and Eulerian methods for a weak M_{sh}=1.5 shock wave problem.

The dynamic time scale for this problem is $\tau_{\rm dyn} = \frac{W_{\rm sh}+{\rm U_{b.b}}}{\Delta x}$, which can be varied by changing the speed of the blunt body $U_{b.b}$. Here, we are interested at the regimes when the dynamic time is significantly larger than the CFL-stability time scales, $\tau_{\rm dyn} \gg \Delta t_{\rm stb}$, which makes this problem particularly challenging (expensive) for explicit methods. The JFNK (preconditioned by the conservative-variable-FC, with maximum number of Krylov vectors maintained below 10) allows for very efficient solution with time steps resolving the dynamic time scales (CFL_{dyn}=$0.1 \div 0.5$) and significantly exceeding those from the stability limit of explicit schemes.

In I-SIM simulations, interfacial markers are moved with shock speed, reconstructed from the exact Riemann solver applied at the interfacial edge between two adjacent cut-cells.

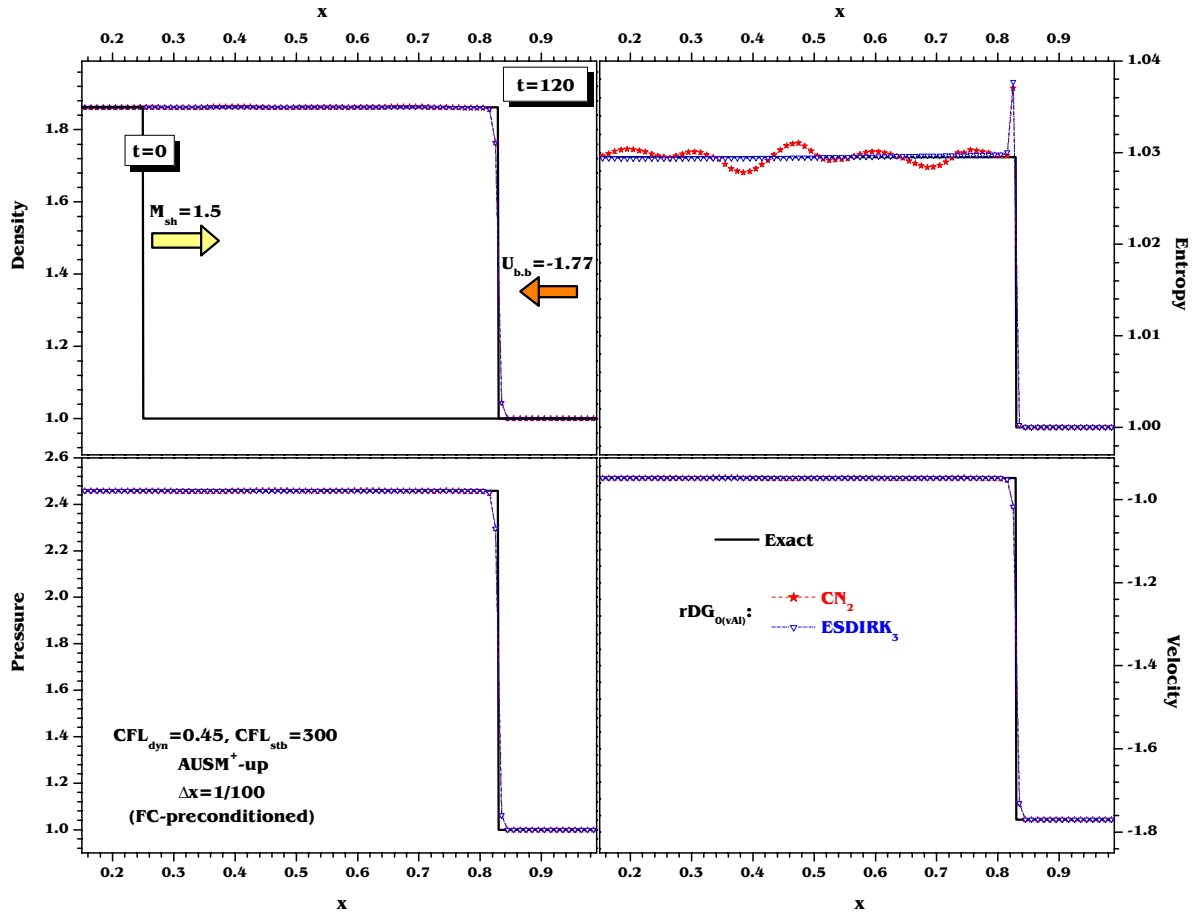

Figure 10. Comparison of the Crank-Nicholson (CN_2) and implicit Runge-Kutta ($ESDIRK_3$) schemes (with Eulerian spatial discretization) for a weak M_{sh}=1.5 shock wave problem.

Results and discussion. Computational results for weak (M_{sh}=1.5) and strong (M_{sh}=10) shock waves are presented in Figure 9, Figure 10 and Figure 11. All simulations are performed on the grid $\Delta x = \frac{1}{100}$ and using AUSM$^+$-up flux treatment[25]. In Figure 9, the I-SIM method is compared with two Eulerian schemes, i.e. the third-order $rDG_{o(vAl)}$ and the fifth-order finite-volume $WENO_5$, for a weak shock wave. All three methods capture shock position very accurately. The Eulerian schemes smear the shock over 2-3 nodes, which results in distinct over-/undershoots of entropy next to the shock. The I-SIM is free of these over-/undershoots, and all variables (density, pressure, velocity and entropy) are sharp/discontinuous at the shock. Both the Eulerian third-order $rDG_{o(vAl)}$ and the I-SIM-$rDG_{i(6)}$ schemes are robust under $CFL_{stb} \sim 300$, corresponding to dynamic-time $CFL_{\rm dyn} = 0.45$. The Eulerian FV-$WENO_5$ scheme requires lower dynamic CFL (~ 100), to make the Newton method converge. We believe this is because the $WENO_5$ is "essentially non-oscillatory" (and TVB), introducing additional time scales

associated with small/bounded oscillations emanating from the shock, which effectively makes the "ball of convergence" for Newton's method smaller.

Performances of different temporal discretizations (second-order Crank-Nicholson vs. the third-order ESDIRK$_3$) are compared in Figure 10. Even though the solutions for density, pressure and velocity are nearly identical, the entropy solution of CN_2 is oscillatory, which is most probably associated with the fact that CN_2 is not L-stable. The ESDIRK schemes on the other hand are L-stable, which result in more "clean" solutions for entropy. Notably, oscillations in entropy of CN_2 scheme might introduce additional unphysical time scales, which will make the "ball of convergence" for Newton's method smaller, and, therefore, requires smaller time steps. We found this especially notable when the CN_2 is combined with the FV-WENO$_5$.

Figure 11 demonstrates performances of the I-SIM and Eulerian methods for a strong shock wave. In this case, the Eulerian schemes require rather small dynamic CFL numbers (~ 0.016) in order for the Newton's method be convergent. This corresponds to CFL$_{stb}$=15, which is still an order of magnitude more efficient than what would be required from explicit schemes. Notably, the I-SIM is able to completely eliminate Eulerian method's "artificial acoustics" emanating from the shock wave, enabling thus very efficient simulation with $\mathrm{CFL}_{\mathrm{dyn}} = 0.21$ and $\mathrm{CFL}_{\mathrm{stb}} = 200$.

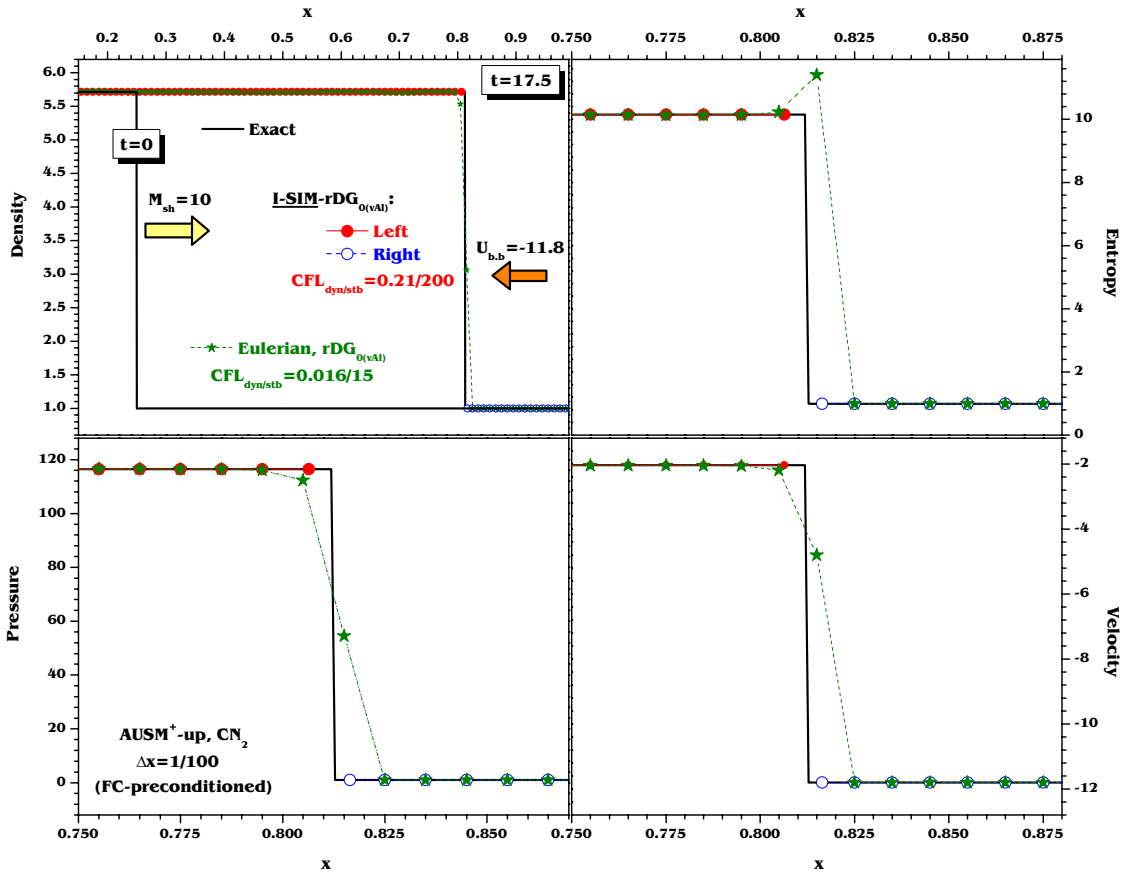

Figure 11. Comparison of I-SIM and Eulerian methods for a strong M_{sh}=10 shock wave problem.

D. Interface tracking

In this section, we demonstrate performance of our I-SIM method for multimaterial interface tracking. Even though most problems considered here are easily/robustly treated with explicit algorithms[22] (since the dynamic times are comparable to those due to stability limit), the benefits of using our I-SIM algorithm are its exceptional robustness

for strong shocks and high-acoustic-impedance interfaces, and, more importantly, *our algorithm is fully conservative even at the interface*, in difference to most previous interface tracking methods for compressible multifluid dynamics[1,2,3,14,16,17,22,34]. All problems are solved without preconditioning of GMRES, as $\tau_{\rm dyn} \approx \Delta t_{\rm stb}$. Interfacial markers are moved with material velocity obtained from the exact Riemann solver applied at $x_{\rm m}$ (i.e., at the interfacial edge between two adjacent cut-cells).

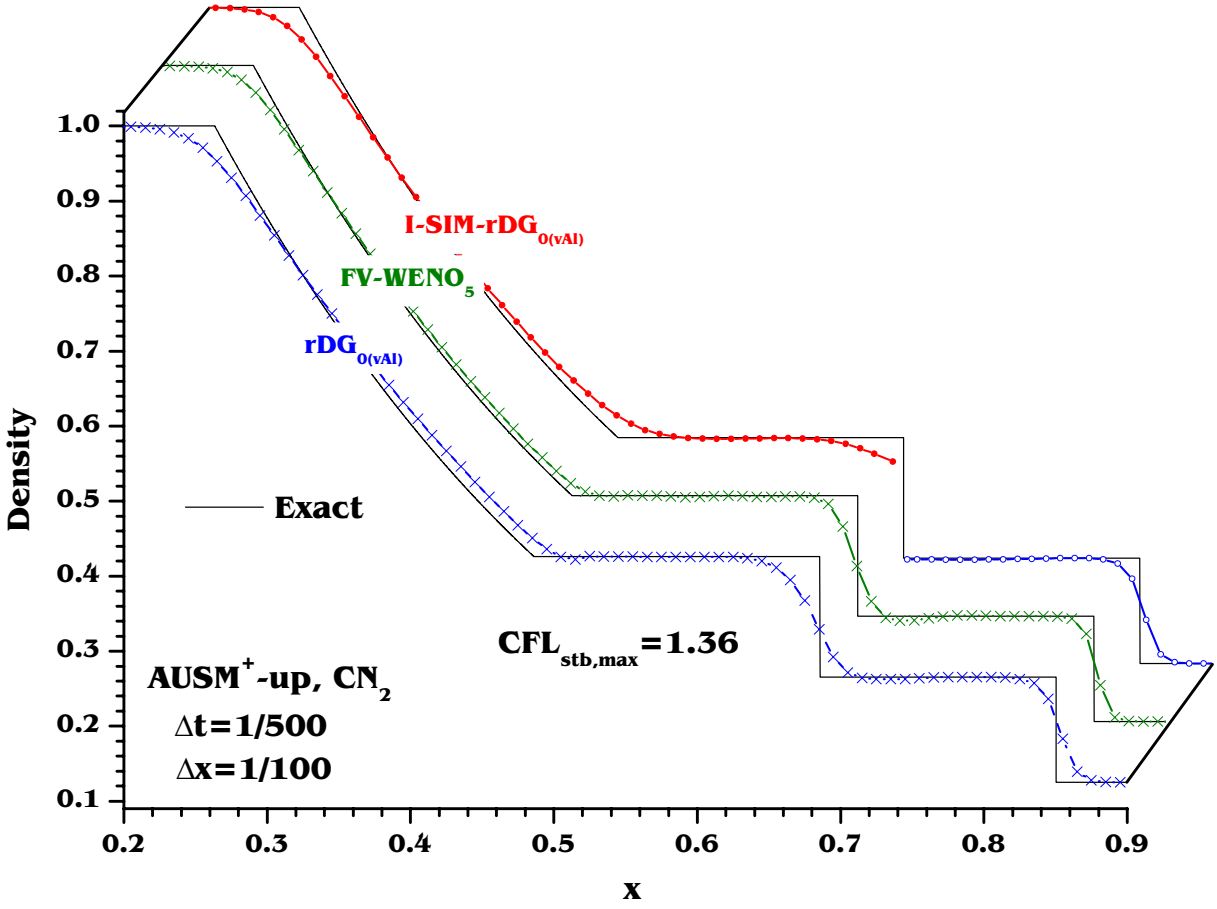

Figure 12. Sod test: comparison of density profiles for different discretization schemes.

Sod test. This is a soft shock-tube problem initiated by placing discontinuity

$$\begin{aligned} (\rho, P, u, \gamma)_{\rm L} &= (1, 1, 0, 1.4) \\ (\rho, P, u, \gamma)_{\rm R} &= (0.125, 0.1, 0, 1.4) \end{aligned} \tag{54}$$

in the single-phase γ-gas, at x=0.5 of the 1-unit-long computational domain.

Computational results are shown in Figure 12 and Figure 13. Performances of all schemes (both Eulerian and I-SIM) are comparable at the rarefaction and shock waves. The major differences are near the contact. The I-SIM captures the jump in density sharply, within a cell, while the Eulerian schemes smear the discontinuity over 5-8 nodes. There are small over-heatings in the post-contact solution with I-SIM method, which are associated with under-resolution of the wave structure at the very beginning of the simulation, when all waves (rarefaction, contact and shock) are within one cell, and I-SIM is unable to single-out the contact sharply.

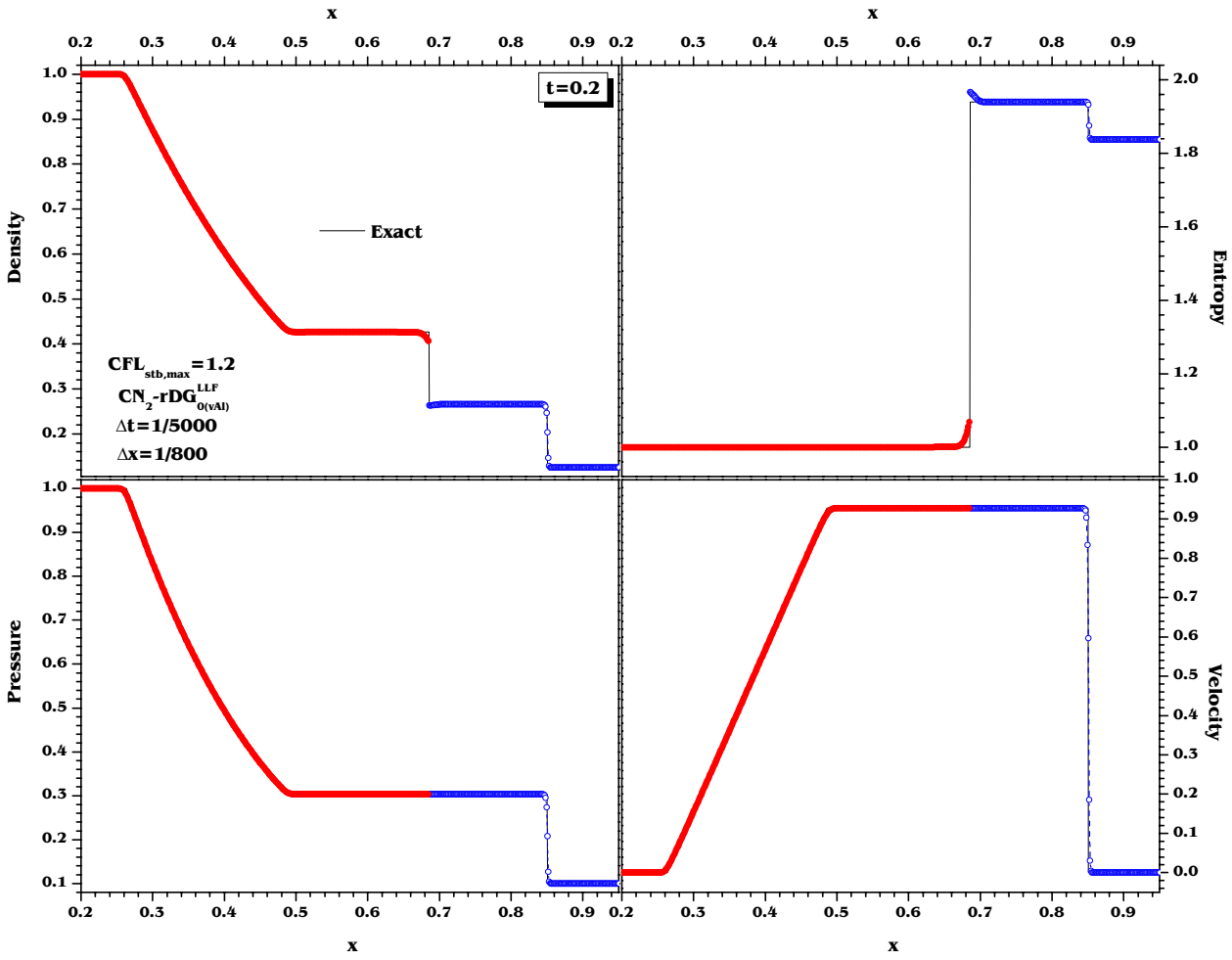

Figure 13. Sod test: density, pressure, velocity and entropy of I-SIM compared to analytical solution.

Multifluid Sod test. Initial conditions of the previous test problem are altered to

$$\begin{aligned} (\rho, P, u, \gamma)_{\mathrm{L}} &= (1, 1, 0, 1.4) \\ (\rho, P, u, \gamma)_{\mathrm{R}} &= (0.125, 0.1, 0, 1.2) \end{aligned} \tag{55}$$

For this problem, the Eulerian methods exhibit unphysical pressure oscillations at the multi-material contact[1,2,16]. A number of recipes have been developed to fix this problem[2,3,14,16,17], all of them leading to the loss of conservation near the interface. As one can see from our results in Figure 14, our I-SIM method does not suffer from pressure/velocity oscillations and, at the same time, it is *conservative to machine accuracy*, even near the contact.

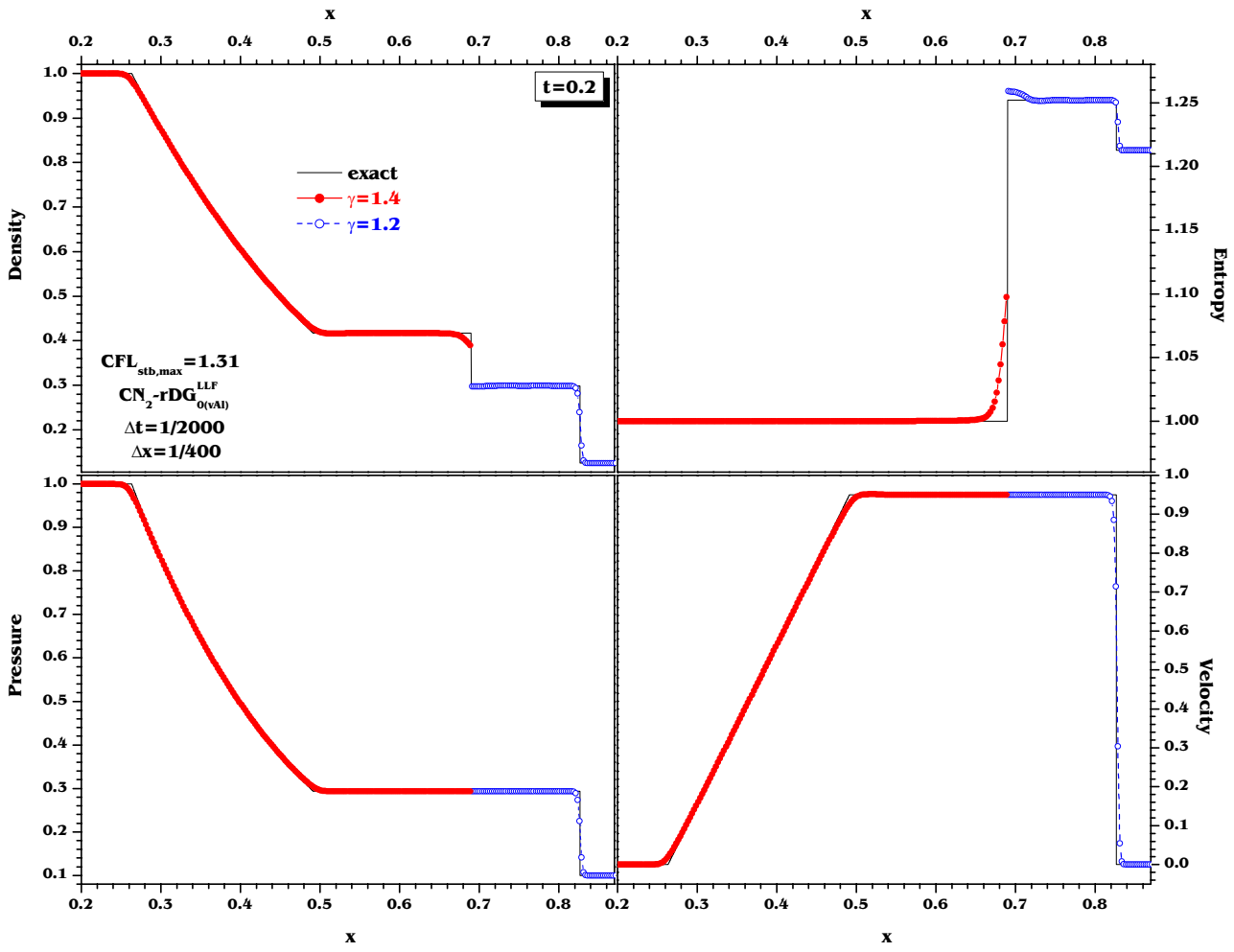

Figure 14. Multifluid Sod test: density, pressure, velocity and entropy of I-SIM compared to analytical solution.

Stiff Gas-Gas Shock-Tube Problem. Our next numerical test was first introduced by Abgrall and Karni[3]. The problem is set up by placing the following discontinuity

$$\begin{aligned} (\rho, P, u, \gamma)_{\mathrm{L}} &= (1, 500, 0, 1.6) \\ (\rho, P, u, \gamma)_{\mathrm{R}} &= (1, 0.2, 0, 1.4) \end{aligned} \tag{56}$$

at x=0.5 of the 1-unit-long computational domain. This results in very strong M_{sh}=31 shock wave transmitted to the right fluid, which is very closely followed by the multi-material contact. The results of our I-SIM method are compared with the analytical solution in Figure 15. Importantly, shock position is captured very accurately. The pair "contact-shock" waves are under-resolved for a quite significant time due to relatively small difference between the post-shock material velocity and shock speed. Nevertheless, only a minor under-heating is formed near the contact.

It is instructive to note that in order for the Newton's method be convergent, we had to use tight linear tolerances even at the beginning of the Newton iterations. This is most probably associated with the elevated sensitivity of the non-linear solvers under stronger (stiff) shock conditions.

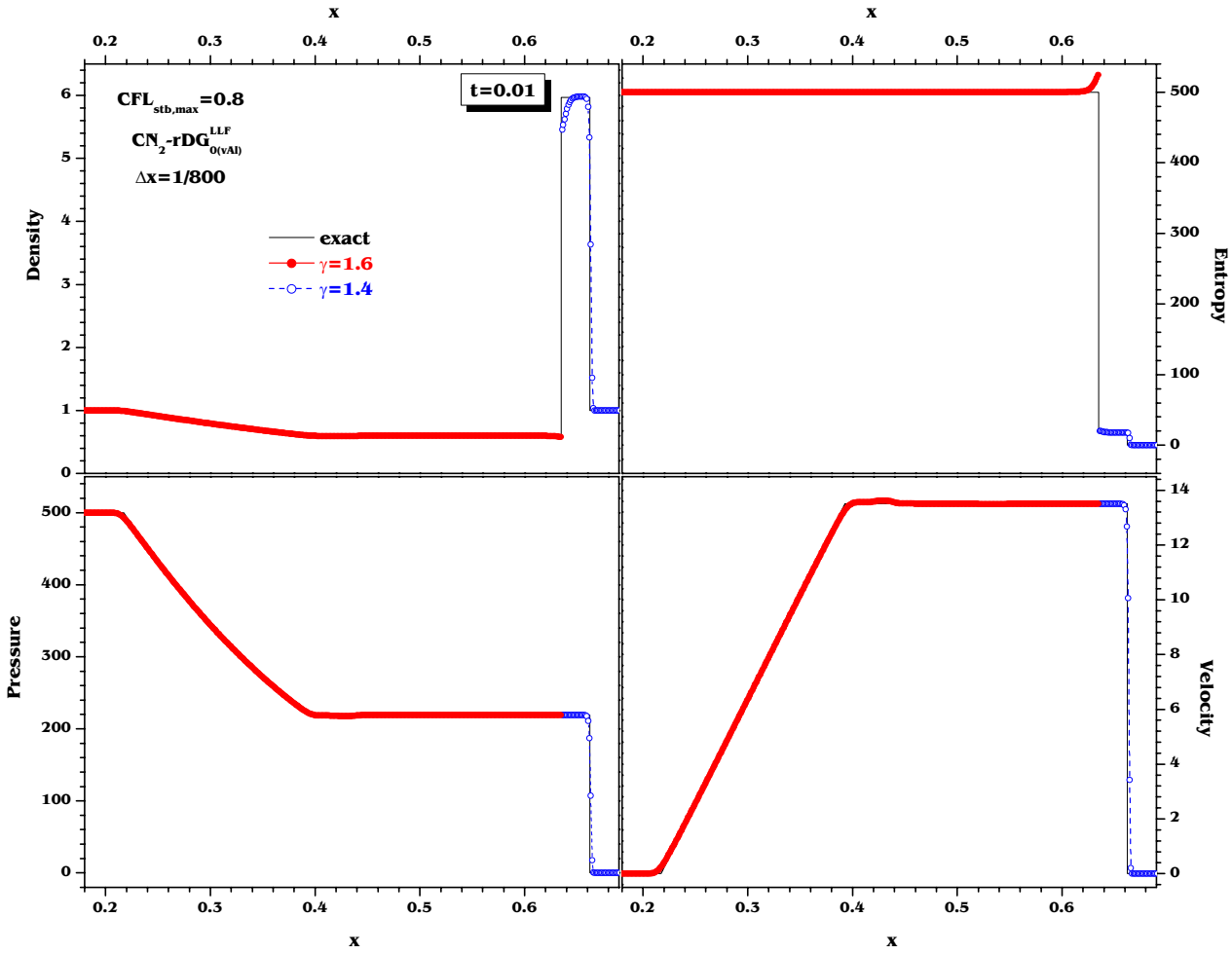

Figure 15. Stiff gas-gas shock-tube: density, pressure, velocity and entropy of I-SIM compared to analytical solution.

Stiff Liquid-Gas Interface Problem. Our last shock-tube problem was originally proposed by Saurel and Abgrall[34]. Initially, liquid-gas interface is placed at x=0.7m of the 1-m-long computational domain. The left and right states are

$$\begin{aligned} (\rho, P, u, \gamma, \Pi)_{\mathrm{L}} &= (1000, 10^9, 0, 4.4, 6 \cdot 10^8) \\ (\rho, P, u, \gamma, \Pi)_{\mathrm{R}} &= (50, 10^5, 0, 1.4, 0) \end{aligned} \tag{57}$$

Computational results for t=0.24 ms are compared to the analytical solution in Figure 16. Both contact and transmitted shock positions are predicted accurately. No pressure oscillation is seen at the liquid-gas interface. The solution is conservative (to machine accuracy). Insignificant overheating is produced at the contact, which is tracked sharply, within one cell.

Due to extreme conditions of this test, once-in-a-while Newton's method has difficulty converging to the specified tolerance of 10^{-16}, because of a limit cycle in its iterations. In these cases, we allow the solution to proceed to the next time step if the maximum permitted number of Newton iterations is exceeded and the maximum $\mathcal{L}_2$ error at the cycle is less than 10^{-2}. This has no impact on conservation and little impact on accuracy of the method, as one can clearly see from Figure 16.

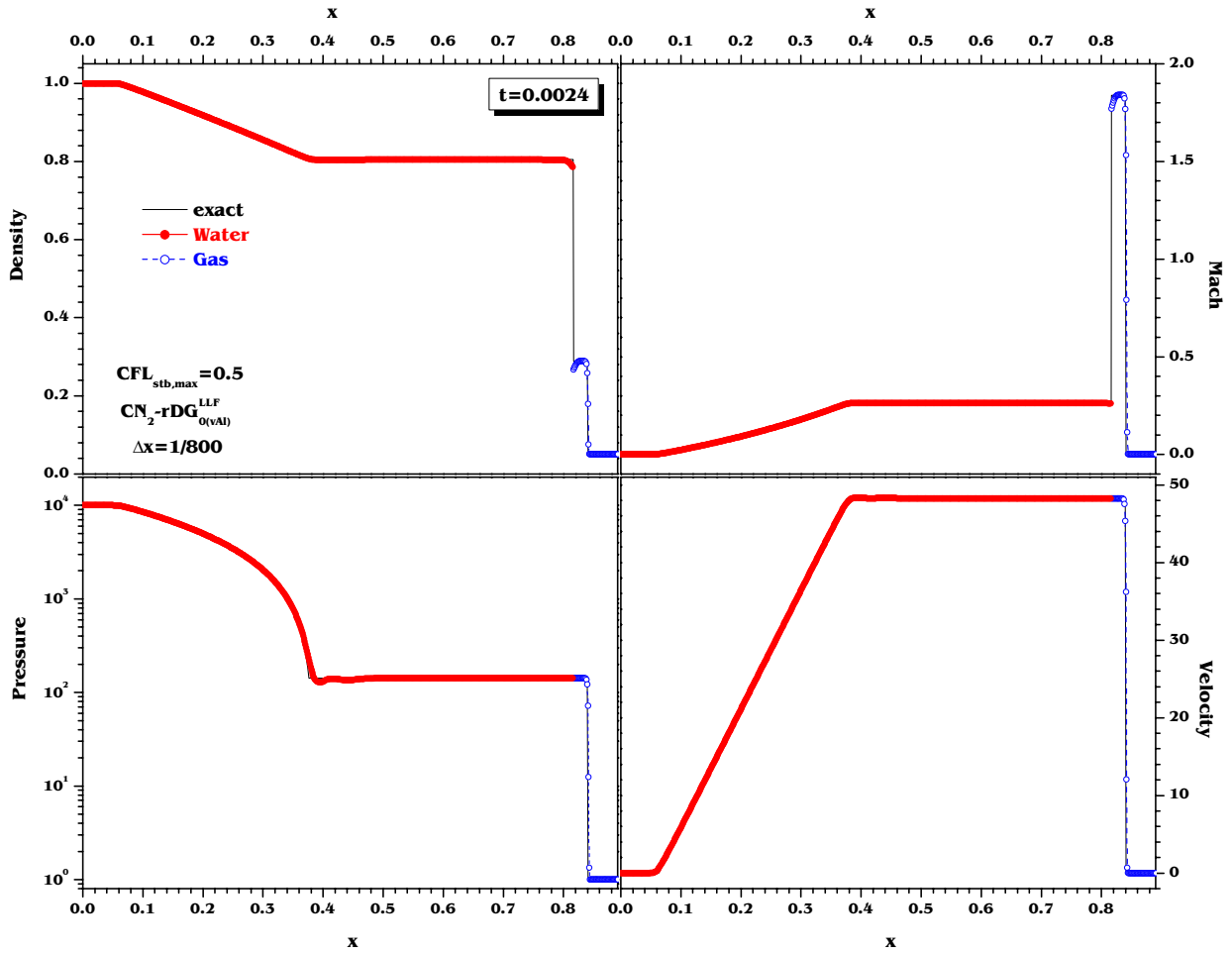

Figure 16. Stiff liquid-gas shock-tube: density, pressure, Mach and velocity of I-SIM compared to analytical solution. Solution is shown in dimensionless units, scaled with $P_0 = 10^5$, $\rho_0 = 10^3$ and *L*=1.

V. Summary

Implicit Sharp-Interface Method (I-SIM) is introduced. The method is based on Jacobian-free Newton-Krylov technology, combining interface tracking, high-order temporal (ESDIRK) and spatial (cut-cell+"recovery" Discontinuous Galerkin) discretizations, with all relevant physics (interface dynamics, convection, diffusion and chemical reaction/sources in fluids) fully-(non-linearly)-coupled, thereby avoiding operator-splitting time discretization errors. Performance (accuracy, convergence, robustness and efficiency) of our method is demonstrated to be accurate and efficient on a number of one-dimensional problems: low-speed manufactured solutions with viscous and heat conduction operators; slow-dynamic-time-scale shock tracking and multi-material interfaces under high-speed (shock-tube) flow conditions. We demonstrated the all-speed and all-fluid-property capabilities of our algorithm, including potentials for a wide range of flow speeds (from very-low-Mach-number to supersonic/large-M_{sh}-number), fluid viscosities (from inviscid/Euler-formulation to very viscous/$Re \to 0$), fluid conductivities and density ratios (gas-gas and gas-liquid interfaces). Two physics-based preconditioners (BD and IPV/PD) of the Krylov (GMRES) method are introduced, their efficiencies are analyzed in terms of "Eigenscopy" of the Jacobian matrices; and the efficacy is demonstrated at the limits $M \to 0$ and $Re \to 0$. Future development/challenges would require the extension/demonstration to/of multi-D cut-cell meshing and the development of SAMR/multigrid algorithms for physics-based preconditioners implemented along the lines of the introduced here IPV/PD preconditioner.

Appendix A: Recovered Degrees of Freedom for $rDG_{0\text{-}3}$

Piecewise-constant, $rDG_{0(3)}$, R=2:

$$\begin{bmatrix} \tilde{u}_j^{(1)} \\ \tilde{u}_j^{(2)} \end{bmatrix} = \begin{bmatrix} \frac{u_{j+1}^{(0)} - u_{j-1}^{(0)}}{4} \\ \frac{u_{j+1}^{(0)} - 2u_j^{(0)} + u_{j-1}^{(0)}}{12} \end{bmatrix} \tag{58}$$

Piecewise-linear, $rDG_{1(6)}$, R=5:

$$\begin{bmatrix} \tilde{u}_j^{(2)} \\ \tilde{u}_j^{(3)} \\ \tilde{u}_j^{(4)} \\ \tilde{u}_j^{(5)} \end{bmatrix} = \begin{bmatrix} \frac{73\left(u_{j+1}^{(0)} + u_{j-1}^{(0)}\right) - 45\left(u_{j+1}^{(1)} - u_{j-1}^{(1)}\right) - 146u_j^{(0)}}{336} \\ \frac{357\left(u_{j+1}^{(0)} - u_{j-1}^{(0)}\right) - 197\left(u_{j+1}^{(1)} + u_{j-1}^{(1)}\right) - 1034u_j^{(1)}}{3888} \\ \frac{2u_j^{(0)} - u_{j-1}^{(0)} - u_{j+1}^{(0)} - u_{j-1}^{(1)} + u_{j+1}^{(1)}}{112} \\ \frac{15\left(u_{j-1}^{(0)} - u_{j+1}^{(0)}\right) + 11\left(u_{j-1}^{(1)} + u_{j+1}^{(1)}\right) + 38u_j^{(1)}}{3888} \end{bmatrix} \tag{59}$$

Piecewise-quadratic, $rDG_{2(9)}$, R=8:

$$\begin{bmatrix} \tilde{u}_j^{(3)} \\ \tilde{u}_j^{(4)} \\ \tilde{u}_j^{(5)} \\ \tilde{u}_j^{(6)} \\ \tilde{u}_j^{(7)} \\ \tilde{u}_j^{(8)} \end{bmatrix} = \begin{bmatrix} \frac{-28082u_j^{(1)} + 11436(u_{j+1}^{(0)} - u_{j-1}^{(0)}) - 8831(u_{j-1}^{(1)} + u_{j+1}^{(1)}) + 4185(u_{j+1}^{(2)} - u_{j-1}^{(2)})}{57024} \\ \frac{-232480u_j^{(0)} - 452074u_j^{(2)} + 116240(u_{j-1}^{(0)} + u_{j+1}^{(0)}) + 84505(u_{j-1}^{(1)} - u_{j+1}^{(1)}) + 35627(u_{j-1}^{(2)} + u_{j+1}^{(2)})}{1235520} \\ \frac{748u_j^{(1)} + 345(u_{j-1}^{(0)} - u_{j+1}^{(0)}) + 316(u_{j-1}^{(1)} + u_{j+1}^{(1)}) + 189(u_{j-1}^{(2)} - u_{j+1}^{(2)})}{16848} \\ \frac{4030u_j^{(0)} + 6238u_j^{(2)} - 2015(u_{j-1}^{(0)} + u_{j+1}^{(0)}) + 1630(u_{j+1}^{(1)} - u_{j-1}^{(1)}) - 809(u_{j-1}^{(2)} + u_{j+1}^{(2)})}{213840} \\ \frac{-250u_j^{(1)} + 120(u_{j+1}^{(0)} - u_{j-1}^{(0)}) - 115(u_{j-1}^{(1)} + u_{j+1}^{(1)}) + 81(u_{j+1}^{(2)} - u_{j-1}^{(2)})}{247104} \\ \frac{-440u_j^{(0)} - 626u_j^{(2)} + 220(u_{j-1}^{(0)} + u_{j+1}^{(0)}) + 185(u_{j-1}^{(1)} - u_{j+1}^{(1)}) + 103(u_{j-1}^{(2)} + u_{j+1}^{(2)})}{1010880} \end{bmatrix} \tag{60}$$

Piecewise-cubic, rDG$_{3(12)}$, R=11:

$$\begin{bmatrix} \tilde{u}_j^{(4)} \\ \tilde{u}_j^{(5)} \\ \tilde{u}_j^{(6)} \\ \tilde{u}_j^{(7)} \\ \tilde{u}_j^{(8)} \\ \tilde{u}_j^{(9)} \\ \tilde{u}_j^{(10)} \\ \tilde{u}_j^{(11)} \end{bmatrix} = \begin{bmatrix} \left(\begin{array}{c} -17008750u_j^{(0)} - 25538674u_j^{(2)} + 8504375(u_{j-1}^{(0)} + u_{j+1}^{(0)}) + 7153735(u_{j-1}^{(1)} - \\ -u_{j+1}^{(1)}) + 4665497(u_{j-1}^{(2)} + u_{j+1}^{(2)}) + 1806705(u_{j-1}^{(3)} - u_{j+1}^{(3)}) \end{array} \right) /44478720 \\ \left(\begin{array}{c} -196469378u_j^{(1)} - 403694182u_j^{(3)} + 82503365(u_{j+1}^{(0)} - u_{j-1}^{(0)}) - 66772041\times \\ (u_{j-1}^{(1)} + u_{j+1}^{(1)}) + 40397287(u_{j+1}^{(2)} - u_{j-1}^{(2)}) - 14038579(u_{j-1}^{(3)} + u_{j+1}^{(3)}) \end{array} \right) /866121984 \\ \left(\begin{array}{c} 1293050u_j^{(0)} + 1560302u_j^{(2)} - 646525(u_{j-1}^{(0)} + u_{j+1}^{(0)}) + 589265(u_{j+1}^{(1)} - u_{j-1}^{(1)}) - \\ -436591(u_{j-1}^{(2)} + u_{j+1}^{(2)}) + 193995(u_{j+1}^{(3)} - u_{j-1}^{(3)}) \end{array} \right) /19388160 \\ \left(\begin{array}{c} 201983810u_j^{(1)} + 320885230u_j^{(3)} + 88115825(u_{j-1}^{(0)} - u_{j+1}^{(0)}) + \\ +75239745(u_{j-1}^{(1)} + u_{j+1}^{(1)}) + 49833007(u_{j-1}^{(2)} - u_{j+1}^{(2)}) + 19243255(u_{j-1}^{(3)} + u_{j+1}^{(3)}) \end{array} \right) /5485439232 \\ \left(\begin{array}{c} -90550u_j^{(0)} - 101002u_j^{(2)} + 45275(u_{j-1}^{(0)} + u_{j+1}^{(0)}) + 42475(u_{j-1}^{(1)} - u_{j+1}^{(1)}) + \\ +33701(u_{j-1}^{(2)} + u_{j+1}^{(2)}) + 16605(u_{j-1}^{(3)} - u_{j+1}^{(3)}) \end{array} \right) /25608960 \\ \left(\begin{array}{c} -123718u_j^{(1)} - 178034u_j^{(3)} + 54943(u_{j+1}^{(0)} - u_{j-1}^{(0)}) - 48027(u_{j-1}^{(1)} + u_{j+1}^{(1)}) + \\ +33509(u_{j+1}^{(2)} - u_{j-1}^{(2)}) - 14009(u_{j-1}^{(3)} + u_{j+1}^{(3)}) \end{array} \right) /66624768 \\ \left(\begin{array}{c} 1162u_j^{(0)} + 1246u_j^{(2)} - 581(u_{j-1}^{(0)} + u_{j+1}^{(0)}) + 553(u_{j+1}^{(1)} - u_{j-1}^{(1)}) - \\ -455(u_{j-1}^{(2)} + u_{j+1}^{(2)}) + 243(u_{j+1}^{(3)} - u_{j-1}^{(3)}) \end{array} \right) /20093184 \\ \left(\begin{array}{c} 5294u_j^{(1)} + 7234u_j^{(3)} + 2375(u_{j-1}^{(0)} - u_{j+1}^{(0)}) + 2103(u_{j-1}^{(1)} + u_{j+1}^{(1)}) + \\ +1513(u_{j-1}^{(2)} - u_{j+1}^{(2)}) + 673(u_{j-1}^{(3)} + u_{j+1}^{(3)}) \end{array} \right) /180838656 \end{bmatrix} \tag{61}$$

Appendix B: Inter-Cell Recovery (Diffusion Operator)

The weak form of the diffusion operator

$$\left[\mathcal{D}(x)\mathbb{U}_x\right]_x = \left[\begin{pmatrix} 0 & 0 & 0 \\ -\frac{m\eta}{\rho^2} & \frac{\eta}{\rho} & 0 \\ \frac{\beta\left(\frac{m^2}{\rho} - E\right) - \frac{\eta m^2}{\rho}}{\rho^2} & \frac{m(\eta-\beta)}{\rho^2} & -\frac{\beta}{\rho} \end{pmatrix} \mathbb{U}_x \right]_x = \left[\begin{pmatrix} 0 & 0 & 0 \\ 0 & \eta & 0 \\ 0 & u\eta & \beta \end{pmatrix} \mathbb{V}_x \right]_x \tag{62}$$

can be written as the recovery form introduced by van Leer & Nomura[44]:

$$\begin{aligned} \int_{\mathfrak{I}_\mathrm{j}} \partial_t \mathbb{U}_h(x,t)\, \mathfrak{L}_{(\mathrm{m})}(x)dx + ... = & \underbrace{\left[\mathcal{D}\left(\mathfrak{L}_{(\mathrm{m})}\mathcal{V}' - \mathfrak{L}'_{(\mathrm{m})}\mathcal{V}\right)\right]_{\mathrm{j}-\frac{1}{2},\mathrm{R}}^{\mathrm{j}+\frac{1}{2},\mathrm{L}}}_{\text{Numerical Diffusion Flux}} + \\ & + \int_{\mathfrak{I}_\mathrm{j}} \left(\mathcal{D}\mathfrak{L}''_{(\mathrm{m})} + \mathcal{D}'\mathfrak{L}'_{(\mathrm{m})}\right) \mathbb{V}_h(x,t)\, dx + ... \end{aligned} \tag{63}$$

A continuous "recovery" profile $\mathcal{V}(\zeta)$ between cells $_{[\mathrm{j}-1]}$ and $_{[\mathrm{j}]}$ is reconstructed from the in-cell discontinuous solutions* $\mathbb{V}_h(x,t)$ as

---

* In-cell distribution of primitive variables is computed as $\mathbb{V}_{h_j}(x) = \sum_{n=0}^{R} \mathbb{V}_j^{(n)} \mathfrak{L}_{(n)}(\xi)$, where

$$\mathbb{V}_j^{(n)} = \frac{1+2n}{\Delta x_j} \int_{\mathcal{I}_j} \mathfrak{L}_{(n)}(\xi)\, \mathbb{V}\left(\mathbb{U}_{h_j}(x)\right) dx$$

and the integral is evaluated using a 12-point Gaussian quadrature formula.

$$\mathcal{V}(\zeta) = \sum_{k=0}^{N=4p+3} \mathcal{H}_{(\mathrm{k})}(\zeta)\, \mathcal{V}_{(\mathrm{k})} \tag{64}$$

where $\zeta \equiv x - x_{\mathrm{j}-\frac{1}{2}}$ and $\mathcal{H}_{(\mathrm{k})}$ is the Hermite polynomial of the k^{th} order. $\mathcal{V}(\zeta)$ is reconstructed in the weak sense, using the following van Leer's "recovery" constraints[44]:

$$\int_{\mathfrak{I}_{\mathrm{m}}} \mathfrak{L}_{(\mathrm{n})}(\xi_m)\, \mathbb{V}_h\, dx = \int_{\mathfrak{I}_{\mathrm{m}}} \mathfrak{L}_{(\mathrm{n})}(\xi_m)\, \mathcal{V}(\zeta)\, dx \qquad \text{where} \quad \begin{pmatrix} \mathrm{n} = 0, ..., p \\ \mathrm{m} = j, j \pm 1, j-2 \end{pmatrix} \tag{65}$$

where $\xi_m = \frac{2(x-x_m)}{\Delta_m}$. By including cells $_{(j-2)}$ and $_{(j+1)}$ we ensure that approximation by eq.(64) is at least as accurate as eq.(5), i.e. $(N = 4p+3) > (R = 3p+2)$), and it involves the same compact stencil *s*=2 for interpolation.

The necessary "recovery" quantities $\mathcal{V}(0)$ and $\mathcal{V}'(0)$ are computed as:

Piecewise-constant, rDG$_{0(4)}$:

$$\begin{aligned} \mathcal{V} &= \frac{7(V_{\mathrm{j}}^{(0)}+V_{\mathrm{j}-1}^{(0)})-V_{\mathrm{j}-2}^{(0)}-V_{\mathrm{j}+1}^{(0)}}{12} \\ \mathcal{V}' &= \frac{15(V_{\mathrm{j}}^{(0)}-V_{\mathrm{j}-1}^{(0)})+V_{\mathrm{j}-2}^{(0)}-V_{\mathrm{j}+1}^{(0)}}{12\Delta x_j} \end{aligned} \tag{66}$$

Piecewise-linear, rDG$_{1(8)}$:

$$\begin{aligned} \mathcal{V} &= \frac{150(V_{\mathrm{j}}^{(0)}+V_{\mathrm{j}-1}^{(0)})+135(V_{\mathrm{j}-1}^{(1)}-V_{\mathrm{j}}^{(1)})+12(V_{\mathrm{j}-2}^{(0)}+V_{\mathrm{j}+1}^{(0)})+7(V_{\mathrm{j}-2}^{(1)}-V_{\mathrm{j}+1}^{(1)})}{324} \\ \mathcal{V}' &= -\frac{2163(V_{\mathrm{j}-1}^{(0)}-V_{\mathrm{j}}^{(0)})+1439(V_{\mathrm{j}}^{(1)}+V_{\mathrm{j}-1}^{(1)})+57(V_{\mathrm{j}-2}^{(0)}-V_{\mathrm{j}+1}^{(0)})+31(V_{\mathrm{j}-2}^{(1)}+V_{\mathrm{j}+1}^{(1)})}{864\Delta x_j} \end{aligned} \tag{67}$$

Piecewise-quadratic, rDG$_{2(12)}$:

$$\begin{aligned} \mathcal{V} &= \begin{pmatrix} 53710\left(V_{\mathrm{j}}^{(0)}+V_{\mathrm{j}-1}^{(0)}\right)+41245\left(V_{\mathrm{j}-1}^{(1)}-V_{\mathrm{j}}^{(1)}\right)+30631\left(V_{\mathrm{j}}^{(2)}+V_{\mathrm{j}-1}^{(2)}\right)- \\ -1870\left(V_{\mathrm{j}-2}^{(0)}+V_{\mathrm{j}+1}^{(0)}\right)+1385\left(V_{\mathrm{j}+1}^{(1)}-V_{\mathrm{j}-2}^{(1)}\right)-601\left(V_{\mathrm{j}-2}^{(2)}+V_{\mathrm{j}+1}^{(2)}\right) \end{pmatrix} /103680 \\ \mathcal{V}' &= \frac{\begin{pmatrix} 55695\left(V_{\mathrm{j}}^{(0)}-V_{\mathrm{j}-1}^{(0)}\right)-41375\left(V_{\mathrm{j}}^{(1)}+V_{\mathrm{j}-1}^{(1)}\right)+22356\left(V_{\mathrm{j}}^{(2)}-V_{\mathrm{j}-1}^{(2)}\right)+ \\ +595\left(V_{\mathrm{j}-2}^{(0)}-V_{\mathrm{j}+1}^{(0)}\right)+425\left(V_{\mathrm{j}-2}^{(1)}+V_{\mathrm{j}+1}^{(1)}\right)+172\left(V_{\mathrm{j}-2}^{(2)}-V_{\mathrm{j}+1}^{(2)}\right) \end{pmatrix}}{12960\Delta x_j} \end{aligned} \tag{68}$$

Piecewise-cubic, $rDG_{3(16)}$:

$$\begin{aligned}
\mathcal{V} &= \left(\begin{array}{c} 4808930\left(V_{\mathrm{j}}^{(0)}+V_{\mathrm{j-1}}^{(0)}\right)+4478047\left(V_{\mathrm{j-1}}^{(1)}-V_{\mathrm{j}}^{(1)}\right)+3170461\left(V_{\mathrm{j}}^{(2)}+V_{\mathrm{j-1}}^{(2)}\right)+ \\ +2047032\left(V_{\mathrm{j-1}}^{(3)}-V_{\mathrm{j}}^{(3)}\right)+89950\left(V_{\mathrm{j-2}}^{(0)}+V_{\mathrm{j+1}}^{(0)}\right)+73101\left(V_{\mathrm{j-2}}^{(1)}-V_{\mathrm{j+1}}^{(1)}\right)+ \\ +44429\left(V_{\mathrm{j-2}}^{(2)}+V_{\mathrm{j+1}}^{(2)}\right)+15416\left(V_{\mathrm{j-2}}^{(3)}-V_{\mathrm{j+1}}^{(3)}\right) \end{array} \right) /9797760 \\
\mathcal{V}' &= -\frac{\left(\begin{array}{c} 96373585\left(V_{\mathrm{j-1}}^{(0)}-V_{\mathrm{j}}^{(0)}\right)+81745993\left(V_{\mathrm{j}}^{(1)}+V_{\mathrm{j-1}}^{(1)}\right)+53235413\left(V_{\mathrm{j-1}}^{(2)}-V_{\mathrm{j}}^{(2)}\right)+ \\ +25147453\left(V_{\mathrm{j}}^{(3)}+V_{\mathrm{j-1}}^{(3)}\right)+475615\left(V_{\mathrm{j-2}}^{(0)}-V_{\mathrm{j+1}}^{(0)}\right)+378021\left(V_{\mathrm{j-2}}^{(1)}+V_{\mathrm{j+1}}^{(1)}\right)+ \\ +220031\left(V_{\mathrm{j-2}}^{(2)}-V_{\mathrm{j+1}}^{(2)}\right)+71741\left(V_{\mathrm{j-2}}^{(3)}+V_{\mathrm{j+1}}^{(3)}\right) \end{array} \right)}{15676416\Delta x_j}
\end{aligned} \tag{69}$$

Appendix C: Butcher Tableau for $ESDIRK_{3,4,5}$

$\underline{\mathrm{ESDIRK}_3}$

0	0	0	0	0
$\frac{1767732205903}{2027836641118}$	$\frac{1767732205903}{4055673282236}$	$\frac{1767732205903}{4055673282236}$	0	0
$\frac{3}{5}$	$\frac{2746238789719}{10658868560708}$	$-\frac{640167445237}{6845629431997}$	$\frac{1767732205903}{4055673282236}$	0
1	$\frac{1471266399579}{7840856788654}$	$-\frac{4482444167858}{7529755066697}$	$\frac{11266239266428}{11593286722821}$	$\frac{1767732205903}{4055673282236}$
b_r	$\frac{1471266399579}{7840856788654}$	$-\frac{4482444167858}{7529755066697}$	$\frac{11266239266428}{11593286722821}$	$\frac{1767732205903}{4055673282236}$
$\hat{\mathsf{b}}_r$	$\frac{2756265671327}{12835298489170}$	$-\frac{10771552573575}{22201958757719}$	$\frac{9247589265047}{10645013368117}$	$\frac{2193209047091}{5459859503100}$

ESDIRK$_4$

0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0
$\frac{83}{250}$	$\frac{8611}{62500}$	$-\frac{1743}{31250}$	$\frac{1}{4}$	0	0	0
$\frac{31}{50}$	$\frac{5012029}{34652500}$	$-\frac{654441}{2922500}$	$\frac{174375}{388108}$	$\frac{1}{4}$	0	0
$\frac{17}{20}$	$\frac{15267082809}{155376265600}$	$-\frac{71443401}{120774400}$	$\frac{730878875}{902184768}$	$\frac{2285395}{8070912}$	$\frac{1}{4}$	0
1	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$
b_r	$\frac{82889}{524892}$	0	$\frac{15625}{83664}$	$\frac{69875}{102672}$	$-\frac{2260}{8211}$	$\frac{1}{4}$
$\hat{\mathsf{b}}_r$	$\frac{4586570599}{29645900160}$	0	$\frac{178811875}{945068544}$	$\frac{814220225}{1159782912}$	$-\frac{3700637}{11593932}$	$\frac{61727}{225920}$

ESDIRK$_5$

0	0	0	0	0	0	0	0	0
$\frac{41}{100}$	$\frac{41}{200}$	$\frac{41}{200}$	0	0	0	0	0	0
$\frac{2935347310677}{11292855782101}$	$\frac{41}{400}$	$\frac{-567603406766}{11931857230679}$	$\frac{41}{200}$	0	0	0	0	0
$\frac{1426016391358}{7196633302097}$	$\frac{683785636431}{9252920307686}$	0	$\frac{-110385047103}{1367015193373}$	$\frac{41}{200}$	0	0	0	0
$\frac{92}{100}$	$\frac{3016520224154}{10081342136671}$	0	$\frac{30586259806659}{12414158314087}$	$\frac{-22760509404356}{11113319521817}$	$\frac{41}{200}$	0	0	0
$\frac{24}{100}$	$\frac{218866479029}{1489978393911}$	0	$\frac{638256894668}{5436446318841}$	$\frac{-1179710474555}{5321154724896}$	$\frac{-60928119172}{8023461067671}$	$\frac{41}{200}$	0	0
$\frac{3}{5}$	$\frac{1020004230633}{5715676835656}$	0	$\frac{25762820946817}{25263940353407}$	$-\frac{2161375909145}{9755907335909}$	$\frac{-211217309593}{5846859502534}$	$\frac{-4269925059573}{7827059040749}$	$\frac{41}{200}$	0
1	$\frac{-872700587467}{9133579230613}$	0	0	$\frac{22348218063261}{9555858737531}$	$\frac{-1143369518992}{8141816002931}$	$\frac{-39379526789629}{19018526304540}$	$\frac{32727382324388}{42900044865799}$	$\frac{41}{200}$
b_r	$\frac{-872700587467}{9133579230613}$	0	0	$\frac{22348218063261}{9555858737531}$	$\frac{-1143369518992}{8141816002931}$	$\frac{-39379526789629}{19018526304540}$	$\frac{32727382324388}{42900044865799}$	$\frac{41}{200}$

Acknowledgments

This work is the leveraged outgrowth of a fundamentally-oriented aspect of two large mission-focused programs: At INL this work is a part of a code development effort for nuclear power reactor safety, under "*Multi-physics Simulation Methods for Advanced Reactor Analysis*" and "*A General Framework for Simulating Fully-Coupled Mass and Energy Transport Phenomena in Nuclear Energy Systems*" LDRD Projects (prepared for the U.S. Department of Energy Office of Nuclear Energy, under DOE Idaho Operations Office Contract DE-AC07-05ID14517). At UCSB, this work is a part of the ASOS program for consequence assessment and risk management of threat agent dispersal scenarios, sponsored by the Joint Science and Technology Center of the Defense Threat Reduction Agency (JSTO/DTRA), Lawrence Livermore National Laboratory (the HOPS program), and the National Ground Intelligence Center (NGIC).

References

[1] Abgrall, R., "Generalization of Roe Scheme for the Computation of Mixture of Perfect Gases," *Rech. A'erosp.*, Vol. 6, 1988. p. 6.

[2] Abgrall, R., "How to Prevent Pressure Oscillations in Multicomponent Flow Calculations: A Quasi Conservative Approach," *Journal of Computational Physics*, Vol. 125, 1996, pp. 150, 160.

[3] Abgrall, R., and Karni, S., "Computations of Compressible Multifluids," *Journal of Computational Physics*, Vol. 169, 2001, pp. 594, 623.

[4] Arnold, D.N., "An Interior Penalty Finite Element Method with Discontinuous Elements," *SIAM Journal on Numerical Analysis*, Vol. 19, 1982, pp. 742, 760.

[5] Bijl, H., Carpenter, M.H., Vatsa, V.N., and Kennedy, C.A., "Implicit Time Integration Schemes for the Unsteady Compressible Navier-Stokes Equations: Laminar Flow," *Journal of Computational Physics*, Vol. 179, 2002, pp. 313, 329.

[6] Carpenter, M.H., Kennedy, C.A., Bijl, H., Vilken, S.A., and Vatsa, V.N., "Fourth-Order Runge-Kutta Schemes for Fluid Mechanics Applications," *Journal of Scientific Computing*, Vol. 25, No. ½, Nov. 2005, pp. 157, 194.

[7] Chorin, A.J., "Numerical Solution of the Navier Stokes Equations," *Mathematics of Computation*, Vol. 22, 745762, 1968.

[8] Cockburn, B., "An Introduction to the Discontinuous Galerkin Method for Convection-Dominated Problems," In "Advanced Numerical Approximation of Nonlinear Hyperbolic Equations", *Lecture Notes in Mathematics*, Vol. 1697/1998, Springer Berlin/Heidelberg.

[9] Cockburn, B., and Shu, C.-W., "TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws II: General Framework," *Mathematics of Computation*, Vol. 52, 1989, pp. 411, 435.

[10] Cockburn, B., Lin, S.-Y., Shu, C.-W., "TVB Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws III: One-Dimensional Systems," *Journal of Computational Physics*, Vol. 84, 1989, pp. 90, 113.

[11] Cockburn, B., Hou, S., and Shu, C.-W., "The Runge-Kutta Local Projection Discontinuous Galerkin Finite Element Method for Conservation Laws IV: the Multidimensional Case," *Mathematics of Computation*, Vol. 54, 1990, pp. 545, 581.

[12] Dinh, T.N., and Theofanous, T.G., "Nucleation Phenomena in Boiling," *Multiphase Science and Technology*, Vol. 15, No.1-4, 2003, pp. 349, 363.

[13] Dinh, T.N., and Tu, J.P., "The Micro-Hydrodynamics that Govern Critical Heat Flux in Pool Boiling," CD-ROM Proceedings of the *International Conference on Multiphase Flow*, Leipzig, Germany, July 9-13, 2007.

[14] Fedkiw, R.P., Aslam, T., Merriman, B., and Osher, S., "Non-oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (the Ghost Fluid Method)," *Journal of Computational Physics*, Vol. 152, 1999, pp. 457, 492.

[15] Harlow, F., and Amsden, A., "A Numerical Fluid Dynamical Calculation Method for All Flow Speed," *Journal of Computational Physics*, Vol. 8, 1971, pp. 197, 214.

[16] Karni, S., "Multicomponent Flow Calculations by a Consistent Primitive Algorithm," *Journal of Computational Physics*, Vol. 112, 1994, pp. 31, 43.

[17] Karni, S., "Hybrid Multifluid Algorithms," *SIAM Journal on Scientific Computing*, Vol. 17, No. 5, 1996, pp. 1019, 1039.

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

[19] Knoll, D.A., and Keyes, D., "Jacobian-free Newton-Krylov Methods: A Survey of Approaches and Applications," *Journal of Computational Physics*, Vol. 193, 2003, pp. 357, 397.

[20] Nourgaliev, R.R., Liou, M.-S., and Theofanous, T.G., "Numerical Prediction of Interfacial Instability: Sharp Interface Method (SIM)," *Journal of Computational Physics*, (in press, January 2008).

[21] Nourgaliev, R.R., and Theofanous, T.G., "High-Fidelity Interface Tracking in Compressible Flows: Unlimited Anchored Adaptive Level Set," *Journal of Computational Physics*, Vol. 224, No. 2, 2007, pp. 836, 866.

[22] Nourgaliev, R.R., Dinh, T.N., and Theofanous, T.G., "Adaptive Characteristics-Based Matching for Compressible Multifluid Dynamics," *Journal of Computational Physics*, Vol. 213, No. 2, 2006, pp. 500, 529.

[23] Nourgaliev, R.R., Dinh, T.N., and Theofanous, T.G., "A Pseudocompressibility Method for the Simulation of Multiphase Incompressible Flows," *International Journal of Multiphase Flow*, Vol. 30, No. 7-8, 2004, pp. 901, 937.

[24] Nourgaliev, R.R., Knoll, D., Mousseau, V., and Berry, R., "Direct Numerical Simulation of Boiling Multiphase Flows: State-of-the-Art, Modeling, Algorithmic and Computer Needs," *Joint International Topical Meeting on Mathematics & Computation and Supercomputing in Nuclear Applications (M&C + SNA 2007)* Monterey, California, April 15-19, 2007, on CD-ROM, American Nuclear Society, LaGrange Park, IL.

[25] Liou, M.-S., "A sequel to AUSM, Part II: AUSM(+)-up for all speeds," *Journal of Computational Physics*, Vol. 214, No. 1, 2006, pp. 137, 170.

[26] Mousseau, V.A., "A fully implicit hybrid solution method for a two-phase thermal-hydraulic model," *Journal of Heat Transfer*, Vol. 127, 2005, pp. 531, 539.

[27] Mousseau, V.A., "Implicitly balanced solution of the two-phase flow equations coupled to nonlinear heat conduction," *Journal of Computational Physics*, Vol. 200, 2004, pp. 104,132.

[28] Oden, J.T., Babuska, I., and Baumann, C.E., "A Discontinuous *hp*-Finite Element Method for Diffusion Problems," *Journal of Computational Physics*, Vol. 146, 1998, pp. 491, 519.

[29] Patankar, S.V., "Numerical Heat Transfer and Fluid Flow," *Hemisphere Publishing*, New York, 1980.

[30] Press, W.H. , Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., "Numerical Recipies in C (The Art of Scientific Computing)," 2nd ed., Cambridge University Press, Cambridge, 1997.
[31] Reed, W., and Hill, T., "Triangular Mesh Methods for the Neutron Transport Equation," *Tech. Rep. LA-UR 73-479*, Los Alamos National Laboratory, 1973.
[32] Saad, Y., and Schultz, M.H., "GMRES: a Generalized Minimal Residual Algorithm for Solving Non-Symmetric Linear Systems," *SIAM J. Sci. Stat. Comput.*, Vol. 7, 1986, p. 856.
[33] Saad, Y., *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[34] Saurel, R., and Abgrall, R., "A Simple Method for Compressible Multifluid Flows," *SIAM Journal on Scientific Computing*, Vol. 21, No. 3, 1999, pp. 1115, 1145.
[35] Shu, C.-W., and Osher, S., "Efficient Implementation of Essentially Non-Oscillatory Shock-Capturing Schemes," *Journal of Computational Physics*, Vol. 77, 1989, pp. 439, 471.
[36] Theofanous, T.G., Tu, J.P., Dinh, A.T., Dinh, T.N., "The Boiling Crisis Phenomenon. Part II: Dryout Dynamics and Burnout," *Experimental Thermal and Fluid Science*, Vol. 26**,** 2002, pp. 793, 810.
[37] Theofanous, T.G. Nourgaliev, R.R., Li, G.J., and Dihn, T.N., "Compressible Multi-Hydrodynamics (CMH): Breakup, Mixing, and Dispersal, of Liquids/Solids in High Speed Flows," *"Computational Approaches to Disperse Multiphase Flow"*, Ed. A.Prosperetti and S.Balachandar, Springer Verlag Heidelberg, 2006.
[38] Theofanous, T.G., Li, G.J., Dinh, T.N., and Chang, C.H., "Aerobreakup in disturbed subsonic and supersonic flow fields," *Journal of Fluid Mechanics*, Vol. 593, 2007, pp. 131, 170.
[39] Theofanous, T.G., Nourgaliev, R.R., and Wiri, S., *Short Communication:* "Direct numerical simulations of two-layer viscosity-stratified flow by Qing Cao, Kausik Sarkar, Ajay K. Prasad," *International Journal of Multiphase Flow*, Vol. 33, No. 7, 2007, pp. 789, 796.
[40] Theofanous, T.G., Nourgaliev, R.R., and Khomami, B., *Short Communication:* "An experimental/theoretical investigation of interfacial instabilities in superposed pressure-driven channel flow of Newtonian and well characterized viscoelastic fluids: Part I: linear stability and encapsulation effects, by Bamin Khomami and Kuan C. Su," *Journal of Non-Newtonian Fluid Mechanics*, Vol. 143, 2007, pp. 131, 132.
[41] Theofanous, T.G., Shushchikh, S.Yu., and Nourgaliev, R.R., "Linear stability of sharp and diffuse interfaces under shear," *5th ASME/JSME Fluids Engineering Conference*, FEDMS2007-37337.
[42] Theofanous, T.G., and Dinh, T.N, "High Heat Flux Boiling and Burnout as Microphysical Phenomena: Mounting Evidence and Opportunities," *Multiphase Science and Technology*, Vol. 18, No. 1, 2006, pp. 1, 26.
[43] van Leer, B., "Towards the Ultimate Conservation Difference Scheme. IV. A New Approach to Numerical Convection," *Journal of Computational Physics*, Vol. 23, 1977, pp. 276, 299.
[44] van Leer, B., and Nomura, S., "Discontinuous Galerkin for Diffusion," AIAA 2005-5108, *17th AIAA Computational Fluid Dynamics Conference*, 6-9 June 2005, Toronto, Ontario, Canada.
[45] van Leer, B., Lo, M., and van Raalte, M., "A Discontinuous Galerkin Method for Diffusion Based on Recovery," AIAA 2007-4083, *18th AIAA Computational Fluid Dynamics Conference*, 26-28 June 2007, Miami, Florida, USA.