Acta Mechanica Sinica (2021) 37(1):35–46
https://doi.org/10.1007/s10409-020-01028-0

RESEARCH PAPER

Machine learning models for the secondary Bjerknes force between two insonated bubbles

Haiyan Chen[1] · Yue Zeng[1] · Yi Li[2]

Received: 31 August 2020 / Revised: 10 October 2020 / Accepted: 25 October 2020 / Published online: 8 January 2021

Abstract
The secondary Bjerknes force plays a significant role in the evolution of bubble clusters. However, due to the complex dependence of the force on multiple parameters, it is highly non-trivial to include its effects in the simulations of bubble clusters. In this paper, machine learning is used to develop a data-driven model for the secondary Bjerknes force between two insonated bubbles as a function of the equilibrium radii of the bubbles, the distance between the bubbles, the amplitude and the center frequency of the ultrasound wave. The sign of the force may change with the phase difference between the oscillating bubbles. Meanwhile, the magnitude of the force varies over several orders of magnitude, which poses a serious challenge for the usual machine learning models. To overcome this difficulty, the magnitudes and the signs of the force are separated and modelled separately. A nonlinear regression is obtained with a feed-forward network model for the *logarithm* of the magnitude, whereas the sign is modelled by a support-vector machine model. The principle, the practical aspects related to the training and validation of the machine models are introduced. The predictions from the models are checked against the values computed from the Keller–Miksis equations. The results show that the models are extremely efficient while providing accurate estimate of the force. The models make it computationally feasible for the future simulations of the bubble clusters to include the effects of the secondary Bjerknes force.

Keywords Bubble clusters · Secondary Bjerknes force · Machine learning · Neural networks · Support-vector machine · Numerical simulations

1 Introduction

The secondary Bjerknes force [1] is an important component in the interaction between two bubbles oscillating in an acoustically driven fluid. The force is induced by the pressure perturbation radiated from the oscillations bubbles. It is thought to be important in the formation and evolution of bubble clusters and has been the focus of considerable research in the past decades [2–11], which explores the effects of nonlinear interaction, multiple scattering, and the coupling with shape oscillation and translation, as well as the experimental measurement of the force. More recently, Doinikov and Bouakaz [12] develop a nonlinear theoretical model that provides a unified description of the coupled radial oscillation and translation of two bubbles. The model is generalised to investigate the acoustic micro-streaming patterns in the vicinity of two interacting bubbles [13]. The asymmetricity of the force is discussed in Pandey [14] taking into account higher order nonlinear coupling between the bubbles, which further highlights the complexity of the force.

Recent experimental evidences [5,10,15–17] further draw attention to the importance of the secondary Bjerknes force in the dynamics of bubble pairs and micro-bubble clusters. The collective behaviors of up to 100 oscillating bubbles are modelled in Haghi et al. [18] using the coupled Keller–Miksis equations [19]. It is found that the interactions between the bubbles can be both constructive and destructive, and the bifurcation sequences of a large system with many bubbles can be very different from a small one. The research again demonstrates the importance of the interactions between the bubbles which are manifested as the secondary Bjerknes force. The force has been used to manipulate bubbles, e.g.,

Executive Editor: Chao Sun.

✉ Yi Li
yili@sheffield.ac.uk

[1] School of Material and Energy, Guangdong University of Technology, Guangzhou 510006, China

[2] School of Mathematics and Statistics, University of Sheffield, Sheffield S3 7RH, UK

as a means to control micro-devices, which potentially have important applications [20–22]. Given that bubble clusters are often observed in biomedical, metallurgical, food processing and other applications (see, e.g., Bermudez-Aguirre et al. [24], Brujan [23], Eskin and Eskin [26], Roberts [25]), the modelling of the secondary Bjerknes force and hence bubble clusters is a question of significant interests.

Few simulations of bubble clusters so far have employed sophisticated models for the secondary Bjerknes force. Numerical simulations conducted in Lazarus et al. [16], with a simple model for the secondary Bjerknes force, qualitatively reproduce the experimental observations on the clustering of bubble clouds. Similarly, simplified models are used in the simulations in Refs. [27–29]. In particular, Mettin et al. [28] qualitatively reproduce the process by which bubbles agglomerate to form a pattern that, as described by the authors, mimics the Lichtenberg figures sometimes observed in sudden electrical discharges such as lightnings (see Fig. 1 in Mettin et al. [28] for an illustration). These simulations follow the movements of individual bubbles, thus are based on a Lagrangian approach. Recently a hybrid Lagrangian-Eulerian method is proposed in Ref. [30] where bubble oscillation is computed, although the secondary Bjerknes force is not explicitly included.

The past research has yielded considerable physical insights on the secondary Bjerknes force. Unfortunately, due to the complexity of bubble clusters, the insights have yet to be translated into accurate and computationally efficient models. We observe, however, that the complexity of the problem makes it an excellent example for which a data-driven approach can be fruitful. Data-driven methods, especially machine learning, have made tremendous progresses in recent years, as exemplified and popularized by the success of AlphaGo [31]. The methods have been successfully applied to many physical and applied sciences. There is, however, not yet any report of such applications in bubble simulations. The objective of this paper is to use machine learning to build a novel model for the secondary Bjerknes force that is more comprehensive than those previously reported, and more generally, introduce this useful method into the investigation and modelling of bubble oscillations.

In order to build machine learning models, a sufficiently large dataset for the secondary Bjerknes force is needed. Experimental measurements of the force have recently been reported in, e.g., Jiao et al. [10], Yoshida et al. [5], Ma et al. [17]. However, to the best of our knowledge, systematic data covering a wide range of parameters are not yet available. Therefore, in this investigation we use numerical simulations based on the Keller–Miksis model to generate the dataset.

The paper is organized as follows. The dynamical equations for the bubbles are reviewed in Sect. 2, where the dependence of the secondary Bjerknes force on relevant parameters are highlighted. The dataset for the force is described in Sect. 3. Section 4 introduces the machine learning models to be used to build the model for the force. The practical aspects of the training and testing of the models are also presented. Additional checks are performed in Sect. 5 where the efficiency of the models is also assessed. The conclusions are summarized in Sect. 6.

2 Governing equations

Let D be the distance between the two bubbles. The radius of bubble i ($i = 1, 2$) is denoted by $R_i(t)$ and its equilibrium radius is R_{Ei}. The bubbles are driven by a uniform pressure field oscillating harmonically in time:

$$p_I(t) = p_0 - p_a \sin(\omega t), \tag{1}$$

where p_0 is the ambient pressure, p_a is the amplitude of the ultrasonic pressure, and $\omega \equiv 2\pi f$ and f are the angular and linear frequencies, respectively. By using a pressure uniform in space, it has been assumed that D is small compared with the wave length of the pressure wave or the bubbles are on the same phase plane of a planar pressure wave. By using simple harmonic pressure oscillation, we have assumed the acoustic pressure is low enough that its propagation can be described by the linear acoustic equation. It is assumed that the fluid has density ρ, speed of sound c, surface tension σ and kinematic viscosity ν.

The radii of the bubbles can be described by the Keller–Miksis model [1,19] with additional pressure coupling terms between the bubbles as introduced in Mettin et al. [6]. Ignoring the time-delay effect, the coupling pressure between bubbles i ($i = 1, 2$) and $j \equiv 3 - i$, denoted as p_{ij}, is given by [6]

$$p_{ij}(t) = \frac{\rho}{D}\frac{\mathrm{d}R_j^2\dot{R}_j}{\mathrm{d}t}, \tag{2}$$

which is valid when the radii R_i and R_j are much smaller than D. With p_{ij} included, the equation for $R_i(t)$ becomes [6]

$$\begin{aligned} &2\rho(1 - c^{-1}\dot{R}_i)R_i\ddot{R}_i + \rho(3 - c^{-1}\dot{R}_i)\dot{R}_i^2 \\ &= 2(1 + c^{-1}\dot{R}_i)(p_{wi} - p_I) + 2c^{-1}R_i(\dot{p}_{wi} - \dot{p}_I) \\ &\quad - 2\rho D^{-1}(2R_j\dot{R}_j^2 + R_j^2\ddot{R}_j), \end{aligned} \tag{3}$$

where

$$p_{wi} = \left(p_0 + \frac{2\sigma}{R_{Ei}}\right)\left(\frac{R_{Ei}}{R_i}\right)^{3k} - \frac{2\sigma}{R_i} - \frac{4\rho\nu\dot{R}_i}{R_i} \tag{4}$$

is the pressure on the outer wall of bubble i and k is the polytropic exponent for the gas inside the bubble.

As this model assumes $D \gg R_i$ and R_j, the same limitation applies to the results to be presented below. It implies, for example, that the machine learning models obtained below should only be applied to the simulations of sparse bubble clusters. To obtain a model for the secondary Bjerknes force that is applicable in densely packed clusters, one might need to use, e.g., the model in Doinikov and Bouakaz [12], which potentially provides a more accurate description when D is comparable with R_i or R_j. Obviously the machine learning approach to be presented below can be used in conjunction with these models as well.

The secondary Bjerknes force is defined as the time-averaged pressure exerted on bubble i due to the oscillations of bubble j [6]. Let f_{ij} be the notation for this force, simple calculation shows that, for small bubbles, f_{ij} can be written as

$$f_{ij} = -\frac{\rho}{D^2}\left\langle V_i \frac{\mathrm{d}R_j^2 \dot{R}_j}{\mathrm{d}t}\right\rangle = \frac{\rho\langle \dot{V}_i \dot{V}_j\rangle}{4\pi D^2}, \tag{5}$$

where V_i is the volume of bubble i. The angle brackets represent time averaging. In the above expression we follow the tradition where f_{ij} is positive when it is attractive. In a bubble cluster, f_{ij} is expected to depend not only on bubbles i and j but also the other bubbles. Nevertheless, when the force was considered in the few bubble cluster simulations reported so far [16,28,29], f_{ij} had all been calculated from 2-bubble systems, where the contributions from other bubbles were neglected. Empirical fitting of f_{ij} as a function of D was used. The dependence of f_{ij} on other parameters have not been considered in these simulations.

For a 2-bubble system, the only secondary Bjerknes force is f_{12}. f_{12} depends on many parameters of the system, including R_{Ei}, D, p_a, ω, ν, ρ, c, σ, and k. In the present investigation, we choose water as the medium, hence fixing ν at 8.9×10^{-7} m^2/s, ρ at 997 kg/m^3, c at 1497 m/s and σ at 0.0721 N/m. An adiabatic process is assumed so that k is fixed at 1.4, whereas p_0 is assumed to be the atmospheric pressure $p_{\mathrm{atm}} = 1.013 \times 10^5$ Pa. The objective of the investigation is to model the dependence of f_{12} on the five parameters: D, p_a, f, R_{E1} and R_{E2}.

3 Data for f_{12}

The machine learning method is used to discover the complicated dependence of f_{12} on the system parameters. The method is data-driven and is based on a large data set for f_{12} obtained over a range of values for the five parameters. The distance D ranges from 100 to 1000 μm with an increment of 100 μm. The pressure amplitude p_a ranges from 40 to 150 kPa with an increment of 10 kPa. This range covers both near harmonic and strongly nonlinear aharmonic oscillations. The forcing frequency f ranges from 20 to 40 kHz with an increment of 10 kHz. Both R_{E1} and R_{E2} start at 1 μm and end at 10 μm with an increment of 2 μm.

The ranges for the parameters have been chosen to cover the conditions used in the numerical simulations reported in Mettin et al. [28]. The pressure amplitudes are in the low to medium range according to the classification in Mettin [27], in which bubble clusters of different natures are categorized. These pressure amplitudes and frequency values are also within the ranges used in, e.g., food processing [32] and metallurgy [25], although typical medical applications have higher pressure amplitudes as well as higher frequencies.

When the other parameters are held fixed, the bubble pairs with radii (R_{E1}, R_{E2}) and (R_{E2}, R_{E1}) would have the same f_{12}. Therefore the parameter combinations with $R_{E1} < R_{E2}$ are removed from the data set, which leaves in total 5400 combinations of parameter values. Equation (3) is numerically integrated for each combination to obtain the corresponding f_{12}. The ode45 solver in MATLAB is used. In each run, the simulation is run for ten periods of the forcing pressure to allow the oscillation becomes stationary. The data from the last two periods are used to calculate the force f_{12} according to Eq. (5). The data for f_{12} obtained this way, and the corresponding parameters, form the dataset for the development of the machine learning models. In the terminology of machine learning, a set of values for the five parameters $(D, R_{E1}, R_{E2}, p_a, f)$ is called a *predictor* while the corresponding f_{12} is the *response*.

As an illustration, Fig. 1 plots the radii of the two bubbles which oscillate in the nonlinear regime for the given set of parameters. Figure 2 plots the force as a function of R_{E2} for $R_{E1} = 9$ μm and several D's. The force displays previously known behaviours: it increases when R_{E2} is increased or D is reduced.

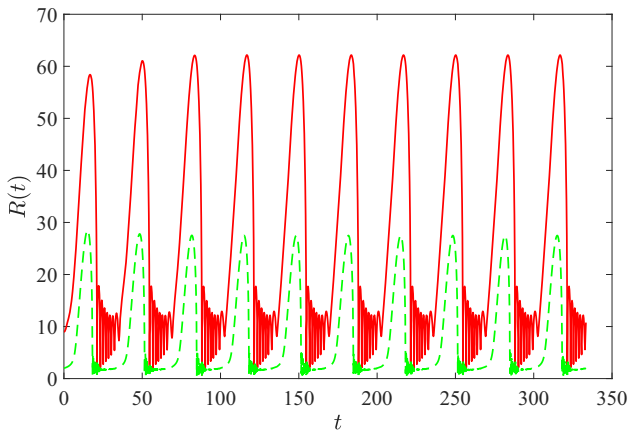

Fig. 1 Radii $R(t)$(μm) for the two bubbles as functions of time t(μs), with $R_{E1} = 9$ μm, $R_{E2} = 2$ μm, $p_a = 150$ kPa, $f = 30$ kHz, $D = 200$ μm

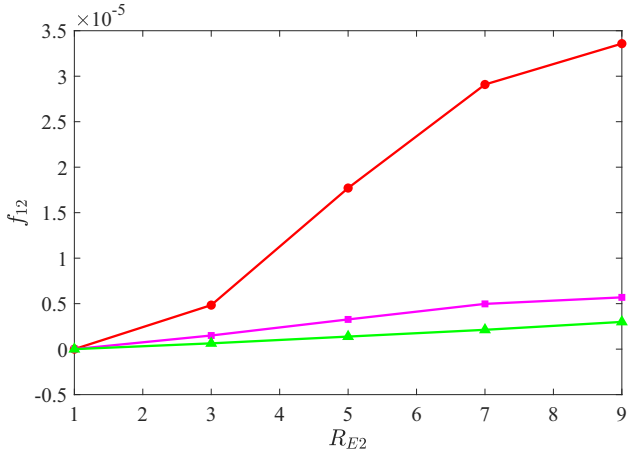

Fig. 2 Force f_{12}(N) as a function of R_{E2}(μm) for $R_{E1} = 9$ μm, $p_a = 150$ kPa, and $f = 30$ kHz. Top: $D = 100$ μm, middle: $D = 300$ μm, bottom: $D = 500$ μm

4 Machine learning models

The secondary Bjerknes force f_{12} depends sensitively on the flow parameters. As a result, the magnitude for f_{12} varies over many orders of magnitude, which is a significant difficulty for the development of the machine learning (ML) models.

To overcome the difficulty, the data set for f_{12} is split into two. The first one contains $\log|f_{12}|$, whereas the second one contains the sign of f_{12} (sgn f_{12}). Two machine learning models are built for the two sets separately, and the prediction for f_{12} is reconstructed from the two models. The first model, given the nature of the data, is a regression model, which is implemented with a feed-forward neural network (FFNN). The data in the second data set are binary (sgn f_{12} is either 1 or -1). A classification model, the support-vector machine (SVM), is thus used. If the predictions from the first and the second models are y_1 and y_2, respectively, the prediction for f_{12} is then given by $y_2 10^{y_1}$.

Working with $\log|f_{12}|$ proves to be crucial. Taking the logarithm reduces the range of the data, and as a result, an FFNN can be found to model $|f_{12}|$ (after exponentiation) and hence f_{12} with good accuracy. Without separating the magnitude and the sign and taking the logarithm of the magnitude, we failed to find a satisfactory ML model for f_{12}. The FFNN and the SVM models are now explained.

4.1 Feed-forward neural network for $\log|f_{12}|$

An FFNN [33] typically includes an input layer, an output layer, and several hidden layers of neurons. The neurons receive inputs from those in previous layers and send outputs to those in later layers. Each neuron is defined by an activation function (also known as transfer function), which processes the inputs using suitable weights and biases associated with the neuron. Figure 3 provides a schematic illustration of the structure of an FFNN. The several components in a large blue rectangle box form a neuron. Only one neuron is shown explicitly in each layer in Fig. 3 but in reality there could be many. In an FFNN, the number of hidden layers L, the number of neurons in each hidden layer N_n ($n = 1, 2, ..., L$), and the activation function $\phi(\cdot)$ are chosen *a priori* and, in practice, mostly empirically. Let $a_j^{(n)}$ be the output of the jth ($j = 1, 2, ..., N_n$) neuron in the nth layer ($n = 1, 2, ..., L$). Then the computation in this neuron is written as

$$a_j^{(n)} = \phi\left(\mathbf{w}_j^{(n)\mathrm{T}}\mathbf{a}^{(n-1)} + b_j^{(n)}\right), \tag{6}$$

where $\mathbf{a}^{(n)}$ is the vector $(a_1^{(n)}, a_2^{(n)}, ..., a_{N_n}^{(n)})^{\mathrm{T}}$, $\mathbf{w}_j^{(n)}$ and $b_j^{(n)}$ are, respectively, the weight and bias associated with the neuron. That is, all the outputs from the $(n-1)$th layer are combined and fed into all the neurons individually in the nth layer. Equation (6) is applicable to the first hidden layer if we let $\mathbf{a}^{(0)}$ be the vector of the input parameters $\mathbf{x} \equiv (D, R_{E1}, R_{E2}, p_a, f)^{\mathrm{T}}$. By repeatedly applying Eq. (6), an FFNN generates an output $\hat{y}$ from the input $\mathbf{x}$.

The weights $\mathbf{w}_j^{(n)}$ and biases $b_j^{(n)}$ are chosen in such a way that the outputs of the FFNN provide the best model for the data in a so-called "training" dataset. The process to find the optimal weights and biases is called training. The training dataset contains a set of input parameters $\mathbf{x}^{(i)}$ and corresponding true outputs $y^{(i)}$ ($i = 1, 2, ..., M$). Let $\hat{y}^{(i)}$ denote the model output for $\mathbf{x}^{(i)}$ predicted by the FFNN. Mathematically, the goal of the training process is to find the optimal $\mathbf{w}_j^{(n)}$ and $b_j^{(n)}$ to minimise the mean squared error (MSE)

$$\frac{1}{2}\sum_{i=1}^{M}\left[y^{(i)} - \hat{y}^{(i)}\right]^2. \tag{7}$$

The optimal solution is usually obtained with a gradient based algorithm where the gradient is found through a process called back-propagation. For more details on FFNNs and machine learning in general, see, e.g., Hagan et al. [33], Goodfellow et al. [34], Hastie et al. [35].

4.1.1 Architecture and the hyperparameters

MATLAB is used to define, train, validate and test the network [36]. The numbers of layers and neurons and the activation functions are the hyperparameters that should be decided at the outset.

For the activation functions, the default setting is adopted, where the hyperbolic tangent sigmoid function $\phi(z) = \tanh(z)$ is used for the neurons in the hidden layers and the linear function $\phi(z) = z$ is used in the output layer. Even though in the machine learning community rectified linear

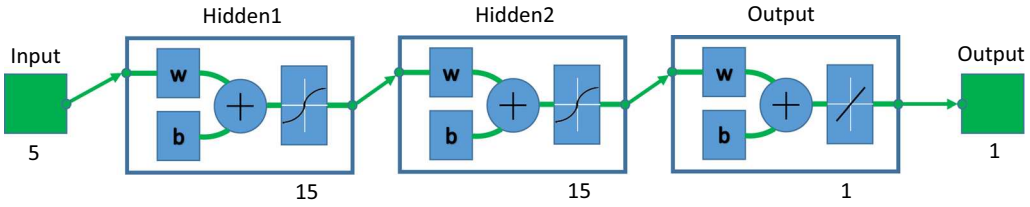

Fig. 3 Architecture of the FFNN with two hidden layers having 15 neurons on each

units (ReLUs) are now the recommended choice for the activation functions in the hidden layers for large scale problems, our tests using the ReLUs for the hidden layers do not show appreciable differences.

As for the numbers of the layers and neurons, it is known that perfect regression can be obtained for any dataset if there is no limit to the number of available neurons. However, the training may become too expensive and the model too inefficient if too many neurons are used. Therefore it is desirable to use as few neurons as possible. Empirical evidences show that, in some cases, the model performance can be improved by using more hidden layers [34]. However, there is not yet theoretical justification or guidelines for the optimal choices. As a result, we have adopted the following trial-and-error strategy to decide these two hyperparameters, which is explained briefly here while related numerical evidence is presented later. We start with an FFNN with only one hidden layer, and train it with increasing number of neurons, until satisfactory performance is obtained. After a number of tests, it is found that consistently good results can be obtained with 30 neurons. The total number of neurons is thus fixed at 30, and networks with different numbers of hidden layers are tested to explore how the performance can be further improved. As demonstrated below with numerical results, the best performance is found with two hidden layers and 15 neurons on each. This architecture is thus chosen, which is illustrated in Fig. 3. More details are given in Sect. 4.1.3.

4.1.2 Training of the FFNN

With the architecture chosen, the weights and the biases in the neurons are then initialized randomly, but they have to be adjusted to improve the performance of the model. This process is called training, in which the dataset for $\log|f_{12}|$ mentioned in Sect. 3 is used. For each predictor (i.e., a parameter combination), the FFNN is used to make a prediction for $\log|f_{12}|$, which is the response in this model. The prediction is compared with the true value obtained as explained in Sect. 3. The Levenberg-Marquardt algorithm [33] is used to optimize the weights and biases iteratively. In the machine learning terminology, an iteration which scans through all training data is called an epoch.

The training is stopped when a certain stopping condition is satisfied. One of these conditions is that the magnitude of the gradient of the MSE should be sufficiently small. However, in our tests, the training is always terminated due to detection of overfitting. Overfitting is a phenomenon where the model performs well in training, but makes poor predictions for data outside of the training dataset. It is a common problem that should be avoided when training a neural network. In this investigation, cross-validation is used to tackle this problem [35]. Specifically, 70% of the dataset for $\log|f_{12}|$ is randomly chosen to form the training set, while 15% is used for validation, and the rest for testing. At each epoch, the current FFNN (with not-yet-optimal weights and biases) is used to make predictions on the validation data set. The MSE of the predictions over the validation set is monitored. If the MSE increases for $N_o = 6$ consecutive epochs (while the MSE for the training set is still decreasing), then overfitting is deemed to have happened and the training is stopped. The value 6 is empirical. If a different N_o is used, one might need to adjust other parameters to obtain a model with similar performance.

4.1.3 Numerical results

The architectures of the FFNN models tested in this investigation are summarized in Table 1. For each architecture, the training is run 32 times with random initialization. 32 models are thus produced, which are slightly different from each other due to random initialization, even though they have

Table 1 Number of neurons (N_n) in each hidden layer for each network architecture

Architecture	N_n in each layer
1	30
2	20, 10
3	10, 20
4	15, 15
5	5, 10, 15
6	10, 10, 10
7	10, 10, 5, 5
8	10, 5, 5, 5, 5

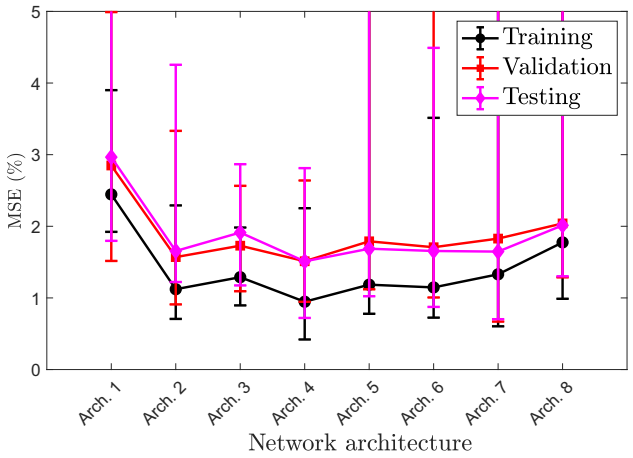

Fig. 4 Median and range of the optimal MSE obtained with different FFNN models in 32 runs

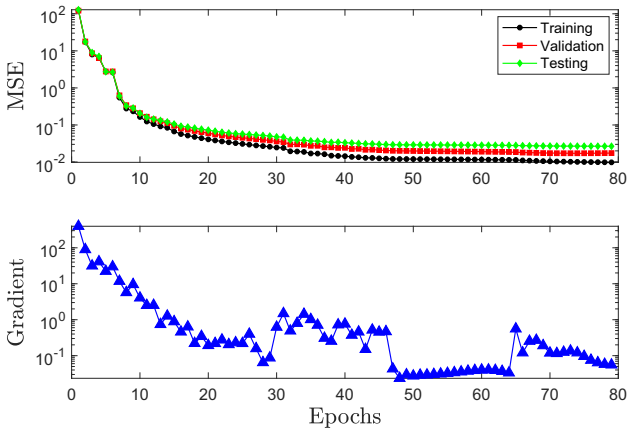

Fig. 5 Changes of the MSE and the gradient with the epochs for a model with the fourth architecture

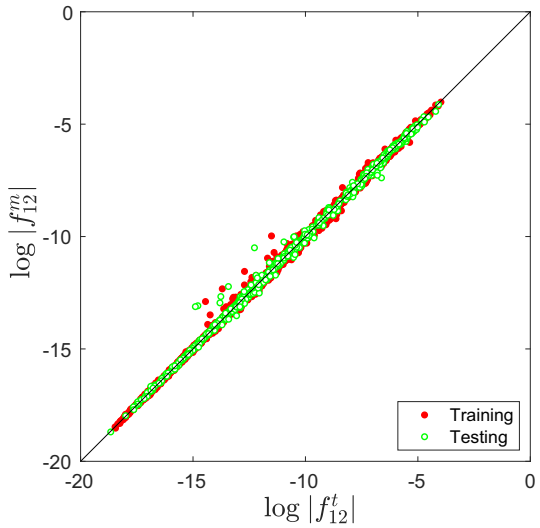

Fig. 6 Regression of the training (filled circles) and the testing (empty circles) data

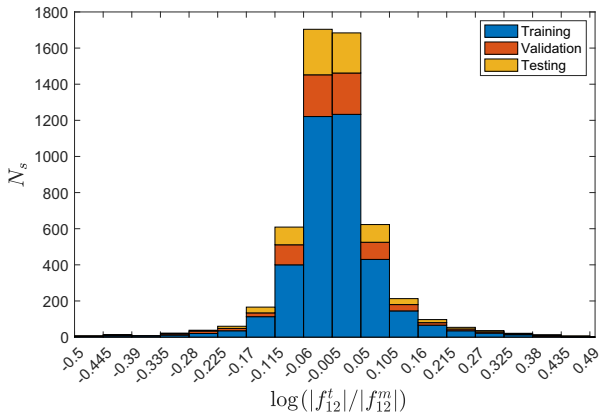

Fig. 7 Bar chart for the error $\varepsilon_a \equiv \log|f_{12}^t| - \log|f_{12}^m| \equiv \log(|f_{12}^t/f_{12}^m|)$ for the training (bottom bars), validation (middle bars) and testing (top bars) data. The height of a bar represents the number of samples N_s in the bin

the same architecture. The training, testing, and validation MSEs for these 32 models are calculated. The median values are plotted with the symbols in Fig. 4 for all 8 architectures. Indicated by the error bars are the maximum and minimum values, which show the ranges of the data. As the number of data points is small (32 for each architecture), we have chosen to plot the medians, which is considered more representative of the overall behaviour of small data set. For the same reason, we have used the ranges to represent the dispersion of the data. The result for architecture 1 shows that, with 30 neurons, the validation MSE can be kept under 5% in all runs. This observation has been the basis to choose 30 as the total number of the neurons. Figure 4 shows clearly that the MSE can be reduced by using multiple hidden layers. Architecture 4, which has two hidden layers with 15 neurons in each, has the best performance. Further increasing the number of hidden layers beyond two appears to degrade the performance. The median testing MSE for architecture 4 is 1.51%, while that of architecture 8 is 2.01%, which is 33% higher. Based on these results, architecture 4 has been chosen to obtain all the results to be presented below. As an illustration, Fig. 5 shows how the performance and the gradient of the model improves by the training.

The next results provide fuller description of the errors. Figure 6 shows the regression of the training and the testing data, where superscripts t and m are used to denote the true and modelled values, respectively. Excellent regression is obtained for both datasets. The coefficient of determination R is more than 99% in both cases.

The bar chart in Fig. 7 shows the distribution of the absolute error for $\log|f_{12}|$ defined by $\varepsilon_a \equiv \log|f_{12}^t| - \log|f_{12}^m|$. Given that the values for $\log|f_{12}|$ approximately range between -20 and -5 (c.f. Fig. 6), the error on a large majority data points is very small.

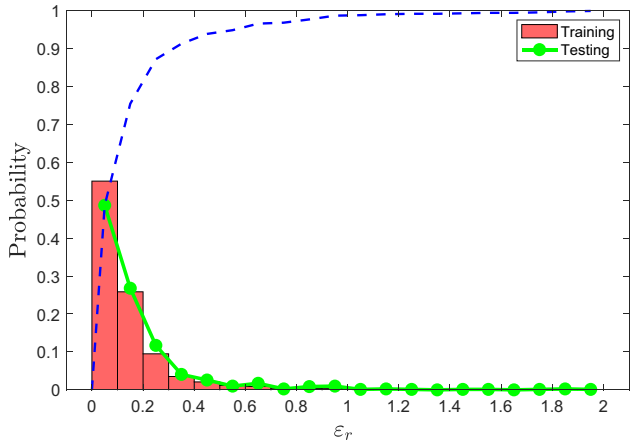

Fig. 8 Probability distribution for the relative error $\varepsilon_r \equiv |(|f_{12}^t| - |f_{12}^m|)|/|f_{12}^m|$ for the training (bars) and the testing (filled circles) sets. The y-axis is the probability for ε_r in each bin. The blue line is the cumulative probability for ε_r for the testing data

The error in $\log|f_{12}|$ is small enough that the magnitude $|f_{12}|$ itself is also accurately modelled. Plotted in Fig. 8 is the relative error for $|f_{12}|$, which is defined as $\varepsilon_r \equiv ||f_{12}^t| - |f_{12}^m||/|f_{12}^m|$ and is related to ε_a by $\varepsilon_r = |10^{\varepsilon_a} - 1|$. There are a small number of extraneous samples where ε_r can be more than 100%, but, as expected, the majority of the samples have small errors. The cumulative probability distribution, plotted with the dashed line, shows that more than 90% samples in the testing set have relative errors smaller than 30%, and about 85% smaller than 20%. Comparing the errors on the training set and the testing set, it is only slightly more probable to observe larger errors on the testing data, which demonstrates that the model generalizes well.

The above results show that the FFNN model (with 15 neurons on each of the two hidden layers) can provide an accurate model for not only $\log|f_{12}|$ but also $|f_{12}|$. In terms of the training cost, typically less than 100 epochs are needed to achieve the accuracy depicted in Figs. 6, 7, and 8, which takes less than one minute to compute on a modern laptop.

4.2 Support-vector machine model for sgn f_{12}

In its most simple form, the SVM [35] is an algorithm that finds the optimal straight line that separates two clusters of points on a plane, hence classifying the points into two classes. In order to introduce some necessary basic concepts, its formulation is briefly explained here. It is assumed that a set of N points $\mathbf{x}^{(j)} = (x_1^{(j)}, x_2^{(j)}) \in R^2$ $(j = 1, 2, ..., N)$ is given as the training set. The points are divided into two groups, the positive and negative classes. For simplicity, the data are assumed to be separable, i.e., it is assumed that a straight line can be found to separate the two classes, and that the classifications are known and labelled by 1 and -1, respectively. The classification of point $\mathbf{x}^{(j)}$ is recorded by $y^{(j)} \in \{-1, 1\}$.

The optimal separating line is defined as the line that separates the two groups whilst leaving the largest margins on both sides of the line. Let the equation for the line be $f(\mathbf{x}) \equiv \mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$, where $\mathbf{w} \in R^2$ and $b \in R$. It can be shown that the best separating line is the solution of the following optimization problem [35]: find $\mathbf{w}$ and b that minimize the objective function $||\mathbf{w}||^2$ such that for all data points $(\mathbf{x}^{(j)}, y^{(j)})$,

$$y^{(j)} f(\mathbf{x}^{(j)}) \geq 1. \tag{8}$$

The optimal objective function maximizes the margins on the two sides of the line. The constraints ensure that the points are found on the correct sides of the separating line. Let $\hat{\mathbf{w}}$ and $\hat{b}$ be the optimal solutions, which define the optimal SVM, and let $\hat{f}(\mathbf{x}) \equiv \hat{\mathbf{w}}^{\mathrm{T}}\mathbf{x} + \hat{b}$. The classification of a data point $\mathbf{x}$ is given by its *label* sgn $\hat{f}(\mathbf{x})$.

In the more general cases where the data are not separable by a straight line, two modifications to the above method have been introduced. Firstly, one may allow a small number of points to be mis-classified, a practice termed using soft margin. Mathematically, this method amounts to replacing Eq. (8) by

$$y^{(j)} f(\mathbf{x}^{(j)}) \geq 1 - \xi^{(j)} \tag{9}$$

for $\xi^{(j)} \geq 0$. Meanwhile, a penalty term is added to the objective function to limit the magnitude of $\xi^{(j)}$. For this investigation, the term takes the form of $C\sum_{j=1}^{N}\xi^{(j)}$ with C being a parameter called the box constraint. A larger C introduces larger penalty, hence reduces $\xi^{(j)}$'s magnitudes, which means fewer mis-classifications are allowed. No mis-classification is allowed when $C \to \infty$. Secondly, one may use a nonlinear curve to separate the data. The idea is implemented by using a function $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{h}(\mathbf{x}) + b$ as the separating curve, where $\mathbf{h}(\mathbf{x})$ is a non-linear function that transforms the separating line to a nonlinear separating curve.

The above exposition of the SVM model has been based on the so-called primal formulation. Used in current investigation is the dual formulation of the model, because the dual problem is convex and guaranteed to converge to the global minimum [37]. The two formulations are equivalent as their optimal solutions provide the same classification. Letting $\alpha^{(j)}$ be the dual variable corresponding to the constraint given in Eq. (9), the dual problem can be written as

$$\min \quad \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}\alpha^{(j)}\alpha^{(k)}y^{(j)}y^{(k)}G(\mathbf{x}^{(j)}, \mathbf{x}^{(k)}) - \sum_{j=1}^{N}\alpha^{(j)} \tag{10}$$

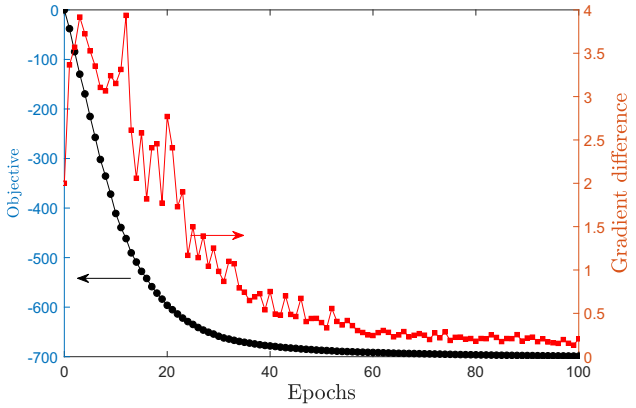

Fig. 9 Objective function and the gradient difference as functions of the training epochs in a typical training session. Only the first 100 epochs are shown

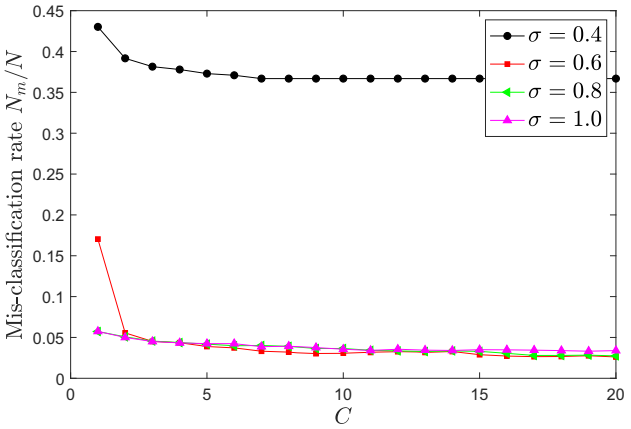

Fig. 10 Mis-classification rate for different kernel scales σ and box constraints C

subject to the constraints

$$\sum_{j=1}^{N} \alpha^{(j)} y^{(j)} = 0, \quad 0 \le \alpha^{(j)} \le C. \tag{11}$$

The bivariate function G in Eq. (10) is the kernel function corresponding to the nonlinear function $\mathbf{h}(\mathbf{x})$. For this investigation, a Gaussian kernel is used where $G(\mathbf{x}^{(j)}, \mathbf{x}^{(k)}) = \exp(-||\mathbf{x}^{(j)} - \mathbf{x}^{(k)}||^2/\sigma^2)$ with σ being the kernel scale.

The dual problem is solved with the sequential minimal optimization (SMO) algorithm. The SMO algorithm is an iterative descent algorithm, in which the convergence is monitored by the gradient of the dual objective function with respect to $\alpha^{(j)}$. Specifically, the difference between the gradient components corresponding to the maximal upper and lower violation of the feasibility conditions is monitored [37]. The iteration is deemed converged when this difference is smaller than a tolerance δ. Once the optimal solution for the dual problem is found, $\hat{\mathbf{w}}$ and $\hat{b}$ can then be found from $\alpha^{(j)}$ using the Karush–Kuhn–Tucker conditions, which are then used to classify new data points. For more details of the SVM methods and the algorithms, see Hastie et al. [35], Cristianini and Shawe-Taylor [37].

The SVM model is applied to classify the data for sgn f_{12}, using the MATLAB function fitcsvm which implements the above algorithms. In this case, $\mathbf{x}$ is a five dimensional vector, i.e., $\mathbf{x} = (D, R_{E1}, R_{E2}, p_a, f)^{\mathrm{T}}$ and the label y is sgn f_{12}. The number of data points is $N = 5400$. The data are standardized when they are fed into the training algorithm. Specifically, the mean is removed from the data which are then rescaled by the standard deviation. Figure 9 illustrates the decay of the gradient difference as well as the objective function in Eq. (10) in a typical training process.

The adjustable parameters in the model are the box constraint C, the kernel scale σ, and the tolerance δ. Tests with $\delta = 10^{-3}$ and 10^{-4} show essentially the same results. Therefore $\delta = 10^{-4}$ has been used. When the separation boundary has a complicated shape, a small σ is required to model the fine features of the boundary. However a too small σ may limit the ability of the model to capture the large scale features in the distribution of the data. The choice of C depends on the accuracy of the data for f_{12}. If the data for f_{12} likely contain large errors, it is not meaningful to insist perfect classification. The physical model being used here (Eq. (3)) has been derived with a few simplifying assumptions. Therefore, it is appropriate to use a large C to limit mis-classification, although it is not necessary to achieve zero mis-classification.

Based on the above discussion, the mis-classification rate has been calculated for several different kernel scales σ and box constraints C. A point $\mathbf{x}^{(j)}$ is mis-classified if $y^{(j)}\hat{f}(\mathbf{x}^{(j)}) < 1$ (c.f. Eq. (9)). The mis-classification rate is the ratio of the number of mis-classified points, N_m, to the total number N. The results are plotted in Fig. 10. The mis-classification rate reaches an approximate plateau quickly when C increases. The results for $\sigma = 0.4$ are clearly much worse, whereas small mis-classification is found for other σ's when C is sufficiently large. These observations demonstrate the robustness of the classification scheme as long as σ is not too small. The smallest mis-classification of 2.61% is found at $(\sigma, C) = (0.6, 20)$, which is considered sufficiently small. Therefore $C = 20$ and $\sigma = 0.6$ are used in what follows.

The robustness of the model is assessed in Fig. 11 using k-fold cross-validation [35]. In k-fold cross-validation, the dataset is divided randomly into k equal sets, and k models (called the partitioned models) are trained. The ith ($i = 1, 2, \ldots, k$) dataset is called the ith test fold, whereas the other $k-1$ sets form the ith training fold. The ith partitioned SVM model is trained on the ith training fold and evaluated on the ith test fold. The overall assessment is based on performance indices averaged over the k partitioned models.

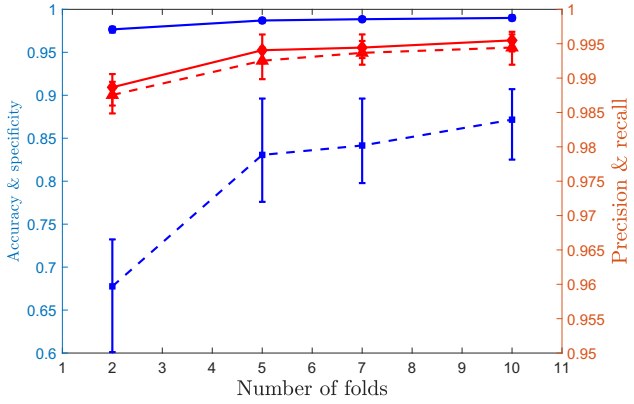

Fig. 11 Left axis: the median accuracy (solid line with circles) and the median specificity (dashed line with squares). Right axis: the median precision (solid line with diamonds) and the median recall (dashed line with triangles). Found in an ensemble of 32 runs. The error bars show the maximum and minimum

The most commonly used performance indices are the accuracy, the precision, the recall and the specificity. For each predictor, the model response could either be positive or negative; in either case, it could be either true or false. As a result, the response falls in one of four categories: a positive response could be a true positive (TP) or, coming erroneously from a predictor in the negative class, a false positive (FP), whereas a negative response could be true negative (TN) or false negative (FN). Let N_{TP}, N_{TN}, N_{FP} and N_{FN} be the numbers of TP, TN, FP, and FN, respectively. The accuracy is defined as

$$\frac{N_{TP}+N_{TN}}{N}, \tag{12}$$

which simply gives the percentage of correct predictions in both classes. The precision, recall and specificity are, respectively, defined as

$$\frac{N_{TP}}{N_{TP}+N_{FP}}, \frac{N_{TP}}{N_{TP}+N_{FN}}, \frac{N_{TN}}{N_{TN}+N_{FP}}. \tag{13}$$

The precision tells us the probability of a positive response being correct; the recall is the probability of the positives being correctly identified as positive, while the specificity gives the probability of the negatives being correctly identified as negative [35].

Due to the randomness in data partition, the indices averaged over the k partitioned models may fluctuate if the cross-validation is conducted multiple times. Therefore, we repeat the validation 32 times to find the medians and ranges of the indices. The results are plotted in Fig. 11. It is clear that the accuracy, the precision, and the recall of the models are consistently high (more than 98% for all of them), although they drop slightly when fewer folds are used, while the ranges increase slightly (the error bars for the accuracy are too narrow to see on the figure). With fewer folds, the number of data points in each fold, hence in the training set, is smaller. Thus the performance of the partitioned models are expected to somewhat deteriorate. The results for accuracy show that more than 98% data points, with either negative or positive f_{12}, are classified correctly. Meanwhile, the result for the precision shows that there is a 98% chance that f_{12} is indeed positive when the model predicts so, and the result for the recall shows that there is a $1-98\% = 2\%$ chance that f_{12} is not predicted to be positive when it is actually positive.

Figure 11 shows that the median specificity and its range display behaviours similar to those of the accuracy, the precision and the recall, but there is stronger dependence on the number of folds, and its range is wider and the median is smaller. The median recall increases with the fold number and reaches approximately 85%, which implies there is a $1-85\% = 15\%$ chance that f_{12} is not predicted to be negative when it is actually negative. The specificity is relatively low compared with the other indices, but this observation does not necessarily reflect poorer performance regarding the samples with negative f_{12}. Rather, this behaviour is due to the fact that there is only about 3.4% data points on which $f_{12} < 0$, thus mis-classification has an outsized impact.

The results presented in this subsection show that, with kernel scale $\sigma = 0.6$, box constraint $C = 20$, and tolerance $\delta = 10^{-4}$, the trained SVM classifier can effectively model the distribution of the signs of f_{12}.

5 Efficiency and accuracy of the combined model

A prediction for the force f_{12} can be made by combining the two ML models obtained in Sect. 4. In this section the predictions f_{12}^m made this way are compared with the true predictions f_{12}^t found from solving Eqs. (3) and (5). 1024 samples for $(D, R_{E1}, R_{E2}, p_a, f)^{\mathrm{T}}$ are randomly chosen, with each component falling in the range covered by the dataset used to train the ML models. Note that the samples are not necessarily in the dataset. Excluding the samples where $R_{E1} < R_{E2}$, 640 samples are used for this test, and 640 pairs of values (f_{12}^t, f_{12}^m) are obtained.

Excellent correlation is found between f_{12}^t and f_{12}^m, with the correlation coefficient being 0.98. The scatter plot for the data is shown in Fig. 12. The figure confirms the good correlation while, in the meantime, shows that the difference tends to increase when the magnitude of the force increases.

The histogram for the relative error $\varepsilon_r^f \equiv |f_{12}^t - f_{12}^m|/|f_{12}^t|$ is shown in Fig. 13. The error distribution has a broader spread than those in Fig. 8, i.e., those for the training data and the testing data. This behaviour is not unexpected as the ML models are being applied to a new dataset here. Nevertheless, more than 45% samples have less than 20% errors,

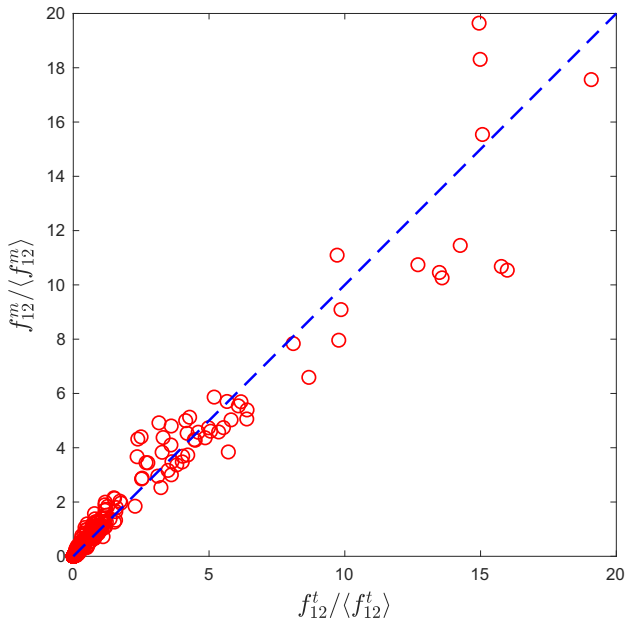

Fig. 12 Scatter plot for $(f^t_{12}/\langle f^t_{12}\rangle, f^m_{12}/\langle f^m_{12}\rangle)$, where $\langle\cdot\rangle$ denotes averaging over the dataset

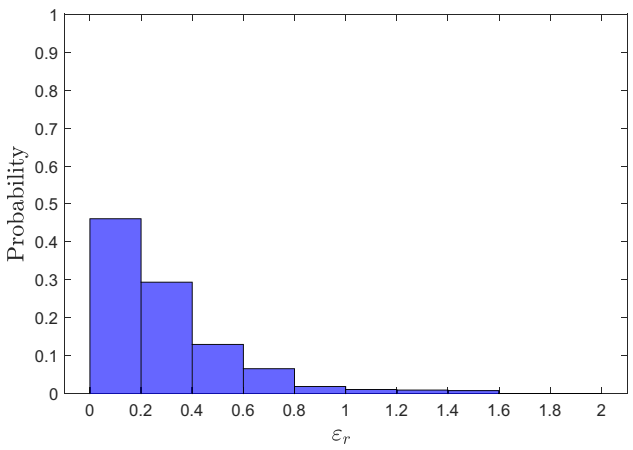

Fig. 13 Probability distribution for the relative error $\varepsilon^f_r = |f^t_{12} - f^m_{12}|/|f^t_{12}|$

and for more than 75% samples the error is less than 40%. Therefore, the results are still quite satisfactory. Obviously, the performance of the ML models can be improved if the training data set can be expanded and refined.

Finally, the ML models are extremely efficient compared with direct numerical integration. Tested on a laptop, it takes 0.8 h to obtain the 640 values for f^t_{12}, whereas it takes only 0.04 second for the ML models to find corresponding f^m_{12} when the 640 values are retrieved in one batch through a single call to the machine learning models. We now consider a situation where only one f^m_{12} value can be calculated in each functional call to the machine learning model. In this case obviously the total computational time increases with

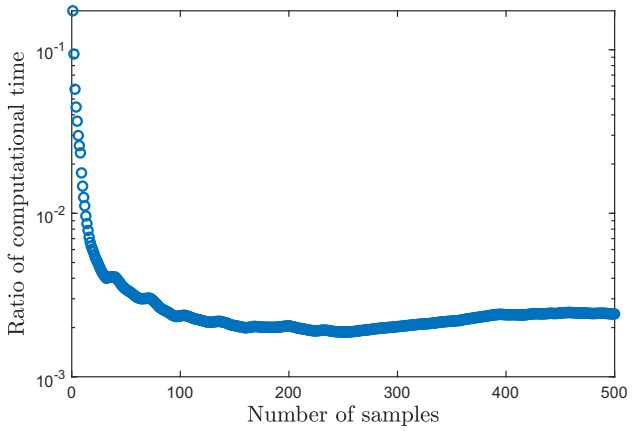

Fig. 14 Ratio of computational time as a function of the number of samples

the number of samples. The ratio between the total computational time needed to calculate f^m_{12} from the model and the time needed to calculated f^t_{12} by solving Eq. (3) is plotted in Fig. 14. The figure shows that the ratio can be reduced to approximately 0.2% for large numbers of samples. Therefore, a very significant saving is still achieved even in this very inefficient way of applying the machine learning model.

6 Conclusions

Machine learning models for the secondary Bjerknes force as a function for several parameters have been developed in a two bubble system. Because the force varies drastically with the parameters, the magnitude and the sign of the force have to be modelled separately, which results in a composite model consisting of a feed-forward neural network for (the logarithm of) the former and a support-vector machine for the latter.

Numerical tests demonstrate the feasibility of using machine learning to tackle this problem. Practical methods for choosing the suitable architecture and hyperparameters for the models are proposed. Accurate machine models are obtained, which are shown to be very efficient compared with direct numerical integration of the bubble evolution equations.

The results demonstrate that machine learning is a viable method in modelling the secondary Bjerknes force. The models developed here have the potential to enhance the future simulations of bubble clusters. Obviously, the model can be further refined and expanded, by, e.g., using a larger dataset covering a wider range of parameters. A particularly interesting question is if the methodology is still valid for higher pressure amplitudes. Increasing the pressure would potentially increase the occurrence of negative (repulsive) secondary Bjerknes force, which tends to reduce the accuracy

of model predictions. However, there is an opposing effect. When the pressure is increased, the domain in the parametric space where the force is negative could become more regular, with smoother boundaries. This behaviour is observed in, e.g., Mettin et al. [6]. This change in principle could improve the performance of the models. Therefore, how increased pressure may change the performance of the models remains an interesting question. Finally, machine learning clearly is equally applicable when a more sophisticate physical model is used to describe bubble oscillations, although it is not obvious that the currently chosen architectures are still sufficient when the dataset grows larger. These interesting topics will be explored in our future investigations.

Acknowledgements The authors gratefully acknowledge the support provided by the Guangzhou Science (Technology) Research Project (Grant 201704030010) and the special fund project of Science and Technology Innovation Strategy of Guangdong Province (Grant PDJH2020B0185).

References

1. Brennen, C.E.: Cavitation and Bubble Dynamics. Cambridge University Press, New York (2014)
2. Barbat, T., Ashgriz, N., Liu, C.S.: Dynamics of two interacting bubbles in an acoustic field. J. Fluid Mech. **389**, 137–168 (1999)
3. Doinikov, A.A., Zavtrak, S.T.: On the mutual interaction of two gas bubbles in a sound field. Phys. Fluids **7**, 1923–1930 (1995)
4. Harkin, A., Kaper, T.J., Nadim, A.: Coupled pulsation and translation of two gas bubbles in a liquid. J Fluid Mech. **445**, 377–411 (2001)
5. Jiao, J., He, Y., Kentish, S.E., et al.: Experimental and theoretical analysis of secondary Bjerknes forces between two bubbles in a standing wave. Ultrasonics **58**, 35–42 (2015)
6. Mettin, R., Akhatov, I., Parlitz, U., et al.: Bjerknes forces between small cavitation bubbles in a strong acoustic field. Phys. Rev. E **56**, 2924–2931 (1997)
7. Pelekasis, N.A., Tsamopoulos, J.A.: Bjerknes forces between two bubbles. Part 1. Response to a step change in pressure. J. Fluid Mech. **254**, 467–499 (1993)
8. Pelekasis, N.A., Tsamopoulos, J.A.: Bjerknes forces between two bubbles. Part 2. Response to an oscillatory pressure field. J. Fluid Mech. **254**, 501–527 (1993)
9. Pelekasis, N.A., Gaki, A., Doinikov, A.A., et al.: Secondary Bjerknes forces between two bubbles and the phenomenon of acoustic streamers. J. Fluid Mech. **500**, 313–347 (2004)
10. Yoshida, K., Fujikawa, T., Watanabe, Y.: Experimental investigation on reversal of secondary Bjerknes force between two bubbles in ultrasonic standing wave. J. Acoust. Soc. Am. **130**, 135–144 (2011)
11. Zhang, Y., Zhang, Y., Li, S.: The secondary Bjerknes force between two gas bubbles under dual-frequency acoustic excitation. Ultrason. Sonochem. **29**, 129–145 (2016)
12. Doinikov, A.A., Bouakaz, A.: Theoretical model for coupled radial and translational motion of two bubbles at arbitrary separation distances. Phys. Rev. E **92**, 043001 (2015)
13. Doinikov, A.A., Bouakaz, A.: Microstreaming generated by two acoustically induced gas bubbles. J. Fluid Mech. **796**, 318–339 (2016)
14. Pandey, V.: Asymmetricity and sign reversal of secondary Bjerknes force from strong nonlinear coupling in cavitation bubble pairs. Phys. Rev. E **99**, 042209 (2019)
15. Fan, Z., Chen, D., Deng, C.X.: Characterization of the dynamic activities of a population of microbubbles driven by pulsed ultrasound exposure in sonoporation. Ultrasound Med. Biol. **40**, 1260–1272 (2014)
16. Lazarus, C., Pouliopoulos, A.N., Tinguely, M., et al.: Clustering dynamics of microbubbles exposed to low-pressure 1-mhz ultrasound. J. Acoust. Soc. Am. **142**, 3135–3146 (2017)
17. Ma, X., Xing, T., Huang, B., et al.: Combined experimental and theoretical investigation of the gas bubble motion in an acoustic field. Ultrason. Sonochem. **40**, 480–487 (2018)
18. Haghi, H., Sojahrood, A.J., Kolios, M.C.: Collective nonlinear behavior of interacting polydisperse microbubble clusters. Ultrason. Sonochem. **58**, 104708 (2019)
19. Keller, J.B., Miksis, M.: Bubble oscillations of large amplitude. J. Acoust. Soc. Am. **68**, 628–633 (1980)
20. Ida, M.: Multibubble cavitation inception. Phys. Fluids **21**, 113302 (2009)
21. Lanoy, M., Derec, C., Tourin, A., et al.: Manipulating bubbles with secondary Bjerknes forces. Appl. Phys. Lett. **107**, 214101 (2015)
22. Ahmed, D., Lu, M., Nourhani, A., et al.: Selectively manipulable acoustic-powered microswimmers. Sci. Rep. **5**, 9744 (2015)
23. Bermudez-Aguirre, D., Mobbs, T., Barbosa-Canovas, G.V.: Ultrasound applications in food processing. In: Feng, H., Barbosa-Cánovas, G.V., Weiss, J. (eds.) Ultrasound Technologies for Food and Bioprocessing, vol. 65. Springer (2011)
24. Brujan, E.A.: Cavitation in Non-Newtonian Fluids. Springer, Berlin (2011)
25. Eskin, G.I., Eskin, D.G.: Ultrasonic Treatment of Light Alloy Metals. CRC Press, Boca Raton (2015)
26. Roberts, W.W.: Development and translation of histotripsy: current status and future directions. Curr. Opin. Urol. **24**, 104–110 (2014)
27. Mettin, R.: Bubble structures in acoustic cavitation. In: Doinikov, A. (ed.) Bubble and Particle Dynamics in Acoustic Fields: Modern Trends and Applications, pp. 1–36. Research Signpost, Kerala (2005)
28. Mettin, R., Luther, S., Ohl, C.D., et al.: Acoustic cavitation structures and simulations by a particle model. Ultrason. Sonochem. **6**, 25–29 (1999)
29. Parlitze, U., Mettin, R., Luther, S., et al.: Spatio-temporal dynamics of acoustic cavitation bubble clouds. Philos. Trans. R. Soc. Lond. A **357**, 313–334 (1999)
30. Maeda, K., Colonius, T.: Bubble cloud dynamics in an ultrasound field. J. Fluid Mech. **862**, 1105–1134 (2019)
31. Silver, D., Schrittwieser, J., Simonyan, K., et al.: Mastering the game of go without human knowledge. Nature **550**, 354 (2017)

32. Louisnard, O., Gonzalez-Garcia, J.: Acoustic cavitation. In: Feng, H., Barbosa-Cánovas, G.V., Weiss, J., eds. Ultrasound Technologies for Food and Bioprocessing, vo. 13. Springer (2011)
33. Hagan, M.T., Demuth, H.B., Beale, M.H., et al.: Neural Network Desgin. (2nd edn.) Martin Hagan (2014)
34. Goodfellow, I., Bengio, Y., Courville, A., et al.: Deep Learning. MIT Press, Cambridge (2017)
35. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York (2009)
36. Ciaburro, G.: MATLAB for Machine Learning. Packt Publishing, Birmingham (2017)
37. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, New York (2014)