Reduced-PINN: An Integration-Based Physics-Informed Neural Networks for Stiff ODEs

Pouyan Nasiri^a, Roozbeh Dargazany^{*a}

^aDepartment of Civil and Environmental Engineering, Michigan State University, USA *corresponding author:e-mail:roozbeh@msu.edu

August 26, 2022

Abstract

Physics-informed neural networks (PINNs) have recently received much attention due to their capabilities in solving both forward and inverse problems. For training a deep neural network associated with a PINN, one typically constructs a total loss function using a weighted sum of different loss terms and then tries to minimize that. This approach often becomes problematic for solving stiff equations since it cannot consider adaptive increments. Many studies reported the poor performance of the PINN and its challenges in simulating stiff chemical active issues with administering conditions of stiff ordinary differential conditions (ODEs). Studies show that stiffness is the primary cause of the failure of the PINN in simulating stiff kinetic systems.

Here, we address this issue by proposing a reduced weak-form of the loss function, which led to a new PINN architecture, further named as Reduced-PINN, that utilizes a reduced-order integration method to enable the PINN to solve stiff chemical kinetics. The proposed Reduced-PINN can be applied to various reaction-diffusion systems involving stiff dynamics. To this end, we transform initial value problems (IVPs) to their equivalent integral forms and solve the resulting integral equations using physics-informed neural networks. In our derived integral-based optimization process, there is only one term without explicitly incorporating loss terms associated with ordinary differential equation (ODE) and initial conditions (ICs). To illustrate the capabilities of Reduced-PINN, we used it to simulate multiple stiff/mild second-order ODEs. We show that Reduced-PINN captures the solution accurately for a stiff scalar ODE. We also validated the Reduced-PINN against a stiff system of linear ODEs. In the last step, we used the Reduced-PINN to simulate the ROBER problem, a significant challenge made by a system of nonlinear ODEs. Our numerical validations show the capability of Reduced-PINN to represent stiff ODEs at a significantly reduced computational cost.

Keywords: Physics-informed neural network; Initial value problem; Stiff ordinary differential equation; Integral equation

1 Introduction

Ordinary and partial differential equations (ODEs and PDEs) are main drivers of the simulation cost in many large-scale simulations, ranging from physical events, chemical reactions, and economic models. In order to solve them, there are a variety of classical methods, e.g., finite element and finite difference methods [1]. These numerical procedures depend on the discretization of the spatiotemporal domain and then solve the resulting linear or nonlinear algebraic equations. As a result, the accuracy of classical numerical approaches relies directly on the element size or the grid size of the discrete version of governing equations. Physics-informed neural networks (PINNs) proposed by Raissi et al. [2] is an alternative for solving ODEs or PDEs. This method overcomes the need to discretize the governing equations. Instead, we need to find the parameters (weights and biases) of a deep neural network through training process. Because of the universal approximation theorem, a deep neural network can approximate a wide range of functions with an arbitrary precision.

PINNs have been successfully applied to many problems from numerous disciplines including solid mechanics [3, 4, 5, 6, 7] and fluid mechanics [8, 9]. In addition, other versions of physics-informed neural networks such as conservative PINNs [10] and variational PINNs [11, 12] allow more flexibility in comparison with regular PINNs. The total loss function of PINNs involves some terms related to differential equations, initial conditions, and boundary conditions. ODEs are widely used to describe the evolution of the species concentrations in chemical kinetics. Predicting stiff chemical systems is fundamental in modeling most real-time chemical reactions ranging from energy storage to material aging and biomedical design. A kinetic equation is called stiff when some involved species evolve slowly while others rapidly change in a short time frame. This forces the numerical method to take small steps over a long range to obtain satisfactory results. Thus, stiffness highly affects the computational efficacy of the solution.

It is known that regular PINN fails to represent stiff chemical systems [13] due to the numerical stiffness of the involved ODEs. Weiqi et al. [13] solved stiff chemical ODEs by imposing steady-state assumptions on some state variables. As a result, they found the solution to a non-stiff/mild system of ODEs instead of the original stiff ODEs. In a recent study, De Florio et al. [14] solved the stiff system of nonlinear ODEs without transformation to non-stiff/mild equations. However, their approach is based on a shallow NN and requires solving a nonlinear system of equations for a general system of ODEs.

To enable PINN for stiff systems, Wang et al. [15] proposed a learning rate annealing procedure to balance the interaction among loss function terms. The weights of terms in the total loss function can be adaptively modified after each epoch or a few epochs.

In this paper, we develop Reduced-PINN, the first weak-form of PINN framework developed for simulating the stiff ODEs based on derivation of the weak form of the involved ODEs. In particular, our main objectives in this paper are to derive

1. Weak-form formulation: we transform a regular PINN for a scalar ODE with multiple loss terms to an integral-based PINN with a single loss term. To this end, an initial value problem is converted into an integral equation and solved by physics-informed neural networks. The suggested method ensures that we are treating the same problem with only one loss term.

2. Long-time simulation of stiff systems: we will apply Reduced-PINN sequentially for long-term simulations. If we divide a long time interval into smaller time steps, it provides us with two main advantages: one is faster training for each Reduced-PINN and the other is to approximate the involved integrals with explicit formulas.

In this paper, we developed and validated a new engine, Reduced-PINN, to simulate stiff systems using weak-form of loss function. The transformation process to convert initial value problems (IVPs) into weak-form integral equations is given in section 2. We discuss classical PINN architecture and how to implement Reduced-PINN in sections 3 and 4. To validate our engine, the performance of Reduced-PINN is studied in multiple mild/stiff problems in section 5. Finally, conclusions and the outlook for future work will be presented in section 6.

2 PROBLEM SETUP

Stiff Chemical Systems Multiple techniques were developed to derive the numerical solution of the above ODE, such as implicit Runge-Kutta time-stepping schemes [16], although they are usually constrained due to the computational cost of solving Eq. (1). In chemical kinetics, most ODE systems that represent the kinetic models are stiff, and thus require ultra-fine time-steps for simulation which makes time-domain integration computationally expensive. Regardless of the nature of reactions, it is often difficult to provide a generalized description for the stiffness of a chemical system, and usually we identify stiffness of a system by the distinct time scales of their species. Fast-reacting species operate on short time scale, whereas others may react slower and have orders of magnitude more extended periods. Accordingly, one can consider an ODE system to be stiff if the computational cost of classical non-stiff ODE solver becomes much higher than implicit ODE integrators.

ODE of Stiff Systems A first-order non-homogeneous chemical reaction system can be modeled using the following ODE

$$\boldsymbol{u}^{(1)}(x) + \mathbf{A}(x)\boldsymbol{u}(x) = \boldsymbol{f}(x), \qquad \forall x \in [a, b]$$

$$\boldsymbol{u}(x) = [u_1(x), ..., u_N(x)]^T, \qquad u_i(a) = \mu_i^{<0>}, \qquad \boldsymbol{u}^{(n)}(x) = \frac{d^n \boldsymbol{u}}{dx^n}.$$
(1)

where the superscript $\bullet^{< j>}$ represents the parameter index, $\bullet^{(j)}$ the order of differentiation, and the subscript \bullet_j represents parameters associated to j-th species. Hereafter, we use the following formatting; \mathbf{A} to represent tensors, \boldsymbol{a} to represent vectors, and a to represent scalar parameters.

Weak-Form Derivation Here, we elaborate the process of converting a general ODE problem into an integral equation. The procedure is derived for each individual species $u_i(x)$. For simplicity, here we assume $\boldsymbol{u}(x)$ to be a single component vector which represents the concentration of one specie at the reaction time $x \in [a, b]$ with initial value to be $\mu^{<0>} := \mu_1^{<0>}$. However, the formulation can be further generalized to note that for multi-component vector $\boldsymbol{u}(x)$, the weak-form derivation will be implemented for each component $u_i(x)$. Let

us assume a kinetic reaction with the following ODE

$$u^{(n)}(x) + \lambda_1(x)u^{(n-1)}(x) + \dots + \lambda_n(x)u(x) = f(x)$$
(2)

with initial conditions

$$u(a) = \mu^{<0>}, u^{(1)}(a) = \mu^{<1>}, \dots, u^{(n-1)}(a) = \mu^{<(n-1)>}$$
 (3)

where $\lambda_i(x)(i=1,\ldots,n)$ and f(x) are continuous and given for $a \leq x \leq b$. Also, the n in Eq. (2) represents the order of ordinary differential equation. We need the following identity to convert the above IVP into an integral equation

$$\underbrace{\int_{a}^{x} \int_{a}^{x_{1}} \cdots \int_{a}^{x_{m-1}} v(x_{m}) dx_{m} dx_{m-1} \dots dx_{1}}_{m \text{ times}} = \int_{a}^{x} \frac{(x-t)^{m-1}}{(m-1)!} v(t) dt$$
(4)

To prove this identity, we adopt the approach given in [17]. Let $I_m v(x)$ be defined as

$$I_m v(x) = \int_a^x \frac{(x-t)^{m-1}}{(m-1)!} v(t) dt$$
 (5)

By using the Leibniz rule and differentiating both sides of Eq. (5), we have

$$\frac{d}{dx}(I_m v(x)) = \frac{(x-x)^{m-1}}{(m-1)!}v(x) + \int_a^x \frac{(x-t)^{m-2}}{(m-2)!}v(t)dt = I_{m-1}v(x)$$
(6)

In addition, we have $I_m v(a) = 0$ for any $m \ge 1$. If we integrate both sides of Eq. (6), we arrive at

$$I_{m}v(x) = I_{m}v(a) + \int_{a}^{x} I_{m-1}v(\eta)d\eta = \int_{a}^{x} I_{m-1}v(\eta)d\eta$$
 (7)

If we use the above equation successively, we derive the identity presented in Eq. (4). Now, let us define $u^{(n)}(x) = v(x)$, by integrating this expression one time, we obtain

$$u^{(n-1)}(x) = \int_{a}^{x} v(t)dt + u^{(n-1)}(a) = \int_{a}^{x} v(t)dt + \mu^{(n-1)}$$
 (8)

Integrating one more time, we have

$$u^{(n-2)}(x) = \int_{a}^{x} \int_{a}^{x_1} v(x_2) dx_2 dx_1 + \mu^{(n-1)}(x-a) + \mu^{(n-2)}$$
(9)

Continuing this process, the following general expression for (k = 0, ..., n - 1) is derived

$$u^{(k)}(x) = \underbrace{\int_{a}^{x} \int_{a}^{x_{1}} \cdots \int_{a}^{x_{n-k-1}} v(x_{n-k}) dx_{n-k} dx_{n-k-1} \dots dx_{1}}_{n-k \text{ times}} + \sum_{i=0}^{n-k-1} \mu^{< n-i-1 > \frac{(x-a)^{n-k-i-1}}{(n-k-i-1)!}}$$

$$(10)$$

where we define $u^{(0)}(x) = u(x)$. The first expression of Eq. (10) can be simplified further by the identity given in Eq. (4). As a result, we have

$$u^{(k)}(x) = \int_{a}^{x} \frac{(x-t)^{n-k-1}}{(n-k-1)!} v(t)dt + \sum_{i=0}^{n-k-1} \mu^{(n-i-1)} \frac{(x-a)^{n-k-i-1}}{(n-k-i-1)!}$$
(11)

The above relationship holds for an arbitrary derivative of u(x). In a compact form, Eq. (2) can be written as

$$u^{(n)}(x) + \sum_{j=0}^{n-1} \lambda_{n-j}(x)u^{(j)}(x) = f(x)$$
(12)

If we substitute Eq. (11) into the above expression for each derivative of u(x) and use $u^{(n)}(x) = v(x)$, we finally derive the following integral equation

$$v(x) + \int_{a}^{x} \psi(x,t)v(t)dt = g(x)$$
(13)

where

$$\psi(x,t) = \sum_{j=0}^{n-1} \lambda_{n-j}(x) \frac{(x-t)^{n-j-1}}{(n-j-1)!}$$
(14)

and

$$g(x) = f(x) - \sum_{j=0}^{n-1} \sum_{i=0}^{n-j-1} \mu^{\langle n-i-1 \rangle} \lambda_{n-j}(x) \frac{(x-a)^{n-j-i-1}}{(n-j-i-1)!}$$
 (15)

The residual associated with the integral equation is written as

$$R(x) = v(x) + \int_{a}^{x} \psi(x,t)v(t)dt - g(x)$$

$$\tag{16}$$

The interested reader can refer to [17] for a detailed discussion on integral equations.

3 Classical PINN Architecture

The success of physics-informed neural networks is essentially based on the ability of deep neural networks to approximate a vast number of functions appearing in practice. In an initial value problem, a neural network can be set up to represent u(x) with x as input, and u as the output as

$$\boldsymbol{u}(x) = l(\Phi^{< h>}(\Phi^{< h-1>} \dots (\Phi^{< 0>}(x)))), \qquad \Phi^{< i>}(\hat{x}) = \sigma(W^{< i>} \cdot \hat{x} + b^{< i>})$$
 (17)

where $\theta = [[W^i], [b^i]]$ is the set of weights W^i and biases b^i , σ a nonlinear activation function, and l a linear amplification function. Note that $u = [u_1, ..., u_N]^T$ represents the vector of species concentrations, and N is the number of acting chemical species in the ODE. Fig. (1) shows a schematic representation of a fully connected feed-forward neural network with two hidden layers and arbitrary units in each layer. As can be seen from this figure, there is only one output to this neural network because we have used separate neural networks

corresponding to each state variable, and that is only our preference in this study. It should be noted that any type of feed-forward neural network including shallow networks can be used for implementation.

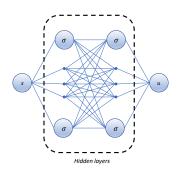


Figure 1: Schematic representation of a deep neural network trained with Reduced-PINN method.

In comparison to recent studies, the proposed Reduced-PINN has two main advantages over Stiff-PINN methods [13, 14]; (1) Our method uses a deep NN as well as a shallow NN, and be trained by back-propagation for any nonlinear system of ODEs. (2) Our method solves the original equations without any transformation to non-stiff/mild ones.

Loss function of classical PINN is derived by implementing automatic differentiation [18] as additional network to NN representing $\boldsymbol{u}(x)$. Accordingly, by substituting derivatives of $\boldsymbol{u}(x)$ in the ODE by automatic differentiated NN, one derives the ODE residual term $\mathcal{L}_{ODE}(\theta)$, and initial condition residual form $\mathcal{L}_{IC_i}(\theta)$ as given below and furthers calculates the total loss function $\mathcal{L}(\theta)$ of a PINN as

$$\mathcal{L}(\theta) = \mathcal{L}_{ODE}(\theta) + \sum_{i} \alpha_{i} \mathcal{L}_{IC_{i}}(\theta)$$
(18)

where α_i are fixed or adaptively tuned parameters. Now, by minimizing the total loss with respect to the weights and biases, PINN solution is derived.

4 Reduced-PINN Implementation

Loss function of proposed Reduced-PINN In the proposed integral-based PINN, our total loss function $\mathcal{L}(\theta)$ involves only one term since we have coupled the ODE and IVP equations into one integral equation. The residual defined in Eq. (16) becomes a function of θ and x after approximating v(x) by a deep neural network with the parameters θ . In this study, we consider the following loss function

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} R(x_i; \theta)^2$$
(19)

where $R(x;\theta)$ is the residual of the integral equation defined in Eq. (16), and n denotes the number of collocation points in time domain $x \in [a, b]$. In this paper, for each state variable, we have considered a separate neural network. Therefore, for a system of ODEs containing

M equations, we have the following total loss

$$\mathcal{L}(\Theta) = \sum_{i=1}^{M} \mathcal{L}(\theta_i) \tag{20}$$

where Θ stands for the neural network parameters of all state variables, and $\mathcal{L}(\theta_i)$ is the loss corresponding to any scalar ODE of our system of ODEs.

Numerical Integration Using TensorFlow [19] for numerical implementation, the integrals are calculated based on the Newton–Cotes rules with the following general form

$$\int_{a}^{b} h(x)dx \approx \sum_{i=0}^{p} \omega_{i} h(x_{i})$$
(21)

where ω_i are weights, and x_i are the equi-spaced timesteps within the time domain [a, b]. Often, a for loop is utilized for a general integration procedure, but this method for a Reduced-PINN takes a long time in terms of computational cost. A feasible approach to speed up the process of integration is to combine the Newton-Cotes rules (e.g., the trapezoidal rule) with the current capabilities of TensorFlow. We have used this technique in our numerical integration wherever possible.

Using trapezoidal rule, numerical integration in Eq. (21) is given as

$$\int_{a}^{b} h(x)dx \approx \left(\frac{\Delta x}{2}\right)(h(x_0) + 2\sum_{i=1}^{p-1} h(x_i) + h(x_p))$$
(22)

where $\Delta x = \left(\frac{b-a}{p}\right)$, and $(x_i)_{i=0}^p$ is assumed to be a uniform partition of the domain. Now, for the summation in the middle of the above expression, we could use the reduce mean command from TensorFlow to evaluate the integral much faster than the usual for loop. In addition, for the sequential implementation of Reduced-PINN, we also have used the following Simpson's rule

$$\int_{a}^{b} h(x)dx \approx \left(\frac{b-a}{6}\right)(h(a) + 4h(\frac{a+b}{4}) + h(b)) \tag{23}$$

We could implement the above formula into our code directly, which gives an accurate approximation to the exact value of the integral when the difference between the end points of the interval of integration (i.e., b-a) is relatively small.

System of ODEs For completeness, we consider a general system of nonlinear ODEs as follows

$$\mathbf{u}^{(1)}(x) = \mathbf{q}(\mathbf{u}(x), x), \quad \Rightarrow \quad \begin{cases} u_1^{(1)}(x) = q_1(u_1, u_2, \dots, u_N, x) \\ \vdots \\ u_N^{(1)}(x) = q_N(u_1, u_2, \dots, u_N, x) \end{cases}$$
(24)

with initial conditions $u_i(a) = \mu_i^{<0>}$ for i = 1, 2, ..., N. We could use the following relations to transform the above equations into an appropriate framework for Reduced-PINN

implementation

$$u_i^{(1)}(x) = v_i(x), i = 1, 2, \dots, N.$$
 (25)

$$u_i(x) = \int_a^x v_i(t)dt + \mu_i^{<0>}, i = 1, 2, \dots, N.$$
 (26)

Next, we approximate each unknown function $v_i(x)$ by a separate neural network, all with the same initial random values. After training, we will recover the Reduced-PINN solution to our IVP from Eq. (26). Our numerical experiments show that, for practical cases, we should use Reduced-PINN sequentially. In other words, as shown in Fig. (2), the total time interval is divided into m+1 subintervals and the solution of the current Reduced-PINN at the end of its corresponding subinterval is given to the next Reduced-PINN as the initial condition. For the first Reduced-PINN, the initial conditions will be the same as the ones given in the problem definition. For optimization procedure, Adam optimizer will be used as a classical gradient-descent algorithm.

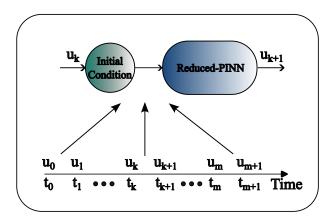


Figure 2: Schematic representation of Sequential Reduced-PINN

5 Numerical Results

In this section, the performance of the proposed Reduced-PINN in different cases of mild and stiff ODEs will be benchmarked against solutions. In the illustrated examples, classical PINN mostly fails due to the stiffness, and the performance of the Reduced-PINN in reducing stiffness and predicting the species concentrations will be provided. The first example is a mild ODE of second-order where we introduce and use sequential Reduced-PINN to solve it. The second case is a stiff scalar ODE of the first order. Next, we consider a stiff system of linear ODEs. Finally, we solve the ROBER problem. For training, an unknown function from integral equations is represented by a deep neural network. For all cases, our neural network has the following hyper-parameters: 4 hidden layers, 50 units per layer, ReLU activation function, and Adam optimizer. Afterward, we obtain the solution to our IVP by related integral formulas. All examples addressed here was coded in Python and ran on Apple M1 chip, and 16 GB of RAM.

5.1 Case Study 1: Mild ODE

We first employed Reduced-PINN to solve the following second order initial value problem as a mild ODE

$$u^{(2)}(x) - 2cu^{(1)}(x) + (c^2 + d^2)u(x) = 0, (0 \le x \le 0.4)$$
(27)

$$u(0) = \mu^{<0>}, u^{(1)}(0) = \mu^{<1>}.$$
 (28)

where c and d are some chosen values and $\mu^{<0>}$ and $\mu^{<1>}$ are arbitrary initial conditions for use in the sequential Reduced-PINNs, i.e., the solution of current Reduced-PINN is transferred to the next Reduced-PINN as initial conditions. The exact solution of the above differential equation is written as

$$u_{exact}(x) = \left(\frac{\mu^{<1>} - c\mu^{<0>}}{d}\right) e^{cx} \sin dx + \mu^{<0>} e^{cx} \cos dx$$
 (29)

and its equivalent integral form is written as

$$v(x) + \int_0^x [(c^2 + d^2)(x - t) - 2c]v(t)dt + \mu^{<1>}[(c^2 + d^2)x - 2c] + \mu^{<0>}(c^2 + d^2) = 0 \quad (30)$$

after training, the Reduced-PINN solution to our problem is derived from

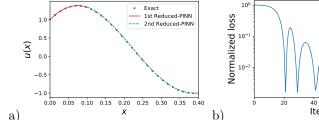
$$u(x) = \int_0^x (x - t)v(t)dt + \mu^{<1>} x + \mu^{<0>}$$
(31)

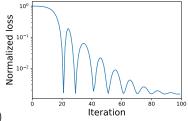
As an example, we choose c=-1, d=10, $\mu^{<0>}=1$, and $\mu^{<1>}=10$ and consider using two sequential Reduced-PINNs. The first Reduced-PINN was implemented on the interval [0,0.1] and its predictions were given to the second Reduced-PINN, which is implemented on [0,0.3]. In Fig. (3a), we benchmarked Reduced-PINN predictions against the exact solution, which shows an excellent agreement. The history of normalized loss value functions in the first and second Reduced-PINNs is also demonstrated in Fig.3b and Fig. 3c, respectively. To get an accurate prediction by sequential Reduced-PINNs, one has to ensure that each Reduced-PINN is optimised to yield its best solution since the solutions are built upon each other.

For this problem, we have used two sequential Reduced-PINNs as we can capture the solution with only two Reduced-PINNs. However, we could use many more Reduced-PINNs if we have a large time span. To elaborate more, initially, we divide the whole time span into smaller time subintervals. In the first time subinterval, we solved the ODE by using the initial conditions given by the problem. Thus, we approximate the main variables at the end of the first subinterval, which become the new initial conditions for the next Reduced-PINN solver, and so on.

5.2 Case Study 2: Stiff Single ODE

The complex behavior in stiff ODE is induced by the presence of steep gradients which often forms a significant computational barrier in computational efforts. Stiff ODEs can be mainly identified by different time-scales of the associated species. In long-time domain problems, a





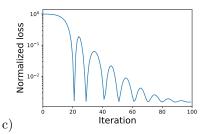


Figure 3: Case Study 1: Mild ODE example. a) Exact solution vs. Reduced-PINN predictions, b) Convergence history of the first Reduced-PINN, c) Convergence history of the second Reduced-PINN.

decomposition of the temporal domain into sub-intervals can minimize the carryover of the error during the chemical reactions between the species. Accordingly, the following initial value problem is used to illustrate capabilities of Reduced-PINN

$$u^{(1)}(x) - \lambda u(x) = e^{-x}, (0 \le x \le 0.05)$$
(32)

$$u(0) = \mu^{<0>} \tag{33}$$

The exact solution of the above IVP is written as

$$u_{exact}(x) = \left(\mu^{<0>} + \frac{1}{1+\lambda}\right)e^{\lambda x} - \frac{e^{-x}}{1+\lambda} \tag{34}$$

and its integral form is

$$v(x) - \lambda \int_0^x v(t)dt = e^{-x} + \lambda \mu^{<0>}$$
 (35)

also, after training and obtaining v(x), the Reduced-PINN solution to our IVP can be derived from

$$u(x) = \int_0^x v(t)dt + \mu^{<0>}$$
 (36)

If we take λ to be a large value in magnitude, then the exact solution includes both fast and slow components. To address the issue, we set $\lambda = -50$ and $\mu^{<0>} = 2$. Our method works quite well for this stiff scalar ODE and numerical results agree with the exact solution as shown in Fig.4a. We also provide the training history of our method in Fig.4b.

5.3 Case Study 3: Stiff ODE System

Next, we benchmarked Reduced-PINN against the following system of stiff linear ODEs that were used to benchmark stiff-ODE solvers by [20]

$$u_1^{(1)}(x) = \left(\frac{\lambda_1 + \lambda_2}{2}\right) u_1(x) + \left(\frac{\lambda_1 - \lambda_2}{2}\right) u_2(x)$$

$$u_2^{(1)}(x) = \left(\frac{\lambda_1 - \lambda_2}{2}\right) u_1(x) + \left(\frac{\lambda_1 + \lambda_2}{2}\right) u_2(x)$$
(37)

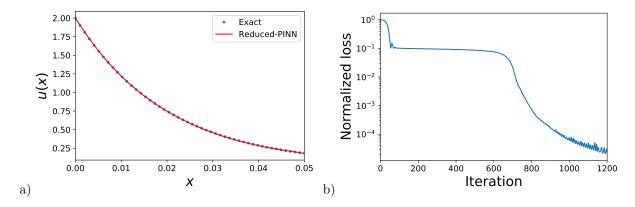


Figure 4: Case Study 2: Stiff Single ODE example. a) Exact solution vs. Reduced-PINN predictions, b) Convergence history

with initial conditions $u_1(0) = \mu_1^{<0>}$ and $u_2(0) = \mu_2^{<0>}$. The solution of the given system of ODEs is written as

$$u_1(x) = \left(\frac{\mu_1^{<0>} + \mu_2^{<0>}}{2}\right)e^{\lambda_1 x} + \left(\frac{\mu_1^{<0>} - \mu_2^{<0>}}{2}\right)e^{\lambda_2 x}$$
(38)

$$u_2(x) = \left(\frac{\mu_1^{<0>} + \mu_2^{<0>}}{2}\right)e^{\lambda_1 x} - \left(\frac{\mu_1^{<0>} - \mu_2^{<0>}}{2}\right)e^{\lambda_2 x}$$
(39)

To put the equations into integral form, let $u_1^{(1)}(x) = v_1(x)$ and $u_2^{(1)}(x) = v_2(x)$, then we have

$$u_1(x) = \int_0^x v_1(t)dt + \mu_1^{<0>} \tag{40}$$

$$u_2(x) = \int_0^x v_2(t)dt + \mu_2^{<0>} \tag{41}$$

Substituting the above expressions into our system of equations yields

$$v_1(x) - (\frac{\lambda_1 + \lambda_2}{2}) \left(\int_0^x v_1(t)dt + \mu_1^{<0>} \right) - \left(\frac{\lambda_1 - \lambda_2}{2} \right) \left(\int_0^x v_2(t)dt + \mu_2^{<0>} \right) = 0$$
 (42)

$$v_2(x) - \left(\frac{\lambda_1 - \lambda_2}{2}\right) \left(\int_0^x v_1(t)dt + \mu_1^{<0>}\right) - \left(\frac{\lambda_1 + \lambda_2}{2}\right) \left(\int_0^x v_2(t)dt + \mu_2^{<0>}\right) = 0$$
 (43)

We have considered the following parameters for the given system: $\lambda_1 = -20$, $\lambda_2 = -2$, $\mu_1^{<0>} = 2$, $\mu_2^{<0>} = 0$ and $(0 \le x \le 1.0)$. To solve this problem, we have used five sequential Reduced-PINNs with identical parameters. The time interval considered for each of them has the same length.

Since there are two unknown functions, we have chosen a deep neural network for each function. The left hand sides of Eqs. (42) and (43) are the residuals associated with the first and second ODEs, respectively. Therefore, the total loss function consists of two parts. Fig.5 shows numerical results vs. exact solutions for $u_1(x)$ and $u_2(x)$. As can be seen from this figure, we have an excellent agreement between Reduced-PINN and exact solution for this stiff system of ODEs.

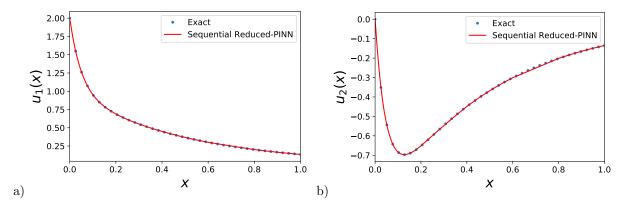


Figure 5: Case Study 3: Stiff ODE System (a) Exact solution vs. the proposed method for the first state variable (b) Exact solution vs. the proposed method for the second state variable

5.4 Case Study 4: ROBER Problem

Our last example is Robertson's system of equations [21] referred to as the ROBER problem first by Wanner and Hairer [22] It describes the following auto-catalytic reaction kinetics

$$A \underset{k_1}{\longrightarrow} B$$

$$B + B \underset{k_2}{\longrightarrow} C + B$$

$$B + C \underset{k_2}{\longrightarrow} A + C,$$

$$(44)$$

where A, B, and C are the chemical species concentrations, and k_1, k_2 , and k_3 the reaction rate constants. The ROBER problem can be written using three nonlinear ODEs as follows

$$u_1^{(1)}(x) = -k_1 u_1 + k_3 u_2 u_3 u_1(0) = 1$$

$$u_2^{(1)}(x) = k_1 u_1 - k_2 u_2^2 - k_3 u_2 u_3 u_2(0) = 0$$

$$u_3^{(1)}(x) = k_2 u_2^2 u_3(0) = 0. (45)$$

Concentrations of species in the above equation are represented by $u_i(x)$ and the reaction constants are $k_1 = 0.04$, $k_2 = 3 \times 10^7$, and $k_3 = 10^4$ which cover a time-span in the range of 9 orders. Such a wide gap in time-span of the species leads to the strong stiffness of the problem. Here, we have considered 250 sequential Reduced-PINNs with identical parameters to solve the problem for the time duration $[10^{-5}, 10^{-1}]$. Also, backward differentiation formula (BDF) is utilized as an implicit integrator for the purpose of comparison. The results obtained from Reduced-PINN for species concentrations are plotted in Fig.6. As shown in this figure, we have an excellent agreement between Reduced-PINN and BDF method results. Our results show that the Reduced-PINN method can be reliably used to solve the ROBER problem, which classical PINN fails to do, even if artifacts to reduce the stiffness of the ODE system are hired, as in Stiff-PINNs [13].

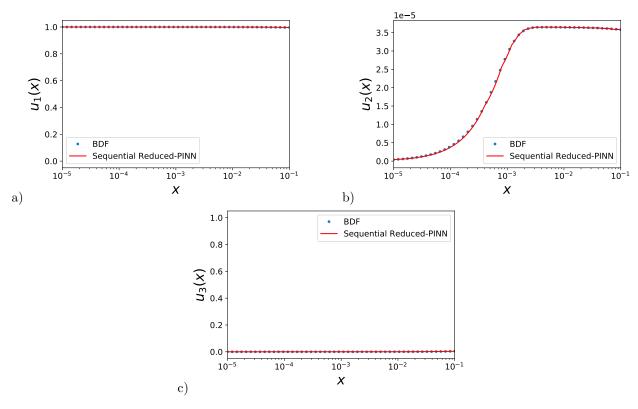


Figure 6: Case Study 4: ROBER Problem; Comparison between the BDF method and sequential Reduced-PINN for ROBER problem (a) Evolution of the first species concentration (b) Evolution of the second species concentration (c) Evolution of the third species concentration

6 Outlook

Future works will focus on developing a robust and general Reduced-PINN framework for stiff chemical kinetic simulations that can handle multiple volatile species with extremely short time scales. Our goal is to demonstrate the relevance of the Reduced-PINN framework for local residuals (e.g., chemically reacting flows) and non-local residuals (e.g., PDEs with spatial derivatives) compared to PINNs and traditional DNNs.

7 Conclusions

We proposed a new neural network engine based on PINN for solving stiff initial value problems using integral-based physics-informed neural networks. This study is motivated by simply using the residual of the PDE as in Physics-Informed Neural Networks (PINNs) is not optimal for the simulation of multi-species chemical reactions. Our engine, referred to as Reduced-PINN, was developed based on the concept of converting ODEs into a weakform integral equation. This approach ensures we combine loss functions and boundary conditions into one loss equation. Next, we show how to build the total loss function for a general system of ODEs. We benchmarked Reduced-PINN prediction against four systems of varying complexity to validate its capabilities. Using Reduced-PINN sequentially is more suitable for solving initial value problems for long-term problems. Next, we used Reduced-PINNs to represent stiff ODE systems, for which the regular-PINN failed to predict the evolution of the systems. By imposing reduced weak form integral on loss function of involved species in the kinetic systems and reducing the stiffness, the Stiff-PINN well captured the dynamic responses of the reference ODE systems. Our numerical benchmarking shows the Reduced-PINN efficacy and accuracy in solving stiff ODE systems at a much lower cost.

References

- [1] A. Iserles, A first course in the numerical analysis of differential equations. No. 44, Cambridge university press, 2009.
- [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] E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, "A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 379, p. 113741, 2021.
- [4] M. Vahab, E. Haghighat, M. Khaleghi, and N. Khalili, "A physics-informed neural network approach to solution and identification of biharmonic equations of elasticity," *Journal of Engineering Mechanics*, vol. 148, no. 2, p. 04021154, 2022.
- [5] S. A. Niaki, E. Haghighat, T. Campbell, A. Poursartip, and R. Vaziri, "Physics-informed neural network for modelling the thermochemical curing process of composite-tool systems during manufacture," Computer Methods in Applied Mechanics and Engineering, vol. 384, p. 113959, 2021.

- [6] E. Samaniego, C. Anitescu, S. Goswami, V. M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, and T. Rabczuk, "An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications," Computer Methods in Applied Mechanics and Engineering, vol. 362, p. 112790, 2020.
- [7] R. Arora, P. Kakkar, B. Dey, and A. Chakraborty, "Physics-informed neural networks for modeling rate-and temperature-dependent plasticity," arXiv preprint arXiv:2201.08363, 2022.
- [8] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," Computer Methods in Applied Mechanics and Engineering, vol. 360, p. 112789, 2020.
- [9] M. M. Almajid and M. O. Abu-Al-Saud, "Prediction of porous media fluid flow using physics informed neural networks," *Journal of Petroleum Science and Engineering*, vol. 208, p. 109205, 2022.
- [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," Computer Methods in Applied Mechanics and Engineering, vol. 365, p. 113028, 2020.
- [11] E. Kharazmi, Z. Zhang, and G. E. Karniadakis, "Variational physics-informed neural networks for solving partial differential equations," arXiv preprint arXiv:1912.00873, 2019.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] J. D. Lambert, Numerical methods for ordinary differential systems: the initial value problem. John Wiley & Sons, Inc., 1991.
- [17] L. G. Chambers, *Integral equations: a short course*. International Textbook Company, 1976.
- [18] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: a survey," *Journal of Marchine Learning Research*, vol. 18, pp. 1–43, 2018.
- [19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat,

- G. Irving, M. Isard, et al., "{TensorFlow}: A system for {Large-Scale} machine learning," in 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp. 265–283, 2016.
- [20] R. Bulirsch, J. Stoer, and J. Stoer, *Introduction to numerical analysis*, vol. 3. Springer, 2002.
- [21] H. Robertson, "The solution of a set of reaction rate equations," *Numerical analysis:* an introduction, vol. 178182, 1966.
- [22] G. Wanner and E. Hairer, Solving ordinary differential equations II, vol. 375. Springer Berlin Heidelberg New York, 1996.