Neural-network-based Riemann solver for real fluids and high explosives; application to computational fluid dynamics

Cite as: Physics of Fluids (in press) (2022); https://doi.org/10.1063/5.0123466
Submitted: 30 August 2022 • Accepted: 15 October 2022 • Accepted Manuscript Online: 20 October 2022

Matteo Ruggeri, Indradip Roy, Michael J. Mueterthies, et al.

Accepted to Phys. Fluids 10.1063/5.0123466

Neural-network-based Riemann solver for real fluids and high explosives; application to computational fluid dynamics

Matteo Ruggeri,[1] Indradip Roy,[2] Michael J. Mueterthies,[3] Tom Gruenwald,[3] and Carlo Scalo[2,4]
[1)]*School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN, 47906, USA*
[2)]*School of Mechanical Engineering, Purdue University, West Lafayette, IN, 47906, USA*
[3)]*Blue Wave AI Labs, West Lafayette, IN, 47906, USA*
[4)]*Chief Executive Officer, HySonic Technologies, LLC, West Lafayette, IN, 47906-4182, USA*

(*Electronic mail: mruggeri@purdue.edu)

(Dated: 6 October 2022)

The Riemann problem is fundamental to most computational fluid dynamics (CFD) codes for simulating compressible flows. The time to obtain the exact solution to this problem for real fluids is high because of the complexity of the fluid model, which includes the equation of state; as a result, approximate Riemann solvers are used in lieu of the exact ones, even for ideal gases. We used fully connected feedforward neural networks to find the solution to the Riemann problem for calorically imperfect gases (CI), supercritical fluids (SF), and high explosives (HE), and then embedded these network into a one-dimensional finite volume CFD code. We showed that for real fluids, the neural networks can be more than 5 orders of magnitude faster than the exact solver, with prediction errors below 0.8%. The same neural networks embedded in a CFD code yields very good agreement with the overall exact solution, with a speed-up of three orders of magnitude with respect to the same CFD code that use the exact Riemann solver to resolve the flux at the interfaces. Compared to the Rusanov flux reconstruction method, the neural network is half as fast, but yields a higher accuracy and is able to converge to the exact solution with a coarser grid.

I. INTRODUCTION

This paper aims to develop a neural network model for the Riemann problem (defined in section II) – a canonical fluid dynamic problem described by partial differential equations and the building block of many compressible computational fluid dynamics (CFD) codes[1]. To our knowledge, only two papers[2,3] have dealt with the application of machine learning to approximate the solution to the Riemann problem, and only one applied this approach to CFD methods. Only Wang[4] dealt with Riemann problem applied to non-ideal equations. He used a physics-informed neural network to solve the Riemann problem for supercritical fluids, and then integrate the network in a CFD solver. Other publications[5–7] have attempted to approximate the solution to the Euler equations, the equations which describe the Riemann problem; however, their neural networks approximate the solution at each grid point in the simulation domain instead of the specific solution to the Riemann problem as will be shown below. All other publications that applied machine learning to CFD models focus on improving the accuracy of commonly used approximate methods. In addition, most of the existing works focus on applying machine learning to ideal gases, while we will focus on real fluids, which are more challenging and expensive to model, with applications including energy reutilization, thermal management, and thermoacoustic instabilities[8–18].

The application of machine learning is starting to be widespread in science and engineering problems due to the significant advances in computational power, algorithmic efficiency and accessibility of machine learning tools. Some of the new research focuses on the development of new neural networks architectures like graph neural networks (GNN[19]) and neural ordinary differential equations (neuralODE[20]). Others instead point out the strength of the learning algorithms and their best configurations. This is the case of Kratsion[21], who shows that a neural network with a leaky ReLu activation function can approximate any continuous function with non-pathological growth. This result is significant for utilizing neural networks in physics, considering that most of the problems are described by equations that currently have no analytical solutions. In the application of machine learning to physical problems, there are mainly two kinds of neural networks: the ones that predict the partial differential equations (PDE) that govern the phenomena[22–24] and the ones that predict the solution of the PDE[25–27]. Raissi et al.[26] introduce the concept of numerical Gaussian processes that are Gaussian processes informed of the equations that govern the physics of the phenomena, and they are used to quantify the uncertainty caused by initial noisy data. On the same line, they used the physical equations inside neural networks[27], developing a new kind of network called physics-informed neural networks (PINN).

In the following paper, we firstly describe the Riemann problem in section II and all of its possible outcomes and expose in subsection II A the exact Riemann solver used to generate the dataset and for training and performance evaluation of the neural networks. In section III, we give an in-depth literature review of the application of machine learning to the Riemann problem (section III A) and in general to CFD codes (section III B). Section IV presents the neural networks and all the configurations used, along with the various equations of state and ranges of values that we considered. In section V, we compare the performance of the neural networks in terms of accuracy and time, with the exact Riemann solver for each of the fluid models considered. And in section VI, we compare the numerical solutions from a full one-dimensional CFD

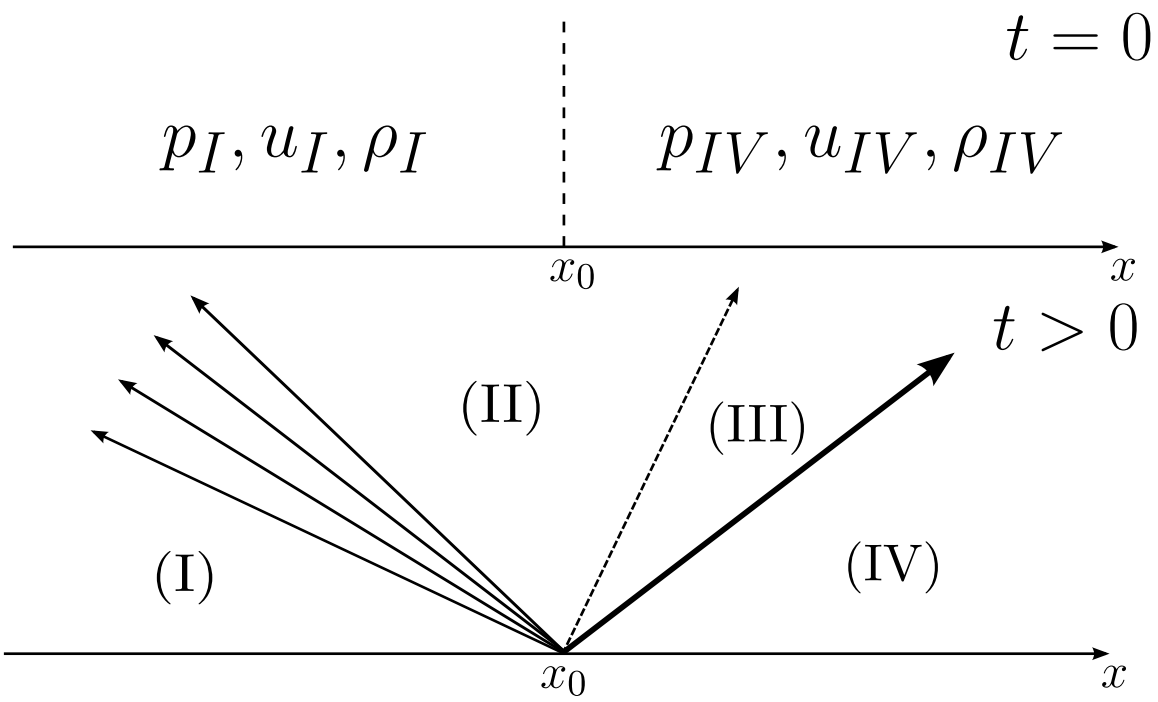


FIG. 1. Initial conditions of the one-dimensional Riemann problem for gas dynamics, with a discontinuity at $x = x_0$ and $t = 0$. With left state denoted as I and right state as IV. Some thermo-fluid-dynamic quantities may be uniform across the interface $x = x_0$. It is enough for the velocity or pressure to be discontinuous to generate waves. If only the density or temperature are discontinuous, the inviscid solution is equal to the initial conditions.

solver obtained with standard numerical methods, exact Riemann solvers and neural networks applied at each computational face.

II. RIEMANN PROBLEM

The Riemann problem is an initial value problem for hyperbolic partial differential equations. In particular, as shown in Figure 1, in gas dynamics, it requires two constant thermo-fluid-dynamic states as initial values, one left (state I) and the other right (state IV) of a specific point $x = x_0$. The solution to this problem is a combination of waves that propagate left and right from the point of the initial discontinuity. Despite being a theoretical problem, it has wide applications in practical compressible flow codes, as it can be used to calculate the numerical flux between two mesh cells.

Our interest is to investigate the Riemann problem for gas dynamics that is described by the Euler equations: three partial differential equations, that govern the dynamics of mass, momentum, and energy per unit volume. They are valid for a non-conducting and inviscid compressible flow and, they read:

$$\mathbf{Q}_t + \mathbf{F}(\mathbf{Q})_x = 0 \tag{1}$$

where $\mathbf{Q}$ is the state vector, and $\mathbf{F}$ is the flux vector. For a one dimensional case, the vectors are:

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix} \tag{2}$$

where ρ is the density, u is the velocity, p is the pressure, $E = \rho e + \rho \frac{u^2}{2}$ is the total energy per unit volume, and e is the specific internal energy. To solve this system of equations, another equation that describes the thermodynamic behavior of the flow and how the state variables are related is required: the equation of state (EoS). In this paper we will examine the ideal gas equation of state with a calorically imperfect gas (CI) closure, and the more complex supercritical fluids (SF) and high explosives (HE) fluid models. Equation (1) can be rewritten as:

$$\frac{\partial \mathbf{Q}}{\partial t} + \left(\frac{\partial \mathbf{F}}{\partial \mathbf{Q}}\right) \cdot \nabla \mathbf{Q} = 0 \tag{3}$$

which reveals the hyperbolic nature of these equations, and the Jacobian of the flux $\frac{\partial \mathbf{F}}{\partial \mathbf{Q}}$ describes the nature of the waves generated in the initial point of discontinuity.

The solution of equations (1) are continuous for rarefaction waves, instead shock waves could have a discontinuity (quasi-linear solution) in the solution function because of the change in the entropy of the flow. It is still possible to find a solution of the Euler equations (1) with shock waves in the problem, and the Rankine-Hugoniot equations describe their behavior:

$$\begin{aligned}\left[\rho(u-s_w)\right] &= 0\\ \left[\rho(u-s_w)^2+p\right] &= 0\\ \left[e+\frac{p}{\rho}+\frac{(u-s_w)^2}{2}\right] &= 0\end{aligned} \tag{4}$$

where s_w is the speed of the shock wave, and the operator $[\cdot]$ indicates the difference between the states before and after the wave. If we consider a tube with gas in two states separated by a diaphragm:

$$\mathbf{Q}(x,0) = \begin{cases}\mathbf{Q}_I & \text{if} \quad x \le x_0\\ \mathbf{Q}_{IV} & \text{if} \quad x > x_0\end{cases} \tag{5}$$

we have a Riemann problem, a phenomenon described by hyperbolic equations and has two constant initial states.

Figure 2 shows the physically admissible outcomes of the Riemann problem, from left to right, and top to bottom: a rarefaction wave to the left and a shock wave to the right (RCS), a shock wave to the left and a rarefaction wave to the right (SCR), shock waves in both directions (SCS), and rarefaction waves in both directions (RCR). In the figure, the S stays for shock wave, the Rankine-Hugoniot equations (4) describe the discontinuous change between states; the C stays for contact discontinuity–the pressure and the velocities are the same in states II and III, but the density, temperature, speed of sound, and energy change–; and R stays for rarefaction wave, in which the values between the states change gradually obeying the Euler equations (1). The first two solutions can be obtained considering a particular case of the Riemann problem–the shock tube problem. Both starting velocities are equal to zero and the two gases are separated by a diaphragm, with different pressures and densities left and right of it. When the diaphragm is removed, the gas with the highest pressure flows toward the other, generating a shock wave that propagates in the gas with the lower pressure and a rarefaction wave in the gas with the highest pressure. The other two scenarios can be obtained considering the interaction between two shock or rarefaction waves that reflect themselves. These scenarios can be generated from two pistons in a tube that moves one against the other to generate two shock waves or having opposite directions to generate two rarefaction waves, as shown in Figure 2. Figure 3 reports the values used to describe each algorithm and state value in this work. The initial state values on the left have subscript I, on the right have subscript IV, and for the middle state subscript II and III respectively for left and right of the contact discontinuity. Since the velocity and the pressure are the same for states II and III, $p^* = p_{II} = p_{III}$ and $u^* = u_{II} = u_{III}$, sometimes the upper * is also used to identify the other thermodynamic values of the middle states.

A. Exact Riemann Solver for real fluids

The EoS for real fluids adds several nonlinearities to the Riemann problem, and hence the Riemann solver for real fluids needs several orders of magnitude of more time than the solver for the ideal gases (expose in appendix A) to obtain the solution. Colella[28] proposed an algorithm to solve the Riemann problem for general convex EoS, and Kamm[29] exposed the main element to write the software to implement this algorithm. The solver iterates the pressure p^* and measures the accuracy with the difference in velocity between states II and III.

The initial value of the pressure is obtained from the acoustic approximation:

$$p^0 = \frac{C_{IV}\rho_I + C_I\rho_{IV} + C_I C_{IV}(u_I - u_{IV})}{C_I + C_{IV}} \tag{6}$$

where $C = \rho a$ is the mass flux. If p^i (p^* at the i-th iteartive step) is greater than the pressure in states I or IV, a shock wave is formed and the Rankine-Hugoniot equations (4) are needed to find the solution. These jump conditions can be reduced to a single, implicit equation to solve for the density in the middle state ρ_N^i:

$$\begin{aligned}e_K(p^i,\rho_K) + \frac{p_K}{\rho_K} + \frac{1}{2}\frac{\rho_N^i}{\rho_K}\frac{p^i - p_K}{\rho_N^i - \rho_K} - e_K(p^i,\rho_N^i) +\\ - \frac{p_i}{\rho_N^i} - \frac{1}{2}\frac{\rho_K}{\rho_N^i}\frac{p^i - p_K}{\rho_N^i - \rho_K} = 0\end{aligned} \tag{7}$$

where $N = \{II, III\}$, $K = \{I, IV\}$. From ρ_N^i, it is possible to compute the velocity:

$$u_N^i = u_K \mp \sqrt{\frac{\rho_N^i}{\rho_K}\frac{p^i - p_K}{\rho_N^i - \rho_K}} \pm \sqrt{\frac{\rho_K}{\rho_N^i}\frac{p^i - p_K}{\rho_N^i - \rho_K}} \tag{8}$$

where the upper sign is from the state I to II and the lower from IV to III.

In case p^i is lower than the pressure in states I or IV, rarefaction waves form and the states are determined by solving the ordinary differential equations for pressure and velocity:

$$\frac{dp}{d\rho} = a^2(\rho, p) \tag{9}$$

$$\frac{du}{d\rho} = \mp\frac{a(\rho,p)}{\rho} \tag{10}$$

that are solved with a fourth-order Runge-Kutta scheme. This part of the code is the most computationally expensive, mainly because the best $\Delta\rho$ for the scheme is unknown apriori, and the code has to find it.

The middle pressure is updated with a secant method instead of Newton-Raphson (used for ideal gases), because it is difficult to find the exact derivatives for the variables. Therefore, a second estimation of the pressure is required in the first part of the algorithm, and it is obtained using (6) with an improved value of the mass flux:

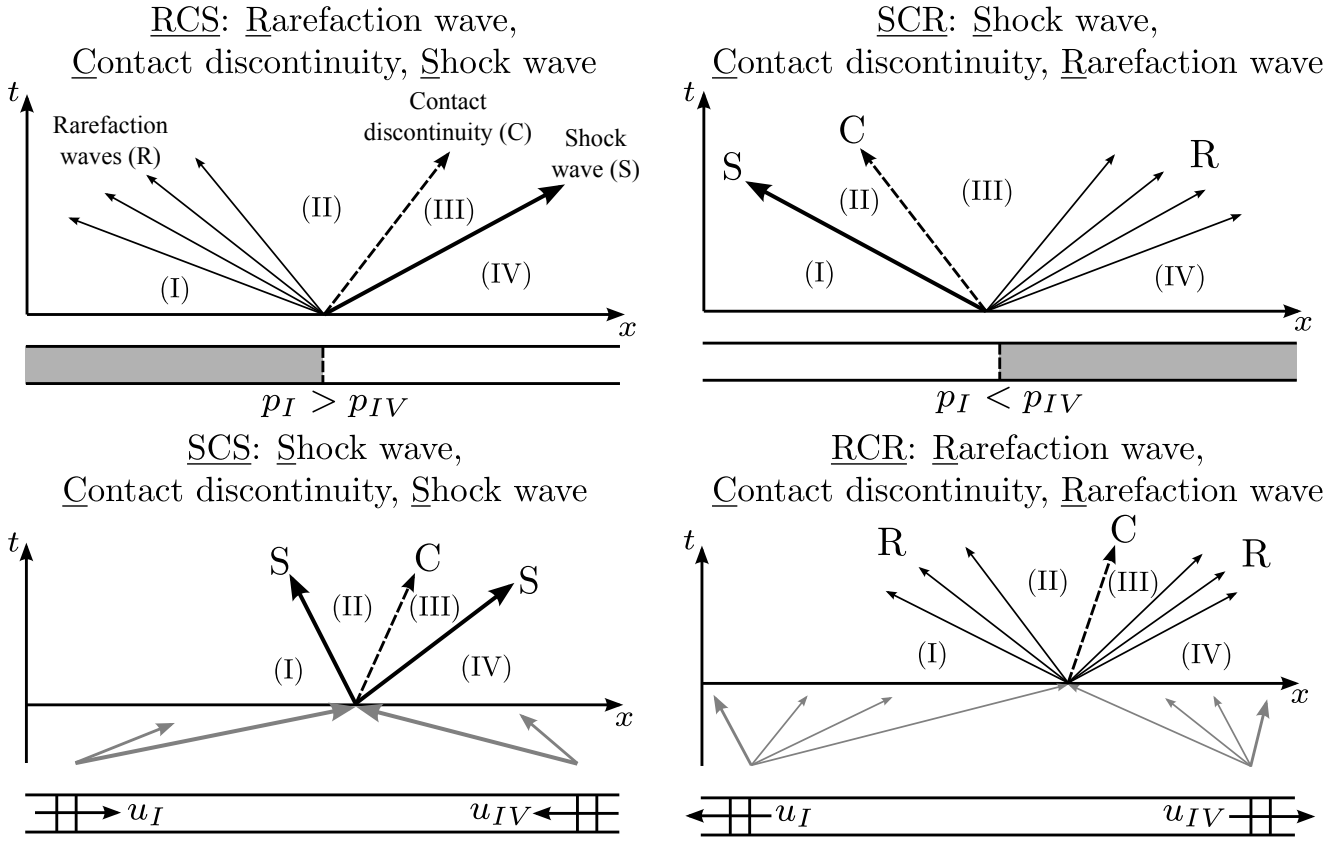


FIG. 2. Physical representations of all possible outcomes of the one-dimensional Riemann problem, with corresponding initial conditions. The shaded regions in the first two cases represent the regions with higher pressures; their outcome is a left-traveling rarefaction wave and a right-traveling shock (**RCS**), and vice-versa (**SCR**). The last two scenarios can be thought of being derived from two pistons that are moving towards and away from each other, with outcomes being the collision of two shock waves (**SCS**) or two rarefactions (**RCR**), respectively. S is a shock wave, R is a fan of rarefaction waves, C is a contact discontinuity, and the roman numerals refer to each state in between wavefronts.

$$C_K = \begin{cases} \frac{|p^i - p_K|}{|u_N^i - u_K|} & \text{if} \quad u_N^i \neq u_K \\ \rho_K a_K & \text{if} \quad u_N^i = u_K \end{cases} \tag{11}$$

then the new thermo-fluid-dynamic values are recomputed, and the new pressure is obtained:

$$p^{i+1} = p^i - (u_{II}^i - u_{III}^i)\frac{p^i - p^{i-1}}{(u_{II}^i - u_{III}^i) - (u_{II}^{i-1} - u_{III}^{i-1})} \tag{12}$$

the algorithm continue to iterate until $\left|u_{II}^i - u_{III}^i\right| < \varepsilon$ (we set $\varepsilon = 10^{-2}$).

III. PREVIOUS WORKS

A. Review of machine learning applied to the Riemann problem

Gyrya et al.[2] applied the Gaussian process and neural network algorithms to the Riemann problem. They first analyzed these models on a basic nonlinear problem like the quadratic equation, and then move on to the Riemann problem with ideal gases. They focused on a specific configuration of waves for the Riemann problem (left rarefaction waves and right shock wave). Their dataset was generated from two reference configurations $W_{\mathrm{I}}^{\mathrm{ref}}$ and $W_{\mathrm{IV}}^{\mathrm{ref}}$, where I and IV referred to the left and right state, as shown in Figure 3. All the other values are derived as $W_{\mathrm{I}} \in [W_{\mathrm{I}}^{\mathrm{ref}}(1-\varepsilon), W_{\mathrm{I}}^{\mathrm{ref}}(1+\varepsilon)]$ and $W_{\mathrm{IV}} \in [W_{\mathrm{IV}}^{\mathrm{ref}}(1-\varepsilon), W_{\mathrm{IV}}^{\mathrm{ref}}(1+\varepsilon)]$, where $\varepsilon = 0.9$. They used three different architectures of neural networks with two hidden layers. With both models, they used as input the initial states $\{\rho_{\mathrm{I}}, u_{\mathrm{I}}, p_{\mathrm{I}}, \rho_{\mathrm{IV}}, u_{\mathrm{IV}}, p_{\mathrm{IV}}\}$, and predicted from 1 to 9 outputs $\{p^*, u^*, \rho_{\mathrm{II}}^*, \rho_{\mathrm{III}}^*, S_{\mathrm{HL}}, S_{\mathrm{TL}}, S_{\mathrm{HR}}, S_{\mathrm{TR}}, S_{\mathrm{M}}\}$, where the star values denote middle states, and S denote the speeds at the head and tail of the left and right waves, and the speed of the contact discontinuity. They found that the neural networks are not sensitive to the number of layers but only to the number of nodes per layer, that both models decrease their accuracy with an increase in the number of outputs, and that the Gaussian process is usually more accurate than the neural networks but its training time is longer. They also analyzed the nonlinear equation for the stiffened EoS (for e.g., water under

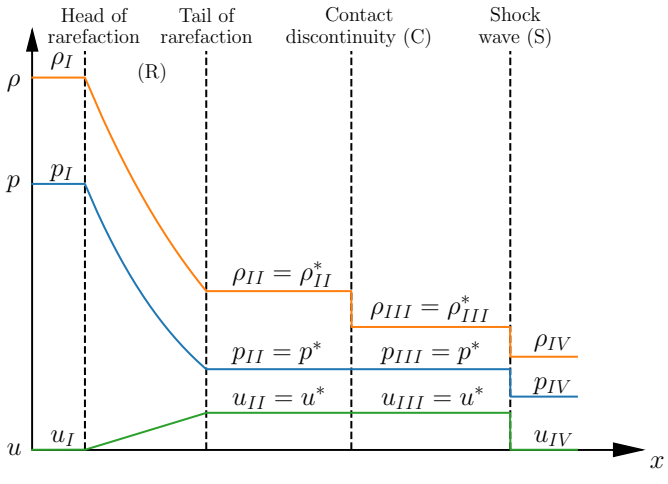


FIG. 3. Qualitative reconstruction of the **RCS** scenario (see Figure 2), used as a reference for the whole paper. The far left and right state variables are indicated respectively with subscripts *I* and *IV*. The intermediate states *II* and *III* share the same pressure and velocity, which are also referred to using the superscript *.

high pressure), with only $\{\gamma, p_0\}$ as inputs, and predicting p^*.

Magiera et al.[3] analyzed the Riemann problem for ideal gases on all possible wave configurations using constrain-aware neural networks and applied the model solutions to a front-capturing scheme. The neural network was composed of 7 fully connected layers with 70 neurons each. The constraints are Constraint-resolving layer methods (CRes) and Constraint-adapted loss method (CAL), which inform the neural networks of the physics of the phenomenon. Constraint-resolving layer methods (CRes) use the prediction of one value and the inputs to predict all the other values analytically. The constraint-adapted loss method (CAL) embed the physics laws in the loss function introducing a new term that measures the difference of the solution to respect the physical laws. The loss function value is the multiplication of this new value for a parameter λ, and its sums to the standard loss function. They tried these new methods (CRes and CAL with three different values of λ) on three different cases: cubic flux flow, isothermal two-phase flow, and Euler equations. In all cases, the CRes neural network has the best accuracy and is the best to reduce the physical error. Instead, CAL is similar to the standard neural network and sometimes worst. They showed that even though the neural networks do not improve the computational time on the front-capturing scheme, it has a big effect on the computational time of particle simulations, decreasing the computational time from 13 hours to 2 s.

B. Review of machine learning applied to CFD simulations

Computational Fluid Dynamics (CFD) methods are computationally intensive because the equations that describe compressible flows are not analytically solvable yet. Therefore, they need to be numerically integrated, and grid quality predominantly dictates the solution accuracy. For this reason, the scientific community produced several works on how to embed machine learning algorithms inside these codes to reduce their execution time. Mohan and Gaitonde[30] show the application of a particular Recurrent Neural Network: LSTM, coupled with Proper Orthogonal Decomposition (POD). Instead of using Proper Orthogonal Decomposition with Galerkin projection to obtain a reduced order model of the solution of a turbulent flow, they substitute the Galerkin projection with LSTM. LSTM achieves very good accuracy for a small variation of the Reynolds number in the field, and it can speed up the time considerably. Analogously, Xiao et al.[31] applied the same method with a Gaussian Process Regression to predict turbulent flow in an urban environment. They show that this method yields similar predictions to its high-fidelity counterpart while being six orders of magnitude faster.

Hanna et al.[32] applied artificial neural network and Random Forests to predict the error between coarse-grid and fine-grid CFD simulations. Coarsening the grid of a CFD method reduces the computational time of the method but increases the error induced by the grid. Predicting this error with machine learning algorithms allows them to correct it. They trained the algorithms on the results of normal and coarse-grid CFD codes and showed that the machine learning methods can correct the error of the coarse-grid CFD simulation. In a similar study, Parish and Duraisamy[33] show the improvement bring by the Gaussian Process and inverse modeling to deduce errors and data from high fidelity methods, and use them correct closure model of the turbulent flow.

Zhao et al.[34] developed a new framework called CFD-driven machine learning. Instead of training the machine learning algorithms on high-fidelity models they use gene-expression programming to calculate the stress tensor inside the RANS method. The reason for good performance of this new framework is because other data-driven approaches focus on the training on high-fidelity models to predict the stress tensor that cannot be directly inserted into RANS and can cause RANS predictions to have bad accuracy.

The approach of physics-informed neural networks is to combine physical laws with machine learning algorithms. Bode et al.[35] used this approach in Generative Adversarial Network that uses physics informed losses to compute the subgrid of a Large Eddy Simulation method (LES), and show that this approach gives good results in apriori and a posteriori tests. Eivazi et al.[36] used physics informed neural networks for incompressible turbulent flows. Their network considers only the data at the domain boundary to balance the loss of information in RANS equations. They show that this approach has excellent predictions for laminar flows, and for turbulent flows has an ℓ_2-norm error of around 3%.

IV. TRAINING DATASET FOR NEURAL NETWORK

We chose to investigate the Riemann problem for three different EoS: the ideal EoS for calorically imperfect gases[37] is used on gas with non-constant heat capacities, the Peng-Robinson EoS[38] that is used for gas near to the critical conditions, and Davis EoS[39] that is used for high explosives (in

TABLE I. List of equations of state (EoS), expressions for internal energy (e), heat capacities (c_v, c_p), closures for calorically imperfect gases, and speed sound (a) considered for each fluid model.

	Ideal Gas (IG-CI)	Supercritical Fluid (SF)	High Explosive (HE)
	$p = \rho RT$	$p = \frac{R_u T}{v_m - b} - \frac{a\alpha}{v_m^2 + 2bv_m - b^2}$	$p = p^s(\rho) + \rho\Gamma[e - e^s(\rho)]$
e	$e = \frac{p}{(\gamma-1)\rho}$	$e(T,\rho) = e^0(T) + \frac{a}{\sqrt{8}bm}\left(T\frac{\partial\alpha}{\partial T} - \alpha\right)\ln\left(\frac{m+(1+\sqrt{2}b\rho)}{m+(1-\sqrt{2}b\rho)}\right)$	$e = e^s(\rho) + \frac{p - p^s(\rho)}{\rho\Gamma}$
a	$a = \sqrt{\gamma(T)\frac{p}{\rho}}$	$a(T,\rho) = \sqrt{(\gamma(T,\rho)\left(\frac{\partial p}{\partial \rho}\right)_\rho}$	$a = \sqrt{\left(\frac{\partial p}{\partial \rho}\right)_e + \frac{p}{\rho^2}\left(\frac{\partial p}{\partial e}\right)_\rho}$
c_v	$c_v = c_{v_0}$ =const.		
$c_v(T)$	$\frac{c_v(T)}{c_{v_0}} = 1 + \ldots$ $+(\gamma-1)\left(\left(\frac{\Theta}{T}\right)^2 \frac{e^{\frac{\Theta}{T}}}{(e^{\frac{\Theta}{T}}-1)^2}\right)$	$c_v(T,\rho) = c_v^0(T) + \ldots$ $+\frac{aT}{\sqrt{8}bm}\frac{\partial^2\alpha}{\partial T^2}\ln\left(\frac{m+(1+\sqrt{2}b\rho)}{m+(1-\sqrt{2}b\rho)}\right)$	$c_v(T,\rho) = c_v^0\left(\frac{T}{T^s(\rho)}\right)^2$
c_p	$c_p = c_{p_0}$ =const.		
$c_p(T)$	$\frac{c_p(T)}{c_{p_0}} = 1 + \frac{\gamma-1}{\gamma}\left(\left(\frac{\Theta}{T}\right)^2 \frac{e^{\frac{\Theta}{T}}}{(e^{\frac{\Theta}{T}}-1)^2}\right)$	$c_p(T,\rho) = c_v(T,\rho) + \frac{T}{\rho^2}\frac{\left(\frac{\partial p}{\partial T}\right)_\rho^2}{\left(\frac{\partial p}{\partial \rho}\right)_T}$	$c_p(T,\rho) = c_v(T,\rho) + \frac{T}{\rho^2}\frac{\left(\frac{\partial p}{\partial T}\right)_\rho^2}{\left(\frac{\partial p}{\partial \rho}\right)_T}$

TABLE II. Range of values of thermo-fluid-dynamic variables in all states for the Riemann problem applied to each compressible fluid model considered. T_c and p_c are the critical temperature and pressure for the supercritical fluid (SF) model, and ρ_0 is the reference density for the High Explosive (HE) model.

	CI	SF	HE
p	$[10^{-6}, 20]$kPa	$[1.1, 1.6]p_c$	$[0.1, 40]$GPa
u	$[-100, 3000]\frac{m}{s}$	$[-100, 4500]\frac{m}{s}$	$[-100, 7000]\frac{m}{s}$
T	$[30, 1000]$K	$[1.1, 2.2]T_c$	n/a
ρ	n/a	n/a	$[0.1, 2]\rho_0$

our case we analyzed a situation with only reactants in case of Davis EoS). In addition to these thee EoS in the appendix B we present the results obtain on the ideal EoS that is used for an ideal gas. Table I reports the formulas for each EoS and its specific gas. Ideal EoS use air, Peng-Robinson EoS CO_2[40], and Davis EoS PBX-9502[39]. Each dataset has been generated, choosing the values of the initial states randomly in a specific range of values for each EoS, as shown in table II. After selecting the initial states, the solution of the Riemann problem is computed with the exact Riemann solvers, and if states II and III do not have their values in the selected range, the sample is excluded. The datasets and testing sets have respectively 100,000 and 20,000 samples with an equal number of samples for every possible waves configuration. For the models training, the datasets are split 80/20 between training and validation sets to avoid overfitting of the neural network.

Machine learning algorithms aim to predict values in a space of dimension m ($\mathbb{R}^m$), given an input in the space $\mathbb{R}^n$. Therefore, machine learning can be seen as a function $f(x) : \mathbb{R}^n \to \mathbb{R}^m$, where both the input and the output can have a discretized (classification problem) or continuous (regression problem) range of values. These algorithms can solve the Riemann problem predicting one value–valid only for u^* and p^*–of the middle states, or all of them. Therefore, they have to solve a regression problem. Among the algorithms used to solve regression problems–Ordinary Least Squares Regression (OLSR), Linear Regression, Logistic Regression, Multivariate Adaptive Regression Splines (MARS)–we chose to use a specific class of feedforward neural network called Multilayer Perceptrons (MLP). The advantages of this network are very good results on the prediction of complex nonlinear problems, and that provides quick predictions after the training. On the other end, it cannot obtain good results on data that are outside the range of the training data, and it requires a long time to be trained. There are other kinds of neural networks such as convolutional neural networks and recurrent neural networks, that are usually used in physics and finance, because they achieve good results at predicting values that depend on the previous configurations–which in physics phenomena are usually time marching problems. We exclude these neural networks because the exact solution in time of the Riemann problem is more complex than the single value solution. Therefore, the model is less accurate and slower. Moreover, one of the middle values is enough to reconstruct the entire solution in time, and hence also have the time evolution of the problem.

An MLP consists of a number of neurons grouped into layers. For fully-connected networks, which we will use here, each neuron in a layer is connected to every neuron in the next layer. The first layer is called the input layer, and the number of neurons is equal to the dimension of the input space because their values are the input values. The last layer is called the output space layer, and the number of neurons is equal to the

dimension of the output because their values are the predicted values of the neural network. All the layers between these two are called hidden layers, and have a variable number of neurons. Each neuron j–except for the input layer–receive the input value x_i from the output of the neurons i of the previous layer and compute its output:

$$y_j = \phi\left(\sum_{i=1}^{n} w_{ji}x_i + b_j\right) \tag{13}$$

where w_{ji} is the weight that the neuron j of the current layer associate to the neuron i of the previous layer; b_j is a constant value called bias, associated with neuron j; n is the number of neurons in the previous layer; and $\phi()$ is the activation function. The activation function is usually a nonlinear function for the neurons of the hidden layers and has an essential role in the functioning of the neural network. It is required because through its nonlinearity, it is possible to compute a gradient of the loss function and update the weights of the neurons to increase the accuracy of the model–this process is called back-propagation.

To build and train the neural networks we used the Python library Tensorflow[41]. We tested four different activation functions: ReLu ($\phi(x) = \max(0,x)$), sigmoid ($\phi(x) = \frac{1}{1+e^{-x}}$), tanh ($\phi(x) = \tanh(x)$), and linear ($\phi(x) = x$) that is used only in the output layer to allow the prediction of negative values (for example, u^*). To update the weights of the networks, we tested three different optimizers: Stochastic gradient descent (SGD, it uses the gradient and a learning rate to update w, b and reduce the error), RMSprop (it is similar to SGD, but it has a variable learning rate), and Adam[42] (it is similar to RMSprop and also use a cumulative history of the gradient). Four different loss functions were used together with the optimizers: Mean Absolute Percentage Error (MAPE $\left|y_i - F_{w,b}(x_i)\right| / |y_i|$), Mean Absolute Error (MAE $\left|y_i - F_{w,b}(x_i)\right|$), Mean Squared Error (MSE $(y_i - F_{w,b}(x_i))^2$), and Huber loss function (that compute the error as linear if $\left|y_i - F_{w,b}(x_i)\right| > 1$, and squared in the other case), where y_i is the correct output and $F_{w,b}(x_i)$ is the prediction of the model.

The SGD optimizer was found to be the worst of all the optimizers with accuracy that were ten times worst than the others. For this reason, we chose not to move forward with its analysis. For all cases, we analyzed the performance of each optimizer with each loss function, on a neural network with architecture 512-256-128-64-32 (each number is the number of nodes in each hidden layer). The best three models mentioned above were analyzed more in-depth on five different architecture (1028-512-256-128-64, 512-512-256-128-64, 512-256-128-64-32, 512-256-128-64, 256-128-64-32). The data normalization has been obtained by dividing each value of the states for the highest value of the specific variable. Even though the degree of freedom of the neural network can seem to be too high, we checked with fewer neurons and layers, and the accuracy is worst. We also checked possible overfitting of the results but also with this high number of neurons, the network is still able to generalize. Moreover, also the network used by Magiera et al.[3] presented a high number of neurons.

V. RESULTS

For all cases analyzed, Adam optimizer always achieves the best accuracy. To compare the loss functions we use the MAPE after training and the best varies among all EoS and cases analyzed. ReLu activation function is always the best, both for normalized and non-normalized data. The best architecture changes for different EoS or if the data are normalized or not. The only exception is for the calorically imperfect gases, in which 1028-512-256-128-64 is the best for normalized and non-normalized data. For calorically imperfect gases, the neural network has the best accuracy if the thermodynamic variable used in addition to the pressure is the speed of sound, and the density is always the worst. Instead, high explosives and supercritical fluids obtained the best results with the density and the worst with temperature. The only exception is for the standalone neural networks in the case of high explosives, where the speed of sound allows better accuracy than the density, even though the difference is mild.

For each EoS analyzed, we report the accuracy obtained by two methods: the neural network (Figure 4) predicting one of the middle state values (p^*) and the other values computed with the Euler equations (7-10), the neural network predicting all the middle values (p^*, u^*, ρ_{II}, ρ_{III}). In appendix B we report the results obtained with the ideal EoS, and in appendix C, we report the accuracy obtained by a physics-informed neural network trained with unsupervised learning in the case of ideal gas. To compare the neural networks and the approximate Riemann solver, we chose the best network between all the ones trained.

Table III reports the accuracy of the variables in states II and III for calorically imperfect gases. The trend on the accuracy follows a similar pattern among all methods for each waves configuration, in which the case with two rarefaction waves has the worst accuracy. Considering that to obtain the values after an expansion for a non-ideal EoS requires the solution of an integral–differently from the ideal EoS in which the integral has an analytical solution–therefore makes it more difficult to approximate. Predicting only the pressure and then computing the other values is the best solution for each configuration, especially for the velocity–where the overall accuracy is ten times better than the one computed by the other method. As expected, the average time that each method required to solve one sample for the Riemann solver for general EoS is the slowest and has the highest time for the two rarefaction waves. Considering that this configuration has to solve two differential equations. Predicting all the values with the neural network is faster than combining neural network and Euler equation, especially for the two rarefaction waves case.

For Davis EoS, the non-normalized dataset brings the accuracy to around 30% compared to the 1% of the normalized data. This result is due to the wider range of the thermodynamic variables, which is not present in the calorically imperfect gas case. Table IV shows that the neural network with the Euler equations can be up to ten times more accurate than the standalone neural network, except for the pressure–which is the value predicted by the network. The exact solver takes more time to predict cases with one rarefaction and one shock

Accepted to Phys. Fluids 10.1063/5.0123466

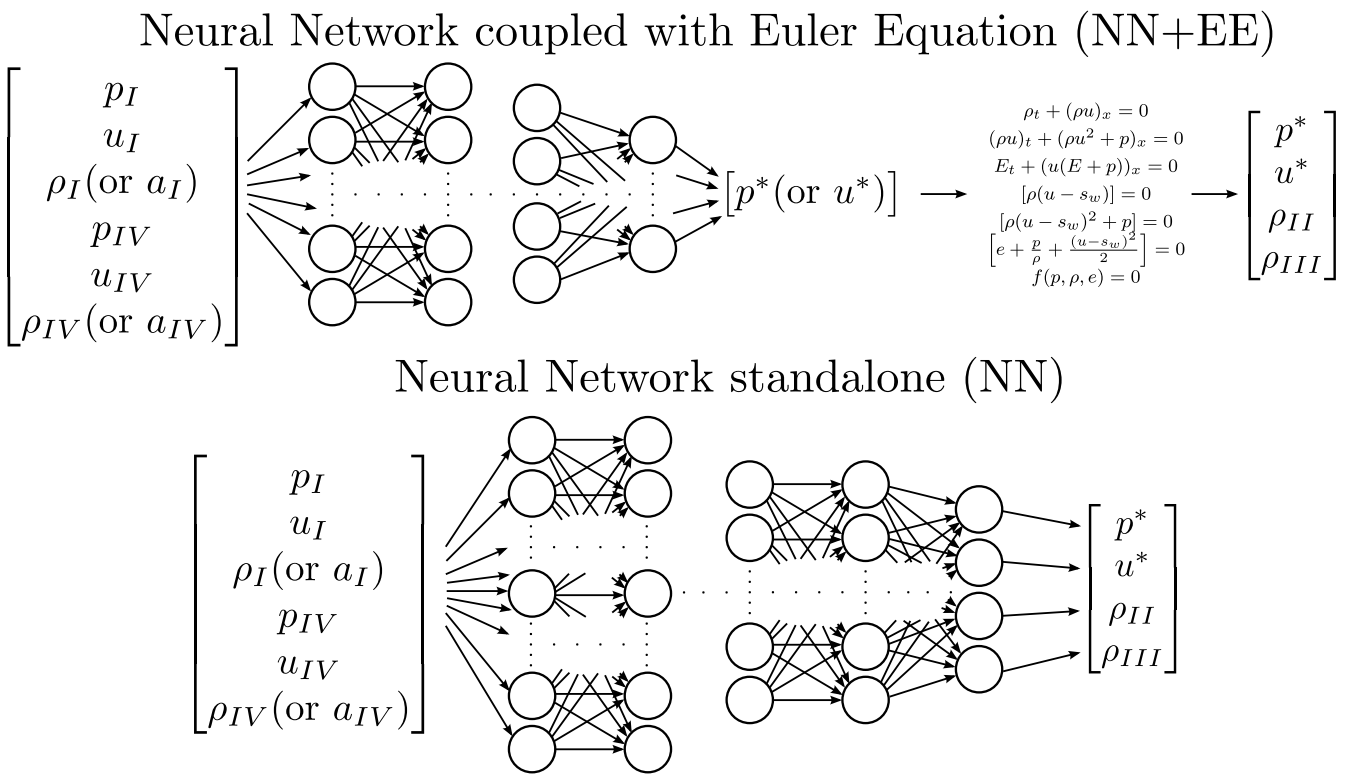


FIG. 4. Top: illustration of a neural network with Euler equations embedded in it (NN+EE); the Euler equations compute the values of states *II* and *III* with the provided array of input values (where ρ may be picked instead of a depending on which one yields better accuracy) and the predicted value p^* for real fluids, and u^* for ideal gas. Bottom: illustration of a neural network that computes all the variables of the middle states with its dense layers (NN).

TABLE III. Mean absolute percentage error on predictions for u^*, p^*, ρ_{II}, ρ_{III}, and average time to solve one sample from different methods including: neural network coupled with Euler equations (NN+EE) and standalone neural network (NN) for calorically imperfect gases (CI).

	ε_{u^*}		ε_{p^*}		$\varepsilon_{\rho_{II}}$		$\varepsilon_{\rho_{III}}$		time		
	NN+EE	NN	NN+EE	NN	NN+EE	NN	NN+EE	NN	Exact RS for general EoS	NN+EE	NN
RCR	0.069%	0.651%	0.457%	1.092%	0.359%	1.209%	0.360%	1.259%	94.894 ms	1.024 ms	27.2 μs
RCS	0.038%	0.899%	0.097%	0.258%	0.075%	0.416%	0.068%	0.577%	76.540 ms	190.7 μs	27.0 μs
SCR	0.073%	2.376%	0.087%	0.275%	0.063%	0.527%	0.069%	0.428%	78.831 ms	169.3 μs	26.9 μs
SCS	0.025%	0.555%	0.125%	0.358%	0.089%	0.619%	0.084%	0.692%	72.888 ms	42.7 μs	27.3 μs
Overall	0.051%	1.120%	0.191%	0.495%	0.147%	0.693%	0.145%	0.739%	76.589 ms	354.5 μs	18.2 μs

wave than two rarefaction waves, differently from the neural network and the Euler equations. Therefore, we can conclude that the case with rarefaction and shock waves requires more iterations to be solved, because otherwise the case with two rarefaction waves should be the slowest because it needs to solve two integrals.

In the case of supercritical fluids, the results for normalized and non-normalized data are similar to the case of high explosives. Table V shows that the neural network combined with the Euler equations for general EoS is the method with the best accuracy, and the solver can further enhance the accuracy of the neural network. As in the previous cases, the exact Riemann solver for general EoS is the slowest and the standalone neural network the fastest.

VI. NEURAL-NETWORK-BASED RIEMANN SOLVERS EMBEDDED IN A CFD SOLVER

We compare the results of CFD simulations for each kind of fluid in a shock tube (initial velocity equal to zero $u_I = 0$ and $u_{IV} = 0$) for a configuration of rarefaction wave to the left and shock wave to the right, and a specific moment in time. The code is 1D, with Rusanov flux (described in appendix D 2) reconstruction between the cells, the most used flux scheme for real fluids, because the others (HLL and HLLC) are derived on

TABLE IV. Mean absolute percentage error on u^*, p^*, ρ_{II}, ρ_{III}, and average time to solve one sample from different methods including: neural network coupled with Euler equations (NN+EE) and standalone neural network (NN) for the PBX-9502 (HE).

	ε_{u^*}		ε_{p^*}		$\varepsilon_{\rho_{II}}$		$\varepsilon_{\rho_{III}}$		time		
	NN+EE	NN	NN+EE	NN	NN+EE	NN	NN+EE	NN	Exact RS for general EoS	NN+EE	NN
RCR	0.067%	0.267%	0.161%	0.260%	0.081%	0.271%	0.100%	0.276%	12.56 s	1.27 s	28.5 μs
RCS	0.044%	0.127%	0.149%	0.187%	0.070%	0.229%	0.074%	0.234%	38.56 s	0.579 s	28.0 μs
SCR	0.304%	1.699%	0.124%	0.187%	0.047%	0.235%	0.061%	0.239%	34.31 s	0.579 s	28.5 μs
SCS	0.048%	0.220%	0.125%	0.139%	0.050%	0.199%	0.046%	0.183%	59.41 ms	69 μs	27.9 μs
Overall	0.116%	0.578%	0.140%	0.193%	0.062%	0.233%	0.070%	0.233%	21.36 s	0.606 s	19.3 μs

TABLE V. Mean absolute percentage error on predicted u^*, p^*, ρ_{II}, ρ_{III}, and average time to solve one sample from different methods including: neural network coupled with Euler equations (NN+EE) and standalone neural network (NN) for supercritical fluid (SF).

	ε_{u^*}		ε_{p^*}		$\varepsilon_{\rho_{II}}$		$\varepsilon_{\rho_{III}}$		time		
	NN+EE	NN	NN+EE	NN	NN+EE	NN	NN+EE	NN	Exact RS for general EoS	NN+EE	NN
RCR	0.051%	0.332%	0.070%	0.082%	0.060%	0.148%	0.060%	0.152%	1.113 s	630 ms	25.0 μs
RCS	0.085%	1.632%	0.060%	0.069%	0.045%	0.127%	0.049%	0.127%	2.691 s	390 ms	25.2 μs
SCR	0.041%	0.433%	0.063%	0.069%	0.051%	0.124%	0.046%	0.116%	2.720 s	392 ms	25.1 μs
SCS	0.006%	0.279%	0.051%	0.064%	0.041%	0.131%	0.041%	0.138%	1.772 ms	422 μs	25.7 μs
Overall	0.046%	0.669%	0.061%	0.071%	0.049%	0.133%	0.049%	0.134%	1.612 s	472 ms	16.4 μs

the assumptions of ideal gases and to be extended to real fluids can required some iterative procedure that can make them slow. For investigation on 2D problem we refer the reader to Wang[4], who used a similar approach to solve a 2D case with two expansions waves and two contact discontinuities. We compare all previous cases except for calorically imperfect gases because of their small difference in accuracy from the ideal gases, and in time per iteration from the real fluids. All results are obtained with Runge-Kutta 4 method for time advancement, first order in space, CFL= 0.2, and 100 grid points, unless otherwise specified. Usually real fluids CFD with shock dominated flows rely on first order accuracy. But to have a better understanding of the neural networks results and stability in a CFD solver, we run some cases with the same condition as before and order 4. The numerical scheme is based on flux reconstruction and the shock capturing used is the one developed by Haga and Kawai[43].

In all cases, we notice that when the neural network is used to solve the Riemann problem, it causes several oscillations in the constant part of the simulation. The small difference in the prediction of the neural network from the two constant input values causes these oscillations. When the network is combined with the Euler equations, the error is reduced on the values that the network does not predict, reducing the oscillations. However, for the standalone neural network, the numerical method amplified them. For this reason, in the case of the standalone neural network, we chose to use it to solve the Riemann problem only if the difference in pressure and velocity of the initial states are higher than specific values. In all other cases, the Riemann problem is solved with the linearized Riemann solver (A5). In our case we set these values

TABLE VI. Average time required to process one iteration of the numerical method of each method used to resolve the flux at the interface for the PBX-9502 (HE).

	Time per iteration
Rusanov flux	0.177 s
Exact RS	134 s
NN+EE	0.615 s
NN	0.215 s

as the initial difference in pressure and velocity multiplied by the network accuracy for the specific output value. We noticed that most of the times, they can be lower than the values that we chose, except when the initial conditions are near the boundary of the training dataset; in that case, they should be higher. It is also important to add, that classic low-order finite-volume CFD solvers use the solution to the Riemann problem to estimate the numerical flux in between computational cells and then time-step accordingly. Assuming that the exact solution to the Riemann problem is used as a numerical flux, this does not guarantee that the CFD solver will yield the exact solution when used to solve the Riemann problem itself (discretized over multiple cells).

In the case of PBX-9502, the initial condition of the shock tube problem was taken as $P_I = 30$ GPa, $\rho_I = \rho_0$, $P_{IV} = 5$ GPa, and $\rho_{IV} = \frac{\rho_0}{2}$. Figure 5 shows the results of all the methods at $t = 0.047$ ms. It is possible to see that Rusanov flux has a smoother rarefaction and shock wave. In addition, in this case, it also has a higher error in estimating the values of the middle state. The Riemann solver and the neural networks

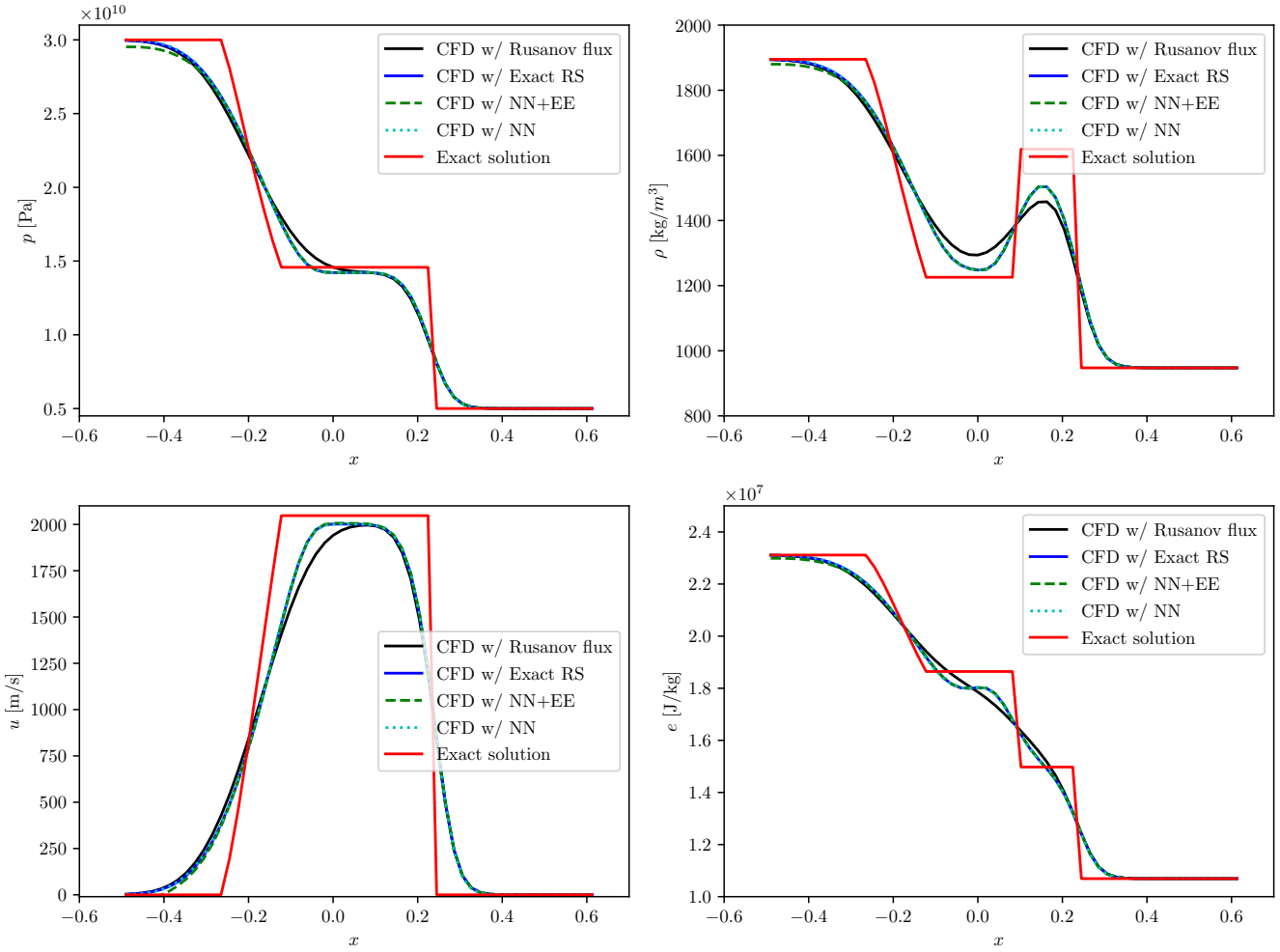


FIG. 5. Numerical reconstruction order 0 of shock tube problem for the PBX-9502 at time $t = 0.047$ ms, and initial pressures $P_I = 30$ GPa and $P_{IV} = 5$ GPa.

have almost similar results. The only difference is for the network with the Euler equations, in which case the left pressure drops from the initial value. This can be explained by the error in the neural network predictions, which has more influence when the initial states are the same, thus forcing their states to be slightly off from each other. The standalone neural network does not have this drop, and hence, it yields more accurate results than the one coupled with Euler equation because the first one is deactivated, i.e. it provides an update to the flux in between cells with the Rusanov flux, when cell-to-cell variations are small. Therefore, in the parts where the correct solution is approximately flat, the standalone neural network outperforms the method based on the Euler equations. Table VI reports the average time per iteration. Rusanov flux is the fastest method, and both neural networks are three orders of magnitude faster than the exact Riemann solver.

For supercritical fluids, the initial condition of the shock tube problem denote: $P_I = 1.5\ p_c$, $\rho_I = 231.74837 \frac{\text{kg}}{\text{m}^3}$, $P_{IV} = 1.15 p_c$, and $\rho_{IV} = 70.62419 \frac{\text{kg}}{\text{m}^3}$. Figure 6 shows the results of each method at $t = 0.966$ ms. Differently from the high explosives, for supercritical fluids, all the methods reached a middle state pressure higher than the correct one, and Rusanov flux is the one with the highest. Instead the neural network numerical flux estimate at the faces yields a more accurate solution to the overall Riemann problem than when using the exact Riemann solver itself to estimate the numerical flux in between cells. The small error made by the neural network in the prediction of the middle state of the Riemann problem used for the numerical flux estimate at each face unexpectedly works

TABLE VII. Average time required to process one iteration of the numerical method of each method used to resolve the flux at the interface for supercritical fluid (SF).

	Time per iteration
Rusanov flux	0.293 s
Exact RS	204 s
NN+EE	0.6 s
NN	0.407 s

Accepted to Phys. Fluids 10.1063/5.0123466

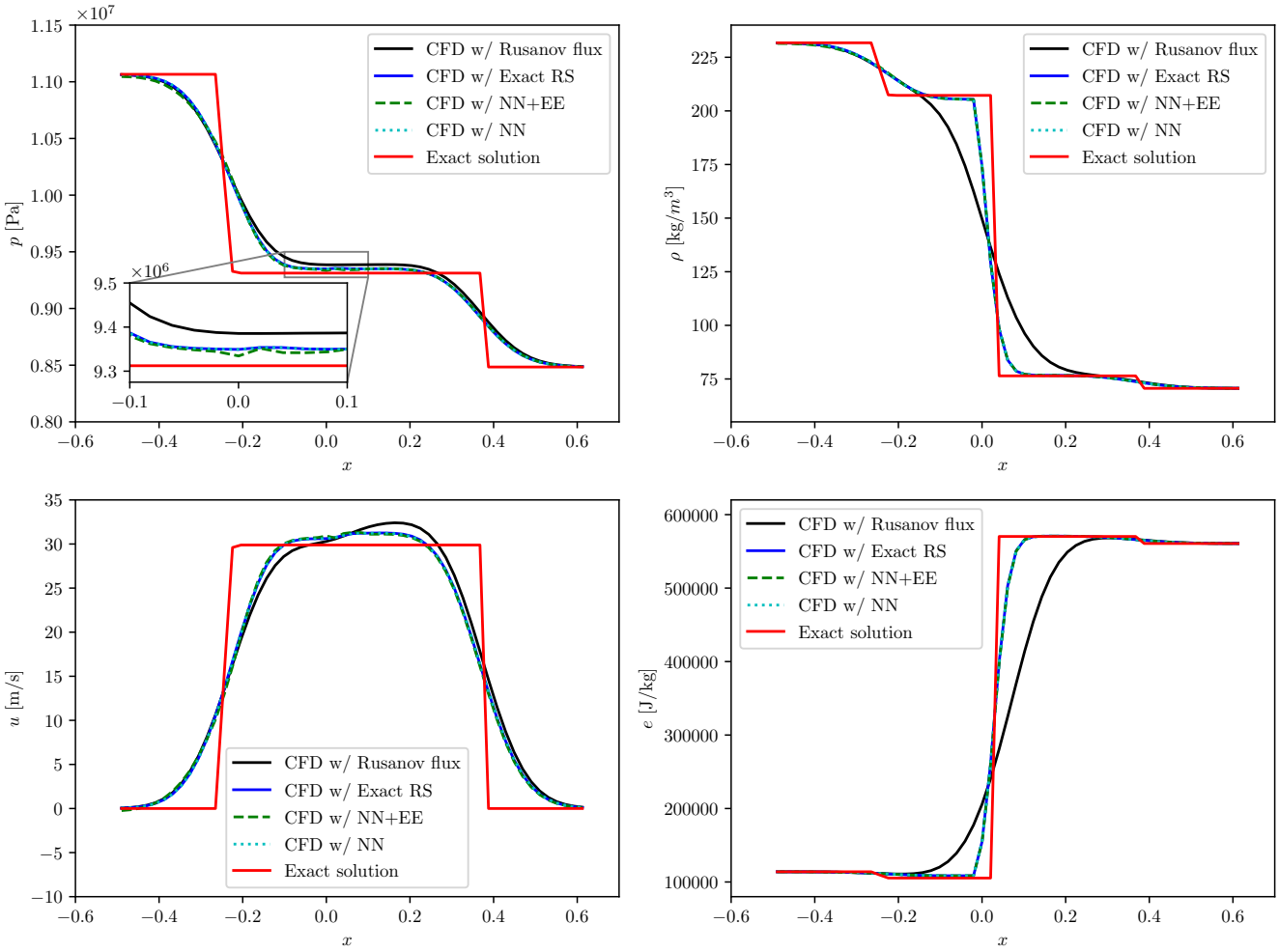


FIG. 6. Numerical reconstruction of shock tube problem for supercritical fluid at time $t = 0.966$ ms, and initial pressures $P_I = 1.5\ p_c$ and $P_{IV} = 1.15\ p_c$.

in the direction of improving the CFD prediction of the overall Riemann problem itself. These results shows that if a neural network is specifically trained to optimize the prediction of the numerical flux in a CFD solver (not shown here) it may obtain better results than an exact Riemann solver. Both networks show small oscillation in the initial discontinuity point. Table VII reports the average iteration time, that have a similar trend to the case of high explosives.

Figure 7 shows the results of the shock tube obtained with numerical scheme order 4, fir supercritical fluids. It is possible to see that in this case the difference among all methods in not as big as for order 1. The biggest difference is for at the contact discontinuity of density, where Rusanov flux approach has more oscillations than the others. The time per iteration of each method has a similar trend as for order 1. In case of high explosives and calorically imperfect gases the results are similar, therefore we can conclude that for high order the advantage of the neural networks start to decrease, because the approximate solver has accurate enough results and are faster than the networks. This is due to the fact that with high order the interfaces of the mesh has a less important role than for order 1.

A. Grid convergence comparison

To give a better comparison between the usual flux solver used in CFD for real fluids (Rusanov flux) and the neural network used in its place, we move forward with a grid converge analysis. We chose to investigate the case of high explosives because is the case were the differences are more visible. Figure 8 shows the results obtained with Rusanov flux and the standalone neural network on 3 different grid, with 100, 500, and 1000 points. We chose to not use the neural network with Euler equations and the exact Riemann solver because they are similar to the standalone neural network. The plots show that the neural network has a faster converge than the Rusanov flux, and there it requires less grid points that can balance the computational time lost in the computation.

Accepted to Phys. Fluids 10.1063/5.0123466

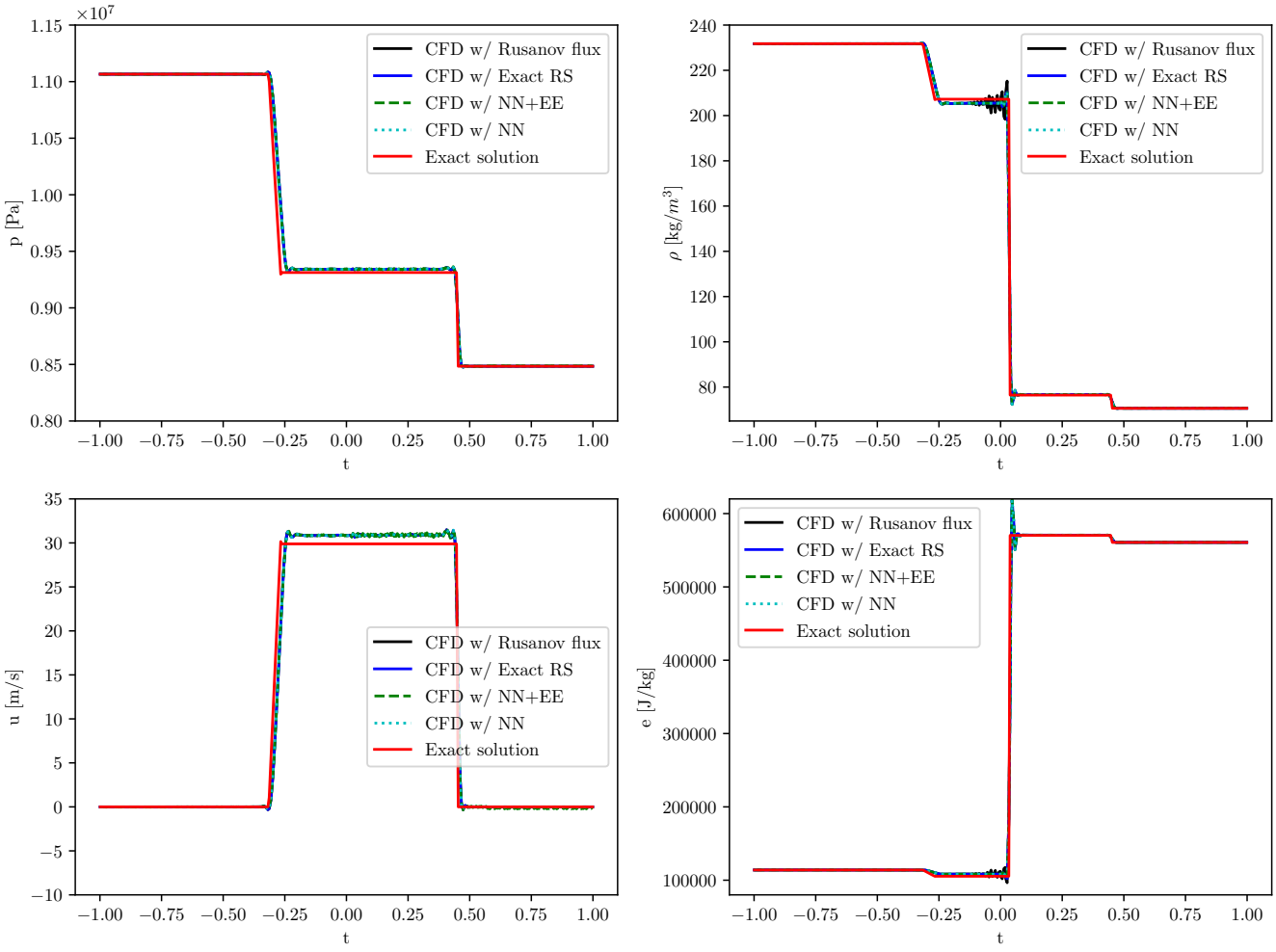


FIG. 7. Numerical reconstruction, order 4, of shock tube problem for supercritical fluid at time $t = 1.15$ ms, and initial pressures $P_I = 1.5\ p_c$ and $P_{IV} = 1.15\ p_c$.

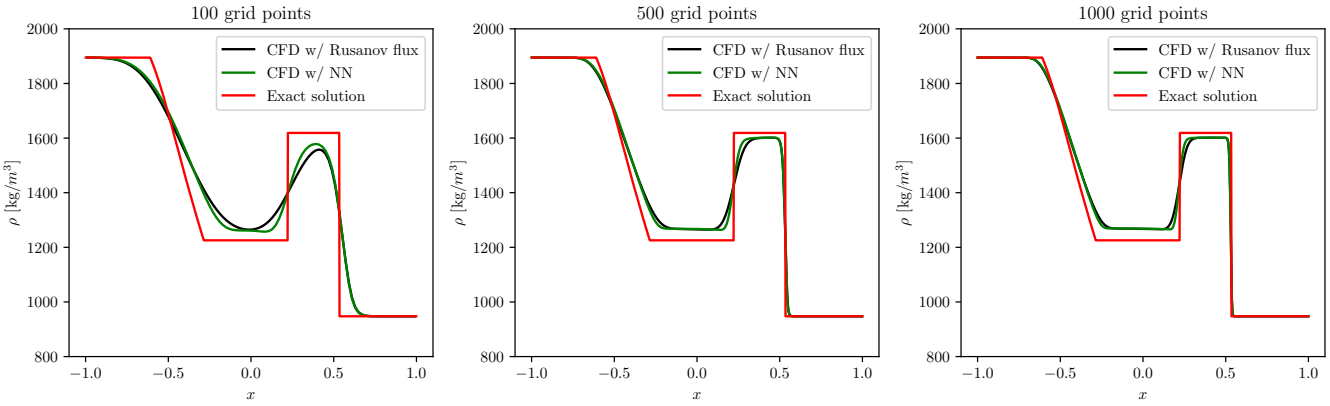


FIG. 8. Grind converge analysis for PBX-9502 (HE), comparison between Rusanov flux and standalone neural network for 100, 500, and 1000 grid points, and physical time $t = 0.108$ ms.

VII. CONCLUSION

In this paper, we showed the advantages that neural network, and combination of neural network and Euler equations, can bring to CFD simulations for real fluids. The neural networks are used to solve the Riemann problem at the

faces between grid cells in a 1D simulation, and the results are compared with Rusanov flux resolution methods and the exact Riemann solver solutions. We analyzed the neural network for three different equation of state: calorically imperfect ideal EoS (air), Peng-Robinson (CO_2), and Davis (PBX-9502).

- The CFD simulations results with the neural networks can match that of the exact Riemann solver, giving a better prediction than using the fluxes resolution method, and being more than 3 order of magnitude faster than the exact Riemann solver.
- The neural networks are half as fast as Rusanov flux, but they have a faster grid convergence than the approximate methods, and hence require less grid points and evaluations.
- In case of high order scheme the advantege of the neural networks are lower compare to the Rusanov flux, because the solution of the Riemann problem at the interface is less important than for order 1.
- The results suggest that future work may look at training a neural network for optimizing the overall accuracy of a CFD calculation, rather than the accuracy of its prediction of the middle state of the Riemann problem.
- If the neural network is coupled with the Euler equations its solution of the Riemann problem leads to more stable results than having a standalone neural network.

From the last point, we can conclude that to have a safe deployment of the model in CFD solver it is important to use a physics-informed neural network, or to avoid extreme cases–as constants initial conditions–which can cause unwanted oscillations.

To our knowledge, there is no approximate Riemann solver for non-ideal EoS that we can use to compare the neural network results, and hence we compare the results between the neural network configurations. For all EoS, the combination of neural network and Euler equations is always better than the standalone neural network. The standalone neural network is always the fastest, with an average time of 28 μs. For neural network and exact Riemann solver, the time to compute the solution mostly depends on the time of the numerical computation of the analytical part, but it can be two to ten times faster than the standalone exact Riemann solver.

Even though neural networks are faster than an accurate solver and more accurate than an approximate one, they still have limitations. Especially when they are used in CFD code, the low accuracy for prediction outside the range of the training dataset can blow up the simulation. Therefore, it is important to know apriori the range of values of the simulation and to remain far from these values during the simulation.

SUPPLEMENTARY MATERIAL

Supplementary materials to the paper contain all the datasets and testsets used to train the neural networks and the codes that are most significant for the study. Specifically, the codes used to train the neural networks and the ones used for the CFD solver with the exact Riemann solver and the NN+EE for PBX-9502.

ACKNOWLEDGMENTS

We acknowledge support through ONR Grant No. N00014-21-1-2475 with Dr. Eric Marineau as Program Manager.

Appendix A: Ideal gases Riemann solvers

1. Exact Riemann solver

The ideal EoS is the only case in which it is possible to obtain an analytical solution of the integral that describes the evolution of the rarefaction wave, and the Rankine-Hugoniot (4) equations can be reduced to simpler equations. Therefore, the Riemann solver for the ideal EoS is only composed of analytical formulas. Since Godunov[44,45], several Riemann solvers have been developed, and Pike[46] developed the fastest of these solvers. We chose to use the solver of Gottlieb and Groth[47], which is the second fastest solver (47% slower than the fastest), and simpler to implement. We did not choose the fastest solver because our analysis focuses on improving the computational time for non-ideal EoS.

Gottlieb and Groth[47] solver use as iteration variable u^* and assess the accuracy of the solution with the difference between the pressure of the two middle states. The utilization of u^* as iteration variable leads to less algebraically complex computations. The first iteration starts predicting the velocity as the result of two isentropic waves:

$$u^0 = \frac{zR_{2I} - R_{1IV}}{\delta(1+z)} \tag{A1}$$

where $z = \frac{a_{IV}}{a_I}\left(\frac{p_I}{p_{IV}}\right)^{\frac{\gamma-1}{2\gamma}}$, $\delta = \frac{\gamma-1}{2}$, $R_{2I} = a_I + \delta u_I$ is the right invariant of state I, and $R_{1IV} = a_{IV} + \delta u_{IV}$ is the left invariant of state IV. From here on, we pose $u^0 = u^i$ because this part of the code is repeated. If $u^i \geq u_I$ or $u^i \leq u_{IV}$ the values for states II and III are computed considering a rarefaction wave:

$$\begin{aligned} a_N^i &= a_K^i \mp \delta(u^i - u_K) \\ p_N^i &= p_K\left(\frac{a_N^k}{a_K}\right)^{\frac{2\gamma}{\gamma-1}} \\ \rho_N^i &= \mp\gamma\frac{p_N^i}{a_N^k} \end{aligned} \tag{A2}$$

and if $u^i < u_I$ or $u^i > u_{IV}$ is a shock wave that propagate to that direction:

$$
\begin{aligned}
M_N^i &= \mp\frac{\gamma+1}{4}\frac{u^i-u_K}{a_K}+\sqrt{1+\left(\frac{\gamma+1}{4}\frac{u^i-u_K}{a_K}\right)^2}\\
p_N^i &= p_K\left[1+\frac{2\gamma}{\gamma+1}({M_N^i}^2-1)\right]\\
\dot{p}_N^i &= \mp 2\gamma\frac{p_K}{a_K}\frac{{M_N^i}^3}{{M_N^i}^2+1}\\
a_N^i &= a_K\sqrt{\frac{\gamma+1+(\gamma-1)\frac{p_N^i}{p_K}}{\gamma+1+(\gamma-1)\frac{p_K}{p_N^i}}}
\end{aligned}
\tag{A3}
$$

where $N=\{II,III\}$, $K=\{I,IV\}$, M_N^i is the Mach number in states I or IV relative to the shock, $\dot{p}_N^k$ is the derivative of the pressure with respect to u, and the upper sign is from states I to II and the lower from IV to III. If $\left|p_{II}^i-p_{III}^i\right|<\varepsilon$ (in our case $\varepsilon=10^{-6}$), the result is the solution of the Riemann problem. If not, u^i is updated with the Newton-Raphson method:

$$
u^{i+1}=u^i-\frac{p_{II}^i-p_{III}^i}{\dot{p}_{II}^i-\dot{p}_{III}^i} \tag{A4}
$$

these values are computed in (A2, A3), and the iterations continue.

2. Adaptive Riemann solver

Toro[1] proposed a non iterative algorithm to solve the Riemann problem. The first step is to give an approximation of p^* and u^* with a linearized Riemann solver that use simple algebraic equations to find the values in the middle states of the problem:

$$
\begin{aligned}
u^* &= \frac{1}{2}(u_I+u_{IV})-\frac{p_{IV}-p_I}{2\bar{\rho}\bar{a}}\\
p^* &= \frac{1}{2}(p_I+p_{IV})-\frac{1}{2}\bar{\rho}\bar{a}(u_{IV}-u_I)\\
\rho_{II} &= \rho_I+\frac{\bar{\rho}}{\bar{a}}(u_I-u^*)\\
\rho_{III} &= \rho_{IV}+\frac{\bar{\rho}}{\bar{a}}(u^*-u_{IV})
\end{aligned}
\tag{A5}
$$

where $\bar{a}$ and $\bar{\rho}$ are the averaged value of the speed of sound and the density, which could be an arithmetic or geometric average. We found the arithmetic average to be better than geometric average, and decided to use it in our calculations ahead, and hence $\bar{a}=\frac{a_I+a_{IV}}{2}$ and $\bar{\rho}=\frac{\rho_I+\rho_{IV}}{2}$. The author shows that using a particular average different from the arithmetic or the geometric makes it possible to have the analytical solution for the two rarefaction waves configuration.

He defined $p_{min}=\min(p_I,p_{IV})$, $p_{max}=\max(p_I,p_{IV})$, and $p^*=p^0$. If p^0 is between p_{min} and p_{max}, the configuration should have one shock and one rarefaction wave, and the solution remains the one found with the linearized equations (A5). If $p_0\leq p_{min}$, the solution should have two rarefaction waves:

$$
\begin{aligned}
p^* &= \left[\frac{a_I+a_{IV}-\Delta u\frac{\gamma-1}{2}}{\frac{a_I}{p_I^{\frac{\gamma-1}{2\gamma}}}+\frac{a_{IV}}{p_{IV}^{\frac{\gamma-1}{2\gamma}}}}\right]^{\frac{2\gamma}{\gamma-1}}\\
u^* &= \frac{1}{2}(u_I+u_{IV})+\frac{1}{2}[f(p^*,U_{IV})-f(p^*,U_I)]\\
\rho_{II} &= \rho_I\left(\frac{p^*}{p_I}\right)^{\frac{1}{\gamma}}\\
\rho_{III} &= \rho_{IV}\left(\frac{p^*}{p_{IV}}\right)^{\frac{1}{\gamma}}
\end{aligned}
\tag{A6}
$$

where $\Delta u=u_{IV}-u_I$, and $f(p^*,U_K)=\frac{2a_K}{\gamma-1}\left[\left(\frac{p^*}{p_K}\right)^{\frac{\gamma-1}{2\gamma}}-1\right]$.

Instead, in the case of $p_0\geq p_{max}$ a two shock waves configuration is considered:

$$
\begin{aligned}
p^* &= \frac{p_I g(p_0,U_I)+p_{IV}g(p_0,U_{IV})-\Delta u}{g(p_0,U_I)+g(p_0,U_{IV})}\\
u^* &= \frac{1}{2}(u_I+u_{IV})+\frac{1}{2}[(p^*-p_{IV})g(p^*,U_{IV})-(p^*-p_I)g(p^*,U_I)]\\
\rho_{II} &= \left[\frac{\frac{p^*}{p_I}+\frac{\gamma-1}{\gamma+1}}{\frac{\gamma-1}{\gamma+1}\frac{p^*}{p_I}+1}\right]\\
\rho_{III} &= \left[\frac{\frac{p^*}{p_{IV}}+\frac{\gamma-1}{\gamma+1}}{\frac{\gamma-1}{\gamma+1}\frac{p^*}{p_{IV}}+1}\right]
\end{aligned}
\tag{A7}
$$

where, $U_K=\{p_K,\rho_K,u_K\}$,

$$
g(p,U_K)=\sqrt{\frac{\frac{2}{(\gamma+1)\rho_K}}{p+\frac{\gamma-1}{\gamma+1}p_K}} \tag{A8}
$$

and $K=\{I,IV\}$.

Appendix B: Ideal gases results

1. Neural network predictions

For ideal EoS, we report the accuracy obtained by three methods: the neural network (Figure 4) predicting one of the middle state values (u^*) and the other values computed with the Euler equations (A2-A3), the neural network predicting all the middle values, and the adaptive Riemann solver. Appendix C reports the accuracy obtained by a physics-informed neural network trained with unsupervised learning for the same EoS.

TABLE VIII. Mean absolute percentage error on u^*, p^*, ρ_{II}, ρ_{III} from different prediction methods including: adaptive Riemann solver, neural network coupled with Euler equations (NN+EE), and standalone neural network (NN) for ideal gases (IG).

	Adaptive RS	NN + EE	NN
		ε_{u^*}	
RCR	0.000%	0.161%	10.190%
RCS	3.365%	0.204%	0.945%
SCR	17.204%	0.862%	3.662%
SCS	1.351%	0.512%	1.306%
Overall	5.480%	0.435%	4.026%
		ε_{p^*}	
RCR	0.000%	0.122%	2.638%
RCS	36.625%	0.141%	1.080%
SCR	38.683%	0.185%	1.297%
SCS	3.601%	0.088%	0.646%
Overall	19.727%	0.134%	1.415%
		$\varepsilon_{\rho_{II}}$	
RCR	0.000%	0.167%	2.835%
RCS	35.570%	0.119%	1.347%
SCR	51.041%	0.314%	2.240%
SCS	17.529%	0.142%	1.090%
Overall	26.035%	0.185%	1.878%
		$\varepsilon_{\rho_{III}}$	
RCR	0.000%	0.151%	2.740%
RCS	47.691%	0.251%	2.255%
SCR	36.570%	0.128%	1.674%
SCS	17.516%	0.140%	1.713%
Overall	25.444%	0.167%	2.096%

Table VIII reports the accuracy obtained by the Riemann solvers and the neural networks for the ideal EoS. The neural network coupled with the Euler equations has the best accuracy for each waves configuration except the two rarefaction waves configuration, in which the adaptive Riemann solver uses the analytical solution. An interesting result is that the neural networks have a different value of accuracy for each configuration. It is reasonable to have the best accuracy for the two rarefaction waves because it is the only configuration with an analytical solution. However, RCS and SCR configurations are specular, and they have very different accuracy. This can be a consequence of the fact that we consider a flow mostly with positive velocity. Another interesting result is that the two shock waves configuration is not the one with the worst accuracy.

Table IX compares the average time to compute each sample. The adaptive Riemann solver is the fastest since it is not iterative, and for RCS and SCR, it has to compute only four values; for this reason, it is faster for these cases. In the case of two rarefaction waves, the exact and adaptive Riemann solvers have similar times, because they are based on the same formula–the analytical solution. Nevertheless, for the two shock waves configuration, the exact Riemann solver

TABLE IX. Average time required to process one sample with exact and adaptive Riemann solvers and the neural networks, for ideal EoS for all possible outcomes.

	Exact RS	Adaptive RS	NN+EE	NN
RCR	0.214 μs	0.260 μs	137 μs	23.7 μs
RCS	0.586 μs	0.068 μs	147 μs	23.7 μs
SCR	0.555 μs	0.068 μs	138 μs	23.7 μs
SCS	0.633 μs	0.113 μs	144 μs	24.0 μs
Overall	0.512 μs	0.149 μs	61.5 μs	14.9 μs

TABLE X. Average time required to process one iteration of the numerical method of each method used to resolve the flux at the interface for an ideal gases (IG).

	Time per iteration
Rusanov flux	0.170 s
HLL flux	0.170 s
HLLC flux	0.155 s
Exact RS	0.160 s
NN+EE	0.220 s
NN	0.200 s

is more than four times slower than the adaptive solver because it required 3 to 5 iterations on average. The neural networks are the slowest, and the standalone one has almost identical time for each configuration. The neural network with the Euler equations is slower due to the execution of the analytical part. When the samples are processed all together instead of in categories, the computation is faster, this result is because the networks process the data all together in an array, and hence the overall time to process the array does not increase linearly with the number of samples. Using the neural network inside a C++ code with Eigen library–to perform the matrix multiplications–42. completes the prediction in 0.55 ms. Therefore, we can conclude that for the ideal EoS, the neural networks do not improve the computational time for the solution of the Riemann problem.

2. Neural network embedded in CFD

Consider that for ideal EoS also other flux solver are used at the interface of the cells, we compare the results also with HLL and HLLC flux (described in appendix D). The initial states of the problem are $P_I = 19$ kPa, $\rho_I = 0.8\frac{\text{kg}}{\text{m}^3}$, $P_{IV} = 3$ kPa, and $\rho_{IV} = 0.1\frac{\text{kg}}{\text{m}^3}$. Figure 9 shows the results of four of the methods used at time $t = 1.22355$ ms, where it is possible to see that the Rusanov flux is the method with the smoothest curve. The other three methods have a similar approximation of the shock wave and better approximation of the rarefaction wave than Rusanov flux. The exact Riemann solver and the neural networks are more accurate and have a similar shape; except at the tail of the rarefaction wave, where there is a small oscillation for the neural network solution. Table X reports the average time per iteration of all methods. As expected, the

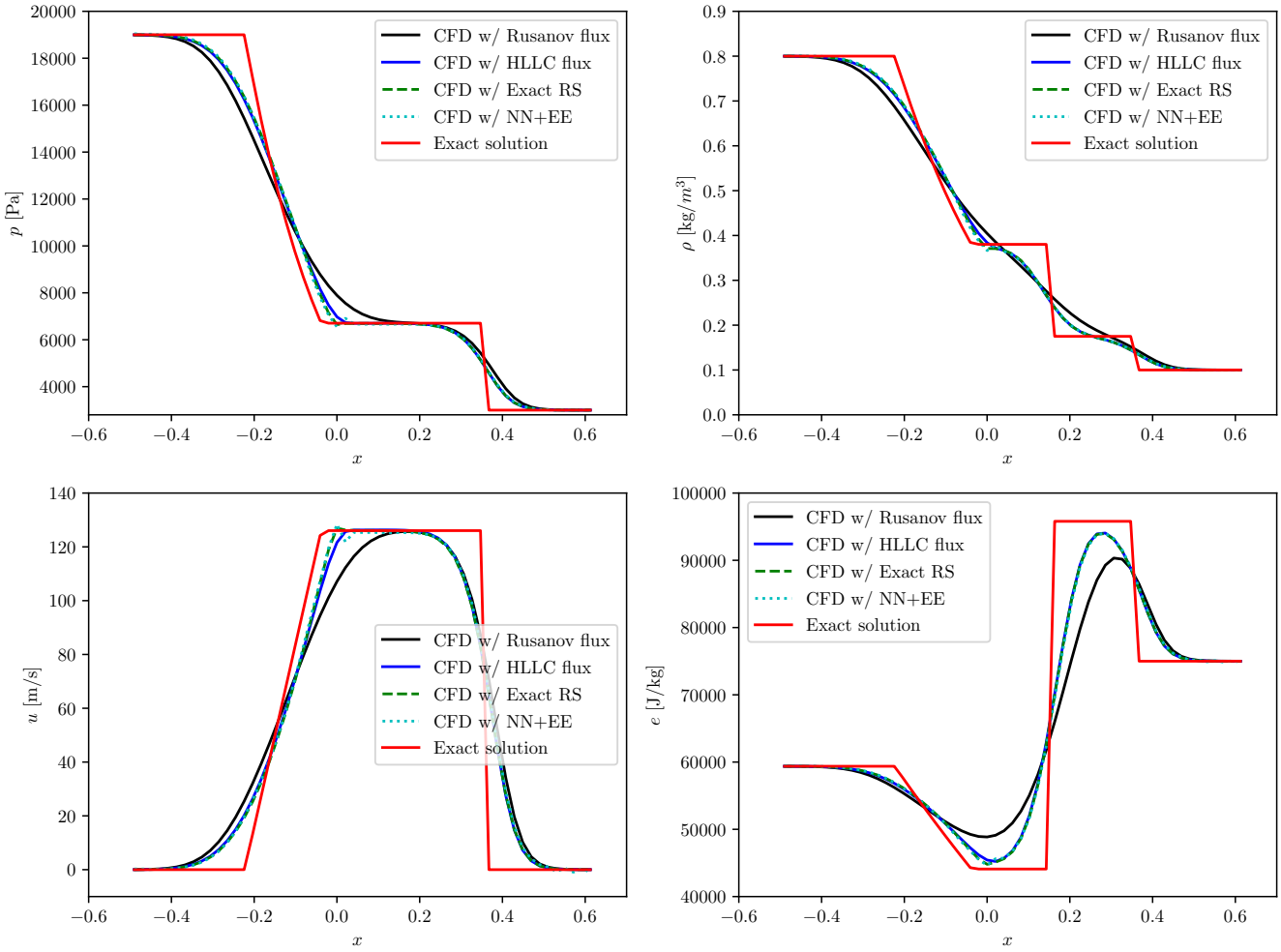


FIG. 9. Numerical simulation of the shock tube problem for ideal gas at time $t = 1.22355$ ms, and initial pressures $P_I = 19$ kPa and $P_{IV} = 3$ kPa.

neural network with the Euler equations is slower than all the other methods. The fact that Rusanov is slower than HLLC even though it has more arithmetical operations is caused by the fact that the Rusanov method as HLL has to compute two absolute values that make them slower than HLLC.

Appendix C: Physics-informed neural network with unsupervised learning

Neural networks can be trained on datasets that already have the solution of the problem, as it can be the values in the middle states of the Riemann problem. Alternatively, they can solve the problem without knowing which one is the correct one. In the latter case, the learning method is called unsupervised. We built a physics-informed neural network with a customized loss function that uses the physical laws of the problem and allows the network to compute one of the middle state values without knowing its correct value.

We chose to investigate this approach only for ideal gases because the computation of the middle states is faster than gases described by real EoS. Therefore, the neural network predicts a value of the velocity (Figure 10) in the middle state, and from this value, it is possible to compute the pressure needed by states I and IV to arrive at this velocity; the difference of these pressures is the loss function value. It is important to consider cases in which it is impossible to compute the pressure because, for example, the rarefaction is too strong. In these cases, the loss function value is set to an arbitrary value

TABLE XI. Ideal gas mean absolute percentage errors for physics-informed neural network trained with unsupervised learning.

	ε_{u^*}	ε_{p^*}	$\varepsilon_{\rho_{II}}$	$\varepsilon_{\rho_{III}}$
RCR	0.511%	0.532%	0.727%	0.744%
RCS	0.300%	0.377%	0.233%	0.608%
SCR	1.425%	0.451%	0.753%	0.273%
SCS	0.444%	0.126%	0.188%	0.205%
Overall	0.670%	0.371%	0.475%	0.458%

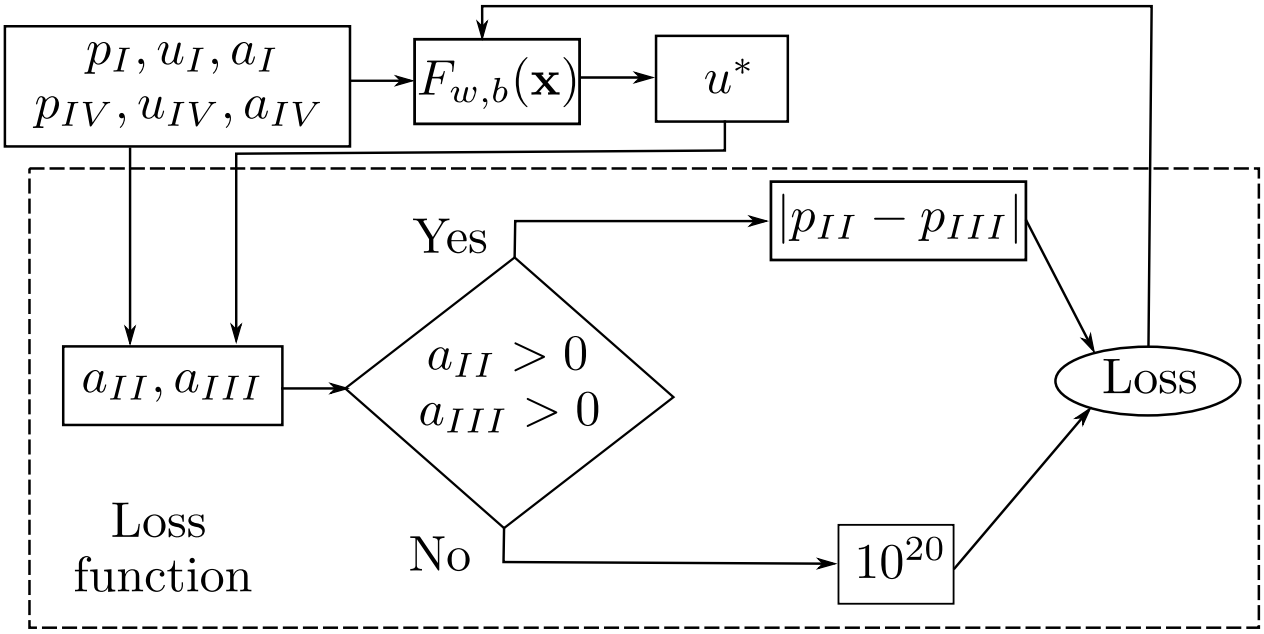


FIG. 10. Algorithm of the loss function for unsupervised learning in case of ideal EoS.

that in our case is 10^{20}. The setup of the arbitrary value can change and lead to a faster training time, but its investigation goes beyond the focus of our study.

Considering that the loss function requires more computation than the standard ones, the training time of this neural network is higher than in the case of supervised learning, but the time to compute the solution remains the same. Table XI reports the accuracy of the physics-informed neural network. It is possible to see that results are better than the standalone neural network for the same problem. Compared to the neural network with the Euler equations the accuracy is worst. However, this result proves that the neural network can learn without knowing the solution with an acceptable accuracy to be used in the CFD, considering that the standalone neural network can work in the CFD.

Appendix D: Numerical approach for CFD

There exist several kinds of CFD codes, and most of them rely on the solution of the Riemann problem. In our analysis, we used the Godunov method that is based on the solution of the Euler equations (1-2). Integrating through the time variable lead to:

$$\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n + \frac{1}{\Delta x}\int_{t^n}^{t^{n+1}} \left(\mathbf{F}_{i+\frac{1}{2}} - \mathbf{F}_{i-\frac{1}{2}}\right) dt \qquad \text{(D1)}$$

where $i+\frac{1}{2}$ and $i-\frac{1}{2}$ are the left and right boundaries of cell i, and $\int_{t^n}^{t^{n+1}} \mathbf{F}_{i+\frac{1}{2}} dt$ is obtained solving the Riemann problem at the boundary between cell i and $i+1$ with initial states the values in these cells. Therefore, the code has to solve the Riemann problem in each boundary between the cells to proceed forward in time, as shown in Figure 11. After the integration, the values in each cell are updated to the next step, and these values are the initial conditions for the following Riemann problems. Even though the Godunov method required only the values at the interface between the cells, and not the entire solution of the Riemann problem, it is impossible to obtain a single state without the entire solution of the problem because to know which one is the state at the interface the method needs to know where is each wave. In some particular cases, one wave is enough to change one of the states to the same pressure and velocity of the other, and hence the state at the interface is equal to the previous or the later cell. In most cases, the algorithms used to solve the Riemann problem are based on an approximate Riemann solvers because it is too computationally expensive with an exact one. Even though the approximate solver can have an error of several percentages, on the CFD code, this error is dumped because it works with the flux of the initial states instead of the states themselves. For our investigation, we used the exact Riemann solver, the neural network, and a combination of neural network and Euler equations to solve the Riemann problem at the interface, and three approximate solvers–that do not depend on the EoS: Rusanov flux[48], HLL (Harten-Lax-van Leer)[49] flux, and HLLC (Harten-Lax-van Leer Contact) flux[50].

1. HLL (Harten-Lax-van Leer) flux

HLL flux solver considered only the fastest wave in each direction, and it does not consider any contact discontinuity in the solution of the Riemann problem. As the fastest wave is

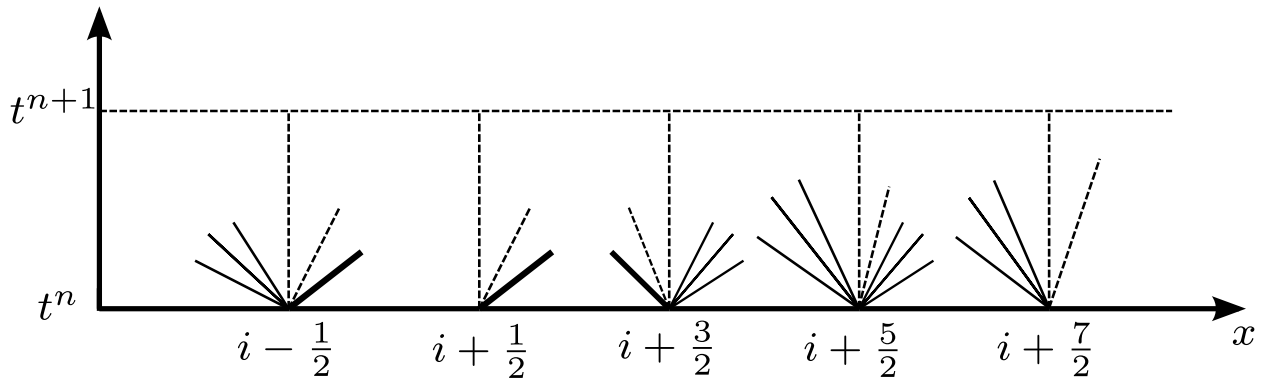


FIG. 11. The plot represents six adjacent cells and shows the generation of two or three waves between them. Because each cell can have different values of pressure, density, or velocity, there is always a Riemann problem at their interface. After the solution of the Riemann problem, from the values at the interface, it is possible to reconstruct the values at t+1.

taken the fastest wave in case of rarefaction that can be computed as $S_I = |u_I| + a_I$ and $S_{IV} = |u_{IV}| + a_{IV}$, because to compute the speed of the shock wave it is required the state after the wave, and hence more difficult to compute. After some algebraic manipulation and approximations of the Euler equations (1):

$$\int_{t^n}^{t^{n+1}} \mathbf{F}_{i+\frac{1}{2}} dt = \begin{cases} \mathbf{F}_I & \text{if} \quad 0 \le S_I \\ \frac{S_{IV}\mathbf{F}_I - S_I\mathbf{F}_{IV} + S_{IV}S_I(\mathbf{Q}_{IV} - \mathbf{Q}_I)}{S_{IV} - S_I} & \text{if} \quad S_I < 0 < S_{IV} \\ \mathbf{F}_{IV} & \text{if} \quad 0 \ge S_{IV} \end{cases} \tag{D2}$$

where the index I and IV are referred to the values in cell i and $i+1$. HLL flux does not solve the Riemann problem but computes the flux values directly from the Euler equations. It allows only the left and right state and an approximation of the middle state as a possible solution.

2. Rusanov flux

Rusanov flux solver is a simplification of the HLL flux solver. This solver considers only one wave speed that is the fastest between the left and right wave $S = \max(S_I, S_{IV})$. Then, it uses S inside the equation for the intermediate states of HLL flux and obtains:

$$\int_{t^n}^{t^{n+1}} \mathbf{F}_{i+\frac{1}{2}} dt = \frac{1}{2}(\mathbf{F}_I + \mathbf{F}_{IV}) - \frac{S}{2}(\mathbf{Q}_{IV} - \mathbf{Q}_I) \tag{D3}$$

where the index I and IV are referred to the values in cells i and $i+1$. For this solver, there is no distinction among left, right, or middle states; the solution is always computed as an approximation.

3. HLLC (Harten-Lax-van Leer Contact) flux

HLLC flux solver is based on the same assumption of HLL, but it also considers the contact discontinuity. The first step of the solver is to compute an approximation of the middle state using the equations (A5) to compute p^*, u^*, $\bar{\rho}$, and $\bar{a}$. From these values:

$$q_K = \begin{cases} 1 & \text{if} \quad p^* \le p_K \\ \sqrt{1 + \frac{\gamma+1}{2\gamma}\left(\frac{p^*}{p_K} - 1\right)} & \text{if} \quad p^* > p_K \end{cases} \tag{D4}$$

where $K = \{I, IV\}$. The speed of the waves are computed as $S_I = u_I - a_I q_I$, $S^* = u^*$, and $S_{IV} = u_{IV} - a_{IV} q_{IV}$. In this case, the algorithm keeps into consideration also the shock waves, as can be seen from q_K, and the speed of the contact discontinuity S^* is the exact speed, except for the non-accuracy in u^*. In this way, it is possible to distinguish among all four states:

$$\int_{t^n}^{t^{n+1}} \mathbf{F}_{i+\frac{1}{2}} dt = \begin{cases} \mathbf{F}_I & \text{if} \quad 0 \le S_I \\ \mathbf{F}_{II} & \text{if} \quad S_I < 0 \le S^* \\ \mathbf{F}_{III} & \text{if} \quad S^* < 0 < S_{IV} \\ \mathbf{F}_{IV} & \text{if} \quad 0 \ge S_{IV} \end{cases} \tag{D5}$$

where $\mathbf{F}_{II} = \mathbf{F}_I + S_I(\mathbf{Q}_{II} - \mathbf{Q}_I)$ and $\mathbf{F}_{III} = \mathbf{F}_{IV} + S_{IV}(\mathbf{Q}_{III} - \mathbf{Q}_{IV})$. The vector of the middle state is computed as:

$$\mathbf{Q}_N = \rho_K \left(\frac{S_K - u_K}{S_K - S^*}\right) \begin{bmatrix} 1 \\ S^* \\ \frac{E_K}{\rho_K} + (S^* - u_K)\left[S^* + \frac{p_K}{\rho_K(S_K - u_K)}\right] \end{bmatrix} \tag{D6}$$

where $N = \{II, III\}$ and $K = \{I, IV\}$.

[1] E. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics, A Practical Introduction* (Springer-Verlag, Berlin Heidelberg, 2009).

[2]V. Gyrya, M. Shashkov, A. Skurikhin, and S. Tokareva, "Machine learning approaches for the solution of the riemann problem in fluid dynamics: a case study," ArXiv (2019).

[3]J. Magiera, D. Ray, J. Hesthaven, and C. Rohde, "Constraint-aware neural networks for riemann problems," Journal of Computational Physics **409** (2020), https://doi.org/10.1016/j.jcp.2020.109345.

[4]J. C. H. Wang, *Riemann solvers with non-ideal thermodynamics: exact, approximate, and machine learning solutions*, Ph.D. thesis (2022).

[5]S. Xiong, X. He, Y. Tong, R. Liu, and B. Zhu, "Roenets: Predicting discontinuity of hyperbolic systems from continuous data," ArXiv (2020), arXiv:2006.04180.

[6]R. Patel, I. Manickam, N. Trask, M. Wood, M. Lee, I. Tomas, and E. Cyr, "Thermodynamically consistent physics-informed neural networks for hyperbolic systems," Journal of Computational Physics **449** (2022), 10.1016/j.jcp.2021.110754.

[7]H. Huang, Y. Liu, and V. Yang, "Neural networks with inputs based on domain of dependence and a converging sequence for solving conservation laws, part i: 1d riemann problems," ArXiv (2021), arXiv:2109.09316.

[8]B. Argrow, "Computational analysis of dense gas shock tube flow," Shock Waves **6** (1996), https://doi.org/10.1007/ BF02511381.

[9]S. Schlamp and T. Rösgen, "Flow in near-critical fluids induced by shock and expansion waves," Shock Waves **14**, 93–101 (2005).

[10]B. Mortimer, B. Skews, and L. Felthun, "The use of a slow sound speed fluorocarbon liquid for shock wave research," Shock Waves **8**, 63–69 (1998).

[11]H. Kim, Y. Choe, H. Kim, D. Min, and C. Kim, "Methods for compressible multiphase flows and their applications," Shock Waves **29**, 235–261 (2019).

[12]B. Re and A. Guardone, "An adaptive ale scheme for non-ideal compressible fluid dynamics over dynamic unstructured meshes," Shock Waves **29**, 73–99 (2019).

[13]A. Martinez, M. Migliorino, C. Scalo, and S. Heister, "Experimental and numerical investigation of standing-wave thermoacoustic instability under transcritical temperature conditions," J Acoust Soc Am. **159** (2021), doi: 10.1121/10.0006659.

[14]S. Hunt, M. Migliorino, C. Scalo, and S. Heister, "Onset criteria for bulk-mode thermoacoustic instabilities in supercritical hydrocarbon fuels," Journal of Fluids Engineering **143** (2021), https://doi.org/10.1115/1.4049401.

[15]M. Migliorino and C. Scalo, "Real-fluid effects on standing-wave thermoacoustic instability," Journal of Fluid Mechanics **883** (2020), doi:10.1017/jfm.2019.856.

[16]M. Yao, Z. Sun, C. Scalo, and J. Hickey, "Vortical and thermal interfacial layers in wall-bounded turbulent flows under transcritical conditions," Phys. Rev. Fluids **4** (2019), https://link.aps.org/doi/10.1103/PhysRevFluids.4.084604.

[17]J. C. H. Wang and J.-P. Hickey, "Analytical solutions to shock and expansion waves for non-ideal equations of state," Physics of Fluids **32**, 086–105 (2020).

[18]J. C. Wang and J.-P. Hickey, "A class of structurally complete approximate riemann solvers for trans- and supercritical flows with large gradients," Journal of Computational Physics **468**, 111521 (2022).

[19]F. Scarselli, M. Gori, A. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," IEEE Transactions on Neural Networks **20**, 61–80 (2009).

[20]R. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," ArXiv (2018), arXiv:1806.07366.

[21]A. Kratsios, "The universal approximation property," Ann Math Artif Intell **89**, 435–469 (2021).

[22]M. Raissi, P. Perdikaris, and G. Karniadakis, "Machine learning of linear differential equations using gaussian processes," Journal of Computational Physics **348**, 683–693 (2017).

[23]M. Raissi and G. Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," Journal of Computational Physics **357**, 125–141 (2018).

[24]S. Udrescu and M. Tegmark, "Symbolic pregression: Discovering physical laws from distorted video," Phys. Rev. E **103** (2021), https://doi.org/10.1103/PhysRevE.103.043307.

[25]M. Raissi, P. Perdikaris, and G. Karniadakis, "Inferring solutions of differential equations using noisy multi-fidelity data," Journal of Computational Physics **335**, 736–746 (335).

[26]M. Raissi, P. Perdikaris, and G. Karniadakis, "Numerical gaussian processes for time-dependent and nonlinear partial differential equations," SIAM Journal on Scientific Computing **40**, A172–A198 (2018).

[27]M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," Journal of Computational Physics **378**, 686–707 (2019).

[28]P. Colella, "Efficient solution algorithms for the riemann problem for real gases," Journal of computational physics **59**, 264–289 (1985).

[29]J. Kamm, "An exact, compressible one-dimensional riemann solver for general, convex equations of state," ArXiv (2015), https://doi.org/10.2172/1172220.

[30]A. Mohan and D. Gaitonde, "A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks," ArXiv (2018).

[31]D. Xiao, C. Heaney, L. Mottet, F. Fang, W. Lin, I. Navon, Y. Guo, O. Matar, A. Robins, and C. Pain, "A reduced order model for turbulent flows in the urban environment using machine learning," Building and Environment **148** (2019), https://doi.org/10.1016/j.buildenv.2018.10.035.

[32]B. Hanna, N. Dinh, R. Youngblood, and I. Bolotnov, "Machine-learning based error prediction approach for coarse-grid computational fluid dynamics (cg-cfd)," Progress in Nuclear Energy **118** (2020), https://doi.org/10.1016/j.pnucene.2019.103140.

[33]E. Parish and K. Duraisamy, "A paradigm for data-driven predictive modeling using field inversion and machine learning," Journal of Computational Physics **305**, 758–774 (2016).

[34]Y. Zhao, H. Akolekar, J. Weatheritt, V. Michelassi, and R. Sandberg, "Rans turbulence model development using cfd-driven machine learning," Journal of Computational Physics **411** (2020), https://doi.org/10.1016/j.jcp.2020.109413.

[35]M. Bode, M. Gauding, Z. Lian, D. Denker, M. Davidovic, K. Kleinheinz, J. Jitsev, and H. Pitsch, "Using physics-informed enhanced super-resolution generative adversarial networks for subfilter modeling in turbulent reactive flows," Proceedings of the Combustion Institute **38** (2021), https://doi.org/10.1016/j.proci.2020.06.022.

[36]H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa, "Physics-informed neural networks for solving reynolds-averaged navier-stokes equations," Physics of Fluids **34** (2021), https://doi.org/10.1063/5.0095270.

[37]Accessed 8 November 2021.

[38]K. Kim, J. Hickey, and C. Scalo, "Pseudophase change effects in turbulent channel flow under transcritical temperature conditions," Journal of Fluid Mechanics **871**, 52–91 (2019).

[39]B. Wescott, D. Stewart, and W. Davis, "Equation of state and reaction rate for condensed-phase explosives," Journal of Applied Physics **98** (2005), https://doi.org/10.1063/1.2035310.

[40]M. Migliorino and C. Scalo, "Heat-induced planar shock waves in supercritical fluids," Shock Waves **30** (2020), https://doi.org/10.1007/s00193-019-00934-y.

[41]M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (2016) pp. 265–283.

[42]D. Kingma and J. Ba, "Adam: A method for stochastic optimization," ArXiv (2014), arXiv:1412.6980.

[43]T. Haga and S. Kawai, "On a robust and accurate localized artificial diffusivity scheme for the high-order flux-reconstruction method," Journal of Computational Physics **376**, 534–563 (2019).

[44]S. Godunov, "A finite difference method for the computation of discontinuous solutions of the equations of fluid dynamics," Mat. Sb. **47**, 357–393 (1959).

[45]S. Godunov, "Numerical solution of multi–dimensional problems in gas dynamics," Nauka Press (1976).

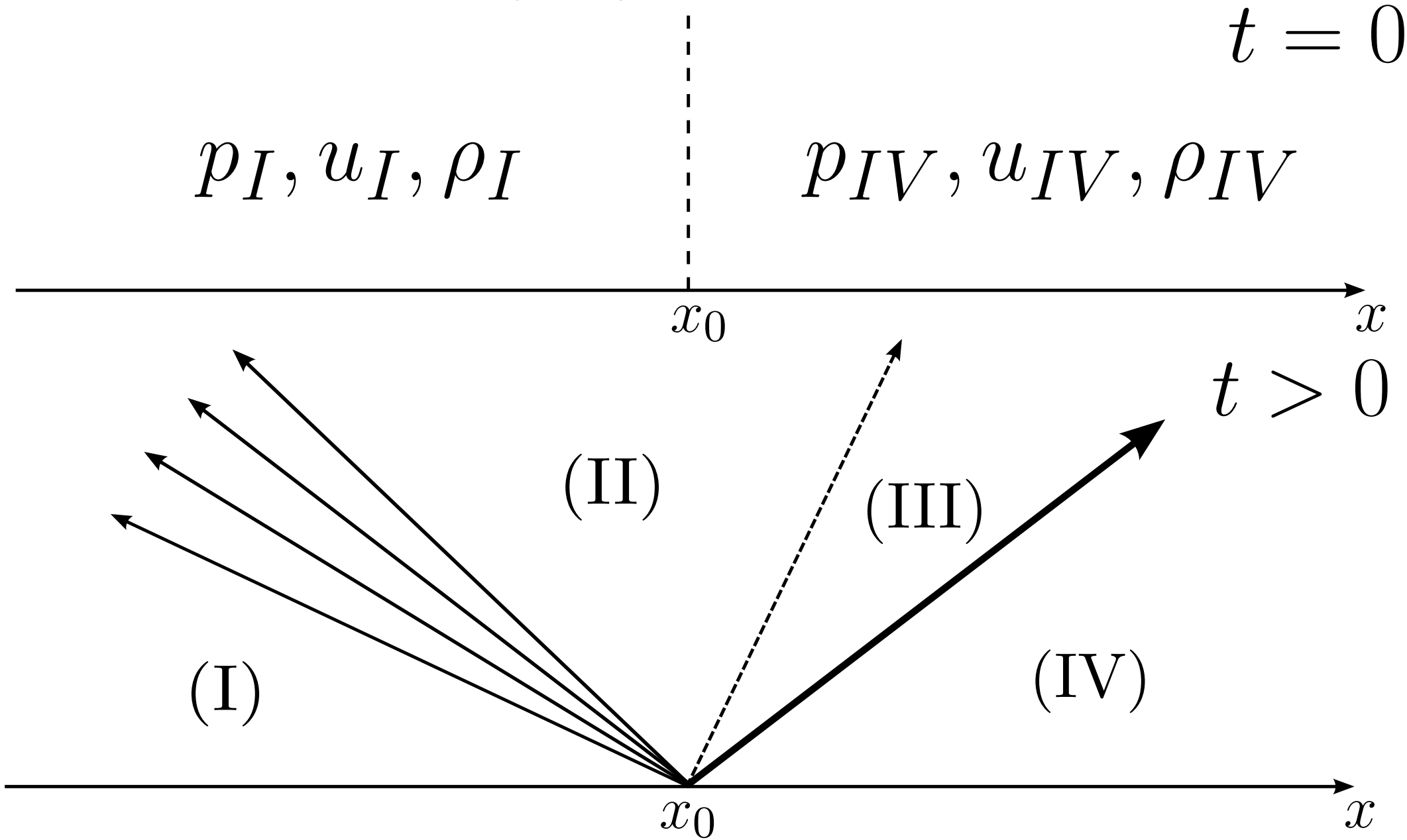
[46]J. Pike, "Riemann solvers for perfect and near-perfect gases," AIAA Journal **31**, 1801–1808 (1993).

[47]J. Gottlieb, "Assessment of riemann solvers for unsteady one-dimensional inviscid flows of perfect gases," Journal of computational physics **78**, 437–458 (1988).

[48]V. Rusanov, "Calculation of interaction of non–steady shock waves with obstacles," USSR Computational Mathematics and Mathematical Physics **1**, 267–279 (1961).

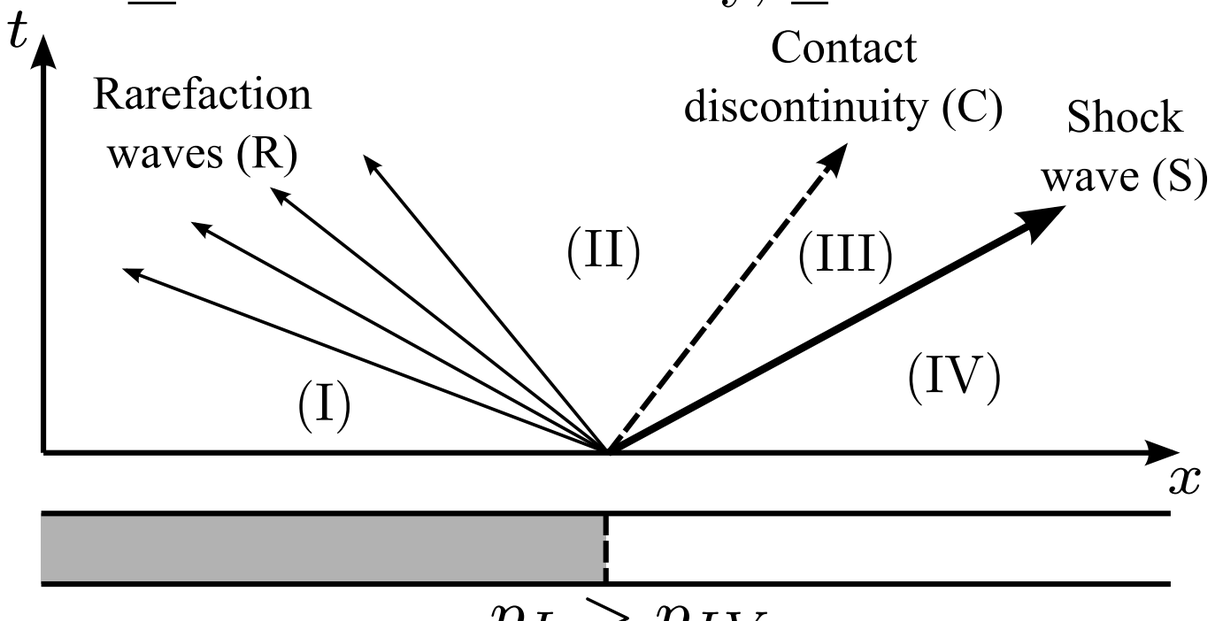
[49]A. Harten, P. Lax, and B. V. Leer, "On upstream differencing and godunov–type schemes for hyperbolic conservation laws," SIAM Review **25**,

35–61 (1983).
[50]E. Toro, M. Spruce, and W. Speares, "Restoration of the contact surface in the hll–riemann solver," Shock Waves **4**, 25–34 (1994).

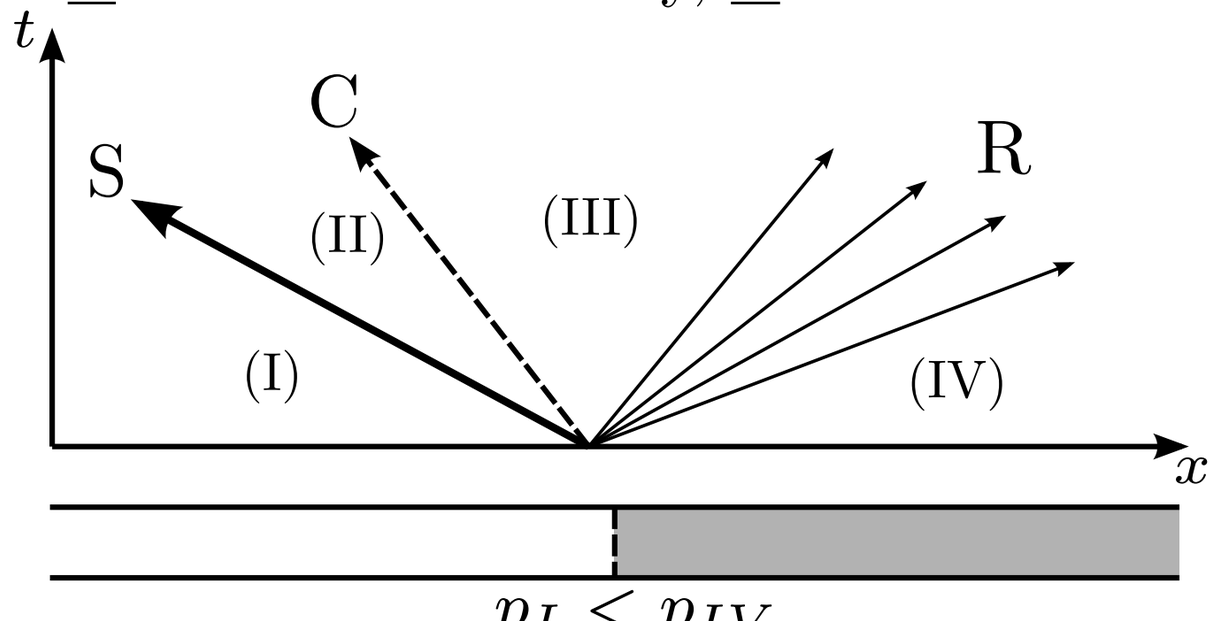


Accepted to Phys. Fluids 10.1063/5.0123466

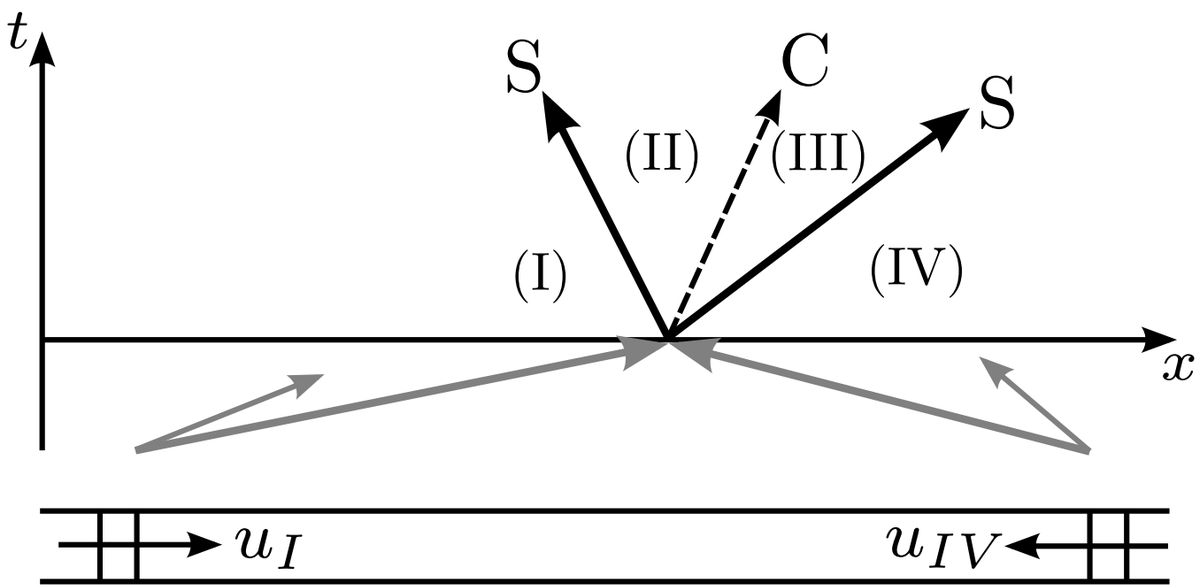
RCS: Rarefaction wave,
Contact discontinuity, Shock wave



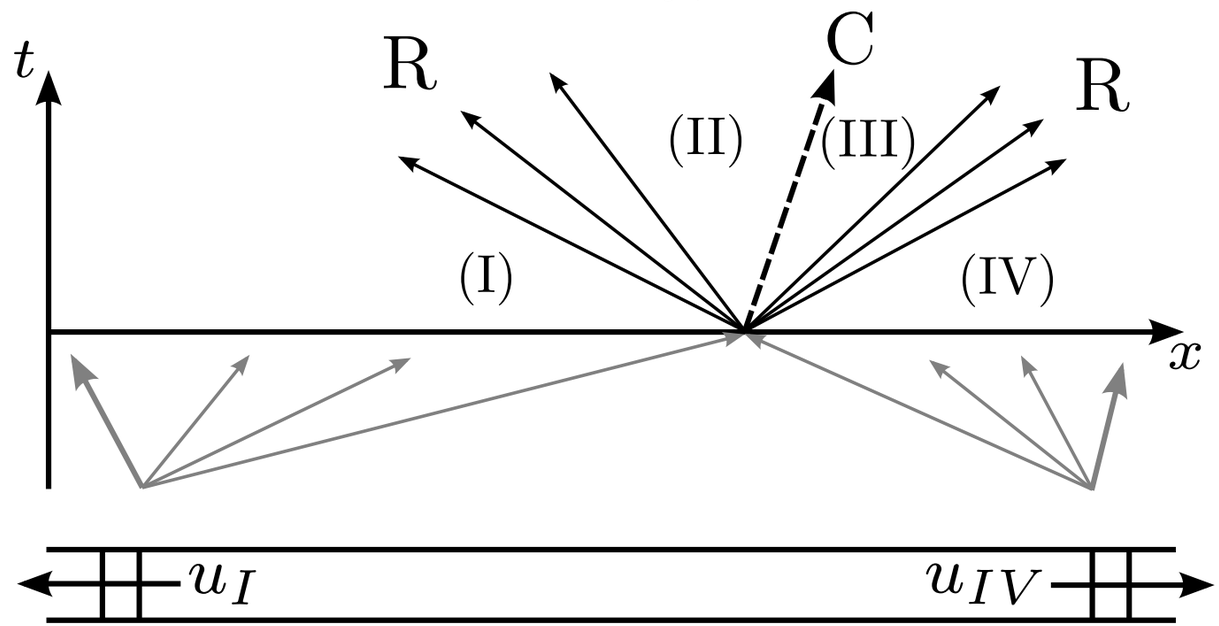
SCR: Shock wave,
Contact discontinuity, Rarefaction wave

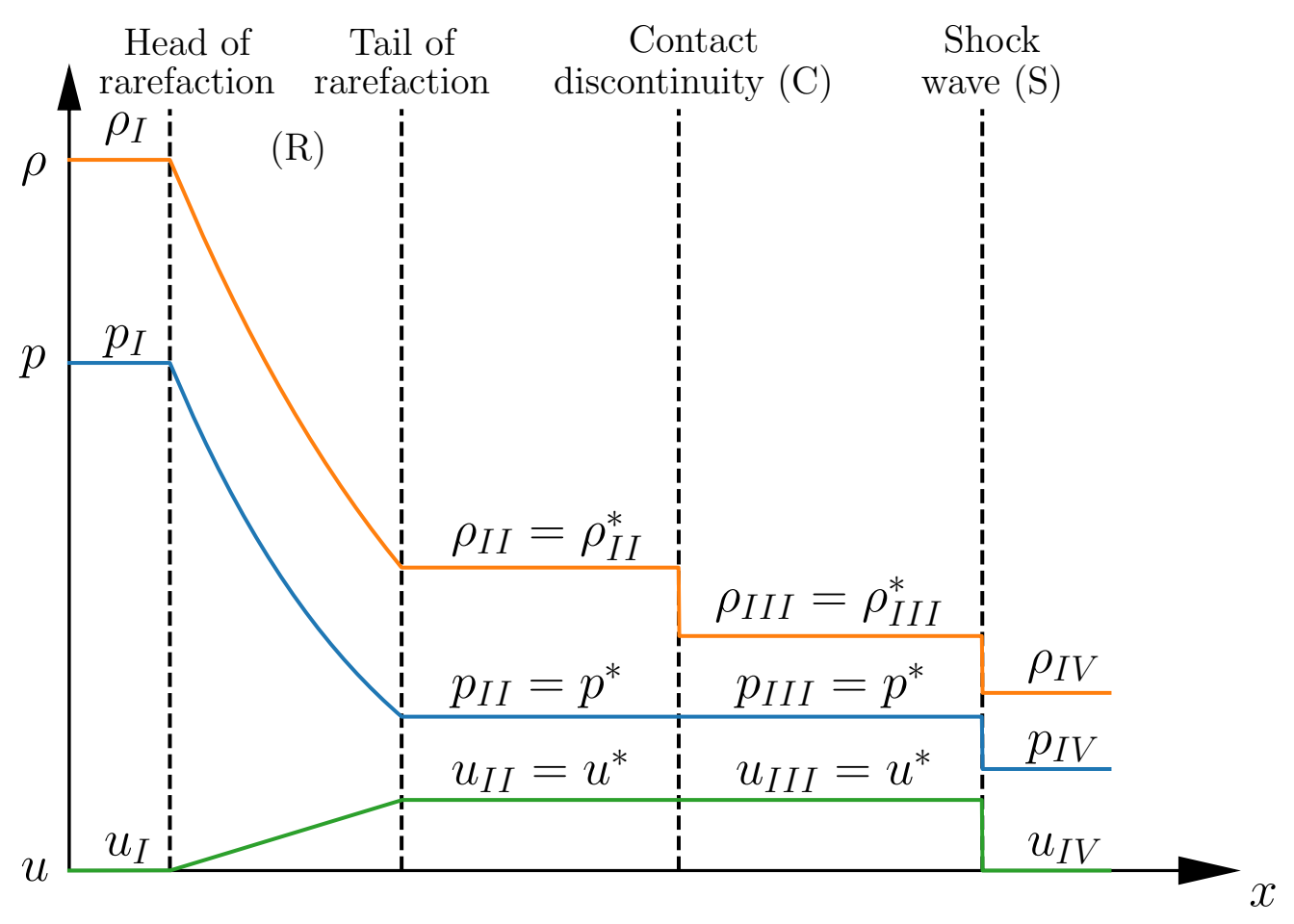


SCS: Shock wave,
Contact discontinuity, Shock wave



RCR: Rarefaction wave,
Contact discontinuity, Rarefaction wave

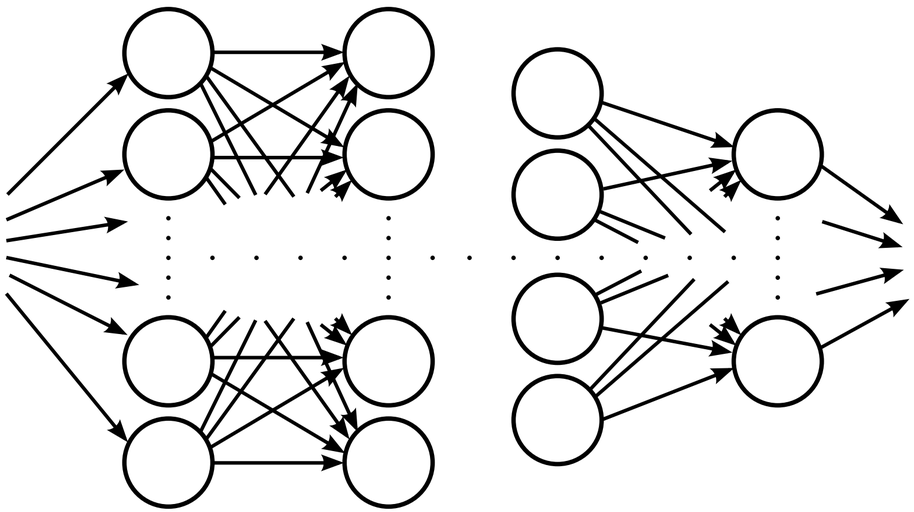



Head of rarefaction
Tail of rarefaction
Contact discontinuity (C)
Shock wave (S)
(R)
ρ
p
u
x
ρ_I
p_I
u_I
$\rho_{II} = \rho^*_{II}$
$\rho_{III} = \rho^*_{III}$
ρ_{IV}
$p_{II} = p^*$
$p_{III} = p^*$
p_{IV}
$u_{II} = u^*$
$u_{III} = u^*$
u_{IV}

Accepted to Phys. Fluids 10.1063/5.0123466

Neural Network coupled with Euler Equation (NN+EE)

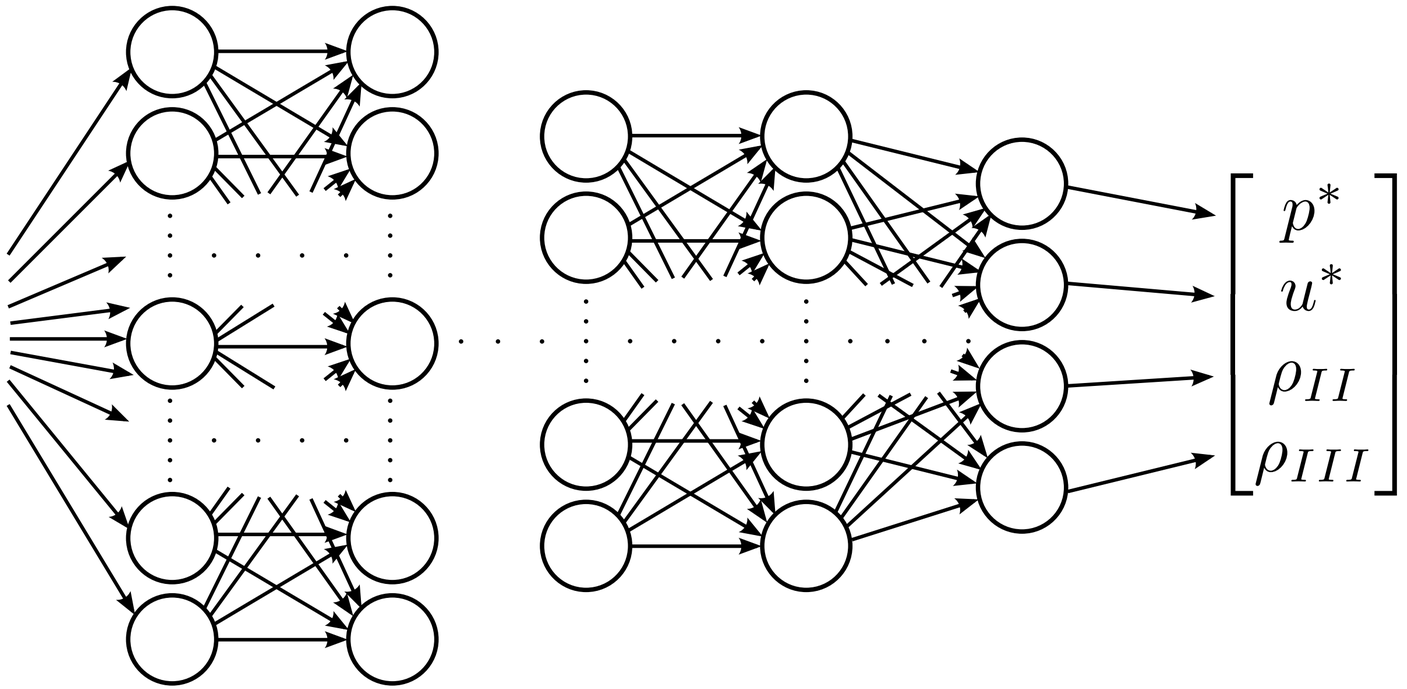
$$\begin{bmatrix} p_I \\ u_I \\ \rho_I (\text{or } a_I) \\ p_{IV} \\ u_{IV} \\ \rho_{IV} (\text{or } a_{IV}) \end{bmatrix}$$



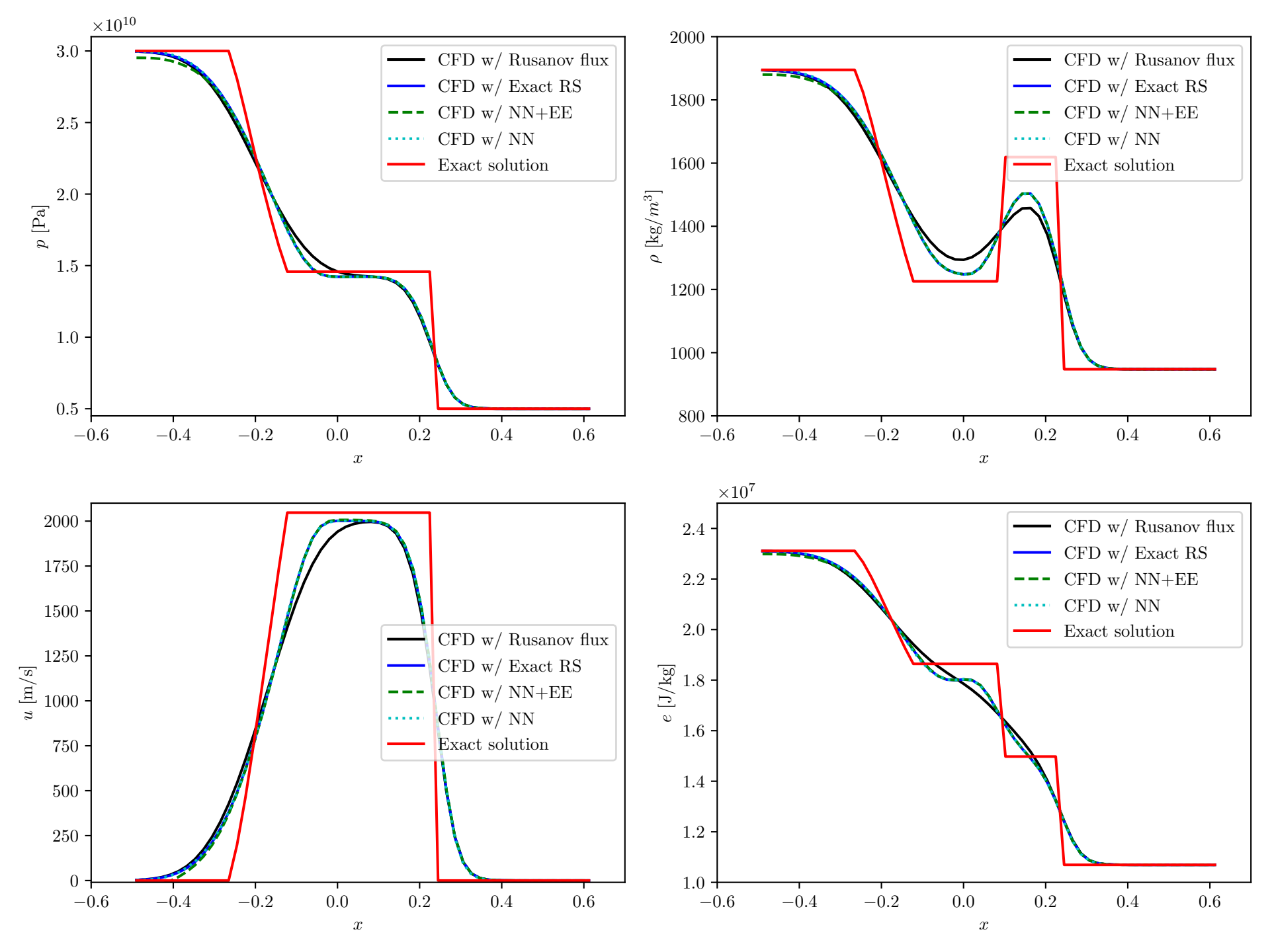
$$\left[p^* (\text{or } u^*)\right] \longrightarrow \begin{array}{c} \rho_t + (\rho u)_x = 0 \\ (\rho u)_t + (\rho u^2 + p)_x = 0 \\ E_t + (u(E+p))_x = 0 \\ [\rho(u - s_w)] = 0 \\ [\rho(u - s_w)^2 + p] = 0 \\ \left[e + \frac{p}{\rho} + \frac{(u-s_w)^2}{2}\right] = 0 \\ f(p, \rho, e) = 0 \end{array} \longrightarrow \begin{bmatrix} p^* \\ u^* \\ \rho_{II} \\ \rho_{III} \end{bmatrix}$$

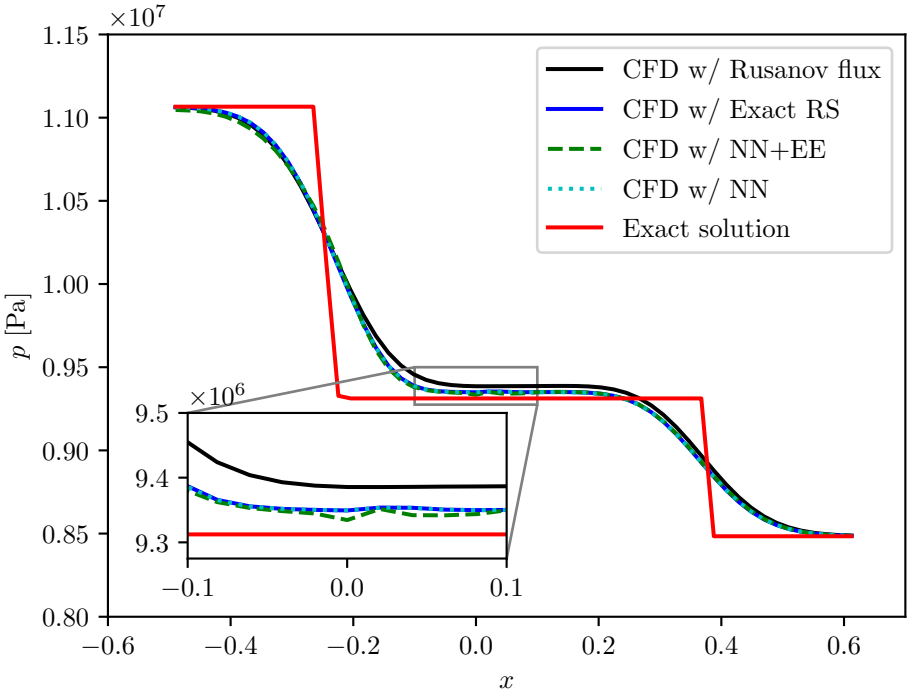
Neural Network standalone (NN)

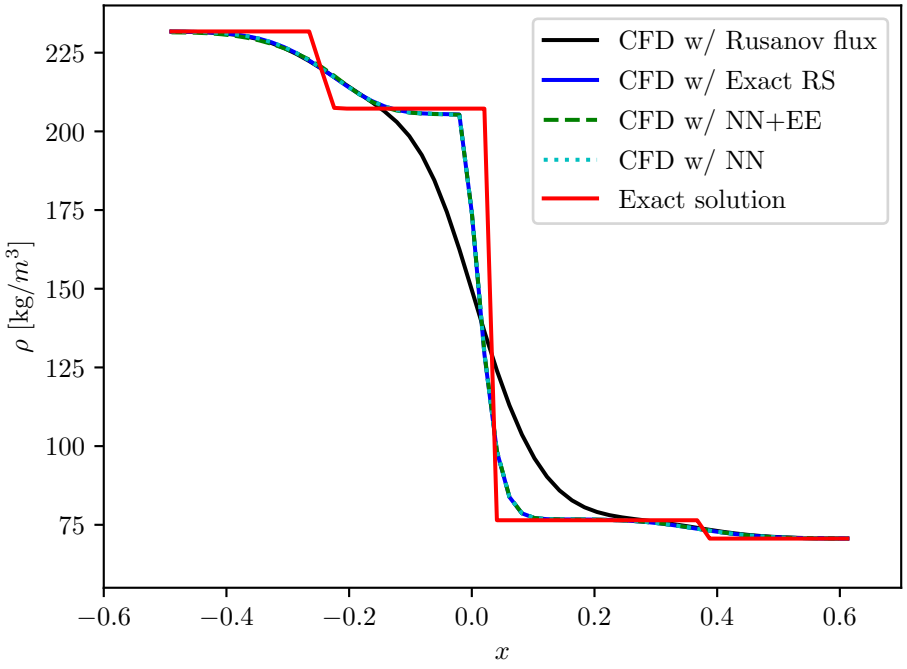
$$\begin{bmatrix} p_I \\ u_I \\ \rho_I (\text{or } a_I) \\ p_{IV} \\ u_{IV} \\ \rho_{IV} (\text{or } a_{IV}) \end{bmatrix}$$

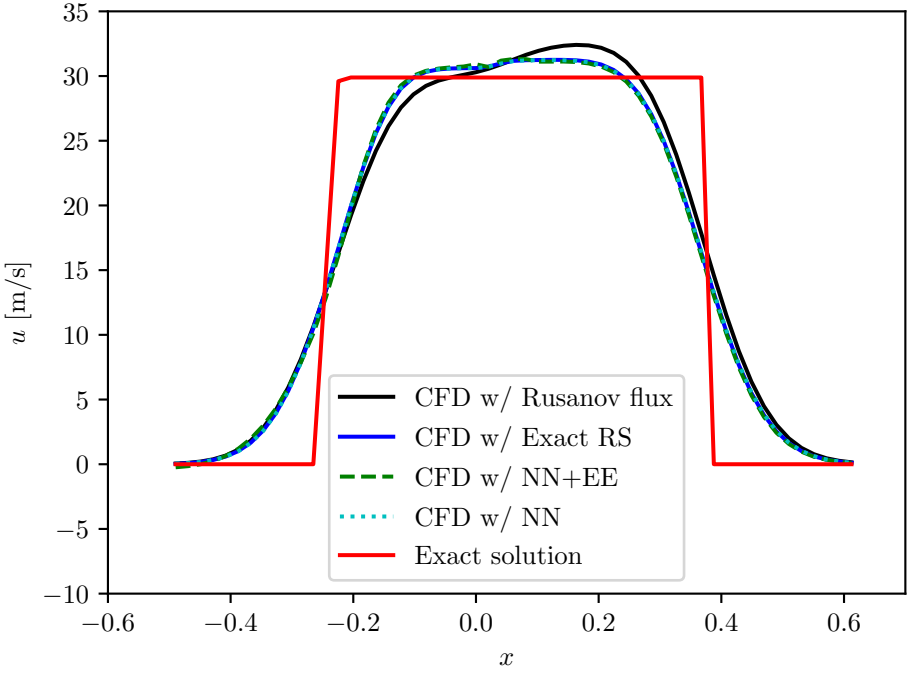


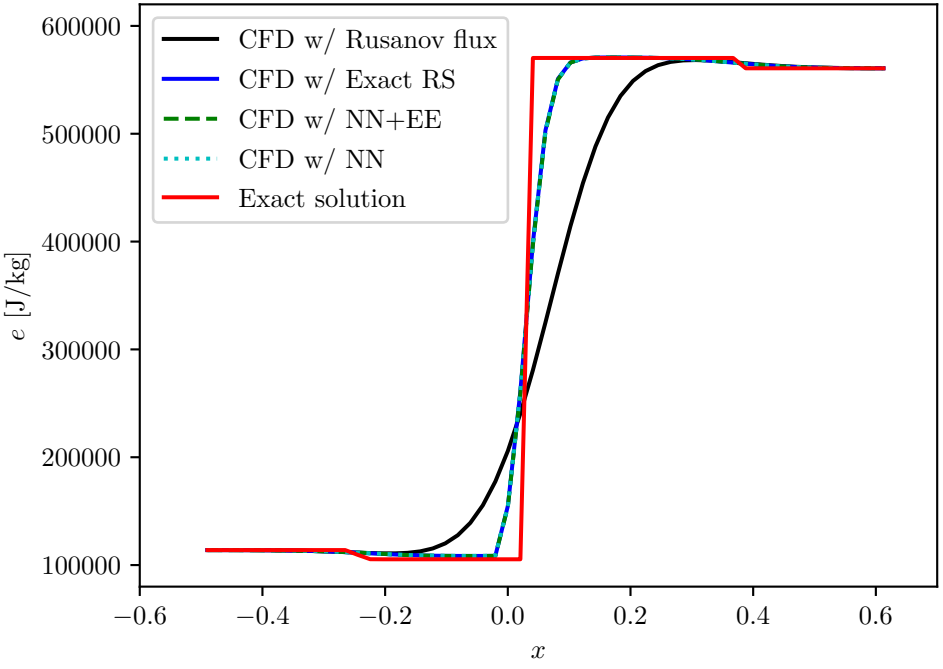
$$\begin{bmatrix} p^* \\ u^* \\ \rho_{II} \\ \rho_{III} \end{bmatrix}$$

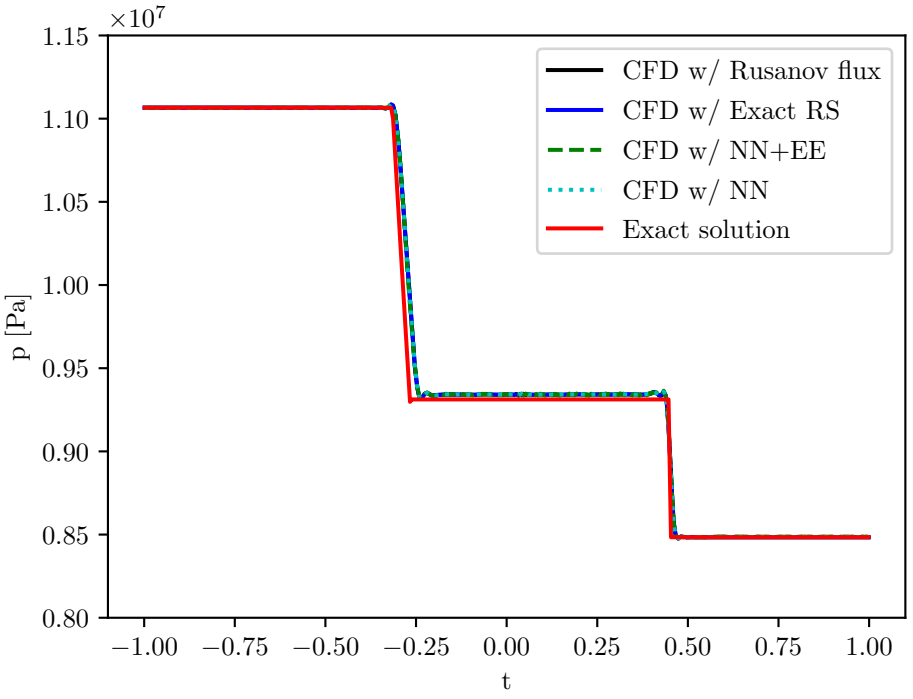

×10^10
p [Pa]
x
CFD w/ Rusanov flux
CFD w/ Exact RS
CFD w/ NN+EE
CFD w/ NN
Exact solution
ρ [kg/m^3]
u [m/s]
×10^7
e [J/kg]

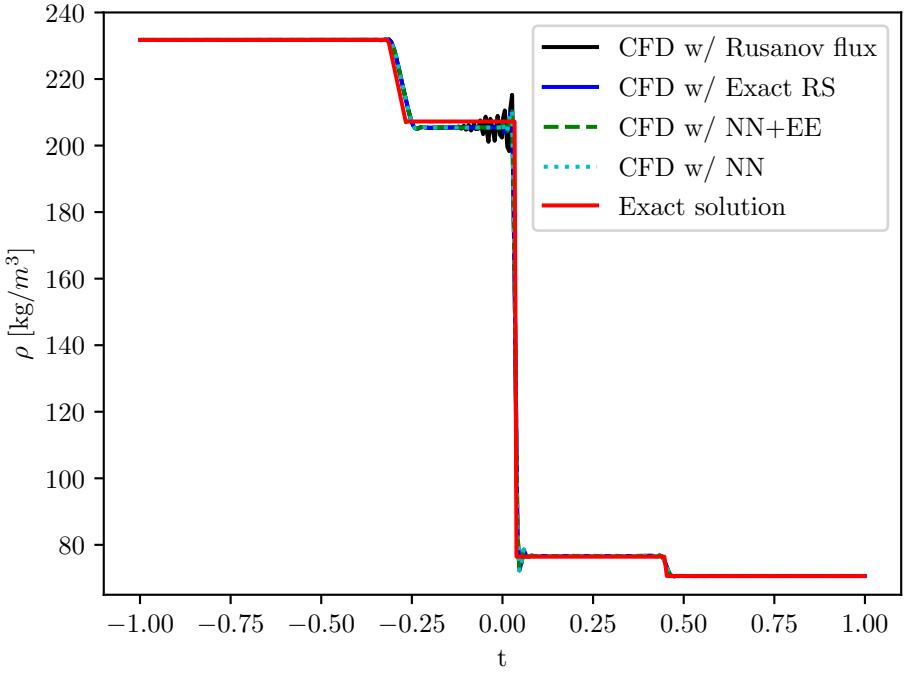

×10^7
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
p [Pa]
×10^6
9.5
9.4
9.3
−0.1
0.0
0.1
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
x
CFD w/ Rusanov flux
CFD w/ Exact RS
CFD w/ NN+EE
CFD w/ NN
Exact solution

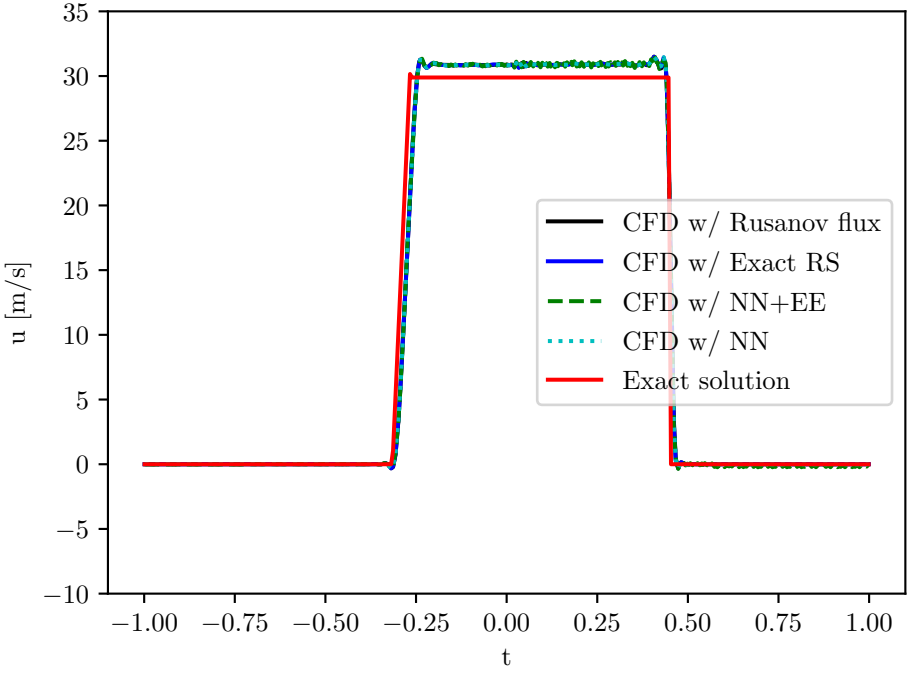

225
200
175
150
125
100
75
ρ [kg/m^3]
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
x
CFD w/ Rusanov flux
CFD w/ Exact RS
CFD w/ NN+EE
CFD w/ NN
Exact solution

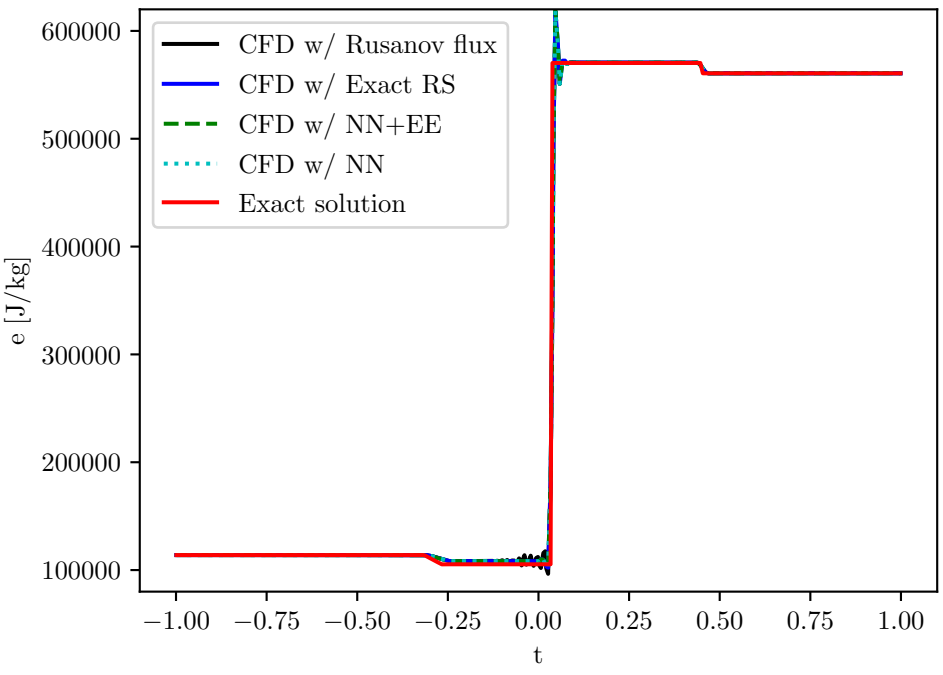

35
30
25
20
15
10
5
0
−5
−10
u [m/s]
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
x
CFD w/ Rusanov flux
CFD w/ Exact RS
CFD w/ NN+EE
CFD w/ NN
Exact solution

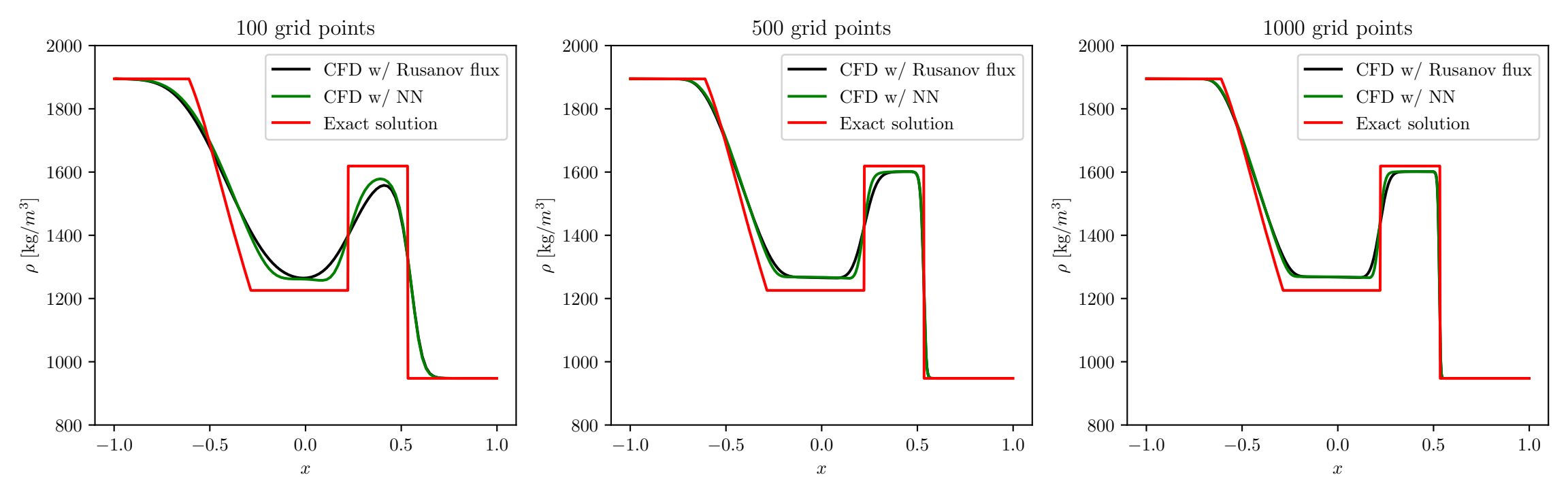

600000
500000
400000
300000
200000
100000
e [J/kg]
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
x
CFD w/ Rusanov flux
CFD w/ Exact RS
CFD w/ NN+EE
CFD w/ NN
Exact solution

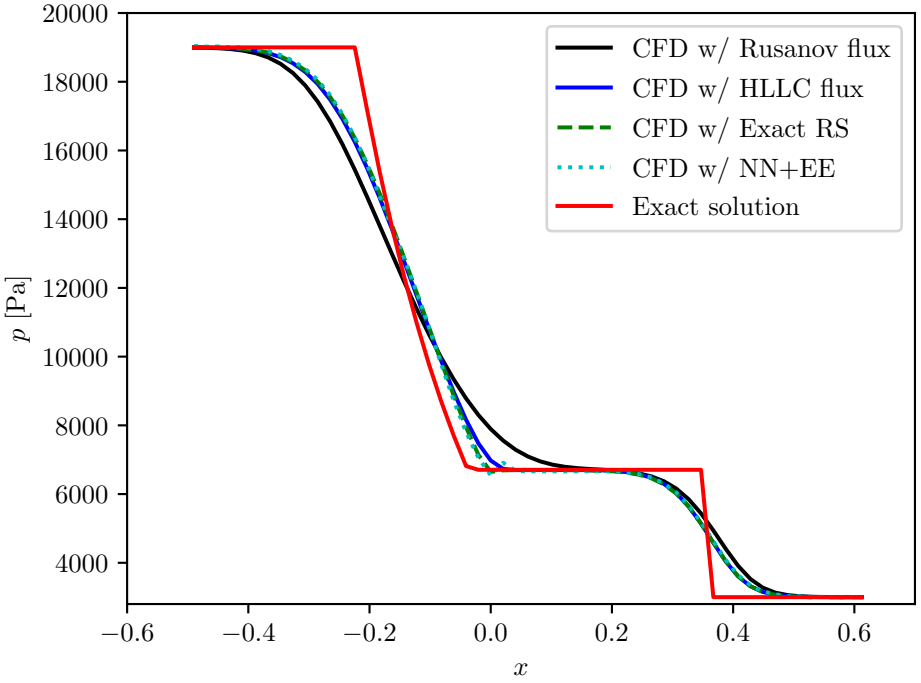

×10^7
1.15
1.10
1.05
1.00
0.95
0.90
0.85
0.80
p [Pa]
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
t
CFD w/ Rusanov flux
CFD w/ Exact RS
CFD w/ NN+EE
CFD w/ NN
Exact solution

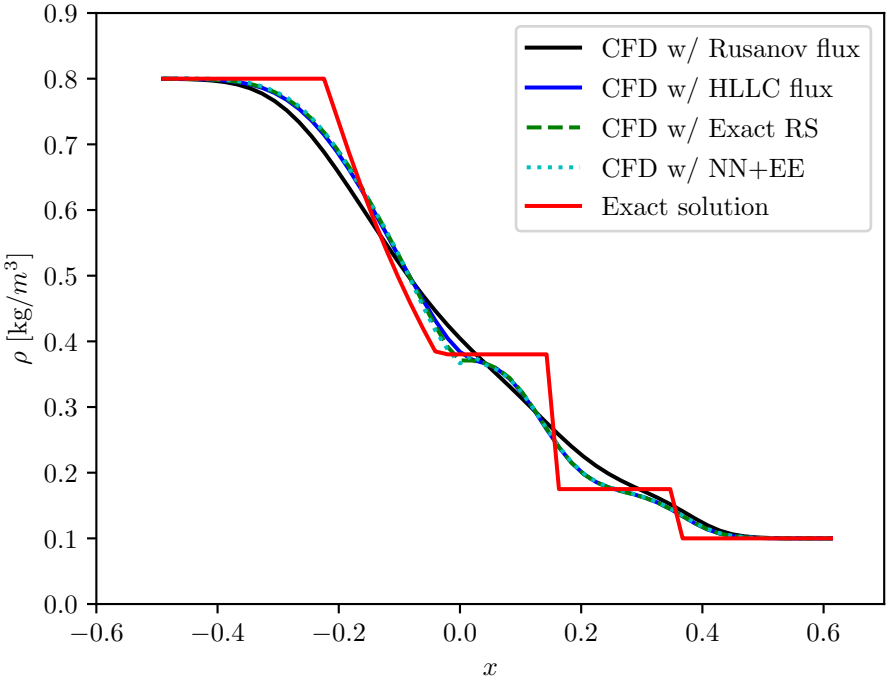

240
220
200
180
160
140
120
100
80
ρ [kg/m^3]
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
t
CFD w/ Rusanov flux
CFD w/ Exact RS
CFD w/ NN+EE
CFD w/ NN
Exact solution

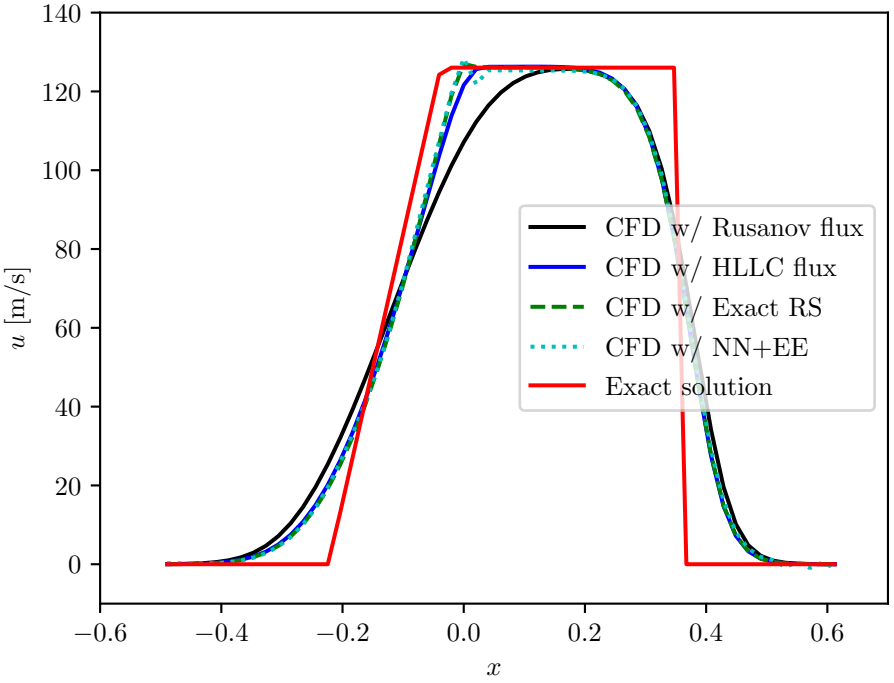

35
30
25
20
15
10
5
0
−5
−10
u [m/s]
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
t
CFD w/ Rusanov flux
CFD w/ Exact RS
CFD w/ NN+EE
CFD w/ NN
Exact solution

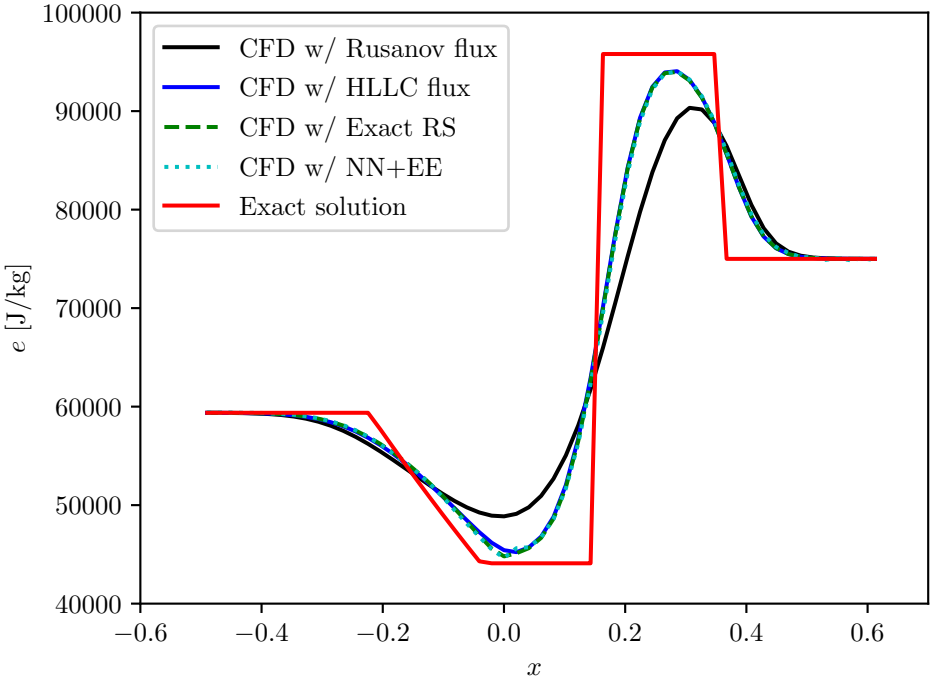

600000
500000
400000
300000
200000
100000
e [J/kg]
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00
t
CFD w/ Rusanov flux
CFD w/ Exact RS
CFD w/ NN+EE
CFD w/ NN
Exact solution

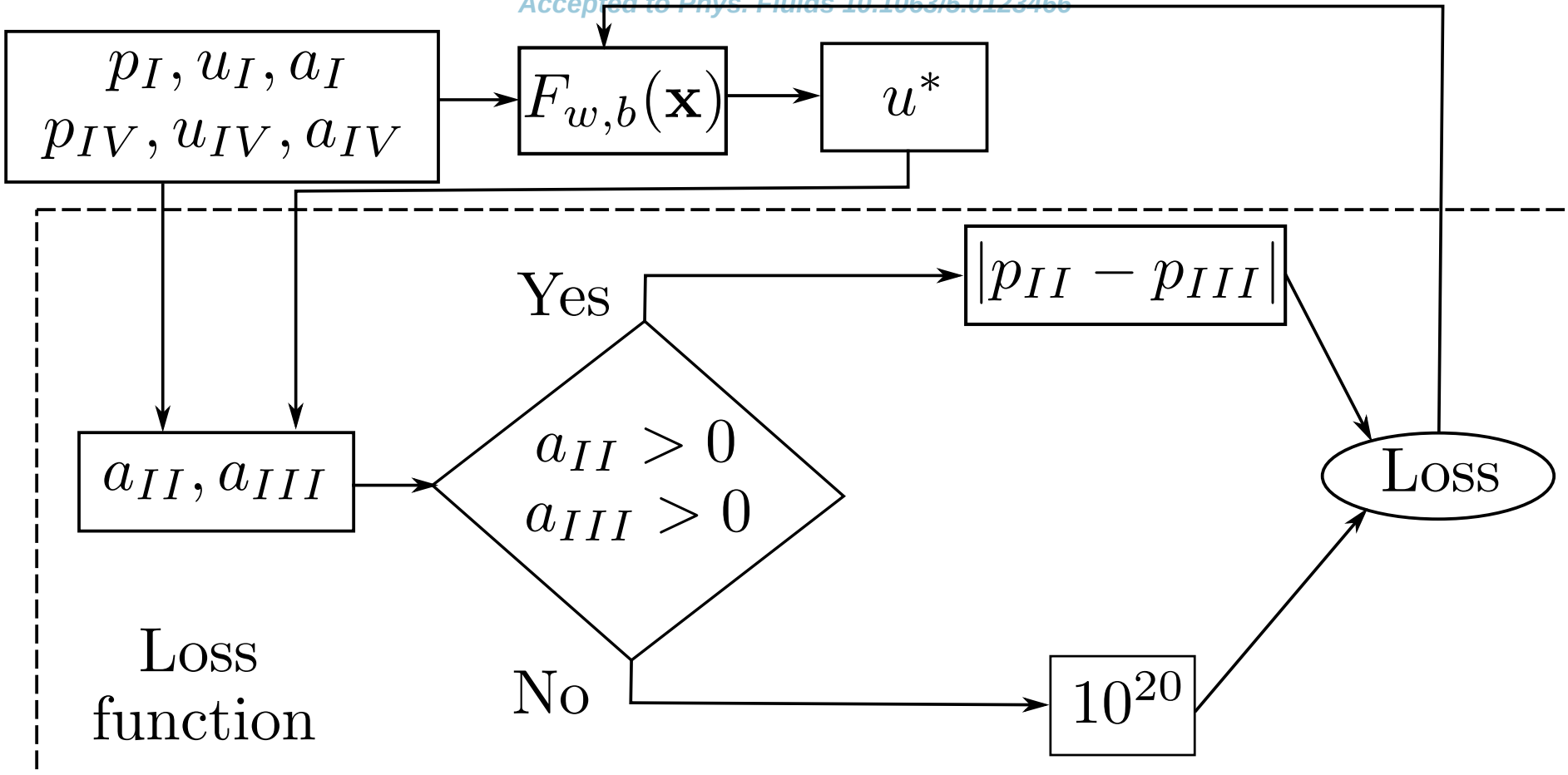

100 grid points
500 grid points
1000 grid points
CFD w/ Rusanov flux
CFD w/ NN
Exact solution
ρ [kg/m^3]
x
2000
1800
1600
1400
1200
1000
800
−1.0
−0.5
0.0
0.5
1.0


CFD w/ Rusanov flux
CFD w/ HLLC flux
CFD w/ Exact RS
CFD w/ NN+EE
Exact solution
p [Pa]
x


CFD w/ Rusanov flux
CFD w/ HLLC flux
CFD w/ Exact RS
CFD w/ NN+EE
Exact solution
ρ [kg/m^3]
x


CFD w/ Rusanov flux
CFD w/ HLLC flux
CFD w/ Exact RS
CFD w/ NN+EE
Exact solution
u [m/s]
x


CFD w/ Rusanov flux
CFD w/ HLLC flux
CFD w/ Exact RS
CFD w/ NN+EE
Exact solution
e [J/kg]
x



Accepted to Phys. Fluids 10.1063/5.0123466