Journal of Computational Physics 519 (2024) 113380

Contents lists available at ScienceDirect

Journal of Computational Physics

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

A new method to compute the blood flow equations using the physics-informed neural operator

Lingfeng Li [a], Xue-Cheng Tai [b,*], Raymond Hon-Fu Chan [c,d,a]

[a] Hong Kong Centre for Cerebro-cardiovascular Health Engineering, Hong Kong, China
[b] Norwegian Research Center, Bergen, Norway
[c] Department of Operations and Risk Management, Lingnan University, Hong Kong, China
[d] School of Data Science, Lingnan University, Hong Kong, China

ARTICLE INFO

Keywords:
Navier-Stokes equation
Blood flow simulation
DeepONet
Physics-informed neural network
Cuffless blood pressure estimation

ABSTRACT

Using the Navier-Stokes equation for real blood flow simulations in patients is a challenging task. Among many of the challenges, initial and boundary conditions are not directly measurable. Many parameters for the flow model are also not directly measurable and differ from patient to patient and over time. New machine learning techniques, like physics-informed neural networks (PINN), offer some new ways to handle these difficulties. In this work, we aim to learn the operator that maps some easily measurable physiological signals to the solution of the blood flow equation. We use our proposed model based on Navier-Stokes equation and PINN to fit real data on blood pressure. A Windkessel boundary condition is used to produce physically correct reflection waves. A time-periodic condition is used to capture the periodicity of blood flow and enables our model to simulate the blood flow without initial and boundary conditions. Furthermore, we allow the periods of each instance of solution to be different, which makes the training of neural operators computationally expensive, but more accuracy and physical correct towards real blood pressure data. Further more, we also propose an efficient implementation to incorporate the periodic condition into our model. Estimating the hyper-parameters in the Navier-Stokes equation is also difficult. We then introduce a hyper-parameter network to estimate these parameters during the training process as well. The blood flow data contains useful information for disease detection and diagnosis, but directly measuring the entire blood flow remains a significant challenge. We apply our proposed method to cuffless blood pressure estimation. More specifically, we aim to predict the blood pressure waveform (continuous blood pressure and velocity in both time and space) from Electrocardiogram (ECG) and photoplethysmogram (PPG) signals, which can be easily measured using wearable devices. Compared to other methods, our method is the first one that can predict blood flow continuously, both with location and time which are valuable for cardiovascular medical treatments and diagnoses.

1. Introduction

Navier-Stokes equation is a very popular and powerful tool for modeling and simulating the dynamics of incompressible flow. [1] introduces a three-dimensional Navier-Stokes equation for blood flow modeling. By applying some physical and mathematical

* Corresponding author.
E-mail addresses: lfli@hkcoche.org (L. Li), xtai@norceresearch.no (X.-C. Tai), raymond.chan@ln.edu.hk (R.H.-F. Chan).

https://doi.org/10.1016/j.jcp.2024.113380
Received 10 April 2024; Received in revised form 25 July 2024; Accepted 26 August 2024

Available online 30 August 2024

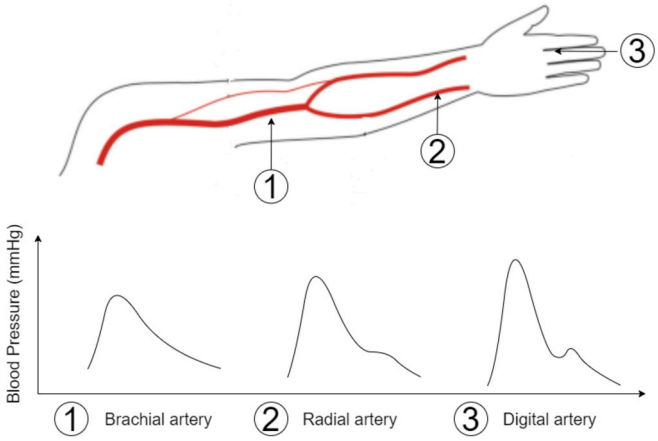


Fig. 1. Illustration of the pulse pressure amplification in arterial systems.

assumptions, the original three-dimensional model can be reduced to a simplified one-dimensional model [1,2]. Though the three-dimensional model can capture the detailed dynamic of the fluids and the vessel walls, its computational complexity is extremely high. The one-dimensional model, on the other hand, provides just the averaged flow information in the arteries, but its computational cost is several orders of magnitude lower than the three-dimensional model. Besides, the one-dimensional model can also accurately describe the pulse propagation in the arteries, which is very useful in many applications that do not require flow details.

For the one-dimensional model, many classical solvers can solve it accurately, such as the MacCormack scheme [3,4], Taylor-Galerkin scheme [5,6], MUSCL [7,8], and the local discontinuous Galerkin scheme [9–11]. Typically, solving this model requires inlet and outlet boundary conditions, compatibility conditions, and initial conditions. Recently, the physics-informed neural network (PINN) has also been applied to solve this equation [12], which demonstrates that PINN has the ability to infer the solution from noisy clinical data. In the real clinical setting, it is very difficult to measure all the required boundary conditions and initial conditions for each patient. Therefore, the physics-informed machine learning approach may be a more suitable method for these applications. The PINN method is designed to solve one instance of PDE. When the boundary/initial condition changes, we need to retrain a new network to solve it. This is not very practical if we need to solve many instances of the same type of PDEs. Another type of machine learning approach for solving PDE is the neural operators. The neural operator is designed to learn the solution operator of a PDE. The input of the neural operator is a function sampled from some function spaces. It can be an initial condition of a time-dependent PDE, the diffusion coefficient of a variable coefficient second-order PDE, or some other functions that are related to the PDE. Some popular neural operators include the DeepONet [13] and the FNO [14].

In the field of biomedical engineering, there are many research works focusing on predicting the blood flow information, especially the blood pressure waveform from some physiological signals like the electrocardiogram (ECG) and the photoplethysmogram (PPG). The arterial blood pressure (ABP) waveforms are able to reflect the cardiovascular status of humans [15–18], but measuring the blood pressure waveform directly is difficult. The gold standard so far is the invasive method, which involves implanting an invasive pressure sensor into the artery. The invasive method is very risky, and the used devices are expensive. Therefore, it is of great importance to predict the blood flow from some physiological signals that can be easily measured by some wearable devices. Many deep learning models have been introduced to estimate blood flow information, especially blood pressure, from physiological signals [19–22]. The models above are all trained in a data-driven way, i.e., trained by minimizing the difference between the ground truth label and the network prediction. Since the labeled data is usually measured by invasive methods in the radial artery [23], the trained neural network will only predict the radial blood pressure waveform. However, ABP waveforms differ at different locations in the arterial systems [24,25], and radial waveforms may not be sufficient to diagnose cardiovascular diseases and can reflect the status of the cardiovascular system [26,27]. Another notable observation is the pulse pressure amplification effect. Pulse pressure is defined as the difference between the maximum and minimum of the pressure waveforms, and this difference will increase through the propagation of the pulse wave in the arterial system (see Fig. 1). All current deep learning-based algorithms can only predict radial ABP waveforms since they are trained with labeled data measured at the radial artery. Current guidelines for the management of hypertension are based on brachial blood pressure, so the radial blood pressure may not be accurate enough to diagnose hypertension [25].

In this work, we aim to develop new methods to learn the solution operator that maps some functions, like physiological signals, to a PDE solution. The blood flow will be modeled by a one-dimensional Navier-Stokes equation [2]. To better match our estimation with the real blood flow data, we pose a Windkessel boundary condition to produce some physically reasonable reflection waves. Since the initial condition cannot be measured in patients, we also pose a time-periodic condition where the period can be easily inferred from the heart rate. This periodic condition can incorporate this condition into the neural operator using the feature expansion method mentioned in [28]. However, the heart rate can be different from person to person and from time to time. Thus, the time-periodic condition posed to the PDE should be different for each instance. This property will cause some difficulties during the training of neural operators. We will also propose efficient implementations to resolve this problem.

We summarize our main contributions as follows:

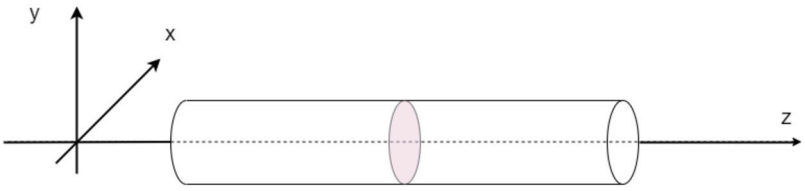


Fig. 2. Simplification of the three-dimensional model to one-dimensional model. The vessel is assumed to be a straight cylinder, and each axial cross-section is circular.

- To the best of our knowledge, this work is the first attempt to apply physics-informed neural operators to learn the solution operator of the blood flow equation.
- The proposed methods incorporate a Windkessel boundary condition to capture the real reflection waves and variable time-periodic conditions. Without using these conditions, it is not possible to match the model prediction with real data.
- The proposed method can also estimate model hyper-parameters automatically from the input physiological signals.
- We apply the proposed method to the application of blood pressure waveform estimation. We also propose a new structure of DeepONet, called BP-DeepONet (BP refers to the blood pressure.), for this specific task which can extract information from physiological signals efficiently.
- The proposed method is the first one to predict blood pressure waveforms continuously at different locations and times in the arteries. It can predict the blood pressure waveforms with reasonably good accuracy. It is remarkable that our model can preserve some physical properties like pulse pressure amplification which is not easy for other models.
- During training, we only require the blood pressure measurement at the outlet side of the domain, and no extra measurement is needed.

2. One-dimensional model for blood flow simulation

The hemodynamics in a vessel segment can be effectively modeled and simulated by the one-dimensional Navier-Stokes equation [1,2]. The one-dimensional model is a simplification of the three-dimensional Navier-Stokes equation by making several assumptions on the computational domain:

- The vessel is a straight cylinder with the axis oriented along the coordinate z direction (see Fig. 2).
- The vessel wall displaces along the radial direction only, which implies that each axial cross-section is always circular.
- The blood pressure and blood flow rate are constant within each axial cross-section.
- All body forces, like the gravity, are neglected.

Thus, the hemodynamics can be described by three quantities, namely the blood flow rate $Q(z,t)$, the axial cross-sectional area $A(z,t)$, and the blood pressure $P(z,t)$, where $z \in [0, L]$ is the spatial coordinates and $t \in [0, T]$ is the temporal coordinates. L denotes the length of the vessel segment, and T represents the time duration. Then, the one-dimensional model can be written as:

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial z} = 0, \tag{1}$$

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial z}\left(\frac{Q^2}{A}\right) + \frac{A}{\rho}\frac{\partial P}{\partial z} + K_r \frac{Q}{A} = 0, \tag{2}$$

where ρ is the blood density, and K_r is a resistance parameter related to blood viscosity. To close this system, the blood pressure P and the cross-sectional area are assumed to satisfy:

$$P = P_{ext} + \beta \frac{\sqrt{A} - \sqrt{A_0}}{A_0}, \tag{3}$$

where P_{ext} is the external pressure, A_0 is the cross-sectional area at the reference state, and β is a constant depending on the physical and mechanical properties of vessels. More details of the model derivation can be found in [1,2]. A simple characteristic analysis shows that the forward characteristic variable W_1 and backward characteristic variable W_2 in the system (1)-(2) are:

$$W_1(A, Q) = \frac{Q}{A} + 4\sqrt{\frac{\beta}{2\rho A_0}}\left(A^{1/4} - A_0^{1/4}\right),$$

and

$$W_2(A, Q) = \frac{Q}{A} - 4\sqrt{\frac{\beta}{2\rho A_0}}\left(A^{1/4} - A_0^{1/4}\right).$$

The forward wave and backward wave are traveling at the speed

$$\lambda_1(A,Q) = \frac{Q}{A} + \sqrt{\frac{\beta}{2\rho A_0}} A^{1/4}$$

and

$$\lambda_2(A,Q) = \frac{Q}{A} - \sqrt{\frac{\beta}{2\rho A_0}} A^{1/4}$$

respectively. In practice, we have

$$\sqrt{\frac{\beta}{2\rho A_0}} A^{1/4} \gg Q/A,$$

so W_1 and W_2 always travel in opposite directions. The speed of the forward wave $\lambda_1 \approx \sqrt{\frac{\beta}{2\rho A_0}} A^{1/4}$ is often called pulse wave velocity. Using the relation (3), we can rewrite the equations (1)-(2) as

$$\frac{\partial U}{\partial t} + H(U)\frac{\partial U}{\partial z} = B(U), \quad (z,t) \in [0,L] \times [0,T] \tag{4}$$

where

$$U = \begin{pmatrix} P \\ Q \end{pmatrix},$$

$$H(U) = \begin{pmatrix} 0 & \frac{\beta}{2a(P)A_0} \\ \frac{a(P)^2}{\rho} - \frac{2A_0Q^2}{\beta a^3(P)} & \frac{2Q}{a(P)^2} \end{pmatrix},$$

$$B(U) = \begin{pmatrix} 0 \\ -K_r \frac{Q}{a(P)^2} \end{pmatrix},$$

and

$$a(P) = \frac{A_0}{\beta}(P - P_{ext}) + \sqrt{A_0}.$$

Even though the one-dimensional model can not capture the blood flow details, it can effectively describe the wave propagation within the vessel. This model has been validated by both *in vitro* data [29,30] and *in vivo* data [31,32]. There are many effective numerical solvers, like the MacCormack scheme [3,4], Taylor-Galerkin scheme [5,6], MUSCL [7,8], and the local discontinuous Galerkin scheme [9–11]. A comparison of different numerical methods has also been done in [33].

3. Physics-informed machine learning for PDEs

Machine learning algorithms for PDEs have been extensively studied in recent years. We may use neural networks to solve one instance of PDE or the solution operator of PDE problems. One of the pioneering work is the physics-informed neural networks (PINNs) proposed in [34]. This work aims to solve an instance of PDEs using a neural network. Compared to traditional numerical solvers, the PINN is mesh-free, easy to implement, and can solve high-dimensional problems. However, it can not achieve very high accuracy for non-linear PDEs. Another popular method is the neural operator which aims to learn the mapping from some input spaces to the solution space, e.g., the DeepONet [13] method. The learned network can quickly predict the PDE solution corresponding to the given input. Neural operators can be trained by minimizing either data-driven loss or physics-informed loss. When the physics-informed loss is used, the method is also called the physics-informed neural operators.

3.1. Physics-informed neural networks (PINNs)

Let's consider a general form of a d-dimensional boundary value problem:

$$\mathcal{L}(u)(x) = f(x), \quad x \in \Omega$$

$$\mathcal{B}(u)(x) = g(x), \quad x \in \partial\Omega$$

where $\Omega \in \mathbb{R}^d$ is the problem domain, $\mathcal{L}(u)$ denotes a general differential operator, and $\mathcal{B}(u)$ denotes corresponding boundary conditions. The PINN method first defines a neural network to approximate the solution to this specific instance of PDE: $\mathcal{N}(x;\theta) : \Omega \to \mathbb{R}$, where θ denotes the set of all trainable parameters in the network. Then, this network is trained by solving the optimization problem:

$$\min_\theta \sum_{j=1}^{N_0} |\mathcal{L}(\mathcal{N}(x_j;\theta)) - f(x_j)|^2 + \omega \sum_{j=1}^{N_1} |\mathcal{B}(\hat{x}_j) - g(\hat{x}_j)|^2$$

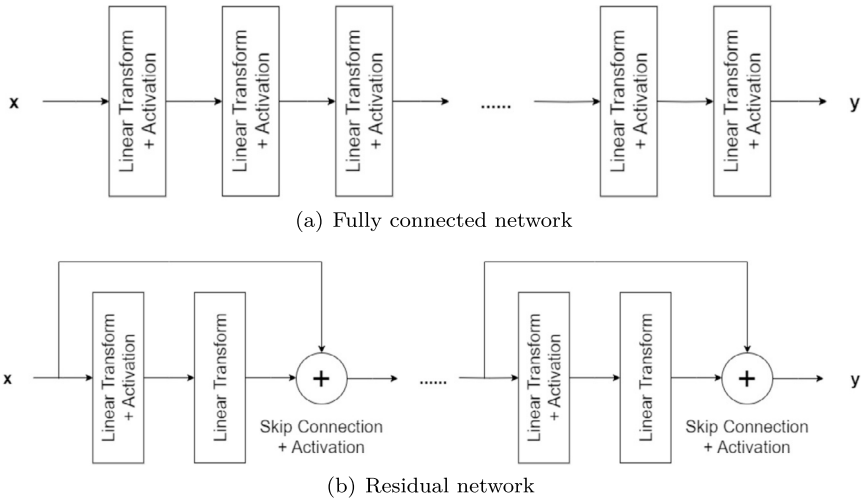


Fig. 3. Structures of the fully connected network and the residual network.

where ω is the weight of the boundary residual, $\{x_j\}_{j=1}^{N_0}$ is collocation points randomly sampled from Ω, and $\{\hat{x}_j\}_{j=1}^{N_1}$ is collocation points randomly generated from $\partial\Omega$. The partial derivatives of $\mathcal{N}$ can be easily calculated through the back-propagation mechanism. An efficient implementation is provided in the DeepXDE package [35] for both PyTorch and TensorFlow platforms. Some similar algorithms in the literature use similar ideas, such as the deep Ritz method (DRM) [36] and the mixed residual method (MIM) [37,38]. The DRM uses the variational formulation of PDEs to train networks, and the MIM transforms high-order PDEs into first-order systems to train networks. Various numerical analyses have also been conducted for these physics-informed methods [39–43].

One popular network structure for the PINN method is the fully connected network (FCN) (Fig. 3(a)). An FCN is typically defined as compositions of many fully connected layers:

$$
\begin{aligned}
&\mathcal{N}_{FCN}(x;\theta) := (\mathcal{F}_L \circ \mathcal{F}_{L-1} \circ \cdots \circ \mathcal{F}_1)(x),\\
&\mathcal{F}_l(z) := \sigma_l(W_l z + b_l), \quad \forall l = 1, \dots, L-1,\\
&W_l \in \mathbb{R}^{d_{l,in} \times d_{l,out}}, \quad b_l \in \mathbb{R}^{d_{l,out}},\\
&d_{1,in} = d, \quad d_{l,in} = d_{l-1,out}, \quad l = 2, \dots, L-1,\\
&\mathcal{F}_L(z) := \sigma_L(W_L z + b_L), \quad W_L \in \mathbb{R}^{d_{L-1,out},1}, \quad b_L \in \mathbb{R},
\end{aligned}
$$

where σ_l is the activation function for each layer, $d_{l,in}$ denotes the input dimension of the lth layer, and $d_{l,out}$ denotes the output dimension of the lth layer. Choices for the activation function usually include Sigmoid, Tanh, ReLU, etc. For the last layer F_L, the activation function can be the identify function, i.e., $\sigma_L(z) = z$. When increasing the depth L, the training of the FCN may become slower because of the vanishing gradient effect. To overcome this limitation, we can use the residual network incorporating the skip connection structure [44]. A residual network (Fig. 3(b)) is defined as:

$$
\begin{aligned}
&\mathcal{N}_{Res}(x;\theta) := (\hat{\mathcal{F}}_L \circ \hat{\mathcal{F}}_{L-1} \circ \cdots \circ \hat{\mathcal{F}}_1)(x),\\
&\hat{\mathcal{F}}_l(z) := \sigma_l(W_{l,2}\sigma_l(W_{l,1} z + b_{l,1}) + b_{l,2} + z), \quad \forall l = 1, \dots, L-1,\\
&W_{l,1} \in \mathbb{R}^{d_{l,in} \times d_{l,out}}, \quad W_{l,2} \in \mathbb{R}^{d_{l,out} \times d_{l,out}}, \quad b_{l,1}, b_{l,2} \in \mathbb{R}^{d_{l,out}},\\
&d_{1,in} = d, \quad d_{l,in} = d_{l-1,out}, \quad l = 2, \dots, L-1,\\
&\mathcal{F}_L(z) := \sigma_L(W_L z + b_L), \quad W_L \in \mathbb{R}^{d_{L-1,out},1}, \quad b_L \in \mathbb{R}.
\end{aligned}
$$

3.2. Neural operators for PDEs

Let's consider a class of parametric PDEs:

$$
\begin{aligned}
&\mathcal{L}(u)(x;\alpha) = f(x;\alpha), \quad x \in \Omega\\
&\mathcal{B}(u;\alpha)(x) = g(x;\alpha), \quad x \in \partial\Omega,
\end{aligned}
$$

where α is the parameters that define the problem; for example, α can be the diffusion coefficient in a convection-diffusion equation. The neural operator aims to approximate the solution operator $\mathcal{M} : \mathcal{A} \to \mathcal{U}$, where $\mathcal{A}$ is the function space for α, and $\mathcal{U}$ is the space

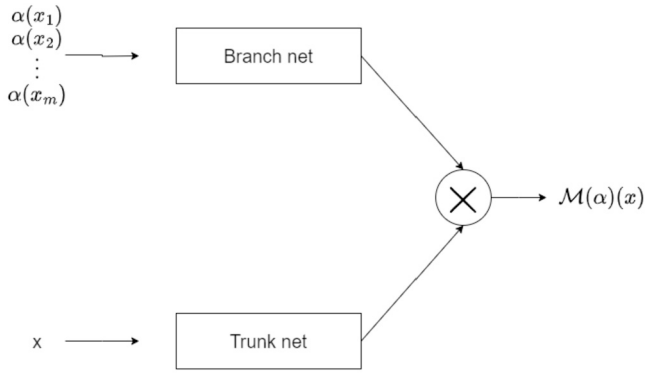


Fig. 4. Structures of the DeepONet.

for the solution u. In practice, the input space can also be other function spaces, e.g., the space of the source term $f(x)$. In this work, we mainly focus on the DeepONet method [13].

A plain DeepONet (Fig. 4(a)) consists of two sub-networks: a branch net and a trunk net. It is defined as

$$\mathcal{N}_{DON}(\hat{\alpha}, x) := Branch(\hat{\alpha})^{\top} Trunk(x)$$

where $\hat{\alpha}$ is the discretization of α on a given mesh, x is any point in Ω. The outputs of the branch net and the trunk net are p-dimensional vectors, where p is a user-specified hyper-parameter. The DeepONet $\mathcal{N}(\hat{\alpha}, x)$ gives a prediction to $\mathcal{M}(\alpha)(x)$. To predict the solution on multiple points, we need to evaluate the trunk net multiple times.

The training of neural operators is fully data-driven. Suppose we have a set of $\{\hat{\alpha}_i\}_{i=1}^N$ from $\mathcal{A}$ and corresponding PDE solutions $\{\hat{u}_i\}_{i=1}^N$ defined on a discretization $\{x_j\}_{j=1}^M$. Then we train the neural operator by minimizing the L_2 difference between the network outputs and the reference solution:

$$\min_{\theta} \frac{1}{MN} \sum_{i=1}^{N} \sum_{j=1}^{M} |\mathcal{N}_{DON}(\hat{\alpha}_i, x_j) - \hat{u}_i(x_j)|^2$$

for the DeepONet. A comprehensive study of neural operators is given in [28], which includes numerical experiments on various linear and non-linear PDEs.

Besides the data-driven training method, we can apply the physics-informed loss like PINN to train neural operators. Such methods are called physics-informed neural operators [45–47]. In many engineering problems, obtaining the entire solution to PDEs may be very difficult or expensive. Consequently, it is more desirable to incorporate the physics laws into the training process to regularize the neural operators. We can construct a similar residual loss as the PINN method for each pair of training samples and minimize the summation over all samples. For the DeepONet, we can use back-propagation to evaluate the derivatives.

4. The physics informed DeepONet for blood flow estimation

4.1. Problem setup

This work aims to learn the operator that maps the physiological signals to the solution of the Navier-Stokes equation:

$$\mathcal{O} : \mathcal{S}([0,T]) \to \mathcal{U}([0,L] \times [0,T]) \tag{5}$$

where $\mathcal{S}([0,T])$ is the set of physiological signals of length T and $\mathcal{U}([0,L]\times[0,T])$ is the space of solutions to Navier-Stokes equations defined on $[0,L]\times[0,T]$.

Suppose we have a training dataset consisting of N pairs of samples:

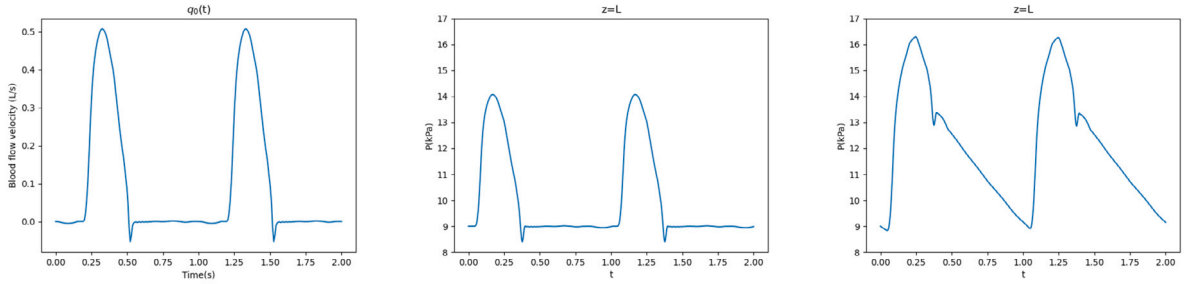
$$\{(s_i, p_i)\}_{i=1}^N,$$

where s_i is a set of physiological signals, and p_i is the corresponding ABP waveform measured in the radial artery that lies in the forearm. All waveform data have the same length. The physiological signal s_i serves as the input to the neural network and the blood pressure waveform p_i is used to define the boundary condition.

For each sample $i = 1, \ldots, N$, its hemodynamics is defined by $U_i = (P_i, Q_i)$. We assume each U_i satisfies the Navier-stokes system described in (4):

$$\frac{\partial U_i}{\partial t} + H(U_i)\frac{\partial U_i}{\partial z} = B(U_i), \quad (z,t) \in [0,L]\times[0,T]. \tag{6}$$

We also need to provide suitable initial and boundary conditions to identify unique PDE solutions. When solving this system using some traditional numerical methods, we usually need to impose an inlet boundary condition and an outlet boundary condition. The



(a) Prescribed inlet boundary condition (b) Simulated blood pressure using the non-reflecting boundary condition (c) Simulated blood pressure using the Windkessel boundary condition

Fig. 5. Comparison of the simulated blood pressure using different outflow boundary conditions. (a) is the inlet boundary condition given at $z = 0$; (b) is the simulated blood pressure at $z = L$ using the non-reflecting outflow boundary condition; and (c) is the simulated blood pressure at $z = L$ using the 3-element Windkessel outflow boundary condition. (b) and (c) are computed using a standard MacCormack scheme.

inlet boundary condition is usually set as a Dirichlet boundary condition which prescribes the value of the blood pressure P or blood flow rate Q at the inlet side. The outlet boundary condition can determine the reflection waves that run backward in the artery. A simple choice for the outlet boundary condition is the non-reflecting boundary condition [1]: $W_2 = 0$ where W_2 is the backward characteristic variable. Though the non-reflecting boundary condition is very easy to implement, it can not provide a physically correct reflecting wave. A better choice is the three-element Windkessel boundary condition [48]:

$$W(P_i, Q_i) := Q_i(L,t)(1 + \frac{R_1}{R_2}) + CR_1\frac{\partial Q_i}{\partial t}(L,t) - \frac{P_i(L,t)}{R_2} - C\frac{\partial P_i}{\partial t}(L,t) = 0, \quad t \in [0,T], \tag{7}$$

where R_1, R_2, and C are some hyper-parameters related to the artery properties.

Remark. The Windkessel model describes the hemodynamics of the arterial system in terms of resistance and compliance, and it is widely used as an outflow boundary condition in the numerical simulation of human arterial systems [29,49,50]. The Navier-Stokes equation with this Windkessel boundary condition can produce physical reflection waves in the solutions. A simple comparison of two outflow boundary conditions is shown in Fig. 5 where we can observe that the blood pressure simulated using the non-reflecting boundary condition is not physically correct.

In our case, since the pressure data p_i, $i = 1, \dots, N$, is measured near the end of radial arteries, it is better to formulate it also as an outlet boundary condition:

$$P_i(L,t) = p_i(t), \quad t \in [0,T]. \tag{8}$$

To better simulate the reflection waves in the artery, we use the three-element Windkessel model (7) as another boundary condition.

For the initial condition, we impose a time-periodic condition on both P_i and Q_i instead of giving a prescribed initial value:

$$U_i(z, t + \delta_i) = U_i(z,t), \quad (z,t) \in [0,L] \times [0,T] \tag{9}$$

where δ_i is the period for the i-th training sample, and it can be estimated from the physiological signals s_i. Typically, the waveforms are almost periodic within a short time.

Remark. Conventional methods directly prescribe the initial value of solutions. One commonly used initial condition is $U(z,0) = (P_{ext}, 0)$. However, it is very difficult to measure the value of P_i and Q_i at the initial time for each sample and it is not realistic to assume an initial condition like $U(z,0) = (P_{ext}, 0)$. Therefore, using a time-periodic condition is more suitable in our situation.

By solving the system (4) with the boundary condition (8) (7) and initial condition (9), we would be able to predict the hemodynamics in the radial artery. For conventional numerical methods, it would be difficult to solve under this setting, because there is no prescribed initial value given and there is no boundary condition at the inlet side; but it is easy to handle by the physics-informed method.

In this problem, the period for each sample is different, which will cause some difficulty in computing the loss function. In this work, we will propose a new efficient implementation method to handle this issue.

4.2. BP-DeepONet

We propose a new physics-informed DeepONet, BP-DeepONet, to learn the solution operator (5). We adopt the same structure as the original DeepONet. For the branch net, we use a structure that combines a one-dimensional convolutional ResNet and a bi-directional LSTM, see Fig. 6. The inputs to the branch net are some segments of physiological signals s_i and the output would be two

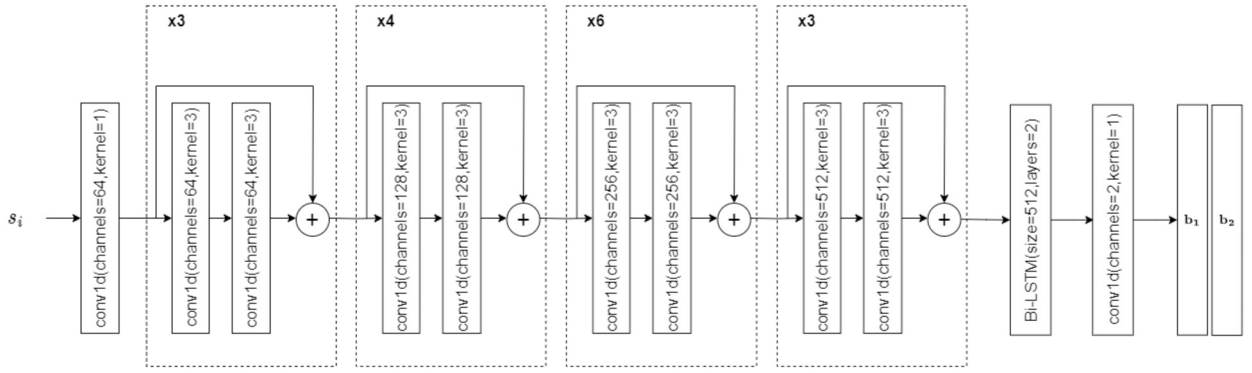


Fig. 6. Structure of the branch net.

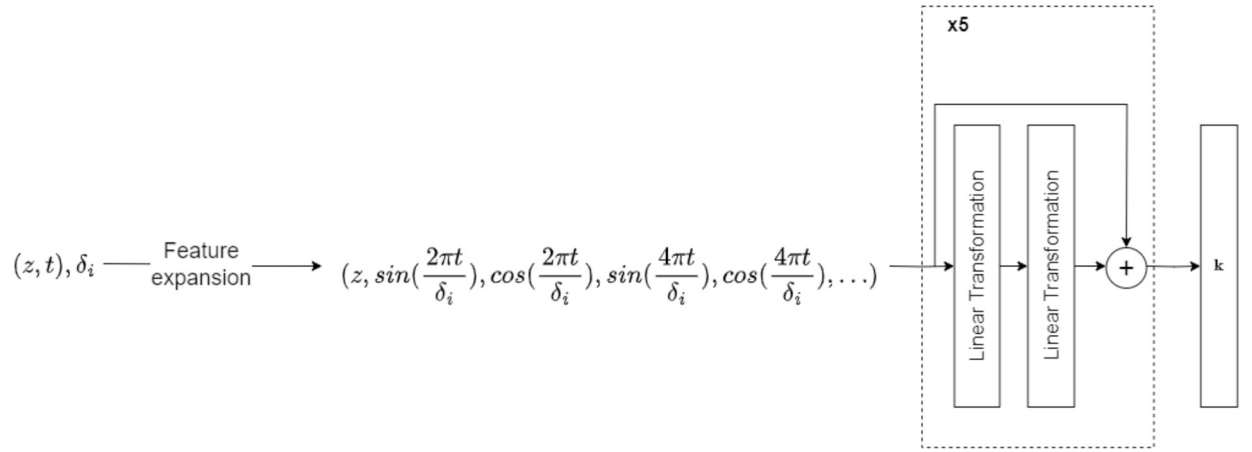


Fig. 7. Structure of the trunk net.

latent vectors $\mathbf{b_1}$ and $\mathbf{b_2}$. conv1d(channels = c, kernel = s) represents a standard one-dimensional convolutional layer whose number of output channels is c and kernel size is s. Bi-LSTM (size = h, layers = l) represents a bi-directional LSTM block whose input size and hidden size are h, and the number of layers is l. Such a structure has been shown in [51] that performs the best in extracting features from physiological signals and predicting blood pressure. For the trunk net, similar to the original implementation of DeepONet [13], we adopt the fully connected network structure with residual connection as Fig. 3(b) with the feature expansion (Fig. 7). The residual connection can improve the efficiency of training deep networks and has been used in physics-informed machine learning problems as well [36]. Feature expansion [52] is a technique to impose the periodic condition on PINN. To impose the time-periodic constraints (9), we can simply expand the inputs (z,t) to

$$(z, \sin(\frac{2\pi t}{\delta_i}), \cos(\frac{2\pi t}{\delta_i}), \sin(\frac{4\pi t}{\delta_i}), \cos(\frac{4\pi t}{\delta_i}), \dots).$$

Then, our network outputs will be strictly time-periodic with period δ_i. The output of the trunk net is a vector $\mathbf{k}$ whose dimension is the same as $\mathbf{b_1}$ and $\mathbf{b_2}$. The final prediction of the BP-DeepONet will be calculated as follows:

$$P_\theta(s_i,\delta_i,(z,t)) = \mathbf{b}_1(s_i)^\top \mathbf{k}(z,t,\delta_i), \quad Q_\theta(s_i,\delta_i,(z,t)) = \mathbf{b}_2(s_i)^\top \mathbf{k}(z,t,\delta_i)$$

where θ is the set of all trainable parameters in the BP-DeepONet.

The training loss function of our network is defined as

$$\mathcal{L}(P_\theta,Q_\theta) = \mathcal{R}_{pde}(P_\theta,Q_\theta) + \omega_1 \mathcal{R}_{b_1}(P_\theta) + \omega_2 \mathcal{R}_{b_2}(P_\theta,Q_\theta) \tag{10}$$

where

$$\mathcal{R}_{pde}(P_\theta,Q_\theta) = \frac{1}{N}\sum_{i=1}^{N} \left\| \left(\frac{\partial}{\partial t}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} + H(P_\theta,Q_\theta)\frac{\partial}{\partial z}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} - B(P_\theta,Q_\theta) \right)(s_i,\delta_i,\cdot) \right\|^2_{L^2([0,L]\times[0,T])},$$

$$\mathcal{R}_{b_1}(P_\theta) = \frac{1}{N}\sum_{i=1}^{N} \left\| P_\theta(s_i,\delta_i,(L,\cdot)) - p_i \right\|^2_{L^2([0,T])},$$

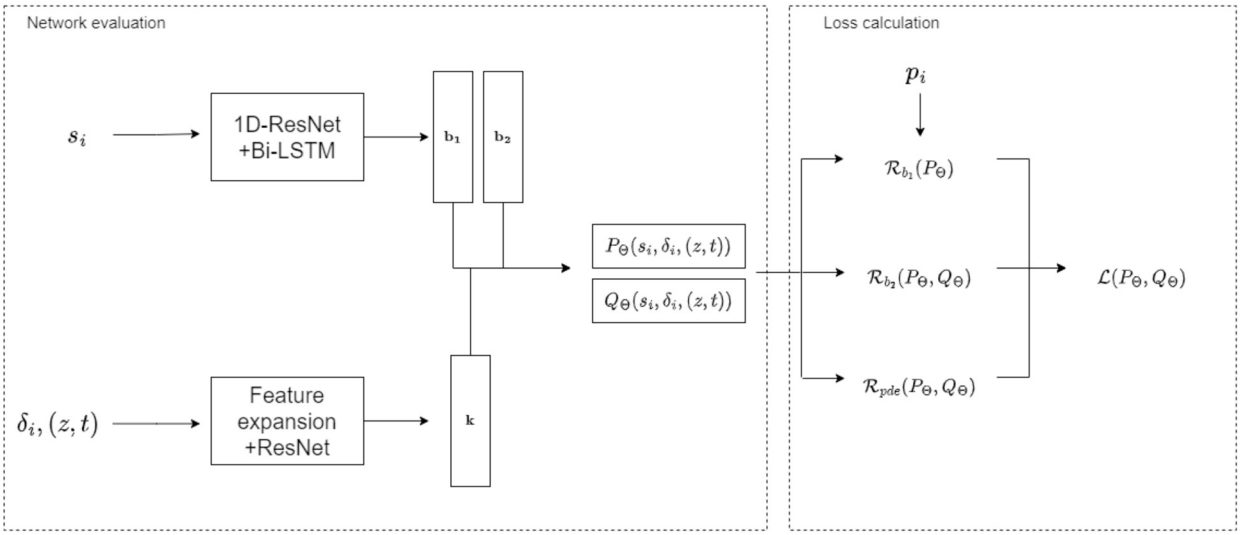


Fig. 8. Structure of the BP-DeepONet.

and

$$\mathcal{R}_{b_2}(P_\theta, Q_\theta) = \frac{1}{N}\sum_{i=1}^{N} \left\| W(P_\theta, Q_\theta)(s_i, \delta_i, (L,\cdot)) \right\|^2_{L^2([0,T])}.$$

The complete structure of the BP-DeepONet is displayed in Fig. 8.

To evaluate the different residual terms $\mathcal{R}_{pde}$, $\mathcal{R}_{b1}$, and $\mathcal{R}_{b2}$, we need to approximate the L_2 norm with some discrete quadrature. For the PDE residual term $\mathcal{R}_{pde}$, we can simply approximate the i-th L_2 residual by

$$\begin{aligned}
&\left\| \left(\frac{\partial}{\partial t}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} + H(P_\theta, Q_\theta)\frac{\partial}{\partial z}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} - B(P_\theta, Q_\theta) \right)(s_i, \delta_i, \cdot) \right\|^2_{L^2([0,L]\times[0,T])} \\
&\approx \sum_{j=1}^{M} \left| \left(\frac{\partial}{\partial t}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} + H(P_\theta, Q_\theta)\frac{\partial}{\partial z}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} - B(P_\theta, Q_\theta) \right)(s_i, \delta_i, (z_j, t_j)) \right|^2,
\end{aligned}$$

where $\{(z_j, t_j)\}_{j=1}^{M}$ is a set of quadrature points randomly sampled from the domain $[0, L] \times [0, T]$. The partial derivatives in the residual terms can be calculated as

$$\begin{aligned}
&\frac{\partial}{\partial x} P_\theta(s_i, \delta_i, (z_j, t_j)) = \mathbf{b_1}(s_i)^\top \frac{\partial}{\partial x}\mathbf{k}(\delta_i, (z_j, t_j)), \quad x = z \text{ or } t, \\
&\frac{\partial}{\partial x} Q_\theta(s_i, \delta_i, (z_j, t_j)) = \mathbf{b_2}(s_i)^\top \frac{\partial}{\partial x}\mathbf{k}(\delta_i, (z_j, t_j)), \quad x = z \text{ or } t.
\end{aligned}$$

However, evaluating this approximation is very time-consuming because the total number of gradient evaluations in one training epoch is $O(MN)$. It is necessary to design a more efficient method to approximate the residuals. This can be achieved by utilizing the periodicity of our BP-DeepONet.

4.3. Efficient implementation of training BP-DeepONet

Given any periodic function $f(t)$ with period δ, we have the norm $\|\cdot\|_{L^2([0,T])}$ is equivalent to $\|\cdot\|_{L^2([0,\delta])}$:

$$\left\lfloor \frac{T}{\delta} \right\rfloor \|f\|^2_{L^2([0,\delta])} \le \|f\|^2_{L^2([0,T])} \le \left\lceil \frac{T}{\delta} \right\rceil \|f\|^2_{L^2([0,\delta])},$$

where $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ denote the floor function and ceiling function, respectively. Then, for the i-th L_2 residual term in $\mathcal{R}_{pde}$, we may replace it with

$$\begin{aligned}
&\left\| \left(\frac{\partial}{\partial t}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} + H(P_\theta, Q_\theta)\frac{\partial}{\partial z}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} - B(P_\theta, Q_\theta) \right)(s_i, \delta_i, \cdot) \right\|^2_{L^2([0,L]\times[0,\delta_i])} \\
&= \int_0^{\delta_i}\int_0^{L} \left| \left(\frac{\partial}{\partial t}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} + H(P_\theta, Q_\theta)\frac{\partial}{\partial z}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} - B(P_\theta, Q_\theta) \right)(s_i, \delta_i, (z,t)) \right|^2 dz dt
\end{aligned}$$

$$= \int_0^{\delta_i}\int_0^{L} \left| \left(\frac{\partial}{\partial t}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} + H(P_\theta, Q_\theta)\frac{\partial}{\partial z}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} - B(P_\theta, Q_\theta) \right)(s_i, 1, (z, t/\delta_i)) \right|^2 dz dt.$$

After a change of variable $\hat{t} = t/\delta_i$, the above integration can be written as

$$\int_0^{1}\int_0^{L} \left| \left(\frac{\partial}{\partial \hat{t}}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} /\delta_i + H(P_\theta, Q_\theta)\frac{\partial}{\partial z}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} - B(P_\theta, Q_\theta) \right)(s_i, 1, (z, \hat{t})) \right|^2 dz d\hat{t}$$

$$\approx \sum_{j=1}^{M} \left| \left(\frac{\partial}{\partial \hat{t}}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} /\delta_i + H(P_\theta, Q_\theta)\frac{\partial}{\partial z}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} - B(P_\theta, Q_\theta) \right)(s_i, 1, (z_j, \hat{t}_j)) \right|^2$$

where $\{(z_j, \hat{t}_j)\}_{j=1}^M$ is randomly sampled from the domain $[0, L] \times [0, 1]$. We approximate $\mathcal{R}_{pde}$ by

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} \left| \left(\frac{\partial}{\partial \hat{t}}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} /\delta_i + H(P_\theta, Q_\theta)\frac{\partial}{\partial z}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} - B(P_\theta, Q_\theta) \right)(s_i, 1, (z_j, \hat{t}_j)) \right|^2 \tag{11}$$

Then, the partial derivatives in (11) can be calculated as

$$\frac{\partial}{\partial x} P_\theta(s_i, \delta_i, (z_j, \hat{t}_j)) = \mathbf{b_1}(s_i)^\top \frac{\partial}{\partial x}\mathbf{k}(1, (z_j, \hat{t}_j)), \quad x = z \text{ or } \hat{t},$$
$$\frac{\partial}{\partial x} Q_\theta(s_i, 1, (z_j, \hat{t}_j)) = \mathbf{b_2}(s_i)^\top \frac{\partial}{\partial x}\mathbf{k}(1, (z_j, \hat{t}_j)), \quad x = z \text{ or } \hat{t}.$$

For different i, we do not need to recompute the gradient of $\mathbf{k}$, so the number of gradient evaluations in one training epoch is only $O(M)$. We can use the same idea to approximate $\mathcal{R}_{b2}$ by

$$\mathcal{R}_{b2}(P_\theta, Q_\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} \left| \left(Q_\theta(1 + R_1/R_2) + CR_1\frac{\partial}{\partial \hat{t}}Q_\theta - P_\theta/R_2 - C\frac{\partial}{\partial \hat{t}}P_\theta \right)(s_i, 1, (L, \hat{t}_j)). \right|^2 \tag{12}$$

For $\mathcal{R}_{b1}$, we simply approximate it by

$$\mathcal{R}_{b1}(P_\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} \left| P_\theta(L, \hat{t}_j) - \bar{p}_i(\hat{t}_j\delta_i) \right|^2 \tag{13}$$

where $\{\hat{t}_j\}_{j=1}^M$ is randomly sampled from the domain $[0, 1]$ and $\bar{p}_i$ is the average of p_i over all periods contained in $[0, T]$. Finally, the entire empirical loss function for training the BP-DeepONet is the combination of three terms (11), (12), and (13):

$$\begin{aligned}\hat{\mathcal{L}}(P_\theta, Q_\theta) = &\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} \left| \left(\frac{\partial}{\partial \hat{t}}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} /\delta_i + H(P_\theta, Q_\theta)\frac{\partial}{\partial z}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} - B(P_\theta, Q_\theta) \right)(s_i, 1, (z_j, \hat{t}_j)) \right|^2 \\ &+ \frac{\omega_1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} \left| P_\theta(s_i, 1, (L, \hat{t}_j)) - \bar{p}_i(\hat{t}_j\delta_i) \right|^2 \\ &+ \frac{\omega_2}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} \left| \left(Q_\theta(1 + R_1/R_2) + CR_1\frac{\partial}{\partial \hat{t}}Q_\theta - P_\theta/R_2 - C\frac{\partial}{\partial \hat{t}}P_\theta \right)(s_i, 1, (L, \hat{t}_j)) \right|^2.\end{aligned}$$

4.4. Hyper BP-DeepONet for sample-dependent hyper-parameters estimation

In the Navier-Stokes equation, many hyper-parameters depend on the physical properties of arterial systems. In the previous BP-DeepONet, we set all these hyper-parameters to be subject-independent, i.e., we use the same set of hyper-parameters for all samples. However, in real situations, these parameters differ from person to person, so it is unrealistic to set them all as constants. To handle this issue, we generalize the model (6) to be sample-dependent. For each training sample i from 1 to N, we denote the set of all hyper-parameters for the i-th sample as

$$\gamma_i := \{\beta_i, A_{0,i}, \rho_i, K_{r,i}, P_{ext,i}, R_{1,i}, R_{2,i}, C_i\},$$

and we assume the solutions P_i, Q_i for the i-th sample satisfy the following PDE:

$$\frac{\partial U_i}{\partial t} + H(U_i, \gamma_i)\frac{\partial U_i}{\partial z} = B(U_i, \gamma_i), \quad (z, t) \in [0, L] \times [0, T] \tag{14}$$

where

$$U_i = \begin{pmatrix} P_i \\ Q_i \end{pmatrix},$$

$$H(U_i, \gamma_i) = \begin{pmatrix} 0 & \frac{\beta_i}{2a(P_i,\gamma_i)A_{0,i}} \\ \frac{a(P_i,\gamma_i)^2}{\rho_i} - \frac{2A_{0,i}Q_i^2}{\beta_i a^3(P_i,\gamma_i)} & \frac{2Q_i}{a(P_i,\gamma_i)^2} \end{pmatrix},$$

$$B(U_i, \gamma_i) = \begin{pmatrix} 0 \\ -K_{r,i}\frac{Q_i}{a(P_i,\gamma_i)^2} \end{pmatrix},$$

and

$$a(P_i, \gamma_i) = \frac{A_{0,i}}{\beta_i}(P_i - P_{ext,i}) + \sqrt{A_{0,i}}.$$

The corresponding boundary and periodic conditions are:

$$\begin{aligned} &P_i(L,t) = p_i(t), \quad t \in [0,T], \\ &W(P_i, Q_i, \gamma_i) := Q_i(L,t)(1 + \frac{R_{1,i}}{R_{2,i}}) + C_i R_{1,i}\frac{\partial Q_i}{\partial t}(L,t) - \frac{P_i(L,t)}{R_{2,i}} - C_i\frac{\partial P_i}{\partial t}(L,t) = 0, \quad t \in [0,T], \end{aligned} \tag{15}$$

and

$$U_i(z, t + \delta_i) = U_i(z,t), \quad z \in [0, L].$$

It is almost impossible to measure γ_i for each sample. To address this issue, we further define a hyper-parameter network on top of the branch net and we name this new method the hyper BP-DeepONet. This network takes $\mathbf{b}_1$ and $\mathbf{b}_2$ as inputs and outputs the estimation of γ_i. This network is optimized together with the BP-DeepONet. We also impose extra constraints to regularize the output of the hyper-network. The first constraint is imposed as:

$$\int_0^T P_i(z,t)dt = \int_0^T p_i(t)dt, \quad \forall z \in [0, L], \tag{16}$$

which implies that the mean blood pressure does not change during the wave propagation. This phenomenon was found in clinical observations [53]. The second constraint is imposed as

$$R_{1,i} + R_{2,i} = \frac{\int_0^{\delta_i} P_i(L,t)dt}{\int_0^{\delta_i} Q_i(L,t)dt}. \tag{17}$$

This equation can be derived by integrating the equation (15) over one period $[0, \delta_i]$ and it is used for regularizing the estimation of $R_{1,i}$ and $R_{2,i}$. If we have other information about the hemodynamics, like the pulse wave velocity, we may introduce them as constraints to regularize the hyper-parameter network. Then, the training loss function of the hyper BP-DeepONet is

$$\begin{aligned} \mathcal{L}_{meta}(P_\theta, Q_\theta, \gamma_\theta) =& \mathcal{R}_{pde}(P_\theta, Q_\theta, \gamma_\theta) \\ &+ \omega_1\mathcal{R}_{b_1}(P_\theta) + \omega_2\mathcal{R}_{b_2}(P_\theta, Q_\theta, \gamma_\theta) + \omega_3\mathcal{R}_{c_1}(P_\theta, Q_\theta) + \omega_4\mathcal{R}_{c_2}(P_\theta, Q_\theta, \gamma_\theta) \end{aligned} \tag{18}$$

where Γ_θ is the set of estimated hyper-parameters given by the hyper-parameter network:

$$\gamma_\theta(s_i) = \{\beta_\theta(s_i), A_{0,\theta}(s_i), \rho_\theta(s_i), K_{r,\theta}(s_i), P_{ext,\theta}(s_i), R_{1,\theta}(s_i), R_{2,\theta}(s_i), C_\theta(s_i)\}, \quad i = 1, \dots, N,$$

and the residual terms in (18) are defined as

$$\mathcal{R}_{pde}(P_\theta, Q_\theta, \gamma_\theta) = \frac{1}{N}\sum_{i=1}^N \left\| \left(\frac{\partial}{\partial t}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} + H(P_\theta, Q_\theta, \gamma_\theta)\frac{\partial}{\partial z}\begin{pmatrix} P_\theta \\ Q_\theta \end{pmatrix} - B(P_\theta, Q_\theta, \gamma_\theta) \right)(s_i, \delta_i, \cdot) \right\|^2_{L^2([0,L]\times[0,T])},$$

$$\mathcal{R}_{b_2}(P_\theta, Q_\theta, \gamma_\theta) = \frac{1}{N}\sum_{i=1}^N \left\| W(P_\theta, Q_\theta, \gamma_\theta)(s_i, \delta_i, (L, \cdot)) \right\|^2_{L^2([0,T])},$$

$$\mathcal{R}_{c_1}(P_\theta, Q_\theta) = \frac{1}{N}\sum_{i=1}^N \left\| \int_0^T P_\theta(s_i, \delta_i, (\cdot, t))dt - \int_0^T p_i(t)dt \right\|^2_{L^2([0,L])},$$

$$\mathcal{R}_{c_2}(P_\theta, Q_\theta, \gamma_\theta) = \frac{1}{N}\sum_{i=1}^N \left(R_{1,\theta}(s_i) + R_{2,\theta}(s_i) - \frac{\int_0^{\delta_i} P_\theta(s_i, \delta_i, (L,t))dt}{\int_0^{\delta_i} Q_\theta(s_i, \delta_i, (L,t))dt} \right)^2,$$

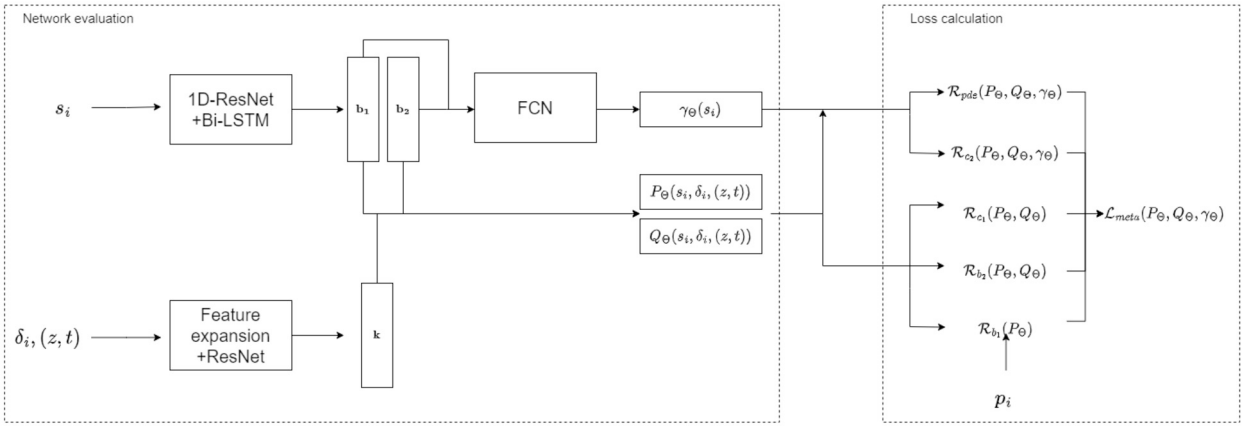


Fig. 9. Structures of the hyper BP-DeepONet.

and $\mathcal{R}_{b_1}(P_\theta)$ is defined the same as (10). To approximate these residual terms, we use the same change of variable technique as section 4.2:

$$\mathcal{R}_{pde}(P_\theta, Q_\theta, \gamma_\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|\left(\frac{\partial}{\partial t}\begin{pmatrix}P_\theta\\ Q_\theta\end{pmatrix} + H(P_\theta, Q_\theta, \gamma_\theta)\frac{\partial}{\partial z}\begin{pmatrix}P_\theta\\ Q_\theta\end{pmatrix} - B(P_\theta, Q_\theta, \gamma_\theta)\right)(s_i, 1, (z_j, \hat{t}_j))\right|^2,$$

$$\mathcal{R}_{b_2}(P_\theta, Q_\theta, \gamma_\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|\left(Q_\theta(1 + R_{1,\theta}/R_{2,\theta}) + C_\theta R_{1,\theta}\frac{\partial}{\partial \hat{t}}Q_\theta - P_\theta/R_{2,\theta} - C_\theta\frac{\partial}{\partial \hat{t}}P_\theta\right)(s_i, 1, (L, \hat{t}_j))\right|^2,$$

$$\mathcal{R}_{c_1}(P_\theta, Q_\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\sum_{j_1=1}^{M'}\left|\sum_{j_2=1}^{M}(P_\theta(s_i, 1, (\hat{z}_{j_1}, \hat{t}_{j_2})) - p_i(\hat{t}_{j_2}))\right|^2,$$

$$\mathcal{R}_{c_2}(P_\theta, Q_\theta, \gamma_\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\left|R_{1,\theta}(s_i) + R_{2,\theta}(s_i) - \frac{\sum_{j=1}^{M}P_\theta(s_i, \delta_i, (L, \hat{t}_j))}{\sum_{j=1}^{M}Q_\theta(s_i, \delta_i, (L, \hat{t}_j))}\right|^2,$$

where $\{(z_j, \hat{t}_j)\}_{j=1}^{M}$ is a set of quadrature points sampled from $[0, L] \times [0, 1]$ and $\{\hat{z}_j\}_{j=1}^{M'}$ is a set of quadrature points sampled from $[0, L]$, and $\mathcal{R}_{b_1}(P_\theta)$ is approximated in the same way as (13). Then, the empirical training loss for the hyper BP-DeepONet is given as

$$\begin{aligned}\hat{\mathcal{L}}_{hyper}(P_\theta, Q_\theta, \gamma_\theta) =& \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|\left(\frac{\partial}{\partial t}\begin{pmatrix}P_\theta\\ Q_\theta\end{pmatrix} + H(P_\theta, Q_\theta, \gamma_\theta)\frac{\partial}{\partial z}\begin{pmatrix}P_\theta\\ Q_\theta\end{pmatrix} - B(P_\theta, Q_\theta, \gamma_\theta)\right)(s_i, 1, (z_j, \hat{t}_j))\right|^2\\ &+ \frac{\omega_1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|P_\theta(s_i, 1, (L, \hat{t}_j)) - \bar{p}_i(\hat{t}_j\delta_i)\right|^2\\ &+ \frac{\omega_2}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|\left(Q_\theta(1 + R_{1,\theta}/R_{2,\theta}) + C_\theta R_{1,\theta}\frac{\partial}{\partial \hat{t}}Q_\theta - P_\theta/R_{2,\theta} - C_\theta\frac{\partial}{\partial \hat{t}}P_\theta\right)(s_i, 1, (L, \hat{t}_j))\right|^2\\ &+ \frac{\omega_3}{N}\sum_{i=1}^{N}\sum_{j_1=1}^{M'}\left|\sum_{j_2=1}^{M}(P_\theta(s_i, 1, (\hat{z}_{j_1}, \hat{t}_{j_2})) - p_i(\hat{t}_{j_2}))\right|^2\\ &+ \frac{\omega_4}{N}\sum_{i=1}^{N}\left|R_{1,\theta}(s_i) + R_{2,\theta}(s_i) - \frac{\sum_{j=1}^{M}P_\theta(s_i, \delta_i, (L, \hat{t}_j))}{\sum_{j=1}^{M}Q_\theta(s_i, \delta_i, (L, \hat{t}_j))}\right|^2.\end{aligned}$$

The overall structures of the hyper BP-DeepONet are given in Fig. 9 where the hyper-parameter network is implemented as a simple FCN.

Remark. During the training of the hyper BP-DeepONet, we only require the measurement of blood pressure at $z = L$. Then the whole solution, all boundary conditions, initial conditions, and hyper-parameters can be automatically learned from this measurement. No extra measurements and parameters are needed.

Remark. Identifying all hyper-parameters is an ill-posed problem when the given information about the flow dynamic is limited, which is similar to the three-element Windkessel problem considered in [54,55]. Thus, we add regularization terms R_{c1} and R_{c2} in

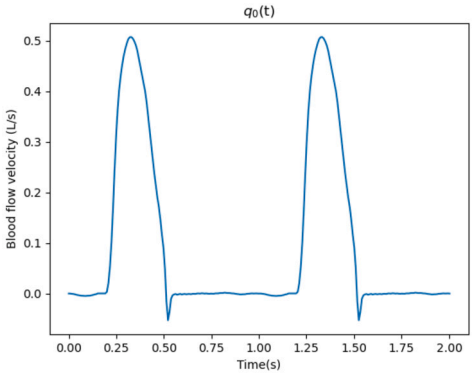


Fig. 10. The inflow boundary condition.

the training loss, which can help the proposed approach to find a set of hyper-parameters that can produce physically reasonable blood flow. More regularization terms can be introduced if we know more information about the patient, like the blood flow velocity and the pulse wave velocity.

5. Numerical experiments of the PINN on simulated data

In this section, we would like to validate the proposed physics-informed training method by solving one PDE instance. We first generate a solution to the Navier-Stokes equation (4) using a MacCormack scheme with a prescribed initial condition and inlet boundary condition. Then, we solve this equation with the boundary conditions (8) and (7) and the time-periodic condition (9) using a simple PINN. We can see in the numerical results that the PINN solution is very close to the simulated solution. We consider the 1D Navier-stokes equation (1) and (2) with the initial condition:

$$A(x,0) = A_0, \quad Q(x,0) = 0,$$

the inflow boundary condition:

$$Q(0,t) = q_0(t),$$

and the three-element Windkessel boundary condition

$$Q(L,t)(1+\frac{R_1}{R_2}) + CR_1\frac{\partial Q}{\partial t}(L,t) = \frac{P(L,t)}{R_2} + C\frac{\partial P}{\partial t}(L,t).$$

Such an initial boundary value problem can be efficiently solved by many classical methods like the MacCormack scheme [3,4], Taylor-Galerkin scheme [5,6], MUSCL [7,8], and the local discontinuous Galerkin scheme [9–11]. We use a numerical example that is similar to the baseline aorta case provided in [50], where the hyper-parameters are chosen as $L = 25$ cm, $A_0 = 4.52$ cm^2, $\beta = 1134.37$ kg/s^2, $\rho = 1060$ kg/m^3, $P_{ext} = 9$ kPa, $R_1 = 1.17 \times 10^7$ Pa s m^{-3}, $R_1 = 1.12 \times 10^8$ Pa s m^{-3}, $C = 1.01 \times 10^{-8}$ m^3 Pa^{-1}, $\upsilon = 0.01 \times 10^6$ m^2/s^2, and the inflow boundary conditions $q_0(t)$ is given in Fig. 10. Then, we solve this problem using the MacCormack scheme.

After having a referenced numerical solution $\hat{A}$ and $\hat{Q}$, we define a residual network $N_\theta(z,t) = (P_\theta(z,t), Q_\theta(z,t))$ shown in Fig. 3 (b) with five residual layers. The time-periodic condition is incorporated into the network using the same feature expansion technique described before. We train this network by minimizing the following loss function:

$$L_{PINN}(\theta) := \left\| \frac{\partial A_\theta}{\partial t} + \frac{\partial Q_\theta}{\partial z} \right\|^2_{L^2([0,L]\times[0,T])} + \left\| \frac{\partial Q_\theta}{\partial t} + \frac{\partial}{\partial z}(\frac{Q_\theta^2}{A_\theta}) + \frac{A_\theta}{\rho}\frac{\partial P_\theta}{\partial z} + K_r\frac{Q_\theta}{A_\theta} \right\|^2_{L^2([0,L]\times[0,T])}$$
$$+ 100\left(\|P_\theta(L,\cdot) - \hat{P}(L,\cdot)\|^2_{L^2([0,T])} + \left\| Q_\theta(L,\cdot)(1+\frac{R_1}{R_2}) + CR_1\frac{\partial Q_\theta}{\partial t}(L,\cdot) - \frac{P_\theta(L,\cdot)}{R_2} - C\frac{\partial P_\theta}{\partial t}(L,\cdot) \right\|^2_{L^2([0,T])} \right)$$

where $P_\theta(z,t) = P_{ext} + \frac{\beta}{A_0}(\sqrt{A_\theta(z,t)} - \sqrt{A_0})$.

We use a standard Adam optimizer with a learning rate of 1e-4, and the total number of training epochs is 50,000. The training is conducted on an NVIDIA RTX A6000 gpu and takes 1.8 hour. We randomly sample 1,000 quadrature points in the domain in each epoch to approximate the L^2 integration. We can see the comparison of the PINN solution and the numerical solution by the MacCormack scheme in Fig. 11. The PINN solution solved using the loss function L_{PINN} is very close to the solution of the MacCormack scheme.

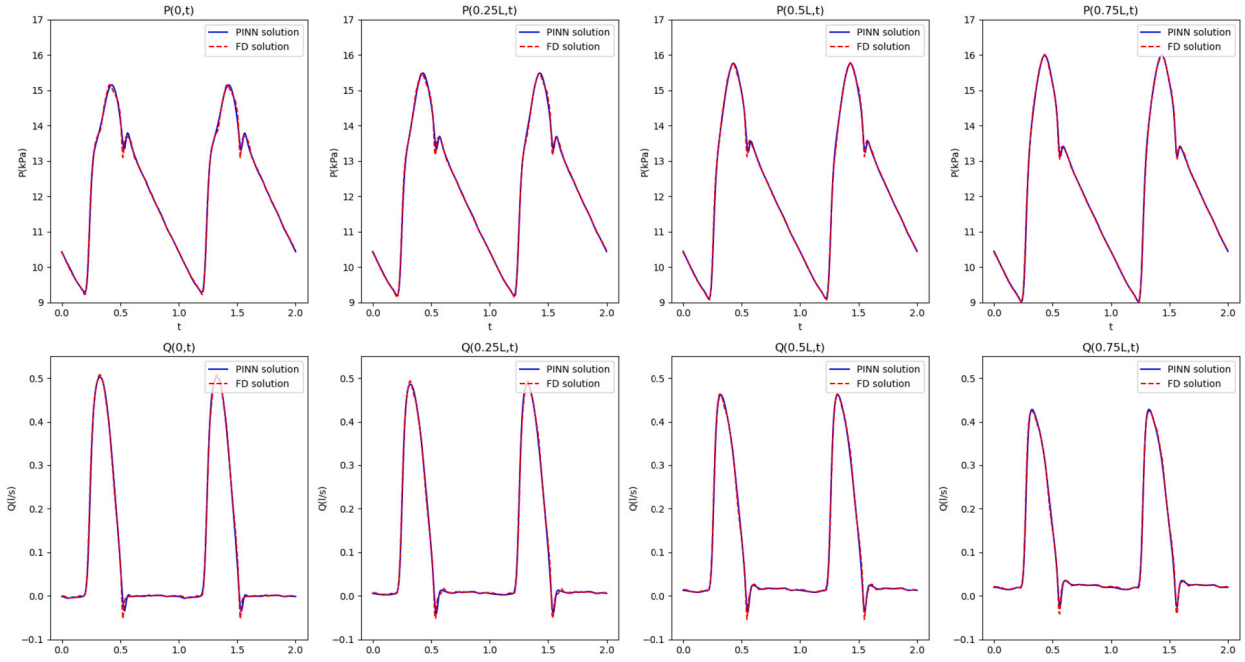


Fig. 11. Comparison of the PINN solution and the finite difference solution (MacCormack).

We also calculate the relative error:

$$\left\| \frac{\hat{P} - P_\theta}{\hat{P}} \right\|_1 \approx 0.30\%$$

and

$$\frac{\|\hat{Q} - Q_\theta\|_1}{\|\hat{Q}\|_\infty} \approx 0.67\%.$$

These two metrics are also used in [50] to measure the error of their numerical solutions for the Navier-Stokes equation. From Fig. 11, we can observe that the error of the PINN solution is larger near the dicrotic notch where the solution is near singular. This case is very common when using the standard PINN method to solve PDEs, and it can be improved by using some adaptive sampling scheme [56]. We may consider this direction in our future work.

6. Numerical experiments of proposed methods on clinical data

6.1. Dataset description

We conducted our experiments on the MIMIC dataset [57], the largest and most commonly used dataset for cuffless blood pressure estimation. We use the pre-processed version of the MIMIC data from [58], which contains recordings of 12,000 subjects after simple pre-processing. Each recording consists of continuous ECG, PPG, and ABP waveform data. The length of each recording is within 10 minutes. All these data are collected from ICU patients, and the invasive method collects the ABP waveform data. Furthermore, we also apply some extra pre-processing steps to these data:

1. Use a bandpass filter to remove noise from PPG and ABP signals.
2. Align the PPG and ABP signals.
3. Randomly select 75% of subjects for training and 25% for testing.
4. 20% of the training set is selected as the validation set.
5. The first 15% of data in each testing subject is used for calibration.
6. Split each recording into many 4-second segments without overlapping.
7. Remove those segments that are highly non-periodic.
8. Remove subjects whose number of valid segments is less than 10.
9. Apply some peak detection algorithms to detect all peaks in each PPG segment, then calculate the average length of intervals between two consecutive peaks.

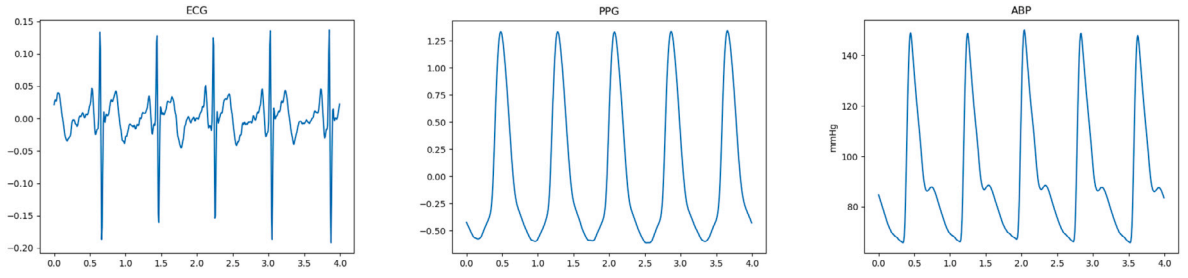


Fig. 12. One segment of ECG, PPG, and ABP signals.

After pre-processing, the total number of training and testing samples is about 70,000. Step 5 is also called individual fine-tuning [59], and it is a necessary process to calibrate and improve the blood pressure measurement. The segments for calibration will not be used when calculating the testing error. One segment of the recording is shown in Fig. 12.

6.2. Training procedures

For the training of the BP-DeepONet and the hyper BP-DeepONet, we apply the standard Adam optimizer with a scheduled learning rate to train the network. The initial learning rate is set to be 1e-4, and decays by a factor of 0.9 every 5 epochs. After 200 epochs, we restart the learning rate with 1e-4 and continue to train with the same learning rate schedule. After 3 rounds of learning rate cycles, we choose the best model based on the performance evaluated on a validation set. All the training is done in an NVIDIA RTX A6000 gpu. It takes 90.9 hours for the BP-DeepONet and 110.0 hours for the hyper BP-DeepONet.

When training the BP-DeepONet, we need first to determine all the hyper-parameters in the NS model, so we set hyper-parameters based on the reference values of the radial artery described in [60] and estimate the Windkessel parameters using the method provided in [50]: $A_0 = 8\times10^{-6}\ m^2$, $\beta = 529\ kg/s^2$, $L = 0.25m$, $K_r = 8 * \pi * 0.01 \approx 0.251$, $\rho = 1.06\ kg/m^3$, $P_{ext} = 6.67\ kPa$, $R_1 = 12.37\ Pa\cdot s/m^3$, $R_2 = 137.5\ Pa\cdot s/m^3$, $C = 0.01\ m^3/Pa$. In the hyper BP-DeepONet, we also use these values as referenced values for the predicted hyper-parameters. We restrict the deviation of the predicted hyper-parameters from the referenced values to be less than or equal to 20% so that the estimation of the hyper-parameters will always be reasonable.

During the training of both BP-DeepONet and hyper BP-DeepONet, we use a batch size of 100. All the training is done on a GPU server with 4 NVIDIA RTX A6000 GPUs.

6.3. Numerical results

After finishing the training of DeepONet, we can obtain the prediction of the BP waveform at any given location z by evaluating the DeepONet at $(z,t_0), (z,t_1), \ldots, (z,t_{N_h})$, where $\{t_j\}_{j=0}^{N_h}$ is a uniform discretization of the domain $[0,T]$.

We first evaluate the waveform prediction at the outlet side $z = L$. Since we have the measured BP wave at $z = L$ by the invasive method, we can compute the prediction error of our method at this location. We evaluate the error using several metrics: waveform MAE, SBP MAE, DBP MAE, and MBP MAE, where MAE is short for mean absolute error. Suppose $P_i = (P_\theta(s_i,\delta_i,(L,t_0)), P_\theta(s_i,\delta_i,(L,t_1)), \ldots, P_\theta(s_i,\delta_i,(L,t_{N_h}))) \in \mathbb{R}^{N_h}$ is the predicted waveform and $p_i \in \mathbb{R}^{N_h}$ is the ground true BP waveform. Then, the four metrics are defined as follows:

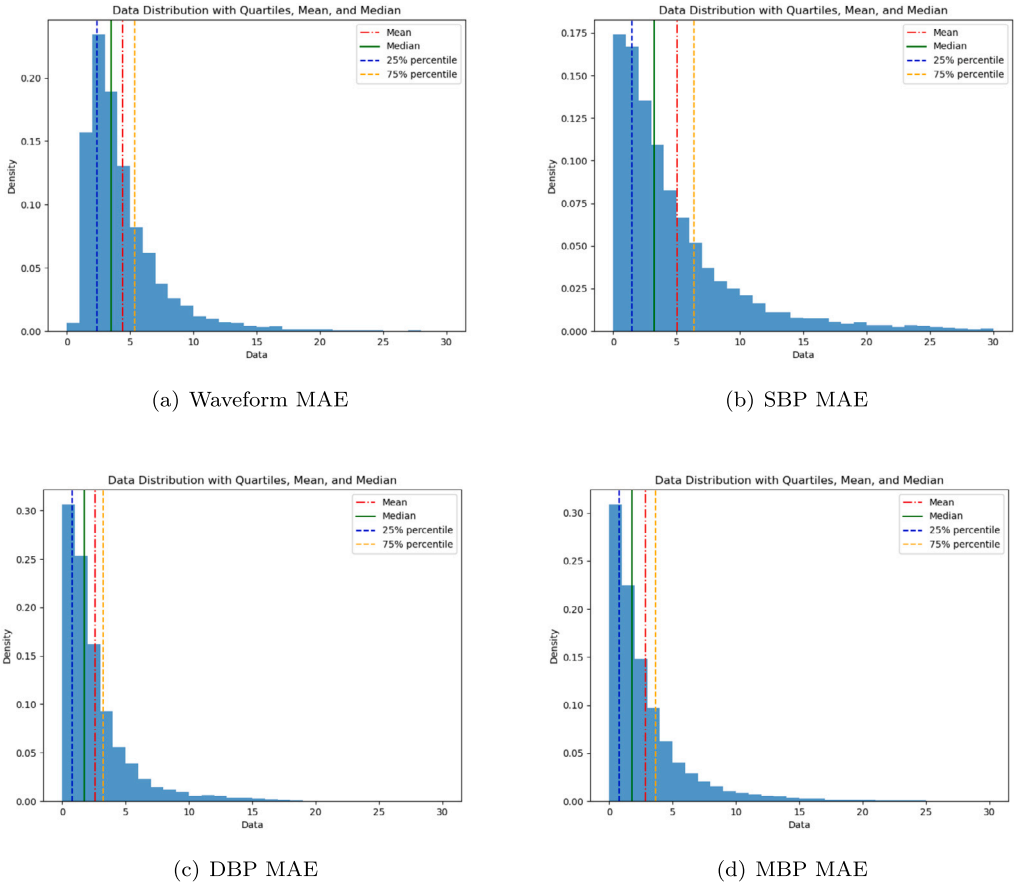
- Waveform MAE: $\frac{1}{N}\frac{1}{N_h}\sum_{i=1}^{N} |P_i - p_i|_1$,
- SBP MAE: $\frac{1}{N}\sum_{i=1}^{N} |\max(P_i) - \max(p_i)|$,
- DBP MAE: $\frac{1}{N}\sum_{i=1}^{N} |\min(P_i) - \min(p_i)|$,
- MBP MAE: $\frac{1}{N}\sum_{i=1}^{N} |\text{mean}(P_i) - \text{mean}(p_i)|$,

where $|\cdot|_1$ denotes the standard vector l_1 norm. In Table 1, we listed the error metrics for both BP-DeepONet and hyper BP-DeepONet. We also compared our results with the IEEE standard for wearable, cuffless blood pressure measuring devices [61]. The suggested grading is also shown in Table 1. We can see the hyper BP-DeepONet achieves better accuracy on the MAE of SBP, DBP, and MBP, while the BP-DeepONet achieves better accuracy on the waveform MAE. According to the IEEE standard for wearable cuffless blood pressure measuring devices, the device or algorithm will be graded as A when the MAE is within 5 mmHg and graded as B when the MAE is within 5 to 6 mmHg. Based on this criterion, the hyper BP-DeepONet is graded as A-level in all metrics, while the BP-DeepONet is graded as A-level except on the SBP measurement. We also visualize the MAE distribution in Fig. 13 and Fig. 14.

Since our methods can predict the blood pressure $P(z,t)$ and blood flow rate $Q(z,t)$ in the whole domain $(z,t) \in [0,L]\times[0,T]$, we also want to evaluate the errors in the interior of this domain. We choose two different metrics to evaluate the error in $(z,t) \in$

Table 1
The prediction error of two proposed methods and their grading according to the IEEE standard.

	Methods	Waveform	SBP	DBP	MBP
MAE (mmHg)	BP-DeepONet	**4.406**	5.057	2.581	2.881
	Hyper BP-DeepONet	4.555(±3.26)	**4.934(±5.80)**	**2.574**	**2.871**
Grading	BP-DeepONet	A	B	A	A
	Hyper BP-DeepONet	A	A	A	A



(a) Waveform MAE (b) SBP MAE

(c) DBP MAE (d) MBP MAE

Fig. 13. Error distribution of the BP-DeepONet on the MIMIC test set. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

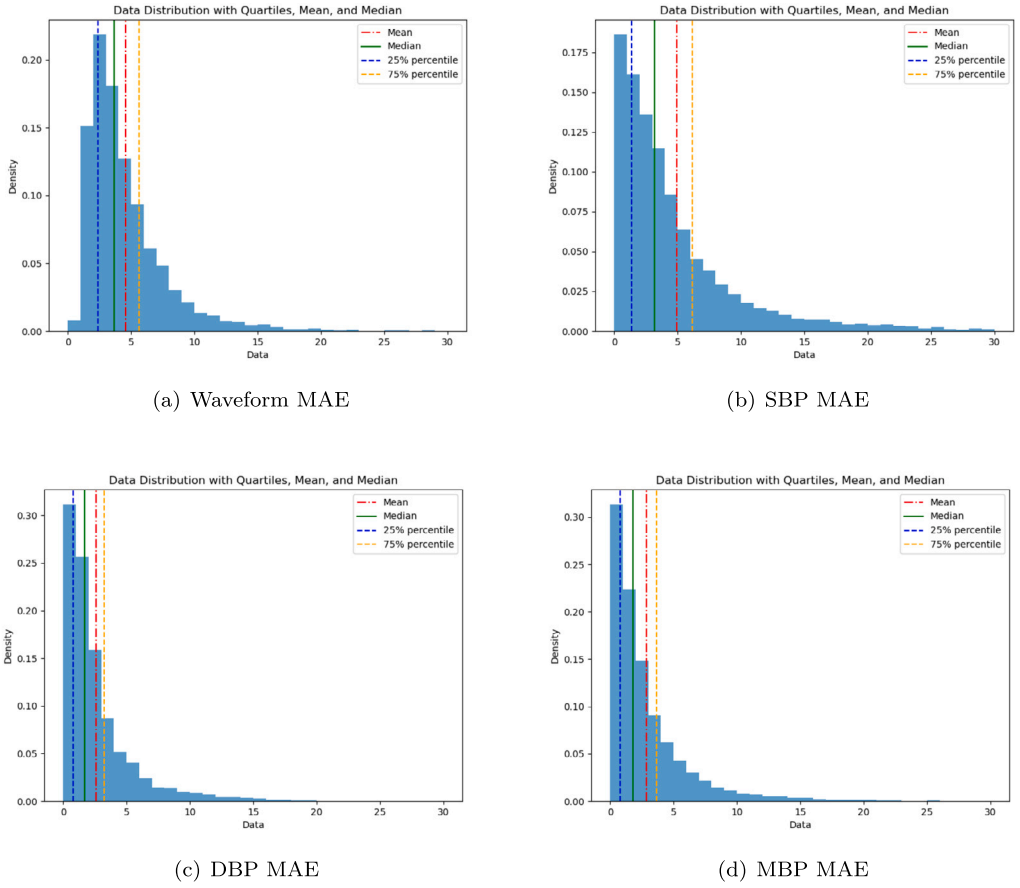
$[0, L] \times [0, T]$. The first metric is the residual errors $\mathcal{R}_{pde}$. The residual error is usually used to estimate the total error when the true solution is unknown. For the BP-DeepONet, the residual error is defined as:

$$RE_1(P_\theta, Q_\theta) = \frac{1}{N}\sum_{i=1}^{N}\left\|\left(\frac{\partial A_\theta}{\partial t} + \frac{\partial Q_\theta}{\partial z}\right)(s_i, \delta_i, (\cdot,\cdot))\right\|_{L^2([0,L]\times[0,T])}$$

$$RE_2(P_\theta, Q_\theta) = \frac{1}{N}\sum_{i=1}^{N}\left\|\left(\frac{\partial Q_\theta}{\partial t} + \frac{\partial}{\partial z}\left(\frac{Q_\theta^2}{A_\theta}\right) + \frac{A_\theta}{\rho}\frac{\partial P_\theta}{\partial z} + K_r\frac{Q_\theta}{A_\theta}\right)(s_i, \delta_i, (\cdot,\cdot))\right\|_{L^2([0,L]\times[0,T])}$$

where $A_\theta = \left(\frac{A_0}{\beta}(P_\theta - P_{ext}) + \sqrt{A_0}\right)^2$. For the hyper BP-DeepONet, the residual error is defined as:

$$RE_1(P_\theta, Q_\theta, \gamma_\theta) = \frac{1}{N}\sum_{i=1}^{N}\left\|\left(\frac{\partial A_{\theta,i}}{\partial t} + \frac{\partial Q_\theta}{\partial z}\right)(s_i, \delta_i, (\cdot,\cdot))\right\|_{L^2([0,L]\times[0,T])}$$



(a) Waveform MAE (b) SBP MAE

(c) DBP MAE (d) MBP MAE

Fig. 14. Error distribution of the hyper BP-DeepONet on the MIMIC test set.

Table 2
The residual error of two proposed methods.

	$RE_1(\sqrt{L})$	$RE_2(\sqrt{dm * L/s})$
BP-DeepONet	1.11×1e-4	2.65×1e-2
Hyper BP-DeepONet	**0.22×1e-4**	**1.14×1e-2**

$$RE_2(P_\theta, Q_\theta, \gamma_\theta) = \frac{1}{N}\sum_{i=1}^{N}\left\|\left(\frac{\partial Q_\theta}{\partial t} + \frac{\partial}{\partial z}\left(\frac{Q_\theta^2}{A_{\theta,i}}\right) + \frac{A_{\theta,i}}{\rho_i}\frac{\partial P_\theta}{\partial z} + K_{r_i}\frac{Q_\theta}{A_{\theta,i}}\right)(s_i, \delta_i, (\cdot,\cdot))\right\|_{L^2([0,L]\times[0,T])}$$

where $A_{\theta,i} = \left(\frac{A_{0,i}}{\beta}(P_\theta - P_{ext,i}) + \sqrt{A_{0,i}}\right)^2$. In [41], the authors show that the error of solutions to the Navier-Stokes by neural networks can be bounded by the residual errors. The residual errors of the two methods are shown in Table 2.

The second metric we used to evaluate the error is the relative error compared to a fine-tuned BP-DeepONet (or hyper BP-DeepONet) on each test sample. Though we do not know the ground true solution, we have shown in Section 5 that a well-trained PINN solution is reasonably accurate compared to the simulated true solution. Therefore, we can fine-tune the network on each test sample, i.e., minimize the PDE residual loss of this particular sample only, and use this fine-tuned network as an approximation to the true solution. During the fine-tuning stage, we fix the parameters in the branch net and only train the parameters in the trunk net so that predictions of hyper-parameters will not change. The error metric is defined as:

$$\sum_{i=1}^{N}\frac{\left\|\left(\hat{P}_{tuned} - P_\theta\right)(s_i, \delta_i, (\cdot,\cdot))\right\|_{L^1([0,L]\times[0,T])}}{\left\|\hat{P}_{tuned}(s_i, \delta_i, (\cdot,\cdot))\right\|_{L^1([0,L]\times[0,T])}}$$

and

Table 3
The relative error of two proposed methods.

	error of P_θ	error of Q_θ
BP-DeepONet	**4.10×1e-2**	**5.35×1e-2**
Hyper BP-DeepONet	4.18×1e-2	1.03×1e-1

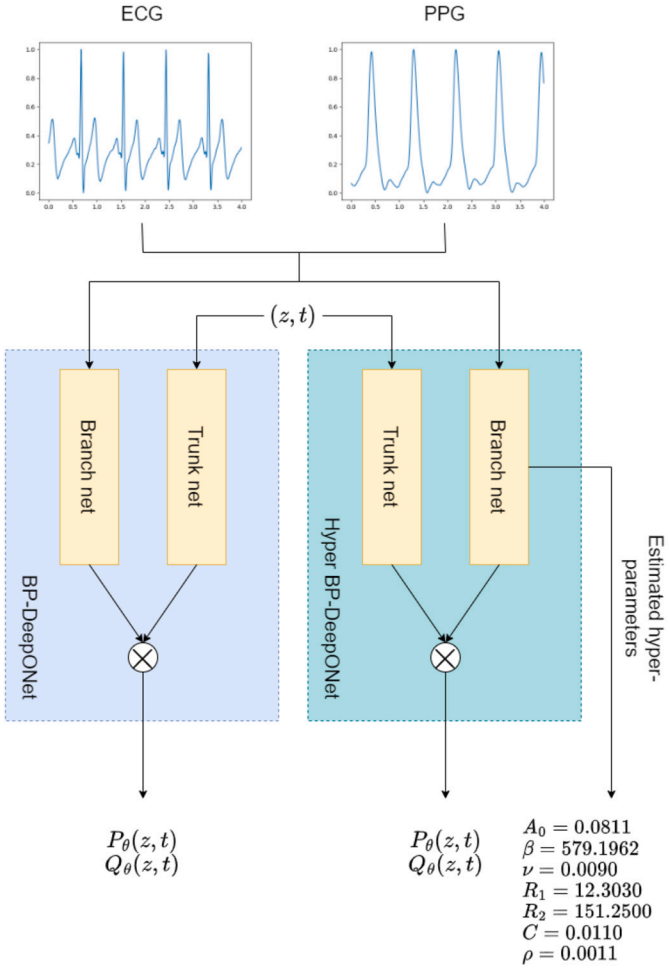
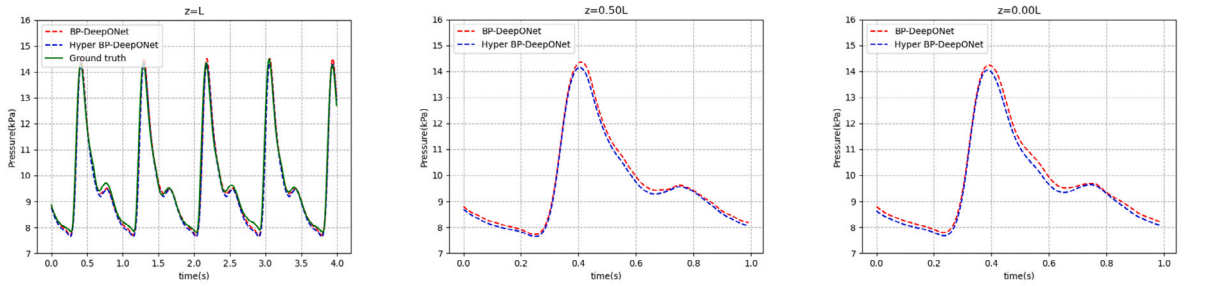


Fig. 15. Visualization of the workflow of BP-DeepONet and hyper BP-DeepONet on one testing sample.

$$\sum_{i=1}^{N} \frac{\left\| \left(\hat{Q}_{tuned} - Q_\theta\right)(s_i, \delta_i, (\cdot,\cdot)) \right\|_{L^1([0,L]\times[0,T])}}{\left\| \hat{Q}_{tuned}(s_i, \delta_i, (\cdot,\cdot)) \right\|_{L^\infty([0,L]\times[0,T])}}$$

where $\hat{P}_{tuned}$ and $\hat{Q}_{tuned}$ denote the solution after fine-tuning. The relative errors are given in Table 3. We can see the BP-DeepONet can predict the PDE solution better than the hyper BP-DeepONet. This result is not surprising because the BP-DeepONet assumes all samples satisfy the same PDE, which greatly simplifies the problem.

We also visualize the workflow of both methods for one test sample in Fig. 15 and the predicted waveforms at different locations in Fig. 16. The predicted hyper-parameters differ from the reference values used for the BP-DeepONet, but it does not deviate too much because of the regularization techniques we used during training. We can observe that the predictions of both the BP-DeepONet and the hyper BP-DeepONet are very close to the ground truth data at the outlet side $z = L$ (Fig. 16 (a)). The pulse pressure, i.e., the difference between the maximum and minimum of the waveforms, also increases when z increases. Because the hyper BP-DeepONet uses different hyper-parameters for each sample, its prediction differs from BP-DeepONet in $z \in (0, 1)$.



(a) Predicted boundary condition (b) pressure waveform at the midpoint (c) pressure waveform at the inlet point

Fig. 16. Predictions of ABP waveforms at different locations z.

7. Conclusions and future works

In this work, we propose two methods based on the physics-informed DeepONet to predict the blood pressure waveform: the BP-DeepONet and the hyper BP-DeepONet. We incorporate the Navier-Stokes equation with time-periodic conditions and Windkessel boundary conditions into the physics-informed training procedure and provide efficient implementation algorithms. The proposed methods are the first ones that can predict waveforms at different locations which may better reflect the cardiovascular status. For example, we can directly predict the brachial blood pressure to indicate the risk of hypertension.

During our numerical experiments, the proposed methods can predict blood pressure waveforms at reasonable accuracy and preserve some physical properties consistent with the clinical observations. Though the training and testing data used in this work are obtained from ICU patients, we still expect our methods can be generalized to some non-ICU patients. When the patients are in a stationary state, our method should still be able to predict the blood flow dynamics reasonably well. Even though the direct measurement of arterial blood pressure is not available, physiological signals like ECG and PPG are still easy to measure accurately. However, it would be difficult to make predictions when patients are in a non-stationary state since the measurements of ECG and PPG are not accurate and the blood flow dynamics may have a significant shift from the training set.

This work only considers very standard DNN models for trunk and branch nets. We may try implementing different DNN architectures in the future, like the transformer, which is very powerful in processing time-series data. Besides, the inputs to the DeepONet are just the pre-processed physiological signals. We may also try including some morphological features in the inputs, effectively improving the prediction accuracy.

We know the arterial system of humans is very complicated. As the first work in this direction, we only consider a simple cylindrical segment of an artery. Still, we may generalize our methods to more complex geometry afterward—for example, a simple arterial tree with bifurcations.

In the future, we will also consider generating a simulated dataset to validate better the accuracy of both BP-DeepONet and hyper BP-DeepONet methods.

CRediT authorship contribution statement

Lingfeng Li: Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **Xue-Cheng Tai:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Raymond Hon-Fu Chan:** Writing – review & editing, Supervision, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Lingfeng Li, Xue-Cheng Tai, and Raymond Chan have patent #CN117257244A pending to Hong Kong Centre for Cerebro Cardiovascular Health Engineering Ltd. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We only use public dataset.

Acknowledgement

This work of Lingfeng Li is supported by the InnoHK project at Hong Kong Centre for Cerebro-cardiovascular Health Engineering (COCHE). This work of Xue-Cheng Tai is partially supported by HKRGC-NSFC Grant N-CityU214/19, HKRGC CRF Grant C1013-21GF and NORCE Kompetanseoppbygging program. This work of Raymond Chan is partially supported by HKRGC GRF grants

CityU1101120, CityU11309922, CRF grant C1013-21GF, HKITF MHKJFS Grant MHP/054/22 and LU BGR 105824. We want to thank Prof Yuan-Ting Zhang and Dr. Alireza Keramat for the helpful discussion during this project.

References

[1] A. Quarteroni, L. Formaggia, Mathematical modelling and numerical simulation of the cardiovascular system, Handb. Numer. Anal. 12 (2004) 3–127.
[2] L. Formaggia, D. Lamponi, A. Quarteroni, One-dimensional models for blood flow in arteries, J. Eng. Math. 47 (2003) 251–276.
[3] D. Elad, D. Katz, E. Kimmel, S. Einav, Numerical schemes for unsteady fluid flow through collapsible tubes, J. Biomed. Eng. 13 (1) (1991) 10–18.
[4] J.-M. Fullana, S. Zaleski, A branched one-dimensional model of vessel networks, J. Fluid Mech. 621 (2009) 183–204.
[5] J. Wan, B. Steele, S.A. Spicer, S. Strohband, G.R. Feijoó, T.J. Hughes, C.A. Taylor, A one-dimensional finite element method for simulation-based medical planning for cardiovascular disease, Comput. Methods Biomech. Biomed. Eng. 5 (3) (2002) 195–206.
[6] S.J. Sherwin, L. Formaggia, J. Peiro, V. Franke, Computational modelling of 1d blood flow with variable mechanical properties and its application to the simulation of wave propagation in the human arterial system, Int. J. Numer. Methods Fluids 43 (6–7) (2003) 673–700.
[7] N. Cavallini, V. Caleffi, V. Coscia, Finite volume and weno scheme in one-dimensional vascular system modelling, Comput. Math. Appl. 56 (9) (2008) 2382–2397.
[8] O. Delestre, P.-Y. Lagrée, A 'well-balanced' finite volume scheme for blood flow simulation, Int. J. Numer. Methods Fluids 72 (2) (2013) 177–205.
[9] K.S. Matthys, J. Alastruey, J. Peiró, A.W. Khir, P. Segers, P.R. Verdonck, K.H. Parker, S.J. Sherwin, Pulse wave propagation in a model human arterial network: assessment of 1-d numerical simulations against in vitro measurements, J. Biomech. 40 (15) (2007) 3476–3486.
[10] E. Marchandise, M. Willemet, V. Lacroix, A numerical hemodynamic tool for predictive vascular surgery, Med. Eng. Phys. 31 (1) (2009) 131–144.
[11] J. Mynard, P. Nithiarasu, A 1d arterial blood flow model incorporating ventricular pressure, aortic valve and regional coronary flow using the locally conservative Galerkin (lcg) method, Commun. Numer. Methods Eng. 24 (5) (2008) 367–417.
[12] Machine learning in cardiovascular flows modeling: predicting arterial blood pressure from non-invasive 4d flow mri data using physics-informed neural networks, Comput. Methods Appl. Mech. Eng. 358 (2020) 112623.
[13] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[14] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, arXiv preprint, arXiv:2010.08895, 2020.
[15] B. Matthieu, N.-G. Karine, C. Vincent, V. Alain, C.J. François, R. Philippe, S. François, Cardiac output measurement in patients undergoing liver transplantation: pulmonary artery catheter versus uncalibrated arterial pressure waveform analysis, Anesth. Analg. 106 (5) (2008) 1480–1486.
[16] R.H. Thiele, M.E. Durieux, Arterial waveform analysis for the anesthesiologist: past, present, and future concepts, Anesth. Analg. 113 (4) (2011) 766–776.
[17] Y. Li, H. Gu, H. Fok, J. Alastruey, P. Chowienczyk, Forward and backward pressure waveform morphology in hypertension, Hypertension 69 (2) (2017) 375–381.
[18] D.A. Hullender, O.R. Brown, Simulations of blood pressure and identification of atrial fibrillation and arterial stiffness using an extended Kalman filter with oscillometric pulsation measurements, Comput. Methods Programs Biomed. 198 (2021) 105768.
[19] P. Su, X.-R. Ding, Y.-T. Zhang, J. Liu, F. Miao, N. Zhao, Long-term blood pressure prediction with deep recurrent neural networks, in: 2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), IEEE, 2018, pp. 323–328.
[20] H. Eom, D. Lee, S. Han, Y.S. Hariyani, Y. Lim, I. Sohn, K. Park, C. Park, End-to-end deep learning architecture for continuous blood pressure estimation using attention mechanism, Sensors 20 (8) (2020) 2338.
[21] N. Ibtehaz, S. Mahmud, M.E. Chowdhury, A. Khandakar, M. Salman Khan, M.A. Ayari, A.M. Tahir, M.S. Rahman, Ppg2abp: translating photoplethysmogram (ppg) signals to arterial blood pressure (abp) waveforms, Bioengineering 9 (11) (2022) 692.
[22] C. Ma, P. Zhang, F. Song, Y. Sun, G. Fan, T. Zhang, Y. Feng, G. Zhang, Kd-informer: cuff-less continuous blood pressure waveform estimation approach based on single photoplethysmography, IEEE J. Biomed. Health Inform. (2022).
[23] K. Lakhal, V. Robert-Edan, Invasive monitoring of blood pressure: a radiant future for brachial artery as an alternative to radial artery catheterisation?, J. Thorac. Dis. 9 (12) (2017) 4812.
[24] S.-P. E, Continuous non-invasive arterial pressure shows high accuracy in comparison to invasive intra-arterial blood pressure measurement, https://api.semanticscholar.org/CorpusID:30046018, 2008.
[25] M.K. Armstrong, M.G. Schultz, D.S. Picone, J.A. Black, N. Dwyer, P. Roberts-Thomson, J.E. Sharman, Brachial and radial systolic blood pressure are not the same: evidence to support the Popeye phenomenon, Hypertension 73 (5) (2019) 1036–1041.
[26] J. Kyriazis, J. Glotsos, L. Bilirakis, N. Smirnioudis, The (dp/dt) max derived from arterial pulse waveforms: prospective applications in the haemodialysis setting, Nephrol. Dial. Transplant. 16 (5) (2001) 1087–1088.
[27] J. Sharman, A. Qasem, L. Hanekom, D. Gill, R. Lim, T. Marwick, Radial pressure waveform dp/dt max is a poor indicator of left ventricular systolic function, Eur. J. Clin. Investig. 37 (4) (2007) 276–281.
[28] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, G.E. Karniadakis, A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data, Comput. Methods Appl. Mech. Eng. 393 (2022) 114778.
[29] J. Alastruey, A.W. Khir, K.S. Matthys, P. Segers, S.J. Sherwin, P.R. Verdonck, K.H. Parker, J. Peiró, Pulse wave propagation in a model human arterial network: assessment of 1-d visco-elastic simulations against in vitro measurements, J. Biomech. 44 (12) (2011) 2250–2258.
[30] M. Saito, Y. Ikenaga, M. Matsukawa, Y. Watanabe, T. Asada, P.-Y. Lagree, One-dimensional model for propagation of a pressure wave in a model of the human arterial network: comparison of theoretical and experimental results, J. Biomech. Eng. 133 (12) (2011).
[31] M.S. Olufsen, C.S. Peskin, W.Y. Kim, E.M. Pedersen, A. Nadim, J. Larsen, Numerical simulation and experimental validation of blood flow in arteries with structured-tree outflow conditions, Ann. Biomed. Eng. 28 (2000) 1281–1299.
[32] B.N. Steele, J. Wan, J.P. Ku, T.J. Hughes, C.A. Taylor, In vivo validation of a one-dimensional finite-element method for predicting blood flow in cardiovascular bypass grafts, IEEE Trans. Biomed. Eng. 50 (6) (2003) 649–656.
[33] X. Wang, 1d modeling of blood flow in networks: Numerical computing and applications, Ph.D. thesis, Paris 6, 2014.
[34] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[35] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, Deepxde: a deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.
[36] B. Yu, et al., The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[37] L. Lyu, Z. Zhang, M. Chen, J. Chen, MIM: a deep mixed residual method for solving high-order partial differential equations, J. Comput. Phys. 452 (2022) 110930.
[38] J. Yang, Q. Zhu, A local deep learning method for solving high order partial differential equations, Numer. Math., Theory Methods Appl. (2021).
[39] Y. Lu, J. Lu, M. Wang, A priori generalization analysis of the deep Ritz method for solving high dimensional elliptic partial differential equations, in: Conference on Learning Theory, PMLR, 2021, pp. 3196–3241.
[40] J. Müller, M. Zeinhofer, Error estimates for the variational training of neural networks with boundary penalty, arXiv preprint, arXiv:2103.01007, 2021.
[41] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes, IMA J. Numer. Anal. 42 (2) (2022) 981–1022.
[42] L. Li, X.-C. Tai, J. Yang, Generalization error analysis of neural networks with gradient based regularization, Commun. Comput. Phys. 32 (4) (2022) 1007–1038.

[43] L. Li, X.-c. Tai, J. Yang, Q. Zhu, Priori error estimate of deep mixed residual method for elliptic pdes, arXiv preprint, arXiv:2206.07474, 2022.
[44] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[45] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed deeponets, Sci. Adv. 7 (40) (2021) eabi8605.
[46] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar, Physics-informed neural operator for learning partial differential equations, arXiv preprint, arXiv:2111.03794, 2021.
[47] S. Goswami, M. Yin, Y. Yu, G.E. Karniadakis, A physics-informed variational deeponet for predicting crack path in quasi-brittle materials, Comput. Methods Appl. Mech. Eng. 391 (2022) 114587.
[48] N. Westerhof, J.-W. Lankhaar, B.E. Westerhof, The arterial windkessel, Med. Biol. Eng. Comput. 47 (2) (2009) 131–141.
[49] J. Alastruey, K.H. Parker, S.J. Sherwin, et al., Arterial pulse wave haemodynamics, in: 11th International Conference on Pressure Surges, vol. 30, Virtual PiE Led t/a BHR Group Lisbon, Portugal, 2012, pp. 401–443.
[50] N. Xiao, J. Alastruey, C. Alberto Figueroa, A systematic comparison between 1-d and 3-d hemodynamics in compliant arterial models, Int. J. Numer. Methods Biomed. Eng. 30 (2) (2014) 204–231.
[51] A. Paviglianiti, V. Randazzo, S. Villata, G. Cirrincione, E. Pasero, A comparison of deep learning techniques for arterial blood pressure prediction, Cogn. Comput. 14 (5) (2022) 1689–1710.
[52] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S.G. Johnson, Physics-informed neural networks with hard constraints for inverse design, SIAM J. Sci. Comput. 43 (6) (2021) B1105–B1132.
[53] A.P. Avolio, L.M. Van Bortel, P. Boutouyrie, J.R. Cockcroft, C.M. McEniery, A.D. Protogerou, M.J. Roman, M.E. Safar, P. Segers, H. Smulyan, Role of pulse pressure amplification in arterial hypertension: experts' opinion and review of the data, Hypertension 54 (2) (2009) 375–383.
[54] Y. Wang, F. Liu, D.E. Schiavazzi, Variational inference with nofas: normalizing flow with adaptive surrogate for computationally expensive models, J. Comput. Phys. 467 (2022) 111454.
[55] G.G. Tong, C.A.S. Long, D.E. Schiavazzi, Invaert networks: a data-driven framework for model synthesis and identifiability analysis, Comput. Methods Appl. Mech. Eng. 423 (2024) 116846.
[56] Z. Gao, L. Yan, T. Zhou, Failure-informed adaptive sampling for pinns, SIAM J. Sci. Comput. 45 (4) (2023) A1971–A1994.
[57] A.E. Johnson, T.J. Pollard, L. Shen, L.-w.H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, R.G. Mark, MIMIC-III, a freely accessible critical care database, Sci. Data 3 (1) (2016) 1–9.
[58] M. Kachuee, M. Kiani, M. Hoda, M. Shabany, Cuff-less blood pressure estimation, UCI machine learning repository, https://doi.org/10.24432/C5B602, 2015.
[59] J. Hong, J. Gao, Q. Liu, Y. Zhang, Y. Zheng, Deep learning model with individualized fine-tuning for dynamic and beat-to-beat blood pressure estimation, in: 2021 IEEE 17th International Conference on Wearable and Implantable Body Sensor Networks (BSN), IEEE, 2021, pp. 1–4.
[60] P.H. Charlton, J. Mariscal Harana, S. Vennin, Y. Li, P. Chowienczyk, J. Alastruey, Modeling arterial pulse waves in healthy aging: a database for in silico evaluation of hemodynamics and pulse wave indexes, Am. J. Physiol., Heart Circ. Physiol. 317 (5) (2019) H1062–H1085.
[61] I.S. Association, et al., IEEE standard for wearable cuffless blood pressure measuring devices, IEEE Std 1708–2014, 2014.