PHYSICS-INFORMED NEURAL NETWORKS AND FUNCTIONAL INTERPOLATION FOR INITIAL VALUE PROBLEMS WITH APPLICATIONS TO INTEGRO-DIFFERENTIAL AND STIFF DIFFERENTIAL EQUATIONS

by

Mario De Florio

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF SYSTEMS AND INDUSTRIAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2022

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Mario De Florio, titled "Physics-Informed Neural Networks and Functional Interpolation for Initial Value Problems with Applications to Integro-Differential and Stiff Differential Equations"

and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Roberto Furfaro — Date: 11/30/22

Clayton T. Morrison — Date: 11/29/2022

Jianqiang Cheng — Date: 11/29/2022

Alejandro Salado — Date: 11/29/2022

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Date: 11/30/22

Dr. Roberto Furfaro
Dissertation Committee Chair
Professor, Systems and Industrial Engineering
Director, Space Situational Awareness

Contents

List of Figures

Abstract

This dissertation aims to show the development and adaptation of the Extreme Theory of Functional Connections (X-TFC) framework, a Physics-Informed Neural Network (PINN) combined with Theory of Functional Connections (TFC), to solve Initial Value Problems (IVPs) modeled by differential equations (DEs). In particular, this research focuses on two branches of IVPs: Linear Integro-Differential Equations and Stiff Systems of Non-Linear Differential Equations. The first types of IVPs are DEs where an integral of the unknown function is present, i.e., Integro-Differential Equations. To compute the integral of the unknown function two approaches are used. The first one is the analytical integration of the unknown function's approximation, which is represented by the TFC's constrained expression (CE). The second one approximates the definite integral of the unknown function via Gauss–Legendre quadrature. This approach is used to solve problems arising from the Boltzmann Partial Integro-Differential Equation for Transport, such as Radiative Transfer and Rarefied-Gas Dynamics problems. The second types of IVPs are large-scale Stiff Systems of non-linear ODEs, mathematical models governing a wide range of real-world problems such as chemistry, biology, epidemiology, engineering, neuroscience, financial systems, and so on. For this reason, recently, there has been a resurgence in the interest in developing new numerical methods capable of solving large time-scale stiff problems. Regular PINNs frameworks are proven to be not accurate (or even not capable) to solve problems when the dynamics are particularly complex (e.g., stiff). This is due to unbalanced back-propagated gradients during the model training. This dissertation shows how X-TFC with a domain decomposition technique overcomes the difficulties of regular PINNs in solving systems of stiff IVPs, and it outperforms the state-of-the-art numerical methods in terms of accuracy and computational times.

Chapter 1

Introduction

Differential Equations (DEs) represent one of the most effective tools for building mathematical and physical models. They are used in the most varied fields of applied sciences, such as physics, chemistry, biology, and epidemiology, to name a few. The study of DEs began with the introduction of the infinitesimal calculus by Newton and Leibniz in the 17th century and subsequently deepened by Bernoulli, d'Alembert, Euler, and Lagrange [1]. Among the more classical methods for solving DEs, we can refer to the Euler, Backward Euler, Crank-Nicolson, and Runge-Kutta methods, still widely used today. Over the years, many different algorithms have been developed, and many more are emerging. Recently, Neural Networks (NNs) have become widespread across many different scientific fields, and their applications for the solutions of DEs are no less. A new class of methods has been developed in the machine learning community, and it is represented by the Physics-Informed Neural Networks (PINNs) [2]. The methods underlying this category can be generalized as algorithms that include the physics of the problem into a data-driven representation of functional relationships underlying collections of input-output pairs. The DEs are added in the training of the NN as a penalty for the loss function. The flaw of the PINNs framework is that the solutions on the constraints are not analytically satisfied but numerically approximated. This means there are competing objectives during the PINN training: learning the DE solution on the domain points and the constraints. This leads to unbalanced gradients when the NN is trained via gradient-based techniques, and this affects the ability of PINNs to approximate the DE solution accurately [3]. The risk is getting stuck in limit cycles or diverging if multiple competing objectives are present. To overcome this issue, the Theory of Functional Connections (TFC) [4] can be adopted. The idea of TFC is to approximate the DE solutions via the so-called Constrained Expressions (CEs), which are made up of the sum of two terms: a free-function and a functional. Regardless of the choice of the free function, the constraints will always be analytically satisfied since they are embedded in the functional (chosen accordingly). The free-chosen function, thus, can be a single-layer NN trained via the Extreme Learning Machine (ELM) [5]. The Extreme Theory of Functional Connections (X-TFC) [6] method was born from this synergy between PINN and TFC and would enhance one's ability to solve challenging differential equations governing real-world physics phenomena, which is the main driving motivation behind this dissertation.

1.1 Scope of Research and Contributions

The scope of this dissertation is the development of the X-TFC framework to solve two major classes of initial value problems (IVPs) modeled by DEs, i.e., integro-differential equations and stiff systems of non-linear differential equations, which are among the most challenging to be solved with traditional numerical methods. The result of this research is the production of six articles published (and under review) in scientific journals, in which I am the lead author, presenting the methodology and real-world applications of X-TFC for the problems under consideration. My contributions to these manuscripts are the conceptualization, coding, validation, and drafting, with the invaluable help of the co-authors. The main novelty of my research journey is merging an interpolation technique (TFC) with an ELM training of the NN, which opens a new research pathway in applied mathematics, to push the family of PINNs methods to obtain ever more accurate solutions of DEs with extremely low computational times. It has been proved that such interpolation technique is able to decrease the generalization error by analytically satisfying the problems' constraints, and an ELM algorithm allows us to train a single-layer neural network in a fast and concise way, making PINNs methods attractive for real-time and online applications.

The first paper presented in Appendix A and published on *Mathematical and Computational Applications - MDPI* [7] is the result of a collaboration between my research team – led by Roberto Furfaro – and Daniele Mortari, who conceived the original idea of the TFC. This work aims to extend the TFC concept of CE for solving linear ordinary DEs (ODEs) subject to integral constraints and linear ordinary integro-differential equations. Although this extension of TFC proved to work well for the considered problems by overcoming the comparative numerical methods, its implementation for more arduous integral kernels would be too laborious or even impossible. To overcome this hurdle, a Gauss-Legendre quadrature approach can be used to approximate the integral of the unknown function and subsequently solve it via X-TFC. Two classes of Transport Theory problems were chosen to test this methodology due to my background in nuclear engineering and my advisor Dr. Furfaro's doctoral research on Radiative Transport. This choice led to the production of an article [8] on the solution of the Chandrasekhar's basic problem in radiative transfer in Appendix B, and a trilogy of articles on classical problems in rarefied-gas dynamics in Appendix C, D, and E: Thermal creep flow [9], Poiseuille flow [10], and Couette flow [11]. The X-TFC results compared with the most accurate benchmarks in the literature showed the accuracy of our methodology in solving physical phenomena modeled by the Boltzmann Integro-Differential equations for transport of radiation and particles. These works were made possible thanks to the exchange of ideas with my advisor Roberto Furfaro, the implementation of the codes in MATLAB with my colleague Enrico Schiassi, and the valuable advice of two recognized experts in the field of transport theory, Barry D. Ganapol and Liliane B. Barichello.

The motivation for producing the sixth paper, presented in Appendix F and published on *Chaos: An Interdisciplinary Journal of Nonlinear Science - AIP Publishing LLC* [12], lies in

the lack of a PINN framework capable of solving problems subject to great stiffness, and in the difficulty of traditional numerical methods in solving the most complex stiff problems. The main idea of this work is to employ a domain decomposition technique in the X-TFC framework to control and minimize the carryover of the generalization error along with the time domain. This new methodology has been tested to solve stiff systems of non-linear differential equations governing chemical kinetics reactions. The choice of this class of problems lies in the wide range of real applications, such as climate modeling and energy conversions. X-TFC proved to be efficient, accurate, and robust in solving all the faced problems, learning the solutions for huge time domains, where all other competitive methods fail.

This dissertation aims to tell the story of my research path, starting with a brief history of the development of humankind's scientific knowledge, up to a literature review of Physics-Informed Machine Learning state-of-the-art approaches and Data-Driven Machine Learning techniques for the discovery of unknown laws of physics, in Chapter 2. Then, Chapter 3 introduces the regular PINN framework, its drawbacks, and how the X-TFC method aims to avoid or mitigate regular PINN's limitations. Chapters 4 and 5 represent the body of my research, where all the journal articles are described, with introductions on the IVPs modeled by integro-differential and stiff differential equations, respectively. Finally, conclusions and future ideas for continuing my research in the direction I started are given in Chapter 6.

Chapter 2

Literature review

The history of the discovery of physics has very ancient roots, dating back to the dawn of human civilization. In the prehistoric era, knowledge of nature and its secrets was handed down mainly through oral tradition and was presented closely related to religion. This knowledge was essentially practical, therefore based on technological and mathematical skills, which will lead to the development of advanced civilizations such as the Indian, Mesopotamian and Egyptian ones, where mathematics would have originated, according to Herodotus [13]. The birth of writing allowed the conservation of knowledge and its transmission with greater accuracy. Many ancient civilizations thus collected systematic astronomical information in detail through *observation* of the sky and advanced calculation techniques [14]. The time could be measured using a stick planted in the ground, which allowed to observe the variations of its shadow produced by the different paths of the Sun during days and years. Through simple triangulations, it was possible to observe the variations in the height of stars or groups of stars and the irregular motion of the planets. Such simple but effective tools led to creating the first catalogs of stars and the first tables of planetary motions to make astronomical predictions in the third century BC. In those years in ancient Greece, the mathematician, geographer, and astronomer Eratosthenes formulated the method for calculating the size of the Earth [15].

Over time, the thirst for knowledge of the physics of nature was not quenched. Indeed, there was a widespread need to develop more sophisticated and technological tools for observing the nature of things among scientists. There was, therefore, a growing need to acquire more and higher quality *data*. At the beginning of the 17th century, the telescope was born, subsequently perfected by Galileo Galilei [16]. Thanks to this technology, Galileo improved the quality of data acquisition and discovered new celestial bodies and features of the solar system, previously unknown. He initiated a new way of describing and explaining the mechanisms that exist at the basis of certain events. Thus was born the “scientific method", which has as its first step the implementation of numerous experiments to obtain many observed data [17]. Riding the wave of new technological inventions, Kepler formulated his laws relating to the motion of the planets starting from the data collected by Tycho Brahe and relating to their apparent positions, in particular of Mars [18]. Kepler accurately determined

the orbit with which the Earth rotated around the Sun, which was almost circular with the Sun placed in an eccentric position. He also found that the Earth moved faster when closer to the Sun. From observations of Mars, he showed that the orbit had an elliptical shape. The scientist, therefore, developed a *data-driven* model for planetary motion, formulating his three laws – the famous "Kepler's laws of planetary motion" [19] – empirically, that is, based on experimental observation. However, Kepler did not explore the fundamental dynamics involved in planetary motion. For this, we will have to wait for the derivation of Newton's law of motion, in which a dynamic relationship between momentum and energy describing the underlying processes responsible for these elliptic orbits is presented [20, 21]. Newton produced a universal model that can predict the behavior of a system in a no-data-available scenario, published in the collection "Philosophiæ Naturalis Principia Mathematica", 1687 [22]. A decade earlier, in "De Methodis Serierum et Fluxionum" [23] he published a powerful tool to study many problems in the natural sciences and technology, still extensively employed in mechanics, astronomy, physics, and in many problems of chemistry and biology, now called Differential Equation (DE). DEs still belong today among the main Physics-based models that govern a physical phenomenon through mathematical language, and they can be used to accurately predict the state of a system by imposing determined constraints and boundary domains [24].

The study of DEs has been widely extended in the following centuries, and classical methods for their solutions have been developed and extensively applied. Among these methods, we can refer to the Euler, Backward Euler, Crank-Nicolson, and Runge-Kutta methods [25, 26, 27]. In the years, many different algorithms have been developed, and many more are emerging [28]. To date, humankind has reached the highest level of knowledge of physics and development of technology ever seen so far [1]. However, there are still situations in which the physics is unknown and it is difficult to retrieve it from data, or the phenomenon is difficult to observe, leading to scarce low-quality data availability. In both cases, Machine Learning (ML) algorithms can come to the aid of scientists to quickly and efficiently solve physics problems still unsolved and shrouded in mystery. The need for ML algorithms began in the last few decades when there was significant progress in understanding multi-scale physics for different science applications. ML approaches have been proven to help discover the physical laws and mathematical expressions underlying experimental data [29, 30]; however, purely data-driven models that fit observed data can bring inconsistency in the physics of the phenomena. To overcome this problem, ML algorithms can be informed by the knowledge of the physics as "informative priors". Depending on the mathematical and physical knowledge and on the data available, three possible categories of the problem under consideration can be defined, as represented in Figure 2.1.

[1] NB, this does not at all mean that it (humankind) is able to use it properly and responsibly.

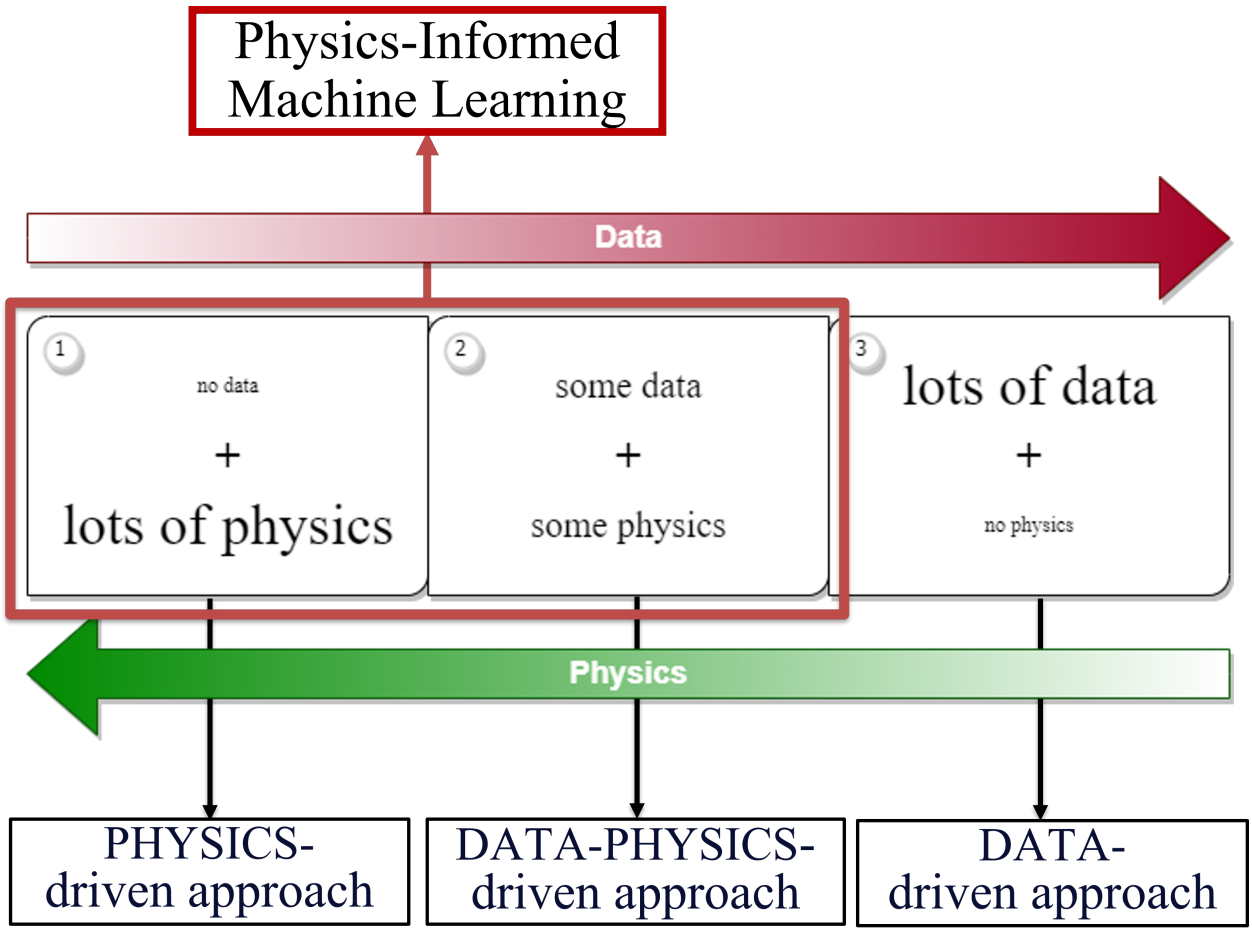


Figure 2.1: Schema of the three possible scenarios based on the availability of observed data and knowledge of physics.

The first category represents the case when no data are observed while the physics (differential equations) governing the problem is well known, therefore, the only known variables are the boundary and initial conditions. This represents a forward problem that requires a numerical method (physics-driven approach) to find the solution of DE. The second category is the most common case in engineering applications, and it occurs when only a few data are observed and only part of the physics governing the phenomena is known. For example, we know the differential equations governing a chemical reaction but not the exact values of the reaction rate constants. In this case, physics-informed machine learning methods integrate available data and the partially known mathematical physics model to learn both the DE's solution and the parameters (data-physics-driven approach). The third category regards the case where lots of observed data are available, but the physics model that can describe the phenomena is completely unknown (data-driven approach), and the whole physics, or some missing parts, needs to be learned [31, 32].

2.1 Physics-Informed Machine Learning

Creating a Physics-Informed Machine Learning (PIML) algorithm means introducing appropriate biases helping the learning process in identifying physically consistent solutions. Predictive models can not be built without assumptions; thus, creating a generalized ML model without appropriate biases is impossible [33]. To improve and accelerate the physics-informed learning process by introducing physics in the ML models, three pathways can be followed, individually or effectively combined, to yield a class of hybrid approaches. An *observational bias* is conceptually the simplest bias that can be introduced in ML, and it is done through the embedding of the observed data that reflect the physics that should generate them. As long as the available data cover the domain of the training points, ML algorithms have been proved to be accurate in the interpolation process between the scattered data [34, 35], learning functions, operators, and vector fields describing the structure of the data. An *inductive bias* consists of embedding some predictions in the form of physical laws into the architecture of the ML model to satisfy their mathematical constraints. The most common ML architectures that fall into this category are the Convolutional Neural Networks (NNs) [36] for computer vision and Graph NNs [37] for processing data best represented by graph data structures. These frameworks proved to work efficiently when the physics of the problem is simple and well defined. However, their extension to more complex and less understood physical problems can bring some difficulties in the learning task, and their embedding into the NN architecture can be tough to code. When the goal is solving DEs, the algorithm architecture can be modified to exactly solve the constraints. X-TFC [6] is one of the methods [38, 39, 40, 41] lying in this category. The third school of thought is introducing *learning biases* into the NNs without designing a specific architecture that enforces the physics law. The idea is to penalize the loss functions by a set of physical constraints (e.g., the conservation mass) while fitting the observed data. Deep Galerkin method [42] and Physics-Informed NNs (PINNs) [2] are good examples in this respect.

PIML models have been already extended and applied to different physical and mathematical fields, such as fractional PDEs [43], stochastic PDEs [44], and conservation laws [45], and for multi-scale problems a domain decomposition is performed. In this way, a different NN represents each subdomain and is assigned to a different GPU [46, 47]. As explained before, PIML can combine data with the knowledge of physics, even when both are poor. For example, in forward and inverse problems when the constraints are not defined, and parameters of the PDEs are unknown (where many numerical methods fail), works like Ref. [48] show the capabilities of PIML frameworks in finding robust solutions even with low resolution and substantial noise in the observation data. When tackling problems in a small data regime, PIML finds its strong advantage in the generalization of the solutions by embedding all that is known about physics.
For some real-world problems involving the evolution forecast of multi-physics systems, a quantification of uncertainties is required. In particular, three different sources of uncertainties can be defined: the physics, the data, and the learning models.

I. When the source of uncertainty is the physics, we are dealing with stochastic DEs because of the randomness of the parameters. To quantify this uncertainty, Ref. [49] proposes a methodology for learning deep NNs surrogate models based on a parameterization of the Deep NN structure, and it is demonstrated with a problem on uncertainty propagation in a stochastic elliptic partial differential equation with uncertain diffusion coefficient.

II. The uncertainty can arise from the noise and the gaps in the observed data. To compensate these deficiencies during data acquisition, a valid method that can be the Bayesian approach. In this regard, a PIML framework for encoding physical laws described by PDEs into Gaussian process priors for nonparametric Bayesian regression was developed [50]. This algorithm aims to infer solutions to time-dependent and nonlinear PDEs and it effectively quantifies and propagates uncertainty due to noisy initial or boundary data. Another framework named Bayesian PINNs (B-PINNs) [51] was proposed to quantify uncertainty and to improve the accuracy of PINNs. This method uses a Bayesian NN, subject to the PDE constraint as a prior, and Hamiltonian Monte Carlo (or the variational inference) as an estimator for the posterior.

III. Finally, uncertainties coming from the learning models, such as training, generalization, and approximation errors, are generally difficult to quantify. The first attempt to quantify the parametric uncertainty in the PINNs was made by by employing the arbitrary polynomial chaos (aPC) expansion for the representation of the stochastic solution [52]. To solve forward and stochastic inverse problems, the authors use observed data to build a set of arbitrary polynomial basis and learn the modal functions of the aPC expansion through deep NNs encoding the underlying stochastic DE.

An interesting application of physics-informed learning for ill-posed inverse hidden fluid mechanics problems applied to the heat flow over an espresso cup [53]. This problem is considered ill-posed since no constraints are provided, and inverse, since information about the physics is sought. As previously mentioned, physics-informed NN can be trained by using both some observed data and some knowledge of the physics underlying the phenomena. For this problem, the observed data is given by the temperature (or density) field acquired using six cameras and processed with LaVision's Tomographic BOS software, as shown in Figure 2.2.

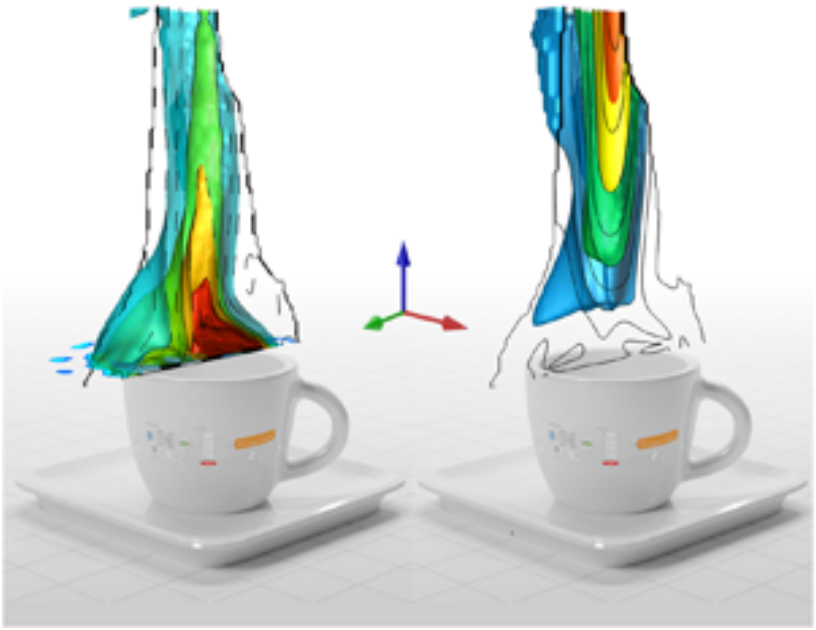


Figure 2.2: 3D temperature (left) and pressure (right) fields derived and reconstructed based on the 2D images from the six cameras. [53]

On the other side, the information of the physics is given into the PINN by the Boussinesq approximation of the incompressible Navier–Stokes equations and the corresponding heat transfer equation. Thus, data and equations are part of the loss that needs to be minimized in the training. The three space coordinates and the time are used as training inputs of the NN, giving the velocity and pressure fields as outputs. The results of this approach shows the high accuracy and flexibility in dealing with observed data of various fluid mechanics problems [53]. Thus, PINNs can become a promising direction in experimental fluid mechanics. Although the PINNs term appeared for the first time in the paper from Raissi et al. [2] in 2019, the first attempt made to use NN to tackle DEs dates back to 1992 by Psichogios and Ungar [54], and subsequently by Lagaris et al. [38]. The schema in Figure 2.3 shows the loss functions of the NNs for the physics-driven approach, data-driven approach, and data-physics-driven approach (for both forward and inverse problems).

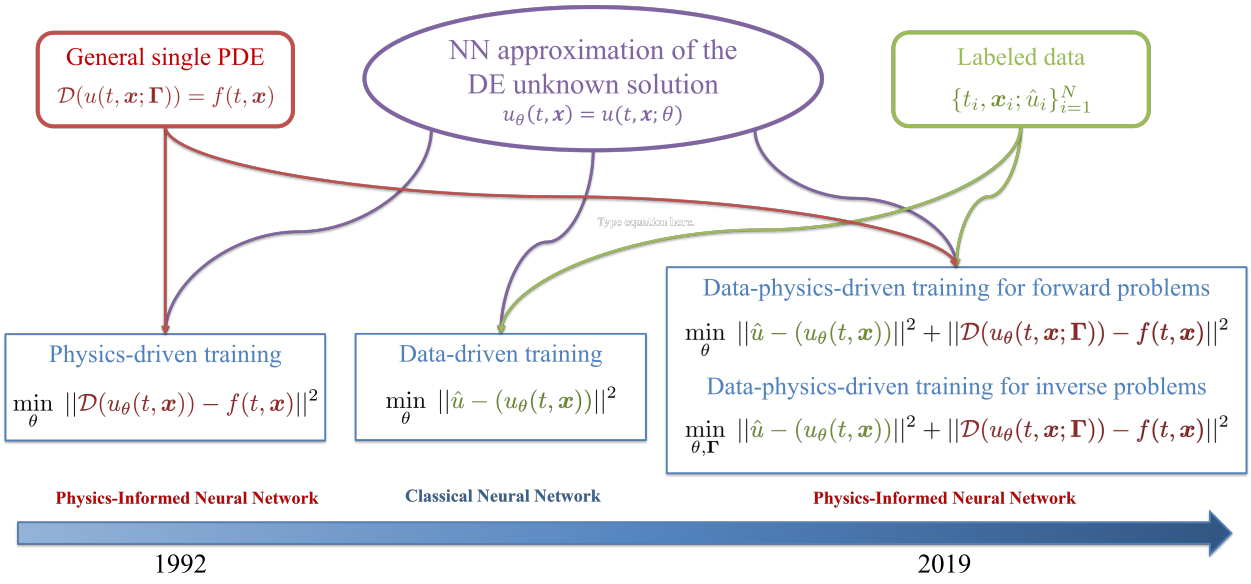


Figure 2.3: Schema of the NNs loss functions for the different approaches.

2.2 Data-Driven Machine Learning

The Scientific Method represents a Human Learning algorithm in which a scientist observes, questions, hypothesizes, predicts, tests, and eventually iterates this process. Machine Learning is a branch of Artificial Intelligence and Computer Science that focuses on using algorithms powered by observed data that mimic human behaviors in the learning phase and improve with experience. The analogy of these two learning algorithms lies in their cognitive systems to build knowledge by interacting with the surrounding environment and learning from it. In the history of science, it may have taken decades for scientists to formulate new physical laws through countless observations, experiments, and failures; or even centuries for subsequent groups of scientists to improve and redefine past studies that led to a final formulation of a generalized law. The human mind has limited computational power and memory, but it has the shrewdness and the intelligence to create machines that can analyze data and work out hypotheses for it. Today, ML algorithms are the protagonists of many technological advancements. They have been used to automatize the analysis of observed data, which saved time and effort for scientists in various fields of science, such as the identification of exoplanets [55], exploring galaxies [56], or to stay grounded, to help in creating the new industry of self-driven vehicles [57]. Supervised ML is widely used today in medicine as a classification algorithm to help medical doctors in disease diagnosis, for example, by predicting whether a tumor is benign or malignant [58]. In short, it is clear that ML has immensely helped the human being in his path toward a further technological and social evolution. Therefore, a question arises spontaneously in this circumstance. Can ML help scientists better understand the physics around us? Can ML help humanity to discover new physical laws of phenomena that we can observe but fail to conceptualize through mathematical language?

This is a recent, pervasive question in the scientific community yet not resolved, but several possible avenues have been opened. One of the first attempts to extrapolate governing equations from observed data is presented in the well-known work by Brunton et al. [31], in which the authors propose a new school of thought for dynamical system discovery problem from the perspective of sparse regression [59] and compressed sensing [60]. In particular, they take advantage of the fact that most physical systems are described by only a few relevant terms governing the dynamics, making the governing equations sparse in a high-dimensional non-linear function space. This method named SINDy – Sparse Identification of Nonlinear Dynamics – depends on the choice of the candidate non-linear functions library and the availability and quality of the data. Thus, it is not a generalized method and works better if guided by available knowledge of the phenomena under study. For example, given the trend of the observed data, one can approximately understand if it is a trigonometric or polynomial trend and build the library accordingly. SINDy has shown its capability in identifying non-linear dynamical systems from data without previous assumptions of the forms of the differential equations governing the phenomena. A particular example that proves its power is the discovery of the PDE for vortex shedding behind an obstacle. The greatness of this result lies in the fact that this PDE formulation required about 30 years and a team of fluid-dynamics experts. Of course, SINDy is not the only attempt to retrieve governing equations from data. Another method has been proposed by Raissi et al. [61], which aims to learn general parametric linear equations from data containing noise by using multi-output Gaussian process regression [62]. In its form, the method is designed to be generalized for linear equations, but it cannot deal with non-linear ones. However, this task may be addressed by transforming the non-linear equation in a system of linear equations and then finding some specific non-linear operators.

Another new and interesting work published by Purdue University researchers proposes a new ML method to identify physical laws from data [63]. The basics of this method lie on the concept of parsimony, for which ML neural networks are often too large and complex to efficiently learn a simple explanation of an easily observed phenomenon. Based on this, the authors aim to enforce parsimony on the NN through stochastic optimization to find a better balance between simplicity and accuracy, thus, effectively retrieving interesting physics from data. In particular, these so-called Parsimonious Neural Networks have been trained with observed data to discover the Lindemann melting law and Newton's second law of motion. These physics laws, which required decades for Newton and Lindemann to be found, took only a few days for Purdue's supercomputers to produce models that demonstrated them.
All the approaches mentioned above using Neural Networks (excluding SINDy) require prior knowledge of the DE's form to discover physics laws, or they will fail even for the most straightforward set of DEs. Although several techniques have been found to be accurate and reliable for identifying already known physical models, the challenge of attempting to predict and model unforeseeable phenomena remains. Questioning where artificial intelligence can be applied to reveal the most unsolved physical doubts can be a good first step that a

scientist must face. Will ML help us to explain the flyby anomaly? Can ML help us to generalize the unexplained empirical Koide formula? Will ML one day be able to find a singular, all-encompassing, coherent theoretical framework of physics that fully explains and links together all physical aspects of the universe? [64]

2.3 Personal Perspectives

Observed data and knowledge of physics are two central pillars of ML algorithms for modeling and forecasting the dynamics of multi-physics and multi-scale systems. After a brief historical overview of the development and contributions of these two factors in the science evolution, three main categories of study problems have been identified. The first category represents the case in which, for a given problem, the knowledge of physics is complete but there is a total lack of data. For this case, the physics model is well known by a set of governing DEs, and the data can be obtained by solving the latter with an explicit or implicit numerical method (where a closed solution is not available). In the second category, where we want to trace information from physics with the help of few experimental data. This is the case where most of the engineering problems belong, and it is the typical case of DE parameters discovery, where the inference of parameters governing a dynamical system is sought. Physics-Informed Machine Learning algorithms aim to solve these problems by leveraging the empirical data with the available knowledge of physics to penalize or regularize the Neural Network in use. The third category represents the scenario in which there is no physics consciousness to explain a particular behavior of a system or an observable phenomenon whose acquired data are challenging to interpret.

The scientific community is pooling efforts to efficiently create and refine algorithms to address all the aforementioned categories' possible problems. To date, there is no general and perfect algorithm, and perhaps never will be. However, there are several algorithms with particular strengths developed to overcome specific obstacles in physics and mathematics. The schools of thought carried out by different research groups are many and varied and I believe that combinations of these in a targeted and rational sense is the way forward for the trend toward perfecting machine learning in helping scientists in their modeling and forecasting purposes. Particularly fascinating is the position of the third category in the timeline of scientific evolution. It represents the birth of critical scientific thinking, characterized by the total absence of knowledge of physics and mathematics and by the observation with the naked eye (or through rudimentary tools) of everyday phenomena of the surrounding world. Paradoxically, it also represents the contemporary epoch of scientific evolution, characterized by a vast knowledge of physics and the technological ability to acquire high-resolution data in large quantities. Thanks to the immutable nature of human curiosity in both eras, the frontiers of science are infinite, and we push ourselves to research and investigate what is unknown to us and the physics known so far. The difference is that today we can do it with a telescope and a Python code, instead of with a gnomon and an hourglass.

Chapter 3

PINNs and X-TFC

As mentioned above, the first category of Figure 2.1 represents the case of differential equations (DEs) subject to particular constraints. DEs represent one of the most effective tools for describing physical phenomena through mathematical models, and they link one or several unknown functions with their derivatives. Usually, the unknown function is a physical quantity and its derivative is its rate of change. For example, let's consider the following general differential equation

$$\frac{d}{dt}y(t) = f(t, y(t)) \qquad \text{subject to} \qquad y(t_0) = y_0$$

This is the case of initial value problem (IVP) where we study the evolution of the unknown quantity starting from a specific initial condition. Not for all DEs there is an available analytical solution, thus an efficient and reliable numerical method is required.

In the last century, many numerical methods have been developed, adapted, and improved to solve the most challenging DEs modeling real-world problems. Among these, we can mention the most classic Euler's method [65], Runge-Kutta methods [66], and Milne's Predictor-Corrector method [67]. For high-dimension Partial DEs (PDEs), the well known finite difference method [68], finite element method [69], and finite volume method [70] have proven to be particularly effective, even for complex geometries. However, like all things, these methods have some limitations. A feature in common is the need to adopt a domain discretization or a mesh domain decomposition. This implies the evaluation of the solution only on the discretization points, or mesh-cells (or training points, according to the machine learning terminology). To evaluate the solution on other points of the domain (test points) one can resort to an interpolation or to a re-computation of the problem. An interpolation is a numerical estimation of the solution in a point between two training points, in which the solutions are already numerically approximated. Thus, this technique will inevitably lead to a loss in accuracy. The re-computation of the problem by adding the new test points among the set of training points will lead to an increase of computational efforts, that in some cases can not be affordable. Another drawback is that once the numerical solution is computed,

we may need to find its derivative or integral, where another numerical approximation is required. This, again, would bring a further loss of accuracy. Last but not least, all these methods are designed to solve forward problems only. To tackle inverse problems, they need to be modified or coupled with other methods, and the code implementation can be intense, which is not a trivial task. Now the question naturally arises: how can we avoid, or at least mitigate, these limitations? An answer can be: Neural Networks (NNs).

3.1 Physics-Informed Neural Networks

Figure 3.1 shows the PINN algorithm for solving a PDE subject to initial and boundary constraint. The PDE's independent variables (in this example x and t) are the inputs of the deep NN, which approximate the solution (u) of the PDE. The loss function includes the residual of the PDE and the initial and boundary constraints, and the goal of the NN is to find the optimal set of NN parameters θ (which are the input weights, bias, and output weights) that minimizes the loss function, by using gradient-based methods [71]. The main flaw of this framework lies in the use of a deep NN for the approximation of the solution, due to the complexity of such a network, which leads to a high computational cost, can cause divergence, and treats linear problems as non-linear problem, because of the expression of the activation function $\sum_{i=1}^{L} \beta_i \sigma(\mathbf{w}_i^T \mathbf{x} + b_i)$ in the i^{th} neuron. Another negative aspect of this framework is the inclusion in the loss function of both the PDE residual and the constraints, since they represent an unsupervised learning and supervised learning, respectively. Thus, we are facing the presence of competing objectives from which gradient pathologies can arise, causing the training failure in learning the PDE constraints [3].

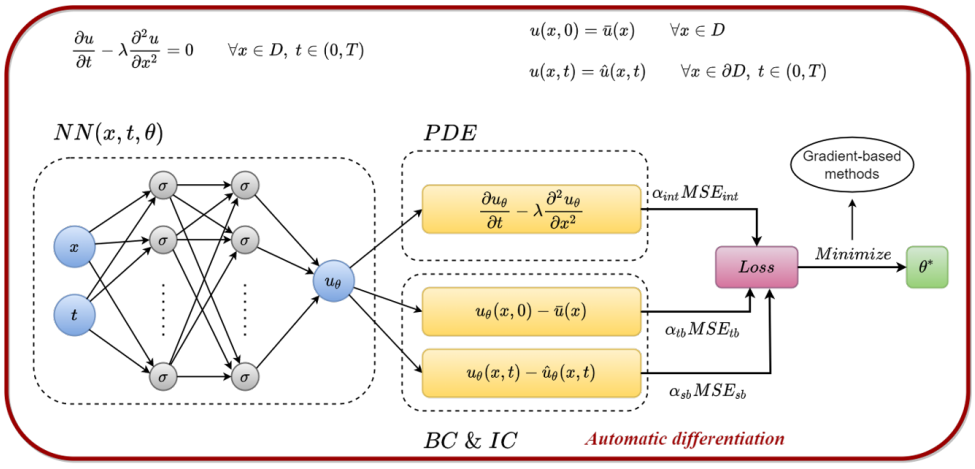


Figure 3.1: Schema of the regular Physics-Informed Neural Networks framework for solving a Partial Differential Equation subject to initial and boundary conditions.

3.2 Extreme Theory of Functional Connections

The Extreme Theory of Functional Connections (X-TFC) is a method I have co-developed with my research team [6], to speed up and optimize the training process of the NN in learning the solution of DEs. The novelty lies on the introduction of two main features into the regular PINN framework. To approximate the solution of the DE, we make use of the constrained expression (CE) from the Theory of Functional Connections [4], which is made by the sum of two terms. The first term is a a free-chosen function, while the second term is a functional in which the DE constraints are embedded. No matter what free-chosen has been selected by the user, the CE will always analitically satisfies the initial and boundary conditions. This, will avoid the presence of competing objectives in the loss function, as mentioned in the previous section. The second novel feature is the choice of the free-chosen function, which is a single layer forward NN, trained via Extreme Learning Machine (ELM) algorithm [5]. According to ELM, the NN input weights and bias are randomly selected before the training process, thus they become NN hyperparameters. This extremely simplify the structure of the network speeding up the training process, since the only NN unknown parameters are now the β output weights, which can be found by minimizing the loss function with least-squares methods.

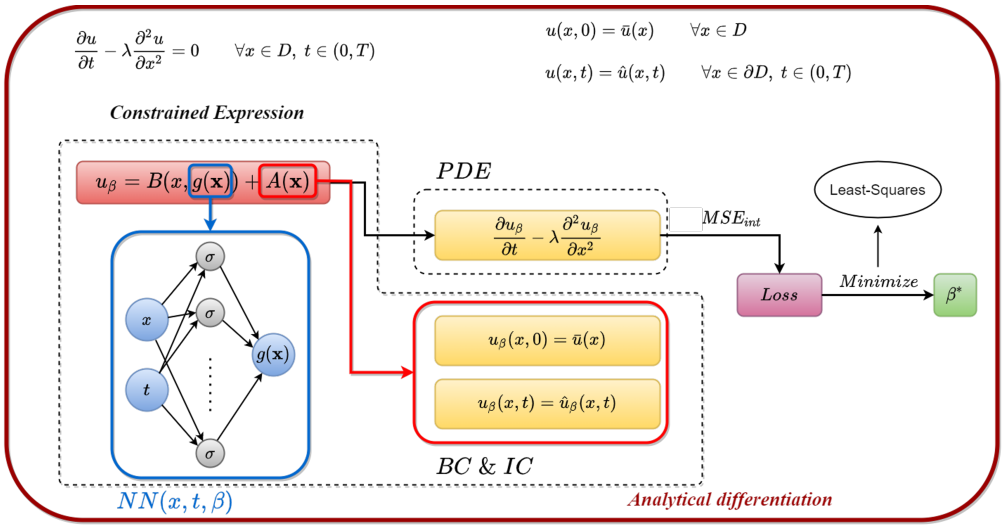


Figure 3.2: Schema of the Extreme Theory of Functional Connections framework for solving a Partial Differential Equation subject to initial and boundary conditions.

Chapter 4

X-TFC for Integro-Differential Equations

Integro-Differential Equations (IDEs) are differential equations in which the unknown function appears both in an integral and derivative form and often appear in many scientific disciplines. Historically, the first studies on Integral Equations can be traced back to Abel's problem [72], Lotka's population analysis [73], Fredholm Theory [74], and Volterra Integral Equation [75]. These studies have been of fundamental importance for the development of mathematical models for problems in mechanics, biology, and economics. Often, for real-world problems, an analytical solution of IDEs is not available, and a numerical method is required, such as Laplace transform method [76], Wavelet–Galerkin method [77], and Euler-Chebyshev methods [78].
This chapter shows how X-TFC can be adapted and employed to solve IDEs. In particular, the adaptation of the TFC methodology to solve IDEs and DEs subject to integral constraints is presented in Section 4.1, and the application of X-TFC to solve some Transport Theory problems is shown in Section 4.2.

4.1 TFC for Integro-Differential Equations (De Florio et al., 2021)

This section aims to introduce the paper [7] in Appendix A:

> De Florio et al. "Theory of functional connections applied to linear odes subject to integral constraints and linear ordinary integro-differential equations." *Mathematical and Computational Applications* 26.3 (2021): 65.

This methodology paper shows how TFC can perform functional interpolation to solve

(i) linear ordinary differential equations subject to integral constraints, and

(ii) linear integro-differential equations.

For the first kind of problem (i), we can find three different sets of constraints: definite integral constraints, integral and linear constraints, and mixed constraints. Both cases show

how to accommodate the integral constraints into the TFC's CE. In particular, after building the CE accordingly to TFC framework [4], we need to retrieve the coefficients η_i (where $i = 1, ..., N$, and N is the number of constraints) by imposing the constraints in the CE, thus by analytically integrating the latter to embed the integral constraint. Similarly, we show how to solve the second kind of problem (iii), where the unknown function in the differential equation appears under the integral sign. Since we aim to test both TFC and X-TFC in solving some toy problems, to evaluate the integral of the CE for the two different methodologies, we need to compute the integration of both types of CE's free-chosen function, such as Chebyshev Polynomials and NN's activation function. This integration can be more or less arduous to implement based on the kernel of the integral (i.e., the function that multiplies the unknown function under the integral). For each problem solved, an analytical solution was available to test each method's accuracy, which has been compared with a state-of-the-art method based on the differential transformation [79]. For the more straightforward problems of type (i) (results of Ref. [79] not available), both TFC and X-TFC computed the solution with error machine accuracy (absolute error with respect to the analytical solution of the order of $10^{-15/-16}$) with a computational time of the order of 10^{-4} seconds. For the more challenging integro-differential equations (ii), TFC and X-TFC outperformed the results in Ref. [79] in terms of accuracy and computational efficiency. As stated in the Discussion section of the paper, the significant difference in accuracy among problems solved is due to the difference in challenging the kernel functions.
My contributions in this work are the conceptualization of the idea of extending the TFC's constrained expression for integro-differential equations, creating the MATLAB functions that analytically integrate the activation functions of the neural network, coding the TFC and X-TFC algorithms to solve test problems published in the literature, comparing our results with other available numerical methods results, and writing the journal paper.

4.2 Transport Theory Problems

As explained in Section 4.1, the complexity of the integral kernel function can determine the degree of the accuracy of the solution of the integro-differential equation and can even result in a too arduous implementation of the analytical integration of the kernel times CE's free-chosen function (i.e., Chebyshev polynomials or NN's activation function). In more challenging real-world physical phenomena, such as physics problems in Transport Theory, the complexity of the integral kernel would make the integration of the CE prohibitive. For example, in the Radiative Transfer problem discussed in Section 4.2.1, the integral kernel represents the phase function expressed in terms of Legendre polynomials, while for the Rarefied-Gas Dynamics problems in Section 4.2.2, the integral kernel is a weight function defined as an exponential function. To tackle these problems, the integrals can be approximated with Gauss-Legendre quadrature

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{N} w_i f(x_i)$$

as shown in the following sections.

4.2.1 Radiative Transfer (De Florio et al., 2021)

This section aims to introduce the paper [8] in Appendix B:

> De Florio et al. "Solutions of Chandrasekhar's basic problem in radiative transfer via theory of functional connections." *Journal of Quantitative Spectroscopy and Radiative Transfer* 259 (2021): 107384.

This paper aims to show the capability of TFC in solving the Chandrasekhar's basic problem in radiative transfer arising from the Boltzmann Integro-Differential Equation. This equation describes the propagation behavior of electromagnetic radiation (considered as particles going through several physics processes, such as absorption, scattering, and emission) through a medium. In particular, the Chandrasekhar's problem models the behavior of scattered photons through a finite slab illuminated by incident light on the surface. The historical method for solving such an equation for photons transport was the Discrete Ordinate Method (DOM), proposed by Wick [80], extended by Chandrasekhar [81], and upgraded and improved over the years [82, 83, 84].

In this paper, after an overview of the state-of-the-art DOM methods and an introduction to the TFC framework to solve general ordinary differential equations and a simple case of radiative transfer problem in the two-stream approximation, we presented the physics problem formulation of the radiative transfer equation, of fundamental importance to apply TFC and compare our results with published benchmarks. In its original form, the Chandrasekhar's problem is a boundary value problem since we have two conditions on the two surfaces of the slab. The unknown function is the intensity of the photons as a function of the slab's optical depth, scattering angle, and azimuthal angle, and it is defined as the sum of scattered term and unscattered term. The scattered term (or diffuse term) can be expressed as a Fourier expansion with L components, whose m^{th} component satisfies another partial integro-differential equation (see Eq. (36) in Appendix B), which can be split into two integro-differential equations with one constraint per each. To notice that we transformed a boundary value problem into two initial value problems. Then, by approximating the integrals with a Gauss-Legendre quadrature, we transform two partial differential equations into a system of $2 \times N$ ordinary differential equations, where N is the number of Gauss-Legendre discretization points.

After reformulating the problem, the TFC formulation can begin. To notice that the independent variable of our $2 \times N$ differential equations is the slab optical depth τ that ranges from 0 to a certain τ_0., and that regular TFC plans to use orthogonal polynomials (e.g., Chebyshev) in the range $[-1, 1]$ as free-chosen function, thus a linear mapping is required. Once built the CEs accordingly, we replace them in our $2 \times N$ system of differential equations, where all the terms are known but the polynomials coefficient ξ. Thus, we rearrange the terms in the $2 \times N$ system to get a linear system in the form $A\xi = B$, which can be

solved for ξ via a least-squares method. By doing this, we are able to compute the value of the CEs, and then the solution of our problem. This needs to be done for all the L Fourier components that then will be used to compute the diffuse term (see Eq. (60) in Appendix B). Our solution is obtained for specific scattering angles, which are the Gauss-Legendre Quadrature nodes. However, in many real-world applications, we must compute the solutions in arbitrary scattering angles. For our case, we need to compute the solutions in the same scattering angles of the literature published benchmarks to compare our results. Thus, we used a post-processing (performed via TFC, see section 5 of appendix B).

To validate our results, we solved the radiative transfer equation with two different phase functions, such as Mie Scattering (9 Fourier components) and Haze L problem (83 Fourier components), for both one and two-angle cases. The algorithm described above has been implemented in both Matlab and Python platforms, and the simulation ran with an Intel Core i7 - 9700 CPU PC with 64 GB of RAM. All results in the paper demonstrate the precision and the efficiency of TFC by achieving all the digits (8) of accuracy of the most accurate benchmarks in literature.
My contributions in this work are the formulation of the problem to reach a convenient form of the integro-differential equation to be solved by X-TFC, the X-TFC formulation to transform the constrained problem into an unconstrained linear system to be solved via least-square, coding the X-TFC algorithm to solve two challenging problems in the literature and comparing our results with published and recognized benchmarks, and writing the journal paper.

4.2.2 Rarefied-Gas Dynamics

The second chapter of my research on Transport Theory problems is represented by the class of Rarefied-Gas Dynamics in the Bhatnagar–Gross–Krook (BGK) approximation. As for the radiative transfer, the rarefied-gas flows studied in this dissertation arise from the Boltzmann Integro-Differential Equation. The rarefied-gas dynamics describes a gas flow where the molecules' mean free path λ is greater than the size d of the chamber under test. Usually, a gas can be considered rarefied if the its Knudsen number $Kn = \frac{\lambda}{d} > 1$. In the next Sections 4.2.2.1, 4.2.2.2, and 4.2.2.3, three classical flows of a gas in rarefied state are presented: Thermal creep flow, Poiseuille flow, and Couette flow, respectively.

4.2.2.1 Thermal creep flow (De Florio et al., 2021)

This section aims to introduce the paper [9] in Appendix C:

> De Florio et al. "Physics-informed neural networks for rarefied-gas dynamics: Thermal creep flow in the Bhatnagar–Gross–Krook approximation." *Physics of Fluids* 33.4 (2021): 047110.

In this first paper of the trilogy on Rarefied-Gas Dynamics, a thermal creep flow in a micro-channel is studied. In the thermal creep flow, a rarefied gas is confined between two parallel

walls, and its flow is driven by a temperature gradient in a direction parallel to the walls of the channel. It usually occurs between long parallel walls very close to each other (in the order of μm or nm), flowing from the colder to the warmer molecules until an equilibrium temperature is reached. In particular, the classical BGK model assumes that the molecules' collision frequency is constant and non-dependent on the velocity. The thermal creep phenomenon was initially studied by Knudsen [85], Maxwell [86], and Reynolds [87], and subsequently mathematically modeled via Boltzmann Integro-Differential Equation by Cercignani [88]. Among the classical numerical methods for its solution we can mention the DOM methods [80, 82, 83, 84], and more recently the Direct Simulation Monte Carlo (DSMC) for both monoatomic and polyatomic gases [89].

This paper provided an overview of the classical methods for solving the thermal creep flow and an introduction to the PINNs and X-TFC frameworks to tackle DEs. To formulate the physics problem, we started from the definition of the full non-linear Boltzmann equation for the gas atom space and velocity distribution. By applying the BGK approximation, we assume a constant collision frequency equal to the molecular mean free path, independent of the velocity. A Maxwell–Boltzmann distribution is assumed to define a diffusely and specularly reflecting wall, which we can use to leverage the computational cost of the simulation by computing our solution on half domain only. Another assumption we can make for the thermal creep flow is that the fluid has zero velocity relative to the wall (no-slip condition). We finally get a partial integro-differential equation (see Eq. (37) in Appendix C) depending on the half-channel width $\tau \in [0, a]$ and the fluid velocity $u \in [-\infty, \infty]$. Here, we need to do a linear mapping of both the independent variable. Since we need to approximate the integral in du with a Gauss-Legendre quadrature, we use the logarithmic transformation $u = -\log(|\mu|)$, with $\mu \in [-1, 1]$. Also, τ will need to be mapped according to the domain of the CE's free-chosen function. Since the constraints of our problem are defined for positive and negative values of μ (see Eq. (48) in Appendix C), it is convenient the split the equation into two partial differential equations with one constraint per each. After approximating the integral with Gauss-Legendre quadrature, we obtain a system of $2 \times N$ ordinary differential equations. To solve the thermal creep flow problem in this form, we can build the CEs accordingly, expanding the free-chosen function with a single-layer NN trained via the ELM algorithm. By replacing the CEs into our $2 \times N$ system and reorganizing the terms, we can obtain the linear system $A\beta = B$, solvable with a least-squares method. Once the NN's output weights β are computed, we can build the solution of the unknown function, which represents the velocity momentum of the flow. We can compute other quantities of interest from this physical quantity to compare our results with available benchmarks.

In the results section of the paper, we show how to compute the macroscopic velocity profile and the mass flow rate for the thermal creep flow as a function of the half-width of the channel a and the accommodation coefficient α, which describe the interaction between gas and wall. Our results are then reported for a wide range of Knudsen numbers, achieving up to 8 digits of accuracy for all the benchmark cases published in the literature. The problem

has been coded in the MATLAB R2019a platform, and all our simulations ran with an Intel Core i7–9700 CPU PC with 64 GB of RAM.
My contributions in this work are the conceiving of the idea of solving the thermal creep flow to test the ability of X-TFC to solve rarefied-gas dynamics problems, the formulation of the problem to reach a convenient form of the integro-differential equation to be solved by X-TFC, the X-TFC formulation to transform the constrained problem into an unconstrained linear system to be solved via least-square, coding the X-TFC algorithm to solve the thermal creep flow problem and obtain numerical solutions of some physical quantities of interest in the literature, comparing our results with published and recognized benchmarks, and writing the journal paper.

4.2.2.2 Poiseuille flow (De Florio et al., 2022)

This section aims to introduce the paper [10] in Appendix D:

> De Florio et al. "Physics-Informed Neural Networks for rarefied-gas dynamics: Poiseuille flow in the BGK approximation." Zeitschrift für angewandte Mathematik und Physik 73.3 (2022): 1-18.

In this second paper of the trilogy on Rarefied-Gas Dynamics, a Poiseuille flow in a microchannel is studied. In the Poiseuille flow, a rarefied gas is confined between two parallel walls, and its flow is induced by a pressure gradient along the walls of the channel. The Boltzmann integro-differential equation has been the most used mathematical model to study this classical rarefied-gas dynamics flow [88], and subsequently, several approximations have been implemented (i.e., BGK [90], and Shakhov [91]). One of the first numerical methods used to solve the Poiseuille flow problem was the DOM method of Chandrasekhar [81], which subsequently has been improved to solve many other problems in transport theory. Among these upgrades we can mention the S_n, C_n, P_N, and F_N methods [83, 82, 92, 93], with the most recent adding and doubling method proposed by Ganapol [94] that achieved the benchmark with the highest accuracy to date.

In this paper, we aim to adapt the X-TFC method to solve the classical and challenging Poiseuille flow of a gas in a rarefied state with the BGK approximation. After introducing the PINNs methodology and why it can bring advances in the field of numerical simulation for rarefied-gas dynamics, the X-TFC approach for solving general linear ODEs is presented. Moreover, a particular focus is given to the choice of CE's free function, explaining the difference between Chebyshev polynomials and single-layer NN using an activation function. It is shown why an expansion of Chebyshev polynomials can be considered a NN by selecting the NN input weights equal to 1 and using the Chebyshev polynomials instead of the activation functions in each NN neuron, resulting in the so-called "Chebyshev Neural Network" (ChNN) [95, 96]. Following the formulation of the problem presented by Siewert et al. [97], our problem reduces to a partial integro-differential equation depending on the space and velocity of the fluid and subject to certain boundary conditions (see Eqs. (4.6) and (4.7) in Appendix D). To transform this boundary value problem into two initial value problems, it

is convenient to split the PDE into two PDEs, with one condition per each. The velocity independent variable u is subject to a logarithmic transformation to map its range to $[-1, 1]$ in order to approximate the integrals on the velocity field with Gauss-Legendre quadrature, and the new velocity variable μ is discretized according to the Gauss-Legendre quadrature scheme. Finally, we obtain a set of $2 \times N$ ordinary differential equations to be solved via X-TFC. Depending on the domain of the activation function chosen as CE free function, the space independent variable needs to be mapped accordingly. Subsequently, the $2 \times N$ CEs are built and substituted into our problem, which now becomes unconstrained. The NN output weights β (which are the only unknown parameters of the linear system $A\beta = B$ are computed via a least-squares method. When using a ChNN as a CE's free function, we found it more efficient to find the least-squares solution of the vector β by computing $\beta = (A^T A)^{-1}(A^T B)$, while when using an activation function σ it is convenient using the MATLAB function `lsqminnorm` for the minimum norm least-squares solution to a linear equation. With the retrieved β outputs, we can compute the velocity momentum through the CEs, and then compute the macroscopic velocity and the mass flow rate.

In the paper, a plot of the macroscopic velocity as a function of the channel width for a fixed accommodation coefficient shows the qualitative results of the computation, providing the characteristic arched profile of the Poiseuille flow. The numerical results provided in the tables show the accuracy of X-TFC, achieving up to 8 digits of accuracy of the state-of-the-art benchmarks published by Barichello et al. [98, 99] and Ganapol [94]. The problem has been coded in the MATLAB R2019a platform, and all our simulations ran with an Intel Core i7–9700 CPU PC with 64 GB of RAM.
My contributions in this work are the formulation of the Poiseuille problem to reach a convenient form of the integro-differential equation to be solved by X-TFC, the X-TFC formulation to transform the constrained problem into an unconstrained linear system to be solved via least-square, coding the X-TFC algorithm to solve the Poiseuille problem and obtain numerical solutions of some physical quantities of interest in the literature, comparing our results with published and recognized benchmarks, and writing the journal paper.

4.2.2.3 Couette flow (De Florio et al., under review)

This section aims to introduce the paper [11] in Appendix E:

> De Florio et al. "Numerical analysis of the plane Couette flow of a rarefied gas with Physics-Informed Neural Networks and Functional Interpolation." *Physics of Fluids* (under review).

In this third paper of the trilogy on Rarefied-Gas Dynamics, a Couette flow in a micro-channel is studied. In a steady Couette flow, a rarefied gas is confined between two parallel plates in relative motion with opposite velocity directions. The flow of the gas is induced by shear stress imposed by the sliding of the walls. The Couette flow is widely used in real-world applications, such as lubrication theory with application in bearings [100], polymer solutions [101], and food processing industry [102, 103]. The pioneering work on the method

of elementary solutions in the gas kinetics theory by Cercignani provided all the essential equations for describing the Couette flow [104]. Over the years, many different techniques have attempted to produce solutions to this problem, for example, the relevant elementary solution Fredholm Integral equations by Williams [105], and Loyalka [106], and subsequently making use of the BGK approximation, solving the Couette problem for a wide range of Knudsen numbers [107, 108].

This paper aims to solve the Couette flow problem in the BGK approximation with the X-TFC method, to achieve the accuracy of the most challenging benchmarks in the literature. In the introduction, a general overview of the birth of the formulation of the problem and the different numerical techniques used in the last century to solve it are given, with a discussion on the benefits of using a PINN-based method for solving problems arising from the Boltzmann integro-differential equation. We start from the formulation of the problem provided by Williams [109] in the form of a partial integro-differential equation (see Eq. (14) in Appendix E) as a function of space and fluid velocity subject to two boundary conditions, one on the wall and one on the center of the channel, by taking advantage of the geometrical rotated symmetry of the flow. As for the Thermal creep and Poiseuille problems, to apply X-TFC, there is the need to approximate the integrals with Gauss-Legendre quadrature, mapping the independent variables, and creating a system of $2 \times N$ of ordinary differential equations with one constraint per each. By building the CEs and replacing them in our system, we compute the NN output weights β via a linear least-squares method. Solving the Couette flow problem means computing the velocity momentum of the flow, with which we can retrieve three quantities of interest to compare our results with published benchmarks: macroscopic velocity, flow mass rate, and pressure tensor, which is caused by the motion of the plates on the gas.

To show the potential of the X-TFC framework in solving the Couette flow, we firstly present the 3D plot of the velocity momentum as a function of channel width and flow velocity. One can see the discontinuity for velocity equal to zero, which could not have been possible to compute without splitting the original boundary value problem into two initial value problems. Also, two plots show the macroscopic velocity profiles as a function of channel width for fixed accommodation coefficient and as a function of accommodation coefficient for a fixed value of channel width. The accuracy of X-TFC is proved by the macroscopic velocity and flow mass rate numerical results matching eight digits of accuracy, but its real strength is shown in the results for the pressure tensor. This physical quantity has been computed for an extended interval of inverse Knudsen numbers, ranging from 10^{-7} up to 10^2, and achieving all the digits of accuracy (8) according to the state-of-the-art benchmark produced by Barichello et al. [99]. The problem has been coded in the MATLAB R2019a platform, and all our simulations ran with an Intel Core i7–9700 CPU PC with 64 GB of RAM.
My contributions in this work are the conceiving of the idea of solving the Couette flow to test the ability of X-TFC to solve rarefied-gas dynamics problems, the formulation of the

problem to reach a convenient form of the integro-differential equation to be solved by X-TFC, the X-TFC formulation to transform the constrained problem into an unconstrained linear system to be solved via least-square, coding the X-TFC algorithm to solve the Couette flow problem and obtain numerical solutions of some physical quantities of interest in the literature, comparing our results with published and recognized benchmarks, and writing the journal paper.

Chapter 5

X-TFC for Stiff Differential Equations

5.1 Stiff Chemical Kinetics (De Florio et al., 2022)

This section aims to introduce the paper [12] in Appendix F:

> De Florio et al. "Physics-informed neural networks and functional interpolation for stiff chemical kinetics." *Chaos: An Interdisciplinary Journal of Nonlinear Science* 32.6 (2022): 063107.

This methodology paper shows how the domain decomposition technique can be employed in the X-TFC framework to solve stiff systems on non-linear differential equations with characteristic timescales spanning a wide range of magnitude. The motivation that prompted the choice of this research is the failure of the regular PINN methods in solving stiff problems. Recently, there have been a few attempts to use NNs to solve these kinds of problems, but with barely acceptable results, both in terms of accuracy and computational time [110, 111, 112].

In this paper, after an introduction to the definition of "stiffness" and an overview of traditional and NNs numerical methods developed over the years, a general introduction to the classic X-TFC framework to solve initial value problems is given. For the first time, the derivation of an estimate of the generalization error of X-TFC in solving a non-linear IVP is proved, showing that it is bounded in terms of training error and quadrature error. Subsequently, the domain decomposition technique merged with the X-TFC framework is presented, providing instructions on building the CE for each sub-domain. To test the method, we solved stiff systems on non-linear differential equations for chemical kinetics. The choice of this typology of problems comes from the great challenge in solving these highly non-linear dynamic systems and for their multiple applications in the real world, such as energy conversions, including combustion engines [113], and battery and fuel cells [114]. Also, understanding the chemical reactions is essential for climate modeling [115] and kinetics of methane combustion [116]. The problems we solved are among the most challenging stiff chemical kinetics systems in the literature, widely used to test new numerical methods. The first problem is given by the ROBER problem [117], which is modeled by a system of 3

ODEs, with reaction rate constants values varying in a range of 9 orders and a time interval of 10^5 seconds. The second problem is the chemical Akzo Nobel problem [118], and it describes a chemical process in which the two species MBT and CHA are initially mixed while oxygen is continuously added. It consists of a set of 6 non-linear differential equations in a time domain of 180 seconds. The third problem we considered is the Belousov–Zhabotinsky reaction mechanism [119], which is represented by five homogeneous chemical reactions modeled by a system of 7 differential equations. This problem is usually reported in the literature for a time interval of 120 seconds, which we extended up to 5000 seconds to show the real strength and robustness of X-TFC for stiff problems. Finally, the last problem is a realistic air pollution model [120, 121], also known as the POLLU problem, developed at The Dutch National Institute of Public Health and Environmental Protection and modeled by a system of 20 non-linear ODEs [122], in a time domain of 60 seconds.

X-TFC with domain decomposition outperformed the state-of-the-art methods and regular PINNs frameworks for all the problems reported in both terms of accuracy and computational cost, finding challenging solutions where the other methods failed. All the codes have been implemented in MATLAB, and the simulation ran with an Intel Core i7 - 9700 CPU PC with 64 GB of RAM.
My contributions in this work are the conceptualization of the time-domain decomposition technique to lighten the computational efforts of the X-TFC algorithm in solving stiff problems, and finding the most challenging stiff problems in the literature that deserve further investigation. Then, for each problem, I coded the X-TFC algorithm for different time domains obtaining quantitative and qualitative results, which have been compared against results published in the literature, and against two MATLAB functions (ode15s and ode15i) which I have implemented and run for the same problems. Finally, I took care of the drafting of the journal manuscript.

Chapter 6

Conclusions and Future Works

In this dissertation, a new Physics-Informed Neural Networks framework named Extreme Theory of Functional Connections is developed to learn the solution of initial value problems, focusing on integro-differential equations and stiff systems of non-linear differential equations.

Two different possible approaches are defined to solve integro-differential equations, depending on the complexity of the integral kernel at hand. The Theory of Functional Connections has been adapted to build the integrated constrained expression by analytically integrating the free chosen function (i.e., orthogonal polynomials or NNs activation function). In realistic problems whose dynamic behavior is modeled by integro-differential equations, the integral kernel can be non-trivial, for example, an exponential function or an expansion of Legendre polynomials. Such integral kernel can be found in the challenging Transport Theory problems, which describe particles and radiation transport processes in a specific system. We proposed the X-TFC methodology with a Gauss-Legendre quadrature for the integral approximation to solve the Boltzmann integro-differential equation for real-world problems in radiative transfer (transport of photons in a medium) such as the Mie scattering problem and the Haze-L problem, and for rarefied-gas dynamics, solving the classical Thermal Creep Flow, Poisueille flow, and Couette flow. The accuracy, efficiency, and robustness of the X-TFC method have been proved by comparing our results with the most accurate benchmarks in the literature to date.

The second part of the dissertation focuses on the development of a new framework to solve the arduous stiff systems of non-linear differential equations, which lies on domain decomposition technique merged with X-TFC methodology, to capture the steep changes of highly non-linear dynamical behaviors in relatively wide time domains. To test and validate this new approach, X-TFC has been employed to solve real-world stiff problems in chemical kinetics. The choice of this class of problems comes from their numerical challenge and their wide applications in real problems, such as energy conversions, batteries, fuel cells, and climate modeling. X-TFC with domain decomposition showed its accuracy and efficiency by outperforming the traditional numerical methods designed to solve stiff differential equations

and the regular PINNs frameworks, learning the solutions of non-linear dynamical systems for extraordinary wide time integration ranges where other methods fail.

The research presented in this dissertation is just the beginning of a path that leads to improving and optimizing the X-TFC methodology to solve physical problems in both physics and data-driven fashion, and its adaptation to solve more challenging and complex problems. An important research direction that needs to be pursued is to extend the derivation of bounds on the generalization error of X-TFC to boundary value problems and PDEs, to provide a rigorous justification for its successful use. This will help to highlight the improvements brought by X-TFC compared with the regular PINNs in terms of training error and error due to a quadrature rule. The forecast would be to derive smaller upper bounds on the generalization error thanks to the constrained expressions, which analytically satisfy the problem constraints. Future directions will focus on using the domain decomposition technique to solve stiff boundary value problems and developing a data-physics-driven approach for inferring parameters and hidden-physics of stiff systems of non-linear differential equations. This will help scientists in many fields of science, such as chemistry, biology, and medicine, to retrieve the dynamics of physical quantities that it is impossible to observe and acquire data in laboratories.

Bibliography

[1] C. J. Edwards, *The historical development of the calculus*. Springer Science & Business Media, 2012.

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

[3] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 43, no. 5, pp. A3055–A3081, 2021.

[4] D. Mortari, "The theory of connections: Connecting points," *Mathematics*, vol. 5, no. 4, p. 57, 2017.

[5] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, vol. 2, pp. 985–990, Ieee, 2004.

[6] E. Schiassi, R. Furfaro, C. Leake, M. De Florio, H. Johnston, and D. Mortari, "Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations," *Neurocomputing*, vol. 457, pp. 334–356, 2021.

[7] M. De Florio, E. Schiassi, A. D'Ambrosio, D. Mortari, and R. Furfaro, "Theory of functional connections applied to linear odes subject to integral constraints and linear ordinary integro-differential equations," *Mathematical and Computational Applications*, vol. 26, no. 3, p. 65, 2021.

[8] M. De Florio, E. Schiassi, R. Furfaro, B. D. Ganapol, and D. Mostacci, "Solutions of chandrasekhar's basic problem in radiative transfer via theory of functional connections," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 259, p. 107384, 2021.

[9] M. De Florio, E. Schiassi, B. D. Ganapol, and R. Furfaro, "Physics-informed neural networks for rarefied-gas dynamics: Thermal creep flow in the bhatnagar–gross–krook approximation," *Physics of Fluids*, vol. 33, no. 4, p. 047110, 2021.

[10] M. De Florio, E. Schiassi, B. D. Ganapol, and R. Furfaro, "Physics-informed neural networks for rarefied-gas dynamics: Poiseuille flow in the bgk approximation," *Zeitschrift für angewandte Mathematik und Physik*, vol. 73, no. 3, pp. 1–18, 2022.

[11] M. De Florio, E. Schiassi, L. B. Barichello, and R. Furfaro, "Numerical analysis of the plane couette flow of a rarefied gas with physics-informed neural networks and functional interpolation," *Physics of Fluids*, under-review.

[12] M. De Florio, E. Schiassi, and R. Furfaro, "Physics-informed neural networks and functional interpolation for stiff chemical kinetics," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, no. 6, p. 063107, 2022.

[13] D. M. Burton, "The history of mathematics: An introduction," *Group*, vol. 3, no. 3, p. 35, 1985.

[14] O. Neugebauer, *The exact sciences in antiquity*, vol. 9. Courier Corporation, 1969.

[15] J. Dutka, "Eratosthenes' measurement of the earth reconsidered," *Archive for History of Exact Sciences*, pp. 55–66, 1993.

[16] H. C. King, *The history of the telescope*. Courier Corporation, 2003.

[17] M. A. Finocchiaro, *Galileo and the art of reasoning: Rhetorical foundation of logic and scientific method*, vol. 61. Springer Science & Business Media, 2012.

[18] D. Brewster, *The martyrs of science, or, The lives of Galileo, Tycho Brahe, and Kepler*. Chatto and Windus, 1874.

[19] J. L. Russell, "Kepler's laws of planetary motion: 1609–1666," *The British journal for the history of science*, vol. 2, no. 1, pp. 1–24, 1964.

[20] B. S. Baigrie, "Kepler's laws of planetary motion, before and after newton's principia: An essay on the transformation of scientific problems," *Studies in History and Philosophy of Science Part A*, vol. 18, no. 2, pp. 177–208, 1987.

[21] J. T. Katsikadelis, "Derivation of newton's law of motion from kepler's laws of planetary motion," *Archive of Applied Mechanics*, vol. 88, no. 1, pp. 27–38, 2018.

[22] I. Newton, *Philosophiae naturalis principia mathematica*, vol. 1. G. Brookman, 1833.

[23] I. Newton, "De methodis serierum et fluxionum," *Mathematical Papers*, vol. 3, pp. 32–353, 1967.

[24] M. Braun and M. Golubitsky, *Differential equations and their applications*, vol. 1. Springer, 1983.

[25] L. Euler, *Institutionum calculi integralis*, vol. 1. impensis Academiae imperialis scientiarum, 1824.

[26] J. Crank and P. Nicolson, "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 43, pp. 50–67, Cambridge University Press, 1947.

[27] J. C. Butcher and G. Wanner, "Runge-kutta methods: some historical notes," *Applied Numerical Mathematics*, vol. 22, no. 1-3, pp. 113–151, 1996.

[28] J. C. Butcher, *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[29] R. Iten, T. Metger, H. Wilming, L. Del Rio, and R. Renner, "Discovering physical concepts with neural networks," *Physical review letters*, vol. 124, no. 1, p. 010508, 2020.

[30] J. P. Crutchfield and B. McNamara, "Equations of motion from a data series," *Complex systems*, vol. 1, pp. 417–452, 1987.

[31] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[32] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *science*, vol. 324, no. 5923, pp. 81–85, 2009.

[33] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[34] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," *arXiv preprint arXiv:2010.08895*, 2020.

[35] A. Kashefi, D. Rempe, and L. J. Guibas, "A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries," *Physics of Fluids*, vol. 33, no. 2, p. 027104, 2021.

[36] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[37] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[38] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE transactions on neural networks*, vol. 9, no. 5, pp. 987–1000, 1998.

[39] H. Sheng and C. Yang, "Pfnn: A penalty-free neural network method for solving a class of second-order boundary-value problems on complex geometries," *Journal of Computational Physics*, vol. 428, p. 110085, 2021.

[40] K. S. McFall and J. R. Mahan, "Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions," *IEEE Transactions on Neural Networks*, vol. 20, no. 8, pp. 1221–1233, 2009.

[41] R. S. Beidokhti and A. Malek, "Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques," *Journal of the Franklin Institute*, vol. 346, no. 9, pp. 898–913, 2009.

[42] J. Sirignano and K. Spiliopoulos, "Dgm: A deep learning algorithm for solving partial differential equations," *Journal of computational physics*, vol. 375, pp. 1339–1364, 2018.

[43] G. Pang, L. Lu, and G. E. Karniadakis, "fpinns: Fractional physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 41, no. 4, pp. A2603–A2626, 2019.

[44] L. Yang, D. Zhang, and G. E. Karniadakis, "Physics-informed generative adversarial networks for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 42, no. 1, pp. A292–A317, 2020.

[45] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 365, p. 113028, 2020.

[46] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, "hp-vpinns: Variational physics-informed neural networks with domain decomposition," *Computer Methods in Applied Mechanics and Engineering*, vol. 374, p. 113547, 2021.

[47] A. D. Jagtap and G. E. Karniadakis, "Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," *Communications in Computational Physics*, vol. 28, no. 5, pp. 2002–2041, 2020.

[48] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations," *Science*, vol. 367, no. 6481, pp. 1026–1030, 2020.

[49] R. K. Tripathy and I. Bilionis, "Deep uq: Learning deep neural network surrogate models for high dimensional uncertainty quantification," *Journal of computational physics*, vol. 375, pp. 565–588, 2018.

[50] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Numerical gaussian processes for time-dependent and nonlinear partial differential equations," *SIAM Journal on Scientific Computing*, vol. 40, no. 1, pp. A172–A198, 2018.

[51] L. Yang, X. Meng, and G. E. Karniadakis, "B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data," *Journal of Computational Physics*, vol. 425, p. 109913, 2021.

[52] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis, "Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems," *Journal of Computational Physics*, vol. 397, p. 108850, 2019.

[53] S. Cai, Z. Wang, F. Fuest, Y. J. Jeon, C. Gray, and G. E. Karniadakis, "Flow over an espresso cup: inferring 3-d velocity and pressure fields from tomographic background oriented schlieren via physics-informed neural networks," *Journal of Fluid Mechanics*, vol. 915, 2021.

[54] D. C. Psichogios and L. H. Ungar, "A hybrid neural network-first principles approach to process modeling," *AIChE Journal*, vol. 38, no. 10, pp. 1499–1511, 1992.

[55] N. Schanche, A. C. Cameron, G. Hébrard, L. Nielsen, A. Triaud, J. Almenara, K. Alsubai, D. Anderson, D. Armstrong, S. Barros, *et al.*, "Machine-learning approaches to exoplanet transit detection and candidate validation in wide-field ground-based surveys," *Monthly Notices of the Royal Astronomical Society*, vol. 483, no. 4, pp. 5534–5547, 2019.

[56] N. M. Ball, R. J. Brunner, A. D. Myers, and D. Tcheng, "Robust machine learning applied to astronomical data sets. i. star-galaxy classification of the sloan digital sky survey dr3 using decision trees," *The Astrophysical Journal*, vol. 650, no. 1, p. 497, 2006.

[57] J. Stilgoe, "Machine learning, social learning and the governance of self-driving cars," *Social studies of science*, vol. 48, no. 1, pp. 25–56, 2018.

[58] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[59] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[60] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

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

[62] P. Boyle and M. Frean, "Dependent gaussian processes," *Advances in neural information processing systems*, vol. 17, 2004.

[63] S. Desai and A. Strachan, "Parsimonious neural networks learn interpretable physical laws," *Scientific reports*, vol. 11, no. 1, pp. 1–9, 2021.

[64] S. Weinberg, *Dreams of a final theory: The scientist's search for the ultimate laws of nature*. Vintage, 2011.

[65] S. Fadugba, B. Ogunrinde, and T. Okunlola, "Euler's method for solving initial value problems in ordinary differential equations.," *Euler's Method for Solving Initial Value Problems in Ordinary Differential Equations.*, vol. 13, no. 2, pp. 1–7, 2012.

[66] J. C. Butcher, "A history of runge-kutta methods," *Applied numerical mathematics*, vol. 20, no. 3, pp. 247–260, 1996.

[67] W. B. Gragg and H. J. Stetter, "Generalized multistep predictor-corrector methods," *Journal of the ACM (JACM)*, vol. 11, no. 2, pp. 188–209, 1964.

[68] S. Godunov and I. Bohachevsky, "Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics," *Matematičeskij sbornik*, vol. 47, no. 3, pp. 271–306, 1959.

[69] J. N. Reddy, *Introduction to the finite element method*. McGraw-Hill Education, 2019.

[70] R. Eymard, T. Gallouët, and R. Herbin, "Finite volume methods," *Handbook of numerical analysis*, vol. 7, pp. 713–1018, 2000.

[71] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," *OpenReview*, 2017.

[72] R. Gorenflo and S. Vessella, *Abel integral equations*, vol. 1461. Springer, 1991.

[73] A. J. Lotka, "On an integral equation in population analysis," *The Annals of Mathematical Statistics*, vol. 10, no. 2, pp. 144–161, 1939.

[74] I. Fredholm, "Sur une classe d'équations fonctionnelles," *Acta mathematica*, vol. 27, pp. 365–390, 1903.

[75] Z. B. Tsalyuk, "Volterra integral equations," *Itogi Nauki i Tekhniki. Seriya" Matematicheskii Analiz"*, vol. 15, pp. 131–198, 1977.

[76] R. Bellman and R. S. Roth, *The laplace transform*, vol. 3. World Scientific, 1984.

[77] A. Avudainayagam and C. Vani, "Wavelet–galerkin method for integro–differential equations," *Applied Numerical Mathematics*, vol. 32, no. 3, pp. 247–254, 2000.

[78] P. Van der Houwen and B. Sommeijer, "Euler-chebyshev methods for integro-differential equations," *Applied Numerical Mathematics*, vol. 24, no. 2-3, pp. 203–218, 1997.

[79] P. Darania and A. Ebadian, "A method for the numerical solution of the integro-differential equations," *Applied Mathematics and Computation*, vol. 188, no. 1, pp. 657–668, 2007.

[80] G. C. Wick, "über ebene diffusionsprobleme," *Zeitschrift für Physik*, vol. 121, no. 11, pp. 702–718, 1943.

[81] S. Chandrasekhar, *Radiative transfer*. Courier Corporation, 2013.

[82] P. Benoist and A. Kavenoky, "A new method of approximation of the boltzmann equation," *Nuclear Science and Engineering*, vol. 32, no. 2, pp. 225–232, 1968.

[83] M. Vilhena, C. Segatto, and L. Barichello, "A particular solution for the sn radiative transfer problems," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 53, no. 4, pp. 467–469, 1995.

[84] B. D. Ganapol, "The response matrix discrete ordinates solution to the 1d radiative transfer equation," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 154, pp. 72–90, 2015.

[85] M. Knudsen, "Thermischer molekulardruck der gase in röhren," *Annalen der Physik*, vol. 338, no. 16, pp. 1435–1448, 1910.

[86] J. C. Maxwell, "Iii. on stresses in rarefied gases arising from inequalities of temperature," *Proceedings of the Royal Society of London*, vol. 27, no. 185-189, pp. 304–308, 1878.

[87] O. Reynolds, "Xviii. on certain dimensional properties of matter in the gaseous state.-part i. experimental researches on thermal transpiration of gases through porous plates and on the laws of transpiration and impulsion, including an experimental proof that gas is not a continuous plenum.-part ii. on an extension of the dynamical theory of gas, which includes the stresses, tangential and normal, caused by a varying condition of gas, and affords an explanation of the phenomena of transpiration and impulsion," *Philosophical Transactions of the Royal Society of London*, no. 170, pp. 727–845, 1879.

[88] C. Cercignani, *Theory and application of the Boltzmann equation*. Elsevier Science & Technology, 1975.

[89] G. A. Bird, "Molecular gas dynamics and the direct simulation of gas flows," *Molecular gas dynamics and the direct simulation of gas flows*, 1994.

[90] P. L. Bhatnagar, E. P. Gross, and M. Krook, "A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems," *Physical review*, vol. 94, no. 3, p. 511, 1954.

[91] E. Shakhov, "Generalization of the krook kinetic relaxation equation," *Fluid dynamics*, vol. 3, no. 5, pp. 95–96, 1968.

[92] C. Siewert and J. Thomas Jr, "A particular solution for the pn method in radiative transfer," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 43, no. 6, pp. 433–436, 1990.

[93] C. Siewert and P. Benoist, "The fn method in neutron-transport theory. part i: Theory and applications," *Nuclear Science and Engineering*, vol. 69, no. 2, pp. 156–160, 1979.

[94] B. D. Ganapol, "Poiseuille channel flow by adding and doubling," in *AIP Conference Proceedings*, vol. 1786, p. 070009, AIP Publishing LLC, 2016.

[95] A. Namatame, "Connectionist learning with chebychev networks and analyses of its internal representation," in *Applications of Learning and Planning Methods*, pp. 35–48, World Scientific, 1991.

[96] S. Mall and S. Chakraverty, "Single layer chebyshev neural network model for solving elliptic partial differential equations," *Neural Processing Letters*, vol. 45, no. 3, pp. 825–840, 2017.

[97] C. Siewert, R. Garcia, and P. Grandjean, "A concise and accurate solution for poiseuille flow in a plane channel," *Journal of Mathematical Physics*, vol. 21, no. 12, pp. 2760–2763, 1980.

[98] L. B. Barichello and C. Siewert, "A discrete-ordinates solution for poiseuille flow in a plane channel," *Zeitschrift für angewandte Mathematik und Physik ZAMP*, vol. 50, no. 6, pp. 972–981, 1999.

[99] L. Barichello, M. Camargo, P. Rodrigues, and C. Siewert, "Unified solutions to classical flow problems based on the bgk model," *Zeitschrift für angewandte Mathematik und Physik ZAMP*, vol. 52, no. 3, pp. 517–534, 2001.

[100] E. Barić and H. Steiner, "Extended lubrication theory for generalized couette flow through converging gaps," *International Journal of Heat and Mass Transfer*, vol. 99, pp. 149–158, 2016.

[101] A. Groisman and V. Steinberg, "Mechanism of elastic instability in couette flow of polymer solutions: experiment," *Physics of Fluids*, vol. 10, no. 10, pp. 2451–2463, 1998.

[102] G. A. Krintiras, J. G. Diaz, A. J. Van Der Goot, A. I. Stankiewicz, and G. D. Stefanidis, "On the use of the couette cell technology for large scale production of textured soy-based meat replacers," *Journal of Food Engineering*, vol. 169, pp. 205–213, 2016.

[103] H. Masuda, *Enhancement of Heat Transfer Using Taylor Vortices in Thermal Processing for Food Process Intensification*. IntechOpen, 2021.

[104] C. Cercignani *et al.*, *Mathematical methods in kinetic theory*, vol. 1. Springer, 1969.

[105] M. Williams, "Rarefied gas flow between parallel plates," in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 66, pp. 189–196, Cambridge University Press, 1969.

[106] S. Loyalka, N. Petrellis, and T. Storvick, "Some exact numerical results for the bgk model: Couette, poiseuille and thermal creep flow between parallel plates," *Zeitschrift für angewandte Mathematik und Physik ZAMP*, vol. 30, no. 3, pp. 514–521, 1979.

[107] E. Gross, E. Jackson, and S. Ziering, "Boundary value problems in kinetic theory of gases," *Annals of Physics*, vol. 1, no. 2, pp. 141–167, 1957.

[108] D. R. Willis, "Comparison of kinetic theory analyses of linearized couette flow," *The Physics of Fluids*, vol. 5, no. 2, pp. 127–135, 1962.

[109] M. Williams, "A review of the rarefied gas dynamics theory associated with some classical problems in flow and heat transfer," *Zeitschrift für angewandte Mathematik und Physik ZAMP*, vol. 52, no. 3, pp. 500–516, 2001.

[110] W. Ji, W. Qiu, Z. Shi, S. Pan, and S. Deng, "Stiff-pinn: Physics-informed neural network for stiff chemical kinetics," *The Journal of Physical Chemistry A*, vol. 125, no. 36, pp. 8098–8106, 2021.

[111] S. Kim, W. Ji, S. Deng, Y. Ma, and C. Rackauckas, "Stiff neural ordinary differential equations," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 9, p. 093122, 2021.

[112] T. S. Brown, H. Antil, R. Löhner, F. Togashi, and D. Verma, "Novel dnns for stiff odes with applications to chemically reacting flows," in *International Conference on High Performance Computing*, pp. 23–39, Springer, 2021.

[113] F. Battin-Leclerc, E. Blurock, R. Bounaceur, R. Fournet, P.-A. Glaude, O. Herbinet, B. Sirjean, and V. Warth, "Towards cleaner combustion engines through groundbreaking detailed chemical kinetic models," *Chemical Society Reviews*, vol. 40, no. 9, pp. 4762–4782, 2011.

[114] M. Winter and R. J. Brodd, "What are batteries, fuel cells, and supercapacitors?," *Chemical reviews*, vol. 104, no. 10, pp. 4245–4270, 2004.

[115] M. Alvanos and T. Christoudias, "Accelerating atmospheric chemical kinetics for climate simulations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 11, pp. 2396–2407, 2019.

[116] M. Frenklach, H. Wang, and M. J. Rabinowitz, "Optimization and analysis of large chemical kinetic mechanisms using the solution mapping method—combustion of methane," *Progress in Energy and Combustion Science*, vol. 18, no. 1, pp. 47–73, 1992.

[117] H. Robertson, "The solution of a set of reaction rate equations," *Numerical analysis: an introduction*, vol. 178182, 1966.

[118] K. Eriksson, C. Johnson, and A. Logg, "Explicit time-stepping for stiff odes," *SIAM Journal on Scientific Computing*, vol. 25, no. 4, pp. 1142–1157, 2004.

[119] A. M. Zhabotinsky, "Periodical oxidation of malonic acid in solution (a study of the belousov reaction kinetics)," *Biofizika*, vol. 9, pp. 306–311, 1964.

[120] A. C. Hindmarsh, "Lsode and lsodi, two new initial value ordinary differential equation solvers," *ACM Signum Newsletter*, vol. 15, no. 4, pp. 10–11, 1980.

[121] F. De Leeuw, "Numerical solution of ordinary differential equations arising from chemical kinetics," *RIVM Rapport 228603005*, 1988.

[122] J. G. Verwer, "Gauss–seidel iteration for stiff odes from chemical kinetics," *SIAM Journal on Scientific Computing*, vol. 15, no. 5, pp. 1243–1250, 1994.

Appendix A

De Florio et al. (2021) TFC for Integro-Differential Equations

Reproduced from De Florio, M., Schiassi, E., D'Ambrosio, A., Mortari, D., & Furfaro, R. (2021). Theory of functional connections applied to linear odes subject to integral constraints and linear ordinary integro-differential equations. *Mathematical and Computational Applications*, 26(3), 65. https://doi.org/10.3390/mca26030065 [7], with the permission of MDPI.

Mathematical and Computational Applications

MDPI

Article

Theory of Functional Connections Applied to Linear ODEs Subject to Integral Constraints and Linear Ordinary Integro-Differential Equations

Mario De Florio [1], Enrico Schiassi [1], Andrea D'Ambrosio [2], Daniele Mortari [3,*] and Roberto Furfaro [1]

[1] Department of Systems and Industrial Engineering, The University of Arizona, 1127 E. James E. Rogers Way, Tucson, AZ 85721, USA; mariodf@email.arizona.edu (M.D.F.); eschiassi@email.arizona.edu (E.S.); robertof@email.arizona.edu (R.F.)
[2] School of Aerospace Engineering, Università degli Studi di Roma "La Sapienza", Via Salaria 851, 00138 Rome, Italy; andrea.dambrosio@uniroma1.it
[3] Department of Aerospace Engineering, College of Engineering, Texas A&M University, 401 Joe Routt Blvd., College Station, TX 77843, USA
* Correspondence: mortari@tamu.edu

Abstract: This study shows how the Theory of Functional Connections (TFC) allows us to obtain fast and highly accurate solutions to linear ODEs involving integrals. Integrals can be constraints and/or terms of the differential equations (e.g., ordinary integro-differential equations). This study first summarizes TFC, a mathematical procedure to obtain *constrained expressions*. These are functionals representing *all functions satisfying a set of linear constraints*. These functionals contain a free function, $g(x)$, representing the unknown function to optimize. Two numerical approaches are shown to numerically estimate $g(x)$. The first models $g(x)$ as a linear combination of a set of basis functions, such as Chebyshev or Legendre orthogonal polynomials, while the second models $g(x)$ as a neural network. Meaningful problems are provided. In all numerical problems, the proposed method produces very fast and accurate solutions.

Keywords: Theory of Functional Connections; Ordinary Differential Equations; integro-differential equations; Extreme Learning Machine; numerical methods

Citation: De Florio, M.; Schiassi, E.; D'Ambrosio, A.; Mortari, D.; Furfaro, R. Theory of Functional Connections Applied to Linear ODEs Subject to Integral Constraints and Linear Ordinary Integro-Differential Equations. *Math. Comput. Appl.* **2021**, *26*, 65. https://doi.org/10.3390/mca26030065

Academic Editor: Miguel Ángel Moreles

Received: 29 August 2021
Accepted: 10 September 2021
Published: 12 September 2021

1. Introduction

This paper shows how to solve linear Ordinary Differential Equations (ODEs) and linear Integro-Differential Equations (IDEs) using a new mathematical framework to perform functional interpolation, called *Theory of Functional Connections* (TFC). TFC derives functionals, called *constrained expressions*, containing a free function and representing all possible functions satisfying a set of linear constraints [1–4]. The most important feature of the constrained expressions is: *they always satisfy all the constraints no matter what the free function is*.

Although it was recently developed (2017), TFC has already found several applications, especially in solving differential equations [5–8]. The free function can be expressed by any set of linearly independent functions, such as an expansion of orthogonal polynomials (e.g., Chebyshev, Legendre, etc.) or Neural Networks (NN), such as shallow NN with random features, or Deep NNs (DNNs). When the free function is expanded by a set of orthogonal polynomials the method is here identified as "classic TFC". When shallow NNs with random features are used, the method has been identified as Extreme-TFC (X-TFC) [9], and when DNNs are used the method is called Deep-TFC [10].

The best expansion of the free function depends on the differential equation considered. Usually, for linear/nonlinear ODEs and simple bivariate PDEs, orthogonal polynomials represent the best choice in terms of accuracy. Indeed, orthogonal polynomials are generally the best mathematical tools for approximating and for convergence properties [11,12].

Nevertheless, for multivariate and more complex PDEs, orthogonal polynomials suffer the "curse of dimensionality". For these problems, NNs represent a better choice as they better tolerate the curse of dimensionality.

In all previous applications, the free function has been expressed as a linear combination of known basis functions and unknown constant coefficients. These coefficients are then estimated by least-squares for linear Differential Equations (DEs) [5] and by nonlinear least-squares for nonlinear DEs [6]. The problems considered in this work will be solved with classic TFC using Chebyshev orthogonal polynomials and X-TFC.

Note that X-TFC and Deep-TFC are also identified as Physics-Informed Neural Networks (PINN) frameworks [9]. PINNs are recently developed machine-learning methods that employ NNs for data-physics-driven regression, data-physics-driven solution of DEs, and data-physics-driven discovery of parameters governing DEs [13]. Since X-TFC and Deep-TFC use NNs as free function, they are considered part of the PINNs family. Thanks to the constrained expression, developed by TFC, both X-TFC and Deep-TFC are more robust and accurate than the standard PINN frameworks as introduced by Raissi et al. [13].

TFC has been developed for univariate and multivariate scenarios [2,7,8] to solve a variety of mathematical problems: a homotopy continuation algorithm for dynamics and control problems [14], domain mapping [15], data-driven parameters discovery applied to epidemiological compartmental models [16], transport theory problems such as radiative transfer [17] and rarefied-gas dynamics [18], nonlinear programming under equality constraints [19], Timoshenko–Ehrenfest beam [20], boundary-value problems in hybrid systems [21], eighth-order boundary value problems [22], and in Support Vector Machine [23]. TFC has been widely used for solving optimal control problems for space application, solved via indirect methods [24]: orbit transfer and propagation [25–28], energy-optimal in relative motion [29], energy-optimal and fuel-efficient landing on small and large planetary bodies [30,31], the minimum time-energy optimal intercept problem [32].

The aim of this paper is to show how TFC can accommodate integral constraints in linear differential equations and solve linear integro-differential equations, is organized as follows. In Section 2, a summary of the TFC framework is provided, with the explanation of how to derive constrained expressions. In Section 3, the application of TFC to solve ODEs with integral constraints is shown, and some case problems are reported as examples. In Section 4, the TFC framework is used to solve linear integro-differential equations.

In all test problems considered in this article, the results have been obtained using MATLAB R2020a software on an Intel Core i7-9700 CPU PC with 64 GB of RAM. The results' accuracy is provided in terms of the absolute error. That is,

$$err = |y_{\text{TFC}} - y_{\text{true}}|,$$

where y_{TFC} is the TFC approximated solution, and y_{true} is the true analytical solution.

Since the classic TFC uses Chebyshev or Legendre orthogonal polynomials as a basis set, the final Appendix A provides a definition, orthogonality, derivatives, and integral expressions and properties for both Chebyshev and Legendre orthogonal polynomials.

2. Theory of Functional Connections Summary

A mathematical generalization of interpolation, called *Theory of Functional Connections*, has recently been developed and successfully applied to solve, by least-squares, initial, boundary, and multi-value problems of linear [5] and nonlinear [6] ODEs and PDEs [8,10]. The theory has been developed for univariate [1] and multivariate rectangular domains [2,8].

The generalization of interpolation consists of a mathematical procedure to obtain analytical expressions describing *all* possible functions subject to n linear constraints. These expressions are functionals that are called *constrained expressions*. Two formally equivalent approaches [1,8] can be used to derive them. These are,

$$y(x, g(x)) = g(x) + \sum_{k=1}^{n} \eta_k(x, g(x))\, s_k(x), \tag{1}$$

$$y(x, g(x)) = g(x) + \sum_{k=1}^{n} \phi_k(x, \boldsymbol{s}(x))\, \rho_k(x, g(x)), \tag{2}$$

where n is the number of the linear constraints, $g(x)$ is a function that can be freely chosen, $\eta_k(x, g(x))$ are *functional coefficients*, $\boldsymbol{s}(x) = \{s_1(x), \cdots, s_n(x)\}$ is a set of n user-defined *support functions* that must be linearly independent (necessary conditions), ϕ_k are *switching functions*, and $\rho_k(x, g(x))$ are *projection functionals*.

By imposing the n constraints to Equation (1), the values of the n functional coefficients, $\eta_k(x, g(x))$, are obtained for some set of n user-assigned functions, $s_k(x)$. Once the n functional coefficients, $\eta_k(x, g(x))$, are computed, by substituting them back into Equation (1) we obtain the constrained expression. To give an example, using $s_1(x) = x$ and $s_2(x) = x^2$, the constrained expression,

$$f(x, g(x)) = g(x) + \underbrace{\frac{x\,(2x_2 - x)}{2(x_2 - x_1)}}_{\phi_1(x)} \underbrace{(f_{x1} - g_x(x_1))}_{\rho_1(x, g(x))} + \underbrace{\frac{x\,(x - 2x_1)}{2(x_2 - x_1)}}_{\phi_2(x)} \underbrace{(f_{x2} - g_x(x_2))}_{\rho_2(x, g(x))},$$

where the switching functions, $\phi_k(x)$, and projection functionals, $\rho_k(x, g(x))$, are identified, always satisfies the two derivative constraints, $f_x(x_1) = f_{x1}$ and $f_x(x_2) = f_{x2}$, no matter what $g(x)$ is. In this article, the use of both formulations, provided by Equations (1) and (2), will be shown.

The projection functional, $\rho_k(x, g(x))$, projects the free function, $g(x)$, on the k-th constraint. Here is an example of projection functionals associated to a set of constraints,

$$\begin{cases} f(3) = 9 \\ 3f(0) - f(2) = 0 \\ \ddot{f}(1) = -1 \end{cases} \quad \rightarrow \quad \begin{cases} \rho_1 = 9 - g(3) \\ \rho_2 = -3g(0) + g(2) \\ \rho_3 = -1 - \ddot{g}(1) \end{cases}.$$

As for the switching functions, $\phi_k(x)$, they are expressed as a linear combination of a set of support functions, $\boldsymbol{s}(x)$. These functions satisfy $\phi_k(x) = 1$ when the k-th constraint is verified and $\phi_k(x) = 0$ when any other constraint is verified. For instance, given the support functions,

$$s_1(x) = x, \qquad s_2(x) = e^x, \qquad \text{and} \qquad s_3(x) = \sin x,$$

the switching functions are computed as $\phi_j(x) = \sum_{i=1}^{3} \alpha_{ji}\, s_i(x)$, where the α_{ji} coefficients are computed by inverting the support matrix. For this example,

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} \begin{bmatrix} s_1(3) & 3s_1(0) - s_1(2) & \ddot{s}_1(1) \\ s_2(3) & 3s_2(0) - s_2(2) & \ddot{s}_2(1) \\ s_3(3) & 3s_3(0) - s_3(2) & \ddot{s}_3(1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Reference [4] presents, in detail, how to derive the TFC constrained expressions using the formalism defined by Equation (2).

Constrained expressions have been used to solve differential equations. This is done by expressing the solution using the TFC constrained expressions from Equation (1) or, equivalently, from Equation (2). These functionals allow us to reduce the whole functions space to only the functions subspace satisfying the constraints. This is particularly important when solving differential equations. In fact, when substituting these functionals into the DE, a new differential equation is obtained in terms of $g(x)$. This new DE is subject to *no constraints* because the constrained expression fully satisfies the constraints. The unknown

free function, $g(x)$, is then expressed as a linear combination of basis functions. In the classic TFC, $g(x)$ is a linear combination of orthogonal polynomials. That is,

$$g(x, \boldsymbol{\xi}) = \sum_{j=0}^{m} \xi_j \, h_j(x) = \boldsymbol{\xi}^{\mathrm{T}} \boldsymbol{h}(x),$$

where m is the number of the basis functions $h_j(x)$ (orthogonal polynomials). In X-TFC, $g(x, \boldsymbol{\xi})$ is a shallow NN trained with Extreme Learning Machine (ELM) [33],

$$g(x, \boldsymbol{\xi}) = \sum_{j=1}^{L} \xi_j \, \sigma_j(w_j \, x + b_j) = \boldsymbol{\xi}^{\mathrm{T}} \boldsymbol{\sigma}(x) = \boldsymbol{\xi}^{\mathrm{T}} \begin{Bmatrix} \sigma_1(x) \\ \vdots \\ \sigma_L(x), \end{Bmatrix}$$

where L is the number of hidden neurons, $w_j \in \mathbb{R}$ is the input weights vector connecting the j-th hidden neuron and the input nodes, $\xi_j \in \mathbb{R}$ with $j = 1, \cdots, L$ is the j-th output weight connecting the j-th hidden neuron and the output node, and b_j is the bias of the j-th hidden neuron, $\sigma_j(\cdot)$ are activation functions, and $\boldsymbol{\sigma} = \{\sigma_1, \cdots, \sigma_L\}^{\mathrm{T}}$. According to the ELM algorithm [33], biases and input weights are randomly selected and not tuned during the training, thus they are known hyper-parameters. The activation functions, $\sigma_j(\cdot)$, are also known as they are user selected. Thus, the only unknown NN hyper-parameters to compute are the output weights $\boldsymbol{\xi} = \{\xi_1, \cdots, \xi_L\}^{\mathrm{T}}$. Therefore, for both classic TFC and X-TFC, the unknown vector, $\boldsymbol{\xi}$, which actually constitutes the *only* unknown of our problem, is then estimated numerically, as for instance by least-squares for linear [5] and nonlinear [6] differential equation s.

In general, the independent variable of the DE (for instance, time) is defined in the $t \in [t_0, t_f]$ range, while the selected basis functions may be defined as a different range, $x \in [x_0, x_f]$ (for instance, Chebyshev and Legendre orthogonal polynomials are defined in the $x \in [-1, +1]$). Thus, a change of variable is needed. The most simple mapping between these two variables is linear,

$$x = x_0 + \frac{x_f - x_0}{t_f - t_0}(t - t_0) \qquad \longleftrightarrow \qquad t = t_0 + \frac{t_f - t_0}{x_f - x_0}(x - x_0). \tag{3}$$

By setting the range ratio, $c = \dfrac{x_f - x_0}{t_f - t_0}$, the derivatives in terms of the new variable are

$$\frac{\mathrm{d}^k f}{\mathrm{d}t^k} = c^k \, \frac{\mathrm{d}^k f}{\mathrm{d}x^k}, \tag{4}$$

and the derivative constraints can be written as:

$$\left.\frac{\mathrm{d}^k f}{\mathrm{d}t^k}\right|_{t_i} = f_{t_i}^{(k)} = c^k \left.\frac{\mathrm{d}^k f}{\mathrm{d}x^k}\right|_{x_i} = c^k \, f_{x_i}^{(k)}. \tag{5}$$

The change of variable in integrals takes advantage from the fact that the mean value is independent from the independent variable. Therefore,

$$\frac{1}{t_f - t_0} \int_{t_0}^{t_f} f(t) \, \mathrm{d}t = \frac{1}{x_f - x_0} \int_{x_0}^{x_f} f(x) \, \mathrm{d}x \qquad \rightarrow \qquad c \int_{t_0}^{t_f} f(t) \, \mathrm{d}t = \int_{x_0}^{x_f} f(x) \, \mathrm{d}x.$$

When expressing the free function as $g(x, \boldsymbol{\xi}) = \boldsymbol{\xi}^{\mathrm{T}} \boldsymbol{h}(x)$, then, the derivatives of $g(x, \boldsymbol{\xi})$ can be written as:

$$\frac{\mathrm{d}^k g}{\mathrm{d}x^k} = \boldsymbol{\xi}^{\mathrm{T}} \frac{\mathrm{d}^k \boldsymbol{h}}{\mathrm{d}x^k} \qquad \text{and} \qquad \left.\frac{\mathrm{d}^k g}{\mathrm{d}x^k}\right|_{x_i} = \boldsymbol{\xi}^{\mathrm{T}} \left.\frac{\mathrm{d}^k \boldsymbol{h}}{\mathrm{d}x^k}\right|_{x_i}.$$

This procedure can be applied to linear differential equations with non-constant coefficients. The final expression obtained is linear in terms of the unknown vector, $\boldsymbol{\xi}$, and can be written as:

$$\boldsymbol{a}^{\mathrm{T}}(x)\,\boldsymbol{\xi} = b(x). \tag{6}$$

To solve the problem numerically, this new equation is discretized for a set of N distinct values of x. A different discretization scheme can be used. Usually, the discretization points are either randomly selected or linearly uniformly spaced. When using Chebyshev polynomials the best discretization is to use the zeros of the Chebyshev polynomials, also called Chebyshev points or nodes or, more formally, Chebyshev–Gauss points [12]. They are defined by the cosine distribution,

$$x_k = \cos\left(\frac{(2k-1)\pi}{2N}\right) \qquad \text{for} \qquad k = 1, \cdots, N.$$

By specifying Equation (6) for these x_k values a system of N linear equations is obtained in m (or L) unknowns that is then solved for $\boldsymbol{\xi}$ by least-squares. Several least-squares methods can be used to solve (6). The optimal least-squares method to use for each problem depends on the problem itself and on what TFC technique is used to solve the problem (e.g., in this paper, classic TFC or X-TFC). For instance, for the classic TFC and linear DEs, our analysis identifies the QR decomposition on a scaled coefficient matrix as the best approach minimizing the condition number of the matrix to invert.

3. TFC for ODEs with Integral Constraints

In this section, we show how TFC is applied to solve linear ODEs with integral constraints. After showing how to derive the constrained expression when dealing with a general integral constraint, we will solve a couple of linear ODEs subjects to boundary conditions and integral constraints.

For the first two examples in this section, we will derive the constrained expression by following the formulation of Equation (1). For subsequent examples and problems, however, the second formulation, Equation (2), will be adopted.

3.1. Definite Integral Constraint

Let us consider the integral constraint,

$$\int_a^b f(x)\,\mathrm{d}x = I. \tag{7}$$

The constrained expression has the form,

$$f(x, g(x)) = g(x) + \eta_1\, s_1(x), \tag{8}$$

where $g(x)$ is a free function and $s_1(x)$ is a user-defined support function, and η_1 is a coefficient that is derived by imposing the constraint. By integrating Equation (8) the following equation:

$$I = \int_a^b g(x)\,\mathrm{d}x + \eta_1 \int_a^b s_1(x)\,\mathrm{d}x$$

is obtained, which can be rearranged to obtain the expression for η_1,

$$\eta_1 = \frac{I - \int_a^b g(x)\,\mathrm{d}x}{\int_a^b s_1(x)\,\mathrm{d}x}.$$

Substituting this value into Equation (8), we obtain a constrained expression for $f(x)$, that is, an expression that always satisfies the integral constraint for any expression of $g(x)$,

$$f(x, g(x)) = g(x) + \frac{s_1(x)}{\int_a^b s_1(x)\,\mathrm{d}x}\left(I - \int_a^b g(x)\,\mathrm{d}x\right). \tag{9}$$

This function becomes undefined when $\int_a^b s_1(x)\,\mathrm{d}x = 0$, and thus $s_1(x)$ must be selected to avoid this condition. By simply selecting $s_1(x) = 1$ Equation (9) becomes,

$$f(x, g(x)) = g(x) + \frac{1}{b-a}\left(I - \int_a^b g(x)\,\mathrm{d}x\right).$$

3.2. Integral and Linear Constraints

In this second example, let us consider the more complex case where, in addition to the integral constraint given in Equation (7), we also consider the additional linear constraints,

$$\alpha\, f(x_0) + \beta\, f(x_f) = 1,$$

where α, β, x_0, and x_f are all assigned. A sketch of this example is shown in Figure 1.

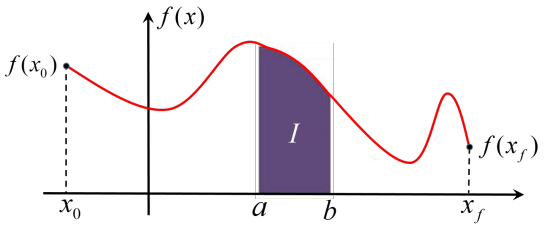


Figure 1. Integral and linear constraints example.

The constrained expression has the form,

$$f(x, g(x)) = g(x) + \eta_1\, s_1(x) + \eta_2\, s_2(x). \tag{10}$$

Then, by applying the constraints the system of linear equations,

$$\begin{Bmatrix} 1 - \alpha\, g_0 - \beta\, g_f \\ I - \int_a^b g(x)\,\mathrm{d}x \end{Bmatrix} = \begin{bmatrix} \alpha\, s_1(x_0) + \beta\, s_1(x_f) & \alpha\, s_2(x_0) + \beta\, s_2(x_f) \\ \int_a^b s_1(x)\,\mathrm{d}x & \int_a^b s_2(x)\,\mathrm{d}x \end{bmatrix} \begin{Bmatrix} \eta_1 \\ \eta_2 \end{Bmatrix} \tag{11}$$

is obtained. This system tells us that $s_1(x)$ and $s_2(x)$ functions can be any functions with the exception of those making the matrix singular. The matrix singularity occurs when

$$\left(\alpha\, s_1(x_0) + \beta\, s_1(x_f)\right) \int_a^b s_2(x)\,\mathrm{d}x = \left(\alpha\, s_2(x_0) + \beta\, s_2(x_f)\right) \int_a^b s_1(x)\,\mathrm{d}x.$$

For instance, by selecting $s_1(x) = 1$ and $s_2(x) = x$, the previous condition becomes,

$$(\alpha + \beta)(b + a) = 2(\alpha\, x_0 + \beta\, x_f)$$

which imply that, in order to avoid singularity, the following relationship must be satisfied,

$$\{\alpha, \quad \beta\} \begin{Bmatrix} b + a - 2\alpha x_0 \\ b + a - 2\beta x_f \end{Bmatrix} \neq 0.$$

When this condition is satisfied, the matrix of Equation (11) can be inverted and the coefficients, η_1 and η, can be computed. Then, the constrained expression for this problem is obtained by substituting the expressions found for η_1 and η_2 in Equation (10).

Problem #1

Now, let us consider the following DE to be solved on the $t \in [0, \pi]$ range, with a constraint in the form of an integral over the integration range,

$$\ddot{f} + f = 0 \qquad \text{subject to: } \begin{cases} f(0) = 1 \\ \int_0^{\pi} f(t)\, \mathrm{d}t = \pi \end{cases},$$

whose analytical solution is

$$f = \frac{\pi}{2} \sin t + \cos t.$$

In this problem, we build the constrained expression according to the formulation of Equation (2), where the *projection functionals* $\rho_k(x, g(x))$ are given by

$$\begin{cases} \rho_1 = 1 - g_0 \\ \rho_2 = \pi - \int_0^{\pi} g(t)\, \mathrm{d}t \end{cases}.$$

Given the support functions,

$$s_1(t) = 1, \qquad \text{and} \qquad s_2(t) = t,$$

the switching functions are computed as $\phi_j(t) = \sum_{i=1}^{2} \alpha_{ji}\, s_i(t)$, where the α_{ji} coefficients are computed just by inverting a matrix. For this problem,

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} \begin{bmatrix} s_1(0) & \int_0^{\pi} s_1(\tau)\, \mathrm{d}\tau \\ s_2(0) & \int_0^{\pi} s_2(\tau)\, \mathrm{d}\tau \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Then, the constrained expression can be built as:

$$f(t, g(t)) = g(t) + \phi_1(t)(1 - g_0) + \phi_2(t)\left(\pi - \int_0^{+\pi} g(\tau)\, \mathrm{d}\tau\right),$$

where the switching functions are

$$\phi_1(t) = \frac{\pi - 2t}{\pi} \qquad \text{and} \qquad \phi_2(t) = \frac{2t}{\pi^2}.$$

Now, expressing the free function according to $g(t, \boldsymbol{\xi}) = \boldsymbol{\xi}^{\intercal} \boldsymbol{h}(x(t))$ the terms of the DE become:

$$f(x, \boldsymbol{\xi}) = \left\{ \boldsymbol{h}(x) - \phi_1(x)\boldsymbol{h}_0 - \phi_2(x) \frac{1}{c} \int_{-1}^{1} \boldsymbol{h}(x)\, \mathrm{d}x \right\}^{\intercal} \boldsymbol{\xi} + \phi_1(x) + \pi\phi_2(x)$$
$$\ddot{f}(x, \boldsymbol{\xi}) = c^2\, \ddot{\boldsymbol{h}}(x)^{\intercal} \boldsymbol{\xi},$$

where the boundary conditions have been mapped to $x \in [x_0, x_f]$. For classic TFC $x \in [-1, +1]$, and for X-TFC $x \in [0, 1]$ according to Equations (3)–(5). The mapping coefficient is $c = \dfrac{x_f - x_0}{t_f - t_0}$.

With the function now expressed in terms of $\boldsymbol{\xi}$ the differential equation can now be transformed into a function of $\boldsymbol{\xi}$,

$$\left\{ c^2\,\ddot{\boldsymbol{h}}(x) + \boldsymbol{h}(x) - \phi_1(x)\,\boldsymbol{h}_0 - \phi_2(x)\frac{1}{c}\int_{-1}^{1} \boldsymbol{h}(x)\,\mathrm{d}x \right\}^{\mathrm{T}} \boldsymbol{\xi} = -\phi_1(x) - \pi\phi_2(x).$$

This equation can then be specified for a discrete values of x. This discretization yields to an over-determined linear system that can be solved by least-squares.

$$\mathbb{A}\,\boldsymbol{\xi} = \boldsymbol{b} \qquad \rightarrow \qquad \boldsymbol{\xi} = (\mathbb{A}^{\mathrm{T}}\mathbb{A})^{-1}\mathbb{A}^{\mathrm{T}}\,\boldsymbol{b}. \tag{12}$$

In this example, using $n = 100$ and $m = 20$, the classic TFC was executed with a computational time is of $\mathcal{O}(10^{-4})$ s, the average absolute error on the discretization points is of $\mathcal{O}(10^{-16})$, the variance of the absolute error on the discretization points is of $\mathcal{O}(10^{-16})$. For X-TFC, we set the following hyper-parameters $n = 100$, $L = 100$, Gaussian activation function, input weight and bias were sampled from $\mathcal{U}[-10, +10]$. The computational time is of $\mathcal{O}(10^{-4})$ s, the average absolute error on the discretization points is of $\mathcal{O}(10^{-16})$, the variance of the absolute error on the discretization points is of $\mathcal{O}(10^{-16})$. Table 1 reports the absolute error with respect to the analytical solutions obtained with classic TFC and X-TFC on 11 points uniformly distributed in the $[0, 1]$ range.

Table 1. Absolute errors on uniform test points for problem #1.

t/π	TFC	X-TFC
0.0	0.0	0.0
0.1	0.0	2.22×10^{-16}
0.2	0.0	2.22×10^{-16}
0.3	0.0	0.0
0.4	0.0	0.0
0.5	0.0	0.0
0.6	4.44×10^{-16}	4.44×10^{-16}
0.7	1.11×10^{-16}	2.22×10^{-16}
0.8	6.66×10^{-16}	0.0
0.9	1.66×10^{-16}	2.77×10^{-16}
1.0	2.22×10^{-16}	6.66×10^{-16}

3.3. Mixed Constraints

Consider a function subject to a value-level, relative, and integral constraint,

$$f(t_1) = f_1, \qquad f(t_0) = f(t_f), \qquad \text{and} \qquad \frac{1}{t_f - t_0}\int_{t_0}^{t_f} f(\tau)\,\mathrm{d}\tau = I.$$

By using Equation (2), the constrained expression has the form,

$$f(t, g(t)) = g(t) + \phi_1(t)\,\rho_1(t, g(t)) + \phi_2(t)\,\rho_2(t, g(t)) + \phi_3(t)\,\rho_3(t, g(t))$$

where the projection functionals, $\rho_k(t, g(t))$, are given by

$$\begin{cases} \rho_1 = f_1 - g(t_1) \\ \rho_2 = g(t_f) - f(t_0) \\ \rho_3 = I - \dfrac{1}{t_f - t_0}\displaystyle\int_{t_0}^{t_f} g(\tau)\,\mathrm{d}\tau \end{cases}$$

Given the support functions,

$$s_1(t) = 1, \qquad s_2(t) = t, \qquad \text{and} \qquad s_3(t) = t^2$$

the α_{ji} coefficients of the switching functions, $\phi_j(t) = \sum_{i=1}^{3} \alpha_{ji}\, s_i(t)$, are computed by inverting the support matrix. Specifically,

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} \begin{bmatrix} s_1(t_1) & s_1(t_0) - s_1(t_f) & \frac{1}{t_f - t_0} \int_{-\pi}^{+\pi} s_1(\tau)\, \mathrm{d}\tau \\ s_2(t_1) & s_2(t_0) - s_2(t_f) & \frac{1}{t_f - t_0} \int_{-\pi}^{+\pi} s_2(\tau)\, \mathrm{d}\tau \\ s_3(t_1) & s_3(t_0) - s_3(t_f) & \frac{1}{t_f - t_0} \int_{-\pi}^{+\pi} s_3(\tau)\, \mathrm{d}\tau \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Once the $\phi_j(t)$ are known, the constrained expression can be obtained as:

$$f(t, g(t)) = g(t) + \phi_1(t)\,(f_1 - g(t_1)) + \phi_2(t)\left(f(t_f) - g(t_0)\right) + \\ + \phi_3(t)\left(I - \frac{1}{t_f - t_0} \int_{t_0}^{t_f} g(\tau)\, \mathrm{d}\tau\right).$$

Problem #2

As an ultimate "stress-test" of the TFC method, a mixed-constraint case is considered where point, relative, and integral constraints are used in the solution of a differential equation, on the range $t \in [-\pi, \pi]$

$$\dddot{y} + \sin t\, \ddot{y} + (1 - t)\dot{y} + ty = f(t) \qquad \text{subject to:} \begin{cases} y(t_0) = 0 \\ y(t_f) = y(t_0) \\ \int_{-\pi}^{+\pi} y(\tau)\, \mathrm{d}\tau = -2\pi \end{cases}$$

where the forcing term is

$$f(t) = (t - 1)\sin^2 t + (2 + 2t - t^2 - 2\cos t)\sin t + t(t - 1)\cos t.$$

This problem admits the analytical solution,

$$y = (1 - t)\sin t.$$

Following the procedure explained, the constrained expression for this problem is,

$$y(t, g(t)) = g(t) + \phi_1(t)(-g_0) + \phi_2(t)\,(-g_f) + \phi_3(t)\left(-2\pi - \int_{-\pi}^{+\pi} g(\tau)\, \mathrm{d}\tau\right),$$

where the α_{ji} coefficients of the switching functions $\phi_j(t) = \sum_{i=1}^{3} \alpha_{ji}\, s_i(t)$ are computed by solving the following system:

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{bmatrix} \begin{bmatrix} s_1(t_0) & s_1(t_f) & \int_{-\pi}^{+\pi} s_1(\tau)\, \mathrm{d}\tau \\ s_2(t_0) & s_2(t_f) & \int_{-\pi}^{+\pi} s_2(\tau)\, \mathrm{d}\tau \\ s_3(t_0) & s_3(t_f) & \int_{-\pi}^{+\pi} s_3(\tau)\, \mathrm{d}\tau \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Now, expressing the free function according to $g(t) = \boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{h}(x(t))$ and rearranging terms leads us to the final constrained expression,

$$y(x,\boldsymbol{\xi}) = \left\{\boldsymbol{h}(x) - \phi_1(x)\,\boldsymbol{h}_0 - \phi_2(x)\,\boldsymbol{h}_f - \phi_3(x)\,\frac{1}{c}\int_{x_0}^{x_f}\boldsymbol{h}(x)\,\mathrm{d}x\right\}^{\mathrm{T}}\boldsymbol{\xi} - 2\pi\,\phi_3(x),$$

and by substituting this constrained expression into the differential equation (by evaluating its derivatives accordingly), we obtain a linear system that can be solved by least-squares using Equation (12).

Even in this case, the differential equation and boundary conditions must be mapped onto $x \in [x_0, x_f]$. For classic TFC $x \in [-1,+1]$, and for X-TFC $x \in [0,1]$ according to Equation (3), where the mapping coefficient is $c = \dfrac{x_f - x_0}{t_f - t_0}$.

For classic TFC we set $n = 35$ and $m = 30$. The computational time is of $\mathcal{O}(10^{-4})$ s, the average absolute error on the discretization points is of $\mathcal{O}(10^{-15})$, the variance of the absolute error on the discretization points is of $\mathcal{O}(10^{-30})$. For X-TFC we set the following hyper-parameters $n = 40$, $L = 40$, sinusoidal activation function, input weight and bias were sampled from $\mathcal{U}[-20,+20]$. The computational time is of $\mathcal{O}(10^{-4})$ s, the average absolute error on the discretization points is of $\mathcal{O}(10^{-15})$, the variance of the absolute error on the discretization points is of $\mathcal{O}(10^{-30})$.

3.4. Discussions

The results from these two problems emphasize the utility and convenience of using TFC. Once the constrained expression is defined, the method to solving the differential equation does not change regardless of the use of different constraints. Moreover, this makes the solution accuracy only dependent on the problem complexity. Finally, once the solution is computed (on the discretization points), we have an analytical representation of it. That is, no further manipulations are needed (e.g., interpolation) if we want to evaluate the solutions in points that are different from the discretization points. Indeed, as shown in the results of both problems we do not lose any accuracy when evaluating the solution on test points, as it would happen with some other state-of-the-art methods such as the Finite Element Method (FEM) [34].

4. TFC for Linear Ordinary Integro-Differential Equation

In this section we show how TFC is applied to solve linear ordinary Integro-Differential Equations (IDEs) using all constrained expressions derived from the formalism of Equation (1).

As first test problem, the linear Fredholm integro-differential equations is considered. Comparisons of the numerical results obtained with TFC, X-TFC, and the method published in Ref. [35] are presented.

4.1. Problem #1

Consider the following integro-differential equation:

$$\dot{y}(x) = 1 - \frac{1}{3}\,x + x\int_0^1 \tau\,y(\tau)\,\mathrm{d}\tau$$

with one constraint $y(0) = 0$, for $x \in [0,1]$. The analytical solution is $y(x) = x$. The constrained expression for this problem is simply

$$y(x,\boldsymbol{\xi}) = (\boldsymbol{h}(x) - \boldsymbol{h}_0)^{\mathrm{T}}\boldsymbol{\xi},$$

where $\boldsymbol{h}_0$ is the vector of the basis function computed at $x = 0$.

For classic TFC $n = 100$ discrete points and $m = 29$ basis functions were used. The computational time is of $\mathcal{O}(10^{-4})$ s, the average absolute error on the discretization

points is of $\mathcal{O}(10^{-5})$, and the variance of the absolute error on the discretization points is of $\mathcal{O}(10^{-9})$. For X-TFC the hyper-parameters were set to $n = 20$, $L = 20$, the activation function was sinusoidal, and the input weight and bias were sampled from $\mathcal{U}[-1, +1]$. The computational time obtained was of $\mathcal{O}(10^{-4})$ s, the average absolute error on the discretization points of $\mathcal{O}(10^{-4})$, the variance of the absolute error on the discretization points of $\mathcal{O}(10^{-8})$. The absolute errors obtained with classic TFC and X-TFC on 10 test points are reported in Figure 2 and compared with those reported by Ref. [35].

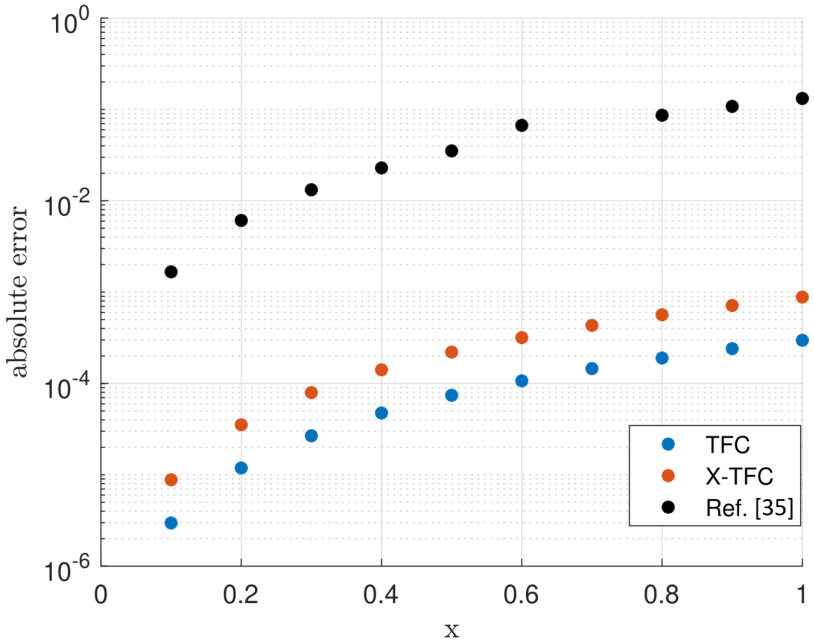


Figure 2. Absolute error on test points for problem #1.

4.2. Problem #2

The second test problem is on the following integro-differential equation:

$$\dot{y}(x) = x\,e^x + e^x - x + x\int_0^1 y(\tau)\,\mathrm{d}\tau,$$

with one constraint $y(0) = 0$, for $x \in [0, 1]$. The analytical solution is $y(x) = xe^x$. Also for this this problem the constrained expression is

$$y(x, \boldsymbol{\xi}) = (\boldsymbol{h}(x) - \boldsymbol{h}_0)^{\mathrm{T}}\boldsymbol{\xi}.$$

For classic TFC $n = 25$ discretization points and $m = 14$ basis functions were adopted. The computational time obtained is of $\mathcal{O}(10^{-4})$ s, the average absolute error on the discretization points is of $\mathcal{O}(10^{-16})$, and the variance of the absolute error on the discretization points is of $\mathcal{O}(10^{-32})$. For X-TFC the hyper-parameters were set to $n = 50$, $L = 50$, the activation function was the hyperbolic tangent, and the input weight and bias were sampled from $\mathcal{U}[-1, +1]$. The computational time is of $\mathcal{O}(10^{-4})$ s, the average absolute error on the discretization points is of $\mathcal{O}(10^{-14})$, and the variance of the absolute error on the discretization points is of $\mathcal{O}(10^{-27})$. The absolute errors on the test points obtained with classic TFC and X-TFC, are reported in Figure 3 and compared with those reported by Ref. [35].

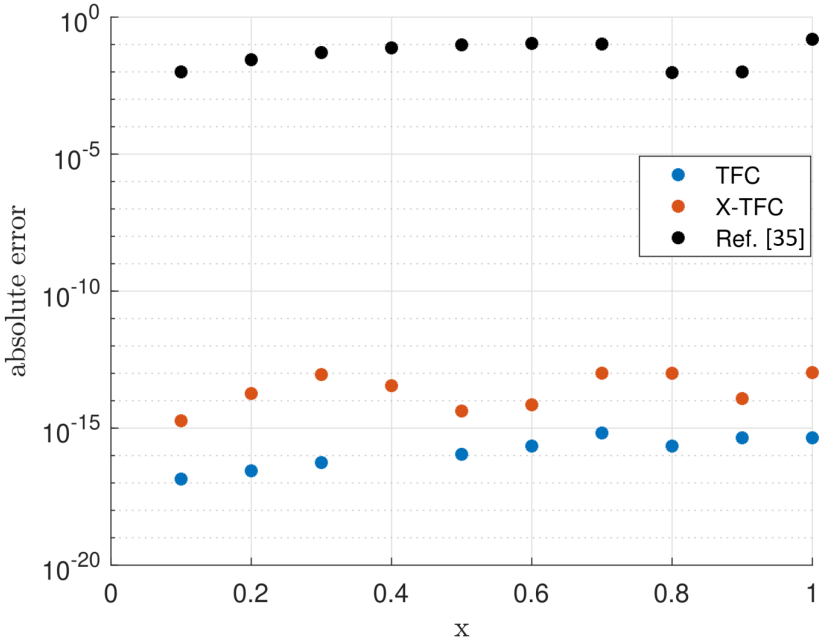


Figure 3. Absolute error on test points for problem #2.

4.3. Problem #3

The last test problem is a second-order integro-differential equation,

$$\ddot{y}(x) = e^x - x + x \int_0^1 \tau\, y(\tau)\, \mathrm{d}\tau,$$

subject to the initial value problem, $y(0) = 1$ and $\dot{y}(0) = 1$, where $x \in [0, 1]$. The analytical solution is $y(x) = e^x$. The constrained expression for this problem is:

$$y(x, \boldsymbol{\xi}) = \left(\boldsymbol{h}(x) - \boldsymbol{h}_0 - x\, \dot{\boldsymbol{h}}_0\right)^{\mathrm{T}} \boldsymbol{\xi} + x + 1.$$

To solve this problem, the classic TFC required $n = 9000$ points and $m = 1000$ basis functions. The computational time was of $\mathcal{O}(1)$ second, the average absolute error on the discretization points was of $\mathcal{O}(10^{-5})$, and the variance of the absolute error on the discretization points was of $\mathcal{O}(10^{-10})$. For X-TFC the setting was $n = 90$ and $L = 94$ hyper-parameters, hyperbolic sine activation function, and input weight and bias were sampled from $\mathcal{U}[-1, +1]$. The computational time was of $\mathcal{O}(10^{-4})$ s, the average absolute error on the discretization points of $\mathcal{O}(10^{-4})$, and the variance of the absolute error on the discretization points of $\mathcal{O}(10^{-7})$. The absolute errors on the test points obtained with classic TFC and X-TFC, are reported in Figure 4 and compared with those reported by Ref. [35].

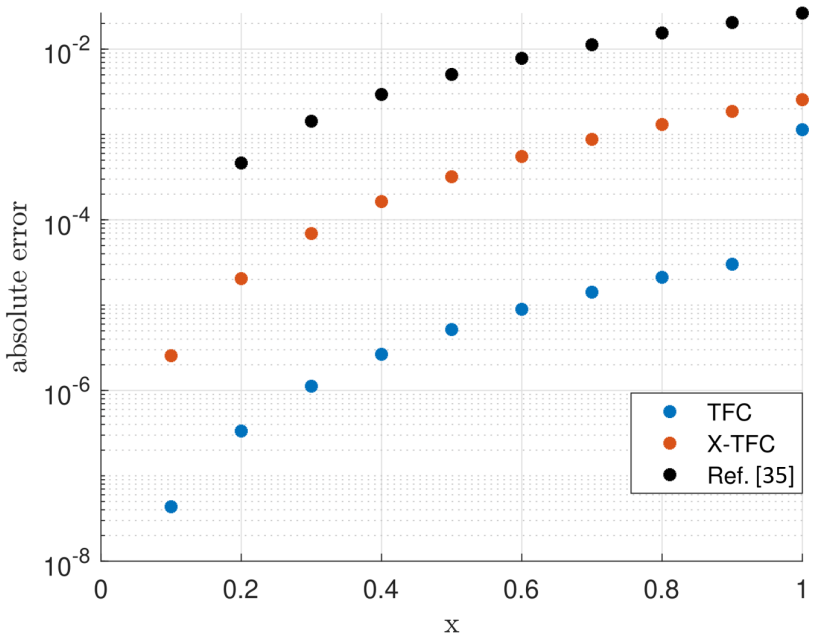


Figure 4. Absolute error on test points for problem #3.

4.4. Discussions

The discussions made for the linear ODE subjects to integral constraints are also valid for linear ordinary integro-differential equations. Nevertheless, the attentive reader will notice that there is a significant difference in terms of accuracy among problems 1 and 3 compared to problem 2. This is due to the fact that the kernels in the integral terms of the equations are different than one for problems 1 and 3 (e.g., for the both problems the kernel is t). This can have a negative impact on the solution accuracy as the function to be integrated can become very complex, and this can lead to numerical issues. For instance, when classic TFC is used in problems 1 and 3, where the kernel is simply t, the coefficients in front of the integrated polynomials become enormous as the number of basis functions (e.g., the degree of the Chebyshev polynomials) reaches above ten, causing numerical issues that impact negatively on the accuracy. Moreover, in many real world applications, such as in radiative transfer equation [17,36], the kernel can be very complex, making the analytical evaluation of the integral prohibitive.

The are several ways of mitigating these issues that are worth exploring. For instance, the integral could be evaluated with a Gauss-quadrature scheme or with a Monte Carlo method. The authors of this manuscript are investigating the possibility of evaluating the integral numerically using NNs. However, the investigation of different ways to overcome these limitations is not the focus of this work, and it will be explored by the authors in future papers.

5. Conclusions

This study presents an extension of the least-squares solution of linear differential equations studied in an earlier publication [5]. This paper aims to explore solutions using different boundary constraints including multi-valued, relative, and integral constraints for linear ordinary differential equations and for linear ordinary integro-differential equations. In all problems, a *constrained expression* is used to embed the constraints. This allows the integration range to be independent from the constraints location. In these expressions, the free function $g(x)$ was expressed as a linear combination of known basis functions and

unknown constant coefficients $\boldsymbol{\xi}$, either using Chebyshev polynomials, or shallow NNs for the X-TFC framework. Expressing $g(x)$ as a linear combination of basis functions and constant coefficients, the coefficient vector $\boldsymbol{\xi}$ remains linear and is solved by using a least-squares method. While this paper only solves linear differential equations as a test case, the proposed methodology can easily be extended to nonlinear differential equations by replacing the least-squares method with a nonlinear least-squares technique as introduced in prior research [6].

For all cases explored, the classic TFC and X-TFC methods consistently produce very fast and accurate solutions. Additionally, for all problems, the constraints are *analytically* satisfied since they are integrated into the *constrained expression*. In general, this means that the TFC methods decouple the constraints from the solution of the differential equation. Due to this, the solution range of the differential equation and, where the constraints are specified, is independent. This fact makes the TFC method a unified framework to solve differential equations with no more distinctions between the initial and boundary values problems, as well as any other constraints distribution problem.

Appendix A. Chebyshev and Legendre Orthogonal Polynomials

This appendix provides compact summaries of Chebyshev, $T_k(x)$, and Legendre, $L_k(x)$, orthogonal polynomials, which are defined in the xß$[-1,+1]$ range. These are:

Appendix A.1. Definition

Starting with $T_0 = L_0 = 1$ and $T_1 = L_1 = x$ these orthogonal polynomials can be conveniently defined recursively,

$$T_{k+1} = 2\,x\,T_k - T_{k-1} \qquad \text{and} \qquad L_{k+1} = \frac{2k+1}{k+1}\,x\,L_k - \frac{k}{k+1}\,L_{k-1}$$

and, specifically ($\forall k$),

$$T_k(-1) = L_k(-1) = (-1)^k \qquad \text{and} \qquad T_k(+1) = L_k(+1) = 1.$$

Appendix A.2. Orthogonality

The orthogonality is provided by the following integrals,

$$\int_{-1}^{+1} T_i(x)\,T_j(x)\,\frac{\mathrm{d}x}{\sqrt{1-x^2}} = \begin{cases} 0 & \text{if } i \neq j \\ \pi & \text{if } i = j = 0 \\ \pi/2 & \text{if } i = j \neq 0 \end{cases}$$

$$\int_{-1}^{+1} L_i(x)\,L_j(x)\,\mathrm{d}x = \frac{2}{2j+2}\,\delta_{ij}.$$

Appendix A.3. Derivatives

All derivatives of Chebyshev orthogonal polynomials can also be computed recursively. Starting from,

$$\frac{\mathrm{d}T_0}{\mathrm{d}z} = 0, \qquad \frac{\mathrm{d}T_1}{\mathrm{d}z} = 1, \qquad \text{and} \qquad \frac{\mathrm{d}^d T_0}{\mathrm{d}z^d} = \frac{\mathrm{d}^d T_1}{\mathrm{d}z^d} = 0 \quad (\forall\, d > 1),$$

the subsequent derivatives can be computed by:

$$\frac{\mathrm{d}^d T_{k+1}}{\mathrm{d}z^d} = 2\left(d\,\frac{\mathrm{d}^{d-1}T_k}{\mathrm{d}z^{d-1}} + z\,\frac{\mathrm{d}^d T_k}{\mathrm{d}z^d}\right) - \frac{\mathrm{d}^d T_{k-1}}{\mathrm{d}z^d} \quad (k \geq 1,\ \forall\, d \geq 1).$$

The derivatives of Legendre orthogonal polynomials can also be computed recursively. Starting from,

$$\frac{\mathrm{d}L_0}{\mathrm{d}z} = 0, \qquad \frac{\mathrm{d}L_1}{\mathrm{d}z} = 1, \qquad \text{and} \qquad \frac{\mathrm{d}^d L_0}{\mathrm{d}z^d} = \frac{\mathrm{d}^d L_1}{\mathrm{d}z^d} = 0 \quad (\forall\, d > 1),$$

the subsequent derivatives can be computed by,

$$\frac{\mathrm{d}^d L_{k+1}}{\mathrm{d}z^d} = \frac{2k+1}{k+1}\left(d\frac{\mathrm{d}^{d-1} L_k}{\mathrm{d}z^{d-1}} + z\frac{\mathrm{d}^d L_k}{\mathrm{d}z^d}\right) - \frac{k}{k+1}\frac{\mathrm{d}^d L_{k-1}}{\mathrm{d}z^d} \quad (k \geq 1,\ \forall\, d \geq 1).$$

Appendix A.4. Integral

- *Chebyshev indefinite.*

$$\int_{-1}^{x} T_0(z)\,\mathrm{d}z = x + 1, \qquad \int_{-1}^{x} T_1(z)\,\mathrm{d}z = \frac{1}{2}\left(x^2 - 1\right),$$
$$\text{then} \qquad \rightarrow \qquad \int_{-1}^{x} T_k(z)\,\mathrm{d}z = \frac{k\,T_{k+1}}{k^2-1} - \frac{x\,T_k}{k-1}$$

- *Chebyshev full range.*

$$\int_{-1}^{+1} T_k(x)\,\mathrm{d}x = \begin{cases} 0 & \text{for } k \text{ odd} \\ \dfrac{(-1)^k + 1}{1 - k^2} & \text{for } k \text{ even} \end{cases}$$

- *Chebyshev internal range* $(-1 \leq a < b \leq +1)$

$$\int_{a}^{b} T_k(x)\,\mathrm{d}x = \frac{k\left[T_{k+1}(b) - T_{k+1}(a)\right]}{k^2-1} - \frac{b\,T_k(b) - a\,T_k(a)}{k-1}$$

- *Legendre indefinite.*

$$\int_{-1}^{x} L_0(z)\,\mathrm{d}z = x + 1, \qquad \int_{-1}^{x} L_1(z)\,\mathrm{d}z = \frac{1}{2}\left(x^2 - 1\right),$$
$$\text{then} \qquad \rightarrow \qquad \int_{-1}^{x} L_k(z)\,\mathrm{d}z = \frac{L_{k+1}(x) - L_{k-1}(x)}{2k+1}$$

- *Legendre full range.*

$$\int_{-1}^{+1} L_0(x)\,\mathrm{d}x = 2 \qquad \text{and} \qquad \int_{-1}^{+1} L_k(x)\,\mathrm{d}x = 0, \quad \forall\, k \neq 0.$$

- *Legendre internal range* $(-1 \leq a < b \leq +1)$

$$\int_{a}^{b} L_k(x)\,\mathrm{d}x = \frac{L_{k+1}(b) - L_{k+1}(a) + L_{k-1}(a) - L_{k-1}(b)}{2k+1}.$$

Author Contributions: Conceptualization, D.M.; methodology, D.M., M.D.F., E.S. and A.D.; software, M.D.F., E.S., A.D. and D.M.; validation, M.D.F., E.S. and D.M.; formal analysis, D.M., M.D.F. and E.S.; writing—original draft preparation, D.M., M.D.F., E.S. and A.D.; writing—review and editing, D.M., E.S. and M.D.F.; supervision, R.F. and D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors acknowledge Hunter Johnston for the initial investigation on integral constraints using the Theory of Functional Connections.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Mortari, D. The theory of connections: Connecting points. *Mathematics* **2017**, *5*, 57.
2. Mortari, D.; Leake, C. The multivariate theory of connections. *Mathematics* **2019**, *7*, 296.
3. Johnston, H.R. The theory of functional connections: A journey from theory to application. Ph.D. Thesis, Texas A&M University, College Station, TX, USA, 2021.
4. Leake, C.D. The Multivariate Theory of Functional Connections: An n-dimensional Constraint Embedding Technique Applied to Partial Differential Equations. Ph.D. Thesis, Texas A&M University, College Station, TX, USA, 2021.
5. Mortari, D. Least-squares solution of linear differential equations. *Mathematics* **2017**, *5*, 48.
6. Mortari, D.; Johnston, H.; Smith, L. High accuracy least-squares solutions of nonlinear differential equations. *J. Comput. Appl. Math.* **2019**, *352*, 293–307.
7. Mortari, D.; Furfaro, R. Univariate theory of functional connections applied to component constraints. *Math. Comput. Appl.* **2021**, *26*, 9.
8. Leake, C.; Johnston, H.; Mortari, D. The multivariate theory of functional connections: Theory, proofs, and application in partial differential equations. *Mathematics* **2020**, *8*, 1303.
9. Schiassi, E.; Furfaro, R.; Leake, C.; De Florio, M.; Johnston, H.; Mortari, D. Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations. *Neurocomputing* **2021**, *457*, 334–356.
10. Leake, C.; Mortari, D. Deep theory of functional connections: A new method for estimating the solutions of partial differential equations. *Mach. Learn. Knowl. Extr.* **2020**, *2*, 37–55.
11. Gil, A.; Segura, J.; Temme, N.M. *Numerical Methods for Special Functions*; SIAM: Philadelphia, PA, USA, 2007.
12. Lanczos, C. *Applied Analysis*; Courier Corporation: Chelmsford, MA, USA, 1988.
13. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707.
14. Wang, Y.; Topputo, F. A TFC-based homotopy continuation algorithm with application to dynamics and control problems. *J. Comput. Appl. Math.* **2021**, *401*, 113777.
15. Mortari, D.; Arnas, D. Bijective mapping analysis to extend the theory of functional connections to non-rectangular 2-dimensional domains. *Mathematics* **2020**, *8*, 1593.
16. Schiassi, E.; De Florio, M.; D'Ambrosio, A.; Mortari, D.; Furfaro, R. Physics-informed neural networks and functional interpolation for data-driven parameters discovery of epidemiological compartmental models. *Mathematics* **2021**, *9*, 2069.
17. De Florio, M.; Schiassi, E.; Furfaro, R.; Ganapol, B.D.; Mostacci, D. Solutions of Chandrasekhar's basic problem in radiative transfer via theory of functional connections. *J. Quant. Spectrosc. Radiat. Transf.* **2021**, *259*, 107384.
18. De Florio, M.; Schiassi, E.; Ganapol, B.D.; Furfaro, R. Physics-informed neural networks for rarefied-gas dynamics: Thermal creep flow in the Bhatnagar–Gross–Krook approximation. *Phys. Fluids* **2021**, *33*, 047110.
19. Mai, T.; Mortari, D. Theory of functional connections applied to nonlinear programming under equality constraints. *arXiv* **2019**, arXiv:1910.04917.
20. Yassopoulos, C.; Leake, C.; Reddy, J.; Mortari, D. Analysis of Timoshenko–Ehrenfest beam problems using the theory of functional connections. *Eng. Anal. Bound. Elem.* **2021**, *132*, 271–280.
21. Johnston, H.; Mortari, D. Least-squares solutions of boundary-value problems in hybrid systems. *J. Comput. Appl. Math.* **2021**, *393*, 113524.
22. Johnston, H.; Leake, C.; Mortari, D. Least-squares solutions of eighth-order boundary value problems using the theory of functional connections. *Mathematics* **2020**, *8*, 397.
23. Leake, C.; Johnston, H.; Smith, L.; Mortari, D. Analytically embedding differential equation constraints into least squares support vector machines using the theory of functional connections. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 1058–1083.
24. Rao, A.V. A survey of numerical methods for optimal control. *Adv. Astronaut. Sci.* **2009**, *135*, 497–528.
25. De Almeida Junior, A.K.; Johnston, H.; Leake, C.; Mortari, D. Fast 2-impulse non-Keplerian orbit transfer using the theory of functional connections. *Eur. Phys. J. Plus* **2021**, *136*, 1–21.
26. Schiassi, E.; D'Ambrosio, A.; Johnston, H.; De Florio, M.; Drozd, K.; Furfaro, R.; Curti, F.; Mortari, D. Physics-informed extreme theory of functional connections applied to optimal orbit transfer. In Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, CA, USA, 9–13 August 2020; pp. 9–13.
27. Johnston, H.; Mortari, D. Orbit propagation via the theory of functional connections. In Proceedings of the 2019 AAS/AIAA Astrodynamics Specialist Conference, Portland, ME, USA, 11–15 August 2019.
28. De Almeida Junior, A.; Johnston, H.; Leake, C.; Mortari, D. Evaluation of transfer costs in the earth-moon system using the theory of functional connections. In Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, CA, USA, 9–13 August 2020.

29. Drozd, K.; Furfaro, R.; Schiassi, E.; Johnston, H.; Mortari, D. Energy-optimal trajectory problems in relative motion solved via Theory of Functional Connections. *Acta Astronaut.* **2021**, *182*, 361–382.
30. Schiassi, E.; D'Ambrosio, A.; Johnston, H.; Furfaro, R.; Curti, F.; Mortari, D. Complete energy optimal landing on small and large planetary bodies via theory of functional connections. In Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, CA, USA, 9–13 August 2020; pp. 20–557.
31. Johnston, H.; Schiassi, E.; Furfaro, R.; Mortari, D. Fuel-efficient powered descent guidance on large planetary bodies via theory of functional connections. *J. Astronaut. Sci.* **2020**, *67*, 1521–1552.
32. D'Ambrosio, A.; Schiassi, E.; Curti, F.; Furfaro, R. Pontryagin neural networks with functional interpolation for optimal intercept problems. *Mathematics* **2021**, *9*, 996.
33. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501.
34. Lagaris, I.E.; Likas, A.; Fotiadis, D.I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Networks* **1998**, *9*, 987–1000.
35. Darania, P.; Ebadian, A. A method for the numerical solution of the integro-differential equations. *Appl. Math. Comput.* **2007**, *188*, 657–668.
36. Mishra, S.; Molinaro, R. Physics informed neural networks for simulating radiative transfer. *J. Quant. Spectrosc. Radiat. Transf.* **2021**, *270*, 107705.

Appendix B

De Florio et al. (2021) Radiative Transfer

Reproduced from De Florio, M., Schiassi, E., Ganapol, B. D., & Furfaro, R. (2021). Solutions of Chandrasekhar's basic problem in radiative transfer via theory of functional connections. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 259, 107384. https://doi.org/10.1016/j.jqsrt.2020.107384 [8], with the permission of Elsevier.

Journal of Quantitative Spectroscopy & Radiative Transfer 259 (2021) 107384

Contents lists available at ScienceDirect

Journal of Quantitative Spectroscopy & Radiative Transfer

journal homepage: www.elsevier.com/locate/jqsrt

Solutions of Chandrasekhar's basic problem in radiative transfer via theory of functional connections

Mario De Florio[a], Enrico Schiassi[a], Roberto Furfaro[a,b,*], Barry D. Ganapol[b], Domiziano Mostacci[c]

[a] Department of Systems & Industrial Engineering, The University of Arizona, Tucson, AZ, USA
[b] Department of Aerospace & Mechanical Engineering, The University of Arizona, Tucson, AZ, USA
[c] Laboratorio di Ingegneria Nucleare di Montecuccolino, Alma Mater Studiorum - Università di Bologna, Bologna, Italy

ARTICLE INFO

Article history:
Received 12 July 2020
Revised 8 October 2020
Accepted 10 October 2020
Available online 14 October 2020

Keywords:
Transport theory
Radiative transfer
Theory of functional connections
Mie scattering
Haze L problem
Least-squares

ABSTRACT

We present a novel approach to solving Chandrasekhar's problem in radiative transfer using the recently developed *Theory of Functional Connections*. The method is designed to elegantly and accurately solve the Linear Boundary Value Problem from the angular discretization of the integrodifferential Boltzmann equation for Radiative Transfer. The proposed algorithm falls under the category of numerical methods for the solution of radiative transfer equations. This new method's accuracy is tested via comparison with the published benchmarks for Mie and Haze L scattering laws.

1. Introduction

Radiative transfer theory, i.e. the study of how electromagnetic radiation propagates through a medium, is described by the Radiative Transfer Equation (RTE). The latter mathematically describes how particles moving through a medium are affected by a variety of physical processes, including absorption, scattering and emission. In this work, we consider what is known as Chandrasekhar's basic problem in radiative transfer [1], i.e. the problem of computing the intensity of the photons in a finite slab illuminated by a beam of light incident on one surface.

Historically, several methods have been proposed in the literature to solve the resulting RTE or the integro-differential Boltzmann equation for photons transport. The Discrete Ordinates Method (DOM) has been an effective, purely numerical method of solution available to the transport analysts for nearly 70 years. First proposed by Wick [2] in 1943 and later extensively developed by Chandrasekhar [1], DOM has undergone numerous changes and improvements over the years, with the goal of obtaining more accurate solutions. One of such changes was given by Carlson [3], who reformulated the DOM by introducing a discretization of the continuous angular variable. This DOM-based approach was named the S_n (S stands for segment, since it uses line segments to approximate the μ-dependence of the angular flux) method and it has been widely employed for many radiative transfer problems [4–6]. Further methods were subsequently explored. For example, in the C_N method, developed by Benoist and Kavenoky [7], the angular flux is approximated by an expansion of a complete orthogonal basis, such as a series of Legendre polynomials. The C_N method in turn was further modified by Siewert and Benoist, who developed the F_N method [8] in 1979. F_N was employed to solve the half-space albedo problem and the half-space constant-source problem. Initially used to solve problems in neutron transport theory, it was subsequently applied in several radiative transfer applications [9–11]. In 1980, Benassi et al. [12] proposed a new P_N method, in which exact particular solutions and the spherical harmonics method [13] were combined to compute the partial heat fluxes for a general class of radiative heat-transfer problems. Siewert and Thomas [14] later in 1989, extended the P_N approximation to the equation of transfer for the case of a general inhomogeneous source term. Finally, at the end of the last millennium (1999), the first version of the Analytical Discrete Ordinates (ADO) method was presented by Barichello and Siewert [15]. Conceived as a modern version of the DOM, the ADO method generated one of the most accurate numerical results for radiative transfer problems [16]. Recently, ADO has been extended to solve radiative transfer

* Corresponding author at: Department of Systems & Industrial Engineering, The University of Arizona, 1127 E James E Rogers Way, Tucson, AZ 85721, USA.
E-mail address: robertof@email.arizona.edu (R. Furfaro).

https://doi.org/10.1016/j.jqsrt.2020.107384
0022-4073/

problems in vegetation [17]. However, research in generating accurate solutions for the basic Chandrasekhar problem and the general RTE continued. In the last decade, Ganapol refined the S_n method in an elegant way, giving life to the Converged S_n method, or CS_n [18]. It is basically an extrapolation of the conventional, fully discretized, S_n solution to near-zero spatial and angular discretization. The distinctiveness of the CS_n method lies in its simplicity, which makes it accessible to those who are not proficient in solving transport equations, while achieving highly accurate solutions. A multi-problem strategy based on S_N acceleration is presented by Picca et al. [19].

All the above mentioned work has shown that in the last 70 years, researchers have developed several methods in order to achieve accurate and efficient numerical solutions for many problems in the radiative transfer field.

In this paper, we propose and develop a novel approach that will open new paths to solve RTEs. The proposed approach applies the *Theory of Functional Connections* (TFC), recently designed and developed by Mortari [20], to solve Chandrasekhar's Basic Problem in Radiative Transfer. TFC was born as a new framework for functional interpolation [20]. It has been successfully applied to solving both linear and non-linear differential equations [21,22] to machine error accuracy in milliseconds. Moreover the authors have successfully applied TFC to space guidance problems like Energy and Fuel Optimal Landing on large and small planetary bodies [23–25].

The method transforms ODEs with boundary and/or initial conditions into unconstrained optimization problems, which are solved via least-square (or iterative least-square for the non-linear case), by embedding the conditions into a constrained expression containing the ICs/BCs and a freely chosen function. Such an expression can be expressed in the following general form [20]:

$$y(t) = g(t) + \sum_{k=1}^{n} \eta_k p_{k(t)} = g(t) + \boldsymbol{\eta}^T \mathbf{p}(t) \tag{1}$$

Eq. (1) is called *constrained expression* (CE). The $p_k(t)$ are n assigned linearly independent functions and the $g(t)$ is a free function, which must be linearly independent from the $p_k(t)$ functions. The values of the η_k parameters are calculated by imposing the set of n linear constraints in $y(t)$. By doing so, the constraints for the differential equations are satisfied for any chosen $g(t)$. So far, it has been demonstrated that TFC method in solving ODEs has the best performance when $g(t)$ is selected as an expansion of basis function such as the Chebyshev orthogonal polynomials [21]. That is:

$$g(t) = \sum_{i=0}^{m_{op}-1} h_i(x)\xi_i = \mathbf{h}^T(t)\boldsymbol{\xi} \tag{2}$$

where the polynomial constant coefficients $\boldsymbol{\xi}$ are the unknown to be computed, $h_i(x)$ is the i_{th} orthogonal polynomial, and m_{op} is the number of basis considered. By substituting the CE in the ODE, for linear problems as the one tackled in this paper, we will obtain a linear system of the type $A\xi = b$to be computed via LS to find the solution $y(t)$.

Here, we propose to extend the TFC framework to solve RTEs. Starting from Chandrasekhar's Basic RTE and applying an angular discretization as in Siewert [16], the basic integrodifferential Boltzmann equation is transformed into a system of ODEs. The latter is cast as a Linear Boundary Value Problem (BVP) that is solved via the TFC method. Constrained expressions are generated and the resulting free-function is expanded in the basis of Chebyshev orthogonal polynomials. Such expansion ultimately results in a linear system that is solved via a Least-Square algorithm. Most importantly, the proposed method is shown to be elegant, concise, versatile and yields a straightforward numerical implementation.

The paper is organized as follows. In Section 2 a summary of the TFC methodology is reported together with an example of methodological application of the RTE to the simple Two-Stream Approximation. In Section 3, the basic Chandrasekhar RTE is stated and formulated. In Sections 4 and 5, the new TFC-based numerical algorithm is detailed. Finally, numerical results are presented and discussed in Section 6.

2. TFC approach to solving ODE boundary value problems

2.1. Method

For the convenience of the reader, how the TFC method works to tackle generic linear and non-linear ODEs is summarized in Fig. 1, and following, we give a simple example of the application of TFC to solving the class of BVPs. The detailed derivation can be found in Mortari [21], Mortari et al. [22]. Consider solving a first-order boundary value problem such that:

$$F(\tau, y, \dot{y}) = 0 \quad \text{subject to:} \quad y(\tau_0) = y_0. \tag{3}$$

The CE is built from Eq. (1) by setting $p_1(\tau) = 1$(see ref.[21,22] for the details of this decision):

$$y(\tau) = g(\tau) + \eta \tag{4}$$

By applying the constraint, we get:

$$y(\tau_0) = y_0 = g_0 + \eta; \qquad \Longrightarrow \qquad \eta_1 = y_0 - g_0$$

which is plugged into Eq. (4) to obtain at the final CE:

$$y(\tau) = g(\tau) + (y_0 - g_0) \tag{5}$$

which represents all possible functions satisfying the boundary value constraint. The derivative then follows:

$$\dot{y}(\tau) = \dot{g}(\tau). \tag{6}$$

By substituting Eqs. (5) and (6) into the Eq. (3), the original differential equation is transformed into a new differential equation $\tilde{F}$,which now is boundary free, and it is only a function of the independent variable τ and the free-function $g(\tau)$, that is:

$$\tilde{F}(\tau, g, \dot{g}) = 0.$$

This differential equation is now *unconstrained* and will always satisfy the boundary-value due to the way Eq. (4) is built. In order to solve this problem numerically, we define the function $g(\tau)$ as an expansion of some known basis with unknown constant coefficients such that:

$$g(\tau) = \boldsymbol{h}(\tau)^T \boldsymbol{\xi} \tag{7}$$

where $\boldsymbol{\xi}$ is a $m_{op} \times 1$ vector of unknown constant coefficients where m_{op} is the number of basis functions used. In general, the basis functions are defined on an inconsistent domain $x \in [x_0, x_f]$ (Chebyshev polynomials are defined on $x \in [-1, +1]$,Fourier series is defined on $x \in [-\pi, +\pi]$,etc.). Thus, these functions must be linearly mapped to the independent variable τ. This can be done using the following linear transformation:

$$x = x_0 + \frac{x_f - x_0}{t_f - t_0}(\tau - \tau_0) \quad \leftrightarrow \quad \tau = \tau_0 + \frac{\tau_f - \tau_0}{x_f - x_0}(x - x_0).$$

where $\tau \in [\tau_0, \tau_f]$.

Defining:

$$c = \frac{x_f - x_0}{\tau_f - \tau_0} \tag{8}$$

by the derivative chain rule, the derivative of Eq. (7) is,

$$\frac{\mathrm{d}g(\tau)}{\mathrm{d}\tau} = c\,\boldsymbol{\xi}^T \frac{\mathrm{d}\boldsymbol{h}(x)}{\mathrm{d}x},$$

which defines all mappings of the free-function. Finally, the domain τ must be discretized by M points. In this paper, we consider $g(\tau)$ as an expansion of Chebyshev polynomials. By defining

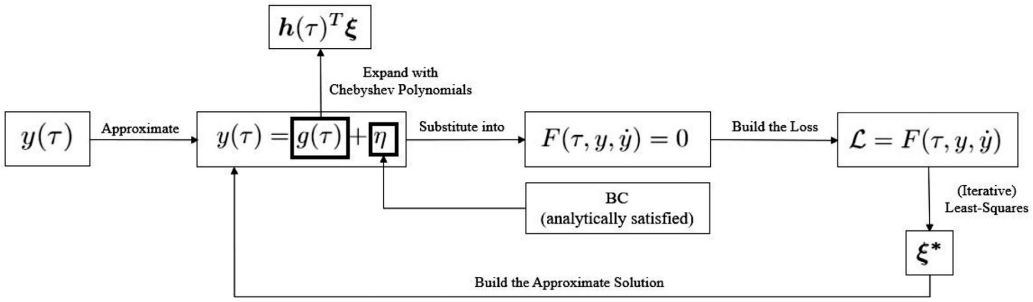


Fig. 1. Schematic of the general TFC framework to solve linear and non-linear ODEs with one constraint in one point.

the free function in this way and then discretizing the domain of the differential equations, $\tilde{F}$becomes:

$$\tilde{F}(\boldsymbol{\xi}) = 0,$$

which is now a function (linear or nonlinear) of only the unknown coefficients $\boldsymbol{\xi}$. Eq. () is now an *unconstrained* optimization problem that can be solved via different optimization schemes. In this paper, as we deal with a linear problem, we use a linear LS to solve for the unknown coefficients $\boldsymbol{\xi}$.

2.2. Example: radiative transfer problem with two-stream approximation

This example shows how to apply the TFC method to solve a simple system of ODEs. The system is the following:

$$\mu\frac{dy_1}{d\tau} + y_1(\tau) = \frac{\omega}{2}[y_1(\tau) + y_2(\tau)] + \frac{\omega}{2}S_0 e^{-\frac{\tau}{\mu_0}} \tag{9}$$

$$-\mu\frac{dy_2}{d\tau} + y_2(\tau) = \frac{\omega}{2}[y_2(\tau) + y_1(\tau)] + \frac{\omega}{2}S_0 e^{-\frac{\tau}{\mu_0}}, \tag{10}$$

subject to,

$$y_1(0) = 0 \tag{11}$$

$$y_2(\tau_0) = 0 \tag{12}$$

that represents an isotropic scattering radiative transfer equation with two-stream approximation [26], arising from Chandrasekhar [1], explained in details in the next section. The solutions of this system of ODEs represent the trend of scattered photons (forward and backward) through a mono-dimensional slab of thickness τ_0. For this example,the following parameters are used:

- $\mu = 0.5$is the cosine of the polar angle
- $\mu_0 = 0.8$is the cosine of incident beam angle
- $\tau_0 = 1$is the optical thickness
- $\omega = 0.9$is the single scattering albedo
- $S_0 = 1$is the source intensity

The detailed explanation of these parameters physically represent will be given in the next section. The unknowns are $y_1(\tau)$ and $y_2(\tau)$ that represent the forward and backward scattered photons fluxes respectively.

The first step of the TFC framework is to build the constrained expressions (CE) for the unknowns. According to previous explanation, the CEs are:

$$y_1(\tau) = (\boldsymbol{h} - \boldsymbol{h}_0)^T\boldsymbol{\xi}_1 \tag{13}$$

$$y_2(\tau) = \left(\boldsymbol{h} - \boldsymbol{h}_f\right)^T\boldsymbol{\xi}_2 \tag{14}$$

where $\boldsymbol{h}$ are the Chebyshev polynomials, and $\boldsymbol{h}_0$ and $\boldsymbol{h}_f$ are the Chebyshev polynomials computed at $\tau = 0$and $\tau = \tau_0$,respectively.

The derivatives of the CEs are:

$$\frac{dy_1}{d\tau} = c\,\boldsymbol{h}'^T\boldsymbol{\xi}_1 \tag{15}$$

$$\frac{dy_2}{d\tau} = c\,\boldsymbol{h}'^T\boldsymbol{\xi}_2 \tag{16}$$

where $\boldsymbol{h}'$ are the derivatives of $\boldsymbol{h}$ with respect to the variable x, and c is the mapping coefficient from τ to x (and vice-versa) as defined in Eq. (8).

By plugging the CEs and their derivatives into the system of ODEs, we get the following:

$$\mu\,c\,\boldsymbol{h}'\boldsymbol{\xi}_1 + (\boldsymbol{h} - \boldsymbol{h}_0)^T\boldsymbol{\xi}_1 = \frac{\omega}{2}\left[(\boldsymbol{h} - \boldsymbol{h}_0)^T\boldsymbol{\xi}_1 + \left(\boldsymbol{h} - \boldsymbol{h}_f\right)^T\boldsymbol{\xi}_2\right] + \frac{\omega}{2}S_0 e^{-\frac{\tau}{\mu_0}} \tag{17}$$

$$-\mu\,c\,\boldsymbol{h}'\boldsymbol{\xi}_2 + \left(\boldsymbol{h} - \boldsymbol{h}_f\right)^T\boldsymbol{\xi}_2 = \frac{\omega}{2}\left[\left(\boldsymbol{h} - \boldsymbol{h}_f\right)^T\boldsymbol{\xi}_2 + (\boldsymbol{h} - \boldsymbol{h}_0)^T\boldsymbol{\xi}_1\right] + \frac{\omega}{2}S_0 e^{-\frac{\tau}{\mu_0}}, \tag{18}$$

The losses are:

$$\mathcal{L}_1 = \mu\,c\,\boldsymbol{h}'\boldsymbol{\xi}_1 + (\boldsymbol{h} - \boldsymbol{h}_0)^T\boldsymbol{\xi}_1 - \frac{\omega}{2}\left[(\boldsymbol{h} - \boldsymbol{h}_0)^T\boldsymbol{\xi}_1 + \left(\boldsymbol{h} - \boldsymbol{h}_f\right)^T\boldsymbol{\xi}_2\right] - \frac{\omega}{2}S_0 e^{-\frac{\tau}{\mu_0}} \tag{19}$$

$$\mathcal{L}_2 = -\mu\,c\,\boldsymbol{h}'\boldsymbol{\xi}_2 + \left(\boldsymbol{h} - \boldsymbol{h}_f\right)^T\boldsymbol{\xi}_2 - \frac{\omega}{2}\left[\left(\boldsymbol{h} - \boldsymbol{h}_f\right)^T\boldsymbol{\xi}_2 + (\boldsymbol{h} - \boldsymbol{h}_0)^T\boldsymbol{\xi}_1\right] - \frac{\omega}{2}S_0 e^{-\frac{\tau}{\mu_0}} \tag{20}$$

As the problem is linear, by imposing the losses to be equal zero and re-arranging the terms we get the following system of linear algebraic equations:

$$\left[\mu\,c\,\boldsymbol{h}' + (\boldsymbol{h} - \boldsymbol{h}_0) - \frac{\omega}{2}(\boldsymbol{h} - \boldsymbol{h}_0)\right]^T\boldsymbol{\xi}_1 - \left[\frac{\omega}{2}\left(\boldsymbol{h} - \boldsymbol{h}_f\right)^T\right]\boldsymbol{\xi}_2 = \frac{\omega}{2}S_0 e^{-\frac{\tau}{\mu_0}} \tag{21}$$

$$\left[-\frac{\omega}{2}(\boldsymbol{h} - \boldsymbol{h}_0)^T\right]\boldsymbol{\xi}_1 + \left[-\mu\,c\,\boldsymbol{h}' + \left(\boldsymbol{h} - \boldsymbol{h}_f\right) - \frac{\omega}{2}\left(\boldsymbol{h} - \boldsymbol{h}_f\right)\right]^T\boldsymbol{\xi}_2 = \frac{\omega}{2}S_0 e^{-\frac{\tau}{\mu_0}} \tag{22}$$

Table 1
ADO and TFC forward scattered flux y_1. The latter is obtained with $M = 100$and $m_{op} = 40$. $M$is the number of discretization points on τ and m_{op} is the number of basis used.

Slab Points	ADO	TFC
0	$1.110223024625157 \times 10^{-16}$	0
0.1	0.121476763490043	0.121476763490043
0.2	0.217584323416591	0.217584323416591
0.3	0.291745312795815	0.291745312795815
0.4	0.346937565918151	0.346937565918151
0.5	0.385747980285517	0.385747980285517
0.6	0.410419885421222	0.410419885421222
0.7	0.422894687247150	0.422894687247150
0.8	0.424848465764781	0.424848465764781
0.9	0.417724123526887	0.417724123526887
1	0.402759611584474	0.402759611584474

Table 2
ADO and TFC backward scattered flux y_2. The latter is obtained with $M = 100$and $m_{op} = 40$.

Slab points	ADO	TFC
0	0.504726885720671	0.504726885720671
0.1	0.467993882643466	0.467993882643466
0.2	0.427166877800790	0.427166877800790
0.3	0.382783602872154	0.382783602872154
0.4	0.335285178981115	0.335285178981115
0.5	0.285027561759344	0.285027561759344
0.6	0.232291508206183	0.232291508206183
0.7	0.177291238875862	0.177291238875862
0.8	0.120181947999640	0.120181947999640
0.9	0.061066295681211	0.061066295681211
1	$-5.551115123125783 \times 10^{-17}$	0

In matrix form the system becomes:

$$\underbrace{\begin{bmatrix} \mu c\boldsymbol{h}'^T + (\boldsymbol{h}-\boldsymbol{h}_0)^T - \frac{\omega}{2}(\boldsymbol{h}-\boldsymbol{h}_0)^T & -\frac{\omega}{2}\left(\boldsymbol{h}-\boldsymbol{h}_f\right)^T \\ -\frac{\omega}{2}(\boldsymbol{h}-\boldsymbol{h}_0)^T & -\mu c\boldsymbol{h}'^T + \left(\boldsymbol{h}-\boldsymbol{h}_f\right)^T - \frac{\omega}{2}\left(\boldsymbol{h}-\boldsymbol{h}_f\right)^T \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \boldsymbol{\xi}_1 \\ \boldsymbol{\xi}_2 \end{bmatrix}}_{\xi} = \underbrace{\begin{bmatrix} \frac{\omega}{2} S_0 e^{-\frac{\tau}{\mu_0}} \\ \frac{\omega}{2} S_0 e^{-\frac{1}{\mu_0}} \end{bmatrix}}_{b} \tag{23}$$

As the problem is linear, we can solve for $\boldsymbol{\xi}$ using a least-squares. That is:

$$\boldsymbol{\xi} = \left(A^T A\right)^{-1} A^T b \tag{24}$$

Once $\boldsymbol{\xi}$ are computed, the final solutions are built by plugging $\boldsymbol{\xi}$ into the CEs (13), as shown in Fig. 1.

The results are reported in Tables 1 and 2, where they are compared to the results computed according to the ADO method. As we can see, the boundary conditions are analytically satisfied.

3. Radiative transfer equation: problem formulation

As we previously mentioned, we consider Chandrasekhar's basic problem for Radiative Transfer for an incoming beam at $x = 0$,as shown in Fig. 2, following the notation used in Siewert [16], Benassi et al. [27] the radiative transfer equation is written as:

$$\mu \frac{\partial}{\partial \tau} I(\tau, \mu, \phi) + I(\tau, \mu, \phi) = \frac{\omega}{4\pi} \int_{-1}^{1} \int_{0}^{2\pi} p(\cos\Theta) I(\tau, \mu', \phi')\, d\phi'\, d\mu' \tag{25}$$

where Θ is the scattering angle, $\mu \in [-1, 1]$is the cosine of the polar angle, $\tau \in [0, \tau_0]$ is the optical variable, τ_0 is the optical thickness, $\phi \in [0, 2\pi]$ is the azimuthal angle, and $\omega \in [0, 1]$ is the single scattering albedo. According to [28], we use the addition theorem to express the phase function in terms of Legendre polynomials:

$$p(\cos\Theta) = \sum_{l=0}^{L} \beta_l P_l(\cos\Theta) \tag{26}$$

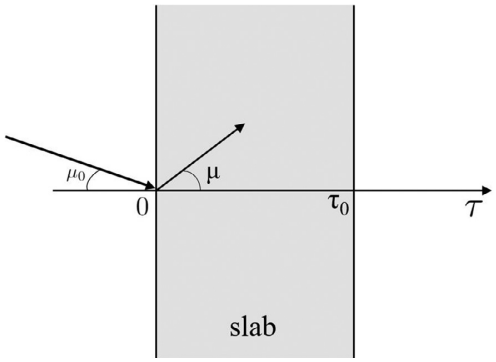


Fig. 2. Geometry of the 1D slab for the considered radiative transfer problem.

where the β_l are the Legendre coefficients in the Lthorder expansion of scattering law.

The problem is subject to the following constraints/boundary conditions:

$$\begin{cases} I(0, \mu, \phi) = S_0 \pi \delta(\mu - \mu_0)\delta(\phi - \phi_0) & \text{for} \quad \mu > 0 \\ I(\tau_0, \mu, \phi) = 0 & \text{for} \quad \mu < 0 \end{cases} \tag{27}$$

where $S_0\pi$ represents the flux carried by a source beam across a plane normal to the direction of incidence, and μ_0 and ϕ_0 determine the inclination of the source beam.

The solution is defined as the sum of collided (or diffuse) term and uncollided term [16]:

$$I(\tau, \mu, \phi) = I^*(\tau, \mu, \phi) + I^o(\tau, \mu, \phi), \tag{28}$$

where the superscript * refers to the collided beam, and the superscript ^{o} refers to the uncollided beam. The problem is therefore split into two parts. The uncollided term solves the following first order PDE:

$$\mu \frac{\partial}{\partial \tau} I^o(\tau, \mu, \phi) + I^o(\tau, \mu, \phi) = 0 \tag{29}$$

subject to:

$$\begin{cases} I^o(0, \mu, \phi) = S_0 \pi \delta(\mu - \mu_0)\delta(\phi - \phi_0) & \text{for} \quad \mu > 0 \\ I^o(0, \mu, \phi) = 0 & \text{for} \quad \mu < 0 \end{cases} \tag{30}$$

where $\mu \in (0, 1]$.

The problem (29) and (30) has the following analytical solutions:

$$\begin{cases} I^o(\tau, \mu, \phi) = S_0 \pi \delta(\mu - \mu_0)\delta(\phi - \phi_0) e^{-\tau/\mu} & \text{for} \quad \mu > 0 \\ I^o(\tau, \mu, \phi) = 0 & \text{for} \quad \mu < 0 \end{cases} \tag{31}$$

Plugging into the Eq. (28), we obtain the following:

$$I(\tau, \mu, \phi) = I^*(\tau, \mu, \phi) + S_0 \pi \delta(\mu - \mu_0)\delta(\phi - \phi_0) e^{-\tau/\mu} \tag{32}$$

The relation of the phase function and the scattering angle can be expressed using the *Addition Theorem of the Spherical Harmonics* [28]:

$$p(\cos\Theta) = \sum_{m=0}^{L}(2-\delta_{0,m})\sum_{l=m}^{L}\beta_l P_l^m(\mu')P_l^m(\mu)\cos[m(\phi'-\phi)] \tag{33}$$

where the following expression is used to denote the normalized function [29,30]:

$$P_l^m(\mu) = \left[\frac{(l-m)!}{(l+m)!}\right](1-\mu^2)^{m/2}\frac{d^m}{d\mu^m}P_l(\mu) \tag{34}$$

Thus, we can express the diffuse component as a Fourier expansion with L terms. That is:

$$I^*(\tau,\mu,\phi) = \frac{1}{2}\sum_{m=0}^{L}(2-\delta_{0,m})I_m(\tau,\mu)\cos[m(\phi-\phi_0)] \tag{35}$$

The $m$thFourier component satisfies the following linear integro - 1storder PDE:

$$\mu\frac{\partial}{\partial t}I_m(\tau,\mu) + I_m(\tau,\mu) = \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(\mu)\int_{-1}^{1}P_l^m(\mu')I_m(\tau,\mu')d\mu' + Q^m(\tau,\mu), \tag{36}$$

where Q is the inhomogeneous source term [16]:

$$Q^m(\tau,\mu) = \frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(\mu) \tag{37}$$

Thus, the problem to solve is the following:

$$\mu\frac{\partial}{\partial t}I_m(\tau,\mu) + I_m(\tau,\mu) = \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(\mu)\int_{-1}^{1}P_l^m(\mu')I_m(\tau,\mu')d\mu' + \frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(\mu) \tag{38}$$

Subject to:

$$\begin{cases} I_m(0,\mu) = 0 & \text{for} \quad \mu > 0 \\ I_m(\tau_0,\mu) = 0 & \text{for} \quad \mu < 0 \end{cases} \tag{39}$$

Via discretizing the angle μ into N points in the interval (0,1] according to the Gauss-Legendre quadrature scheme:

$$\mu \quad \longrightarrow \quad \mu = \{\mu_i\}_{i=1}^{N}$$

it follows that:

$$I_m(\tau,\mu) \quad \longrightarrow \quad \mathbf{I}_m(\tau) = \{I_m(\tau,\mu_i)\}_{i=1}^{N} \tag{40}$$

Following the notation used by Siewert [16], suppressing some of the m indexes, and splitting the problem into two equations, for positive and negative values of μ, we get the following set of equations:

$$\mu_i\frac{\partial}{\partial\tau}I_m(\tau,\mu_i) + I_m(\tau,\mu_i) = \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(\mu_i)\sum_{k=1}^{N}w_k P_l^m(\mu_k)[I_m(\tau,\mu_k) + (-1)^{l-m}I_m(\tau,-\mu_k)] + \frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(\mu_i) \tag{41}$$

$$-\mu_i\frac{\partial}{\partial\tau}I_m(\tau,-\mu_i) + I_m(\tau,-\mu_i) = = \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(-\mu_i)\sum_{k=1}^{N}w_k P_l^m(-\mu_k)[(-1)^{l-m}I_m(\tau,\mu_k) + I_m(\tau,-\mu_k)] + \frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(-\mu_i) \tag{42}$$

where the Gauss-Legendre quadrature rule is used to evaluate the integral in the range [0,1].

Eqs. (41) and (42) with the constraints (39) are a set of N linear first-order ODEs that we will solve via TFC.

4. TFC formulation of the problem

In our case, we assume to have one constraint in one point; hence, according to (1), $y(\tau)$ can be expressed as following:

$$y(\tau) = g(\tau) + \eta p(\tau) \tag{43}$$

We set $p(\tau) = 1$,as suggested by Mortari [21], and the function $g(\tau)$ as an expansion of Chebyshev polynomials as follows:

$$g(\tau) = \boldsymbol{h}^T(\tau)\boldsymbol{\xi} \tag{44}$$

Since Chebyshev Polynomials are used as basis functions, we need to define a new variable x that ranges in $[-1,1]$. The new $x$variable is defined as $x = c\tau - 1$,where c is the mapping coefficient:

$$c = \frac{x_f - x_0}{\tau_f - \tau_0}$$

Considering the new independent variable, we have:

$$\begin{cases} I_m(\tau,\mu) = I_m(x,\mu) \\ \dfrac{\partial}{\partial\tau}I_m(\tau,\mu) = c\dfrac{\partial}{\partial x}I_m(x,\mu) \end{cases} \tag{45}$$

To simplify the notation, for the remainder of this manuscript, we give for granted the x and μ dependency and write the terms for the forward and backward flux as $I_m^+$and I_m^-,respectively. Thus, our problem becomes:

$$c\mu_i\frac{\partial}{\partial x}I_m^+ + I_m^+ = \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(\mu_i)\sum_{k=1}^{N}w_k P_l^m(\mu_k)[I_m^+ + (-1)^{l-m}I_m^-] + \frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(\mu_i) \tag{46}$$

$$-c\mu_i\frac{\partial}{\partial x}I_m^- + I_m^- = \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(-\mu_i)\sum_{k=1}^{N}w_k P_l^m(-\mu_k) \times [(-1)^{l-m}I_m^+ + I_m^-] + \frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(-\mu_i) \tag{47}$$

Subject to:

$$\begin{cases} I_m^+(0) = I_0^+ = 0 \\ I_m^-(\tau_0) = I_f^- = 0 \end{cases} \tag{48}$$

Our constrained expressions, for forward and backward terms, are:

$$I_m^{\pm}(x) = g^{\pm}(x) + \eta^{\pm} \tag{49}$$

Therefore, by using the boundary conditions to calculate $\eta^+$and η^-,we obtain:

$$I_0^+ = 0 = g_0^+ + \eta^+ \quad \longrightarrow \quad \eta^+ = -g_0^+$$

$$I_f^- = 0 = g_f^- + \eta^- \quad \longrightarrow \quad \eta^- = -g_f^-$$

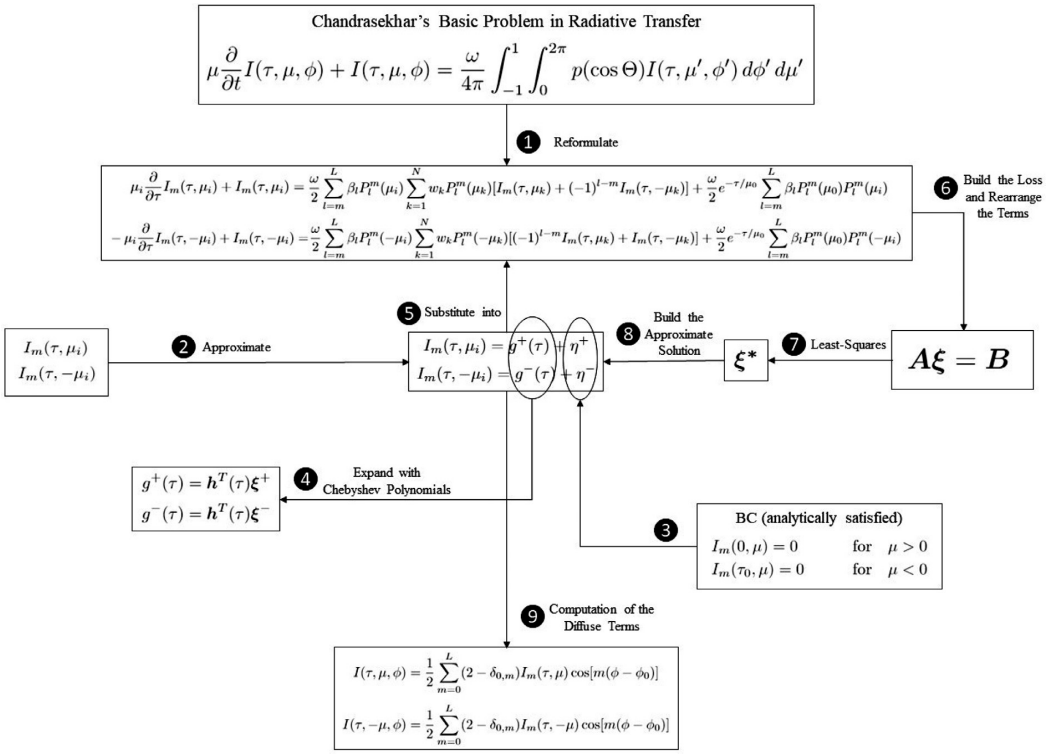


Fig. 3. Schematic of the theory of functional connections algorithm to solve Chandrasekhar's basic problem in radiative transfer.

Table 3
Mie scattering—The Fourier component $I\cdot(\tau,\mu)$ for $m=0$,with a computational time of approximately 0.026 s.

μ	$\tau=0$	$\tau=0.05\tau_0$	$\tau=0.1\tau_0$	$\tau=0.2\tau_0$	$\tau=0.5\tau_0$	$\tau=0.75\tau_0$	$\tau=\tau_0$
−1.0	4.76807e−2	4.41912e−2	4.06467e−2	3.37099e−2	1.58572e−2	5.45297e−3	0
−0.9	6.45644e−2	6.03742e−2	5.60139e−2	4.72899e−2	2.38121e−2	9.01083e−3	0
−0.8	8.45877e−2	7.96763e−2	7.44425e−2	6.37540e−2	3.38287e−2	1.36919e−2	0
−0.7	1.08350e−1	1.02719e−1	9.65678e−2	8.37488e−2	4.64918e−2	1.98845e−2	0
−0.6	1.36504e−1	1.30204e−1	1.23128e−1	1.08063e−1	6.26048e−2	2.81721e−2	0
−0.5	1.69677e−1	1.62852e−1	1.54919e−1	1.37618e−1	8.32921e−2	3.94792e−2	0
−0.4	2.08230e−1	2.01201e−1	1.92626e−1	1.73358e−1	1.10124e−1	5.53631e−2	0
−0.3	2.51677e−1	2.45065e−1	2.36310e−1	2.15806e−1	1.45144e−1	7.86298e−2	0
−0.2	2.97523e−1	2.92401e−1	2.84241e−1	2.63800e−1	1.90035e−1	1.14557e−1	0
−0.1	3.40126e−1	3.38476e−1	3.31984e−1	3.12888e−1	2.41217e−1	1.70697e−1	0
−0.0	3.59379e−1	3.74485e−1	3.73800e−1	3.59347e−1	2.88258e−1	2.25623e−1	0
0.0	0	3.74485e−1	3.73800e−1	3.59347e−1	2.88258e−1	2.25623e−1	1.51520e−1
0.1	0	1.55620e−1	2.51427e−1	3.38864e−1	3.30703e−1	2.69855e−1	2.03073e−1
0.2	0	9.24370e−2	1.65080e−1	2.61887e−1	3.35517e−1	3.01553e−1	2.44242e−1
0.3	0	6.70726e−2	1.24144e−1	2.10383e−1	3.14805e−1	3.09713e−1	2.70302e−1
0.4	0	5.30170e−2	9.98939e−2	1.75304e−1	2.88183e−1	3.02939e−1	2.81066e−1
0.5	0	4.37182e−2	8.32383e−2	1.49251e−1	2.61304e−1	2.88464e−1	2.80822e−1
0.6	0	3.67864e−2	7.05194e−2	1.28323e−1	2.35259e−1	2.69793e−1	2.73029e−1
0.7	0	3.11411e−2	5.99908e−2	1.10387e−1	2.09939e−1	2.48476e−1	2.59822e−1
0.8	0	2.62225e−2	5.07166e−2	9.42001e−2	1.84982e−1	2.25179e−1	2.42453e−1
0.9	0	2.17133e−2	4.21538e−2	7.90039e−2	1.60034e−1	2.00187e−1	2.21667e−1
1.0	0	1.74231e−2	3.39716e−2	6.43193e−2	1.34819e−1	1.73627e−1	1.97932e−1

and then:

$$I_m^+(x)=g^+(x)-g_0^+$$

$$I_m^-(x)=g^-(x)-g_f^-$$

Finally, we get the CEs for the fluxes in the following form:

$$I_m^+=(\boldsymbol{h}^T-\boldsymbol{h}_0^T)\boldsymbol{\xi}^+ \quad ; \quad I_m^-=(\boldsymbol{h}^T-\boldsymbol{h}_f^T)\boldsymbol{\xi}^-, \tag{50}$$

where $\boldsymbol{\xi}^+$and $\boldsymbol{\xi}^-$are the unknowns to be computed via LS. Plugging (50) into our DEs, we get:

$$c\mu_i\boldsymbol{h'}^T\boldsymbol{\xi}_i^+ + (\boldsymbol{h}-\boldsymbol{h_0})^T\boldsymbol{\xi}_i^+ = \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(\mu_i)\sum_{k=1}^{N}w_k P_l^m(\mu_k)[(\boldsymbol{h}-\boldsymbol{h_0})^T\boldsymbol{\xi}_k^+ + (-1)^{l-m}(\boldsymbol{h}-\boldsymbol{h_f})^T\boldsymbol{\xi}_k^-]+\frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(\mu_i) \tag{51}$$

Table 4
Mie scattering—The Fourier component $I_{\cdot}(\tau, \mu)$ for $m = 1$,with a computational time of approximately 0.026 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	0	0	0	0	0	0	0
−0.9	3.60181e−2	3.36746e−2	3.12064e−2	2.62787e−2	1.32903e−2	5.19802e−3	0
−0.8	6.16596e−2	5.78589e−2	5.38160e−2	4.56764e−2	2.38338e−2	9.72915e−3	0
−0.7	9.10359e−2	8.57207e−2	8.00113e−2	6.84223e−2	3.67820e−2	1.56203e−2	0
−0.6	1.26254e−1	1.19295e−1	1.11742e−1	9.62823e−2	5.33398e−2	2.35586e−2	0
−0.5	1.68911e−1	1.60192e−1	1.50609e−1	1.30818e−1	7.48561e−2	3.44878e−2	0
−0.4	2.20425e−1	2.09925e−1	1.98188e−1	1.73691e−1	1.03142e−1	4.99460e−2	0
−0.3	2.81715e−1	2.69635e−1	2.55778e−1	2.26467e−1	1.40633e−1	7.26822e−2	0
−0.2	3.52226e−1	3.39192e−1	3.23503e−1	2.89666e−1	1.89868e−1	1.07900e−1	0
−0.1	4.28566e−1	4.16266e−1	3.99406e−1	3.61374e−1	2.49443e−1	1.63513e−1	0
−0.0	4.90592e−1	4.93468e−1	4.79629e−1	4.40175e−1	3.12405e−1	2.22786e−1	0
0.0	0	4.93468e−1	4.79629e−1	4.40175e−1	3.12405e−1	2.22786e−1	1.40086e−1
0.1	0	2.12101e−1	3.36358e−1	4.36674e−1	3.78485e−1	2.79507e−1	1.92369e−1
0.2	0	1.27652e−1	2.24094e−1	3.44034e−1	3.98038e−1	3.26312e−1	2.41187e−1
0.3	0	9.31423e−2	1.69547e−1	2.78455e−1	3.79153e−1	3.42883e−1	2.74467e−1
0.4	0	7.34351e−2	1.36109e−1	2.31599e−1	3.47468e−1	3.37079e−1	2.87903e−1
0.5	0	5.97940e−2	1.11999e−1	1.94715e−1	3.11216e−1	3.17429e−1	2.84863e−1
0.6	0	4.89801e−2	9.23641e−2	1.62887e−1	2.72157e−1	2.88095e−1	2.68557e−1
0.7	0	3.94407e−2	7.47181e−2	1.33107e−1	2.29842e−1	2.50372e−1	2.40503e−1
0.8	0	3.01812e−2	5.73658e−2	1.02969e−1	1.82413e−1	2.03320e−1	2.00130e−1
0.9	0	2.00568e−2	3.82174e−2	6.90058e−2	1.24817e−1	1.41801e−1	1.42461e−1
1.0	0	0	0	0	0	0	0

Table 5
Mie scattering—The Fourier component $I_{\cdot}(\tau, \mu)$ for $m = 2$,with a computational time of approximately 0.026 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	0	0	0	0	0	0	0
−0.9	6.13369e−3	5.63258e−3	5.14146e−3	4.22728e−3	2.08109e−3	8.51138e−4	0
−0.8	1.40473e−2	1.29199e−2	1.18125e−2	9.74639e−3	4.86758e−3	2.02908e−3	0
−0.7	2.42374e−2	2.23272e−2	2.04470e−2	1.69330e−2	8.59010e−3	3.65704e−3	0
−0.6	3.72977e−2	3.44146e−2	3.15724e−2	2.62520e−2	1.35554e−2	5.91266e−3	0
−0.5	5.39364e−2	4.98590e−2	4.58321e−2	3.82838e−2	2.01851e−2	9.06805e−3	0
−0.4	7.49853e−2	6.94667e−2	6.40030e−2	5.37486e−2	2.90737e−2	1.35709e−2	0
−0.3	1.01366e−1	9.41479e−2	8.69734e−2	7.34927e−2	4.10537e−2	2.02184e−2	0
−0.2	1.33969e−1	1.24806e−1	1.15624e−1	9.83434e−2	5.71259e−2	3.05232e−2	0
−0.1	1.73539e−1	1.62321e−1	1.50803e−1	1.28934e−1	7.75407e−2	4.68764e−2	0
−0.0	2.17610e−1	2.07223e−1	1.93748e−1	1.66780e−1	1.01645e−1	6.54823e−2	0
0.0	0	2.07223e−1	1.93748e−1	1.66780e−1	1.01645e−1	6.54823e−2	3.94802e−2
0.1	0	9.13850e−2	1.41198e−1	1.74338e−1	1.30506e−1	8.62088e−2	5.45248e−2
0.2	0	5.47834e−2	9.40008e−2	1.38386e−1	1.41997e−1	1.05409e−1	7.12370e−2
0.3	0	3.94265e−2	7.02551e−2	1.11034e−1	1.35966e−1	1.12775e−1	8.32078e−2
0.4	0	3.03255e−2	5.50820e−2	9.03843e−2	1.22905e−1	1.10240e−1	8.74206e−2
0.5	0	2.37574e−2	4.36468e−2	7.32857e−2	1.06676e−1	1.01104e−1	8.46547e−2
0.6	0	1.83638e−2	3.39907e−2	5.79582e−2	8.84652e−2	8.72842e−2	7.61525e−2
0.7	0	1.35348e−2	2.51835e−2	4.34139e−2	6.86137e−2	6.97897e−2	6.28579e−2
0.8	0	8.96183e−3	1.67383e−2	2.90913e−2	4.72161e−2	4.91804e−2	4.54269e−2
0.9	0	4.47932e−3	8.39018e−3	1.46739e−2	2.43199e−2	2.58191e−2	2.43409e−2
1.0	0	0	0	0	0	0	0

$$
\begin{aligned}
&-c\mu_i \boldsymbol{h}'^{T}\boldsymbol{\xi}_i^- + (\boldsymbol{h}-\boldsymbol{h_f})^T\boldsymbol{\xi}_i^- \\
&= \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(-\mu_i)\sum_{k=1}^{N} w_k P_l^m(-\mu_k)[(-1)^{l-m}(\boldsymbol{h}-\boldsymbol{h_0})^T\boldsymbol{\xi}_k^+ \\
&+ (\boldsymbol{h}-\boldsymbol{h_f})^T\boldsymbol{\xi}_k^-] + \frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(-\mu_i) \qquad (52)
\end{aligned}
$$

Reorganizing all the terms we get:

$$
\begin{aligned}
&(c\mu_i \boldsymbol{h}' + \boldsymbol{h}-\boldsymbol{h_0})^T\boldsymbol{\xi}_i^+ - \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(\mu_i)\sum_{k=1}^{N} w_k P_l^m(\mu_k) \\
&\times[(\boldsymbol{h}-\boldsymbol{h_0})^T\boldsymbol{\xi}_k^+ + (-1)^{l-m}(\boldsymbol{h}-\boldsymbol{h_f})^T\boldsymbol{\xi}_k^-] \\
&= \frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(\mu_i) \qquad (53)
\end{aligned}
$$

$$
\begin{aligned}
&(-c\mu_i \boldsymbol{h}' + \boldsymbol{h}-\boldsymbol{h_f})^T\boldsymbol{\xi}_i^- - \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(-\mu_i)\sum_{k=1}^{N} w_k P_l^m(-\mu_k) \\
&\times[(-1)^{l-m}(\boldsymbol{h}-\boldsymbol{h_0})^T\boldsymbol{\xi}_k^+ + (\boldsymbol{h}-\boldsymbol{h_f})^T\boldsymbol{\xi}_k^-] \\
&= \frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(-\mu_i) \qquad (54)
\end{aligned}
$$

For the sake of simplicity, we write the inhomogeneous term as:

$$
\begin{aligned}
\boldsymbol{b_i^+} &= \frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(\mu_i) \\
\boldsymbol{b_i^-} &= \frac{\omega}{2}e^{-\tau/\mu_0}\sum_{l=m}^{L}\beta_l P_l^m(\mu_0)P_l^m(-\mu_i)
\end{aligned}
$$

So, the two problems become:

Table 6
Mie scattering—The Fourier component $I_m(\tau, \mu)$ for $m = 3$,with a computational time of approximately 0.026 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	0	0	0	0	0	0	0
−0.9	4.71882e−4	4.27922e−4	3.86504e−4	3.12323e−4	1.49303e−4	6.12974e−5	0
−0.8	1.63700e−3	1.48492e−3	1.34202e−3	1.e-08650e-3	5.24875e−4	2.18940e−4	0
−0.7	3.62794e−3	3.29256e−3	2.97807e−3	2.41643e−3	1.18117e−3	5.01304e−4	0
−0.6	6.65125e−3	6.04087e−3	5.46943e−3	4.44992e−3	2.20550e−3	9.55263e−4	0
−0.5	1.09558e−2	9.96046e−3	9.02968e−3	7.37068e−3	3.71563e−3	1.65092e−3	0
−0.4	1.68361e−2	1.53260e−2	1.39154e−2	1.14039e−2	5.87321e−3	2.70031e−3	0
−0.3	2.46361e−2	2.24603e−2	2.04291e−2	1.68191e−2	8.90196e−3	4.29905e−3	0
−0.2	3.47536e−2	3.17326e−2	2.89122e−2	2.39119e−2	1.30799e−2	6.81521e−3	0
−0.1	4.77051e−2	4.36203e−2	3.97892e−2	3.29968e−2	1.85468e−2	1.08405e−2	0
−0.0	6.40589e−2	5.89247e−2	5.38601e−2	4.47620e−2	2.52999e−2	1.55900e−2	0
0.0	0	5.89247e−2	5.38601e−2	4.47620e−2	2.52999e−2	1.55900e−2	9.40491e−3
0.1	0	2.65947e−2	4.04696e−2	4.85984e−2	3.37870e−2	2.12142e−2	1.30156e−2
0.2	0	1.59125e−2	2.69468e−2	3.87779e−2	3.75052e−2	2.66263e−2	1.73623e−2
0.3	0	1.12859e−2	1.98696e−2	3.07668e−2	3.58014e−2	2.85953e−2	2.04272e−2
0.4	0	8.43804e−3	1.51551e−2	2.44012e−2	3.16829e−2	2.74886e−2	2.11807e−2
0.5	0	6.31959e−3	1.14883e−2	1.89493e−2	2.64242e−2	2.42961e−2	1.98181e−2
0.6	0	4.56717e−3	8.36998e−3	1.40335e−2	2.05701e−2	1.97297e−2	1.67997e−2
0.7	0	3.04358e−3	5.61002e−3	9.51751e−3	1.44725e−2	1.43314e−2	1.26141e−2
0.8	0	1.71669e−3	3.17792e−3	5.43950e−3	8.50729e−3	8.63650e−3	7.80304e−3
0.9	0	6.32858e−4	1.17545e−3	2.02593e−3	3.23965e−3	3.35492e−3	3.09582e−3
1.0	0	0	0	0	0	0	0

Table 7
Mie scattering—The Fourier component $I_m(\tau, \mu)$ for $m = 4$,with a computational time of approximately 0.026 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	0	0	0	0	0	0	0
−0.9	6.69258e−5	6.02239e−5	5.40838e−5	4.33436e−5	2.05304e−5	0.851494e−6	0
−0.8	3.02494e−4	2.72377e−4	2.44794e−4	1.96548e−4	9.39014e−5	3.93973e−5	0
−0.7	7.61720e−4	6.86418e−4	6.17472e−4	4.96877e−4	2.39851e−4	1.02062e−4	0
−0.6	1.50358e−3	1.35622e−3	1.22134e−3	9.85431e−4	4.81766e−4	2.08696e−4	0
−0.5	2.59147e−3	2.34013e−3	2.11014e−3	1.70803e−3	8.48455e−4	3.76292e−4	0
−0.4	4.09363e−3	3.70143e−3	3.34275e−3	2.71612e−3	1.37699e−3	6.30869e−4	0
−0.3	6.08324e−3	5.50815e−3	4.98269e−3	4.06626e−3	2.11599e−3	1.01667e−3	0
−0.2	8.63901e−3	7.83189e−3	7.09526e−3	5.81430e−3	3.12207e−3	1.61559e−3	0
−0.1	1.18659e−2	1.07645e−2	9.75898e−3	8.01357e−3	4.41270e−3	2.55497e−3	0
−0.0	1.59510e−2	1.44948e−2	1.31484e−2	1.08036e−2	5.96834e−3	3.63154e−3	0
0.0	0	1.44948e−2	1.31484e−2	1.08036e−2	5.96834e−3	3.63154e−3	2.19670e−3
0.1	0	6.42599e−3	9.72886e−3	1.15783e−2	7.86933e−3	4.86919e−3	2.96365e−3
0.2	0	3.72830e−3	6.28643e−3	8.98115e−3	8.53292e−3	5.98094e−3	3.86184e−3
0.3	0	2.53962e−3	4.45371e−3	6.85213e−3	7.85349e−3	6.20648e−3	4.39482e−3
0.4	0	1.80267e−3	3.22602e−3	5.16374e−3	6.61455e−3	5.68628e−3	4.34786e−3
0.5	0	1.26267e−3	2.28772e−3	3.75288e−3	5.16865e−3	4.71326e−3	3.81819e−3
0.6	0	8.35644e−4	1.52664e−3	2.54654e−3	3.68968e−3	3.51216e−3	2.97182e−3
0.7	0	4.93531e−4	9.07021e−4	1.53136e−3	2.30332e−3	2.26473e−3	1.98170e−3
0.8	0	2.32528e−4	4.29264e−4	7.31400e−4	1.13210e−3	1.14161e−3	1.02575e−3
0.9	0	6.20029e−5	1.14862e−4	1.97113e−4	3.12097e−4	3.21141e−4	2.94782e−4
1.0	0	0	0	0	0	0	0

$$(c\mu_i \boldsymbol{h}' + \boldsymbol{h} - \boldsymbol{h_0})^T \boldsymbol{\xi}_i^+ - \frac{\omega}{2} \sum_{l=m}^{L} \beta_l P_l^m(\mu_i) \sum_{k=1}^{N} w_k P_l^m(\mu_k)$$
$$[(\boldsymbol{h} - \boldsymbol{h_0})^T \boldsymbol{\xi}_k^+ + (-1)^{l-m} (\boldsymbol{h} - \boldsymbol{h_f})^T \boldsymbol{\xi}_k^-] = \boldsymbol{b_i^+} \tag{55}$$

$$(-c\mu_i \boldsymbol{h}' + \boldsymbol{h} - \boldsymbol{h_f})^T \boldsymbol{\xi}_i^- - \frac{\omega}{2} \sum_{l=m}^{L} \beta_l P_l^m(-\mu_i) \sum_{k=1}^{N} w_k P_l^m(-\mu_k)$$
$$[(-1)^{l-m} (\boldsymbol{h} - \boldsymbol{h_0})^T \boldsymbol{\xi}_k^+ + (\boldsymbol{h} - \boldsymbol{h_f})^T \boldsymbol{\xi}_k^-] = \boldsymbol{b_i^-} \tag{56}$$

Expanding the summations, we get the following matrix form:

$$\begin{bmatrix} \bigstar_i + \blacklozenge_i^k & \clubsuit_i^k \\ \spadesuit_i^k & \blacktriangledown_i + \heartsuit_i^k \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi_i^+} \\ \boldsymbol{\xi_i^-} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b_i^+} \\ \boldsymbol{b_i^-} \end{bmatrix} \tag{57}$$

where:

$$\blacktriangledown_i = (-c\mu_i \boldsymbol{h}' + \boldsymbol{h} - \boldsymbol{h_f})^T$$

$$\bigstar_i = (c\mu_i \boldsymbol{h}' + \boldsymbol{h} - \boldsymbol{h_0})^T$$

$$\clubsuit_i^k = -\frac{\omega}{2} w_k (\boldsymbol{h} - \boldsymbol{h_f})^T \sum_{l=m}^{L} \beta_l P_l^m(\mu_i) P_l(\mu_k) (-1)^{l-m}$$

$$\spadesuit_i^k = -\frac{\omega}{2} w_k (\boldsymbol{h} - \boldsymbol{h_0})^T \sum_{l=m}^{L} \beta_l P_l^m(-\mu_i) P_l(-\mu_k) (-1)^{l-m}$$

$$\blacklozenge_i^k = -\frac{\omega}{2} w_k (\boldsymbol{h} - \boldsymbol{h_0})^T \sum_{l=m}^{L} \beta_l P_l^m(\mu_i) P_l(\mu_k)$$

$$\heartsuit_i^k = -\frac{\omega}{2} w_k (\boldsymbol{h} - \boldsymbol{h_f})^T \sum_{l=m}^{L} \beta_l P_l^m(-\mu_i) P_l(-\mu_k)$$

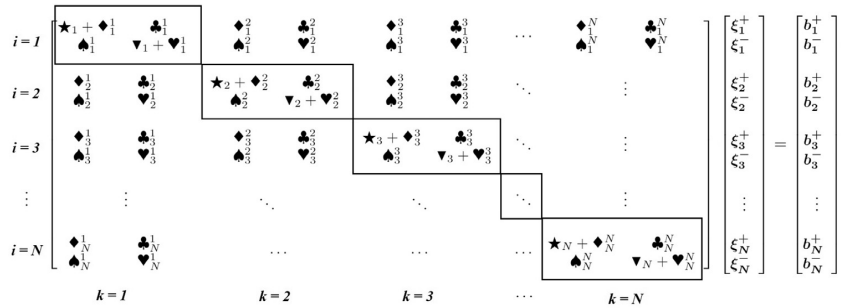
Table 8
Mie scattering—The Fourier component $I_{\cdot}(\tau, \mu)$ for $m = 5$,with a computational time of approximately 0.026 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	0	0	0	0	0	0	0
−0.9	6.28063e−6	5.64038e−6	5.05780e−6	4.04469e−6	1.91071e−6	7.93886e−7	0
−0.8	3.73124e−5	3.35331e−5	3.00940e−5	2.41116e−5	1.14857e−5	4.82442e−6	0
−0.7	1.07458e−4	9.66557e−5	8.68253e−5	6.97210e−5	3.35504e−5	1.42849e−5	0
−0.6	2.29470e−4	2.06608e−4	1.85802e−4	1.49599e−4	7.28951e−5	3.15824e−5	0
−0.5	4.15167e−4	3.74235e−4	3.36993e−4	2.72200e−4	1.34741e−4	5.97452e−5	0
−0.4	6.75372e−4	6.09580e−4	5.49751e−4	4.45739e−4	2.25137e−4	1.03090e−4	0
−0.3	1.01957e−3	9.21504e−4	8.32420e−4	6.77822e−4	3.51320e−4	1.68651e−4	0
−0.2	1.45558e−3	1.31706e−3	1.19142e−3	9.74059e−4	5.20758e−4	2.69137e−4	0
−0.1	1.99273e−3	1.80388e−3	1.63272e−3	1.33734e−3	7.32823e−4	4.23514e−4	0
−0.0	2.65403e−3	2.40328e−3	2.17551e−3	1.78221e−3	9.79104e−4	5.94123e−4	0
0.0	0	2.40328e−3	2.17551e−3	1.78221e−3	9.79104e−4	5.94123e−4	3.60119e−4
0.1	0	1.04360e−3	1.57774e−3	1.87309e−3	1.26582e−3	7.80603e−4	4.74525e−4
0.2	0	5.87087e−4	9.88700e−4	1.40973e−3	1.33332e−3	9.31756e−4	6.00443e−4
0.3	0	3.83898e−4	6.72490e−4	1.03283e−3	1.17920e−3	9.29569e−4	6.57017e−4
0.4	0	2.58469e−4	4.62075e−4	7.38429e−4	9.42637e−4	8.08592e−4	6.17257e−4
0.5	0	1.69091e−4	3.06064e−4	5.01326e−4	6.88265e−4	6.26405e−4	5.06701e−4
0.6	0	1.02291e−4	1.86706e−4	3.10999e−4	4.49276e−4	4.26898e−4	3.60733e−4
0.7	0	5.34205e−5	9.80933e−5	1.65394e−4	2.48078e−4	2.43517e−4	2.12817e−4
0.8	0	2.09675e−5	3.86763e−5	6.58150e−5	1.01604e−4	1.02297e−4	9.18069e−5
0.9	0	4.03113e−6	7.46208e−6	1.27901e−5	2.02003e−5	2.07549e−5	1.90300e−5
1.0	0	0	0	0	0	0	0

Table 9
Mie scattering—-The Fourier component $I_{\cdot}(\tau, \mu)$ for $m = 6$,with a computational time of approximately 0.026 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	0	0	0	0	0	0	0
−0.9	3.64612e−7	3.27338e−7	2.93459e−7	2.34597e−7	1.10774e−7	4.60360e−8	0
−0.8	2.90480e−6	2.60976e−6	2.34156e−6	1.87544e−6	8.92958e−7	3.75132e−7	0
−0.7	9.72738e−6	8.74687e−6	7.85546e−6	6.30582e−6	3.03290e−6	1.29146e−6	0
−0.6	2.27862e−5	2.05098e−5	1.84403e−5	1.48421e−6	7.22835e−6	3.13189e−6	0
−0.5	4.37887e−5	3.94595e−5	3.55246e−5	2.86845e−5	1.41912e−5	6.29250e−6	0
−0.4	7.40943e−5	6.68560e−5	6.02802e−5	4.88581e−5	2.46633e−5	1.12928e−5	0
−0.3	1.14597e−4	1.03542e−4	9.35097e−5	7.61152e−5	3.94263e−5	1.89249e−5	0
−0.2	1.65614e−4	1.49803e−4	1.35478e−4	1.10719e−4	5.91524e−5	3.05665e−5	0
−0.1	2.27191e−4	2.05580e−4	1.86022e−4	1.52303e−4	8.33921e−5	4.81822e−5	0
−0.0	3.00490e−4	2.71920e−4	2.46056e−4	2.01467e−4	1.10580e−4	6.70733e−5	0
0.0	0	2.71920e−4	2.46056e−4	2.01467e−4	1.10580e−4	6.70733e−5	4.06782e−5
0.1	0	1.16112e−4	1.75494e−4	2.08255e−4	1.40600e−4	8.66594e−5	5.26746e−5
0.2	0	6.35970e−5	1.07078e−4	1.52621e−4	1.44236e−4	1.00747e−4	6.49064e−5
0.3	0	4.00550e−5	7.01511e−5	1.07705e−4	1.22887e−4	9.68330e−5	6.84233e−5
0.4	0	2.56452e−5	4.58379e−5	7.32305e−5	9.34258e−5	8.01121e−5	6.11407e−5
0.5	0	1.56977e−5	2.84085e−5	4.65197e−5	6.38310e−5	5.80755e−5	4.69672e−5
0.6	0	8.68979e−6	1.58582e−5	2.64083e−5	3.81303e−5	3.62204e−5	3.06005e−5
0.7	0	4.01435e−6	7.37013e−6	1.24236e−5	1.86253e−5	1.82779e−5	1.59706e−5
0.8	0	1.31217e−6	2.42003e−6	4.11715e−6	6.35302e−6	6.39473e−6	5.73797e−6
0.9	0	1.81710e−7	3.36316e−7	5.76320e−7	9.09821e−7	9.34571e−7	8.56760e−7
1.0	0	0	0	0	0	0	0

Thus, the problem reduce to the following linear system of algebraic equations:



which is a problem of the type:

$$\boldsymbol{A}\boldsymbol{\xi} = \boldsymbol{B} \tag{58}$$

Table 10
Mie scattering—The Fourier component $I_{\cdot}(\tau, \mu)$ for $m = 7$,with a computational time of approximately 0.026 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	0	0	0	0	0	0	0
−0.9	1.27031e−8	1.14041e−8	1.02235e−8	8.17255e−9	3.85880e−9	1.60370e−9	0
−0.8	1.42973e−7	1.28447e−7	1.15244e−7	9.22992e−8	4.39445e−8	1.84614e−8	0
−0.7	5.79668e−7	5.21220e−7	4.68089e−7	3.75735e−7	1.80707e−7	7.69491e−8	0
−0.6	1.53649e−6	1.38294e−6	1.24336e−6	1.00071e−6	4.87335e−7	2.11154e−7	0
−0.5	3.21082e−6	2.89328e−6	2.60469e−6	2.10309e−6	1.04041e−6	4.61327e−7	0
−0.4	5.75018e−6	5.18825e−6	4.67782e−6	3.79130e−6	1.91370e−6	8.76244e−7	0
−0.3	9.22443e−6	8.33427e−6	7.52656e−6	6.12622e−6	3.17306e−6	1.52308e−6	0
−0.2	1.36060e−5	1.23065e−5	1.11294e−5	9.09502e−6	4.85872e−6	2.51067e−6	0
−0.1	1.87914e−5	1.70031e−5	1.53850e−5	1.25956e−5	6.89601e−6	3.98428e−6	0
−0.0	2.47201e−5	2.23679e−5	2.02394e−5	1.65707e−5	9.09428e−6	5.51598e−6	0
0.0	0	2.23679e−5	2.02394e−5	1.65707e−5	9.09428e−6	5.51598e−6	3.34558e−6
0.1	0	9.38991e−6	1.41917e−5	1.68401e−5	1.13680e−5	7.00631e−6	4.25867e−6
0.2	0	4.99787e−6	8.41462e−6	1.19931e−5	1.13331e−5	7.91561e−6	5.09952e−6
0.3	0	3.02141e−6	5.29147e−6	8.12382e−6	9.26819e−6	7.30285e−6	5.16014e−6
0.4	0	1.83088e−6	3.27241e−6	5.22780e−6	6.66903e−6	5.71842e−6	4.36413e−6
0.5	0	1.04255e−6	1.88668e−6	3.08937e−6	4.23874e−6	3.85640e−6	3.11870e−6
0.6	0	5.24606e−7	9.57346e−7	1.59419e−6	2.30167e−6	2.18631e−6	1.84705e−6
0.7	0	2.12804e−7	3.90688e−7	6.58548e−7	9.87231e−7	9.68785e−7	8.46476e−7
0.8	0	5.74705e−8	1.05990e−7	1.80314e−7	2.78220e−7	2.80038e−7	2.51273e−7
0.9	0	5.68457e−9	1.05210e−8	1.80285e−8	2.84596e−8	2.92330e−8	2.67986e−8
1.0	0	0	0	0	0	0	0

Table 11
Mie scattering—The Fourier component $I_{\cdot}(\tau, \mu)$ for $m = 8$,with a computational time of approximately 0.026 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	0	0	0	0	0	0	0
−0.9	6.56254e−10	5.89143e−10	5.28152e−10	4.22197e−10	1.99346e−10	8.28473e−11	0
−0.8	9.16369e−9	8.23263e−9	7.38637e−9	5.91575e−9	2.81653e−9	1.18325e−9	0
−0.7	4.02493e−8	3.61908e−8	3.25016e−8	2.60890e−8	1.25472e−8	5.34291e−9	0
−0.6	1.09663e−7	9.87034e−8	8.87413e−8	7.14229e−8	3.47818e−8	1.50704e−8	0
−0.5	2.29190e−7	2.06523e−7	1.85923e−7	1.50118e−7	7.42637e−8	3.29292e−8	0
−0.4	4.03647e−7	3.64199e−7	3.28368e−7	2.66136e−7	1.34335e−7	6.15090e−8	0
−0.3	6.29436e−7	5.68693e−7	5.13578e−7	4.18023e−7	2.16513e−7	1.03927e−7	0
−0.2	8.94476e−7	8.09046e−7	7.31658e−7	5.97914e−7	3.19414e−7	1.65053e−7	0
−0.1	1.18131e−6	1.06889e−6	9.67164e−7	7.91809e−7	4.33507e−7	2.50465e−7	0
−0.0	1.47573e−6	1.33530e−6	1.20823e−6	9.89214e−7	5.42893e−7	3.29281e−7	0
0.0	0	1.33530e−6	1.20823e−6	9.89214e−7	5.42893e−7	3.29281e−7	1.99719e−7
0.1	0	5.28593e−7	7.98899e−7	9.47982e−7	6.39936e−7	3.94403e−7	2.39731e−7
0.2	0	2.63292e−7	4.43289e−7	6.31803e−7	5.97029e−7	4.16994e−7	2.68642e−7
0.3	0	1.47638e−7	2.58560e−7	3.96958e−7	4.52873e−7	3.56839e−7	2.52139e−7
0.4	0	8.20707e−8	1.46688e−7	2.34338e−7	2.9894e−7	2.56328e−7	1.95622e−7
0.5	0	4.22497e−8	7.64582e−8	1.25197e−7	1.71775e−7	1.56280e−7	1.26385e−7
0.6	0	1.88253e−8	3.43539e−8	5.72066e−8	8.25936e−8	7.84537e−8	6.62795e−8
0.7	0	6.54561e−9	1.20171e−8	2.02561e−8	3.03658e−8	2.97984e−8	2.60363e−8
0.8	0	1.42836e−9	2.63425e−9	4.48146e−9	6.91474e−9	6.95991e−9	6.24498e−9
0.9	0	9.88569e−11	1.82964e−10	3.13522e−10	4.94919e−10	5.08367e−10	4.66032e−10
1.0	0	0	0	0	0	0	0

whose the dimensions are:

$\boldsymbol{\xi}^{\pm}_{\boldsymbol{i}\,[m_{op}\times 1]}$

$\boldsymbol{\xi}_{[2m_{op}N\times 1]}$

$\boldsymbol{b}^{\pm}_{\boldsymbol{i}\,[M\times 1]}$

$\boldsymbol{B}_{[2MN\times 1]}$

$\bigstar_{\boldsymbol{i}\,[M\times m_{op}]}$
(same dimension for the other terms of the matrix $\boldsymbol{A}$)

$\boldsymbol{A}_{[(2MN)\times(2m_{op}N)]}$

where M is the number of discretization points of τ, and m_{op} is the number of basis functions used.

That is, the original Linear BVP has been reformulated as an *unconstrained* optimization problem. In this case, $\boldsymbol{\xi}$ is calculated with only one direct pass via LS methods, as the original problem is linear [21]. That is:

$$\boldsymbol{\xi} = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{B} \tag{59}$$

As can be seen from (58) the unknowns are computed using a classic LS method. However, the authors have tried all the LS methods suggested in Appendix A of reference [22], find the classic LS method the most convenient in terms of accuracy and computational time.

Once $\boldsymbol{\xi}$ are computed, the final solutions are built according to (50).

After finding the solutions for all m^{th} Fourier components (with $m = 0, 1, \ldots, L$), we can compute the diffuse terms expressed as:

$$I(\tau, \mu, \phi) = \frac{1}{2}\sum_{m=0}^{L}(2 - \delta_{0,m})I_m(\tau, \mu)\cos[m(\phi - \phi_0)] \tag{60}$$

Table 12
Mie scattering—The Fourier component $I_m(\tau, \mu)$ for $m = 0$.according to Ganapol [32], with a computational time of approximately 2.6 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	4.7680739e−2	4.4191232e−2	4.0646720e−2	3.3709854e−2	1.5857241e−2	5.4529708e−3	0
−0.9	6.4564440e−2	6.0374289e−2	5.6013853e−2	4.7289947e−2	2.3812095e−2	9.0108349e−3	0
−0.8	8.4587655e−2	7.9676269e−2	7.4442518e−2	6.3753956e−2	3.3828695e−2	1.3691858e−2	0
−0.7	1.0834976e−1	1.0271869e−1	9.6567817e−2	8.3748795e−2	4.6491789e−2	1.9884482e−2	0
−0.6	1.3650449e−1	1.3020361e−1	1.2312760e−1	1.0806281e−1	6.2604846e−2	2.8172141e−2	0
−0.5	1.6967721e−1	1.6285165e−1	1.5491892e−1	1.3761804e−1	8.3292056e−2	3.9479159e−2	0
−0.4	2.0822991e−1	2.0120060e−1	1.9262576e−1	1.7335796e−1	1.1012425e−1	5.5363056e−2	0
−0.3	2.5167744e−1	2.4506518e−1	2.3630972e−1	2.1580644e−1	1.4514429e−1	7.8629773e−2	0
−0.2	2.9752321e−1	2.9240107e−1	2.8424081e−1	2.6379979e−1	1.9003527e−1	1.1455737e−1	0
−0.1	3.4012557e−1	3.3847566e−1	3.3198375e−1	3.1288756e−1	2.4121715e−1	1.7069703e−1	0
−0.0	3.5937904e−1	3.7448490e−1	3.7380021e−1	3.5934745e−1	2.8825763e−1	2.2562269e−1	0
0.0	0	3.7448490e−1	3.7380021e−1	3.5934745e−1	2.8825763e−1	2.2562269e−1	1.5152044e−1
0.1	0	1.5561969e−1	2.5142670e−1	3.3886425e−1	3.3070258e−1	2.6985545e−1	2.0307340e−1
0.2	0	9.2437025e−2	1.6508021e−1	2.6188673e−1	3.3551731e−1	3.0155334e−1	2.4424153e−1
0.3	0	6.7072588e−2	1.2414425e−1	2.1038294e−1	3.1480528e−1	3.0971262e−1	2.7030221e−1
0.4	0	5.3016958e−2	9.9893927e−2	1.7530448e−1	2.8818268e−1	3.0293863e−1	2.8106600e−1
0.5	0	4.3718197e−2	8.3238337e−2	1.4925136e−1	2.6130432e−1	2.8846360e−1	2.8082243e−1
0.6	0	3.6786430e−2	7.0519358e−2	1.2832314e−1	2.3525903e−1	2.6979273e−1	2.7302917e−1
0.7	0	3.1141106e−2	5.9990823e−2	1.1038662e−1	2.0993897e−1	2.4847557e−1	2.5982225e−1
0.8	0	2.6222496e−2	5.0716634e−2	9.4200130e−2	1.8498212e−1	2.2517903e−1	2.4245347e−1
0.9	0	2.1713290e−2	4.2153804e−2	7.9003938e−2	1.6003444e−1	2.0018674e−1	2.2166696e−1
1.0	0	1.7423141e−2	3.3971636e−2	6.4319263e−2	1.3481854e−1	1.7362739e−1	1.9793246e−1

Table 13
Mie scattering—The Fourier component $I_m(\tau, \mu)$ for $m = 8$,according to Ganapol [32], with a computational time of approximately 0.026 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	0	0	0	0	0	0	0
−0.9	6.5625391e−10	5.8914304e−10	5.2815237e−10	4.2219714e−10	1.9934608e−10	8.2847308e−11	0
−0.8	9.1636878e−9	8.2326319e−9	7.3863711e−9	5.9157513e−9	2.8165373e−9	1.1832473e−9	0
−0.7	4.0249326e−8	3.6190815e−8	3.2501619e−8	2.6088956e−8	1.2547246e−8	5.3429052e−9	0
−0.6	1.0966293e−7	9.8703386e−8	8.8741262e−8	7.1422850e−8	3.4781849e−8	1.5070373e−8	0
−0.5	2.2918971e−7	2.0652282e−7	1.8592286e−7	1.5011771e−7	7.4263736e−8	3.2929240e−8	0
−0.4	4.0364654e−7	3.6419928e−7	3.2836806e−7	2.6613595e−7	1.3433461e−7	6.1509004e−8	0
−0.3	6.2943646e−7	5.6869337e−7	5.1357757e−7	4.1802334e−7	2.1651285e−7	1.0392686e−7	0
−0.2	8.9447618e−7	8.0904612e−7	7.3165782e−7	5.9791370e−7	3.1941422e−7	1.6505266e−7	0
−0.1	1.1813127e−6	1.0688908e−6	9.6716367e−7	7.9180936e−7	4.3350657e−7	2.5046511e−7	0
−0.0	1.4757328e−6	1.3352990e−6	1.2082288e−6	9.8921442e−7	5.4289266e−7	3.2928109e−7	0
0.0	0	1.3352990e−6	1.2082288e−6	9.8921442e−7	5.4289266e−7	3.2928109e−7	1.9971892e−7
0.1	0	5.2859344e−7	7.9889942e−7	9.4798249e−7	6.3993602e−7	3.9440254e−7	2.3973118e−7
0.2	0	2.6329242e−7	4.4328927e−7	6.3180326e−7	5.9702947e−7	4.1699385e−7	2.6864210e−7
0.3	0	1.4763750e−7	2.5856043e−7	3.9695812e−7	4.5287275e−7	3.5683886e−7	2.5213926e−7
0.4	0	8.2070715e−8	1.4668783e−7	2.3433850e−7	2.9894033e−7	2.5632829e−7	1.9562189e−7
0.5	0	4.2249686e−8	7.6458207e−8	1.2519740e−7	1.7177455e−7	1.5627982e−7	1.2638468e−7
0.6	0	1.8825271e−8	3.4353900e−8	5.7206557e−8	8.2593552e−8	7.8453748e−8	6.6279496e−8
0.7	0	6.5456093e−9	1.2017087e−8	2.0256111e−8	3.0365796e−8	2.9798350e−8	2.6036280e−8
0.8	0	1.4283575e−9	2.6342575e−9	4.4814591e−9	6.9147374e−9	6.9599115e−9	6.2449834e−9
0.9	0	9.8856917e−11	1.8296411e−10	3.1352189e−10	4.9491896e−10	5.0836713e−10	4.6603207e−10
1.0	0	0	0	0	0	0	0

$$I(\tau, -\mu, \phi) = \frac{1}{2}\sum_{m=0}^{L}(2-\delta_{0,m})I_m(\tau, -\mu)\cos[m(\phi-\phi_0)] \tag{61}$$

For the convenience of the reader, the all process described in this section is summarized in the schematic of Fig. 3.

5. Post-processing

Via the discretization of the cosine of the polar angle μ, the solutions obtained are for specific angles μ that are the Gauss–Legendre quadrature nodes. However, in many applications, it is necessary to compute the solutions for any arbitrary angles, that can be different than the Gauss–Legendre quadrature nodes. To compute the solutions at any arbitrary query points μ a post–processing is needed. The post–processing is also performed via TFC. Introducing some changes to the Eqs. (55) and (56), we get:

$$(c\gamma_j\boldsymbol{h}' + \boldsymbol{h} - \boldsymbol{h_0})^T\boldsymbol{\zeta}_j^+ = \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(\gamma_j)\sum_{k=1}^{N}w_k P_l^m(\mu_k)[(\boldsymbol{h}-\boldsymbol{h_0})^T\boldsymbol{\xi}_k^+ + (-1)^{l-m}(\boldsymbol{h}-\boldsymbol{h_f})^T\boldsymbol{\xi}_k^-] + \boldsymbol{b_i^+} \tag{62}$$

$$(-c\gamma_j\boldsymbol{h}' + \boldsymbol{h} - \boldsymbol{h_f})^T\boldsymbol{\zeta}_j^- = \frac{\omega}{2}\sum_{l=m}^{L}\beta_l P_l^m(-\gamma_j) \times \sum_{k=1}^{N}w_k P_l^m(-\mu_k)[(-1)^{l-m}(\boldsymbol{h}-\boldsymbol{h_0})^T\boldsymbol{\xi}_k^+ + (\boldsymbol{h}-\boldsymbol{h_f})^T\boldsymbol{\xi}_k^-] + \boldsymbol{b_i^-} \tag{63}$$

where γ_j are the query angles, and $\boldsymbol{\zeta_j^{\pm}}$are the vectors of the new unknowns corresponding to the angle γ_j, and the right–hand sides of the equations are known. Hence, we obtain the following linear

Table 14
Haze L problem—The intensity $I_\cdot(\tau, \mu)$ for the Haze L phase function with $\omega = 0.9$and $\mu_0 = 1$,with a computational time of approximately 1.6 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	2.79717e−2	2.65834e−2	2.51795e−2	2.23421e−2	1.37519e−2	6.70439e−3	0
−0.9	3.01802e−2	2.87427e−2	2.72763e−2	2.42824e−2	1.50370e−2	7.32793e−3	0
−0.8	3.14478e−2	3.00707e−2	2.86411e−2	2.56621e−2	1.60962e−2	7.85504e−3	0
−0.7	3.43839e−2	3.30557e−2	3.16407e−2	2.86060e−2	1.83142e−2	9.00263e−3	0
−0.6	3.91308e−2	3.78910e−2	3.65135e−2	3.34278e−2	2.20977e−2	1.10619e−2	0
−0.5	4.56379e−2	4.46171e−2	4.33840e−2	4.04032e−2	2.80086e−2	1.45143e−2	0
−0.4	5.35113e−2	5.29854-2	5.21410e−2	4.96862e−2	3.68540e−2	2.02308e−2	0
−0.3	6.15420e−2	6.19914e−2	6.19786e−2	6.08863e−2	4.96165e−2	2.98277e−2	0
−0.2	6.69562e−2	6.90564e−2	7.04996e−2	7.20372e−2	6.66970e−2	4.63006e−2	0
−0.1	6.55296e−2	7.00041e−2	7.34324e−2	7.85904e−2	8.44628e−2	7.36110e−2	0
−0.0	5.17485e−2	6.17096e−2	6.80163e−2	7.74665e−2	9.39979e−2	9.74837e−2	0
0.0	0	6.17096e−2	6.80163e−2	7.74665e−2	9.39978e−2	9.74837e−2	7.93126e−2
0.1	0	2.24949e−2	3.95181e−2	6.26530e−2	9.62543e−2	1.08718e−1	1.08189e−1
0.2	0	1.31007e−2	2.53401e−2	4.67729e−2	9.15916e−2	1.13815e−1	1.24212e−1
0.3	0	1.01943e−2	2.02703e−2	3.94722e−2	8.74675e−2	1.16620e−1	1.35712e−1
0.4	0	9.52906e−3	1.90677e−2	3.77703e−2	8.83323e−2	1.22503e−1	1.48268e−1
0.5	0	1.02637e−2	2.05023e−2	4.06492e−2	9.64183e−2	1.35817e−1	1.67516e−1
0.6	0	1.25293e−2	2.49095e−2	4.90656e−2	1.15336e−1	1.62230e−1	2.00701e−1
0.7	0	1.74171e−2	3.44152e−2	6.70811e−2	1.53982e−1	2.13563e−1	2.61672e−1
0.8	0	2.85622e−2	5.60204e−2	1.07697e−2	2.38486e−1	3.22550e−1	3.86921e−1
0.9	0	6.16331e−2	1.19763e−1	2.26124e−1	4.76103e−1	6.19703e−1	7.17745e−1
1.0	0	3.28124e−1	6.29065e−1	1.15632	2.24839	2.74147	2.97766

Table 15
Haze L problem—The intensity $I_\cdot(\tau, \mu)$ for the Haze L phase function with $\omega = 1$and $\mu_0 = 1$,with a computational time of approximately 1.6 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	3.61452e−2	3.43394e−2	3.25109e−2	2.88122e−2	1.76286e−2	8.52589e−3	0
−0.9	3.97819e−2	3.78723e−2	3.59207e−2	3.19303e−2	1.96202e−2	9.45731e−3	0
−0.8	4.27313e−2	4.08406e−2	3.88734e−2	3.47677e−2	2.16019e−2	1.03959e−2	0
−0.7	4.80051e−2	4.61319e−2	4.41307e−2	3.98292e−2	2.52479e−2	1.22171e−2	0
−0.6	5.58214e−2	5.40432e−2	5.20594e−2	4.75986e−2	3.11837e−2	1.53618e−2	0
−0.5	6.60942e−2	6.46296e−2	6.28449e−2	5.84971e−2	4.02740e−2	2.05621e−2	0
−0.4	7.81481e−2	7.74403e−2	7.62508e−2	7.27049e−2	5.37300e−2	2.91285e−2	0
−0.3	8.99682e−2	9.07706e−2	9.08784e−2	8.94711e−2	7.29643e−2	4.34688e−2	0
−0.2	9.70815e−2	1.00421e−1	1.02789e−1	1.05506e−1	9.83777e−2	6.79949e−2	0
−0.1	9.29328e−2	9.98187e−2	1.05195e−1	1.13497e−1	1.24037e−1	1.08399e−2	0
−0.0	6.98774e−2	8.46673e−2	9.41663e−1	1.08727e−1	1.35762e−1	1.42779e−1	0
0.0	0	8.46673e−2	9.41663e−1	1.08727e−1	1.35762e−1	1.42779e−1	1.14808e−1
0.1	0	2.95418e−2	5.24346e−2	8.45649e−2	1.35096e−1	1.56106e−1	1.56976-1
0.2	0	1.64907e−2	3.22817e−2	6.07527e−2	1.24350e−1	1.58925e−1	1.76818e−1
0.3	0	1.23421e−2	2.48488e−2	4.93968e−2	1.14811e−1	1.57937e−1	1.88301e−1
0.4	0	1.11879e−2	2.26450e−2	4.57547e−2	1.12269e−1	1.60862e−1	2.00019e−1
0.5	0	1.17959e−2	2.37910e−2	4.80003e−2	1.19079e−1	1.73191e−1	2.19633e−1
0.6	0	1.42049e−2	2.84584e−2	5.68731e−2	1.39051e−1	2.01445e−1	2.55983e−1
0.7	0	1.95833e−2	3.89248e−2	7.67454e−2	1.82004e−1	2.58986e−1	3.25125e−1
0.8	0	3.19532e−2	6.29430e−2	1.22045e−1	2.77182e−1	3.82767e−1	4.68658e−1
0.9	0	6.87267e−2	1.33917e−1	2.54259e−1	5.44601e−1	7.19447e−1	8.46084e−1
1.0	0	3.64940e−1	7.00266e−1	1.28955	2.52255	3.09319	3.38091

system:

$$\left(c\gamma_j \boldsymbol{h}' + \boldsymbol{h} - \boldsymbol{h_0}\right)^T \boldsymbol{\zeta}_j^+ = \frac{\omega}{2} \sum_{l=m}^{L} \beta_l P_l^m(\gamma_j) \sum_{k=1}^{N} w_k P_l^m(\mu_k) \times \left[(\boldsymbol{h} - \boldsymbol{h_0})^T \boldsymbol{\xi}_k^+ + (-1)^{l-m} (\boldsymbol{h} - \boldsymbol{h_f})^T \boldsymbol{\xi}_k^-\right] + \boldsymbol{b_i^+} \tag{64}$$

$$\left(-c\gamma_j \boldsymbol{h}' + \boldsymbol{h} - \boldsymbol{h_f}\right)^T \boldsymbol{\zeta}_j^- = \frac{\omega}{2} \sum_{l=m}^{L} \beta_l P_l^m(\gamma_j) \sum_{k=1}^{N} w_k P_l^m(-\mu_k) \times \left[(-1)^{l-m} (\boldsymbol{h} - \boldsymbol{h_0})^T \boldsymbol{\xi}_k^+ + (\boldsymbol{h} - \boldsymbol{h_f})^T \boldsymbol{\xi}_k^-\right] + \boldsymbol{b_i^-} \tag{65}$$

This is solved for $\boldsymbol{\zeta}_j^{\pm}$via LS. Once $\boldsymbol{\zeta}_j^{\pm}$are computed, the final solutions are again built according to (50).

After finding the solutions for all m^{th} Fourier components (with $m = 0, 1, \ldots, L$) we can compute the diffuse terms as follows:

$$I^+(\tau, \gamma, \phi) = \frac{1}{2} \sum_{m=0}^{L} (2 - \delta_{0,m}) I_m^+(\tau, \gamma) \cos[m(\phi - \phi_0)] \tag{66}$$

$$I^-(\tau, -\gamma, \phi) = \frac{1}{2} \sum_{m=0}^{L} (2 - \delta_{0,m}) I_m^-(\tau, -\gamma) \cos[m(\phi - \phi_0)] \tag{67}$$

6. Results and discussions

The algorithm described above has been implemented in both Matlab and Python platforms. To validate the accuracy of our framework, we compared our results with some of typical benchmarks for the Radiative Transfer problems, such as Mie Scattering and Haze L Problem. We ran all our simulations with an Intel Core i7 - 9700 CPU PC with 64 GB of RAM. The results reported below

Table 16
Haze L problem—The intensity $I(\tau, \mu, \phi)$ for the Haze L phase function with $\omega = 0.9$, $\mu_0 = 0.5$,and $\phi - \phi_0 = 0$,with a computational time of approximately 132 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	2.28190e−2	2.14170e−2	1.99920e−2	1.71574e−2	9.34719e−3	4.02513e−3	0
−0.9	4.11125e−2	3.86276e−2	3.60559e−2	3.08703e−2	1.64542e−2	6.81049e−3	0
−0.8	6.49983e−2	6.12972e−2	5.73958e−2	4.94000e−2	2.66078e−2	1.10213e−2	0
−0.7	9.99446e−2	9.47422e−2	8.91210e−2	7.73509e−2	4.26132e−2	1.79328e−2	0
−0.6	1.50993e−1	1.44083e−1	1.36343e−1	1.19657e−1	6.81169e−2	2.95065e−2	0
−0.5	2.24768e−1	2.16263e−1	2.06169e−1	1.83479e−1	1.09068e−1	4.93125e−2	0
−0.4	3.29336e−1	3.20182e−1	3.08045e−1	2.78889e−1	1.75210e−1	8.41056e−2	0
−0.3	4.72536e−1	4.65520e−1	4.52954e−1	4.18719e−1	2.82082e−1	1.47239e−1	0
−0.2	6.56834e−1	6.58390e−1	6.49495e−1	6.15206e−1	4.51466e−1	2.66183e−1	0
−0.1	8.70325e−1	8.94525e-1	8.97446e−1	8.72172e−1	6.97514e−1	4.91555e−1	0
−0.0	1.03177	1.14775	1.18797	1.19440	1.00873	7.95748e−1	0
0.0	0	1.14775	1.18797	1.19440	1.00873	7.95748e−1	5.24167e−1
0.1	0	6.07548e−1	9.98592e−1	1.38473	1.41181	1.15806	8.76470e−1
0.2	0	5.08883e−1	9.07223e−1	1.43790	1.82630	1.61273	1.29144
0.3	0	5.49994e−1	9.99898e−1	1.64825	2.30787	2.15587	1.80549
0.4	0	6.38615e−1	1.16925	1.95778	2.87301	2.77822	2.40104
0.5	0	6.34271e−1	1.16869	1.98321	3.03333	3.03426	2.71055
0.6	0	4.18260e−1	7.78087e−1	1.34620	2.18395	2.29506	2.15369
0.7	0	2.04883e−1	3.85810e−1	6.83932e−1	1.19334	1.33200	1.32682
0.8	0	8.64752e−2	1.65150e−1	3.01013e−1	5.69583e−1	6.78608e−1	7.19934e−1
0.9	0	3.13664e−2	6.09010e−2	1.14612e−1	2.37335e−1	3.03207e−1	3.43478e−1
1.0	0	5.07113e−3	1.01191e−2	2.00435e−2	4.76083e−2	6.73492e−2	8.37579e−2

Table 17
Haze L problem—The intensity $I(\tau, \mu, \phi)$ for the Haze L phase function with $\omega = 0.9$, $\mu_0 = 0.5$,and $\phi - \phi_0 = \pi/2$,with a computational time of approximately 132 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	2.28190e−2	2.14170e−2	1.99920e−2	1.71574e−2	9.34719e−3	4.02513e−3	0
−0.9	2.69861e−2	2.54001e−2	2.3770e−2	2.04885e−2	1.12507e−2	4.83998e−3	0
−0.8	3.23251e−2	3.05433e−2	2.86841e−2	2.48816e−2	1.38576e−2	5.98687e−3	0
−0.7	3.90915e−2	3.71288e−2	3.50364e−2	3.06624e−2	1.74617e−2	7.63435e−3	0
−0.6	4.75194e−2	4.54446e−2	4.31587e−2	3.82274e−2	2.24929e−2	1.00585e−2	0
−0.5	5.76960e−2	5.56800e−2	5.33274e−2	4.79966e−2	2.95696e−2	1.37243e−2	0
−0.4	6.92921e−2	6.76843e−2	6.55506e−2	6.02592e−2	3.95485e−2	1.94423e−2	0
−0.3	8.09723e−2	8.04082e−2	7.90373e−2	7.47154e−2	5.34553e−2	2.86762e−2	0
−0.2	8.94088e−2	9.08993e−2	9.11597e−2	8.93864e−2	7.18225e−2	4.41114e−2	0
−0.1	8.86327e−2	9.36078e−2	9.65669e−2	9.91642e−2	9.15413e−2	6.94491e−2	0
−0.0	6.76014e−2	8.16018e−2	8.92220e−2	9.83762e−2	1.03484e−1	9.32369e−2	0
0.0	0	8.16018e−2	8.92220e−2	9.83762e−2	1.03484e−1	9.32371e−2	6.29164e−2
0.1	0	2.74475e−2	4.83619e−2	7.57571e−2	1.04622e−1	1.04387e−1	8.95907e−2
0.2	0	1.41330e−2	2.75945e−2	5.09061e−2	9.28868e−2	1.04678e−1	1.01178e−1
0.3	0	9.26644e−3	1.87294e−2	3.68737e−2	7.85470e−2	9.74004e−2	1.02990e−1
0.4	0	6.96644e−3	1.42411e−2	2.88244e−2	6.68034e−2	8.82947e−2	9.95180e−2
0.5	0	5.75720e−3	1.17847e−2	2.40718e−2	5.82338e−2	8.01154e−2	9.43192e−2
0.6	0	5.11030e−3	1.04251e−2	2.12819e−2	5.23831e−2	7.37544e−2	8.93298e−2
0.7	0	4.79703e−3	9.73440e−3	1.97631e−2	4.87196e−2	6.93677e−2	8.54626e−2
0.8	0	4.71115e−3	9.50506e−3	1.91501e−2	4.68396e−2	6.68825e−2	8.31200e−2
0.9	0	4.80640e−3	9.64244e−3	1.92635e−2	4.64998e−2	6.62130e−2	8.24990e−2
1.0	0	5.07113e−3	1.01191e−2	2.00435e−2	4.76083e−2	6.73492e−2	8.37579e−2

are obtained via Matlab, although the ones computed with Python achieved the same order of accuracy.

The first benchmark of interest is the Radiative Transfer problem with scattering modeled via the Mie Scattering theory for spherical particles. This problem is defined by a $L = 8$phase function. The results obtained using the TFC approach are compared with the benchmarks reported by Garcia and Siewert [31]. The latter employed a discretization of $N = 20 - 50\mu$ directions. Our results are reported in Tables 3–11, where thee following the following parameters have been employed: $N = 15$, $M = 100$, $m_{op} = 40$, $\tau_0 = 1$, $S_0 = 1$, $\mu_0 = 0.5$, $\omega = 0.95$. The TFC algorithm achieved the reported results with a computational time of 0.026 s. Importantly, the TFC method achieved exactly the same results as reported by Garcia and Siewert [31].

Furthermore, we ran the TFC algorithm by increasing the angle and spatial discretizations to achieve the 7-digit accuracy published by Ganapol [32] for $m = 0$. Our results are presented in Table 12. For those simulations, the following parameters have been employed: $N = 35$, $M = 300$,and $m = 90$. TFC has achieved the reported solutions with a computational time of 2.6 s. Conversely, for $m = 8$Fourier component, the 7-digit accuracy of Ganapol have been achieved without the need to increase eitheer the angle or spatial discretization. Indeed, as in the previous case, the following parameters have been used: $N = 15$, $M = 100$,and $m = 40$. Results are shown in Table 13.

For the Haze L problem, which was posed as a basic test problem by the Radiation Commission of the International Association of Meteorology and Atmospheric Physics (IAMAP) [33], we compared our results with the ones reported in Benassi et al. [27] and [31]. In such cases, the authors used an angular discretization of $N = 100 - 150$. For all our simulations, we ran the TFC algorithm with a discretization of $N = 30$. The same parame-

Table 18
Haze L problem—The intensity $I_*(\tau, \mu, \phi)$ for the Haze L phase function with $\omega = 0.9$, $\mu_0 = 0.5$,and $\phi - \phi_0 = \pi$,with a computational time of approximately 132 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	2.28190e−2	2.14170e−2	1.99920e−2	1.71574e−2	9.34719e−3	4.02513e−3	0
−0.9	2.61852e−2	2.46217e−2	2.30415e−2	1.99042e−2	1.11589e−2	4.97196e−3	0
−0.8	3.20368e−2	3.01730e−2	2.82831e−2	2.45151e−2	1.38928e−2	6.24448e−3	0
−0.7	3.74817e−2	3.54358e−2	3.33331e−2	2.90811e−2	1.67591e−2	7.61010e−3	0
−0.6	4.12139e−2	3.92700e−2	3.72011e−2	3.28685e−2	1.95280e−2	9.02300e−3	0
−0.5	4.99464e−2	4.77799e−2	4.54343e−2	4.04517e−2	2.46595e−2	1.16636e−2	0
−0.4	5.50134e−2	5.32394e−2	5.11567e−2	4.64342e−2	2.98815e−2	1.47743e−2	0
−0.3	6.44934e−2	6.30132e−2	6.10801e−2	5.64129e−2	3.85985e−2	2.04385e−2	0
−0.2	7.08868e−2	7.03673e−2	6.91450e−2	6.55321e−2	4.91145e−2	2.92042e−2	0
−0.1	6.95600e−2	7.10895e−2	7.13778e−2	7.01215e−2	5.90302e−2	4.25269e−2	0
−0.0	5.47761e−2	6.23325e−2	6.55280e−2	6.81201e−2	6.39396e−2	5.39412e−2	0
0.0	0	6.23325e−2	6.55280e−2	6.81201e−2	6.39396e−2	5.39412e−2	3.42189e−2
0.1	0	1.99925e−2	3.42555e−2	5.10066e−2	6.25009e−2	5.80166e−2	4.66801e−2
0.2	0	9.41145e−3	1.79861e−2	3.18930e−2	5.27234e−2	5.56537e−2	5.06958e−2
0.3	0	5.52988e−3	1.10120e−2	2.10538e−2	4.15040e−2	4.87980e−2	4.91257e−2
0.4	0	3.67310e−3	7.45174e−3	1.48051e−2	3.24697e−2	4.12526e−2	4.48014e−2
0.5	0	2.67314e−3	5.46407e−3	1.10690e−2	2.59261e−2	3.47946e−2	3.99910e−2
0.6	0	2.11016e−3	4.31933e−3	8.82241e−3	2.14934e−2	2.99722e−2	3.59328e−2
0.7	0	1.81107e−3	3.69988e−3	7.56877e−3	1.88391e−2	2.69599e−2	3.33330e−2
0.8	0	1.72244e−3	3.50577e−3	7.15364e−3	1.79613e−2	2.61129e−2	3.29719e−2
0.9	0	1.93240e−3	3.91318e−3	7.93848e−3	1.98733e−2	2.90534e−2	3.70748e−2
1.0	0	5.07113e−3	1.01191e−2	2.00435e−2	4.76083e−2	6.73492e−2	8.37579e−2

Table 19
Haze L problem—The intensity $I_*(\tau, \mu)$ for the Haze L phase function with $\omega = 0.9$and $\mu_0 = 1$,according to Ganapol [32], with a computational time of approximately 4.4 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	2.7971665e−2	2.6583431e−2	2.5179489e−2	2.2342144e−2	1.3751918e−2	6.7043863e−3	0
−0.9	3.0180197e−2	2.8742728e−2	2.7276328e−2	2.4282402e−2	1.5036989e−2	7.3279307e−3	0
−0.8	3.1447755e−2	3.0070750e−2	2.8641094e−2	2.5662054e−2	1.6096218e−2	7.8550379e−3	0
−0.7	3.4383906e−2	3.3055747e−2	3.1640694e−2	2.8605980e−2	1.8314210e−2	9.0026264e−3	0
−0.6	3.9130810e−2	3.7890987e−2	3.6513506e−2	3.3427829e−2	2.2097749e−2	1.1061916e−2	0
−0.5	4.5637920e−2	4.4617111e−2	4.3383966e−2	4.0403191e−2	2.8008565e−2	1.4514343e−2	0
−0.4	5.3511337e−2	5.2985449e−2	5.2140980e−2	4.9686294e−2	3.6854018e−2	2.0230769e−2	0
−0.3	6.1542012e−2	6.1991418e−2	6.1978635e−2	6.0886294e−2	4.9616521e−2	2.9827680e−2	0
−0.2	6.6956243e−2	6.9056402e−2	7.0499635e−2	7.2037240e−2	6.6696988e−2	4.6300639e−2	0
−0.1	6.5529583e−2	7.0004105e−2	7.3432378e−2	7.8590386e−2	8.4462845e−2	7.3611021e−2	0
−0.0	5.1748534e−2	6.1709609e−2	6.8016336e−2	7.7466538e−2	9.3997859e−2	9.7483668e−2	0
0.0	0	6.1709609e−2	6.8016336e−2	7.7466538e−2	9.3997859e−2	9.7483668e−2	7.9312594e−2
0.1	0	2.2494931e−2	3.9518122e−2	6.2652973e−2	9.6254337e−2	1.0871841e−1	1.0818930e−1
0.2	0	1.3100677e−2	2.5340076e−2	4.6772905e−2	9.1591615e−2	1.1381468e−1	1.2421167e−1
0.3	0	1.0194331e−2	2.0270341e−2	3.9472225e−2	8.7467481e−2	1.1662007e−1	1.3571209e−1
0.4	0	9.5290644e−3	1.9067650e−2	3.7770256e−2	8.8332315e−2	1.2250259e−1	1.4826767e−1
0.5	0	1.0263750e−2	2.0502330e−2	4.0649220e−2	9.6418345e−2	1.3581653e−1	1.6751584e−1
0.6	0	1.2529327e−2	2.4909477e−2	4.9065634e−2	1.1533634e−1	1.6223048e−1	2.0070062e−1
0.7	0	1.7417124e−2	3.4415206e−2	6.7081148e−2	1.5398186e−1	2.1356335e−1	2.6167192e−1
0.8	0	2.8562211e−2	5.6020429e−2	1.0769702e−2	2.3848565e−1	3.2254956e−1	3.8692070e−1
0.9	0	6.1633112e−2	1.1976311e−1	2.2612375e−1	4.7610276e−1	6.1970346e−1	7.1774509e−1
1.0	0	3.2812354e−1	6.2906510e−1	1.1563161	2.2483946	2.7414726	2.9776602

ters, as reported below, are used in all cases. An exception is represented by the parameter m, which is varied across cases. The one–angle case, when the incident beam direction is $\mu_0 = 1$,requires to compute only the 1stFourier component $m = 0$,i.e. without considering the azimuthal angle. In the two–angle cases, we calculated all the $L + 1$Fourier components. The TFC results are reported in Tables 14–18. For the normal incidence case, the following parameters have been used: $N = 30$, $M = 250$, $m_{op} = 90$, $\tau_0 = 1$, $S_0 = 1$, $\mu_0 = 1$, $\omega = 0.9$. For the conservative case, the following parameters have been used: $N = 30$, $M = 250$, $m_{op} = 90$, $\tau_0 = 1$, $S_0 = 1$, $\mu_0 = 1$, $\omega = 1$. For both simulations, the computational times are approximately 1.6 s.

For the two-angle case, the incident beam is not $\mu_0 = 1$,so we needed to compute all the $L + 1$Fourier components. For the Haze L problem, 83 components are required. However, we found out that it was not necessary to modify the parameters for each case. The three different cases are reported below:

(1) $\phi - \phi_0 = 0$,
(2) $\phi - \phi_0 = \pi/2$,
(3) $\phi - \phi_0 = \pi$.

The set of parameters used was: $N = 30$, $M = 250$, $m_{op} = 90$, $\tau_0 = 1$, $S_0 = 1$, $\mu_0 = 0.5$, $\omega = 0.9$. The TFC computational time is reported to be approximately 132 s. Even in this case we were able to obtain all the digits published by Siewert et al. [27,31].

Again, we increased the TFC angular and spatial discretizations to improve the accuracy of the solutions, and to achieve the 7-digit accuracy as published by Ganapol for the Haze L problem. We computed the incident cases ($\mu_0 = 1$), with both conservative single scattering albedo $\omega = 1$and non-conservative single scattering albedo $\omega = 0.9$. We found that the optimal parameters are $N = 35$,$M = 400$,and $m_{op} = 100$. The TFC computational time is reported to be approximately 4.4 s. The results are reported in Tables 19and 20.

Table 20
Haze L problem—The intensity $I(\tau, \mu)$ for the Haze L phase function with $\omega = 1$and $\mu_0 = 1$,according to Ganapol [32], with a computational time of approximately 4.4 s.

μ	$\tau = 0$	$\tau = 0.05\tau_0$	$\tau = 0.1\tau_0$	$\tau = 0.2\tau_0$	$\tau = 0.5\tau_0$	$\tau = 0.75\tau_0$	$\tau = \tau_0$
−1.0	3.6145156e−2	3.4339396e−2	3.2510866e−2	2.8812216e−2	1.7628611e−2	8.5258908e−3	0
−0.9	3.9781870e−2	3.7872320e−2	3.5920682e−2	3.1930313e−2	1.9620173e−2	9.4573134e−3	0
−0.8	4.2731263e−2	4.0840607e−2	3.8873442e−2	3.4767734e−2	2.1601856e−2	1.0395857e−2	0
−0.7	4.8005147e−2	4.6131929e−2	4.4130697e−2	3.9829198e−2	2.5247889e−2	1.2217079e−2	0
−0.6	5.5821353e−2	5.4043177e−2	5.2059351e−2	4.7598583e−2	3.1183660e−2	1.5361836e−2	0
−0.5	6.6094221e−2	6.4629636e−2	6.2844874e−2	5.8497071e−2	4.0273976e−2	2.0562127e−2	0
−0.4	7.8148081e−2	7.7440255e−2	7.6250769e−2	7.2704873e−2	5.3729974e−2	2.9128534e−2	0
−0.3	8.9968154e−2	9.0770642e−2	9.0878384e−2	8.9471128e−2	7.2964349e−2	4.3468799e−2	0
−0.2	9.7081540e−2	1.0042085e−1	1.0278927e−1	1.0550594e−1	9.8377715e−2	6.7994924e−2	0
−0.1	9.2932814e−2	9.9818714e−2	1.0519502e−1	1.1349749e−1	1.2403692e−1	1.0839912e−2	0
−0.0	6.9877391e−2	8.4667310e−2	9.4166299e−1	1.0872694e−1	1.3576248e−1	1.4277947e−1	0
0.0	0	8.4667310e−2	9.4166299e−1	1.0872694e−1	1.3576248e−1	1.4277947e−1	1.1480771e−1
0.1	0	2.9541820e−2	5.2434564e−2	8.4564915e−2	1.3509602e−1	1.5610649e−1	1.5697621-1
0.2	0	1.6490681e−2	3.2281653e−2	6.0752687e−2	1.2435036e−1	1.5892546e−1	1.7681766e−1
0.3	0	1.2342100e−2	2.4848764e−2	4.9396789e−2	1.1481121e−1	1.5793652e−1	1.8830112e−1
0.4	0	1.1187938e−2	2.2644991e−2	4.5754673e−2	1.1226864e−1	1.6086203e−1	2.0001870e−1
0.5	0	1.1795943e−2	2.3790964e−2	4.8000306e−2	1.1907946e−1	1.7319074e−1	2.1963289e−1
0.6	0	1.4204907e−2	2.8458379e−2	5.6873102e−2	1.3905102e−1	2.0144487e−1	2.5598334e−1
0.7	0	1.9583294e−2	3.8924848e−2	7.6745368e−2	1.8200357e−1	2.5898644e−1	3.2512495e−1
0.8	0	3.1953231e−2	6.2942983e−2	1.2204484e−1	2.7718191e−1	3.8276705e−1	4.6865779e−1
0.9	0	6.8726703e−2	1.3391677e−1	2.5425935e−1	5.4460066e−1	7.1944669e−1	8.4608373e−1
1.0	0	3.6493954e−1	7.0026634e−1	1.2895497	2.5225517	3.0931861	3.3809098

7. Conclusions and future outlooks

This paper highlights the potential of the *Theory of Functional Connections* to solve one-dimensional, anisotropic Radiative Transfer Problems in homogeneous media, with high accuracy for both one angle and two angles incident source beam, at least up to the 7thfigure.

The advantage of the proposed method lies in its elegant and concise implementation, in the analytically satisfaction of boundary conditions, as well as in the absence of numerical issues for the conservative case ($\omega = 1$), and in the ability to compute the photon flux in any exact query spatial and directional points without further manipulations.

Moreover, it has been shown that via TFC, one is able to directly solve the Radiative Transfer Equation in the form of Eq. (38), without further transformations on the equation itself as usually necessary in other methods [16,32]. Future works will focus on the application of the TFC to the Matrix Riccati Equation (MRE) solution of the 1D RTE arising from the application of the interaction principle of particle transport to Eq. (38) [34]. Indeed, using the MRE would allow us to tackle the problem via recursively applying the TFC method as presented in Mortari et al. [22] for problems with a long integration space (e.g., τ for our case). Furthermore, we believe that this approach will allow us to combine the TFC methodology with the adding-doubling method proposed by Ganapol in Ganapol [35–38], in synergy with the Wynn-Epsilon Convergence Acceleration to further improve the performance of the TFC framework in solving even more challenging Radiative Transfer problems, such as the Cloud C1 problem, posed by IAMAP [33]. This benchmark requires a large optical depth ($\tau_0 = 64$). This is causing the matrix to invert to be large. Thus the problem becomes too computationally expensive both in terms of memory and computational time. Hence, at this stage, large optical depth is still the limitation of our proposed method. Indeed, while our TFC based method for solving the RTE is comparable with the other state of the art methods in tackling problems with small optical depths (e.g., Mie Scattering and Haze L problems), it appears to suffer when solving problems with large ones.

Additionally, future works will focus on the application of the newly developed *Extreme Theory of Functional Connections* (X-TFC) [39], that is a Physics-Informed Neural Network framework, to tackle some of the most challenging Radiative Transfer problems such as multi-slab case [40], time-dependent integrodifferential Boltzmann Equations, and, subsequently, 3D time-dependent Radiative Transfer Problems. Moreover, the application of X-TFC to time-dependent radiative transfer problems would aim to offer an alternative to the existing frameworks based on spectral and collocation methods such as [41–43]. We are currently applying the X-TFC physics-informed framework in solving a series of Rarefied-Gas Dynamics problems such as Poiseuille Flow and Thermal Creep Flow in a plane channel. Extension of the X-TFC to multi-dimensional RTEs is expected to be relatively straightforward.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Mario De Florio: Methodology, Investigation, Software, Writing - original draft. **Enrico Schiassi:** Methodology, Investigation, Software. **Roberto Furfaro:** Supervision, Conceptualization, Investigation, Writing - review & editing. **Barry D. Ganapol:** Resources, Writing - review & editing. **Domiziano Mostacci:** Supervision.

References

[1] Chandrasekhar S. Radiative transfer. Courier Corporation; 2013.
[2] Wick GC. Über ebene diffusionsprobleme. Z Phys 1943;121(11–12):702–18.
[3] Carlson B, Bell G. Solution of the transport equation by the Sn method. Tech. Rep.. Los Alamos Scientific Lab, N Mex; 1958.
[4] Vilhena M, Segatto C, Barichello L. A particular solution for the Sn radiative transfer problems. J Quant Spectrosc Radiat Transf 1995;53(4):467–9.
[5] Chien K-Y. Application of the S, method to spherically symmetric radiative–transfer problems. AIAA J 1972;10(1):55–9.
[6] Simch M, Segatto C, Vilhena M. An analytical solution for the Sn radiative transfer equation with polarization in a slab by the LTSN method. J Quant Spectrosc Radiat Transf 2006;97(3):424–35.
[7] Benoist P, Kavenoky A. A new method of approximation of the Boltzmann equation. Nucl Sci Eng 1968;32(2):225–32.
[8] Siewert C, Benoist P. The FN method in neutron-transport theory. Part I: theory and applications. Nucl Sci Eng 1979;69(2):156–60.
[9] Devaux C, Siewert C. The FN method for radiative transfer problems without azimuthal symmetry. Z Angew Math Phys ZAMP 1980;31(5):592–604.

[10] Garcia R, Siewert C. The FN method for radiative transfer models that include polarization effects. J Quant Spectrosc Radiat Transf 1989;41(2):117–45.
[11] Ganapol BD, Myneni R. The FN method for the one-angle radiative transfer equation applied to plant canopies. Remote Sens Environ 1992;39(3):213–31.
[12] Benassi M, Cotta R, Siewert C. The PN method for radiative transfer problems with reflective boundary conditions. J Quant Spectrosc Radiat Transf 1983;30(6):547–53.
[13] Benassi M, Garcia R, Karp A, Siewert C. A high-order spherical harmonics solution to the standard problem in radiative transfer. Astrophys J 1984;280:853–64.
[14] Siewert C, Thomas J Jr.. A particular solution for the PN method in radiative transfer. J Quant Spectrosc Radiat Transf 1990;43(6):433–6.
[15] Barichello L, Siewert CE. A discrete-ordinates solution for a non-grey model with complete frequency redistribution. J Quant Spectrosc Radiat Transf 1999;62(6):665–75.
[16] Siewert C. A concise and accurate solution to Chandrasekhar's basic problem in radiative transfer. J Quant Spectrosc Radiat Transf 2000;64(2):109–30.
[17] Picca P, Furfaro R. Analytical discrete ordinate method for radiative transfer in dense vegetation canopies. J Quant Spectrosc Radiat Transf 2013;118:60–9.
[18] Ganapol B. Radiative transfer with internal reflection via the converged discrete ordinates method. J Quant Spectrosc Radiat Transf 2011;112(4):693–713.
[19] Picca P, Furfaro R, Ganapol BD. An efficient multiproblem strategy for accurate solutions of linear particle transport problems in spherical geometry. Nucl Sci Eng 2012;170(2):103–24.
[20] Mortari D. The theory of connections: connecting points. Mathematics 2017;5(4):57.
[21] Mortari D. Least-squares solution of linear differential equations. Mathematics 2017;5(4):48.
[22] Mortari D, Johnston H, Smith L. High accuracy least-squares solutions of nonlinear differential equations. J Comput Appl Math 2019;352:293–307.
[23] Furfaro R, Mortari D. Least-squares solution of a class of optimal space guidance problems via theory of connections. Acta Astronaut 2019. doi:10.1016/j.actaastro.2019.05.050.
[24] Johnston H., Schiassi E., Furfaro R., Mortari D.. Fuel-Efficient Powered Descent Guidance on Large Planetary Bodies via Theory of Functional Connections. arXiv:200103572 2020.
[25] Schiassi E., D'Ambrosio A., Johnston H., Furfaro R., Curti F., Mortari D.. Complete Energy Optimal Landing on Planetary Bodies via Theory of Functional Connections. AAS 20-557, AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, CA, August 9-13, 2020.
[26] Meador W, Weaver W. Two-stream approximations to radiative transfer in planetary atmospheres: a unified description of existing methods and a new improvement. J Atmos Sci 1980;37(3):630–43.
[27] Benassi M, Garcia R, Karp A, Siewert C. A high-order spherical harmonics solution to the standard problem in radiative transfer. Astrophys J 1984;280:853–64.
[28] Gradshteyn IS, Ryzhik IM. Table of integrals, series, and products. Academic Press; 2014.
[29] Chalhoub E, Garcia R. On the solution of azimuthally dependent transport problems with the ANISN code. Ann Nucl Energy 1997;24(13):1069–84.
[30] Chalhoub E, Garcia R. A new quadrature scheme for solving azimuthally dependent transport problems. Transp Theory Stat Phys 1998;27(5–7):607–24.
[31] Garcia R, Siewert C. Benchmark results in radiative transfer. Transp Theory Stat Phys 1985;14(4):437–83.
[32] Ganapol BD. The response matrix discrete ordinates solution to the 1D radiative transfer equation. J Quant Spectrosc Radiat Transf 2015;154:72–90.
[33] Lenoble J. Standard procedures to compute atmospheric radiative transfer in a scattering atmosphere-volume I. International association of meteorology and atmospheric physics (IAMAP), Boulder, Colorado, USA; 1977.
[34] Ganapol B.D.. Matrix riccati equation solution of the 1d radiative transfer equation. arXiv:200702437 2020.
[35] Ganapol BD. 1D thermal creep channel flow in the BGK approximation by adding and doubling. Ann Nucl Energy 2019;134:441–51.
[36] Ganapol BD. Particle transport in a 3d duct by adding and doubling. J Comput Theor Transp 2017;46(3):202–28.
[37] Ganapol BD. Poiseuille channel flow by adding and doubling. In: AIP conference proceedings, 1786. AIP Publishing LLC; 2016. p. 070009.
[38] Ganapol BD. A 1d radiative transfer benchmark with polarization via doubling and adding. J Quant Spectrosc Radiat Transf 2017;201:236–50.
[39] Schiassi E., Leake C., De Florio M., Johnston H., Furfaro R., Mortari D.. Extreme Theory of Functional Connections: A Physics-Informed Neural Network Method For Solving Parametric Differential Equations. arXiv2020b.
[40] Previti A, Furfaro R, Picca P, Ganapol BD, Mostacci D. Solving radiative transfer problems in highly heterogeneous media via domain decomposition and convergence acceleration techniques. Appl Radiat Isot 2011;69(8):1146–50.
[41] Chen S-S, Li B-W, Sun Y-S. Chebyshev collocation spectral method for solving radiative transfer with the modified discrete ordinates formulations. Int J Heat Mass Transf 2015;88:388–97.
[42] Kim AD, Moscoso M. Chebyshev spectral methods for radiative transfer. SIAM J Sci Comput 2002;23(6):2074–94.
[43] Sun Y-S, Li B-W. Chebyshev collocation spectral method for one-dimensional radiative heat transfer in graded index media. Int J Therm Sci 2009;48(4):691–8.

Appendix C

De Florio et al. (2021) Thermal Creep Flow

Reproduced from De Florio, M., Schiassi, E., Ganapol, B. D., & Furfaro, R. (2021). Physics-informed neural networks for rarefied-gas dynamics: Thermal creep flow in the Bhatnagar–Gross–Krook approximation. *Physics of Fluids*, 33(4), 047110. https://doi.org/10.1063/5.0046181 [9], with the permission of AIP Publishing.

Physics of Fluids

Physics-informed neural networks for rarefied-gas dynamics: Thermal creep flow in the Bhatnagar–Gross–Krook approximation

Cite as: Phys. Fluids **33**, 047110 (2021); https://doi.org/10.1063/5.0046181
Submitted: 01 February 2021 • Accepted: 27 March 2021 • Published Online: 21 April 2021

Mario De Florio, Enrico Schiassi, Barry D. Ganapol, et al.

Physics of Fluids ARTICLE scitation.org/journal/phf

Physics-informed neural networks for rarefied-gas dynamics: Thermal creep flow in the Bhatnagar–Gross–Krook approximation

Cite as: Phys. Fluids **33**, 047110 (2021); doi: 10.1063/5.0046181
Submitted: 1 February 2021 · Accepted: 27 March 2021 ·
Published Online: 21 April 2021

Mario De Florio,[1] Enrico Schiassi,[1] Barry D. Ganapol,[2] and Roberto Furfaro[1,2,a)]

AFFILIATIONS

[1]Department of Systems and Industrial Engineering, The University of Arizona, 1127 James E. Rogers Way, Tucson, Arizona 85719, USA
[2]Department of Aerospace and Mechanical Engineering, The University of Arizona, 1130 N Mountain Ave, Tucson, Arizona 85721, USA

[a)]**Author to whom correspondence should be addressed**: robertof@email.arizona.edu

ABSTRACT

This work aims at accurately solve a thermal creep flow in a plane channel problem, as a class of rarefied-gas dynamics problems, using Physics-Informed Neural Networks (PINNs). We develop a particular PINN framework where the solution of the problem is represented by the Constrained Expressions (CE) prescribed by the recently introduced Theory of Functional Connections (TFC). CEs are represented by a sum of a free-function and a functional (e.g., function of functions) that analytically satisfies the problem constraints regardless to the choice of the free-function. The latter is represented by a shallow Neural Network (NN). Here, the resulting PINN-TFC approach is employed to solve the Boltzmann equation in the Bhatnagar–Gross–Krook approximation modeling the Thermal Creep Flow in a plane channel. We test three different types of shallow NNs, i.e., standard shallow NN, Chebyshev NN (ChNN), and Legendre NN (LeNN). For all the three cases the unknown solutions are computed via the extreme learning machine algorithm. We show that with all these networks we can achieve accurate solutions with a fast training time. In particular, with ChNN and LeNN we are able to match all the available benchmarks.

I. INTRODUCTION

Accurate calculations of rarefied gas flows at an arbitrary Knudsen number is extremely challenging. Rarefied Gas Dynamics (RGD) describes the fluid dynamics of a gas where the mean free path (λ) of the molecules is larger than the size (d) of the chamber under test. Generally, a gas flow is considered rarefied when the Knudsen number $Kn = \frac{\lambda}{d} > 1$. In particular, a thermal creep flow, also known as Stokes flow, is a flow of a slightly rarefied gas caused by the temperature gradient along a wall, and when in the fluid the viscous forces dominate over inertial forces.

The phenomenon of thermal creep effect has been initially studied independently by Reynolds,[1] Maxwell,[2] and Knudsen.[3] Subsequently, many researchers have addressed the problem with an experimental approach.[4–9] From a theoretical point of view, the thermal creep problem in the field of RGD has been investigated to achieve trustworthy solutions for a large interval of Knudsen number, with the fundamentals of kinetic theory.[10–12] The fundamental mathematical model generally employed to describe thermal creep flows is the Boltzmann equation,[13] or its approximations such as Bhatnagar, Gross, and Krook (BGK)[14] and Shakhov,[15] which have been extensively used in the literature to find their numerical solutions with several methods.[16–26]

In the last decay, the thermal creep flow has been widely studied based on the Direct Simulation Monte Carlo (DSMC),[27] for both monatomic[28–31] and polyatomic gases.[17,32–34] Over the years, several methods have been developed and improved to solve Boltzmann integrodifferential transport equation applied to problems of physics, such as radiative transfer, neutron transport, and RGD. The first family of methods developed for the transport theory dates back to the 1940s, named Discrete Ordinates Method (DOM).[35,36] Subsequently, the DOM has undergone improvements and changes to achieve ever higher accuracy. Among these, we count the S_n method, which uses line segments to approximate the $\mu-$dependence of the angular flux,[37–39] and the C_N method, where the angular flux is approximated by an expansion of orthogonal polynomials.[40] During the 1980s the F_N method and the P_N method were developed, extended, and applied

to solve problems in neutron and photon transport.[41–46] Around 20 years ago, Barichello and Siewert presented the first version of the Analytical Discrete Ordinates (ADO) method, which achieved some of the most accurate benchmark in the field of radiative transfer.[47,48]

Many other methods and applications for solving problems arising from the Boltzmann equation have been published in the footsteps of the previous ones to date,[49–51] and as many new ones were born.

In particular, recently, new methods to solve differential equations (DE) were developed within the Machine Learning community. Named Physics-Informed Neural Networks (PINNs), such methods train Neural Networks (NN) to learn the solution of Differential Equations (DE). However, PINNs are not only employed to solve DEs and can be generally considered as machine learning algorithms that embed physics into data-driven representation of functional relationships underlying collections of input–output pairs. The term PINN, as described by Raissi *et al.*,[52] defines NNs that use the physics as regularizer in the loss function. As an example, assume that one wants to perform a regression of a set of experimental data using a NN, and that such collected data represent a physical phenomena modeled via a set of DEs. In standard regression, one could approximate the data using a NN (either shallow or deep) trained to minimize a Mean Squared Error (MSE) as loss function. However, doing so, there is no guarantee that the physics phenomena governing the data would not be violated.

More precisely, especially when the data are noisy, one could experience the issue of over-fitting. That is, the data would be accurately fitted (e.g., negligible error between the NN approximation and the experimental data), but the relation input–output, governed by the physics phenomena, could be poorly learned. For instance, imagine to kick a ball, and to measure its position (which represent the output of the dataset) at certain times (which represent the input of the dataset), from when the ball is kicked till it lands on the ground, to learn, from the measurements, what the trajectory (which represents the relation input–output) is. In this easy example, it is known, from physics principles, that the trajectory is a parabola. However, if one wants to understand what the trajectory is by fitting the experimental data (e.g., the position-time pairs), if the data are noisy and the over-fitting occurs the trajectory may not be learned correctly. For example, the trajectory could be approximated as a periodic function, instead of as a parabola, and this would violate the physics phenomena governing the collected experimental data.

PINN are therefore introduced to ensure that DEs, coming from first principles and modeling the physics of the experimental dataset, are added as a penalty to the loss function. This additional term acts as a regularizer that penalizes the training when the DE and the Initial Conditions (ICs) and/or Boundary Conditions (BCs) are violated. Overall, one ensures that the physics underlying the process is not violated. This approach is called data-driven solution of DEs. Conversely, when the goal is to estimate parameters governing some physical phenomena modeled through a DE (e.g., the thermal conductivity in the heat equation) one conventionally talks about data-driven parameters discovery of DEs (e.g., inverse problems).[52,53] When data are not available, and therefore the loss function contains only the residual of the DEs, as well as the corresponding ICs and/or BCs, PINNs are used for approximating the solutions of problems involving ordinary differential equations (ODEs) and is Partial Differential Equations (PDEs). To this end PINNs can be also viewed as methods that approximate solutions of DEs using NNs.

The goal of this paper is to develop and evaluate a PINNs-based methodology to solve the 1D thermal creep channel flow in a plane channel problem. Following the formulation presented by Ganapol,[54] we demonstrate that the proposed PINN-based approach is accurate in solving this class of problems arising from the Boltzmann integro-differential equation. Specifically, to solve this problem, we develop a particular PINN framework that brings together NNs and the Theory of Functional Connections (TFC).[55,56] Recently developed by Mortari,[57] TFC enables functional interpolation for a large class of mathematical objects. According to TFC, any solution of a mathematical problem can be represented via the so-called Constrained Expressions (CEs). The CEs are a sum of a free-function and a functional that analytically satisfies the problem constraints, regardless to the choice of the free-function.[57] Among its many applications, TFC has been applied to solve ODEs with generally excellent results, leading to machine-level accuracy solutions in a short computational time.[58,59] The authors already demonstrated that TFC can be successfully employed to solve the Chandrasekhar's basic problem in radiative transfer,[60] as well as optimal control problems for aerospace applications.[61–63] Here, we consider a PINN TFC-based approach where shallow NN are employed to solve DEs while analytically satisfying IC/BCs via TFC prescriptions. Such approach aims at overcoming the main limitations of the standard PINN and TFC methods (e.g., solution accuracy and curse of dimensionality, respectively). We solve the Thermal Creep Flow problem using, as free function in the CE, three different types of shallow NNs: standard shallow NN,[56] Chebyshev NN (ChNN),[64] and Legendre NN (LeNN).[65] For all the three cases the unknown solutions are then computed via the Extreme Learning Machine (ELM) algorithm.[66] According to the ELM algorithm, input weights and bias are randomly selected and not tuned during the training, leaving the outputs weight to be the only trainable parameters. Thus, the training is reduced to a fast least squares.

The paper is organized as follows: in Sec. II the PINN methodology and how to apply it to solve a generic first order linear ODE are presented. In Sec. III how to apply PINN TFC-based methods to tackle generic first order linear ODEs is explained. The PINN formulation of the Thermal creep flow in a plane channel problem is showed in Secs. IV and V. Numerical results are presented and discussed in Sec. VI.

II. PHYSICS-INFORMED NEURAL NETWORKS

PINN have been proposed by Raissi *et al.*[52] to solve two main classes of problems: data-driven solution and data-driven discovery of PDEs, validating the accuracy with benchmark comparisons. Although in Ref. 52 PINN were introduced to tackle problem governed by PDEs, the same framework can be used to solve problem involving ODEs as well. The most known mesh-based methods [such as Finite Difference Method (FDM) and Finite Element Method (FEM)[67]] use piece-wise polynomial functions as basis functions. Conversely, PINN employs a single deep neural network (DNN) to fully represent the solution. In particular, the training set consists on randomly parsed data points from the full high-resolution dataset, and randomly sampled collocation points for enforcing BCs and to enforce the PDE, on the boundary and inside the solution domain, respectively. Once the output of DNN is computed, it is plugged in the residual of the DE and in the residual of the BCs. The training of the NN is used to learn the weights and biases parameters that minimize the physics-based cost function, which can be DE and BCs least squares

Phys. Fluids **33**, 047110 (2021); doi: 10.1063/5.0046181 **33**, 047110-2

solved with gradient-based optimization methods. Thus, the key feature of the PINN is the embedding of the information of the physics of the problem, derived from the DE and the BCs, in their cost functions. To better clarify how a PINN works, the mathematical formulation to solve a first order linear ODE is reported below.

A. PINN approach to solving first order linear ODEs

Consider solving the following first-order ODE:

$$F(t, y, \dot{y}) = 0 \quad \text{subject to :} \quad y(t_0) = y_0. \tag{1}$$

The unknown solution is approximated by using a NN (that can be either deep or shallow). That is,

$$y(t) \approx y_{NN}(t, \Theta), \tag{2}$$

where Θ are the NN parameters (e.g., weights and biases). The derivative is

$$\dot{y}(t) \approx \dot{y}_{NN}(t, \Theta) = \frac{d}{dt} y_{NN}(t, \Theta), \tag{3}$$

which is computed using automatic differentiation.[68,69] Thus, the approximated solution and its derivative defined by Eqs. (2) and (3) are plugged into (1). Doing so, we have the following residuals for both ODE and BC

$$\mathscr{R}_F(t) = F(t, y_{NN}, \dot{y}_{NN}), \tag{4}$$

$$\mathscr{R}_{BC}(t) = y_{NN}(t_0, \Theta) - y_0, \tag{5}$$

where $y_{NN}(t_0, \Theta)$ is the NN comptued in $t = t_0$. Now, we need to build the MSE loss as

$$MSE = MSE_F + MSE_{BC}, \tag{6}$$

where,

$$MSE_F = \frac{1}{N_F} \sum_{i=1}^{N_F} \mathscr{R}_F^2(t_i) \tag{7}$$

and

$$MSE_{BC} = \mathscr{R}_{BC}^2(t_0). \tag{8}$$

Here, N_F are the training points and t_i is the independent variable computed in the ith training point.

Now, the NN parameters Θ are learned by minimizing the MSE loss with gradient-based optimization methods, such as stochastic gradient descent, and plugged into (2) to obtain the approximated solution of the ODE.

III. PINN TFC-BASED APPROACH TO SOLVING FIRST ORDER LINEAR ODEs

As previously mentioned, in this work we develop an alternative PINN framework. In the proposed approach, we merge the standard PINN with the TFC method.[55,56]

Here, we present the step-by-step procedure for solving boundary value linear ODEs via the proposed PINN-TFC based method.

Consider solving the following first-order ODE

$$F(t, y, \dot{y}) = 0 \quad \text{subject to :} \quad y(t_0) = y_0. \tag{9}$$

The unknown solution is approximated via the constrained expression, defined within the TFC framework. That is,

$$y(t) = g(t) + \sum_{k=1}^{n} \eta_k p_k(t) = g(t) + \boldsymbol{\eta}^T \mathbf{p}(t), \tag{10}$$

where, $p_k(t)$ are n assigned linearly independent functions and the $g(t)$ is the free function, which must be linearly independent from the $p_k(t)$ functions. The values of the η_k coefficients are calculated by imposing a set of n linear constraints in $y(t)$. By doing that, the constraints for the differential equations are satisfied for any chosen $g(t)$.

For a first order ODE the CE becomes

$$y(t) = g(t) + \eta\, p(t), \tag{11}$$

where $p(t) = 1$ (see Refs. 58 and 59 for the details of this decision). Thus, the CE becomes

$$y(t) = g(t) + \eta. \tag{12}$$

By imposing the constraint, we get

$$y(t_0) = y_0 = g_0 + \eta \quad \Rightarrow \quad \eta = y_0 - g_0, \tag{13}$$

which is plugged into Eq. (12) to obtain the CE in its final form

$$y(t) = g(t) + (y_0 - g_0), \tag{14}$$

which represents all possible functions satisfying the boundary value constraint. The derivative then follows:

$$\dot{y}(t) = \dot{g}(t). \tag{15}$$

The CE defined by Eq. (14) and its derivative are plugged into our ODE which then becomes

$$F(t, y, \dot{y}) = 0, \tag{16}$$

that is now a boundary condition free. By substituting Eqs. (14) into (16), the differential equation F is transformed into a new differential equation $\tilde{F}$, which is only a function of the independent variable t and the free-function $g(t)$, that is,

$$\tilde{F}(t, g, \dot{g}) = 0. \tag{17}$$

This differential equation is now unconstrained and will always satisfy the boundary-value due to the way Eq. (12) is built. In this work, we choose the free function $g(t)$ to be a shallow NN, such that

$$g(t) = \sum_{j=1}^{L} \beta_j \sigma(w_j t + b_j) = \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_L \end{bmatrix}^T \boldsymbol{\beta} = \boldsymbol{\sigma}^T \boldsymbol{\beta}, \tag{18}$$

where L is the number of hidden neurons, $w_j \in \mathbb{R}$ is the input weights vector connecting the jth hidden neuron and the input nodes, $\beta_j \in \mathbb{R}$ with $j = 1, \ldots, L$ is the jth output weight connecting the jth hidden neuron and the output node, and b_j is the bias of the jth hidden neuron, and $\sigma_j(\cdot)$ are activation functions. Once the free-function is defined, the unknown solution $y(t)$, defined by Eq. (14), is computed via the ELM algorithm. According to the ELM algorithm Ref. 66, input weights and biases are randomly selected and not tuned during the training, thus they are known parameters. The activation functions,

Phys. Fluids 33, 047110 (2021); doi: 10.1063/5.0046181

$\sigma_j(\cdot)$, are chosen by the user, so they are also known. Therefore, the only unknowns to compute are the output weights $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_L]^T$. The interested reader can find more details about the ELM and its convergence proofs in Ref. 66. In this work we use three different types of shallow NNs: standard shallow NN, ChNN, and LeNN. From previous works,[70] we noticed that for ChNN and LeNN, the choice of input weights and biases that guarantees the best accuracy, is to set $w_j = 1$ and $b_j = 0\,, \forall j = 1, \ldots, L$. For the convenience of the reader, in Fig. 1 we show a representation of a general shallow NN and the ChNN, and LeNN that we use to solve the problem presented in this research. It may happen that the activation functions are defined on an inconsistent domain $z \in [z_0, z_f]$ (for instance, Chebyshev and Legendre polynomials are defined on $z \in [-1, +1]$). Thus, our independent variable $t \in [t_0, t_f]$ needs to be mapped in the z domain. In general to have better training performance, regardless to the activation function used, it is convenient to map the independent variable $t \in [t_0, t_f]$ in the z domain (usually $z \in [0, +1]$ or $z \in [-1, +1]$). This can be done using the following linear transformation:

$$z = z_0 + c(t - t_0) \quad \leftrightarrow \quad t = t_0 + \frac{1}{c}(z - z_0),$$

where c is the mapping coefficient

$$c = \frac{z_f - z_0}{t_f - t_0}.$$

By the derivative chain rule, the nth derivative of Eq. (18) is

$$\frac{\mathrm{d}^n g(t)}{\mathrm{d}t^n} = c^n \boldsymbol{\beta}^T \frac{\mathrm{d}^n \boldsymbol{\sigma}(z)}{\mathrm{d}z^n}, \tag{19}$$

which defines all mappings of the free-function. Finally, the domain t must be discretized by M points, and then, the DE reduces to the following:

$$\tilde{F}(\boldsymbol{\beta}) = 0. \tag{20}$$

Equation (20) is an unconstrained optimization problem that can be solved via different optimization schemes. As the problem is linear, rearranging the term in Eq. (20), it reduces to a linear system of algebraic equation: $A\boldsymbol{\beta} = \boldsymbol{b}$. Therefore,

$$\boldsymbol{\beta} = (A^T A)^{-1} A^T \boldsymbol{b}. \tag{21}$$

For the convenience of the reader, a schematic that summarizes how the PINN-TFC based method works for first order linear ODEs is reported in Fig. 2

IV. THERMAL CREEP FLOW IN A PLANE CHANNEL: PROBLEM FORMULATION

Three basic problems defined by flow in a finite plane-parallel channel are considered important when evaluating the effectiveness of a new kinetic model of the linearized Boltzmann equation or a new computational method. These problems are the Couette flow, the Poiseuille flow, and the thermal creep flow, and they are widely studied by Cercignani.[71]

In the Couette flow, a rarefied gas is confined between two parallel walls, one of which moves in its own plane with respect to the other. The Poiseuille flow is a particular case of flow in a plane channel driven by a pressure gradient in a direction parallel to the walls that enclose the rarefied gas. A thermal creep flow, unlike the Poiseuille flow, is driven by a wall temperature gradient. It occurs between two long parallel plates at a very short distance (in the scale of micro/nano-dimension), and the momentum of the flow varies from the cooler to the warmer molecules, until an equilibrium temperature is reached. Only then will the differences in molecular velocities and the shear stress through the wall equal zero.

As explained by Siewert in Ref. 72, the Poiseuille flow and the thermal creep flow have much in common, and their problems present a similar formulation. However, for the purpose of this work, we report only the formulation of the thermal creep problem in BGK approximation. The latter significantly simplifies the mathematics of the problem by replacing the collision integral with a simple relaxation model. It has proved to be a useful qualitative model, because it retains most of the qualitative characteristics of the Boltzmann equation. The validity of the BGK model has been evaluated by comparing its solution with the Boltzmann solutions for several relaxation problems,[73] showing satisfactory accuracy and history of the velocity distribution function during the relaxation, when the flow is not far from a Maxwellian distribution. Also, there is no direct connection with the relaxation problem to the Knudsen number, thus the BGK model can be applied to flow of highly rarefied gas, as long as degree of non-equilibrium is not strong in velocity distribution. One of its main disadvantages is that it uses a Prandtl number $Pr = 1$, while for a monatomic gas a value around 2/3 would be more appropriate.[10,74–76]

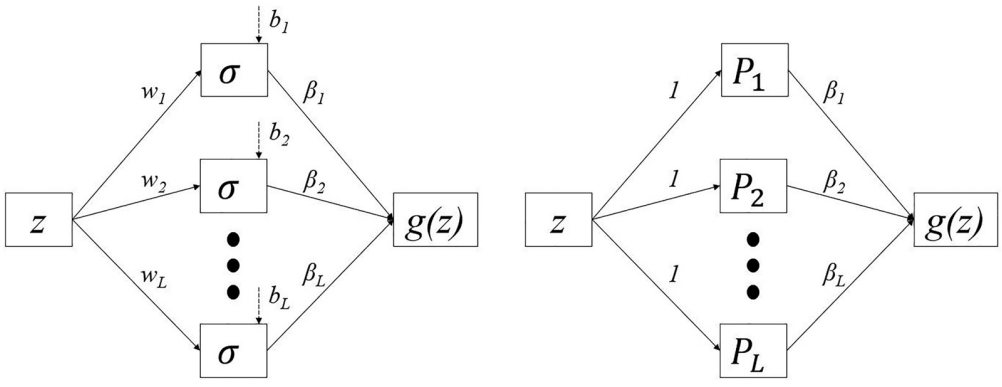


FIG. 1. Standard shallow neural network (left) and shallow Chebyshev/Legendre neural network (right).

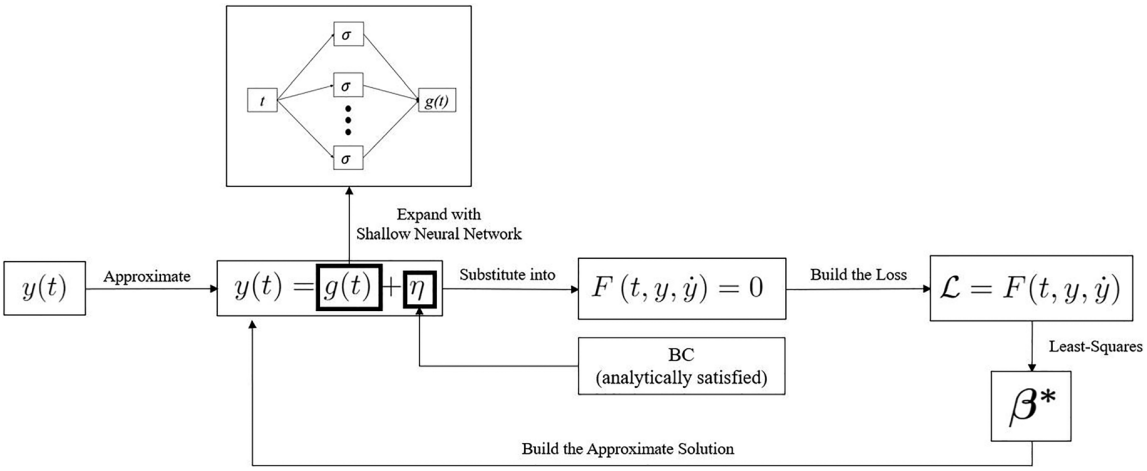


FIG. 2. Schematic of the general PINN-TFC based framework to solve linear ODEs with one constraint in one point.

Thus, the BGK model cannot match the viscosity and the thermal conductivity of a real gas simultaneously. Therefore, when heat transfer is involved in the flow calculations, particular precautions must be taken. To bypass this problem, many modified models. such as the ellipsoidal-statistical BGK model[77] and the Shakhov model[15] have been proposed. In this paper, we consider the creep flow in the 1D micro channel shown in Fig. 3.

The full non-linear Boltzmann equation for the gas atom space and velocity distribution function $f(\mathbf{r},\mathbf{v})$ with the collision operator J can be written as

$$\mathbf{v}\cdot\nabla_r f(\mathbf{r},\mathbf{v}) = J(f,f'), \tag{22}$$

and by following the formulation proposed by Williams,[78] we can introduce a perturbation and express f as

$$f(\mathbf{r},\mathbf{v}) = f_0(\mathbf{r},\mathbf{v})[1+h(\mathbf{r},\mathbf{v})], \tag{23}$$

where h is the noise caused by the presence of the walls to the local Maxwellian $f_0(\mathbf{r},\mathbf{v})$, which has the following form:

$$f_0(\mathbf{r},\mathbf{v}) = n_\infty(x,z)\left[\frac{m}{2\pi k T_\infty(x,z)}\right]^{3/2} \times \exp\left[-\frac{m}{2\pi k T_\infty(x,z)}\left\{v_x^2+v_y^2+[v_z-\bar{u}(x)]^2\right\}\right], \tag{24}$$

where,

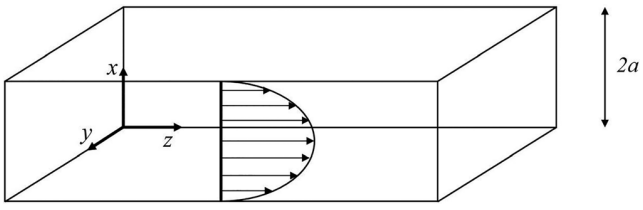


FIG. 3. Creep microchannel flow.

$$n_\infty(x,z) = n_0(1+R_x x + R_z z),$$
$$T_\infty(x,z) = T_0(1+K_x x + K_z z),$$
$$\bar{u}(x) = Kx,$$

here, R_i and K_i are the relative density and temperature gradients in the ith direction, respectively. Let's consider the non-dimension velocity

$$\mathbf{c} = \mathbf{v}\left(\frac{m}{2kT_0}\right)^{1/2}.$$

Now, if we substitute the Eq. (23) into Eq. (22), we obtain the following:

$$c_x\left[\left(c^2-\frac{3}{2}\right)K_x + R_x + 2c_z K_0\right] + c_z\left(c^2-\frac{3}{2}\right)K_x + c_z R_z + c_x\frac{\partial h(x,\mathbf{c})}{\partial x} + V(c)h(x,\mathbf{c}) = \int K(\mathbf{c},\mathbf{c}')h(x,\mathbf{c}')d\mathbf{c}', \tag{25}$$

with

$$K_0 = K\left(\frac{m}{2kT_0}\right)^{1/2}, \quad V(c) = K\left(\frac{m}{2kT_0}\right)^{1/2}V(v),$$

where $V(v)$ represents the collision frequency of the atoms. Now we can take use of the classical BGK model, which assumes that the collision frequency $V(c)=\lambda$ is constant and independent of velocity. Usually, this value is chosen to be equal to the viscosity of the modeled gas.[79] Thus, the scattering kernel of Eq. (25) is

$$K(\mathbf{c},\mathbf{c}') = \frac{\lambda_0}{\pi^{3/2}}\left[1+2\mathbf{c}\cdot\mathbf{c}'+\frac{2}{3}\left(c^2-\frac{3}{2}\right)\left(c'^2-\frac{3}{2}\right)\right], \tag{26}$$

where,

$$\lambda_0 = \left(\frac{m}{2kT_0}\right)^{1/2}\lambda.$$

Then, Eq. (25) becomes

Phys. Fluids **33**, 047110 (2021); doi: 10.1063/5.0046181

$$c_x\left[\left(c^2-\frac{3}{2}\right)K_x+R_x+2c_zK_0\right]+c_z\left(c^2-\frac{3}{2}\right)K_x+c_zR_z+c_x\frac{\partial h(x,\mathbf{c})}{\partial x}+\lambda_0 h(x,\mathbf{c})=\int K(\mathbf{c},\mathbf{c}')h(x,\mathbf{c}')d\mathbf{c}'. \tag{27}$$

By considering the upper and lower channel plates at $x=-a$ and a, the most general form of the gas-wall boundary condition at the plates is

$$-c_x f(\pm a,\mathbf{c})=\int_{\substack{c_x'<0\\ c_x'>0}} c_x'\Gamma(\mathbf{c},\mathbf{c}')f(\pm a,\mathbf{c}')d\mathbf{c}'. \tag{28}$$

More details regarding the derivation of Eq. (27) are reported in Ref. 78. We assume a local Maxwellian that characterizes a diffusely and specularly reflecting wall

$$F(\mathbf{v},T_w,u_w)=N\left(\frac{m}{2\pi kT_w}\right)^{3/2}\exp\left[-\frac{m}{2kT_w}\left\{v_x^2+v_y^2+(v_z-u_w)^2\right\}\right], \tag{29}$$

where N is the molecular density and T_w and u_w are the temperature and the velocity at the wall, respectively. By using the perturbation of Eq. (23) with F instead of f_0 into Eq. (28) gives

$$h(\pm a,\mathbf{c})=\alpha\left[2c_zu_w+\delta(c^2-2)\right]+(1-\alpha)h(\pm a,\mathbf{c}(-c_x,c_y,c_z))+\frac{2\alpha}{\pi}\int e^{-c'^2}h(x\pm a,\mathbf{c}')d\mathbf{c}', \tag{30}$$

for $c_x<0$ and $c_x>0$, where $\delta=\frac{T_w-T_0}{T_0}$. The accommodation coefficient $\alpha\in(0,1]$ represents the fraction of diffuse reflection.

Following the formulation proposed by Ganapol,[54] the problem to be solved is the following:

$$c_xK_0+\frac{1}{2}R_z+\frac{1}{2}K_z\left(c_x^2+\frac{3}{2}\right)+c_x\frac{\partial}{\partial x}Z(x,c_x)+\lambda_0Z(x,c_x)=\lambda_0\pi^{-1/2}\int e^{-c_x^2}Z(x,c_x)dc_x, \tag{31}$$

for $x\in[-a,a]$ and $c_x\in(-\infty,\infty)$ and subject to

$$\begin{cases} Z(-a,c_x)=\alpha u_w+(1-\alpha)Z(-a,-c_x),\\ Z(a,-c_x)=\alpha u_w+(1-\alpha)Z(a,c_x), \end{cases} \tag{32}$$

for $c_x\in(0,\infty)$, where a is the channel half-thickness, R_z and K_z are gradients in the flow direction z, K is the scattering kernel, K_0 is proportional to K, λ_0 is proportional to the frequency of collisions between the atoms, x is the spatial variable, $\alpha\in(0,1]$ is the accommodation coefficient, and the moment $Z(x,c_x)$ is

$$Z(x,c_x)=\pi^{-1}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}e^{-(c_y^2+c_z^2)}c_zh(x,c_x,c_y,c_z)dc_ydc_z, \tag{33}$$

where (c_x,c_y,c_z) are the three components of the molecular velocity and h is a perturbation from Maxwell distribution, according to Cercignani.[80]

For our case of interest, the creep flow, we can make some assumptions, such as K_0 and u_w are equal to 0 (no-slip velocity condition), and $R_z=-K_z$, obtaining

$$\frac{1}{2}K_z\left(u^2-\frac{1}{2}\right)+u\frac{\partial}{\partial x}Z(x,u)+\lambda_0Z(x,u)=\lambda_0\pi^{-1/2}\int_{-\infty}^{\infty}e^{-u^2}Z(x,u)du, \tag{34}$$

where u is set to c_x, subject to

$$\begin{cases} Z(-a,u)=(1-\alpha)Z(-a,-u),\\ Z(a,-u)=(1-\alpha)Z(a,u), \end{cases} \tag{35}$$

for $u\in(0,\infty)$. By using

$$Z(x,u)=Y(x,u)-\frac{K_z}{2\lambda_0}\left(u^2-\frac{1}{2}\right), \tag{36}$$

and plugging it into (34) and (35), we find that $Y(\tau,u)$ satisfies the following linear integrodifferential equation:

$$u\frac{\partial}{\partial\tau}Y(\tau,u)+Y(\tau,u)=\int_{-\infty}^{\infty}\Psi(u)Y(\tau,u)du, \tag{37}$$

where $\tau\in(0,a)$ replaces x by taking advantage of spatial symmetry, and $u\in(-\infty,\infty)$; subject to the following boundary conditions:

$$\begin{cases} Y(0,u)=Y(0,-u),\\ Y(a,-u)=(1-\alpha)Y(a,u)+g(u), \end{cases} \tag{38}$$

for $u\in(0,\infty)$, where $g(u)=\frac{\alpha}{2}\left(u^2-\frac{1}{2}\right)$. Also, we used the normalization $\frac{K_z}{\lambda_0}\equiv 1$, and $\Psi(u)$ is a weight function defined as follows:

$$\Psi(u)=\pi^{-1/2}e^{-u^2}. \tag{39}$$

The problem (37) and (38) is the one to be solved via the PINN TFC-based method.

V. PINN TFC-BASED FORMULATION OF THE PROBLEM

As we deal with a Boundary Value Problem (BVP), according to (10), $y(\tau)$ can be expressed as following:

$$y(\tau)=g(\tau)+\eta\,p(\tau). \tag{40}$$

We choose $p(\tau)=1$, as suggested by Ref. 58, and the free function $g(\tau)$ as a shallow NN

$$g(\tau)=\boldsymbol{\sigma}^T\boldsymbol{\beta}. \tag{41}$$

We map the independent variable τ to a new independent variable x that ranges in $[-1,1]$. The new independent variable x is defined as $x=c(\tau+a)-1$, where c is the following mapping constant:

$$c=\frac{x_f-x_0}{\tau_f-\tau_0}=\frac{1}{a}.$$

Considering the new independent variable, we have

$$\begin{cases} Y(\tau,u)=Y(x,u),\\ \frac{\partial}{\partial\tau}Y(\tau,u)=c\frac{\partial}{\partial x}Y(x,u). \end{cases} \tag{42}$$

Then the problem becomes

$$cu\frac{\partial}{\partial x}Y(x,u)+Y(x,u)=\int_{-\infty}^{\infty}\Psi(u)Y(x,u)du. \tag{43}$$

In order to use a Gauss–Legendre quadrature in the range $[-1,1]$, we can use a further change of variable

$$u = -\log(|\mu|), \quad du = -\frac{1}{\mu}d\mu, \quad \Psi(\mu) = \pi^{-\frac{1}{2}}e^{-\left(-\log(|\mu|)\right)^2},$$

where $\mu \in [-1; 1]$. We can then rewrite the problem in the following form:

$$-c\log(\mu)\frac{\partial}{\partial x}Y(x,\mu) + Y(x,\mu) = \int_{-1}^{1} -\frac{1}{\mu}\Psi(\mu)Y(x,\mu)d\mu. \quad (44)$$

Via discretizing the velocity μ into N points in the interval $(0, 1]$ according to the Gauss–Legendre quadrature scheme we have

$$\mu \quad \rightarrow \quad \boldsymbol{\mu} = \{\mu_i\}_{i=1}^{N},$$

it follows that:

$$Y(x,\mu) \quad \rightarrow \quad \mathbf{Y}(x) = \{Y(x,\mu_i)\}_{i=1}^{N}. \quad (45)$$

The problem can be split for both positive and negative molecular velocity following Ref. 54, where the Gauss–Legendre quadrature rule is used to evaluate the integral in the range $[0, 1]$:

$$\begin{aligned} &-c\log(\mu_i)\frac{\partial}{\partial x}Y(x,\mu_i) + Y(x,\mu_i) \\ &\quad = \sum_{k=1}^{N} w_k \frac{1}{\mu_k}\Psi(\mu_k)[Y(x,\mu_k) + Y(x,-\mu_k)], \quad (46) \\ &c\log(\mu_i)\frac{\partial}{\partial x}Y(x,-\mu_i) + Y(x,-\mu_i) \\ &\quad = \sum_{k=1}^{N} w_k \frac{1}{\mu_k}\Psi(\mu_k)[Y(x,\mu_k) + Y(x,-\mu_k)], \quad (47) \end{aligned}$$

subject to:

$$\begin{cases} Y(0,\mu_i) = Y(0,-\mu_i), \\ Y(a,-\mu_i) = (1-\alpha)Y(a,\mu_i) + g_i, \end{cases} \quad (48)$$

where $g_i = \frac{\alpha}{2}\left(\log^2\mu_i - \frac{1}{2}\right)$. To simplify the notation, we suppress the x and μ dependency and write the terms for the positive and negative molecular velocity as Y^+ and Y^-, respectively. Thus, our problem becomes:

$$-c\log(\mu_i)\frac{\partial}{\partial x}Y_i^+ + Y_i^+ = \sum_{k=1}^{N} w_k \frac{1}{\mu_k}\Psi(\mu_k)\left(Y_k^+ + Y_k^-\right), \quad (49)$$

$$c\log(\mu_i)\frac{\partial}{\partial x}Y_i^- + Y_i^- = \sum_{k=1}^{N} w_k \frac{1}{\mu_k}\Psi(\mu_k)\left(Y_k^+ + Y_k^-\right), \quad (50)$$

subject to

$$\begin{cases} Y_0^+ = Y_0^-, \\ Y_f^- = (1-\alpha)Y_f^+ + g_i. \end{cases} \quad (51)$$

Our constrained expressions, for positive and negative terms, are

$$Y^{\pm}(x) = g^{\pm}(x) + \eta^{\pm}, \quad (52)$$

and according to the boundary conditions, we have

$$\begin{aligned} Y_0^+ &= \boldsymbol{\sigma_0^T}\boldsymbol{\beta}^+ + \eta^+, \qquad Y_f^- = \boldsymbol{\sigma_f^T}\boldsymbol{\beta}^- + \eta^-, \\ Y_f^+ &= \boldsymbol{\sigma_f^T}\boldsymbol{\beta}^+ + \eta^+, \qquad Y_0^- = \boldsymbol{\sigma_0^T}\boldsymbol{\beta}^- + \eta^-. \end{aligned}$$

Replacing them in the previous system of equations, we obtain

$$\begin{cases} \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^+} + \eta_i^+ = \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^-} + \eta_i^-, \\ \boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^-} + \eta_i^- = (1-\alpha)\boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^+} + (1-\alpha)\eta_i^+ + g_i, \end{cases}$$

and then

$$\begin{cases} \eta_i^+ - \eta_i^- = \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^-} - \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^+}, \\ (1-\alpha)\eta_i^+ - \eta_i^- = \boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^-} - (1-\alpha)\boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^+} - g_i, \end{cases}$$

where in matrix form is

$$\begin{bmatrix} 1 & -1 \\ 1-\alpha & -1 \end{bmatrix}\begin{bmatrix} \eta_i^+ \\ \eta_i^- \end{bmatrix} = \begin{bmatrix} \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^-} - \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^+} \\ \boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^-} - (1-\alpha)\boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^+} - g_i \end{bmatrix}.$$

Keeping the ηs vector on the LHS, we get

$$\begin{bmatrix} \eta_i^+ \\ \eta_i^- \end{bmatrix} = -\frac{1}{\alpha}\begin{bmatrix} -1 & 1 \\ -(1-\alpha) & 1 \end{bmatrix}\begin{bmatrix} \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^-} - \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^+} \\ \boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^-} - (1-\alpha)\boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^+} - g_i \end{bmatrix},$$

that is,

$$\begin{bmatrix} \eta_i^+ \\ \eta_i^- \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha} & -\frac{1}{\alpha} \\ \frac{1-\alpha}{\alpha} & -\frac{1}{\alpha} \end{bmatrix}\begin{bmatrix} \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^-} - \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^+} \\ \boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^-} - (1-\alpha)\boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^+} - g_i \end{bmatrix}.$$

For the sake of simplicity, we introduce new parameters

$$\varphi = \frac{1-\alpha}{\alpha} \quad \text{and} \quad \theta = \frac{1}{\alpha},$$

and the system of equations becomes

$$\begin{bmatrix} \eta_i^+ \\ \eta_i^- \end{bmatrix} = \begin{bmatrix} \theta & -\theta \\ \varphi & -\theta \end{bmatrix}\begin{bmatrix} \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^-} - \boldsymbol{\sigma_0^T}\boldsymbol{\beta_i^+} \\ \boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^-} - (1-\alpha)\boldsymbol{\sigma_f^T}\boldsymbol{\beta_i^+} - g_i \end{bmatrix},$$

and thus

$$\begin{cases} \eta_i^+ = \left(\boldsymbol{\varphi\sigma_f} - \boldsymbol{\theta\sigma_0}\right)^T\boldsymbol{\beta_i^+} + \boldsymbol{\theta}(\boldsymbol{\sigma_0} - \boldsymbol{\sigma_f})^T\boldsymbol{\beta_i^-} + \theta g_i, \\ \eta_i^- = \boldsymbol{\varphi}(\boldsymbol{\sigma_f} - \boldsymbol{\sigma_0})^T\boldsymbol{\beta_i^+} + \left(\boldsymbol{\varphi\sigma_0} - \boldsymbol{\theta\sigma_f}\right)^T\boldsymbol{\beta_i^-} + \theta g_i. \end{cases} \quad (53)$$

Replacing them in the constrained expressions, we get

$$\begin{aligned} Y_i^+ &= \left(\boldsymbol{\sigma} + \boldsymbol{\varphi\sigma_f} - \boldsymbol{\theta\sigma_0}\right)^T\boldsymbol{\beta_i^+} + \boldsymbol{\theta}(\boldsymbol{\sigma_0} - \boldsymbol{\sigma_f})^T\boldsymbol{\beta_i^-} + \theta g_i, \\ Y_i^- &= \boldsymbol{\varphi}(\boldsymbol{\sigma_f} - \boldsymbol{\sigma_0})^T\boldsymbol{\beta_i^+} + \left(\boldsymbol{\sigma} + \boldsymbol{\varphi\sigma_0} - \boldsymbol{\theta\sigma_f}\right)^T\boldsymbol{\beta_i^-} + \theta g_i, \end{aligned} \quad (54)$$

and replacing the constrained expressions in the equations of our problem, we obtain

$$\begin{aligned} &\left(-c\log(\mu_i)\boldsymbol{\sigma'} + \boldsymbol{\sigma} + \varphi\boldsymbol{\sigma_f} - \theta\boldsymbol{\sigma_0}\right)^T\boldsymbol{\beta_i^+} + \theta(\boldsymbol{\sigma_0} - \boldsymbol{\sigma_f})^T\boldsymbol{\beta_i^-} \\ &\quad -\sum_{k=1}^{N} w_k \frac{1}{\mu_k}\Psi(\mu_k)\Big[\left(\boldsymbol{\sigma} + 2\varphi\boldsymbol{\sigma_f} - \theta\varphi\boldsymbol{\sigma_0}\right)^T\boldsymbol{\beta_k^+} \\ &\quad +\left(\boldsymbol{\sigma} + \theta\varphi\boldsymbol{\sigma_0} - 2\theta\boldsymbol{\sigma_f}\right)^T\boldsymbol{\beta_k^-}\Big] = -\theta g_i + \sum_{k=1}^{N} 2w_k\Psi_k\theta g_k, \end{aligned} \quad (55)$$

Phys. Fluids **33**, 047110 (2021); doi: 10.1063/5.0046181

$$\varphi(\boldsymbol{\sigma_f}-\boldsymbol{\sigma_0})^T\boldsymbol{\beta_i^+}+\left(c\log(\mu_i)\boldsymbol{\sigma'}+\boldsymbol{\sigma}+\varphi\boldsymbol{\sigma_0}-\theta\boldsymbol{\sigma_f}\right)\boldsymbol{\beta_i^-}$$
$$-\sum_{k=1}^{N}w_k\frac{1}{\mu_k}\Psi(\mu_k)\Big[\left(\boldsymbol{\sigma}+2\varphi\boldsymbol{\sigma_f}-\theta\varphi\boldsymbol{\sigma_0}\right)^T\boldsymbol{\beta_k^+}$$
$$+\left(\boldsymbol{\sigma}+\theta\varphi\boldsymbol{\sigma_0}-2\theta\boldsymbol{\sigma_f}\right)^T\boldsymbol{\beta_k^-}\Big]=-\theta g_i+\sum_{k=1}^{N}2w_k\Psi_k\theta g_k. \quad (56)$$

For the sake of simplicity, we write the inhomogeneous terms as

$$b_i^+=b_i^-=-\theta g_i+\sum_{k=1}^{N}2w_k\Psi_k\theta g_k.$$

Expanding the summations, we get the following matrix form:

$$\begin{bmatrix}\blacklozenge_k+\bigstar_i & \heartsuit_k+\clubsuit_i\\ \blacklozenge_k+\spadesuit_i & \heartsuit_k+\blacktriangledown_i\end{bmatrix}\begin{bmatrix}\beta_i^+\\ \beta_i^-\end{bmatrix}=\begin{bmatrix}b_i^+\\ b_i^-\end{bmatrix} \quad (57)$$

where,

$$\bigstar_i=-c\log(\mu_i)\boldsymbol{\sigma'}^T+\boldsymbol{\sigma}^T+\varphi\boldsymbol{\sigma_f}^T-\theta\boldsymbol{\sigma_0}^T$$
$$\blacktriangledown_i=c\log(\mu_i)\boldsymbol{\sigma'}^T+\boldsymbol{\sigma}^T+\varphi\boldsymbol{\sigma_0}^T-\theta\boldsymbol{\sigma_f}^T$$
$$\spadesuit_k=\varphi(\boldsymbol{\sigma_f}-\boldsymbol{\sigma_0})^T$$
$$\clubsuit_k=\theta(\boldsymbol{\sigma_0}-\boldsymbol{\sigma_f})^T$$
$$\blacklozenge_k=-w_k\frac{1}{\mu_k}\Psi(\mu_k)(\boldsymbol{\sigma}+2\varphi\boldsymbol{\sigma_f}-\theta\varphi\boldsymbol{\sigma_0})^T$$
$$\heartsuit_k=-w_k\frac{1}{\mu_k}\Psi(\mu_k)(\boldsymbol{\sigma}+\theta\varphi\boldsymbol{\sigma_0}-2\theta\boldsymbol{\sigma_f})^T$$

Thus, the problem reduce to the system in Fig. 4, which is a problem of the type

$$\boldsymbol{A\beta}=\boldsymbol{B}, \quad (58)$$

whose the dimensions are

$$\boldsymbol{\beta^{\pm}}_{i\,[L\times1]} \quad \boldsymbol{\beta}_{[2LN\times1]}$$
$$\boldsymbol{b^{\pm}}_{i\,[M\times1]} \quad \boldsymbol{B}_{[2MN\times1]}$$
$$\bigstar_{i\,[M\times L]} \quad (\text{same dimension for the other terms of } \boldsymbol{A})$$
$$\boldsymbol{A}_{[(2MN)\times(2LN)]},$$

where M is the discretization order for τ, and L is the number of neurons of the shallow NN.

That is, the original BVP has been reformulated as an unconstrained optimization problem. In this case, according to the ELM algorithm, the output weights of the network $\boldsymbol{\beta}$'s are calculated with only one direct pass via least squares methods, as the original problem is linear.[58] That is,

$$\boldsymbol{\beta}=(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{B}. \quad (59)$$

As can be seen from Eq. (59) the unknowns are computed using a classic least-square method. The authors have tried all the least squares methods suggested in Appendix A of Ref. 59, finding the classic one the most convenient in terms of accuracy and computational time. Once the output weights $\boldsymbol{\beta}$'s are computed, the final solutions are built according to (52). To be noticed that, although the output weights are evaluated on specific values of τ (e.g., training points), once the $\boldsymbol{\beta}$'s are known the solution can be computed in any desired query points without further manipulation (e.g., post-processing or interpolation). That is, once the solution is learned on the training points, we have an analytical representation (by NNs) of it. Therefore, it can be evaluated everywhere else in the channel with no additional computational efforts. For example, we do not have to retrain the network and/or use a separate interpolation routine. We simply evaluate the solution learned on the training points in the desired new point in the channel.

For the convenience of the reader, the all process described in this section is summarized in the schematic of Fig. 5.

VI. RESULTS AND DISCUSSIONS

To generate the results, we coded the problem in the Matlab R2019a platform and ran all our simulations with an Intel Core i7–9700 CPU PC with 64 GB of RAM. To demonstrate the accuracy of the proposed PINN TFC-based method in solving the thermal creep problem, we report the macroscopic velocity profile in a inverse Knudsen number range of $[0.05, 9.00]$, that according to Refs. 20, 54, and 81 is given by

$$q(\tau)=\int_{-\infty}^{\infty}\Psi(\mu)Y(\tau,\mu)d\mu=Y_0(\tau) \quad \text{for} \quad \tau\in[0,a], \quad (60)$$

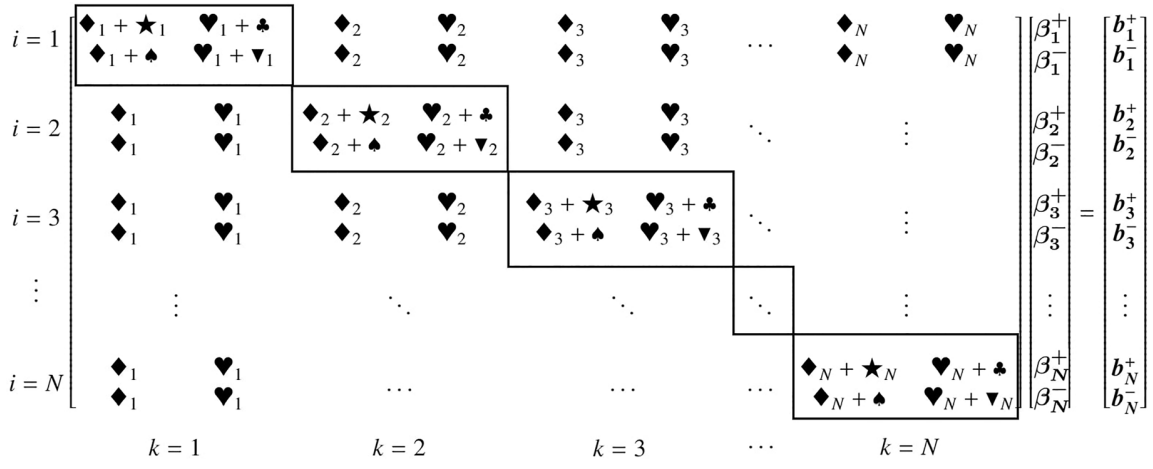


FIG. 4. Linear system solved via least-squares.

Phys. Fluids 33, 047110 (2021); doi: 10.1063/5.0046181

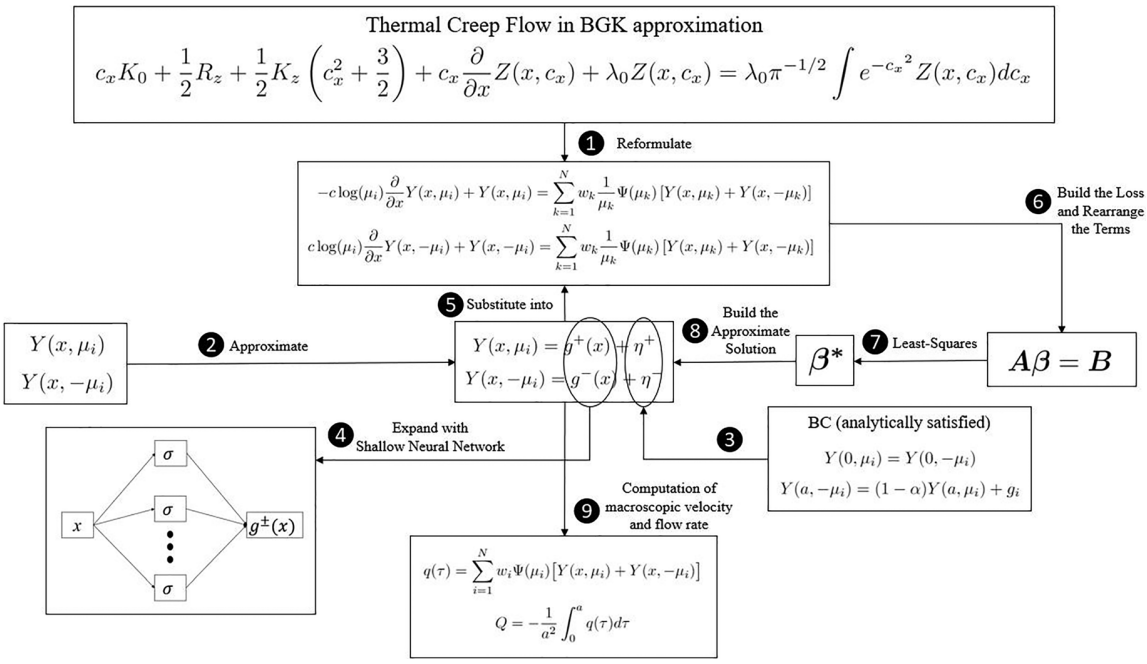


FIG. 5. Schematic of the PINN TFC-based algorithm to solve thermal creep flow problem in BGK approximation.

where $Y_0(\tau)$ is given by[81]

$$Y_0(\tau) = \int_{-\infty}^{\infty} \Psi(\mu) Y(\tau,\mu) d\mu, \tag{61}$$

which it can be computed as

$$Y_0 = \sum_{i=1}^{N} w_i \Psi(\mu_i)\left(Y_i^+ + Y_i^-\right). \tag{62}$$

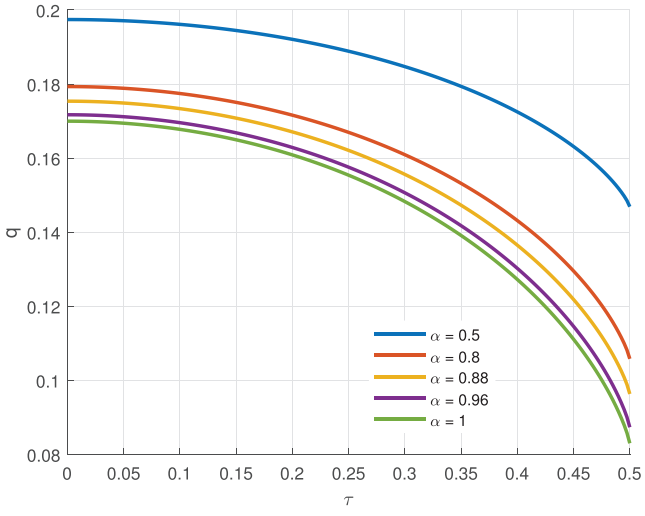


FIG. 6. Macroscopic velocity profiles in a half channel for different values of α.

Figure 6 shows the qualitative behavior of the macroscopic velocity in a half channel for different values of α, computed with all the three different types of NNs tested. The accommodation coefficient quantifies the interaction between a fluid and a solid (in this case the wall). It indicates the degree to which molecules are accommodated to the surface.[82] Thus, the macroscopic velocity increases for increasing accommodation coefficient α. Hence, Fig. 6 proves that all the NNs tested are able to capture the qualitative behavior of the macroscopic velocity.

In Table I we reported the values of the macroscopic velocity profile q, at $\tau = 0$ and $\tau = a$, with accommodation coefficient $\alpha = 1$. The results were computed with different type of shallow NNs as free-function: standard shallow NNs (using hyperbolic tangent as activation function), ChNN, and LeNN (we reported in bold the digits that differ from the benchmarks published by Ganapol[54]). The results we obtained using the following parameters: $N = 30$, $M = 200$, 20 neurons for the standard shallow NN, and 74 for ChNN and LeNN. The training time was on the order of seconds for all the networks. Although we achieved good results (both qualitatively and quantitatively) with all the networks, with ChNN and LeNN we were able to match all the digits of the benchmarks published by Ganapol.[54]

TABLE I. Macroscopic velocity profile q computed with different type of shallow NNs as free-function, for a channel half-thickness $a = 0.5$.

Neural network	$q(\tau = 0)$	$q(\tau = a)$
Standard	$1.974\,216\,9 \times 10^{-1}$	$1.468\,959\,0 \times 10^{-1}$
ChNN	$1.974\,209\,2 \times 10^{-1}$	$1.468\,959\,4 \times 10^{-1}$
LeNN	$1.974\,209\,2 \times 10^{-1}$	$1.468\,959\,4 \times 10^{-1}$

Phys. Fluids **33**, 047110 (2021); doi: 10.1063/5.0046181

In Tables II and III, the results computed with ChNN and LeNN, for different values of the accommodation number at center of channel $\tau = 0$ and at the wall $\tau = a$, are reported. All our digits match with the benchmarks reported in Ref. 54. Furthermore, we computed the flow rate

$$Q = -\frac{1}{2a^2}\int_{-a}^{a} q(\tau)d\tau = -\frac{1}{a^2}\int_{0}^{a} Y_0(\tau)d\tau, \tag{63}$$

where the results, computed with ChNN and LeNN, are reported in Table IV. The chosen parameters and the computational time for each

TABLE II. The macroscopic velocity profile $q(a, \alpha)$ for $\tau = 0$. All the digits match the benchmark published by Ganapol.[54]

$2a$	$\alpha = 0.50$	$\alpha = 0.80$	$\alpha = 0.88$	$\alpha = 0.96$	$\alpha = 1.00$
0.05	$4.222\,500\,3 \times 10^{-2}$	$2.822\,852\,1 \times 10^{-2}$	$2.572\,389\,8 \times 10^{-2}$	$2.352\,962\,1 \times 10^{-2}$	$2.252\,976\,7 \times 10^{-2}$
0.1	$6.509\,553\,8 \times 10^{-2}$	$4.571\,936\,2 \times 10^{-2}$	$4.214\,646\,6 \times 10^{-2}$	$3.898\,974\,2 \times 10^{-2}$	$3.754\,310\,5 \times 10^{-2}$
0.3	$1.187\,663\,9 \times 10^{-1}$	$9.267\,451\,1 \times 10^{-2}$	$8.752\,494\,4e \times 10^{-2}$	$8.288\,705\,5 \times 10^{-2}$	$8.073\,408\,3 \times 10^{-2}$
0.5	$1.504\,850\,8 \times 10^{-1}$	$1.248\,201\,3 \times 10^{-1}$	$1.195\,421\,8 \times 10^{-1}$	$1.147\,314\,3 \times 10^{-1}$	$1.124\,801\,4 \times 10^{-1}$
0.7	$1.729\,512\,0 \times 10^{-1}$	$1.498\,255\,6 \times 10^{-1}$	$1.449\,292\,7 \times 10^{-1}$	$1.404\,273\,7 \times 10^{-1}$	$1.383\,081\,9 \times 10^{-1}$
0.9	$1.901\,655\,4 \times 10^{-1}$	$1.703\,262\,7 \times 10^{-1}$	$1.660\,349\,2 \times 10^{-1}$	$1.620\,635\,1 \times 10^{-1}$	$1.601\,858\,0 \times 10^{-1}$
1.0	$1.974\,209\,2 \times 10^{-1}$	$1.793\,256\,3 \times 10^{-1}$	$1.753\,784\,8 \times 10^{-1}$	$1.717\,162\,8 \times 10^{-1}$	$1.699\,817\,9 \times 10^{-1}$
2.0	$2.439\,083\,9 \times 10^{-1}$	$2.420\,532\,0 \times 10^{-1}$	$2.417\,017\,5 \times 10^{-1}$	$2.413\,990\,6 \times 10^{-1}$	$2.412\,644\,8 \times 10^{-1}$
5.0	$2.931\,138\,2 \times 10^{-1}$	$3.168\,530\,9 \times 10^{-1}$	$3.229\,393\,6 \times 10^{-1}$	$3.289\,278\,1 \times 10^{-1}$	$3.318\,861\,4 \times 10^{-1}$
7.0	$3.047\,934\,0 \times 10^{-1}$	$3.354\,423\,3 \times 10^{-1}$	$3.433\,559\,0 \times 10^{-1}$	$3.511\,647\,6 \times 10^{-1}$	$3.550\,306\,2 \times 10^{-1}$
9.0	$3.108\,636\,2 \times 10^{-1}$	$3.451\,505\,4 \times 10^{-1}$	$3.540\,327\,0 \times 10^{-1}$	$3.628\,093\,0 \times 10^{-1}$	$3.671\,587\,0 \times 10^{-1}$

TABLE III. The macroscopic velocity profile $q(a, \alpha)$ for $\tau = a$. All the digits match the benchmark published by Ganapol.[54]

$2a$	$\alpha = 0.50$	$\alpha = 0.80$	$\alpha = 0.88$	$\alpha = 0.96$	$\alpha = 1.00$
0.05	$3.911\,603\,2 \times 10^{-2}$	$2.390\,993\,9 \times 10^{-2}$	$2.110\,893\,4 \times 10^{-2}$	$1.862\,664\,4 \times 10^{-2}$	$1.748\,562\,7 \times 10^{-2}$
0.1	$5.880\,243\,5 \times 10^{-2}$	$3.700\,835\,1 \times 10^{-2}$	$3.284\,860\,8 \times 10^{-2}$	$2.912\,348\,4 \times 10^{-2}$	$2.739\,890\,6 \times 10^{-2}$
0.3	$1.004\,132\,6 \times 10^{-1}$	$6.702\,653\,1 \times 10^{-2}$	$6.013\,395\,6 \times 10^{-2}$	$5.381\,871\,5 \times 10^{-2}$	$5.084\,945\,4 \times 10^{-2}$
0.5	$1.214\,816\,5 \times 10^{-1}$	$8.373\,539\,5 \times 10^{-2}$	$7.557\,561\,3 \times 10^{-2}$	$6.799\,631\,4 \times 10^{-2}$	$6.439\,969\,8 \times 10^{-2}$
0.7	$1.345\,641\,9 \times 10^{-1}$	$9.479\,089\,9 \times 10^{-2}$	$8.591\,182\,3 \times 10^{-2}$	$7.758\,311\,5 \times 10^{-2}$	$7.360\,456\,6 \times 10^{-2}$
0.9	$1.434\,574\,8 \times 10^{-1}$	$1.026\,763\,3 \times 10^{-1}$	$9.335\,172\,6 \times 10^{-2}$	$8.453\,946\,8 \times 10^{-2}$	$8.030\,847\,2 \times 10^{-2}$
1.0	$1.468\,959\,4 \times 10^{-1}$	$1.058\,190\,1 \times 10^{-1}$	$9.633\,441\,4 \times 10^{-2}$	$8.734\,303\,1 \times 10^{-2}$	$8.301\,685\,5 \times 10^{-2}$
2.0	$1.643\,019\,4 \times 10^{-1}$	$1.227\,565\,4 \times 10^{-1}$	$1.126\,192\,6 \times 10^{-1}$	$1.028\,313\,2 \times 10^{-1}$	$9.806\,182\,8 \times 10^{-2}$
5.0	$1.737\,460\,7 \times 10^{-1}$	$1.330\,060\,5 \times 10^{-1}$	$1.227\,130\,6 \times 10^{-1}$	$1.126\,479\,4 \times 10^{-1}$	$1.076\,988\,9 \times 10^{-1}$
7.0	$1.746\,701\,8 \times 10^{-1}$	$1.340\,824\,0 \times 10^{-1}$	$1.237\,910\,6 \times 10^{-1}$	$1.137\,133\,7 \times 10^{-1}$	$1.087\,530\,3 \times 10^{-1}$
9.0	$1.749\,563\,0 \times 10^{-1}$	$1.344\,208\,9 \times 10^{-1}$	$1.241\,314\,3 \times 10^{-1}$	$1.140\,511\,0 \times 10^{-1}$	$1.090\,878\,3 \times 10^{-1}$

TABLE IV. The flow rate $Q(a, \alpha)$. All the digits match the benchmark published by Ganapol.[54]

$2a$	$\alpha = 0.50$	$\alpha = 0.80$	$\alpha = 0.88$	$\alpha = 0.96$	$\alpha = 1.00$
0.05	$-1.653\,688\,8$	$-1.080\,865\,1$	$-9.775\,525\,3 \times 10^{-1}$	$-8.867\,589\,5 \times 10^{-1}$	$-8.452\,892\,6 \times 10^{-1}$
0.1	$-1.266\,441\,6$	$-8.659\,804\,7 \times 10^{-1}$	$-7.914\,281\,9 \times 10^{-1}$	$-7.253\,123\,2 \times 10^{-1}$	$-6.949\,271\,6 \times 10^{-1}$
0.3	$-7.580\,823\,6 \times 10^{-1}$	$-5.712\,072\,1 \times 10^{-1}$	$-5.338\,214\,3 \times 10^{-1}$	$-4.999\,735\,7 \times 10^{-1}$	$-4.841\,992\,5 \times 10^{-1}$
0.5	$-5.705\,722\,9 \times 10^{-1}$	$-4.551\,663\,9 \times 10^{-1}$	$-4.310\,382\,9 \times 10^{-1}$	$-4.089\,057\,3 \times 10^{-1}$	$-3.984\,992\,8 \times 10^{-1}$
0.7	$-4.649\,665\,6 \times 10^{-1}$	$-3.864\,581\,3 \times 10^{-1}$	$-3.695\,099\,4 \times 10^{-1}$	$-3.538\,098\,0 \times 10^{-1}$	$-3.463\,780\,9 \times 10^{-1}$
0.9	$-3.953\,836\,9 \times 10^{-1}$	$-3.392\,428\,5 \times 10^{-1}$	$-3.268\,193\,4 \times 10^{-1}$	$-3.152\,202\,1 \times 10^{-1}$	$-3.097\,001\,1 \times 10^{-1}$
1.0	$-3.685\,434\,6 \times 10^{-1}$	$-3.205\,049\,0 \times 10^{-1}$	$-3.097\,629\,6 \times 10^{-1}$	$-2.997\,000\,5 \times 10^{-1}$	$-2.948\,999\,2 \times 10^{-1}$
2.0	$-2.245\,046\,2 \times 10^{-1}$	$-2.129\,203\,2 \times 10^{-1}$	$-2.101\,613\,5 \times 10^{-1}$	$-2.075\,211\,6 \times 10^{-1}$	$-2.062\,428\,8 \times 10^{-1}$
5.0	$-1.075\,322\,0 \times 10^{-1}$	$-1.116\,569\,5 \times 10^{-1}$	$-1.127\,116\,1 \times 10^{-1}$	$-1.137\,481\,4 \times 10^{-1}$	$-1.142\,597\,5 \times 10^{-1}$
7.0	$-8.036\,290\,7 \times 10^{-2}$	$-8.533\,480\,4 \times 10^{-2}$	$-8.661\,542\,2 \times 10^{-2}$	$-8.787\,780\,4 \times 10^{-2}$	$-8.850\,228\,2 \times 10^{-2}$
9.0	$-6.421\,908\,1 \times 10^{-2}$	$-6.907\,720\,3 \times 10^{-2}$	$-7.033\,297\,6 \times 10^{-2}$	$-7.157\,269\,6 \times 10^{-2}$	$-7.218\,664\,1 \times 10^{-2}$

TABLE V. Parameters used to compute the flow rate $Q(a)$ and the macroscopic velocity profile $q(\tau)$ for each channel width.

$2a$	N	M	m	Comp. time (s)
0.05	69	100	34	1.35
0.10	59	120	54	2.39
0.30	35	140	60	0.78
0.50	30	140	60	0.51
0.70	30	140	60	0.52
0.90	30	200	67	0.81
1.00	30	200	74	0.92
2.00	24	400	90	1.30
5.00	24	700	130	4.31
7.00	24	900	150	6.92
9.00	24	1400	150	10.12

channel width are reported in Table V. The parameters reported in the Table V are also plotted in Fig. 7 to better appreciate their variations with respect the channel thickness $2a$. As it can be seen from Tables II–IV, our results match all the digits reported in the published benchmarks.[54]

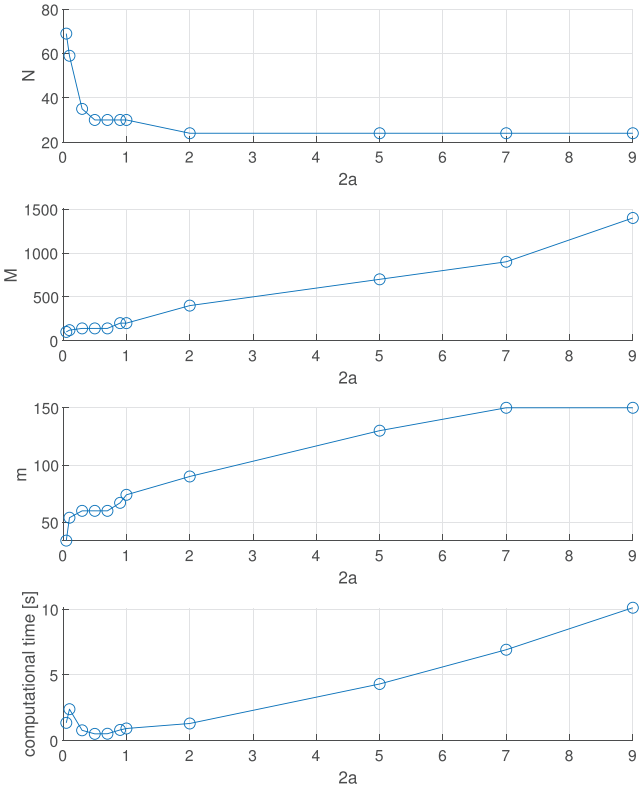


FIG. 7. Variations of the chosen parameters and computational times in function of the channel width.

VII. CONCLUSIONS

We have developed and successfully applied a PINN TFC-based method to solving the thermal creep flow in a plane channel problem. The proposed method is proved to be accurate and efficient. To check the accuracy of our solutions, we have compared them with the benchmarks published by Loyalka *et al.*,[20] Barichello *et al.*,[81] and Ganapol,[54] matching their results up to the fifth and seventh digits.

The results of this paper show the feasibility of the proposed methodology and set the stage for the application of PINNs to obtain accurate solutions for a large variety of problems arising from the Boltzmann linear integrodifferential equation. We are currently working on applying the proposed PINN-TFC based method to solving more RGD problems such as Poiseuille flow in a plane channel[83] as formulated by Barichello *et al.*[84,85] Moreover, we are currently working on solving high-dimensional problems with the X-TFC algorithm,[56] which has been proved being a significant improvement for the PINN frameworks for estimating solutions of differential equations,[52] in particular, for PDEs.

Future works then will focus on the application of this new method to solve 3D time-dependent problems arising from the Boltzmann linear integrodifferential equation, in rarefied gas dynamics, radiative transfer, and neutron transport problems.

ACKNOWLEDGMENTS

The authors would like to acknowledge Mr. Andrea D'Ambrosio for his precious advise that helped to improve this manuscript.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1]O. Reynolds, "Xviii. On certain dimensional properties of matter in the gaseous state.-Part I. Experimental researches on thermal transpiration of gases through porous plates and on the laws of transpiration and impulsion, including an experimental proof that gas is not a continuous plenum.-Part II. On an extension of the dynamical theory of gas, which includes the stresses, tangential and normal, caused by a varying condition of gas, and affords an explanation of the phenomena of transpiration and impulsion," Philos. Trans. R. Soc. London **170**, 727–845 (1879).
[2]J. C. Maxwell, "III. On stresses in rarefied gases arising from inequalities of temperature," Proc. R. Soc. London **27**(185-189), 304–308 (1878).
[3]M. Knudsen, "Thermischer molekulardruck der gase in röhren," Ann. Phys. **338**(16), 1435–1448 (1910).
[4]T. Edmonds and J. Hobson, "A study of thermal transpiration using ultrahigh-vacuum techniques," J. Vac. Sci. Technol. **2**(4), 182–197 (1965).
[5]J. Hobson, "Surface smoothness in thermal transpiration at very low pressures," J. Vac. Sci. Technol. **6**(1), 257–259 (1969).
[6]R. Healy and T. Storvick, "Rotational collision number and Eucken factors from thermal transpiration measurements," J. Chem. Phys. **50**(3), 1419–1427 (1969).
[7]A. D. Gupta and T. Storvick, "Analysis of the heat conductivity data for polar and nonpolar gases using thermal transpiration measurements," J. Chem. Phys. **52**(2), 742–749 (1970).
[8]B. Annis, "Thermal creep in gases," J. Chem. Phys. **57**(7), 2898–2905 (1972).
[9]H. Niimi, "Thermal creep flow of rarefied gas between two parallel plates," J. Phys. Soc. Jpn. **30**(2), 572–574 (1971).
[10]C. Cercignani, "The Boltzmann equation," *The Boltzmann Equation and Its Applications* (Springer, 1988), pp. 40–103.

Phys. Fluids **33**, 047110 (2021); doi: 10.1063/5.0046181

[11]J. Ferziger and H. Kaper, "Mathematical theory of transport processes in gases," Am. J. Phys. **41**(4), 601–603 (1973).
[12]F. Sharipov, *Rarefied Gas Dynamics: Fundamentals for Research and Practice* (John Wiley & Sons, 2015).
[13]C. Cercignani, *Theory and Application of the Boltzmann Equation* (Scottish Academic Press, 1975).
[14]P. L. Bhatnagar, E. P. Gross, and M. Krook, "A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems," Phys. Rev. **94**(3), 511 (1954).
[15]E. Shakhov, "Generalization of the Krook kinetic relaxation equation," Fluid Dyn. **3**(5), 95–96 (1972).
[16]S. Loyalka, "Thermal transpiration in a cylindrical tube," Phys. Fluids **12**(11), 2301–2305 (1969).
[17]S. Loyalka and T. Storvick, "Kinetic theory of thermal transpiration and mechanocaloric effect. III. Flow of a polyatomic gas between parallel plates," J. Chem. Phys. **71**(1), 339–350 (1979).
[18]S. Loyalka, "Comments on 'Poiseuille flow and thermal creep of a rarefied gas between parallel plates'," Phys. Fluids **17**(5), 1053–1055 (1974).
[19]V. G. Chernyak, B. T. Porodnov, and P. E. Suetin, "Application of the variational method to the problem of thermomolecular pressure difference in a cylindrical capillary," J. Eng. Phys. **26**(3), 309–312 (1974).
[20]S. Loyalka, N. Petrellis, and T. Storvick, "Some exact numerical results for the BGK model: Couette, Poiseuille and thermal creep flow between parallel plates," Z. Angew. Math. Phys. **30**(3), 514–521 (1979).
[21]D. Valougeorgis and J. Thomas, Jr., "Exact numerical results for Poiseuille and thermal creep flow in a cylindrical tube," Phys. Fluids **29**(2), 423–429 (1986).
[22]S. Loyalka and K. Hickey, "Kinetic theory of thermal transpiration and the mechanocaloric effect: Planar flow of a rigid sphere gas with arbitrary accommodation at the surface," J. Vac. Sci. Technol. A **9**(1), 158–163 (1991).
[23]F. Sharipov, "Rarefied gas flow through a long tube at any temperature ratio," J. Vac. Sci. Technol. A **14**(4), 2627–2635 (1996).
[24]K. Ritos, Y. Lihnaropoulos, S. Naris, and D. Valougeorgis, "Pressure-and temperature-driven flow through triangular and trapezoidal microchannels," Heat Transfer Eng. **32**(13-14), 1101–1107 (2011).
[25]T. Ohwada, Y. Sone, and K. Aoki, "Numerical analysis of the shear and thermal creep flows of a rarefied gas over a plane wall on the basis of the linearized Boltzmann equation for hard-sphere molecules," Phys. Fluids A **1**(9), 1588–1599 (1989).
[26]T. Kanki and S. Iuchi, "Poiseuille flow and thermal creep of a rarefied gas between parallel plates," Phys. Fluids **16**(5), 594–599 (1973).
[27]G. A. Bird, *Molecular Gas Dynamics and the Direct Simulation of Gas Flows* (Clarendon Press, 1994).
[28]H. Akhlaghi, M. Balaj, and E. Roohi, "Hydrodynamic behaviour of micro/nanoscale Poiseuille flow under thermal creep condition," Appl. Phys. Lett. **103**(7), 073108 (2013).
[29]H. Akhlaghi, E. Roohi, and M. Balaj, "A thorough study on thermal mass flux of rarefied flow through micro/nanochannels," Appl. Phys. Lett. **104**(7), 073109 (2014).
[30]H. Akhlaghi and E. Roohi, "Mass flow rate prediction of pressure–temperature-driven gas flows through micro/nanoscale channels," Continuum Mech. Thermodyn. **26**(1), 67–78 (2014).
[31]V. Shahabi, T. Baier, E. Roohi, and S. Hardt, "Thermally induced gas flows in ratchet channels with diffuse and specular boundaries," Sci. Rep. **7**(1), 41412 (2017).
[32]C. Tantos, "Polyatomic thermal creep flows through long microchannels at large temperature ratios," J. Vac. Sci. Technol. A **37**(5), 051602 (2019).
[33]F. Hanson and T. Morse, "Kinetic models for a gas with internal structure," Phys. Fluids **10**(2), 345–353 (1967).
[34]I. Chemyaninov, V. Chernysk, and G. Fomyagin, "Heat and mass transfer processes of a polyatomic gas in a flat channel for arbitrary Knudsen numbers," J. Appl. Mech. Tech. Phys. **25**(6), 881–888 (1984).
[35]B. Carlson and G. Bell, "Solution of the transport equation by the Sn method," Technical Report No. 15/P/2386 (Los Alamos Scientific Lab, New Mexico, 1958).
[36]S. Chandrasekhar, *Radiative Transfer* (Courier Corporation, 2013).
[37]M. Vilhena, C. Segatto, and L. Barichello, "A particular solution for the Sn radiative transfer problems," J. Quant. Spectrosc. Radiat. Transfer **53**(4), 467–469 (1995).
[38]K.-Y. Chien, "Application of the s, method to spherically symmetric radiative-transfer problems," AIAA J. **10**(1), 55–59 (1972).
[39]M. Simch, C. Segatto, and M. Vilhena, "An analytical solution for the Sn radiative transfer equation with polarization in a slab by the LTSN method," J. Quant. Spectrosc. Radiat. Transfer **97**(3), 424–435 (2006).
[40]P. Benoist and A. Kavenoky, "A new method of approximation of the Boltzmann equation," Nucl. Sci. Eng. **32**(2), 225–232 (1968).
[41]C. Siewert and P. Benoist, "The F_N method in neutron-transport theory. Part I: Theory and applications," Nucl. Sci. Eng. **69**(2), 156–160 (1979).
[42]C. Devaux and C. Siewert, "The F_N method for radiative transfer problems without azimuthal symmetry," Z. Angew. Math. Phys. **31**(5), 592–604 (1980).
[43]R. Garcia and C. Siewert, "The F_N method for radiative transfer models that include polarization effects," J. Quant. Spectrosc. Radiat. Transfer **41**(2), 117–145 (1989).
[44]B. D. Ganapol and R. Myneni, "The F_N method for the one-angle radiative transfer equation applied to plant canopies," Remote Sens. Environ. **39**(3), 213–231 (1992).
[45]M. Benassi, R. Garcia, A. Karp, and C. Siewert, "A high-order spherical harmonics solution to the standard problem in radiative transfer," Astrophys. J. **280**, 853–864 (1984).
[46]C. Siewert and J. Thomas, Jr., "A particular solution for the PN method in radiative transfer," J. Quant. Spectrosc. Radiat. Transfer **43**(6), 433–436 (1990).
[47]L. Barichello and C. E. Siewert, "A discrete-ordinates solution for a non-grey model with complete frequency redistribution," J. Quant. Spectrosc. Radiat. Transfer **62**(6), 665–675 (1999).
[48]C. Siewert, "A concise and accurate solution to Chandrasekhar's basic problem in radiative transfer," J. Quant. Spectrosc. Radiat. Transfer **64**(2), 109–130 (2000).
[49]P. Picca and R. Furfaro, "Analytical discrete ordinate method for radiative transfer in dense vegetation canopies," J. Quant. Spectrosc. Radiat. Transfer **118**, 60–69 (2013).
[50]B. Ganapol, "Radiative transfer with internal reflection via the converged discrete ordinates method," J. Quant. Spectrosc. Radiat. Transfer **112**(4), 693–713 (2011).
[51]P. Picca, R. Furfaro, and B. D. Ganapol, "An efficient multiproblem strategy for accurate solutions of linear particle transport problems in spherical geometry," Nucl. Sci. Eng. **170**(2), 103–124 (2012).
[52]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[53]E. Schiassi, A. D'Ambrosio, M. De Florio, R. Furfaro, and F. Curti, "Physics-informed extreme theory of functional connections applied to data-driven parameters discovery of epidemiological compartmental models," arXiv:2008.05554 (2020).
[54]B. D. Ganapol, "1D thermal creep channel flow in the BGK approximation by adding and doubling," Ann. Nucl. Energy **134**, 441–451 (2019).
[55]C. Leake and D. Mortari, "Deep theory of functional connections: A new method for estimating the solutions of partial differential equations," Mach. Learn. Knowl. Extr. **2**(1), 37–55 (2020).
[56]E. Schiassi, C. Leake, M. De Florio, H. Johnston, R. Furfaro, and D. Mortari, "Extreme theory of functional connections: A physics-informed method for solving parametric differential equations," arXiv:2005.10632 (2020).
[57]D. Mortari, "The theory of connections: Connecting points," Mathematics **5**(4), 57 (2017).
[58]D. Mortari, "Least-squares solution of linear differential equations," Mathematics **5**(4), 48 (2017).
[59]D. Mortari, H. Johnston, and L. Smith, "High accuracy least-squares solutions of nonlinear differential equations," J. Comput. Appl. Math. **352**, 293–307 (2019).
[60]M. De Florio, E. Schiassi, R. Furfaro, B. D. Ganapol, and D. Mostacci, "Solutions of Chandrasekhar's basic problem in radiative transfer via theory of functional connections," J. Quant. Spectrosc. Radiat. Transfer **259**, 107384 (2021).

[61]R. Furfaro and D. Mortari, "Least-squares solution of a class of optimal space guidance problems via theory of connections," Acta Astronaut. **168**, 92 (2020).
[62]H. Johnston, E. Schiassi, R. Furfaro, and D. Mortari, "Fuel-efficient powered descent guidance on large planetary bodies via theory of functional connections," arXiv:2001.03572 (2020).
[63]E. Schiassi, A. D'Ambrosio, H. Johnston, R. Furfaro, F. Curti, and D. Mortari, "Complete energy optimal landing on planetary bodies via theory of functional connections," in Proceedings of the Astrodynamics Specialist Conference, AAS, pp. 20–557.
[64]M. Liu, M. Hou, J. Wang, and Y. Cheng, "Solving two-dimensional linear partial differential equations based on Chebyshev neural network with extreme learning machine algorithm," Eng. Comput. **38**, 874 (2020).
[65]Y. Yang, M. Hou, and J. Luo, "A novel improved extreme learning machine algorithm in solving ordinary differential equations by Legendre neural network methods," Adv. Differ. Equations **2018**(1), 469.
[66]G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," Neurocomputing **70**, 489–501 (2006).
[67]J. N. Reddy, "An introduction to the finite element method," J. Pressure Vessel Technol. **111**, 348–349 (1989).
[68]A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," J. Mach. Learn. Res. **18** (2018).
[69]L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," arXiv:1907.04502 (2019).
[70]M. De Florio, E. Schiassi, B. D. Ganapol, and R. Furfaro, "An accurate solution for Poiseuille flow in a plane channel via theory of functional connections" (unpublished).
[71]C. Cercignani, *Rarefied Gas Dynamics: From Basic Concepts to Actual Calculations* (Cambridge University Press, 2000), Vol. 21.
[72]C. Siewert, "Poiseuille, thermal creep and Couette flow: Results based on the CES model of the linearized Boltzmann equation," Eur. J. Mech.-B **21**(5), 579–597 (2002).
[73]Q.-H. Sun, C.-P. Cai, and W. Gao, "On the validity of the Boltzmann-BGK model through relaxation evaluation," Acta Mech. Sin. **30**, 133–143 (2014).
[74]C. H. Kruger and W. Vincenti, *Introduction to Physical Gas Dynamics* (John Wiley & Sons, 1965).
[75]S. Chapman and T. Cowling, *The Mathematical Theory of Non-Uniform Gases* (Cambridge University Press, Cambridge, England, 1970).
[76]Y. Sone, *Kinetic Theory and Fluid Dynamics* (Springer Science & Business Media, 2002).
[77]L. H. Holway, Jr., "Kinetic theory of shock structure using an ellipsoidal distribution function," Rarefied Gas Dyn. **1**, 193 (1965).
[78]M. Williams, "A review of the rarefied gas dynamics theory associated with some classical problems in flow and heat transfer," Z. Angew. Math. Phys. **52**(3), 500–516 (2001).
[79]N. G. Hadjiconstantinou, "The limits of Navier-Stokes theory and kinetic extensions for describing small-scale gaseous hydrodynamics," Phys. Fluids **18**(11), 111301 (2006).
[80]C. Cercignani, "Plane Poiseuille flow according to the method of elementary solutions," J. Math. Anal. Appl. **12**(2), 254–262 (1965).
[81]L. Barichello, M. Camargo, P. Rodrigues, and C. Siewert, "Unified solutions to classical flow problems based on the BGK model," Z. Angew. Math. Phys. **52**(3), 517–534 (2001).
[82]S. K. Prabha and S. P. Sathian, "Molecular-dynamics study of Poiseuille flow in a nanochannel and calculation of energy and momentum accommodation coefficients," Phys. Rev. E **85**(4), 041201 (2012).
[83]F. Sharipov, "Application of the Cercignani–Lampis scattering kernel to calculations of rarefied gas flows. I. Plane flow between two parallel plates," Eur. J. Mech.-B **21**(1), 113–123 (2002).
[84]L. Barichello and C. Siewert, "A discrete-ordinates solution for Poiseuille flow in a plane channel," Z. Angew. Math. Phys. **50**(6), 972–981 (1999).
[85]C. Siewert, R. Garcia, and P. Grandjean, "A concise and accurate solution for Poiseuille flow in a plane channel," J. Math. Phys. **21**(12), 2760–2763 (1980).

Appendix D

De Florio et al. (2022) Poiseuille Flow

Reproduced from De Florio, M., Schiassi, E., Ganapol, B. D., & Furfaro, R. (2022). Physics-Informed Neural Networks for rarefied-gas dynamics: Poiseuille flow in the BGK approximation. *Zeitschrift für angewandte Mathematik und Physik*, 73(3), 1-18. https://doi.org/10.1007/s00033-022-01767-z [10], with the permission of Springer Nature.

Z. Angew. Math. Phys. (2022) 73:126

https://doi.org/10.1007/s00033-022-01767-z

Zeitschrift für angewandte Mathematik und Physik ZAMP

Physics-Informed Neural Networks for rarefied-gas dynamics: Poiseuille flow in the BGK approximation

Mario De Florio, Enrico Schiassi, Barry D. Ganapol and Roberto Furfaro

Abstract. We present a new accurate approach to solving a class of problems in the theory of rarefied–gas dynamics using a *Physics-Informed Neural Networks* framework, where the solution of the problem is approximated by the constrained expressions introduced by the *Theory of Functional Connections*. The constrained expressions are made by a sum of a free function and a functional that always analytically satisfies the equation constraints. The free function used in this work is a Chebyshev neural network trained via the extreme learning machine algorithm. The method is designed to accurately and efficiently solve the linear one-point boundary value problem that arises from the Bhatnagar–Gross–Krook model of the Poiseuille flow between two parallel plates for a wide range of Knudsen numbers. The accuracy of our results is validated via the comparison with the published benchmarks.

Mathematics Subject Classification. 76P05, 68T07, 35Q20.

Keywords. Physics-Informed Neural Networks, Extreme learning machine, Functional interpolation, Rarefied gas dynamics, Poiseuille flow, Boltzmann equation.

1. Introduction

Rarefied–gas dynamics (RGD) has been widely studied in fluid dynamics, both from theoretical and experimental perspectives. Gas is considered in a rarefied state when its Knudsen number, defined by the ratio of the mean free path to the chamber under test, is greater than one. The behavior of the particles of a rarefied gas can be described through the Boltzmann equation. In particular, a Poiseuille flow is a case of motion of a gas induced by a pressure gradient along the walls of the channel. The Poiseuille problem in the RGD has been initially theoretically studied by Reynolds [1] and Maxwell [2]. Experimental studies [3,4] followed, where a minimum in the flow rate has been observed in the transition region (as a function of the pressure Knudsen number). Many researchers applied several methods to solve the class of linear one-point boundary value problems arising from the Boltzmann integrodifferential equation for a rarefied gas, such as [5–7], and [8]. The fundamentals of kinetic theory [9–11] have been used to investigate the Poiseuille flow problem in the field of RGD from a theoretical point of view to find solutions for a large interval of Knudsen number. The Boltzmann equation has been the most used fundamental mathematical model to describe the Poiseuille flow [12], and different models and approximations have been developed, such as the Bhatnagar–Gross–Krook (BGK) model [13] and the Shakhov model [14]. In the years, many numerical methods have been employed to solve such problems in RGD [5,6,15–23]. One of the state-of-the-art methods to tackle these problems is the variation of the discrete-ordinates method (DOM) of Chandrasekhar [24,25]. During the last decays, this family of methods has been widely improved and used to solve the Boltzmann integrodifferential transport equation in many fields, such as RGD and transport of neutrons and photons. We can mention the S_n [26–28], the C_N [29], P_N, and F_N methods [30–35]. Barichello et al. [36,37] used it to tackle the Poiseuille flow in a plane channel problem, generating accurate results used as benchmarks. Siewert [38] applied the ADO method in solving the basic Radiative transfer equation. Recently, Ganapol [39–41] has extended the benchmarks

Published online: 26 May 2022

to additional digits. A new category of numerical methods in the machine learning community has been developed, called Physics-Informed Neural Networks (PINNs). Such methods train neural networks (NNs) in learning the solutions of differential equations (DEs). Nevertheless, PINNs can generally be considered machine learning methods that embed physics into a data-driven representation of functional relationships underlying collections of input–output pairs. As described by Raissi et al. [42], the term PINN defines NNs that utilize physics as a regularizer in the loss function. For example, assume that one wants to perform a regression of a set of experimental data using NNs and that such collected data represent some physical phenomena represented by a set of DEs. One would approximate the data using NNs trained to minimize a mean squared error as a loss function in standard regression. However, there is no guarantee that the physics phenomena governing the data would not be violated. More precisely, especially when the data is noisy, one could experience the issue of over-fitting. That is, the data would be accurately fitted (e.g., with a negligible error between the NN approximation and the experimental data), but the relation input–output, governed by the physics phenomena, could be poorly learned. To this matter, PINNs are introduced to ensure that DEs, modeling the physics of the experimental dataset, are added as a penalty to the loss function. This additional term acts as a regularizer that penalizes the training when the DE and its constraints (e.g., the initial conditions ICs and/or boundary conditions BCs) are violated. Overall, one guarantees that the physics governing the data is not violated. This approach is defined data-driven solution of DEs. Conversely, when the goal is to estimate parameters governing some physical phenomena modeled through a DE (e.g., the thermal conductivity in the heat equation), one refers to data-driven parameters discovery of DEs (e.g., inverse problems) [42,43]. When data is not available, and hence the loss function contains only the residual of the DEs (and its constraints), PINNs are trained for learning the solutions of problems involving DEs. When data is not available, PINNs model a NN representation of functions for which the residuals of the DEs are as close to zero as possible, i.e., PINNs model the solutions of the DEs. To this end, PINNs can also be viewed as methods that approximate solutions of DEs using NNs. In this paper, we consider the linearized BGK model suggested by Bhatnagar, Gross, and Krook [13] to describe the physical problem of the Poiseuille flow [44,45] between two parallel plates following the formulation presented by Williams [7]. Our task is to develop and evaluate a PINNs-based framework to solve the Poiseuille flow between two parallel plates. Following the formulation proposed by Ganapol [40], we demonstrate that our PINN-based approach is accurate in learning the solutions of this class of problems arising from the Boltzmann integrodifferential equation. In particular, to solve this problem, we develop a PINN framework that brings together NNs and the Theory of Functional Connections (TFC), named Extreme Theory of Functional Connections (X-TFC) [46]. TFC, developed by Mortari [47], is a functional interpolation technique for a large class of mathematical objects. According to TFC, any mathematical problem's solution can be represented via constrained expressions (CEs). The CEs are made of a free function and a functional which analytically satisfies the problem constraints, regardless of the choice of the free function. It can be expressed as

$$f(\boldsymbol{x}) \simeq f_{\mathrm{CE}}(\boldsymbol{x}, g(\boldsymbol{x})) = A(\boldsymbol{x}) + B(\boldsymbol{x}, g(\boldsymbol{x})) \tag{1.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $f(\boldsymbol{x})$ is the unknown function to be interpolated, $f_{\mathrm{CE}}(\boldsymbol{x}, g(\boldsymbol{x}))$ is the CE, $A(\boldsymbol{x})$ is the functional that analytical satisfies any given linear constraint, and $B(\boldsymbol{x}, g(\boldsymbol{x}))$ projects the free function $g(\boldsymbol{x})$, which is a real function that needs to be defined on the constraints, onto the space of functions that vanish at the constraints [48]. Among its many applications, TFC has been applied to solve ODEs with usually excellent results, leading to machine-level accuracy solutions in a short computational time [49–51]. The authors already employed PINN TFC-based methods to solve Chandrasekhar's basic problem in radiative transfer [52], thermal creep flow in the BGK approximation [53], optimal control problems for aerospace applications [54–58]. Here, based on the results of De Florio et al. [53], we consider a X-TFC approach where Chebyshev NNs (ChNNs) are employed as free function in the CE. ChNNs have been initially proposed by Namatame [59] and successively adopted by Mall and Chakraverty [60] for solving elliptic PDEs. They are a class of functional link artificial neural networks in which the hidden layer is

replaced by an expansion of Chebyshev polynomials as a functional expansion block for enhancement of the input patterns. In this particular architecture of NN, the input weights and bias are selected to be equal to one and zero, respectively. With this choice of input weights and bias, ChNN become equivalent to a linear combination of Chebyshev polynomials, where the unknowns are the coefficients, similarly to a collocation method [61]. Indeed, under the machine learning community prospective, according to the definition of PINNs, collocation methods using orthogonal polynomials can be considered as a particular family of PINNs. The ChNNs are trained via the extreme learning machine (ELM) algorithm [62,63]. In the ELM algorithm, input weights and bias are randomly selected and not tuned during the training, leaving the output weights the only trainable parameters. Hence, the training is reduced to a fast least squares. The paper is organized as follows. In Sect. 2, the standard PINN methodology and how to apply it to solve a generic first-order linear ODE are presented. In Sect. 3, how to apply PINN TFC-based methods to tackle generic first-order linear ODEs is explained. The PINN TFC-based formulation of the Poiseuille flow in a plane channel problem is shown in Sects. 4 and 5. Numerical results are presented and discussed in Sect. 6. Conclusions are drawn in Sect. 7.

2. Physics-Informed Neural Networks

PINNs have been introduced by Raissi et al. [42], following the original idea proposed by Lagaris et al. [64], to solve two main classes of problems: data-driven solution and data-driven discovery of PDEs. Although in [42] PINNs were presented to face problems governed by PDEs, the same framework can be used to tackle problems involving ODEs as well. PINNs use a single deep neural network (DNN) to approximate the equation solution fully. In particular, the training set is made of randomly parsed data points from the entire high-resolution dataset and randomly sampled collocation points for enforcing the equation constraints (e.g., initial and/or boundary conditions) and the differential equation (DE) on the boundary and inside the solution domain, respectively. Once the output of DNN is computed, it is plugged in the residual of the DE and into the residual of the equation constraints. The NN is then trained to learn the weights and biases parameters that minimize the physics-based cost function, which can be DE and its constraints least squares solved with gradient-based optimization techniques. Thus, the key feature of the PINN is the embedding of the information of the physics of the problem, derived from the DE and its constraints, in their cost functions. To better clarify how a PINN works, the mathematical formulation to solve a first-order linear ODE is reported in detail in Ref. [53]. PINNs, as described above, are the standard PINN frameworks [42]. One of the main shortcomings of the standard PINN frameworks is that the constraints of the equation are not analytically satisfied. Thus, there are competing objectives during the PINN training: learning the DE solution within the domain and learning its constraints. This process leads to have unbalanced gradients during the network training via gradient-based techniques that causes PINNs to have often difficulties approximating the DE solution accurately [65]. Indeed, gradient-based optimization techniques may get stuck in limit cycles or diverge if multiple competing objectives are present [66,67]. In Ref. [65], to surmount this limitation, the authors proposed a learning rate annealing algorithm that employs gradient statistics to adaptively assign proper weights to different terms (e.g., DE residuals within the domain and on the boundaries) in the PINNs loss function while the network is training. The approach employed here is different and is based on the TFC [47] which allows satisfying the equation constraints analytically, hence completely removing the competing objective in the NN training. The following section provides an overview of our PINN TFC-based approach along with the proposed learning algorithm.

3. X-TFC approach to solving first-order linear ODEs

As previously mentioned, we develop an improved PINN framework, that we have already successfully applied to solve the thermal creep flow in the BGK approximation [53]. In the proposed approach, we merge the standard PINN with the TFC method [46,48].Here, we present the step-by-step procedure for solving boundary value linear ODEs via the proposed X-TFC method. Consider solving the following first-order ODE:

$$F(t, y, \dot{y}) = 0 \quad \text{subject to:} \quad y(t_0) = y_0 \tag{3.1}$$

The unknown solution is approximated via the constrained expression, defined within the TFC framework. The general form of the constrained expression of Eq. (1.1) can be rewritten for a univariate problem (e.g., an ODE) in the following simpler form:

$$y(t) = g(t) + \sum_{k=1}^{n} \eta_k p_k(t) = g(t) + \boldsymbol{\eta}^T \boldsymbol{p}(t) \tag{3.2}$$

where $p_k(t)$ are n assigned linearly independent functions and the $g(t)$ is the free function, which must be linearly independent from the $p_k(t)$ functions. The values of the η_k coefficients are calculated by imposing a set of n linear constraints in $y(t)$. By doing that, the constraints for the differential equations are satisfied for any chosen $g(t)$. For a first-order ODE, the CE becomes:

$$y(t) = g(t) + \eta\, p(t) \tag{3.3}$$

where $p(t) = 1$ (see ref. [49,50] for the details of this decision). Thus, the CE becomes:

$$y(t) = g(t) + \eta \tag{3.4}$$

By imposing the constraint, we get:

$$y(t_0) = y_0 = g_0 + \eta \qquad \Longrightarrow \qquad \eta = y_0 - g_0 \tag{3.5}$$

which is plugged into Eq. (3.4) to obtain the CE in its final form:

$$y(t) = g(t) + (y_0 - g_0) \tag{3.6}$$

which represents all possible functions satisfying the boundary value constraint. The derivative then follows:

$$\dot{y}(t) = \dot{g}(t) \tag{3.7}$$

The CE defined by Eq. (3.6) and its derivative are plugged into our ODE which then becomes:

$$F\left(t, y, \dot{y}\right) = 0 \tag{3.8}$$

that is now a boundary condition free. By substituting Eq. (3.6) into Eq. (3.8), the differential equation F is transformed into a new differential equation $\tilde{F}$, which is only a function of the independent variable t and the free function $g(t)$, that is:

$$\tilde{F}\left(t, g, \dot{g}\right) = 0 \tag{3.9}$$

This differential equation is now *unconstrained* and will always satisfy the boundary-value due to the way Eq. (3.4) is built. In this work, we choose the free function $g(t)$ to be a shallow NN, such that:

$$g(t) = \sum_{j=1}^{L} \beta_j \sigma\left(w_j t + b_j\right) = \begin{bmatrix} \sigma(w_1 t + b_1) \\ \vdots \\ \sigma(w_L t + b_L) \end{bmatrix}^T \boldsymbol{\beta} = \boldsymbol{\sigma}^T \boldsymbol{\beta} \tag{3.10}$$

where L is the number of hidden neurons, $w_j \in \mathbb{R}$ is the input weights vector connecting the jth hidden neuron and the input nodes, $\beta_j \in \mathbb{R}$ with $j = 1, ..., L$ is the jth output weight connecting the j^{th} hidden neuron and the output node, and b_j is the bias of the jth hidden neuron, and $\sigma_j(\cdot)$ are activation

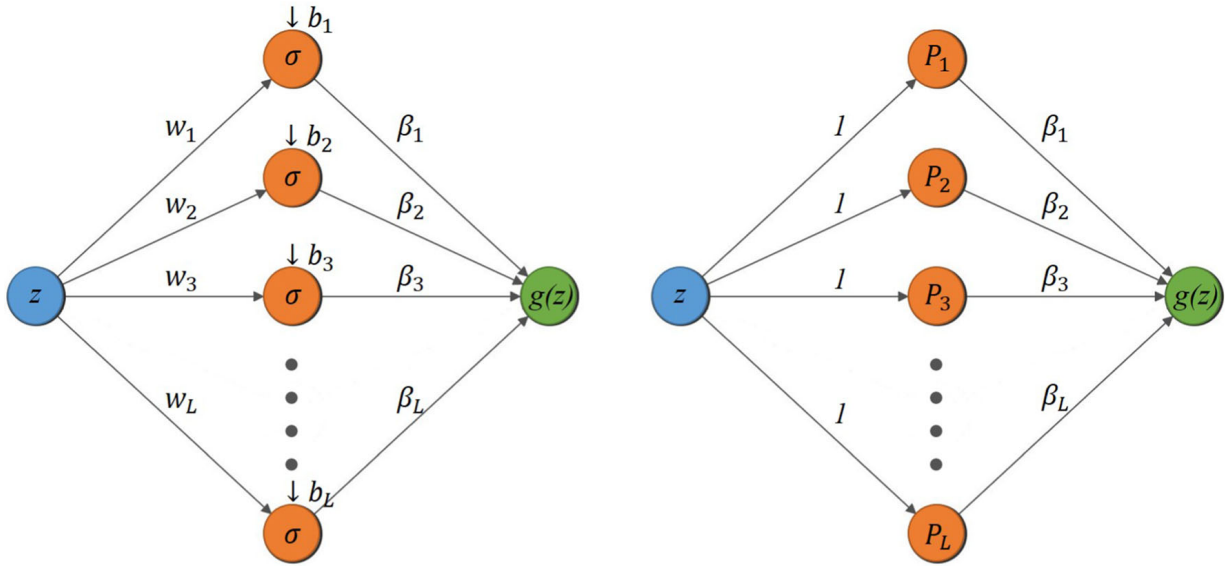


FIG. 1. Standard shallow neural network (left) and shallow Chebyshev neural network (right)

functions. Once the free function is defined, the unknown solution $y(t)$, defined by Eq. (3.6), is computed via the ELM algorithm. According to the ELM algorithm Ref. [62], input weights and biases are randomly selected and not tuned during the training, thus they are known parameters. The activation functions, $\sigma_j(\cdot)$, are chosen by the user, so they are also known. Therefore, the only unknowns to compute are the output weights $\boldsymbol{\beta} = [\beta_1, ..., \beta_L]^T$. The interested reader can find more details about the ELM and its convergence proofs in [62]. In this work, we use ChNN as, based on the results presented in [53] is the best choice to achieve the highest accuracy for these kinds of problems. When using ChNN, the free function becomes,

$$g(t) = \sum_{j=1}^{L} \beta_j P_j \left(w_j t + b_j\right) = \begin{bmatrix} P_1(w_1 t + b_1) \\ \vdots \\ P_L(w_L t + b_L) \end{bmatrix}^T \boldsymbol{\beta} = \boldsymbol{\sigma}^T \boldsymbol{\beta} \tag{3.11}$$

where P_j for $j + 1, ..., L$ is the Chebyshev polynomial of order j. We noticed that for ChNN the choice of input weights and biases that guarantees the best accuracy, is to set $w_j = 1$ and $b_j = 0\,, \forall j = 1, ..., L$. Thus, using ChNNs with this choice on input weights and bias, Eq. (3.10) modifies as follows:

$$g(t) = \sum_{j=1}^{L} \beta_j P_j \left(t\right) = \begin{bmatrix} P_1(t) \\ \vdots \\ P_L(t) \end{bmatrix}^T \boldsymbol{\beta} = \boldsymbol{P}^T \boldsymbol{\beta} \tag{3.12}$$

With this choice of input weights and bias, ChNNs become equivalent to a linear combination of Chebyshev polynomials, where the unknowns are the coefficients, similarly to a collocation method. For the convenience of the reader, in Fig. (1) we show a representation of a general shallow NN and the ChNN that we use to solve the problem presented in this manuscript.

It may happen that the activation functions are defined on an inconsistent domain $z \in [z_0, z_f]$ (for instance, Chebyshev and Legendre polynomials are defined on $z \in [-1, +1]$). Thus, our independent variable $t \in [t_0, t_f]$ needs to be mapped in the z domain. In general to have better training performance, regardless to the activation function used, it is convenient to map the independent variable $t \in [t_0, t_f]$

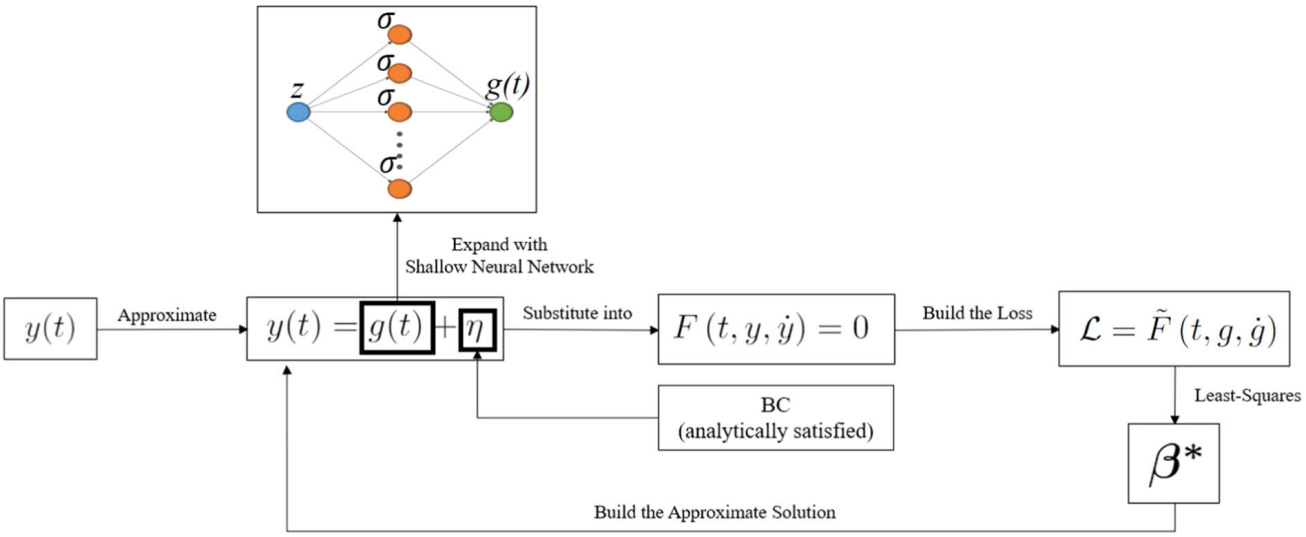


FIG. 2. Schematic of the X-TFC framework to solve linear ODEs with one constraint in one point

in the z domain (usually $z \in [0, +1]$ or $z \in [-1, +1]$). This can be done using the following linear transformation:

$$z = z_0 + c(t - t_0) \quad \longleftrightarrow \quad t = t_0 + \frac{1}{c}(z - z_0)$$

where c is the mapping coefficient:

$$c = \frac{z_f - z_0}{t_f - t_0}$$

By the derivative chain rule, the nth derivative of Eq. (3.10) is,

$$\frac{\mathrm{d}^n g(t)}{\mathrm{d}t^n} = c^n \boldsymbol{\beta}^T \frac{\mathrm{d}^n \boldsymbol{\sigma}(z)}{\mathrm{d}z^n}, \tag{3.13}$$

which defines all mappings of the free function. Finally, the domain t must be discretized by M points, and then, the DE reduces to the following:

$$\bar{F}(\boldsymbol{\beta}) = 0 \tag{3.14}$$

Equation (3.14) is an *unconstrained* optimization problem that can be solved via different optimization schemes. As the problem is linear, rearranging the term in Eq. (3.14), it reduces to a linear system of algebraic equation: $A\boldsymbol{\beta} = \boldsymbol{b}$. Therefore,

$$\boldsymbol{\beta} = \left(A^T A\right)^{-1} A^T \boldsymbol{b} \tag{3.15}$$

For the convenience of the reader, a schematic that summarizes how the PINN TFC-based method works for first-order linear ODEs is reported in Fig. 2

4. Poiseuille flow in a plane channel: problem formulation

In the underlying physical problem, a rarefied gas flows between two parallel plates, separated by a distance d, in the z direction, as shown in Fig. 3.

A pressure gradient causes a density gradient $k\rho_0$ (where ρ_0 is the density at the boundary $z = 0$, and k is proportional to the pressure gradient), which in turn causes the flowing of the gas. The temperature

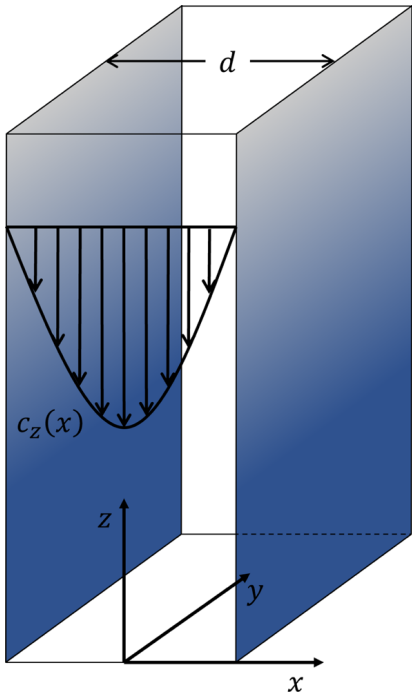


FIG. 3. Poiseuille microchannel flow

T is assumed to be constant on the walls, and therefore also in the gas because of the linearization. The problem to be solved follows the formulation presented by Siewert et al. [68]

$$\frac{1}{2}k\theta + \theta c_x \frac{\partial}{\partial x} Z(x, c_x) + Z(x, c_x) = \pi^{-1/2} \int_{-\infty}^{\infty} e^{-u^2} Z(x, u) du \tag{4.1}$$

for $x \in (-d/2, d/2)$ and $c_x \in (-\infty, \infty)$, and subject to:

$$\left\{ Z(-d/2, c_x) = (1-\alpha)\, Z(-d/2, -c_x) Z(d/2, -c_x) = (1-\alpha)\, Z(d/2, c_x) \right. \tag{4.2}$$

for $c_x \in (0, \infty)$, where $\alpha \in (0, 1]$ is the accommodation coefficient, θ is the mean-free time. Following Ref. [36], we introduce new variables: $\tau = x/\theta$, $\delta = d/\theta$ (Knudsen number) and $u = c_x$, obtaining:

$$\frac{1}{2}k\theta + u \frac{\partial}{\partial \tau} Z(\tau, u) + Z(\tau, u) = \pi^{-1/2} \int_{-\infty}^{\infty} e^{-u^2} Z(\tau, u) du \tag{4.3}$$

for $\tau \in (\delta/2, \delta/2)$ and $u \in (0, \infty)$, and the boundary conditions

$$\begin{cases} Z(-\delta/2, u) = (1-\alpha) Z(-\delta/2, -u) \\ Z(\delta/2, -u) = (1-\alpha) Z(\delta/2, u) \end{cases} \tag{4.4}$$

for $u \in (0, \infty)$. Now, by using a particular solution that accounts for the inhomogeneous term in Eq. (4.3), we can obtain the equation in a homogeneous form. Using

$$Z(\tau, u) = \frac{1}{2}k\theta \left[\tau^2 - 2\tau u + 2u^2 - a^2 - 2Y(\tau, u) \right] \tag{4.5}$$

where $2a = \delta$, and plugging it into (4.1) and (4.2), we find that $Y(\tau, u)$ satisfies the following linear integrodifferential equation:

$$u\frac{\partial}{\partial \tau}Y(\tau,u) + Y(\tau,u) = \int_{-\infty}^{\infty} \Psi(u)Y(\tau,u)du \tag{4.6}$$

for $\tau \in (-a, a)$ and $u \in (-\infty, \infty)$, subject to the following boundary conditions:

$$\begin{cases} Y(-a,u) = (1-\alpha)Y(-a,-u) + \alpha u^2 + au(2-\alpha) \\ Y(a,-u) = (1-\alpha)Y(a,u) + \alpha u^2 + au(2-\alpha) \end{cases} \tag{4.7}$$

for $u \in (0, \infty)$. Here, $\Psi(u)$ is a weight function defined as follows:

$$\Psi(u) = \pi^{-1/2}e^{-u^2} \tag{4.8}$$

We can take advantage of spatial symmetry to reduce the computational cost. Therefore, the boundary conditions become

$$\begin{cases} Y(0,u) = Y(0,-u) \\ Y(a,-u) = (1-\alpha)Y(a,u) + \alpha u^2 + au(2-\alpha) \end{cases} \tag{4.9}$$

For simplicity, we let $K(u) = \alpha u^2 + au(2-\alpha)$, then:

$$\begin{cases} Y(0,u) = Y(0,-u) \\ Y(a,-u) = (1-\alpha)Y(a,u) + K(u) \end{cases} \tag{4.10}$$

The problem (4.6)–(4.10) is the one to be solved via the TFC.

5. X-TFC formulation of the problem

Here, we are dealing a boundary value problem (BVP). Thus, according to (3.2), $y(\tau)$ can be expressed as following:

$$y(\tau) = g(\tau) + \eta p(\tau). \tag{5.1}$$

We choose $p(\tau) = 1$, as suggested by [49], and the free function $g(\tau)$ as a Chebyshev NN:

$$g(\tau) = \boldsymbol{\sigma}^T(\tau)\boldsymbol{\beta} \tag{5.2}$$

We need to map the independent variable τ to a new independent variable x that ranges in $[-1, 1]$. The new independent variable χ is defined as $\chi = c(\tau + a) - 1$, where c is the following mapping constant:

$$c = \frac{\chi_f - \chi_0}{\tau_f - \tau_0} = \frac{1}{a}$$

Considering the new independent variable, we have:

$$\begin{cases} Y(\tau,u) = Y(\chi,u) \\ \dfrac{\partial}{\partial \tau}Y(\tau,u) = c\dfrac{\partial}{\partial \chi}Y(\chi,u) \end{cases} \tag{5.3}$$

Then, the problem becomes:

$$cu\frac{\partial}{\partial \chi}Y(\chi,u) + Y(\chi,u) = \int_{-\infty}^{\infty} \Psi(u')Y(\chi,u')du' \tag{5.4}$$

In order to use a Gauss–Legendre quadrature in the range $[-1, 1]$, we can use a further change of variable:

$$u = -\log(|\mu|); \qquad du = -\frac{1}{\mu}d\mu; \qquad \Psi(\mu) = \pi^{-1/2}e^{-(-\log(|\mu|))^2} \tag{5.5}$$

where $\mu \in [-1; 1]$. We can then rewrite the problem in the following form:

$$-c\log(\mu)\frac{\partial}{\partial\chi}Y(\chi,\mu) + Y(\chi,\mu) = \int_{-1}^{1} -\frac{1}{\mu'}\Psi(\mu')Y(\chi,\mu')d\mu' \tag{5.6}$$

Via discretizing the velocity μ into N points in the interval $(0, 1]$ according to the Gauss–Legendre quadrature scheme, we have:

$$\mu \quad \longrightarrow \quad \boldsymbol{\mu} = \{\mu_i\}_{i=1}^{N} \qquad ; \qquad \boldsymbol{\mu} \in (N \times 1)$$

it follows that:

$$Y(\chi,\mu) \quad \longrightarrow \quad \mathbf{Y}(\chi) = \{Y(\chi,\mu_i)\}_{i=1}^{N} \qquad ; \qquad \mathbf{Y}(\chi) \in (N \times 1) \tag{5.7}$$

The problem can be split for both positive and negative molecular velocity following Ref. [36], where the Gauss–Legendre quadrature rule is used to evaluate the integral in the range $[0, 1]$:

$$-c\log(\mu_i)\frac{\partial}{\partial\chi}Y(\chi,\mu_i) + Y(\chi,\mu_i) = \sum_{k=1}^{N} w_k\frac{1}{\mu_k}\Psi(\mu_k)\left[Y(\chi,\mu_k) + Y(\chi,-\mu_k)\right] \tag{5.8}$$

$$c\log(\mu_i)\frac{\partial}{\partial\chi}Y(\chi,-\mu_i) + Y(\chi,-\mu_i) = \sum_{k=1}^{N} w_k\frac{1}{\mu_k}\Psi(\mu_k)\left[Y(\chi,\mu_k) + Y(\chi,-\mu_k)\right] \tag{5.9}$$

subject to:

$$\begin{cases} Y(0,\mu_i) = Y(0,-\mu_i) \\ Y(a,-\mu_i) = (1-\alpha)Y(a,\mu_i) + K_i \end{cases} \tag{5.10}$$

where $K_i = \alpha\log^2(\mu_i) - a\log(\mu_i)(2-\alpha)$. To simplify the notation, we suppress the χ and μ dependency and write the terms for the positive and negative molecular velocity as Y^+ and Y^-, respectively. Thus, our problem becomes:

$$-c\log(\mu_i)\frac{\partial}{\partial\chi}Y_i^+ + Y_i^+ = \sum_{k=1}^{N} w_k\frac{1}{\mu_k}\Psi(\mu_k)\left(Y_k^+ + Y_k^-\right) \tag{5.11}$$

$$c\log(\mu_i)\frac{\partial}{\partial\chi}Y_i^- + Y_i^- = \sum_{k=1}^{N} w_k\frac{1}{\mu_k}\Psi(\mu_k)\left(Y_k^+ + Y_k^-\right) \tag{5.12}$$

subject to:

$$\begin{cases} Y_0^+ = Y_0^- \\ Y_f^- = (1-\alpha)Y_f^+ + K_i \end{cases} \tag{5.13}$$

Our constrained expressions, for positive and negative terms, are:

$$Y^{\pm}(\chi) = g^{\pm}(\chi) + \eta^{\pm} \tag{5.14}$$

and according to the boundary conditions, we have:

$$Y_0^+ = \boldsymbol{\sigma_0}^T\boldsymbol{\beta}^+ + \eta^+ \qquad Y_f^- = \boldsymbol{\sigma_f}^T\boldsymbol{\beta}^- + \eta^-$$
$$Y_f^+ = \boldsymbol{\sigma_f}^T\boldsymbol{\beta}^+ + \eta^+ \qquad Y_0^- = \boldsymbol{\sigma_0}^T\boldsymbol{\beta}^- + \eta^-$$

Replacing them in the previous system of equations, we obtain:

$$\begin{cases} \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^+ + \eta_i^+ = \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^- + \eta_i^- \\ \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^- + \eta_i^- = (1-\alpha)\, \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^+ + (1-\alpha)\, \eta_i^+ + K_i \end{cases} \tag{5.15}$$

and then

$$\begin{cases} \eta_i^+ - \eta_i^- = \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^- - \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^+ \\ (1-\alpha)\, \eta_i^+ - \eta_i^- = \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^- - (1-\alpha) \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^+ - K_i \end{cases}$$

that in matrix form becomes:

$$\begin{bmatrix} 1 & -1 \\ 1-\alpha & -1 \end{bmatrix} \begin{bmatrix} \eta_i^+ \\ \eta_i^- \end{bmatrix} = \begin{bmatrix} \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^- - \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^+ \\ \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^- - (1-\alpha) \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^+ - K_i \end{bmatrix}$$

And therefore

$$\begin{bmatrix} \eta_i^+ \\ \eta_i^- \end{bmatrix} = -\frac{1}{\alpha} \begin{bmatrix} -1 & 1 \\ -(1-\alpha) & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^- - \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^+ \\ \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^- - (1-\alpha) \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^+ - K_i \end{bmatrix}$$

that is

$$\begin{bmatrix} \eta_i^+ \\ \eta_i^- \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha} & -\frac{1}{\alpha} \\ \frac{1-\alpha}{\alpha} & -\frac{1}{\alpha} \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^- - \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^+ \\ \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^- - (1-\alpha) \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^+ - K_i \end{bmatrix}$$

By calling $\varphi = \frac{1-\alpha}{\alpha}$ and $\gamma = \frac{1}{\alpha}$, we can rewrite:

$$\begin{bmatrix} \eta_i^+ \\ \eta_i^- \end{bmatrix} = \begin{bmatrix} \gamma & -\gamma \\ \varphi & -\gamma \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^- - \boldsymbol{\sigma_0}^T \boldsymbol{\beta_i}^+ \\ \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^- - (1-\alpha) \boldsymbol{\sigma_f}^T \boldsymbol{\beta_i}^+ - K_i \end{bmatrix}$$

and thus

$$\begin{cases} \eta_i^+ = \left(\varphi \boldsymbol{\sigma_f} - \gamma \boldsymbol{\sigma_0}\right)^T \boldsymbol{\beta_i}^+ + \gamma \left(\boldsymbol{\sigma_0} - \boldsymbol{\sigma_f}\right)^T \boldsymbol{\beta_i}^- + \gamma K_i \\ \eta_i^- = \varphi \left(\boldsymbol{\sigma_f} - \boldsymbol{\sigma_0}\right)^T \boldsymbol{\beta_i}^+ + \left(\varphi \boldsymbol{\sigma_0} - \gamma \boldsymbol{\sigma_f}\right)^T \boldsymbol{\beta_i}^- + \gamma K_i \end{cases} \tag{5.16}$$

Replacing them in the constrained expressions, we have:

$$\begin{aligned} Y_i^+ &= \left(\boldsymbol{\sigma} + \varphi \boldsymbol{\sigma_f} - \gamma \boldsymbol{\sigma_0}\right)^T \boldsymbol{\beta_i}^+ + \gamma \left(\boldsymbol{\sigma_0} - \boldsymbol{\sigma_f}\right)^T \boldsymbol{\beta_i}^- + \gamma K_i \\ Y_i^- &= \varphi \left(\boldsymbol{\sigma_f} - \boldsymbol{\sigma_0}\right)^T \boldsymbol{\beta_i}^+ + \left(\boldsymbol{\sigma} + \varphi \boldsymbol{\sigma_0} - \gamma \boldsymbol{\sigma_f}\right)^T \boldsymbol{\beta_i}^- + \gamma K_i \end{aligned} \tag{5.17}$$

and replacing the constrained expressions in the equations of our problem, we obtain:

$$\begin{aligned} &\left(-c \log(\mu_i)\, \boldsymbol{\sigma'} + \boldsymbol{\sigma} + \varphi \boldsymbol{\sigma_f} - \gamma \boldsymbol{\sigma_0}\right)^T \boldsymbol{\beta_i}^+ + \gamma \left(\boldsymbol{\sigma_0} - \boldsymbol{\sigma_f}\right)^T \boldsymbol{\beta_i}^- \\ &\quad - \sum_{k=1}^{N} w_k \frac{1}{\mu_k} \Psi(\mu_k) \Big[\left(\boldsymbol{\sigma} + 2\varphi \boldsymbol{\sigma_f} - (\gamma + \varphi) \boldsymbol{\sigma_0}\right)^T \boldsymbol{\beta_k}^+ + \left(\boldsymbol{\sigma} + (\gamma + \varphi) \boldsymbol{\sigma_0} - 2\gamma \boldsymbol{\sigma_f}\right)^T \boldsymbol{\beta_k}^- \Big] \\ &= -\gamma K_i + \sum_{k=1}^{N} 2 w_k \Psi_k \gamma K_k \end{aligned} \tag{5.18}$$

$$\begin{aligned} &\varphi \left(\boldsymbol{\sigma_f} - \boldsymbol{\sigma_0}\right)^T \boldsymbol{\beta_i}^+ + \left(c \log(\mu_i)\, \boldsymbol{\sigma'} + \boldsymbol{\sigma} + \varphi \boldsymbol{\sigma_0} - \gamma \boldsymbol{\sigma_f}\right) \boldsymbol{\beta_i}^- \\ &\quad - \sum_{k=1}^{N} w_k \frac{1}{\mu_k} \Psi(\mu_k) \Big[\left(\boldsymbol{\sigma} + 2\varphi \boldsymbol{\sigma_f} - (\gamma + \varphi) \boldsymbol{\sigma_0}\right)^T \boldsymbol{\beta_k}^+ + \left(\boldsymbol{\sigma} + (\gamma + \varphi) \boldsymbol{\sigma_0} - 2\gamma \boldsymbol{\sigma_f}\right)^T \boldsymbol{\beta_k}^- \Big] \\ &= -\gamma K_i + \sum_{k=1}^{N} 2 w_k \Psi_k \gamma K_k \end{aligned} \tag{5.19}$$

For the sake of simplicity, we write the inhomogeneous terms as:

$$b_i^+ = b_i^- = -\gamma K_i + \sum_{k=1}^{N} 2 w_k \Psi_k \gamma K_k$$

Expanding the summations, we get the following matrix form:

$$\begin{bmatrix} \blacklozenge_k + \bigstar_i & \heartsuit_k + \clubsuit \\ \blacklozenge_k + \spadesuit & \heartsuit_k + \blacktriangledown_i \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_i^+ \\ \boldsymbol{\beta}_i^- \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_i^+ \\ \boldsymbol{b}_i^- \end{bmatrix} \tag{5.20}$$

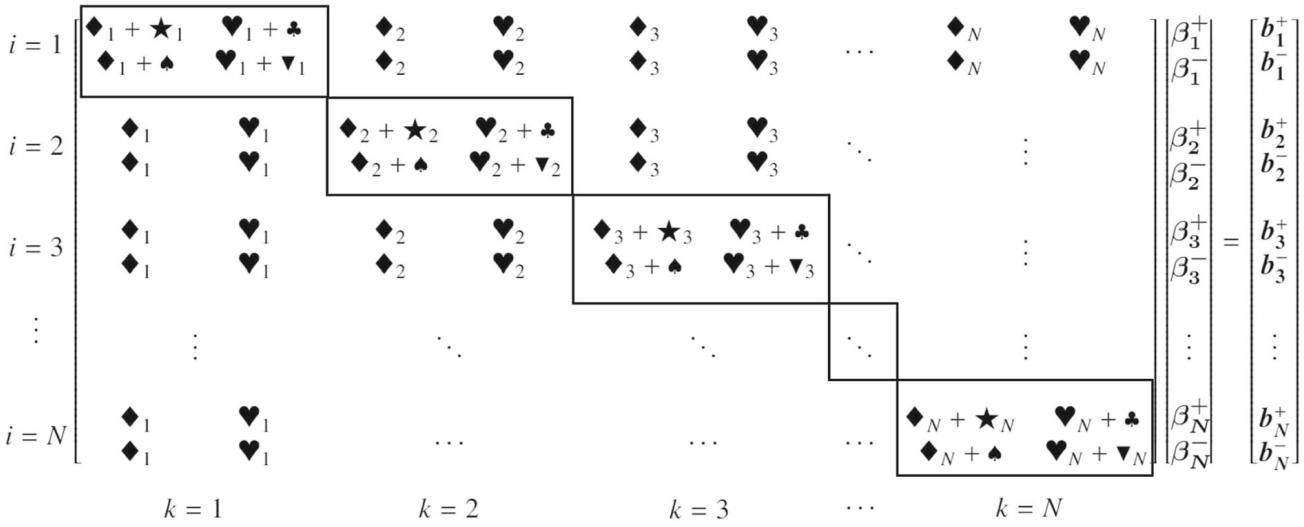
where:

$$\bigstar_i = -c \log(\mu_i)\, \boldsymbol{\sigma'}^T + \boldsymbol{\sigma}^T + \varphi \boldsymbol{\sigma_f}^T - \gamma \boldsymbol{\sigma_0}^T \qquad \blacktriangledown_i = c \log(\mu_i)\, \boldsymbol{\sigma'}^T + \boldsymbol{\sigma}^T + \varphi \boldsymbol{\sigma_0}^T - \gamma \boldsymbol{\sigma_f}^T$$

$$\spadesuit = \varphi (\boldsymbol{\sigma_f} - \boldsymbol{\sigma_0})^T \qquad \clubsuit = \gamma (\boldsymbol{\sigma_0} - \boldsymbol{\sigma_f})^T$$

$$\blacklozenge_k = -w_k \frac{1}{\mu_k} \Psi(\mu_k) (\boldsymbol{\sigma} + 2\varphi \boldsymbol{\sigma_f} - (\gamma + \varphi) \boldsymbol{\sigma_0})^T \qquad \heartsuit_k = -w_k \frac{1}{\mu_k} \Psi(\mu_k) (\boldsymbol{\sigma} + (\gamma + \varphi) \boldsymbol{\sigma_0} - 2\gamma \boldsymbol{\sigma_f})^T$$

Thus, the problem reduces to the following linear system:



which is a problem of the type:

$$\boldsymbol{A\beta} = \boldsymbol{B} \tag{5.21}$$

whose the dimensions are:

$\boldsymbol{\beta}^{\pm}{}_{\boldsymbol{i}\ L\times 1}$
$\boldsymbol{\beta}_{(2LN)\times 1}$
$\boldsymbol{b}^{\pm}{}_{\boldsymbol{i}\ M\times 1}$
$\boldsymbol{B}_{(2MN)\times 1}$
$\bigstar_{\boldsymbol{i}\ (M\times L)}$ (same dimension for the other terms of the matrix $\boldsymbol{A}$)
$\boldsymbol{A}_{(2MN)\times(2LN)}$

where M is the discretization order for τ, and L is the number of neurons used. That is, the original BVP has been reformulated as an *unconstrained* optimization problem. In this case, $\boldsymbol{\beta}$ is calculated via LS, and then the final solutions are built according to (5.17).

6. Results and discussions

To demonstrate the precision of the X-TFC in solving the problem, we report the macroscopic velocity profile, that according to Loyalka et al. [6] is given by:

$$q(\tau) = \frac{1}{k\theta} \int_{-\infty}^{\infty} \Psi(u) Y(\tau, u) du \tag{6.1}$$

By plugging Eq. (4.5) into Eq. (6.1), we obtain:

$$q(\tau) = \frac{1}{2}\left(1 - a^2 + \tau^2\right) - Y_0(\tau) \tag{6.2}$$

where $Y_0(\tau)$ is given by [36]:

$$Y_0(\tau) = \int_{-\infty}^{\infty} \Psi(u) Y(\tau, u) du \tag{6.3}$$

that is

$$Y_0(\tau) = \int_{-1}^{1} \frac{1}{\mu} \Psi(\mu) Y(\tau, \mu) d\mu \tag{6.4}$$

which it can be computed as:

$$Y_0 = \sum_{i=1}^{N} w_i \frac{1}{\mu_i} \Psi(\mu_i) \left(Y_i{}^{+} + Y_i{}^{-}\right) \tag{6.5}$$

Figure 4 shows the qualitative behavior of the macroscopic velocity for different values of Knudsen number δ, computed with different types of NNs. It can be seen how the macroscopic velocity increases for increasing the thickness of the channel, and then of *Kn*. Hence, Fig. 4 proves that all the NNs tested are able to capture the qualitative behavior of the macroscopic velocity.

We coded the problem in the MATLAB R2019a platform and ran all our simulations with an Intel Core i7-9700 CPU PC with 64 GB of RAM. Our numerical results are presented below. In Table 1, the results for a channel of half width $a = 1$, for different values of the accommodation number, are reported. All our digits match with the benchmark reported in Ref. [36]. These results are obtained by setting $N = 22$, $M = 200$, and $L = 50 \pm 2$. The computational time was 0.24 seconds.

Furthermore, we computed the flow rate:

$$Q = -\frac{1}{2a^2} \int_{-a}^{a} q(\tau) d\tau \tag{6.6}$$

and, again, by making use of Eq. (4.5), we find:

$$Q = \frac{1}{2a^2} \int_{-a}^{a} Y_0(\tau) d\tau - \frac{1}{2a}\left(1 - \frac{2}{3}a^2\right) \tag{6.7}$$

The numerical results for several values of channel width and accommodation number are listed in Table 2. All our digits match with the benchmark reported in Ref. [36]. We used different spatial discretizations based on the Knudsen number, from $M = 40$ and $L = 24$ for $\delta = 2a = 0.05$, to $M = 300$ and $L = 90$ for $\delta = 2a = 9$. All the results are obtained with a velocity discretization of $N = 30$. The computational time ranges from 0.04 to 1.6 seconds.

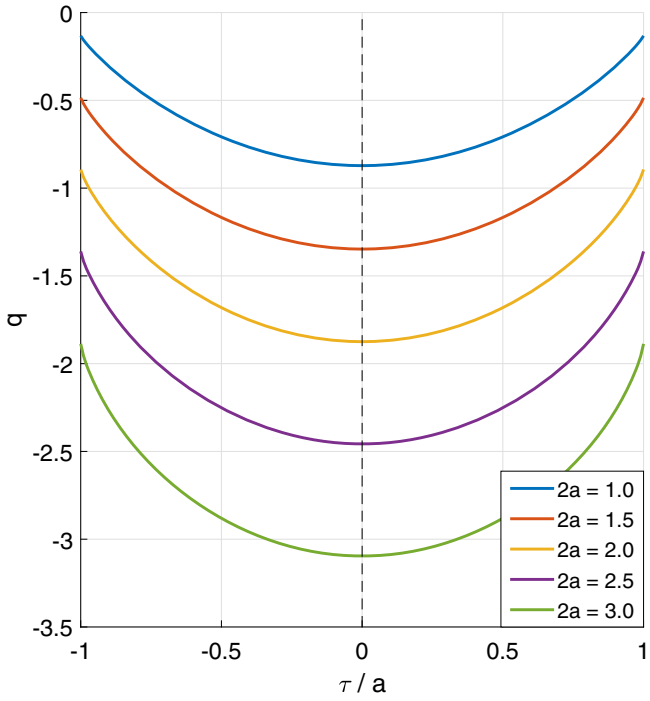


FIG. 4. Macroscopic velocity profiles for different values of a

TABLE 1. The macroscopic velocity profile $q(\tau)$ for a plan channel of half width $a = 1$, with $L = 50 \pm 2$, $M = 200$, and $N = 22$. All the digits match the benchmark published in Refs. [36,37]

τ	$\alpha = 0.50$	$\alpha = 0.80$	$\alpha = 0.88$	$\alpha = 0.96$	$\alpha = 1.00$
0.0	− 3.652222	− 2.319616	− 2.117410	− 1.948801	− 1.874577
0.1	− 3.644836	− 2.312148	− 2.109921	− 1.941293	− 1.867059
0.2	− 3.622577	− 2.289638	− 2.087351	− 1.918663	− 1.844401
0.3	− 3.585117	− 2.251759	− 2.049368	− 1.880582	− 1.806271
0.4	− 3.531852	− 2.197901	− 1.995366	− 1.826440	− 1.752062
0.5	− 3.461789	− 2.127072	− 1.924350	− 1.755244	− 1.680778
0.6	− 3.373321	− 2.037666	− 1.834718	− 1.665394	− 1.590822
0.7	− 3.263728	− 1.926991	− 1.723784	− 1.554211	− 1.479519
0.8	− 3.127917	− 1.790039	− 1.586566	− 1.416741	− 1.341927
0.9	− 2.954020	− 1.615281	− 1.411628	− 1.241643	− 1.166756
1.0	− 2.676407	− 1.340372	− 1.137528	− 9.683813e-1	− 8.939247e-1

Finally, we pushed the code increasing the discretization of space, velocity, and number of neurons. We were able to match all the seven digits reported in the most recent benchmark for the flow rate for the Poiseuille flow problem between two parallel plates [40], achieved with adding and doubling method. The results are reported in Tables 3. The parameters chosen and the computational time for each channel width are reported in Table 4. The parameters reported in Table 4 are also plotted in Fig. 5 to better appreciate their variations with respect to the channel thickness $2a$.

Table 2. The flow rate $Q(a)$ for [36] digits

$2a$	$\alpha = 0.50$	$\alpha = 0.80$	$\alpha = 0.88$	$\alpha = 0.96$	$\alpha = 1.00$
0.05	5.22330	3.08971	2.73834	2.43735	2.30226
0.10	4.55641	2.70774	2.40605	2.14824	2.03271
0.30	3.77847	2.24477	2.00107	1.79451	1.70247
0.50	3.54437	2.10227	1.87662	1.68634	1.60187
0.70	3.43767	2.03877	1.82201	1.63985	1.55919
0.90	3.38389	2.00924	1.79764	1.62022	1.54180
1.00	3.36822	2.00187	1.79206	1.61631	1.53868
2.00	3.37657	2.04139	1.83856	1.66937	1.59486
5.00	3.77440	2.43823	2.23506	2.06548	1.99077
7.00	4.08811	2.74611	2.54144	2.37038	2.29493
9.00	4.41019	3.06346	2.85756	2.68530	2.60925

Table 3. The flow rate $Q(a)$ for Ganapol digits [40]

$2a$	$\alpha = 0.50$	$\alpha = 0.80$	$\alpha = 0.88$	$\alpha = 0.96$	$\alpha = 1.00$
0.05	5.2232964	3.0897113	2.7383403	2.4373544	2.3022564
0.10	4.5564062	2.7077408	2.4060457	2.1482414	2.0327143
0.30	3.7784723	2.2447708	2.0010675	1.7945088	1.7024740
0.50	3.5443709	2.1022657	1.8766202	1.6863424	1.6018742
0.70	3.4376693	2.0387670	1.8220109	1.6398495	1.5591860
0.90	3.3838869	2.0092408	1.7976360	1.6202230	1.5417996
1.00	3.3682182	2.0018669	1.7920590	1.6163124	1.5386785
2.00	3.3765738	2.0413852	1.8385632	1.6693655	1.5948569
5.00	3.7744018	2.4382339	2.2350591	2.0654781	1.9907674
7.00	4.0881078	2.7461124	2.5414362	2.3703751	2.2949322
9.00	4.4101902	3.0634644	2.8575645	2.6852950	2.6092536

Table 4. Parameters used to compute the flow rate $Q(a)$ for Table 3 for each channel width

$2a$	N	M	L	Comp. time (s)
0.05	69	100	34	1.35
0.10	59	120	54	2.39
0.30	35	140	60	0.78
0.50	30	140	60	0.51
0.70	30	140	60	0.52
0.90	30	200	67	0.81
1.00	30	200	74	0.99
2.00	24	400	90	1.30
5.00	24	700	130	4.31
7.00	24	900	150	6.92
9.00	24	1400	150	10.12

7. Conclusions

We have developed and successfully applied the PINN-TFC method, Extreme Theory of Functional Connections, to solving the Poiseuille flow in a plane channel problem. The proposed method is proved to be accurate and efficient. To check the accuracy of our solutions, we have compared them with the benchmarks published in Refs. [6,37,40], matching their results up to the fifth and seventh digits. The

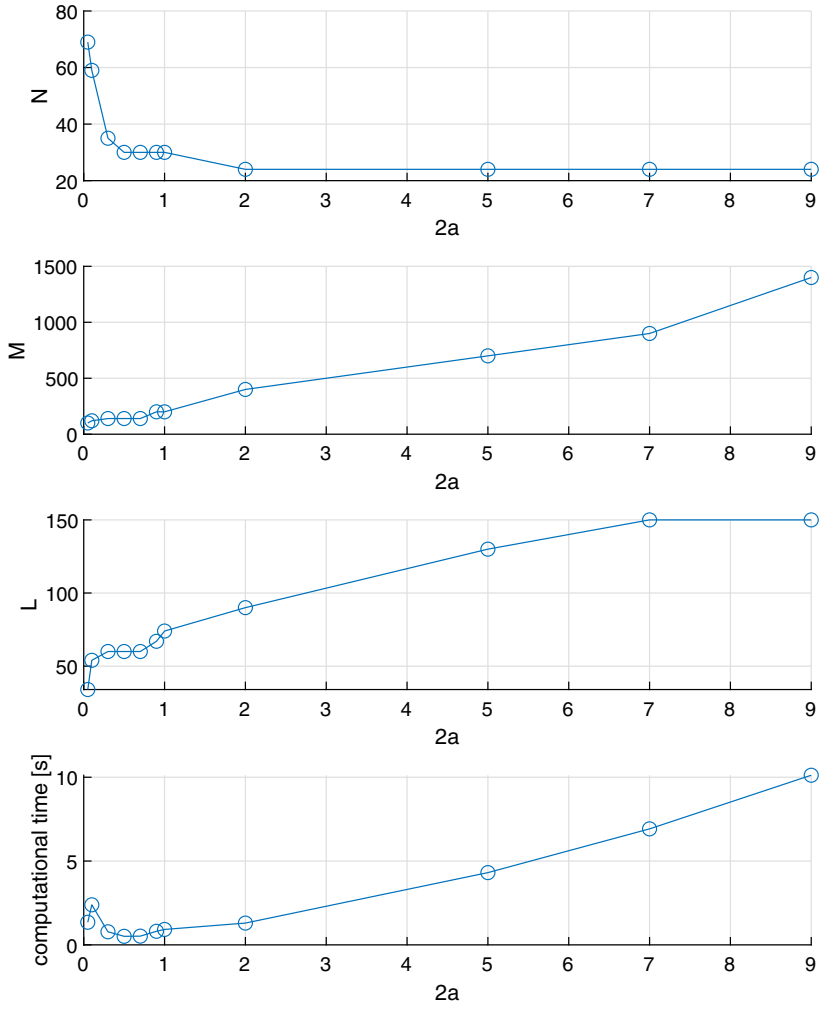


FIG. 5. Variations of the chosen parameters and computational times in function of the channel width reported in Table 4

results of this paper show the feasibility of the proposed methodology and demonstrate the accuracy of PINNs for a large variety of problems arising from the Boltzmann linear integrodifferential equation. We are currently working on applying the proposed X-TFC method to solving more RGD problems such as Couette flow in a plane channel [69] as formulated by Williams [7], and the thermal-jump problem. Moreover, we are currently working on solving high-dimensional problems with the X-TFC algorithm, which has been proved to be a significant improvement for the PINN frameworks for estimating solutions of differential equations [42], in particular for PDEs. Future works will then focus on applying this new method to solve 3D time-dependent problems arising from the Boltzmann linear integrodifferential equation in rarefied gas dynamics, radiative transfer, and neutron transport problems.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

[1] Reynolds, O.: Xviii. on certain dimensional properties of matter in the gaseous state.-Part I. Experimental researches on thermal transpiration of gases through porous plates and on the laws of transpiration and impulsion, including an experimental proof that gas is not a continuous plenum.-part ii. on an extension of the dynamical theory of gas, which includes the stresses, tangential and normal, caused by a varying condition of gas, and affords an explanation of the phenomena of transpiration and impulsion. Philos. Trans. R. Soc. Lond. **170**, 727–845 (1879)

[2] Maxwell, J.C.: Iii. on stresses in rarefied gases arising from inequalities of temperature. Proc. R. Soc. Lond. **27**(185–189), 304–308 (1878)

[3] Knudsen, M.: Die gesetze der molekularströmung und der inneren reibungsströmung der gase durch röhren. Ann. Phys. **333**(1), 75–130 (1909)

[4] Knudsen, M.: Thermischer molekulardruck der gase in röhren. Ann. Phys. **338**(16), 1435–1448 (1910)

[5] Loyalka, S.: Comments on "poiseuille flow and thermal creep of a rarefied gas between parallel plates"." Phys. Fluids **17**(5), 1053–1055 (1974)

[6] Loyalka, S., Petrellis, N., Storvick, T.: Some exact numerical results for the BGK model: Couette, Poiseuille and thermal creep flow between parallel plates. Z. Angew. Math. Phys. ZAMP **30**(3), 514–521 (1979)

[7] Williams, M.: A review of the rarefied gas dynamics theory associated with some classical problems in flow and heat transfer. Z. Angew. Math. Phys. ZAMP **52**(3), 500–516 (2001)

[8] Boffi, V., De Socio, L., Gaffuri, G., Pescatore, C.: Rigorous constructive solution to monodimensional poiseuille and thermal creep flows. Meccanica **11**(4), 183–190 (1976)

[9] Cercignani, C.: The boltzmann equation. In: The Boltzmann Equation and its Applications, pp. 40–103. Springer, New york (1988)

[10] Sharipov, F.: Rarefied Gas Dynamics: Fundamentals for Research and Practice. Wiley, New Jersey (2015)

[11] Ferziger, J., Kaper, H.: Mathematical theory of transport processes in gases. Am. J. Phys. **41**(4), 601–603 (1973)

[12] Cercignani, C.: Theory and Application of the Boltzmann Equation. Scottish Academic Press, UK (1975)

[13] Bhatnagar, P.L., Gross, E.P., Krook, M.: A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. Phys. Rev. **94**(3), 511 (1954)

[14] Shakhov, E.: Generalization of the krook kinetic relaxation equation. Fluid Dyn. **3**(5), 95–96 (1968)

[15] Loyalka, S.: Thermal transpiration in a cylindrical tube. Phys. Fluids **12**(11), 2301–2305 (1969)

[16] Loyalka, S., Storvick, T.: Kinetic theory of thermal transpiration and mechanocaloric effect. III. Flow of a polyatomic gas between parallel plates. J. Chem. Phys. **71**(1), 339–350 (1979)

[17] Chernyak, V., Porodnov, B., Suetin, P.: Application of the variational method to the problem of thermomolecular pressure difference in a cylindrical capillary. Inzh.-Fiz. Zh. **26**, 446–450 (1974)

[18] Valougeorgis, D., Thomas, J., Jr.: Exact numerical results for poiseuille and thermal creep flow in a cylindrical tube. Phys. Fluids **29**(2), 423–429 (1986)

[19] Loyalka, S., Hickey, K.: Kinetic theory of thermal transpiration and the mechanocaloric effect: Planar flow of a rigid sphere gas with arbitrary accommodation at the surface. J. Vac. Sci. Technol. A Vac. Surf. Films **9**(1), 158–163 (1991)

[20] Sharipov, F.: Rarefied gas flow through a long tube at any temperature ratio. J. Vac. Sci. Technol. A Vac. Surf. Films **14**(4), 2627–2635 (1996)

[21] Ritos, K., Lihnaropoulos, Y., Naris, S., Valougeorgis, D.: Pressure-and temperature-driven flow through triangular and trapezoidal microchannels. Heat Transf. Eng. **32**(13–14), 1101–1107 (2011)

[22] Ohwada, T., Sone, Y., Aoki, K.: Numerical analysis of the shear and thermal creep flows of a rarefied gas over a plane wall on the basis of the linearized boltzmann equation for hard-sphere molecules. Phys. Fluids A **1**(9), 1588–1599 (1989)

[23] Kanki, T., Iuchi, S.: Poiseuille flow and thermal creep of a rarefied gas between parallel plates. Phys. Fluids **16**(5), 594–599 (1973)

[24] Chandrasekhar, S.: Radiative Transfer. Courier Corporation, USA (2013)

[25] Barichello, L., Siewert, C.E.: A discrete-ordinates solution for a non-grey model with complete frequency redistribution. J. Quant. Spectrosc. Radiat. Transf. **62**(6), 665–675 (1999)

[26] Vilhena, M., Segatto, C., Barichello, L.: A particular solution for the sn radiative transfer problems. J. Quant. Spectrosc. Radiat. Transf. **53**(4), 467–469 (1995)

[27] Chien, K.-Y.: Application of the s, method to spherically symmetric radiative-transfer problems. AIAA J. **10**(1), 55–59 (1972)

[28] Simch, M., Segatto, C., Vilhena, M.: An analytical solution for the sn radiative transfer equation with polarization in a slab by the ltsn method. J. Quant. Spectrosc. Radiat. Transf. **97**(3), 424–435 (2006)

[29] Benoist, P., Kavenoky, A.: A new method of approximation of the boltzmann equation. Nucl. Sci. Eng. **32**(2), 225–232 (1968)

[30] Siewert, C., Benoist, P.: The fn method in neutron-transport theory. Part I: Theory and applications. Nucl. Sci. Eng. **69**(2), 156–160 (1979)

[31] Devaux, C., Siewert, C.: Thef n method for radiative transfer problems without azimuthal symmetry. Z. Angew. Math. Phys. ZAMP **31**(5), 592–604 (1980)
[32] Garcia, R., Siewert, C.: The fn method for radiative transfer models that include polarization effects. J. Quant. Spectrosc. Radiat. Transf. **41**(2), 117–145 (1989)
[33] Ganapol, B.D., Myneni, R.: The fn method for the one-angle radiative transfer equation applied to plant canopies. Remote Sens. Environ. **39**(3), 213–231 (1992)
[34] Benassi, M., Garcia, R., Karp, A., Siewert, C.: A high-order spherical harmonics solution to the standard problem in radiative transfer. Astrophys. J. **280**, 853–864 (1984)
[35] Siewert, C., Thomas, J., Jr.: A particular solution for the pn method in radiative transfer. J. Quant. Spectrosc. Radiat. Transf. **43**(6), 433–436 (1990)
[36] Barichello, L., Siewert, C.: A discrete-ordinates solution for poiseuille flow in a plane channel. Z. Angew. Math. Phys. ZAMP **50**(6), 972–981 (1999)
[37] Barichello, L., Camargo, M., Rodrigues, P., Siewert, C.: Unified solutions to classical flow problems based on the BGK model. Z. Angew. Math. Phys. ZAMP **52**(3), 517–534 (2001)
[38] Siewert, C.: A concise and accurate solution to chandrasekhar's basic problem in radiative transfer. J. Quant. Spectrosc. Radiat. Transf. **64**(2), 109–130 (2000)
[39] Ganapol, B.D.: The response matrix discrete ordinates solution to the 1d radiative transfer equation. J. Quant. Spectrosc. Radiat. Transf. **154**, 72–90 (2015)
[40] Ganapol, B.D.: Poiseuille channel flow by adding and doubling. In: AIP Conference Proceedings, vol. 1786, p. 070009. AIP Publishing LLC (2016)
[41] Ganapol, B.D.: 1d thermal creep channel flow in the bgk approximation by adding and doubling. Ann. Nucl. Energy **134**, 441–451 (2019)
[42] Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
[43] Schiassi, E., De Florio, M., D'ambrosio, A., Mortari, D., Furfaro, R.: Physics-informed neural networks and functional interpolation for data-driven parameters discovery of epidemiological compartmental models. Mathematics **9**(17), 2069 (2021)
[44] Cercignani, C., Daneri, A.: Flow of a rarefied gas between two parallel plates. J. Appl. Phys. **34**(12), 3509–3513 (1963)
[45] Cercignani, C.: Plane Poiseuille flow according to the method of elementary solutions. J. Math. Anal. Appl. **12**(2), 254–262 (1965)
[46] Schiassi, E., Furfaro, R., Leake, C., De Florio, M., Johnston, H., Mortari, D.: Extreme theory of functional connections: a fast physics-informed neural network method for solving ordinary and partial differential equations. Neurocomputing **457**, 334–356 (2021)
[47] Mortari, D.: The theory of connections: connecting points. Mathematics **5**(4), 57 (2017)
[48] Leake, C., Mortari, D.: Deep theory of functional connections: a new method for estimating the solutions of partial differential equations. Mach. Learn. Knowl. Extr. **2**(1), 37–55 (2020)
[49] Mortari, D.: Least-squares solution of linear differential equations. Mathematics **5**(4), 48 (2017)
[50] Mortari, D., Johnston, H., Smith, L.: High accuracy least-squares solutions of nonlinear differential equations. J. Comput. Appl. Math. **352**, 293–307 (2019)
[51] De Florio, M., Schiassi, E., D'Ambrosio, A., Mortari, D., Furfaro, R.: Theory of functional connections applied to linear odes subject to integral constraints and linear ordinary integro-differential equations. Math. Comput. Appl. **26**(3), 65 (2021)
[52] De Florio, M., Schiassi, E., Furfaro, R., Ganapol, B.D., Mostacci, D.: Solutions of chandrasekhar' s basic problem in radiative transfer via theory of functional connections. J. Quant. Spectrosc. Radiat. Transf. **259**, 107384 (2020)
[53] De Florio, M., Schiassi, E., Ganapol, B.D., Furfaro, R.: Physics-informed neural networks for rarefied-gas dynamics: thermal creep flow in the bhatnagar-gross-krook approximation. Phys. Fluids **33**(4), 047110 (2021)
[54] Furfaro, R., Mortari, D.: Least-squares solution of a class of optimal space guidance problems via theory of connections. Acta Astronaut. **168**, 92–103 (2019)
[55] Johnston, H., Schiassi, E., Furfaro, R., Mortari, D.: Fuel-Efficient Powered Descent Guidance on Large Planetary Bodies Via Theory of Functional Connections. arXiv preprint arXiv:2001.03572. (2020)
[56] Schiassi, E., D'Ambrosio, A., Johnston, H., Furfaro, R., Curti, F., Mortari, D.: Complete energy optimal landing on planetary bodies via theory of functional connections. Acta Astronaut. Prep. (2020)
[57] Drozd, K., Furfaro, R., Schiassi, E., Johnston, H., Mortari, D.: Energy-optimal trajectory problems in relative motion solved via theory of functional connections. Acta Astronaut. **182**, 361–382 (2021)
[58] D'Ambrosio, A., Schiassi, E., Curti, F., Furfaro, R.: Pontryagin neural networks with functional interpolation for optimal intercept problems. Mathematics **9**(9), 996 (2021)
[59] Namatame, A.: Connectionist learning with chebychev networks and analyses of its internal representation. In: Applications of Learning and Planning Methods, pp. 35–48. World Scientific, Singapore (1991)

[60] Mall, S., Chakraverty, S.: Single layer chebyshev neural network model for solving elliptic partial differential equations. Neural Process. Lett. **45**(3), 825–840 (2017)
[61] Russell, R., Shampine, L.F.: A collocation method for boundary value problems. Numer. Math. **19**(1), 1–28 (1972)
[62] Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing **70**, 489–501 (2006)
[63] Liu, M., Hou, M., Wang, J., Cheng, Y.: Solving two-dimensional linear partial differential equations based on chebyshev neural network with extreme learning machine algorithm. Eng. Comput. (2020)
[64] Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neural Netw. **9**, 987–1000 (1998)
[65] Wang, S., Teng, Y., Perdikaris, P.: Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM J. Sci. Comput. **43**(5), A3055–A3081 (2021)
[66] Mertikopoulos, P., Papadimitriou, C., Piliouras, G.: Cycles in adversarial regularized learning. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 2703–2717. SIAM (2018)
[67] Balduzzi, D., Racaniere, S., Martens, J., Foerster, J., Tuyls, K., Graepel, T.: The mechanics of n-player differentiable games. In: International Conference on Machine Learning, pp. 354–363. PMLR (2018)
[68] Siewert, C., Garcia, R., Grandjean, P.: A concise and accurate solution for Poiseuille flow in a plane channel. J. Math. Phys. **21**(12), 2760–2763 (1980)
[69] Sharipov, F.: Application of the cercignani-lampis scattering kernel to calculations of rarefied gas flows. I. Plane flow between two parallel plates. Eur. J. Mech.-B/Fluids **21**(1), 113–123 (2002)

Mario De Florio, Enrico Schiassi and Roberto Furfaro
Department of Systems and Industrial Engineering
The University of Arizona
Tucson AZ85721-0020
USA
e-mail: mariodf@email.arizona.edu

Enrico Schiassi
e-mail: eschiassi@email.arizona.edu

Barry D. Ganapol and Roberto Furfaro
Department of Aerospace and Mechanical Engineering
The University of Arizona
Tucson AZ85721-0020
USA
e-mail: robertof@email.arizona.edu

Barry D. Ganapol
e-mail: ganapol@cowboy.ame.arizona.edu

(Received: November 18, 2021; revised: April 20, 2022; accepted: April 21, 2022)

Appendix E

De Florio et al. (under review) Couette Flow

Reproduced from De Florio, M., Schiassi, E., Barichello, L. B., & Furfaro, R. (under-review). Numerical analysis of the plane Couette flow of a rarefied gas with Physics-Informed Neural Networks and Functional Interpolation. *Physics of Fluids* [11].

Numerical analysis of the plane Couette flow of a rarefied gas with Physics-Informed Neural Networks and Functional Interpolation

Mario De Florio,[1] Enrico Schiassi,[1] Liliane B. Barichello,[2] and Roberto Furfaro[1, 3]
[1)]*Department of Systems & Industrial Engineering, The University of Arizona, 1127 James E. Rogers Way, Tucson, AZ, 85719, USA*
[2)]*Instituto de Matemática e Estatística, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS 91509, Brazil*
[3)]*Department of Aerospace & Mechanical Engineering, The University of Arizona, 1130 N Mountain Ave, Tucson, AZ 85721, USA*

(*Electronic mail: robertof@arizona.edu)

(Dated: 8 November 2022)

The Boltzmann equation with the Bhatnagar-Gross-Krook collision model has been widely employed to describe the evolution of a gas through a kinetic theory. We propose a new Machine Learning approach to accurately and efficiently learn the solution of a class of problems in the transport theory of rarefied–gas dynamics, employing a particular Physics-Informed Neural Networks method called Extreme Theory of Functional Connections (X-TFC). According to X-TFC, the unknown function is approximated by the Constrained Expressions defined int the Theory of Functional Connections. Constrained Expressions are functionals (e.g., function of functions), which are the sum of a free function and a functional in which the constraints are analytically embedded. The constraints will therefore always be satisfied analytically, no matter what the free-chosen function is. In this work, a single-layer Neural Network trained with Extreme Learning Machine algorithm is employed. The proposed Machine Learning approach learns the solution of the Linear Boundary Value Problem arising from the Couette Flow problem in the Bhatnagar-Gross-Krook approximation between two parallel plates in relative motion, for a wide range of Knudsen numbers. The accuracy of X-TFC method is tested and compared with published benchmarks in literature.

I. INTRODUCTION

The Bhatnagar–Gross–Krook (BGK) model[1,2] has been widely studied in the last decades as a simplified model of the Boltzmann equation[3,4]. In particular, in two basic papers in the rarefied-gas dynamics (RGD), Cercignani and Daneri[5,6] explore the behavior of a gas in a rarefied-state flowing between two parallel walls from both a theoretical and numerical point of view. From these works, many others were born by different authors, where numerical methods have been applied to solve several classic RGD problems[7–16], such as Poiseuille, Thermal creep, and Couette flows, to name a few.
In this paper, we consider the linearized Couette flow simplified by the BGK approximation, which makes use of a statistical model for the molecular collisions[1]. Around 1960, in many works, the Couette problem was solved for a wide range of Knudsen numbers[17,18], creating comparable benchmarks for future works. Among these, we can mention Refs.[14,19–21] using the Analytical Discrete Ordinates (ADO) method, introduced by Barichello and Siewert in the field of radiative transfer[22]. The ADO method can be seen as a modern version of the discrete ordinates method proposed by Wick[23], and Chandrasekhar[24]. ADO is expected to be more efficient than the numerical one since the spatial variable is not discretized, and its approach does not require an iterative procedure.

A new numerical method called Physics-Informed Neural Networks (PINNs) is arousing a lot of interest in the machine learning community[25]. Neural Networks (NNs) are generally used to perform a regression on observed data representing the evolution of a physical quantity, by minimizing the mean squared error (MSE) between the data and the NN model, as a loss function. The novelty of PINN lies in including the knowledge of physics (represented by the residuals of the DE and its constraints) as a penalizer in the loss function. From the computer/data science perspective, PINNs define a data-physics-driven approach to performing data regression using NNs. From the physics perspective, PINNs define a data-physics-driven approach to solving DEs. Solving DEs with this data-physics-driven approach is a unique feature of PINNs (and in general, Physics-Informed Machine Learning PIML) as real data intrinsically hold information about perturbations and/or unmodeled terms of the real physics that were not accounted for in the modeled one via DE. In the absence of observed or synthetic data – which is the case with the present work – PINNs are built to learn the solutions of DEs solely in a physics-driven fashion.
The main shortcoming of the standard PINN framework, as presented by Raissi et al.[25], is that the DE constraints are not analytically satisfied, and then they need to be simultaneously learned with the DE solution inside the domain. This presence of competing objectives in the loss function can lead to a gradient flow pathologies in PINNs, causing the failure in learning the DEs constraints. Different approaches have been proposed to try to overcome this limitation. For example, in[26] a learning rate annealing algorithm is employed to adaptively assign proper weights to each term in the PINNs loss function during the network's training.

The authors of this manuscript developed a different PINN framework, which we will use to solve the Couette flow problem, to overcome this limitation of the standard PINN

methods. This framework brings together NN, specifically shallow NNs trained via the Extreme Learning Machine (ELM) algorithm and the Theory of Functional Connections (TFC). ELM is a training algorithm for shallow NN proposed by Huang et al.[27], in which the input weights and bias are randomly selected before the NN's training process, thus becoming NN's hyperparameters. Thus the only trainable parameters remaining are the output weights, learned via least-squares. TFC[28] is a mathematical framework for linear functional interpolation. It derives functionals including a free function and always satisfies a set of specified linear constraints regardless of the free function's choice. These functionals, called Constrained Expressions (CEs), reduce the solutions search space of constrained problems to solely the subspace of functions that analytically satisfy the specified linear constraints. Although TFC was recently developed (2017), it has found many applications, especially for solving Differential Equations (DEs), both Ordinary DE (ODE)[29–31] and Partial DEs (PDEs)[32,33]. When applied to DEs, the analytically satisfied constraints are the initial and/or boundary conditions, and the free function is usually an expansion of Chebyshev or Legendre polynomials. Orthogonal polynomials are generally a good choice for their approximating and convergence properties. For instance, Chebyshev polynomials generate a function that minimizes the maximum error in its application, and thus, they are well suited for approximating other functions[34,35]. TFC applied to DEs with Chebyshev or Legendre polynomials as free function is also known as Vanilla-TFC (V-TFC)[36,37]. The main issue of V-TFC is that when solving PDEs with more than two independent variables is affected by the "curse of dimensionality," e.g., the problem formulation and the computational cost become prohibitive. To this end, in[38] and[39], the authors propose the use of NN as the free function, as NN is not affected by the curse of dimensionality. Therefore, these TFC frameworks, as they employ NNs to learn DEs solutions, can be labeled as PINNs frameworks, and they are known as PINN-TFC based methods[39]. In[38] deep NNs are used as free functions, and the framework is called Deep-TFC. In[39] shallow NN trained via ELM algorithm are used as the free function, and the framework is called Extreme-TFC (X-TFC). X-TFC is already been successfully applied to learn the solutions of optimal control problems for space applications[40–46], numerical integration[31], problems in transport theory such as radiative transfer[47] and rarefied gas dynamics[48,49], and inverse problems for epidemiological models[50].

After this introduction, we present an overview on PINNs and X-TFC framework, and how the latter is used to solve linear ODEs, in section II. Section III presents the formulation of the Couette flow in a plane channel, and the X-TFC formulation of the problem is shown in section III A. Section IV presents the numerical results of several physical quantities on interest with discussions, and finally conclusions are stated in section V.

II. PHYSICS-INFORMED NEURAL NETWORKS

PINNs have been presented by Raissi et al.[25] to solve physics problems with three main approaches: physics-driven, data-physics-driven, and data-physics-driven PDEs' parameters discovery approaches. In a regular PINN framework, a deep neural networks (DNNs) is employed to learn the DE solution in strong form[51]. When PINNs are used solely in a physics-driven fashion, the training set consists of points to enforce the equation constraints on the boundary and the DE into the solution domain. The training points can be either randomly sampled or sampled following a specific quadrature scheme[52]. When PINNs are used in a data-physics-driven fashion, the training set is made for the physics-driven scenario and must include the data points from the experimental data-set[25]. The DE solution, approximated via DNN, is plugged in the DE's residual and initial/boundary conditions' residuals. The DNN is next trained via gradient-based methods (usually Adam optimizer) in learning the network hyperparameters (e.g., input weights and bias) that minimize the physics-based loss function made by the DE and its constraints. Hence, the characteristic feature of PINNs is to embed, in the loss function, the problem's knowledge of the physics acquired from the DE and its constraints. The interested reader, to better understand how classic PINNs work, can find the detailed step-by-step procedure to solve a first-order linear ODE in Ref.[48].

As previously mentioned, the main drawback of the classic PINN frameworks is that the DE's constraints are not analytically satisfied. Hence, these constraints must be learned simultaneously with the DE solution within the domain during the NN's training. This leads to having competing objectives during the network's training, which, as explained above, can cause the PINN's failure in learning the DE's correct solution. The method we employ in this work completely removes this issue by combining NN with the TFC. TFC allows satisfying the equation constraints analytically. The PINN-TFC method employed in this work is X-TFC. The following subsection gives a detailed explanation of how X-TFC works for solving a general linear ODE.

A. X-TFC Approach to Solving First Order Linear ODEs

Consider the following ODE as example:

$$\mathscr{D}(t,u,\dot{u}) = 0 \quad \text{subject to:} \quad u(t_0) = u_0 \tag{1}$$

The CE defined in the TFC framework is built accordingly to approximate the solution of the unknown function $u(t)$ as follows:

$$u(t) = g(t) + \sum_{k=1}^{n} \eta_k p_k(t) = g(t) + \boldsymbol{\eta}^T \boldsymbol{p}(t) \tag{2}$$

where, n is the number of constraints, $p_k(t)$ are assigned linearly independent functions and the $g(t)$ is the free-chosen function. The values of the η_k coefficients are retrieved by

forcing a set of n linear constraints in $u(t)$, to analitically satisfy the DE's constraints, regardless of the choice of the free function $g(t)$. When facing a linear ODE with initial condition, the CE is built as follows:

$$u(t) = g(t) + \eta\, p(t) \tag{3}$$

where $p(t) = 1$ (for the details about this selection, see ref.[29,30]). Hence the CE is:

$$u(t) = g(t) + \eta \tag{4}$$

By forcing the initial condition, we have:

$$u(t_0) = u_0 = g_0 + \eta \qquad \Longrightarrow \qquad \eta = u_0 - g_0 \tag{5}$$

which is substituted into Eq. (4) to get the CE with embedded constraints:

$$u(t) = g(t) + (u_0 - g_0) \tag{6}$$

representing all the possible functions satisfying the constraints. Then, the derivative is:

$$\dot{u}(t) = \dot{g}(t) \tag{7}$$

The CE in the form of Eq. (6) and its derivative are replaced into the ODE to get:

$$\mathscr{D}(t, u, \dot{u}) = 0 \tag{8}$$

that is now an unconstrained problem. By plugging Eq. (6) into Eq. (8), we obtain a new DE as a function of t and $g(t)$, i.e.:

$$\tilde{\mathscr{D}}(t, g, \dot{g}) = 0 \tag{9}$$

In this work, we employ a single-layer NN as free function $g(t)$, such as:

$$g(t) = \sum_{j=1}^{L} \beta_j \sigma(w_j t + b_j) = \begin{bmatrix} \sigma(w_1 t + b_1) \\ \vdots \\ \sigma(w_L t + b_L) \end{bmatrix}^T \boldsymbol{\beta} = \boldsymbol{\sigma}^T \boldsymbol{\beta} \tag{10}$$

where L is the number of neurons, $w_j \in \mathbb{R}$ is the j^{th} input weight connecting the input node with the j^{th} neuron, $\beta_j \in \mathbb{R}$ with $j = 1, ..., L$ is the j^{th} output weight connecting the output node with the j^{th} neuron, b_j is the bias of the j^{th} neuron, and $\sigma_j(\cdot)$ is the NN's activation function, which is selected by the user. As mentioned before, this is a single-layer NN trained via ELM algorithm[27], thus the only unknown parameters of Eq. (10) that need to be computed are the output weights $\boldsymbol{\beta} = [\beta_1, ..., \beta_L]^T$. When using a Chebyshev NN (ChNN), the choice of input weights and biases that guarantees the best accuracy is to set $w_j = 1$ and $b_j = 0, \forall j = 1, ..., L$[53]. For this case, Eq. (10) can be rewritten as follows:

$$g(t) = \sum_{j=1}^{L} \beta_j P_j(t) = \begin{bmatrix} P_1(t) \\ \vdots \\ P_L(t) \end{bmatrix}^T \boldsymbol{\beta} = \boldsymbol{P}^T \boldsymbol{\beta} \tag{11}$$

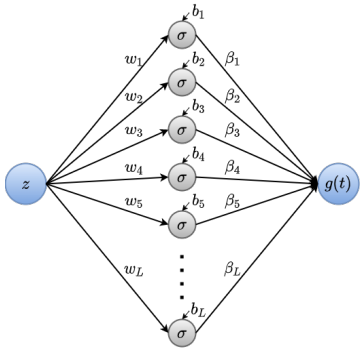


FIG. 1: Single-layer Neural Network with L neurons.

Where P_j for $j+1, ..., L$ is the Chebyshev polynomial of order j. Figure (1) shows the schema of the single-layer NN used in this manuscript as CE's free-chosen function. When the activation functions are defined on an inconsistent domain $z \in [z_0, z_f]$, the independent variable $t \in [t_0, t_f]$ needs to be mapped into the z domain, with a linear transformation, such as:

$$z = z_0 + c(t - t_0) \qquad \longleftrightarrow \qquad t = t_0 + \frac{1}{c}(z - z_0)$$

where the mapping coefficient is defined as:

$$c = \frac{z_f - z_0}{t_f - t_0}$$

According to the derivative chain rule, the n^{th} derivative of Eq. (10) is defined as,

$$\frac{\mathrm{d}^n g(t)}{\mathrm{d}t^n} = c^n \boldsymbol{\beta}^T \frac{\mathrm{d}^n \boldsymbol{\sigma}(z)}{\mathrm{d}z^n}, \tag{12}$$

After discretizing the t domain with M discretization points, the unconstrained optimization problem will have the following form

$$\tilde{\mathscr{D}}(\boldsymbol{\beta}) = 0 \tag{13}$$

and since it is a linear problem, can be rearranged into a linear system of algebraic equations $A\boldsymbol{\beta} = \boldsymbol{b}$, which can be solved via least-squares methods to find the optimal NN's output weights vector $\boldsymbol{\beta}$. To better understand the steps of the X-TFC framework for linear ODEs, a simplified schema is reported in Fig. 2.

III. COUETTE FLOW IN A PLANE CHANNEL: X-TFC PROBLEM FORMULATION

In this work, we consider a steady Couette flow confined between two parallel plates in relative motion between them, with opposite velocities u_w (here considered equal to 1), as shown in Fig. 3. This relative motion imposes a shear stress

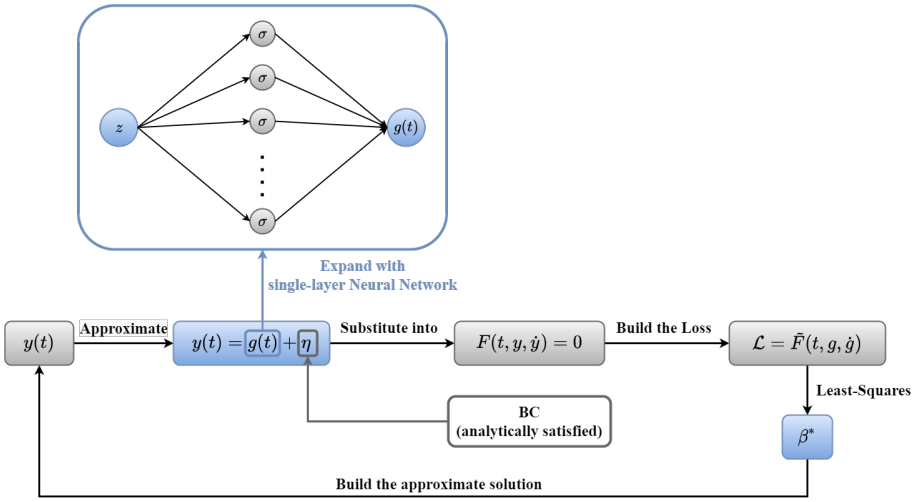


FIG. 2: Schematic of X-TFC framework to solve linear ODEs with one constraint in one point.

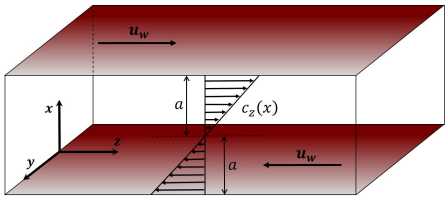


FIG. 3: Couette microchannel flow.

on the fluid which causes its flow. The plane parallel walls are separated by a distance d, or two semi-distances a. The problem we aim to solve follows the formulation of the Couette flow as presented by Williams[54]. The full non-linear Boltzmann equation for an atomic gas can be written as

$$\boldsymbol{v}\cdot\nabla_r f(\boldsymbol{r},\boldsymbol{v}) = J(f,f') \tag{14}$$

where J is the collision operator defined in Refs.[55,56], and $f(\boldsymbol{r},\boldsymbol{v})$ is the space and velocity distribution function of the atoms

$$f(\boldsymbol{r},\boldsymbol{v}) = f_0(\boldsymbol{r},\boldsymbol{v})[1+h(\boldsymbol{r},\boldsymbol{v})] \tag{15}$$

Here, h is a perturbation to the local Maxwellian $f_0(\boldsymbol{r},\boldsymbol{v})$ caused by the plates. The general form of the local Maxwellian is

$$f_0(\mathbf{r},\mathbf{v}) = n_\infty(x,z)\left[\frac{m}{2\pi kT_\infty(x,z)}\right]^{3/2} \exp\left[-\frac{m}{2\pi kT_\infty(x,z)}\left\{v_x^2+v_y^2+[v_z-\bar{u}(x)]^2\right\}\right] \tag{16}$$

With some assumptions on n_∞, T_∞, and $\bar{u}$[54], and converting the velocity v to the non-dimensional velocity variable

$$\boldsymbol{c} = \boldsymbol{v}\left(\frac{m}{2kT_0}\right)^{1/2}$$

we can substitute Eq. (15) into Eq. (14) to get

$$c_x\left[\left(c^2-\frac{3}{2}\right)K_x + R_x + 2c_zK_0\right] + c_z\left(c^2-\frac{3}{2}\right)K_z + c_zR_z + c_x\frac{\partial h(x,\boldsymbol{c})}{\partial x} + V(c)h(x,\boldsymbol{c}) = \int e^{-c'^2}K(\boldsymbol{c},\boldsymbol{c}')h(x,\boldsymbol{c}')d\boldsymbol{c}' \tag{17}$$

where

$$K_0 = K\left(\frac{m}{2kT_0}\right)^{1/2}$$

Here, $R_{x,z}$ and $K_{x,z}$ represent the relative density and temperature gradients in the x and z directions, respectively, and K is the gradient of c_z in the x-direction. $V(c)$ is the collision frequency, which is assumed constant and independent on the velocity for the BGK model:

$$V(c) = 1$$

Finally, the previous equation becomes[54]

$$c_x\left[\left(c^2-\frac{3}{2}\right)K_x + R_x + 2c_zK_0\right] + c_z\left(c^2-\frac{3}{2}\right)K_z + c_zR_z + c_x\frac{\partial h(x,\mathbf{c})}{\partial x} + h(x,\mathbf{c}) = = \int e^{-c'^2}h(x,\mathbf{c}')\left[1+2\boldsymbol{c}\cdot\boldsymbol{c}' + \frac{2}{3}\left(c^2-\frac{3}{2}\right)\left(c'^2-\frac{3}{2}\right)\right]d\mathbf{c}'$$

When a fluid dynamics problem involves the velocity of the gas rather than the temperature, by definition the gas velocity in the z-direction is given by

$$q(x) = \frac{\int v_z f(x,\boldsymbol{v}) d\boldsymbol{v}}{\int f(x,\boldsymbol{v}) d\boldsymbol{v}} \tag{19}$$

Now, assuming that the temperature and density are constant, and linearizing, we get

$$q(x) = \frac{1}{n_0} \int f_{00}(\boldsymbol{v}) v_z \left[h(x,\boldsymbol{v}) + \frac{m v_z K x}{k T_0} \right] d\boldsymbol{v} \tag{20}$$

where $f_{00}(\boldsymbol{v})$ is the equilibrium Maxwellian distribution function. Eq. (20) reduces to

$$q(x) = K_0 x + \frac{1}{\pi^{3/2}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} c_z e^{-c^2} h(x,c_x,c_y,c_z) dc_x dc_y dc_z \tag{21}$$

that is in units of $(2kT_0/m)^{1/2}$. Let's introduce the following function

$$Z(x,c_x) = \frac{1}{\pi} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} c_z e^{-c_y^2 - c_z^2} h(x,c_x,c_y,c_z) dc_y dc_z \tag{22}$$

then, Eq. (21) can be rewritten as

$$q(x) = K_0 x + \frac{1}{\pi^{1/2}} \int_{-\infty}^{\infty} e^{-c_x^2} Z(x,c_x) dc_x \tag{23}$$

Now, we can multiply Eq. (18) by c_x and integrate over c_y and c_x, we get

$$K_0 c_x + \frac{1}{2} R_z + \frac{1}{2} K_z \left(c_x^2 + \frac{1}{2} \right) + c_x \frac{\partial Z(x,c_z)}{\partial x} + Z(x,c_x) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-c_x'^2} Z(x,c_x') dc_x' \tag{24}$$

As we mentioned in the beginning of this section, the problem for the Couette flow is to have the upper and lower plates traveling at velocity u_w in the directions z and $-z$, respectively. Thus, the relative density and temperature gradients in the ith-direction, R_i and K_i, are null. Now, we can make use of the extension[54] of the formulations adopted in references[9] and[6] to the case of the accommodation coefficient α ranging from 0 to 1, and so we look for a solution of following linear integro-differential equation:

$$u \frac{\partial Z(\tau,u)}{\partial \tau} + Z(\tau,u) = \int_{-\infty}^{\infty} \Psi(u') Z(\tau,u') du' \tag{25}$$

where $\tau \in [0,a]$ is the half-width of the channel (thanks to the spatial symmetry of the problem, which will ease the computational efforts), $u \in (-\infty,\infty]$ is the cosine direction of the gas molecules, and $Z(\tau,\mu)$ is the unknown function of the differential equation, representing the velocity momentum. The problem is then subject to the following boundary conditions, whose derivations can be found by the interested reader in Ref.[54]:

$$\begin{cases} Z(0,u) = -Z(0,-u) \\ Z(a,-u) = (1-\alpha) Z(a,u) - \alpha \end{cases} \tag{26}$$

for $u \in (0,\infty)$. The weight function $\Psi(u)$ is defined as follows

$$\Psi(u) = \pi^{-1/2} e^{-u^2} \tag{27}$$

Therefore, this work aims to solve the problem given by Eqs. (25) and (26) using the X-TFC framework.

A. X-TFC formulation of the problem

Considering the integro-differential equation (25) with constraints (26), we rewrite them according to the mapping

$$cu \frac{\partial Z(\chi,u)}{\partial \chi} + Z(\chi,u) = \int_{-\infty}^{\infty} \Psi(u') Z(\chi,u') du' \tag{28}$$

By following Ref.[22], it is convenient to break the equation in two parts and eliminate the dependence on the velocity u by we can discretizing it into N points:

$$cu_i \frac{d}{d\chi} Z(\chi,u_i) + Z(\chi,u_i) = \sum_{k=1}^{N} \omega_k \frac{1}{u_k} \Psi(u_k) \left[Z(\chi,u_k) + Z(\chi,-u_k) \right] \tag{29}$$

$$-cu_i \frac{d}{d\chi} Z(\chi,-u_i) + Z(\chi,-u_i) = \sum_{k=1}^{N} \omega_k \frac{1}{u_k} \Psi(u_k) \left[Z(\chi,u_k) + Z(\chi,-u_k) \right] \tag{30}$$

In this way, we are considering that the N weights ω_k and quadrature points u_k are defined for use on the integration interval $[0,\infty)$. We can use a further mapping transformation to map the interval $[0,\infty)$ onto $[0,1]$ by using the new following variable:

$$u = -\ln(|\mu|)$$

which leads to $du = -\dfrac{1}{|\mu|} d\mu$, and the characteristic function $\Psi(u)$ becomes

$$\Psi(\mu) = \pi^{-1/2} e^{-(-\ln(|\mu|))^2}$$

in order to use a Gauss–Legendre scheme mapped onto the interval $[0,1]$.
We can reformulate our problem as

$$-c\ln(\mu_i) \frac{d}{d\chi} Z(\chi,\mu_i) + Z(\chi,\mu_i) = \sum_{k=1}^{N} \omega_k \frac{1}{\mu_k} \Psi(\mu_k) \left[Z(\chi,\mu_k) + Z(\chi,-\mu_k) \right] \tag{31}$$

$$c\ln(\mu_i) \frac{d}{d\chi} Z(\chi,-\mu_i) + Z(\chi,-\mu_i) = \sum_{k=1}^{N} \omega_k \frac{1}{\mu_k} \Psi(\mu_k) \left[Z(\chi,\mu_k) + Z(\chi,-\mu_k) \right] \tag{32}$$

subject to:

$$\begin{cases} Z(0,\mu_i) = -Z(0,-\mu_i) \\ Z(a,-\mu_i) = (1-\alpha)Z(a,\mu_i) - \alpha \end{cases} \tag{33}$$

Here, we take advantage of the symmetry of the flow in the channel, given by the first expression in the system (33) above. For the reader's convenience, we suppress the variables' dependence of the unknown functions by making use of $^{+,-}$ superscripts, and $_{i,k}$ subscripts

$$-c\ln(\mu_i)\frac{d}{d\chi}Z_i^+ + Z_i^+ = \sum_{k=1}^{N}\omega_k\frac{1}{\mu_k}\Psi(\mu_k)\left(Z_k^+ + Z_k^-\right) \tag{34}$$

$$c\ln(\mu_i)\frac{d}{d\chi}Z_i^- + Z_i^- = \sum_{k=1}^{N}\omega_k\frac{1}{\mu_k}\Psi(\mu_k)\left(Z_k^+ + Z_k^-\right) \tag{35}$$

subject to:

$$\begin{cases} Z_0^+ = -Z_0^- \\ Z_f^- = (1-\alpha)Z_f^+ - \alpha \end{cases} \tag{36}$$

Since we are dealing with a problem made up of two ODEs with one boundary condition each, we can build the constrained expressions according to Eq. (2):

$$y(\tau) = g(\tau) + \eta\, p(\tau). \tag{37}$$

The vector $p(\tau)$ is chosen to be a vector of 1s, according to Mortari[28], and the free function $g(\tau)$ is the single-layer NN:

$$g(\tau) = \boldsymbol{\sigma}^T(\tau)\boldsymbol{\beta} \tag{38}$$

Since the activation functions range in particular domains, it is appropriate to map the independent variable τ to a new independent variable, χ, that ranges in the activation function domain, e.g. $[-1,1]$. Thus, χ is defined as

$$\chi = c(\tau + a) - 1$$

where the constant c is called mapping constant, that is

$$c = \frac{\chi_f - \chi_0}{\tau_f - \tau_0} = \frac{1}{a}$$

It only comes into action when we need to compute the derivative of the unknown function. Then, our mapping transformation gives us

$$Z(\tau,u) = Z(\chi,u) \qquad \text{and} \qquad \frac{\partial}{\partial\tau}Z(\tau,u) = c\frac{\partial}{\partial\chi}Z(\chi,u)$$

According to Eq. (37), the constrained expressions are

$$Z^{\pm}(\chi) = \boldsymbol{\sigma}^{\pm}\boldsymbol{\beta}^{\pm} + \eta^{\pm} \tag{39}$$

To find the form of the $\eta_i^{\pm}$, we use the boundary conditions of the problem as follows

$$Z_0^+ = \boldsymbol{\sigma_0}^T\boldsymbol{\beta}^+ + \eta^+ \qquad\qquad Z_f^- = \boldsymbol{\sigma_f}^T\boldsymbol{\beta}^- + \eta^-$$

$$Z_f^+ = \boldsymbol{\sigma_f}^T\boldsymbol{\beta}^+ + \eta^+ \qquad\qquad Z_0^- = \boldsymbol{\sigma_0}^T\boldsymbol{\beta}^- + \eta^-$$

that replaced in the Eqs. (36) we have

$$\begin{cases} \boldsymbol{\sigma_0}^T\boldsymbol{\beta_i}^+ + \eta_i^+ = \boldsymbol{\sigma_0}^T\boldsymbol{\beta_i}^- + \eta_i^- \\ \boldsymbol{\sigma_f}^T\boldsymbol{\beta_i}^- + \eta_i^- = (1-\alpha)\,\boldsymbol{\sigma_f}^T\boldsymbol{\beta_i}^+ + (1-\alpha)\,\eta_i^+ - \alpha \end{cases} \tag{40}$$

that gives us

$$\begin{cases} \eta_i^+ = [\theta\boldsymbol{\sigma_0} + \theta(1-\alpha)\boldsymbol{\sigma_f}]^T\boldsymbol{\beta_i}^+ + [\theta\boldsymbol{\sigma_0} - \theta\boldsymbol{\sigma_f}]^T\boldsymbol{\beta_i}^- - \theta\alpha \\ \eta_i^- = [-\varphi\boldsymbol{\sigma_0} + \theta(1-\alpha)\boldsymbol{\sigma_f}]^T\boldsymbol{\beta_i}^+ + [-\varphi\boldsymbol{\sigma_0} + \theta\boldsymbol{\sigma_f}]^T\boldsymbol{\beta_i}^- + \theta\alpha \end{cases} \tag{41}$$

where θ and φ are parameters introduced to simplify the notation:

$$\theta = \frac{1}{\alpha - 2} \qquad\qquad \varphi = \frac{\alpha - 1}{\alpha - 2}$$

Now that $\eta_i^{\pm}$ are found, the CEs can be built in the following form:

$$\begin{aligned} Z_i^+ &= (\boldsymbol{\sigma} + \theta\boldsymbol{\sigma_0} - \varphi\boldsymbol{\sigma_f})^T\boldsymbol{\beta_i}^+ + \theta\left(\boldsymbol{\sigma_0} - \boldsymbol{\sigma}_f\right)^T\boldsymbol{\beta_i}^- - \theta\alpha \\ Z_i^- &= \varphi\left(\boldsymbol{\sigma}_f - \boldsymbol{\sigma_0}\right)^T\boldsymbol{\beta_i}^+ + \left(\boldsymbol{\sigma} - \varphi\boldsymbol{\sigma_0} + \theta\boldsymbol{\sigma}_f\right)^T\boldsymbol{\beta_i}^- + \theta\alpha \end{aligned} \tag{42}$$

and replacing them in the DEs of the problem, we get:

$$\begin{aligned} &(-c\ln(\mu_i)\boldsymbol{\sigma'} + \boldsymbol{\sigma} + \theta\boldsymbol{\sigma_0} - \varphi\boldsymbol{\sigma_f})^T\boldsymbol{\beta_i}^+ \\ &\qquad + \theta\,(\boldsymbol{\sigma_0} - \boldsymbol{\sigma_f})^T\boldsymbol{\beta_i}^- - \sum_{k=1}^{N}\omega_k\frac{1}{\mu_k}\Psi(\mu_k) \\ &\left\{[\boldsymbol{\sigma} + (\theta-\varphi)\boldsymbol{\sigma_0}]^T\boldsymbol{\beta_k}^+ + [\boldsymbol{\sigma} + (\theta-\varphi)\boldsymbol{\sigma_0}]^T\boldsymbol{\beta_k}^-\right\} = \theta\alpha \end{aligned} \tag{43}$$

$$\begin{aligned} &(c\ln(\mu_i)\boldsymbol{\sigma'} + \boldsymbol{\sigma} - \varphi\boldsymbol{\sigma_0} + \theta\boldsymbol{\sigma_f})^T\boldsymbol{\beta_i}^- \\ &\qquad + \varphi\,(\boldsymbol{\sigma_0} - \boldsymbol{\sigma_f})^T\boldsymbol{\beta_i}^+ - \sum_{k=1}^{N}\omega_k\frac{1}{\mu_k}\Psi(\mu_k) \\ &\left\{[\boldsymbol{\sigma} + (\theta-\varphi)\boldsymbol{\sigma_0}]^T\boldsymbol{\beta_k}^+ + [\boldsymbol{\sigma} + (\theta-\varphi)\boldsymbol{\sigma_0}]^T\boldsymbol{\beta_k}^-\right\} = -\theta\alpha \end{aligned} \tag{44}$$

Thus, the original boundary value constrained problem has been turned into an unconstrained optimization problem $\boldsymbol{A\beta} = \boldsymbol{B}$, where the unknown vector β is computed via Least-Squares, and the solutions are computed according to CEs expressed in Eqs. (42). More specifically, $\beta = [\beta^+, \beta^-]$.

IV. RESULTS AND DISCUSSIONS

A qualitative plot of the velocity momentum $Z(\mu,\tau)$ for $a = 0.5$ and $\alpha = 1$ is given in Figure 4, obtained with activation function *tanh* and showing the spatial symmetry of the Couette flow. In the Couette problem, two quantities are interesting to compute to prove the efficiency of a new numerical method. One involves the gas molecular velocity between

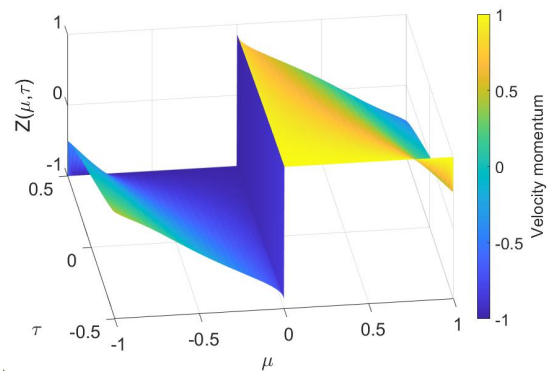


FIG. 4: Velocity momentum in a channel of semi-width $a = 0.5$ and accommodation coefficient $\alpha = 1$.

the channel's walls. The definition of its velocity in the z-direction, as previously mentioned, is given by

$$q(x) = K_0 x + \frac{1}{\pi^{1/2}} \int_{-\infty}^{\infty} e^{-c_x^2} Z(x, c_x) dc_x \tag{45}$$

For the Couette flow assumptions[8], it becomes

$$q(x) = \frac{1}{\pi^{1/2}} \int_{-\infty}^{\infty} e^{-u^2} Z(x, u) du \tag{46}$$

which we rewrite as

$$q(\tau) = \int_{-\infty}^{\infty} \Psi(u) Z(\tau, u) du \qquad \text{for} \qquad \tau \in [0, a] \tag{47}$$

By making use of change of variable we get

$$q(\tau) = \int_{-1}^{1} \frac{1}{\mu} \Psi(\mu) Z(\tau, \mu) d\mu \tag{48}$$

which can be computed with a Gauss-Legendre quadrature:

$$q(\tau) = \sum_{i=1}^{N} \omega_i \frac{1}{\mu_i} \Psi(\mu_i) \left(Z_i^+ + Z_i^- \right) \tag{49}$$

Once we have the expression of the velocity profile, the mass flow rate for half channel can be easily calculated as follows:

$$Q = \frac{1}{2a^2} \int_0^a q(\tau) d\tau \tag{50}$$

In Tables Ia and Ib, we show the compatibility of our results with the ones provided by Scherer et al.[20]. For the velocity profile, the results are accordant up to the fifth digit almost in all the channel points, while for the mass flow rate, all the digits of accuracy are reached for each channel width. Since there is not a vast availability of benchmarks in the literature for the BGK model of Couette flow, we propose here a new benchmark for the macroscopic velocity for a wide range of Knudsen numbers and different values of accommodation coefficient α. The results we print here are shown in Table II, which we believe to be accurate up to the 7th digit given the similarity with the thermal creep problem already solved with the same method[48,49]. The parameters used and the computational times are shown in Table III, and obtained with Matlab R2021a with an Intel Core i7 - 9700 CPU PC with 64 GB of RAM. In the plots of Figure 5, the trend of the macroscopic velocity profile is plotted for two different case scenarios: in Figure 5a the value of the accommodation coefficient is fixed ($\alpha = 1$) while the channel width $2a$ (inverse Knudsen number) varies in the range $[0.1, 100]$. As one can see, the velocity profile reaches higher values for greater inverse Knudsen numbers. In Figure 5b, the inverse Knudsen number is fixed ($2a = 2$), while the accommodation coefficient varies in a range $[0.2, 1]$. As one can expect, the velocity profile reaches higher values for greater accommodation coefficients since the latter quantifies the interaction between the fluid and the wall, or also the degree to which the gas molecules are accommodated to the solid surface[57]. The second macroscopic quantity of interest consists in the pressure tensor[54], or stress tensor, caused in the gas by the motion of the plates[55]. Its xz component is given by

$$P_{xz} = \frac{n_0}{\pi^{3/2}} \frac{2kT_0}{m} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c_x c_z e^{-c^2} h(x, \boldsymbol{c}) dc_x dc_y dc_z \tag{51}$$

and following Barichello et al.[14], we wish to compute the (constant) normalized stress in the form

$$P_{xz} = \pi^{1/2} \int_{-\infty}^{\infty} \Psi(u) Z(\tau, u) u \, du \tag{52}$$

Again, we make use of the change of variable for the velocity as

$$P_{xz} = \pi^{1/2} \int_{-1}^{1} -\frac{1}{\mu} \Psi(\mu) Z(\tau, \mu) \ln(|\mu|) d\mu \tag{53}$$

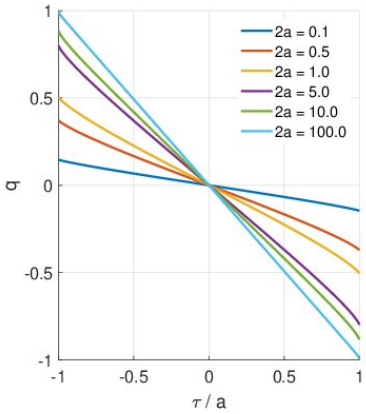
which can be computed with a Gauss-Legendre quadrature:

$$P_{xz} = \pi^{1/2} \sum_{i=1}^{N} -\omega_i \Psi(\mu_i) \ln(\mu_i) \left(\frac{1}{\mu_i} Z_i^+ + \frac{1}{-\mu_i} Z_i^- \right) \tag{54}$$

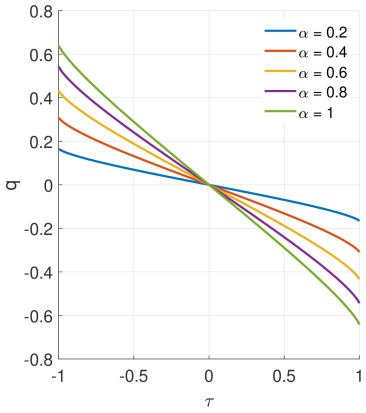
Our results are reported in Table IV for a wide range of inverse Knudsen numbers (from 10^{-7} to 20). By producing these results, we can confirm the accuracy of the results reported in Ref.[14] up to 7th digit. Also, the comparison with other two benchmarks produced by Loyalka et al.[9] and Willis[18], where the number of digits of accuracy is limited, is reported.

V. CONCLUSIONS

This work highlighted the accuracy, efficiency, and robustness of the Extreme Theory of Functional Connections framework[39] to solve classic flow problems in rarefied-gas dynamics, in particular the Couette flow. The accuracy has been proved by comparison with benchmarks published by Scherer et al.[20], Barichello et al.[14], Loyalka et al.[9], and Willis[18], matching their solutions up to the $7th$ digit. In this work, we



(a) Macroscopic velocity profiles for different values of a, when $\alpha = 1$.



(b) Macroscopic velocity profiles for different values of α, when $a = 1$.

FIG. 5: Qualitative behaviour of the macroscopic velocity profile for a given accommodation coefficient α (left) and for a given channel width (right).

presented a new benchmark for the macroscopic velocity profile on the wall for a wide range of inverse Knudsen numbers and different values of the accommodation coefficient, obtaining accuracy that we believe to be up to the $7th$ digit. Also, the pressure tensor has been evaluated, obtaining a precise benchmark according to Ref.[14].

This work has proven the feasibility of this new PINN method (that uses TFC constrained expressions, thanks to which the boundary values are analytically satisfied) to solve particle transport problems arising from the Boltzmann integro-differential equation. The BGK model has been used to approximate the Boltzmann equation, but more problems using other models are already under investigation. Also, the

TABLE I: Comparison between X-TFC results and benchmarks proposed in Ref.[20]

(a) The macroscopic velocity profile $q(\tau)$ for a plane channel of width $2a = 1$ and $\alpha = 1$.

τ/a	X-TFC	Ref.[20]
0.0	0.0	0.0
0.1	–4.44498(–2)	–4.44498(–2)
0.2	–8.90639(–2)	–8.90638(–2)
0.3	–1.34020(–1)	–1.34019(–1)
0.4	–1.79526(–1)	–1.79525(–1)
0.5	–2.25844(–1)	–2.25844(–1)
0.6	–2.73338(–1)	–2.73338(–1)
0.7	–3.22559(–1)	–3.22558(–1)
0.8	–3.74467(–1)	–3.74467(–1)
0.9	–4.31190(–1)	–4.31189(–1)
1.0	–5.03723(–1)	–5.03722(–1)

(b) The mass flow rate $Q(\tau)$ for $\alpha = 1$.

$2a$	X-TFC	Ref.[20]
0.10	–6.85779(–1)	–6.85779(–1)
1.00	–2.32188(–1)	–2.32188(–1)
10.0	–4.22811(–2)	–4.22811(–2)

X-TFC method has been proved to be efficient and accurate in solving PDEs without being affected by the curse of dimensionality. Therefore, we are currently working on solving high-dimensional transport theory problems such as two-angle Radiative Transfer, time-dependent Radiative Transfer, 3D Radiative Transfer, and various problems in rarefied-gas dynamics and neutron transport.

FUNDING

L.B. Barichello acknowledges partial financial support from Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq).

AVAILABILITY OF DATA AND MATERIALS

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

COMPETING INTERESTS

The authors declare that they have no competing interests.

[1] P. L. Bhatnagar, E. P. Gross, and M. Krook, "A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems," Physical review **94**, 511 (1954).
[2] P. Welander, "On the temperature jump in a rarefied gas," Arkiv fysik **7** (1954).
[3] S. Harris, *An introduction to the theory of the Boltzmann equation* (Courier Corporation, 2004).
[4] C. Cercignani, "The Boltzmann equation," in *The Boltzmann equation and its applications* (Springer, 1988) pp. 40–103.
[5] C. Cercignani and A. Daneri, "Flow of a rarefied gas between two parallel plates," Journal of Applied Physics **34**, 3509–3513 (1963).
[6] C. Cercignani, "Plane Poiseuille flow according to the method of elementary solutions," Journal of Mathematical Analysis and Applications **12**, 254–262 (1965).
[7] V. Boffi, L. De Socio, G. Gaffuri, and C. Pescatore, "Rigorous constructive solution to monodimensional poiseuille and thermal creep flows," Meccanica **11**, 183–190 (1976).
[8] S. Loyalka, N. Petrellis, and T. Storvick, "Some numerical results for the BGK model: Thermal creep and viscous slip problems with arbitrary accomodation at the surface," The Physics of Fluids **18**, 1094–1099 (1975).
[9] S. Loyalka, N. Petrellis, and T. Storvick, "Some exact numerical results for the BGK model: Couette, Poiseuille and thermal creep flow between parallel plates," Zeitschrift für angewandte Mathematik und Physik ZAMP **30**, 514–521 (1979).
[10] S. Loyalka, "Thermal transpiration in a cylindrical tube," The Physics of Fluids **12**, 2301–2305 (1969).
[11] S. Loyalka, "Comments on Poiseuille flow and thermal creep of a rarefied gas between parallel plates," The Physics of Fluids **17**, 1053–1055 (1974).
[12] C. Siewert, R. Garcia, and P. Grandjean, "A concise and accurate solution for poiseuille flow in a plane channel," Journal of Mathematical Physics **21**, 2760–2763 (1980).
[13] L. Barichello and C. Siewert, "A discrete-ordinates solution for poiseuille flow in a plane channel," Zeitschrift für angewandte Mathematik und Physik ZAMP **50**, 972–981 (1999).
[14] L. Barichello, M. Camargo, P. Rodrigues, and C. Siewert, "Unified solutions to classical flow problems based on the BGK model," Zeitschrift für angewandte Mathematik und Physik ZAMP **52**, 517–534 (2001).
[15] Y. Sone and Y. Sone, *Kinetic theory and fluid dynamics* (Springer, 2002).
[16] F. Sharipov and V. Seleznev, "Data on internal rarefied gas flows," Journal of Physical and Chemical Reference Data **27**, 657–706 (1998).
[17] E. Gross, E. Jackson, and S. Ziering, "Boundary value problems in kinetic theory of gases," Annals of Physics **1**, 141–167 (1957).
[18] D. R. Willis, "Comparison of kinetic theory analyses of linearized couette flow," The Physics of Fluids **5**, 127–135 (1962).
[19] C. Siewert and D. Valougeorgis, "An analytical discrete-ordinates solution of the S-model kinetic equations for flow in a cylindrical tube," Journal of Quantitative Spectroscopy and Radiative Transfer **72**, 531–550 (2002).
[20] C. S. Scherer, J. F. Prolo Filho, and L. B. Barichello, "An analytical approach to the unified solution of kinetic equations in rarefied gas dynamics. I. Flow problems," Zeitschrift für angewandte Mathematik und Physik **60**, 70–115 (2009).
[21] C. Siewert, "Poiseuille, thermal creep and Couette flow: results based on the CES model of the linearized Boltzmann equation," European Journal of Mechanics-B/Fluids **21**, 579–597 (2002).
[22] L. Barichello and C. E. Siewert, "A discrete-ordinates solution for a non-grey model with complete frequency redistribution," Journal of Quantitative Spectroscopy and Radiative Transfer **62**, 665–675 (1999).
[23] G. C. Wick, "Über ebene diffusionsprobleme," Zeitschrift für Physik **121**, 702–718 (1943).
[24] S. Chandrasekhar, *Radiative transfer* (Courier Corporation, 2013).
[25] M. Raissi, P. Perdikaris, and G. E. 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).
[26] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient pathologies in physics-informed neural networks," arXiv preprint arXiv:2001.04536 (2020).
[27] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, " Extreme learning machine: Theory and applications ," Neurocomputing **70**, 489–501 (2006).
[28] D. Mortari, "The theory of connections: Connecting points," Mathematics **5**, 57 (2017).
[29] D. Mortari, "Least-squares solution of linear differential equations," Mathematics **5**, 48 (2017).
[30] D. Mortari, H. Johnston, and L. Smith, "High accuracy least-squares solutions of nonlinear differential equations," Journal of Computational and Applied Mathematics **352**, 293 – 307 (2019).
[31] M. De Florio, E. Schiassi, A. D'Ambrosio, D. Mortari, and R. Furfaro, "Theory of functional connections applied to linear odes subject to integral constraints and linear ordinary integro-differential equations," Mathematical and Computational Applications **26**, 65 (2021).
[32] D. Mortari and C. Leake, "The Multivariate Theory of Connections," MDPI Mathematics **7**, 296 (2019).
[33] C. Leake and D. Mortari, "An Explanation and Implementation of Multivariate Theory of Connections via Examples," in *2019 AAS/AIAA Astrodynamics Specialist Conference, Portland, MN, August 11–15, 2019* (AAS/AIAA, 2019).
[34] A. Gil, J. Segura, and N. Temme, *Numerical Methods for Special Functions* (Society for Industrial and Applied Mathematics, 2007) pp. 51–80.
[35] C. Lanczos, *Applied Analysis* (Dover Publications, Inc., New York, 1957) p. 453.
[36] H. Johnston, "The theory of functional connections: A journey from theory to application," arXiv preprint arXiv:2105.08034 (2021).
[37] C. Leake, "The multivariate theory of functional connections: An n-dimensional constraint embedding technique applied to partial differential equations," arXiv preprint arXiv:2105.07070 (2021).
[38] C. Leake and D. Mortari, "Deep Theory of Functional Connections: A New Method for Estimating the Solutions of Partial Differential Equations," Machine Learning and Knowledge Extraction **2**, 37–55 (2020).
[39] E. Schiassi, R. Furfaro, C. Leake, M. De Florio, H. Johnston, and D. Mortari, "Extreme Theory of Functional Connections: A Fast Physics-Informed Neural Network Method for Solving Ordinary and Partial Differential Equations," Neurocomputing (2021).
[40] R. Furfaro and D. Mortari, "Least-squares solution of a class of optimal space guidance problems via Theory of Connections," Acta Astronautica (2019), https://doi.org/10.1016/j.actaastro.2019.05.050.
[41] H. Johnston, E. Schiassi, R. Furfaro, and D. Mortari, "Fuel-efficient powered descent guidance on large planetary bodies via theory of functional connections," arXiv preprint arXiv:2001.03572 (2020).
[42] E. Schiassi, A. D'Ambrosio, H. Johnston, R. Furfaro, F. Curti, and D. Mortari, "Complete Energy Optimal Landing on Planetary Bodies via Theory of Functional Connections," Acta Astronautica - in preparation (2020).
[43] K. Drozd, R. Furfaro, E. Schiassi, H. Johnston, and D. Mortari, "Energy-optimal trajectory problems in relative motion solved via theory of functional connections," Acta Astronautica **182**, 361–382 (2021).
[44] A. D'Ambrosio, E. Schiassi, F. Curti, and R. Furfaro, "Pontryagin neural networks with functional interpolation for optimal intercept problems," Mathematics **9**, 996 (2021).
[45] E. Schiassi, A. D'Ambrosio, A. Scorsoglio, R. Furfaro, and F. Curti, "Class of optimal space guidance problems solved via indirect methods and physics-informed neural networks," (2021).
[46] A. D'Ambrosio, E. Schiassi, F. Curti, and R. Furfaro, "Physics-informed neural networks applied to a series of constrained space guidance problems," (2021).
[47] M. De Florio, E. Schiassi, R. Furfaro, B. D. Ganapol, and D. Mostacci, "Solutions of Chandrasekhar' s Basic Problem in Radiative Transfer via Theory of Functional Connections," Journal of Quantitative Spectroscopy and Radiative Transfer , 107384 (2020).
[48] M. De Florio, E. Schiassi, B. D. Ganapol, and R. Furfaro, "Physics-informed neural networks for rarefied-gas dynamics: Thermal creep flow in the Bhatnagar–Gross–Krook approximation," Physics of Fluids **33**, 047110 (2021).
[49] M. De Florio, E. Schiassi, B. D. Ganapol, and R. Furfaro, "Physics-informed neural networks for rarefied-gas dynamics: Poiseuille flow in the bgk approximation," Zeitschrift für angewandte Mathematik und Physik **73**, 1–18 (2022).
[50] E. Schiassi, M. De Florio, A. D'ambrosio, D. Mortari, and R. Furfaro, "Physics-informed neural networks and functional interpolation for data-driven parameters discovery of epidemiological compartmental models,"

Mathematics **9**, 2069 (2021).
[51]F. Jacob and B. Ted, *A first course in finite elements* (Wiley, 2007).
[52]S. Mishra and R. Molinaro, "Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs," IMA Journal of Numerical Analysis (2021).
[53]M. Liu, M. Hou, J. Wang, and Y. Cheng, "Solving two-dimensional linear partial differential equations based on Chebyshev neural network with extreme learning machine algorithm," Engineering Computations (2020).
[54]M. Williams, "A review of the rarefied gas dynamics theory associated with some classical problems in flow and heat transfer," Zeitschrift für angewandte Mathematik und Physik ZAMP **52**, 500–516 (2001).
[55]C. Cercignani *et al.*, *Mathematical methods in kinetic theory* (Springer, 1969).
[56]M. M. Williams, "Mathematical methods in particle transport theory." (1971).
[57]S. K. Prabha and S. P. Sathian, "Molecular-dynamics study of poiseuille flow in a nanochannel and calculation of energy and momentum accommodation coefficients," Physical Review E **85**, 041201 (2012).
[58]M. Abramowitz and I. Stegun, "Handbook of Mathematical Functions," (Dover, 1970) p. 3.3.7, 9th ed.
[59]T. Barker, D. G. Schaeffer, P. Bohórquez, and J. M. N. T. Gray, "Well-posed and ill-posed behaviour of the $\mu(I)$-rheology for granular flows," J. Fluid. Mech. **779**, 794–818 (2015).
[60]G. Batchelor, "Small-scale variation of convected quantities like temperature in turbulent fluid Part 1. General discussion and the case of small conductivity." J. Fluid Mech. **5**, 113–133 (1971).
[61]A. V. Briukhanov, S. S. Grigorian, S. M. Miagkov, M. Y. Plam, I. E. Shurova, M. E. Eglit, and Y. L. Yakimov, "On some new approaches to the dynamics of snow avalanches," in *Physics of Snow and Ice, Proceedings of the International Conference on Low Temperature Science*, Vol. 1 (Institute of Low Temperature Science, Hokkaido University, Sapporo, Hokkaido, Japan, 1967) pp. 1221–1241.
[62]C. Brownell and L. Su, "Planar measurements of differential diffusion in turbulent jets," AIAA Paper 2004-2335 (2004).
[63]C. Brownell and L. Su, "Scale relations and spatial spectra in a differentially diffusing jet," AIAA Paper 2007-1314 (2007).
[64]S. Dennis, "Compact explicit finite difference approximations to the Navier–Stokes equation," in *Ninth Intl Conf. on Numerical Methods in Fluid Dynamics*, Lecture Notes in Physics, Vol. 218, edited by Soubbaramayer and J. Boujot (Springer, 1985) pp. 23–51.
[65]A. N. Edwards, S. Viroulet, B. P. Kokelaar, and J. M. N. T. Gray, "Formation of levees, troughs and elevated channels by avalanches on erodible slopes," J. Fluid Mech. **823**, 278–315 (2017).
[66]L.-S. Hwang and E. Tuck, "On the oscillations of harbours of arbitrary shape," J. Fluid Mech. **42**, 447–464 (1970).
[67]D. D. Joseph and J. C. Saut, "Short-wave instabilities and ill-posed initial-value problems," Theoretical and Computational Fluid Dynamics **1**, 191–227 (1990), 10.1007/BF00418002.
[68]M. Worster, "The dynamics of mushy layers," in *Interactive dynamics of convection and solidification*, edited by S. Davis, H. Huppert, W. Muller, and M. Worster (Kluwer, 1992) pp. 113–138.
[69]W. Koch, "Resonant acoustic frequencies of flat plate cascades," J. Sound Vib. **88**, 233–242 (1983).
[70]J.-J. Lee, "Wave-induced oscillations in harbours of arbitrary geometry," J. Fluid Mech. **45**, 375–394 (1971).
[71]C. Linton and D. Evans, "The radiation and scattering of surface waves by a vertical circular cylinder in a channel," Phil. Trans. R. Soc. Lond. **338**, 325–357 (1992).
[72]P. Martin, "On the null-field equations for the exterior problems of acoustics," Q. J. Mech. Appl. Maths **33**, 385–396 (1980).
[73]R. Rogallo, "Numerical experiments in homogeneous turbulence," Tech. Rep. 81835 (NASA Tech. Mem., 1981).
[74]F. Ursell, "Surface waves on deep water in the presence of a submerged cylinder I," Proc. Camb. Phil. Soc. **46**, 141–152 (1950).
[75]L. van Wijngaarden, "On the oscillations near and at resonance in open pipes," J. Engng Maths **2**, 225–240 (1968).
[76]P. Miller, *Mixing in high Schmidt number turbulent jets*, Ph.D. thesis, California Institute of Technology (1991).
[77]J.-Y. Bouguet, "Camera calibration toolbox for matlab (2008)," URL http://www. vision. caltech. edu/bouguetj/calib_doc **1080** (2008).
[78]C. Cercignani, *Theory and application of the Boltzmann equation* (Scottish Academic Press, 1975).
[79]B. D. Ganapol, "The response matrix discrete ordinates solution to the 1d radiative transfer equation," Journal of Quantitative Spectroscopy and Radiative Transfer **154**, 72–90 (2015).
[80]C. Siewert, "A concise and accurate solution to chandrasekhar's basic problem in radiative transfer," Journal of Quantitative Spectroscopy and Radiative Transfer **64**, 109–130 (2000).
[81]R. Furfaro and D. Mortari, "Least-squares solution of a class of optimal space guidance problems via theory of connections," Acta Astronautica **168**, 92–103 (2020).
[82]B. D. Ganapol, "1D thermal creep channel flow in the BGK approximation by adding and doubling," Annals of Nuclear Energy **134**, 441–451 (2019).
[83]T. Kanki and S. Iuchi, "Poiseuille flow and thermal creep of a rarefied gas between parallel plates," The Physics of Fluids **16**, 594–599 (1973).
[84]I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," IEEE Transactions on Neural Networks **9**, 987–1000 (1998).
[85]R. Fletcher, *Practical Methods of Optimization*, 2nd ed. (Wiley, New York, 1987).
[86]S. A. Coons, "SURFACES FOR COMPUTER-AIDED DESIGN OF SPACE FORMS," Tech. Rep. (Massachusetts Institute of Technology, Cambridge, MA, USA, 1967).
[87]J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," (2018), arXiv:1708.07469v5.
[88]J. Argyris and S. Kelsey, "Energy Theorems and Structural Analysis: A Generalized Discourse with Applications on Energy Principles of Structural Analysis Including the Effects of Temperature and Non-Linear Stress-Strain Relations," Aircraft Engineering and Aerospace Technology **26**, 347–356 (1954), https://doi.org/10.1108/eb032482.
[89]M. J. Turner, R. W. Clough, H. C. Martin, and L. J. Topp, "Stiffness and Deflection Analysis of Complex Structures," Journal of the Aeronautical Sciences **23**, 805–823 (1956).
[90]R. W. Clough, *The finite element method in plane stress analysis* (American Society of Civil Engineers, 1960) pp. 345–378.
[91]J. N. Reddy, "An Introduction to the Finite Element Method," Journal of Pressure Vessel Technology **111**, 348–349 (1989), https://asmedigitalcollection.asme.org/pressurevesseltech/article-pdf/111/3/348/5110690/348_1.pdf.
[92]H. Johnston and D. Mortari, "Linear Differential Equations Subject to Multivalued, Relative and/or Integral Constraints with Comparisons to Chebfun," SIAM Journal of Numerical Analysis (2018), submitted.
[93]A. G. Baydin, B. A. Pearlmutter, and A. A. Radul, "Automatic differentiation in machine learning: a survey," CoRR **abs/1502.05767** (2015), arXiv:1502.05767.
[94]TensorFlow™, "Automatic differentiation and gradient tape," `https://www.tensorflow.org/tutorials/eager/automatic_differentiation` (2018), accessed: 11-04-2018.
[95]X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, Vol. 9, edited by Y. W. Teh and M. Titterington (PMLR, Chia Laguna Resort, Sardinia, Italy, 2010) pp. 249–256.
[96]D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," CoRR **abs/1412.6980** (2014), arXiv:1412.6980.
[97]J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," J. Mach. Learn. Res. **12**, 2121–2159 (2011).
[98]T. Tieleman and G. Hinton, "Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning. Technical report, 2012." (2012).
[99]C. Leake, H. Johnston, L. Smith, and D. Mortari, "Analytically Embedding Differential Equation Constraints into Least Squares Support Vector Machines Using the Theory of Functional Connections," Machine Learning and Knowledge Extraction **1**, 1058–1083 (2019).
[100]S. Mall and S. Chakraverty, "Single Layer Chebyshev Neural Network Model for Solving Elliptic Partial Differential Equations," Neural Processing Letters **45**, 825–840 (2017).
[101]H. Sun, M. Hou, Y. Yang, T. Zhang, F. Weng, and F. Han, "Solving Partial Differential Equation Based on Bernstein Neural Network and Extreme Learning Machine Algorithm," Neural Processing Letters **50**, 1153–1172

Appendix F

De Florio et al. (2022) Stiff Chemical Kinetics

Chaos

Physics-informed neural networks and functional interpolation for stiff chemical kinetics

Cite as: Chaos **32**, 063107 (2022); https://doi.org/10.1063/5.0086649
Submitted: 27 January 2022 • Accepted: 06 May 2022 • Published Online: 01 June 2022

Mario De Florio, Enrico Schiassi and Roberto Furfaro

Physics-informed neural networks and functional interpolation for stiff chemical kinetics

Cite as: Chaos **32**, 0631 07 (2022); doi: 1 0.1 063/5.0086649
Submitted: 27 January 2022 · Accepted: 6 May 2022 ·
Published Online: 1 June 2022

Mario De Florio,[1] Enrico Schiassi,[1] and Roberto Furfaro[1,2,a)]

AFFILIATIONS

[1] Department of Systems & Industrial Engineering, The University of Arizona, 1127 James E. Rogers Way, Tucson, Arizona 85719, USA
[2] Department of Aerospace & Mechanical Engineering, The University of Arizona, 1130 N Mountain Ave, Tucson, Arizona 85721, USA

[a)] **Author to whom correspondence should be addressed:** robertof@email.arizona.edu

ABSTRACT

This work presents a recently developed approach based on physics-informed neural networks (PINNs) for the solution of initial value problems (IVPs), focusing on stiff chemical kinetic problems with governing equations of stiff ordinary differential equations (ODEs). The framework developed by the authors combines PINNs with the theory of functional connections and extreme learning machines in the so-called extreme theory of functional connections (X-TFC). While regular PINN methodologies appear to fail in solving stiff systems of ODEs easily, we show how our method, with a single-layer neural network (NN) is efficient and robust to solve such challenging problems without using artifacts to reduce the stiffness of problems. The accuracy of X-TFC is tested against several state-of-the-art methods, showing its performance both in terms of computational time and accuracy. A rigorous upper bound on the generalization error of X-TFC frameworks in learning the solutions of IVPs for ODEs is provided here for the first time. A significant advantage of this framework is its flexibility to adapt to various problems with minimal changes in coding. Also, once the NN is trained, it gives us an analytical representation of the solution at any desired instant in time outside the initial discretization. Learning stiff ODEs opens up possibilities of using X-TFC in applications with large time ranges, such as chemical dynamics in energy conversion, nuclear dynamics systems, life sciences, and environmental engineering.

The recently developed physics-informed neural networks (PINNs) is attracting much attention among the numerical methods for solving differential equations. Despite its many advantages, its failure in solving stiff systems of ordinary differential equations has been demonstrated. This study shows that merging PINN with the theory of functional connections allows one to obtain highly accurate solutions for stiff non-linear chemical kinetics problems. The initial conditions are always analytically satisfied via particular expressions called constrained expressions, and the boundary-free solution is expanded with a single-layer forward neural network, trained via the extreme learning machine algorithm. The proposed method is tested by solving classic, challenging chemical kinetics problems from the literature, and it can be applied to many chemical, biological, and nuclear systems subject to stiffness.

I. INTRODUCTION

Artificial neural networks (NNs) have been widely used in the recent past to solve linear and non-linear ordinary differential equations (ODEs) starting from the 1990s.[1–3] In 1998, Lagaris *et al.*[4] extended this idea to a new numerical method based on feedforward neural networks (FNNs) for the solution of non-linear ODEs and partial differential equations (PDEs). The method aims to train the FNNs to minimize the error between the FNN prediction and the right-hand side of the differential equations. Subsequently, Filici[5] proposed a single-layer multiple linear output perceptron, in which the training of the NN was performed with a Levenberg–Marquardt algorithm (LMDER optimization solver) from MINPACK,[6] which was tested by solving simple non-stiff ODE problems. Gerstberger and Rentrop[7] developed one of the first frameworks to solving stiff ODEs and differential-algebraic equations involving NNs.

Recently, a new class of methods to solve differential equations has been developed in the machine learning community, and it is called physics-informed neural networks (PINNs). The methods underlying this category can be generalized as algorithms that include the physics of the problem into a data-driven representation of functional relationships underlying collections of input–output pairs.[8–13]

Large-scale stiff systems of ODEs are mathematical models governing a pretty wide range of real-world problems such as chemistry, biology, epidemiology, engineering, neuroscience, financial systems, and so on. For this reason, recently, there has been a resurgence in the interest in developing new numerical methods capable of solving large time-scale stiff problems.[14–17] Two recent works aim to solve stiff chemical kinetics problems using PINN and neural ODEs.[18,19] In particular, Ref. 18 highlights the failure of regular PINNs in solving stiff problems and proposes a quasi-steady-state assumption (QSSA) to reduce the stiffness of the systems of ODEs. This artifact converts the problems into non-/mild-stiff problems, where the PINN framework can be applied. The proposed algorithm has been named Stiff-PINN. In Ref. 19, the authors employ neural ordinary differential equations to solve stiff systems of ODEs. In both papers, two classical stiff chemical kinetics problems are used for testing the performance of their methods: the ROBER problem[20] and the POLLU problem.[21] The results presented in both papers are not numerically compared with any other method and any benchmark in the literature, and no computational times required for the training are given. Although these two methods clean up the regular PINNs failure, they are far for being considered accurate and fast methods for solving stiff problems since the results reported in the manuscripts are limited to reproduction of the species profiles only.

Wang *et al.*[13] have recently demonstrated that PINNs are not accurate (or even not capable) to solve problems when the dynamics are particularly complex (e.g., stiff). This is due to unbalanced back-propagated gradients during the model training.

We propose a numerical scheme based on PINN and Theory of Functional Connections (TFC), named Extreme Theory of Functional Connections (X-TFC),[22] which has already proved its successful performance in many applications.[23–28] We show how X-TFC, by using extreme learning machine (ELM) and the TFC constrained expressions (CEs), overcomes the difficulties of PINNs in solving stiff systems of ODEs, and it is comparable with the state-of-the-art numerical methods in terms of accuracy and computational times. In particular, we will focus on the solution of four classical stiff chemical kinetics problems: the ROBER problem,[20] the Chemical Akzo Nobel problem,[29] the Belousov–Zhabotinsky reaction,[30] and the POLLU problem.[21]

The manuscript is organized as follows. First, in Sec. II, the general PINN methodology is presented, explaining how we efficiently merge it with TFC and extreme learning machine (ELM) to obtain highly accurate solutions. Also, a rigorous upper bound on the generalization error of X-TFC frameworks in learning the solutions of initial value problems (IVPs) for ODEs is provided. In Sec. III, the general form of the chemical kinetics equations and the X-TFC algorithm with time decomposition are presented. In Sec. IV, the accuracy of the proposed PINN-based method is tested against several methods and benchmarks from the literature for four classical stiff chemical kinetics problems. Conclusions are drawn in Sec. V.

II. X-TFC TO SOLVING NON-LINEAR INITIAL VALUE PROBLEMS

PINNs is a recently developed algorithm by Raissi *et al.*[8] for solving forward and inverse problems governed by ODEs and PDEs. As opposed to traditional methods for solving DEs such as finite element method[31] and finite difference method, the PINN framework constructs a loss function (cost function) informed by the physics of the problem. Then, the optimal set of weights and biases parameters in the NN are sought to minimize the error. The NN parameters weights and biases are learned by minimizing the mean squared error (MSE) loss with gradient-based methods, such as stochastic gradient descent (e.g., Adam optimizer).

One of the significant drawbacks of the standard PINN frameworks is that the initial and boundary conditions (i.e., the equation constraints) are not analytically satisfied. Hence, during the PINN training, there are competing objectives: learning the DE solution within the domain and satisfying the equation constraints. This leads to unbalanced gradients during the network training via gradient-based methods that causes PINNs to have often difficulties in properly approximating the solution of DEs.[32] Indeed, it is well known that gradient-based methods may get stuck in limit cycles or even diverge if multiple competing objectives are present.[33,34] In Ref. 32, to overcome this issue, the authors developed a learning rate annealing algorithm that employs gradient statistics to assign appropriate weights to different terms adaptively (e.g., DE residuals within the domain and DE residuals on the boundaries) in the PINNs loss function during the network training. The approach used here is different and is based on the TFC[35] which allows to analytically satisfy the equation constraints, hence removing the competing objective in the NN training.

In this work, we aim to learn the solutions of initial value problems (IVPs) governed by the chemical kinetics equations using a recently developed PINN framework named X-TFC. The solutions are approximated with the so-called constrained expressions (CEs), defined in the TFC.[35] The structure of the X-TFC algorithm for solving first-order ODEs is presented in Sec. II A and the estimate of the generalization error is given in Sec. II B.

A. X-TFC to solving first-order ODEs

A general non-linear first-order IVP can be written as follows:

$$\begin{cases} \dfrac{dy}{dt} = f(y,t), & \forall t \in [0,T], \\ y(0) = y_0. \end{cases} \tag{1}$$

According to the TFC method,[35] the unknown function can be expanded via the CE, as follows:

$$y(t) \simeq y_{\mathrm{CE}}(t) = g(t) + \eta p(t). \tag{2}$$

The first term $g(t)$ is a freely chosen function, which must exist on the constraint. In the second term, we have $p(t)$, that is an user-defined function which must be independent from $g(t)$ and an unknown constant parameter η. With this form of the CE, the constraint of the problem is always analytically satisfied, regardless of the choice of $g(t)$. For IVPs, the best selection of independent function is $p(t) = 1$. The reasons for this choice are explained in detail in Refs. 36 and 37. Thus, the CE becomes

$$y_{\mathrm{CE}}(t) = g(t) + \eta. \tag{3}$$

By embedding the constraint into the CE, we have

$$y_{\mathrm{CE}}(0) = y_0 = g_0 + \eta \Longrightarrow \eta = y_0 - g_0, \tag{4}$$

which substituted into Eq. (3) gives us

$$y_{\mathrm{CE}}(t) = g(t) + (y_0 - g_0) \tag{5}$$

and its derivative

$$\frac{dy_{\mathrm{CE}}}{dt} = \frac{dg}{dt}. \tag{6}$$

By plugging Eqs. (5) and (6) into the general problem (1), we obtain an unconstrained ODE

$$\frac{dy_{\mathrm{CE}}}{dt} = f(y_{\mathrm{CE}}, t), \quad \forall t \in [0, T]. \tag{7}$$

Thus, the new ODE depends only on t and $g(t)$,

$$\frac{dg}{dt} = f(g(t) + y_0 - g_0, t), \quad \forall t \in [0, T]. \tag{8}$$

X-TFC employs a shallow NN as the free function $g(t)$, which is represented as

$$g(t) = \sum_{j=1}^{L} \beta_j \sigma\left(w_j t + b_j\right) = \begin{bmatrix} \sigma^1 \\ \vdots \\ \sigma^L \end{bmatrix}^T, \boldsymbol{\beta} = \boldsymbol{\sigma}^T \boldsymbol{\beta}, \tag{9}$$

where $j = 1, \ldots, L$, and L is the number of neurons, $w_j, \beta_j, b_j \in \mathbb{R}$ are the input weight, output weight, and bias, respectively, corresponding to the jth neuron. The activation function $\sigma^j(\cdot)$ is selected by the user.

The NN parameters in Eq. (9) are trained via the ELM algorithm, whose convergence proofs can be found in Ref. 38. The known parameters are the input weights and biases, randomly selected. Thus, the only unknown parameters learned by X-TFC to solve the IVP are the output weights $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_L]^T$. Figure (1) shows a graphical representation of a general single-layer NN used in this work.

The domain of some activation functions ($x \in [x_0, x_f]$) and the domain of the problem under study ($t \in [t_0, t_f]$) may not be consistent. Hence, a mapping of the independent variable $t \in [t_0, t_f]$ in the x domain is required. In general, to achieve better training accuracy, $x \in [0, 1]$ or $x \in [-1, 1]$ domains are employed, depending on the codomain of the chosen activation function. The mapping linear transformation used is

$$x = x_0 + c(t - t_0) \longleftrightarrow t = t_0 + \frac{1}{c}(x - x_0),$$

where the mapping coefficient c is

$$c = \frac{x_f - x_0}{t_f - t_0}.$$

The nth derivative of Eq. (9) is given by

$$\frac{d^n g(t)}{dt^n} = c^n \boldsymbol{\beta}^T \frac{d^n \boldsymbol{\sigma}(x)}{dx^n}, \tag{10}$$

according to the derivative chain rule.

A discretization in N quadrature (training) points of the x domain is performed, and then the following loss function is minimized via *unconstrained* optimization techniques:

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{N} w_i \left| c \frac{d\boldsymbol{\sigma_i}^T \boldsymbol{\beta}}{dx} - f\left(\boldsymbol{\sigma_i}^T \boldsymbol{\beta} + y_0 - \boldsymbol{\sigma_0}^T \boldsymbol{\beta}, x_i\right) \right|^2. \tag{11}$$

Here, w_i is the ith quadrature weight corresponding to the i_{th} quadrature point, $\boldsymbol{\sigma_i}$ and $\boldsymbol{\sigma_0}$ are the activation functions computed in the ith training point and the first training point (initial time instant), respectively. To note that, throughout this paper, even though σ_0 actually coincides with σ_1, the authors prefer to maintain the notation σ_0, because it refers to the traditional subscript of the initial time instant (e.g., t_0).

When the DEs under study are non-linear, an iterative least-squares approach is employed[37] to minimize Eq. (11). By following the iterative procedure, at each k_{th} iteration corresponds an update of the output weights. That is,

$$\boldsymbol{\beta}^{k+1} = \boldsymbol{\beta}^k + \Delta\boldsymbol{\beta}^k, \tag{12}$$

where $\Delta\boldsymbol{\beta}^k$ is evaluated as

$$\Delta\boldsymbol{\beta}^k = -\left(\mathbb{J}^{\mathrm{T}}(\boldsymbol{\beta}^k)\mathbb{J}(\boldsymbol{\beta}^k)\right)^{-1} \mathbb{J}^{\mathrm{T}}(\boldsymbol{\beta}^k)\mathcal{L}(\boldsymbol{\beta}^k). \tag{13}$$

$\mathbb{J}$ is the Jacobian matrix of the loss functions. By imposing a tolerance ϵ (user-defined), the iterative procedure continues until the following conditions is reached:

$$L_2[\mathcal{L}(\boldsymbol{\beta}^k)] < \epsilon. \tag{14}$$

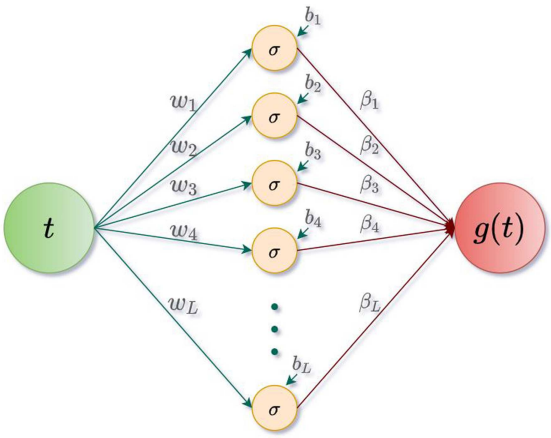


FIG. 1. Example of a shallow NN, trained via ELM, used as the free-function in the X-TFC method.

B. Estimate on the generalization error

In this subsection, by following the work by Mishra and Molinaro,[39] we derive an estimate of the generalization error when using X-TFC for learning the solution of a general non-linear IVP.

We consider a single non-linear ODE, but the results can be extended to systems of non-linear ODEs, without loss of generality,

$$\begin{cases} \dfrac{dy}{dt} = f(y, t), & \forall t \in [0, T], \\ y(0) = y_0. \end{cases} \tag{15}$$

Here, $y_0 \in \mathbb{R}$ are the initial data, $y \in [0, T] \subset \mathbb{R}$ is the solution of (15), and $f : \mathbb{R}$ is the non-linear forcing term. We assume that f is

globally Lipschitz i.e., there exists a constant $C_f \geq 0$ (independent of t) such that

$$|f(t_v) - f(t_w)| \leq C_f|t_v - t_w|, \quad t_v, t_w \in [0, T]. \tag{16}$$

In order to apply the X-TFC algorithm to learn the solution of the IVP, we need to specify the training set $\mathcal{S}$ and define the appropriate residual.

1. Training set

Consider the time domain $[0, T] \in \mathbb{R}$. We will choose the training set $\mathcal{S} \subset [0, T]$, based on suitable quadrature (training) points. Since the IVP constraints are analytically satisfied thanks to the CE, our training points are

$$\mathcal{S} = \{t_i\} \text{ for } 1 \leq i \leq N, \tag{17}$$

with each $t_i \in [0, T]$. According to Ref. 39, these points can be the quadrature points, corresponding to a suitable time grid-based composite Gauss quadrature rule or randomly selected points.

2. Residuals

In order to apply the X-TFC algorithm, we need to define appropriate residuals. We approximate the IVP solution with the following constrained expression:

$$y_{\boldsymbol{\beta}}(t) = g_{\boldsymbol{\beta}}(t) + y_0 - g_{\boldsymbol{\beta}}(0), \tag{18}$$

where $g_{\boldsymbol{\beta}}(t)$ is a single-layer NN with random features. Then, we define the following residuals:

$$\mathcal{R}_{\boldsymbol{\beta}}(t) = \frac{d}{dt}y_{\boldsymbol{\beta}}(t) - f(y_{\boldsymbol{\beta}}(t), t) = \frac{d}{dt}g_{\boldsymbol{\beta}}(t) - f\big(g_{\boldsymbol{\beta}}(t) + y_0 - g_{\boldsymbol{\beta}}(0), t\big). \tag{19}$$

3. Loss function

To learn the unknown solution, we need to minimize the following loss function:

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{N} w_i|\mathcal{R}_{\boldsymbol{\beta}}(t_i)|^2. \tag{20}$$

The X-TFC algorithm is now completely specified and it can be run for learning the solution of the non-linear IVP of Eq. (15), which we denote as $y^* = y_{\boldsymbol{\beta}^*}$, where $\boldsymbol{\beta}^*$ are the computed optimal output weights.

We are interested in estimating the generalization error for X-TFC, which is defined as

$$\varepsilon_G = \left(\int_0^T |y(t) - y^*(t)|^2 dt\right)^{1/2}. \tag{21}$$

ε_G, according to Ref. 39 and references within, is the error emerging from the approximation of the true solution, $y(t)$, via X-TFC, within in the domain $0 \leq t \leq T$. We estimate the generalization error in terms of the *training error* defined as

$$\varepsilon_T^2 = \sum_{i=1}^{N} w_i|\mathcal{R}_{\boldsymbol{\beta}^*}(t_i)|^2. \tag{22}$$

Note that the training error can be readily computed *a posteriori* from the loss function (20).

According to Ref. 39, we also need the following assumptions on the quadrature error: for any function $u \in C^l([0, T])$, the quadrature rule corresponding to quadrature weights w_i at points $t_i \in \mathcal{S}$, with $1 \leq i \leq N$, satisfies

$$\left|\int_0^T u(t)dt - \sum_{i=1}^{N} w_i u(t_i)\right| \leq C_{\text{quad}}(||u||_{C^l})N^{-\alpha}. \tag{23}$$

Here, $\alpha > 0$ and different order quadrature rules can be used. The estimate of the generalization error for X-TFC is given in the following theorem.

Theorem 1. *Let $y \in C^k([0, T])$ be the unique classical solution of the non-linear problem (15) with the dynamics f satisfying Eq. (16). Let $y^* = y_{\boldsymbol{\beta}^*}$ be the solution approximated via X-TFC, corresponding to loss function (20). Then, the generalization error (21) can be estimated as*

$$\varepsilon_G \leq C_1\left(\varepsilon_T^2 + C_{\text{quad}}N^{-\alpha}\right)^{\frac{1}{2}}, \tag{24}$$

where the constant C_1 is given by

$$C_1 = \left(\frac{1}{1 + 2C_f}\left(e^{(1+2C_f)T} - 1\right)\right)^{\frac{1}{2}}$$

and

$$C_{\text{quad}} = C_{\text{quad}}(||\mathcal{R}^2||_{C^{k-1}}),$$

which is a positive constant depending on the number of training points, the quadrature scheme used, and the residuals evaluated on the training points.[39]

Proof. By the definitions of the residuals (19) and the underlying ODE (15), we can verify that the error $\hat{y} = y^* - y$ satisfies the following equation:

$$\begin{aligned} \frac{d\hat{y}}{dt} &= f(y^*, t) - f(y, t) + \mathcal{R}, \qquad \forall t \in [0, T], \\ \hat{y}(0) &= 0. \end{aligned} \tag{25}$$

Here, we have denoted $\mathcal{R} = \mathcal{R}_{\boldsymbol{\beta}^*}$ for convenience of notation. Multiplying both sides of the ODE (25) by $\hat{y}$ yields

$$\hat{y}\frac{d\hat{y}}{dt} = \hat{y}\big[f(y^*, t) - f(y, t)\big] + \hat{y}\mathcal{R}, \tag{26}$$

that is,

$$\frac{1}{2}\frac{d}{dt}|\hat{y}|^2 = \hat{y}[f(y^*, t) - f(y, t)] + \hat{y}\mathcal{R}. \tag{27}$$

By leveraging on the following inequality $\big(\hat{y} - \mathcal{R}\big)^2 \geq 0$, Eq. (27) is bounded by

$$\frac{1}{2}\frac{d}{dt}|\hat{y}|^2 \leq |\hat{y}|\,\big|f(y^*, t) - f(y, t)\big| + \frac{1}{2}|\mathcal{R}|^2 + \frac{1}{2}|\hat{y}|^2. \tag{28}$$

By (16), we have the following:

$$\frac{1}{2}\frac{d}{dt}|\hat{y}|^2 \leq |\hat{y}|C_f\underbrace{|y^* - y|}_{|\hat{y}|} + \frac{1}{2}|\mathcal{R}|^2 + \frac{1}{2}|\hat{y}|^2. \tag{29}$$

Rearranging, we get

$$\frac{d}{dt}|\hat{y}|^2 \leq \left(1 + 2C_f\right)|\hat{y}|^2 + |\mathcal{R}|^2. \tag{30}$$

Integrating the above inequality over $[0, \bar{T}]$ for any $\bar{T} \leq T$, we obtain

$$|\hat{y}(\bar{T})|^2 \leq \left(1 + 2C_f\right)\int_0^{\bar{T}} |\hat{y}|^2 dt + \int_0^T |\mathcal{R}|^2 dt. \tag{31}$$

Applying the integral form of Grönwall's inequality to the above, we get

$$|\hat{y}(\bar{T})|^2 \leq \left(e^{(1+2C_f)\bar{T}}\right)\int_0^T |\mathcal{R}|^2 dt. \tag{32}$$

Integrating over $[0, T]$,

$$\varepsilon_G^2 = \int_0^T |\hat{y}(\bar{T})|^2 d\bar{T} \leq \left(\frac{1}{1+2C_f}\left(e^{(1+2C_f)T} - 1\right)\right)\int_0^T |\mathcal{R}|^2 dt. \tag{33}$$

That is,

$$\varepsilon_G^2 \leq \left(\frac{1}{1+2C_f}\left(e^{(1+2C_f)T} - 1\right)\right)\left(\sum_{i=1}^{N} w_i|\mathcal{R}(t_i)|^2 + C_{\text{quad}}(||\mathcal{R}^2||_{C^{k-1}})N^{-\alpha}\right). \tag{34}$$

Thus,

$$\varepsilon_G^2 \leq \left(\frac{1}{1+2C_f}\left(e^{(1+2C_f)T} - 1\right)\right)\left(\varepsilon_T^2 + C_{\text{quad}}(||\mathcal{R}^2||_{C^{k-1}})N^{-\alpha}\right). \tag{35}$$

Finally,

$$\varepsilon_G \leq \left(\frac{1}{1+2C_f}\left(e^{(1+2C_f)T} - 1\right)\right)^{\frac{1}{2}}\left(\varepsilon_T^2 + C_{\text{quad}}N^{-\alpha}\right)^{\frac{1}{2}}, \tag{36}$$

where

$$\left(\frac{1}{1+2C_f}\left(e^{(1+2C_f)T} - 1\right)\right)^{\frac{1}{2}} = C_1. \tag{37}$$

This completes the proof. □

The estimate (24) bounds the generalization error in terms of training and quadrature errors. The training error is computed once the training is completed. As long as X-TFC is well trained, i.e., the training error is small, the bound (24) implies that the generalization error will be small for a large enough number of training points. The estimate is not sharp as triangle inequalities, and Grönwall's inequality is used. If random training points are used, the estimate needs to be slightly modified for the generalization error (24). Since random training points are not used in this work, we suggest the interested reader to see these details in Ref. 39.

III. CHEMICAL KINETICS EQUATIONS AND ADAPTIVE SCHEME

The stiff chemical kinetics systems of equations[40] we aim to solve are IVPs of ODEs, having the following form:

$$\begin{cases} \dfrac{dy_i(t)}{dt} = f_i(t, y_1, y_2, \ldots, y_m), \\ y_i(0) = y_{i_0}, \end{cases} \tag{38}$$

with $i = 1, 2, \ldots, m$, where the unknowns are the functions $y_i(t)$, representing the time-dependent chemical species concentrations. The non-linear functions f_i and the initial conditions y_{i_0} are known. The computational challenge for numerical methods with this kind of problem lies in the complex behavior and existence of sharp gradients in the unknown solution. This particular behavior is the so-called stiffness of a problem whose complete and precise definition has not yet been formulated. A problem can be considered stiff when a solution varies very slowly, but other nearby solutions can present steep changes. According to Lambert,[41] if a numerical method used to solve IVPs of ODEs is forced to decompose the time domain into multiple excessively small subintervals in relation to the smoothness of the exact solution in that domain, then that system of ODEs is stiff in that domain. Or even, according to Ernst Hairer,[42] a problem is considered being stiff if explicit methods are not capable of solving it.

For the convenience of the reader, the schematic of Fig. 2 summarizes the all X-TFC process used in this work.

Since we are facing long-time domain problems, a decomposition of the temporal domain into subintervals was performed to minimize the carryover of the error during the chemical reactions between the species. The temporal domain (whose decomposition is pictured in Fig. 3) is divided into n time steps of equal size $h = t_k - t_{k-1}$ for $k = 1, \ldots, n$, each of which is discretized into n_x points.

Thus, this decomposition into time steps leads to the solution of several consecutive IVPs

$$\begin{cases} \dfrac{dy_i^{(k)}(t)}{dt} = f_i(t, y_1^{(k)}, y_2^{(k)}, \ldots, y_m^{(k)}), \\ y_i^{(k)}(0) = y_{i_0}^{(k)}, \end{cases} \tag{39}$$

with $i = 1, 2, \ldots, m$. Initially, we solve the system of ODEs in the time step 1 by using the initial conditions $y_i(t) = y_{i_0}$ given by the problem, thus finding the unknown solutions at time t_1, which become the new initial conditions for the next step, and so on,

$$y_{i_0}^{(k)} = y_{i_f}^{(k-1)}(t_k). \tag{40}$$

Thus, according to Eq. (5), the CE for the kth time step is

$$y_i^{(k)}(t_k) = g(t_k) + \left(y_{i_f}^{(k-1)}(t_k) - g_0^{(k)}\right). \tag{41}$$

IV. NUMERICAL RESULTS AND DISCUSSIONS

In this section, four chemical kinetics benchmark problems are analyzed, and numerically solved, and the solutions are compared with state-of-art methods' results. All the problems tackled in this manuscript have been coded in Matlab R2021a and run with an Intel Core i7-9700 CPU PC with 64 GB of RAM.

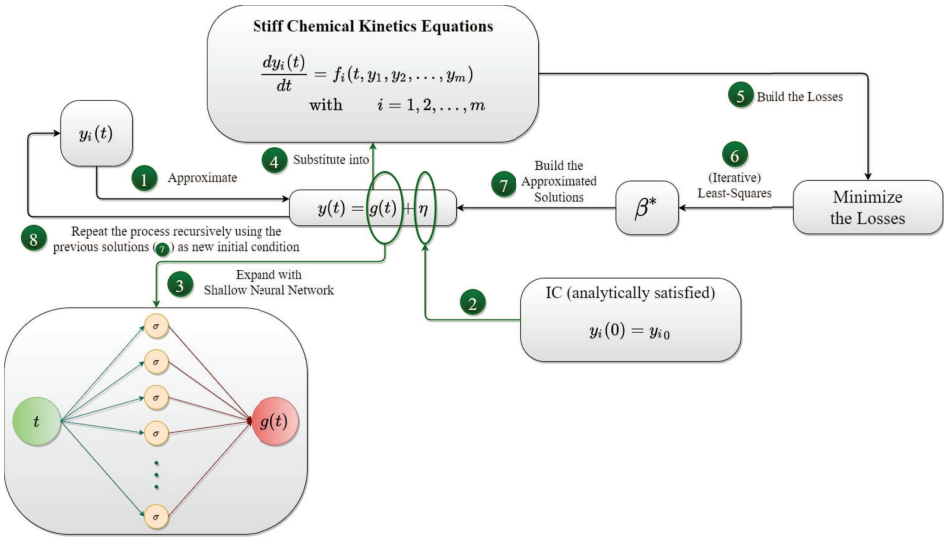


FIG. 2. Schematic of the X-TFC algorithm to solve the chemical kinetics equations.

Four benchmark problems in chemical kinetics are used for our simulations: Robertson's chemical reaction model, as known as the ROBER problem,[20] the Chemical Akzo Nobel problem,[43] the Belousov–Zhabotinsky reaction,[44] an air pollution problem, as known as POLLU problem.[21]

In the X-TFC framework, several hyperparameters can be adjusted to obtain accurate solutions, such as the number of training points, n_x, the number of neurons, L, the type of activation function, and the probability distribution where input weights and bias are sampled. For all the problems, we have used a hyperbolic tangent (*tanh*) as a activation function, and the weights and biases are randomly sampled from $\mathcal{U}[-1, 1]$. An analysis to study the X-TFC's sensitivity to these hyperparameters has been performed in our previous work,[22] which showed that the accuracy of the solution is not as sensitive to the type of activation function and/or to the probability distribution used to sample the weights and bias of the inputs. However, it is to the number of neurons and training points.

A. ROBER problem

The first test case is given by Robertson's system of equations,[20] denoted as ROBER problem by Wanner and Hairer.[42] This model has been developed to describe the chemical kinetics of the following autocatalytic reaction:

$$A \xrightarrow{k_1} B,$$

$$B + B \xrightarrow{k_2} C + B,$$

$$B + C \xrightarrow{k_3} A + C,$$

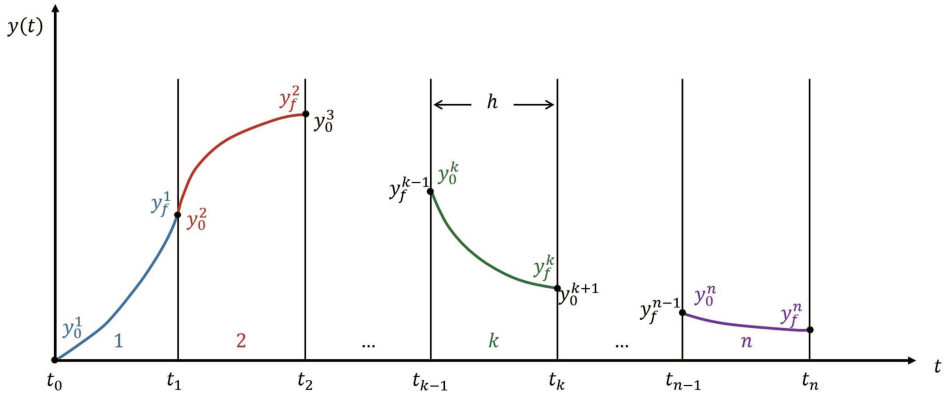


FIG. 3. Decomposition of the time domain into subintervals of length h.

where k_1, k_2, and k_3 are the reaction rate constants, and A, B, and C are the chemical species, whose evolution of concentrations (y_1, y_2, y_3) is governed by the following system of three non-linear ODEs:

$$\begin{cases} y_1' = -k_1 y_1 + k_3 y_2 y_3, \\ y_2' = k_1 y_1 - k_2 y_2^2 - k_3 y_2 y_3, \\ y_3' = k_2 y_2^2, \end{cases} \quad \text{s.t.} \quad \begin{cases} y_1(0) = 1, \\ y_2(0) = 0, \\ y_3(0) = 0. \end{cases} \tag{42}$$

The values of the reaction rate constants are $k_1 = 0.04$, $k_2 = 3 \times 10^7$, and $k_3 = 10^4$, thus varying in a range of 9 orders. This large variation of the parameters is the cause of the strong stiffness of the problem. The constrained expressions and their derivatives are

$$\begin{aligned} y_1(t) &= (\boldsymbol{\sigma} - \boldsymbol{\sigma}_0)\boldsymbol{\beta}_1 + y_1(0), \quad & y_1'(t) &= c\boldsymbol{\sigma}'\boldsymbol{\beta}_1, \\ y_2(t) &= (\boldsymbol{\sigma} - \boldsymbol{\sigma}_0)\boldsymbol{\beta}_2 + y_2(0), \quad & y_2'(t) &= c\boldsymbol{\sigma}'\boldsymbol{\beta}_2, \\ y_3(t) &= (\boldsymbol{\sigma} - \boldsymbol{\sigma}_0)\boldsymbol{\beta}_3 + y_3(0), \quad & y_3'(t) &= c\boldsymbol{\sigma}'\boldsymbol{\beta}_3. \end{aligned}$$

As we are facing a non-linear problem, we need to compute the losses and their derivatives

$$\begin{aligned} \mathcal{L}_1 &= y_1' + k_1 y_1 - k_3 y_2 y_3, \\ \mathcal{L}_2 &= y_2' - k_1 y_1 + k_2 y_2^2 + k_3 y_2 y_3, \\ \mathcal{L}_3 &= y_3' - k_2 y_2^2, \\ \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_1} &= c\boldsymbol{\sigma}' + k_1(\boldsymbol{\sigma} - \boldsymbol{\sigma}_0), \\ \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_2} &= -k_3 y_3(\boldsymbol{\sigma} - \boldsymbol{\sigma}_0), \\ \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_3} &= -k_3 y_2(\boldsymbol{\sigma} - \boldsymbol{\sigma}_0), \\ \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_1} &= -k_1(\boldsymbol{\sigma} - \boldsymbol{\sigma}_0), \\ \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_2} &= c\boldsymbol{\sigma}' + (2k_2 y_2 + k_3 y_3)(\boldsymbol{\sigma} - \boldsymbol{\sigma}_0), \\ \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_3} &= k_3 y_2(\boldsymbol{\sigma} - \boldsymbol{\sigma}_0,) \\ \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_1} &= \mathbf{0}, \\ \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_2} &= -2k_2 y_2(\boldsymbol{\sigma} - \boldsymbol{\sigma}_0), \\ \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_3} &= c\boldsymbol{\sigma}'. \end{aligned}$$

The linear system that we want to iteratively solve is $\mathbb{J}\Delta\beta = \mathcal{L}$, that is,

$$\begin{bmatrix} \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_1} & \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_2} & \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_3} \\ \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_1} & \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_2} & \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_3} \\ \mathbf{0} & \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_2} & \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_3} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\beta}_1 \\ \Delta\boldsymbol{\beta}_2 \\ \Delta\boldsymbol{\beta}_3 \end{bmatrix} = \begin{bmatrix} \mathcal{L}_1 \\ \mathcal{L}_2 \\ \mathcal{L}_3 \end{bmatrix}.$$

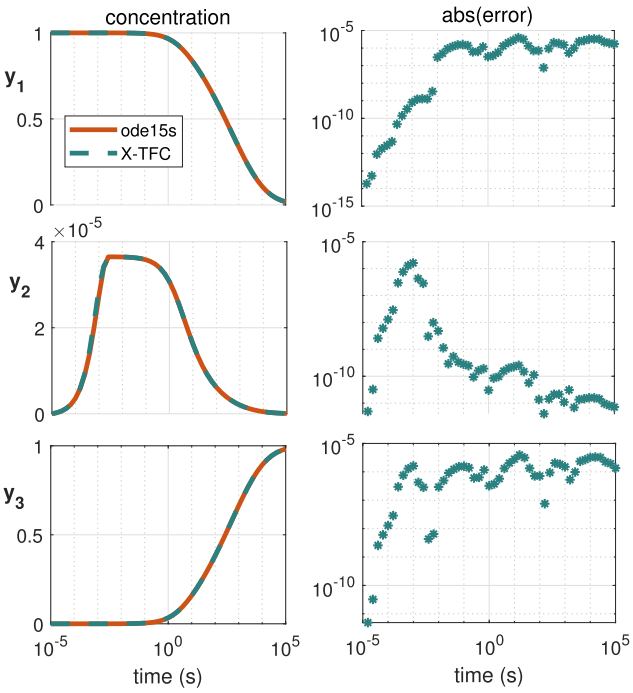


FIG. 4. Species concentration of the ROBER problem computed with MATLAB ode15s and X-TFC (top) and absolute values of the errors between the two methods (bottom).

In Table I, the solutions over $t = [0, 4000]$ are reported. The results for $t = 0.4, 40, 4000$ are obtained with $L = 11, 9, 10$, and $n_x = 11, 9, 10$, respectively, fixed step size $h = 0.001$, and tolerance of 10^{-12}. The training points are uniformly spaced in a linear time scale. The computational times for $t = 0.4, 40, 4000$ are ≈ 0.06 , 6.7, and 580 s, respectively. The average training errors for the three time instants are 7.3×10^{-16}, 7.1×10^{-17}, and 3.2×10^{-13}, respectively.

In this table, we can appreciate a good compatibility with the hybrid method presented in Ref. 45, the MEBDF code,[46] and LSODE method presented by Shampine,[47] up to the 11th significant digit.

In Fig. 4, we reported X-TFC solutions compared with those obtained by the MATLAB stiff ode solver *ode15s*, which we consider here the true solution for our qualitative and quantitative comparison. The time span for producing this plot is $t \in [10^{-5}, 10^5]$, and the training points are uniformly spaced in a logarithmic time scale. The step size used is $h = 2000$, which allows us to obtain these results in a computational time of about 0.015 s, which is particularly low, contrary to what is reported in Ref. 19. The average training error is 8.5×10^{-12}. In the top three subplots, we can see how our solutions overlap with those computed by *ode15s*, with absolute error in the order of $10^{-15}/10^{-05}$ (bottom subplots). This proves that the X-TFC method is much more reliable in solving the ROBER problem than the regular PINN, which fails in this task, even if artifacts to reduce the stiffness of the ODE system are employed, as as for Stiff-PINNs.[18]

B. Chemical Akzo Nobel problem

The second taste case is named chemical Akzo Nobel problem, and it describes a chemical process which originates from the Akzo Nobel Centeral Research in Arnhem, The Netherlands. In the reactions, the two species *MBT* and *CHA* are initially mixed, while oxygen is continuously added. The resulting species of importance is *CBS*, and secondary products of reactions are *BT*, H_2O, and sulfate. According to Ref. 29, the chemical reaction scheme is as follows:

$$\begin{aligned}
2\,MBT + \frac{1}{2}\,O_2 &\xrightarrow{k_1} MBTS + H_2O,\\
CBS + MBT &\underset{k_2}{\overset{k_2/K}{\rightleftharpoons}} MBTS + CHA\\
MBT + 2\,CHA + O_2 &\xrightarrow{k_3} BT + \text{sulfate},\\
MBT.CHA + \frac{1}{2}O_2 &\xrightarrow{k_4} CBS + H_2O,\\
MBT + CHA &\rightleftharpoons MBT.CHA.
\end{aligned}$$

The last equation describes an equilibrium, where the value $K_s = \dfrac{[MBT \cdot CHA]}{[MBT] \cdot [CHA]}$ is used in parameter estimation. The other balance equations describe chemical reactions. The mathematical model describing this set of reactions is a stiff of 6 non-linear ODEs,

$$\begin{cases} y_1' = -2r_1 + r_2 - r_3 - r_4, \\ y_2' = -\frac{1}{2}r_1 - r_4 - \frac{1}{2}r_5 + F_{in}, \\ y_3' = r_1 - r_2 + r_3, \\ y_4' = -r_2 + r_3 - 2r_4, \\ y_5' = r_2 - r_3 + r_5, \\ y_6' = -r_5 \end{cases} \text{s.t.} \begin{cases} y_1(0) = 0.437, \\ y_2(0) = 0.001\,23, \\ y_3(0) = 0, \\ y_4(0) = 0, \\ y_5(0) = 0, \\ y_6(0) = 0.367, \end{cases} \tag{43}$$

for $t \in [0, 180]$, where the reaction rates are given by

$$r_1 = k_1 y_1^4 y_2^{\frac{1}{2}}, \quad r_2 = k_2 y_3 y_4, \quad r_3 = \frac{k_2}{K} y_1 y_5,$$
$$r_4 = k_3 y_1 y_4^2, \quad r_5 = k_4 y_6^2 y_2^{\frac{1}{2}},$$

and the values of the parameters are

$$k_1 = 18.7, \quad k_2 = 0.58, \quad k_3 = 0.09,$$
$$k_4 = 0.42, \quad K = 34.4.$$

F_{in} denotes the inflow of oxygen per volume unit and satisfies the following:

$$F_{in} = klA\left(\frac{p(O_2)}{H} - y_2\right), \tag{44}$$

where $klA = 3.3$ is the mass transfer coefficient, $H = 737$ is the Henry constant, and $p(O_2) = 0.9$ is the partial oxygen pressure. The chemical process begins by mixing 0.437 mol/l of *MBT* and 0.367 mol/l of *MBT.CHA*, with a concentration of oxygen of 0.001 23 mol/l. The other species are created afterwards, and the process simulation is performed in the range of [0, 180] minutes. Again, we need to compute the losses and their derivatives with the same procedure reported in the previous problem. For this and the following problems, we report only the linear system $\mathbb{J}\Delta\beta = \mathcal{L}$ that we want to solve iteratively,

$$\begin{bmatrix}
\frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_1} & \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_2} & \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_3} & \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_4} & \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_5} & \mathbf{0} \\
\frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_1} & \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_2} & \mathbf{0} & \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_4} & \mathbf{0} & \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_6} \\
\frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_1} & \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_2} & \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_3} & \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_4} & \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_5} & \mathbf{0} \\
\frac{\partial \mathcal{L}_4}{\partial \boldsymbol{\beta}_1} & \mathbf{0} & \frac{\partial \mathcal{L}_4}{\partial \boldsymbol{\beta}_3} & \frac{\partial \mathcal{L}_4}{\partial \boldsymbol{\beta}_4} & \frac{\partial \mathcal{L}_4}{\partial \boldsymbol{\beta}_5} & \mathbf{0} \\
\frac{\partial \mathcal{L}_5}{\partial \boldsymbol{\beta}_1} & \frac{\partial \mathcal{L}_5}{\partial \boldsymbol{\beta}_2} & \frac{\partial \mathcal{L}_5}{\partial \boldsymbol{\beta}_3} & \frac{\partial \mathcal{L}_5}{\partial \boldsymbol{\beta}_4} & \frac{\partial \mathcal{L}_5}{\partial \boldsymbol{\beta}_5} & \frac{\partial \mathcal{L}_5}{\partial \boldsymbol{\beta}_6} \\
\mathbf{0} & \frac{\partial \mathcal{L}_6}{\partial \boldsymbol{\beta}_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathcal{L}_6}{\partial \boldsymbol{\beta}_6}
\end{bmatrix}
\begin{bmatrix} \Delta\boldsymbol{\beta}_1 \\ \Delta\boldsymbol{\beta}_2 \\ \Delta\boldsymbol{\beta}_3 \\ \Delta\boldsymbol{\beta}_4 \\ \Delta\boldsymbol{\beta}_5 \\ \Delta\boldsymbol{\beta}_6 \end{bmatrix} =
\begin{bmatrix} \mathcal{L}_1 \\ \mathcal{L}_2 \\ \mathcal{L}_3 \\ \mathcal{L}_4 \\ \mathcal{L}_5 \\ \mathcal{L}_6 \end{bmatrix}.$$

In Table II, the solutions of the concentration at the final time $t = 180$ are reported, obtained with $L = 20$, $n_x = 50$, fixed step size $h = 0.01$, and tolerance of 10^{-12}. The training points are uniformly spaced in a linear time scale. The computational times for $t = [0, 180]$ is ≈ 64 s with an average training error of 2.1×10^{-16}. In this Table, we compare our results with the *Test set for IVP solvers* benchmarks presented in Ref. 48, and the results obtained using the

TABLE I. Species concentrations at times $t = 0.4$, 40, 4000 of X-TFC, Ref. 45, MEBDF, and SDMM, for the ROBER problem.

t	$\mathbf{y_i}$	X-TFC	Hybrid method	MEBDF	LSODE
0.4	y_1	9.851 721 138 609 84E−01	9.851 721 138 632 31E−01	9.851 721 138 61E−01	9.851 7E−01
	y_2	3.386 395 378 974 81E−05	3.386 395 378 909 47E−05	3.386 397 378 97E−05	3.386 4E−05
	y_3	1.479 402 218 522 60E−02	1.479 402 218 548 82E−02	1.479 402 218 54E−02	1.479 4E−02
40	y_1	7.158 270 687 081 93E−01	7.158 270 687 189 94E−01	7.158 270 687 82E−01	7.158 2E−01
	y_2	9.185 534 764 122 04E−06	9.185 534 764 567 52E−06	9.185 534 765 64E−06	9.185 1E−06
	y_3	2.84163745757042E−01	2.841 637 457 463 61E−01	2.841 637 457 33E−01	2.841 7E−01
4000	y_1	1.832 022 577 274 15E−01	1.832 020 418 731 52E−01	1.832 022 818 48E−01	1.832 0E−01
	y_2	8.942 371 250 213 86E−07	8.942 358 400 025 91E−07	9.942 372 681 15E−07	8.942 3E−07
	y_3	8.167 968 480 012 25E−01	8.167 970 638 913 67E−01	8.167 941 451 52E−01	8.168 0E−01

TABLE II. Species concentrations at time $t = 180$ of X-TFC, test set IVP solvers,[48] and exponential propagation method,[49] for the chemical Akzo Nobel problem.

t	$\mathbf{y_i}$	X-TFC	Test set IVP solvers	Exp propagation
180	y_1	1.161 602 274 780 192E−01	1.161 602 274 780 192E−01	1.161 583 704 781 95E−01
	y_2	1.119 418 166 040 844E−03	1.119 418 166 040 848E−03	0.011 194 091 556 33E−01
	y_3	1.621 261 719 785 830E−01	1.621 261 719 785 814E−01	1.621 265 677 283 22E−01
	y_4	3.396 981 299 297505E−03	3.396 981 299 29 7459E−03	0.033 959 155 455 71E−01
	y_5	1.646 185 108 335 093E−01	1.646 185 108 335 055E−01	1.646 185 137 343 56E−01
	y_6	1.989 533 275 954337E−01	1.989 533 275 954 281E−01	1.989 543 899 415 37E−01

exponential propagation method from Ref. 49. As one can see, our results match from 13 to 16 digits of accuracy with the benchmarks of Ref. 48, while with the exponential propagation method there is a compatibility up to the 8th digit (different figures are marked in bold).

In Fig. 5, we reported X-TFC solutions compared with those obtained by the MATLAB stiff ode solver *ode15s*, which we consider here the true solution for our qualitative and quantitative comparison. In the concentration subplots, we can see how our solutions overlap with those computed by *ode15s*, with absolute error in the order of $10^{-15}/10^{-05}$ (error subplots). This shows the accuracy of X-TFC for larger time steps (lower computational times).

C. Belousov–Zhabotinsky reaction

The third test case we consider is the Belousov–Zhabotinsky reaction mechanism,[30] which can be represented by the following scheme of homogeneous chemical reactions in a symbolic

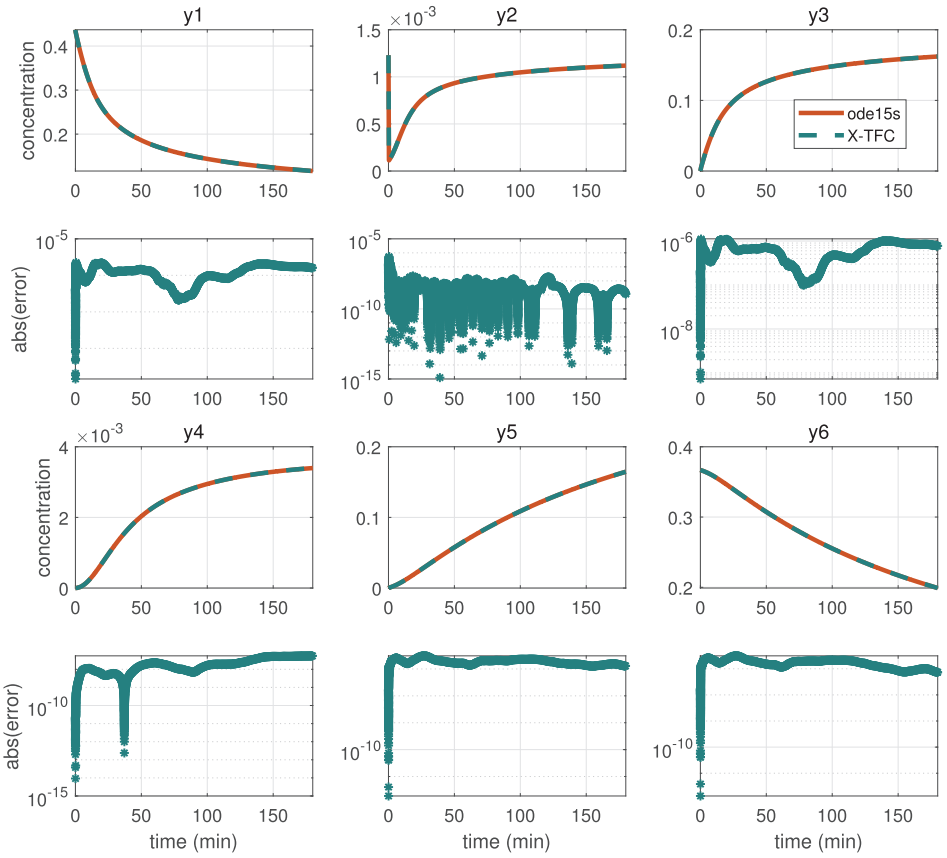


FIG. 5. Species concentration of the chemical Akzo Nobel problem computed with MATLAB ode15s and X-TFC, and absolute values of the errors between the two methods.

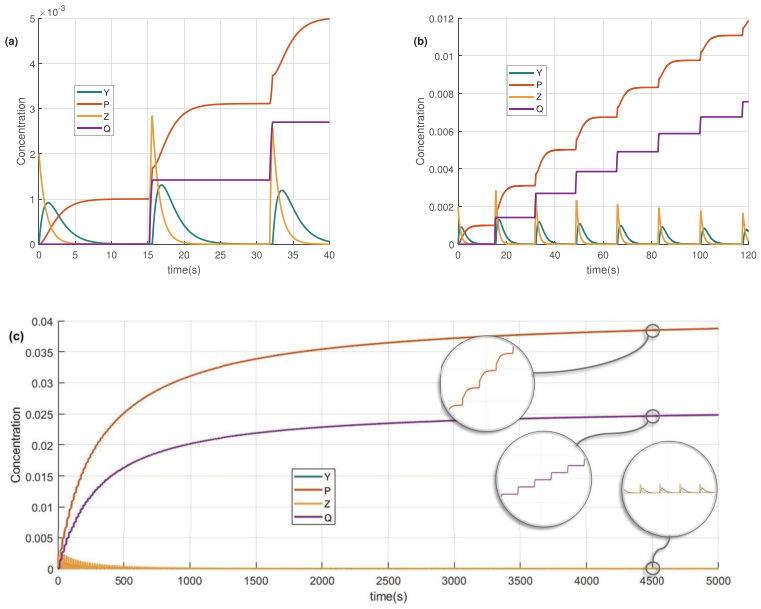


FIG. 6. Species concentrations (Y, P, Z, Q) of the Belousov–Zhabotinsky reaction computed with X-TFC for integration time of 40 s (a) and 120 s (b). A simulation for integration time of 5000 s shows the robustness of the X-TFC method for very large time domains (c).

form:

$$\begin{aligned} A + Y &\xrightarrow{k_1} X, \\ X + Y &\xrightarrow{k_2} P, \\ B + X &\xrightarrow{k_3} 2X + Z, \\ 2X &\xrightarrow{k_4} Q, \\ Z &\xrightarrow{k_5} Y, \end{aligned} \tag{45}$$

where the reaction rate constants are $k_1 = 4.72$, $k_2 = 3 \times 10^9$, $k_3 = 1.5 \times 10^4$, $k_4 = 4 \times 10^7$, and $k_5 = 1\left[\frac{l}{\text{mol}\, s}\right]$. The letters denote the species taking part of the reactions, and the initial concentrations at time $t = 0$ (expressed in $\frac{mol}{l}$) are

$$Y = X = P = Q = 0, \quad A = B = 0.066, \; Z = 0.002.$$

The system of ODE modeling the Belousov–Zhabotinsky reaction is

$$\begin{cases} y_1' = -k_1 y_1 y_2, \\ y_2' = -k_1 y_1 y_2 - k_2 y_3 y_2 + k_5 y_6, \\ y_3' = -k_2 y_3 y_2 + k_3 y_3 y_5 - 2k_4 y_3^2 + k_1 y_1 y_2, \\ y_4' = k_2 y_3 y_2, \\ y_5' = -k_3 y_5 y_3, \\ y_6' = k_3 y_5 y_3 - k_5 y_6, \\ y_7' = k_4 y_3^2, \end{cases} \tag{46}$$

subject to $y_i(0) = (0.066, 0, 0, 0, 0.066, 0.002, 0)^T$, for $t \in [0, 40]$. After computing the losses and their derivatives, we obtain the following linear system:

$$\begin{bmatrix} \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_1} & \frac{\partial \mathcal{L}_1}{\partial \boldsymbol{\beta}_2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_1} & \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_2} & \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_3} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathcal{L}_2}{\partial \boldsymbol{\beta}_6} & \mathbf{0} \\ \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_1} & \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_2} & \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_3} & \mathbf{0} & \frac{\partial \mathcal{L}_3}{\partial \boldsymbol{\beta}_5} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathcal{L}_4}{\partial \boldsymbol{\beta}_2} & \frac{\partial \mathcal{L}_4}{\partial \boldsymbol{\beta}_3} & \frac{\partial \mathcal{L}_4}{\partial \boldsymbol{\beta}_4} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathcal{L}_5}{\partial \boldsymbol{\beta}_3} & \mathbf{0} & \frac{\partial \mathcal{L}_5}{\partial \boldsymbol{\beta}_5} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathcal{L}_6}{\partial \boldsymbol{\beta}_3} & \mathbf{0} & \frac{\partial \mathcal{L}_6}{\partial \boldsymbol{\beta}_5} & \frac{\partial \mathcal{L}_6}{\partial \boldsymbol{\beta}_6} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\partial \mathcal{L}_7}{\partial \boldsymbol{\beta}_3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathcal{L}_7}{\partial \boldsymbol{\beta}_7} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta\beta}_1 \\ \boldsymbol{\Delta\beta}_2 \\ \boldsymbol{\Delta\beta}_3 \\ \boldsymbol{\Delta\beta}_4 \\ \boldsymbol{\Delta\beta}_5 \\ \boldsymbol{\Delta\beta}_6 \\ \boldsymbol{\Delta\beta}_7 \end{bmatrix} = \begin{bmatrix} \mathcal{L}_1 \\ \mathcal{L}_2 \\ \mathcal{L}_3 \\ \mathcal{L}_4 \\ \mathcal{L}_5 \\ \mathcal{L}_6 \\ \mathcal{L}_7 \end{bmatrix}. \tag{47}$$

Figure 6 shows the simulation results obtained using X-TFC with $L = 20$, $n_x = 20$, fixed step size $h = 0.01$, and tolerance of 10^{-9}, for three different integration times. The training points are uniformly spaced in a linear time scale, and the computational times for $t = [0, 40]$ is ≈ 3 s. One can clearly see the rapid "jumps" in the solutions with a period of approximately 16 s. The qualitative analogy with the converged solutions presented by Gear's method[50] and the Almost Runge–Kutta and Aluffi–Pentini methods (obtained with CPU times of ≈ 9 s)[51] is evident. As one can see from Fig. 6(c), we can extend the integration time to a very large range (e.g., 5000 s) while continuing to obtain accurate solutions without negative values.

Two MATLAB functions, *ode15s* and *ode15i*, have been implemented to compare their results with the X-TFC framework. The

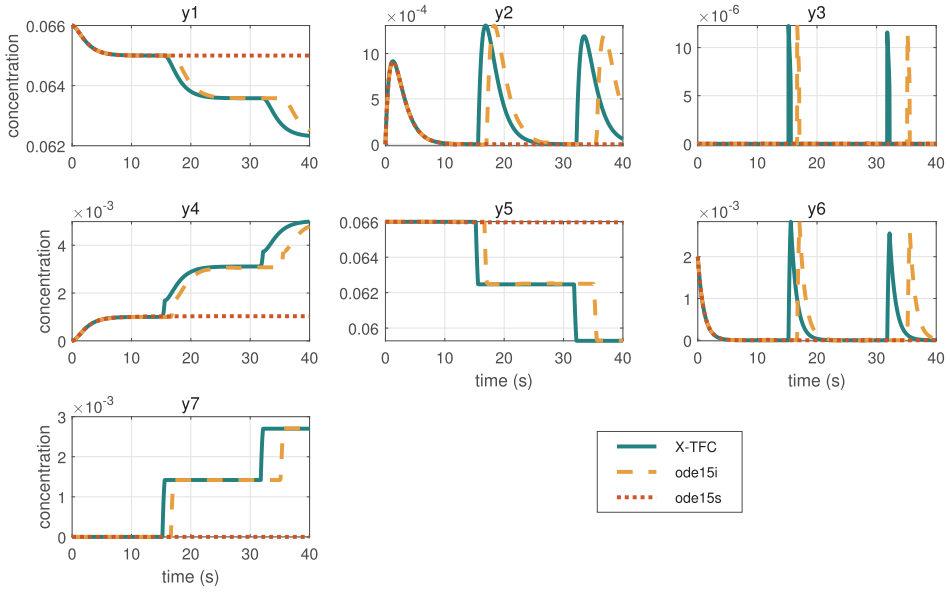


FIG. 7. Species concentrations of the Belousov–Zhabotinsky reaction computed with X-TFC, ode15i, and ode15s, for integration time of 40 s.

solutions of the three methods for all the seven species reacting in the chemical process are reported in Figs. 7 and 8, for the integration times of 40 and 120 s, respectively. Both solutions of *ode15s* and *ode15i* have been obtained with time step $h = 0.0001$ and tolerance of 10^{-9}. We can see that *ode*15*s* is able to find the solution only for the first jump, after which it fails, keeping the species concentrations constant in time. From Fig. 7, we can note that *ode*15*i* accurately captures the solutions for the first jump, but with a delay of about 1 s for the second jump and about 3 s for the third jump. By extending the time interval (see Fig. 8), we see that after the third jump, it stop working, keeping the species concentrations constant in time.

D. POLLU problem

The fourth and last test case we use in this work to test the robustness and efficiency of our method is a realistic air pollution model[52,53] from atmospheric chemistry with 20 components and 25 reactions. This air pollution model, also known as POLLU problem, has been developed at The Dutch National Institute of Public Health and Environmental Protection (RIVM) and described by Verwer.[21] The reaction scheme, reaction rates, and rate constants are given in Table III.

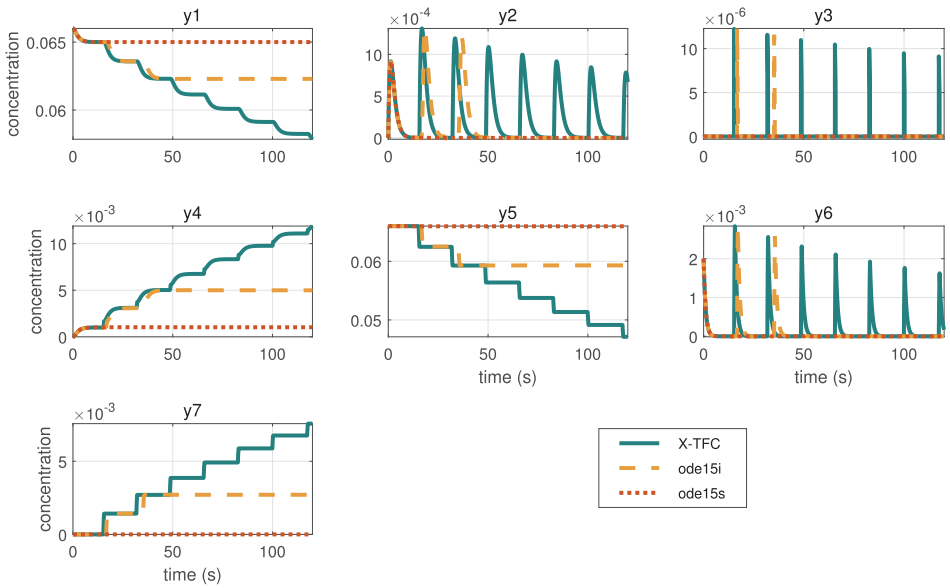


FIG. 8. Species concentrations of the Belousov–Zhabotinsky reaction computed with X-TFC, ode15i, and ode15s, for integration time of 120 s.

The mathematical modeling of the POLLU problem is given by the following system of 20 ODEs with initial conditions:

$$\begin{cases} y_1' = -r_1 - r_{10} - r_{14} - r_{23} - r_{24} + r_2 + r_3 + r_9 + r_{11} + r_{12} + r_{22} + r_{25}, \\ y_2' = -r_2 - r_3 - r_9 - r_{12} + r_1 + r_{21}, \\ y_3' = -r_{15} + r_1 + r_{17} + r_{19} + r_{22}, \\ y_4' = -r_2 - r_{16} - r_{17} - r_{23} + r_{15}, \\ y_5' = -r_3 + 2r_4 + r_6 + r_7 + r_{13} + r_{20}, \\ y_6' = -r_6 - r_8 - r_{14} - r_{20} + r_3 + 2r_{18}, \\ y_7' = -r_4 - r_5 - r_6 + r_{13}, \\ y_8' = r_4 + r_5 + r_6 + r_7, \\ y_9' = -r_7 - r_8, \\ y_{10}' = -r_{12} + r_7 + r_9, \\ y_{11}' = -r_9 - r_{10} + r_8 + r_{11}, \\ y_{12}' = r_9, \\ y_{13}' = -r_{11} + r_{10}, \\ y_{14}' = -r_{13} + r_{12}, \\ y_{15}' = r_{14}, \\ y_{16}' = -r_{18} - r_{19} + r_{16}, \\ y_{17}' = -r_{20}, \\ y_{18}' = r_{20}, \\ y_{19}' = -r_{21} - r_{22} - r_{24} + r_{23} + r_{25}, \\ y_{20}' = -r_{25} + r_{24}, \end{cases} \quad \text{s.t.} \quad \begin{cases} y_1(0) = 0, \\ y_2(0) = 0.2, \\ y_3(0) = 0, \\ y_4(0) = 0.04, \\ y_5(0) = 0, \\ y_6(0) = 0, \\ y_7(0) = 0.1, \\ y_8(0) = 0.3, \\ y_9(0) = 0.01, \\ y_{10}(0) = 0, \\ y_{11}(0) = 0, \\ y_{12}(0) = 0, \\ y_{13}(0) = 0, \\ y_{14}(0) = 0, \\ y_{15}(0) = 0, \\ y_{16}(0) = 0, \\ y_{17}(0) = 0.007, \\ y_{18}(0) = 0, \\ y_{19}(0) = 0, \\ y_{20}(0) = 0, \end{cases} \tag{48}$$

After computing the losses and their derivatives, we obtain the following linear system, where the symbol ★ represents the non-zero terms:

$$\begin{bmatrix} ★ & ★ & 0 & ★ & ★ & ★ & 0 & 0 & 0 & ★ & ★ & 0 & ★ & 0 & 0 & 0 & 0 & 0 & ★ & ★ \\ ★ & ★ & 0 & ★ & ★ & 0 & 0 & 0 & 0 & ★ & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ★ & 0 \\ ★ & 0 & ★ & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ★ & 0 & 0 & ★ & 0 \\ ★ & ★ & ★ & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ ★ & ★ & 0 & 0 & ★ & ★ & ★ & 0 & ★ & 0 & 0 & 0 & 0 & ★ & 0 & 0 & ★ & 0 & 0 & 0 \\ ★ & ★ & 0 & 0 & ★ & ★ & ★ & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & ★ & ★ & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & ★ & ★ & 0 & 0 & 0 & 0 & 0 & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & ★ & ★ & ★ & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & ★ & 0 & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & ★ & ★ & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ ★ & ★ & 0 & 0 & 0 & ★ & 0 & 0 & ★ & 0 & ★ & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ★ & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ ★ & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ★ & 0 & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ ★ & 0 & 0 & 0 & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ★ & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ★ & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ★ & ★ & 0 & 0 & 0 \\ ★ & 0 & 0 & ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ★ & ★ & 0 \\ ★ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ★ & ★ & 0 \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\beta}_1 \\ \Delta\boldsymbol{\beta}_2 \\ \Delta\boldsymbol{\beta}_3 \\ \Delta\boldsymbol{\beta}_4 \\ \Delta\boldsymbol{\beta}_5 \\ \Delta\boldsymbol{\beta}_6 \\ \Delta\boldsymbol{\beta}_7 \\ \Delta\boldsymbol{\beta}_8 \\ \Delta\boldsymbol{\beta}_9 \\ \Delta\boldsymbol{\beta}_{10} \\ \Delta\boldsymbol{\beta}_{11} \\ \Delta\boldsymbol{\beta}_{12} \\ \Delta\boldsymbol{\beta}_{13} \\ \Delta\boldsymbol{\beta}_{14} \\ \Delta\boldsymbol{\beta}_{15} \\ \Delta\boldsymbol{\beta}_{16} \\ \Delta\boldsymbol{\beta}_{17} \\ \Delta\boldsymbol{\beta}_{18} \\ \Delta\boldsymbol{\beta}_{19} \\ \Delta\boldsymbol{\beta}_{20} \end{bmatrix} = \begin{bmatrix} \mathcal{L}_1 \\ \mathcal{L}_2 \\ \mathcal{L}_3 \\ \mathcal{L}_4 \\ \mathcal{L}_5 \\ \mathcal{L}_6 \\ \mathcal{L}_7 \\ \mathcal{L}_8 \\ \mathcal{L}_9 \\ \mathcal{L}_{10} \\ \mathcal{L}_{11} \\ \mathcal{L}_{12} \\ \mathcal{L}_{13} \\ \mathcal{L}_{14} \\ \mathcal{L}_{15} \\ \mathcal{L}_{16} \\ \mathcal{L}_{17} \\ \mathcal{L}_{18} \\ \mathcal{L}_{19} \\ \mathcal{L}_{20} \end{bmatrix}.$$

In Figs. 9 and 10, we reported X-TFC solutions and the absolute errors, respectively, compared with the solutions obtained by the MATLAB stiff ode solver *ode15s*, which we consider here the true solution for our qualitative and quantitative comparison. The results are obtained with $L = 10$, $n_x = 10$, fixed step size $h = 0.001$, and tolerance of 10^{-9}. The training points are uniformly spaced in a linear time scale. The computational times for $t = [0\ \ 60]$ is ≈ 380 s (against 2 h needed in Ref. 19) and the average training error

TABLE III. Reaction scheme of the air pollution problem.

No.	Reaction	Reaction rate	Rate constant
1	$NO_2 \rightarrow NO + O_3\ P$	$r_1 = k_1\ y_1$	$k_1 = 0.35$
2	$NO + O_3 \rightarrow NO_2$	$r_2 = k_2\ y_2\ y_4$	$k_2 = 26.6$
3	$HO_2 + NO \rightarrow NO_2 + OH$	$r_3 = k_3\ y_5\ y_2$	$k_3 = 0.123 \times 10^5$
4	$HCHO \rightarrow 2HO_2 + CO$	$r_4 = k_4\ y_7$	$k_4 = 0.86 \times 10^{-3}$
5	$HCHO + OH \rightarrow HO_2 + CO$	$r_5 = k_5\ y_7$	$k_5 = 0.82 \times 10^{-3}$
6	$HCHO \rightarrow CO$	$r_6 = k_6\ y_7\ y_6$	$k_6 = 0.15 \times 10^5$
7	$ALD \rightarrow MEO_2 + HO_2 + CO$	$r_7 = k_7\ y_9$	$k_7 = 0.13 \times 10^{-3}$
8	$ALD + OH \rightarrow C_2O_3$	$r_8 = k_8\ y_9\ y_6$	$k_8 = 0.24 \times 10^5$
9	$C_2O_3 + NO \rightarrow NO_2 + MEO_2 + CO_2$	$r_9 = k_9\ y_{11}\ y_2$	$k_9 = 0.165 \times 10^5$
10	$C_2O_3 + NO_2 \rightarrow PAN$	$r_{10} = k_{10}\ y_{11}\ y_1$	$k_{10} = 0.9 \times 10^4$
11	$PAN \rightarrow C_2O_3 + NO_2$	$r_{11} = k_{11}\ y_{13}$	$k_{11} = 0.22 \times 10^{-1}$
12	$MEO_2 + NO \rightarrow CH_3O + NO_2$	$r_{12} = k_{12}\ y_{10}\ y_2$	$k_{12} = 0.12 \times 10^5$
13	$CH_3O \rightarrow HCHO + HO_2$	$r_{13} = k_{13}\ y_{14}$	$k_{13} = 1.88$
14	$NO_2 + OH \rightarrow HNO_3$	$r_{14} = k_{14}\ y_1\ y_6$	$k_{14} = 0.163 \times 10^5$
15	$O_3P \rightarrow O_3$	$r_{15} = k_{15}\ y_3$	$k_{15} = 0.48 \times 10^7$
16	$O_3 \rightarrow O_1D$	$r_{16} = k_{16}\ y_4$	$k_{16} = 0.35 \times 10^{-3}$
17	$O_3 \rightarrow O_3P$	$r_{17} = k_{17}\ y_4$	$k_{17} = 0.0175$
18	$O_1D \rightarrow 2OH$	$r_{18} = k_{18}\ y_{16}$	$k_{18} = 0.1 \times 10^9$
19	$O_1D \rightarrow O_3P$	$r_{19} = k_{19}\ y_{16}$	$k_{19} = 0.444 \times 10^{12}$
20	$SO_2 + OH \rightarrow SO_4 + HO_2$	$r_{20} = k_{20}\ y_{17} y_6$	$k_{20} = 0.124 \times 10^4$
21	$NO_3 \rightarrow NO$	$r_{21} = k_{21}\ y_{19}$	$k_{21} = 2.1$
22	$NO_3 \rightarrow NO_2 + O_3P$	$r_{22} = k_{22}\ y_{19}$	$k_{22} = 5.78$
23	$NO_2 + O_3 \rightarrow NO_3$	$r_{23} = k_{23}\ y_1\ y_4$	$k_{23} = 0.0474$
24	$NO_3 + NO_2 \rightarrow N_2O_5$	$r_{24} = k_{24}\ y_{19}\ y_1$	$k_{24} = 0.178 \times 10^4$
25	$N_2O_5 \rightarrow NO_3 + NO_2$	$r_{25} = k_{25}\ y_{20}$	$k_{25} = 3.12$

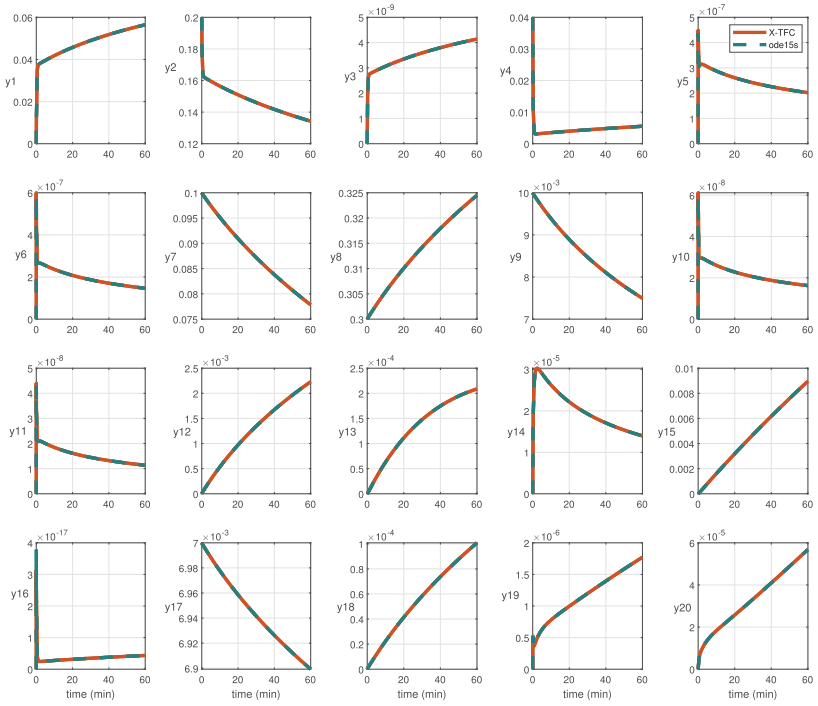


FIG. 9. Species concentrations of the air pollution (POLLU) problem computed with MATLAB ode15s and X-TFC.

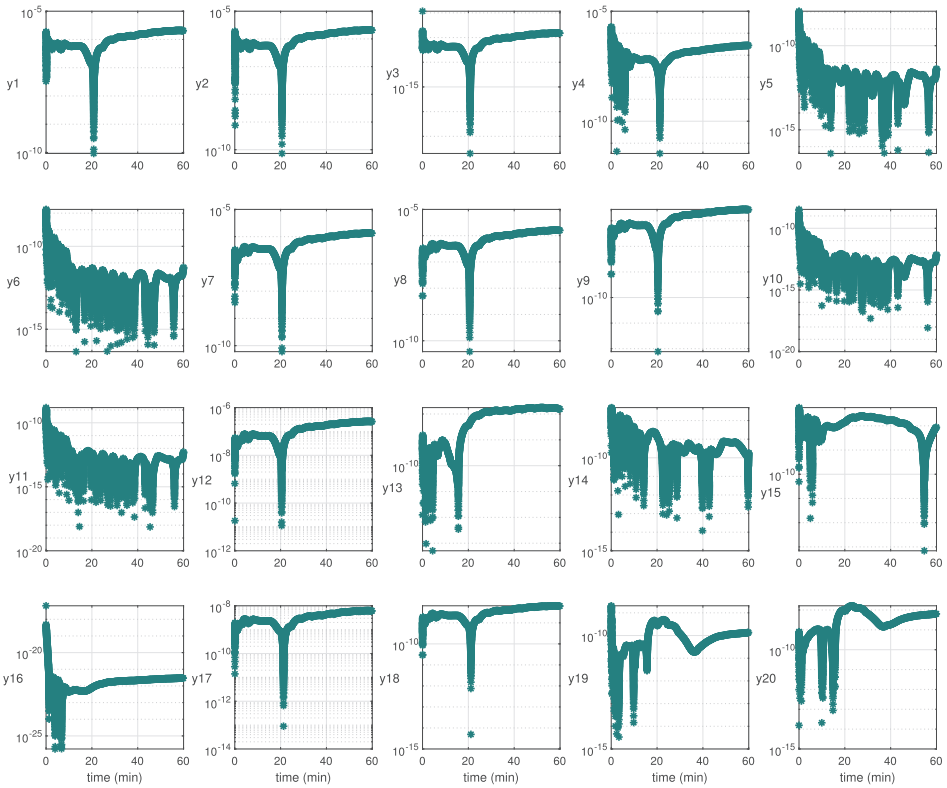


FIG. 10. Absolute values of the errors between X-TFC and MATLAB *ode15s* for each species concentration of the air pollution (POLLU) problem.

is 3.5×10^{-08}. We can see how our solutions overlap with those computed by *ode15s*, and the absolute errors are in the order of $10^{-25}/10^{-05}$.

V. CONCLUSIONS

A new, efficient, and precise algorithm, named X-TFC, is employed in this work for solving stiff chemical kinetics problems. In X-TFC, the physics-informed neural network idea has been merged with the theory of functional connections to analytically satisfy the initial conditions, removing the issue of having competing objectives in the loss function. To reach high computational efficiency, the shallow NN is trained via the ELM algorithm. An important highlight of X-TFC is that we obtain an analytical representation (via NNs) of chemical kinetics equations' solutions once the NN output weights are evaluated on the training points. Thus, the solutions can be calculated at any required time instant in the domain without further computational efforts (e.g., the NN does not need to be re-trained).

In this work, the X-TFC algorithm with time decomposition has been applied to solve stiff systems of ODEs, showing its accuracy and efficiency by comparing it with state-of-the-art methods and benchmarks in the literature. No artifacts were needed to reduce the stiffness of the problems, directly solving them in their original form. For the first time, we provided a rigorous upper bound on the generalization error of X-TFC frameworks in learning the solutions of IVPs for ODEs. Source code is available from the authors upon reasonable request.

Future works will focus on stiff problems related to different fields such as biological systems. Also, the authors are currently working on estimating the generalization error of X-TFC frameworks for boundary value problems.

ACKNOWLEDGMENTS

The authors would like to acknowledge Dr. Thomas Rigotti (Department of Chemistry, The Technical University of Munich) and Dr. Andrea D'Ambrosio (Department of Aeronautics and Astronautics, Massachusetts Institute of Technology) for fruitful discussions, and Jason Curtis for his support during the drafting of the manuscript.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

DATA AVAILABILITY

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

REFERENCES

[1] H. Lee and I. S. Kang, "Neural algorithm for solving differential equations," J. Comput. Phys. **91**(1), 110–131 (1990).
[2] A. J. Meade, Jr. and A. A. Fernandez, "The numerical solution of linear ordinary differential equations by feedforward neural networks," Math. Comput. Model. **19**(12), 1–25 (1994).
[3] A. J. Meade, Jr. and A. A. Fernandez, "Solution of nonlinear ordinary differential equations by feedforward neural networks," Math. Comput. Model. **20**(9), 19–44 (1994).
[4] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," IEEE Trans. Neural Netw. **9**(5), 987–1000 (1998).
[5] C. Filici, "On a neural approximator to ODEs," IEEE Trans. Neural Netw. **19**(3), 539–543 (2008).
[6] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, "User guide for minpack-1," Technical Report, CM-P00068642 (1980).
[7] R. Gerstberger and P. Rentrop, "Feedforward neural nets as discretization schemes for ODEs and DAEs," J. Comput. Appl. Math. **82**(1-2), 117–128 (1997).
[8] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[9] S. Dong and Z. Li, "Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations," Comput. Methods Appl. Mech. Eng. **387**, 114129 (2021).
[10] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," Comput. Methods Appl. Mech. Eng. **365**, 113028 (2020).
[11] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," Nat. Rev. Phys. **3**(6), 422–440 (2021).
[12] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, "Ppinn: Parareal physics-informed neural network for time-dependent PDEs," Comput. Methods Appl. Mech. Eng. **370**, 113250 (2020).
[13] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient flow pathologies in physics-informed neural networks," SIAM J. Sci. Comput. **43**(5), A3055–A3081 (2021).
[14] D. A. Goussis and M. Valorani, "An efficient iterative algorithm for the approximation of the fast and slow dynamics of stiff systems," J. Comput. Phys. **214**(1), 316–346 (2006).
[15] M. Hadjinicolaou and D. A. Goussis, "Asymptotic solution of stiff PDEs with the CSP method: The reaction diffusion equation," SIAM J. Sci. Comput. **20**(3), 781–810 (1998).
[16] M. Valorani and D. A. Goussis, "Explicit time-scale splitting algorithm for stiff problems: Auto-ignition of gaseous mixtures behind a steady shock," J. Comput. Phys. **169**(1), 44–79 (2001).
[17] E. Galaris, F. Calabrò, D. di Serafino, and C. Siettos, "Numerical solution of stiff ordinary differential equations with random projection neural networks," arXiv:2108.01584 (2021).
[18] W. Ji, W. Qiu, Z. Shi, S. Pan, and S. Deng, "Stiff-PINN: Physics-informed neural network for stiff chemical kinetics," J. Phys. Chem. A **125**(36), 8098–8106 (2021).
[19] S. Kim, W. Ji, S. Deng, Y. Ma, and C. Rackauckas, "Stiff neural ordinary differential equations," Chaos **31**(9), 093122 (2021).
[20] H. Robertson, "The solution of a set of reaction rate equations," *Numerical Analysis: An Introduction* (Academic Press, 1966), Vol. 178182.
[21] J. G. Verwer, "Gauss–Seidel iteration for stiff ODEs from chemical kinetics," SIAM J. Sci. Comput. **15**(5), 1243–1250 (1994).
[22] E. Schiassi, R. Furfaro, C. Leake, M. De Florio, H. Johnston, and D. Mortari, "Extreme theory of functional connections: A fast physics-informed neural network method for solving ordinary and partial differential equations," Neurocomputing **457**, 334–356 (2021).
[23] M. De Florio, E. Schiassi, B. D. Ganapol, and R. Furfaro, "Physics-informed neural networks for rarefied-gas dynamics: Thermal creep flow in the Bhatnagar–Gross–Krook approximation," Phys. Fluids **33**(4), 047110 (2021).
[24] M. De Florio, E. Schiassi, R. Furfaro, B. D. Ganapol, and D. Mostacci, "Solutions of Chandrasekhar's basic problem in radiative transfer via theory of functional connections," J. Quant. Spectrosc. Radiat. Trans. **259**, 107384 (2020).
[25] M. De Florio, E. Schiassi, A. D'Ambrosio, D. Mortari, and R. Furfaro, "Theory of functional connections applied to linear ODEs subject to integral constraints and linear ordinary integro-differential equations," Math. Comput. Appl. **26**(3), 65 (2021).
[26] E. Schiassi, M. De Florio, A. D'Ambrosio, D. Mortari, and R. Furfaro, "Physics-informed neural networks and functional interpolation for data-driven parameters discovery of epidemiological compartmental models," Mathematics **9**(17), 2069 (2021).
[27] E. Schiassi, M. De Florio, B. D. Ganapol, P. Picca, and R. Furfaro, "Physics-informed neural networks for the point kinetics equations for nuclear reactor dynamics," Ann. Nucl. Energy **167**, 108833 (2021).
[28] E. Schiassi, A. D'Ambrosio, H. Johnston, M. De Florio, K. Drozd, R. Furfaro, F. Curti, and D. Mortari, "Physics-informed extreme theory of functional connections applied to optimal orbit transfer," in *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Lake Tahoe, CA* (AAS, 2020), pp. 9–13.
[29] K. Eriksson, C. Johnson, and A. Logg, "Explicit time-stepping for stiff ODEs," SIAM J. Sci. Comput. **25**(4), 1142–1157 (2004).
[30] A. M. Zhabotinsky, "Periodical oxidation of malonic acid in solution (a study of the Belousov reaction kinetics)," Biofizika **9**, 306–311 (1964).
[31] J. N. Reddy, "An introduction to the finite element method," J. Pressure Vessel Technol. **111**, 348–349 (1989).
[32] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient pathologies in physics-informed neural networks," arXiv:2001.04536 (2020).
[33] P. Mertikopoulos, C. Papadimitriou, and G. Piliouras, "Cycles in adversarial regularized learning," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms* (SIAM, 2018), pp. 2703–2717.
[34] D. Balduzzi, S. Racaniere, J. Martens, J. Foerster, K. Tuyls, and T. Graepel, "The mechanics of n-player differentiable games," in *International Conference on Machine Learning* (PMLR, 2018), pp. 354–363.
[35] D. Mortari, "The theory of connections: Connecting points," Mathematics **5**(4), 57 (2017).
[36] D. Mortari, "Least-squares solution of linear differential equations," Mathematics **5**(4), 48 (2017).
[37] D. Mortari, H. Johnston, and L. Smith, "High accuracy least-squares solutions of nonlinear differential equations," J. Comput. Appl. Math. **352**, 293–307 (2019).
[38] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," Neurocomputing **70**, 489–501 (2006).
[39] S. Mishra and R. Molinaro, "Estimates on the generalization error of physics informed neural networks (pinns) for approximating PDEs," arXiv:2006.16144 (2020).
[40] M. R. Roussel, "The use of delay differential equations in chemical kinetics," J. Phys. Chem. **100**(20), 8323–8330 (1996).
[41] J. D. Lambert *et al.*, *Numerical Methods for Ordinary Differential Systems* (Wiley, New York, 1991), Vol. 146.
[42] G. Wanner and E. Hairer, *Solving Ordinary Differential Equations II* (Springer, 1996), Vol. 375.
[43] W. M. Lioen and J. J. de Swart, *Test Set for Initial Value Problem Solvers* (Centrum voor Wiskunde en Informatica, 1998).
[44] A. M. Zhabotinskii, "Periodic course of the oxidation of malonic acid in a solution (studies on the kinetics of Beolusov's reaction)," Biofizika **9**, 306–311 (1964).
[45] M. Mehdizadeh Khalsaraei, A. Shokri, and M. Molayi, "The new high approximation of stiff systems of first order IVPs arising from chemical reactions by k-step l-stable hybrid methods," Iranian J. Math. Chem. **10**(2), 181–193 (2019).
[46] T. Abdulla, J. Cash, and M. Diamantakis, "An MEBDF package for the numerical solution of large sparse systems of stiff initial value problems," Comput. Math. Appl. **42**(1-2), 121–129 (2001).

[47]L. F. Shampine, *Numerical Solution of Ordinary Differential Equations* (Routledge, 2018).
[48]W. M. Lioen, J. J. deSwart, and W. van deVeen, "Test set for IVP solvers," *Department of Numerical Mathematics [NM]*, no. R 9615 (1996).
[49]M. Falati and G. Hojjati, "Integration of chemical stiff ODEs using exponential propagation method," J. Math. Chem. **49**(10), 2210–2230 (2011).
[50]O. Klymenko and I. Svir, "Modelling complex chemical processes in homogeneous solutions: Automatic numerical simulation," Nonlinear Anal. Model. Control **11**(3), 247–261 (2006).
[51] V. Shulyk, O. Klymenko, and I. Svir, "Numerical solution of stiff ODEs describing complex homogeneous chemical processes," J. Math. Chem. **43**(1), 252–264 (2008).
[52]A. C. Hindmarsh, "LSODE and LSODI: Two new initial value ordinary differential equation solvers," ACM Signum Newslett. **15**(4), 10–11 (1980).
[53]F. De Leeuw, "Numerical solution of ordinary differential equations arising from chemical kinetics," *RIVM Rapport 228603005* (1988).