scientific reports

OPEN

Physics-informed neural networks for predicting gas flow dynamics and unknown parameters in diesel engines

Kamaljyoti Nath[1,5], Xuhui Meng[2,5], Daniel J. Smith[3] & George Em Karniadakis[1,4]✉

This paper presents a physics-informed neural network (PINN) approach for monitoring the health of diesel engines. The aim is to evaluate the engine dynamics, identify unknown parameters in a "mean value" model, and anticipate maintenance requirements. The PINN model is applied to diesel engines with a variable-geometry turbocharger and exhaust gas recirculation, using measurement data of selected state variables. The results demonstrate the ability of the PINN model to predict simultaneously both unknown parameters and dynamics accurately with both clean and noisy data, and the importance of the self-adaptive weight in the loss function for faster convergence. The input data for these simulations are derived from actual engine running conditions, while the outputs are simulated data, making this a practical case study of PINN's ability to predict real-world dynamical systems. The mean value model of the diesel engine incorporates empirical formulae to represent certain states, but these formulae may not be generalizable to other engines. To address this, the study considers the use of deep neural networks (DNNs) in addition to the PINN model. The DNNs are trained using laboratory test data and are used to model the engine-specific empirical formulae in the mean value model, allowing for a more flexible and adaptive representation of the engine's states. In other words, the mean value model uses both the PINN model and the DNNs to represent the engine's states, with the PINN providing a physics-based understanding of the engine's overall dynamics and the DNNs offering a more engine-specific and adaptive representation of the empirical formulae. By combining these two approaches, the study aims to offer a comprehensive and versatile approach to monitoring the health and performance of diesel engines.

Powertrains of the future must meet increasingly stringent requirements for emissions, performance, reliability, onboard monitoring, and serviceability. Capable system models for estimating states and adapting to an individual system's behaviour are critical elements to meet control and health monitoring needs. Leveraging purely data-driven models to meet these requirements provides simplicity in modelling and captures dynamics difficult to formulate analytically. However, large data needs, poor physical interpretability, challenges with systems with long memory effects and sparse sensing, as well as inability to extrapolate beyond the training datasets present onerous burdens to practical implementation. Relying on purely theory-based models allows for directly interpretable results with higher confidence and fewer data for calibration but often causes a tradeoff of modelling relevant dynamics versus model complexity, challenges in systems with high uncertainties, poor modelling where dynamics are not well understood, and slow solution of higher-order models. Modelling solutions that leverage the strengths of theory-guided as well as data-driven models have the potential to reduce data needs, increase robustness, and effectively use theoretical and practical knowledge of the system.

To investigate model architectures, balancing the strengths of both theory-based models and data-driven models, this work explores the application of physics-informed neural networks (PINNs) to a diesel internal combustion engine model for the purposes of simultaneous parameter and state estimation. The physical portion

[1]Division of Applied Mathematics, Brown University, Providence, RI, USA. [2]Institute of Interdisciplinary Research for Mathematics and Applied Science, School of Mathematics and Statistics, Huazhong University of Science and Technology, Wuhan, China. [3]Cummins Inc., Columbus, IN, USA. [4]School of Engineering, Brown University, Providence, RI, USA. [5]These authors contributed equally: Kamaljyoti Nath and Xuhui Meng. ✉email: george_karniadakis@brown.edu

is based on the mean value model of a diesel engine with a variable geometry turbocharger (VGT), and exhaust gas recirculation (EGR) proposed by Wahlström and Eriksson[1].

Physics-informed neural networks (PINNs)[2] is a new method of training neural networks, which takes into account the physics of a problem while evaluating the parameters of the neural network. The method is suitable for both evaluation of the solution of PDF (forward problem) and the data-driven identification of parameters of PDF (inverse problem). It takes advantage of automatic differentiation[3] in formulating a physical loss in the loss function along with data loss. Jagtap et al.[4] proposed conservative PINNs (cPINNs) for conservation laws, which employs domain decomposition with a PINN formulation in each domain. Further, Jagtap and Karniadakis[5] introduced domain decomposition for general PDEs using the so-called extended PINN (XPINN). hp-VPINNs is a variational formulation of PINN with domain decomposition proposed by Kharazmi et al.[6]. Meng et al.[7] proposed the Parareal PINN (PPINN) approach for long-time integration of time-dependent partial differential equations. The authors of[8] proposed "separable" PINN, which can reduce the computational time and increase accuracy for high dimensional PDEs. In PINN, there are multiple loss functions, and the total loss function is given by the weighted sum of individual losses. McClenny and Braga-Neto[9] proposed a self-adaptive weight technique, which is capable of tuning the weights automatically. PINN and its variants were also considered in various inverse problems like supersonic flows[10], nano-optics and metamaterials[11], unsaturated groundwater flow[12]. Detailed reviews of PINN can be found in[13–15].

Modelling of diesel engines using neural networks has been considered in the past. Biao et al.[16] considered Nonlinear Auto-Regressive Moving Average with eXogenous inputs (NARMAX) method for system identification of locomotive diesel engines. The model has three inputs to the network, i.e. the fuel injected, the load of the main generator, and the feedback rotation speed (from the output); the outputs are rotation speed and diesel power. The authors considered Levenberg–Marquardt (LM) algorithm to train the network. Finesso and Spessa[17] developed a three-zone thermodynamic model to predict nitrogen oxide and in-cylinder temperature heat release rate for direct injection diesel engines under steady state and transient conditions. The model is zero-dimensional, and the equations can be solved analytically. Thus, it required a very short computational time. Tosun et al.[18] predicted torque, carbon monoxide, and oxides of nitrogen using neural networks (3 independent networks) for diesel engines fueled with biodiesel-alcohol mixtures. The authors considered three fuel properties (density, cetane number, lower heating value) and engine speed as input parameters and the networks are optimized using the Levenberg-Marquardt method. The authors observed that neural network results are better than the least square method. González et al.[19] integrated a data-driven model with a physics-based (equation-based) model for the gas exchange process of a diesel engine. The authors modelled the steady-state turbocharger using a neural network. Further, the authors integrated the data-driven model with an equation-based model. Recently, Kumar et al.[20] considered DeepONet[21] to predict the state variables of the same mean value engine model[1] we considered in this study. The authors consider dynamic data to train the model. However, the model predicts the state variables only at the particular (trained) ambient temperature and pressure, as variations of ambient temperature and pressure are not considered in the training of DeepONet. The model also does not predict the parameters of the engine model. While the model was trained using dynamic data, the physics of the problem was not considered while training the network. The model (DeepONet) is capable of predicting dynamic responses.

In the present study, we formulate a PINN model for the data-driven identification of parameters and prediction of dynamics of system variables of a diesel engine. In PINN, the physics of the system is directly included in the form of physics loss along with data loss. While data-driven models require large amount of data over the entire operational range in training, PINN can be trained with a smaller amount of data as it is trained online. The dynamics characteristic of the state variables is automatically incorporated. PINN may be used for the solution of differential equations or for the identification of parameters and prediction of state variables. In the present study, we are specifically interested in estimating unknown parameters and states when we know a few state variables from field data. The dynamics of the state variables of the mean value engine[1] are described by first-order differential equations. We will utilize these equations in the formulation of the physics-informed loss function. The unknown parameters are considered trainable and updated in the training process along with the neural network parameters.

The engine model also considers a few empirical formulae in its formulation. These equations are engine-specific, and the coefficients of these equations need to be evaluated from experimental data. These equations are static in nature, and thus may be trained with smaller data compared to dynamic equations. We know that deep neural networks (DNNs) are universal approximators of any continuous function, thus, DNNs may be considered more general approximators of these empirical formulae. One of the advantages of considering DNNs over empirical formulae is that we do not need to assume the type of non-linearity between the input out variables. The neural network learns the non-linearity if trained with sufficient data. We approximate the empirical formulae using DNNs and train them using laboratory test data. Once these networks are trained using laboratory test data, these are considered in the PINNs model in places of the empirical formulae. During the training of the PINNs model, the parameters of these networks are remain constant.

The training data for the inverse problem and laboratory data are generated using the Simulink file[22] accompanied in[1]. The input to Simulink is taken from actual field data. By doing this, we are trying to generate data as realistic as field data. Furthermore, we also consider noise to the field data generated. We observed that the proposed PINNs model can predict the dynamics of the states and unknown parameters. We summarize below a few of the salient features of the present study:

1. We formulated PINNs-based parameter identification for real-world dynamical systems, in the present case, a diesel engine. This is significant as it started a new paradigm for future research for onboard systems for the health monitoring of engines.

2. We showed how PINNs could be implemented in predicting important unknown parameters of diesel engines from field data. From these predicted parameters, one can infer the health and serviceability requirements of the engine.
3. We showed the importance of self-adaptive weights (given the fast transient dynamics) in the accuracy and faster convergence of results for PINNs for the present study.
4. The engine model generally considers empirical formulae to evaluate a few of its quantities. These empirical formulae are engine-specific and require lab test data for the evaluation of the coefficients. We have shown how neural networks can be considered to model the empirical formulae. We have shown how we can train these networks from lab-test data. This is important as it may provide a better relationship for the empirical formulae.
5. The field data for the inverse problem are generated considering input recorded from actual engine running conditions. Further, we consider appropriate noise in the simulated data, mimicking near real-world field data.

We organize the rest of the article as follows: in "Problem setup", we discuss the detailed problem statement and different cases considered for simulation studies. In "Methodology", first, we discuss PINNs for the inverse problems for the diesel engine and the surrogates for the empirical formula. We discuss a detailed flow chart for the inverse problem for the PINN engine model in "Flowchart for PINN model for diesel engine". In "Data generation", we discuss the laboratory data required and their generation for the training of surrogates for the empirical formulae. We also discuss the field data generation for the inverse problem. We present the results and discussion in "Results and discussions". The conclusions of the present study are discussed in "Summary".

Problem setup.

In this section, we first introduce the mean value model for the gas flow dynamics[1] in the diesel engine, and then we will formulate the inverse problems that we are interested in.

As shown in Fig. 1, the engine model considered in the present study mainly comprises six parts: the intake and exhaust manifold, the cylinder, the exhaust gas recirculation (EGR) valve system, the compressor and the turbine. More details on each engine part can be seen in "Appendix 2". We note that the engine considered here is the same as in[1].

To describe the gas flow dynamics in the engine illustrated in Fig. 1, e.g., the dynamics in the manifold pressures, turbocharger, EGR and VGT actuators, a mean value model of the diesel engine with variable geometric turbocharger and exhaust gas recirculation was proposed in[1]. We will also utilize the same model as the governing equations to describe the gas flow dynamics considered in the current study. Specifically, the model proposed in[1] has eight states expressed as follows:

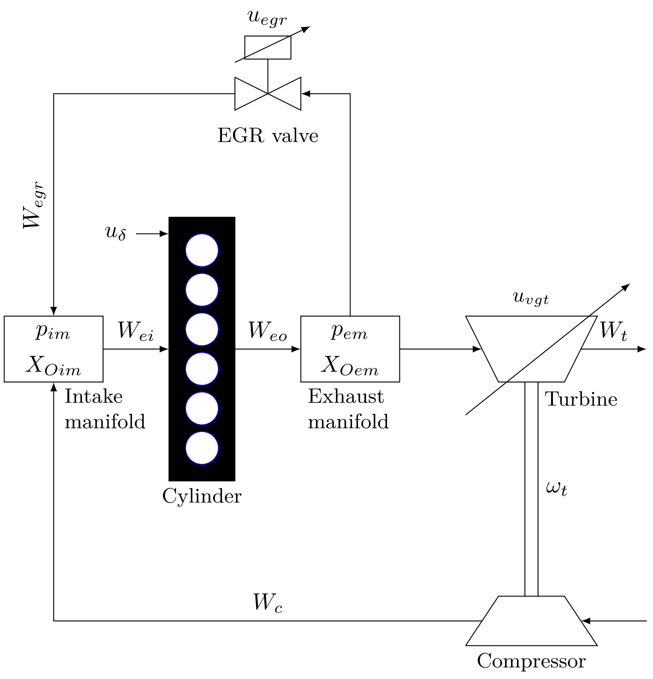


Figure 1. Schematic diagram of the diesel engine: a schematic diagram of the mean value diesel engine with a variable-geometry turbocharger (VGT) and exhaust gas recirculation (EGR)[1]. The main components of the engine are the intake manifold, the exhaust manifold, the cylinder, the EGR valve system, the compressor, and the turbine. The control input vector is $\boldsymbol{u} = \{u_\delta, u_{egr}, u_{vgt}\}$, and engine speed is n_e. (Source: Figure is adopted from[1]).

$$\boldsymbol{x} = \{p_{im},\ \ p_{em},\ \ X_{Oim},\ \ X_{Oem},\ \ \omega_t,\ \ \tilde{u}_{egr1},\ \ \tilde{u}_{egr2},\ \ \tilde{u}_{vgt}\}, \tag{1}$$

where p_{im} and p_{em} are the intake and exhaust manifold pressure, respectively, X_{Oim} and X_{Oem} are the oxygen mass fractions in the intake and exhaust manifold, respectively, ω_t is the turbo speed; $\tilde{u}_{vgt}$ represents the VGT actuator dynamics. A second-order system with an overshoot and a time delay is used to represent the dynamics of the EGR-valve actuator. The model is represented by subtraction of two first-order models, $\tilde{u}_{egr1}$ and $\tilde{u}_{egr2}$, with different gains and time constants. Further, the control inputs for the engine are $\boldsymbol{u} = \{u_\delta,\ \ u_{egr},\ \ u_{vgt}\}$ and the engine speed is n_e, in which u_δ is the mass of injected fuel, u_{egr} and u_{vgt} are the EGR valve position and VGT actuator positions, respectively. Furthermore, the position of the valves, i.e., u_{egr} and u_{vgt}, may vary from 0 to 100%, which indicates the complete close and opening of the valves, respectively. The mean value engine model is then expressed as

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, n_e). \tag{2}$$

In addition, the states describing the oxygen mass fraction of the intake and exhaust manifold, i.e., X_{Oim} and X_{Oem}, are not considered in the present study as the rest of the states do not depend on these two states. Also, the parameters of the oxygen mass fractions are assumed to be constant and known. The governing equations for the remaining six states are as follows:

$$\frac{d}{dt}p_{im} = \frac{R_a T_{im}}{V_{im}}(W_c + W_{egr} - W_{ei}), \tag{3}$$

$$\frac{d}{dt}p_{em} = \frac{R_e T_{em}}{V_{em}}(W_{eo} - W_t - W_{egr}), \tag{4}$$

$$\frac{d}{dt}\omega_t = \frac{P_t \eta_m - P_c}{J_t \omega_t}, \tag{5}$$

$$\frac{d\tilde{u}_{egr1}}{dt} = \frac{1}{\tau_{egr1}}\left[u_{egr}(t - \tau_{degr}) - \tilde{u}_{egr1}\right], \tag{6}$$

$$\frac{d\tilde{u}_{egr2}}{dt} = \frac{1}{\tau_{egr2}}\left[u_{egr}(t - \tau_{degr}) - \tilde{u}_{egr2}\right], \tag{7}$$

$$\frac{d\tilde{u}_{vgt}}{dt} = \frac{1}{\tau_{vgt}}\left[u_{vgt}(t - \tau_{dvgt}) - \tilde{u}_{vgt}\right]. \tag{8}$$

Two additional equations used for the computation of T_{em} in Eq. (4) read as:

$$T_1 = x_r T_e + (1 - x_r) T_{im}, \tag{9}$$

$$x_r = \frac{\Pi_e^{1/\gamma_a} x_p^{-1/\gamma_a}}{r_c x_v}, \tag{10}$$

where T_1 is the temperature when the inlet valve closes after the intake stroke and mixing, and x_r is the residual gas fraction. A brief discussion on the governing equations of the engine model is presented in "Appendix 2". Interested readers can also refer to[1] for more details.

In the present study, we have field measurements on a certain number of variables, i.e., p_{im}, p_{em}, ω_t, and W_{egr} as well as the inputs, i.e., $\boldsymbol{u}$ and n_e, at discrete times. Further, some of the parameters in the system, e.g., A_{egrmax}, η_{sc}, h_{tot} and A_{vgtmax}, which are difficult to measure directly, are unknown. A_{egrmax} is the maximum effective area of the EGR valve, η_{sc} is the compensation factor for non-ideal cycles, h_{tot} is the total heat transfer coefficient of the exhaust pipes and A_{vgtmax} is the maximum area in the turbine that the gas flows through. From the field prediction of these parameters, we can infer the health of the engine; a higher deviation from their design value may indicate a fault in the system. We are interested in (1) predicting the dynamics of all the variables in Eqs. (3)–(10), and (2) identifying the unknown parameters in the system, given field measurements on p_{im}, p_{em}, ω_t, and W_{egr} as well as Eqs. (3)–(10). We refer to the above problem as the *inverse problem* in this study. Specifically, the following cases are considered for a detailed study:

Case 1 Prediction of dynamics of the system and identification of 3 unknown parameters A_{egrmax}, η_{sc} and h_{tot} with clean data of p_{im}, p_{em}, ω_t, and W_{egr}.

Case 2 Prediction of dynamics of the system and identification of 3 unknown parameters A_{egrmax}, η_{sc} and h_{tot} with noisy data of p_{im}, p_{em}, ω_t, and W_{egr}.

Case 3 Prediction of dynamics of the system and identification of 4 unknown parameters A_{egrmax}, η_{sc}, h_{tot} and A_{vgtmax} with clean data of p_{im}, p_{em}, ω_t, and W_{egr}.

Case 4 Prediction of dynamics of the system and identification of 4 unknown parameters A_{egrmax}, η_{sc}, h_{tot} and A_{vgtmax} with noisy data of p_{im}, p_{em}, ω_t, and W_{egr}.

In the present study, we consider self-adaptive weights[9] (discussed in "Methodology" and "Appendix 1") in our loss function. We study the above four cases using self-adaptive weight. In order to understand the effect and importance of self-adaptive weights in the convergence and accuracy of results, we consider one more case without self-adaptive weight,

Case 5 Prediction of dynamics of the system and identification of 4 unknown parameters A_{egrmax}, η_{sc}, h_{tot} and A_{vgtmax} with clean data of p_{im}, p_{em}, ω_t, and W_{egr} without self-adaptive weights.

The results of **Case 1** and **Case 2** are presented in "Appendix 7". First, we study the results of **Case 3** and **Case 5** to understand the accuracy and convergence of PINN method and the importance of self-adaptive weights. Then, we study the results of **Case 4**. The results for **Case 3**, **Case 4** and **Case 5** are discussed in "Results and discussions".

Methodology

We consider to employ the deep learning algorithm, particularly, the physics-informed neural networks (PINNs), to solve the inverse problem discussed in "Problem setup". To begin with, we first briefly review the basic principle of PINNs, and then we discuss how to employ PINNs for the present inverse problem.

PINNs for inverse problems in the diesel engine.

We first briefly review the PINNs[2,4–7] for solving inverse problems, and then we introduce how to employ the PINNs to solve the specific problem that we are of interest for the diesel engine.

As illustrated in Fig. 2, the PINN is composed of two parts, i.e., a fully-connected neural network which is to approximate the solution to a particular differential equation and the physics-informed part in which the automatic differentiation[3] is employed to encode the corresponding differential equation. Further, $\boldsymbol{\Lambda}$ represents the unknowns in the equation, which can be either a constant or a field. In particular, $\boldsymbol{\Lambda}$ are trainable variables as the unknowns are constant, but they could also be approximated by a DNN if the unknown is a field. The loss function for solving the inverse problems consists of two parts, i.e., the data loss and the equation loss, which reads as:

$$\mathcal{L}(\boldsymbol{\theta};\boldsymbol{\Lambda}) = \underbrace{\frac{1}{M}\sum_{i=1}^{M}|\hat{y}(t_i;\boldsymbol{\theta}) - y(t_i)|^2}_{\text{data loss}} + \underbrace{\frac{1}{N}\sum_{i=1}^{N}|r(t_i;\boldsymbol{\theta};\boldsymbol{\Lambda})|^2}_{\text{equation loss}} \tag{11}$$

where $\boldsymbol{\theta}$ denotes the parameters in the DNN; M and N represent the number of measurements and the residual points, respectively; $\hat{y}(t_i;\boldsymbol{\theta})$ denotes the prediction of DNN at the time t_i; $y(t_i)$ is the measurement at t_i, and $r(t_i;\boldsymbol{\theta};\boldsymbol{\Lambda})$ represents the residual of the corresponding differential equation, which should be zero in the entire domain. By minimizing the loss in Eq. (11), we can obtain the optimal parameters, i.e., $\boldsymbol{\theta}$, of the DNN as well as the unknowns, i.e., $\boldsymbol{\Lambda}$, in the system. In the present study, we have a few empirical equations that we approximate using DNNs. These DNNs are trained first using data and considered in place of these empirical formulae. We fixed the parameters of these networks when we minimized the loss function for the PINN model. Furthermore, note that here we employ the system described by one equation as the example to demonstrate how to use PINNs for solving inverse problems. For the system with more than one equation, we can either utilize an DNN with multiple outputs or multiple DNNs as the surrogates for the solutions to differential equations. In addition, a similar idea can also be employed for systems with multiple unknown fields. The loss function can then be rewritten as

$$\mathcal{L}(\boldsymbol{\theta};\boldsymbol{\Lambda}) = \underbrace{\sum_{k=1}^{K}\left[\frac{1}{M_k}\sum_{i=1}^{M_k}|\hat{y}_k(t_i;\boldsymbol{\theta}) - y_k(t_i)|^2\right]}_{\text{data loss}} + \underbrace{\sum_{l=1}^{L}\left[\frac{1}{N_l}\sum_{i=1}^{N_l}|r_l(t_i;\boldsymbol{\theta};\boldsymbol{\Lambda})|^2\right]}_{\text{equation loss}} \tag{12}$$

where K and L denote the number of variables that can be measured as well as the equations, respectively; M_k and N_l are the number of measurements for the kth variable and the number of residual points for the lth equation, respectively; and $\boldsymbol{\Lambda}$ collects all the unknowns in the system.

For the inverse problem presented in "Problem setup", we are interested in (1) learning the dynamics of the six states (2) inferring the unknown parameters in the system, given measurements on $\{p_{im}, p_{em}, \omega_t, W_{egr}\}$ as well as Eqs. (3)–(10), using PINNs. Specifically, we utilize six DNNs as the surrogates for the solutions to different equations, and the corresponding equations are encoded using the automatic differentiation, as illustrated in Table 1. In addition, the loss for training the PINNs for Case 1 to Case 4 is expressed as follows:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\Lambda},\boldsymbol{\lambda}_{p_{im}},\boldsymbol{\lambda}_{p_{em}},\boldsymbol{\lambda}_{\omega_t},\boldsymbol{\lambda}_{W_{egr}},\lambda_{T_1}) =& \mathcal{L}_{p_{im}} + \mathcal{L}_{p_{em}} + \mathcal{L}_{\omega_t} + \mathcal{L}_{u_{egr1}} \\ &+ \mathcal{L}_{u_{egr2}} + \mathcal{L}_{u_{vgt}} + 10\times\mathcal{L}_{x_r} + \lambda_{T_1}\times\mathcal{L}_{T_1} \\ &+ \mathcal{L}^{ini}_{p_{im}} + \mathcal{L}^{ini}_{p_{em}} + \mathcal{L}^{ini}_{\omega_t} + \mathcal{L}^{ini}_{\tilde{u}_{egr1}} \\ &+ \mathcal{L}^{ini}_{\tilde{u}_{egr2}} + \mathcal{L}^{ini}_{\tilde{u}_{vgt}} + \mathcal{L}^{ini}_{x_r} + 100\times\mathcal{L}^{ini}_{T_1} \\ &+ \mathcal{L}^{data}_{p_{im}}(\boldsymbol{\lambda}_{p_{im}}) + \mathcal{L}^{data}_{p_{em}}(\boldsymbol{\lambda}_{p_{em}}) \\ &+ \mathcal{L}^{data}_{\omega_t}(\boldsymbol{\lambda}_{\omega_t}) + \mathcal{L}^{data}_{W_{egr}}(\boldsymbol{\lambda}_{W_{egr}}),\end{aligned} \tag{13}$$

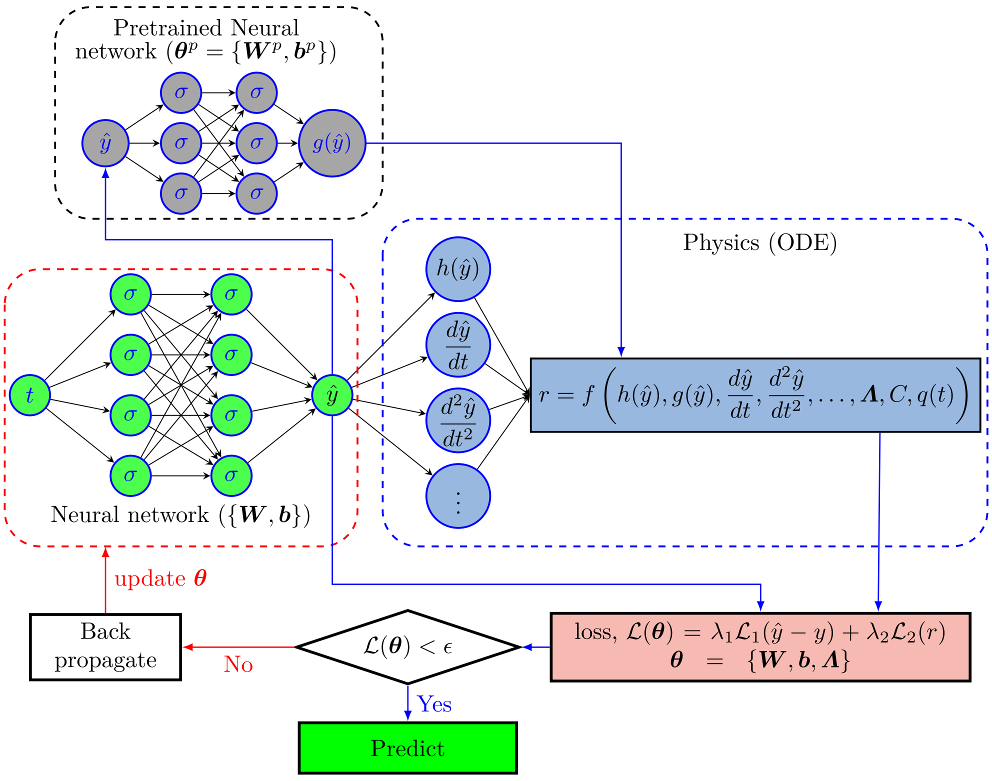

Pretrained Neural network ($\boldsymbol{\theta}^p = \{\boldsymbol{W}^p, \boldsymbol{b}^p\}$)
$\hat{y}$
σ
$g(\hat{y})$
Physics (ODE)
$h(\hat{y})$
$\frac{d\hat{y}}{dt}$
$\frac{d^2\hat{y}}{dt^2}$
$\vdots$
$r = f\left(h(\hat{y}), g(\hat{y}), \frac{d\hat{y}}{dt}, \frac{d^2\hat{y}}{dt^2}, \ldots, \boldsymbol{\Lambda}, C, q(t)\right)$
t
Neural network ($\{\boldsymbol{W}, \boldsymbol{b}\}$)
update $\boldsymbol{\theta}$
Back propagate
No
$\mathcal{L}(\boldsymbol{\theta}) < \epsilon$
loss, $\mathcal{L}(\boldsymbol{\theta}) = \lambda_1 \mathcal{L}_1(\hat{y} - y) + \lambda_2 \mathcal{L}_2(r)$
$\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\Lambda}\}$
Yes
Predict

Figure 2. Schematic of physics-informed neural networks (PINNs) for inverse problems: the left part of the figure, enclosed in the red dashed line, shows a DNN whose input is time. The DNN is to approximate the solution (y) to a differential equation. The top left part of the figure enclosed in the black dashed line shows an DNN whose input is the output y (maybe with other input, e.g. ambient condition). The output of this network is a function $g(y)$. This network is pre-trained with laboratory data of y and $g(y)$. The right part of the figure, enclosed in the blue dashed line, denotes the physics loss/residue. The DNN (enclosed in a red dashed line) approximates the solution to any differential equation, and the equation is encoded using automatic differentiation. The total loss $\mathcal{L}(\boldsymbol{\theta})$ includes the loss of equation as well as the data. The λ_1 and λ_2 are two weights to the data loss and physics loss, which may be fixed or adaptive depending upon the problem and solution method. $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\Lambda}\}$ represents the parameters in DNN, $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weights and biases of DNN, respectively and $\boldsymbol{\Lambda}$ are the unknown parameters of the ODE; σ is the activation function, $q(t)$ is the right-hand side (RHS) of the differential equation (source term), h is the function of the predicted variable, and r is the residual for the equation. $\boldsymbol{\theta}^P = \{\boldsymbol{W}^P, \boldsymbol{b}^P\}$ represents the parameters in pre-trained neural network, $\boldsymbol{W}^P$ and $\boldsymbol{b}^P$ are the weights and biases of the pre-trained neural network.

Variables	p_{im}, p_{em}	x_r	T_1	$\tilde{u}_{egr1}, \tilde{u}_{egr2}$	ω_t	$\tilde{u}_{vgt}$
Surrogate	$\mathcal{N}_1(t;\boldsymbol{\theta}_1)$	$\mathcal{N}_2(t;\boldsymbol{\theta}_2)$	$\mathcal{N}_3(t;\boldsymbol{\theta}_3)$	$\mathcal{N}_4(t;\boldsymbol{\theta}_4)$	$\mathcal{N}_5(t;\boldsymbol{\theta}_5)$	$\mathcal{N}_6(t;\boldsymbol{\theta}_6)$
Equations	(3) and (4)	(10)	(9)	(6) and (7)	(5)	(8)

Table 1. Neural network surrogates employed PINNs for solving the inverse problems. $\mathcal{N}_i(t;\boldsymbol{\theta}_i), i = 1, \ldots, 6$ denotes the surrogate for the ith DNN parameterized by $\boldsymbol{\theta}_i$ with the input t. In particular, $\mathcal{N}_1(t;\boldsymbol{\theta}_1)$ and $\mathcal{N}_4(t;\boldsymbol{\theta}_4)$ have two outputs, which are used to approximate $\{p_{im}, p_{em}\}$ and $\{\tilde{u}_{egr1}, \tilde{u}_{egr2}\}$, respectively; the remaining DNNs have only one output.

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_6)$ are the parameters of all NNs in PINNs, $\boldsymbol{\Lambda}$ are the unknown parameters, which will be inferred from the given measurements, $\mathcal{L}_\phi$, $\phi = (p_{im}, p_{em}, \omega_t, \tilde{u}_{egr1}, \tilde{u}_{egr2}, \tilde{u}_{vgt}, x_r, T_1)$ are the losses for the corresponding equations, and $\mathcal{L}_\psi^{data}$, $\psi = (p_{im}, p_{em}, \omega_t, W_{egr})$ are the losses for the corresponding measurements, and $\mathcal{L}_\phi^{ini}$, $\phi = (p_{im}, p_{em}, \omega_t, \tilde{u}_{egr1}, \tilde{u}_{egr2}, \tilde{u}_{vgt}, x_r, T_1)$ are the losses for the initial conditions, λ_{T_1}, $\boldsymbol{\lambda}_{p_{im}}$, $\boldsymbol{\lambda}_{p_{em}}$, $\boldsymbol{\lambda}_{\omega_t}$, and $\boldsymbol{\lambda}_{W_{egr}}$ are the weights for different loss terms which are used to balance each term in the loss function. In particular, the self-adaptive weight technique proposed in[9], which is capable of tuning the weights automatically, is utilized here to obtain the optimal, λ_{T_1} $\boldsymbol{\lambda}_{p_{im}}$, $\boldsymbol{\lambda}_{p_{em}}$, $\boldsymbol{\lambda}_{\omega_t}$, and $\boldsymbol{\lambda}_{W_{egr}}$. More details for self-adaptive weights in PINN can be found in "Appendix 1".

In the Case 5, where we have not considered self-adaptive weights, so the loss function is given as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Lambda}) =& \mathcal{L}_{p_{im}} + \mathcal{L}_{p_{em}} + \mathcal{L}_{\omega_t} + \mathcal{L}_{u_{egr1}} \\ &+ \mathcal{L}_{u_{egr2}} + \mathcal{L}_{u_{vgt}} + 10 \times \mathcal{L}_{x_r} + 10^3 \times \mathcal{L}_{T_1} \\ &+ \mathcal{L}^{ini}_{p_{im}} + \mathcal{L}^{ini}_{p_{em}} + \mathcal{L}^{ini}_{\omega_t} + \mathcal{L}^{ini}_{\tilde{u}_{egr1}} \\ &+ \mathcal{L}^{ini}_{\tilde{u}_{egr2}} + \mathcal{L}^{ini}_{\tilde{u}_{vgt}} + \mathcal{L}^{ini}_{x_r} + 100 \times \mathcal{L}^{ini}_{T_1} \\ &+ 10^3 \times \mathcal{L}^{data}_{p_{im}} + 10^3 \times \mathcal{L}^{data}_{p_{em}} \\ &+ 10^3 \times \mathcal{L}^{data}_{\omega_t} + 10^3 \times \mathcal{L}^{data}_{W_{egr}},\end{aligned} \tag{14}$$

As for training the PINNs in the present study, we first employ the first-order optimizer, i.e., Adam[23], to train the parameters in the NNs, unknowns in the systems as well as the self-adaptive weights for a certain number of steps. We then fix the self-adaptive weight and employed Adam to train the parameters in the NNs, and unknowns in the systems for another certain number of steps. We then switch to the second-order accuracy optimizer, i.e., LBFGS-B, to further optimize the parameters in the NNs and the unknowns in the systems. Note that the self-adaptive weights are optimized at the first training stage of Adam only, and they are fixed during the second training stage of Adam and LBFGS-B training with the values at the end of the first stage of Adam optimization.

Neural network surrogates for empirical formulae. In the mean value engine model proposed in[1], empirical formulae, e.g., polynomial functions, are employed for the volumetric efficiency (η_{vol}), effective area ratio function for EGR valve (f_{egr}), turbine mechanical efficiency (η_{tm}), effective area ratio function for VGT (f_{vgt}), choking function (for VGT) (f_{Π_t}), compressor efficiency (η_c), and volumetric flow coefficient (for the compressor) (Φ_c). Note that these empirical formulae are engine-specific and may not be appropriate for the diesel engines considered in the present study. Deep neural networks (DNNs), which are known to be universal approximators of any continuous function, are thus utilized as more general surrogates for the empirical formulae here. Particularly, we employ six DNNs for the aforementioned variables, and the inputs for each DNN are presented in Table 2.

We now discuss the training of the DNNs illustrated in Table 2. In laboratory experiments, measurements on all variables are available. We can then train the neural network surrogates in Table 2 using the data collected in the laboratory. The loss function considered for the training of these networks is

$$\mathcal{L}_i(\boldsymbol{\theta}^P_i) = \frac{1}{n_i}\sum_{j=1}^{n_i}\left[y_i^{(j)} - \hat{y}_i^{(j)}\right]^2 = \frac{1}{n_i}\sum_{j=1}^{n_i}\left[y_i^{(j)} - \mathcal{N}_i^{(P)}(\boldsymbol{x}_i; \boldsymbol{\theta}^P_i)^{(j)}\right]^2, \quad i = 1, 2, \ldots, 6 \tag{15}$$

where $i = 1, 2, \ldots, 6$ are the different neural networks for the approximation of the empirical formulae, $\boldsymbol{x}_i$ are the input corresponds to the ith network, $\hat{y}_i$ and y_i are the output of the ith network and the corresponding labelled values respectively, n_i is the number of labelled dataset corresponds to the ith neural network. The laboratory data required for calculating labelled data for training each of these networks are shown in Table 3 (in "Data generation"). We discuss the calculation of labelled data from the laboratory data in "Appendix 4". We train these networks using the Adam optimizer. Upon the training of these DNNs, we will plug them in the PINNs to replace the empirical models, which are represented by the pretrained neural network with the output $g(y)$ in Fig. 2.

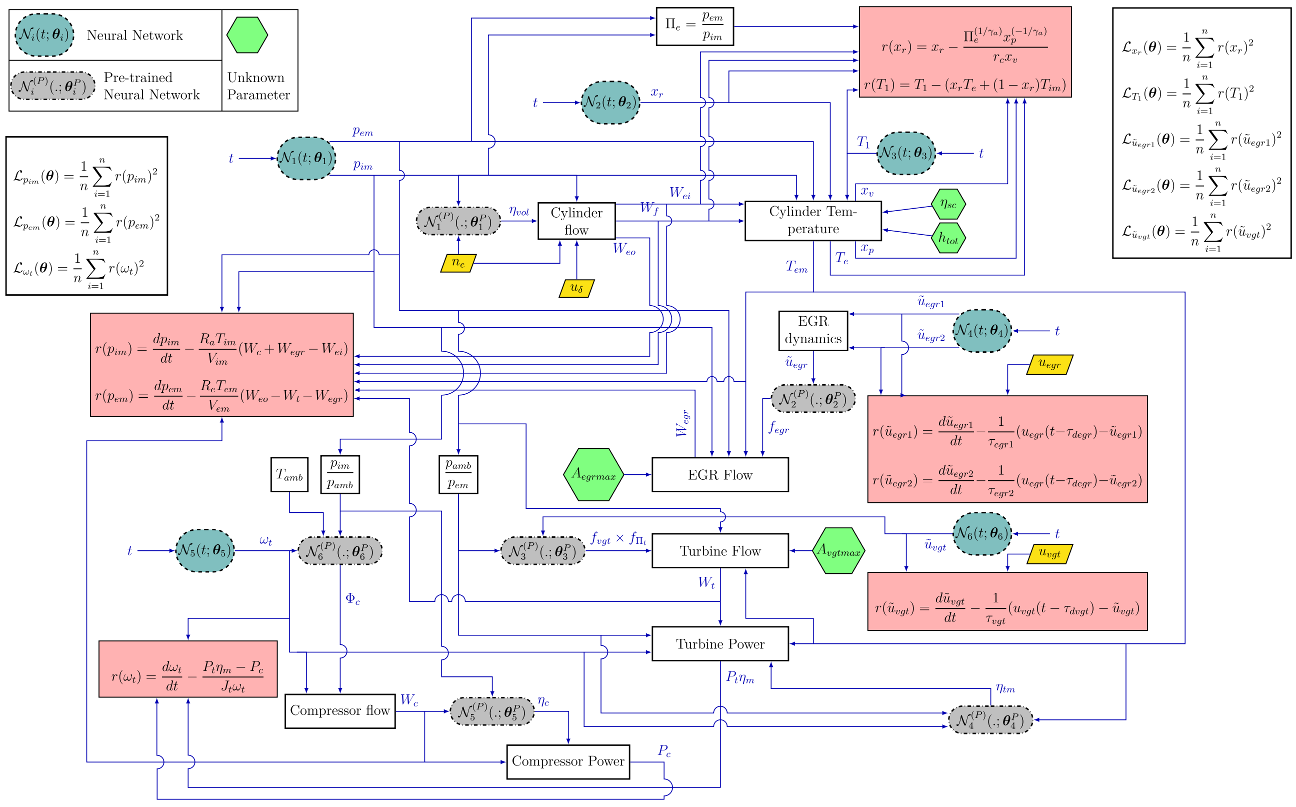
Flowchart for PINN model for diesel engine. In "Problem setup", we discussed the problem setup, and in subsequent sections, we discussed the approximation of different variables using neural networks as well as the basics of the PINN method and the implementation of PINN in the present problem. In Fig. 2, we have shown a schematic diagram along with a pre-trained network for a general ordinary differential equation. In this section, we show a complete flowchart for the calculation of physics loss functions for the engine problem. In Fig. 3, we show the flow chart for calculating the physics-informed loss for the present problem. Note that we have not shown the data loss and the self-adaptive weights in the flow chart.

Data generation. We now discuss the generation of data for training the NNs utilized in this study. Specifically, we have mainly two different types of data here: (1) the data collected from the laboratory that are used to train the DNN surrogates to replace the empirical formulae used in[1]; and (2) field data p_{im}, p_{em}, ω_t and W_{egr}.

In laboratory experiments, we have measurements on state variables, some of which can be employed for training the neural network surrogates for the empirical formulae. The laboratory data required to calculate the

Variable	η_{vol}	f_{egr}	F_{vgt,Π_t}	η_{tm}	η_c	Φ_c
Surrogate	$\mathcal{N}_1^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_1^P)$	$\mathcal{N}_2^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_2^P)$	$\mathcal{N}_3^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_3^P)$	$\mathcal{N}_4^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_4^P)$	$\mathcal{N}_5^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_5^P)$	$\mathcal{N}_6^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_6^P)$
Input ($\boldsymbol{x}$)	$\{p_{im}, n_e\}$	$\tilde{u}_{egr}$	$\{\tilde{u}_{vgt}, \Pi_t\}$	$\{\omega_t, T_{em}, \Pi_t\}$	$\{W_c, \Pi_c\}$	$\{T_{amb}, \Pi_c, \omega_t\}$
Equations[†]	(27)	(43)	(73)	(52)	(59)	(69)

Table 2. Neural network surrogates for empirical formulae $\mathcal{N}_i^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_i^P), i = 1, \ldots, 6$ denotes the surrogate for the ith DNN parameterized by θ_i^P with the input $\boldsymbol{x}$. All the neural networks have one output each. [†]The empirical equations are discussed in Appendix.



$\mathcal{N}_i(t;\boldsymbol{\theta}_i)$ Neural Network
$\mathcal{N}_i^{(P)}(.;\boldsymbol{\theta}_i^P)$ Pre-trained Neural Network
Unknown Parameter
$\mathcal{L}_{p_{im}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} r(p_{im})^2$
$\mathcal{L}_{p_{em}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} r(p_{em})^2$
$\mathcal{L}_{\omega_t}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} r(\omega_t)^2$
$\Pi_e = \frac{p_{em}}{p_{im}}$
$r(x_r) = x_r - \frac{\Pi_e^{(1/\gamma_a)} x_p^{(-1/\gamma_a)}}{r_c x_v}$
$r(T_1) = T_1 - (x_r T_e + (1 - x_r) T_{im})$
$\mathcal{L}_{x_r}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} r(x_r)^2$
$\mathcal{L}_{T_1}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} r(T_1)^2$
$\mathcal{L}_{\tilde{u}_{egr1}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} r(\tilde{u}_{egr1})^2$
$\mathcal{L}_{\tilde{u}_{egr2}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} r(\tilde{u}_{egr2})^2$
$\mathcal{L}_{\tilde{u}_{vgt}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} r(\tilde{u}_{vgt})^2$
t $\mathcal{N}_1(t;\boldsymbol{\theta}_1)$ p_{em} p_{im}
t $\mathcal{N}_2(t;\boldsymbol{\theta}_2)$ x_r
T_1 $\mathcal{N}_3(t;\boldsymbol{\theta}_3)$ t
$\mathcal{N}_1^{(P)}(.;\boldsymbol{\theta}_1^P)$ η_{vol} Cylinder flow W_f W_{ei} W_{eo}
n_e u_δ
Cylinder Temperature x_v x_p η_{sc} h_{tot} T_{em} T_e
$r(p_{im}) = \frac{dp_{im}}{dt} - \frac{R_a T_{im}}{V_{im}}(W_c + W_{egr} - W_{ei})$
$r(p_{em}) = \frac{dp_{em}}{dt} - \frac{R_e T_{em}}{V_{em}}(W_{eo} - W_t - W_{egr})$
EGR dynamics $\tilde{u}_{egr1}$ $\tilde{u}_{egr2}$ $\mathcal{N}_4(t;\boldsymbol{\theta}_4)$ t u_{egr}
$\tilde{u}_{egr}$ $\mathcal{N}_2^{(P)}(.;\boldsymbol{\theta}_2^P)$ f_{egr} W_{egr}
$r(\tilde{u}_{egr1}) = \frac{d\tilde{u}_{egr1}}{dt} - \frac{1}{\tau_{egr1}}(u_{egr}(t-\tau_{degr}) - \tilde{u}_{egr1})$
$r(\tilde{u}_{egr2}) = \frac{d\tilde{u}_{egr2}}{dt} - \frac{1}{\tau_{egr2}}(u_{egr}(t-\tau_{degr}) - \tilde{u}_{egr2})$
A_{egrmax} EGR Flow
T_{amb} $\frac{p_{im}}{p_{amb}}$ $\frac{p_{amb}}{p_{em}}$
t $\mathcal{N}_5(t;\boldsymbol{\theta}_5)$ ω_t $\mathcal{N}_6^{(P)}(.;\boldsymbol{\theta}_6^P)$ Φ_c
$\mathcal{N}_3^{(P)}(.;\boldsymbol{\theta}_3^P)$ $f_{vgt} \times f_{\Pi_t}$ Turbine Flow A_{vgtmax} W_t
$\mathcal{N}_6(t;\boldsymbol{\theta}_6)$ t $\tilde{u}_{vgt}$ u_{vgt}
$r(\tilde{u}_{vgt}) = \frac{d\tilde{u}_{vgt}}{dt} - \frac{1}{\tau_{vgt}}(u_{vgt}(t-\tau_{dvgt}) - \tilde{u}_{vgt})$
Turbine Power $P_t\eta_m$ η_{tm} $\mathcal{N}_4^{(P)}(.;\boldsymbol{\theta}_4^P)$
$r(\omega_t) = \frac{d\omega_t}{dt} - \frac{P_t\eta_m - P_c}{J_t\omega_t}$
Compressor flow W_c $\mathcal{N}_5^{(P)}(.;\boldsymbol{\theta}_5^P)$ η_c
Compressor Power P_c

Figure 3. Flow chart for the proposed PINN model for the inverse problem for the engine for prediction of dynamics of the system variables and estimation of unknown parameters. The inputs are input control vector $\{u_\delta, u_{egr}, u_{vgt}\}$ and engine speed n_e. Six neural network $\boldsymbol{\mathcal{N}}_i(t;\boldsymbol{\theta}), i = 1, 2, \ldots, 6$ indicated in dashed rectangular oval takes time t as input and predict p_{im}, p_{em}, x_r and T_1 $\tilde{u}_{egr1}$, $\tilde{u}_{egr2}$, ω_t and $\tilde{u}_{vgt}$ as shown in Table 1. Four unknown parameters indicated in hexagon are η_{sc}, h_{tot}, A_{egrmax} and A_{vgtmax}. Six pre-trained neural networks $\boldsymbol{\mathcal{N}}_i^{(P)}(.;\boldsymbol{\theta}), i = 1, 2, \ldots, 6$ indicated in dashed-dotted rectangular oval takes appropriate input and predict the empirical formulae as shown in Table 2. The parameters (weights and biases) of these pre-trained DNNs are kept fixed to predict the empirical formulae. There are eight main blocks calculating different variables. The equations for the calculation of each of the quantities are shown in Appendix 6. Cylinder flow: calculates W_{ei}, W_f and W_{eo} using Eqs. (24), (26) and (25), respectively. Cylinder temperature: calculates x_v, x_p, T_e and T_{em} using Eqs. (33), (32), (28) and (35), respectively. h_{tot} and η_{sc} are considered as learnable parameters in the calculation of T_{em} and T_e respectively. EGR dynamics: calculated $\tilde{u}_{egr}$ using Eq. (38). EGR flow: calculates EGR mass flow W_{egr} using Eq. (39). A_{egrmax} is considered as learnable parameter. Compressor flow: calculates compressor mass flow W_c using Eq. (68). Compressor power: calculates compressor power P_c using Eq. (58). Turbine flow: calculates turbine mass flow W_t using Eq. (46). A_{vgtmax} is considered as trainable parameter. Turbine power: calculates effective turbine power $P_t\eta_m$ using Eq. (51). There are five blocks, which calculate the residual of the equation. The first block calculates the residual for state equations for p_{im} and p_{em}; the second one calculates the residual for the equations of x_r and T_1; the third block calculates the residuals for state equations for $\tilde{u}_{egr1}$ and $\tilde{u}_{egr2}$; the fourth block calculates the residual for the state equation for $\tilde{u}_{vgt}$, and the fifth block calculates the residual for the state equation for ω_t. There are another two blocks, which calculate the physics loss. The first one calculates the physics loss corresponding to state variable p_{im}, p_{em} and ω_t. The second block calculates state physics loss corresponding to state variables $\tilde{u}_{egr1}$, $\tilde{u}_{egr2}$, $\tilde{u}_{vgt}$ and physics loss corresponding to x_r and T_1. The data losses can be calculated from the variables calculated from the appropriate blocks.

labelled data for each of the surrogates are shown in Table 3. The calculation of labelled data from laboratory data is discussed in "Appendix 4". After training, we consider these pre-trained surrogates in field experiments in place of the empirical formulae. The parameters of these networks are kept constant in the PINNs model for the field experiment.

In field experiments, we have records for the four inputs, i.e., $\boldsymbol{u} = \{u_\delta, \; u_{egr}, \; u_{vgt}\}$ and n_e. In addition, we only have measurements on four variables in field experiments, i.e., the intake manifold pressure (p_{im}), exhaust manifold pressure (p_{em}), turbine speed (ω_t) and EGR mass flow (W_{egr}).

In both the laboratory and field experiments, we have the records for the inputs (i.e., $\boldsymbol{u} = \{u_\delta, \; u_{egr}, \; u_{vgt}\}$ and n_e) from an actual engine running conditions. Considering that we only have a certain number of records for the variables in the running engine, which cannot be used to verify our PINN model since our objective is to use it to predict the whole gas flow dynamics in the engine. We, therefore, take the records for the real inputs (i.e., $\boldsymbol{u} = \{u_\delta, \; u_{egr}, \; u_{vgt}\}$ and n_e) and employ them as the inputs for the governing equations Eqs. (3)–(8). We then

Empirical quantities [††, †††]	**Symbol**	**Laboratory test data required** [††††]
Volumetric efficiency ("Cylinder")	η_{vol}	Intake manifold pressure (p_{im})
		Engine speed (n_e)
		Total mass flow from the intake manifold into the cylinders (W_{ei})
		Intake manifold temperature (T_{im})
Effective area ratio function for EGR ("EGR valve")	f_{egr}	EGR position ($\tilde{u}_{egr}$)
		EGR mass flow (W_{egr})
		Exhaust manifold pressure (p_{em})
		Intake manifold pressure (p_{im})
		Exhaust manifold temperature (T_{em})
Effective area ratio function for VGT (f_{vgt}) and chocking function (f_{Π_t}) ("Turbocharger")	$f_{vgt} \times f_{\Pi_t}$	VGT position ($\tilde{u}_{vgt}$)
		Exhaust manifold pressure (p_{em})
		Ambient pressure (p_{amb})
		Turbine mass flow (W_t)
		Exhaust manifold pressure (p_{em})
		Exhaust manifold temperature (T_{em})
Turbine mechanical efficiency [†††††] ("Turbocharger")	η_{tm}	Turbine speed (ω_t)
		Exhaust manifold temperature (T_{em})
		Exhaust manifold pressure (p_{em})
		Ambient pressure (p_{amb})
		Compressor mass flow (W_c)
		Compressor temperature (T_c)
		Ambient temperature (T_{amb})
		Turbine mass flow (W_t)
Compressor efficiency ("Compressor")	η_c	Intake manifold pressure (p_{im})
		Compressor mass flow (W_c)
		Temperature after the compressor (T_c)
		Ambient temperature (T_{amb})
		Ambient pressure (p_{amb})
Volumetric flow coefficient for compressor ("Compressor")	Φ_c	Turbine speed (ω_t)
		Compressor mass flow (W_c)
		Intake manifold pressure (p_{im})
		Ambient temperature (T_{amb})
		Ambient pressure (p_{amb})

Table 3. List of empirical formulae represented using a pre-trained neural network and lab test data required for their training [†]. [†]It is assumed that the parameters/constant are known, however not the coefficients for the empirical formulae. [††]The definition of the quantities are discussed in relevant sections in "Appendix 2". [†††]The calculations of the empirical quantify from the laboratory data are included in "Appendix 4". [††††]A brief discussion on instrumentation and test procedure is included in "Appendix 3". [†††††] For calculation of η_{tm}, dynamic data are required (discussed in "Appendix 4" and "Appendix 8: Neural network surrogates for empirical formulae").

solve these equations using Simulink[22] to obtain the dynamics for all variables. We use the data from Simulink to mimic the real-world measurements, which are employed as the training data for PINNs and the DNN for the pre-trained networks. The remaining data are used as the validation data to test the accuracy of PINN for reconstructing the gas dynamics in a running engine, given partial observations. Given that the real measurements are generally noisy, we add 3%, 3%, 1% and 10% Gaussian noise in p_{im}, p_{em}, ω_t and W_{egr}, respectively. These different signals have different noise values because they are different measurements with different noise characteristics.

In the present study, we consider two sets of input data in the training and testing of the surrogate neural networks for the empirical formulae. The first set of data (Set-I) is two (2) h of data collected at a sampling rate of 1 s. This control input vector $\{u_\delta, u_{egr}, u_{vgt}\}$ and n_e are considered to generate simulated data with different ambient conditions, which are shown in Table 4. The second set of data (Set-II) is twenty-minute (20 min) data collected at a sampling rate of 0.2 s. This control input vector $\{u_\delta, u_{egr}, u_{vgt}\}$ and n_e are considered to generate simulated data with Case-V ambient conditions.

The labelled data for the training of surrogate neural network for η_{vol}, F_{vgt,Π_t}, η_c and Φ_c are generated for Case-I to Case-IV with a $dt = 0.2$ s. The testing data are generated for Case-V with the same dt. We observed from the engine model that the EGR valve actuator is independent of the other system of the engine and depends only on the EGR control signal (u_{egr}). Thus, for the training of surrogate neural network for f_{egr} ($\mathcal{N}_2^{(P)}(:,\boldsymbol{\theta})$), we

Case	T_{amb} (kelvin)	$p_{amb} \times 10^5$ (Pa) (Approx. elevation)	Input	Sampling (s)	Purpose
				dt	
Case-I	233.15 (−40 °C)	0.7000 (at 3000 m)	Set-I	1	Training
Case-II	233.15 (−40 °C)	1.0111 (at 17.9 m)	Set-I	1	Training
Case-III	270.15 (−3 °C)	0.7000 (at 3000 m)	Set-I	1	Training
Case-IV	313.15 (40 °C)	1.0111 (at 17.9 m)	Set-I	1	Training
Case-V	298.15 (25 °C)	0.8000 (at 1837 m)	Set-II	0.2	Testing

Table 4. Ambient conditions for training and testing of neural networks: the different ambient conditions are considered for generating training and testing data. The input data Set-I is 2 h of input control vector $\{u_\delta, u_{egr}, u_{vgt}\}$ and n_e collected from an actual engine running condition. Similarly, Set-II is 20-min of input control vector $\{u_\delta, u_{egr}, u_{vgt}\}$ and n_e collected from an actual engine running condition. Case-I to Case-IV are considered for the training of the surrogate neural network for the empirical formulae ($\boldsymbol{\mathcal{N}}_i^{(P)}(:, \boldsymbol{\theta}_i)$), while Case-V is considered for testing of these networks. The data for the field data are also considered from Case-V.

consider the training data set corresponding to Case-I only and the testing data set corresponding to Case-V. The labelled data for η_{tm} are calculated from Eq. (44) ("Turbocharger"), which is a differential equation, thus requires a finer dt. The simulated data for the calculation of labelled η_{tm} are generated with $dt = 0.025$ s in all the Cases. We assume that the Set-I data for the input control vector includes a good operating range for training surrogate neural networks for the empirical formulae. The field data (p_{im}, p_{em}, ω_t and W_{egr}) for the inverse problem are considered from Case-V.

Results and discussions

In this section, we demonstrate the applicability of proposed PINNs for solving the inverse problems discussed in "Problem setup". Case 1 and Case 2 have three unknown parameters, while Case 3 to Case 5 have four unknown parameters. The predicted values of the unknowns for all five cases are shown in Table 5. In this section, we will discuss the results of Case 3 to Case 5. The results of Case 1 and Case 2 are presented in "Appendix 7".

First, we study the results of Case 3 and Case 5 to understand the applicability of PINN and the importance of self-adaptive weight in accuracy and convergence. Then, we study the results of Case 4, which is similar to Case 3; however, with added noise in the field data considered. We also discuss the results for the surrogate for the empirical formulae in Appendix 12. Note that the results for all variables are presented in a normalized scale from zero to one using the following equation,

$$x_{scale} = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{16}$$

where x and x_{scale} are the data before and after scaling, respectively, x_min is the minimum value of true data of x within the time span considered, x_{max} is the maximum value of true data of x within the time span considered.

We are considering the input control vector $\{u_\delta, u_{egr}, u_{vgt}\}$ and engine speed n_e from an actual field record. It is assumed that these data have inherent noise in their records. Detailed studies are carried out considering a 1-min duration. The number of residual points considered in the physics-informed loss and data loss is 301 at equal $dt = 0.2$ s. The initial conditions considered for $\{p_{im},\ p_{em},\ x_r,\ T_1,\ \tilde{u}_{egr1},\ \tilde{u}_{egr2},\ \omega_t,\ \tilde{u}_{vgt}\}$ are $\{8.0239 \times 10^4,\ \ 8.1220 \times 10^4,\ \ 0.0505,\ \ 305.3786,\ \ 18.2518,\ \ 18.1813,1.5827 \times 10^3,\ \ 90.0317\}$ respectively. The measured field data are also considered for 1 min with equal $dt = 0.2$ s. Thus, each of the measured field quantities has 301 records.

The details of the neural networks considered for the PINN problem are shown in Table 6. We consider $\sigma(\cdot) = tanh(\cdot)$ activation function for hidden layers for all the neural networks. We would also like to emphasize

	A_{egramx}	η_{sc}	h_{tot}	A_{vgtmax}	Known variables	Predicted variables
True	4×10^{-4}	1.102	96.28	8.456×10^{-4}		
Case 1	3.93×10^{-4}	1.12	110	NA	Clean data of $p_{im}, p_{em}, \omega_t, W_{egr}$	The neural networks predict: $p_{im}, p_{em}, \tilde{u}_{vgt}, \tilde{u}_{egr1}, \tilde{u}_{egr2}, T_1, x_r$. The pretrained neural networks predict: $\eta_{vol}, \eta_{tm}, \eta_c, \Phi_c, F_{vgt,\Pi_t}, f_{egr}$. Other variables are derived from these predicted quantities
Case 2	3.93×10^{-4}	1.12	109	NA	Noisy data of $p_{im}, p_{em}, \omega_t, W_{egr}$	
Case 3	3.61×10^{-4}	0.962	113	7.86×10^{-4}	Clean data of $p_{im}, p_{em}, \omega_t, W_{egr}$	
Case 4	3.51×10^{-4}	0.834	134	7.27×10^{-4}	Noisy data of $p_{im}, p_{em}, \omega_t, W_{egr}$	
Case 5	2.28×10^{-4}	0.829	140	7.27×10^{-4}	Case 3 without self-adaptive weights	
Mask and scale considered						
Mask	Exponential	Softplus	Exponential	Exponential	For faster convergence and to have positive value	
Scale	$\times 10^{-4}$	$\times 1$	$\times 10$	$\times 10^{-4}$	Scale to obtain the parameters in physical domain	

Table 5. Predicted unknowns: predicted unknown parameters for different cases considered.

Neural network	Network size	Output	Outputs transformation [‡‡]	Scaling
$\mathcal{N}_1(t;\boldsymbol{\theta}_1)$	[1, 10, 10, 10, 2]	p_{im}, p_{em}	$S_P(p_{im}) + 0.5, S_P(p_{em})$	$\times 10^5$
$\mathcal{N}_2(t;\boldsymbol{\theta}_2)$	[1 10, 10, 1]	x_r	$S_P(x_r)$	$\times 0.03$
$\mathcal{N}_3(t;\boldsymbol{\theta}_3)$	[1, 15, 15, 15, 1]	T_1	$S_P(T_1) + 230/300$	$\times 300$
$\mathcal{N}_4(t;\boldsymbol{\theta}_4)$	[1, 10, 10, 10, 2]	$\tilde{u}_{egr1}, \tilde{u}_{egr2}$	$S(\tilde{u}_{egr1}, \tilde{u}_{egr2})$	$\times 100$
$\mathcal{N}_5(t;\boldsymbol{\theta}_5)$	[1, 10, 10, 1]	ω_t	$S_P(\omega_t)$	$\times 5 \times 10^3$
$\mathcal{N}_6(t;\boldsymbol{\theta}_6)$	[1, 10, 10, 1]	$\tilde{u}_{vgt}$	$S(\tilde{u}_{vgt})$	$\times 100$

Table 6. Details of neural network for PINNs: details of neural networks considered to approximate the state variables and T_1 and x_r. The input to the neural networks is time t and the activation functions for the hidden layers are $\sigma(\cdot) = tanh(\cdot)$. The outputs for each network are shown in the "Output" column. The "Output transformation" column shows whether the output from the neural network is passed through any other function. The last column, "Scaling", shows the scaling factor to be multiplied by the final output to obtain the variable in physical space. The input to the networks is time 0–60 s and scaled between $[-1, 1]$. [‡‡] $S_p(\cdot) \longrightarrow$ softplus function. $S(\cdot) \longrightarrow$ sigmoid function.

that the scaling of output is one of the important considerations for faster and an accurate convergence of the neural network. Furthermore, output transformation is another important consideration. The outputs are physical quantity and always positive. The governing equations are valid only for positive quantities (e.g. in Eq. 39, a negative p_{im} will result in negative W_{egr}). The output transformation will ensure that the predicted quantities are always positive in each epoch. Similarly, as shown in Table 5, the mask for the unknown parameters will ensure a positive value. We also observed that the unknown parameters are of different scales. The scale considered for the unknown parameters will ensure the optimization of these parameters is on the same scale. The parameters of the neural network are optimized first using Adam optimized in Tensorflow-1 with 200×10^3 epoch and further with LBFGS-B optimized. It is also important to note that we have considered self-adaptive weights in the proposed method; thus, we considered different optimizers for each set of self-adaptive weights. Further, self-adaptive weights are optimized only during the process of Adam optimization up to 100×10^3 epoch. After 100×10^3 epoch and during the process of optimization using LBFGS-B, the self-adaptive weights are considered constants with the values at 100×10^3 epoch of Adam optimization. The sizes of self-adaptive weight are 301×1 for $\boldsymbol{\lambda}_{p_{im}}$, $\boldsymbol{\lambda}_{p_{em}}$, $\boldsymbol{\lambda}_{\omega_t}$ and $\boldsymbol{\lambda}_{W_{egr}}$. The size of self adaptive weight of $\boldsymbol{\lambda}_{T_1}$ is 1×1. Softplus masks are considered for all the self-adaptive weights.

PINN for the inverse problem with four unknown parameters.

Results for Case 3 and Case 5. We first consider Case 3 and Case 5 in which we have four unknown parameters A_{egrmax}, η_{sc}, h_{tot} and A_{vgtmax}. The dynamics of p_{im}, p_{em}, ω_t and W_{egr} can be obtained from the corresponding sensor measurements. We then employ the PINN to predict the dynamics for the variables and infer the four unknowns in the system. The difference between the two cases is that in Case 3, we have considered self-adaptive weights, while in Case 5, we have not considered self-adaptive weights. We consider these two cases to study the applicability of PINN and the importance of self-adaptive weights in the present problem.

The predicted output from the neural networks, i.e., the states and T_1 and x_r are shown in Fig. 4. The predicted values of the unknown parameters are shown in Table 5. We observe that the predicted states are in good approximation with the true value in both cases. However, the predicted T_1 and x_r are not in good agreement with the true value. We study the effect of T_1 and x_r on the other variables by comparing the predicted dynamics of T_e and T_{em} (ref Eqs. (28) and (35)). We also note that T_e depends on the unknown η_{sc} and T_{em} depends on unknowns T_e and h_{tot}. The predicted dynamics of T_e and T_{em} are shown in Fig. 5b,c, respectively. We observe that both T_e and T_{em} show somewhat good agreement even T_1 and x_r do not match with the true value. The accuracy is more in Case 3 compared to Case 5. We believe that the difference in the true value and the predicted value is due to the error in the predicted value of unknown parameters. We also study the dependent variables A_{egr} and W_t of unknown A_{egrmax} and A_{vgtmax}, and are shown in Fig. 5a,d, respectively. We observe that in Case 3, the predicted dynamics for both variables show good agreement with true value. However, in Case 5, the A_{egr} does not show good agreement with true value. This is because the predicted value of A_{egrmax} has more error than Case 3.

In order to study the importance of self-adaptive weights, we study the convergence of the unknown parameters for both cases with self-adaptive weight (Case 3) and without self-adaptive weight (Case 5). The convergences of the unknown parameters with epoch for both cases are shown in Fig. 6. In Case 3 (with self-adaptive weights), we can observe that the unknown parameters converge faster and are more accurate. Furthermore, we also study the effect of different initialization of network parameters for PINN and self-adaptive weights. We run the PINN model for Case 3 and Case 5 with different initialization of parameters of PINN (DNN and unknown parameters) and self-adaptive weight keeping other hyperparameters (number of epoch considered, learning rate scheduler etc.) the same. The results for both cases are shown in Fig. 7. It is observed that for unknowns, η_{sc} and A_{vgtmax} for both cases show similar accuracy. However, for unknowns, A_{egrmax} and h_{tot}, Case 3, which is with self-adaptive weights, shows better accuracy than Case 5 (without self-adaptive weights) for all the runs. In Fig. 8, we show the self-adaptive weights for p_{im}, p_{em}, ω_t and W_{egr} after 100×10^3 epoch (constant value after

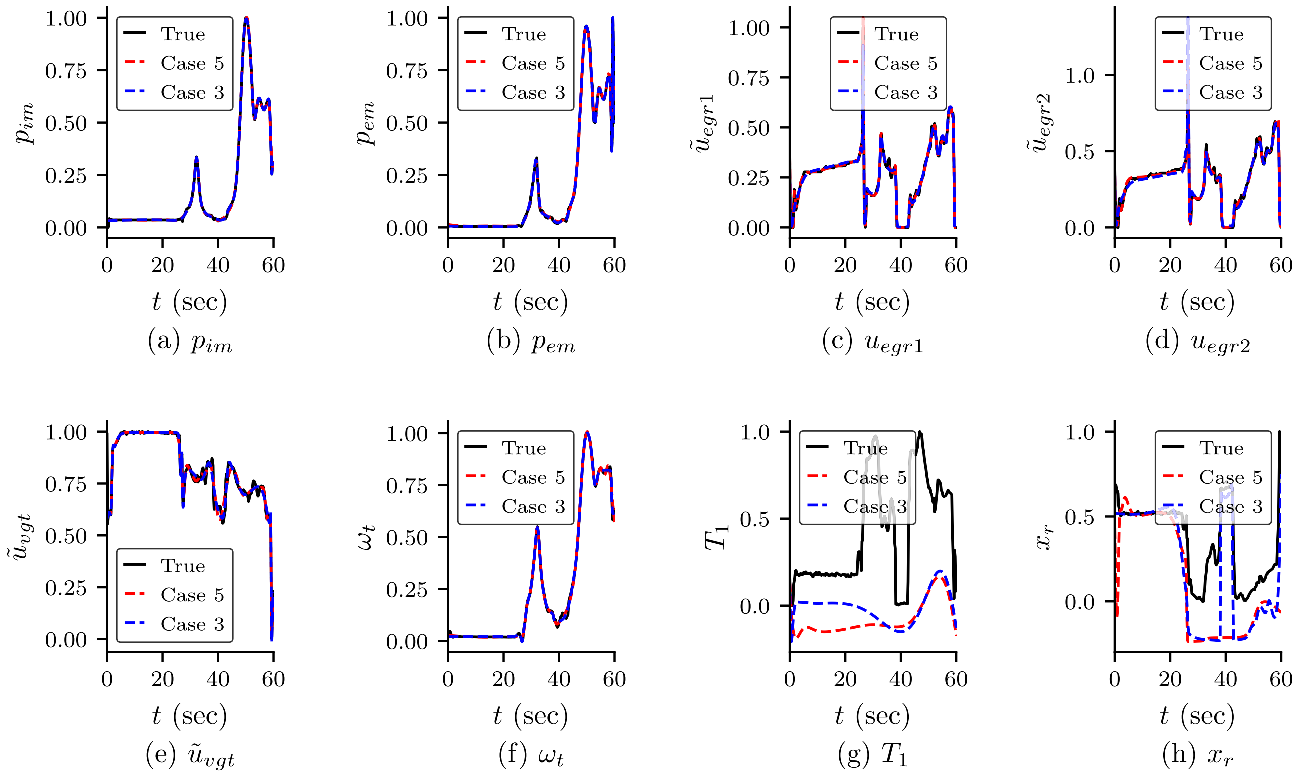


Figure 4. Predicted states and T_1 and x_r for Case 3 and Case 5: predicted dynamics of the state variables of the engine and T_r and x_1 for Case 3 (PINN with self-adaptive weights) and Case 5 (standard PINN without self-adaptive weights). The variables are scaled using Eq. (16). It can be observed that the predicted dynamics of the states are in good agreement with the true values. However, T_1 and x_r do not match with the true value. We study the dependent variables of these two variables, and are shown in Fig. 5.

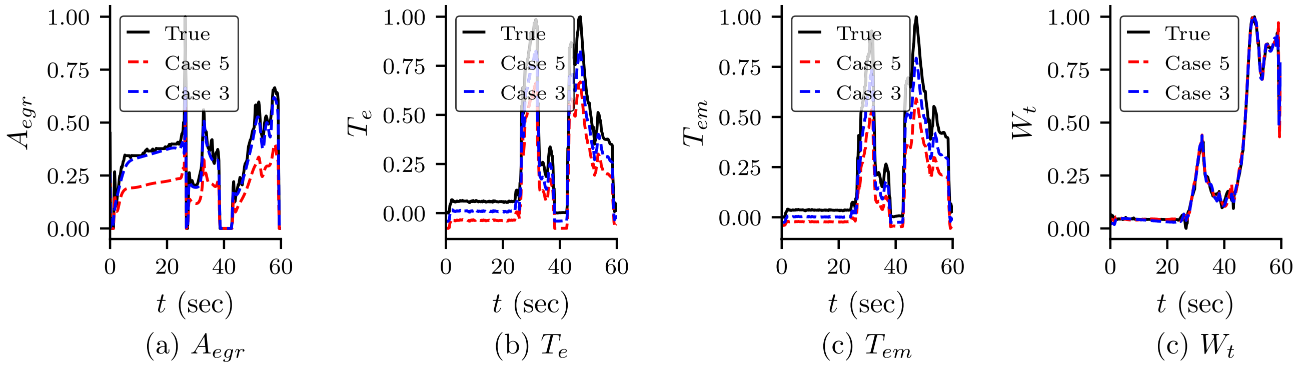


Figure 5. Predicted dynamics of dependent variables for Case 3 and Case 5: predicted dynamics of A_{egr}, T_e, T_{em} and W_t for Case 3 and Case 5. These variables depend on the unknown parameters A_{egrmax}, η_{sc}, h_{tot} and A_{vgtmax} respectively. We also note that T_e depends on T_1 and x_r.

100×10^3 epoch). Thus, we conclude that self-adaptive weights are important for better accuracy and convergence for the present problem.

Results for Case 4: four unknowns with noisy measurement data. In the previous section, we have shown the effectiveness of PINN and the importance of self-adaptive weights. In this section, we test the robustness of the proposed PINN formulation for predicting the gas flow dynamics of the diesel engine given noisy data. In particular, we are considering Case 4 (the same Case 3 but noisy measure data), in which we have four unknown parameters A_{egrmax}, η_{sc}, h_{tot} and A_{vgtmax}, with noise measurement of p_{im}, p_{em}, ω_t, W_{egr}.

We contaminate the training data p_{im}, p_{em}, ω_t, W_{egr} considered in Case 3 with Gaussian noise and consider these as synthetic field measurements. We present the predicted dynamics of the known data in Fig. 9a–d and unknown parameters in Table 5. We observe that the dynamics of the predicted p_{im}, p_{em}, ω_t, W_{egr} matches with the reference solution. However, in the case of W_{egr}, there is a small discrepancy in the predicted values near

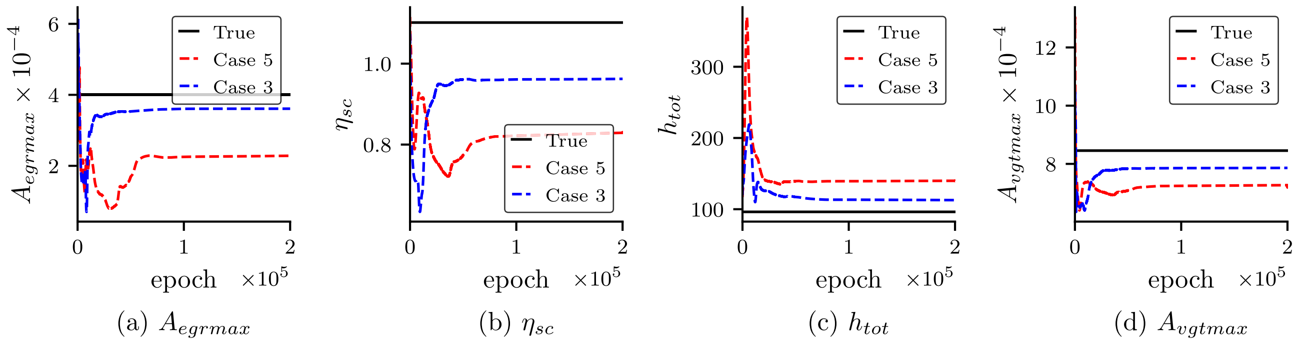


Figure 6. Convergence of the unknown parameters for Case 3 and Case 5: Convergence of the unknown parameters with epoch for Case 3 (PINN with self-adaptive weights) and Case 5 (standard PINN without self-adaptive weights). It is observed that Case 3 converges faster and also shows better accuracy.

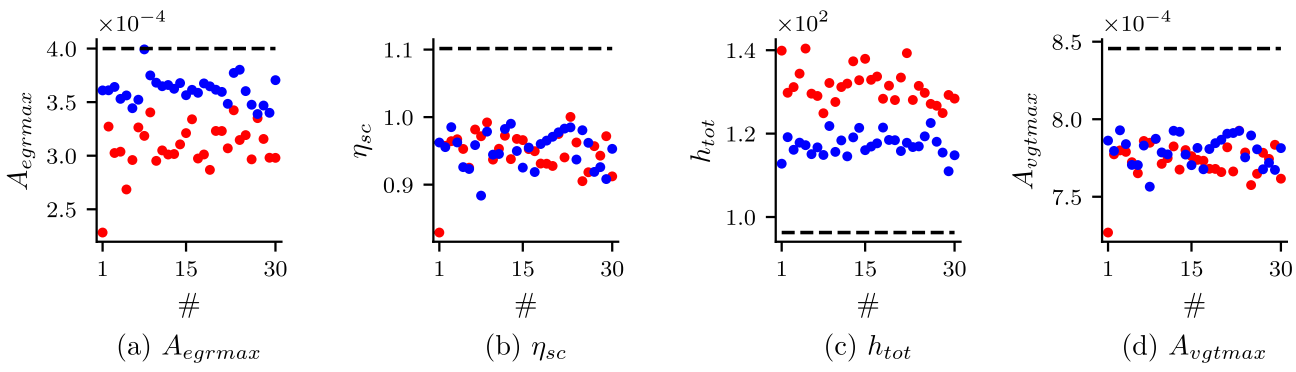


Figure 7. Predicted unknown parameters for Case 3 and Case 5: predicted unknown parameters for Case 3 (PINN with self-adaptive weights) and Case 5 (standard PINN) when prediction is made multiple times with different initialisation of the parameters of PINN, self-adaptive weights, and the unknown parameters Black dashed line → true value, Blue dots → Case 3, Red dots → Case 5.

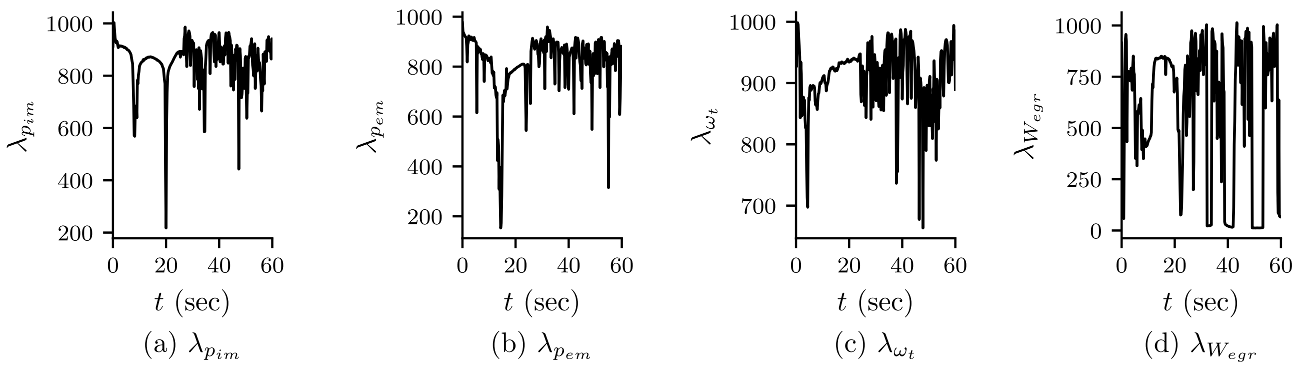


Figure 8. Self-adaptive weights for Case 3: self-adaptive wights for p_{im}, p_{em}, ω_t and W_{egr} after 100×10^3 epoch of Adam optimization. The values of the self-adaptive weights after 100×10^3 Adam optimization and LBFGS-B optimization are constant with the values of self-adaptive weight at 100×10^3 epoch.

20–25 s, which we can attribute to over-fitting caused by the noisy training data. We study the dynamics of T_{em}, W_{ei} A_{egr} and W_t on which we do not have any measured data, and these are shown in Fig. 9e–h. We observe that W_{ei} matches with the reference results. Most of the dynamics of A_{egr} and W_t match with the reference solution. The mismatch in these two variables may also be attributed to over-fitting caused by noisy data. The profile of T_{em} matches with the reference solution, however, it is not an exact match with the reference solution. This is because of the error in the predicted value of unknown parameter η_{sc} and h_{tot}. We also note that in the present study, we do not have any temperature measurements of field data. Thus, we expect errors in the predicted temperature measurements and unknown parameters. We also study the convergence of the unknown parameters with epoch and shown in Fig. 10. We note that, in this case, we consider the same hyperparameters in the optimization process. In some cases, we see over-feeting due to noisy data. This may be controlled by changing the hyperparameters, specially the learning rate for the self-adaptive weights.

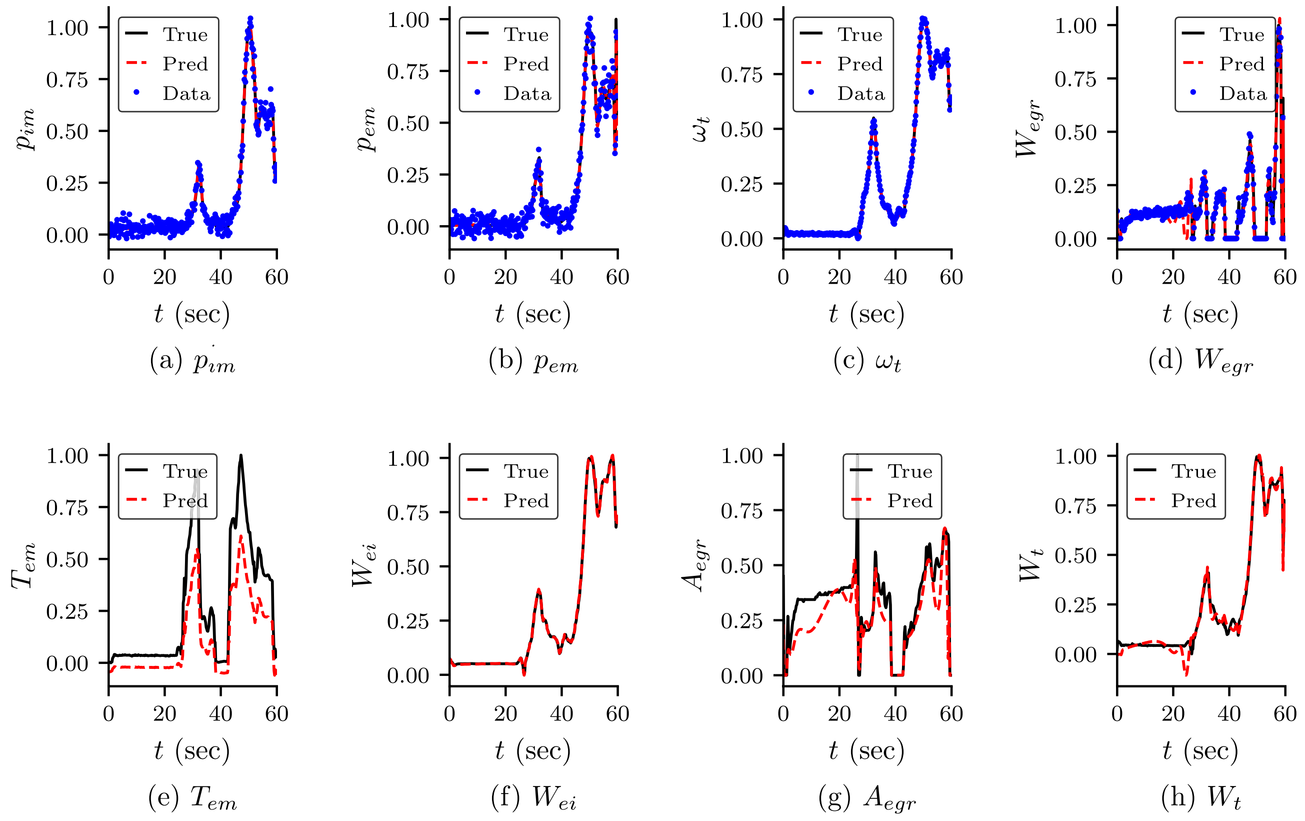


Figure 9. Predicted dynamics for variables for Case 4: predicted dynamics of (**a**–**d**) variables whose noisy field measurements are known. (**e**–**h**) dynamics of other important variables, which are also dependent on the unknown parameters. These results are for Case 4 with 4 unknown parameters and noisy field measurements.

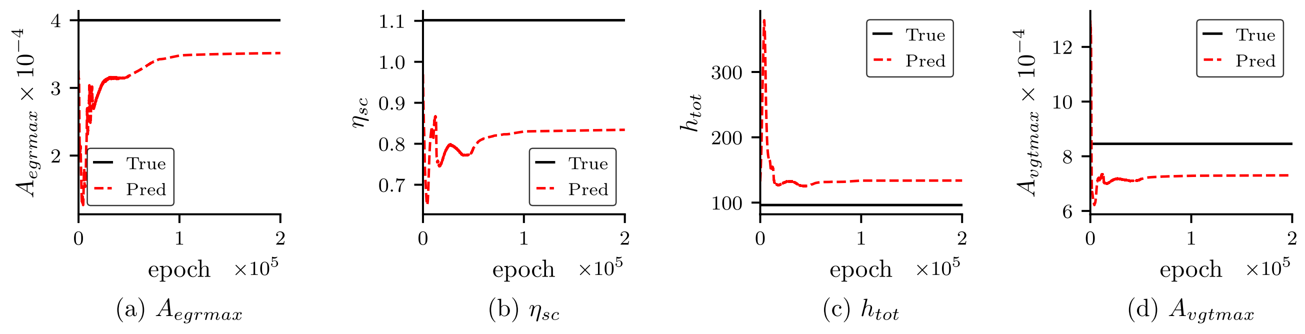


Figure 10. Convergence of the unknown parameters for Case 4: convergence of the unknown parameters with epoch for Case 4 (PINN with self-adaptive weights and noisy field data).

We also study the prediction of empirical formulae in this case and shown in Fig. 11. We observe that η_{vol} and η_c match with the reference solution. These two quantity gives the volumetric efficiency of the cylinder and the efficiency of the compressor. The other four quantities (f_{egr}, $f_{vgt} \times f_{\Pi_t}$, η_{tm}, Φ_c), also match most of its points. The discrepancy can be attributed to the noisy measurement of field data.

Summary

In this study, we proposed a PINNs-based method for estimating unknown parameters and predicting the dynamics of variables of a mean value diesel engine with VGT and EGR, given the measurement of a few of its variables. Specifically, we know field data of intake manifold pressure (p_{im}), exhaust manifold pressure (p_{em}), turbine speed (ω_t) and EGR flow (W_{egr}). We predicted the dynamics of the system variables and unknown parameters (A_{egrmax}, η_{sc}, h_{tot} and A_{vgtmax}). The input data for the study are considered from actual engine running conditions and show good accuracy in predicted results. We also studied the importance of self-adaptive weight in the accuracy and convergence of results. Furthermore, we also showed how we could approximate empirical formulas for different quantities using neural networks and train them. We believe the proposed method could be considered for an online monitoring system of diesel engines. The field-measured data are collected using

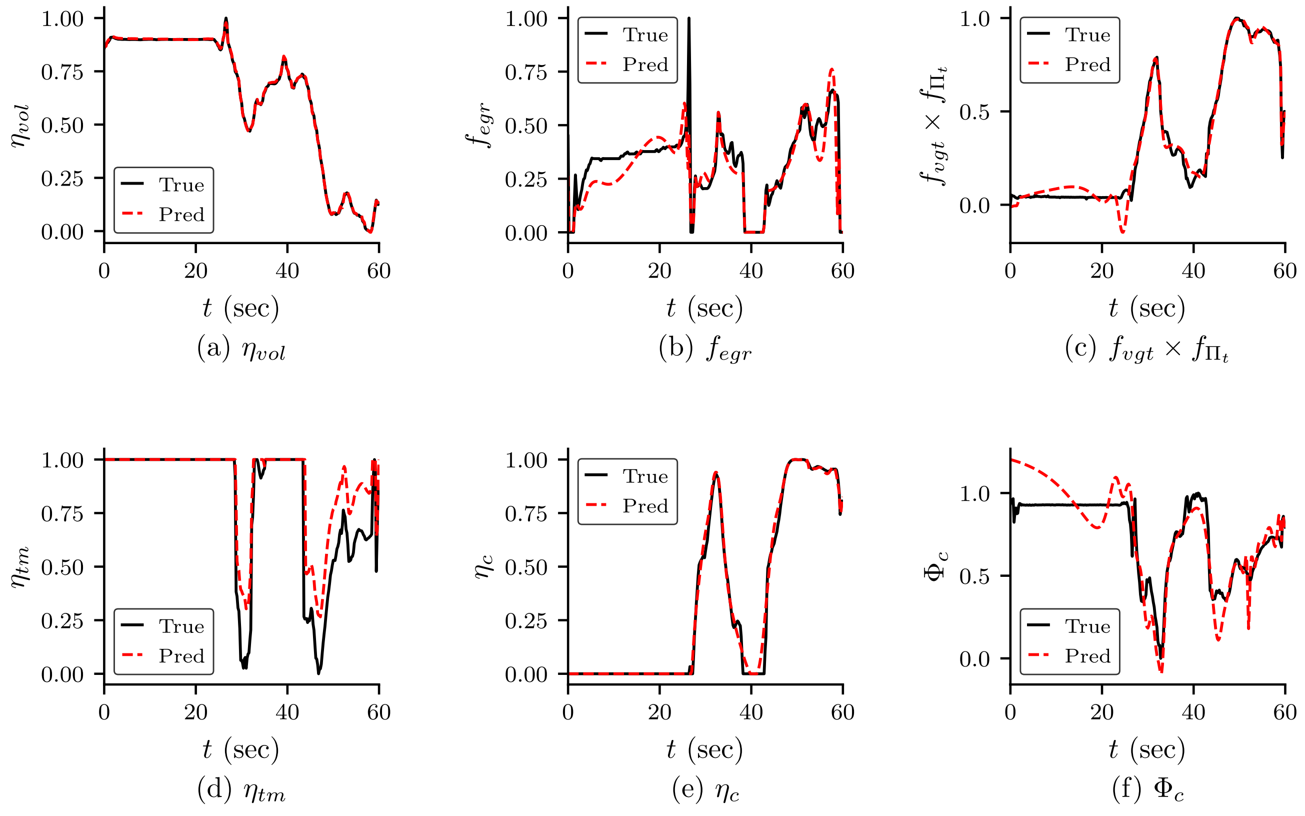


Figure 11. Empirical formulae for Case 4: the prediction of empirical formulae for Case 4 (PINN with self-adaptive weights and noisy field data).

individual sensors. Thus, in the event of sensor failure or erroneous data, the method may give erroneous results. The method also does not consider a failure of engine components, e.g. leakage in the EGR valve. We considered the engine model proposed in[1]. In the present study, we do not have any temperature measurement of field data, and thus we expect errors in predicted temperature measurements and unknown parameters as observed. Future research may include modelling of failure of engine components. Since the proposed PINN consider online training, with change in input data, field measured data or ambient condition, the PINN networks are required to train again. The accuracy of the results also depends on the size of neural networks and the optimization strategy (e.g. optimizer, learning rate scheduler) considered. For example, a large neural network or higher value in learning rate may result in overfitting of predicted results. The activation function also plays an important role in the accuracy and computational cost[24]. Further study may include a neural architecture search for optimal network sizes considering different operational ranges. Future studies may include a more robust and efficient PINN method for the problem that can be used with edge systems, including proper transfer learning strategies to reduce the computation cost. As there is noise in the measured data, the future study in this regard may also be towards uncertainty quantification of the predicted dynamics and unknown parameters.

Input data and data generations

The input data Set-I and Set-II are collected from actual engine running conditions. These data are considered to generate simulated data with different ambient conditions using the Simulink file[22] accompany[1].

Appendix 1: Note on neural network and training of PINN

In this section, we present more details on neural networks, PINN and optimization for inverse problems. As shown in Fig. 2, the neural network (FFN/DNN) takes time t as input and approximates the unknown variable y. For a DNN with $n-1$ hidden layers, the equation for the neural network can be written as,

$$\boldsymbol{y}_0 = t \qquad \textit{Input} \tag{17a}$$

$$\boldsymbol{y}_i = \sigma\left(\boldsymbol{W}_i \boldsymbol{y}_{i-1} + \boldsymbol{b}_i\right) \qquad i \;\forall\; 1 \le i \le n-1 \qquad \text{Hidden layers} \tag{17b}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{y}_n = \boldsymbol{W}_n \boldsymbol{y}_{n-1} + \boldsymbol{b}_{n-1} \qquad \text{Output layer} \tag{17c}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weights matrices and bias vectors of the network, $\sigma(\cdot)$ is an activation function, which is considered as hyperbolic tangent function in the present study. The output $\hat{y}$ is a function of input t parameterized by the weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$. We can tune the parameters of the network to predict a large number of snapshots y by minimizing a loss function using an appropriate optimization technique.

In the case of a data-driven model of neural networks, the loss function is generally considered as the mean square error (MSE) between the predicted ($\hat{y}$) and the exact value (y). On the other hand, in the case of PINN, the neural network output is made to satisfy the differential equation and the initial/boundary conditions. The derivatives of the equation are generally evaluated using automatic differentiation. As discussed in "PINNs for inverse problems in the diesel engine", the loss function is a weighted sum of physics loss which is MSE of residual and boundary/initial loss, which is MSE between predicted and exact boundary/initial value. The optimal parameters (weights and biases) of the network are obtained using an optimization method such as Adam or L-BFGS-B. In the case of an inverse problem using PINN where the objective is to predict unknown parameters (Λ) along with the variable (y). The unknown parameters are also optimized along with the network parameters. Thus, the trainable parameters are weights, biases and the unknown parameters ($\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{\Lambda}\}$). Also, an additional loss function is added, which is data loss between the predicted and the known value of y. Furthermore, in the present study we have considered self adaptive wights[9] for the loss function. The loss function is maximized with respect to the self self adaptive weights (λ). Thus, the optimization process may be written as,

$$\min_{\boldsymbol{\theta}} \ \max_{\boldsymbol{\lambda}} \ \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) \tag{18}$$

Consider the updates of a gradient descent/ascent approach to this problem

$$\boldsymbol{\theta} = \boldsymbol{\theta} - lr_{\theta} \nabla_{\theta} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) \tag{19a}$$

$$\boldsymbol{\lambda} = \boldsymbol{\lambda} + lr_{\lambda} \nabla_{\lambda} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}) \tag{19b}$$

where lr_{θ} and lr_{λ} are the learning rate associated with $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$.

In the present study, the parameters are optimized using Adam and L-BFGS-B optimizer in Tensorflow-1 (with single precision floating point). Further, self-adaptive weights are optimized only during the process of Adam optimization up to fixed epoch as discussed in "Results and discussions".

Appendix 2: Engine model

As discussed in "Problem setup", we consider a mean value engine model proposed by Wahlström and Eriksson[1] in our present study. The engine has eight states, and we have considered six in the present study. These are,

$$\boldsymbol{x} = \{p_{im}, \ \ p_{em}, \ \ \omega_t, \ \ \tilde{u}_{egr1}, \ \ \tilde{u}_{egr2}, \ \ \tilde{u}_{vgt}\} \tag{20}$$

where p_{im} and p_{em} are the intake and exhaust manifold pressure, respectively, ω_t is the turbo speed. $\tilde{u}_{egr1}$ and $\tilde{u}_{egr2}$ are the two states for the EGR actuator dynamics, and $\tilde{u}_{vgt}$ represents the VGT actuator dynamics. The control inputs for the engine are $\boldsymbol{u} = \{u_{\delta}, \ \ u_{egr}, \ \ u_{vgt}\}$ and the engine speed is n_e. Where u_{δ} is the mass of injected fuel, u_{egr} and u_{vgt} are the EGR and VGT valve positions, respectively. The mean value engine model is then expressed as

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, n_e). \tag{21}$$

The engine model consists of 6 parts intake and exhaust manifold, the cylinder, the turbine, EGR valve system, and the compressor system. A schematic diagram of the engine is shown in Fig. 1. In this section, we briefly discuss the equation required for the present study and these are taken from[1]. For detail of the engine model, the interested reader may refer to Wahlström and Eriksson[1].

Manifold pressures.

The pressure at the intake manifold (p_{im}) is modelled using a first-order differential equation as,

$$\frac{d}{dt} p_{im} = \frac{R_a T_{im}}{V_{im}} \left(W_c + W_{egr} - W_{ei} \right) \tag{22}$$

where T_{im} and V_{im} are the temperature and volume of the intake manifold, respectively, and both are assumed to be constant, W_c, W_{egr} and W_{ei} are the compressor mass flow, EGR mass flow and total mass flow, respectively. The ideal gas constant and specific heat capacity of the air are R_a and γ_a, respectively.

Similarly, the exhaust manifold pressure p_{em} is modelled as,

$$\frac{d}{dt} p_{em} = \frac{R_e T_{em}}{V_{em}} \left(W_{eo} - W_t - W_{egr} \right) \tag{23}$$

where R_e is the ideal gas constant of the exhaust gas with specific heat capacity γ_e, T_{em} and V_{em} are the exhaust manifold temperature, and volume, W_{eo}, W_t are the mass flow out from the cylinder and turbine mass flow, respectively.

Cylinder.

The total mass flow from the intake manifold to the cylinder W_{ei}, and the total mass flow out of the cylinder W_{eo} are modelled as,

$$W_{ei} = \frac{\eta_{vol} p_{im} n_e V_d}{120 R_a T_{im}} \tag{24}$$

$$W_{eo} = W_f + W_{ei} \tag{25}$$

where W_f is the fuel mass flow into the cylinder is given by,

$$W_f = \frac{10^{-6}}{120} u_\delta n_e n_{cyl}, \tag{26}$$

V_d, n_e and n_{cyl} are the displaced volume, engine speed and the number of cylinders, respectively. The volumetric efficiency, η_{vol} of the cylinder may be modelled as

$$\eta_{vol} = c_{vol1}\sqrt{p_{im}} + c_{vol2}\sqrt{n_e} + c_{vol3} \tag{27}$$

where c_{vol1}, c_{vol2} and c_{vol3} are constant.

The temperature at cylinder out based upon ideal-gas Seliger cycle (or limited pressure cycle) and given as,

$$T_e = \eta_{sc}\Pi_e^{1-1/\gamma_a} r_c^{1-\gamma_a} x_p^{1/\gamma_a - 1}\left[q_{in}\left(\frac{1-x_{cv}}{c_{pa}} + \frac{x_{cv}}{c_{Va}}\right) + T_1 r_c^{\gamma_a - 1}\right] \tag{28}$$

where the pressure ratio (Π_e) over the cylinder is ratio of pressure at exhaust (p_{em}) and intake (p_{im}),

$$\Pi_e = \frac{p_{em}}{p_{im}}, \tag{29}$$

the fuel consumed during constant-volume combustion is x_{cv} and fuel consumed during constant pressure combustion is $1 - x_{cv}$, η_{sc} and r_c are compensation factor for non-ideal cycles and compression ratio. The temperature, T_1 when the inlet valve closes and after the intake stroke and mixing is given by,

$$T_1 = x_r T_e + (1 - x_r)T_{im} \tag{30}$$

The residual gas fraction (x_r) is model as

$$x_r = \frac{\Pi_e^{1/\gamma_a} x_p^{-1/\gamma_a}}{r_c x_v} \tag{31}$$

The pressure ratio (x_p) and the volume ratio (x_v) in the Seliger cycle between point 3 (after combustion) and point 2 (before combustion) are modelled as,

$$x_p = \frac{p_3}{p_2} = 1 + \frac{q_{in} x_{cv}}{c_{Va} T_1 r_c^{\gamma_a - 1}} \tag{32}$$

$$x_v = \frac{v_3}{v_2} = 1 + \frac{q_{in}(1 - x_{cv})}{c_{pa}\left[(q_{in} x_{cv}/c_{Va}) + T_1 r_c^{\gamma_a - 1}\right]} \tag{33}$$

where the specific energy constant of the charge is modelled as

$$q_{in} = \frac{W_f q_{HV}}{W_{ei} + Wf}(1 - x_r) \tag{34}$$

The temperature at cylinder out modelled in Eq. (28) is the temperature at the cylinder exit. However, it is not the same as the temperature as the exhaust manifold. This is due to the heat loss in the exhaust pipes between the cylinder and the exhaust manifold. The exhaust manifold temperature (T_{em}) is given as

$$T_{em} = T_{amb} + (T_e - T_{amb})\exp\left(\frac{-h_{tot}\pi d_{pipe} l_{pipe} n_{pipe}}{W_{eo} c_{pe}}\right) \tag{35}$$

where T_{amb} is the ambient temperature, d_{pipe}, l_{pipe} and n_{pipe} are the pipe diameter, pipe length and the number of pipes, respectively.

EGR valve. The actuator dynamics of the EGR-valve are modelled as,

$$\frac{d}{dt}\tilde{u}_{egr1} = \frac{1}{\tau_{egr1}}\left[u_{egr}(t - \tau_{degr}) - \tilde{u}_{egr1}\right] \tag{36}$$

$$\frac{d}{dt}\tilde{u}_{egr2} = \frac{1}{\tau_{egr2}}\left[u_{egr}(t - \tau_{degr}) - \tilde{u}_{egr2}\right] \tag{37}$$

$$\tilde{u}_{egr} = K_{egr}\tilde{u}_{egr1} - (K_{egr} - 1)\tilde{u}_{egr2} \tag{38}$$

where τ_{egr1}, τ_{egr2} are time constants, τ_{degr} is the time delay and K_{egr} is a constant that affect the overshoot.

We model the mass flow through the EGR valve through the restriction ($p_{em} < p_{im}$) as,

$$W_{egr} = \frac{A_{egr} p_{im} \Psi_{egr}}{\sqrt{T_{em} R_e}} \tag{39}$$

where Ψ_{egr} is a parabolic function

$$\Psi_{egr} = 1 - \left(\frac{1 - \Pi_{egr}}{1 - \Pi_{egropt}} - 1 \right)^2 \tag{40}$$

The effective area is modelled as,

$$A_{egr} = A_{egrmax} f_{egr}(\tilde{u}_{egr}) \tag{41}$$

When the sonic conditions are reached (flow is choked) in the throat and when no backflow can occur ($1 < p_{im}/p_{em}$), the pressure ratio Π_{egr} over the valve is limited and modelled as,

$$\Pi_{egr} = \begin{cases} \Pi_{egropt} & \text{if } \frac{p_{im}}{p_{em}} < \Pi_{egropt} \\ \frac{p_{im}}{p_{em}} & \text{if } \Pi_{egropt} \le \frac{p_{im}}{p_{em}} \le 1 \\ 1 & \text{if } 1 < \frac{p_{im}}{p_{em}} \end{cases} \tag{42}$$

A_{egrmax}, Π_{egropt} are constant and $f_{egr}(\tilde{u}_{egr})$ is modelled as a polynomial function,

$$f_{egr}(\tilde{u}_{egr}) = \begin{cases} c_{egr1} \tilde{u}_{egr}^2 + c_{egr2} \tilde{u}_{egr} + c_{egr3} & \text{if } \tilde{u}_{egr} \le -\frac{c_{egr2}}{2c_{egr1}} \\ c_{egr3} - \frac{c_{egr2}^2}{4c_{egr1}} & \text{if } \tilde{u}_{egr} > -\frac{c_{egr2}}{2c_{egr1}} \end{cases} \tag{43}$$

where c_{egr1}, c_{egr2} and c_{egr3} are constant.

Turbocharger. The turbo speed, ω_t is modelled as a first-order differential model as,

$$\frac{d}{dt}\omega_t = \frac{P_t \eta_m - P_c}{J_t \omega_t} \tag{44}$$

where J_t is the inertia, P_t and P_c are the power delivered by the turbine and power required to drive the compressor, respectively, η_m is the mechanical efficiency of the turbocharger.

The VGT actuator system is modelled as a first-order system

$$\frac{d\tilde{u}_{vgt}}{dt} = \frac{1}{\tau_{vgt}} \left[u_{vgt}(t - \tau_{dvgt}) - \tilde{u}_{vgt} \right] \tag{45}$$

where τ_{vgt} and τ_{dvgt} are the time constant and time delay respectively.

The turbine mass flow (W_t) is calculated using

$$W_t = \frac{A_{vgtmax} p_{em} f_{\Pi_t}(\Pi_t) f_{vgt}(\tilde{u}_{vgt}))}{\sqrt{T_{em} R_e}} \tag{46}$$

A_{vgtmax} is the maximum area in the turbine that the gas flow through.

$$f_{\Pi_t}(\Pi_t) = \sqrt{1 - \Pi_t^{K_t}} \tag{47}$$

where K_t a constant and $\Pi_t = p_{es}/p_{em}$. $p_{es} > p_{amb}$ if there is a restriction like an after-treatment system. However, in the model we consider, there is no restriction after the turbine, thus

$$\Pi_t = \frac{p_{amb}}{p_{em}} \tag{48}$$

Further, with the increase in VGT control signal (u_{vgt}), the effective area increases and thus also increases the flow. The effective area of the VGT $f_{vgt}(\tilde{u}_{vgt})$ is modelled as an ellipse

$$\left[\frac{f_{vgt}(\tilde{u}_{vgt}) - c_{f2}}{c_{f1}} \right]^2 + \left[\frac{\tilde{u}_{vgt} - c_{vgt2}}{c_{vgt1}} \right]^2 = 1 \tag{49}$$

which is

$$f_{vgt}(\tilde{u}_{vgt}) = c_{f2} + c_{f1} \sqrt{\max\left(0, 1 - \left(\frac{\tilde{u}_{vgt} - c_{vgt2}}{c_{vgt1}}\right)^2\right)} \tag{50}$$

The power delivered by the turbine, P_t and the mechanical efficiency of the turbocharger η_m are modelled as,

$$P_t\eta_m = \eta_{tm} W_t c_{pe} T_{em}\left(1 - \Pi_t^{1-1/\gamma_e}\right) \tag{51}$$

$$\eta_{tm} = \eta_{tm,max} - c_m(BSR - BSR_{opt})^2 \tag{52}$$

where the blade speed ratio (BSR) is defined as the ratio of the turbine blade tip speed to the speed which a gas reaches when expanded entropically at the given pressure ratio Π_t.

$$BSR = \frac{R_t\omega_t}{\sqrt{2c_{pe}T_{em}(1 - \Pi_t^{1-1/\gamma_e})}} \tag{53}$$

where R_t is the turbine blade radius, and

$$c_m = c_{m1}[max(0, \omega_t - c_{m2}]^{c_{m3}} \tag{54}$$

Compressor. The compressor model consists of two models: the compressor efficiency model and the compressor mass flow model. The compressor efficiency is defined as the ratio of the power from the isentropic process ($P_{c,s}$) to the compressor power (P_c)

$$\eta_c = \frac{P_{c,s}}{P_c} = \frac{T_{amb}(\Pi_c^{1-1/\gamma_a} - 1)}{T_c - T_{amb}} \tag{55}$$

where T_c is the temperature after the compressor, and the pressure ratio is given by,

$$\Pi_c = \frac{p_{im}}{p_{amb}} \tag{56}$$

The power from the isentropic process is given as,

$$P_{c,s} = W_c c_{pa} T_{amb}(\Pi_c^{1-1/\gamma_a} - 1) \tag{57}$$

where W_c is the compressor mass flow and c_{pa} is a constant. Thus, the compressor power can be modelled from Eqs. (55) and (57) as

$$P_c = \frac{P_{c,s}}{\eta_c} = \frac{W_c c_{pa} T_{amb}}{\eta_c}(\Pi_c^{1-1/\gamma_a - 1}) \tag{58}$$

η_c is modelled as an ellipses, which depends on the pressure ratio (Π_c) and compressor mass flow (W_c),

$$\eta_c = \eta_{cmax} - \boldsymbol{\mathcal{X}}^T \mathbf{Q}_c \boldsymbol{\mathcal{X}} \tag{59}$$

where $\boldsymbol{\mathcal{X}}$ is a vector and given as,

$$\boldsymbol{\mathcal{X}} = \begin{bmatrix} W_c - W_{copt} \\ \pi_c - \pi_{copt} \end{bmatrix} \tag{60}$$

where W_{copt} and π_{copt} are the optimum values of W_c and π_c respectively. π_c is a non linear transformation of Π_c as

$$\pi_c = (\Pi_c - 1)^{c_\pi} \tag{61}$$

and $\boldsymbol{Q}_c$ is a semi-definite matrix

$$\mathbf{Q}_c = \begin{bmatrix} a_1 & a_3 \\ a_3 & a_2 \end{bmatrix} \tag{62}$$

The model for compressor mass flow, W_c is modelled using two non-dimensional variables: energy transfer coefficient (Ψ_c) and volumetric flow coefficient (Φ_c). The energy transfer coefficient is defined as,

$$\Psi_c = \frac{2c_{pa}T_{amb}(\Pi_c^{1-1/\gamma_a} - 1)}{R_c^2\omega_t^2} \tag{63}$$

where R_c is the compressor blade ratio. The volumetric flow coefficient is defined as,

$$\Phi_c = \frac{W_c/\rho_{amb}}{\pi R_c^3\omega_t} = \frac{R_a T_{amb}}{p_{amb}\pi R_c^3\omega_t} W_c \tag{64}$$

The energy transfer coefficient (Ψ_c) and volumetric flow coefficient (Φ_c) can be described by a part of an ellipse,

$$c_{\Psi 1}(\omega_t)(\Psi_c - c_{\Psi 2})^2 + c_{\Phi 1}(\omega_t)(\Phi_c - c_{\Phi 2})^2 = 1 \tag{65}$$

where $c_{\Psi 1}$ and $c_{\Phi 1}$ are function of turbine speed (ω_t) and modelled as a second order polynomial as,

$$c_{\Psi 1}(\omega_t) = c_{\omega_{\Psi 1}}\omega_t^2 + c_{\omega_{\Psi 2}}\omega_t + c_{\omega_{\Psi 3}} \tag{66}$$

$$c_{\Phi 1}(\omega_t) = c_{\omega_{\Phi 1}}\omega_t^2 + c_{\omega_{\Phi 2}}\omega_t + c_{\omega_{\Phi 3}} \tag{67}$$

Solving Eq. (65) for Φ_c and Eq. (64) for W_c, the compressor mass flow is given as,

$$W_c = \frac{p_{amb}\pi R_c^3 \omega_t}{R_a T_{amb}}\Phi_c \tag{68}$$

$$\Phi_c = \sqrt{\max\left(0, \frac{1 - c_{\psi 1}(\Psi_c - c_{\Psi 2})^2}{c_{\Phi 1}}\right)} + c_{\Phi 2} \tag{69}$$

where $c_{\omega_{\Psi 1}}$, $c_{\omega_{\Psi 2}}$, $c_{\omega_{\Psi 3}}$, $c_{\omega_{\Phi 1}}$, $c_{\omega_{\Phi 2}}$, $c_{\omega_{\Phi 3}}$, $c_{\Phi 2}$ and $c_{\Psi 2}$ are constant.

Appendix 3: Brief discussion on lab test data

The engine model considered in this study uses empirical formulae. These equations are engine specific and may not be appropriate for the present study. As discussed in "Neural network surrogates for empirical formulae", we consider surrogate neural networks and uses lab test data to train these model. In this section, we briefly discuss lab test data.

Practical limitations exist when instrumenting engines for testing. Some physical phenomena are easily measurable, while others are not. When conducting modelling efforts, one must consider the necessary measurements for model tuning to ensure the experimental setup is adequate. The data collection capabilities can also impact loss function weights based on data trustworthiness, as well as noise values applied in the analysis. There are a few signals that pose particular challenges in cost-effective and simple measurement in part due to the high temperature, pressure, dynamics and flow constituents in some areas.

Often as areas closer to the cylinder are considered, measurements become increasingly difficult. For example, exhaust port flow, W_{eo} is difficult to measure directly, as the gas is very hot and reactive. In-cylinder measurements are limited by high pressure and temperatures, requiring specialized equipment. Even measuring charge flows directly can be challenging. Because of these limitations, care must be taken in the experimental methods and analysis design to ensure enough data is gathered to be able to observe and identify the system. Sometimes steady state characterizations are used to obtain a static characterization. Consider volumetric efficiency as an example: because measuring flow directly into or out of the cylinder is difficult, a fresh air flow measurement combined with an EGR flow measurement can be used to estimate charge flow to enable the calculation of volumetric efficiency. However, any intake, EGR, or Exhaust leak impacts this measurement, as does the tolerance stack up of both measurements.

Appendix 4: Calculation of labelled data for training of the neural network for the empirical formulae

We approximate the empirical formula using surrogate neural networks and are discussed in "Neural network surrogates for empirical formulae". The lab test data required for calculating each of these quantities are shown in Table 3. In this section, we discuss the calculation of labelled data from lab-test data. The functional approximation of the empirical formulae is independent of time; thus, static data may be considered for the calculation of labelled data. However, in the case of calculation of labelled for η_{tm}, the differential equation Eq. (44) is considered. Thus, we consider dynamic data for this calculation.

We approximate the volumetric efficiency using a surrogate neural network $(\boldsymbol{\mathcal{N}}_1^{(P)}(:, \boldsymbol{\theta})_1^P)$. The inputs to the network are intake manifold pressure (p_{im}) and engine speed (n_e) and trained using labelled data of η_{vol}. The labelled η_{vol} are calculated from measurement of W_{ei} using Eq. (24),

$$\eta_{vol} = \frac{120 R_a T_{im} W_{ei}}{p_{im} n_e V_d} \tag{70}$$

The effective area ratio function for EGR valve is approximated using a surrogate neural network $(\boldsymbol{\mathcal{N}}_2^{(P)}(:, \boldsymbol{\theta})_2^P)$. The input to the network is $\tilde{u}_{egr}$ and trained using labelled data of f_{egr}. The labelled f_{egr} are calculated from the measurement of W_{egr} using Eqs. (39) and (41),

$$A_{egr} = \frac{\sqrt{T_{em} R_e}}{p_{im}\Psi_{egr}} W_{egr} \tag{71}$$

$$f_{egr} = \frac{A_{egr}}{A_{egrmax}} \tag{72}$$

It is important to node that the value of Ψ_{egr} varies from 0 to 1 (Eq. 40), thus in the calculation of A_{egr}, a situation may occurs where division by 0. This situation occurs when $p_{em} < p_{im}$ (Eq. 42). In order to avoid this, labelled data are calculated only for $\Psi_{egr} > 10^{-15}$.

The neural network approximating $(\boldsymbol{\mathcal{N}}_3^{(P)}(:, \boldsymbol{\theta})_3^P)$ for $F_{vgt,\Pi} = f_{vgt} \times f_{\Pi_t}$ is trained using labelled data which are calculated from the measurement of turbine mass flow (W_t) using Eq. (46),

$$F_{vgt,\Pi_t}(\tilde{u}_{vgt}, \Pi_t) = f_{vgt}(\tilde{u}_{vgt}) \times f_{\Pi_t}(\Pi_t) = \frac{W_t\sqrt{T_{em}R_e}}{A_{vgtmax}p_{em}} \tag{73}$$

The training for the neural network $(\boldsymbol{\mathcal{N}}_4^{(P)}(:, \boldsymbol{\theta})_4^P)$ for the surrogate model of turbine mechanical efficiency (η_{tm}) is done using labelled η_{tm} which is calculated from the measurement of ω_t using Eqs. (44) and (51)

$$P_t\eta_m = P_c + J_t\omega_t\frac{d\omega_t}{dt} \tag{74}$$

The compressor power (P_c) is calculated as,

$$P_c = W_c c_{pa}(T_c - T_{amb}) \tag{75}$$

In the present study, we consider a five-point method to approximate the derivative present in Eq. (74).

$$f^{(1)}(x) \approx \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} \tag{76}$$

Once $P_t\eta_m$ calculated from Eq. (74), the labelled η_{tm} are calculated using Eq. (51)

$$\eta_{tm} = \frac{P_t\eta_m}{W_t c_{pe} T_{em}\left(1 - \Pi_t^{1-1/\gamma_e}\right)} \tag{77}$$

The values of η_{tm} are restricted to maximum value $\eta_{tm,max}$

$$\eta_{tm} = \min(\eta_{tm,max}, \eta_{tm}), \qquad \eta_{tm,max} = 0.8180 \tag{78}$$

The labelled data for the training of neural network $(\boldsymbol{\mathcal{N}}_5^{(P)}(:, \boldsymbol{\theta})_5^P)$ for compressor efficiency (η_c) is calculated using Eqs. (58) and (55)

$$\eta_c = \frac{P_{c,s}}{P_c} \tag{79a}$$

$$= \frac{W_c c_{pa} T_{amb}\left(\Pi_c^{1-1/\gamma_a} - 1\right)}{W_c c_{pa}(T_c - T_{amb})} \tag{79b}$$

$$= \frac{T_{amb}\left(\Pi_c^{1-1/\gamma_a} - 1\right)}{T_c - T_{amb}} \tag{79c}$$

To avoid any division by 0, the value of $T_c - T_{amb}$ less than 10^{-6} are considered as 10^{-6}. Further, the value of η_c is clipped between 0.2 and η_{cmax}

$$\eta_c = \max(0.2, \eta_c) \tag{80a}$$

$$\eta_c = \min(\eta_{cmax}, \eta_c), \qquad \eta_{cmax} = 0.7364 \tag{80b}$$

The training for the neural network $(\boldsymbol{\mathcal{N}}_6^{(P)}(:, \boldsymbol{\theta})_6^P)$ for surrogate model of volumetric flow coefficient (Φ_c) is done using labelled data which are calculated from the measurement of compressor mass flow (W_c) Eq. (64)

$$\Phi_c = \frac{R_a T_{amb}}{p_{amb}\pi R_c^3\omega_t} W_c \tag{81}$$

Appendix 5: Detail loss function for the PINNs model for the engine

We consider the following loss function for Case 1 to Case 4, which have self-adaptive weights in the loss function,

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Lambda}, \boldsymbol{\lambda}_{p_{im}}, \boldsymbol{\lambda}_{p_{em}}, \boldsymbol{\lambda}_{\omega_t}, \boldsymbol{\lambda}_{W_{egr}}, \lambda_{T_1}) = {} & \mathcal{L}_{p_{im}} + \mathcal{L}_{p_{em}} + \mathcal{L}_{\omega_t} + \mathcal{L}_{u_{egr1}} \\ & + \mathcal{L}_{u_{egr2}} + \mathcal{L}_{u_{vgt}} + 10 \times \mathcal{L}_{x_r} + \lambda_{T_1} \times \mathcal{L}_{T_1} \\ & + \mathcal{L}^{ini}_{p_{im}} + \mathcal{L}^{ini}_{p_{em}} + \mathcal{L}^{ini}_{\omega_t} + \mathcal{L}^{ini}_{\tilde{u}_{egr1}} \\ & + \mathcal{L}^{ini}_{\tilde{u}_{egr2}} + \mathcal{L}^{ini}_{\tilde{u}_{vgt}} + \mathcal{L}^{ini}_{x_r} + 100 \times \mathcal{L}^{ini}_{T_1} \\ & + \mathcal{L}^{data}_{p_{im}}(\boldsymbol{\lambda}_{p_{im}}) + \mathcal{L}^{data}_{p_{em}}(\boldsymbol{\lambda}_{p_{em}}) \\ & + \mathcal{L}^{data}_{\omega_t}(\boldsymbol{\lambda}_{\omega_t}) + \mathcal{L}^{data}_{W_{egr}}(\boldsymbol{\lambda}_{W_{egr}}),\end{aligned} \tag{82}$$

In the Case 5 where we have not considered self-adaptive weights, the loss function is given as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\Lambda}) = {} & \mathcal{L}_{p_{im}} + \mathcal{L}_{p_{em}} + \mathcal{L}_{\omega_t} + \mathcal{L}_{u_{egr1}} \\ & + \mathcal{L}_{u_{egr2}} + \mathcal{L}_{u_{vgt}} + 10 \times \mathcal{L}_{x_r} + 10^3 \times \mathcal{L}_{T_1} \\ & + \mathcal{L}^{ini}_{p_{im}} + \mathcal{L}^{ini}_{p_{em}} + \mathcal{L}^{ini}_{\omega_t} + \mathcal{L}^{ini}_{\tilde{u}_{egr1}} \\ & + \mathcal{L}^{ini}_{\tilde{u}_{egr2}} + \mathcal{L}^{ini}_{\tilde{u}_{vgt}} + \mathcal{L}^{ini}_{x_r} + 100 \times \mathcal{L}^{ini}_{T_1} \\ & + 10^3 \times \mathcal{L}^{data}_{p_{im}} + 10^3 \times \mathcal{L}^{data}_{p_{em}} \\ & + 10^3 \times \mathcal{L}^{data}_{\omega_t} + 10^3 \times \mathcal{L}^{data}_{W_{egr}},\end{aligned} \tag{83}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_6)$ are the hyperparameters of all NNs in PINNs, which include both weights and biases, $\boldsymbol{\Lambda}$ are the unknown parameters of the equations which need to be found out. $\boldsymbol{\lambda}_{p_{im}}$, $\boldsymbol{\lambda}_{p_{em}}$, $\boldsymbol{\lambda}_{\omega_t}$ and $\boldsymbol{\lambda}_{W_{egr}}$ are the self-adaptive weight[9] for the data loss in p_{im}, p_{em}, ω_t and W_{egr} respectively. λ_{T_1} is the self-adaptive weight for physics loss in T_1. $\mathcal{L}_{p_{im}}(\boldsymbol{\theta})$, $\mathcal{L}_{p_{em}}(\boldsymbol{\theta})$, $\mathcal{L}_{\omega_t}$, $\mathcal{L}_{\tilde{u}_{egr1}}$, $\mathcal{L}_{\tilde{u}_{egr2}}$, and $\mathcal{L}_{\tilde{u}_{vgt}}$ are the physics loss corresponding to the differential equations of the states of the diesel engine p_{im}, p_{em}, ω_t $\tilde{u}_{egr1}$, $\tilde{u}_{egr2}$ and $\tilde{u}_{vgt}$ respectively. $\mathcal{L}_{x_r}$ and $\mathcal{L}_{T_1}$ are the physics loss correspond to x_r and T_1 respectively.

$$\mathcal{L}_{p_{im}} = \frac{1}{n}\sum_{i=1}^{n} r(p_{im})^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{dp_{im}}{dt} - \frac{R_a T_{im}}{V_{im}}\left(W_c + W_{egr} - W_{ei}\right)\right)^2 \tag{84a}$$

$$\mathcal{L}_{p_{em}} = \frac{1}{n}\sum_{i=1}^{n} r(p_{em})^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{dp_{em}}{dt} - \frac{R_e T_{em}}{V_{em}}\left(W_{eo} - W_t - W_{egr}\right)\right)^2 \tag{84b}$$

$$\mathcal{L}_{\omega_t} = \frac{1}{n}\sum_{i=1}^{n} r(\omega_t)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{d\omega_t}{dt} - \frac{P_t\eta_m - P_c}{J_t\omega_t}\right)^2 \tag{84c}$$

$$\mathcal{L}_{\tilde{u}_{egr1}} = \frac{1}{n}\sum_{i=1}^{n} r(\tilde{u}_{egr1})^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{d\tilde{u}_{egr1}}{dt} - \frac{1}{\tau_{egr1}}\left[u_{egr}(t - \tau_{degr}) - \tilde{u}_{egr1}\right]\right)^2 \tag{84d}$$

$$\mathcal{L}_{\tilde{u}_{egr2}} = \frac{1}{n}\sum_{i=1}^{n} r(\tilde{u}_{egr2})^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{d\tilde{u}_{egr2}}{dt} - \frac{1}{\tau_{egr2}}\left[u_{egr}(t - \tau_{degr}) - \tilde{u}_{egr2}\right]\right)^2 \tag{84e}$$

$$\mathcal{L}_{\tilde{u}_{vgt}} = \frac{1}{n}\sum_{i=1}^{n} r(\tilde{u}_{vgt})^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{d\tilde{u}_{vgt}}{dt} - \frac{1}{\tau_{vgt}}\left[u_{vgt}(t - \tau_{dvgt}) - \tilde{u}_{vgt}\right]\right)^2 \tag{84f}$$

$$\mathcal{L}_{x_r} = \frac{1}{n}\sum_{i=1}^{n} r(x_r)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_r - \frac{\Pi_e^{1/\gamma_a} x_p^{-1/\gamma_a}}{r_c x_v}\right)^2 \tag{84g}$$

$$\mathcal{L}_{T_1} = \frac{1}{n}\sum_{i=1}^{n} r(T_1)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(T_1 - (x_r T_e + (1 - x_r)T_{im})\right)^2 \tag{84h}$$

where n and $r(\cdot)$ are the number of residual points and residual, respectively.

In the Case 1 to Case 4, $\mathcal{L}^{data}_{p_{im}}(\boldsymbol{\lambda}_{p_{im}}$, $\mathcal{L}^{data}_{p_{em}}(\boldsymbol{\lambda}_{p_{em}})$, $\mathcal{L}^{data}_{\omega_t}(\boldsymbol{\lambda}_{\omega_t})$, and $\mathcal{L}^{data}_{W_{egr}}(\boldsymbol{\lambda}_{W_{egr}})$ are the data loss in p_{im}, p_{em}, ω_t and W_{egr} respectively and defined as,

$$\mathcal{L}^{data}_{p_{im}}(\boldsymbol{\lambda}_{p_{im}}) = \frac{1}{n}\sum_{j=1}^{n}\left[\left(p^{(j)}_{im_{data}} - \hat{p}^{(j)}_{im_{\mathcal{N}_1}}\right)\lambda^{(j)}_{p_{im}}\right]^2 \tag{85a}$$

$$\mathcal{L}^{data}_{p_{em}}(\boldsymbol{\lambda}_{p_{em}}) = \frac{1}{n}\sum_{j=1}^{n}\left[\left(p^{(j)}_{em_{data}} - \hat{p}^{(j)}_{em_{\mathcal{N}_1}}\right)\lambda^{(j)}_{p_{em}}\right]^2 \tag{85b}$$

$$\mathcal{L}^{data}_{\omega_t}(\boldsymbol{\lambda}_{\omega_t}) = \frac{1}{n}\sum_{j=1}^{n}\left[\left(\omega^{(j)}_{t_{data}} - \hat{\omega}^{(j)}_{t_{\mathcal{N}_5}}\right)\lambda^{(j)}_{\omega_t}\right]^2 \tag{85c}$$

$$\mathcal{L}^{data}_{W_{egr}}(\boldsymbol{\lambda}_{W_{egr}}) = \frac{1}{n}\sum_{j=1}^{n}\left[\left(W^{(j)}_{egr_{data}} - \widehat{W}^{(j)}_{egr_{NN}}\right)\lambda^{(j)}_{W_{egr}}\right]^2 \tag{85d}$$

The same in the Case 5 is given as,

$$\mathcal{L}^{data}_{p_{im}} = \frac{1}{n}\sum_{j=1}^{n}\left[p^{(j)}_{im_{data}} - \hat{p}^{(j)}_{im_{\mathcal{N}_1}}\right]^2 \tag{86a}$$

$$\mathcal{L}^{data}_{p_{em}} = \frac{1}{n}\sum_{j=1}^{n}\left[p^{(j)}_{em_{data}} - \hat{p}^{(j)}_{em_{\mathcal{N}_1}}\right]^2 \tag{86b}$$

$$\mathcal{L}^{data}_{\omega_t} = \frac{1}{n}\sum_{j=1}^{n}\left[\omega^{(j)}_{t_{data}} - \hat{\omega}^{(j)}_{t_{\mathcal{N}_5}}\right]^2 \tag{86c}$$

$$\mathcal{L}^{data}_{W_{egr}} = \frac{1}{n}\sum_{j=1}^{n}\left[W^{(j)}_{egr_{data}} - \widehat{W}^{(j)}_{egr_{NN}}\right]^2 \tag{86d}$$

where $p_{im_{data}}$, $p_{em_{data}}$, $\omega_{t_{data}}$ and $W_{egr_{data}}$ are the measured data of p_{im}, p_{em}, ω_t and W_{egr} respectively. $\hat{p}_{im_{\mathcal{N}_1}}$, $\hat{p}_{em_{\mathcal{N}_1}}$ and $\hat{\omega}_{t_{\mathcal{N}_5}}$ are the predicted values in p_{im}, p_{em} and ω_t respectively from $\mathcal{N}_1(t;\boldsymbol{\theta}_1)$, $\mathcal{N}_1(t;\boldsymbol{\theta}_1)$ and $\mathcal{N}_5(t;\boldsymbol{\theta}_5)$ respectively. Similarly, $\widehat{W}_{egr_{NN}}$ is predicted value in W_{egr} from NNs output. n is the number of measured data points.

$\mathcal{L}^{ini}_{p_{im}}$, $\mathcal{L}^{ini}_{p_{em}}$, $\mathcal{L}^{ini}_{\omega_t}$, $\mathcal{L}^{ini}_{\tilde{u}_{egr1}}$, $\mathcal{L}^{ini}_{\tilde{u}_{egr2}}$, $\mathcal{L}^{ini}_{\tilde{u}_{vgt}}$, $\mathcal{L}^{ini}_{x_r}$ and $\mathcal{L}^{ini}_{T_1}$ are the losses in initial conditions in p_{im}, p_{em}, ω_t $\tilde{u}_{egr1}$, $\tilde{u}_{egr2}$, $\tilde{u}_{vgt}$, x_r and T_1 respectively.

$$\mathcal{L}^{ini}_{p_{im}} = \frac{1}{1}\sum_{j=1}^{1}\left(p^{(j)}_{im_0} - \hat{p}^{(j)}_{im_0}\right)^2 \tag{87a}$$

$$\mathcal{L}^{ini}_{p_{em}} = \frac{1}{1}\sum_{j=1}^{1}\left(p^{(j)}_{em_0} - \hat{p}^{(j)}_{em_0}\right)^2 \tag{87b}$$

$$\mathcal{L}^{ini}_{\omega_t} = \frac{1}{1}\sum_{j=1}^{1}\left(\omega^{(j)}_{t_0} - \hat{\omega}^{(j)}_{t_0}\right)^2 \tag{87c}$$

$$\mathcal{L}^{ini}_{\tilde{u}_{egr1}} = \frac{1}{1}\sum_{j=1}^{1}\left(\tilde{u}^{(j)}_{egr1_0} - \hat{\tilde{u}}^{(j)}_{egr1_0}\right)^2 \tag{87d}$$

$$\mathcal{L}^{ini}_{\tilde{u}_{egr2}} = \frac{1}{1}\sum_{j=1}^{1}\left(\tilde{u}^{(j)}_{egr2_0} - \hat{\tilde{u}}^{(j)}_{egr2_0}\right)^2 \tag{87e}$$

$$\mathcal{L}^{ini}_{\tilde{u}_{vgt}} = \frac{1}{1}\sum_{j=1}^{1}\left(\tilde{u}^{(j)}_{vgt_0} - \hat{\tilde{u}}^{(j)}_{vgt_0}\right)^2 \tag{87f}$$

$$\mathcal{L}_{x_r}^{ini} = \frac{1}{1}\sum_{j=1}^{1}\left(x_{r_0}^{(j)} - \hat{x}_{r_0}^{(j)}\right)^2 \tag{87g}$$

$$\mathcal{L}_{T_1}^{ini} = \frac{1}{1}\sum_{j=1}^{1}\left(T_{1_0}^{(j)} - \hat{T}_{1_0}^{(j)}\right)^2 \tag{87h}$$

where p_{im_0}, p_{em_0}, ω_{t_0}, $\tilde{u}_{egr1_0}$, $\tilde{u}_{egr2_0}$, $\tilde{u}_{vgt_0}$, x_{r_0} and T_{1_0} are the initial conditions and $\hat{p}_{im_0}$, $\hat{p}_{em_0}$, $\hat{\omega}_{t_0}$, $\hat{\tilde{u}}_{egr1_0}$, $\hat{\tilde{u}}_{egr2_0}$, $\hat{\tilde{u}}_{vgt_0}$, $\hat{x}_{r_0}$ and $\hat{T}_{1_0}$ are corresponding output from neural network at time $t = 0$ for p_{im}, p_{em}, ω_t, $\tilde{u}_{egr1}$, $\tilde{u}_{egr2}$, $\tilde{u}_{vgt}$, x_r and T_1 respectively.

Appendix 6: Additional tables

See Tables 7, 8 and 9.

Appendix 7: Additional figures

In this section, we present the results for Case 1 and Case 2.

	Description	Symbol	Value
1	Ideal gas constant of air	R_a	287
2	Intake manifold temperature	T_{im}	300.6186
3	Intake manifold volume	V_{im}	0.0220
4	Ideal gas constant of exhaust gas	R_e	286
5	Exhaust manifold volume	V_{em}	0.0200
6	Displaced volume of the cylinder	V_d	0.0127
7	Number of cylinder	n_{cyl}	6
8	Specific heat capacity ratio of air	γ_a	1.3964
9	Specific heat capacity at constant pressure of air	c_{pa}	1011
10	Specific heat capacity at constant volume air	c_{va}	724
11	Compression ratio	r_c	17
12	Fuel consumed during constant-volume combustion	x_{cv}	2.3371×10^{-14}
13	Heating value of fuel	q_{HV}	42900000
14	Diameter of exhaust pipe	d_{pipe}	0.1
15	Length of exhaust pipe	l_{pipe}	1
16	Number of exhaust pipe	n_{pipe}	2
17	Specific heat capacity at constant pressure of exhaust gas	c_{pe}	1332
18	Time constant 1 for EGR	τ_{egr1}	0.05
19	Time constant 2 for EGR	τ_{egr2}	0.13
20	Time delay constant for EGR	τ_{degr}	0.065
21	Constant for EGR overshoot	K_{egr}	1.8
22	Optimal value of pressure ratio of EGR	Π_{egropt}	0.6500
23	Inertial of turbocharger	J_t	2.0×10^{-4}
24	Time constant for VGT	τ_{vgt}	0.025
25	Time delay constant for VGT	τ_{dvgt}	0.04
26	Specific heat capacity at constant pressure of exhaust	c_{pe}	1332
27	Specific heat capacity ratio of exhaust gas	γ_e	1.2734
28	Turbine blade radius	R_t	0.04
29	Compressor blade radius	R_c	0.0400

Table 7. Values of the constants considered in the present study.

Unknown	η_{sc}	h_{tot}	A_{egrmax}	A_{vgtmax}
Value	1.1015	96.2755	4.0×10^{-4}	8.4558×10^{-4}

Table 8. True value of the unknown parameters.

Symbol	Value	Symbol	Value	Symbol	Value
c_{vol1}	-2.0817×10^{-4}	c_{egr1}	-1.1104×10^{-4}		
c_{vol2}	$-\ 0.0034$	c_{egr2}	0.0178		
c_{vol3}	1.1497	c_{egr3}	0		
$c_{\omega\Psi1}$	1.0882×10^{-8}	$c_{\omega\Phi1}$	-1.4298×10^{-8}	$c_{\Psi2}$	0
$c_{\omega\Psi2}$	-1.7320×10^{-4}	$c_{\omega\Phi2}$	-0.0015	$c_{\Phi2}$	0
$c_{\omega\Psi3}$	1.0286	$c_{\omega\Phi3}$	29.6462		
π_{copt}	1.0455	c_{m1}	1.3563	c_{vgt1}	126.8719
W_{copt}	0.2753	c_{m2}	$2.7692e+03$	c_{vgt2}	117.1447
a_1	3.0919	c_{m3}	0.0100	c_{f1}	1.9480
a_2	2.1479	BSR_{opt}	0.9755	c_{f2}	-0.7763
a_3	-2.4823	$\eta_{tm,max}$	0.8180	K_t	2.8902
η_{cmax}	0.7364				
c_{π}	0.2708				

Table 9. Value for the coefficients of the empirical formulae.

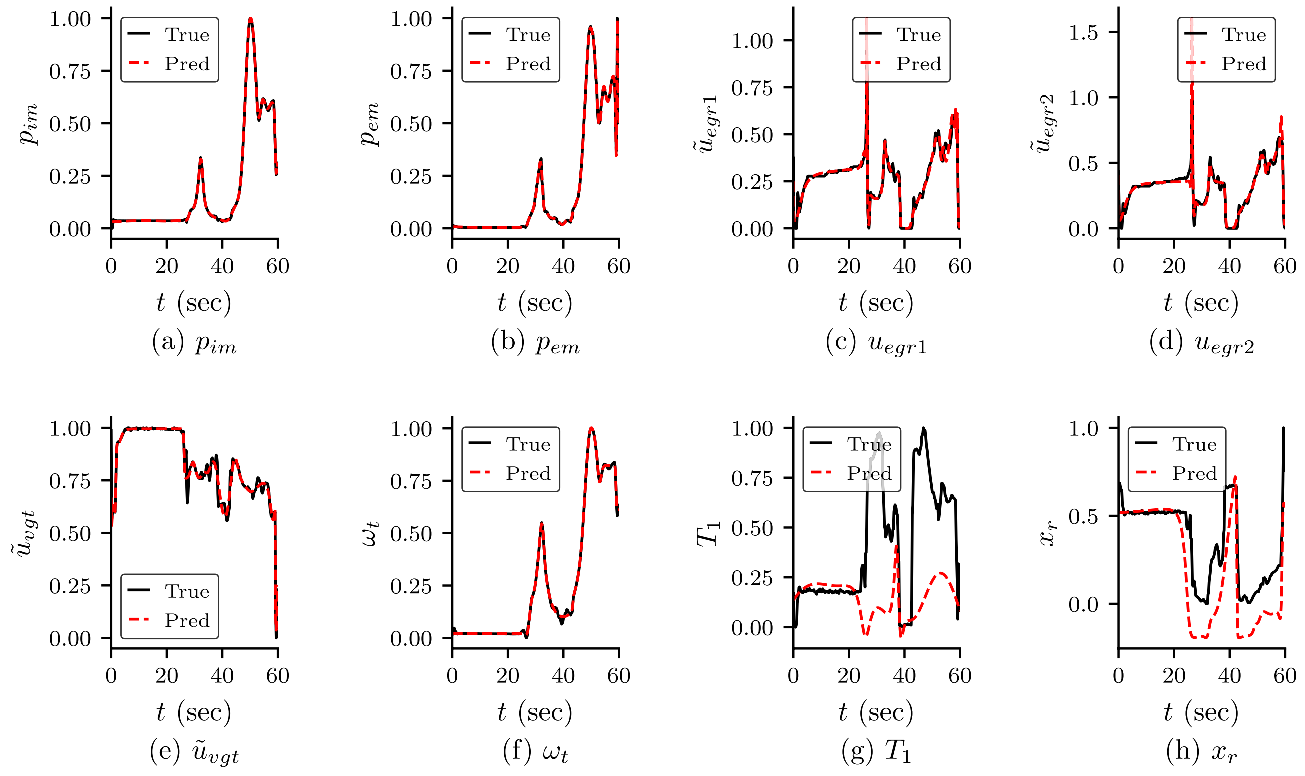


Figure 12. Predicted states and T_1 and x_r for Case 1: predicted dynamics of the state variables of the engine and T_1 and x_r for Case 1 (PINN with self-adaptive weights for 3 unknown parameters). It can be observed that the predicted dynamics of the states are in good agreement with the true values. However, similar to 3 unknown parameters T_1 and x_r do not match with the true value.

For Case-1: 3 unknown parameters with clean data. The predicted state variables and T_1 and x_r for Case 1 (3 unknown with clean data) are shown in Fig. 12. The predicted dynamics of the known variables are shown in Fig. 13a–d. In Fig. 13e–h, we have shown the dynamics of variables which are dependent on the unknown parameters. The predicted empirical formulae are shown in Fig. 14. We also studied the convergence of the unknown parameters, which are shown in Fig. 15.

For Case-2: 3 unknown parameters with noisy data. The predicted state variables and T_1 and x_r for Case 2 (3 unknown with noisy data) are shown in Fig. 16. The predicted dynamics of the known variables are shown in Fig. 17a–d. In Fig. 17e–h, we have shown the dynamics of variables which are dependent on the unknown parameters. The predicted empirical formulae are shown in Fig. 18. We also studied the convergence of the unknown parameters, which are shown in Fig. 19.

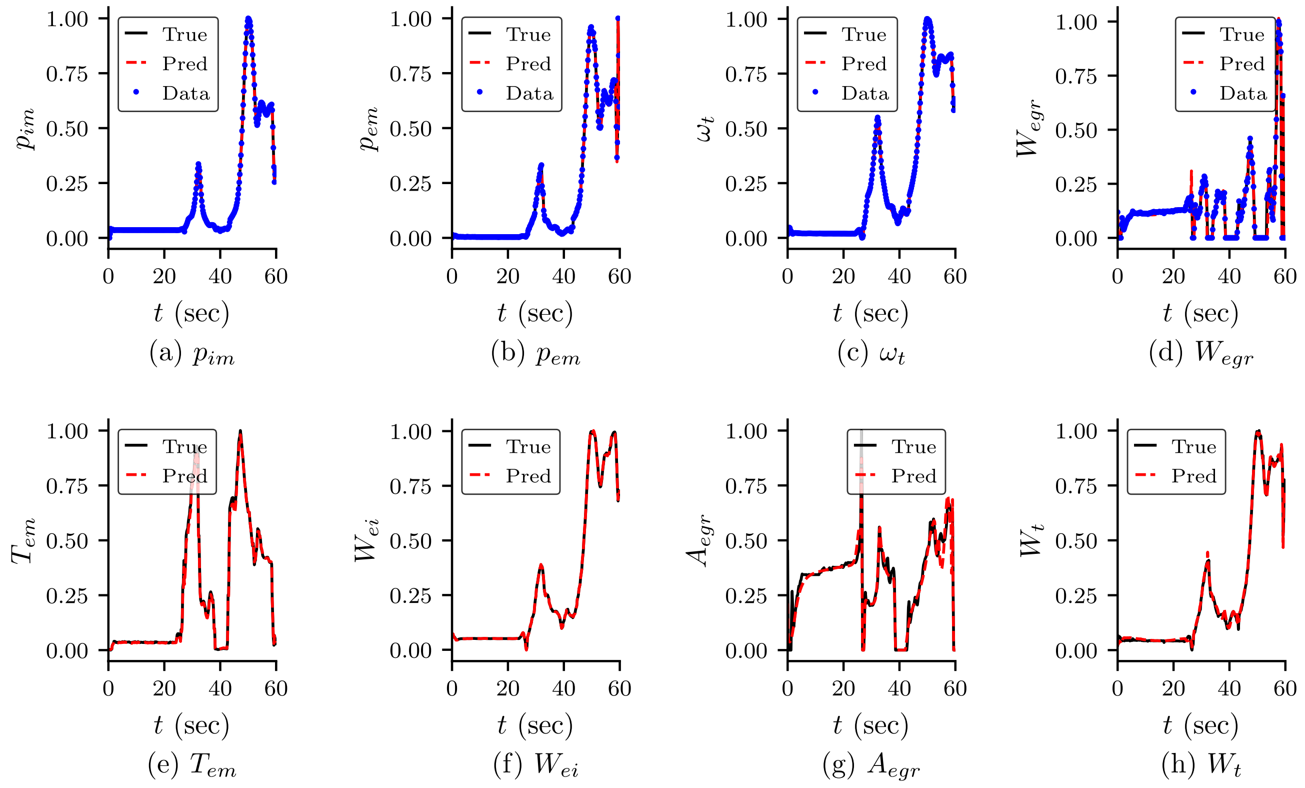


Figure 13. Predicted dynamics of variables for Case 1: (**a**–**d**) predicted dynamics of the variables whose field measurement data are known. (**e**–**h**) dynamics of important variables which also depend on the unknown parameters for Case 1.

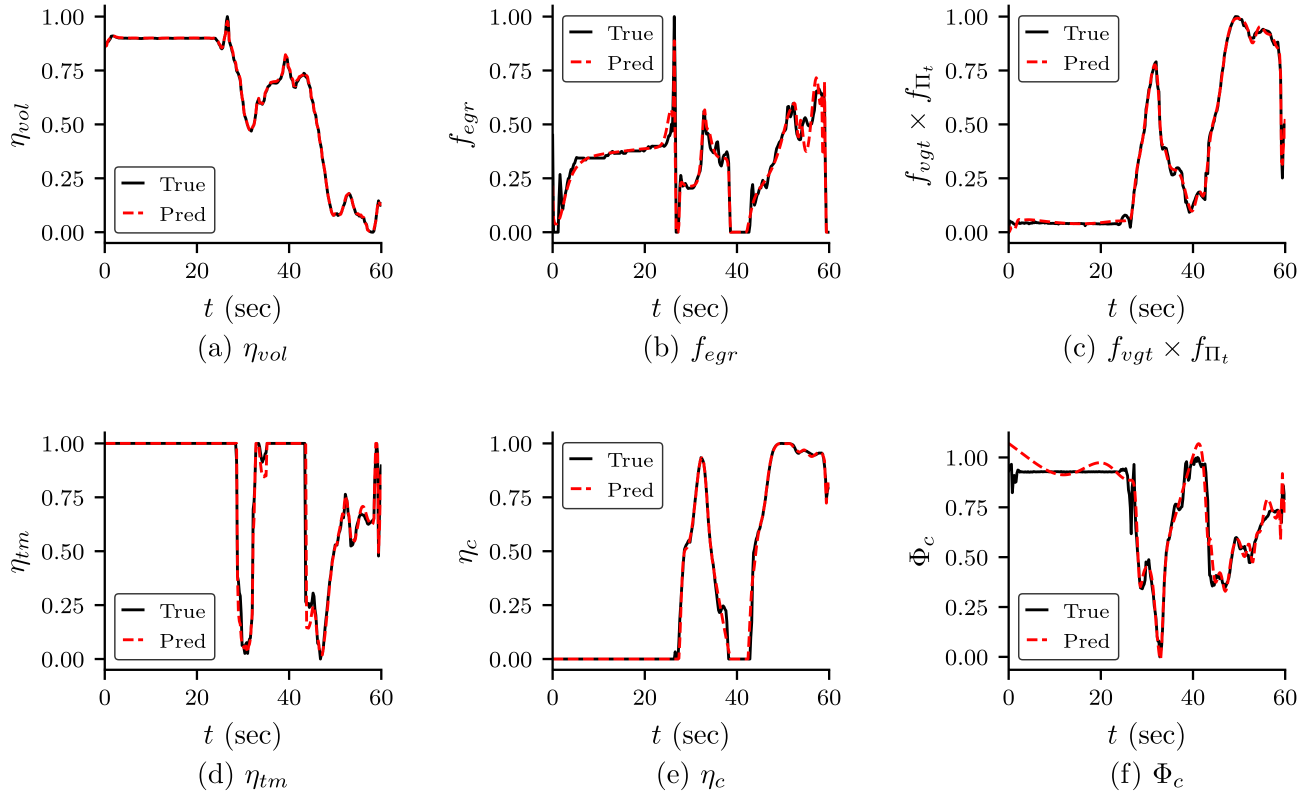


Figure 14. Empirical formulae for Case 1: the predicted values of empirical formulae for Case 1 (3 unknown parameters with clean data).

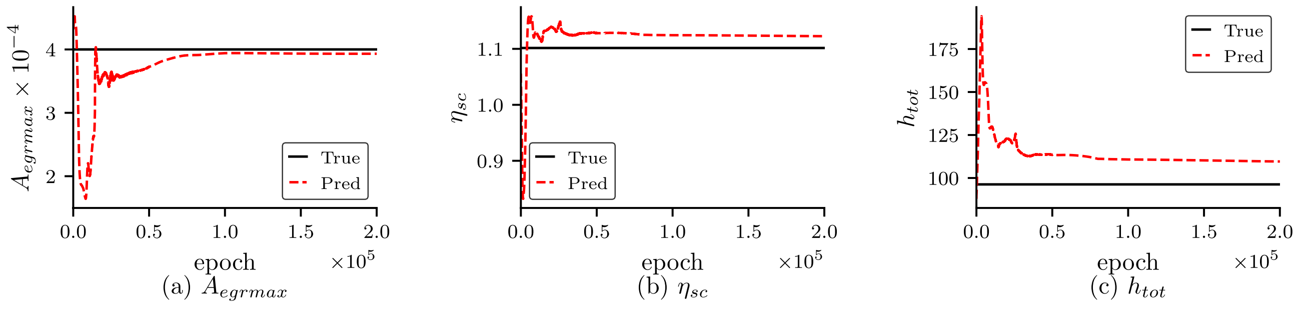
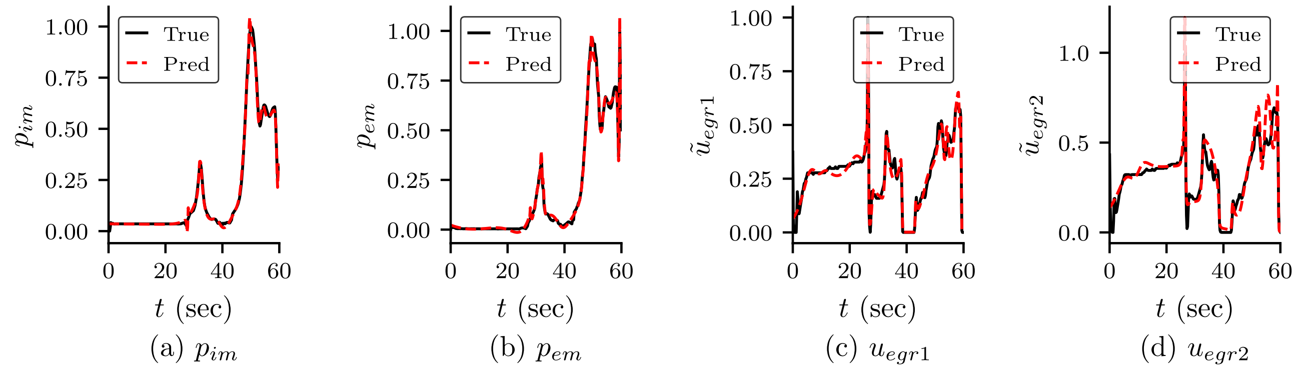


Figure 15. Convergence of the unknown parameters for Case 1: convergence of the unknown parameters with epoch for Case 1 (3 unknown parameters with clean data).



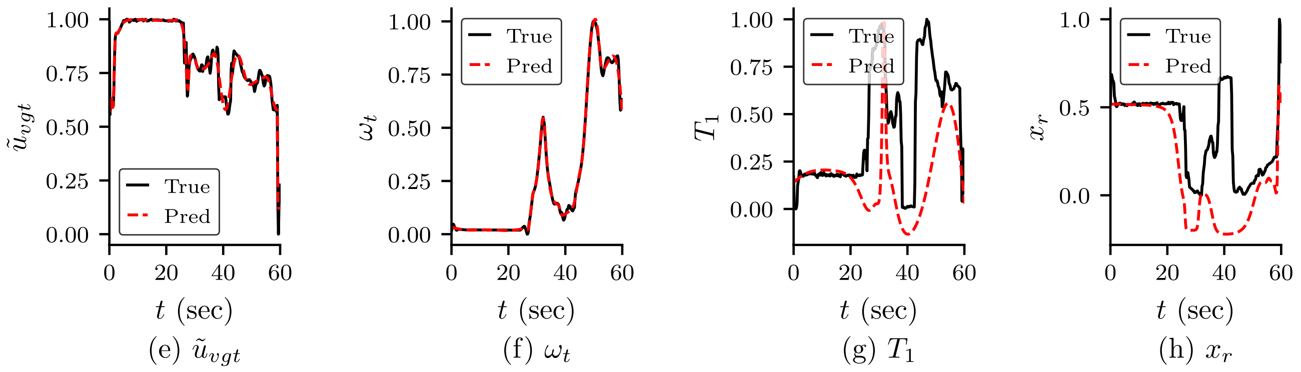


Figure 16. Predicted states and T_1 and x_r for Case 2: predicted dynamics of the state variables of the engine and T_1 and x_r for Case 2 (PINN with self-adaptive weights for 3 unknown paramters). It can be observed that the predicted dynamics of the states are in good agreement with the true values. However, similar to 4 unknown parameters T_1 and x_r do not match with the true value.

Appendix 8: Neural network surrogates for empirical formulae

The empirical formulae of the engine model are approximated using surrogate neural networks and are discussed in "Neural network surrogates for empirical formulae2.1". In "Data generation", we discuss the laboratory data required to train these neural networks. The laboratory data required for training of each neural network are shown in Table 3 ("Data generation"). The labelled data for training these neural networks may be calculated from static data on the entire operational range of each quantity except for turbine mechanical efficiency (η_{tm}). The labelled data for the turbine mechanical efficiency is calculated using Eq. (44) ("Turbocharger"), which is a differential equation, thus requiring dynamic data with fine dt. The calculations of the labelled data from the laboratory measurements are discussed in "Appendix 4". In the case of training of neural network $\mathcal{N}_3^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_3^P)$ for the approximation of F_{vgt,Π_t}, L_2 weight regularizer (except last layer) is considered in the loss function with a coefficient 5×10^{-10}.

The predicted values of the empirical formulae for Case-V (Table 4 in "Data generation") with the true values for 1-min duration are shown in Fig. 20. The % relative L_2 errors for training and testing data set are shown in the last two columns of Table 10. We observe that the neural networks are able to predict the empirical quantity with very good accuracy. The testing error in the case of the surrogate neural network for f_{egr} is smaller than the

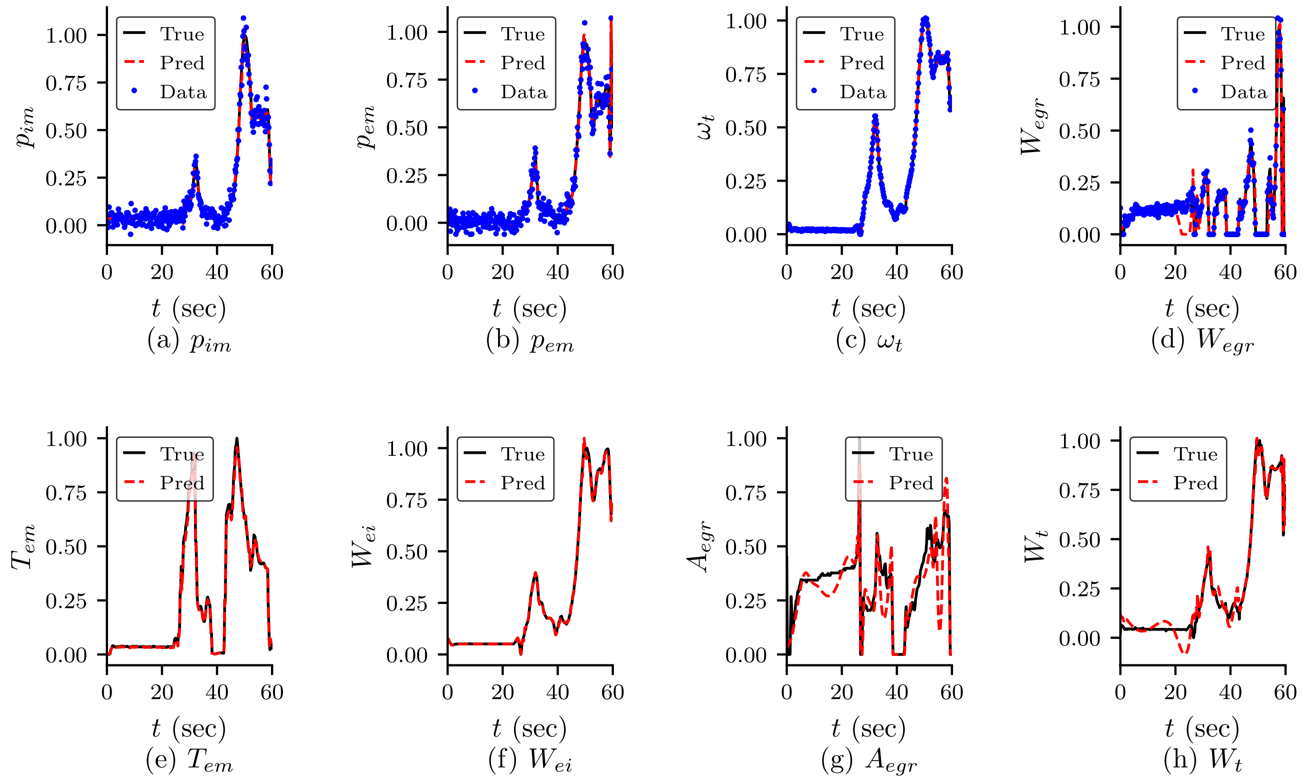


Figure 17. Predicted dynamics of variables for Case 2: (**a**–**d**) predicted dynamics of the variables whose field measurement data are known. (**e**–**h**) dynamics of important variables which also depend on the unknown parameters for Case 2.

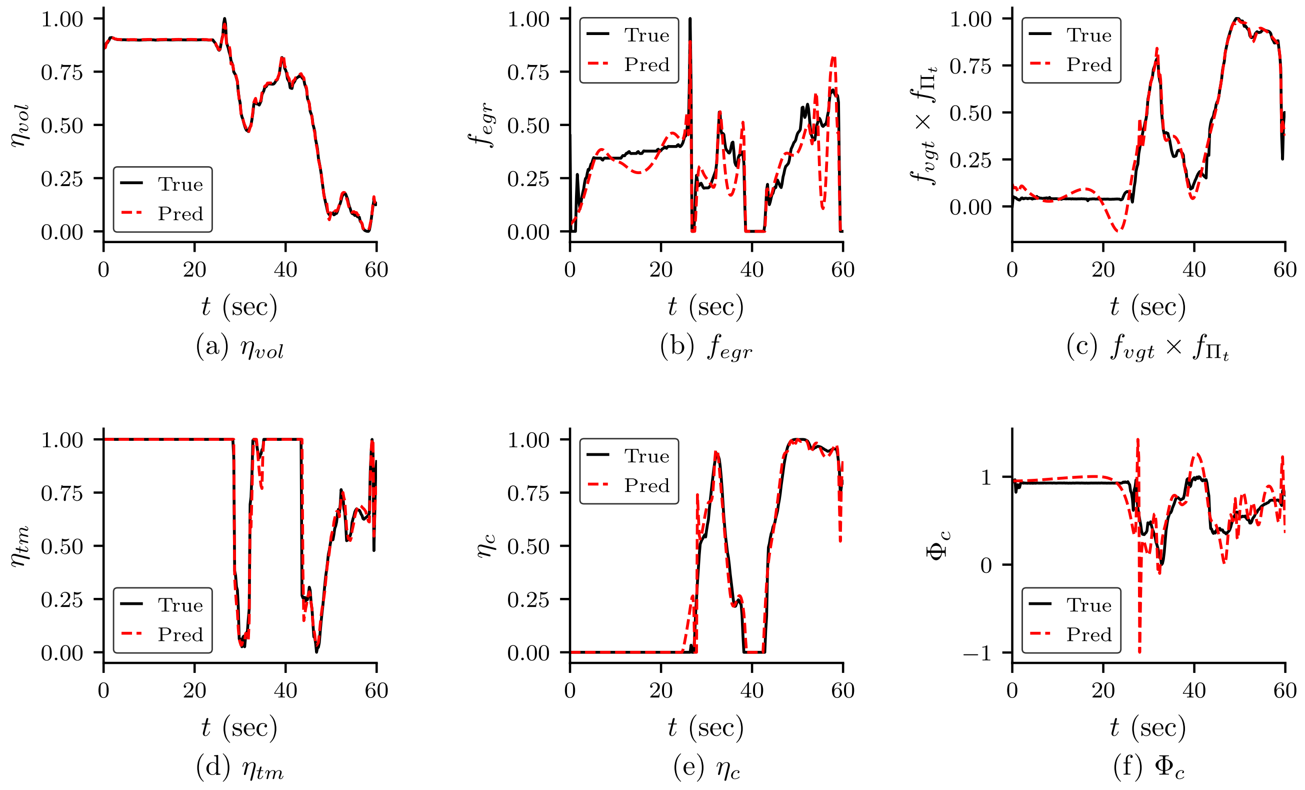


Figure 18. Empirical formulae for Case 2: the predicted values of empirical formulae for Case 2 (3 unknown parameters with noisy data).

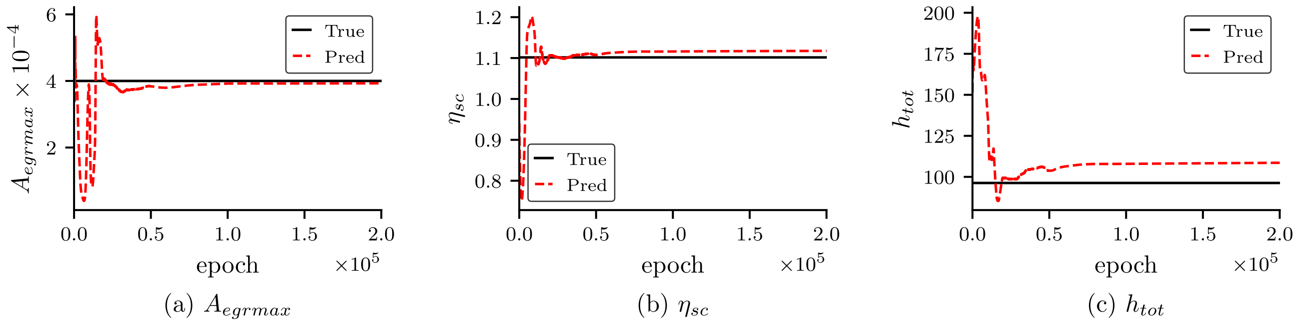


Figure 19. Convergence of the unknown parameters for Case 2: convergence of the unknown parameters with epoch for Case 2 (3 unknown parameters with noisy data).

Neural network	Input (x)	Network size	Output	Output restrict [‡]	L_2 error (%)	
					Train	Test
$\mathcal{N}_1^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_1^P)$	n_e, p_{im}	[2, 4, 4, 1]	η_{vol}		0.01	0.03
$\mathcal{N}_2^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_2^P)$	$\tilde{u}_{egr}$	[1, 4, 4, 1]	f_{egr}	$S(f_{egr})$	0.14	0.10
$\mathcal{N}_3^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_3^P)$	$\Pi_t, \tilde{u}_{vgt}$	[2, 8, 8, 8, 1]	F_{vgt,Π_t}	$1.1 \times S(F_{vgt,\Pi_t})$	0.03	0.52
$\mathcal{N}_4^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_4^P)$	ω_t, Π_t, T_{em}	[3, 4, 4, 4, 1]	η_{tm}	$\min(0.818, \eta_{tm})$	1.62	1.32
$\mathcal{N}_5^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_5^P)$	W_c, Π_c	[2, 4, 4, 4, 1]	η_c	$\max(0.2, S(\eta_c))$	0.16	0.18
$\mathcal{N}_6^{(P)}(\boldsymbol{x}; \boldsymbol{\theta}_6^P)$	ω_t, Π_c, T_{amb}	[3, 10, 10, 10, 1]	Φ_c	$S(\Phi_c)$	0.76	1.13

Table 10. Details of neural networks for empirical formulae: details of the DNNs to approximate empirical formulae. The first two columns specify the neural network (Table 2) and their input, respectively. The third column indicates the neural network size considered. The activation function of the hidden layers is $\sigma(\cdot) = tanh(\cdot)$. The "Output" column specifies the empirical quantity the neural network approximated. The last two columns give the results for the test data after the completion of the training. The column "Error" specifies the relative $\%L_2$ error for the test case (Case V). Appropriate scaling of input and output are considered in the training of neural networks. [‡] $S \longrightarrow$ sigmoid function.

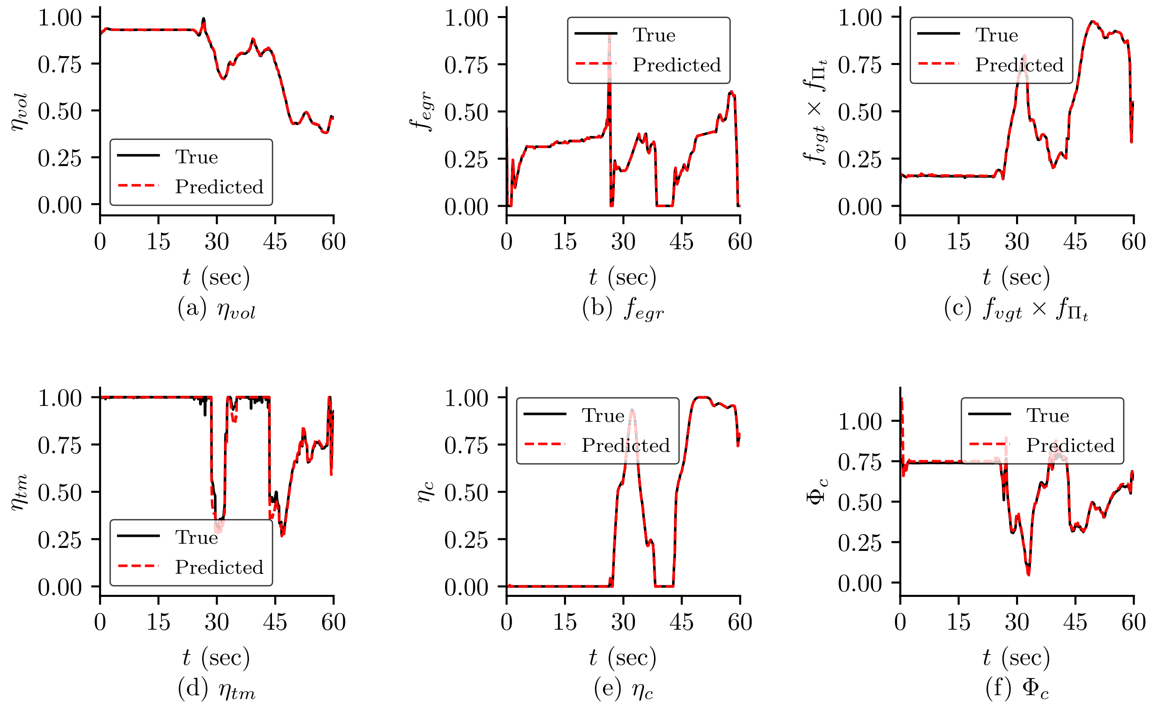


Figure 20. Prediction of empirical formulae: the predicted and true values of the empirical formulae for test case (Case-V). The plots are normalized within 20-min data, and only a portion (0–1 min) of the results are shown. The predicted empirical formulae are in good agreement with the true values.

training error. This is because of the nature of the function and the data considered. The input-output relationship is simple, with only one input and one output. The maximum value of the testing data is smaller than the maximum value of the training data. Similarly, the minimum value of testing data is larger than the minimum value of training data. We also observed that the standard deviation of testing data is smaller than the standard deviation of the training data. Since the EGR system is independent and f_{egr} depends only on $\tilde{u}_{egr}$, not any other variables (e.g. ambient temperature and pressure), we assume that most of the testing set of data might be within the training data set (training data set is 2 h while testing data set is 20 min). Thus, the testing error is marginally smaller than the training error. The testing error in the case of the surrogate model for η_{tm} is smaller than that of the training error. The approximation considered in calculating the labelled data for η_{tm} from the laboratory data, we have considered a five-point method to approximate the differentiation present in Eq. (44) ("Turbocharger"). Thus, a few noisy data are observed in both training and testing data sets. As the duration of the training data set is larger than the testing dataset, the amount of noisy data is more in the training data. Thus, the error in training is slightly higher than the testing error. These neural networks, after training, will be used in the places of the empirical formulae in the inverse problem. The trained weights and biases will be considered fixed in the inverse problem.

Received: 3 April 2023; Accepted: 3 August 2023
Published online: 22 August 2023

References

1. Wahlström, J. & Eriksson, L. 'Modelling diesel engines with a variable-geometry turbocharger and exhaust gas recirculation by optimization of model parameters for capturing non-linear system dynamics. *Proc. Inst. Mech. Eng. Part D J. Automobile Eng.* **225**(7), 960–986. https://doi.org/10.1177/0954407011398177 (2011).
2. Raissi, M., Perdikaris, P. & Karniadakis, G. 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).
3. Baydin, A. G., Pearlmutter, B. A., Radul, A. A. & Siskind, J. M. Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.* **18**, 1–43 (2018).
4. Jagtap, A. D., Kharazmi, E. & Karniadakis, G. E. 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).
5. Jagtap, A. D. & Karniadakis, G. E. Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Commun. Comput. Phys.* **28**, 2002–2041 (2020).
6. Kharazmi, E., Zhang, Z. & Karniadakis, G. E. hp-VPINNs variational physics-informed neural networks with domain decomposition. *Comput. Methods Appl. Mech. Eng.* **374**, 113547 (2021).
7. Meng, X., Li, Z., Zhang, D. & Karniadakis, G. E. PPINN: Parareal physics-informed neural network for time-dependent PDEs. *Comput. Methods Appl. Mech. Eng.* **370**, 113250 (2020).
8. Cho, J., Nam, S., Yang, H., Yun, S.-B., Hong, Y. & Park, E. Separable pinn: Mitigating the curse of dimensionality in physics-informed neural networks. arXiv:2211.08761 (arXiv preprint) (2022).
9. McClenny, L. & Braga-Neto, U. Self-adaptive physics-informed neural networks using a soft attention mechanism. arXiv:2009.04544 (arXiv preprint) (2020).
10. Jagtap, A. D., Mao, Z., Adams, N. & Karniadakis, G. E. Physics-informed neural networks for inverse problems in supersonic flows. *J. Comput. Phys.* **466**, 111402 (2022).
11. Chen, Y., Lu, L., Karniadakis, G. E. & Negro, L. D. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Opt. Express* **28**(8), 11618–11633 (2020).
12. Depina, I., Jain, S., Mar Valsson, S. & Gotovac, H. Application of physics-informed neural networks to inverse problems in unsaturated groundwater flow. *Georisk Assess. Manage. Risk Eng. Syst. Geohazards* **16**(1), 21–36 (2022).
13. Cai, S., Mao, Z., Wang, Z., Yin, M. & Karniadakis, G. E. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta. Mech. Sin.* **20**, 1–12 (2022).
14. Lawal, Z. K., Yassin, H., Lai, D. T. C. & Che Idris, A. Physics-informed neural network (PINN) evolution and beyond: A systematic literature review and bibliometric analysis. *Big Data Cognit. Comput.* **6**, 4 (2022).
15. Cuomo, S. *et al.* Scientific machine learning through physics-informed neural networks: Where we are and what's next. *J. Sci. Comput.* **92**(3), 88 (2022).
16. Biao, L., Qing-chun, L., Zhen-hua, J. & Sheng-fang, N. System identification of locomotive diesel engines with autoregressive neural network. In *2009 4th IEEE Conference on Industrial Electronics and Applications*, 3417–3421 (2009).
17. Finesso, R. & Spessa, E. A real time zero-dimensional diagnostic model for the calculation of in-cylinder temperatures, hrr and nitrogen oxides in diesel engines. *Energy Convers. Manage.* **79**, 498–510 (2014).
18. Tosun, E., Aydin, K. & Bilgili, M. Comparison of linear regression and artificial neural network model of a diesel engine fueled with biodiesel-alcohol mixtures. *Alex. Eng. J.* **55**(4), 3081–3089 (2016).
19. González, J. P., Ankobea-Ansah, K., Peng, Q. & Hall, C. M. On the integration of physics-based and data-driven models for the prediction of gas exchange processes on a modern diesel engine. *Proc. Inst. Mech. Eng. Part D J. Automobile Eng.* **236**(5), 857–871. https://doi.org/10.1177/09544070211031401 (2022).
20. Kumar, V., Goswami, S., Smith, D. J. & Karniadakis, G. E. Real-time prediction of multiple output states in diesel engines using a deep neural operator framework. arXiv:2304.00567 (arXiv preprint) (2023).
21. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **3**(3), 218–229 (2021).
22. Software packages from Vehicular Systems (2010) http://www.fs.isy.liu.se/Software. [Online].
23. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. arXiv:1412.6980 (arXiv preprint) (2014).
24. Jagtap, A. D. & Karniadakis, G. E. How important are activation functions in regression and classification? A survey, performance comparison, and future directions. *J. Mach. Learn. Model. Comput.* **4**, 1 (2023).

Acknowledgements

K.N., X.M. and G.E.K. would like to acknowledge the support by Cummins Inc. USA.

Author contributions

K.N.: conceptualization, formal analysis, investigation, methodology, software, validation, visualization, writing—original draft, writing—review and editing. X.M.: conceptualization, formal analysis, investigation, methodology, software, validation, visualization, writing—original draft, writing—review and editing. D.J.S.: conceptualization, data curation, project administration, supervision, writing—original draft, writing—review and editing. G.E.K.: conceptualization, funding acquisition, project administration, resources, supervision, writing - original draft, writing—review and editing. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to G.E.K.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.